Comment lines: Tom Alcott: Everything you always wanted to know about WebSphere Application Server but were afraid to ask -- Part 3

Tom Alcott

February 07, 2014
(First published June 21, 2006)

Answers to more very frequently asked questions about IBM® WebSphere® Application Server, including how to run it over multiple data centers, which JDK to use, and why (and when) you should migrate to Version V6.1.

From the IBM WebSphere Developer Technical Journal.

Third time's a charm

Twice before I have used this forum in an attempt to provide answers to common queries about WebSphere Application Server and each time I have received generally positive feedback. Perhaps this time, by the time you reach the end of this column, I will have finally answered all of your questions. Then again, probably not. Nevertheless, as before, this will not be a discussion of the things you're afraid to ask about WebSphere Application Server, but rather questions I get asked over and over. Also as before, I may need to rely on my most oft used answer, *it depends*, when it truly is the most appropriate. (Hey, if you get to ask me the same questions, it's only fair that I get to use my same answer, right?) Although a definitive answer, while preferred, is not always possible, I will attempt in every case to at least provide some guidance on what you need to consider to determine the best answer for your specific situation.

Q: What's new in WebSphere Application Server V6.1?

A: It's no surprise that I have been asked this question a great deal of late. The short answer is "lots," but I'm sure you'd like a bit more detail, so I will try to put the highlights in a nutshell.

WebSphere Application Server V6.1 extends the capability delivered in WebSphere Application Server V6.0 by adding a number of features aimed at improving the usability (or consumability as some refer to it) in several key areas:

- **System administration:**
  - There are a number of administration improvements, in both the administration console, which is now implemented using portlets, as well as the wsadmin scripting language.

  - The console has been redesigned to make navigation easier and a number of guided tasks and fast path dialogs have been added to make configuration and administration of common operations much faster.
  - Command assistance is also available in the console, logs, or via JMX notifications that detail the Jython wsadmin commands for the administrative actions just performed on the console (not all commands are available with this assistance, but this will improve over time). This makes creation of administration scripts much easier.
- **Security:**
  - Security is now enabled by default, using a file-based registry. Additionally, you can map multiple standalone registries into a single federated registry, known as the Virtual Member Manager. These registries can be either a replacement for or in addition to the file-based registry.
  - Key management is now integrated into WebSphere Application Server administration (both the console and wsadmin), and, related to this, the DummyKeyRing is no longer shipped with the product; instead, unique keys are generated during profile creation (this occurs even if security is not enabled).
  - Instance-based administration is now available for the cell, nodes, servers, and applications, enabling command line administration of any of those scopes to be limited to a specific role or group (instance-based administration is not implemented in the console).
  - For those desiring a Windows® Single Sign-On (SSO) solution, there's now a SPENGO-based TAI which enables Windows-based Web clients to use the Windows login without having to login again when accessing applications running in WebSphere Application Server.
- **Portlet container:**
  - Along with the use of Portlets for the admin console implementation, a JSR-168 compliant portlet container is now part of WebSphere Application Server V6.1. (A portlet container is only a small portion of what a portal server provides, so those of you with portal server implementations should not be thinking of migrating them to WebSphere Application Server V6.1).
  - Not only are portlets now a supported programming option, so too are Session Initiation Protocol (SIP) servlets -- in addition to the familiar HTTP servlet. All three of these APIs are now implemented in a converged container that enables sharing of application state between these APIs in a given application.
  - Another key addition to the programming model is support for Java™ 2 Standard Edition 5 (J2SE 5 or JDK 5), which not only provides additional Java language APIs (such as Generics, Annotations, Enumerations, and AutoBoxing), the implementation of the WebSphere Application Server runtime on J2SE 5 results in a significant performance improvement (final testing is underway as I write this, so I cannot provide specific figures).
  - There are also a number of Web services additions and improvements as well.

More information on WebSphere Application Server V6.1 is available in the Resources section, but I would also invite you to contact your local IBM representative for more specifics, or even consider

attending a two-day (no cost) hands-on WebSphere Application Server V6.1 Proof of Technology event at a local IBM office (your local IBM rep can provide you POT locations and dates).

Anyway, one question leads to another and one that typically comes up when I'm asked about the features and functions in one version of WebSphere Application Server or another is:

Q: Should I migrate to version "x" of WebSphere Application Server?

A: Usually, this question is being asked about a specific version (such as V5.1, V6.02, V6.1, and so on), and while my answer to a question with respect to a specific version may vary, the principals underlying my response have been consistent for several years now.

In deciding when to proceed with a production deployment for a specific software release, the most important factor should be the maturity and stability of that release. As a result, it has been my experience (as well as my practice when I worked for a living before joining IBM) that most customers do not deploy any software on the "dot zero" release, be it a new release of WebSphere Application Server or Windows or what have you. Therefore, you need to consider where the software version is in terms of maintenance releases. Let's use WebSphere Application Server V6.1, which was just released earlier this month; it is a "dot zero" (6.1.0) software release and will be for a few months. If you're looking at deployments in 6-7 months from now, then I would say it is safe to proceed, with, of course, appropriate testing prior to a production deployment. During this time, the maintenance stream will have provided updates and fixes to any problems that may have been missed in pre-release testing.

If your timeframe is sooner then that, then the use, criticality, and size of the deployment would be the next most important factors to consider. Another consideration that must be factored into deciding when to migrate is the timing of production replacement or rollover cycles for hardware, operating system, and third party applications. Some customers choose to upgrade or renew their entire infrastructure at once, while others prefer to stagger upgrades. Either way, your own policy and plans in this area should also carry some weight.

In this forum, I hesitant to make a blanket, universal recommendation, since the specifics of the individual deployment will determine whether the better idea is to migrate now or later -- or ever.

Similarly, I am often asked about new deployments or additional deployments on a mature release. Let's use WebSphere Application Server again as an example, this time Version 5.1. In looking at the maturity of WebSphere Application Server V6.02 (and WebSphere Application Server V6.1), as well as the End of Service (EOS) date for WebSphere Application Server V5.1, I would be very hesitant to recommend any deployments on WebSphere Application Server V5.1 at this time. It's not that WebSphere Application Server V5.1 isn't stable and production proven -- it is -- but EOS is less than 18 months from now (see References for support lifecycle information), so any deployment on Version 5.1 will have to be migrated in a relatively short timeframe. On the other hand, moving to the most current proven, stable release, which is Version V6.02 at this writing (recall the preceding paragraph), maximizes the service life for the WebSphere Application Server version in production, and minimizes the migration effort.

Last, while on the subject of migration, I will add that many customers cringe at the mention of migration, mainly based on their experiences in moving from WebSphere Application Server V3.0 or V3.5 to WebSphere Application Server V4.x or V5.x. This is the result of the application remediation that was required to upgrade the application code from "pre-J2EE" to J2EE. More recent experience -- which I confess is somewhat anecdotal, based on the customers that I've worked with -- indicates that most customers can redeploy J2EE 1.2 or J2EE 1.3 applications on WebSphere Application Server V6.x WITHOUT application changes. I should mention, though, that the J2EE 1.4 specification states that you must be able to run 1.2 and 1.3 applications without change. This might somewhat deceiving, since in truth this means they must run "as is" -- it does NOT mean the must run "the same," **hence testing all applications in the new environment is critical**.

Returning to the few cases where applications did require changes, such changes were the result of deviations from J2EE or the use of APIs that were deprecated. To put this into perspective, I'm talking about hundreds of customer applications and of those, only 1-2% of the total required changes. That's pretty impressive for any technology and compares favorably to my (ancient) experience with z/OS® (S370 as it was know then) and COBOL, which has offered similar results when upgrading.

Q: Why does WebSphere Application Server require that I use an IBM JDK?

A: First, to set the record straight, the requirement for support is use of the "WebSphere-supplied JDK," and on the Sun™ Solaris™ and HP-UX platforms, the WebSphere-supplied JDK is one from Sun and HP, respectively, although the IBM ORB and IBM security implementation replace those that come from Sun and HP. The IBM ORB is needed for EJB WLM, and the IBM security implementation is FIPS compliant, while the Sun and HP JDKs are not. For all other platforms the WebSphere-supplied JDK is an IBM JDK.

Now, as far as "why the WebSphere supplied JDK?", this goes back to our support experience with WebSphere Application Server V2.x and WebSphere Application Server V3.x. For those versions, we listed the JDK that was supported, where the customer could go to download it, and provided instructions for installing the JDK and configuring WebSphere Application Server to use it. This approach generally resulted in numerous problems and support calls resulting from:

- Incorrect install of the JDK.
- Incorrect install of WebSphere Application Server (or configuring it to use the JDK).
- Inability of the customer to obtain the required version of the JDK and installing a different non-equivalent one in its place.
- Regressions or defects in "newer" JDKs.

By supporting and supplying a specific (and tested) JDK with WebSphere Application Server, we avoid all of these issues (and many more), improving the experiences of WebSphere Application Server installation, usability, and reliability, both from our perspective and from our customers'. Additionally, the supplied IBM JDKs are not only more secure than the Sun (and HP) counterparts, but they are also faster! See the SPECJBB benchmark site in Resources for more information.

Of course, there has to be one exception to the use of a WebSphere-supplied JDK, and that is for specific versions of the Sun JDK on Windows for use with the pluggable client. (See Resources.)

Q: Can I run a WebSphere Application Server cell over multiple data centers?

A: Did you just say, "Wait a minute, didn't he write about this before?" Well, the answer is "yes, I did," but this continues to come up, so I'm going to take this opportunity to expand on what I said in Part 2. You don't need to try and flip back and forth between that article and this one; I have expanded my previous answer with some additional information, below, that reinforces the recommendation not to do so (and no, I'm not getting paid by the word for this!).

Let's look at the most obvious issue first: network speed and reliability between the data centers. In many cases, the performance and reliability of a WAN is not as good as a LAN, though there are environments where it is asserted that the WAN is highly reliable and also provides LAN bandwidth; in such a case, the WAN appears to be the same as a LAN to applications (such as WebSphere Application Server). So the simple answer would be: Sure, go ahead, if you have a fast and reliable WAN. In practice, however, the assertion or presumption of a WAN that is as fast or reliable is seldom (if ever) realized. In fact, I can cite numerous real world examples where the theoretical WAN performance and reliability fell far short of the actual WAN resiliency, resulting in some of the issues discussed below. I'd add that if, for some reason, you feel that your WAN is somehow not subject to interruptions or outages, then your failover testing start with severing the network link between the data centers. In the past, doing so has provided a great deal of clarity and realism to users that is lacking when reading about the problems and issues that will ensure.

All that stated, overlooked in all of this is the far more important question: Why have multiple data centers? Normally, you do so to increase reliability, thus if one entire data center fails or is lost (in other words, a true "disaster"), you want the remaining data center to be able to handle work without major problems. Given that, one needs to plan for data center outages that are not brief, and reliability in this state will become very important as a result. Additionally, failover in this condition would be likely very difficult to test correctly, since this is outside the realm of a "normal" WebSphere Application Server failover, which is at the component level (server, Web, EJB, and so on), not the data center level.

- What happens to WLM endpoints in such a case, specifically when the clients are in one data center and the servers are in another? This can arise with either the EJB WLM or HTTP server plug-in WLM, depending on the deployment and network architecture, and while both WLM implementations will recover (via timeouts), this is one more situation to consider and likely avoid.
- The WebSphere Application Server Network Deployment deployment manager is a single point of failure. As a result, if you lose the data center where the deployment manager is running, you lose the ability to manage the cell until a backup deployment manager is brought up in the surviving data center. While it is possible to deal with this, it's still one more thing to deal with during a failure.
- If you have chosen to distribute your HTTP session objects, and you're using a database for this purpose, what happens to your session information if the database is on the now non-functioning data center?

In most cases, hardware and software are very reliable, and so human error is the most often the cause of an outage. If you are only running a single cell and a mistake or problem results in an outage, then you've lost ALL of your capacity. Two cells (or more) aligned along data center boundaries provide redundancy that a single cell does not. Somewhat surprisingly, some customers that I've discussed this with have taken care to align their other middleware, databases, messaging, and so on, along data center boundaries, but somehow think that the same fundamental operational principles do not apply to WebSphere Application Server Network Deployment or WebSphere Virtual Enterprise. The principles are unchanged, primarily the fact that a single implementation of anything is a single point of failure, and if you require high availability single points of failure need to be eliminated. In fact, not only do these fundamentals apply to WebSphere Application Server Network Deployment and WebSphere Virtual Enterprise, they apply as well to any IBM middleware product that is based on WebSphere Application Server Network Deployment. If someone has constructed two data centers, high availability is almost always the primary driver; building two data centers really has no other purpose. If all you want is scalability, just add more computers, network bandwidth, and so on, to a single data center.

Related to this, I have also been asked about running WebSphere Application Server Network Deployment (or WebSphere Virtual Enterprise) clusters across data centers. While the notion of a single cell across data centers is bad from a risk aversion perspective, running a cluster across two data centers not only requires you to forget about minimizing risk, as noted above (since a cluster cannot span cells), but further increases risk along multiple dimensions.

First, let's look at what happens with the Web server plug-in. The loss of one data center means that one-half of the application server endpoints are no longer reachable. As a result, requests with affinity to the servers in the failed data center (as well as new requests that the workload manager is distributing to the failed servers) will wait until the ConnectTimeout elapses before marking the server as unavailable (hopefully you've specified a value for this, normally in the range of 5-10 seconds, rather than relying on the default, which is the TCP/IP connection timeout for the OS!). The plug-in will then redirect the request to another server, which is fine if the workload manager directs the request to a server in the operational data center -- but what if the request is directed to yet another server in the non-operational data center? Once again, the request will wait the ConnectTimeout before the plug-in redirects the request to another server. Eventually, at some point, each plug-in process will identify all the unavailable servers, but until all that takes place, response time is going to be very slow for the requests that are directed or redirected to servers in the non-operational data center. Further, the value you've specified for the RetryInterval (the default is 60 seconds) will govern how much time will elapse before the plug-in attempts to send another request to a server that was marked as unavailable. As a result, the performance and response time degradation experienced when the data center was initially lost will repeat every RetryInterval.

Second, since you're going to align your clusters on core groups (the "HA domain" in WebSphere Application Server V6.x), if you suffer a network failure between the two data centers, you run the risk of application or data inconsistency. Here's why: core groups don't require quorum, so both halves can continue to run. WebSphere Application Server assumes that consistency/locking is managed through other means, like database locks, file systems, and so on. Thus, the system

will either become inconsistent or simply stop until the failover (to only one data center) of some shared resource. The details, of course, depend on your environment, but let's discuss one example.

Suppose multiple servers in the same cluster are configured to share transactional logs for failover purposes. As a practical matter, those logs are stored in one data center (possibly with automatic replication to the second). If the data center that contains the logs goes down, then the other data center is unable to proceed because the logs aren't accessible. Once they are failed over, then one data center can continue. Now, consider a more insidious failure. Suppose the network link between the two data centers fails. Now what happens? The data center with the logs will probably carry on as usual. The other data center is still running, but it can't get to the logs -- or, we hope it can't. If you have some kind of automatic failover, imagine what would happen if the second data center started using the copy of the logs. The result is a "split brain" scenario, where the two data centers will move to inconsistent transactional states. Ouch! Imagine this with something more complicated like a database. Can you make this work? Possibly. Is it worth the risk? Not from my perspective.

Often a rationalization for a cell spanning data centers is a requirement for a brief failover time — say, a small number of seconds or minutes. However, all the factors cited above as well as the recreation of the WebSphere Application Server Network Deployment core group in practice make achievement of this requirement impossible. This is because failure detection and reconstruction of the WebSphere Application Server Network Deployment run time state will take far longer than the time allotted for such stringent requirements. It's actually far faster and more effective to have independent cells where the load is switched at the network layer (in front of the data centers) to meet these types of service levels. See this article for more information.

Another associated issue is the fact that lab testing has determined that core groups should in many cases be comprised of no more than 100 processes to optimize failover and minimize recovery time, so even if you choose to run a single cluster (in a single cell) across two data centers, the sizing of the core group will limit the size of your clusters, and will likely require some of the additional administration effort you hoped to avoid by having a single cell.

An often-asked question is one of support for a cell that spans data centers. IBM Product Support will accept PMRs for deployments when a cell spans data centers, but if Product Support determines that any issues are outside the design criteria for WebSphere Application Server Network Deployment (and in turn IBM middleware that leverages WebSphere Application Server Network Deployment), this will be classified as "works as designed" and no product fixes or changes will be delivered, leaving the customer solely responsible for problem resolution.

I will add that, with respect to opening PMRs when a cell spans data centers, this is because WebSphere Application Server Network Deployment doesn't respond across a WAN (between data centers) as it does when using a LAN (in one data center). It has been my experience that such PMRs inevitably result in the collection of lots of trace and some tuning changes, but in the end don't resolve the issues that prompted the PMRs. It is only when the deployment is changed to a cell (or cells) aligned along data center boundaries that this issues are resolved.

While it's certainly possible to construct (and test) a cross center clustering solution -- if you exercise extreme care -- there is always the risk that you will have missed something that will occur during a real disaster. It is because of these issues that I have not mentioned (and those I haven't thought of) that leads me to advise against running a WebSphere Application Server cell across multiple data centers. As I often mention, a disaster is not a time you want to be learning on the job.

Hopefully, I have convinced you that a single cell spanning two data centers is not a good idea. In closing on this particular topic I would like to thank Jason McGee, Billy Newport, and Keys Botzum for their thoughts.

Conclusion

Once again, that's the end of my ramblings. For now, at least. Hopefully I have provided some food for thought for the questions that I didn't provide definitive answers for, as well as some reasonably definitive answers for the others.

Acknowledgements

I would also like to thank Bill Hines for reviewing this article and providing comments.

More articles in this series

- Part 1
- Part 2