Why WebSphere Extended Deployment for z/OS

WebSphere Extented Deployment (WXD) brings significant value to the WebSphere Application Server environment. It vastly expands the flexibility to distribute work across server instances potentially deployed on a variety of system platforms, allows for effective monitoring and control of production environments, automates the non-disruptive installation of application versions and extends the Java2 Enterprise Edition (J2EE) programming model to support Java batch with the management controls typically found in robust batch environments.

Much of the early enthusiasm for WXD came from the distributed systems community driven by its support for dynamic clustering. This allows businesses to dramatically increase the utilization of their servers by sharing physical systems that would otherwise be dedicated to specific applications. Within the z/OS environment, however, this feature is generally of lesser value. The reason is that System z9 is already a shared system and z/OS handles physical resource allocation automatically under the covers. The combination of the integrated Workload Manager (zWLM) and the Intelligent Resource Director (IRD) bring physical resources to the virtualized application servers based on business priority goals. Dynamic clustering can be beneficial in a z/OS setting where there are several physical machines configured within a Sysplex and you would like to dynamically introduce a Sysplex member when load reaches a certain point.

WXD is a multi-dimensional product supporting many key features in addition to dynamic clustering. The significant value of WXD for z/OS is derived from several important features of the offering. They have a positive impact on z/OS operations, availability and flexibility in addition to opening the door for innovative new application use cases.

A good way to determine if WXD is right for you is to answer a few simple questions:

Do I want to gain the cost benefits of offloading existing batch processing to Java and leverage zAAP processors?

Do I need to support long running “batch” style applications with checkpoint facilities and the ability to monitor and control the long running job from a web console or an existing z/OS-based batch scheduler such as the Tovoli Workload Scheduler?

Do I need more granular classification options in order to route work to specific server instances. WXD provides extremely flexible classification options. For example, incoming requests may be classified and routed based on time of day, source IP address, HTTP header content or user/group identity just to name a few key options?

Do I need an automated approach for installing new versions of applications across a cluster in a non-disruptive manner or do I need the ability to pilot a new version with a subset of my users and gradually expose use of the new version as confidence in its operational characteristics increases?
Do I want to perform goal-based workload routing across cells (multiple data centers or a heterogeneous collection of servers to support the distribution of work across servers deployed on zSeries and pSeries physical systems for example.)?

Do I need to monitor a collection of servers for performance and availability from a web console and have the servers emit alerts based on performance thresholds while capturing runtime statistics for trend analysis and problem determination?

Would server management become easier if I didn’t have to manage the creation and deployment of plugins for front end web servers?

Would I like to move the processing performed by the HTTP server on z/OS, which is executes on general purpose processors, to the zAAP in order to reduce cost?

Would I like to eliminate the front end web server tier and make the environment simpler?

Do I need to prevent a server from being flooded with too many requests?

Can my application benefit from partitioning which routes requests to specific servers in order to maximize the effectiveness of the data cache and minimize physical I/O?

Can my application benefit from an object grid which provides a distributed object cache accessible from a collection of servers in order to minimize physical I/O and improve throughput?

If the answer to any of these questions is yes, WXD is a good solution for you.

The remainder of this paper will explore WXD features in greater detail within a z/OS server context.

**Support for Java Batch**

Sometimes, a web application needs to trigger the execution long running background tasks and it’s preferable to run these asynchronously. For example, a web application may collect information about a request. Perhaps this request needs to be performed at a future date or will execute for many minutes or even hours updating a production data store. Once the interactive dialog is complete, the application is ready to submit the work to be performed. Architecting this solution can be challenging especially where features such as checkpoint/restart, task monitoring, and task control are required. WebSphere/XD provides an excellent solution for this scenario. It provides a task scheduler, referred to as the Grid Scheduler, along with a long running task manager which is essentially a batch container for the J2EE environment. Tasks may be submitted to the scheduler for execution. The scheduler will dispatch the work to a batch container. Batch containers may be executing on a number of LPARs within the Sysplex to load balance across many
systems. Since long running tasks often prefer isolation from other tasks, the ability of the WebSphere for z/OS application server to spawn servant address spaces offers an excellent vehicle for delivering this isolation. The job management console is itself a web application providing an interface for viewing jobs and their status, submitting jobs, specifying job classification rules and controlling the execution of jobs. Jobs may be started, stopped, suspended, resumed, restarted and purged.

In addition to supporting the need for long running tasks initiated by a web application, this facility may be used for traditional batch job execution. It supports the concept of multi-step jobs and allows for a checkpoint interval so that a job may be restarted at a specific recovery point in the event of a disruption. The tasks being executed may be plain old java objects (POJO’s), Java main, compiled programs such as COBOL and C++, UNIX processes or commands. It opens the door to a wide variety of useful scenarios. Sometimes tasks need to be executed that are not running under the control of the WebSphere Application Server. An XDAgent is provided for these environments allowing a task running on a UNIX system or a Windows workstation to be invoked.

Jobs are submitted by passing an XML document to the Grid Scheduler. This may be accomplished via remote method invocation, a web service request, a command or through the job management console web application. In addition, jobs may be submitted from schedulers such as the Tivoli Workload Scheduler. This allows z/OS environments to integrate the J2EE batch support offered by WXD with their traditional batch job scheduler. Jobs may also be submitted from JES allowing new J2EE applications to be executed in one step and traditional COBOL applications to be executed in other steps of the same job. Businesses that want to gradually replace their existing batch jobs with new J2EE Java programs, perhaps to exploit the price advantage of the zAAP processor and consolidate development skills, may now accomplish this.

Management of Workload Distribution - On-Demand Router

Today, there are two options for distributing work across a collection of z/OS logical partitions (LPARs). One is to use the Sysplex Distributor that is an integral feature of z/OS. This provides a single virtual IP address for requestors and sprays the work across a number of specific IP destinations. The distribution of work is influenced by zWLM to ensure that work is allocated to optimize zSeries resources and meet application performance goals. Sysplex Distributor is a highly efficient, fault tolerant, solution for workload distribution across a Parallel Sysplex. However, it does not understand session affinity and hence is not a total solution for many web applications that are of a conversational style. The other approach is to use the WebSphere plugin. The WebSphere Administration application server will generate a plugin with routing rules and distribute it to a front end Web Server. The Plugin understands session affinity. It may direct new requests to specific application server instances on z/OS or direct them to the Sysplex Distributor to intelligently load balance new requests. Requests that arrive with a valid session token will be directly to the appropriate server instance on z/OS. The Web Server may be placed on z/OS or elsewhere in the network. The plugin approach, however,
begins to break down in situations where work needs to be distributed across multiple cells, perhaps multiple Parallel Sysplexes in a z/OS setting. Or, in situations where work needs to be distributed across a heterogeneous collection of systems consisting of z/OS and other Linux or UNIX platforms. Furthermore, the plugin does not understand complex routing rules. It distributes work based on a mapping of the URL to a collection of servers representing themselves as virtual servers. Sometimes, there is a need to distribute work based on the user’s security identity or relationship with a security group, the time of day, source IP address or the content, including variable values, contained in the HTTP header for the request. This flexibility is provided by the On-Demand Router (ODR) feature of WXD. It is appropriate for all platforms. In addition to flexible routing, the ODR is able to associate requests with service classes that represent performance goals. The genesis of this feature is zWLM. So, now it is possible to construct complex routing rules and flexibly associate requests with different performance goals for a heterogeneous set of server instances. One nice feature of ODR for z/OS is that the transaction class is forwarded to the application server instance on z/OS and is then used by zWLM to manage dispatching priority. ODR does not use a plugin model, it dynamically detects servers. Hence, the task of generating the plugin and distributing it the web server is avoided. Finally, the ODR does not run in a web server; it is a Java application and may be deployed on any platform, including z/OS. Since it is a Java application, it is eligible for execution on the lower cost zAAP processor. On z/OS, the ODR is a started task and runs in its own address space. The ODR can be configured for high availability where it is employed on multiple LPARs with Sysplex Distributor providing a single virtual IP address for client access to the Sysplex as a single consolidated target.

**Application placement across a heterogeneous collection of platforms**

In addition, the ODR may be used to route work across many systems. With the plugin or Sysplex Distributor approach you are limited to routing work to a cluster than exists within the WebSphere server domain or cell. With ODR, you may route work to multiple cells. For example, a user belonging to one security group may be routed to Sysplex A while another user belonging to a different security group would be routed to Sysplex B. ODR routing supports a heterogeneous collection of servers as well. This allows work to be spread across both z/OS and Linux servers, for instance. The routing decision can be based on performance goals; similar to the way zWLM works, allowing the ODR to determine the optimal place to direct work. For example, work that benefits from co-location with z/OS resource managers such as DB2 and CICS will generally route to the z/OS server. If, however, at a specific point in time, the z/OS server is exceptionally busy, the work may route to the Linux server. Likewise, work that primarily accesses non-z/OS resources may tend to route to the Linux servers first. This is a major step toward allowing the infrastructure to make dynamic decisions about task placement based on up to the minute operational considerations.

**Application versioning and roll-out**
The ODR offers a number of additional beneficial services. It supports application versioning. This vastly simplifies the introduction of new applications into a collection of operational servers. It is now possible to automate the rollout of a change across a cluster of servers without any disruption for the users. New versions may be introduced gradually, one server instance at a time, allowing for some users to be routed to the older version while others, perhaps a selected subset of the user community, are routed to the new version. The need to devise and manage your own process for application upgrade is eliminated.

**Optimizing I/O**

Often, application performance may benefit by intelligently routing requests to specific servers in order to optimize the effectiveness of a cache. Rather than randomly distributing requests across a collection of servers, establishing a uniform distribution pattern predicated on an awareness of the underlying data access pattern can yield dramatic results. WXD facilitates this by supporting two key technologies; partitioning and an object grid. The first approach routes work to specific servers based on the content of the URL request. For example, in a bidding system, routing all bids for product XYZ to a specific server will minimize stress on the data cache and data associated with this product and the associated bidders will tend to reside in memory. The second approach is to setup a shared object pool that spans several servers. A change to an object in one pool is replicated to other pools.

WebSphere application servers running on z/OS benefit from DB2 data sharing which provides a highly efficient caching mechanism. Leveraging the coupling facility, each WebSphere application server instance running in a different logical partition (LPAR) has fast cross-memory access to the local DB2 cache. DB2 data sharing ensures that the cache contents are valid. When a change to the cache contents occurs in one LPAR, the corresponding content in other LPARs is instantly invalidated. If the server in another LPAR attempts to access the invalidated cache content, it is directed to the coupling facility and/or the data base itself to retrieve the new content. This highly efficient mechanism has been fine tuned over the years to support CICS and IMS and WebSphere application server on z/OS benefits from this optimization.

**Monitoring servers**

WXD extends the support built into the WebSphere application server for monitoring server activity. The Health Manager and Visualization engine provided by WXD allows you to understand the status and relative health of a collection of servers. Statistics are captured to enable analysis of trends.

**Benefits and justification**
WXD extends the capability of the application server to provide greater efficiency, improved operational control, and enhanced availability to dramatically lower the cost associated with managing a collection of servers in addition to extending the J2EE programming model into the realm of long running batch processing.

For a distributed system configuration, the justification is often driven by the value of increasing utilization of physical assets. For z/OS, which delivers this capability inherently within the operating system, the benefits are derived by improving the operational environment.

Application versioning offers significant value and greatly simplifies the non-disruptive introduction of new versions of an application. By keeping the application up and running across a version change, a 24X7 environment is supported offering better service to the end users.

Workload routing benefits are application dependent. But, when you need this flexibility, it saves the time that would be necessary to design and implement a workaround.

Likewise, partitioning and the object grid value depend upon the application pattern. Some scenarios have improved throughput by more than 30% while reducing overall system utilization. So, the benefits can be substantial if the application pattern lends itself to cache optimization.

The benefit associated with the advanced monitoring and notification features can be substantial. Alerts can replace manual monitoring which can have a direct impact on the cost of operations and help to detect issues early.

Finally, the batch support offers a unique opportunity to tie short running and long running tasks together. Some have attempted to design and build this type of capability with existing J2EE constructs and the results have been, more times than not, less than fulfilling. This is because batch support requires a number of features to be effective and it’s difficult to construct these solely at the application layer. The WXD batch support provides the necessary infrastructure to make Java batch processing practical. It covers many aspects of batch processing that might be overlooked in a simple design since it is patterned after the highly successful batch processing support provided by JES on z/OS.

The justification for WXD is dependent upon the environment that it will support. Monitoring and notification is useful to nearly every environment. Batch, partitioning and data grid benefits require an assessment of application needs. When this type of support is required, the benefits of an off the shelf implementation versus a home grown one can be substantial, both in the short term and over the long term.

As systems evolve to incorporate greater interaction across diverse server platforms, the value of the On-Demand Router and its ability to route work based on service levels increases. The ODR represents a major breakthrough by bringing diverse server platforms together under a uniform service and workload distribution policy.