Multiple Machines

Multiple machine environments

Managing workloads

Redirecting servlet requests

7: Multimachine management

WebSphere Application Server applications can be scaled up from the basicsingle-machine configuration to run on systems comprised of multiple machines. Using amultimachine configuration enables applications to devote more processing power to clientrequests, distribute and balance loads among the machines in the system, and have betteraccessibility and throughput than single machine systems.

This section discusses the following topics:

- Using WebSphere Application Server in a multimachine environment discusses scaling up WebSphere systems, multimachine topologies, and related issues.
- Managing workloads discusses workload management using models and clones.
- Redirecting servlets discusses using remote Open Servlet Engine (OSE), thick servlet redirectors, and thin servlet redirectors to redirect client requests to servlet clones.

7.1: Using WebSphere Application Server in a multimachine environment

The basic single machine WebSphere configuration can be extended by distributing theapplication over multiple machines and by making more efficient use of the processingpower of each machine in the configuration. Some of the reasons for creating WebSphereApplication Server applications that run on multimachine systems include:

- **Scalability**. Increasing processing power by adding more machines enables the system to handle a higher client load than that provided by the basic, single-machine configuration. Ideally, it is possible to handle any given load by adding more servers and machines. Each additional machine must process its fair share of client requests. (That is, a share of the total system load that is proportional to its processing power.)
- **Security**. Multimachine configurations can use firewalls to protect sensitive resources from unauthorized access.
- **Shared data access**. Placing back-end resources such as databases on different machines can enable these resources to be shared more easily.
- **Availability and failover support**. In a single-machine configuration, any failure means that the entire system is unavailable. However, in multimachine configurations, the system continues to operate if any one machine or server in the system fails for any reason. Failover support distributes client requests to the remaining servers, which ensures continued client access without significant interruptions. (In practice, failover is not entirely transparent to clients.)
- **Fault isolation**. Configurations that provide simple failover support are concerned only with individual server failures that have no effect on the performance of other servers. However, in some situations a malfunctioning server can create problems for other servers that are otherwise functioning normally. For example, it can consume more than its share of system and database resources, preventing other servers from gaining adequate access to these resources. A configuration that provides more robust failover support includes a degree of fault isolation, reducing the potential for the failure of one server to affect other servers. WebSphere Application Server can be configured to provide fault isolation between different parts of a system.
- **Dynamic changes to configurations**. Administrators can modify the system's configuration without interrupting its operation. For instance, they can add or remove clones of servers to handle variations in the client load, change server characteristics and propagate the changes to its clones, temporarily stop servers for maintenance, and so forth. This enhances the manageability and flexibility of the system.
- **Mixed application server configurations**. Some multimachine configurations allow multiple versions of an application server to be deployed simultaneously. Applications can be deployed in stages and the system's hardware and software can be easily upgraded. When combined with the ability to make dynamic changes to the configuration, a mixed server configuration can be used to upgrade an application without any interruption of service.

Note: The ability to run different versions of an application server in aconfiguration applies only if the application servers are running under the same versionof the application code. You cannot run application servers under different versions ofWebSphere Application Server in the same administrative domain.

This section describes how you can achieve these goals in multimachine configurations.It is an overview of the various ways that you can use to scale up the basic,single-machine WebSphere system to meet the needs of your organization, and is notintended to be an exhaustive discussion of WebSphere configurations.

7.1.1: Scaling up WebSphere applications

Multimachine applications can be configured in a variety of ways to scale up a systemto add more processing power, improve security, maximize availability, and balanceworkloads. The WebSphere Application Server, Advanced Edition provides several ways to implementconfigurations that address these issues. These scaling techniques are generally combinedto maximize the benefits and minimize the problems associated with multimachine systems.

- **Cloning**. Cloning allows the creation of multiple copies of an object such as a servlet, enterprise bean, or entire application server. The first step is to create a model of the object based upon the object's current configuration. From the model, you can then create clones of that object. Clones can be created on the same physical machineor on different machines. Using clones can improve the performance of a server, simplify its administration, and enable the use of workload managment; however, there is a point of diminishing returns when adding more clones slows down the system due to the extra network traffic required for managing the clones.
- **Workload management (WLM)**. Incoming processing requests from clients are transparently distributed among the clones of an application server. WLM enables both load balancing and failover, improving the reliability and scalability of WebSphere applications. In addition, administrative servers can participate in WLM for failover support.
- **Open Servlet Engine (OSE)**. A Web server is responsible for receiving client requests, filtering them, and forwarding them to the servlet engine in an application server for processing. This forwarding is accomplished through a transport mechanism called OSE. OSE is a proprietary internal protocol that uses IPC mechanisms provided by the underlying operating system to transport data. OSE can be used to perform two types of load distribution in WebSphere. It can be configured to forward different URLs to different application servers, and it automatically distributes client requests among all available clones of a servlet (whether local, remote, or both). OSE automatically handles failover and changes in the available clones.
- **Servlet redirection**. A servlet redirector is a special-purpose application server that forwards HTTP requests from clients to a servlet engine in a cloned application server. It can participate in workload management for load balancing and failover support.
- **IP sprayer**. An IP sprayer transparently redirects incoming HTTP requests from Web clients to a set of Web servers. Although the clients behave as if they are communicating directly with a given Web server, the IP sprayer is actually intercepting all requests and distributing them among all the available Web servers in the cluster. IP sprayers (such as IBM Network Dispatcher or Cisco Local Director) can provide scalability, load balancing, and failover for Web servers.

7.1.2: Availability management

One of the benefits of scaling up to a multimachine configuration is that it improvesthe availability of the system. Applications hosted on multiple machines generally haveless down time and are able to service client requests more consistently.

The following commonly used scaling techniques can be combined to take advantage of the best features of each topology and create a highly available system. (Note that thisis not an exhaustive list of ways to improve availability.)

- Eliminate single points of failure in the system by providing hardware and process redundancy:
    - Use horizontal scaling to distribute application servers over multiple physical machines. If a hardware or process failure occurs, clones are still available to handle client requests. Web servers and IP sprayers can also benefit from horizontal scaling.
    - Use backup servers for databases, Web servers, IP sprayers, and other important resources. This ensures that they remain available if hardware or process failure occurs.
    - Deploy an application in multiple administrative domains. If an entire domain goes offline, the others are still available to handle client requests.
    - Run administrative servers with workload management enabled. The failover support that workload management provides eliminates a single administration server as a point of failure.
- Provide process isolation so that failing servers do not negatively impact the remaining healthy servers in the configuration. The following configurations provide some degree of process isolation:
    - Deploy the Web server onto a different machine from the application servers. This ensures that problems with the application servers do not affect the Web server, and vice versa.
    - Use horizontal scaling, which physically segregates application server processes onto different machines.
    - Deploy an application in multiple administrative domains. Problems are confined to one domain while the other remains available.
- Use load-balancing techniques to make sure that individual servers are not overwhelmed with client requests. These techniques include the following:
    - Use workload management. It is automatically implemented for cloned application servers, but must be explicitly enabled for administrative servers.
    - Use an IP sprayer to distribute requests to the Web servers in the configuration.
    - When using remote OSE, direct requests from high traffic URLs to more powerful servers. You can also make use of OSE's simple load-balancing facility to distribute requests among servers.
- Provide failover support. The application must continue to process client requests when servers are stopped or restarted. Ways to provide failover support include the following:
    - Use horizontal scaling with workload management to take advantage of its failover support.
    - Use remote OSE to distribute client requests among application servers.
    - Use the servlet redirector to distribute client requests among servlet engine clones.
    - Enable the Session Manager to store session information in a persistent database. This preserves session state in case of server failure.

7.1.3: Multimachine topologies

WebSphere Application Server supports a wide variety of ways to deploy applications inmultimachine environments. The most commonly used topologies fall into one of thefollowing broad categories:

- **Multi-tiered topologies.** The components of an application (the Web server, application servers, databases, and so forth) are physically separated onto different machines.
- **Vertical scaling topologies**. Additional application server processes are created on a single physical machine by using models and clones.
- **Horizontal scaling topologies**. Additional application server processes are created on multiple physical machines by using models and clones. HTTP redirector products such as Network Dispatcher can also be used to implement horizontal scaling.
- **HTTP server separation topologies**. The Web (HTTP) server is located on a different physical machine than the application server. Requests can be redirected to application servers through a variety of methods.
- **Demilitarized zone (DMZ) topologies**. Firewalls can be used to create demilitarized zones -- machines that are isolated from both the public Internet and other machines in the configuration. This improves security for the application, especially for sensitive back-end resources such as databases.
- **Multidomain topologies**. Applications are deployed onto multiple WebSphere Application Server administrative domains.
- **Multiapplication topologies**. More than one version of an application is deployed onto the same physical machines and administrative domain.

Keep in mind that these topologies are not mutually exclusive. Basic topologyelements can be combined in many different ways, as shown in the example topologiesfeatured in this section. These examples are not intended to be an exhaustive listof topologies that you can create in WebSphere Application Server. Instead, theyare intended to suggest various ways that you can set up applications in a multimachineenvironment.

7.1.3.1: Selecting a topology

A variety of factors typically are considered when you are deciding on the besttopology for deploying a WebSphere application. The major factors for picking atopology include:

- **Security**. Some security concerns can be addressed by physically separating the Web server from the application server by using firewalls.
- **Performance**. To maximize performance, the response time for transactions needs to be as short as possible. Two topologies can be used to improve transaction performance:
  - *Vertical scaling*, in which additional application server processes are created on a single physical machine. See article 7.1.3.3, Vertical scaling sample topology, for more information.
  - *Horizontal scaling*, in which additional application server processes are created on multiple physical machines to take advantage of the additional processing power available on each machine. See article 7.1.3.4, Horizontal scaling with clones sample topology, and article 7.1.3.5, Horizontal scaling with Network Dispatcher sample topology, for more information.
- **Throughput**. To process as many transactions as possible within a given time period, application server clones can be created to increase the number of concurrent transactions that the application can perform. These application server clones can be added through vertical or horizontal scaling.
- **Availability**. To avoid a single point of failure and maximize the system's availability, the topology must have some degree of process redundancy. High-availability topologies typically involve horizontal scaling across multiple machines. (Vertical scaling can improve availability by creating multiple processes, but the machine itself becomes a point of failure.) A Network Dispatcher server can direct client HTTP requests to the available Web servers, bypassing any that are offline; it can also be backed up by another server to eliminate it as a single point of failure. Workload management of application servers and administrative servers also improves availability and failover support.
- **Maintainability**. The system's topology affects the ease with which its hardware and software can be updated. For instance, using multiple WebSphere domains or horizontal scaling can make a system easier to maintain because individual machines can be taken offline for hardware and software upgrades without interrupting the application. However, sometimes maintainability conflicts with other topology considerations. For example, limiting the number of application server instances makes the application easier to maintain but can have a negative effect on its throughput, availability, and performance.
- **Maintaining session state between client HTTP requests**. This does not apply if your application runs on a single application server instance or is completely stateless. However, session state is an important consideration for stateful applications and applications that run on multiple machines or application server instances. A session can be shared between multiple application server processes (clones) by saving the session state to a database. In addition, the configuration of an HTTP redirector such as Network Dispatcher affects how the session state is maintained.

Whichever topology you decide on, a best practice is to partition your testing andproduction acceptance environments in exactly the same way as your production environment.This helps you recognize and address problems with your application before it is actuallydeployed.

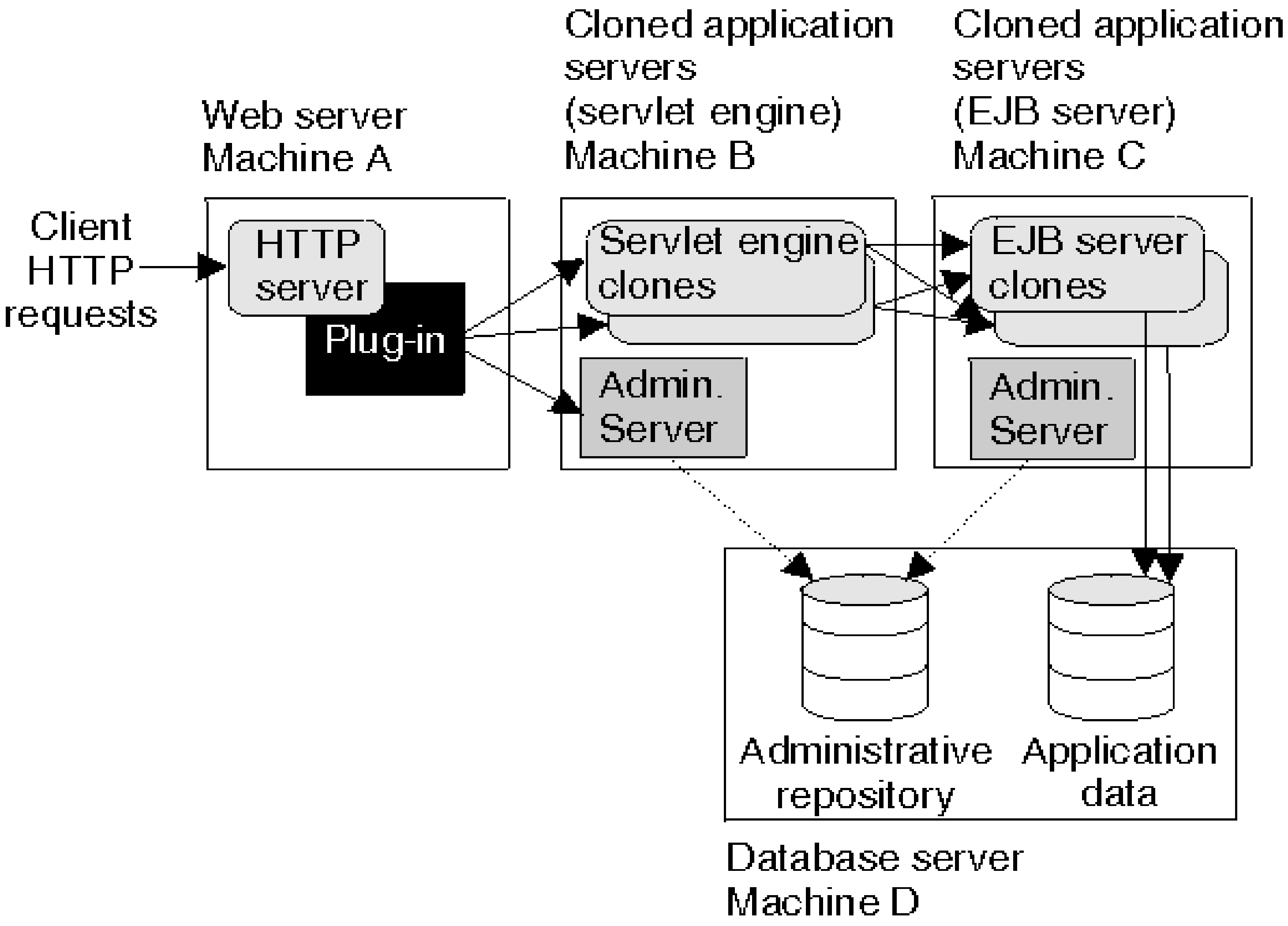
7.1.3.2: Multi-tiered system sample topology

- Overview
- Typical use

Overview

Multi-tiered topologies locate the Web server and application server processes onseparate physical machines.An additional tier can contain databases, enterpriseinformation systems, and other types of persistent storage.

The following illustration shows an example of this type of topology.


Cloned application servers (servlet engine) Machine B
Cloned application servers (EJB server) Machine C
Web server Machine A
Client HTTP requests
HTTP server
Plug-in
Servlet engine clones
Admin. Server
EJB server clones
Admin. Server
Administrative repository
Application data
Database server Machine D

In this example, the application server processes that run a servlet are closer innetwork terms to the HTTP server, improving their response to client requests. Theapplication server processes that run enterprise beans (Machine C) are closer in networkterms to the application data, which is represented in an application by entity beans andstored on the database server (Machine D). An administrative server process is running onthe two application server nodes.

Application servers are cloned on Machine B and Machine C to help maximize the use ofeach machine's resources. (Two clones of each are shown in the example, but depending onthe machine's hardware setup, more can potentially be added.)

Typical use

The clones in a multi-tiered topology provide process redundancy and enable memory tobe used more efficiently than in similar topologies that host only single instances ofapplication servers. The additional resources that are available on the machines in thistopology can improve the application's throughput and performance.

If firewalls are introduced between the three application tiers, the same level ofsecurity can be provided for the entity beans in the application server as for theapplication data.

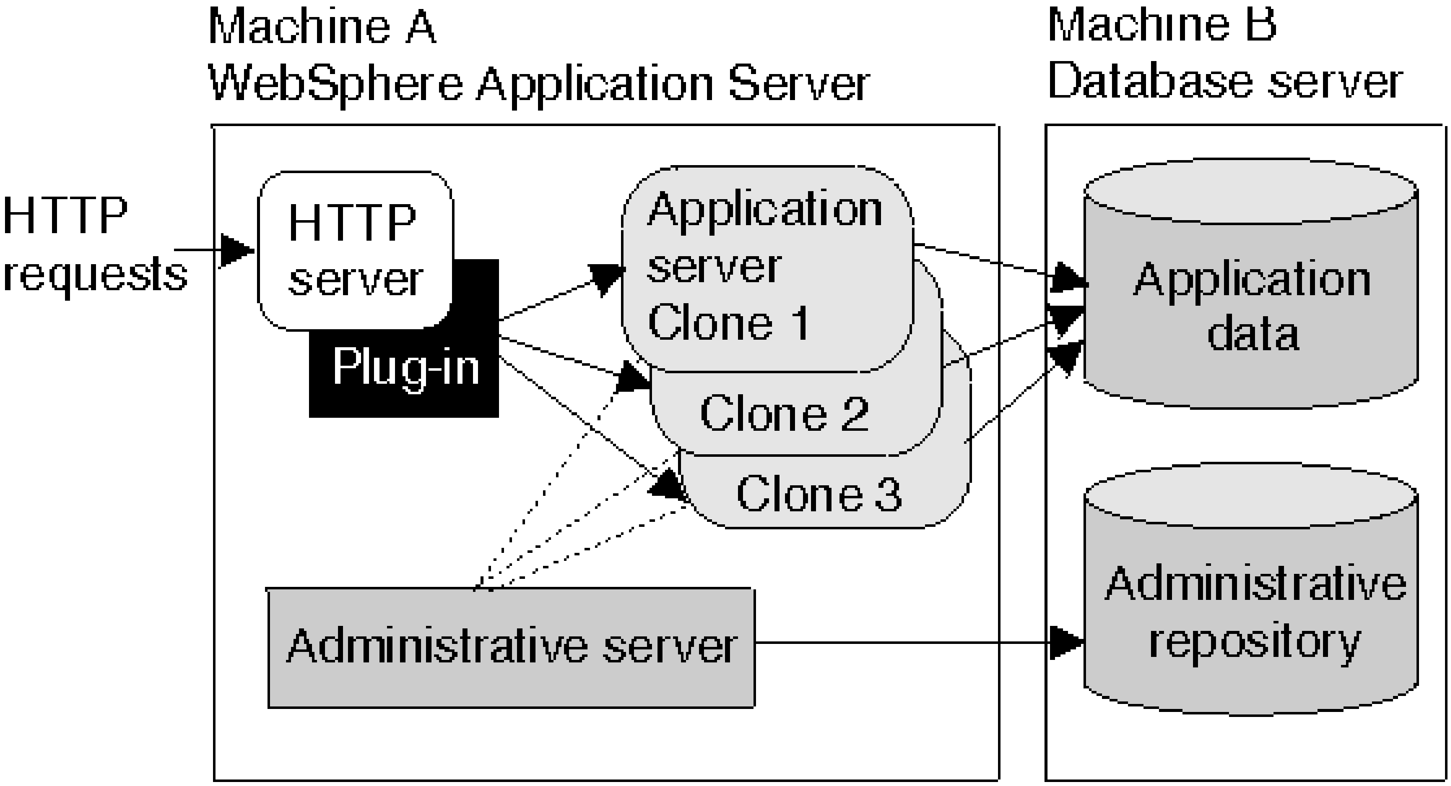
Implementing a multi-tiered toplogy eliminates the local Java Virtual Machine (JVM)optimizations that occur when both the servlet engine and EJB server run in the sameapplication server. It also introduces network latency. Both of these factors tend to slowdown system performance. Although they provide more redundancy for application serverprocesses, multi-tiered topologies also introduce more possible points of failure. Thelevel of redundancy can make maintenance more complicated.

7.1.3.3: Vertical scaling sample topology

- Overview
- Typical use
- Instructions

Overview

Vertical scaling refers to setting up multiple application servers, typically by usingclones, on a machine.



In this simple example, vertical scaling is done by creating multiple clones of anapplication server on Machine A. Although this example shows vertical scaling on asingle machine, you can implement vertical scaling on more than one machine in aconfiguration. (The Advanced Edition application server run time must be installedon each machine.) Combine vertical scaling with the other topologies described in thissection to boost performance and throughput.

Typical use

Vertical scaling offers the following advantages:

- More efficient use of the machine's processing power. An instance of an application server runs in a single Java Virtual Machine (JVM) process. However, the inherent concurrency limitations of a JVM process prevents it from fully utilizing the processing power of a machine. Creating additional JVM processes provides multiple thread pools, each corresponding to the JVM associated with each application server process. This avoids concurrency limitations and enables the machine's processing power to be fully used.

  Vertical scaling provides a straightforward mechanism for creating multiple instances of an application server, and hence multiple JVM processes. This enables the application server to make the best possible use of the processing power of the host machine.
- Load balancing. Vertical scaling topologies can make use of the WebSphere Application Server workload

management facility.

- Process failover. A vertical scaling topology also provides failover support among application server clones. If one application server instance goes offline, the other instances on the machine continue to process client requests.

Single machine vertical scaling topologies have the drawback of introducing the hostmachine as a single point of failure in the system. However, this can be avoided byusing vertical scaling on multiple machines.

Instructions

To set up a vertical scaling topology, use the administrative client to configure a setof application server clones that reside on the same machine. See Article7.2, Managing workloads, for more information on cloning an application server. To setup vertical scaling, you need only perform the tasks pertaining to local clones.

It is recommended that you plan vertical scaling configurations ahead of time. However,since they do not require any special installation steps, you can always implement themlater on an as-needed basis.

When you are deciding how many clones to create on a machine, you need to take severalfactors into account:

- The version of the Java development software. Version 1.2 and above of the IBM Java 2 Software Development Kit (SDK) handles parallel JVM processes better than earlier versions.
- How the application is designed. Applications that make use of more components require more memory, limiting the number of clones that can be run on a machine.
- The hardware environment. Vertical scaling is best done on machines with plenty of memory and processing power. However, eventually the overhead of running more clones cancels out the benefits of adding them.

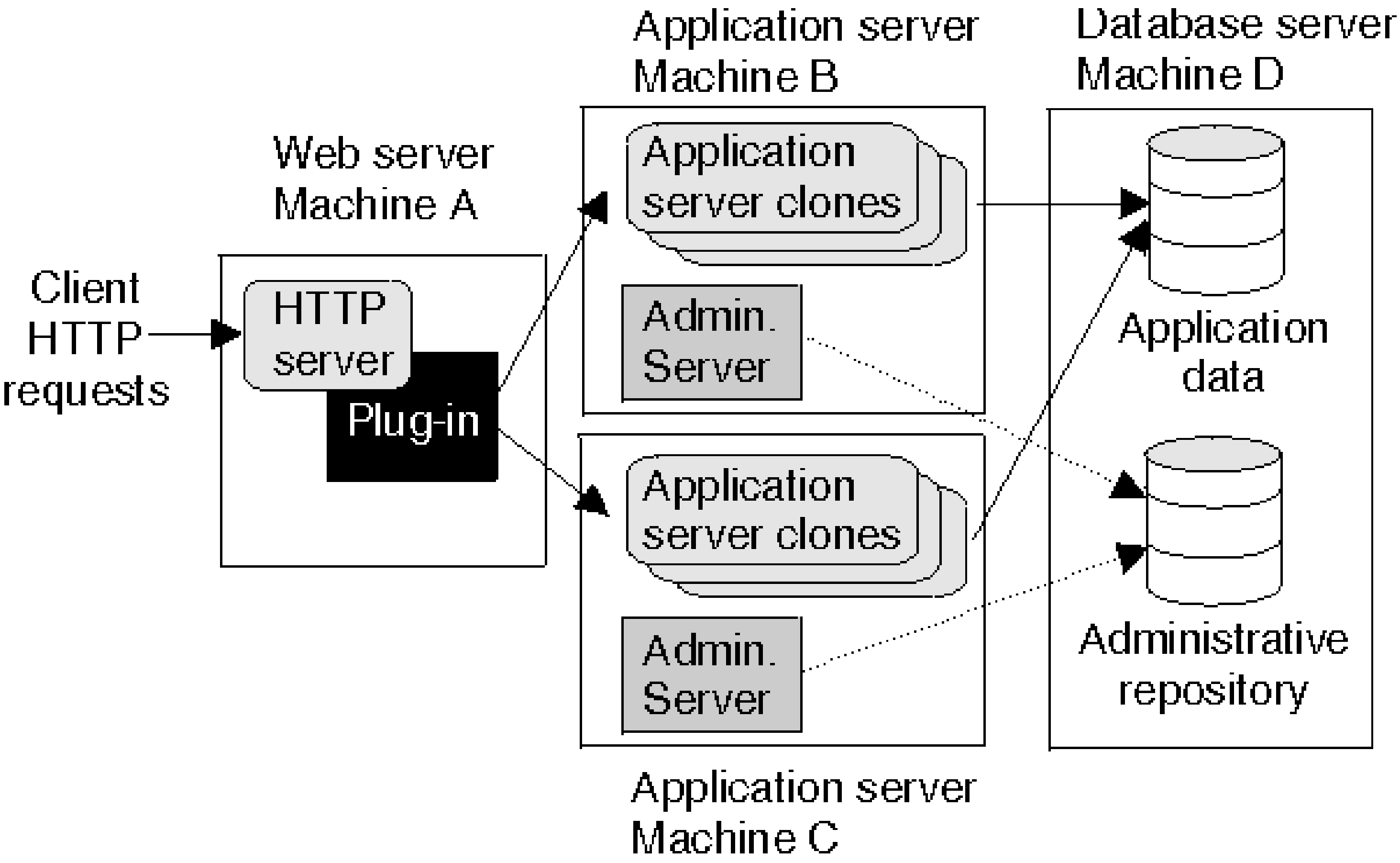
The best way to ensure good performance in a vertical scaling configuration is to tunea single instance of an application server for throughput and performance, thenincrementally add clones. Test performance and throughput as each clone is added.Always monitor memory use when you are configuring a vertical scaling topology and do notexceed the available physical memory on a machine.

7.1.3.4: Horizontal scaling with clones sample topology

- Overview
- Typical use

Overview

The following figure shows an example of horizontal scaling using clones of anapplication server.



In horizontal scaling, clones of an application server are createdon multiple physical machines. This enables a single WebSphereapplication to span several machines yet still present a single systemimage. In this example of a horizontal scaling topology, the Webserver on Machine A distributes requests to the cloned applicationservers on Machines B and C. The application server clones onMachines B and C are created from the samemodel. Machine Dacts as the database server for the application.

Products such as Network Dispatcher that distribute client HTTP requests can becombined with cloning to reap the benefits of both types of horizontal scaling. See section 7.1.3.5, Horizontal scaling with Network Dispatcher sampletopology, for more information on this system configuration.

Typical use

Horizontal scaling can provide both increased throughput andfailover support when compared to vertical scaling topologies. Bothapplication server process failure and hardware failure can be handledwithout significant interruption to client service. Horizontalscaling topologies can also be used to optimize the distribution ofclient requests through mechanisms such as workload management or the remote Open Servlet Engine (OSE) transport.

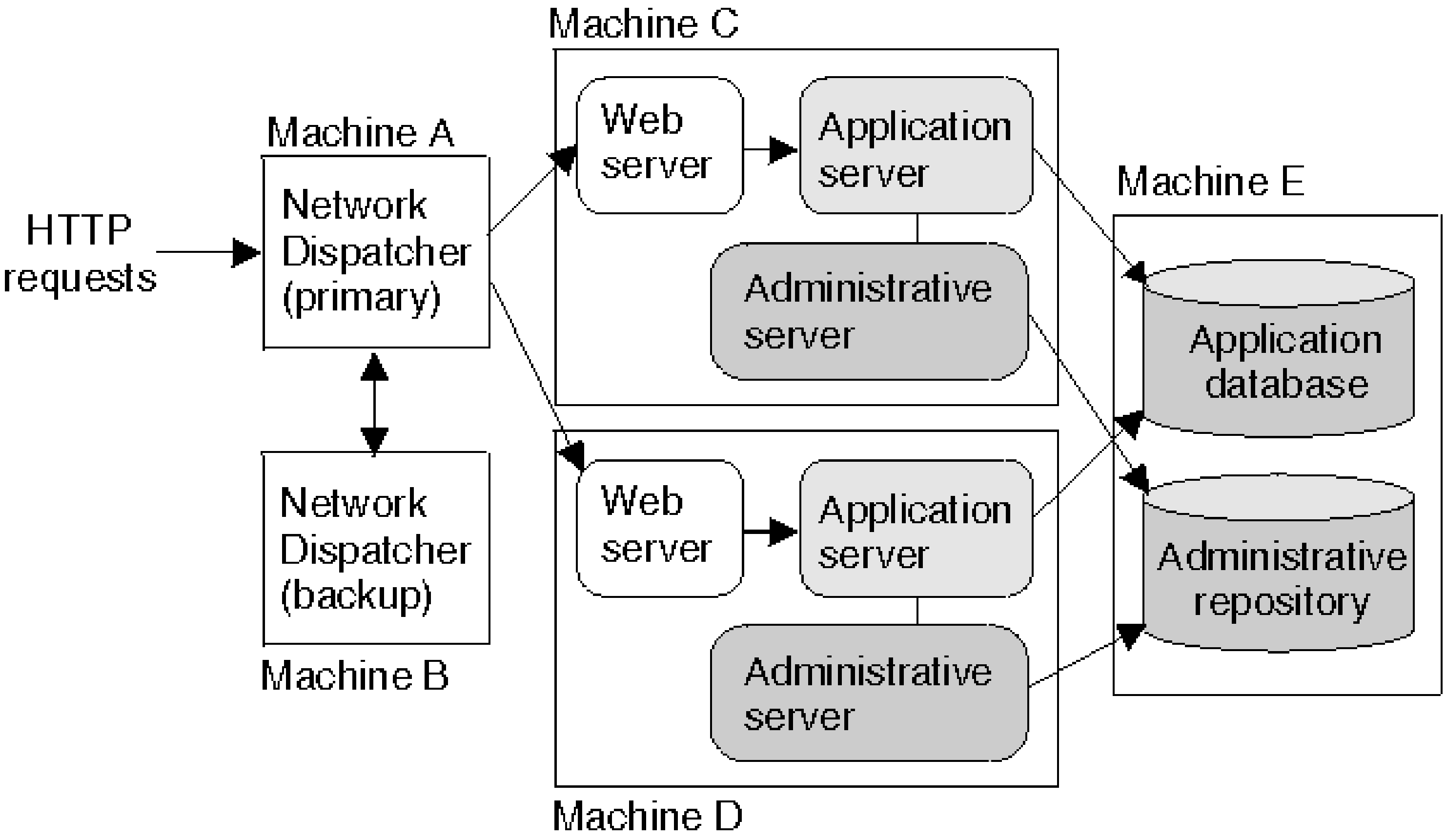
7.1.3.5: Horizontal scaling with Network Dispatcher sample topologies

Overview

A load-balancing product such as Network Dispatcher can be used to distribute HTTPrequests among application server instances that are running on multiple physicalmachines. Network Dispatcher is part of the IBM WebSphere Edge Server, which ispurchased separately from WebSphere Application Server It performs intelligent loadbalancing by using server availability, capability, workload, and other user-definablecriteria to determine which server the TCP/IP request is sent to.

A simple Network Dispatcher topology

The following figure illustrates a simple horizontal scaling configuration that usesNetwork Dispatcher to distribute requests among application servers that are located ondifferent machines.

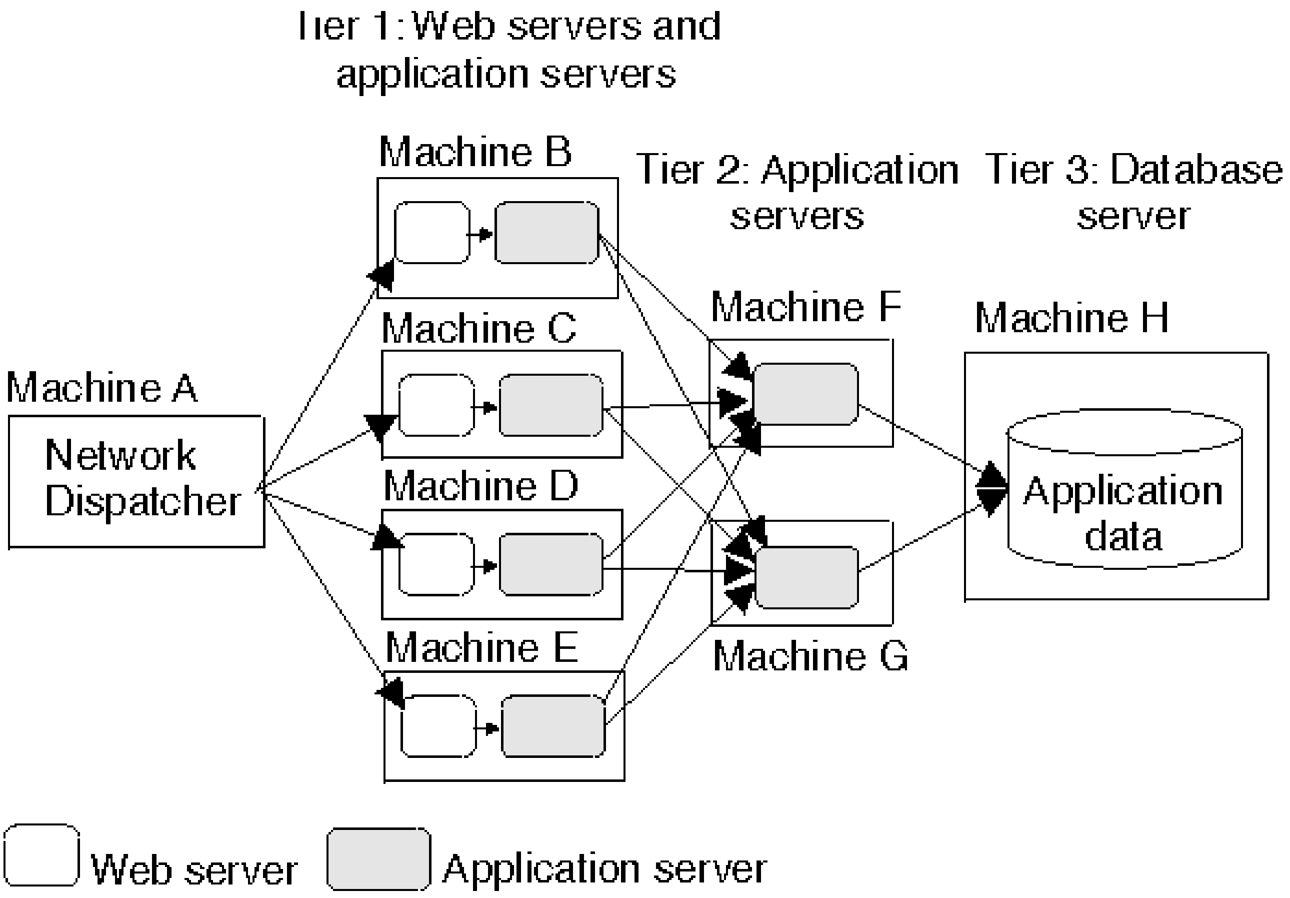


A Network Dispatcher machine is generally configured with a backup node to eliminate itas a single point of failure. In this example, the backup Network Dispatcher node (MachineB) can be set up to take over if the primary Network Dispatcher node (Machine A) fails.

The application servers in this example can be cloned from the same model or configuredindependently.

A more complex Network Dispatcher topology

The next figure shows a more complex configuration where Network Dispatcher is used todistribute requests among several machines containing clones of Web servers andapplication servers. For the sake of simplicity, the backup Network Dispatcher node andthe administrative servers are not shown in this example.



This example shows two tiers of application servers. The first tier Web server machineshost servlet-based applications, while the second tier application servers contain mostlyenterprise beans that access application data and execute business logic. This enables youto employ numerous, less powerful machines on the first tier and fewer but more powerfulmachines on the second tier.

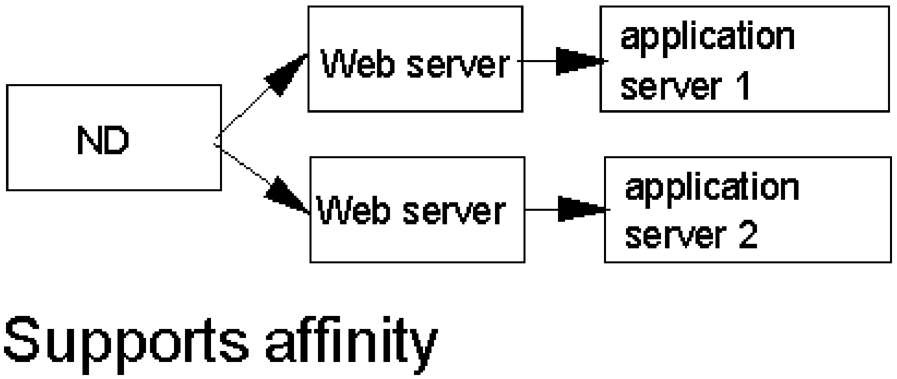
Using Network Dispatcher with firewalls

A load-balancing product such as Network Dispatcher can also be used with demilitarizedzone (DMZ) topologies. For example, it can simplify the creation of a DMZ topology whereone firewall protects the Web server from the public Web site and a second firewallprotects back-end systems from the Web server in the DMZ by using proxy services.

The Network Dispatcher machine is placed between the outside firewall and the clusterof Web servers that it serves. The outside firewall provides filtering to allow only HTTPand HTTPS traffic. The firewall to the back-end systems (DBMS, CICS, SAP, etc.) handlenon-HTTP protocols such as IIOP and JDBC. Because the administrative server needs toaccess the database for its configuration information, it is recommended that you placethe administrative server on the same side of the firewall as the database, rather than inthe DMZ. See section 7.1.3.7 and section 7.1.4 for more information on DMZconfigurations.

Network Dispatcher and session affinity

In a topology that uses Network Dispatcher or a product of similiar functionality, Webservers must be associated with separate application servers, rather than with clonedapplication servers, in order to preserve affinity among Web servers and applicationservers.



Cloned application servers use WebSphere workload management (WLM), which does notsupport session affinity. Requests originating at a Web server can be routed to any of theclones of an application server, and sessions cannot be guaranteed to remain intact.

Discussion

Adding a mechanism for distributing HTTP requests (such as the Network Dispatchercomponent of WebSphere Edge Server) provides the following advantages:

- It improves the performance of servers by distributing the incoming TCP/IP requests (in this case, HTTP requests) among a group of servers.
- It increases the number of connected users.
- It eliminates the Web server as a single point of failure. It can also be used in combination with WebSphere workload management to eliminate the application server as a single point of failure.
- It improves throughput by enabling multiple servers and CPUs to handle the client workload.

Instructions

To set up the machines containing Web servers and application servers, see theinstructions for the topology you plan to implement.

To place Network Dispatcher or another load-balancing product in front of the Webserver machines, see the documentation for the load-balancing product. Instructions varyby product.

The load-balancing product communicates with the Web server, which in turn communicateswith application servers. The configuration involves setting up communications between theload-balancing product and the Web server.

It does not matter to the load-balancing product whether the Web server is routingrequests along to an application server or processing them itself. Therefore, it is notnecessary to perform any special configuration to make the load- balancing product andapplication servers aware of one another. This is true with Network Dispatcher, based ontesting with IBM WebSphere Application Server. Results can vary with other load-balancingproducts.

7.1.3.6: HTTP server separation sample topologies

These topologies physically separate the Web (HTTP) server from the applicationservers, placing the Web server on a different machine in the configuration.Compared to a configuration where the Web server and the application servers are locatedon the same physical server, separating the Web server can improve applicationperformance, provide better fault isolation, and enhance security. These topologiesare often used with firewalls to create a secure demilitarized zone (DMZ) surrounding theWeb server.

WebSphere Application Server provides alternatives for physically separating the HTTPserver from the application server:

- HTTP transport configurations
- Reverse proxy (IP forwarding) configurations
- Open Servlet Engine (OSE) configurations, including remote and semi-remote OSE
- Thick servlet redirector configurations
- Thick servlet redirector with administrative-agent configurations
- Thin servlet redirector configurations

These system topologies are described in more detail in the articles in this section.

The following table summarizes the advantages and disadvantages of each of theseconfigurations. The criteria are explained after the table.

Topology	SSL	Database password required?	WLM	NAT	Performance	Administration
Remote OSE	No	No	Yes	Yes	High	Manual
Semi-remote OSE	No	No	Yes	Yes	Medium	Manual
Reverse proxy	Yes	No	No	Yes	High	Manual
Thick servlet redirector	Yes	Yes	Yes	No	Medium	Automated
Thick servlet redirector with administrative agent	Yes	No	Yes	No	Medium	Automated
Thin servlet redirector	Yes	No	Yes	No	Medium	Manual

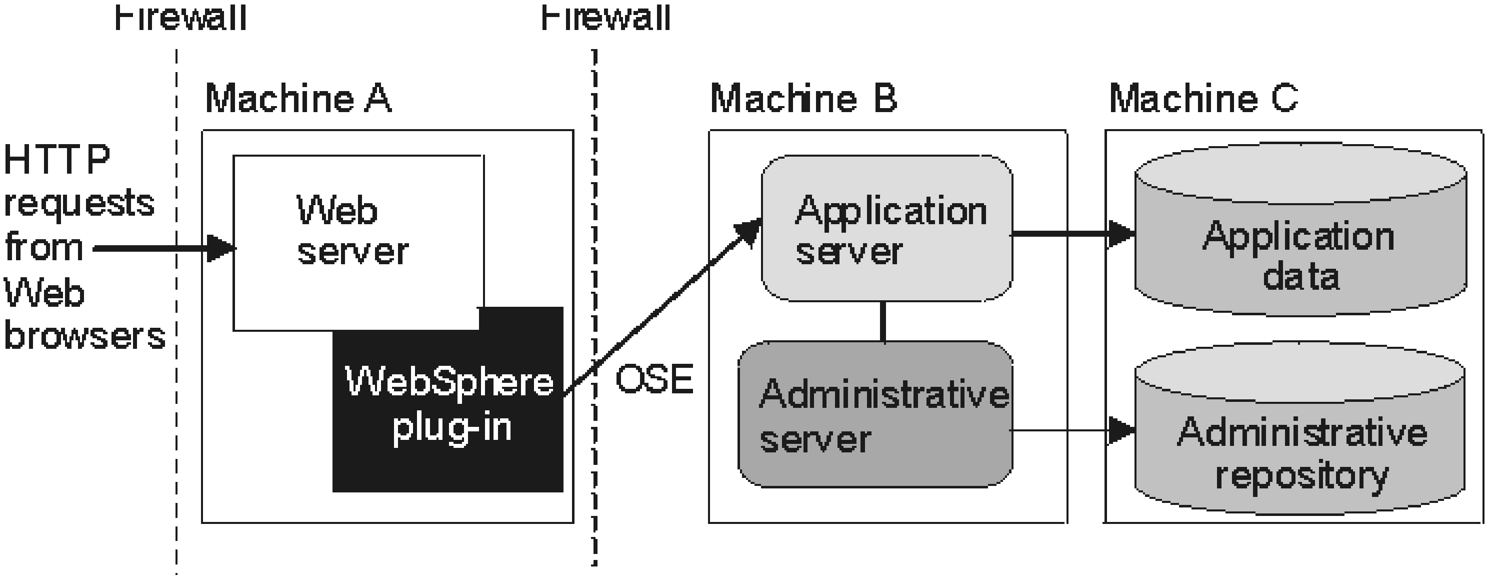
- **SSL.** Supports Secure Sockets Layer (SSL) security.
- **Database password required?** Requires a database user ID and password to be stored on the machine for use by the database processes.
- **WLM.** Uses the WebSphere workload management service to balance client workloads.
- **NAT.** Supports Network Address Translation (NAT) firewalls. NAT firewalls receive packets for one IP address, translate the headers of the packets, and send the packets to a second IP address.
- **Performance.** Compares the relative performance of each of these configurations.
- **Administration.** Specifies whether the configuration must be administered manually or can be administered through the Administrative Console. This gives you a basis to compare the relative difficulty of administering each configuration.

7.1.3.6.1: Remote OSE sample topology

- Overview
- Load-balancing support
- Failover support
- Typical use
- Instructions
- Updating the remote OSE configuration

Overview

Open Servlet Engine (OSE) is a lightweight communication protocol developed by IBM forinterprocess communication. Remote OSE uses this proprietary transport to route requestsfrom the Web server plug-in to application servers on remote machines.



In the diagram, Machine A hosts the Web server and receives HTTP requests from clients.The Web server forwards the requests to the application server on Machine B by using theWebSphere plug-in for the Web server and some special configuration. Machine C hosts theapplication and administrative repository databases.

Variations on this configuration include vertical scaling of the application servers,which is discussed in Vertical scaling sample topology.Additional application server machines (D, E, ... N), can be added to the configuration toimplement horizontal scaling.

Usually, a WebSphere administrative server on a Web server machine generates Web server plug-in configuration files to tell the Web serverhow to route requests. However, the Remote OSE configuration does not place anadministrative server on the Web server machine. Instead, a Remote OSE script runs on theWeb server machine, communicating with an administrative server on the remote applicationserver machine. The script gathers the necessary information about the application serverconfiguration and generates the plug-in configuration files.

The script uses RMI/IIOP to communicate with the remote administrative server. RMI/IIOPdoes not work through a firewall that performs Network Address Translation (NAT). However,Remote OSE can be modified to support environments using NAT. The instructions in thissection include the additional steps required to run Remote OSE in a NAT firewallenvironment. (See Instructions for details).

Remote OSE requires the following firewall ports to be opened:

- One port for each application server or clone process.
- A port if WebSphere security is used on the machine that hosts the Web server (Machine A).
- A port to run the remote OSE configuration script, OSERemoteConfig.

See Instructions for details.

For more information on firewall configurations in WebSphere Application Server, seearticles 7.1.3.7 and 7.1.4.

Load-balancing support

OSE is fully integrated with WebSphere Application Server's cloning facility. Itbalances loads between individual application servers and their clones, and among theclones of an application server. This load balancing is separate from the workloadmanagement facility.

- **Load balancing between application servers.** OSE can be configured to forward requests from each URL to a different application server and its clones, enabling manual load balancing. For instance, URLs that generate a large number of requests can be forwarded to application servers on more-powerful machines.
- **Load balancing among application server clones.** OSE automatically distributes requests among the clones of an application server that

is defined to respond to a single URL.The method for selecting which clone handles a particular request combines a round-robin selection policy with server affinity.

If session persistence is not enabled (the default), requests are distributed among all available clones of an application server using a strict round-robin policy. Each clone gets the next request in turn. The only exception is when a clone is added or restarted; see Failover support (later in this article) for details.

If session persistence is enabled (that is, session clustering and server affinity are enabled), requests are distributed as follows:

- OSE distributes the first request of each session and all requests that are not associated with a session as if session persistence is not enabled. That is, they are distributed using a round-robin policy except when clones are added or restarted.
- OSE attempts to distribute all requests associated with a particular session to the same clone of an application server. Different sessions are assigned to different clones of the application server.

  Be aware that there is no guarantee that the same clone will be used for all requests within a session. Session affinity cannot always be maintained in situations where the number of available clones changes during the lifetime of a session. The Session Manager's session clustering facility ensures that session state is not lost if requests are switched to another clone during a session.In any case, applications that require session information to be available across multiple client invocations must store session information in a database.

Session persistence cannot be used with the workload management facility.

Failover support

OSE automatically handles failover and changes in the number of available clones.

- If a clone is stopped or unexpectedly fails, all subsequent requests are distributed among the remaining clones. The unavailable clone is skipped.
- If a clone is added or restarted, the system automatically begins to distribute requests to it. The next several requests are dispatched to that clone before OSE resumes its normal methods for distributing requests to the clones of an application server based on whether session persistence is enabled. (See Load-balancing support, for details.)

Typical use

Remote OSE has the following advantages:

- It supports load balancing and failover.
- It does not require database access through the firewall. The administrative server runs on the machine that hosts the application server, which is typically behind the firewall.
- It supports WebSphere security.
- It works with NAT firewalls.
- Performance is relatively fast.

Remote OSE also has the following disadvantages:

- It does not support Secure Sockets Layer (SSL) encryption for communications between the Web server and the application server.
- It requires at least one firewall port, more if multiple application server clones are configured, the OSERemoteConfig script is used, or WebSphere security is used on the machine hosting the Web server.
- Its configuration is relatively complicated.

Instructions

The instructions describe how to set up the illustrated configuration, allowing formultiple Web server machines (like Machine A) and application server machines (likeMachine B). They are summarized as follows:

1. Install the appropriate WebSphere Application Server components.
2. Start the database server, administrative server and administrative console.
3. Configure the application server on Machine B.
4. Start the application server on Machine B.
5. Configure the Web server plug-in on Machine A for Remote OSE.
6. If a firewall is being used between Machine A and the other machines in the configuration, open the appropriate firewall ports.
7. If WebSphere security is being used to secure resources on the Web server, modify the appropriate configuration files.
8. Restart all Web servers, application servers, and administrative servers.

The rest of this section describes how to complete these steps.

Install WebSphere Application Server components

The product components required for Remote OSE are listed in the following table:

	Web server	**Application server**	**Web server plug-in**	**Administrative server**
Machine A (and possibly others)	✓	✓ When doing the installation, select **Core server** and the appropriate plug-in for your Web server.	✓	✓ The administrative server, Web server plug-in, and IBM Java 2 SDK must be installed on Machine A in order for the OSERemoteConfig tool to run successfully.
Machine B		✓ Perform a production application server installation.		✓
Machines D, E, ...N		✓ Perform a production application server installation.		✓

Machines D through N are additional, optional application server machines that hostclones of the application server on Machine B. They do not contain the default resources.

Machine C contains the administrative database for all of the above administrativeservers, as well as the database for application data. However, the arrangement shownunder Overview is not the only option for database placement. Forexample, Machine C could be omitted if the database resides on B, D, E or another machineinstead. Note that a DB2 client is required for accessing a remote DB2 database.

Start the database server, administrative server, andadministrative console

Before you configure the plug-in for OSE remote, start the software that is needed forthe configuration:

1. Start a DB2 instance for the administrative repository (if it was not already started).
2. Start the administrative server on Machine B.
3. Start the administrative console on Machine B.
4. Make sure that Machine B is listed in the administrative console.

Configure the application server on Machine B

Configure the application server on Machine B by completing the following steps:

1. Add the aliases of each Web server machine to the alias list of the virtual host for Machine B by following these steps:
    a. Locate and click the virtual host in the Topology tree view to display its properties.
    b. In the virtual host alias list, specify the host names and IP addresses of the Web server machines. Add the port numbers, if not port 80.
    c. Save your changes and close the properties dialog box.

    For example, if the configuration contains two Web servers (a.bigcorp.com and b.bigcorp.com) talking to the application server against the same virtual host, add aliases for a.bigcorp.com and b.bigcorp.com and their IP addresses.

    ℹ Application servers that have already been started must be stopped and restarted before they can use the new host aliases.
2. Configure the transport type for the OSE queue. The servlet engine must be configured to use INET sockets instead of local pipes as the transport type. This is the default for Solaris machines. However, you must explicitly select INET sockets as the transport type on other supported operating systems. To do so, follow these steps:
    a. Locate and click the servlet engine in the Topology tree view to display its properties.
    b. In the advanced properties, access the plug-in configuration settings.
    c. Change the transport type to INET Sockets.
    d. Click OK.
    e. Save your changes and close the properties dialog box. The new transport does not take effect until you restart the application server.

On Windows-based systems, the default application server does not start when the servlet engine configuration is changed to OSE INET sockets. You must change the servlet engine Maximum Connections parameter to 25 (from the default value of 50).

3. If NAT is being used to resolve IP addresses, the interoperable object reference (IOR) must include the host's short name to enable clients inside the firewall to locate the desired server using a DNS server or a HOSTS file on the client machine. To include the short name instead of the IP address or the fully qualified name in the IOR, add the following line to the Command line arguments field, which is located in the general settings tab for the default server on the administrative console:

   ```
   -Dcom.ibm.CORBA.LocalHost=short_name
   ```

   If NAT is using indirect IORs or IORs returned from naming services, this line must also be added to the following file:

   ```
   product installation root/bin/admin.config
   ```

4. Make any other desired changes to the application server.
5. If you are planning to clone the application server, do so now. As a model, use the application server you just configured.

Start the application server

Start the application server (and its clones, if applicable).

Wait for the product to refresh the automatically generated plug-in configurationfiles. This usually takes a few minutes. Verify that a refresh operation has occurred bychecking the time stamp in any of the three properties files in the *product installation root*/temp directory.

Configure the Web server plug-in on Machine A for Remote OSE

You can configure the Web server plug-in on Machine A either automatically or manually.

Automatic configuration

You can automatically configure the Web server plug-in by running the remote OSEconfiguration script, **OSERemoteConfig**, on each Web server machine. TheOSERemoteConfig script cannot be used if a NAT firewall is between the Web server machine(Machine A) and the application server machines (Machine B and D through N).

1. If a non-NAT firewall is in use, you must open up a port to run the OSERemoteConfig script. Configure the CORBA listener port for the administrative server that is running on Machine B as follows:
   a. Add the -Dcom.ibm.CORBA.ListenerPort parameter to the line com.ibm.ejs.sm.util.process.Nanny.adminServerJvmArgs in the *product installation root*/bin/admin.config file on Machine B. For example:

      ```
      com.ibm.ejs.sm.util.process.Nanny.adminServerJvmArgs -Dcom.ibm.CORBA.ListenerPort=33000
      ```

   b. Stop and restart the administrative server on Machine B.
2. Open a command window on Machine A.
3. Change directory to *product installation root*/bin.
4. Issue the following command to start and configure remote OSE:
   - On Windows systems:

     ```
     OSERemoteConfig -adminNodeName administration_server_hostname
     ```

   - On UNIX-based systems:

     ```
     OSERemoteConfig.sh -adminNodeName administration_server_hostname
     ```

     where *administration_server_hostname* is the host name of the WebSphere application server machine to which requests should be routed.

     OSERemoteConfig has additional, optional parameters. For details, type OSERemoteConfig (add .sh on UNIX systems) without any parameters.
5. Wait for OSERemoteConfig to complete.

Manual configuration

Manually configure the Web server plug-in for remote OSE if a NAT firewall is beingused between the Web server machine (Machine A) and application server machines (MachinesB and D through N) or if you do not want to open up an additional port in a non-NATfirewall. Follow these steps:

1. Copy the following Web server plug-in configuration files from the *product_installation_root*/temp directory of Machine B to the same directory on Machine A:
   - vhosts.properties
   - rules.properties
   - queues.properties

All users must have read and write access to these files.

2. Add a host entry for each application server or clone to the queues.properties file on the Web server machine (Machine A). The format of the entry is:

   ```
   ose.srvgrp.ibmoselink.clone.host=clone_hostname
   ```

   where *clone*.is the name of the application server or clone and *clone_hostname* is the IP address or host name of the machine where the application server or clone is running. The original application server or server group is *clone1*. For example:

   ```
   ose.srvgrp.ibmoselink.clone1.host=bigcorp1
   ```

3. If you are running WebSphere 3.5 Fix Pack 1 or earlier, you must modify the queues.properties file on Machine B. For each clone on a machine other than Machine B (in other words, for each clone on a machine other than the one specified in the -adminNodeName argument to OSERemoteConfig), do the following:
   a. Create a type entry, such as:

      ```
      ose.srvgrp.ibmoselink.clone2.type=remote
      ```

   b. Create a port entry, such as:

      ```
      ose.srvgrp.ibmoselink.clone2.port=servlet_engine_port
      ```

      To find the servlet engine port, use the administrative console to check the Advanced properties for the servlet engine contained in the cloned application server.
   c. Create a host entry, such as:

      ```
      ose.srvgrp.ibmoselink.clone2.host=clone_hostname
      ```

      where *clone_hostname* is the host name of the machine where the cloned application server is running.
   d. Update the clonescount property to match the number of clones configured, such as:

      ```
      ose.srvgrp.ibmoselink.clonescount=2
      ```

Open ports in the firewall

If a firewall is in use between Machine A and Machines B and D through N, open theports listed in the file:

`product_installation_root/temp/queues.properties`

Using remote OSE with WebSphere security

To secure the resources on the Web server (Machine A) with WebSphere security, you needto do the following:

1. If a firewall is in use, open a port between Machine A and Machine B. See Open ports in the firewall for details.
2. Modify the *product_installation_root*/properties/bootstrap.properties files on both Machine A and Machine B (the administrative server machine) as follows:
   a. Modify the bootstrap.properties file on Machine A as follows:
      i. Set the clone type property to remote:

         ```
         ose.srvgrp.ibmappserve.clone1.type=remote
         ```

      ii. If necessary, change the clone port number:

         ```
         ose.srvgrp.ibmappserve.clone1.port=port_number
         ```

         where port_number is the number of an unused port that is assigned to the clone. The clone port numbers on Machines A and B must be identical.
   b. Modify the bootstrap.properties file on Machine B as follows:
      i. Set the clone type property to remote:

         ```
         ose.srvgrp.ibmappserve.clone1.type=remote
         ```

      ii. If necessary, change the clone port number:

         ```
         ose.srvgrp.ibmappserve.clone1.port=port_number
         ```

         where port_number is the number of an unused port that is assigned to the clone. The clone port numbers on Machines A and B must be identical.
      iii. Set the clone host name property:

         ```
         ose.srvgrp.ibmappserve.clone1.host=admin_server_hostname
         ```

         where *admin_server_hostname* is the host name of Machine B, the machine where the administrative server is running.

Restart all Web servers, administrative servers, and applicationservers

For the remote OSE configuration to take effect, you must stop and restart all Webservers, administrative servers, and application servers on each machine in the system.

Updating the remote OSE configuration

You need to update the remote OSE configuration after doing any of the following:

- Adding or removing a URL (Web resource) in the environment
- Adding or removing a virtual host alias
- Changing the queue properties of a servlet engine (such as the name or port)
- Adding or removing a servlet engine
- Adding or removing an application server clone

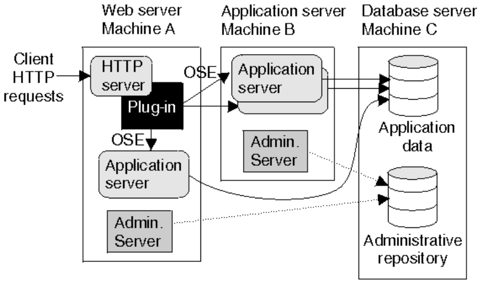
See Configure the WebSphere plug-in on Machine A for remote OSEfor details.

7.1.3.6.2: Semi-remote OSE sample topology

- Overview
- Typical use
- Instructions

Overview

Semi-remote OSE is a variation on the remote OSE topology. The difference between thetwo topologies is whether an instance of the application server runs on the machinethat hosts the Web server. A semi-remote OSE configuration has an instance of anapplication server running on the same machine as the Web server; a remote OSEconfiguration does not.



Like remote OSE, semi-remote OSE can be used to direct client requests to additionalapplication server clones on other machines. In this example, it redirects client requeststo both the application server instance running on Machine A and the clones running onMachine B.

Typical use

Semi-remote OSE requires more steps than remote OSE. Using a semi-remote OSEconfiguration is recommended only in situations where hardware limitations prevent youfrom hosting the Web server on a dedicated machine.

In many production environments, one set of servers is configured to run HTTP serversand another set of servers is configured to run application servers. If a customer eitherneeds to add capacity in a production environment or cannot fully replicate the productionconfiguration in the production test environment, semi-remote OSE provides a means ofload distribution between a machine hosting both the HTTP and application server andmachines hosting just the application server.

A semi-remote OSE configuration can also be used as a WebSphere proof of concept fordemonstrating OSE load distribution in situations where there are a limited number ofmachines.

Instructions

To create a semi-remote OSE configuration, do the following:

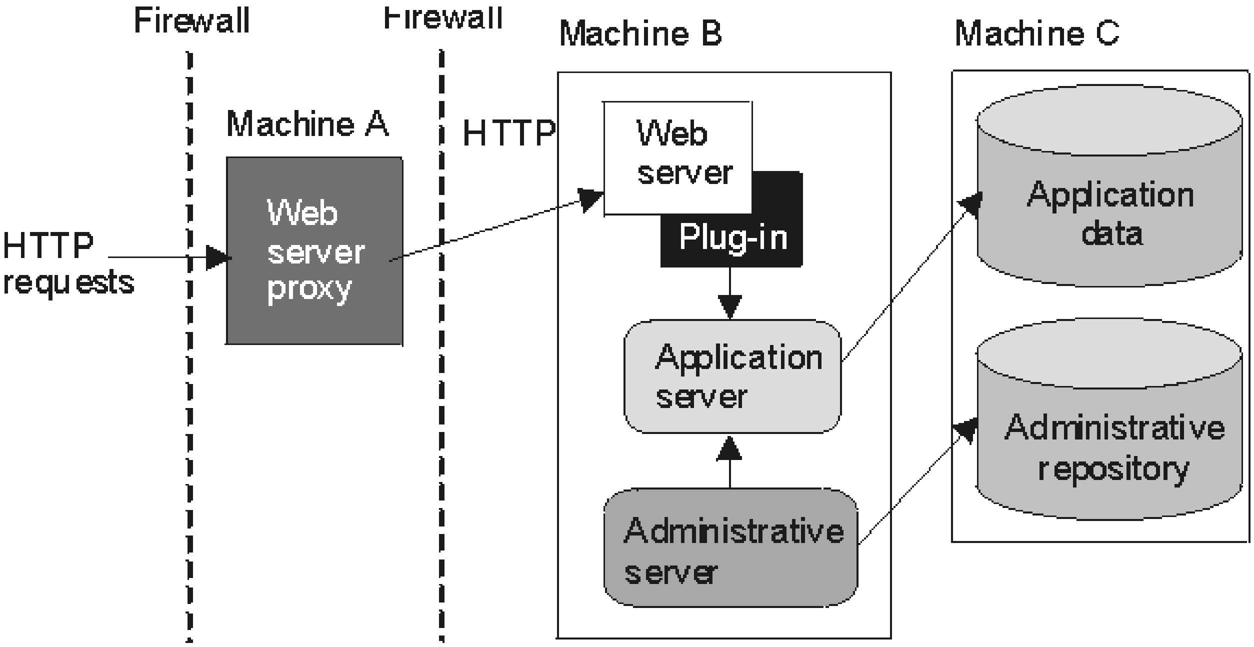
1. Create a remote OSE configuration, following the steps in article 7.1.3.6.1, Remote OSE sample topology.
2. Copy the *product_installation_root*/temp/queues.properties file from the machine where the application server runs (Machine B) to the *product_installation_root*/temp directory on the machine where the Web server runs (Machine A).
3. Modify the the queues.properties file to identify both machines as remote machines. The following is example of a modified queues.proerties file; the changed lines are highlighted in bold:
   ```
   # IBM WebSphere Plugin Communication Queues
   ```
   ```
   #Friday March 23 18:05:13 PDF
   2001ose.srvgrp.ibmoselink.clonescount=2ose.srvgrp=ibmoselinkose.srvgrp.ibmoselink.type=FASTLINKose.srvgrp.ibmoselink.clone1.port=8110ose.srvgrp.ibmoselink.clone1.type=remoteose.srvgrp.ibmoselink.clone1.host=was-nt1#ose.srvgrp.ibmoselink.clone2.port=8110ose.srvgrp.ibmoselink.clone2.type=remoteose.srvgrp.ibmoselink.clone2.host=was-nt2
   ```
4. Copy the modified queues.properties file, the rules.properties file, and the vhosts.properties file to another directory, for example, *product_installation_root*/temp/http. Copying these files to a different directory prevents the administrative server from overwriting them.
5. Create a new directory under the *product_installation_root*/properties directory, for example, *product_installation_root*/properties/http.
6. Copy the *product_installation_root*/properties/bootstrap.properties file to the new directory you just created.
7. In the copied bootstrap.properties file, change the value of the ose.tmp.dir parameter to the *product_installation_root*/temp/http directory. (That is, the directory where you placed the modified queues.properties file, the rules.properties file, and the vhosts.properties file.)
   ```
   ose.tmp.dir=product_installation_root/temp/http
   ```
   This parameter tells the HTTP server plug-in where to find the modified properties files.
8. In the HTTP server configuration file *product_installation_root*/properties/httpd.conf, change the plug-in entry for the bootstrap.properties file to refer to the modified of the file:
   ```
   NcfAppServerConfig BootfFileproduct_installation_root/properties/http/bootstrap.properties
   ```
9. Restart the administrative servers, application servers, and HTTP server.

7.1.3.6.3: Reverse proxy (IP forwarding) sample topology

- Overview
- Typical use
- Instructions

Overview

Reverse proxy (or *IP-forwarding*) topologies use a reverse proxyserver to receive incoming HTTP requests and forward them to a Web server. The Web serverin turn forwards the requests to the application servers that do the actual processing.The following figure shows a simple reverse proxy topology.



In this example, a reverse proxy resides in a demilitarized zone (DMZ) between theouter and inner firewalls. It listens on an HTTP port (typically port 80) for HTTPrequests. The reverse proxy then forwards those requests to an HTTP server that resides onthe same machine as WebSphere Application Server. After the requests are fulfilled, theyare returned through the reverse proxy to the client, hiding the originating Web server.

Typical use

Reverse proxy servers are typically used in DMZ configurations to allow additionalsecurity between the public Internet and the Web servers (and application servers)servicing requests. A reverse proxy product used with WebSphere Application Server mustsupport Network Address Translation (NAT) and WebSphere security.

Reverse proxy configurations support high-performance DMZ solutions that require as fewopen ports in the firewall as possible. The reverse proxy capabilities of the Web serverinside the DMZ require as few as one open port in the second firewall (potentially two ifusing SSL - port 443).

The advantages of using a reverse proxy server in a DMZ configuration include thefollowing:

- The reverse proxy server does not need database access through the firewall.
- It supports WebSphere security and NAT firewalls.
- The basic reverse proxy configuration is well-known and tested in the industry, resulting in less customer confusion than other DMZ configurations.
- It is reliable and its performance is relatively fast.
- It eliminates protocol switching by using the HTTP protocol for all forwarded requests.
- It does not affect the configuration and maintenance of a WebSphere application.
- It uses only one firewall port (HTTP) for requests and responses.

This is also a disadvantage in some environments wheresecurity policies prohibit the same port or protocol being used for inbound and outboundtraffic across a firewall.

The disadvantages of using a reverse proxy server in a DMZ configuration include thefollowing:

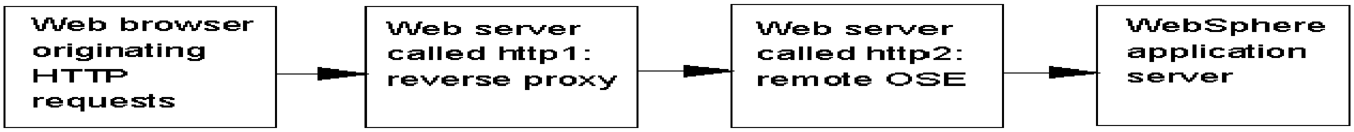
- The presence of a reverse proxy server in a DMZ might not be suitable for some environments.
- It requires more hardware and software than similar topologies that do not include a reverse proxy server, which makes it more complicated to configure and maintain.
- The reverse proxy server does not participate in WebSphere workload management.

Article 7.1.4, Firewall and demilitarized zone (DMZ)configurations, compares the reverse proxy topology to other topologies that support aDMZ configuration.

Instructions

The implementation specifics are determined by the reverse proxy server; refer to thedocumentation for the product you are using. No additional WebSphere administration isrequired for the reverse proxy server, although it can be needed for other elements of thereverse proxy topology.

The following figure shows how a reverse proxy server can be used with Remote OSE.


Web browser originating HTTP requests
Web server called http1: reverse proxy
Web server called http2: remote OSE
WebSphere application server

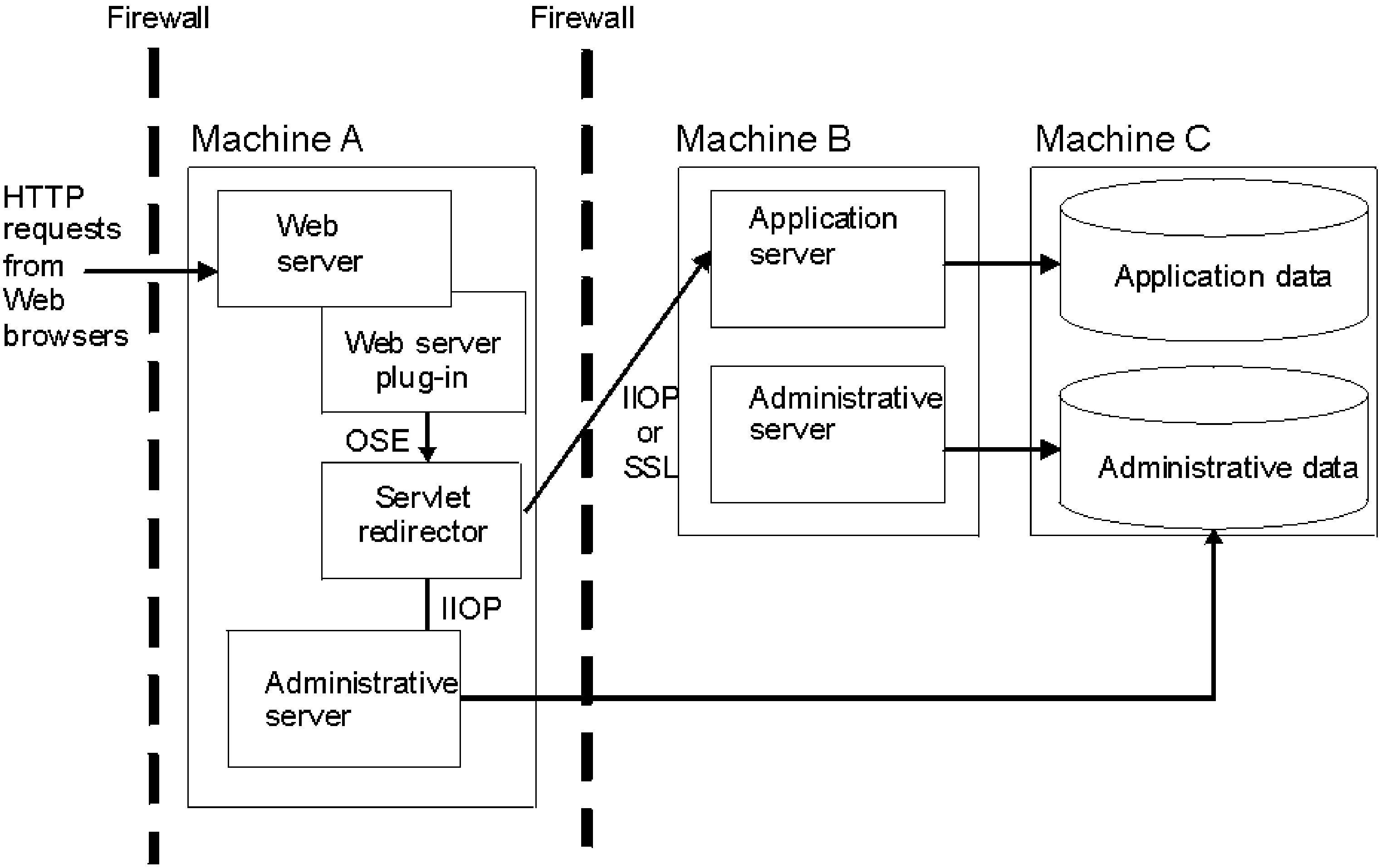
In this case, the reverse proxy server (located on host http1) passes requests to asecond Web server (located on host http2) that uses Remote OSE to forward the requests toan application server. The application server requires virtual host configurations for thephysical hosts http1 and http2, but not for the reverse proxy host name.

7.1.3.6.4: Thick servlet redirector sample topology

- Overview
- Typical use
- Instructions

Overview

In a *thick servlet redirector* configuration, the machine where the servletredirector runs is configured as a full WebSphere Application Server node with anadministrative server and its associated processes. The following figure shows a thickservlet redirector topology used with two firewalls.



The thick servlet redirector isinstalled on Machine A with the Web server. Requests are forwarded from the Webserver to the servlet redirector by using the Open Servlet Engine (OSE)transport.   The servlet redirector then forwards the requests to theapplication server on Machine B by using Remote Method Invocation (RMI) and InternetInter-ORB Protocol (IIOP). Encrypted requests can be forwarded by using the SecureSockets Layer (SSL) protocol.

Typical use

The thick servlet redirector topology has the following advantages:

- It supports encrypted communication with SSL between the servlet redirector and the application server.
- Because the thick servlet redirector includes a full administrative server, it can be configured and managed from WebSphere administrative clients. It also automatically updates the Web server plug-in files when administrative changes are made.
- A servlet redirector communicates with application servers through EJB client invocations and can participate in workload management. This allows it to forward HTTP requests to cloned application servers and provides load balancing and failover support.

It also has the following disadvantages:

- The administrative server requires access to the repository database, which is inapproprate for some secure environments. This requires an open port in the firewall for database communication.
- A database client must also be installed on Machine A. In addition to running another process on the machine, running a database client in an insecure environment is often inappropriate. A database ID and password must be stored on the machine, which can pose a security risk.
- It requires multiple ports in a firewall.
- It requires the firewall to support IIOP.
- It does not support Network Address Translation (NAT) firewalls.
- The thick servlet redirector performs relatively slowly compared to other servlet redirector mechanisms such as Remote OSE. The administrative server and database client processes use system resources that are needed by the Web server, which can negatively affect its performance.

The thick servlet redirector is often used in situations where an organization wants tomaintain a Web server in one department and secure applications in another. Theapplication server provides dynamic content (such as servlets and JSP files) to clientswith minimal maintenance requirements for the Web server machines. The thick servletredirector can be configured from the WebSphere administrative console, allowing users tomaintain it from any machine within a WebSphere Application Server installation.

Article 7.1.4, Firewall and demilitarized zone (DMZ)configurations, compares the thick servlet redirector topology to other topologiesthat support a DMZ configuration.

Instructions

The following instructions describe how to set up the configuration shown in theprevious figure, with the possibility of additional Web server machines (Machines D, E,... N) communicating with the application server on Machine B.

1. Install the product components:

	Web server	**Web server plug-in**	**administrative server**	**administrative console**	**application server**
Machine A	✓	✓	✓		
Machine B			✓	✓	✓
Machines D, E, ... N (optional)	✓	✓	✓		

   Machine C contains the administrative database for all of the above administrative servers, as well as the database for application data. The arrangement shown in the previous figure is just one option for database placement. For example, Machine C can be omitted if the database on one of the machines already in the configuration (A or B).
2. Start the administrative servers on each machine.
3. Configure the virtual hosts:
   a. Start the Java administrative client (WebSphere Administrative Console).
   b. Display the Topology view.
   c. Expand the WebSphere Administrative Domain, verifying that all machines sharing the administrative database (in this case, Machines A, B, D, E ... N) are displayed as administrative nodes.
   d. In the tree, locate and click the default virtual host to display its properties on the right side of the console.
   e. In the Advanced properties, add host names (and ports, if the port number is other than port 80) for the Web servers running on Machines A, D, E, ... N. Save the changes.
   f. Locate the application server under Machine B in the Topology tree and start it. If Machine A does not contain an application server, configure a new one and start it.
4. Configure and test the thick servlet redirector on Machine A:
   a. Use the Java administrative client (WebSphere Administrative Console) to display the Topology view.
   b. Expand the WebSphere Administrative Domain, verifying that all machines sharing the administrative database (in this case, Machines A, B, D, E ... N) are displayed as administrative nodes.
   c. Locate and right-click the node representing the machine on which the thick redirector runs. A menu is displayed.
   d. On the menu, click **Create -> Servlet Redirector** to display servlet redirector properties.
   e. Specify the properties, then save them.
   f. Locate and right-click the newly created thick servlet redirector in the Topology tree to display a menu.
   g. On the menu, click **Enabled**.
   h. From the same menu, click **Start** to start the thick servlet redirector process on this machine. Wait for it to start.
   i. Start the Web server on the same machine.
   j. Verify that the thick servlet redirector is running.
   k. Verify that the thick servlet redirector is set up correctly.

      Start a Web browser. Enter a URL that is valid for Machine B, but send it to the Web server machine (Machine A).

      For example, if the Machine B configuration defines a path of /servlet/snoop for accessing the Snoop diagnostic servlet on Machine B, use this path as part of the URL to access the Snoop servlet on the Web server machine. Type:

      ```
      http://hostname/servlet/snoop
      ```

      where *hostname* is a valid host name for Machine A.
5. Repeat the previous step for each thick servlet redirector node (Machines D, E, ...N).

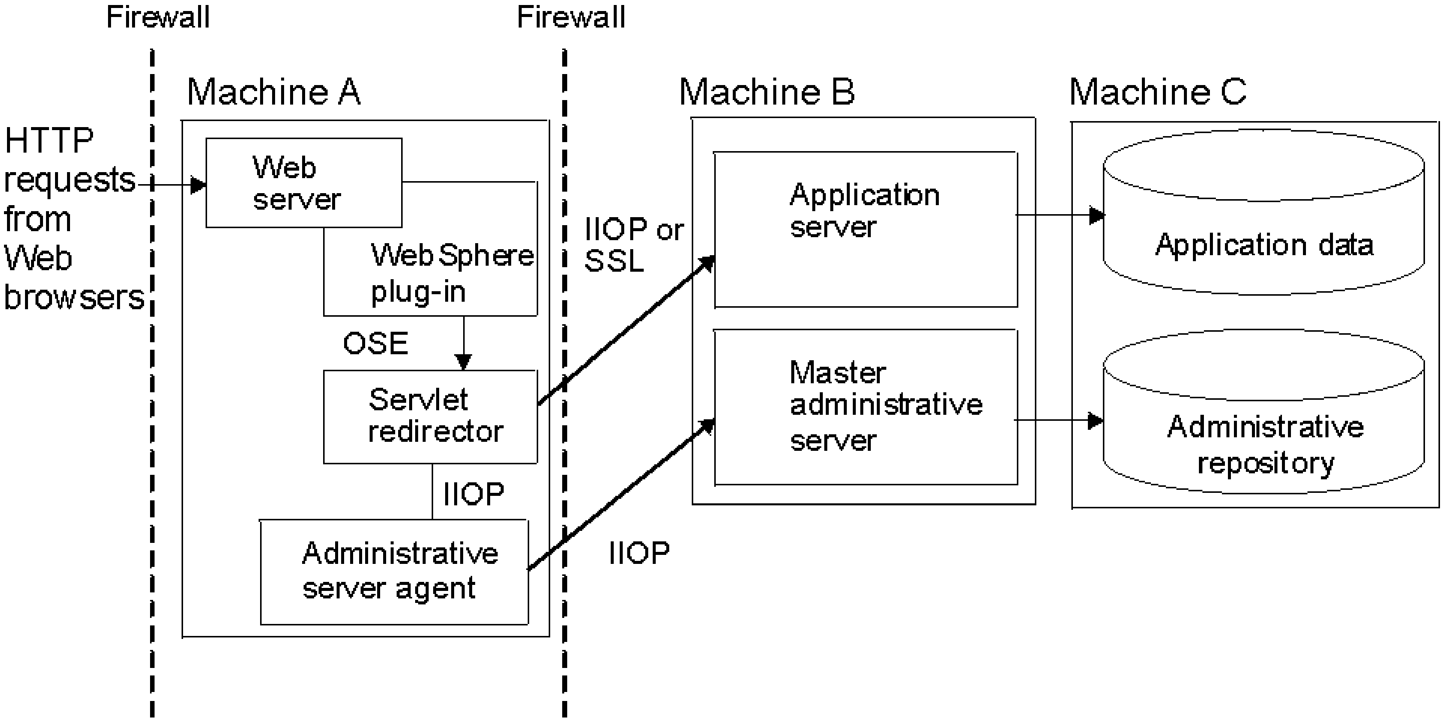
7.1.3.6.5: Thick servlet redirector with administrative agent sample topology

- Overview
- Typical use
- Instructions

Overview

In a configuration for a *thick servlet redirector with administrativeagent*, an administrative server agent providesadministrative support for the servlet redirector. An agent is controlled by a fullinstance of an administrative server located on another machine. It provides all ofthe services of an administrative server, but does not require a direct connection to theadministrative repository database. The agent can access the administrative server througha firewall.

The following figure shows a thick servlet redirector that is configured to run with anadministrative server agent.



The thick servlet redirector isinstalled on Machine A with the Web server. Requests are forwarded from the Webserver to the servlet redirector by using the Open Servlet Engine (OSE)transport. The servlet redirector then forwards the requests to the applicationserver on Machine B by using Remote Method Invocation (RMI) and Internet Inter-ORBProtocol (IIOP). Encrypted requests can be forwarded by using the Secure SocketsLayer (SSL) protocol.

The administrative agent on Machine A communicates with the master administrativeserver on Machine B. The master administrative server handles all communication withthe administrative repository database on Machine C. This eliminates the need toinstall a database client on Machine A. To retrieve configuration information, theadministrative agent communicates by using RMI/IIOP through the firewall.

Typical use

The advantages of using a thick servlet redirector with an administrative agent are asfollows:

- It does not require database access through a firewall, making it appropriate for use in many secure demilitarized zone (DMZ) configurations.
- Communications between the servlet redirector and the remote application server can be encrypted by using SSL.
- It does not require a database client on the Web server machine.
- The administrative server agent can be easily configured though the administrative console, making maintenance easier.
- A servlet redirector communicates with application servers through EJB client invocations and can participate in workload management. This allows it to forward HTTP requests to cloned application servers and provides load balancing and failover support.
- It supports WebSphere Application Server product security.
- Because it has fewer associated processes (no database client and administration through an agent instead of a full administrative server), this topology generally has better Web server performance than the thick servlet redirector.

The disadvantages of this configuration are as follows:

- The remote administrative server can potentially be a single point of failure. However, running administrative servers with workload management enabled eliminates this possibility.
- It does not support Network Address Translation (NAT) firewalls.
- It requires multiple ports in a firewall.

- It requires the firewall to support IIOP.
- It performs relatively slowly in a DMZ configuration compared to other servlet redirection mechanisms such as Remote OSE.

The thick servlet redirector with administrative agent is often used when anorganization wants to maintain a Web server in one department and secure applications inanother. The application server provides dynamic content (such as servlets and JSP files)to clients with minimal maintenance requirements for the Web server machines. It alsoisolates the back-end database from the Web server.

Article 7.1.4, Firewall and demilitarized zone (DMZ)configurations, compares the thick servlet redirector with administrative agenttopology to other topologies that support a DMZ configuration.

Instructions

In order for an administrative agent to be set up, an administrative server must beinstalled and running. Record the host name of the administrative server. To set up anadministrative server as an administrative agent:

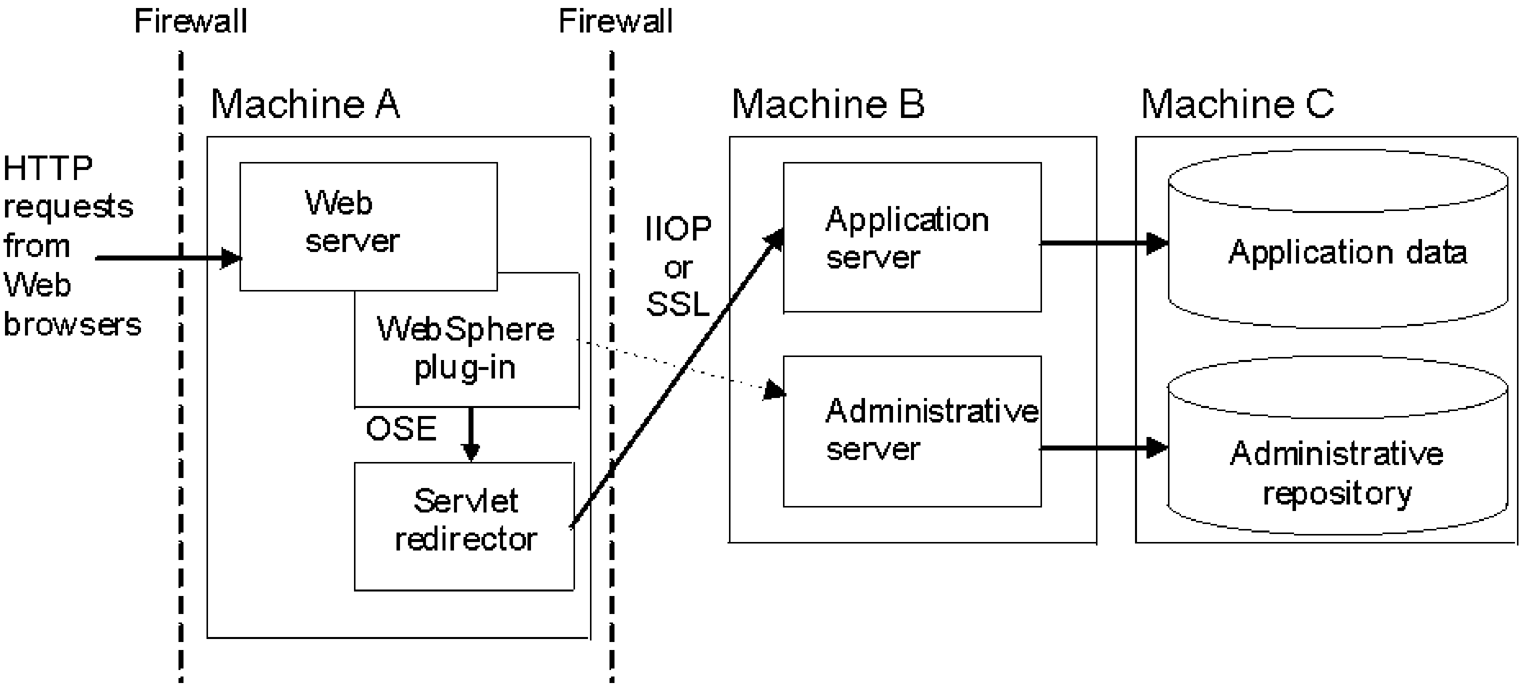
1. Install WebSphere Application Server. During the installation, do not create a default configuration.
2. After installation, change to the in the *product_installation_root*/bin directory and make a backup copy of the admin.config file.
3. Open the original `admin.config` file and delete the following lines:
   ```
   com.ibm.ejs.sm.adminServer.nameServiceJar=jar_filecom.ibm.ejs.sm.adminServer.dbUser=user_namecom.ibm.ejs.sm.adminServer.dbUrl=locationcom.ibm.ejs.sm.adminServer.dbPassword=password
   ```
4. Add the following lines, where *host_name* is the host name of the administrative server.
   - These lines are required:
     ```
     com.ibm.ejs.sm.adminServer.bootstrapHost=host_namecom.ibm.ejs.sm.adminServer.primaryNode=host_name
     ```
   - These lines are required if the administrative server is using nondefault port numbers.   Both nodes must use the same values.
     ```
     com.ibm.ejs.sm.adminServer.bootstrapPort=port_numbercom.ibm.ejs.sm.adminServer.lsdPort=port_number
     ```
   - These lines are optional:
     ```
     com.ibm.ejs.sm.adminServer.lsdHost=host_namecom.ibm.CORBA.ListenerPort=port_number
     ```
5. Save your changes and close the file.
6. Start the WebSphere administrative server on Machine B.
7. Start the administrative server agent on Machine A.
8. Start the administrative console on Machine B and follow the directions for configuring virtual host aliases and the thick servlet redirector, as described in article 7.1.3.6.4, Thick servlet redirector sample topology.

7.1.3.6.6: Thin servlet redirector sample topology

-
- Typical use
- Instructions

Overview

In a *thin servlet redirector* configuration, a stand-alone version of theservlet redirector runs on the Web server machine. The following figure shows anexample of a thin servlet redirector being used with firewalls.



The thin servlet redirector isinstalled on Machine A with the Web server. Requests are forwarded from the Webserver to the servlet redirector by using the Open Servlet Engine (OSE) transport. The servlet redirector then forwards the requests to the application server on Machine Bby using Remote Method Invocation (RMI) and Internet Inter-ORB Protocol (IIOP). Encrypted requests can be forwarded by using the Secure Sockets Layer (SSL)protocol.

No administrative server is installed on the Web server machine. Instead, scripts are used to configure the Web server plug-in, start the servlet redirector, and stop theservlet redirector. The Web server plug-in files must be manually generated.

If WebSphere security is being used to secure the HTML files on your Web server, theWeb server plug-in also connects to an administrative server

Typical use

The advantages of the thin servlet redirector configuration are as follows:

- It does not require database access through a firewall, making it appropriate for use in many secure demilitarized zone (DMZ) configurations.
- Communication between the servlet redirector and the remote application server can be encrypted by using SSL.
- It does not require a database client on the Web server machine.
- A servlet redirector communicates with application servers through EJB client invocations and can participate in workload management. This allows it to forward HTTP requests to cloned application servers and provides load balancing and failover support.
- It supports WebSphere Application Server product security.
- Because it has fewer associated processes (no database client, administrative server or administrative server agent), this topology generally has better Web server performance than the other thick servlet redirector configurations in non-DMZ configurations.

The disadvantages of this configuration are as follows:

- It does not support Network Address Translation (NAT) firewalls.
- It requires the firewall to support IIOP.
- It performs relatively slowly in a DMZ configuration compared to other servlet redirection mechanisms such as Remote OSE.

Article 7.1.4, Firewall and demilitarized zone (DMZ)configurations, compares the thin servlet redirector topology to other topologies thatsupport a DMZ configuration.

Instructions

The instructions describe how to set up the configuration shown in the previous figure.

1. Install the appropriate product components:

	Web server	Web server plug-in	administrative server	administrative console	application server
Machine A	✓	✓	✓	✓ (optional)	
Machine B			✓	✓	✓
Machines D, E, ...N			✓	✓	✓

Machine B must contain the default resources in order to obtain the Admin Web application referred to in the instructions for this sample topology.

Machines D through N are additional, optional application server machines onto which the application server on Machine B can be cloned. Install the default resources only on Machine

B, cloning them to Machines D, E, ... N if applicable.

Machine C contains the administrative database for all of the above administrative servers, as well as the database for application data. The arrangement shown in the previous figure is just one option for database placement. For example, Machine C can be omitted if the database resides on Machine B, D, E or another machine.

Although it is not required, having a Web server installed on the administrative server machine is useful for verification testing.

2. Test the installation on the application server machine:
    a. Start the administrative server.
    b. Start the administrative console.
    c. Start the application server. If necessary, create a new application server.
    d. From the administrative console, add the host name of the Web server machine (Machine A) to the alias list of the virtual host containing the Admin Web application on Machine B:
        i. Locate and click the default host in the Topology tree view to display its properties in the right side of the console.
        ii. In the advanced properties list, add the host name and IP address of the Web server machine to the alias list of the virtual host.
        iii. Add the machine name and IP address as separate entries in the Host Aliases table. If the Web server port is not 80, append the port to these entries.
        iv. Save your changes.
    e. Stop and restart the application servers running under the virtual host on Machine B. For each application server:
        i. In the Topology tree view, locate the application server.
        ii. Right-click it, displaying its menu. Click **Stop**.
        iii. Watch for the administrative console message that the application server was stopped successfully.
        iv. Right-click the server to display its menu again.
        v. Click **Start**.
        vi. Ensure the application server on Machine B is running. Use a browser to view the following:

           ```
           http://machine_B_host_name/servlet/snoop
           ```

           where *machine_B_host_name* is a valid host name for Machine B.
3. Test the installation on Machine A:
    a. Start the thin servlet redirector.
        i. Use a text editor to open the following file:

           ```
           product_installation_root/properties/iiopredirector.xml
           ```

           Change all occurrences of localhost to the host name of Machine B.
        ii. Open a command window and change directory to:

           ```
           product_installation_root/bin
           ```
        iii. Run the thin servlet redirector configuration script:
            - On Windows-based systems, type:

              ```
              thinRedirectorConfig -adminNodeName Machine_B_hostname
              ```
            - On UNIX-based systems, type:

              ```
              thinRedirectorConfig.sh -adminNodeName Machine_B_hostname
              ```

              You must be logged in as root to run the thin servlet redirector configuration script on the AIX platform.

           The program generates three properties files in the directory:

           ```
           product_installation_root/temp
           ```

           There are optional arguments for the scripts, such as:
            - -serverRoot *product_installation_root*
            - -nameServiceNodeName *application server machine host name*
            - -queueProps *product_installation_root*/properties/iiopredirector.xml

           To view a complete list of arguments for the configuration script, change the directory to the *product_installation_root*/bin directory and type the following at the command prompt:
            - On Windows-based systems, type:

              ```
              thinRedirectorConfig
              ```
            - On UNIX-based systems, type:

              ```
              ./thinRedirector.sh
              ```
        iv. Run the thin redirector start script:
            - On Windows-based systems, type:

              ```
              thinRedirectorStart
              ```
            - On UNIX-based systems, type:

              ```
              thinRedirectorStart.sh
              ```
4. Make sure that all of the following conditions are true:
    - The **thinRedirectorConfig** script ran and stopped successfully.
    - The **thinRedirectorStart** script is still running.
    - The application server on Machine B is running.
5. Confirm that the Web server is running, and was started after running the scripts and starting the application server. Use a Web browser to open the following:

   ```
   http://Machine_B_hostname/servlet/snoop
   ```
6. Whenever the configuration on Machine B changes, you need to:
    1. Run the **thinRedirectorConfig** and **thinRedirectorStart** scripts again to regenerate the Web server plug-in configuration files.
    2. Stop the Web server and start it again.

Installing additional Web servers with different ports

These instructions describe how to add another Web server on Machine A or anothermachine and configure it to send requests to Machine B (the application server machine).The details of each step were described previously under Thin servlet redirectorconfiguration procedure.

1. If you are adding a new machine, install the components that you installed on Machine A.
2. Install the Web server and plug-in.
3. In the virtual host settings on Machine B (application server machine), define the Web server host name and IP address and associated ports as virtual host aliases, as you did for Machine A.
4. Stop the administrative server and console on Machine B and start them again.
5. Run the **thinRedirectorConfig** and **thinRedirectorStart** scripts.
6. Start the new Web server on the new machine.
7. At a Web browser, verify the configuration by accessing the following:

   `http://new_machine_hostname/servlet/snoop`

Accessing clones

To use the servlet redirector to access Machines D, E, ... N, which contain clones ofthe application server on Machine B, when you set up the thin redirector machine (MachineA):

- Specify the machine's host name in the file iiopredirector.xml.
- Specify the machine's host name as an argument to the **thinRedirectorConfig** command.

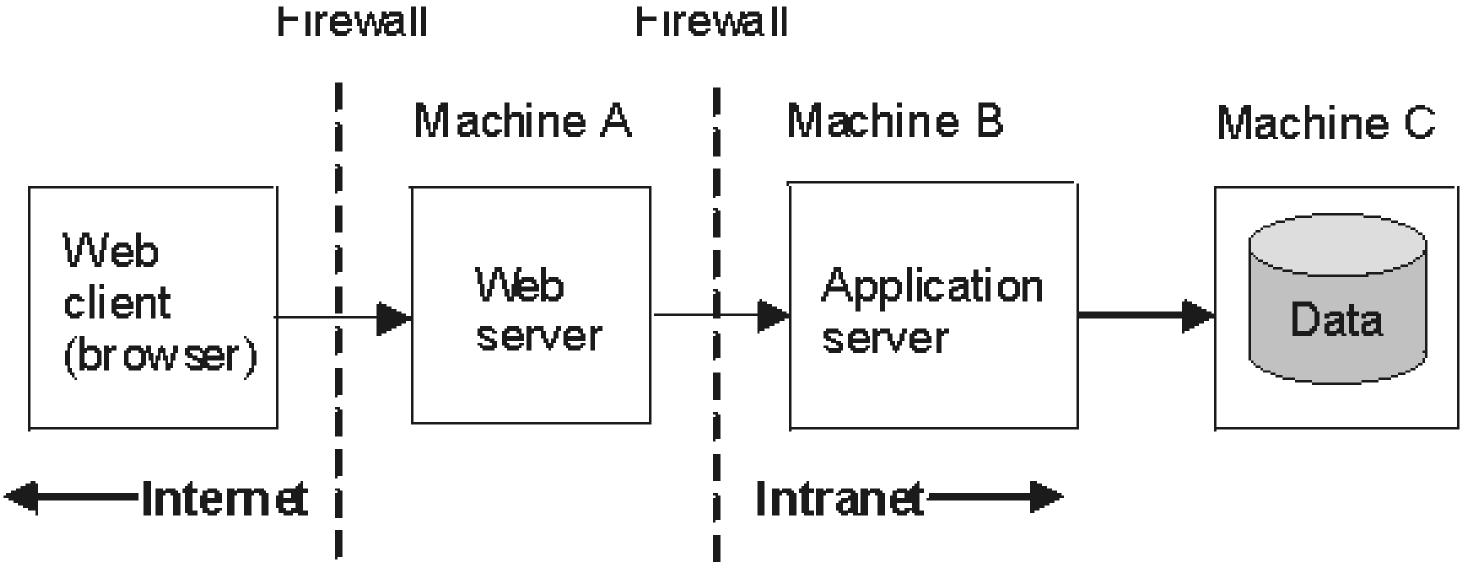
Using with firewalls (DMZ configuration)

You need to open the following ports in the firewall:

- The RMI/IIOP port
- The Location Service Daemon (LSD) port
- The bootstrap port

7.1.3.7: Demilitarized zone (DMZ) sample topology

A *demilitarized zone* (DMZ) configuration involves multiple firewalls that addlayers of security between the Internet and a company's critical data and business logic.The following figure shows an example of a simple DMZ topology.



The main purpose of a DMZ configuration is to protect the business logic and data inthe environment from unauthorized access. A typical DMZ configurationincludes:

- An outer firewall between the public Internet and the Web server or servers processing the requests originating on the company Web site.
- An inner firewall between the Web server and the application servers to which it is forwarding requests. Company data also resides behind the inner firewall.

The area between the two firewalls gives the DMZ configuration its name. Additionalfirewalls can further safeguard access to databases holding administrative and applicationdata.

DMZ configurations can be implemented for a wide variety of multi-tiered systems. Article 7.1.4, Firewall and demilitarized zone configurations,compares some DMZ configuration options and can help you to select which one is right foryour organization.

Typical use

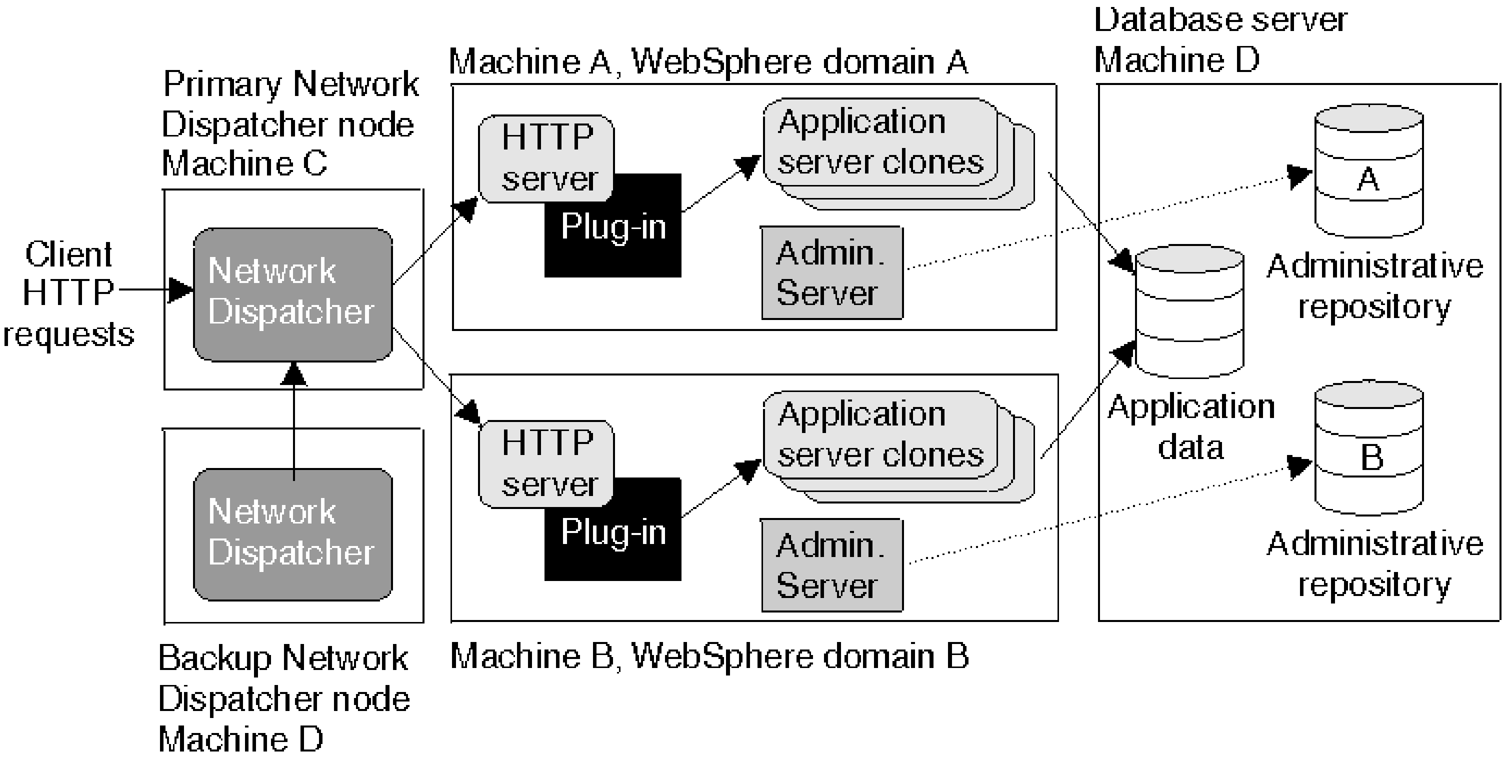
The advantage of using a DMZ topology is heightened security. Its drawbacks are morecomplex administration and maintenance. In addition, an administration server often cannotbe run on the DMZ node. The firewall is intended to protect the back-end database serverfrom unauthorized access, but it can prevent the administrative server from gaining accessto the administrative repository.

7.1.3.8: Multiple WebSphere domains sample topology

- Overview
- Typical use

Overview

The following figure shows an example of how an application can be implemented overmultiple WebSphere Application Server administrative domains.


Primary Network Dispatcher node Machine C
Client HTTP requests
Network Dispatcher
Network Dispatcher
Backup Network Dispatcher node Machine D
Machine A, WebSphere domain A
HTTP server
Plug-in
Application server clones
Admin. Server
Machine B, WebSphere domain B
HTTP server
Plug-in
Application server clones
Admin. Server
Database server Machine D
A
Administrative repository
Application data
B
Administrative repository

The example application runs simultaneously in two administrative domains, each hostedon a different physical machine (Machines A and B). Network Dispatcher is used todistribute incoming HTTP requests among the two domains, presenting a single image of theapplication to clients. A backup Network Dispatcher node provides failover support.

In this example, the application server clones in both domains are created from thesame model so that identical versions of the application run in each domain. However, youcan run a different version of the application in each domain. Because the domains areisolated from one another, you can also run different versions of the WebSphereApplication Server software in each domain.

In this example, both domains share a common application database. However, each domainis administered independently and maintains a separate administrative repository.

Typical use

Topologies that incorporate more than one administrative domain have the followingadvantages:

- Isolation of hardware failure. If one domain goes offline due to hardware problems, the others can still process client requests.
- Isolation of software failure. Running an application in two or more domains isolates any problems that occur within a domain, while the other domains continue to handle client requests. This can be helpful in a variety of situations:
  - When rolling out a new application or a revision of an existing application. The new application or revision can be brought online in one domain and tested in a live situation while the other domains continue to handle client requests.
  - When deploying a new version of the WebSphere Application Server software. The new version can be brought into production and tested in a live situation without interrupting service.
  - When applying fixes or patches to the WebSphere Application Server software. Each domain can be taken offline and upgraded without interrupting the application.

  If an unforeseen problem occurs with the new software, using multiple domains can prevent an outage to an entire site. A rollback to a previous software version can also be accomplished more quickly. Hardware and software upgrades can be handled on a domain-by-domain basis during offpeak hours.
- Improved performance. Running an application using multiple smaller domains can provide better performance than a single large domain because there is less interprocess communication in a smaller domain.

Using multiple domains has several drawbacks:

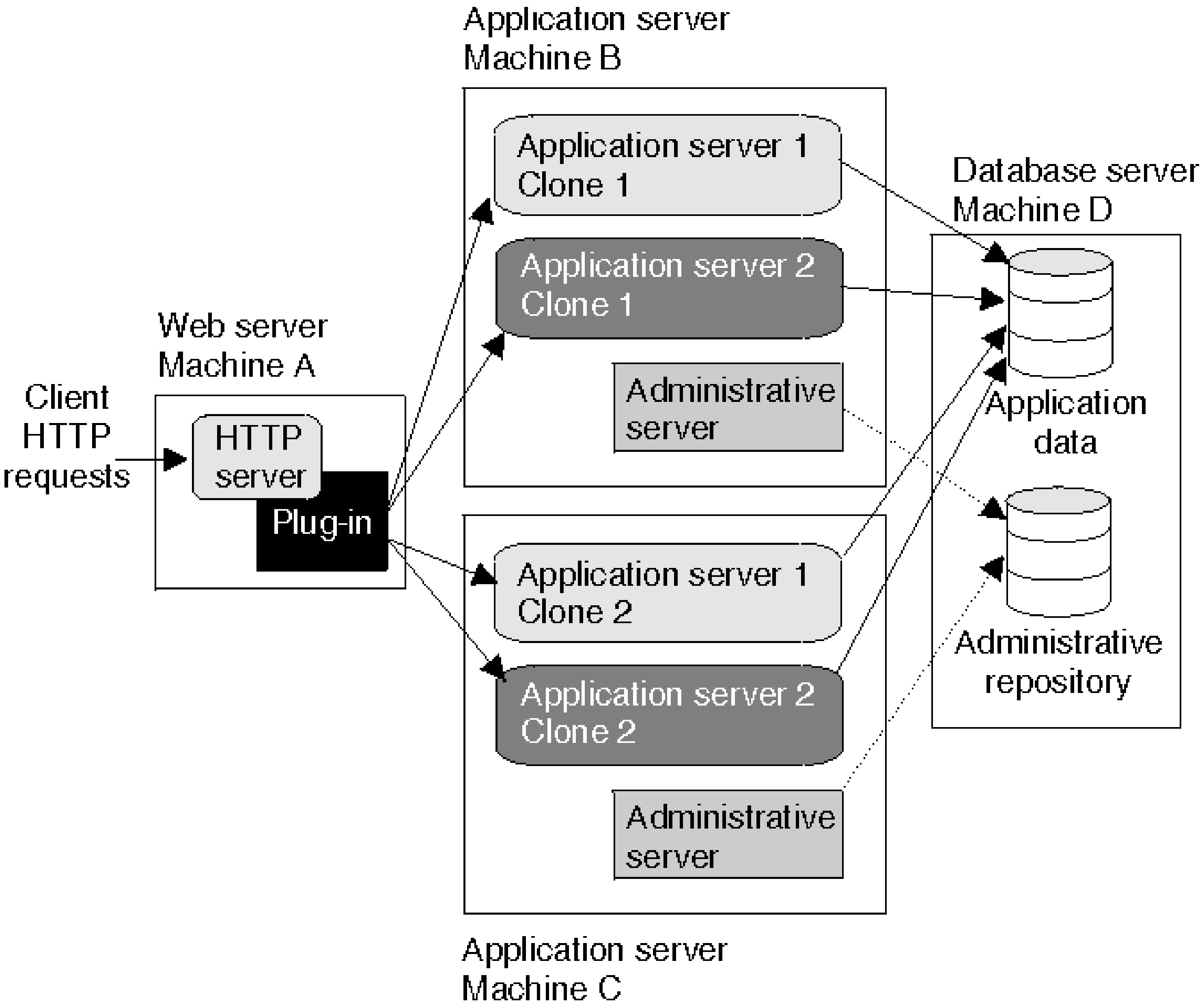
- Deployment is more complicated than for a single administrative domain. Using a distributed file system that provides a common file mount point can make this task easier.
- Multiple domains require more administration effort because each domain is administered independently. This problem can be reduced by using **wscp** and **XMLConfig** scripts to standardize and automate common administrative tasks.
- Using multiple administration repositories (databases) makes performing backups more complicated.

7.1.3.9: Multiple applications within a node sample topology

- Overview
- Typical use

Overview

The following figure shows a topology in which clones of more than one applicationserver are hosted on a physical node.



The example topology is a variation of the basic horizontal scaling topology. Theclones of an application server are not hosted on a single machine but are distributedthroughout all of the machines in the system. (In this example, a clone of each ishosted on both Machine B and Machine C.) Machine A serves as the Web server for theapplication and distributes client requests to the application server clones on eachnode. Machine D serves as the database server for both nodes.

Typical use

Hosting clones of multiple application servers within a node provides the followingbenefits:

- Improved throughput. Cloning an application server enables it to handle more client requests simultaneously.
- Improved performance. Hosting clones on multiple machines enables each clone to make use of the machine's processing resources.

- Hardware failover. Hosting clones on multiple nodes isolates hardware failtures and provides failover support. Client requests can be redirected to the application server clones on other nodes if one node goes offline.
- Application server failover. Hosting clones on multiple nodes also isolates application software failures and provides failover support if a clone stops running. Client requests can be redirected to clones of the application server on other nodes.
- Process isolation. If one application server process fails, its clones on the other nodes are unaffected.

Drawbacks of this topology include the following:

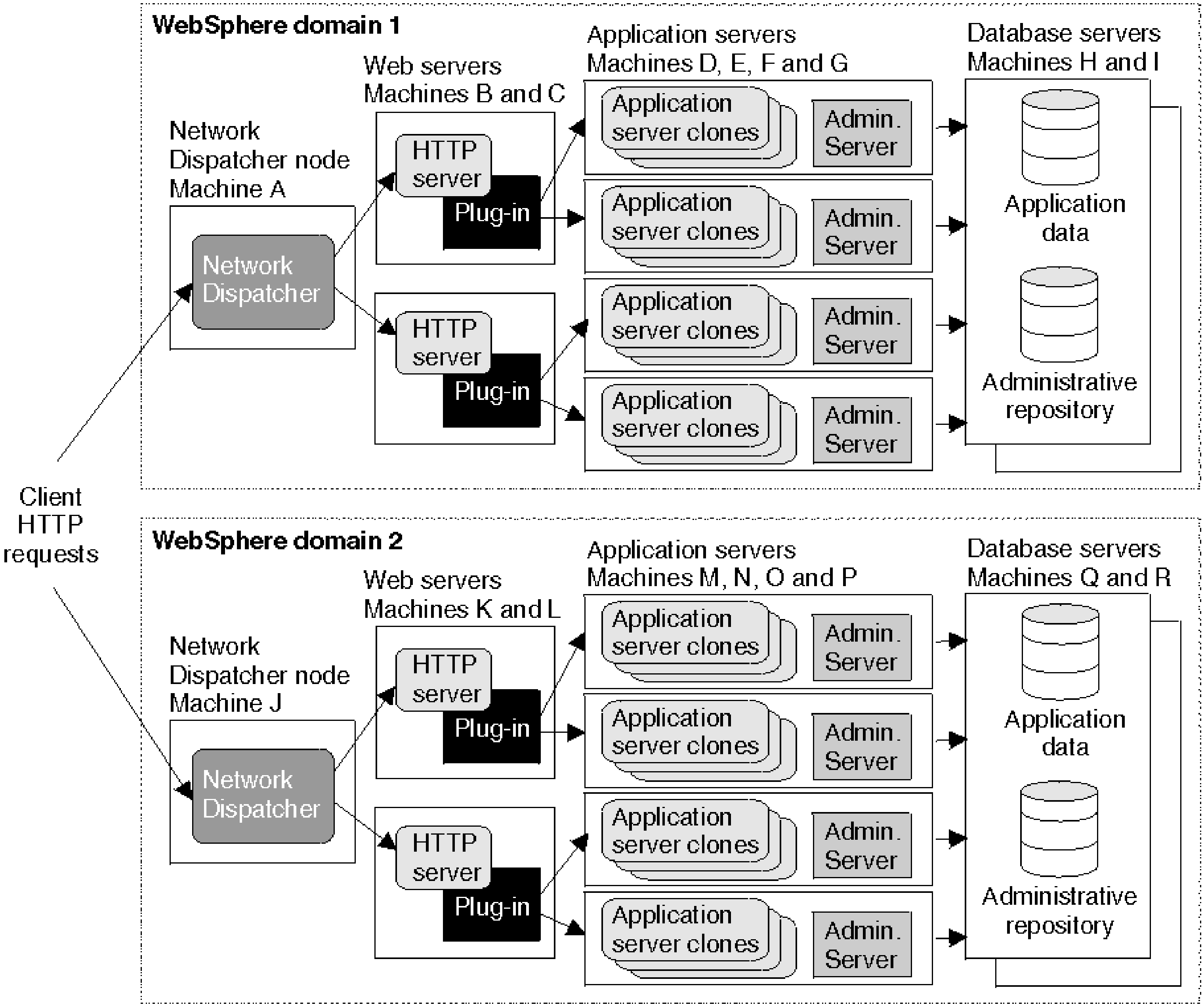
- More complex deployment. Application executable files must be distributed across multiple machines in a cluster. Using a distributed file system that provides a common file mount point for all nodes can make this task easier.
- More complex maintenance. Clones of each application server must be maintained on multiple machines.

7.1.3.10: Putting it all together - a combined topology

- Overview
- Typical use

Overview

An example of a topology that combines the best elements of the other topologiesdiscussed in this section is shown in the following figure.



WebSphere domain 1
Network Dispatcher node Machine A
Network Dispatcher
Web servers Machines B and C
HTTP server
Plug-in
HTTP server
Plug-in
Application servers Machines D, E, F and G
Application server clones
Admin. Server
Application server clones
Admin. Server
Application server clones
Admin. Server
Application server clones
Admin. Server
Database servers Machines H and I
Application data
Administrative repository
Client HTTP requests
WebSphere domain 2
Network Dispatcher node Machine J
Network Dispatcher
Web servers Machines K and L
HTTP server
Plug-in
HTTP server
Plug-in
Application servers Machines M, N, O and P
Application server clones
Admin. Server
Application server clones
Admin. Server
Application server clones
Admin. Server
Application server clones
Admin. Server
Database servers Machines Q and R
Application data
Administrative repository

This topology combines elements of several different basic topologies:

- Two WebSphere Application Server administrative domains
- Two Network Dispatcher nodes (machine A in domain 1; machine J in domain 2)
- Two HTTP servers for each domain (machines B and C in domain 1; machines K and L in domain 2)
- Four application server nodes for each domain (machines D, E, F, and G in domain 1; machines M, N, O and P in domain 2)
- The use of clones for both vertical and horizontal scaling. In the example topology, each node hosts three clones; in practice, the number of clones is limited by the computing resources of each node.
- Two database servers for each domain (machines H and I in domain 1; machines Q and R in domain 2). These servers host mirrored copies of the application database and administrative database.

Typical use

This topology is designed to maximize thoughput, availability, and performance. It incorporates the best practices of the other topologies discussed in this section:

- Having more than one Network Dispatcher node, HTTP server, application sever, and database server in each domain eliminates single points of failure.
- Multiple administrative domains provide both hardware and software failure isolation, especially when upgrades of the application or the application server software are rolled out. (Hardware and software upgrades can be handled on a domain-by-domain basis during off-peak hours.)
- Horizontal scaling is done by using both cloning and Network Dispatcher to maximize availability and eliminate single points of process and hardware failure.

- Application performance is improved by using several techniques:
    - Hosting application servers on multiple physical machines to boost the available processing power.
    - Creating multiple smaller domains instead of one large domain. There is less interprocess communication in a smaller domain, which allows more resources to be devoted to processing client requests.
    - Using clones to vertically scale application servers on each node, which makes more efficient use of the resources of each machine.
- Applications with this topology can make use of several workload management techniques. In this example, workload management can be done through one or more of the following:
    - Using the Advanced Application Server workload management facility to distribute work among the application server clones.
    - Using Network Dispatcher to distribute client HTTP requests to each Web server.
    - Using the servlet redirector to distribute requests from the Web server to servlets running in the various application servers.
    - Using Remote OSE, which provides a simple workload distribution facility.

  For example, an application can manage workloads at the Web server level with Network Dispatcher and at the application server level with WebSphere workload managment. Using multiple workload management techniques in an application provides finer control of load balancing.

  Regardless of which workload management techniques are used in the application, administrative servers participate in workload management to provide failover support.

In this topology, only the loss of an entire domain can normally be noticed by users.If this occurs, the active HTTP sessions are lost for half of the clients. Thesystem can still process HTTP requests although its perfomance is degraded.

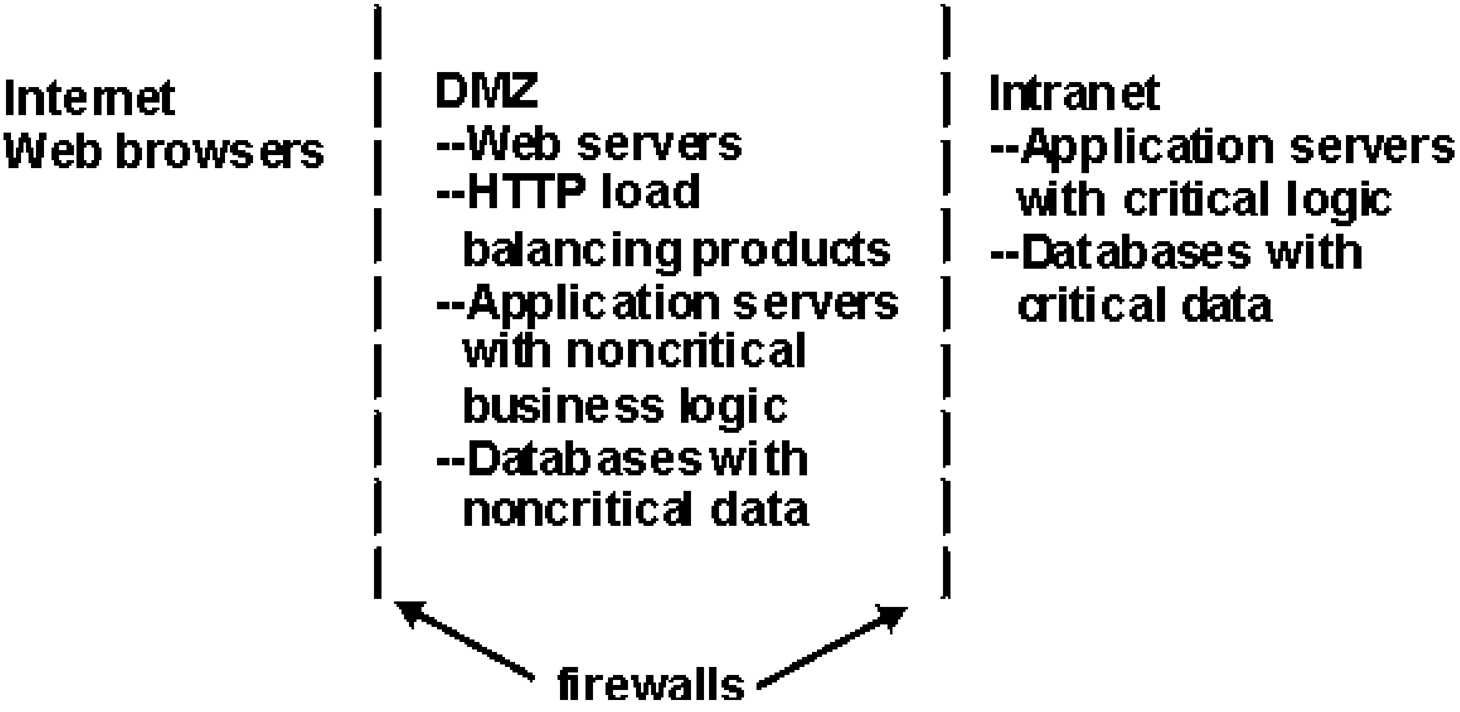
The combined topology has several drawbacks:

- Deployment is more complicated. The WebSphere Application Server software and application files must be deployed in each domain, which would not be the case for applications that run in a single administrative domain. Using a distributed file system that provides a common file mount point can make this task easier.
- Multiple domains require more administration effort, since each domain is administered independently. This problem can be reduced by using **wscp** and **XMLConfig** scripts to standardize and automate common administrative tasks.

7.1.4: Firewalls and demilitarized zone (DMZ) configurations

Firewalls are often used in multimachine systems to protect back-end resources such asdatabases. They can also be used to protect application servers and even Web servers fromunauthorized outside access.

A *demilitarized zone* (DMZ) configuration involves multiple firewalls that addlayers of security between the Internet and a company's critical data and business logic.A wide variety of topologies are appropriate for a DMZ environment. WebSphere ApplicationServer provides great flexiblity in configuring DMZ topologies, but the basic locations ofelements are as follows:


Internet
Web browsers
DMZ
--Web servers
--HTTP load balancing products
--Application servers with noncritical business logic
--Databases with noncritical data
Intranet
--Application servers with critical logic
--Databases with critical data
firewalls

Comparison of DMZ configurations

Somehow, requests for applications being managed by WebSphere Application Server mustget from the Web server to the application servers, passing through firewalls. WebSphereApplication Server offers many configuration choices for accomplishing this goal. Thefollowing table summarizes the benefits of each DMZ configuration option supported by theproduct. The criteria for each topology are described after the table.

A checkmark (✓) represents anadvantage. Remote OSE and reverse proxy are the recommended configurations, but you willstill need to evaluate whether one of them suits your particular environment better thanthe other options.

Benefit (✓) or statistic	**Remote OSE**	**Thick servlet redirector**	**Thin servlet redirector**	**Administrative agent**	**Reverse proxy**
Compatible with product security	✓	✓		✓	✓
Avoids data access from DMZ	✓		✓	✓	✓
Supports NAT	✓				✓
Avoids DMZ protocol switch					✓

Allows encrypted link between Web server and application server		✓	✓	✓	Depends on Web server
Avoids single point of failure	✓	✓			
Minimum firewall holes	1 per application server, plus 1 if WebSphere security is used on the Web server machine, plus 1 if the OSERemoteConfig script is used to configure remote OSE on the Web server machine	3, plus 1 per application server	3, plus 1 per application server	3, plus 1 per application server	1

- **Compatible with product security.** IBM WebSphere Application Server security protects applications and their components by enforcing authorization and authentication policies. Configuration options compatible with product security are desirable because they do not necessitate alternative security solutions.
- **Avoids data access from DMZ.** A DMZ configuration protects application logic and data by creating a demilitarized zone between the public Web site and the servers and databases where this valuable information is stored. Desirable DMZ topologies do not have databases or servers that directly access databases in the DMZ. Because a WebSphere administrative server needs access to a database for its configuration information, it is often not a viable solution to run an administrative server in the DMZ.
- **Supports Network Address Translation (NAT).** A firewall product that runs NAT receives packets for one IP address, and translates the headers of the packet to send the packet to a second IP address. In environments with firewalls employing NAT, avoid configurations involving complex protocols in which IP addresses are embedded in the body of the IP packet, such as Java Remote Method Invocation (RMI) or Internet Inter-Orb Protocol (IIOP). These IP addresses are not translated, making the packet useless.
- **Avoids DMZ protocol switch.** The Web server sends HTTP requests to application servers behind firewalls. It is simplest to open an HTTP port in the firewall to let the requests through. Configurations that require switching to another protocol (such as IIOP), and opening firewall ports corresponding to the protocol, are less desirable. They are often more complex to set up, and the protocol switching overhead can impact performance.
- **Allows encrypted link between Web server and application server.** Configurations that support encryption of communication between the Web server and application server reduce the risk that attackers will be able to obtain secure information by "sniffing" packets sent between the Web server and application server. A performance penalty usually accompanies such encryption.
- **Avoids single point of failure.** A point of failure exists when one process or machine depends on another process or machine. A single point of failure is especially undesirable because if the point fails, the whole system will become unavailable. When comparing DMZ solutions, a single point of failure refers to a single point of failure between the Web server and application server. Various failover configurations can minimize downtime and possibly even prevent a failure. However, these configurations usually require additional hardware and administrative resources.
- **Minimum required number of firewall holes.** Configurations that minimize the number of firewall ports are desirable because each additional firewall port leaves the firewall more vulnerable to attackers.
- **Relative performance.** Some solutions are faster than others, in terms of the number of client requests they can process per unit of time.
- **Relative administrative maintenance.** Some solutions require little or no maintenance after you establish them, while others require periodic administrative steps, such as stopping a server and starting it again after modifying resources that affect the configuration. To learn about the necessary maintenance for a topology, review the instructions for setting up and maintaining that topology. Of course, if you can automate the necessary administrative steps, this might not concern you. See article 6.6.0.2 for information about the available command-line clients and scripting possibilities.

7.1.5: Remote database access with DB2 Universal Database (UDB)

DB2 databases can be installed on the same machine as the WebSphere Application Serversoftware or on a different machine. Installing the database on a different machinehas several advantages:

- Placing the database and application server software on different machines improves their performance because they do not need to compete for system resources.
- You can independently tune the machines that host the database server and the application server to achieve optimal performance.
- Many organizations have invested in high-availability solutions for their database servers, reducing the possibility of it being a single point of failure in a system.

All remote database clients use a communications product to support the protocol thatis used to access a remote database server. The protocol stack must be installed andconfigured before a client can communicate with a remote DB2 UDB server. ForWebSphere Application Server, the recommended protocol is TCP/IP.

See the WebSphereApplication Server installation documentation for your platform for instructions on how toconfigure a remote DB2 database.

7.1.6: Managing state

Multimachine scaling techniques rely on using multiple copies of an application server;multiple consecutive requests from various clients can be serviced by different servers.If each client request is completely independent of every other client request, it doesnot matter whether consecutive requests are processed on the same server. However, inpractice, client requests are not independent. A client often makes a request, waits forthe result, then makes one or more subsequent requests that depend on the results receivedfrom the earlier requests. This sequence of operations on behalf of a client falls intotwo categories:

- **Stateless**: A server processes requests based solely on information provided with each request and does not reply on information from earlier requests. In other words, the server does not need to maintain state information between requests.
- **Stateful**: A server processes requests based on both the information provided with each request and information stored from earlier requests. In other words, the server needs to access and maintain state information generated during the processing of an earlier request.

For stateless interactions, it does not matter whether different requests are processedby different servers. However, for stateful interactions, the server that processes arequest needs access to the state information necessary to service that request. Eitherthe same server can process all requests that are associated with the same stateinformation, or the state information can be shared by all servers that require it. In thelatter case, accessing the shared state information from the same server minimizes theprocessing overhead associated with accessing the shared state information from multipleservers.

The load distribution facilities in WebSphere Application Server make use of severaldifferent techniques for maintaining state information between client requests:

- **Session affinity**, where the load distribution facility recognizes the the existence of a client session and attempts to direct all requests within that session to the same server.
- **Transaction affinity**, where the load distribution facility recognizes the existence of a transaction and attempts to direct all requests within the scope of that transaction to the same server.
- **Server affinity**, where the load distribution facility recognizes that although multiple servers might be acceptable for a given client requests, a particular server is best suited for processing that request.

The WebSphere Session Manager, which is part of each application server, stores clientsession information and takes session affinity and server affinity into account whendirecting client requests to the clones of an application server. The workload managementservice takes server affinity and transaction affinity into account when directing clientrequests among the clones of an application server.

- 0.11: What are sessions and Session Managers?
- 4.4.1: Tracking sessions

7.1.6.1: HTTP sessions, servlets, and the session manager

When an HTTP client interacts with a servlet, the state information associated with aseries of client requests is represented as an HTTP session and identified by a sessionID. The Session Manager is responsible for managing HTTP sessions, providing storage forsession data, allocating session IDs, and tracking the session ID associated with eachclient request through the use of cookies or URL rewriting techniques. The Session Managercan store session-related information in memory in two ways:

- In application server memory (the default). This information cannot be shared with other application servers.
- In a database shared by all application servers. This is also known as *persistent sessions* or *session clustering*.

Persistent sessions are essential for using HTTP sessions with a load distributionfacility. When an application server receives a request associated with a session ID thatit currently does not have in memory, it can obtain the required session state byaccessing the session database. If persistent sessions are not enabled, an applicationserver cannot access session information for HTTP requests that are sent to servers otherthan the one where the session was originally created. The Session Managerimplements caching optimizations to minimize the overhead of accessing the sessiondatabase, especially when consecutive requests are routed to the same application server.

Storing session states in a persistent database also provides a degree of faulttolerance. If an application server goes offline, the state of its current sessions isstill available in the session database. This enables other application servers tocontinue processing subsequent client requests associated with that session.

Saving session state to a database does not completely guarantee that it is preservedin case of a server failure. For example, if a server fails while it is modifying thestate of a session, some information is lost and subsequent processing using that sessioncan be affected. However, this situation represents only a very small period of time whenthere is a risk of losing session information.

The drawback to saving session state in a persistent database is that accessing thesession state database can use valuable system resources. The Session Manager can improvesystem performance by caching the database data at the server level. Multiple consecutiverequests that are directed to the same server can find the required state data in thecache, reducing the number of times that the actual session state database must beaccessed (and thus the overhead associated with database access).

7.1.6.2: EJB sessions and transaction affinity

When an EJB client interacts with one or more enterprise beans, the WebSphereApplication Server container manages the state information associated with a series ofclient requests. Whether session state is managed at all depends on the types ofenterprise beans that participate in fulfilling these requests. Each type of enterprisebean is handled differently by the container.

Stateless session bean

By definition, a stateless session bean maintains no state information. Each clientrequest directed to a stateless session bean is independent of the previous requests thatwere directed to the bean. The container maintains a pool of instances of statelesssession beans, and provides an arbitrary instance of the appropriate stateless sessionbean when a client request is received. Requests can be handled by any stateless sessionbean instance in any clone of the application server, regardless of whether the beaninstance handled the previous client requests.

Stateful session beans

A stateful session bean is used to capture state information that must be shared acrossmultiple consecutive client requests that are part of a logical sequence of operations.The client must obtain an EJB object reference to a stateful session bean to ensure thatit is always accessing the same instance of the bean.

WebSphere Application Server supports the cloning of stateful session bean home objectsamong multiple application servers. However, it does not support the cloning of a specificinstance of a stateful session bean. Each instance of a particular stateful session beancan exist in just one application server and can be accessed only by directing requests tothat particular application server. State information for a stateful session bean cannotbe maintained across multiple application server clones.

Entity beans

An entity bean represents persistent data. Most external clients access entity beans byusing session beans, but it is possible for an external client to access an entity beandirectly. The information contained in an entity bean is not usually associated with asession or with the handling of one client request or series of client requests. However,it is common for a client to make a succession of requests targeted at the same entitybean instance. It is also possible for more than one client to independently access thesame entity bean instance within a short time interval. The state of an entity bean musttherefore be kept consistent across multiple client requests.

For entity beans, the concept of a session is replaced by the concept of a transaction.An entity bean is instantiated in a container for the duration of the client transactionin which it participates. All subsequent accesses to that entity bean within thattransaction are performed against that instance of the bean in that particular container.The container needs to maintain state information only within the context of thattransaction. The workload management service uses the concept of transaction affinity todirect client requests, After a server is selected, client requests are directed towardsit for the duration of the transaction.

Between transactions, the state of the entity bean can be cached. The EJB containersupports both option A and option C caching.

- With option A caching, WebSphere Application Server assumes that the entity bean is used within a single container. Clients of that bean must direct their requests to the bean instance within that container. The entity bean has exclusive access to the underlying database, which means that the bean cannot be cloned or participate in workload management if option A caching is used.
- With option C caching (the default), the entity bean is always reloaded from the database at the beginning of each transaction. A client can attempt to access the bean and start a new transaction on any container that has been configured to host that bean. This is similar to the session clustering facility

described for HTTP sessions, since the entity bean's state is maintained in a shared database that can be accessed from any server when required.

7.1.6.3: Server affinity

A load distribution facility (such as the workload management service) is not alwaysfree to pick any available server when it redirects client requests.

- For stateful session beans or entity beans within the context of a transaction, there is only one valid server. An entity bean is instantiated on a single server in a single container during the context of a transaction. Subsequent client requests must be directed to that server. The workload management service always directs client requests to a stateful session bean to the single server instance containing the bean. In either case, directing the request to the wrong server either causes the request to fail or forces the server to forward it to the correct server at a high performance cost.
- For clustered HTTP sessions or entity beans between transactions, the underlying shared database ensures that any available server can be used to process client requests.
- For stateless session beans, any available server can be used because each bean instance is identical.

Server affinity refers to the characteristics of each load distribution facility thattake these constraints into account. The load distribution facility recognizes thatmultiple servers can be acceptable targets for a request. However, it alsorecognizes that each request can be directed to a particular server where it is handledbetter or faster.

Server affinity can be weak or strong.

- In *weak server affinity*, the system attempts to enforce the desired affinity for the majority of requests, but does not always guarantee that this affinity will be respected.
- In *strong server affinity*, the system guarantees that affinity is always respected and generates an error when it cannot direct a request to the appropriate server.

7.2 Managing workloads

Workload management optimizes the distribution of work-processing tasks in theWebSphere Application Server environment. Incoming work requests are distributed to theapplication servers and other objects that can most effectively process the requests.Workload management also provides failover when servers are not available.

Workload management is most effective when used in systems that contain servers onmultiple machines. It also can be used in systems that contain multiple servers on asingle, high-capacity machine. In either case, it enables the system to make the mosteffective use of the available computing resources.

Implementing workload management

WebSphere Application Server, Advanced Edition implements workload management asfollows:

1. By using clones and models. Multiple copies, or *clones*, of an object can be created from a template known as a *model*. Clones and models are most commonly used to create copies of entire application servers, but can be made for any component of a WebSphere application.
2. Enterprise beans and servlets have additional steps for implementing workload management.
    - For enterprise beans, by using workload management-enabled Java Archive (JAR) files. These JAR files can be created by using either the Administrative console or the **wlmjar** utility.
    - For servlets, by using servlet redirection to route client requests to remote servlet clones.

Benefits of workload management

Workload management provides the following benefits to WebSphere applications:

- It balances client workloads, allowing processing tasks to be distributed according to the capacities of the different machines in the system.
- It provides failover capability by redirecting client requests if one or more servers is unable to process them. This improves the availability of applications and administrative services.
- It enables systems to be scaled up to serve a higher client load than provided by the basic configuration. With cloning and modeling, additional instances of servers, servlets, and other objects can easily be added to the configuration.
- It enables servers to be transparently maintained and upgraded while applications remain available for users.
- It centralizes the administration of servers and other objects.

Interoperability between WebSphere Application Server versions

All resources that participate in workload management must be running under the sameversion of WebSphere Application Server. For instance, application servers running underversion 3.5.x of WebSphere Application cannot participate in workload management withapplication servers running under version 3.0.x.

7.2.1 Workload management for enterprise beans and application servers

Workload management for enterprise beans is enabled as follows:

1. Create models and clones of enterprise beans, containers, and application servers.
2. Create a workload management-enabled Java Archive (JAR) file.

Creating models and clones

Models and clones can be created for objects at any level of the containment hierarchy.A model contains the objects that are deployed into the object from which it is created.For example, individual enterprise beans can be modeled and cloned without cloning thecontainer in which they are deployed. Clones of containers include the enterprise beansthat are deployed into the modeled container. Clones of application servers include thecontainers and enterprise beans that are deployed onto their model, and so forth.

Creating a model and clones of an application server and enterprise beans is the firststep in enabling workload management of enterprise beans. Models must be created at theapplication server level for workload management to function.

WebSphere Application Server uses the concept of an application server group, or *cluster*,to identify which application servers participate in workload management. The clones ofone model of an application server constitute an application server cluster. Clientprocessing requests are distributed among the application server instances in the cluster.

Creating workload management-enabled JAR files

A workload management-enabled JAR file enables EJB clients to access the enterprisebeans through the workload management service. These JAR files can be created in two ways:

- By using the **wlmjar** command-line utility. Run this utility on a JAR file into which you have deployed one or more enterprise beans.
- By enabling workload management through the WebSphere Administrative Console. When you are deploying a JAR file (or specifying a deployed JAR file when creating an enterprise bean using the administrative console), you can enable workload management of the JAR file. This option is easier to use than the **wlmjar** utility.

Regardless of which method you use to enable workload management, the resulting JARfile contains stub code that allows EJB clients to access enterprise beans through theworkload management service. Simply set the value of the client's CLASSPATH environmentvariable to the location of the workload management-enabled JAR file.

In general, deploy JAR files with workload management enabled. It does not affect theirnormal behavior unless models and clones are created, at which point workload managementis activated.

How enterprise beans participate in workload management

The workload management service provides load balancing for the following types ofenterprise beans:

- All clones of the home object of an entity or session bean
- All clones of an instance of a specific entity bean or stateless session bean

The reason why stateful session bean instances are treated differently than statelesssession bean instances has to do with how their state is managed. As their name implies,stateless session beans do not maintain state information. All instances of a statelesssession bean are considered to be identical, and each client request that it handles istreated as being made independently of any other requests.

In contrast, stateful session beans are used to store state information that must beshared among multiple and consecutive client requests that are part of a logical sequenceof operations. Each instance of a particular stateful session bean is unique. It existsonly in one application server and can be accessed only by directing requests to thatparticular application server.

Specific instances of stateful session beans cannot be shared between applicationservers. However, their homes can be cloned in the context of cloning the applicationserver in which they are contained. Cloning the home object of a stateful session beanenables an application to create new instances of that bean in an application server.Multiple instances of a specific stateless session bean can exist in clones of anapplication server, but each instance is unique and cannot be shared.

Entity beans exist in a container only within the context of a transaction, regardlessof whether the beans themselves are transactional. The workload management service usesthe concept of transaction affinity to direct client requests for entity beans. After anapplication server is selected, client requests for that entity bean are forwarded to itfor the duration of the transaction. Workload management can be used only if option Ccaching is enabled in the container.

7.2.2 Workload management for servlets

Workload management for servlets can be enabled as follows:

- Clone application servers that host servlet engines.
- Use an HTTP redirector or a transport mechanism to distribute processing requests across multiple machines.

Clone application servers

Cloning an application server in which a servlet engine is running automaticallyenables workload management for the application server and the servlets it hosts.Configure the servlet engine to maximize its performance and throughput, then create aserver group and clones by using the WebSphere Administrative Console.

Redirect processing requests

A second way to manage workloads is to distribute HTTP requests to clones that resideon a machine other than the machine containing the Web server. You can enableworkload mangement through an HTTP redirector or the underlying transport mechanism usedin WebSphere Application Server.

A *servlet redirector* is a special-purpose application server that uses RemoteMethod Invocation (RMI) over the Internet Inter-ORB Protocol (IIOP) to distribute HTTPrequests to application servers on machines remote to the Web server. It is used todistribute requests to clones of a servlet engine.

Servlet redirectors act as EJB clients to the cloned application servers in which theservlet engines run. Because they use EJB client invocations to communicate with theapplication servers, they can participate in workload management. This enables them toperform load balancing and provides failover support.

Redirecting processing requests to application server clones by using remote OSEprovides a simplified form of load balancing and supports failover. Load balancingis done manually by redirecting requests from URLs to specific application servers andtheir clones and by a round-robin selection policy between clones. Remote OSE itselfis not part of the workload management service.

7.2.3 Workload management for administrative servers

Administrative servers can participate in workload management. Workload managementprovides failover capability, improving the availability of administrative and namingservices. It also eliminates the possibility of an administrative server being a singlepoint of failure in a system.

Workloadmanagement must be enabled or disabled for all administrative servers in a domain.

When an administrative server participates in workload management, an exception isthrown if the administrative server fails during an administrative task. Subsequentrequests are redirected to the other administrative servers in the domain, minimizing thedisruption to administrative operations.

For example, a command issued through the WebSphere Administrative Console can fail ifan administrative server goes offline while the command is being executed. If workloadmanagement is enabled, any subsequent attempts to execute the command are redirected toanother administrative server. This allows the command to be successfully reissued,possibly with a delay for the initial redirection. Subsequent requests are noticeably slower. The original administrative server will picks up its share ofadministrative requests when it comes back online.

Enabling workload management

To begin workload management, start all administrative servers in the domain withworkload management enabled. WebSphere Application Server provides two ways to enableworkload management:

- By setting the following property in the admin.config file:

  ```
  com.ibm.ejs.sm.AdminServer.wlm=true
  ```

  This enables workload management for all administrative servers that are started by using this configuration file.

- By specifying the -wlm argument when starting an administrative server from the command line. For instance:

  ```
  java com.ibm.ejs.sm.server.AdminServer -wlm ...
  ```

  where . . . represents any other arguments that are specified when starting the server.

Enabling workload management through the admin.config file is recommended because it iseasier to administer than enabling it through the command line.

Disabling workload management

To discontinue workload management, stop all administrative servers in the domain andrestart them with workload management disabled.

Disable workload management in one of the following ways:

- By placing comment markers around the com.ibm.ejs.sm.AdminServer.wlm property in the admin.config file.
- By restarting the server from the command line without specifying the -wlm argument.

7.2.4 Using models and clones

A *model* is a template for creating copies of a server or process instance,such as an application server or servlet engine. The copies are called *clones*.The act of creating the clones is called *cloning*.

Cloning and modeling allows identical copies of objects (such as application servers,servlets, and so forth) to be created. A system administrator first creates a model thatabstractly represents an object. From this model, one or more clones can be created. Theclones represent real application server processes; when first created, they are identicalto the model in every way.

Changes to a model are propagated to its clones when the clones are restarted. You canefficiently administer several copies of a server or other resource by administering itsmodel.

Models and clones can be created for objects at any level of the containment hierarchy.However, it is recommended that you create them for application servers. A modelcontains the objects that are deployed into the object from which it is created. Forexample, individual enterprise beans can be modeled and cloned without cloning thecontainer in which they are deployed. Clones of containers include the enterprise beansthat are deployed into the modeled container. Clones of application servers include thecontainers and enterprise beans that are deployed onto their model, and so forth.

Modeling and cloning objects at different levels of the application server containmenthierarchy (from individual servlets and enterprise beans to entire application servers)gives administrators a great deal of flexibility for implementing and administeringapplications. For example, a system administrator can create models of an applicationserver, servlets, containers, and enterprise beans; adjust the properties of the models tooptimize the performance of these objects; then deploy them by creating and startingclones from these models.

Working with models and clones

The basic procedure for using models and clones is as follows:

1. Create the original instance of the object that you want to clone (such as an application server, servlet, or enterprise bean). Configure it exactly as you would like it. For example, you can configure an application server to meet specific performance goals.
2. Create a model of the object by using the administrative console. Making the original instance a clone is recommended but not required. The original instance can remain freestanding.
3. Create clones from the model.
4. When changes are necessary, apply them to the model, which in turn modifies the original instance (which is now a clone) and the other clones.

Determining which resources can be cloned

To determine whether a resource can be cloned, right-click the resource in the Topologytree to display a pop-up menu. If the menu contains a **Create -> Model**option, the resource can be cloned.

Resources that can be cloned include the following:

- Application servers
- EJB containers
- Enterprise beans
- Servlet engines
- Servlets
- Web applications

Although WebSphere Application Server supports cloning at all levels of the containmenthierarchy, it is recommended that you create models and clones at the application serverlevel. The cloned application server contains clones of the enterprise beans, servlets,and other resources that are deployed onto it. Creating models and clones at theapplication server level simplifies their administration. It is easier to manage theclones of an application server than to individually manage clones of all of the resourcescontained in it.

7.2.4.1 Cloning for workload management, failover, and scaling

Cloning supports workload management, failover, and scaling.

Workload management

Models and clones provide necessary support for workload management. Whenyou modify a model, the change is propagated to its clones when they arerestarted. Besides making it easy to administer several servers as one logical server,this keeps the clones identical so that requests can be routed to any one of them with thesame results.

This ability to route a request to any server in a group of identical servers allowsthe servers to share work, improving throughput of client remote method invocations.Requests can be evenly distributed to servers to prevent workload imbalances in which oneor more servers have idle or low activity while others are overburdened. This load-balancing activity is a benefit of workload management.

Failover

With several clones available to handle requests, it is more likely that failures willnot damage throughput and reliability. With clones distributed to various nodes, an entiremachine can fail without producing devastating consequences (unless, of course, the failedmachine is a single point of failure). Requests can be routed to other nodes if one nodefails.

Scaling

Cloning is an effective way to perform vertical and horizontal scaling of applicationservers.

- In *vertical scaling*, clones are defined on a single machine to allow the machine's processing power to be more efficiently allocated. It is particularly useful if your environment contains large, underutilized machines. A single application server is implemented by a single Java Virtual Machine (JVM) process and cannot fully utilize the power of a large machine. (This is especially true on large multiprocessor computers because of concurrency limitations within a single JVM process.) Vertical scaling allows multiple application server clones (and therefore JVM processes) to be created, which makes use of the machine's processing power more effectively. Vertical scaling is described in more detail in article 7.1.3.3.
- In *horizontal scaling*, clones are defined on multiple machines in a system. This allows a single WebSphere application to run on several machines while presenting a single system image, making the most effective use of the resources of a distributed computing environment. Horizontal scaling is especially effective in environments that contain many smaller, less powerful machines. Client requests that overwhelm a single machine can be distributed over several machines in the system. Failover is another benefit of horizontal scaling. If a machine becomes unavailable, its work can be routed to other machines containing server clones. Horizontal scaling is described in more detail in article 7.1.3.4 and article 7.1.3.5.

WebSphere applications can combine horizontal and vertical scaling to reap the benefitsof both scaling techniques.

7.2.4.2 Modifying models and clones

To perform an administrative action on a clone (such as modifying the clone'sproperties), perform the action on the associated model. For example, to add an enterprisebean to an application server clone, you must add the bean to the server group. With oneaction, you can add an enterprise bean to all clones of the application server.

Changes related to workload management (such as selection policy changes, startingclones, and stopping clones) are propagated to workload management clients. Other modelchanges (such as adding or removing enterprise beans from an application server) arepicked up by the clones when they are restarted. System administrators can perform a *ripplerestart* to propagate changes by stopping and restarting clones one after anotherwithout stopping the model.

If you modify a clone directly (instead of through its model), the clone no longer isidentical to its model. However, it continues to be part of its model unless it isdissociated from the model.

Freestanding (disassociated) clones

A clone must be explicitly disassociated from its model. It then becomes a *freestanding*object and can be administered independently. Any changes you make to the former model ofthe clone are *not* propagated to the clone.

Freestanding clones can be created by using the **wscp** command-lineutility.

7.2.4.3 Advice for cloning

Create clones based on your knowledge of the application and on the expected workload.Some considerations:

- Clones do not need to reside on the same machine.
- Clients can have inconsistent views of configuration information in the model. This can occur when an application server instance is stopped, started, added, or deleted. The period of inconsistency is short-lived, however. Clients eventually refresh their caches of server information. Application server instances that are unchanged during the period of inconsistency remain available.
- If you make changes to a model, clients do not need to be restarted. The changes are eventually propagated to them.
- You can make changes to a model while application servers are running. However, incremental changes (such as adding or removing one or two clones) have less impact on client performance than wholesale changes.
- It is always best to make changes when few clients and application servers are running.
- You can add or remove server clones later in response to the load on the application. Alternattively, you can clone the initial number of application server instances based on expected load.
- If a machine becomes unavailable, you do not need to reconfigure the clones of other application servers to compensate for any unavailable application servers on that machine. However, if the machine is going to be unavailable for an extended period, you can reconfigure the other servers to optimize performance.

7.2.4.4 Containment relationships

While cloning a resource, you can specify the relationship of the clone to its modeland other models.

Freestanding or contained?

Specify a containment relationship for each model, which determines whether the modelis freestanding or contained.

A model is *freestanding* if it is not contained by another resources's model.If a model is part of another resource's model, it is *contained*.

For example, a model of an enterprise bean can be *contained* by the model of anapplication server on which the bean is installed. Clones of the application server willcontain clones of the enterprise bean.

Model recursively?

When you create a servlet engine model that contains a Web application, you canoptionally specify to include the Web application, and any servlets it contains, in themodel.

Containment relationships are preserved when you model and clone an instance that iscontained by another instance. For example, if you create a model of a Web applicationthat is contained by a particular servlet engine, the clones of the Web application modelwill also be associated with that servlet engine.

7.2.4.5 Server selection policies and transaction affinity

When you are cloning an application server, you need to take the following things intoaccount:

- Application server models and clones
- Server selection policies
- Transaction affinity for application servers

Application server models and clones

A *servlet engine* is a server process that works with your Web server to handlerequests for servlets and Web resources such as HTML, JavaServer Pages (JSP) files, andsuch. A server object, such as an application server or servlet engine, can be used as amodel for creating multiple clones of that server. The clones are defined by a model.Changes to the model are propagated to the clones when the server clones are restarted.

The clones remain basically identical to the model, allowing work to be distributed toany one of them. Server selection policies determine how clientschoose server instances within the group.

Server selection policies

The workload management server selection policy defines how clients choose amongapplication server clones (instances). Select among these policies:

- Random
- Round-robin
- Random prefer local
- Round-robin prefer local

See article 6.6.22.0, for a detailed description of theserver selection policies.

Transaction affinity for application servers

Regardless of the selection policy used, the workload management service attempts tochoose an application server clone based on *transaction affinity*. Within atransaction, the first time a server is picked, the prevailing selection policy for theserver group is applied. After a server is selected, it remains bound for the duration ofthe transaction.

For example, suppose the round-robin policy is specified for server group A with twoapplication server clones, S1 and S2. A client has two concurrent threads, t1 and t2, withtransaction contexts T1 and T2, respectively. Assume that thread t1 is first and needs toselect a server from server group A; clone S2 is randomly chosen. When t2 tries to selecta server from server group A, S1 is chosen based on the round-robin policy in effect forthe server group. Subsequent requests to server group A are serviced by S2 for t1 and S1for t2, based on transaction affinity.

7.2.4.6 Security for cloned resources

The workload management service has its own built-in security, which works with theWebSphere application server security service to protect cloned resources. When youare creating clones of application servers, enable security before you create a model ofthe application server. This enables security for all of the application serverclones created from that model.

Protecting cloned enterprise beans

Enterprise beans that are cloned in the context of cloning an application server areprotected under the application server's security. Enterprise bean instances and theirclones have separate identities, although workload management treats them as beingidentical. Therefore, you must protect every cloned enterprise bean by configuringresource security for the enterprise bean and including it in a secured enterpriseapplication.

Protecting cloned servlets

Servlets that are cloned in the context of cloning an application server are nottreated as separate resources by WebSphere security. If the original servlet isprotected, its clones are too, with no additional steps required by the administrator. Tosecure a servlet, add its Web resource configuration (URI) to a secured enterpriseapplication.

7.2.4.7: Creating clones on machines with different WebSphere installation directories or operating systems

Different hardware and operating system platforms do not usually have the sameWebSphere Application server *product installation root*directories. The following steps are required to create clones on multiple machineswhen WebSphere Application Server is installed in different directories on differentmachines or when different directory structures exist across multiple platforms:

1. On one node, create a model of the application server to be cloned. The platform does not matter if all machines share the same administrative repository database.
2. Make the original application server instance a clone and recursively model all instances under the application server. If you are creating a model of the default application server, make sure that it is not already installed on the machines that it will be cloned to.
3. For all other nodes in the configuration:
   a. Create a clone on the machine.
   b. If desired, copy the application files (the files containing servlet, enterprise bean, JavaServer Pages, and HTML code) to the machine.
   c. Modify the following properties of the clone. The directory structures of these fields must be changed to match the directory structure of the *product_ installation_root* directory and the Web application file locations on the machine where the clone is running.
      - The **Standard Output** field of cloned application servers
      - The **Standard Error** field of cloned application servers
      - The **JAR File** field of cloned enterprise beans
      - The **Document Root** field of cloned web applications
      - The **Classpath** table of cloned web applications

      These changes do not make the clones freestanding.
   d. Start the cloned application server.

   Repeat these steps for each machine in the configuration. Changes made to individual clones are not propagated to the other clones in the system.

Modifying the model can overwrite these changes, requiring you to redo them.

7.2.5 Using workload management - a sample procedure

The following procedure shows how to implement basic workload management by cloningapplication servers and enterprise beans. In this scenario, client requests aredistributed among the clones of an application server on a single machine. (A clientrefers to any servlet, Java application, or other program or component that connects theend user and the enterprise beans being accessed.) In more complex workload managementscenarios, you can distribute clones to remote machines, clone servlets, or configure aservlet redirector.

1. Decide which application server you are going to clone.
2. Deploy the enterprise beans that you plan to clone. Optionally, chose the deployment option to enable workload management for the JAR file.
3. After configuring the server and enterprise beans exactly as you want them to be, create a model of the server. This is the first step in cloning the server. Make sure that the model includes the enterprise beans that are deployed on the server. It is recommended that you make the original server instance a clone that is administered through the model.
4. Create one or more clones of the server model.
5. Start all of the application servers by starting the model.
6. If you did not enable workload management when you were deploying the enterprise beans, use the **wlmjar** command against the deployed JAR file of the enterprise bean to produce a WLM-enabled JAR file.
7. Add the WLM-enabled JAR file to the class path of the client you want to enable to exercise workload management.
8. If the client is a servlet, also specify the WLM-enabled JAR file in the class path of the application server on which the servlet resides.

Workload management automatically begins when you start the clones of the applicationserver.

ℹ You need to define a bootstrap host forstand-alone Java clients -- that is, clients that are located on a different machine fromthe application server and have no administrative server for the client. Add the followingline to the Java Virtual Machine (JVM) arguments for the client:

```
-Dcom.ibm.CORBA.BootstrapHost=machine_name
```

where *machine_name* is the name of the machine on which the administrativeserver is running.

7.2.6 Tuning a workload management configuration

The workload management service uses several parameters to control the behavior of theworkload management run time. In the majority of cases, you do not need to explicitly setthe values of these parameters. However, if you are experiencing problems with yourworkload management configuration, you can adjust these properties to tune the behavior ofthe workload management run time.

⚠ Set the values of these properties only inresponse to problems that you encounter in your environment. If workload management isfunctioning correctly, changing these properties can produce undesirable results.

Workload management client properties

A workload management client can be a cloned resource or an application server thatacts as an EJB client to a cloned resource. The following properties can be used tocontrol the behavior of the workload management client run time. They are set ascommand-line arguments for the Java Virtual Machine (JVM) process in which the workloadmanagement client is running. In many cases, such as where a servlet is a client to anenterprise bean, this means that these parameters are specified as part of the command-line arguments for the application server where the servlet is running.

- **com.ibm.CORBA.requestTimeout.** This property specifies the timeout period for responding to workload management requests. Set this value in the Command Line Arguments field by using the -D option as follows:

  ```
  -Dcom.ibm.CORBA.requestTimeout=timeout_interval
  ```

  where *timeout_interval* is the timeout period in seconds. If your network is subject to extreme latency, specify a large value to prevent timeouts. If you specify a value that is too small, an application server that particpates in workload management can ttime out before it receives a response.

ℹ Be very careful whenyou specify this property: it has no recommended value. Set it only if your application isexperiencing problems with timeouts.

- **com.ibm.ejs.wlm.MaxCommFailures**. This property specifies the number of attempts that a workload management client makes to contact the administrative server that manages workloads for the client. The workload management client run time does not identify an administrative server as unavailable until a certain number of attempts to access it have failed. This allows workload management to continue if the server suffers from transient errors that can briefly prevent it from communicating with a client. However, it can also propagate nontransient administrative server failures to the client. Set this value in the **Command Line Arguments** field in the administrative console by using the -D option as follows:

  ```
  -Dcom.ibm.ejs.wlm.MaxCommFailures=max_failures
  ```

  where *max_failures* specifies how many times the client attempts to contact the administrative server after the first failure. The default value is zero, which means that the workload management run time does not attempt to use the administrative server after the first failure until a timeout interval (specified by the **com.ibm.ejs.wlm.UnusableInterval** parameter) expires. This reduces the possibility of further server failures being propagated to the client.

- **com.ibm.ejs.wlm.UnusableInterval**. This property specifies the time interval that the workload management client run time waits after it marks an administrative server as unavailable before it attempts to contact the server again. Set this value in the **Command Line Arguments** field in the administrative console by using the -D option as follows:

  ```
  -Dcom.ibm.ejs.wlm.UnusableInterval=interval
  ```

  where *interval* is the time in seconds between attempts. The default value is 900 seconds. If this parameter is set to a large value, the server is marked as unavailable for a long period of time. This prevents the workload management refresh protocol from refreshing the workload management state of the client until after this time period has ended.

Administrative server properties

The administrative server for the cloned resources that participate in a workloadmanagment group (such as an application server cluster) acts as the workload managementserver.

- **com.ibm.ejs.wlm.RefreshInterval**. This property specifies the interval at which the administrative server updates the server group information to the cloned application servers that participate in workload management. It is appended to the arguments for the **com.ibm.ejs.sm.util.process.Nanny.adminServerJvmArgs** entry in the administrative server configuration file. The value of this property is specified as follows:

  ```
  com.ibm.ejs.wlm.RefreshInterval=interval
  ```

  where *interval* is the number of seconds that elapse between the administrative server updates. The default value is 300 seconds.

7.2.7 Run-time exceptions and failover strategies for workload management

Workload management run-time exceptions

The workload management service can throw the following exceptions if it encountersproblems:

- `org.omg.CORBA.NO_IMPLEMENT`. This exception is thrown if the workload management service cannot contact any of the EJB application servers that participate in workload management.
- `org.omg.CORBA.INTERNAL`. This exception is thrown when an internal software failure occurs. The error is listed in the WebSphere client trace log. (Be aware that if WebSphere is not installed on the client machine, no logging is performed.)
- `org.omg.CORBA.COMM_FAILURE`. This exception is thrown by the ORB when a communications failure occurs. Any current transactions are rolled back, and nontransactional requests are redone.
- `org.omg.CORBA.NO_RESPONSE`. This exception is thrown by the ORB when a communications failure occurs.

The WebSphere Application Server client can catch these exceptions and then implementits own strategies to handle the situation; for example, it can display an error messageif no servers are available.

Workload management failover strategies

The workload management service uses the following failover strategies, some of whichare based on the return values of these exceptions:

- If the workload management service cannot contact an application server clone, it automatically redirects the request to another clone, providing automatic failover.
- If the application throws an exception, automatic failover does not occur. The workload management service does not retry the request because it cannot know whether the request was completed.
- If an `org.omg.CORBA.NO_IMPLEMENT` exception is thrown, the workload management service has attempted repeatedly to contact the application servers without success. Workload management resumes when application servers become available again.
- If an `org.omg.CORBA.INTERNAL` exception is thrown, the workload management service is no longer operating properly and no failover occurs.
- If the `org.omg.CORBA.COMM_FAILURE` or `org.omg.CORBA.NO_RESPONSE` exceptions are thrown, their return value determines whether automatic failover occurs:
  - If one of these exceptions is thrown with a `COMPLETION_STATUS` of `COMPLETED_NO`, automatic failover occurs because the request was not completed.
  - If one of these exceptions is thrown with a `COMPLETION_STATUS` of `COMPLETED_YES`, failover does not occur because the request was successfully completed.
  - If one of these exceptions is thrown with a `COMPLETION_STATUS` of `MAYBE` (which maps to a `java.rmi.RemoteException`), automatic failover does not occur. The workload management service cannot verify whether the request was completed. In this situation, the client application must anticipate a failure and retry the request. The workload management service then attempts to direct the request to a surviving application server clone.

7.2.8 Workload management for stand-alone Java clients

- Enabling workload management for a stand-alone Java client
- Enabling workload management and security for a stand-alone Java client

Stand-alone Java applications (Java applications that do not run under WebSphereApplication Server), J2EE clients, administrative agents, and other types of Javaapplications can participate in WebSphere workload management. This extends thebenefits of workload management (such as load balancing and failover support) to Javaapplications that run on machines where WebSphere Application Server is not installed. TheJava client can optionally participate in WebSphere security.

These procedures have been tested only onthe Windows NT platform.

Enabling workload management for a stand-alone Java client

To enable stand-alone Java applications to participate in workload management, do thefollowing:

1. Copy WebSphere Application Server Java Archive (JAR) files to the machine where the Java application runs.
2. Add the names of the JAR files to the CLASSPATH environment variable on the machine where the Java application runs.
3. Add the executable files of the supported version of the Java 2 SDK to the PATH environment variable on the machine where the Java application runs.
4. Start the Java application, setting the appropriate Java system properties to enable workload management.

The rest of this section describes this setup procedure in more detail.

Copy the JAR files to client machine

The following JAR files must be copied from a machine where WebSphere ApplicationServer is installed to the machine where the client application runs:

- Copy the *product_installation_root*/lib/ujc.jar file to the WebSphere/jars directory.
- Copy the *_wlm_deployedBean*.jar file to the WebSphere/jars directory, where *_wlm_deployedBean* is the name of the workload management-enabled JAR file that contains the enterprise beans being used by the Java client.

Install the SDK on the client machine

Workload management for stand-alone clients is supported for the Java 2 SDK, versionIBM 1.2.2. If the machine where the client application runs is not currently usingthis version of the SDK, you must install it. See the SDK installation instructionsfor details.

Add the JAR files to the CLASSPATH variable on the client machine

Add the names of the JAR files to the CLASSPATH variable on the machine where theclient application runs. For example:

```
CLASSPATH=D:\WebSphere\jars\_wlm_deployedBean.jar;D:\WebSphere\jars\ujc.jar;%CLASSPATH%
```

Add the directories containing the SDK executable files to the PATH variable on theclient machine

Add the directories containing the executable files of the SDK to the PATH variable onthe machine where the client application runs. For example:

```
PATH=C:\WebSphere\SDK\bin;C:\WebSphere\jdk\jre\bin;%PATH%
```

Run the client application

To enable the client application to participate in workload management, start it withthe following system parameters:

- com.ibm.ejs.wlm.BootstrapNode=*admin_server_node*
- com.ibm.CORBA.BootstrapHost=*admin_server_node*
- com.ibm.CORBA.BootstrapPort=900

where *admin_server_node* is the name of the machine where the WebSphereadministrative server is located. You can specify either the short name, the IPaddress, or the fully qualified name of the machine. For example:

```
java -Dcom.ibm.ejs.wlm.BootstrapNode=greenland
-Dcom.ibm.CORBA.BootstrapHost=greenland.rh1.ibm.com     -Dcom.ibm.CORBA.BootstrapPort=900 WlmApp
```

Enabling workload management and security for astand-alone Java client

Enabling workload management with security requires additional steps to be performed:

1. Copy the JAR files and the sas.client.props file to the machine where the Java application runs.

2. Add the names of the JAR files to the CLASSPATH environment variable on the machine where the Java application runs.
3. Add the executable files of the supported version of the Java 2 SDK to the PATH environment variable on the machine where the Java application runs.
4. Enable the Java client application to access the EJB application on the machine where WebSphere Application Server is installed.
5. Start the Java application, setting the appropriate Java system properties to enable workload management.

The rest of this section describes this setup procedure in more detail.

Copy the files to the client machine

The following files must be copied from a machine where WebSphere Application Server isinstalled to the machine where the client application runs:

- Copy the *product_installation_root*/properties/sas.client.props file to the WebSphere/properties directory. This file contains security configuration properties.
- Copy the *product_installation_root*/lib/sslight.jar file to the WebSphere/jars directory.
- Copy the *product_installation_root*/lib/ujc.jar file to the WebSphere/jars directory.
- Copy the *_wlm_deployedBean*.jar file to the WebSphere/jars directory, where *_wlm_deployedBean* is the name of the workload management-enabled JAR file that contains the enterprise beans being used by the Java client.

Install the SDK on the client machine

Workload management with security supports the same version of the SDK as workloadmanagement alone: Java 2 SDK, version IBM 1.2.2. To install it, follow the SDKinstallation instructions.

Add the JAR files to the CLASSPATH variable on the client machine

Add the names of the JAR files to the CLASSPATH environment variable on the machinewhere the client application runs. For example:

```
CLASSPATH=D:\WebSphere\jars\_wlm_deployedBean.jar;D:\WebSphere\jars\ujc.jar;
D:\WebSphere\jars\sslight.jar;%CLASSPATH%
```

Add the directories containing the SDK executable files to the PATH variable on theclient machine

Add the directories containing the executable files of the SDK to the PATH variable onthe machine where the client application runs. For example:

```
PATH=C:\WebSphere\SDK\bin;C:\WebSphere\jdk\jre\bin;%PATH%
```

Set up the server to enable Java client access

A WebSphere administrator must create an application that specifies which enterprisebeans require authorization and security checking by using the appropriate administrationconsole wizards. The administrator must set the security permissions for all clients ofthe application to allow access to the read, write, remove, create, execute, and findermethods of the enterprise beans. See article 6.6.18.1,Securing applications, for details on setting up security.

Run the client application

To enable the client application to participate in workload management, start it withthe following system parameters:

- com.ibm.CORBA.ConfigURL=file:/C:/Websphere/properties/sas.client.props
- com.ibm.ejs.wlm.BootstrapNode=*admin_server_node*
- com.ibm.CORBA.BootstrapHost=*admin_server_node*
- com.ibm.CORBA.BootstrapPort=900

where *admin_server_node* is the name of the machine where the WebSphereadministrative server is located. You can specify either the short name, the IPaddress, or the fully qualified name of the machine. For example:

```
java -Dcom.ibm.CORBA.ConfigURL=file:/C:/Websphere/properties/sas.client.props
-Dcom.ibm.ejs.wlm.BootstrapNode=greenland      -Dcom.ibm.CORBA.BootstrapHost=greenland.rh1.ibm.com
-Dcom.ibm.CORBA.BootstrapPort=900 WlmApp
```

7.3: Redirecting servlets

Servlet redirectors are used to direct requests from a Web server to one or more clonesof a servlet engine in an application server. WebSphere Application Server supportstwo mechanisms for redirecting HTTP requests to servlets:

- **Open Servlet Engine (OSE)**. OSE is a proprietary mechanism for transporting data. It can be used to direct client requests from the Web server to servlets on the same machine (local OSE) or on a remote machine (remote OSE). OSE is fully integrated with WebSphere's modeling and cloning facility.
- **Servlet redirector**. A servlet redirector is a dedicated application server that is specifically designed to forward client requests to remote servlet engines. The servlet redirector receives requests locally through OSE. However, instead of processing them itself, it redirects them to other application servers. The servlet redirector can be run in three different configurations:
    - The *thick servlet redirector* is configured on a machine that is a full WebSphere Application Server node with an administrative server. It can be administered like any other application server.
    - The thick servlet redirector can also run with a simplified version of the administrative server known as an *administrative agent*. The administrative agent manages the servlet redirector, but it is controlled by a master administrative server on another node.
    - The *thin servlet redirector* runs without the administrative server. Special scripts are provided to manually start it and generate its configuration files.

Servlet redirection is generally used to scale up configurationsthat use a single servlet engine on a single machine to configurationsthat use multiple servlet engines on multiple machines. Bothservlet redirection mechanisms provide load balancing, failoversupport, and availability management for servlet engines.

7.3.1: OSE and remote OSE

- OSE
- Remote OSE

OSE

Open Servlet Engine (OSE) is a lightweight, proprietary protocol for transporting data.It is used by WebSphere to forward requests from the Web server to an application serverfor processing.

OSE is configured though the Web server plug-in. It associates a uniform resourcelocator (URL) with one or more data queues. Each data queue is associated with anapplication server (and its clones) that is designated to service requests from thatURL. Data can be sent through a variety of protocols, including pipes, Unix domainsockets, and TCP/IP sockets. OSE does not support data encryption between the Webserver and the application server, although HTTPS requests can be used between a browserand the Web server.

OSE (or *local OSE*) was originally designed to be used for transporting data ona single machine. It is the default mechanism for transporting data between between aWeb server and an application server running on the same machine.

Remote OSE

An extension to OSE enables it to be used to transport data remotely. *RemoteOSE* can be used to enable communication between a Web server on one machine andapplication server clones that are running on one or more other machines in the system. Remote OSE uses only the TCP/IP protocol for transporting data between machines. All other OSE characteristics remain the same as in the local case, including theability to route URLs to different application servers. Remote OSE supportsWebSphere security.

In a Remote OSE configuration, the Web server is physically separated from theapplication servers. The Web server sends requests that require intensive processing toother machines, enabling it to process more requests. There are no applicationserver instances running on the machine that hosts the Web server, although Remote OSEcurrently requires the WebSphere Application Server software to be installed on the Webserver machine. The **OSERemoteConfig** script is used to configure theWeb server plug-in to enable Remote OSE.

Remote OSE is the preferred method for redirecting client requests to servlet engineclones. It provides better performance than the servlet redirector. Remote OSEsupports firewalls with Network Address Translation (NAT) and can be used in demilitarizedzone (DMZ) configurations where the Web server runs on a secure server. It does notrequire database access through a firewall.

Because the OSE transport does not support data encryption, Remote OSE is not suitablefor configurations where data encryption is required between the Web server and theapplication server. Unlike other WebSphere Application Server internal IIOP traffic,Remote OSE does not implicitly use Secure Sockets Layer (SSL) when security is enabled.

Article 7.1.3.6.1, Remote OSE sample topology, describesthe Remote OSE configuration in more detail.

Semi-remote OSE is a variant on the Remote OSE configuration. Anapplication server instance runs on the same machine as the Web server; other applicationserver instances run on remote machines. It is generally used when hardwarelimitations prevent the full Remote OSE configuration from being used. Article 7.1.3.6.2, Semi-remote OSE sample topology, describesthe semi-remote OSE configuration in more detail.

7.3.2: Servlet redirector

- How the servlet redirector works
- Thick servlet redirector
- Thick servlet redirector with administrative agent
- Thin servlet redirector

How the servlet redirector works

The servlet redirector is a dedicated, special-purpose application server that runs onthe same machine as the Web server. It forwards HTTP requests received by a Webserver to one or more application servers. The servlet redirector receives HTTPrequests through a local OSE channel just like any other application server. However, instead of processing the requests itself, it sends them to servlet enginesrunning in remote application servers for processing.

The WebSphere Application Server EJB facility is used to forward requests from theservlet redirector to a remote application server. Each application server thataccepts requests from the servlet redirector contains a special stateless session beancalled the RemoteSRP bean. The RemoteSRP bean exports a method that forwards theincoming HTTP request to the servlet engine for execution.

The servlet redirector acts as an EJB client to the application server. When itreceives an HTTP request, it looks up the RemoteSRP bean in the target application server. It then remotely invokes the forwarding method through Remote Method Invocation(RMI) over the Internet Inter-ORB Protocol (IIOP). The HTTP request is thenforwarded from the servlet redirector to the servlet engine in the target applicationserver for processing.

Because the servlet redirector communicates with application servers through EJB clientinvocations, it can participate in workload management. This allows it to distributerequests to cloned application servers and provides support for load balancing andfailover. See article 7.2, Managing workloads, for moreinformation on the workload management service.

The servlet redirector supports WebSphere security for servlets and enterprise beans. It also supports encrypted communications between the servlet redirector and theapplication server. However, it does not support Network Address Translation (NAT)firewalls.

The servlet redirector can be configured in three different ways, depending on howadministration is set up on the Web server machine:

- Thick servlet redirector
- Thick servlet redirector with administrative agent
- Thin servlet redirector

These configurations are described in the rest of this article.

Thick servlet redirector

In this configuration, the machine where the servlet redirector runs is configured as afull WebSphere Application Server node with an administrative server and its associatedprocesses. The thick servlet redirector is administered like any other applicationserver process. It requires a database client to access the WebSphere administrativerepository; a database user ID and password must be stored on the machine for use by thedatabase processes. The administrative server also requires at TCP connection to theremote database. If the thick servlet redirector is being used with a firewall, aport must be opened for database traffic and the firewall must support

IIOPcommunications.

Article 7.1.3.6.4, Thick servlet redirector sample topology,gives a more detailed description of this configuration of the servlet redirector.

Thick servlet redirectorwith administrative agent

In this configuration, the administrative server on the machine where the servletredirector runs is configured as an agent of an administrative server running on anothermachine. The servlet redirector is administered through the remote administrativeserver. The agent receives configuration and administration information from theremote administrative server. Access to the WebSphere administrative repository isthrough the remote administrative server, not the agent. This reduces the number ofprocesses that run on the Web server machine. It also eliminates the need to installa database client on the machine. If a firewall is in use, running the servletredirector with an administrative agent instead of a full administrative server eliminatesthe need to open up a port for database communications.

Article 7.1.3.6.5, Thick servlet redirector withadministrative agent sample topology, gives a more detailed description of thisconfiguration of the servlet redirector.

Thin servlet redirector

In this configuration, there is no administrative server or agent on machine where theservlet redirector runs. Instead, scripts are used to configure the Web server plugin to communicate with the servlet redirector, start the servlet redirector, and stop theservlet redirector. The thin servlet redirector is harder to administer than otherservlet redirector configurations. However, because no administrative server oragent processes are running on the Web server machine, more processing power is availableto handle HTTP requests.

Article 7.1.3.6.6, Thin servlet redirector sample topology,gives a more detailed description of this configuration of the servlet redirector.