Designing of Optimized Load balancing algorithms for Grid computing system

Poonam[1], Kirti Gautam[2]

[1]M.Tech Scholar, CSE Department, Jind Institute of Engineering & Technology, Haryana

[2]Assistant Professor, CSE Department, Jind Institute of Engineering & Technology, Haryana

***Abstract*—Grid computing is an infrastructure that involves the integrated and collaborative use of computers, networks, databases and scientific instruments owned and managed by multiple organizations. A major contribution is to utilize the dynamism of virtualized Grid resources in various workflow management operations. Several algorithms are proposed throughout the dissertation, each focusing on a different aspect of the larger problem, from monitoring individual services, to placing a new service workflow in the Grid, to dynamically reallocating resources across different services to satisfy demands and reduce costs. The goal is to add an end-to-end solution to the Grid provider's offerings to workflow owners so that the latter can host their workflows in the Grid smoothly without worrying about managing the underlying Grid resources themselves. We show through experimental results, from both real world cluster trace logs and synthetic data, that the proposed approaches can perform various management tasks for service workflows efficiently.**

***Keywords*—ACO, Grid Computing, MIPS rating.**

I. INTRODUCTION

Grid computing is all the rage. "It's become the phrase du jour," says Gartner senior analyst Ben Pring, echoing many of his peers. The problem is that (as with Web 2.0) everyone seems to have a different definition. As a metaphor for the Internet, "the Grid" is a familiar cliché, but when combined with "computing," the meaning gets bigger and fuzzier. Some analysts and vendors define Grid computing narrowly as an updated version of utility computing: basically virtual servers available over the Internet. Others go very broad, arguing anything you consume outside the firewall is "in the Grid," including conventional outsourcing.

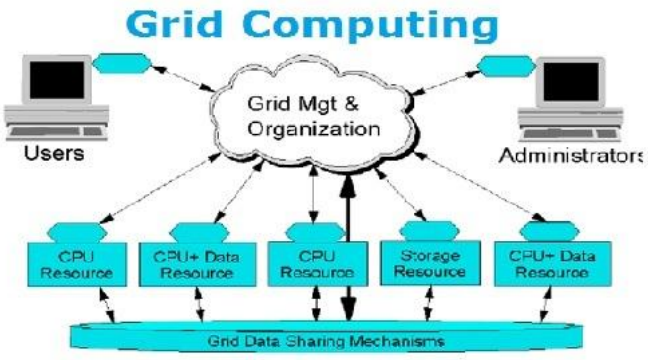

Figure 1

Grid computing comes into focus only when you think about what IT always needs: a way to increase capacity or add capabilities on the fly without investing in new infrastructure, training new personnel, or licensing new software. Grid computing encompasses any subscription-based or pay-per-use service that, in real time over the Internet, extends IT's existing capabilities.

Grid computing is at an early stage, with a motley crew of providers large and small delivering a slew of Grid-based services, from full-blown applications to storage services to spam filtering. Yes, utility-style infrastructure providers are part of the mix, but so are SaaS (software as a service) providers such as Salesforce.com. Today, for the most part, IT must plug into Grid-based services individually, but Grid computing aggregators and integrators are already emerging.

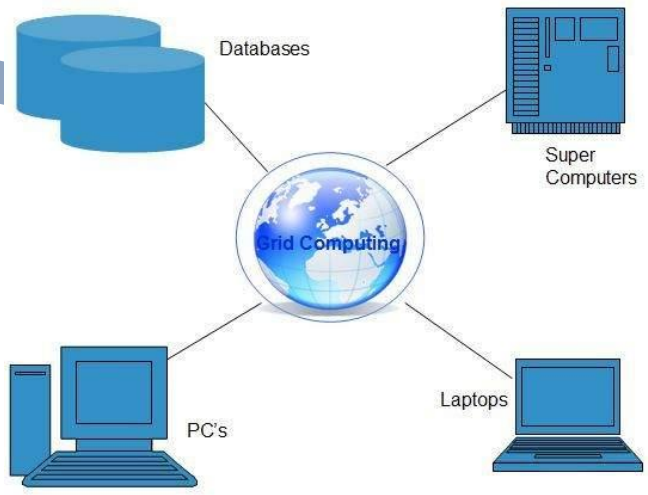

Figure 2

A scientist studying proteins logs into a computer and uses an entire network of computers to analyze data. A businessman accesses his company's network through a PDA in order to forecast the future of a particular stock. An Army official accesses and coordinates computer resources on three different military networks to formulate a battle strategy. All of these scenarios have one thing in common: They rely on a concept called grid computing.

At its most basic level, grid computing is a computer network in which each computer's resources are shared with every other computer in the system. Processing power, memory and data

storage are all community resources that authorized users can tap into and leverage for specific tasks. A grid computing system can be as simple as a collection of similar computers running on the same operating systemor as complex as inter-networked systems comprised of every computer platform you can think of.

Today, with such Grid-based interconnection seldom in evidence, Grid computing might be more accurately described as "sky computing," with many isolated Grids of services which IT customers must plug into individually. On the other hand, as virtualization and SOA permeate the enterprise, the idea of loosely coupled services running on an agile, scalable infrastructure should eventually make every enterprise a node in the Grid. It's a long-running trend with a far-out horizon. But among big metatrends, Grid computing is the hardest one to argue with in the long term.

II. RESOURCE MANAGEMENT ISSUE

Resource management is a core function required of any man-made system. It affects the three basic criteria for system evaluation: performance, functionality and cost. Inefficient resource management has a direct negative effect on performance and cost. It can also indirectly affect system functionality. Some functions the system provides might become too expensive or ineffective due to poor performance.

A Grid computing infrastructure is a complex system with a large number of shared resources. These are subject to unpredictable requests and can be affected by external events beyond your control. Grid resource management requires complex policies and decisions for multi-objective optimization. It is extremely challenging because of the complexity of the system, which makes it impossible to have accurate global state information. It is also subject to incessant and unpredictable interactions with the environment.

The strategies for Grid resource management associated with the three Grid delivery models, Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS), differ from one another. In all cases, the Grid services providers are faced with large, fluctuating loads that challenge the claim of Grid elasticity. In some cases, when they can predict a spike can be predicted, they can provision resources in advance. For example, seasonal Web services may be subject to spikes.

For an unplanned spike, the situation is slightly more complicated. You can use Auto Scaling for unplanned spike loads, provided there's a pool of resources you can release or allocate on demand and a monitoring system that lets you decide in real time to reallocate resources. Auto Scaling is supported by PaaS services such as Google App Engine. Auto Scaling for IaaS is complicated due to the lack of standards.

In the Grid, where changes are frequent and unpredictable, centralized control is unlikely to provide continuous service and performance guarantees. Indeed, centralized control can't provide adequate solutions to the host of Grid management policies you have to enforce.

Autonomic policies are of great interest due to the scale of the system, the large number of service requests, the large user population and the unpredictability of the load. The ratio of the mean to the peak resource needs can be large.

Policies and mechanisms

A policy typically refers to the principal guiding decisions, whereas mechanisms represent the means to implement policies. Separating policies from mechanisms is a guiding principle in computer science. Butler Lampson and Per Brinch Hansen offer solid arguments for this separation in the context of OS design.

You can loosely group Grid resource management policies into five classes:

The explicit goal of an admission control policy is to prevent the system from accepting workloads in violation of high-level system policies. For example, a system may not accept an additional workload that would prevent it from completing work already in progress or contracted. Limiting the workload requires some knowledge of the global system state. In a dynamic system, this information is often obsolete at best.

Capacity allocation means allocating resources for individual instances. An instance is a service activation. Locating resources that are subject to multiple global optimization constraints requires you to a search a large space when the state of individual systems is changing so rapidly.

You can perform load balancing and energy optimization locally, but global load-balancing and energy-optimization policies encounter the same difficulties as the ones already discussed. Load balancing and energy optimization are correlated and affect the cost of providing the services.

The common meaning of the term load balancing is that of evenly distributing the load to a set of servers. For example, consider the case of four identical servers, A, B, C and D. Their relative loads are 80 percent, 60 percent, 40 percent and 20 percent, respectively, of their capacity. Perfect load balancing would result in all servers working with the same load—50 percent of each server's capacity.

In Grid computing, a critical goal is minimizing the cost of providing the service. In particular, this also means minimizing energy consumption. This leads to a different meaning of the term load balancing. Instead of having the load evenly distributed among all servers, we want to concentrate it and use the smallest number of servers while switching the others to standby mode, a state in which a server uses less energy. In our example, the load from D will migrate to A and the load from C will migrate to B. Thus, A and B will be loaded at full capacity, whereas C and D will be switched to standby mode.

Quality of service is that aspect of resource management that's probably the most difficult to address and, at the same time, possibly the most critical to the future of Grid computing. Resource management strategies often jointly target performance and power consumption.

Dynamic voltage and frequency scaling (DVFS) techniques such as Intel SpeedStep and AMD PowerNow lower the voltage and the frequency to decrease power consumption. Motivated initially by the need to save power for mobile devices, these techniques have migrated to virtually all processors, including those used in high-performance servers. As a result of lower voltages and frequencies, the processor

performance decreases. However, it does so at a substantially slower rate than the energy consumption.

Virtually all optimal or near-optimal mechanisms to address the five policy classes don't scale up. They typically target a single aspect of resource management, such as admission control, but ignore energy conservation. Many require complex computations that can't be done effectively in the time available to respond. Performance models are complex, analytical solutions are intractable, and the monitoring systems used to gather state information for these models can be too intrusive and unable to provide accurate data.

Therefore, many techniques are concentrated on system performance in terms of throughput and time in system. They rarely include energy tradeoffs or QoS guarantees. Some techniques are based on unrealistic assumptions. For example, capacity allocation is viewed as an optimization problem, but under the assumption that servers are protected from overload.

Control the Grid

Allocation techniques in computer Grids must be based on a disciplined approach, rather than ad hoc methods. The four basic mechanisms for implementing resource management policies are:

- Control theory: Control theory uses feedback to guarantee system stability and predict transient behavior, but it can only predict local behavior.
- Machine learning: A major advantage of machine-learning techniques is that they don't need a performance model of the system. You could apply this technique to coordinating several autonomic system managers.
- Utility-based: Utility-based approaches require a performance model and a mechanism to correlate user-level performance with cost.
- Market-oriented/economic mechanisms: Such mechanisms don't require a system model, such as combining auctions for bundles of resources.

A distinction should be made between interactive and non-interactive workloads. The management techniques for interactive workloads (Web services, for example) involve flow control and dynamic application placement, whereas those for non-interactive workloads are focused on scheduling.

A fair amount of work reported in the literature is devoted to resource management of interactive workloads—some to non-interactive and only a few to heterogeneous workloads, a combination of the two. Planning ahead for how you are going to manage these will help ensure a smooth transition to working with the Grid

III. EXISTING APPROACHES

Qiang et al. (2009) using feedback control theory, we present VM-based architecture for adaptive management of virtualized resources in Grid computing and model an adaptive controller that dynamically adjusts multiple virtualized resources utilization to achieve application Service Level Objective (SLO) in Grid computing. Compared with Xen, KVM is chosen as a virtual machine monitor (VMM) to implement the architecture. Evaluation of the proposed controller model showed that the model could allocate resources reasonably in response to the dynamically changing resourcerequirements of different applications which execute on different VMs in the virtual resource pool to achieve applications SLOs.

Younge et al. (2010) presented a new framework is presented that provides efficient green enhancements within a scalable Grid computing architecture. Using power-aware scheduling techniques, variable resource management, live migration, and a minimal virtual machine design, overall system efficiency will be vastly improved in a data center based Grid with minimal performance overhead..

Zhang et al. (2012) present an adaptive power management framework in the Grid to achieve autonomic resource configuration. We propose a software and lightweight approach to accurately estimate the power usage of virtual machines and Grid servers. It explores hypervisor-observable performance metrics to build the power usage model. To configureGrid resources, we consider both the system power usage and the SLA requirements, and leverage learning techniques to achieve autonomic resource allocation and optimal power efficiency. We implement a prototype of the proposed power management system and test it on a Gridtestbed. Experimental results show the high accuracy (over 90%) of our power usage estimation mechanism and our resource configuration approach achieves the lowest energy usage among the compared four approaches

Datta et al. (2012) ease and simplified the web services rendering it user friendly, stretchable, affordable and adaptable with the growing demand and complexity of developing web services and based on the analysis of Human Resource Management and information system requirements for numerous enterprises. A GridHR Management web services would provide a technologically viable solution to the IT world and other enterprises relating to Human Resource Management. A Grid HR Management is an open-source HR Information System that covers Personal Information Management, Employee Self Services, Benefits, Leave and Salary Information Management.

Kaewpuang et al. (2013) propose a framework for resource allocation to the mobile applications, and revenue management and cooperation formation among service providers. For resource allocation to the mobile applications, we formulate and solve optimization models to obtain the optimal number of application instances that can be supported to maximize the revenue of the service providers while meeting there source requirements of the mobile applications. For sharing the revenue generated from the resourcepool (i.e., revenue management) among the cooperative mobile Grid service providers in a coalition, we apply the concepts of core and Shapley value from cooperative game theory as a solution. Based on the revenue shares, the mobile Grid service providers can decide whether to cooperate and share there sources in the resource pool or not. Also, the provider can optimize the decision on the amount ofresources to contribute to the resource pool.

Pengbo et al. (2014) aim to design the network as the integration of the mobile access part and the Grid computing part, utilizing the inherent heterogeneity to meet the diverse quality of service (QoS) requirements of tenants. Furthermore,

we propose a novel cross-network radio and Grid resource management scheme for HMC networks, which is QoS-aware, with the objective of maximizing the tenant revenue while satisfying the QoS requirements. The proposed scheme is formulated as a restless bandits problem, whose ??indexability?? feature guarantees the low complexity with scalable and distributed characteristics. Extensive simulation results are presented to demonstrate the significant performance improvement of the proposed scheme compared to the existing ones.

Zhao et al. (2014) propose a reference service framework for integrating scientific workflow management systems into various Grid platforms, which consists of eight major components, including Grid workflow managementservice, Grid resource manager, etc., and 6 interfaces between them. We also present a reference framework for the implementation of Grid Resource Manager, which is responsible for the provisioning and management of virtual resources in the Grid. We discuss our implementation of the framework by integrating the Swift scientific workflow management system with the OpenNebula and EucalyptusGrid platforms, and demonstrate the capability of the solution using a NASA MODIS image processing workflow and a production deployment on the Science@Guoshi network with support for the Montage image mosaic workflow..

IV. ANT COLO ALGORITHM

In computer science and operations research, the ant colony optimization algorithm (ACO) is a probabilistic technique for solving computational problems which can be reduced to finding good paths through graphs.

This algorithm is a member of the ant colony algorithms family, in swarm intelligence methods, and it constitutes some metaheuristic optimizations. Initially proposed by Marco Dorigo in 1992 in his PhD thesis,[1][2] the first algorithm was aiming to search for an optimal path in a graph, based on the behavior of ants seeking a path between their colony and a source of food. The original idea has since diversified to solve a wider class of numerical problems, and as a result, several problems have emerged, drawing on various aspects of the behavior of ants

Ant colony optimization algorithms have been applied to many combinatorial optimization problems, ranging from quadratic assignment toprotein folding or routing vehicles and a lot of derived methods have been adapted to dynamic problems in real variables, stochastic problems, multi-targets and parallel implementations. It has also been used to produce near-optimal solutions to the travelling salesman problem. They have an advantage over simulated annealing and genetic algorithm approaches of similar problems when the graph may change dynamically; the ant colony algorithm can be run continuously and adapt to changes in real time. This is of interest in network routing and urban transportation systems.

The first ACO algorithm was called the Ant system [8] and it was aimed to solve the travelling salesman problem, in which the goal is to find the shortest round-trip to link a series of cities. The general algorithm is relatively simple and based on a set of ants, each making one of the possible round-trips along the cities. At each stage, the ant chooses to move from one city to another according to some rules:

1) It must visit each city exactly once;
2) A distant city has less chance of being chosen (the visibility);
3) The more intense the pheromone trail laid out on an edge between two cities, the greater the probability that that edge will be chosen;
4) Having completed its journey, the ant deposits more pheromones on all edges it traversed, if the journey is short;
5) After each iteration, trails of pheromones evaporate

With an ACO algorithm, the shortest path in a graph, between two points A and B, is built from a combination of several paths. It is not easy to give a precise definition of what algorithm is or is not an ant colony, because the definition may vary according to the authors and uses. Broadly speaking, ant colony algorithms are regarded as populated metaheuristics with each solution represented by an ant moving in the search space. Ants mark the best solutions and take account of previous markings to optimize their search. They can be seen asprobabilistic multi-agent algorithms using a probability distribution to make the transition between each iteration. In their versions for combinatorial problems, they use an iterative construction of solutions. According to some authors, the thing which distinguishes ACO algorithms from other relatives (such as algorithms to estimate the distribution or particle swarm optimization) is precisely their constructive aspect. In combinatorial problems, it is possible that the best solution eventually be found, even though no ant would prove effective. Thus, in the example of the Travelling salesman problem, it is not necessary that an ant actually travels the shortest route: the shortest route can be built from the strongest segments of the best solutions. However, this definition can be problematic in the case of problems in real variables, where no structure of 'neighbours' exists. The collective behaviour of social insects remains a source of inspiration for researchers. The wide variety of algorithms (for optimization or not) seeking self-organization in biological systems has led to the concept of "swarm intelligence", which is a very general framework in which ant colony algorithms fit.

V. RESULTS& CONCLUSION

In experimental work the execution time and cost incurred by proposed algorithm and existing random resource selection algorithm to execute tasks is compared against varying number of resources and tasks.

TIME_SHARED and SPACE_SHARED resource allocation policies are used to perform the experiments. In all twelve experiments are carried out:

Experiment 1 to Experiment 4 are performed using TIME_SHARED allocation policy with varying number of tasks and resources and execution time and cost is compare.

Experiment 5 to Experiment 8 are performed using SPACE_SHARED allocation policy with varying number of tasks and resource and execution time and cost is compared.

Experiment: 1

The Total Execution Time of Heuristic Resource Scheduling Algorithm (HRSA) is compared with Random Resource Scheduling Algorithm (RRSA) with the following parameters.

Resource Allocation Policy=TIME_SHARED

Number of Resources =25

Number of Tasks = 10 to 50

TABLE 1

Sr. No.	No of Tasks	Execution Time using HRSA(Sec.)	Execution Time using RRSA(Sec.)	Average Improvement %
1	10	5181	8910	71.97
2	20	18182	29597	62.78
3	30	38747	67431	74.02
4	40	66593	116923	75.57
5	50	101353	180192	84.82

Average Improvement in Total Execution Time is = 72.42 %.

Figure 1 shows that as the number of tasks increases the difference between execution time taken by two algorithms increases.

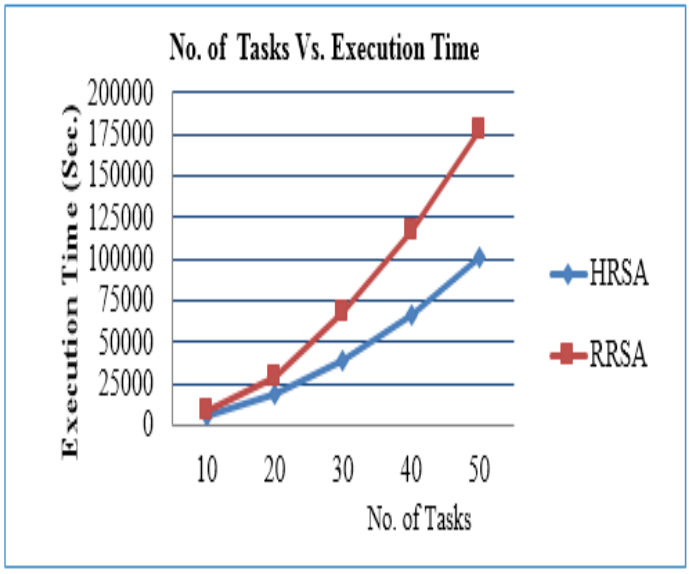

Figure 1 Number of Tasks Vs. Execution Time in TIME_SHARED Allocation.

VI. REFRENCES

[1]. H. El-Rewini, T. G. Lewis, H. H. Ali. Task scheduling in parallel and distributed systems, Prentice-Hall, Inc., Upper Saddle River, NJ, 1994

[2]. D. Gupta, P. Bepari. Load sharing in distributed systems, In Proceedings of the National Workshop on Distributed Computing, January 1999.

[3]. Z. Xiao, W. Song, and Q. Chen, "Dynamic resource allocation using virtual machines for cloud computing environment," IEEE Transactions on Parallel and Distributed Systems, vol. 24, no. 6, pp. 1107–1117, 2013.

[4]. L. D. Dhinesh Babu and P. Venkata Krishna, "Honey bee behavior inspired load balancing of tasks in cloud computing environments," Applied Soft Computing Journal, vol. 13, no. 5, pp. 2292–2303, 2013.

[5]. J. Cao, K. Li, and I. Stojmenovic, "Optimal power allocation and load distribution for multiple heterogeneous multicore server processors across clouds and data centers," IEEE Transactions on Computers, vol. 63, no. 1, pp. 45–58, 2014.

[6]. R. N. Calheiros and R. Buyya, "Meeting deadlines of scientific workflows in public clouds with tasks replication," IEEE Transactions on Parallel and Distributed Systems, vol. 25, no. 7, pp. 1787–1796, 2014.

[7]. R. Basker, V. Rhymend Uthariaraj, and D. Chitra Devi, "An enhanced scheduling in weighted round robin for the cloud infrastructure services," International Journal of Recent Advance in Engineering & Technology, vol. 2, no. 3, pp. 81–86, 2014.

[8]. Z. Yu, . Menng, and H. Chen, "An efficient list scheduling algorithm of dependent task in grid," in Proceedings of the 3rd IEEE International Conference on Computer Science and Information Technology (ICCSIT '10), IEEE, Chengdu, China, July 2010.

[9]. H. M. Fard and H. Deldari, "An economic approach for scheduling dependent tasks in grid computing," in Proceedings of the 11th IEEE International Conference on Computational Science and Engineering (CSEWorkshops '08), pp. 71–76, IEEE, San Paulo, Brazil, July 2008.

[10]. W. Kadri, B. Yagoubi, and M. Meddeber, "Efficient dependent tasks assignment algorithm for grid computing environment," in Proceedings of the 2nd International Symposium on Modelling and Implementation of Complex Systems (MISC '12), Constantine, Algeria, May 2012.

[11]. S. Ijaz, E. U. Munir, W. Anwar, and W. Nasir, "Efficient scheduling strategy for task graphs in heterogeneous computing environment," The International Arab Journal of Information Technology, vol. 10, no. 5, 2013.

[12]. Y. Xu, K. Li, L. He, and T. K. Truong, "A DAG scheduling scheme on heterogeneous computing systems using double molecular structure-based chemical reaction optimization," Journal of Parallel and Distributed Computing, vol. 73, no. 9, pp. 1306–1322, 2013.

[13]. L.-T. Lee, C.-W. Chen, H.-Y. Chang, C.-C. Tang, and K.-C. Pan, "A non-critical path earliest-finish algorithm for interdependent tasks in heterogeneous computing environments," in Proceedings of the 11th IEEE International Conference on High Performance Computing and Communications (HPCC '09), pp. 603–608, Seoul, Republic of Korea, June 2009.

[14]. B. Xu, C. Zhao, E. Hu, and B. Hu, "Job scheduling algorithm based on Berger model in cloud environment," Advances in Engineering Software, vol. 42, no. 7, pp. 419–425, 2011.

[15]. B. Mondal, K. Dasgupta, and P. Dutta, "Load balancing in cloud computing using stochastic hill climbing-a soft computing approach," Procedia Technology, vol. 4, pp. 783–789, 2012.

[16]. M. Rahman, R. Hassan, R. Ranjan, and R. Buyya, "Adaptive workflow scheduling for dynamic grid and cloud computing environment," Concurrency and Computation: Practice and Experience, vol. 25, no. 13, pp. 1816–1842, 2013.

[17]. G. Gharooni-fard, F. Moein-darbari, H. Deldari, and A. Morvaridi, "Scheduling of scientific workflows using a chaosgenetic algorithm," Procedia Computer Science, vol. 1, no. 1, pp. 1445–1454, 2010, International Conference on Computational Science, ICCS 2010.

[18]. C. Lin and S. Lu, "Scheduling scientific workflows elastically for cloud computing," in Proceedings of the IEEE 4th International Conference on Cloud Computing, Washington, DC, USA, July 2015.

[19]. Vijindra and S. Shenai, "Survey on scheduling issues in cloud computing," Procedia Engineering, vol. 38, pp. 2881–2888, 2016, Proceedings of the International Conference on Modelling Optimization and Computing.

[20]. M. Xu, L. Cui, H. Wang, and Y. Bi, "A multiple QoS constrained scheduling strategy of multiple workflows for cloud computing," in Proceedings of the IEEE International Symposium on Parallel and Distributed Processing with Applications (ISPA '09), pp. 629–634, IEEE, Chengdu, China, August 2015.

[21]. C. Lin, S. Lu, X. Fei et al., "A reference architecture for scientific workflow management systems and the VIEW SOA solution," IEEE Transactions on Services Computing, vol. 2, no. 1, pp. 79– 92, 2009.

[22]. S. Ghanbari and M. Othman, "A priority based job scheduling algorithm in cloud computing," in Proceedings of the International Conference on Advances Science and Contemporary Engineering, pp. 778–785, October 2012.

