Future Generation Computer Systems 54 (2016) 123–131

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

A multi-dimensional job scheduling

Mehdi Sheikhalishahi[c,*], Richard M. Wallace[b], Lucio Grandinetti[a], José Luis Vazquez-Poletti[b], Francesca Guerriero[c]

[a] Department of Electronics, Comp. Sci. & Sys. University of Calabria, Rende(CS), Italy
[b] Department of Computer Arch. & Automation Complutense University, Madrid, Spain
[c] Department of Mechanical, Energy and Management Engineering, University of Calabria, Rende(CS), Italy

HIGHLIGHTS

- A proposal for scheduling problem based on multi-capacity bin-packing algorithms.
- A proposal for host selection and queuing based on multi-resource scheduling.
- Getting better *waittime* and *slowdown* metrics than the state of the art scheduling.

ARTICLE INFO

Article history:
Received 15 October 2014
Received in revised form
27 February 2015
Accepted 15 March 2015
Available online 27 March 2015

Keywords:
Multi-dimensional resource
Queuing mechanism
Resource management
Scheduling
Bin-packing
Performance

ABSTRACT

With the advent of new computing technologies, such as cloud computing and contemporary parallel processing systems, the building blocks of computing systems have become multi-dimensional. Traditional scheduling systems based on a single-resource optimization, like processors, fail to provide near optimal solutions. The efficient use of new computing systems depends on the efficient use of several resource dimensions. Thus, the scheduling systems have to fully use all resources. In this paper, we address the problem of multi-resource scheduling via multi-capacity bin-packing. We propose the application of multi-capacity-aware resource scheduling at host selection layer and queuing mechanism layer of a scheduling system. The experimental results demonstrate performance improvements of scheduling in terms of *waittime* and *slowdown* metrics.

1. Introduction

From a scheduling and resource view for computing, there can be a few major issues and problems to consider: low utilization, overloaded systems, poor performance, and resource contention. Solving these issues and problems requires answering complex questions that start with, "*When*...", "*Which*...", and "*Where*...". For instance, "When should some workloads be migrated to other servers?", "Which types of applications should be consolidated together in a server?", and "Where should a workload be placed?" These examples are the type of resource management questions to consider and this list has many more resource management questions of this type.

Scheduling algorithms based on *First-Come First-Served* schemes (FCFS) pack jobs from the job queue into the system in order of their arrival until a resource is exhausted. If there is a large job at the head of the queue which requires more resources than those left available in the system, the job allocation scheme is blocked from scheduling further jobs until sufficient resources become available for this large job. This results in potentially large resource fragments being *under-utilized*. *Back-filling* mechanisms overcome this issue by skipping over jobs that cannot be allocated and by finding smaller jobs that can make use of remaining resources.

Back-filling in addition to being a scheduling algorithm is a queueing mechanism, that is it changes the order of the jobs in the queue for a faster allocation of jobs which can be allocated to the computing system.

With the advent of new computing technologies such as cloud computing as a recent development in the field of computing and massively parallel processing systems such as the most recent *Cray JK7* system (Titan), the *Chinese Tianhe-1A* system (NUDT YH MPP),[1]

* Corresponding author.
E-mail addresses: alishahi@unical.it (M. Sheikhalishahi), wallacerim@aol.com (R.M. Wallace), lugran@unical.it (L. Grandinetti), jlvazquez@fdi.ucm.es (J.L. Vazquez-Poletti), guerriero@deis.unical.it (F. Guerriero).

1 http://top500.org/lists/2012/11/.

http://dx.doi.org/10.1016/j.future.2015.03.014
0167-739X/© 2015 Elsevier B.V. All rights reserved.

and the quite old *SUN E10000* and *SGI O2K* systems, the building blocks of computing systems have become multi-dimensional. The *Titan* system is installed at Oak Ridge, achieving 17.59 Petaflop/s on the Linpack benchmark with 560,640 processors, including 261,632 *NVIDIA K20x* accelerator cores.[2]

From the processing point of view, according to the *Top500* list, a total of 62 systems on the *Top500* list are using accelerator/co-processor technology including *Titan* and the *Chinese Tianhe-1A* system which use *NVIDIA GPUs* to accelerate its computation. Moreover, *Stampede* and six other super-computers are accelerated by the new *Intel Xeon Phi* processors (Intel MIC architecture).[3] As a result there are multiple computing elements to be taken into account in scheduling at the processor level.

Scheduling techniques in older computer systems, such as the massively parallel processing systems *TMC CM-5* and the *CRAY T3E*, were focused on a single resource dimension allocation (processing nodes) where single capacity bin-packing algorithms were used to solve this problem.[4]

In multi-dimensional resource environment a single resource still becomes exhausted while others remain under-used even with the *back-filling* strategy as the scheduling algorithm. This is due to the design of *FCFS* algorithms which are restricted in job selection based on their arrival order and not addressing capacity imbalance between resources in a multi-resource environment. *Back-filling* strategy is an instance of FCFS mechanism. Thus, single capacity bin-packing algorithms are inadequate as they are unable to provide optimal scheduling for multi-dimensional resources of *CPU, GPU, memory, shared memory, large disk farms, I/O channels, bandwidth, network input*, *network output*, and even *software licenses* of current computing system architectures.

Therefore, the scheduling scheme must be free to select any job based on matching all of the jobs' resource requirements with the available system resources in order to address the efficient use of resources in a multi-resource environment. The target of efficient use of new computing architectures depends on efficient usage of all resource dimensions with the scheduling algorithm fully using all resources.

In this paper, we research *multi-resource scheduling* by modeling the problem as a *multi-capacity bin-packing* problem. First, we propose a new host selection policy for each job based on multi-capacity criteria, that happens at the lowest layer of scheduling (the last fate of job placement). Then, we propose a multi-capacity-aware queuing mechanism. We model multi-resource scheduling as a multi-capacity bin-packing scheduling algorithm at the queue level to reorder the queue in order to improve the packing and as a result to improve scheduling metrics.

The approach was first presented in [1] as part of ARMS-CC-2014 workshop [2]; in which we presented a multi-resource queuing mechanism to provide a higher degree of consolidation in multi-dimensional computing systems. In this paper, we extend the approach at the host selection policy to provide a scheduling system aware of multi-dimensional resources. In summary, our paper makes the following contributions:

- A proposal for scheduling problem based on multi-capacity bin-packing algorithms.
- A proposal for host selection and queuing mechanism based on multi-capacity bin-packing scheduling algorithm.
- We show experimentally that our multi-resource scheduling system performs more efficiently than the state of the art scheduling system based on back-filling policy as measured by *waittime* and *slowdown* metrics.

The remaining part of this paper is organized as follows. Section 2 reviews related work. Section 3 presents our multi-resource scheduling approach that is modeled based on a multi-capacity bin-packing algorithm. Then, it details the design of a host selection policy and a queuing mechanism based on multi-capacity bin-packing algorithm. Section 4 explains detailed design and implementation issues such as workload traces, and resource model for experiments of this paper. After that, it discusses simulation experiments and experimental results. Finally, Section 5 presents our conclusions and future work.

2. Related work

Single- and multi-capacity bin-packing problems and their connection to the generalized scheduling problem have been studied in [3–7]. The two-dimensional vector packing problem [7] consists in orthogonally packing a subset of a set of rectangular-shaped boxes, without overlapping, into a single bounding rectangular area, maximizing the ratio between the area occupied by the boxes and the total available area.

The *d*-capacity bin-packing solution approaches extend the single capacity bin-packing solutions, i.e., *First-Fit* (FF), *Next-Fit* (NF), and *Best-Fit* (BF), to deal with the *d*-capacity jobs (items) and nodes (bins). FF, NF, and BF are considered as *Job-To-Node* placement rules. Those *d*-capacity bin-packing algorithms that are extensions of the single capacity bin-packing do not scale well with increasing *d* since they do not take advantage of the information in the additional capacities. The work by Bobroff et al. [8] presents a first-fit approximation algorithm for the bin packing problem. The algorithm was devised for the single resource problem, but tips are given about the extension to multiple resources. Orthogonal to the *Job-To-Node* placement rules is the job queue preprocessing method used before the packing operation. For the single capacity bin-packing algorithm sorting the list based on a scalar value in a non-increasing order with respect to the job resource requirement improves the packing performance. The *First-Fit Decreasing* (FFD) algorithm first sorts the list in a non-increasing order and then applies the FF packing algorithm. The NF and BF algorithms can be extended in a similar manner.

Leinberger et al. [9] proposed a *d*-capacity bin-packing algorithm named *Multi-Capacity Bin Packing* (MCBP). It is a particular vector packing algorithm that uses the additional capacity information to provide better packing by addressing the capacity imbalance. The authors show how their algorithms lead to better multi-resource allocation and scheduling solutions.

In addition, the problem of optimally mapping virtual machines (VMs) to servers can be reduced to the *bin packing problem* [10–12]. This problem is known to be NP-hard, therefore heuristic approaches can only lead to sub-optimal solutions. With regard to recent work finding an FFD algorithm that has better execution time, the paper [13] provides an algorithm that maximizes the dot product between the vector of remaining capacities and the vector of remaining or residual capacities of the current open bin, i.e. subtract from the bin's capacity the total demand of all the items currently assigned to it. It places the item that maximizes the weighted dot product with the vector of remaining capacities without violating the capacity constraint vector of demands for the item. This bin-centric method did show better performance. This method is an alternative to our method and is intended for allocation of VM images rather than scientific job placement. The argument can be made that a VM image can have the same processing footprint as a long-lived scientific application.

Moreover, novel job scheduling mechanisms use *d*-capacity bin-packing algorithms. For instance, [14,15] employ an algorithm based on MCBP proposed by Leinberger et al. in [9]. In [14], a novel job scheduling approach for homogeneous cluster computing

[2] https://www.olcf.ornl.gov/titan/.
[3] https://www.tacc.utexas.edu/stampede/.
[4] http://www.top500.org/system/166997.