Apache Hadoop YARN: Yet Another Resource Negotiator

Rakesh Gopinath Nirmala Arunakumari Yedurupaka

I. Motivation

Apache Hadoop is a software used to deal with large amounts of data storage and processing in a shared and distributed computing platforms. In classic Hadoop MapReduce jobs are used to process data. With increased usage across diverse environments, it is found that Hadoop suffers from two rudimentary problems Hadoop is highly dependent on the underlying programming framework for resource management and its computations are handled in a centralized approach. These drawbacks limits the ease of adoption and scalability is reduced. Many other specific issues are found while using Hadoop for specific applications. To address these limiting factors, the authors have designed a better version of Hadoop called YARN that separates the programming model and resource manager and provides a more scalable solution with support to different programming models.

II. Contributions

One of the major contributions of the authors is the study and analysis of the real-world deployment of classical Hadoop at Yahoo. They clearly elucidate the real production issues (Scalability, Multi-tenancy, Serviceability, etc.) incurred with Hadoop usage and based on the analyses present us the findings that form the basis for development of more scalable and generic solutions.Next, based on the observations, they have designed and implemented a new Hadoop framework YARN. They have also deployed and evaluated it on Yahoo clusters and for functionality, performance, scalability and adaptability. The paper also outlines the improvements gained in comparison with classical Hadoop.

III. Solution

The proposed solution distributes various responsibilities to Resource Manager (RM), Application Master (AM) and Node Manager (NM). RM via its public interfaces maintains a global view of resources in a cluster. It accepts the resource requests from the applications, allocates bundle of resources based on the demand, applications priority and resources availability and deals with contention for resources. It also monitors the AMs and NMs running in the cluster. AM is responsible for an applications execution life cycle within the cluster. It negotiates with RM for resources, periodically informs its presence to RM monitoring module, handles application faults, performs computations and other involved optimizations. AM works with NM and reports the exit status of the application container to RM. NM running as a daemon is responsible for setting up the environment for the execution of the app container, authenticating the resource allocations, managing the dependencies between containers and is also tracked by RM. It is responsible for launching, killing and cleaning up the containers when the need arises. In addition, NM monitors the node for any issues and reports them to RM.

Container Launch Context (CLC) is the specification language used to define application containers and AMs. It describes the dependencies, security requirements, resources and necessary commands to launch processes. To make YARN reliable, persistent stores are used to store and load state information of RM when it crashes. Once RM recovers, all the old AM instances are destroyed and relaunched. NM failures are detected by RM and the corresponding Nodes containers are terminated. In temporary down scenarios, NMs resynchronize with RM. Container failures are managed by health checks and coordinated reporting between RM, AMs and NMs.

IV. Strong Points

1) The author successful decouples the resource management, job scheduling and cluster monitoring operations and provides a more scalable and modular architecture.
2) YARN is capable of interoperating with many existing frameworks such as MapReduce, Apache Tez, Dryad, Storm making it easier and flexible to use and adopt across diverse working environments.
3) As evident from the results, YARN outperforms classical Hadoop and other frameworks offering high efficiency, greater scalability and low latency
4) YARN with its reliable isolation between various functionalities offers many services than classical Hadoop as evident from deployment on Yahoo clusters

V. Weak Points

1) The Resource Manager with its centralized control would be a bottleneck for cluster performance and RM failure is kind of a Single Point Failure in the system though it can recover.
2) The paper mentions that RM deals with resource contention, however, it does not discuss how RM actually deals with contention issues and what is the delay or effect of contention on the system.
3) After RM recovery, all the related AMs are killed and relaunched without restoring the state. This leads to loss of work and could significantly delay data computations.

VI. Questions

For each of the following requirements, select appropriate scheduler(s) from the following: YARN, Mesos, Omega, Sparrow

1) Scalability to tens of thousands of machines is the key requirement.
   Oemga - With its shared state, lock-free, optimistic concurrent control, it reduces the waiting time and induces more parallelism and can scale well in comparison to monolithic schedulers.

2) Control over dierent scheduling policies is the key requirement.
   Mesos - suits this need well due to its decentralized scheduling model wherein we can employ multiple scheduling policies for different frameworks with each framework can take its own decisions.

3) You want to exploit data locality.
   Sparrow - it executes the jobs by assuming the inherent data locality properties resulting in lesser latency