HPC-Reuse: efficient process creation for running MPI and Hadoop MapReduce on supercomputers

Thanh-Chung Dao and Shigeru Chiba
The University of Tokyo, Japan
Running Hadoop and Spark on supercomputers

- Good choices to run MapReduce and Machine Learning algorithms

- Mature frameworks and providing standard APIs
- Easy to write applications
Challenge: running Hadoop/Spark on PBS

- PBS: Portable Batch System
  - The de facto resource manager of supercomputers
  - Created to support MPI style-programming and run coarse-grained HPC applications

- So, dynamic process creation is not the first citizen

Dynamic process creation: adding a new process to the running job at any time.
Restriction of process creation on PBS

- Hadoop and Spark require dynamic process creation
  - Minimizing the cost of changes in architecture
- Gang scheduling (of processes) more favorable on PBS
  - All-or-nothing scheduling strategy
    - Statically creating all processes at the beginning
    - Since resizing running jobs might affect performance and fairness
  - Dynamically adding a new process is optional, but not recommended
    - MPI-Spawn is slow (not allowed on some supercomputers, e.g. FX10)
    - Process fork causes MPI connection loss
Virtualize dynamic process creation

Create a pool of processes at the beginning of a job (for PBS) and dynamically allocate them to Hadoop/Spark
Implementation: reuse JVM processes

Hadoop YARN (NodeManager)

Container request

Pool Manager

Round-robin scheduling

JVM process pool

Process Pool on each node (Create JVM processes and make a pool)

Allocation
- Set busy
- Export env. variables
- Create new class loader
- Invoke main() method

De-allocation
- Clean static fields
- Unset busy

Process
Request sending
State changing
Technical issues

• Class loading
  • How to load user’s classes
    • Use new class loader to load them
    • Not reload Hadoop or Spark’s classes
    • Reduce class loading time and exploit compilation technology in JVM

• Clean-up
  • Security problem due to reusing static fields
    • E.g. loginUser static field is kept unchanged whenever its value is not null
  • Reset all static fields
    • Current implementation, reset only static fields containing user information and job configuration
Evaluation of HPC-Reuse

• **Test case**
  - Fork-based YARN
    • The original
  - MPI-Spawn YARN
    • Process fork mechanism is replaced with MPI-Spawn
  - HPC-Reuse YARN
    • Our proposal

• **Cluster**
  - 33 TSUBAME nodes
  - 33 FUJITSU FX10 nodes
  - One master and 32 slaves
  - Hadoop v2.2.0
  - OpenJDK 7 and OpenMPI 1.6.5
Evaluation of HPC-Reuse

• Purpose
  - Show HPC-Reuse is as good as fork-based approach in general
  - HPC-Reuse shortens start-up time in iterative workloads

- Tera-sort on TSUBAME (33 nodes)
- Iterative PageRank on FX10 (33 nodes)

Graphs showing:
- Up to 6% faster than original, 2.5x than MPI-Spawn
- Up to 26% faster than original
Summary

• Running Hadoop/Spark on PBS of supercomputers
  • Hadoop and Spark require “dynamic process creation”
  • But it is not the first citizen
    • Gang scheduling is more favorable

• Our proposal: HPC-Reuse
  • Virtualization layer
  • Using process pool
  • Up to 26% improvement in iterative PageRank

• Future work
  • MPI-based data shuffle on HPC-Reuse
  • In-memory Hadoop MapReduce
Thank you!
Email: chung@csg.ci.i.u-tokyo.ac.jp