

More Coherence War Stories

Patrick Peralta, Oracle

Have you ever seen this?

Experienced a 4811 ms
communication delay

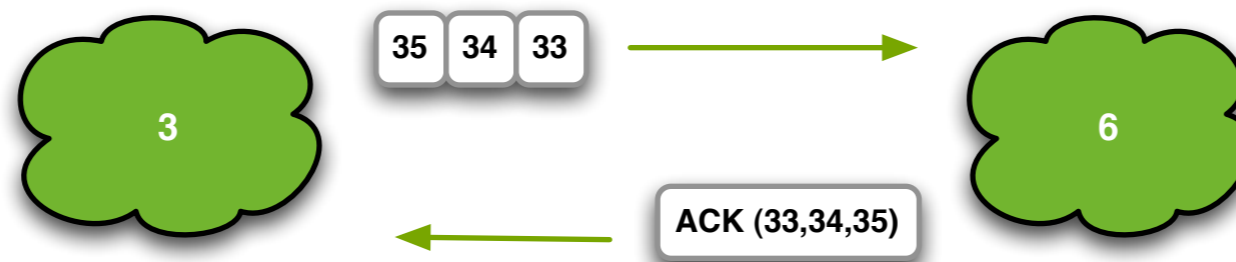
Or this?

Timeout while
delivering a packet

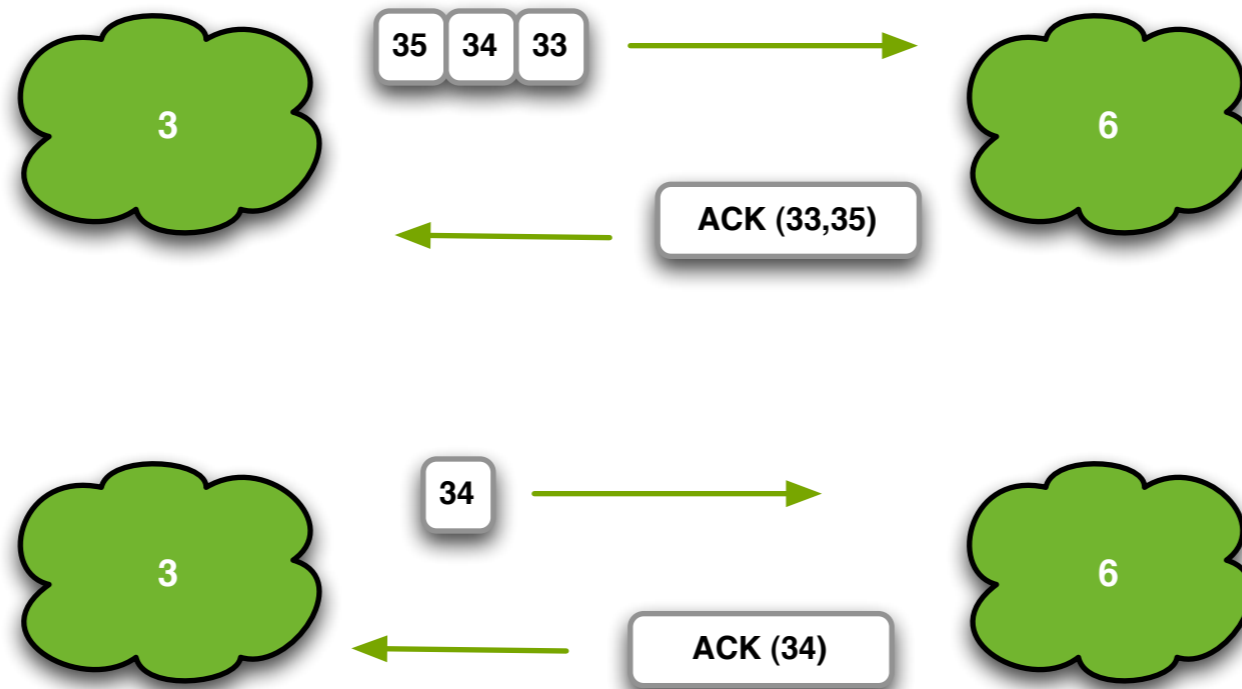
Why does it happen?



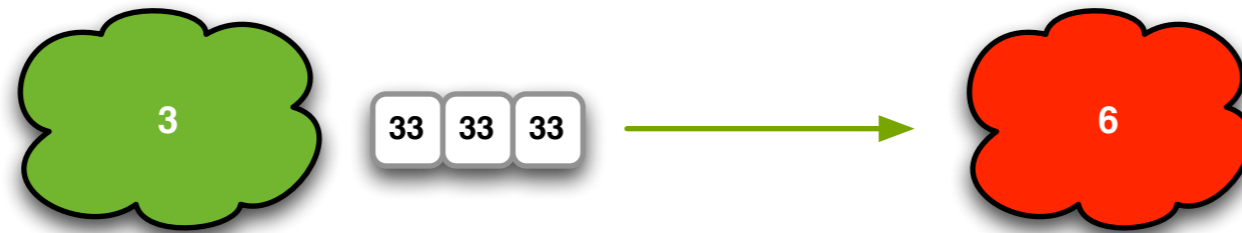
Packet Delivery



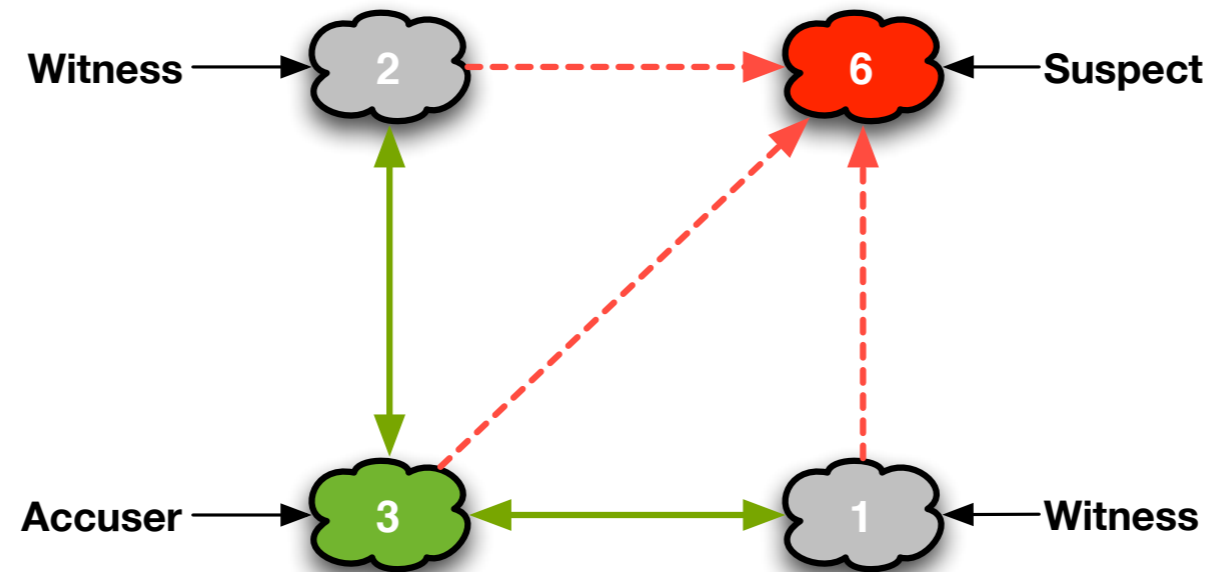
Packet Delay



Packet Timeout



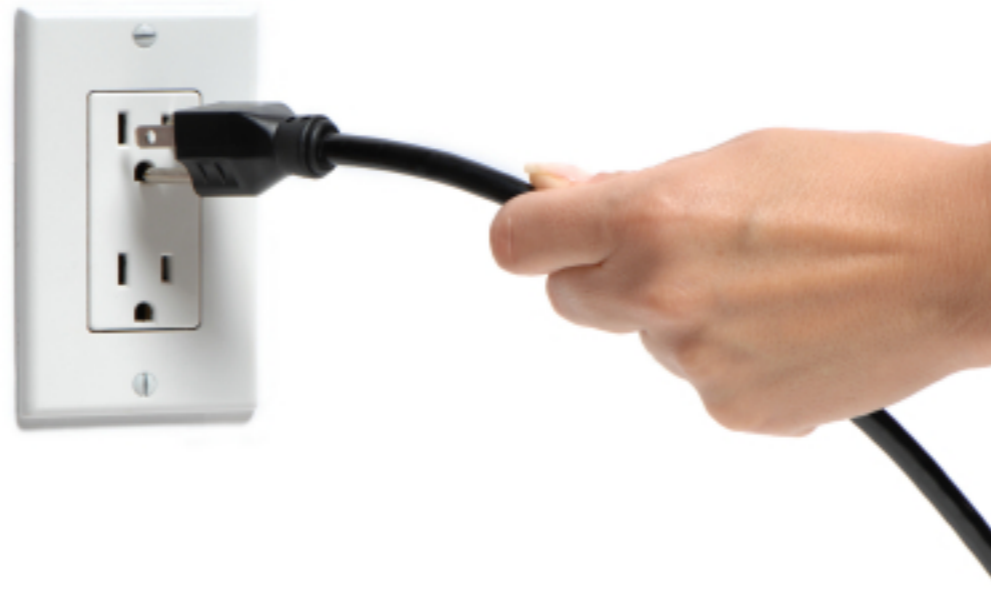
Witness Protocol



Why does it happen?

- Packet delivery failure
- But what are the common root causes?

Network Disconnect



“Let’s take this off-line”

- Customer had communication delays and timeouts every night at the same time
- It turned out that firewall rules are configured every night which resulted in a network disconnect

Lessons Learned

- Avoid assumptions about the network
- TALK to your network infrastructure team
- Run the datagram test to ensure your network will provide adequate performance for your application

Network Bandwidth Exceeded



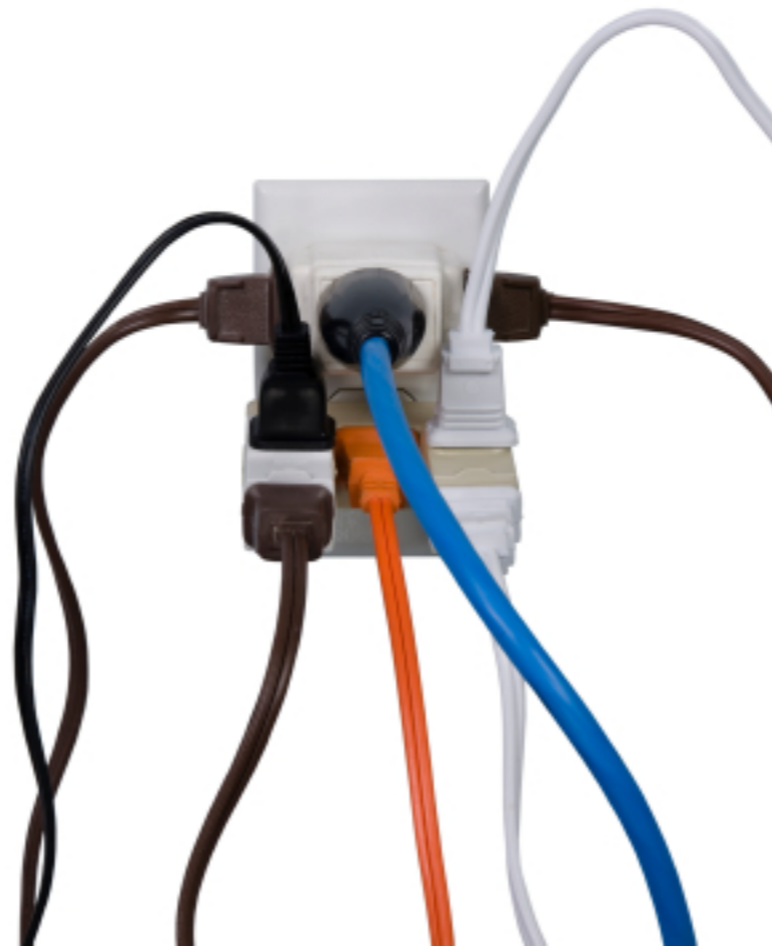
“The perfect storm”

- 120 node cluster
- Near cache with invalidation strategy “all”
- Large keys (100K)
- Calling “clear” caused flood of invalidation events, exceeding network capacity

Lessons Learned

- Monitor the amount of traffic passing through the network interface
- Coherence assumes small keys; don't use large objects as keys
- Avoid “heavy” map listeners that listen to all events, consider a “lite” listener and/or MapEvent filters

Extreme CPU Consumption



“Missing index”

- A missing index was not detected during application testing
- In production, the missing index resulted in very high CPU usage

Lessons Learned

- Ensure that indexes are created for filters
- Monitor CPU usage on the box
 - High CPU usage from other processes may also result in packet loss

Swapping



“Tag team”

- A commerce customer uses two application server “clusters” for their site
- While one of the clusters is live, the other cluster is updated with site changes
- At midnight, the load balancer switches to the “passive” cluster and the “active” cluster is shut down

“Tag team”

- Each box contained two application servers
 - one for each cluster
- There wasn't enough physical memory to run both at the same time
- When both servers were running during switchover at midnight, Coherence would report packet loss

Why is swapping bad?

- GC algorithms scan the entire heap and move objects around
 - Generational GC
 - Compaction / defragmentation
- If the collector has to wait for memory to be paged in from disk, it will take longer for the collector to finish

GC while swapping

```
2008/09/17 10:19:53 | [GC 931996K->856503K(1017024K), 0.0758996 secs]
2008/09/17 10:19:55 | [GC 659228K->589962K(1014848K), 0.0395841 secs]
2008/09/17 10:19:59 | [GC 931006K->859644K(1020928K), 0.0287045 secs]
2008/09/17 10:20:26 | [GC 938176K->865107K(1021376K), 19.7179554 secs]
2008/09/17 10:20:29 | [GC 660588K->591451K(1014912K), 2.5831922 secs]
2008/09/17 10:20:33 | [GC 943524K->861061K(1030528K), 3.8399128 secs]
2008/09/17 10:20:34 | [GC 947829K->863184K(1030208K), 0.0404099 secs]
```

Lessons Learned

- Ensure that JVM memory allocation does not exceed physical memory
- Remember that JVM heap size < JVM memory footprint

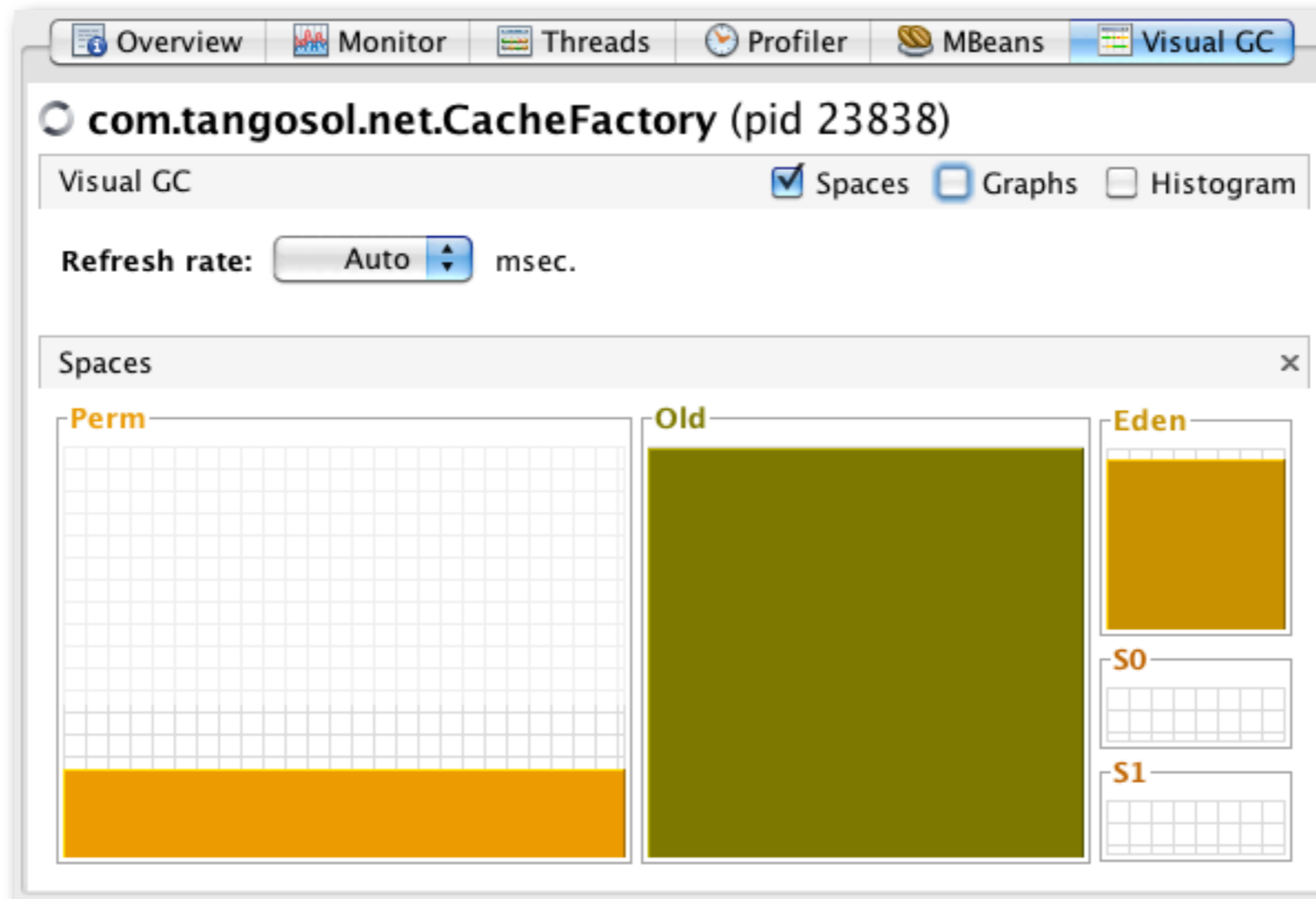
Garbage Collection



Reduce, Reuse, Recycle

- Modern 1.6 VMs are very good at GC
- Most trouble is caused by running with a full heap (> 66%)
- Full heaps cause the garbage collector to work harder

Full Heap



GC's cousin: OOME

- `OutOfMemoryError` is a common cause for havoc in production environment
- It may be caused by
 - Running out of heap
 - Running out of off-heap (NIO) storage

Causes of heap OOME we've seen

- Filling the cache and/or near cache without a proper high-units setting
- Application memory leaks
- Slow consumers causing a queue backlog
- Accessing a HashMap from multiple threads without synchronization

What to do with OOM

- If OOM is thrown because heap is full, don't waste your time *speculating why!*
- Instead, generate a heap dump - this is the fastest way to resolve OOM issues

Heap Dump

Eclipse Memory Analyzer

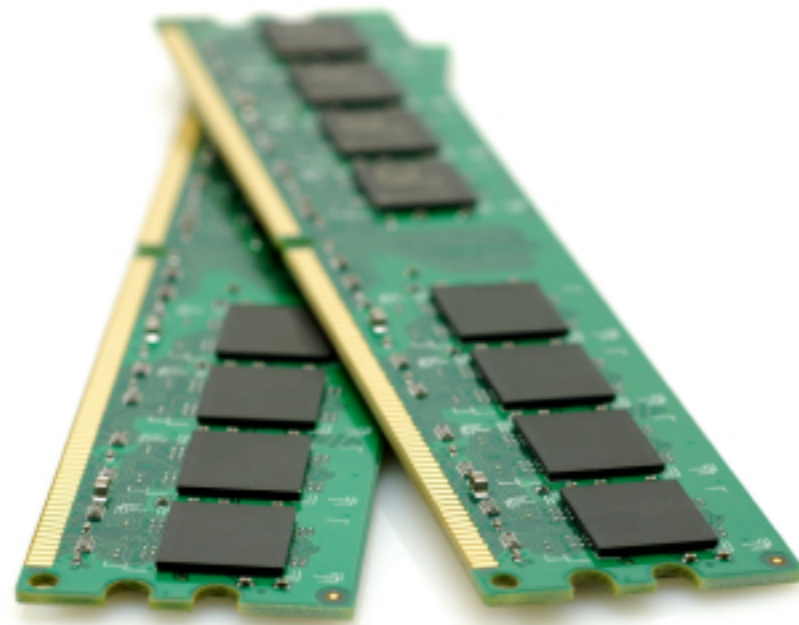
java_pid23310.hprof

Overview dominator_tree

Class Name	Shallow Heap	Retained Heap	Percentage
<Regex>	<Numeric>	<Numeric>	<Numeric>
com.tangosol.net.cache.LocalCache @ 0x107132d	256	73,347,960	86.26%
com.tangosol.util.SafeHashMap\$Entry[100003]	800,048	69,650,144	81.91%
com.tangosol.net.cache.LocalCache\$Entry @	104	10,584	0.01%
com.tangosol.net.cache.LocalCache\$Entr	104	9,408	0.01%
com.tangosol.util.Binary @ 0x106aa5248	40	1,072	0.00%
Σ Total: 2 entries			
com.tangosol.net.cache.LocalCache\$Entry @	104	7,056	0.01%
com.tangosol.net.cache.LocalCache\$Entry @	104	7,056	0.01%
com.tangosol.net.cache.LocalCache\$Entry @	104	7,056	0.01%
com.tangosol.net.cache.LocalCache\$Entry @	104	5,880	0.01%
com.tangosol.net.cache.LocalCache\$Entry @	104	5,880	0.01%
com.tangosol.net.cache.LocalCache\$Entry @	104	5,880	0.01%
com.tangosol.net.cache.LocalCache\$Entry @	104	5,880	0.01%
com.tangosol.net.cache.LocalCache\$Entry @	104	5,880	0.01%
com.tangosol.net.cache.LocalCache\$Entry @	104	5,880	0.01%
com.tangosol.net.cache.LocalCache\$Entry @	104	5,880	0.01%
com.tangosol.net.cache.LocalCache\$Entry @	104	5,880	0.01%
com.tangosol.net.cache.LocalCache\$Entry @	104	5,880	0.01%

45M of 125M

Lessons Learned



JVM

- Upgrade to JDK 1.6
 - Improved GC
 - Heaps can be larger (4 to 6 GB)
 - **But don't fill them to capacity!**

JVM Flags

- Log verbose GC - with timestamps
- Generate heap dump when an `OutOfMemoryError` is thrown
- JVM kill switch `OutOfMemoryError`
- See your JVM's manual or the Coherence production checklist for specific flags

Summary of Best Practices

- Coherence Logging
- JMX / JMX Reporter
- OS Monitoring



Log files

- Some problems with Coherence require multiple logs to track down and resolve
- Best practice: submit log files for the entire cluster when submitting a service request
- Don't just cherry pick the lines you think are important!

Log files

- Logs go to STDERR by default
- If running a stand alone JVM, consider using Log4j to rotate logs
- Don't combine log files from different cluster members - this makes it harder for support to analyze

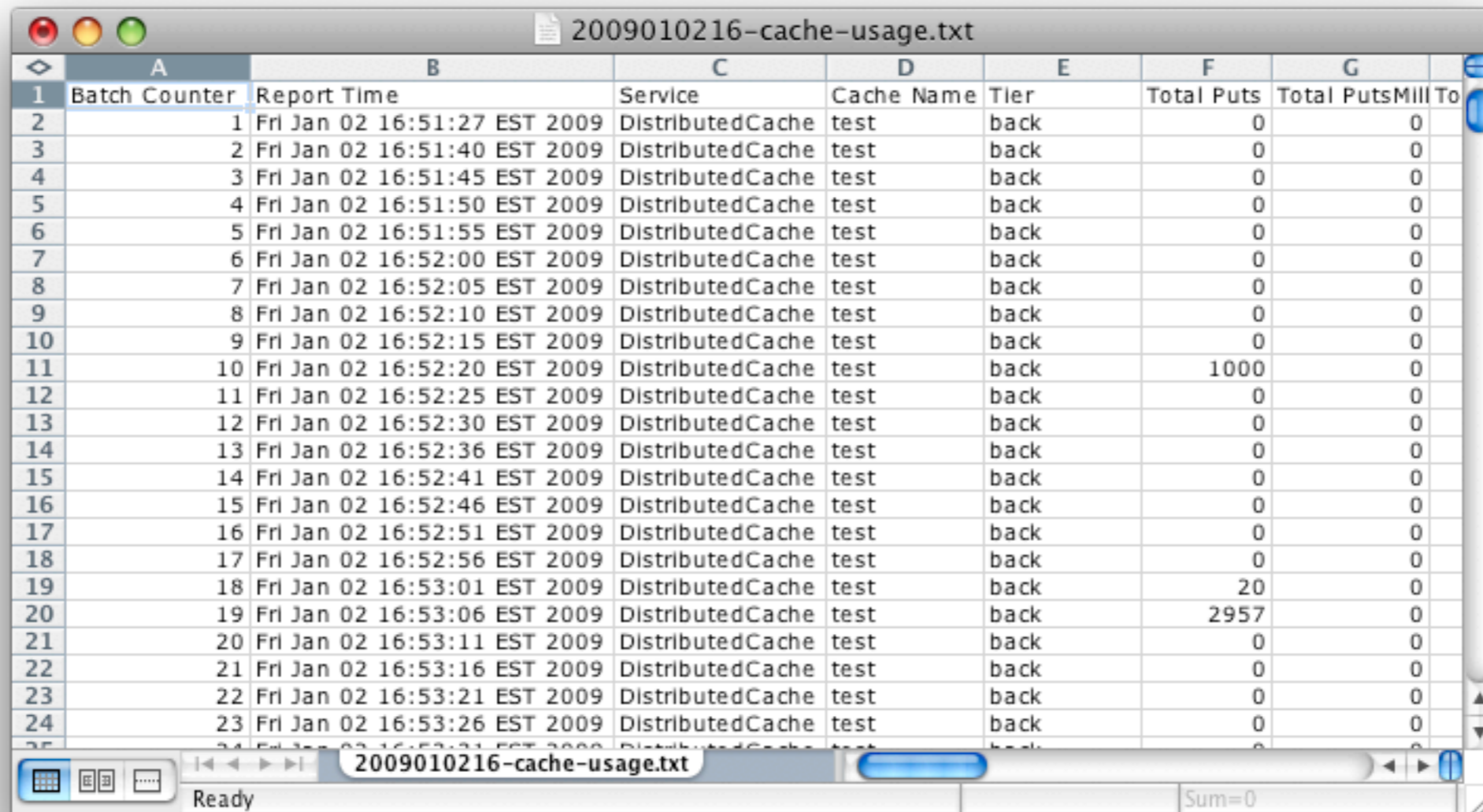
Script files

- You should have script files to
 - Gather logs from the entire cluster
 - Generate thread dumps on demand
 - Always take more than one

JMX Reporter

- JMX Reporter is a feature in Coherence that will log JMX statistics to CSV files
- This information can be very useful in diagnosing cluster issues

JMX Reporter



The screenshot shows a window titled "2009010216-cache-usage.txt" displaying a table of cache usage data. The table has columns for Batch Counter, Report Time, Service, Cache Name, Tier, Total Puts, and Total PutsMill To. The data shows a series of batches for a "DistributedCache" service, with a significant spike in "Total Puts" at batch 19 (2957) and batch 11 (1000). The status bar at the bottom indicates "Ready" and "Sum=0".

Batch Counter	Report Time	Service	Cache Name	Tier	Total Puts	Total PutsMill To
1	Fri Jan 02 16:51:27 EST 2009	DistributedCache	test	back	0	0
2	Fri Jan 02 16:51:40 EST 2009	DistributedCache	test	back	0	0
3	Fri Jan 02 16:51:45 EST 2009	DistributedCache	test	back	0	0
4	Fri Jan 02 16:51:50 EST 2009	DistributedCache	test	back	0	0
5	Fri Jan 02 16:51:55 EST 2009	DistributedCache	test	back	0	0
6	Fri Jan 02 16:52:00 EST 2009	DistributedCache	test	back	0	0
7	Fri Jan 02 16:52:05 EST 2009	DistributedCache	test	back	0	0
8	Fri Jan 02 16:52:10 EST 2009	DistributedCache	test	back	0	0
9	Fri Jan 02 16:52:15 EST 2009	DistributedCache	test	back	0	0
10	Fri Jan 02 16:52:20 EST 2009	DistributedCache	test	back	1000	0
11	Fri Jan 02 16:52:25 EST 2009	DistributedCache	test	back	0	0
12	Fri Jan 02 16:52:30 EST 2009	DistributedCache	test	back	0	0
13	Fri Jan 02 16:52:36 EST 2009	DistributedCache	test	back	0	0
14	Fri Jan 02 16:52:41 EST 2009	DistributedCache	test	back	0	0
15	Fri Jan 02 16:52:46 EST 2009	DistributedCache	test	back	0	0
16	Fri Jan 02 16:52:51 EST 2009	DistributedCache	test	back	0	0
17	Fri Jan 02 16:52:56 EST 2009	DistributedCache	test	back	0	0
18	Fri Jan 02 16:53:01 EST 2009	DistributedCache	test	back	20	0
19	Fri Jan 02 16:53:06 EST 2009	DistributedCache	test	back	2957	0
20	Fri Jan 02 16:53:11 EST 2009	DistributedCache	test	back	0	0
21	Fri Jan 02 16:53:16 EST 2009	DistributedCache	test	back	0	0
22	Fri Jan 02 16:53:21 EST 2009	DistributedCache	test	back	0	0
23	Fri Jan 02 16:53:26 EST 2009	DistributedCache	test	back	0	0

OS Monitoring

- OS monitoring is a **must!**
 - CPU
 - Swap file usage (should be **0%**)
 - NIC utilization

Unix/Linux

- top
- vmstat

```
Default (80,25)
[pperalta@ellis pperalta]$ vmstat 5
procs
r b swpd free buff cache si so bi bo in cs us sy wa id
1 0 936 26064 50036 166512 0 0 26 27 116 27 10 1 0 89
1 0 936 21864 50044 166512 0 0 0 6 104 19 99 1 0 0
1 0 936 18848 50044 166512 0 0 0 0 102 21 99 1 0 0
1 0 936 9112 50044 166512 0 0 0 0 101 11 98 2 0 0
1 0 936 5032 38736 166512 0 0 0 0 103 63 87 3 0 0
1 0 936 16:08:21 up 4:17, 3 users, load average: 1.02, 1.06, 0.91
1 0 936 56 processes: 54 sleeping, 2 running, 0 zombie, 0 stopped
1 0 936 CPU states: 95.0% user, 0.0% nice, 0.0% system, 0.0% irq, 0.0% softirq, 0.0% iowait, 0.0% idle
1 0 936 18020 37 total 5080 0.0% 0.0% 0.0% 33.2% 33.4% 33.3% 0.0%
Mem: 514616k av, 486592k used, 28024k free, 0k shrd, 25492k buff
337604k active, 47236k inactive
Swap: 1044216k av, 1056k used, 1043160k free 152556k cached

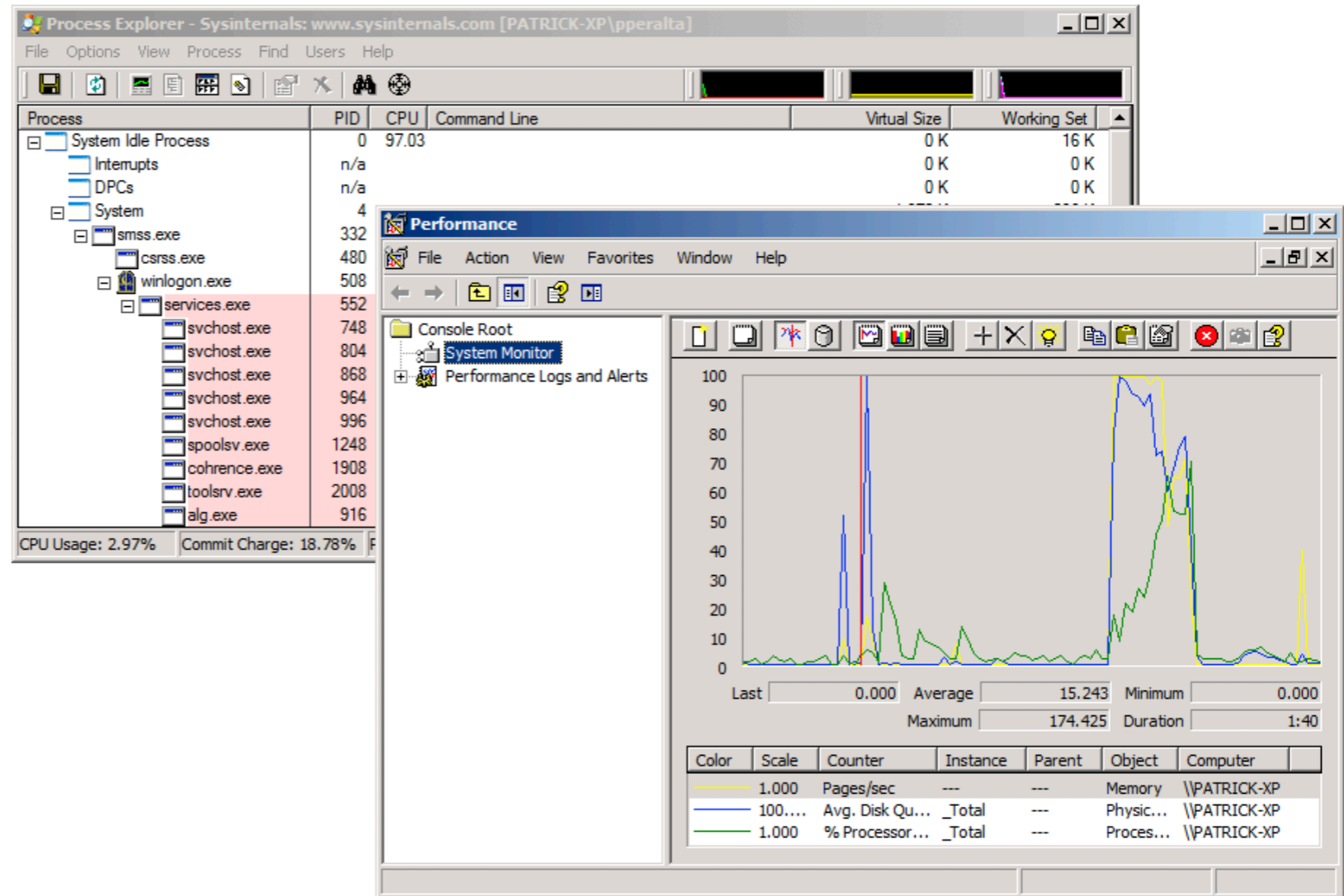
  PID USER   PRI  NI  SIZE  RSS  SHARE STAT  %CPU  %MEM  TIME CPU  COMMAND
20960 pperalta 25   0 165M 165M 2852 R    99.1  32.8  4:48  0  cc1
21015 pperalta 17   0 1152 1152  888 R     0.7   0.2   0:00  0  top
   1 root    16   0  420  420  360 S     0.0   0.0   0:06  0  init
   2 root    15   0    0    0    0 SW     0.0   0.0   0:00  0  keventd
   3 root    15   0    0    0    0 SW     0.0   0.0   0:00  0  kapmd
   4 root    34  19    0    0    0 SWN    0.0   0.0   0:00  0  ksoftirqd/0
   6 root    25   0    0    0    0 SW     0.0   0.0   0:00  0  bdflush
   5 root    16   0    0    0    0 SW     0.0   0.0   0:01  0  kswapd
   7 root    15   0    0    0    0 SW     0.0   0.0   0:00  0  kupdated
   8 root    21   0    0    0    0 SW     0.0   0.0   0:00  0  mdrecoveryd
  12 root    15   0    0    0    0 SW     0.0   0.0   0:00  0  kjournald
  74 root    15   0    0    0    0 SW     0.0   0.0   0:00  0  khubd
1063 root    15   0    0    0    0 SW     0.0   0.0   0:00  0  kjournald
1434 root    16   0  600  600  516 S     0.0   0.1   0:00  0  syslogd
1438 root    16   0  372  372  312 S     0.0   0.0   0:00  0  klogd
1466 rpc     16   0  564  564  492 S     0.0   0.1   0:00  0  portmap
```

SAR and kSar



Windows

- perfmon
- sysinternals



Conclusion

- JVMs don't work in a vacuum - be aware of their surroundings!
- Monitoring of JVM and OS is key
- The more data you capture, the better

Tools

- Eclipse MAT (<http://www.eclipse.org/mat/>)
- JVisualVM (<https://visualvm.dev.java.net/>)
- sar (<http://pagesperso-orange.fr/sebastien.godard/>)
- kSar (<http://ksar.atomique.net/>)