Event service support in PanDA/JEDI

Tadashi Maeno (BNL)

JEDI and Event service

- Event service uses JEDI database tables
  - The file table for metadata, such as GUID, file name, and # of events
  - The event table for event level bookkeeping
    - Very simple structure optimized for fast update and select
- JEDI generates jobs to use event service if the task is configured accordingly with some task parameters
  - Input file(s) is internally split to multiple event ranges
    - N events in each event range (N≥1)
  - Job is dispatched to the pilot and event ranges are dispatched to AthenaMP processes via the pilot

Initial Implementation

- One or more ES jobs consume all event ranges sequentially
  1. One pilot takes a job and consumes event ranges which are associated to input(s) of the job until the execution time reaches the limit
  2. New job is generated when the first one is terminated
  3. Next pilot gets the job and consumes remaining event ranges
  4. Once all event ranges are processed, a merge job is generated
- Scalability issue
  - Only one node is used at a time
    - # of cores is limited

New Implementation with Multiple Consumers 1/3

- Multiple ES jobs consume event-ranges in parallel
  1. Multiple jobs are created beforehand and waiting in the pool. All jobs are associated to the same input(s)
  2. One pilot takes a job and consumes event ranges as much as possible
  3. Another pilot pops up to get another job when a node becomes available
     - The job can get started while the first job is still running
     - The number of event ranges consumed by this job could be different from that of the first job
  4. Once all event ranges are processed, un-used jobs are killed and a merge job is generated

New Implementation with Multiple Consumers 2/3

- Well fit with existing Panda mode = minimal changes
  - Each consumer is a separate job
    - The pilot itself doesn't need to know each other
    - Existing job-view of pandmon
  - All jobs contribute to the same output but each of them produces independent pre-merged files
    - One per-merged file per event rage
  - One log file per job
    - No need to produce one log file per event range
  - Retry or merge job creation via existing built-in mechanism in Panda/JEDI

New Implementation with Multiple Consumers 3/3

- Useful when many nodes are available but each of them has very short or unpredictable execution time
  - Amazon spot market
    - Clouds are good for CPU-intensive jobs (longer execution time)
    - Usage under 1h is free
  - Voluntary resources
    - Nodes could suddenly disappear
  - HPC
    - Back-fill
    - Multiple consumers or Single consumer + pilot level splitting (TBD) ?

Current Status and Plans

- All ES-related functions are available on production panda server and JEDI nodes
  - Stand-alone tests done
  - Tests with the real pilot to be done
- Production with Amazon spot market, HPC, voluntary resources, ...