



LinuxWorld Conference & Expo

Server Farms and XML Web Services

Jorgen Thelin, CapeConnect Chief Architect
PJ Murray, Product Manager
Cape Clear Software



WWW.CAPECLEAR.COM

CAPE CLEAR™
Web Services Software



Objectives

- ✦ What aspects must a developer be aware of when a Web Services will be run in clustered environment such as a server farm?
- ✦ Do Web Services implementations need to be "cluster aware", or can this be handled transparently by the runtime platform?
- ✦ We revisit the subject of why keeping Web Services implementations as stateless as possible really helps in these circumstances, and the effect of using session-based facilities on scalability.





Basic Definitions

- ✦ Server Farms are loosely coupled clusters of *hardware, networks, and software*
- ✦ Web Services are *software components* with a well defined *XML interface* that can be incorporated in a distributed application – *Based on Standards*



WWW.CAPECLEAR.COM

CAPE CLEAR™
Web Services Software



Why cluster?

- ✧ High Availability
 - Transparent backup and failover, redundancy of systems
 - Cheaper than tightly coupled multiprocessor system
- ✧ Parallel processing/Performance
 - E.g. Beowulf
- ✧ Server Integration
 - Different servers perform different tasks
 - For example, WAS – Web Application Server





Why cluster?

- ✦ Scalability
 - More scalable than multiprocessor systems
 - Capacity on demand possible

- ✦ Automated cross-platform system management

- ✦ Failover and back-up

- ✦ Load Balancing
 - E.g. Linux Virtual Server project



WWW.CAPECLEAR.COM

CAPE CLEAR™
Web Services Software



Determining Requirements

- ✦ Downtime has a cost
 - Typically expressed as a cost per hour
 - Makes it easy to calculate ROI on clustering projects
 - Reduces other costs – disaster recovery
- ✦ Five 9s
 - Currently considered the industry standard for highly-available distributed systems
 - Means less than 6 minutes of annual downtime





What are critical factors?

- ✦ Size: CPU speed, memory speed, memory size, cache size, disk speed, disk size, network bandwidth?
- ✦ Too many interdependent factors
- ✦ Benchmark, find bottleneck, fix it. Repeat.
- ✦ For Web Services, the bottleneck is usually the network rather than the CPU
- ✦ Many general guidelines: dual CPU machines scales better – application runs uninterrupted on one CPU while the other CPU handles all network interrupts





Linux Virtual Server Project

- ✦ The Linux virtual server is a highly scalable and highly available server built on a cluster of real servers.
- ✦ The architecture of the cluster is transparent to end users, and the users see only a single virtual server.
- ✦ <http://www.linuxvirtualserver.org>

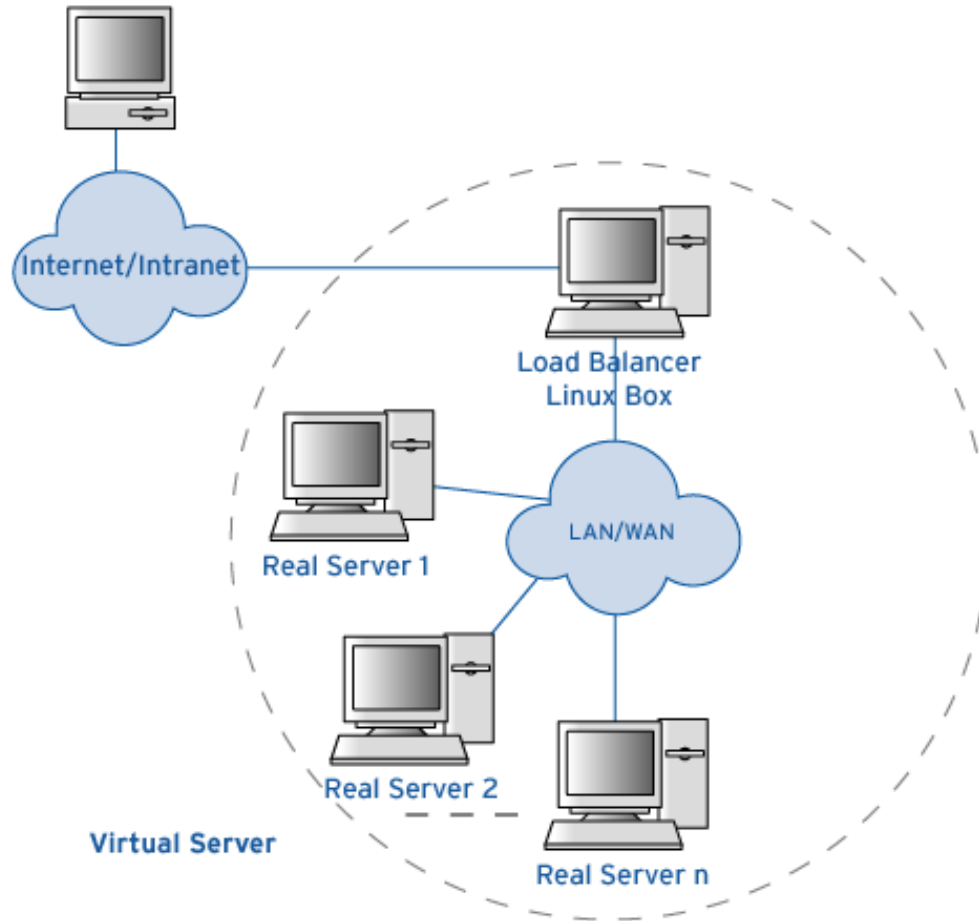


WWW.CAPECLEAR.COM

CAPE CLEAR™
Web Services Software



Virtual Server Architecture



WWW.CAPECLEAR.COM

CAPE CLEAR™
Web Services Software



Load Balancing Algorithms

- ✦ Basic round robin
- ✦ Weighted round robin
 - Good for servers with different capacity
- ✦ Least-Connection
 - Use least busy server
- ✦ Weighted Least-Connection
 - Percentage share of active connections is ratio to its weights





Web Services Defined (Again)

- ✦ Self-describing, self-contained modular entities:
 - Platform and language independent
 - Implementation neutral
 - Open standards based
 - Loosely coupled
 - Programmatically connect business processes
 - Typically requiring integration with existing systems



WWW.CAPECLEAR.COM

CAPE CLEAR™
Web Services Software



Web Services

- ✦ Web Services are software components:
 - Defined in WSDL.
 - Remotely accessible via SOAP.
 - Registered in a UDDI.



WWW.CAPECLEAR.COM

CAPE CLEAR™
Web Services Software



WSDL

✦ WSDL:

- The Web Services Description Language.
- Describes the methods & parameters.
- Like a “users manual” for the Web Service.
- Is based on XML.



WWW.CAPECLEAR.COM

CAPE CLEAR™
Web Services Software



SOAP

✦ SOAP:

- The Simple Object Access Protocol.
- Defines the message contents and processing.
- Like a “transport” for calling Web Services.
- Is based on XML and runs on HTTP(S).



WWW.CAPECLEAR.COM

CAPE CLEAR™
Web Services Software



UDDI

✦ UDDI:

- The Universal Discovery, Description, & Integration.
- Stores Web Services descriptions and endpoints.
- Like a “Yellow Pages” for Web Services.
- Ultimately will require advanced DNS type features
 - Round robin endpoint allocation

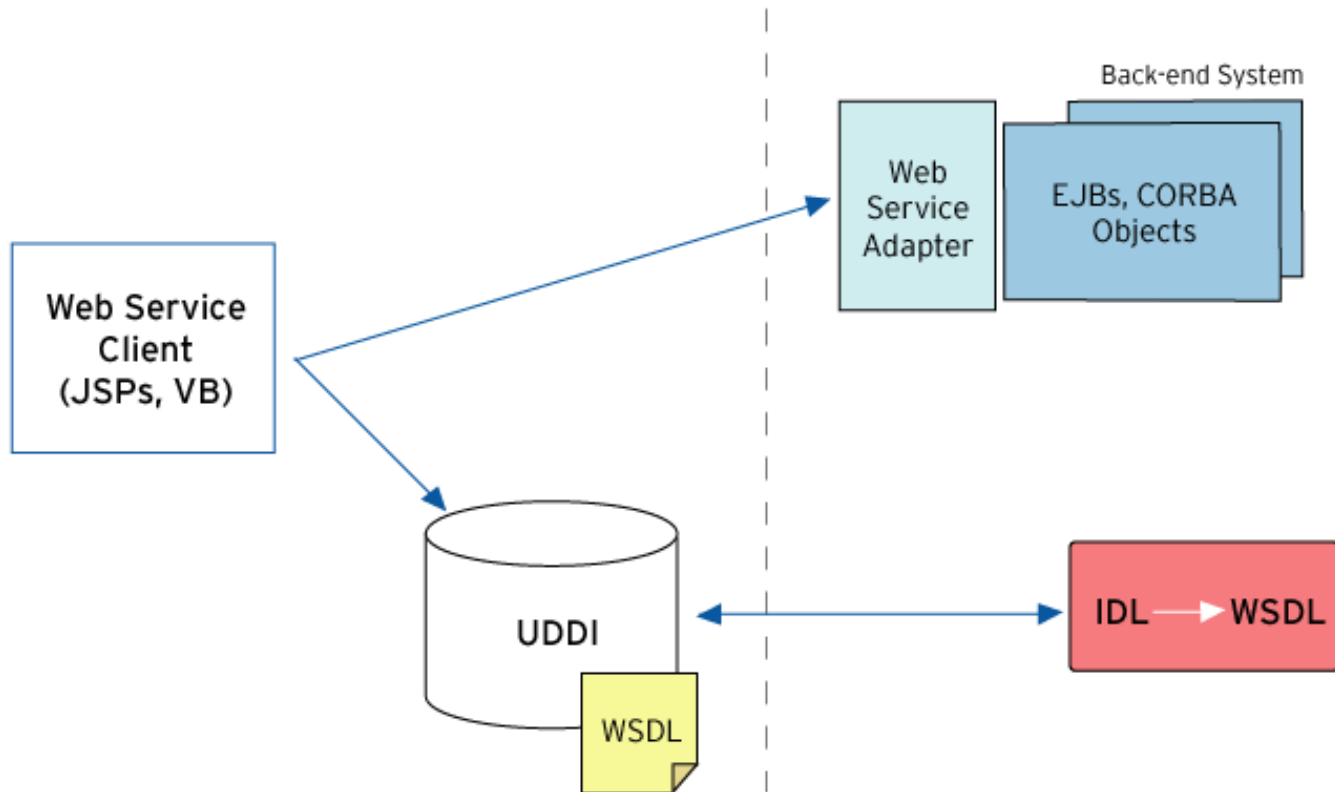


WWW.CAPECLEAR.COM

CAPE CLEAR™
Web Services Software



Web Services Example





Web Services Standards

✦ Web Services standards look familiar:

- WSDL ↔ Java interfaces, CORBA IDL
- SOAP ↔ Java RMI, CORBA IIOP
- UDDI ↔ JNDI, CosNaming, CosTrader

✦ But they...

- Abstract Java, CORBA and .NET technologies.
- Are being added to many packaged applications.

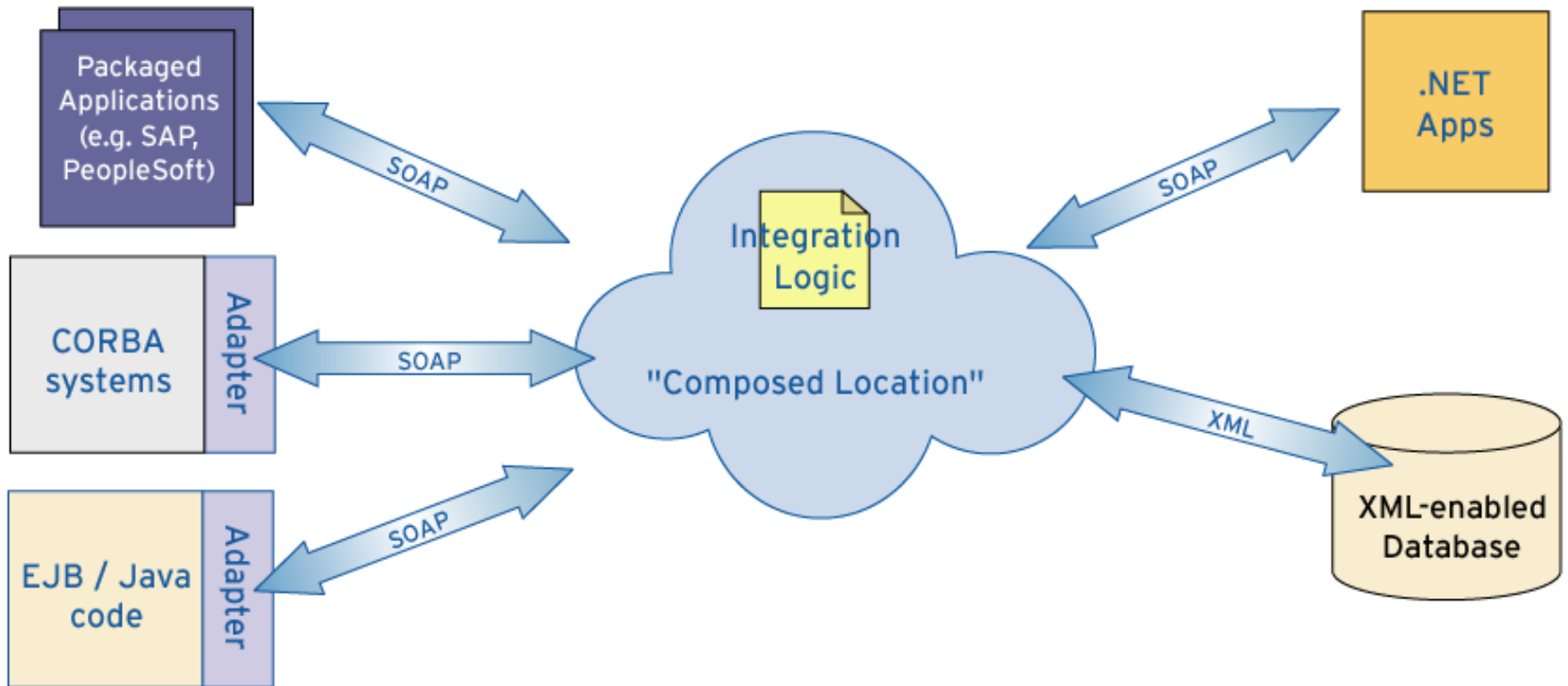


WWW.CAPECLEAR.COM

CAPE CLEAR™
Web Services Software



EAI via Web Services



WWW.CAPECLEAR.COM

CAPE CLEAR™
Web Services Software



Why Clustering *and* Web Services?

- ✦ On the back-end (producer)
 - Expose - business logic
 - Integrate - lightweight EAI – technology integration
- ✦ On the front-end (consumer)
 - “Single Point of Access” for disparate client types
 - HTTP-enabled client interfaces
 - Inter-application communication interface
 - Process inbound XML
- ✦ Private Directory Systems (UDDI)

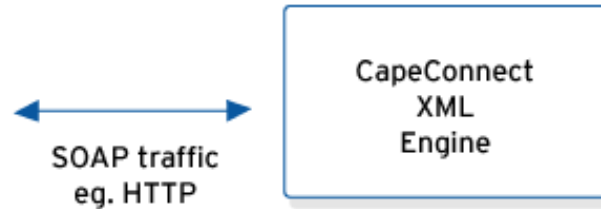


WWW.CAPECLEAR.COM

CAPE CLEAR™
Web Services Software



Standalone Web Services Platform



- ✦ Single instance of CapeConnect server
- ✦ Single server host
- ✦ Several points of failure

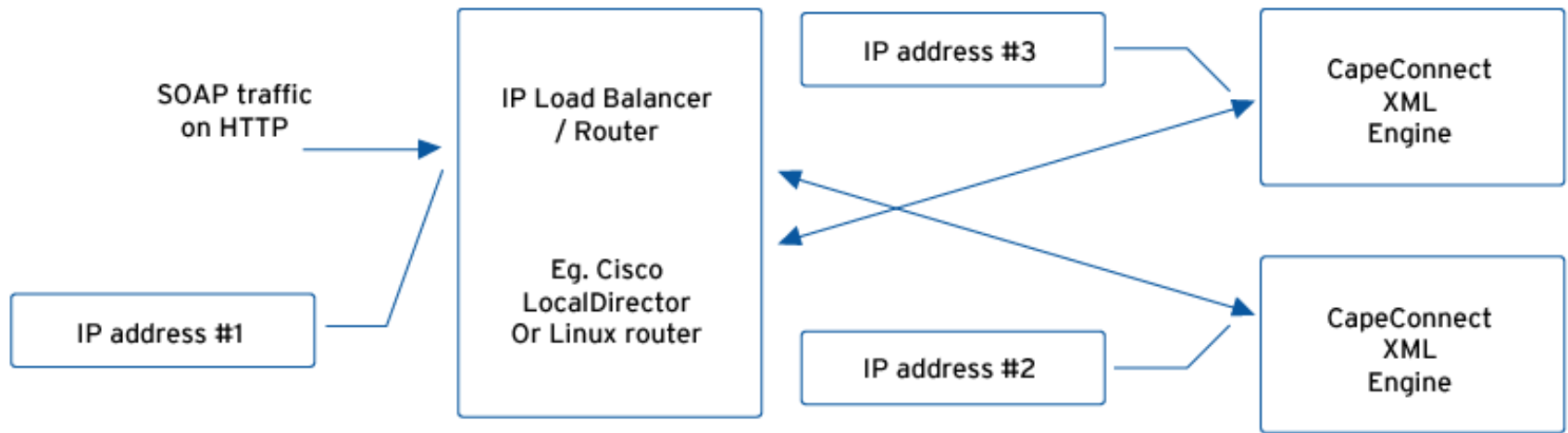


WWW.CAPECLEAR.COM

CAPE CLEAR™
Web Services Software



Web Services Platform with clustering IP router



WWW.CAPECLEAR.COM

CAPE CLEAR™
Web Services Software



CapeConnect + IP Router

- ✦ Multiple CapeConnect server instances
- ✦ Multiple Linux server hosts
- ✦ Router provides single endpoint IP address to the outside world
- ✦ Each server instance runs on a host with its own IP address



WWW.CAPECLEAR.COM

CAPE CLEAR™
Web Services Software



CapeConnect + IP Router

- ✦ Router provides failover among server hosts
- ✦ Router can provide load balancing among server hosts
 - BUT - requires completely stateless Web Services
- ✦ Router is the only single point of failure



WWW.CAPECLEAR.COM

CAPE CLEAR™
Web Services Software



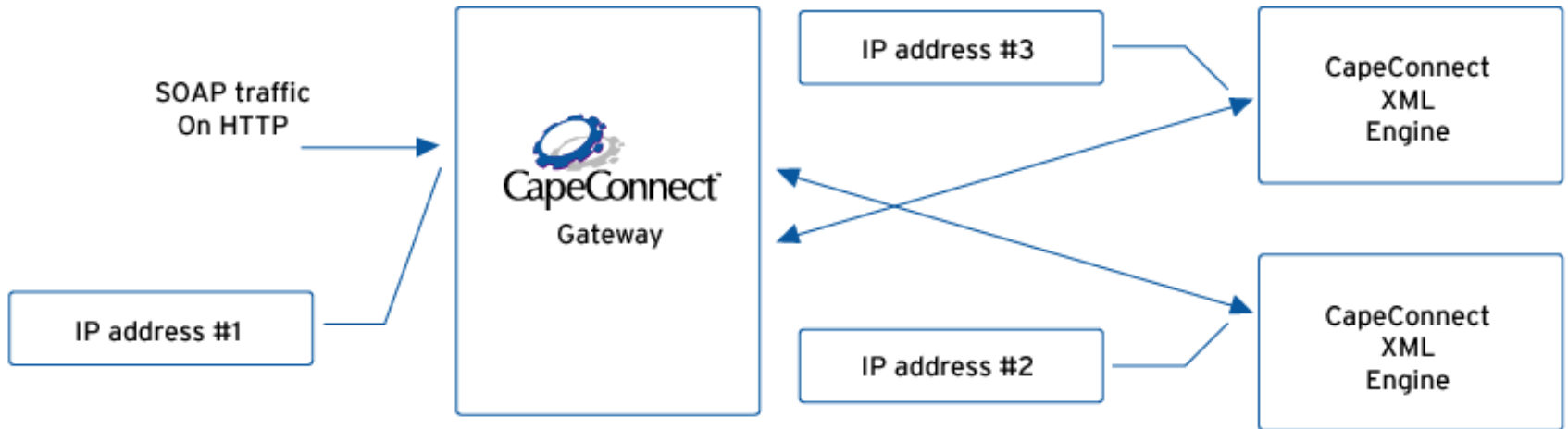
A Web Services Gateway

- ✧ Application level message router
 - Receives and forwards SOAP messages across network topology boundaries
 - Can be used to bridge between different transport schemes
For example: HTTP in, JMS out
- ✧ Usually placed in the DMZ for securely connecting Internet SOAP traffic into a corporate network
- ✧ Provides a very convenient place to build clustering and failover into the service hosting architecture for Web Services
- ✧ Also allows load balancing to be used across a cluster of Web Services platforms, thereby avoiding operational hotspots.





Web Services Platform with Web Services Gateway



WWW.CAPECLEAR.COM

CAPE CLEAR™
Web Services Software



CapeConnect + Gateway

- ✦ Functionally equivalent to using a Clustering IP Router
- ✦ Allows use of other transport types such as JMS between the Gateway and XML Engines
- ✦ Some processing (such as security checks) can be handled by the Gateway



WWW.CAPECLEAR.COM

CAPE CLEAR™
Web Services Software



CapeConnect + Gateway

- ✧ Gateway can provide “application failover” facilities such as when upgrading a service
- ✧ If the web services are not stateless, some state data needs to be shared between XML Engine instances – either by the middleware or application itself
- ✧ Gateway still provides a single point of failure

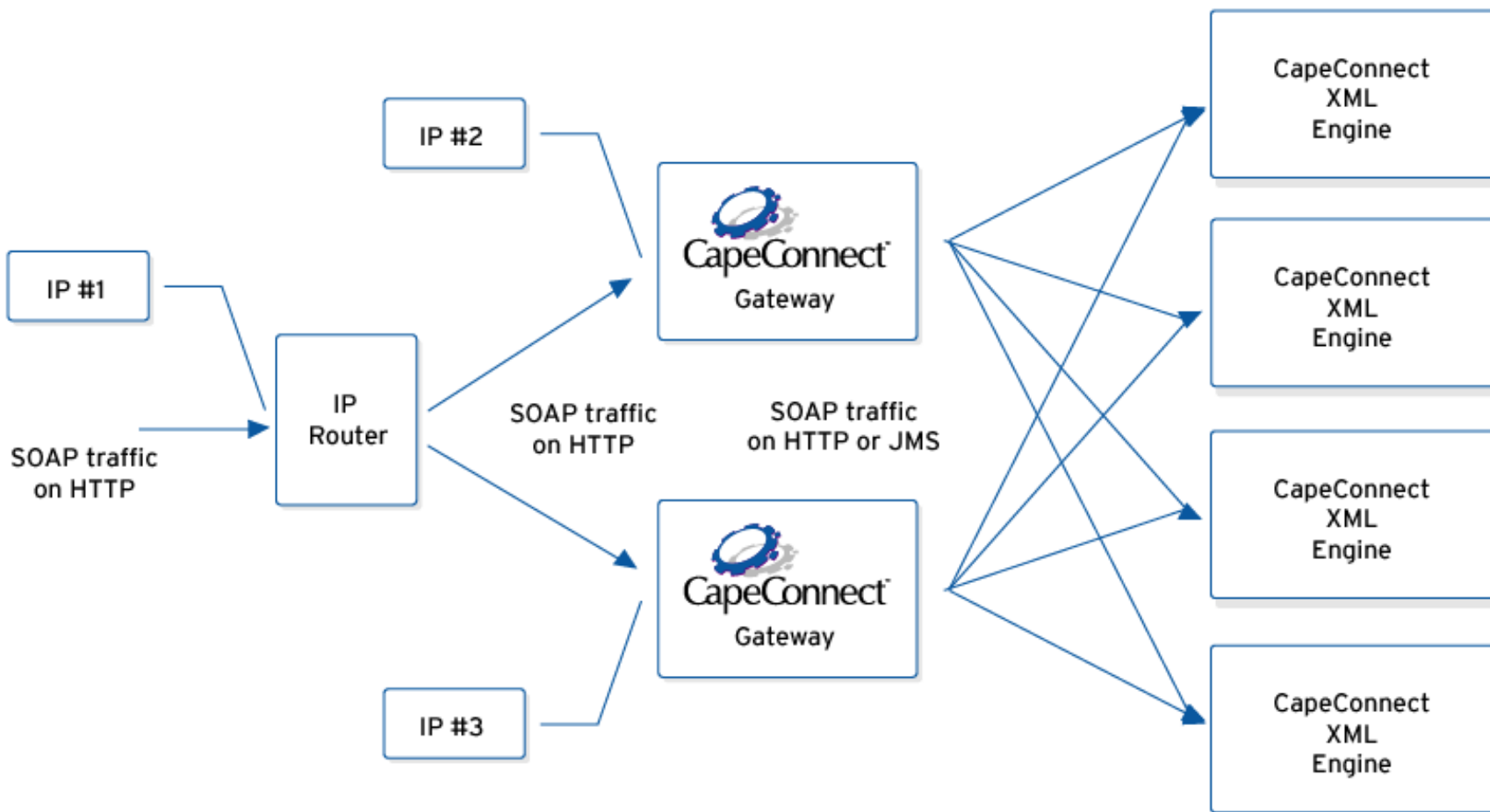


WWW.CAPECLEAR.COM

CAPE CLEAR™
Web Services Software



The Ultimate Web Services Cluster



WWW.CAPECLEAR.COM

CAPE CLEAR™
Web Services Software



The Ultimate Web Services Cluster

- ✦ Only single point of failure is the IP router which are generally very reliable
- ✦ CapeConnect Gateway makes “cluster management” tasks such as service upgrades easier due to the application level failover facilities



WWW.CAPECLEAR.COM

CAPE CLEAR™
Web Services Software



The Ultimate Web Services Cluster

- ✦ Any number of CapeConnect XML Engines can be deployed to handle the required service load
- ✦ Two Gateway instances should be sufficient to handle most scenarios, but more can be added if required



WWW.CAPECLEAR.COM

CAPE CLEAR™
Web Services Software



Service Development Considerations

Issues that Web Service developers need to consider for making their applications “cluster friendly”



WWW.CAPECLEAR.COM

CAPE CLEAR™
Web Services Software



The Need for Stateless Services

- ✦ The stateful-ness of a Web Services implementation is the main determinant of the ease of deployment into a cluster environment
- ✦ Stateless services deliver the maximum performance as there are no additional “overheads” on each call



WWW.CAPECLEAR.COM

CAPE CLEAR™
Web Services Software



Stateless Service Architectures

- ✦ If service components are completely stateless:
 - a wide range of configuration options can be used to create a completely fault tolerant deployment architecture
 - Scalability can be increased by simply adding more processing nodes
 - Service processors can easily be moved from node to node for load balancing purposes



WWW.CAPECLEAR.COM

CAPE CLEAR™
Web Services Software



Stateful Service Architectures

- ✦ If service components are stateful:
 - Failover has to be explicitly handled by either the service or the server hosting the service
 - Load balancing incurs a cost as service instance state needs to be reconstructed on the new node
 - Service applications have to be “cluster aware” to ensure any relevant state information is preserved in persisted data after each request
 - Preserving state information will add an overhead to all calls, which will ultimately reduce performance (response time and scalability)





Service Instance configuration

- ✦ Any configuration information required by the Web Services needs to be available to all nodes the service is deployed onto
- ✦ To use a single set of config data for all instances, needs to be stored in:
 - Config file on a shared file system drive accessible from all nodes
 - Database accessible to all nodes
- ✦ The Web Services will still need to be deployed into each node in the cluster





Service Upgrades

- ✦ One of the hardest part of managing a cluster / server farm is how to perform upgrades on individual applications
- ✦ Upgrade each processing node in the cluster in turn (see next slide)
- ✦ Requires “application level” router such as CapeConnect Gateway (IP Routers can’t handle this)





How To Handle Service Upgrades



General approach is:

- For each processing node in the cluster:
 1. Disable an application on one processing node at a time
 2. CapeConnect Gateway will route requests for that application to other nodes in the cluster
 3. Redeploy the application on the "offline" node
 4. Other applications on the "offline" node can still be active – only the application being upgraded is actually unavailable
 5. Bring the new application version back on line
 6. Gateway will start routing requests to that node again
- Proceed with the next node in the cluster until all service instances have been upgraded





Sessions

- ✦ Any use of “sessions” makes service invocations stateful
 - Requests need to be routed to the same processor node as the previous requests in this session – “affinity”.
 - Or, session state will need to be reconstructed if a request is sent to a different processing node from the last request in this session
- ✦ Load balancing algorithm needs to be session aware – often referred to as “sticky sessions” or “session affinity”





Conclusions

- ✦ The main aspect Web Services developers need to consider is the stateless-ness of their services
- ✦ Many deployment options are available for creating Web Services clusters
- ✦ Stateless services can be scaled more easily simply by adding more processing nodes
- ✦ Stateless services deliver higher performance



WWW.CAPECLEAR.COM

CAPE CLEAR™
Web Services Software