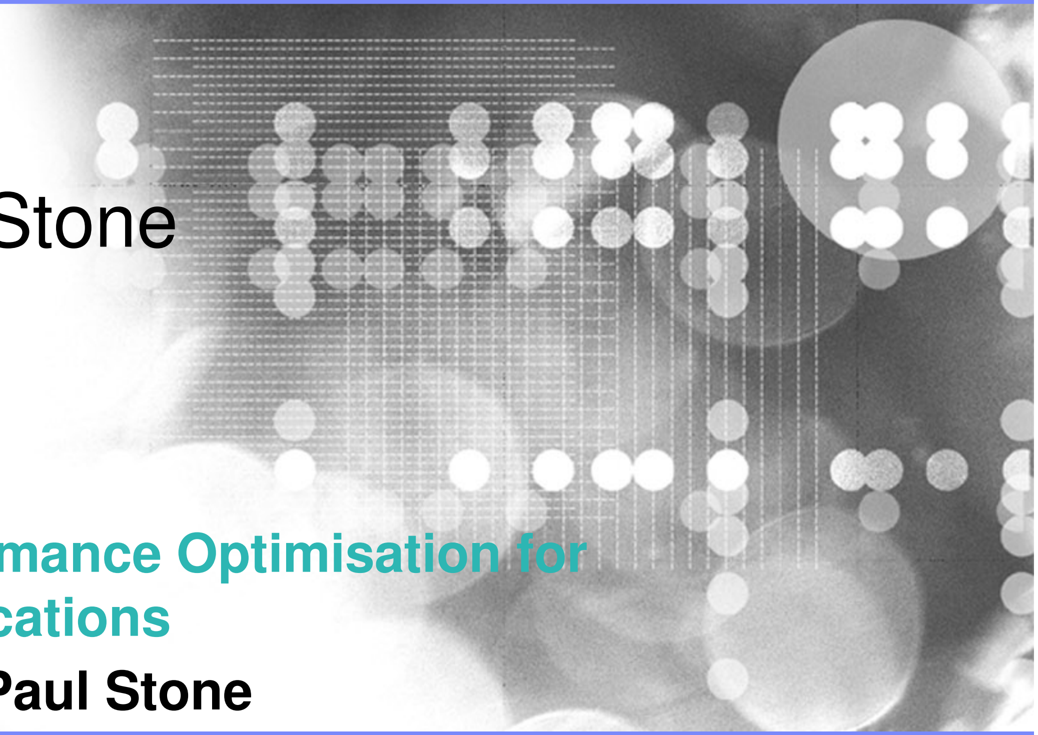
ISSW

Gareth J Jones & Paul Stone



Real World Performance Optimisation for WebSphere Applications

Gareth J Jones & Paul Stone

March 2008 | Bedfont Lakes

Real World Performance Optimisation for WebSphere Applications

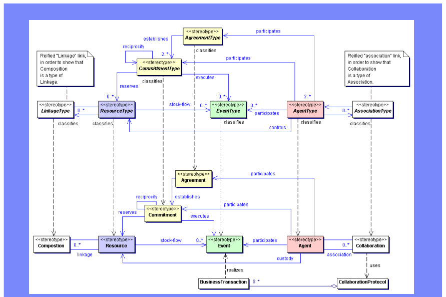
- Introduction - Supply and Demand
- Overview of Performance Optimisation Process
- Useful Tools

Real World Performance Optimisation for WebSphere Applications

- **Introduction - Supply and Demand**
- Overview of Performance Optimisation Process
- Useful Tools

Performance is a case of Supply and Demand

- Applications demand resources
- Platforms supply resources
- Performance problems occur when demand is greater than supply.
- Tuning needs to consider the Platform and the application together.



Application

Java Web Application

Demand per transaction	Supply
0.1 rPerf s CPU	200 rPerf CPU
5 MB Disk IO	1500 MBps Disk IO
75K Network IO	50 Mbps Network IO

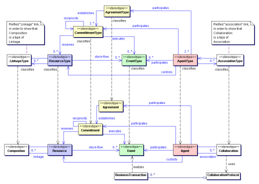
Platform

WebSphere
Java
Unix

Workload

1,000 Transactions per second

Limited Computing Resources:



Component	"Resource" Type
Application	Shared Data Structures, External Systems (e.g. databases, legacy systems)
WAS	Thread Pool, Connections (DB or Messaging)
JRE	Heap Memory, Threads, Monitors (Locks, Synchronized blocks, Mutex)
OS	Threads, Caches, Buffers
HW	CPU Cycles, Disk IO, Network IO, Memory

Dominant Bottleneck and Iterative optimisation

Demand per transaction	Supply	Maximum Achievable throughput
0.1 rPerf s CPU	200 rPerf	2000 txs per second
0.5 MB Disk IO	1500 MBps Disk IO	3000 txs per second
75k Network IO	500 Mbps Network IO	6800 txs per second

Demand per transaction	Supply	Maximum Achievable throughput
0.1 rPerf s CPU	**200 rPerf**	**2000 txs per second**
5 MB Disk IO	**1500 MBps Disk IO**	**300 txs per second**
75k Network IO	**50 Mbps Network IO**	**680 txs per second**

- At any one time, one of these resources will be the "dominant bottleneck", dictating the maximum workload of the application.
- Eliminating the dominant bottleneck will uncover the next one.

Real World Performance Optimisation for WebSphere Applications

- Introduction - Supply and Demand
- **Overview of Performance Optimisation Process**
- Useful Tools
- Common Performance Problems

Overview of Performance Optimisation Process

- Define performance **targets**.
- Establish access to a suitable **environment**.
- **Measure** performance (throughput, response time), compare to target.
  If performance is below target:
- Identify **limiting resource**, the dominant bottleneck.
- Identify **source** of bottleneck.
- **Remove** bottleneck.
- **Repeat** process until performance target is met.

Define performance targets

- Clear performance targets help focus the effort required to improve performance.
  - Review terms such as “hit”, “request”, “user”, “active user”, “day”, “visit”

- Typically a combination of **throughput** and **response time**. Usually derived from the SLAs and Volumetrics of a system.

- Determine the profile of requests coming in to the system at **different times**.

- Other targets - include **scalability** i.e. how well the system deals with increasing workload, particularly over multiple servers.

Establish access to a suitable environment.

- When measuring performance, the characteristics of the system need to be monitored.

Development Environment	Live Environment
More effort required to setup	Minimal setup costs
Freedom to change	Limited ability to change
Freedom to inject load	Limited ability to inject load
Can use “Invasive” monitoring tools e.g. profiling, debugging	Monitoring tools must not interfere with live operation
Allows more proactive optimisation	Testing will be reactive
Needs to be like for like	

- Set up load injectors, testing scripts and stubs / test harnesses, test data.

Review and Understand environment.

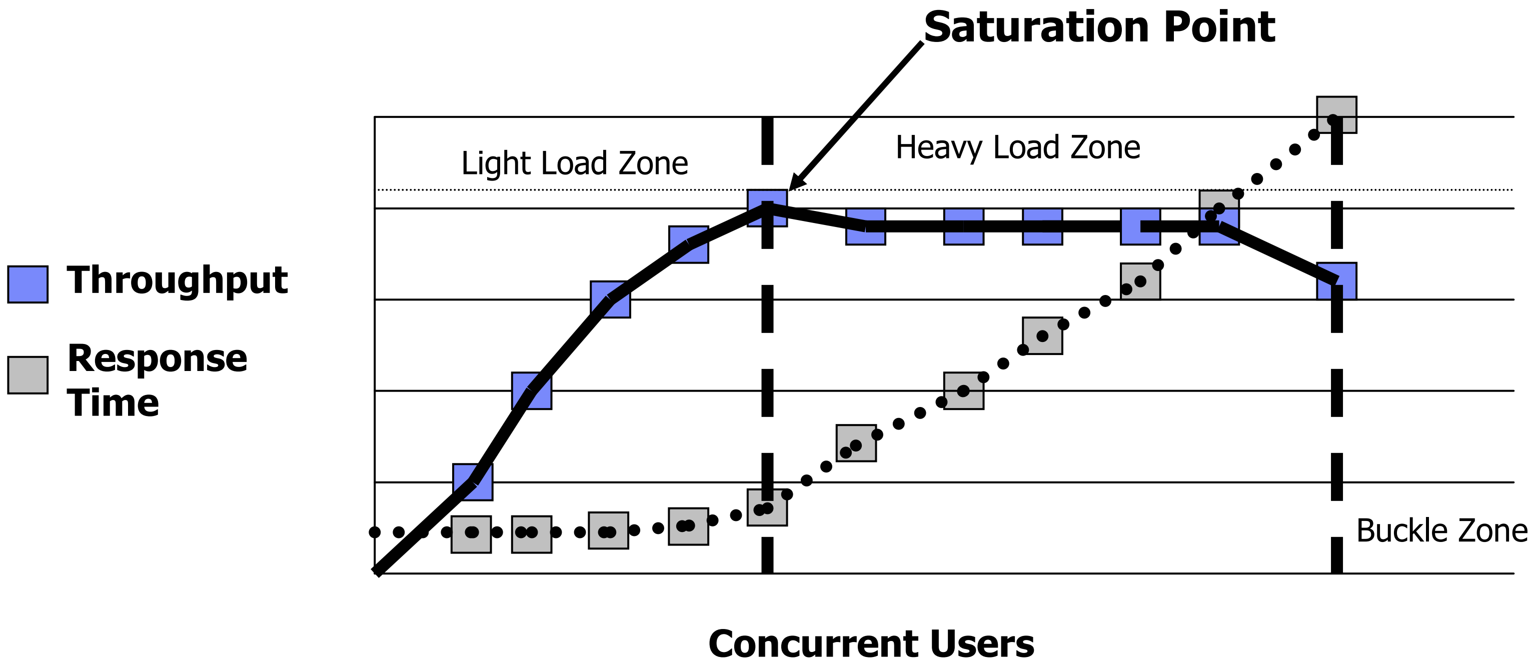
- Understand system components, hardware and software.
  - What is the 'application', what code or configuration can be changed?
  - What middleware and OS is the application running on?
  - What is the Hardware Platform Configuration - CPUs, Networks, Disks, Memory?
  - What monitoring and profiling tools arise from the standard middleware/OS?
  - What Additional Diagnostic tools are available?
  - What components can be changed, Can we add hardware? Modify application code?
  - Who is responsible for each component?

Measure performance

- Aim is to measure throughput and/or response time with system is under full load and compare to target.
- If **consistently** better than target, then no optimisation is required.
- Performance test tools e.g. Rational Performance tester and Loadrunner will monitor the system under load and provide throughput and response time data.
- Log Files/system output can be analysed to measure performance
- If the brief is just to 'make things faster' it is still important to measure performance so that we can show the results and improvement due to optimisations.

Measure Throughput and Response Time

- Focus on the **throughput** and **response time** requirements for the system as the results are usually related.



Identify limiting resource

- With the system under full load, monitor the use of system resources to determine which resources are causing the **dominant bottleneck**.

- The OS will come with a number of tools to monitor CPU load, Disk IO, Network IO etc.

- Middleware components may provide tools to monitor their own resource managers.

  WebSphere - PMI, ARM,

  Java - Verbose GC

Bottleneck Diagnosis

- **CPU** - a CPU load constantly higher than 90% should be considered in Over-Demand
- **Network** –indicated by CPU wait time and significant network load.
- **Disk** –look at the physical disk\sec per read and sec per write counters for your disk drives. If the latencies are larger than 20 ms then there is a potential bottleneck
- **Java Heap** - If GC takes more than 15% of the time then Java Heap considered to be in Over-Demand.
- **WAS Connection Pool** - A fully allocated pool with multiple waiting threads should be considered in Over-Demand
- **WAS Thread Pool**- Again A fully allocated pool with multiple threads should be considered in Over-Demand
- **DB2 Buffer Pool** - a high level of Buffer Pool misses may be causing a performance bottleneck

Identify Source of Bottleneck.

- If we need to reduce the resource demand, we need to find which component of the application is responsible

- This can require more detailed monitoring tools
  - Profilers
  - detailed OS tools

- These tools are likely to be more invasive, so are more useful in a development environment.

Remove Bottleneck.

- Manage the resource in a more efficient manner – tune or configure the resource managers

- Reduce demand on the resource - typically by application modification

- Supply more resource – typically by physical addition of hardware

Real World Performance Optimisation for WebSphere Applications

- Introduction - Supply and Demand
- Overview of Performance Optimisation Process
- **Useful Tools**
- Common Performance Problems

Tools

Resource Monitoring Tools

Platform	Multi	CPU	Disk	Network
Unix	vmstat, topas, nmon	tprof	iostat, filemon	netstat
Windows	Windows Task Manager			

Resource Monitoring Tools

Platform	Tool	Resources
Java	Java Core generation	Locks,Common Thread Execution
Java	Verbose GC GCCollector	Java Heap memory
WAS	TPV PMAT MDD4J ThreadAnalyser	Thread Pools, Connection Pools, Statement Caches etc.
WAS	ITCAM	All WAS Resources
DB2	DB2 Monitor	Buffer Pools, Agents, SQL execution etc.

OS Tools

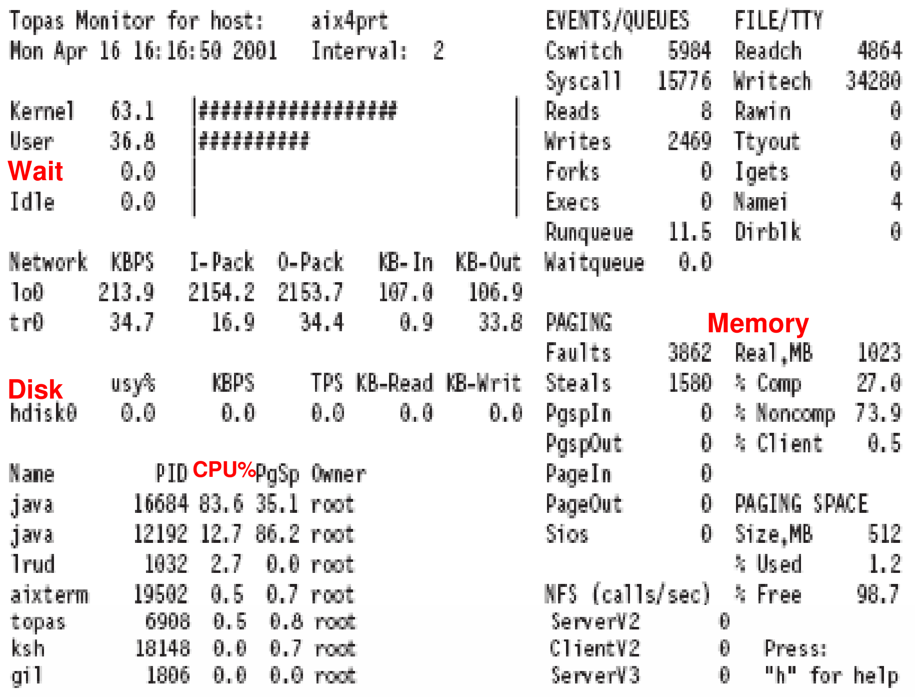
OS Tools

- **vmstat(AIX/Linux)**
  - to display a summary of memory. swap space, io and context switch data plus cpu usage.
- **top(UNIX/LINUX)**
  - Shows a second by second state of a machine, including CPU/IO usage as well as top processes. Hitting "1“ in top toggles the mode to show "per CPU" stats, useful if a single thread is CPU bound.
- **topas(UNIX/AIX)**
  - Like top but for AIX only.

topas

On this example the disk perspective shows no I/O activity at all.

The wait section can help determine if the system is I/O bound. High numbers here then use other tools, such as filemon to help figure out which processes, adapters, or file systems are causing bottlenecks.



```
Topas Monitor for host:    aix4prt              EVENTS/QUEUES    FILE/TTY
Mon Apr 16 16:16:50 2001   Interval:  2         Cswitch    5984  Readch     4864
                                                Syscall   15776  Writech   34280
Kernel   63.1   |##################          |  Reads         8  Rawin         0
User     36.8   |##########                  |  Writes     2469  Ttyout        0
Wait      0.0   |                            |  Forks         0  Igets         0
Idle      0.0   |                            |  Execs         0  Namei         4
                                                Runqueue   11.5  Dirblk        0
Network  KBPS   I-Pack  O-Pack   KB-In  KB-Out  Waitqueue   0.0
lo0     213.9   2154.2  2153.7   107.0   106.9
tr0      34.7     16.9    34.4     0.9    33.8  PAGING           Memory
                                                Faults     3862  Real,MB    1023
Disk     usy%     KBPS     TPS KB-Read KB-Writ  Steals     1580  % Comp     27.0
hdisk0    0.0      0.0     0.0     0.0     0.0  PgspIn        0  % Noncomp  73.9
                                                PgspOut       0  % Client    0.5
Name          PID CPU% PgSp Owner               PageIn        0
java        16684 83.6 35.1 root                PageOut       0  PAGING SPACE
java        12192 12.7 86.2 root                Sios          0  Size,MB     512
lrud         1032  2.7  0.0 root                                 % Used      1.2
aixterm     19502  0.5  0.7 root                NFS (calls/sec)  % Free     98.7
topas        6908  0.5  0.8 root                 ServerV2     0
ksh         18148  0.0  0.7 root                 ClientV2     0    Press:
gil          1806  0.0  0.0 root                 ServerV3     0    "h" for help
```

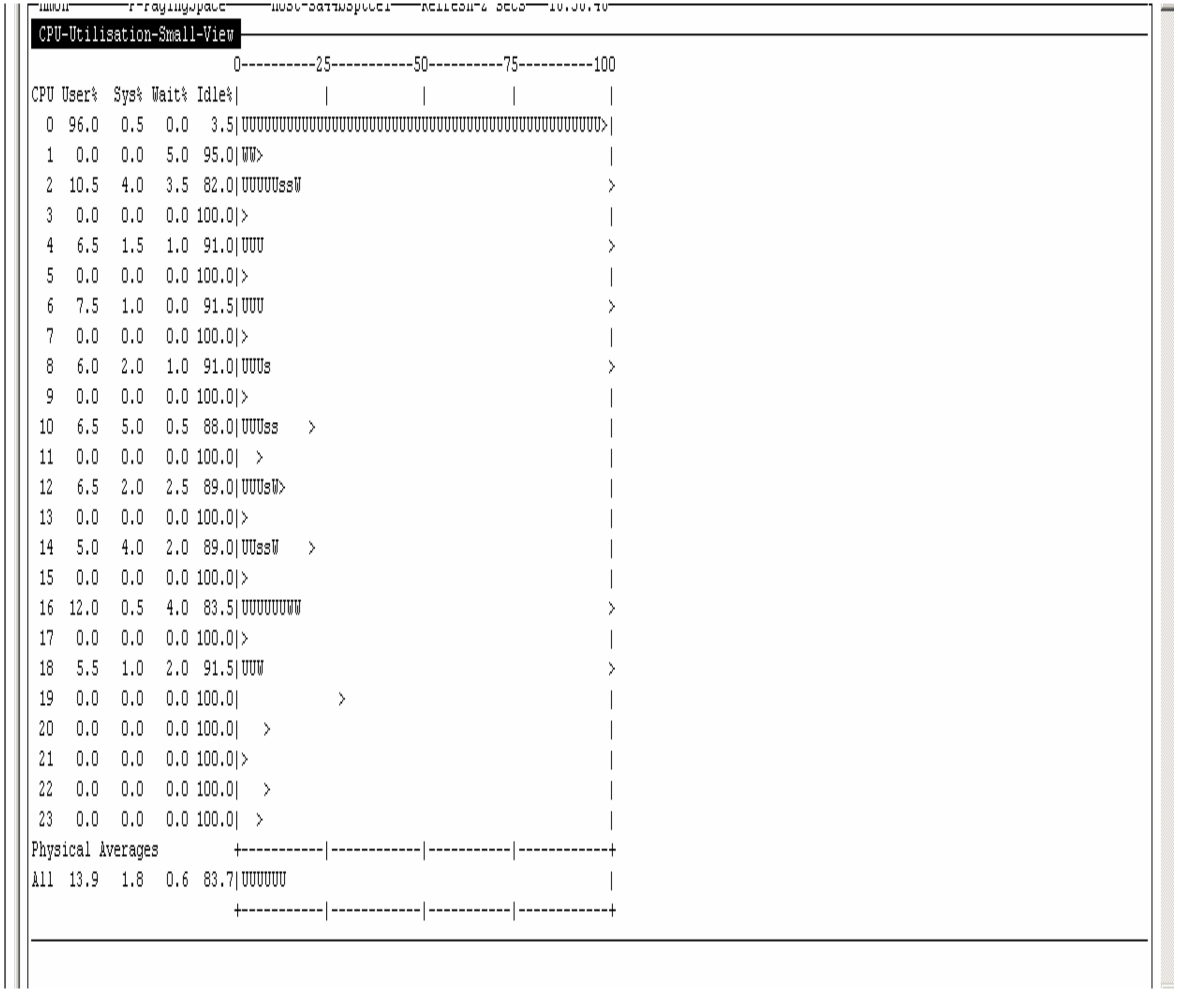
OS Tools – Unix/AIX/Linux

- **nmon (AIX)**
  - free interactive tool that gives much of the same information as topas, but saves the information to a file in Lotus 123 and Excel formats.
  - Spreadsheet Analysis tools available
  - The download site is www.ibm.com/servers/esdd/articles/analyze_aix/.

nmon – CPU, memory use, kernel internal stats

The information that is collected includes CPU, disk, network, adapter statistics, kernel counters, memory, and the ’top’ process information.

This example is the CPU view



```
-nmon---------p-PagingSpace-------host=sa4wsptce1----Refresh=2 secs---10:50:40-
CPU-Utilisation-Small-View
                           0----------25-----------50----------75----------100
CPU User%  Sys% Wait% Idle%|           |            |           |            |
  0  96.0   0.5   0.0   3.5|UUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUU>|
  1   0.0   0.0   5.0  95.0|WW>                                              |
  2  10.5   4.0   3.5  82.0|UUUUUssW                                         >
  3   0.0   0.0   0.0 100.0|>                                                |
  4   6.5   1.5   1.0  91.0|UUU                                              >
  5   0.0   0.0   0.0 100.0|>                                                |
  6   7.5   1.0   0.0  91.5|UUU                                              >
  7   0.0   0.0   0.0 100.0|>                                                |
  8   6.0   2.0   1.0  91.0|UUUs                                             >
  9   0.0   0.0   0.0 100.0|>                                                |
 10   6.5   5.0   0.5  88.0|UUUss    >                                       |
 11   0.0   0.0   0.0 100.0|  >                                              |
 12   6.5   2.0   2.5  89.0|UUUsW>                                           |
 13   0.0   0.0   0.0 100.0|>                                                |
 14   5.0   4.0   2.0  89.0|UUssW    >                                       |
 15   0.0   0.0   0.0 100.0|>                                                |
 16  12.0   0.5   4.0  83.5|UUUUUUWW                                         >
 17   0.0   0.0   0.0 100.0|>                                                |
 18   5.5   1.0   2.0  91.5|UUW                                              >
 19   0.0   0.0   0.0 100.0|             >                                   |
 20   0.0   0.0   0.0 100.0|   >                                             |
 21   0.0   0.0   0.0 100.0|>                                                |
 22   0.0   0.0   0.0 100.0|   >                                             |
 23   0.0   0.0   0.0 100.0|  >                                              |
Physical Averages          +-----------|------------|-----------|------------+
All  13.9   1.8   0.6  83.7|UUUUUU                                           |
                           +-----------|------------|-----------|------------+
```

OS Tools – Unix/AIX

- iostat(AIX)
  - Useful to determine if a system has an I/O bottleneck. The read and write rate to all disks is reported.
- tprof(AIX/Linux/Windows)
  - Useful to determine which sections of a program are most heavily using the CPU.
- netstat(AIX/Linux)
  - Provides detailed data on network usage if a system is network bound. Network activity and connections can be displayed
- filemon(AIX)
  - Use this command to find which filesystems and files are most heavily accessed.

Using OS Tools to find a bottleneck

- Start with a tool such as vmstat on AIX
- vmstat produces a compact report that details the activity of these three areas

vmstat 1 10 outputs:

```
kthr     memory               page            faults               cpu
 -----   ----------- ------------------------ ------------         -----------
r b      avm       fre  re pi po fr sr cy in   sy        cs        us sy id wa
0 0      189898  612  0  0  0   3 11 0 178 606   424     6  1  92 1
1 0      189898  611  0  1  0   0 0   0 114 4573 122     96 4  0   0
1 0      189898  611  0  0  0   0 0   0 115 420   102     99 0  0   0
1 0      189898  611  0  0  0   0 0   0 115 425    91     99 0  0   0
1 0      189898  611  0  0  0   0 0   0 114 428    90     99 0  0   0
1 0      189898  610  0  1  0   0 0   0 117 333   102     97 3  0   0
1 0      189898  610  0  0  0   0 0   0 114 433    91     99 1  0   0
```

Using OS Tools to find a CPU bottleneck

- For a CPU bound system follow up with a tool such as tprof
- > tprof -s -k -x sleep 60

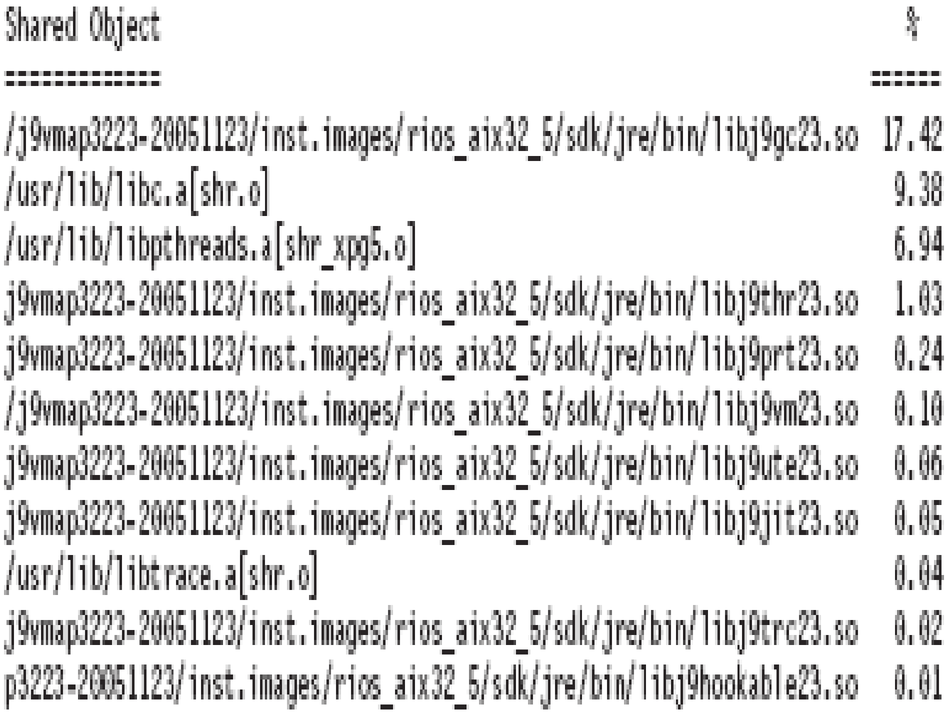
```
Process                 Freq    Total    Kernel   User    Shared   Other
=======                 ====    =====    ======   ====    ======   =====
./java                  5       59.39    24.28    0.00    35.11    0.00
wait                    4       40.33    40.33    0.00    0.00     0.00
/usr/bin/tprof          1       0.20     0.02     0.00    0.18     0.00
/etc/syncd              3       0.05     0.05     0.00    0.00     0.00
/usr/bin/sh             2       0.01     0.00     0.00    0.00     0.00
gil                     2       0.01     0.01     0.00    0.00     0.00
afsd                    1       0.00     0.00     0.00    0.00     0.00
rpc.lockd               1       0.00     0.00     0.00    0.00     0.00
swapper                 1       0.00     0.00     0.00    0.00     0.00
=======                 ====    =====    ======   ====    ======   =====
Total                   20      100.00   64.70    0.00    35.29    0.00

Process      PID        TID     Total    Kernel   User    Shared   Other
=======      ===        ===     =====    ======   ====    ======   =====
./java       467018     819317  16.68    5.55     0.00    11.13    0.00
./java       467018     766019  14.30    6.30     0.00    8.00     0.00
./java       467018     725211  14.28    6.24     0.00    8.04     0.00
./java       467018     712827  14.11    6.16     0.00    7.94     0.00
…….etc
```

Using OS Tools to find a CPU bottleneck

- The example shows majority of time in **kernel** and **shared –** indicates the Java process is spending its time doing work inside the JVM (or some other native code).

- Further examination of the tprof shared library section would show which shared lib was taking the CPU

- In this case the culprit is libj9gc23.so – part of the JVM installation related to garbage collection – indicating a g.c. issue



```
Shared Object                                                          %
=============                                                     ======
/j9vmap3223-20051123/inst.images/rios_aix32_5/sdk/jre/bin/libj9gc23.so  17.42
/usr/lib/libc.a[shr.o]                                                  9.38
/usr/lib/libpthreads.a[shr_xpg5.o]                                      6.94
j9vmap3223-20051123/inst.images/rios_aix32_5/sdk/jre/bin/libj9thr23.so  1.03
j9vmap3223-20051123/inst.images/rios_aix32_5/sdk/jre/bin/libj9prt23.so  0.24
/j9vmap3223-20051123/inst.images/rios_aix32_5/sdk/jre/bin/libj9vm23.so  0.10
j9vmap3223-20051123/inst.images/rios_aix32_5/sdk/jre/bin/libj9ute23.so  0.06
j9vmap3223-20051123/inst.images/rios_aix32_5/sdk/jre/bin/libj9jit23.so  0.05
/usr/lib/libtrace.a[shr.o]                                              0.04
j9vmap3223-20051123/inst.images/rios_aix32_5/sdk/jre/bin/libj9trc23.so  0.02
p3223-20051123/inst.images/rios_aix32_5/sdk/jre/bin/libj9hookable23.so  0.01
```

Using OS Tools to find a CPU bottleneck

- If the time is spent in **other** the method is different – tprof does not detail which java methods are being run
- In this case a javacore file (or a series of javacore files) can be used to determine the stack trace for the TIDs shown to be taking the CPU time

Using OS Tools to find a Memory bottleneck

- With a memory bottleneck, find which processes are using large amounts of memory, and which of these are growing:
- > svmon -P -t 5

```
-------------------------------------------------------------------------------
Pid      Command    Inuse     Pin     Pgsp      Virtual   64-bit    Mthrd
38454    java       76454     1404    100413    144805    N         Y
-------------------------------------------------------------------------------
Pid      Command    Inuse     Pin     Pgsp      Virtual   64-bit    Mthrd
15552    X          14282     1407    17266     19810     N         N
-------------------------------------------------------------------------------
Pid      Command    Inuse     Pin     Pgsp      Virtual   64-bit    Mthrd
14762    dtwm       3991      1403    5054      7628      N         N
-------------------------------------------------------------------------------
Pid      Command    Inuse     Pin     Pgsp      Virtual   64-bit    Mthrd
15274    dtsessi    3956      1403    5056      7613      N         N
-------------------------------------------------------------------------------
Pid      Command    Inuse     Pin     Pgsp      Virtual   64-bit    Mthrd
21166    dtpad      3822      1403    4717      7460      N         N
```

- Useful for memory audit – takes native heap into account

Java Diagnostics

Verbose GC

- Verbose GC is an option provided by the JVM runtime
- Provides a log of garbage collection activity
  - Interval between collections
  - Duration of collection
  - Compaction required
  - Memory size/memory freed/memory available
- Often recommended to have verbose GC enabled permanently in production
- There are tools to analyse the GC Log:
  - % time in GC, pause time, memory comsumption rate.

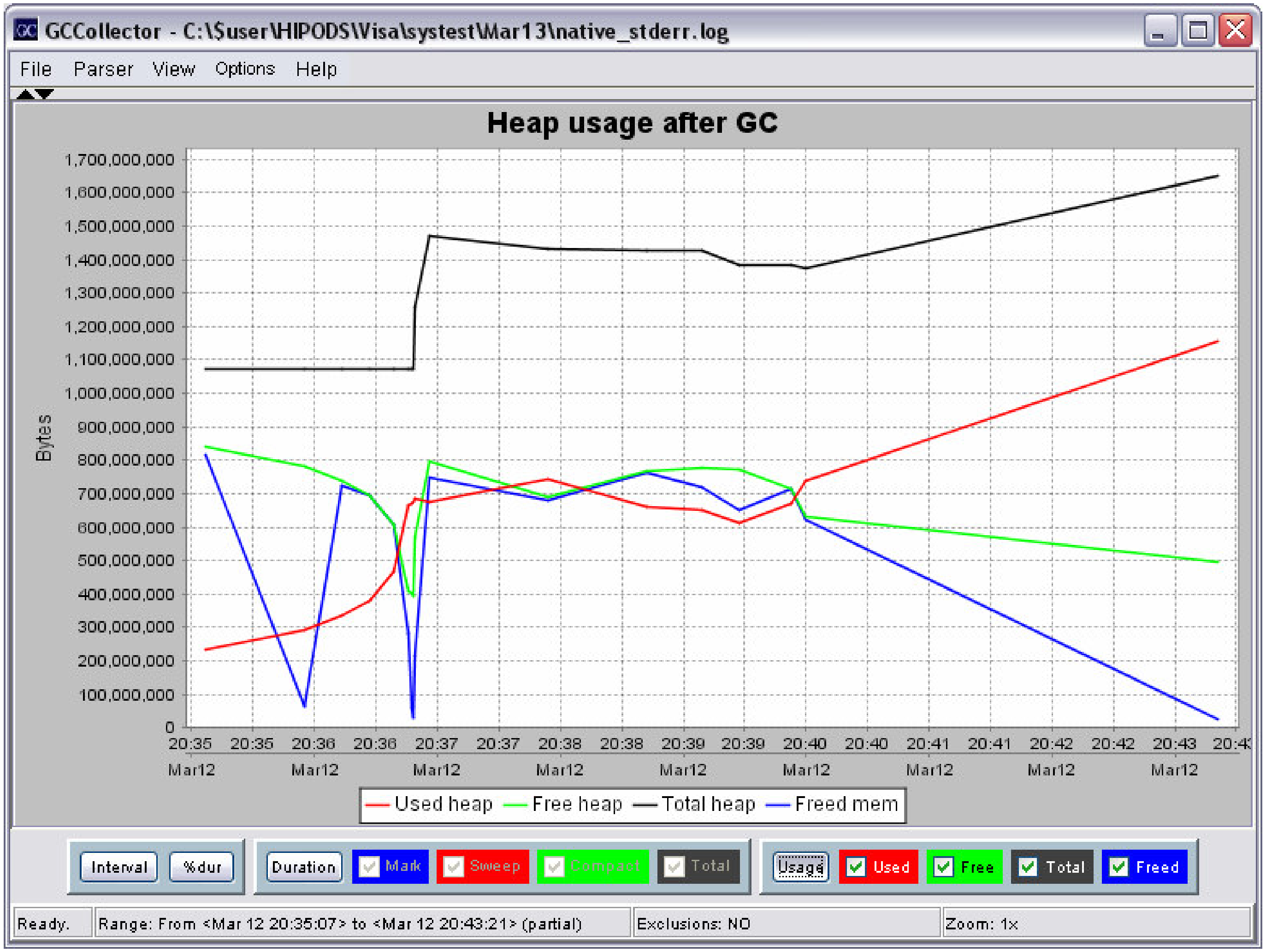
Verbose GC

- This log show a high rate of Garbage Collection

```
<af type="tenured" id="13" timestamp="Tue Apr 24 15:51:13 2007" intervalms="666.855">
  <minimum requested_bytes="4016" />
  <time exclusiveaccessms="0.134" />
  <tenured freebytes="6441984" totalbytes="1073741824" percent="0" >
    <soa freebytes="0" totalbytes="1067299840" percent="0" />
    <loa freebytes="6441984" totalbytes="6441984" percent="100" />
  </tenured>
  <gc type="global" id="13" totalid="13" intervalms="667.094">
    <refs_cleared soft="0" weak="0" phantom="0" />
    <finalization objectsqueued="0" />
    <timesms mark="231.870" sweep="7.244" compact="0.000" total="239.316" />
    <tenured freebytes="783784584" totalbytes="1073741824" percent="72" >
      <soa freebytes="778416776" totalbytes="1068374016" percent="72" />
      <loa freebytes="5367808" totalbytes="5367808" percent="100" />
    </tenured>
  </gc>
  <tenured freebytes="783780568" totalbytes="1073741824" percent="72" >
    <soa freebytes="778412760" totalbytes="1068374016" percent="72" />
    <loa freebytes="5367808" totalbytes="5367808" percent="100" />
  </tenured>
  <time totalms="239.689" />
</af>
```

- Increase Heap Size, Change GC Policy,

GC Collector tool



GCCollector - C:\$user\HIPODS\Visa\systest\Mar13\native_stderr.log

Ready.

Range: From <Mar 12 20:35:07> to <Mar 12 20:43:21> (partial)

Exclusions: NO

Zoom: 1x

Java Cores

- Standard JVM functionality

- Provides a snapshot of the JVM state. Key performance diagnosis sections are:
  - LOCKS (Monitors) section
  - THREADS dump
  - MEMINFO – memory segment allocation

- Generated by kill -3 {pid} or using wsadmin command line,
- Temporarily halts the JVM, which continues afterwards

- Allows Diagnosis of:

  Common locking points,

  Common execution bottlenecks

  Slow external systems – DB, others

JavaCores – Thread locking example

```
0SECTION       LOCKS subcomponent dump routine
NULL           =================================
....
2LKMONINUSE      sys_mon_t:0x11CDF3010 infl_mon_t: 0x11CDED0D0:
3LKMONOBJECT       java.lang.Class@700000009263E68/700000009263E78: owner
  "WorkManager.Test : 8" (0x11E7DACC8), entry count 1
3LKWAITERQ            Waiting to enter:
3LKWAITER                "WorkManager.Test : 9" (0x11E7D80C8)
3LKWAITER                "WorkManager.Test : 5" (0x11E5AC348)
3LKWAITER                "WorkManager.Test : 3" (0x11E5A24C8)
3LKWAITER                "WorkManager.Test : 1" (0x11E598548)
3LKWAITER                "WorkManager.Test : 0" (0x11E5930C8)
....
0SECTION       THREADS subcomponent dump routine
NULL           =================================
....
3XMTHREADINFO      "WorkManager.Test : 8" (TID:0x7000000000D28B0,
  sys_thread_t:0x11E7DACC8, state:R, native ID:0x7496) prio=5
4XESTACKTRACE          at sun.awt.font.NativeFontWrapper.initializeFont(Native Method)
4XESTACKTRACE          at java.awt.Font.initializeFont(Font.java(Compiled Code))
4XESTACKTRACE          at java.awt.Font.initFromMap(Font.java(Compiled Code))
4XESTACKTRACE          at java.awt.Font.<init>(Font.java(Compiled Code))
4XESTACKTRACE          at java.awt.Font.deriveFont(Font.java(Compiled Code))
4XESTACKTRACE          at jet.util.FontSet.getFont(FontSet(Compiled Code))
....

3XMTHREADINFO      "WorkManager.Test : 9" (TID:0x7000000000D27E0,
  sys_thread_t:0x11E7D80C8, state:MW, native ID:0x7597) prio=5
4XESTACKTRACE          at jet.util.FontSets.getFont(FontSets(Compiled Code))
4XESTACKTRACE          at jet.util.FontSets.getFont(FontSets(Compiled Code))
```

Javacore – Database bottleneck example

One Example had 23 Application threads, all showing the same stack trace:

```
3XMTHREADINFO      "Default : 6139" (TID:0x6E55A100, sys_thread_t:0x6C64DDA8, state:CW, native ID:0x6C64DDD8) prio=5
4XESTACKTRACE          at java/net/SocketInputStream.socketRead0(Native Method)
4XESTACKTRACE          at java/net/SocketInputStream.read(SocketInputStream.java:153)
4XESTACKTRACE          at com/ibm/db2/jcc/c/gb.b(gb.java:168)
4XESTACKTRACE          at com/ibm/db2/jcc/c/gb.c(gb.java:222)
4XESTACKTRACE          at com/ibm/db2/jcc/c/gb.c(gb.java:340)
4XESTACKTRACE          at com/ibm/db2/jcc/c/gb.v(gb.java:1441)
4XESTACKTRACE          at com/ibm/db2/jcc/c/jb.a(jb.java:63)
4XESTACKTRACE          at com/ibm/db2/jcc/c/w.a(w.java:48)
4XESTACKTRACE          at com/ibm/db2/jcc/c/dc.c(dc.java:312)
4XESTACKTRACE          at com/ibm/db2/jcc/a/id.cb(id.java:1685)
4XESTACKTRACE          at com/ibm/db2/jcc/a/id.b(id.java:2889)
4XESTACKTRACE          at com/ibm/db2/jcc/a/id.a(id.java:2704)
4XESTACKTRACE          at com/ibm/db2/jcc/a/id.executeBatch(id.java:2516)
4XESTACKTRACE          at com/ibm/db2/jcc/a/id.executeBatch(id.java:1348)
4XESTACKTRACE          at com/ibm/ws/rsadapter/jdbc/WSJdbcStatement.pmiExecuteBatch(WSJdbcStatement.java:1142)
```

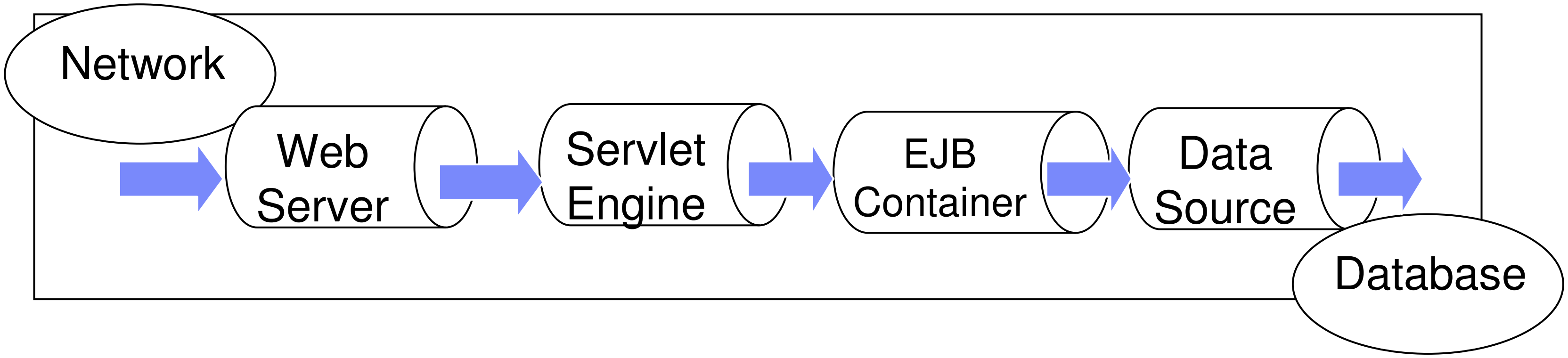
WAS Tools

Tools – PMI & TPV

- **Typical web app consists of static and dynamic web components, java components - sometimes EJBs, and a database element**
- **WAS has a built in performance monitoring infrastructure which allows these elements to be monitored**
- **WAS also has built in graphical PMI client in Tivoli Performance Viewer**

Tools – PMI & TPV

- WAS interrelated components need harmonious tuning to achieve maximum throughput while maintaining stability.
- These components are known as a queuing network.
- They include the network, Web server, Web container, EJB container, data source and possibly a connection manager to a custom back-end system.
- Each represents a queue of requests waiting to use that resource



Tools – TPV – Top 10 Monitoring Hotlist

Servlets and Enterprise JavaBeans

1. Average response time
Performance Modules -> **Web Applications** > *ServiceTime*.

2. Number of requests (Transactions)
Performance Modules -> **Web Applications** > *RequestCount*.

3. Live HTTP Sessions
Performance Modules -> **Servlet Session Manager** > *Live Count*.

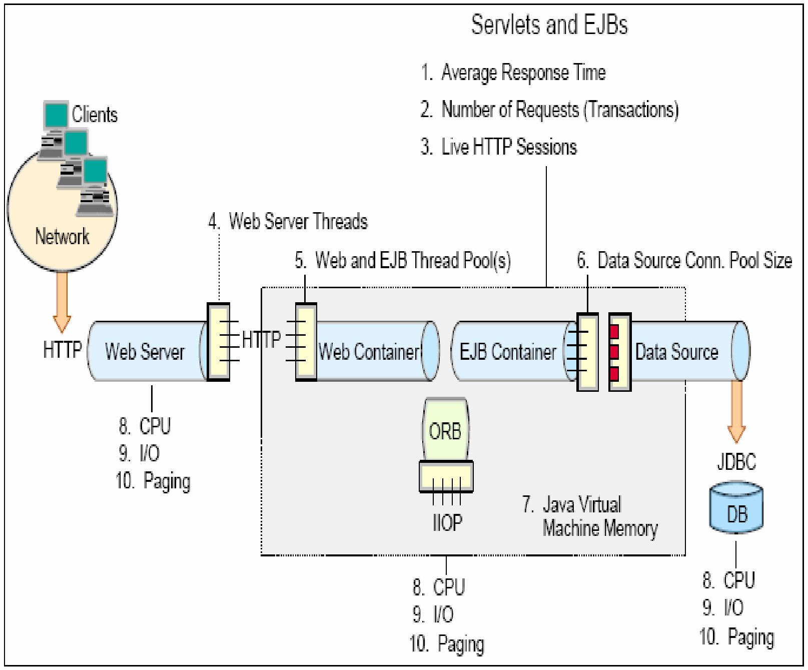


Figure 17-11 Top ten monitoring items checklist

Tools – TPV – Top 10 Monitoring Hotlist

Thread pools

4. Web server threads
Enable this in the Web Server

5. Web container and EJB
container thread pool

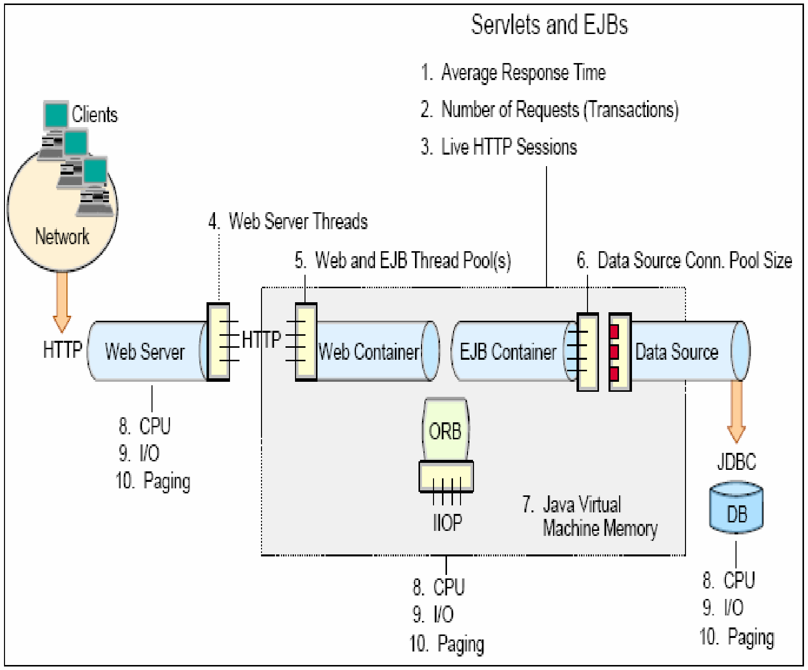


Figure 17-11 Top ten monitoring items checklist

Tools – TPV – Top 10 Monitoring Hotlist

Data sources

6. Datasource connection pool size

Java Virtual Machine

7. Memory, GC stats
Performance Modules -> **JVM Runtime**.

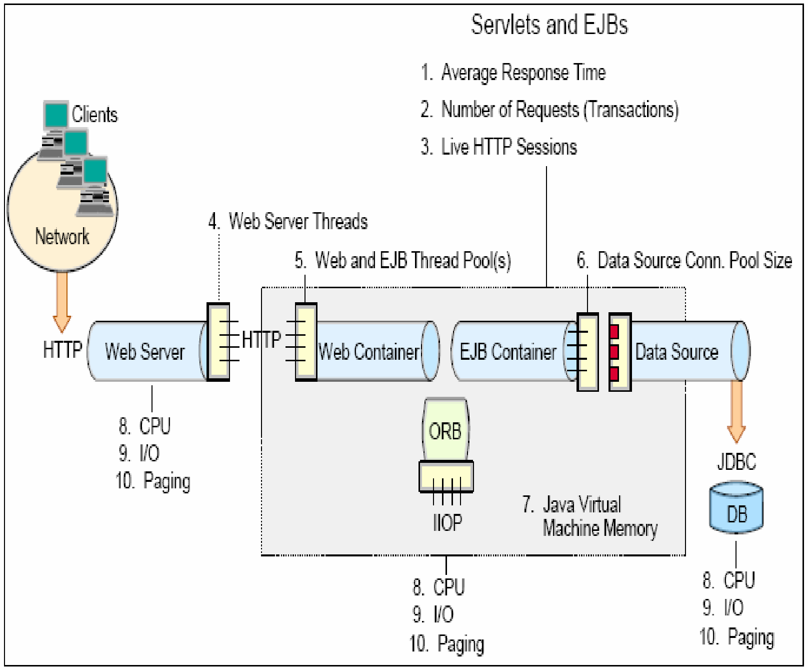


Figure 17-11 Top ten monitoring items checklist

Tools – TPV – Top 10 Monitoring Hotlist

System resources on Web, App and Db servers

8. CPU utilization
9. Disk and network I/O
10.Paging activity

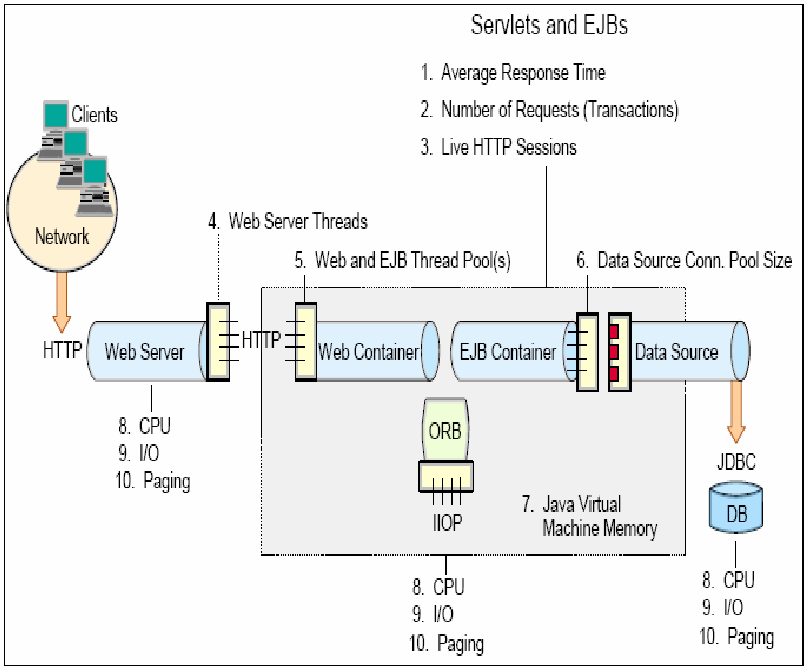


Figure 17-11 Top ten monitoring items checklist

Tools - ISA

- Other tools integrated into admin console;

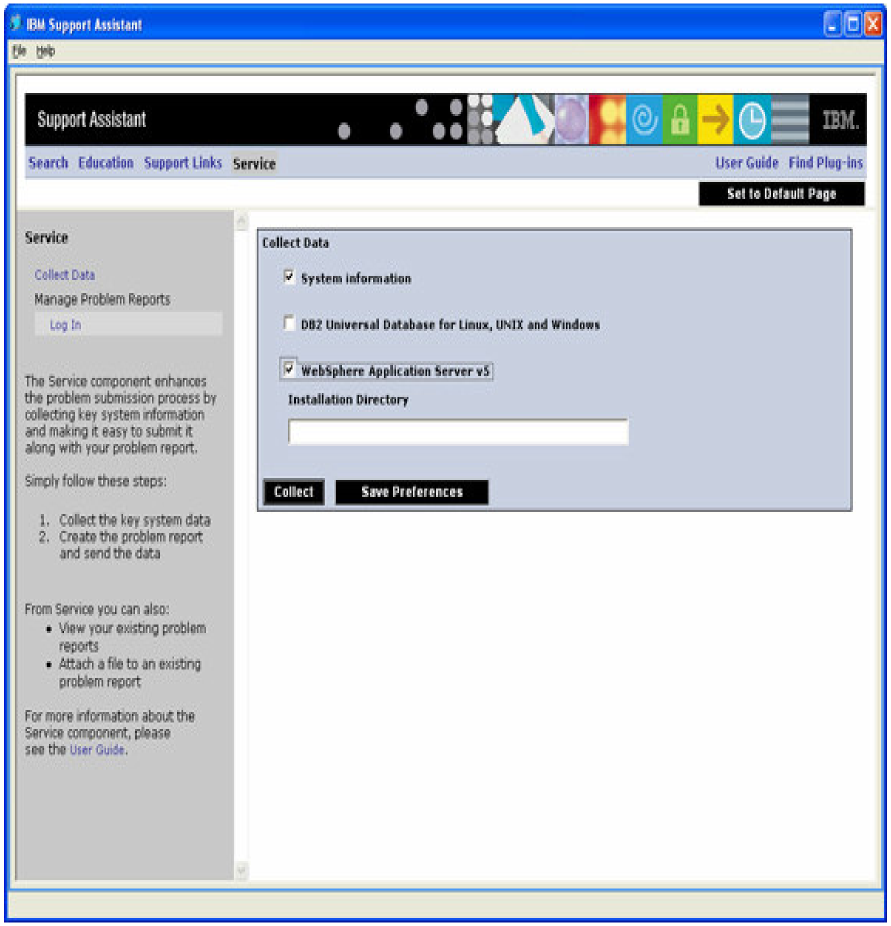
  Request Metrics - Enables tracking of transactions. Output to standard logs or Application Response Measurement (ARM) based tool

  Performance advisors - Analyse performance data and make recommendations to improve performance. Output to TPV or console runtime messages

Tools - ISA

- Other tools are shipped with WAS or as separate products;
  - GCCollector
  - PMAT
  - MDD4J

Tools - ISA



The Service component enhances the problem submission process by collecting key system information and making it easy to submit it along with your problem report.

Simply follow these steps:

1. Collect the key system data
2. Create the problem report and send the data

From Service you can also:

- View your existing problem reports
- Attach a file to an existing problem report

For more information about the Service component, please see the User Guide.

☑ System information

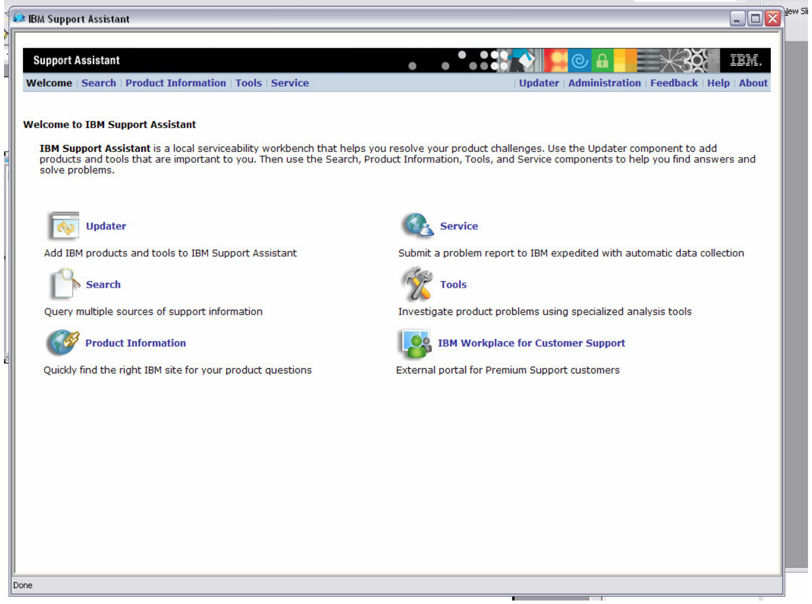
□ DB2 Universal Database for Linux, UNIX and Windows

☑ WebSphere Application Server v5

Installation Directory

IBM Support Assistant V3 (ISA) is and Eclipse base utility which serves as a central point from which many tools can be found

Tools - ISA


IBM Support Assistant
Support Assistant
IBM.
Welcome | Search | Product Information | Tools | Service
Updater | Administration | Feedback | Help | About
Done

Welcome to IBM Support Assistant

IBM Support Assistant is a local serviceability workbench that helps you resolve your product challenges. Use the Updater component to add products and tools that are important to you. Then use the Search, Product Information, Tools, and Service components to help you find answers and solve problems.

Updater

Add IBM products and tools to IBM Support Assistant

Service

Submit a problem report to IBM expedited with automatic data collection

Search

Query multiple sources of support information

Tools

Investigate product problems using specialized analysis tools

Product Information

Quickly find the right IBM site for your product questions

IBM Workplace for Customer Support

External portal for Premium Support customers

- ISA provides both services and tools
- Intended as a “one-stop shop”
- Contains
  - Search facility – infocentre, developerworks, google etc
  - Recommended links, fixes and forums
  - Repository of downloadable tools
  - Service – open PMRs, run collector tool
  - Updater – downloadable tools

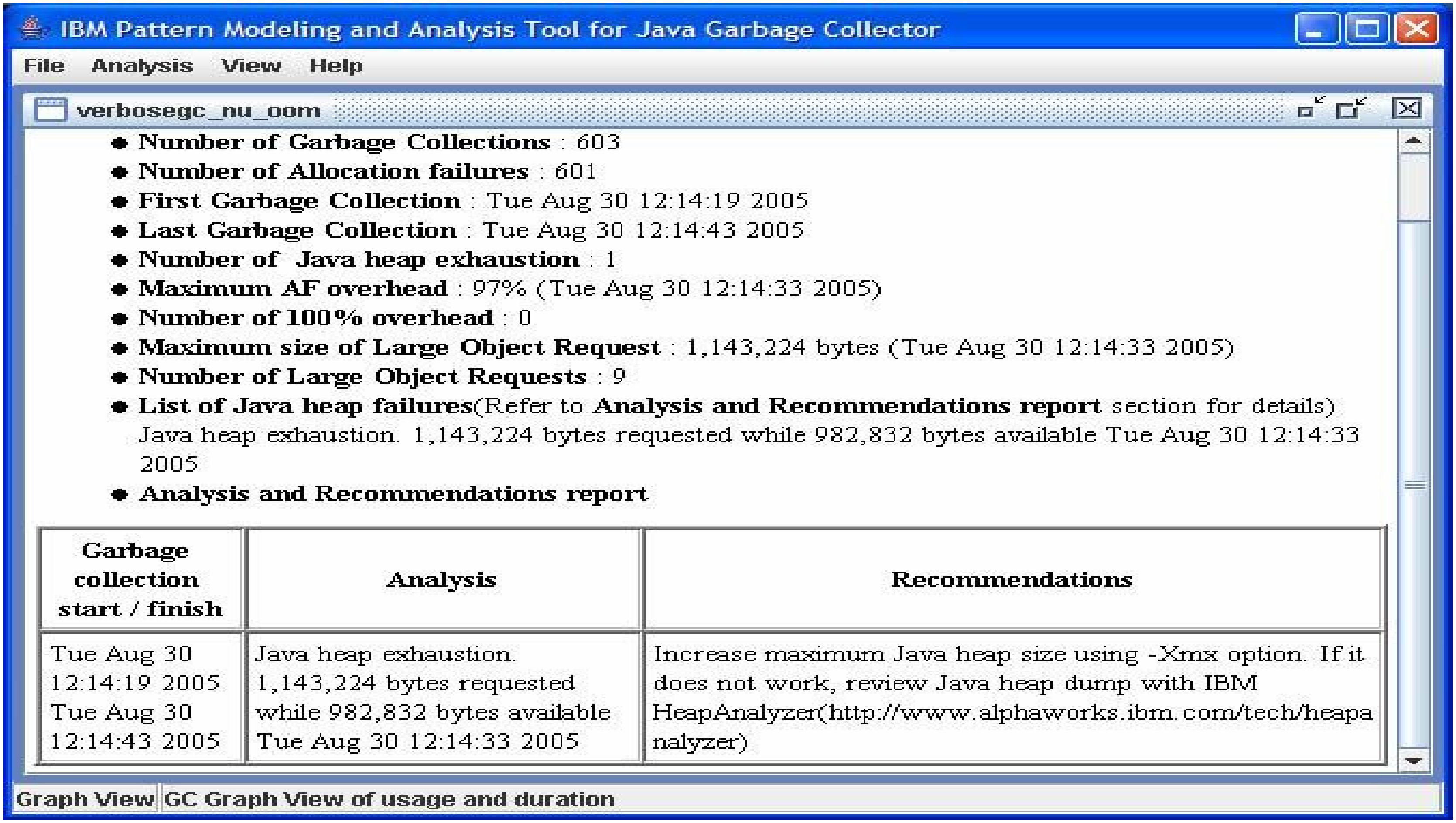
Tools – ISA – Memory

- Tools for memory analysis:
  - PMAT
  - GCCollector
- Can all map out memory usage and provide GC stats

Tools – ISA – Memory

- PMAT
  - –Analyzes a JVM verboseGC log to diagnose out-of-memory conditions
  - –Available in ISA or standalone download

Tools – ISA – Memory: PMAT example

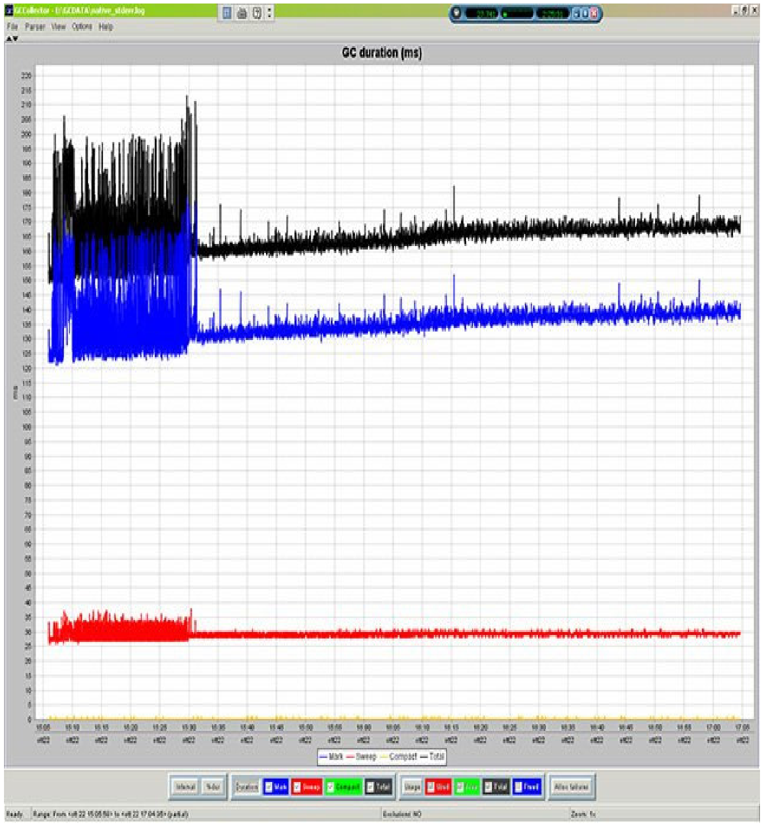

IBM Pattern Modeling and Analysis Tool for Java Garbage Collector
File
Analysis
View
Help
verbosegc_nu_oom
Graph View
GC Graph View of usage and duration

- **Number of Garbage Collections** : 603
- **Number of Allocation failures** : 601
- **First Garbage Collection** : Tue Aug 30 12:14:19 2005
- **Last Garbage Collection** : Tue Aug 30 12:14:43 2005
- **Number of Java heap exhaustion** : 1
- **Maximum AF overhead** : 97% (Tue Aug 30 12:14:33 2005)
- **Number of 100% overhead** : 0
- **Maximum size of Large Object Request** : 1,143,224 bytes (Tue Aug 30 12:14:33 2005)
- **Number of Large Object Requests** : 9
- **List of Java heap failures**(Refer to **Analysis and Recommendations report** section for details)
  Java heap exhaustion. 1,143,224 bytes requested while 982,832 bytes available Tue Aug 30 12:14:33 2005
- **Analysis and Recommendations report**

Garbage collection start / finish	**Analysis**	**Recommendations**
Tue Aug 30 12:14:19 2005 Tue Aug 30 12:14:43 2005	Java heap exhaustion. 1,143,224 bytes requested while 982,832 bytes available Tue Aug 30 12:14:33 2005	Increase maximum Java heap size using -Xmx option. If it does not work, review Java heap dump with IBM HeapAnalyzer(http://www.alphaworks.ibm.com/tech/heapanalyzer)

Tools – ISA – Memory: GC Collector

Graphs GC details e.g. this shows memory use during mark, sweep, compact phases and overall total



Tools – ISA – Hung Threads

- Hung Threads – root causes?
- Infinite loops
- Synchronised code deadlocks
- Don’t need to wait until WAS is entirely hung – use threadmonitor

Tools – ISA – Hung Threads

- WAS5 – ThreadMonitor architecture introduced
- Monitors WAS Pools: web container thread pool, ORB thread pool, Async Beans thread pool
- NOT threads started from code

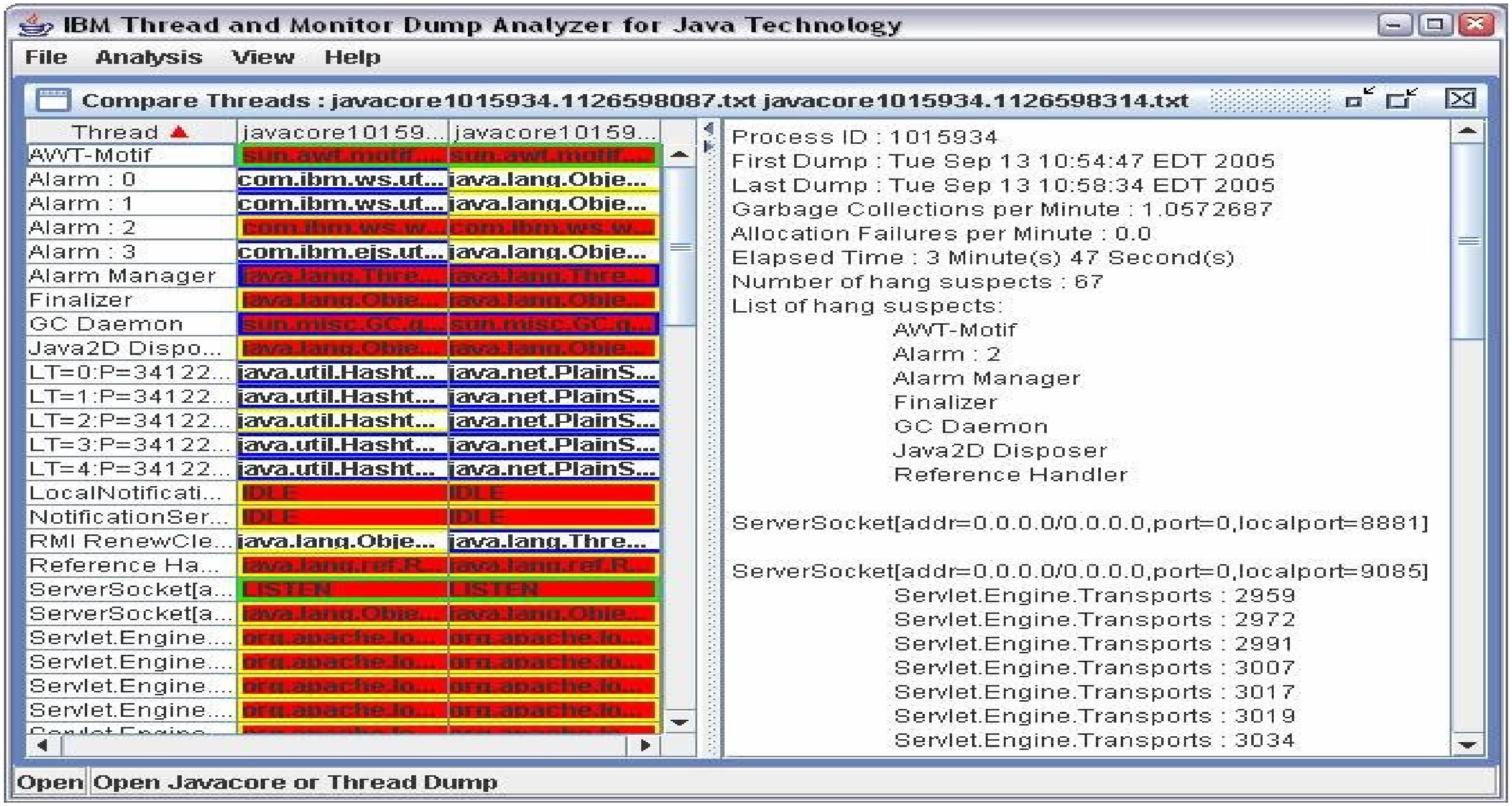
Tools – ISA – Hung Threads

- Administrator can determine how long a thread can run before being termed hung
- Default is 10 minutes and check interval of 3 minutes
- Can be reset (on the fly)
- Degradation less than 1%
- Notification sent to SystemOut.log, JMX listeners and PMI clients

Tools – ISA – Hung Threads: Action

- System sends notification of hung thread – what to do next?
- Generate a thread dump (javacore file)
- Perform some analysis
- Paul speaks Threadish but mere mortals can use ThreadAnalyzer (available in ISA or stand-alone)
- Look for the hung thread

Tools – ISA – ThreadAnalyser: Example



IBM Thread and Monitor Dump Analyzer for Java Technology
File
Analysis
View
Help
Compare Threads : javacore1015934.1126598087.txt javacore1015934.1126598314.txt

Thread ▲	javacore10159...	javacore10159...
AWT-Motif	sun.awt.motif....	sun.awt.motif....
Alarm : 0	com.ibm.ws.ut...	java.lang.Obje...
Alarm : 1	com.ibm.ws.ut...	java.lang.Obje...
Alarm : 2	com.ibm.ws.w...	com.ibm.ws.w...
Alarm : 3	com.ibm.ejs.ut...	java.lang.Obje...
Alarm Manager	java.lang.Thre...	java.lang.Thre...
Finalizer	java.lang.Obje...	java.lang.Obje...
GC Daemon	sun.misc.GC.g...	sun.misc.GC.g...
Java2D Dispo...	java.lang.Obje...	java.lang.Obje...
LT=0:P=34122...	java.util.Hasht...	java.net.PlainS...
LT=1:P=34122...	java.util.Hasht...	java.net.PlainS...
LT=2:P=34122...	java.util.Hasht...	java.net.PlainS...
LT=3:P=34122...	java.util.Hasht...	java.net.PlainS...
LT=4:P=34122...	java.util.Hasht...	java.net.PlainS...
LocalNotificati...	IDLE	IDLE
NotificationSer...	IDLE	IDLE
RMI RenewCle...	java.lang.Obje...	java.lang.Thre...
Reference Ha...	java.lang.ref.R...	java.lang.ref.R...
ServerSocket[a...	LISTEN	LISTEN
ServerSocket[a...	java.lang.Obje...	java.lang.Obje...
Servlet.Engine....	org.apache.io....	org.apache.io....
Servlet.Engine....	org.apache.io....	org.apache.io....
Servlet.Engine....	org.apache.io....	org.apache.io....
Servlet.Engine....	org.apache.io....	org.apache.io....

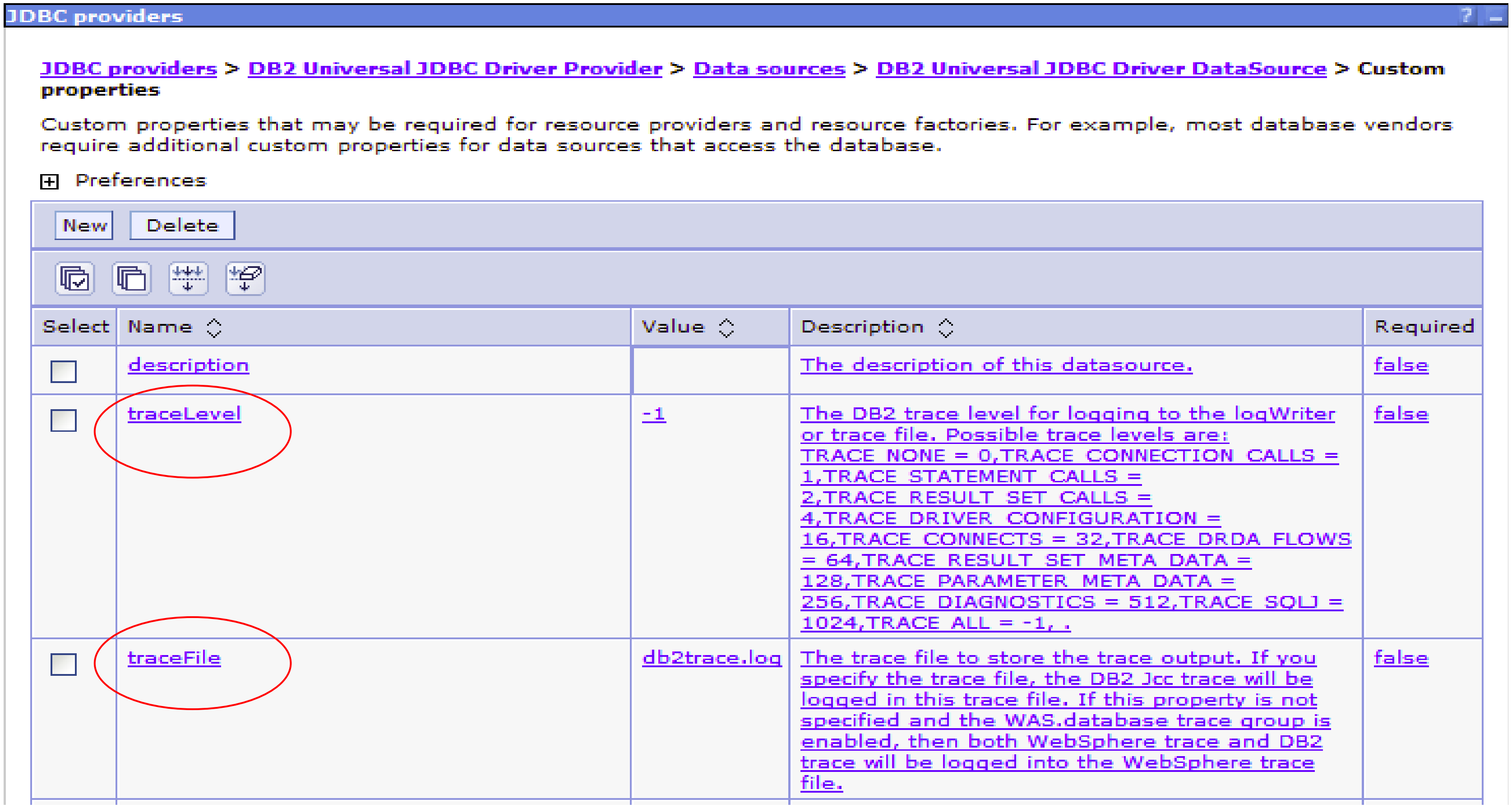
```
Process ID : 1015934
First Dump : Tue Sep 13 10:54:47 EDT 2005
Last Dump : Tue Sep 13 10:58:34 EDT 2005
Garbage Collections per Minute : 1.0572687
Allocation Failures per Minute : 0.0
Elapsed Time : 3 Minute(s) 47 Second(s)
Number of hang suspects : 67
List of hang suspects:
            AWT-Motif
            Alarm : 2
            Alarm Manager
            Finalizer
            GC Daemon
            Java2D Disposer
            Reference Handler

ServerSocket[addr=0.0.0.0/0.0.0.0,port=0,localport=8881]

ServerSocket[addr=0.0.0.0/0.0.0.0,port=0,localport=9085]
            Servlet.Engine.Transports : 2959
            Servlet.Engine.Transports : 2972
            Servlet.Engine.Transports : 2991
            Servlet.Engine.Transports : 3007
            Servlet.Engine.Transports : 3017
            Servlet.Engine.Transports : 3019
            Servlet.Engine.Transports : 3034
```

Database Tools

Database Tools

- Industry strength db drivers will have a trace or debug facility.
- The DB2 Universal Driver has trace capabilities that can be activated from the WAS admin console.

Database Tools



JDBC providers

JDBC providers > DB2 Universal JDBC Driver Provider > Data sources > DB2 Universal JDBC Driver DataSource > **Custom properties**

Custom properties that may be required for resource providers and resource factories. For example, most database vendors require additional custom properties for data sources that access the database.

⊞ Preferences

New Delete

Select	Name	Value	Description	Required
□	description		The description of this datasource.	false
□	traceLevel	-1	The DB2 trace level for logging to the logWriter or trace file. Possible trace levels are: TRACE_NONE = 0,TRACE_CONNECTION_CALLS = 1,TRACE_STATEMENT_CALLS = 2,TRACE_RESULT_SET_CALLS = 4,TRACE_DRIVER_CONFIGURATION = 16,TRACE_CONNECTS = 32,TRACE_DRDA_FLOWS = 64,TRACE_RESULT_SET_META_DATA = 128,TRACE_PARAMETER_META_DATA = 256,TRACE_DIAGNOSTICS = 512,TRACE_SQLJ = 1024,TRACE_ALL = -1, .	false
□	traceFile	db2trace.log	The trace file to store the trace output. If you specify the trace file, the DB2 Jcc trace will be logged in this trace file. If this property is not specified and the WAS.database trace group is enabled, then both WebSphere trace and DB2 trace will be logged into the WebSphere trace file.	false

DB2 Trace Example

```
[ibm][db2][jcc][Time:1143567094964][Thread:WebContainer :
0][DB2ConnectionPoolDataSource@59247092] getPooledConnection (UserName, <escaped>) called
[ibm][db2][jcc] BEGIN TRACE_DRIVER_CONFIGURATION
[ibm][db2][jcc] Driver: IBM DB2 JDBC Universal Driver Architecture 2.4.19
[ibm][db2][jcc] Compatible JRE versions: { 1.3, 1.4 }…
[ibm][db2][jcc] END TRACE_DRIVER_CONFIGURATION
[ibm][db2][jcc] BEGIN TRACE_CONNECTS
[ibm][db2][jcc] Attempting connection to localhost:50001/ENTDBx
[ibm][db2][jcc] Using properties: { cliSchema=null, clientProgramId=null,…currentPackagePath=null,
portNumber=50001, serverName=localhost,…traceFile=db2trace.log, useCachedCursor=true,
dataSourceName=null, fullyMaterializeLobData=true, databaseName=ENTDBx,
kerberosServerPrincipal=null, jdbcCollection=NULLID, clientUser=null, traceLevel=-1, currentRefreshAge=-
…
[ibm][db2][jcc] END TRACE_CONNECTS…
[ibm][db2][jcc] BEGIN TRACE_DIAGNOSTICS
[ibm][db2][jcc][SQLException@65877092] java.sql.SQLException
[ibm][db2][jcc][SQLException@65877092] SQL state = 08004
[ibm][db2][jcc][SQLException@65877092] Error code = -4499
[ibm][db2][jcc][SQLException@65877092] Message = The application server rejected establishment of the
connection. An attempt was made to access a database, ENTDBx, which was not found.
[ibm][db2][jcc][SQLException@65877092] Stack trace followscom.ibm.db2.jcc.a.DisconnectException: The
application server rejected establishment of the connection. An attempt was made to access a database,
ENTDBx, which was not found.at com.ibm.db2.jcc.c.cb.u(cb.java:1595)
```

Common Performance Problems

Common Problems

- Excessive Object Allocation
- DB Access
- Logging
- Calls to slow external systems
- Shared data structures
- Un-cached reference data

References

- SC34-6650-05 - IBM Developer Kit and Runtime Environment, Java Technology Edition, Version 5.0 Diagnostics Guide
- SG246392 - WebSphere Application Server V6 Scalability and Performance Handbook
- SC23-4905-04 AIX 5L Version 5.3 Performance management

Questions