Survey on Improved AutoScaling in Hadoop into Cloud Environments

Masoumeh Rezaei Jam
Department of Computer Engineering, Faculty of Electrical & Computer Engineering, University of Tabriz,
Tabriz, Iran
m_rezaeijam90@ms.tabrizu.ac.ir

Leyli Mohammad Khanli
Department of Computer Engineering,
Faculty of Electrical & Computer Engineering, University of Tabriz,
Tabriz, Iran
l-khanli@tabrizu.ac.ir

Mohammad Kazem Akbari
Computer Engineering and Information Technology
Amirkabir University of Technology (Tehran Polytechnic),
Tehran, IRAN
akbarif@aut.ac.ir

Elham Hormozi
Computer Engineering and Information Technology
Mazandaran University of Science and Technology,
Babol, IRAN
e.hormozi@ustmb.ac.ir

Morteza Sargolzaei Javan
Computer Engineering and Information Technology
Amirkabir University of Technology (Tehran Polytechnic),
Tehran, IRAN
msjavan@aut.ac.ir

Abstract—**Nowadays technologies for analyzing big data are evolving rapidly. Because of that models and methods to design and analyze parallel processing of data is done automatically. So MapReduce is one of these methods in order to overcome the complexity of very large data. MapReduce-based systems are suited for performing analysis at this scale since they were designed from the beginning to scale to thousands of nodes in a shared-nothing architecture. This model has been developed under a cloud computing platform. There are many implementations of MapReduce. One of them is the Apache Hadoop project that is an Apache's Open Source implementation of Google's MapReduce parallel processing framework. Running Hadoop on a cloud means that we have the facilities to add or remove computing power from the Hadoop cluster within minutes or even less by provisioning more machines or shutting down currently running ones. In this survey, we investigate some methods to improve scalability of Hadoop platform and Autoscaling of that. Based on the evaluation methods we understand that "The controller module and BEEMR" are best way to improve energy performance.**

Keywords- mapreduce; cloud computing; hadoop; auto scaling; dynamic resource allocation; energy efficiency.

I. INTRODUCTION

Cloud computing can be defined as a new mode of computing in which dynamically scalable and often virtualized resources are provided as a service over the internet. Cloud computing has become a significant technology trend, and lots of experts expect that cloud computing will reshape IT progresses and marketplace [3]. The benefits of cloud computing technology include cost savings, high availability, and easy scalability. In fact, anything executing inside a browser that aggregates and stores user-generated content now equalizes as an instance of cloud computing. This includes social-networking services such as Google Docs, Facebook, YouTube and Gmail. In this context, the cloud simply refers to the servers that power these sites, and user data is said to reside "in the cloud". The accumulation of vast quantities of user data creates large-data issues, many of which are suitable for MapReduce [4]. The computing resources in a cloud can be scaled up automatically to meet the seeking demands of users. Parallel programming paradigm similar to MapReduce is generally used for developing scalable applications deployable on the cloud, because of that cloud is a reasonable distributed system platform. Indeed cloud availability on the web to a big number of user's calls for its capability to scale vastly and supply intelligent service to the customers having changeable demands. This intelligence can permit to enhance its performance, scalability of cloud resources and give its users the cloud clients as well as back end operators the cloud vendors and partners, a superior skill using the computing model. So, efficient management of resources at cloud scale while providing proper performance isolation, higher consolidation and elastic use of underlying hardware resources is an important key to a successful cloud deployment [9].

The rest of this paper is structured as follows: Section II starts with introducing the Apache Hadoop Project. In Section III we present Energy Efficiency of Hadoop with A Covering subset of Nodes. Section IV explains about the Nephele Model for Efficient Parallel Data Processing in the Cloud. Section V presents Autoscaling hadoop with Control Panel. In Section VI we describe Dynamic Energy Efficient of Data with Controller Module and Section VII explains BEEMR for Dynamic Energy Efficient data. Section VIII presents GreenHDFS that is energy-conserving and highly scalable. Finally, we compare the mentioned methods in section IX.

II. APACHE HADOOP PROJECT

The Apache Hadoop project develops open-source software for reliable, scalable, distributed computing. This project software library is a framework that allows for the distributed

processing of big data sets across clusters of computers using plain programming models. Scalability is one of the original forces driving reputation and adoption of the hadoop project. This project is designed to scale up from single servers to thousands of machines, each offering local computation and storage. The library itself is designed to detect and handle failures at the application layer, rather than rely on hardware to deliver high-availability, so delivering a highly-available service on the upside of a cluster of computers, each of which may be prone to failures [5].

A. Hadoop MapReduce Paradigm

Hadoop MapReduce is an *Apache Software Foundation* software framework for reliable, scalable, distributed parallel processing of big data sets across multiple hosts. The Hadoop core framework includes a shared file system (HDFS), a set of usual productivities to backing distributed processing, and an implementation of the MapReduce programming model. On the other hand, MapReduce is a programming model and software framework introduced by Google to support distributed computing on large data sets on clusters of computers. In other words, to handle large-scale web search applications, it was essentially proposed by Google. Hadoop MapReduce is an Apache open source project that develops parallel applications without any parallel programming techniques. The applications could be easy deployed and executed. Hadoop processes huge datasets and works with the HDFS [1] on distributed clusters in parallel. HDFS is written in Java for the Hadoop framework and it is a distributed, scalable, and portable file system. Data in a Hadoop cluster is broken down into smaller called blocks. MapReduce works by breaking the processing into two phases: the map phase and the reduce phase (Fig. 1). Each phase has key-value pairs as input and output, the types of which may be chosen by the programmer. The programmer also specifies two functions: the map function and the reduce function [6].

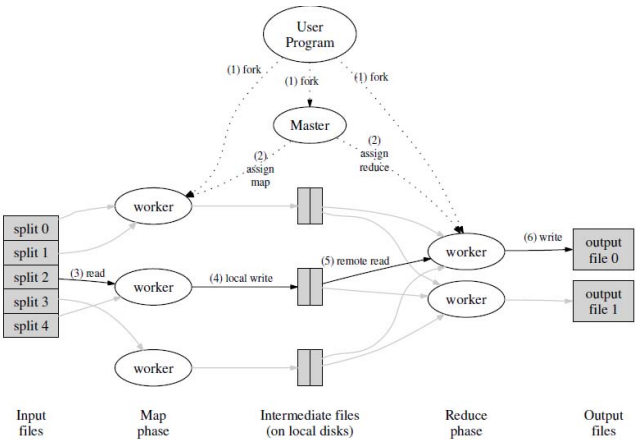

Figure 1. MapReduce Framework [1]

[1] **H**adoop **D**istributed **F**ile **S**ystem

III. ENERGY EFFICIENCY OF HADOOP CLUSTERS WITH A COVERING SUBSET OF NODES

Nowadays energy consumption and cooling are large components of the operational cost of datacenters and dash serious limitations in terms of reliability and scalability. Jacob Leverich and et al. in [7] presents work on shifting hadoop to permit scale-down of usable clusters. While Hadoop's self-determining re-replication trait can, allow further nodes to be disabled over time, this comes with severe storage volume and resource penalties. They offer a new permanent for use during block replication to address short-fall in Hadoop's data-layout tactic: at least one replica of a data-block must be stored in a subset of nodes that they refer to as a covering subset. The premise behind a covering subset is that it contains an adequate set of nodes to warrant the prompt availability of data, even were all nodes not in the covering subset to be disabled [7]. The purpose in establishing a covering subset and utilizing this storage stable is so that large numbers of nodes can be elegantly removed from a cluster without affecting the availability of data or interrupting the normal operation of the cluster; thus, it should be an infancy portion of the cluster. Just as replication factors can be specified by users on a file by file fundament, covering subsets should be established and specified for files by users. In large clusters, this permits covering subsets to be intelligently managed as current activity dictates, rather than as an agreement between several potentially active users or applications. Thus, if a particular user or application vacates a cluster for some period of time, without affecting the availability of resident users and applications, the nodes of its associated covering subset can be turned off. To study the impact of disabling nodes on fractional utilization workloads, Jacob Leverich and et al. in [7] performed tests with batches of hadoop jobs. They depict the distribution of system inertia periods of the job trace when 18 nodes are disabled, and the distribution when all nodes are energetic is overlaid for collation. Disabling 18 nodes significantly enhancement the length of time spent idle for more than 82 seconds and moderately growth the length of time spent idle for more than 10 seconds.

IV. EFFICIENT PARALLEL DATA PROCESSING IN THE CLOUD WITH NEPHELE MODEL

D. Warneke and et al. in [8] have presented Nephele and also presented the disadvantages and advantages for impressive parallel data processing in cloud environments, the first data processing framework to draw out the dynamic resource provisioning offered by current IaaS clouds for task scheduling and execution (Fig. 2). During the job execution, particular tasks of a processing job can be assigned to variant kinds of virtual machines which are automatically instantiated and terminated. Based on this framework, they perform evaluations on a compute cloud system and compare the outcomes to the existing data processing framework hadoop. The performance evaluation gives a first impression on how the ability to assign specific virtual machine kinds to specific tasks of a processing job, as well as the possibility to automatically allocate/deallocate virtual machines in the course of a job execution, can help to reduce the processing cost and improve the overall resource utilization. Their experiments show the

average instance utilization over time. Also, Nephele's savings in time and cost might appear edgy at first glance. However, these savings happen by the size of the input data set. More complex processing jobs become feasible for larger data sets, for larger data set which also promises more principle savings [10].

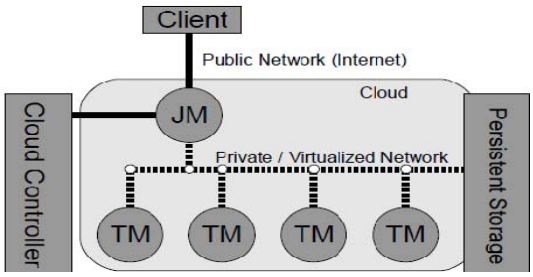

Figure 2. Structural overview of Nephele running inside a compute cloud [10]

V. AUTOSCALING HADOOP CLUSTERS WITH CONTROL PANEL

Running Apache Hadoop on a cloud platform means that we have the facilities to add or remove computing power from the hadoop cluster within minutes or even less by provisioning more machines or shutting down currently running ones. This permits the elasticity to have only the needed hardware for the job and not to waste CPU cycles on idling. Based on some rules and without any human intervention, Toomas Romer and et al. in [2] prototype a system where this scaling is done automatically. Their purpose is to build a dynamic MapReduce cluster using the hadoop implementation. The cluster will scale itself based on the load of the cluster. The cluster can be executed both in a public and a private cloud. They also monitor and debug the cluster in real-time to realize good autoscaling tactic or to adapt it. The system consists of control panel or a dashboard to monitor a hadoop cluster in real-time and debug for possible issues, virtual machine images for the hadoop nodes that the images are preconfigured Linux OS that automatically start hadoop software on boot and the nodes will automatically connect a hadoop cluster and report performance metrics to the control panel, identifying and collecting system metrics of the running nodes to the control panel to make assessments of the load and health of the cluster, providing an autoscaling tactic to scale the cluster horizontally based on the monitored performance metrics of the cluster. They chose Amazon Web Services (AWS) as public cloud provider and the eucalyptus private cloud software executing at the scientific computing on the cloud at the University of Tartu as their private cloud provider. The nodes of this cluster send heartbeats to the control panel every minute. This aids to determine the size of the cluster and log and update the IP address of the master node. With the IP of the machine they also log several performance metrics of each node every 30 seconds via custom Python script into a database. Also hadoop will divide the computations for a larger set of nodes and will make sure that all of them are fully utilized even, when the size of the cluster is increased [2].

VI. DYNAMIC ENERGY EFFICIENT DATA WITH CONTROLLER MODULE

MapReduce framework incorporates mechanisms to certify reliability, load balancing, fault tolerance, etc. But such mechanisms can have an impact on energy efficiency as even idle machines remain powered on to ensure data availability. In [11] the server and energy efficiency report states that more than 15% of the servers are run without being used actively on a daily fundament. Given the scale at which these applications are deployed, minimizing power consumption of these clusters can significantly reduce their carbon footprint and grind operational costs. Nitesh Maheshwari and et al. in [12] address energy conservation for clusters of nodes that execute MapReduce jobs. Based on the current workload the algorithm seeking reconfigures the cluster and turns cluster nodes on or off when the average cluster utilization rises above or falls below administrator specified thresholds, respectively. As shown in Fig. 3, the algorithm investigates if there is sufficient space available on HDFS to store the file and turns on more nodes if needed. At all times, they keep some space reserved on the DataNodes, so that write operations do not block or fail waiting for more space to be provisioned while new DataNodes are being powered on. The cluster is rebalanced as there is no data on the newly activated DataNodes while other DataNodes are already filled substantially, after the nodes are powered on. They use the GridSim toolkit for evaluate the algorithm and results demonstrate that algorithm achieves an energy reduction of 33% under average workloads and up to 54% under low workloads.

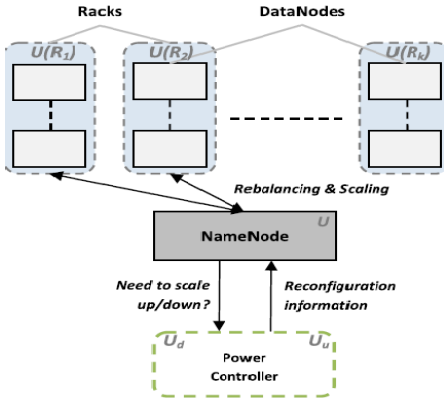

Figure 3. Interaction of power controller module with the NameNode [12].

The techniques employed in these works compromise on the replication factor of data blocks stored on the cluster. Algorithm described in [12] without compromising on the replication factor for the data blocks, attempts to keep the cluster utilized to its maximum permitted potential and accordingly scale the number of nodes. However, the efficiency of this method would depend on how quickly a node can be activated.

VII. DYNAMIC ENERGY EFFICIENT DATA WITH BEEMR

Approaches to increasing datacenter energy efficiency depend on the workload in question. One option is to raise machine utilization. This method is favored by vast web search

companies whose machines have persistently low utilization and waste considerable energy such as Google [13]. Y. Chen and et al. in [14] focus on a case that called MIA[2]. MIA workloads include traditional batch processing, interactive services, large-scale and latency sensitive processing. Big fractions of the overall cost of ownership of datacenters are energy costs [15]. MIA workloads need a method to energy-efficiency, which focuses on decreasing the volume of energy used to service the workload. Y. Chen and et al. present BEEMR[3], an energy efficient MapReduce system that is similar to an usual Hadoop MapReduce cluster, with main differences in how resources are allocated to jobs. Also it is motivated by an empirical analysis of a real-life MIA workload at Facebook. The BEEMR cluster architecture is shown in Fig. 4. The workload manager of BEEMR (job tracker) classifies each job into one of three classes which determines which cluster region will service the job. This workload to meet stringent design requirements, such as minimal impact on capacity, interactive job latency, write bandwidth, memory set size, and data locality needs BEEMR, as well as compatibility with distributed file system fault tolerance using error correction codes rather than replication. In the batch region, batchable and interruptible jobs are serviced, while in the interactive region, interactive jobs are serviced. Energy savings come from summation jobs in the batch region to accede high utilization, performing them in regular batches, and then when the batch completes transitioning machines in the batch region to a low-power state.

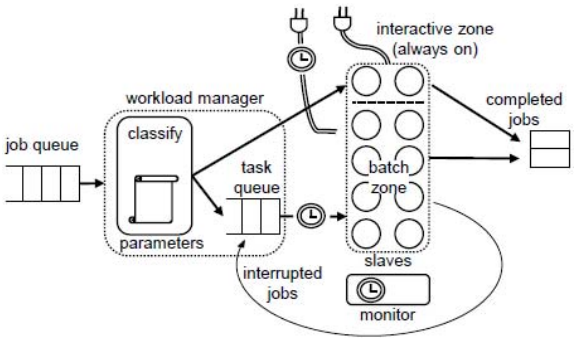

Figure 4. BEEMR Architecture [14]

Therefore under tight design constraints, BEEMR achieves 40-50% energy savings, and represents improving energy efficiency for an increasingly major class of datacenter workloads [12, 14].

VIII. GREENHDFS

R. Kaushik and et al. in [16] present an energy-conserving, highly scalable variant of the HDFS called GreenHDFS. In other word, GreenHDFS is an energy-conserving, self-adaptive, hybrid, logical multizoned variant of HDFS. Instead of an energy-efficient placement of computations or using a small covering set for primary replicas as done in earlier research, Green HDFS [7] partitions HDFS into disjoint hot and cold regions. Hot zone is always powered and the frequently accessed data is placed in hot zone. Green HDFS fills the cold zone using one powered on machine at a time, to preserve write capacity. Because of the output of every job would be located on a small number of machines and creating a severe data hotspot for future accesses this scheme is problematic. Furthermore, running the cluster at partial capacity decreases the available write bandwidth and memory. In other words, GreenHDFS relies on insightful data-classification driven energy-conserving data placement to realize guaranteed, substantially long periods of idleness in a significant subset of servers in the cold zone. Cold zone consist of files with low to rare accesses, hot zone consist of files that are being accessed currently and the newly created files. The aim of GreenHDFS is to minimize the number of servers allocated to the cold zone and maximize the allocation of the servers to the hot zone to minimize the performance impact of zoning and. Fig. 5 shows a 24% reduction in energy consumption of a 1560 server datacenter with 80% capacity utilization [16].

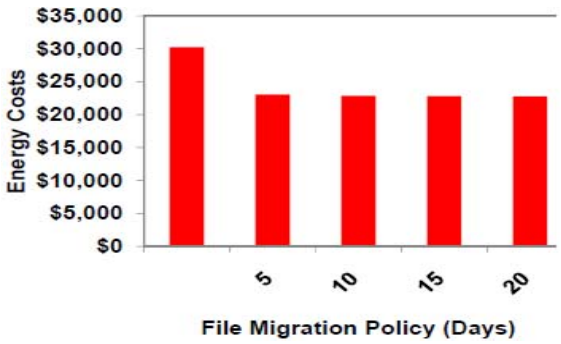

Figure 5. Energy Savings with GreenHDFS [16]

Evaluation results in [16] show that GreenHDFS in spite of the unique scale-down challenges present in a hadoop cluster, is able to meet all the scale-down mandates.

IX. COMPARISON AND EVALUATION OF METHODS

In Table 1 we present comparison of the methods described in above sections. As you see energy efficiency of hadoop cluster with a covering subset of nodes provides saving energy, improving utilization and CPU efficiency [7]. Efficient parallel data processing on cloud with Nephele model in addition to three previous supports network efficiency and instance efficiency [8, 10]. Hadoop makes better adjustments to the size of the cluster to facilitate the increasing or decreasing demand because it has many options to predict the work load of a cluster. As was mentioned, autoscaling hadoop cluster with control panel supplies memory efficiency, CPU efficiency and instance efficiency but some metrics such as memory usage, jobtracker and tasktracker counters has not been considered [2].

The efficacy of the method used in [12] would depend on how rapidly a node can be activated. Also dynamic energy efficient data with controller module covers saving energy, improving utilization and instance efficiency [12].

In [7] one replica of every block within a small subset of machines called the covering subset maintain by the covering

[2] **M**apReduce with **I**nteractive **A**nalysis

[3] **B**erkeley **E**nergy **E**fficient **M**ap**R**educe

subset scheme. This subset while the rest is powered down remains fully powered to preserve data availability. In fact, a fraction of the cluster decreases write capacity, write bandwidth and the size of current memory. Also, when error correction codes are used instead of replication, this scheme becomes unusable since the covering subset becomes the total cluster [14].

Even under BEEMR [14], long jobs with low levels of parallelism stay a challenge. If job's task durations are large, and batch jobs otherwise, these jobs are classified as interruptible jobs. They could prevent batches from completing, if such jobs are classified as batch jobs. Their inherent low levels of parallelism cause the batch zone to be weekly utilized when running only these long jobs, resulting in wasted energy [14].

Finally in [16] HDFS divides into disjoint hot and cold regions by Green HDFS. Green HDFS to preserve write capacity fills the cold zone using one powered-on machine at a time and the frequently achieved data is placed in the hot zone, which is always powered. Consequently, the output of every job would be located on a small number of machines, creating a severe data hotspot for future accesses. Indeed, executing the cluster at partial capacity decreases the available write memory and bandwidth [14]. So, [16] simulate results with real world Yahoo! HDFS traces show that GreenHDFS can achieve 24% energy cost reduction.

TABLE I. COMPARISON OF METHODS

Criterion / Method	Year	Proposal	Saving Energy	Improving cluster Utilization	Memory Utilization	CPU Utilization	Network Utilization	Instance Utilization	Prediction
[7]	2009	Covering Subset of Nodes	✓	✓	×	✓	×	×	×
[8]	2011	Nephele Model	×	✓	✓	✓	✓	✓	×
[2]	2010	Control Panel	×	×	✓	✓	✓	×	×
[12]	2011	Controller Module	✓	✓	×	×	×	✓	×
[14]	2012	BEEMR	✓	✓	✓	×	×	✓	Calculate the task execution time
[16]	2012	GreenHDFS	✓	×	✓	✓	×	×	×

X. CONCLUSION

We investigated some methods for autoscaling of hadoop. It should be noted, mechanisms may facilitate autoscaling of hadoop, but they may impose some of requirements on the processing framework design and ways that tasks are defined, too. To make better decisions, scheduler of such a framework must consider costs while add and remove resources. Whilst it should know about the several types of available virtual machines plus their costs and also it can allocate or release them behalf of the client to the cloud. As was observed in any of the methods listed, so had tried to improve energy efficiency in hadoop. Therefore according to estimates in the literature [14] and [12] that also listed in the evaluation methods table "The controller module" and "BEEMR" manners are better in improving energy efficiency. As future work, we decided on a new way of ideas and try to use articles [14] and [2] to improve energy consumption and autoscaling of hadoop.

XI. ACKNOWLEDGMEN

The authors would like to thank all those who contributed to this paper. Further to this, we gratefully acknowledge those in the cloud computing team at the Department of Computer engineering and Information Technology, Amirkabir University, IRAN and Cloud Computing lab in University of Tabriz, IRAN.

REFERENCES

[1] J. DEAN, S. GHEMAWAT, "Mapreduce: simplified data processing on large clusters," In OSDI'04: Proceedings of the 6th conference on Symposium on Opearting Systems Design & Implementation, USENIX Association, Berkeley, CA, USA, 2004.

[2] T. Römer, "Autoscaling Hadoop Clusters," MSc thesis, University of Tartu, 2010.

[3] B. Furht, A. Escalante, "Handbook of Cloud Computing," Department of Computer and Electrical Engineering and Computer Science Florida Atlantic University, Springer, USA, 2010.

[4] J. Lin and C. Dyer, "Data-Intensive Text Processing with MapReduce," Morgan and Claypool, 2010.

[5] http://hadoop.apache.org/#What+Is+Apache+Hadoop%3F, this page was last modified on 12/05/2012 18:24:07.

[6] E. Hormozi, M. K. Akberi, M. S. Javan, H. Hormozi, "Performance Evaluation of Fraud Detection based Artificial Immune System on the Cloud," In procciding of the 8th International Conference on Computer Science & Education, Colombo, Seri Lanka IEEE, 2011.

[7] J. Leverich, C. Kozyrakis, "On the energy (in)efficiency of hadoop clusters," In *HotPower*, 2009.

[8] D. Warneke, O. Kao, "Nephele: Efficient Parallel Data Processing in the Cloud," In I. Raicu, I. T. Foster, and Y. Zhao, editors, *SC-MTAGS*. ACM, 2009.

[9] E. Hormozi, H. Hormozi, M. K. Akbari, M. S. Javan, "Using of Machine Learning into Cloud Environment (A Survey) Managing and Scheduling of Resources in Cloud Systems," In procciding of the 7th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing IEEE, 2012.

[10] D. Warneke, O. Kao, "Exploiting dynamic resource allocation for efficient parallel data processing in the Cloud," Parallel and Distributed Systems, IEEE Transactions on 22 (6) (2011) 985 –997, 2011.

[11] Server energy and efficiency report, Tech. Rep., 1E, 2009.

[12] N. Maheshwari, R. Nanduri, V. Varma, "Dynamic energy efficient data placement and cluster reconfiguration algorithm for MapReduce framework," Future Generation Computer Systems, vol. 28, no. 1, pp. 119–127, 2012.

[13] L. A. Barroso, "Warehouse-Scale Computing Entering the Teenage Decade," FCRC plenary talk, 2011.

[14] Y. Chen, S. Alspaugh, D. Borthakur, R. Katz, "Energy Efficiency for Large-Scale MapReduce Workloads with Significant Interactive Analysis," Proceedings of the 7th ACM european conference on Computer System, pp. 43-56, 2012.

[15] C. Belady. In the data center, power and cooling costs more than the IT equipment it supports. Electronics Cooling Magazine, Feb. 2007.

[16] R. Kaushik, M. Bhandarkar, K. Nahrsted, "Evaluation and Analysis of GreenHDFS: A Self-Adaptive, Energy-Conserving Variant of the Hadoop Distributed File System," IEEE, 2012.