An Energy-Efficient Data Placement Algorithm and Node Scheduling Strategies in Cloud Computing Systems

Yanwen Xiao
Massive Data Computing Research Lab
Harbin Institute of Technology, HIT
Harbin, China
elvinhit@gmail.com

Jinbao Wang
Massive Data Computing Research Lab
Harbin Institute of Technology, HIT
Harbin, China
wangjinbaosky@gmail.com

Yaping Li
Internet & Information Center
Harbin Institute of Technology, HIT
Harbin, China
LYP@hit.edu.cn

Hong Gao
Massive Data Computing Research Lab
Harbin Institute of Technology, HIT
Harbin, China
honggao@hit.edu.cn

***Abstract*—With the rise of the cloud computing, saving energy consumed by cloud systems has become a tricky issue nowadays. How to place data efficiently and schedule the nodes effectively in a cloud platform are very important issues from the view of the energy-saving. However, the state-of-the-art node-scheduling strategies can't save large amount of energy for the cloud computing platforms significantly. This paper proposes a heuristic data placement algorithm and two node scheduling strategies for cloud platforms to save energy with tasks guaranteed. The Cloudsim is employed to simulate a private cloud system. Energy-saving is achieved by turning on minimum nodes to cover maximum data blocks. The problem of covering data block with computing nodes is abstracted as a set cover problem, and a greedy algorithm is utilized to solve this problem. This approach is practical to any cloud computing infrastructure. The designed experiment verifies the efficiency of the data placement algorithm and node scheduling strategies proposed in this paper.**

Keywords-cloud computing; data placement algorithm; node-scheduling strategies; energy efficiency

I. INTRODUCTION

Recently, with the development of cloud computing, various cloud computing products have been tremendously beneficial for network applications, such as Google cloud platform, Amazon EC2, IBM Blue Cloud etc. However, data centers are composed of thousands of servers which consume a large amount of energy. Thus, energy-saving is a critical issue for IT organizations. As addressed in [3] [4], data centers in the U.S. consumed about 4.5 billion kWh in 2005, equaling roughly 1.2% of the total electricity consumption and about 6.1 billion kWh in 2006, roughly 45 billion dollars. The storage energy consumption rate for all IT equipment will increase more and more in the short time because the amount of digital data is increasing quickly[5], energy consumption has gradually beyond equipment hardware costs[3]. The huge energy consumption has been emphasized by lots of enterprises.

Enormous energy has been wasted due to idle resources. A report of NRDC pointed that idle servers use 69% to 97% of the total energy even if the power management function is enabled[1]. As a result, shutting down idle server will save large amount of energy. [9] is limited to the energy research of Hadoop, and using HDFS internal architecture to optimize data placement is too limited and is not applied to most of the cloud computing environment. [15] [16] focus on solving the problem of over-load scheduling and large resource utilization.[17] proposes a novel approach which enables application tasks placement dynamically with consideration of energy efficient with the aim of minimizing the number of the active nodes. Hence, how to place data in a cloud platform is an important issue to reduce energy cost.

Recently, [2][10][11][12][13][14] present a lot of energy-saving methods at the hardware level, such as voltage settings, processor speed adjustment, enlarging memory and using low power solid state hard drive etc. However, those methods are only useful for PC or single computer and cannot achieve maximum energy optimization. Because the energy saved by these techniques such as scaling down the CPU voltage is far less than turning off a computer. Therefore, an efficient energy-saving approach for the whole cloud platform is needed.

This paper proposes an energy efficient data placement algorithm and node-scheduling strategies in cloud computing systems. Data blocks are placed rational by data placement algorithm to find minimum nodes sets that contains all the data blocks collection, then the remaining nodes are turned off by the node scheduling strategies to save energy.

The major contributions are summarized as follows：

We propose an energy-efficient data placement algorithm which supports node scheduling strategy and optimizes more space for node scheduling.

Two optimization goals for batch processing are addressed: (1) Given power consumption upper bound, minimizing the execution time of task requests by node scheduling. (2) Given task request execution time upper bound, minimizing power consumption of the set of active

nodes by node scheduling. We show that the second optimization problem is NP-complete by reducing to a weighted set cover problem.

The remaining parts of this paper are organized as follows. Section II introduces related works. Section III gives backgrounds. A dynamic data placement algorithm is devised in Section IV. Section V presents the definition of the scheduling problem. Section VI reports evaluation results by experiment and uses Cloudsim to verify the proposed algorithm. Finally, we conclude the paper in Section VII.

II. RELATED WORKS

[6] [7] [8] are based on energy-saving strategies for disk storage systems research. [6] proposes PDC (Popular Data Concentration) algorithm, the idea behind PDC is to dynamically migrate the popular disk data (i.e., the most frequently accessed data on disk) to a subset of the disks in the array, so that the load becomes skewed towards a few of the disks and others can be sent to energy-saving mode.[7] proposes a dynamic data reorganization algorithm and the basic idea is the dynamic block exchange algorithm which switches data between such units based on the observed workload such that frequently accessed blocks end up residing on a few "hot" units thus allowing the majority nodes to experience longer idle periods. [7] proposes an energy-saving strategy that the application layer and the storage layer are both considered and using cache to make storage nodes experience longer idle periods.

Currently there are some energy-saving articles in the cloud computing platform, [9] [17] proposes a method that turn off a certain number of nodes to save energy. [9] turns on (turns off) cluster nodes based on the presently utilization of the whole cluster nodes, this method uses HDFS framework which can't be fully used in a cloud computing environment. [17] considers energy-saving in the application layer perspective and proposes EnCloud algorithm which minimize the number of the active nodes by allocating resources for each application. But they did not consider whether the data blocks in the active nodes satisfied the applications. If data blocks meet applications request, it will lead to turn on lots of redundant nodes and can't save energy efficiently. [18] proposes a cloud computing energy-saving framework based on resources allocation, which gives a virtual node placement strategy and decides to turn on some physical nodes by the active virtual nodes, but they did not consider data placement problem.

III. BACKGROUND

A. Assumptions and Notations

The infrastructure of cloud computing environment is usually composed of hundreds or even thousands of server nodes. Every server node consists of processor, memory.

We assume that the data are already prepared before requests arrive and can be accessed any time in the beginning. The initial deployment of data in cloud computing platform is needed. Notations are shown in Table I.

TABLE I. NOTATIONS

Notation	*Description*
D(set)	data blocks set
S(set)	cluster nodes set
S^**(set)*	*The min active nodes set*
G*(set)*	*group set*
g_i*(set*）	*nodes in group i*
$J_{request}$	*order request queue*
N	*N stands for the number of nodes*
K	*replica of data blocks*
M	*M stands for the number of data blocks*
p	*primary block of replica data blocks*
Map	*the table of data block map to globe position*
T_{max}	*max execution time of job request*
P_{max}	*max power of cluster consume*
P_{avg}	*stand for avgre power cost of each node*
$P_{cluster}$	*power cost of active nodes*
T	*schedule time*
a	*the number of active nodes*
JS	*job schedule strategy*
$\gamma(s)$	*cover rate of node s*
U_{up}	*node resource utilization threshold*
U_{down}	*node resource utilization threshold*
U_s	*resource utilization of node s*
B	*size of each data block*

IV. AN DYNAMIC DATA PLACEMENT ALGORITHM

A. Initial Deployment of Data Blocks

Firstly, the data are partitioned into several blocks with size of B, and then generate to the whole data blocks set D = $\{d_0, d_1, d_2, \ldots, d_{M-1}\}$. This paper uses the replica of data blocks to ensure the reliability of data and define the number of replica of each data block as replica factor K (K > 1). Every data block d_i ($0 \le i \le M$) has a primary replica block p and K-1 slavery replica blocks $h_1, \ldots, h_{K-1}$. We assume that the primary replica block p will be accessed firstly unless there are updates in p and updates will be transferred to other blocks in the same time. Job requests will not access other slavery replica until p overloads or has some errors.

Denote S = $\{s_0, s_1, s_2, \ldots, s_{N-1}\}$ as a storage nodes set. Data blocks are placed as follows: firstly, partition all data containing primary replica blocks and slavery replica blocks into K groups, and each group equals the whole data set which contains M data blocks. Secondly, store the K groups into N/K nodes, and the node allocate to each group is different. In each group, there is a map table which maps data blocks to position of nodes and all map tables in each group are stored in *Map* which is stored in the main console model. Group set is defined as $G=\{g_0, g_1, g_2, \ldots, g_{N/K-1}\}$, where g_i stands for number i group，$g_i=\{\ldots, s_j, \ldots s_z, \ldots\}$($0 \leqslant j \leqslant N$，$0 \leqslant z \leqslant N$，$j \neq z$)，the element in set g_i stands for cluster nodes. From two aspects of the response time and energy considerations, this paper adopts different arrangement strategy and we use random mapping mode to map data blocks to the nodes and store those information in the *Map*.

B. Dynamic Data Placement Algorithm

Definition 1： Each node $s \in S$, in node s q is the number of data blocks , p is the number of data blocks which meet the job requests , the definition of node cover rate $\gamma(s) = p/q$.

Definition 2: If node s and node s_t have the same data block replica, then node s and node s_t are data-exchangeable.

The formula (1) is used to compute resource utilization of nodes :

$$U=e \cdot U_{cpu}+(1-e) \cdot U_{disk} \quad (1)$$

U_{cpu} stands for utilization of CPU, U_{disk} stands for utilization of the disk, e is a scale factor. Dynamic data placement algorithm is shown in Algorithm 1:

Algorithm 1 Dynamic Data Placement Algorithm

Input : Map
Output: Data placement strategy
1. Find S_{util} in which resource utilization > U_{up} (<U_{down}) with *Map*
2. while $S_{util} \neq \emptyset$ do
3. $\forall s \in S_{util}$, compute U_s
4. If $s \in S_{util}$ and $U_s > U_{up}$ do
5. Find *s'* data-exchangeable node s_t, where s_t meet s $Us_t < U_{up}$
6. Turn on node s_t ,transfer p which overloads in node s to s_t
7. Scan *Map*, find *s'* which meets $U_{s'} > U_{up}$ and add *s'* into S_{util}
8. end if
9. If $s \in S_{util}$ and $U_s < U_{down}$ do
10. Find *s'* data-exchangeable node s_t, where s_t meet $Us_t > U_{down}$
11. Transfer p which belongs to node s into s_t, and turn off node s_t
12. Scan *Map*, find *s'* which meets $U_{s'} < U_{down}$ and add *s'* into S_{util}
13. end if
14. end while

V. SCHEDULING ALGORITHM

Energy-saving effect is influenced by the node scheduling which is the key of research in cloud computing platform. This paper proposes optimization algorithm for the batch scheduling and the online scheduling.

A. Batch Scheduling Strategy

Job requests set is $J_{request}=\{r_0, r_1, r_2,\ldots r_w\}$ (w > 0), assume that job execution time of each node is $T_i= T_{cpu}+T_{i/o}$, P_{avg} is average power of each node and $P_{cluster}= aP_{avg}$ (a stands for the number of active nodes).

Algorithm 2-1 Batch Scheduling Algorithm –Optimization Objective 1

Input : *Map*, P_{max}, $J_{request}$
Output : $JS= \{ JS \mid J_{request} \rightarrow S \}$
1. Compute $\gamma(s)$ of all nodes in *S* by *Map*
2. Compute $V = P_{max}/P_{avg}$
3. While $J_{request} \neq$ □ do
4. Turn on nodes that the range of $\gamma(s)$ is from 1 to *V*
5. If r_i is read operation ($r_i \in J_{request}$) do
6. Allocate r_i to any active node， update JS
7. else
8. r_i do some operation on *P* in active nodes， update *JS*
9. end if
10. If there are not job requests for the active node do
11. Find the node whose $\gamma(s)$ is highest and turn it on
12. Turn off currently active node
13. end if
14. Call dynamic data placement algorithm
15. end while

Given the upper bound of power consumption .

Optimization Objective 1:

minimize $T_e= \sum_{i=0}^{a-1} T_i$

subject to $P_{cluster} \leq P_{max}$

Problem Definition 1:

Input: *Map*; P_{max}; $J_{request}= \{ r_0, r_1, r_2,\ldots r_n \}$ (n > 0)

Output: $JS= \{ JS \mid J_{request} \rightarrow S \}$

Constraints: $P_{cluster} \leq P_{max}$

As shown in Algorithm 2-1, we can compute the number of active nodes $V=P_{max}/P_{avg}$ by P_{avg}, so the problem is transformed to: given the number of available nodes *V*, minimizing the execution time of all job requests T_e by using node scheduling strategy.

There is another situation that given the upper bound of execution time T_{max}, find the node set that minimizes $P_{cluster}$ which stands for power consumption of currently cluster nodes.

Optimization Objective 2:

Minimize $P_{cluster}$

Subject to $T_e \leq T_{max}$

Problem Definition 2:

Given the problem of batch scheduling $F(J_{reuqest}, S, T)$, find the active node set S^* whose amount is minimum.

Input: *Map*; *T*; $J_{request} =\{ r_0, r_1, r_2,\ldots r_n \}$ (n > 0)

Output: $S^*= \min \{ |S^*| \mid S^* \subseteq S \}$

We have the theorem as follows:

Theorem 1: The problem of $F(J_{reuqest}, S, T)$ is equivalent to the weighted set cover problem $WSC(U,S^{\#})$.

Proof: $d_i \in s$, d_i meets r_i and r_i belongs to $J_{request}$. s is a single node and $s \in S$. $J_{request}$ can be abstracted as *U*, and *S* is the set that contains all nodes and positions of all data blocks, so *S* can be abstracted as $S^{\#}$. Because *T* is the limit of this problem, so *T* can be viewed as the weight of finding min set. Therefore, the above description accords the definition of set cover problem, and the problem of $F(J_{reuqest}, S, T)$ can be abstracted as the weighted set cover problem.

Similarly, we can prove that the weighted set cover problem can be abstracted as the problem of $F(J_{reuqest}, S, T)$.

Algorithm 2-2 Batch Scheduling Algorithm –Optimization Objective2

Input : *U*, $S^{\#}$, *T*, $J_{request}$
Output: S_*
1. $S_* \leftarrow \emptyset$
2. while $J_{request} \neq \emptyset$ do
3. Compute $\gamma(s)$ of nodes in $S^{\#}$
4. while $f(S_*) < |U|$ and $\sum_{i=0}^{a-1} T_i < T$ do
5. $f(S_*) \leftarrow \sum |s_i|$ （ $s_i \in S_*$ ）
6. Chose the node *s* whose$\gamma(s)$ *is max , add s to* S_* where $s \in S_*$
7. end while
8. Turn on all nodes in S_* to process job requests
9. Call dynamic data placement algorithm
10. end while

As shown in Algorithm 2-2, in $WSC(U,S^{\#})$ *U* stands for the position of data blocks that job requests need in nodes , $S^{\#}$ stands for the position of data blocks in nodes.

The greedy algorithm of WSC is an approximation algorithm and proved the most possible to solve the set cover problem in polynomial time.

VI. EXPERIMENT

A. Experiment Setup

The experimental environment of this paper consists of four servers (Intel i5 2400 3.1GHz, 4G RAM, Linux 2.6.18). We measure resource utilization and power cost of each server with a power analyzer. A Voltech PM 1000+ power analyzer is used to measure server energy consumption of each part of the server. This paper uses Cloudsim toolkit to simulate cloud computing environment for evaluating the proposed approach in this paper.

B. Experiment Result

We compares the strategy that uses batch scheduling algorithm to optimize objective 1 with that does not use to optimize. The result is shown in Figure 1 and it verifies that the execution time of the strategy using batch scheduling algorithm is shorter than that does not use.

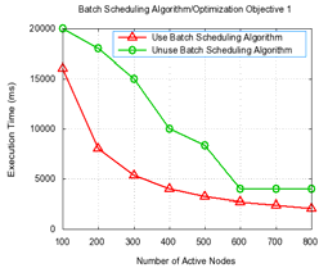


Figure 1 Result of Comparison Comparison.

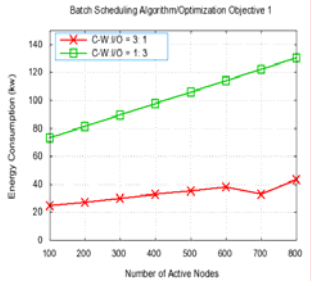
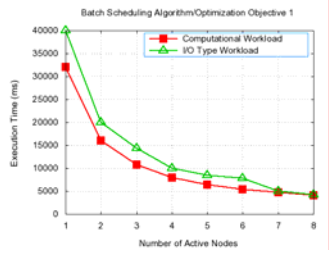


Figure 2 Load Energy Consumption

The energy consumption is different for different types of workloads. This paper tests energy consumption of the computational load and I/O load and sets the number of the computational load and I/O load to 3:1 and 1:3. U_{up} is set to 0.95 and U_{down} 0.05. Figure 2 shows energy consumption comparison results. This paper omits the energy consumption of the network and memory, considering only the energy consumption of the CPU and the storage systems. This result shows that the energy consumption of the I/O load is three times larger than that of the computational load, which is the same as the proportion of the number of two workloads.



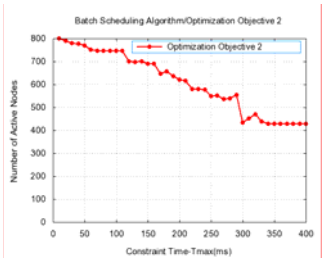


Figure 3 Execution Time Comparison Figure 4 Test Result of Optimization Object 2

The execution time is different because of the differences of energy consumption of different types workloads. In this paper, the number of the computational load and I/O load is set to 3:1 and 1:3 respectively with other settings remain fixed. Given the power upper limit, Figure 3 shows the result that the execution time of computational load is faster than that of I/O load. Specially, the execution time of I/O load is 1.25 times longer than that of computational load. As shown in Figure 4, this paper uses several T_{max} to optimize objective 2.

VII. CONCLUSION

This paper proposes dynamic data placement algorithm which effectively solve the problem of data placement in cloud computing platform, and this algorithm makes use of large optimization space for node scheduling strategies. The batch scheduling optimization strategy not only achieves energy-saving effect but also successfully solves the constrained problems which are power-restraint problem and time-constrained problem in cloud computing platforms. The time-restraint problem is abstracted as a weighted set cover problem. Finally, this paper uses Cloudsim toolkit to do a series of simulate experiments which shows this approach is effective in cloud computing platforms.

REFERENCES

[1] http://www.energystar.gov/ia/partners/prod_development/revisions/downloads/computer/RecommendationsTierICompSpecs.pdf

[2] MeikelPoess, Nambiar, Raghunath Othayoth. Tuning Servers Storage and Database for Energy Efficient Data Warehouses. IEEE, 2010, 1006-1017

[3] Koomey , J. G. Koomey. Estimating Total Power Consumption by Servers in the US and the World. 2007

[4] Stavros Harizopoulos, MehulA.Shah, Justin Meza . Energy Efficiency: The New Holy Grail of Data Management Systems Research. CIDR, 2009, January 4-7

[5] S.Worth. Green stroage - the big picture. Storage Networking World. 2011

[6] E. Pinheiro, R. Bianchini. Energy conservation techniques for disk array based servers. ACM, 2004, (26):68–78

[7] E. Otoo, D. Rotem, S.-C.Tsao. Dynamic Data Reorganization for Energy Savings in Disk Storage Systems. InSSDBM, (6187) :322–341, 2010.

[8] Norifumi Nishikawa, Miyuki Nakano, Masaru Kitsuregawa. Energy Efficient Storage Management Cooperated with Large Data Intensive Applications. ICDE, 2012

[9] Nitesh, Maheshwari, Radheshyam. Dynamic Energy Effcient Data Placement and Cluster Reconfguration Algorithm for MapReduce Framework. FGCS , 2011

[10] Willis Lang, Jignesh M. Patel. Towards Eco-friendly Database Management Systems. 2009

[11] Andreas Beckmann, Ulrich Meyer Peter, Sanders. Energy-Efficient Sorting using Solid State Disks. DFG grant SA 933/3-2, 2010.

[12] Elnozahy, E.N., Kistler, M. Energy conservation policies for web servers. In: USITS, 2003

[13] Elnozahy, Kistler, Rajamony. Energy-efficient server clusters. Springer, 2003. 179–196

[14] Rao L, Liu X, Xie L. Minimizing electricity cost: Optimization of distributed internet data centers in a multi-electricity-market environment. In: INFOCOM , 2010

[15] Pengcheng Xiong, Yun Chi, Shenghuo Zhu . Intelligent Management of Virtualized Resources for Database Systems in Cloud Environment. ICDE, 2011

[16] Willis lang, Jignesh M.Patel, Jeffrey F. Naughton.On. Energy Management, Load Balancing and Replication. SIGMOD, 2009

[17] Bo Li, Jianxing Li, Jinpeng Huai. EnaCloud : An Energy-saving Application Live Placement Approach for Cloud Computing Environments. IEEE, 2009

[18] Anton Beloglazova, Jemal Abawajyb, Rajkumar Buyyaa. Energy-aware resource allocation heuristics for efficient management of data centers for Cloud computing. Future Generation Computer Systems, 2012