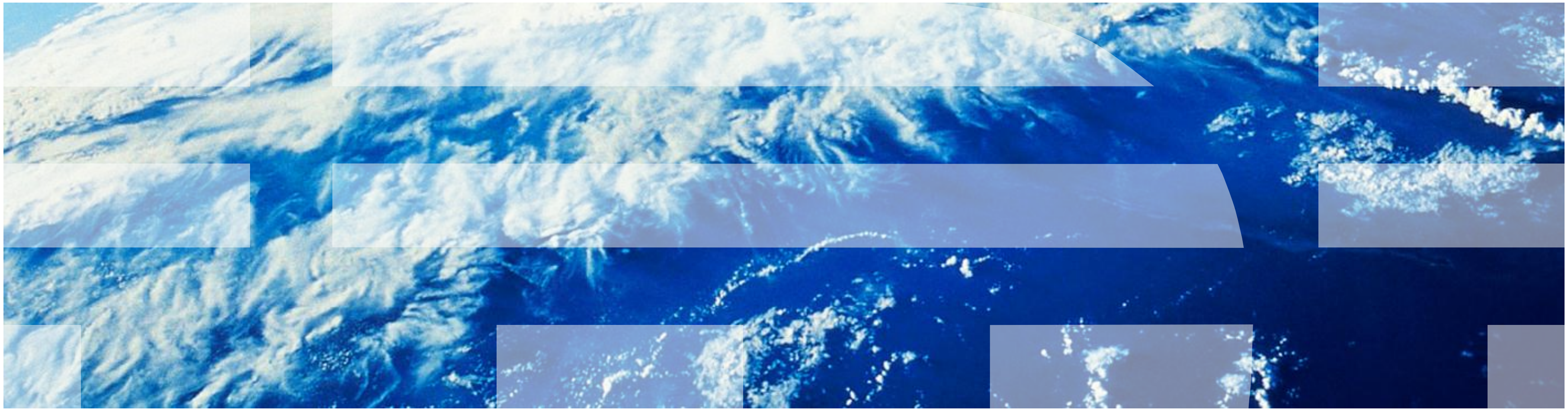
Optimize Java for Cloud

San Hong Li
Technical lead of Java for multi-tenancy



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

Introduction to the speaker

- Started from improving runtime security on Expeditor(OSGi-based and Eclipse-based platform) and progressed to working on the development of IBM's Java Virtual Machine in 2010.
- Recent work focus:
  - Java Virtual Machine improvements for ‘cloud’
    - Multi-tenancy technology
    - Footprint and performance
- My contact information:
  - mail: lisanh@cn.ibm.com
  - weibo: sanhong_li

Agenda

- Cloud computing
  - Basics & Definition
  - Use cases & Challenges
- IBM JDK support for virtualization
  - Cross-guest class sharing
  - Softmx
  - JMXBean for virtualization
- IBM JDK support for multi-tenancy

Cloud Computing Is…

- A style of computing in which dynamically scalable and often virtualized resources are provided as a service over the Internet. Users need not have knowledge of, expertise in, or control over the technology infrastructure in the “cloud” that supports them.

- Wikipedia

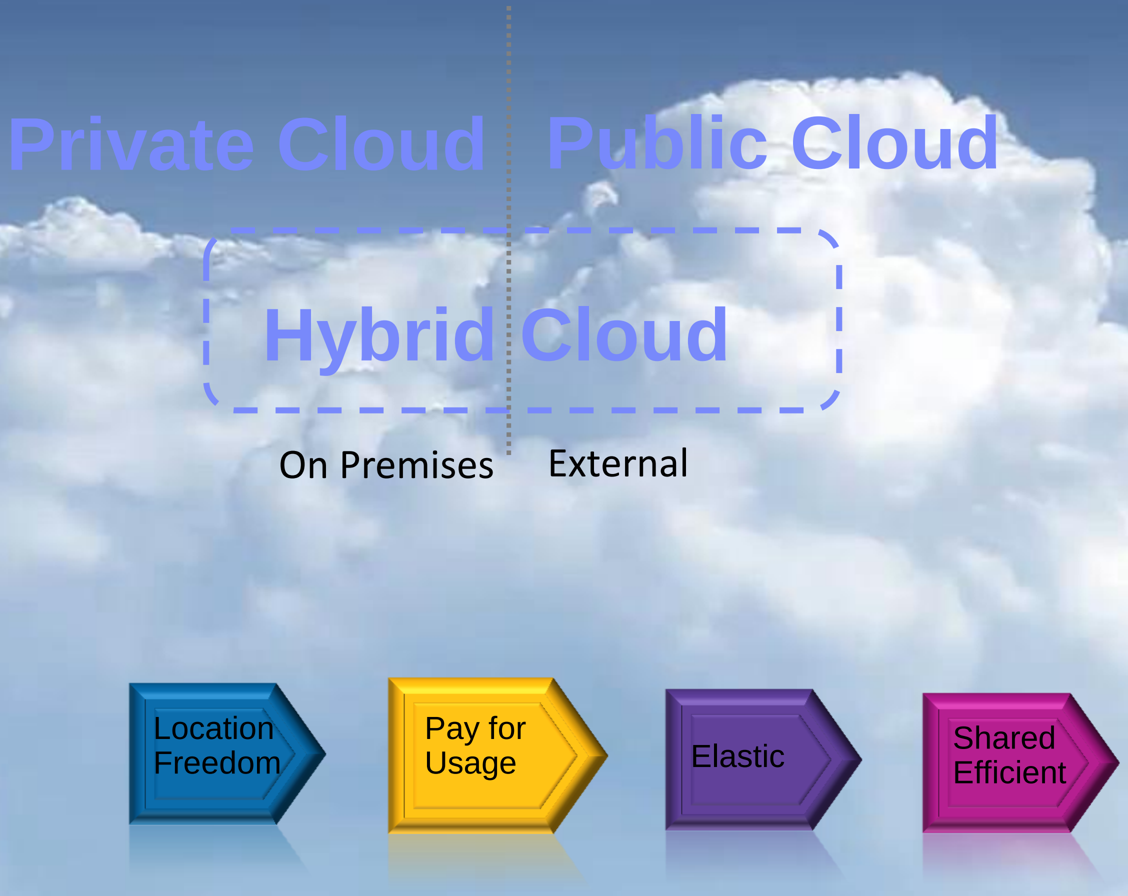
Cloud Computing Is…

- Computing resources are provided as a service
  - Infrastructure as a Service – IaaS
  - Platform as a Service – PaaS
  - Software as a Service – SaaS
- Computing Resources

Infrastructure	***Platform***	***Software***
Processor	LAMP Stack	Email
Memory	**JVM**	CRM System
Storage	Python VM	ERP System
Network	MapReduce	SCM System
amazon web services slicehost	MOSSO the rackspace cloud	salesforce.com Success On Demand. Gmail by Google BETA SAP

- As a Service
  - Pay-as-you-go or Subscription

Cloud Types



Private Cloud

Public Cloud

Hybrid Cloud

On Premises

External

Location Freedom

Pay for Usage

Elastic

Shared Efficient

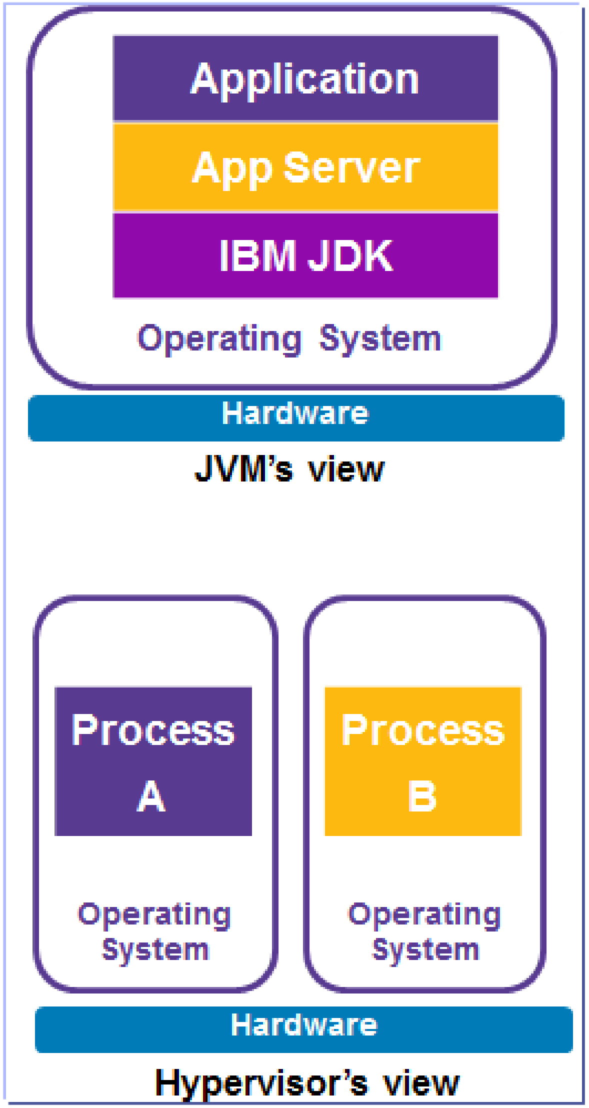
Enables innovative business models by providing IT services from a dynamic infrastructure faster, simpler, and cheaper

Scenarios where ‘Cloud’ works well

- **Server Consolidation**
  - Cost savings, merging low resource utilization systems
- **Evaluation**
  - Easy demonstration & test platform
- **Isolation**
  - Protect sensitive processes in their own OS
- **Shared Systems**
  - Development environments with varied workloads
- **Legacy Support**
  - Replace obsolete hardware via emulation

JVM vs. Hypervisor

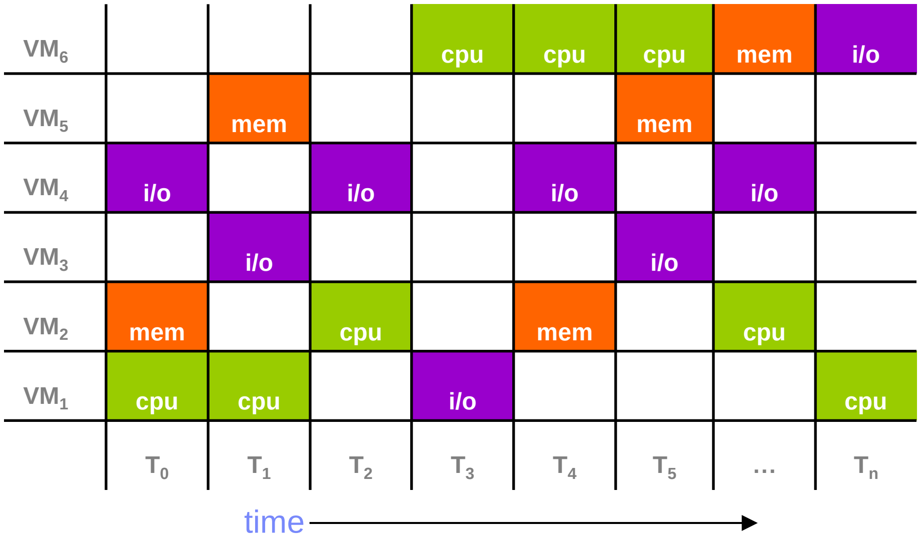
- JVM is managing resources (primarily memory) on behalf of the user
- JVM built assuming it has full control over resources for the duration of the process
- Hypervisor managing resources used by each Guest
- Hypervisor assumes it has full control over resources provided to each Guest
- Both are attempting to manage resources (without consulting each other)



Exploiting Idleness can yield better utilization

- **Baseline**: Today's JVM makes static decisions about the environment & never changes
  - Example: Heap sizes are a percentage of installed RAM
  - Example: Helper thread counts are a percentage of installed cores
- **Impact**: Static decisions hurt Java elasticity, makes it hard to shift resources toward need.
- **Example**: Consider dynamic workload spread across 6 VMs on a machine that has 1 unit of each resource to share:

cpu mem i/o



	T_0	T_1	T_2	T_3	T_4	T_5	…	T_n
VM_6				cpu	cpu	cpu	mem	i/o
VM_5		mem				mem		
VM_4	i/o		i/o		i/o		i/o	
VM_3		i/o				i/o		
VM_2	mem		cpu		mem		cpu	
VM_1	cpu	cpu		i/o				cpu

Challenge: make workload 'fit' (it should)

- Be able to measure reliably
- Make the JVM elastic enough to respond
- Make middleware elastic too
- Avoid JVM startup requirement for elasticity

Cloud Challenges

- **Hypervisor will attempt to shift resources toward need**
  - **Memory**: force action in guest via ballooning
  - **CPUs**: either by doling out a smaller slice, or ‘hotplug’ changes
  - Naïve JVM behaviour based on static decisions can foil the hypervisor’s efforts
- **Lesson: ‘Dynamic decisions are the new normal’**
  - # of installed CPUs or memory can change mid-run
  - Static decisions made at start-up are no longer good enough
- **Lesson: ‘Conserve. Every byte of memory or CPU cycle we waste could have been used to help others’**
  - Idle behaviour is critical (you can help here)

IBM JDK: Innovate Java for Cloud

- **Reduce** the Java appetite for resources
- **Reuse** artifacts between Java invocations
- **React** to changes in hypervisor, O/S, and other Java VMs

IBM JDK: Innovate Java for Cloud

Multitenancy JDK(XiHu)
- Sharing JVM process between multiple applications using isolates

Cross-Guest Class Sharing
- Share classes cache cross-guest

Hypervisor & Guest MxBean
- Identify & query the underlying hypervisor
- Provide enhanced O/S performance info

Automatic ‘softmx’
- Reduce heap size under memory pressure

Delivering on the Cloud Promise

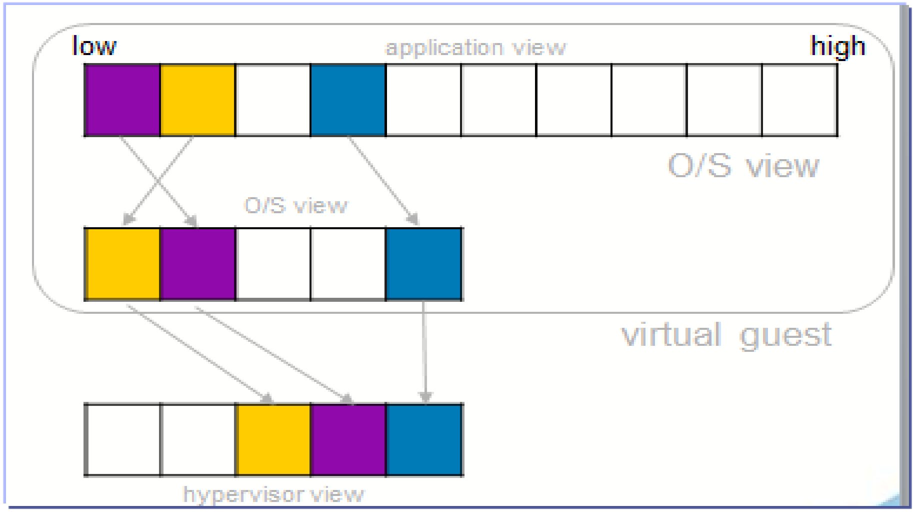
- **Over-commit is the golden solution**
  - –2:1 or better over-commit ratios possible with careful tuning

- **Memory Over-commit**
  - –**Swapping** – increase memory by using disk
  - –**Ballooning** – trick Guest into choosing what to swap
  - –**Page Sharing** – find identical pages and share

- **Exploiting idleness**
  - –Collocating workloads with different periods of demand

Memory Basics

- **Applications** work at the level of pages in a virtual address space
- **Operating system** maps the virtual address space onto pages (vRAM)
- **Hypervisor** adds another layer of mapping from vRAM to physical RAM (pRAM)
- Each layer adds:
  - –some overhead
  - –challenges for passing intent
  - –different views of state

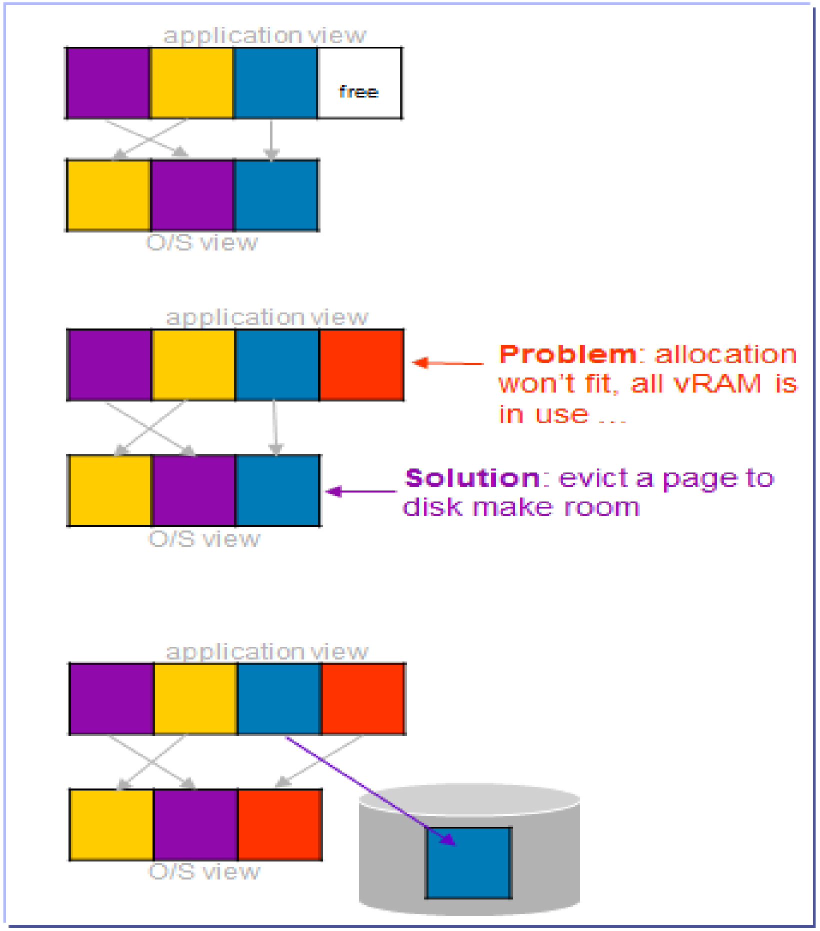


Swapping Trades CPU & I/O for Memory

- **Before**: vRAM is full, let's try to allocate another page (in red)

- **Decision**: vRAM is full, we'll need to evict a page to satisfy the request

- **After**: Allocation satisfied, one page (blue) moved to disk
  - Virtualized I/O can be slow
  - Paging is real work

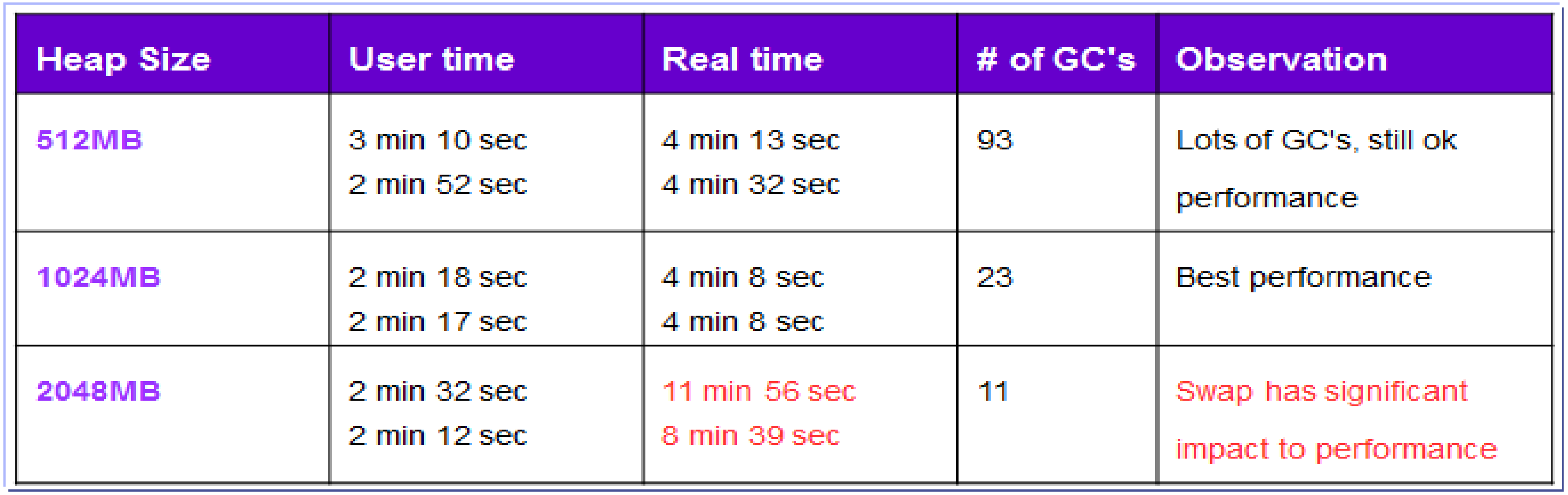


Problem: allocation won't fit, all vRAM is in use …

Solution: evict a page to disk make room

Swapping Can Severely Impact Java Performance

- **Experiment: How badly does swapping hurt?**
  - –Benchmark: Heap dump analysis – memory, I/O and processor intensive
  - –Lower times are better, two concurrent runs
  - –Hardware: Thinkpad W500 2.53GHz w/ **4GB** RAM

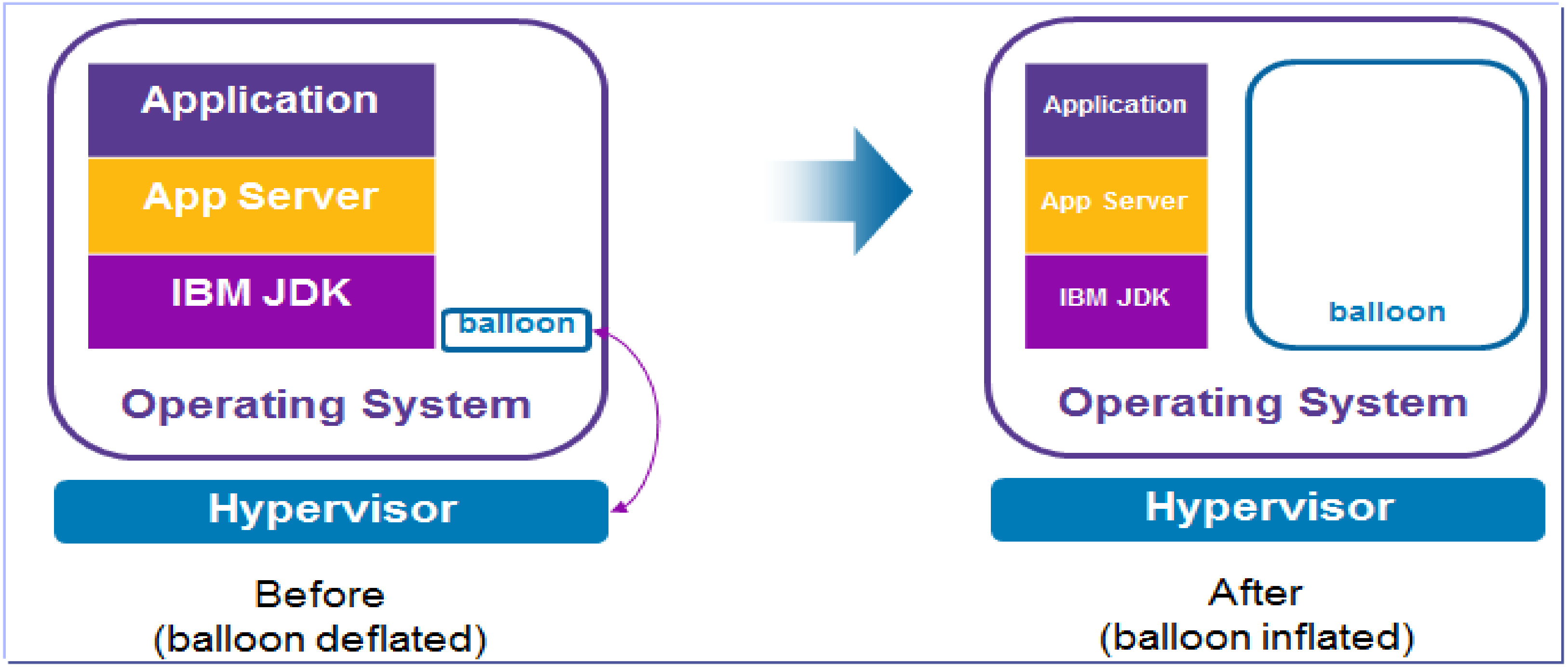


Heap Size	User time	Real time	# of GC's	Observation
512MB	3 min 10 sec 2 min 52 sec	4 min 13 sec 4 min 32 sec	93	Lots of GC's, still ok performance
1024MB	2 min 18 sec 2 min 17 sec	4 min 8 sec 4 min 8 sec	23	Best performance
2048MB	2 min 32 sec 2 min 12 sec	11 min 56 sec 8 min 39 sec	11	Swap has significant impact to performance

More on Java performance tuning
@Room 403 : Wed 10:15 - my JavaOne session

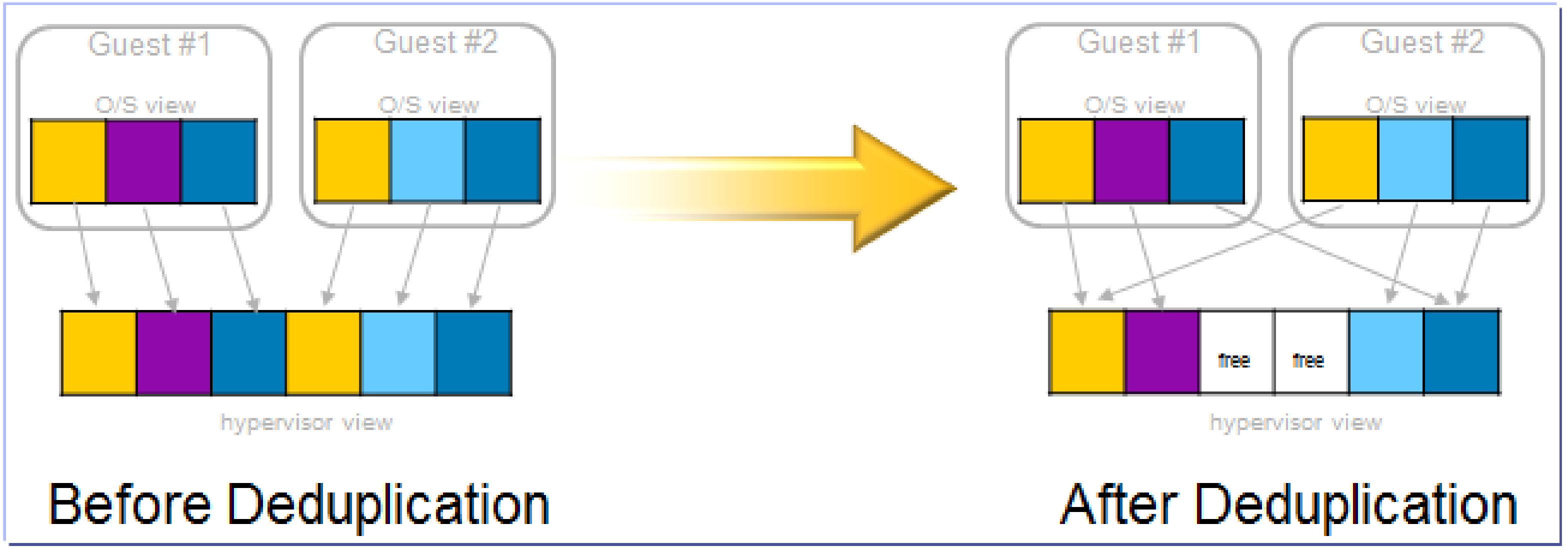
Hypervisor Trick : Ballooning

- **Encourage the operating system to move pages to disk**
- **Size of balloon controlled by the hypervisor**
  - No need to back the balloon with physical RAM
  - May result in paging to disk (swap), or dropping of buffers



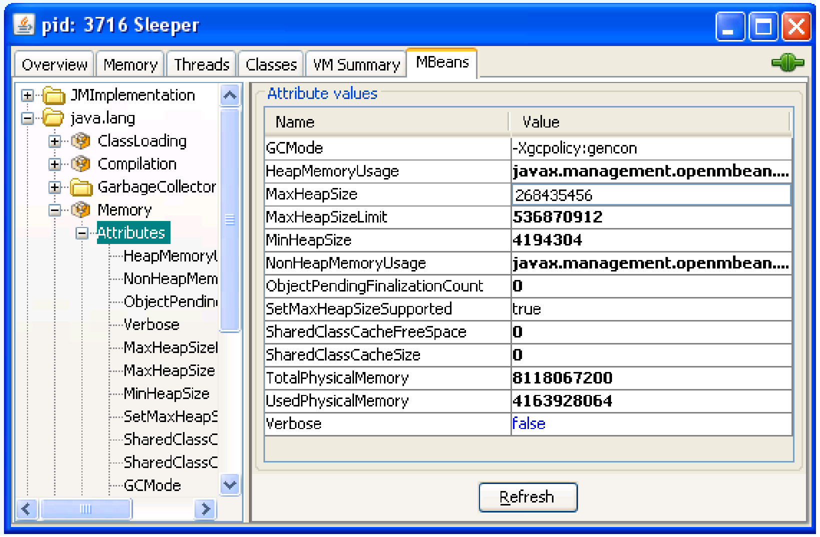
Saving Memory via Page Sharing

- **Share memory pages with identical content across guests**
  - –Use copy-on-write behavior if pages change
  - –Sharing Opportunities: shared libs, some JVM artifacts
  - –“Rebate” model – pay up front, refund some time later

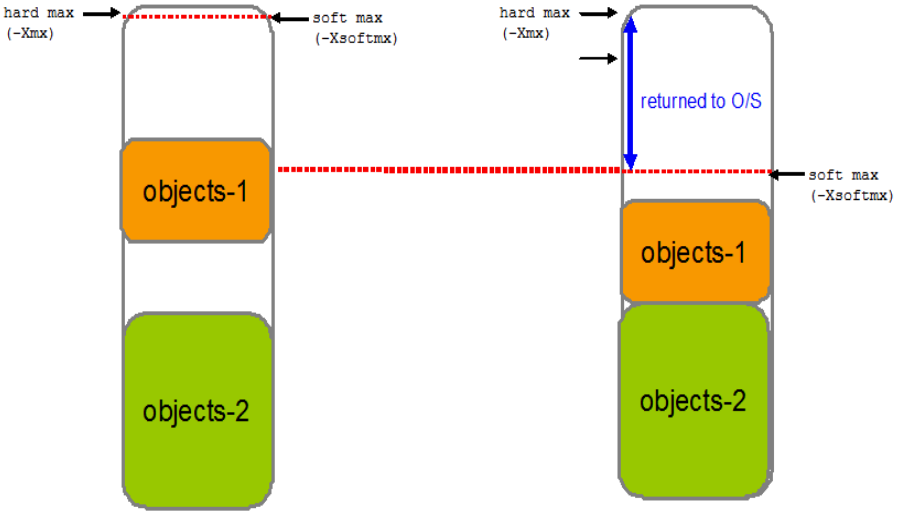


Dynamic Heap Adjustment (softmx) : Core function

- **-Xmx**
  - fixed at startup
- **-Xsoftmx**
  - can be set dynamically through JMX
  - <= -Xmx
  - 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



Name	Value
GCMode	-Xgcpolicy:gencon
HeapMemoryUsage	javax.management.openmbean....
MaxHeapSize	268435456
MaxHeapSizeLimit	536870912
MinHeapSize	4194304
NonHeapMemoryUsage	javax.management.openmbean....
ObjectPendingFinalizationCount	0
SetMaxHeapSizeSupported	true
SharedClassCacheFreeSpace	0
SharedClassCacheSize	0
TotalPhysicalMemory	8118067200
UsedPhysicalMemory	4163928064
Verbose	false



Am I Real or Virtual ?!

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

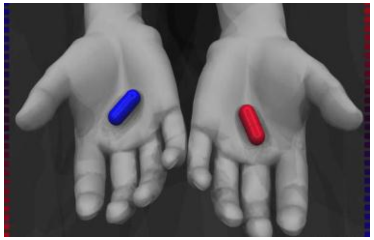
A. Mostly No. Only a small class of applications benefit

- Load Balancers
- Monitoring tools
- Debuggers and RAS Tools

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

A. Absolutely Yes !

Virtualization Aware JVM: Benefits



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

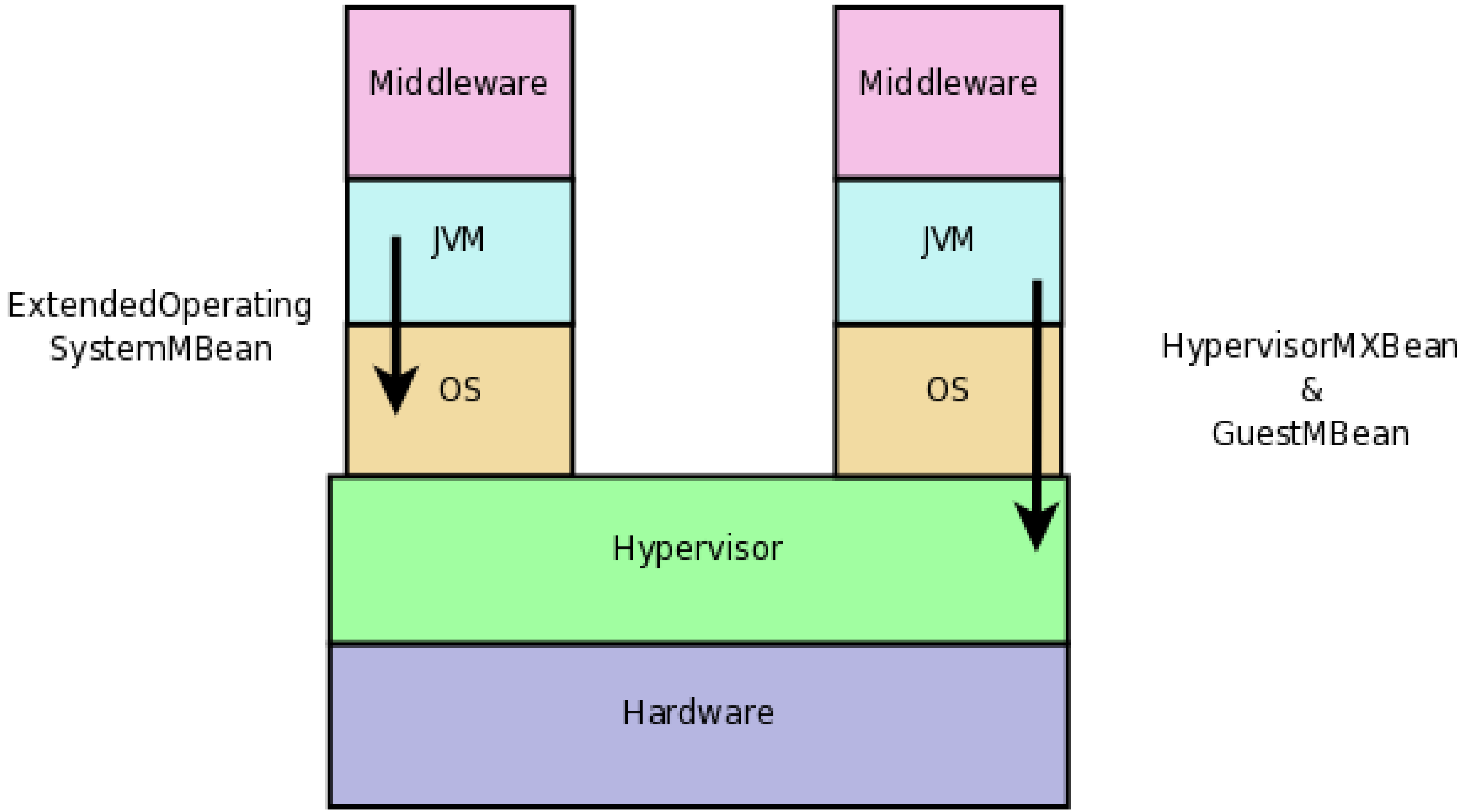
JMX Beans for Virtualization

- **ExtendedOperatingSystemMBean**
  - Extended OS usage statistics
    - Processor
    - Memory
  - OS Support: Aix, Linux, Windows and z/OS

- **HypervisorMXBean**
  - Detect if we are running on a hypervisor
  - Hypervisors Supported: z/VM, PowerVM, VMWare, KVM (x86 only), Hyper-V and OracleVM

- **GuestMBean**
  - Guest OS Statistics as seen from the hypervisor
  - Usage Statistics
    - Processor
    - Memory
  - Aix & Linux on PowerVM, Linux and Windows on VMWare, KVM, Hyper-V and OracleVM, z/OS & zLinux on z/VM

JMX Beans for Virtualization



What is Multitenancy?

Wikipedia
The Free Encyclopedia

Multitenancy refers to a principle in software architecture where a single instance of the software runs on a software-as-a-service (SaaS) vendor's servers, serving multiple client organizations (tenants).

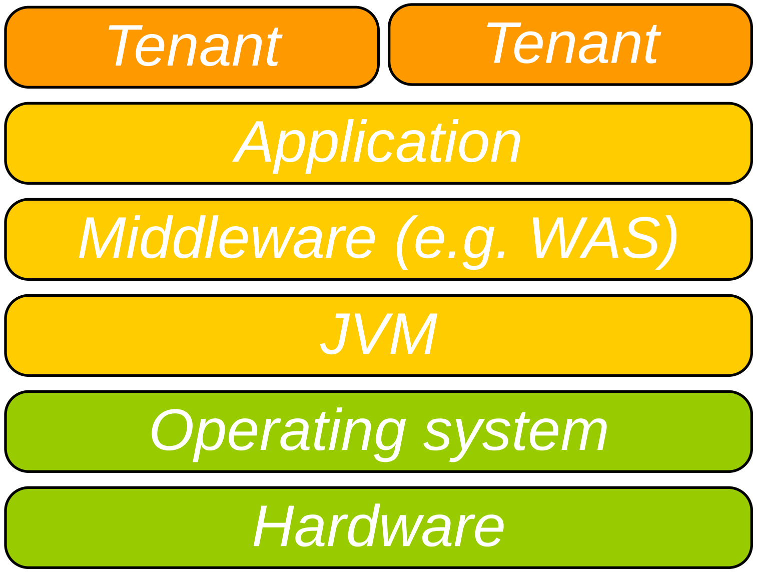
With a multi-tenant architecture, a software application is designed to virtually partition its data and configuration so that each client organization works with a customized virtual application instance.

Examples: ebay salesforce Google App Engine

- **Why you should care about multitenant applications:**
  - Cheaper to run: enables consolidation of IT resources
  - Data Aggregation: all the data is one place and easier to mine
  - Complexity: easier to manage upgrades and deployment of new tenants

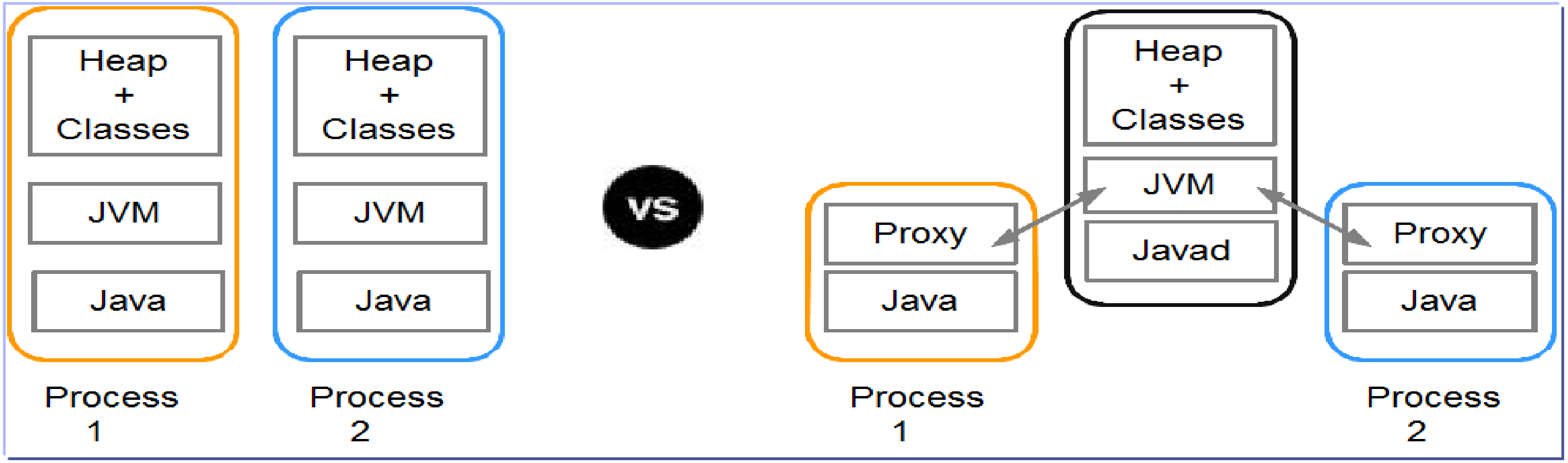
Multi-tenancy defined at different level

- Can make the MT at different stack level, from low HW level to high application level.
- Different levels of MT means the different deployment density
- Depending on the business scenarios, one may select different MT strategies



Multi-tenancy in jvm

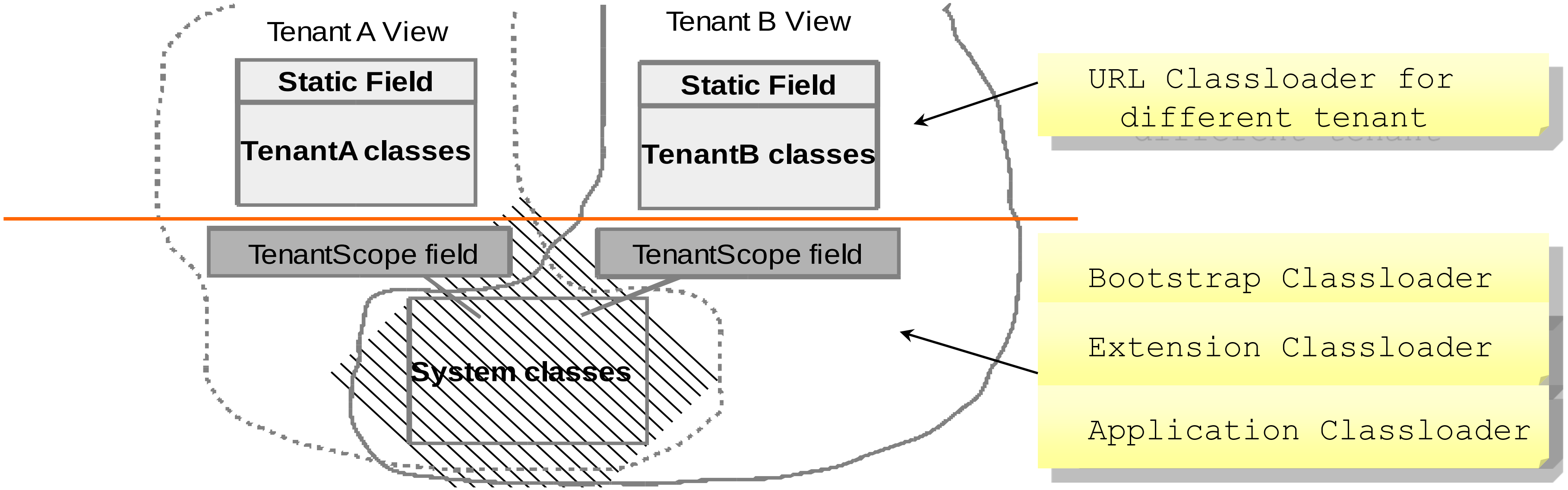
- **Allow for collocation of multiple Java applications in a single instance of JVM**
  - *Isolate application from one another and each "thinks" it has the whole VM all to itself.*
  - *Share metadata aggressively and transparently, such as:*
    - *bytecodes of methods*
    - *GC*
    - *JIT*



More on Multi-tenant JVM
@Room 430 : Wed 9:00 - my JavaOne session

Anatomy of multi-tenant JRE

- Making all static fields in the classes loaded by the bootstrap, extension and app classloaders as “@TenantScope”
- Loading each application with a separate classloader
- Changing the class library to be tenant aware, such like
  - System properties
  - Standard in/out/err
  - System.exit



Thanks!

Your questions?