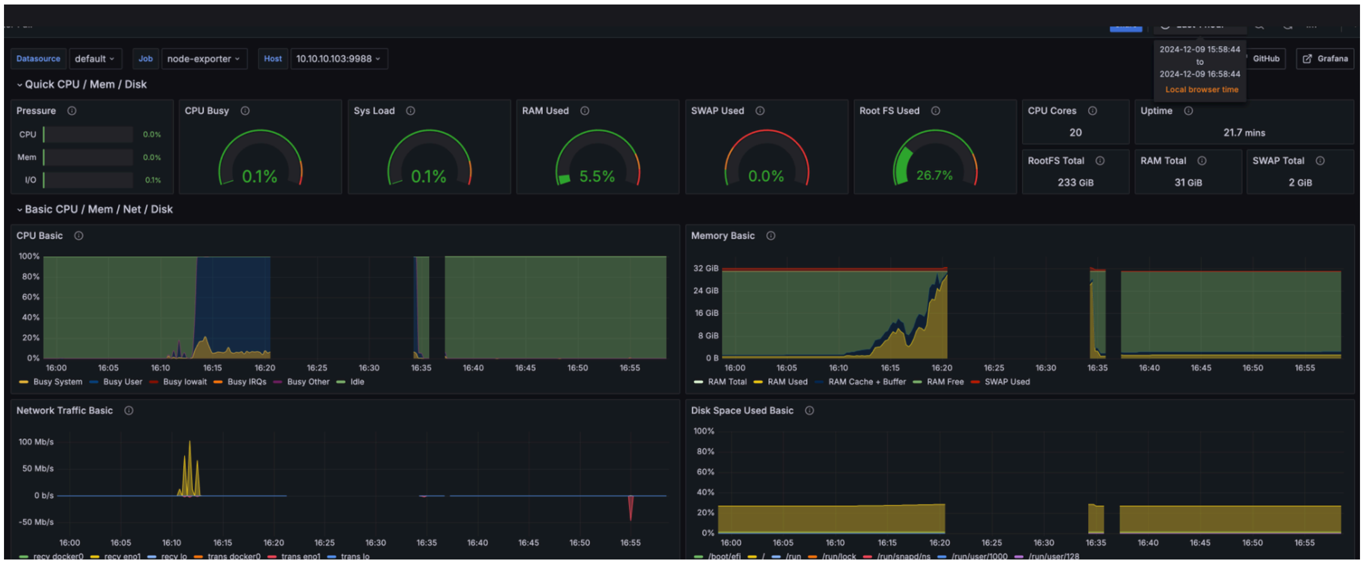
Modern Linux Chapter 8

Memory Overcommit Issue (Sysctl)

This is a case that why we have to toward building observability system

Cause of the issue



On fresh Ubuntu 22.04 installation, the CI runner crashes due to Out Of Memory killer

Current Build with Nix and Perform CodeQL Analysis in CodeQL workflow require lots of memory resource
They consume memory over 32GB in some steps. So the OS tries to use swap memory if there is
Usually under 42GB I guess (according to several records)

But sometimes workflow use memory as real over physical memory + swap memory, and then OOM is started

List of Investigation

- Compare requested CodeQL memory from workflow run history
- This value could be different on previous configuration and fresh installation
- Run CodeQL interactively, with RAM/Thread option from previous workflow run. to ensure that does not breaks system.
- Swap could have affected, compare config and try running after disabling swap

- Compare linux kernel tunables that is potentially related. Running diff on sysctl -a could reveal related config.
- Run memory stress test (i.e. running stress-ng with cpu, memory option on Actions workflow)
- Run CodeQL run multiple times
- Monitor memory usage with grafana dashboard

vm.overcommit_memory

It seemed to pass the CI workflow in some nodes which have vm.overcommit_memory in sysctl config as 1.
(Actually sometimes they failed too)
So investigated the principle of vm.overcommit_memory

Basically as we all know, Linux kernel has Virtual memory to manage memory efficiently. (over the size of physical memory)
vm.overcommit is config to set policy for how kernel manage memory allocation strictly.

- 0 (default) : Following heuristic memory overcommitment.
- 1 : Enabling full overcommitment.
  - It doesn’t care whether the OOM occurs or not. Just allocate memory address even if request of memory size is over physical + swap memory size
  - simply malloc doesn’t return error
- 2 : Don’t overcommit (based on vm.overcommit_ratio - default 50%)
  - CommitLimit = physical memory size * vm.overcommit_ratio/100 + swap memory size
  - malloc return error if process request over CommitLimit

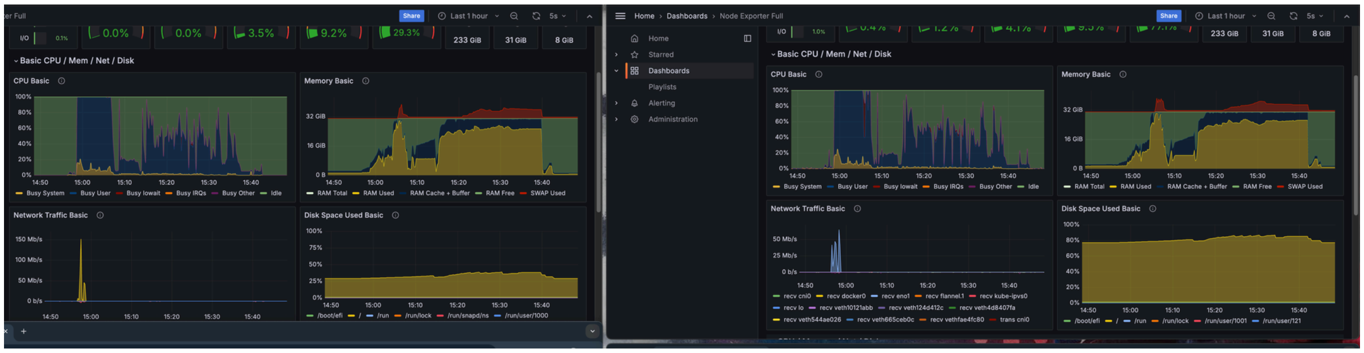
Why 10.10.10.104 (vm.overcommit_memory=1) passed CI sometimes?

It just allowed all the malloc requests from CI process. So sometimes it passed, but killed other processes to continue. (swap size was 2GB at first)
You can check the grafana records that prometheus could not pull the metric in short period.

Why AWS instance passed CI ?

Assume that aws instances are provisioned with server image(iso) by terraform to process CI workflows only within certain time.
By comparison, office servers are provisioned with desktop image and handle other tasks include ssh, office server monitoring, process for implementing GUI, etc…

Solution



Whenever changes occurs to CI workflow, Just check the maximum memory size (physical + swap) and configure the swap size to be enough

Reference

[docs - vm overcommit-accounting]

[code - vm_enough_memory]