An Introduction to CICS JVMServers

Ian J Mitchell
IBM Distinguished Engineer, CICS Transaction Server

1st March 2011
Session 8265

Topics

- Evolution of the JVM in CICS Transaction Server
- How are JVM Servers different to JVM Pools
- Defining JVM Server resources
- How applications run in JVM Servers
- The future for JVMs in CICS TS

Notes

- **Information regarding potential future products is intended to outline our general product direction and it should not be relied on in making a purchasing decision. The information mentioned regarding potential future products is not a commitment, promise, or legal obligation to deliver any material, code or functionality. Information about potential future products may not be incorporated into any contract. The development, release, and timing of any future features or functionality described for our products remains at our sole discretion.**

- Unless specifically advised otherwise, you should assume that all information presented in Inner Circle sessions and contained in these presentations is IBM Confidential and restrict access to this information in accordance with the WebSphere Inner Circle Participation Agreement.

- The workshops, sessions and materials have been prepared by IBM or the session speakers and reflect their own views. They are provided for informational purposes only, and are neither intended to, nor shall have the effect of being, legal or other guidance or advice to any participant. While efforts were made to verify the completeness and accuracy of the information contained in this presentation, it is provided AS-IS without warranty of any kind, express or implied. IBM shall not be responsible for any damages arising out of the use of, or otherwise related to, this presentation or any other materials. Nothing contained in this presentation is intended to, nor shall have the effect of, creating any warranties or representations from IBM or its suppliers or licensors, or altering the terms and conditions of the applicable license agreement governing the use of IBM software.

- References in this presentation to IBM products, programs, or services do not imply that they will be available in all countries in which IBM operates.

- Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput or performance that any user will experience will vary depending upon many factors, including 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 results similar to those stated here.

- All customer examples described are presented as illustrations of how those customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics may vary by customer. Nothing contained in these materials is intended to, nor shall have the effect of, stating or implying that any activities undertaken by you will result in any specific sales, revenue growth or other results.

Topics

- Evolution of the JVM in CICS Transaction Server
- How are JVM Servers different to JVM Pools
- Defining JVM Server resources
- How applications run in JVM Servers
- The future for JVMs in CICS TS

CICS and the z/OS JVM

- **JVM implementation in CICS continues to evolve**

– JDK 1.1.8	CICS TS V1.3
– HPJ and Hotpooling	“ “ “
– IBM Persistent Reusable JVM (Shiraz) JDK 1.4.2	CICS TS v2.x
– zAAP *(from 2004 and z/OS 1.6)*	
– Continuous mode	CICS TS v3.1
– Java 5 *(via PTF)*	CICS TS v3.2
– Java 6	CICS TS v4.1

Topics

- Evolution of the JVM in CICS Transaction Server
- How are JVM Servers different to JVM Pools
- Defining JVM Server resources
- How applications run in JVM Servers
- The future for JVMs in CICS TS

Differences between pools and servers

- **CICS JVM pools**
  - A JVM pool is a set of single task, serially reuseable JVMs
  - Large memory footprint
  - Excellent isolation characteristics

- **CICS JVM Servers**
  - Multiple CICS tasks execute as threads in a JVM concurrently
  - Much larger capacity for concurrent workload
  - Some risk of collateral damage
  - **Exploited for Dynamic Scripting FeaturePack**
  - Limited customer application use in v4.1
  - Will be primary execution model in vNext
    - Pooled model retained for migration only

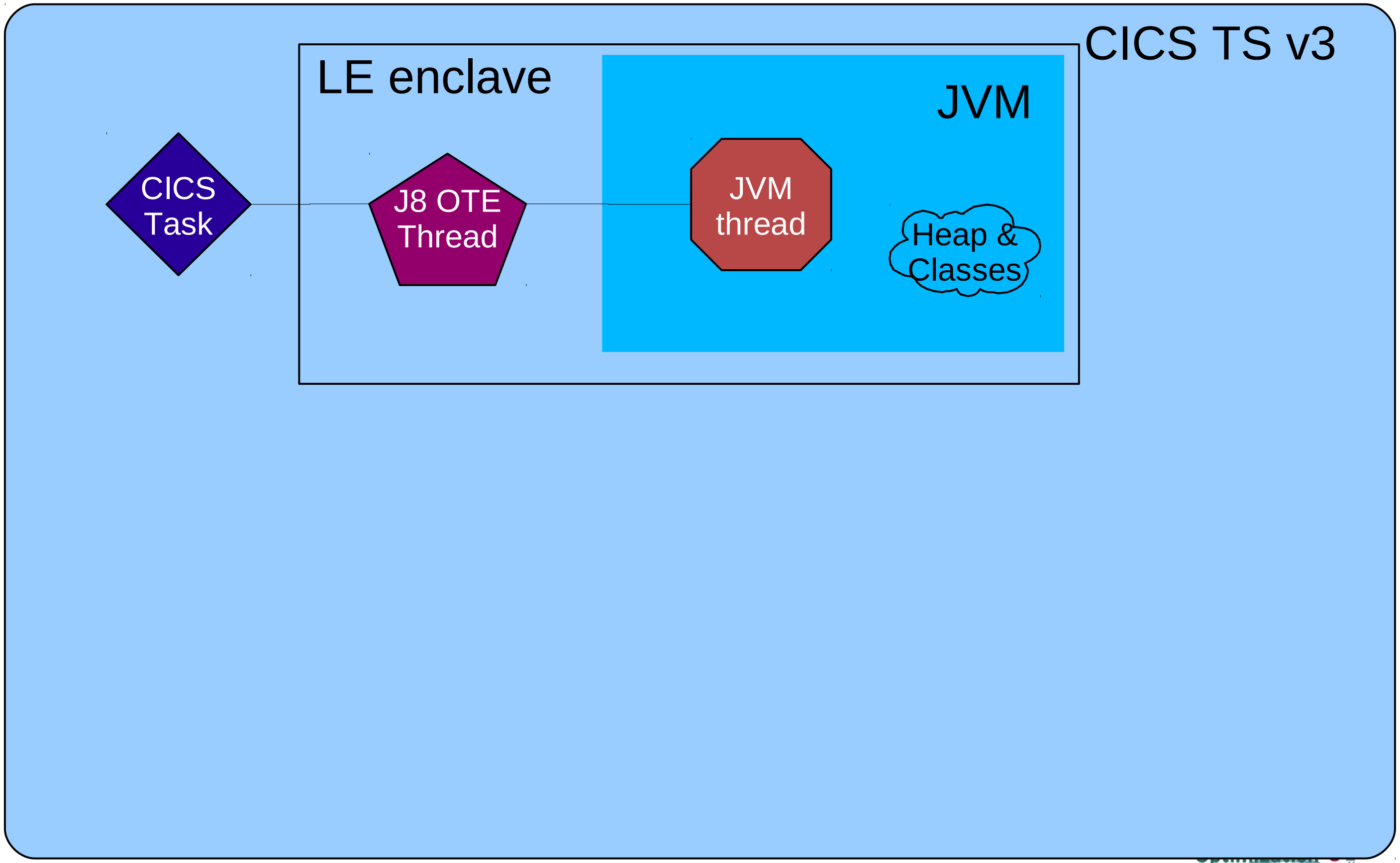
JVM Pool Architecture - CICS TS v3 (and v2)

A CICS task needing to execute Java application code must acquire a JVM from the pool.

A JVM in the pool is managed by an OTE (J8) TCB.

When a task needs a JVM is switches to an appropriate J8 TCB owning a JVM.

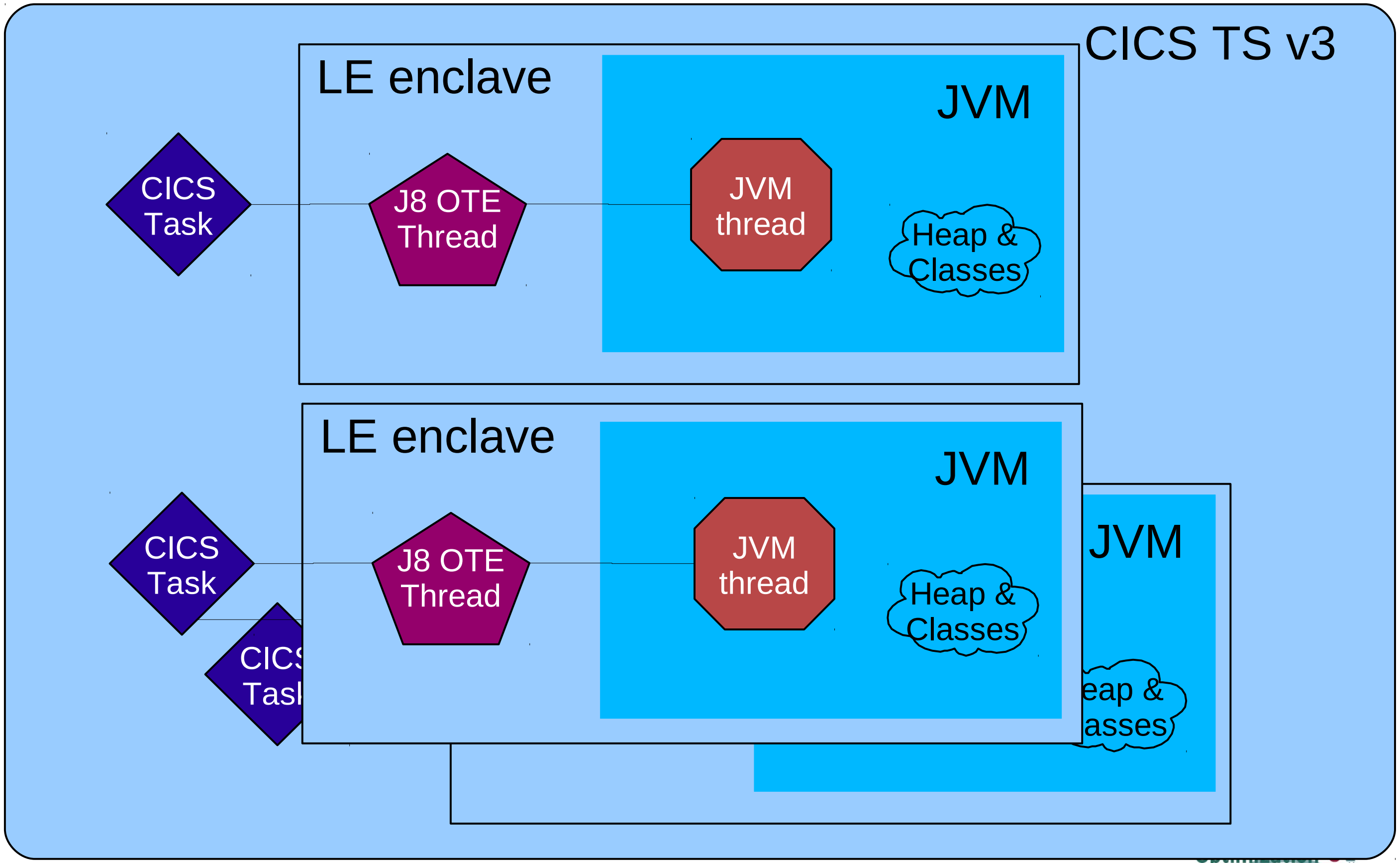
The J8 and JVM is allocated to the task until the completes.



JVM Pool Architecture - CICS TS v3 (and v2)

So concurrent task count limited to the number of JVMs in the pool.

That is limited by the number that can fit in the 31-bit address space.

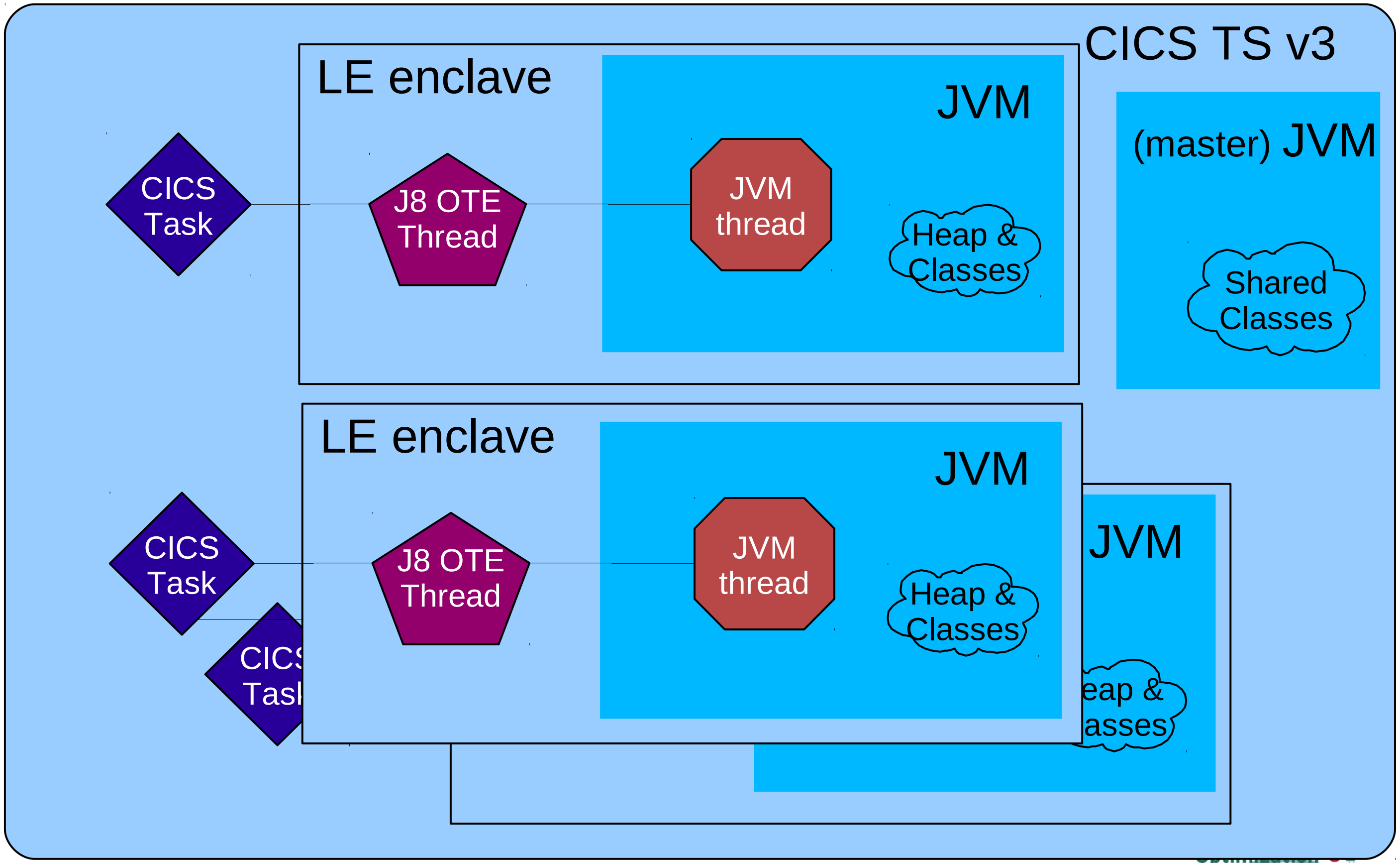


JVM Pool Architecture - CICS TS v3 (and v2)

Each JVM 'costs' approx. 20Mb plus the application heap value.

Bytecode of loaded classes may be shared between JVMs using the classcache.

Result is about 20 task/JVMs concurrently in each region.



JVM Server Architecture

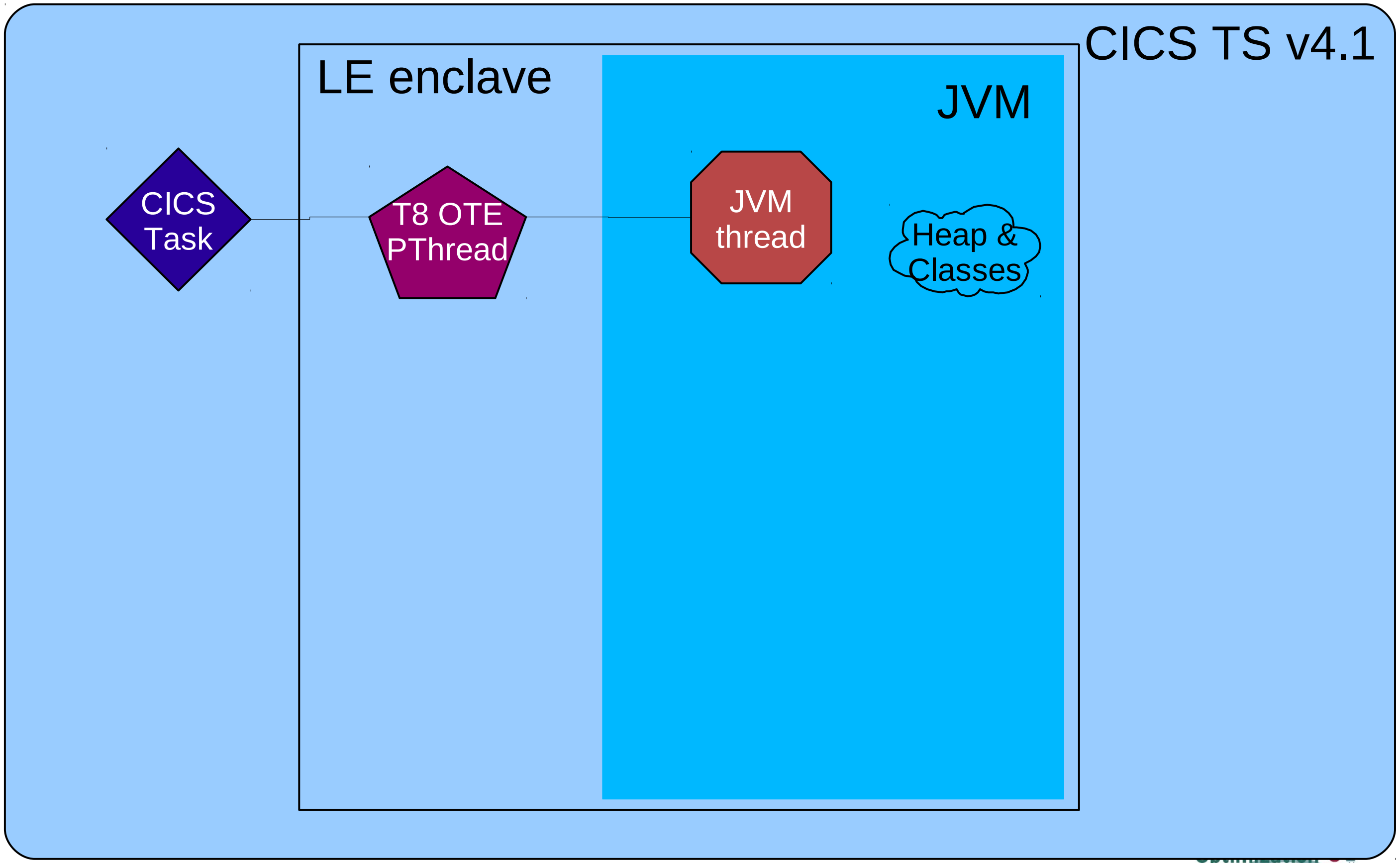
In a JVMServer,
a task will still use an OTE TCB and
a thread in the JVM.

The OTE TCB is a new "T8" mode rather than a J8.

It is also an LE "pthread".

A Java Native Interface (JNI) call is used to make the CICS Task/T8/LE pthread combo into a Java thread.

JNI mechanism is called "AttachCurrentThread".



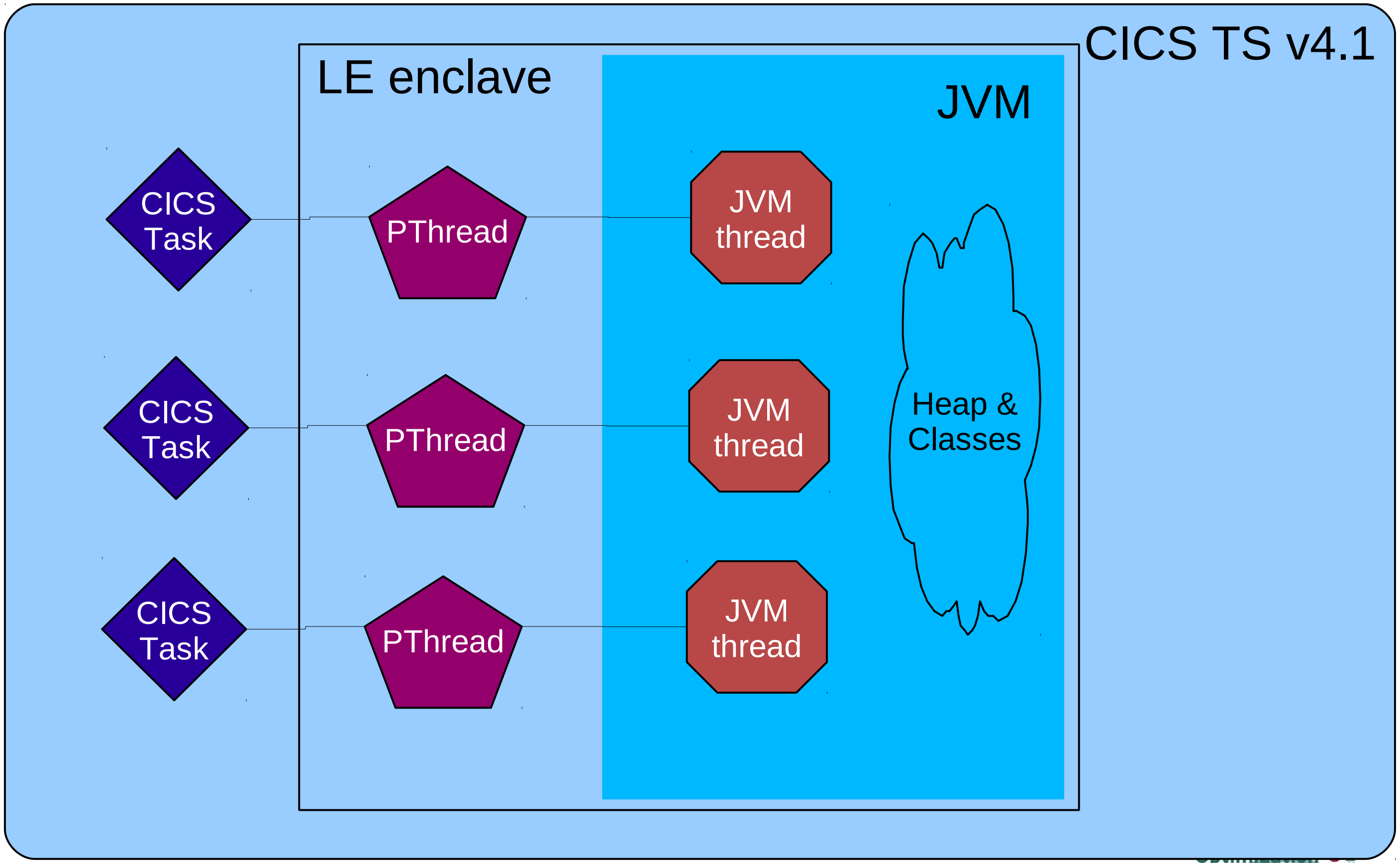
JVMServer Architecture

The difference is that other CICS tasks are attached to the same JVM at the same time.

Therefore the system serves **more requests** using a single JVM.

JVMServer thread memory “cost” is very small.

Result is **hundreds of tasks** concurrently per region.

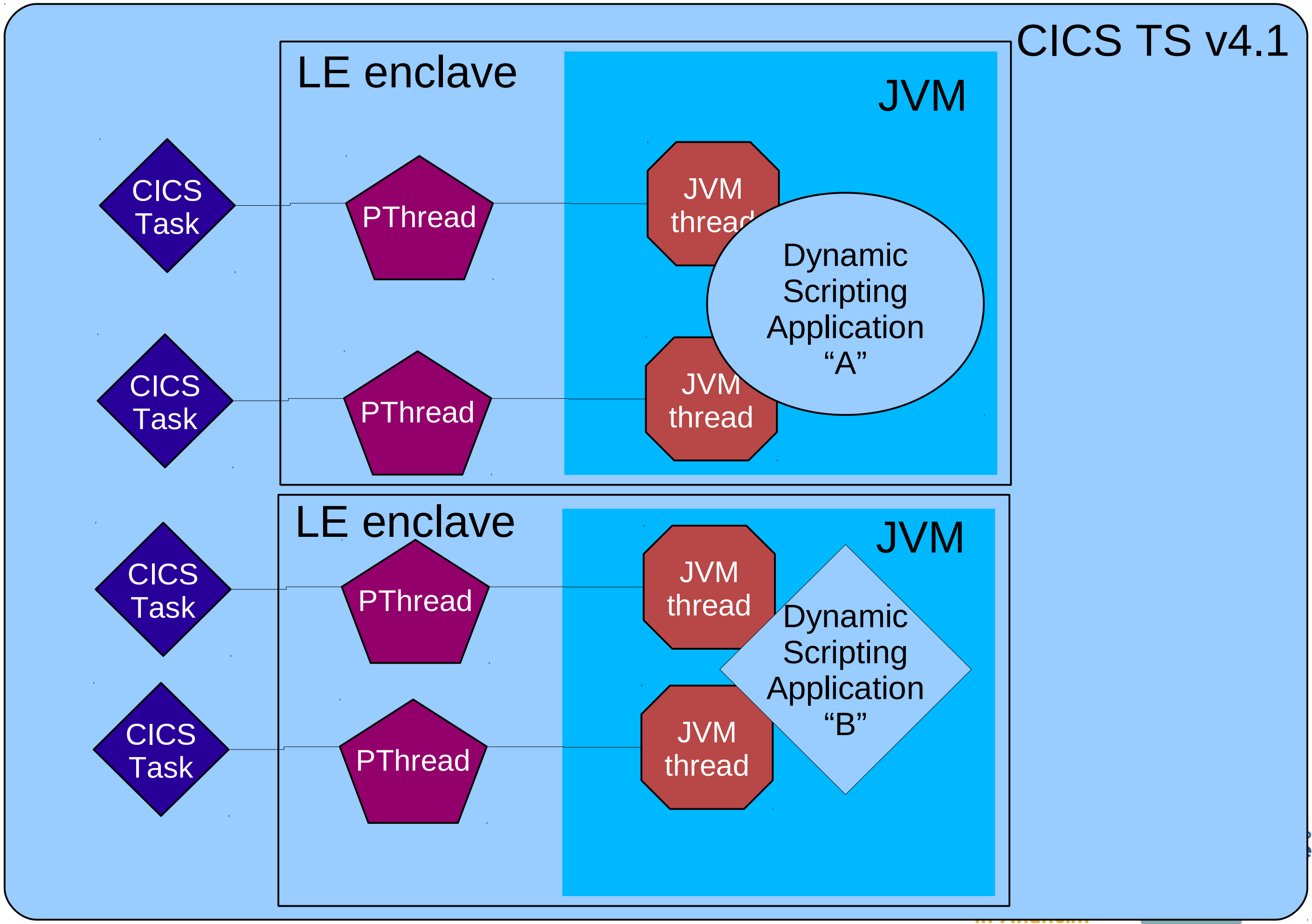


JVMServer Architecture

Architected to allow multiple JVMServers in a single CICS.

Different types of work, or just a degree of isolation.

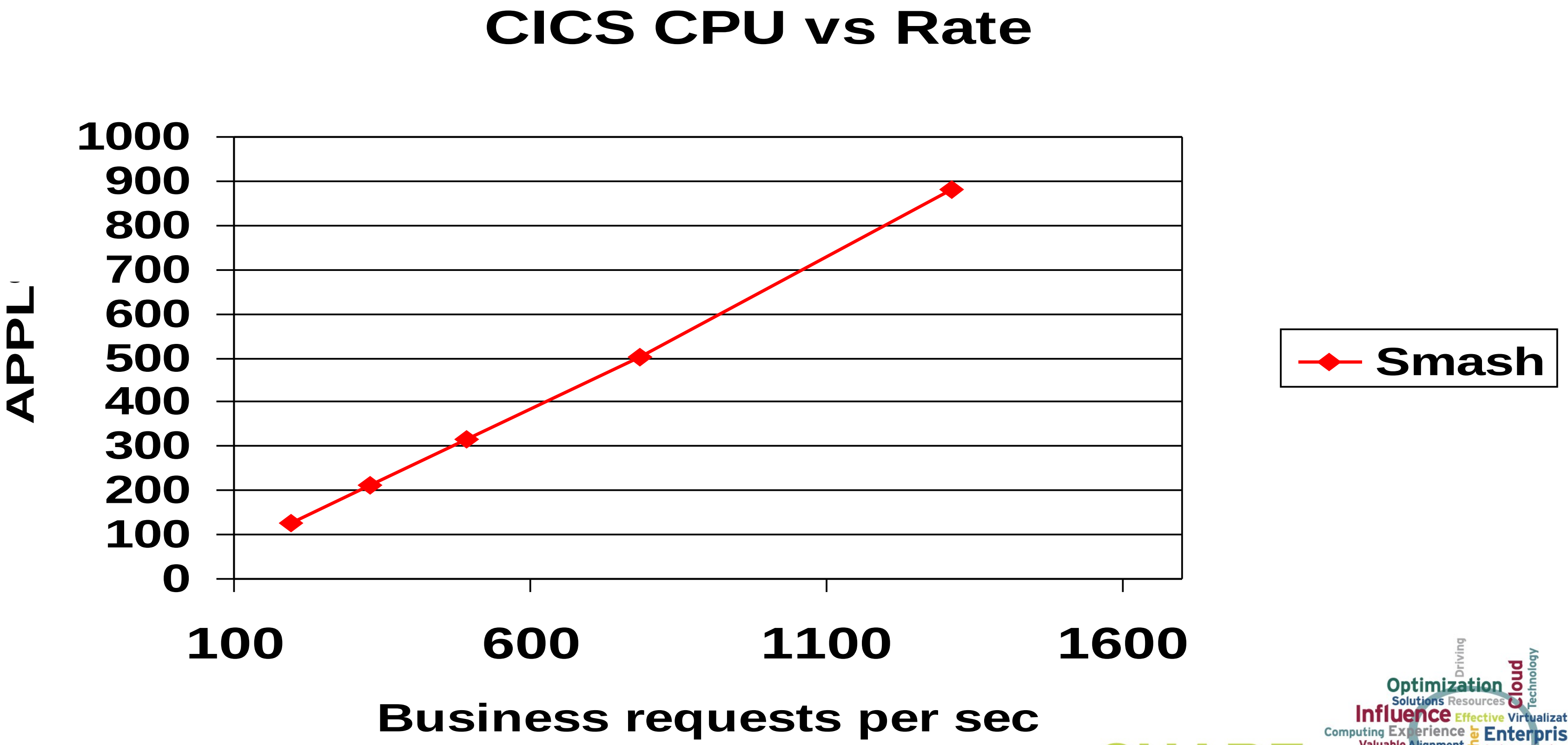
Typically the same limit of 20 JVMServers exists as for JVMs in a pool.



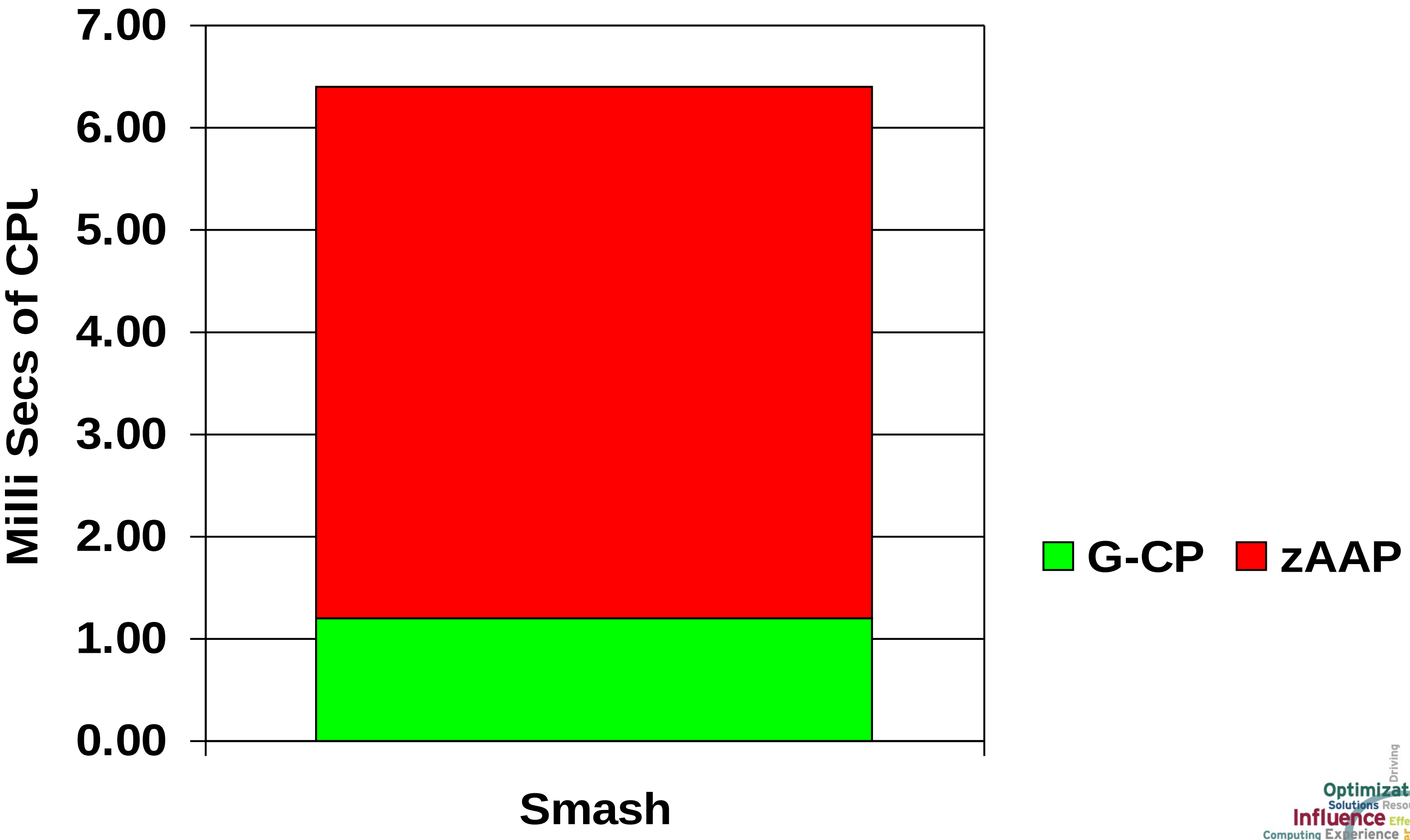
So what can JVMServers do?
Dynamic Scripting in CICS

- 3 hour test using HTTP and HTTPS into a single region.
  - 250 HTTPS clients & 100 HTTP clients.
  - The HTTPS clients read 12,150 MB of data, and wrote 1,121MB and the HTTP clients read 3,417 MB of data, and wrote 301MB.
- One CICS region, MXT=500, JVMSERVER Threads set to **50**.
  - At the end of the test, we'd run 4.2 million tasks, at a transaction rate of 326.59 per second.

“Library” application,
xmx = 500M, machine 2097 710



Millisecs of CPU per request zAAP vs CP



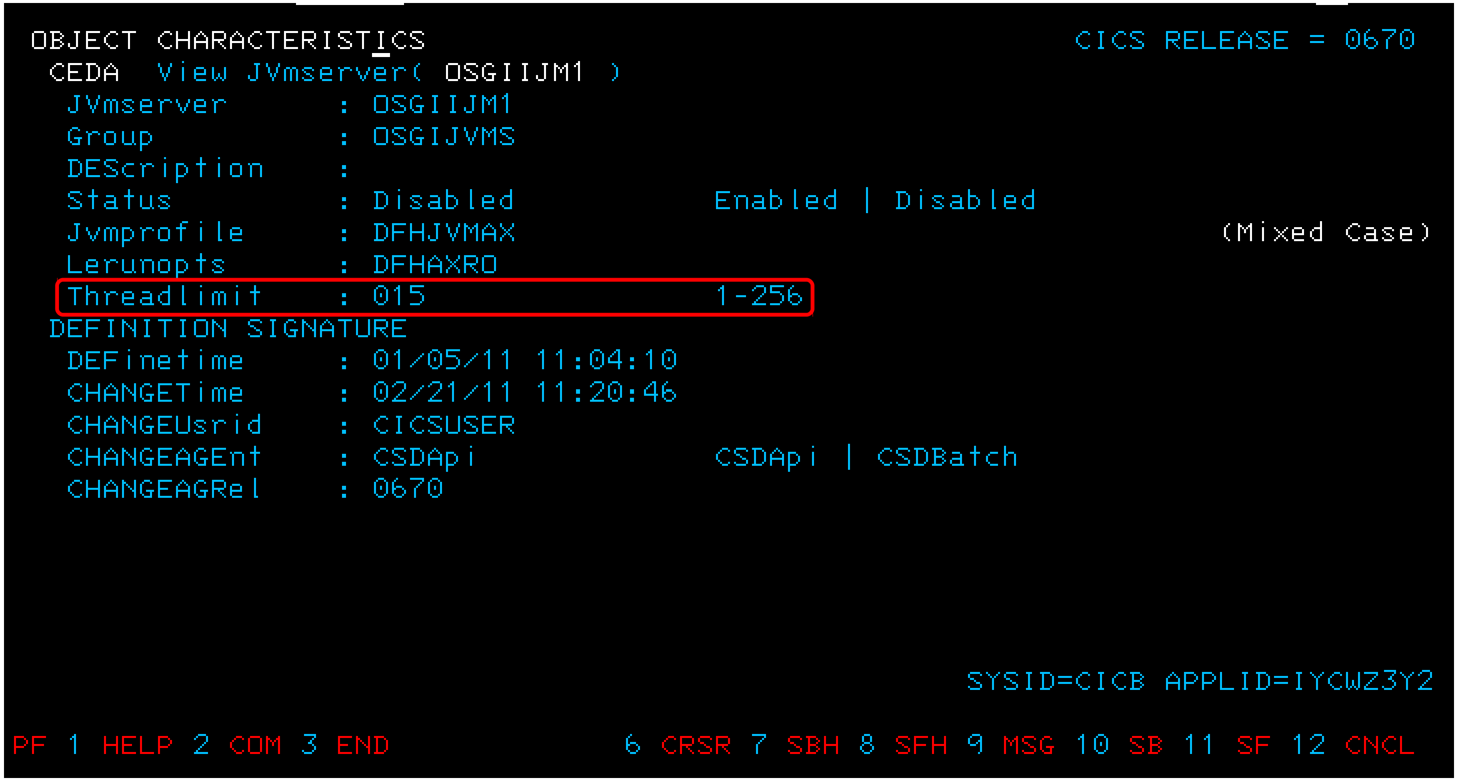
Topics

- Evolution of the JVM in CICS Transaction Server
- How are JVM Servers different to JVM Pools
- Defining JVM Server resources
- How applications run in JVM Servers
- The future for JVMs in CICS TS

Defining JVM Servers

- Resource Definition
  - CSD, BAS
- Profile – configures the JVM
  - In zFS
- LE RunOpts – configures the LE Enclave in which the JVM executes
  - In SDFHLOAD

JVM Server Resource Definition



```
OBJECT CHARACTERISTICS                                          CICS RELEASE = 0670
 CEDA  View JVmserver( OSGIIJM1 )
  JVmserver       : OSGIIJM1
  Group           : OSGIJVMS
  DEScription     :
  Status          : Disabled            Enabled | Disabled
  Jvmprofile      : DFHJVMAX                                            (Mixed Case)
  Lerunopts       : DFHAXRO
  Threadlimit     : 015                 1-256
 DEFINITION SIGNATURE
  DEFinetime      : 01/05/11 11:04:10
  CHANGETime      : 02/21/11 11:20:46
  CHANGEUsrid     : CICSUSER
  CHANGEAGEnt     : CSDApi              CSDApi | CSDBatch
  CHANGEAGRel     : 0670




                                                           SYSID=CICB APPLID=IYCWZ3Y2

PF 1 HELP 2 COM 3 END                  6 CRSR 7 SBH 8 SFH 9 MSG 10 SB 11 SF 12 CNCL
```

JVM Server Profile

- Sample is DFHJVMAX
- CICS finds such profiles in the directory identified in the SIT parm JVMPROFILEDIR
- File is an EBCDIC text file (typically IBM-1047)

- *(Note, use of “JVM Property Files” is deprecated in v4.1)*

JVM Server Profile contents

- Libpath – path to the native code required to run the JVM
  - `LIBPATH=/usr/lpp/java/J6.0/bin/j9vm:\ /usr/lpp/java/J6.0/bin`
- Workdir – directory for zFS files written by the JVM
  - STDIN, STDOUT, STDERR
- Standard JVM options - “-Xblah-blah”
  - -Xms16M,-Xmx32M, -Xoss4M, -Xss512K
- JVMServer classpath – where the apps will be found
  - `-Djava.class.path`
  - (We'll come back to this later!)

Notes – Example profile

SETUP_CLASSES and TERMINATION_CLASSES used to specify Java classes which are invoked when starting/terminating the JVM

THREAD_TIMEOUT – specifies how long a setup/termination class can run for before being timed out

- default of 60 secs (min 1, max 60,000)

LIBPATH – tells the shell which directories to search for dynamic-link libraries

- note, LIBPATH_PREFIX and LIBPATH_SUFFIX not supported

WORK_DIR – is a CICS option – (in the wrong section, will be moved in APAR) – set to /tmp by default

-Xms<size> – sets initial Java heap size

-Xmx<size> – sets maximum Java heap size

-Xoss<size> – sets maximum Java stack size for any thread

-Xss<size> – sets maximum native stack size for any thread

-Xgcthreads<size> – sets number of garbage collection helper threads

(defaults to one less then number of physical CPUs present)

-Djava.class.path – tells the JVM where to look for user-defined classes and packages in Java programs

JVMSERVER’s LE runtime options

- Sample is DFHAXRO
  - Compiled version in SDFHLOAD
- Override by recompiling source and placing in load library
- Certain options forced by CICS
  - XPLINK(ON)
  - POSIX(ON)
- Useful options:
  - RPTOPTS(ON) - generates, after an application has run, a report of the run-time options in effect while the application was running
  - RPTSTG(ON) - generates, after an application has run, a report of the storage the application used

Topics

- Evolution of the JVM in CICS Transaction Server
- How are JVM Servers different to JVM Pools
- Defining JVM Server resources
- How applications run in JVM Servers
- The future for JVMs in CICS TS

Applications running in JVM Servers

- Applications in JVM Servers have ALL the same facilities available as in a JVM in a pool
  - JCICS classes for accessing CICS services
  - JDBC *(however with a current restriction, more later)*
  - MQ classes
  - Debug via standard JPDA interfaces
- As with the change to ***Continuous mode*** there are semantic changes to sharing of ***static*** data
  - “static” data in a JVM Server will be concurrently accessed by multiple threads
  - Need to consider... um... threadsafety!

Static data in a JVM server

- The “static” keyword makes a variable into a 'class variable'
  - It is part of the class data so one copy exists – not one per instance of the class
- Uses for static data...
  - 'global' values
  - A 'global' collection or cache

Semantic evolution of static scope in CICS JVMs

- Version 2, JDK 1.4.2 Persistent Reusable JVM
  - One task at a time in the JVM
  - Application statics reinitialised between uses of the JVM
  - Cannot use it to hold application state between requests
- Version 3, Continuous mode
  - One task at a time in the JVM
  - Application statics persist from one use to the next
  - Can hold long-lived data, but serial access
- Version 4, JVM server
  - Multiple tasks at the same time in the JVM
  - Application statics persist
  - Visible to multiple tasks concurrently

JVM Server threadsafety – an example

- The JCICS implementation had lent on pooled JVM serialisation.
- The 'task' object declared as a static (implementing the singleton pattern).
  - OK when there IS only one task in the JVM
- Corrected to be “ThreadLocal”
  - ThreadLocal introduced in Java 5 to handle precisely this sort of requirement
- MORAL – CICS Java is now just like Java everywhere with respect to concurrency
  - See http://java.sun.com/developer/technicalArticles/J2SE/concurrency/

JVM Server JDBC restriction

- We fixed JCICS use of static data with affinity to a single task in the JVM.
- The JDBC driver has a similar issue – Hursley is working with SVL to address this.
- Current restriction is one task at a time can use JDBC
  - Documented restriction for Dynamic Scripting
  - Fully intend to remove the restriction ASAP.

Topics

- Evolution of the JVM in CICS Transaction Server
  - We keep learning from customer experience and investing
- How are JVM Servers different to JVM Pools
  - Revolution in scaleability
- Defining JVM Server resources
  - Getting easier
- How applications run in JVM Servers
  - Just like everywhere else
- The future for JVMs in CICS TS
  - Is very BRIGHT indeed!

Any Questions?

Thank You for your Attention

Please fill out a session evaluation form