Journal of Computers Vol. 30 No. 4, 2019, pp. 57-62
doi:10.3966/199115992019083004005

Virtual Machine Placement Algorithm for Minimizing Run Time in Cloud Environment

Chia-Cheng Hu[1,2*], Zhong-bao Liu[2], Su-Zhen Ge[2], Hong-Bo Zhou[2], Chong-Jie Zhang[2]

[1] College of Information Engineering, Yango University, Fuzhou 350015, Fujian, China

[2] School of Software, Quanzhou Institute of Information Engineering, Quanzhou 362000, Fujian, China
cchu.chiachenghu@gmail.com, zhongbao@hotmail.com, 592260817@qq.com, zhb591111@163.com, 475488008@qq.com

Received 29 September 2017; Revised 10 October 2017; Accepted 10 October 2017

Abstract. As enterprises store more data, cloud computing has emerged as a powerful and popular paradigm on processing and analyzing the large-scale data. The cloud consumers can access the computing resources through cloud technologies and build their own computing platforms on virtual machines (VMs). The VMs will be placed on the hardware computing resources provided by the Cloud. An important problem is to allocate the VMs to the physical computers in an efficient way. For this issue, MapReduce has emerged as the leading platform to scale-out to large clusters of machines capable of processing PBytes of data. One of the biggest challenges from the perspective of the cloud provider is to offer this MapReduce service in the cloud effectively. Most cloud providers are focusing their attention on the runtime efficiency of the computers in the Cloud. The majority of studies converted the problem of mapping VMs to physical computers to be a Bin-packing problem, which is NP-Complete. In this paper, we propose algorithms for solving the problem of allocating VMs to physical computers with minimum run time. The problem is formulated as a 0/1 integer linear programming (0/1 ILP). Then, a rounding algorithm is proposed for obtaining a feasible solution.

Keywords: 0/1 integer linear programming, mapreduce, RFID cloud computing, virtual machine,

1 Introduction

In Cloud computing, a pool of hardware computing resources is available to users (called cloud consumers) via the Internet [1]. Computing resources, e.g., processing power, storage, software, and network bandwidth, are represented to cloud consumers as the accessible public utility services. The cloud consumers can access the computing resources through cloud technologies without detailed knowledge on the underlying infrastructure, and build their own computing platforms on virtual machines (VMs). Finally, the VMs are placed on the hardware computing resources provided by the Cloud.

An important problem is to manage the hardware computing resources of the Cloud for satisfying the cloud consumer requirements in an efficient way of allocating the VMs to the physical computers. For this issue, MapReduce [2] has emerged as the leading platform with seamless ability to scale out large clusters of machines in order to be able of processing PBs of data. It is a framework for processing parallelizable problems across huge datasets using a large number of computers, collectively referred to as a cluster. It breaks a user computation into small tasks that run in parallel on multiple computers, and scales these tasks to the cluster. Pioneered by Google and popularized by the open-source Hadoop [3], a large number of enterprises including technology and Internet companies as well as traditional businesses like retail [4] and pharmaceutical research [5] have used MapReduce for business analytics.

One of the biggest challenges from the perspective of the cloud provider is to offer this MapReduce service in the cloud effectively. Most cloud providers are focusing their attention on the runtime

[*] Corresponding Author

efficiency of the computers in the Cloud. The runtime duration of a distributed job in the Cloud depends on the size of data it processes and the amount of VM resources allocated to it. The duration of a computer is determined by the longest runtime of the VMs running on the computer. If the runtime of most VMs is much shorter than the runtime of the longest one, the runtime efficiency of the computer becomes low.

In this paper, we focus on allocate VMs to physical computers with minimum runtime. The problem is formulated as a 0/1 integer linear programming (0/1 ILP). Then, a rounding algorithm is proposed for obtaining a feasible solution. Simulation is also executed to evaluate the performance of the proposed algorithm.

2 Related Works

In [6], the authors used nature-inspired method to propose Ant Colony based workload placement algorithm, but the algorithm had high complexity and slow convergence speed. In [7], the authors proposed a placement algorithm using VM migration to enhance more space saving of physical servers, but this placement approach may lead to local optimum result and more migration steps. Strategies proposed in [8] and [9] emphasized the cost of migration and analyzed whether the energy saving after migration could offset the cost of migration process. In [10], a placement strategy making spatio-temporal tradeoffs for VMs was proposed, but it was only applicable to the environment where jobs arrive at the same time. If the jobs arrive dynamically and continuously, it was not applicable.

In [11-12], the authors evaluate the power consumption of data centers by using MapReduce. In [13], how the impact of different parameters of a Hadoop job is studied. In [14], the authors propose a new data layout by turning off some of the computers for saving energy. In [15], the authors adopt a strategy, where incoming jobs are batched and all computers are used to store the data and execute jobs after the entire cluster can be suspended.

3 ILP and Rounding Algorithm

In this section, a formal definition of the problem of allocating VMs to physical computers with minimum runtime will be given. The following notations in the problem formulation are adopted:

- The set of VMs is denoted by V and $|V| = v$;
- The set of physical computers is denoted by P and $|P| = p$;
- Let t_j to denote that the runtime of VM j, where $1 \le j \le v$.

Prior to the problem formulation, the following decision variables are defined:

- $x_{i,j}=1$ ($x_{i,j}=0$) is used to denote that VM j is (is not) assigned to group j;
- T is used to denote the total runtime among all physical computers.

The objective is to minimize the total runtime among all physical computers, i.e., to minimize T. In the assignment, each VM j is required to be assigned to exactly one physical computer, i.e., $\sum_{j=1}^{n} x_{i,j} = 1$. Therefore, constraint (1) is induced. On the other hand, $\sum_{j=1}^{v} t_j x_{i,j}$ is the total runtime of physical computer i for executing all VMs j assigned to physical computer i. Since all $\sum_{j=1}^{n} x_{i,j} = 1$. should be smaller than T, the constraint (2) is induced. The 0/1 ILP formulation is as follows.

Minimize T

Subject to

$$\sum_{j=1}^{n} x_{i,j} = 1 \text{ for } 1 \le j \le v \qquad \textbf{(1)}$$

$$T - \sum_{j=1}^{v} t_j x_{i,j} \geq 0 \text{ for } 1 \leq i \leq p \text{ and } 1 \leq j \leq v \quad \textbf{(2)}$$

$$x_{i,j} \in \{0, 1\} \text{ for } 1 \leq i \leq p \text{ and } 1 \leq j \leq v \quad \textbf{(3)}$$

If $x_{i,j} \in \{0, 1\}$ are relaxed to $0 \leq x_{i,j} \leq 1$, then an LP results, which is polynomial-time solvable. Suppose that $x^*_{i,j}$ is the optimal solution to the LP, where $1 \leq i \leq p$ and $1 \leq j \leq v$. In the following, we present a rounding algorithm that can round $x^*_{i,j}$ to $x'_{i,j}$, where $x'_{i,j} \in \{0,1\}$ is a feasible solution to the 0/1 ILP. Initially, we set $x'_{i,j} = x^*_{i,j}$ for all $1 \leq i \leq n$ and let $X = \{ x^*_{i,j} \mid 0 < x^*_{i,j} < 1\}$.

The rounding algorithm is executed iteratively until X is empty. In each iteration, a $x^*_{k,j}$ is selected if the increment $\Delta_{k,j}$ of the objective is minimum, where $x^*_{i,j} \in X$. The increment $\Delta_{k,j} = \Delta^*_{k,j} - \Delta^-_{k,j}$, where $\Delta^+_{k,j}$ is the increment amount of T by setting $x^*_{i,j} = 1$, and $\Delta^-_{k,j}$ is the decrement amount of T by setting all $x^*_{i,j} = 0$ for $1 \leq i \leq p$ and $i \neq k$. The rounding algorithm is presented below.

```
(1)  sset $X = \{ x^*_{i,j} \mid 0 < x^*_{i,j} < 1\}$;
(2)  for each $x^*_{i,j} \in X$ do
(3)      set $\Delta^+_{k,j}$ to be equal to the increment amount of $T$ by if setting $x^*_{i,j} = 1$;
(4)      set $\Delta^-_{k,j}$ to be the decrement amount of $T$ by setting all $x^*_{i,j} = 0$ for $1 \leq k \leq p$ and $i \neq k$;
(5)      $\Delta_{i,j} = \Delta^+_{i,j} - \Delta^-_{i,j}$;
(6)  determine $\Delta_{k,j} \in X$ is minimal;
(7)  set $x^*_{i,j} = 1$;
(8)  set all $x^*_{i,j} = 0$ for $1 \leq i \leq p$ and $i \neq k$;
(9)  delete all $x^*_{i,j}$ from $X$ for $1 \leq i \leq p$;
(10) If $X$ is not empty, go to (1).
```

4 Simulation and Performance Analysis

We simulate our algorithms in a large cloud datacenter with various MapReduce jobs to be run and allows for VMs to be placed on physical servers. After all VMs on a particular server have finished, we simulate the server being powered off or hibernated by the cloud service provider.

Workload demand on the system depends on the parameters of the MapReduce jobs submitted. Specifically, the parameters that impact the nature of the workload include: (1) Number of jobs, (2) Minimum number of VMs required for each job, (3) Type of VM required for each job, (4) Estimated completion time for each job. We explore a range of values for all these parameters, and in turn explore several workload mixes. The default configuration uses (1) 50 MapReduce jobs, (2) the minimum number of VMs required is uniform on [1, 10] with integer rounding, (3) a round-robin assignment is used to assign VM types to successive jobs, and (4) estimated completion times for each respective job is taken from a uniform distribution on [10, 100] minutes.

The environment is characterized by the following:

- Number of physical machines available,
- Resource characteristics of physical machines,
- Allowed resource characteristics of the virtual machines.

In our simulations, we make the following decisions, respectively:

- There are sufficient server resources to accommodate all MapReduce jobs running in parallel with their respective number of VMs required,

- We use a three dimensional normalized resource capacity model (e.g., CPU, memory, storage) where the resource capacity of each server along each dimension is 200 units by default, and
- We use seven VM types with pre-set resource configurations. To ensure a good mix of workloads for any particular configuration, we run 10 trials for each configuration and take an average.

We use the following three metrics to measure the efficiency of our placement algorithms:

- Machine uptime: the total time of the servers is up.
- Resource inefficiency: The amount of resources wasted on the server at the time of the initial placement.
- Time imbalance: The Difference in the runtimes of the first and the last VMs to finish on the server.

We compare our placement algorithm with random first-fit placement in which VMs randomly shuffled before a first-fit placement is performed, thus exploiting VM diversity.

Fig. 1 shows that our placement algorithm has less machine uptime than random first-fit placement. In fact, its machine uptime is 33% better than random first-fit placement. The reason for the efficiency of our placement algorithm can be seen from Fig. 2 and Fig. 3 which show that our placement algorithm has lower resource inefficiency and time imbalance than random first-fit placement.

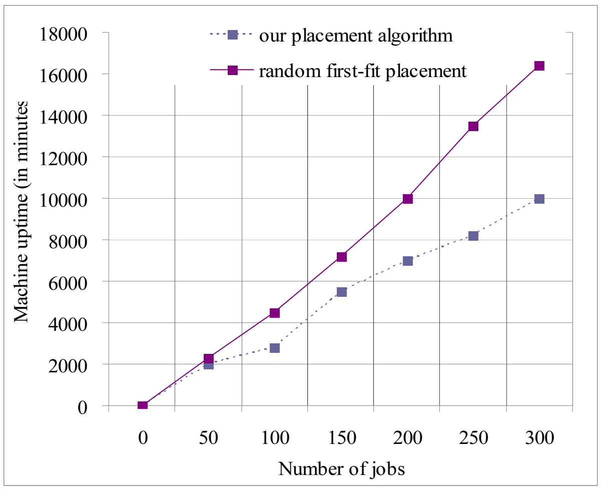

Fig. 1. Machine uptime

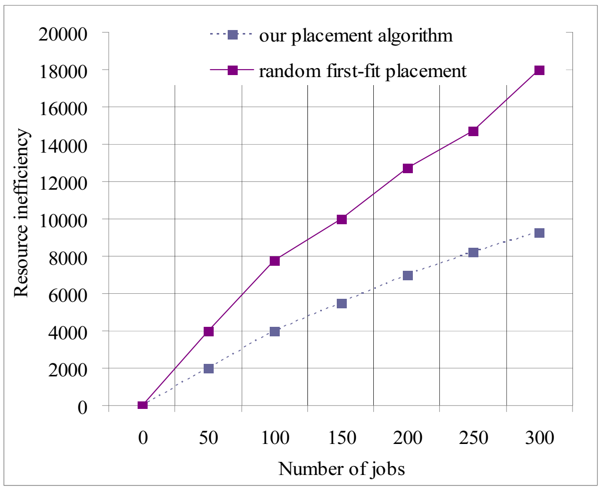

Fig. 2. Resource inefficiency

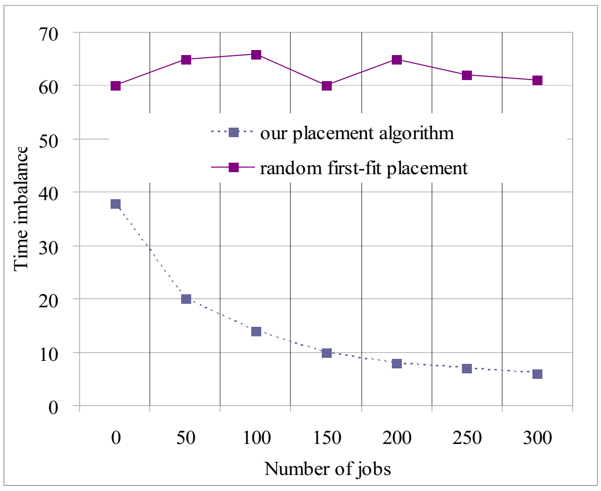

Fig. 3. Time imbalance

5 Conclusion

In this paper, we studied the problem of energy efficient MapReduce in a private cloud environment. Our techniques place MapReduce VMs within the cloud in a manner that is not only an efficient spatial fit, but also a balanced temporal fit. The results of Fig. 1, Fig. 2, and Fig. 3 show that our placement algorithm successfully exploits both resource and time to achieve significant energy savings of physical computers based on two key principles. First, we allocate VMs with similar runtimes to a server such that the server runs at a high utilization throughout its uptime. Second, while the VMs with similar runtimes are allocated to the server, their complementary requirements (CPU, memory, storage and network) are also considered such that the available resources of the server are fully utilized.

Acknowledgements

This work was supported in part by the Fujian Provincial Key Laboratory of Cloud Computing and Internet-of-Thing Technology, China.

References

[1] I. Foster, Y. Zhao, S. Lu, Cloud computing and grid computing 360-degree compared, in: Proc. Grid Computing Environments Workshop, 2008.

[2] J. Dean, S. Ghemawat, MapReduce: simplified data processing on large clusters, Commucations of the ACM 51(1)(2008) 107-113.

[3] Hadoop, <http://hadoop.apache.org>, 2012.

[4] AsterData, <http://www.asterdata.com/customers/barnes-andnoble>, 2012.

[5] M. Dublin, <http://www.genomeweb.com/informatics/gothadoop>, 2012.

[6] E. Feller, L. Rilling, C. Morin, Energy-aware ant colony based workload placement in clouds, in: Proc. 12th IEEE/ACM International Conference on Grid Computing, 2011.

[7] B. Li, J.X. Li, J.P. Huai, T.Y. Wo, Q. Li, L. Zhong, EnaCloud: an energy-saving application live placement approach for cloud computing environments, in: Proc. Cloud Computing, 2009.

[8] G. Jung, M.A. Hiltunen, K.R. Joshi, R.D. Schlichting, C. Pu, Mistral: dynamically managing power, performance, and adaptation cost in cloud infrastructures, in: Proc. Distributed Computing Systems, 2010.

[9] A. Verma, P. Ahuja, A. Neogi, pMapper: power and migration cost aware application placement in virtualized systems, in: Proc. the ACM/IFIP/USENIX International Middleware Conference, 2008.

[10] M. Cardosa, M. Korupolu, A. Singh, Shares and utilities based power consolidation in virtualized server environments, in: Proc. 11th IFIP/IEEE Int'l Conf. Integrated Network Management, 2009.

[11] K. Lim, P. Ranganathan, J. Chang, C. Patel, T. Mudge, S. Reinhardt, Understanding and designing new server architectures for emerging warehouse-computing environments, in: Proc. 35th Ann. Int'l Symp. Computer Architecture, 2008.

[12] Y. Chen, L. Keys, R. Katz, Towards energy efficient MapReduce, Technical Report UCB/EECS-2009-109, Univ. California, 2009.

[13] J. Leverich, C. Kozyrakis, On the energy (In) efficiency of hadoop clusters, in: Proc. SOSP Workshop Power Aware Computing and Systems (HotPower), 2009.

[14] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, A. Warfield, Xen and the art of virtualization, in: Proc. ACM Symp. Operating Systems Principles (SOSP), 2003.

[15] Scheduling in Hadoop, < http://www.cloudera.com/blog/tag/scheduling/>, 2012.