Deadline-aware Resource Scheduling Service in Multiple Infrastructure Including Hybrid Cloud

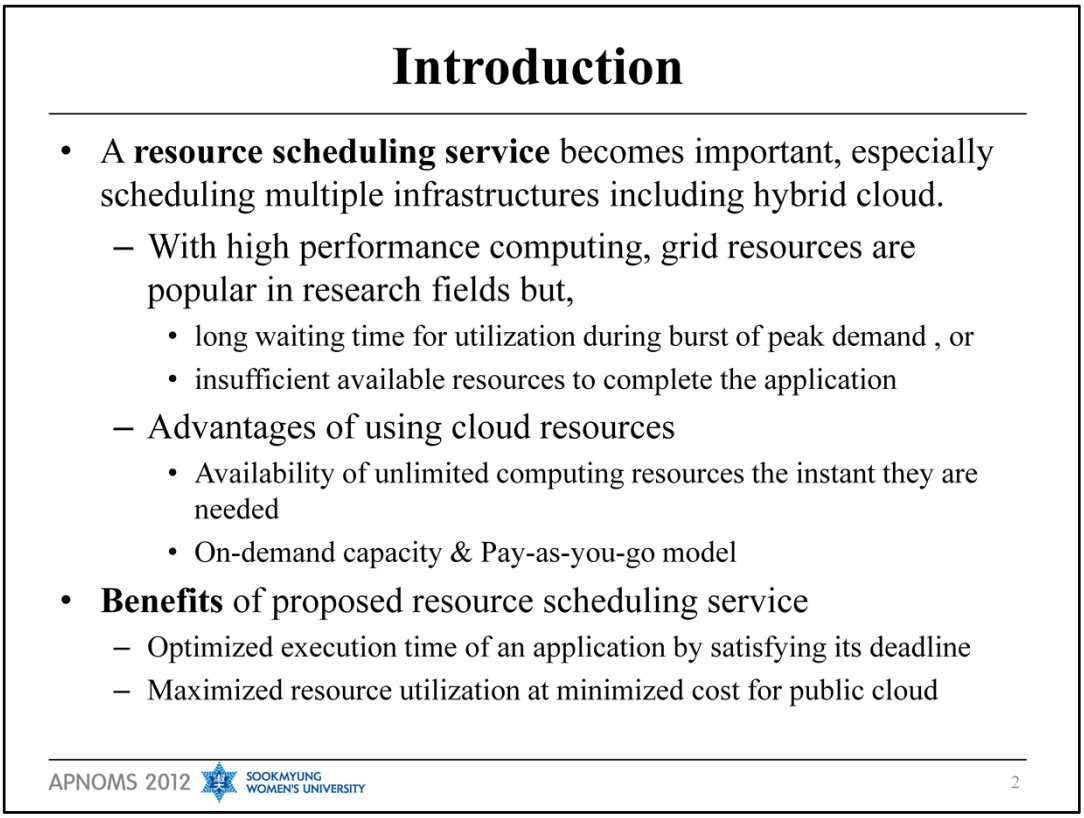
Jung-In Koh, Seoyoung Kim, Yoonhee Kim and Soon-wook Hwang*
Distributed Systems Lab.
Dept. of Computer Science
Sookmyung Women's Univ, Seoul Korea
Email: {jungin, sssyyy77, yulan}@sookmyung.ac.kr,
http://dslab.sookmyung.ac.kr
*Korea Institute of Science and Technology Information, Deajeon Korea
Email: hwang@kisti.re.kr,

This research was supported by the Application Service Development Project of the KISTI (Korea Institute of Science and Technology Information) (K-12-L06-C02-S05).

Abstract

Over the years, grid resources have been instrumental in promoting development of various scientific research fields using high performance computing. Furthermore, as the recent advent of cloud computing facilitates elastic extension of computing, there are increasing number of studies on putting cloud computing resources to good use.

In this paper, we present a deadline-driven scheduling service that maximizes efficient utilization of grid and hybrid cloud resources. The scheduling service enables to increase resource utilization by satisfying a deadline of each application, and optimizes an execution time of the application by allocating public cloud resources under control of a monitor. Besides, the scheduling service ensures an application is finished within its deadline at minimum financial cost.

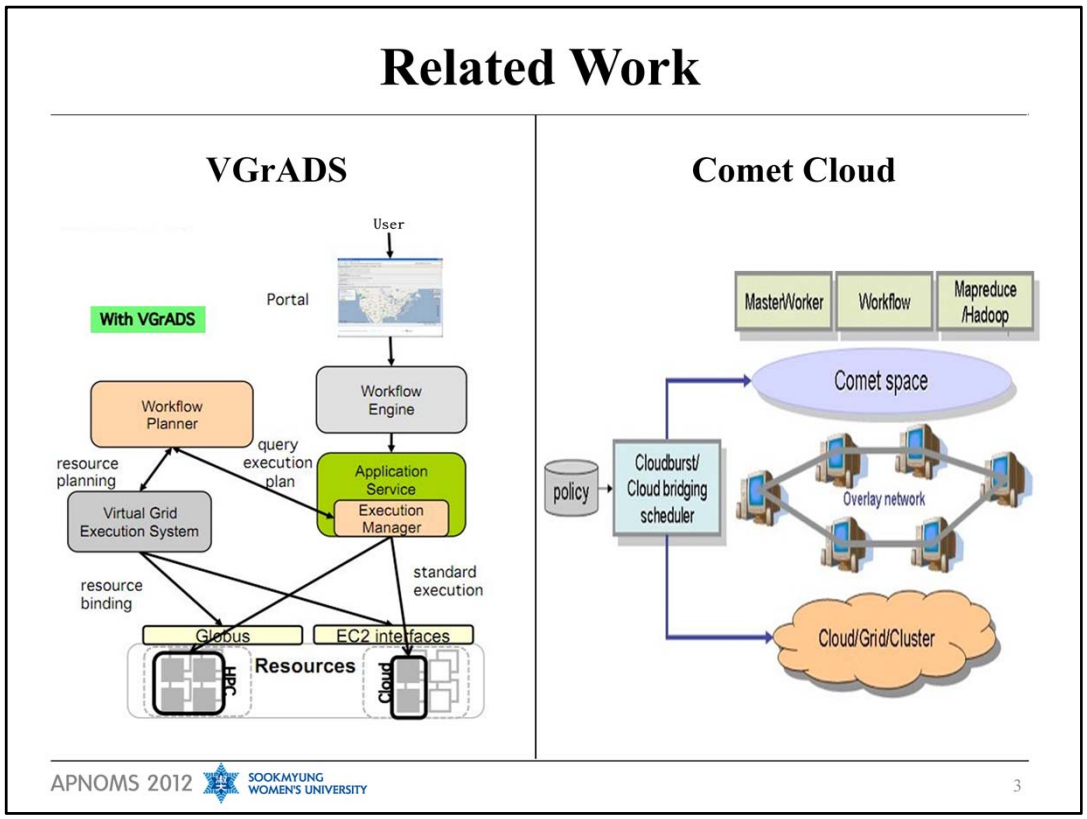


1. Introduction

Grid computing has a significant ability to harness together the power of computing resources. But the high performance computing offers a static and finite computational capacity to users, For example, peak demand for computing resources may be seen in sometimes, which can lead to long waiting times for utilization for these resources, or the available resources for one application may be insufficient to complete the application before its deadline. In these cases, flexibility of cloud computing resources may be the solution. Moreover, the use of cloud computing offerings has found acceptance in both industry and research. It is thus critical to study on hybrid cloud scheduling service.

In this paper, we focus on the deadline-aware resource scheduling service in multiple infrastructures including hybrid cloud. Our scheduling service enables to increase resource utilization by satisfying a deadline of each application, and optimize an execution time of the application by allocating public cloud resources under control of a monitor. Furthermore, the scheduling service ensures an application is finished within its deadline at minimum financial cost.

Our paper is organized in the following sections. We firstly discuss some related works in Section 2. After that, in Section 4, we introduce a general overview of our scheduling service with algorithms and an example of our proposed adaptive scheduling concept in Section 5. Finally, we present experiment plans for an ongoing progress of our work and conclude this paper in last section.

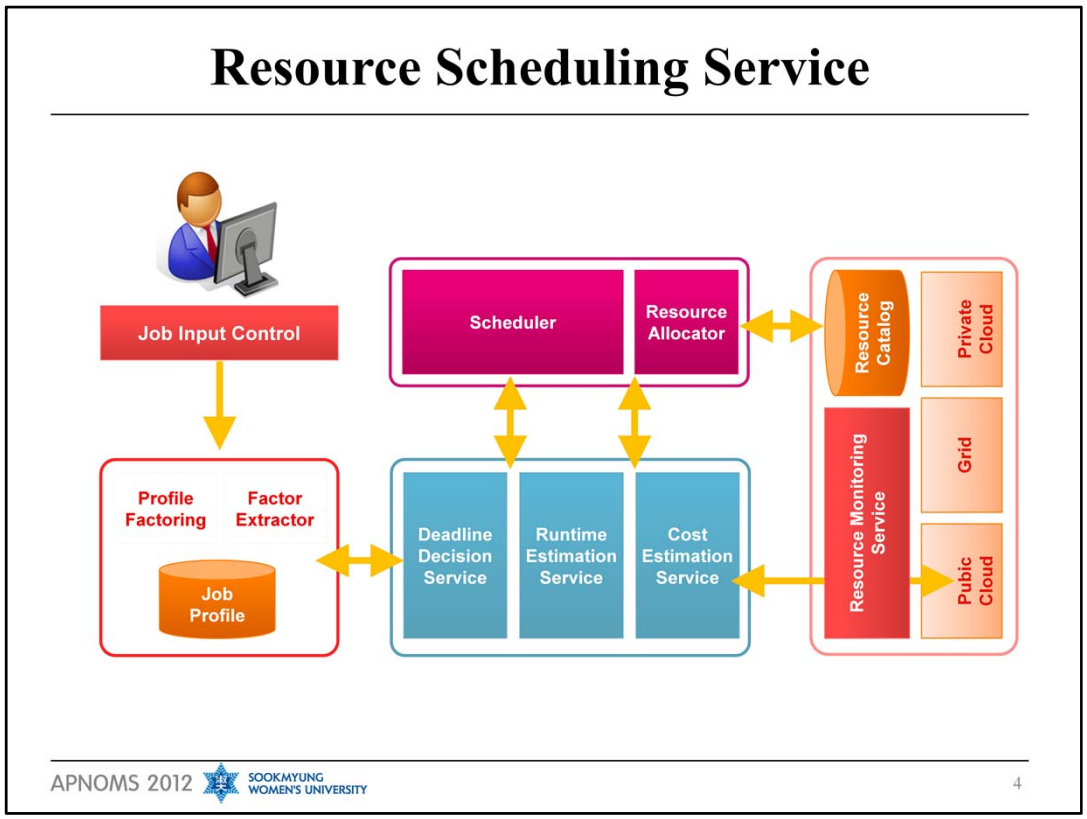


2. Related Work

Several works tackle the problem of scheduling hybrid cloud in efficient way while fulfilling the applications' quality of service constraints.

VGrADS(Virtual Grid Application Development Software) project[1] implements a cloud computing environment by extending Amazon EC2 resources with open-source cloud software platform, Eucalyptus to execute HPC(High Performance Computing) workflows. As a goal of the project is implementing a scheduling policy and infrastructures particularly for time-sensitive application among e-science applications, load balancing of multiple infrastructures (such as TeraGrid[10] and Amazon EC2) and stability have been studied for a new workflow. However, it has not been shown any research results about a provisioning based on additional policies (deadline, cost and etc.).

Comet Cloud[2] provides a deadline-constrained resource provisioning which chooses suitable set of resources among large-scale grid and public/private cloud by deadline policy. Although it respectively considers deadline policy and budget constraint, until now, they does not present any experimental data considering both deadline and budget constraint.



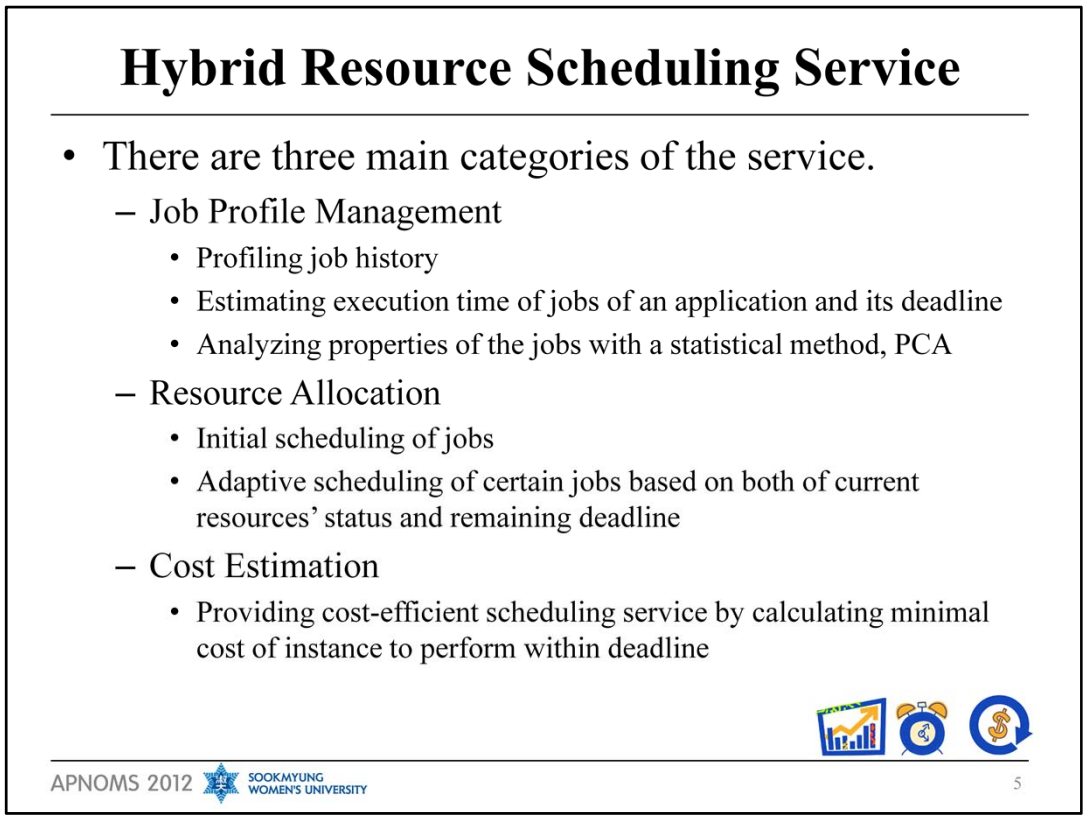
4. Resource Scheduling Service

4. 1 Hybrid Resource Scheduling Service Architecture

This slide shows the hybrid resource scheduling service architecture. If application is submitted via *Job Input Control*, *Profile Factoring* generates numeric values about how much each parameter influences on the application. After job histories are extracted by high number of the generated values, *Deadline Decision Service* provides reasonable deadline for the application among them.

Resource Scheduling, related to *Scheduler*, *Runtime Estimator*, and *Resource Monitor,* schedules one or more jobs of the application based on the deadline. *Runtime Estimato*r predicts execution time of each job and *Resource Monitor* periodically keeps watch on status of resources. As considering the predicted execution time of a job and current status of resources from the modules, *Scheduler* decides a resource in which the job will be scheduled, and *Resource Allocator* is responsible to service practical job execution by delivering the job to a proper adapter among *Resource Adapters* such as *Grid*, *Private* and *Public Cloud adapter.* While the jobs are executed by initial scheduling, in a case of insufficient available local resources for computing, scheduler allocates public cloud. *Cost Estimation Service* services minimum-cost instance for the allocation.

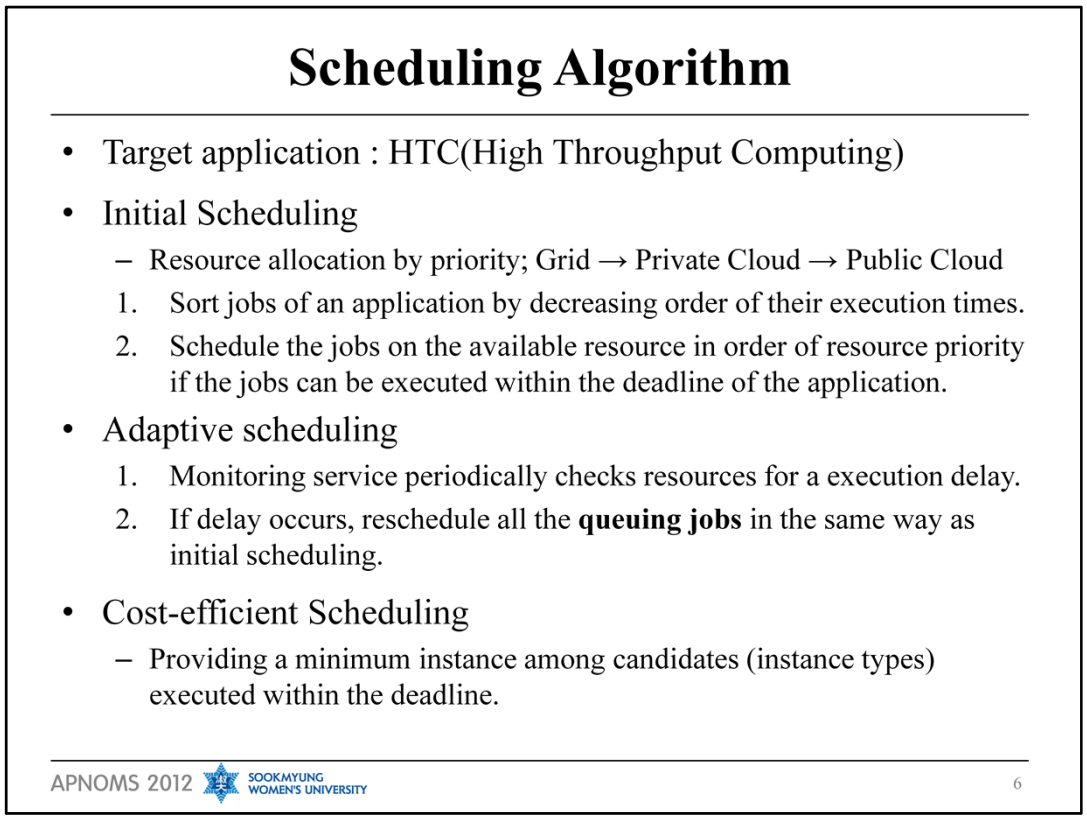
Additionally, adaptive scheduling can be performed. If delayed jobs may be violated the deadline, which are detected by *Resource Monitor*, *scheduler* will reschedules queuing jobs to prevent job executions of violating the deadline. The adaptive scheduling methodology is same as initial scheduling.



4.2 Hybrid Resource Scheduling Service

The proposed scheduling service is based on multiple types of resources and is categorized of three main parts

Firstly, *Job profile Mgmt.* module is responsible for profiling job history, estimating execution time of job and deadline(in specific cases). In this module, we use a statistical method, PCA(Principal Component Analysis)[3] to analyze properties of jobs. Details about this module can be found in [4]. *Resource allocation* covers initial scheduling of jobs and adaptive scheduling of particular jobs according to both of the current resources' status and their deadline. In case of detecting delays on the scheduled jobs, it performs adaptive scheduling of the jobs which only hard to finish within their own deadline. This module serves the allocation services in this order**;** *grid*, *private* and then *public cloud*. *Cost estimation* module offers cost-efficient scheduling service on the public cloud. That is, the module chooses one or more instance(s) having minimal costs for leasing within the desired execution time.



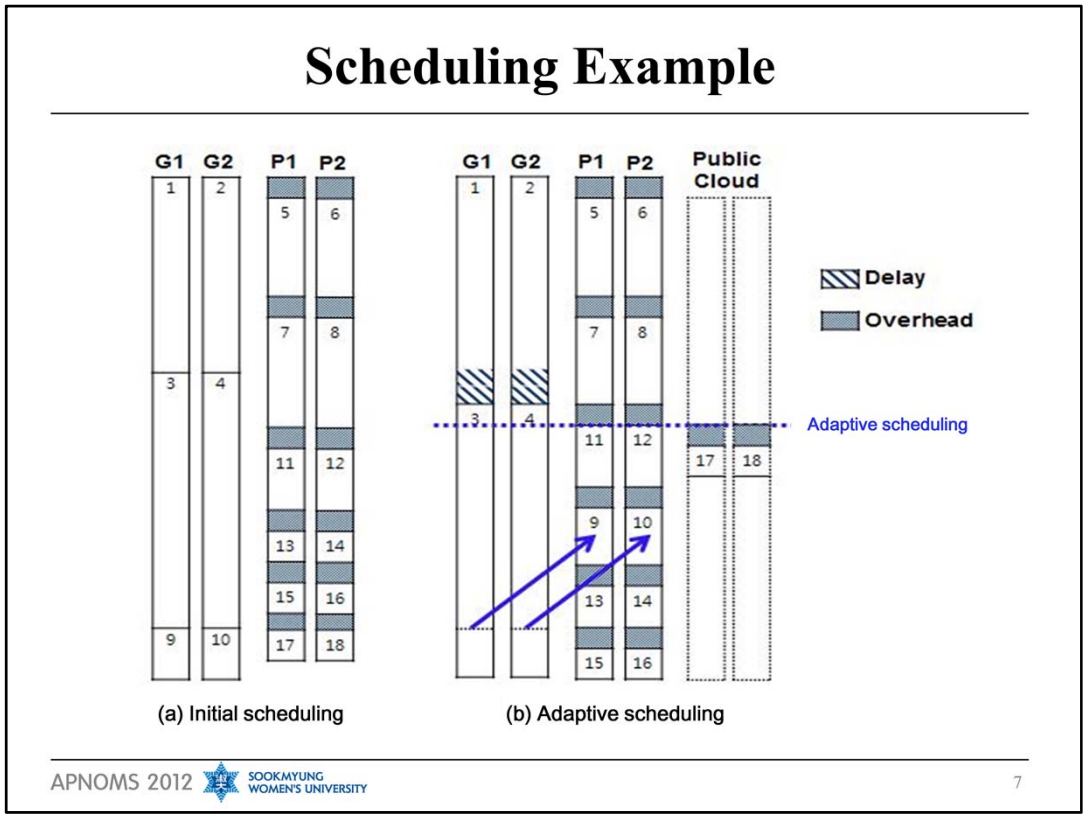
4.3 Scheduling Algorithm

In this research, as mentioned before, we target HTC (High Throughput Computing) applications which is also known as **BoT**(Bag of Tasks)[5]. Accordingly, we assume that an each user's request contains one or more sub-jobs and that each request has a stipulated maximum desired time in which all sub-jobs on the request need to be finished. We also assume that when a request is arrived, the sub-jobs belonging to the request are sorted in order of the estimated execution time. The proposed algorithm is separated in three parts as the followings:

1) ***Initial Scheduling*****:** For all sub-jobs of each request, schedule the jobs on the available resource. Suppose that grid has the highest priority and public cloud has the lowest one. The scheduler checks these resources in order of resource priority. In each resource, each job is expected to be finished within deadline under its estimated execution time via *Job profile management* module. If any specific job(s) has risks of violating deadline (it can be inferred by sum of the estimated execution time and current time), then scheduler will check the next resource for scheduling the job.

2) ***Adaptive scheduling*****:** For each scheduled job, *Monitoring* service periodically checks delay which is due to large difference between the estimated one and real execution time. If delay is occurred and violates deadline, it performs adaptive scheduling of all jobs queued, not in running. The way to adaptive schedule them is performed in the same way as the above 1).

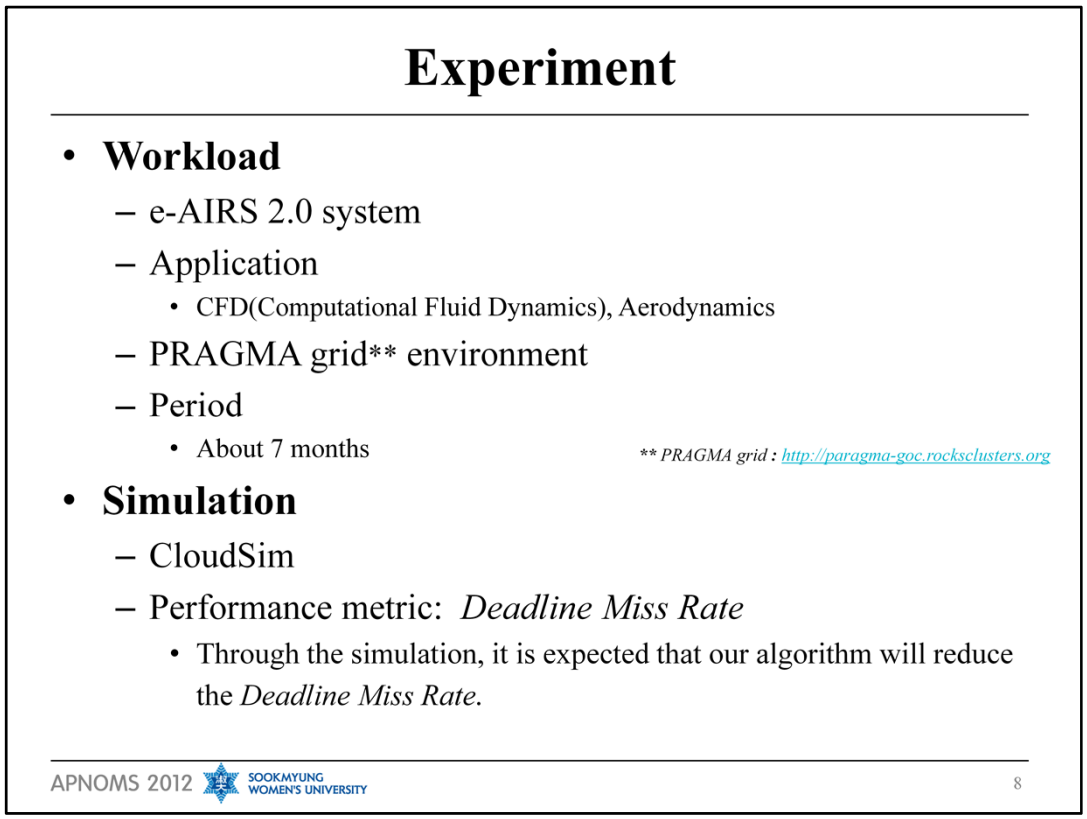
3) ***Cost-efficient scheduling*****:** In a case of schedules on public cloud (if the deadline is violated in both grid and private cloud), we choose instance type(s) of virtual machine which can support performance condition for the job. Among the candidates, we lastly choose the most cost-efficient instance.



5. Scheduling Example

In this section, we present an example of our scheduling algorithm. This figure shows the example of initial scheduling and adaptive scheduling. In this example, there are two cores in grid(G1, G2) and two cores in private cloud(P1, P2) as each bar means one core. Each part of the bar is corresponded to a sub-job. In addition, overhead is included in public/private cloud for each job. In case of initial scheduling (left of figure), jobs are submitted in order of job size. We labeled sequential numbers on the sub-jobs in decreasing order of their estimated execution time.

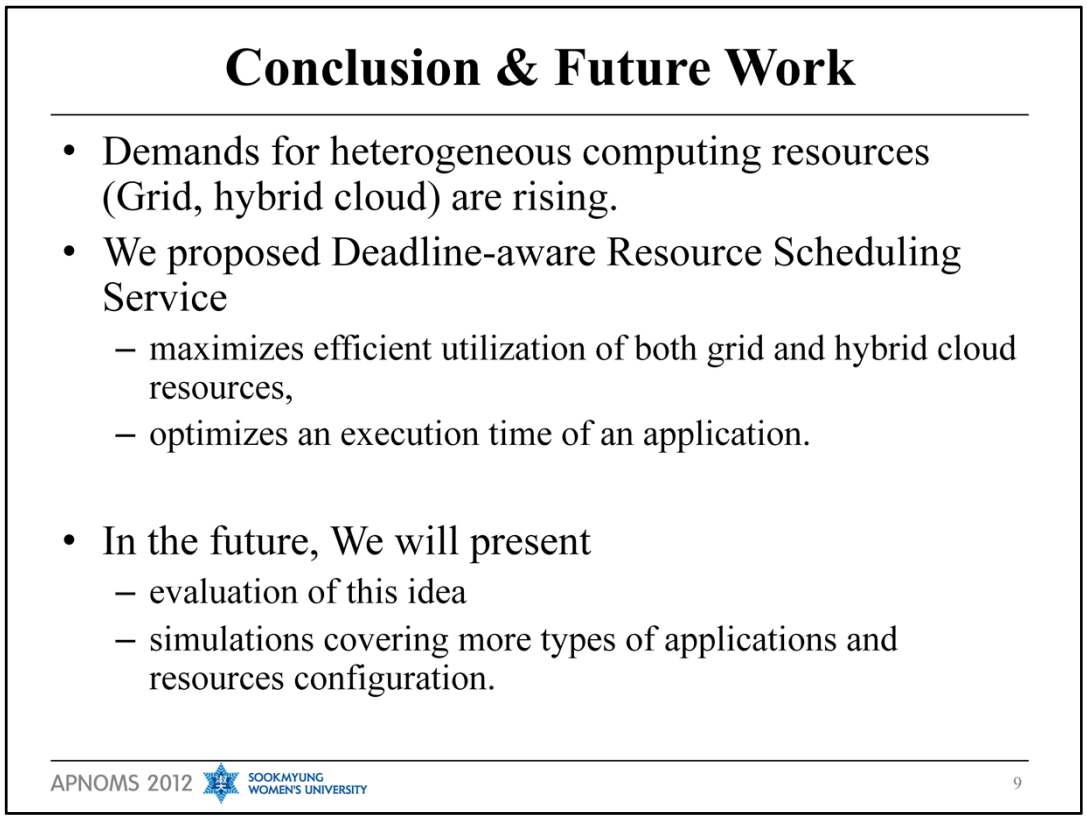
This schedule is achieved starting with the schedule on the left-hand side of cores in numerical order of label numbers as shown on **(a)**, left part of Figure 1. In case of job#5, it is hard to finish the job execution on both of G1 and G2 within its deadline. Therefore, it scheduled on P1 instead of G1 and G2. However, if some delays occurred, algorithm reschedules all of the queued jobs according to deadline. As shown on **(b)** part of the figure, delays occur on G1 and G2, then adaptive scheduling is enacted at the point where monitoring service recognized the delays (the dotted line on the fig.). Overall, job#9, job#10 will be scheduled on P1.



6. Experiment

To demonstrate our scheduling algorithm, we use workload of e-AIRS 2.0[6] system for CFD [8] applications of aerodynamics running over PRAGMA[7] grid environment. The job traces have been collected for approximately 7 months.

Using this traces, we recreated new job traces which are based on both grid and hybrid cloud via CloudSim[9]. We use these trained data for estimating execution time with its parameters and the details can be described in [4]. The metric for evaluation is *Deadline Miss Rate*. Through the simulation, it is expected that our algorithm will reduce the *Deadline Miss Rate* in comparisons with other scheduling schemes of the related works. Besides, it is expected to decrease total computational costs even on public clouds.



7. Conclusion and Future work

In this paper, we proposed a deadline-based scheduling service that maximizes efficient utilization of both grid and hybrid cloud resources. Our scheduling service is able to increase resource utilization by satisfying a deadline of each application, and optimize an execution time of the application by allocating public cloud resources under control of a monitor. Besides, the scheduling service ensures an application is finished within its deadline at minimum financial cost.

In the future, we will present evaluations of this idea and show simulations covering more types of applications and resource configurations.

References

[1] Ramakrishnan, L., Koelbel, C., Kee, Y., Wolski, R., Nurmi, D., Gannon, D., Obertelli, G., YarKhan, A., Mandal, A., Huang, T.M., Thyagaraja, K., and Zagorodnov, D., VGrADS: enabling e-Science workflows on grids and clouds with fault tolerance, In Proceedings of SC, 2009.
[2] Comet Cloud, http://www.cometcloud.org
[3] Jolliffe, I.T. “Principal Component Analysis”, Springer Verlag. 2002.
[4] Seoyoung Kim, Jung-in Koh, Yoonhee Kim “Profile Analysis of Scientific Application Using PCA for Cloud Computing”, Proceeding of the 2011 IEEE Seoul Section, Korea Univ., pp.79 -82, Dec. 3, 2011.
[5] W.Cirne, et al, “Grid Computing for Bag of Tasks Applications,” Proceedings of the 3rd IFIP Conference on E-Commerce, E-Business and E-Government, September 2003.
[6] Kim, Y.; Kim, E.-k.; Kim, J. Y.; Cho, J.-h.; Kim, C. & Cho, K. W. (2006). e-AIRS: An e-Science Collaboration Portal for Aerospace Applications. HPCC 2006, LNCS (Lecture Note in Computer Science), Vol.4208, 813-822, 0302-9743
[7] PRAGMA Grid and Cloud Monitoring homepage, http://pragma-goc.rockscluster.org
[8] Computational Fluid Dynamics, http://www.cfd-online.com
[9] Rodrigo N. Calheiros, et al, “CloudSim: A Toolkit for Modeling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms, Software: Practice and Experience(SPE)”, Vol. 41, Num.1, pp.23-50, ISSN: 0038-0644, Wiley Press, New York, USA, January, 2011
[10] US TeraGrid, http://www.teragrid.org