Berlin Mainframe Summit

Java on z/OS

Martina Schmidt

© 2006 IBM Corporation

Agenda

- About the mainframe
- Java runtime environments under z/OS
- For which applications should I use a mainframe?
- Java on z/OS cost and performance
- Java Development for z/OS
- Summary and literature

Agenda

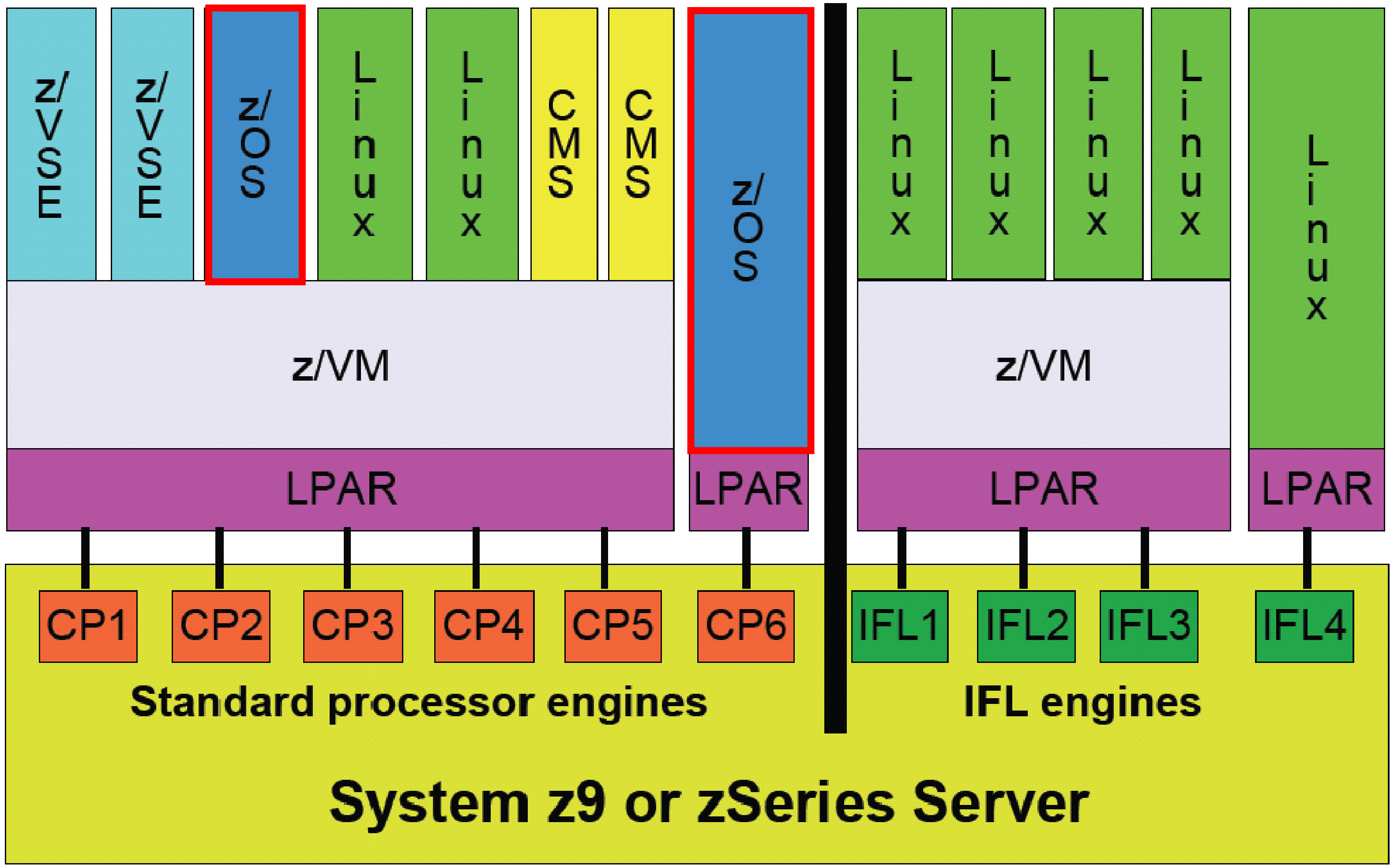
- About the mainframe
- Java runtime environments under z/OS
- For which applications should I use a mainframe?
- Java on z/OS cost and performance
- Java Development for z/OS
- Summary and literature

© 2006 IBM Corporation

System z9 – The IBM Mainframe

- **40 years of evolution**
- **Enterprise server for** highest availability
- **High efficient** workload management
- **Capacity on Demand Offerings**
- **Outstanding** security **concept:**
  - Key management
  - Encryption
  - Data integrity
- Virtualisation **engine**
  - LPAR concept
  - z/VM

System z9 – Operating systems



Agenda

- About the mainframe
- Java runtime environments under z/OS
- For which applications should I use a mainframe?
- Java on z/OS cost and performance
- Java Development for z/OS
- Summary and literature

© 2006 IBM Corporation

JVM 5.0 – A complete new JVM for z/OS

- **Sun IP-free, but Java 2 (1.3) compliant (J2ME) and J2SE (1.4.2, 5.0)**

- Common code base **across all platforms**
  - PowerPC, IA32, x86-64, and 390 (Linux or **z/OS**)

- **Flexible and sophisticated technology oriented to:**
  - Performance (throughput and application startup time)
  - Scalability
  - Reliability, Availability and Servicability (RAS)

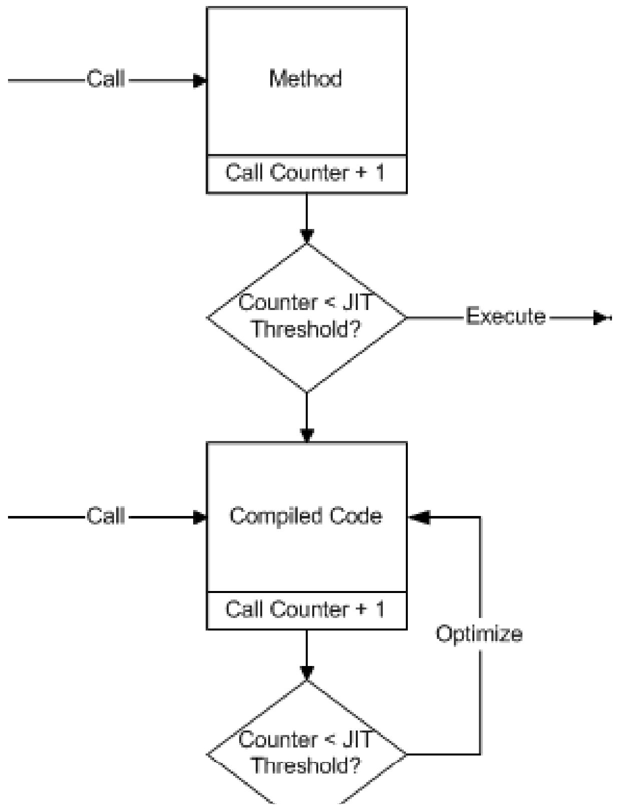
JVM 5.0 – Just-in-time compiler

- **The just-in-time compiler (JIT) is not really part of the JVM, but is essential for a** high performing Java application
- **Java is Write Once Run Anywhere thus it is interpreted by nature and without the JIT could not compete with native code applications**
- **As your code accesses methods the** JIT determines how frequently specific methods are accessed **and compiles those touched often quickly to optimize performance**
- -Xquickstart **helps to improve JVM startup time for short running Java applications**
  - causes the JIT to run with a subset of optimizations

Just in time compiler

- uses a call counter for every call to a method
- when threshold is exceeded the method will be compiled
- also calls to compiled methods are counted
- when threshold exceeds again method will be recompiled with stronger optimization rules
- happens every time the threshold is reached

  → most code is optimized at the highest level.
- some methods will never be compiled



Performance of short-running applications

- JIT is tuned for long-running applications typically used on a server
- `-Xquickstart` is used for improving startup time of Java applications
- causes the JIT to run with a subset of optimizations; that is, a quick compile
- appropriate for short-running applications, especially those where execution time is not concentrated into a small number of methods
- can degrade performance if it is used on longer-running applications that contain hot methods
- also try adjusting the JIT threshold (using trial and error) for short-running applications to improve performance

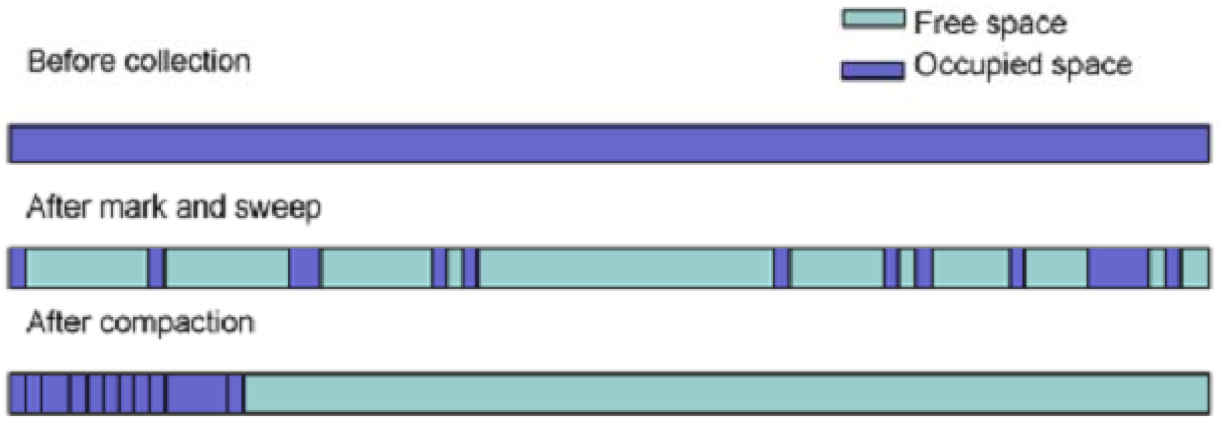
JVM 5.0 – Garbage collection

Memory management is configurable using four different policies with varying characteristics

1. **Optimize for Throughput** – flat heap collector focused on maximum throughput
   - -Xgcpolicy:optthruput
2. **Optimize for Pause Time** – flat heap collector with concurrent mark and sweep to minimize GC pause time
   - -Xgcpolicy:optavgpause
3. **Generational Concurrent** – divides heap into “nursery” and “tenured” segments providing fast collection for short lived objects. Can provide maximum throughput with minimal pause times
   - -Xgcpolicy:gencon
4. **Subpool** – a flat heap technique to help increase performance on multiprocessor systems , commonly greater than 8. Available on IBM pSeries™ and zSeries™
   - -Xgcpolicy:subpool

Garbage collection

- Garbage collector = memory manager for Java heap
- Phases of garbage collection:
  - Mark
  - Sweep
  - Compaction (optional)

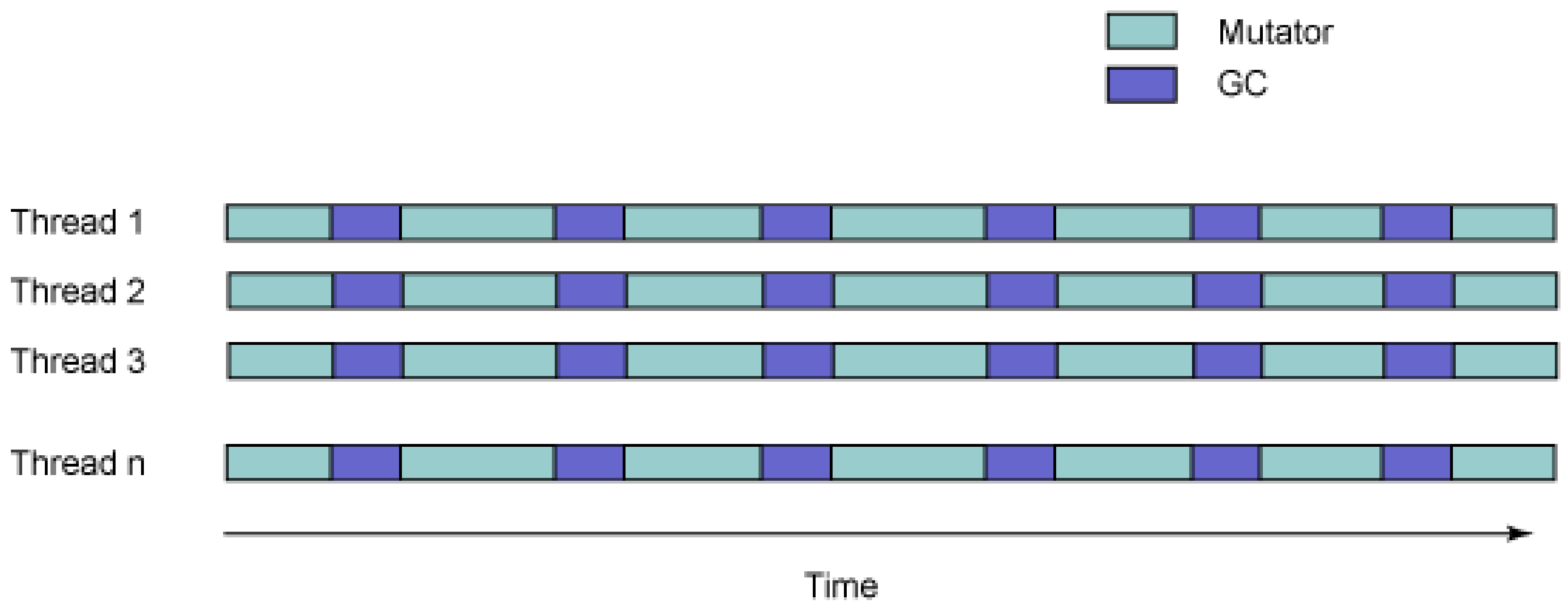


Which policy in which situation?

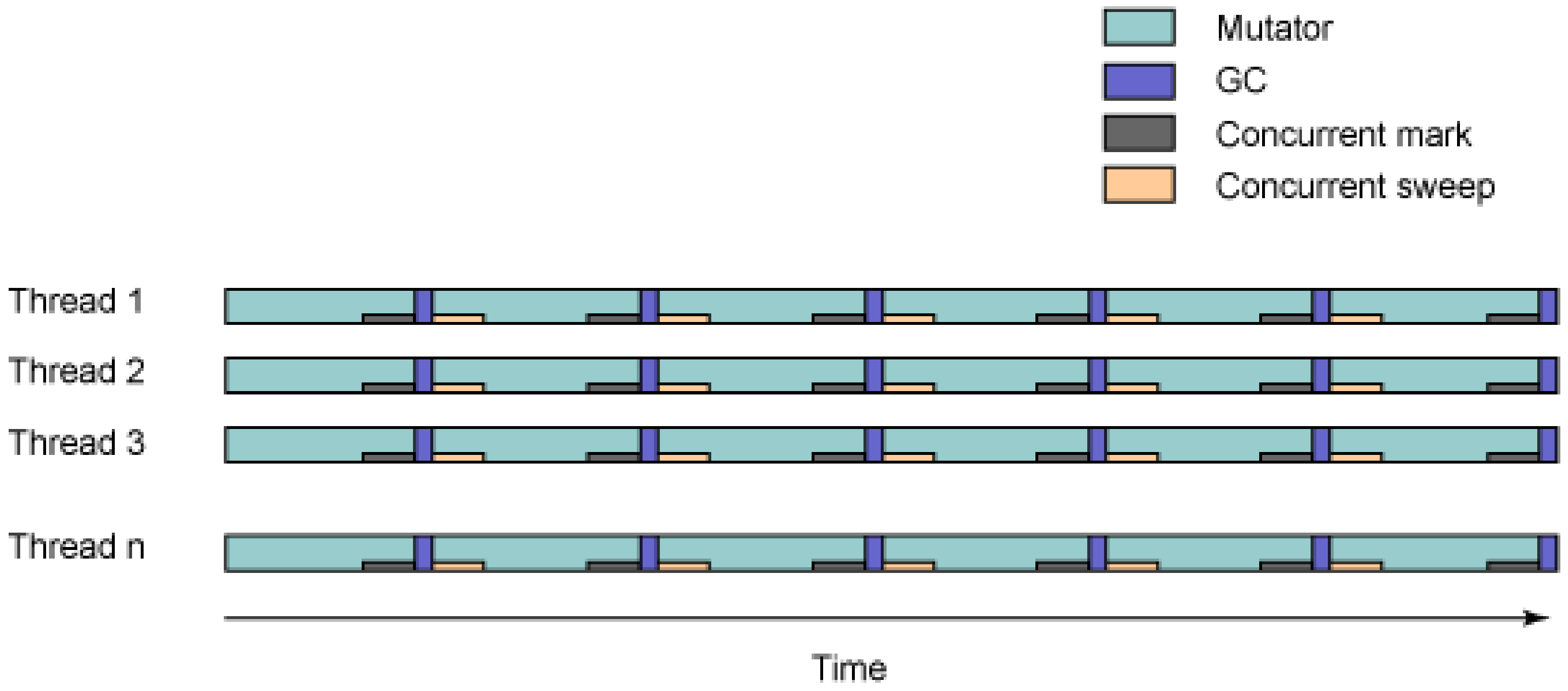
- **ALWAYS start with the default policy!**
- `optavgpause`
  - application cannot tolerate length of GC pauses (degradation in performance is acceptable)
  - 64-bit platform and very large heap -- more than 3 or 4GB
  - GUI application and concerns about the user response times
- `gencon`
  - Allocation of many short-lived objects
  - fragmented heap space
  - transaction-based application (objects in the transaction don't survive beyond the transaction commit)
- `subpool`
  - scalability problems on large multiprocessor machines

Optthruput

Distribution of CPU time between application threads (mutators) and GC threads in the optthruput policy



Optavgpause



Distribution of CPU time between mutators and GC threads in the optavgpause policy

Concurrent mark and sweep:

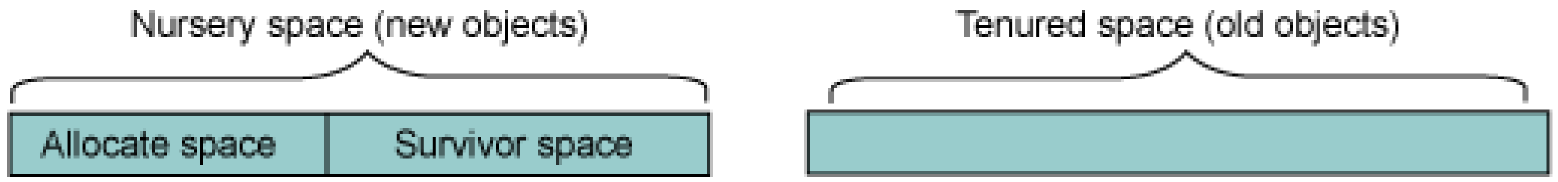
each mutator helps out and marks objects before heap is filled up

(concurrent mark)

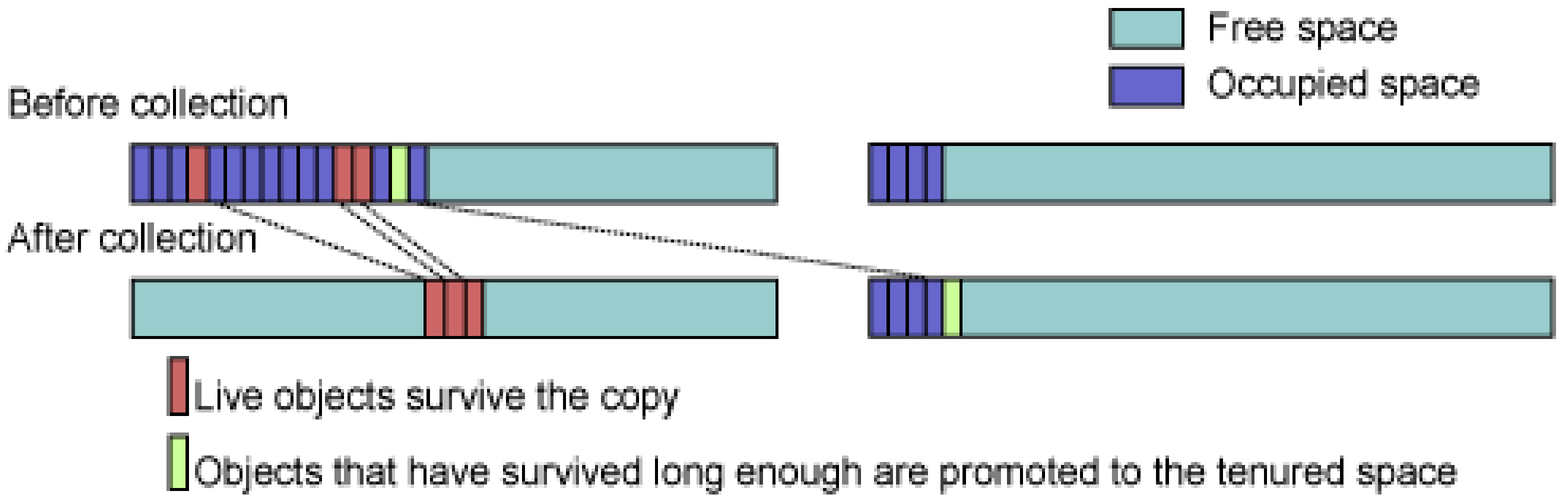
still a stop-the-world GC, but pause significantly shorter

after GC, mutator threads help out and sweep objects (concurrent sweep)

Gencon

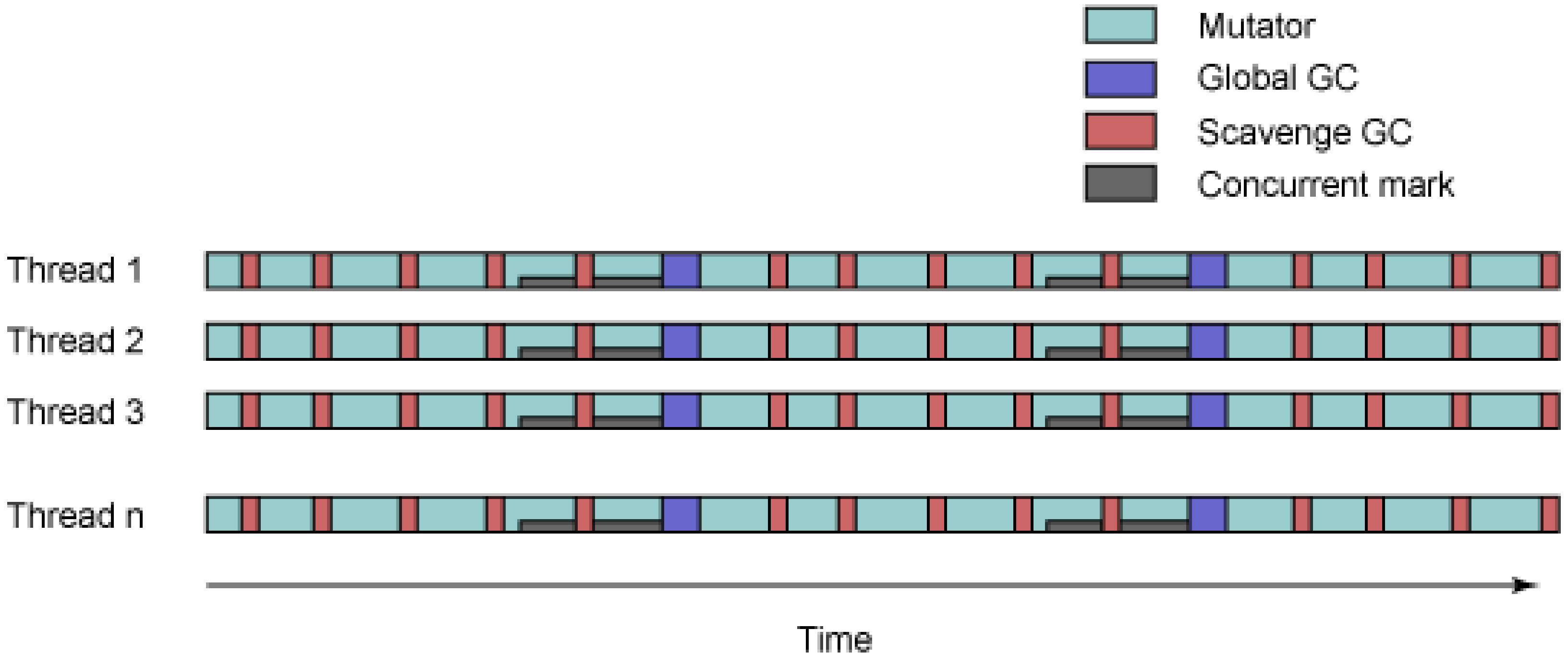


New and old area in gencon GC


Free space
Occupied space
Before collection
After collection
Live objects survive the copy
Objects that have survived long enough are promoted to the tenured space

Example of heap layout before and after GC

Gencon



Distribution of CPU time between mutators and GC threads in gencon

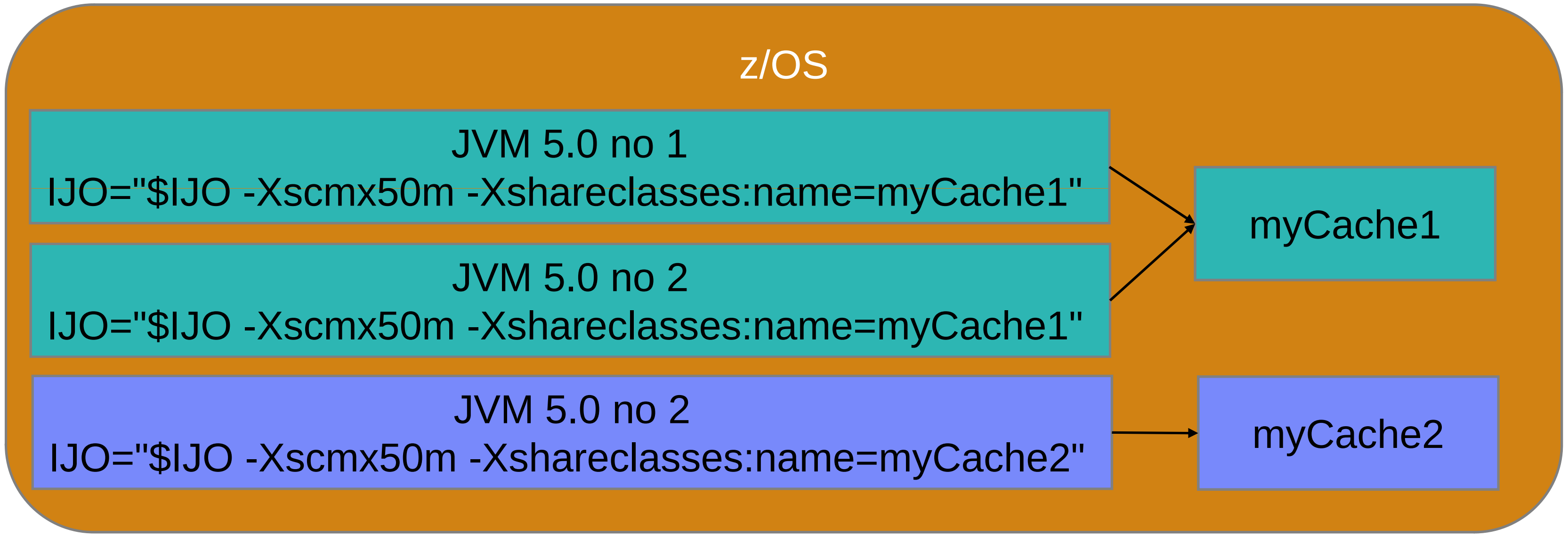
Subpool

- Like default policy
- uses improved object allocation algorithm to achieve better performance when allocating objects on the heap

© 2006 IBM Corporation

JVM 5.0 – Shared classes

- **A shared class area for one or more JVMs**
- Improves startup **time**
  - – Lots of classes are already preloaded
- **One class cache for many JVMs**


z/OS
JVM 5.0 no 1
IJO="$IJO -Xscmx50m -Xshareclasses:name=myCache1"
JVM 5.0 no 2
IJO="$IJO -Xscmx50m -Xshareclasses:name=myCache1"
JVM 5.0 no 2
IJO="$IJO -Xscmx50m -Xshareclasses:name=myCache2"
myCache1
myCache2

Shared classes

- **offers a completely transparent and dynamic means of sharing all loaded classes, both application classes and system classes, and placing no restrictions on JVMs that are sharing the class data**
- **switch on shared classes with the -Xshareclasses command-line option**
- **obvious benefits:**
  - virtual memory footprint reduction when using more than one JVM
  - loading classes from a populated cache is faster than loading classes from disk → improved startup
  - classes are already in memory and partially verified
  - benefits applications that regularly start new JVM instances doing similar tasks
  - cost to populate an empty cache with a single JVM is minimal

Shared classes – key points

- Classes are stored in a named "class cache" - area of shared memory of fixed size, allocated by the first JVM that needs to use it
- any JVM can read from or update the cache
- a JVM can connect to only one cache at a time
- cache persists beyond the lifetime of any JVM connected to it, until it is explicitly destroyed or operating system is shut down
- for class loading JVM looks first in the cache
- if it does not find class, it loads class from disk and adds it to cache
- when a cache is full, classes can still be shared, but no new classes can be added

Shared classes – key points

- if changes are made to classes on the file system, classes in the cache might become out of date
- updated version of the class is detected by the next JVM that loads it and the class cache is updated
- sharing of modified byte code at runtime is supported, but must be used with care
- access to the class cache is protected by Java Permissions if a security manager is installed
- resources, objects, JIT'd code, and class data that changes cannot be stored in the cache

© 2006 IBM Corporation

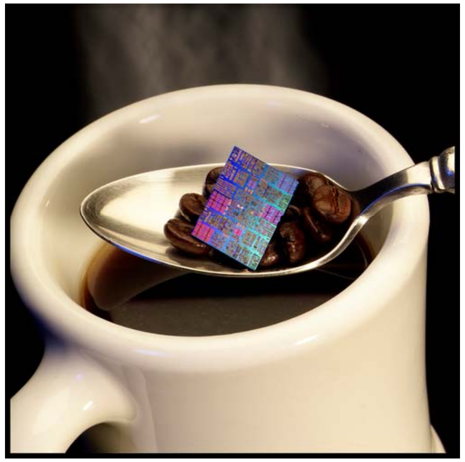
Agenda

- About the mainframe
- Java runtime environments under z/OS
- For which applications should I use a mainframe?
- Java on z/OS cost and performance
- Java Development for z/OS
- Summary and literature

© 2006 IBM Corporation

For which Java applications does a mainframe fit very well?

- **Batch is still one of the mainframes biggest strengths**
  - The mainframe was designed for batch (punch cards)
  - The mainframe has the longest experience in the batch environment
  - Special facilities in z/OS allow a huge complex job management for
    batch jobs (JES, SDSF,...)
  - **Java inherits these functionalities**



- **Business critical Java based servers that need:**
  - High avalibilty (99,999%)
  - Best security
- **Java applications which use lots of transactions**
  - Data proximity

Agenda

- About the mainframe
- Java runtime environments under z/OS
- For which applications should I use a mainframe?
- Java on z/OS cost and performance
- Java Development for z/OS
- Summary and literature

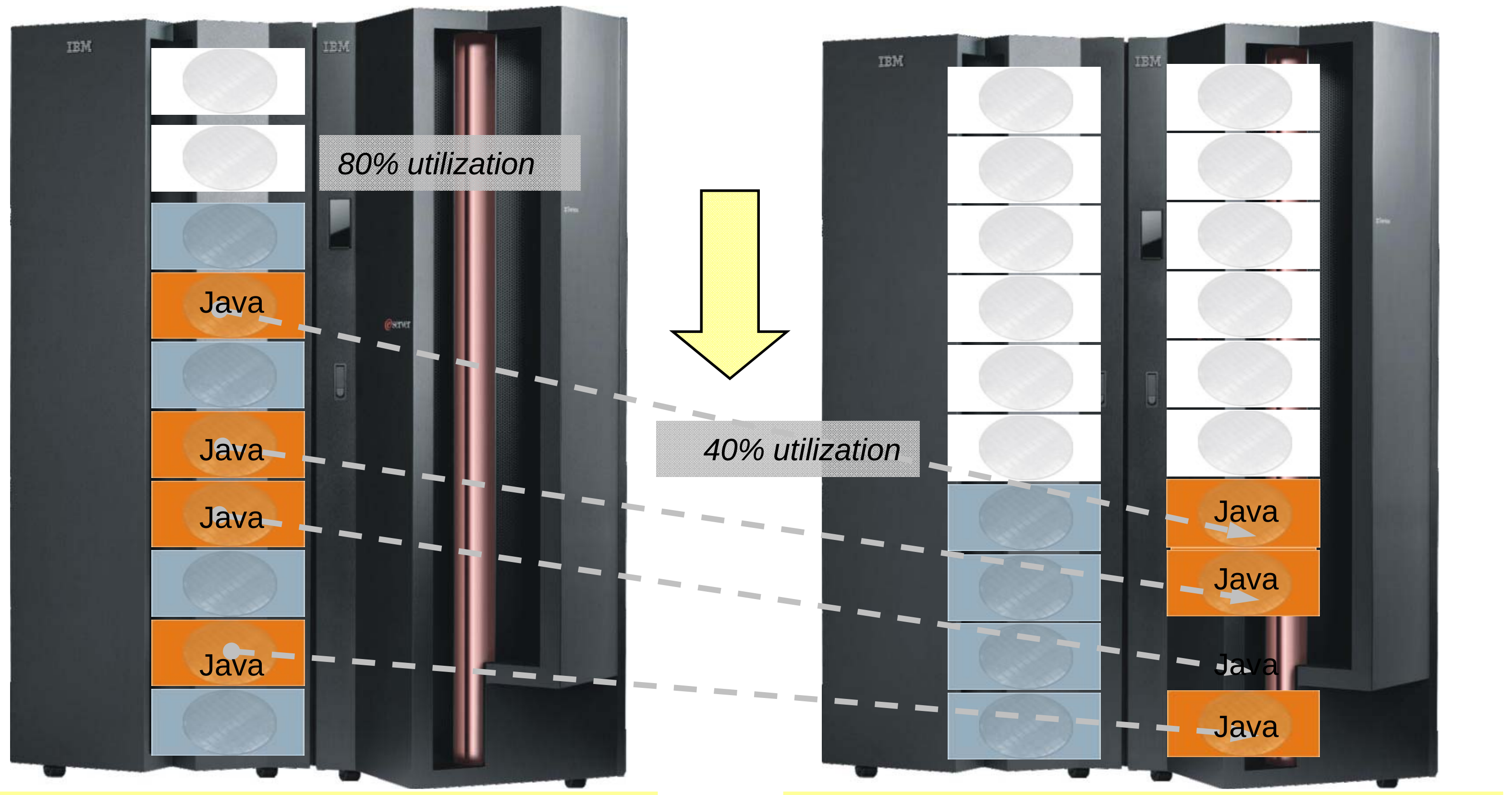
© 2006 IBM Corporation

zAAP: Java Workload Engine on z/OS

- **zAAP (zSeries Application Assist Processor)**
- **New processor type on z890, z990, z9-109 hardware supporting z/OS**

- **A specialized z/OS Java execution environment for Java-based applications**
  - WAS V5.1
  - CICS/TS V2.3
  - DB2 V7 and V8, IMS V8
  - WebSphere WBI for z/OS
- **Require z/OS V1R6 and SDK 1.4**
- **Usage projection**
  - z/OS V1.6 RMF report to collect the ‘Would Have Been” zAAP usage

System z Application Assist Processor: zAAP

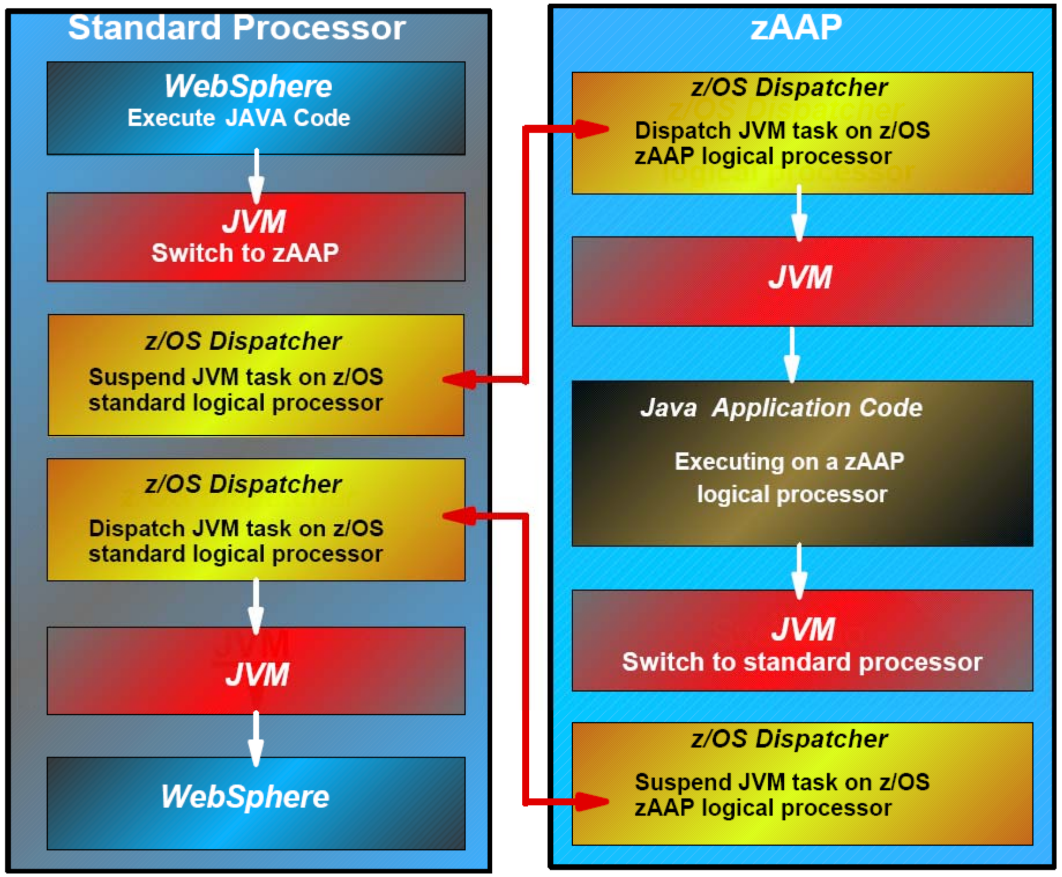


1000 MIPS for JAVA App

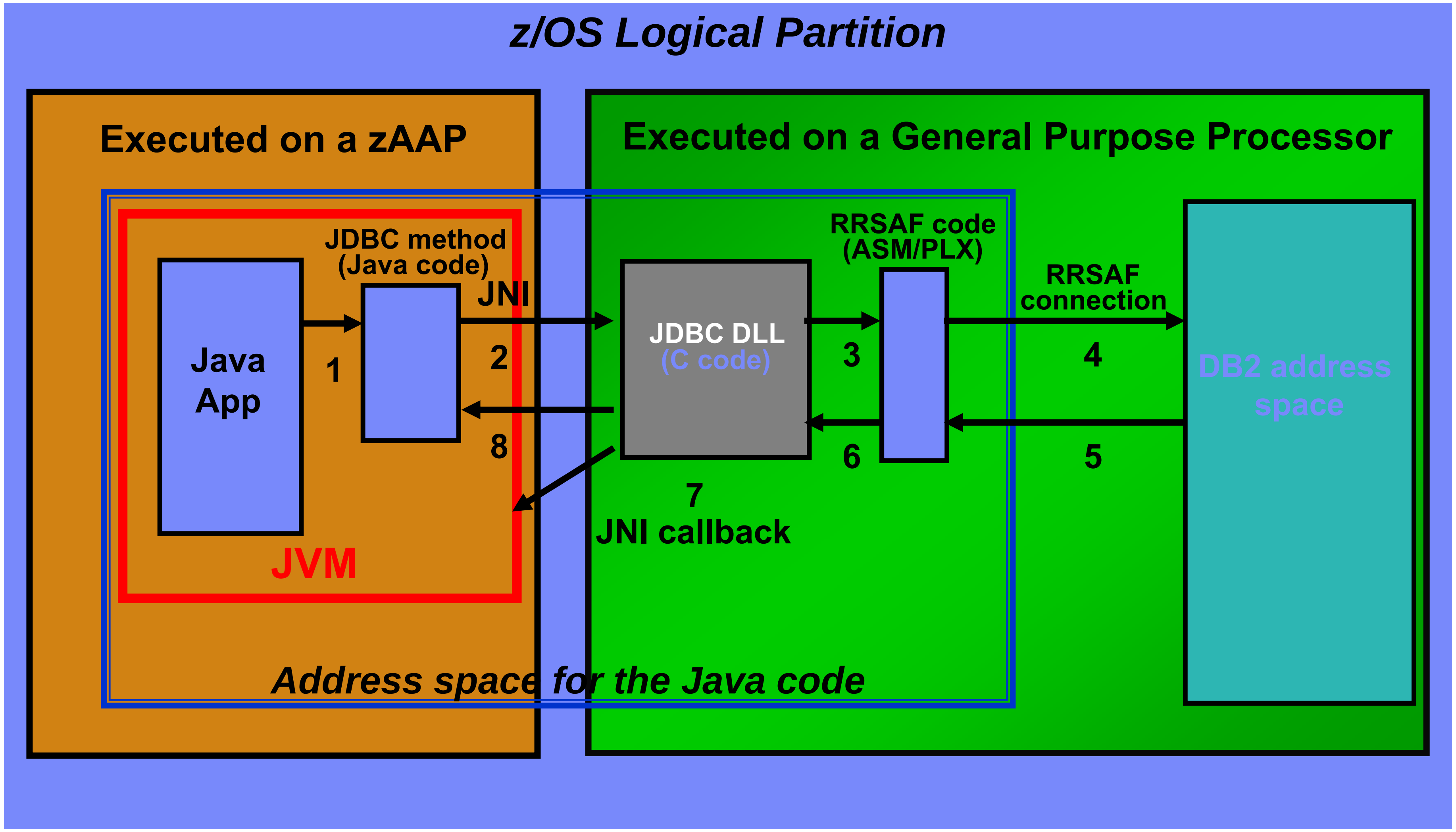
500 MIPS for Java App +
500 MIPS now available for additional workloads

© 2006 IBM Corporation

System z application assist processor: zAAP

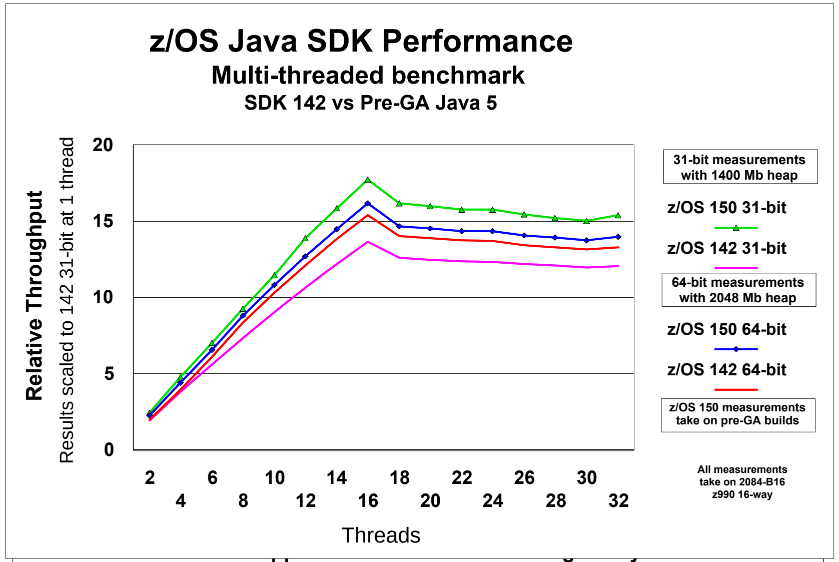

Standard Processor
WebSphere
Execute JAVA Code
JVM
Switch to zAAP
z/OS Dispatcher
Suspend JVM task on z/OS standard logical processor
z/OS Dispatcher
Dispatch JVM task on z/OS standard logical processor
JVM
WebSphere
zAAP
z/OS Dispatcher
Dispatch JVM task on z/OS zAAP logical processor
JVM
Java Application Code
Executing on a zAAP logical processor
JVM
Switch to standard processor
z/OS Dispatcher
Suspend JVM task on z/OS zAAP logical processor

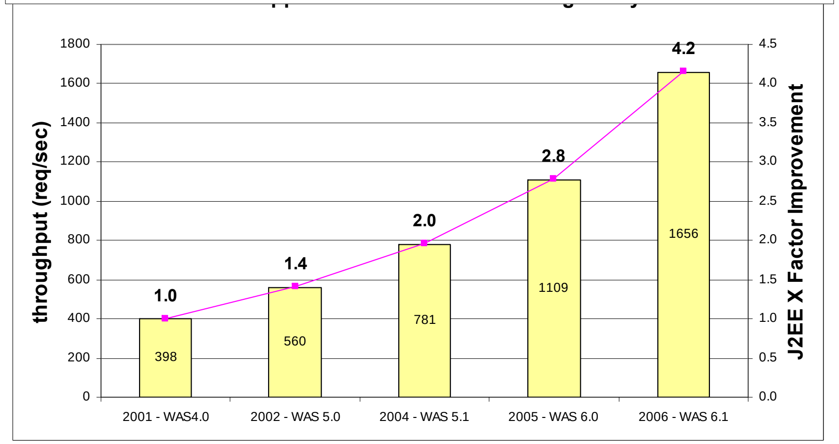
zAAP Integration at Work: Java App calling DB2



Performance

- Java is getting better and better
- Almost everything is fast enough for batch
- Java on z/OS is measurable
- Cost of Java Batch is the question to discuss



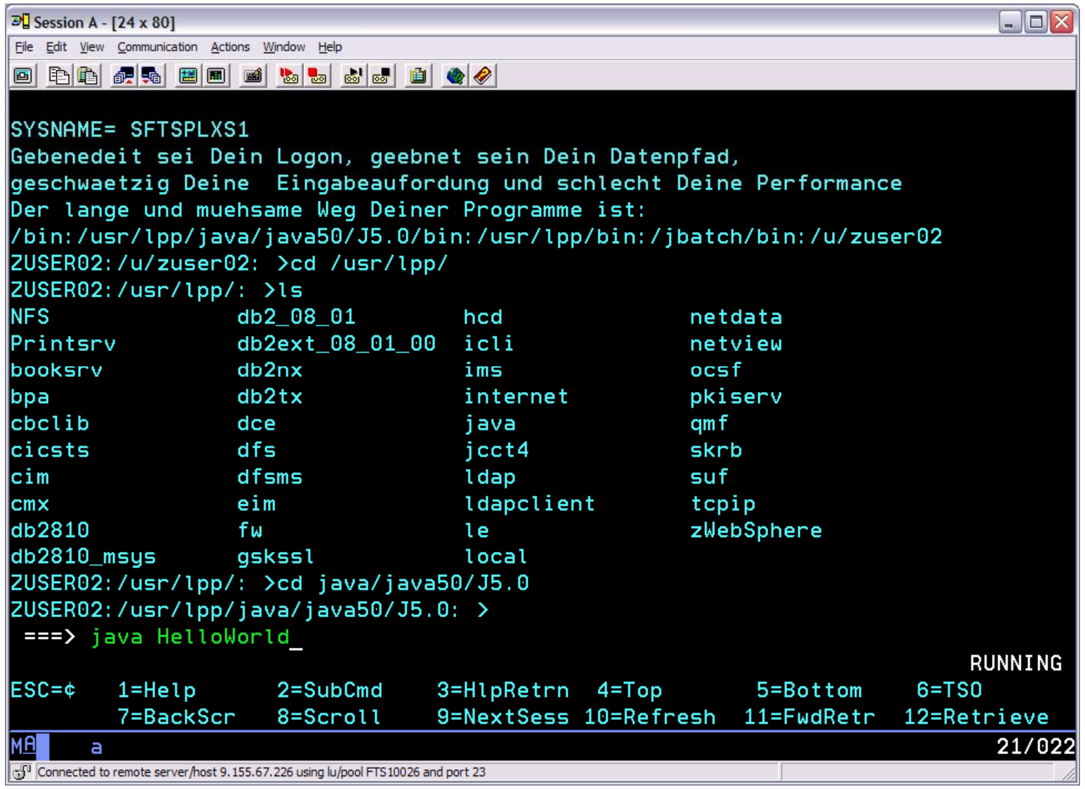


Agenda

- **About the mainframe**
- **Java runtime environments under z/OS**
- **For which applications should I use a mainframe?**
- **Java on z/OS cost and performance**
- **Java Development for z/OS**
- **Summary and literature**

How to develop Java applications for the mainframe

- **This is a picture people often associate with the mainframe:**



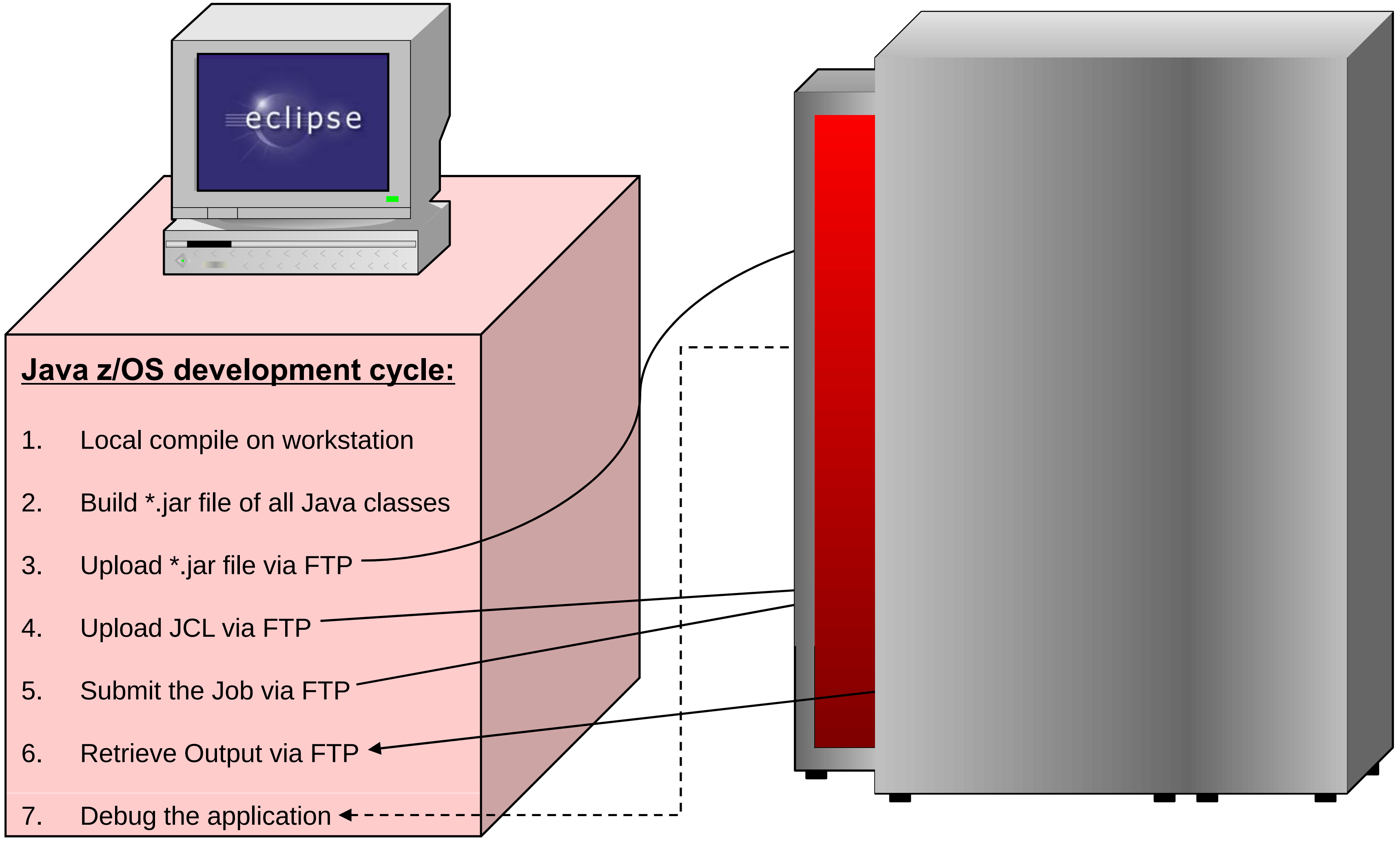
Session A - [24 x 80]

```
SYSNAME= SFTSPLXS1
Gebenedeit sei Dein Logon, geebnet sein Dein Datenpfad,
geschwaetzig Deine  Eingabeaufordung und schlecht Deine Performance
Der lange und muehsame Weg Deiner Programme ist:
/bin:/usr/lpp/java/java50/J5.0/bin:/usr/lpp/bin:/jbatch/bin:/u/zuser02
ZUSER02:/u/zuser02: >cd /usr/lpp/
ZUSER02:/usr/lpp/: >ls
NFS                db2_08_01        hcd              netdata
Printsrv           db2ext_08_01_00  icli             netview
booksrv            db2nx            ims              ocsf
bpa                db2tx            internet         pkiserv
cbclib             dce              java             qmf
cicsts             dfs              jcct4            skrb
cim                dfsms            ldap             suf
cmx                eim              ldapclient       tcpip
db2810             fw               le               zWebSphere
db2810_msys        gskssl           local
ZUSER02:/usr/lpp/: >cd java/java50/J5.0
ZUSER02:/usr/lpp/java/java50/J5.0: >
 ===> java HelloWorld_
                                                                RUNNING
ESC=¢    1=Help      2=SubCmd    3=HlpRetrn  4=Top       5=Bottom    6=TSO
         7=BackScr   8=Scroll    9=NextSess 10=Refresh  11=FwdRetr  12=Retrieve
MA      a                                                             21/022
```

Connected to remote server/host 9.155.67.226 using lu/pool FTS10026 and port 23

- **... But it is much easier!**
  - Eclipse as an IDE can be easily used for Mainframe Java development

Development tools for Java Batch: 1) Eclipse



Java z/OS development cycle:

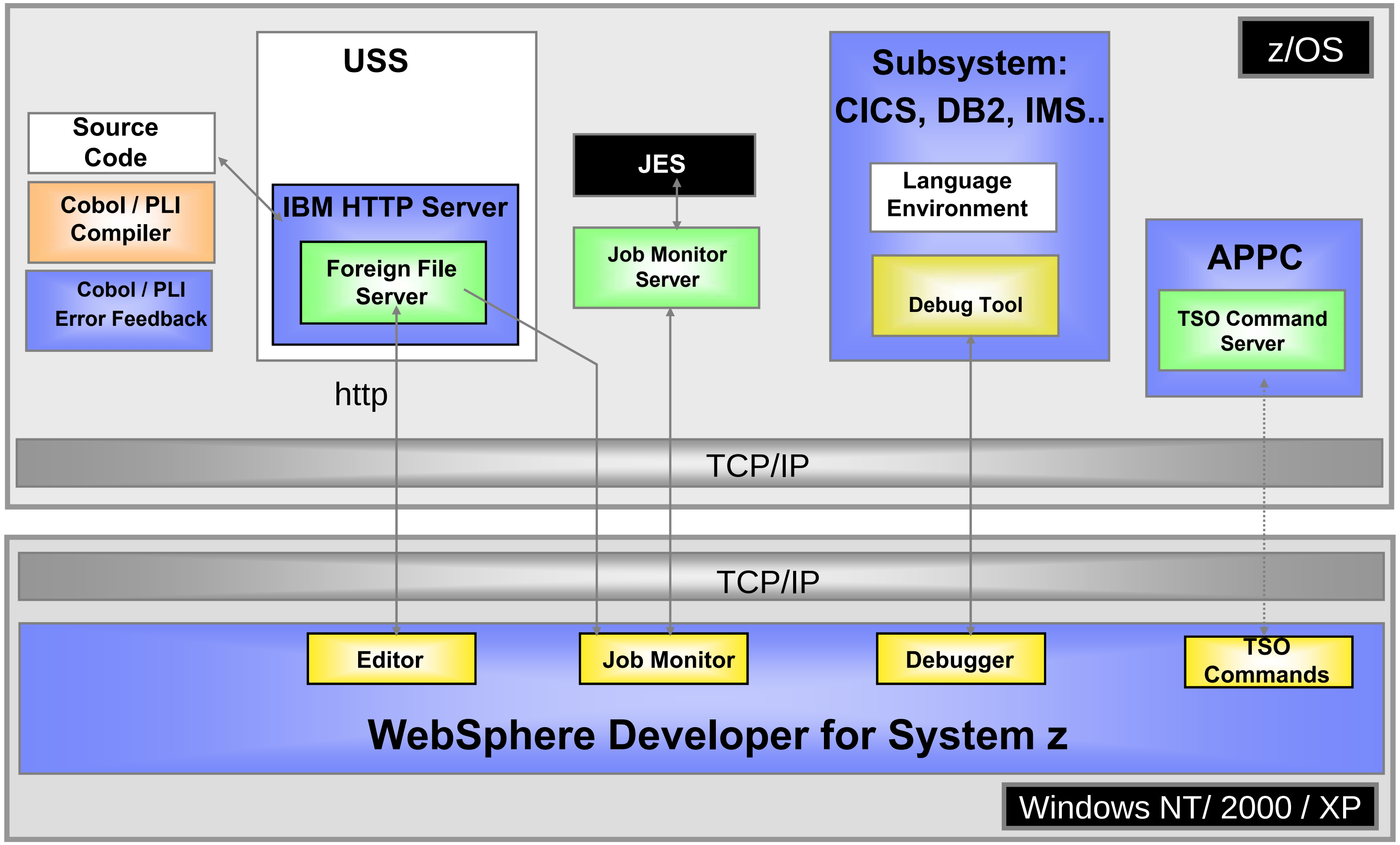
1. Local compile on workstation
2. Build *.jar file of all Java classes
3. Upload *.jar file via FTP
4. Upload JCL via FTP
5. Submit the Job via FTP
6. Retrieve Output via FTP
7. Debug the application

Development tools for Java Batch: 2) WDz

- **Based on Eclipse**
- **Inherits Eclipse functions plus:**
  - **JES integration**
  - J2EE and Web programming
  - UML modeling
  - **Cobol, PL/I and C/C++ development**
  - CICS Web Service Support
  - Web Service Test Client Generation
  - IBM Debug Tool integration
  - ...

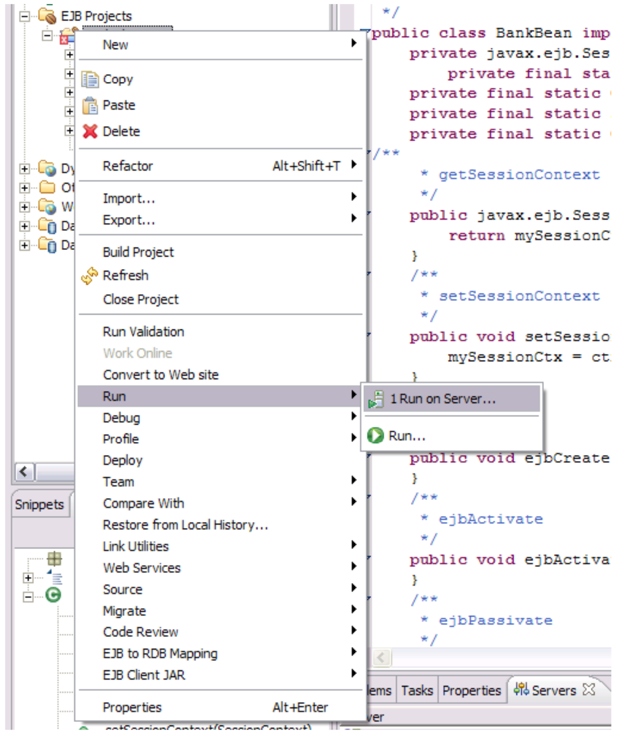
© 2006 IBM Corporation

Development tools for Java Batch: 2) WDz



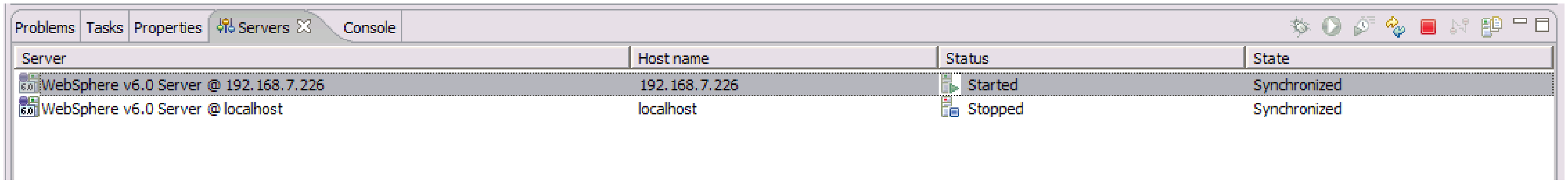
WDz and J2EE development

- **WDz is put on top of Rational Application Developer**
- Integrated **WebSphere Application** Server test environment
- Remote deployment **of applications**
- **Wizards for EJB creation**
- **EJB Test client**



```
 */
public class BankBean imp
    private javax.ejb.Ses
        private final sta
    private final static
    private final static
    private final static
/**
     * getSessionContext
     */
    public javax.ejb.Sess
        return mySessionC
    }
    /**
     * setSessionContext
     */
    public void setSessio
        mySessionCtx = ct
    }

    public void ejbCreate
    }
    /**
     * ejbActivate
     */
    public void ejbActiva
    }
    /**
     * ejbPassivate
     */
```



Server	Host name	Status	State
WebSphere v6.0 Server @ 192.168.7.226	192.168.7.226	Started	Synchronized
WebSphere v6.0 Server @ localhost	localhost	Stopped	Synchronized

WDz and J2EE / Web development

- **Web Development:**
  - JSF support
  - Web site designer
  - JSP and servlet wizard
  - Java Visual Editor for JSF Component Layout
  - Portlet development
- **Web Service development**
  - Web Services Wizard
  - Web Services Explorer
  - Generation of Cobol Web Services

© 2006 IBM Corporation

Resource access and backend integration

- **System resources like the security facility RACF can be accessed via easy Java APIs**
- **The mainframe provides special, well approved** transaction monitors **(CICS and IMS) that can easily be integrated into Java applications**
- Local connectors**, that use cross-memory functions allow** high transactional performance computing
  - **Example: Local JDBC Driver for DB2 z/OS**

Agenda

- About the mainframe
- Java runtime environments under z/OS
- For which applications should I use a mainframe?
- Java on z/OS cost and performance
- Java Development for z/OS
- Summary and literature

Summary

- **The mainframe is the** ultimate server for enterprise and mission critical applications

- **There are more or less** no differences in the development of Java applications between distributed servers and mainframes

© 2006 IBM Corporation

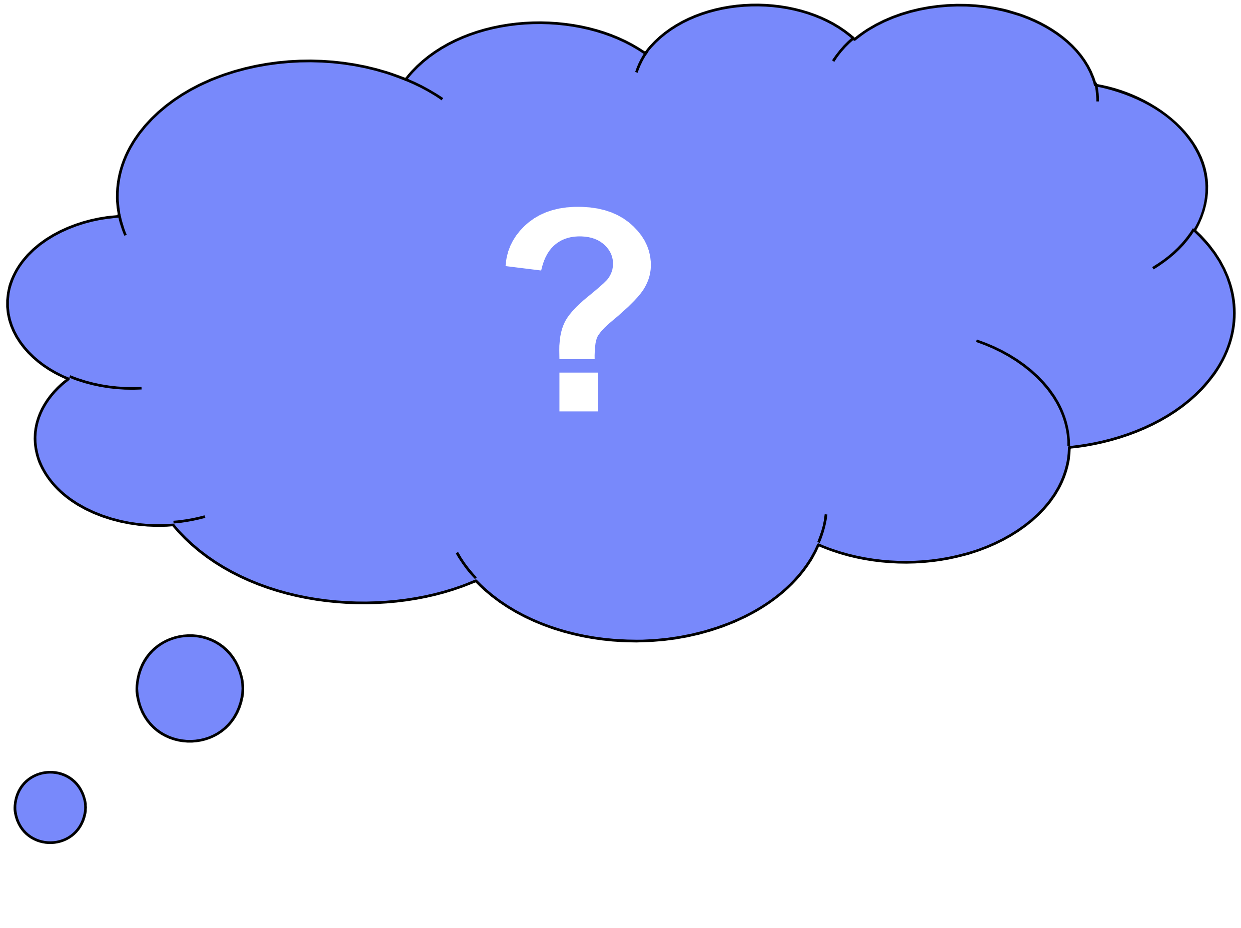
Literature

- **IBM Redbooks:**
  - Java Stand-alone Applications on z/OS, Volume I

    http://www.redbooks.ibm.com/abstracts/sg247177.html
  - Java Stand-alone Applications on z/OS, Volume II

    http://www.redbooks.ibm.com/abstracts/sg247291.html?Open

Questions



© 2006 IBM Corporation