MapReduce Support in Slurm: Releasing the Elephant

Ralph H. Castain, Wangda Tan, Jimmy Cao, and Michael Lv
Greenplum/EMC
What is MR+?

Port of Hadoop’s MR classes to the general computing environment

- Allow execution of MapReduce programs on any cluster, under any resource manager, without modification
- Utilize common HPC capabilities
  - MPI-based libraries
  - Fault recovery, messaging
- Co-exist with other uses
  - No dedicated Hadoop cluster required
Why would someone use it?

• Flexibility
  - Let customer select their preferred environment (e.g., SLURM)
  - Share resources

• Scalability
  - Launch scaling: Hadoop (~N), MR+ (~logN)
  - Wireup: Hadoop (~N^2), MR+ (~logN)

• Performance
  - Launches ~1000x faster, runs faster
  - Enables interactive use-case

• MPI library access
  - ScaLAPACK, CompLearn, PetSc, ...
How does it work?

• “Overlay” JobClient class
  - JNI-based integration to Open MPI’s run-time (ORTE)
  - ORTE provides virtualized shim on top of native resource manager
    • Launch, monitoring, and wireup at logN scaling
    • Inherent MPI support, but can run non-MPI apps
    • “Staged” execution to replicate MR behavior
  - Preposition files using logN-scaled system

• Extend FileSystem class
  - Remote access to intermediate files
  - Open, close, read, write access
  - Pre-wired TCP-based interconnect, other interconnects (e.g., Infiniband, UDP) automatically utilized to maximize performance
What about faults?

• Processes automatically restarted
  – Time from failure to relocation and restart
    • Hadoop: ~5-10 seconds
    • MR+: ~5 milliseconds
  – Tunable number of local restarts, relocations
  – Sensors provided to monitor resource usage, progress

• Future state recovery based on HPC methods
  – Process periodically saves “bookmark”
  – Restart provided with bookmark so it knows where to start processing
  – Prior intermediate results are preserved, appended to new results during communication
Resource management

- Currently need to get allocation prior to execution
  - Mismatch with typical MapReduce procedure

- MR procedure
  - Program identifies files needing to be accessed and queries file system for locations
    - Files may be shared across multiple locations
  - File system returns list of nodes housing files and/or shards
  - Execution optimized if it can occur on those nodes, but can proceed if placed anywhere
    - Preference: nodes on same switch (typically same rack)
SLURM extension (via socket)

• Resource queries
  - Number of nodes and slots in system
  - Number currently available (non-guaranteed)

• Resource allocations
  - Provided a list of desired nodes, allocate a specified number to this job
  - Flag indicates whether nodes are mandatory, or optional
    • Mandatory: allocate all specified nodes
    • Optional: allocate all immediately available desired nodes, fill rest with anything
  - Async callback
    • Rolling allocation (as nodes available), or block allocation
Benefits

• Allows SLURM to manage MapReduce jobs
  – Avoids cross-licensing issues via use of defined socket-based message exchange
  – Retains transparency to RM in MapReduce
• Allows MapReduce to operate on SLURM clusters without modifying code or user behavior
  – Execution transparently ports to SLURM environment
• Avoids requiring SLURM integration to file system
  – File location queries maintained at the MapReduce level
Status

• Proof of concept
  - Running MR on example problems
  - Benchmark results to be announced in December
  - Initial SLURM integration code prototype complete

• Near-term plans
  - Demonstrate on-the-fly fault recovery
  - Performance comparisons at large scale
  - Contribute SLURM-MR+ integration to SLURM
  - Release MR+ (open source, probably Apache)