



January 30, 2008

Consolidating Java Workloads On System Z

zAAP Specialty Engine As The Enabler

by **Brad Day**

with Simon Yates and Rachel A. Dines

EXECUTIVE SUMMARY

Many Forrester clients are considering both the technical and economic benefits of running Java workloads on IBM System z. The technology enabler that makes this compelling is a specialty engine called an IBM zAAP, or System z Application Assist Processor. IBM zAAP allows you to consolidate and integrate Java-based Web applications with your core business database environment. It provides a more cost-effective, specialized z/OS Java execution environment than running the application server and associated database management system (DBMS) environment on separate physical servers.

QUESTIONS

1. How do zAAPs work on IBM System z?
2. What is the core value proposition in consolidating Java workloads using zAAP?
3. What are the limitations of zAAPs?
4. How do I determine if using zAAP is right for an application?
5. What performance improvements can I expect in native Java workloads?
6. How can I estimate the zAAP performance of new Java workloads?

SYSTEM Z'S JAVA SPECIALTY ENGINE — WORKLOAD OPTIMIZATION ON A BUDGET

The integrated for Linux (IFL) specialty engine created a strong business case for the consideration of System z as a highly available, extremely secure consolidation target for running Linux applications. zAAP specialty engine now makes a strong case for consolidating Java workloads onto your System z because it simplifies and reduces distributed server infrastructures, maximizes the value of mainframe investments, and lowers the life-cycle costs for technology-based applications.

1. How do zAAPs work on IBM System z?

When configured with the System z general-purpose processors within logical partitions running the z/OS operating system, zAAPs operate asynchronously with the general processors to execute Java programming under control of the IBM Java Virtual Machine (JVM). In addition, the zAAP specialty engine offloads the general-purpose processor, freeing it up to be re-allocated to other application workloads. JVM processing cycles can be executed on the configured zAAP with no modifications to the Java applications. In fact, the execution of the JVM processing cycles on a zAAP is based on the three

software prerequisites: the IBM software development kit (SDK) 1.4, the z/OS 1.6 operating system, and the Processor Resource/Systems Manager (PR/SM).

2. What is the core value proposition in consolidating Java workloads using zAAP?

zAAP can enable your System z to run Java Web applications side-by-side with mission-critical data for a more tightly integrated, highly secure, and efficient application and DBMS serving environment. Consolidating Java applications with the same z/OS logical partition (LPAR) as their associated database subsystems simplifies server infrastructures and improves operational efficiencies. This is accomplished by reducing the number of TCP/IP programming stacks, firewalls, physical connections, and their associated processing latencies that might otherwise be required when application and database servers are deployed on separate physical servers.

zAAPs are priced at a one-time-only fee of \$125,000 each, which is significantly less than the combined one-time fee as well as monthly license charges often incurred for software running on the general-purpose CPs. Using zAAP specialty engines, you can lower the cost of computing for z/OS Java-based applications like WebSphere by allowing the Java programming to execute specifically on the lower-cost zAAP, reducing the demand and capacity requirements on the general-purpose CPs. In addition, zAAP lets you purchase additional processing power exclusively for Java workload execution without affecting the total MSU (millions of service units) rating or machine model designation, as zAAP doesn't carry a rated pricing capacity. Essentially, IBM doesn't impose software charges on zAAP capacity. Finally, zAAP reduces charges for sub-capacity-eligible IBM software products by lowering the overall "rolling 4-hour average" MSU for logical partitions with assigned zAAPs.

3. What are the limitations of zAAPs?

The purpose of the zAAP is to provide a specialized engine at an attractive price to support specific Java workloads using the IBM JVM. The zAAP engine runs only in a z/OS environment with z/OS 1.6 or higher. It operates asynchronously with the general processors in a z/OS system to execute Java programming cycles with no anticipated modifications to the Java applications. While zAAPs are capable of executing up to 100% of Java cycles, the reality is that most applications are not 100% Java. In addition, general-purpose processor savings depend on the number of Java cycles the relevant applications use and on the zAAP parameters you selected.

4. How do I determine if using zAAP is right for an application?

IBM has developed a zAAP Projection Tool for Java 2 Technology Edition, SDK 1.3.1 that can determine how much time spent executing Java code would be eligible for execution on zAAPs. Running a Java workload that is actually representative of your production systems operations creates a Java log that reports how much of that workload could have executed on zAAPs. This information is useful in predicting the number of zAAPs that might be necessary to provide an optimum zAAP configuration.

5. What performance improvements can I expect in native Java workloads?

zAAPs do not provide performance improvements for Java or WebSphere workloads at this time, but when configured with general-purpose processors within logical partitions running z/OS, they do help lower the cost of computing for z/OS Java technology-based applications.

6. How can I estimate the zAAP performance of new Java workloads?

The amount of general-purpose processor savings will vary based on the amount of Java application code that can actually be executed on the zAAP. It follows that this depends on the number of Java cycles used by the application and on the zAAP execution model that you select.

You may want to consider the use of zAAPs to support application workloads that aren't currently running on general-purpose processors. Since the actual measurements are not possible, an estimate can be obtained using a known workload to evaluate the ratio of time of Java work eligible to execute on a zAAP versus a general-purpose processor. Since the zAAP can only run Java workload, you must consider the service level of all eligible Java work only when trying to determine the sustained use of zAAPs. To estimate how a zAAP can effectively offload the CP of its Java compute cycles, IBM has measured several known workloads to estimate the ratio of Java execution time to total system CPU time. The Java percent of total CPU time can vary for any given workload. Many factors, such as type of processor configurations; the software levels including z/OS, WebSphere, and the SDK; and the system/subsystem tuning and customizable parameters, can influence Java execution time or total system time.

IBM ran two industry-standard Java benchmarks, both using zAAP, instead of a general processor. The actual performance impact of offloading Java to the zAAP from the general-purpose CP varied, based on how much of the actual code consisted of pure Java. The first application workload test, called ML Parse, ran at 98% of the total sustained available CPU time, suggesting that the Java content of this application workload was extremely high. The XML Parse workload test is a CPU-intensive parsing workload that repeatedly parses XML documents with or without validity checking. The second workload, however, called Trade 3, ran at only 60% of the total sustained available CPU time, suggesting that the actual Java content of this application workload was substantially lower. The Trade 3 application workload benchmark is a third-generation WebSphere end-to-end trading benchmark covering the use of J2EE 1.3, including the Enterprise JavaBeans (EJB) 2.0 component architecture with point-to-point asynchronous messaging. This benchmark is characterized as light SQL processing with a small database component.