Euphorion

Euphorion Inc
1900 McCarthy Blvd #204
Milpitas, CA 95035

408.894.6800
408.894.6899 fax
http://www.euphorion.com/

# IBM WebSphere Architecture Optimization

## Hardware and System Software Solutions To Optimize WebSphere

## Notice of Confidentiality

*This document, and the material, ideas, concepts, and intellectual property contained within it, is the proprietary, confidential information of Euphorion™ Incorporated, of 1900 McCarthy Boulevard, Milpitas, California, 95035, USA. It may not be duplicated or modified, in whole or in part (including, but not restricted to electronic and physical copies), without the express written consent of Euphorion Incorporated.*

*This document is provided to eBay for the purpose of assessing the suitability of Euphorion Incorporated to undertake work on the behalf of eBay related to system implementation, tuning, architecture, and testing. The document may be distributed to appropriate personnel within eBay for the exclusive purpose of this assessment. The document may not be distributed to agents of eBay without the express written consent of Euphorion.*

*The policies, practices and procedures described in this document are guidelines that Euphorion reserves the right to modify or eliminate at any time, for any reason. Description of any policies practices and procedures contained in this document, and any representations communicated by a Euphorion Incorporated employee regarding this document, do not constitute a contract.*



*For information regarding duplication and distribution, please contact:*

*Jeff Hunter*
*President and CEO*
*Euphorion Incorporated*
*1900 McCarthy Boulevard, Suite 204*
*Milpitas, CA, 95035, USA.*
jeff@euphorion.com

# Guidelines

Euphorion's unique WebSphere solution approach to increase the performance, flexibility, scalability, reliability, and maintainability of high-transaction J2EE implementations is based on the following guidelines.

## Hardware

### More memory and more processors -- fewer different boxes

Configuring fewer, bigger boxes for WebSphere is advantageous not only because it saves maintenance costs (fewer boxes need fewer people) but also because it allows WebSphere to take advantage of the extra computing power during infrequent spikes such as java re-compiles. In addition, configuring and using fewer, more powerful machines postpones limits to the speed-up resulting from horizontal scale-up in the hardware architecture. Fewer mid-tier boxes prevent undo load on the back-end SAN or Data Source because the number and size of connections scales sub-linearly with traffic and is therefore more easily controlled. Finally, cost savings for the same computing power are realized using this strategy.

## Open Connections

### Reverse proxy queues, controls simultaneous connections

One of the most important resources in a large, high-traffic web environment is the number of web connections that are open simultaneously from web clients. WebSphere has an extremely large memory footprint for each connection it has open. Therefore it is imperative that some form of reverse proxy cache system or other solution be implemented that holds on to client web connections and drops connections to the web server connecting to WebSphere. Every connection to a web server that is configured to connect to WebSphere has this large memory requirement. Never serve static content from web servers that will connect to WebSphere.

Installing a reverse proxy results in the largest performance gain of any software configuration optimization for WebSphere.

# More Tiers

## Dedicated hardware -- single purpose, optimized servers

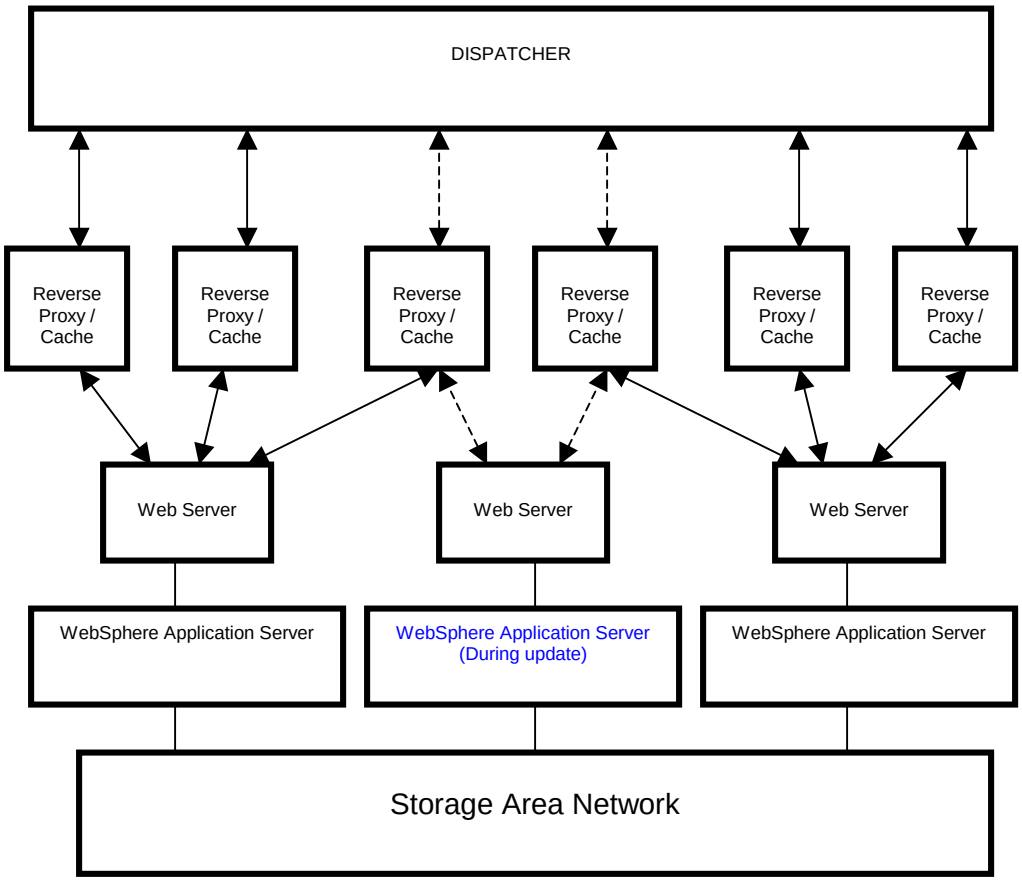
Separating web servers from the WAS servers helps isolate network and computing resource requirements from competing resources. In other application servers this separation is not recommended but with WebSphere it is a good idea.

# Avoid WebSphere Race Conditions

## More memory and more processors -- fewer different boxes

When source Java Server Pages (JSPs) and other Java code changes, WAS re-compiles it on-the-fly. Serialized Java will also be re-serialized dynamically. Unfortunately these "features" are very dangerous because when several hits to the same resources occur, several different re-compile processes or re-serialization processes start up. The system becomes unstable and eventually freezes. A release process is required that rolls each application server through one hit each to the Java elements then quickly and automatically brings each server back online.

Therefore a more flexible and finer-grained load balance solution is required than is available through most hardware solutions. We recommend using the reverse proxy software tier to perform this function as part of the update process when Java elements of the system have changed.




# Faster Recovery – not Fault Tolerance

## Fail-over reliably and maintain extra capacity

Hardware and system software tuning parameters underneath WebSphere that allow each server to recover from certain faults is not as cost-effective or as reliable as tuning these systems reliably to fail-over and maintaining capacity among other servers to handle the load when failures occur. In other words the guiding principle is to use courser grained fail-over and to tune the global system to swap servers in and out quickly.

# Avoid Known Issues in WebSphere

## Java Coding practices and load control

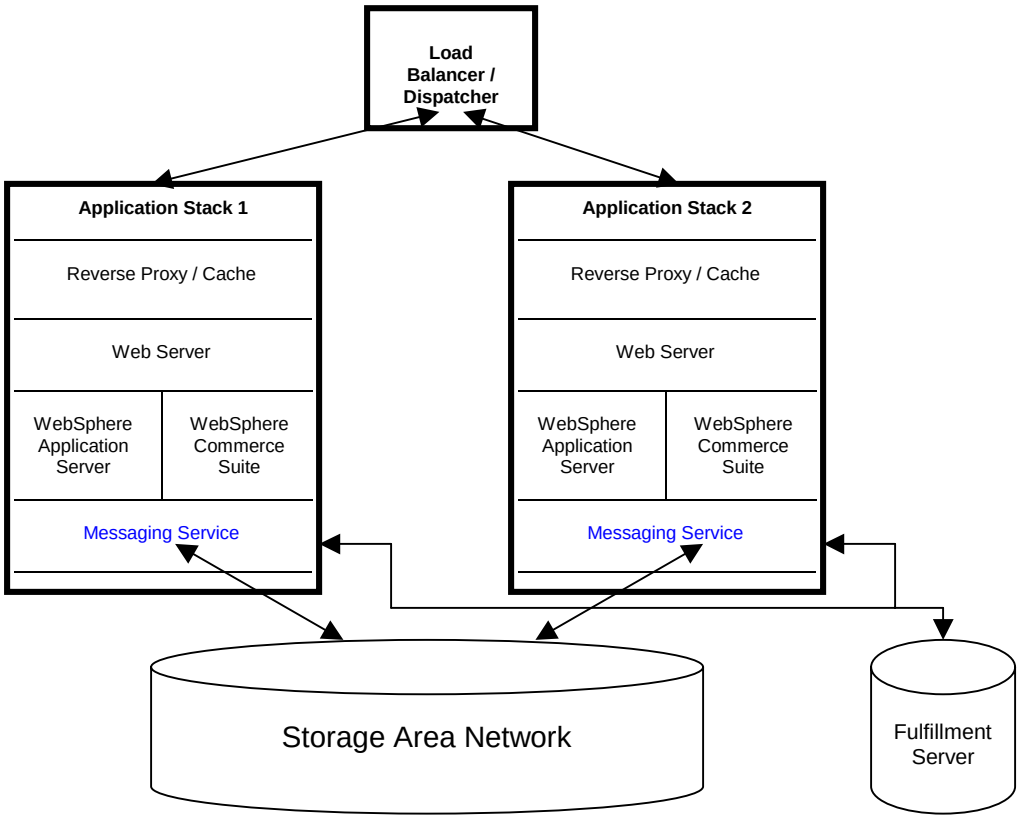
There are several undocumented conditions that cause WAS to crash. Some of these conditions are caused by load. Other issues are caused by coding practices. Testing and documenting these conditions, then coding around them is important for the long-term stability of applications as they evolve.

# More Sophisticated Monitoring

## Network and System monitors coupled with “deeper” web testing

WebSphere 3.5 has much better monitoring than previous versions. However it does not give a “global” system perspective of network and computing resources that may be strained. Standard OS utilities and experienced WAS administrators are indispensable. In addition, sophisticated, script- and cookie-capable monitoring software should be leveraged not only for regression testing but also for system monitoring. There are too many failure modes for simple URL-based network monitors.

# Case Study: Optimized Architecture




A Euphorion client wanted to maximize their investment in hardware augmentation for their highly successful online store. They needed the new system architecture to accommodate more traffic, more transactions, and more frequent updates. They wanted a system that was more flexible and capable of handling new shop flows, special sales, and end-user experiences. They wanted an increase in system reliability and they wanted a more scalable system to which capacity could easily be added if needed.

Euphorion developed an architecture for this client that increased flexibility, reliability, and provided enormous performance improvements on a modest hardware investment. The new site architecture provides a 10X improvement in transaction volume capability for a 1.5X increase in hardware cost. It allows for more frequent updates because of the smooth failover and reverse proxy functionality. It has increased flexibility enormously as was proven by the easy release of a Japanese language store using the new architecture.

In this architecture, application stacks are segregated vertically in different ways, depending on application dependencies. In this WebSphere implemenation web servers were tied to the

WebSphere Commerce Suite (WCS) components. Separating WCS onto separate machines was therefore not feasible. In addition, the messaging Services for this implementation were tied to WebSphere Commerce Suite.

The Reverse Proxy functions, web servers, and application Servers can and should all run on separate clusters of machines optimized for their respective single functions.

WebSphere has a number of problems with race conditions during recompilation of new JSPs and with the reloading of new servlets; these race conditions cause application server crashes. To avoid crashes, the load balancer/dispatcher tier must give fine control over which application stacks are accessed. Fine control using a reverse proxy allows for faster, cleaner, and more flexible releases and other site changes. Most pure-hardware solutions do not enable this important capability. The Reverse Proxy layer of this Euphorion configuration enables the client to switch quickly and seemlessly among other machines deeper in the application stack. This flexibility enabled unanticipated benefits in the management of application stacks resulting in a remarkably fast “cycle time” for services deeper in the stack and sophisticated release capabilities.