Linux performance monitoring

Linux performance basics

I want to write about Cassandra performance tuning, but first I need to cover some basics: how to use vmstat, iostat, and top to understand what part of your system is the bottleneck -- not just for Cassandra but for any system.

vmstat

You will typically run vmstat with "vmstat sampling-period", e.g., "vmstat 5." The output looks like this:

```
procs -----------memory---------- ---swap-- -----io---- -system-- ----cpu----
r  b   swpd   free   buff  cache   si   so    bi    bo   in   cs us sy id wa
20  0 195540  32772   6952 576752   0    0    11    12   38   43  1  0 99  0
22  2 195536  35988   6680 575132   6    0  2952    14  959 16375 72 21 4  3
```

The first line is your total system average since boot; typically this will not be very useful, since you are interested in what is causing problems NOW. Then you will get one line per sample period; most of the output is self explanatory. The reason to start with vmstat is the "swap" section: si and so are swap in (memory read from disk) and swap out (memory written to disk). Remember that a little swapping is normal, particularly during application startup: by default, Linux will swap infrequently used pages of application memory to disk to free up more room for disk caching, even if there is enough ram to accommodate all applications.

iostat

To get more details of io, use iostat -x. Again, you want to give it a sampling interval, and ignore the first set of output. iostat also gives you some cpu information but top does that better; let's focus on the Device section:

```
Device:         rrqm/s   wrqm/s     r/s     w/s   rsec/s   wsec/s avgrq-sz avgqu-sz   await
svctm  %util
sda               9.80     0.20   36.60    0.40  5326.40     4.80   144.09     0.06    1.62
```

```
1.41   5.20
```

There are 3 easy ways to tell if a disk is a probable bottleneck here, and none of them show up without the -x flag, so get in the habit of using that. "avgqu-sz" is the size of the io request queue; if it is large, there are lots of requests waiting in line. "await" is how long (in ms) the average request took to be satisfied (including time enqueued); recall that on non-SSDs, a single seek is between 5 and 10ms. Finally, "%util" is Linux's guess at how fully saturated the device is.

top

To learn more about per-process CPU and memory usage, use "top." I won't paste top output here because everyone is so familiar with it, but I will mention a few useful things to know:

- "P" and "M" toggle between sorting by cpu usage and sorting by memory usage
- "1" toggles breaking down the CPU summary by CPU core
- SHR (shared memory) is included in RES (resident memory)
- Amount of memory belonging to a process that has been swapped out is VIRT - RES
- a state (S column) of D means the process (or thread, see below) is waiting for disk or network i/o
- "steal" is how much CPU the hypervisor is giving to another VM in a virtual environment; as virtual provisioning becomes more common, avoiding noisy neighbors is increasingly important

"top -H" will split out individual threads into their own lines; both per-process and per-thread views are useful. The per-thread view is particularly useful when dealing with Java applications since you can easily correlate them with thread names from the JVM to see which threads are consuming your CPU. Briefly, you take the PID (thread ID) from top, convert it to hex -- e.g., "python -c 'print hex(12345)'" -- and match it with the corresponding thread ID from jstack.

Now you can troubleshoot with a process like: "Am I swapping? If so, what processes are using all the memory? If my application makes a lot of disk read requests, are my reads being cached or are they actually hitting the disk? If I am hitting the disk, is it saturated? How much 'hot data' can I have before I run out of cache room? Are any/all of my cpu cores maxed? Which threads are actually using the CPU? Which threads spend most of their time waiting for i/o?" Then if you go to ask for help tuning something, you can show that you've done your homework.

Posted by Jonathan Ellis

Revision #3
Created 2026-04-13 18:28:53 CEST by Philip
Updated 2026-04-13 19:28:41 CEST by Philip