Multi-Resource Packing for Job Scheduling in Virtual Machine Based Cloud Environment

Daochao Huang, Peng Du, Chunge Zhu, Hong Zhang, Xinran Liu
National Computer network Emergency Response Technical Team Coordination Center of China (CNCERT or CNCERT/CC)
Beijing, China
huangdc@cert.org.cn

***Abstract*—To efficiently schedule jobs with highly diverse resource requirements along CPU, memory and bandwidth for job performance and resource utilization in a virtual machine based cloud environment, the multi-resource job scheduler is proposed to pack tasks to virtual machines under the notion of fairness and efficiency. Given the definition of job scheduling proportional fairness and utility function, the multi-resource job scheduling algorithm which fulfills capacity constraints of virtual machines is conducted. Comparative analysis illustrates our scheme improves average job completion time by preferentially grouping jobs that has different resource requirements. Compared to existing methods, multi-resource packing algorithm significantly improves the cloud system's resource utilization, yet with a substantial reduction of average job completion times.**

Keywords—job scheduling; multi-dimensional packing; completion time; cloud computing; fairness

I. INTRODUCTION

Today's cloud platform perform rich and complex tasks with the growing trend that many enterprises migrate their business or service to the clouds instead of building their own datacenters. Cloud service provider usually operates multiple data centers that are placed in different geographic regions to satisfy this increasing needs of cloud service, where cloud users request for different compute resources (i.e., CPU, memory, disk) in a pay-as-you-go manner[1]. The emergency of distributed cloud systems gives rise to problems deal with scheduling jobs upon request under the notion of efficiency and fairness. Specifically, requests arrive in real-time and resource placement decisions must be made when a request arrives, without any prior knowledge of future arrivals. Since the amount of resource is limited, only a subset of the demands can be serviced. How to select the most profitable subset and to grant service in a feasible way becomes a key optimal problem for a cloud service provider.

To place the tasks at the 'right place' so that not only the needs for the job are satisfied, but also the placement decision keeps enough room for admission of future jobs is the job of the scheduling algorithm to find the best virtual machine, in the appropriate physical machine and host enough resource. However, the heterogeneity of workload demands poses significant technical challenges on the resource allocation mechanism. Since different jobs are likely to request a variety of resource requirements, with different configurations in terms of processing capabilities, memory sizes, storage spaces and bandwidth demands. That is, depending on the underlying applications, the workload spanning multiple cloud users may require vastly different amounts of resources. For example, numerical computing jobs are usually CPU intensive, while database operations typically require high-memory support.

Towards addressing the inefficiency of the current allocation system, many recent works focus on increasing resource utilization while avoiding application performance degradation not only benefits cloud provider but also the tenants by improving job scheduling efficiency. Among them, some works [2][3] employ simple single resource abstractions. In these methods, a note is partitioned into slots with fixed fractions of resources, and allocate resources jointly at the slot granularity. However, the allocation of a single resource type is not enough, multi-resource allocation is needed to complete the job schedule. [4]-[8] are the literature presenting system investigation on the multi-resource allocation problem. They propose Dominant Resource Share (DRF) to equalize the dominant share of all jobs. In these works, both the efficiency and fairness notions have been guaranteed by capturing the trade-offs between allocation fairness and efficiency. For example, in the Bottleneck-Based Fairness (BBF) research [5], authors suggest two fairness properties that DRF possesses are also guaranteed. [6] considers another settings of DFR with a more general domain of user utilizations. [8], on the other hand, extend the all-in-one resource model to a multi-resource allocation model, called DRFH, that generalizes the notion of DRF from a single server to multiple heterogeneous servers. All these works assume, explicitly or implicitly, that all resources are concentrated into a cluster, which is not the case in the distributed system. Moreover, with the number of resources being allocated due to lack of task packing, resource fragmentation increase gradually under these resolutions described above.

This paper presents iVCE (Internet-based virtual computing environment, our cloud platform) scheduler, a multi-resource packing based scheduling system for large-scale distributed cloud platform, which has been deployed on iVCE clusters to schedule thousands of computations with millions of tasks running on tens of thousands of machines daily. The scheduler performs allocation decision via a multi-resource packing mechanism. Our contributions can be summarized as follows:

To achieve high-quality scheduling decisions, iVCE scheduler groups demands of tasks and schedules them together, which can reduce both the resource utilization and the average completion time as well as tail completion time for

978-1-4799-8356-8/15 $31.00 © 2015 IEEE
DOI 10.1109/SOSE.2015.30

typical iVCE applications by reducing the probability of demand conflicts between tasks.

To achieve multi-resource allocation under the notion of fairness and efficiency, a proportional fairness model is adopted to cooperate with multi-resource packing method. Thus, the packing process become the optimal task placement problem to maximize task utility.

To cope with the unexpected cluster dynamics and other abnormal runtime behaviors, multi-copy mechanism is adopted to guarantee the job results. When combined with iVCE scheduler, the mechanism improves the job execution success rate by concurrently running more than one replica of a task. If a task fail to obtain the expected results, its copy will be scheduled immediately to ensure the results.

The rest of this paper is organized as follows. Section II outlines the design of our job scheduler system, followed by the model and problem formulation in Section III. Simulation results are provided in Section IV. Finally, Section V concludes the paper.

II. THE DESIGN OF iVCE SCHEDULER

iVCE Scheduler serves as the underlying scheduling framework for iVCE, a distributed computation platform which supports large-scale internet-based computing for a variety of job needs. How to efficiently allocate capacity to jobs is a key problem in iVCE clusters which contains nearly one thousands of commodity servers, interconnected by a distributed network.

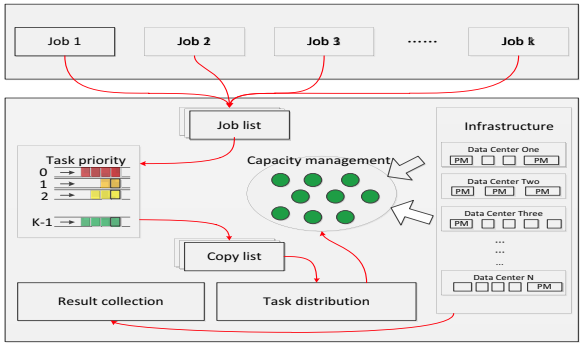


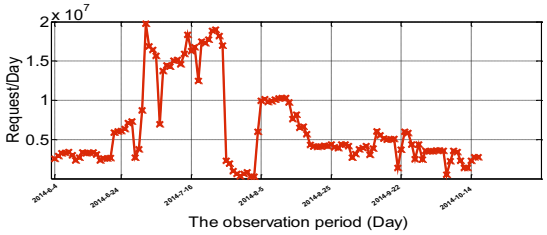
Figure 1 Execution graph of iVCE scheduler

Figure 1 show a common job processing graph in iVCE, when a job arrives at iVCE, the scheduling process involves the following responsibilities: (1) job list: maintain a list of jobs that are needed to be scheduled; (2) task priority: sort the application list according to the importance of the application. All tasks within the job use this priority, irrespective of when these tasks start or which part of the resource pool they use. iVCE platform assigns jobs with different priorities which is measured by the importance of the job. By generating job priorities in a task-aware fashion and using multiple queues mechanism, iVCE scheduler transforms the task-aware scheduling problem into the task prioritization problem. Typically, each job corresponds to a queue, and jobs with higher priority level enter the queue with higher sequence number, and give top priority for resource allocations. Moreover, different jobs with the same priority wait to be dispatched capacity according to the FIFO (First in First out) rule; (3) copy list: create a list of copies for each task, which is used to help failure recovery. In iVCE, if and only if the all tasks in a job are completed, the scheduling process repeat until the expected results are obtained; (4)capacity management: manage the resource capacity representing different points in the configuration space of resources such as processing, memory, storage and bandwidth; (5) task distribution: decide where to distribute a task and dispatch it to the selected server; (6) result collection: collect the runtime execution results before returning the results to users.

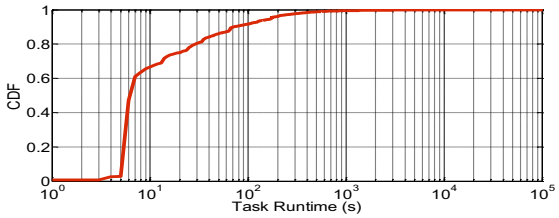
A. Task characterization

The distributed nature and scale of iVCE results in complex resource demand profiles. Different jobs are likely to request a variety of resource requirements, with different configurations in terms of processing capabilities, memory sizes, storage spaces and bandwidth demands. That is, depending on the underlying applications, the workload spanning multiple cloud users may require vastly different amounts of resources.

iVCE can support up to 20,000,000 tasks on a daily basis. At any point in time, there are thousands of tasks running concurrently. Those tasks vary in almost every dimension, to meet a wide range of Internet-based virtual computing scenarios and requirements. For example, numerical computing jobs are usually CPU intensive, while database operations typically require high-memory support. The workload diversity poses tremendous challenges for the underlying scheduling framework to deal with jobs under the notion of fairness and efficiency. As shown in Figure 2, we depict several task characteristics in iVCE environment to illustrate the diverse and dynamic nature of the computation workload.



(a) Statistic of scheduling requests



(b) Distribution of task runtime

Figure 2 Heterogeneous workload

We use data from June 4 to October 15 studies to characterize two features of application tasks in iVCE: 1) the number of tasks per day; 2) the task runtime. The two are critical when making scheduling decision; the first influences

the scheduling policy, while the latter determines whether iVCE job scheduling results are ensured, as the user expected.

From the observation shown in Figure 2(a), we see that the maximum of tasks taking by iVCE is up to 20,000,000 in a day. In addition, from Figure 2(b), task execution times range from less than 100ms to a few hours, as shown in Figure 2(a). 68% of tasks run for less than 10 seconds and 95% of tasks' runtime is less than 3 minutes, which means that except for some tasks that require external files such as configurations for their execution, most of the tasks can be completed quickly.

B. iVCE scheduler architecture

From task characterization described above, we found that applications running on iVCE have dynamic and diverse characteristics. On the one hand, tasks need varying amounts of each resource. On the other hand, demands for resources are weakly correlated. Existing schedulers usually allocate resource per slots or VMs, focusing on fairness and efficiency. However, with the number of resources being allocated due to lack of task packing, resource fragmentation increase gradually. At the same time, existing schedulers mainly focus on single resource such as CPU or memory, network usually are over-allocated. In iVCE, most of applications create Internet-based computing, bandwidth is often the bottleneck resource.

To solve resource fragmentation and over-allocated problem, supporting the scale and scheduling rate required for huge amount workload, iVCE scheduler adopts a distributed and multi-resource packing based architecture, where the scheduling of each job is performed independently and incorporates aggregated global cluster load information which includes task execution analysis and real-time resource utilization.

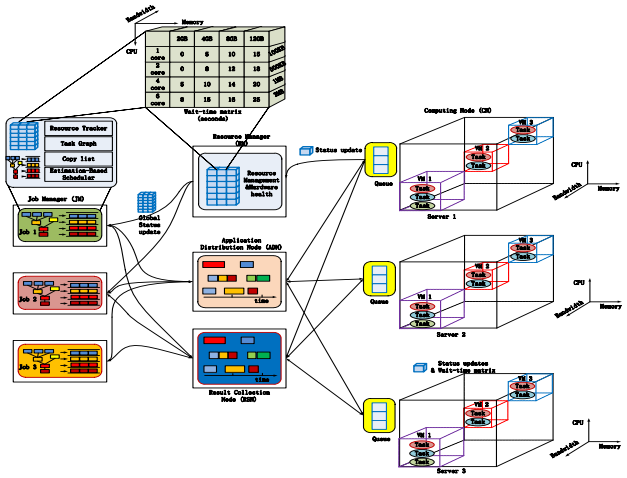


Figure 3 iVCE job scheduler architecture overview

Figure 3 provides an overview of our scheduler's architecture. We implement a set of modules including Job Manager (JM) , Resource Manager (RM) and Computing Node (CN) to work together to complete job scheduling in iVCE. Here, the JM is namely iVCE scheduler, which is designed to manage the life cycle of each job. The critical part of JM is task queue. To satisfy job diversity, multiple queues are employed in iVCE platform, in which each job is assigned to a queue with given priority. The global cluster load information used by each JM is provided through the cooperation of two additional entities in the iVCE framework: RM for each cluster and a CN on each server. A CN program running on each server is responsible for managing the local resources such as processing, memory, storage and bandwidth on that server and performing local scheduling, while the RM aggregates load information from CNs cross the entire iVCE clusters continuously, providing a global view of the clusters status for each JM to make informed scheduling decisions.

RM is implemented in iVCE to monitor dynamically changing state of a collection of distributed resources at a large scale. We implemented the RM in a master-slave configuration and its primary functions include resource registration, resource collection, resource aggregation, resource publication, resource provision and resource domain management. To better predict resource utilization in the near future and to optimize scheduling quality, each cluster in iVCE deploy one or several application distribution nodes (ADNs) to maintain the queue of tasks running in its domain, while each CN in the cluster maintains a local queue of tasks assigned to the server and advertises its future resource availability to the cluster's ADN, where a partial wait-time matrix is inferred from the queue. All of the partial wait-time matrices, together with the individual characteristics of tasks to be scheduled in the cluster, such as the location of inputs, are aggregated by the RM.

The ADN on each cluster manages a queue of tasks assigned to the servers in its domain in order to provide projections on future resource availability, while each CN on each server also maintains a queue of tasks assigned to the server to monitor local CPU, memory, storage and bandwidth usage. When a JM schedules a task on a cluster, the ADN of the cluster first sends a task-creation request with resource requirement and a list of files required to run the task (e.g., executables and configuration files). Then, it distributes tasks to selected servers according to our multi-resource packing algorithm described in the following section.

III. PROBLEM STATEMENT AND OBJECTIVE

A. Problem Formulation

In iVCE, the resource pool is composed of a cluster of virtual machines $V = \{v_1, v_2, ..., v_n\}$, each contributing d resources (e.g., CPU, memory, bandwidth). For each virtual machine i , let $c_i = (c_{i1}, c_{i2}, ..., c_{id})$ be its resource capacity vector, where each element c_{ir} denotes the total amount of resource r available in virtual machine i . We assume that the tasks running on iVCE is denoted by $TASK = \{s_1, s_2, ..., s_m\}$. Then, the set of tasks hosted by virtual machine $v \in V$ can be denoted by $TASK(v)$. Let $Load_v$ be the vector of resource requirements of task v , such as the vector of CPU, memory and bandwidth. The aggregate resource demands of tasks located at virtual machine v at time t are defined as $Load_v(t)$. The

resource demands of task s hosted by virtual machine v at time t are denoted as $Load_{vs}(t)$.

Let $y=(y_v, v\in V)$ denote a job scheduling approach satisfying proportional fairness [9] assuming that (a) job scheduling policy is feasible, namely, resources (such as CPU, memory, bandwidth, etc.) allocated to each task is less than the total capacity of the hosting virtual machine. (b) for any other alternative job scheduling approach $\tilde{y}=(\tilde{y}_v, v\in V)$,the following condition is satisfied: $\sum_{v:v\in V} w_p \frac{\tilde{y}_v - y_v}{y_v} \le 0$, where w_v is the weight of virtual machine performance or its willingness to offer resources. In other words, there is no other scheduling where all jobs are at least as well off and at least one job is strictly better off. This property ensures the allocation efficiency and is critical for high resource utilization. We call this scheduling scheme is Pareto optimal.

As we can see, proportional fairness depends on the difference between two scheduling approaches, which means that both fairness and efficiency are integrated in one model. Based on this definition, the objective function of the overall utilities can be denoted as:

$$\sum_{v:v\in V} U_v(y_v(t)) = \begin{cases} w_v \log y_v(t) & if \quad \alpha_v = 1 \\ w_v \dfrac{y_v^{1-\alpha_v}(t)}{1-\alpha_v} & if \quad \alpha_v \neq 1 \end{cases} \tag{1}$$

Here α_v is the indicator of fairness among a connection of virtual machines. When $\alpha_v = 1$, the utility function of virtual machine v at time t is given by $U_v(y_v(t)) = w_v \log y_v(t)$. In this paper, we consider proportional fairness model ($\alpha_v = 1$), then at time t, for all $v\in V$, the overall utilities can be denoted as

$$\sum_{v\in V} U_v(y_v(t)) = \sum_{v\in V} w_v \log y_v(t) \tag{2}$$

B. Muli-reource packing job schedulling

Typically, a well-justified scheduling should never give a job more resource than it can actually use in a virtual machine. Then under a truthful scheduling mechanism, the job scheduling objective function is given by maximizing the aggregate utilities of all virtual machines in the cloud data center.

Instead of allocating task to independent virtual machines, iVCE platform jointly packing a group of tasks with different dominant resource requirements into one VM to further achieve high resource utilization. In other words, multiple tasks can reside on the same virtual machine and each task occupies part of resources. Let $Load_{vs}(t)$ denote the load of task s which resides on VM $v\in V$, and $Load_v(t)$ denote the aggravate loads of VM $v\in V$, the following constraints must be satisfied: $Load_v(t) = \sum_{s:s\in TASK(v)} Load_{vs}(t)$. Here $Load(t)$ is defined as the d dimensional vector of resource requirements of task, when CPU, memory and bandwidth are considered, $d=3$, $Load(t)$ is given by $Load(t) = (Load^{CPU}(t), Load^{memory}(t), Load^{bandwidth}(t))$. Further, we define as the available capacity on VM $v\in V$ regarding its CPU, memory, bandwidth etc. To ensure that the total load on any virtual machine is less than or equal to its capacity, the following must hold:

$$\sum_{s:s\in TASK(v)} Load_{vs}(t) \le c_v \tag{3}$$

We study the problem of scheduling tasks on a set of virtual machines in this paper. Typically, tightly packing tasks onto a small number of servers and turning off other servers is an effective way to maximize resource utilization. However, concentrating workloads on a subset of the system resources can create hot spots, which may degrade task execution performance due to some bottleneck resource constraints. An effective strategy should consider tradeoffs efficiency and fairness. To reflect these two considerations, in our analysis, proportional fairness model is utilized to obtain the packing job scheduling $(\mathbf{P}_1)$:

$$(\mathbf{P}_1): \quad \max \sum_{v:v\in V} U_v(Load_v(t))$$

$$\text{Subject to} \quad \sum_{s:s\in TASK(v)} Load_{vs}(t) = Load_v(t), \forall v\in V \tag{3}$$

$$\sum_{s:s\in TASK(v)} Load_{vs}(t) \le c_v, \forall v\in V$$

$$\text{Over} \quad Load_{vs}(t) \ge 0, v\in V, s\in TASK$$

In order to facilitate the subsequent derivation of the formula, let $y_v(t) = Load_v(t)$. To maximizing the overall aggregate utilities of data center and obtain the optimal solution of $(\mathbf{P}_1)$, a Lagrange function is defined as:

$$L(x,y;\lambda,\eta) = \sum_{v:v\in V} \left\{ U_v(y_v(t)) + \lambda_v \left(\sum_{s:s\in TASK(v)} x_{vs}(t) - y_v(t) \right) \right\} + \sum_{v:v\in V} \eta_v \left(c_v - \sum_{s:s\in TASK(v)} x_{vs}(t) - \varepsilon_v^2 \right) \tag{4}$$

Where both $\lambda=(\lambda_v, v\in V)$ and $\eta=(\eta_s, s\in S)$ are Lagrange multiplier vectors , $\varepsilon^2=(\varepsilon_v^2, v\in V)$ is the relaxation factor vector. Let λ_{vs} denote the load requirements of task s. Let η_v be the available capacity of virtual machine s. Let the total resources occupied by all the tasks running on virtual machine v be denoted by $\sum_{s:s\in TASK(v)} x_{vs}(t)$. $\varepsilon_v^2 \ge 0$ will indicates there are remaining resources exist on virtual machine v.

Theorem 1: The packing job scheduling $(\mathbf{P}_1)$ is a convex programming problem, the optimal resources distribution

solution for each task v, i.e., $x_{vs}(t)$, exists, but not unique, while the total load requirements of virtual machines, i.e., $y_v(t)$, have unique solution.

Based on packing job scheduling (P_1), the derivation yields: $\partial L(x,y;\lambda,\eta)/\partial y_v(t)=0$, then

$$y_v(t)=w_v/\lambda_v \tag{5}$$

Such that:

$$\hat{L}(x;\lambda,\eta)=\sum_{v:v\in V}\left\{w_v\log\left(\frac{w_v}{\lambda_v}\right)-w_v+\sum_{s:s\in TASK(v)}x_{vs}(t)\lambda_v\right\} +\sum_{v:v\in V}\eta_v\left(c_v-\sum_{s:s\in TASK(v)}x_{vs}(t)-\varepsilon_v^2\right) \tag{6}$$

Let $\partial L(x;\lambda,\eta)/\partial\lambda_s=0$, then $\lambda_v=w_v\Big/\sum_{s:s\in TASK(v)}x_{vs}(t)$. Simplifying (6) with respect to λ_v, we get:

$$\hat{L}(x;\eta)=\sum_{v:v\in V}\left\{w_v\log\left(\sum_{s:s\in TASK(v)}x_{vs}(t)\right)\right\} +\sum_{v:v\in V}\eta_v\left(c_v-\sum_{s:s\in TASK(v)}x_{vs}(t)-\varepsilon_v^2\right) \tag{7}$$

Let $\partial L(x;\eta)/\partial x_{vs}(t)=0$, it is driven that $y_s(t)=\sum_{s:s\in TASK(v)}x_{vs}(t)=w_v/\eta_v$. Compared to (5), we get $\lambda_v=\eta_v$. It is concluded that, when $\lambda_v=\eta_v$, i.e., the load requirements of tasks are just equal to the availability capacity of virtual machines, the optimal unique solution of $(\mathbf{P}_1)$ is obtained .

C. Packing job scheduling Algorithm

From the above and in view of $(\mathbf{P}_1)$, a packing job scheduling algorithm is given. To maximize the utilization of virtual machines in cloud data centers is job schedule objective function. The pseudo-codes of our job scheduling scheme is captured in Algorithm 1.

Algorithm 1 JobScheduling

Require : Virtual machines $V=\{v_1,v_2,...,v_n\}$, tasks $TASK=\{s_1,s_2,...,s_n\}$, VM capacity $c_{v_i},v_i\in V$

1 : **for** each VM v_i in S

2: $V(v_i)\to\varnothing$; $U(v_i)_{\max}=\inf$;

3 : **for** each task s_j

4 : **if** $c_{v_i}\geq Load(s_j)$

5 : $TASK(v_i)=TASK(v_i)\cup\{s_j\}$; Update capacity of v_i;

6: **end if**

7 : **end for**

8: $Load(v_i)=\sum_{\forall s_j\in TASK(v_i)}Load(s_j)$

9: $U(v_i)=w_{v_i}\log(Load(v_i))$

10 : **if** $U(v_i)>U(v_i)_{\max}$

11 : $U(v_i)_{\max}=U(v_i)$;

12 : **end if**

13: **return** $V(v_i)$;

14 : **end for**

IV. EVALUATION

We evaluate multi-resource packing based job scheduling using the prototype implementation on our virtual machine based cloud environment, iVCE, which consists of 33 small data centers across the country. Each data center has 10~200 physical hosts according to the scale of the data center. Totally 512 hosts are deployed in the cloud, among which 467 hosts are new equipment that has 16 cores, 32GB memory, and 4TB storage. These heterogeneous servers provide a large scale resource pool which is further encapsulated into 5400 virtual machines. In this real world platform, each physical machine hosts a set of VMs, while each of VM supports up to 10 tasks running on it at the same time.

To evaluate our scheduler thoroughly, we use a combination of the following methods: 1) We compare our scheduling scheme to state-of-the-art scheduling algorithms implemented in Hadoop [10] and DRFH, a recent multi-resource fairness algorithm that is available. 2) We use trace-driven simulations on certain specific points in the multi-resource packing scheduler design to compare with alternatives. By representing these two groups of evaluations, we intend to answer the following questions: 1) How well does our algorithm improves average job completion time in a virtual machine based cloud environment? 2) How does our algorithm enhance the utilization of resource in dynamic cloud environment?

We first measured the aggregated scheduling rate in a cluster over time to understand iVCE's scalability. Figure 5 shows the peak scheduling rates for each day over the past 4 months. Since multiple task copies policy is adopted in job scheduling process, we find that the job concurrency and task concurrency are independent. The unexpected peak of job requests further confirms the need for a multi-resource packing based scheduling scheme.

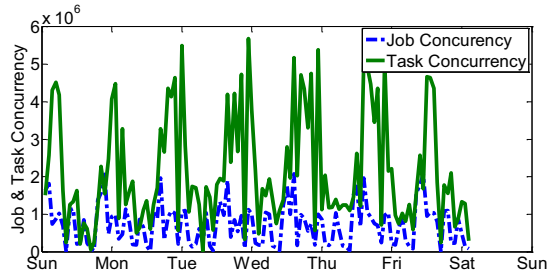


Figure 5 Concurrent jobs and tasks

To illustrate scheduling quality of the proposed method, Figure 6 compares the performance of the proposed multi-resource packing algorithm with traditional Slot based job scheduling and DRFH schemes. Each experiment runs our workload on the iVCE platform. We see that our job scheduling method improves job completion time, at most, by over 65%. Our scheduler significantly outperform the Slots scheduler and DRFH scheduler with much shorter job completion times, mainly because the Slots scheduler ignores the heterogeneity of both jobs and workload, and the DRFH scheduler does not consider the resource fragmentation problem due to lack of task packing.

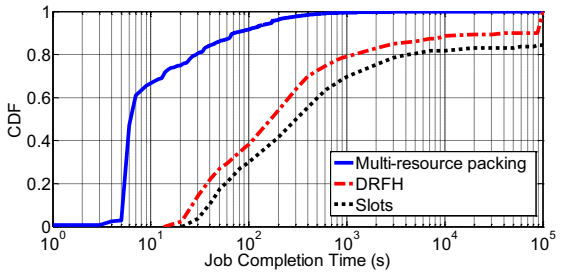


Figure 6 The proposed method improvements on job completion times over DRFH and Slots scheduler

We then study the resource utilization of the proposed algorithm. We take the 1,048,575 records of resource share (usage) in the terms of CPU during the past four months. We compare our method with the first-fit benchmark, the traditional job scheduler that schedules tasks onto the VMs by choosing the first VM that fits the task's resource requirements. For the latter, we try different VMs sizes and choose the one with the highest CPU and memory utilization.

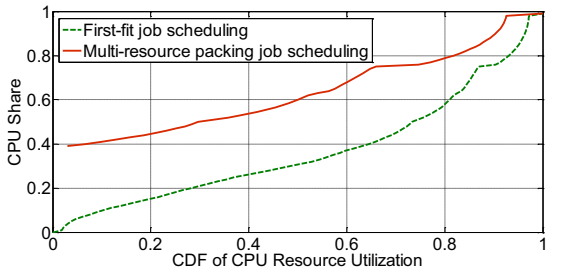


Figure 7 CPU utilization.

Figure 7 shows that the multi-resource packing based job scheduling method makes use of all the available resource and improves resource utilization, at median, by 20%. The top CPU usage improve by nearly 40%, compared to first-fit packing method. Mainly because the latter only focus on single resource allocation.

V. COCLUSIONS

In this paper, we study job scheduling schemes using multi-resource packing algorithm in a virtual machine based cloud environment where the resource pool is composed of a large number of virtual machines with different configurations in terms of resources such as processing, memory, and bandwidth. Our resolution provides a proportional fairness model to pack multi-resource to tasks with diverse characteristics. A prototype is implemented in a real-world system, iVCE, our large-scale distributed cloud platform. The large-scale evaluations show that, compared to the traditional Slot scheduler and DRFH model, the proposed algorithm can reduce job completion time, at most, by over 65%, and resource utilization, at median, by 20%, providing an efficient resolve for large-scale distributed cloud job scheduling.

ACKNOWLEDGMENT

The authors would like to thank the reviewers for their detailed reviews and constructive comments, which have helped improve the quality of this paper. This work was supported in part by National Key Basic Research Program of China (973 Program) under Grant No. 2011CB302605.

REFERENCES

[1] M. Armbrust, A. Fox, R. Griffith, A. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "A view of cloud computing," Commun. ACM, 2010, vol. 53, pp. 50–58.

[2] N. Jain, I. Menache, J. Naor, J. Yaniv, "A truthful mechanism for value-based scheduling in cloud computing," Theory of Computing Systems, 2014, vol. 54, pp. 388-406.

[3] M. Isard, V. Prabhakaran, J. Currey, U. Wieder, K. Talwar, and A. Goldberg, "Quincy: Fair scheduling for distributed computing clusters," in Proc. ACM SOSP, 2009, pp.261-276.

[4] A. Ghodsi, M. Zaharia, S. Shenker, and I. Stoica, "Choosy: Max-min fair sharing for datacenter jobs with constraints," in Proc. ACM EuroSys, 2013, pp. 365-378.

[5] C. Joe-Wong, S. Sen, T. Lan, and M. Chiang, "Multi-resource allocation: Fairness-efficiency tradeoffs in a unifying framework," IEEE/ACM Transactions on Networking (TON), 2013, vol. 21, pp. 1785-1798.

[6] D. Dolev, D. Feitelson, J. Halpern, R. Kupferman, and N. Linial, "No justified complaints: On fair sharing of multiple resources," in Proc.ACM ITCS, 2012, pp. 68-75.

[7] A. Gutman and N. Nisan, "Fair allocation without trade," in Proc. AAMAS, 2012, pp. 719-728.

[8] D. Parkes, A. Procaccia, and N. Shah, "Beyond dominant resource fairness: Extensions, limitations, and indivisibilities," in Proc. ACM EC, 2012, pp. 808-825.

[9] M. Chiang, S. H. Low, A. R. Calderbank, J. C. Doyle. Layering as optimization decomposition:A mathematical theory of network architectures, Proceedings of the IEEE, 2007, vol. 95, pp. 255-312.

[10] " Hadoop Fair Scheduler," http://hadoop.apache.org/docs/r2.2.0/hadoop-yarn/hadoop-yarn-site/FairScheduler.html