SFU SIMON FRASER UNIVERSITY
THINKING OF THE WORLD

Optimizing Shared Resource Contention in HPC Clusters

Sergey Blagodurov sergey_blagodurov@sfu.ca
Alexandra Fedorova alexandra_fedorova@sfu.ca

The problem: contention for shared multicore resources (shared caches, memory controllers, NUMA domains, etc.) within cluster nodes incurs up to 40% severe degradation to job performance. HPC clusters are not contention-aware, with no virtualization to migrate jobs around.

The solution: Clavis-HPC, a novel contention-aware virtualized HPC framework. Here is how:

1) We monitor job processes on-the-fly and classify them with 2 parameters:
   a) a process is a *devil* if it is memory intensive, has high last-level cache missrate, otherwise - a *turtle*.
   b) if a given process is communicating with other processes.
2) We develop a multi-objective scheduling algorithm Clavis-Cluster that simultaneously:
   a) minimizes the number of devils on each node;
   b) maximizes the number of communicating processes on each node;
   c) minimizes the number of powered up nodes in the cluster.
3) After the new schedule is found, we enforce it by introducing a low-overhead live migration into cluster: the job scheduler places processes into OpenVZ containers, Clavis-Cluster migrates containers.

A typical HPC job management cycle with our modifications highlighted in red: (state-of-the-art ➡ Clavis-HPC)

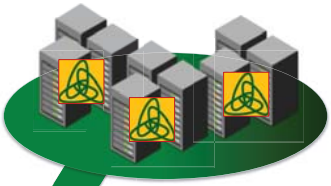
Clients (tablet, laptop, desktop, etc)

1). User connects to the HPC cluster via client and submits a job with a PBS script.

2). Resource Manager (RM) on the head node receives the submission request and passes it to the Job Scheduler (JS).
3). JS determines what jobs execute on what (nodes ➡ *OpenVZ* containers) and passes the scheduling decision back to RM.
4). RM starts/stops the jobs on the given (nodes ➡ *OpenVZ* containers).

Head node
RM, JS, Clavis-Cluster

Cluster network
(Ethernet, InfiniBand)

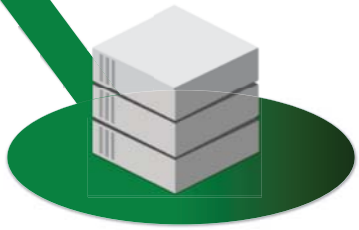


Compute nodes
Clavis contention monitor
OpenVZ containers
libraries (OpenMPI, etc)
RM daemons (pbs_mom)

5). The (baremetal ➡ virtualized) jobs execute on the (nodes ➡ containers) under the (default OS kernel scheduler ➡ contention aware user-level scheduler Clavis-DINO). They access cluster storage to get their input files and store the results.

6). RM generates a report about resource usage in the cluster during the last scheduling interval. The report contains process classes (devil or turtle, communicates or not).
7). Users can checkpoint their jobs with (BLCR or app-specific ➡ universal OpenVZ snapshots).
8). Sysadmins can (manually move jobs ➡ perform automatic on-the-fly job consolidation) across the nodes through (checkpoint/restart ➡ OpenVZ live migration).

Monitoring (JS GUI), control (IPMI, iLO3, etc)



Centralized cluster storage (NFS, Lustre)