



IBM Software Group

Java on System z

Marcel Mitran
Senior Technical Manager
IBM Java Technology Center

Compilation Technology

* Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

© 2008 IBM Corporation

Performance - Disclaimer

- Based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput improvements equivalent to the performance ratios stated here.

Agenda

1. Java – IBM's Approach
 2. Java on System z
 - SDK Extensions for zOS
 - jZOS
 - Execution environments
 - zAAPs
 3. Java/COBOL Interoperability
 4. System z Hardware Exploitation by Java
 5. Performance Data
 - System z10™ Performance
 - SDK6
- Appendices/Backup

1. Java – IBM's Approach

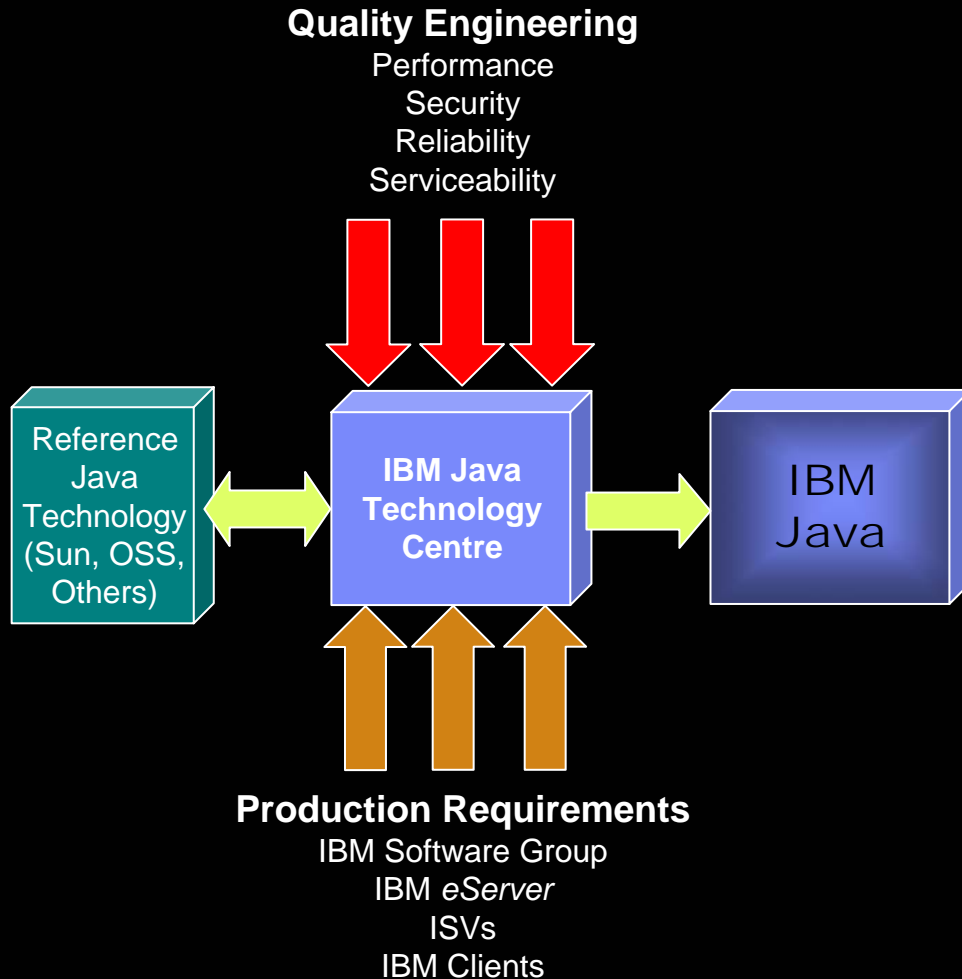
IBM and Java

- **Java is critically important to IBM**
 - As a fundamental infrastructure for IBM's software portfolio including WebSphere and Rational products.

- **IBM is investing strategically in Java and in virtual machines**
 - As of Java 5.0 Single Java™ platform supporting ME, SE and EE
 - New technology base (J9 JVM and TR-JIT) on which to delivery improved performance, reliability and serviceability

- **IBM is also looking to engender accelerated public innovation in Java**
 - Support of Eclipse, Apache (XML, Derby, Geronimo, Harmony, Tuscan, ...)
 - Broad participation in relevant open standards bodies such as JCP, OSGi

IBM's Approach to Java Technology



- ✓ *Listen to and act upon market requirements*
- ✓ *World class service and support*
- ✓ *Available on more platforms than any other Java implementation*
- ✓ *Highly optimized*
- ✓ *Embedded in IBM's middleware portfolio and available to ISV partners*

JSE Road Map

SUN SDKs

Sun Supported Platforms at Java 5.0

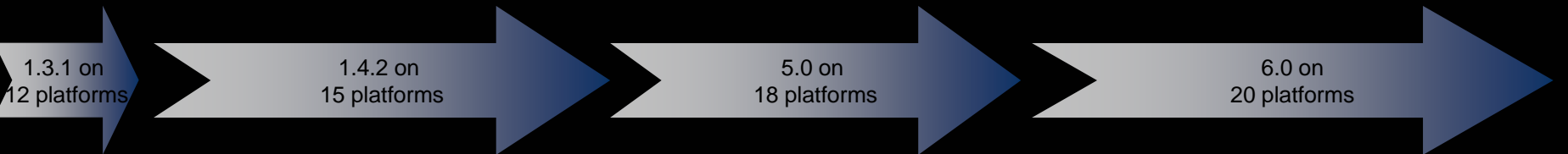
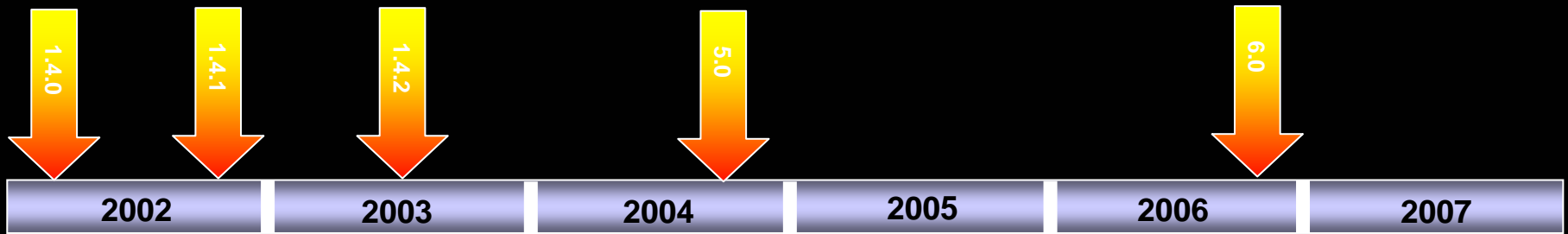
- Solaris x64
- Solaris SPARC
- Solaris x86
- Windows IA32
- Windows AMD64
- Windows EAMT64
- Linux IA32
- Linux AMD64
- Linux EAMT64

Java 5.0 delivers

- New Language features
 - Autoboxing
 - Enumerated types
 - Generics
 - Meta Data

Java 6.0 delivers

- Performance improvements
- Improved UI
- Client WebServices Support



IBM SDKs

IBM Supported Platforms at Java 6.0

- AIX PPC 32
- AIX PPC 64
- Linux PPC 32
- Linux PPC 64
- zLinux 31
- zLinux 64
- HP-UX PA-RISC 32
- HP-UX PA-RISC 64
- HP-UX IA64
- zOS 31
- zOS 64
- Solaris x64
- Solaris SPARC
- Solaris x86
- Windows IA32
- Windows AMD64
- Windows EAMT64
- Linux IA32
- Linux AMD64
- Linux EAMT64

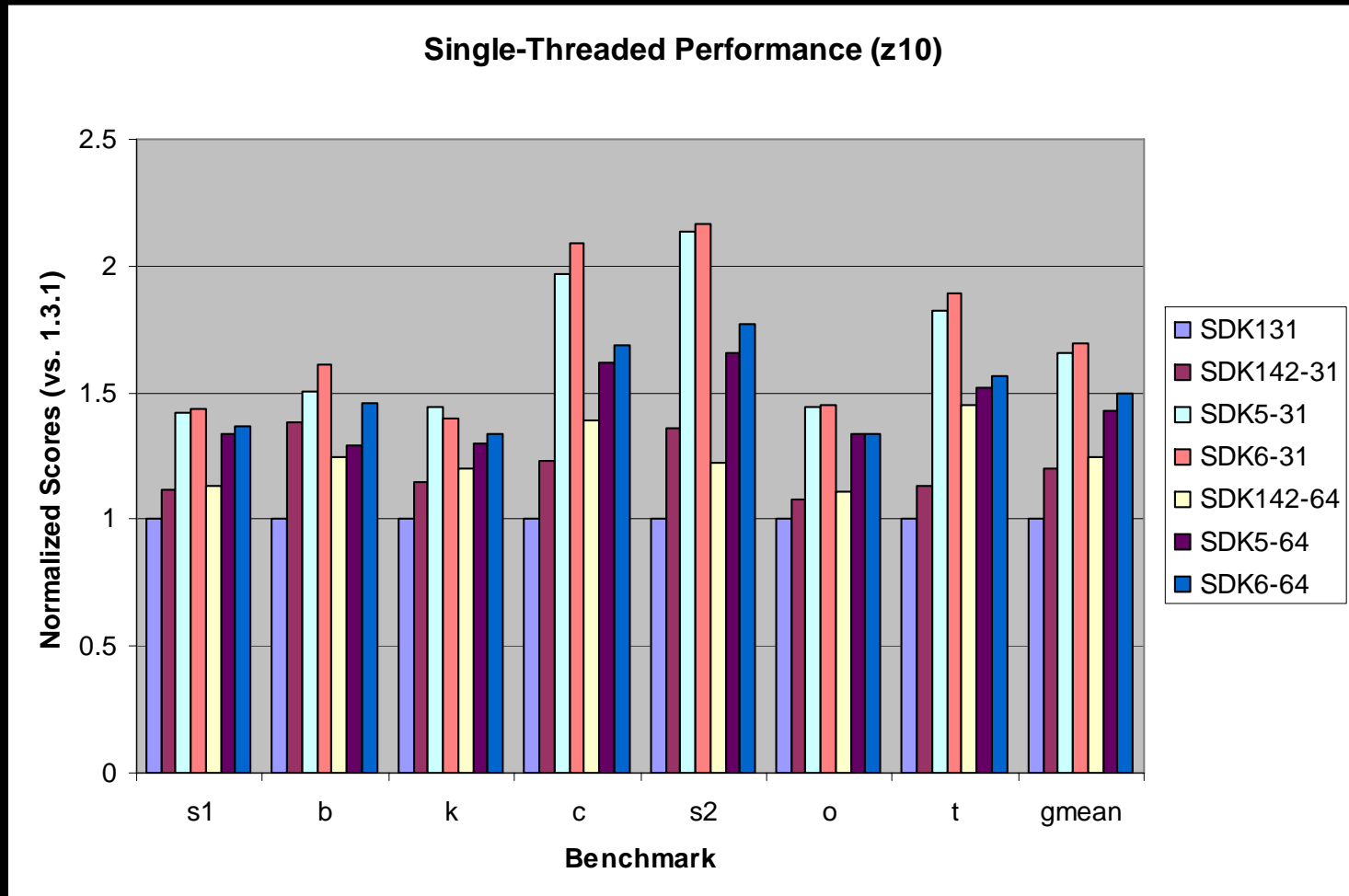
IBM Java 5.0 features

- Improved performance
 - Generational Garbage Collector
 - Shared classes support
 - New JIT technology
- First Failure Data Capture
- Configurable Trace
- Full Speed Debug
- Hot Code Replace
- Common runtime technology
 - ME, SE, EE

IBM Java 6.0 features

- Improvements in
 - Platform coverage
 - Performance
 - Serviceability tooling
- New Functionality
 - IBM WebSphere Real-Time V1.0
- XML parser improvements
- z10 Exploitation
 - DFP exploitation for BigDecimal
 - Large Pages
 - New ISA features

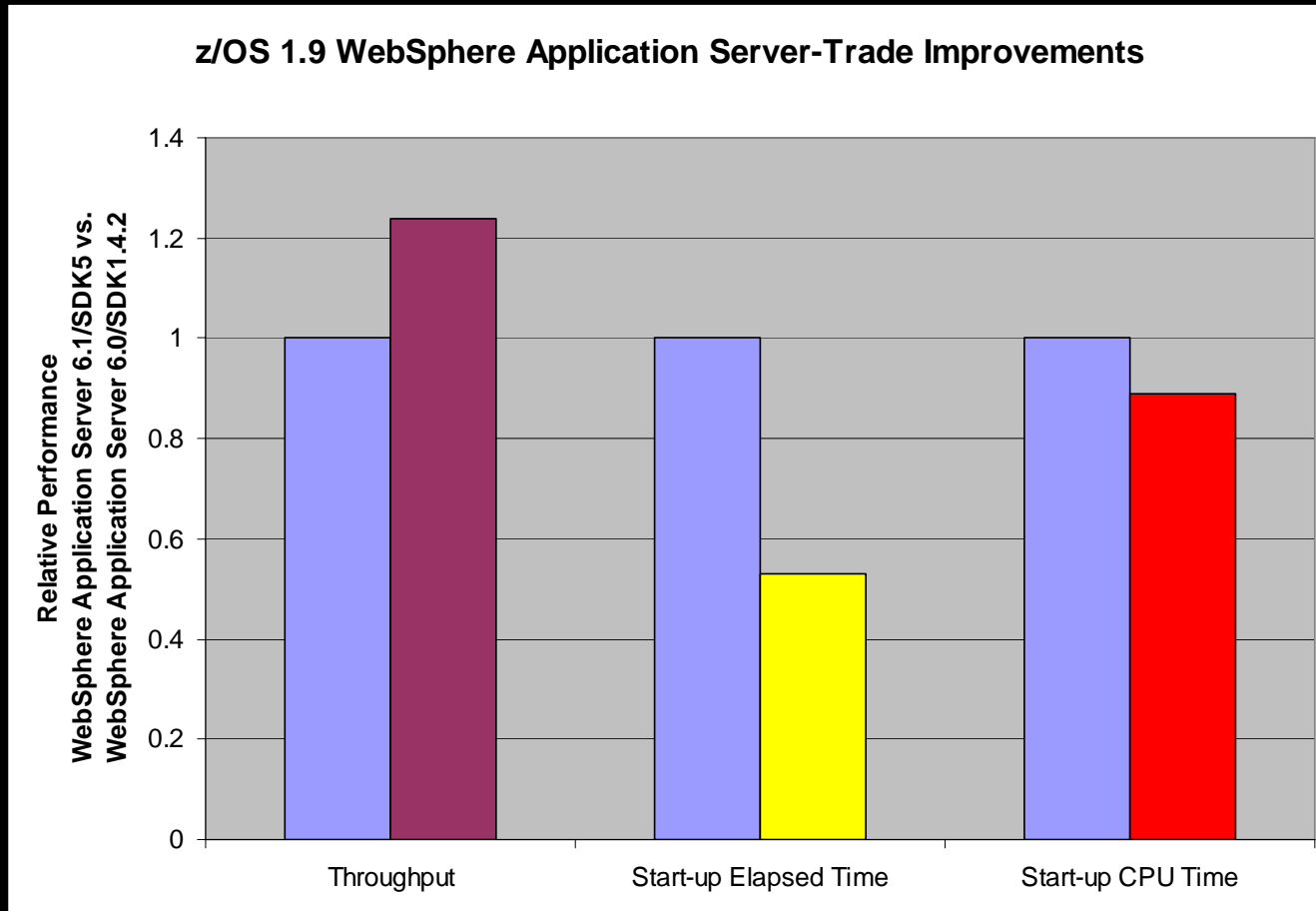
SDK6 – Performance – Single Threaded Benchmarks



Bigger is better

z10 – zOS V1.9

Performance – WAS6.0 to WAS6.1 Trade6 on zOS



z10 - zOS V1.9

Resources

- **Documentation**

- <http://www.ibm.com/developerworks/java/jdk/docs.html>

- **zOS SDK**

- <http://www.ibm.com/servers/eserver/zseries/software/java>

- **System z Linux SDK**

- <http://www.ibm.com/developerworks/java/jdk/linux/download.html>

- **GC Tuning documentation**

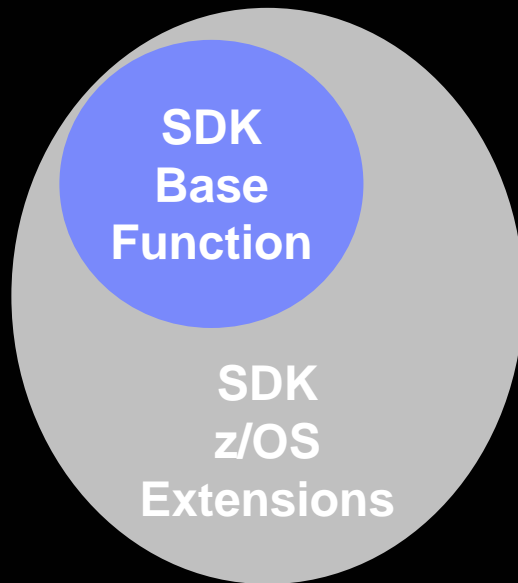
- http://www.ibm.com/developerworks/views/java/libraryview.jsp?search_by=java+technology+ibm+style:

- **GCMV (aka EVTK)**

- <http://www.ibm.com/software/support/isa/>

2. Java on System z

z/OS – System z Extensions



All SDKs support the ‘standards’, Java on z/OS extends the SDK.

- Access to z/OS services
- Access to all types of data
- Access under control of z/OS security mechanisms
- Integration into existing operational infrastructure

Services available in JEE and JSE environments under the restrictions of the container.

z/OS – System z Extensions

SDK for z/OS extensions to allow access to z/OS facilities

- JAAS wrapper of SAF (RACF, ACF2, or TopSecret)
- Traditional OS dataset access
- Cryptographic hardware (Cards and CPACF)
- z/OS Console (modify and messages)
- z/OS system logger
- JES job submission
- DFSORT
- SMF
- Etc.

System specific extension allow you to write robust middleware and applications that integrate with traditional z/OS operating environment.

- Allow for maintaining platform independent design development.
- Platform specific implementations when required
- Allows for operational and resource optimization

z/OS – System z Extensions

Tight integration with data resource managers

Type-2 or local connectors for:

- o DB2
- o MQ
- o CICS
- o IMS DL/I

These connectors allow for efficient and low latency access.

Java Execution Environments

IBM offerings

- Transactional/Interactive
 - WebSphere for z/OS (WAS z/OS)
 - WebSphere Process Server for z/OS (WPS)
- Batch oriented
 - WebSphere eXtended Deployment (WAS-XD)
 - JEE based
 - WAS runtime extensions
 - JZOS - JES environment
 - JSE based

Open source or non-IBM vendor Application Server and Frameworks

- Tomcat, JBoss
- IBatis, Hibernate, Spring
- Ant

These environments allow you to capitalize on pre-existing assets, artifacts, processes, core competencies, platform strengths.

z/OS – System z Extensions

JZOS acquisition and integration

- Non OMVS environment JVM launcher
 - Allow Java job steps to be included in batch jobs
 - Traditional accounting records
 - Access via DD statements
 - Return code surfaced to subsequent steps
- Many Java wrappers of z/OS services
- Integration into standard operational infrastructure (OPC, NetView, etc.)

The System z Application Assist Processor (zAAP)

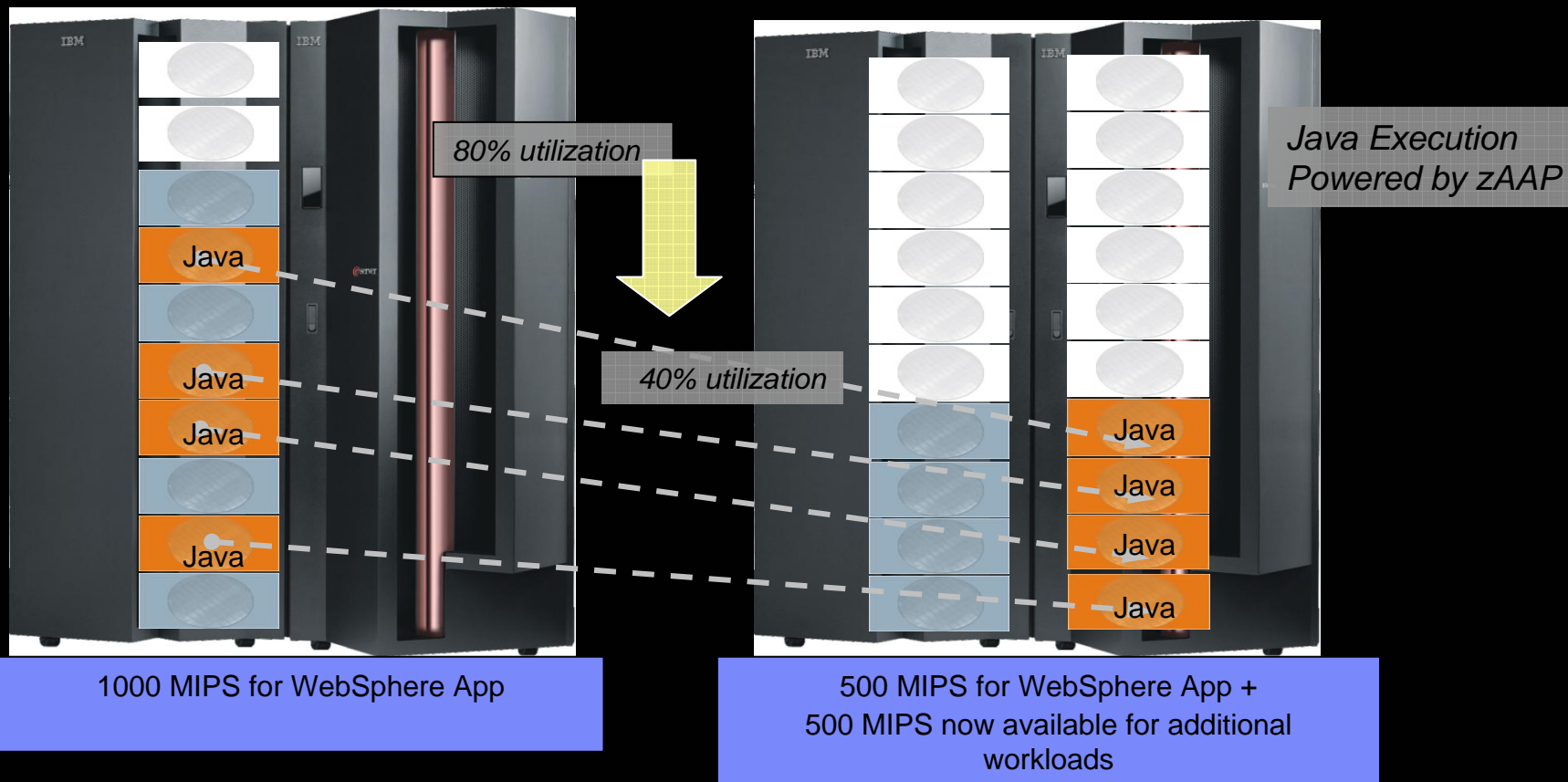
- Helps enable integration of Java-based applications into z/OS environment
- Can help simplify and reduce server infrastructure and improve operational efficiencies over distributed multi-tier solutions.
- Executes Java with the reliability, availability, and security you expect from the mainframe...with no changes to applications
- Reduces up front and maintenance costs for workloads with Java cycles e.g.: WebSphere, DB2
- Provides additional capacity at low cost of acquisition to process Java without affecting MSU rating or machine model designation...No additional IBM software costs



Objective: Enable integration of new Java based Web applications with core z/OS backend database environment for high performance, reliability, availability, security, and lower total cost of ownership

zAAP: An Example...

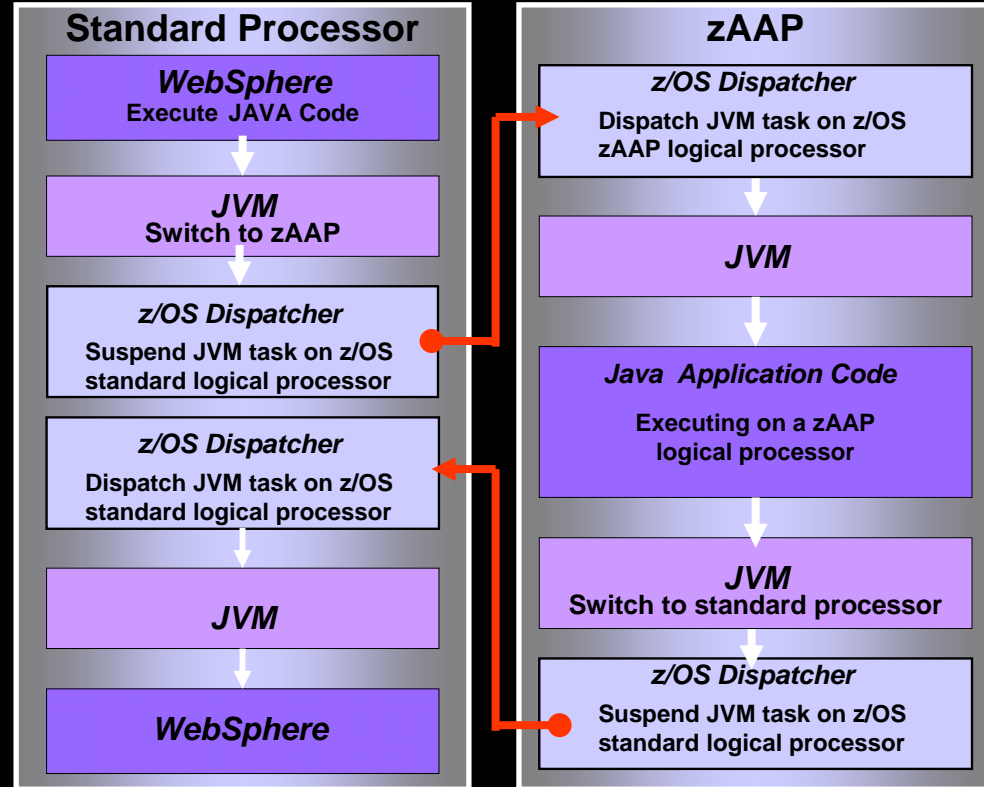
WebSphere Java-based Application that is transactional in nature and requires 1000 MIPS on System z.



In this example, with zAAP, we can reduce the standard CP capacity requirement for the Application to 500 MIPS or a 50% reduction. * For illustrative purposes only

zAAP Architecture and Workflow: Executing Java under IBM JVM control

- IBM JVM, parts of LE runtime, and z/OS Supervisor needed to support JVM execution can operate on zAAPs
- IBM JVM communicates to z/OS dispatcher when Java code is to be executed
 - When Java is to be executed, the work unit is "eligible" to be dispatched on a zAAP
- z/OS dispatcher attempts to dispatch zAAP eligible work on a zAAP (when present)
 - zAAP ineligible work only dispatched on standard processors
- If there is insufficient zAAP capacity available, or standard processors are idle, the dispatcher may dispatch zAAP eligible work on a standard processor
 - There is an installation control to limit the use of standard processors to execute zAAP eligible work (see Java code execution options)



Resources

- **zAAP site** = <http://www-03.ibm.com/systems/z/zaap/>
- **Redbook: zAAP Implementation, SG24-6386**
- **Redbook: Java Stand-alone Applications on z/OS I & II, SG24-7177-00, SG24-7291-00**
- **JZOS site** = <http://www-03.ibm.com/servers/eserver/zseries/software/java/products/jzos/overview.html>
- **JZOS Batch Launcher and Toolkit Installation and User's Guide, SA23-2245-00**
- **Techdocs**
 - **#TD102183 New SMF Support for zAAPs and SMF Accounting**
 - **#PRS1632 Everything zAAP Technical Presentation**
 - **#TD102878 Viewing Potential zAAP Workload on non-zAAP Systems with OMEGAMON XE on z/OS**
 - **#PRS1224 Implementing zAAPs in the CICS Environment**
 - **FLASH10432 z/OS: Dispatcher Enhancements for zAAPs**
 - **#WP100431 Obtaining the zAAP Usage Estimation Information in WebSphere for z/OS Version 5**
 - **#PRS1088 Checklist for a Successful Migration to z990, z890 and zAAP**
 - **#TD103516 Specialty Engines zIIP and zAAP Software Update**
 - **#TD103460 zAAP Estimation with Java5 and WebSphere for z/OS V6.1 11/15/2006 Mike Cox**
 - **#WP100417 z/OS Performance: Capacity Planning Considerations for zAAP Processors**

3. COBOL/Java Inter-Operability

COBOL/Java Inter-Operability

- **Object-oriented COBOL syntax**
 - enables COBOL and Java interoperation within an address space
 - COBOL invocation of Java
 - COBOL class defines methods that can be invoked from Java
- **Implementation based on the Java Native Interface (JNI)**
 - COBOL INVOKE statement maps onto Java JNI calls
 - COBOL class methods definitions define Java native methods
- **Mapping of Java data types to / from COBOL**
- **Support for JNI programming in COBOL**
 - COBOL COPY file analogous to jni.h, enables access to JNI callable services
- **Prerequisite: IBM Java 2 Technology Edition SDK 1.4**
 - Java 5 and Java 6 support planned for next COBOL delivery

COBOL/Java interoperation - environments

- **z/OS Unix**
- **z/OS Batch**
 - Use BPXBATCH utility or JZOS
- **IMS Java dependent regions**
 - JMP - Java Message Processing region
 - JBP - Java Batch Processing region
- **Windows**
 - with COBOL component of RDz
- **AIX**
 - with IBM COBOL for AIX

Note: not supported under CICS

- *Under CICS, Java and COBOL can interoperate using JCICS commands*

4. System z Hardware Exploitation

System z Support – Hardware exploitation

- **New Hardware Facility Exploitation**
 - z10 compare-and-branch/trap facility exploitation
 - z10 Decimal Floating Point **
 - z9 Extended immediate support
 - z990 long displacement support
 - Exploiting 64-bit instructions on 31-bit JVMs for Long Arithmetic
- **Super-scalar instruction scheduler**
- **Specialized register allocation**
- **Idiom recognition framework for CISC instruction exploitation**

** *Mitran et al. Decimal floating-point in Java 6: Best practices*

https://www-304.ibm.com/jct09002c/partnerworld/wps/servlet/ContentHandler/whitepaper/power/java6_sdk/best_practice

System z Support – CISC Instruction Exploitation

- **CISC Instruction Exploitation for Specialized Loop Reductions ****

- MVC, XC, CLC, SRST and TR[OT | TO | TT | OO]

```

while (i < end) {
    value = table[arrB[i+offsetB]];
    if (value == termChar) break;
    arrA[i+offsetA] = value;
    ++i;
}

```

➔

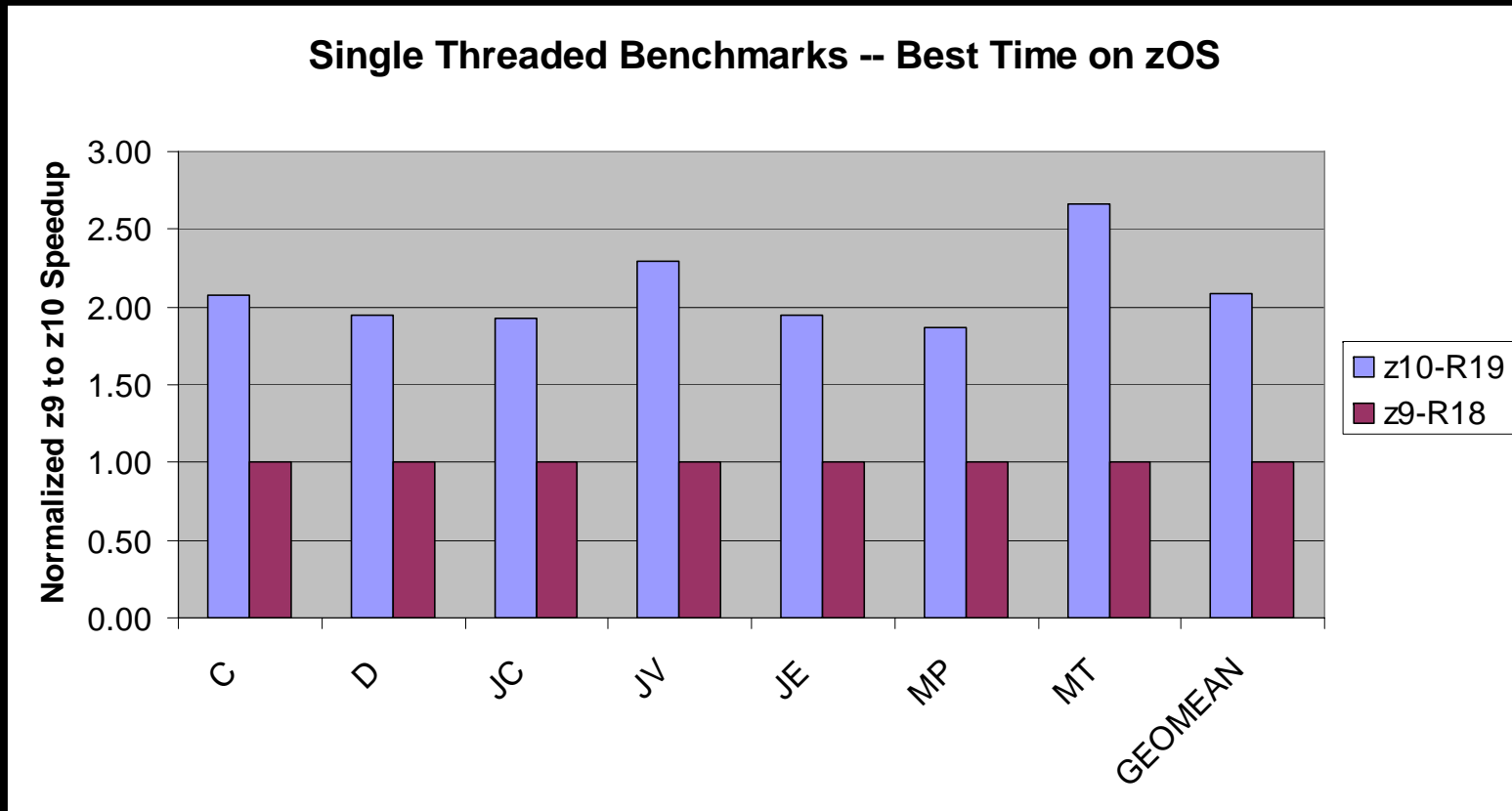
```

LB: DS 0H
TRxx    // xx depends on arrA/B types
BRC LB  // re-drive long xlate

```

** M. Kawahito, *et al.*, A new idiom recognition framework for exploiting hardware-assist instructions, ASPLOS-XII: Proceedings of the 12th international conference on Architectural support for programming languages and operating systems, 2006

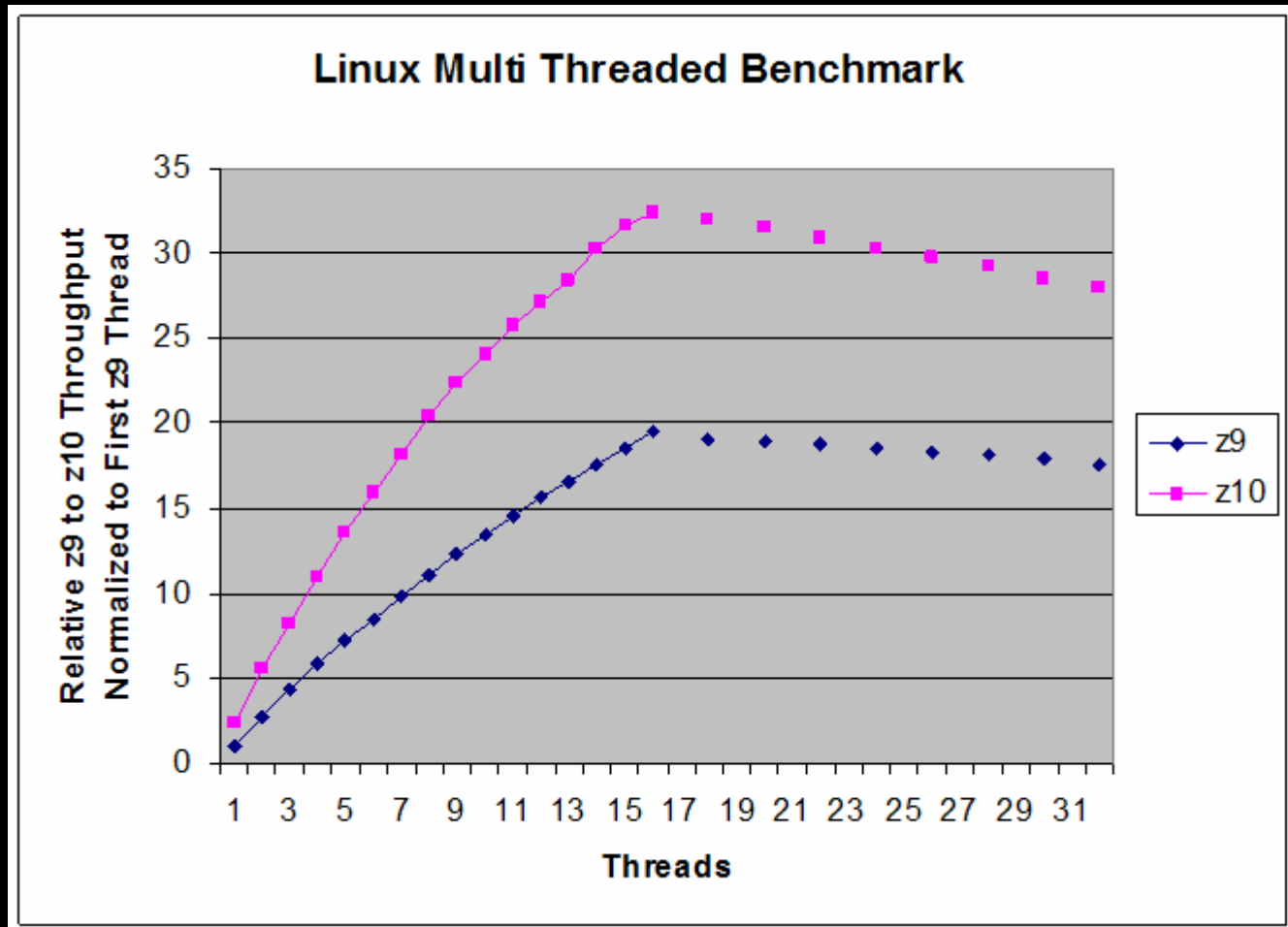
System z10 to z9 Performance – Single Threaded Benchmarks



Bigger is better

Java6 31bit - z9 - zOS V1.8 vs z10 - zOS V1.9

System z10 to z9 Performance – Multi-Threaded Benchmark

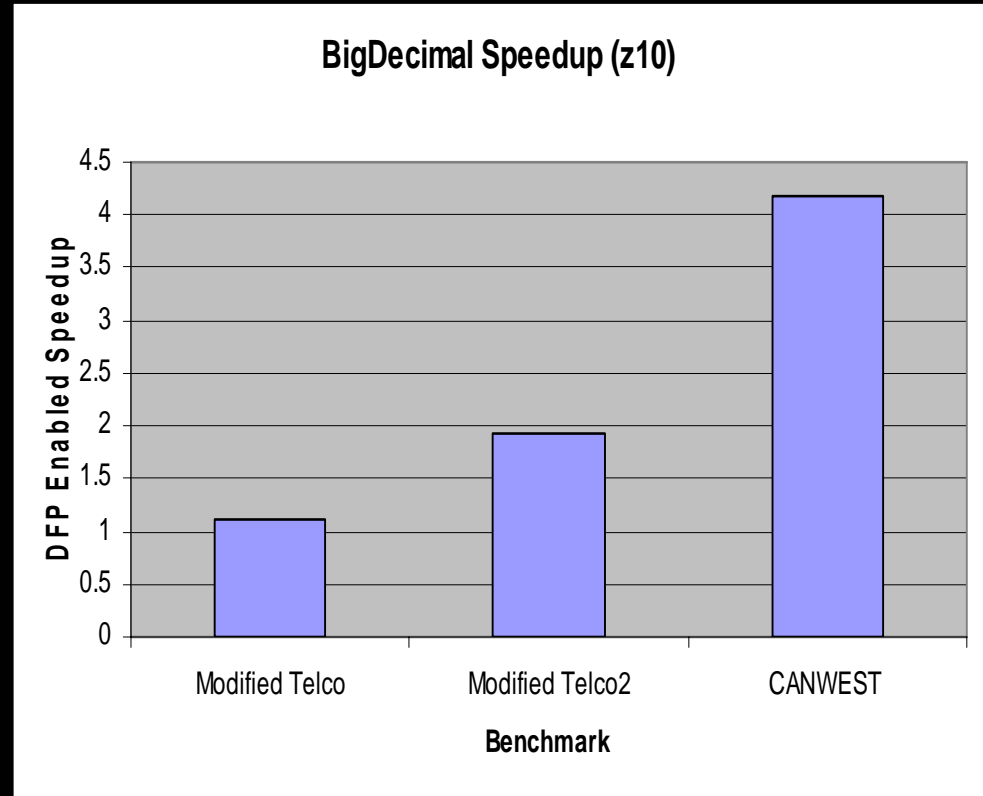


Bigger is better

Java6 31bit- z9 – SLES7 vs z10 – SLES9

BigDecimal Performance Improvements

- DFP exploited through the Java BigDecimal class
- Not all uses of BigDecimal warrant exploitation of DFP – data dependent**
- The class observes uses of BigDecimal and makes a decision about exploiting DFP
- MathContext64 must be used to ensure optimal performance



**

Mitran et al., *Decimal floating-point in Java 6: Best Practices*, Partner World, 2008

https://www-304.ibm.com/jct09002c/partnerworld/wps/servlet/ContentHandler/whitepaper/power/java6_sdk/best_practice

Bigger is better

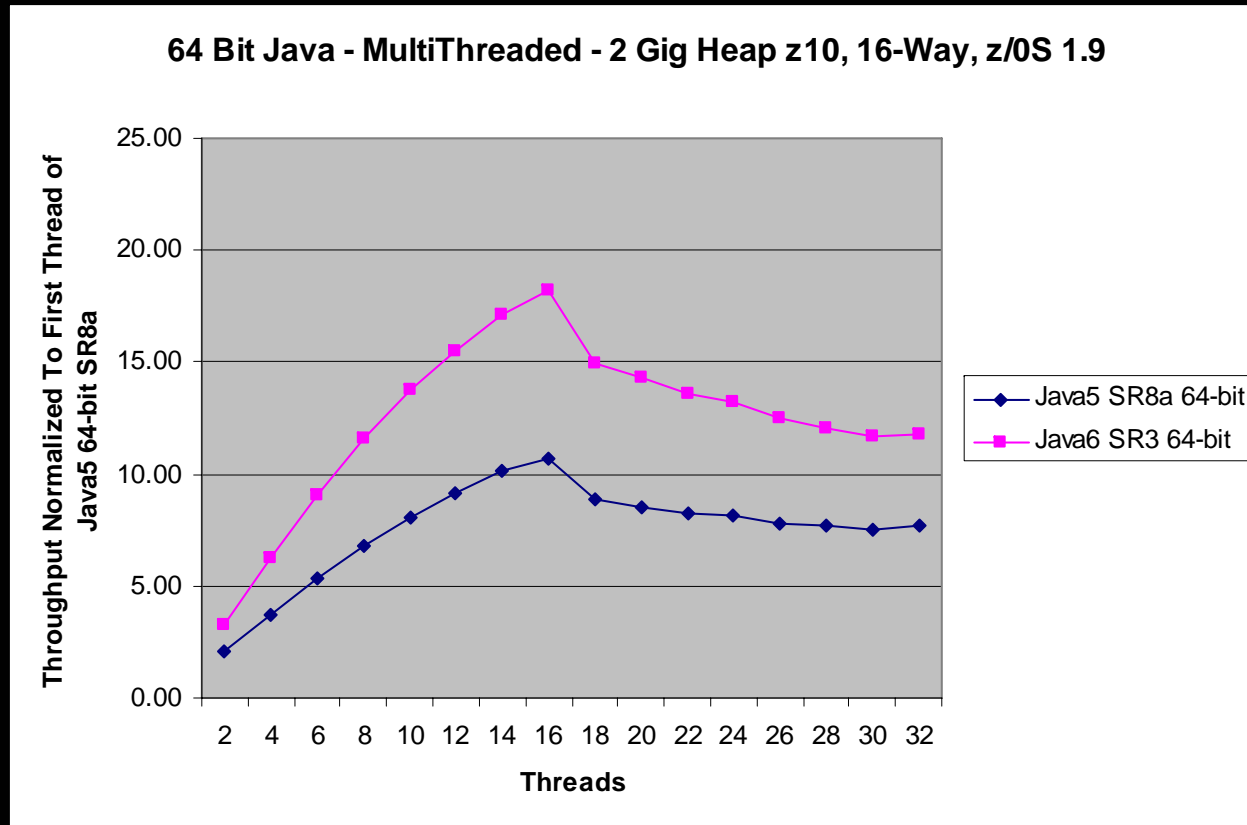
z10 – zOS V1.9 - Java 6 SR1

5. SDK6 Performance

SDK6 Performance

- What's new
 - Exploits z10 ISA features
 - Multi-threaded performance improvements
 - Garbage collection improvements
 - Class library work
 - JIT improvements
 - 64-bit SDK performance improvements in Java6 SR3
 - Compressed References (-Xcompressrefs)
 - XML performance improvements
 - Ahead-of-time JIT support for shared-classes
- <http://www.ibm.com/developerworks/java/jdk>

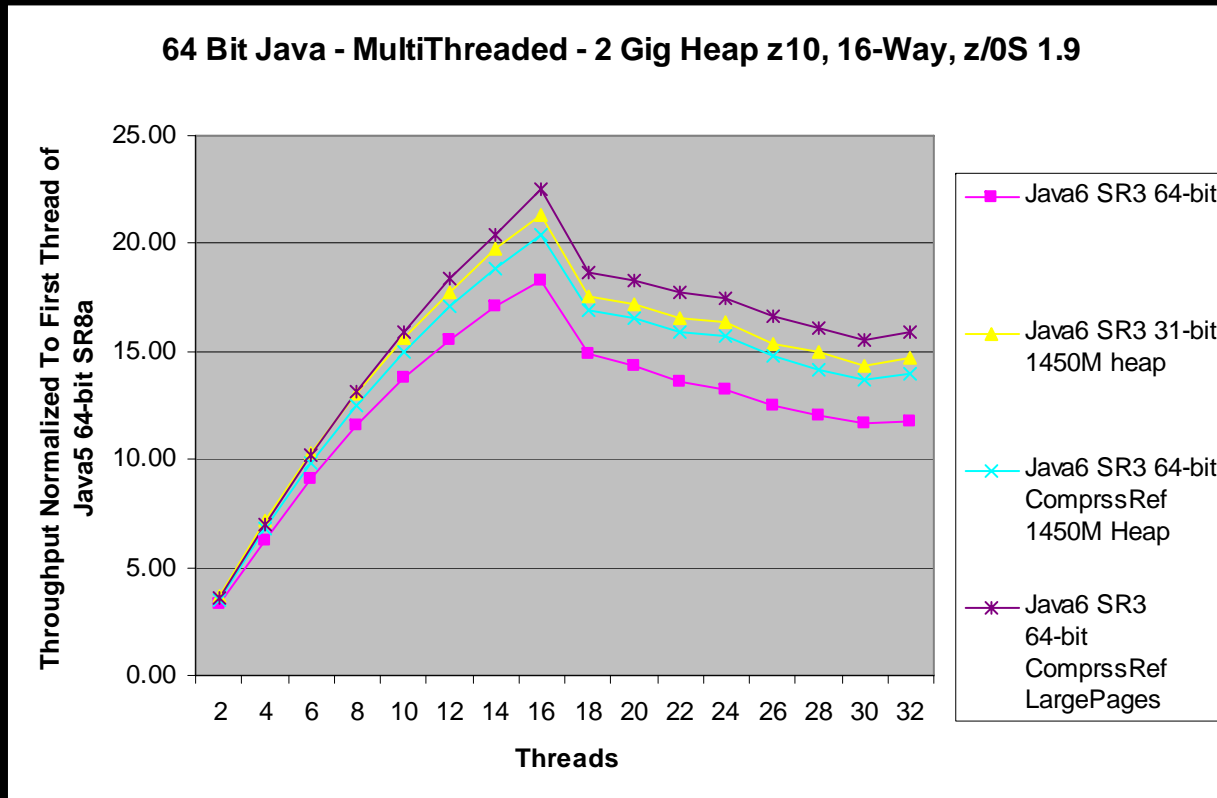
SDK6 -- Multi-Threaded Benchmark



Bigger is better

z10 - zOS V1.10

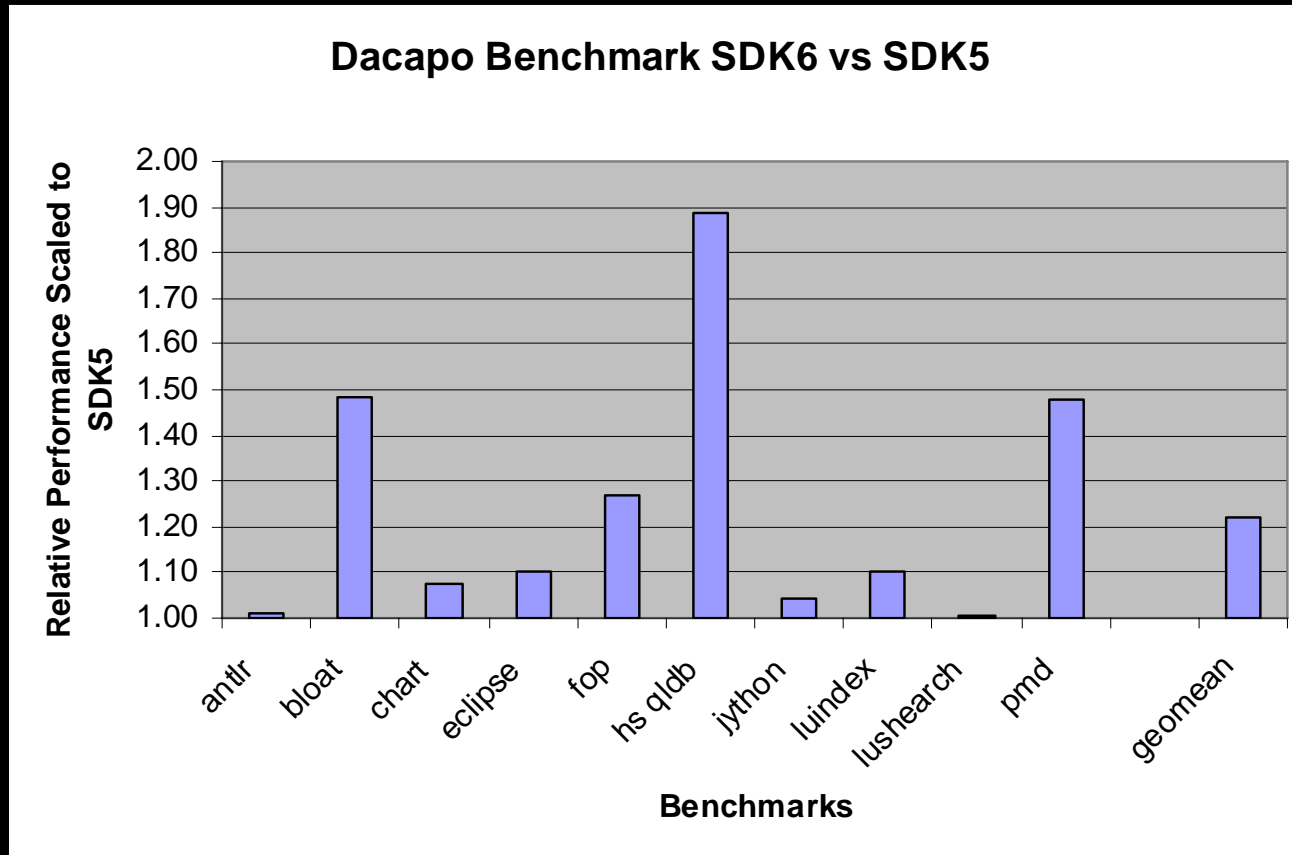
SDK6 -- Compressed References



Bigger is better

z10 - zOS V1.10

SDK6 – Dacapo Benchmark**



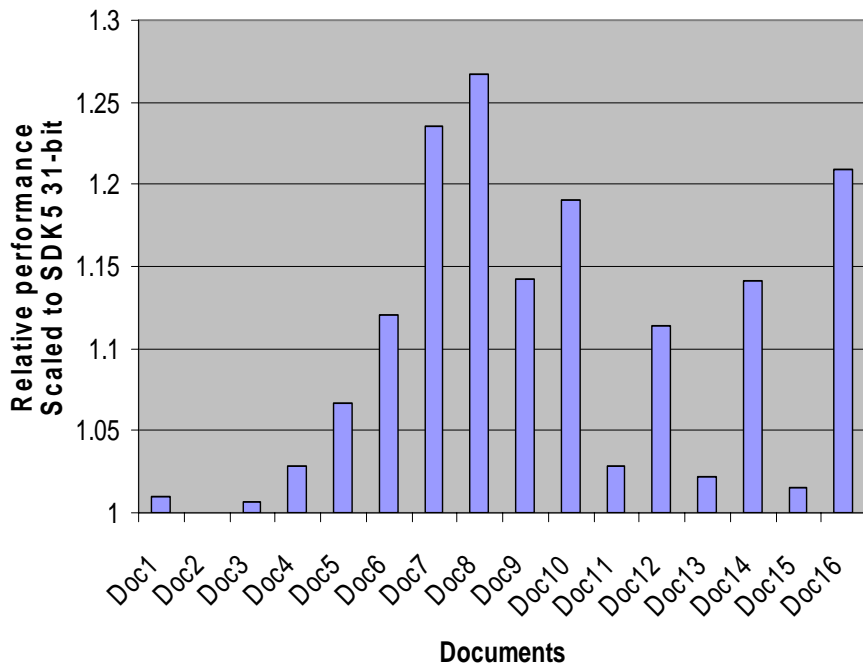
Bigger is better

z990 - zLinux SLES7

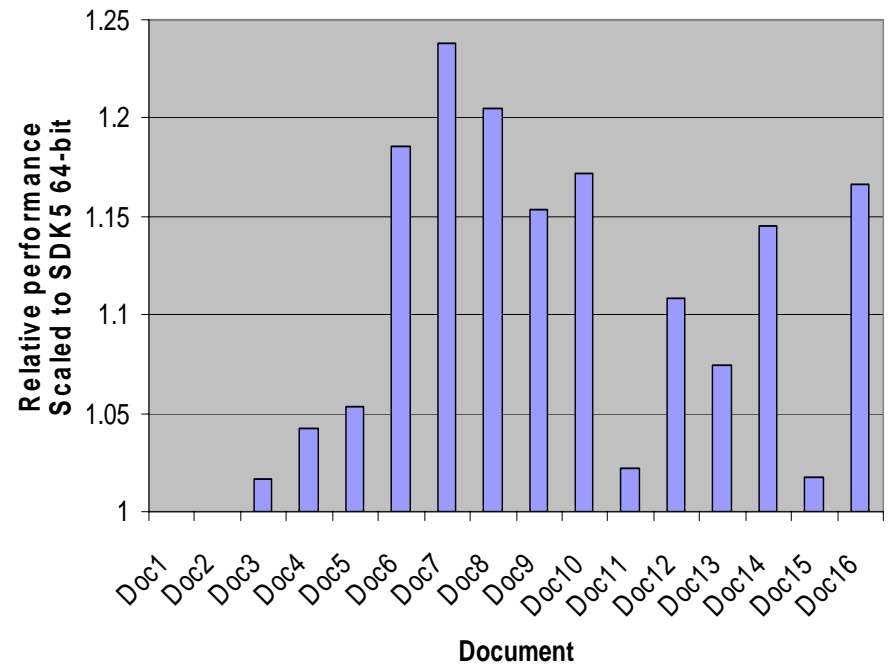
** Blackburn, S. M., *et al.* **The DaCapo Benchmarks: Java Benchmarking Development and Analysis**, OOPSLA '06, Portland, OR, October 2006

SDK6 – XML Performance

XML Parsing 31-bit SDK6 vs SDK5



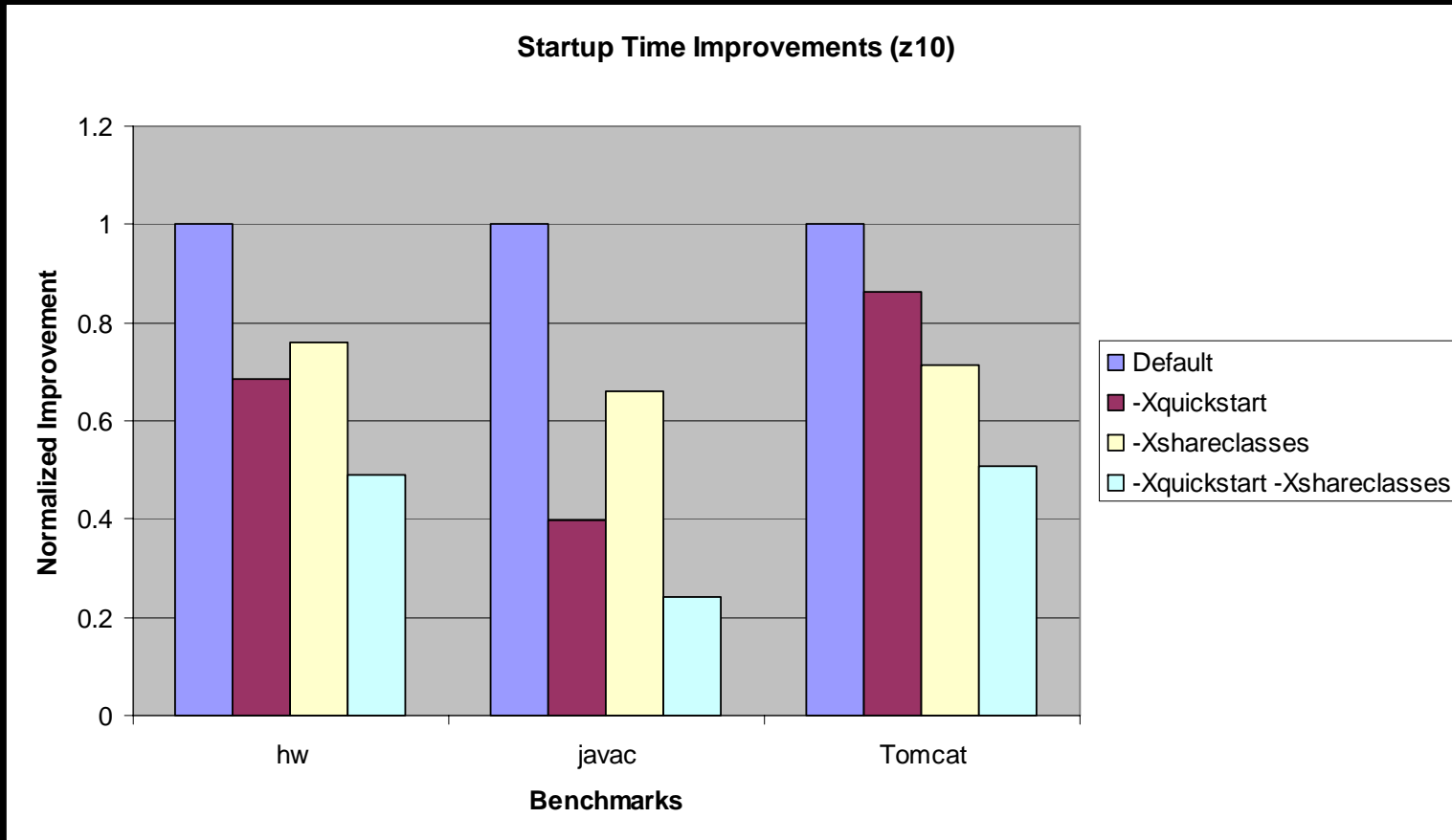
XML Parsing 64-bit SDK6 vs SDK5



Bigger is better

z9 - zOS V1.8

SDK6 – AOT Startup Time Improvements



Smaller is better

z10 - zOS V1.9

Trademarks and Copyrights

- Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both
- Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries
- Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Appendices/Backup

A. Compressed References

Compressed References: Motivation

- **31-bit address space => 2G addressable virtual**
 - Many customers reaching limits (OOM)
 - WAS7 uses 64-bit runtime as default
- **Move to 64-bit runtime is not free**
 - Object references are 2x bigger
 - 30-40% increase in Java heap footprint
 - Cache/TLB pressure doubled => throughput cost
 - Footprint growth => increased memory cost

Compressed References: Technical Details

- 32-bit Object (24 bytes – 100%)



- 64-bit Object (48 bytes – 50%)



- 64-bit Compressed References (24 bytes – 100%)



Use 32-bit values (offsets) to represent object fields

With scaling, between 4 GB and 32 GB can be addressed

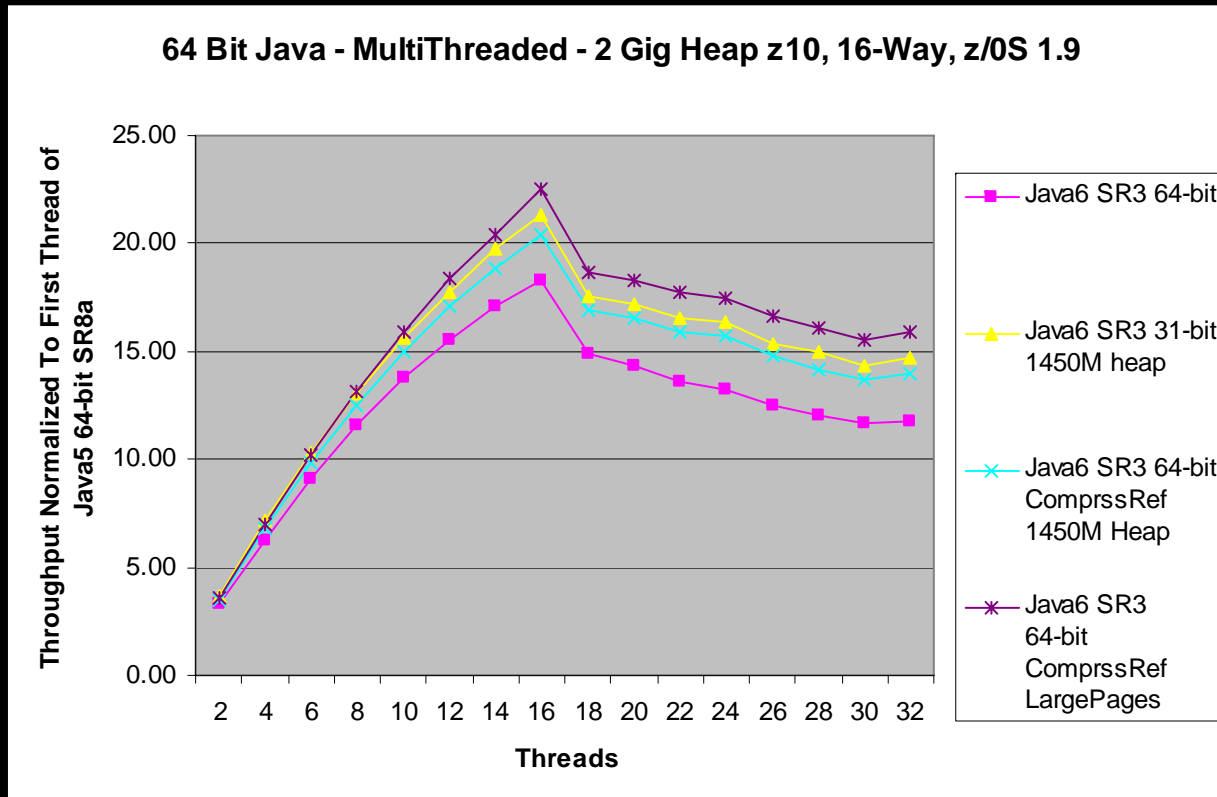
Compressed References – Java 6 SR3 Implementation

- **Option to enable compression in 64-bit Java SDK**
 - -Xcompressedrefs
- **Java objects are 8-byte aligned (low 3 bits are 0)**
 - Main idea is to store 32-bit shifted offset in objects
 - Shift values of 0, 1 and 3 are used
- **Address range restrictions**
 - Java heap allocated in 0-32GB range
- **Maximum allowable heap in theory is 32GB**
 - z/OS still limited to ~1.3GB
 - Working in conjunction with platform to allow ~30GB limit

Large Pages Performance Improvements

- z10 supports 1MB Large Pages
- Java 6 SR1 supports Large Pages
- Use: `-xlp1M`
- zOS 1.9
 - **APAR OA20902 + APAR OA25485 (End of Aug)**
- Linux
 - **SLES10 SP2 or RHEL5U2**

SDK6 -- Multi-Threaded Benchmark Prototype



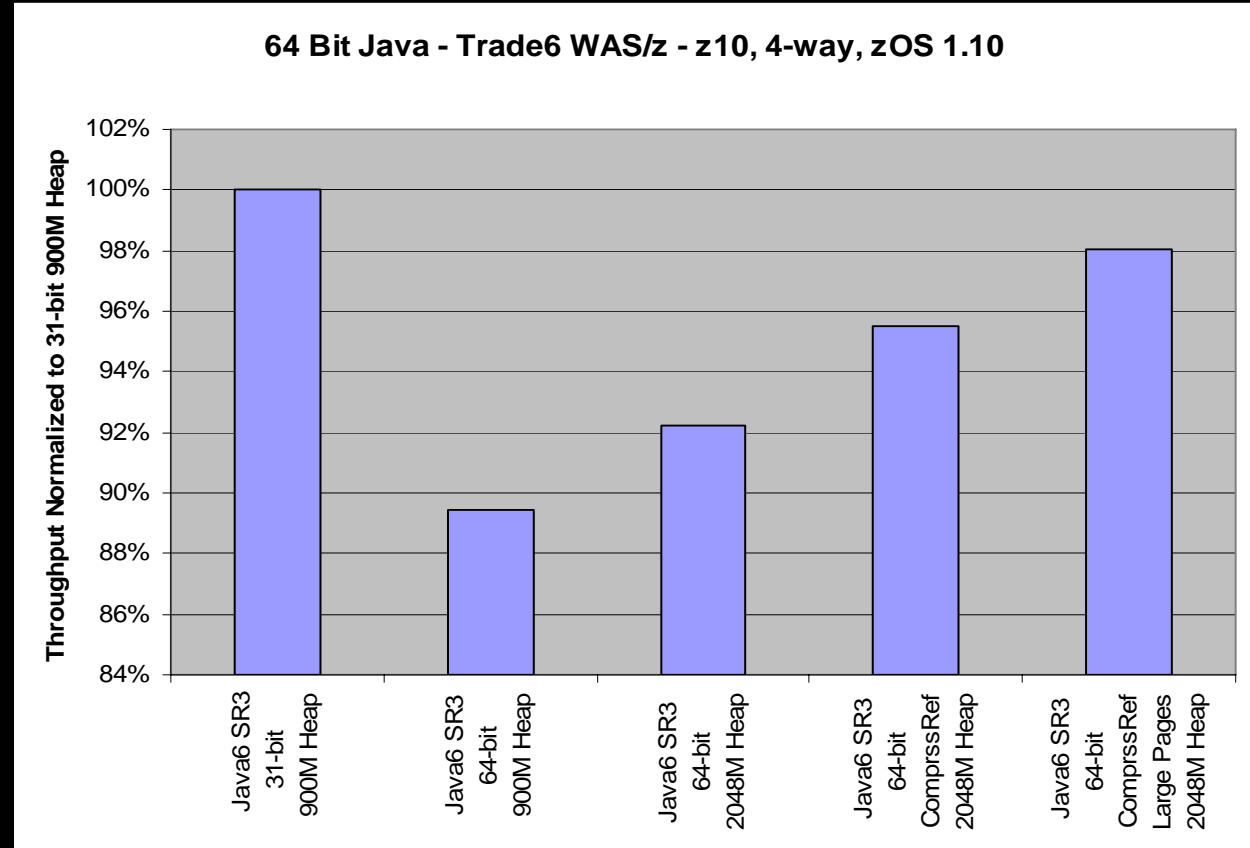
Bigger is better

z10 - zOS V1.10

SDK6 – 31-bit vs 64-bit Trade6 Prototype

Prototype performance

- 64-bit performance is within 2% of 31-bit performance
- 64-bit footprint is reduced by 30-40%

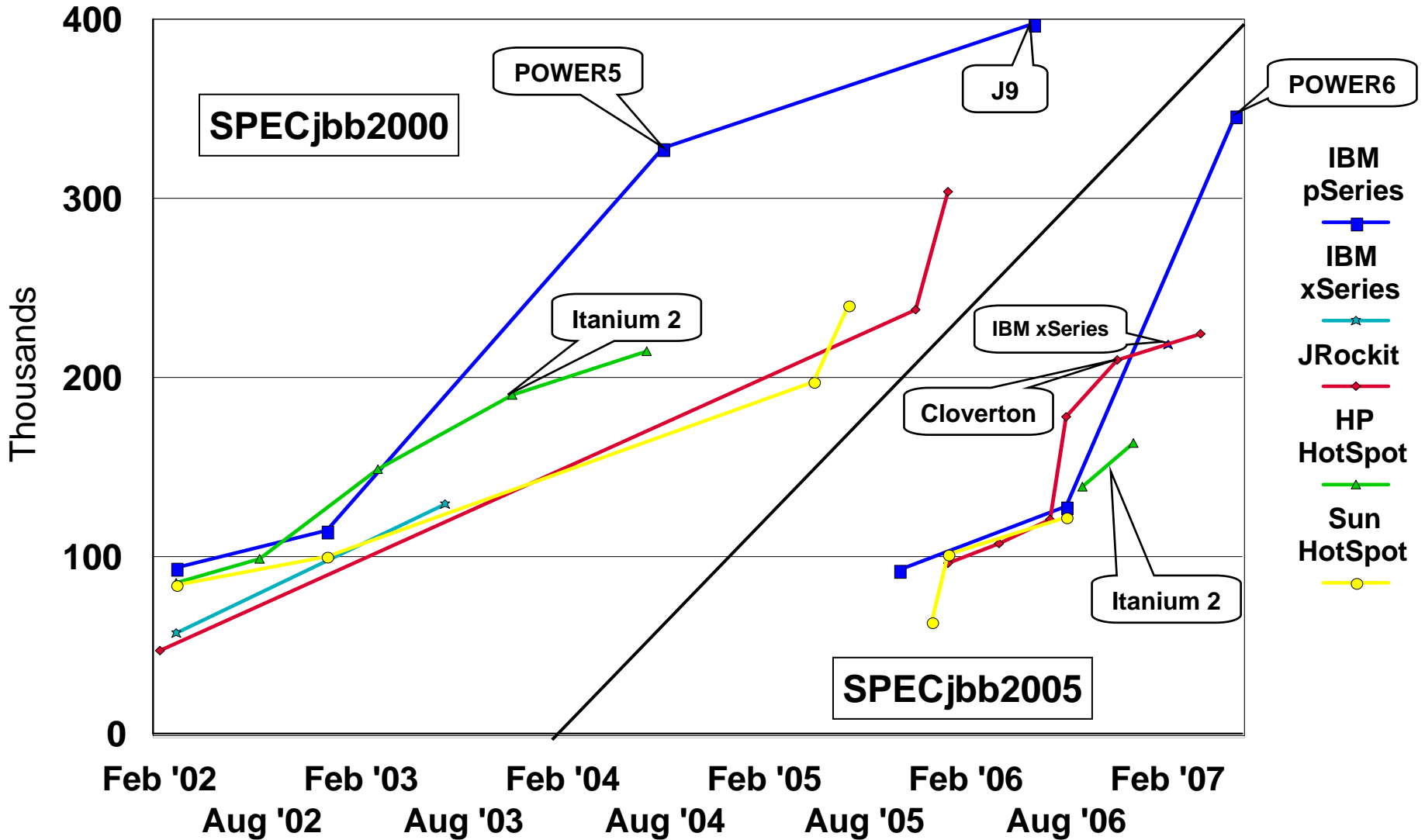


Bigger is better

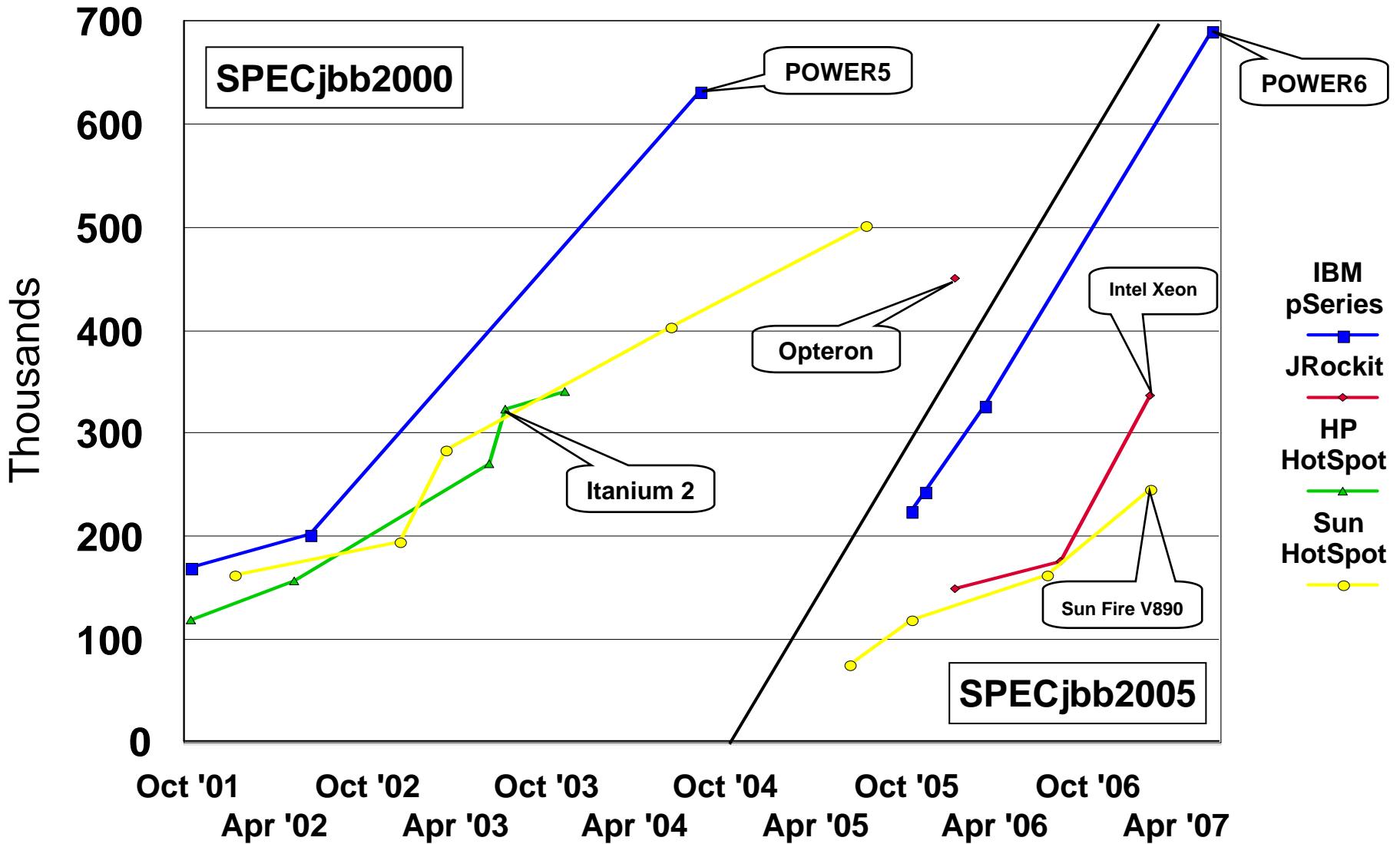
z10 - zOS V1.10

B. External Publication

SPECjbb2000 and SPECjbb2005 8 Core Published Results



SPECjbb2000 and SPECjbb2005 16 Core Published Results



C. IBM J9 Garbage Collector

What is Garbage Collection?

Conceptually, garbage collection (GC) creates the illusion of infinite free space

- Java has a create (“new”) but no destroy
- Applications create objects as needed on the Heap

In reality, GC reclaims unused memory back to the free lists

- Finds objects that are no longer used
- Makes their storage available for allocation

What is Garbage Collection?

GC is typically incurred when free space is exhausted

- Application has used all free space on the Heap through creating objects
- Virtual Machine GC's the Heap to create free space for allocation
- Execution resumes

What is Garbage Collection?

IBM Java GC has a number of selectable policies under which it will recycle objects

Why have many policies? Why not just “the best”?

- Cannot always dynamically determine what tradeoffs the user/application are willing to make
 - Pause time vs. Throughput
 - Footprint vs. Frequency

J9 – Garbage Collector Policies

- **-Xgcpolicy:optthroughput (default)**
 - Recommended starting point for tuning
 - My applications is designed around raw throughput and short GC pauses are acceptable.
 - There is no intermingling of garbage collection and application work – better raw throughput
 - The application is stopped each time garbage is collected.

- **-Xgcpolicy:optavgpause**
 - My application cannot tolerate the length of observed GC pauses. A degradation in throughput performance is acceptable as long as the GC pause time is reduced.
 - I'm concerned about user response times in interactive applications

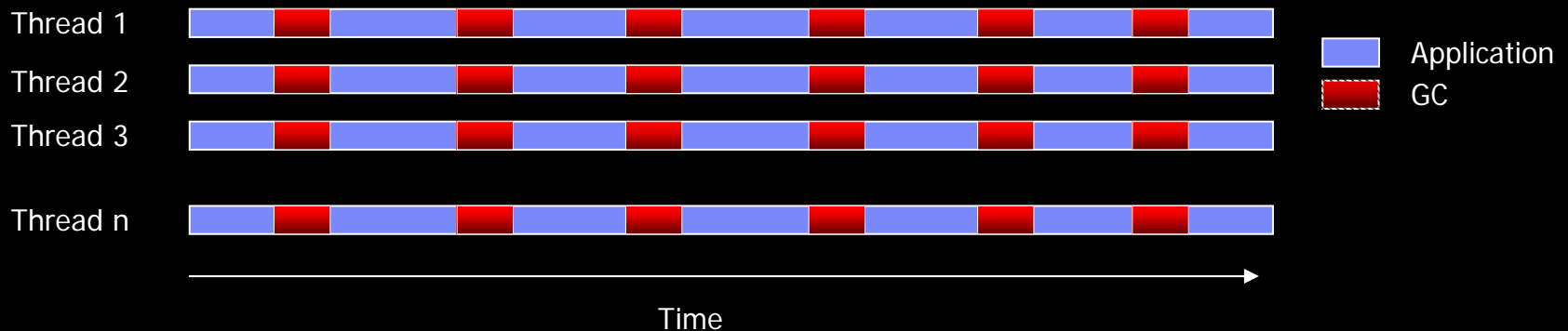
- **-Xgcpolicy:gencon**
 - My application allocates many short-lived objects.
 - The heap space is fragmented.
 - My application is transaction-based
 - Majority of objects in the transaction don't survive beyond the transaction commit.

- **-Xgcpolicy:subpool**
 - I have scalability problems on large multiprocessor machines.

J9 GC: `-Xgcpolicy:optthruput` & `-Xgcpolicy:subpool`

The default policy. Used for applications where raw throughput is more important than short GC pauses. The application is stopped each time that garbage is collected.

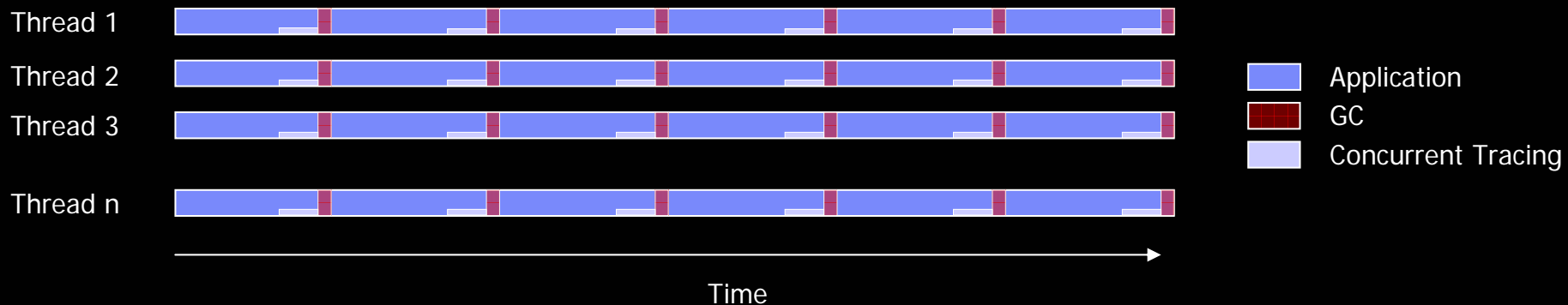
Subpool employs an allocation strategy that is more suitable for multiprocessor machines. We recommend this policy for SMP machines with 16 or more processors.



Picture is only illustrative and doesn't reflect any particular real-life application. The purpose is to show theoretical differences in pause times between GC policies.

J9 GC: `-Xgcpolicy:optavgpause`

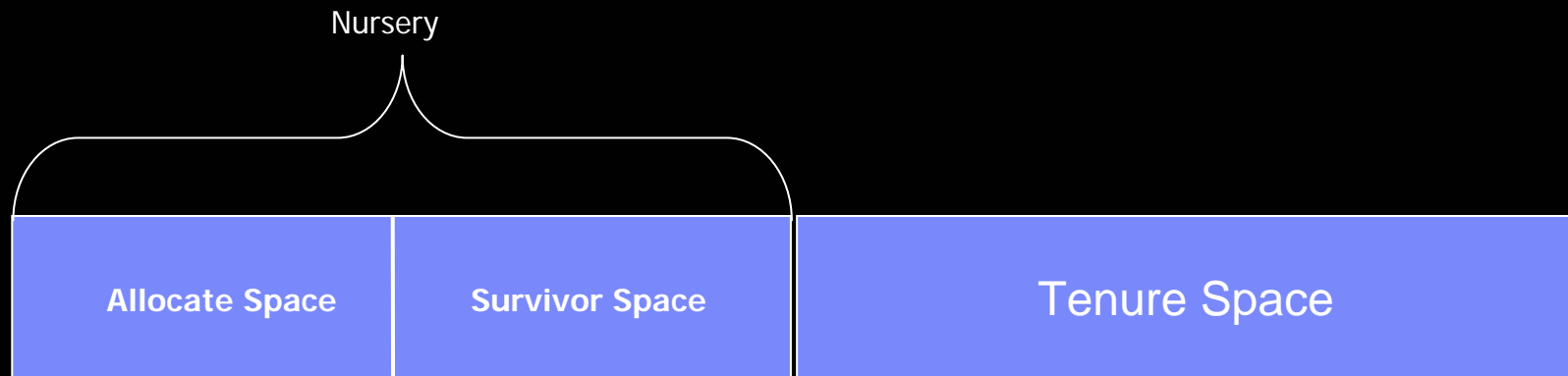
Trades high throughput for shorter GC pauses by performing some of the garbage collection concurrently. The application is paused for shorter periods.



Picture is only illustrative and doesn't reflect any particular real-life application. The purpose is to show theoretical differences in pause times between GC policies.

J9 GC -- A closer look inside –Xgcpolicy:gencon

- **Heap is split into two areas**
 - Objects are created in the *nursery* (a small but frequently collected area)
 - Objects that survive some number of collections are promoted into the *tenured* area (less frequently collected)
- **Nursery is further split into two spaces: 'allocate' and 'survivor'.**
- **A collection in the nursery (scavenge) copies objects from the 'allocate' space to the 'survivor' space.**
- **If an object survive X number of scavenges it gets promoted to the 'tenure' space, where $1 \leq X \leq 14$ and dynamic. (tenuring)**

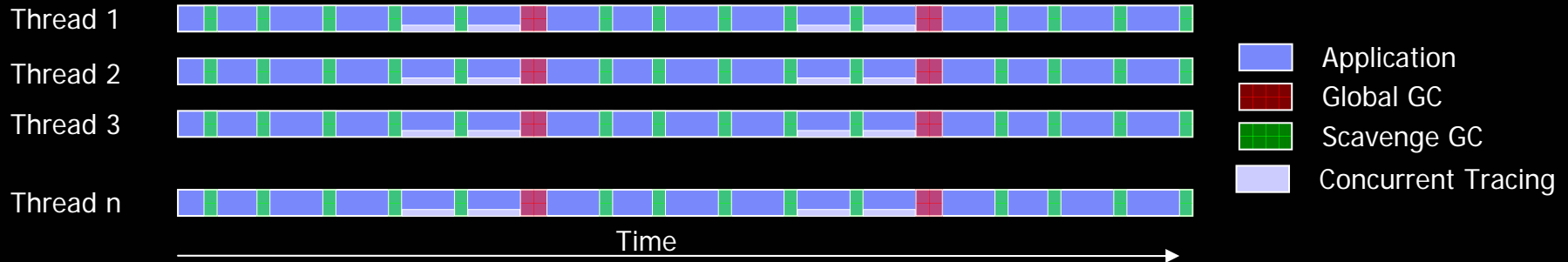


The division between allocate and survivor space is dynamic.

It will be adjusted depending on the survival rate.

J9 GC choices: `-Xgcpolicy:gencon`

Handles short-lived objects differently than objects that are long-lived. Applications that have many short-lived objects can see shorter pause times with this policy while still producing good throughput.



Picture is only illustrative and doesn't reflect any particular real-life application. The purpose is to show theoretical differences in pause times between GC policies.

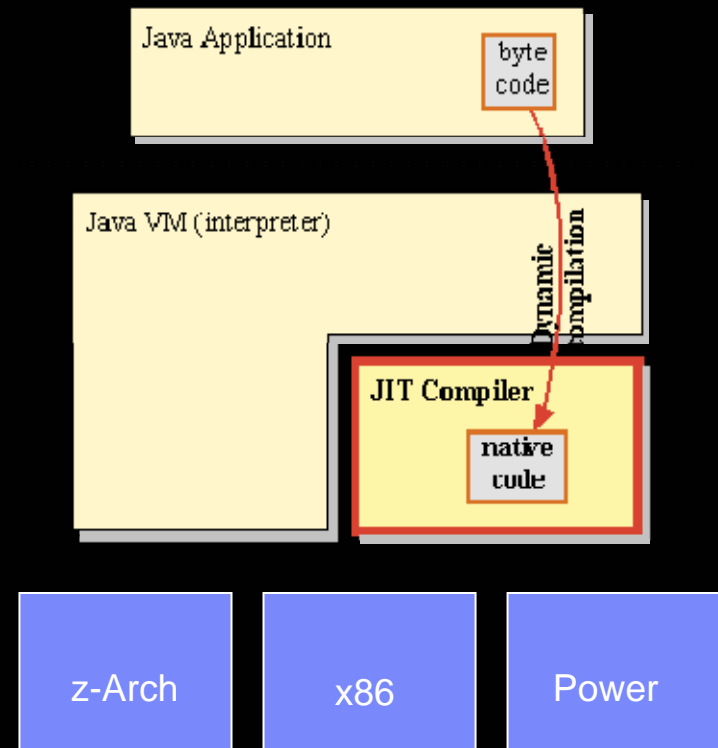
D. IBM TR-JIT Compiler

IBM TR-JIT Compiler – Introduction

- **TR-JIT is IBM's Production JIT on all Platforms for SDK5 and SDK6**
- **Developed at the IBM Toronto Lab**
- **The Toronto Lab has 30+ years of expertise in compilation and optimization technologies**
- **Close relationships with:**
 - Research: productizing innovative ideas and experimental technologies.
(Tokyo/Watson Research Lab)
 - Hardware: best possible performance with the underlying system and processor.
(Poughkeepsie, Austin, xSeries)
 - IBM Middleware: work with DB2® , WAS to provide strong performance
(SVL, Toronto, Raleigh)

IBM TR-JIT Compiler – Introduction

- **Compile byte-code down to native assembly**
- **Overhead of interpretation is removed**
- **Feedback is used to make smart compilation decisions**
- **Significantly more efficient use of computational resource**
 - ~10-100x faster than interpretation
- **Compilation cost is included in application runtime**
 - Choose what to compile
 - How much effort to invest in compilation

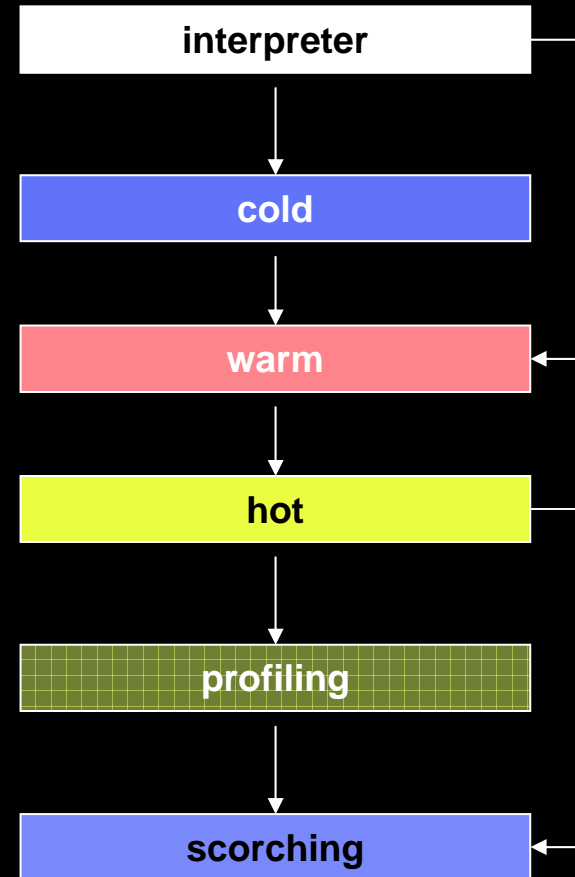


IBM TR-JIT Compiler – Introduction

- Complete suite of classical, OO and Java-specific optimizations
- Highly customizable intermediate language optimizer
- High performance code-generation with deep System z micro-architecture exploitation
- Multiple optimization strategies for different code quality/compile-time tradeoffs
- Fast startup time
- Dynamic recompilation with profile-directed feedback and adaptive optimizations

TR-JIT – Compilation Strategy

- Goals:
 - Focus compilation CPU time where it matters
 - Stager investment over time to amortize cost
- Methods start as interpreted
- After N invocations methods get compiled at 'warm' level
- Sampling thread used to identify hot methods
- Methods may get recompiled at 'hot' or 'scorching' levels
- Transition to 'scorching' goes through a temporary profiling step



E. IBM Java Consumability

What is IBM Support Assistant?

- IBM Support Assistant
 - A free application that simplifies and automates software support
 - Helps customers analyze and resolve questions and problems with IBM software products.
 - Includes rich features and serviceability tools for quick resolution to problems



Find Information

Easily find the information you need including product specific information and search capabilities.



Analyze Problem

Diagnose and analyze problems through serviceability tools, collection of diagnostic artifacts, and guidance through problem determination.



Manage Service Request

Effectively submit, view and manage your service requests enhanced with automated collection of diagnostic data.

IBM Monitoring and Diagnostic Tools for Java™ - Health Center

- Allows running JVMs to be observed and health-checked without perturbing them
- Recommendations and analysis
- Enables insight into general system health, application activity, and garbage collection activity
- Very low overhead
- Now in Beta

IBM Monitoring and Diagnostic Tools for Java™ - Health Center

File Edit Data View Help

Status

- Classes ? Data analysis not available in beta
- [Garbage Collection](#) ! The application seems to be using some quite large objects. The largest request which triggered an allocation failure (and was recorded in the verbose gc log) was for 9766 KB.
- Heap ? Data analysis not available in beta
- I/O ? Data analysis not available in beta
- Just-In-Time ? Data analysis not available in beta
- [Locking](#) ✓ No problems detected.
- [Profilin](#) ✓ Execution time was relatively evenly balanced between methods. No obvious candidates for optimization were found.

2 sources. Idle.

Consumability Tools - Links

- Find the tuning/diagnostic information you need more quickly:

<http://www.ibm.com/developerworks/java/jdk/tools/index.html>

- **IBM Support Assistant**

- URL: <http://www.ibm.com/software/support/isa>

- **Add the JDK documentation**

- Select the “Updater” tab

- Add the “IBM Developer Kit for Java 5.0” (or 6.0) set