

# Energy Efficient and Reliable Job Submission in Hadoop Clusters

DG Sudha Sadasivam,  
Professor, CSE,  
PSG College of Technology,  
Coimbatore - 641 004.

S Sangeetha,  
Lecturer IT,  
PSG Polytechnic College,  
Coimbatore - 641 004.

R Radhakrishnan,  
Student, CSE,  
PSG College of Technology,  
Coimbatore - 641 004.

## ABSTRACT

MapReduce paradigm is highly suitable for large scale data intensive applications in the cloud environment. The scale of these applications necessitates minimization of cluster power consumption to reduce operational costs and carbon footprint. Energy consumption can be reduced by selective power down of nodes during periods of low utilization. Hadoop is basically used for batch processing of huge jobs. Before jobs are submitted, the files used them are uploaded into the cluster. A file is split up into a number of chunks and distributed across the Hadoop cluster. This paper addresses the problem of block allocation in distributed file system to improve reliability and energy efficiency. A framework to reduce power requirements of a cluster by identifying the number of replicas and their placement for reliable completion of the job has been designed. This will address the issues like block allocation, reliable job submission and minimization of cluster nodes to reduce power consumption. This framework is integrated with Hadoop's namenode. The scheduler component in Hadoop has also been modified to enable submission of jobs to active data node containing data to be operated on. A greedy approach and an evolutionary approach using Particle Swarm Optimization (PSO) has been designed to identify suitable nodes to be activated in a cluster. Experimental results demonstrate the performance of these approaches.

## Keywords

Energy efficiency, hadoop, reliability, PSO.

## 1. INTRODUCTION

A distributed file system is designed to hold a large volume of data distributed across the network for access by a number of clients. The Network File System (NFS), the most ubiquitous distributed file system, provides remote access to a single logical volume stored on a single machine. An NFS server makes a portion of its local file system visible to external clients. The clients can then mount this remote file system directly into their own Linux file system, and interact with it as though it were part of the local drive. Although it is transparent, it is limited in its power. The files in an NFS volume all reside on a single machine. Hence it is less reliable, scalable and server can get overloaded. Hadoop Distributed File System (HDFS) [3] is designed to be robust by storing large amount of information (terabytes or petabytes) across a large number of machines. It is tolerant to failures and scalable.

Hadoop [1] is a large scale distributed framework of commodity hardware that is suitable to process a batch of data intensive applications in parallel. Files in HDFS are chunked, replicated and distributed across multiple machines to for reliability. Conventional Hadoop grid ensures reliability by

maintaining all nodes in active state. At any point of time atleast 15% of the nodes are idle [2]. This affects the energy efficiency of the cluster. In Hadoop data and computations are clustered. Choosing proper nodes to replicate the blocks and then submitting the jobs to those nodes can reduce the energy consumption. By default HDFS [3] maintains three copies of information in the cluster for reliability. Decreasing the replicas can compromise on reliability. This paper addresses the problem of block allocation in the cluster to maintain energy efficiency and reliability. It identifies the minimum number of nodes in a cluster to maintain reliability and consume less power. A greedy approach and an evolutionary approach using PSO have been designed and implemented. The scheduler component is also modified to submit the jobs to suitable datanodes.

Energy consumed by datacenters doubles on an average over a period of 5 years [1]. A power controller to address power conservation requirement in hadoop cluster has been proposed. It dynamically reconfigures the cluster based on current workload based on the workload. Migration Of blocks has been proposed. This approach has been tested on GridSim [4]. Such dynamic reconfiguration on hadoop clusters requires major changes in HDFS code.

Optimisation of energy efficiency of servers can be done at component level in processors, memories and disks [7, 8, 9, 10]. Dynamic VoltageScaling (DVS), request batching and multi-speed disks [11, 12, 13] can also be used to improve energy efficiency. The proposed approach uses the global state of cluster to handle workload imposed on it.

A multi-tiered infrastructure [14] that enables dynamic migration of virtual machine execution flow within and across computer nodes has been studied. Within a node the framework allocates and re-allocates virtual processors to their physical counterparts. Across nodes the framework employs live migration to relocate complete guest operating system instances to distinct physical hosts. Hadoop forms a single data compute cluster. So migration of jobs can be done only based on data block location.

A power-aware application placement controller for heterogeneous virtualized server clusters is proposed [15]. It considers power, migration costs and benefits. Hadoop does not advocate the migration of partially executed jobs. The blocks and the job have to be migrated to a new node and re-execution should occur. Server virtualization technologies to enable the replication and migration of server functions across WANs are proposed [16]. Extensions to existing cloudsim toolkit [17] to model power aware resource provisioning, which includes generation of dynamic workload patterns, workload prediction and adaptive provisioning, dynamic lifecycle management of random workload, and

implementation of power aware allocation policies and chip aware VM scheduler has been proposed. This paper suggests its implementation on a simulator. A metascheduler called Adaptive Power-Aware Virtual Machine Provisioner (APA-VMP) that schedules the workload in such a way that the total incremental power drawn by the server pool is minimum without compromising the performance objectives has been proposed [18]. The APA-VMP makes use of swarm intelligence methodology to detect and track the changing optimal target servers for VM placement very efficiently. The work was implemented on CloudSim and is more suitable for computational clouds that involves migration of jobs and does not consider data placement.

A power-aware linear programming based scheduling policy [19] for heterogeneous compute clusters has been proposed. This work is not suitable for data intensive clusters like Hadoop. GreenHDFS [20] relies on data classification driven approach for data placement. It moves data blocks dynamically from hot zone to cold zone of less power consumption. Such movement involves considerable changes in HDFS code. The approach has only been simulated.

Two standard approaches are used to improve energy efficiency in a cluster

- Covering Set(CS): This approach keeps only a small fraction of the cluster nodes powered on by ensuring that at least one copy of data is maintained in these nodes. It is based on altering the data placement policy of the underlying DFS to save energy. Some of the drawbacks of this approach include longer execution time of workloads and modifications in DFS software.
- All In Strategy (AIS) uses all the nodes in the cluster to run a batch of jobs and then powers down the entire cluster. Here the entire cluster is made active to execute the jobs and then moved to low power state. Energy efficiency of AIS approach for large workloads is lower than that of CS approach. A fraction of time is wasted to power up/down all the servers.

The proposed approach uses a combination of both to balance data availability and performance. It selects the minimum number of nodes to be maintained in active state in the cluster by taking into consideration data availability. When the cluster is idle all nodes are powered down. The framework also includes components for reliable execution of jobs. A reliability factor based on history and utilization based on processor MIPS, CPU usage and memory usage are considered to aid submission of jobs to reliable and efficient nodes.

## 2. SYSTEM ARCHITECTURE

The proposed scheduler is developed for the purpose of energy efficient and reliable job completion in hadoop. This scheduler examines the utilization level and number of replicas in nodes of the cluster and submits the job only to the nodes that are selected to live in the cluster to run the job. To submit the job to the selected nodes, the active trackers for selected nodes are obtained. Then directions are given to the namenode explicitly to submit the job to those active trackers. Check for the utilization levels of those active trackers ensures reliability. The various steps for the proposed approach are as follows:

- 1) Load the file(s) required for the proposed jobs into HDFS
- 2) Read name node metadata to identify the location of blocks of files.

- 3) Use greedy / PSO approach to identify the minimum set of machines to hold all the blocks of information for a particular batch of jobs (details in section 4).
- 4) Use Wake On LAN (WOL) [23] to turn move redundant machines to sleep state. Wake-On-LAN (WOL) allows a server to be moved into power saving mode. When a wake-up packet is received, the system powers up as normal.
- 5) Inform the namenode not to create new replicas.
- 6) Capture the static and dynamic node features. Dynamic node features include CPU load average, RAM space, Hard disk space, Disk I/O rate and static node features include number of CPUs, total hard disk capacity and total RAM capacity. The dynamic node price is evaluated using equation 1.

$$\begin{aligned} \text{Dynamic node price} &= \text{Fixed Price} + \text{Fractional Price} \\ \text{Fractional Price} &= \text{Price} * (\text{Lavg} + \text{HDutil} + \text{RAMutil}) \end{aligned} \quad (1)$$

Where

*Lavg* is Load average

$$\text{HDutil} = \frac{\text{Hard Disk utilized}}{\text{total Hard Disk space}}$$

$$\text{RAMutil} = \frac{\text{RAM utilized}}{\text{Total RAM available}}$$

- 7) A learning scheduler that contains classifier finds the best tracker for a job. It identifies the best tracker(s) for the job based on equation 2, 3 and 4. The classifier takes in to account the number of job features and satisfying number of node features. If the features are not satisfying then the probability of that feature variable is taken as zero. The node which satisfies the highest probability is considered as the best fit for the task. If a task is suitable for more than one node, the most suitable node is selected based on equation 3. Here the reliability depends on the history of node reliability. Reliability of active nodes (reliability\_active) identified in step 4 is set to 1. Reliability of inactive nodes is set to a very low value. A history is also maintained for node reliability (reliability\_history) based on how many times a task were completed successfully by the node. It is gathered from the job tracker logs. Now the reliability is given by equation 3. The provider rating that is minimum is chosen.

$$P(\text{Node}) = \frac{(\# \text{ job features} + \# \text{ node features} * (P(F1) + P(F2) + \dots + P(Fn)))}{\text{Total Features}} \quad (2)$$

$$\begin{aligned} \text{reliability} &= \text{reliability\_active} * \\ &\text{reliability\_history} \end{aligned} \quad (3)$$

$$\text{Provider Rating} = \text{current price} + \left( \frac{1}{\text{reliability}} \right) \quad (4)$$

If many jobs satisfy a single node then based on the utility function they are passed to the tracker from the queues. The utility function is first in first out type (with backfilling of small jobs between large jobs).

- 8) *Job Submission:* The job is submitted explicitly to most suitable node.

- 9) *Updating history file:* Once a job has been executed successfully, history files are updated, so that in future if same job is submitted to the user, the scheduler can directly submit the job to suitable tracker. Two history files are maintained - one for the job and another for the node reliability. Once a job has been executed successfully, the node reliability is incremented; else the node reliability is decremented.

The details of jobs executed successfully and the node details are maintained in another history file. The history file contains jobs features like RAM, Hard disk space, input size, maps and reduces and the name of the cluster node(s) to which the job has been assigned. Thus the response time is improved by gathering the knowledge of past runs.

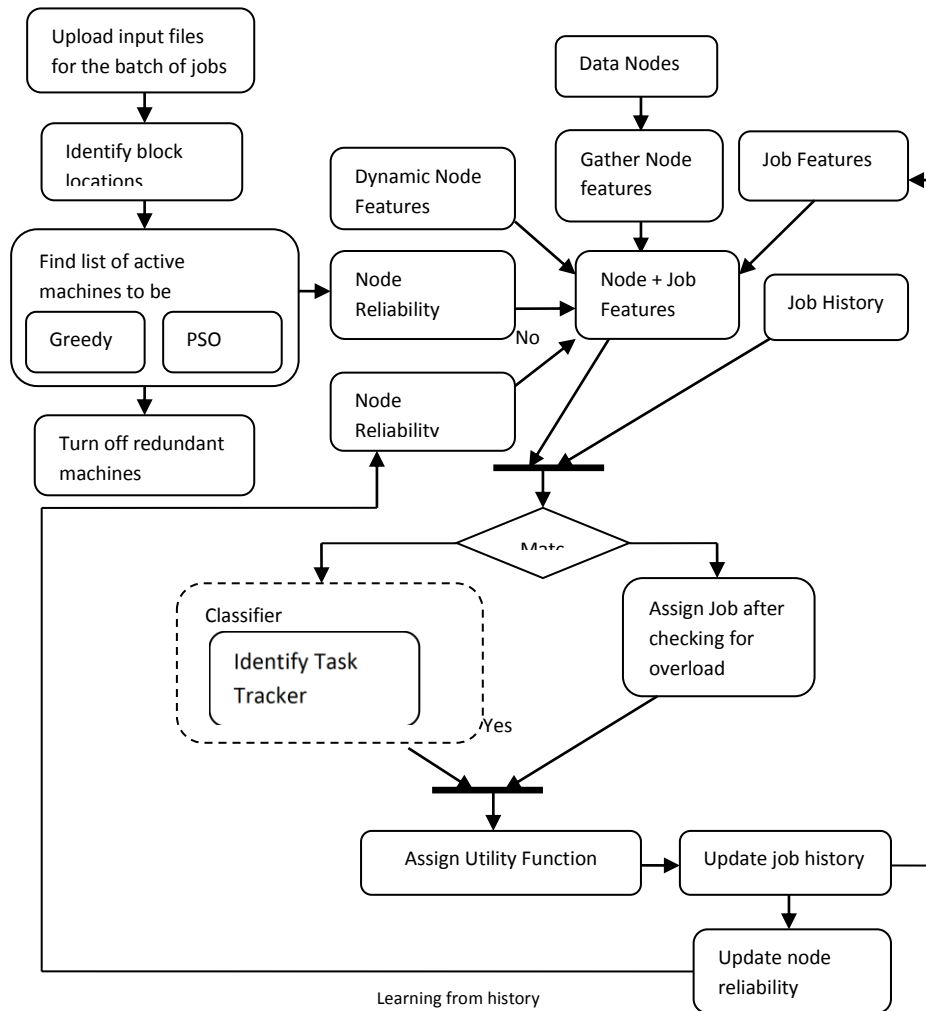


Fig 1: Flow Chart

### 3. A CASE STUDY

Consider the node features of machines in a cluster as in table 1. The other features are provided in table 2. Let all nodes have 400 GB hard disk (HD) and 2000 MB RAM totally. Let the Fixed Price be assigned as 100.

$$\text{Dynamic Price} = \text{Fixed Price} + \text{Fixed Price} * (\text{Load Avg} + \text{Hard Disk Utilised} / \text{Total Hard Disk} + \text{RAM utilized} / \text{Total RAM})$$

Load Avg	Hard Disk Utilised	RAM utilized	Dynamic Price	Node Name
0.4	80	963	136.05	C1
0.3	144	1300	143.67	C2
0.5	272	876	153.93	C3
0.6	300	1563	171.05	C4
0.2	272	460	137	C5
0.15	320	1544	157.4	C6

The details of jobs J1 and J2 submitted are as follows:

J1: RAM=500MB. HD=120 GB M=8 R=0, reliability = 0.5, price between 125 to 150

J2: RAM=400Mb, HD=100GB, M=6, R=2, reliability = 0.7, price between 130 to 170

Let J1 and J2 are jobs and C (I) are cluster nodes. P(J1 CI) indicates the probability of success of submitting J1 to CI machine.

P(J1C1)=0.92 ,P(J1C2)=0.92 ,P(J1C3)=0.83,P(J1C4)=0.67, P(J1C5)=0.83, P(J1C6)=0.67.

P(J2C1)=0.92, P(J2C2)=0.83, P(J2C3)=0.75, P(J2C4)=0.92, P(J2C5)=0.83, P(J2C6)=0.83

Hence J1 can be submitted to cluster nodes C1, C2 and J2 is submitted to C1, C4, based on pricing. C2 is switched off. Hence the jobs are submitted to C1 and C4.

**Table 1, Features captured during one execution (1)**

Free RAM (MB)	Free Hard disk (GB)	No. map tasks	No. reduce tasks	Reliability history	Reliability_active (nodes selected)	Load Average	Node name
1037	320	8	2	0.3	1	0.4	C1
700	256	5	3	0.5	0.0001	0.3	C2
1124	128	5	1	0.6	1	0.5	C3
437	100	6	3	0.8	1	0.6	C4
1540	128	6	2	0.4	0.0001	0.2	C5
456	80	5	2	0.7	0.0001	0.15	C6

#### 4. NODE SELECTION

The selection of nodes from the cluster is based on the number of replicas the node that it has. To select the node from different size of cluster we have implemented two approaches:

- Greedy Approach
- Particle Swarm Optimization Approach

Greedy approach is implemented based on the node that has total number of replicas of different blocks of file in HDFS. The nodes are sorted based on total number of replicas and the nodes are selected one by one until it covers at least one of the replicas of a block. This approach is efficient for the small cluster size.

Since the subset cover problem is a NP hard problem, the number of combinations to be explored becomes very large. To accommodate for the node selection from large cluster Particle Swarm Optimization is used. PSO [...] shares many similarities with evolutionary computation techniques such as Genetic Algorithms (GA). The system is initialized with a population of random solutions and searches for optima by updating generations. However, unlike GA, PSO has no evolution operators such as crossover and mutation. In PSO, the potential solutions, called particles, fly through the problem space by following the current optimum particles.

PSO is initialized with a group of random particles (solutions) from which an optimal solution is searched by updating generations. Each particle is updated in every iteration by following two "best" values. The first one is the best solution (fitness) the particle has achieved so far. This value is called personal best (pbest). The "best" value that is obtained so far by any particle in the population is the second one. This best value is a global best (gbest). When a particle takes part of the

population as its topological neighbors, the best value is a local best and is called lbest.

After finding the two best values, the particle updates its velocity and positions in iteration 'i+1' with following equation (4) and (5).

$$v[i+1] = 0.7298(v[i] + c1 * rand() * (pbest[i] - present[i]) + c2 * rand() * (gbest[i]-present[i])) \quad (4)$$

$$present[i+1] = present[i] + v[i+1] \quad (5)$$

The pseudo code of the procedure is as follows:

- 1) Initialize the particles by reading in the block locations.
- 2) Identify the valid particles using the constraint
- 3) Repeat for each particle in parallel.
  - a. Calculate current fitness (Number of blocks covered)
  - b. If current >pbest then update pbest
- 4) Update gbest by comparing values of all pbest
- 5) Update particle velocity according to equation (4)
- 6) Update particle position based on equation (5)
- 7) Repeat steps 3 to 6 for convergence.

v[i] is the particle velocity, present[i] is the current particle fitness (solution). pbest[i] and gbest[i] are personal and global based. rand for the ith generation. rand() is a random number between (0,1). c1, c2 are learning factors. Usually c1 = c2 = 0.5.

In our problem each particle is a subset of nodes in the cluster. Valid particles and each particle (solutions) are subject to satisfy the constraint that they cover all of the blocks needed for the batch of jobs. The fitness function is the count of number of blocks covered. The computation of velocity, local best and global best is found for every particle and iterated over time until the solution converges at certain amount of time.

#### 5. EXPERIMENTAL RESULTS

This section details on the results of node selection using PSO. This chapter explains the results that were taken to measure time required to select the nodes to live using Particle Swarm optimization, time required to complete MapReduce jobs with configurations, minimum number of nodes required to cover all of the block replicas in HDFS and amount energy saved from the proposed system. The results provided here are based on the storage of block replicas in cluster and behavior of the algorithm used in selection of nodes and scheduling of MapReduce job in the cluster.

##### 5.1. Performance of PSO

Table 3 shows the result of applying PSO after obtaining the block locations. The results provided in the table 1 are based on the location of replicas among the nodes of a cluster. There may be variation in results when the cluster status varies. It can be seen that the number of nodes required has reduced by 55% to 65%. From table 2, it can be seen that even though the data blocks remain a constant, as the number of machines increases the minimum number of machines required also increases. This is because as the cluster size becomes large, the blocks gets placed sparsely. As the data size increases, minimum number of required blocks also increases. The reduction in number of machines required to do the job varies from 80% to 60% (table 3).

**Table 3, PSO results for various numbers of machines and blocks**

NUMBER OF MACHINES	NUMBER OF BLOCKS	MINIMUM NO OF MACHINES REQUIRED	NUMBER OF ITERATIONS	TIME (sec)
9	120	4	10	1
500	1000	150	16	4
1000	5000	380	19	9
2000	10000	700	23	12

**Table 4, Performance with varying number of machines**

NUMBER OF MACHINES	NUMBER OF BLOCKS	MINIMUM NO OF MACHINES REQUIRED
500	10000	200
1000	10000	320
2000	10000	650

**Table 5: Performance with varying number of blocks**

NUMBER OF MACHINES	NUMBER OF BLOCKS	MINIMUM NO OF MACHINES REQUIRED
1000	1000	200
1000	5000	350
1000	10000	400

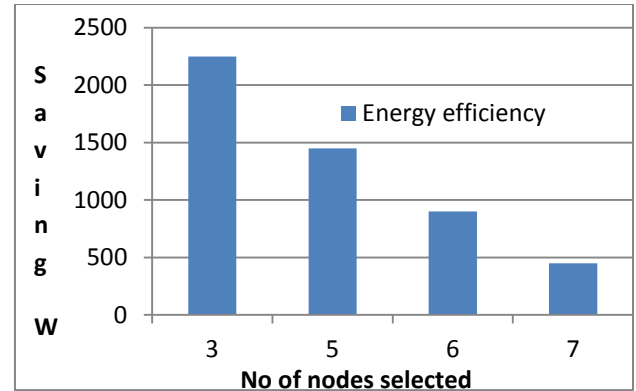
When the greedy approach used for the selection of nodes from the large from cluster, it takes more time. So it is suggested to use intelligence approach like PSO reach the near optimal solution quickly.

## 5.2. Measure of energy efficiency and reliability

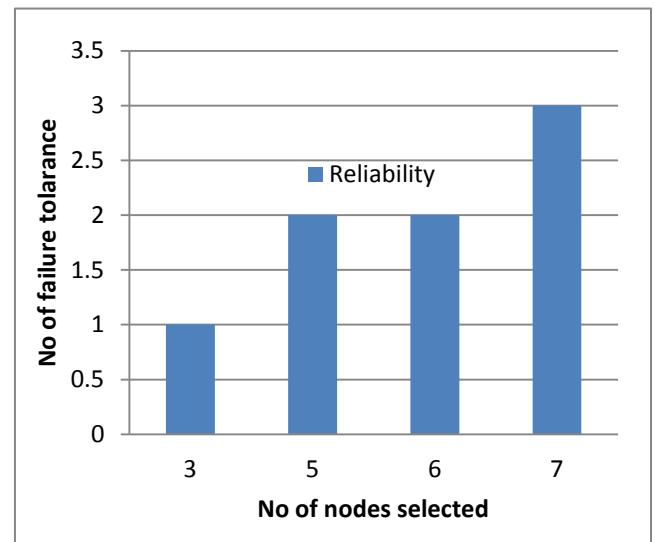
The result provided below is based on the nodes selected to be live in cluster to perform the MapReduce job and power required for nodes in the cluster. The power savings is measured in terms of power required for Dell PowerEdge SC1425 Rack Server without including power consumption by Air Conditioners. Since power supply for SC 1425 Rack Server is 450 watts and cluster consists of 9 nodes including NameNode. So the power saving is found by power consumed the other nodes in the cluster. Reliability is measured in terms of running the MapReduce job completely with failure of any live nodes in the cluster. It simply tells how many failures of nodes can be tolerated during the process of MapReduce job. The result of power savings is shown in fig.2. The reliability of cluster is shown in fig.3.

**Table 6, Measure of reliability and energy efficiency**

NUMBER OF MACHINES	TIME		POWER SAVINGS (Watts)	RELIABILITY
	min	Sec		
3	10	52	2250	1
5	9	29	1350	2
6	8	35	900	2
7	5	20	450	3



**Fig 2: Energy efficiency**



**Fig 3: Reliability measure**

## 5.3 Time efficiency

The table 7 shows the time required for the completion of job with our proposed scheduler, default scheduler with some nodes powered down and default scheduler. Here the size of input file processed by a MapReduce job is about 1.2 GB and each block is of size 64 MB. We have encountered errors when using default scheduler with nodes powered down. As the proposed scheduler submits the job only to live nodes in the cluster, it reduces time required to run the MapReduce job and ensures the reliability during the job processing.

**Table 7: Time measurement with various situations**

TYPE OF CONFIGURATION	TIME	
	min	Sec
Proposed Scheduler	10	52
Default Scheduler with some nodes powered down	12	57
Default Scheduler	14	19


## 6. CONCLUSION

This paper proposes an energy efficient and reliable block placement strategy in hadoop. Two approaches have been suggested to identify the set of active nodes. A learning scheduler is used to submit the jobs to the cluster in an efficient and reliable manner. Job history files are maintained that can perform job matching and assign it to correct tracker in the cluster. If information about a job is not available in the history file then job classification takes place based on job and node features. This minimizes the time required to run the job and ensures reliability in the completion of job. The system provides flexibility in powering up and down the nodes using Wake On Lan mechanism for energy efficiency. Experimental results demonstrate that the proposed approach performs better than default schedulers in Hadoop in terms of energy consumption and reliability.

## 7. ACKNOWLEDGEMENTS

The authors acknowledge the team members of Grid and Cloud Systems Group, Yahoo R&D, India for their help in carrying out this project. The authors thank Dr R Rudramoorthy, Principal for his support. This project is an outcome of PSG-Yahoo Research on grid and cloud computing.

## 8. REFERENCES

- [1] Hadoop - <http://developer.yahoo.com> – Hadoop internals and tutorial.
- [2] HDFS - <http://hadoop.apache.org/hdfs> – Study about HDFS and its development.
- [3] R. Buyya, M. Murshed, Gridsim: a toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing, *Concurrency and Computation: Practice and Experience* 14 (2002), 1175{1220. doi:<http://dx.doi.org/10.1002/cpe.710>.
- [4] Willis Lang and Jignesh M. Patel, Energy Management for MapReduce Clusters, Computer Sciences Department, University of WisconsinMadison, USA.
- [5] Nitesh Maheshwari, Radheshyam Nanduri, Vasudeva Varma, Dynamic Energy Efficient Data Placement and Cluster Reconfiguration Algorithm for MapReduce Framework, Search and Information Extraction Lab, Language Technologies Research Centre (LTRC), IIIT Hyderabad.
- [6] Jacob Leverich, Christos Kozyrakis, On the Energy (In)efficiency of Hadoop Clusters, Computer Systems Laboratory, Stanford University.
- [7] Yanpei Chen, Laura Keys, Randy Katz , Hadoop Summit 2009 – Towards Energy Efficient Hadoop -, RAD Lab, UC Berkeley.
- [8] Hyeong S. Kim Dong In Shin Young Jin Yu Hyeonsang Eom Heon Y. Yeom., Towards Energy Proportional Cloud for Data Processing Frameworks, School of Computer Science and Engineering, Seoul National University.
- [9] M. Weiser, B. Welch, A. Demers, S. Shenker, Scheduling for reduced cpu energy, in: OSDI '94: Proceedings of the 1st USENIX conferencon Operating Systems Design and Implementation, USENIX Association, Berkeley, CA, USA, 1994, p. 2.
- [10] A. Rangasamy, R. Nagpal, Y. Srikant, Compiler-directed frequency and voltage scaling for a multiple clock domain micro architecture, in: CF '08: Proceedings of the 5th conference on Computing frontiers, ACM, New York, NY,USA, 2008, pp. 209{218.doi:<http://doi.acm.org/10.1145/1366230.1366267>
- [11] R. Lebeck, X. Fan, H. Zeng, C. Ellis, Power aware page allocation, in: ASPLOS-IX: Proceedings of the ninth international conference on Architectural support for programming languages and operating systems, ACM,NewYork,NY,USA,2000,pp.105{116.doi:<http://doi.acm.org/10.1145/378993.379007>.
- [12] D. P. Helmbold, D. D. Long, T. L. Sconyers, B. Sherrod, Adaptive diskspindown for mobile computers, *Mobile Networks and Applications* 5 (2000) 285{297.
- [13] M. Elnozahy, M. Kistler, R. Rajamony, Energy conservation policies for web servers, in: USITS'03: Proceedings of the 4th conference on USENIX Symposium on Internet Technologies and Systems, USENIX Association, Berkeley, CA, USA, 2003.
- [14] E. V. Carrera, E. Pinheiro, R. Bianchini, Conserving disk energy in net-work servers, in: ICS '03: Proceedings of the 17th annual international conference on Supercomputing, ACM, New York, NY, USA, 2003, pp.86{97. doi:<http://doi.acm.org/10.1145/782814.782829>.
- [15] S. Gurusurthi, A. Sivasubramaniam, M. Kandemir, H. Franke, Drpm: Dynamic speed control for power management in server class disks, *Computer Architecture, International Symposium on* 0 (2003) 169.doi:<http://doi.ieeecomputersociety.org/10.1109/ISC.A.2003.1206998>.
- [16] Jan Stoess , Christoph Klee , Stefan Domthera , Frank Bellosa, Transparent, Power-Aware Migration in Virtualized Systems.
- [17] Akshat Verma, Puneet Ahuja and Anindya Neogi, pMapper: Power and Migration Cost Aware Application Placement in Virtualized Systems
- [18] Live Data Center Migration acrossWANs: A Robust Cooperative Context Aware Approach K.K. Ramakrishnan, Prashant Shenoy , Jacobus Van der Merwe AT&T Labs-Research /  University of Massachusetts
- [19] intelligence R. Jeyarani , R. Vasanth Ram , N. Nagaveni, Design and implementation of adaptive power-aware virtual machine provisioner (APA-VMP) using swarm
- [20] Power-aware linear programming based scheduling for heterogeneous computer clusters. Rini T Kaushik, Milind Bhandarkar, GreenHDFS: Towards an energy-conserving, storage-efficient, hybrid Hadoop compute cluster.
- [21] WOL – <http://wikipedia.org/wol>.
- [22] PSO reference - <http://www.swarmintelligence.org> – Study about PSO.