Apache Hadoop YARN: Yet Another Resource Manager

V. Vavilapalli, A. Murthy, C. Douglas, S. Agarwal, M. Konar, R. Evans, T. Graves, J. Lowe, H. Shah, S. Seth, B. Saha, C. Curino, O. O'Malley, S. Radia, B. Reed, E. Baldeschwieler

Presented by Erik Krogen

Motivation: Issues in Hadoop 1

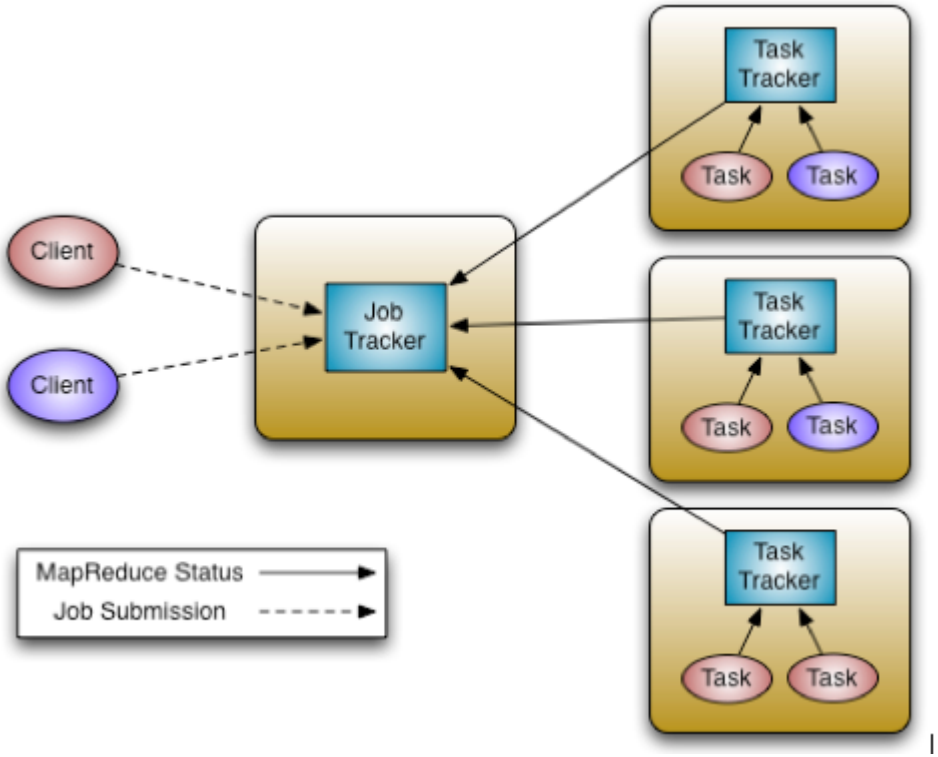
People were (ab)using MapReduce in unexpected ways, e.g. using it as a job scheduler

Single JobTracker node for the entire cluster - scalability issues

Statically allocated map and reduce task slots cause utilization issues

Fix these issues while maintaining backwards compatibility

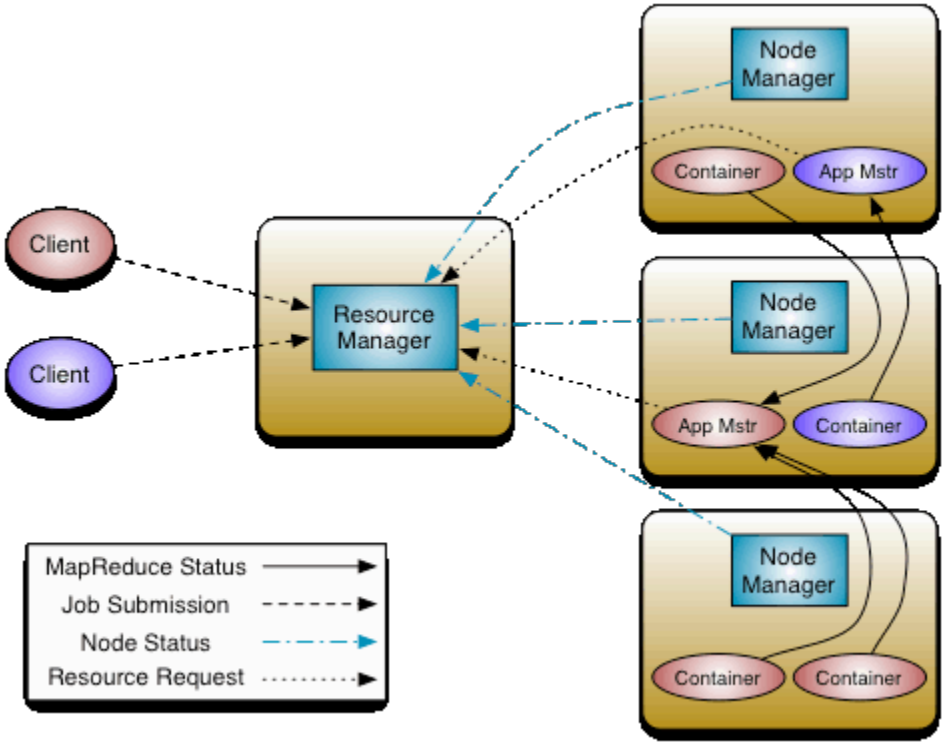
Architecture: Hadoop 1



Architecture: Concepts

- Application
  - Job submitted to the framework
  - e.g. a MapReduce job, Spark job, Spark Streaming job
- Container
  - Basic unit of allocation, variable-size resources
  - Analogous to tasks

Architecture: Hadoop 2 (YARN)



Architecture: ResourceManager

- Dedicated node, only one per cluster - single point of failure
- Centralized scheduler, but has fewer tasks than JobTracker
  - Pluggable scheduling policies
- Tracks resource usage, node liveness
- Allocates resources to applications
- Also requests resources back from applications

Architecture: ApplicationMaster

One per application - application-type specific

Requests resources from RM - can specify number of containers, resources per container, locality preferences, priority

Can subsequently update requests with new requirements

Manages all scheduling/execution, fault tolerance, etc.

Runs on a worker node - must handle its own failures

Architecture: NodeManager

Daemon on each node, communicates status to RM

Monitor resources, report faults, manage containers

Physical hardware checks on node

Provide services to container, e.g. log aggregation

Architecture: NodeManager

Send Container Launch Context (CLC) to NM to start container

Dependencies, env vars, tokens, payloads, command, etc.

Containers can share dependencies (e.g. exes, data files)

Can configure auxiliary services into NM, e.g. shuffle between map and reduce is an auxiliary shuffle service

Fault Tolerance

- Left completely up to ApplicationMaster - RM doesn't help
- AM must have fault tolerance for its containers as well as itself
- Some higher-level frameworks on top of YARN exist to make this and other things easier, e.g. Apache REEF (Retainable Evaluator Execution Framework)

Performance

Performance generally on-par with Hadoop 1, sometimes slightly worse (overhead of containers, RM to AM communication, etc)

Much better utilization

large part due to removal of static map and reduce slots

Yahoo: “upgrading to YARN was equivalent to adding 1000 machines [to this 2500 machine cluster]”

Why YARN?

- It's in Hadoop 2 => adoption is automatically widespread
- Per-job scheduler means big scalability
- Per-job scheduler means multiple versions of e.g. MR are possible
- Less general than Mesos (e.g. request vs offer model)
  - Can be good and bad