Sterling Integrator®

Clustered Installations

Version 5.1

Sterling Commerce
An IBM Company

Contents

Clustering Overview

Overview of Using IFC Clustered Installations

A clustered installation of the application connects one database to more than one installation of the application. It differs from a federated system of application installations. Each installation in a federated system uses its own database. Installations in a federated system communicate with each other through URLs at each installation. In clustering, servers are typically in one location, while installations in a federated system can be at any location. An application cluster can be one node in a federated system. For more information about federated systems, refer to the System Administration Guide.

Note: This clustered installation documentation is a general description of how clustering works. For specific information about installing or patching a cluster configuration, refer to the *Installation Guide*.

Clustered installations can help your operation of the application in the following areas:

- Reliability, which ensures correct operation.
- Availability, which minimizes scheduled and unscheduled downtime.
- Scalability, which ensures that business processes can handle changes in volume.

You can use clusters for failovers. That is, any time one server fails, the other server simply takes over. Clusters are a more simple failover option than any of the system-level failover options for most installations and can be combined with system-level failover options to increase availability.

The cluster solution has the following characteristics:

- Allows the use of all cluster nodes all of the time to handle peak processing volumes.
- Fewer technical issues than the system-level high availability solutions.
- Portability. A change to a different platform doesn't require significant effort.
- A single vendor/technology solution.

All of the application nodes in a cluster share the same database server, but little else is shared between nodes. A cluster operates like a single node installation with the capabilities to communicate with other nodes to exchange information, distribute load, and perform other tasks.

Your decision to use a clustered system is affected by your performance priorities:

- For high availability and failover, evaluate each of the following approaches:

- Hot standby or backup system
- Clustering

  When considering these options, all single points of failure must be addressed to achieve a truly highly available system.

- For load balancing and performance, evaluate each of the following approaches:
  - Scaling vertically, by increasing resources on the same computer (such as CPUs or memory).
  - Scaling horizontally, by clustering.

  When considering these options, future growth and required service levels also must be evaluated.

This clustered installation documentation can help you decide about whether to install a clustered version of the application. It also can give you background information before you set up a clustered installation of the application.

Clustering Benefits

Benefits of Using an IFC Clustered Installation

The following sections describe how clustered installations can help your operation of the application in the following areas:

- Reliability, which ensures correct operation.
- Availability, which minimizes scheduled and unscheduled downtime.
- Scalability, which ensures that business processes can handle changes in volume.

All three of these areas contribute to the serviceability of your application installation.

Reliability of IFC Clustered Installations

Mission-critical business processes require reliable systems. As organizations rely more on automated processes, the consequences of even small, unintentional outages can be severe:

- Payments not made
- Orders not received
- Products not delivered

A reliable application environment requires a variety of other reliable components, including the network, disk storage, and database. Redundancy improves reliability, as measured by Mean Time Between Failures (MTBF).

A reliable application installation makes use of the following:

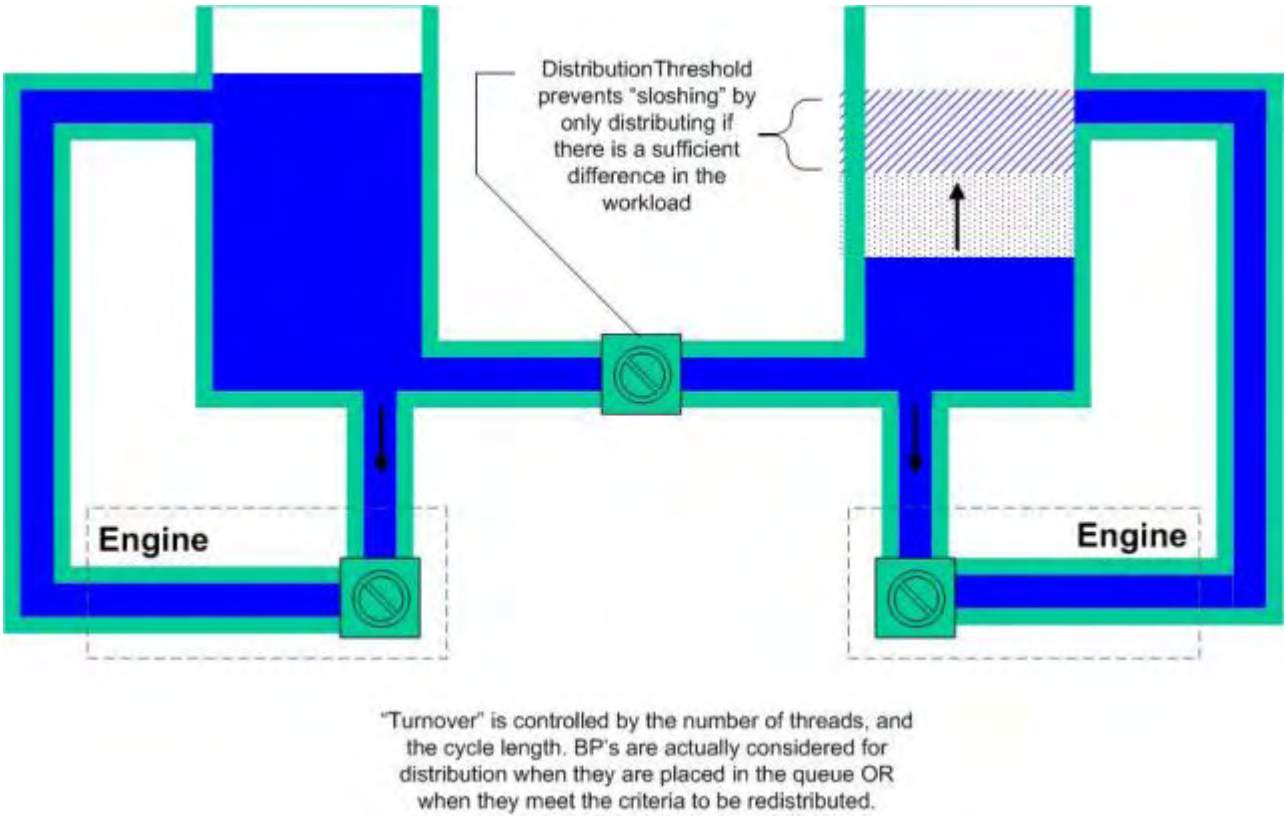
- Redundant network connections

  More than one connection from the server to the network is necessary. This includes the connections from the data center to the point of presence, where a communications service provider takes over. More than one connection must be leaving the premises, by widely separated routes.

- Redundant network paths

  Upstream network components are duplicated, including DNS (Domain Name System) servers, firewalls, HTTP servers, and any other components with which the application server communicates.

The following illustration shows both redundant network connections and end-to-end paths:



- Reliable server hardware

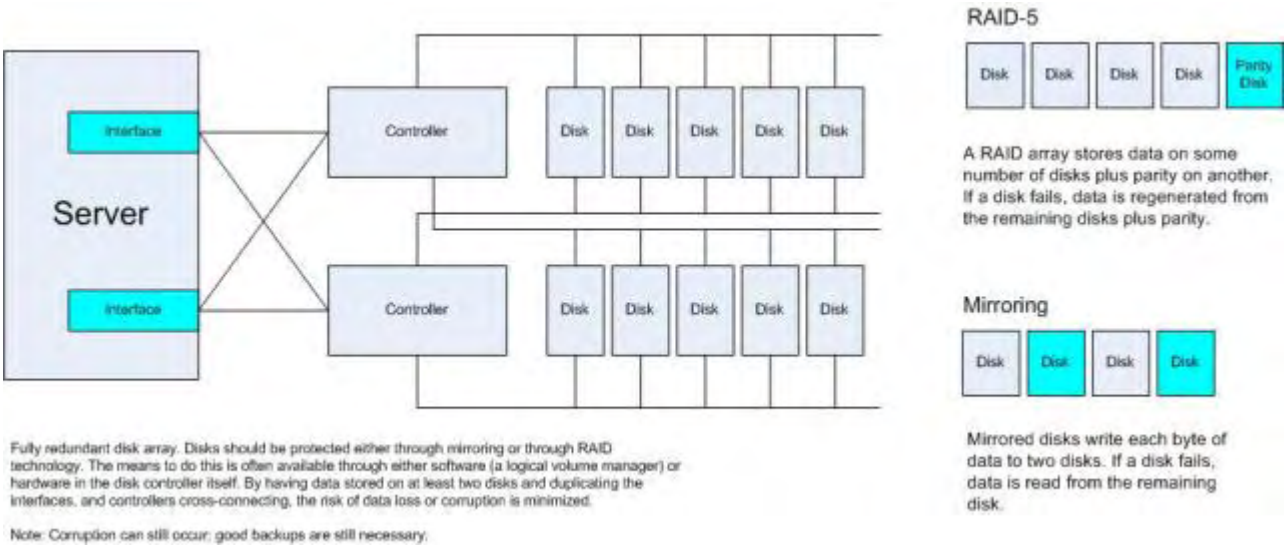
  Reliable server hardware includes error detecting/correcting memory, like ECC or SECDED, and hot swappable components. Because the basic components of servers are very reliable, the hardware prevents data corruption by diagnosing internal failures and allowing repair without interrupting operation. This increases the reliability of the entire environment, because these kinds of errors are not easily detectable by other means.

- Mirrored or RAID disk storage

  Disk storage must be either mirrored or Redundant Array of Inexpensive Disks (RAID), including any temporary storage that is used. The risk of data loss or corruption is minimized by having data stored on at least two disks and duplicating the interface, and by controllers cross-connecting.

  Mirrored disks write each byte of data to two disks. If one disk fails, data is read from the other disk. A RAID array stores data on some number of disks plus parity on another. If a disk fails, data is regenerated from the remaining disks plus parity.

  The following illustration shows the mirrored and RAID techniques:



- High availability database server

  A database server environment must be configured for high availability, either through clustering or failover. This is a total database vendor-based solution for load balancing and/or failover. For the application, any JDBC database driver that already supports transparent failover should work, but might require a small amount of customization. If database redundancy is achieved through failover (including IP address takeover), then the application rolls back any transactions during the time the database is unavailable.

- High availability auxiliary components

  If other facilities are being incorporated into the application environment, like middleware, ERP systems, or web servers, these additional components must be included in the high availability evaluation and design. If these auxiliary components fail, a substantial amount of the business functionality for the application environment might be unavailable, even if the application is fully functional.

- Reliable power

  Reliable power is required, ideally with full redundancy all the way from the power substations, through electrical panels, to the components themselves. Un-interruptible power supplies (UPS) and/or generators are also necessary. Single points of failure and outages in this area, such as cable cuts, often require hours to remedy.

- Technical and procedural methods

  Business process developers must account for possible failure modes, both technical and procedural. You should focus on detecting problems and ensuring that correct and timely notification occurs. Not every kind of failure can be handled through technical solutions. Providing for human intervention in troubleshooting and correcting high-level process problems can be the critical factor in deploying a truly reliable process.

- Monitoring

  The application detects certain kinds of system errors, but it is vitally important that you monitor the network, participating servers, and database environment to provide detailed diagnostic information and detect and locate failures quickly in any domain. This infrastructure is outside the scope of the application. Although the application is not packaged with components to integrate specifically with such environments, it is generally possible to use standard application functionality (OPS Command Line, HTTP, FTP, file system or database adapters).

Many monitoring packages are available, including some from network or server vendors and others from third parties. IBM, HP, SUN, Computer Associates and many others provide monitoring products. The open source products Nagios and Big Brother also are widely used.

At a higher level, the application provides an alert mechanism that provides notification in the event of a business process failure. On-fault processing also is available to take corrective action (possibly different action) for each individual business process. These mechanisms can be integrated with other monitoring or help desk-type software that is already in use.

- Documentation and automation

From an operational standpoint, documentation and automation are the two biggest improvements that can be made to overall system reliability. Documentation dramatically reduces the risk of a staff not understanding what to do, and automation helps ensure that complex procedures are executed consistently and reliably. Automation is unique for each installation and is probably the most important element of any reliable business process.

Availability of IFC Clustered Installations

Availability and reliability are often confused. Reliability refers to the performance of the application, while availability refers to being able to use the application. Making an application highly available (HA) means reducing the amount of unscheduled and scheduled downtime.

Before beginning to implement an HA environment, you must determine the requirements. HA features can be extremely complex and expensive to implement and operate, so they must be chosen wisely. Some of the key questions to ask are:

- What is the real cost of downtime?
- Does the length of the individual incidents make a difference?
- Are there times when it is acceptable to be down (such as for maintenance)?
- Does the system always have to be available at full capacity?

The first steps toward high availability involve improving reliability. Once the environment is reliable, it can be made highly available using a variety of different technologies. A large percentage of outages are not caused by hardware or system-level failures. You must expend significant effort to streamline operations, particularly recovery (database recovery, server recovery, application recovery) and make sure that operations are tested and well documented. You also must ensure that planned outages are coordinated across the environment. Some portions of the infrastructure used by the application are likely to be shared by other systems, possibly with widely different availability requirements.

The simplest approach to availability (conceptually, though not necessarily to implement) is to use more than one server (with all of the redundancy covered previously in this document), which is called “failover.” With this method, any time one server fails; the other server simply takes over.

Scalability of IFC Clustered Installations

Scalability refers to the ability of the application to handle changes in volume through the solution of technical and operational issues. Clustering the database, the file system, web servers, and the application can provide

significant scalability while improving availability and reliability. Small clusters are fairly easy to implement and can provide a better solution than using a larger server.

You should pay careful attention to database performance in a clustered environment. With two nodes, generally twice as many business processes are executing at the same time, so the database and all of the other components must be scaled to match. Adding a node (or even upgrading the application server platform to have more processors) increases the load across the environment.

The overall performance of the application in a clustered environment is limited by four factors:

- The granularity of the work being done
- The arrangement of the adapters
- The performance of the database engine
- Interaction between processes (for example, locks and synchronization)

You can improve the scalability of a business process by doing the following:

- Changing the amount of data processed in each business process.
- Changing the amount of work done in each step.
- Changing the division of labor between business processes.
- Avoiding locking or synchronization between processes.

However, some business processes will not scale well. One example of this is a process in which a single row in the database must be updated by each instance of the business process. This constraint will force the processes to run sequentially, at least while they are updating the row.

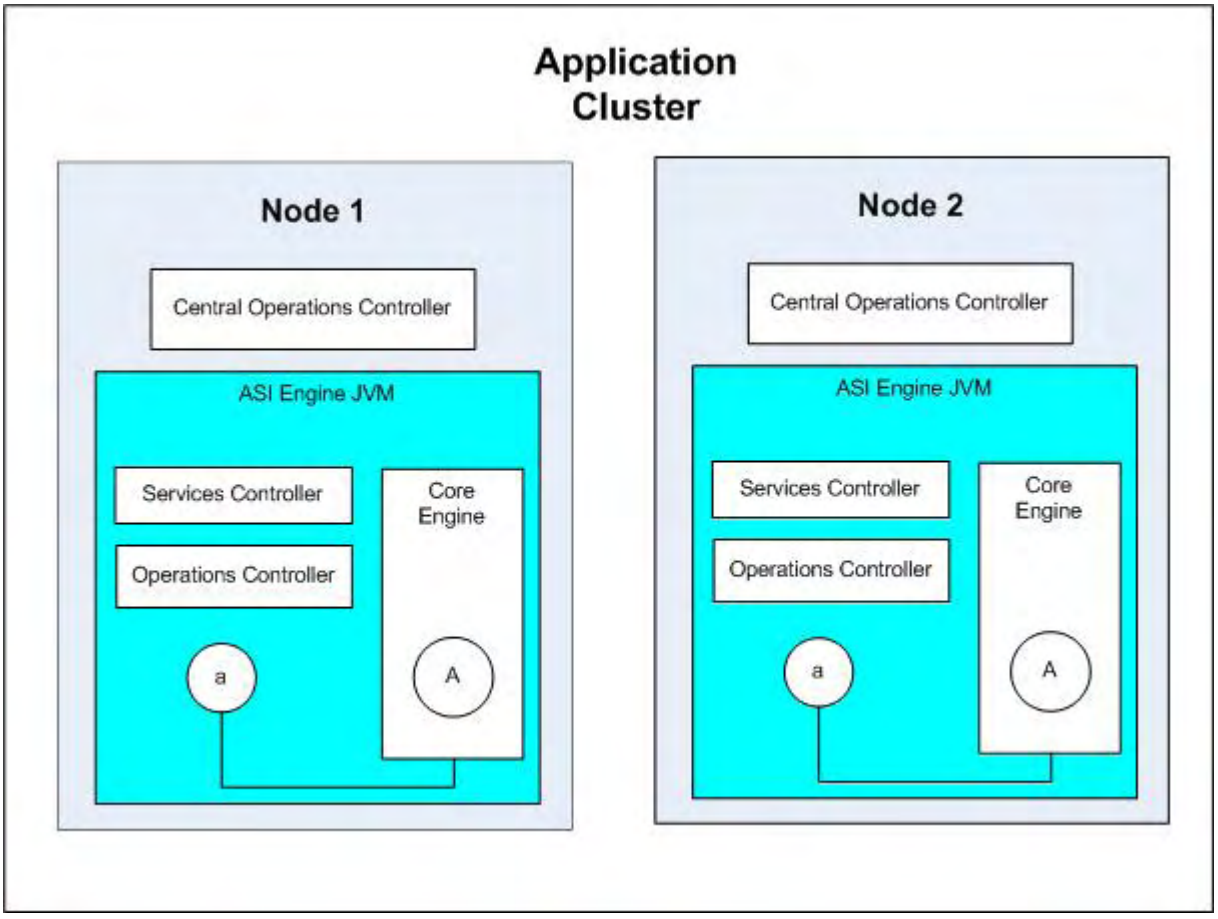
Clustering Architecture

IFC Cluster Architecture

Clustered installations of the application use the different components of an application node (that is, one of the cluster installations):

- Core engine components, which execute business processes.
- Services and adapters, which execute business process steps and communicate with external systems such as databases, ERP (enterprise resource planning) and CRM (customer relationship management) systems, and other technologies and packages.
- The Operations Controller, which manages resources across Java Virtual Machine (JVM) boundaries.
- The Services Controller, which provides the mechanisms to manage, configure, query, and cache all service/adapter-related information.
- The Central Operations Controller, which provides a single point of contact for all operational questions and communicates with the Operations Controller to coordinate component operations.

The following diagram illustrates a basic cluster configuration for the application. “ASI” stands for Application Server Independent. The instances of lowercase and uppercase letter “A” refer to different parts of adapters.



Resource Sharing in an IFC Clustered Installation

ASI clustering locates resources throughout the cluster using the Java Naming Directory Interface (JNDI), which is a standard technology that enables configuration information to be shared locally and across a network. These resources include the application servers, adapters and many other components. Each node maintains a JNDI server for locating resources. When a node starts up, it registers by communicating its status to the database so that other nodes can find information about this node. Database tables store locations of slowly changing resources like adapters. The JNDI directory is refreshed from these tables. Problems in resource sharing, such as failed RMI (Remote Method Invocation) calls, are almost always due to either misconfigured ports or misconfigured adapters.

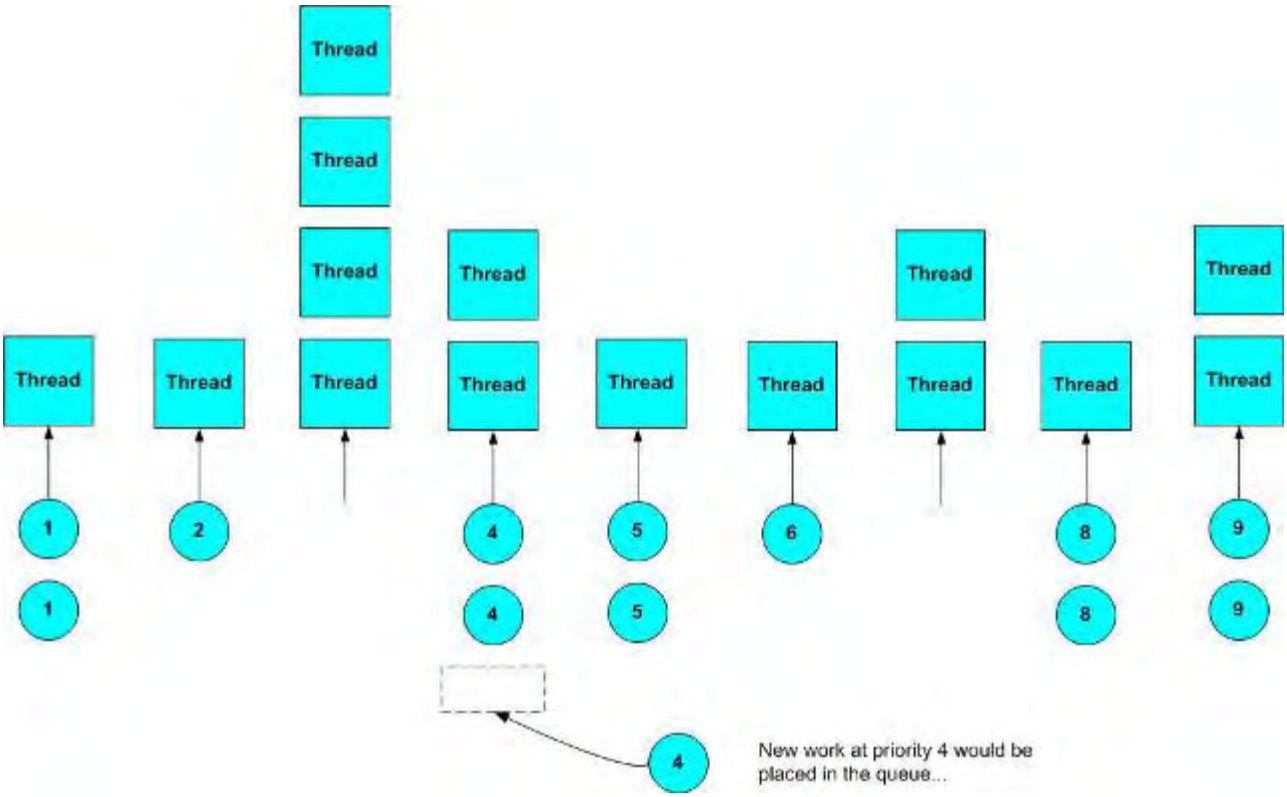
ASI Engine and Workflow Queuing in an IFC Clustered Installation

In a cluster, performance is affected by the number of threads allocated for each queue on each cluster node. Load balancing is dependent on the following factors:

- The number of threads.
- The number of steps for a business process to execute before being rescheduled and then possibly distributed to another node.

For most servers, the optimum number of threads is a relatively small multiple of the number of CPUs. At any time, though, a heavily loaded application server is processing far more tasks than the number of allowed processes. These additional tasks are kept on a queue. Each task executes a configurable number of steps and is returned to the queue. There is a separate queue for each priority (1-9). Each queue has its own resources assigned and managed through a scheduling policy (there is only one policy active at a time). This enables the management of mixed workloads.

The following graphic shows an example of this process:



The application currently supports the following two scheduling policies:

- The Basic Scheduling Policy only allows static control of the number of threads allocated to each queue.
- The Fair Share Scheduling Policy is a model in which each participant (queue) is assigned a certain share of the available resources. Over time, this policy ensures that each queue receives its share of resources.

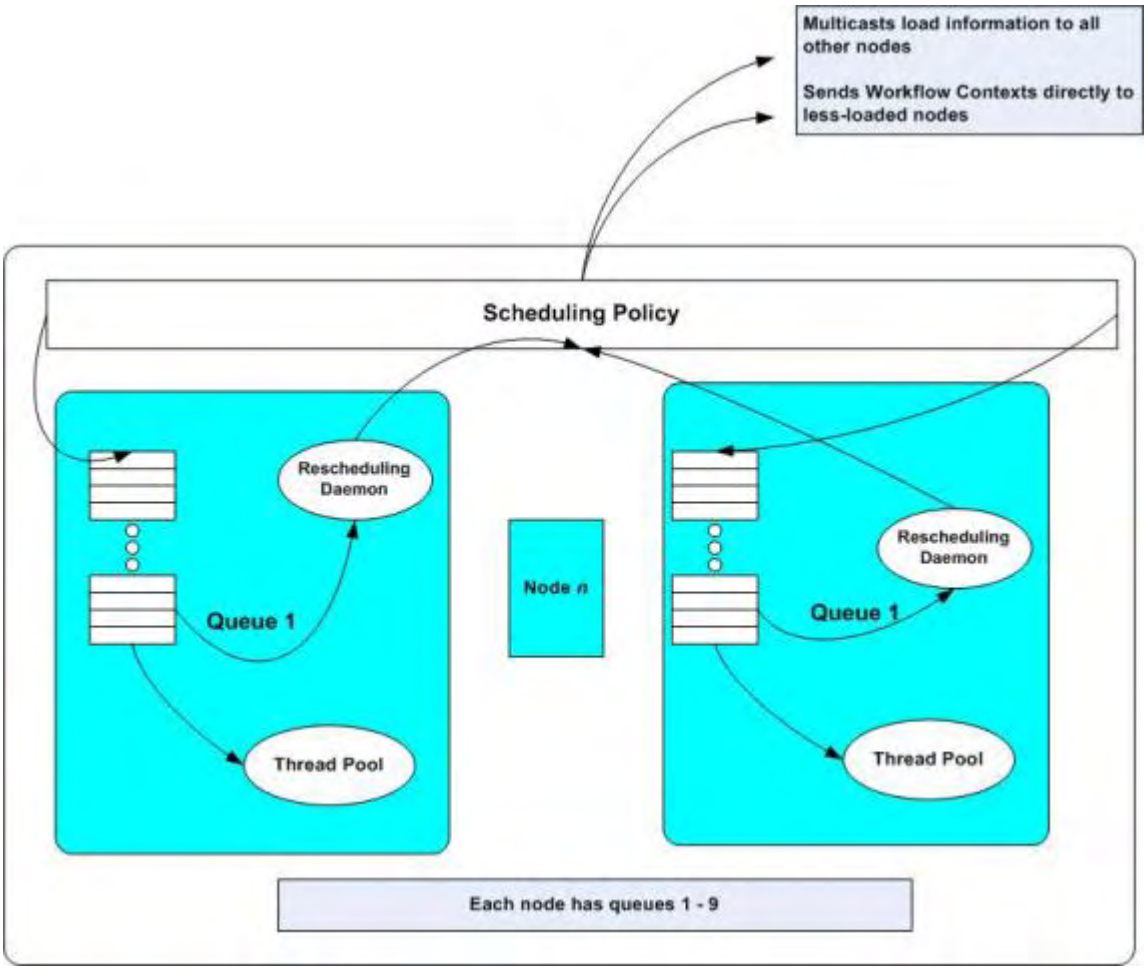
To conserve resources, business processes can be cached to disk while in the queue. There are several parameters that control exactly when and how business processes are cached, allowing memory resources to be used efficiently. A distinction is made between smaller and larger contexts, with the definition of “small” being a parameter.

Load Balancing in an IFC Clustered Installation

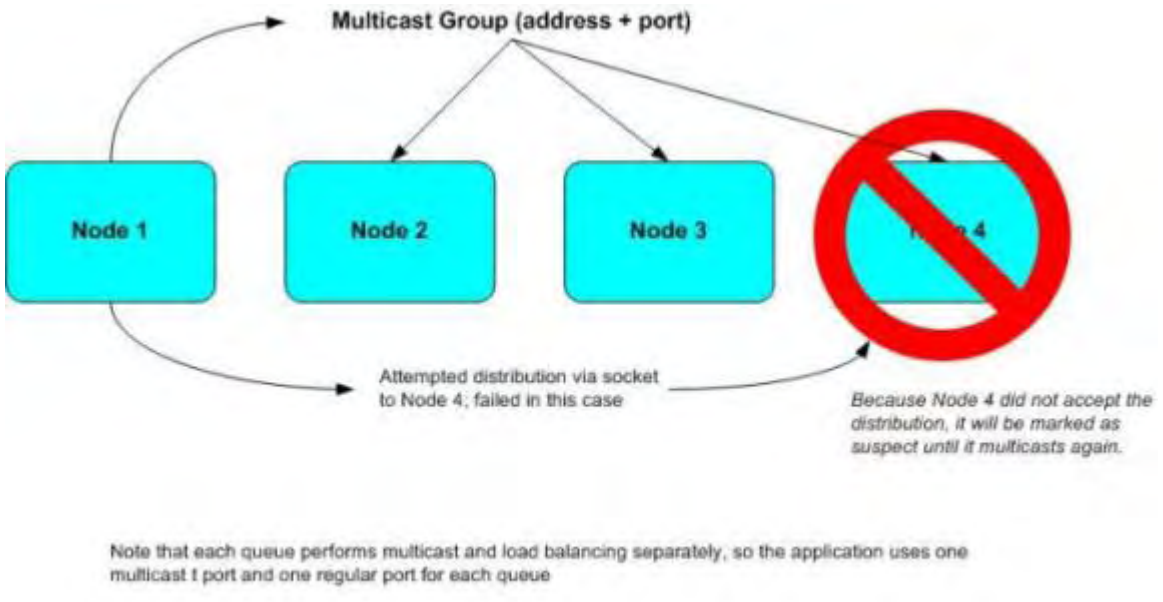
Application clusters distribute business processes internally using a load-balancing algorithm determined by the scheduling policy. In addition, external load balancing mechanisms distribute incoming HTTP, FTP or other network traffic. When a business process is placed in the queue, the relative loading of the servers participating in a cluster determines if the work should be distributed or executed locally.

Each node notifies the other nodes in a cluster that it is available and what its current load is through a multicast mechanism that is a network technology between communicating point-to-point and broadcasting to the whole network. The broadcasted load factor is derived from the average time that business processes in a particular queue are executing before relinquishing the CPU and the number of business processes waiting in the queue. The distribution algorithm is based on the load factor of each node and configuration parameter to determine on which node the job is going to be executed. This describes how quickly a business process would run after it is sent from one node to another node.

The following graphic shows how this process works:



The algorithm that decides whether to distribute work is determined by active scheduling policy. A configurable threshold percentage determines when to distribute work to a node that is more lightly loaded than the current node. If there is more than one, it chooses the one with the lowest load factor. If a node fails to respond to an attempted distribution of work, it will not receive any more work until it communicates to the cluster that it is functional. If an attempt at distribution fails, the context is placed back in the queue to execute locally and the target node is marked as being “suspect” until it asserts to the cluster that it is functional.



Note that each queue performs multicast and load balancing separately, so the application uses one multicast t port and one regular port for each queue

Alternative Load Balancing in an IFC Clustered Installation

Two other load balancing alternatives exist, in addition to the default load balancing method, which is based on queue depth. The queue depth load balancing method considers distribution to other nodes only when all the active threads in the specific queue of the current node are completely used up.

The following alternatives provide a way to configure load balancing based on the specific requirements of the customer:

- Load balancing with active threads. This is configurable per queue level and enables load balancing before all the threads are used up in a node and queuing starts.
- Business process cost/weight-based load balancing. If this load balancing method is used, the queue depth method of load balancing cannot be used.

These alternatives are described in detail in the following sections.

About Active Threads Load Balancing Method

The active thread method of load balancing starts distributing before all active threads are completely used up. This might be required when there are CPU-intensive business processes that are few in number but each take considerable resources being kicked off from the same node. In the queue depth method of load balancing, all of these business processes will execute on the same node before all threads are used up, causing one node to have 100 percent CPU utilization while other nodes are still idle.

To enable distribution within active threads, use the FindActiveThreads.x property in the noapp.properties.in property file. x refers to the queue number. This can be configured on a per queue basis. This load balancing method depends on both queue depth and the active threads per queue in the system.

To test the active threads method of load balancing, follow this procedure:

1. Set up two node clusters.

2. Set FindActiveThreads.4 = true. This enables active thread level load balancing for queue 4 from node1.
3. Start a long running business process (such as SleepService) that uses slightly less than the maximum threads configured for queue 4 from node1.
4. On the thread monitor, look for threads running on node1 and node2 and you should see some of them distributed to node2 even before all threads in node1 are used up.

About the Business Process Cost/Weight Load Balancing Method

In the business process cost/weight method of load balancing, each business process is given a weight in a range of 1 to 9 (the default is 3). Work distribution is based on the weight of all business processes in a specific queue (active threads plus queued). When the load on a queue is greater than a specified percentage, then work is distributed to the queue that has the least load factor.

Note: You cannot use both distribution based on weight and default load balancing based on queue depth.

To configure the business process cost/weight method of load balancing, set these property values in the noapp.properties.in file:

• DistributionOnWeight = true
• DistributionThreshold = Percentage for the distribution decision

When DistributionOnWeight = true, multicast communications broadcasts the weight for a node instead of the queue depth. The weight for a node is the sum of all business process weights in active threads and in the queue. The distribution decision is based on the percentage from the following formula:

(current node weight - the minimum weight) / current node weight

Multicast Workload Communications in an IFC Clustered Installation

Communication among cluster nodes about the workload of each node helps provide business process load balancing. You have a choice between multicast communication (which services all requests with a single stream of data) and unicast communication (which services each request with its own stream of data).

Multicast is used by an ASI cluster node to make information about its workload available to the rest of the cluster. It functions similarly to a broadcast, except that the message is only delivered to nodes that have registered to receive messages on a particular address and port. Multicast requires the participation of a routing component (where multicast needs to be turned on). Most routers that support multicast also support controlling the scope of multicast by restricting it to a single segment or a particular subset of the network. This is an important element of security and can also be a potential source of problems if configured incorrectly.

The application ASI configures multicast (and other network services) based on the application base port, to simplify installation. Because of the way it is assigned, it is unlikely to be used by another system, but if it is, the addresses can be changed in the application properties files. Two cluster nodes may use different port ranges for the system (non-multicast) base port and all other network settings derived from this, but they must share the same multicast base port setting for clustering to work properly.

Also, for multicast in an application cluster to work effectively, it is critical that all nodes in the cluster be in the same subnet.

A cluster will broadcast packets that will be received by any other cluster that:

• Uses the same multicast port range
• Is reachable via multicast on the network

If the two clusters are connected to the same database instance, they are treated as a single cluster. If they are connected to a different database instance, each will reject the other's packets but produce many exceptions in the log. This occurs if an exact replica of the production environment is required for testing or development. If this is required, the replicas must be on different network segments and the routers must be configured to restrict multicast to a single segment, or the clusters must use different multicast ports.

Cluster multicast and distribution communication occurs at each node rather than at each queue in a node. This prevents the opening of separate socket connections for each queue in each node, which opens a minimum of 20 socket connections per node for cluster communication. If you need a large number of cluster nodes per machine, working at each queue might cause problems because of the large number of sockets connections and the large number of ports.

One port and socket is used for multicast communication across all queues in a node. Similarly, for the distribution of workflows, one port and socket per node is used across all queues. The cluster distributes the information to specific queues within a node.

To disable multicasting, set the disableMCast property in the noapp.properties file to true.

JGroups Workload Communications in an IFC Clustered Installation

Communication among cluster nodes about the workload of each node helps provide business process load balancing. You have a choice between multicast communication (which services all requests with a single stream of data) and unicast communication (which services each request with its own stream of data).

This method can use either IP multicast communication or TCP unicast and multicast communication. The unicast method lets one node communicate directly with another node, without the communication being distributed across more than one node (as with multicast).

You can use TCP when IP multicast is restricted because of security reasons or when IP multicast is a bottleneck to providing cluster support across a WAN for geographically distributed clustering.

The JGroups communication toolkit enables you to configure the following protocols during application installation and setup:

- UDP (IP multicast)
- TCP unicast and multicast (does multiple unicast)

You can choose between UDP and TCP during configuration.

JGroups is used in cluster communication for broadcasting the business process load factor across all nodes in a cluster. It also can be used for workflow context distribution for load balancing. JGroups provides for reliable communication and group membership management, including notification and handling of new memberships, departing members, and failure detection and broadcasts. The underlying protocol can be switched during installation and configuration without compromising on the features of group membership management and communication. It also provides tunnelling over the firewall, which expands the cluster communication support for clusters on geographically distributed networks.

The JGroups protocol property settings (like type of protocol and ping interval) are managed in the jgroups_cluster.properties file. Multiple JGroups channels can be configured and managed in the application by supporting multiple protocols.

The default protocol set for JGroups for business process load balancing information is IP multicast. Workflow context distribution is TCP unicast.

For node to node communications, the properties are defined in jgroups_cluster.properties. The attributes used to define communications are:

- property_string - default value is UDP
- distribution_property_string - default value is TCP. This attribute should never be set to UDP. If you want to change the communication for cluster multicast from TCP to UDP, contact Sterling Commerce Support.

In addition, if you are using TCP for both property_string and distribution_property_string, the initial_hosts list for TCPPING should contain all hosts in the cluster.

About Operations Controllers in an IFC Clustered Installation

Each application server component has an operations controller, which provides local and remote interfaces for controlling a server component and inquiring about its status. Using JNDI, server components on the same node or different nodes can directly communicate to all operations controllers. The application provides a command line tool for running operations commands for automating the operations.

Role of the Central Operations Controller in an IFC Clustered Installation

The Central Operations Controller provides a single point of contact for all cluster operations, and communicates with the operational controller on each cluster node VM to coordinate operations. The Central Operations Controller runs on a separate JVM. There is one Central Operations Controller for each cluster node instance.

During application startup, the Central Operations Controller coordinates the operations of the local cluster node startup and its corresponding components, such as the Service Controller and the Scheduler.

Features of the Central Operations Controller include:

- A heartbeat mechanism that provides support for central operations failover and node status update.
- Token nodes, which allow individual nodes to assume the role of being the Central Operations Controller.

Central Operations Controller Heartbeat Mechanism

The heartbeat thread interrogates the cluster nodes to obtain status updates and identify new members. It runs in the same JVM as the Central Operations Controller. It is started inside the Central Operations Controller when it registers itself as part of the startup procedure. It updates its local cache of node status that is used for communicating with cluster nodes.

When the cluster node, along with its Central Operations Controller, is fully up and running, the heartbeat thread detects the presence of other nodes and registers the identity and status of other cluster nodes to the Central Operations Controller. From that point on, each Central Operations Controller connects to all the nodes in the cluster and is ready to service requests.

Central Operations Controller Token Node

Each node running a Central Operations Controller can assume the role of being the Central Operations Controller at any time. This is managed by a token. The ownership and passing of the token are mediated through the database.

One of the Central Operations Controllers in the cluster holds a token that identifies that controller as the master Central Operations Controller. The Central Operations Controller with the token is responsible for coordinating recovery if a node fails. The Central Operations Controller that possesses the token when it detects a node failure does the following:

1. Releases locks placed by the failed node.
2. Initiates automatic scheduler jobs failover. That is, it identifies another active node that can take the jobs from a failed node.

If the failed node is the token-possessing Central Operations Controller, the Central Operations Controller on an active node takes over the token and performs the operations of a token node. This operation ensures that only a single node holds the token at one time.

This arrangement provides for automated failover of the Central Operations Controller and enables operations to be distributed across the cluster.

Specifying Mandatory or Preferred Nodes for BP Steps in an IFC Clustered Installation

Each service step in a business process can be configured to be executed on a particular node. This feature keeps business process steps local to a non-clusterable adapter and moves a business process to the correct node to access a document stored on disk.

You can force execution to one of the following kinds of nodes:

- A mandatory node. A business process fails if the mandatory node is not available.
- A preferred node. A step executes on the preferred node if it is available. If the preferred node is not available, the step executes in the local node where the step is currently running.

You can specify mandatory and preferred nodes at the business process level. This feature allows users to specify that a particular business process should run on a particular node in a cluster. If the node is down, the business process fails. All steps of the business process run on the specified mandatory node.

A business process-level preferred node tries to run the business process on the specified node. If the node is down, it runs on one of the active nodes available on the cluster.

Use the Process Level page in the business process checkin wizard to specify or update business process-level mandatory/preferred nodes during the check-in of a business process.

Specifying Nodes as Execution Roles for BPs in an IFC Clustered Installation

You also can target a business process to a group of nodes configured to assume a role called Abstract Execution role, which is a group of application nodes in a cluster that are assigned to do a specific job/role. When a business process is configured to run on an execution role, the business process will be executed only on nodes assigned to do the specific execution role.

For example, you can specify multiple nodes with the same role, so that the business processes with the same role are distributed among the nodes. You can define node 1, node 5, and node 9 with an execution role like “POProcess” and configure a business process to execute within this execution role. Then, when you run the business process, it could run on node 1, node 5, or node 9. This feature gives you the control to specify on which node or nodes the business processes can be run in the cluster environment.

Use the Process Level page in the business process checkin wizard to specify an Execution role for the business process during the check-in of a business process.

A business process can be set to use either a business process-level mandatory node or an execution role, but not both at the same time. An execution role with just one node is equivalent to setting a mandatory node at the business process level.

When you set a business process to use an execution role, a node within a specific execution role is selected, based on the following criteria:

- Load factor-based distribution
- Healthiness of the node

After a node within that execution role is identified, the business process is distributed to that node and it executes on that node.

To set up a role for a node, run the following command from the installation directory:

Caution: You can run this command any time after the application starts, but if you change the role of a node, the business process that uses the old role will not work any more.

Note: You cannot set up a role for a node from the user interface.

- (UNIX/Linux) `bin/opscmd.sh -cUPDATESERVER -n<node_name> -pID=<node_role>`

  For example: `bin/opscmd.sh -cUPDATESERVER -nnode1 -pID=node1role`

- (Windows) `bin\opscmd.cmd -cUPDATESERVER -n<node_name> -pID=<node_role>`

Business Process Recovery in an IFC Clustered Installation

When a cluster node fails, the application reacts like it would when a single node environment goes down in a non-clustered implementation. The application tracks and persists all mid-stream activities and process states within the system, and retains the details of the business processes and services. Any active transaction on a failed node halts.

The default recovery operation marks any business processes running on the failed node as interrupted and reverts to the last edge (the state at the last completed step) of each process for recovery. You can manually resume a business process at that point on another node or restart a business process on another node. This default operation can be changed, but changing the default operation is not recommended.

Document Storage in an IFC Clustered Installation

The application provides the option of using document storage either in a local file system or a database. In cluster mode, the default for document storage is the database since all nodes use the same database, and a business process running on any node has access to the document for processing.

To enable usage of a file system as document storage in cluster, all nodes must have access to the file system where the document is stored and thus need to use an NFS (network file system)-mounted or clustered file system to store the documents, which enables all the nodes to have access to the document.

The mandatory node feature provides the ability to mark a specific step of a business process to be run on a specific node by marking the node where it needs to run. This provides the option to use file system-based document storage (without clustered or NFS-mounted file system) where the line of sight of the document is

just to one node. However, if the node where the documents are targeted fails, the business process fails and the business process needs to be re-configured to use a different node. Depending on the business process and the usage of the document, some of the failed business processes can be restarted or resumed, but others cannot unless the node is brought back up.

Special Considerations for Clustered Environments

Non-Clusterable Adapters in an IFC Clustered Installation

In a cluster, the connection of adapters to external resources makes some of those adapters deployable only to a certain node in a cluster. Adapters that can be deployed to all cluster nodes can share the load across all cluster nodes, facilitating scalability and load balancing.

Some adapters are non-clusterable and tied to specific nodes for the following reasons:

- Resource affinity to specific nodes.

  For example, the File System adapter needs to be deployed on a specific node where it needs access to the local file system, where it collects and extracts files.

- Connectivity restrictions to external resources.

  For example, with the Connect:Direct and Connect:Enterprise servers, the connection parameters are configured for a specific connection and used only by a single adapter, thus forcing it to be deployed on a specific node.

  All adapters that use perimeter servers come under this category.

- Licensing restrictions.

  There might be a limit on the number of instances that can be deployed in an installation.

  For example, the SAP and WebMethods Enterprise adapters might not be fully clustered across all server instances because of the licensing agreement.

The most commonly used adapters that require special consideration under a clustered environment include the following:

- File System adapter
  - The file system must be mounted on a common drive, which can be accessed from all the nodes.
  - Use a clustered file system.
- Connect:Direct and Connect:Enterprise

The Connect:Direct Server adapter and Connect:Direct Requester adapter establish sessions with Connect:Direct. The Connect:Enterprise adapter establishes a session with Connect:Enterprise.

Neither of these sessions are tied to a resource. Neither the Connect:Direct nor the Connect:Enterprise adapters are cluster-enabled.

- Command Line2 adapter

  The commands executed by this service are supported and valid on all of the nodes.

Also, all adapters that use perimeter services are not clusterable and are deployed to a specific node attached to the perimeter server to which they are connected.When configuring these services, choose either node1 or node2 in the environment section.

However, these kinds of adapters can be grouped together using service groups and accessed as a single entity. These adapters include:

- FTP Server adapter
- HTTP Server adapter
- SFTP Server adapter
- FTP Client adapter
- HTTP Client adapter
- SFTP Client adapter
- Connect:Direct adapter
- Connect:Enterprise adapter
- B2B Communication adapter

The following table classifies adapters to be either clusterable or non-clusterable. It also provides ways to configure for certain adapters which by default are not clusterable but can be configured differently to be deployed on two or more nodes in cluster to support failover.

Clusterable by Default	Clusterable - Service Groups, etc.
BP Fault Log	Command Line2 Adapter
Gentran Server NT Adapter	File System Adapter
JMS Adapter	All adapters that use Perimeter Services
Oracle AQ Adapter	
Adapter for PeopleSoft	
Tuxedo Adapter	
WebSphere MQ Adapter	
WebSphere MQ Suite	
E5	
IM Adapter (XMPP)	
IWAY	
JCA	
LDAP	
MSMQ	

Clusterable by Default	Clusterable - Service Groups, etc.
Oracle EBusiness	
RMIIOP	
SNMP	
Adapter for PeopleSoft CRM	

Even if an adapter can only be installed on a single node in a cluster, it has cluster-wide visibility and business processes that use it can be distributed across the cluster. Business processes running on any node in a cluster can call the adapter. When a business process calls an adapter that is on more than one node, the process first calls the adapter that is on the same node as the process. If that adapter is not available, the process calls an adapter on another node.

If an adapter is deployed on only one node, and the node where the adapter is deployed goes down, the business process fails. To circumvent single point of failover situations for non-clusterable adapters, you should attain clusterability of adapters across more than one node on a case-by-case basis, depending on the nature of the adapter.

Configuring a Cluster with Non-Clusterable Adapters

In the case of resource affinity to specific nodes, such as the File System adapter, you should use a third party vendor solution on external resources like clustered/mounted file systems by which all nodes have access to this common clustered file system. The File System adapter can then be deployed on all nodes.

In cases of connectivity or external resource restrictions, a separate instance of an adapter can be deployed on each node connected to its external resource. They can be grouped together as a service group so a business process can view them as a single entity and thus can be made clusterable to support failover and load balancing.

The final method is to have a business process detect the failure of adapters and redirect the functionality to a different component on another active node. For example, if the File System adapter is installed on a node and it fails, you can use fault detection to write files using the FTP adapter installed on a different node.

In addition, business applications that adapters communicate with may not be suitable for clustering. Each node must have connectivity to the target system in these cases, to avoid single points of failure within the application. Moreover, if another technique is used to accomplish failover for the external business applications (for example, warm standby failover), application adapters must be configured to connect to the standby system if a fault is detected in attempting to connect to the primary system. This is managed through various mechanisms, as governed by the specifics of the system interactions and communication protocols.

To support configuration of adapters to be deployed on a single node, a few nodes, or all nodes in the cluster, the application provides support through the user interface for adapters to be targeted to specific nodes or all nodes in a cluster. This provides a facility to dynamically configure adapters to be deployed on one or more nodes. Also, on failover situations, if an adapter configured to a specific node fails and needs to be redeployed on another active node to continue processing, it can be achieved by targeting on an active node.

Service Groups in an IFC Clustered Installations

Load balancing and failover in a cluster can be helped by service groups, which are groups of services or adapters of the same type that can act as peers and share the load. More than one service instance of the same service/adapter type can be configured and placed within a service group. Thus, all adapters in a service group

are viewed as a single entity and a service group name (instead of the service instance name) is referenced in the business process.

In a cluster environment, the service group does a round robin of all adapters on a per node basis. Each node handles its own load balancing across all adapters belonging to the service group. If a service group detects failure of a specific adapter, it goes out to get a good one, thus providing not only load balancing but also failover.

Adapters that cannot be configured on every cluster node (because of resource constraints or connectivity to external systems) can be deployed one per node and placed on a service group. The business process can be configured to use a service group instead of a service instance, thus attaining clusterable adapters deployed on all nodes, facilitating failover and load balancing.

Use the **Deployment** menu to create a service group. Select **Services** > **Configuration** and then create a new service. For more information, refer to the documentation about creating service configurations.

Web Server Considerations in an IFC Clustered Installations

If the application HTTP Server adapter (which is used beneath several other adapter types) is being used in a clustered application server environment, software or hardware load balancing is required, even if the web server is not in a cluster. This setup enables incoming client requests to be directed to a particular node in the application server cluster.

Note: Sterling Commerce Professional Services can provide input on load balancing, but your client IT and network staff can determine the optimum approach for your environment.

There are several ways to perform load balancing when using the HTTP Server adapter, including these common techniques:

- DNS load balancing (round robin)
- Proxy servers
- Hardware load balancers

DNS Load Balancing (IP Method)

The simplest way to access an application server cluster from the web tier is by using a single Domain Name System (DNS) name that maps to the IP addresses of all of the clustered servers. When a DNS name is mapped to multiple addresses, the DNS server cycles through the list on each successive lookup of that name. This provides a simple form of load balancing and failover. Each time a client resolves the URL, it gets the next address in the cycle. This ensures that client connections are evenly balanced across the cluster. If a client request fails, the client can failover the request by looking up the name again and retrying with the new address. This is a simple and sufficient approach for some applications, but it does not provide the level of performance and manageability that other solutions can provide.

Proxy Servers (HTTP Method)

You can access an application server cluster from the web tier by using another web server that proxies back to the cluster. Some application servers also act as a web server to do this, or provide a plug-in to other web servers to handle this. For example, BEA WebLogic can be configured as the proxy server, or you can use Apache, Netscape, or Microsoft Web Server with the WebLogic plug-in as the proxy server.

The proxy server is set up to redirect certain types of requests to the servers behind them. For example, the proxy server can be configured to handle requests for static HTML pages and redirect requests for Servlets and Java Server Pages to an application cluster behind the proxy.

The proxy serves a function similar to the hardware load balancer in that the proxy server performs load balancing, distributing requests across the multiple servers in the cluster behind it. When a session is established, it continues to proxy all requests for that session to a single server. If that server fails, it fails over to a secondary server.

Hardware Load Balancers (IP Method)

Hardware load balancers avoid the drawbacks of the DNS approach by working at the IP level rather than at the naming level. Hardware load balancers tend to have higher performance than the software load balancers implemented by proxy servers. A hardware load balancer acts as a proxy to a cluster. The client connects to the load balancer and routes the connection to one of the clustered servers behind it. Load-balancing hardware can track the status and load of each server in its load-balancing decisions.

Advantages of using load-balancing hardware include:

- A wider choice of load-balancing algorithms.
- Fewer network hops.
- SSL (Secure Sockets Layer) acceleration (offloading SSL processing from the application server to the dedicated SSL accelerator).
- The skipping of dead servers, which improves response times.

Disadvantages of using load-balancing hardware include:

- They are normally more expensive than using a proxy server or DNS.
- They cannot take advantage of HTTP cookies that carry cluster-specific information about the location of the primary and secondary instances of the clustered objects. Session-aware load-balancing hardware is required to ensure that HTTP sessions are properly maintained (after the initial connection, additional packets that are part of that session go to the same server).

Using a Perimeter Server in an IFC Clustered Installation

In a clustered environment, each node must be configured with one or more perimeter servers, which is communication management software installed in a Demilitarized Zone (DMZ). The application uses the perimeter server to minimize firewall/DMZ issues, enhance scalability, efficiently handle large files, and improve performance.

Note: If a cluster node is not active, you cannot configure a perimeter server on that node.

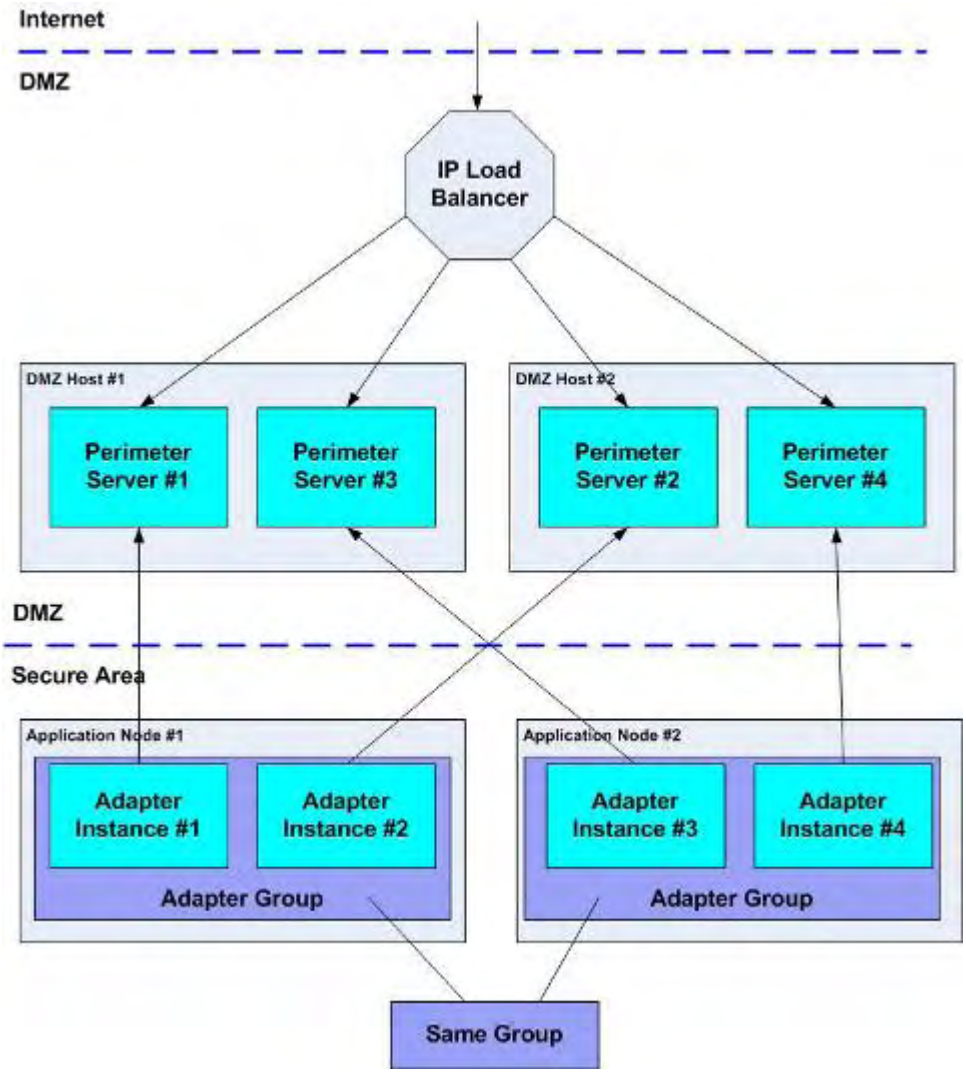
The perimeter server is tied to a node; adapters/services that use perimeter services (such as FTP, Connect Direct, and the HTTP adapter) must use service groups for load balancing and failover.

The perimeter server has a one-to-one relationship with the client. Because the perimeter server is tied to a particular node, adapters and services that are using the perimeter server also must be deployed in that particular node. Multiple adapters and services can use a single perimeter server.

Inbound Load Balancing and Failover Using a Third Party Vendor

A third party vendor such as Network Dispatcher can be employed for load balancing and failover.

The following figure shows a typical configuration for perimeter server for high availability:



This can be extended to more servers in the cluster or in the DMZ, with the following restrictions:

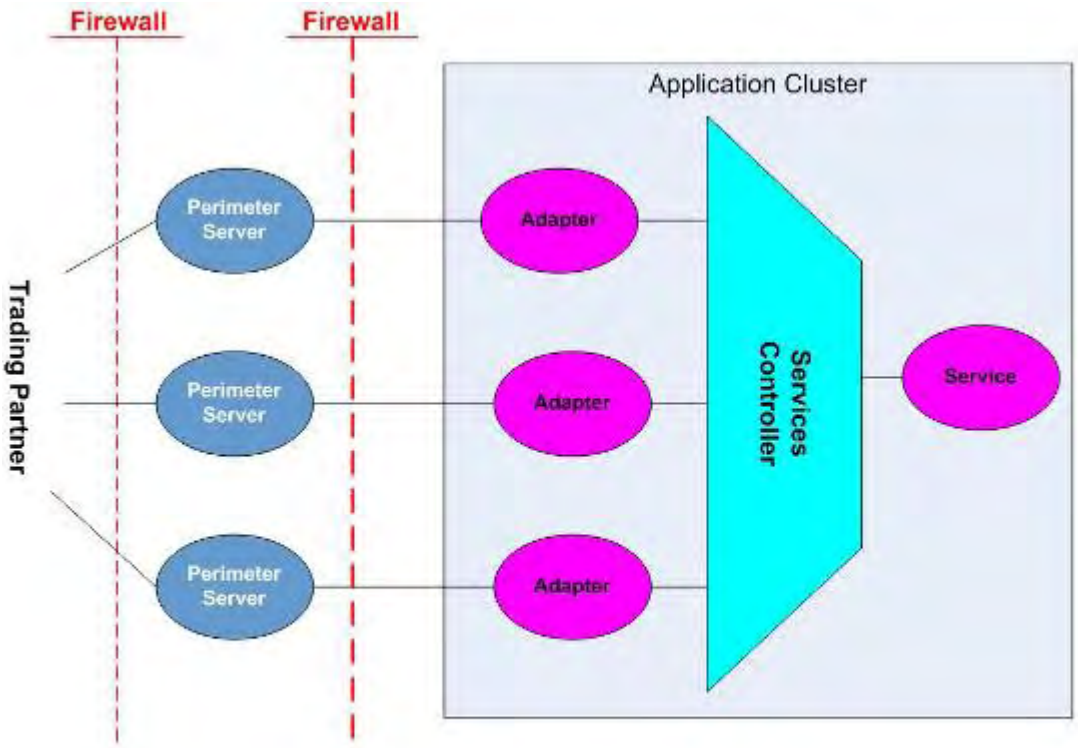
- All adapter instances must have compatible (nearly identical) configurations.
- For inbound (server) load-balancing and failover, either the DMZ hosts must support virtual IP addressing to sepa rate the listening port of the Perimeter servers, or the load balancer must do Port Address Translation (PAT).

Outbound Load Balancing with Service Controller

Outbound traffic is load-balanced by using Service Controller and Service Group.

For outbound (client) load balancing and failover, all of the adapter instances must be placed in the same group. Business processes must reference this group rather than a specific adapter instance.

The following figure demonstrates how load balancing works with a Service Controller:



Sticky Sessions (Persistence)

You can configure sticky session support if you are accessing Sterling Integrator through a load-balanced solution. It utilizes the stickiness based on the client IP address, which is the recommended option. Alternatively, you can also utilize session ID's, for example JSESSIONID or SSL Session ID, contained within service response headers, a less preferred option.

For more details about configuring persistence, refer to your load balancer documentation.

Performance Considerations for Clustered Environments

Performance Requirements for an IFC Clustered Installation

To determine the performance requirements for an application environment, you must understand the work to be done and the scheduling constraints. There are two distinct classes of business processes: without deadlines and with deadlines.

Capacity for processes without deadlines depends on the arrival rate and an appropriate margin. In a cluster environment, if you are relying on more than one node to meet the workload, and a node fails, the system will not meet the workload.

Processes with deadlines can be a lot more complicated. Usually, you must consider three scenarios:

- The process under normal circumstances.
- The process when everything is running late. How much lateness is tolerated?
- The recovery process, in which, in addition to the normal workload, a series of complex processes that failed due to a processing problem may be re-running.

In the recovery process, the system needs to meet deadlines for current work while reprocessing business processes that have failed. Determine what the largest jobs are and consider the workload imposed by running them outside their allotted schedules. If additional processing capacity is required, a node can be added to a running cluster environment.

Careful scheduling spreads workload throughout the time period (day or week or month) more effectively than if the system processes every incoming document immediately.

Capacity Planning for an IFC Clustered Installation

Capacity planning for the application includes the following specifications:

- Disk space for locally stored documents
- Disk space in the database (which is significantly affected by the archive policy):
  - For documents
  - For tracking information
  - For the persisting state of running business processes

- Database processing power
- Database connections (along with memory and other database resources)
- Application node memory
- Application node processing power
- Network bandwidth:
  - To and from the application adapters
  - To and from the database
  - Between cluster nodes
- Number of application nodes in the cluster

Scaling Vertically in an IFC Clustered Installation

Application cluster performance can be scaled just like single node performance, by improving performance with features like:

- More CPUs

  More CPUs allow more concurrent threads to run effectively, increasing the amount of concurrent work the application node can accomplish.

- Faster CPUs

  Faster CPUs speed up the execution of individual business processes, but have less effect on the amount of concurrent activity that the node can support.

- More memory

  Adding memory improves performance in every dimension, if parameters affecting caching and document persistence are adjusted accordingly. The number of threads a node can support is related to its processing power and its memory.

- More input/output capacity

  Increasing input/output capacity can be a huge improvement if the File System adapter is used heavily or if local document storage is turned on, causing poor adapter performance.

Scaling Horizontally in an IFC Clustered Installation

Application cluster performance can also be increased by adding nodes to the cluster.

Scaling in this fashion has the following advantages and disadvantages:

- Advantages
  - Extreme scalability when coupled with parallel database technology.
  - Nodes can be added to a running cluster without causing any kind of outage.
  - An increase in the level of redundancy in the environment.
  - More options for partitioning the adapter/input/output workload.
  - More cost effectiveness if the existing nodes are already fully configured and upgrading would involve replacing them.
  - High performance installations even using I/O-limited PC-based hardware.
- Disadvantages
  - The addition of nodes is always less efficient than enhancing the existing nodes because of cluster overhead.
  - Some additional management overhead for adding a machine to your environment

In the application, clusters can be expanded while running and new nodes do not have to be identical to existing hardware (although it is strongly recommended that they be as similar as possible). They do need to be running the same JVM versions.

For example, you could buy two nodes, about half configured with CPU, RAM and I/O cards. After about a year, you upgrade the boxes to their maximum configurations. Then you need still more capacity. Usually, by this time, the system vendor has moved on to new models. Rather than replacing your system, you just add a node using the vendor's new technology. Because nodes can be individually tuned with respect to threads and other performance parameters, the new node can be more or less powerful than the existing nodes. Once tuned, faster nodes automatically assume more of the workload.

Defining Service Level Objectives for an IFC Clustered Installation

Once the system processing power is known, you must determine the level of service that you want to achieve with your clustered installation. Factors in this decision include:

- Degree of fault tolerance and availability they require
- Normal workload requirements
- Exception/recovery processing requirements

If you are considering clustering for failover or high availability, the entire system must be examined to find single points of failure. For more information, see *Managing Single Points of Failure*.

There are multiple ways to handle failover. You must decide which option is best depending on the service levels you require. Failover can be accomplished by having redundancy built into some or all of the different layers of the environment, allowing one instance of a given system to take over for another upon failure. Address all of the following layers:

- Platform (including hardware, networks, file systems and services like DNS)
- Database
- Application (including the application server and its components)
- Interface (including web components)

It is recommended that the file systems be redundant to eliminate this as a single point of failure. It also is recommended that the database be clustered or mirrored to provide redundancy at this layer. Depending on the number of users accessing the web tier, this layer can also be clustered, or provide multiple web servers to provide load balancing and failover.

At the application tier, failover can be accomplished using multiple approaches. Clustering can be implemented, with a separate node on each of two different servers, or a standby strategy can be implemented and the backup activated when the primary system is unavailable. In certain circumstances, the standby server can be used by other applications during the time the primary system is available, and it would need to handle the additional workload of the main system only when the primary system fails.

Managing Single Points of Failure in an IFC Clustered Installation

Failover and clustering are not the same thing. In general, the application should be scalable, reliable and have minimal downtime. Clustering is one way to meet this objective.

To achieve high availability in a system, you must:

1. Consider potential failure points.
2. Establish a level of risk or possible rate of occurrence and recovery for each failure. Most components cannot be unavailable for very long.

3. Weigh this level of risk with the cost to eliminate the potential failure. This cost may be a redundant backup action to reduce the time to recover the loss. For example, if information stored in a database can be recovered by restoring from a backup eight hours old and then re-running any updates, this may be an acceptable risk compared to doing backups every two to four hours, or to the cost of having a mirrored image of the database.

A system without a single point of failure must consider all levels of the environment. If any level is unprotected, then the whole system is not protected. Consider all of the following levels:

1. System hardware
2. Database server
3. External applications/systems that the application is integrating
4. Application
5. Application server
6. Web server

To protect each level, that level would have to be replicated, and clustering affects the handling of these replications. This is often used not only to back up the primary component, but also to offload processing to the backup component while the primary component is still running.

While you must consider all of these issues, the Sterling Commerce services staff can provide expertise with configuring and setting up the application to avoid single points of failure for the application server and the application, with some general guidance on avoiding single points of failure in other areas. Even with this help, however, you must still provide expertise in single points of failure implementation for external applications, the database server, and system hardware.

Tuning and Configuring in an IFC Clustered Installation

Please refer to the Performance and Tuning Guide to tune engine, database and other settings. This section describes performance tuning related to application clustering.

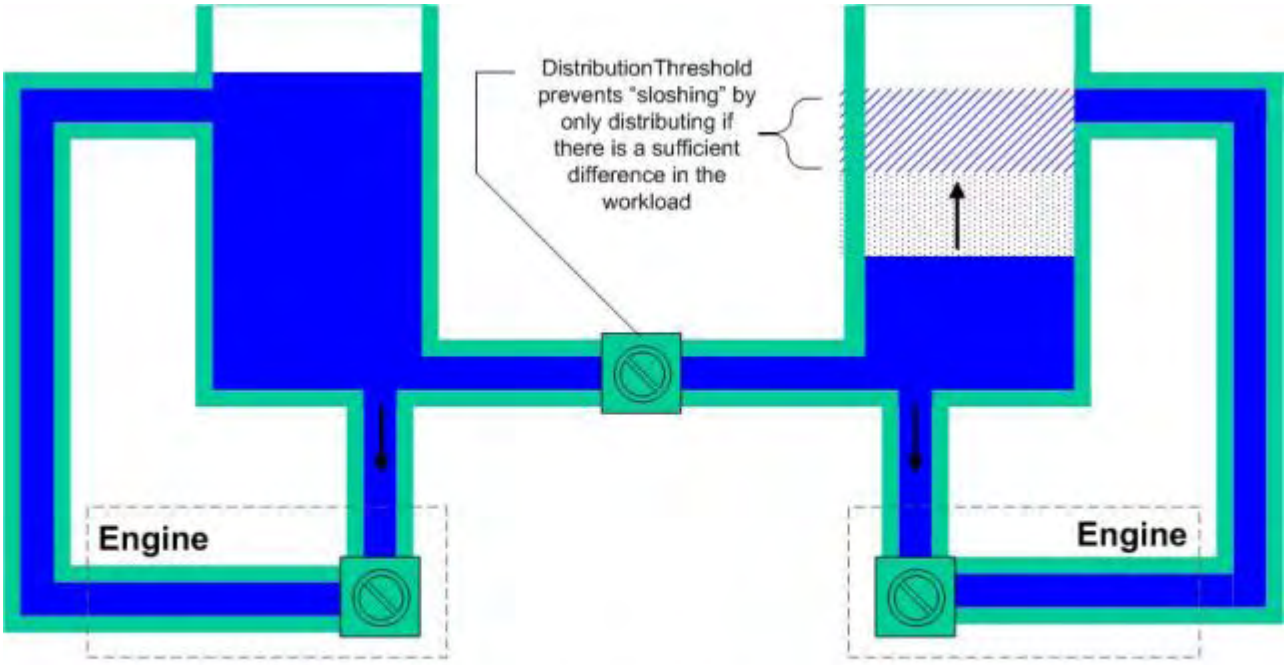
In a cluster, the primary factors that can be used to tune performance are:

- The location and configuration of adapters.
- The number of threads allocated for each queue on each node.
- The number of steps for a business process to execute before being rescheduled (and possibly distributed to another node).

The location and number of adapters is important because using adapters that cannot be clustered forces all activity for a particular adapter to go through one node. Failure to use an effective network load balancing technology in front of HTTP, FTP and other network-oriented adapters can also force activity to one node, because activity may be concentrated on one IP address.

The number of threads, used in combination with the execution cycle, places proportionately more workload from specific queues onto specific nodes.

An effective way to visualize distribution is to imagine that each queue on each node is a tank and the business processes in the queue are a liquid partially filling the tanks, as shown in the following graphic:



The following are the parameters that will directly impact distribution:

- DistributionThreshold

  The percentage load difference that must exist between two nodes before load balancing occurs. At 100 (percent), no load balancing will occur. At 1 (percent), the load will tend to move back and forth between the nodes with temporary imbalances. The correct number for this parameter is best determined by benchmarking with your workload, but generally, between 20 percent and 40 percent works well for many conditions. Other characteristics of your workload will have a much more significant impact on your scalability than tuning this parameter.

- Rescheduling

  Determines if business processes waiting in the queue will be rescheduled if they meet certain criteria. This gives them another opportunity to be distributed.

The length of the execution cycle (controlling the turnover rate), thread configurations, and the particular services and amount of data involved all have an impact on how smoothly work is distributed. For some workloads, the performance will be almost linear, approximately doubling the performance by doubling the number of nodes. For other workloads, there will be little improvement, especially if a single non-clusterable adapter is the bottleneck.

About Performance and Tuning for IFC Clustered Installations

Clustering involves these hardware sizing questions:

- What equipment satisfies processing requirements?
- What equipment satisfies service level agreements?

These questions relate to the two main benefits for clustering: load balancing and high availability. Multiple approaches can meet these objectives, and you need to understand the reasons behind considering clustering, and to evaluate alternatives to clustering to address these objectives. You should answer the following questions:

- What are the processing or throughput requirements?
- What are the service level or availability objectives?

You must answer both of these questions at the complete system level, and not just at the level of the application servers, since many factors can affect the final design.

Refer to *Tuning and Configuring* for details on application tuning for performance on cluster.

Clustering FAQs

In a clustered installation, which J2EE application servers and versions are supported?

The application does not directly run on any J2EE application servers. Instead, it sits beside and integrates with a J2EE application server.

Which clustering patterns does the application support?

- Active/active clusters, with all nodes processing work at all times (preferred).
- Active/passive clusters, with a second application/hardware node configured exactly like the first, using either automated or manual techniques to start the second application instance in the event that the first fails.
- Hybrid configurations in which an active/active cluster is backed up by a passive node that can replace either of the active nodes in the event of hardware failure.

The application also supports the dynamic addition of nodes to the cluster (although for the application, the installation of such nodes on a running cluster has several additional steps).

Who is responsible for configuring the clustering hardware/software?

There are many different clustering tiers and options. The customer is responsible for establishing a clustering plan based on their system objectives. Sterling Commerce can consult on these options, but the customer must have the necessary hardware/software expertise to cover their architecture. The primary focus of Sterling Commerce is the successful implementation of the application within this environment.

Do any of the recommended performance settings change for clustered installations?

Performance fine-tuning is still applicable to a clustered environment and those parameters must still be reviewed. Because clustering dramatically increases the options available for tuning the whole environment, it is likely that some settings will change.

How does an application cluster recover from a major environmental failure?

When a cluster node fails, the application reacts like it would when a single node environment goes down in a non-clustered implementation. The application tracks all mid-stream activities within the system and maintains the details of the business processes and services. Recovery business processes define how quickly to recover and the application server automatically assigns the business processes to a new node. The system can perform automatic recovery or can be configured to have any business processes running on that node fail as interrupted and go back to the last edge of the process for recovery. The application also includes the ability to resume and restart multiple business processes at one time.

In a clustered installation, how is load balancing achieved?

As a business process is placed in one of the application queues prior to being executed, an evaluation is made as to whether it should be distributed or executed on the current node. If the determination is to distribute it, the current node selects another node to execute the business process. The algorithm that makes the determination of whether to try to distribute the load and where to send it is controlled by a scheduling policy. The default policy “FairShareSchedulingPolicy” determines if there is any node that has 10 percent (or more) load than the other nodes. The load factor is an approximation of how long before the business process executes (a factor of the queue depth and the average time that business processes are executing before relinquishing their thread). Each node in an application cluster multicasts its current load to all other participants in the cluster every few seconds.

How does the installation process differ when using clustering?

Each cluster instance of the application must first be installed as usual. After a standard installation is complete, the clustering configuration and setup described in this document must be performed on each cluster instance.

Can application clustering be used to achieve high availability?

An application cluster is the primary way to achieve high availability (HA). This must be combined with other HA components, including the database engine and front-end load balancing of TCP/IP connections.

In a clustered installation, how does the application system support fault tolerance?

If business processes are configured to take advantage of full persistence, many business processes can resume executing after a system failure by restarting at the step that was executing when the failure occurred.

The exceptions are generally steps that touch something outside of the application environment. For example, if an application node crashed while doing an HTTP POST and missed the acknowledgement, it would have no certain way of knowing if the POST completed or not. To address these exceptions, the default configuration for business processes is to require manual intervention to determine the correct way to restart after a failure.

What is the difference between failover and clustering?

Failover is having redundancy built into the environment, so that if a server fails, another server takes its place. This can be done in more than one manner, including by clustering. To use clustering for failover, configure a second server in the environment on a different computer than the primary server to handle some of the processing. When the primary server fails, the second server would continue to process, thus acting as a backup to the primary server (active/active approach).

Another failover method is to have additional hardware that is being used as a backup for this purpose but is not being used while the primary server is up and running (active/passive approach).

In a clustered installation, if an external resource isn't available, how does the application deal with this?

When a process that is trying to access a particular resource fails and does not fail over to another node, manual intervention is necessary to either continue the process or wait for the external resource to become available and the process is restarted or resumed.

The exception to this is the database. Application clusters assume that your database environment is configured for high availability. An application instance can survive short database outages, but it is strongly recommended that you avoid this situation. At the very least, a database outage causes some business processes to fail because they are unable to persist their state.

How does the application deal with clustering issues for external resources?

Resources that cannot be clustered must be identified and deployed to a specific node by configuring the properties files for that cluster instance. The failure of external resources is dealt with using on fault processing within a business process.

How does a clustered installation differ from a normal installation?

For specific information about installing or patching a cluster configuration, refer to the *Installation Guide*. Refer to the Installation Guide for information about both the common setup for all application servers and the specific setup for the individual application servers.

In a clustered installation, are there components other than the application that should be clustered?

This depends on the reason for clustering:

- If clustering is used for failover and the need to eliminate single points of failure, then all tiers of the environment must be protected.
- If you want to ensure the highest possible reliability for specific business processes, then additional effort is required in the design of those processes.
- If the purpose is performance, then additional effort is needed in arranging business processes so that the work can be efficiently split to take full advantage of the performance capabilities of a cluster.

In a clustered installation, do the application servers operate differently?

The most significant differences are the following items:

- Workload distribution

  In a cluster, business processes can migrate between nodes during their execution, according to the scheduling policy.

- Adapters

  In a cluster, the location of an adapter is significant.

- Operations

  In a cluster, an operation reports information from more than one node.

In a clustered installation, how is application scaling supported?

The application can be scaled through increasing the processing resources of each cluster node or by increasing the number of cluster nodes. In either case, you must ensure that the rest of the environment (network, database and underlying file systems) can handle these cluster node changes.

Whether clustered or not, the application responds well to tuning. Some tuning might help whenever more capacity is added to the application environment. Clusters, in particular, are sensitive to the length of the execution cycle and the number of threads allocated per node per queue and the distribution of business processes across queues (priorities).

Cluster Failover

Configuring a Clustered Installation for Failover

The simplest approach to availability (conceptually, though not necessarily to implement) is to use more than one server (with all of the redundancy covered previously in this document). With this method, any time one server fails, the other server simply takes over. This is called failover.

Application Level Failover (Active/Active Setup) in an IFC Clustered Installation

Another failover approach is to create a cluster in which the servers that participate are “active/active,” meaning that both nodes are working all the time, sharing the workload. This approach can be implemented at a higher level in the software stack, meaning that it generally does not have to be specific to any particular platform/operating system and usually does not require deep technical knowledge to configure. It also provides instant failover because both nodes are always active. This approach can be more cost effective than others, because the same hardware that provides failover also provides increased performance (except when there is a failure).

Clustering provides this kind of solution. All of the nodes participating in the cluster are available all the time. In an application cluster, nothing is required to be shared between the nodes, except for the database or (if the file system storage is used) the file system. Everything else can be configured separately. Performance and availability for some kinds of business processes can be enhanced by using a clustered file system like GPFS, Systina or Lustre (for example).

Failover and clustering can be combined to give each node in a cluster a “failover node.” This approach works very well when combined with test/development servers and/or disaster recovery. While complex, this kind of arrangement can provide extremely high levels of availability and provide for other needed services.

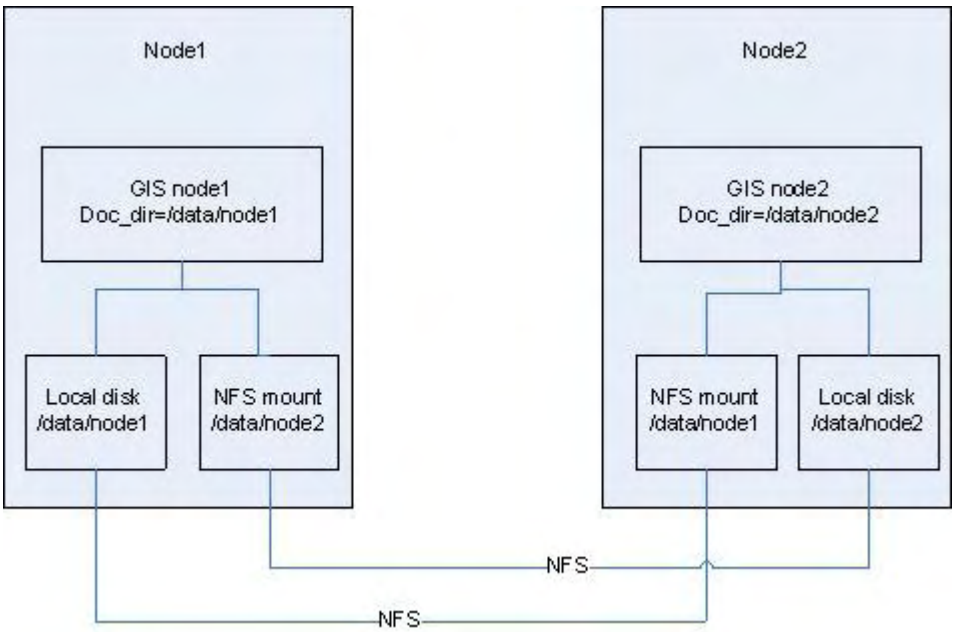
The components to consider for clustering are:

- Web servers
- Database servers
- Application server

In a cluster, administration and performance are enhanced by using a shared installation disk, with the disk for directories used by file system adapters. This can be accomplished through the use of a cluster file system, such as CFS, GPFS, or Sistina, or through any one of many highly available file servers (HA-NFS, for example) or Network Attached Storage devices with failover.

When using NFS to share files in a clustered environment, you must take care to reduce NFS use where possible. You can arrange the NFS directory organization to provide for local disk access in all cases except the initial step when a business process is distributed to a new cluster node. Organize the NFS directory by pointing the document_dir property in jdbc.properties to the local disk on each box and cross-mounting the local disk to the other cluster nodes. The document_dir property is used to tell the system where to write files when file system storage is used. When the file is read, the full path to the file is used instead of the document_dir property.

For example, in the following graphic, when a file is written on Node1, the Doc_dir on Node1 is /data/node1, so the full path to the file is /data/node1/file1, which is stored in the database. When either node reads the payload, the full path is read from the database and the file is accessed by that path. This works as long as both nodes have access to the path /data/node1.



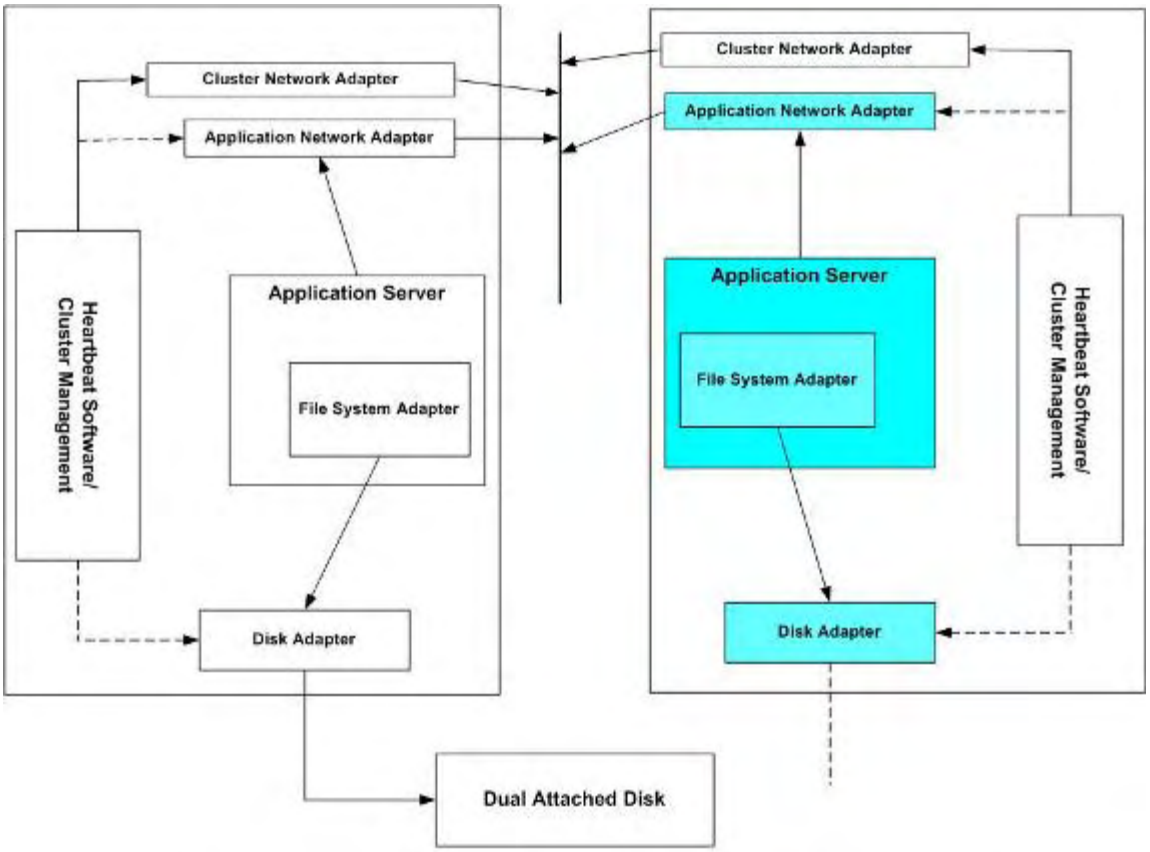
Hardware/Operating System Level Failover (Active/Passive Setup)

The most transparent way to implement failover is to use software packages/mechanisms from operating system vendors and third parties, such as SUN, IBM, HP or Veritas, that allow one server to become another server, taking over its network address, its disks and its workload. There is always some time delay involved in the switch.

Using this method requires that the system have at least one spare server. The spare could either be working on something else or it could be just waiting to be activated. If full capacity must be available, the spare must sit idle or only be used for tasks such as testing or development that can be interrupted. The spare can be

configured cold, where the node is switched off, or warm, where the node and as much higher-level software as possible is running to minimize takeover time.

The following graphic depicts a simple warm failover environment in which disk connections and IP addresses are taken over by the second node after a failure is detected by the cluster management software. In a real implementation, there would most likely be redundant paths to the disk and network for each adapter as well as an out-of-band heartbeat connection using serial or parallel ports.



Note: Configuring and testing this type of configuration could be very complex and time-consuming, unless you have significant experience in implementing failover. There are a large number of partial failure scenarios which need to be accounted for, including adapter failures, network partitioning and software failures.

Operating System Level Failover Without a Cluster

Operating System Level Failover (Without an Application Cluster)

The option of implementing failover in an application environment through the use of operating system level failover is a viable option for those not wishing to implement an application cluster. The possible reasons for implementing this kind of failover include:

- The designated backup platform may not always be available (possibly used for other purposes)
- A particular failover technology is widely used and understood at a particular site
- There are no facilities available to handle the front end redirection of network traffic
- Application requires no specific special configuration for OS level failover
- The primary purpose is disaster recovery at a remote site

There are also some reasons for avoiding OS level failover:

- Dependence on platform-specific technology
- Does not improve performance
- Usually has a longer failover time (it generally requires a start of the application, among other things)
- There is a risk in most environments for the two systems to become slightly out of synch which can cause a failure during the takeover process.
- It is not possible to reconfigure the application without taking an outage
- There are a variety of partial failure scenarios that have to be accounted for to ensure that the takeover is complete

Configuring for Operating System Level Failover

These solutions generally entail four key components, all of which are managed by OS/environment specific cluster management software.

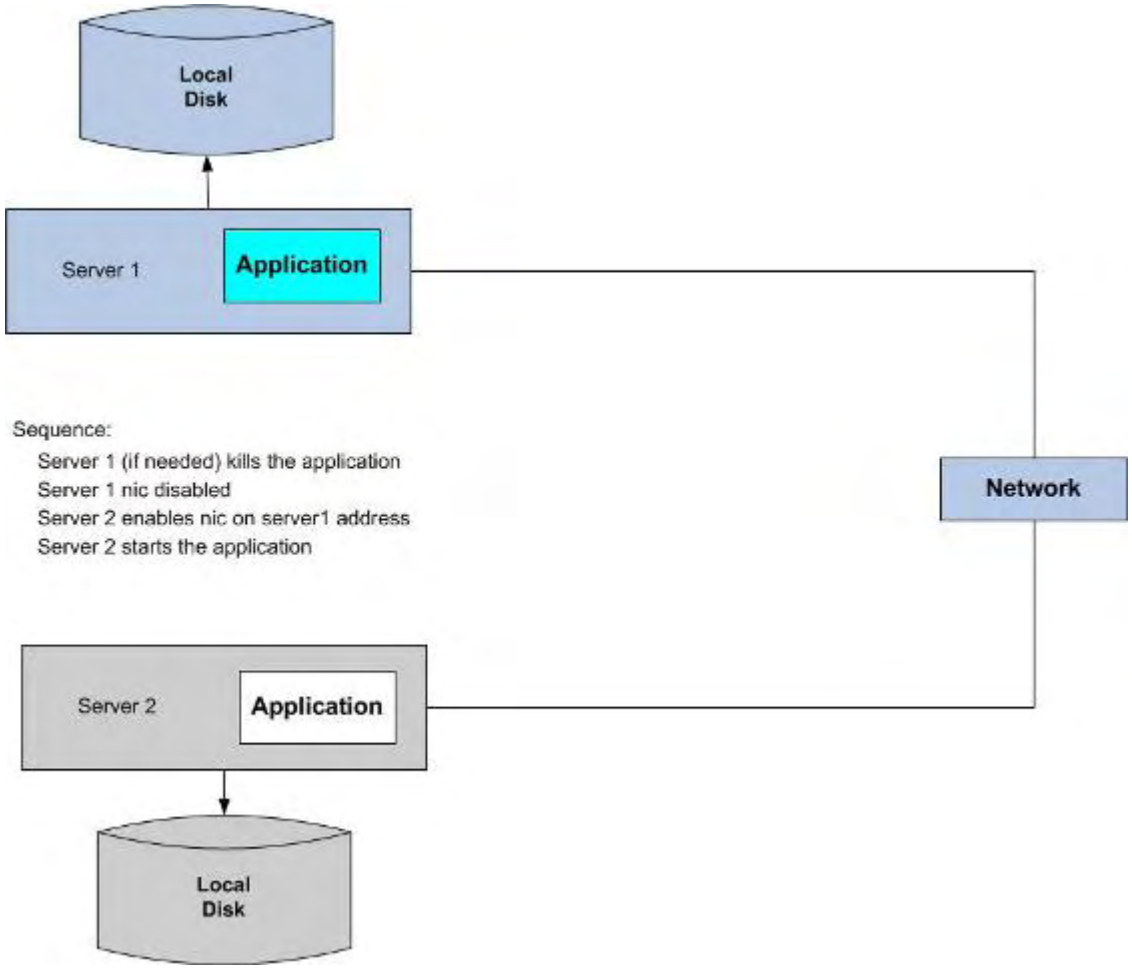
1. IP Takeover: The designated backup platform will take over the IP address of the primary platform. This might be accomplished through mechanisms within a particular host or through external network facilities like NAT (Network Address Translation).
2. Disk Takeover: The failover platform will get access to the disks that were being used by the primary platform (access from the primary host must cease). An alternative to this is to have the required disk storage attached using a shared filesystem or cluster filesystem.
3. Restart Logic: There will be some kind of restart logic to manage dependencies and restart all necessary components only when their resources are available. In a Unix environment, these are often scripts, in a Windows environment, it is likely to be a combination of scripts and configuration in the Microsoft Clustering Software.
4. Failure Detection: Usually handled by the OS level failover technology, it monitors the platform and/or the application to determine when a failover is needed. In addition to software, there are often hardware components like serial connections, additional isolated network connections and sometimes shared disk devices that are part of this mechanism to try to prevent partial takeovers in the event of partial failures.

The objective is to take a standard application installation on one node, detect failures, disable the first node, enable a second node, and start an identically configured application. The exact mechanics of this process will vary, depending on the hardware and software involved.

The following sections show some common arrangements of this process. Each of these configurations has advantages and disadvantages.

In all options except the first (separate local disk), the application is installed as if it were a standalone installation and configuration is done for the low level failover technology.

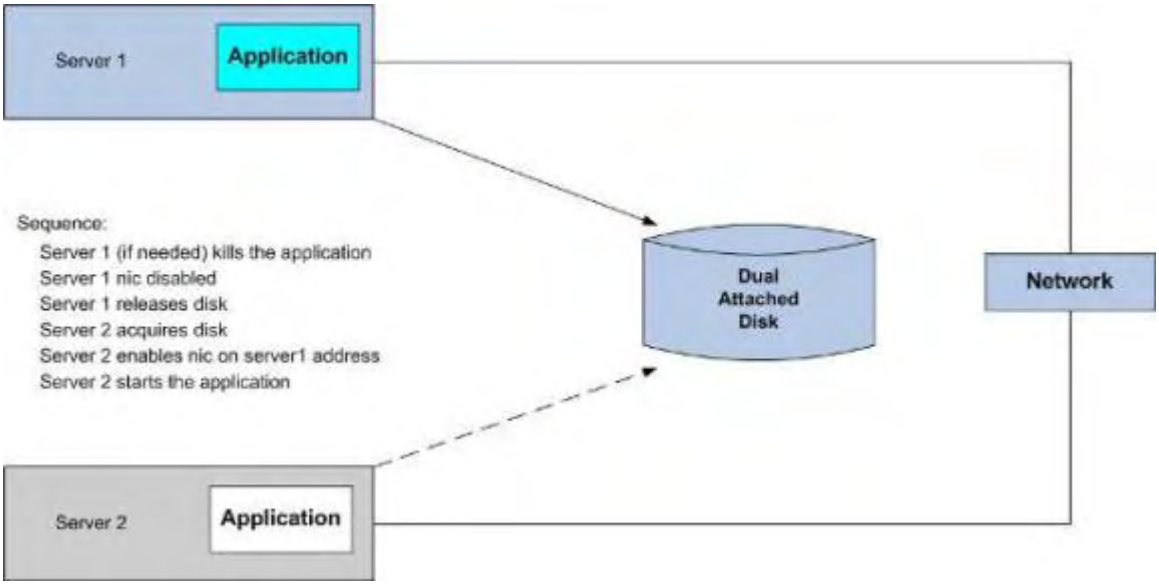
Option 1: Separate Local Disk (Application Installed Separately On Each Node)



This option, with a completely separate disk, is in many ways the simplest and cheapest way to go. It amounts to having two identically configured computers with the application installed identically on each.

The biggest problem with this approach is that the application installations are not automatically kept synchronized and because the application is licensed to IP addresses, it is not possible to bring up the application in the second environment while the first is operating. The synchronization issue can be partially mitigate through the use of automated file copying tools. Documents must be stored to the database in this configuration.

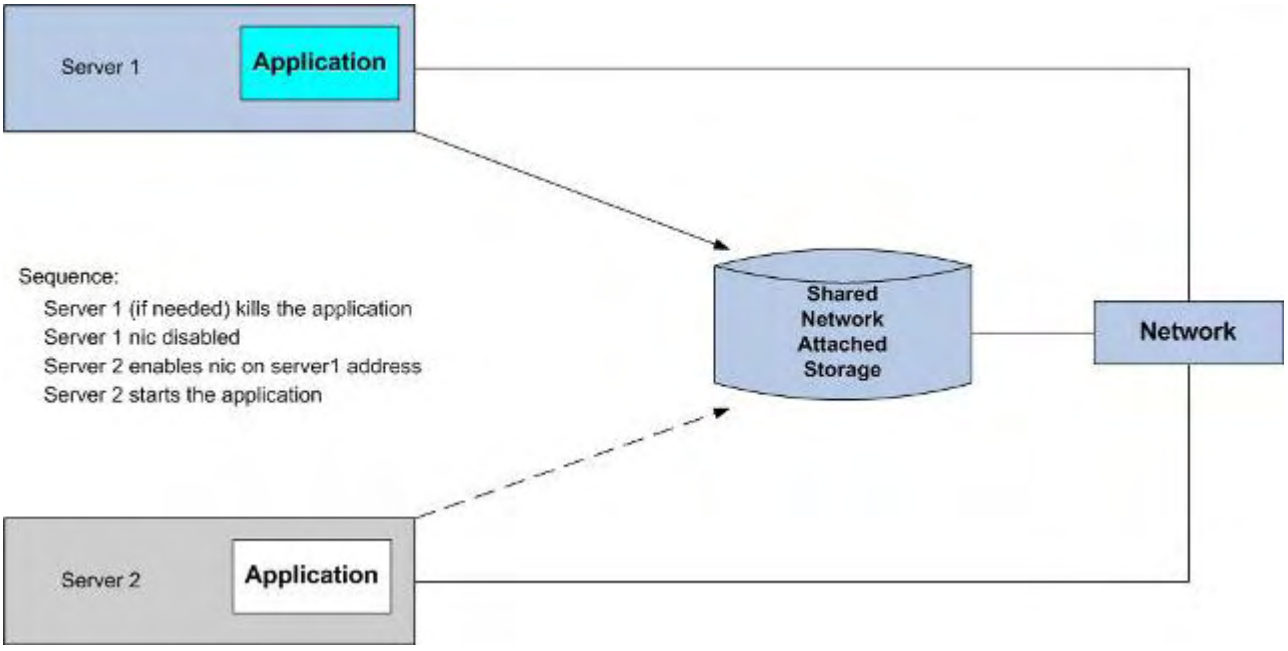
Option 2: Dual Attached Disk



This option, using dual attached hardware, requires disks that have the capability of being attached to two (or more) servers. Most modern external storage subsystems can do this (with varying degrees of sophistication).

In this environment, the failover process has the additional steps of logically moving the disks to the second server. This is usually handled more or less transparently by the OS level failover technology. It has the same performance as local disk, but the application instance is always up to date.

Option 3: Shared Network Attached Storage



This option (shared network attached storage), like NFS or Windows shared drives is also viable, but it does have two important issues:

- There is almost always a performance hit because the network attached storage is slower than local disk (the exception being if the application is configured to use local disk for temporary files and does not use the file system adapter extensively)
- The Network Attached Storage itself must be capable of essentially instantaneous/transparent failover.

This is an attractive option in lower volume configurations or where there is already Network attached Storage available with the desired characteristics.

Cluster Configuration Technical Options

Some of the technical options for cluster configurations include:

- Microsoft Windows
  - Windows Clustering (from Microsoft)
  - LifeKeeper (from Open Minds)
  - Cluster Server (Veritas)
- IBM AIX
  - HA CMP (IBM)
- Sun Solaris
  - SUN Cluster (Sun)
  - Cluster Server (Veritas)
- HP HP-UX
  - Service Guard (HP)
  - Cluster Server (Veritas)
- IBM OS/400
  - OS/400 Cluster Manager (IBM)
- Red Hat/SUSE Linux
  - Red Hat Cluster Suite
  - Service Guard (HP)
  - Other options, including some open source

Key Considerations

For all of these scenarios, key considerations are:

- Making sure that the machine configurations remain correctly synchronized (to whatever degree is needed), which may include maintaining differences as well as propagating changes. This is particularly critical after a failure, when repairs may involve configuration changes.
- Making sure that the partial failure scenarios are handled properly. In particular, network partitioning in which the cluster management software sees a failure, but the node is actually still running and is possibly accessing the database.

- Evaluating what the correct actions are in the case that disk takeover occurs, but the filesystem is corrupted and must be repaired. With modern journaling file systems, this is less of a problem, but it does happen, particularly in a heavily loaded environment.
- The failover capabilities must be tested and the level of assurance that it will work each time is much lower than with an active-active type cluster when both nodes are functioning all the time. This can make testing failover a risk in itself.

The most common scenarios for deploying this kind of failover technology are:

1. The application is configured to fail over to a development or test environment. This gets maximum utilization of hardware and with a test box, can help justify the expense of making it identical to the production box. This approach, however, does increase the risk of divergent system configurations.
2. The application is configured to fail over to a box that does nothing else except serve as a failover platform. This works well when a single failover platform serves several production systems (although this entails risks of its own).
3. The application is configured to fail over to a system running another application, often the database server supporting the application (with the database server also being configured to fail over to the application platform). While attractive from a cost standpoint, combining applications, even in a failure scenario can often result in inadequate performance.

Sample Operating System Level Failover: Testing Failover Using Windows Cluster Solution

1. Bring the application resource online from cluster administrator.
2. From the cluster administrator, select the resource **application at port xxxx**. Right click and select **Bring online**.
3. Go to machine 1 (primary node) application *install_dir*\install\logs and see if the application is coming up.
4. Once the application has completely started, hit the application URL (http://<*cluster-name*>:xxxxx/ws) and do basic testing to ensure that the application is running successfully.

Sample Operating System Level Failover: Initiating Failure from Cluster Administrator

1. From the cluster administrator, bring this resource online and wait for at least three minutes. Start Internet Explorer and specify the URL (similar to the following: http://<*cluster-name*>:xxxxx/ws) and verify that the application is working correctly. Verify there are no errors in the Event viewer application log and system log.
2. Go to the application *install_dir*\install\bin directory on machine 1 and issue a hardstop command. Wait several minutes to see the status of the Cluster administrator. It should say online pending - back to online.
3. Check the application logs on the primary node to see if the application is shut down on machine 1. Check the services on machine 1 (primary node) to see if they have stopped.
4. Check the cluster administrator status and see if it is back to online. Go to machine 2 (secondary node) and verify that the services have started. You can also check logs to see if the application is up and running without any errors on machine 2.

5. Go to http://*cluster IP*:*port*/ws to see if you can log in and do operations.
6. The application has now failed over to secondary node (node2)
7. You can repeat the procedure on machine 2 (secondary node) - issue a hardstop command, and see if it fails back to machine 1 (primary node).

Sample Operating System Level Failover: Bringing an Application Resource Online from Cluster Administrator

1. Go to cluster Administrator and Open connection to cluster configured.
2. Select resources from **Cluster Administrator**
3. Select - **Add New resource**
   - Provide a name of the resource - application at port xxxx
   - Description
   - Resource Type - Generic Service
   - Group - Cluster group that is configured
4. Go to the next:
   - Select the nodes that needs to participate for this resource in this cluster
5. Go to the next:
   - Dependencies - Just select **Cluster IP address (IP takeover - on failover)**
6. Go to next:
   - Service name - application service name which is "application at port xxxx"
   - No service parameters are required
7. Hit **Finish**

   This should create an entry for the new resource created with "application at port xxx" as the name in cluster administrator. The application is now configured to do failover.

Sample Operating System Level Failover: Windows Cluster Installation (Local Drive)

Install the application on machine 1 using the local drive from machine 1 at port xxxxx. Do not override the default IP address during the application installation or use the default machine 1 IP address. Apply service packs that needs to be applied as well.

1. Start the application on machine 1 and make sure the application is coming up successfully.
2. Get the cluster name and cluster IP address of the Windows cluster configured and setup.

3. Go to the application *install_dir*/install/properties dir. Backup sandbox.cfg and do the following changes:

```
EXT_HOST_ADDR= cluster IP address
HOST_NAME=cluster IP address
LOCALHOST=cluster IP address
```

4. Repeat the above installation procedure and testing in machine 2 nodes base port. The application must be installed in a local drive tested first.

Copyright