Virtualization Aware JVM

Making the most of a virtualized environment

Tim Ellison – Hursley labs

Important Disclaimers

THE INFORMATION CONTAINED IN THIS PRESENTATION IS PROVIDED FOR INFORMATIONAL PURPOSES ONLY.

WHILST 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.

ALL PERFORMANCE DATA INCLUDED IN THIS PRESENTATION HAVE BEEN GATHERED IN A CONTROLLED ENVIRONMENT. YOUR OWN TEST RESULTS MAY VARY BASED ON HARDWARE, SOFTWARE OR INFRASTRUCTURE DIFFERENCES.

ALL DATA INCLUDED IN THIS PRESENTATION ARE MEANT TO BE USED ONLY AS A GUIDE.

IN ADDITION, THE INFORMATION CONTAINED IN THIS PRESENTATION IS BASED ON IBM’S CURRENT PRODUCT PLANS AND STRATEGY, WHICH ARE SUBJECT TO CHANGE BY IBM, WITHOUT NOTICE.

IBM AND ITS AFFILIATED COMPANIES SHALL NOT BE RESPONSIBLE FOR ANY DAMAGES ARISING OUT OF THE USE OF, OR OTHERWISE RELATED TO, THIS PRESENTATION OR ANY OTHER DOCUMENTATION.

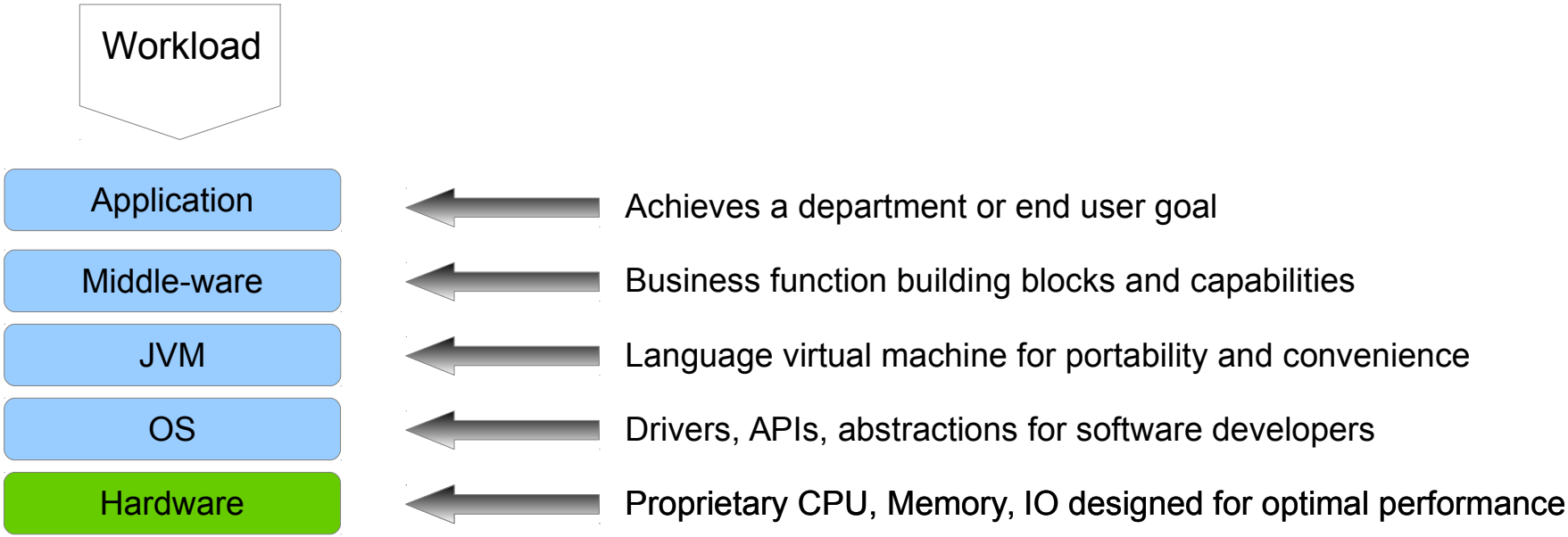
NOTHING CONTAINED IN THIS PRESENTATION IS INTENDED TO, OR SHALL HAVE THE EFFECT OF:

- CREATING ANY WARRANT OR REPRESENTATION FROM IBM, ITS AFFILIATED COMPANIES OR ITS OR THEIR SUPPLIERS AND/OR LICENSORS

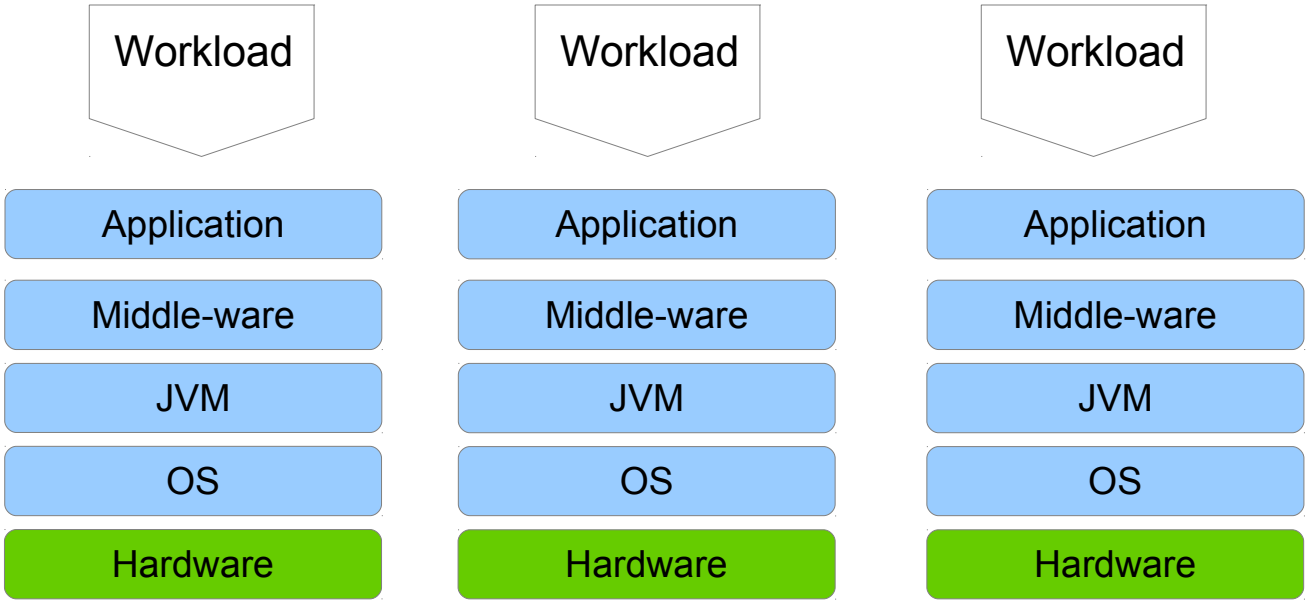
Goals of this talk

- Describe the impact of deep systems virtualization on the Java Virtual Machine (JVM) and Java Applications.
- Explain the benefits of making the JVM virtualization aware.
- Discuss techniques to optimize implementations for virtualized environments.

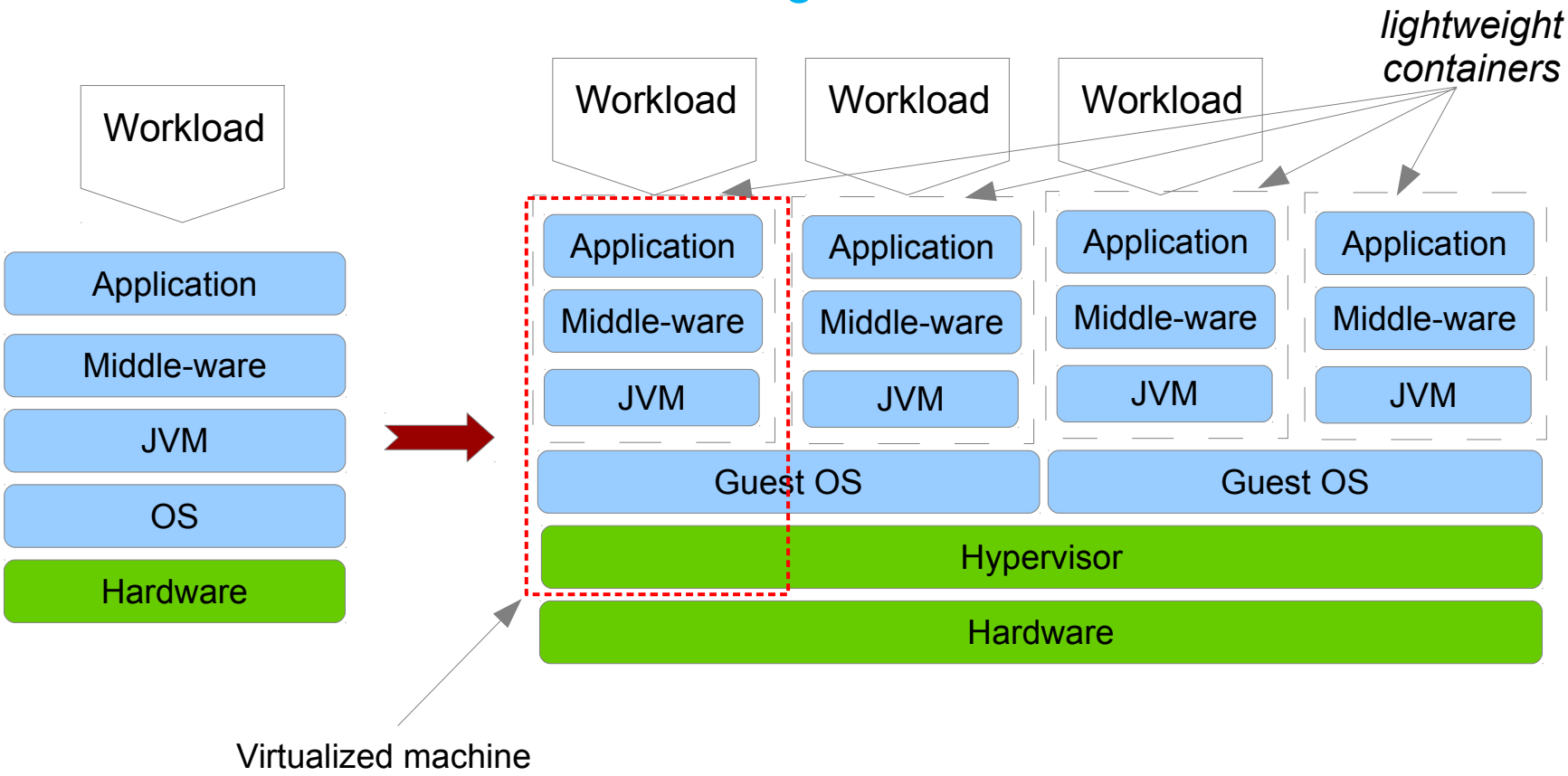
A stylized system environment



A data centre environment is no longer what it was...



A data centre environment is no longer what it was...



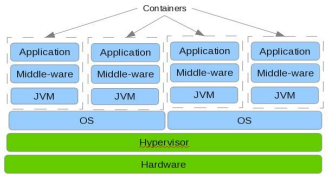
Virtualization available at every layer

- **Hardware Virtualization**
  - Helps pack in more applications compared to bare metal
  - These provide very good isolation at a cost of duplication of services → Higher Overhead
  - Usecase: Multiple customers can be hosted on the same infrastructure.

- **OS Virtualization - Containers**
  - New light weight containers such as Docker, Warden, AIX WPAR, Solaris Containers provide basic isolation mechanism with very little overhead.
  - Usecase: Multiple applications each running inside a container of its own for a single customer.

- **Runtime Virtualization – Virtualization aware Java VM**
  - Runtime Virtualization increases the resources that can be shared while still offering reduced isolation.
  - Usecase: Several instances of an application hosted on a single instance of Java for a single customer.

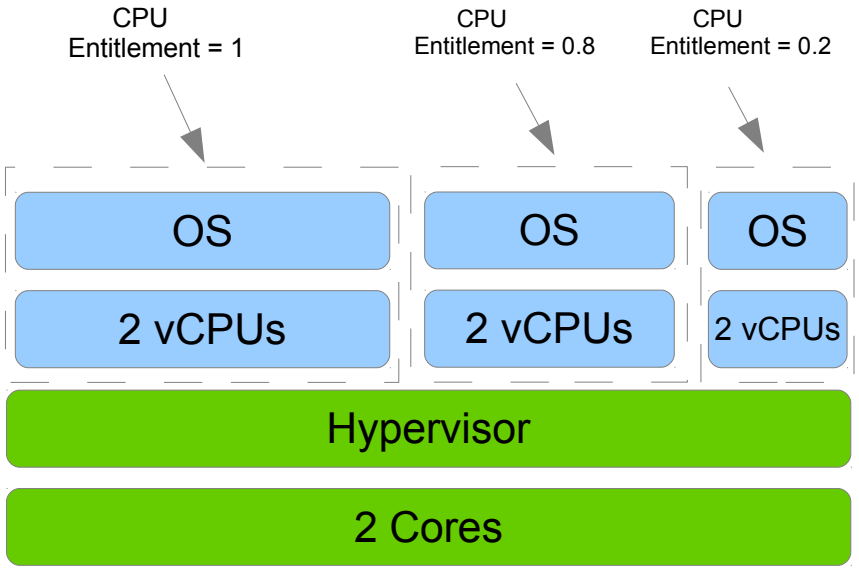
What does this virtualization story mean to the JVM ?

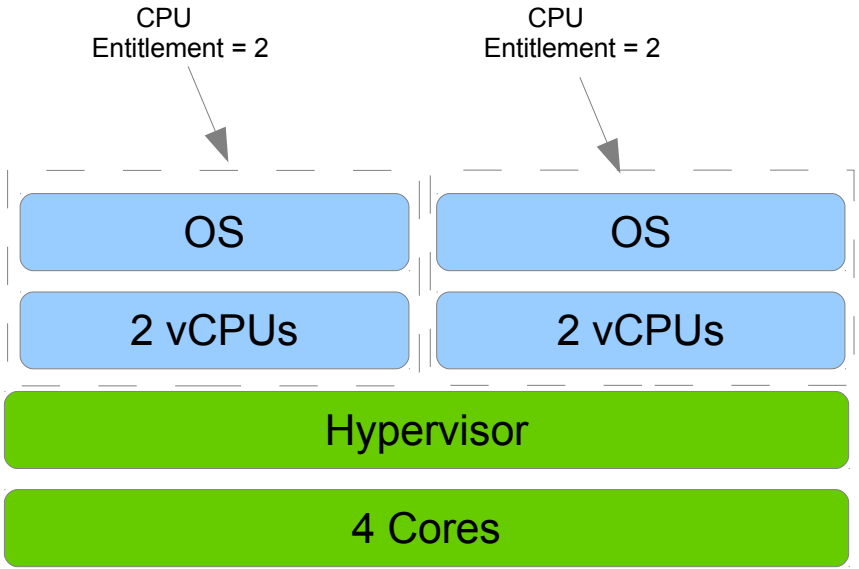


- Fixed view of hardware resources.
- JVM resources do not change during application life-cycle.
- Configuration usually determined at start-up, and doesn't change during the entire run.

- Resources are elastic. Hypervisors might “re-assign” resources based on need.
- Resource reporting APIs are unreliable !
- Resources are shared, and limits can change dynamically.

Physical View vs. Logical View – “How many CPUs do I have?”





Am I Real or Am I Virtual ?!

Q. Do Java applications need to know if they are running inside a Guest OS ?

A. Mostly No. Only a small class of applications benefit

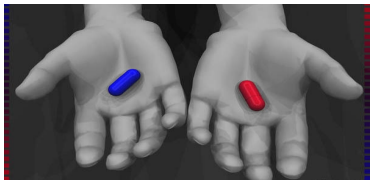
- Resource Orchestrators and Load Balancers
- Monitoring tools
- Debuggers and RAS Tools

Q. Does the JVM implementation need to know if it is running inside a Guest OS ?

A. Absolutely Yes !

- Manage Java resources based upon real physical resources.
- Provide virtualization info to applications.

Benefits of a virtualization aware JVM

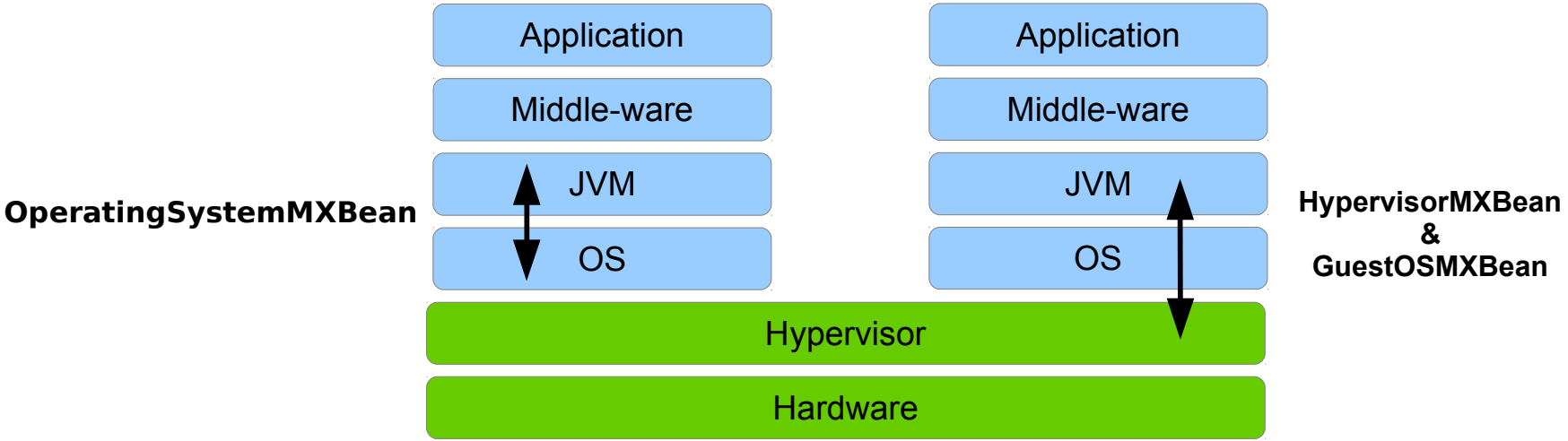


- Make the most efficient use of resources in a virtualized environment.
- Propagate the knowledge up the software stack to enable load balancers to take appropriate decisions.
- Remove necessity for multiple middle-ware products to understand intricacies of hypervisors.
- Provide unified interface to be able to deal with a multitude of hypervisors.

IBM's JMX Beans for Virtualization

- `com.ibm.lang.management.OperatingSystemMXBean` – with IBM Extensions
  - Extended OS usage statistics – Logical view
    - Processor
    - Memory
  - OS Support: AIX, Linux, Windows and z/OS
- `com.ibm.virtualization.management.HypervisorMXBean`
  - Detect if we are running on a hypervisor.
  - Hypervisor Details (Currently only Vendor Name string)
  - Hypervisors Supported: z/VM, PR/SM, PowerVM, VMWare, KVM (x86 and Power), Hyper-V
- `com.ibm.virtualization.management.GuestOSMXBean`
  - Guest OS usage statistics as seen from the hypervisor – Physical View
  - Usage Statistics
    - Processor
    - Memory
  - AIX & Linux on PowerVM, Linux and Windows on VMWare, z/OS & zLinux on z/VM

JMX Beans for Virtualization



OperatingSystemMXBean – New APIs

com.ibm.lang.management

Interface OperatingSystemMXBean

`MemoryUsage`	`retrieveMemoryUsage()` Instantiates and returns an instance of `MemoryUsage` object that represents the current snapshot of Memory usage statistics.
`MemoryUsage`	`retrieveMemoryUsage(MemoryUsage memObj)` Returns an updated `MemoryUsage` object that represents the current snapshot of Memory usage statistics.
`ProcessorUsage[]`	`retrieveProcessorUsage()` Instantiates and returns an array of `ProcessorUsage` objects that represent the current snapshot of individual Processor usage times.
`ProcessorUsage[]`	`retrieveProcessorUsage(ProcessorUsage[] procArray)` Returns an updated array of `ProcessorUsage` objects that represent the current snapshot of individual Processor usage times.
`ProcessorUsage`	`retrieveTotalProcessorUsage()` Instantiate and return a new `ProcessorUsage` object that represents the current snapshot of Processor usage statisticss.
`ProcessorUsage`	`retrieveTotalProcessorUsage(ProcessorUsage procObj)` Returns an updated `ProcessorUsage` object that represents the current snapshot of Processor usage statistics.

Method Summary

Methods

Modifier and Type	Method and Description
long	`getBuffered()` The physical RAM used for file buffers (in bytes) or -1 if not available on the underlying machine.
long	`getCached()` The physical RAM used as cache memory (in bytes) or -1 if not available on the underlying machine.
long	`getFree()` Returns the physical memory unused on the system (in bytes) or -1 if this is not available on the underlying machine.
long	`getSwapFree()` Total amount of free swap memory (in bytes) or -1 if this is not available on the underlying machine.
long	`getSwapTotal()` Total swap memory on the machine (in bytes) or -1 if this is not available on the underlying machine.
long	`getTimestamp()` Returns the timestamp when memory usage statistics were last sampled (in microseconds).
long	`getTotal()` Returns the total usable physical memory installed on the system (in bytes).

Method Summary

Methods

Modifier and Type	Method and Description
long	**getBusy()** Returns the time spent by the current processor in executing a non-idle thread (in microseconds) or else -1, if this is not available on the underlying machine.
int	**getId()** Returns a unique identifier assigned to the current processor.
long	**getIdle()** Returns the time spent by the processor sitting idle (in microseconds) or else -1, if this is not available on the underlying machine.
boolean	**getOnline()** Tells whether the current processor is online or not.
long	**getSystem()** Returns the time spent in system mode (in microseconds) or else -1, if this is not available on the underlying machine.
long	**getTimestamp()** The timestamp when processor usage statistics were last sampled.
long	**getUser()** Returns the time spent in user mode (in microseconds) or else -1, if this is not available on the underlying machine.
long	**getWait()** Returns the time spent by the current processor over IO wait (in microseconds) or else -1, if this is not available on the underlying machine.

Interface HypervisorMXBean

All Known Implementing Classes:

HypervisorMXBeanImpl

`public interface` **`HypervisorMXBean`**

The IBM-specific interface for finding out if the current system is running in a virtualized environment, and if so, information about that environment

Method Summary

Methods

Modifier and Type	Method and Description
`java.lang.String`	`getVendor()` Returns the vendor of the hypervisor if running in a virtualized environment
`boolean`	`isEnvironmentVirtual()` Indicates if the system is running in a virtualized environment or not

Interface GuestOSMXBean

All Superinterfaces:

java.lang.management.PlatformManagedObject

All Known Implementing Classes:

GuestOS

```
public interface GuestOSMXBean
extends java.lang.management.PlatformManagedObject
```

The IBM Specific MXBean interface that provides Guest OS statistics as seen by the hypervisor host

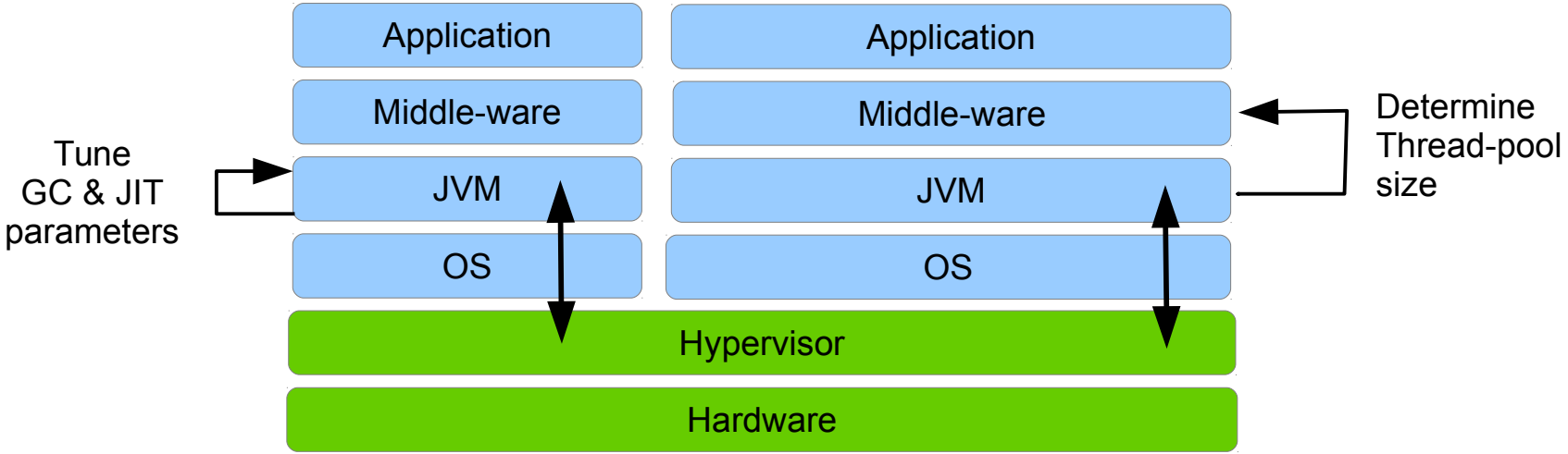
See Also:

GuestOS

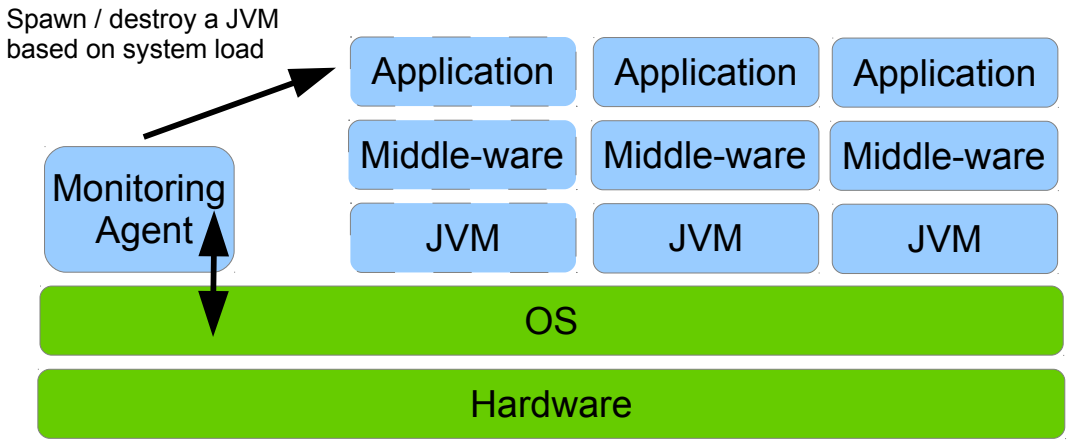
Method Summary

Modifier and Type	Method and Description
GuestOSMemoryUsage	retrieveMemoryUsage() Function instantiates a GuestOSMemoryUsage object with the current snapshot of Memory Usage statistics of the guest as seen by the hypervisor.
GuestOSMemoryUsage	retrieveMemoryUsage(GuestOSMemoryUsage gmUsage) Function updates the user provided GuestOSMemoryUsage object with Memory Usage statistics of the guest as seen by the hypervisor.
GuestOSProcessorUsage	retrieveProcessorUsage() Function instantiates a GuestOSProcessorUsage object with the current snapshot of Processor Usage statistics of the guest as seen by the hypervisor.
GuestOSProcessorUsage	retrieveProcessorUsage(GuestOSProcessorUsage gpUsage) Function updates the user provided GuestOSProcessorUsage object with Processor Usage statistics of the guest as seen by the hypervisor.

GuestOSMXBean use-case – Tune internals of JVM & Websphere



OperatingSystemMXBean use-case – Load Balance JVM Instances



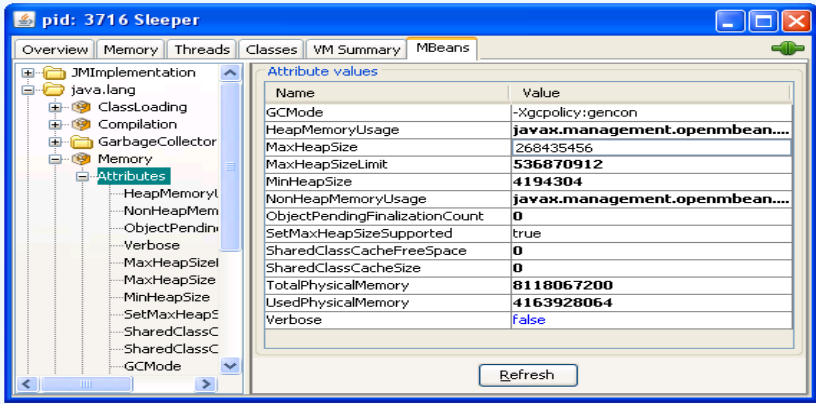
Implementation Notes

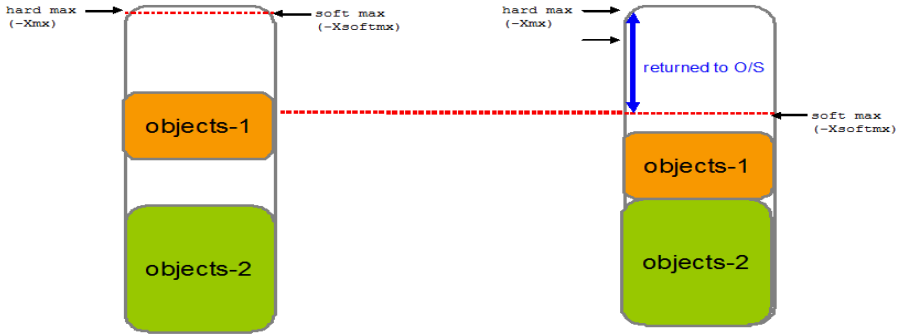
- The Beans are available in IBM Java SDK version 7.1 onwards
- Hypervisor specific setup may be required to obtain usage data.
  - VMWare requires VMGuestLib to be installed (Is part of VMware tools)
  - Hypfs needs to be mounted on zLinux
- Hypervisor detection known to fail on certain older Intel processors
  - IBM_JAVA_HYPERVISOR_SETTINGS environment variable can be used as a workaround
  - See javadoc for HypervisorMXBean for more details.
- Some data may not be available on specific OS / Hypervisor combinations.
  - Javadoc should have all relevant info.
- Currently only one level of hypervisor is supported. In case of multiple layers of hypervisor (e.g. on zSystems), only the top level hypervisor info is returned.
- Javacore now contains virtualization info.

Additional virtualization optimizations

Dynamic Heap Adjustment (softmx)

- `-Xmx`
  - absolute limit, fixed at startup
- `-Xsoftmx`
  - soft limit <= -Xmx, set dynamically through JMX
  - Garbage collector tries to shrink to softmx over time
  - Once at target will not expand beyond it
- OS Interaction
  - JVM advises OS when memory freed
  - Effectiveness depends on OS support
- Use cases
  - Cap early, grow later
  - Shrink to free unused memory

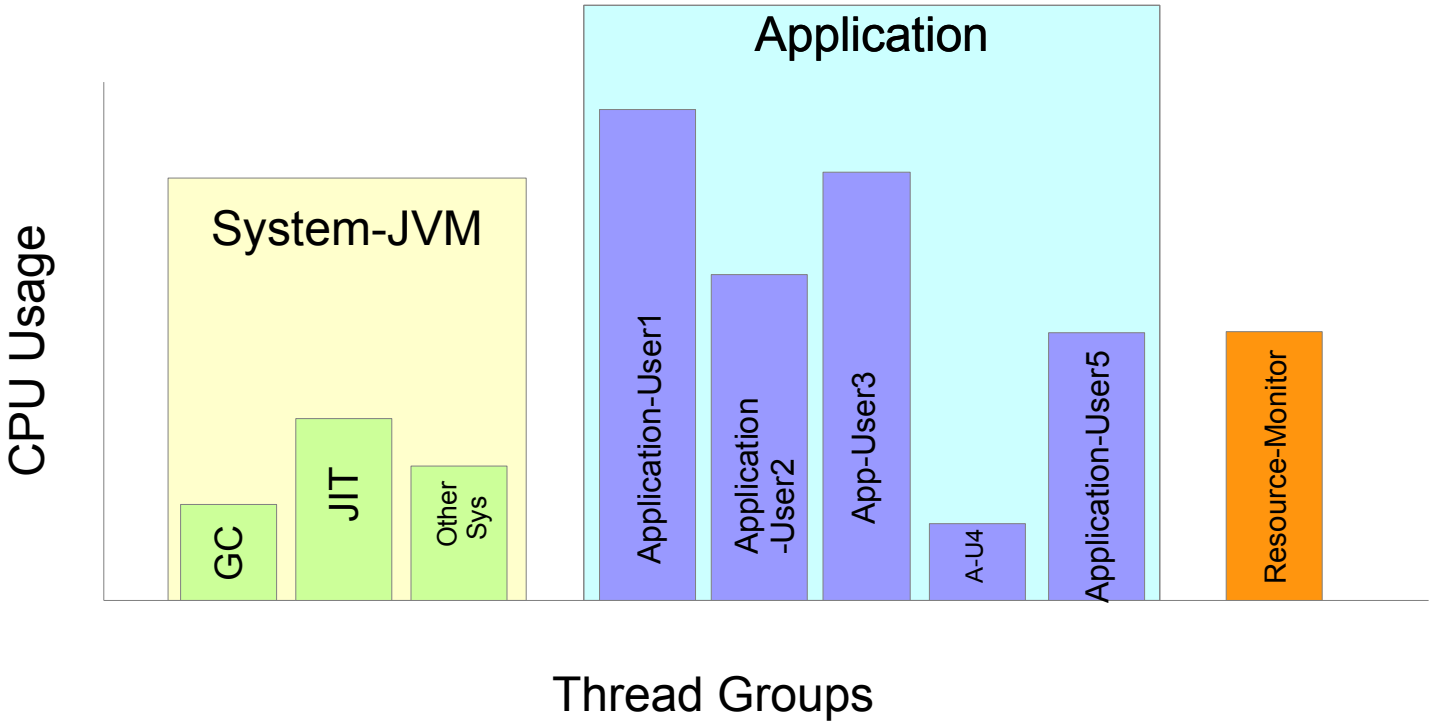




What is your application doing when Idle ?

- Does your application use lots of CPU even when Idle ?
- Historically poor Java idle behavior causes CPU burn
  - Inefficient CPU usage in the Cloud
    - Starves other VM's / JVM instances
  - Increased client costs for CPU Usage
    - Especially so on zSystems
- Memory hoarding of Idle JVM's.
  - Reduce memory footprint when idle.
    - “Boilerplate” applications in Bluemix that are never used, but are taking up lots of memory.
- First step is to be able to measure precise JVM CPU usage

Ability to categorize JVM threads for CPU usage



JvmCpuMonitorMXBean

com.ibm.lang.management

Interface JvmCpuMonitorMXBean

All Superinterfaces:
java.lang.management.PlatformManagedObject

```
public interface JvmCpuMonitorMXBean
extends java.lang.management.PlatformManagedObject
```

This interface provides APIs to obtain JVM CPU usage information in terms of thread categories. APIs are also available to get and set the thread category.

Method Summary

All Methods | Instance Methods | Abstract Methods

Modifier and Type	Method and Description
java.lang.String	getThreadCategory(long id) This function gets the current value of the thread category for the target thread.
JvmCpuMonitorInfo	getThreadsCpuUsage(JvmCpuMonitorInfo jcmInfo) This function updates the user provided JvmCpuMonitorInfo object with CPU usage statistics of the various thread categories.
int	setThreadCategory(long id, java.lang.String category) This function sets the thread category of the target thread.

JvmCpuMonitorInfo

Method Summary

All Methods | **Static Methods** | **Instance Methods** | **Concrete Methods**

Modifier and Type	Method and Description
boolean	**equals**(java.lang.Object obj)
static **JvmCpuMonitorInfo**	**from**(javax.management.openmbean.CompositeData cd) Receives a CompositeData representing a **JvmCpuMonitorInfo** object and attempts to return the root **JvmCpuMonitorInfo** instance.
long	**getApplicationCpuTime**() This method returns the total CPU usage for all application threads.
long[]	**getApplicationUserCpuTime**() This method returns an array of CPU usage for all user defined thread categories.
long	**getGcCpuTime**() This method returns the total CPU usage of all GC threads.
long	**getJitCpuTime**() This method returns the total CPU usage of all JIT Threads.
long	**getResourceMonitorCpuTime**() This method returns the total CPU usage for all threads of the "Resource-Monitor" category.
long	**getSystemJvmCpuTime**() This method returns the total CPU usage of the "System-JVM" category, which includes GC, JIT and other JVM daemon threads.
long	**getTimestamp**() This method returns the last sampling time stamp.
int	**hashCode**()
java.lang.String	**toString**() Text description of this **JvmCpuMonitorInfo** object.

Javacore info

```
1XMTHDSUMMARY  Threads CPU Usage Summary
NULL           =========================
NULL
1XMTHDCATINFO  Warning: to get more accurate CPU times for the GC, the option -XX:-ReduceCPUMonitorOverhead can be used
NULL
1XMTHDCATEGORY All JVM attached threads: 50.129351000 secs
1XMTHDCATEGORY |
2XMTHDCATEGORY +--Resource-Monitor: 9.637264000 secs
1XMTHDCATEGORY |
2XMTHDCATEGORY +--System-JVM: 1.104925000 secs
2XMTHDCATEGORY |  |
3XMTHDCATEGORY |  +--GC: 0.010265000 secs
2XMTHDCATEGORY |  |
3XMTHDCATEGORY |  +--JIT: 0.486622000 secs
1XMTHDCATEGORY |
2XMTHDCATEGORY +--Application: 39.387162000 secs
2XMTHDCATEGORY |  |
3XMTHDCATEGORY |  +--Application-User1: 9.337640000 secs
2XMTHDCATEGORY |  |
3XMTHDCATEGORY |  +--Application-User2: 9.450345000 secs
2XMTHDCATEGORY |  |
3XMTHDCATEGORY |  +--Application-User3: 9.443287000 secs
2XMTHDCATEGORY |  |
3XMTHDCATEGORY |  +--Application-User4: 4.235790000 secs
2XMTHDCATEGORY |  |
3XMTHDCATEGORY |  +--Application-User5: 4.413613000 secs
NULL
NULL           ------------------------------------------------------------------------
```

Use Cases for Thread-based CPU Usage

- Reporting transaction CPU usage
- Identifying "expensive" transactions
- Reporting JVM overhead over specific intervals
- Foundation for future work on tracking idle behaviour
- …

- Available in IBM Java SDK version 8 onwards

References

- http://www-01.ibm.com/support/knowledgecenter/SSYKE2_8.0.0/com.ibm.java.api.80.doc/com.ibm.lang.management/com/ibm/lang/management/JvmCpuMonitorMXBean.html
- http://www-01.ibm.com/support/knowledgecenter/SSYKE2_8.0.0/com.ibm.java.lnx.80.doc/diag/appendixes/cmdline/xxreducecpumonitoroverhead.html?lang=en

-Xtune:virtualized

- Available from Java 7 SR4 onwards.
- Reduces JVM CPU consumption when Idle (mostly JIT).
- Needs a large shared class cache to maintain peak performance.
- AOT space in the Shared Class Cache (SCC) must not be capped.

Summary

- Virtualization layers hide “real” resource information.
- JVM needs to know the underlying architecture at start time.
  - It then needs to periodically monitor for any changes.
- Applications can make use of the MBeans to do the same.
- Use only as much memory as you need.
  - Make use of SoftMX to reduce the heap dynamically.
- Idle detection and deep sleep helps cut costs.
  - Use MXBean to monitor Idle behaviour.

Copyright and Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., and registered in many jurisdictions worldwide.

Other product and service names might be trademarks of IBM or other companies.

A current list of IBM trademarks is available on the Web – see the IBM “Copyright and trademark information” page at URL: www.ibm.com/legal/copytrade.shtml