A SLA-Aware Scheduling Architecture in Grid System Using Learning Techniques

Seyedeh Yasaman Rashida[1], Amir Masoud Rahmani[2]
[1]Department of Computer Engineering, Shirgah Branch, Islamic Azad University
Shirgah, Mazandaran, Iran

[2]Department of Computer Engineering, Islamic Azad University, Science and Research Branch
Tehran, Tehran, Iran

Abstract
In the Grid environment, the relationship between a customer and a service provider should be clearly defined. The responsibility of each partner can be stated in the so-called Service Level Agreement (SLA). A SLA is a formal contract between end-user and system to guarantee that customers' service quality expectation can be achieved. In recent years, extensive research has been conducted in the area of SLA for utilizing computing systems and also, various SLA-based scheduling are proposed but the number of resources and tasks to be scheduled is usually variable and dynamic in nature. Most of proposed algorithms don't have flexibility in all situations, because every scheduling algorithm cannot improve all grid factors like resource utilization, load balancing, etc and cannot notice all parameters at the moment. In this paper, we propose SLA aware scheduling architecture which uses learning techniques for selecting best way to schedule resources in different situations. The proposed model causes increasing user satisfaction, number of completed tasks and system utilization and resource load balancing. At the end, we formulize relation between number of completed tasks and system utilization.

Keywords: *SLA, Scheduling, Grid Computing, Learning Technique, Load Balancing.*

1. Introduction

Grid computing system is a collection of distributed heterogeneous computing resources available over a local or wide area network that appears to an end user or application as one large virtual computing system. Grid computing is to provide an unlimited power, collaboration, and information access to everyone connected to grid [1].

A schedule is defined as a function $f: T \longrightarrow R$ which maps every task $T_i \in T$ on a resource $R_j \in R$ that has attached to a queue Q_j. The goal of any schedule is to minimize the cost function such as scalability and lateness.

The grid scheduler has four phases, which consists of resource discovery, resource selection, job selection and job execution. A grid scheduler acts as an interface between the user and distributed resources. It hides the complexity of the computational grid from the grid user. The main responsibility of a scheduler is selecting resources and scheduling tasks in such a way that the user and application constraints are satisfied, in terms of overall execution time and cost of the resources utilized. To achieve these goals, Service Level Agreement (SLA) can play a critical role. In general, SLAs are defined as an explicit statement of expectations and obligations in a business relationship between service providers and customers. SLAs specify priori negotiated resource requirements, the quality of service (QoS), and costs.

Most research applies one or two scheduling algorithms to achieve their goals such as maximize number of completed jobs, system utilization, etc. But it is important to notice that each scheduling algorithm can improve some of the expected factors. In order to dynamic grid environment, it is possible to confront critical situation which applied proposed scheduling cannot achieve the whole goal. In this paper, we propose a SLA-aware scheduling scheme which uses learning techniques to select best way of scheduling for achieving system goals in variable situations.

The rest of this paper is organized as following. In section 2, we discuss related work. Section 3 describes the proposed scheduling architecture. In section 4, we obtain relationship between number of completed jobs and utilization. Section 5 gives the concluding remarks.

2. Related Work

Distributed resource allocation is one of the most challenging problems in resource management field. This problem has attracted a lot of attention from the research community in the last few years. In the following we provide a review of some relevant prior work.

Bin Zeng et al. [5] propose a negotiation based model, where adaptive learning agents, representing individual resources and tasks, co-operate among themselves to help achieving a near optimal schedule. N.Malarvizhi and V.Rhymend Uthariaraj [6] describe a scalable grid-architecture involving a Grid Resource Manager, assuming the role of a resource broker to select computational resources based on job requirements and the capacity of grid resources, so as to minimize the time to process each application along with transmission time associated with it. D. P. Spooner et al. [7] develop a multi-tiered scheduling architecture (TITAN) that uses a performance prediction system (PACE), along with brokers that are involved in distribution of jobs in the grid, to meet deadlines and significantly increase the efficiency of resource utilization.

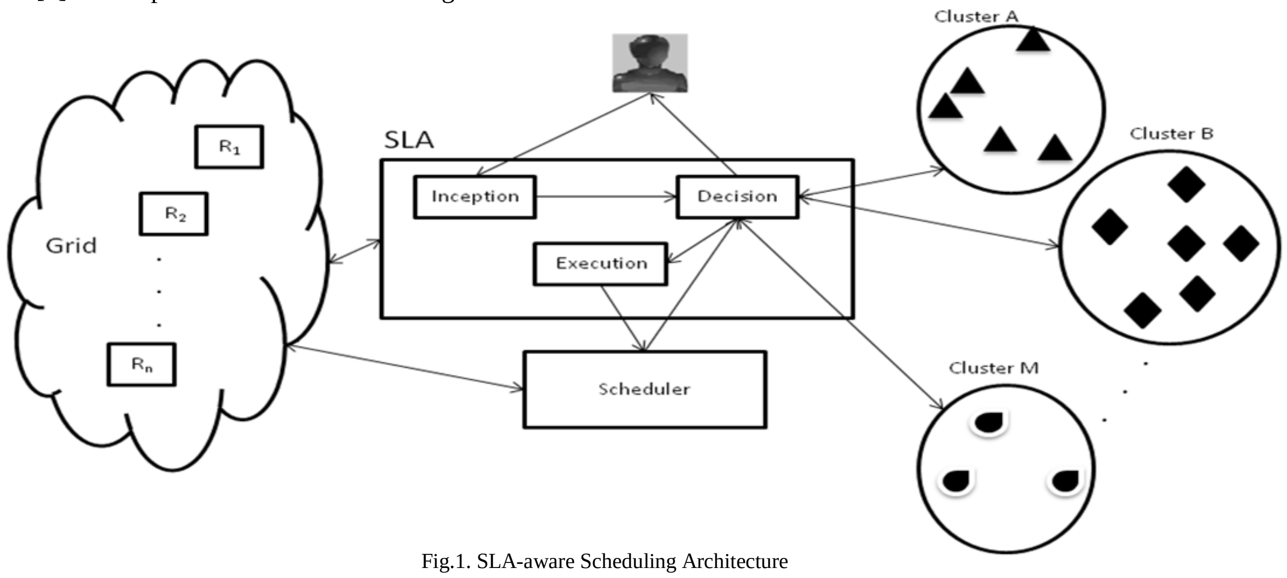

Fig.1. SLA-aware Scheduling Architecture

The paper [8] presents a novel load balancing approach in a heterogeneous distributed environment. The scheduler takes into account the threshold value, based on the ratio of service rates, along with the queue length to determine whether it is beneficial to migrate a given local task to another node in the system or not. Markov process model is used to describe the behavior of the heterogeneous distributed system under the proposed policies. Kumar also proposes a Load balancing algorithm for fair scheduling, and compares it to other scheduling schemes such as the Earliest Deadline First, Simple Fair Task order, Adjusted Fair Task Order and Max Min Fair Scheduling for a computational grid. It addresses the fairness issues by using mean waiting time. It scheduled the task by using fair completion time and rescheduled by using mean waiting time of each task to obtain load balance. This algorithm scheme tries to provide optimal solution so that it reduces the execution time and expected price for the execution of all the jobs in the grid system is minimized [27]. In [30], Anandharajan and Bhagyaveni propose to find the best EFFICIENT cloud resource by Co-operative Power aware Scheduled Load Balancing solution to the Cloud load balancing problem. The algorithm developed combines the inherent efficiency of the centralized approach, energy efficient and the fault-tolerant nature of the distributed environment like Cloud. Shahu Chatrapati et al. [28] propose *Competitive Equilibrium Scheme* (CES) that simultaneously minimizes mean response time of all jobs, and the response time of each job individually.
Ruay-Shiung Chang et al. [9] propose an Adaptive Scoring Job Scheduling algorithm (ASJS) for a distributed grid environment to reduce the completion time of submitted jobs, by assigning jobs to resources after looking into recent scheduling history of every available resource and then choosing the most optimal one.
Computing intensive jobs and data intensive jobs handled differently, and local and global updates are used to obtain the most recent status of grid resources to schedule jobs more effectively in real time. System ModelSyed Nasir Mehmood Shah et al. [10] propose an algorithm for CPU scheduling of a modern multiprogramming operating system, design and development of new CPU scheduling algorithms (the Hybrid Scheduling Algorithm and the Dual Queue Scheduling Algorithm) with a view to minimize overall task schedule. The following paper extends this prioritized round robin heuristic from a single system multiprogramming environment, onto a multi-processor distributed architecture. As each scheduling strategy optimizes some of performance parameters such as making span, resource utilization, response time, workload balancing, service time, reliability, fairness deviation and throughput**,** we propose a SLA-aware scheduling model to achieve four important parameters such as resource utilization, response time, workload balancing and throughput.
In [26], Murugesan and Chellappan introduce a new resource allocation model with multiple load originating processors as an economic model. Solutions for an optimal allocation of fraction of loads to nodes obtained to minimize the cost of the grid users via linear programming approach. It is found that the resource allocation model can effectively allocate workloads to proper resources.

In [25], presents a clustering technique for gene expression data which can also handle incremental data. It is called GenClus and designed based on density based approach. Experimental results show the efficiency of GenClus in detecting quality clusters over gene expression data. Our approach improves the cluster quality by identifying sub-clusters within big clusters.

3. Proposed Scheduling Architecture

We represent a SLA aware scheduling architecture (Given at fig. 1) with making decision ability to achieve grid goals such as increasing resource utilization, number of completed jobs, percentage of user satisfaction, load balancing, etc. In the proposed model, SLA encompasses three parts– inception, decision and execution– that make decision based on request status and system condition what scheduler policy would be best way to achieve grid goals.
As shown fig. 2, requests and system status as input would be given to SLA. SLA selects some operations as action to operate on inputs. Based on action which be done on inputs, grid system status would be changed, on the other hands, grid goals would be changed. So it is important to apply best action for achieving grid goals.

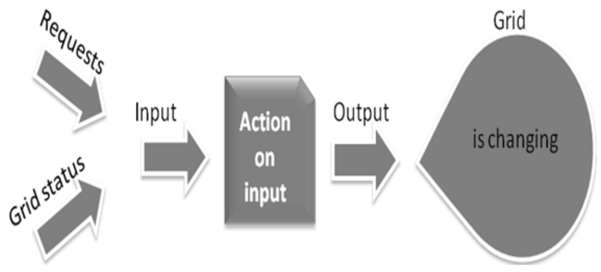

Fig.2. General representation of proposed procedure

3.1. How To Specify Inputs

In this section, we discuss the way for specifying requests and system status as inputs of our model.

3.1.1 Request Properties

SLA requires information in both requests (jobs) and system status, in order to making the decision to do a function (action) mapping inputs–requests and system status–to desired outputs. Since Deadline, service time, priority of jobs and system workload are very important factors for SLA to recognize current status and make decision what to act for producing sufficient output, we use these parameters to obtain required information and define a job as follow:

$$J =< R, Q, D, S, P >$$

Each job will have some requirements as resource, R, quality of service of resource, Q, job deadline, D, job service time, S, job priority, P.
Jobs prioritize based on applied action. For example, privileged program priority would be more than batch job priority. Job deadline is made on two parts, service time, S, and laxity, L, as shown in eq. 1. *Service time* is the time a job takes to finish executing on resources. *Laxity* is the time a job holds resources with no using, as shown in fig. 3.

$$D = S + L \qquad (1)$$

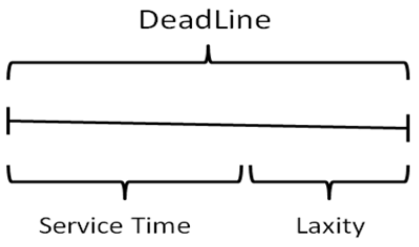

Fig. 3. Division of Deadline

Since there are numerous submitted jobs at the moment, selecting each job as discrete input and surveying them discretely will increase computing overhead. So, we suppose a time slot that jobs are surveyed at the start of time slot. Based on system workload, time slot fluctuates in time. It means if system workload is high, then time slot would grow until system workload decreased and vice versa.

For specifying jobs status as part of input, it requires to calculate mean of request deadline, μ_d, and mean of request service time, μ_s and also mean of request priority, μ_P, at start of time slot, as shown in eq. (2)-(4). In our model, jobs status would be defined by these three parameters.

Table 1. Characteristics of the proposed system

n	number of submitted jobs at the start of time slot
P_i	priority of ith request
D_i	deadline of ith request
S_i	service time of ith request
μ_d	mean of request deadline
μ_s	mean of request service time
μ_P	mean of request priority

$$\mu_d = \frac{\sum_{i=1}^{n} P_i \times D_i}{n} \qquad (2)$$

$$\mu_s = \frac{\sum_{i=1}^{n} P_i \times S_i}{n} \qquad (3)$$

$$\mu_P = \frac{\sum_{i=1}^{n} P_i}{n} \tag{4}$$

3.1.2 Specifying System Status

Inputs are job and system status. In previous section, we explained how to specify job status and in this section, we want to explain how to specify system status. For obtaining information about the system, there are some smart agents monitoring system status and alarming the status to the SLA. SLA notices obtained information about requests and system status as inputs and proceeds to set best action to do on the input so that desired output has been produced. In the next section, it will be described how to select best action regarding to inputs.

3.2 How To Choose Action

As explained later, based on inputs, SLA should choose a sufficient action to obtain desired output- resource utilization, number of completed jobs, percentage of user satisfaction, load balancing. It is difficult to map inputs to desired output. To solve the problem, we use clustering technique together with supervised learning. Cluster analysis or clustering is the process of grouping the objects into subsets so that the objects in subset are similar in some sense. Clustering is a method of unsupervised learning and a common technique for statistical data analysis used in many fields, including machine learning, data mining, pattern recognition, image analysis and bioinformatics [29]. Fig. 4 represents the clustering process. In this section, we discuss algorithms and methods implemented for grouping similar inputs and generating sufficient actions.

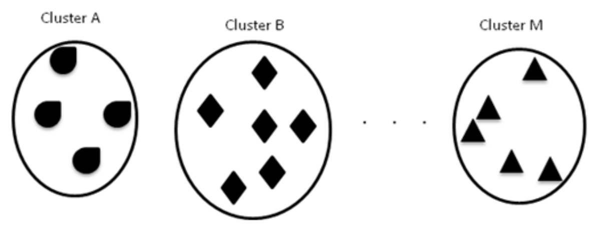

Fig. 4. representation of clusters

3.2.1 Clustering Algorithm For Grouping Inputs

Cluster is the same template of items– Input, Action and Output– which have been registered in tuple with fields are listed eq. 5. The first four parameters are related to inputs and the others are related to system and actions. As shown on eq. 6, system status comprises two options – load balancing rate and resources utilization rate – that alarm by smart agents. And as shown in eq. 7, request status comprises two field- percentage of user satisfaction and percentage of completed jobs. Percentage of completed jobs is computed easily but it needs to compute some parameters in order to obtain percentage of user satisfaction that it is shown in eq. 8. All parameters which exist in eq. 5-8, are listed on table 2.

$$\langle \mu_D, \mu_S, \mu_P, request - status, systemstatus_{current}, action_{related}, systemstatus_{next}, T_{Slot} \rangle \tag{5}$$

$$systemstatus = \langle load - balance, utilization\ rate \rangle \tag{6}$$

$$request - status = < percentage\ of\ satisfaction, percentage\ of\ completed\ jobs > \tag{7}$$

$$P_{satisfy} = p_i \left(D_i - \left(\frac{Ts_{ij}}{S_i^{min}} + \frac{L_i}{Bw_{ij}} T_{ij} \right) \right) + p_i'(1 - p_i)(D_i - T_c - T_N) \tag{8}$$

Table 2. Existing parameters in eq. 5-8

T_{Slot}	Determined time slot by system
$action_{related}$	Selected action for the input
$systemstatus_{next}$	System status after doing action on inputs
$status_{request}$	Status of requests which submitted to SLA
$systemstatus_{current}$	Current system status
$P_{satisfy}$	Percentage of user satisfaction
D_i	Deadline of job *i*
Ts_{ij}	The *i-th* job service time on the resource *j*
S_i^{min}	Speed of the slowest resource on which the job *i* can be executed
L_i	The size of a given job *i*
Bw_{ij}	The bandwidth between *i-th* job and the resource *j* on which the job can be executed
p_i'	Probability that *i-th* job submitted for the second negotiation
p_i	Probability that *i-th* job submitted for the first negotiation
T_{ij}	Required time to transfer data from *i-th* job to resource *j*
T_c	Completed job time
T_N	Required duration for the first negotiation

At first, there is no knowledge about system, on the other hands, no cluster exists. So selecting randomly an action to apply on requests called as inputs can cause undesirable results. For solving the problem, we early apply supervised technique. It means that grid system manger supposes some inputs which might happen in system and then manager makes decision which action or policy is conducted to goals. Supposed inputs and system status are classified on different clusters. Each cluster includes group of similar status, as shown in fig. 4.

As shown in fig. 5, we describe only six statuses with proposed action across all statuses which grid manager can suppose:

Status 1: Jobs deadline are too short and they have high priority. It means jobs should do as fast as possible with low rejection rate.

Action 1: Scheduler should use a quick searching algorithm to find sufficient resources with matching requests like hill climbing. It is better to find resources nearby request because transmitting jobs to remote resources would spend time. Even if there is no idle local resource, it is better to find resource by which young request, J_i, with long deadline is executed. The new request is sent to the resource and J_i is sent to remote idle resource.

Status 2: Jobs deadline are too short but no have high priority and system workload is high. And there is no scarcity of resources. It means jobs should be done fast in order to more accepted jobs. But it is never forget that load system is high.

Action 2: Scheduler should use a quick searching algorithm to find sufficient resources with matching requests like hill climbing.
It is better to find resources nearby request Because of transmitting jobs to remote resources would spend time. Also, Time slot, T_{Slot}, should grow because of heavy system load, it causes that fewer number of job would be submitted so system load would decline.

Status 3: Jobs deadline are normal but there is no load balancing.

Action 3: Scheduler should use a searching algorithm to provide system load balancing like BACO (Balance ant colony optimization).

Status 4: Jobs deadline is normal but system workload is high and there is scarcity of resources.

Action 4: Scheduler can use an Economic heuristic algorithm to find sufficient resources with matching requests like Game theory.

Status 5: Jobs deadline are too short. System does not have heavy system workload.

Action 5: Scheduler should use hill climbing algorithm to search resources nearby request.

Status 6: Jobs deadline are normal and there is load balancing.

Action 6: Scheduler use PSO algorithm to search resources.

As illustrated above, we only use four scheduling algorithms, PSO (partial swarm optimization), BACO (balanced ant colony optimization), Hill climbing and Economic based heuristic like game theory. Each algorithm can improve some grid factors. For example, BACO is capable of achieving system load balance better than other job scheduling algorithms and also economic heuristic deals with matching jobs to available resources in economical way such that resource provider and consumer get sufficient incentive to stay and play in competitive market [2, 3].
Each upper status makes a distinct cluster based on input, action and output. Input and action will be registered in related cluster but output will be registered after executing action on input and observing the result on grid.
Anyway, after requests submit to SLA, the new observation should be lied on clusters. For the purpose of grouping similar sets of inputs, we apply k-means clustering algorithm. The registered information on cluster is shown in eq. (5)-(7).
K-means is a clustering algorithm that, given an initial set of k means, assigns each observation to a cluster with the closest mean. It then calculates new means to be centers of observations in the clusters and stops when the assignments no longer change [22]. A cluster center is a newly generated input for a group of requests.
A frequent problem in k-means algorithm is the estimation of the number k. Two implemented approaches are explained as follow [23]:

1. *Rule-of-Thumb* is a simple but very effective method in which k is set to $\sqrt{N/2}$, where N is the number of entities.
2. *Hartigan's Index* is an internal index introduced in [24]. Let $W(k)$ represents the sum of squared distances between cluster members and cluster center for k clusters. When grouping n items, the optimal number k is chosen so that the relative change of $W(k)$ multiplied with the correction index $\gamma(k) = n - k - 1$ does not significantly change for $k + 1$,

$$H(k) = \gamma(k)\frac{W(k) - W(k+1)}{W(k+1)} < 10$$

The threshold 10 shown in Hartigan's index is also used in our model. It is "a crude rule of thumb" suggested by Hartigan [24].

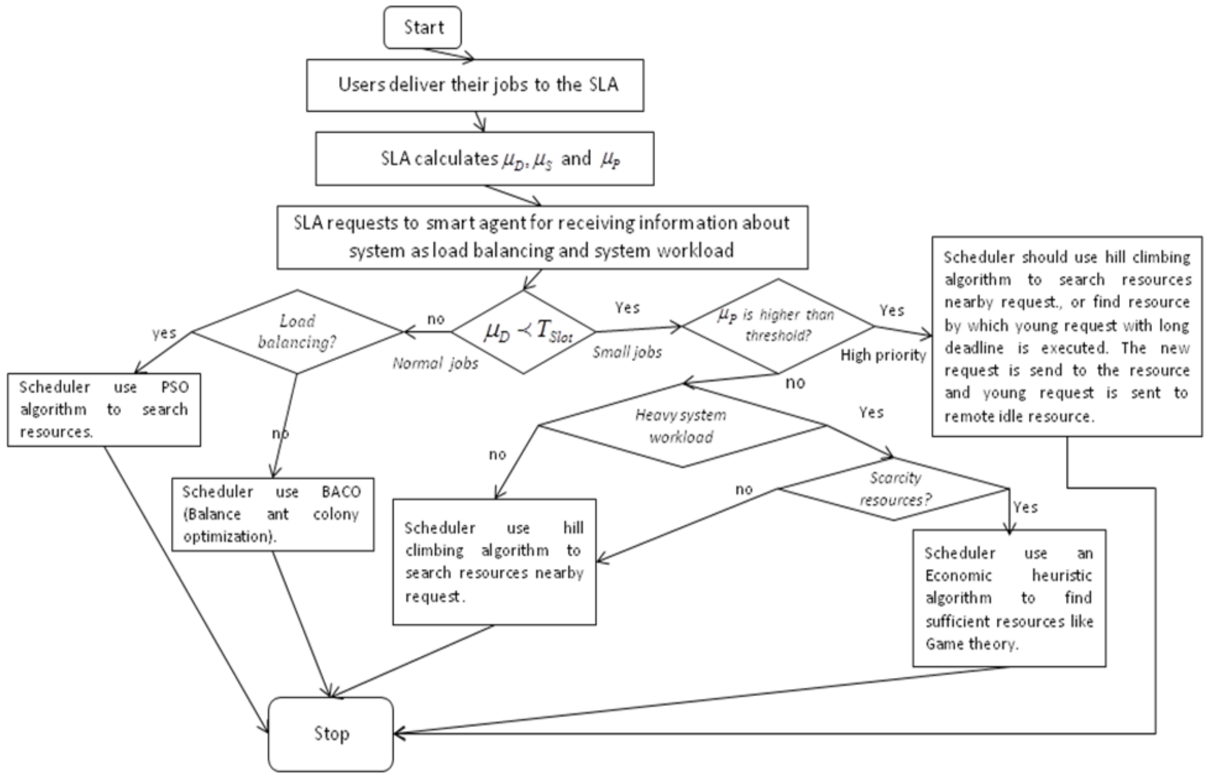
Start
Users deliver their jobs to the SLA
SLA calculates μ_D, μ_S and μ_P
SLA requests to smart agent for receiving information about system as load balancing and system workload
$\mu_D \prec T_{Slot}$
no
Normal jobs
Yes
Small jobs
Load balancing?
yes
no
Scheduler use PSO algorithm to search resources.
Scheduler use BACO (Balance ant colony optimization).
μ_P is higher than threshold?
Yes
High priority
no
Scheduler should use hill climbing algorithm to search resources nearby request., or find resource by which young request with long deadline is executed. The new request is send to the resource and young request is sent to remote idle resource.
Heavy system workload
no
Yes
Scheduler use hill climbing algorithm to search resources nearby request.
Scarcity resources?
no
Yes
Scheduler use an Economic heuristic algorithm to find sufficient resources like Game theory.
Stop

Fig.5. Describing six proposed initial statuses by system manager

3.2.2 Computing Distance Between Input Template

In order to utilize clustering algorithm, the measure of distance between two clustering items must be defined, as well as the distance between a clustering item and a cluster center. As already mentioned in the previous section, an item for clustering as called *Template* generally is a set of inputs, actions and outputs, while a cluster center is a master template for the given group of inputs.
Since having knowledge about the content of requests and about system status, we can exactly reconstruct templates. Therefore, both the distance between two clustering items as well as between an item and a cluster center can be reduced to computing distances between two templates. For this purpose, we introduce the n-tuple representation of decision. A decision consists of input, action and output parameters.
We distinguish between the structure of a decision, the list of input, action and output parameters with their names, and the values of a template, a list of numerical, boolean and string values of clustering items attributes. We introduce the n-tuple representation of a decision, where first $n-1$ elements contain values of the template, while the last element contains the template structure. By using such a representation, we can define the distance between two templates as an n-tuple, where first $n-1$ elements contain the differences between the two values of each of the parameters, while the final element contains a value representing the difference between the structures of the templates.
To formalize, we observe two decision templates T_1 and T_2 defined by their values $\{\alpha, \beta, \gamma, \dots\}$ and the template structure τ.

$$T_1 = (\alpha_1, \beta_1, \gamma_1, \dots, \tau_1)$$

$$T_2 = (\alpha_2, \beta_2, \gamma_2, \dots, \tau_2)$$

The n-tuple D_{T_1,T_2} representing the distance between two clustering templates is defined as eq.(9) [23].

$$D_{T_1,T_2} = \left(f(\alpha_1,\alpha_2), f(\beta_1,\beta_2), \dots, F(\tau_1,\tau_2)\right) \qquad (9)$$

The result of the function f for calculating the difference between two template values α_1 and α_2 depends on the type of its arguments and is defined as eq. (10) [23].

$$f(\alpha_1,\alpha_2) = \begin{cases} |\alpha_1 - \alpha_2|, & \textbf{if } \alpha_1 \text{ and } \alpha_2 \text{ are numerical} \\ 0, & \textbf{else if } \alpha_1 \text{ and } \alpha_2 \text{ are not numerical and } \alpha_1 = \alpha_2 \\ 1, & \textbf{else if } \alpha_1 \text{ and } \alpha_2 \text{ are not numerical and } \alpha_1 \neq \alpha_2 \end{cases} \qquad (10)$$

The distance between the structures of clustering templates is expressed as a number of differences between properties of templates. This value is calculated by iterating through all parameters contained by at least one of the cluster templates, calculating the distance between the templates with respect to the parameters. We define the distance function F calculating the difference between structures τ_1 and τ_2 of two templates T_1 and T_2 as shown in eq. (11) [23].

$$F(\tau_1, \tau_2) = \sum_{p \in T_1 \cup T_2} d_p\,(T_1, T_2) \qquad (11)$$

where the distance between two parameters of two cluster templates with respect to its properties is defined as shown eq. (12).

$$d_p(T_1, T_2) = \begin{cases} 0, & \textbf{if} \text{ name and metric of p are the same in } T_1 \text{ and } T_2 \\ 1, & \textbf{else if } T_1 \text{ and } T_2 \text{ does not contain p} \\ 1, & \textbf{else if} \text{ only name or metric of p differs in } T_1 \text{ and } T_2 \\ 2, & \textbf{else if} \text{ both name and metric of p differs in } T_1 \text{ and } T_2 \end{cases} \qquad (12)$$

After the result tuple has been calculated, it can be used to generate a single numerical value representing the distance between the two clustering templates. In order to do so, the result tuple must be normalized beforehand so that the tuple elements can be mutually comparable. Normalization is executed on each of the n tuple elements separately, where a value of an element is divided by a range of possible values for the element (maximum value minus the minimum value). Then, the final value representing the distance between clustering items can be computed by a simple function.
Note, in this paper, we discuss only the final element of the distance tuple, the difference between structures of two SLA templates. We plan to consider the values of SLA templates in our future work.

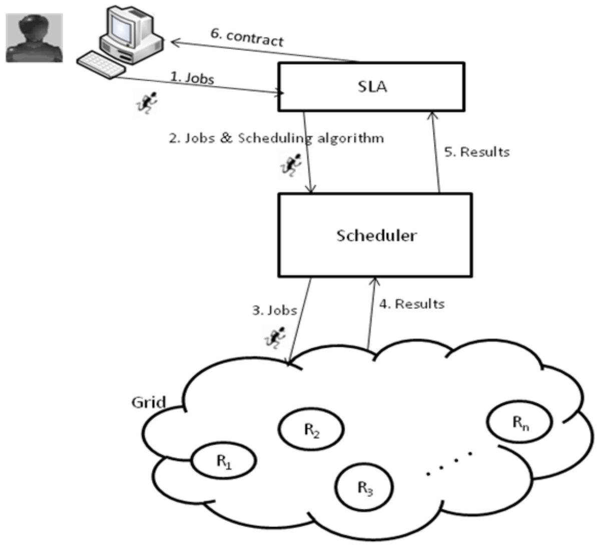

Fig. 6. Overview of the proposed model

However, SLA makes decision to choose sufficient action based on these clusters. If a status is found to which there is no matching cluster, SLA should make a new cluster. SLA selects clusters which their inputs are nearby current input and based on these clusters, SLA guesses the sufficient action which is better in this status. Then new cluster would be created. So, as time pass, number of clusters will be changed.
It is important to note if there is a status which there is no algorithm to improve all of grid goals, SLA selects the algorithm that is able to improve user factors (number of completed jobs and user satisfaction) instead of system factors. In our model, user has high priority.

3.3 Glimpse Of The Proposed Model

As shown in fig. 6, requests submit to SLA. SLA surveys and analyses current grid status and received requests and based on input distinguishes related cluster for selecting sufficient scheduling policy to improve grid factors.
For example, it is possible that a status happens which SLA recommend user to change his job deadline otherwise job would be reject or other bad events happen. If user accepts, SLA's suggestion would be executed otherwise, user should resubmit his job. Described scenario of grid operation is illustrated in fig. 7.

3.4 SLA Infrastructure

In the proposed model, SLA encompasses three parts– inception, decision, execution– that make decision based on request status and system condition what scheduler policy would be best policy to achieve grid goals as shown in fig. 1.

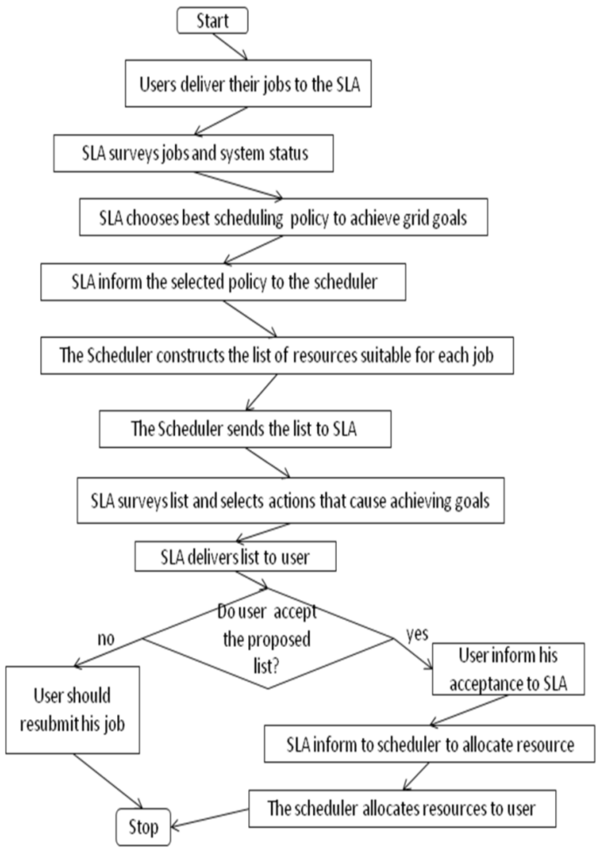
Start
Users deliver their jobs to the SLA
SLA surveys jobs and system status
SLA chooses best scheduling policy to achieve grid goals
SLA inform the selected policy to the scheduler
The Scheduler constructs the list of resources suitable for each job
The Scheduler sends the list to SLA
SLA surveys list and selects actions that cause achieving goals
SLA delivers list to user
Do user accept the proposed list?
no
yes
User should resubmit his job
User inform his acceptance to SLA
SLA inform to scheduler to allocate resource
The scheduler allocates resources to user
Stop

Fig.7. Describing scenario of grid operation

3.4.1 Inception Unit

When user submits his job to SLA, it is received by inception unit. The unit is responsible for surveying requests and system status as input and calculating mean of request deadline, μ_d, and mean of request service time, μ_s and also mean of request priority, μ_P. And also communicates with smart agents to gain system status. Then sends obtained information to the decision unit.

3.4.2 Decision Unit

This unit is responsible to survey received result from the inception unit and determine the cluster which is related to current situation. If it distinguishes the cluster, then send the related action to the execution unit. But while no cluster match, the unit should create new cluster with current status as its input and choose its sufficient action using the approximately similar clusters but cannot specify its real output. The output would be specified after the action is done.
However, the unit chooses the action then sends it to the execution unit. After scheduler do the action and sends result to the decision unit, the unit surveys the result and selects best actions that best result would be provided for requests and system.

3.4.3 Execution Unit

It is responsible to send parameters to which scheduler requires executing selective action. Based on proposed action, scheduler finds resources then sends the result to the decision unit.

4. Relationship Between Number Of Completed Jobs and Utilization

In real world, there is no detailed mathematic relationship for most of phenomenon, so that specifying one phenomenon cause specifying the other one. For example, suppose that there is relation between company publicity and number of sell. The relation often is not precise, so that we can say if spend n dollar for publicity, company would sell q number of stuff. The relation that is between number of completed jobs and utilization is instance of this kind of relationship. First, draw transmittal diagram and then select best line which have minimum variance respect to spots in diagram. In this part, we intend to obtain relationship, $\hat{r}$, between the number of completed jobs, NCr, and system utilization, U, as illustrated in eq. (5) – (9). For specifying the relationship it needs n experiments which number of completed jobs, NCr_i, and system utility, U_i, in i-th experiment should measure in each experiment.

$$\mu_{NCr} = \sum_{i=1}^{n} \frac{NCr_i}{n} \tag{5}$$

$$\mu_U = \sum_{i=1}^{n} \frac{U_i}{n} \tag{6}$$

$$b = \frac{\sum_{i=1}^{n} NCr_i \times U_i - n \times M_{NCr} \times \mu_U}{\sum_{i=1}^{n} {NCr_i}^2 - n \times {\mu_{NCr}}^2} \tag{7}$$

$$a = \mu_U - b \times \mu_{NCr} \tag{8}$$

$$\hat{r} = a + b \times NCr \tag{9}$$

5. Conclusion

The following paper describes a novel SLA-aware model to schedule tasks efficiently in a grid environment. To apply best scheduling policy, this model uses clustering technique. Clustering is a method of unsupervised learning, and a common technique for statistical data analysis. Clustering is the assignment of a set of observations into subsets (called *clusters*) so that observations in the same cluster are similar in some sense. We put past scheduling experiments in disjoint clusters and in future, we use clusters to choose best scheduling policies.

References

1. Y. Gao, H. Rong, J. Z. Huang, "Adaptive grid job scheduling with genetic algorithms", J. Future Generation Computer Systems, Vol. 21, 2005, pp. 151-161.
2. Y. S. Dai, X. L. Wang, "Optimal resource allocation on grid systems for maximizing service reliability using a genetic algorithm", Reliability Engineering and System Safety, Vol. 91, 2006, pp. 1071-1082.
3. A. V. Chandak, B. Sahoo, A. K. Turuk, "Heuristic Task Allocation Strategies for Computational Grid", Int. J. Advanced Networking and Applications, Vol. 2, 2011, pp.804-810.
4. D. D. H. Miriam, K. S. Easwarakumar, "A Double Min Min Algorithm for Task Metascheduler on Hypercubic P2P Grid systems", International Journal of Computer Science Issues, Vol. 7, No. 5, 2010, pp. 8-18.
5. B. Zeng, J. Wei, H. Liu, "Dynamic Grid Resource Scheduling Model Using Learning Agent", IEEE International Conference on Networking, Architecture, and Storage (NAS’09), 2009, pp. 67-73.
6. N. Malarvizhi, V. R. Uthariaraj, "A Minimum Time To Release Job Scheduling Algorithm in Computational Grid Environment", Proceedings of Fifth International Joint Conference on INC, IMC and IDC, 2009, pp. 13-18.
7. D. P. Spooner, S. A. Jarvis, J. Cao, S. Saini, G. R. Nudd, "Local Grid Scheduling Techniques using Performance Prediction", IEEE Proceedings of Computers and Digital techniques, 2003, pp. 87–96.
8. S. Bansal, B. Kothari, C. Hota, "Dynamic Task-Scheduling in Grid Computing using Prioritized Round Robin Algorithm", International Journal of Computer Science Issues, Vol. 8, 2011, pp. 472-477.
9. M. Cochran, P. D. Witman, "Governance and Service Level Agreement Issues in A Cloud Computing Environment", Journal of Information Technology Management, Vol. 22, No. 2, 2011, pp. 41-55.
10. T. Altameem, "On the Design of Job Scheduling Strategy Using Agent Replication for Computational Grids", IJCSNS International Journal of Computer Science and Network Security, Vol. 11, No. 3, 2011, pp. 269-276.
11. J. Li, J. Peng, Z. Lei, W. Zhang, "An Energy-efficient Scheduling Approach Based on Private Clouds", Journal of Information & Computational Science, Vol. 8, No. 4, 2011, pp.716-724.
12. C. Ray, N. Guha, "Determination of Cost Model for Constraint based Query Optimization in Data Grids", Proceedings of International Conference on Advances in Computer Science, 2010, pp. 237-240.
13. L. M. Khanli, M. Etminan Far, A. Ghaffari, "Reliable Job Scheduler using RFOH in Grid Computing", Journal of Emerging Trends in Computing and Information Sciences, Vol. 1, No. 1, 2010, pp. 43-47.
14. M. Amoon, M. Mowafy, T. Altameem, "A Multiagent-Based System for Scheduling Jobs in Computational Grids", ICGST- AIML journal, Vol. 9, 2009, pp. 19-27.
15. M. Hovestadt, "Operation of an SLA-aware Grid Fabric", Journal of Computer Science, Vol. 2, No. 6, 2006, pp. 550-557.
16. R. J. S. Raj, V. Vasudevan, "Beyond Simulated Annealing in Grid Scheduling", International Journal on Computer Science and Engineering (IJCSE), Vol. 3, No. 3, 2011, pp. 1312-1318.
17. R. M. R. Kovvur, S. Ramachandram, V. Kadappa, A. Govardhan, "A Reliable Distributed Grid Scheduler for Independent Tasks", International Journal of Computer Science Issues, Vol. 8, 2011, pp. 296-301.
18. H. Izakian, B. T. Ladani, A. Abraham, V. Snasel, "A Discrete Particle Swarm Optimization Approach For Grid Job Scheduling", International Journal of Innovative Computing, Information and Control, Vol. 6, No. 9, 2010, pp. 1-15.
19. A. Revar, M. Andhariya, D. Sutariya, "Load Balancing in Grid Environment using Machine Learning-Innovative Approach", International Journal of Computer Applications, Vol. 8, No. 10, 2010, pp. 31-34.
20. G. D. Parmar, S. K. Mitra, "Performance Analysis of Unsupervised Probabilistic", IJCA Special Issue on Computer Aided Soft Computing Techniques for Imaging and Biomedical Applications CASCT, 2010, pp. 93-98.
21. J. MacQueen, "Some methods for classification and analysis of multivariate observations", Proceedings of the fifth Berkeley Symposium on Mathematical Statistics and Probability, 1967, Vol. 1, pp. 281-297.
22. K. V. Mardia, J. T. Kent, "Multivariate Analysis", Academic Press, 1980.
23. I. Breskovic, M. Maurer, V. C. Emeakaroha, I. Brandic, J. Brandic, "Towards Autonomic Market Management in Cloud Computing Infrastructures", In Proceedings of CLOSER, 2011.
24. J. A. Hartigan, Clustering Algorithms, John Wiley & Sons Inc, 1975.
25. S. Sauravjyoti, D. K. Bhattacharyya, "An Effective Technique for Clustering Incremental Gene Expression data", International Journal of Computer Science Issues, Vol. 7, No. 3, 2010, pp. 31-41.
26. G. Murugesan, Dr. C. Chellappan, "An Economic-based resource Management and Scheduling for Grid Computing Applications", International Journal of Computer Science Issues, Vol. 7, No. 2, 2010, pp. 20-25.
27. U. Karthick Kumar, "A Dynamic Load Balancing Algorithm in Computational Grid Using Fair Scheduling", International Journal of Computer Science Issues, Vol. 8, No. 1, 2011, pp. 123-129.
28. K. S. Chatrapati, J. U. Rekha, A.V. Babu, "Competitive Equilibrium Approach for Load Balancing a data Grid", International Journal of Computer Science Issues, Vol. 8, 2011, pp. 427-437.
29. S. R. Nimmagadda, P. Kanakamedla, V. B. Yaramala, "Implementation of Clustering Through Machine Learning Tool", International Journal of Computer Science Issues, Vol. 8, 2011, pp. 295-401.

30. T. R. V. Ananadharajan, Dr. M. A. Bhagyaveni, "Co-operative Scheduled Energy Aware Load-Balancing technique for an Efficient Computational Cloud", International Journal of Computer Science Issues, Vol. 8, 2011, pp. 571-576.

Seiiedeh Yasaman Rashida received his B.S. in computer engineering from Sari Azad University, Mazandaran, in 2006, the M.S. in computer engineering from Arak Azad University, Markazi, in 2010. She is a teacher at Shirgah University in Mazandaran. Her research interests include grid computing and Fuzzy logic. She has some papers in these fields.

Amir Masoud Rahmani received his B.S. in computer engineering from Amir Kabir University, Tehran, in 1996, the M.S. in computer engineering from Sharif University of technology, Tehran, in 1998 and the PhD degree in computer engineering from IAU University, Tehran, in 2005. He is assistant professor in the Department of Computer and Mechatronics Engineering at the IAU University. He is the author/co-author of more than 80 publications in technical journals and conferences. He served on the program committees of several national and international conferences. His research interests are in the areas of distributed systems, ad hoc and sensor wireless networks, scheduling algorithms and evolutionary computing.