Big Data Framework - Cluster Resource Management with Apache Mesos

Solaimurugan Vellaipandiyan
*National Resource Centre for Free and Open Source Software (**NRCFOSS**)*
Centre for Development of Advanced Computing(C-DAC)
Chennai, TamilNadu, India

Abstract- Extract the actionable information from massive and messy data need the different approach, data analytics tools are emerging to help user to extract actionable insights from Big Data. These approaches – namely an open source distributed framework called Hadoop, and data management by NoSQL databases. These takes different approaches for collecting, storing, data processing, analytics and applications than traditional database analytics tools and technologies. Big Data processing framework broadly categorized by batch processing and real time processing. The requirement of frameworks are different by nature, there is no single framework to satisfy them all. Running or experimenting multiple or same with its newer version framework on different cluster utilize poor resource management and need more storage for data to replicate for all frameworks. Both end up with money which reduce the ROI. Instead, organization want to share resource and data between different frameworks.

This is an extended version of Resource Management in Big Data Analytics - Integrating and Running diverse framework[1]. In this paper we are focusing on coexistence multiple data analytics tools that share the resource with the resource management framework called Apache Mesos, for that framework for managing Big Data will be presented along with ways to implement it around Apache Mesos. We also evaluated performance, scalability and data locality.

Keywords – spark cluster computing; Hadoop batch processing; big data analytics framework; dynamic resource sharing, Mesos support diverse framework.

I. INTRODUCTION

As data volume getting added day to day life, the huge amount of data generated through various source i.e. social media, e-commerce, health-care, manufacture, energy-scientific and finance sector etc.. it remains in structured and unstructured form [2]. Big Data study shows that nearly half the data (49%) is unstructured or semi-structured, while 51% is structured [3]. The heavy use of former, though it was nearly zero few years ago.

To rigging such a data and process by typical traditional data analytics from high volume and heterogeneous data sources remains daunting, expensive and time-consuming process to get the actionable insight. The traditional data management technologies does not meet the requirements, New concepts and models are emerging to meet these challenges [4]. As shows Fig 1, data analytics tools that emerging from the traditional approach to Big Data Analytics to and New Big Data Analytics for real-time streaming process.

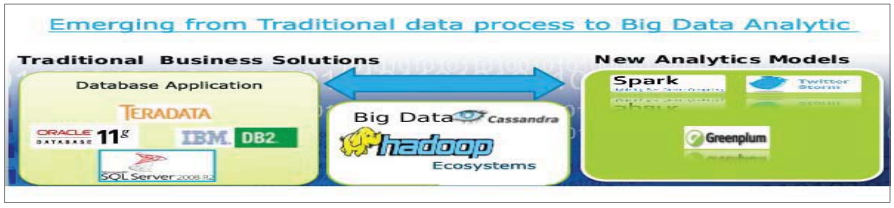

Figure 1. Data analytics tool emerging from traditional to real time streaming

Different data analytics frameworks such as open-source batch processing analytics tools Hadoop, Zettaset, HPCC system, Dremel, GreenPlum HD, ParAccel, GridGrain, HortonWorks and open source real time streaming analytics tools are Kafka, Yahoo's S4, Twitters Strom, Flume, Scribe, Google bigquery, Coludera impala, Titan, GraspJ, Gigaspaces, Parstream frameworks emerges to meet the specific requirement. A selection of these frameworks based on the computing environment, amount of data that can be processed, decision making capabilities, ease of use, data latency and scalability [23].

Running all of these frameworks in different cluster cast money and poor resource utilization. to avoid such case integration and multiple framework need run on same cluster and share resource across them for better resource utilization. This paper evaluates the performance and scalability of Hadoop and Spark frameworks to efficiently share the resources of the same cluster.

Table 1 summarize the important characteristics of traditional and Big Data approach.

TABLE I. CHARACTERISTICS OF TRADITIONAL VS BIG DATA APPROACHES

Characteristics	**Data Management**	
	Relational Data	***Big Data***
Architecture	Centralized	Distributed
Data Volume	Tera-bytes	Peta-bytes to Exa-bytes
Data Relational	Known relation	Unknown/complex relation
Data Model	Static or schema-based	Dynamic or Schema-less
Data veracity	Related Data	Messy/imprecise
Data Velocity	Generated by human interaction	Machine generated data sensor,medical and FSI
Data source	Known	Heterogeneous
Scalability	Nonlinear	Liner
Data processing	Interactive	Batch and stream processing

II. BIG DATA ANALYTICS TOOLS

A. Hadoop

Apache Hadoop is on open-source software for reliable, scalable, distributed computing for processing, storing and analyzing massive amounts of distributed, unstructured data across clusters of computer using a simple programming model[5]**.** Hadoop clusters run on commodity hardware which is inexpensive, so application can scale-out with minimal capital expenditure. Fundamental concept of Hadoop is, Rather than banging away at one, huge block of data with a single machine, Hadoop breaks up huge block of data into multiple parts by default 64MB block size and it distributed across Hadoop cluster nodes. Hadoop Map and Reduce functions can executed on smaller subsets of larger data sets, which is spread across cluster nodes, this provides the scalability[6].

The underlying architecture of Hadoop is HDFS (Hadoop Distributed File System). It provides fault-tolerance by replicating data blocks so that failures of nodes containing subsets of larger data set will not affect the computations that use that data. The NameNode in Hadoop architecture stores name space of data blocks, the DataNodes usually one per node cluster stores data blocks, and Map-Reduce for computation, JobTracker is used to track and submit the jobs, detects failure and TaskTracker to execute the job. Hadoop can accesses data via HDFS, which maps all the local disks of the computing nodes to a single file-system hierarchy, allowing the data to be dispersed across all the data/computing nodes[7].

B. Spark

Handling complex jobs, interactive queries and on-line processing all need one thing Hadoop lacks. Hadoop for batch processing and it is neither real time nor interactive [8]. Real Time data processing challenges are very complex. The real time system needs to handle the velocity of the data as well and handling the velocity of Big Data is not an easy task [9].

Spark is an apache incubator project, and it is open source data analytics cluster computing framework that aims to make analytics fast, both fast to run and fast to write[10]. Spark offers an abstraction called RDD (resilient distributed dataset), which provides efficient data reuse and fault-tolerant with out replication. Instead, they can rebuild lost data due to failure. The process of rebuilding a portion of the dataset relies on a fault-tolerance mechanism i.e *lineage.* With RDD, Spark can able to very elegantly unify the batch and streaming process into a

single comprehensive framework. Fig 2, shows that performance comparison with Hadoop[11]. Spark Streaming module is a extension and on top of Spark framework spark streaming module can process near real time data[12].

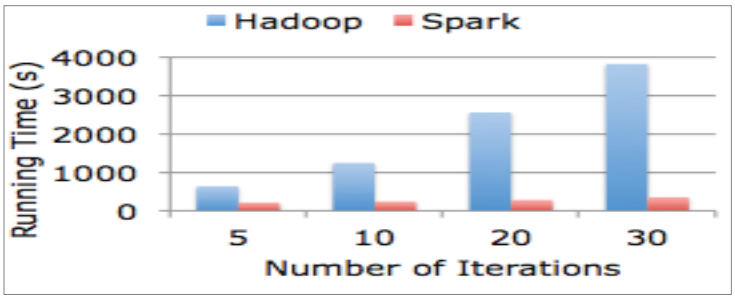

Figure 2. Logistic regression in Spark vs Hadoop

Spark framework for low-latency *iterative*, *interactive* jobs. Iterative can process different algorithm like machine learning and graph algorithms. Interactive job can load data into memory across a cluster and query it repeatedly.

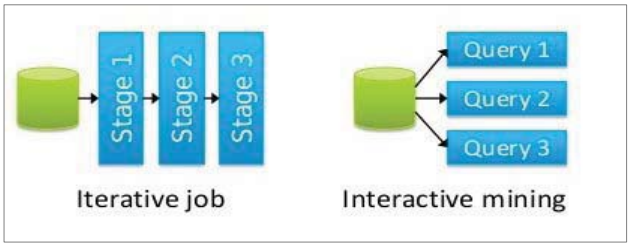

Figure 3. Spark data processing

III. Why Resource Management

Managing Big Data is critical and staying on top the latest Big Data analytics tools keeps developers in control and processing of data faster. It's not architecture to change each time the new Big Data analytics tools emerge, is an insight of information and how quick to get it. Keep the core framework as such and integrate new data analytics with exciting framework. It's not only reduce the time as it reduce the cost of setting a cluster and replicate data for to all analytic tools. Resource utilization of each framework will not always 100%. Fig 4, shows that Hadoop and Spark frame work installed on different cluster and processing job independently, which shows, current resource utilization pattern, each analytic tool on cluster uses only below half of the resource and even idle for some time.

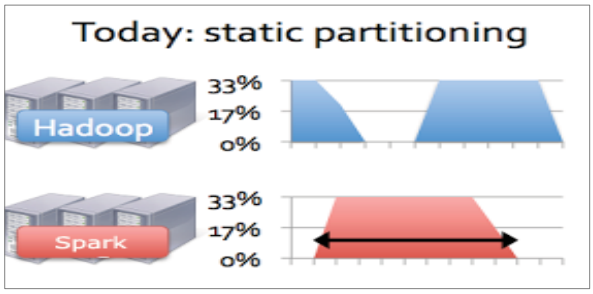

Figure 4. Multiple framework with out sharing resource

IV. RESOURCE MANAGEMENT SYSTEM - MESOS

Recently the Apache Software Foundation announced that Apache Mesos has graduated from the Apache Incubator to Top-Level Project status[13]. Mesos is a resource management platform for clusters. It aims to increase resource utilization of clusters by sharing cluster resources among multiple processing frameworks i.e Hadoop, Spark, MPI or multiple instances of same framework[14].

B. Mesos Architecture

Fig 5, Mesos architecture consists of a four components, namely Mesos-master, Mesos-slave, Framework and Scheduler.

Mesos-master: It's responsible for managing various frameworks, slave and allocate resource on slave to framework.

Mesos-slave: It's responsible for executing commands received from Mesos-master. Mesos-salve will send its own resource i.e CPU and RAM, to the Mesos-master for allocation.

Framework: Computing framework, i.e Hadoop, Spark,MPI etc., through MesosSchedulerDriver access Mesos.

Executer: its is mounted on Mesos-salve, run the framework task.

A framework running on top of Mesos consists of two components: 1.Scheduler 2.Executor. Scheduler registers with the master to be offered resources, and an executor process that is launched on slave nodes to run the framework's tasks. Mesos uses zookeeper for state maintenance, fault tolerance[18].

C. Mesos scheduling mechanism

Mesos use double layer scheduling mechanism in order to support diverse framework, first Mesos allocator allocate resource to the framework, then processing framework own scheduler to assign resource to the task. Mesos scheduling mechnanish based on fine grained resource sharing called resource offer[15]. Using this approch Mesos-slave reporting a mount of resource i.e CPU and RAM to the Mesos-master. Mesos-master uses DRF (Dominant Resource Fairness) to allocate amount of resource to the framewofrrk[19]. DRF is multi-resource max-min fair resource allocation mechanism, where max represents *max{CPU,mem}*, and min represents *min{user1,user2,user3...userN}=min{max{CPU1,mem1},max{CPU2,mem2},..}*. [22]

Mesos provide *reject offer* mechanism to solve the two major problem called resource requirment of the framework and data locality by simply denial of offereed resource similar in Hadoop delay scheduling mechanism[20]. To handle failure in job scheduling across distriputed process the following mechanishm achive the efficiency and roubtness.

Filter mechanism : Schedluling process need to commnicate with Mesos-master, its become network overhaed due to reject offer framework. To avoid this issue Mesos filter mechanism allows framework to receive only remaining resource is greater then L.

Rrescinds mechanism : Mesos recover and reallocate resource to some other framework if framework scheduler not assignd to task for certain period of time.

Algorithm 1 DRF pseudo-code

$R = \langle r_1, \cdots, r_m \rangle$ ▷ total resource capacities
$C = \langle c_1, \cdots, c_m \rangle$ ▷ consumed resources, initially 0
$s_i \ \ (i = 1..n)$ ▷ user i's dominant shares, initially 0
$U_i = \langle u_{i,1}, \cdots, u_{i,m} \rangle \ \ (i = 1..n)$ ▷ resources given to user i, initially 0

pick user i with lowest dominant share s_i
$D_i \leftarrow$ demand of user i's next task
if $C + D_i \leq R$ **then**
  $C = C + D_i$ ▷ update consumed vector
  $U_i = U_i + D_i$ ▷ update i's allocation vector
  $s_i = \max_{j=1}^{m} \{u_{i,j}/r_j\}$
else
  return ▷ the cluster is full
end if

D. Resource Offer

Mesos distributed fine-grained scheduling mechanism called resource offers, which enables Mesos to decide how many resources to offer to each framework and scheduler in framework will select which offered resource to use. DRF policy is performed as to equalize the fraction that each framework shares of its dominant resource. For example, if a cluster contains 100 CPUs and 500GB RAM, and Hadoop needs 4 CPUs and 5 GB RAM to execute each task and Spark needs 1 CPU and 40GB RAM per task, then the policy allocate 80 CPUs and 100 GB memory for Hadoop to execute 20 tasks and allocates 10 CPU cores and 400GB memory for Spark to execute 10 tasks. Thus, the fraction of dominant resource CPU of Hadoop is 80%, which is equal to the fraction of the dominant resource RAM of Spark.

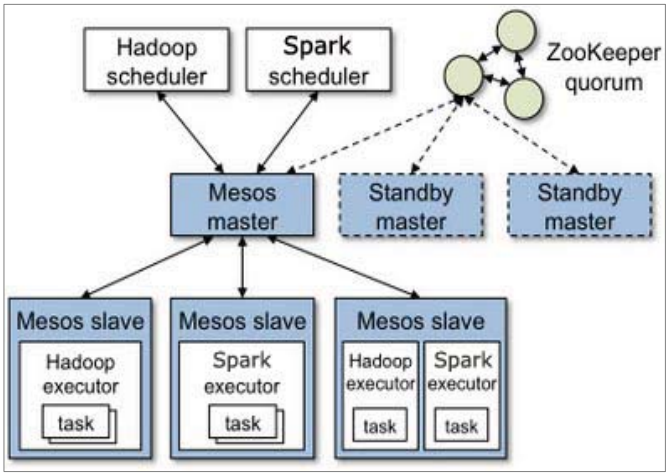

Figure 5. Mesos Architecture

V. EVALUATION

As for experiment, evaluated Mesos on Amazon elastic commuting cloud(EC2). Performed system resource shares across 4 framework, data locality and scalability.

A. Macro-benchmark

To evaluate the performance of running multiple frameworks to efficiently share the resources of the same cluster, a macrobenchmark is executed consisting of two Hadoop frameworks and two spark frameworks. A number of jobs with different sizes are executed on the four frameworks. Hadoop runs WordCount jobs and spark runs text search through the error messages in a log file. We used twenty five EC2 instances each with 4 CPU cores and 10 GB RAM.

B. Data locality

93 EC2 instances each with 6 CPU cores and 20 GB RAM are employed to perform the evaluation of data locality. Three scenarios are considered: each instance runs 3-4 static partition of the cluster, all instances using either no delay scheduling, 1s or 5s delay scheduling. Using Mesos improves data locality to a great extent even without delay scheduling. With one second delay scheduling, the data locality exceeds 90% and with five seconds delay scheduling the data achieves 95% locality. In addition, as we can see that the job running time decreases with the use of delay scheduling.

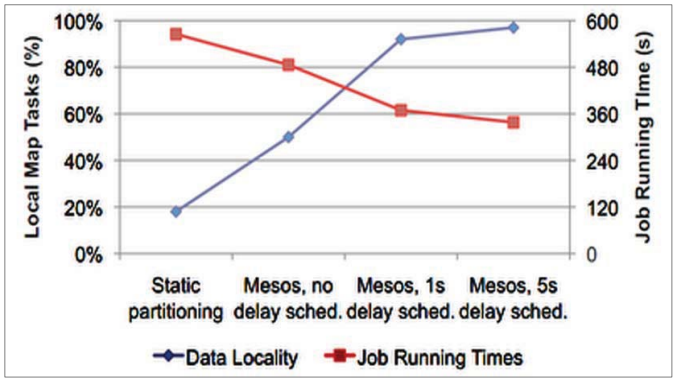

Fig 6. Data locality and job running time-consuming

C. Scalability

To evaluate the scalability of Mesos, 50000 salve daemons on 99 instances each with 12 CPU cores and 9GB RAM are used. Each of 200 frameworks continuously launches one task after receiving an offer. Each task from framework sleep for a particulaer period based on a normal distribution with mean of 20 seconds with a standard deviation of 5, before end. Once the cluster achieves the stable state, a single framework will be launched to run a single 20 second task. Fig 9, shows the average of 4 runtimes for launching the single framework after reaching stable state. Becase of the limits of EC2[21], the number of slaves limits to 50000. With the 30 second average task length, Mesos imposes less than 1 second overhead on frameworks, which achieves high scalability.

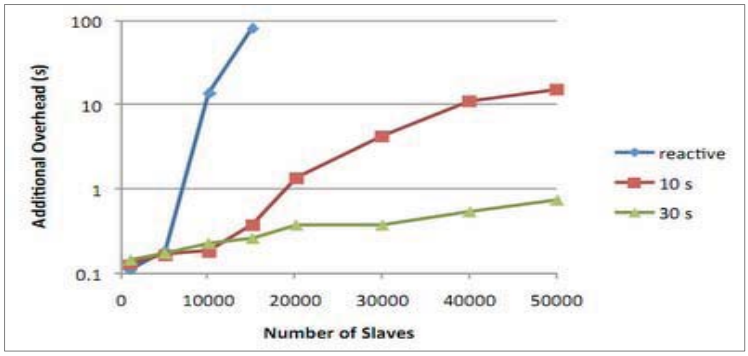

Fig 7. Scalability

VI. Challenges and Issues

Distributed computing framework used for process the Big Data, the security is more important element. Secure computations in Distributed programming frameworks as well as security across the framework with Mesos. The recent project Add Security and Authentication support to Mesos[16], aim to provide only authenticated Framework will summit and run a job. Multiple framework sharing the data across, data synchronization issue between framework running on top of Mesos.

VII. Conclusion and Future Work

Innovation leads to emerge of new Big Data analytics and different framework suit for different purpose. Organization needs a framework like Mesos to provide better resource utilization and reduce the cost of data replicate. In case of computations are operated not on the same node with the data needed, the cost of computations will be tremendously expensive[17]. Therefore, it is desirable to employ a fine-grained scheduling model, where applications take turn performing computations on each node. Existing framework, such as Hadoop and Spark has implemented its own fine-grained scheduling model to achieve high performance. However, since each framework is designed independently, no way can be found to performance a fine-grained scheduling model for a set of frameworks. Mesos is developed to enable fine-grained sharing across a set of computing frameworks. Further in this area we are focusing on the running multiple framework on state data center to serve different e-Governance application.

REFERENCES

[1] Solaimurugan Vellaipandiyan, Resource Management in Big Data Analytics : Integrating and Running diverse framework, presented for Proceedings of International Conference on Computing, Cybernetics and Intelligent Information Systems (CCIIS) 2013.

[2] Impetus white paper, March 2011, "Planning Hadoop/NoSQL Projects for 2011" by Technologies

Available:http://www.techrepublic.com/resource-library/whitepapers/planning-hadoop-nosql-projects-for-2011/

[3] Big Data Study – The 10 Key Findings by TCS

http://sites.tcs.com/big-data-study/kinds-of-digital-data/

[4] Dr. G V N Appa Rao, PLENARY TALK on Big data and Real Time Analytics, Recent Trends in Information Technology (ICRTIT), 2011 International Conference ISBN : 978-1-4577-0588-5 http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5972495

[5] http://hadoop.apache.org/

[6] David Floyer, David Vellante Original publication date: February 12, 2013 New Approaches to Big Data Processing and Analytics,

http://articulosit.files.wordpress.com/2013/03/new-approaches-to-big-data-processing-and-analytics.pdf

[7] Apache Hadoop documentation : http://hadoop.apache.org/docs/stable/index.html

[8] Mike Driscoll, CEO of MetaMarkets http://www.forbes.com/sites/danwoods/2012/07/27/how-to-avoid-a-hadoop-hangover/

[9] Dibyendu Bhattacharya, Manidipa Mitra, ANALYTICS ON BIG FAST DATA USING REAL TIME STREAM DATA PROCESSING ARCHITECTURE
http://www.happiestminds.com/sites/default/files/article/ANALYTICS_ON_BIG_FAST_DATA_USING_A_REALTIME_STREAM_DATA_PROCESSING_ARCHITECTURE.pdf

[10] AmpLab web page : https://amplab.cs.berkeley.edu/projects/spark-lightning-fast-cluster-computing/

[11] Matei Zaharia, Mosharaf Chowdhury, Michael J. Franklin, Scott Shenker, Ion Stoica Spark: Cluster Computing with Working Sets

http://dl.acm.org/citation.cfm?id=1863113

[12] Data intensive systems: Real Time stream processing. https://www.cs.duke.edu/~kmoses/cps516/dstream.html

[13] https://blogs.apache.org/foundation/entry/the_apache_software_foundation_announces45

[14] Apache Mesos. http://mesos.apache.org/

[15] Mesos Architecture https://github.com/apache/mesos/blob/master/docs/Mesos-Architecture.md

[16] Add security and authentication support to Mesos (including integration with LDAP). https://issues.apache.org/jira/browse/MESOS-418

[17] https://sites.google.com/site/cps516mesosproject/my-page

[18] Tom White, Hadoop: The Definitive Guide, 3rd edition,

[19] Ali Ghodsi, Matei Zaharia, Benjamin Hindman, Andy Konwinski, Scott Shenker, Ion Stoica, Dominant Resource Fairness: Fair Allocation of Multiple Resource Types. Published in, NSDI'11 proceedings of the 8th USENIX conference on Networked systems design and implementation, Pages 24-24

[20] Matei Zaharia,Dhruba Borthakur, Joydeep Sen Sarma, Khaled Elmeleegy, Scott Shenker, Ion Stoica. Delay scheduling: a simple technique for achieving locality and fairness in cluster scheduling, Proceeding EuroSys '10 Proceedings of the 5th European conference on Computer systems.Pages 265-278

[21] Cloud Managment blog from RightScale blog.

http://www.rightscale.com/blog/cloud-management-best-practices/benchmarking-load-balancers-cloud

[22] Ali Ghodsi, Matei Zaharia, Benjamin Hindman, Andy Konwinski, Scott Shenker, Ion Stoica. Dominant Resource Fairness: Fair Allocation of Multiple Resource Types

[23] Udaigiri Chandrasekhar, Amareswar Reddy, Rohan Rath, A Comparative Study of Enterprise and Open Source Big Data Analytical Tools Proceedings of 2013 IEEE Conference on Information and Communication Technologies (ICT 2013)