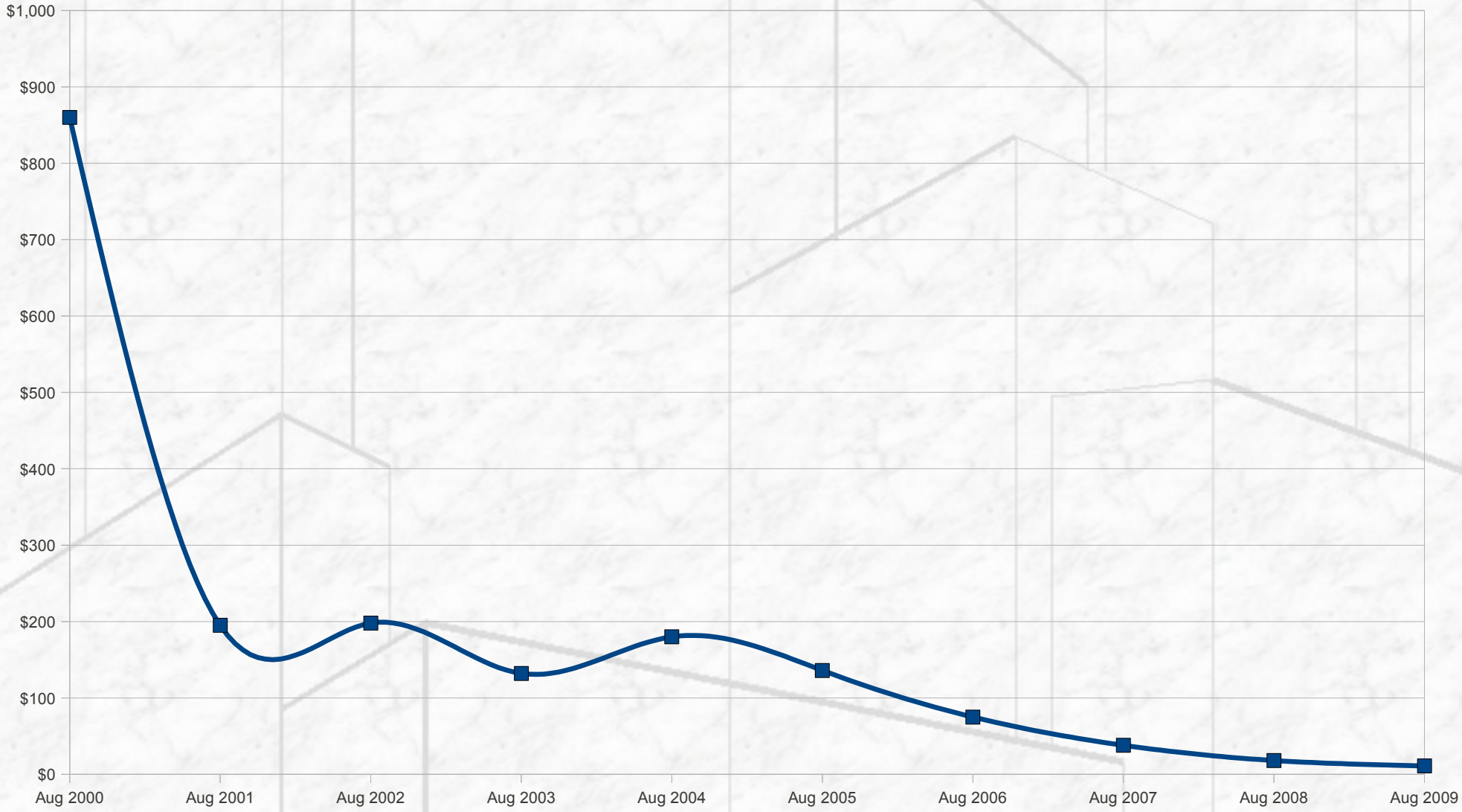
When the Kernel Runs Out of Memory

David Rientjes
Google, Inc

August 10, 2010

Cost per Gigabyte RAM



Source: jcmit.com

Cost per Gigabyte RAM

Aug 1990	$83,072
Aug 1991	$42,240
Aug 1992	$31,744
Aug 1993	$30,720
Aug 1994	$37,888
Aug 1995	$31,334
Aug 1996	$9,277
Aug 1997	$4,229
Aug 1998	$1,055
Aug 1999	$845

Aug 2000	$860
Aug 2001	$195
Aug 2002	$198
Aug 2003	$132
Aug 2004	$180
Aug 2005	$136
Aug 2006	$75
Aug 2007	$38
Aug 2008	$18
Aug 2009	$11

Google

Source: jcmit.com

What is Out of Memory?

Out of Memory (OOM) occurs when an application cannot allocate pages and no allowed memory may be reclaimed or compacted.

This may happen as the result of a complete depletion of system memory, memory controller limits, cpuset constraints, mempolicies, and/or fragmentation.

In a blockable context, the OOM Killer is the kernel's last resort to free memory and does so by killing the task that will most likely prevent subsequent page allocation failures.

Google

OOM Killer Rewrite

- Self-nominating of current when it has a fatal signal
- Child with highest badness heuristic score is sacrificed for parent if it does not share the same memory
- When a cpuset is OOM, the killed task's set of allowed nodes must intersect that of current
- For MPOL_BIND policies, the killed task must be allowed to allocate from current's nodes
- OOM killer is not called for DMA allocations
- Tasklist dump is enabled by default to show memory usage of each candidate task

Google

OOM Killer Rewrite

- All architectures share same page fault OOM behavior, now unified with the same semantics
- Entirely new badness heuristic used to determine which task to kill
- Introduce new /proc/pid/oom_score_adj interface to tune heuristic from userspace
- Deprecated old /proc/pid/oom_adj interface
- Currently in -mm tree, on track for 2.6.36

Google

Mempolicies

- MPOL_BIND policies bind VMAs to nodes
- Restricts memory allocations only to nodes in the mempolicy mask
- When nodes are full of unreclaimable memory, the OOM Killer is called to free memory
- In 2.6.35 and earlier, current is always killed since it is guaranteed to prevent subsequent failures
- With the OOM Killer rewrite, tasklist is iterated to find best task to kill
- Tasks that have MPOL_BIND or MPOL_INTERLEAVE policies that have disjoint nodemasks are immune from OOM kill

Google

Cpusets

- Bind applications to a set of cpus and a set of nodes
- Used by large NUMA machines to optimize for memory latency
- May be hierarchical, child cpusets must have a subset of cpus and nodes
- When a cpuset is OOM, killed task must be allowed to allocate on current's set of allowed nodes
- Doesn't help to kill a task if current still can't allocate memory
- Exceptions: GFP_ATOMIC, TIF_MEMDIE, irqs

Google

Memory Controller

- Enforces limits on the number of user pages a set of tasks may allocate
- May be hierarchical
- Reclaim is attempted prior to calling OOM Killer
- When a memory controller is OOM and OOM killing is enabled, a task **must** be killed to enforce the limit
- Killed task must be from same memory controller or child memory controller, if hierarchical

Google

OOM Killer

- Kills a memory-hogging task to recover a large amount of memory to prevent subsequent failures
- Avoids killing tasks that will not recover memory for current
- Waits for OOM killed task to fully exit before killing additional tasks
- Serialized by zones in the page allocator's zonelist to prevent parallel killing
- Unfortunately susceptible to mm->mmap_sem livelock

Google

/proc/sys/vm/oom_dump_tasks

```
[ pid ]   uid  tgid total_vm      rss cpu oom_adj name
[    1]     0     1     5920      492   1       0 init
[  542]     0   542     4258      237   1       0 upstart-udev-br
[  544]     0   544     4336      304   1     -17 udevd
[ 1144]   101  1144    31687      592   1       0 rsyslogd
[ 1160]   102  1160     6063      455   0       0 dbus-daemon
[ 1165]     0  1165    19148      873   0       0 gdm-binary
[ 1168]     0  1168    23807     1357   0       0 NetworkManager
[ 1172]   104  1172     8512      406   0       0 avahi-daemon
[ 1173]   104  1173     8481      144   0       0 avahi-daemon
[ 1175]     0  1175    14466      631   0       0 modem-manager
[ 1215]     0  1215     1519      161   1       0 getty
[ 1218]     0  1218     1519      162   1       0 getty
[ 1225]     0  1225     1519      161   0       0 getty
[ 1227]     0  1227     1519      161   0       0 getty
[ 1231]     0  1231    30116      917   1       0 console-kit-dae
[ 1233]     0  1233     1519      161   0       0 getty
[ 1236]     0  1236     1271      454   0       0 acpid
[ 1305]     0  1305     5268      254   1       0 cron
[ 1306]     0  1306     4720      115   0       0 atd
[ 1323]     0  1323    23374     1051   1       0 gdm-simple-slav
[ 1367]     0  1367    42158     7914   0       0 Xorg
```

- Filtered by tasks eligible to be killed depending on the context
- With OOM Killer rewrite, enabled by default

Google

New Heuristic

- “Badness” score ranges from 0 (never kill) to 1000 (always kill)
- Very large memory allocators (swapoff, ksm) are chosen automatically
- Heuristic baseline is now the task's resident set size (rss) and swap divided by the amount of allowed memory
- Root tasks are given 3% memory bonus, similar to LSMs
- Not used if /proc/sys/vm/oom_kill_allocating_task is enabled

Google

/proc/pid/oom_score_adj

- Powerful userspace influence to either prioritize or penalize a task for OOM kill
- Ranges from -1000 (OOM_DISABLE) to +1000
- May disable OOM killing completely for a task by writing OOM_DISABLE
- Adjusts the “badness” of a task by adding its value directly into the heuristic's score
- Deprecates /proc/pid/oom_adj (scheduled removal in August 2012)
- Currently backwards compatible with oom_adj users

Google