Apache Hadoop YARN: Yet Another Resource Negotiator

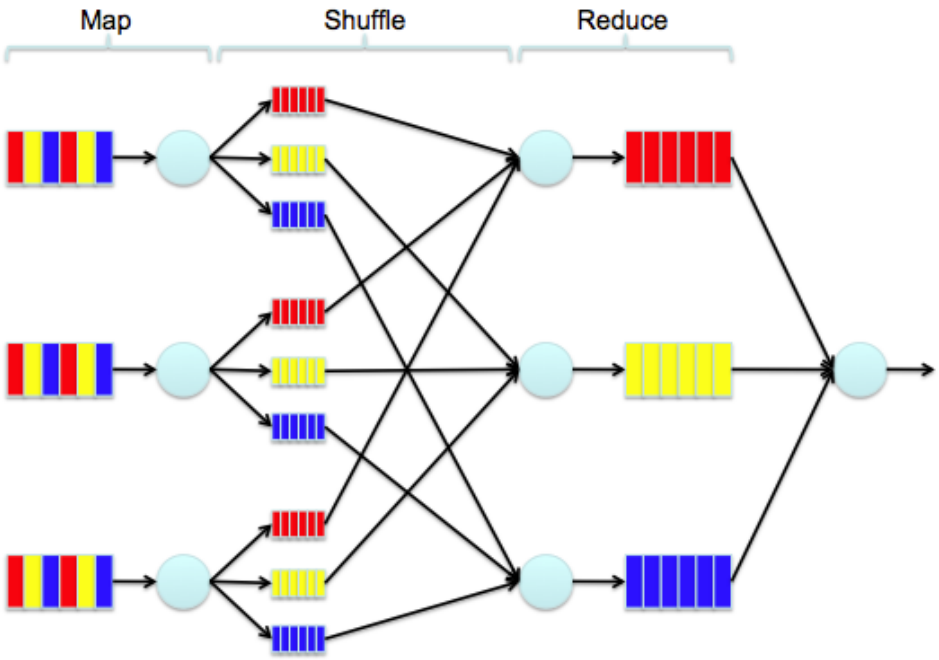
BDCS'14F
Yunseong Lee
2014/09/03

Contents

- History
- Architecture
- YARN in the real-world
- Experiments
- Related works
- Conclusion

History

- MapReduce
  - OSDI'04 from Google
  - Parallel, distributed Programming model

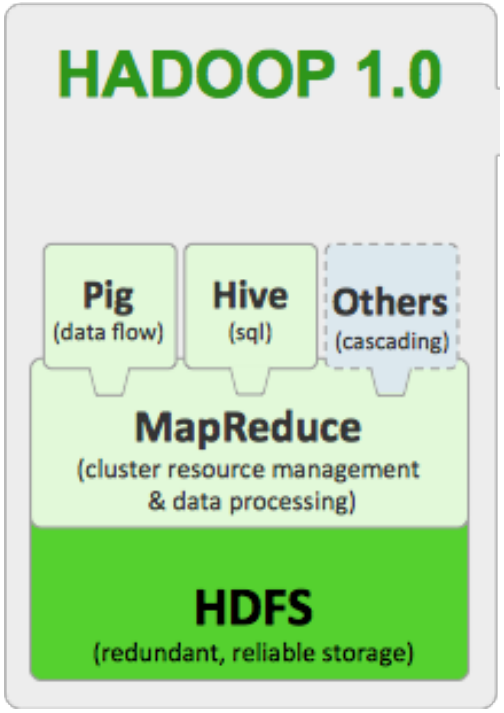


https://mm-tom.s3.amazonaws.com/blog/MapReduce.png

- Map() : filter <K, V>
- Shuffle()
- Reduce() : aggregate

History

- Hadoop
  - Yahoo!'s implementation of MapReduce

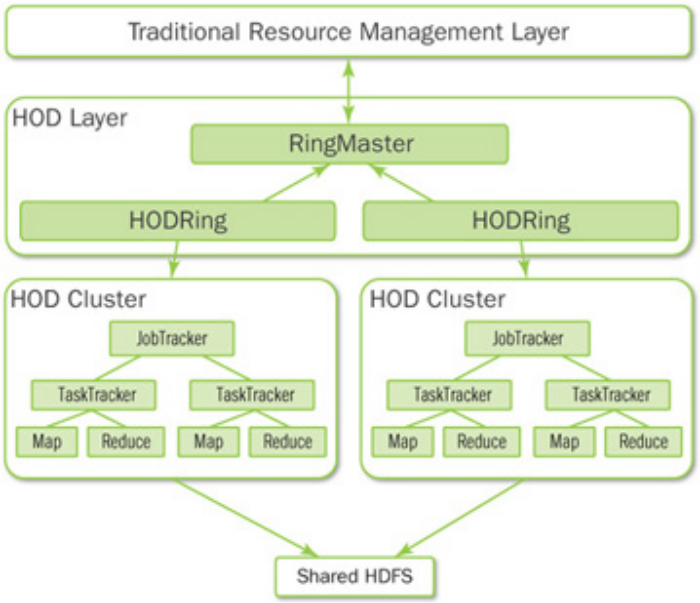


- MapReduce : Programming model
- HDFS : Distributed File System
- Applications : based on M/R

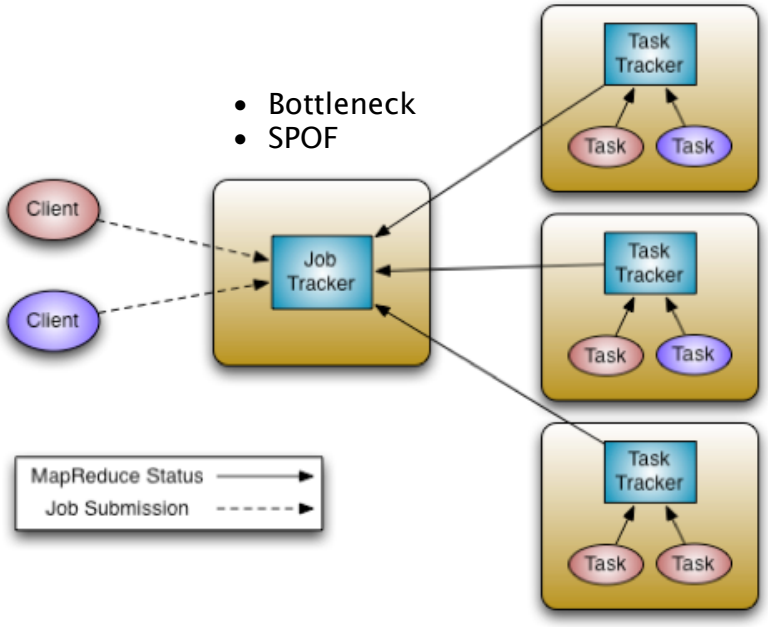
http://hortonworks.com/blog/apache-hadoop-2-is-ga/

History

- Hadoop On Demand(HOD)
  - Provisioning virtual Hadoop clusters



- Hadoop v1
  - Bottleneck
  - SPOF



- MapReduce ONLY

[Apache Hadoop YARN: Moving beyond MapReduce and Batch Processing with Apache Hadoop 2]

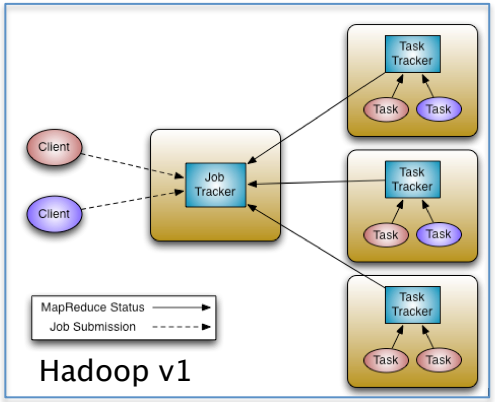
Requirements for YARN

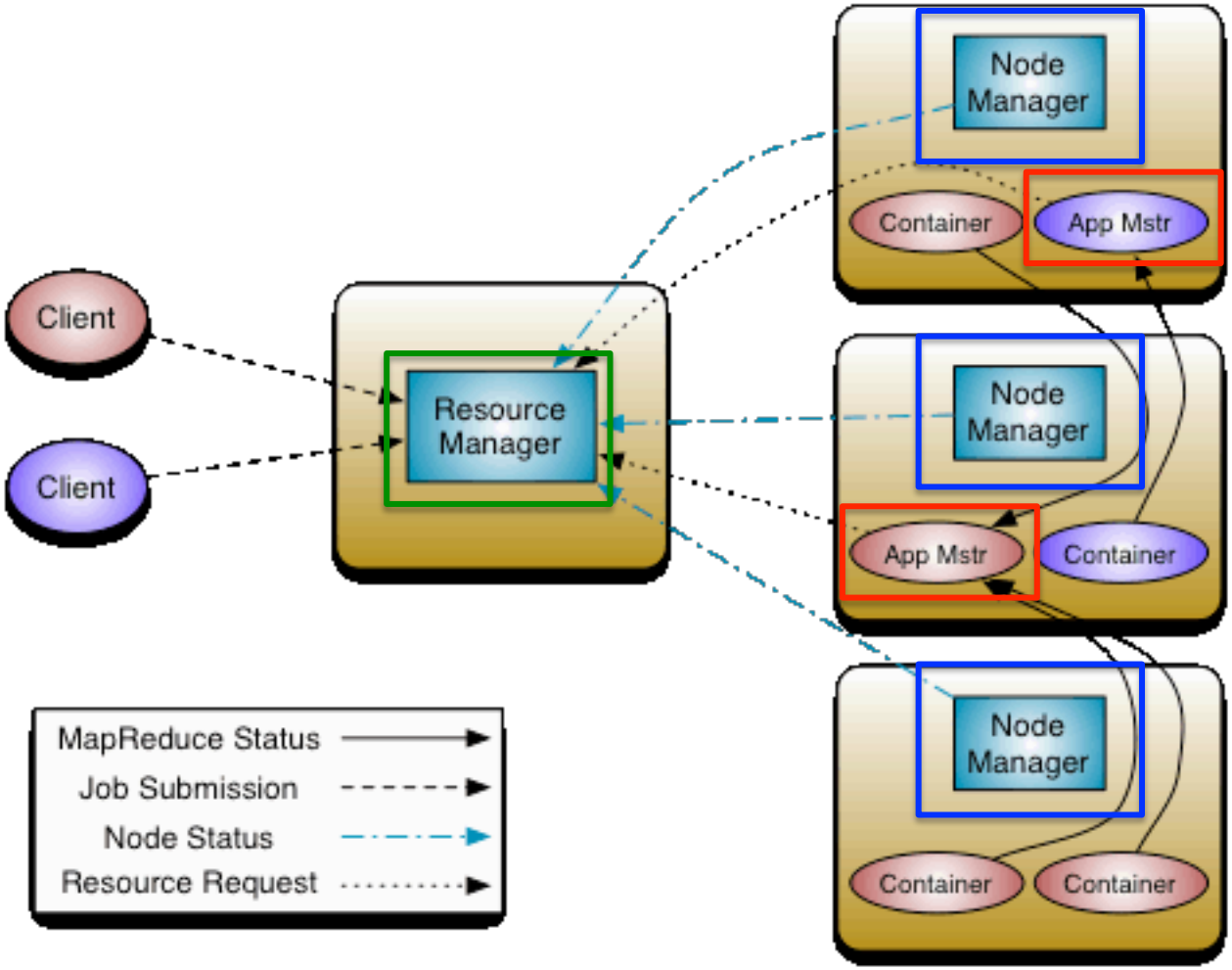
- (R1)Scalability
  - Serve very Large-scale cluster
- (R2)Multi-tenancy
  - Search / Ad analytics / Spam filtering, etc
- (R3)Serviceability
  - Decouple upgrade dependencies
- (R4)Locality Awareness
  - Latency from remote transportation
- (R5)High Cluster Utilization
  - Cluster allocation latency

Requirements for YARN

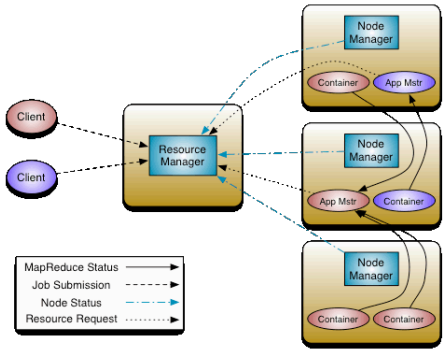
- (R6)Reliability/Availability
  - Avoid SPOF, onitor workload continuously
- (R7)Secure and auditable operation
  - Authentication
- (R8)Support for Programming Model Diversity
  - Extensible to multiple usage
- (R9)Flexible Resource Model
  - Typed slots are not proper even in M/R
- (R10)Backward compatibility
  - Compatible to existing ecosystem

Architecture Overview



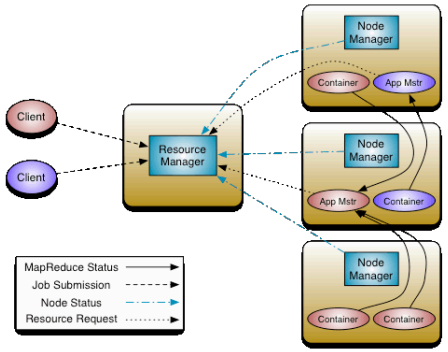


Resource Manager(RM)



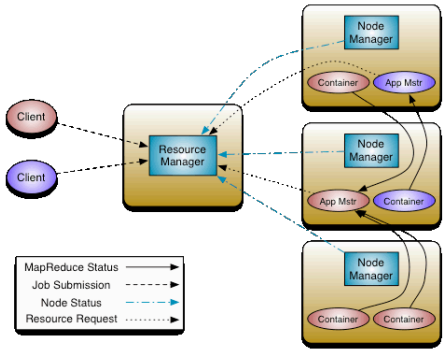
- Expose APIs
  - Application submission
  - Resource access negotiation / management
  - Cluster monitoring
- Compact and efficient Communication
- Can specify Resource request
  - Locality preferences(R4)
  - Clear description for Applications needs(R1, R5, R9)
- Not responsible for
  - Coordinating application execution
  - Task fault-tolerance

Application Master(AM)



- Run as a Container
- Communicate with RM via Heartbeat
- Coordinate the application's execution
  - Resource Request w/ preferences and constraints
  - Receive a container lease (*Late binding*)
  - Fungible and framework-specific semantics (R3, R8, R10)
- Progress/Status tracking
- Fault tolerance for Application

Node Manager(NM)



- Manage containers
  - Authenticates for Container lease
  - Containers' dependencies
  - Monitors Container's execution
  - Kill containers
    - directed by RM/AM
    - when Container exceed the capacity
- Communicates with RM
  - report its status
  - receives instructions
- Monitor health of physical node
- Local services
  - *log aggregation* service
- *Auxiliary services*
  - e.g. MapReduce shuffle

Yarn framework/application writers

- Author should follow the protocol described in 3.5
- A framework to ease development to serve Client libraries (TBD)
  - YarnClient
  - NMClient
  - AMRMClient

Fault tolerance and availability

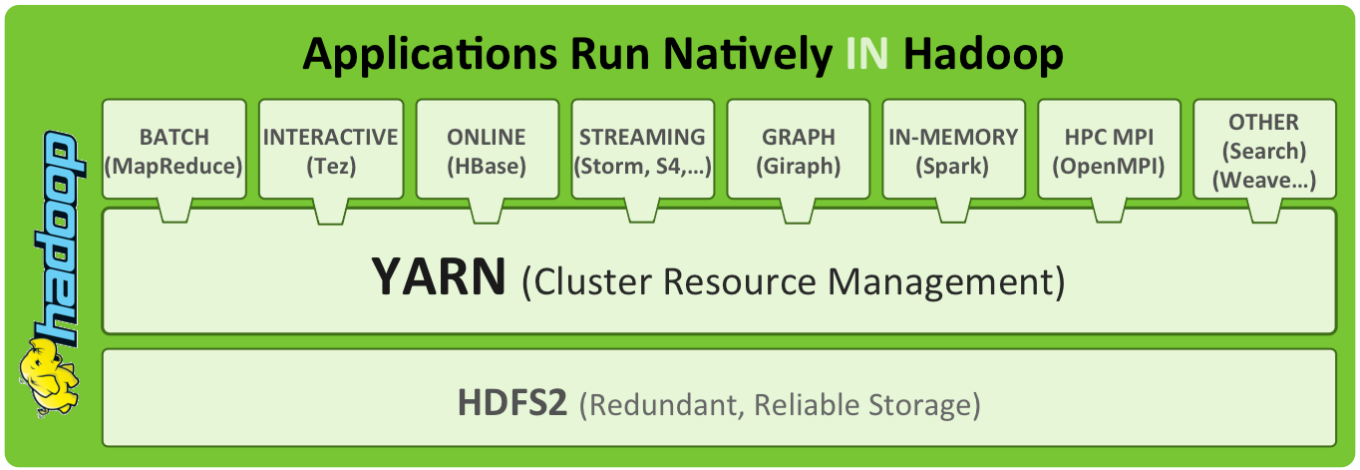
- Distributed Responsibility
- Different mechanisms for
  - RM Failure
  - NM Failure
  - AM Failure
- For Container Failure
  - Framework is the only one in charge

YARN in the real-world

- YARN at Yahoo!
  - No HW upgrade!
  - 5M jobs/day
  - Storage : 350PB
  - 4000 nodes => 7000 nodes
  - HDFS NameNode is a scalability bottleneck
    - Improve NameNode throughput
    - Log aggregation optimization

YARN in the real-world

- Appications and frameworks



  - REEF : Eases cost to write an application (meta-framework)
    - container reuse / caching / push-based control flow / fault-detection / checkpoint

Experiments

- Beating the sort record
  - http://sortbenchmark.org/Yahoo2013Sort.pdf
- MapReduce benchmarks
- Benefits of preemption
- Improvements with Apache Tez
- REEF : low latency with sessions

Related works

- Common inspiration & Goal
  - Scalability, latency, programming model flexibility
- Omega (Google)
- Corona (Facebook)
- Mesos (UC Berkeley)
- Cosmos (Microsoft)

Conclusions

- Key Contribution
  - Support a variety of Frameworks
  - More efficient resource scheduling
- Strong points
  - Requirements from History
  - Yahoo! workload
  - Local optimization
- Weak points
  - Memory management in JVM
  - Scheduling overhead

Questions?

Thank you