ISSN 2349-4506
Impact Factor: 2.785

Global Journal of Engineering Science and Research Management

SCALABILITY AND FAULT TOLERANCE OF MAPREDUCE FOR SPATIAL DATA

Hari Singh*, Seema Bawa
* Computer Science & Engineering Department, Panipat Institute of Engineering and Technology, Panipat, Haryana, India

DOI: 10.5281/zenodo.61147

KEYWORDS: Hadoop, Spatial Data, Fault Tolerance, Scalability.

ABSTRACT
This paper discusses the processing of spatial data on MapReduce – Hadoop platform. The Hadoop is known for its efficiency, ease of implementation and fault tolerance. The earlier existing technologies for parallel processing, such as, Grid Computing also provided fault tolerance, but its implementation using the replica management is not easy as compared to the internal replica management in Hadoop. This paper implements a Hadoop-GIS cluster for a spatial dataset and a spatial query is implemented in the MapReduce. The experimental run demonstrates the effectiveness of Hadoop cluster in terms of the scalability and fault tolerance.

INTRODUCTION
Over the last several years, huge amount of spatial data has been generated all over the world as a result of advancement in the field of GIS and GIS applications. The advancement in computer networks, popularity of Internet, reduced costs of storage and processing, and the development of distributed applications has given a pace to the generation of data. The spatial data is complex and voluminous due to its characteristics. The huge volume of spatial data faced problem on the storage front. The complex and unstructured spatial data was difficult to process with traditional sequential programming methods and on traditional GIS software. However, as GIS data generated through a wide GIS sources was in different forms and formats, so interoperability was a big issue. Due to the interoperability issue, it was very difficult to use the spatial data generated from other GIS agencies.

An early breakthrough for this bottleneck, towards the effective use of spatial data, was provided by Open Geospatial Consortium (OGC) [12]. The OGC Web Services enabled the seamless integration of widely distributed heterogeneous geo-processing and location services on heterogeneous spatial data. A second advancement was provided by the Internet based GIS (Web-GIS). It provided interoperability between different GIS platforms and the GIS data to different users. However, the stateless nature of Web-GIS services could not provide collaborated environment for complex geospatial processing that required a chain of simple GIS services. Later on, the traditional sequential programming environment was also not efficient enough to process the huge and voluminous spatial data. It led to adaptation of parallel processing for efficient processing over a set of collaborated computer machines.

Integrating spatial databases with Grid Technology provides a solution to the data storage, for the huge volume of the spatial data being generated through development of the GIS technology, and heterogeneity of the spatial databases. It also provided a parallel task execution environment for efficient processing of spatial data. The stateless characteristic of Web-GIS was also solved with the integration of Open Grid Service Infrastructure (OGSI), Web Service Description Language (WSDL) and Extended Mark-up Language (XML) [10]. However, the state-full information of grid services was integrated with the web services. The Web Service Resource Framework (WSRF) provided a solution by keeping two aside [9]. With the evolution of web services, the service oriented architecture Open Grid Service Architecture along with the WSRF technology (OGSA)/WSRF became popular for grid computing [6, 7]. The development of various grid middleware for specific purposes, such as, for spatial data integration in grid, had happened in grid computing. The combination of GIS and Grid Computing, Grid-GIS [3] became a new research tendency. A number of OGSA/WSRF based Grid-GIS architectures had been proposed in the past. However, on the fault tolerance front, such as the single point of failure of the domain manager and the distributed nodes carrying the local data, researches are still going on. The fault tolerance in the grid with GT 4.0 is done externally using the Replica Management. It is implemented with replica management and catalog management.

ISSN 2349-4506
Impact Factor: 2.785
Global Journal of Engineering Science and Research Management

The emergence of the MapReduce technology had given a new dimension to the fault tolerance problem of the Grid-GIS. There is no need to externally configure anything for fault tolerance management, it is provided internally. One needs to only provide the number of data-block replicas to be created during execution. The MapReduce implementation Hadoop automatically provides fault tolerance of the Domain Manger (Namenode) through realizing a Secondary Domain Manager. If the main Domain Manager goes off due to any reason then the Seconadry takes its place and the cluster keeps on running. Similarly, it also provides fault tolerance for the spatially distributed data on data nodes by creating replicas of the data block chunks in data nodes. In this way, if one data node holding a spatial data chunk goes-off due to any reason then this data is retrieved from the other data nodes due to replica management policy.

HADOOP AND HADOOP-GIS

The MapReduce implementation Hadoop [1, 2] is proved excellent for fault tolerance and ease of parallelization. The MapReduce is favored over Hadoop for its fault-tolerant, scalable, and parallelization features [4, 5, 8]. The optimization of the Hadoop system is presented with the concepts of data locality and redundant execution. Data locality is used for reading and writing data to local disc and hence reducing the network overhead and bandwidth. Redundant executions are used for map and reduce tasks over slow machines, which cause bottlenecks and slow down the whole operation. The MapReduce framework consists of Map and Reduce components that work on key-value pair concept. The MapReduce abstracts all the details of parallel processing from users and the users get a very simplified framework for programming [13, 14, 15]. The MapReduce has become very popular for parallel processing of arbitrary data. It works on divide-and-conquer strategy and breaks a computation into sub-computations over a set of computers in a cluster that operate in parallel. Each smaller computation is handled separately and the result of computation is returned at a central point.

HADOOP-GIS ARCHITECTURE

The architecture of Hadoop-GIS is the integration of Hadoop and the GIS, as presented in Figure 1. The client executes a query, upon the distributed spatial data among the cluster data nodes, through distributed data processing module that initiates a Job Tracker. A set of Map and Reduce functions is used for carrying out this task. The distributed spatial data among the cluster nodes is managed by the distributed data storage module on the HDFS though the Spatial data Handler part of the Domain Manager. The Spatial Data Handler of the Domain Manager in consultation with the Spatial Data Handlers of the data nodes creates the HDFS. The Indexer is used for indexing the spatial data for organized storage of the spatial data that provides fast data retrieval. During execution an HDFS Log is created by the Domain Manager, which is helpful for creating a backup by the Secondary Domain Manager. The Secondary Domain Manager creates a replica of the Domain Manager through a bidirectional communication with the HDFS Log and the Distributed Data Storage module. The Certification Authority (CA) manages the authenticity of the users.

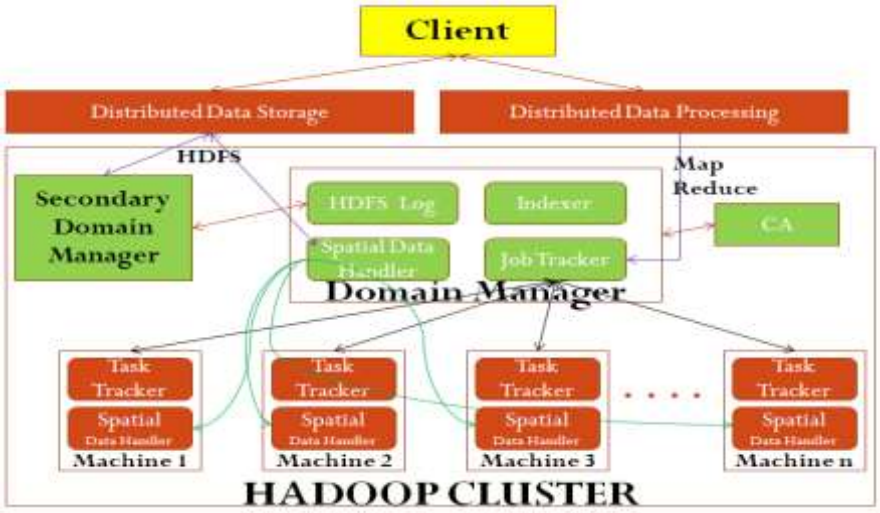

Figure 1. Architecture of Hadoop-GIS

The two main components of the Domain Manager, Spatial Data Handler and Job Tracker, use data nodes for creating the HDFS data storage and for parallel processing of sub-tasks. The data nodes are the computer nodes that become part of the Hadoop cluster and participate in providing their storage and the computational resources. These data nodes can be added either statically during cluster configuration initially or dynamically on the fly, when the cluster is running. The sub-tasks are the decompositions of the main task into smaller parts by the Job Tracker. These sub-tasks execute at the Task Tracker of the data nodes. The Task Trackers use the local storage of the data node as well as the remote storage of other data nodes through the HDFS.

FACTORS AFFECTING THE PERFORMANCE OF HADOOP CLUSTER

The following parameters affect the performance of Hadoop cluster. These parameters are a sort of calibrations for the Hadoop cluster. The proper calibration of the these parameters provide a better performance.

Partition Function - When the input dataset is put on Hadoop's HDFS, it is split into data chunks of the size of the data blocks on the data nodes with a default value is 64 MB. If data is split in such a way that the map computation in a data node does not find the data locally then more data transfers will occur and this will put extra burden on the network and also decrease the efficiency of computation. The performance can be increased if input data is partitioned in a particular way so as to maximize the data locality then movement of intermediate data generated as a result of Map computation over the network can be reduced and consecutively performance can be enhanced. Memory Size-As the namespace uses primary memory for building the metadata about the data over the HDFS file system. This metadata generally consists of information about the file chunks contained in data blocks on different slave nodes over the HDFS. So its size is determined by the size of input dataset. A shortage of RAM on name node crashes the job so generally the RAM on the name node is more than the RAM on slave nodes and approximately close to one-third of the total memory of the Hadoop cluster.

Block Size-The size of the input dataset and the input split affects the number of Map tasks. Setting the block size for the input dataset can configure the splits. If size of data blocks is kept small then CPU bursts are short and upon task completion significant data transfer takes place to collect the intermediate data for the reducers. Increasing the data block size and consecutively reducing the number of tasks can reduce this data transfer time. However, large data blocks increase the execution time per block and hamper the parallelism. The output of the Map task per data block affects the performance of Hadoop cluster if the former is directly proportional on the latter. In such a situation, large block size causes map-side spills.

The Number of Map and Reduce Tasks-The Number of Map and Reduce Tasks on the slave nodes is determined by the strength of the processor. However, when the processor strength allows many map tasks and many reduce tasks then the numbers must be chosen carefully. High Map/Reduce tasks means short CPU burst per Map/Reduce computations and low Map/Reduce tasks means large CPU burst per Map/Reduce computations. Number of Map/Reduce also relates to the data chunk in each data block and hence to the CPU execution time. If map computation is simple then CPU bursts are short but the startup overhead becomes significant as compared to the computation and upon task completion intermediate data transfer takes place for data shuffling to partition the data among the reducers. Larger CPU burst computations decrease the parallelism feature and hence reduces the performance.

Replicas-Providing a fault tolerance system with data replication has been the big characteristics of Hadoop cluster. However, it may degrade performance, so a limit must be kept over the number of replication over the data blocks. There are several reasons to it. Firstly, creating replicas over a large cluster becomes an expensive operation. Secondly, when replicas are created and put on different data blocks then data transfer in the form of replica takes place over the network and causes network resource consumption and puts load over the network. Thirdly, the generated replicas are put on the data blocks of the slave data nodes and hence consume local disk capacity.

ISSN 2349-4506
Impact Factor: 2.785
Global Journal of Engineering Science and Research Management

IMPLEMENTATION AND EXPERIMENTAL EVALUATION

The implementation of the proposed work was done on ten machines having the following configurations: Dual core processor 2.1 GHz, RAM 1 GB, 100 Mbps Ethernet, Ubuntu 11.04, Java-6-openjdk. The Hadoop cluster was set-up using the Hadoop 0.21.0. A Hadoop-GIS was set-up as per the architecture discussed in earlier section. The mentioned ten computers are taken from the two adjustment laboratories; each manages a start topology and interconnected in a LAN and connected via a switch/rack, as shown in Figure 2. The spatial dataset for the experimentation is taken from [11]. A sample of spatial dataset in Comma Separated Value (CSV) form with the schema is presented in Table 1.

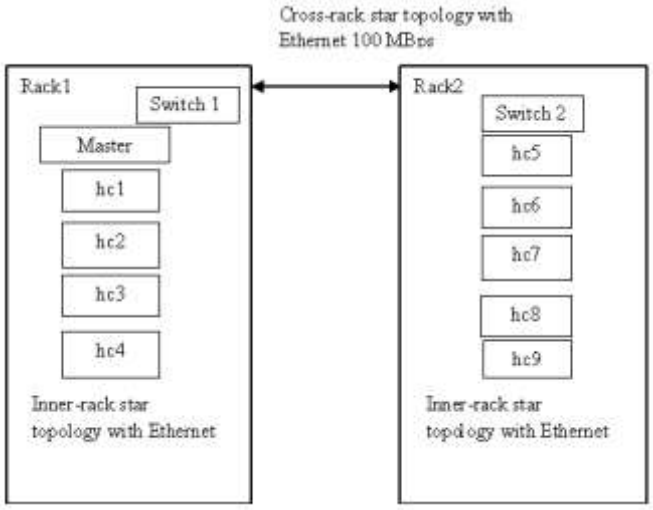

Figure 2. Topology of the Hadoop Cluster

A query is written for counting all the geographical names from the spatial dataset. The query is written in Java and run on the Grid-GIS. The following algorithm was used in MapReduce for querying over the distributed spatial dataset. The algorithm takes input in CSV file and the data is put on the HDFS. This task is accomplished in the MapReduce through the Map and Reduce functions. A number of Map function executes in parallel in the cluster and reads the partitioned spatial dataset records. Each Mapper stores one geograpnical name in a variable and passes it to the Reducer function. The Reducer function collects similar geographical names and computes the total for the entire spatial dataset.

Table 1. Sample data from basetable 040204.csv file [11]

Geographic ID Code	Geographic Name	Table Number	Table Order	Stub	Estimate	Lower Bound	Upper Bound
04000US01	Alabama	B01001	2	Male:	2122034	2113946	2130122
04000US01	Alabama	B01001	3	Under 5 years	147965	144298	151632
04000US01	Alabama	B01001	4	5 to 9 years	134647	123232	146062
04000US12	Florida	B24010H	19	Protective service occupations	99870	88342	111398
04000US12	Florida	B24010H	20	Food preparation and serving related occupations	117249	106366	128132

Algorithm for processing spatial data (.csv format) from [11]
Input: CSV data for which a sample data is shown in Table 1.
Output: Counted data on the basis of Geographic Name field of the Sample Database.

1. Create a job.
2. Input file(s) is/are read in FileInputOutputFormat line by line till the end of file is reached from HDFS.
3. A map function (Mapper) reads all the tokens from the file line by line in (key , value) pairs as (LongWritable, Text) format and also writes all the line tokens of the line to a variable itr of type StringTokenizer type.
4. A loop extracts the tokens from the line written to itr.
5. A conditional statement extracts only the tokens of second column.
6. Each token along with its count i.e. (1) is written to the temporary buffer in the form of (Key, Value) pair, which acts as input for the Reduce function (Reducer).
7. Reducer gets shuffled values from the mapper and through iteration generates cumulative sum for the occurrence of the key.
8. Reducer writes for each key the cumulative sum in (Key as Text, Value as IntWritable) in the temporary buffer.
9. This result is finally written to the HDFS.

RESULTS AND DISCUSSIONS

This section discusses the results obtained for the query run on the Hadoop-GIS for the spatial dataset.

Evaluation of configured capacity of the cluster with the addition of a new node

The storage of the HDFS in the Hadoop cluster is presented in Figure 3. An increase in the storage capacity ensures that the cluster is working fine and each node is contributing its configured capacity in the HDFS cluster. Each node is having 160 GB of hard disc space but approximately 145 GB is available after operating system and other essential software use the remaining memory. The disk capacity of each node is added to the cluster whenever a node becomes part of it.

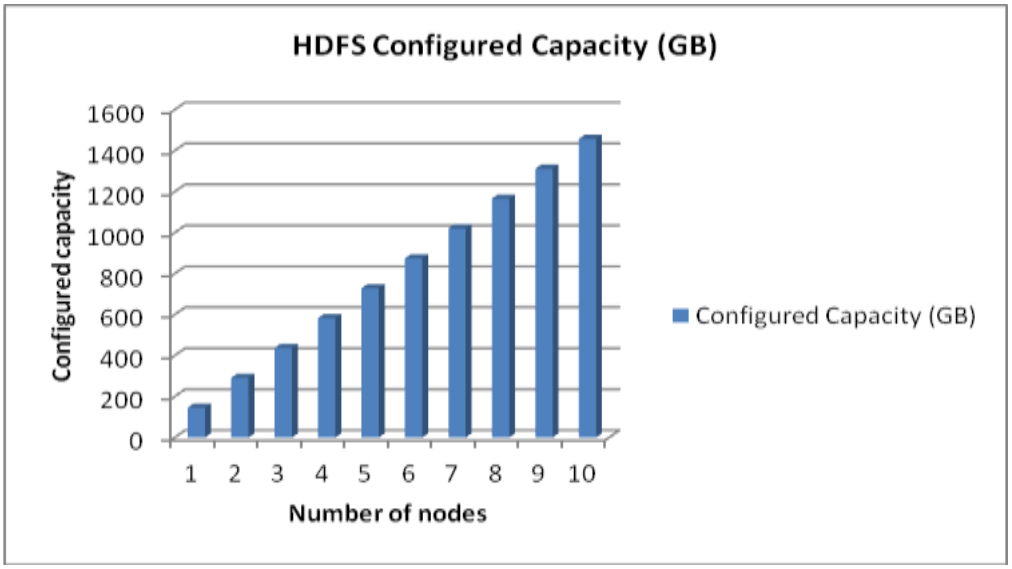

Figure 3. HDFS configured capacity vs. Hadoop cluster size

Effect of cluster size on execution time

The performance of Hadoop cluster increases in terms of the overall execution time, as the size of the cluster increases by adding nodes in the cluster. Similarly, the execution time of map phase and reduce phase also decreases separately with an increase in the size of the cluster as is evident from the Figure 5. A job is distributed to subtasks on task trackers and through parallelism all subtasks perform in parallel and minimize the overall execution time.

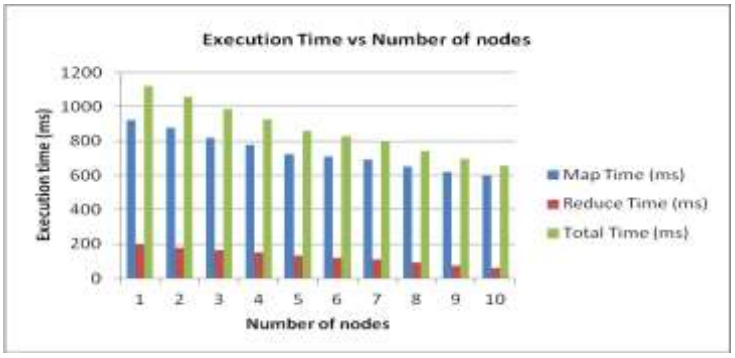

Figure 4. Execution Time vs. Number of nodes

Effect of HDFS Block size on Execution time

The client machine that puts the data file on the HDFS configures HDFS block size. The block size is also specified in the hdfs-site.xml configuration file of Hadoop cluster. Input splits for a data file is set through setting-up the block size and the split size also determines the number of Map and Reduce tasks. It is evident from Figure 5 that, for each configured block size (64 MB, 128 MB, 256 MB), the execution time for a job decreases with increase in the size of the Hadoop cluster. When the same number of nodes is kept fixed, the execution time for the job is at minimum and it is 128 MB block size. The execution time decreases when job is switched from a cluster with 64 MB block size to 128 MB block size. However, the execution time increases when the job is put on a cluster from 128 MB block size to 256 MB block size. It is due to the insufficient java heap memory available for the output mapped intermediate data that causes mapped-spill.

The effect of varying the number of data block replicas

The replica feature in Hadoop takes care of the fault tolerance by replicating data blocks among cluster nodes and improves the data availability. We set the number of replica (x) for a data block through the configuration file. It was found that when we set x=1 and run the task then sometimes the task completes successfully but for a few times the task failed to complete. When we set x=2 then the task runs successfully all the times baring a negligible number of times. When we set x=3, then the task takes abnormally longer time. And when we further increase x=4, the task does not complete and the machine/cluster is hanged. The reason for this kind of behavior is due to the number of replicas of data blocks that influences the execution. When insufficient numbers of replicas are in the cluster then sub-tasks fail as data blocks are not found due to locality problem or there are more failures during data transfer. In another case, when the numbers of data block replicas are large in number than the size of HDFS data becomes large and more data transfer takes place over the network that causes consumption of more network resources and puts load over the network and consequently slow down or sometimes even hang the whole system.

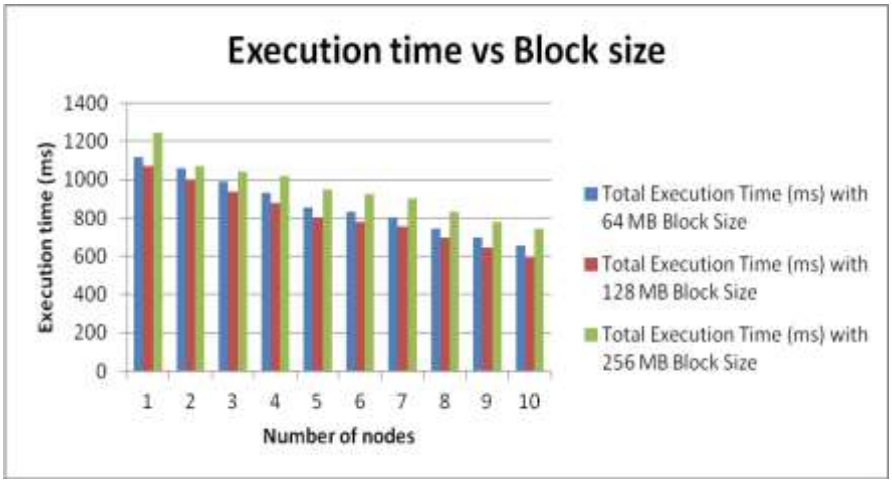

Figure 5. Execution time vs. Block size

CONCLUSION AND THE FUTURE SCOPE

This experimental work helps in understanding the significance of the cluster size and the data block size in a Hadoop cluster. When cluster grows in size, each node contributes with its data holding capacity of the disk and task processing capacity of the processor. A cluster processes a job by decomposing it into subtasks and executing these subtasks in parallel and consecutively reducing the execution time. The practical demonstration also elaborates the scalability and fault tolerance characteristic of Hadoop.

In future, we wish to work on the Spatial Indexer part of the Domain Manager in Hadoop-GIS. There are many traditional spatial indexing algorithms that had been designed and implemented for serial programming models. A few of these have high potential for parallelization. Exploring such spatial indexes in the MapReduce will be a challenging task.

REFERENCES

1. Applications powered by Hadoop: http://wiki.apache.org/Hadoop/poweredby
2. Hadoop. In http://Hadoop.apache.org.
3. L. Yu, "Study on the Architecture of GIS Grid," Geomatics and Information Science of Wuhan University, VOL. 29, 2004, pp. 153–156.
4. K. Lee, Y-J. Lee, H. Choi, Y.D. Chung and B. Moon "Parallel Data Processing with MapReduce: A Survey", ACM SIGMOD Record, VOL. 40, NO. 4, December 2011, pp. 11-20.
5. D. Jiang, B.C. Ooi, L. Shi and B. Wu, "The Performance of MapReduce: An In-depth Study", Journal, Proceedings of the VLDB Endowment, VOL. 3, NO. 1-2, 2010, pp. 472-483.
6. J. Frey and T. Tannenbaum, "Condor-G: A computation Management Agent for multi-Institutional Grids," Journal of Cluster Computing, VOL. 5, 2002, pp. 237-242.
7. Jinpeng Huai, Hailong Sun, Chunming Hu, Yanmin Zhu, Yunhao Liu, Jianxin Li," ROST: Remote and hot service deployment with trustworthiness in CROWN Grid", Future Generation Computer Systems, VOL.23 , NO. 6, 2007.
8. J. Dean and S. Ghemawat, "MapReduce: SimplifiedDataProcessing on Large Clusters", Magazine Communications of the ACM - 50th anniversary issue: 1958 – 2008, VOL. 51, NO. 1, 2008, pp. 107-113.
9. K. Czajkowski, D. Ferguson, I. Foster, et al., "The WS-Resource Framework Version 1.0," Tech. Rep. Available from http://www.globus.org/wsrf/specs/ws-wsrf.pdf, 2004.
10. I. Foster, J. Frey, S. Graham, S. Tuecke, K. Czajkowski, et al, "Modeling Stateful Resources with Web Services version 1.1," 2004.
11. www.census.gov.in.
12. Open Geospatial Consortium (OGC), http://www.opengeospatial.org/about, 2005.
13. Understanding Hadoop Clusters and the Network, http://bradhedlund.com/2011/09/10/understanding-hadoop-clusters-and-the-network/
14. D.A. Hager et al., "Hadoop Design, Architecture & MapReduce, Performance", http://www.datanubes.com/mediac/HadoopArchPerfDHT.pdf
15. D.A. Patterson, "Technical perspective: the data center is the computer", Communications of the ACM, VOL. 51, NO. 1, 2008, pp. 105-105.