Production System Evolution

Tadashi Maeno (BNL)

Outline

- Recent developments
- Future plans

Memory Usage Estimation

- The pilot gets various information about memory usage using a MemoryMonitor tool
  - max/min of PSS, RSS, VMEM, SWAP
- Tasks estimate memory usage based on the number of CPU cores and measured maxPSS
- Estimated pss = ramCount x coreCount + baseRamCount
  - coreCount : # of CPU cores
  - baseRamCount : Offset. Default value per task type
  - ramCount : is set for each task using the largest maxPSS of the first 5 finished jobs
- The brokerage checks the estimated pss with site.maxrss and site.minrss

Walltime Estimation

- Tasks use new scheme to estimate walltime by taking HS06 score at each site
- walltime = cpuTime x cpuEfficiency / coreCount / corePower x nEvents + baseWalltime
  - corePower : HS06 score
  - coreCount : # of CPU cores
  - cpuEfficiency : 90% by default. 0 if no scaling with nEvents
  - baseWalltime : Offset. 300 sec by default
  - cpuTime : is set for each task using the longest walltime of the first 5 finished jobs
- The brokerage uses the estimated walltime
- Estimated walltime is available in job. maxwalltime

Changes to Log files

- Special task parameter is set by default to leave log files where they are produced
  - No transfers for log files
  - One log dataset for each task
  - Files are added to the dataset but they are distributed among many sites
- Log files are going to be uploaded to ObjectStores in addition to traditional storages
  - Only ObjectStores in the future?
- No log files merging in merge step

Staging from TAPE

- When jobs are submitted to read files from TAPE, Panda makes small datasets to copy those files from TAPE to associated DISK endpoints, instead of sending pre-staging requests
  - DATADISK for production jobs
  - SCRATCH/USERDISK for analysis jobs
- Rucio can throttle tape access using own algorithm rather than relying on read buffers of tape systems
- Jobs go to assigned state and changes to activated once all files are copied to DISK

Post Actions on Job Failures

- Task and/or job requirements are automatically changed when jobs are failed due to particular errors
  - failed with memory shortage → increase memory requirement
- Not to keep jobs failing with the same error
- Several actions
  - Disable further reattempts
  - Limit the number of attempts
  - Increase RAM requirement
  - Increase CPU requirement
- Combination of error and action defined in DB

Job Cloning

- To generate identical jobs at multiple sites
- Two modes
  - Run once
    - Once the pilot gets a job from the panda server at a site, waiting jobs at other sites are killed
  - Store once
    - Once the pilot registers output files for a job, waiting/running jobs at other sites are killed
- Pairing pledged+opportunistic sites with job cloning
  - No pilots are sent to a site if there are no jobs at the site. No jobs are sent to a site if there are no pilots at the site
    - Sending jobs to no-pilot sites → many timeout jobs if those sites are really inactive
    - Too late for some resources if jobs are assigned after they become active
  - Jobs could be cloned at inactive sites if the jobs are assigned to a pledged site. Only the first job is used

Unified Task Parameters for Event Service

- ES jobs require some special jobOptions
- If tasks are configured only for ES, jobs can run only on sites where ES is enabled with jobSeed=any or es in site.catchall
  - Cannot easily switch ES to non-ES or vise-versa once tasks are submitted
- Not so many sites enable ES → resources for ES tasks are limited → negative incentive for migration to ES
- There is a new way to configure tasks to generate both ES and non-ES (normal) jobs
  - A single task can generate ES jobs at ES-enabled sites and non-ES jobs at ES-disabled sites

Closed Status

- New job status "closed" has been introduced to distinguish manually-killed jobs and system-killed jobs
  - For accounting
  - Too many cancelled jobs in Event Service tasks
- Generally closed jobs can be ignored in terms of site operation
  - E.g., jobs go to closed state once they are killed to be reassigned to another site
    - They are not very interesting since they didn't use CPU on WN
- For Event Service, jobs go to closed when they are terminated to generate subsequent jobs, and changes to finished/failed once the final merge job is done

WORLD Cloud

- Single partitioning to contain all sites
- Multiple nuclei (T1s + big T2s defined in AGIS)
- Each task is assigned to a nucleus, jobs run at the nucleus and multiple satellite sites, and output files are aggregated to the nucleus
  - Association between nucleus and satellites is dynamically defined based on network metrics
- Migration to WORLD cloud is on-going
  - 10% of requests are automatically fed into WORLD
  - Full migration in a few weeks if there is no problem
- Old geographical clouds will not be used in terms of workload management
  - Regional cloud supports should remain for operation

Brokerage for WORLD 1/2

- Task brokerage
  - Data locality
  - Total assigned workload (aka RW)
  - Destination endpoint matching
    - e.g., Tasks for TAPE → Nuclei with TAPE endpoints
  - Endpoint blacklist and free space
  - Ability for job execution
  - To be improved to consider
    - CPU capacity of each nucleus
      - e.g., max or avg # of running jobs in last N hours
      - Large tasks → Large nuclei
    - Network
      - e.g., avoid nuclei if many transfers are queue in the FTS channel to/from the nuclei

Brokerage for WORLD 2/2

- Job brokerage
  - Same as old clouds
    - Nucleus weight instead of T1 weight
    - High pro jobs go to Nuclei instead of T1s
  - Network is not well considered
    - Indirect usage : sites are avoided if there are many transferring jobs
    - Will use network info provided by Configurator
      - e.g. larger weight if the site has good connection to/from the nucleus

Alternative Stage-out

- The pilot tries to upload output files to local storage first. If fails, uploads them to alternative locations
  - To T1 for old clouds
  - To nucleus for WORLD
- Files uploaded to T1/Nucleus are added directly to final datasets and jobs go to finished instead of transferring if no file transfers are required
- If the job has multiple outputs and outage of the local storage happens in the middle of stage-out, only some of them can be uploaded to T1/nucleus
  - The pilot reports which files were alternatively staged-out
  - PandaMon will show them

Future Plans

Network-aware Brokerage

- Network information is important for WORLD
  - High priority tasks could choose sites with good connections to nucleus
  - Low priority tasks could give larger weights to sites with good connections
- Network Configurator
  - First version has been deployed
  - Collects network information from Network Weather Service
    - Static closeness
    - Queue depth
    - Throughput
- New algorithm based on network to be implemented in the brokerage

Global Share via Distributor

- Resource shares (SCORE,MCORE,HIMEM) X Activity shares (MC,group,repro,HLT,T0,...)
  - e.g., 10 % of ATLAS CPU to group HIMEM jobs
- Shares can change dynamically based on physics needs
  - e.g. More CPU to some activity just before a conference
- Requirements to be collected
  - Sites want to change resource partitioning by themselves? If so, all or some?
  - Shares should be defined at site-level or globally?
    - Each site provides 10% of CPU, or
    - Some sites provide CPU so that sum of them corresponds to 10% of ATLAS CPU
  - OK if a site occasionally runs only one type of jobs?

Preemption for Event Service

- To run low priority ES jobs in backfill mode
  - Release CPU as soon as high priority jobs come into the queue
- Possibly a new cron (to be integrated to Distributor in the future)
  - Running ES jobs would be killed based on priority of ES jobs and waiting time of high priority jobs
  - The kill command would be propagated to the pilot via regular heartbeats in addition to responses of requests to update/get event ranges
  - High priority jobs would be dispatched to the next pilot
  - ES jobs would be automatically re-generated to resume remaining work and will wait until all high priority jobs are dispatched

Misc for Opportunistic Resources

- New Workqueue
  - Tasks for opportunistic resources have lower priorities
    - Could interfere with normal tasks in the same workqueue (mcore_evgensimul,evgensimul)
      - e.g. high prio tasks could block opp tasks
  - Better to have a separate workqueue
    - More workqueues for WORLD anyway
- Brokerage to be aware of opportunistic resources
  - Three site policies: opp-only, no-opp, mixture
  - A separate attribute in catchall (like ES jobSeed=es, std,any) or new filed
    - To be propagated from AGIS
  - The brokerage could avoid opp sites for urgent jobs
    - E.g. only low priority jobs go to opp sites?

Job Filler for Small Sites

- JEDI gathers tasks from a task pool based on their priorities in descending order, and stops to generate jobs once enough jobs are queued (nQueued/nRuning>2)
  - High priority tasks first
- Priority ~= Urgency
  - Jobs from high priority tasks are sent to large sites, in order to quickly get results
- Small sites tend to be lack of jobs
- Possible solution (TBD)
  - To introduce a new workqueue for low priority tasks
    - E.g., (evgen or simul) and priority < XYZ and ...
    - JEDI gathers tasks per workqueue → Low prio jobs can go even if high prio jobs from other workqueues are already there
    - Jobs could go to large sites as well but they would be eventually reassigned

Event Duplication Protection

- Event duplication caused by various reasons
  - SW bugs
  - Lack of unified scheme to specify task input
  - Highly parallelized system architecture
- Adding multilevel protection
  - DEFT for request-level check
  - JEDI for dataset/job-level checks
  - EventIndex for asynchronous validation
- Possible container-level protection
  - Each job would report the first and last event numbers in jobReport.json and JEDI would check if there is overlap of the event range in tasks which contribute to the same output container

Conclusions

- Production system has been actively evolved while steadily running for Run2
- Still many new developments coming in near future