A novel task deployment approach based on graph theory for power saving

Gaochao Xu[1], Peng Liu[1], Xiaodong Fu[1], Yunmeng Dong[1], Jia Zhao[2], Yan Ding[1]*

[1] *College of Computer Science and Technology, Jilin University, Qianjin Str. 2699, 130012 Changchun, China*

[2] *College of Computer Science and Engineering, ChangChun University of Technology, Yan'an Str. 2055, 130012 Changchun, China*

Received 1 April 2014, www.tsi.lv

Abstract

With the increasing of the big datacenter, the power consumption seems to be another overhead except the equipment cost. Saving the power of big datacenter is the hotspot now. In this paper, we proposed TA-BG algorithm based on the linear weighted and graph theory to speed up the execution of tasks. Firstly, utilizing linear weighted to execute first filter to reduce the searching scope for the next research. Secondly, seeking out the hosts that can execute tasks fast based on graph theory. Finally, placing the host on the hosts selected above. The experiments indicate that TA-BG can save power of datacenter by reducing the executing time. Besides, the TA-BG even performs well on load balance.

Keywords: Cloud Data Centre, Task Allocation, Power Saving, Graph Theory

1 Introduction

Cloud computing can provide the users infrastructure, platform, and software in the form of service. Users can obtain the service as needed conveniently by the network. The cloud data centre provides on-demand compute and storage resources for the users. With the development of cloud computing, the number of big data centre is increasing. Meanwhile, the data centre needs to execute the tasks submitted by the users, and respond to the users quickly to guarantee the good user experience. Because the number of physical host is large in the cloud data centre to provide services and the amount of remaining resources are changing constantly. Therefore, how to allocate the tasks, which comes within an internal, is a research hotspot. The allocation policy will have influences on the speed of executing tasks, overhead, and load balancing etc.

Nowadays, cloud data centres often utilize the algorithm of random allocation (RA) or optimum allocation (OA) to deploy tasks to the data centres. The RA allocate the tasks to the physical hosts randomly, the advantages are getting a fast allocation and may have the lowest overhead and highest load balance in a short time. However, after a long time, the data centre can`t perform well. The OA is the most used algorithm to allocate tasks in the data centre now. The algorithm selects the hosts, which have the most remaining resources to execute the tasks. The algorithm can get a fast execution, and better load balance. However, it often leads to high overheads and resource waste.

In this paper, aiming to propose an algorithm to find the allocation policy, which can make the data centre execute tasks fast, reducing the overhead of the data centre. Certainly, to make the data centre fast, low overhead in the long period, a high efficient and reasonable method to allocate the tasks to the resource pool is needed. The proposed allocation algorithm can finish the work of selecting the best physical hosts for the tasks efficiently. Especially, when used in large scale data centres, it performs well.

The rest of the paper is organized as follows. In Section 2, we present the related work about task allocation algorithms of data centres. In Section 3, present the problem the proposed algorithm solved and model of the problem. In Section 4, firstly describe the structure of the algorithm, and then state the idea of the proposed algorithm, give the detail procedure of the algorithm. In Section 5, the experimental results and analysis on CloudSim platform are given. Finally, in Section 6, we summarize the full paper and future work is put forward.

2 Related work

As far as we know, some task allocation algorithms [14, 15] are proposed and they almost stress all the goals of the task allocation. Some pay focus on the overhead of the datacenters, some stress load balance of datacenters and some aim to have a better exposed service. The algorithms that pay attention to low overhead include

*___Corresponding author___ e-mail: dingyan11@mails.jlu.edu.cn using the idea of constraint programming (CP) [1, 7] and the allocation based on forecast.

CP is a popular idea to solve the problem of task allocation [2, 5-7]. [2] adopts the idea of CP, thinking over a variety of constraints comes when allocate tasks. Furthermore, formulating the constraints and converting them to restricted condition of the objective function which make it convenient to get the optimal allocation policy that satisfy constraints. [3] also adopt the idea of constraint programming, it divided the task allocation into two phases, the first phase confirm the number of the host needed by the task using the idea of CP, the second phase also utilizing the CP to allocate the task to the physical host. [4] assume the traditional task allocation algorithm, which is based on forecasting, it forecast the load of the datacenter according to the history based on which they confirm the amount of the host, which should work. Then allocate the tasks to the fixed number of host which is got above. Because of the surplus host can be halt, it can reduce the overhead.

Some allocations with hope of making the load balance. Related algorithms often divided into static load balance algorithm and dynamic load balance algorithms. The static allocations [8-10] often use Round Robin, Weighed Round Robin, Weighted Least-Connection Scheduling and Least-Connection Scheduling. [8] Wei Qun adopt the Weighted Least-Connection Scheduling, that is, show the host`s performance with different weight and allocate the tasks to the hosts which has the lowest ratio of amount and weight. These static algorithm only utilize some static information, they cannot adapt to the dynamic load variation of the datacenter efficiently. Dynamic load balance algorithm [11, 12] is a classic combined optimization problem and is proved to be NP hard. In addition, it often incurs extra communication overhead during the procedure of balancing the load dynamically. Nowadays, the best algorithm to solve dynamic load balance seems to be greedy algorithm. In [13], Lau combined heavy-load preferred and light-load preferred and proposed an adaptive load-dispatching algorithm, reducing the communication overhead during balancing the load.

3 Proposed problem and its formulation

In the data centre, a group of tasks need to allocate to the host in the data centre to deal with. However, the number of host in the data centre is large, and this will certainly lead to some different kinds of allocation policy. Furthermore, what the policy is adopted will bring the data centre diverse cost. Therefore, to find a high-efficient, lower-cost task allocation policy is necessary.

We now formulate the problem of allocating n tasks onto *m* physical hosts. Its solution can be represented by an n dimension of solution vector, each element of which denotes the target host of the tasks. We defined as follows: H is a set of m available physical hosts denoted by H (h, t) = {h_1, h_2, h_3, …., h_m}, available at time t, each host have the cost denoted by c(*s*) (s∈{1, 2, 3, …, m}). *R* represents a set of resource, including CPU, memory, disk resources. u_{rj}(*r*∈*R*, *j*∈*H*)b represents the amount of resource *r* in the host *j*. Besides, *A* is a set of the tasks which come to the data centre within ∆t, denoted by A = {a_1, a_2, a_3, …, a_n}. t_{ri} (r∈R,*i*∈*A*)n represents how many *r* resources the task *i* need. Define the 0-1 variable x_{ij}, when task *i* is allocated to the host *j*, x_{ij} = 1, otherwise x_{ij} = 0. Similarly, define the 0-1 variable y_j, when the host *j* have tasks y_j = 1, otherwise y_j = 0. Above all, the target function can be present as follows:

$$\min \sum_{j\in H} y_j c_j \,, \tag{1}$$

$$\text{s.t. } \sum_{i\in A}\sum_{j\in H} x_{ij} = n$$

$$\text{s.t. } t_{ri} \le u_{ri}\,,\ (r\in R, x_{ij}=1)$$

$$c(s) = \alpha \bullet CPU(s) + \beta \bullet Mem(s) + d(s)\,, \tag{2}$$

where the CPU(*s*) denotes the power consumption of host s, the unit is *kw/h*, it can be got from the time t's function.

The mem(s) denotes the power consumption of the memory of host *s*, the unit is *kw/h*, it is the function of one variable time t.

The d(s) denotes other cost, including depreciation cost, using cost and so on. It is determined by the data centre manager, each host has different value *d*. *α*, *β* denote the unit price of the power, the unit is *yuan* per *kw/h*.

The first constraint makes sure that all the tasks are allocated to the hosts and the second constraint confirms that host *i* must have enough resources when the task *j* is allocated to the host *i*. Finally, we will get the solution vector denoted by *TH*, *TH* = {s_1, s_2, s_3, …, s_n} (s_k∈H).

4 Description of the proposed allocation algorithm TA-BG

4.1 THE PROPOSED SYSTEM ARCHITECTURE

Figure 1 illustrates the system architecture of the data centre, in which it utilizes the TA-BG algorithm to allocate tasks. Task allocation controller consists of four parts: the part of task collection, task allocation, host information base and the core of the architecture, which is used for computing the allocation policy using TA-BG algorithm. The part of task collection is responsible for collecting the tasks from the internet, and cache them. *t* seconds later, it sends the set of tasks to the TA-BG controller. The TA-BG controller will compute the allocation policy combined the tasks and the information of hosts. At last, the optimal policy, which computed by TA-BG will send to the part of task allocation, the task is allocated to the target physical hosts.

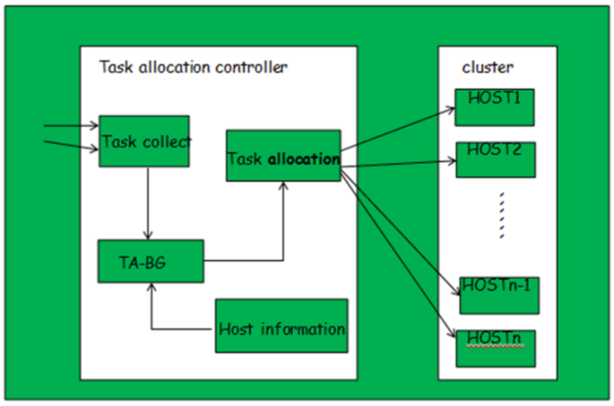


FIGURE 1 System Architecture of TA-BG

4.2 THE SOLUTION REPRESENTATION

By using the above system architecture, we can get the optimal allocation policy. In detail, since there are n tasks during one time of allocation, so the solution should be denoted by n dimensional vector, that is TH = {s_1, s_2, s_3, …, s_n} ($s_k \in H$). Every component of solution vector should be computed by the TA-BG algorithm, denote that which host the task is allocated to. For example, $s_1 = 2$ represent that task 1 is allocated to the host 2.

4.3 PROPOSED TA-BG ALGORITHM

The proposed TA-BG algorithm mainly utilizes the graph theory to solve the problem of task allocation of data centre. It can obtain an allocation policy with high efficient and low overhead. TA-BG is comprised of three steps. Firstly, filter hosts preliminary, filtering the hosts which even cannot execute the task with minimum resource in the purpose of reducing the number of next selection. Secondly, filter in the set that is got above. Here we use the concept of degree in the graph theory, regarding each host as a node and two nodes have edges only when they communicate each other. By doing so, the physical cluster will abstract to an undirected graph. Finally sort the hosts from large to small based on degree, then allocate the tasks from the beginning of the ordinal sequence.

4.3.1 Filter the hosts with few resources

The step is to filter the host that cannot execute any task roughly and quickly, to reduce the searching scope for the next step. So we take linear weighted to denote the resource of host j, that is the sum of CPU, memory etc. denoted by variable R_j^h.

$$R_j^h = \alpha \bullet CPU_j + \beta \bullet mem_j + \gamma disk_j , \tag{3}$$

s.t. $\alpha + \beta + \gamma = 1$.

where CPU_j denotes the surplus resources of host *j*, mem_j denotes the surplus memory of host *j*. *α*, *β*, *γ* is the weight factor, representing the importance of every variable and their sum must be 1.

To compare, we need to abstract the task demand to the similar variable. Denoted by R_i^t:

$$R_i^t = \alpha \bullet CPU_i + \beta \bullet mem_i + \gamma \bullet disk_i . \tag{4}$$

After the formulation, it`s convenient to compare, we need not to compare the every component of the vector respectively. Besides, we can utilize heap sort to find the task with minimum resources request, and then compare with each host`s surplus resources, weeding out the hosts that do not satisfy the condition $R_i^t \le R_j^h$. From this step, we can get the set *H1* preliminary.

4.3.2 Filter the hosts with better connectivity

Here we need to abstract the data centre to an undirected graph so that we can find the better connectivity host conveniently, so regarding each host in the cluster to a node and two hosts have an edge only when they have communicated each other. After the formulation, the connectivity of the host in the data centre can be represented by the degree of the nodes in the undirected graph. According to the experience and analysis, the host with better connectivity may have better performance. What`s more, the better connectivity hosts must be in the centre of the cluster that is near from controller of data centre. Moreover, because the host must communicate with controller when execute tasks. So using the better connectivity host to execute the task will certainly reduce the communication overhead. Now CPUs seems to be fast enough, but communication delay still have not enough progress because restrict of physical links. By doing so, we can reduce the communication overhead skilfully.

Sort the host in the H1 from large to small based on degree, and dispatch the tasks to the host with larger degree. We hold the principle that the task should be allocated to the host with lager degree as possible, the surplus hosts can be halt to reduce overhead.

5 Evaluation

In this section, we have experimentally verified the performance of the proposed TA-BG approach. The verification includes three aspects: the speed of executing tasks, the load balance and power consumption. Simultaneously, we have taken other two-allocation algorithm to conduct experiment, which also includes the same above three aspects to show the performance of TA-BG obviously. The first comparison experiment adopts RA to allocate the tasks, and the second comparison experiment takes OA to allocate the tasks. In experiment, we allocated 100 tasks to the data centre using the three-allocation algorithm separately and observe their performance.

To simulate a dynamic cloud data centre, we adopted the cloud simulation tools Cloudsim to conduct experiments. During the simulation period, Cloudsim can calculate power consumption by the getPower() method and view the condition of load. Cloudsim can create all kinds of entities and delete or add entities on the running machine dynamically, which is the reason why we choose the Cloudsim to conduct experiments.

5.1 EXPERIMENTAL SCENARIOS

In the Cloudsim platform, establishing a resource pool with 100 physical hosts and each host has the different number of resources. Simulating 100 task requests, all of them have disparate resource request like CPU, memory etc. By using the Cloudsim, we can get the information of the cloud data centre to further analysis.

5.2 COMPARISON OF RA, OA AND TA-BG IN EXECUTING SPEED

The experiment is designed for verifying the executing speed of the TA-BG algorithm. We firstly prepare 500 tasks with the same size. Allocating the 500 tasks to the cloud datacenters in succession, and recording the time every 100 tasks are finished. We conclude the followed bar graph according to the experiment:

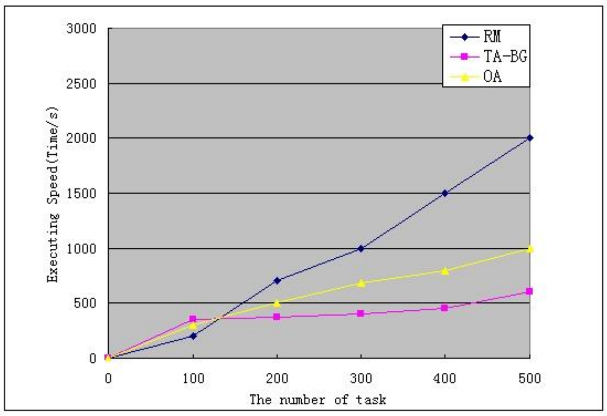


FIGURE 2 Comparison of Executing Speed

From the Figure 2 we can conclude that: the data centre executes the tasks fastest and be the most stable when taking TA-BG approach to allocate the tasks.

5.3 COMPARISON OF RA, OA AND TA-BG IN LOAD BALANCE

In the experiment scenario, we evaluate the TA-BG algorithm in load balance. We allocated 100 tasks to the host of datacentres that adopt three different allocation algorithms respectively. We recorded the condition of load on each host every ten minutes, and then calculated the variance. Here we only utilize the using rate of CPUs to represent the host`s load. $u_i$ denotes the current using rate of host i, m denotes the number of physical host. The average utilization of all hosts` CPUs is denoted by the formula (5) the degree of load balance denoted by the formula (6):

$$\bar{u} = \frac{\sum_{i=1}^{m} u_i}{m}, \tag{5}$$

$$B = \sqrt{\frac{1}{m} * \sum_{i=1}^{m} (u_i - \bar{u})^2}. \tag{6}$$

We get the curve chart according to the recode:

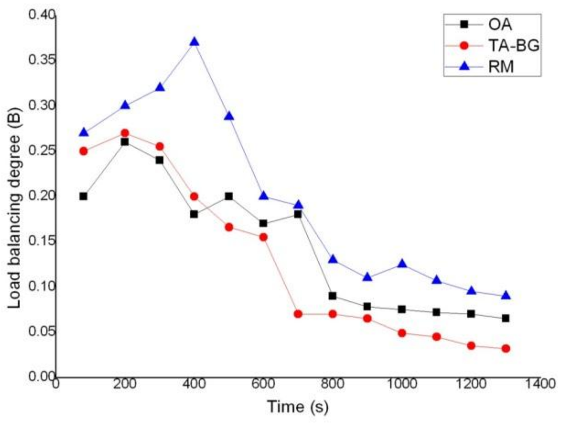


FIGURE 3 Comparison of load balancing degree

Small variance means the using rate of all host`s CPUs are closed. So small variance means the load is more balanced. Namely, the value B is the smaller the better.

From the Figure 3, when using the traditional RA algorithm the load balance is worse than other two algorithm from begging to end. At the beginning, the datacenter, which adopt OA algorithm is more balanced than the one adopt TA-BG algorithm. However, when t>420s the datacenter with TA—BG algorithm is better than them with OA algorithm on load balance. Therefore, we can conclude that the proposed TA—BG have good performance on load balance especially on the long-running datacenter.

5.4 COMPARISON OF RA, OA AND TA-BG IN POWER CONSUMPTION

In this experimental scenario, verifying the efficiency and availability of TA-BG in power saving. We also need three experiments, allocating 100 tasks to the different datacenters, the first one adopt RA to allocate tasks, the second adopt OA and the third use proposed TA-BG. We tested the power consumption every ten minutes, and draw the Figure 4.

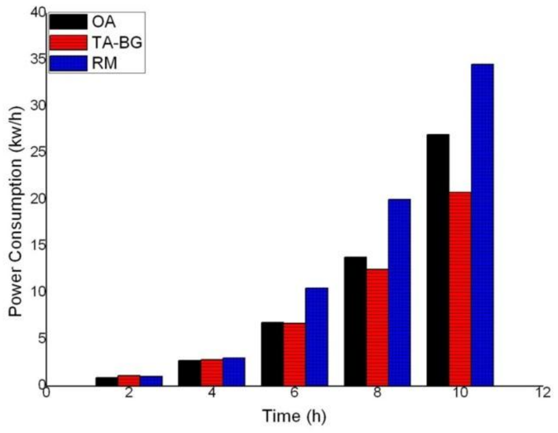


FIGURE 4 Comparison of energy consumption

As illustrated in Figure 4. When t>6h the power consumption is still lower than other two datacenters. And with the increase of the load in the cloud data center, the incremental power consumption of TA-BG is less than that of OA and RA obviously. And when t>6h the power consumption is still lower than other two datacenters.

6 Conclusion and future work

In this paper, an allocation algorithm of datacenter is proposed, and we give its idea, implementation and evaluation. It is based on linear weighted and graph theory. The first step, utilize linear weighted to filter the hosts with small resources. The second step, use graph theory to find the hosts that can finish the tasks as soon as possible. The third step, allocate the tasks to the hosts which are chosen from above steps. Finally, we conduct three teams of experiments, selecting OA and RA as the comparison experiment, to verify the performance of the TA-BG algorithm. They separately verify the advantages of TA-BG on executing speed, load balance and power saving. From above experiment, we can conclude that the TA-BG algorithm has better performance on several aspects than other allocation algorithm, especially fitting for the long-running datacenters. It is an available and efficient allocation algorithm for the real physical datacenter.

Aiming to further improve the performance of TA-BG, we hope to apply some other algorithm to allocate the task to the hosts selected by the first two steps instead of allocating them randomly. By doing this, it seems to be more predominant on all kinds of performance mentioned above like speed, load balance and power consumption.

References

[1] Müller T 1994 *Interactivity in Constraint Programming* DFKI of documentation series, German research center for artificial intelligence (dfki), stuh.

[2] Dhyani K, Gualandi S, Cremonesi P 2010 A Constraint Programming Approach for the Service Consolidation Problem *CPAIOR 2010* **LNCS 6140** 97–101

[3] Hermenier F, Lorca X, Menaud J-M 2009 Entropy: a Consolidation Manager for Clusters *VEE'09, March Washington, DC, USA*11-3

[4] Rolia J, Andrzejak A, Arlitt M F 2003 Automating enterprise application placement in resource utilities *DSOM 2003 LNCS* Eds Brunner M, Keller **2867** 118–29

[5] Hermenier F, Demassey S, Xavier L 2011 *CP 2011 LNCS* **6876** 27–41

[6] Sirdey R, Carlier J, Kerivin H, Dritan N 2007 *European Journal of Operational Research* **183** 546–63

[7] Fukunaga A S 2009 Search Spaces for Min-Perturbation Repair. In proceeding of: *Principles and Practice of Constraint Programming - CP 2009 15th International Conference, CP 2009 Lisbon, Portugal, September* 20-24

[8] Wei Qun, Xu Guangli, Li Yuling 2011 Research on duster and load balance based on LINUX virtual server *Information Computing and Applications. Berlin: Springer Heidelberg* 169-76

[9] Song S, Lv T, Chen X 2014 A Static Load Balancing algorithm for Future Internet *TELKOMNIKA Indonesian Journal of Electrical Engineering* **12**(6)

[10] Chen Wei, Zhang Yufang, Xiong Zhongyang 2010 Research and realization of the load balancing algorithm for heterogeneous cluster with dynamic feedback *Journal of Chongqing University* **33**(2) 2-14

[11] Willebeek-LeMair M H,Reeves A P 1993 Strategies for dynamic load balancing on highly parallel computers *IEEE Transactions on Parallel and Distributed Systems* **4**(9) 979-93

[12] You T, Li W, Fang Z 2014 Performance Evaluation of Dynamic Load Balancing Algorithms *TELKOMNIKA Indonesian Journal of Electrical Engineering* **12**(4)

[13] Lau S M, Lu Q, Leung K S 2006 Adaptive load distribution algorithms for heterogeneous distributed systems with multiple task classes *Journal of Parallel and Distributed Computing* **66**(2) 163-80

[14] Shen Kaiji, Zheng Xiaoying 2012 Fair multi-node multi-resource allocation and task scheduling in datacenter, *Cloud Computing Congress (APCloudCC), 2012 IEEE Asia Pacific* 59 – 63

[15] Pedram Massoud 2012 Energy-Efficient Datacenters *IEEE Trans. on Computer Aided Design* **31**(10) 1465 – 84

Authors

Gaochao Xu, born in 1966, Wuhan

Current position, grades: Changchun, Professor
University studies: BS, MS and PhD on College of Computer science and Technology of Jilin University in 1988, 1991 and 1995.
Scientific interest: Cloud Computing, Mobile Cloud Computing, distributed system, grid computing, cloud computing, Internet of things, information security, software testing and software reliability assessment
Publications: SCI 10
Experience: he is the professor and PhD supervisor of College of Computer Science and Technology, Jilin University, China. As a person in charge or a principal participant, Dr Gaochao Xu has finished more than 10 national, provincial and ministerial level research projects of China.

Peng Liu, born in 1990, Jixi of Heilongjiang province of China

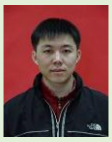
Current position, grades: a postgraduate student of the college of computer science and technology of Jilin University
University studies: the study of computer science at Daqing Normal University in 2009 and got his bachelor degree in 2013.
Scientific interest: Virtualization, Cloud Computing, Mobile Cloud Computing, SDN

Xiaodong Fu, Changchun

Current position, grades: Senior engineer in the College of Computer Science and Technology, Jilin University of China
University studies: B.Sc. degree from Jilin University
Scientific interest: Distributed System, Grid Computing, Cloud Computing, Internet Things, etc
Publications: 14 research articles

Yunmeng Dong, born in 1989, Yushu of Jilin province of China in, ChangChun

Current position, grades: Changchun, Master
University studies: studies of computer science at Changchun University of Technology in 2008, bachelor degree in 2012. Now he is a postgraduate candidate of the college of computer science and technology of Jilin University Virtualization
Scientific interest: Cloud Computing, Mobile Cloud Computing Virtualization, Cloud Computing, Mobile Cloud Computing
Publications: EI 1
Experience: Yunmeng Dong. He began the. His main research interests include distributed system, cloud computing and virtualization technology



Jia Zhao, born in 1982, Changchun of Jilin province of China, Changchun

Current position, grades: PhD candidate of the college of computer science and technology of Jilin University
University studies: master degree in 2008
Scientific interest: Virtualization, Cloud Computing, Mobile Cloud Computing
Publications: SCI 4
Experience: distributed system, cloud computing, network technology. He has participated in several projects.

Yan Ding, born in 1988, Yichun of Heilongjiang province of China, Tieli

Current position, grades: Changchun, Master
University studies: bachelor degree in 2011. Now he is a postgraduate candidate of the college of computer science and technology of Jilin University.
Scientific interest: Virtualization, Cloud Computing, Mobile Cloud Computing
Publications: SCI 1
Experience: He began the study of computer science at Jilin University in 2007 and got his degree