IJFANS INTERNATIONAL JOURNAL OF FOOD AND NUTRITIONAL SCIENCES
ISSN PRINT 2319 1775 Online 2320 7876
Research paper Volume 10, Iss 1, 2021

Spark Cluster Performance When Using a Cloud Service vs. a Typical Configuration

B. V. Ramana1, B. R. Sarath Kumar2*
1Dept of IT, Aditya Institute of Technology and Management, Tekkali, AP, India.
2Dept of CSE, Lenora College of Engineering, Rampachodavaram, A.P, India.
*Corresponding Author: iamsarathphd@gmail.com

Abstract

Apache Spark was developed as a free and open-source cluster computing platform for handling massive amounts of data. This article focuses on setting up a Spark cluster in the cloud using OpenStack. Spark cluster as a service is compared to a traditional Spark cluster using the HiBench test suite. Using Spark as a cloud service yields more promising results in terms of time, effort, and throughput, as shown by the results.

*Keywords:*Benchmark; Cloud Service; OpenStack; Spark Cluster; HiBench;

1. Introduction

New technological breakthroughs have resulted in an explosion in the volume and variety of data sources. The Internet of Things (IoT), RFID-equipped locations, social networks, massive eCommerce, phones, credit cards, atmospheric research, medical records, railroads, buses, biological, astronomical, genomic, military surveillance, video archives, and photographic archives are only some of the sources available today. The term "big data" describes datasets whose size exceeds the capabilities of traditional database software tools to handle, record, store, and analyze. This might be because of its volume, velocity, or complexity. While "big data" is sometimes referred to as a "technology," it really encompasses a wide range of approaches, from parallel processing to distributed file systems to virtualized in-memory database systems and beyond. [1], [8].

Today's big data computing problems span a wide range of areas, from storage and processing to administration and analytics to visualization. The processing of data is the most difficult part of this. Apache Hadoop, Apache Spark, HPCC, HPCC Systems (High Performance Computing Cluster), Storm, Lambdoop, etc. [2] are just some of the many programming frameworks available to analyze such massive amounts of data.

Apache Spark is a free and open source framework for working with large datasets. It all began as a study at UC Berkeley's AMPLab. It's a cluster computing engine that can be used for a variety of purposes, and it has libraries for things like streaming, machine learning, and graph processing. Java, Python, and Scala APIs are also available. Spark is an open-source big data framework with the primary goals of simplicity, advanced analytics, and rapid execution. Spark's in-memory cluster computing functionality [3, 4, 5] allows for faster processing times.

Large computational infrastructure, costly software, and substantial effort are all necessitated by the complexity and problems of big data computing. In the end, cloud computing is the answer to all of these issues. It does this by making resources available on-demand and charging for them in accordance with their actual use. Additionally, the system may be swiftly scaled up or down to meet the needs of the business [6]. Complex, large-scale computations are within the capabilities of cloud computing. The requirement for costly on-site upkeep of specialized space, processing gear, and software is removed [7].

In this article, we will focus on setting up an Apache Spark cluster as a SAAS on OpenStack cloud. Scalability, backup and restore capability, user-friendliness, speed, throughput, affordability, and more are just some of the numerous advantages of offering Apache Spark as SAAS [8]. This study presents an in-depth evaluation of Spark cluster's efficiency as a SAAS. This is achieved by contrasting the output of a Spark cluster set up as a cloud service with that of a more traditional setup. The HiBench large data benchmark suite is used for the comparison. Nine benchmarks are run on both Spark cluster implementations to facilitate the comparison. Micro benchmarking, Web search, machine learning, and

ISSN PRINT 2319 1775 Online 2320 7876
Research paper Volume 10, Iss 1, 2021

analytical querying are the four subsets that make up these standards. Micro benchmarks consist of the following: Sort, WordCount, Sleep, and TeraSort. Both PageRank and Bayesian Classification may be seen as examples of best practices in web search and machine learning, respectively. The analytical query utilizes Hive Join, Hive Scan, and Hive Aggregate as its three separate benchmarks [9]. The final findings show how an OpenStack-hosted Apache Spark cluster outperforms a traditional cluster in terms of both performance and throughput.

Here is how the rest of the paper is structured: Works cited in Section 2 are discussed. Section 3 depicts the deployment of an Apache Spark cluster using both a SAAS on cloud and a traditional cluster. The HiBench benchmark suite and the performance indicators utilized in this study are described in Section 4. In Section 5 we provide a comprehensive analysis of the available performance data for Spark. Section 6 provides the summary and conclusion.

2. RelatedWork

The biggest open source communities are those working on OpenStack and Apache Hadoop, respectively. Users of both communities will benefit from their merging. There have been various attempts in this direction, the most noteworthy of which being the SAHARA project [11]. Hadoop clusters in the cloud are managed and set up with the help of this integration. Even though Spark is now officially supported by SAHARA, it can only be deployed in standalone mode without YARN or Mesos [12]. The innovative aspect of our study is that we have successfully implemented Spark cluster as a service in a distributed fashion on an Openstack cloud with complete YARN support.

Fig.1.ComponentsofSpark[4]

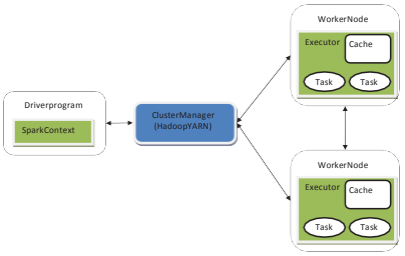


Fig.2. ApacheSparkClusterManager[4]

3.ApacheSparkClusterDeployment

Apache Spark was developed in 2009 at AMPLabs, UC Berkeley, and is a cluster-computing framework that is open source, expressive, fast, and general-purpose. It's now one of the most prominent Apache initiatives. It's a high-powered processing tool with advanced analytics, user-friendliness, and rapid processing speeds. It has high-level application programming interfaces (APIs) for the languages Scala, Java, R, and Python. Spark also provides a highly tuned engine for use with generic execution graphs. A wide variety of high-level tools, including as MLlib for machine learning, GraphX for graph processing, Spark Streaming, and Spark SQL for SQL and structured data processing, are also supported. Spark can handle a broad variety of tasks, from streaming to iterative algorithms to streaming to interactive queries. There is less work for administrators to do in terms of monitoring and updating several tools when utilizing the Spark framework [3, 4, 5]. As can be seen in Fig. 1, the current Apache Spark stack allows a wide variety of add-ons.

The experiment was carried out by setting up two identical spark clusters, one as a cloud service and the other on regular workstations. As can be seen in Fig. 2, Spark clusters are constructed atop Hadoop, with HDFS serving as the data storage layer and YARN handling cluster administration. Apache Spark cluster startup calls for the Spark daemon in addition to the Hadoop daemon. Starting the namenode, secondarynamenode, resourcemanager, and master daemon processes on the master node is essential. Nodemanager, datanode, and worker are the three daemon processes that are launched on a slave node.

ISSN PRINT 2319 1775 Online 2320 7876
Research paper Volume 10, Iss 1, 2021

Table 1 displays the Spark cluster's setup settings.
An OpenStack-based cloud hosting a Spark cluster was put up. One controller node, one network node, and fifteen computing nodes make up the resource-rich cloud. Each node runs Ubuntu server 14, which is installed on top of a hardware configuration that includes a Quad-Core AMD OpteronTM CPU, 8 GB RAM, and roughly 200 GB disk. The configuration of the controller nodes is twice as large. In order to set up a Spark cluster as a service with four nodes, four distinct instances are created, with one node serving as the master and the other three serving as slaves. Each Spark cluster node in the cloud has 4 GB of RAM, 50 GB of disk space, two processing cores, and Ubuntu server version 14.

In the same way, the standard Spark cluster's configuration required the establishment of base computers. The nodes run on an AMD OpteronTM CPU with 4 GB of RAM, 20 GB of disk space, and 2 processing cores, with the Linux CentOS Release 6.5 64 bit Kernel providing the underlying software. The Spark cluster consists of four computers, one of which serves as the master and the other three as its slaves.

4.SparkBenchmarksandPerformanceMetrics

ThissectionmainlydiscussesabenchmarksuiteknownasHiBench whichisusedtotesttheperformanceofSparkcluster.Also,theperformancemetricsnamelyelapsedtime,throughput,andspeeduparediscussedhere.

4.1. HiBenchBenchmarkSuite

HiBench is a comprehensive and representative benchmark suite for Hadoop, Spark, Storm, Storm-Trident andSamza.Itconsists of a setof programs including both real-worldapplications and synthetic micro-benchmarks.Foreach workload, the input data of benchmarks is automatically generated by using prepare scripts. Presently, HiBenchcontainsthirteenworkloadswhichareclassifiedintofivecategories[9],[10].

HiBenchprovidesfourdifferentworkloadsforMicrobenchmarksinSpark.Allfourareusedintheexperimentbeingdiscussedinthepaper.Thedescriptionaboutthesebenchmarksisasfollows:

- Sort:ItSortsthe*text*inputdata,whichisproducedusingRandomTextWriter.
- WordCount:Itcountstheexistenceofeachwordintheinputdata,whichareproducedusingRandomTextWriter.
- TeraSort:TeraSortisastandardbenchmarkcreatedbyJimGray.ItsinputdataisgeneratedbyHadoopTeraGen.
- Sleep:Thisworkloadteststheframeworkschedulerbysleepinganamountofsecondsineachtask.

HiBench provides two workloads namely NutchIndexing, PageRank for Web Search benchmarks. The PageRankusedinthisexperimentasitisresponsibletobenchmarkPageRankalgorithmimplementedinSpark-MLLib/Hadoop. The data source was generated from Web data.In case of Machine Learning,HiBench providefollowing two workloadsnamely Bayesian Classification and K-means Clustering. The Bayesian classification wasused in thispaper;it is responsible tobenchmarka popular Classification algorithm known asNaiveBayesian.Forthe task of analytical query three benchmarks namely Join, Scan and Aggregate are available in HiBench. They areresponsibletoperformthetypicalOLAPqueriesbyHivequeries.Also,itrequiresautomaticallygeneratedwebdataasinputsource.Allofthesethreebenchmarkshavebeenusedinthispaper.

Hence,total nine benchmarksof HiBench suiteare used in this paperincluding WordCount,TeraSort,Sleep,Sort,Bayesian,Aggregate,Join,PageRank,andScanrespectively.Theinputdata isvaryingasitisgeneratedautomaticallyforeachbenchmarkusingthepreparescript.Thecomputationshavebe

ISSN PRINT 2319 1775 Online 2320 7876
Research paper Volume 10, Iss 1, 2021

enperformedmultipletimes,thusthe averageresultvaluesare reportedinthispaperforboththeSparkclusters.

4.2. PerformanceMetrics

- ElapsedTime:Itisthetimerequiredtoperformanevent.Itisthedifferencebetweenbeginningtimeandanendingtime.Itcandifferentiatetheperformanceaslesselapsedtimeindicatesgoodperformance.Itismeasuredin seconds.
- Throughput:Itistheamountofworkthatcanbeperformedinagivenperiodof
- time.Itismeasuredin bytes/seconds.
- Speedup:ItistheratioofT1overTNwhichiselapsedtimeof*1*and*N*workers.

5.ResultAnalysis

This article primarily focuses on demonstrating how to deploy an Apache Spark cluster as SAAS on the OpenStack cloud. Using the HiBench benchmark suite, we compare the performance of a Spark cluster hosted in the cloud to that of a traditional on-premises Spark cluster.
When running on a traditional cluster, the final statistics for each benchmark are shown in Table 2. This includes the input data size, execution time, and throughput. When running benchmarks on a Spark cluster set up as a cloud service, the final results for each test are shown in Table 3. When compared to a traditional cluster, the results of running the benchmark Aggregate on the cloud are more encouraging. If you look at Fig. 3, you'll find that using the cloud to run Aggregate as SAAS results in a substantially smaller elapsed time compared to using a traditional Spark cluster. Similarly, running the TeraSort algorithm on the cloud significantly improves performance, as measured by the amount of time saved. Results are similarly consistent when using other performance measures, such as Join, PageRank, Scan, Sort, and WordCount. Again, Spark in the cloud provides faster results than a traditional cluster setup in all of our test executions. When compared to traditional clusters, benchmark Bayesian analysis takes much less time on a cloud-based Spark cluster. Finally, in the case of the sleep test, Spark as a SAAS again outperforms its competitor. Even though there is no I/O involved in the Sleep test, its busy wait state uses quite a lot of CPU time. All the experiments run on OpenStack cloud as a SAAS demonstrate greater performance compared to Spark's typical cluster, both in terms of raw throughput (in bytes per second) and throughput per node. In Figure 4, we see a bar chart of throughput in bytes per second, whereas in Figure 5, we see a bar chart of throughput per node. The greater promise of Spark as a SAAS can be seen in Figs. 4 and 5, where the bars are noticeably bigger than in the case of Spark's traditional cluster. For both cases, the speedup is also estimated, providing quantitative evidence of Spark's superiority as a cloud service compared to traditional cluster. When compared to other benchmarks, TeraSort's speedup of 3.384 is the highest.

Table 4 summarizes CPU consumption and system load for all nine of the abovementioned benchmarks, providing more insight into the performance of both clusters. Micro benchmarks contain largely system or user CPU operations with little I/O wait time. The TeraSort test, on the other hand, demonstrates significantly increased CPU participation in the I/O delay. Table 4 indicates that, with the exception of TeraSort, all of the Micro workloads are heavily CPU bound and just marginally I/O constrained. As can be observed, the CPU is only exhibiting activity for a certain period of time throughout the benchmark Sleep's idle wait. There is greater processing time spent on user activities in the PageRank benchmark. Although there is still some I/O delay, it is shorter. The user's job also predominates in the CPU processing time for the Bayesian workload. The

Processing power is a bottleneck for analytical query workloads since the CPU is engaged in both user and system operations. Analytical queries also have a short I/O time. on general, the Spark cluster on the cloud uses less resources than a regular Spark cluster. In addition, the system load on a traditional Spark cluster is higher than it is on Spark as a service.

Table1.SparkClusterBenchmarkHiBenchRunTimeParameters

ISSN PRINT 2319 1775 Online 2320 7876
Research paper Volume 10, Iss 1, 2021

Properties	Values
SparkVersion	Spark1.5.2builtforHadoop2.4.0
NoofNodes	4(1Masterand3Slaves/Workers)
HadoopVersion	Hadoop-2.4.0
HiBenchVersion	HiBench-5.0
Benchmarks	WordCount,TeraSort,Sleep,Sort,Bayesian,Aggregate,Join,PageRank,Scan
HadoopReplicationFactor	2
NumberofExecutor	2
ExecutorMemory	2GB
DriverMemory	1GB

Table2.SparkHiBenchExecutionResultsonConventionalCluster

Benchmarks	InputDataSize(bytes)	Duration(s)	Throughput(bytes/s)	Throughput/node
Aggregate	37276833	128.082	291038	97012
Bayesian	450822628	231.789	1944969	648323
Join	192861180	145.934	1321564	440521
PageRank	259928115	576.603	450792	150264
Scan	183579314	122.945	1493182	497727
Sleep	0	329.042	0	0
Sort	328490396	88.918	3694307	1231435
TeraSort	3200000000	461.694	6930997	2310332
WordCount	2204456452	142.487	15471281	5157093

Table3.SparkHiBenchExecutionResultsonCloud

Benchmarks	InputDataSize(bytes)	Duration(s)	Throughput(bytes/s)	Throughput/node	Speedup
Aggregate	37276833	90.990	409680	136560	1.4076
Bayesian	375706036	99.165	3788695	1262898	2.3374
Join	194078124	101.190	1917957	639319	1.4421
PageRank	259928115	346.854	749387	249795	1.6623
Scan	184796438	93.425	1978019	659339	1.3159
Sleep	0	301.69	0	0	1.09

ISSN PRINT 2319 1775 Online 2320 7876
Research paper Volume 10, Iss 1, 2021

		6			06
Sort	328490787	53.018	6195835	2065278	1.67 71
TeraSort	3200000000	136.43 1	23455079	7818359	3.38 40
WordCount	2204457594	68.753	32063438	1068781 2	2.07 24

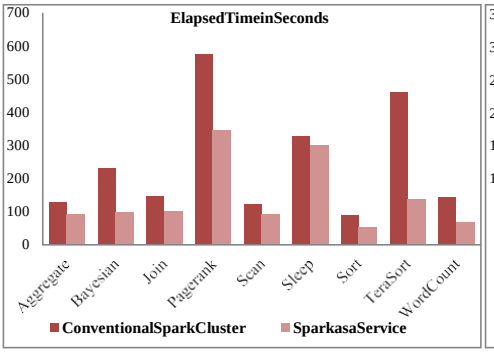


Fig.3.SparkClusterBenchmarksExecutionTime

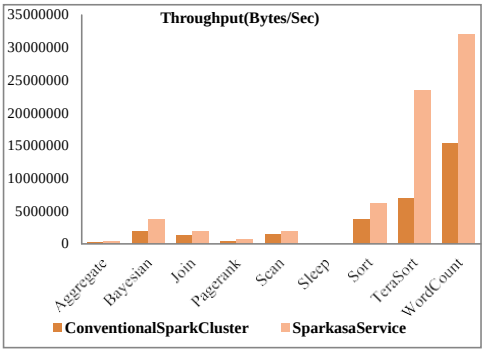


Fig.4. SparkClusterBenchmarksThroughput

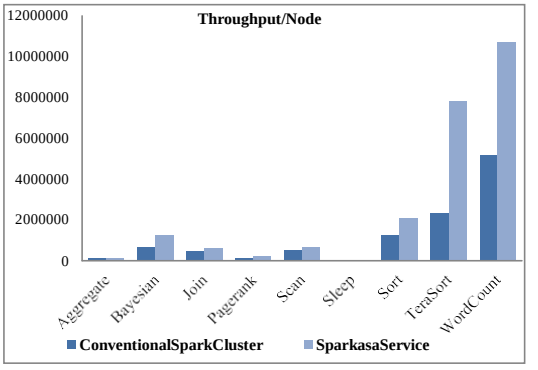
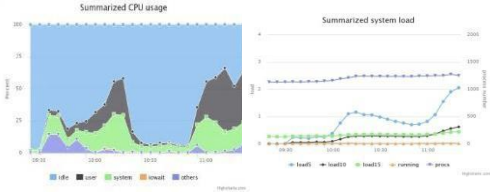
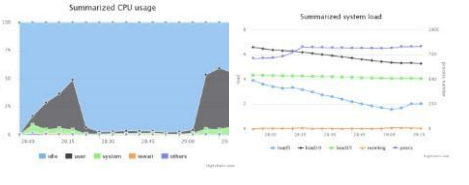
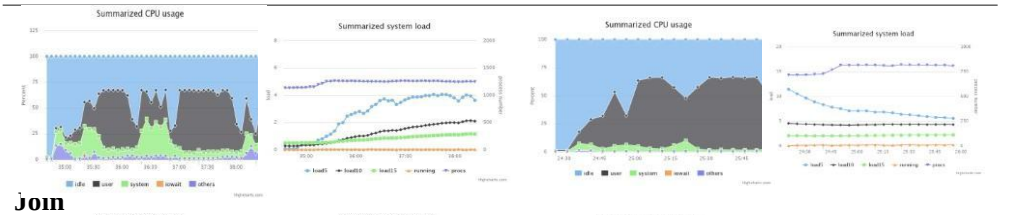


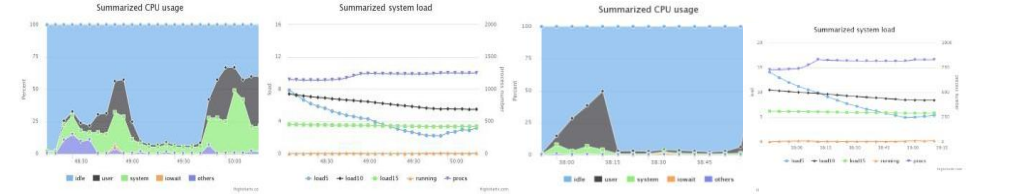
Fig. 5. Spark Cluster Benchmarks Throughput/NodeTable4.SummarizedCPUusageandsystemloadforvariousHiBenchBenchmarks

	SparkConventionalCluster	**SparkClusterasaService**
Aggregate		
Bayesian		

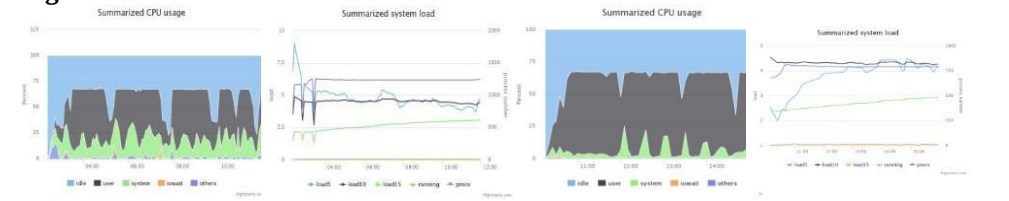
ISSN PRINT 2319 1775 Online 2320 7876
Research paper Volume 10, Iss 1, 2021



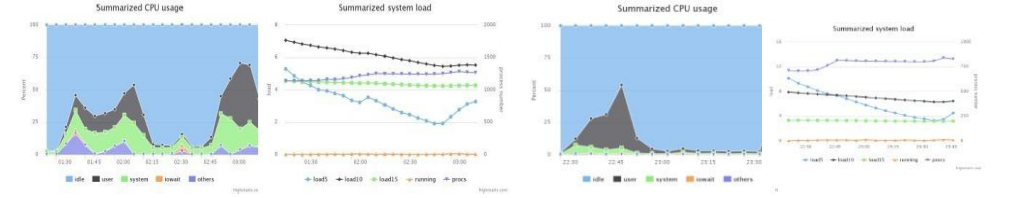
Join



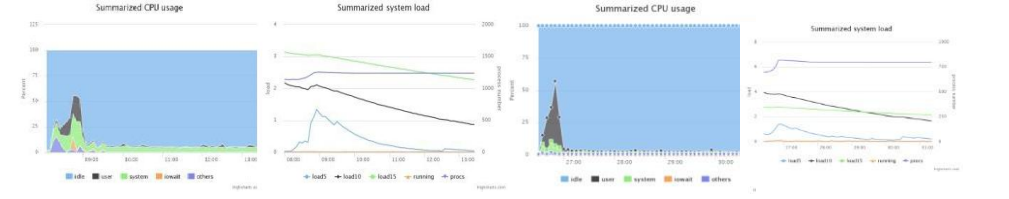
PageRank



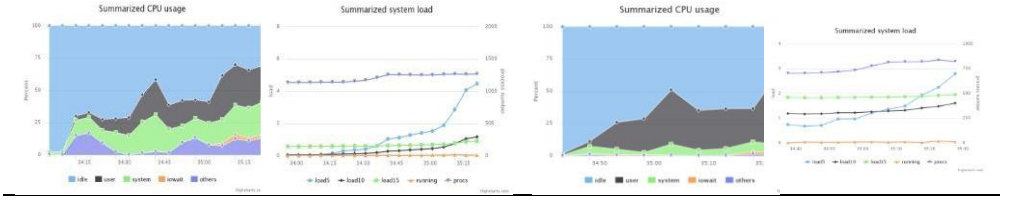
Scan



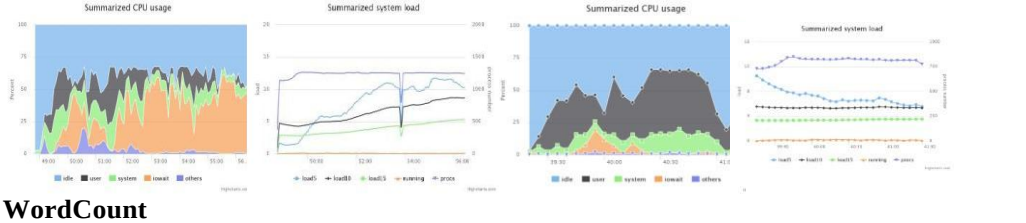
Sleep



Sort

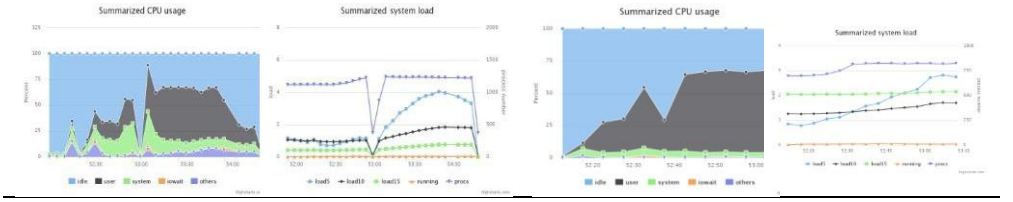


TeraSort



WordCount

ISSN PRINT 2319 1775 Online 2320 7876
Research paper Volume 10, Iss 1, 2021



6. Conclusions

This article makes it very evident how important it is to operate Spark cluster as a SAAS. By contrasting the results of running the HiBench test suite on a cloud and on-premises cluster, we can gauge Spark's efficacy as a SAAS. Spark as a SAAS performs better in terms of both speed and throughput, as shown by the final findings. When running the HiBench benchmark suite, which includes the four primary workload categories of "Micro benchmarks," "Analytical Query," "Web Search," and "Machine Learning," Spark as a SAAS exhibits decreased CPU consumption and system stress.

Acknowledgements

ThisresearchworkissupportedbytheHighPerformanceComputingCentre(HPCC),NEDUniversityofEngineeringand Technology,Pakistan.

References

1. One example is "Performance Optimization of Hadoop cluster using Linux Services," written by H. Ahmed, M. A. Ismail, and M. F. Hyder and published in the Proceedings of the 17th IEEE International Multi-Topic Conference (INMIC), Karachi, Pakistan,8-10 December, 2014, pages 167-172.
2. 2. "Parallel Programming Paradigms and Frameworks in the Big Data Era," by C. Dobre and F. Xhafa, published in International Journal of Parallel Programming, volume 42, issue 5, 2014, pages 710-738.
3. "A survey of open source tools for machine learning with big data in the Hadoop ecosystem," by S. Landset, T. M. Khoshgoftaar, A. N. Richter, and T. Hasanin, Journal of large Data, volume 2, issue 1, 2015, pages 1-36.
4. 4. "Apache Spark." (On the Internet). For use as of 2016-01-01, see http://Spark.apache.org/.
5. Article 5: "Comparing Apache Spark and Map Reduce with Performance Analysis using K-Means," by S. Gopalani and R. Arora, published in International Journal of Computer Applications, volume 113, issue 1 (2015).
6. Sixth Assunço, M.D., R.N. Calheiros, S. Bianchi, M.A. Netto, and R. Buyya, 2015. What the future holds for Big Data and cloud computing. 79:3-15 in Journal of Parallel and Distributed Computing.
7. Hashem, Ibrahim AbakerTargio, and others. "The rise of "big data" on cloud computing: review and open research issues." Pages 98-115 in Information Systems, 47 (2015).
8. Three more names: M. F. Hyder, M. A. Ismail, and H. Ahmed. The 10th IEEE International Conference on Emerging Technologies (ICET), held in Islamabad, Pakistan from December 8-10, 2014, included a paper titled "Performance comparison of Hadoop Clusters configured on virtual machines and as a cloud service," which was published in the conference proceedings.
9. Number Nine, the "HiBench Benchmark Suit." [Online]. Accessible as of the New Year at https://github.com/intel-hadoop/HiBench.
10. 10 Huang, Shengsheng, et al. "Hibench: A Representative and Comprehensive Hadoop Benchmark Suite." ICDE Workshop Proceedings, 2010.
11. Eleventhly, "Sahara Project." [Online]. Documentation for Sahara is at http://docs.openstack.org/developer/sahara/. [March. 3, 2016].
12. Twelve, "Spark Plugin for Sahara." [Online]. You may get this information at http://docs.openstack.org/developer/sahara/userdoc/spark_plugin.html. [March. 3, 2016].