
CERN

CPU thoughts for multiONE

IT-CM

NB. Relative scales

- The majority of traffic volume on LHCONE comes from managed 3rd party copies via FTS
- Remote access via worker nodes is a somewhat 2nd order effect
  - but still allowed and needed, of course

CPU properties

- CPU jobs run on a Batch system (HTCondor) which gives us:
  - Queuing, dispatch & monitoring of jobs
  - Agreed fair share among competing groups/experiments
  - Assigns useful work to under-utilised shares, allowing others to "burst" if spare capacity

- Much work over the years to prevent hard partitioning
  - …which leads to inefficiencies when users have varying workloads (e.g. CMS "cloud"), because others cannot make use of it
- Whatever we do, we would like to preserve these properties, notably we'll be running different jobs (from different experiments) on the same physical Linux box and the same VM

Machines are quantised

- …with respect to the size of the different jobs that are running on them
  - This imposes some efficiency limits on how dynamic we can be with recycling VMs
- 8-core dynamic slot on a VM can run 1 8-core job or 8 independent 1-core jobs or any tessellation
  - Recycling the machine loses efficiency (you drain leaving slots free or you kill jobs)
  - Similarly at the (32/40-core) physical machine level
- Per-job VM creation is bad: significant (re)creation cost (time w.r.t. job runtime and infrastructure load) -> 1/2 million VMs per day, tessellation logic in the wrong place

- Any solution needs to work without having to drain the things that are running the jobs

Possibles: box-centric view

- The packets from your job need to be different from the other's guys' jobs that are also running on the same box
  - ~~Different VMs~~
  - Different interfaces on same box
    - Different source IP [v4/v6] address, VXLAN
  - TOS bits in IPv4 (DSCP is 6-bit)
    - Application set or iptables mangle
  - Anything else we can do?

Application-set DSCP field

- Explicit setockopt() in application or similar
  - Network infra then routes according to TOS field
- Potentially doable assuming we control all the clients
  - xroot could, condor could be made to, HTTP libs are a struggle
- Do we really [want to] control all the (future) data access clients?
  - Somewhat limits the HTTP-based future imagined by some DOMArs in WLCG
- Force LD_PRELOAD on all jobs to rewrite the network functions? It's a bit exciting...

Namespaces

- Linux network namespaces can help
  - Independent routing tables
    - This means many network interfaces, one per “ONE” experiment on every box
      - Dynamic just-in-time creation (of network)? N.B. >500k jobs per day!
    - Tungsten managed VXLANs could help here
  - Independent iptables (TOS/DSCP with mangle)
    - Network infra then routes according to DSCP field

Namespacables status

- What can make a network namespace for the job it starts?
  - Docker can do it (hence Kubernetes presumably)
    - Don’t see us moving the scheduling of WLCG/ONE grid jobs to direct Kubernetes (away from HTCondor) any time soon, despite my occasional enthusiasm
  - HTCondor could be changed to do it
    - Either just network namespaces or possibly with Docker / Kubernetes’ help
    - Would probably require development
      - We have started talking to HTCondor devs about what it might involve
      - Somewhat pins the “solution” to running HTCondor

Questions

- DSCP (TOS) has 6-bits.. Are they all free and is 64 enough?
- If we have one interface per box, per ONE experiment, is that OK?
  - Does is help / make sense / possible to provision them at job start-time? Or is better to provision statically?
- The solution needs to work for T1 and T2 sites as well, right?

Or re-architect?

- Can we just access the remote data storage via a set of site-caches which proxy the connection
  - Then only the storage and proxies need be on the multiONE - worker nodes not
  - Being discussed in WLCG DOMA for latency hiding for non-data-lake sites – would it work at large sites too?
  - Same model (CDN) for HTTP access