White Paper

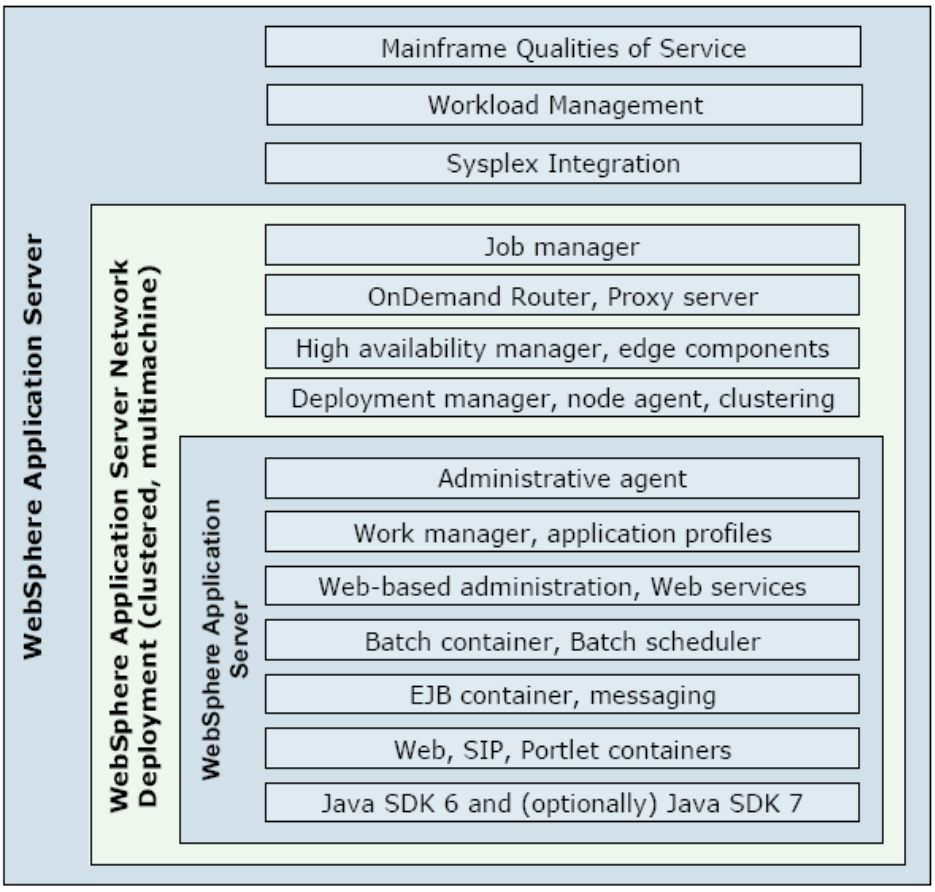
Tuning Websphere Application Server For Enhanced Performance

Table of Contents

Introduction and Background Information

WAS family supports a wide range of products and assists you in developing and serving all the relevant applications.

IBM Websphere Application Server is the leading software foundation especially for service oriented architecture (SOA) services and applications of your business. You can build business critical enterprise solutions and provide innovative functionality with Websphere Application Server (WAS). The WAS family supports a wide range of products and assists you in developing and serving all the relevant applications. Customers can utilize these products to create and manage dynamic websites and other complex applications effectively.



Websphere Application Server provides the environment to run your applications and to integrate them with any sort of system or platform. The major component in Websphere Application Server is the server runtime environment. An application server offers the necessary infrastructure for executing your enterprise applications.

It properly insulates the infrastructure from hardware, network and operating system. All the components of WAS package are depicted in the above figure.

This white paper gives in-depth information regarding the performance tuning approach to be adopted in implementing a highly available and scalable WAS environment.

Performance Considerations

A real performance run campaign is necessary before switching your new environment to the production. This is helpful in determining whether your infrastructure is well sized.

We need to keep in mind that 80% of the tuning is made on the application, middleware, and database layers. The remaining 20% tunes the hardware and operating system layer.

Performance is one of the major requirements of every WAS environment. Performance must be tracked on a regular basis while implementing your project. A real performance run campaign is necessary before switching your new environment to the production. This is helpful in determining whether your infrastructure is well sized.

When a system is rolled out, many users are prepared for small functional issues but performance issues are not always acceptable. Performance issues may affect all the stakeholders working on the project. So you nee to perform the necessary load tests that symbolize a realistic user load against your system.

How do you manage the activity of a real performance run campaign?

The performance demands should be measurable for proper evaluation. You need to establish success criteria to evaluate your scaling tasks.

Here are the Targets:

Throughput

The Throughput gives the total number of requests in a period that could be processed by the system. For instance, if an application has the ability to handle at least 25 simultaneous client requests and each request consumes one second, this application has a potential throughput of 25 requests per second.

Response time

The time from entering a system at a specific entry point until exiting the system at an exit point is known as response time. In a WAS environment,

this is measured as usually the time taken for the request submitted by a browser until the response is received.

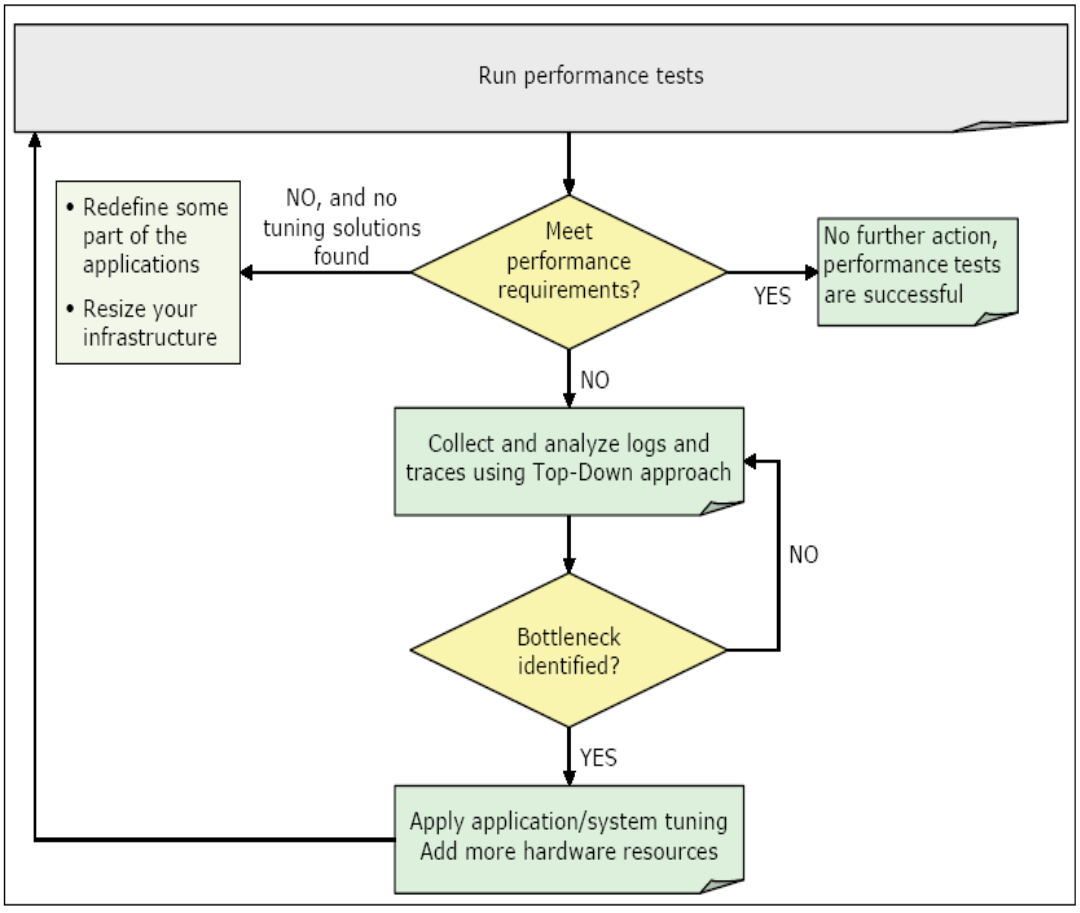
Batch style applications - Maximum time frame

Infrastructure tuning is an iterative process and involves optimizations in all the layers of environment. At first, you need to run your performance tests and later compare it with the requirements.

Batch applications often execute during a time frame, most probably at night to take the advantage of low peak hours and to avoid application customers during the day. Another important criteria for batch applications is to utilize all the resources.

Tuning Approach

Infrastructure tuning is an iterative process and involves optimizations in all the layers of environment. At first, you need to run your performance tests and later compare it with the requirements.



If the performance meets your objectives, ensure that you have planned effectively for your future growth and you are able to meet all the performance goals. Then you need to document all your findings in a performance tuning report and archive it. Just include the changed settings

to reach your objectives. To properly analyze the performance, collect the details such as logging and tracing.

The above figure summarizes the performance testing approach.

As per your analysis, you can locate the bottleneck and apply the changes. For instance, the solution would be to include more resources such as memory or processor. Later, you have to run the performance tests again and redo the same until all your requirements are met. This process will help you resolve all your major bottlenecks.

After finishing your performance campaign, you can make the required changes and update your production environment.

Workload Management

With workload management, you can use the resources effectively by distributing various loads more evenly. The management algorithms are useful in routing around the potential bottlenecks and overloaded components.

The process of sharing requests across many instances of a resource is known as Workload management. This is an important phenomenon for scalability and performance. Such techniques can help the system in serving more concurrent requests.

The two main features of workload management are Load balancing and Affinity.

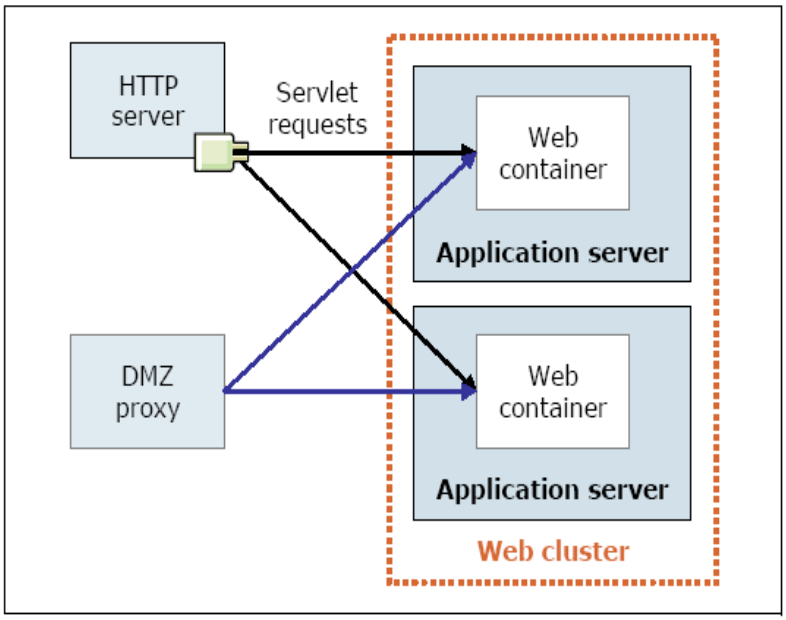
The capability of sending requests to alternate instances is called Load balancing. With workload management, you can use the resources effectively by distributing various loads more evenly. The management algorithms are useful in routing around the potential bottlenecks and overloaded components. In case of failed components, these management techniques offer high resiliency by routing all the requests to duplicate resource copies.

Affinity denotes the ability of routing concurrent requests to the same resource that served the first request.

Workload management in Websphere Application Server could be achieved by sharing requests across one or more application servers, each running a copy of the application. To maintain the session state, websphere server can route concurrent requests from a user to the same application server.

By clustering the application servers that are hosting web containers, you can immediately enable plug-in workload management for the application servers and the hosted servlets.

The following figure describes the routing of servlet requests between the clustered application servers and web server plug-in.



By clustering the application servers that are hosting web containers, you can immediately enable plug-in workload management for the application servers and the hosted servlets.

We can have a couple of load balancing options with Websphere Application Server:

Round-robin

This option is based on the weight associated with cluster members. If all the members have similar weights, the plug-in distributes equal requests to all members of the cluster (without heavy affinity configurations). If all the weights are scaled from 0 to 15, the plug-in sends requests more often to the members with higher values of weights. Requests are not sent to the members with a weight of 0 unless no other servers are available.

You can use the following formula to determine the routing preference, where n cluster members are in a cluster:

Percentage routed to the first server,

Server1 = weight1 / (weight1+weight2+...+weightn)

Random

A cluster member could be picked randomly by the plug-in with this Random option. Session affinity can affect the load balancing options. As the session gets created at the first request, all other requests should be served by the same member. By analyzing the session identifier, the plug-in retrieves the application server that served the previous request.

Session Replication

Time based write can write latest session information to the back-end on a time interval. End of Servlet service writes the updated information after the execution of servlet.

Copying the session data of users to a remote store is known as session replication. In the event of a failure, this data could be retrieved. If there is an issue, another process can handle all other requests by retrieving the appropriate session content. This is very essential in case of highly available systems as you need to provide all users with an uninterrupted browser experience.

Tuning Considerations

When session replication is enabled, here are four factors to be considered in tuning the environment:

- Write frequency
- Write contents
- Session timeout
- Background session invalidation interval

Write frequency

Write frequency is an important factor in session replication reliability and performance. Write frequency can dictate how often the sessions are replicated to the domain. As the interval between the replicates (i.e. write frequency) is minimized, the performance is decreased. This is because the CPU cycles are being used more frequently to replicate the data instead of handling subsequent requests. The three options of write frequency setting are Time based write, End of servlet service and Manual.

Time based write can write latest session information to the back-end on a time interval. For instance, if a user makes four requests between a given write interval then it is considered as just one update. This setting could be utilized for maximum performance and the default value is 10 seconds.

End of Servlet service writes the updated information after the execution of servlet. You can use this setting for failover support and maximum reliability.

Manual gives developers the most flexibility as to when the session is backed up, but the owner of the application should consider this mode with caution.

Write contents

The Write contents factor denotes whether "Only updated attributes" or "All session attributes" should be persisted. It is advised to use "Only updated attributes" option so that you can minimize overhead of persisting data that has not changed.

Note: You must use the "write all" setting if you are unable to call setAttribute on an updated session attribute.

Session timeout

The session timeout specifies the validity of a session. The default value of the timeout is 30 minutes. So after a session is accessed, it is valid for 30 minutes. You can increase this value if sessions are going to be idle for more than 30 minutes. By decreasing this value, you can invalidate the sessions faster. Unused sessions could be invalidated quickly with small values of session timeout.

Background session invalidation interval

This is the interval at which a background process looks for timed out sessions that should be invalidated. This value is actually based on session timeout value and an internal algorithm. The default value is 150 seconds.

Controlling Memory Buffers

By setting the buffer sizes, you can enhance the communication of a messaging engine with its data store.

Every messaging engine manages few memory buffers. By setting the buffer sizes, you can enhance the communication of a messaging engine with its data store.

You can set the following properties:

sib.msgstore.discardableDataBufferSize

Here the default value is 320000 (i.e. around 320 kilobytes). The discardable data buffer contains the data related to active transactions and any other non-persistent data that the messaging engine has neither consumed nor discarded.

The messaging engine holds this data completely within this memory buffer and never writes it to the data store. When the messaging engine includes data to the discardable data buffer, it may discard the existing data in the buffer to create certain space.

The messaging engine can only discard data that is not involved in active transactions. This behavior helps the engine to get rid of best effort non-persistent messages.

By increasing the discardable data buffer size, more best effort non-persistent data has to be handled before the engine begins discarding.

sib.msgstore.cachedDataBufferSize

The cached data buffer can optimize the performance of the messaging engine. The default value is 320 kilobytes. The messaging engine adds data to the cached data buffer as it is written and read from the data store. As mentioned above, the engine may discard the existing data in the buffer to create certain space.

sib.msgstore.transactionSendLimit

This denotes the maximum number of operations that an engine can include in each transaction. For instance, each JMS send or receive is an operation that is being counted towards the transaction send limit. Its default value is 100.

Security Configuration

While changing the new security policy, you can consider propagating new security settings to all nodes before restarting the deployment manager and node agents.

Security configuration could be tuned to balance the performance with functionality.

You can achieve this balance by considering the following:

- Consider what sort of security is required and the things to be disabled in your environment. For instance, if your servers are running in a Virtual Private Network (VPN), you may consider disabling Secure Sockets Layer (SSL).
- You may consider disabling Java 2 security manager if you know exactly what code is put onto your server and you need not protect process resources.
- While changing the new security policy, you can consider propagating new security settings to all nodes before restarting the deployment manager and node agents.
- You normally get access denied errors if your security configurations are not consistent across all servers. So you should propagate new security settings when enabling or disabling administrative security.
- If you feel your environment is secure enough, you can increase the cache and token timeout. By increasing the values, you need not re-authenticate frequently. This can help all other requests to reuse the credentials. To choose the initial size of the hashtable caches, you can use security cache properties. This affects the frequency of rehashing and the distribution of the hash algorithms.

- As RMI (Remote Method Invocation) uses stateful connections and SOAP (Simple Object Access Protocol) is absolutely stateless, it is normally recommended to change your administrative connector from SOAP to RMI.

- By distributing the workload to many JVMs (Java virtual machines) instead of a single JVM, you can increase the security performance as there is less contention for certain authorization decisions.

High Availability

High availability could be achieved by adding redundancy in the infrastructure to handle most of the failures. Availability has a direct impact on the performance and scalability.

The ability of a system to tolerate certain number of failures and still remain operational is known as High availability or Resiliency.

High availability normally denotes that your platform continues to respond to client requests irrespective of the circumstances. Depending on issues, your infrastructure can run in a degraded mode. High availability could be achieved by adding redundancy in the infrastructure to handle most of the failures. Availability has a direct impact on the performance and scalability.

According to your requirements, you must define the high availability levels. The most common method to define availability is by the Nines or by the percentage of availability. For instance, the system availability of 99.9% represents 8.76 hours of outage in a single year.

We can calculate the availability by using the formula:

Availability = (MTBF/(MTBF + MTTR)) X 100

Here,

MTBF is the mean time between failures.
MTTR is the maximum time to recovery.

A Websphere Application Server platform consists of various components such as HTTP servers, load balancers, and database servers. Availability is determined by the weakest component and the cost of the infrastructure is directly linked to the level of availability.

You are supposed to calculate the business loss of the downtime and make sure that the business case justifies the total cost. This is because moving system availability from 99.9% to 99.99% can be more expensive.

The system is generally used on regular working days and only during regular business hours. So this implies that 99.9% availability is just enough to meet the operational window.

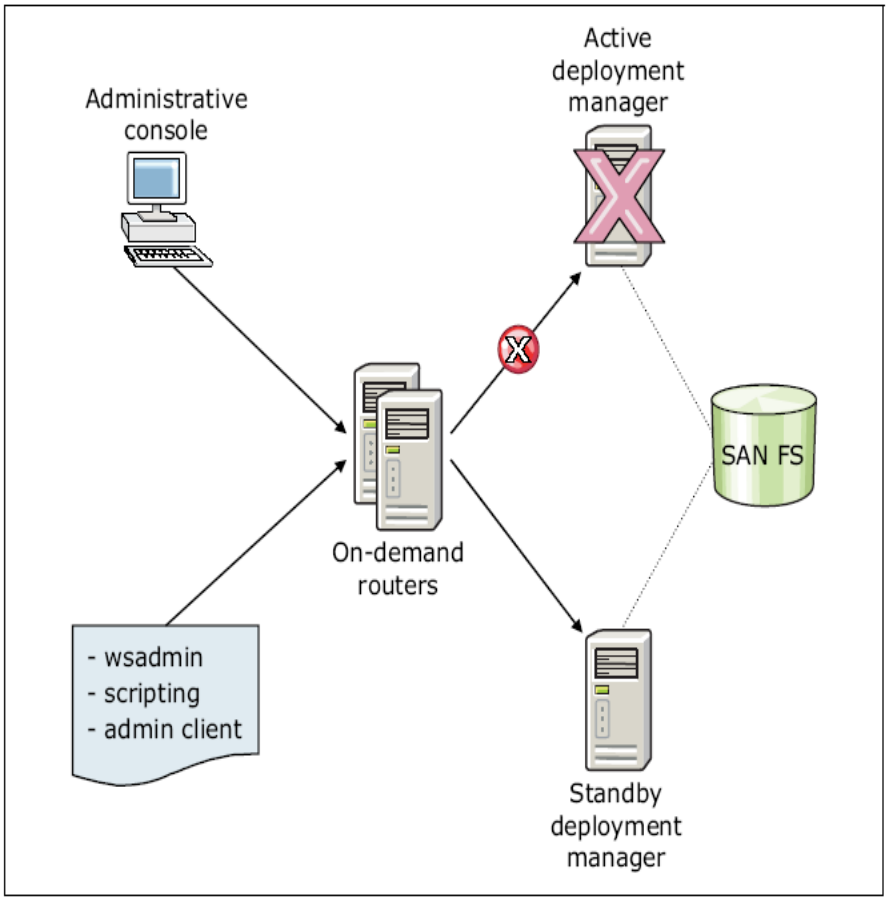
Note: The features of availability could have an impact on the cost of the solution. You need to calculate this increment in the implementation cost.

Highly Available Deployment Manager

A highly available component executes in each deployment manager and participates in the selection of an active deployment manager.

High availability could be achieved by utilizing redundant deployment managers with a standby model and a shared file system. In this process, one of the deployment managers is considered as primary. Other deployment managers could be considered as backups in the standby mode. These deployment managers are always available to take over the active role in case of failure of the primary.

A highly available component executes in each deployment manager and participates in the selection of an active deployment manager. The deployment managers in standby mode are unable to perform certain administrative tasks. So the standby normally rejects login and JMX requests.



New elements are at the heart of high availability for the deployment manager. One such element is the on-demand router. The on-demand router can automatically recognize the active deployment manager.

Multiple on-demand routers could be configured in the environment to promote high availability. They are usually started first in order to recognize the primary deployment manager in the environment.

It also has the knowledge of endpoint configuration for routing administrative communications. Multiple on-demand routers could be configured in the environment to promote high availability. They are usually started first in order to recognize the primary deployment manager in the environment.

The above figure illustrates a common topology for highly available deployment manager.

After configuring all the components of highly available deployment manager, they need to be restarted to take immediate effect. You can also include on-demand routers and any other active deployment managers in this restart.

Conclusion

As no two applications will utilize the resources of an application server in a similar way, a single set of tuning parameters is just not enough to achieve high performance levels for any two applications. Normally you can achieve certain improvements in the performance by tuning the JVM and thread pools.

The default setting would perform well in few cases. To get the best performance, some additional tuning is always necessary. This white paper gives you that starting point.