FAST INSTREAM

version 4.1

DEPLOYMENT GUIDE

Document Number: INS1005, Document Revision: A, March 23, 2005

Copyright

Trademarks

Restricted Rights Legend

Contents

FAST Support

Website

Please visit us at:

`http://www.fastsearch.com/`

Contacting FAST

Fast Search & Transfer, Inc.
Cutler Lake Corporate Center
117 Kendrick Street, Suite 100
Needham, MA 02492 USA
Tel: +1 (781) 304-2400 (8:30am - 5:30pm EST)
Fax: +1 (781) 304-2410

Technical Support and Licensing Procedures

Technical Support for customers with active FAST Maintenance and Support agreements,

E-mail: `tech-support@fastsearch.com`

For obtaining FAST licenses or software, contact your FAST Account Manager or

E-mail: `customerservice@fastsearch.com`

For evaluations, contact your FAST Sales Representative or FAST Sales Engineer.

Product Training

E-mail: `fastuniversity@fastsearch.com`

Sales

E-mail: `sales@fastsearch.com`

About this Guide

Purpose of this Guide

This guide describes what to take into consideration when planning how to deploy and implement FAST InStream to meet all system requirements in a cost-efficient manner.

FAST InStream is a derivative of FAST Data Search for OEM deployments.

Note! For information about other guides included in the FAST InStream documentation set refer to Chapter 1 *The FAST InStream Documentation Set* in the *Product Overview Guide*.

Audience

This guide provides information for several types of users:

- *Managers* and *supervisors* who need to understand how FAST InStream functions in order to plan the system design, configuration, implementation, and operation.
- *System administrators* and *operators*, who need to understand the data flow and basic working in FAST InStream.

Conventions

This guide uses the following textual conventions:

- Terminal output, contents of plaintext ASCII files will be represented using the following format:

  ```
  Answer yes to place the node in the known_hosts file.
  ```

- Terminal input from operators will be in the same but bold format:

  ```
  chmod 755 $HOME
  ```

- Input of some logic meaning will be enclosed in <> brackets:

  `setup_<OS>.tar.gz`

  where <OS> represents a specific operating system that must be entered.

- URLs, directory paths, commands, and the names of files, tags, and fields in paragraphs appear in the following format:

  The default home directory is the *C:\DataSearch* directory.

- User Interface page/window texts, buttons, and lists appear in the following format:

  Click **Next** and the **License Agreement** screen is displayed.

- *$FASTSEARCH* (UNIX) or *%FASTSEARCH%* (Windows) refer to an environment variable set to the directory where FAST InStream is installed.

Chapter 1

Introduction

About this Chapter

This chapter introduces you to the main concepts used in this guide.

It includes:

- Overview
- Basic Concepts

Overview

The goal of this guide is to describe what you need to consider when planning and implementing a deployment of FAST InStream to meet all system requirements in a cost-efficient manner.

System requirements cover functional requirements regarding features, fault-tolerance, and performance requirements.

As FAST InStream is a scalable system, most parts of this guide are concerned with scaling issues.

Basic Concepts

You scale a FAST InStream system by basing it on multiple servers in a cluster to provide a specific system service. The following terms are used when discussing these scaling concepts:

- *Service*: A service offers some well-defined functionality through an interface. A service can be defined independently of any physical realization of the service. A service is provided by a cluster.

  Example: search service, indexing service

- *Cluster*: This is a functional unit that provides one or more services. Internally, FAST InStream consists of several interconnected clusters that provide the different services making up the system. A cluster consists of one or more servers.

  Example: Search Cluster, Indexing Cluster

- *Server*: This is an instance of a software component, and one or more servers make a cluster. A server runs on a single machine.

  Example: Search Server, Indexing Server

- *Machine*: This is a physical machine used for running servers. A machine can run several servers, also of the same type.

Note! This guide only covers planning and initial deployment of the system. Day-to-day operations and changes to the system are covered in the *Operations Guide*.
The guide focuses on the deployment of medium-scale systems, but small-scale systems are also briefly described. The deployment of large-scale systems is outside the scope of this guide. Please contact FAST Professional Services at FAST Support if you need assistance to set up a large-scale system.
The subsection *Small-, Medium-, and Large-Scale Systems* on page 15 explains the difference between small-, medium-, and large-scale systems.

Chapter 2

System Deployment Model

About this Chapter

This chapter provides a generic system deployment model that serves as a basis for the descriptions and considerations given further on in the guide.

It includes:

- Generic Deployment Model
- Scaling
- Fault-Tolerance
- Heterogeneous Systems

Generic Deployment Model

The deployment model presented here complements the more common component-focused model presented in Chapter 2 *FAST InStream at a Glance* in the *Product Overview Guide*.

Figure 2-1 depicts a model for system deployment. Each box represents a system server, and most servers are replicated in a cluster to provide the necessary system capacity and fault-tolerance. The arrows indicate the main flow of data through the system.

Servers represented by black boxes are not part of FAST InStream, but represent typical interfaces to the rest of the application.

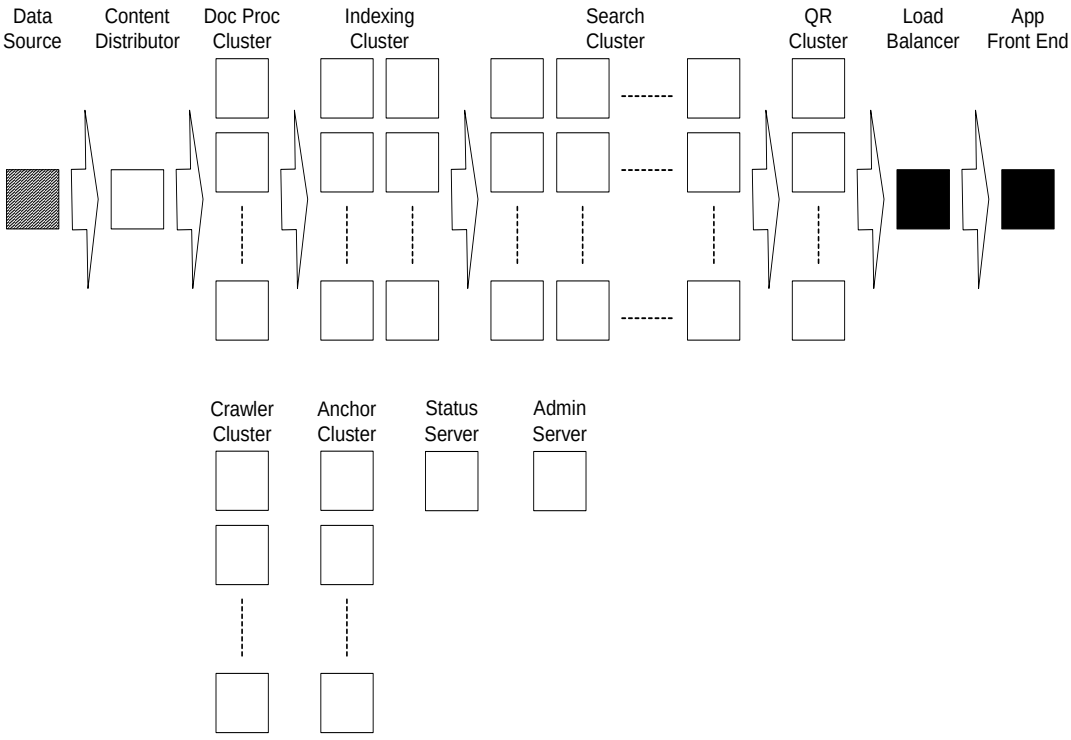

Figure 2-1 FAST InStream System Deployment Model

Note that the data source can take many different forms. This can either be a Crawler Cluster or one of the content connectors that can be used with FAST InStream. Furthermore, it can be a custom connector using the FAST Content API.

Note! Multi-node configuration for the Enterprise Crawler requires support from FAST Professional Services.

Servers are deployed across a set of machines. There are few restrictions in FAST InStream on how you can deploy servers. As a general rule, it is recommended not to run several different servers on a single machine unless you have verified that none of these servers represents a performance bottleneck in the system.

Scaling

Servers are replicated to provide the necessary system capacity. Replicated servers provide a single service and constitute a cluster for that service.

Scaling Axes

System scale is measured primarily along the following three main axes:

- Content volume:

  Content volume measures the amount of content to be handled by the system. Content characteristics play also an important role.

- Query rate:

  Query rate measures the number of queries per second that the system must be able to handle.

- Content dynamics:

  Content dynamics measure how the content changes and how these changes must be reflected in the system.

Search Server Matrix

The servers constituting the Search Cluster are arranged in a matrix to accommodate linear scaling both with document volume and query rate. Content is partitioned among the columns of this matrix. Every column holds different parts of the content, and the Search Cluster can scale with more content if you add an extra column of Search Servers. Similarly, incoming queries are partitioned among the rows of the matrix. Every row holds the same content, and the Search Cluster can scale with higher query rate by adding additional rows of Search Servers.

General Scaling Considerations

The following gives you some general rules of thumb on how scaling along the three axes content, volume, content dynamics, and query rate affects the different parts of FAST InStream.

- Scaling with content volume requires adding extra columns to the Search and Indexing Clusters. In general, extra servers must also be added in the data source Cluster if the data source keeps content state that grows with content volume. For instance, the Crawler keeps content state and must be scaled with content volume.
- Scaling with content dynamics requires adding extra columns to the Search and Indexing Clusters and extra servers to the data source Cluster and Document Processing Cluster.
- Scaling with query rate generally requires adding extra rows to the Search Cluster and extra Query and Result Servers.

Note! The number of Search Engine and Indexing Engine columns must be equal.

Fault-Tolerance

FAST InStream provides several options for fault-tolerance and can be configured to provide a suitable balance between fault-tolerance requirements and system cost.

Failure Models

Generally, there are three failure modules in FAST InStream:

- *Fail Safe*: A fail safe system continues to provide full functionality and meet all requirements even in the case of failure of system components.

- *Fail Soft*: A fail soft system continues to operate but with reduced functionality in case of failure of system components. Such systems are also often said to provide graceful degradation.
- *Fail Stop*: A fail stop system will not provide any functionality in case of failure of system components. Without fail stop fault-tolerance, the system would provide incorrect functionality in case of failures, providing for example wrong results from computations. This situation is often denoted Byzantine failures.

The default failure concept in FAST InStream is based on fail stop. This means that servers or a machine running one or more servers will shut down upon detecting an unrecoverable error or other failure. FAST InStream can however be configured to be fail safe or fail soft for any number of concurrent failures. Different FAST InStream functions can be configured with different fault-tolerance levels. For instance, the search function can be configured to be fail safe while the content submission function can be configured to be fail soft. This way, it is possible to balance system cost with fault-tolerance requirements.

Redundancy

Fault-tolerance is achieved by running redundant servers on redundant machines. Functionality in the FAST InStream servers ensures that the redundant servers replace failed servers whenever necessary. The number of redundant servers and machines determine the number of concurrent failures the system can handle. More redundancy generally increases the fault-tolerance, but also increases system cost.

Query and Result Cluster and Search Cluster

Redundancy in the Query and Result Cluster and Search Cluster is required for a fail safe search service and to always be able to handle the query load. Both components have built-in support for fail safe fault-tolerance.

Note! There must be an entire row of spare machines in the Search Cluster because each column holds unique data. A redundant machine in the Search Cluster can only replace a failed machine in the same column.

Alternatively, the search service can be deployed in various fail soft configurations. The Query and Result Cluster will degrade in terms of query load capacity unless there are redundant servers. The Search Cluster can be configured to either degrade in terms of query load capacity or content volume when no redundant servers are available.

Data Source Cluster

The content submission service cannot be made fully fail safe in the current version of FAST InStream. Several of the server types involved in content submission can operate in a fail safe manner to ensure high availability of the system, but as a whole the content submission system is fail soft.

There are many different types of data sources, and fault-tolerance support is different for each type. Many connectors are stateless, and failover instances can easily be added. A Crawler maintains an extremely large state, and good system for backing up or mirroring the crawler data is necessary to ensure uninterrupted fault-tolerance.

Indexing Cluster

It is recommended that you add redundancy to the Indexing Cluster to ensure fail safe operation of indexing. Indexing Servers keep state. It is therefore necessary to have at least two rows of Indexing Servers to be fail safe with one concurrent failure. You can add more rows can to handle more concurrent failures. If only a single row of Indexing Servers is used, the indexing service will be interrupted, and there is a risk that content must be resubmitted if the failure led to loss of data, for example, in case of a disk failure.

Load Balancer

As any complex network-based system, FAST InStream requires a load balancer. The load balancer plays an important role for fault-tolerance. In addition to balancing query load between the different Query and Result Servers, it must also detect availability of these servers and forward queries only the functioning servers. In this way the load balancer provides a single access point to the search service, hiding fault-tolerance and failover systems.

Note! There is no load balancer included in the FAST InStream distribution. You need to provide a third-party load balancer.

The load balancer must be located between the Query and Result Cluster and the application front-end. Normally, the front-end is implemented as a web application using server-side programming. For fail-over to work correctly if the load balancer is located only in front of the front-end, it is necessary for the front-end web server to be shut down if a failure is detected somewhere in the system. This is usually hard to achieve. If load balancing is needed in front of the front-end, the best approach is to set the system up with two levels of load balancing, possible implemented using a single load balancing device. This is illustrated in Figure 2-2:

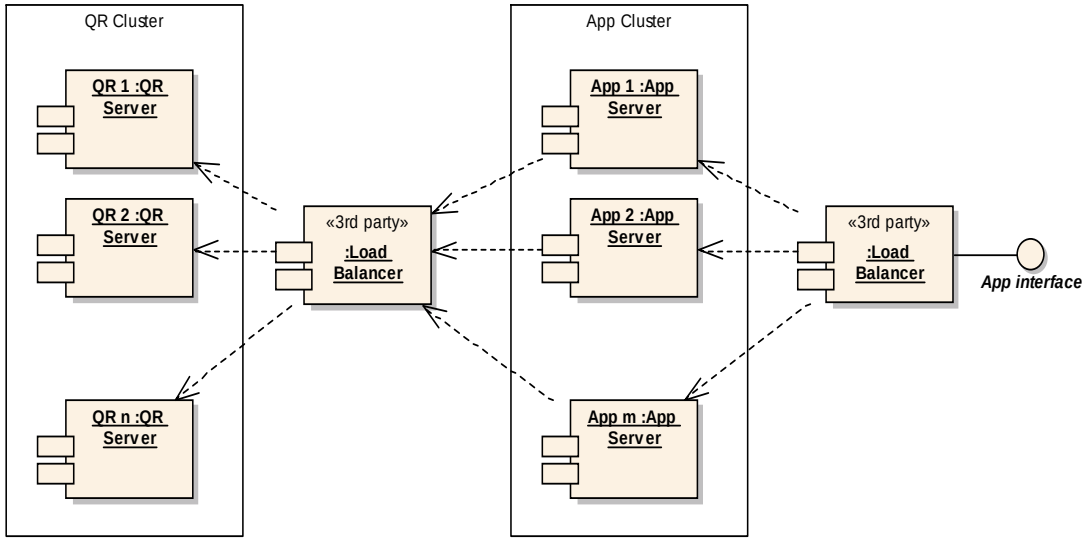

Figure 2-2 Two-level Load Balancing

It is recommended that you use a commercial, off-the-shelf load balancing device. These devices usually come with their own built-in fault-tolerance to avoid the load balancer itself becoming a single point of failure in the system. If in doubt, consult FAST Professional Services about whether a specific load balancer device is suitable for use with FAST InStream.

Heterogeneous Systems

This guide mainly considers deployment of homogenous applications. This means that all content and all queries will be handled similarly and according to the same set of requirements. This is not always the case in real applications. For example:

- An application may provide a unified search interface towards several sources of content, but content dynamics or other performance requirements may be different for each data source.
- An application similar to the first example may include fault-tolerance requirements that are different for each data source. All content of a specific type must always be queried, but for other content types it is acceptable that a subset of the content volume is not queried during a failure situation.
- An application may provide search in two (or more) different content types that require significantly different index profiles. A number of fields may be equal

between the two content types and facilitate queries against both types, while other fields are specific to the content type.

- A significant part of the queries targets a specific subset of the content.

In general, every part of a FAST InStream system is dimensioned to meet the maximum performance requirements that may occur in any part of the system. For heterogeneous systems however, this is not the most cost-efficient way of dimensioning the system. Instead, it is more cost-efficient to dimension the different parts of the FAST InStream system according to the different types of requirements. This usually involves deploying a heterogeneous system with parallel content processing pipelines with different performance for each pipeline.

In the first example above, it may be possible to deploy a system with two parallel Search Clusters, one for the high-dynamics content and one for the low-dynamics content. High content dynamics usually require more Indexing and Search Cluster columns with fewer documents per column for rapid indexing. Placing the low-dynamics content in a different Search Cluster with more documents per column reduces the total number of machines required for deploying the system.

For detailed information on how to deploy a heterogeneous system, contact FAST Professional Services.

Chapter 3

Deployment Planning

About this Chapter

This chapter gives you some indications on what to consider when planning your deployment.

It includes:

- Overview
- Functional and Performance Requirements
- Small-, Medium-, and Large-Scale Systems
- Dimensioning

Overview

System dimensioning is an important part of planning system deployment. The objective is to specify a deployment that meets all system requirements in a cost-efficient manner. System dimensioning involves the following steps:

1 Check the *Installation Guide* and the FAST InStream release notes to learn about any issues that might be relevant for the application, such as platform of choice.

2 Understand the requirements for the system. To facilitate cost-efficient dimensioning of the system, it is important to know the requirements that influence the system scaling.

3 Verify that the system is within the scope of this guide. This guide covers deployment of medium-scale systems with deployment of small-scale systems briefly addressed in Chapter 5 *Small-Scale Deployment*. Deployment of large-scale systems is not covered in this guide.

4 Evaluate factors that dominate system scaling, and adapt the system deployment model accordingly. This includes identifying heterogeneous systems as discussed in *Heterogeneous Systems* on page 11 or special options that can be enabled or disabled for this system.

5 Estimate system dimensioning based on performance requirements, deployment model, and estimates for the capacity of individual servers.

6 Optimize system bottlenecks by looking at individual system servers and change the system configuration accordingly. This may yield new estimates for the capacity of certain servers. Chapter 4 *Performance Optimization* addresses server optimization.

7 Verify system performance on a possibly scaled down version of the system.

8 Install full-scale version of the system and verify that all requirements are met. This test must cover functionality, performance and fault-tolerance requirements.

9 Establish and verify day-to-day operational procedures for the system. These procedures must cover hardware monitoring and repairs, performance monitoring, system failure monitoring, and system resource utilization.

It is often necessary to iterate steps 5 and 6 until further optimization does not result in significant improvements. Benchmarking on actual data, configuration and equipment is usually necessary both to get good estimates for the capacity of system components and to discover system bottlenecks. Initial iterations can usually be done on scaled down versions of the system.

Functional and Performance Requirements

For proper dimensioning of the system, it is necessary to have a good understanding of the system requirements that influence scaling most. Appendix B *System Requirements Checklist* contains a list of questions that are intended for aiding this process. Answering all or most of the questions provides a good basis for proceeding with system dimensioning.

Recall that system scale is measured along several axes, primarily content volume, content dynamics, and query rate. Questions 1 through 3 in the Appendix B *System Requirements Checklist* aim at these scaling parameters. Several features of FAST InStream influence performance significantly. Questions 4 and 5 aim at clarifying which of these features will be used. Question 6 addresses availability requirements, which typically dictate the level of equipment redundancy that will be needed.

Note! A properly working installation of FAST InStream requires that the basic system requirements such as the exact version of the operating system in use are met. For a detailed list of system requirements, refer to *Chapter 1 Installation Requirements and Concepts* in the *Installation Guide*.

Small-, Medium-, and Large-Scale Systems

Small-Scale Systems

In a small-scale system the machines have spare capacity. In a small-scale system, small changes in content volume, content dynamics or query rate will not influence the performance of the system. Fault-tolerance requirements and other non-performance requirements dictate the number of machines required for the installation. A system that satisfies all of the following requirements is generally considered to be a small-scale system:

- The system handles a maximum of 1,000,000 documents.
- The system handles a maximum of 10 queries per second.
- The system handles a maximum of 2 document updates per second.
- The system uses the default index profile.

Note! A larger system can be considered small-scale as long as each server is operating well within its capacity. Deployment of small-scale systems is covered in Chapter 5 *Small-Scale Deployment*.

Medium-Scale Systems

This guide focuses on deployment of medium-scale systems. Medium-scale systems are systems where the number of machines is dictated mostly by the performance scaling factors described in Chapter 2 *System Deployment Model*, section *Scaling*. The system scales linearly with any scaling factor, and changes to any one of content volume, content dynamics or query rate will influence the number of machines required for the system. For medium-scale systems there is usually a cost and complexity incentive to optimize deployment to reduce the number of machines required as much as possible.

Large-Scale Systems

In a large-scale system, network layout and data center deployment becomes increasingly important. There is no well-defined point at which a system changes from medium- to large-scale. Section *Network Performance Issues* on page 36 contains some information on network issues, and these issues are generally a sign of large-scale systems that normally comprise more than 100 machines. Contact FAST Professional Services for assistance with deploying large-scale systems.

Dimensioning

The processing capacity of the individual servers constituting FAST InStream depends heavily on the content being processed, the configuration, and the equipment used. Table 3-1 contains typical capacities for FAST InStream servers. These values can be used as initial estimates for system dimensioning. The values in Table 3-1 are based on a typical application for searching web content using the default FAST InStream configuration. All servers must be run on dedicated machines to achieve these performance figures.

With this information, it is possible to use the template provided in Appendix C *Dimensioning Template* to compute an initial deployment estimate for the system.

Note! The index expansion factor quoted for Search defines the system capacity indirectly. This performance attribute indicates the disk storage requirement for a Search Server in terms of the size of the input content.

Table 3-1 Typical Capacity of FAST InStream Servers

Server Type	Performance Attribute	Capacity
Query and Result Server	Queries per second	50
Search Engine Server	Total number of documents	4,000,000
	Document updates per second	5
	Queries per second	40
	Index expansion factor	2.5
Document Processor	Document updates per second	40
Content Distributor	Document updates per second	40
Crawler	Document retrievals per second	5
	Total number of documents	1,500,000

Chapter 4

Performance Optimization

About this Chapter

This chapter provides information on how to optimize the performance of the different components constituting an installation of FAST InStream.

It includes:

- Overview
- Query Load Performance
- Content Dynamics Performance
- Content Volume Performance

Overview

This chapter provides information on how to optimize the performance of the different components constituting an installation of FAST InStream. After initial dimensioning as described in Chapter 3 *Deployment Planning*, these guidelines can be used to locate performance bottlenecks and improve the total system performance in a cost efficient manner.

Query Load Performance

In FAST InStream query load performance is measured as the maximum number of queries per second (QPS) that the system is able to process with acceptable response times. If the query load is too high for the system, query response times will start to increase and the system may even return error messages.

Problems with query load performance are caused by a bottleneck in the Search Cluster, the Query and Result Cluster, or the application front-end. Chapter 7 *Benchmarking* describes how FAST InStream can be benchmarked to identify whether the problem occurs in the FAST InStream deployment or in the application front-end.

Query and Result Cluster Performance

Diagnosis

Performance problems in the Query and Result Cluster are relatively easy to diagnose. The Query and Result Server is a CPU-intensive component. If monitoring of the machines in the Query and Result Cluster shows that the CPUs are fully utilized most of the time, then the Query and Result Cluster is probably the bottleneck limiting query load performance.

In certain configurations and for certain features, the Query and Result Server may also use a lot of memory. If monitoring indicates any paging activity for the Query and Result Server processes, this will severely limit its performance. If this is the case, the only solution is to make more memory available to the Query and Result Server either by stopping other processes consuming memory or adding more physical memory to the machines.

The Query and Result Server uses disk only for logging, and this is usually not a performance bottleneck. Disk performance can become a performance problem if the disk system is very slow or heavily used by other processes on the same machine.

Solution

If the Query and Result Cluster turns out to be the performance bottleneck, you may try the following:

- If the machines in the Query and Result Clusters are paging, then more memory must be made available to the Query and Result Server processes. Any paging will add significantly to the query response time.
- Add more or faster CPUs to the machines in the Query and Result Cluster. The Query and Result Server is a multi-threaded process and is able to fully utilize multiple CPUs in a machine.
- Add more machines and run more Query and Result Servers in the Query and Result Cluster.
- Some features are expensive in terms of CPU usage in the Query and Result Server. Consider disabling some of these features if they are not required by the application. Some of the most expensive features are clustering, dynamic drill-down, lemmatization by reduction, and spell checking. For advanced linguistics features, make sure to enable support only for the required languages where applicable. Refer to the *Configuration Guide* for details on configuring these features.

Search Cluster Performance

Diagnosis

Performance problems in the Search Cluster are difficult to diagnose because of the complexity of the Search Server. The Search Server uses both CPU and disk IO extensively. In most practical systems, disk performance turns out to be the problem when the Search Cluster is the bottleneck limiting query performance.

Disk performance problems can usually be verified by measuring the number of physical disk operations per second on the machines in the Search Cluster. Disk transfer rate is usually not the problem. Disk operations rate is usually measured in transactions or operations per second. The maximum rate of disk operations varies between disk systems and is usually in the range of 200-400 operations per second per physical disk. The true capacity can only be determined by benchmarking.

When monitoring disk utilization, pay special attention to periods when the Indexing Servers are active either building a new index when running on the same machine or when transferring an index from a separate indexing machine. Disk utilization is highest during index updates.

Search Servers also use extensive CPU capacity. Monitoring CPU utilization is usually much simpler than monitoring disk utilization. The Search Server is a multi-threaded process that can fully utilize multiple CPUs.

Make sure to also monitor Search Cluster machines for any paging activity. Any paging will increase the query response time significantly, and it will also increase the load on the disk system making that even more likely to become a performance bottleneck.

Solution

Performance bottlenecks in the Search Cluster can be eliminated by the following approaches:

- If monitoring indicates that the Search Cluster machines are paging, make more memory available to the Search Server processes. This can be done by either stopping other processes running on these machines or adding more physical memory.
- If disk performance is the bottleneck then consider moving Indexing Servers to a separate set of machines if they share hardware with the Search Servers.
- Use disk with high spindle speeds or add more physical disks to increase the maximum number of disk operations per second for the machines. Disks must be tied together to make a single logical unit using RAID 0 or similar. Eventually, the disk IO busses will become saturated and must be scaled up as well.
- The number of disk operations on a Search Server is proportional both to the number of documents handled by the server and by the number of hits returned in response to queries. Reducing the number of documents on each Search Server will thus reduce the load on the disk system. This usually means adding more columns to the Search Cluster. The cost of additional systems versus the cost of improving the performance of a single system will determine the optimal balance between number of columns and documents per Search Server.
- Add additional rows of Search Servers to the Search Cluster. Queries are distributed among rows and this will reduce the load on individual servers.
- Some FAST InStream features are expensive in terms of Search Server performance. Consider disabling these features if they are not required. Some of the most expensive features are dynamic drill-down, full sorting, clustering, and dynamic duplicate removal.

Other Potential Performance Bottlenecks

Network capacity between the Search Cluster and the Query and Result Cluster may become a query load bottleneck. The risk of network capacity becoming a problem is mainly a problem in systems with many columns in the Search Cluster. Monitor network

utilization and especially look for packet losses in any switching equipment used and the network interface on the Query and Result Cluster machines.

If network capacity turns out to be the performance bottleneck, try to move other traffic away from the relevant network segments or upgrade to faster networks. If this does not help, then this system falls under the definition of a large-scale FAST InStream installation, which is outside the scope of this guide. Contact FAST Professional Services for assistance with configuring such a system.

Content Dynamics Performance

Content dynamics performance is measured as the rate at which documents can be submitted to the system and indexed to become searchable constrained by a maximum processing latency. Poor content dynamics performance is caused either by a bottleneck in document processing or indexing.

Indexing Cluster Performance

Diagnosis

It is usually simple to verify whether the Indexing Cluster is a bottleneck with respect to content dynamics performance. In the FAST Administrator Interface, the number of non-indexed documents and index freshness is shown for individual Search Servers under the **Matching Engines** tab. If there is a large number of non-indexed documents and freshness is high, then the Indexing Cluster is a bottleneck.

Indexing Servers use a lot of disk IO capacity and a lot of CPU. It is necessary to monitor machines in the Indexing Cluster with respect to both to determine what the limiting factor is. For indexing, both the rate of disk operations and the transfer rate must be monitored as both may limit performance. If machine monitoring reveals paging activity in the Indexing Server processes, this will also limit performance significantly.

Solution

The performance of the Indexing Cluster can be improved by any of the following means:

- If the machines in the Indexing Cluster are paging, then make more memory available to the Indexing Servers either by stopping other processes or by adding more physical memory to the machines.
- If disk performance is a problem and the Indexing Cluster is sharing machines with the Search Cluster, then consider moving the Indexing Cluster to a separate set of dedicated indexing machines.

- Improve the performance of the storage system by using faster disks or add more physical disks. Several physical disks must be configured into a single logical disk using RAID 0 or similar. Note that Indexing Servers do a lot of write operations to the disks and some RAID configurations show poor performance for disk writes.
- Reduce the number of documents indexed by each machine in the Indexing Cluster. This will generally increase the number of columns in the Indexing and Search Clusters.

Note! Note that systems using the partial updates feature of the FAST Content API will load the Indexing Cluster significantly more than systems not using this feature. Consider alternatives to this feature if possible. Note that the Anchor Cluster uses this feature if enabled in the document-processing pipeline.

Document Processing Performance

Diagnosis

A good indication that the Document Processing Cluster is the bottleneck with respect to content updates is that requests to the FAST Content API take a long time to return. Content submission through this API will not return until there is a Document Processor Server available to handle the submitted content, and long response times indicate that all Document Processor Servers are utilized.

The performance bottleneck may not be located in the Document Processor Cluster itself. Document processing also relies on other servers that may be performance bottlenecks.

Monitor CPU utilization and paging activity for the machines in the Document Processing Cluster. This is the system bottleneck if CPU utilization is high or the system is paging. Note that each Document Processor Server is running as a single thread. Several Document Processing Servers can be run on a single machine to fully utilize multiple CPUs. Good CPU utilization is usually achieved by running two Document Processor Servers per available CPU assuming that no other CPU-intensive processes are running on the machine.

The Status Server may also be a bottleneck for content updates. The best way to check whether the Status Server is the bottleneck is to inspect the statistics for the individual Document Processor Server stages. If the stage named *RTSOutput* shows *wtime* values that are significantly higher than the *utime* values, then the Document Processor Server is either waiting for the Indexing Cluster or the Status Server.

For systems where the Business Manager's Control Panel (BMCP) is used extensively to tune ranking, the Cache Server may become a system bottleneck. If the *wtime* is significantly higher than the *utime* for the document-processing stage named *RankTuning* then the Cache Server is probably the bottleneck.

Solution

When document-processing performance is too low, consider making the following changes:

- If any of the machines used by the Document Processing Cluster, Status Server, Content Distributor or Anchor Cluster are paging then make more memory available to these processes.
- If the machines in the Document Processing Cluster show high CPU utilization, add more machines with more Document Processing Servers to this cluster. If these machines show low CPU utilization, then consider adding more Document Processing Servers to the existing machines.
- Documents are submitted through the FAST Content API in batches. If the Status Server is the performance bottleneck, then increasing the batch size will usually improve overall performance. Increasing the batch size too much will make transferring batches to Document Processor Servers a bottleneck, and this can be seen by very busty CPU utilization in the individual Document Processor Server processes. A good batch size is typically a few megabytes of content.

Other Potential Performance Bottlenecks

Network utilization may become a bottleneck for content dynamics, but this is normally not the case for medium scale systems that do not share network segments with other systems.

Content Volume Performance

Content volume is usually not a problem in itself. Scaling with content volume generally requires adding columns to the Search and Indexing Clusters to ensure that these clusters operate within their other performance requirements.

The main problem with very high content volumes is that high content volumes require a lot of storage capacity in the Search and Indexing Clusters. In combination with the fact that these clusters typically represent the bulk of the machines used for a deployment, this can drive costs. Disk usage for a single row in the Search and Indexing Cluster is proportional to the content volume. All rows keep copies of the same content. Thus the total storage requirement is also proportional to the number of Search and Indexing Cluster rows.

The indexers generate data structures that are very efficient for searching. Different data structures are optimal for supporting different search features. This makes storage requirement highly dependent on features enabled, and more features generally requires higher total storage requirement. Table 4-1 illustrates the cost of some features in terms of index size.

Table 4-1 Cost of search features in terms of index storage requirement

Feature	Index Size	Enabled by default
Phrase search	+240%	Yes
Proximity support	+130%	Yes
Full wildcard support	+160%	No
Lemmatization	+160%	Yes (if selected during installation)

Chapter 5

Small-Scale Deployment

About this Chapter

This chapter provides deployment guidelines for small-scale installations of FAST InStream.

It includes:

- Characteristics of Small-Scale Installations
- Disk and Memory Usage in a Default Configuration Installation
- Minimal Deployment Example
- Disabling Individual Components to Free Disk Space and Memory

Characteristics of Small-Scale Installations

This chapter provides deployment guidelines for small-scale installations of FAST InStream. A small-scale installation typically consists of one or a few machines, and there is usually not a significant cost incentive for optimizing the system configuration. A small-scale system is characterized by not being constrained by available hardware resources for its performance.

Disk and Memory Usage in a Default Configuration Installation

Table 5-1 summarizes disk and memory usage for a typical installation of FAST InStream with default configuration. For some components, resource usage varies with scale.

Table 5-1 Disk and Memory Usage for Small-Scale Installations

Component	Memory (MB)	Disk (MB)
Crawler	100	S
Content Distributor	10	0
Processor Engine	300	0
Search Engine	1000	2.5S
Query and Result Engine	300	0
Administration	100	varying
Base Software	N/A	800

Note! *S* denotes the total size of the content.

Search Cluster disk usage depends heavily on the system configuration and the characteristics of the input data. The table quotes an index expansion factor of 2.5, which is typical for default configuration without lemmatization when indexing web content comprised of HTML documents. For web content, the index expansion factor typically varies between 1 and 4 depending on the configuration. For structured content, such as database content, the index expansion factor is usually higher. It is necessary to perform benchmarking on representative content to discover the correct index expansion factor for a particular system.

Minimal Deployment Example

Although it is possible to deploy FAST InStream on a single machine for demonstration and testing purposes, this is not recommended for a production system. Figure 5-1 depicts a typical minimal deployment of FAST InStream comprising four machines in addition to the load balancer. The shaded boxes represent physical machines, and the other boxes represent servers running on these machines.

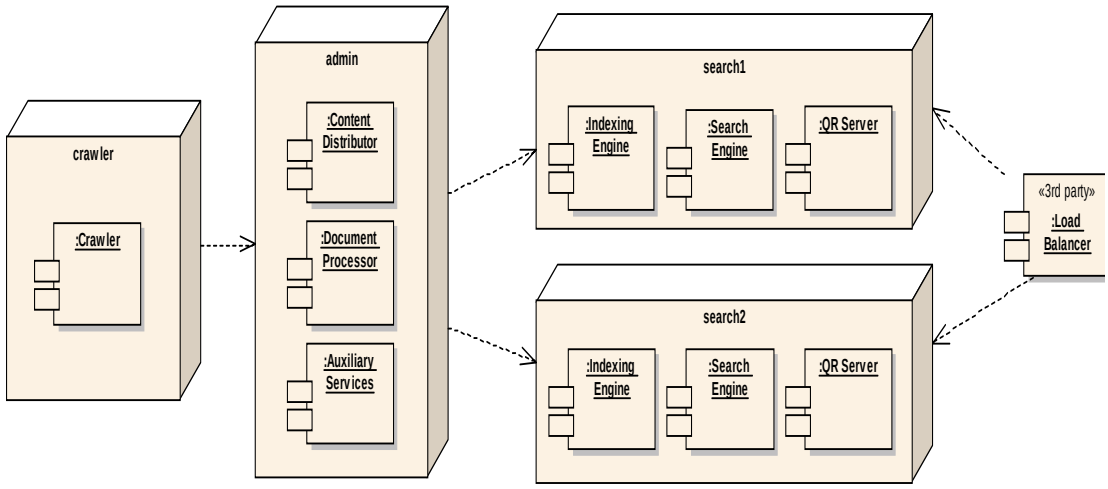

Figure 5-1 Small-Scale FAST InStream Deployment Example

This system has:

- one machine dedicated to crawling. The Crawler has the potential to consume a lot of disk and CPU resources. So it is usually best to dedicate one machine exclusively for crawling to avoid the risk of starving other servers in the system.
- one machine running Content Distributor, Document Processor and auxiliary services, including administration, logging and license server.
- two machines in a fault-tolerant configuration running Indexing, Search, and Query and Result Servers. Robustness against single-machine failures to ensure fail safe operation for the Search Cluster is usually required even for small-scale systems. The Search Cluster should use dedicated machines to provide predictable search performance.
- a load balancer to provide query load balancing and failover support for the Search Cluster machines. Alternatively, other mechanisms external to FAST InStream can be used to provide the necessary failover support in the front-end.

Disabling Individual Components to Free Disk Space and Memory

If you have limited hardware resources, you may free disk space and memory by disabling individual components. Table 5-2 shows the hardware usage of individual FAST InStream components that can be removed from a full FAST InStream installation.

This may serve as an indication of how you can free disk space and memory.

Table 5-2 FAST InStream Components and Hardware Usage

Component	Disk	Memory	Virtual Memory	Comments
Crawler	Varies	30 MB - 2 GB	30 MB - 2 GB	In a medium and large scale installation, there are usually one or more dedicated Crawler nodes. Disk requirements are about 1.0 - 1.2x of the total content crawled. Memory requirements vary with the number of sites and documents crawled. A Crawler that has downloaded 3.5 MB documents and crawling 3.2 documents per second is using about 600 MB of memory.
Anchor Server	Varies	~320 MB, varying with the number of docs	10 - 300 MB varying with the number of docs	The Anchor Server footprint can be reduced from 300 MB down to ~10 MB, with a performance penalty. Disk usage of the Anchor Server is about 7 GB / 1 M documents.
J2EE	~150MB	300 MB	100 MB	J2EE is used in the default search front-end, in rank tuning, reporting, and logging.
MySQL	50MB	40 MB	10 MB	Used for storing rank tuning information.
FAST Administrator User Interface	20MB	4 MB	2 MB	Used for web based administration
Log Server	Varies	7 MB	4 MB	Used for collecting logs from various FAST InStream components.
License Manager	N/A	5 MB	3 MB	

Table 5-2 FAST InStream Components and Hardware Usage

Component	Disk	Memory	Virtual Memory	Comments
Advanced Linguistics	N/A	70 MB	70 MB	***Note!*** For details on how to reduce memory consumption, refer to Chapter 7 *Benchmarking*, section *Lemmatization Memory Consumption*

For a small installation with 100k documents of an average size of 2k , no fault tolerance or failover requirements and moderate QPS requirements (<10qps), a single node with 2x2.8GHz CPUs and 1GB RAM is recommended.

Note! Slower CPUs will just reduce feed rate and search performance.

Disk requirements are approximately 3GB.

The list below shows some key numbers from a small sample FAST InStream installation with a hardware of 1x1.0GHz and 512MB RAM loading 100k documents with random text:

- Time to feed documents: 2522 s
- Average feed speed: 39.7/s
- Average allocated memory used while feeding: 1135 MB
- Max allocated memory used while feeding: 2097 MB
- Average resident memory used while feeding: 297 MB
- Max resident memory used while feeding: 549 MB
- Space used for index (max): 1310 MB
- Space used for index: 782 MB
- Space used for fixml: 150 MB
- Allocated memory after indexing: 873 MB
- Resident memory after indexing: 196 MB
- Resident and virtual memory usage for indexing 100k documents:

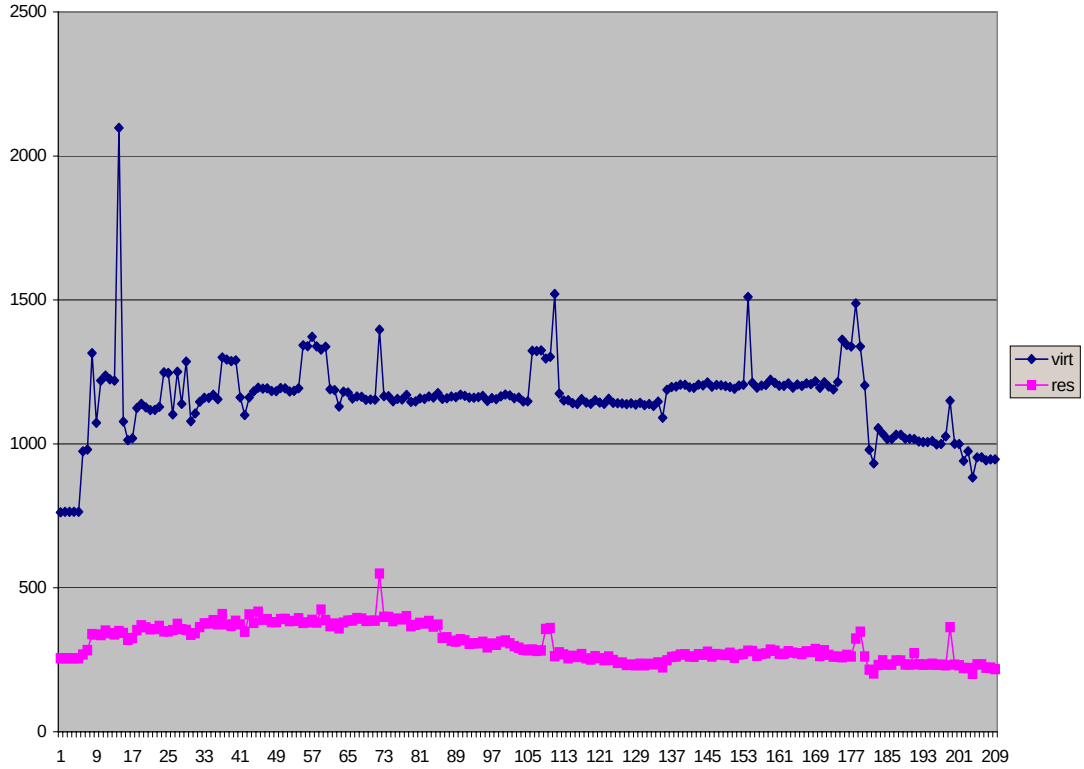

Figure 5-2 Sample Resident and Virtual Memory Usage for Indexing 100,000 documents

Chapter 6

Practical Considerations

About this Chapter

This chapter elaborates on a few practical issues that are important when deploying FAST InStream in certain scenarios.

It includes:

- Deployment Behind a Firewall
- API Security
- Disk Performance Issues
- Network Performance Issues

Deployment Behind a Firewall

FAST InStream can be deployed behind a firewall. The firewall must be configured to make a number of FAST InStream interfaces available from outside the firewall depending on the requirements to the system. It is recommended that you make the following interfaces available outside the firewall:

- Search interface
- Content submission interface
- Administrator interface

The search interface will normally be available via an application front-end, or at least a load balancer, as shown in Figure 2-1. In this case, it is not necessary to expose any network ports of the FAST InStream system to make the search interface available. If direct access to the FAST InStream search interface is required, access to port 15100 on the Query and Result Servers is required assuming that a default configuration is used. For systems that use customized port ranges, the relevant port numbers can be found in the **System Overview** selection of the FAST InStream Administrator Interface.

Some systems may require content push from outside the firewall. Clients that push content to FAST InStream using the FAST Content API must have access to the FAST InStream Configuration Server, Content Distributor, Status Server, and Name Server. By default, these servers are not tied to specific server ports. Contact FAST Professional Services for assistance with configuring these services at fixed ports to enable easy access to the FAST Content API through a firewall.

Administration of a FAST InStream installation from outside a firewall requires access to a large number of machines and a large number of ports. The best way to enable remote administration is to install a small HTTP proxy inside the firewall and access the FAST InStream Administrator Interface via that proxy. The firewall must be configured to permit access to the proxy from outside.

API Security

When APIs are exposed outside a data center security perimeter as described in the previous section, it is recommended that you secure these interfaces against unauthorized access. FAST InStream supports securing both the FAST Content API and the FAST Query API by the use of SSL. SSL is used both to authenticate clients accessing the APIs and to secure the connection against eavesdropping by encrypting all data transmitted. This provides much stronger protection than the IP filtering usually provided by standard firewalls.

SSL support is provided by an SSL Proxy Server. For information on how to set up an SSL Proxy Server, contact FAST Professional Services.

Disk Performance Issues

FAST InStream relies heavily on data structures stored on disk, even for performance critical operations. Disk performance is one of the defining factors for overall system performance. There are multiple factors that influence disk performance. This subsection summarizes some information on this subject.

A disk-based storage system is characterized by several parameters, and focus is usually on the storage capacity. However, all parameters must be considered and are important when specifying a high-performance FAST InStream installation. The most important parameters are:

- disk transfer rate (very important for sequential file operations typical during indexing)
- disk transaction rate (very important for small random access operations typical during searching)

These parameters again are based on physical disk attributes such as storage capacity, disk rotation speed, disk system bus bandwidth, or disk controller type and features

The disk transaction rate is often the bottleneck in systems with a very high query load. The only way to increase the transaction rate of the storage system, while keeping other performance metrics unchanged, is to add more physical disks. A common technique to increase performance is to use several small disks combined to a single logical disk by using RAID 0, but even a RAID system must be configured correctly. The RAID stripe size is important for performance. A large stripe size of 4 MB can provide good results as this reduces the transaction rate.

For very high-performance systems, other processes may interfere with the storage system to reduce performance in unacceptable ways. Operating systems or dedicated system monitoring tools may for instance scan disks in an effort to discover failures. Modern disk drives usually support SMART technology which also influences the performance of the disks by executing self-tests as part of normal operation. This can usually be disabled or tuned.

FAST recommends using 70 GB SCSI disks with a spindle speed of 15000 RPM for medium-scale systems. Larger disk partitions must be made by using RAID 0 to combine several physical disks into a single partition as described above.

FAST does not recommend using separate storage systems, such as Network Attached Storage (NAS) or Storage Area Networks (SAN), to replace physical disks in the machines. Even though many such systems boast impressive performance figures and cost-efficient storage, they will often constitute a bottleneck in a medium-scale high- performance installation of FAST InStream.

Network Performance Issues

This subsection provides an overview of network traffic in a FAST InStream system to facilitate dimensioning of the network infrastructure. Network traffic is dominated by four factors:

- inbound content to be processed by the system
- internal distribution of content in the system
- outbound query results
- internal distribution of queries and result sets in the system

Inbound bandwidth for content submission is determined by the content submission rate and the document size. Both of these performance metrics should be part of the system requirements as described in *Functional and Performance Requirements* on page 15. Using the nomenclature from the Appendix B *System Requirements Checklist*, the inbound bandwidth for content submission is given by:

$$DocInputBandw = UpdateRate = TotDocSize / TotDocCount$$

The above equation gives the peak bandwidth as defined by the system requirements. The actual bandwidth usage for a running system can be estimated by using information provided in the administrator interface. An estimate for *UpdateRate* can be found by first selecting **Matching Engines**, then selecting a Search Server, and taking the value for

Receiver affecting API calls rate. Remember to sum the rate for all Search Servers in a single Search Cluster row.

Internally, the data source sends content to the Content Distributor, one of the Document Processor Servers, all Indexing Servers constituting a column, and finally to all Search Servers constituting a single Search Cluster column.

The outbound bandwidth for returning query result sets is determined by the query rate and the size of result sets. The peak query rate is one of the performance requirements to the system. The size of a result set depends on the index profile used and the number of results returned. For the default index profile for standard web content and 10 results, the size of the result set is typically 20 KB.

ResOutputBandw = QueryRate

Internally, search results are returned from all Search Servers in a Search Cluster row but this data is encoded differently than the result set returned from the system. First, each Search Server returns up to 1000 bytes representing result candidates to the Query and Result Server when the Query and Result Server asks for 10 results. Then the Query and Result Server retrieves summary information for the results actually becoming part of the result set from the Search Servers that returned these results.

Small and medium-scale systems usually do well with a simple switched network architecture. The burstyness of the network traffic may become a problem for large-scale systems, and this may require changes to the network architecture. This kind of problem usually manifests itself as packet loss in network switches and the Query and Result Server.

Chapter 7

Benchmarking

About this Chapter

This chapter gives you some benchmarking indications.

It includes:

- Overview
- Search Server Capacity
- Search Server Disk Usage
- Query and Result Server Capacity
- Document Processor Server Capacity
- Lemmatization Memory Consumption

Overview

FAST InStream is a platform product that offers extensive possibilities for configuration and customization and can be deployed across a wide range of hardware platforms and configurations. As this guide cannot cover performance for all possible configurations, it is very important that you do benchmarking both during deployment planning and before the actual deployment as part of a live system.

Benchmarking is used to verify that the system or a single server meets its performance requirements. To determine the capacity of the system or a server, it is recommended that you repeat benchmarking at different loads and find the highest load where the performance requirements are satisfied. It is important that the component being benchmarked sees a realistic load. This load comprises:

- content updates
- query load
- feature set
- total number of documents and document size

In all cases, the load must be as close to the expected real-life situation as possible. For instance, the content updates should match both the update rate and content type of the real-life system. The data used for loading must also be reasonably large to avoid caching effects in FAST InStream to affect the results. You may often obtain realistic loads from logs of existing systems.

When benchmarking a system component, it is important to ensure that no other component in the test system constitutes a bottleneck. For instance, when benchmarking the query performance of a Search Server, make sure that the Query and Result Cluster is not the component that actually limits the query rate. You can achieve this by either dimensioning the benchmark system with ample Query and Result Cluster capacity or by bypassing the Query and Result Cluster altogether. Sometimes the system bottleneck can even be the benchmarking system itself or the network. Test the benchmark results by increasing the capacity of the component being benchmarked and verify that the expected performance improvement can be measured using the same benchmarking setup.

It is possible to benchmark scaled down versions of the system to estimate the capacity of the complete system. This is a useful methodology when dimensioning a relatively large installation. When doing this, it is important that you understand how the full-scale system will influence performance. For instance, the slowest Search Server in a row will always limit the query performance of a Search Cluster row. Benchmarking a shortened row will thus always yield slightly better results than benchmarking on a full row.

Note! Contact FAST Professional Services for assistance with specifying benchmarking setups and interpreting results.

Search Server Capacity

Figure 7-1 shows a set up that is suitable for benchmarking the capacity of a Search Server. This set up consists of a single Search Server being benchmarked. Additionally, a Document Processor Server, and Indexer Server and a Query and Result Server are used to ensure a realistic environment for the Search Server. You need to ensure that the Document Processor Server, the Indexer Server or the Query and Result Server do not constitute a bottleneck while benchmarking.

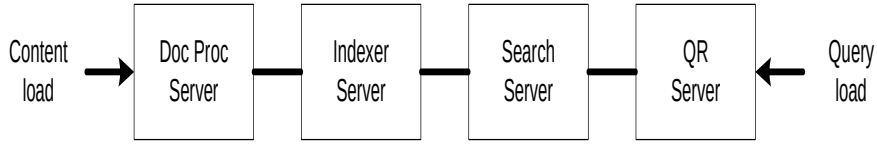

Figure 7-1 Setup for Benchmarking Capacity of a Search Server

You discover the capacity of a Search Server by applying load to the server and then measuring its response characteristics. By trying different loads and measuring whether each load satisfies or violates the response time requirements, you can determine the capacity. Thus, finding the capacity of a server is very much a trial-and-error process.

The load on a Search Server is defined by the number of document updates per second, the number of queries per second, and the total number of documents being handled by the server. The response characteristics are defined by the time to respond to a query and the time to index an updated document. Figure 7-2 shows a typical distribution of query response time, which is characterized by a mean value and a tail. Requirements to response time usually restrict both the mean response time and the length of the tail in the distribution.

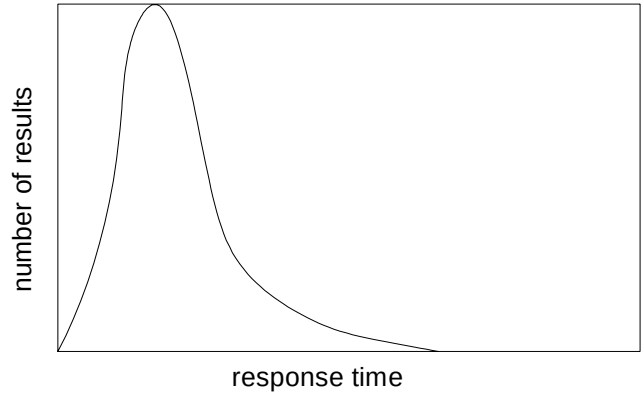

Figure 7-2 Typical Distribution of Query Response Time

Use these techniques to determine the parameters *SearchDoc*, *SearchQuery*, and *SearchIndex* used in the template in the Appendix C *Dimensioning Template*.

Search Server Disk Usage

Measuring the server disk usage is quite simple. Use a setup comprised of one Document Processor Server, one Indexer Server and one Search Server and submit content to it while monitoring the combined size of the following directories on the Search Server:

- *$FASTSEARCH/data/data_fixml* (UNIX) or *%FASTSEARCH%\data\data_fixml* (Windows)
- *$FASTSEARCH/data/data_index* (UNIX) or *%FASTSEARCH%\data\data_index* (Windows).

The index expansion factor, *IndexExp*, in the template in Appendix C *Dimensioning Template* is defined as the fraction between the maximum data size and the total size of the content submitted to the system.

Note that the Search Server disk usage fluctuates over time as indexes are being updated. It is very important to take these fluctuations into account and use the maximum disk usage when computing *IndexExp*. Figure 7-3 illustrates the fluctuations in Search Server disk usage for a search server that starts receiving new content. Each peak corresponds to new content being incorporated in the index.

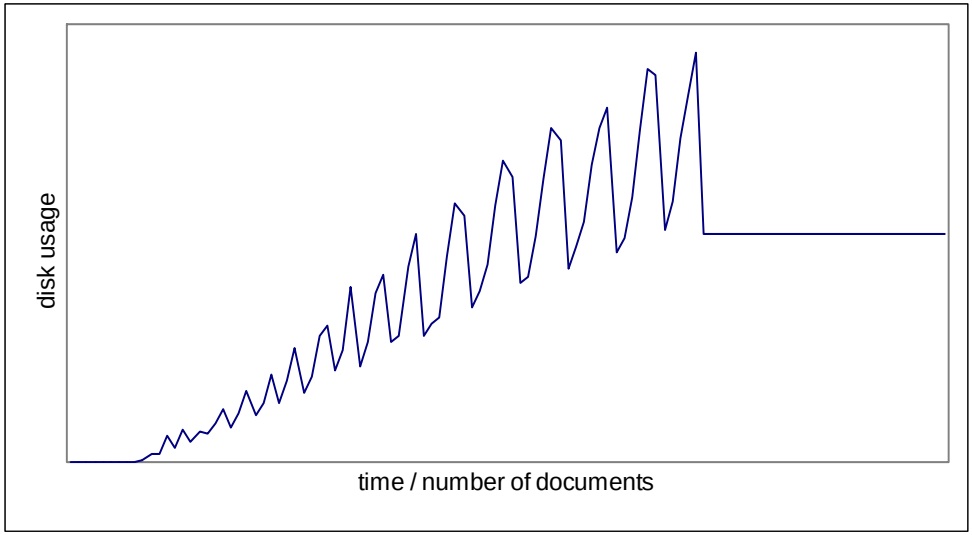

Figure 7-3 Search Server disk usage

Query and Result Server Capacity

Benchmarking the Query and Result Server capacity closely follows the approach outlined in subsection *Search Server Capacity* on page 41 about Search Server benchmarking. The difference is that the Query and Result Server capacity does not depend on the content update rate or the number of documents handled by a Search Server.

Be sure that the Search Server does not constitute a bottleneck when benchmarking the Query and Result Server. It is usually necessary to use several Search Servers to determine the maximum capacity of a Query and Result Server. Use this approach to determine *QRQuery* in the template in Appendix C *Dimensioning Template*.

Document Processor Server Capacity

Benchmarking the Document Processor Server capacity uses a setup similar to the one in Figure 7-1 with the exception that it is not necessary to use a Query and Result Server and load the Search Server with queries. Be sure that the Search Server does not constitute a bottleneck while benchmarking. It may be necessary to use several Search Servers.

The capacity of the Document Processor Server is determined by submitting documents at a certain rate to it and measuring that the Search Servers receive documents at the same rate. You need to test different rates until you can determine the maximum capacity. Use this approach to determine *DPUpdate* with the template in the Appendix C *Dimensioning Template*.

Lemmatization Memory Consumption

Lemmatization impacts the memory consumption, as it is based on automata which consume memory. The memory impact depends on which type of lemmatization you are applying.

Note! For a functional description of what lemmatization is and how it works, refer to Chapter 10 *Advanced Linguistic Processing*, section *Lemmatization* in the *Product Overview Guide*.

Lemmatization Memory Consumption

Table 7-1 gives you an overview of how the two types of lemmatization impact memory:

Table 7-1 Lemmatization Memory Consumption on Unix Systems

	Virtual Memory	Physical Memory
Lemmatization by Reduction	(N + M) * the size of *<LANG>_<POS>_red.aut*	the size of *<LANG>_<POS>_red.aut*
Lemmatization by Expansion	N * the size of *<LANG>_<POS>_exp.aut*	the size of *<LANG>_<POS>_red.aut*

with:

- N being the number of document processor instances in use,
- M being the number of query processor instances in use,
- *<LANG>* being the ISO 639 language code for the language applied, such as *en*, *fr*, or *ru*,
- *<POS>* being a part of speech code indicating the combination of parts of speech processed for lemmatization (N representing nouns, A representing adjectives, and V representing verbs)

Note! The figures given in Table 7-1 are based on one language at a time. If you apply lemmatization for multiple languages, you need to multiply the memory consumption accordingly.

Reducing Memory Consumption

You can reduce memory consumption by

- commenting out all languages you do not need in the dictionary configuration files for both
  - lemmatization by reduction (*%FASTSEARCH%\etc\DictionaryConfig.xml* (Windows) or *$FASTSEARCH/etc/DictionaryConfig.xml* (Unix))
  - and lemmatization by expansion (%FASTSEARCH%\etc\DictionaryConfig_exp.xml (Windows) or $FASTSEARCH/etc/DictionaryConfig_exp.xml (Unix))
- and reducing the number of parts of speech for which you lemmatize. For instance, lemmatizing only nouns (N) requires less memory than lemmatizing both nouns and adjectives (NA).

Chapter 8

Deployment Case Study

About this Chapter

This chapter contains a case study demonstrating the use of these guidelines in planning deployment for a fictitious system.

It includes:

- Application Overview
- Initial Dimensioning
- Component Optimization
- Final Dimensioning

Application Overview

Figure 8-1 shows an overview of the example application where FAST InStream will be deployed. The application consists of a content database for storing unformatted content, a middleware layer providing functionality for rendering the content, and a web server for providing the content to the users of the service. Content producers update the data in the content database.

In this example FAST InStream will take content directly from the database. This is an attractive solution when possible since it is then easy to avoid polluting the content with any formatting and personalization that may be added by the middleware layer, for instance banner ads. The middleware layer uses the FAST InStream search service to provide the functionality behind the search function on the application.

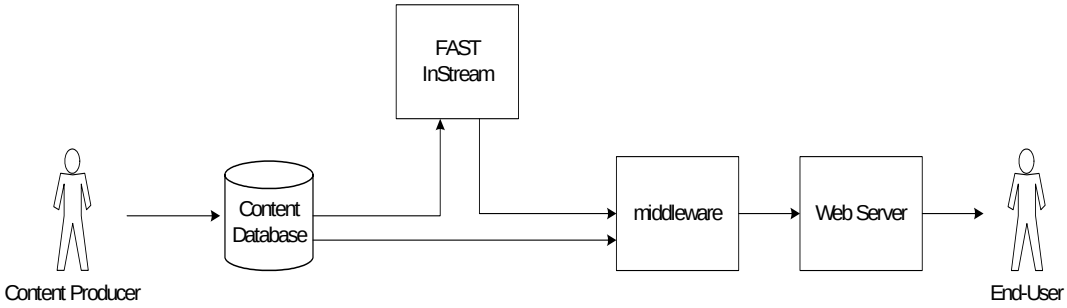

Figure 8-1 Sample Application Overview

Describe the system by answering the qualifying questions in Appendix B *System Requirements Checklist*:

1 Content volume:

- **a** The content database contains 23 million documents.
- **b** The average document size is 10 KB.
- **c** The total content volume is 230 GB.
- **d** The documents vary somewhat in size. No document is larger than approximately 50 KB.
- **e** The content database is expected to grow to 27 million documents, or approximately 270 GB of content, in the coming year.

2 Content dynamics:

- **a** Continuous indexing is required.

b The system must be able to handle a peak document update rate of one document update per second. The indexing latency is not critical, but new or updated documents should be indexed within a couple of minutes.

c Document updates are uniformly distributed.

3 Query rate:

a The system must be able to handle a peak query load of 75 queries per second.

b The highest query rate is between 8 AM and 5 PM. The query load is also higher at the beginning of each month. At no time have a query load of more than 70 queries per second been observed.

c The query rate is not expected to grow in the coming year.

4 Content characteristics:

a All content is HTML.

b All content stored in the content database is HTML content. Images and other non-text content are stored elsewhere and not included in the size estimates.

c All content is in English.

d It must be possible to search on the entire document text, document title only, document headers only, document URL, and the document modification date (mostly for filtering results).

e It must be possible to sort on relevance and modification date.

f The content is stored in the content database, which is hosted on a commercial RDBMS.

5 Feature set:

a Lemmatization is not required.

b Phrase indexing is required.

c Substring search is not required.

d Dynamic teasers is required.

e Categorization of results is required.

f The Business Manager's Control Panel is required.

g The system does not use custom features.

6 Availability requirements:

a The system must be able to handle queries even while indexing.

b The system must be able to handle queries even if a single machine fails. It is acceptable that new content is not indexed when a machine fails.

7 Platform:

a The system will be hosted on RH Linux 7.3 on an Intel platform.

Initial Dimensioning

By comparing the previous answers to the values in Table 3-1 we clearly see that this system qualifies as a medium-scale system. Also, the answers indicate that the system is homogenous. There are no big fluctuations in document size (Question 1 d) or requirements to content update dynamics (Question 2 c) that can be utilized for a more efficient deployment model. Changes in query rate (Question 3 b) does not help us much as the system must be dimensioned for the peak query rate. However, this can be useful if parts of the system must be disabled for maintenance during normal operation.

We proceed by using the template provided in Appendix C *Dimensioning Template* to compute an initial estimate for the number of machines required for deploying this system. We use values from Table 3-1 as the initial estimates for values in the System Information subsection of the dimensioning template. Table 8-1 shows the results from using the electronic dimensioning template. Note that we use the future estimated capacity requirements rather than the current requirements to avoid having to frequently resize the system.

Note that this estimate does not include machines for redundancy. We need one additional row in the Search Cluster and one additional machine in the Query and Result Cluster for the required level of fault-tolerance, adding another eight machines to the system.

Table 8-1 Initial Dimensioning Estimates for Example System

Application Information		
Parameter	**Parameter Name**	**Value**
Total content size	*TotDocSize*	270,000 MB
Total number of documents	*TotDocCount*	27,000,000
Average document size	*DocSize*	10,000 bytes
Peak query rate	*QueryRate*	75 queries / second
Peak document update rate	*UpdateRate*	1 document / second
System Information		
Parameter	**Parameter Name**	**Value**
Index expansion factor	*IndexExp*	2.5
Max document count per search node	*SearchDocMax*	4,000,000
Search node query capacity	*SearchQuery*	40 queries / second
Search node indexing capacity	*SearchIndex*	5 documents / second
Query and Result node query capacity	*QRQuery*	50 queries / second
Document processing node capacity	*DPUpdate*	40 documents / second

Table 8-1 Initial Dimensioning Estimates for Example System

Dimensioning Search Cluster		
Parameter	**Parameter Name**	**Value**
Number of search columns:	*SearchCols*	7
Number of search rows:	*SearchRows*	2
Document count per search node:	*SearchDoc*	3,857,143
Disk per search node:	*SearchDisk*	96,429 MB
Memory per search node:	*SearchMem*	1,000 MB
Dimensioning Query and Result Server		
Parameter	**Parameter Name**	**Value**
Number of Query and Result Servers:	*QRCount*	2
Disk per Query and Result Server:	*QRDisk*	0 MB
Memory per Query and Result Server:	*QRMem*	300 MB
Dimensioning Document Processing Server		
Parameter	**Parameter Name**	**Value**
Number of Doc Proc Servers:	*DPCount*	1
Disk per Doc Proc Server:	*DPDisk*	0 MB
Memory per Doc Proc Server:	*DPMem*	300 MB

Component Optimization

We will now try to optimize the system components in an attempt to improve their performance. In this way we may be able to reduce the number of machines required for the system and thus reduce the cost of the system. The Search Cluster represent the bulk of the machines so we will have the biggest potential for reducing the cost of the system by optimizing this part. On the other hand, optimizing the Document Processing Cluster will not help at all since there is only one Document Processing Server.

Figure 8-2 shows a setup of three machines suitable for benchmarking and tuning the Search Server. It consists of a single Document Processing Server for generating a content update load to the Search Server. This machine also contains the auxiliary FAST InStream services, such as the Configuration Server and the Administration Server. The Query and Result Server generates a query load to the Search Server.

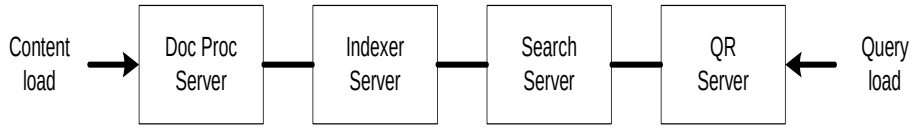

Figure 8-2 Setup for Benchmarking of a Search Server

The purpose of the tuning and benchmarking is to achieve at least one of the following:

- reduce the number of search rows. This requires that a Search Server is able to handle at least 75 queries per second.
- reduce the number of columns. This requires that a Search Server is able to handle a content volume that is significantly larger than the estimated 4 million documents. Each Search Server must also handle more document updates since fewer columns will share the document update load.

We start by benchmarking a configuration corresponding to the initial deployment estimate. This means that we benchmark with a load of 38 queries per second, 0.15 document updates per second and an index of 3.9 million documents. We only benchmark one Search Server, and this corresponds to the load seen by a single server in the system.

In our hypothetical system, we assume that the Search Server easily handles this load. We then try to increase the document volume to 6 million documents and rerun the benchmarking. This corresponds to only 5 columns, so we must also increase the content load to 0.2 document updates per second. We re-run the benchmarking and still the system is able to handle the load with an acceptable response time.

To be able to reduce the system to only four columns, we'll have to load each Search Server with 6.8 million documents and 0.25 document updates per second. We try another benchmarking of that setup, but our hypothetical system is not able to handle this load. The query response time grows too much and becomes unacceptably large.

We can run similar tests to figure out whether we are able to increase the query capacity of our search server to 75 queries per second. We may have to reduce the document volume below 4 million documents to do that, but the potential saving of all machines in a row may warrant that. In the end, the configuration that requires the fewest machines is often the best. The procedure is similar to the procedure for reducing the number of columns; details are not included in this guide.

Final Dimensioning

Through simple benchmarking we found that our Search Server has a higher capacity than our first estimate indicated. Table 8-2 shows updated system dimensioning information after component optimization, excluding machines for fault-tolerance.

Table 8-2 System Dimensioning after Optimization

Application Information		
Parameter	**Parameter Name**	**Value**
Total content size	*TotDocSize*	270,000 MB
Total number of documents	*TotDocCount*	27,000,000
Average document size	*DocSize*	10,000 bytes
Peak query rate	*QueryRate*	75 queries / second
Peak document update rate	*UpdateRate*	1 document / second
System Information		
Parameter	**Parameter Name**	**Value**
Index expansion factor	*IndexExp*	2.5
Max document count per search node	*SearchDocMax*	6,000,000
Search node query capacity	*SearchQuery*	40 queries / second
Search node indexing capacity	*SearchIndex*	5 documents / second
Query and Result node query capacity	*QRQuery*	50 queries / second
Document processing node capacity	*DPUpdate*	40 documents / second
Dimensioning Search Cluster		
Parameter	**Parameter Name**	**Value**
Number of search columns:	*SearchCols*	5
Number of search rows:	*SearchRows*	2
Document count per search node:	*SearchDoc*	5,400,000
Disk per search node:	*SearchDisk*	135,000 MB
Memory per search node:	*SearchMem*	1,000 MB
Dimensioning Query and Result Server		
Parameter	**Parameter Name**	**Value**
Number of Query and Result Servers:	*QRCount*	2
Disk per Query and Result Server:	*QRDisk*	0 MB
Memory per Query and Result Server:	*QRMem*	300 MB

Table 8-2 System Dimensioning after Optimization

Dimensioning Document Processing Server		
Parameter	**Parameter Name**	**Value**
Number of Doc Proc Servers:	*DPCount*	1
Disk per Doc Proc Server:	*DPDisk*	0 MB
Memory per Doc Proc Server:	*DPMem*	300 MB

Table 8-3 summarizes all equipment needed for deploying the system:

Table 8-3 System Deployment Summary

Count	Description	Comments
15	Search Cluster	5 columns and 3 rows, including one extra row for redundancy
3	Query and Result Engine	Including one extra server for redundancy
1	Document Processor Engine	
1	Auxiliary services	
20	**Total**	

Appendix A

Deployment Checklist

This appendix contains a list of deployment issues that can be checked easily. It is recommended to use this checklist as a last step in deployment planning before actually starting full-scale system rollout.

Note! Items in the list are numbered, but this numbering does not imply a prioritization of the issues or that they must be checked in this particular order. The numbering is solely for facilitating easy reference to checklist items.

1. Check the *Installation Guide* and the FAST InStream release notes. Make sure that any issues listed in the release notes regarding features, performance, or the deployment platform are either irrelevant for this system or taken into account in the deployment.
2. Make sure there is a load balancer connecting directly to the machines constituting the Query and Result Engine. Refer to subsection *Fault-Tolerance* on page 8 for more details on the load balancer.
3. Verify that all machines in a Search Cluster row have approximately the same performance. The performance of a Search Cluster is determined by its slowest node, so using a few powerful machines on an otherwise slow row is a waste of resources. If equipment of different performance is available, it is better to try to assemble a complete row using the more powerful equipment.
4. Verify that no server sharing a physical machine with other servers constitutes a performance bottleneck in the system. As a general rule, in a medium-scale system all servers should run on dedicated machines.
5. Make sure that machines dedicated to the Document Processor Engine are fully utilized. A dedicated machine is usually able to run two Document Processor Servers per CPU installed in the machine.
6. Verify that processes have been established to monitor hardware and software to detect failures early and to take the necessary steps to remedy the situation.
7. Verify that processes have been established for monitoring system performance and load and in a timely manner initiate a process for scaling up the system if that becomes necessary.

Appendix B

System Requirements Checklist

This appendix contains a number of questions aimed at clarifying critical system requirements. Answering all of these questions for a system ensures a good understanding of the system requirements. Subsection *Functional and Performance Requirements* on page 15 provides more information.

1 Content volume:

- **a** What is the total number of documents that will be searchable?
- **b** What is the average document size?
- **c** What is the total amount of data to be processed by the system?
- **d** What is known about the document size statistics? For instance, is it possible to partition the document set in sets with very different document size distributions?
- **e** How much is the total content size expected to grow in the next year?

2 Content dynamics:

- **a** Is continuous (real-time) indexing required, or is periodic (batch) indexing sufficient?
- **b** What is the rate of document updates in the system, and what is the maximum latency permitted from a document is updated to it is indexed and becomes searchable?
- **c** What is known about the document change statistics? For instance, it is possible to partition the document set in sets with very different update rates or indexing latency requirements?

3 Query rate:

- **a** What is the expected peak query rate?
- **b** Will there be significant periodic changes in the query rate?

c How much is the query rate expected to grow in the next year?

4 Content characteristics:

a What is the format of the content? (HTML, XML, PDF, Microsoft Word, database, or other?)

b How much of the document size is actual indexable content, and how much is figures, formatting or meta-information not to be indexed?

c How much of the content is in Eastern-European languages or other languages not covered by the Latin character set? What languages and character sets are used?

d Which document fields must be separately searchable?

e Which sorting options must be available for the result sets?

f Where is the content stored? FAST InStream can retrieve content from web servers, file servers, databases, and a number of other systems.

5 Feature set:

a Is lemmatization or thesaurus support required?

b Is phrase indexing required?

c Is substring searching required?

d Are dynamic teasers required?

e Is categorization of results required?

f Is the Business Managers Control Panel required?

g Are any custom features used? If yes, make sure that all performance implications of these features are known and adjust the recommendations in this guide accordingly.

6 Availability requirements:

a Is the system required to handle queries during indexing? (This question is only relevant for systems using periodic or batch indexing.)

b What fault-tolerance is required? What service requires fault-tolerance, and how should fault-tolerance be handled? Possibilities included full fault-tolerance, degradation of service quality, service unavailability.

7 Platform:

a What platform will be used for hosting the system?

Appendix C

Dimensioning Template

This appendix provides information to compute the number of instances of the FAST InStream system components using information about the system.

Table C-1 defines the relationship between a number of performance metrics for the system. Values under the system information heading must be based on benchmarking relevant content, equipment and configuration to get accurate dimensioning results. Hints on benchmarking can be found in Chapter 7 *Benchmarking*.

Note that the disk usage per node specifies only disks required by FAST InStream for storing application data. The quoted disk footprint is the minimum required disk available after installing FAST InStream, the operating system, and any other required software. The disk footprint of the base FAST InStream software is approximately 800 MB (approximately 2 GB on AIX systems).

Table C-1 Initial Dimensioning Estimates for Example System

Application Information		
Parameter	**Parameter Name**	**Calculation / Value**
Total content size	*TotDocSize*	
Total number of documents	*TotDocCount*	
Average document size	*DocSize*	*TotDocSize / TotDocCount*
Peak query rate	*QueryRate*	
Peak document update rate	*UpdateRate*	
System Information		
Parameter	**Parameter Name**	**Calculation / Value**
Index expansion factor	*IndexExp*	
Max document count per search node	*SearchDocMax*	
Search node query capacity	*SearchQuery*	queries per second
Search node indexing capacity	*SearchIndex*	documents per second
Query and Result node query capacity	*QRQuery*	queries per second
Document Processing node capacity	*DPUpdate*	documents per second
Dimensioning Search Cluster		
Parameter	**Parameter Name**	**Calculation / Value**
Number of search columns:	*SearchCols*	Maximum: *TotDocCount / SearchDocMax* *Update Rate / SearchIndex*
Number of search rows:	*SearchRows*	*QueryRate / SearchQuery*
Document count per search node:	*SearchDoc*	*TotDocCount / SearchCols*
Disk per search node:	*SearchDisk*	*SearchDoc x DocSize x IndexExp*
Memory per search node:	*SearchMem*	1 GB
Dimensioning Query and Result Server		
Parameter	**Parameter Name**	**Calculation / Value**
Number of Query and Result Servers:	*QRCount*	*QueryRate / QRQuery*
Disk per Query and Result Server:	*QRDisk*	0 MB
Memory per Query and Result Server:	*QRMem*	300 MB

Table C-1 Initial Dimensioning Estimates for Example System

Dimensioning Document Processing Server		
Parameter	**Parameter Name**	**Calculation / Value**
Number of Doc Proc Servers:	*DPCount*	*UpdateRate / DPUpdate*
Disk per Doc Proc Server:	*DPDisk*	0 MB
Memory per Doc Proc Server:	*DPMem*	300 MB

Index