Efficient Resource Scheduling in Data Centers using MRIS

S. NAGENDRAM
Electronics and Computers Engineering
Asst. Professor, Koneru Lakshmaiah College of Engineering
Guntur, Andhra Pradesh, India
reena1286@gmail.com

J.VIJAYA LAKSHMI
Electronics and Computers Engineering
IV/IV B.Ttech, Koneru Lakshmaiah College of Engineering
Guntur, Andhra Pradesh, India
Y8em242@gmail.com

D.VENKATA NARASIMHA RAO
Electronics and Computers Engineering
IV/IV B.Ttech, Koneru Lakshmaiah College of Engineering
Guntur, Andhra Pradesh, India
narasimha.dvnr@gmail.com

CH.NAGA JYOTHI
Electronics and Computers Engineering
IV/IV B.Ttech, Koneru Lakshmaiah College of Engineering
Guntur, Andhra Pradesh, India
jyothiklu@gmail.com

Abstract

Scheduling the resources is decisive in shared data centers. Today scheduling algorithms focus only on one-dimensional resource models, infact the multiple resources (e.g. Memory, Storage, CPU and Network bandwidth) can be consumed concurrently. There is a lot of demand for multiple resources in cloud computing which allows various users to share a data center. This paper depicts resource scheduling problem to a bounded multi-dimensional knapsack problem, taking into account the requirement dependency among multi-dimensional resources. Due to the NP hardness of B-MDKP, we present Multi-dimensional Resource Integrated Scheduling (MRIS), an inquisitive algorithm to obtain the approximate optimal solution. MRIS will be highly efficient and achieves high performance for a various set of workloads.

***Keywords*: scheduling, bandwidth; multidimensional; optimal**

I. Introduction

Cloud computing is a new type of service which provides large scale computing resource to each customer. Creating effective scheduling systems is an extremely important factor in reducing costs, improving safety, and increasing efficiency in a variety of industries. It has revolutionized the computation paradigm. Cloud computing and shared data centers play an important role in supporting uncoordinated and heterogeneous users and their applications. Scheduling resources in data centers is crucial to efficient resource utilization and satisfactory application performance.

The existing schemes have numerous limitations in perspective of data centers. These schemes typically consider either a one-dimensional resource type or treat all resources as one abstract capacity when determining the sequence or combination of applications. These resource modeling does not does not focus on the fact that applications need a variety of resources (e.g. CPU power, memory, storage, network bandwidth and so on) simultaneously. In this paper we establish a multi-dimensional resource model and propose effective algorithms to solve the resource scheduling problem. Considering the multiple resources with limited capacities, we formulate the problem as a bounded multi-dimensional knapsack problem, an NP-hard problem even when the dimension is set to one.

For the execution and multiplexing of heterogeneous applications in a shared data center, we propose an efficient Multi-dimensional Resource Integrated Scheduling algorithm called *MRIS*. This heuristic algorithm is an approximation of progressive filling in which the usage of all resources are increased at a similar rate or one dimension after another. In other words, a shared-resource pool prioritizes jobs differently by their preferred resource types. To meet the needs of resource requirement we formalize a job schedule model considering the dependency among multi-dimensional resources. The jobs with dissimilar demands also works by using our novel heuristic for scheduling multiple resources is proposed which will run effectively on a shared data center to achieve nearly optimal resource utilization in each dimension.

II. Related Work

We should consider that problem of scheduling independent jobs has been typically formulated as the constraint satisfaction problem in many researches. These works mainly attempts to turn the issue into one-dimensional resource allocation problem via requirement abstraction. As, multi-dimensionality of job demands is more advantageous, and there will be a significant improvement in the integrated utilization by co-allocating the multiple resource type. In reality, jobs typically require diverse resources simultaneously. A VM placement scheme that handles the hierarchical and multi-dimensional resource constraints to avoid hotspots in a specific virtualized environment is already proposed. In an approach of considering multiple resources is explored in the grid computing environment using a dimensionless computation index to provide resources suited for a job.

In contrast, we attempt to address the scheduling issue by considering both the dominant resource requirements from applications and the scarcity of resources in the pool. Multiplexing of heterogeneous applications in a shared data center is studied in which is similar to our scenario. The difference between our work and theirs is that we explicitly aim at improving the resource utilization with the promise of ensuring an application’s basic performance, while they emphasizes more about the resource usage fairness among applications in multiple dimensions. A similar approach to model the resource allocation with multi-dimension constraints is the well-known vector packing problem. In vector packing assumes the same properties for each container (bin). However, this assumption does not hold under the shared-resource condition such as in data centers or clouds because virtual machines usually act as the ”containers” in a hardware sharing environment, whose sizes could be dynamically re-allocated using virtualization techniques.

III. Problem Definition

We first present an example and then giving a more precise model and formally describe the scheduling problem.

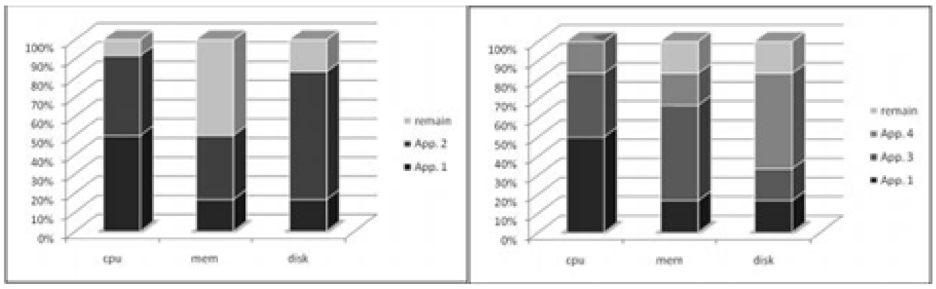
A. Example

We assume a shared resource pool consisting of 12 CPU cores, 12 GB memory, 12TB disk, and four application types. Each application submits a same kind of jobs which are independent of each other. The first application wants to use 6CPU cores, 2GB memory and 2TB disk per job, which represented as the demand vector $\langle 6; 2; 2\rangle$. The requirements of the next three applications are $\langle 5; 4; 8\rangle$, $\langle 4; 6; 2\rangle$ and $\langle 2; 2; 6\rangle$ respectively. Assume that these four jobs arrive sequentially.

We try to schedule jobs into the shared resources pool with two different strategies (S1 and S2).

***Schedule 1 (S1)**:* A traditional job scheduling typically treats the job set as a queue with a FIFO-like rule. The scheduler chooses the first two applications, but cannot add more because of the limited CPU capacity. The utilization of each kind of resource is illustrated in Fig.1 (a). We can find that although the CPU utilization is impressive, the memory use is only 50%.

Schedule 2 (S2): We combine the candidate jobs in accordance with their different resource requirements, and obtain the scheduling result shown in Fig. 1(b). Trying to reasonably co-locate applications with disparate resource demands, we can obtain high integrated system utilization up to 89% as three resource dimensions reach 100%, 83% and 83%, respectively.



(a)Schedule with traditional queue-based strategy (b) Schedule with awareness of multiple resource utilizations

Figure 1: An example of scheduling multiple resources.

The imbalanced utilizations among different resource types, as the result of Schedule 1, are caused by ignoring the associated relationship among heterogeneous job requirements. We pay more attention to the diversity of different resource vectors. That is to say, various application instances often have specific preference to their dominant resource. Selecting the appropriate applications to form a parallel job scheduling set provides great opportunity to improve the integrated resource utilization, since heterogeneous jobs could complement each other due to their different dominant resources. To this end, we propose a multi-dimensional resource aware mechanism to address this resource utilization problem.

B. Problem Definition and Model

***1) Problem Definition**:* We denote the resource capacity of a data center with *m*-dimensional resources as a vector $R = \langle r_1 : : r_m \rangle$, where *ri* denotes the total quantity of resource *Ri*. Assume that a data center supports *n* applications*AP*1: *:: APn*. To recall from Section III-A, each application submits the same kind of jobs which are independent of each other (if a user has different resource demands for different jobs, we model the jobs as different applications). Each application *j* has a demand vector w_J , which describes its resource requirement on per job. We refer to application *j's* job demand of resource *Ri* as $w_{i,J}$, thus the demand vector can be represented as $w_J = \langle w_{1,J}$ *: : :* $w_{m,J} \rangle$. With the number of jobs submitted of each applicationμ_J, *a* scheduling strategy defines an allocation *A* = $\langle$*x*1 *: : : xn*$\rangle$ to signify that application *APj* gets to run *xj* instances of job *j*, whose demand vector is *wj* . Given a data center capacity *R* and status of heterogeneous applications (w_J,μ_j), our problem is to derive a feasible allocation *A* to maximize the utilization in each resource dimension as well as to improve system performance with acceptable fairness tradeoffs.

2) Problem Model: The general job scheduling with multi-dimensional resource constraints is summarized as one of the particular combinatorial optimization problems. According to the application scenario and our problem statement, we map the multiple resource-aware job scheduling to a bounded multi-dimensional knapsack problem (B-MDKP)."Bounded" means that the amount of jobs submitted from each application instance are not infinite. In our model, we assume (1) Rigid job instance. Resource requirement for single-job instance of an application does not change during execution and (2) Non-preemptive Resources are locked for a job execution once the distribution of resources start. We summarize the symbols in Table 1. Note that we use the terms “job” and “item” interchangeably. The description of various symbols

Symbol	Description
r_i	The constraint of resource $1 \leq i \leq n$
p_j	the profit to schedule job j
μ_j	the total number of jobs of type j,$1 \leq j \leq n$
$w_{i,j}$	requirement of i_{th} resource submitted by j_{th} application type
I_j	The permitted number of jobs allocated in each scheduling iteration for application type j.

Table1. Model Symbol List

The original conception of profit *pj* represents the economic gain or use value of the given item. We borrow this term from into our model to express the system throughput benefit from each job with respect to its resource demands, which is not inherited the independency, between profit and size in the traditional knapsack problem. we employ $p_j$as inversely proportional to the Average resource consumption of a given job *j*, shown as the denominator part in Formula (1).

$$pj = \frac{1}{\frac{1}{m}\sum_{i=1}^{m}\frac{w_{i,j}}{c_j}} \qquad (1)$$

Similar to the traditional knapsack problem, the objective function is to maximize the total profit,

$$max \sum_{j=0}^{n} p_j * x_j \qquad (2)$$

With the constraints of resource capacities and item quantities.

$$\sum_{j=0}^{n} w_{i,j}x_j \leq rj, \;\; i = 1,2 \ldots..m \qquad (3)$$

$$x_j \leq \mu_j, \;\; j=1,2\ldots.m \qquad (4)$$

x_j: integer

IV. Resource Scheduling Algorithm

A. MRIS Algorithm

Even when the additional bounds of B-MDKP are not mentioned, it is known that the general multi-dimensional knapsack problem is strongly NP-hard, therefore, an effective heuristic algorithm with less-complexity is necessary. Since previous study concludes that greedy based heuristics dominate other methods both on execution cost and performance in multi-dimensional optimization, we develop novel heuristics to obtain the approximate optimal solution based on the traditional greedy algorithms of knapsack problems.

We try to redefine the efficiency metric (*ej*), a metric used in solving classical knapsack problems, by considering both the profit and the resource consumption of a job. When evaluating the resource consumption, we need to assess the scarcity of a resource so that we do not immediately schedule jobs which require resources of low availability. To achieve this, we first calculate the relationship between multi-dimensional resource supply and demand that shown as α_i in Formula (6), whereμ_j and r_i are the number of un-scheduled jobs and resource constraint of each dimension during current scheduling computation. The expression of α_i is to compute the over-requiring proportion of a certain resource type *i* from all the submitted jobs as a magnification coefficient, which indicates the degree of scarcity among various resource dimensions.

Then we multiply $wi;j$ by α_i to expand the influence of requiring more scarce resource. By using this product as the denominator, *ej* can dynamically adjust to prioritize jobs by selecting those requiring less scarce resources. From the view of the whole data center, our multi-dimensional scheduling is an approximation of progressive filling in which all resources' usage is increased at a similar rate or one dimension after another.

$$E_j=\frac{1}{\sum_{i=0}^{m} W(i,j)\alpha_j} \qquad (5)$$

$$\alpha_i = \frac{\sum_{j=1}^{n} w_{i,j}.\mu_j - r_i}{\sum_{j=1}^{n} w_{i,j}.\mu_j} \qquad (6)$$

The complete algorithm is explained in two stages, namely the logarithmic transformation and job selection. The transformation process provides an effective method to solve the bounded knapsack problem by converting it into an equivalent 0-1 form. And the job selection process is responsible for the combination and assignment of various application instances due to their different dominates resource. We first make a logarithmic transformation to generate an extension of 0-1 knapsack problem with limited increase in the number of variables. For each item-type *j* of B-MDKP, we introduce a series of $\lceil \log_2(\mu_J + 1) \rceil$ items, whose profits and weights are (p_J ; $w_{i,J}$), (2p_J ; 2$w_{i,J}$), (4p_J ; 4$wi;j$),. . . respectively, and one remaining item such that the profit (resp., weight) of the new items equals b_J *pj*(*resp:; b_J* $w_{i,J}$). Note that the efficiency metric e_j of each application instance does not change since the profits and weights are multiplied at the same magnification. In the following job selection process, we sort the new job sequence by non-decreasing order of each job's efficiency metric and non-decreasing order of the p_J value if their e_j are equal. Next, we employ a greedy-like strategy to pick up the first application instance with the highest efficiency metric, and determine whether the remaining resource constraints could meet the multi-dimensional resource demand. If the candidate job cannot be assigned due to the capacity limitation, we further check whether this job is the intermediate package during the previous logarithmic transformation process, and divide it into smaller package according to the current resource provided. Afterward, we repeat the division for all the jobs with the same highest e_J and compare the updated p_J values to determine which one should be scheduled at this iteration. Otherwise, we skip to the next group of jobs with smaller e_J in the waiting queue until no more job could be allocated. The resulting selection algorithm is shown in Algorithm.

B. Algorithm Complexity

First, the transformation process takes *O* (N$\log_2 N$) operations definitely, where *N* is the number of jobs. We mark the total number of transformed jobs as $\overline{N}$ and mainly focus on the job selection mechanism step by step. Assume that the initialization in *Step 1* and the final update in *Step 6* take negligible complexity. *Step 2&3* require *O* ($\overline{N}$ d + $\log_2 \overline{N}$) operations, while the inner loop at *Step 4&5* is invoked N times in the worst case and takes *O* ($\overline{N}$ d) basic operations. This implies that our algorithm is a polynomial time scheduling strategy with the computational complexity *O* (*N* $\log_2 N$ + $\overline{N}$ d ($\overline{N}$ d + $\log_2 \overline{N}$)).

Algorithm

(1). Initialize the upper bounds and the capacities for application instances.
(2). Do the computation on the efficiency metric of each job and sort them by non-decreasing order according to e_J and p_J. And then form a sequence *S*.
(3). Select the first job or a group of candidate jobs with the exact highest e_J value.
(4). Check whether the capacities of multi-dimensional resources satisfy the job's requirements, assign this job into data center; otherwise divide each job in this group according to the resource limitation.
X_i = $\boldsymbol{min_j} \left\lfloor {}^{\boldsymbol{c_j}}/_{w_{i,j}} \right\rfloor$, if X_i = 0 then skip the job.
(5). Again calculate the $p_J$values of the divided jobs in *Step 4*, $\acute{p}_j$ = *Xi* . p_j. Select the new job with the highest $\acute{p}_j$ value. If no job is chosen, remove these jobs from the sequence *S*, back to *Step 3* until the sequence is *NULL*.

6). From time to time update the capacities of resources and the upper bounds of application instances if allocation happened during iteration.

CONCLUSION

In this paper, the problem of efficient resource scheduling with multi-dimensional constraints in a shared data center was depicted. We have modeled the scheduling problem as a bounded multi-dimensional knapsack problem, and presented MRIS, a heuristic algorithm that approximates progressive filling. In such a way, MRIS enables jobs with dissimilar multiple demands to be run effectively in a sharing environment while optimizing resource utilization in each dimension. We demonstrate that MRIS achieves high system efficiency and application performance with an acceptable unfairness metric. In our future work, we plan to extend the model and MRIS to the fine-grained scheduling that introduce the task workflow in job execution stage.

REFERENCES

[1] Isard.M, Prabhakaran.V, Currey.J, Wieder.U, Talwar, and Goldberg.A, "Quincy: Fair Scheduling for Distributed Computing Clusters," in Proc. of the ACMSIGOPS 22nd symposium on Operating systems principles (SOSP), Oct. 2009.

[2] Khoo.B.B, Veeravalli.B, Hung, and C. W. S. See, "A multi-dimensional scheduling scheme in a Grid computing environment," Journal of Parallel and Distributed Computing, 2007.

[3] Merkle.D, M. Middendorf, and Schmeck.H, "Ant colony optimization for resource constrained project scheduling," IEEE Transactions on Evolutionary Computation, no. 4, 2002.

[4] Padala.P, K. Shin, X. Zhu, M. Uysal, Wang.Z, Singhal.S, Merchan,.A and Salem.K, "Adaptive control of virtualized resources in utility computing environments, "in Proc. of the 2nd ACM SIGOPS/EuroSysEuropean Conference on Computer Systems.

[5] "Parallel workload archive: Models," http://www.cs.huji.ac.il/labs/parallel/workload/models.html.

[6] Singh.A, Korupolu.M, and Mohapatra.D, "Serverstorage virtualization: Integration and load balancing in data centers," in Proc. of IEEE/ACM Supercomputing, 2008.

[7] Zaharia.M, Borthakur.D, SarmaJ, K. Elmeleegy, Shenker.S, and Stoica.I, "Delay scheduling: A simple technique for achieving locality and fairness in cluster scheduling," in European Conference on Computer Systems (EuroSys), Apr. 2010.