



# Multitenancy and the Multi Tenant Java Virtual Machine

Session AAD-1357

*Peter Whitehead*

*Program Director, IBM Java Technology Center*

## Impact2014

Be **First.** ▶ ▶ ▶

**April 27 – May 1** | The Venetian – Las Vegas, NV

#ibmimpact

© 2014 IBM Corporation



# Please Note

IBM's statements regarding its plans, directions, and intent are subject to change or withdrawal without notice at IBM's sole discretion.

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.

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.



# What I hope to achieve in the next 60 mins

- ▶ Provide an overview of the Multitenancy capability being provided in the IBM Java™ Technology Edition, Version 7 Release 1
- ▶ Provide an overview of where this technology could be applied
- ▶ A demonstration of the resource control features of the Multi Tenant Virtual machine

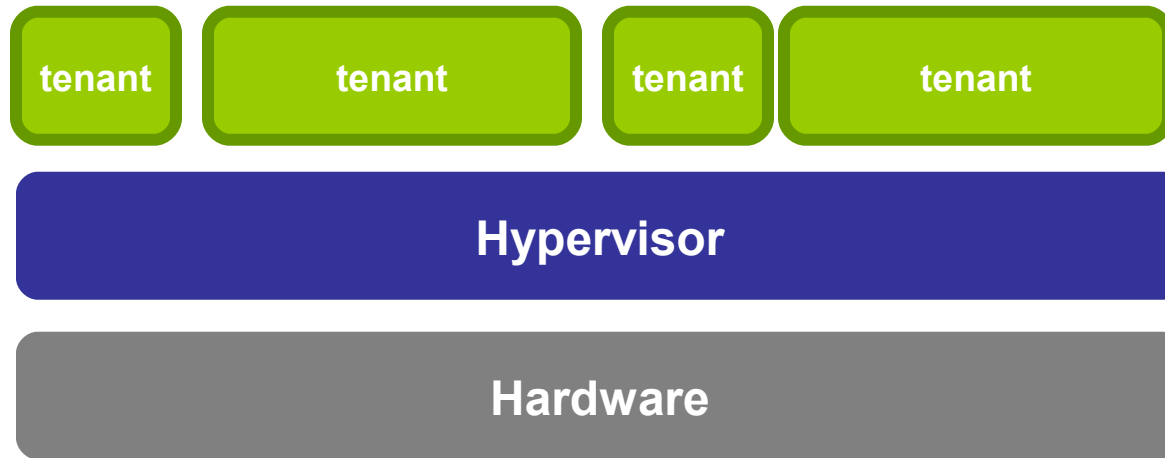


# Increase Application Density

- **Do more with less !!**
- Focus on making the IBM JRE as efficient and effective as possible
  - Foot print
  - Start up
  - Throughput
- To make a step change in density requires us to look at the problem differently
  - Increase sharing of VM artefacts
  - improve isolation between Java applications
- Multitenant Java is born
- Leading to high density Java applications solutions



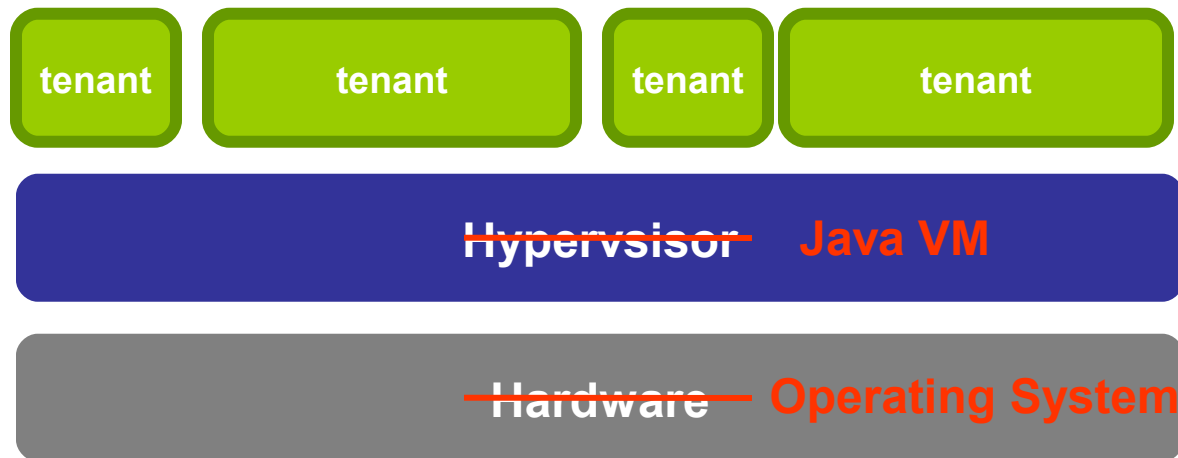
# Hardware Virtualization



- **Hypervisors run multiple applications side-by-side safely**
  - Examples: VMware, kvm, PowerVM, zVM
- **Advantages**
  - Capture idle CPU cycles
  - Ability to meter and shift resource toward demand
  - No need to change applications (tenants)
  - Many apps that do not need all system resources



# JVM Virtualization



- ~~Hypervisors~~ JVMs can run multiple applications side-by-side safely
- **Advantages**
  - Capture idle CPU cycles
  - Automatic de-duplication (ability to share Java artifacts)
  - Ability to meter and shift resource toward demand



# Multitenancy: Basics- What is it ?

- **What we're building: basically a 'virtual JVM'**

- Transparent multitenancy for 100% pure Java applications, opt-in via –Xmt option
- Shared JVM (javad) process hosts all tenants with in/out/err redirection to launcher
- JVM-enforced resource controls on Heap, Threads, I/O, and CPU
- Will behave exactly like a dedicated JVM, only smaller

- **Performance Goals (work in progress)**

- 10x density improvement on “Hello World” style applications
- 3x density improvement on larger OSGi applications (Liberty)
- Less than 10% throughput degradation on TradeLite



# Multitenancy: Basics – How do it get it ?

Tech preview in the IBM® SDK, Java™ Technology Edition, Version 7  
Release 1:

–Full platform evaluation Linux-x86, z/OS, AIX, zLinux, pLinux

▪ Download from:

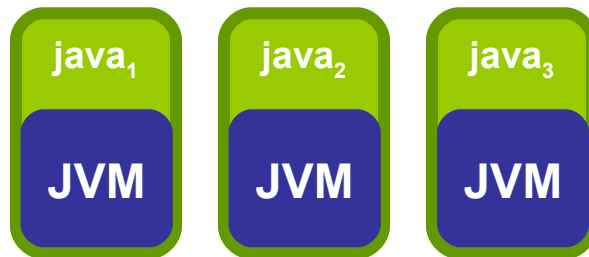
–<https://www.ibm.com/developerworks/java/jdk/linux/download.html>



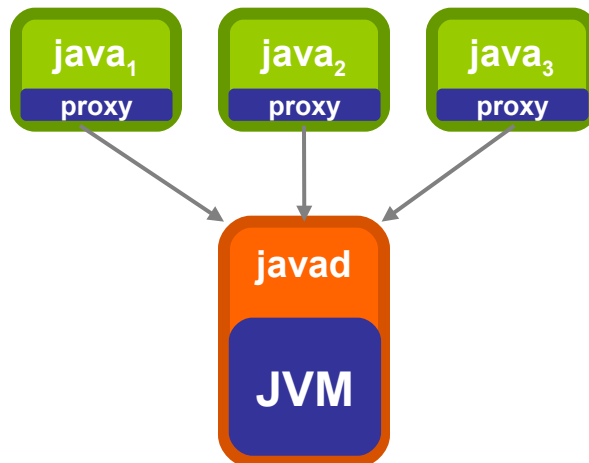


# Multitenancy: Basics – How it works

- A standard **java** invocation creates a dedicated (non-shared) JVM in each process



- IBM's Multitenant JVM puts a lightweight 'proxy' JVM in each **java** invocation. The 'proxy' knows how to communicate with a shared JVM daemon called **javad**.



- **javad** is launched and shuts down automatically
- no changes required to the application
- **javad** process is where aggressive sharing of runtime artifacts happens



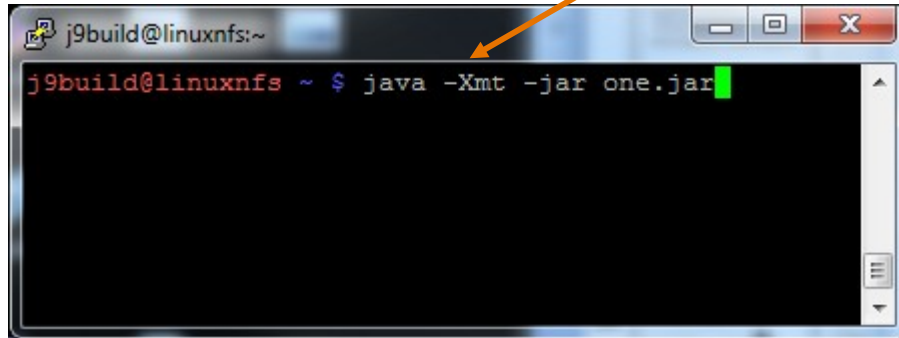
# Multitenancy: Getting Started

- **An opt-in feature of IBM Java™ Technology Edition Version 7 Release 1**
  - Just add the `-xmt` command-line option to opt-in
  - Enables a model very similar to JSR-121: Isolates but doesn't require any new API
- **Daemon startup and communications is handled automatically by the 'java' launcher**
  - One daemon per user to keep permissions aligned between launcher & daemon
  - Launcher:daemon rendezvous accomplished using advertisement files
- **Standard in / out / error streams are connected to daemon**
  - e.g. `System.out.println()` in the daemon works as expected
  - JVM will multicast messages like dump events to all connected tenants
- **Most standard JVM options are used as-is**
  - `-classpath` / `-jar` entries
  - `-Dname=value` system properties
- **Select JVM options are mapped to tenant-specific values**
  - `-Xmx` applies to the tenant being launched
  - See documentation for details
- **Daemon-wide options are stored in `JAVA_HOME/bin/javad.options` file**
- **Documentation available at:**
  - [http://pic.dhe.ibm.com/infocenter/java7sdk/v7r0/index.jsp?topic=%2Fcom.ibm.java.aix.71.doc%2Fdiag%2Fpreface%2Fchanges\\_71%2Foverview\\_mt\\_evaluation.html](http://pic.dhe.ibm.com/infocenter/java7sdk/v7r0/index.jsp?topic=%2Fcom.ibm.java.aix.71.doc%2Fdiag%2Fpreface%2Fchanges_71%2Foverview_mt_evaluation.html)



# Multitenant JDK: Launch your application

- Opt-in to multitenancy by adding `-Xmt`



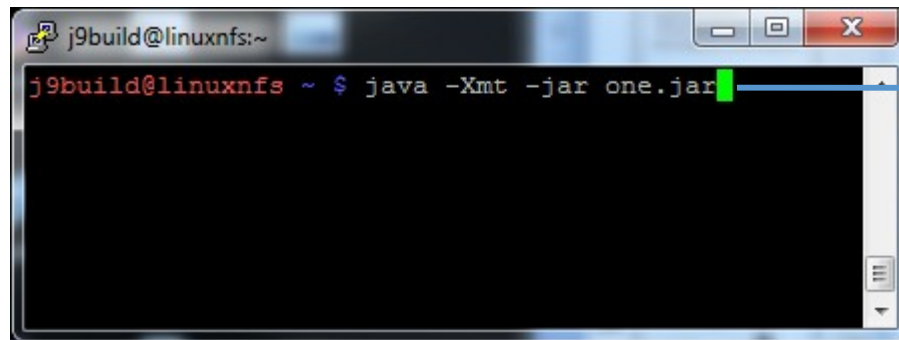
```
j9build@linuxnfs:~$ java -Xmt -jar one.jar
```

A terminal window titled 'j9build@linuxnfs:~' showing the command 'java -Xmt -jar one.jar' being executed. An orange arrow points from the text '-Xmt' in the list item above to the '-Xmt' flag in the terminal command.



# Multitenant JDK: Register with `javad` daemon

- JVM will locate/start daemon automatically



```
j9build@linuxnfs:~  
j9build@linuxnfs ~ $ java -Xmt -jar one.jar
```

locate

javad

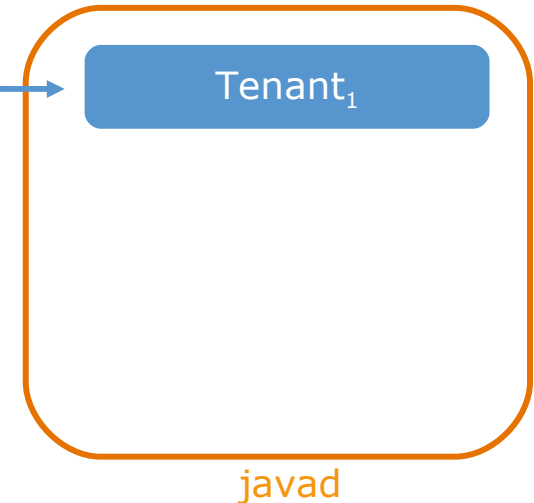


# Multitenant JDK: Create a new tenant

- New tenant created inside the `javad` daemon

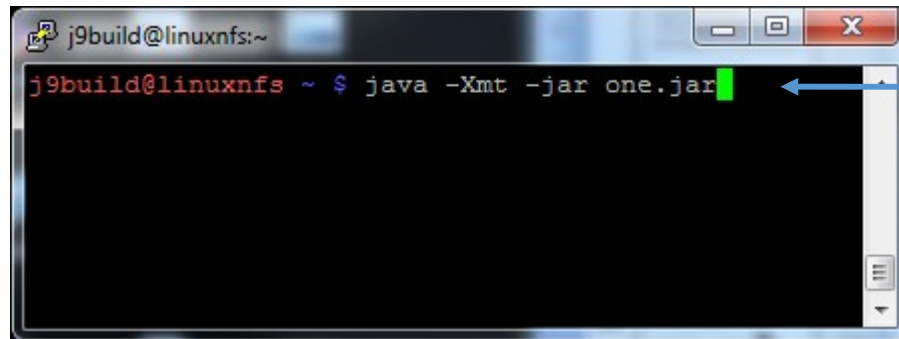


```
j9build@linuxnfs:~  
j9build@linuxnfs ~ $ java -Xmt -jar one.jar
```

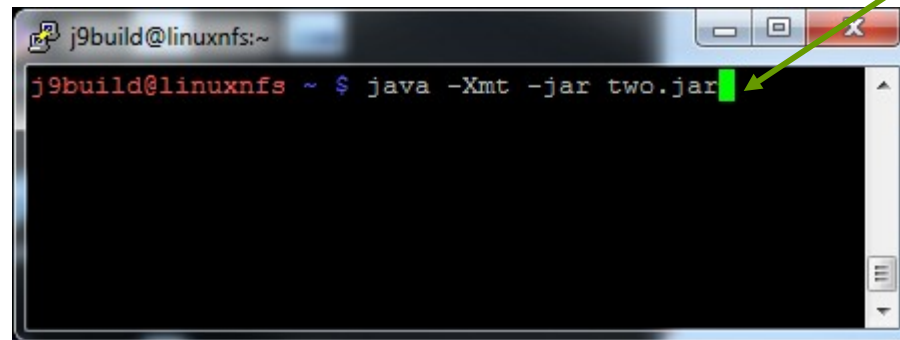


# Multitenant JDK: Create a second tenant

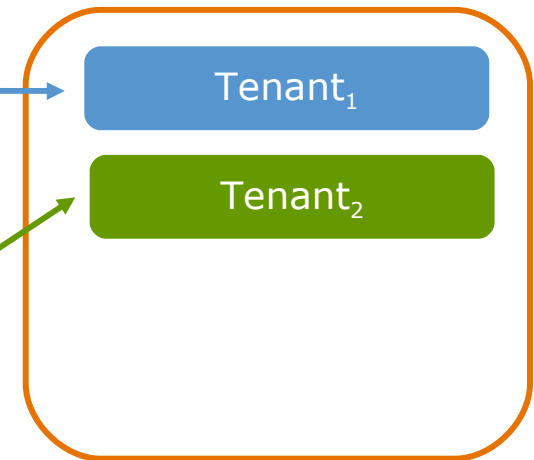
- New tenant created inside the `javad` daemon



```
j9build@linuxnfs:~  
j9build@linuxnfs ~ $ java -Xmt -jar one.jar
```



```
j9build@linuxnfs:~  
j9build@linuxnfs ~ $ java -Xmt -jar two.jar
```



`javad`



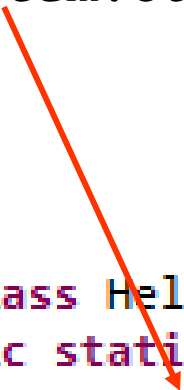
One copy of common code lives in the `javad` process.

Most runtime structures are shared.



# Multitenant JVM: Separating State

- **Static Variables are a problem for sharing**
- **Consider use of `System.out` in code we want to share below**



```
1 public class HelloWorld {  
2     public static void main(String[] args) {  
3         System.out.println("Hello");  
4     }  
5 }
```



# Multitenant JVM: Separating State - Isolation

```
1 public class HelloWorld {  
2     public static void main(String[] args) {  
3         System.out.println("Hello");  
4     }  
5 }
```

HelloWorld

```
| // access flags 0x9  
public static main([Ljava/lang/String;)V  
L0  
LINENUMBER 3 L0  
GETSTATIC java/lang/System.out : Ljava/io/PrintStream;  
LDC "Hello"  
INVOKEVIRTUAL java/io/PrintStream.println(Ljava/lang/String;)V  
L1  
LINENUMBER 4 L1  
RETURN
```

## getstatic does 2 things

1. Triggers class initialization on first contact
  - Notable: Each 'tenant' needs to do this
2. Resolves a name (`out`) to a storage location and reads from it
  - Notable: Each tenant needs dedicated storage





# Multitenancy: Controlling Resource Consumption

- **The second key feature for safe multitenancy is resource control**
  - Based on JSR-284 for resource configuration management
  - Internally uses a token-bucket algorithm commonly applied to network traffic shaping
- **Throttling is controlled using a new `-Xlimit` command-line option**
  - General form is: `-Xlimit:<resource_name>=<min_limit>-<max_limit>`
  - `<min_limit>`: Specifies the minimum amount of the resource that must be available for the tenant to start. This value is optional.
  - `<max_limit>`: Specifies the maximum amount of the resource that the tenant is allowed to use.
- **Resources that can be throttled include:**
  - CPU & Threads**
  - Heap** memory consumption
  - Disk and Network **I/O**



# Multitenancy: Controlling Resource Consumption

## ■ Examples:

— **-Xlimit:cpu=10-30**

- requires a 10% share of the processor to start and limits processor consumption to 30%.

— **-Xlimit:threads=5-20**

- requires a minimum reservation of five threads and an upper limit of 20

— **-Xmx20m**

- Limit heap consumption to 20 megabytes



# Multitenancy Sweet Spot

- **How low can you go?**
  - Simple ('Hello World') applications showing per-tenant sizes of ~170 KB of heap
  - This equates to a **5-6x** more applications running on the same hardware
- **Performance**
  - Target is 10% throughput overhead, still a work in progress
- **Second-run start-up times are significantly better**
  - Faster because the JVM is already up and running – batch like operations
- **Application Sweet spot:**
  - **One of:**
    - Relatively large class:heap ratio (JRuby and other JVM languages) lots of classes but not a lot of heap – scripting in java
    - Require fast startup: run-and-done / batch
    - Workloads with varying busy:idle cycles – MT JDK is better at shifting resource between tenants
  - **100% pure Java Code**



# Multitenancy: Caveats and Limitations

## ▪ Main Limitations of the MT Model

### JNI Natives

- The operating system allows the shared JVM process to load only one copy of a shared library. Only native libraries present on the bootclasspath of the JVM usable.

### JVMTI

- Because debugging and profiling activities impact all tenants that share the JVM daemon process, these features are not supported in the multitenant JVM process model. Note: we do have per-tenant -javaagent: support.

### GUI programs

- Libraries such as the Standard Widget Toolkit (SWT) are not supported in the multitenant JVM process model because the libraries maintain a global state in the native layer.

## – Full list available at

- [http://pic.dhe.ibm.com/infocenter/java7sdk/v7r0/index.jsp?topic=%2Fcom.ibm.java.aix.71.doc%2Fuser%2Fmt\\_limitations.html](http://pic.dhe.ibm.com/infocenter/java7sdk/v7r0/index.jsp?topic=%2Fcom.ibm.java.aix.71.doc%2Fuser%2Fmt_limitations.html)



# Use Cases

- MT-UC1 – Small Application Consolidation
- MT-UC2 – Run and Done
- MT-UC3 – Resource Time Sharing
- MT-UC4 – Resource Limiting based on SLA
- MT-UC5 – Resource Limiting for Safety
- MT-UC6 – Health Monitoring and Recovery



# MT-UC1: Small Application Consolidation

- Key attributes
  - Customer has multiple small applications
  - Non EE deployment OR more isolation needed between applications than provided by EE deployment OR need per application Middleware instances (ex liberty) due to management/operational requirements – boot class path isolation
  - Application memory/CPU overhead low compared to JVM overhead
  - Density
- MT Benefit:
  - Lower total footprint/memory requirements
    - Limit overhead to that of 1 JVM versus many
    - Limit heap to 1 shared head heap



# MT-UC2: Run and Done

- Key attributes
  - Short running application with multiple invocations
  - Startup/Shutdown dominates run time
  - Need Isolation between invocations
  - Examples: Ant scripts, compilation with javac, IMS, Z Batch, JRuby scripts, Jython scripts etc.
- MT Benefit:
  - Faster startup/shutdown
    - Avoid full JVM startup/shutdown for each invocation



# MT-UC3: Resource Time Sharing

- Key attributes
  - Customer has multiple applications that have load at different times
  - Non EE deployment OR more isolation needed between applications than provided by EE deployment OR need per application Middleware instances (ex liberty) due to management/operational requirements
  - Keep heap to a defined limit – peak concurrent load – no swapping
- MT Benefit:
  - Lower total footprint/memory requirements
    - Limit overhead to that of 1 JVM versus many
    - Shared heap sized to match concurrent maximum instead of sum of all application maximums





# MT-UC4: Resource Limiting based on SLA

- Key attributes
  - Customer has multiple applications sharing same OS instance
  - Some applications have higher SLA levels than others
- MT Benefit:
  - Able to control CPU, Network IO , File IO resource usage to favor application with higher SLA



# MT-UC5: Resource Limiting for Safety

- Key attributes
  - Customer has multiple applications sharing same OS instance
  - Some applications un-trusted or buggy, concern they will affect performance of other applications.
- MT Benefit:
  - Able to control CPU, Network IO , File IO resource usage to limit maximum impact of “runaway” application



# MT-UC6: Health monitoring and recovery

- Key attributes
  - Health monitoring/recovery runs in JVM with application
  - Application failures should not affect health monitoring (ex OOM on app)
- MT Benefits
  - Ability to ensure minimum amount of memory available to health monitoring/recovery components



# Demo: Scenario

- JVM Health monitoring
  - Want heartbeat to track “liveness” of server
  - Need this to run reliably as long as application is still running
  - Requires some memory and cpu to generate heartbeat
  - Simulate in demo with thread that prints out “heartbeat” at 2 second interval
- Application Transactions
  - Transactions submitted from external system
  - Use variable amount of cpu depending on request
  - If transaction uses too much memory it can starve heartbeat thread
  - Simulate with thread(s) that uses as much cpu as they can



# Demo: What happens today

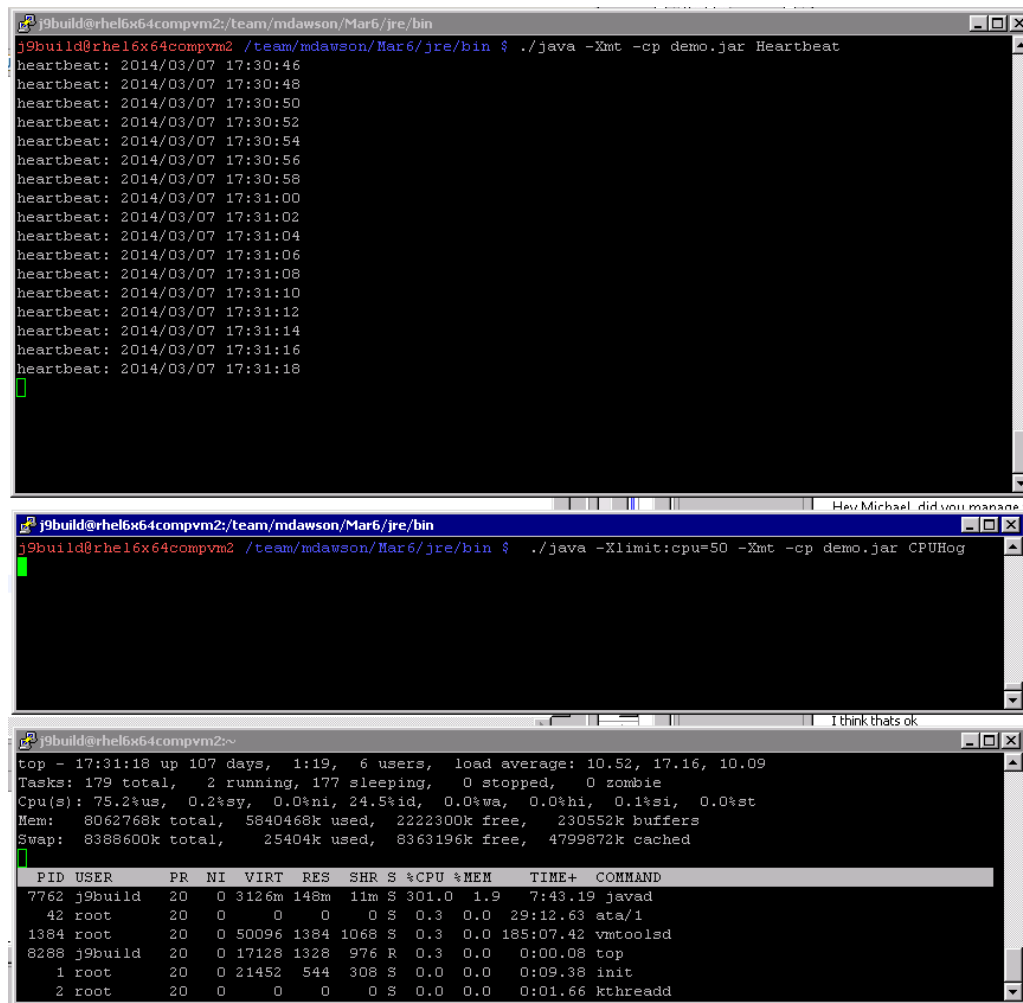
- Run HeartbeatAndCPUHog
- `./java -cp demo.jar HeartbeatAndCPUHog`
- Shows running both heartbeat thread and transaction in regular jvm
- Note that times between heartbeat messages stretch out once cpu hog starts

```
j9build@rhel6x64compvm2 /team/mdawson/Mar6/jre/bin $ cat javad.options^C
j9build@rhel6x64compvm2 /team/mdawson/Mar6/jre/bin $ ./java -cp demo.jar HeartbeatAndCPUHog
heartbeat: 2014/03/07 17:23:26
heartbeat: 2014/03/07 17:23:28
heartbeat: 2014/03/07 17:23:30
heartbeat: 2014/03/07 17:23:32
heartbeat: 2014/03/07 17:23:34
CPU Hog starting
heartbeat: 2014/03/07 17:23:36
heartbeat: 2014/03/07 17:23:44
heartbeat: 2014/03/07 17:23:53
heartbeat: 2014/03/07 17:24:03
heartbeat: 2014/03/07 17:24:12
heartbeat: 2014/03/07 17:24:22
heartbeat: 2014/03/07 17:24:33
```



# Demo: Use MT

- `./java -Xmt -cp demo.jar Heartbeat` in one window
- `./java -Xlimit:cpu=10 -Xmt -cp demo.jar CPUHog` in another window
- Top in third window
- Note that heartbeat remains consistent even after hog starts
- Top shows that cpu varies, but average looks to be around 50%
- You might have to play with limit depending on your machine as heartbeat task does use reasonable amount of cpu to show affect.



```
j9build@rhel6x64compvm2:/team/mdawson/Mar6/jre/bin
j9build@rhel6x64compvm2 /team/mdawson/Mar6/jre/bin $ ./java -Xmt -cp demo.jar Heartbeat
heartbeat: 2014/03/07 17:30:46
heartbeat: 2014/03/07 17:30:48
heartbeat: 2014/03/07 17:30:50
heartbeat: 2014/03/07 17:30:52
heartbeat: 2014/03/07 17:30:54
heartbeat: 2014/03/07 17:30:56
heartbeat: 2014/03/07 17:30:58
heartbeat: 2014/03/07 17:31:00
heartbeat: 2014/03/07 17:31:02
heartbeat: 2014/03/07 17:31:04
heartbeat: 2014/03/07 17:31:06
heartbeat: 2014/03/07 17:31:08
heartbeat: 2014/03/07 17:31:10
heartbeat: 2014/03/07 17:31:12
heartbeat: 2014/03/07 17:31:14
heartbeat: 2014/03/07 17:31:16
heartbeat: 2014/03/07 17:31:18

j9build@rhel6x64compvm2:/team/mdawson/Mar6/jre/bin $ ./java -Xlimit:cpu=50 -Xmt -cp demo.jar CPUHog

top - 17:31:18 up 107 days, 1:19, 6 users, load average: 10.52, 17.16, 10.09
Tasks: 179 total, 2 running, 177 sleeping, 0 stopped, 0 zombie
Cpu(s): 75.2%us, 0.2%sy, 0.0%ni, 24.5%id, 0.0%wa, 0.0%hi, 0.1%si, 0.0%st
Mem: 8062768k total, 5840468k used, 2222300k free, 230552k buffers
Swap: 8388600k total, 25404k used, 8363196k free, 4799872k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU  %MEM    TIME+  COMMAND
 7762 j9build   20   0 3126m 148m 11m S 301.0  1.9   7:43.19 javad
   42 root      20   0    0    0    0 S   0.3   0.0   29:12.63 ata/1
 1384 root      20   0 50096 1384 1068 S   0.3   0.0 185:07.42 vmttoolsd
 8288 j9build   20   0 17128 1328  976 R   0.3   0.0   0:00.08 top
    1 root      20   0 21452  544  308 S   0.0   0.0   0:09.38 init
    2 root      20   0    0    0    0 S   0.0   0.0   0:01.66 kthreadd
```



# Key Links and Contacts

- Download
  - <https://www.ibm.com/developerworks/java/jdk/linux/download.html>
- Documentation
  - [http://pic.dhe.ibm.com/infocenter/java7sdk/v7r0/index.jsp?topic=%2Fcom.ibm.java.aix.71.doc%2Fdiag%2Fpreface%2Fchanges\\_71%2Foverview\\_mt\\_evaluation.html](http://pic.dhe.ibm.com/infocenter/java7sdk/v7r0/index.jsp?topic=%2Fcom.ibm.java.aix.71.doc%2Fdiag%2Fpreface%2Fchanges_71%2Foverview_mt_evaluation.html)
- Contacts for feedback
  - **Peter Whitehead** ([peter\\_whitehead@uk.ibm.com](mailto:peter_whitehead@uk.ibm.com))
  - **Michael Dawson** ([michael\\_dawson@ca.ibm.com](mailto:michael_dawson@ca.ibm.com))



Questions?

**Impact**2014

Be **First.** ▶▶▶

#ibmimpact





# Other Java Related Sessions

Wednesday, 30-April:

13:00-14:00	<b>Multitenancy and the Multi-Tenant JVM</b>	Delfino 4103
14:15-15:15	<b>Performance Optimization Using IBM Java on z/OS &amp; IBM WebSphere Application Server on z/OS V8.5.5</b>	Lando 4301B
17:00-18:00	<b>Understand &amp; Improve the Performance of Your Application</b>	Delfino 4105

Thursday, 01-May:

09:00-11:30	<b>Lab: Diagnostic and Performance Tools for WebSphere</b>	Murano 3205
-------------	--	-------------



# We Value Your Feedback

- ▶ Don't forget to submit your Impact session and speaker feedback! Your feedback is very important to us – we use it to continually improve the conference.
- ▶ Use the Conference Mobile App or the online Agenda Builder to quickly submit your survey
  - Navigate to “Surveys” to see a view of surveys for sessions you’ve attended





# Thank You

## Impact2014



#ibmimpact

Be **First.** ▶▶▶

## Legal Disclaimer

- © IBM Corporation 2014. All Rights Reserved.
- The information contained in this publication is provided for informational purposes only. While efforts were made to verify the completeness and accuracy of the information contained in this publication, it is provided AS IS without warranty of any kind, express or implied. In addition, this information is based on IBM's current product plans and strategy, which are subject to change by IBM without notice. IBM shall not be responsible for any damages arising out of the use of, or otherwise related to, this publication or any other materials. Nothing contained in this publication 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. Product release dates and/or capabilities referenced in this presentation may change at any time at IBM's sole discretion based on market opportunities or other factors, and are not intended to be a commitment to future product or feature availability in any way. 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.
- If the text contains performance statistics or references to benchmarks, insert the following language; otherwise delete:  
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.
- If the text includes any customer examples, please confirm we have prior written approval from such customer and insert the following language; otherwise delete:  
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.
- Please review text for proper trademark attribution of IBM products. At first use, each product name must be the full name and include appropriate trademark symbols (e.g., IBM Lotus® Sametime® Unyte™). Subsequent references can drop "IBM" but should include the proper branding (e.g., Lotus Sametime Gateway, or WebSphere Application Server). Please refer to <http://www.ibm.com/legal/copytrade.shtml> for guidance on which trademarks require the ® or ™ symbol. Do not use abbreviations for IBM product names in your presentation. All product names must be used as adjectives rather than nouns. Please list all of the trademarks that you use in your presentation as follows; delete any not included in your presentation. IBM, the IBM logo, Lotus, Lotus Notes, Notes, Domino, Quickr, Sametime, WebSphere, UC2, PartnerWorld and Lotusphere are trademarks of International Business Machines Corporation in the United States, other countries, or both. Unyte is a trademark of WebDialogs, Inc., in the United States, other countries, or both.
- If you reference Adobe® in the text, please mark the first use and include the following; otherwise delete:  
Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.
- If you reference Java™ in the text, please mark the first use and include the following; otherwise delete:  
Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.
- If you reference Microsoft® and/or Windows® in the text, please mark the first use and include the following, as applicable; otherwise delete:  
Microsoft and Windows are trademarks of Microsoft Corporation in the United States, other countries, or both.
- If you reference Intel® and/or any of the following Intel products in the text, please mark the first use and include those that you use as follows; otherwise delete:  
Intel, Intel Centrino, 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.
- If you reference UNIX® in the text, please mark the first use and include the following; otherwise delete:  
UNIX is a registered trademark of The Open Group in the United States and other countries.
- If you reference Linux® in your presentation, please mark the first use and include the following; otherwise delete:  
Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both. Other company, product, or service names may be trademarks or service marks of others.
- If the text/graphics include screenshots, no actual IBM employee names may be used (even your own), if your screenshots include fictitious company names (e.g., Renovations, Zeta Bank, Acme) please update and insert the following; otherwise delete: All references to [insert fictitious company name] refer to a fictitious company and are used for illustration purposes only.

