A COMPARATIVE STUDY IN DYNAMIC JOB SCHEDULING APPROACHES IN GRID COMPUTING ENVIRONMENT

Amr Rekaby[1] and Mohamed Abo Rizka[2]

[1] Egyptian Research and Scientific Innovation Lab (ERSIL), Egypt
[2]Arab Academy for Science, Technology and Maritime Transport College of Computing & Information Technology, Cairo, Egypt

ABSTRACT

Grid computing is one of the most interesting research areas for present and future computing strategy and methodology. The dramatic changes in the complexity of scientific applications and part of non-scientific applications increase the need for distributed systems in general and grid computing specifically. One of the main challenges in grid computing environment is the way of handling the jobs (tasks) in the grid environment. Job scheduling is the activity to schedule the submitted jobs in the grid environment. There are many approaches in job scheduling in grid computing.

This paper provides an experimental study of different approaches in grid computing job scheduling. The involved approaches in this paper are "4-levels/RMFF" and our previously published approach "X-Levels/XD-Binary Tree". First of all, introduction to grid computing and job scheduling techniques is provided. Then the description of currently existing approaches will be presented. After that, experiments and provided results give a practical evaluation of these approaches from different perspectives. Conclusion of the comparative study states that overall average tasks waiting time is enhanced by approximately 30% by using the X-levels/XD-binary tree approach against 4-levels/RMFF approach.

KEYWORDS

Grid computing, dynamic jobs scheduling, job scheduling algorithms experimental study, X-Levels/XD-Binary Tree approach evaluation

1. INTRODUCTION

Grid computing is a modern generation of distributed computing. The target of grid paradigm is how to construct strong power processing and storage resources by many small and weak resources integration. Gird computing is a mesh of interconnected resources which construct massive powerful capabilities. Grid computing is composed of many resources from different platforms and specifications (heterogeneous resources) not like regular "Distributed Processing" which based on similar resources (homogeneous resources)[1].

The user of the grid can use any (or many) of these interconnected resources in the environment to solve his problems, which cannot be solved by locally owned resources capabilities [2, 3].
Jobs scheduling problem is the problem that is raised when a resource in the grid submits tasks and the manager of the grid would like to find a suitable resource to host these tasks.

DOI: 10.5121/ijgca.2013.4301

The selection of the resource depends on the task's needs like operating system type, processor speed, memory capacity, and other needs comparing with the resources specifications.

Load balancing problem sometimes comes from unbalanced load distribution over distributed (grid) resources. The problem exists if some resources in the grid are overloaded (which lead to a delay in the completion time of the tasks) while other resources are unutilized. Here, the role of the scheduling algorithm appears. Scheduling algorithm should manage this issue and provide smart and dynamic scheduling to utilize all grid resources as much as possible. In case of failure happened the taking over of the tasks is also part of the scheduling and load balancing solution responsibilities [4]. The rest of this paper is organized as follows: section two describes a "4-levels/RMFF" job scheduling approach. Section 3 presents "X-Levels/XD-Binary Tree" Job Scheduling Approach. Afterwards, section four shows the contribution of this research which is experimental works to provide a comparison between the two job scheduling approaches. A conclusion is provided in section five.

2. 4-LEVELS/RMFF JOB SCHEDULING APPROACH

The job scheduling approaches are categorized based on two main factors: the overall grid structure including the communication topology, and the job scheduling algorithms used in each component in the grid structure including the participation of each role in the job scheduling activities.

In 4-levels model, the grid resources are connected on a form of tree structure. The grid is divided over clusters, in each cluster there is an owner (cluster manager). Each cluster is constructed from different sites. Inside each site, the actual resources of the grid exist. The naming of 4-levels comes from the levels of this model (grid, cluster, site and resources levels). Figure 1 represents G/S/M example of 4-levels tree model. This example indicates that: the grid contains (G) clusters, (S) sites inside each cluster, and (M) resources exist in each site. Dotted rectangle in figure 1 shows the subgroups in this example like 1/S/M, which means cluster (1) with (S) sites under it, containing (M) resources in each site. Dashed rectangles describe model 1/1/M, which indicates cluster (1), site (1) with (M) resources inside this site [5].

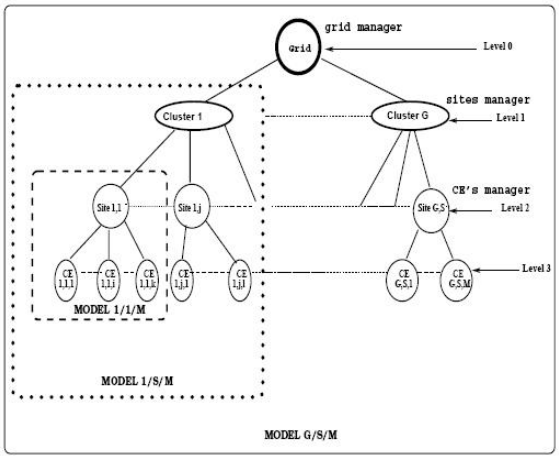

Figure 1. Grid Environments, 4-Levels Tree Topology [5]

One of the advantages of this model is the organized connectivity, which gives the administrator of the grid the window of organizing it into sites and cluster according to his preferences (considering four levels limitation).

The work flow of 4-levels/RMFF approach is described in figure 2.

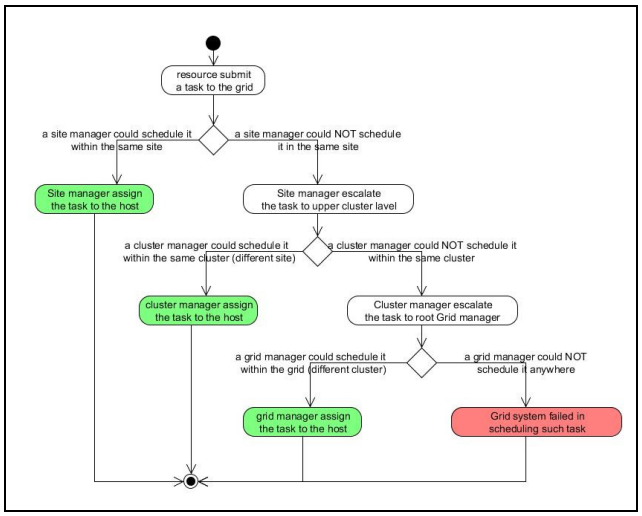

Figure 2. Grid Environment, 4-Level Tree work flow Logic

As described in figure 2, each group of resources (site or cluster) has an internal preference to its group resources higher than other resources. By applying this rule, a resource in site (A) will run its tasks in another resource within the site (A) if it is possible. If it is not possible, the tasks should run in the same cluster, in sibling site (if possible). If the cluster could not serve these tasks, the tasks could run anywhere in the grid environment.

According to 4-levels tree model, there are three types of job scheduling algorithms scopes: intra-site, intra-cluster and intra-grid scheduling scopes. In Intra-site scheduling algorithms, there are a lot of factors that could be considered during the scheduling. Depending on the complexity of task-resource matching, these factors are considered or not. Dynamic scheduling could consider static resource features like processor speed, operating system type. It also should consider dynamic factors like the processor utilization status, this is happened by continues resource monitoring [6]. In all cases, the algorithm has two inputs, the first input is the task needs (hardware, network, utilization needs), and the second one is the available resources capabilities. RMFF (Rare Monotonic scheduling with the First-Fit task allocation) is one of the most commonly used search algorithms in intra-site level [7, 8]. So 4-levels/RMFF approach uses 4-levels tree model as overall grid model and RMFF as a search algorithm in intra-site level. Regarding the intra-cluster and intra-grid scopes, there is no intelligent way used in these scopes. As mentioned in figure 2, if the site failed to schedule a task, so it is escalated to the upper cluster, which push it into any site under its supervision using a “try and error” technique. If the new site can host this task, the scheduling will be completed. If it cannot host the task, the cluster will push the task to another site, and so on. If the cluster fails in scheduling the task to any of its child sites, it will escalate it to the grid level (root level in 4-levels tree model).

3. X-LEVELS/XD-BINARY TREE JOB SCHEDULING APPROACH

After describing a scheduling approach (4levels/RMFF) in last section, the paper describes the second comparable scheduling approach (X-levels/XD-binary tree). X-levels model is an inherited version of 4-levels model [9]. In X-levels model, the grid is constructed from X levels of layers, it is not limited to just four levels. As shown in figure 3, a cluster could be a parent of another cluster or site or both. It indicates that, there are intra-site and intra-cluster scheduling algorithms. The intra-grid scope that exists in 4-levels tree, is vanished in x-levels model.

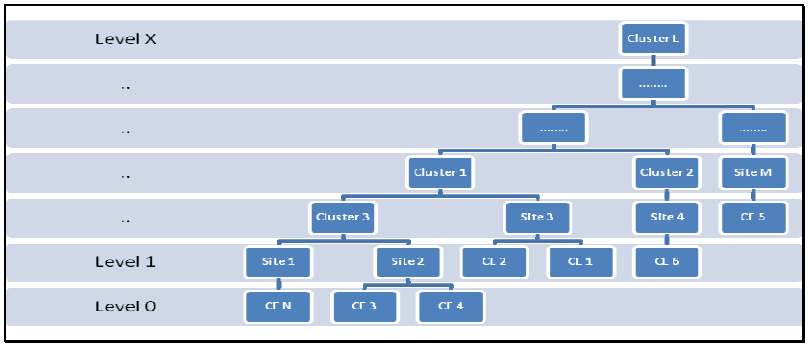

Figure 3. Grid Environments, X-Levels Tree Topology

In X-levels tree, there are different types of components in the grid environment. The following categories are explaining these components types [9]:

- Computation Elements (CE): a client machine that could submit or host tasks.
- Site Resource Broker (SRB): a site manager responsible manager for task scheduling and task-resource matching algorithm execution.
- Site Information Service Catalog (SISC): a site information repository that hosts all site information and relatives resources up to date statuses.
- Cluster Resource Broker (CRB): a cluster manager responsible manager for task scheduling on cluster level.
- Cluster Information Service Catalog (CISC): a cluster information repository that hosts all cluster information and relatives child sites or clusters updated statuses.

In X-levels/XD-binary tree approach, the used scheduling algorithm in intra-site level is "XD-binary search" algorithm. XD-binary search algorithm is presented by an activity diagram in figure 4 [10].

In Intra-cluster scope, "X-Levels/XD-binary tree" approach uses "Cluster Job Scheduling Search" algorithm in job scheduling activities. "Cluster Job Scheduling Search" is presented by its activity diagram in figure 5 [9].

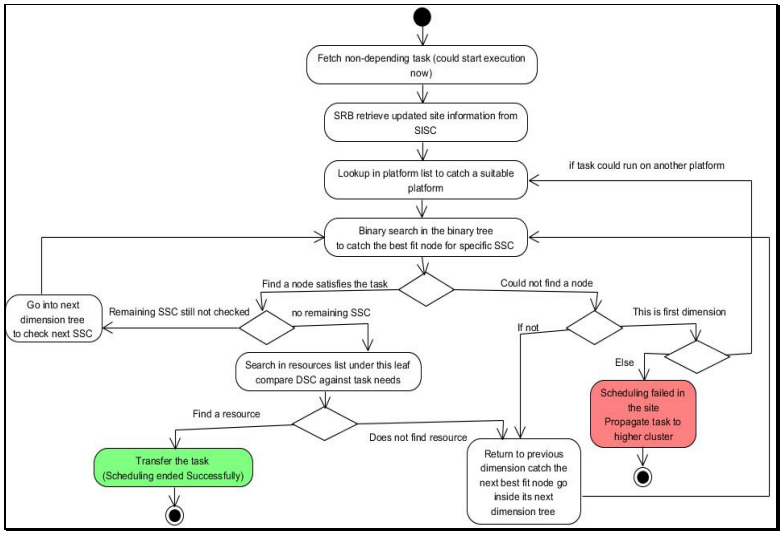

Figure 4. XD-binary searches (SRB search algorithm in XD-binary tree data structure)

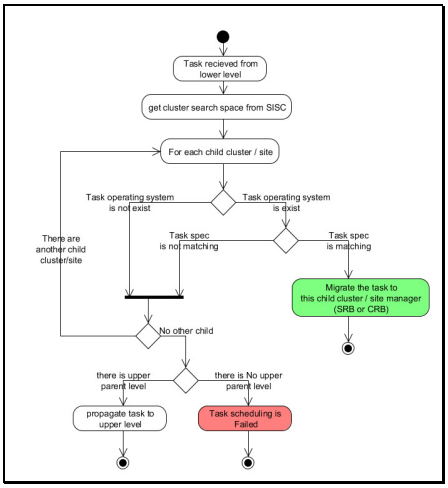

Figure 5. Cluster Job Scheduling Search Algorithm

3. SIMULATION AND RESULTS

In this section, the applied simulation experiments are presented, and their results are shown. Table 1 represents a comparison between the already existing solutions that are addressed in this paper.

Table 1. Simulation environments comparison

Features	4-Levels/RMFF Approach	X-Levels/XD-binary tree Approach
Approach label	4-Levels/RMFF	X-Levels/XD-binary tree
Overall grid model	4-Levels tree	X-Levels tree
Intra-site scheduling algorithm	RMFF resource allocation	XD-binary tree search algorithm
Intra-cluster scheduling algorithm	Try and error methodology	Cluster Job Scheduling Search Algorithm
Intra-grid scheduling algorithm	Try and error methodology	N/A

The experiments were implemented by simulation environments. These environments are built using a single machine; it simulates virtually different entities' roles. The used machine actual physical specifications are:

- Windows XP professional, SP 3.
- Intel core 2 Duo CPU.
- 2996 Mb Ram total size.
- 2.8 GHz processor speed.
- 32-B Memory bandwidth.

Two simulators were implemented, one for X-levels/XD-tree approach and another one for 4-levels/RMFF approach. The simulation environment acts as different virtual machines communicating through messages.

Benchmark is created for these experiments; the same benchmark is used for both approaches. The benchmark contains two sections: input tasks' needs, and resources' specifications. The test benchmark is created randomly to have many random lists of tasks and resources. Each property of task or resource has a valid predefined set of values; a random method is used to create 100-tasks, 200-tasks, and 500-tasks lists. On the other hand, a random method is used to create 100-resources, and 200-resources lists. The experiments are run with fixed tasks count and changing the resources count, and vice versa.

According to X-levels model, different grid hierarchy can be created, they are mentioned in the experiments results, X(5)-levels means a hierarchy from 5 levels is used as implementation of X-levels approach. The average input tasks' burst time was 10 seconds in all trials.
All the experiments are repeated for ten times, the mean value is captured to be presented in the provided results graphs.

The next paragraphs describe the comparisons results between 4-levels/RMFF and X-levels/XD-tree approaches from completion time, waiting time, working time, and data structure handling (updating) factors.

As in figure (6) with fixed resources count and changing the tasks count, there is a better performance and shorter completion time in X(4)-levels/XD-binary search tree model than 4-levels/RMFF model. By graph, this difference of performance increases with tasks count increase.

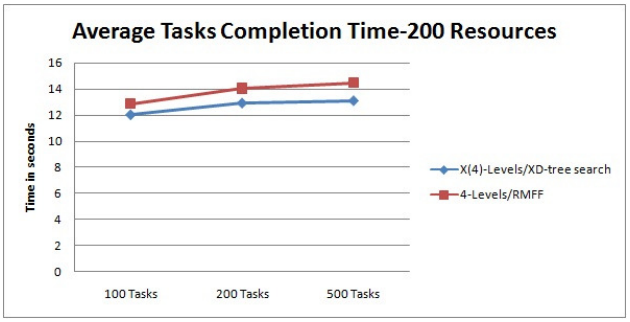

Figure 1. Average tasks completion time with 200 resources

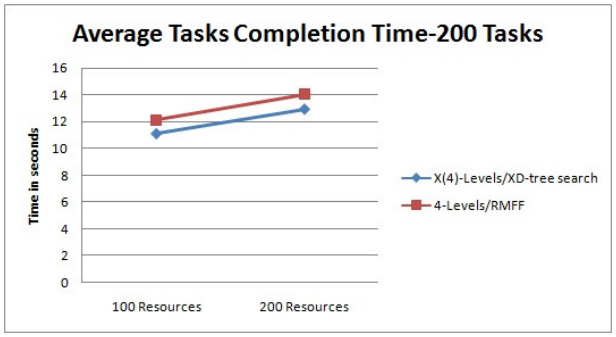

Figure 2. Average tasks completion time with 200 tasks

Figure (7) shows that, there is a better performance and shorter completion time in 4-levels/XD-binary search tree model than 4-levels/RMFF model when tasks count is fixed and resources counts are changed.

In the following figures, the concentration is pointed to the tasks waiting time which reflect the actual scheduling approach performance.

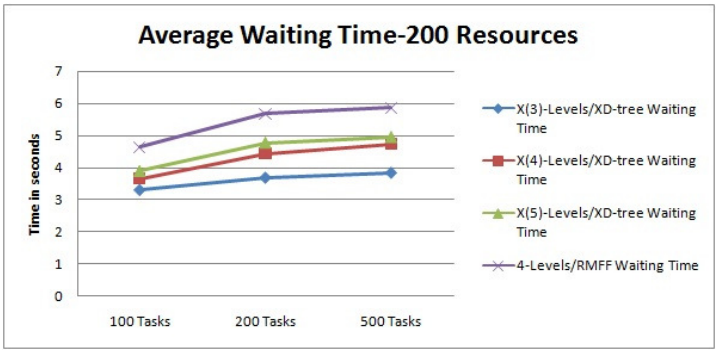

Figure 8. Average waiting time with 200 resources

Figure (8) shows that, there is a better performance and shorter waiting time in X-levels/XD-tree model than 4-levels/RMFF model. By graph, this difference of performance increases with tasks count increase. This is aligned with the previous results that, there is a difference between completion times. The waiting time is mainly the consumed time in the scheduling logic. X-levels in this experiment are applied with three, four and five levels. The results mention that 3-

levels model is the best one. The decision of how many levels is the best ever is not a solid decision; it depends on the resources grouping in the tree, if resources need to be grouped for specific reason (high configuration resources are grouped in specific sites for example) so extending the levels would help in this target achievement. In general, we don't have to add extra levels grouping if there is no reason for that (administration or architecture grouping reason).

The same is happened when tasks count is fixed and resources count is changed as presented in figure (9).

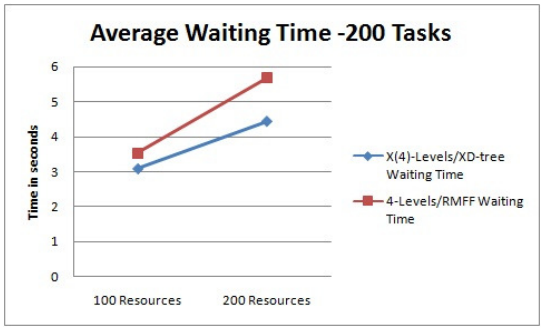

Figure 9. Average waiting time with 200 tasks

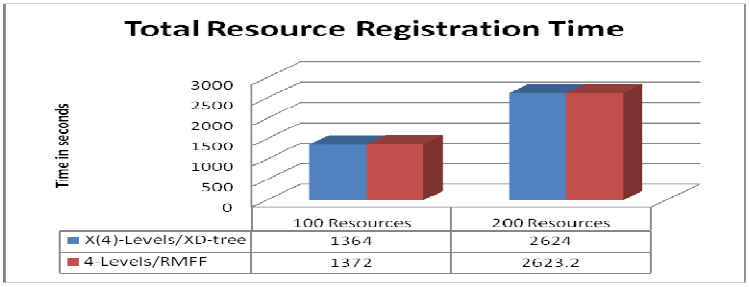

Figure 10. Average resources registration time

X-levels/XD-tree solution is mainly based on the XD-binary tree data structure, so there is a doubt that: it could affect the registration time of the resource. The registration time is the needed time for register a new resource to the grid and let this resource join the site. It is calculated starting from the resource request to join the site, till it is registered in the site search space and a confirmation message is sent to the resource from the site manager. Taking into consideration that, RMFF uses a flat array as a site search space repository, while XD-binary tree search uses XD-binary tree data structure as a site search space as presented in research [10].

Figure (10) shows that, there is no big difference performance between the two solutions. So this could not be taken as a disadvantage of X-levels/XD-binary tree solution.

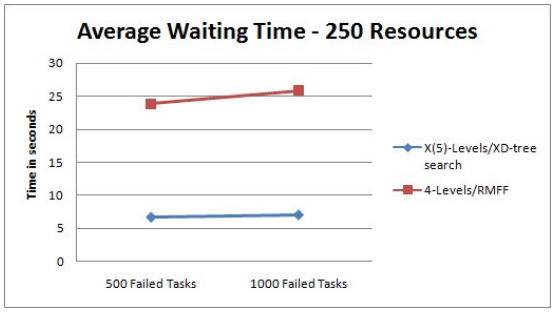

Figure. 11. Average failed tasks waiting time with 250 resources

In figure (11), experiments with improper tasks are applied, these tasks need resources' capabilities that do not exist in the grid environment. X-levels/XD-binary tree could shortcut the search space and find the solution shortly. Also if the proper resource does not exist, it could find that by fewer steps than needed in RMFF algorithm. By graph, this difference of performance is extraordinary in this case and has a direct proportional relation with tasks count.

5. CONCLUSIONS

The experiments that are provided in this paper used different number of tasks and resources in a combinational way. The experiments present that, "X-levels/XD-tree" is more efficient than "4-levels/RMFF" approach. The scheduling time is the main enhanced point in X-levels/XD-tree model, the model do the scheduling activity in less time than 4-levels/RMFF approach. The minimizing of the scheduling time reflects on the overall grid performance through the actual tasks' completion time.

The research approaches the role of the levels in X-levels model and its relation with the overall grid performance. It states that, it depends mainly on resources grouping in the grid environment. The added or removed levels of the grid hierarchy might enhance the performance of the scheduling time, and also it might not do, according to the resources capabilities and grouping as described in this paper.

In this paper, experiments are provided with a benchmark of failed tasks (tasks that need a specification that does not exist in the grid). X-levels/XD-tree approach is significantly better that 4-levels/RMFF approach in failed tasks case. XD-tree data structure and relevant search algorithm that is used in X-levels/XD-tree approach give the scheduler the opportunity to decide faster if the needed requirements do exist in such grid available resources, or it shortcut the way to the conclusion that these requirements don't exist. On the other hand, RMFF algorithm needs more time to cover all the search space, and this time will be directly proportion affected by tasks count and resources count.

Although XD-binary tree data structure might be seen as a complex data model that affects the grid performance negatively, but the experiments present that, it doesn't affect the grid resources registration time by any means. The time consumed in new resource registration to the grid is not apparently affected by XD-binary tree data structure.

Overall average tasks waiting (scheduling) time is enhanced by approximately 30% by using the X-levels/XD-binary tree approach against 4-levels/RMFF approach.

Although XD-binary tree search algorithm gets the best fit resource allocation based on the tasks' needs, task-resource efficiency is not covered in this paper; the main focus here is the job scheduling consuming time.

REFERENCES

[1] International Technical Support Organization "Introduction to Grid Computing" – IBM Red Book – December 2005.

[2] Igor Sfiligoi, "Making Science in the Grid World:Using Glideins to Maximize Scientific Output", 2007 IEEE Nuclear Science Symposium Conference Record.

[3] R. Al-Khannak, B. Bitzer, "Load Balancing for Distributed and Integrated Power Systems using Grid Computing", 2007 IEEE, ISBN: 1-4244-0632-3, pages 123- 127.

[4] Babar Nazir, Taimoor Khan,"Fault Tolerant Job Scheduling in Computational Grid", IEEE--ICET 2006 2nd International Conference on Emerging Technologies Peshawar, Pakistan, 13-14 November 2006.

[5] Belabbas Yagoubi, Yahya Slimani, "Dynamic Load Balancing Strategy for Grid Computing", PROCEEDINGS OF WORLD ACADEMY OF SCIENCE, ENGINEERING AND TECHNOLOGY VOLUME 13 MAY 2006 ISSN 1307-6884, pages 260- 265.

[6] Cui Zhendong, Wang Xicheng, "A Grid Scheduling Algorithm Based on Resources Monitoring and Load Adjusting", 2008 IEEE, ISBN: 978-1-4244-3531-9, pages 873 - 876.

[7] Raja Talili, Yahya Slimani "Mining Association Rules on Grid Platform" http://www.di.unipi.it/CGWS2011/MiningAssocRulesOnGrid.swf

[8] ROBERT I. DAVIS/ALAN BURNS "A Survey of Hard Real-Time Scheduling for Multiprocessor Systems" ACM Computing Surveys, Vol. 43, No. 4, Article 35, Publication date: October 2011.

[9] Amr Rekaby, Mohamed Abu Rizkaa, "The Improvement of Dynamic Load Balancing Strategy in Grid Computing" The International Conference on Grid Computing and Applications, 2011.

[10] M. Abo Rizka, A. Rekaby "Dynamic Job Scheduling and Load balancing algorithm In Grid Environment via X-Dimension binary tree data model", International Journal of Intelligent Computing & Information Science, volume 12, no 2, July 2012.

[11] Yu Liang, Zhou Jiliu, "The Improvement of A Task Scheduling Algorithm in Grid Computing", First International Symposium on Data, Privacy and E-Commerce.

Author

Amr Rekaby is MSc in computer science (Grid Computing specialization) in 2012, worked as a technical coach / leader for 8 years in multinational companies like IBM and HP. He is now a founder of "Egyptian Research and Scientific Innovation Lab" (ERSIL), where he is working as a researcher in AI, software engineering and parallel computing fields.