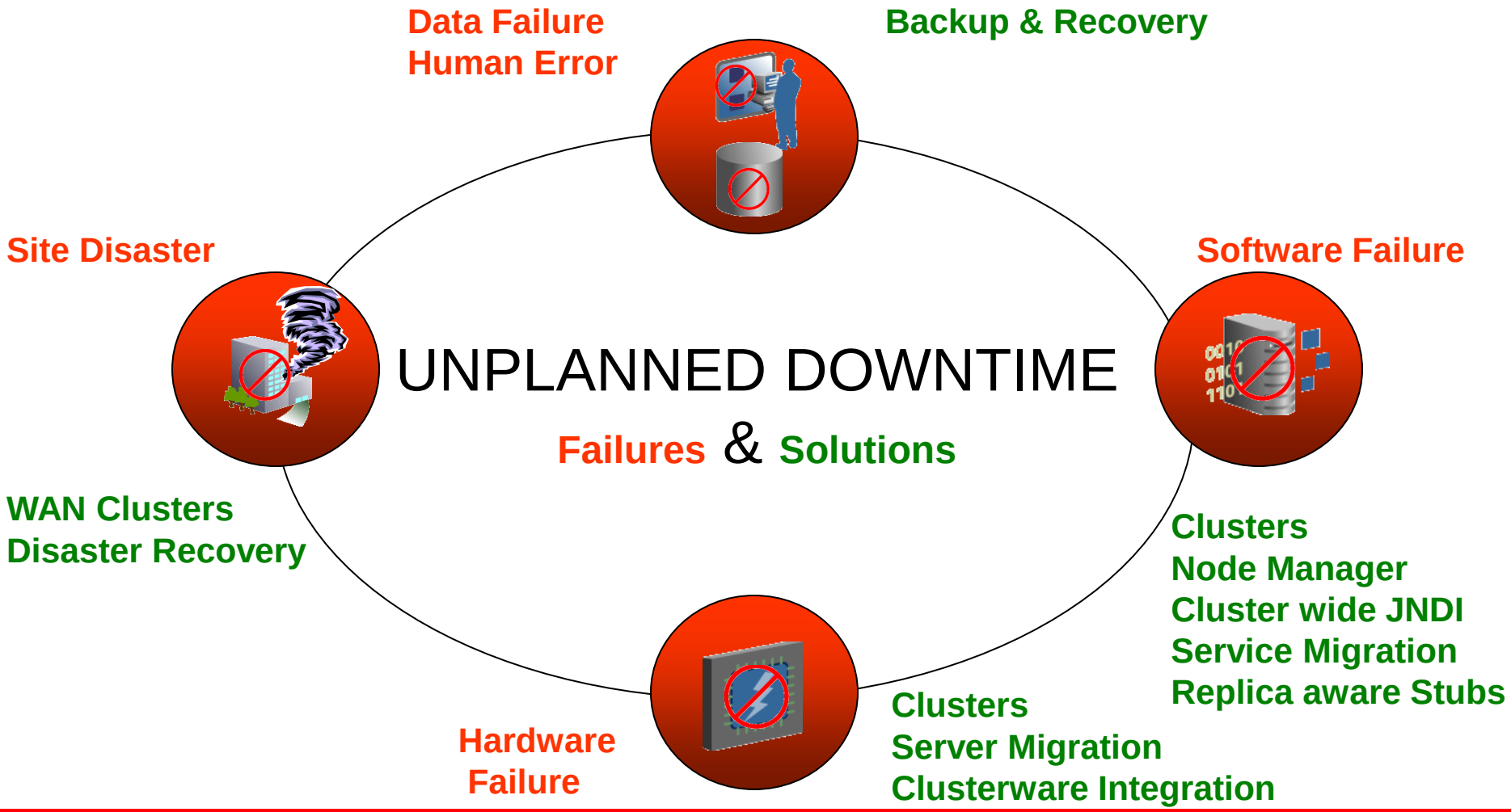
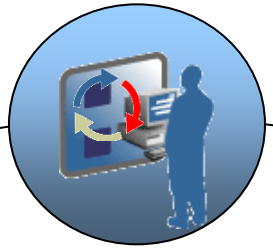
ORACLE®

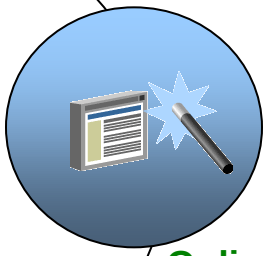
WebLogic High Availability Infrastructure
WebLogic Server 11gR1 Labs

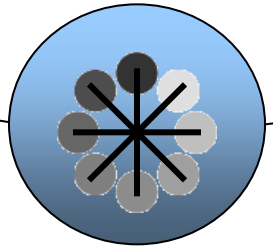
WLS High Availability



WLS High Availability







PLANNED DOWNTIME

Operations & Solutions

Deploy and Re-deploy Applications

Hot Deployment
Side By Side Deployment

Configuration Changes

Upgrades

Rolling cluster upgrade

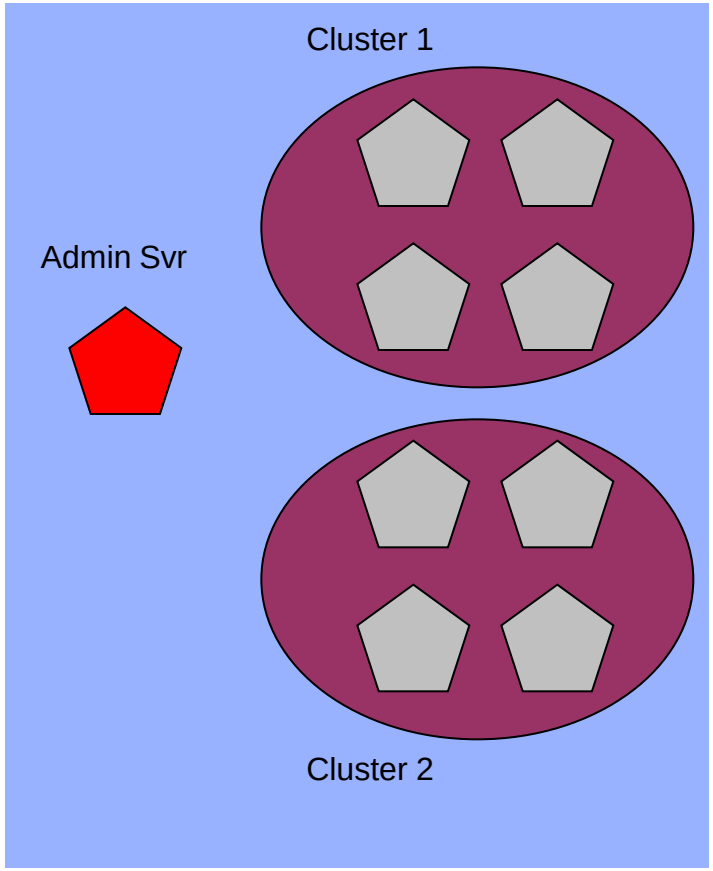
•Online Configuration Changes and warnings
•Batching changes/Deferred Activation

Transformations, scalability and topology extensions

•Cluster wide JNDI
•Dynamic Clusters
•Cloning

Cluster

- A group of Managed Servers running simultaneously and working together to provide increased scalability and reliability
  - Scalability: through parallelism
  - Reliability/Availability: through replication and redundancy
- Appears as a single instance to most clients.
- Enables some advanced features
  - Whole Server Migration
  - Service Migration, and clustered JMS destinations.



Benefits of Clustering

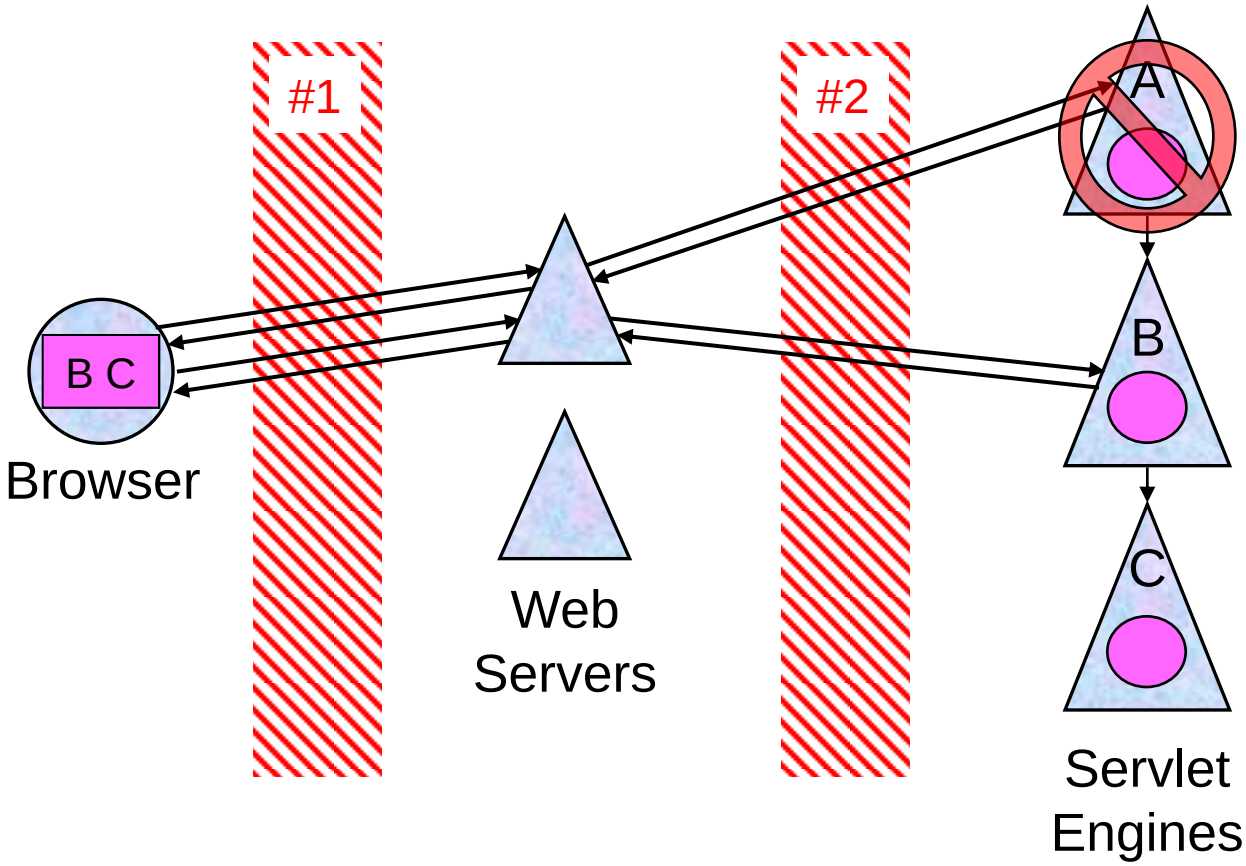
Concept	Description
Scalability	It provides more capacity for an application by adding servers, without having to make major architectural changes.
Load balancing	It distributes work (client requests and so on) across the members of a cluster.
Application failover	When an object in an application that is performing a task becomes unavailable, the object from the application in another server takes over to finish the job.
Availability	After a system failure on one server, it automatically continues ongoing work on another server.
Migration	After a system failure on one server, it continues ongoing work by moving the component to another server.

Load Balancing in a Cluster

- For JSPs and Servlets: load balancing is external
  - Web server proxy plug-in (round robin)
  - HTTP Proxy servlet (i.e., using WLS as a load balancer)
  - 3rd party hw or sw load balancer
- EJBs and RMI Objects: load balancing is done at connection
  - Objects are cluster-aware
  - Load balancing algorithm is stored in the clustered object's stub
  - Objects are available on all cluster members; remote objects connect/use according the LB algorithm in the stub
  - Load balancing algorithms: Round robin, weighted, random, server affinity

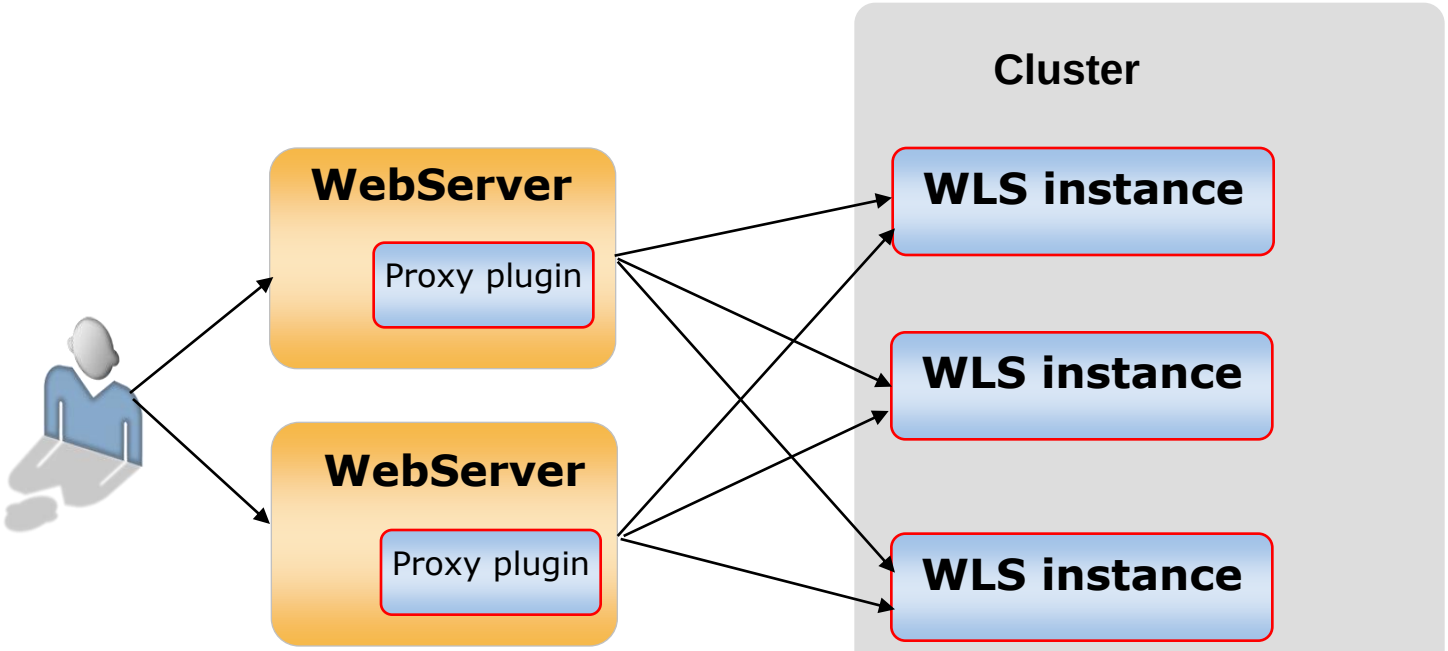
Session State Replication

JSPs and Servlets



Web Cluster

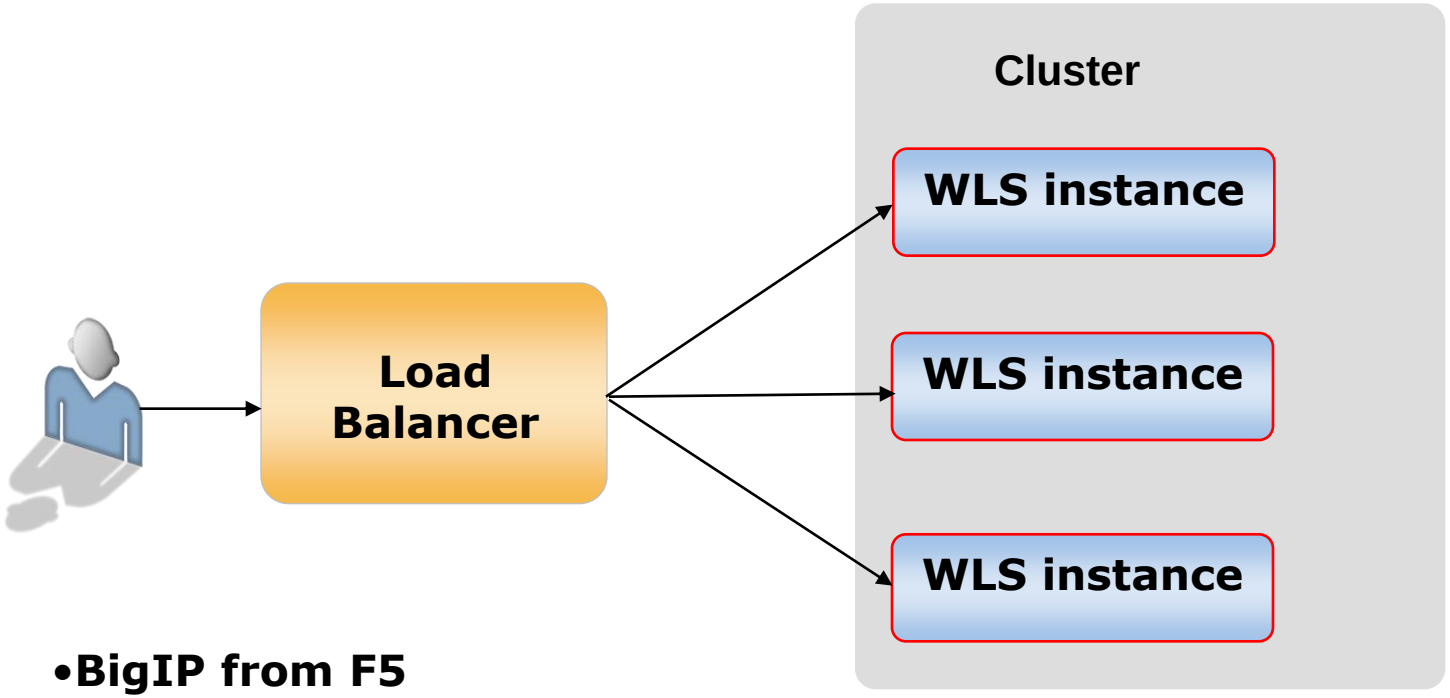
WebServer with proxy plug-in



- iPlanet/SunOne
- Apache
- IIS
- WLS with HttpCLusterServlet

Web Cluster

External Load Balancer



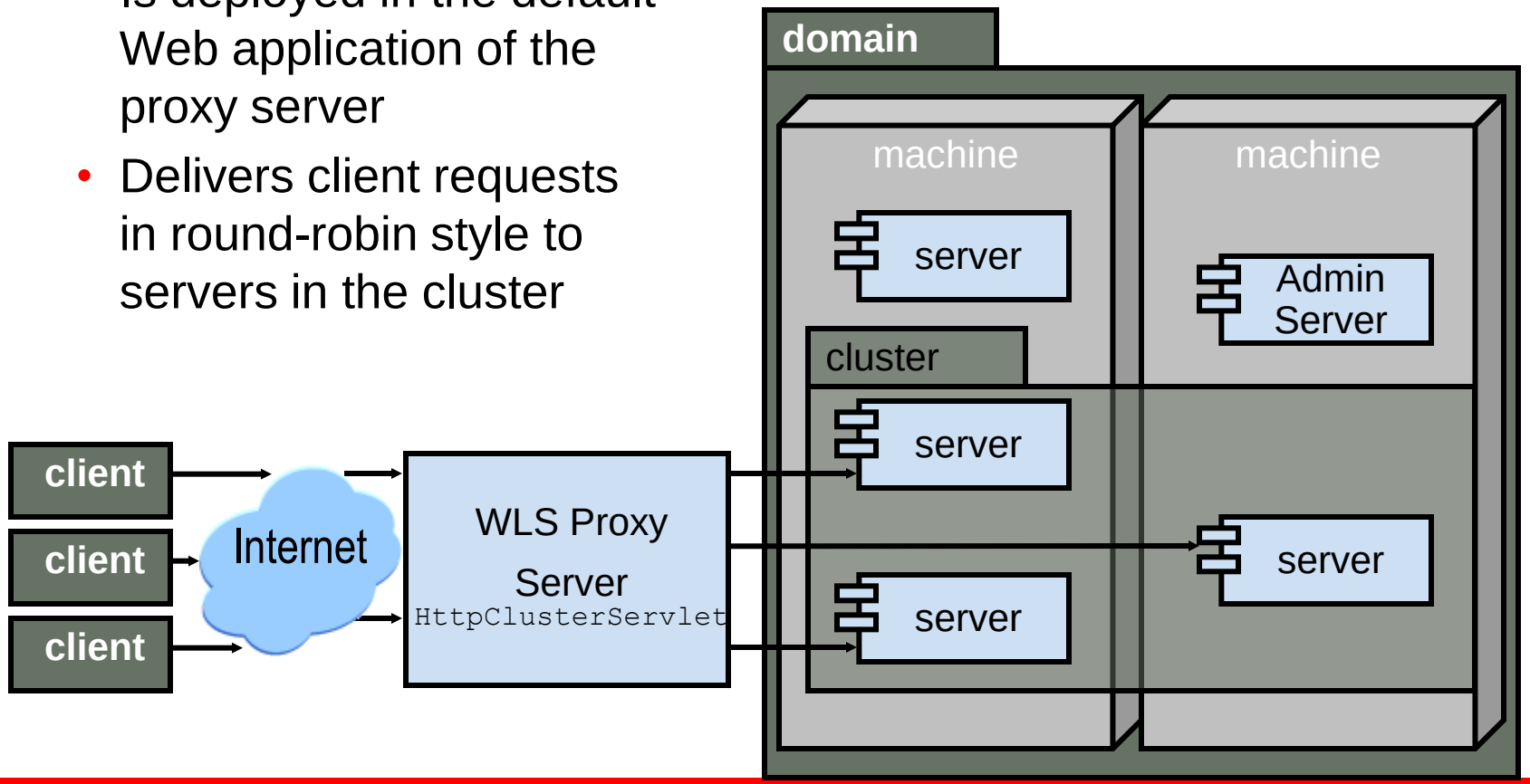
- BigIP from F5
- Alteon from Nortel
- Cisco

OHS as Proxy Web Server

- Oracle HTTP Server (OHS) is a Web server that:
  - Is based on Apache
  - Serves static and dynamic content
  - Supports content generation in many languages, such as Java, C, C++, PHP, PERL, or PL/SQL
  - Contains a WebLogic Server plug-in (`mod_wl_ohs`) by default
  - Can be easily integrated with other Oracle Fusion Middleware components
  - Can be managed using the Fusion Middleware Control along with other components

WLS `HttpClusterServlet`

- `HttpClusterServlet`:
  - Is deployed in the default Web application of the proxy server
  - Delivers client requests in round-robin style to servers in the cluster

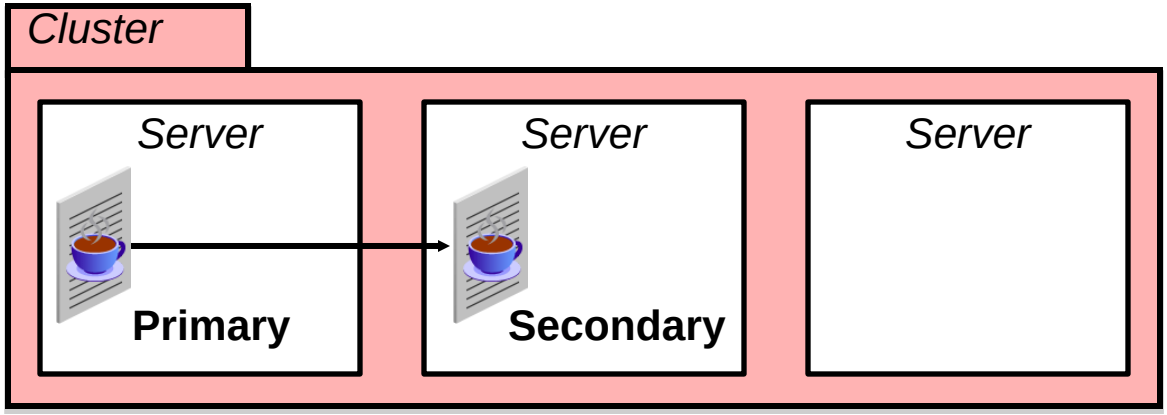


HTTP Session Failover

- Web applications use HTTP sessions to track information in server memory for each client.
- By default, when a client fails over to another server in the cluster, its session information is lost.
- Oracle WebLogic Server supports several *Session Replication* strategies to recover sessions from failed servers:
  - In-memory replication
  - JDBC replication
  - File replication
  - Using Coherence for Session replication
- Session persistence is configured using the `<session-descriptor>` element in the `weblogic.xml` deployment descriptor file.

HTTP Session: In-Memory Replication

- Each user's session always exists on two servers:
  - Primary
  - Secondary
- Every update to the primary session is automatically replicated on the secondary server, either synchronously (default) or asynchronously (batch).
- Oracle WebLogic Server uses nonpersistent cookies to track the primary and secondary servers for each client.



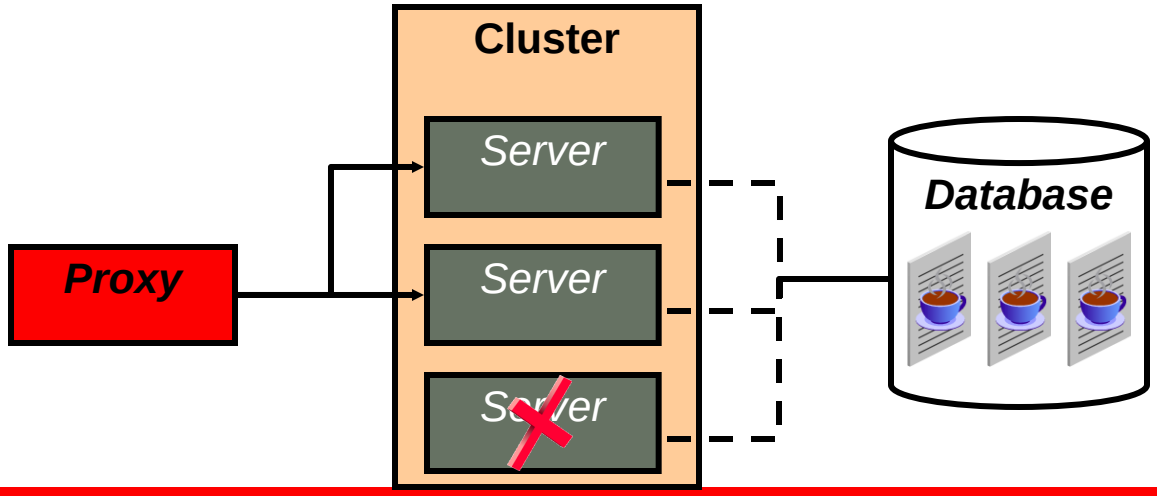
Configuring In-Memory Replication

- 1. Configure the proxy server (if applicable).
- 2. Optionally, define replication groups or machines, or both.
- 3. Specify the persistence type in the `weblogic.xml` deployment descriptor; the options include:
  - `replicated`
  - `replicated-if-clustered`
  - `async-replicated`

```
...
<session-descriptor>
   <persistent-store-type>replicated</persistent-store-type>
</session-descriptor>
...
```

HTTP Session: Replication Using JDBC

- HTTP sessions can be persisted to a database using a common JDBC data source.
- The required Data Definition Language (DDL) file is available in the documentation.
- *All* members of the cluster have access to any client's session for failover purposes (no primary or secondary).



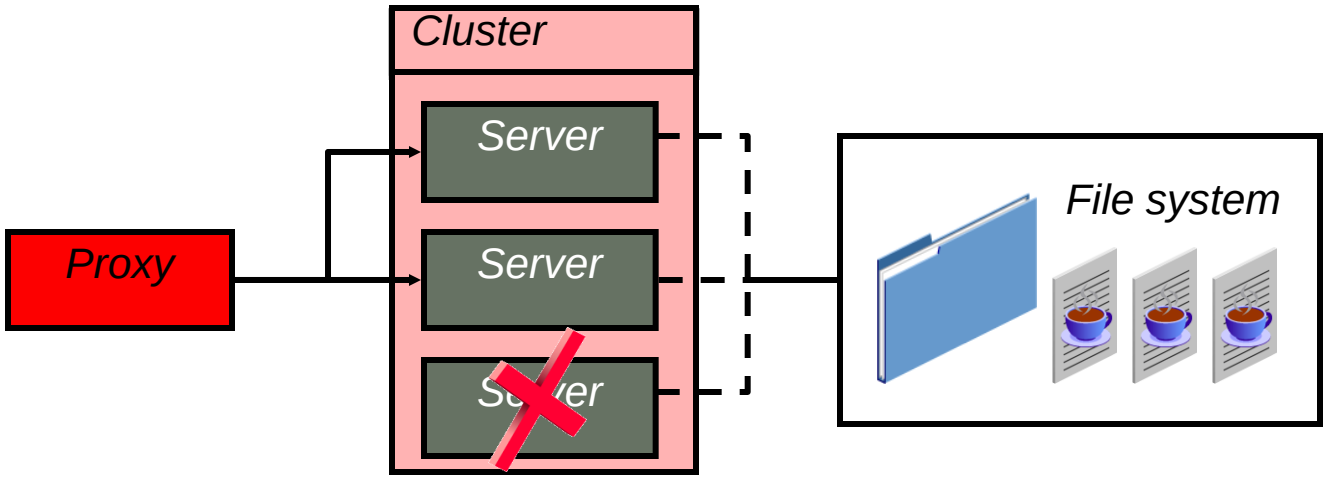
Configuring JDBC Replication

1. Create `WL_SERVLET_SESSIONS` table in the database.
2. Create a JDBC data source that has read/write privileges for your database.
3. Configure JDBC session persistence in the `weblogic.xml` deployment descriptor.

```
...
<session-descriptor>
   <persistent-store-type>jdbc</persistent-store-type>
   <persistent-store-pool>MyDataSource</persistent-store-pool>
</session-descriptor>
...
```

HTTP Session Replication Using File

- File replication is similar to JDBC replication, but it persists sessions to a highly available file system.



Configuring File Replication

1. Create a folder shared by all servers on the cluster on a highly available file system.
2. Assign read/write privileges to the folder.
3. Configure file session persistence in the `weblogic.xml` deployment descriptor.

```
...
<session-descriptor>
   <persistent-store-type>file</persistent-store-type>
   <persistent-store-dir>/mnt/wls_share</persistent-store-dir>
</session-descriptor>
...
```

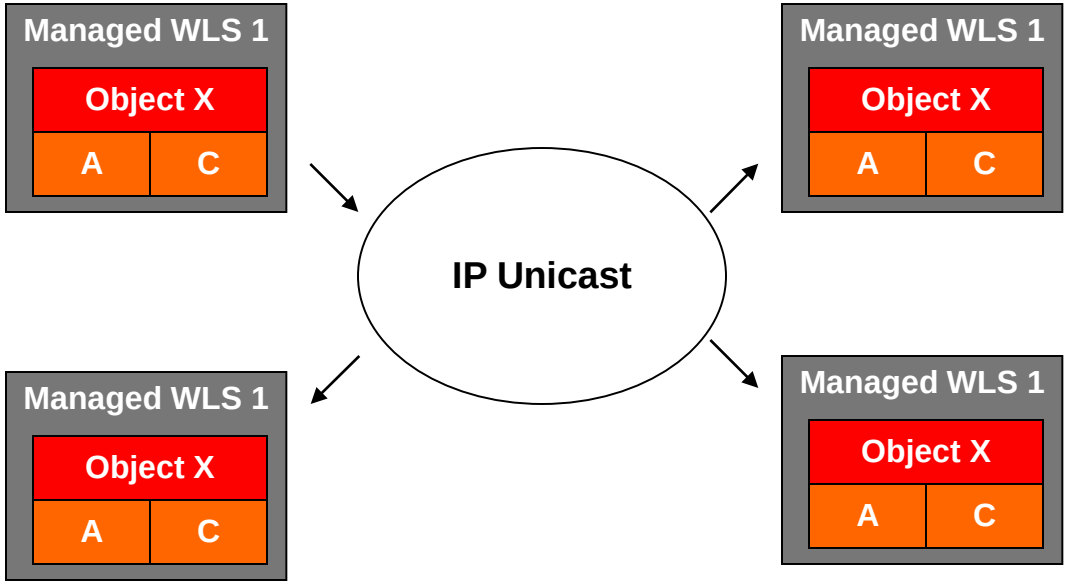
Replication Groups

- Replication groups:
  - Represent a subset of servers within a cluster
  - Help to determine the placement of secondary sessions (for example, avoid replicating within the same room)
  - Are not explicitly defined in the console-like machines and clusters
- WLS attempts to:
  - Send secondary sessions to servers that are assigned to the *preferred secondary replication group* of the primary server
  - Avoid sending secondary sessions to servers that are assigned to the same replication group as the primary server

HA with WebLogic Clustered JNDI

Surviving a Failed WebLogic Server

- Clusterable objects include EJB, JDBC, JMS, Custom Objects
- Each server creates and maintains a local copy of cluster wide JNDI tree

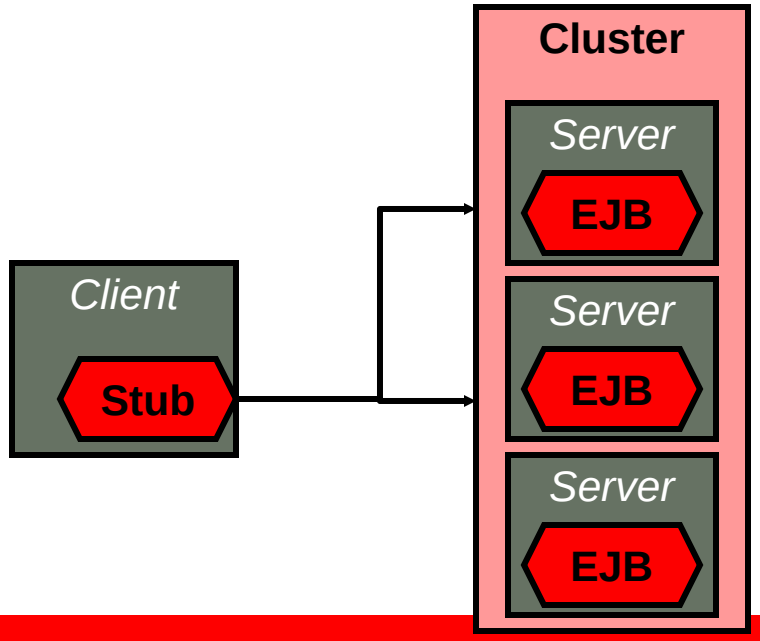


EJB/RMI Object Cluster
EJB invocations

- Stateless Session EJB
  - Method invocations are load balanced to all members where the EJB is deployed ("*replicas*") based on the load-balance policy (Round robin, weighted, random, server affinity)
- Stateful Session EJB
  - If not clustered, they are "pinned" to the member where created
  - If clustered, the state is replicated (to a secondary server instance) and the "*replica aware*" stub is aware of locations (primary and secondary).

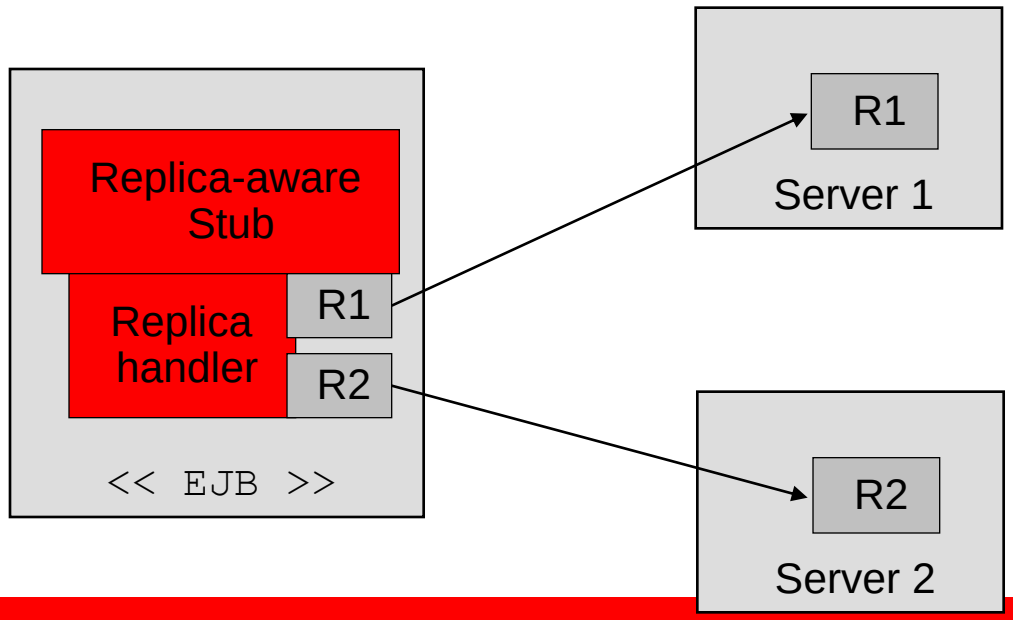
High Availability for EJBs

- WebLogic provides the EJB client applications with cluster-aware stubs that transparently perform load balancing and failover.
- You can enable and configure clustering for each EJB using the application deployment descriptor `weblogic-ejb-jar.xml`.



Clustering EJB Objects: Replica-Aware Stub

- Failover and load-balancing of EJBs is done with replica-aware stubs.
- Replica-aware stubs are generated at compile time for clusterable EJBs.



Configuring EJB Clustering in Deployment Descriptors

- Clustering of EJB based on version 2 are configured in the application-specific deployment descriptors.
- When using clustering based on EJB version 3.0, you can use the deployment plans to implement clustering.

- A snippet from `weblogic-ejb-jar.xml`:

```
...
<stateless-clustering>
   <stateless-bean-is-clusterable>True
   </stateless-bean-is-clusterable>

   <stateless-bean-load-algorithm>random
   </stateless-bean-load-algorithm>
   ...
</stateless-clustering>
...
```

Load-Balancing Clustered EJB Objects

- WebLogic Server supports the following load-balancing algorithms for clustered EJB objects:
  - Round-robin
  - Weight-based
  - Random
  - Parameter-based routing (programmatic)
- Server affinity configuration enables calls to objects to remain with the same server and minimizes client-side load balancing.

Server Communication in a Cluster

- WebLogic Server instances in a cluster communicate with one another using:
  - IP sockets, which are the conduits for peer-to-peer communication between clustered server instances
  - IP unicast or multicast, which server instances use to broadcast availability of services and heartbeats that indicate continued availability
- Multicast broadcasts one-to-many communications among clustered instances.
- Unicast is an alternative to multicast to handle cluster messaging and communications. The unicast configuration is much easier because it does not require cross-network configuration that multicast requires.

One-to-Many Communications

- Oracle WebLogic Server uses one-to-many communication for:
  - Clusterwide JNDI updates
  - Cluster “heartbeats”
- Because all one-to-many communications occur over IP multicast, when you design a cluster, consider the following factors:
  - If your cluster spans multiple subnets, your network must be configured to reliably transmit messages.
  - A firewall can break IP multicast transmissions.
  - The multicast address should not be shared with other applications.
  - Multicast storms may occur.

Considerations When Using Unicast

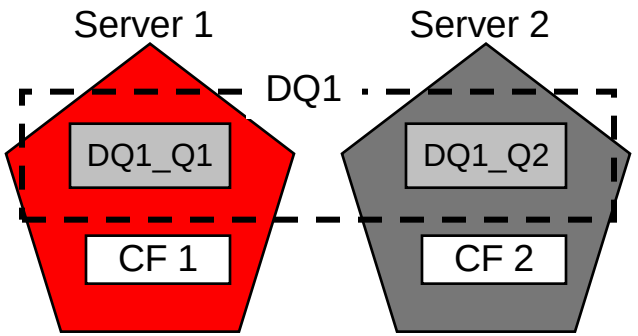
- Unicast messaging type:
  - Is much easier to configure because it does not require cross-network configuration that multicast requires
  - Reduces potential network errors that can occur from multicast address conflicts
- You cannot mix and match cluster messaging types within a cluster.

HA for JMS Infrastructure

Continued ability to send and receive messages	Distributed Destinations
All messages sent are processed	Whole Server and Service Migration
Seamless client failover	Automatic Reconnect
Continued ability to send when no remote servers are available	Store and Forward Client SAF

HA Scenario – A Server Fails, Messages are Trapped

- Server is hosting JMS destinations
- Messages are persisted (file or database storage)
- Server fails; messages are trapped



- Solutions:
  - Restart the server
  - Restart the server in another location (Whole Server Migration)
  - Restart the JMS service in another location (Service Migration)

Automatic Service Migration

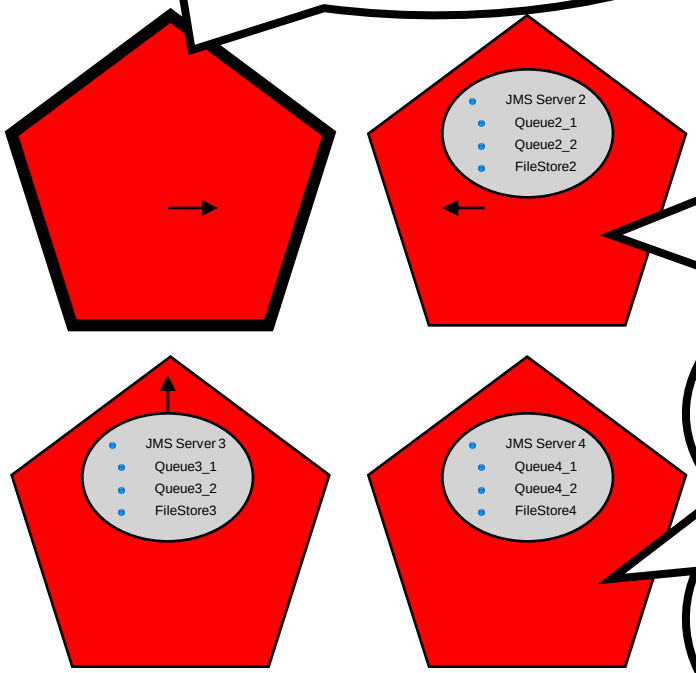
- Service Migration can be triggered because of
  - Bad Service Health
  - Server Shutdown
  - Server Failure
  - Machine Failure
  - Network Failure

Migratable Targets

- Migratable Services are deployed on a Migratable Target (MT).
- Allows grouping of targeted migratable services that should move/migrate together.
  - For JMS, target the JMS server/SAF Agent/Path Service and the persistent store to the same migratable target
  - The persistent store needs to store messages where they will be available after migration – on either a shared disk, a SAN, or in a database
- High availability is achieved by migrating a migratable target from one clustered server to another.

How a[illegible]MS migration works

Servers in a cluster compete to be the cluster leader. The cluster leader grants leases to servers to host migratable targets



Other servers poll the lease master and compete for the lease to host a migratable target. The lease master grants a lease to the most appropriate host.

it's le[illegible]
migratable target.

[illegible]
groups of deployed objects that migrate together as a unit

Automatic Service Migration Configuration

- Cluster Setup
- Leasing Configuration
- Migratable Target configuration
- Migration Polices
- Shared Access to the Persistent Store
  - File Store - SAN, Dual ported SCSI disks
  - JDBC Store – JDBC Multipool connection

Whole Server Migration
WebLogic High Availability

WebLogic Cluster Services

- Typically deployed homogeneously, enabling transparent fail over
- “Pinned services" such as JMS and JTA are targeted to individual servers, requiring migration

Before WLS 9.0, migration was manual

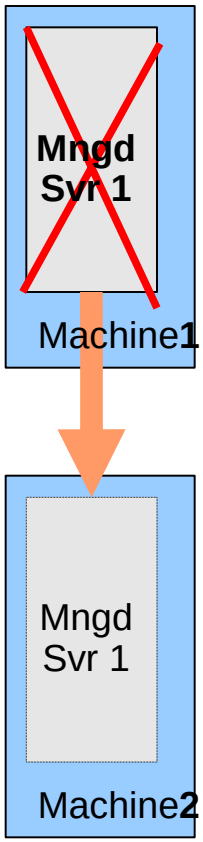
- Unless using an expensive third-party solution (Veritas, Sun Cluster, ...)
- Service Level Migration of services was a manual process

High Availability targets such as 5 9's

- 99.999% ≡ 5.26 minutes downtime per year
- Is there an automated process for migration of pinned services?

Whole Server Migration

Automatic Whole Server Migration

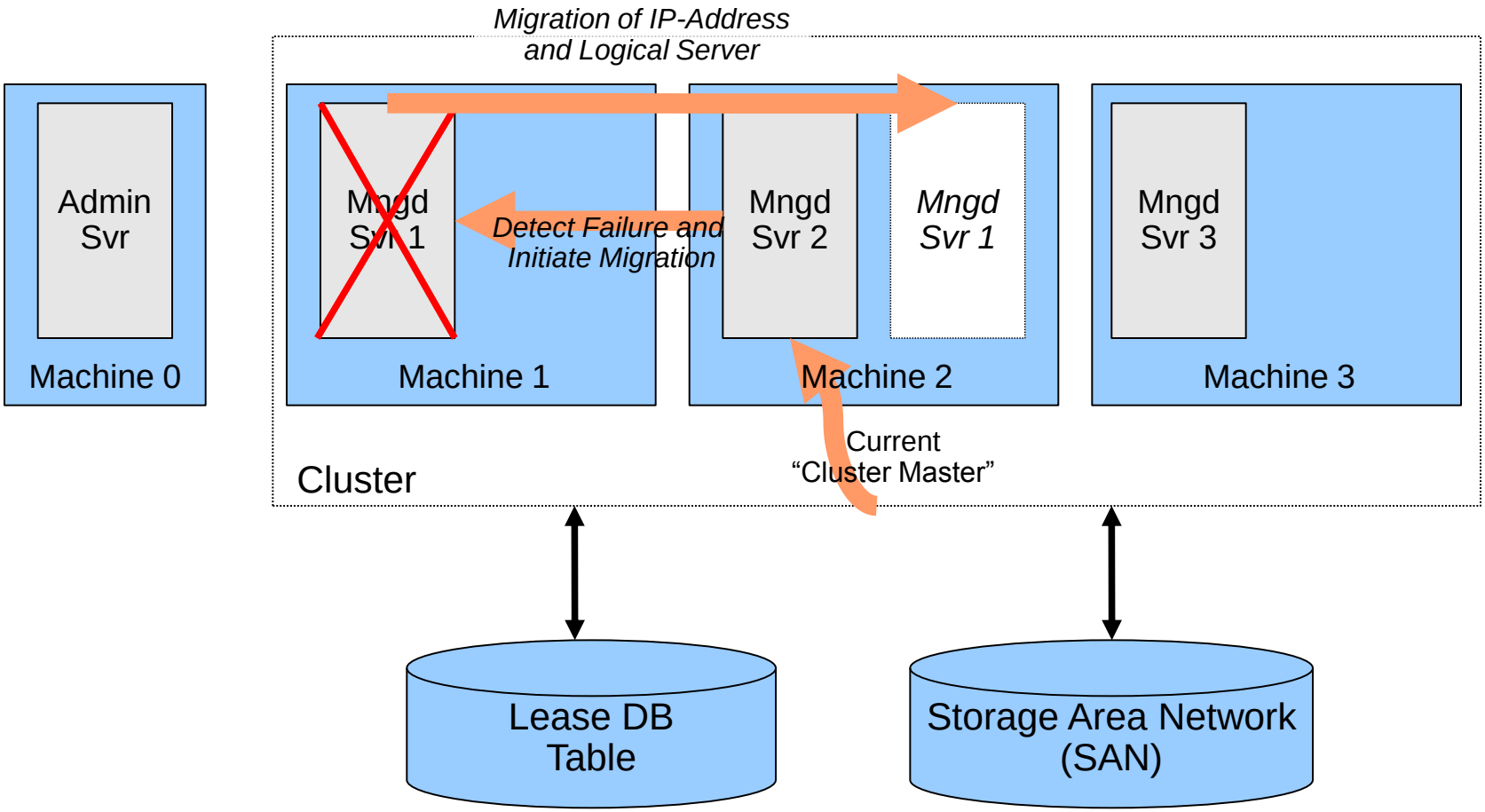


- Automated failure recovery of “pinned” services using migration, as opposed to fail over
- The failed server is re-started on a different machine
- Requires transition of IP address to new machine
- Maintains transactional integrity for JMS services

- WLS 10 provides automated **service migration** for some services
- JTA, Singleton, and Job Scheduler automated
- No JMS yet

Whole Server Migration

Server Migration Process



Whole Server Migration
Migration Requirements

Spare capacity

- Excess RAM and CPU on existing server host machines, or redundant 'idling' hardware

Node Manager

- Starts servers during migration
- Migrates floating IP addresses to new physical network devices,
- Mounts and unmounts shared disks

Highly Available Shared Storage (e.g. SAN)

- Provides access to JMS and JTA persistent stores from other machines

Network Time Protocol

- WebLogic hosts plus database (DB leasing) must be time synchronised

Whole Server Migration
Leasing

Leasing for election of Cluster Master

- Cluster Master manages current assignment of logical servers to physical machines
- Cluster Master monitors servers in the cluster and initiates server migration when any fail

Choice of leasing mechanisms

Database Leasing

- Requires a database containing lease table
- Consider using an highly available database (e.g. Oracle RAC)

Consensus Leasing

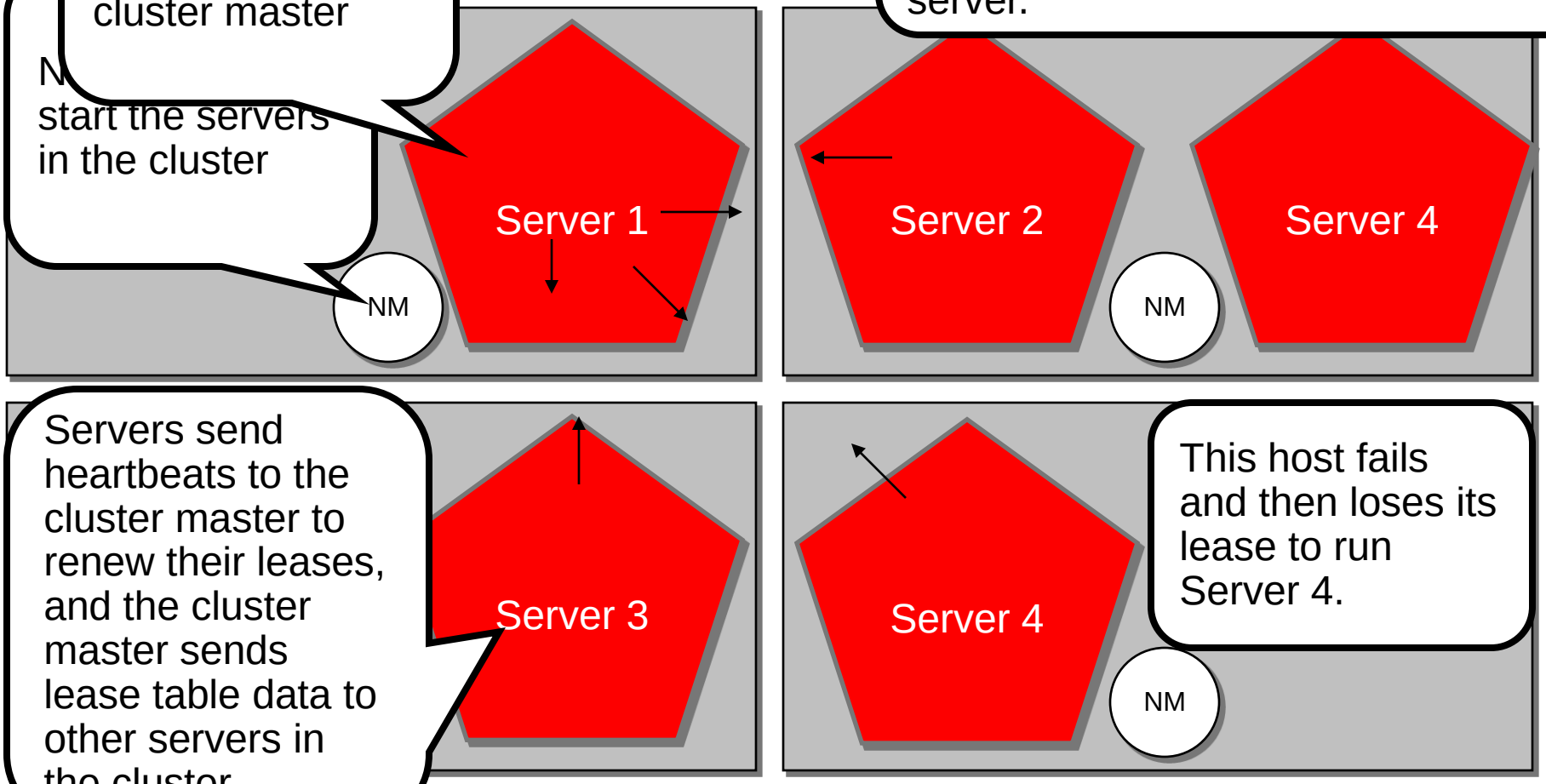
- Quorum based network algorithm that doesn't require a database
- Nominated cluster leader maintains in-memory lease records (replicated for fail over)
- Uses the Node Manager for server status checking

Scenario with W

The server with the earliest start time becomes the cluster master

N
start the servers in the cluster

All hosts compete to get the lease to run Server 4. The cluster master grants the lease to this server and calls the node manager to start the server.



Servers send heartbeats to the cluster master to renew their leases, and the cluster master sends lease table data to other servers in the cluster.

This host fails and then loses its lease to run Server 4.

Cluster Master

Not a Single Point of Failure

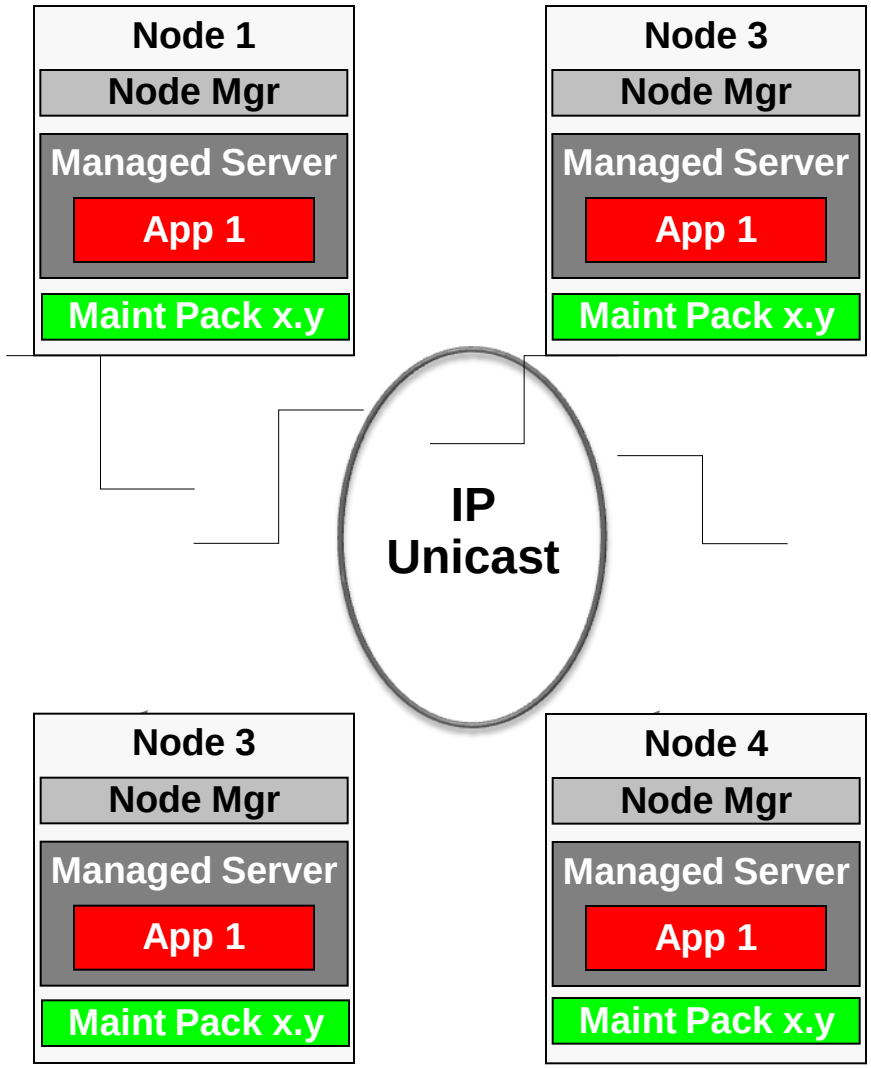
- Cluster Master is responsible for monitoring all servers' liveness

- What happens when the cluster master fails?
  - All servers compete to be the new cluster master
  - New cluster master is determined as the server that can write the cluster master record in the database (DB leasing) or with the earliest start time (consensus leasing)
  - Lease data is available either through replication in memory or in a database

HA with Rolling Upgrade

Zero Server Infrastructure Downtime

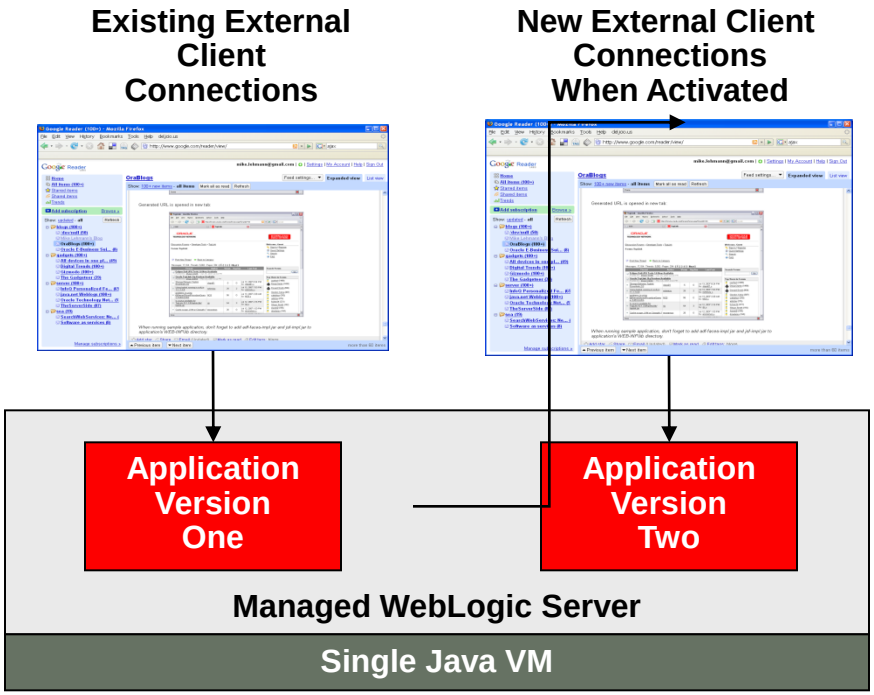
- **Upgrade a running cluster** with a patch, maintenance pack, or minor release without shutting down the entire cluster
- During the rolling upgrade of a cluster, **each server** in the cluster **is individually upgraded** and restarted while the **other servers** in the cluster **continue to host** your application
- You can also **roll back** the patch, maintenance pack, or minor release in a similar fashion

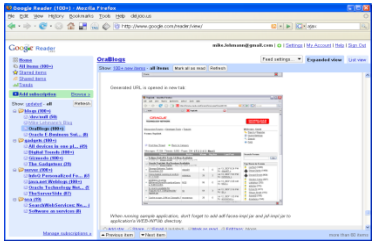


HA with Side by Side Deployment

Zero Application Downtime

- ☒Newer version of application deployed **side-by-side** with older version **in same JVM**
- ☒Clients already connected continued to be served by older version
- ☒New clients connect to newer version
- Test versions before opening up to users
- Rollback to previous versions
- Automatic retirement – graceful or timeout





Internal Client Connections

Test First in Administrative Mode