Lotus. software

IBM

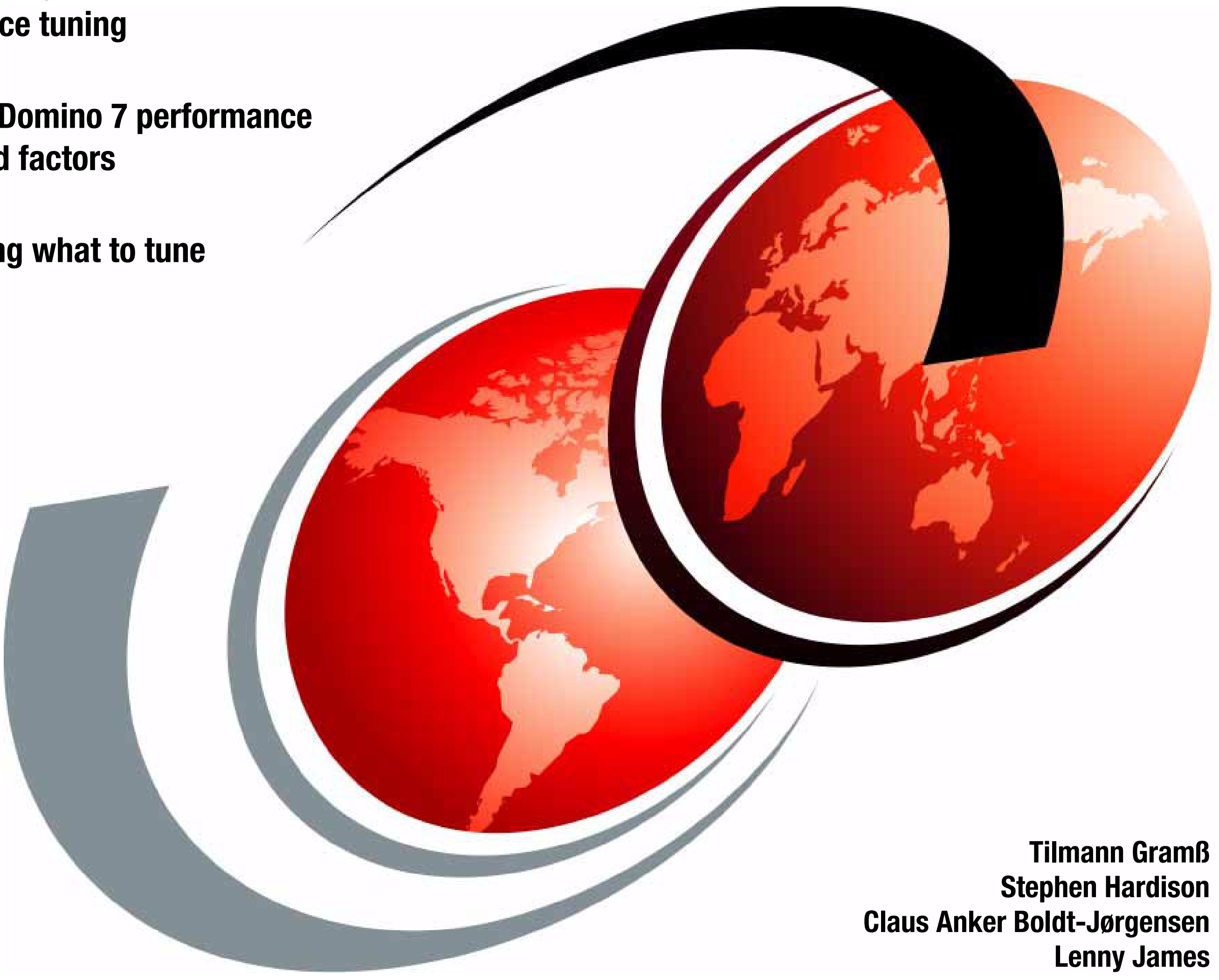
Domino 7 Performance Tuning

Best Practices to Get the Most Out of Your Domino Infrastructure

Understanding the approach to performance tuning

Analyzing Domino 7 performance and related factors

Determining what to tune and why



Tilmann Gramß
Stephen Hardison
Claus Anker Boldt-Jørgensen
Lenny James

Redpaper

ibm.com/redbooks

International Technical Support Organization

Domino 7 Performance Tuning: Best Practices to Get the Most Out of Your Domino Infrastructure

September 2006

Note: Before using this information and the product it supports, read the information in "Notices" on page vii.

First Edition (September 2006)

This edition applies primarily to IBM Lotus Domino Release 7. Many of the concepts and best practices discussed here also apply to earlier versions of Domino, including release 5.x and Domino 6.x.

Contents

Notices

Trademarks

Preface

Performance tuning, or optimization, is the process of modifying a system's configuration in order to improve its efficiency at meeting *specified functional goals*. The systems in question can be a single application, one Domino® server, a collection of servers participating in a common activity, such as mail routing or replication, or an entire Domino infrastructure.

Performance tuning is goal-focused, and involves balancing the use of available system resources in order to meet those goals. This usually involves some sort of trade off, where one or more resources are optimized at the expense of others. A classic example of this is caching, which attempts to improve I/O performance by buffering data to memory. The memory allocated to the cache service is unavailable for use by any other programs on the platform, but if overall system performance is improved, the benefits outweigh the cost.

This IBM® Redpaper discusses a best practices approach to Performance Tuning in Domino 7. It addresses both how to approach the science of performance tuning in a structured, logical manner, while also providing an in-depth discussion of specific configuration parameters to tune in specific situations.

After reading this Redpaper, this should further assist Domino Administrators in tuning Domino systems to more effectively:

- ► Improve user or application response time
- ► Improve system stability
- ► Reduce transaction times
- ► Balance the use of system resources

The team that wrote this Redpaper

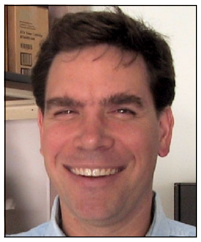
This Redpaper was produced by a team of specialists from around the world working at the International Technical Support Organization, Cambridge, MA, USA Center.

Tilmann Gramß is a Technical Solution Architect at IBM's Strategic Outsourcing - Integrated Technology Delivery based in Frankfurt, Germany. He has been working with the e-mail and collaboration engineering team since 1998. He has a broad experience with the design, setup, and integration as well as the operating of complex and global messaging infrastructures. Tilmann is also a subject matter expert within the IBM European service line organization and has years of experience with e-mail security, Lotus® Notes® and Domino security assessments, and Internet mail encryption.

Stephen Hardison is a Consulting I/T Specialist with IBM Software Services for Lotus based in Texas. He has 25 years of IT experience, with over 12 years experience implementing and managing Lotus Domino based solutions. He has also worked extensively implementing WebSphere® Portal and portal-based collaborative and business applications. He attended the University of Houston and the University of Texas at Austin majoring in Computer Science.

Claus Anker Boldt-Jøergensen has been working on Lotus Notes / Domino since Version 2. He has been working as a consultant for the past 10 years in the Domino arena (since Version 3). He started with a Lotus business partner and later for Lotus Professional Services, which now is a part of the IBM Software Group Services organization (ISSL). He has been employed within IBM (Lotus) for the past seven years, where he has pursued a career path as a Certified Advisory IT Architect. His current focus in the ISSL organization is to assist customers in Consolidation, Migration, and Performance optimization of Domino infrastructures.

Lenny James is an Advisory Software Engineer in IBM Austin, acting as a technical lead for the Domino Core Server support team. In his eight years of supporting customers' Domino environments, he has specialized in replication, adminp, security, HTTP, SNMP, and server crash and performance troubleshooting. Lenny also has a strong OS background, including Solaris™, Linux®, and Microsoft® Windows®. In addition to an MCSE, he holds multiple IBM Lotus Notes and Domino certifications in both System Administration and Application Development.



John Bergland is a project leader at the International Technical Support Organization, Cambridge Center. He manages projects that produce Redbooks™ about IBM/Lotus Software products. Before joining the ITSO in 2003, John worked as an Advisory IT Specialist with IBM Software Services for Lotus (ISSL), specializing in Notes and Domino messaging and collaborative solutions.

Thanks to the following people for their contributions to this project:

- **Rob A. Gearhart** - Advisory Software Engineer, IBM Software Group, WPLC, IBM, Austin, TX.
- **Peter Mierswa** - STSM, Domino Server Team, IBM Software Group, WPLC, IBM, Westford, MA
- **Jane L. Wilson -** Technical Information Architect, IBM Software Group, WPLC, IBM, Westford, MA

Become a published author

Join us for a two- to six-week residency program! Help write an IBM Redbook dealing with specific products or solutions, while getting hands-on experience with leading-edge technologies. You'll team with IBM technical professionals, Business Partners, and clients.

Your efforts will help increase product acceptance and client satisfaction. As a bonus, you'll develop a network of contacts in IBM development labs, and increase your productivity and marketability.

Find out more about the residency program, browse the residency index, and apply online at:

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

We want our papers to be as helpful as possible. Send us your comments about this Redpaper or other Redbooks in one of the following ways:

- Use the online **Contact us** review redbook form found at:

  ibm.com/redbooks

- Send your comments in an e-mail to:

  redbook@us.ibm.com

- Mail your comments to:

  IBM Corporation, International Technical Support Organization
  Dept. HYTD Mail Station P099
  2455 South Road
  Poughkeepsie, NY 12601-5400

1

Introduction to performance tuning in Domino

This chapter defines Domino performance tuning at a high level. If also defines the scope of this Redpaper, and identifies the audience. The following topics are covered:

- Introduction to performance tuning
- Goals of this Redpaper
- Target audience

1.1 Introduction to performance tuning

Domino is an extremely versatile server platform. Businesses can install Domino servers on several different vendors' operating systems, take advantage of different network connectivity options, and configure Domino servers to perform specialized functions, such as mail routing, Web hosting, or LDAP directory access. Given this level of flexibility, Domino administrators must know how to tune their systems to make the best use of available resources.

Performance tuning, or optimization, is the process of modifying a system's configuration in order to improve its efficiency at meeting *specified functional goals*. The systems in question can be a single application, one Domino server, a collection of servers participating in a common activity, such as mail routing or replication, or an entire Domino infrastructure.

Performance tuning is goal-focused, and involves balancing the use of available system resources in order to meet those goals. This usually involves some sort of trade off, where one or more resources are optimized at the expense of others. A classic example of this is caching, which attempts to improve I/O performance by buffering data to memory. The memory allocated to the cache service is unavailable for use by any other programs on the platform, but if overall system performance is improved, the benefits outweigh the cost.

Key point: Performance tuning is goal-focused and usually involves some sort of *trade off*, where one or more resources are optimized at the expense of others.

The goals for tuning Domino systems is similar to those of most other systems:

- Improve user or application response time
- Improve system stability
- Reduce transaction times
- Balance the use of system resources

1.1.1 Tuning and Systems Management

From the Systems Management perspective, performance tuning touches several areas, as shown in Figure 1-1 on page 3:

- Configuration management: Software packages (such as Domino and Domino-based applications) must be configured to make optimal use of system resources.
- Performance management: System level statistics and events provide the primary data for performance tuning analysis and recommendations.
- Fault management: Bottlenecks identified during troubleshooting are the primary indicators that a system needs to be optimized.
- Account management: Analysis of usage statistics can provide useful load and response time data from users' perspectives.

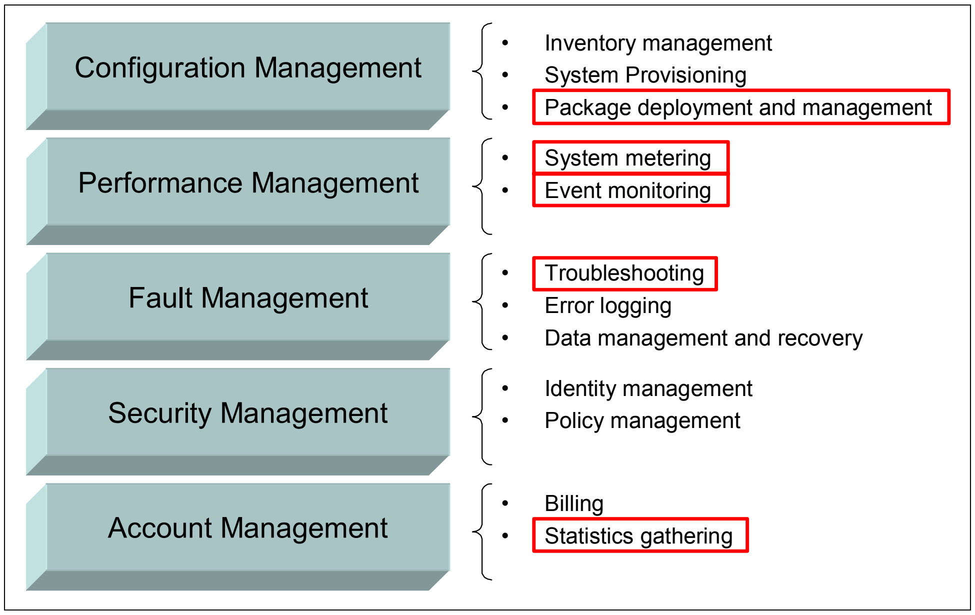


Figure 1-1 Performance tuning as an element of Systems Management

1.1.2 Tuning process

Performance tuning is a cyclical activity which has four distinct parts, as shown in Figure 1-2:

- Identify: Reviewing the symptoms of the performance issue, and determining potential root cause(s)
- Analyze: Weighing the options available to correct the performance issue and the potential impacts or costs of those options
- Modify: Making changes to the system's configuration to attempt to resolve the issue
- Measure: Reviewing the effectiveness of the change

Once a particular cycle is complete, additional issues are identified and addressed in an incremental manner.

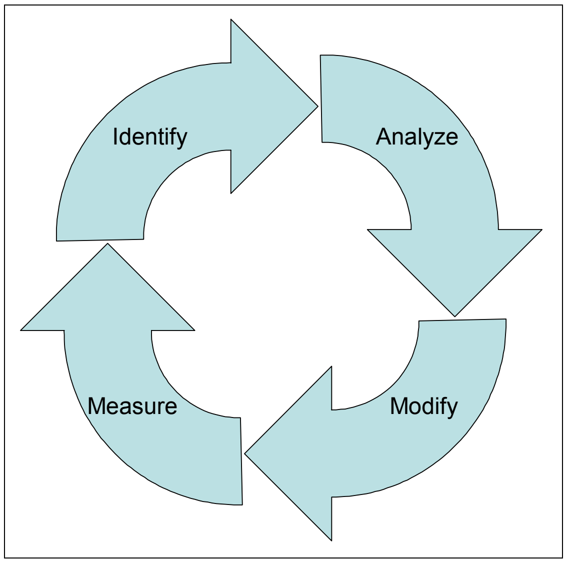


Figure 1-2 Tuning cycle

Important: *Make only one tuning change at a time*, and measure the results of that change before attempting another. Additionally, prior to making changes to the system, verify that you have a consistent and accurate data baseline to measure against.

Making simultaneous tuning changes can lead to the following situations:

- When measuring results, it may not be possible to determine which modification led to the performance improvements.
- Likewise, if a system becomes unstable after multiple modifications have been made, it may not be possible to determine which change caused the instability.
- Some changes may conflict with one another, leading to no measurable change in system performance.
- Applying (or removing) multiple changes can be a cumbersome process and prone to human error.

1.1.3 Tuning activities

The tuning process is most frequently associated with trying to resolve some sort of problem on a Domino system. However, tuning can also be initiated through routine monitoring, or the review and analysis of historical data. Tuning activities are usually classified as one of the following:

- Bottleneck elimination
- Proactive tuning
- Trend analysis

Bottleneck elimination

Bottleneck elimination, or reactive tuning, is an activity undertaken after a problem is identified with the performance of the system. It is intended to identify and correct situations where particular resources are overused or in contention with one another.

Bottlenecks may be the result of:

- Ineffective or incomplete testing of a system before it was put into production
- Changes in the underlying infrastructure
- Changes in usage patterns
- Differing performance profiles of Domino versions

Resolving bottlenecks involves looking at the system from end-to-end, and identifying all the elements that could be impacting performance. Depending to the impact, this could include:

- User workstation configuration and connectivity
- Network topology, bandwidth, and router configuration
- Domino system topology, routing and replication settings, cluster configuration, and database distribution
- Hardware, disk I/O, physical memory allocation, and processor configuration
- Operating system parameters, virtual memory configuration, and file systems
- Third-party monitoring, virus scanning, and backup tools
- Domino server settings, task configuration, caching, and transaction logging

- Domino-based Notes or web application architecture, database design and settings, and usage patterns
- External systems and data sources used by Domino at runtime

Proactive tuning

Proactive tuning, or monitoring, is the regular review of performance metrics to identify short-term changes in a system's behavior or resource utilization.

Routine monitoring does not usually result in tuning changes to a system. However, monitoring can often help administrators identify issues that may impact performance or lead to bottlenecks in the future. Corrective action can then be planned and taken before the issue manifests itself and impacts users or service availability.

Care must be taken when making tuning changes based on monitoring. It is possible to implement changes that actually introduce performance problems where there were none before. Experimenting with configuration changes, or "tweaking" system settings without a specific goal or a clear understanding of the potential impact, should be avoided.

Trend analysis

Trend analysis is a logical extension of proactive tuning, and involves examining a system's long term behavior. While it is most commonly associated with capacity planning, trend analysis can be useful in identifying changes in usage patterns, and can provide insight into how long running Domino servers make use of resources over time. Trend analysis can also help administrators project the potential performance impact of an upcoming system change, such as a major application launch or the implementation of a major architectural change.

Trend analysis depends on the collection and retention of large amounts of system performance data, both from within Domino, and from infrastructure systems, such as hardware, operating system, and network statistics. Since this information comes from multiple data sources, administrators must develop a process for capturing and storing these statistics. Leveraging Domino's native statistics and event collection facilities can provide administrators with a starting point for managing this data, but these facilities need to be augmented to capture external data.

Analysis of historical data can be accomplished with either common workplace tools, such as spreadsheets and graphs, or can make use of more versatile vendor tools, which provide additional capabilities, such as large volume data consolidation, statistical analysis, and scripting.

One element that is often overlooked when analyzing historical data are changes made to the system over time. This can include:

- Hardware changes: Adding memory, processors, and changing vendor platform.
- Software changes: Operating system upgrades or reconfigurations, and Domino upgrades.
- Re-tasking: Changing the functions or services provided by a system.
- Consolidation: Dramatically changing the workload of a system due to user or application redistribution.
- Tuning activities: Presumably, a particular change will lead in turn to a change in the system's performance profile.

Since these activities alter the meaning of historical data, the detailed record of when the changes took place must be kept in sync with statistical data pulled from Domino or infrastructure systems in order to provide the necessary context when performing the analysis.

1.1.4 Performance baselines

In order to understand any changes to Domino performance, and ultimately, the impacts as a result of Domino performance tuning, it is critical to first understand a baseline of performance, *prior to making any changes or modifications*.

Note: Chapter 3, "Managing Performance Data and establishing a baseline" on page 37 discusses how to manage and utilize historical data to establish a baseline.

Here is a list of the key considerations in regard to establishing a meaningful baseline from which you can measure against:

- You need clear understanding of system behavior before you tune.
- Establishing the baseline: This is established by monitoring and capturing / tracking performance characteristics a system under certain conditions.
- A baseline is based on system statistics gathered on Domino and infrastructure.
- Performance tuning/measurement requires *multiple* baselines for comparison (for example, little or no load, normal peak load, and during maintenance windows).
- Carefully compare before and after statistics to determine the effectiveness of a change.
- *M*ake sure the after statistics are representative (no special conditions, such as end of quarter processing or major organizational events).
- Development and maintenance of baseline data is an ongoing process; eventually, the newer statistics made after modifications and a return to a stable state become new baselines.

1.2 Goals of this Redpaper

Performance tuning is an ongoing, iterative process. If done correctly, tuning is likely to produce diminishing returns over time. The goal of optimization is a stable, well tuned system with proper monitoring. reporting, and maintenance processes in place to help make sure that the system continues to meet a business' functional and service level goals efficiently.

Performance tuning has the potential of touching every area of a Domino system's design and architecture. Instead of looking in detail at all of these aspects, such as application optimization, network tuning, or hardware configurations for specific vendor platforms, this paper looks at Domino specific tuning considerations. It attempts to summaries performance tuning best practices by looking at the following areas:

- Understanding Domino performance
  - Infrastructure considerations
  - Different types of Domino servers
  - Finding bottlenecks
  - Monitoring and reporting
- Identifying what to tune
  - External elements (such as operating systems, networks, and I/O subsystems)

  - Domino server
  - Domino architecture
- Special considerations
  - Domino partitioning
  - Domino cluster workload balancing
  - Transaction logging
  - Domino system sizing
- Managing historical data
  - The importance of historical data
  - Types of information and retention options
  - Using historical data

1.3 Target audience

This Redpaper is focused primarily on providing background information and best practices regarding Domino specific performance tuning, rather than providing a deep dive regarding external tuning (such as operating systems, networks, and I/O subsystems). It was written for professionals fulfilling the following roles:

- Domino system architects
- Domino system administrators

2

Understanding performance tuning and the Domino tools available

This chapter further defines the concept *Lotus Domino performance tuning* by addressing the following topics:

- Highlighting and differentiating between performance tuning issues directly related to the Domino server versus performance factors that are influenced by the larger IT infrastructure and *ultimately impact the overall Domino Architecture*.
- Discussing in detail the activities and Domino tools involved with monitoring and reporting, as this ultimately leads to an understanding of how to gather and analyze performance data.
- Finally, we show basic requirements to be fulfilled when identifying performance bottlenecks and how an appropriate approach to monitoring and reporting may help in this process.

2.1 Defining Domino performance within the IT infrastructure and the context of the Domino Architecture

When speaking about Lotus Domino performance tuning, you have to first look at the larger infrastructure landscape, namely considering *all* the components of the infrastructure within which the Domino architecture resides. You also need to understand and appreciate how components of the overall Domino Architecture interact and may impact performance.

Certainly, you will want to eventually focus on the individual Domino server(s) for primary tuning, but you must first have an appreciation for how all the infrastructure components work together and ultimately impact overall performance. For example, a high performance server has limited value and will be constrained by a poorly designed network infrastructure with limited bandwidth. Likewise, server performance would be restricted by an infrastructure having a poorly designed application topology.

In order to address issues specific to *Domino* performance tuning, we need to first differentiate between the following performance tuning aspects:

- Performance issues and performance tuning of the *larger, overall Domino architecture* and its management
- Performance issues and performance tuning within a *specific server instance*

Appreciating that there is a difference in the approach allows us to build a foundation of understanding about performance tuning for both the Domino Server and the larger Domino Architecture.

Note: The approach taken in this chapter is to focus on the broader infrastructure issues that apply to the larger Domino Architecture, and then work more specifically toward tuning a specific Domino Server instance.

2.1.1 Components of the Domino architecture, its management, and impact on performance

To cover the whole story of performance tuning, we have to extend our focus to those things happening outside one particular server instance: namely, we need to focus on the Domino Architecture as a whole and how this infrastructure is managed.

Most Domino performance tuning initiatives will only take effect within the network when Domino servers are communicating and collaborating with peers, thus, other Domino servers. Since Domino is not designed as a stand-alone application, individual server performance is only beneficial if the system peripherals are based on a efficient, high performance design and managed against this target. Figure 2-1 on page 11 illustrates how issues such as Topology Design, Capacity Management, System Management, and Application Design Management all factor into over all Domino architecture performance.

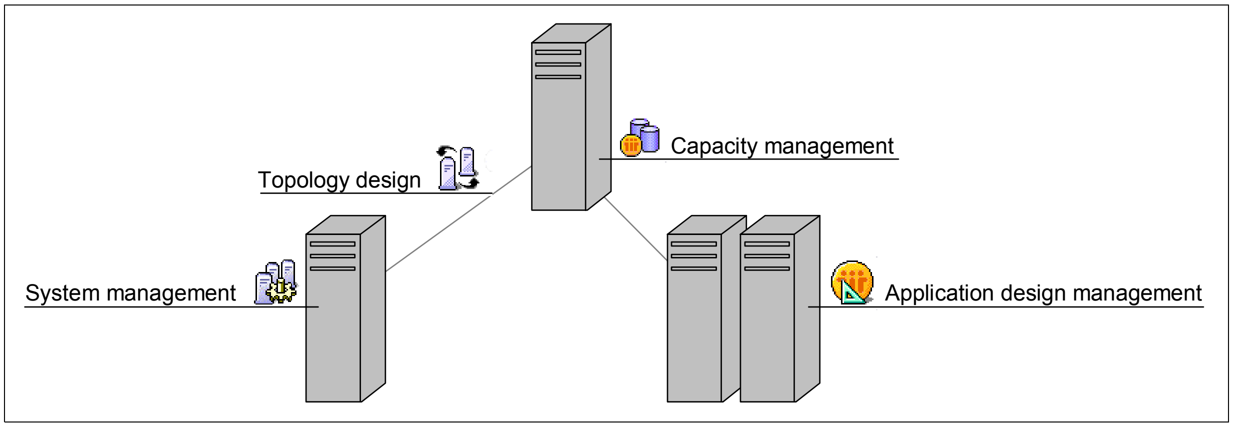


Figure 2-1 Domino performance tuning is also an architecture business

As each of these factors play a role in the overall Domino architecture performance, there is a high potential for performance improvement by changing and more efficiently managing the architecture. The next sections address each of these topics briefly.

Note: It is beyond the scope of this Redpaper to be able to address each of these influencing factors in detail. The key point here is that each of these factors can play a significant role in overall Domino related performance.

Topology design

The design of your Domino server topology may affect your overall infrastructure performance in several ways.

- By optimizing mail routing or replication lanes between servers and domains, or by building logical Domino Named Networks across the infrastructure, you can reduce server load and ineffective connections causing delays.
- By splitting server usage and load according to their roles (see 2.2, “Considerations for different types (roles) of Domino servers” on page 15), you can prevent conflicting performance issues.

Examples:

- A dedicated application server may have special performance tuning targets according to its hosted data, which may have a negative performance impact for a mail only server. By separating applications from mail files, and ultimately dedicating each Domino Server to a specific function, you are able to more effectively tune each server according to its role
- Central infrastructure servers, such as *hub servers*, with a significant e-mail routing load, can be more effectively tuned against this target if many of the complex and time intensive replication tasks are moved to another machine

Capacity management

Managing server capacity according to your infrastructure also improves the overall infrastructure performance.

Some examples are listed below:

- On a Domino mail server cluster, the average load of each server within the cluster is reduced by balancing the number of concurrent users between the cluster members. Instead of having one dedicated active home server for user mail files, you may wish to split the load and force half of the users to use the cluster partner as their primary home mail server.
- For Web servers in a cluster, load balancing the http port on a network layer improves access times and the application performance itself on each cluster partner.
- Exercising proper storage capacity management, including utilizing monitoring facilities on Domino servers, prevents server performance reduction or crashes caused by missing storage resources. You can learn more about these tools in 2.4, “Domino toolset for monitoring and reporting” on page 22.

System management

The term *system management* includes managing tasks that are mandatory for the overall health of your Domino infrastructure. Some of these tasks include:

- System monitoring
- Virus scanning
- Backup management
- Managing transaction logging
- Utilizing other system management tools from the IBM Tivoli® family

System Management includes Domino related system management tasks (monitoring Domino server tasks and managing transaction logging) as well as managing the other third-party software components running per server instance, such as virus scanning and backup management. Since each of these tasks demand systems resources, they will impact the performance of the overall system. Therefore the configuration and schedule of these tasks must be adapted to the servers requirements (out of peak times), reduced to a minimum, and, where possible, monitored to avoid redundancies.

Managing Domino application development

The flexibility and programmability of Lotus Notes / Domino enables companies to build their own business specific applications and workflows. The efficiency of the design of Domino applications - even a single Domino application - can have an immense impact on the server’s performance. It is important to implement respective design policies and guidelines to ensure:

1. Performance committing application development
2. Quality assurance procedures
3. Performance test cycles prior to application deployment

Attention: Specific applications designed without any focus on performance may have a negative impact on:

- The application itself
- The hosting Domino server
- The complete Domino infrastructure

Important: While it is beyond the scope of this Redpaper to discuss efficient design of Domino Applications, we strongly encourage you to refer to the Redpaper *Lotus Domino 7 Application Development*, REDP-4102, found at `http://www.redbooks.ibm.com/abstracts/redp4102.html`, as well as articles on the Lotus Developer Domain at `http://www.lotus.com/ldd`.

In Chapter 5, "Understanding what to tune" on page 49, you will find more details about Domino architecture related performance tuning, as well as an in-depth analysis of specific variables to tune for a specific Domino Server instance.

More specifically, 5.1, "Performance tuning outside of Domino" on page 50 covers basic discussions regarding performance tuning outside of Domino, including the following topics:

- Network performance
- Platform architecture and system performance
- Operating system performance

2.1.2 Performance issues and performance tuning within a server instance

In this section, we now begin to narrow our approach and focus on the most common elements where performance is a key player, namely within a specific instance of the Domino server.

Important: When discussing a *server instance* within this section, we are referring to this as one particular machine, including common hardware components, a network interface, an operating system, and at least one Notes / Domino application installed on the Lotus Domino server.

Although we are now focused on the specific Domino Server, we still need to appreciate that each Domino server is influenced by much more than simply the configuration settings and specific notes.ini parameters. Each Domino server runs on a specific hardware platform, is controlled by a specific operating system, and will also be impacted by a network interface. It is the grouping of all these factors together, and their tuning, which ultimately impacts the Domino server. As shown in Figure 2-2 on page 14, Domino server performance is impacted by an *entire framework* of factors, including:

- Network interfaces
- Hardware components
- Operating system
- Applications that reside on the server

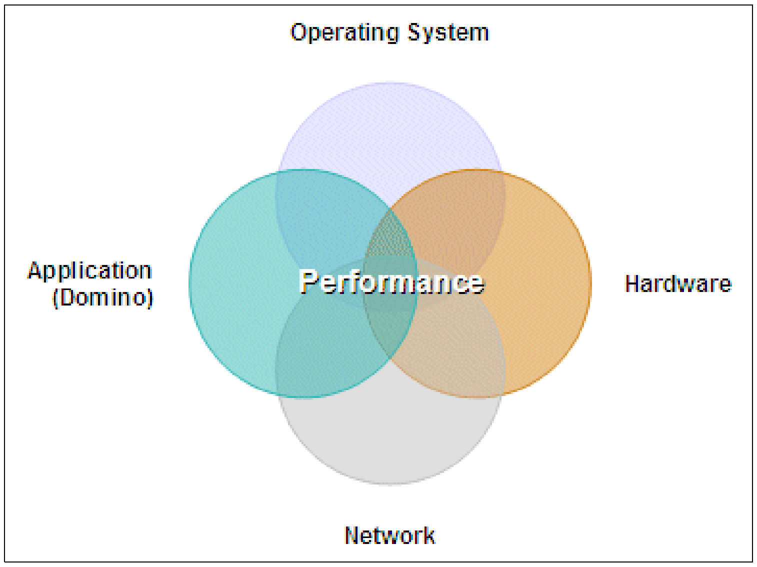


Figure 2-2 Server framework as one performing unit

Dependencies between the Domino server and other components of a server instance

Tuning your Domino server includes analyzing and tuning your system hardware and software for optimal performance of all major components, including memory, CPUs, disk drives, and the network.

Note: The intent of this section is to discuss and identify dependencies between the Domino server and other components of a server instance. Chapter 5, "Understanding what to tune" on page 49 will provide more in-depth technical detail about what and how to tune specific server settings.

The following examples illustrate dependencies between the Domino server and other components of a server instance:

- Processors affect the Indexer speed, Replicator speed, number of maximum possible database transactions, and number of add-ins that can run in parallel.
- Disk access rates and configurations affect the speed of database and view opening and of opening and navigating a collection (view index).
- Memory affects the maximum possible simultaneous Notes client connections (sessions), size of caches, and server add-in task performance (because of less paging to disk).

Therefore, prior tuning the Domino server as the application of a server instance itself, it is important to review and verify the stability and scaling of the server's fundamentals.

Terminology review

For clarification around the discussion on server performance throughout this Redpaper, it is helpful to review the following key terms. Below is a list of terms and definitions that will be used within the context of this Redpaper.

Real memory	Physical RAM installed in the system.
Page file	Physical disk space used to supplement real memory.
Swap file	Same as page file.
Virtual memory	Combination of real memory and page/swap.

Paging/Swapping	The process where data is moved from page file into real memory for processing and then back to disk again.
Thrashing	A condition where the operating system is busier swapping memory than processing CPU requests.
Semaphore	Varies the controls access to a particular resource.

2.1.3 Approaching Domino Server specific performance tuning

Based on the optimal performance threshold for your Domino server, it is worth examining the Domino server specific areas and configurations that have any effect on its performance.

As mentioned before, you cannot simply tune a Lotus Domino server by activating dedicated configuration parameters in isolation in hopes that this makes the server run faster. Proper performance tuning is undertaken in an iterative approach, and involves careful, planned adjustment of different settings and configurations designed to meet specific goals for your individual server. For example, a Domino server running as a Web server will have a different tuning objective than a server specifically designated for hosting user e-mail files.

5.2, “Domino specific performance tuning” on page 54 outlines detailed performance considerations and settings that may have a positive effect on you server performance.

The following topics are covered in depth in this chapter:

- Server tasks
- Database performance
- Minimizing logging activity
- Scheduling utilities
- Impact of Third-party applications
- Tuning user sessions

2.2 Considerations for different types (roles) of Domino servers

In 2.1.1, “Components of the Domino architecture, its management, and impact on performance” on page 10, we covered the Lotus Domino performance tuning in conjunction with the Domino topology design and recommended separate servers according to a special *role* within the infrastructure.

By defining roles for your servers, you determine what functions they perform and thus what tasks they need to execute. For example, if you specify that a Domino server will primarily support Domino Web Access clients, your Domino server takes on a Web server role.

We describe a number of Domino server roles in the following sections. Remember that these are not necessarily hard and fast specifications, just common server roles that many customers implement in their Domino environments. Defining the types of users each of your Domino servers will support will also make it easier to identify possible bottle necks and later to tune your servers efficiently.

All Domino servers will benefit from having balanced performance across their major subsystems: CPU, disk, and network. However, some servers, when performing a specific role, place heavier demand on the CPU. This is indicated in the descriptions that follow.

2.2.1 Mail servers

Mail servers are used in the Domino network to store and route mail messages among users, servers, and applications. Dedicating one or more servers to these responsibilities gives you the following benefits:

- Easier administration of mailboxes and users. This allows improved auditing and simplifies installation procedures.
- Reduced load on the server by minimizing the number of databases that need to be replicated with other servers and by decreasing the amount of database indexing performed.
- Reduction in the amount of mail traffic (by careful organization of users per mail server). Mail traffic delivered to users on the same server will not be routed across the network. This can also allow for immediate mail delivery by eliminating the need to create mail delivery schedules.
- Establishment of a better scalable model for expansion as usage and storage needs increase by providing you with the ability to define more specific requirements, such as response time.
- Improved mail routing performance by adding multiple mail.box databases and tuning of mail router and local delivery threads.
- Improved I/O performance by distribution of mail files across multiple disk channels, controllers, or both.

2.2.2 Database (or application) servers

Database servers are used to store the data for Domino applications. Customized applications, such as help desk databases or support tracking systems, for example, are typically implemented on dedicated servers. In sophisticated Domino-based applications, interaction between numerous databases is quite common. Application servers usually place heavier demands than usual on the CPU.

Distributing interdependent databases across multiple servers can place excessive loads on the servers, the network, or both, depending on the amount of interaction between applications. Dedicating your servers to application roles will provide the following benefits:

- Enables you to group applications by their usage. This will typically reduce replication needs and simplify administration of both the server and database security.
- Enables you to optimize server performance for application database usage without regard to mail usage.
- Enables you to define more specific requirements, such as response time, thereby establishing a better scalable model for expansion as usage and storage needs increase.
- Improves I/O performance by distribution of application databases across multiple disk controllers.

2.2.3 Hub servers

Hub servers are designed to be used as dedicated servers to route mail and replicate databases to other servers within the network. This typically results in hub servers being used only in larger Domino environments to minimize the performance impact on production servers. Domino hub servers are primarily implemented to facilitate communications between other servers, so users normally do not have access to these systems (or even know that they exist). Because of this, hub servers usually do not have large numbers of simultaneous connections. Therefore, you might assume that a hub server would not need to be based on a high-end server system to fulfill its role. However, this is typically not the case.

You must consider the fact that each hub server is required to support the storage of all common databases and be capable of replicating these databases to each additional server in a timely fashion. Each hub server must also be capable of routing mail and ensuring that messages are delivered within an acceptable time frame. To fulfill these requirements, each hub server should be designed with both powerful processing capacity and a high-performance disk subsystem.

We recommend that you use hub servers as the administration server for the Domino directory and applications. The Domino administration process uses this server to execute requests and updates to the directory and applications.

Providing database replication and mail routing to several other servers within the network places heavy loads on all server subsystems. Your target should be to design high-capacity scalable platforms. Areas you should consider include:

- The system should have a fast disk subsystem for replicating and archiving.
- Full text indexing should not be allowed on these systems, because index updates degrade performance, and users will not be accessing these databases.
- Remove access to the server for all normal users so that they do not degrade performance.
- Improve mail routing performance by using multiple mail.box databases.

2.2.4 Web servers

Web servers provide data requested by Web browsers. This information can be normal Web pages consisting of graphics files and text files on the server stored in HTML format, or in a Domino environment, pages built from Notes databases. Where practical, all information available from your Web site should be placed on a single server to reduce stresses on both the system and the network, as discussed in 2.2.2, “Database (or application) servers” on page 16.

Web servers usually carry a lot of network traffic and can have a significant CPU load if complex Web applications are implemented. Dedicating servers to Web tasks will provide the following benefits:

- Simplifies security for both intranet use and external Internet connections. Only databases to be viewed by Web browsers should be placed on the server, especially if this server is to be browsed by Internet clients.
- Enables you to optimize server performance for Web usage without regard to mail usage.
- Enables you to define more specific requirements, such as response time, thereby establishing a better scalable model for expansion as usage and storage needs increase.

2.3 Defining monitoring and reporting

The following section introduces specific tools and techniques for monitoring and reporting performance within Domino.

Before proceeding directly with the functionality of these tools, we want to first clarify and differentiate between these two terms - monitoring and reporting - to set a common understanding. While these terms are often used casually and interchangeably, we believe it is important to highlight the exact meaning and significance of each one.

As shown in Figure 2-3, monitoring and reporting are different terms, particularly when considered from a chronological perspective.

- Monitoring is related to the *present status* and health of an infrastructure,
- Reporting is a follow on action from monitoring, including the actions taken in analyzing, editing, and filing the data over a period of time according to the report requirements.
- Finally, monitoring and reporting do have different infrastructure prerequirements, target audiences, and objectives as part of system management activities.

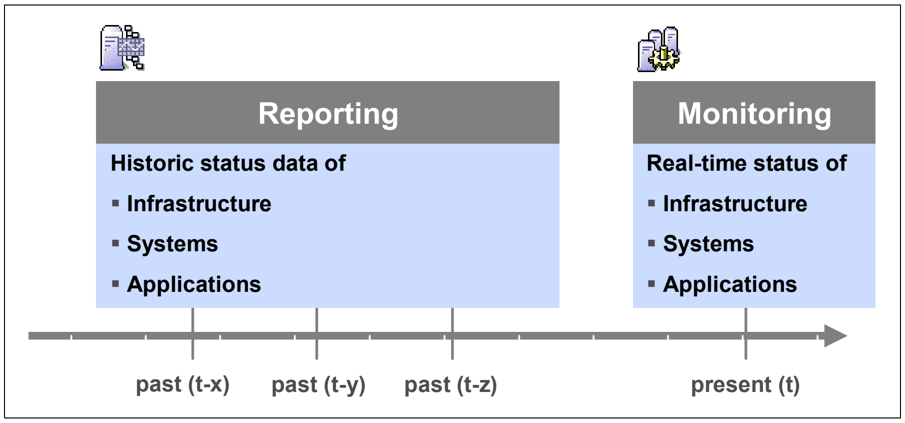


Figure 2-3 Differentiation between monitoring and reporting from a chronological perspective

A discussion of monitoring and reporting in the context of performance tuning will lead to some general statements:

- Monitoring and reporting objectives may conflict with performance tuning objectives, *if the respective tools and agents used for monitoring have a negative effect on the system performance*. This must be considered in planning and sizing monitoring and reporting implementations
- Not all results from monitoring and reporting are applicable when seeking performance bottle necks or identifying tuning opportunities.

In the following sections, we provide more detailed definitions for the terms *monitoring* and *reporting*. We then describe native Lotus Domino 7 tools and their respective functionality, focusing primarily on their usefulness within the context of performance tuning.

2.3.1 What is monitoring?

System monitoring covers all systems, tools, and processes ensuring a real-time view of the overall health and status of the infrastructure. The goal of monitoring is to analyze the situation of the current infrastructure.

Depending on the size of your enterprise, monitoring tools may require their own infrastructure, such as:

- Servers and systems exclusively used to monitor third systems
- Dedicated network infrastructure or protocols, such as SNMP
- Add-on system software to:
  - Monitor target systems and infrastructure
  - Manage and transmit alerts and messages, including correlation of events
  - Display these events as used in command centers, and so on
- Message interfaces to third systems, such as pager notification and SMS
- Interfaces to service management disciplines, such as ticketing systems
- Processes and guidelines to regularize event handling, communication, and problem resolution activities

Monitoring a target system can be done in different ways, depending on the system and infrastructure to monitor. Examples of different approaches include:

- Checking the system's health by monitoring the system's server console directly
- Requesting and providing the status provided by the monitored system on a remote console
- Polling health checks from the target machine by a third system, for example, by frequently trying to send network requests or checking tasks from outside
- Checking on-time system status by simulating common user activity, for example, by sending test e-mails

Once the expected behavior of the monitored system deviates from the norm, specific monitoring tools are able to notify the administrator in different ways. Examples are:

- Visual notification on the monitoring screen
- Audio-visual notification on dedicated displays
- E-mail notification
- Short Message Service (SMS)
- Pager notification

The quality and required extent of performing monitoring in your Lotus Domino infrastructure depends on the organization's needs and, subsequently, the infrastructure's availability requirements. Ultimately, this is determined through Service Level Agreements (SLAs) you need to fulfill. Additionally, the structure of your IT organization and the related administration model influence the need for different monitoring tools and processes.

System monitoring for specific services, such as messaging and collaboration, must cover all infrastructure components around Domino to ensure reliable service. This includes the basic availability of:

- Datacenter infrastructure
- Network
- Hardware
- Operating system
- Backup and recovery
- Storage

- Additional applications

Monitoring of just some system fragments (subsystems) will not guarantee any availability for the overall service and is not worth the effort. For example, the best Domino server monitoring efforts are of little or no value if there is no monitoring in place to cover the network connecting the servers and ensuring the communication between them.

Lotus Domino related monitoring activities may include checking the status of:

- Server availability
- File size on the disk system
- Database file sizes
- Domino server tasks
- Pending messages
- Data traffic (mails, replication, and so on)
- Availability Index (on clusters)
- Error or warning messages on the server console or log.nsf
- Further infrastructure specific monitoring requirements

A reliable monitoring strategy will help administrators to track down problems *before* the respective system gets out of control as part of *proactive tuning* (see 1.1.3, “Tuning activities” on page 4). It minimizes overall system downtime through the ability to react quickly on failures and taking the proper actions in time.

Additionally, the system’s reputation will rise due to higher availability and problem solving activities that forestall help desk calls.

Finally, monitoring may help administrators to understand why and where in the system persistent problems occur and simplify problem and performance bottleneck identification.

2.3.2 What is reporting?

Reporting is defined as the process of storing and editing monitoring data over time (refer to Figure 2-3 on page 18). Reporting is supported by several tools to create meaningful and relevant analyses of the data. Such reports can be used and adapted for the following purposes, including the respective relevant data:

- Trend analysis

  As described in 1.1.3, “Tuning activities” on page 4, trend analysis will help you to:

  - Foresee possible resource bottlenecks, such as storage issues
  - Outline the possible impact of future architectural changes or additional system loads, such as increasing the number of users
  - Develop future platform choices or other strategic decisions
  - Outline system usage peaks for load balancing planning
  - Identify *hidden* performance bottle necks based on chronological system evolution, such as the impacts of system caches or long term memory allocation as a result of application design errors

- Billing purposes

  Based on reporting numbers, such as mail file size, mail routing rate, and so on. this data may be used for billing purposes or segmenting the *Total Cost of Ownership (TCO)* on a per server or per user basis.
- Compliance checks

  Based on detailed user data included in the reporting, enterprises are able to perform any compliance checks on business or security guidelines.
- Service Level Agreement validation

  Providing evidence about server availability within a reporting period, system response, or message delivery times.
- Security vulnerability assessments

  Analyzing virus alerts, affected target accounts, or possible sources of infection.
- System landscape and consolidation analysis

  Using usage reports and mail traffic reports to determine possible consolidation targets because of poor usage of infrastructure parts.

Important: By collecting and storing data from the monitoring activities over a period of time, reporting is a a powerful instrument to not only manage performance, availability, and functionality of your IT infrastructure, but to ultimately analyze business and technical trends as well.

The usage of reporting allows you to measure the system results against any Service Level Agreements or Key Performance Indicators (KPIs).

The need and expansion of reporting and reporting tools depends on the enterprise and technical requirements, as well as the complexity of the system monitoring tools / system already in place. Of course, not all data relevant for system monitoring may be important for reporting purposes. This depends upon your organization’s specific goals and requirements.

Domino provides some native reporting facilities to provide the following data:

- Platform statistics
- Domino server availability
- Domino server utilization
- Database growth and size
- Response times
- Mail usage and peaks
- Mail tracking information
- Replication traffic
- Directory Catalog aggregation

These reports, or a relevant subset of these reports, will help you to meet reporting requirements and enable you to find answers on performance related requirements. More importantly, knowing the data will help keep your infrastructure under control.

Attention: The terms *monitoring* and *reporting* are not clearly separated within the Domino 7 Administrator Help, since monitoring and reporting tools provided in Domino are closely linked to each other in their technical implementation.

2.4 Domino toolset for monitoring and reporting

Having provided a clear definition and distinction between monitoring and reporting, we now look into specific tools provided by Domino.

Important: The tools discussed in the upcoming section will enable administrators to better understand and perform the primary types of tuning. We recommend reviewing 1.1, “Introduction to performance tuning” on page 2, as well as Chapter 5, “Understanding what to tune” on page 49 for details on how to execute the actual tuning.

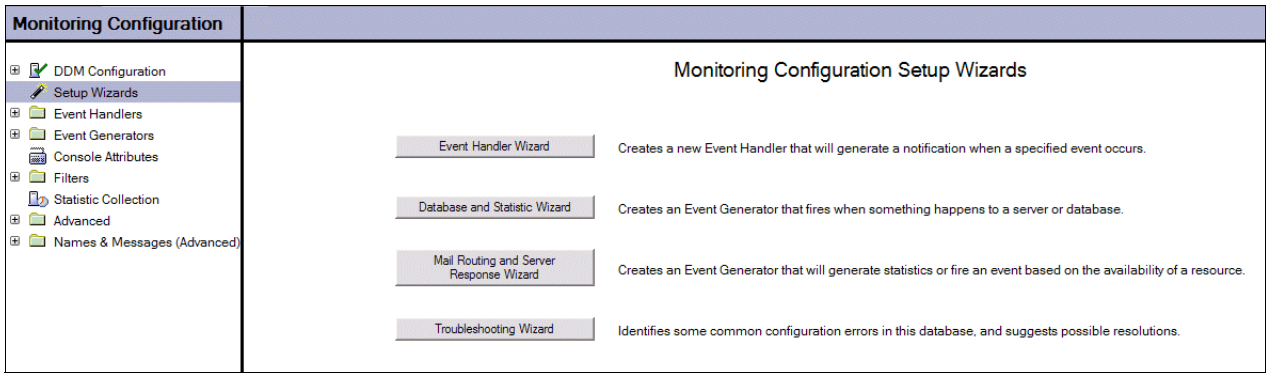
These types of tuning include:

- *Bottleneck elimination*
- *Proactive tuning*
- *Trend analysis*

The main components to work with when enabling Domino monitoring and reporting are as follows:

- Domino monitoring databases

  These databases store monitoring documents, information, and results. The Monitoring Configuration database (EVENTS4.NSF) stores the documents you use to set up monitoring. It also includes information about statistics, statistic thresholds, and event messages (see Figure 2-4).



Monitoring Configuration Setup Wizards

Creates a new Event Handler that will generate a notification when a specified event occurs.

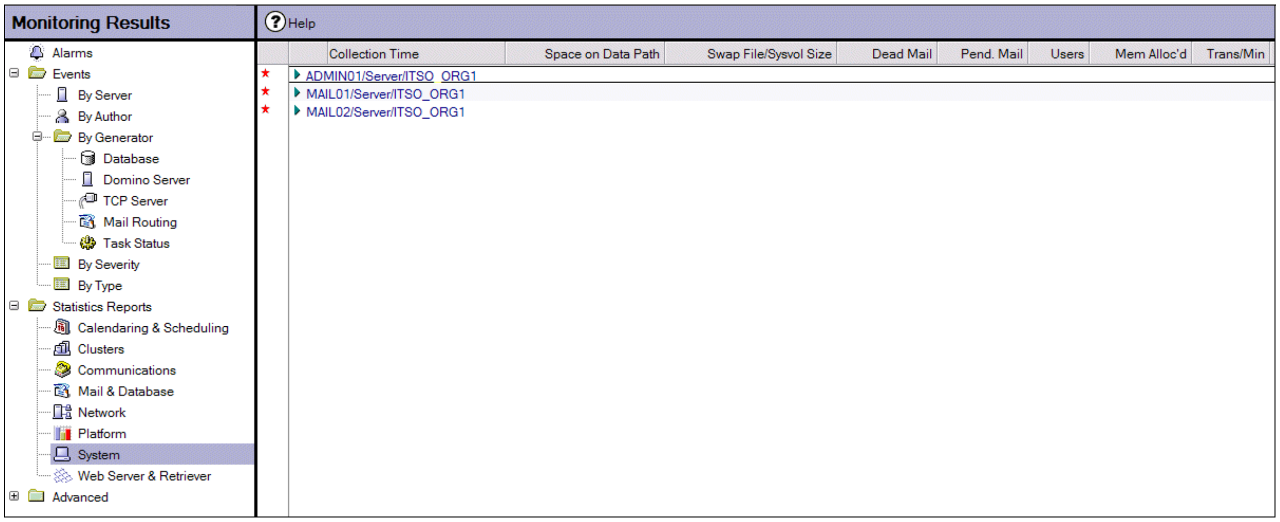
Creates an Event Generator that fires when something happens to a server or database.

Creates an Event Generator that will generate statistics or fire an event based on the availability of a resource.

Identifies some common configuration errors in this database, and suggests possible resolutions.

Figure 2-4 The Monitoring Configuration database

The Monitoring Results database (STATREP.NSF) stores the gathered statistics reports and will be configured to store information about logged events (see Figure 2-5 on page 23).



Monitoring Results

Help

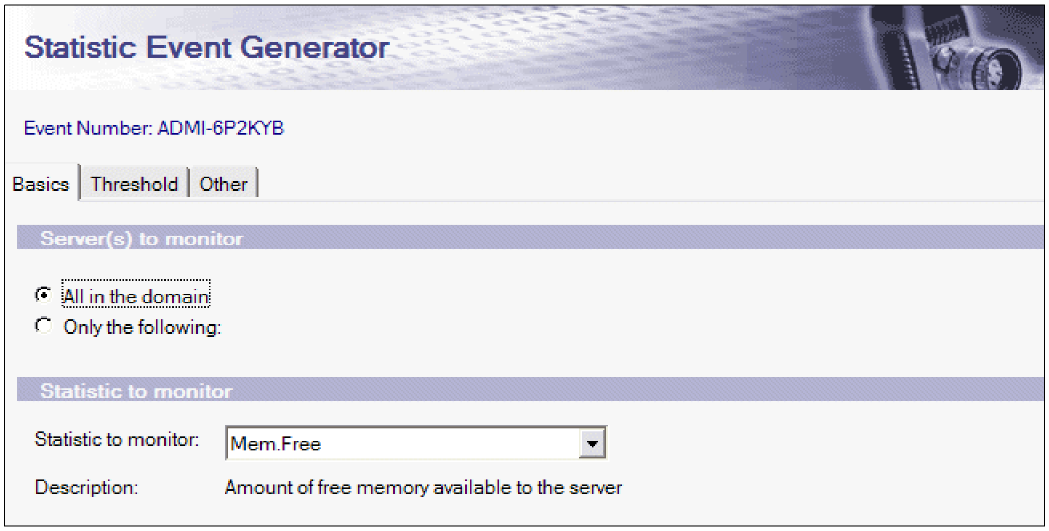
- Alarms
- Events
  - By Server
  - By Author
  - By Generator
    - Database
    - Domino Server
    - TCP Server
    - Mail Routing
    - Task Status
  - By Severity
  - By Type
- Statistics Reports
  - Calendaring & Scheduling
  - Clusters
  - Communications
  - Mail & Database
  - Network
  - Platform
  - System
  - Web Server & Retriever
- Advanced

Collection Time	Space on Data Path	Swap File/Sysvol Size	Dead Mail	Pend. Mail	Users	Mem Alloc'd	Trans/Min
ADMIN01/Server/ITSO_ORG1							
MAIL01/Server/ITSO_ORG1							
MAIL02/Server/ITSO_ORG1							

Figure 2-5 The Monitoring Results database

- ► Monitoring configuration documents

  These allow you to define and configure what constitutes an event, and how the event is handled (see Figure 2-6).



Statistic Event Generator

Event Number: ADMI-6P2KYB

Basics | Threshold | Other

Server(s) to monitor

(●) All in the domain
( ) Only the following:

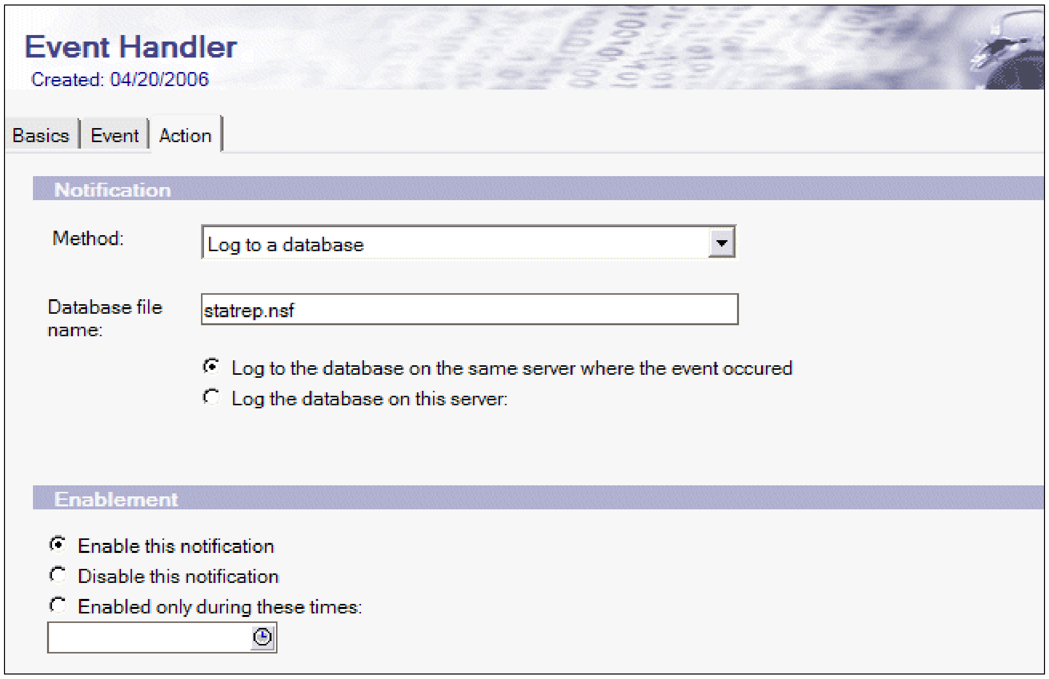
Statistic to monitor

Statistic to monitor: Mem.Free

Description: Amount of free memory available to the server

Figure 2-6 Statistic Event Generator example

Figure 2-7 illustrates the event handler, which allows you to specify exactly how an event gets handled and where the information is stored. In this case, all events are logged to the statrep.nsf database.


Event Handler
Created: 04/20/2006

Notification

Method: Log to a database

Database file name: statrep.nsf

(●) Log to the database on the same server where the event occured
( ) Log the database on this server:

Enablement

(●) Enable this notification
( ) Disable this notification
( ) Enabled only during these times:

Figure 2-7 Event Handler example

- Server tasks

  Server tasks collect and record information about the Domino system.

  – The Shutdown Monitor task ensures that Domino terminates when requested to do so.

  – The Process Monitor task applies only to Domino on Microsoft Windows platforms, and monitors the processes that should be running in the Domino server environment.

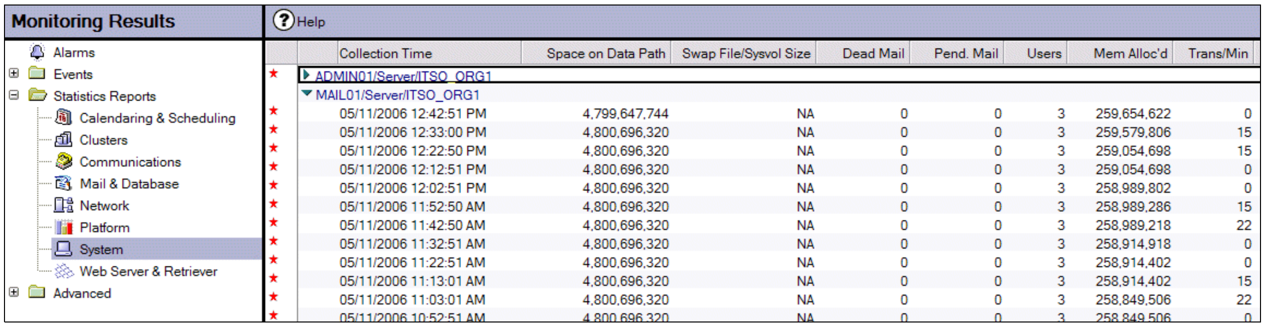
  – The Event Monitor task determines if an Event Handler has been configured for the event, and if so, routes the event to the specified person, database, or server-management program for processing.

  – The Statistic collector task gathers Domino server statistics and creates statistics reports in the Monitoring Results database (STATREP.NSF) or to another database you can specify.

  – The ISpy task executes TCP server and mail-routing event generators.

- Statistics

  Domino gathers statistics that show the status of processes currently running on the system. For example, the statistic Free space on drive C indicates the amount of free space available on drive C. You use these statistics along with the predetermined statistics thresholds to monitor both your Domino system and platform statistics (see Figure 2-8 on page 25).



Monitoring Results

- Alarms
- Events
- Statistics Reports
  - Calendaring & Scheduling
  - Clusters
  - Communications
  - Mail & Database
  - Network
  - Platform
  - System
  - Web Server & Retriever
- Advanced

Help

Collection Time	Space on Data Path	Swap File/Sysvol Size	Dead Mail	Pend. Mail	Users	Mem Alloc'd	Trans/Min
ADMIN01/Server/ITSO_ORG1							
MAIL01/Server/ITSO_ORG1							
05/11/2006 12:42:51 PM	4,799,647,744	NA	0	0	3	259,654,622	0
05/11/2006 12:33:00 PM	4,800,696,320	NA	0	0	3	259,579,806	15
05/11/2006 12:22:50 PM	4,800,696,320	NA	0	0	3	259,054,698	15
05/11/2006 12:12:51 PM	4,800,696,320	NA	0	0	3	259,054,698	0
05/11/2006 12:02:51 PM	4,800,696,320	NA	0	0	3	258,989,802	0
05/11/2006 11:52:50 AM	4,800,696,320	NA	0	0	3	258,989,286	15
05/11/2006 11:42:50 AM	4,800,696,320	NA	0	0	3	258,989,218	22
05/11/2006 11:32:51 AM	4,800,696,320	NA	0	0	3	258,914,918	0
05/11/2006 11:22:51 AM	4,800,696,320	NA	0	0	3	258,914,402	0
05/11/2006 11:13:01 AM	4,800,696,320	NA	0	0	3	258,914,402	15
05/11/2006 11:03:01 AM	4,800,696,320	NA	0	0	3	258,849,506	22
05/11/2006 10:52:51 AM	4,800,696,320	NA	0	0	3	258,849,506	0

Figure 2-8 Server statistic collection in the Monitoring Results database (STATREP.NSF)

- The Lotus Domino Administrator client

  The Administrator client does not only provide a front end to configure and administer your Domino server, it is also a powerful tool to perform monitoring activities, server health checks, and statistic collection. It is the first entry point for administrators to step into the different types of performance tuning.

The Monitoring Configuration database (EVENTS4.NSF) is used as administrator interface to set up monitoring and reporting within your Domino infrastructure. The main types of configurations are:

- Domino Domain Monitoring (DDM)
- Event Handlers
- Event Generators
- Console Attributes
- Filters
- Statistic collection

Domino Domain Monitoring (DDM)

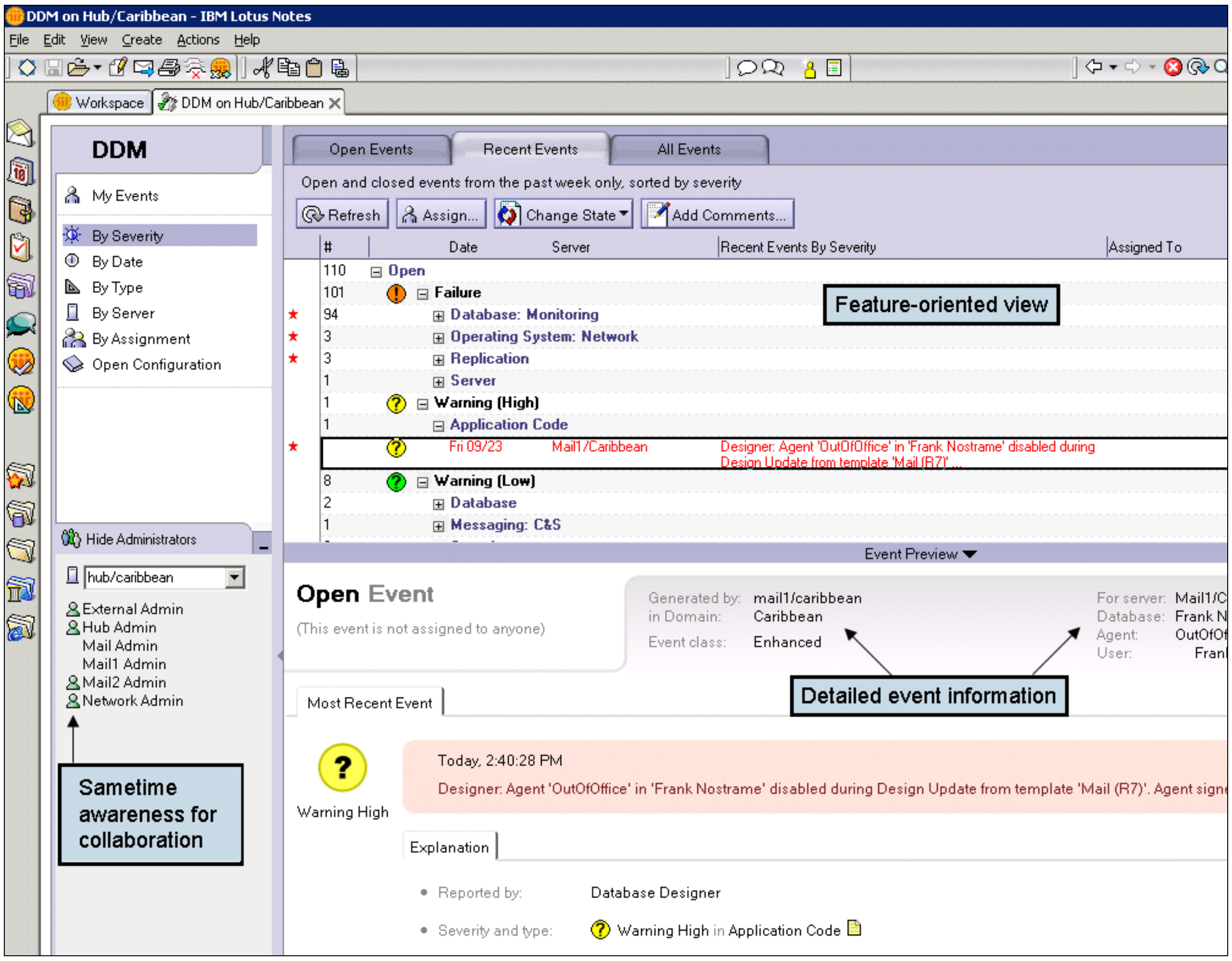
Domino Domain Monitoring, a new feature of Domino 7, provides a single feature-oriented view through which administrators can see the status of multiple servers and then use the information provided to quickly resolve problems as part of *bottleneck elimination* and *proactive tuning* activities.

With Domino Domain Monitoring, administrators can quickly locate and resolve issues before they cause more serious problems or outages.

The key features of Domino Domain Monitoring include:

- Top-down, feature-oriented view of the domain status
- Highly-configurable probes categorized by feature areas
- Probable cause and possible solution determination
- Automation of corrective actions
- Default settings for easy out-of-the-box setup
- Domino Domain Monitoring data aggregation with collection hierarchies

The Monitoring Configuration (Events4.nsf) database is used for all Domino Domain Monitoring configuration. The new Domino Domain Monitoring probes generate Event report documents that get consolidated and reported into the new Domino Domain Monitor database (DDM.nsf). Figure 2-9 illustrates the interface for Domino Domain Monitoring.



Open and closed events from the past week only, sorted by severity

#	Date	Server	Recent Events By Severity	Assigned To
110	Open			
101	Failure			
94	Database: Monitoring			
3	Operating System: Network			
3	Replication			
1	Server			
1	Warning (High)			
1	Application Code			
	Fri 09/23	Mail1/Caribbean	Designer: Agent 'OutOfOffice' in 'Frank Nostrame' disabled during Design Update from template 'Mail (R7)' ...	
8	Warning (Low)			
2	Database			
1	Messaging: C&S			

Open Event
(This event is not assigned to anyone)

Generated by: mail1/caribbean
in Domain: Caribbean
Event class: Enhanced

For server: Mail1/C
Database: Frank N
Agent: OutOfOf
User: Fran

Warning High

Today, 2:40:28 PM
Designer: Agent 'OutOfOffice' in 'Frank Nostrame' disabled during Design Update from template 'Mail (R7)'. Agent sign

- Reported by: Database Designer
- Severity and type: Warning High in Application Code

Figure 2-9 DDM.NSF: Domino Domain Monitoring user interface

Domino Domain Monitoring supports performance activities that are covered under the term *Proactive tuning* by improving the processes for bottleneck elimination.

The Redpaper *Lotus Domino Domain Monitoring*, REDP-4089, covers this new functionality in detail:

http://www.redbooks.ibm.com/redpapers/pdfs/redp4089.pdf

Domino events and event handlers

Every occurrence that happens on the Domino system is an event. Events signal both that the system is working smoothly, processing data, and performing tasks, and that the system is malfunctioning, perhaps by not processing data or performing required tasks.

Domino generates events continuously. Therefore, to monitor the Domino system efficiently to create reports used for performance tuning purposes, you must decide which events you want to know about.

For example, the event Replicating files with servername occurs every time a file replicates with a specified server; consequently, you may want to know about the event only if it fails. You configure events that you want to know about, based on what type of information is important to you. To configure an event, you determine three critical pieces of information: what type of event it is, what the severity level is, and how you want it handled. You configure your events using event generator and event handler documents. Event generators describe the condition that must be met for an event to be generated; event handlers describe what happens when the event occurs.

After deciding which events you want to know about, decide what will happen when the event occurs. You have several choices. You can log the event to the log file (LOG.NSF), you can mail a notification of the event to a file or an administrator, or mail the event to another application for further processing.

You create an event handler document to specify to log the event to a specified destination, and simultaneously receive notification of the event's occurrence and run a program for additional processing. You can also prevent the event from being logged or handled at all. However, if you want to know about an event, you must have an event handler document; otherwise, the event is not recorded. There is no default way of handling an event. So if you do not create event handlers, then events are not logged or stored anywhere (except for server or add-in task events, which are stored in the log). After an event is passed to the Event Monitor task, it can invoke one or more configured event handlers.

Tip: Domino events and event handlers are best used as monitoring tools to minimize *bottleneck elimination* activities to improve *proactive tuning* and therefore reducing user impact because of already occurred performance problems. Their configuration is based in the Monitoring Configuration database (EVENTS4.NSF).

Statistics collection

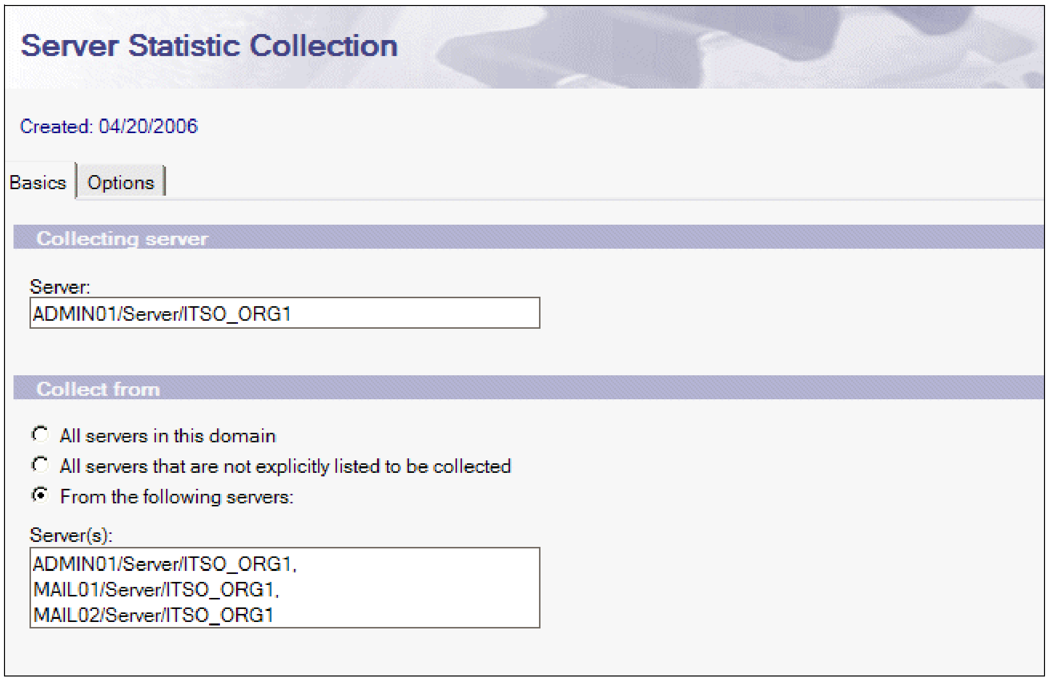
Domino continuously generates and updates server statistics, which you can collect and monitor in a number of ways.

- To collect server statistics and store them in the *server's Monitoring Results database* (STATREP.NSF), the Statistic Collector task (also called the Collector task) must be running on the server or on a server designated to collect statistics from one or more other servers.
- To use the Domino Administrator to monitor statistics, you must set up statistic Administration Preferences to generate statistics reports, which are stored in the *local Monitoring Results database* (STATREP.NSF). Then you can use the Domino Administrator to monitor and chart the statistics.

For continuos reporting to be utilized as *trend analysis*, we will focus on running statistics on the server(s) and writing them into one dedicated Monitoring Results database (STATREP.NSF).

You use a Server Statistic Collection document to designate one collector server and one or more other servers from which the collector server collects statistics. By default, the collector server reports the statistics to the local Monitoring Results database (STATREP.NSF), unless you specify a different database.

Figure 2-10 shows the Server Statistic collection setup basics, while Figure 2-11 shows the setup options.


Server Statistic Collection

Created: 04/20/2006

Basics | Options

Collecting server

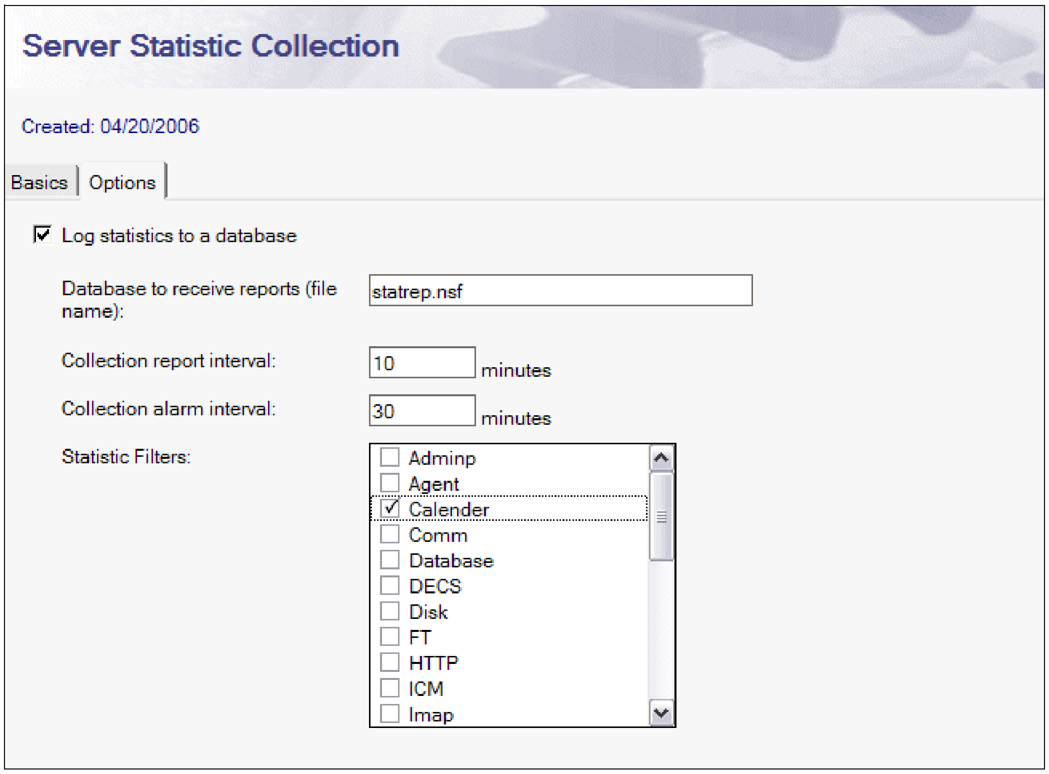
Server:
ADMIN01/Server/ITSO_ORG1

Collect from

○ All servers in this domain
○ All servers that are not explicitly listed to be collected
● From the following servers:

Server(s):
ADMIN01/Server/ITSO_ORG1,
MAIL01/Server/ITSO_ORG1,
MAIL02/Server/ITSO_ORG1

Figure 2-10 Server Statistic Collection setup, basics


Server Statistic Collection

Created: 04/20/2006

Basics | Options

☑ Log statistics to a database

Database to receive reports (file name):	statrep.nsf
Collection report interval:	10 minutes
Collection alarm interval:	30 minutes
Statistic Filters:	☐ Adminp ☐ Agent ☑ Calender ☐ Comm ☐ Database ☐ DECS ☐ Disk ☐ FT ☐ HTTP ☐ ICM ☐ Imap

Figure 2-11 Server Statistic Collection setup, options

Note: The *Monitoring Results database* (STATREP.NSF) gets created once you enable the Statistic Collector task on a dedicated machine. This database has its own replicas on each server the collector task is running on, unlike the *Monitoring Configuration database* (EVENTS4.NSF), which gets replicated throughout the domain.

Statistic Collector task

The Statistic Collector task gathers statistics for one or more servers in a domain and, by default, creates statistic reports in the Monitoring Results database (STATREP.NSF). There are two ways to set up statistic collection. You can start the Statistic Collector task on each server, which then collects its own statistics and creates reports in the local Monitoring Results database. Or you can start the Statistic Collector on one server that you set up to collect statistics from one or more servers and create reports in a specified Monitoring Results database. Figure 2-12 shows both methods. In total, there are two collector tasks running: one on the domain's administration server, and the other on the mail server.

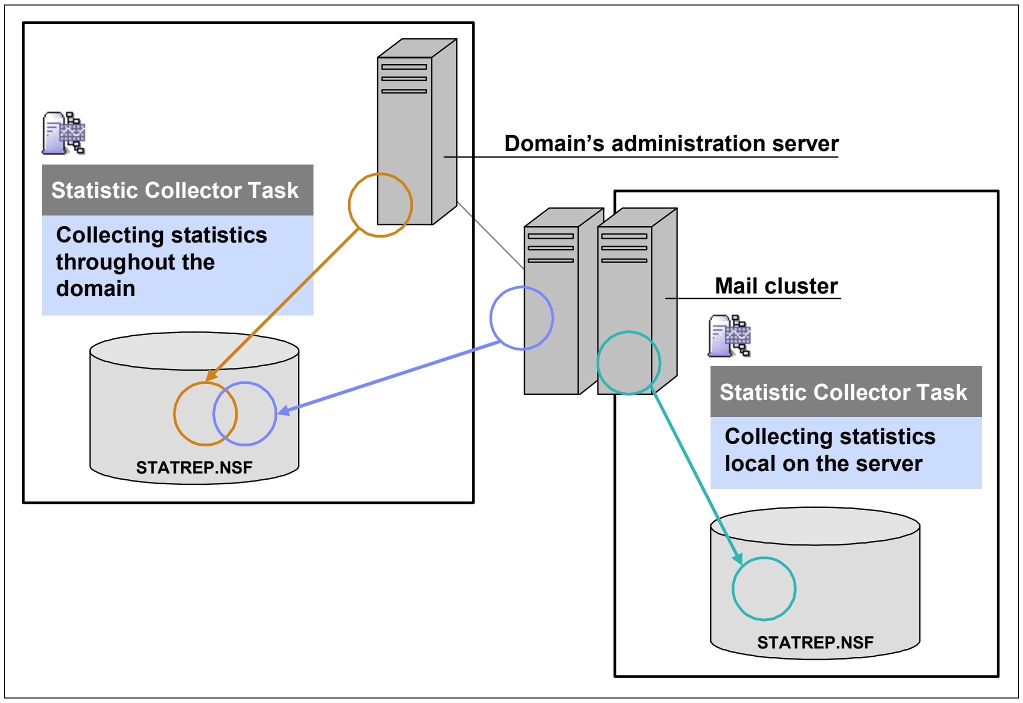

Domain's administration server
Statistic Collector Task
Collecting statistics throughout the domain
STATREP.NSF
Mail cluster
Statistic Collector Task
Collecting statistics local on the server
STATREP.NSF

Figure 2-12 Shared statistic collection within one domain: one locally, one centralized

You can also you use one designated server to collect statistics from all other servers. You have to start the Statistic Collector task only on that server and create a *Server Statistic Collection* document to identify the servers from which to collect statistics. Reports are created in the Monitoring Results database (STATREP.NSF) on the designated server (see Figure 2-13). Using this approach for collecting statistics, you must be aware that any server downtime or network interruption may impact your statistic collection.

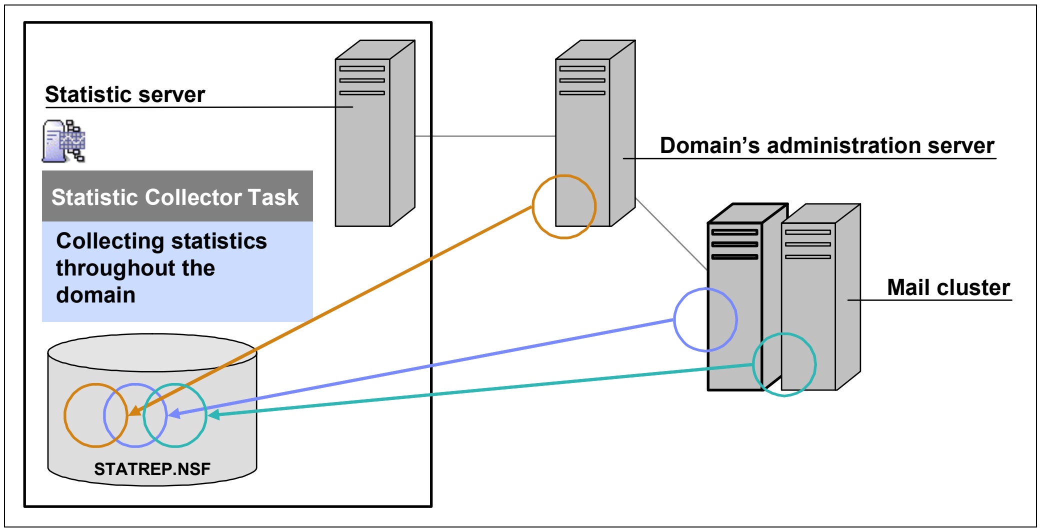

Statistic server
Statistic Collector Task
Collecting statistics throughout the domain
STATREP.NSF
Domain's administration server
Mail cluster

Figure 2-13 Designated statistic server to collect statistics from other servers

The Statistic Collector task loads automatically on a server if it is in the task line of the NOTES.INI file.

The Statistic Collector tasks provides the following areas of statistics in the Monitoring Results database:

- Calendaring and Scheduling
- Clusters
- Communications
- Mail and Database
- Network
- Platform
- System
- Web Server and Retriever

Note: The Statistic Collector task is used to produce long term statistics used for reporting and therefore to perform *trend analysis* as part of performance tuning.

Platform statistics

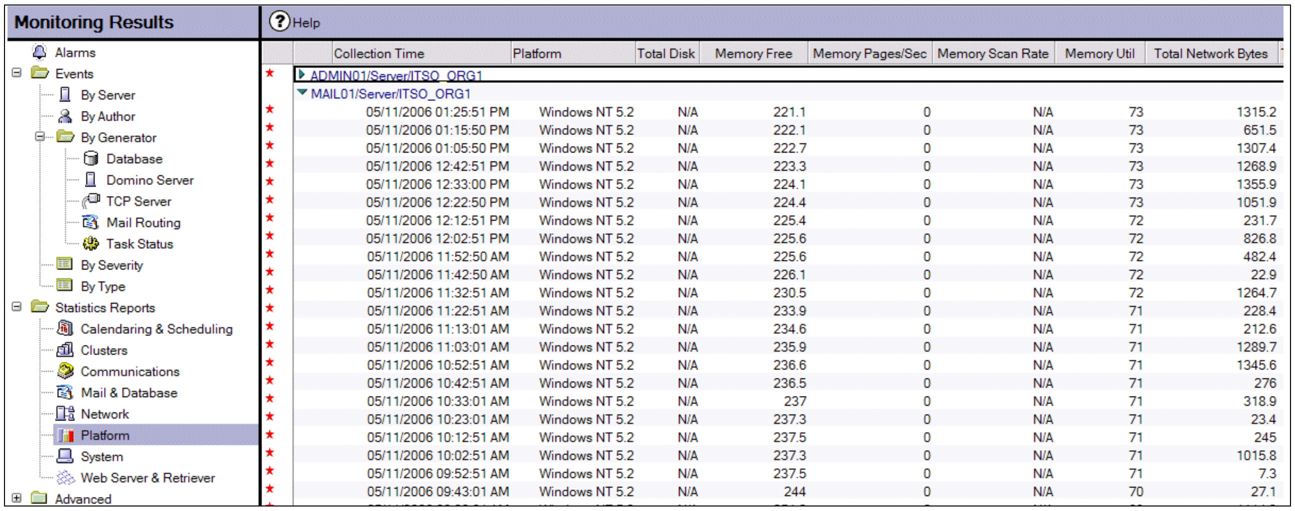
In addition to tracking server statistics, Domino tracks operating-system performance statistics. You can also view these statistics from the Domino Administrator, along with your Domino statistics, which helps you with Domino server monitoring and tuning.

You can include platform statistics in any statistic monitoring task you perform with the Domino statistics, including using them in monitoring and statistic profiles, and charting them.

There may be a slight overhead incurred while running platform statistics, but the overhead is insignificant. No disk space is consumed by enabling platform statistics, since no log files are created. As with Domino statistics, disk space is used only if you log platform statistics to the log file or to the Monitoring Results database (STATREP.NSF). The amount of disk space used depends on the frequency of capture.

By default, the Statistic Collector task continuously gathers these statistics:

Logical disk	Statistics for individual disks and total percent use of all disks
Paging file	Statistics that show the use of paging files
Memory	Statistics showing memory allocation and use, including available memory
Network	Statistics for individual network adapters and, cumulatively, for all the network adapters on the system
Process	Statistics that show the percent of CPU use, along with the process ID of Domino tasks, if the task is present. (Information for idle tasks is reported as zero.)
System	Statistics on the information captured, for example, a summary of system CPU use and queue length.



Collection Time	Platform	Total Disk	Memory Free	Memory Pages/Sec	Memory Scan Rate	Memory Util	Total Network Bytes
ADMIN01/Server/ITSO_ORG1							
MAIL01/Server/ITSO_ORG1							
05/11/2006 01:25:51 PM	Windows NT 5.2	N/A	221.1	0	N/A	73	1315.2
05/11/2006 01:15:50 PM	Windows NT 5.2	N/A	222.1	0	N/A	73	651.5
05/11/2006 01:05:50 PM	Windows NT 5.2	N/A	222.7	0	N/A	73	1307.4
05/11/2006 12:42:51 PM	Windows NT 5.2	N/A	223.3	0	N/A	73	1268.9
05/11/2006 12:33:00 PM	Windows NT 5.2	N/A	224.1	0	N/A	73	1355.9
05/11/2006 12:22:50 PM	Windows NT 5.2	N/A	224.4	0	N/A	73	1051.9
05/11/2006 12:12:51 PM	Windows NT 5.2	N/A	225.4	0	N/A	72	231.7
05/11/2006 12:02:51 PM	Windows NT 5.2	N/A	225.6	0	N/A	72	826.8
05/11/2006 11:52:50 AM	Windows NT 5.2	N/A	225.6	0	N/A	72	482.4
05/11/2006 11:42:50 AM	Windows NT 5.2	N/A	226.1	0	N/A	72	22.9
05/11/2006 11:32:51 AM	Windows NT 5.2	N/A	230.5	0	N/A	72	1264.7
05/11/2006 11:22:51 AM	Windows NT 5.2	N/A	233.9	0	N/A	71	228.4
05/11/2006 11:13:01 AM	Windows NT 5.2	N/A	234.6	0	N/A	71	212.6
05/11/2006 11:03:01 AM	Windows NT 5.2	N/A	235.9	0	N/A	71	1289.7
05/11/2006 10:52:51 AM	Windows NT 5.2	N/A	236.6	0	N/A	71	1345.6
05/11/2006 10:42:51 AM	Windows NT 5.2	N/A	236.5	0	N/A	71	276
05/11/2006 10:33:01 AM	Windows NT 5.2	N/A	237	0	N/A	71	318.9
05/11/2006 10:23:01 AM	Windows NT 5.2	N/A	237.3	0	N/A	71	23.4
05/11/2006 10:12:51 AM	Windows NT 5.2	N/A	237.5	0	N/A	71	245
05/11/2006 10:02:51 AM	Windows NT 5.2	N/A	237.3	0	N/A	71	1015.8
05/11/2006 09:52:51 AM	Windows NT 5.2	N/A	237.5	0	N/A	71	7.3
05/11/2006 09:43:01 AM	Windows NT 5.2	N/A	244	0	N/A	70	27.1

Figure 2-14 Platform statistics from the Domino's Monitoring Results database

By default, platform statistics are enabled. To disable platform statistics, enter this setting in the NOTES.INI file and then restart the Domino server:

```
Platform_Statistics_Disabled=1
```

Note: To get more detailed platform statistics to be used as input for Domino 7 server performance tuning and trend analysis, you may use the performance monitoring and reporting tools provided by your platform operating system. These are, for example, the Performance console (Perfmon) on Windows platforms or KDE System Guard for Linux. Since we keep focusing on the Domino performance tuning part, platform-specific monitoring and reporting tools are not covered within this Redpaper.

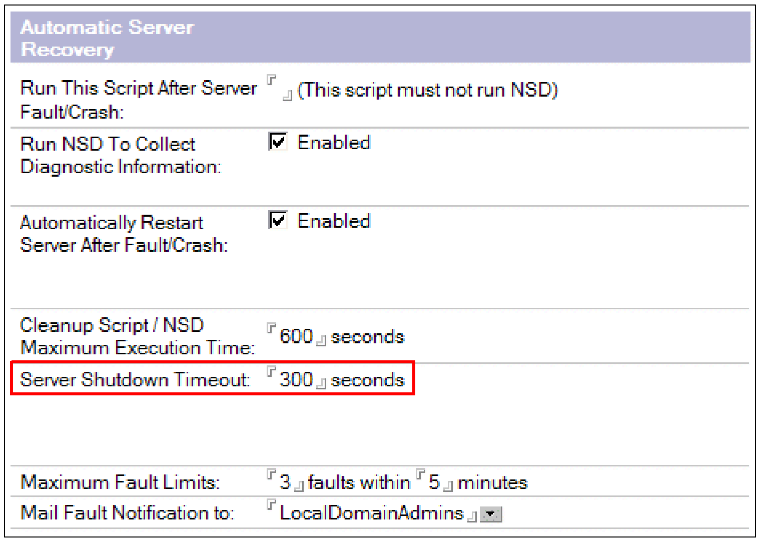
Please refer to platform specific IBM Redbooks and Redpapers available at:

http://www.redbooks.ibm.com

Monitoring server shutdown

Domino's Shutdown Monitor is a new Lotus Domino 7 feature that is an extension to the *Automatic Server Recovery* toolset. This task ensures that Domino terminates when requested to do so, that is, the server does not hang while attempting to shut down. When you request a server shutdown, the System Monitor task monitors the shutdown activity. If no activity occurs for a specified amount of time, Domino automatically acts as though the server has crashed and takes appropriate action, including collecting data about the state of the server and then terminating the server. An NSD log is generated before the server terminates. This reduces server downtimes and helps you to identify possible failure causes even if a **`quit`** command was sent as part of *bottleneck elimination* activities.

You specify the amount of time that Domino monitors the shutdown activity in the field, Server Shutdown Timeout, in the Automatic Server Recovery section of the Server document. To disable this feature, enter 0 (zero) in the Server Shutdown Timeout field (see Figure 2-15).



Automatic Server Recovery	
Run This Script After Server Fault/Crash:	(This script must not run NSD)
Run NSD To Collect Diagnostic Information:	✓ Enabled
Automatically Restart Server After Fault/Crash:	✓ Enabled
Cleanup Script / NSD Maximum Execution Time:	600 seconds
Server Shutdown Timeout:	300 seconds
Maximum Fault Limits:	3 faults within 5 minutes
Mail Fault Notification to:	LocalDomainAdmins

Figure 2-15 Monitoring server shutdown

Tip: If you are using *scheduled tasks* or **`at`** commands to shut down the Domino service and perform maintenance work on your Domino system, the Shutdown Monitor ensures a clean termination of all Domino tasks. This is important if you plan to run **`fixup`** or **`updall`** tasks outside the Domino server session.

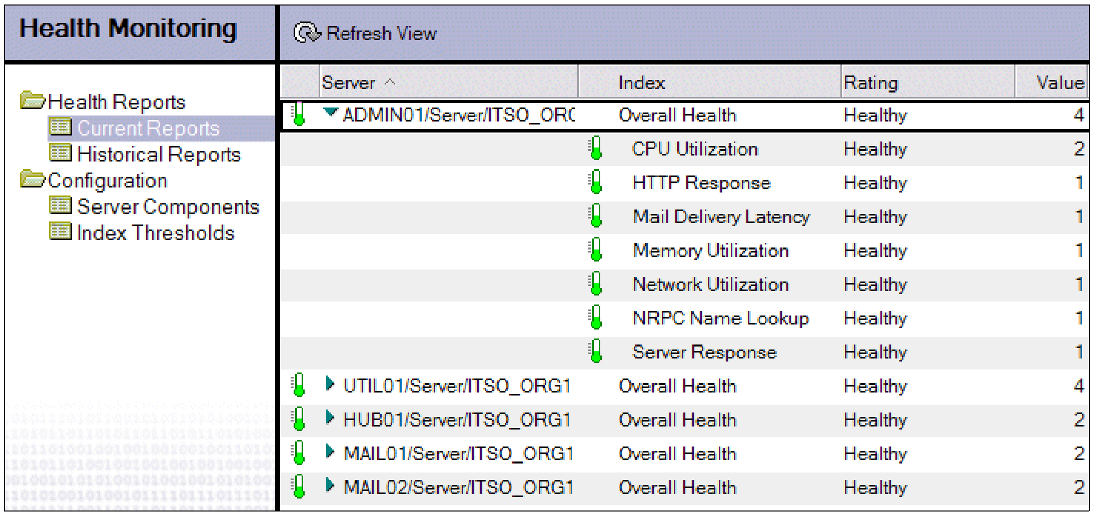
Server Health Monitor

The Server Health Monitor extends the usefulness of traditional performance troubleshooting by automatically calculating health statistics, comparing those statistics to predefined thresholds, and reporting on overall server health. If the server health rating is Warning or Critical, a health report, which is stored in the Health Monitoring database (DOMMON.NSF), suggests short-term and long-term recommendations for tuning the server and returning its performance status to Healthy.

The Server Health Monitor is incorporated into the Domino server monitor, which is part of the Domino Administration client. All health statistics generated by the Server Health Monitor are local to the Domino Administration client.

For each server being monitored, the Server Health Monitor reports a health rating for the server and for all enabled individual server components, namely, CPU, disk, memory, and network utilization; NRPC name lookup; mail delivery latency; and server, HTTP, LDAP, and IMAP response.

If you use the Server Health Monitor, the Current Reports view of the Health Monitoring database (DOMMON.NSF) displays a health rating for each monitored server and server component (see Figure 2-16).

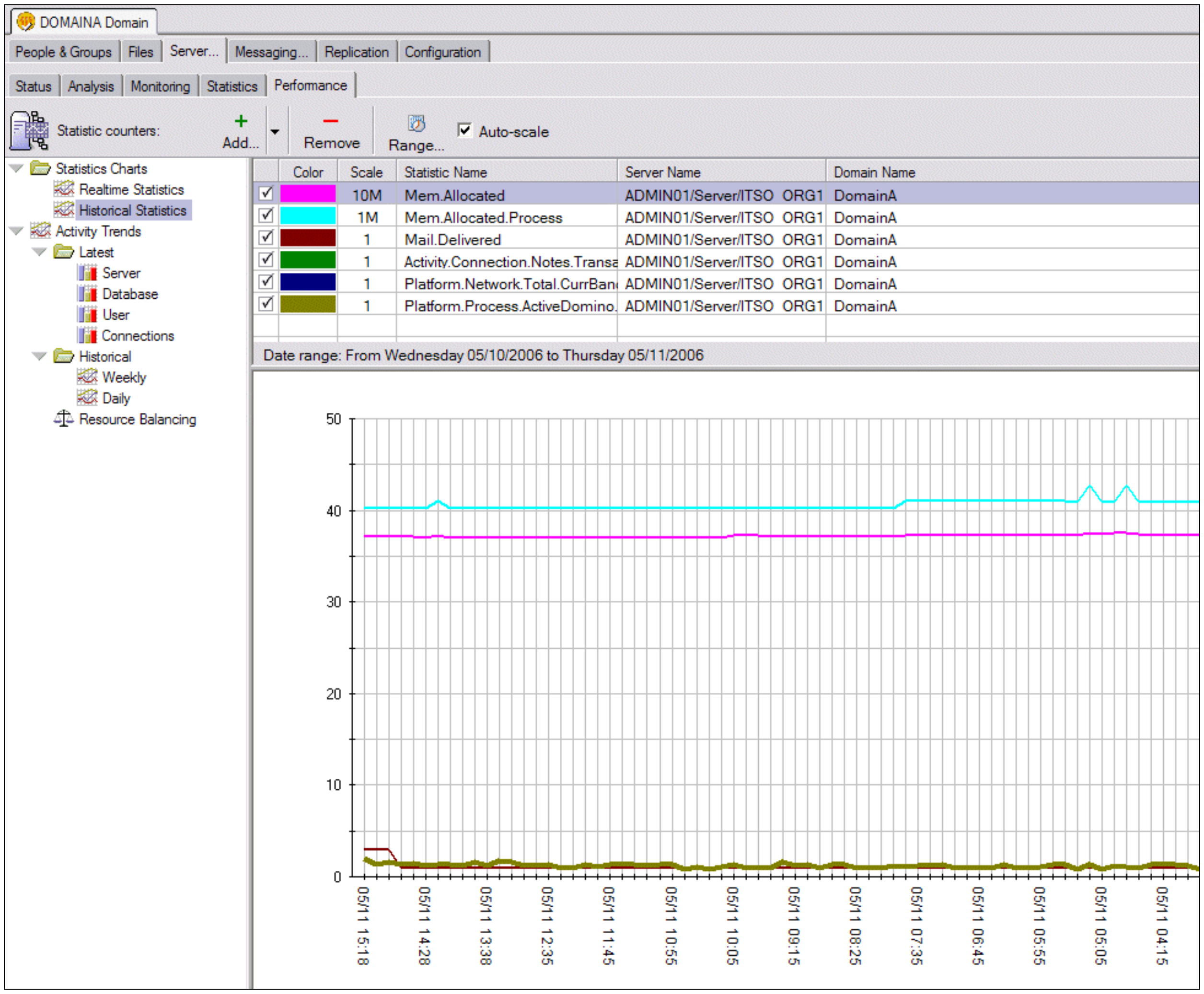


Server	Index	Rating	Value
ADMIN01/Server/ITSO_ORC	Overall Health	Healthy	4
	CPU Utilization	Healthy	2
	HTTP Response	Healthy	1
	Mail Delivery Latency	Healthy	1
	Memory Utilization	Healthy	1
	Network Utilization	Healthy	1
	NRPC Name Lookup	Healthy	1
	Server Response	Healthy	1
UTIL01/Server/ITSO_ORG1	Overall Health	Healthy	4
HUB01/Server/ITSO_ORG1	Overall Health	Healthy	2
MAIL01/Server/ITSO_ORG1	Overall Health	Healthy	2
MAIL02/Server/ITSO_ORG1	Overall Health	Healthy	2

Figure 2-16 Current Reports view of the Health Monitoring database (DOMMON.NSF)

Activity Trends

Activity Trends, which is part of the Domino Administrator, collects and stores activity statistics as current observations and historical trends. The activity statistics relate to the server, databases, users, and connections of users to databases. You can explore the collected data to see how database workload is distributed across servers (see Figure 2-17). Using the data, Activity Trends recommends a resource-balancing plan.



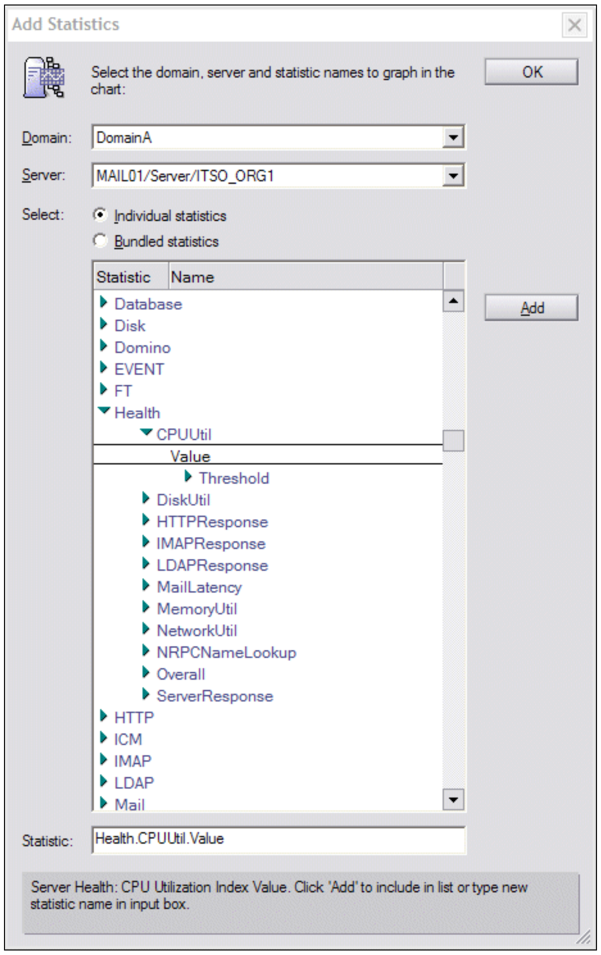
	Color	Scale	Statistic Name	Server Name	Domain Name
✓	🟪	10M	Mem.Allocated	ADMIN01/Server/ITSO ORG1	DomainA
✓	🟦	1M	Mem.Allocated.Process	ADMIN01/Server/ITSO ORG1	DomainA
✓	🟥	1	Mail.Delivered	ADMIN01/Server/ITSO ORG1	DomainA
✓	🟩	1	Activity.Connection.Notes.Transa	ADMIN01/Server/ITSO ORG1	DomainA
✓	🟦	1	Platform.Network.Total.CurrBan	ADMIN01/Server/ITSO ORG1	DomainA
✓	🟨	1	Platform.Process.ActiveDomino.	ADMIN01/Server/ITSO ORG1	DomainA

Date range: From Wednesday 05/10/2006 to Thursday 05/11/2006

Figure 2-17 Displaying historical activity trends in the Domino Administrator client

Domino activity trends may be used as reports for *trend analysis* performance tuning. It provides visualization of a couple of Domino server, platform, and network statistics. You can create different profiles to examine and select the target servers, statistics, and time ranges. This helps you to focus on the respective performance relevant data for the different server types.

Figure 2-18 on page 35 shows a statistic selection for Activity Trends.



Select the domain, server and statistic names to graph in the chart:

Domain: DomainA

Server: MAIL01/Server/ITSO_ORG1

Select: [✓ Individual statistics] Bundled statistics

Statistic Name
Database
Disk
Domino
EVENT
FT
Health
CPUUtil
Value
Threshold
DiskUtil
HTTPResponse
IMAPResponse
LDAPResponse
MailLatency
MemoryUtil
NetworkUtil
NRPCNameLookup
Overall
ServerResponse
HTTP
ICM
IMAP
LDAP
Mail

Statistic: Health.CPUUtil.Value

Server Health: CPU Utilization Index Value. Click 'Add' to include in list or type new statistic name in input box.

Figure 2-18 Statistic selection for Activity Trends

A list of all statistics available in Domino can be found in the Advanced section of the Monitoring Configuration database (EVENTS4.NSF). The activity logging itself get enabled via *configuration* documents in the Domino Directory. A detailed description on how to set up Activity Trends are available in the Domino 7 administrator Help.

3

Managing Performance Data and establishing a baseline

From the point of view of performance analysis, historical statistics are most valuable as baseline information, both for making tuning decisions, and for analyzing the results of tuning events. Having a set of baseline historical data provides a Domino administrator with a foundation against which changes in performance can be measured.

This chapter looks at the retention of Domino-related performance information, and how that information is used in managing a Domino environment. The following topics are covered:

- The importance of historical data
- Types of information and retention options
- Using historical data

3.1 The importance of historical data

Domino administrators understand the value of gathering statistics regularly to monitor their systems' activity and platform usage. The analysis of server logs and events can be particularly useful for identifying and resolving issues that might occur on a daily basis. However, in the long run, it is the collection, retention, and analysis of Domino historical statistics gathered by the Statistics Collector task that gives Domino administrators the ability to identify trends in the behavior and usage of servers in their domain.

While statistical data is necessary for daily troubleshooting and bottleneck analysis, maintaining a collection of long-term statistical data is also critical to validating decisions related to performance management, system-wide load balancing, and capacity planning.

Maintaining a collection of historical data can help administrators directly address comparative analysis questions, such as:

- What are the performance characteristics of our system under normal user loads as opposed to heavy load days?
- How have the operational characteristics of our system changed over time?
- What has been the impact of user and application growth on our Domino environment?
- What is the potential impact of adding additional users or applications, assuming a similar usage profile?
- Do we have the excess capacity on any given servers to better balance mail users or collaborative applications?
- How fast are my servers consuming system resources, such as memory or disk storage, and when might we need to consider upgrades?
- What impact have we seen as the result of recent system tuning or upgrades?

By effectively leveraging historical statistical data, administrators can better project when and where performance problems might occur, and provide quantitative justification for the execution of proactive plans to address potential issues before they become problems visible to users and management.

3.1.1 Limits of historical data

Capturing and analyzing long-term statistics can be very beneficial when performing performance analysis, capacity planning, and system re-architecture. However, there are limits to the viability of historical data.

It is important to remember that any given set of statistics have to be taken within the context of the specific system load and configuration that existed at the time that those statistics were gathered. Thus, it is critical to maintain a separate, detailed chronology of major system changes to be used by the Domino administrator as a part of their overall analysis of the statistics.

Important: Avoid making assumptions when analyzing historical statistics. Make sure you know what system changes occurred, and when they occurred, so you can *properly* interpret the statistic.

3.2 Creating baselines for performance tuning

From the point of view of performance analysis, historical statistics are most valuable as baseline information, both for making tuning decisions, and for analyzing the results of tuning events. Having a set of baseline historical data provides a Domino administrator with:

- A complete view of the infrastructure's performance
- The means to compare the benefits of potential tuning with its implementation costs
- A set of quantitative data points against which the results of the tuning event can be measured
- A sampling that can be used as the basis for long term trend analysis

Baselines performance data is not a simple snapshot of a system before a tuning event takes place. It should:

- Be representative of normal system loads
- Be a sampling for a set period of time (at least one week, preferably more)
- Not include data from periods with unusual events (system failures, high usage volumes, and so on)
- Include descriptions of what components or services were running on individual servers

Baseline data is a specific subset of the full collection of historical statistical data, and should be saved separately from the full collection. This baseline data can be saved as:

- An archived subset of data in the Notes database
- A spreadsheet export from a period of the historical data

A new set of baseline statistics should be saved after the infrastructure has stabilized upon the completion of every major tuning change or system upgrade.

3.3 Types of information and retention options

The majority of the historical data in a Domino system can be collected by the Statistics Collector task, and stored in the Monitoring Results database (starep.nsf). The Statistics Collector task can collect statistics from Domino services and from the platform on which the Domino server runs.

The Statistics Collector can be configured to run on each server, storing the statistics in a local Monitoring Results database on that server. Also, the Statistics Collector task on one server can extract statistics from many servers and collate them into a single Monitoring Results database.

In general, is usually easiest on the Domino administrator to have a single server collect statistics from multiple server and collate the results into a single database. However, there may be situations where this is not feasible. For example:

- If there are a very large number of servers in the data center or the Domino domain, it may not be possible for a single server to gather and collate statistics from all the server.
- If there are low speed or intermittent network or communication connections between the servers, the collector server may not be able to gather from the remote server in a timely manner.

It may be necessary to have a number of collection scenarios in a given Domino domain, in which case it becomes the Domino administrator's responsibility to collate the statistics when performing domain-wide analysis.

Domino administrators should develop a process for archiving statistics from Monitoring Results databases in order to keep the size of the statrep.nsf database on collection servers to a reasonable size, and to provide a means of organizing statistics by time frame to make the management, usage, and retention of those statistics easier. Depending on the amount of data and the retention cycle required, it might be appropriate to create and maintain archives by month, by quarter, or by year.

3.3.1 Domino data

The Statistics Collector task gathers detailed metrics on Domino processes:

- Scheduled agents
- Calendaring and Scheduling
- Communications
- Database replication and usage
- Mail routing and usage
- Protocol handing tasks (LDAP, IMAP, POP, NNTP, and so on)
- Server tasks
- Web servers

Domino statistics are specific to a particular Domino server instance. They may be:

- Cumulative from the time of last server restart
- Counts over a specific period of time
- A snapshot of the condition at the time the collector task ran

3.3.2 Non-Domino infrastructure data

The Statistic Collector task can also continuously gather metrics for the physical platform on which the Domino server is running:

- Logical disk
- Memory
- Network
- Process
- System

When collecting statistics from a partitioned server, Domino collects platform statistics that pertain to the system as a whole, not to an individual partition. For example, memory use or CPU use statistics are the same value on a partitioned and non-partitioned server. The only statistics that are specific to a partition are those that reflect tasks, such process statistics, where one partition might run 10 tasks, while another partition runs 15 tasks.

3.3.3 System change data

Domino administrators must maintain a record of all changes made to the Domino environment so they can have the necessary context information needed to interpret statistics generated by the Domino system. These changes may include:

- Performance tuning activities
- Hardware upgrades
- Deployment or removal of major applications
- Addition or removal of mail users
- Re-tasking servers to perform different utility functions

Often, organizations have a formal change management process in place that can be used to track this system change data. However, the Domino administrator should make sure that the change records contain *all* changes made to the system over time, and that retention period for the change records is the same as that of the historical statistics data.

If necessary, Domino administrators should maintain their own change record to the degree of detail they find necessary. At a minimum, these change records should include:

- The date and time the change was made.
- The party who made the change.
- The systems affected by the change.
- The specific value, setting, or component that was changed (both before and after condition).
- Whether the change was committed or rolled back.

3.3.4 Retention periods

Raw historical statistics tend to lose their value over long periods of time. In general, unless there is some statutory reason for doing so, historical statistics should probably be kept for no more than one or two years.

At least three sets of historical benchmarks should be available at any given time to use for comparative analysis against the current system.

3.4 Using historical data

In addition to providing support for system tuning, Domino statistics can be leveraged in several other ways to help administrators better manage their system, and to provide quantitative feedback to users and management regarding system health and usage.

3.4.1 Capacity planning

Historical statistics can provide Domino administrators with a a strong foundation for performing capacity analysis and for making planning decisions. Domino statistics give an as-is picture of the current Domino environment. Administrators can use this historical data to model various what-if scenarios, and attempt to project how the system would preform under different loads.

It is important to recognize that historical data can be useful in defining and validating capacity models, but only within a limited context. Historical data is, by definition, a trailing indicator of capacity and performance.

While it may be possible to make some generalized capacity predictions based on past performance, statistics tend to lose their “predictive value” when dealing with major system upgrades, changes to host operating system, or the re-tasking of a server or a system to perform different functions.

3.4.2 Reporting

Domino administrators often find it useful to create reports based on statistical information on a regular basis. These may be created for administrative analysis, or they may be required to provide users or management with a summary review of the system activity or performance. Reports might include weekly or monthly mail usage summaries, graphs showing server performance trends, or spreadsheets detailing issues with agent failures or dropped sessions.

In some cases, reports can be generated using the Domino Administrator Client with statistics pulled from a server’s active Monitoring Results database. However, if the statistical data is stored in an offline archive, it will be necessary to export the data from the archive and create a custom report. If exported to a spreadsheet format, the Domino administrator can use the the spreadsheet’s data analysis and graphing tools to generate custom reports.

4

Road map for how and where to begin performance tuning

The previous chapters covered a set of performance tuning basics and definitions as well as tips and recommendations about the importance of establishing a baseline and how to begin approaching performance tuning. We will now focus on where and how you should start your work for performance tuning activities.

4.1 Seeking performance bottlenecks

By knowing the different areas where tuning may be applicable (as covered in 2.1, “Defining Domino performance within the IT infrastructure and the context of the Domino Architecture” on page 10) as well as classifying the different roles of your Domino server (see 2.2, “Considerations for different types (roles) of Domino servers” on page 15), we now concentrate on *how to find* the areas where performance pain may exist.

Since Domino is handled as an application within a server instance, there are many items interacting within the whole system. Each of these items may become a bottleneck when speaking about performance. This could be the hardware itself, the operating system, the Domino server, or even the interfaces between these components. Each of these components have several parts that may be key when speaking of performance. Figure 4-1 shows an overview of these components, the subcomponents, and their relationship.

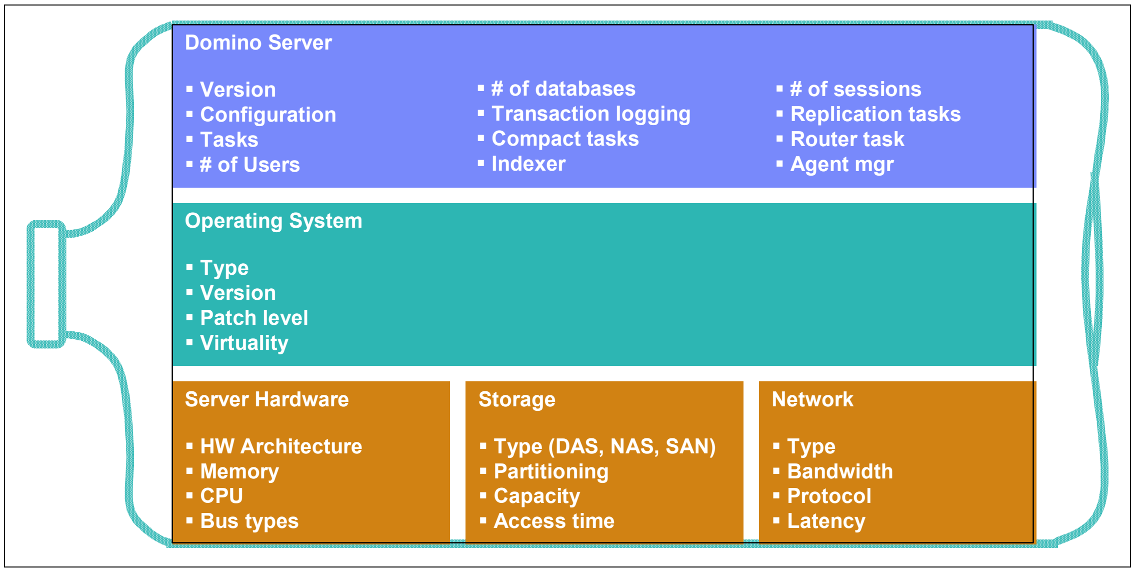


Figure 4-1 Performance bottlenecks may occur through the entire server instance in several parts

As you can see from this figure, with the foundation consisting of the server hardware, the storage component, and the network interface, it is important to select your platform, including the hardware, the OS, and other components, very carefully according to your business requirements. These are the foundation components of your Domino infrastructure.

Within the scope of this Redpaper, we focus primarily on the Lotus Domino component and where bottlenecks may occur within the scope of Domino and its immediate interaction with the underlying foundation infrastructure components. This will help Domino administrators to keep their systems stable and to intervene prior to failures because of performance issues.

Assuming that a Domino server is assigned for a specific role (see 2.2, “Considerations for different types (roles) of Domino servers” on page 15), there are also different areas to focus on when searching for performance bottlenecks and analyzing trends. For example, the number of pending messages or delivery times are critical for a hub server’s performance, but not for an application server.

Searching for bottlenecks and identifying them prior to when they manifest themselves as critical points us directly to two essential activities in server operating: *Monitoring and Reporting*. We covered these topics in 2.3, “Defining monitoring and reporting” on page 18.

4.2 Ten steps about where and how to start

The following section is intended as a road map, giving you an initial plan on where to start. From here, you can then proceed to Chapter 5, “Understanding what to tune” on page 49 to begin making appropriate changes within the Domino Server.

1. Have a plan.

   Performance tuning activities, whether doing a dedicated performance bottleneck search or a trend analysis, must be planned well in advance. A systematic procedure lets you maintain control about the ongoing processes and helps you execute your plan in an orderly, correct way.

2. Know your infrastructure.

   To correctly execute performance tuning activities, you must have detailed knowledge about your Domino infrastructure, including hardware configurations, system platforms and network topology. You will need this to determine the right components to be included in analysis activities.

3. Break down components.

   Split your analysis targets according to your infrastructure component’s tasks. Consider the different roles of your infrastructure servers, as described in 2.2, “Considerations for different types (roles) of Domino servers” on page 15. The analysis scope and target differ within the Domino infrastructure, and will be influenced by the role of each specific server.

   Once you have completed the step above, consider the most important dependencies of each component. For example, to do performance tuning on a user’s mail server, the following reports and statistics would be important to analyze:

   – Platform statistics report (Memory, CPU, and disk I/O)
   – Notes mail statistics
   – Network report
   – Server Load Statistics

   The statistic collection task (see “Statistics collection” on page 27) will provide you with this information.

   For web servers, mail statistics will be less important, but you will certainly need a Web statistics report. Accordingly, you must adapt the performance bottleneck analysis to your individual infrastructure.

4. Have a baseline as a reference.

   Before you start tuning, you must set a baseline to measure against (see 1.1.4, “Performance baselines” on page 6, as well as Chapter 3, “Managing Performance Data and establishing a baseline” on page 37). Once the components to be included in the analysis are set, you should define a time line in advance to gather current system state / “as-is” statistics. Do this for each type of performance measurement and target infrastructure component, for example, one component and time baseline for mail server, one for application servers, and one for hub servers.

Important: It is absolutely critical to have a baseline to measure against; otherwise, any changes in performance cannot be measured accurately.

5. Measure baseline and analysis results over time.

   In order to have consistent results for both the baseline and the validation data after performance tuning, the statistic collection mechanism should run several cycles. It is important to perform a trend analysis on a good baseline to have an accurate picture of what is happening. If your time line for the performance baseline is defined between Monday, 8 a.m. until Friday 8 p.m. (because this is the company's core business time), collect the statistics for several weeks without changing the infrastructure in order to stabilize the measurement data (see Figure 4-2).

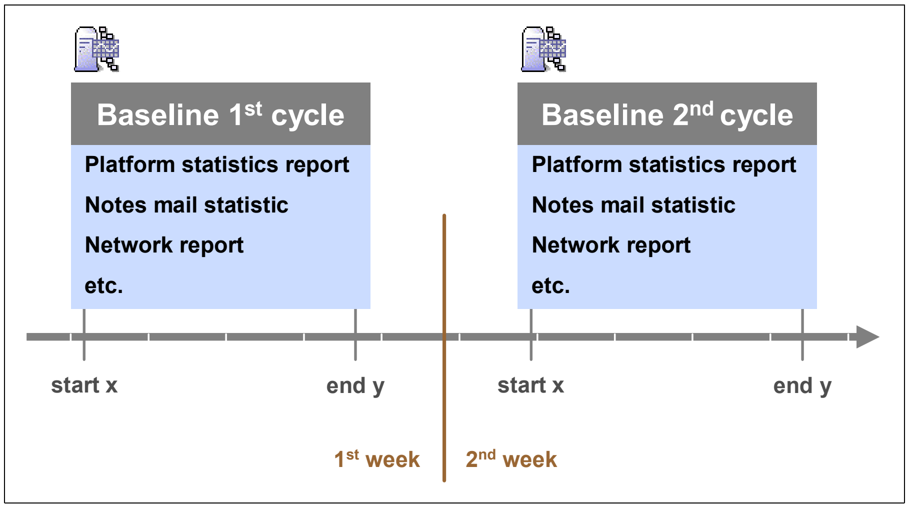


Figure 4-2 Multiple baseline cycles

   Once you have a baseline defined, you should be able to analyze the data and identify possible bottlenecks. Do this by comparing the behavior of each reported component against each other over the reported time. On Lotus Domino mail servers, for example, have a look at the system memory and CPU usage in comparison with the current number of users and disk I/Os on the machine. Pay close attention and note the peaks of each reporting component, as this will give you information about possible bottlenecks in your overall system (see Figure 4-3 on page 47). If the results are not satisfying, extend the components within the reporting by adding more data, for example, disk capacity, database sizes, additional Lotus Domino tasks, and so on.

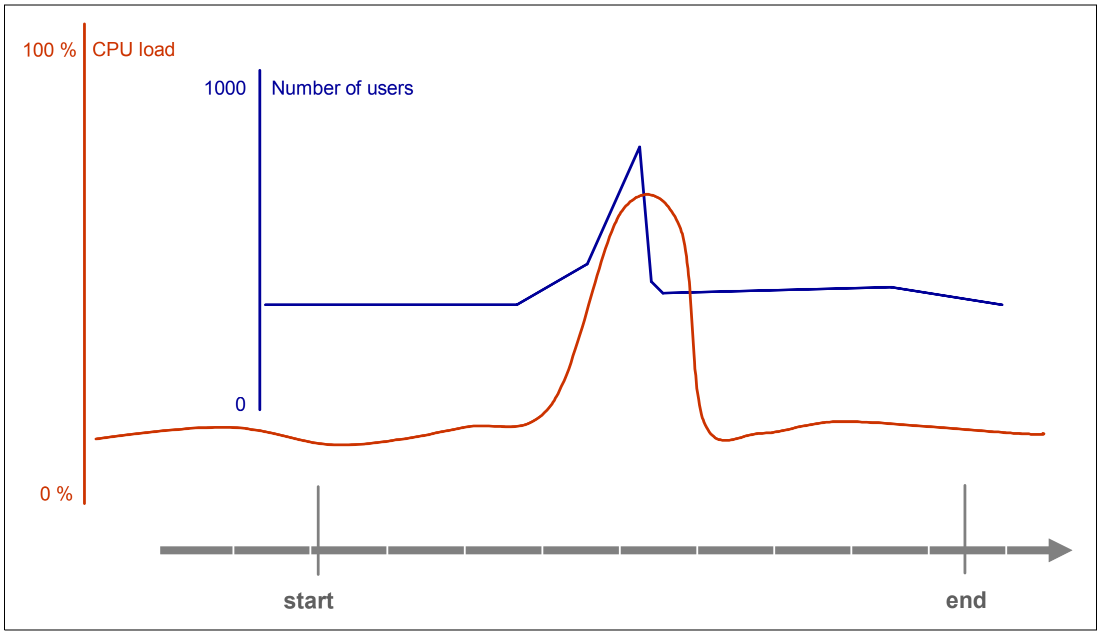


Figure 4-3 Analyzing performance bottlenecks based on baseline reporting

6. Estimate benefits versus costs.

   Before planning the implementation of any performance tuning related changes, estimate the performance benefit of these changes and compare them against the implementation costs the change causes. If you are planning to perform a couple of performance related changes within your infrastructure, start with those having less implementation effort and cost. They may result in big performance improvements. A couple of big changes like restructuring the system landscape or changing the architecture often result in less improvement when compared to the *smaller, more incremental changes*.

7. Have a change management plan.

   Plan and document every change you implement to *make your performance tuning activities traceable*. This is very important in order to have knowledge about the root causes of possible issues, changes in baselines, or ongoing systems behavior. Additionally, discuss your planned changes with your customers, whether internal or external, to determine the best time slots for change implementations or to outline possible business impacts.

8. Prepare a back out and recovery plan.

   If the planned change does not generate the expected results or even downgrades your server performance, be aware of the requirements to undo changes. This has to be planned in advance before implementing the change.

9. Do only one thing at a time; isolate individual changes.

   As described in 1.1.2, “Tuning process” on page 3, it is important to do performance tuning *step by step*. Before changing additional variables, do data collection to proof and verify the results (see Figure 4-4).

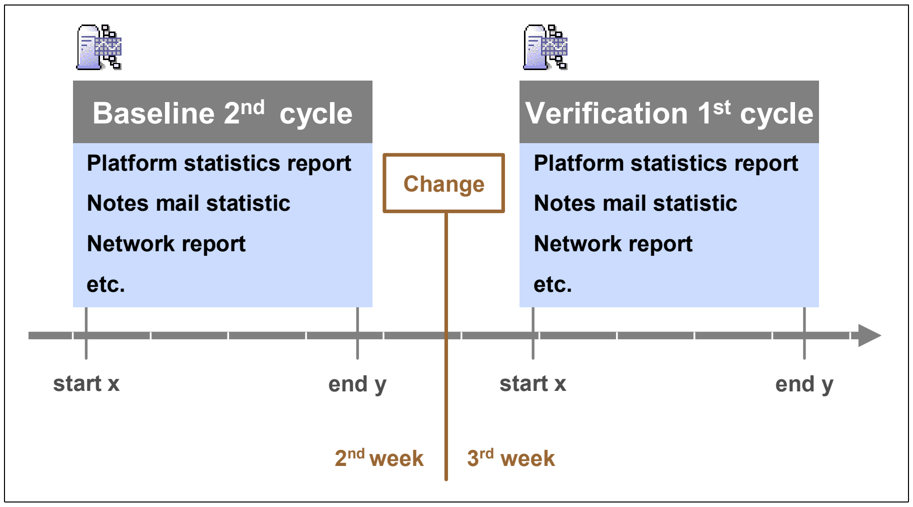


Figure 4-4 Result verification after change implementation

10. Finally, let your changes take hold and measure them over time.

    After a successful change, do not implement the next change right away. Do a careful verification of your changes, multiple verification cycles (as shown in Figure 4-4) and finally *adapt your baseline* before starting the next change cycle. Performance tuning by doing trend analysis is a long term process during an infrastructure lifetime. Additionally, changes of business requirements or reorganizations may impact the infrastructure performance and requires a reassessment of your reports.

5

Understanding what to tune

This chapter primarily discusses several functional areas of Domino that may need tuning, as well as some specific examples of what to tune and how. The focus will be on the most prevalent Domino tasks, such as replication, adminp, mail routing, indexing, and so on.

This chapter also briefly touches on some networking, server hardware, and operating system considerations, since these are an integral part of the larger Domino Architecture and it is impossible to ignore the impact they have on the overall stability and performance of the Domino server itself.

Note: A further discussion of similar topics, specifically as they relate to troubleshooting, can be found in Technote TN #1234550, Domino Server Performance Troubleshooting Cookbook. This technote can be accessed at:
`http://www.ibm.com/support/docview.wss?rs=899&uid=swg21234550`

5.1 Performance tuning outside of Domino

In order for Domino to perform at its best, many other areas outside of Domino must be performing at their best as well. Domino relies on the operating system hosting the server, the hardware running the operating system, and the network connecting it all to perform well. If any one of those components are configured poorly, or are underpowered, then Domino's performance will be negatively impacted.

This section will *not* provide detailed configuration and tuning information for any one type of network, server hardware, or operating system, but will instead touch on tuning concepts that should be considered across the board, with some OS specific examples of how to determine whether or not tuning may be necessary.

5.1.1 Network performance

The network is the backbone for the entire Domino infrastructure. The connection between a client (Notes, Web, and so on) and a Domino server can only be as fast as the network will allow. Therefore, if the network does not have sufficient bandwidth to accommodate the load, then *every* network related activity will be negatively impacted, including Domino.

There is no way to give a thorough discussion on network performance tuning within the confines of this Redpaper. The attention to detail that networking requires and deserves could fill volumes. This is due the nearly infinite combination of network types, topologies, and hardware devices that are possible within an enterprise.

For information about configuring and optimizing your network, the hardware vendor(s) responsible for providing the hardware should be able to provide guidance on where to go for help.

5.1.2 Platform architecture and system performance

As is the case with network performance, the hardware platform (the physical machine running the software) can dictate whether or not Domino can perform at an acceptable level. The system's architecture (x86, RS/6000®, and so on), CPU(s), RAM, and disk subsystem all play parts in the performance of the system, and therefore Domino. If any one of these components is underperforming, then the system as a whole will perform poorly.

Note: The hardware for both servers *and* clients have an impact on the perceived performance of a Domino server, so both need to be configured correctly.

One of Domino's biggest strengths is the variety of platforms that can host Domino servers. That said, selecting the best platform for your business falls outside of the scope of this Redpaper. There are simply too many considerations unique to each client's needs to cover the topic here. Instead, this section will cover topics common to *all* platforms.

Most of the tools used to determine whether a bottleneck exists within a specific component are operating system specific. For this reason, examples will be covered in 5.1.3, "Operating system performance" on page 52 for some of the available tools.

CPU

In most cases, the number of CPUs or processors on a system has a more significant impact on a server's performance than the actual speed of the CPUs. Loosely speaking, each CPU can process a separate instruction from other CPUs on the system simultaneously instead of having to wait for a single CPU.

Important: Architectural limitations in both hardware and software prevent a direct 1:1 correlation between performance improvement and additional CPUs.

Note: While the above explanation is a bit simplistic, it should make it easier to understand why multiprocessor systems are generally more advantageous than uniprocessor systems.

In reality, newer technologies, such as *simultaneous multithreading* (SMT), also known as *hyper-threading*, or chip-level multiprocessors (CMP) allow for multiple instructions to be executed by each CPU in a single time slice.

RAM

Here we discuss the importance of RAM in performance tuning.

Why is RAM important?

Physical RAM is the place where assorted process and thread data is temporarily stored so that calculations and manipulations of that data by the CPU can occur. Effectively, this means that the more RAM a system has, the more temporary data can be held there for processing.

What makes RAM important from a performance standpoint is the fact that the amount of time it takes a processor to access the data from RAM is measured in nanoseconds, while hard drive speed is measured in milliseconds. This means that it may be thousands of times faster to access and manipulate the data from RAM than from the hard drive on a system.

The operating system itself is responsible for determining when information should be placed into memory and when it should be committed back onto the disk. This process is generally referred to as paging, or swapping.

How much RAM can be used?

Different platforms have different limitations on the amount of RAM they can support directly. For instance, 32-bit platforms, such as the x86 architecture, are only able to access 4 GB (2^32-1) of physical RAM *directly*, while 64-bit architectures, such as RS/6000 or SPARC, are *theoretically* able to access 16 EB (2^64-1)MB.

Note: While 32-bit hardware is inherently limited to 4 GB worth of addressable memory, modern versions of both Linux and Windows have implemented software based methods of accessing more memory on x86 systems.

The amount of RAM available to a given process (including Domino) is further restricted by operating systems as well. Operating systems also have the ability to combine the physical RAM with additional hard drive space to create *virtual memory*. This virtual memory is actually where allocations of memory to processes occur, and it is up to the operating system to distribute the memory to the applications. More information about this area may be found in "OS memory allocation" on page 52.

Disk subsystem

Hard drive performance may be the one of the most often neglected components in a system, yet increasing performance here can yield significant improvements for the entire system. Faster and larger hard drives are constantly becoming available at lower and lower costs; with this in mind, drive performance should be high on the list of potential areas to tune.

Disk I/O

Using separate physical disks to distribute operating system files, swap/page space, transaction logs, and data can improve server performance. Furthermore, using RAID 0+1 hardware for the disk drives that the data files are on or using separate striped volumes for swap space improves server performance at high loads on systems that have high swap rates.

See 6.5, “Hard disk performance” on page 108 for more information about the disk subsystem.

5.1.3 Operating system performance

The operating system hosting Domino is responsible for allocating the system’s resources for Domino to use. Any time a Domino server needs more memory, CPU time, disk space, and so on, it is up to the operating system to provide it. If those resources are not readily available in a timely manner, then Domino’s performance will suffer as a result.

Each operating system handles performance tuning a bit differently, but they all have some basic components in common. They all offer ways to analyze the utilization of *CPU, memory, disk*, and *networking* resources to determine if any improvements can be made.

Note: This portion of the Redpaper will only touch on memory, CPU, and disk utilization, due to the tremendous scope of networking, as previously discussed in 5.1.1, “Network performance” on page 50.

An important concept to understand regarding OS performance tuning is that a change to one area of the system may not yield a noticeable difference if the bottleneck is in a different area, or it may reveal other problems with the system that were previously masked. For example, adding additional CPUs or disk space to a system may do nothing if there is too little RAM. Likewise, adding more RAM to the system may not help if the CPU is insufficient to process the workload. Money and time can be wasted if this point is taken for granted.

OS memory allocation

A common misconception is that Domino can use all of the real memory on a system. So if a system has 2 GB of RAM installed, then Domino should be able to use up to that amount if necessary. This is not how it really works.

To understand how much memory Domino *can* use in a given OS requires understanding how operating systems allocate virtual memory. Each OS implements this a bit differently, but conceptually, they are all very similar.

Every operating system requires some memory dedicated for overhead, that is, hardware control, paging operations, and so on. They also need to make some memory available for applications. In order to keep the operating system safe from applications monopolizing memory, every modern OS divides virtual memory into two distinct areas, referred to as *kernel mode* and *user mode memory*.

Kernel mode	Memory reserved for use by the operating system itself

User mode Memory available to applications running on the OS

Note: When a 32-bit process, such as Domino, is started, it is allocated a 4 GB virtual memory space, regardless of the available virtual memory. The OS then handles how that virtual address space will be used. Some operating systems, like Windows and Linux, create the kernel mode and user mode memory areas from that single 4 GB virtual memory space. Other operating systems, such as AIX®, create a separate 4 GB virtual memory space for kernel mode memory, and therefore allow processes to use almost all of the 4 GB memory space that was allocated when the process started.

Within the user mode memory space, memory may be further divided into *shared* and *private* memory.

Shared memory Contiguous memory that can be referenced and utilized by more than one process.

Private memory Region of memory reserved solely for use by a single process.

Consideration: While a process can (and in Domino's case, usually will) use both private and shared memory, the sum of the two cannot go beyond the user memory limit imposed by the OS.

Knowing that each process is limited to a 4 GB virtual memory space, the *default* memory limits for various operating systems can be seen in Table 5-1.

Table 5-1 Default memory limits

Operating system	Default user address space limit	Shared memory limit	Private memory limit
Windows (32-bit)	2 GB	2 GB	2 GB
Linux	3-4 GB	3-4 GB	3-4 GB
AIX	3.75 GB	2.25 GB	512 MB
Solaris	3.9 GB	3.9 GB	3.9 GB
OS/390®	2 GB	2 GB	2 GB

The User mode address space limit represents the absolute maximum amount of Virtual Memory available to a Domino Server. If a server has more RAM than can be used as described above, then additional Domino Server partitions may be added to make better use of the available memory.

Attention: By default, Windows reserves 2 GB of virtual address space for kernel mode memory, and 2 GB to user mode memory. This setting may be changed using the /3GB switch in Windows' boot.ini. Using this setting tells the operating system to make 3 GB available for user memory, and only 1 GB for kernel memory.

In order for an application to take advantage of this option, it needs to be *large address aware*. Without this awareness, a 32-bit application will still only make use of 2 GB worth of its address space. Additionally, only 1 GB will have been allocated to the kernel, leaving 1 GB worth of address space completely unavailable for use at all.

Beginning with release 7.0.1, Domino has been made large address aware in Windows, and can therefore take advantage of more memory than previous Domino releases.

5.2 Domino specific performance tuning

Each Domino server should be tuned for the specific role it serves in the infrastructure. For this reason, suggestions on what to tune are separated into some of the different tasks a server might be assigned. For example, there would be little reason to configure a Replication Hub to optimize client connections, nor would it make sense, in most cases, to tune an Application Server for optimal mail routing performance.

5.2.1 Server tasks

Each task running on a Domino server requires additional resources. Care must be taken to remove unnecessary tasks that may adversely affect server performance. Minimizing the number of server tasks on the server, their frequency and specific execution times will enable you to optimize performance of the Domino server. For example, if an agent is run, which requires a large amount of system resources to execute, then running that agent after hours will help ensure better availability for users on that server during business hours.

The Lotus Domino server tasks to be automatically launched by the server are set using the following NOTES.INI settings:

```
ServerTasks
ServerTasksAtxx
```

Note: *xx* represents a number from 0 (representing midnight) through 23 (11:00 p.m.).

The ServerTasks line is used for tasks that will be started when the Domino server is started, while tasks defined in the ServertasksAtxx line are executed at the specified time. In the following example, the *Router, Replica, Update, AMgr, Adminp*, and *HTTP* tasks will be started with the server, while *Catalog* and *Design* will run at 1:00 a.m., and *Statlog* will run at 5:00 a.m.:

```
ServerTasks=Router,Replica,Update,Amgr,Adminp,HTTP
ServerTasksAt1=Catalog,Design
ServerTasksAt5=Statlog
```

The next few sections will supply useful information to help optimize the performance of individual Domino server tasks. The information may also serve to dispel some common misunderstandings regarding task specific performance as well.

Replica

The database replicator (replica) on a server is responsible for handling scheduled replication requests, as configured in server connection documents. Replication plays such a significant part in any multiple Domino server environment, that it is critical to optimize replication scheduling. Failure to do so can directly, and indirectly, affect performance on many other areas within Domino.

This section will focus on replication scheduling, which is a common area of misconfiguration. To better understand replication scheduling, there are some key points that need to be discussed

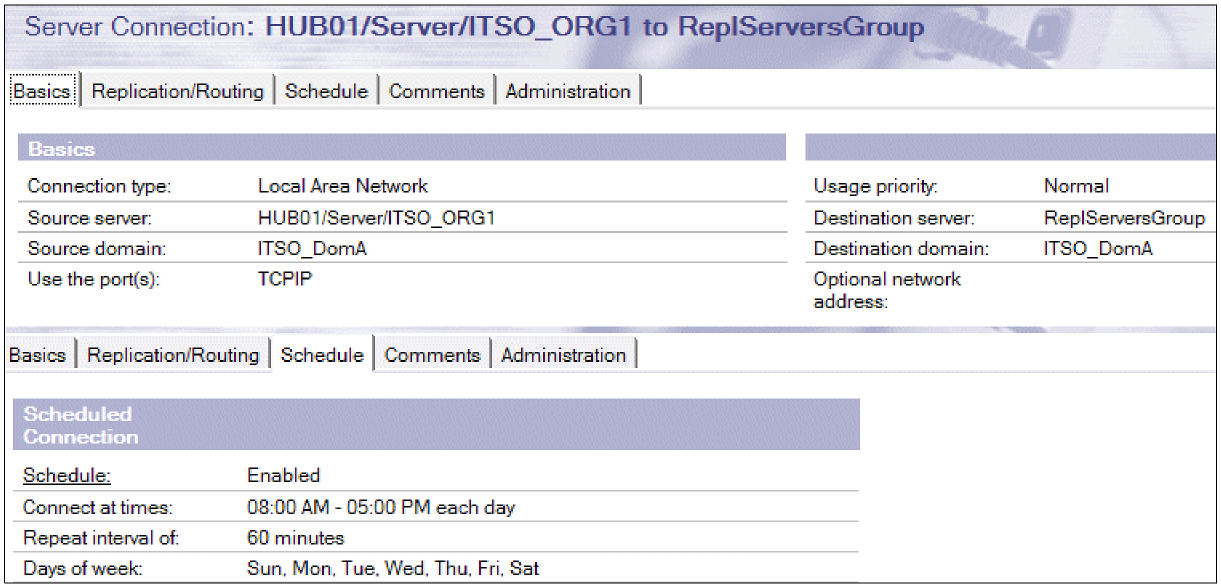
- Each replica task only replicates with one remote server at a time.
- Each replica task only replicates one database at a time.
- Repeat interval is *NOT* the period of time between the *start* of one replication cycle to the start of the next.

The first two points above simply mean that when scheduled replication on a server is occurring, the replica task performing the work can only replicate one database with one server at a time, even if multiple databases or a server group are defined in the connection document.

The third point is to dispel a common misunderstanding regarding the repeat interval in a connection document. The repeat interval is actually the time between the *end* of the specific scheduled replication job and the *start* of the next.

Scheduling replication with a misunderstanding of any of the three topics above can severely impact expectations on replication speed and reliability, while understanding them and applying them to replication schedules can help minimize problems.

In Figure 5-1, a connection document has been created to replicate All Databases from the hub server, HUB01/Server/ITSO_ORG1, to all servers in the server group called ReplServersGroup, from 8:00 a.m. to 5:00 p.m., repeating every 60 minutes.



Server Connection: **HUB01/Server/ITSO_ORG1 to ReplServersGroup**

Basics | Replication/Routing | Schedule | Comments | Administration

Basics

Connection type:	Local Area Network	Usage priority:	Normal
Source server:	HUB01/Server/ITSO_ORG1	Destination server:	ReplServersGroup
Source domain:	ITSO_DomA	Destination domain:	ITSO_DomA
Use the port(s):	TCPIP	Optional network address:	

Basics | Replication/Routing | Schedule | Comments | Administration

Scheduled Connection

Schedule:	Enabled
Connect at times:	08:00 AM - 05:00 PM each day
Repeat interval of:	60 minutes
Days of week:	Sun, Mon, Tue, Wed, Thu, Fri, Sat

Figure 5-1 Connection Document example

If the ReplServersGroup contains a list of 10 servers, and each of those servers have 50 database replicas in common with the initiating server, then this one connection document will be responsible (effectively) for 500 separate replications.

It is obviously easier to manage and maintain this one document as opposed to utilizing 10 server specific connection documents, or 500 individual connection documents. Domino administrators can, and do, take advantage of these sorts of connection documents to ease administrative overhead, and simplify scheduling, but there are ways to improve replication performance itself as well.

In a default Domino server configuration, only a single database replicator task is enabled. With the above example, this means that once the scheduled replication begins at 8:00 a.m., 500 individual replication iterations will be performed, serially, starting with all databases on the first server, then the second server, and so on. Depending on the amount of data being replicated, this can take a significant amount of time to complete.

For the purposes of this discussion, let us assume that it takes four hours to replicate all 500 of those databases. The single database replicator then waits for one hour (the repeat interval from the connection document) before replicating again. This means that at 1:00 p.m., the second replication of the day will begin for all 500 databases.

By the time the second replication occurrence completes, and the repeat interval is factored in again, the schedule times for this connection document will already have ended.

This example may be sufficient in some environments, but if data needs to be replicated more often than twice per day during business hours, and enough processing resources exist on a system, it is possible to make a significant improvement in replication times by adding additional replicator tasks. This can be either by adding additional instances of replica to the servertasks line or by using the replicators parameter in the notes.ini to define the number of database replicators to run. For example, adding the following to the server's notes.ini will load four replica tasks when the server starts:

```
replicators=4
```

As discussed earlier in this section, each database replicator can only replicate one database to a server at a time. Adding additional replica tasks allows one Domino server to replicate more than one database at a time, resulting in a significant improvement in replication time.

In the same example, increasing the number of replicators to four would allow the single connection document to initiate replication with four servers simultaneously. By distributing the replication across four separate tasks, what originally took four hours to complete may only take an hour. This would then allow those same 500 databases to replicate four times per day.

Tip: A good rule of thumb is to run one replicator per CPU on a server. If resources allow, additional replicators may be added if necessary.

AMgr

Consideration: Only one database replicator per server session will work. You cannot run multiple database replicators between a single pair of servers. In other words, if a spoke replicates with five other servers, then you can run five replicators in tandem if the resources are sufficient, but at no point will more than one replicator be allocated between the hub and one of the other five servers.

Agent manager (AMgr) controls when agents run on a server. Every time an agent runs, it uses server resources. The higher the number of agents, and the more complex those agents are, the higher the resource demands on the server are. Settings in the Server document and in the NOTES.INI control when scheduled and event-triggered agents run. Customizing when agents run may conserve server resources, but it may also delay when agents run.

For mail servers, we recommend running only those agents that ship with the standard mail and system database templates. Additional home-grown agents or third-party tools invoking agents must be evaluated separately for their impact on system resources. We strongly discourage giving users the ability to run their own mail agents because this essentially gives them a blank check for using system resources. To prohibit users from running private agents, set the following in the server's notes.ini:

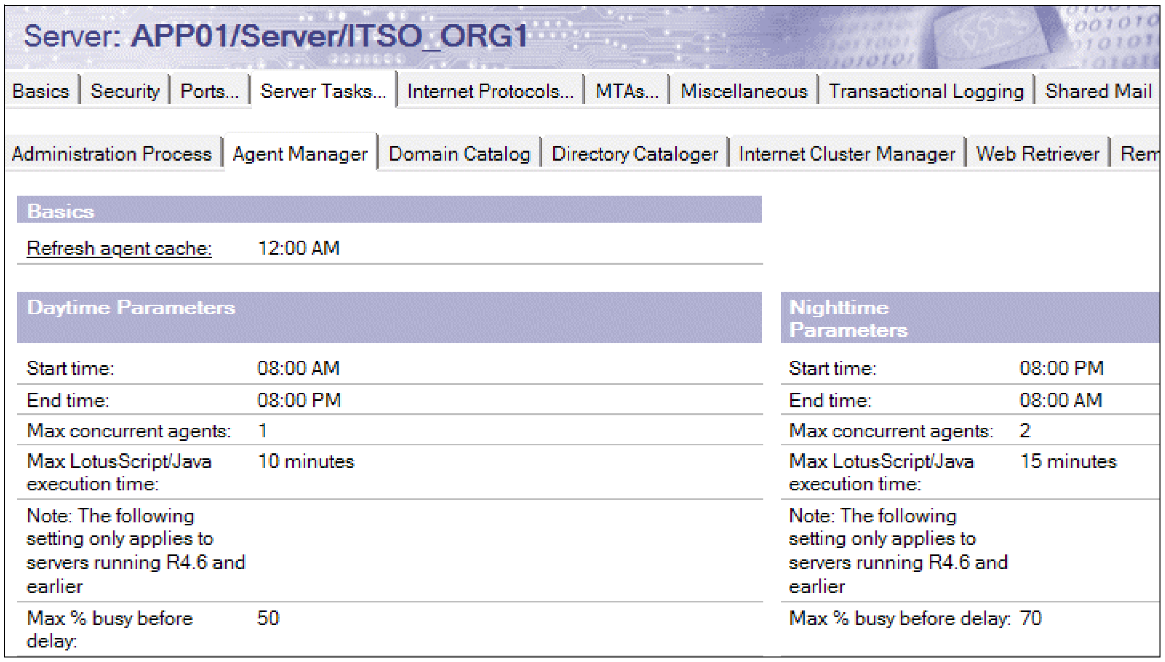
```
Enforce_Personal_Agents=1
```

On application servers, it may be more appropriate to have higher AMgr activity because agents are often an integral part of an application(s). For these environments, agents need to be designed for optimum efficiency, and monitored for changes in resource consumption triggered by updates to them or the database(s) they touch.

Concurrent agents

The agent manager task is actually comprised of one or more agent manager *executives* as well as a single Agent manager *controller*. Among other things, the controller is responsible for queuing eligible agents so that the agent manager executives can process them. This system allows Domino to run multiple agents concurrently to better utilize available system resources.

Each agent executive runs each concurrent agent. The number of agent manager executives on a server is defined in the server document under **Server Tasks** → **Agent Manager**. Figure 5-2 shows the fields relevant to AMgr's configuration.



Server: APP01/Server/ITSO_ORG1
Basics | Security | Ports... | Server Tasks... | Internet Protocols... | MTAs... | Miscellaneous | Transactional Logging | Shared Mail
Administration Process | Agent Manager | Domain Catalog | Directory Cataloger | Internet Cluster Manager | Web Retriever | Rem

Basics

Refresh agent cache:	12:00 AM

Daytime Parameters		Nighttime Parameters	
Start time:	08:00 AM	Start time:	08:00 PM
End time:	08:00 PM	End time:	08:00 AM
Max concurrent agents:	1	Max concurrent agents:	2
Max LotusScript/Java execution time:	10 minutes	Max LotusScript/Java execution time:	15 minutes
Note: The following setting only applies to servers running R4.6 and earlier		Note: The following setting only applies to servers running R4.6 and earlier	
Max % busy before delay:	50	Max % busy before delay:	70

Figure 5-2 Server document showing fields relevant to AMgr's configuration

As illustrated in Figure 5-2, the server document provides separate options for both daytime and nighttime configurations. To maximize the server's responsiveness for user transactions, be sure to keep the daytime settings as low as possible (1 is the default). At nighttime, when user responsiveness is less critical, increasing the number of concurrent agents will allow more agent activity to occur.

To see a snapshot of the Agent Manager status, including the number of Agent Executives currently running, enter the `tell amgr status` command at the server console. To see a list of scheduled agents, enter the `tell amgr schedule` command at the server console.

Tip: For help interpreting the output from the `tell amgr schedule` command, see Lotus Software Knowledge Base Document #1097473 at:

`http://www.ibm.com/support/docview.wss?rs=899&uid=swg21097473`

Controlling how often Agent Manager runs agents

These NOTES.INI settings affect how often the Agent Manager executes agents. In general, the more frequently agents run, the sooner they perform their tasks. Running agents more frequently, however, may increase demand on server resources and adversely affect overall system performance.

AMgr_DocUpdateAgentMinInterval

This setting specifies the minimum elapsed time, in minutes, between executions of the same document update-triggered agent. This lets you control the time interval between executions of a given agent. The default is 30 minutes. A longer interval can result in the agent running less often, reducing server demand. If document update events are infrequent, you can reduce the delay.

AMgr_DocUpdateEventDelay

This setting specifies the delay time, in minutes, the Agent Manager schedules a document update-triggered agent after a document update event. The default is five minutes. The delay time ensures that the agent runs no more often than the specified interval, regardless of how frequently document update events occur. When the agent executes, it will also process all additional events (if any) that occurred during the interval. A longer interval results in the agent running less often, thus reducing demand for server time. If document update events are infrequent, however, you can reduce the delay to ensure the agent runs soon after the event occurs.

AMgr_NewMailAgentMinInterval

This setting specifies the minimum elapsed time, in minutes, between execution of the same new mail-triggered agent. The default is 0 (no interval between executions). Similar to AMgr_DocUpdateAgentMinInterval, entering an interval can result in the agent running less frequently.

AMgr_NewMailEventDelay

This setting specifies the time (in minutes) that the Agent Manager delays before scheduling a new mail-triggered agent after new mail is delivered. The default is one minute. Similar to AMgr_DocUpdateEventDelay, the delay time ensures the agent runs no more often than the specified interval. When the agent executes, it will also process all additional events (if any) that occurred during the interval. A longer interval results in the agent running less often, thus reducing demand for server time. If document update events are infrequent, however, you can reduce the delay to ensure the agent runs soon after the event occurs.

Controlling how quickly the Agent Manager queues agents

The Agent Manager periodically checks to see if it has any new agents that it needs to schedule. These NOTES.INI settings control how quickly an agent gets into the schedule queue.

AMgr_SchedulingInterval

This setting specifies a delay (in minutes) between running of the Agent Manager's scheduler. Valid values are one minute to 60 minutes. The default value is one minute.

AMgr_UntriggeredMailInterval

This setting specifies a delay (in minutes) between running of the Agent Manager's check for untriggered mail. Valid values are one minute to 1440 minutes (the number of minutes in a day). The default value is 60 minutes.

HTTP

Domino's HTTP task reserves significant resources as soon as it is started, so disable it if Web access is not needed. If the HTTP server is needed, then determine whether or not HTTP server logging is needed. Disabling both text and Domlog.nsf logging types will improve Domino Web server performance.

HTTP logging options are enabled under **Internet Protocols** → **HTTP** → **Enable Logging to** in the server document. Both logging types are disabled by default.

Domino memory cache

Mapping information about databases and authenticating users can take time. To optimize response time, Domino uses several cache settings for Domino HTTP servers that include:

- Images and files
- Commands
- Design elements
- Authenticated users

The design and functionality of your Web site dictate the optimal web cache settings. By evaluating the statistics returned by the following command, changes may be warranted to improve performance:

```
show stat domino.cache
```

Table 5-2 provides definitions for the available Domino.Cache statistics.

Table 5-2 Definitions for the available Domino.Cache statistics

Statistic	Definition
Domino.Cache.Command.Count	Number of commands that are contained in the HTTP Command Cache.
Domino.Cache.Command.MaxSize	Maximum number of commands that can be cached as defined in the *Maximum cached commands* field on the Server Document under **Internet Protocols** → **HTTP** → **R5 Basics**.
Domino.Cache.Design.Count	The actual number of cached design elements.
Domino.Cache.Design.DisplaceRate	The percentage-ratio of the number of times a new entry added to the design cache displaces an aged design entry compared to the number of times the cache was investigated for a cache entry.
Domino.Cache.Design.HitRate	The percentage-ratio of the number of times valid entries in the design cache are used compared to the number of times the design cache was investigated for a cache entry.
Domino.Cache.Design.MaxSize	The maximum number of design elements that can be cached, as defined in the Maximum cached designs field in the Server document under Internet Protocols → Domino Web Engne → Memory Caches.
Domino.Cache.iNote WA Forms file.Count	The number of forms files in use for Domino Web Access since the server started.

Statistic	Definition
Domino.Cache.Session Cache.Count	Peak number of open HTTP sessions
Domino.Cache.Session Cache.MaxSize	Maximum number of HTTP sessions allowed on the server at the same time as defined by the Maximum active sessions field in the Server Document under Internet Protocols → Domino Web Engine → HTTP Sessions.
Domino.Cache.User Cache.Count	Number of authenticated user credentials stored in the Domino user cache.

As a rule of thumb, any of the Domino.Cache.*.Count statistics should be slightly less than their Domino.Cache.*.MaxSize counterpart. If the Count matches the MaxSize, then try slowly incrementing the Maximum value for that cache until the Count no longer reaches the maximum:

```
Domino.Cache.Design.Count = 128
Domino.Cache.Design.MaxSize = 128
Domino.Cache.User.Count = 61
Domino.Cache.User.MaxSize = 64
```

The statistics above indicate that the design cache should be increased to optimize the Domino memory cache, while the number of cached users already appears to be set correctly.

To manage memory cache on a Web server, complete the fields referenced in Table 5-3. These fields are located in the server document under **Internet Protocols** → **Domino Web Engine** → **Memory Caches**.

Table 5-3 Fields for managing memory cache on a Web server

Field	Action
Maximum cached designs	Enter the number of database design elements to cache for users. The default is 128. When a user opens a database, Domino maps each design element name to an identification number. This mapping procedure takes time. Use this field to specify how many elements you want to store in memory so the next time a user accesses that element, it is immediately available.
Maximum cached users	Enter the number of users to cache. The default is 64. After a user successfully authenticates with a server, Domino stores in memory the user's name, password, and the list of groups to which the user belongs. Use this field to increase the number of users for whom Domino stores this information.

Field	Action
Cached user expiration interval	Enter the time interval in seconds during which Domino regularly removes user names, passwords, and group memberships from the cache. The default is 120. Remove user names, passwords, and group memberships from the cache periodically to force Domino to look up credentials in the directory the next time those users access the server.

Specify session time outs

Open, inactive sessions can prevent users from accessing the server. Imposing time limits for activities between the Domino Web server and Web clients will prevent idle connections from becoming a problem.

Modify the fields, listed in Table 5-4, which are found in the Timeouts section of the server document under **Internet Protocols** → **HTTP.**

Table 5-4 Time out settings

Field	Action
Request time out	Specify the amount of time for the server to wait to receive an entire request. The default is 60 seconds. If the server does not receive the entire request in the specified time interval, the server terminates the connection.
Input time out	Enter the time, in seconds, that a client has to send a request after connecting to the server. The default is 15 seconds. If no request is sent in the specified time interval, then the server terminates the connection. If only a partial request is sent, the input timer is reset to the specified time limit in anticipation of the rest of the data arriving.
Output time out	Enter the maximum time, in seconds, that the server has to send output to a client. The default is 180 seconds.
CGI time out	The maximum time, in seconds, that a CGI program started by the server has to finish. The default is 180 seconds.

The Timeouts section also includes the field HTTP persistent connection, which controls whether or not Domino allows persistent connections over HTTP with the server. Persistent connections require more server overhead than connections that are limited by network activity. Table 5-5 explains the settings relevant to persistent connections.

Table 5-5 Persistent connections settings

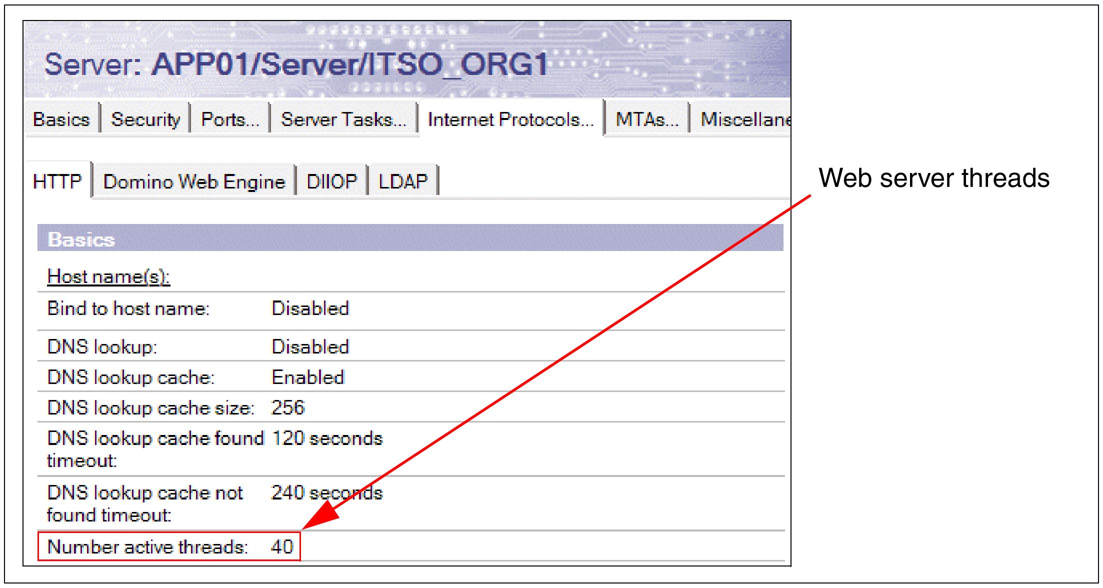
Field	Action
Maximum requests per persistent connection	Specify the maximum number of HTTP requests that can be handled on one persistent connection. The default is 5.
Persistent connection time out	Specify the length of time for which you want persistent connections to remain active. The default is 180 seconds.

Specifying the number of threads used by the Web server

An HTTP request is processed by a thread. A thread, in turn, can handle a number of network connections. You can specify the number of threads the Web server can process. In general, the number of threads specified is an indication of the number of users who can access the server simultaneously.

If the number of active threads is reached, the Domino server queues new requests until another request finishes and threads become available. The more power your machine has, the higher the number of threads you should specify. If your machine spends too much time on overhead tasks, such as swapping memory, specify a lower number of threads.

Thread options appear under the HTTP sub-tab in the Internet Protocols tab in the Basics section of the Server document, as shown in Figure 5-3.


Server: APP01/Server/ITSO_ORG1
Basics | Security | Ports... | Server Tasks... | Internet Protocols... | MTAs... | Miscellane
HTTP | Domino Web Engine | DIIOP | LDAP
Basics
Web server threads

Host name(s):	
Bind to host name:	Disabled
DNS lookup:	Disabled
DNS lookup cache:	Enabled
DNS lookup cache size:	256
DNS lookup cache found timeout:	120 seconds
DNS lookup cache not found timeout:	240 seconds
Number active threads:	40

Figure 5-3 HTTP thread options

Restricting the amount of data Web users can send

The HTTP POST and PUT methods enable users to send data to the Domino server. The Server record field Maximum size of request content sets a limit on the amount of data that can be sent using either POST or PUT. This limit is enforced for all POST and PUT methods, whether the target is a database, CGI program, or Java™ servlet, and applies to all Web sites.

The Web Site document contains two additional settings that control POST and PUT methods that target a database (for example, filling in a form or uploading a file attachment). Because these settings are located in Web Site documents, it is possible to specify different values for each Web site.

To restrict the amount of data that can be sent to a Domino database:

1. From the Domino Administrator, click the **Configuration** tab, expand the Web section, and click **Internet Sites**.
2. Choose the Web Site document you want to edit, and click **Edit Document**.
3. Click the **Domino Web Engine tab**. Under POST Data, complete the fields shown in Table 5-6.

Table 5-6 POST Data fields

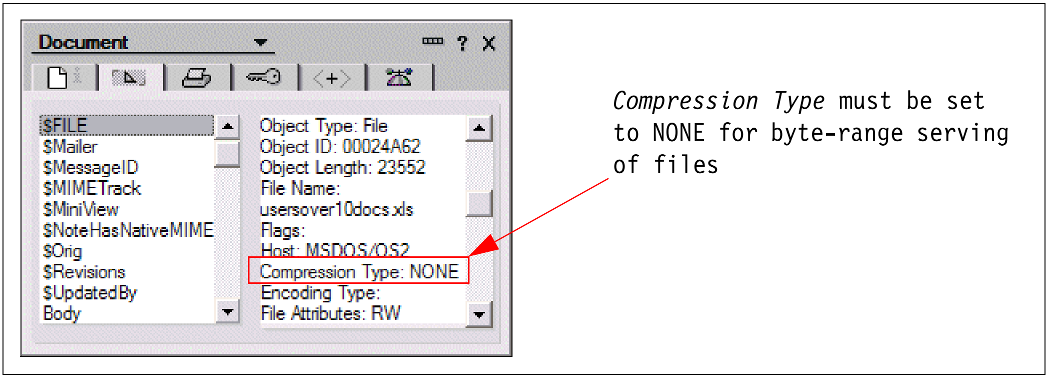
Field	**Action**
Maximum POST data	Enter the amount of data in KB that a user is allowed to send to the Web site in a POST request that targets a database. The default is 0, which does not restrict the amount of data that users can send (however, the amount is still limited by the Server record setting Maximum request content). This limit applies to both the PUT and the POST HTTP methods. If users try to send more than the maximum allowed data, Domino returns an error message to the browser.
File compression on upload	Choose one: ► Enabled: To compress files before adding them to a database. Compressing files saves disk space on the server. ► Disabled (default): If clients use a browser that supports byte-range serving. Note: Compressed files cannot be downloaded using Domino byte-range serving.

Improving file download performance for Web clients

Web clients can download a file that is attached to a page or that is in a server directory that is mapped by a URL. If the Web client supports byte-range serving (available in HTTP 1.1 and later), the client downloads the file in sections (ranges of bytes) and tracks the progress of each file download. If an interruption occurs, the client can resume the download from the point where it was interrupted. Without byte-range serving, users must repeat interrupted downloads from the beginning.

Domino is compatible with clients that support the HTTP 1.1 specification. The clients can be implemented in a variety of ways, for example, as browser plug-ins, applets, or stand-alone programs. Domino automatically uses byte-range serving if the Web client uses a product that supports this feature. No configuration is necessary.

However, attached files must be decompressed in order for clients that support byte-range serving to access them. When you attach a file, you must clear the Compress option. To verify that an existing attachment is decompressed, choose **File** → **Document Properties**, select the **$FILE** item from the Design tab, and verify that the Compression Type property is NONE, as shown in Figure 5-4.



Object Type: File
Object ID: 00024A62
Object Length: 23552
File Name:
usersover10docs.xls
Flags:
Host: MSDOS/OS2
Compression Type: NONE
Encoding Type:
File Attributes: RW

Compression Type must be set to NONE for byte-range serving of files

Figure 5-4 Compression Type Setting

Update

One of the most common, preventable causes of poor response time and performance of Domino servers is excessive and unnecessary activity of the Domino Update task. This task is responsible for updating and rebuilding the indexes of Domino database views and full text indexes. It is designed to run in the background and is intended to improve response time and performance by ensuring that when a user opens a database view, the user does not have to wait for it to be indexed. Ironically, it often turns out to have the opposite effect.

This section explains how the Update task is supposed to work. It looks at how Domino administrators and designers can monitor and control it to improve server performance. It also explains the potential impact to performance when the UPDATE task is running excessively. It provides a hint of the kind of improvements you might see by correcting the problem.

The Update task, also known as the indexer, is responsible for automatic updates of both view and full text indexes. It is loaded at server startup, by default, and runs continuously, checking its work queue for views and folders that require updating.

Update maintains two work queues: an immediate queue and a deferred queue. Deferred update requests are held for 15 minutes before they are processed so that requests to update the same database in that time are ignored as duplicate requests. Updated queues can grow if the server has a high update rate because of heavy database use.

Important: Immediate updates do not necessarily occur immediately. They are simply added to the Pending queue where they are processed as soon as an indexer thread is available to process them.

By default, the indexer waits five seconds between each database update operation that it performs. When a request is made to update a view, the view is only updated if there are at least twenty note changes since the last update, and if the view has been accessed within the last seven days.

Tip: Domino 7 allows a separate thread to be created solely for full text updates, if needed. This separate thread will only be responsible for updating full text indexes, so view updates will continue to occur even if a large full text index is being rebuilt. To enable the full text indexer-only thread, add the following to the notes.ini:

```
UPDATE_FULLTEXT_THREAD=1
```

Note: A separate *directory indexer* thread is automatically spawned by update to handle view updates of any local or remote Domino Directory or Extended Directory Catalog that a server uses for directory services. This directory indexer thread runs every minute and does not lock the views *during updates*, permitting new server sessions while this task is running.

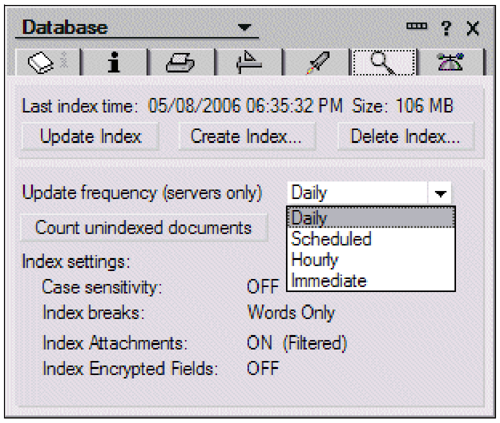
Important: If the views are being rebuilt (**`updall -R`**), then the views will be locked, preventing authentication until the rebuild is completed. For this reason, it is important to defer view rebuilds in the Domino Directory until after peak hours.

Full text indexes

Full text indexing options and update frequency can affect server disk space and processing speed, so they deserve special consideration to ensure they perform adequately without hurting overall server performance.

You must periodically update full text indexes on servers to keep them synchronized with changes to the databases. When you create an index, you can either accept the default schedule for updating it (nightly at 2 a.m.) or specify a different schedule. This schedule may be changed at any time via database properties.

Figure 5-5 shows the different update frequencies as well as the full text settings for a particular database. Table 5-7 on page 66 and Table 5-8 on page 66 explain the indexing options and update frequencies, respectively, available for full text indexes.



Last index time: 05/08/2006 06:35:32 PM Size: 106 MB

Update frequency (servers only) Daily

Index settings:

Case sensitivity:	OFF
Index breaks:	Words Only
Index Attachments:	ON (Filtered)
Index Encrypted Fields:	OFF

Figure 5-5 Update frequency

Table 5-7 Indexing options

Indexing option	Description
Index attached files	Indexes attachments. Also choose either **With found text** to include just the ASCII text of attachments, or **With file filters** to include the full binary content of attachments. Note: Choosing **With found text** creates the index faster than choosing **With file filters**, but is less comprehensive.
Index encrypted fields	Indexes text in encrypted fields.
Index sentence and paragraph breaks	Includes sentence and paragraph breaks in addition to word breaks to allow users to do proximity searches.
Enable case sensitive searches	Allows searches by exact case match. Note: This option increases the size of the index by about 15%, as each word must be indexed twice, for example, apple and Apple.

Table 5-8 Update frequencies

Update frequency option	Updates occur
Daily	Nightly when the Updall server program runs by default at 2 a.m.
Hourly	Every hour, as scheduled by the Chronos server task. Note: Can be disabled using the setting Debug_Disable_Chronos=1 in the notes.ini.
Immediate	As soon as possible after you close the database.
Scheduled	As scheduled by a Program document for the Updall server task in the Domino Directory. Note: If you select the Scheduled option and do not create a Program document for Updall, scheduled updates do not occur.

Monitoring update performance

If a server has a high update rate due to heavy application database use, a large number of mail users, or a large volume of mail, the default resource usage configuration can cause the updater queues to become large, typically indicating that views and full text indexes are not up to date.

To determine whether the updater queues are large, examine the queue length statistics that are available in Lotus Domino 7 using the following command on the Domino server console:

```
show stat update
```

Table 5-9 lists the available statistics and their definitions.

Table 5-9 Available statistics and their definitions

Statistic	Definition
Update.DeferredList	Number of requests for view updating or full text indexing on the deferred queue
Update.DeferredList.Duplicates	Number of requests for view updating or full text indexing avoided because they were already waiting on the deferred queue
Update.DeferredList.Max	Maximum number of requests waiting for view updating or full text indexing on the deferred queue
Update.DeferredList.Processed.AllViews	Number of all view updates processed from the deferred queue
Update.DeferredList.Processed.Compactions	Number of compactions processed from the deferred queue
Update.DeferredList.Processed.SingleViews	Number of single view updates processed from the deferred queue
Update.FullTextList	Number of requests on the full text index queue
Update.FullTextList.Duplicates	Number of requests for full text indexing avoided because they were waiting on the full text index queue
Update.FullTextList.Max	Maximum number of requests waiting for full text indexing
Update.FullTextList.Processed	Number of full text indexing requests processed
Update.NAB.Updates	Number of Domino Directory view updates processed
Update.PendingList	Number of requests for view updating or full text indexing on the immediate queue
Update.PendingList.Max	Maximum number of requests waiting for view updating or full text indexing on the immediate queue
Update.PendingList.Processed.AllViews	Number of all view updates processed from the immediate queue
Update.PendingList.Processed.Compactions	Number of compactions processed from the immediate queue
Update.PendingList.Processed.SingleView	Number of single view updates processed from the immediate queue

Attention: Since deferred updates are delayed 15 minutes by design, it is important to monitor the Update.Deferred* statistics over time, instead of acting on a single set of data.

If either the Update.FullTextList or Update.PendingList statistics show more than just a few updates in the queue, then updates may be backing up in the queue. For example, the following statistic output indicates an indexer with no real backlog:

```
Update.PendingList = 8
```

while this statistic definitely indicates that updates are backing up in the queue:

```
Update.PendingList = 2738
```

If this happens, it is possible to tune the indexer's performance by using the following notes.ini settings.

- `Update_Access_Frequency`

  This setting, in days, will allow updates to occur against views that have been accessed within the number of days specified here as opposed to the default of 7. Lowering this number will reduce the frequency of updates to rarely accessed views, allowing faster updates to more frequently used views instead.

- `Update_Note_Minimum`

  This setting controls how many changes must exist for a view update to occur. The default is 20. Lowering this number will force more frequent updates of views to occur.

- `Update_Suppression_Time`

  Use this variable to specify the length of time that deferred update requests are held in queue before being serviced. The default is 15 minutes. Lowering this value will cause deferred updates to occur more frequently.

- `Update_Suppression_Limit`

  Use this variable to limit the size (number of requests) in the Update task's deferred requests queue. The default is 32767.

- `Update_Idle_Time`

  This setting, in seconds, changes the amount of time the indexer waits between database operations. The default is five seconds. Lowering this will allow the indexer to process updates more frequently, and as a result will consume more resources. If it is necessary to get more granular, `Update_Idle_Time_MS` may be used to make changes in milliseconds.

Note: FTUpdate_Idle_Time and FTUpdate_Idle_Time_MS may be used to separately control the delay between operations performed on full text operations if the separate full text thread is being used (`UPDATE_FULLTEXT_THREAD=1`).

Additionally, if the system's resources allow it, additional Update tasks may be started on the server, by adding an `Updaters` line in the notes.ini. We recommend that a maximum of one Update task be run per CPU. So, to enable four Update tasks on a Domino server, add the following to the notes.ini:

```
Updaters=4
```

In some situations, it may make sense to disable automatic view or full text updates completely, for example, if fast access to data is more important than the accuracy of the full text index. In these cases, it is possible to disable automatic view and full text indexing if needed by adding settings to the server's notes.ini:

- `UPDATE_DISABLE_VIEWS=1`

  Disables view updates on the server. With this setting in place, views will be updated automatically when a view is accessed, which may significantly increase the time to open a database or view.

- `UPDATE_DISABLE_FULLTEXT=1`

  Disables full text updates on the server.

With either of the above in place, a program document could be used to run Updall in order to update the indexes for specific databases, if needed.

Updall

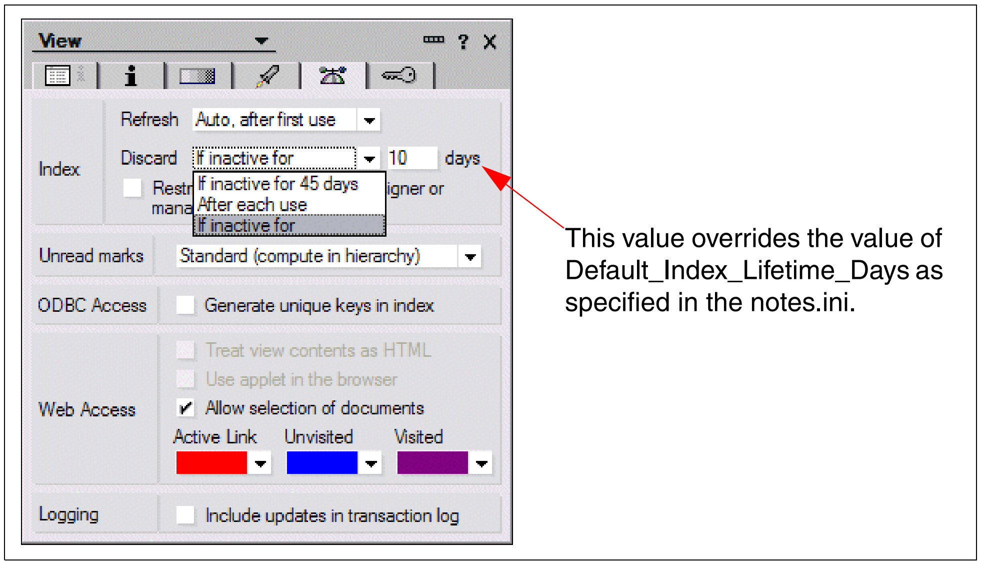
Updall performs the same basic functions as the Update task, but is run on an as-needed basis, as scheduled using program documents, or by the ServerTasksAtxx line in the server's notes.ini. Unlike update, Updall accepts arguments to control its behavior.

Tip: A complete list of available options and their behavior is available in the Domino Administrator Help 7 database in the document titled *Updall options.*

Updall also purges deletion stubs from databases and discards view indexes for views that have been unused for 45 days, by default. To change when Updall discards unused view indexes when no criteria has been defined by the database designer, add the following to the server's notes.ini:

```
Default_Index_Lifetime_Days=number of days
```

The default value for this variable is 45 days unless a database designer has specified a different lifetime by selecting **Design** → **View Properties** → **Advanced Tab** → **Discard index** to reach the dialog box, as shown in Figure 5-6.



View

Index: Refresh: Auto, after first use

Discard: If inactive for 10 days

If inactive for 45 days / After each use / If inactive for

Unread marks: Standard (compute in hierarchy)

ODBC Access: Generate unique keys in index

Web Access: Treat view contents as HTML; Use applet in the browser; ✓ Allow selection of documents

Active Link 🟥 | Unvisited 🟦 | Visited 🟪

Logging: Include updates in transaction log

This value overrides the value of Default_Index_Lifetime_Days as specified in the notes.ini.

Figure 5-6 Default index lifetime

Lowering the number of days to a lower value can reduce the number of view indexes on a server and therefore reduce the amount of time and effort that the Indexer server task must spend on updating view indexes.

It is the responsibility of the Updall task to remove the view index if a discard option is specified. In other words, if a Discard Index option is selected for a view, the index is not actually discarded immediately. Rather, the index is removed the next time the Updall tasks runs. For example, a view has a Discard View option of "After each use". If you exit from a view at 1:00 p.m., the view will not be removed until Updall is run (usually done at 2:00 a.m.).

Directory Cataloger (Dircat)

Here we discuss the Directory Cataloger (Dircat).

Directory Catalogs

A Directory Catalog is an optional directory database that aggregates information from multiple Domino Directories. There are two types of Directory Catalogs: Extended Directory Catalogs and condensed Directory Catalogs. An Extended Directory Catalog is larger than a condensed Directory Catalog, but is the recommended Directory Catalog for server use because it allows faster and more flexible directory lookups and uses less CPU.

The Extended Directory Catalog uses the same design as the Domino Directory, so it includes multiple views that sort names in different ways. Regardless of the format of a name, there is a view in the Extended Directory Catalog that a server can use to quickly find the name. A condensed Directory Catalog has one view used for lookups, which you choose how to sort when you configure it. To look up a name in a condensed Directory Catalog that does not correspond to the selected sort order, the server uses the full text index to search for the name, which takes longer than a view search.

Using an Extended Directory Catalog on servers that route mail is a particular advantage, because a mail server can use views to quickly find an address regardless of the address format. When a mail server uses a condensed Directory Catalog, mail routing can back up if the Router uses the full text index to look up addresses, for example, some Internet addresses, that do not correspond to the selected sort order.

Directory Cataloger

The Dircat task (Directory Cataloger) is the server task that initially aggregates information from source Domino Directories into a Directory Catalog, and then continues to run at scheduled intervals to update the Directory Catalog to reflect changes to the source Domino Directories, or to the Directory Catalog configuration. The Dircat task aggregates both condensed Directory Catalogs and Extended Directory Catalogs.

A server that runs the Dircat task (a Dircat server) should:

- Have enough disk space to store local replicas of the source Domino Directories that are aggregated, if you choose to store the directories locally on the server, rather than have the server access them over the network.
- Have enough disk space to store the resulting aggregated Directory Catalog(s) and full text indexes. Only condensed Directory Catalogs have full text indexes by default.
- Be able to replicate the Directory Catalog(s) it aggregates to any servers and clients that will use them.

Typically, it is best to run the Dircat task to build and maintain a Directory Catalog on a server in one domain, and then replicate the Directory Catalog to servers throughout an organization that need to use the Directory Catalog. This is beneficial because only one server then does the CPU-intensive Dircat processing of the Directory Catalog. Aggregate the primary Domino Directory of the domain in which you build the Directory Catalog so that servers in other domains can use the Directory Catalog to look up information from the directory.

The Dircat task on one server can process more than one Directory Catalog. The Dircat task is single-threaded, so it processes Directory Catalogs sequentially rather than simultaneously. Because Dircat is a CPU-intensive task, it is often beneficial to dedicate one server solely to Dircat processing.

Dircat can either update or rebuild Directory Catalogs in addition to creating them initially. An update should be relatively quick, while rebuilds (full and partial) require more resources to complete.

Updates to Directory Catalogs

When Dircat updates a Directory Catalog, it checks for changes to the contents of fields in the source Domino Directories, and then makes the appropriate changes to the Directory Catalog. This is the most common activity performed by Dircat.

Partial rebuilds

During a partial rebuild, the Directory Cataloger compares all documents in the source Directories with the contents of the Directory Catalog, looking for any changes that may have been made.

In most cases, a partial rebuild is performed by Dircat when Fixup makes some sort of change to one of the source Domino Directories. This could happen if a new OS copy of one the Directories replaces the existing one, or if Fixup deleted some documents in an existing source replica.

Full rebuilds

In a full rebuild, the Directory Cataloger re-aggregates of all the configured source Domino Directories, similar to what occurs during the initial build of the Directory Catalog. After a full rebuild, there must also be a full replication of the Directory Catalog to the servers and clients that use it, which can be time consuming, especially for replication of condensed Directory Catalogs.

The Dircat task must do a full rebuild of the Directory Catalog to incorporate changes into the Directory Catalog when any of the following fields in the Directory Catalog configuration document are made:

- Directories to include
- Additional fields to include
- Sort by (condensed Directory Catalog)
- Use Soundex (condensed Directory Catalog)
- Remove duplicate users
- Group types
- Include Mail-in Databases
- Include Servers (Extended Directory Catalog)
- Selection Formula

Note: Changes to the "Directories to include" field in a configuration document for an Extended Directory Catalog causes only a partial rebuild.

Tip: Plan which elements to include in the Directory Catalog from the start to minimize changes to the configuration document, since changes, after the fact, may result in a full rebuild being required.

Router

Domino mail servers use a MAIL.BOX database to hold messages that are in transit. Mail clients and other servers use SMTP or Notes routing protocols to deposit messages into MAIL.BOX. The Router on each server checks the address of each message in MAIL.BOX and either delivers the message to a local mail file or transfers it to the MAIL.BOX database on another server.

Controlling message delivery

You set delivery controls in the Configuration Settings document in the Router/SMTP → Restrictions and Controls → Delivery Controls tab, under Delivery Controls.

- Maximum delivery threads

  This setting determines the maximum number of threads the router can create to perform local mail delivery. Increasing this value can improve message throughput for local deliveries. The ideal number ranges from 3 to 25. This is determined by a formula, based upon the NSFBufferPoolSize. You can increase or decrease the value based on the server configuration Monitor Mail.Waiting over a period of time. If there is a backlog over a period of time, increase the number Monitor Mail.Delivery.Threads.Total. If the value is less than Mail.Delivery.Threads.Max, set the value to the total.

- Setting transfer limits

  You set transfer limits in the Configuration Settings document on the Router/SMTP → Restrictions and Controls → Transfer Controls tab, under Transfer Controls.

- Maximum concurrent transfer threads

  This setting determines the maximum number of concurrent transfer threads per destination. The default is the value entered for Maximum transfer threads divided by two.

- Maximum transfer threads

  This setting determines the maximum number of threads the mail Router can create to perform mail transfers. Without this variable, the default is one thread per server port. Increasing this number creates more threads to handle mail transfers. However, additional threads may increase the demand for server processing time.

Multiple mail boxes

Server processes, including server threads and the router, that write to MAIL.BOX require exclusive access to it. To ensure exclusive access, processes that write to or read from MAIL.BOX lock the database to prevent simultaneous access by other processes. Other processes trying to access the database must wait until the currently active process completes and unlocks the database before they can complete.

In most cases, a mail process locks MAIL.BOX for only an instant. However, longer wait times occur when the router or another process reads or writes a large message. When there is a large amount of new mail, for example, on a busy system with heavy mail traffic, several server threads may try to deposit mail into MAIL.BOX while the router attempts to read and update mail. Under heavy loads, such contention for a single MAIL.BOX database degrades performance.

To remove the contention for MAIL.BOX, enable multiple MAIL.BOX databases. This allows multiple concurrent processes to act on messages, and increases server throughput. While reading one MAIL.BOX, the router marks the database "in use" so other server threads trying to deposit mail move to the next MAIL.BOX. As a further benefit, having multiple MAIL.BOX databases provides failover in the event that one MAIL.BOX becomes corrupted.

When creating additional MAIL.BOX databases, consider placing each one on a separate disk. Because disk contention is rarely an issue for MAIL.BOX, placing each additional

MAIL.BOX database on a different disk will not improve performance *per se*. However, distributing the databases across multiple disks does ensure greater availability in the event of a disk failure.

Creating a second MAIL.BOX database offers a large performance improvement over using a single MAIL.BOX. Depending on server mail traffic, adding a third and fourth MAIL.BOX database may further improve performance. However, the improvement gained with each additional MAIL.BOX is increasingly smaller.

To create multiple MAIL.BOX databases, perform the following:

1. Make sure you already have a Configuration Settings document for the server(s) to be configured.
2. From the Domino Administrator, click the **Configuration** tab, and expand the **Messaging** section.
3. Click **Configurations**.
4. Select the **Configuration Settings** document for the mail server or servers you want to administer, and click **Edit Configuration**.
5. Click the **Router/SMTP - Basics** tab.
6. Complete the Number of mailboxes field (maximum of 10) and then click **Save** and then **Close**.

Once completed, the server will need to be restarted for the changes to take effect.

Once multiple MAIL.BOX databases have been enabled, new statistics become available that can be used to determine the optimal number of mailboxes for the server, as shown in Table 5-10.

Table 5-10 MAIL.BOX statistics

Statistic name	Description
Mail.Mailbox.Accesses	Total number of times that threads accessed any mailbox on the server.
Mail.Mailbox.AccessConflicts	The number of times that a thread attempting to access a mailbox had to wait because the number of concurrent threads exceeded the number of mailboxes configured. For example, if there are three mailboxes configured, and there are four concurrent accesses, the conflict count would be incremented. If the number of access conflicts consistently exceeds two percent of the value of Mail.Mailbox.Accesses, consider creating an additional mailbox.
Mail.Mailbox.CurrentAccesses	The total number of current accesses (for example. a count of two would indicate that two threads are accessing mailbox at this time).

Statistic name	Description
Mail.Mailbox.AccessWarnings	The number of times that the number of threads accessing the mailbox (that is, the value of Mail.Mailbox.CurrentAccesses) reached one less than the number of configured mailboxes. For example, the warning count is incremented when two threads attempt to access MAIL.BOX concurrently and there are three mailboxes configured. If the number of warnings consistently exceeds ten percent of the value of Mail.Mailbox.Accesses, consider creating an additional mailbox.
Mail.Mailbox.MaxConcurrentAccesses	The highest number of current accesses recorded.
Mail.Mailbox.Accesses	Total number of times that threads accessed any mailbox on the server.
Mail.Mailbox.AccessConflicts	The number of times that a thread attempting to access a mailbox had to wait because the number of concurrent threads exceeded the number of mailboxes configured. For example, if there are three mailboxes configured, and there are four concurrent accesses, the conflict count would be incremented. If the number of access conflicts consistently exceeds two percent of the value of Mail.Mailbox.Accesses, consider creating an additional mailbox.
Mail.Mailbox.CurrentAccesses	The total number of current accesses (for example. a count of two would indicate that two threads are accessing mailbox at this time.
Mail.Mailbox.AccessWarnings	The number of times that the number of threads accessing the mailbox (that is, the value of Mail.Mailbox.CurrentAccesses) reached one less than the number of configured mailboxes. For example, the warning count is incremented when two threads attempt to access MAIL.BOX concurrently and there are three mailboxes configured. If the number of warnings consistently exceeds ten percent of the value of Mail.Mailbox.Accesses, consider creating an additional mailbox.
Mail.Mailbox.MaxConcurrentAccesses	The highest number of current accesses recorded.

By calculating the number of access conflicts as a percentage of total accesses, you can determine whether a server will benefit from the addition of another MAIL.BOX. In general, the number of access conflicts should be no more than two percent of the total number of accesses. However, because some access conflicts may result from unusually high peak loads, there is no need to eliminate all access conflicts. Only when the percentage of access conflicts remains consistently greater than 2 percent is an additional MAIL.BOX database warranted.

Note: Mailbox statistics are available only on servers where two or more MAIL.BOX databases are configured. You must restart the server to put into effect any changes to the number of mailboxes.

MinNewMailPoll

This notes.ini setting determines how often workstations can contact the server to see if new mail has arrived for the user. This setting overrides the user's selection in the Mail Setup dialog box. You can increase the mail polling interval if there are a large number of mail users on your server and you want to prevent frequent polling from affecting server performance.

AdminP

The Administration Process (AdminP) automates routine administrative tasks, such as ID management, database moves, and so on. Some of these administration requests can require significant server resources to complete. During a rename for single user's common name, for example, each of the following elements *may* require processing:

- Person document
- Group documents containing the user's name
- Databases where the user's name is in the ACL
- Documents with readers, authors, or other names fields

In an environment where few administration requests are made, then there are few issues in allowing AdminP to update the Domino Directory and mail/application files during prime-shift. But, when the number of AdminP requests is large, there is the potential of a huge drain on CPU resources and also contention against the Domino Directory, which could impact user response times.

Implementing a designated administration server for AdminP allows administrators to create/stage AdminP requests during prime-shift, and then propagate the changes to the rest of the domain at an appropriate low-activity time by replicating the updated Domino Directory or the AdminP request database (admin4.nsf) for other changes.

Important: Even when using a single administration server for administration requests, the AdminP task must still run on any server affected by the requests.

Every Administration Request database, admin4.nsf, within a domain, will automatically have the same replica ID. Replication of the admin4.nsf is critical to ensure that requests that impact multiple databases on multiple servers will be processed correctly.

Reducing the size of admin4.nsf

Environments consisting of large numbers of servers, or where a large number of requests are common, may allow the size of administration requests database to become unmanageable. To conserve disk space, and reduce replication and indexing traffic, there are a couple of suggestions that can help.

1. Change the field, Store Admin Process log entries when status of no change is recorded, to No. This field, found in the server document under Server Tasks → Administration Process → Miscellaneous, controls whether or not administration request logs are saved if no action was performed by a server. The default setting of Yes, allows response logs with the status of This name did not appear anywhere or This file is not on this server to be

created. Unless it is important to confirm that the administration process of a server has its processing requests, even if it does not apply to that server, disable this setting.

2. Ensure that the Replication setting for Remove documents not modified in the last xx days is enabled for admin4.nsf. This setting is enabled and set to seven days by default. Without this setting enabled, admin4.nsf will not purge previously processed requests and logs, causing it to grow larger and larger over time. Likewise, increasing the number of days will also allow the database to grow as well.

Note: Replication formulas can also be used to keep the admin4.nsf's size down. For example, by using `Type!=AdminLog` for a selective replication formula, response logs from requests will not be replicated. This may make it more difficult to monitor a request's progress though, since the logs will be distributed among servers, so use this option at your own discretion.

5.2.2 Domino Memory Manager

Domino provides its own memory management mechanism that works in conjunction with the OS memory manager to dynamically allocate memory to Domino related tasks as needed. Using its own memory manager provides several benefits, including:

- The ability to track dynamic memory allocations for debugging purposes
- Allows more granular control over how certain allocations are done (based on size or purpose)
- Allows better cross-platform compatibility, by allowing memory requests by Domino processes to be made using a standard set of platform independent APIs
- Improves performance by retaining most its memory for later use

Even though Domino has its own memory manager, it still relies on the operating system to allocate memory to it in the first place. Once Domino has allocated memory from the OS, it will not typically release this memory, but hold onto it for future use. Some limited types of allocations are immediately returned to the OS, but a majority of allocations are persistent.

On a typical Domino server, the vast majority of memory managed by Domino is shared. Furthermore, the majority of that shared memory is usually part of the *Unified Buffer Manager (UBM)*, also known as the NSF Buffer Pool. The UBM may use 70-80% of the total memory allocated to Domino. For this reason, it is critical to effectively manage the UBM.

To determine whether the UBM needs to be resized or not, monitor the Database.BufferPool.PerCentReadsInBuffer statistic.

A high percentage here (>95%) indicates that the UBM is large enough, although it may be still be larger than necessary. This means that some portion of memory will go unused at all, so it might be a good idea to implement some sort of controls over the UBM directly

Domino 7 provides multiple notes.ini settings that may be used to configure the UBM as needed.

Setting `NSF_BUFFER_POOL_SIZE_MB` in the notes.ini sets the size of the UBM directly, while using any of the following settings in the notes.ini will affect how RAM is calculated on the system, therefore affecting the UBM size:

```
PercentAvailSysResources
ConstrainedSHM or ConstrainedSHMSizeMB
MEM_AddressableMem or MEM_AddressableMemSizeMB (ND 7.0.2)
```

Here is the logic Domino 7 uses for determining the physical RAM installed on a server.

1. Queries system for amount of *installed* RAM.
   - W32: Full amount of installed RAM returned in ND7.
   - UNIX®: Full amount installed RAM returned in ND7.
   - OS390: Domino hardcodes to 512 MB in ND7.
2. Applies PercentAvailSysResource INI to installed RAM size.
3. Applies User Space Limit (as of Domino 7.0.2) (if lower than #2).
   - Applies MEM_AddressableMemSizeMB, if specified.
   - Applies default size, if MEM_AddressableMem is set.
4. Applies Constrained Shared Memory Usage (if lower than #3).
   - Applies ConstrainedSHMSizeMB, if specified.
   - Applies ConstrainedSHM, if ConstrainedSHMSizeMB not specified.
   - Applies 4 GB cap, if neither are specified.

The physical memory never goes above 4 GB.

The maximum UBM size is then based on 3/8th calculated RAM from above, unless NSF_BUFFER_POOL_SIZE_MB is defined.

Here are some comparisons of the above settings that may help determine which is best suited for a particular need.

PercentAvailSysResources

- Typically used with multiple Domino Partitions on a single box
- Affects the calculation of RAM (and, indirectly, the UBM)
- Can specify what percentage of RAM each partition sees as installed
- Does *not* take into account inherent address space limitations
- Does *not* place a limit on shared memory usage outside of the UBM

ConstrainedSHM and ConstrainedSHMSizeMB

- Can be used with either partitioned servers or a stand-alone server
- Affects the calculation of RAM (and, indirectly, the UBM)
- Takes into account inherent address space limitations
- Enforces a limit on overall amount of shared memory usage, not just the UBM.
- Enforced directly by the Domino memory manager during allocation of shared memory

MEM_AddressableMem and MEM_AddressableMemSizeMB (7.0.2)

- Can be used with either partitioned servers or a stand-alone server
- Takes into account inherent address space limitations
- Affects the calculation of RAM (and, indirectly, the UBM)
- Does *not* place a limit on overall shared memory usage (cannot control overall shared memory usage with this INI)

NSF_BUFFER_POOL_SIZE_MB

- Can be used with either partitioned servers or a stand-alone server
- Does *not* take address space limitations into account
- Does *not* affect calculation of RAM, but does set a limit on UBM size
- Does *not* place a limit on overall shared memory usage

Note: ConstrainedSHM can generally replace the use of PercentAvailSysResources, although we recommend limiting ConstrainedSHMSizeMB to no higher than 2048 on AIX and Linux systems.

Setting ConstrainedSHM=1 will enforce a limit on shared memory that varies between 1.5 GB and 3 GB, depending on the platform.

Important: AIX uses a segmented address space consisting of 16 segments of 256 MB each. So, when using ConstrainedSHMSizeMB on AIX, be sure to adjust the value in increments of 256.

Tip: In most cases, any changes to the UBM size through the above settings should be in an effort to reduce the overall size of the UBM, not enlarge it

5.2.3 Database performance

Often, the most pervasive performance problems are due to poor database design. Unfortunately, a discussion on Database performance is simply too broad for this Redpaper.

The Domino Designer® 7 Help database provides a good starting point on optimizing database performance:

`http://doc.notes.net/domino_notes/7.0/help7_designer.nsf`

Additionally, two excellent articles on tuning the performance of Notes/Domino 7 databases may found at the following URLs:

`http://www-128.ibm.com/developerworks/lotus/library/notes7-application-performance1`
`http://www-128.ibm.com/developerworks/lotus/library/notes7-application-performance2`

5.2.4 Minimizing logging activity

Limit the amount of information that is logged to log.nsf and the console log.

For those parameters that provide various levels of logging (mail logging, log_replication, log_update, and so on), choose the less verbose versions of logging if possible to reduce the amount of output and potentially lower CPU requirements.

Disable HTTP server logging to improve Web performance.

Logging options are stored in the Server document. In the HTTP server Enable logging to section, there are two fields: Log files and DOMLOG.NSF. Disabling both of these fields improves Web server performance.

Disable parameters starting with "debug" when troubleshooting has been completed. They can add considerable CPU overhead as well as generate a large amount of output.

Do not log things that you never review!!!

5.2.5 Scheduling utilities

Schedule CPU-intensive utilities during non-peak processing times. Here are some examples.

Domino administrative utilities

Updall, Compact, and Fixup are Domino administrative utilities that, when run against a large number of databases (or even a small number of very large databases), can be very CPU intensive. For these scenarios, they should never be run during daytime peak processing periods because of the likelihood of elongated user response times.

In general, the following schedules are recommended for these utilities:

- Updall should be run nightly to update all view and full text indexes for all Domino databases.
- Compact should be run against all databases on a weekly basis. With the -B option, all unused/white space is recovered and file sizes are reduced; with the -b option, white space is recovered, but file sizes are not reduced. The -S option followed by a percentage will only compact databases when white space equals or has exceeded the specified percentage. Compact with the -b option is the recommended compaction style, but there are many others documented in the Domino Administration Help database. Which style you choose is dependent on your goals of reducing disk space, reorganizing databases to improve performance, or some other reasons. But keep in mind that running compact against a database with 5-10% of white space will not realize any noticeable performance gains.

  If running transaction logging in conjunction with a backup tool providing point-in-time database recovery, then the style, frequency, and schedules of compactions need to be considered in planning your backup/recovery strategy.
- Run Fixup *only* as needed against selected databases to fix corrupted views and documents. For databases that are transaction logged, there should never be a reason to run Fixup (unless requested to by the Support Center) if the Translog_AutoFixup parameter in NOTES.INI is set to 1. This setting will cause Domino to automatically repair any databases with corruption issues.

  If running transaction logging in conjunction with a backup tool providing point-in-time database recovery, a full backup of any databases that have been repaired with Fixup will be required immediately after running this utility.

Tip: Neither Compact nor Updall are destructive by nature, in that no actual data is removed during their execution. This makes them both perfectly safe to run on a regular basis.

Unlike Compact and Updall, Fixup does have a destructive component to it. If Fixup is unable to repair a corrupt element that it finds, it will simply delete that element. Therefore if Fixup is run on a schedule over the weekend, there is a chance (albeit a slim one) that it will delete documents from a database with no prior warning to administrators that the documents were corrupt.

By running Fixup manually, when indications of corruption are already present, administrators should be better prepared to address the problem, and not be surprised by it.

Updall, Compact, and Fixup are intended to execute serially to avoid locking and other issues. Each of these utilities offers the flexibility to run against specific databases, all databases within a folder, and all databases on a Domino server. If CPU cycles are available during off-shift periods, then multiple instances of these utilities may be run against different folders to shorten processing windows as long as there is no overlap of folder processing between any of the utilities.

- Backups of a large number of databases and, in some cases, recoveries of individual databases, should be scheduled to run off-shift and at different times than other Domino utilities because of the potentially large demand on system resources.

5.2.6 Impact of third-party applications

Domino may not be the only application installed on a system, so it is important to understand the impact that other applications may have on a Domino server's performance. Each process actively running on a machine is using some amount of memory, CPU, and so on. This is true whether the application is running in the foreground or in the background.

Some applications require more resources than others, and some may require large amounts of resources only on occasion, depending on the application, and its configuration.

Antivirus and backup software tend to be two of the most common third-party applications sharing resources with Domino. In many cases, both of these types of applications may directly interact with Domino using some of Domino's shared resources using our publicly available APIs; these are usually referred to as add-ins.

This Redpaper will not cover any specific applications or add-ins, but please keep the following recommendations in mind when trying to improve performance:

- Disable unnecessary third-party applications
- Schedule automated tasks, such as backups, to occur off hours.

5.2.7 Tuning user sessions

When all is said and done, performance is all about perception. The majority of performance problems are only considered problems because response times are unacceptable in one way or another. Domino provides many settings to help tune the way Domino handles individual user sessions, allowing administrators to cater to the needs of their users.

Maximum sessions

Notes clients create sessions to Domino servers to perform the various tasks necessary to perform their work. Domino does make a distinction between a *user* and a *session* for that user. A single Notes client user can potentially have several different sessions open with a single server. For example, a single user may have one session opened to their mail file, another session open to perform background replication, and still another open when polling for new mail.

All of this is expected, and is by design, but each idle session does utilize some resources that may be better utilized by servicing active requests. The notes.ini settings in Table 5-11 on page 81 allow administrators to customize the maximum number of users and sessions, and the timeouts associated with them.

Table 5-11 notes.ini settings related to user sessions

Setting	Definition
Server_MaxUsers	Sets the maximum number of users that are allowed to access a server. When this number is reached, the server state becomes MaxUsers, and the server stops accepting new Database Open requests. The default is 0 (unlimited access to server by user). Note: Unlike the below settings, when this value is reached, new users simply cannot create a connection to the server until the active number of users falls below this setting. This setting does *not* affect replication.
Server_Session_Timeout	Specifies the number of minutes of inactivity after which the server automatically terminates network and mobile connections. The minimum recommended setting is 15 minutes. If you specify a lower time, the server must reopen database server sessions too frequently, which slows server performance. For best performance, the recommended time is 45 minutes. Note: The default session time out is 240 minutes.
Server_MaxSessions	Defines the number of user sessions a Domino server can accommodate. When a new user attempts to log on, if the current number of sessions is greater than the value of SERVER_MAXSESSIONS= (in the NOTES.INI), the Domino server closes the least-recently-used session. In order for a session to be considered for closing, it must have been inactive for at least one minute. Note: Server_MaxSessions will not prevent the server from allowing more than that number of concurrent active users on the server, but will drop the sessions soon after they become inactive in order to free resources. Even in this case, the session must be idle for at least one minute before it will be dropped. Note: Server_MaxSessions parameter will never prevent a user from opening a session on the Domino server.

Attention: As mentioned in Table 5-11, Server_MaxUsers sets a hard limit on the number of users allowed to access the server, while Server_Session_Timeout and Server_MaxSessions only control the expiration of idle sessions.

6

Special considerations and advanced topics

In Chapter 5, “Understanding what to tune” on page 49, we provided specific insight and recommendations on configuration parameters and notes.ini variables to change. In this chapter, we go beyond simply discussing specific parameters and configuration settings related to performance. Instead, we focus on topics that can lead to a more reliable Domino infrastructure and result in users experiencing greater availability.

Specific topics covered in this chapter include:

- Domino clustering
- Failover
- Hard disk performance
- Redundant Arrays of Independent Disks (RAID)

6.1 Domino clustering

A Domino cluster links multiple Domino servers together so that they appear as one resource from the client perspective. The cluster functions as a “single” provider of resources, enabling client requests to be processed in a timely manner. If any given server is unavailable or too busy when the request arrives, the cluster transparently passes the request to a server that is capable of handling the work.

The cluster members can be on a mixture of the supported Domino platforms and does not have to be from the same vendor to be supported. The clusters support Notes clients only.

Domino clustering is accomplished entirely at the application level, while the operating system-level cluster provided by an operating system vendor or third party is accomplished on the OS level. No special hardware is needed for the Domino cluster, but be aware that you have to add resources to your system (CPU and RAM). Clustering has to be considered in the sizing of a system.

Domino clusters replicate database changes to all replica copies of the database as the changes occur. This synchronization of cluster components is key to Domino’s high availability. This style of replication is referred to as event-driven (immediate) replication, in contrast to standard replication that occurs on a schedule. Event-driven replication is a function of the cluster replicator.

6.1.1 Domino cluster components

The following section discusses the components of a Domino cluster.

Cluster Manager

The Cluster Manager resides on each cluster member and is responsible for exchanging messages with other cluster members (probes) to determine who is available in the cluster and their current capacity. The Cluster Manager decides where to send a connection request based on this data and reports the server status to other cluster members.

Cluster Administration

Cluster Administration creates the cldbdir.nsf database, which defines the databases contained within the cluster. The other components of clustering use this database to perform their functions. Cluster Replicator uses this database to decide which databases to replicate.

Cluster Replicator

Cluster Replicator handles the replication of the databases within the cluster. Replication in clustering is event-driven: When a change is made to a database, the Cluster Replicator immediately propagates the change to the other cluster members.

Internet Cluster Manager

The Internet Cluster Manager (ICM) enables you to use Domino clusters to provide fail-over and workload balancing to HTTP clients (Internet browsers) when they access Domino Web servers. This makes your Web servers and databases highly available to clients. You can run the ICM on a Lotus Domino 7 Enterprise server, a Lotus Domino 7 Utility server, a Lotus Domino 6 or 6.5 Enterprise server, a Lotus Domino 6 or 6.5 Utility server, or a Domino Release 5 Enterprise Server. Install and configure Domino clusters as you normally would, then configure the ICM. The ICM supports the HTTP and HTTPS protocols.

The ICM acts as an intermediary between HTTP clients and the Domino Web servers in a cluster. When Domino Web servers are running in a cluster, they generate URLs that direct HTTP client requests to the ICM. The ICM maintains information about the availability of servers and databases in the cluster. When the ICM receives a client request, it redirects the client to the most available server that contains a replica of the requested database.

The ICM sends periodic probes to the Web servers in the cluster to determine their status and availability. When the ICM receives a client request, it looks at the information in the Cluster Database Directory to find a server that contains the requested database. The ICM determines the most available server that contains the requested database and then redirects the client to that server. This results in the client closing the session with the ICM and opening a new session with the selected server. The user might see this as a change in the host name in the URL. The user might also see the path to the database change in the URL because the database might have a different path on the target server.

Note: Network dispatchers and load balance software can be used to cluster Internet protocols, but the Domino Cluster is the only tool to cluster NRPC.

6.1.2 Clustering from a technical perspective

Figure 6-1 on page 86 depicts the Domino application-level cluster based on Domino replication technology. Each Domino server is working as a logically different server, and they are replicating almost in real time with each other, which is known as cluster replication. The Notes client knows what servers are in the same cluster and has a feature of dynamic failover in case it is unable to access one of the servers. The clustering solution provides failover whether the access failure is caused by the Domino server, operating system, hardware, or network issues.

Another key advantage of Domino clustering is load balancing. In a Domino cluster, two or more servers are working and providing the same service simultaneously, and you can configure them to provide load balancing as well. You can control the server load for system availability or concurrent users.

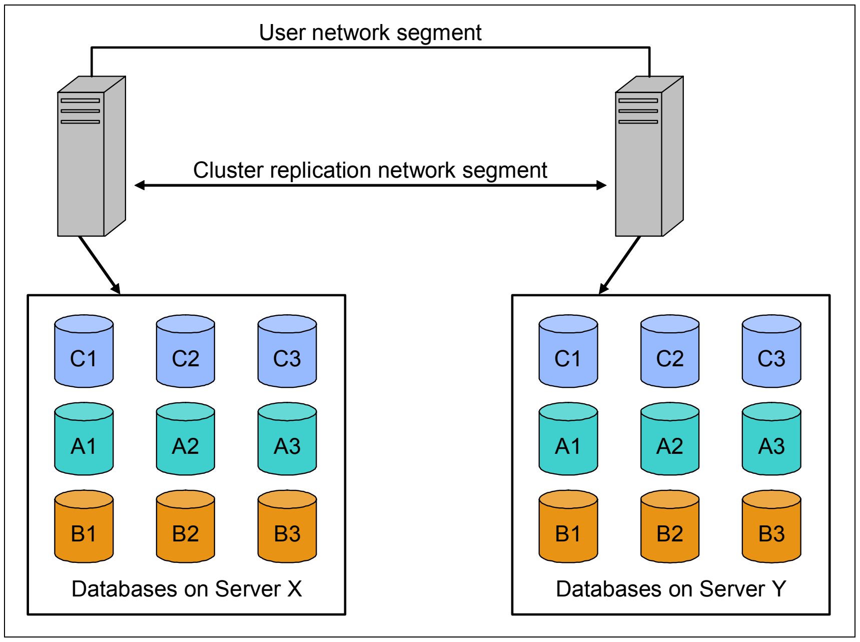


Figure 6-1 The basic architecture of Domino clustering

Browser failover

Unlike the Notes client, a browser does not offer dynamic fail-over functionality. Fortunately, the Domino server offers a feature for Web clustering called Internet Cluster Manager (ICM). ICM works as a front-end Web server for Web browsers followed by a back-end Domino cluster. The main function of ICM is redirecting the very first HTTP request from a Web browser to an available Domino Web server.

An alternative Web browser solution is IP sprayers instead of ICM. Domino cluster itself has an automatic replication feature and works fine with IP sprayers, such as IBM WebSphere Edge Server. This architecture is quite common for deploying clustering for WebSphere Application Server. You can achieve a similar architecture if you fit Domino cluster technology and WebSphere Edge Server technology together. In this case, Domino does the data replication, and IP sprayers perform the fail-over and load balancing.

Hardware cluster

The other way to build a cluster is by using operating system clustering, which depends on each operating system (for example, HACMP™ on IBM AIX 5L™) or cluster products. Note that this cluster solution is similar to the Domino cluster, but the architecture of the cluster is absolutely different (see Figure 6-2 on page 87). Operating system clustering requires a shared disk instead of data duplication, and both servers are connected with a heartbeat line and provide failover if the other server is unavailable. One server is usually providing service and the other is waiting, so it cannot provide a load balancing solution. If there is some problem for the primary server, the secondary server takes over all the resource, including IP address and shared disk, and starts up as though it was the primary server.

The server monitor usually handles only the health condition of the operating system and not the application process status, such as a crash or hang. Fortunately, Domino has a very useful function called Fault Recovery, which enables the server to automatically recover from its crash. Because an operating system cluster is based on a shared disk model, the fail-over time usually depends on the environment and how much disk quota is available. Because the minutes for takeover will be critical for clustering, consult an OS clustering expert. In contrast, Domino clustering provides a zero downtime cluster because the Notes client automatically detects a failure and shifts to the other cluster member.

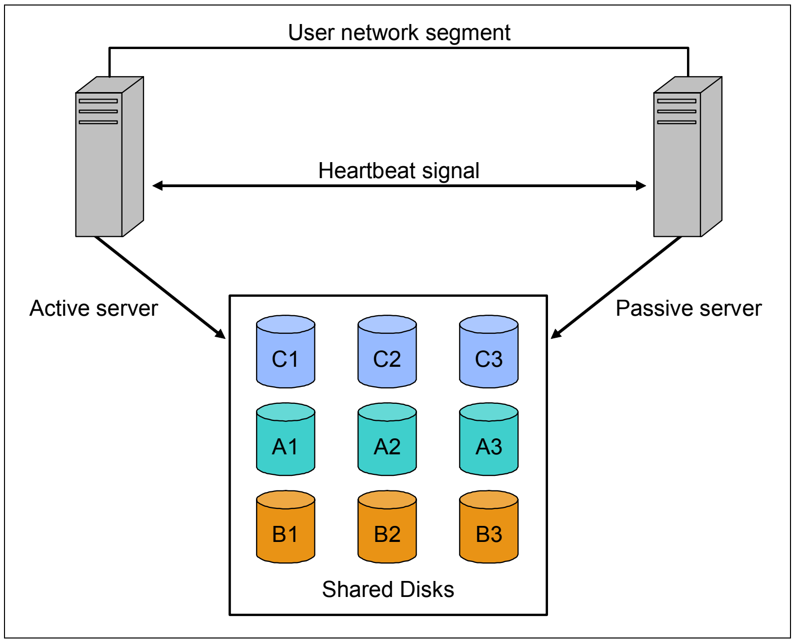


Figure 6-2 The basic architecture of operating system clustering

6.1.3 Workload balancing

With clustering, multiple copies of databases on multiple servers provide high availability. In addition, Domino distributes the workload between the cluster members (this is called workload balancing), allowing for lower overall response times and more consistency in response times during peak intervals.

Domino clusters provide workload balancing by redistributing user requests to an overloaded server to other servers in the cluster that have available capacity. To optimize workload balancing, you can modify the following settings:

- Database distribution
- Server availability index limit (parameter `SERVER_AVAILABILITY_THRESHOLD`)
- Location document of the clients (specify different home servers)
- Maximum number of concurrent users in a server (parameter `SERVER_MAXUSERS`)

Database distribution

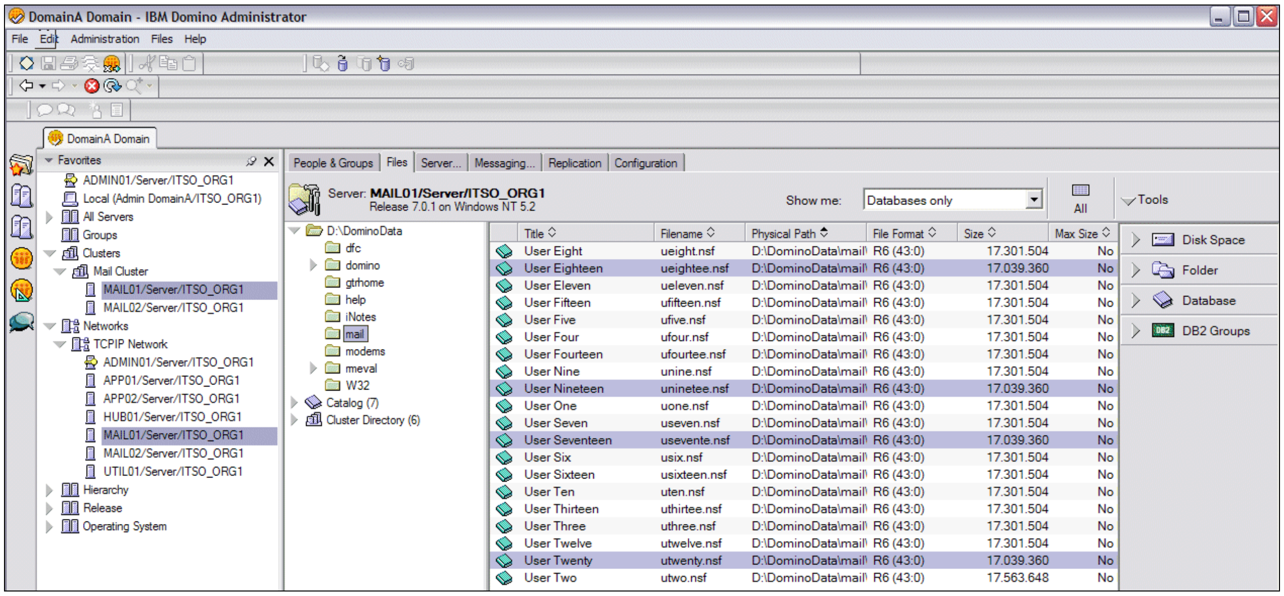
Make sure that you distribute databases evenly in the cluster. When a server in the cluster fails or becomes overloaded, user requests are automatically redirected to other servers in the cluster.

Ideally, this load should be spread equally across all other servers in the cluster. However, this can happen only when replicas of the databases on the failed server are spread roughly equally across the other servers in the cluster.

Note that if you distribute the databases evenly across the servers, you are assuming that the databases have about the same activity. If you have power users or particularly active databases, you might need to fine-tune the distribution of those databases to make the activity on each server approximately equal.

You can use the Administration Process to make database distribution easier:

1. Open the administrator client. Select the source server, open the tab for your current domain, and click the pushpin icon.
2. Hold down the Ctrl key while you select as many databases as you want to create new replicas for. Drag them over the server that you want to be the destination of the new replicas (see Figure 6-3).



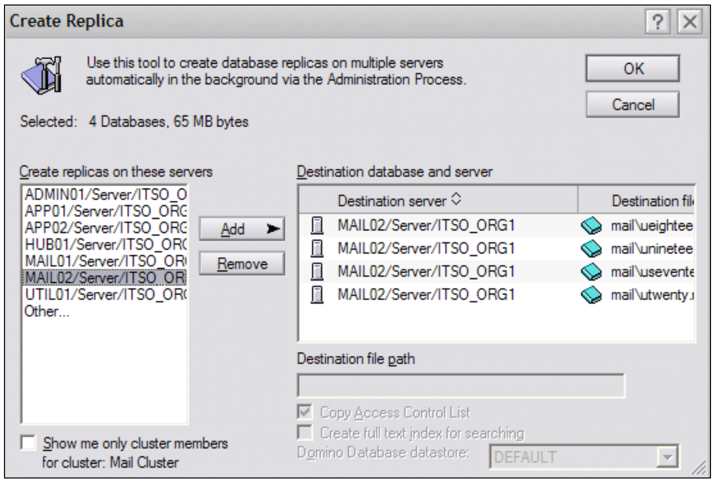
Server: **MAIL01/Server/ITSO_ORG1**
Release 7.0.1 on Windows NT 5.2

Show me: Databases only

Title	Filename	Physical Path	File Format	Size	Max Size
User Eight	ueight.nsf	D:\DominoData\mail\	R6 (43:0)	17.301.504	No
User Eighteen	ueightee.nsf	D:\DominoData\mail\	R6 (43:0)	17.039.360	No
User Eleven	ueleven.nsf	D:\DominoData\mail\	R6 (43:0)	17.301.504	No
User Fifteen	ufifteen.nsf	D:\DominoData\mail\	R6 (43:0)	17.301.504	No
User Five	ufive.nsf	D:\DominoData\mail\	R6 (43:0)	17.301.504	No
User Four	ufour.nsf	D:\DominoData\mail\	R6 (43:0)	17.301.504	No
User Fourteen	ufourtee.nsf	D:\DominoData\mail\	R6 (43:0)	17.301.504	No
User Nine	unine.nsf	D:\DominoData\mail\	R6 (43:0)	17.301.504	No
User Nineteen	uninetee.nsf	D:\DominoData\mail\	R6 (43:0)	17.039.360	No
User One	uone.nsf	D:\DominoData\mail\	R6 (43:0)	17.301.504	No
User Seven	useven.nsf	D:\DominoData\mail\	R6 (43:0)	17.301.504	No
User Seventeen	usevente.nsf	D:\DominoData\mail\	R6 (43:0)	17.039.360	No
User Six	usix.nsf	D:\DominoData\mail\	R6 (43:0)	17.301.504	No
User Sixteen	usixteen.nsf	D:\DominoData\mail\	R6 (43:0)	17.301.504	No
User Ten	uten.nsf	D:\DominoData\mail\	R6 (43:0)	17.301.504	No
User Thirteen	uthirtee.nsf	D:\DominoData\mail\	R6 (43:0)	17.301.504	No
User Three	uthree.nsf	D:\DominoData\mail\	R6 (43:0)	17.301.504	No
User Twelve	utwelve.nsf	D:\DominoData\mail\	R6 (43:0)	17.301.504	No
User Twenty	utwenty.nsf	D:\DominoData\mail\	R6 (43:0)	17.039.360	No
User Two	utwo.nsf	D:\DominoData\mail\	R6 (43:0)	17.563.648	No

Figure 6-3 Domino Administrator Files Tab

3. The databases are replicated by the Administration Process to the selected destination server (see Figure 6-4).



Use this tool to create database replicas on multiple servers automatically in the background via the Administration Process.

Selected: 4 Databases, 65 MB bytes

Create replicas on these servers

ADMIN01/Server/ITSO_O
APP01/Server/ITSO_ORG
APP02/Server/ITSO_ORG
HUB01/Server/ITSO_ORG
MAIL01/Server/ITSO_ORG
MAIL02/Server/ITSO_OR
UTIL01/Server/ITSO_ORG
Other...

Destination database and server

Destination server	Destination fil
MAIL02/Server/ITSO_ORG1	mail\ueightee
MAIL02/Server/ITSO_ORG1	mail\uninetee
MAIL02/Server/ITSO_ORG1	mail\usevente
MAIL02/Server/ITSO_ORG1	mail\utwenty.i

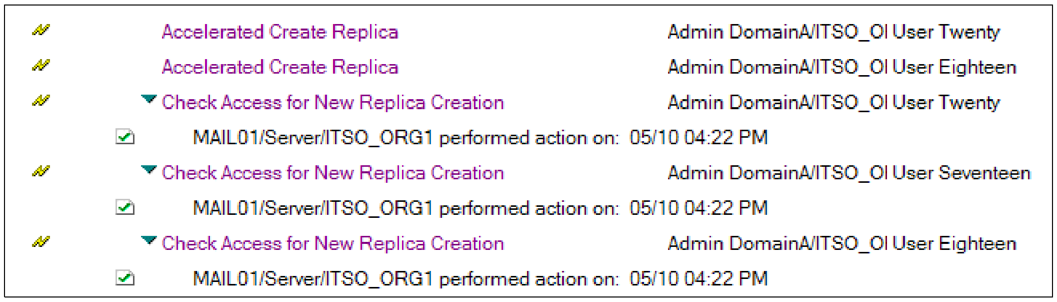
Destination file path

[✓] Copy Access Control List
[ ] Create full text index for searching
Domino Database datastore: DEFAULT

[ ] Show me only cluster members for cluster: Mail Cluster

Figure 6-4 Distributing four selected databases in the Domino cluster

4. Check for possible errors and confirmation of the move in the admin4.nsf database from the administration server in the Domino Directory (see Figure 6-5 on page 89).



Accelerated Create Replica — Admin DomainA/ITSO_OI User Twenty
Accelerated Create Replica — Admin DomainA/ITSO_OI User Eighteen
Check Access for New Replica Creation — Admin DomainA/ITSO_OI User Twenty
MAIL01/Server/ITSO_ORG1 performed action on: 05/10 04:22 PM
Check Access for New Replica Creation — Admin DomainA/ITSO_OI User Seventeen
MAIL01/Server/ITSO_ORG1 performed action on: 05/10 04:22 PM
Check Access for New Replica Creation — Admin DomainA/ITSO_OI User Eighteen
MAIL01/Server/ITSO_ORG1 performed action on: 05/10 04:22 PM

Figure 6-5 Administration servers admin4.nsf confirmation of move

6.1.4 Clustering over a WAN

A cluster over a wide area network (WAN) works the same way as a cluster on a LAN. However, if you have a low-speed WAN, you should consider disabling cluster replication. Instead, use scheduled replication more frequently than usual, such as every hour. This reduces WAN traffic, bottlenecks, and the cost of continual transmission.

Also keep in mind that Domino fails over to the most available server in the cluster, not the closest server. For example, if you have three servers, one in Boston, one in New York, and one in Japan, the Boston server would fail over to the Japan server if it is more available than the New York server. You can control this behavior to some extent by changing the server availability thresholds on the cluster servers.

Using a cluster over a WAN is a good idea for disaster planning or simply extending the availability service for travelling users within your organization. Having emergency backup servers at different locations is a good way to ensure that necessary data is always available when you need it.

You must define what is required in terms of databases within the cluster to avoid taking up bandwidth with unnecessary replication. By conducting this exercise, the performance within the cluster and the WAN network link is optimized.

If the purpose is to have a backup site utilizing the WAN, you might not need to cluster the data but could rely on a frequent scheduled replication instead. This will also optimize your WAN performance / bandwidth. If the goal of disaster recovery is simply providing the infrastructure for emergency e-mail routing, you do not need to hold and synchronize the original data at the backup data center.

In most cases, Domino clustering across a wide area network (WAN) works well. There are, however, some points of consideration you should evaluate to avoid to many performance issues:

- Required network bandwidth
  - An IBM guideline for supporting clustered Domino mail servers across a WAN is that *one T1 will handle 1,000 registered users*.
  - Other than replication traffic, you must consider the user's direct access in case you allow failover or load balancing.
- Features that are not supported in Domino cluster
  - Even though the Domino cluster works well to redirect users to the backup server, you need to consider server agents, tasks, and so on that are not supported in the Domino cluster.

- Accessed when it is not a disaster solution
  - Domino cluster is not designed primarily for disaster recovery. If you want to make users access backup servers only at the time of disaster, configure in this manner. For example, restrict users using the NOTES.INI file `Server_Restricted=2` parameter and remove it at the time of the disaster.

6.1.5 Server availability index

Each server in a cluster periodically determines its own workload based on the response time of the requests the server has processed recently. The workload is expressed as a number from 0 to 100, where 0 indicates a heavily loaded server and 100 indicates a lightly loaded server. This number is called the server availability index. As response times increase, the server availability index decreases.

The server availability index is based on the *expansion factor*, which indicates the current workload on a server. The expansion factor is determined by comparing recent response times for specific types of transactions to the minimum time in which the server has ever completed the same types of transactions. For example, if the server is currently averaging 12 microseconds to perform Database Open transactions, but the minimum time the server has ever performed a Database Open transaction is 3 microseconds, the expansion factor for Database Open transactions would be 4 (the current time of 12 microseconds divided by the fastest time of 3 microseconds). In other words, the expansion factor determines how many times longer it takes for a transaction to complete currently than it takes under optimal conditions.

Note: It was brought to our attention that the Administrator Help file documentation indicates that the SAI calculation is a linear indication of the amount of resources being used. After further investigation, it was noticed that this information is not completely true and in fact SAI is not a linear number. It represents a calculation based in a resource average estimate. For example, if SAI is equal to 65, that means you have less then 35% of free resources, and not 35% free as the Help files indicates.

Domino stores the minimum time for each type of transaction in memory and in the loadmon.ncf file, which the server reads each time it starts. When the server shuts down, Domino updates the loadmon.ncf file with the latest information.

Important: Remember to delete loadmon.ncf in case of a version upgrade. The cluster service will create a new one and this will prevent version conflicts.

To determine the current expansion factor, Domino tracks the most commonly used types of Domino transactions for specified periods of time. By default, Domino tracks these transactions for five periods of 15 seconds each. Domino then determines the average time it took to complete each type of transaction and divides that time by the minimum time it ever took to complete that same type of transaction. This determines an expansion factor for each type of transaction. To determine the expansion factor for the entire server, Domino averages the expansion factors for all the types of transactions, giving a heavier weighting to the most frequently used types of transactions.

As the server gets busier, adding more load has an increasingly greater effect on performance and availability. Thus, adding more load to a busy server increases the expansion factor faster than adding more load to a less busy server.

Because servers differ in speed, capacity, and power, servers also differ in the workload they can handle. Therefore, the same expansion factor on two different servers does not necessarily indicate the same workload relative to the ability of the servers. For example, on a small server that takes a long time to perform transactions when the server is otherwise idle, an expansion factor of 40 might indicate that users are waiting many seconds for responses. On a very large, fast server, however, an expansion factor of 400 might indicate that users are waiting less than a second for responses.

How the availability index compares to the expansion factor

To determine the availability index, Domino uses a formula that converts the expansion factor into an approximation of the percentage of the total server capacity that is still available. Table 6-1 shows a few examples of expansion factors converted to availability indexes.

Note: The values in the table are based an expansion factor of 64, indicating a fully loaded server.

Table 6-1 Expansion factors converted to availability indexes

Expansion factor	Availability index
1	100
2	83
4	67
8	50
16	33
32	17
64	0

Note: The expansion factor and the availability index measure only the response time of the server, which is usually only a small portion of the response time client's experience. For example, the network response time between a client and a server often accounts for a significant portion of the response time the client experiences.

Changing the value of the expansion factor that indicates a fully loaded server

To use Domino workload balancing effectively, you must adjust the relationship between the expansion factor and the availability index so that servers fail over when they reach the workload at which you want them to fail over. You do this by specifying the expansion factor value that you want to represent a fully loaded server. The default value in Domino is 64. When the expansion factor reaches that value, the server is considered to be fully loaded, and the availability index drops to 0 (zero).

Note: You can use the `Show AI` server command to view a suggested availability index setting.

If your server is particularly powerful and fast, you might want to increase the value of the expansion factor that is considered fully loaded. On some very fast servers, you might want to make this value several hundred or higher. If your server is particularly slow, you might want to decrease this value.

To change the expansion factor value that indicates a fully loaded server, add the following setting to your notes.ini file, and then restart the server:

```
SERVER_TRANSINFO_RANGE=n
```

For the value of *n*, choose a number such that 2 raised to the power of *n* equals the expansion factor value that you want to indicate a fully loaded server. The default value for *n* is 6, which leads to an expansion factor value of 64, because 2 raised to the 6th power is 64. If you set SERVER_TRANSINFO_RANGE to 7, then the expansion factor value that indicates a fully loaded server becomes 128. If you set SERVER_TRANSINFO_RANGE to 8, the value becomes 256.

To determine the optimal value for SERVER_TRANSINFO_RANGE:

1. During a period of heavy usage, monitor the expansion factor on your server. You can use the console command **`show stat server.expansionfactor`** to do this. You can also monitor performance statistics during these periods. Record enough values for the expansion factor during heavy usage so that you can determine the expansion factor value you want to indicate a fully loaded server.
2. Determine a value for SERVER_TRANSINFO_RANGE so that 2 raised to the power of that value results in the expansion factor value you chose in step 1.

When you change the expansion factor value that indicates a fully loaded server, the relationship between the expansion factor and the availability index changes. The example below shows an example of expansion factors converted to availability indexes when the value of SERVER_TRANSINFO_RANGE is 8. The maximum expansion factor in this example is 256 because 2 raised to the power of 8 is 256:

```
SERVER_TRANSINFO_RANGE = 8
```

The server console command **`Show AI`** is used to obtain an appropriate value for this variable. Type the command after running the server under load for a while, and it will display the history of the expansion factor and availability index for the server.

Changing the amount of data to compute the expansion factor

Although it is not usually necessary, you can use the following notes.ini settings to change the amount of data that Domino collects in order to figure the expansion factor.

To change the number of data collection periods that Domino uses, use the notes.ini setting `Server_Transinfo_Max=x`, where *x* is the number of collection periods you want Domino to use.

To change the length of each data collection period, use the notes.ini setting `Server_Transinfo_Update_Interval=x`, where *x* is the length of each period in seconds.

Note: Improvements in the SAI calculation algorithms were made in Domino versions 6 and above that improve the accuracy and, therefore, reliability of the SAI number. This is useful when the load balance feature is used. Based on the SAI, the load balance feature can direct client connection requests to the cluster member with more free resources.

6.2 Failover

In a Domino cluster, if one member of the cluster fails, another member of the cluster transparently assumes the failing member's workload.

This action is called *failover*. Failover is client-driven, and when a client cannot connect to a server, it checks in its cluster cache for other servers in the cluster, thus redirecting requests to another server in the cluster that has a replica of the database needed to service the request.

Note: To quickly test a failover, use the SERVER_RESTRICTED=1 variable from the server console **`set config SERVER_RESTRICTED=1`**. A server in restricted state does not accept any connection. Remember to disable the server restriction by resetting the variable to 0.

6.3 Creating the Domino cluster

The cluster configuration consists of two parts: the configuration of the port that will be used by the server within the cluster, and the cluster itself. The correct configuration of the Cluster ports ensures more reliability and good performance of the cluster, and well as the rest of the factors mentioned before.

6.3.1 Configuring the cluster ports

We recommend that users create the cluster port from the Administrator client. This helps the system configure the best parameters for your ports settings, such as port compression.

1. Make sure you have administrative rights on the server (at least server administrator access). Open the administrator client to start the configuration. The window shown in Figure 6-6 should appear.

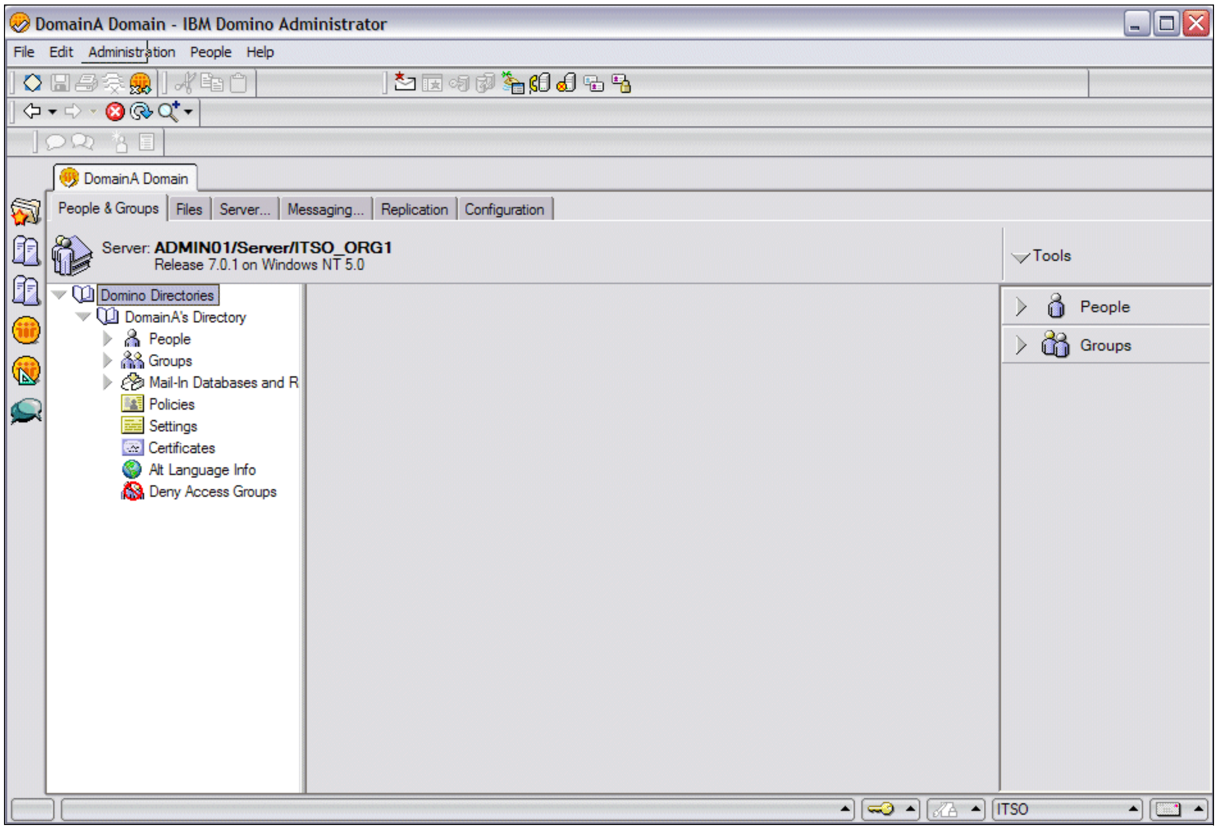


Figure 6-6 Administrator client initial window

2. Select the server within your Domain to configure the cluster port. You can use the Domain server list on the left pane of the administrator client (Figure 6-7 on page 95), or from the menu, select **File** → **Open** server.

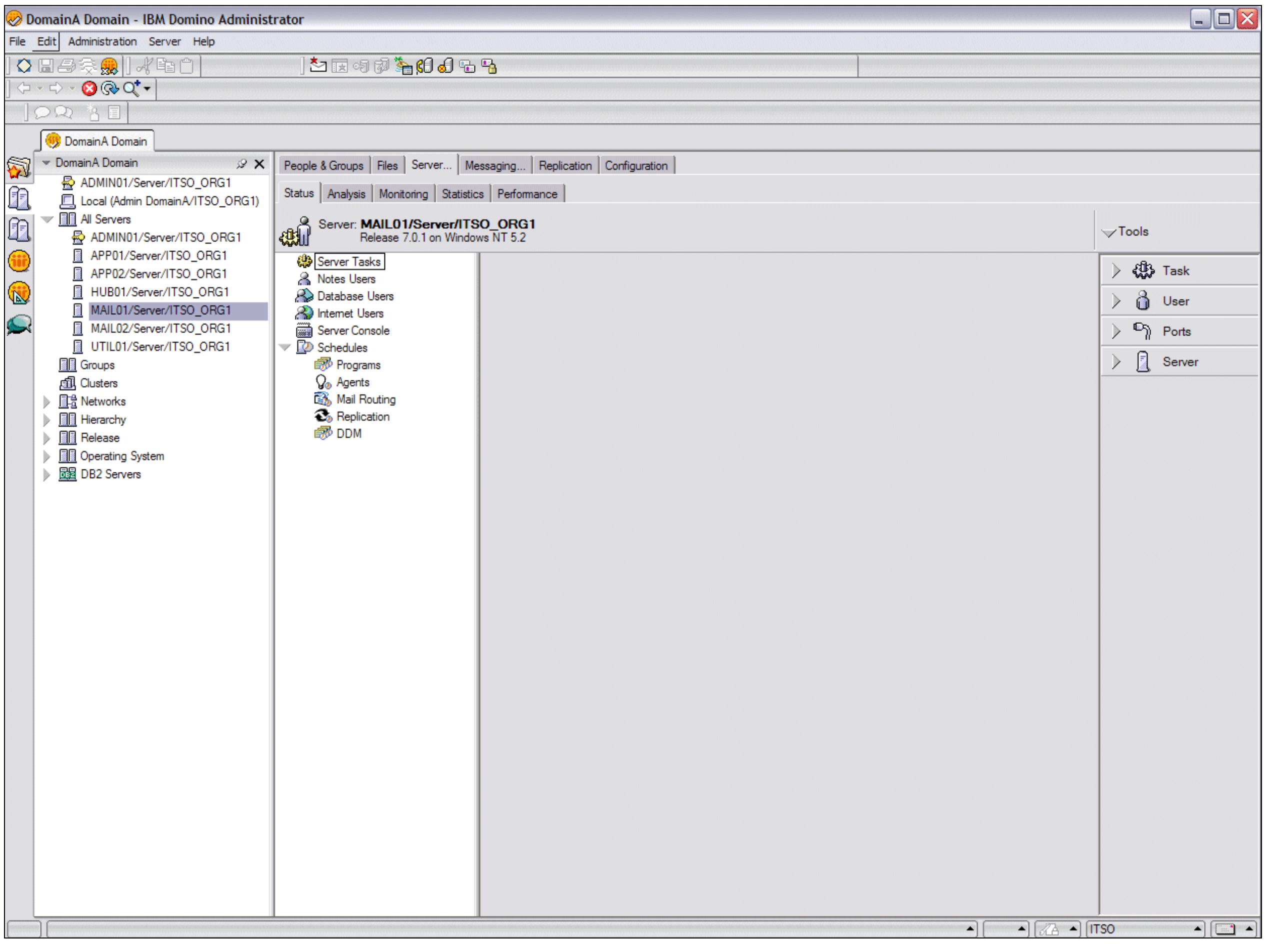
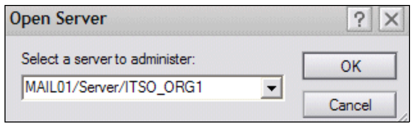


Figure 6-7 Selecting the server

3. You might need to type in the server DNS host name or IP address, or create a connection document if this is the first time you connect to the server (see Figure 6-8).

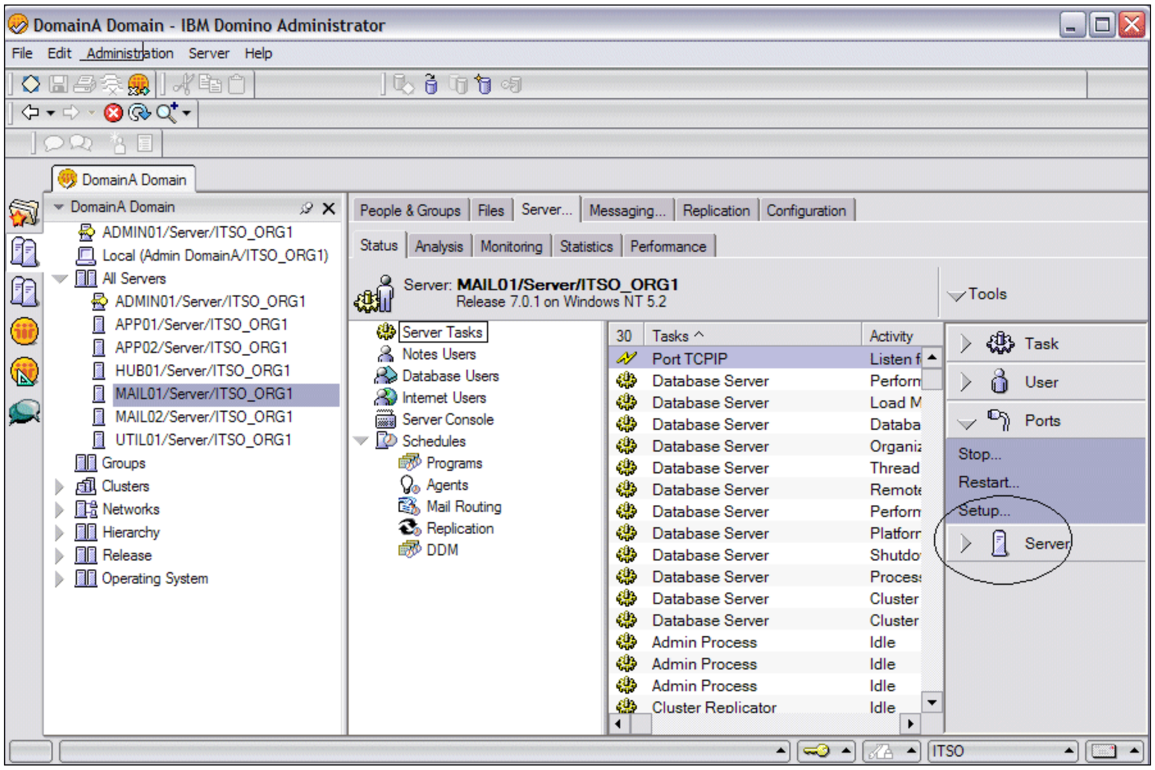

Open Server

Select a server to administer:

MAIL01/Server/ITSO_ORG1

Figure 6-8 Directly selecting the server within the Domino domain

4. Click the **Server** tab. Under the ports configuration, click **Setup** (circled in Figure 6-9) to configure and start the cluster ports.



DomainA Domain
- ADMIN01/Server/ITSO_ORG1
- Local (Admin DomainA/ITSO_ORG1)
- All Servers
  - ADMIN01/Server/ITSO_ORG1
  - APP01/Server/ITSO_ORG1
  - APP02/Server/ITSO_ORG1
  - HUB01/Server/ITSO_ORG1
  - MAIL01/Server/ITSO_ORG1
  - MAIL02/Server/ITSO_ORG1
  - UTIL01/Server/ITSO_ORG1
- Groups
- Clusters
- Networks
- Hierarchy
- Release
- Operating System

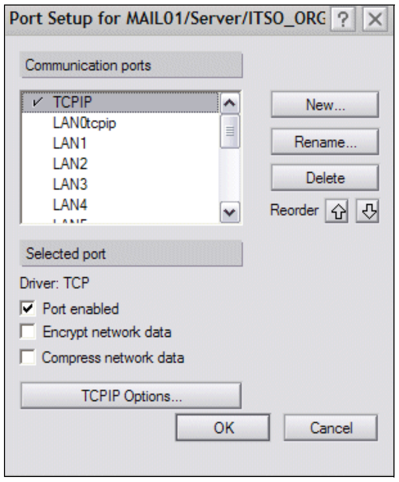
Server: **MAIL01/Server/ITSO_ORG1**
Release 7.0.1 on Windows NT 5.2

- Server Tasks
- Notes Users
- Database Users
- Internet Users
- Server Console
- Schedules
  - Programs
  - Agents
  - Mail Routing
  - Replication
  - DDM

30	Tasks	Activity
	Port TCPIP	Listen f
	Database Server	Perforn
	Database Server	Load M
	Database Server	Databa
	Database Server	Organiz
	Database Server	Thread
	Database Server	Remote
	Database Server	Perforn
	Database Server	Platforr
	Database Server	Shutdo
	Database Server	Proces:
	Database Server	Cluster
	Database Server	Cluster
	Admin Process	Idle
	Admin Process	Idle
	Admin Process	Idle
	Cluster Replicator	Idle

Figure 6-9 Selecting the port configuration in the Server tab

5. There should be a existing port that the server is using to communicate with the other servers in the environment and the clients. In the Port Setup window, click **New** to set up the new cluster port (see Figure 6-10).



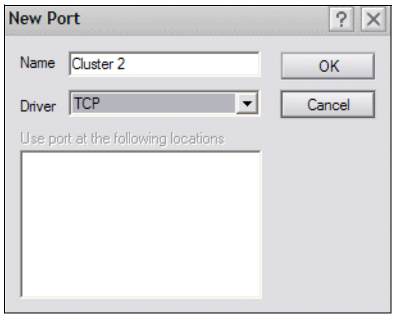
Communication ports:
- ✓ TCPIP
- LAN0tcpip
- LAN1
- LAN2
- LAN3
- LAN4

Driver: TCP
- [✓] Port enabled
- [ ] Encrypt network data
- [ ] Compress network data

Figure 6-10 New port creation

Note: This new port might want to represent a different NIC dedicated only to cluster traffic. This is highly recommended for best results. Also, 1.1 KBps/user is a good start to measure cluster replication traffic load between the cluster members. Although this number is not an absolute reference, you might want to use it as a base and include more users if the environment behaves efficiently.

Figure 6-11 shows the window where you name the new port.



New Port

Name: Cluster 2

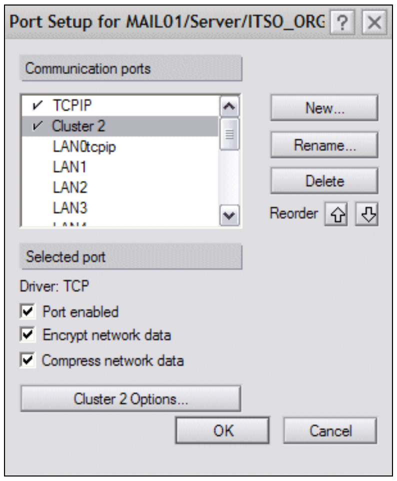
Driver: TCP

Use port at the following locations

Figure 6-11 Naming the cluster port

Attention: The new cluster port is created, but to ensure that all cluster communication uses only the cluster port, the new port must be reorganized as the first port listed in the Communication ports box list.

Figure 6-12 shows the new port's settings.



Port Setup for MAIL01/Server/ITSO_ORG

Communication ports

- ✓ TCPIP
- ✓ Cluster 2
- LAN0tcpip
- LAN1
- LAN2
- LAN3

Selected port

Driver: TCP

- ✓ Port enabled
- ✓ Encrypt network data
- ✓ Compress network data

Figure 6-12 New port settings

Depending on your network topology, it might be interesting to enable/disable port compression. If the servers are using a dedicated and fast network segment, it is best to disable network compression to prevent the additional workload required by the compression algorithm.

The port configuration changes will be available after the server is restarted.

The port configuration in notes.ini should look similar to Example 6-1.

Example 6-1   notes.ini port configuration example

```
Ports=CLUSTER2,TCPIP
TCPIP=TCP, 0, 15, 0,,32800
TCPIP_TCPIPADDRESS=0,9.33.85.102:1352
CLUSTER=TCP,0,15,0,,45056,
CLUSTER_TCPIPADDRESS=0,192.168.1.102:1352
```

You may also include the Server_Default_Cluster_Port parameter to force the cluster to communicate using the desired cluster port:

```
Server_Default_Cluster_Port=CLUSTER 2
```

There is a disadvantage to using the Server_Cluster_Default_Port setting to assign a port to the private LAN for cluster traffic. If a cluster server encounters a problem connecting over this port, it will not try another port. Therefore, the server will not be able to communicate or replicate with other cluster servers. You will have to resolve the network problem or remove this setting from the notes.ini file before the server will be able to communicate with the cluster again.

Note: The port configuration must be performed on each server that will belong to the Domino Cluster. Make sure you configure all servers' ports before moving on to the next step.

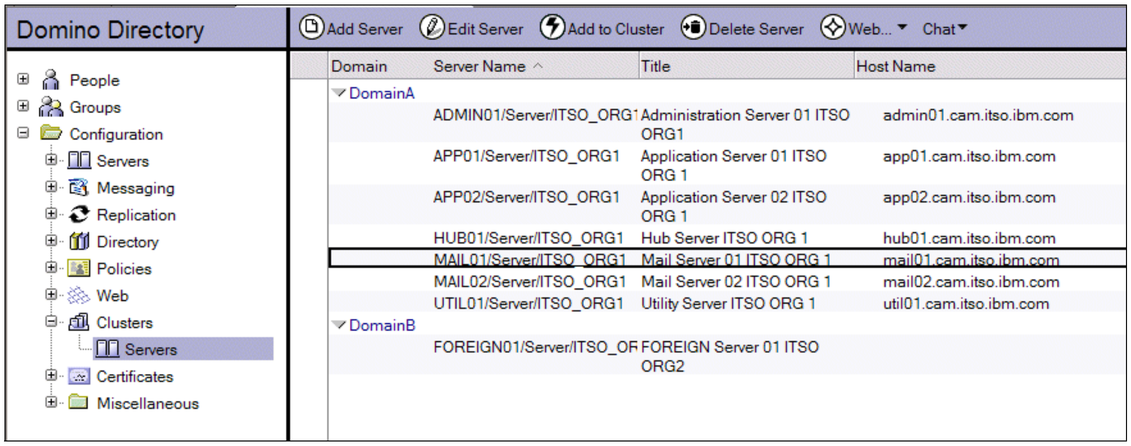
6.3.2 Setting up the Domino Cluster

After port configuration, we move through the steps of the cluster configuration procedure:

1. Open the Domino Directory database (names.nsf) in the administration server.

Note: Make sure you open the Domino Directory database located at administration server in order to have the administration take care of the final cluster configurations.

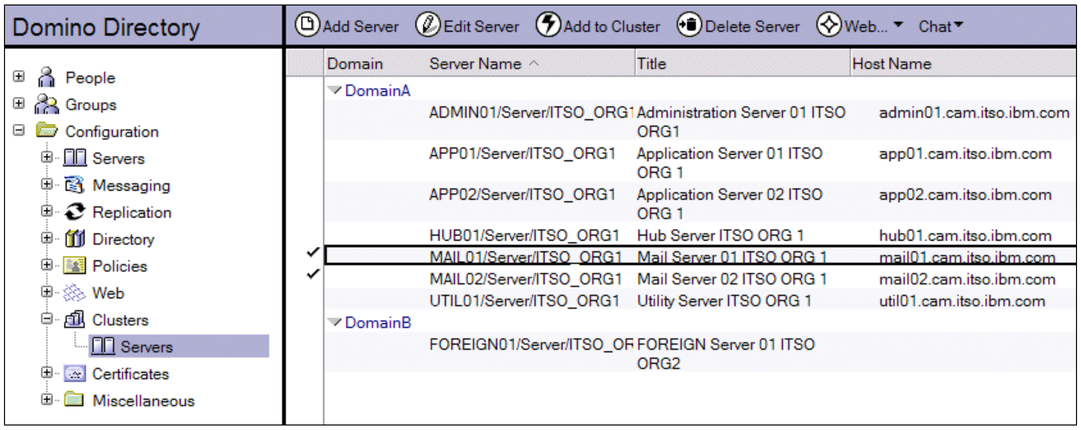
2. Select the view **All Servers Documents** under the **Server** → **Configurations** view, as shown in Figure 6-13.



Domain	Server Name	Title	Host Name
DomainA			
	ADMIN01/Server/ITSO_ORG1	Administration Server 01 ITSO ORG1	admin01.cam.itso.ibm.com
	APP01/Server/ITSO_ORG1	Application Server 01 ITSO ORG 1	app01.cam.itso.ibm.com
	APP02/Server/ITSO_ORG1	Application Server 02 ITSO ORG 1	app02.cam.itso.ibm.com
	HUB01/Server/ITSO_ORG1	Hub Server ITSO ORG 1	hub01.cam.itso.ibm.com
	MAIL01/Server/ITSO_ORG1	Mail Server 01 ITSO ORG 1	mail01.cam.itso.ibm.com
	MAIL02/Server/ITSO_ORG1	Mail Server 02 ITSO ORG 1	mail02.cam.itso.ibm.com
	UTIL01/Server/ITSO_ORG1	Utility Server ITSO ORG 1	util01.cam.itso.ibm.com
DomainB			
	FOREIGN01/Server/ITSO_OF	FOREIGN Server 01 ITSO ORG2	

Figure 6-13   Domino Directory database

3. Mark the servers that will be clustered and click **Add to Cluster** (see Figure 6-14).


Domino Directory

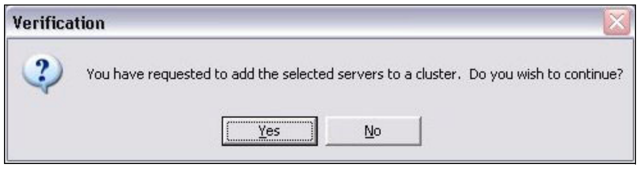
- People
- Groups
- Configuration
  - Servers
  - Messaging
  - Replication
  - Directory
  - Policies
  - Web
  - Clusters
    - Servers
  - Certificates
  - Miscellaneous

Add Server | Edit Server | Add to Cluster | Delete Server | Web... | Chat

	Domain	Server Name	Title	Host Name
	DomainA			
		ADMIN01/Server/ITSO_ORG1	Administration Server 01 ITSO ORG1	admin01.cam.itso.ibm.com
		APP01/Server/ITSO_ORG1	Application Server 01 ITSO ORG 1	app01.cam.itso.ibm.com
		APP02/Server/ITSO_ORG1	Application Server 02 ITSO ORG 1	app02.cam.itso.ibm.com
		HUB01/Server/ITSO_ORG1	Hub Server ITSO ORG 1	hub01.cam.itso.ibm.com
✓		MAIL01/Server/ITSO_ORG1	Mail Server 01 ITSO ORG 1	mail01.cam.itso.ibm.com
✓		MAIL02/Server/ITSO_ORG1	Mail Server 02 ITSO ORG 1	mail02.cam.itso.ibm.com
		UTIL01/Server/ITSO_ORG1	Utility Server ITSO ORG 1	util01.cam.itso.ibm.com
	DomainB			
		FOREIGN01/Server/ITSO_OF	FOREIGN Server 01 ITSO ORG2	

Figure 6-14 Selecting server to be clustered

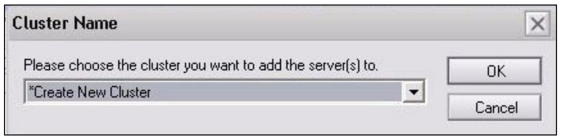
4. If you are sure you marked the right servers, click **Yes**; otherwise, click **No** and reselect the desired servers (see Figure 6-15).


Verification

You have requested to add the selected servers to a cluster. Do you wish to continue?

Figure 6-15 Confirming the setup of the selected servers

5. A combo box appears for you to create a new cluster or include another server in a existing Domino Cluster. If this is a new cluster, select ***Create New Cluster** in the drop-down menu (see Figure 6-16). If this is a existing server, select the Domino Cluster in which the server or servers will be included.


Cluster Name

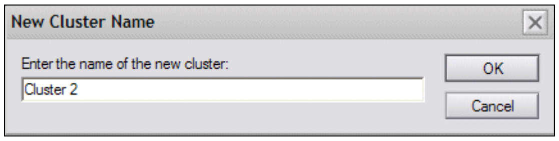
Please choose the cluster you want to add the server(s) to.

*Create New Cluster

Figure 6-16 Setting up the Cluster Name

6. Type the new cluster name and click **OK** (see Figure 6-17).

Attention: Make sure that this cluster name is unique in the Domino Directory.

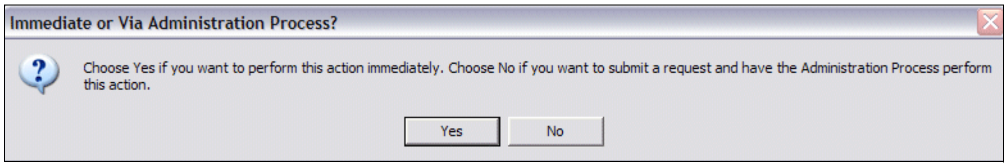

New Cluster Name

Enter the name of the new cluster:

Cluster 2

Figure 6-17 Naming the cluster

7. The system asks whether you want to create the cluster immediately (click **Yes**) or let the Administration Process create it for you (click **No**) (see Figure 6-18).


Immediate or Via Administration Process?

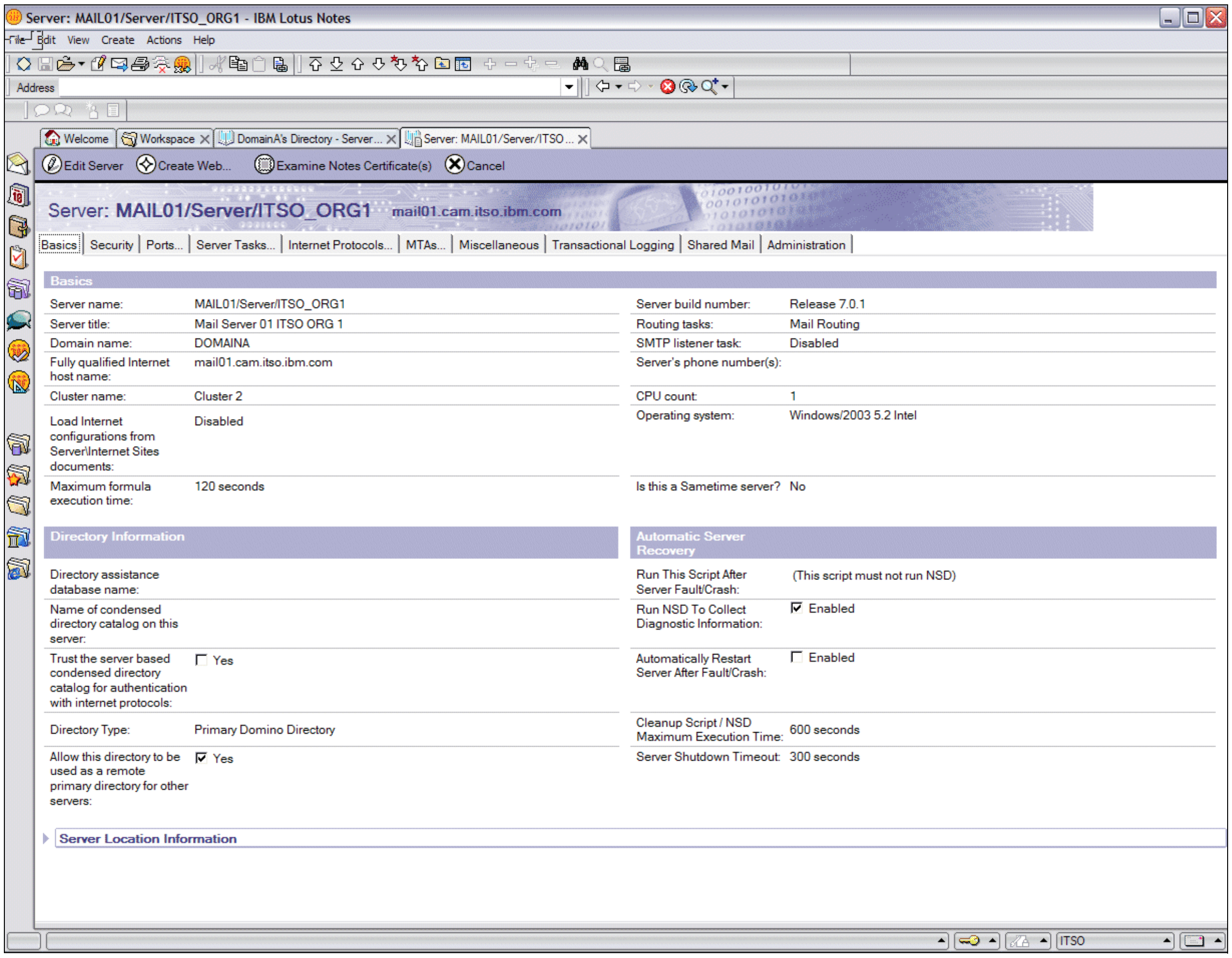
Choose Yes if you want to perform this action immediately. Choose No if you want to submit a request and have the Administration Process perform this action.

Figure 6-18 Creating the cluster using the Administration process

Attention: Use the Administration Process whenever possible. This automates the rest of the cluster creation and ensures the proper cluster setup across all servers within the Domino Cluster.

6.3.3 Checking the cluster creation and correct replication flow

To check cluster creation, determine whether it was logically created and whether the cluster traffic is being addressed by the right port (see Figure 6-19). This prevents the cluster traffic from creating unnecessary broadcasts in the public segment and helps it to concur with user access.


Server: MAIL01/Server/ITSO_ORG1 - IBM Lotus Notes

Server: MAIL01/Server/ITSO_ORG1 mail01.cam.itso.ibm.com

Basics

Server name:	MAIL01/Server/ITSO_ORG1	Server build number:	Release 7.0.1
Server title:	Mail Server 01 ITSO ORG 1	Routing tasks:	Mail Routing
Domain name:	DOMAINA	SMTP listener task:	Disabled
Fully qualified Internet host name:	mail01.cam.itso.ibm.com	Server's phone number(s):	
Cluster name:	Cluster 2	CPU count:	1
Load Internet configurations from Server\Internet Sites documents:	Disabled	Operating system:	Windows/2003 5.2 Intel
Maximum formula execution time:	120 seconds	Is this a Sametime server?	No

Directory Information		Automatic Server Recovery	
Directory assistance database name:		Run This Script After Server Fault/Crash:	(This script must not run NSD)
Name of condensed directory catalog on this server:		Run NSD To Collect Diagnostic Information:	☑ Enabled
Trust the server based condensed directory catalog for authentication with internet protocols:	☐ Yes	Automatically Restart Server After Fault/Crash:	☐ Enabled
Directory Type:	Primary Domino Directory	Cleanup Script / NSD Maximum Execution Time:	600 seconds
Allow this directory to be used as a remote primary directory for other servers:	☑ Yes	Server Shutdown Timeout:	300 seconds

Server Location Information

Figure 6-19 Server document of a cluster member

The Administration Process updates the Cluster name field with the name of the newly created Domino Cluster. If the configuration is not updated, confirm that the Administration Process already ran by opening the admin4.nsf database.

If the process has not run yet, you can manually force processing of the pending requests by issuing the following command in the administration server Domino Console:

```
tell adminp process all
```

Then you can replicate admin4.nsf to all the members of the Domino Cluster.

6.3.4 Cluster directory database

A replica of the Cluster Database Directory (CLDBDIR.NSF) resides on every server in a cluster. The Cluster Database Directory contains a document about each database and replica in the cluster, as well as such information as the database name, server name, path, and replica ID, and other replication and access information. The cluster components use this information to perform their functions, such as determining fail-over paths, controlling access to databases, and determining which events to replicate and where to replicate them to.

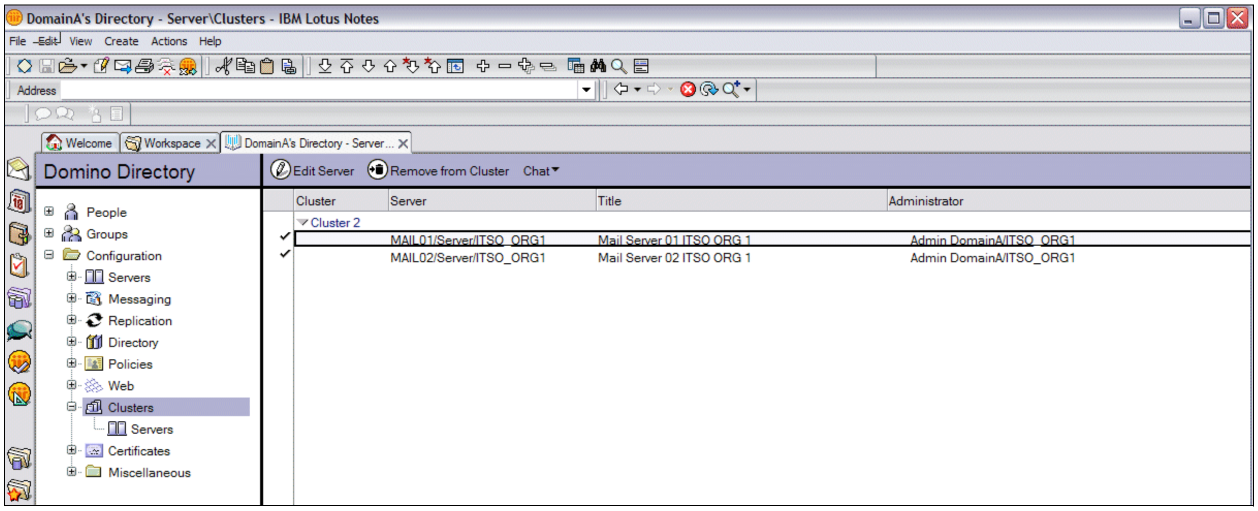
Note: Be sure to disable replication between system databases, unless you know exactly what you are doing. Most of the system databases can be replicated in a normal replication fashion and do not have to be synchronized as regularly as the cluster replication provides.

6.3.5 Removing a server from a cluster

Use these steps to remove a server from the Domino Cluster.

Note: Do this procedure in The Administration server in the Domino domain.

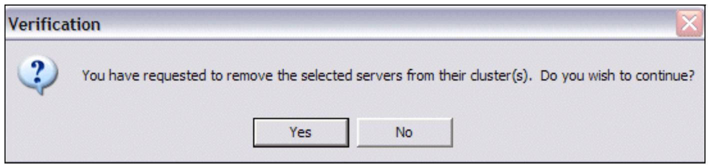
1. Select **Configuration** → **Clusters** in the Domino Directory database (names.nsf) for the Clusters view.
2. Select the servers to be removed from the Cluster, and click **Remove from Cluster** (see Figure 6-20).



Cluster	Server	Title	Administrator
Cluster 2			
✓	MAIL01/Server/ITSO_ORG1	Mail Server 01 ITSO ORG 1	Admin DomainA/ITSO_ORG1
✓	MAIL02/Server/ITSO_ORG1	Mail Server 02 ITSO ORG 1	Admin DomainA/ITSO_ORG1

Figure 6-20 Cluster view in Domino Directory

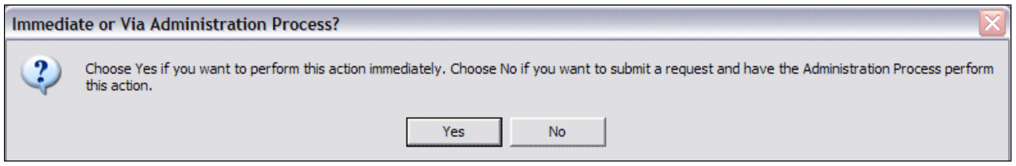
3. Click **Yes** to confirm (see Figure 6-21).


Verification

You have requested to remove the selected servers from their cluster(s). Do you wish to continue?

Figure 6-21 Delete from cluster confirmation window

4. Choose whether to delete the server from the cluster immediately or use the Administration Process (see Figure 6-22). We recommend using the Administration Process whenever possible.


Immediate or Via Administration Process?

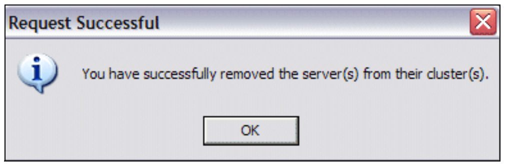
Choose Yes if you want to perform this action immediately. Choose No if you want to submit a request and have the Administration Process perform this action.

Figure 6-22 Delete immediately or using adminp

The cluster processes clrepl and cldbdir are stopped on the server, and the entries are removed from the ServerTasks line in notes.ini.

The database entries in the Cluster Directory database (cldbdir.nsf), for databases on the server being removed, are purged from the replica copy on the least busy of the remaining cluster servers.

The local Cluster Directory database is deleted from the server that is being removed from the cluster.


Request Successful

You have successfully removed the server(s) from their cluster(s).

Figure 6-23 Deletion request process success notice

6.3.6 Troubleshooting a cluster

Troubleshooting a cluster can be an exhaustive task because many variables are involved in a cluster setup. We recommended first trying to isolate the problem as best as possible, then working with the specific issue.

Cluster creation/deletion issues

Letting the Administration Process create the cluster for you is the best way to make sure that all cluster setup steps are performed correctly; however, because it relies on the Administration process, every time a problem is noticed, we recommend checking how the Administration Process is behaving across the domain. Be sure the admin4.nsf database is being replicated properly across all of the cluster members and that the Administration Process is running as well. You might need to run the Updall and Compact tasks against the admin4.nsf database at least weekly, because this database is being updated constantly.

Connectivity

Confirm that the members of the cluster can communicate issuing **trace** commands in the problematic server console to the other members in the cluster. Be sure to use the full Domino server names, and issue the same commands using the truly qualified name. Being able to connect through FQN is not required, but could indicate DNS issues:

```
mail01.cam.ibm.com
```

Make sure the servers are using the desired NIC selected as the dedicated cluster traffic handler.

Ensuring that cluster replication is using the private LAN

It is also a good idea to check the cluster to be sure that cluster replication is using the private LAN. To do this, you can look at some of the cluster statistics:

1. From the Domino Administrator or the Web Administrator, click the **Statistics** tab under the Server tab.
2. In the list of statistics, expand **NET** and the port name you gave to the cluster.
3. Look for the BytesReceived/BytesSent statistics for the private LAN port.
4. Expand **Replica** and **Cluster**.
5. Expand **SessionBytes**, and look for the In/Out replication statistics for the cluster.
6. Compare the NET.portname.BytesReceived value with the Replica.Cluster.SessionBytes.In value. These values should be fairly close to each other, although they will not be the same.
7. Compare the NET.portname.BytesSent value with the Replica.Cluster.SessionBytes.Out value. These values should also be close to each other. They will not match exactly because the private network is used for more than just cluster replication.

You can fine cluster troubleshooting tips in Administrator Help by selecting **Troubleshooting → Troubleshooting the Domino system → Clusters - Troubleshooting**.

6.4 Transaction logging

Transaction logging was introduced in Domino 5 and greatly improved in Domino 6. The basic architecture of transaction logging is to catch all the changes or transactions to each database and log them to a disk drive sequentially.

6.4.1 Overview of benefits of transaction logging

Transaction logging poses huge advantages for the Domino server. The advantages of transaction logging include:

- Quick and safe recovery from a server crash

  After a server crash, each database that was in use at the time of failure requires fixing up to ensure its integrity. Applying transaction logging will let the server skip this process, recovering the database directly from the transaction log instead. This will give you quick and safe recovery from a system failure.

  Furthermore, you can take advantage of the View Logging feature introduced in Domino 6, which provides transaction logging support for views and folders. Enabling this feature is especially effective for the Domino Directory $Users and $ServerAccess views, because these hidden views are crucial to the Domino system and also require a long time to rebuild after a crash if you are not using this view logging feature.

- Performance benefit

  Because transaction logging writes transactions sequentially to disk, it can help reduce the bottleneck of disk I/O. If transaction logging is not implemented, the Domino server must write transactions directly to each database, requiring the disk arm to have random access, which is much slower than sequential access. If you use transaction logging, the Domino server will write transaction logging first and update these transaction later in the each database.

Note that the capacity of disk drives for transaction logging is critical to performance, and they must be physically dedicated to transaction logging to make the most efficient use of sequential writing. We recommend that you have separate physical disk drives for transaction logging, data directory, and the paging file of the operating system.

Finally, the main benefit of transaction logging is availability and reliability of the server. Any administrator who has waited a long time for a system that crashed to restart will immediately appreciate transaction logging. Just as journaling improves the integrity of the file system, Domino transaction logging improves the integrity of Domino databases.

Note: A transaction is a related series of changes made to a database on a server. For example, opening a new document, adding text, and saving the document is one transaction. In this case, the transaction consists of three separate implicit API calls: NotesOpen, NoteUpdate, and NoteClose.

6.4.2 Transaction logging defined

A transaction log is a record of changes made to Notes databases. The transaction log consists of log extents and the log control file (NLOGCTRL.LFH). A log extent is one of the log files into which the transaction logs are written. It has the form Sxxxxxxx.TXN, where the x characters represents a seven-digit number that is unique to that server. Domino fills each extent sequentially before writing data to a new one. The records are secured using a proprietary byte-stream format. Each server has only one transaction log that captures all the changes to databases that are enabled for transaction logging.

Use transaction logging to:

- Schedule regular backups. Backups based on transaction logs are faster and easier than full database backups that do not use transaction logging.
- Recover from a media failure. If you have a media failure, you can restore the most recent full backup from tape, then use the transaction logs to add the data that was not written to disk.
- Recover from a system crash. When the server restarts, it runs through the end of the transaction logs and recovers any writes that were not made to disk at the time of the crash. Logged databases do not require a consistency check, which speeds up the restore time of the server.
- Log the database views. You can avoid most view rebuilds.

To use all the features of transaction logging for backups and backup recovery, you need a third-party backup utility that uses the backup and recovery methods of the Domino C API Toolkit (Release 5 or later). For example, in the case of a media recovery, a database backup is made with the third-party utility, while logging keeps track of updates to the database. When the database is then lost, the backup is brought up to its current state by going through the transaction log and applying any updates that have happened to that databases since the database backup was taken.

Note that restart recovery does not require a third-party utility. In this case, logging goes on while updates are happening. When the server crashes and then restarts, any updates that would have otherwise been lost are written to the database. This significantly reduces lost data and database corruption because of server crashes, and reduces overall restart time, since the consistency check of databases is not required.

Note: Domino supports transaction logging for servers that run Domino 5 and later, and for databases that are in a Domino 5 or later on-disk structure.

6.4.3 Understanding the database instance ID (DBIID)

When you enable transaction logging, Domino assigns a unique database instance ID (DBIID) to each Domino database. When Domino records a transaction in the log, it includes this DBIID. During recovery, Domino uses the DBIID to match transactions to databases.

Some database maintenance activities, such as using the Compact command with options, cause Domino to reconstruct the database in such a way that old transaction log records are no longer valid. When this happens, a new DBIID is assigned to the database. From that point on, all new transactions recorded in the log for that database use the new DBIID. After a database is assigned a new DBIID, take a new full backup of the database. The new full backup captures the database in its current state with the new DBIID. Then, if you have to restore the database, Domino needs only the new transactions that contain the new DBIID.

Domino assigns a new DBIID when:

- You enable transaction logging for the first time.
- You run the Compact task with an option, for example, the option to reduce file size.
- You run the Fixup task on corrupted databases.
- You move a Domino database to a logged server.

6.4.4 Logging styles

There are three logging styles to choose from:

- With *circular* logging, Domino reuses a fixed amount of disk space (up to 4 GB) for transaction logs. After the disk space is used up, Domino starts overwriting old transactions, starting with the oldest. When the space fills up, perform a backup on the databases. You may need to do daily backups to capture database changes before they are overwritten, depending on the server activity level. Use circular logging if the size of the log needed between full database backup intervals is less than 4 GB.
- *Linear* logging is like circular logging, except it allows more than 4 GB. Use linear logging if the size of the log needed between full database backup intervals is greater than 4 GB, and you are not using archive media.
- *Archived* logging creates log files as needed. It simplifies backup and restoration, and provides online and partial backups. The log files are not overwritten until you archive them. With archived logging, you must have a backup utility to back up the filled log extents so that they are ready if needed. If you do not have a backup utility, the server continues to create log extents, fills up the disk space, and then panics.

6.4.5 Planning transaction logging

Transaction logging captures all the changes that are made to databases and writes them to a transaction log. The logged transactions are written to disk in a batch when resources are available or at specified intervals.

Use this checklist for your transaction logging planning.

- Allocate space for the log files. Use a dedicated, mirrored device, such as RAID level 1 with a dedicated controller for optimal performance and data integrity.
- Plan a backup strategy. Plan to archive the transaction logs daily using incremental backups. Schedule weekly full database backups. You will then be prepared if you have a media failure.
- Decide which servers and databases will use transaction logging. Transaction logging is available for servers running Domino 5 and later. Consider enabling transaction logging for all databases on the server.
- Select a Domino-compatible backup utility. The utility must be able to use the backup and recovery methods of the Domino C API Toolkit (Release 5 or later).

6.4.6 Transaction logging & disk I/O tuning

If you are not going to use your transaction logs in conjunction with a Domino-aware backup system, all you need is a pair of 4 GB hard disk drives configured for RAID 1. You need to dedicate these drives to the Domino Transaction Log to avoid a performance degradation.

Ideally, everything, from the OS to Swap to Domino, would have its own hard disk drives utilizing multiple RAID controllers, but this is an unlikely scenario for all but very high-end servers. When possible, use hardware-based RAID 1 for the OS and Swap, another RAID 1 for the transaction logs, and RAID 5 for the Domino data directory due to the many read operations.

Given the advantages of transaction logs in achieving data integrity and the substantially faster restarts of your server, we recommend enabling transaction logging even if you only have a single, hardware-based RAID 1 for OS, Swap, and the transaction logs, and a RAID 5 for the data directory. This is admittedly not an ideal scenario, since the Swap and transaction logs will be competing for I/O resources. Unlike NT, however, Windows does rely heavily on Swap. As long as your server has enough memory, this should be a suitable configuration. Be certain to test this scenario, however, in order to make certain that you are content with the resulting performance.

Ensure that the transactional logs have a *separate controller* so that congestion is avoided. If not, it might work fine, but with increasing load on the server, the transaction log will have to compete with other I/O operations of the Domino server and eventually not have enough. Since the transaction logging is an overall Domino server performance / health issue, you do not want these problems down the line in a year or two, and to be looking for them and finally identifying this as being a controller issue. Install that separate controller from the beginning when planning for transaction logging.

How about using a RAID 5 configuration for the entire server? While RAID 5 is an option for getting the most out of your hard disk drives, it is not suitable for the entire server, especially with transaction logging enabled (see also 6.6, “Redundant Arrays of Independent Disks (RAID)” on page 110). Remember that when writing to disk, RAID 5 will need to read data from the disk in order to recalculate the parity, except when performing a *full-stripe-write* in which the data is already in the cache. The additional I/O overhead from *partial-stripe-write* and *read-modify-write* operations results in RAID 5 exhibiting slower write performance than

RAID 1. Read performance is typically slower as well, since RAID 1 offers two drives from which data may be read. Although RAID 5 offers more drives for simultaneous reads, the requests would have to be ideally broken up so that no two requests ever needed information on the same drive. This is because in RAID 5, the data is not mirrored (redundancy is provided by the parity stripe) and so there is only one location from which data can be read. RAID 5, preferably through hardware, not software, should be utilized only for the Domino data directory. If you have to use RAID 5 for the entire server, you should not enable transaction logging.

Attention: Please understand that because we have mentioned RAID 5 in the above text does not mean in any way that RAID 5 is the best RAID technology for a Domino server. We do not endorse RAID 5 over the other types, since the RAID type chosen is dependant on your organization's requirements and needs. For more details on RAID technology and configurations, please refer to 6.6, "Redundant Arrays of Independent Disks (RAID)" on page 110.

6.4.7 Notes.ini parameters for transaction logging

Here we discuss the notes.ini parameters for transaction logging.

TransLog_Performance=value

The following notes.ini setting specifies the trade-off between transactional log runtime and restart recovery time, as follows:

- 1 - Favor runtime. The system stores more database changes in memory writes fewer changes to the transaction log. Fewer writes to disk improves server runtime.
- 2 - Standard (default)
- 3 - Favor restart recovery time. The system stores fewer database changes in memory and writes more changes to the transaction log. More writes to the transaction log improves restart recovery time.

TransLog_Path=path

Specifies the path to the transaction log. The default location is \logdir in the server's data directory.

However, we strongly recommend storing the transaction log on a separate mirrored device, such as a RAID level 0 or 1 device with a dedicated controller.

If you change this field and have an existing transaction log, you must use the operating system to move all the log files to the new log path.

TransLog_MaxSize=number of megabytes

The maximum size, in MB, for the transaction log. A value of at least 192 MB is recommended. If you do not specify a value, the system determines a log size approximately three times the size of the server's RAM.

TransLog_Status=value

Enables transaction logging for all Domino 5 databases on the server, as follows:

- 0 - Disabled (Default)
- 1 - Enabled

You must upgrade the databases to the Domino 5 format before they can use transaction logging.

TransLog_Style=value

Specifies the type of transaction logging. The options are as follows:

- 0 - Circular (default). The system continuously reuses the extent log files, overwriting old transactions.
- 1 - Archive. The system does not reuse extent log files and allows you to use a backup utility to archive log files. This is recommended.

MailBoxDisableTDNLogging=value

Allows mail.boxes created by the server during startup to be created with transaction logging disabled when set to any non-zero value (Default).

TransLog_UseAll=value

Specifies whether or not to use all available disk space on the log device, as follows:

- 0 - (Default). The system uses the default or specified value in "TransLog_MaxSize".
- 1 - Use all available space on the disk for the transaction log extent. This is recommended if you use a separate device dedicated to storing the extent.

6.5 Hard disk performance

Hardware has improved during the years, and so have the hard disk mechanisms within systems. Even though they are still important in terms of performance, they can also be a determining factor of how stable and reliable your Domino server is.

CPU and memory are also important to stability, and often receive more attention when talking about hardware. Typically, servers load software and load and save data frequently. All of these operations require frequent reading / writing to the disk. Therefore, hard disk performance and reliability is a critical issue.

6.5.1 Performance equation

The importance of the CPU, memory, and other core components is very important. Much as the strength of a chain is equal only to that of its weakest link, in many ways the performance of a system is only equal to that of its poorest-performing component. Compared to the solid state components in a server, hard disks have, by far, the worst performance. Hard disks are improving in speed, but so are CPUs, memory and motherboards, and these other components tend to get faster more often, therefore widening the gap. Therefore, hard disks are still the overall main constraint of the performance of servers.

Figure 6-24 on page 109 shows how storage is just one of the many components of a Domino solution.

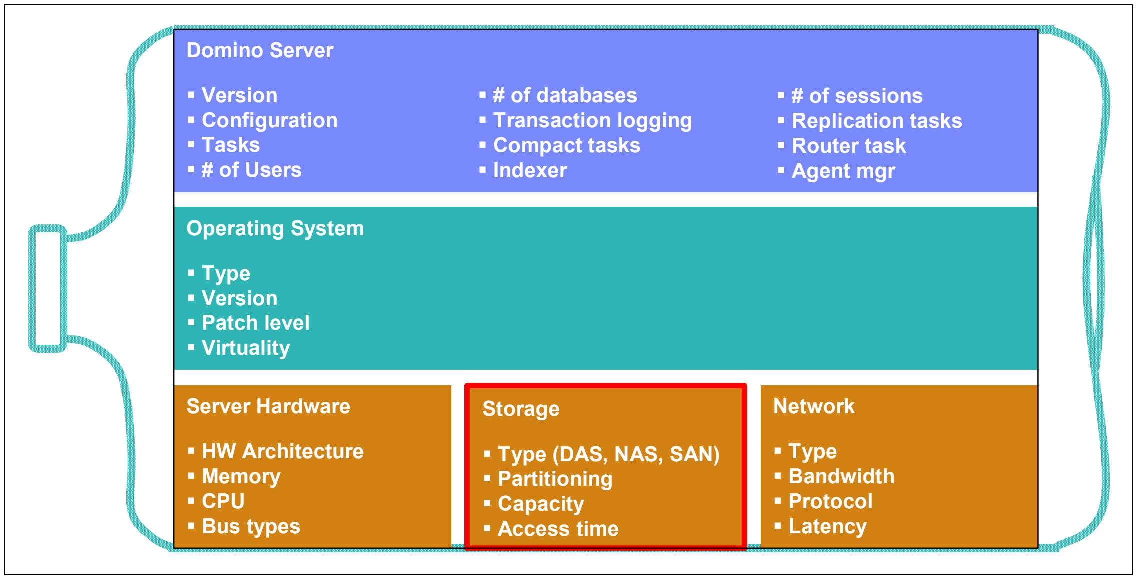


Figure 6-24 The weak storage link

In this section, we will focus mainly on the hard disk performance, as this can eventually constrain the overall performance of the Domino server. Even the best and most optimal Domino server can be crippled if the hard disks are misconfigured.

6.5.2 Hard disk performance: keeping speed in perspective

Given the amount of time it takes to perform one random access cycle to the hard disk, today's processors can execute an incredible amount of instructions in that time. The processors are developed to become even faster with a speed that is breathtaking; however, this is not really of much benefit if the hard disks are still the bottleneck. Improving the hard disk's speed to reduce the wait time will improve the overall performance of your server.

The speed of the hard disks becomes very important for applications where hard disk operations are intensive. The Domino server is disk intensive in terms of reading and writing to the hard disk. This is also referred to as *I/O bound*.

If the main limiting factor in a certain hardware setup is the system bus being used for the hard disk interface, putting a faster hard disk into that system will have very little impact on performance. On the other hand, if the hard disks are slow, adding a faster controller will not improve the overall performance, since the disks are the bottleneck.

It is important to remember that the performance of hard drives is not exactly the same when performing an read operation as opposed to a write operation on the disk. Some operations are identical, but the seek time is different as opposed to write instructions.

The hard drive has to locate the exact location on the drive for the I/O operations. The time used for these operations can be very different for two different drive types.

Note: Comparing different types of drives, even from the same manufacturer, should be conducted with caution, as it is often hard to compare the figures in question. Almost all performance specifications given for hard disks are based upon how the hard disk performs while reading, not while writing.

The need for improved performance and data security is a major driving factor behind the rise in popularity of RAID systems in server configurations.

6.6 Redundant Arrays of Independent Disks (RAID)

Originally referred to as Redundant Arrays of Inexpensive Disks, RAID is a way of storing the same data in different places on multiple hard disks. The data is placed on multiple disks and the I/O operations can overlap in a balanced way. This setup will improve performance and stability compared to a single hard drive solution.

The fundamental principle behind RAID is the use of multiple hard disk drives in an array that behaves like a single large hard drive. There are a number of ways that this can be done, depending on your needs, but in every case the use of multiple drives allows the resulting storage subsystem to exceed the capacity, data security, and performance of the drives that make up the system, to one extent or another.

Important: Using RAID adds to your complexity and costs, so these are trade-offs you will have to consider.

6.6.1 RAID benefits

A key benefit of RAID is the use of redundancy; most RAID levels provide protection for the data stored on the array. Through redundancy, the array of hard disks can withstand even the complete failure of one hard disk (or sometimes more) without any data loss. All RAID levels, except RAID level 0, provide some degree of data protection, which depends on the exact implementation.

Due to the increased number of disks making up the RAID, the performance is improved, because the server has more "spindles" to read from or write to when data is accessed. This is a way to get around the performance-limiting mechanical issues that plague individual hard disks.

Availability is increased because the RAID controller can recreate lost data from parity information. Parity is basically a checksum of the data that was written to the disks, which gets written along with the original data. The controller recreates lost data when the drive went corrupt by using the parity information stored on the surviving disks in the RAID array.

6.6.2 RAID costs

For a decent-sized RAID system, you must allow for resources for planning what type of RAID will be implemented, deciding on array size, choosing hardware, and so on.

Hardware costs include hard disks, enclosures, power supplies, power protection, and possibly a hardware RAID controller. The costs in terms of your hard disks is dependent on the amount of space for your files. You will eventually have to buy the hard disks, regardless. RAID only costs extra in this regard to the extent that additional drives must be purchased for redundancy. When implementing RAID 0, there is no additional disk drive cost, unless you are buying extra-large drives to cover future needs.

Maintenance is another component to think about when talking about RAID, since they require ongoing maintenance if they are to continue to provide the organization with high availability and performance.

6.6.3 RAID trade-offs

One can easily get something that:

- Inexpensive and fast, but at the cost of quality
- High quality and fast, but it will not be cheap
- Cheap and high quality, but with lower performance

The triangle between performance, cost, and reliability are obviously seen when contrasting the different RAID levels. Many people chose performance over fault tolerance, others do the opposite, while a third group tries to balance the entire setup. Some RAID systems cost more than others, where others offer very high fault tolerance but relatively low capacity.

6.6.4 Important terms

Here we define some important terms that we will use in our discussion of RAID.

Disk striping

As mentioned in 6.6, “Redundant Arrays of Independent Disks (RAID)” on page 110, data striping transparently distributes data over multiple disks to make them appear as a single, fast, and large disk. Striping improves the I/O performance by servicing the disks in parallel through the increased number of spindles.

There are two types of disk striping: single user and multi-user. Single user disk striping allows multiple hard disks to simultaneously service multiple I/O requests from a single workstation. Multi-user disk striping allows multiple I/O requests from several workstations to be sent to multiple hard disks. This means that while one hard disk is servicing a request from a workstation, another hard disk is handling a separate request from a different workstation. This decreases the queuing time seen by I/O requests, which is very important for Domino, since it is a I/O intensive application. The performance benefits increase with the number of disks in the array. However, a large number of disks lowers the overall reliability of the disk array.

Disk striping is used with or without parity. When disk striping is used with parity, an additional stripe that contains the parity information is stored on its own partition and hard disk. If a hard disk fails, a fault tolerance driver makes the lost partition invisible, allowing reading and writing operations to continue which provides time to create a new stripe set.

Disk mirroring (redundancy)

In this technique, data is written to two duplicate disks simultaneously. In this way, if one of the disk drives fails, the system can switch to the other disk without any loss of data or service.

Since a larger number of disks lowers the overall reliability of the array of disks, it is important to incorporate redundancy in the array of disks to tolerate disk failures and allow for the continuous operation of the system without any loss of data or service.

As already mentioned, the selection between the many possible data striping and redundancy schemes involves complex trade-offs. This is the triangle of reliability, performance, and cost.

6.6.5 RAID 0: striped disk array

RAID 0, also known as a *striped set*, splits data evenly across two or more disks with no parity information for redundancy. It is important to note that RAID 0 was not one of the original RAID levels, and is *not redundant*. RAID 0 is normally used to increase performance, although it can also be used as a way to create a small number of large virtual disks out of a large number of small physical ones. A RAID 0 can be created with disks of differing sizes, but the storage space added to the array by each disk is limited to the size of the smallest disk. Although RAID 0 was not specified in the original RAID paper, an idealized implementation of RAID 0 would split I/O operations into equal-sized blocks and spread them evenly across two disks.

Important: RAID 0 implementations with more than two disks are also possible; however, the reliability of a given RAID 0 set is equal to the average reliability of each disk divided by the number of disks in the set.

If all accessed hard disk sectors are entirely on one disk, then the apparent seek time would be the same as a single disk. If the accessed sectors are spread evenly among the disks, then the apparent seek time would be reduced by half for two disks, by two-thirds for three disks, and so on, assuming identical disks. For normal data access patterns, the apparent seek time of the array would be between these two extremes. The transfer speed of the array will be the transfer speed of all the disks added together, limited only by the speed of the RAID controller

Figure 6-25 gives an overview of RAID 0.

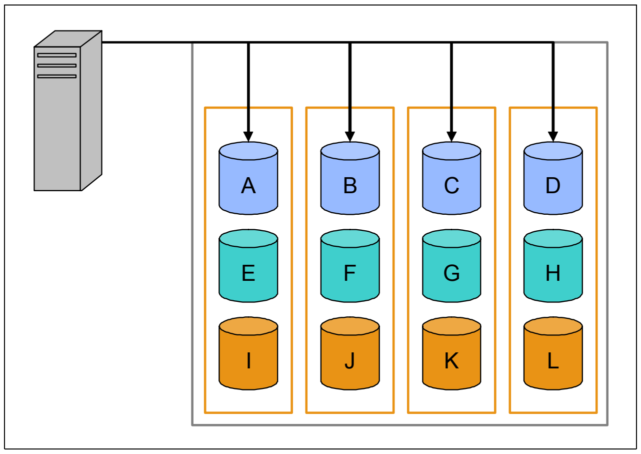


Figure 6-25 RAID 0: striped disk array

Attention: RAID 0 improves performance, but does not deliver fault tolerance. If one drive fails, then all data in the array is lost.

6.6.6 RAID 1: mirroring and duplexing

RAID 1 creates an exact copy (or mirror) of a set of data on two or more disks. This is useful when read performance is more important than data capacity. Such an array can only be as big as the smallest disk. A classic RAID 1 mirrored pair contains two disks, which increases reliability exponentially over a single disk. Since each member contains a complete copy of the data, and can be addressed independently, ordinary wear-and-tear reliability is raised by the power of the number of self-contained copies. For example, consider a model of disk drive with a weekly probability of failure of 1:500. Assuming defective drives are replaced weekly, a two-drive RAID 1 installation would carry a 1:250,000 probability of failure for a given week. That is, the chances that both drives experience an ordinary mechanical failure during the same week is the square of the chances for only one drive.

Figure 6-26 gives an overview of RAID 1.

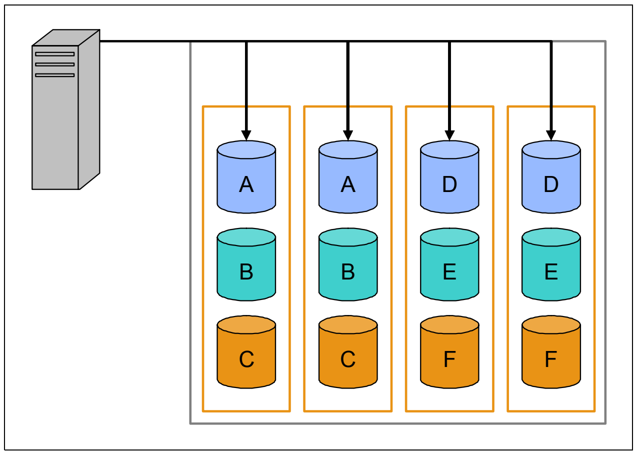


Figure 6-26 RAID 1: mirroring and duplexing

Additionally, since all the data exists in two or more copies, each with its own hardware, the read performance goes up roughly as a linear multiple of the number of copies. That is, a RAID 1 array of three drives can be reading in three different places at the same time. To maximize the performance benefits of RAID 1, independent disk controllers are recommended, one for each disk. This is also referred to as splitting or duplexing. When reading, both disks can be accessed independently. Like RAID 0, the average seek time is reduced by half when randomly reading, but because each disk has the exact same data, the requested sectors can always be split evenly between the disks and the seek time remains low. Write performance is the same as for single disk storage. The transfer rate would also be doubled. For three disks, the seek time would be a third and the transfer rate would be tripled. The only limit is how many disks can be connected to the controller and its maximum transfer speed.

6.6.7 RAID 2: striping and error checking correction

RAID 2 uses striping across disks with some disks storing error checking and correcting (ECC) information. This is not a typical implementation and rarely used. Data is striped at the bit level rather than the block level and uses a Hamming code for error correction. The disks are synchronized by the controller to run in perfect tandem.

With RAID 2, extremely high data transfer rates are possible.

6.6.8 RAID 3: byte-level striping / interleaved parity

RAID 3 uses byte-level striping with a dedicated parity disk. RAID 3 is very rare in practice. One of the side effects of RAID 3 is that it generally cannot service multiple requests simultaneously. This comes about because any single block of data will, by definition, be spread across all members of the set and will reside in the same location, so any I/O operation requires activity on every disk.

In our example below, a request for block "A", consisting of bytes A1-A3, would require all three data disks to seek to the beginning (A1) and reply with their contents. A simultaneous request for block B and or C would have to wait.

Figure 6-27 gives an overview of RAID 3.

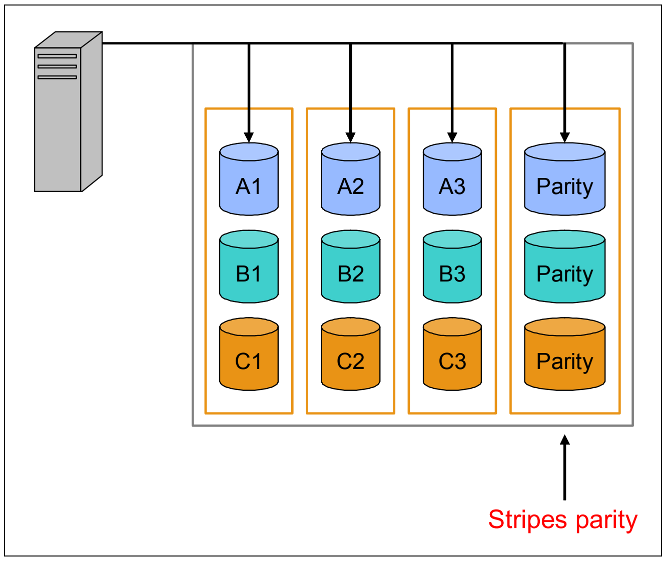


Figure 6-27 RAID 3: byte-level striping / interleaved parity

6.6.9 RAID 4: block-level striping / dedicated parity drive

A commonly used implementation of RAID 4, dedicated parity drive, provides block-level striping (like Level 0) with a parity disk. RAID 4 looks similar to RAID 3 except that it stripes at the block rather than the byte level. If a data disk fails, the parity data is used to create a replacement disk. RAID 4 uses large stripes, which means you can read records from any single drive. This allows you to take advantage of overlapped I/O for read operations. Since all write operations have to update the parity drive, no I/O overlapping is possible. A disadvantage to RAID 4 is that the parity disk can create write bottlenecks.

Figure 6-28 on page 115 gives an overview of RAID 4.

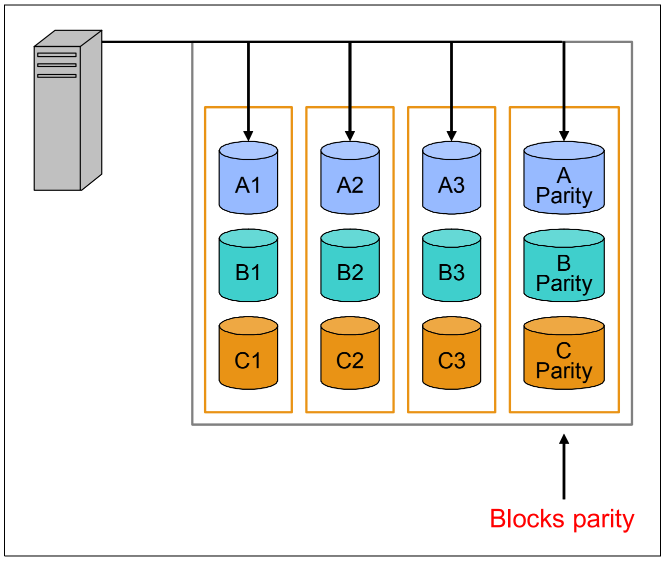


Figure 6-28 RAID 4: block-level striping / dedicated parity drive

6.6.10 RAID 5: data striping / block interleaved distribution parity

Block interleaved distributed parity includes a rotating parity array, thus addressing the write limitation in RAID 4. RAID 5 stores parity information, not redundant data, and with that, the hard disks data can be reconstructed. RAID 5 provides data striping at the byte level, distributed across all member disks, and also stripe error correction information. This results in excellent performance and good fault tolerance. RAID Level 5 is *one of the most popular* implementations of RAID.

Every time a block is written to a disk in a RAID 5, a parity block is generated within the same stripe. The block is often composed of many consecutive sectors on a disk. A series of blocks (a block from each of the disks in an array) is collectively called a "stripe". If another block, or some portion of a block, is written on that same stripe, the parity block (or some portion of the parity block) is recalculated and rewritten. For small writes, this requires reading the old data, writing the new parity, and writing the new data. The disk used for the parity block is staggered from one stripe to the next, hence the term "distributed parity blocks".

Important: RAID 5 writes are expensive in terms of disk operations and traffic between the disks and the controller.

The parity blocks are not read on data reads, since this would be unnecessary overhead and would diminish performance. The parity blocks are read, however, when a read of a data sector results in a cyclic redundancy check (CRC) error. In this case, the sector in the same relative position within each of the remaining data blocks in the stripe, and within the parity block in the stripe, are used to reconstruct the errant sector. The CRC error is thus hidden from the main computer. Likewise, should a disk fail in the array, the parity blocks from the surviving disks are combined mathematically with the data blocks from the surviving disks to reconstruct the data on the failed drive through Interim Data Recovery Mode.

The computer knows that a disk drive has failed, but this is only so that the operating system can notify the administrator that a drive needs replacement; applications running on the computer are unaware of the failure. Reading and writing to the drive array continues seamlessly, though with some performance degradation. The difference between RAID 4 and RAID 5 is that, in interim data recovery mode, RAID 5 might be slightly faster than RAID 4, because, when the CRC and parity are in the disk that failed, the calculation does not have to be performed, while with RAID 4, if one of the data disks fails, the calculations have to be performed with each access.

Important: In RAID 5, where there is a single parity block per stripe, the failure of a second drive results in total data loss.

The maximum number of drives in a RAID 5 redundancy group is theoretically unlimited, but it is a common practice to limit the number of drives. The trade-offs of larger redundancy groups are a greater probability of a simultaneous double disk failure, the increased time to rebuild a redundancy group, and the greater probability of encountering an unrecoverable sector during RAID reconstruction. As the number of disks in a RAID 5 group increases, the Mean Time Between Failure (MTBF) can become lower than that of a single disk. This happens when the likelihood of a second disk failing out of (N-1) dependent disks, within the time it takes to detect, replace, and recreate a first failed disk, becomes larger than the likelihood of a single disk failing.

Figure 6-29 gives an overview of RAID 5.

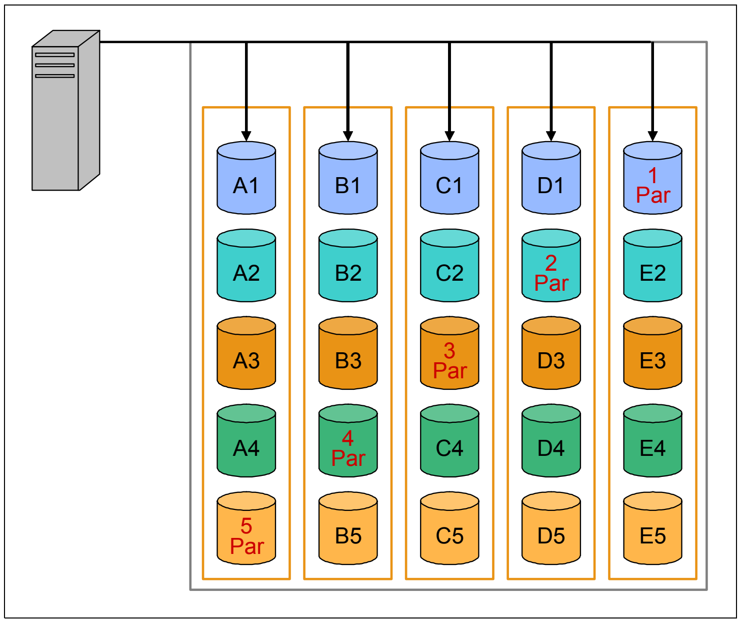


Figure 6-29 RAID 5: data striping / block interleaved distribution parity

Important: RAID 5 implementations suffer from poor performance when faced with a workload that includes many write operations

RAID 5 implementations *suffer from poor performance* when faced with a workload that includes *many writes that are smaller than the capacity of a single stripe*; this is because parity must be updated on each write, requiring read-modify-write sequences for both the data block and the parity block.

6.6.11 RAID 6: block level striping / double parity

RAID 6 extends RAID 5, but includes a second block-level parity striping scheme that is distributed across different drives and therefore offers extremely high fault- and drive-failure tolerance. RAID 6 was not one of the original RAID levels.

Like RAID 5, the parity is distributed in stripes, with the parity blocks in a different place in each stripe.

RAID 6 is inefficient when used with a small number of drives, but as arrays become bigger and have more drives, the loss in storage capacity becomes less important and the probability of two disks failing at once is bigger.

Important: RAID 6 *provides protection against double disk failures* and failures while a single disk is rebuilding.

In the case where there is only one array, it makes more sense than having a "hot spare" disk.

Figure 6-30 gives an overview of RAID 6.

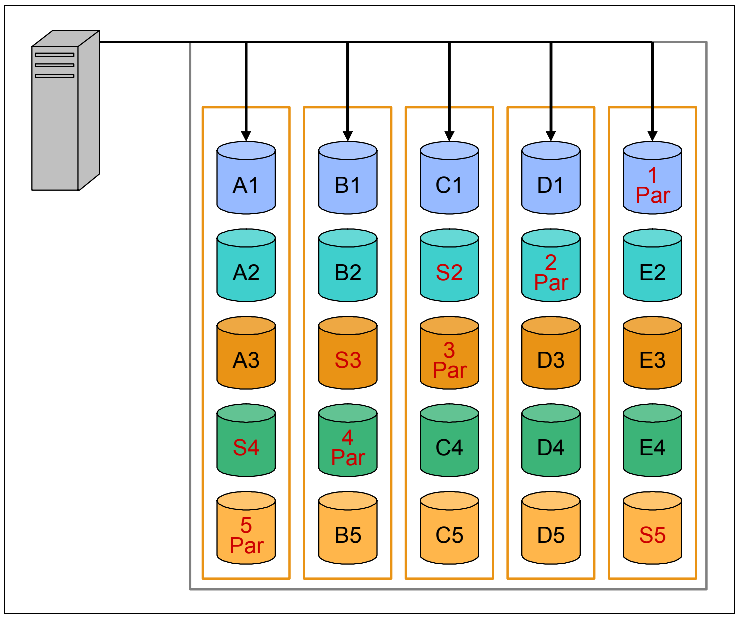


Figure 6-30 RAID 6: block level striping / double parity

The user capacity of a RAID 6 array is n-2, where n is the total number of drives in the array.

RAID 6 does not have a performance penalty for read operations, but it does have a performance penalty on write operations due to the overhead associated with the additional parity calculations.

Important: RAID 6 implementations suffer from poor performance when faced with a workload that includes many write operations due to the additional parity calculations.

6.6.12 Nested RAID levels

Many storage controllers allow RAID levels to be nested. That is, one RAID can use another as its basic element, instead of using physical disks. It is instructive to think of these arrays as layered on top of each other, with physical disks at the bottom.

Nested RAIDs are usually signified by joining the numbers indicating the RAID levels into a single number, sometimes with a '+' in between. For example, RAID 10 (or RAID 1+0) conceptually consists of multiple level 1 arrays stored on physical disks with a level 0 array on top, striped over the level 1 arrays. In the case of RAID 0+1, it is most often called RAID 0+1 as opposed to RAID 01 to avoid confusion with RAID 1. Opposed to this, when the top array is a RAID 0 (such as in RAID 10 and RAID 50), most vendors choose to omit the '+', probably because RAID 50 sounds fancier than the more explicit RAID 5+0.

When nesting RAID levels, a RAID type that provides redundancy is typically combined with RAID 0 to boost performance. With these configurations, it is preferable to have RAID 0 on top and the redundant array at the bottom, because fewer disks then need to be regenerated when a disk fails. Therefore, RAID 10 is preferable to RAID 0+1, but the administrative advantages of splitting the mirror of RAID 1 would be lost.

6.6.13 RAID 0+1: mirror of stripes

RAID level 0+1 is a mirror of stripes where two RAID 0 stripes are created, and a RAID 1 mirror is created over them. It is used for both replicating and sharing data among disks.

A RAID 0+1 is a RAID used for both replicating and sharing data among disks. The difference between RAID 0+1 and RAID 1+0 is the location of each RAID system; it is a mirror of stripes.

The advantage is that when a hard drive fails in one of the level 0 arrays, the missing data can be transferred from the other array. However, adding an extra hard drive to one stripe requires you to add an additional hard drive to the other stripes to balance out storage among the arrays.

It is not as robust as RAID 10 and *cannot tolerate two simultaneous disk failures*, if not from the same stripe. That is, once a single disk fails, each of the mechanisms in the other stripe is a single point of failure. Also, once the single failed mechanism is replaced, in order to rebuild its data, all the disks in the array must participate in the rebuild.

The layout of the blocks for RAID 0+1 and RAID 10 are identical, except that the disks are in a different order.

Figure 6-31 on page 119 gives an overview of RAID 0+1.

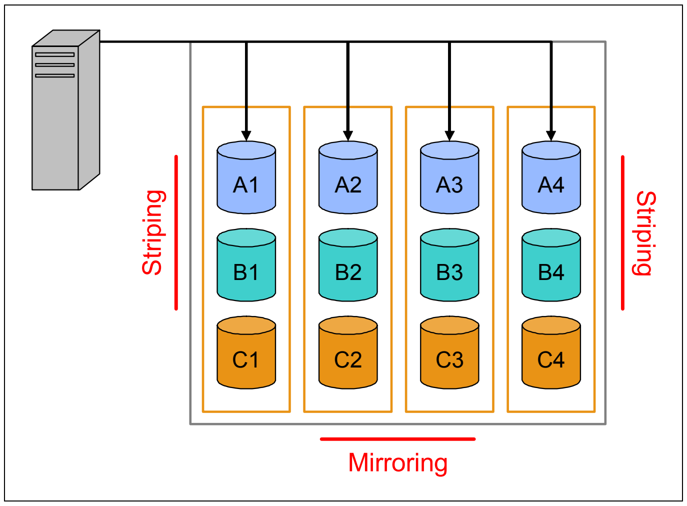


Figure 6-31 RAID 0+1: mirror of stripes

6.6.14 RAID 10: striped and mirrored disk array

Combining RAID 0 and RAID 1 is often referred to as RAID 10, which offers higher performance than RAID 1 but at much higher cost. There are two subtypes within this RAID level:

- RAID 0+1 data is organized as stripes across multiple disks and the striped disk sets are mirrored.
- RAID 1+0 data is mirrored and the mirrors are striped.

Both subtypes are similar and the difference is that the RAID levels used are reversed.

RAID 10 is often the primary choice for high-load databases, because the lack of parity to calculate gives it faster write speeds.

Figure 6-32 gives an overview of RAID 10.

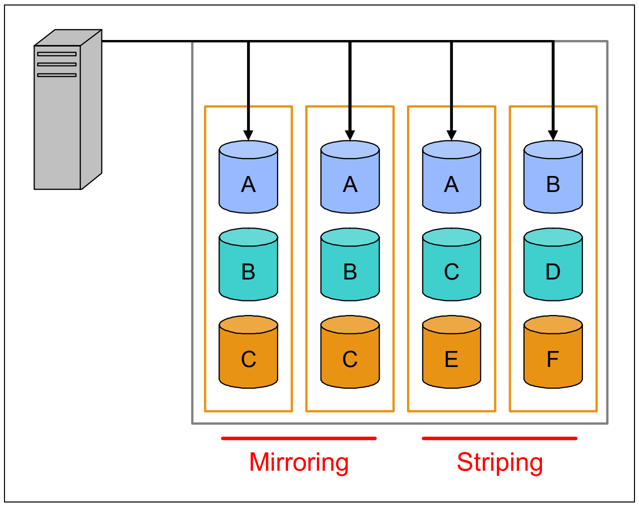


Figure 6-32 RAID 10: striped and mirrored disk array

Important: This must be the primary choice for Domino servers hosting high-load applications, because the lack of parity to calculate gives it faster write speeds.

6.6.15 Proprietary RAID levels

Although all implementations of RAID differ from the idealized specification to some extent, some companies have developed entirely proprietary RAID implementations that differ substantially from the rest on the market.

Therefore, it is important to understand which kind of RAID system you are about to implement and, as for the more proprietary ones, they are out of scope of this Redpaper.

Combinations are typically denoted with two digits. For example, RAID 50 is a combination of RAID 5 and RAID 0, and is sometimes noted as RAID 5+0. As another example, RAID 10 is actually RAID 1 and RAID 0 implemented together, or RAID 1+0.

6.6.16 Questions to ask when considering RAID

In the following section, you will find some questions to ask yourself and others in your organization when considering a RAID. This is a complex decision, but it is our hope that we have given you the necessary knowledge to make a more informed decision.

Here are some of the variables that need to be considered when determining the optimum raid level.

- Nature of the I/O? (Random or sequential)
- Read/write ratio? (80/20 or 60/40)
- Type of disks are being used? (ATA, SATA, FC, or solid state)
- Size are the disks? (73 GB, 146 GB, 300 GB, or larger)
- Rotational speed of the disks? (5400, 10 K, or 15 K)
- Cache is available?
- OS? (Windows, Solaris, Linux, or AIX?)
- Type of Domino databases on the server?
- Stripe size used by the OS?
- Drives per RAID group? (4, 8, or 16)
- RAID type? (0, 1, 1+0, 0+1, 5, 5+0, or 10+0)
- Software- or hardware-based RAID?
- Number of spindles?
- Host Bus Adapters (HBAs) in the server?
- Fabric design? (Star, Core, or Meshed)

6.6.17 Domino server types and RAID technology

Implementing RAID in your organization must be carefully done, as there are many topologies to chose from. RAID 5 is very popular and is often the default, because it has become an unofficial standard and easy to implement.

Actually, the RAID system used should be determined by your needs matched against the Domino server type. Not all Domino servers goes well with RAID 5 "as is" out of the box, since they have different scope and functions within the infrastructure (see 2.2, “Considerations for different types (roles) of Domino servers” on page 15 for more information).

Selecting the right RAID level for Domino servers really comes down to their functionality and especially whether they will have many or fewer I/O operations. This calls for the distinction between server functions like:

- Is it an application server?
  - Is it an application server under load by users?
    - Are the users heavy users or light users working all day in applications hosted on the particular server?
    - Are there many write operations in the applications or does the users read more content than write it?
  - Is it an application server under load by replication?
  - Is it an application server hosting business critical applications?
- Is it an mail server?
  - Is it used heavily by users during the entire day?
    - Is there a peak hour pattern?
    - Are there large or medium sized mail databases?
  - Is the mail server regarded as business critical or may it suffer downtime during business hours?
  - Is there a high turnover of mail during work hours?
- Is it an HUB server?
  - Is it a HUB server under load through replication?
  - Is it a HUB server hosting business critical applications?
  - Is the HUB server regarded as business critical or can it undergo downtime during business hours? In case of yes, does this cripple the entire replication schema or will the infrastructure still be able to replicate appropriately?
- Is it a Web server?
  - Is it used heavily by users during the entire day?
    - Is there a peak hour pattern?
    - Are there large or medium sized mail databases?
    - Are there large or medium sized application databases?
  - Is the mail server regarded as business critical or can it undergo downtime during business hours?
  - Is it a Web server under load by users? Are the users heavy users or light users working all day in applications hosted on the particular server?
  - Are there many write operations in the applications or does the users read more content than write it?

The above questions are pertinent to the selection of the right RAID level for your servers. It all comes down to:

- Do your servers need high fault tolerance?
- Do your servers need high performance?
- May high performance be implemented at the cost of high fault tolerance?

Also, the maintenance and costs of implementation have not been addressed in this section, but do play a role in the overall architecture of your RAID system.

How much money the company is willing to pay for the security that RAID implementations provide is again determined by the above three listed points. Their decision actually influences the price of the system.

6.6.18 RAID and disk controllers

Implementing servers with RAID technology allows data to be striped across a number of hard disks that are seen by the operating system as one large disk drive. The three most popular levels of RAID used are RAID 0, RAID 1, and RAID 5. RAID 0 generally gives the highest disk performance, but there is no fault tolerance should one of the individual drives fail. RAID 1 provides excellent fault tolerance and is essentially disk mirroring. RAID 5 technology uses parity blocks so that, should one drive fail, the controller can rebuild the information on a replacement hard drive, while still providing valid data to the system. RAID 5 is more efficient in its use of disk space, but RAID 1 has 50% more throughput than RAID 5 when using an equivalent number of drives if the percentage of writes is high. The disk subsystem is the most overlooked performance bottleneck in the server.

We recommend the following for your Domino server:

- Implement RAID 1 on all of your servers if possible, or RAID 5 as your second choice. As your Domino servers become more critical to your business, you will want the best combination of disk fault tolerance and performance you can afford.
- Implement a second RAID 1 logical drive to store the Domino transaction logs.
- Only use hardware-controlled RAID arrays. Software-controlled RAID arrays place an additional strain on your server in terms of CPU utilization. The CPU should be dedicated to providing Domino functions, not implementing a RAID subsystem.
- If possible, install an additional hard disk for use as a hot spare. Hot spares can replace failed disks automatically, or be managed by the administrator.
- Choose the fastest hard drives. High-performance disks provide fast data rates and low latency times. xSeries® servers support very fast 15,000 RPM drives.
- Do not mix SCSI Fast with SCSI Fast/Wide drives.
- Do not mix drives that spin at different speeds.
- Do not mix drives of different sizes in a RAID array.
- Enable write-back cache on the RAID controller. This setting improves system performance by caching information before it is written back to disk. Some controllers do not have battery backups, and cached data will be lost in the event of a power failure. In this situation, installing an uninterruptible power supply (UPS) for your server is particularly important.
- Set the RAID stripe size to 16 K. when dealing with Domino servers.

A

Important notes.ini parameters

This appendix lists, in functional groups, some of the most important parameters in the notes.ini file.

Steps for changing the notes.ini file

There are multiple ways that you can edit the notes.ini file. We outline the three most common ways here:

- Add lines to the notes.ini section of a configuration document in the Domino Directory (names.nsf, previously called Public Address Book). If you create a default configuration document, you can propagate these changes throughout your domain simply by replicating the Domino Directory.
- Directly edit the notes.ini file using an appropriate text editor, such as Notepad in Windows or vi in UNIX. When using this method, be sure to end the new line with a carriage return.
- Use the **`set config`** command at the Domino server console (or remote console). For example, **`set config debug_threadid=1`** will place `debug_thread=1` into the notes.ini. This method will dynamically enable or disable (that is, **`set config debug_thread=0`**) any settings that can be enabled on the fly, and is therefore the preferred method for most notes.ini settings.

Tips on editing or adding entries in the notes.ini file:

You can place notes.ini variables anywhere in the notes.ini file. However, we recommend that you group them near the end of the file to make it easier to find your changes in the future. Here are some tips:

- Add them before the last entry in the notes.ini file to avoid potential problems with an incorrect final character in the file. If the final character is not a correct character, the server might not recognize the final entry in the file.
- It is always a good idea to put in a comment before entering a change to the notes.ini file so you have a good record of the change that was made, why, and when the update occurred.
- For the server to pick up most changes in the notes.ini file, the server needs to be stopped and restarted after the changes are made.

Description of notes.ini variables

Table A-1 Notes.ini parameters affecting performance

Domino system parameters	Definition
CleanupTimeout	Allows changes to the default time of five minutes allowed for CleanupScriptPath to execute. Valid range is from 30 seconds to 30 minutes.
DbDir_Refresh_Interval= <minutes>	Reduces database directory scan frequency. Domino periodically scans your data directory for new files or files that have been removed. By default, this scan happens every 15 minutes. If you have a lot of files in your data directory, or a complicated directory structure, traversing the directory can take a significant amount of CPU. You can reduce the overhead by increasing the refresh interval. Keep in mind, however, that it will take Domino longer to recognize new files. Default=15 (minutes). 0=Disabled (Never scan).
EnableJavaAgents	Disables Java Agents when = 0. Domino Agent Manager (AMgr) and HTTP server automatically support running Java agents. You can disable Java agents by adding the following line to the notes.ini file for the Domino server: `Default: 1 (ENABLED)`
Fixup_Tasks	The maximum number of Fixup tasks that are allowed to be started upon server startup to look at databases that need fixing. Default = 2 x the number of CPUs. Note: If fewer databases need Fixup than defined by FIXUP_TASKS, then only one Fixup task per database will be started.
Memory_Quota	Sets a limit on the total memory allocation available from the HEAP. Any value under 1000 is interpreted as MB, 1000 and higher is interpreted as bytes. Minumum is 4 MB.
NSF_Buffer_Pool_Size NSF_Buffer_Pool_Size_MB	The NSF_Buffer_Pool_Size (bytes) and NSF_Buffer_Pool_Size_MB specify the maximum size of the NSF buffer pool. See Chapter 5, “Understanding what to tune” on page 49 for more information.

NPN	When set to 1, this setting tells the Domino server setup program to initialize the Notes for Public Networks preferences. Adds the following to the notes.ini:
	`Billing added to the ServerTasks line`
	`BillingClass=Agent,Database,Document,Mail,Replication,Session`
	`BILLINGADDINOUTPUT=1`
	`MAILCLUSTERFAILOVER=1`
	Note: This setting is effectively obsolete.

NOTESPARTITION	Obsolete. This setting is no longer needed to set the Domino Server partition numbers.
NSF_DbCache_Maxentries= <#-of-DBs>	Sets the max number of databases that can be cached in the DBCache. Increasing the database cache size may well improve system performance, but may require additional memory and will requires Maxentries to be increased. The minimum is 25 and the maximum is 2000 (platform). Default: None, but if this setting is omitted, the number is either 25 or the NSF_Buffer_Pool_Size value is divided by 300 K (whichever is greater).

Domino system parameters	Definition
PercentAvailSysResources	Controls the amount of memory allocated to each Domino Server by percentage. This setting assigns a portion (%) of total system memory to each Domino server by specifying a value from 2% to 100%, which represents an absolute percentage of the system's total physical memory. This in turn limits NSF_Buffer_Pool_Size (not present in notes.ini) to 3/8 of the memory allocated by PercentAvailSysResources, leaving 5/8 for other Domino buffers to maintain proportion and balance.
Previous_TransLog_Path	Describes the path to transaction logs in the previous configuration. Allows the system to restart after manual changes to logfile status, location, or style or loss of log files. Note: Some operating systems require a trailing path separator. Use just for recovery and emergencies only. Default=DATA\LOGDIR.
Previous_TransLog_Status	Describes the status of transaction logging in the previous configuration. Allows the system to restart after manual changes to logfile status, location, or style or loss of log files. Use just for recovery and emergencies only. Default=0 (DISABLED).
Previous_TransLog_Style	Logging style in the previous configuration. Allows the system to restart after manual changes to logfile status, location, or style or loss of log files. Use for recovery and emergencies only. Default=0 (CIRCULAR).
Replicators	Specifies the number of Replicator tasks that can run concurrently on the server. A replicator task synchronizes a source database with its replicas. Starting more than one replicator allows a server to replicate its databases with more than one server simultaneously. If, however, you enable two replicators, and the server only needs to replicate with one other server, then only one replicator is used. The other task remains idle. Multiple replicators will speed up synchronization but may increase workload significantly on the server. Default=1. Recommendation: (number of processors -1). See Chapter 5, “Understanding what to tune” on page 49 for more information.
Server_Max_Concurrent_Trans	Regulates the number of concurrent transactions that can occur in a server at one time. When the number of concurrent transactions is reached, the other transactions are put into a wait state until one of the active transactions completes. The maximum value for this setting is 50. If you set this parameter, be aware that IOCP threads will default to two times this number. Default=20.
Server_MaxSessions	Defines the number of user sessions a Domino server can accommodate. When a new user attempts to log on, if the current number of sessions is greater than the value of Server_MaxSessions, Domino closes the least-recently-used session. In order for a session to be considered for closing, it must have been inactive for at least one minute. Default= Unlimited.

Domino system parameters	Definition
Server_MaxUsers	Specifies the maximum number of active NRPC user sessions allowed on a server. This is only effective for NRPC (Notes Client) users and does not affect HTTP users or sessions. When this number is reached, the server state becomes MaxUsers, and the server stops accepting new Database Open requests from NRPC users. Default= Unlimited.
Server_Pool_Tasks	Sets the number of IOCP thread pools, per port, running on the server. Enables a small pool of threads to do server work instead of one per user. Default: 2. This value is multiplied by the value in the SERVER_MAX_CONCURRENT_TRANS setting to determine the actual number of server threads on the server. We do not generally recommend using this setting.
Server_Max_Concurrent_Trans	Controls the number of physical threads that are allowed to concurrently execute transactions. This control was very important before IOCP thread pools were introduced in Domino 5. From Domino 5 onwards, this control still has use in controlling the number of concurrent transactions, which affects how virtual memory may be needed at any one time.
Server_Session_Timeout=<minutes>	Causes Domino to terminate an NRPC connection or session that has been idle for this amount of time. A session is considered idle if no activity detected. If it is set too low, it may cause the server to have to re-open database server sessions too often. The best setting for this parameter is determined by the server load, numbers of concurrent users on the server, and so on. Note that creating a new session is *very* CPU expensive; for example, re-using sessions is *much* more efficient, which is why we have the parameter. The term “Session” also includes server “internal” sessions, such as agents, replicator tasks, server to server and so on, so plan for it! Default: 4 hours (240). Recommendation: 30/45 minutes (but never less).
Server_Show_Performance	This parameter writes server performance events to the Domino server console every minute. The messages will appear similar to the following: `03:41:55 p.m. 41 Transactions/Minute, 5 Users` The console command SHOW PERFORMANCE is used to both enable (set to 1) and disable (set to 0) this parameter in the Notes.ini.
ServerTasks=<task1>,<task2>, <task3>	Tasks that will start automatically at server startup. These may include entries such as Replica, Router, Update, Stats, HTTP, LDAP, POP3, IMAP, clrepl, cldbdir, cladmin, and so on.
skip_fixup=1	Obsolete in Domino 7.
Translog_AutoFixup	Lets FIXUP –J run automatically if Domino cannot use the transaction logs to fully recover a database. Default: 1 (Enabled). Notes: ▶ Translog_* parameters are overwritten by the values from the server document, so they should not be changed directly in the notes.ini. ▶ When this parameter is disabled, Domino does not run the FIXUP -J task automatically. It just notifies the administrator to run the FIXUP task with the -J parameter on the corrupt databases, and those databases will not be accessible.

Domino system parameters	Definition
Translog_MaxSize= <size in Mb>	Defines the maximum size in Mb that Circular of Linear transaction log files can grow to before being forced to start over-writing (re-using) themselves. This parameter is ignored when using Archival transaction logging. Notes: ▸ The minimum size is 192 MB (it ignores all lower values!). The maximum size is 4 GB. ▸ The archive logs are allocated up front, so be sure enough space exists to generate them. ▸ While it is possible to use this with Linear logging, we recommend using it only with circular logging. ▸ Translog_* parameters are overwritten by the values from the server document, so they should not be changed directly in the notes.ini directly.
Translog_Path=<Full-Directory-Path>	Specifies the path to the transaction log. The default location is \logdir in the server's data directory. However, we strongly recommend storing the transaction log on a separate mirrored device, such as a RAID level 0 or 1 device with a dedicated controller. If you change this field and have an existing transaction log, you must use the operating system to move all the log files to the new log path. Note: Translog_* parameters are overwritten by the values from the server document, so they should not be changed directly in the notes.ini.
Translog_Performance	Specifies the trade-off between transactional log runtime and restart recovery time, as follows: 1 = Favor runtime. The system stores more database changes in memory and writes fewer changes to the transaction log. Fewer writes to disk improves the server runtime. 2 = Standard (default). 3 = Favor restart recovery time. The system stores fewer database changes in memory and writes more changes to the transaction log. More writes to the transaction log improves restart recovery time. Default: 2 (Standard). Note: Translog_* parameters are overwritten by the values from the server document, so they should not be changed directly in the notes.ini.
TransLog_ReCreate_LogCTRL	Used when the all the transaction logs (*.txn) exist, but the control file (logctrl.lfh) is missing. 1=ENABLED.
Translog_Status	Enable transaction logging for all Release 5 (or higher) databases. 0 - Transactional logging disabled. 1 - Transactional logging enabled. You must upgrade databases to the Domino Release 5 format before they can be transactionally logged. Note: Transaction Logging dramatically improves restart times after a Domino crash on all platforms, improves data integrity, and significantly reduces data loss after a crash. Default is 0 (DISABLED). Recommendation: ENABLED. Note: Translog_* parameters are overwritten by the values from the server document, so they should not be changed directly in the notes.ini.

Domino system parameters	Definition
Translog_Style	Type of transaction logging. 0 = Circular (default). The system continuously re-uses the extent log files, overwriting old transaction logs. 1 = Archive. To re-use the log files after they are archived. A log file can be reused when it is inactive, which means that it does not contain any transactions necessary for a restart recovery. Use a third-party backup utility to copy and archive the existing log. When Domino uses the existing file again to start, Domino increments the log file name. If all the log files become inactive and are not archived, Domino creates additional log files. 2= Linear. Re-use the log files and overwrite old transactions for log size greater than 4 GB. Note: Translog parameters are always overwritten by the values from the server document, so there is no point to changing them directly in the notes.ini.
Translog_UseAll	Allow transaction logging to use all the available space in the location defined by the “Translog_Path=<directory> parameter” for Circular or Linear Transaction log files. 0=DISABLED (Default). 1=ENABLED. Notes: ▶ Translog parameters are always overwritten by the values from the server document, so there is no point to changing them in the notes.ini directly. ▶ This parameter is ignored when using Archival transaction logging. ▶ Circular logging will still be held to a maximum of 4 GB.
View_Rebuild_Dir=<path to desired directory>	Specifies the directory where temporary files will be created for optimized view rebuilds. Recommend specifying the full path to designated directory. Default: None, although Domino will use the operating system’s designated Temp directory.

Table A-2 Logging parameters

Logging variable	Description
Log	Modifies logging behavior. Variable 1 = Log's file name. Variable 2 = Logging options to be added together. 1= Log to console. 2= Force database Fixup when opening log. 4= Full document scan (as opposed to quick scan or open). Variable 3 = Not currently used. Variable 4 = Number of days at which to delete log documents. Variable 5 = Size of log text in event documents. Variable 6 = Optional number of days to keep activity trends data documents.
Log_Replication	Indicates level of replication session logging. 0 - Do not log replication events. 1 - Log that a database is replicating. 2 - Log summary information about each database. 3 - Log information about each replicated document (both design and data documents). 4 - Log information about each replicated field.
Log_Sessions	Specifies whether individual sessions are recorded in the log file and displayed on the console. 0 - Do not log. 1 – Log the start and stop of individual sessions.

Logging variable	Description
Log_Update	Specifies the level of detail of Indexer events displayed at the server console and in the log file: 0 - Records when the Indexer starts and shuts down. 1 - Records when the Indexer starts and shuts down and when the Indexer updates views and full text indexes for specific databases. 2 - Records when the Indexer starts and shuts down and when the Indexer updates views and full text indexes for specific databases. Also records the names of views the Indexer is updating.
Log_View_Events	Enable logging of view rebuilds. 0 - Do not log messages when views are rebuilt. 1 - Log messages when views are rebuilt.
Mail_Log_To_MiscEvents	Determines whether *all* mail event messages are displayed in the Miscellaneous Events view of the log file: 0 - The router determines whether to log messages to the Mail Events view, the Miscellaneous Events view, or to both views. This is also true if this variable is not defined. 1 - The router ensures that all messages are logged to the Miscellaneous Events view. If a message is typically logged only to the Mail Events view, it would now be logged to both the Miscellaneous Events view and the Mail Events view. Note: To prevent logging to the Domino Console, the NOTES.INI variable MailLogToEventsOnly can be set to a value of 1. There is no NOTES.INI setting to prevent logging to the Miscellaneous Events view.
No_Force_Activity_Logging	Sets the database activity logging mode. Without this variable, setting the Statlog server task enables the Record Activity option for every database on the server and adds 64 KB to each database. 0 - Allows automatic activity logging on all databases. 1 - Prevents automatic activity logging on all databases. Note: Even when activity is not being recorded for the database, the information is still recorded in the Activity entry of the Database Usage view in the server's log file.
Passthru_LogLevel	Specifies the level of trace information recorded for all Notes RPC network connections (including pass-through) in the Miscellaneous Events view of the log file. 0 - No information is recorded. 1 - Only errors are recorded. 2 - Summary progress information is recorded. 3 - Detailed progress information is recorded. 4 - Full trace information is recorded. 5 - Full trace information plus driver messages are recorded.
PhoneLog	Settings to define which, if any, phone calls are written to the Notes Log. 0 = Do not log phone calls. 1 = Log all phone calls except fails due to “busy”. 2 = Log all phone calls.

Logging variable	**Description**
Platform_Statistics_Disabled	Disable the tracking of performance metrics of operating system and output results to server and statrep. Platform Statistics are enabled by default. Notes: ► The `show stat platform` command will display the logged statistics. The PLATFORM command controls this feature at the console using the following arguments: ► TIME <n>: Change sample rate n (minutes); default is one minute. ► RESET: Begin new stats session; reset metrics. ► WAIT: Pauses the collection of performance data. ► RUN: Resume the collection of performance data.
RTR_Logging	Enables or disables monitoring of Cluster Replicator activity. 0 - Disables monitoring of the Cluster Replicator. 1 - Enables default monitoring of the Cluster Replicator. **Note:** For additional values, and more on how to use this setting, see *Lotus Software Knowledge Base Document #1214739*, found at: `http://www.ibm.com/support/docview.wss?rs=899&uid=swg21214739`
Show_Task_Detail	Enables additional information about the Show Task command.
Client_Clock	Notes Client setting to enable logging of Notes RPC traffic. 0-Disabled. 1- Enabled. Notes: ► This setting can negatively impact client performance, so it should be used only for troubleshooting purposes. ► Use in conjunction with *Debug_Console* and *Console_Log_Enabled* (below)
Debug_Console	For use with Client_Clock. This setting creates a DOS window to visually display the Notes Client's RPC traffic in real time. 0-Disabled. 1-Enabled. Notes: ► This setting can negatively impact client performance, so it should be used only for troubleshooting purposes. ► This setting will not generate a text log of the traffic. Use Console_Log_Enabled for this.

Logging variable	Description
Console_Log_Enabled	Enables logging of all console messages to a text file. 0- Disabled. 1- Enabled. This setting provides several improvements over its predecessor, Debug_Outfile, which is now effectively obsolete. Some of the most important improvements are: ► Consistent default file name and location on all platforms. console.log is located in the data\IBM_TECHNICAL_SUPPORT folder. ► Automatic renaming of previous console.log files. The previous console.log files are renamed using the following convention: `console_MachineName_YYYY_MM_DD@HH_MM_SS.log` More information about the Dynamic Console Logging feature can be found in *Lotus Software Knowledge Base Document #1085072*, found at: http://www.ibm.com/support/docview.wss?rs=899&uid=swg21085072
Console_Loglevel	Logging level for the Domino server console. 0 - No information displayed. 1 - Only errors are displayed. 2 - Summary progress information is displayed. 3 - Detailed progress information is displayed. 4 - Full trace information is displayed.
Log_Agentmanager Debug_Amgr	0 - Do not show logging 1 - To show partial and complete successes. 2 - To show complete successes. The Log_AgentManager setting provides you with a subset of the debugging information that Debug_AMgr generates. This option provides less output, but it has a smaller impact on performance. Some people keep the Log_AgentManager setting turned on even when there are no problems, just to have additional information in the log. If you have both notes.ini variables specified, Debug_AMgr settings will take precedence. *Lotus Software Knowledge Base Document #1115204* provides further details on how to implement Debug_Amgr at: http://www.ibm.com/support/docview.wss?rs=899&uid=swg21115204
Log_Dircat	Controls which information related to the Directory Cataloger task is logged to the console and to the Miscellaneous Events view of the log file (LOG.NSF): 1 - Logs when the Directory Cataloger starts and finishes, the name and domain of each source Domino Directory as it is aggregated, and the number of entries processed. 2- Logs the above, plus the number of entries before and after aggregation. 3 - Logs the information from 1 and 2, and adds the names of all entries processed. Using 3 is not recommended, because it slows performance and fills the log file. If you do use 3, use it only temporarily.
Log_Disable_Session_Info	Allows administrators to turn off the writing of session events to the log. 0 - Disabled. 1 - Enabled.

Logging variable	Description
Log_MailRouting	Degree of logging for the router task. 0 - Note: Defaults to 20. 10 - Shows errors, warning, and major events. 20 - As 10, but successful deliveries and transfers are also logged. 30 - Add thread information. 40 - Add transfer messages, message queues, and full document information for mail.boxes.

Table A-3 Agent Manager and the Adminp parameters that affect performance

Agent Manager variable	Function
AdminPInterval	Interval cycle (in minutes) for when the Administration Process (ADMINP) carries out many types of requests. This setting corresponds to the interval setting in the Administration Process section of a Server document. The Administration Process looks in the Server document first, and if it does not find a value in the Interval field, it refers to the notes.ini setting. Default if setting not defined: 60 (minutes)
AMgr_DocUpdateAgentMinInterval	Specifies the minimum elapsed time (in minutes) between execution of the same document update-triggered agent. A delay between the execution of the same event-triggered agent (triggered by a document update or by new mail) exists so the user has control over the interval between the given agents. If the agent is triggered and its corresponding minimum interval has not passed, the Agent Manager schedules the task to run at its last run time plus the minimum interval. Default if setting not defined: 30 minutes
AMgr_DocUpdateEventDelay	Specify a delay of execution (in minutes) that the Agent Manager includes in the schedule for a document update-triggered agent after a document update event. This delay between when an event occurs (a document is updated or new mail is delivered) and the time at which the agent is scheduled to be executed, exists so that any rapid series of events will only result in one execution of a given agent. Then the Agent Manager as well as the overall server can perform more efficiently. Default if setting not defined: 5 (minutes)
AMgr_NewMailAgentMinInterval	Specifies the minimum elapsed time (in minutes) between execution of the same new mail-triggered agent. A delay between the execution of the same event-triggered agent (triggered by a document update or by new mail) exists so the user has control over the interval between the given agents. If the agent is triggered and its corresponding minimum interval has not passed, the Agent Manager schedules the task to run at its last run time plus the minimum interval. Default if setting not defined: none
AMgr_NewMailEventDelay	Specifies a delay of execution (in minutes) that the Agent Manager includes in the schedule for a new mail-triggered agent after a new mail message is delivered. This delay between when an event occurs (a document is updated or new mail is delivered) and the time at which the agent is scheduled to be executed, exists so that any rapid series of events will only result in one execution of a given agent. Then the Agent Manager and the overall server can perform more efficiently. Default if setting not defined: 1 (minute)
AgentManagerSchedulingInterval	Controls how quickly an agent gets into the schedule queue by specifying a delay (in minutes) between the running of the Agent Manager's scheduler. Default if setting not defined: 1 (minute) Valid values: 1 minute to 60 minutes Notes: ► This setting is *not* recommended for general use, as it may create significant delays in the running of scheduled agents. ► The Domino Administrator 7 Help incorrectly refers to this setting as AMgr_SchedulingInterval.

Agent Manager variable	Function
AMgr_UntriggeredMailInterval	Specifies a delay (in minutes) between running of the Agent Manager's check for untriggered mail. Incoming Mail (or indeed any notes document) is classed as "untriggered" when it is added to a database through replication, which does not cause a “new mail” trigger to fire. This parameter helps guarantee that the replicated mail will be processed by an agent. **Default:** 60 (minutes) Valid values: 1 minute to 1440 minutes (the number of minutes in a day) The Agent Manager automatically uses the default values for this variable. To override the default values, you can add it to the server's notes.ini file. In general, this rarely require tuning.
DominoAsynchronizeAgents	Specifies whether agents on a Web server triggered by browser clients can run at the same time. Options are: 0 - Only run one agent at a time (serially). 1 - Run more than one at a time (asynchronously). Applies to: Web Browsers

Table A-4 Indexing and full text (FT) indexing parameters

Parameter	Definition
Default_Index_Lifetime_Days	Allows administrators to set a default lifetime for indexes associated with database views if none was selected by the database designer in the Design View Attributes dialog box ("Discard index on server after n days of inactivity"). Default: 45 days.
DominoNoAnalyzeFormulas	Web pages containing @functions may be cached through the Formula Analyzer, which “analyzes” the formulas. The Formula Analyzer is enabled by default. This setting disables the Formula Analyzer, and therefore the Domino Web Cache.
FT_Alternate_Filter	Alternate filter added for attachment indexing, which enhances the capabilities of the Keyview filter. This alternate filter provides support for file types, such as the Ichitaro file format, which is important in international regions. For example: `FT_ALTERNATE_FILTER=naltfltr.dll` Note: The alternate filter must exist.
FT_Use_AltFltr	Enables the indexing of file attachments in the Ichitaro format. Full Text Search supports attachments in the Ichitaro file format. Default: 0 (DISABLED).
FT_Domain_Directory_Name	Select location and name of Domain Index. Default is FTDOMAIN.DI. in Data Directory. By adding this setting, Domino will support directory links and Index relocation.
FT_DOMAIN_IDXTHDS	Specifies the number of indexing threads to use for Domain Search. Using more threads lets the Domain Catalog server index more files simultaneously, but requires more CPU utilization, and response to search queries may be slow. With fewer indexing threads, the search speeds up because of greater CPU availability, but changes are not reflected in the index as quickly. Default: None, although if this setting is omitted, the default number of threads used is two per CPU.
FT_HTML_Title	By default, when Domino indexes HTML files for Domain Search, only the text in the <BODY> field is indexed. Domino does not index text included in any other fields. To include all HTML fields in a Domain index, for example, <META> or <TITLE> fields; edit the FORMATS.INI file. Find the 210=htm statement and remark out the statement `REM 210=htm`.
FT_Index_Accent_Sens	Enables accent sensitive full text searching for special language characters. Default=0 (DISABLED).
FT_Index_Attachments	Specifies whether to exclude types of document attachments in full text indexes that are not already excluded by default. A value of 1 includes these document attachments in the index, and a value of 2 excludes them. The following types of attachments are excluded from the Domain Index by default: .au, .cca, .dbd, .dll, .exe, .gif, .img, .jpg, .mp3, .mpg, .mov, .nsf, .ntf, .p7m, .p7s, .pag, .sys, .tar, .tif, .wav, .wpl, and .zip.
FT_Index_Ignore_Attachment_Types	Provides the ability to specify attachment types that are not to be included in full text indexes. Unlike FT_Index_Attachments, this setting provides the ability to exclude specific file types, while allowing others.

Parameter	Definition
FT_Intl_Setting	Imposes several limitations on full text functionality to let Notes work properly with the Japanese language. When enabled (set to 1), this setting turns off stemming, makes all full text indexes case-sensitive.
FT_KV_LOG	Logs full text indexing messages related to Keyview filtering. 0 - Disabled. 1 - Enabled.
FT_LIBName	Allows an alternate full text indexing engine to be defined. The syntax of the command is: `FT_LIBName=ftgtrxx` Default: None, but Domino 7 uses the GTR40 engine. Notes: ▶ At the time of this writing, there are no alternate full text engines available for Domino 7. ▶ Only full text engines of at least GTR40 are allowed. ▶ If you activate the alternate search engine, using this .ini parameter, Domino will re-build the indices for all databases that are currently indexed.
FTG_KEY_MAX= <max # of words>	Governs the maximum unique keys found by a wildcard search. Default: 15000.
FT_Max_Search_Results	Increases the default return limit for FT searches. Example: FT_MAX_SEARCH_RESULTS=10000. Default: 5000.
FT_No_CompWinTitle	Setting the variable to 1 will disable computing of the window title. This variable was introduced because computing a window title can be expensive in terms of CPU resources. By default, Domain Indexing will Compute Window titles. Many Notes documents do not populate a Title, Subject, or Topic field. Consequently, the Domain Indexer assigns the title "Document has no title" to these documents. Even if set, it is still possible to get a “Document has no title” warning. Default: 0 (Window Titles will be computed).
FT_Summ_Default_Language	Sets the Domain Search Summarizer language. Whenever the Summarizer assigns a language to a document, and is then unable to summarize in that language, it will then try to use the default language instead. Whenever the Summarizer cannot summarize a document because it is too short, or it cannot summarize in the language it is trying to summarize in, no error message and no summary will be produced. Values: bokmal, danish, dutch, english, finnish, french, german, italian, nynorsk, portugue (use this value in lieue of portuguese), spanish, swedish. Default: english (if null string) or, if not present, use Native language from locale.
FTG_Index_Limit	Specifies the amount of memory available for full text indexes. You can increase this index size limit by setting the value of X calculated as follows: 8 MB is calculated as 8*1024 *1024=8388608; thus FTG_INDEX_LIMIT=8388608 12 MB is calculated as 12*1024*1024 = 12582912; thus FTG_INDEX_LIMIT=12582912 Default: 6 MB Note: There is no upper limit to the size.

Parameter	**Definition**
FTG_No_Summary	Specifies whether document summaries can be displayed in search results. Domain Search can, under certain circumstances, return a result to a user who cannot access the result document. If summarization is also enabled, this problem is exacerbated, because the result of a search will, if detailed results are specified, now include several key sentences from the document. If you have such a situation, you may want to disable the Summarizer by setting this variable to 1. Default: 0 (Summarizer is enabled).
FullTextMultiProcess	Allows chronos to process full text updates directly, instead of having to add a full text index request to the update queue. Default: 0.
QueryMaxResults	Defines the maximum number of documents that the query engine will return. The query engine cannot handle a query that results in more than 5000 matches in 800 documents, and will return the error "Query is not understandable".
Update_No_Fulltext	Disables the full text indexer when set to 1. Default: If omitted, full text indexing is ENABLED.
Update_Suppression_Time	Defines the delay time in minutes between full text index and view updates, even if immediate indexing is set. The server builds a queue of changed documents for the indexer and the agent manager, which wake up periodically and process from the queue. This *will* increase your CPU load significantly. Default: 5 minutes.
Update_Suppression_Limit	Overrides the Update_Suppression_Time variable if <value> number of duplicate requests are received to update indexes and views. The server builds a queue of changed documents for the indexer and the agent manager, which wake up periodically and process from the queue. This *will* increase your CPU load significantly. Default: <not applicable, uses suppress time>.
Updaters	This setting controls the number of Update tasks started when the Domino server is started. Set this value according to your organization's requirements.

Table A-5 The mail related notes.ini parameters that most affect performance

Parameter	**Definition**
MailClusterFailover	Previously used to enable mail failover in a clustered environment. While it still functions, this is now configured in the Server Document.
MailCompactDisabled	Enables or disables the routine compacting of the server's mail.box. Without this setting in the notes.ini file, mail.box is compacted routinely when the Compact server task runs. 0 - Enables compacting of mail.box. 1 - Disables compacting of mail.box. Default: Not applicable.
MailDisablePriority	Used to disable high priory mail delivery (when requested by the sender) and treats all mail as though it were normal priority mail. 0 - Allow High Priority Mail (Default). 1- Treat all mail as normal priority.
MailLowPriorityTime	Routes low-priority mail during the specified time. Without this setting in the notes.ini file, the Notes server routes low-priority mail only between 12 a.m. (midnight) and 6 a.m. Set the time range using the 24 hour format. You must specify a time range for this setting; Notes will not deliver low-priority mail if you specify only one specific time, such as 4 p.m. For example: `MailLowPriorityTime=21:00-23:00` Specifies that low-priority mail is routed from 9 p.m. to 11 p.m. Default: 00:00-06:00.
MinNewMailPoll	This setting determines how often workstations can contact the server to see if new mail has arrived for the user. This setting overrides the user's selection in the Mail Setup dialog box. You can increase the mail polling interval if there are a large number of mail users on your server and you want to prevent frequent polling from affecting server performance.
NoMsgCache	This setting disables the per-user message caching by the IMAP task. This can improve capacity (number of users) on a server by reducing memory consumption. However, response time for some user operations may be slower.
SecureMail	Setting this variable to 1 causes the Notes mailer to sign and encrypt all mail sent from the workstation and removes the Sign and Encrypt options from all dialog boxes. Note: This setting should *not* be used on a server.

Table A-6 Clustering parameters

Paremeter	Definition
Server_TransInfo_Max	The number of collection periods used to calculate the expansion factor for determining Server Availibility. The availability of cluster servers and the process of load-based failover is based on the comparison of the Server_Availability_Index and Server_Availability_Threshold. The Server_Availability_Index is calculated by the following formula: 100-(transactions/Server_Transinfo_Normalize) Transactions are counted within the statistic interval Server_Transinfo_Update_Interval. The number of intervals that are used to average the transaction counts may be set by Server_TransInfo_Max. Default: 5
Server_TransInfo_Update_Interval	Transactions counted within the statistic interval. The availability of cluster servers and the process of load-based failover is based on the comparison of the Server_Availability_Index and Server_Availability_Threshold. The Server_Availability_Index is calculated by the following formula: 100-(transactions/Server_Transinfo_Normalize) Transactions are counted within the statistic interval Server_Transinfo_Update_Interval. The number of intervals that are used to average the transaction counts may be set by Server_TransInfo_Max. Default: 15 seconds
Server_Cluster_Probe_Timeout	The interval in which the cluster servers probe each other for availability. Can cause serious performance issues if set too fast depending on the server's load. Default: 1 (minute) Valid up to 120 (minutes)
CLREPL_Poll_Interval	Cluster Polling Interval (Seconds). Default: 15 seconds
Cluster_Replicators	Parameter to define the number of Cluster Replicators. Default: 1 Note: Excessive or unnecessary cluster replication will seriously impact performance.
Disable_Cluster_Replicator	This setting disables cluster replication when set to 1.
RTR_INITIAL_RETRY_INTERVAL	Cluster initial retry. The duration between attempts to contact other cluster mates. Default: 60 (seconds)
RTR_MAX_RETRY_INTERVAL	Maximum interval allowed to pass between cluster replication attempts. Default: 900 (seconds)
Server_Availability_Threshold=<%>	Works directly with the Domino statistic Server.Availability. When Server.Availability reaches the threshold set in the Server_Availability_Threshold parameter, the server begins rejecting user requests. The threshold may need to be set high (95 to 97) for the best failover characteristics. In addition, Server_Transinfo_Normalize may need to be set. This allows you to "normalize" the response times experienced by your Domino server and insures that the failover processing will occur only when it is supposed to. This value can be tailored for your environment. Tip: Manually set to a carefully calculated value.

Paremeter	Definition
Server_transinfo_Normalize	Used when calculating the server availability index to "normalize" the response times observed at the server (that is, it divides the observed response times by this normalize value). Until now, this setting was undocumented, but it is available in both R4.6 and R5. For the availability index calculation to work properly, the normalize value should be roughly equal to the average Domino transaction time (for the server in question) in milliseconds*100. The default value is 3000 ms, corresponding to an average response time of 30 ms per transaction. Note: This default setting was appropriate for "the average server" when clustering was first shipped several years ago, but it is too large for the current generation of servers. You should use a lower normalize value with today's faster servers, so loads failover correctly. Tip: Manually set to a sensible value.
Server_Cluster_Default_Port	This directs cluster communications between the servers through TCP/IP. Even if your network uses multiple protocols, you should configure all servers in a single cluster to support the same set of network protocols. This ensures that clients can connect to any server in the cluster in the event of failover or for workload balancing. You should specify the TCP/IP port of the server on the Server_Cluster_Default_Port setting in the notes.ini file of all servers that belong to a cluster.
Server_Cluster_On=<0 or 1>	Enables clustering when set to 1.

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this Redpaper.

IBM Redbooks

For information about ordering these publications, see "How to get IBM Redbooks" on page 143. Note that some of the documents referenced here may be available in softcopy only.

- *Domino 7 Server Consolidation: Best Practices to Get the Most Out of Your Domino Infrastructure*, REDP-4181
- *Domino for IBM eServer xSeries and BladeCenter Sizing and Performance Tuning*, REDP-3851
- *Lotus Domino for S/390 Release 5: Performance Tuning and Capacity Planning*, SG24-5149

Online resources

These Web sites and URLs are also relevant as further information sources:

- Lotus Notes and Domino white papers

  `http://www-10.lotus.com/ldd/notesua.nsf/White%20Papers?OpenView`
- Lotus Documentation and Release Notes

  `http://www-10.lotus.com/ldd/notesua.nsf/RN?OpenView`
- Domino Performance

  `http://www-128.ibm.com/developerworks/lotus/performance/`

How to get IBM Redbooks

You can search for, view, or download Redbooks, Redpapers, Hints and Tips, draft publications and Additional materials, as well as order hardcopy Redbooks or CD-ROMs, at this Web site:

ibm.com/redbooks

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

Domino 7 Performance Tuning

Best Practices to Get the Most Out of Your Domino Infrastructure

Redpaper

Understanding the approach to performance tuning

Analyzing Domino 7 performance and related factors

Determining what to tune and why

Performance tuning, or optimization, is the process of modifying a system's configuration in order to improve its efficiency at meeting specified functional goals. The systems in question can be a single application, one Domino server, a collection of servers participating in a common activity, such as mail routing or replication, or an entire Domino infrastructure.

Performance tuning is goal-focused, and involves balancing the use of available system resources in order to meet those goals. This usually involves some sort of trade off, where one or more resources are optimized at the expense of others. A classic example of this is caching, which attempts to improve I/O performance by buffering data to memory. The memory allocated to the cache service is unavailable for use by any other programs on the platform, but if overall system performance is improved, the benefits outweigh the cost.

This IBM Redpaper discusses a best practices approach to Performance Tuning in Domino 7. It addresses both how to approach the science of performance tuning in a structured, logical manner, while also providing an in-depth discussion of specific configuration parameters to tune in specific situations.

After reading this Redpaper, this should further assist Domino Administrators in tuning Domino systems to more effectively:

- Improve user or application response time
- Improve system stability
- Reduce transaction times
- Balance the use of system resources