Though MapReduce supports different use cases, it is not suitable for large scale computation and is not flexible to integrate a wide range of emerging projects.

2.4 Addressing the availability issue

2.4.1 Tasktracker Failure

If a tasktracker fails by crashing, the JobTracker stops receiving heartbeat signals. Once the JobTracker notices a failed tasktracker, it removes the node from the list of

such a way that each user gets a share of cluster capacity. The capacity scheduler maintains a cluster in the form of queues, and the jobs in the queue are schedule based on FIFO scheduling.

3. 2 Application Master

The Application Master [2] takes the role of TaskTracker of classic MapReduce. The Application Master has control on all the applications that runs in the cluster. This component is responsible for requesting resources from Resource Manager and coordinating the execution of all the tasks within the application. The Application Master also monitors the resource usage, resource allocation. The applications are run by Application Master using the containers that are

The new architecture makes no difference between resources available for map tasks, resources available for reduce tasks. The default memory allocations for map and reduce tasks are avoided in the enhanced approach by using memory based scheduling. Problems of over and under utilization of resources is eliminated by replacing the allocation of slots to the nodes with containers, which are fine-grained. Each node

is configured with a set of memory, cpu[8] for executing their tasks. Nodes are configured with a number of virtual cores, which are resources meant to represent parallelism. In almost all cases, a node's virtual core capacity on the Node Manager should be set as the number of physical cores on the machine**.**

Applications may request a memory capability between the minimum allocation and a maximum allocation. Default memory allocations are scheduler-specific, and for the capacity scheduler the default minimum and maximum are 1024 MB, 10240 respectively. The tasks can obtain memory between these two values by setting *mapreduce. map. Memory. mb, reduce. reduce. memory. mb*[8] appropriately. This approach drastically improved resource utilization as the resource needs and capacity can be balanced easily. This approach also enables the sharing of resources between MapReduce and other frameworks, allowing more sensible and finer-grained resource configuration for better cluster utilization.

3. 6 Non-Mapreduce Workloads

Mapreduce is considered as one of the instance of YARN application. By decoupling resource management and scheduling tasks from JobTracker, YARN provides support for more varied processing approaches and a broader array of applications. Routines that repeat many times across a data set, such as machine learning algorithms or interactive real time Business Intelligence applications, processing large data streams where immediate computation is required, attempts to define a parallel processing model for graph processing that can work on Hadoop-style clusters are being implemented on YARN. YARN can be used for the creation of new frameworks and execution models that can leverage both the compute power of an Hadoop cluster and its rich data storage models to solve specific new classes of problems.

3. 7 High Availability

YARN is more available compared to classic MapReduce by eliminating the single points of failure.

3. 7. 1 Application Master Failure

In the event of Application Master Failure [4], the Resource Manager fails to receive heartbeats. Once the Resource Manager detects the failure, a new instance of Application Master running on new container is started. The state of MapReduce tasks on the failed Application Master can be recovered avoiding re-running them. The client interacting with Application Master for application status, contacts Resource Manager to locate the new instance of Application Master in the case of failure.

3. 7. 2 Node Manager Failure

In the case of Node Manager failure[4] failure, it stops sending heartbeats to the resource manager. The Resource Manager removes the failed Node Manager from the list of available nodes. The applications on the failed Node Manager can be recovered.

3. 7. 3 Resource Manager Failure

Failure of Resource Manager [4] is the most serious issue since the jobs or containers can't be launched without it. This issue is overcome by checkpoint mechanism by saving the state of Resource Manager to persistent storage. On recovery from failure the Resource Manager's state contains the Node Managers and running applications and the Application Master checkpoints completed tasks, so the completed tasks need not be re-run.

3. 8 Job Submission Process

When a client submits an application, the Resource Manager launches Application Master to run the application. The Resource Manager maintaining the status of available resources initiates the scheduler to decide the allocation strategy. The Application Master once launched, registers with Resource Manager and requests the Resource Manager for containers required for application execution. Once the resources are allocated the Application Master interacts with Node Manager for launching the required containers. When the application execution completes the Application Master releases all the allocated resources, de-registers with Resource Manager.

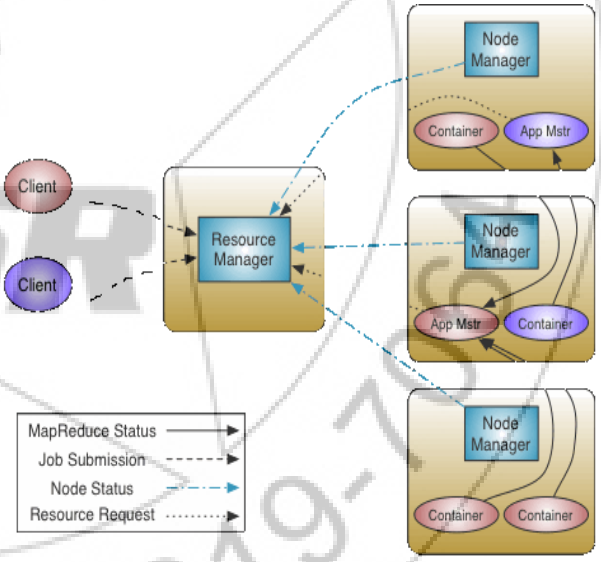

Figure 2: Running applications on YARN

4. Experimental Results

The Hadoop TeraSort program was used to run GraySort and MinuteSort benchmarks[7], using Hadoop HDFS to store the input and output. The input data was generated with gensort version 1. 5. For GraySort, the data was 102. 5TB in size, spread across 1025 files each with 100,000,000,000 bytes. For the Indy MinuteSort, the data was 1612. 22GB (1612223312700 Bytes) in size, spread across 1001 files each with 1610612700 bytes. For Daytona MinuteSort, the data was 1497. 86 GB (1497869841679 Bytes) in size, spread across 920 files each with 1610612700 bytes. Both

skewed and non-skewed data were generated as needed. The experiments were run on 2100 nodes each equipped with two 2. 3Ghz hexcore Xeon E5-2630, 64 GB memory, and 12x3TB disks each. This is shown in Table1.

The benchmarks on a 260 node cluster comparing MapReduce with YARN [2] is provided in Table2. Each node is running 2. 27GHz Intel(R) Xeon(R) CPU totaling to 16 cores, has 38GB physical memory, and 6x1TB 7200 RPM disks each, formatted with ext3 file-system. The network bandwidth per node is 1Gb/sec. Each node runs a DataNode and a Node- Manager with 24GB RAM allocated for containers. 6 maps and 3 reduces are run in 1. 2. 1, and 9 containers in 2. 1. 0.

Table 1: Daytona and GraySort benchmarks

Benchmark	Data Type	Amount of Data Sorted	Time	Rate	Duplicate Keys
Daytona GraySort	non-skewed	102.5 TB	72 minutes 8.053 seconds	1.42 TB per minute	0
Daytona GraySort	skewed	102.5 TB	117 minutes 48.261 seconds	0.87 TB per minute	31867643140
Indy MinuteSort	non-skewed	1612.22 GB	58.027 seconds		0
Daytona MinuteSort	non-skewed	1497.86 GB	59.223 seconds		0
Daytona MinuteSort	skewed	1497.86 GB	1 minute 27.242 seconds		0

Each map occupies 1. 5GB JVM heap and 2GB total memory, while each reduce takes 3GB heap 4GB total. JobTracker/ResourceManager run on a dedicated machine so is the HDFS NameNode.

Table 2:. Hadoop benchmarks results

Benchmark	Avg runtime (s)		Throughput(GB/s)	
	1.2.1	2.1.0	1.2.1	2.1.0
RandomWriter	222	228	7.03	6.84
Sort	475	398	3.28	3.92
Shuffle	951	648	-	-
AM Scalability	1020	353/303	-	-
Terasort	175.7	215.7	5.69	4.64
Scan	59	65	-	-
Read DFSIO	50.8	58.6	-	-
Write DFSIO	50.82	57.74	-	-

5. Conclusion

Hadoop continues to grow as one of the popular distributed data processing frameworks in big data market. However a number of issues are yet to be addressed and resolved to make hadoop suitable for large scale data workloads. YARN has evolved as a next generation compute platform providing greater scalability, availability, sharing on Hadoop cluster. YARN is the re-architecture of Hadoop that is responsible for managing and monitoring workloads, maintaining a multi-tenant environment, implementing security controls. YARN provides significant advantages on classic MapReduce, allowing the development of new distributed applications beyond MapReduce.

6. Future Work

YARN is still undergoing major improvements to bring multi-workload capabilities to Hadoop. As YARN provides flexibility for emerging data access, data management, security and integration tools, new projects such as Spark, Storm, Tez, Knox are being integrated into Hadoop.

References

[1] Apache hadoop. http://hadoop. apache. org.
[2] V. K. Vavilapalli, A. C. Murthy, C. Douglas et al. , “Apache hadoop yarn: Yet another resource negotiator,” in Proceedings of the 4th annual Symposium on Cloud Computing. ACM, 2013.
[3] J. Dean and S. Ghemawat, “MapReduce: simplified data processing on large clusters,” Proceedings of the 6th Symposium on Opearting Systems Design & Implementation, 2004.
[4] T. White, Hadoop: The definitive guide. ” O’Reilly Media, Inc.”, 2012.
[5] HADOOP YARN:http://hortonworks. com/hadoop/yarn [6] T. Graves. GraySort and MinuteSort at Yahoo on Hadoop 0. 23 http://sortbenchmark. org/Yahoo2013Sort. pdf, 2013.
[6] Karthik Kambatla, Wing Yew Poon, and Vikram Srivastava “How Hadoop YARN HA works “ http://blog. cloudera. com/blog/category/yarn “how-yarn-overcomes-mapreduce-limitations-in-hadoop-2-0” http://saphanatutorial. com/
[7] Dan Sullivan “Hadoop 2 vs. Hadoop 1: Understanding HDFS and YARN” www. tomsitpro. com/articles/hadoop-2-vs-1,2-718. html

[8] James Kobielus “YARN unwinds MapReduce's grip on Hadoop” http://www. infoworld. com/
[9] O'Reilly Strata “An Introduction to Hadoop 2. 0: Understanding the New Data Operating System” http://radar. oreilly. com
[10] http://ibm. com/developerworks/library/bd- hadoop yarn
[11] B. Hindman, A. Konwinski, M. Zaharia, A. Ghodsi, A. D. Joseph, R. Katz, S. Shenker, and I. Stoica. ” Mesos: a platform for fine-grained resource sharing in the data center”. In Proceedings of the 8th USENIX conference on Networked systems design and implementation, NSDI’11, pages 22–22, Berkeley, CA, USA, 2011. USENIX Association.
[12] M. Schwarzkopf, A. Konwinski, M. Abd-El- Malek, and J. Wilkes. “Omega: flexible, scalable schedulers for large compute clusters”. In Proceedings of the 8th ACM European Conference on Computer Systems, EuroSys ’13, pages 351–364, New York, NY, USA, 2013. ACM.
[13] R. Chaiken, B. Jenkins, P. -A. Larson, B. Ramsey, D. Shakib, S. Weaver, and J. Zhou. “Scope: easy and efficient parallel processing of massive data sets”. Proc. VLDB Endow. , 1(2):1265–1276, Aug. 2008.

Author Profile

G. Prabhakar Raju received the M. C. A degree from Osmania University and M. Tech degree in Computer Science and Engineering from JNTU University. He is an associate professor in the Department of Computer Science and Engineering, Anurag Group of Institutions. His research interests include data mining, data warehousing, bigdata, hadoop, natural language processing and artificial intelligence.

Sandeep Raj R received the B. Tech degree in Information Technology from JNTU Hyderabad. in 2012. He is pursuing M. Tech in Computer Science from JNTU Hyderabad. His research interests include bigdata, hadoop and cloud computing.