QoS-aware Parallel Job Scheduling Framework for Simulation Execution as a Service

Zhen Li, Bin Chen*, Xiaocheng Liu, Dandan Ning, Wei Duan, Xiaogang Qiu
College of Information System and Management
National University of Defense Technology
Changsha, 410073, China

Chengda Xu
Bengbu Automobile NCO Academy
Bengbu, 233000, China

***Abstract*—Cloud computing is attracting an increased number of researches in delivering modeling and simulation abilities as a service. Among which, simulation execution as a service (EaaS) is a hot spot. It aims at releasing users from complex running configurations and meanwhile guaranteeing the QoS requirements. Under the motivation, focusing on EaaS for parallel and distributed simulation (PADS) application , the paper proposes a QoS-aware job scheduling framework in two-tier virtualization-based private cloud data center. In PADS EaaS, an adaptive job size adjustment component is designed to realize intelligent and adaptive job size setting for PADS instead of assigning by users. Furthermore, an adaptive deadline-aware job size adjustment algorithm, named ADaSA, is designed in the adjustment component to realize efficient job scheduling with high job responsiveness. ADaSA algorithm firstly computes a minimum processor requested that leads to maximum runtime stretch. It makes sure that more jobs can be scheduled at the same time while satisfying current job's deadline requirements. On other hand, ADaSA tries to pick up all possible idle CPU time in background virtual machines and reserved ones for other jobs. Through that way, more chances are generated to response more jobs in waiting queue. Finally, we conduct extensive experiments with trace-driven simulation. The results show that ADaSA outperforms both cloud-based job scheduling algorithm KCEASY and traditional EASY in terms of response time (up to 90%) and bounded slow down (up to 95%), and at the same time guarantees approximately equivalent deadline-missed rate. ADaSA also outperforms two representative moldable scheduling algorithms in terms of deadline-missed rate (up to 60%).**

***Keywords*—Parallel Simulation; Job Scheduling; QoS; Cloud-based Simulation; Execution as a Service.**

I. Introduction

Computer simulation techniques are widely used in many domains, and they give domain users the opportunities to conduct intensive and repeatable experiments on simulated models conveniently. However, computer simulation is an interdisciplinary science which combines domain related knowledge, software engineering and computer science. Consequently, it is often time-consuming and expensive when conducting large scale complex system simulation.

With the emergency of cloud computing, cloud-based simulation has provided great opportunities to promote efficiency and reducing cost for building and executing simulation. In cloud computing, IT resources are provided transparently through cloud computing service, eg., Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS); the consumers then access the IT resources service in an easy-to-use, on-demand and pay-as-you-go manner. Base on the underlying cloud computing paradigm, cloud-based simulation consolidates the underlying computation and simulation resources in data centers and provides modeling and simulation as a service (M&SaaS) to support the whole process of M&S. Different users (including domain experts, modelers, execution operators and analysts etc.) can concentrate on their own domains and complete the whole simulation collaboratively and rapidly with extensively reuse. Nevertheless, researches in cloud-based simulation are still in a preliminary stage and have been mentioned as one of the grand challenges in M&S by various experts [1] .

Among previous pioneering works, the CSim [2] proposed by Xiaocheng Liu gave an architecture design of cloud-based simulation. According to the whole process of simulation system engineering, the M&SaaS was further divided into MaaS (Modeling as a Service), EaaS (Execution as a Service) and AaaS (Analysis as a Service) in CSim . Among which, EaaS is thought to be the important and most valuable functionality provided by cloud-based simulation [3]. With the help of auto-scaling characteristic of cloud computing supported by hardware virtualization, EaaS can realize automatic deployment of simulation applications and offer fast response through scale in/out and scale up/down of resources.

PADS (Parallel and Distributed Simulation) application utilizes multi-processors to run multiply logical processes (LPs) of simulation models cooperatively through communication and synchronization mechanism. It has been extensively used in simulating complex system to accelerate execution time. Accordingly, realizing EaaS for PADS applications has become a hot topic in cloud-based simulation. Guan et al. [4] proposed a multi-layered scheme for distributed simulation in the cloud. It is designed to ease the management of underlying simulation resources and achieve rapid computing capacity elasticity for different scale simulation applications considering energy consumption, security and scalability. Wang et al. [5] proposed a simulation middleware called CloudRISE which helps simplify the deployment and management processes of simulation resources. Then, through the web service interface in CloudRISE, end users can run simulation everywhere on demand. These researches above provide excellent

example implementations of EaaS, however they mainly care about the deployment and resource management for single simulation application, and seldom consider the underlying PADS job scheduling problems in multi-tenancy cloud environment. When multiple PADS jobs (in the paper we regard a PADS application as a job) are submitted simultaneously during a time window, how to arrange the execution priority and map resources to jobs has critical influences on guaranteeing users' QoS (Quality of Service) requirements and jobs' response time.

Aiming at the PADS job scheduling problems in cloud, Liu et al. [6] proposed a novel two-tier virtualization architecture to make full use of the wasted idle CPU time caused by communication and synchronization between LPs in PADS application. Based on the two-tier virtualization architecture, four novel consolidation-based PADS job scheduling algorithms [7] [8] were devised to shorten the makespan of scheduling submitted jobs and improve the average response time. However, these proposed algorithms are designed based on rigid jobs that users should specify the size (namely number of processors needed) of their jobs before submission. Actually, the PADS applications are moldable that job size parameters can be assigned at any time before execution. However, users are usually not familiar with the parallelism characteristics of the underlying PADS jobs. In that case, users often feel confused and simply request as many processors as possible under budget constrain when submitting jobs, which will cause low utilization of cloud data centers. Aiming at the problems, realizing an intelligent and efficient adaptive processor allocation component for PADS applications becomes a valuable research issue in EaaS considering including jobs' parallelism characteristics, current workloads in cloud and users' QoS requirements. The reason is due to: 1) It offers more transparent and convenient simulation execution service that users can easily obtain simulation results by simply submitting jobs and specifying related QoS requirements (eg. soft deadlines etc.). 2) It has the potential to improve the response time for submitted jobs by adaptive job size setting as long as the deadline constrains are guaranteed.

The adaptive processor allocation for PADS jobs can be viewed as moldable parallel job scheduling problem. Several moldable job scheduling algorithms were proposed in previous researches [9] [10] [11] [12]. Most of them utilized Downey model as the parallelism characteristic model, and decided the job size based on different kinds of heuristic information at scheduling time or submission time. Reference [9] [10] designed fairness-based method through adding job weights and parallelism characteristic information by revising traditional greedy-based algorithm , but they seldom considered the current workload states and QoS demands of jobs. Reference [11] synthesized multiple metrics (eg. over-deadline, makespan and idle cpu time) as scheduling objects and designed several heuristic scheduling algorithms to manage job allocation among multi-clusters. But they are designed oriented to grid environment which is quite different with cloud. Reference [12] then designed a cloud-oriented moldable job scheduling strategy that realized intelligent and efficient execution service for end users. But its target jobs are general HPC workloads which are quite different with PADS jobs in communication and synchronization mechanisms between processes. Unlike previous researches, our work focuses on the scheduling of PADS jobs in cloud-based simulation platform considering both job QoS demands and job response time. Our work is designed based on CSim, and makes the following contributions: 1) a QoS-aware scheduling framework simulation EaaS is proposed to realize adaptive processor allocation for PADS jobs. 2) a deadline-aware moldable PADS job scheduling algorithms is devised based on CSim. 3) extensive experiments and comparisons are carried out to show the efficiency of our proposed algorithm.

The rest of the paper is organized as follows: section II presents the QoS-aware scheduling framework and the problem definition of the scheduling model. The detail illustrations of deadline-aware moldable PADS job scheduling algorithm are given in section III. Section IV shows the experiment settings and result analysis. Section V concludes our work.

II. QoS-aware Parallel Simulation Job scheduling Framework

Because of high bandwidth delay, public cloud environment is still not suitable for tightly coupling PADS jobs. However, as PADS jobs can hardly make full use of requested processors caused by the communication and synchronization between LPs , local private cloud data center raises great opportunities for improving the whole PADS jobs scheduling performance [6] with advantage of hardware virtualization (resource reuse and consolidation etc.) in spite of introducing acceptable performance degrade of single job (less than 5% for HPC application [13]). Therefore, in the paper, in order to realize good performance for our PADS EaaS, we employ private cloud environment and the two-tier virtualization architecture proposed in CSim [2] .

A. QoS-aware scheduling architecture

Supposing that the speedup model of submitted PADS job can be obtained through history data, then PADS-oriented EaaS can offer effortless and high performance simulation execution service for users which only requires job deadline setting. Accordingly, in this paper, we extend the CSim and design a QoS-aware scheduling architecture for PADS-oriented EaaS, as shown in Figure 1.

The architecture consists of three layers: user layer, scheduler layer and data center layer. In PADS-oriented EaaS, multi-users share the same service and submit their PADS jobs dynamically to the scheduling broker. According to certain optimization objects and users' QoS requirements, scheduling broker then decides job execution order and the mappings between jobs and VMs. The data center layer is enhanced with two-tier virtualization architecture and provides workload consolidation opportunities to improve resource utilization.

The scheduling broker for PADS-oriented EaaS is the primary concern of this paper. It consists of four major

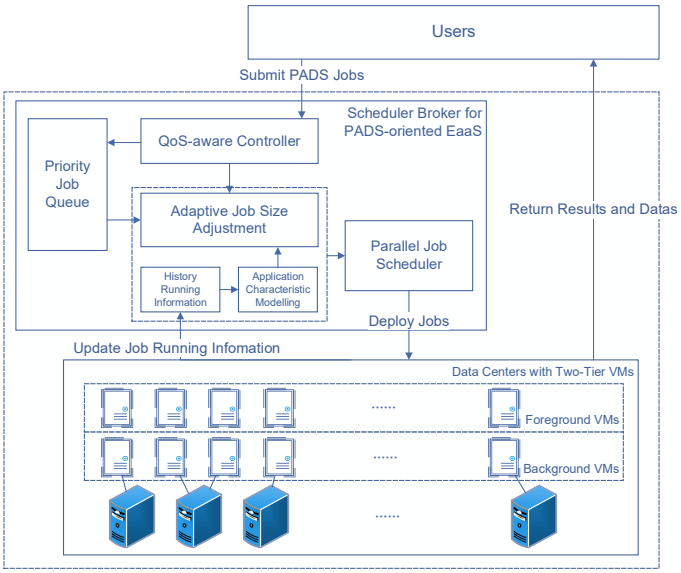


Fig. 1: The QoS-aware scheduling architecture for PADS EaaS

components: QoS-aware Controller (QaC), Priority Job Queue (PJQ), Adaptive Job Size Adjustment (AJSA) and Parallel Job Scheduler (PJS). QaC is responsible for ranking the jobs in PJQ according to users' QoS requirements and offering QoS-based heuristic information for AJSA. PJQ accommodates the waiting jobs. AJSA is responsible for making decision of job size for waiting jobs in PJQ. Two sub-components are needed for implementing the function: one is "History Running Information Database component" and another is "Application Characteristic Modeling component". PJS is responsible for scheduling PADS jobs to VMs in data centers.

B. Job scheduling method

Based on the architecture, we design an ADaSA (Adaptive Deadline-aware Job Size Adjustment) algorithm for PADS EaaS under two-tier virtualization architecture [2].

In two-tier virtualization architecture, the predicted execution time in foreground VMs is supposed to be accurate, and the size of submitted jobs can be adjusted at any time before running in VMs. Therefore, jobs have chances to adjust job size according to current number of idle processors, as designed in some popular moldable job scheduling algorithms, such as parallel allocation strategy [12] and fair-shared allocation strategy [9] [14]. Parallel allocation strategy tries to schedule as many jobs as possible at the same time which focuses on minimizing the average response time. Fair-shared allocation strategy tries to allocate as many processors as possible to the first job in the queue which focus on minimizing the average turnaround time. However, these methods above are all greedy strategies which only focus on optimizing single objective without considering jobs' QoS requirements. Accordingly, our proposed ADaSA algorithm tries to make a trade-off between multiply optimization objectives defined in section II. The algorithm is designed based on KCEASY(Kill&Consolidation enhanced EASY) [8] in CSim. On VM^{fg} layer, KCEASY acts as an Extensible Argonne Scheduling sYstem (EASY, which is actually an aggressive backfill algorithm), which ensures that a job is dispatched to run in foreground VMs whenever it is schedulable, and at the same time it should not delay the start time in reserve of jobs with higher priority. Meanwhile, on VM^{bg} layer, it schedules waiting jobs to background VMs in SJF (Short Job First) strategy when no jobs are schedulable in VM^{fg} layer, which improves the utilization of CPU time.

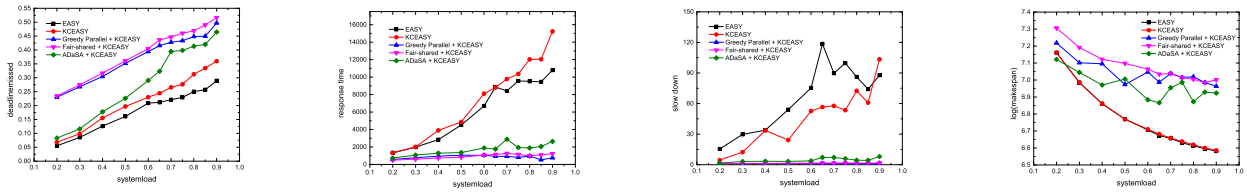
The main idea of ADaSA is to improve the response performance, meanwhile keeping a relative low deadline violation. Previous parallel allocation strategy tries to allocate as many jobs as possible to the idle VMs, which leads to small setting of job size. However, in many cases, a job in the setting of small job size leads long execution time and may cause deadline violation. In order to solve the problem, ADaSA firstly assigns each waiting jobs an initial minimum job size setting that achieves maximum job runtime stretch. Minimum job size of a job gives as job size setting constrain that job cannot set smaller job size in order to satisfy the deadline requirement. In this paper, we denote the minimum job size setting as min_size. When jobs are scheduled with min_size, the deadline requirements are guaranteed, and at the same time more jobs can be scheduled in parallel to improve response performance. Each job in waiting queue will update its min_size after any jobs is released from foreground VMs. The min_size setting of a job can be obtained through a pre-schedule process (simulated scheduling process) of EDF (Earliest Deadline First) according to the state of Priority Queue $Q(t)$ and foreground VMs set $VM^{fg}(t)$ at current time t.

When there are no schedulable jobs in VM^{fg} with current job size settings, ADaSA then chooses jobs in waiting queue and adjusts the job size to match the idle resource separately on VM^{fg} and VM^{bg}.

In VM^{fg} layer, a set of VMs can be idle when the head job is not schedulable under minimum job size constrain at current time. Under the situation, ADaSA choose the jobs in the waiting queue that can backfill to the idle VMs if they can finish before the predicted start time of head job. In VM^{bg} layer, it can pick up the idle CPU time left without affecting execution efficiency of jobs in VM^{fg}. Accordingly, shorter jobs (smaller job runtime) have higher priority to be scheduled to VM^{bg}, as they have higher probability to finish in VM^{bg}. At the same time, jobs scheduled to VM^{bg} should queue up in the PJQ for waiting resource in VM^{fg}. If they come to the head of priority queue before finishing in VM^{bg}, they will be killed and rescheduled or migrated to VM^{fg}. Accordingly, when selecting jobs running on VM^{bg}, in order to satisfy the deadline requirement when jobs are migrated to VM^{fg}, the minimum job size constrain should be considered.

III. Performance Evaluation

In this section, we evaluate ADaSA under PADS EaaS architecture by using trace-driven simulation. In the simulation, a cluster with 300 physical processors is offered as data center,



(a) deadline missed rate (b) the average response time (c) the average bounded slow down (d) scheduling makespan

Fig. 2: Performance comparison using janned parallel workload

and each processor is pinned with a foreground VM and a background VM. "Janned workload" [15] is used as workload model in the simulation.

In PADS EaaS, jobs are actually scheduled supported by two components as shown in Figure 1. The first one is an Adaptive Job Size Adjustment (AJSA) component with ADaSA, the second one is a Parallel Job Scheduler (PJS) component with KCEASY.

Accordingly, we compare PADS EaaS with other four scheduling services including:

1) Scheduling service without AJSA, and use EASY-based PJS;

2) Scheduling service without AJSA, and use KCEASY-based PJS;

3) Scheduling service with greedy parallel algorithm-based AJSA, and use KCEASY-based PJS;

4) Scheduling service with fair-shared algorithm-based AJSA, and use KCEASY-based PJS;

The comparison is conducted under four different metrics and the results are shown in Figure 2. From the observations above, it can be seen that our proposed PADS EaaS with proposed ADaSA algorithm can offer good QoS, which significantly improves service response time for users while ensuring low deadline-missed rate.

IV. Conclusion

In this paper, we investigate the framework design of simulation execution as a service for parallel and distributed simulation applications in private cloud data center. An adaptive job size adjustment component is designed in order to provide QoS-aware and intelligent job scheduling service without any users' running configurations before submission. An adaptive deadline-aware job size adjustment algorithm, named ADaSA, is devised in the adjustment component to realize efficient job scheduling with high job responsiveness and low deadline-missed rate. Through extensive experiments with trace-driven simulation, our proposed algorithm obtains significantly improvement compared with a popular cloud-based job scheduling algorithm in terms of average job response time (up to 90%) while keeping relative low deadline-missed rate.

Acknowledgment

The study is supported by the National Natural of Science Foundation of China under Grant Nos.71673292, 61503402, 61403402, 61374185, 71373282 and Shanghai Special Foundation of Software and Integrated Circuit under Grant No. 150312.

References

[1] S. J. E. Taylor, A. Khan, K. L. Morse, A. Tolk, L. Yilmaz, and J. Zander, "Grand challenges on the theory of modeling and simulation," in *Symposium on Theory of Modeling & Simulation - Devs Integrative M&s Symposium*, 2013, pp. 34:1–34:8.

[2] X. Liu, X. Qiu, B. Chen, and K. Huang, "Cloud-based simulation: The state-of-the-art computer simulation paradigm," in *Principles of Advanced and Distributed Simulation*, 2012, pp. 71–74.

[3] B. S. S. Onggo, S. J. E. Taylor, and A. Tulegenov, "The need for cloud-based simulation from the perspective of simulation practitioners," 2014.

[4] S. Guan, R. D. Grande, and A. Boukerche, "A multi-layered scheme for distributed simulations on the cloud environment," *IEEE Transactions on Cloud Computing*, no. 99, pp. 1–1, 2015.

[5] S. Wang and G. Wainer, "Modeling and simulation as a service architecture for deploying resources in the cloud," *International Journal of Modeling Simulation & Scientific Computing*, vol. 07, no. 01, 2016.

[6] X. Liu, Q. He, X. Qiu, B. Chen, and K. Huang, "Cloud-based computer simulation: Towards planting existing simulation software into the cloud," *Simulation Modelling Practice & Theory*, vol. 26, no. 6, pp. 135–150, 2012.

[7] X. Liu, C. Wang, B. B. Zhou, and J. Chen, "Priority-based consolidation of parallel workloads in the cloud," *IEEE Transactions on Parallel & Distributed Systems*, vol. 24, no. 9, pp. 1874–1883, 2013.

[8] X. Liu, Y. Zha, Q. Yin, Y. Peng, and L. Qin, "Scheduling parallel jobs with tentative runs and consolidation in the cloud," *Journal of Systems & Software*, vol. 104, pp. 141–151, 2015.

[9] S. Srinivasan, S. Krishnamoorthy, and P. Sadayappan, "A robust scheduling strategy for moldable scheduling of parallel jobs," in *IEEE International Conference on CLUSTER Computing, 2003. Proceedings*, 2003, pp. 92–99.

[10] G. Sabin, M. Lang, and P. Sadayappan, "Moldable parallel job scheduling using job efficiency: an iterative approach," in *International Conference on Job Scheduling Strategies for Parallel Processing*, 2006, pp. 94–114.

[11] L. He, S. A. Jarvis, D. P. Spooner, X. Chen, and G. R. Nudd, "Hybrid performance-oriented scheduling of moldable jobs with qos demands in multiclusters and grids," in *Grid and Cooperative Computing - GCC 2004: Third International Conference, Wuhan, China, October 21-24, 2004. Proceedings*, 2004, pp. 217–224.

[12] K. C. Huang, T. C. Huang, M. J. Tsai, H. Y. Chang, and Y. H. Tung, "Moldable job scheduling for hpc as a service with application speedup model and execution time information," *JoC*, vol. 4, pp. 14–22, 2013.

[13] A. Iosup, S. Ostermann, M. N. Yigitbasi, and R. Prodan, "Performance analysis of cloud computing services for many-tasks scientific computing," *IEEE Transactions on Parallel & Distributed Systems*, vol. 22, no. 6, pp. 931–945, 2011.

[14] S. Srinivasan, V. Subramani, R. Kettimuthu, P. Holenarsipur, and P. Sadayappan, "Effective selection of partition sizes for moldable scheduling of parallel jobs," 2002.

[15] J. Jann, P. Pattnaik, H. Franke, F. Wang, J. Skovira, and J. Riordan, "Modeling of workload in mpps," in *Job Scheduling Strategies for Parallel Processing*, 1997, pp. 95–116.