# A SOA-based Resource Intensive and Data Aware (RIADA) Approach for Grid Computing

Humera Bashir
Lecturer: Department of Computer Science, University of Karachi.

Sadiq Ali Khan
Assistant Professor: Department of Computer Science, University of Karachi

Shaista Rais
Lecturer: Department of Computer Science, University of Karachi

## ABSTRACT

The Grid technology is flowing into large scale service oriented architecture-- a universal podium for delivering future high demand computational services. The management of resources and requests scheduling in this big range distributed environment is a complicated job, no contemplation may result in efficiency deprivation in a Grid environment and may possibly bring about big handling queues and task running delays. This paper outlines a simple and straight forward approach to incrementally maintain the area of Grid technology addressing challenges related to the problem of maintaining a Grid wide view of Grid user's resource utilization. To remain flexible this paper presents a SOA- Based RIADA (Resource Intensive and Data Aware) approach for providing a basis for more efficient and user friendlier management of resources and resource scheduling techniques in a future Grid offering a rich blend of diverse applications.

## General Terms

Computation, Communication, Distributed System, Cloud Computing, Computer Architecture, Data Management, Systems Design, Databases, Schedulers

## Keywords

Grid technology, Data Intensive Scheduling Techniques, Resources Management

## 1. INTRODUCTION

A Grid is definitely an anthology of devices sometimes referred to "nodes, users, client, resources, hosts" engines along with other such terms. They all lead any amalgamation of resources to the Grid as a complete. Many different methods may be used by all consumers of the Grid while the others may have certain boundaries.

Grid is huge degree, decentralized and heterogeneous NC systems that will scale to web-size milieu with devices spread i.e. distributed across numerous organizations and administrative domains. A defining feature of Grids is the sharing of networks, computers, and other resources and services. User requirements for performance must be translated into resource requirements and conflicting resource requirements must be resolved. Grid computing adapted from [13] is much concerned with "matched resource sharing discussing and issue resolving in active, multi-institutional electronic organizations." Grid doesn't enforce utter control over these resources and management of resources is at the mercy of numerous and deviating organizational management strategies. From consumer's notion, a Grid is really a collaborative problem-solving milieu, wherever one or several personal jobs may be presented without understanding where in fact the resources are as well as who possess the resources. A Grid should assure the caliber of support of job's execution. The consumers' resource necessities in the Grids differ depending on their objects, time constraints, priorities and budgets. Assigning their projects to suitable resources in the Grids to ensure that efficiency demands are pleased and expenses are topic for their finances is definitely a very complex problem.

On the way to construct a Grid, the progress and exploitation of several services is required. They accept low-level support such as protection, data, listing, management and scheduling (resource trading, reference provision, sighting of resources, entry charge concession, resource collection, methods of scheduling, QoS (Quality of Services), and running performance management), and higher level services/tools for software development. Two challenging facets of Grid processing are management of resources and scheduling. The surfacing of a number of new application necessities that help Grids for successful information, data and resource administration mechanisms. Planning a Grid architecture that will match these demands is complicated because of numerous machine issues. The difficulty of the allocation issue and the dynamically adjusting efficiency features of Grid assets (due to competition and failure of resources) are in a way that intelligent schedulers for applications are essential to assign and re-assign resource. With allocation decisions below regional get a handle on, the possible of instability exists as competitive software schedulers continually conform to fluctuations of load they themselves induce. A Grid fundamentally contains two different areas i.e. compute and information [11].

This paper contemplates parallel tasks that consist of separate, not dependent, and similar nature of jobs but examine these tasks for numerous jobs which are somewhat of small size i.e. much like the amount of accessible resources. This paper is targeted on optimizing the efficiency of numerous, competitive tasks or jobs. But, the SOA-based RIADA approach heuristics for Grid offer important components for planning powerful work management methods (e.g. for performing suitable space-sharing among jobs, choosing which assets are useful for which work, deciding job duplication degrees for every single job). The SOA-based RIADA approach heuristics provide main elements for scheduling may then produce knowledgeable evaluation by enchanting into consideration, the varying state of the system, locality and measurement of the data and the available pool of cycles utilized for processing. The designs mentioned in this SOA-based RIADA can be used right as a cause for creating, studying, and analyzing new algorithms of resource management. But, it is comparatively difficult to acquire analytic benefits regarding the usefulness of various calculations of management and have to count on seen evaluations and comparisons.

## 2. PROCESSING AND DATA GRID

***Compute Grid:*** Administrating services for Grid computing, discussing, administration (task of services management and providing i.e. core resources) circulation of responsibilities centered on configurable service-level policies.

***Data Grid***: offers the information and data administration function allow knowledge entry, synchronization, and circulation of a Grid.

Each has their beginnings in academia and study wherever complicated diagnostic strategies running around big knowledge models were getting the norm. The surfacing of computational Grids as recent high-performance research infrastructures provide the consumers use of research methods at an unprecedented range in the real history of computing. But, computational Grids change from past research infrastructures simply because they display equally similar and spread elements: a computational Grid is some numerous and commonly spread research methods [12]. The target is to produce the impression of an easy however big and strong self-managing electronic-virtual PC out of a sizable assortment of related heterogeneous programs discussing numerous mixtures of resources. The emerging standardization for discussing methods, combined with the option of larger bandwidth are operating a probably similarly big transformative part of Grid computing. Because it migrates from the arms of beauty to the sphere of executive via the applying of tried-and-true executive principles-computing becomes a basic power in exactly the same way that fuel and energy era and distribution is just a utility. The grade of the service will soon be calculated by their capacity to generally meet the supply-and-demand shapes of the suppliers and consumers. The suppliers (also named resource owners) and consumers (who would be the consumers) have various targets, purposes, methods, and need designs [5] As gird programs are stationed, many different executive techniques to Grid allocation of resources has been and can remain applied.

## 3. RESOURCE MANAGEMENT

Challenging for the successful usage of Grids for compute-intensive jobs is that of management of resources due to insufficient management and job scheduling methods that account fully for volatility of resource, the conventional utilization of Grids has dedicated to high-throughput programs that include vast quantities (i.e. purchases of magnitude bigger than the amount of accessible resources) of distinct tasks. Management of Resources [4, 5] is a main job in virtually any Grid system. Resource may possibly contain “conventional” resources such as for instance cycles of computations, system bandwidth, and storage methods, Common resource administration and managing methods which were proposed are Globus GRAM [15], WMS [16], from EDG [17], Grid Way [9], SGE (Sun Grid Engine) [10], Condor [18] and the Euro Grid-Unicore [19] reference broker projects.

For a Grid to successfully support many different purposes, the resource managing (RMS) that is main to its function should handle the problems and issues under along with dilemmas such as for example problem threshold and security [2]. Dilemmas are (a) encouraging versatility, extensibility, and scalability; (b) letting techniques with various administrative procedures to inter-operate while keeping website autonomy; (c) co-allocating assets; (d) encouraging eminence of services; and (e) conference computational price limitation. Applying these styles, Grid techniques may be position to the classes as revealed in Figure-1 [1].

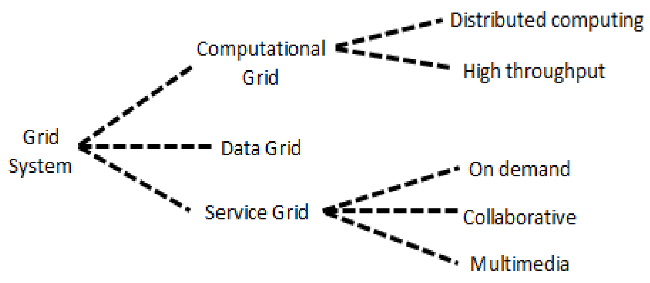

**Figure-1. A nomenclature of Grid Systems**

As a result of problems such as for instance extensibility, flexibility, autonomy of site, QoS, co-allocation, management and administration of resources in Grid systems is extra difficult than in conventional distributed milieu for Computing (DCEs). In general a Grid application may have a number of various parts mapped onto different resources. Such circumstances, the RMS should work with a worldwide abstraction of the application to accredit and allocate node i.e. resources [3].

Whenever a work is given into a Grid scheduling system the phrases “scheduling” is not to be puzzled with “reservation” of methods in advance to enhance the quality of service. Occasionally the word “resource broker” can be used instead of “scheduler,” but term resource broker implies that some kind of bartering ability is factored into scheduling. The scheduling system has got the task to pick an appropriate source and then to manage the task execution. Your choice which source should be utilized is the outcome of a dating method between give in requested and accessible resources. But, in this dating method, we truly need some flexible scheduling systems and techniques with appropriate heuristics that may take into account the characteristics of the network to enable effective scheduling of source extensive jobs to practical research resources. A “meta scheduler” can help the seeking of methods across multiple machines for jobs and can accomplish fill managing of workloads across multiple sites. Each site even offers a unique regional scheduler to ascertain how their work queue is processed. A meta-scheduling system performs on the foundation that the “new task” which must be accomplished has to make itself proven to a “matchmaker.” That matchmaker works as a gate way to the Grid. It selects methods from a worldwide directory.

As interaction technologies, storage and computational technologies gradually increase, significantly huge, complicated, and resource-intensive jobs are increasingly being produced both in study institutions and in industry. It is a popular surveillance that computational resources are deteriorating to meet the demand of these applications. The standard computational and research Grid is rolling out service driven research architecture with a brilliant regional source administration and scheduling strategy. In Globus Tookit4 [20] (GT4, the official implementation of the existing Grid standards), eight high-level Grid services described by Open Grid Company Architecture (OGSA) [21] are implemented using Web Services process to provide functionalities such as for example source administration, scheduling, etc. As these series are expected be stateful and services regarding internet are usually stateless, Services For Web Reference Platform (WSRF) [22] was introduce so the stateful data can be preserved as WS-Resources between various support invocations. Stateful resources do not only contain old-fashioned Grid resources such as for example equipment, but any such thing which includes state and needs to be restored across multiple support interactions. These included entities such as for example sessions, class account, recommendations, and subscriptions to topics. While WS-RF efforts to reconcile the referencing of stateful resources with stateless support relationships, it however keeps a combining between support endpoint addresses and source identifies. This may minimize its capacity to take care of conditions concerning nodes at the edges that will not have straight available addresses that is considered as static.

## 4. IMPORT OF RESOURCE CONSUMPTION

Usually, people calculate utilization of resource by the total amount of computational facet of the corresponding form of resources used. Like usually use of CPU is assessed by the number of time slots used. At present, users learned this measuring convention in computational Grids. The price options derive from computational units of source utilization in the economic-model source allocation strategies for a computational Grid. Based with this cost hurt system, in order to assign their jobs to suitable source manufacturers, consumers must be sensitive of the quantity and forms of resource components they meet for a specific task to estimate the fee and predicate the execution. Nevertheless, not like most conventional computational podiums, also exactly the identical types of resources in computational Grids range over a broad variety of capabilities. The jobs in computational Grids typically require extensive sources that have different capabilities. An average example is a task that involves a big degree difference in clock charges of CPUs. It is maybe not appropriate to gauge the CPU utilization just based on the number of time slots expected, because the computational capabilities of time slots of different CPU units should not merely reveal enough time slots but additionally the differentiation of bodily capabilities of the CPUs. However, it becomes excessively complex for users to gauge whether they should outsource a resource extensive task if different CPU units have different prices. Meanwhile, the available source in a Grid could change at different times. Ergo, also identical task must certainly be evaluated at every time when they'll be executed. Also, different users have different efficiency conditions and requirements. They want to find the least expensive sources that could satisfy their efficiency requirements. It is also complicated to utilize the price of computational source units to represent users different preferences. Price of source usages must certainly be produced to hide the differentiation of bodily capabilities of exactly the same form of sources, also reflecting different consumer tastes in a Grid.

For the sake of simplicity, this paper use CPUs (processors) at 5 distinct sites because the example resource to explain how to product the capabilities of a resource by taking into consideration the specific efficiency it may achieve based on specific tasks instead of utilizing the utilization of resource units. To start, consider an easy observation showing the basic idea behind the job-oriented system of measuring the value of resource utilization in this paper.

Assume you can find three processors P1, P2, and P3 which may have various rates of speed (Here, we don't establish what correct indicating of the rate of a CPU. Maybe it's calculated by MIPs, time charges, clock or some other sort of common units) from the greatest to the cheapest respectively.

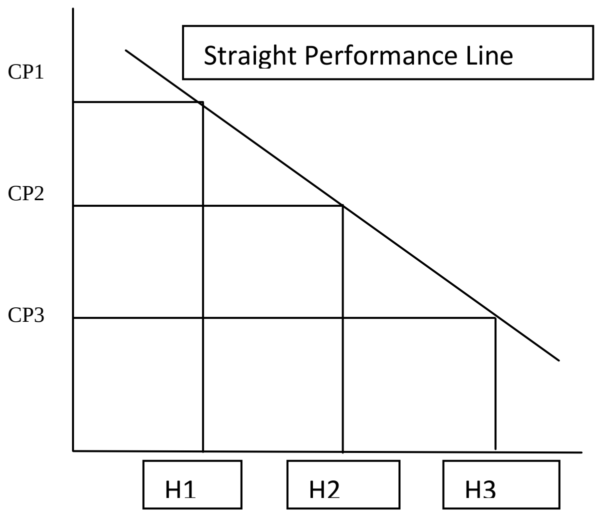

**Figure 2: Similar performance by diverse Processors**

Provided the identical work, these three processors could end it in various timeframe (Here a presumption is manufactured that most different situations are same, e.g., same level of RAM associating with each processor). Figure 2 reveals the performance and efficiency of every processor. The same efficiency range shows the fact these three models P1, P2, and P3 end the identical work within H1, H2, and H3 hours of CPU respectively. The total amount of function they did is same; nevertheless they applied various quantity of time. If a resource user provides exactly the same work to P1, P2, and P3 , how if the model manager cost an individual for applying various processors? If all processors may meet the timeline of work, the resource consumer would rather perhaps not to cover additional for employing P1. However, if the timeline of the task is limited or tight, he might be ready to cover more for employing P1. Thus, to be able to collection appropriate rates provide applying P1, P2, and P3 to implement the identical work, the resource manufacturers require to take into account equally the various features of the processors and the consumers' performance.

## 5. METHODOLOGY OF SURVEILLANCE

As opposed to monitoring task incognito, our in depth analysis investigated the time that the resource spent allocated to the activity during the course of the simulation, access frequencies which is the major data requirement related to applications, the attributes (resources) accessed by job with their combinations and their access frequency in addition to the size of computation, on a consumer-by-consumer source. That analysis revealed numerous enormously tempting results. Let Q = q1, q2, ……..,$q_q$ function as group of consumer queries (applications ). Next every query $q_i$ and each feature Aj, relate an attribute usage value, represented as use (qi, Aj), and defined as follows:

$$\text{Use}(q\mathbf{i}, Aj) = \begin{cases} 1 \quad 0 & \text{; if Attribute } Aj \text{ is referenced by } q\mathbf{i} \\ 1 & \end{cases}$$

**eq (1)**

**q**1: access A1, A2, A5, A6, and A7, given cond1.

q2: access A1, A2, A4, and A6of while relation R1

“

“

“

q**40**:access **A1, A2, A3, A4 and A7for each A4 = value**

Relating to the considered 40 jobs , the feature utilization prices may be defined. These utilization values establish in matrix eq1, wherever (i, j) indicates Use (qi, Aj), total calculations aren't revealed here.

$$\begin{array}{c|cccccccc} & A1 & A2 & A3 & A4 & A5 & A6 & A7 & A8 \\ \hline q_1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 \\ q_2 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ .. & .. & .. & .. & .. & .. & .. & .. & .. \\ .. & .. & .. & .. & .. & .. & .. & .. & .. \\ .. & .. & .. & .. & .. & .. & .. & .. & .. \\ q_{40} & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 0 \end{array}$$

Attribute Usage Matrix

Utilization of Attributes values is not adequately standard to create the cornerstone of attribute or feature breaking and fragmentation, because the values do not signify the weight of application frequencies. This paper includes the frequency measure along with the feature or attribute affinity measure *aff*(Ai, Aj), measuring the link between two attributes of a connection in accordance with how they are accessed by queries (applications) , which can be described w.r.t. the set of applications Q= {$q_1$,$q_2$…….$q_q$}.

$$\mathbf{aff}(\mathbf{A_i}, \mathbf{A_j}) = \sum_{\mathbf{K \mid use\ (q_k, A_i) = 1\ \wedge\ use(q_k, A_j)}} \sum \mathbf{ref_l\ (q_k) acc_l\ (\ q_k)}$$

Wherever $ref_1$(qk) is the amount of accesses to feature or attributes (Ai, Aj) for every running of request $q_e$ site $S_1$ and $acc_1$ ($q_e$) is the application form entry volume evaluate, for ease, think that $ref_1$(qk)=1 for many $q_e$ and $S_1$.If the application form wavelengths are:

$$\begin{array}{llll} acc_1\ (q_1) = 360 & acc_2\ (q_1) = 0 & ….. & acc_8\ (q1) = 0 \\ acc_1\ (q2) = 546 & acc_2\ (q_2) = 0 & ….. & acc_8\ (q2) = 935 \\ .. & .. & .. & .. \\ .. & .. & .. & .. \\ acc_1\ (q_{40}) = 206 & acc_2\ (q_{40}) = 0 & ….. & acc_8\ (q_{40}) = 0 \end{array}$$

We tested and evaluated our heuristics in simulation on various Grid platforms, but, here discussion centers around the platform on which remarkable results has been found, namely the GridSim platform. Figure 3 illustrate the occasions when the most functioning ten consumers submitted tasks to the Grid during phase that's been analyzed in depth. A bar against an individual and a date suggests that the user has submitted jobs during that day. The main surveillance to be produced from that figure3 is that, although they were the most functioning consumers, all of them didn't submit jobs all the time. Moreover, you will find very obvious patterns of task submission. For example User (u9) and User (u8) utilize the Grid for extended periods of time. User (u2) and User (u4) utilize the Grid for 1-2 days of concentrated activity followed by a period of inactivity that's at the very least equally long. By using worksheet software, many different statistical analysis that verifies our heuristics has been performed.

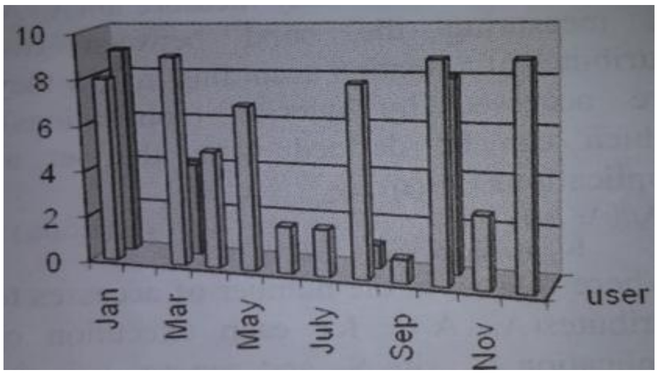

**Figure 3: User Movement in Analysis**

Figure 4 shows the particular circulation task arrivals within a arbitrarily chosen about three day phase. The particular figure 4 exhibits the three details collection, a consumer for every day, the place every ray exhibits the sheer numbers of tasks which were submitted in a 6 minute period. Exactly what is so visible via figure 2 is for every day job submissions don't arise using a common circulation nevertheless people distribute more and more jobs throughout a single go as well as then you'll find extended phases when virtually no tasks usually are give in. The actual cause of this particular tendencies is users infrequently distribute jobs personally nevertheless, alternatively, automatic systems submitter making use of computational work-flow tools, like the OMII-BPEL ecosystem [23] condor's DAGMan[24] or just using layer website programs

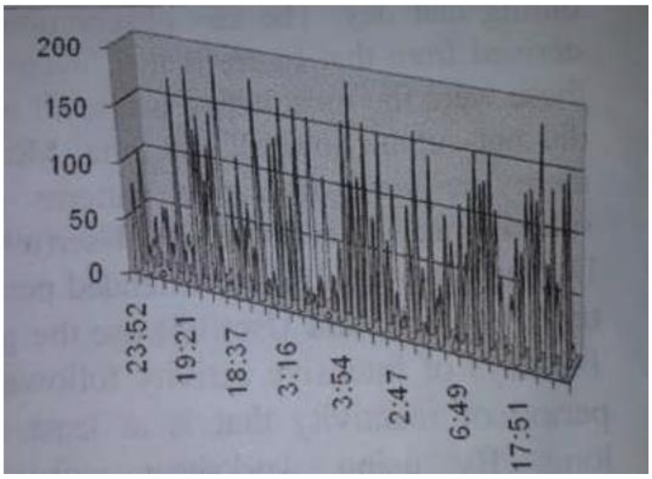

**Figure 4: Work Submission Tarif during three arbitrarily elected days**

On the whole, fascinating surveillance revealed in Figure 4. The figure 4 demonstrates the allocation of task length for the four most functioning consumers. It demonstrates a large proportion of tasks that user (9) give in are less than an hour or so, at the same time as u7 regularly give in long executing tasks, the size of which regularly surpasses 10 hours. User5 give in jobs by having an middling length of approximately 7 hours and his task distributions is that consumers utilize identical pair of computational applications over comprehensive interlude of time, contained by or across sessions, and pertain them to distinct studies and data sets. The dialogue of the observed illustrations above implies that the behaviors of consumers who give in tasks to a Grid is not at all random, but, rather, follows conventional patterns. It's this observation that can be utilized and exploit desire to boost the class of scheduling decisions that meta-schedulers, portals or Grid federations have to build. The fundamental thought is that, because job submission take place in distinctive patterns and the periods of jobs can differ between minutes and days, conventional prediction techniques may give up improved scheduling decisions than the standard technique of submitting we use this insight to derive the two major hypotheses of this paper. The analysis is not confined to extracting the statistical characteristics of these resources, but delves also into expressing techniques of exploiting theses resources in performing. The second hypothesis of this paper is that the quality of Grid scheduling can be improved with techniques that use SOA mechanism.

Security problems come up when services tend to be concatenated. Objects, called processes, of a concatenate service application may run for prolonged length of time and also has to be vigorous in the face of remote-service failures.

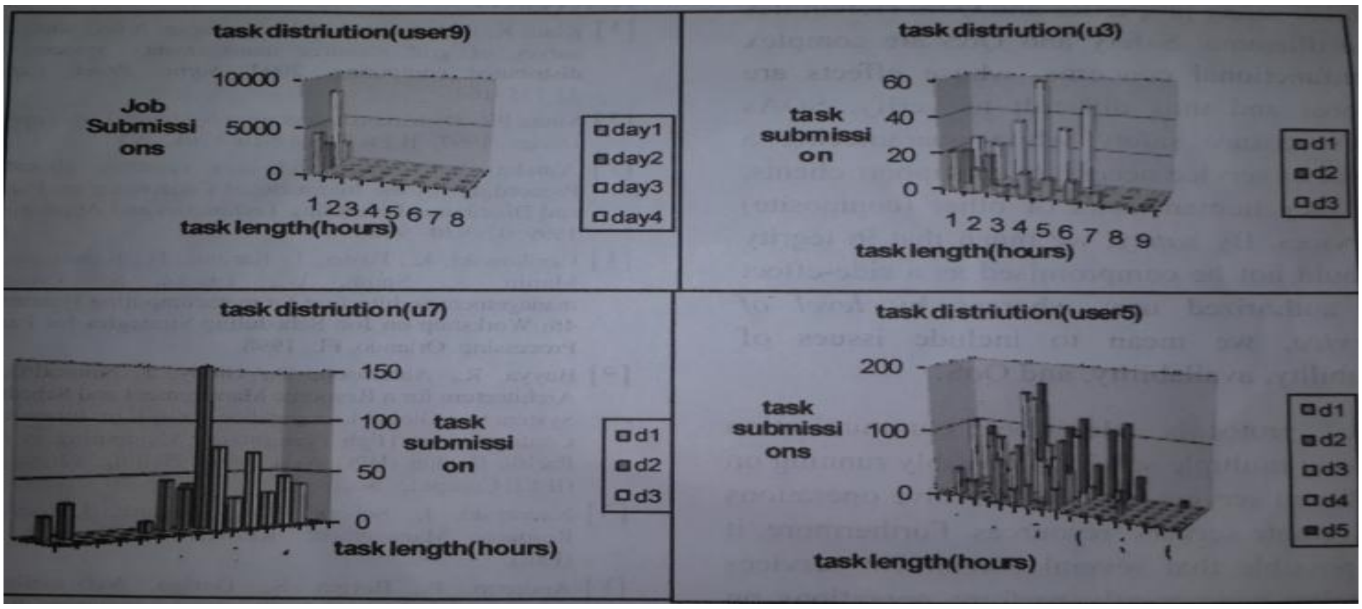

**Figure 5: Task Length by Users**

Long-running processes, such as user7 constantly submits prolonged running jobs, the size of which oftenly exceeds 10 hours seeing that shown in figure 4 may possibly degrade performance of server perhaps to the level associated with inducing the server so that server deny services to the consumers. To control these problems, the developer must normally enhance or else standard simple interaction protocol with logic which e.g. monitors any time used up anticipating a solution in addition to which causes some kind of remediation when an increased period of time has elapses. Faults in theses protocols can result in wellbeing violations in addition to may very well be exploited by attackers.

In order to build secure SOAs require the skill to layout, and package services as mechanism to state compositions and motive concerning world-wide well being properties of safety in addition to QoS. Here is situated the trouble; Safety in addition to QoS will be elaborate nonfunctional complex problems, whose effects are world-wide therefore complicated and tricky to verify. SOAs must make certain safety through guaranteeing any qualifying measures of services and information necessary to sustain clients, be that they human consumers and also other (composite) services. By safety, means, integrity shouldn't be give and take as a side-effect of authorized use; whereas by amount of service, means, to add issues of instability availability and QoS.

The protocols illustrate communication and interaction between numerous services perhaps running and managing on several distinct servers, and could include expeditions on remote server resources. Additionally, it is also possible that many major services running and managing in parallel execute operations on several dissimilar resources seeing as part of a particular transaction. To appropriately put into practice a distributed transaction from corner to corner manifold constituent services needs intricate rules of message exchange among the constituents. If the constituent series are urbanized in an ad hoc manner, delicate implementation blemish could end in security troubles and degraded ranks of service.

## 6. CONCLUSION AND FUTURE WORK

However, our RIADA heuristics present important components for modeling efficient resource management policies (for doing suitable space-sharing among tasks, techniques for choosing which resources are utilized for which task , for determining job duplication levels for task) and in future one can enhance the caliber of Grid scheduling by embedding techniques that utilize SOA (Service Oriented Architecture) mechanisms.

## 7. REFERENCES

[1] Klaus K, R buyya and M Maheswaran. A taxonomy and survey of Grid resource management systems for distributed computing 2002. Softw. Pract. Exper., 32:135-164.

[2] Sinha PK. Distributed Operating Systems: Concepts and Design. 1997. IEEE Press: NewYork, NY.

[3] Vahdat A. Toward wide-area resource allocation Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications. 1999. 02:930-936.

[4] Czakowski, K., Foster, I., Karonis, N., Kesselman, C., Martin, S., Smith, W., Tuecke, S. A Ressource management architecture for metacomputing systems. In 4th Workshop on job scheduling strategies for Parallel Processing. Orlando, FL, 1998.

[5] Buyya, R., Abramson, D., Giddy, J. Nimrod/G. An Architecture for a Resource Management and Scheduling System in a Global Computational Grid. In: International Conference on High Performance Computing in Asia—Pacific Region (HPC Asia 2000). Beijing , China, 2000(IEEE Computer Society)

[6] Nabrzyski, J., Schopf, J.M., Weglarz., J. (eds). Grid Resource Management. Kluwer, Boston, MA 2003(Fall)

[7] Andretto, P., Borgia, S., Dorigo, A., Gianelle, A., Mordacchini, M., et al. Practical approaches to Grid workload & resource management in the EGEE project. In. CHEP 2004, Interlaken, Switzerland, 2005.

[8] http://www.glite.org/, May 2006. European Data Grid Project http://eu-dataGrid.web.cern.ch/eu-dataGrid/

[9] Huedo, E., Montero, R.S., Liorente, I. M. A Framework for adaptive execution on Grids. Softw. Prac. Exp. 2004.34, 631-651.

[10] Dss Sun Grid Engine , http://www.sun.com/software/Gridware/

[11] Manikandan. T., Thamizharasi, M., Chitra, R.Distributed Heterogeneous Data Management in Grid Computing .

[12] Alexandre, D., Christian, P., Thierry, P. Network Communications in Grid Computing: At a Crossroads Between Parallel and Distributed Worlds.

[13] Ian, F., Carl, K., Steven, T. The Anatomy of the Grid Enabling Scalable Virtual Organizations.

[14] Chervenak, A., Ian, F., Carl. K., Salisbury, C.,Tuecke, S. The data Grid: Towards an architecture for the distributed management and analysis of large scientific datasets.

[15] J. Nabrzyski, J.M. Schopf, J. Weglarz (Eds). Grid Resource Management. Kluwer Published, Fall 2003.

[16] Andretto, P.,Borgia, S., Dorigo, A., Gianelle, A., Mordacchini ,M., et al. Practical Approaches to Grid Workload & Resource Management in the EGEE Project, CHEP04, Interlaken, Switzerland.

[17] European Data Grid Project: http://eu-dataGrid.web.cern.ch/eu-dataGrid/

[18] Basney, J., Livny, L., Mazzanti, P., Utilizing Widely Distributed Computational Resources Efficiently with Execution Domains. Computer Physics Communications, 2001.

[19] Brooke, J.,Fellows, D., MacLaren, J. Resource Brokering: The EUROGRID/GRIP Approach, UK e-Science All Hands Meeting, Nottingham, UK, 31 Aug. - 3 Sep. 2004

[20] Globus Alliance (2005), Globus Toolkit 4.0 (GT4). http://www-unix.globus.org/toolkit/docs/4.0/GT4Facts/.

[21] Foster, I., Kesselman, C., Nick, J. M. &Tuecke, S. (2002), The Physiology of the Grid: An Open Grid Service Archetecture for Distributed Systems Integration. http://www.globus.org/research/papers/ogsa.pdf.

[22] Foster, I., Czajkowski, K., Ferguson, D., Frey, J., Graham, S., Maguire, T., Snelling, D. and Tuecke, S. (2005), 'Modeling and managing state in distributed systems: the role of ogsi and wsrf', Proceedings of the IEEE 93(3), 604–612.

[23] Emmerich, W., Butchart, B., Chen, L., Wassermann, B. and Price, S. (2005). Grid Service Orchestration using the Business Process Execution Language (BPEL). Journal of Grid Computing, 3(3-4):283-304.

[24] Frey, J., et al., "Condor-G: A Computation Management Agent for Multi-Institutional Grids," Cluster Computing, vol. 5, 2002, pp. 237-246.