Load Testing and Benchmarking for BigData

Aayush Agrawal, Sunil Raiyani, Jayam Modi

July 4, 2014

Outline

Aim of the Project

The aim of the project is Load Testing and Benchmarking for BigData.

The major task is to setup a distributed file system on a cluster, test the data and query processing capacity of the system using BigBench and predict the performance of the system for larger data sets.

Important Terms

1. **Load Testing** - It involves testing the system by steadily increasing the load on the system till it reaches its threshold limit
2. **BenchMarking** - It is the process of comparing the performance metrics of own systems with the industry standards.
3. **BigData** - It is a term that covers data sets so large and complex that it becomes impossible to process them using on-hand database management tools and traditional data processing applications.

Distributed Processing Tools Used

- **Hadoop**
  Apache Hadoop [1] provides a Distributed file system named HDFS which is a platform to store huge amounts of data divided into blocks across multiple hosts and a MapReduce engine which performs the processing of BigData

- **Hive**
  Apache Hive, as reported in [2] , is a data warehouse infrastructure built on top of hadoop for providing data analysis and querying features

- **Ganglia**
  Ganglia, as reported in [3] ,is a scalable distributed monitoring system for high-performance computing systems such as clusters and Grids

- **BigBench**
  [4] introduces BigBench as an industry standard benchmark for big data analytics. All the major characteristics in the lifecycle of a big data system are covered in BigBench which is an end-to-end benchmark.

Block Diagram

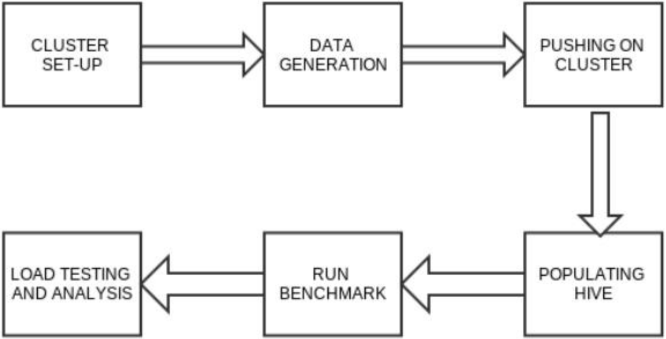

Figure : Average Query Response Times for different data sizes

Experimental Set-up

- **Namenode** : [Dual CPU] Intel Xeon E5-2620 v2 @ 2.10GHz server with 8 * 16384 MB @1600 MHz Samsung Synchronous DDR3 RAM and LSI MegaRAID SAS 9240-4i disk with 6 Gb/s SATA on each of 4 internal ports. The operating system is Ubuntu-Linux 12.04 Server.
- **Datanode** : Intel(R) Core(TM)2 CPU E7500 @ 2.93GHz commodity machine with 2048 MB @800MHz Synchronous DDR RAM and Seagate's 500GB 7200 RPM 3.5" Internal Hard Drive with 16MB Cache and 3 Gb/s SATA. The operating system is Ubuntu-Linux 14.04 Desktop.

The **network** connecting namenode and datanodes is a 100Mb/s Wired Ethernet.

Data Generation and Workload Customization

- PDGF generator at [5], is used to generate data.
- Modification of the workload to suit our experiment.
- Data is first generated on the local machine.
- Pushed data onto the cluster.
- Populated hive tables using this data.
- Chose only 9 queries out of 30 from [6] as most of them generated empty set results on the synthetically generated data.

Query Distribution in Customized Workload

The distribution of queries among different types of data is as follows:

Query-Type	Queries	Percentage
Declarative	3,6,7,8	44.4%
Mixed	2,5,9	33.4%
Procedural	1,4	22.2%

Data-Type	Queries	Percentage
Structured	3,6,7,8,9	55.5%
Semi-Structured	1,2	22.2%
Unstructured	4,5	22.3%

Load Testing

The following table lists the average query response time(in sec) for different data sizes(in GB) obtained experimentally:

Data Size	Query Response Time (sec)		
	2 DataNodes	3 DataNodes	4 DataNodes
1	293	188	172
5	358	317	275
10	1125	617	530
25	-	1154	1040
40	-	1451	1682
45	-	1793	1924
50	-	1956	2205
75	-	3054	3029
100	-	4491	4312

Load Testing

The figures 2 and 3 indicate the comparison between the average query response times for 3 datanodes and 4 datanodes.

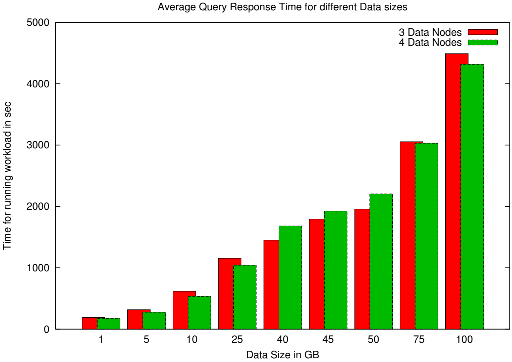

Figure : Average Query Response Times for different data sizes

Load Testing

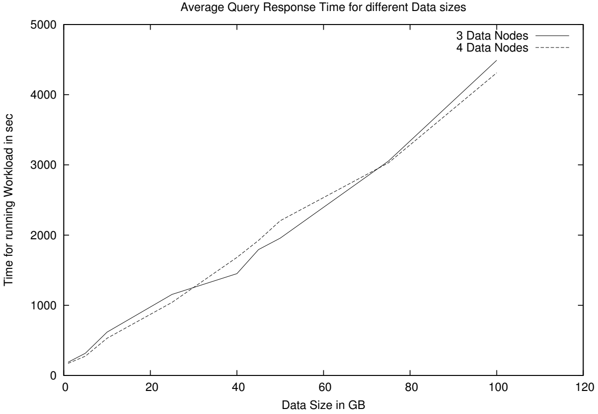

Figure : Average Query Response Times for different data sizes

Observations

- Initially, the average query response time using 4 datanodes was less than that using 3 datanodes.
- Gradually, as the data size increased, the time for 4 datanodes was more in comparison to that for 3 datanodes
- Furthermore, when the data size was increased further, the average response time behavior becomes same as it was initially.

Interpretation of Results

- The average query response time is affected by a combination of multiple factors such as:
  1. Division and distribution of work of queries across multiple nodes on the HDFS system.
  2. Time to transfer intermediate results over the network.
  3. Degree of swapping that takes place.
- Network transfer time is not significant for smaller intermediate results of small data size.
- For larger data size, swapping at each node and network transfer time dominates faster computation since intermediate results are also large. This brings the change of behaviour observed in the graph.

Interpretation of Results

- Data is distributed over more number of nodes and hence network transfer is also more.
- With further increase in data size, swapping becomes constant. So more distribution of workload dominates other factors.
- Computations become faster and behaviour of average response time becomes same as before.

Apart from these, there may be several other factors involved which affect the average response time of the system. Further investigation is required for analyzing this behavior.

Predictive Analysis

We have drawn a mean line which can be extrapolated to predict the scale factors for larger data sizes. The figures 4 and 5 indicate the scale factor for change in query response time w.r.t 1 GB of data.

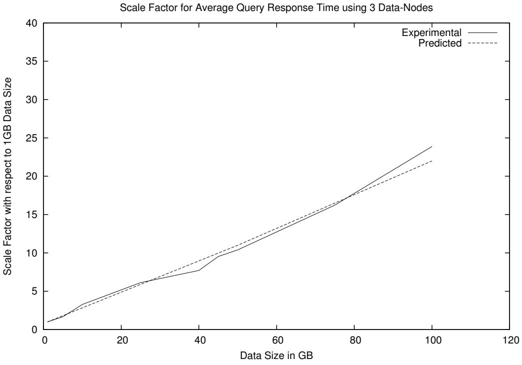

Figure : Scale factor for 3 datanodes

Predictive Analysis

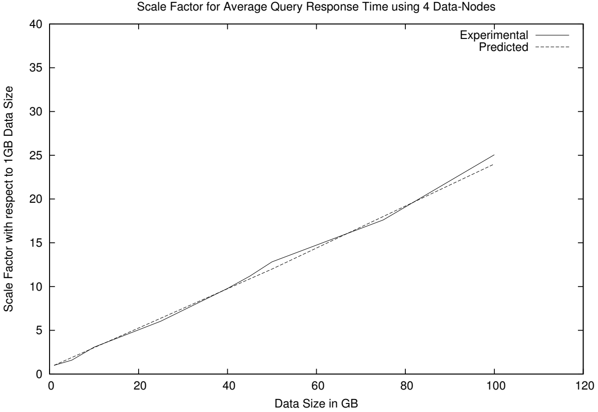

Figure : Scale factor for 4 datanodes

Predictive Analysis

Table : Error in Predicted time w.r.t Experimental time for 3 Datanodes

Data Size	Time (sec)		
	Experimental	Predicted	Deviation
1	188	188	0
5	317	338	21
10	617	526	91
25	1154	1090	64
40	1451	1654	103
45	1793	1842	49
50	1956	2030	74
75	3054	2970	84
100	4491	3910	481

Predictive Analysis

Table : Error in Predicted time w.r.t Experimental time for 4 Datanodes

Data Size	Time (sec)		
	Experimental	Predicted	Deviation
1	172	170	2
5	275	321	46
10	530	510	20
25	1040	1078	38
40	1682	1646	32
45	1924	1835	79
50	2205	2024	181
75	3029	2970	59
100	4312	3916	396

Initial Tasks

The following tasks have been completed:

- Study of the TPC-H and TPC-C benchmarks.
- Study of the research papers. [7] [8]
- Hadoop and Hive installation.
- Working with BigBench.
- Load testing experiments.

Future Scope

The following work is considered for future:

- Run the experiment for clusters of larger size and try to determine the optimum size of cluster of the given configuration of nodes to manage a particular data-set size.
- Analyze other parameters affecting the average query response time of the system.
- Open Source Release of our customized version of BigBench.

References I

[1] "Hadoop Architecture. Available at http://docs.hortonworks.com. Accessed on 30th June 2014," November 2011.

[2] A. Thusoo, J. S. Sarma, N. Jain, Z. Shao, P. Chakka, S. Anthony, H. Liu, P. Wyckoff, and R. Murthy, "Hive - A Warehousing Solution Over a Map-Reduce Framework," *Proceedings of the VLDB Endowment*, pp. 1626–1629, August 2009.

[3] "Ganglia Installation. Available at http://www.slashroot.in/how-install-and-configure-ganglia-gmod-and-ganglia-gme
Accessed on 26th June 2014," March 2013.

References II

[4] A. Ghazal, T. Rabl, M. Hu, F. Raab, M. Poess, A. Crolotte, and H.-A. Jacobsen, "Bigbench: Towards an industry standard benchmark for big data analytics," in *Proceedings of the 2013 international conference on Management of data*, pp. 1197–1208, ACM, 2013.

[5] "BigBench Installation `https://github.com/intel-hadoop/Big-Bench/blob/master/README.md`. Accessed on July 4, 2014."

[6] T. Rabl, A. Ghazal, M. Hu, A. Crolotte, F. Raab, M. Poess, and H.-A. Jacobsen, "Bigbench specification v0. 1," in *Specifying Big Data Benchmarks*, pp. 164–201, Springer, 2014.

References III

[7] Y. Chen, S. Alspaugh, and R. Katz, "Interactive analytical processing in big data systems: A cross-industry study of mapreduce workloads," *Proceedings of the VLDB Endowment*, vol. 5, no. 12, pp. 1802–1813, 2012.

[8] Z. Ming, C. Luo, W. Gao, R. Han, Q. Yang, L. Wang, and J. Zhan, "Bdgs: A scalable big data generator suite in big data benchmarking," *arXiv preprint arXiv:1401.5465*, 2014.