



# Performance Best Practices for WebSphere Application Server Version 6.1 on Sun Solaris 10

**Session Number: 2447A**

**Session Time:**

**Thu, 10/Apr, 04:45 PM - 06:00 PM**

**Location: MGM Grand - Room 118**

**Albert Leigh, Solution Architect**

**Dileep Kumar, Sr. Staff Engineer**

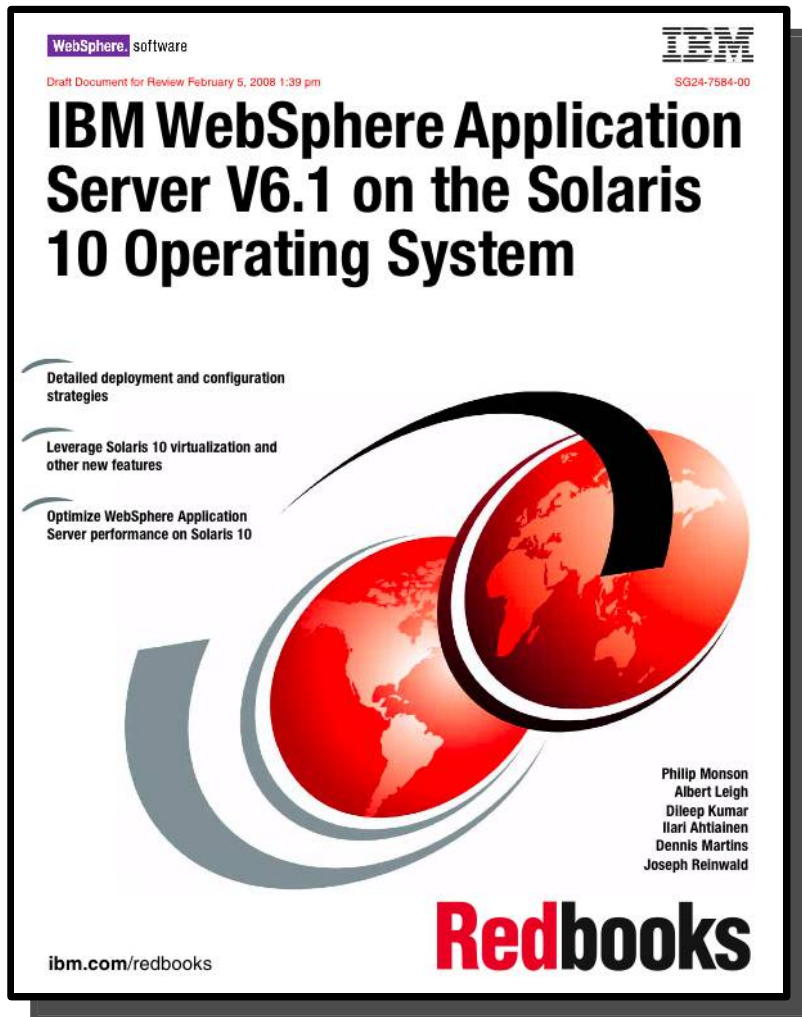
**Sun IBM WebSphere Team, Sun Microsystems, Inc.**



# Agenda

- The Redbook
- Background
- Performance Dependencies
- Performance Tuning
  - > WebSphere
  - > Java
  - > Solaris
  - > Systems
- Challenges on different H/W architecture

# The Redbook



Comprehensive material, a result of joint Sun and IBM collaboration

- > Detailed deployment topologies
- > Tools/Techniques described with samples
- > Performance Management explained in detail
- > 470 pages
- > Available at:

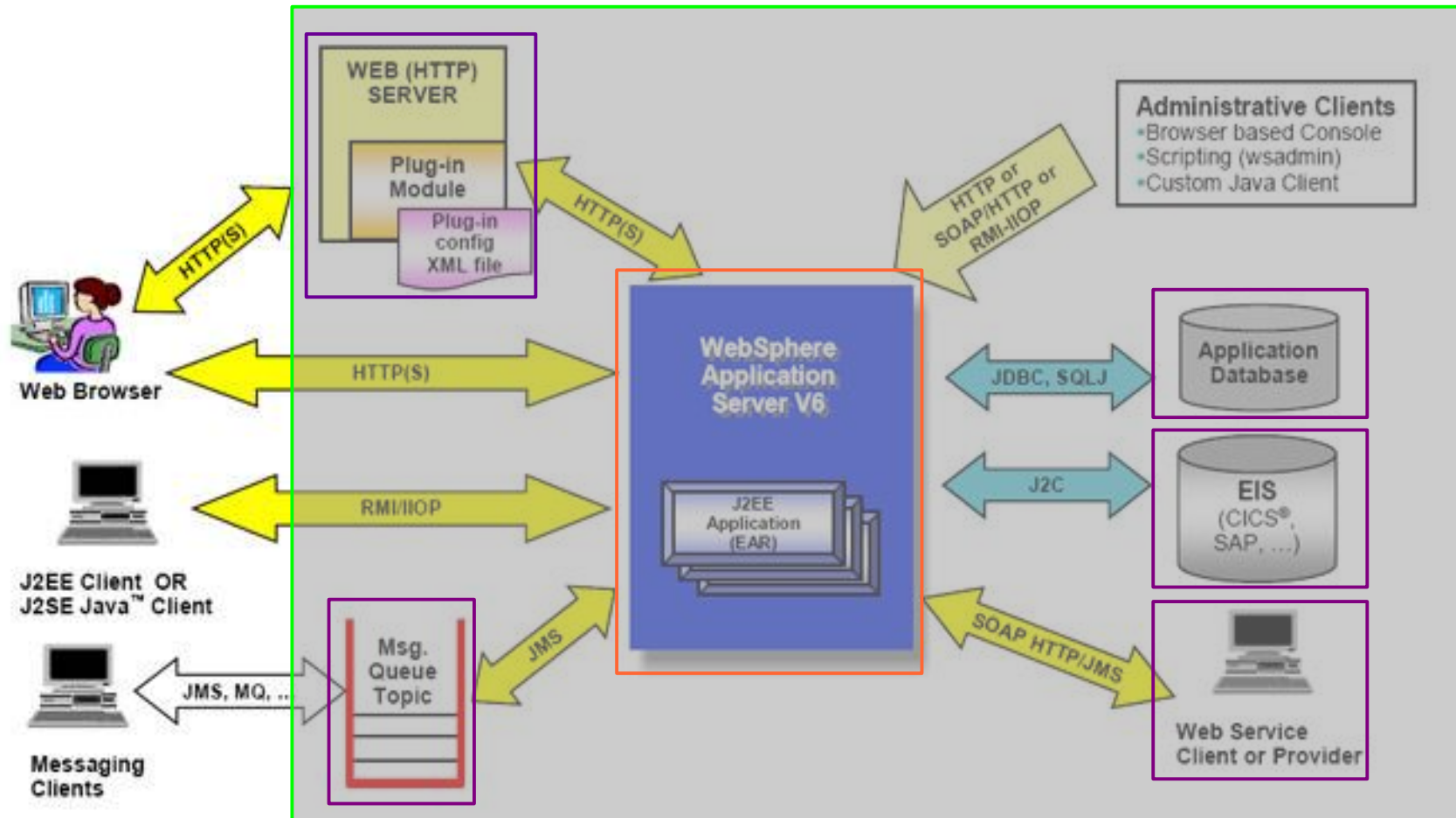
<http://redbooks.ibm.com/redpieces/abstracts/sg247584.html>

# Background

- Solaris is one of the many OS'es on which WebSphere Application Server (WAS) is available
  - > WAS features and functionality same
  - > Sun's JDK packaged and distributed by IBM
    - > Some components replaced with IBM implementation (ORB, XML, GSKit)
  - > Performance is dependent on JVM
- Considerations for
  - > WAS and its JVM tuning on Solaris
  - > Solaris on various hardware architectures

# Background

## Various Views of the System



Source: IBM

# Performance Dependencies

## WebSphere Performance Check List

- WebSphere & its Fix Pack versions
- The Java Platform
- Operating System
- Deployment topology
- Deployment hardware architecture
- Virtualization

# Performance Dependencies

## WebSphere & Fix Pack versions

- WebSphere Application Server versions have varying Java versions

| WAS    | Java EE | Java SE (JDK) |
|--------|---------|---------------|
| V6.1   | 1.4     | 5             |
| V6.0.2 | 1.4     | 1.4.2         |
| V5.1.1 | 1.3     | 1.4.2         |
| V5.1   | 1.3     | 1.4.1         |

Also available with 64-bit JVM

- Refresh Packs and Fix Packs can affect performance
- WAS is the core for other WebSphere products
  - > Portal Server, Process Server, Customer Center, etc.

# Performance Dependencies

## The Java Platform

- JDK bundled with WAS on Solaris (SPARC and x64) is Sun JDK with IBM modifications (i.e. ORB's, XML processors)
- JIT, JVM (Memory Management, GC) operations make significant impact on WebSphere performance
- With proper Fix Packs or JDK SR, JDK can be updated to later versions

| WAS Version | JDK Version on Solaris |
|-------------|------------------------|
| 6.1.0.x     | 1.5.0_06               |
| 6.0.2.x     | 1.4.2_08               |
| 6.0.1.x     | 1.4.2_07               |
| 6.0         | 1.4.2_05               |
| 5.1.1.x     | 1.4.2_05               |
| 5.1.1.x     | 1.4.2_11               |
| 5.1         | 1.4.1_05               |
| 5.0.2.x     | 1.3.1_08               |

\* Solaris Container Support



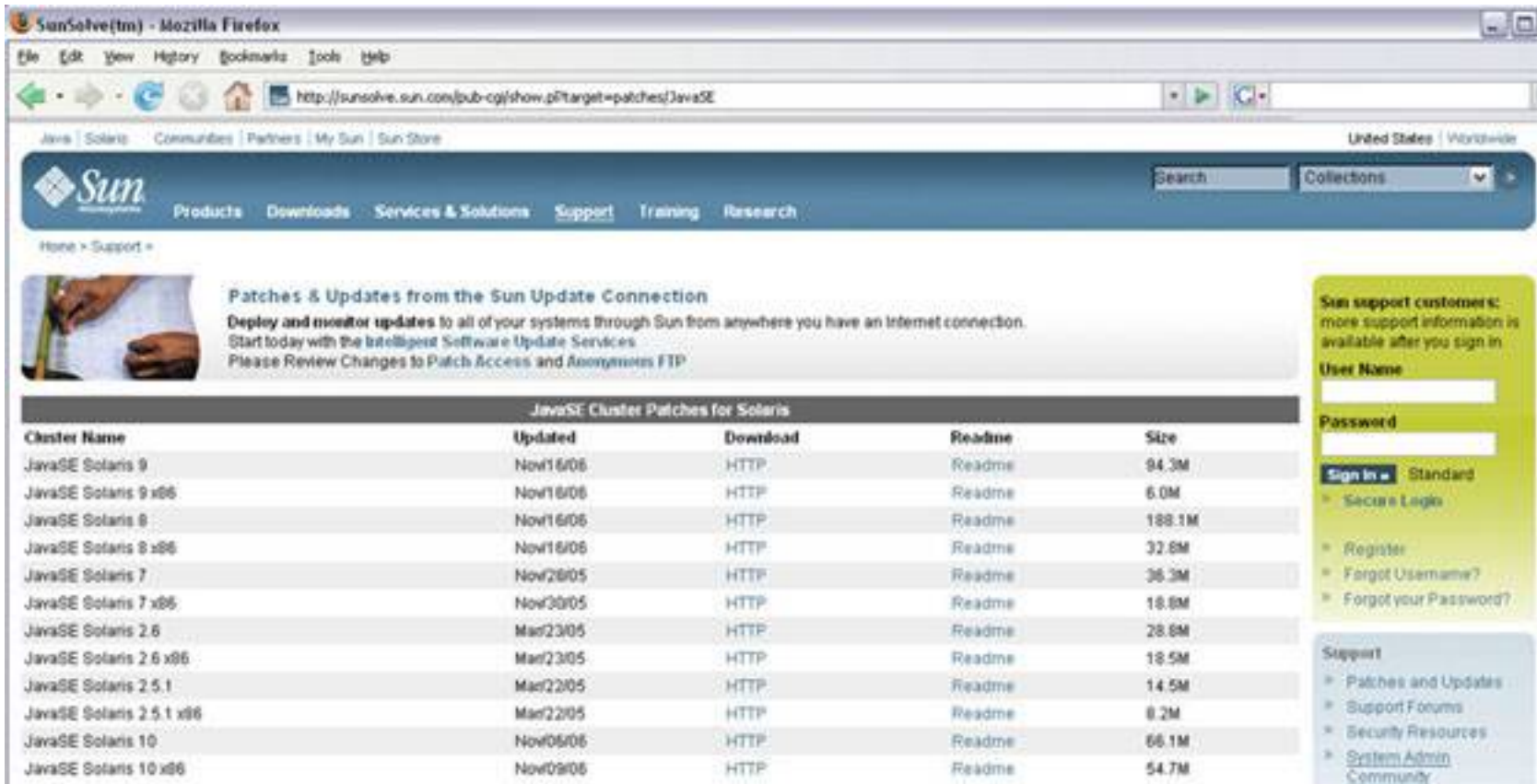
# Performance Dependencies

## Operating System

- Solaris versions
  - > Solaris 8, Solaris 9 or Solaris 10
  - > Solaris Updates (e.g. Solaris 10 11/06 vs 8/07)
  - > Which Binaries
    - > SPARC
      - WAS is available with 32-bit or 64-bit JVM
    - > x64
      - WAS is available with 64-bit JVM
- Keep system patches up to date
  - > Some patches are targeted for Java

# Solaris Patches for Java

- Keep the Java Patches for Solaris up-to-date
  - > <http://sunsolve.sun.com/pub-cgi/show.pl?target=patches/JavaSE>



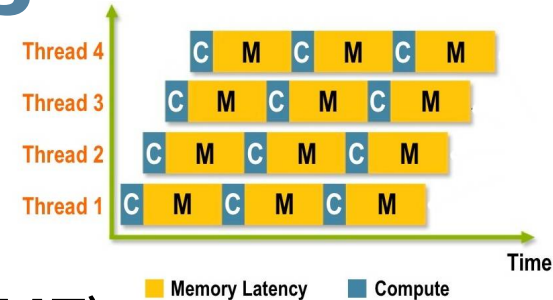
The screenshot shows the SunSolve website in a Mozilla Firefox browser. The page title is "SunSolve(tm) - Mozilla Firefox". The address bar shows the URL <http://sunsolve.sun.com/pub-cgi/show.pl?target=patches/JavaSE>. The page features a navigation bar with links for Java, Solaris, Communities, Partners, My Sun, and Sun Store. Below the navigation bar, there is a search bar and a "Collections" dropdown menu. The main content area is titled "Patches & Updates from the Sun Update Connection" and includes a brief description of the service. A table titled "JavaSE Cluster Patches for Solaris" lists various patches with columns for Cluster Name, Updated, Download, Readme, and Size. On the right side, there is a login section for Sun support customers, including fields for User Name and Password, and links for Sign In, Standard, Secure Login, Register, Forgot Username?, and Forgot your Password?. A "Support" section at the bottom right lists links for Patches and Updates, Support Forums, Security Resources, and System Admin Community.

| Cluster Name             | Updated  | Download | Readme | Size   |
|--------------------------|----------|----------|--------|--------|
| JavaSE Solaris 9         | Nov16/06 | HTTP     | Readme | 94.3M  |
| JavaSE Solaris 9 x86     | Nov16/06 | HTTP     | Readme | 6.0M   |
| JavaSE Solaris 8         | Nov16/06 | HTTP     | Readme | 188.1M |
| JavaSE Solaris 8 x86     | Nov16/06 | HTTP     | Readme | 32.8M  |
| JavaSE Solaris 7         | Nov26/05 | HTTP     | Readme | 36.3M  |
| JavaSE Solaris 7 x86     | Nov30/05 | HTTP     | Readme | 18.8M  |
| JavaSE Solaris 2.6       | Mar23/05 | HTTP     | Readme | 28.8M  |
| JavaSE Solaris 2.6 x86   | Mar23/05 | HTTP     | Readme | 19.5M  |
| JavaSE Solaris 2.5.1     | Mar22/05 | HTTP     | Readme | 14.5M  |
| JavaSE Solaris 2.5.1 x86 | Mar22/05 | HTTP     | Readme | 8.2M   |
| JavaSE Solaris 10        | Nov05/06 | HTTP     | Readme | 66.1M  |
| JavaSE Solaris 10 x86    | Nov09/06 | HTTP     | Readme | 54.7M  |

# Performance Dependencies

Deployment hardware architecture

- SPARC
  - > Chip Multi-threading Technology (CMT)
    - > UltraSPARC T1 / T2
      - Many hardware threads
      - Results in high scalability
      - WAS Thread Pools can be sized big
      - Parallel GC – reduce GC times and increase throughput
      - Concurrent GC -- improve determinism
      - Concurrent/Parallel GC



# Performance Dependencies

## > CMT Consideration

- Ergonomics – Default can have 32/64 GC Threads
- Processor binding to localize the memory
  - Core/thread affinity
  - Keep data close to processor
- Watch the total active thread count
- Keep it more than threads in the system
- Minimize writes to shared data as you can have more workers

# Performance Dependencies

Deployment hardware architecture

- SPARC
  - > Chip Multi-processing (CMP)
    - > UltraSPARC IV+ and SPARC64 VI processors
      - NUMA Aware allocators (Next release of WAS)
      - Faster threads
      - Larger Cache
      - Host multiple instances

# Performance Dependencies

Deployment hardware architecture

- AMD/Intel
  - > Opteron 64
  - > Intel Core2/Quad Core
  - > Intel Xeon 64
- Only 64-bit WAS is available
- On x64 platform 64-bit can be faster than 32-bit
- Offers virtually unlimited heap
- Requires very careful heap sizing

# Performance Dependencies

## Virtualization

- Solaris Containers
  - > OS virtualization based on single Solaris instance
  - > **Process space separation**
  - > Virtually no overhead
- Logical Domains
  - > Available on CMT based systems
  - > Firmware based Hypervisor
  - > Usual virtualization overhead
- Dynamic System Domains (high-end systems)
- Sun xVM (future)

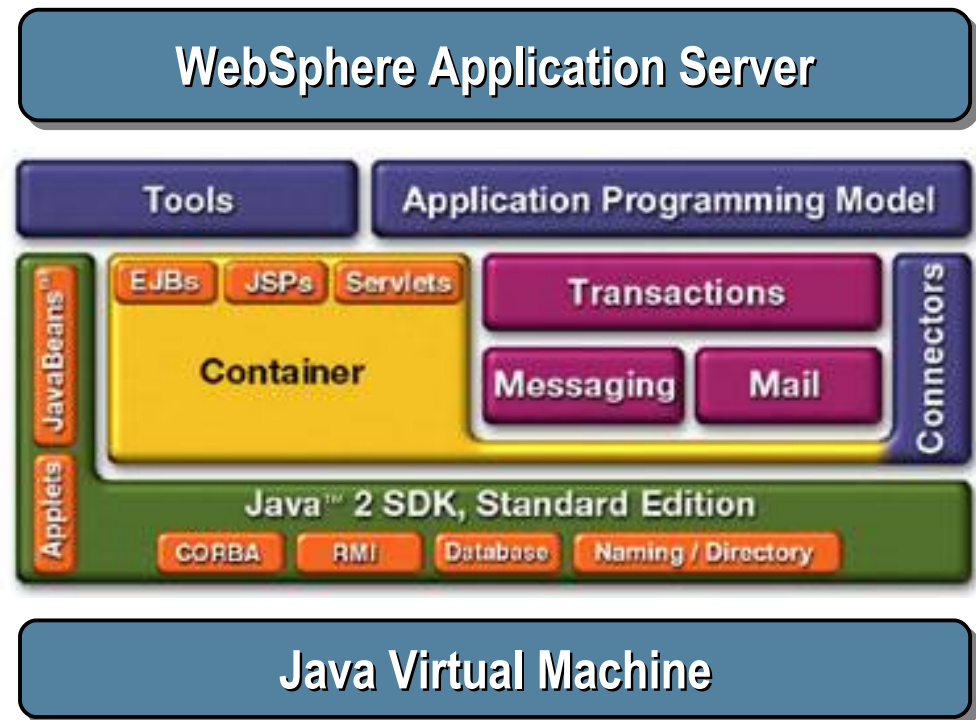
# Deployment Considerations

- Standard vs Virtualized Configurations
  - > Sun Virtualization Technologies
  - > Account for virtualization overhead
- Resource Control
  - > Manage/Allocate resources yourself to address the needs
- Solaris Schedulers (e.g. FSS, FX)
- Understanding of the underlying H/W architecture
  - > CPU (SPARC, Intel/AMD, CMT)
  - > Network devices

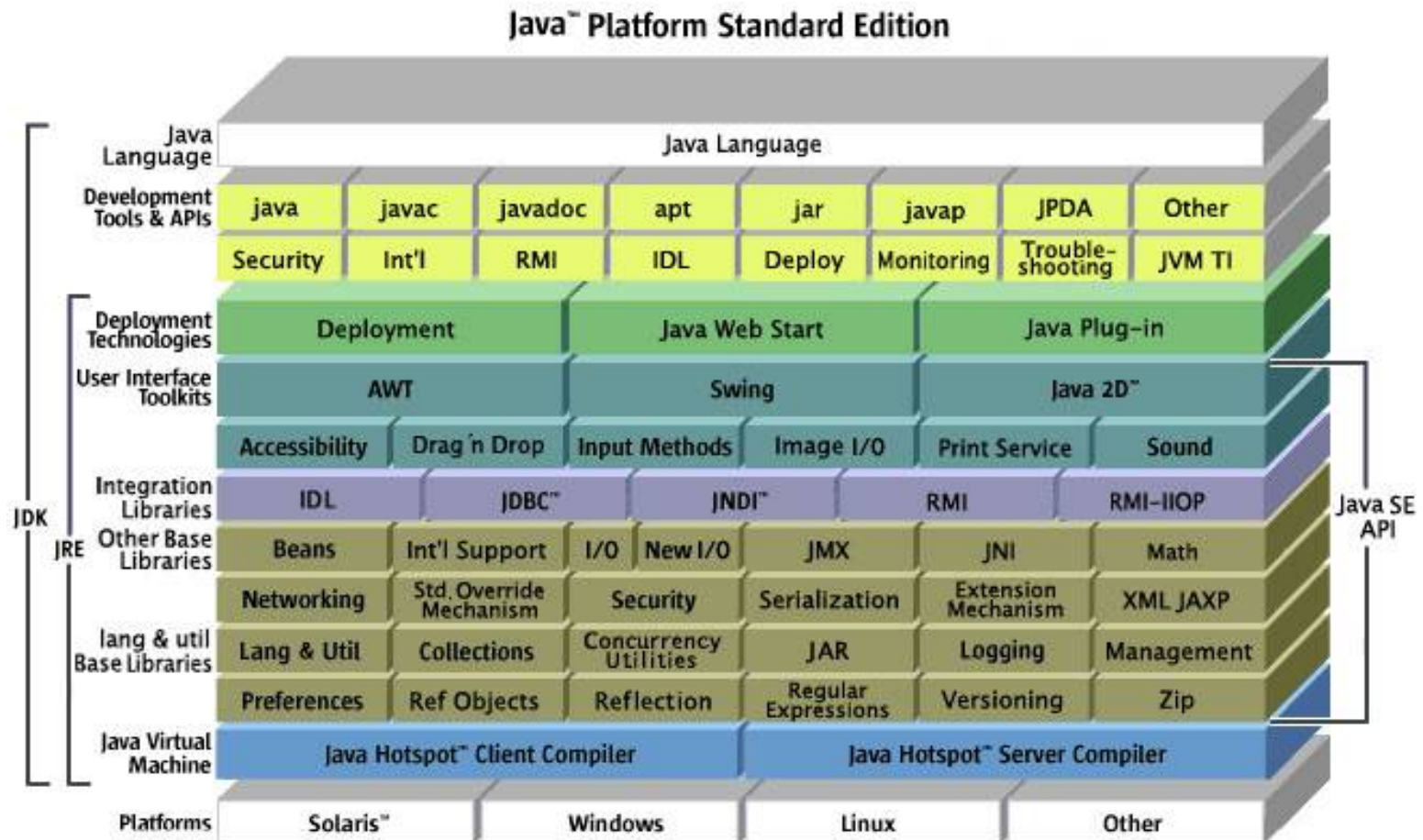


# Java Technology Foundation

- WebSphere relies on the Java Technologies
  - > Java Standard Edition
    - > JDK
    - > JRE
  - > Java Enterprise Edition
    - > Components, API
    - > Containers
    - > Messaging
    - > Transaction
    - > Web Services
    - > Application Programming Model

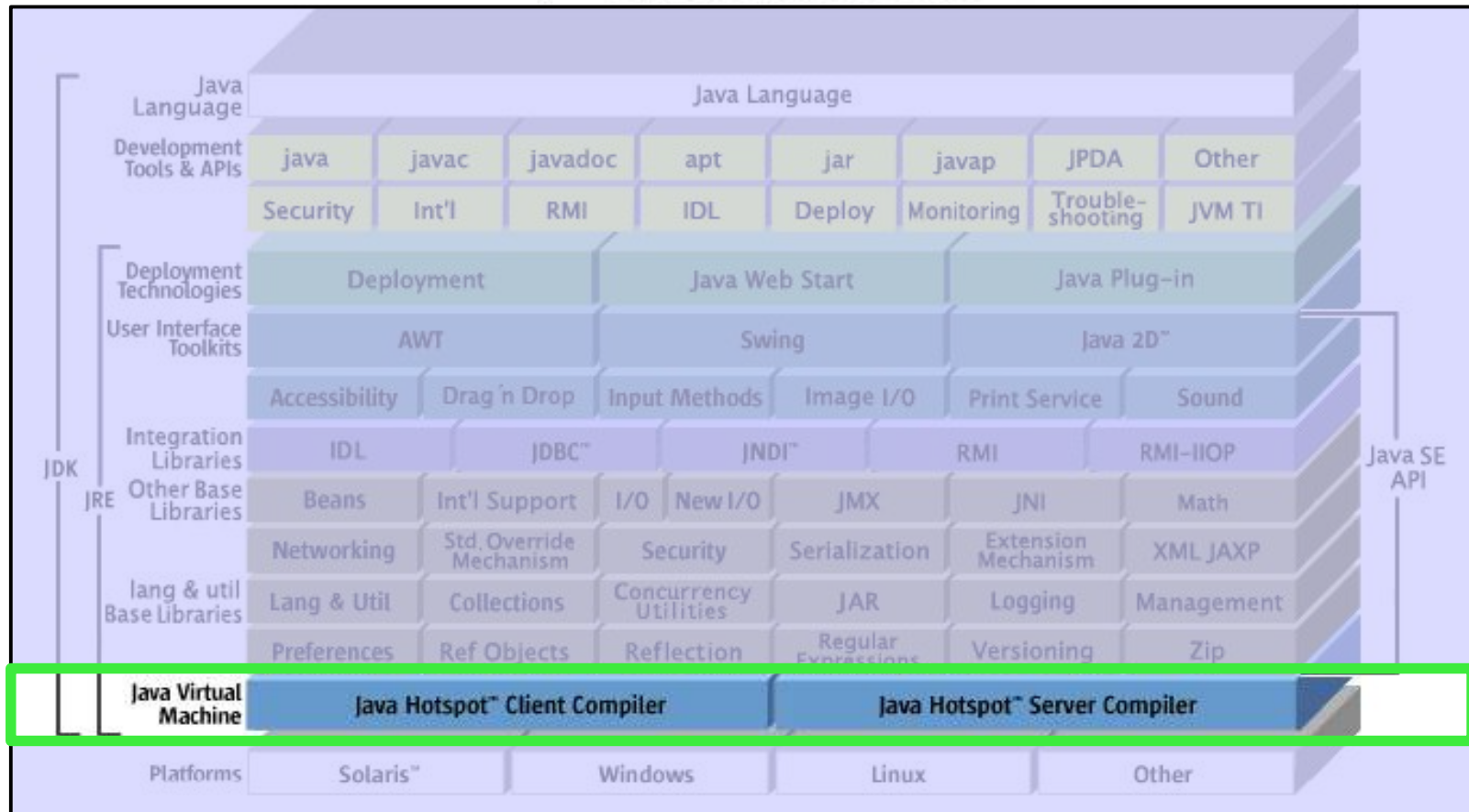


# Java Platform Standard Edition



# Java Virtual Machine (HotSpot)

Java™ Platform Standard Edition



# JDK 5 Improvements over JDK 1.4.2

- New Language Features:
  - > Generics, Autoboxing/Unboxing, Varargs
  - > Enhanced for loop (“foreach”), Type-safe enumerations
  - > printf, Static import, Metadata
- Performance, Scalability, Reliability
- JVM Ergonomics
- Class data sharing
- Monitoring & Manageability (JMX, MBean)
- New JVM profiling API
- Improved Diagnostic ability

# JVM Self Tuning (Ergonomics)

- JVM does self-tuning
- Just changing the heap could yield better performance
- Server class performance for Server class machines out of the box
  - > 2 CPU and 2 GB Ram
  - > Parallel (throughput) Collector (**-XX:+UseParallelGC**)
  - > Server compiler (**-server**)
  - > Max Heap (**-Xmx**)  $\frac{1}{4}$  Max Memory or 1GB
  - > Initial Heap (**-Xms**)  $\frac{1}{64}$  Max Memory or Max Heap

# JVM Self Tuning (Ergonomics)

- Maximum GC pause time goal
  - > **-XX:MaxGCPauseMillis=<n>**
  - > This is a hint, not a guarantee
  - > GC will adjust parameters to try and meet goal
  - > Can adversely effect application throughput
- Throughput goal
  - > **-XX:GCTimeRatio=<n>**
  - > GC Time : Application time = 1 / (1 + n)
  - > e.g. **-XX:GCTimeRatio=19** (5% of time in GC)

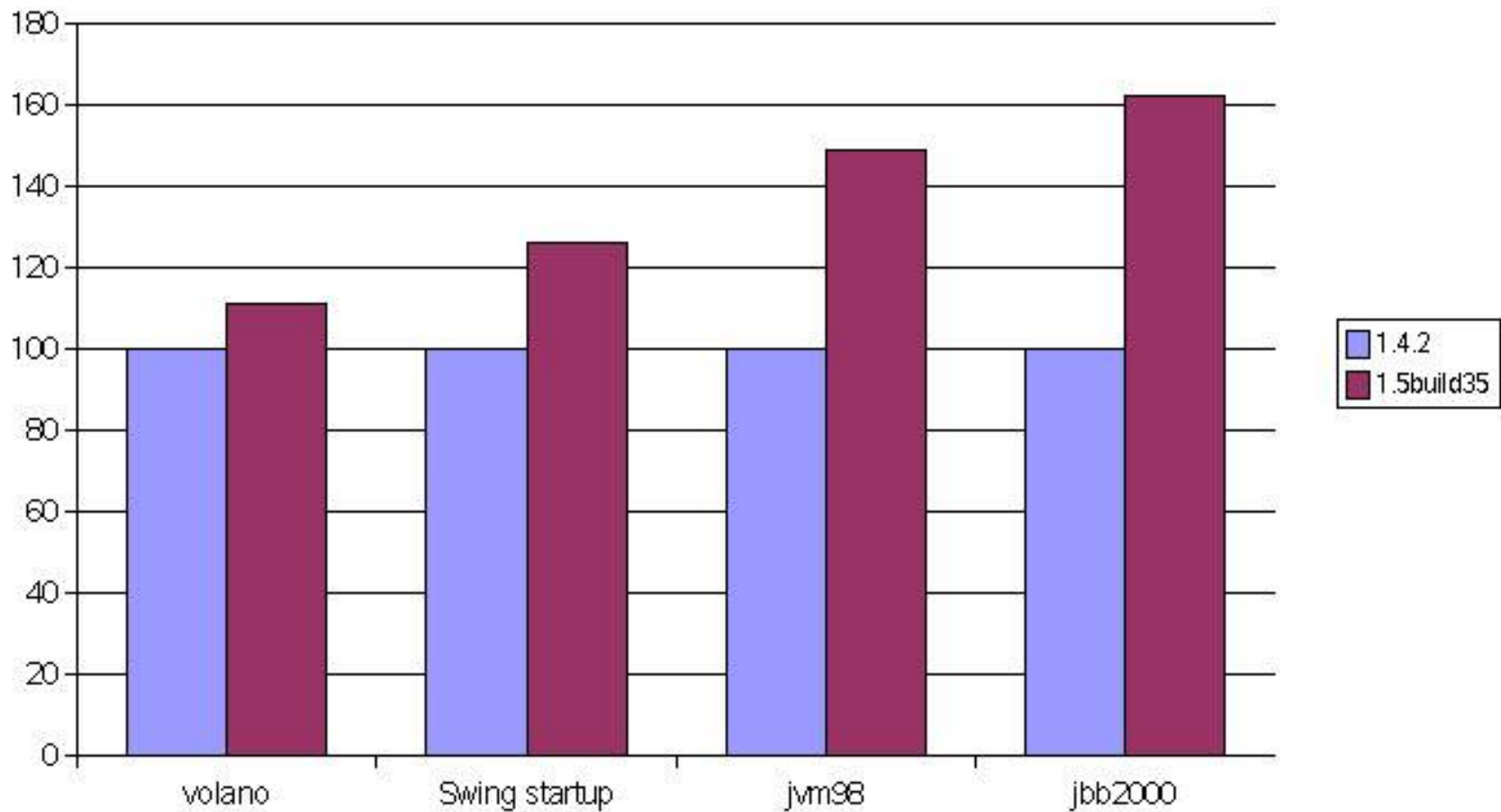
# JVM Self Tuning (Ergonomics)

## Challenges

- Multiple instance scenario
  - > JVM's are not aware of each other
- Only takes care of performance / Throughput
- Constrained environment need more manual tunings
- We do need “**-client**” VM in some cases
- Application requirements are different

# Java Performance Improvement

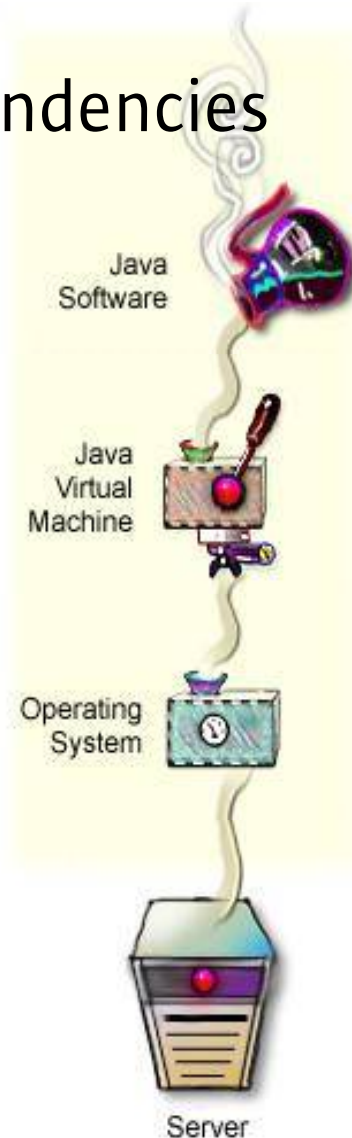
Solaris Sparc





# WebSphere Performance Issues

- Multi-tier environment with many inter-dependencies
- Performance issues can be caused by
  - > User applications
  - > Underlying JVM
  - > Underlying Operating Systems
  - > Infrastructure
    - > Network, Firewall, Storage, etc.
  - > Other dependent tiers
    - > Web Server, DB, ERP, Web Services
- Follow best practices to prevent from pitfalls



# Performance Tuning – Check List

## Read the Redbook and IBM's WAS Info Center Documentation

- > <http://www.redbooks.ibm.com/redpieces/abstracts/sg247584.html>
- > [http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.nd.doc/info/ae/ae/rprf\\_hotparameters.html](http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.nd.doc/info/ae/ae/rprf_hotparameters.html)

- Review HW/SW prerequisites for WAS
- Install the most current refresh pack, fix pack, and recommended interim fixes
- Check HW configuration and settings
- Review your application design
- Tune the Solaris Operating System

# Performance Tuning – Check List

- Tune the Java Virtual Machine settings
- Use the right JDBC driver eg. Type-4
- Provide sufficient Resources
- Follow John Stecher/IBM advice
  - > TSM-1999, TSM-1553
  - > plus our advice on Sun JVM tuning
- Tune related components, such as DB, MQ, etc.
- Disable functions that are not needed

# Performance Tuning – Solaris OS

Still using Solaris 8? Upgrade to Solaris 10

- Make use of LWP or Alternate Thread Library
  - > IBM Technote (SWG21107291) explains the details  
<http://www.ibm.com/support/docview.wss?rs=180&context=SSEQTP&uid=swg21107291>
- To examine, run the Solaris command
  - > **pldd <PID of WAS java process>**
- Check the output to verify path to **libthread.so**
  - > `/usr/lib/libthread.so` (VS)
  - > `/usr/lib/lwp/libthread.so`
- Set env **LD\_LIBRARY\_PATH** in **startServer.sh** or set the JVM argument **-XX:UseBoundThreads** for older than JDK 5

# Performance Tuning – Solaris OS

- Detail explanation of the Java and Solaris Thread Models is available:  
<http://java.sun.com/docs/hotspot/threads/threads.html>
- On Solaris 9, LWP or the Alternate Thread Library became standard
- Use of Alternative Heap Allocator:
  - > **LD\_PRELOAD=/usr/libumem.so**
  - > Then, run the WAS java process with this env setting
  - > <http://java.sun.com/performance/reference/whitepapers/tuning.html>

# Performance Tuning – Solaris OS

- Solaris kernel settings in `/etc/system` as described in the IBM Info Center

```
set shmsys:shminfo_shmmax = 4294967295
set shmsys:shminfo_shmseg = 1024
set shmsys:shminfo_shmmni = 1024
set semsys:seminfo_semaem = 16384
set semsys:seminfo_semmni = 1024
set semsys:seminfo_semmap = 1026
set semsys:seminfo_semmns = 16384
```

```
set semsys:seminfo_semmsl = 100
set semsys:seminfo_semopm = 100
set semsys:seminfo_semmnu = 2048
set semsys:seminfo_semume = 256
set msgsys:msginfo_msgmap = 1026
set msgsys:msginfo_msgmax = 65535
set rlim_fd_cur=1024
```

- For WAS v6.x, you only need: **set rlim\_fd\_cur=1024**
- The rests are legacy from embedded messaging in WAS v5.x
- If you need to change, use the `/etc/project` file. No reboot needed.

```
# projmod -sK 'process.max-file-descriptor=(privileged,1024,deny)' user.wasadm
```

# Performance Tuning – Solaris OS

- Tune the network
  - > TCP Tuning (nnd)
  - > Distribute network interrupts
  - > Use available tuning guidelines for you network driver
- Tune the File System
- General tuning on Solaris 10 :
  - > Use Fixed Priority Scheduling Class (FX) (+~10%) instead of the default FSS class
    - # /usr/bin/priocntl -s -c FX -m 59 -p 59 -i pid <PID of JAVA Process>
  - > Mutex Contention (mpstat “**smtx**” column)
  - > Multiple WAS instances for better performance

# Performance Tuning – Solaris OS

- Steps to identify hotlocks
  - > **plockstat(1M)**
  - > High Mutex Spin Count
  - > High Context switch rates
  - > Unable to leverage 100% CPU
- Disk I/O
- Interrupt processing (**intrstat(1M)**)
- Examine Run Queue (**vmstat(1M)**)
- Core utilization on CMT (**corestat**)

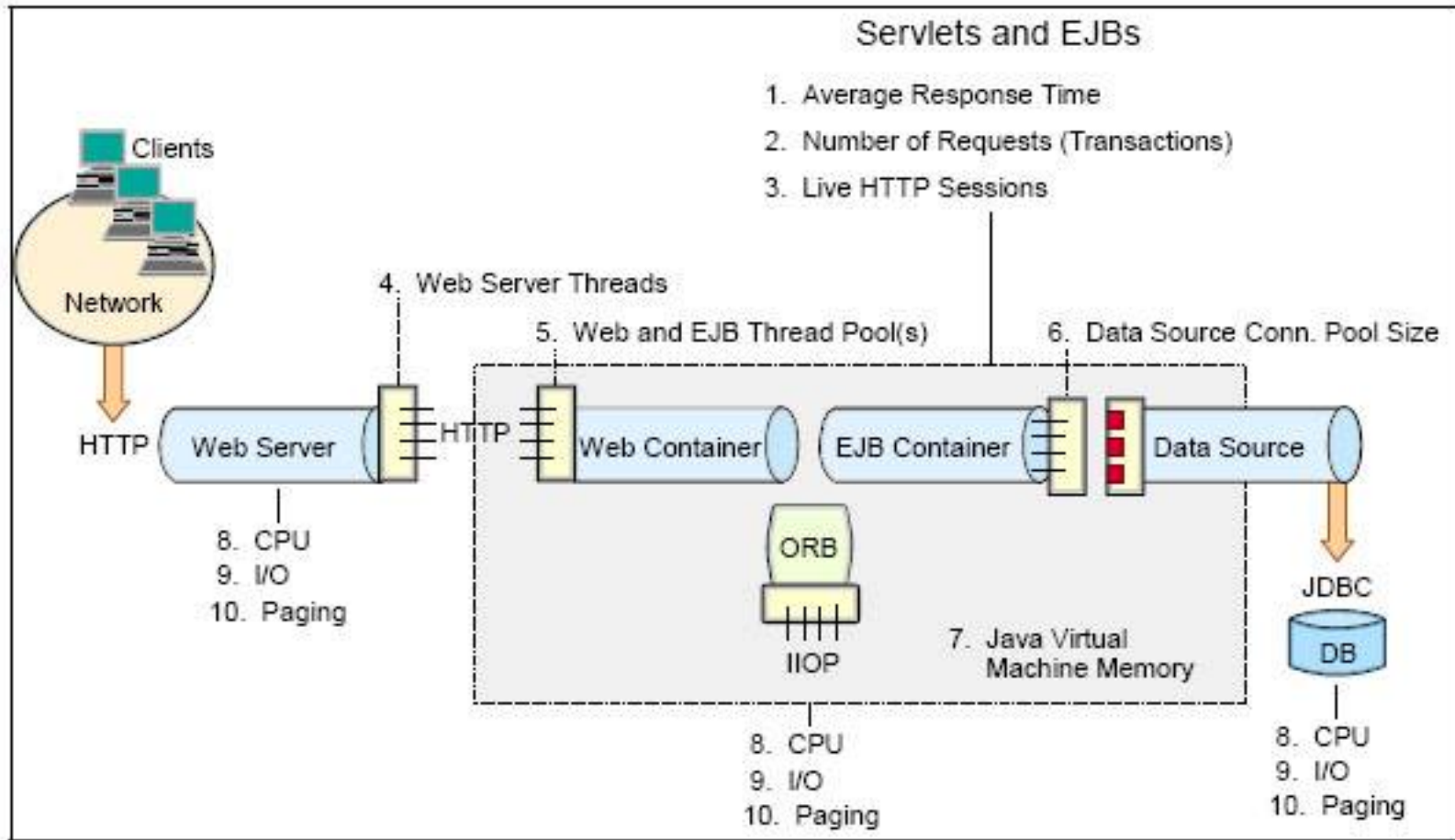


# Performance Tuning – Solaris OS

- Solaris TCP tunings
  - `ndd -set /dev/tcp tcp_conn_req_max_q 8192`
  - `ndd -set /dev/tcp tcp_conn_req_max_q0 8192`
  - `ndd -set /dev/tcp tcp_max_buf 4194304`
  - `ndd -set /dev/tcp tcp_cwnd_max 2097152`
  - `ndd -set /dev/tcp tcp_recv_hiwat 400000`
  - `ndd -set /dev/tcp tcp_xmit_hiwat 400000`
- These settings are a good start to handle thousands of connections and may require additional tuning
- General/specific OS tuning discussed here are not uncommon or unique to Solaris

# Performance Tuning – WebSphere

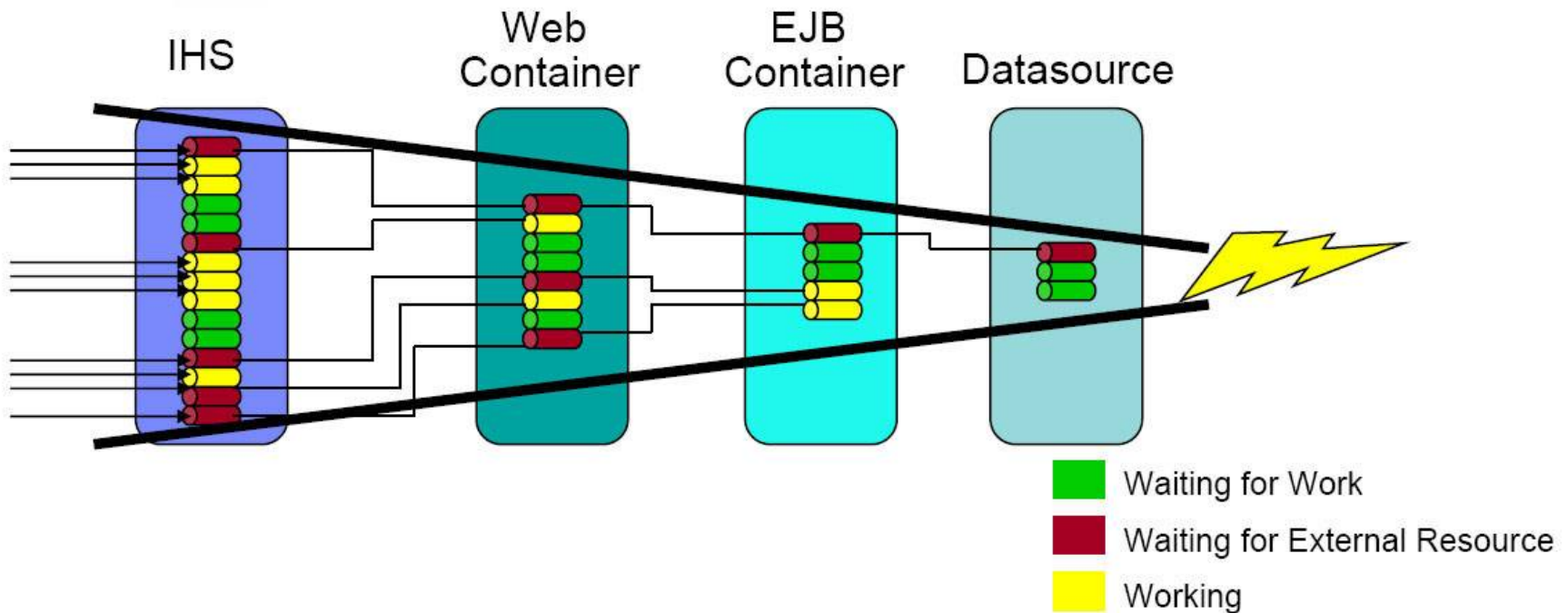
## Checklist



Source: IBM

from IBM WebSphere Performance Tuning Guide

# Performance Tuning – WebSphere



## Funnel Queuing Model

Source: IBM

# Performance Tuning – WebSphere

- Thread Pool Sizes
  - > Fixed sized pool (Min=Max) gives better performance
  - > Lower is better – CMT can have high numbers
- Default Thread Pool
- EJB Container
  - > Bean Cache
  - > Timeout
- Web Container HTTP Transport
  - > Keep Alive
  - > Buffer Size
  - > TimeOuts

# Performance Tuning – WebSphere

- JDBC
  - > Driver - Type 4
  - > Keep it optimal
  - > Statement Cache
- Persistent Manager (CMP)
- Disable unnecessary feature/functions
  - > PMI (Performance Monitoring Infrastructure)
  - > Debug logs
  - > GC details/verbose

# Performance Tuning – JVM

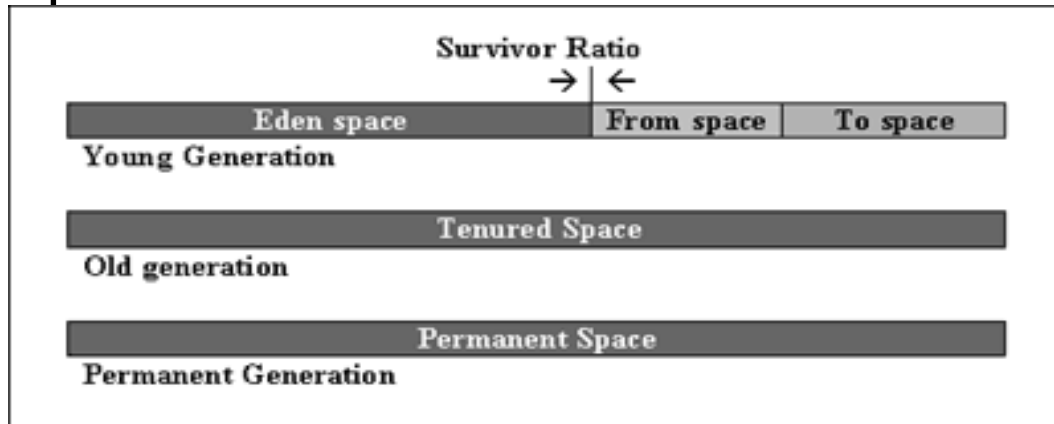
Key considerations for tuning the JVM

- Heap Sizing
- Garbage Collector
- Compiler Optimization
- Hardware-based optimization

[http://java.sun.com/j2se/reference/whitepapers/memorymanagement\\_whitepaper.pdf](http://java.sun.com/j2se/reference/whitepapers/memorymanagement_whitepaper.pdf)

# Performance Tuning – JVM Heap Sizing

## HotSpot Heap Generations

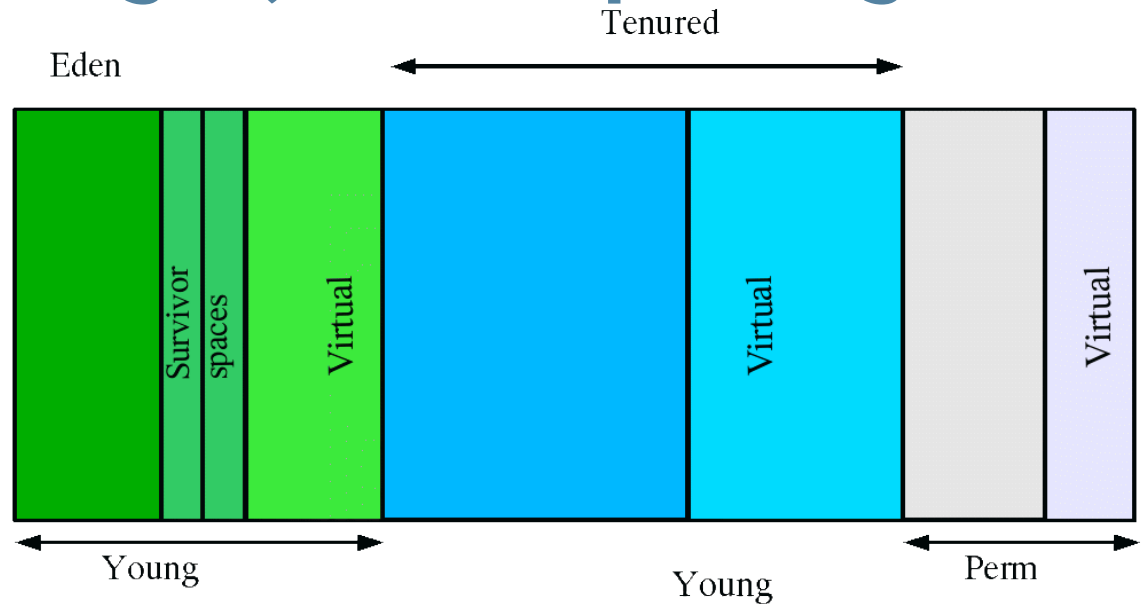


- > Young Generation
  - > Eden: new objects + 2 Survivor Spaces
- > Old Generation: Survived partial GC cycles
- > Permanent Generation
  - > JVM's reflective class and method objects
  - > Native Objects and threads

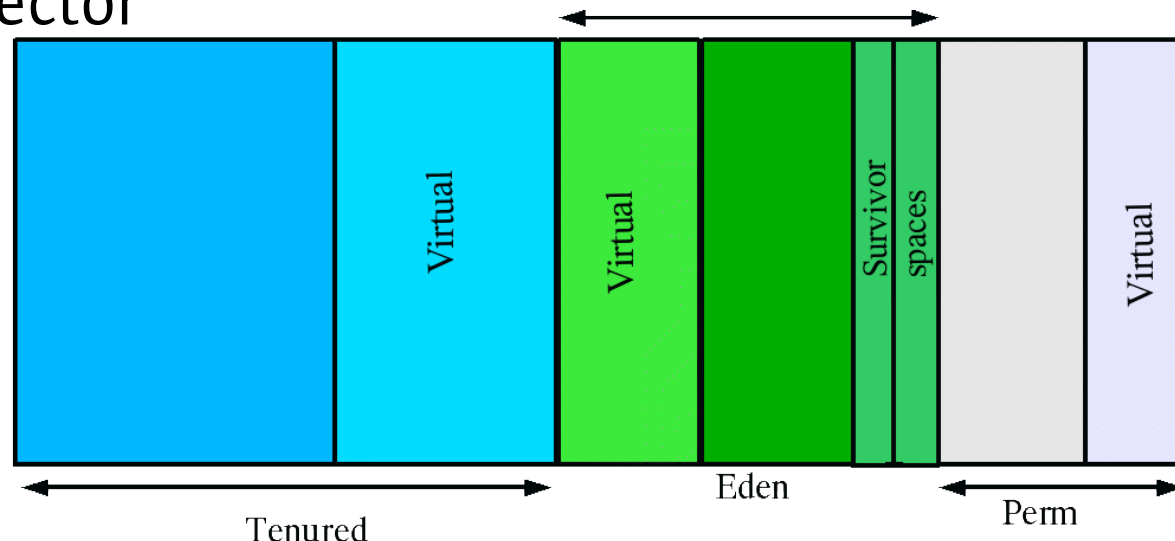
# Performance Tuning – JVM Heap Sizing

## HotSpot Heap Layout

> Default Layout



> Throughput Collector





# Performance Tuning – JVM Heap Sizing

- Proper Java heap sizing is key to good JVM performance
  - > **initialHeapSize (-Xms), maximumHeapSize (-Xmx)**
  - > Fixed Size heap (i.e. **initialHeapSize=maximumHeapSize**) usually gives better performance

Note: Setting -Xms and -Xmx to the same value increases predictability by removing the most important sizing decision from the virtual machine. On the other hand, the virtual machine can't compensate if you make a poor choice.

- Proper young generation size is equally important
  - > **-XX:NewSize=<n> -XX:MaxNewSize=<n>**
  - > **“-Xmn<n>” == “-XX:NewSize=<n> -XX:MaxNewSize=<n>”**
  - > Making Fixed Size young generation and Fixed Size Heap usually give better performance

# Performance Tuning – JVM Heap Sizing

- Permanent Generation
  - > **-XX:PermSize=<initial size>** and **-XX:MaxPermSize=<max size>**
  - > The default values between Sun JDK and IBM WebSphere's JDK are not the same
- Stack Size (-Xss)
  - > **Xss=512k**
  - > The default values between Sun JDK and IBM WebSphere's JDK are not the same
- Threads
  - > They all take memory from the address space
  - > For 32-bit need to be accounted
  - > GC + WebSphere Thread Pool + Native Threads

<http://java.sun.com/docs/hotspot/VMOptions.html>

# Performance Tuning – JVM Garbage Collectors

How to choose the right Collector?

- Best is Tune->Run->Analyze->Decide
- Serial Copying Collector (default for **-client**)
  - > co-locating multiple instances of WAS (JVM's)
- Parallel or Throughput Collector
  - > for better performance on multi-CPU servers
  - > **-XX:+UseParallelGC & -XX:ParallelGCThreads=?**
- Concurrent Low Pause Collector
  - > for constant response time on multi-CPU servers
  - > **-XX:+UseConcMarkSweepGC**
- Reduce Full GC cycle time
  - > [http://java.sun.com/docs/hotspot/gc5.0/gc\\_tuning\\_5.html](http://java.sun.com/docs/hotspot/gc5.0/gc_tuning_5.html)

# Performance Tuning – JVM Throughput

- **-XX:+UseParallelOldGC**
  - > For Old Generation
  - > Minimize garbage collection impact on throughput
  - > Does not target low pause times (Use CMS for that)
- **-XX:ParallelGCThreads=<n>**
  - > Large-scale deployment running multiple JVM tools
  - > By default, set to number of hardware threads
  - > On US-T1 and T2, total GC threads on the system should not exceed 20: You might have expected 32 or 64
  - > Experiment needed, mileage will vary

# Performance Tuning – JVM Througput

- **-XX:+AggressiveOpts**
  - > Turns on point performance compiler optimizations that are available in or after v1.5.0\_06
  - > Code Quality optimization, still need to tune GC
- **-XX:+AggressiveHeap**
  - > attempts to set various parameters to be optimal for long-running, memory allocation-intensive jobs.
  - > Will be replaced by **-XX:+AggressiveOpts**
- **-XX:+UseBiasedLocking (5–10%)**
  - > Will be on by default from Java SE v.6 platform
  - > Bias synchronized object to the thread that created it
  - > If the synchronized block is never accessed by another thread, uses cmp+branch, not atomics, to lock/unlock

# Performance Tuning – JVM Low Pause Time

- **-XX:NewRatio=<n> -Xmn -XX:[Max]NewSize**
- Choose other suitable options:
  - > **-XX:SurvivorRatio=<n>**
- **-XX:MaxTenuringThreshold=<n>**
  - > Smaller young generation can put more pressure on CMS old generation
  - > Larger young generation can increase young generation pause times
  - > Experiment and verify

# Performance Tuning – JVM Low Pause Time

Key parameters for CMS

- **-XX:ParallelCMSThreads=<n>**
  - > Dynamically set based on ParallelGCThreads
- **-XX:ParallelGCThreads=<n>**
  - > Default: number of hardware threads (ncpus)
  - > Try  $\text{ncpus} = \text{ncpus} \leq 8 ? \text{ncpus} : \text{ncpus} * 5/8;$
- **-XX:CMSInitiatingOccupancyFraction=<n>**
  - > Old gen occupancy at which CMS starts collecting
  - > Larger values improve throughput and Full GC risk
  - > Lower values reduce throughput and Full GC risk

# Performance Tuning – JVM Large Pages

- Enabled by default on Solaris – It just works
  - > **-XX:LargePageSizeInBytes=<n>**
- Default page size is 8k on SPARC, 4k on x64
- US-III and US-IV systems – 4m
- US-IV+ - 32m
- US-T1 - 256m
- X64 (AMD and Intel) - 2m

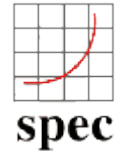


# Performance Tuning – JVM

## NUMA Optimizations

- Multiple JVM tools
  - > Use processor sets (Redbook Ch 5.4.3 pg 145)
  - > zonecfg
    - zonecfg:my-zone> add dedicated-cpu
    - zonecfg:my-zone:dedicated-cpu> set ncpus=1-3
    - zonecfg:my-zone:dedicated-cpu> set importance=2
    - zonecfg:my-zone:dedicated-cpu> end
  - > Significant improvement (5–10%) on x64 and US-IV+ systems
  - > /etc/system: **set lgrp\_mem\_pset\_aware=1**
    - > Default random policy applies only to lgroups with a process' processor set

# Performance Tuning – Samples



- Good place to see sample configurations and settings  
 > <http://www.spec.org/jAppServer2004/results/jAppServer2004.html>
- Example of WebSphere tuning on Sun T2000 Server:

## JVM Options:

```
initialHeapSize="2880" maximumHeapSize="2880" verboseModeGarbageCollection="false"
-server -Xmn780m -Xss128k -XX:-ScavengeBeforeFullGC -XX:+UseParallelGC
-XX:ParallelGCThreads=24 -XX:PermSize=128m -XX:MaxTenuringThreshold=16
-XX:+UseParallelOldGC
```

EJB Cache Size = 37543

HTTP Channel maximum persistent requests = -1

HTTP Channel readTimeout/writeTimeout = 6000/6000

HTTP Channel persistentTimeout = 3000

Web Container threads (Minumum/Maximum) = 56/56

ORB threads (Minumum/Maximum) = 40/40

Default threads (Minumum/Maximum) = 15/15

- > **Java process changed to run in FX scheduler class:**

```
# /usr/bin/priocntl -s -c FX -m 59 -p 59 -i <pid>
```

# Performance Tuning – Tools

- WebSphere Monitoring (Tivoli Perf. Viewer)
    - > Help locate the app server performance issues
  - Solaris system monitoring
    - > Indicator of the system behavior
      - > **vmstat, mpstat, prstat, iostat, netstat, DTrace**
        - System Activity
        - Context switches
        - Number of threads or lwp
        - Disk contention for disk where bean passivation and transaction logs are logged
        - Network activity
- > [http://developers.sun.com/solaris/articles/performance\\_tools.html](http://developers.sun.com/solaris/articles/performance_tools.html)
- > <https://solaris10-dtrace-vm-agents.dev.java.net>

# Performance Tuning – Tools

- Tools to monitor GC
  - > JVM GC output (`-verboseModeGarbageCollection="true"`)
    - > Frequency, Pause time, heap reclaimed

```
[GC 50650K->21808K(76868K) , 0.0478645 secs]
[GC 51197K->22305K(76868K) , 0.0478645 secs]
[GC 52293K->23867K(76868K) , 0.0478645 secs]
[Full GC 52970K->1690K(76868K) , 0.54789968 secs]
```
- **-XX:+PrintGCDetails -XX:+PrintGCTimeStamps**
- Need more details
  - > **-XX:-PrintTenuringDistribution**
  - > **-XX:-PrintCommandLineFlags**
  - > **-XX:-PrintCompilation**

# JVM Debugging – Tools

- **-XX:-HeapDumpOnOutOfMemoryError**
- **-XX:OnOutOfMemoryError="`<cmd args>;<cmd args>`"**
- **-XX:OnError="`<cmd args>;<cmd args>`"**
- JVM Errors/Output directed to:
  - > WAS\_LOG\_DIR/native\*log files.
- *MustGather* Tool for Solaris
  - > <http://www-1.ibm.com/support/docview.wss?rs=180&uid=swg21049530>

# Performance Tuning – Tools

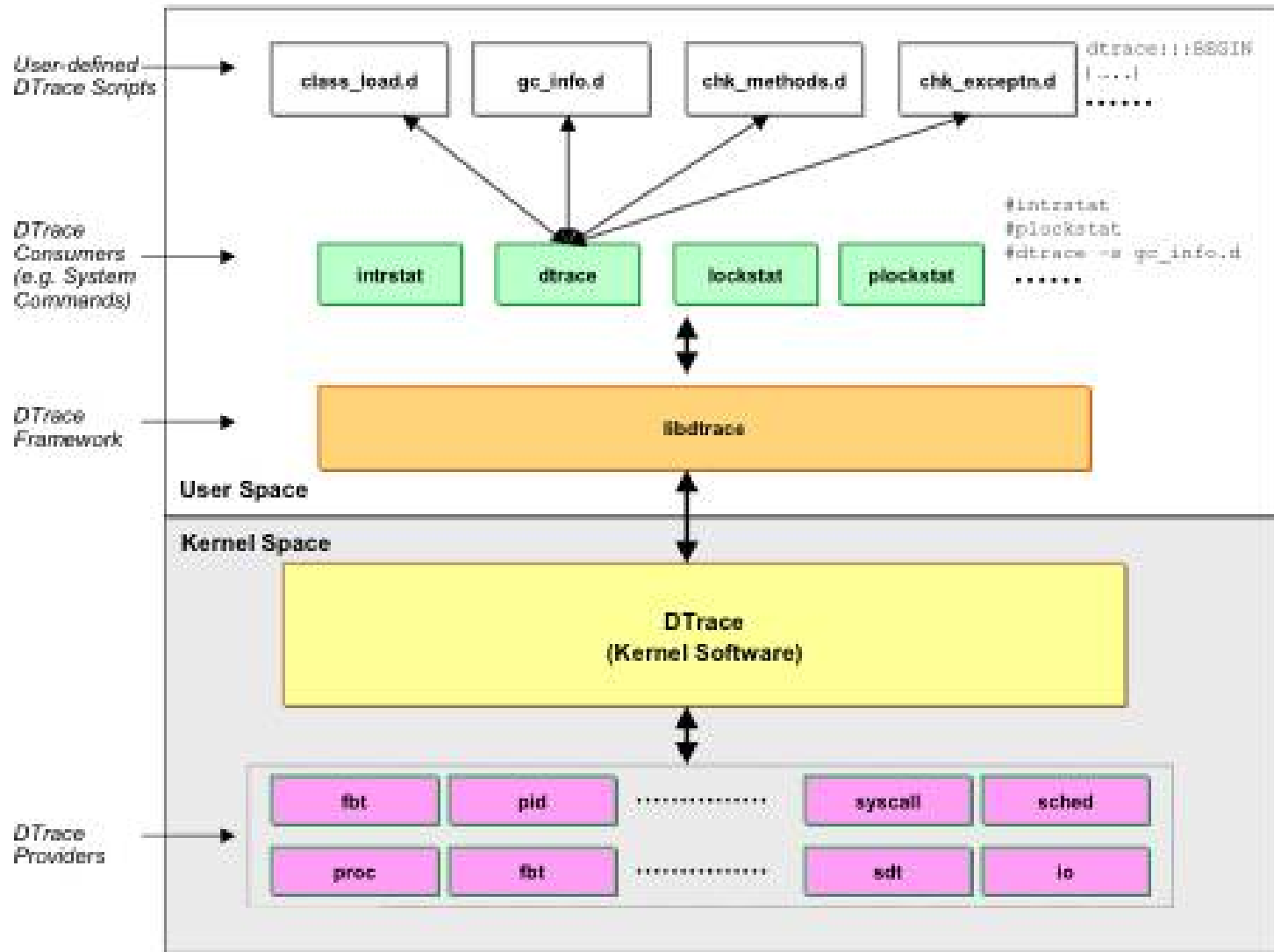
## Examine the WAS process and threads

- **/usr/ucb/ps auwwx <pid of WAS process>**
  - > Lists out how the WAS JVM started
  - > Useful in inspecting some of the process argument
- **pmap(1M)**
  - > Can get information about stack size
- **Thread Dump (kill -3) allows examining real-time thread activity within the server**
  - > Take 3-4 dumps for few minutes apart
  - > Look for problem trends/contention points
  - > WebSphere Thread Analyzer (IBM download)

# Performance Tuning – Tools

- jvmstat Tool
  - > <http://java.sun.com/performance/jvmstat>
- Java heap analysis tool (HAT)
  - > <https://hat.dev.java.net>
- Profiling WebSphere Applications using Sun Studio Collector/Analyzer
  - > [http://developers.sun.com/sunstudio/articles/profiling\\_websphere.html](http://developers.sun.com/sunstudio/articles/profiling_websphere.html)
- Other Tools
  - > <http://java.sun.com/tools>
  - > Wily Introscope
  - > Quest PerformaSure
  - > tuneWAS.jacl (ibm.com)

# Performance Tuning – Tools DTrace





# WebSphere with 64-bit JVM

- As of WAS v6.1, IBM provides added support for 64-bit JVM on Solaris SPARC and x64 platforms
- With 64-bit address space, more than 4GB of heap memory is available (“unlimited”)
- Server VM (Client VM in 32-bit only)
- Solaris OS must be installed with 64-bit mode
- JNI code requires re-compilation

# WebSphere with 64-bit JVM

- Gain higher precision for encryption/decryption algorithms
- Applications capable of taking advantage of WAS 64-bit features may gain performance
- Otherwise, adverse effect: performance loss
  - > All address references are 64-bit wide, twice the size of address references in 32-bit deployments
  - > An increased memory footprint reducing hardware cache efficiency and thus performance
  - > May not be issue on some platform

# Summary

- Be aware of the WebSphere application environment design, configuration and implementation
  - > Follow the guidelines provided in this presentation
- Performance management is an iterative process and requires holistic approach
- Capacity Planning for changes in workload conditions (Tools: TeamQuest, OpTier, etc.)
- Good performance management will enable
  - > better Return on your Investments
  - > higher competitive advantage

# References

- Sun WebSphere Blogs
  - > <http://blogs.sun.com/dkumar>
  - > <http://blogs.sun.com/sunabl>
- IBM WebSphere Info Center
  - > <http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.nd.doc/info/ae/ae/welc6toptuning.html>
- IBM WebSphere Redbooks (SG247584)
  - > <http://www.redbooks.ibm.com/redpieces/abstracts/sg247584.html>
- HotSpot VM and Java Tuning
  - > <http://java.sun.com/javase/technologies/hotspot>
  - > <http://java.sun.com/performance/reference/whitepapers/tuning.html>
  - > [http://java.sun.com/performance/reference/whitepapers/5.0\\_performance.html](http://java.sun.com/performance/reference/whitepapers/5.0_performance.html)
- Solaris Tuning
  - > [http://developers.sun.com/solaris/articles/tuning\\_solaris.html](http://developers.sun.com/solaris/articles/tuning_solaris.html)



# Q & A

## Thank You.

[Albert.Leigh@Sun.COM](mailto:Albert.Leigh@Sun.COM)

[Dileep.Kumar@Sun.COM](mailto:Dileep.Kumar@Sun.COM)

