Apache Hadoop YARN: Yet Another Resource Negotiators

Vinod Kumar Vavilapalli et al.
Hortonworks, Yahoo, Microsoft, Inmobi and Facebook

SoCC’13 Best Paper
Presenter: Hongwei Wang

Some slides are borrowed from Hortonworks and Apache Hadoop
Agenda

- Why YARN?
- YARN Architecture
- Experiments
- Conclusion
Hadoop 1.0: Batch

HADOOP 1.0
Built for Web-Scale Batch Apps

Tight coupling of MapReduce model with the resource management infrastructure

All other usage patterns must leverage the same architecture
Hadoop MapReduce Classic

- **JobTracker**
  - Manage cluster resources
  - Job/task scheduling

- **TaskTracker**
  - Per-node agent
  - Manage tasks
MapReduce Classic: Limitations

- **Scalability**
  - Maximum cluster size: 4,000 nodes
  - Maximum concurrent tasks: 40,000

- **Overloaded JobTracker, single point of failure**

- **Hard partition of resources into map and reduce slots**
  - Low resource utilization

- **Lack support for alternative paradigms and services**
  - Iterative applications implemented using MapReduce are 10x slower
Hadoop 2: Next-Gen Platform

**Single Use System**
Batch Apps

**HADOOP 1.0**
- MapReduce
  - (cluster resource management & data processing)
- HDFS
  - (redundant, reliable storage)

**Multi Purpose Platform**
Batch, Interactive, Online, Streaming, …

**HADOOP 2.0**
- MapReduce
  - (data processing)
- YARN
  - (cluster resource management)
- Others
  - (data processing)
- HDFS2
  - (redundant, reliable storage)
Hadoop YARN

Store ALL DATA in one place…

Interact with that data in MULTIPLE WAYS

with Predictable Performance and Quality of Service

Applications Run Natively IN Hadoop

- BATCH (MapReduce)
- INTERACTIVE (Tez)
- ONLINE (HBase)
- STREAMING (Storm, S4,...)
- GRAPH (Giraph)
- IN-MEMORY (Spark)
- HPC MPI (OpenMPI)
- OTHER (Search) (Weave...)

YARN (Cluster Resource Management)

HDFS2 (Redundant, Reliable Storage)
Key Improvements in YARN

• **Framework support multiple applications**
  - Decouple generic resource management from programming framework
  - Share same Hadoop cluster across applications

• **Improve cluster utilization**
  - Generic resource container replaces based fixed map/reduce slots (2 CPU, 2 GB Memory)

• **Scalability**
  - Remove complex application logic from RM to scale further
YARN Concepts

• JobTracker is decoupled into
  - Resource Manager (RM): global resource scheduler
  - Application Master (AM): manage per-application scheduling and task execution

• TaskTracker is changed into
  - Node Manager (NM): per-node agent, manage the life-cycle of container and monitor container resources
YARN Architecture and Workflow

1) Client -> Resource Manager
   Submit App Master
2) Resource Manager -> Node Manager
   Start App Master
3) Application Master -> Resource Manager
   Request containers
4) Resource Manager -> Application Master
   response allocated containers
5) Application Master -> Node Manager
   Assign resources to tasks (assignment)
   Start tasks in containers (start Container -> stop container)
6) Node Manager -> Resource Manager
   report running and terminated container,
   trigger new round of scheduling.
Fault Tolerance

• **RM Failure**
  - Single point of failure
  - Recovery from persistent state, kill and restart all AMs

• **AM Failure**
  - AM sends periodic heartbeat to RM
  - RM will restart AM and re-run tasks

• **NM Failure**
  - NM sends periodic heartbeat to RM
  - RM marks containers as failure and report to AMs
  - AM is responsible for reacting to node failures, re-run tasks.
Experiments

(a) Daily jobs

(b) Daily tasks

Figure 2: YARN vs Hadoop 1.0 running on a 2500 nodes production grid at Yahoo!
Conclusion

• **YARN decouples resource management and programming framework to provide**
  
  - Greater scalability
  
  - Higher utilization
  
  - Enable a large number of different frameworks to efficiently share a cluster

• **Cons:**
  
  - RM single point of failure, waste resources and time by restarting all AMs.
  
  - NM/AM: simple re-run failed/killed tasks leads to wastes
  
  - Log aggregation increases the pressure of HDFS NameNode, making it as a bottleneck