An Enhanced Workflow Scheduling Algorithm in Cloud Computing

Nora Almezeini and Alaaeldin Hafez
College of Computer and Information Sciences, King Saud University, Riyadh, Saudi Arabia

Keywords: Cloud Computing, Scheduling Algorithm, Fault Tolerant, Pricing Models, Workflow.

Abstract: Cloud Computing has gained high attention by provisioning resources and software as a service. Throughout the years, the number of users of clouds is increasing and thus will increase the number of tasks and load in the cloud. Therefore, scheduling tasks efficiently and dynamically is a critical problem to be solved. There are many scheduling algorithms that are used in cloud computing but most of them are concentrating on minimizing time and cost and some of them concentrate on increasing fault tolerance. However, very few scheduling algorithms that considers time, cost, and fault tolerance at the same time. Moreover, Considering pricing models in developing scheduling algorithms to provide cost-effective fault tolerant techniques is still in its infancy. Therefore, analysing the impact of the different pricing models on scheduling algorithm will lead to choosing the right pricing model that will not affect the cost. This paper proposes developing a scheduling algorithm that combines these features to provide an efficient mapping of tasks and improve Quality of Service (QoS).

1 INTRODUCTION

Cloud computing is considered as a distributed system that offers services to the Internet users through service providers such as Amazon, Google, Apple, Microsoft, and others. Cloud computing uses Internet technologies to offer elastic services that support dynamic access to the computing resources and support variable workloads.

However, cloud computing still requires more scientific research across a variety of different topics in order to gain its full benefits. One of the important topics that need to be researched is the performance efficiency of scheduling where workflow scheduling focuses on efficiently mapping tasks to appropriate resources. Finding optimal solution in cloud computing is considered as NP-Complete Problem (Kalra and Singh, 2015). Each scheduling algorithm based on one or more strategy. The most important strategies or objectives commonly used are time, cost, energy, Quality of Service (QoS), and fault tolerance (Chandrashekar, 2015). There are many scheduling algorithms have been applied in cloud systems such as Min-Min, Max-Min, Genetic Algorithm, Particle Swarm Optimization (PSO), and Heterogeneous Earliest Finish Time (HEFT) (Rahman et al., 2013, Chaudhary and Kumar, 2014b, Devipriya and Ramesh, 2013). Those algorithms and others have been concentrating on minimizing the overall completion time of the schedule (makespan) and the cost. However, very few algorithms have taken fault tolerance in consideration at the same time. (Chandrashekar, 2015)

Developing a fault tolerant system is important to provide proper and continuous work even if failures are detected. There are two types of fault tolerant techniques that can be implemented in the cloud computing services: Proactive techniques that predict failures and prevent them by replacing suspected components with working components, and Reactive techniques try to reduce the impact of failure after it occurs and recover it. (Nazari Cheraghlou et al., 2015, Sarmila et al., 2015)

Applying fault tolerant techniques in the system in order to detect and recover any failure attack the system is very important. However, it costs a lot. For example, replication technique needs more resources that will increase the cost of scheduling. Also, considering pricing models in developing scheduling algorithms to provide cost-effective fault tolerant techniques is still in its infancy. Therefore, it is important to analyse the impact of the fault tolerant technique on cost by considering the pricing models that will lead to a cost-effective fault tolerance. (Chandrashekar, 2015)

Pricing in cloud environment is a very important concept since cloud providers are mostly concerns on maximizing revenue and profitability while the

Almezeini, N. and Hafez, A.
An Enhanced Workflow Scheduling Algorithm in Cloud Computing.
In *Proceedings of the 6th International Conference on Cloud Computing and Services Science (CLOSER 2016) - Volume 2*, pages 67-73
ISBN: 978-989-758-182-3

end users are more focused on cost-effectiveness in addition to QoS. It is challenging to choose the right pricing model that satisfies all parties. There are two general pricing models in cloud computing: Fixed Pricing Model (Pay-per-Use) where the consumers are charged once for a computing capacity, and Dynamic Pricing Model to provision fair resource allocation with service differentiation.(Arshad et al., 2015, AlRoomi et al., 2013)

The aim of this research is to show that it is necessary to produce an enhanced workflow scheduling system that combines the previous main goals: reducing makespan and cost, increasing the fault tolerance and robustness, and applying the proper pricing model. Thus it will satisfy the provider and consumer in the same time.

The rest of this short paper will talk about the workflow scheduling and the most popular scheduling algorithms used in clouds. Then, a brief description of the important techniques of fault tolerant that can be used in cloud computing. Also, the pricing models of cloud computing will be described in section 4. Then, research objectives and methodology of the proposed research are presented in section 5 and 6.

2 WORKFLOW SCHEDULING

Workflow scheduling assigns tasks based on their dependencies to shared resources that are controlled by a workflow scheduler (Kalra and Singh, 2015). The aim of workflow scheduling is to obtain the desired QoS. Workflow scheduling has a lot of advantages. It increases computing performance and throughput which increase the user satisfaction and reduces the execution cost and time. However, it is important to map the tasks efficiently to given resources in order to provide high QoS to users. Therefore, workflow scheduling is considered a Non-deterministic polynomial (NP)-Complete problem (Kalra and Singh, 2015, Chandrashekar, 2015).

There are many workflow scheduling methodologies and techniques are used to map tasks to resources which can be classified into two types: *Heuristics* and *Metaheuristics*. Heuristic techniques are based on exhaustive search to provide an approximate optimal solution although the operating cost and complexity of generating schedules is very high. Examples of heuristic techniques are Min-Min and Max-Min algorithms. On the other hand, metaheuristic techniques are able to solve large and complex problems effectively and efficiently in reasonable time. Some of the well known metaheuristic techniques are Genetic Algorithms (GA), and Ant Colony Algorithms (ACO).(Kalra and Singh, 2015, Chandrashekar, 2015)

Each scheduling algorithm should be based on one or more strategies. The most important strategies or objectives commonly used are time, cost, energy, QoS, and fault tolerance (Chandrashekar, 2015). Moreover, workflow scheduling has two types of planning schemes: *static* and *dynamic* (Chandrashekar, 2015, Devipriya and Ramesh, 2013). If the number of tasks is known beforehand, static scheme is applied so the tasks are mapped at the compile time. On the other hand, the dynamic scheme is applied when the tasks are arrived in a dynamic manner so some assumptions are provided before execution and scheduling decisions are made just in time.

Famous cloud providers are heavily reliant on big data analytics such as Google that developed Map-Reduce framework for processing big data workflows. Also, Yahoo and Facebook use Hadoop which is the open source implementation of Map-Reduce framework. Hadoop has four choices of scheduling algorithms: FIFO, Fair scheduler, Capacity scheduler and Dynamic scheduler. Moreover, IBM blue cloud is based on Xen and Hadoop clusters. Amazon EC2 uses FIFO, default algorithm in Hadoop. (Wang et al., 2014, Chandrashekar, 2015)

2.1 Workflow Scheduling Algorithms

2.1.1 Min-Min Scheduling Algorithm

Min-Min scheduling algorithm was proposed (Rahman et al., 2013) for scheduling tasks in grid projects. It is simple and fast and works as the base for most of cloud scheduling algorithms. It provides better performance by scheduling the task with minimum size to the resource that has the minimum completion time (MCT) for all unmapped tasks (Chaudhary and Kumar, 2014b). Finally, this task is removed from the set of unmapped tasks and the process is repeated again by Min-Min algorithm till every task is assigned.

However, the performance of this algorithm is not perfect since it schedules the small tasks at first (Devipriya and Ramesh, 2013). Also, it works well if the number of smaller tasks is greater than the number of larger tasks. Min-Min uses a single resource; therefore, it is unable to execute tasks concurrently.

QoS Guided Min-Min Algorithm (Chaudhary and Kumar, 2014a) is a Min-Min algorithm with Quality of Service (QoS) constraints added to it such as bandwidth, time, and memory. Tasks with high QoS request parameters are assigned to resources first. In this, resources are utilized in higher rate and they are scaled elastically in execution.

Segmented Min-Min Scheduling Algorithm (Chaudhary and Kumar, 2014a) where the tasks are sorted based on the Estimated Time to Complete (ETC). Then it creates segments with the equal size using task list partitioning scheme. Segments with larger tasks are scheduled and executed before the smaller tasks. In each segment, Min-Min algorithm is applied to assign tasks to resources.

Double Min-Min Algorithm (Chaudhary and Kumar, 2014a, Kong et al., 2011) reselects the task that is greater than mean Completion Time and then schedules the reselected task again using Min-Min algorithm. This will provide better load balancing and resource utilization.

2.1.2 Max-Min Scheduling Algorithm

Max-Min scheduling algorithm is very similar to Min-Min except that Max-Min selects the task that has the maximum MCT and assign it to the resources having minimum execution time (Rahman et al., 2013, Chaudhary and Kumar, 2014b, Devipriya and Ramesh, 2013) . This algorithm gives the priority to large tasks rather than small ones and executes many small tasks while executing the large task concurrently. Therefore, Max-Min performs better than Min-Min algorithm if the number of short tasks is more than long tasks (Devipriya and Ramesh, 2013).

Many improvements have been applied to these algorithms in order to minimize their drawbacks especially that they are not useful for large scale distributed systems. (Devipriya and Ramesh, 2013) has improved the Max-Min algorithm by assigning the task with maximum execution time to the resource that has minimum completion time instead of assigning task with maximum completion time to the resource which produce minimum execution time in the original Max-Min algorithm. The goal is to assign the largest task to the slowest resource in order to give the opportunity to small tasks to be finished by high speed resources concurrently. This improved algorithm has reduced the makespan. However, it concerned only with the number of resources and tasks.

In (Kaur, 2014), the authors have improved the Max-Min algorithm as in (Devipriya and Ramesh, 2013) and combine it dynamically with Ant Colony algorithm as hybrid approach. The improved Max-Min algorithm produces an optimal solution in the initial stage, but it reduces after some time. However, the searching speed during the early stage in the Ant Colony algorithm is very slow due to the lacking of pheromones, and then the speed of optimal solution increases quickly after pheromones reach a certain degree. Therefore, the aim of the integration in (Kaur, 2014) is to benefit from Max-Min algorithm in the initial stage and then get the optimal solution by Ant Colony algorithm in last stage.

Resource Aware Scheduling Algorithm (RASA) (Devipriya and Ramesh, 2013, Chaudhary and Kumar, 2014a) is a hybrid algorithm based on Min-Min and Max-Min algorithms. They are applied alternatively in order to avoid their main drawbacks. This algorithm calculates the completion time of each task on the available resources and then applies Max-Min and Min-Min algorithms alternatively to take advantage of both of them and avoid their drawbacks.

2.1.3 Genetic Algorithm

Genetic Algorithm is a metaheuristic technique that provides useful solution to optimization problems by applying the principles of evolution. A Genetic Algorithm depends on two techniques to be effective (Rahman et al., 2013):

- *Exploitation*: exploit the best solution from past results.
- *Exploration*: explore the new areas of solution space.

Genetic Algorithm begins by initializing a ***population*** with random candidate solutions called individuals. Each individual is evaluated by a ***fitness function*** which can be different according to the given optimization objective. Then a proportion of population is ***selected*** to reproduce a new generation. After that, two main genetic operators are used to generate new generation population. These two operators are: ***crossover*** and ***mutation***. (Rahman et al., 2013, Kumar and Verma, 2012)

Using genetic algorithms in scheduling is a powerful approach since they provide better solutions with the increase of population size and number of generations. However, the random generation of initial population leads to schedules that are not so much fit, so when these schedules are mutated with each other, there are a very low probability to produce better child than themselves. Therefore, many researches have worked on improving genetic algorithms especially in the early steps in order to increase the performance.

In (Kumar and Verma, 2012), the authors have improved genetic algorithm by using the Min-Min and Max-Min algorithms in generating initial population. This will provide better initial population and better solutions than initializing population randomly in standard genetic algorithms.

(Liu et al., 2014) proposed an algorithm that combines between genetic algorithm and Ant Colony algorithm. They benefit from the strong global search capability of genetic algorithms in the early stages of Ant Colony algorithm to generate the initial population then convert it to initial pheromone of ACO to provide the optimal solution. This integration of global search capability and high accuracy shows the good performance of task scheduling and load balancing.

(Wang et al., 2014) have proposed a scheduling algorithm through improved genetic algorithm in order to minimize the makespan and load balancing. They chose greedy algorithm to initialize the population. This proposed algorithm showed better performance in load balancing than genetic algorithm.

2.1.4 Other Scheduling Algorithms

Particle Swarm Optimization (PSO) based Heuristic (Pandey et al., 2010) is a scheduling algorithm that considers computation cost and data transmission cost to schedule applications to cloud resources. The evaluation results show that PSO can obtain greater cost savings and good workload distribution to resources in the cloud.

Heterogeneous Earliest Finish Time Algorithm (HEFT) (Chaudhary and Kumar, 2014a, Wieczorek et al., 2005) determines the average execution time for each task and also the average communication time between the resource of two tasks. After that, it orders tasks by rank function so the task with a higher rank value is given a higher priority. So the tasks are scheduled based on the priority order and every task is assigned to resources that complete it at earliest time.

Scalable Heterogeneous Earliest Finish Time Algorithm (SHEFT)(Chaudhary and Kumar, 2014a, Hirales-Carbajal et al., 2012) works similar to HEFT in addition to scaling resources elastically at runtime. Therefore, it obtains an optimized execution time.

Round Robin Algorithm (Chaudhary and Kumar, 2014a, Chaudhary and Singh Chhillar, 2013) is a static algorithm that determines a time slot for each task, and suspends the task if it is not completed during this time slot. Once all the tasks have finished their time slots the first uncompleted task will get again another allocation and the cycle will repeat. However, this will overload the nodes in some times and under load at other times. Therefore, the *Weighted Round Robin Algorithm* (Chaudhary and Kumar, 2014a, Chaudhary and Singh Chhillar, 2013) assigns weights in order to each task so the tasks are allocated to resources based on their weights and time slots for optimal utilization of resources.

3 FAULT TOLERANT MECHANISMS IN CLOUDS

Fault is a defect that affects the system and changes its status to be unable to continue to work. In general, there are three types of faults (Sarmila et al., 2015, Nazari Cheraghlou et al., 2015):

- Transient: a fault that vanished when it fixed.
- Intermittent: a fault that could appears again and again.
- Permanent: a fault that cannot be fixed.

Faults in cloud computing are different based on computing resources. (Kumar et al., 2015) have indicated that fault in cloud computing could be Network Fault, Physical Fault, Processor Fault, and Service Expiry Fault. Moreover, (Chandrashekar, 2015) mentioned that faults could be variations in performance of resources, or unavailable files.

It is obvious that developing a fault tolerant system is important to provide proper and continuous work even if failures are detected. There are different fault tolerant techniques that can be implemented in the cloud computing services. These techniques can be classified into two categories: Proactive and Reactive techniques. Proactive techniques avoid recovery after the occurrence of fault. It predicts failures and prevents them by replacing suspected components with working components. This technique is more efficient than reactive but not always the predictions are accurate. Reactive techniques try to reduce the impact of failure after it occurs and recover it. It is more reliable. Therefore, it is mostly used. (Sarmila et al., 2015, Nazari Cheraghlou et al., 2015)

3.1 Proactive Fault Tolerant Techniques

Self-heading: this technique handles the failed instances of an application on multiple virtual machines, since multiple instances of the application are running on multiple VMs.

Preemptive Migration: this technique is based on monitoring and analysing the application continuously according to the feedback-loop mechanism.

3.2 Reactive Fault Tolerant Techniques

Replication: tasks are replicated on different resources in order to be able to recover quickly from failures.

Resubmission: this technique resubmits the failed task to alleviate failures.

Check Point/Restart: while the application is running, it makes checkpoints in different parts. This technique restarts the application at the last checkpoint achieved when a failure occurs.

Rescue Workflow: this technique ignores the failed tasks and allows the workflow to continue until no more forward steps can be made and becomes impossible to continue without handling the failures.

User Defined Exception Handling: this technique allows the user to specify a predefined action for particular failures in a workflow.

Job Migration: when a failure occurs, this technique will send the task to other similar virtual machine.

4 PRICING MODELS IN CLOUDS

Cloud providers are mostly concerns on maximizing revenue and profitability which can be employed by several pricing models. On the other hand, end users are more focused on Quality of Service (QoS), availability and usability of resources, and cost-effectiveness. It is seriously challenging cloud providers to keep balance between these two important purposes and choose the right pricing model that satisfies all parties.(Arshad et al., 2015, AlRoomi et al., 2013)

Pricing models in cloud computing can be classified into two general models:

4.1 Fixed Pricing (Pay-Per-Use)

Cloud providers mainly used this approach in order to provision their services to consumers. Consumers are charged once for a computing capacity. This strict model has some drawbacks since the user may not use all the resources he charged for and also the provider will not consider the QoS and other satisfaction measures after charging the user. Examples of cloud providers that employ this model is Amazon on-demand instances and Google App Engine.(AlRoomi et al., 2013)

Pay per use fixed pricing model is suitable for IaaS and PaaS. It has two sub categories: Pay for resources and Subscription. In Pay for resources technique, users will be charged for using storage or bandwidth size of resource. While in Subscription technique, the user will subscribe to a service provider with a fixed price per unit for a long time.(Arshad et al., 2015)

4.2 Dynamic Pricing

Whereas the nature of cloud environment is dynamic and the number of cloud users increasing, vendors have adopted the modification from fixed pricing to dynamic pricing in order to provision fair resource allocation with service differentiation. Amazon's spot instances pricing model was the first concept of dynamic pricing model which profits from the available unused resources in the data centres executing the requests of on-demand instances.(Arshad et al., 2015)

5 RESEARCH OBJECTIVES

Based on the brief literature review, it is obvious that scheduling tasks efficiently and dynamically is a critical problem to be solved. Throughout the years, the number of tasks in the cloud system and dependencies between tasks are increasing. This will affect the makespan and the cost. Therefore, it is necessary to develop a scheduling algorithm that prevents these challenges and provide better performance in time and cost.

Moreover, the system will be susceptible to failures and performance variations. Thus, making the workflow scheduling algorithm efficient and fault tolerant is very important in order to overcome the drawbacks of previous algorithms.

Pricing in cloud environment is a very important concept. Considering pricing models in developing scheduling algorithms to provide cost-effective fault tolerant techniques is still in its infancy. Therefore, analysing the impact of the different pricing models on scheduling algorithm will lead to choosing the right pricing model that will not affect the cost.

The goal of this research is to propose an enhanced workflow scheduling system that combines the main goals: reducing makespan and cost, increasing the fault tolerance and robustness, and applying the proper pricing model. Thus it will satisfy the provider and consumer in the same time.

6 METHODOLOGY

As a first step, the research methodology starts with studying the domain of cloud computing and techniques and mechanisms of cloud computing. Following this step, we will investigate previous and current studies and works related to our research. As a major step towards achieving our aim, we will develop a scheduling algorithm that improves the performance of the current scheduling algorithms in terms of reducing time and cost. Secondly, a proper fault tolerant mechanism will be integrated to the developed scheduling algorithm so that does not affect the time and cost and increases the reliability and robustness of the system. Finally, analysing the impact of the different pricing models on the proposed scheduling algorithm will lead to choosing the right pricing model for the proposed algorithm in which it will not affect the previous steps.

A simulator such as CloudSim will be used to evaluate the performance of the proposed algorithm since it is very difficult to conduct large scale experiments on real cloud infrastructures as well as it is time consuming and costly. The proposed algorithm will be evaluated first in terms of time and cost. The results will be compared with previous algorithms that consider the same goal. Then we will evaluate the algorithm in terms of fault tolerant and compare the results with the algorithms that apply the same techniques. After that, we will analyse the contribution of the selected pricing model in reducing the cost of applying the fault tolerant mechanism in the proposed algorithm.

Finally, the proposed algorithm will be evaluated as whole and analysed in terms if it achieved the desired objective and if it increased the efficiency of workflow scheduling.

7 CONCLUSIONS

It is explicit that mapping tasks efficiently to given resources in order to ensure Quality of Service (QoS) is a major challenge. If this is not achieved, the user will hesitate to join the cloud and pay. Minimizing makespan and cost, increasing fault tolerance, and choosing the proper pricing model are very important objectives that will improve the QoS. Therefore, combining the features in a workflow scheduling is necessary to provide efficiency and gain satisfaction from providers and consumers.

ACKNOWLEDGEMENTS

This research project is supported by a grant from the Deanship of Graduate Studies, King Saud University, Saudi Arabia.

REFERENCES

Alroomi, M., Alebrahim, S., Buqrais, S. & Ahmad, I. 2013. Cloud Computing Pricing Models: A Survey. *International Journal Of Grid And Distributed Computing,* 6**,** 93-106.

Arshad, S., Ullah, S., Khan, S. A., Awan, M. D. & Khayal, M. S. H. A Survey Of Cloud Computing Variable Pricing Models. Evaluation Of Novel Approaches To Software Engineering (Enase), 2015 International Conference On, 29-30 April 2015 2015. 27-32.

Chandrashekar, D. P. 2015. *Robust And Fault-Tolerant Scheduling For Scientific Workflows In Cloud Computing Environments.*

Chaudhary, D. & Kumar, B. 2014a. An Analysis Of The Load Scheduling Algorithms In The Cloud Computing Environment: A Survey. *2014 9th International Conference On Industrial And Information Systems (Iciis).*

Chaudhary, D. & Kumar, B. 2014b. Analytical Study Of Load Scheduling Algorithms In Cloud Computing. *2014 International Conference On Parallel, Distributed And Grid Computing.*

Chaudhary, D. & Singh Chhillar, R. 2013. A New Load Balancing Technique For Virtual Machine Cloud Computing Environment. *International Journal Of Computer Applications,* 69**,** 37-40.

Devipriya, S. & Ramesh, C. 2013. Improved Max-Min Heuristic Model For Task Scheduling In Cloud. *2013 International Conference On Green Computing, Communication And Conservation Of Energy (Icgce).*

Hirales-Carbajal, A., Tchernykh, A., Yahyapour, R., González-García, J. L., Röblitz, T. & Ramírez-Alcaraz, J. M. 2012. Multiple Workflow Scheduling Strategies With User Run Time Estimates On A Grid. *Journal Of Grid Computing,* 10**,** 325-346.

Kalra, M. & Singh, S. 2015. A Review Of Metaheuristic Scheduling Techniques In Cloud Computing. *Egyptian Informatics Journal*.

Kaur, R. 2014. Hybrid Improved Max Min Ant Algorithm For Load Balancing In Cloud. *In:* Ghumman, N. (Ed.) *International Conference On Communication, Computin G & Systems.*

Kong, X., Lin, C., Jiang, Y., Yan, W. & Chu, X. 2011. Efficient Dynamic Task Scheduling In Virtualized Data Centers With Fuzzy Prediction. *Journal Of Network And Computer Applications,* 34**,** 1068-1077.

Kumar, P. & Verma, A. 2012. Independent Task Scheduling In Cloud Computing By Improved Genetic Algorithm. *International Journal Of Advanced Research In Computer Science And Software Engineering (Ijarcsse),* 2**,** 111-114.

Kumar, S., Singh Rana, D. & Chandra Dimri, S. 2015. Fault Tolerance And Load Balancing Algorithm In Cloud Computing: A Survey. *International Journal Of Advanced Research In Computer And Communication Engineering,* 4**,** 92-96.

Liu, C.-Y., Zou, C.-M. & Wu, P. 2014. A Task Scheduling Algorithm Based On Genetic Algorithm And Ant Colony Optimization In Cloud Computing. *2014 13th International Symposium On Distributed Computing And Applications To Business, Engineering And Science*.

Nazari Cheraghlou, M., Khadem-Zadeh, A. & Haghparast, M. 2015. A Survey Of Fault Tolerance Architecture In Cloud Computing. *Journal Of Network And Computer Applications*.

Pandey, S., Wu, L., Guru, S. M. & Buyya, R. 2010. A Particle Swarm Optimization-Based Heuristic For Scheduling Workflow Applications In Cloud Computing Environments. *2010 24th Ieee International Conference On Advanced Information Networking And Applications*.

Rahman, M., Hassan, R., Ranjan, R. & Buyya, R. 2013. Adaptive Workflow Scheduling For Dynamic Grid And Cloud Computing Environment. *Concurrency And Computation: Practice And Experience,* 25**,** 1816-1842.

Sarmila, G. P., Gnanambigai, N. & Dinadayalan, P. 2015. Survey On Fault Tolerant — Load Balancing Algorithmsin Cloud Computing. *2015 2nd International Conference On Electronics And Communication Systems (Icecs)*.

Wang, T., Liu, Z., Chen, Y., Xu, Y. & Dai, X. 2014. Load Balancing Task Scheduling Based On Genetic Algorithm In Cloud Computing. *2014 Ieee 12th International Conference On Dependable, Autonomic And Secure Computing*.

Wieczorek, M., Prodan, R. & Fahringer, T. 2005. Scheduling Of Scientific Workflows In The Askalon Grid Environment. *Acm Sigmod Record,* 34.