Scheduling Jobs in Face of Status Update Timing of Resources in Computational Grids

M. Amoon[1] and H. M. Faheem[2]
[1]*Computer Science and Eng. Dept., Faculty of Electronic Eng., Menofia University, Egypt*
King Saud University, P. O. Box: 28095-11437 Riyadh-Saudi Arabia.
[2]*Faculty of Computer and Information Science, Ain Shams University, Egypt.*
m_amoon74@yahoo.com, hmfaheem@btexperts.com

Abstract

Users and resources frequently join and leave computational grid, hence the state of the grid changes dynamically. So, an effective job scheduling strategy is needed that consider the dynamically changed conditions in the grid. Most of the existing scheduling strategies are mainly based on selecting resources that have less resource response time or less grid bandwidth while these resources may be out of date at the scheduling time. This paper proposes a scheduling strategy with the objective of minimizing the effects of grid dynamism on the grid performance. The strategy schedules jobs to the resources according to status update timing combined with response time of the resources. The proposed strategy is compared with the time-based strategy in terms of the number of jobs executed successfully within the specified deadline and the makespan of user applications. Experimental results have shown that the proposed strategy can considerably improve the performance of the grid.

Keywords*: Job scheduling, makespan, computational grid, grid dynamism*

1. Introduction

Computational grids allow large-scale resource sharing in widely connected networks such as the Internet. Resources in grids can include computing systems, storage systems, date sources, distributed applications and management systems. Grid resources are different from resources in conventional distributed computing systems by their dynamism, heterogeneity, and they are geographically scattered [1]. Therefore, scheduling of jobs to resources on computational grids represents a challenging problem addressed by many researchers in order to exploit the potentials of these resources in an effective and efficient manner [2, 3].

In grid computing, job scheduling can be done statically or dynamically. In static scheduling, each job is allocated a resource before its execution start and the job cannot stop or move to another resource after starting execution. On the other hand, dynamic scheduling allows changing the scheduling decision during the execution time. This means that jobs can be stopped or moved to be executed on another resource rather than the initially allocated resource [4].

K. Ranganathan and I. Foster [5] had stated that scheduling of jobs to resources of a grid is greatly affected by the grid dynamism which means that some resources can change their status at any time, especially after scheduling time, without advance notice. This is due to the lack of ownership of resources. Hence, static scheduling techniques that assume permanent availability and map out the entire schedule before executing jobs may not be suitable for computational grids. In grids, scheduling techniques have to work with the assumption that

the information used for mapping jobs to resources will quickly be out of date. These complications lead to a need of an effective and efficient scheduling strategy that considers grid frequent changes to be implemented.

The main contribution of this work is to introduce a strategy for scheduling jobs to resources on computational grids. When scheduling, the proposed strategy depends on the time at which status of grid resources is updated combined with resources response time. Through simulation, the proposed strategy is compared with a scheduling strategy that selects resources according to resources response time only.

This paper is organized as follows: section 2 briefly explains related work of scheduling jobs in computational grids. In Section 3, the problem is described and the scope is explained. Section 4 elaborates the proposed strategy. Section 5 describes the simulation environment. Section 6 augments results and discusses the performance of the system. Section 7 concludes the paper.

2. Related Work

The efficiency of job scheduling has a great and direct impact on the performance of the entire grid environment [6]. So, many research works related to the job scheduling problem have been designed and evaluated in the field of grid computing systems.

In [7], M. Mishra, R. Sharma, et. al. focused on grouping based job scheduling taking into account memory constraint, expected execution and transfer time at the job level rather than at group level in grid system. It supports dynamic grid environment and reduces processing and communication time.

N. Keat and A. Fong [8] use bandwidth-awareness concept in job scheduling to improve the performance. The concept of bandwidth-awareness is based on considering both computational and communication capabilities of the grid resources when taking scheduling decisions. It uses network bandwidth of resources to rank resources.

R. Chang, J. Chang and P. Lin [9] proposed an ant algorithm that uses the response time as the main criterion in the scheduling process. But, they neglected the time needed for returning results of jobs. K. Ruhana, K. Mahamud and H. Abdul Nasir [10] proposed an ant colony algorithm for job scheduling in computational grids. Their algorithm combined the techniques from ant colony system and max-min ant system. Ant algorithms are based on the concept of colony of ants that work together to find the shortest path between their nest and food resource. The algorithm considerd assigning jobs to resources that are suitable based on the resource processing ability as well as the characteristics of the job.

T. altameem and M. Amoon [11] proposed a scheduling system for computational grids that is a combination of a static scheduling and dynamic adjustment through using agents. The scheduling algorithm of their system used the response time as the main criteria for selecting suitable resources for each job.

Reviewing literatures reveals that all previous works are mainly based on using resource response time or grid bandwidth as the main criteria when taking scheduling decisions. There is no scheduling work done on the area of grid computing that considers the updating time of the status of grid resources. In this paper, a job scheduling strategy that depends on both the response time and the status updating time of resources is presented and evaluated. The strategy aims to reduce the overall processing time of applications by selecting the most recently updated resources to execute user application jobs.

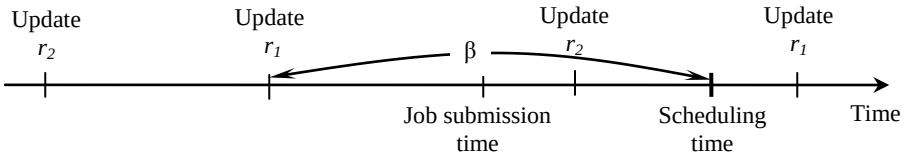

Figure 1. Example of Update Timing

3. Problem Formulation

In computational grids, the main function of the resources is to execute user jobs and the main objective of the scheduling is to greatly achieve the user requirements. Therefore, users submit their jobs to the grid along with their QoS requirements. These requirements may include the deadline in which users want their jobs to be executed within, the type of resources required to execute the jobs and the type of the platform needed. In current scheduling strategies, the scheduler of the grid selects the most suitable resource to execute the job. Selecting the most suitable resource is often based on the resource response time. After complete execution of the job, results are returned to the user. In computational grids, the status of the resources are periodically updated every a certain period determined by the system. If the selected resource is out of date at the time of scheduling, then more time will be consumed for processing the job than expected. This more time is due to job waiting if the resource was heavily loaded or rescheduling if the resource is moved to another place in the grid. Thus, the user's QoS requirements are not satisfied.

Figure 1 represents an example that shows the importance of the timing of updating status of resources. Assume there are two grid resources r_1 and r_2 that can execute job *i* and resource r_1 has response time that is less than response time of r_2. In time-based strategies, r_1 is selected to execute job *i*. If r_1 is moved to another place in the grid or overloaded by more jobs during the time period β then it will be out of date and more time be added if it is chosen for executing job *i*. This is because that the status of r_1 will be updated with its new state after the scheduling time and then the old status of r_1 will be used by the scheduler at the scheduling time. Thus, choosing r_1 may lead to more time when executing job *i*.

In order to address this problem, the time at which resources are updated is considered an important criterion when scheduling jobs. Using this criterion, we can avoid scheduling jobs on heavily loaded resources or on resources that have been moved to another place in the grid and their status is still not updated. In this way, we can improve the performance of the grid.

4. Proposed Job Scheduling Strategy

The aim of this work is to optimize the performance of the grid by trying to minimize the time spent in processing jobs on computational grids. In order to achieve this, the proposed scheduling strategy depends on both the response time and the status update time of resources.

The architecture of the grid system is shown in Figure 2. The system contains user interface (UI), job scheduler (GS), information server (IS) and grid resources. Users submit their jobs with job information and their QoS requirements to the grid through the UI. Job information include job number, job type, and job size. Also, users submit QoS requirements of each job such as the deadline to complete its execution, the number of required resources and the type of these resources. GS receives jobs with their information and QoS requirements from UI and allocates each job to the most suitable resource. The IS periodically

collects information about resources of the grid. The information include resource speed, current load and current place.

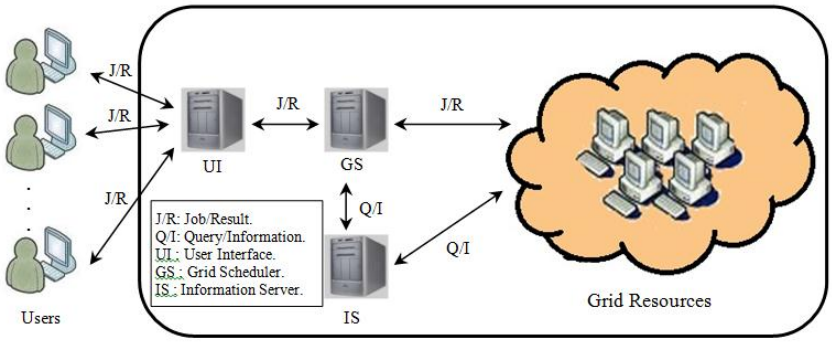

Figure 2. The Architecture of Grid System

Also, it calculates and stores the update factor of each resource. The latter is an indicator about the status update of resource information. Grid resources provide computing services to users.

The main function of GS is to find and sort the most suitable resources that can execute the job and satisfy user QoS requirements. In order to perform this function, GS connects to IS to get information of available resources that can execute the job. The GS uses response time and the update factor of each resource to construct the list of suitable resources that can execute the job. The update factor of a resource j is defined by:

$$\mu f_j = (\tau_c - \tau_{jlu})/(\tau_c);\ 0 < \mu f_j \leq 1 \quad (1)$$

where τ_c is the current time and τ_{jlu} is the last time at which information about resource *j* was updated. A small value of update factor means a recently updated resource and a large value of update factor means oldest updated resource.

Assuming there is a grid that has *m* resources and there are *n* jobs submitted by users. The GS constructs a matrix of m × n enteries, called scheduling matrix (SM). The SM matrix will be as follows:

$$SM = \begin{array}{c} \\ j_1 \\ . \\ . \\ . \\ j_n \end{array} \begin{array}{c} \begin{array}{cccc} r_1 & r_2 & \ldots & r_m \end{array} \\ \begin{bmatrix} SI_{11} & SI_{12} & \ldots & SI_{1m} \\ . & . & \ldots & . \\ . & . & \ldots & . \\ . & . & \ldots & . \\ SI_{n1} & SI_{n2} & \ldots & SI_{nm} \end{bmatrix} \end{array}$$

where SI_{ij} is the scheduling indicator of job *i* when scheduled on resource *j*. In this paper, scheduling indicator is a combination of response time and update factor of resources. The

response time of a resource is the summation of job transfer time, job waiting time, job execution time, and results transfer time. The response time of a resource j for a job *i* is defined by:

$$\tau_{ij} = \tau_{tij} + \tau_{eij} + \tau_{wij} + \tau_{rij} \quad (2)$$

where τ_{tij} is the transfer time of job *i* from the GS to the resource *j*, τ_{eij} is the time of executing job *i* on the resource *j*, τ_{wij} is the waiting time of job *i* in the queue of resource *j* and τ_{rij} is the time for transferring results of executing job *i* from resource *j* to GS.

Therefore, scheduling indicator for job *i* when scheduled on resource *j* can be defined by:

$$SI_{ij} = \tau_{ij} \times \mu f_j. \quad (3)$$

The scheduling indicator shows that when a job is assigned to a resource, the resource response time and the resource update status time are considered to select the suitable resource for the job. The smaller the value of SI_{ij} is, the more efficient for resource *j* to execute job *i*. After constructing the SM matrix, each row in the matrix is sorted in an ascending order according to the value of the scheduling indicator.

The sequence of interactions between components of the grid using the proposed strategy is shown in Figure 3 as follows:

1. At the start, grid resources register themselves with the IS.
2. GS receives jobs with user QoS requirements through UI.
3. GS sends a query to IS to get a list of available resources for each job.
4. IS responds to this query by sending a list of registered resources that are suitable for executing the job and their information to the GS.
5. After receiving the available list of resources, the GS performs the following:
    a. Computes the SI matrix.
    b. Sorts each row in the SI matrix in an ascending order according to the value of the scheduling indicator of each resource.
    c. Submits each job to the first resource in the resources row of each job.
6. If the resource moved then job is rescheduled again by the scheduler.
7. If the job is finished, results will be returned to the GS after executing each job.
8. GS returns results to the user.

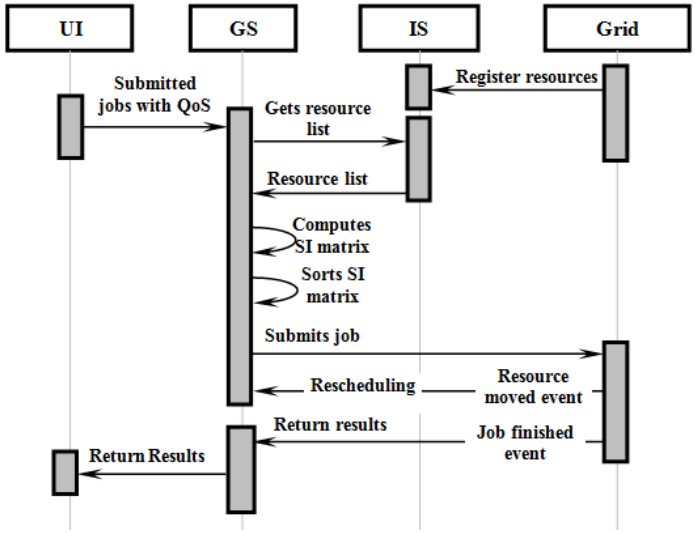

Figure 3. The Compements Interaction Sequence of the Proposed Strategy

5. Simulation Environment

Grid is a complex environment and the behavior of the resources in the grid is unpredictable. So, it is difficult to build a grid on a real scale to validate and evaluate scheduling systems. Therefore, simulation is often used. There is a number of well-known grid simulators, such as GridSim [12], SimGrid [13] and NSGrid [14]. However, most of these simulators have a limited modeling for dynamic grids [15]. So, in order to carry out this study, our implemented grid simulator in [16] is used in this paper.

The simulator supports modeling of grid resources and it enables the creation of application jobs and mapping of these jobs to resources in the grid. It provides the ability to deal with grid dynamism. It can allow us to evaluate the performance of scheduling strategies.

In our experiments, we modeled applications with sizes of 1000, 2000, 3000, 4000 and 5000 jobs. The size of each job is selected to be 200MB. The number of resources in the grid can reach up to 1000 resources. These specifications remain the same in all experiments.

6. Results

We have conducted different simulation experiments and have measured the number of jobs completed within the deadline. The proposed strategy is compared with a time-based scheduling strategy in [10] that depends on the response when selecting resources.

6.1 The Effect of the Update Time

In this section, the effect of the update time of resources status is studied. The periods of update time are chosen to be 1, 2, 3, 4 and 5 hours before the time at which the scheduling process will be done. The number of jobs submitted by the user into the grid is 1000 jobs.

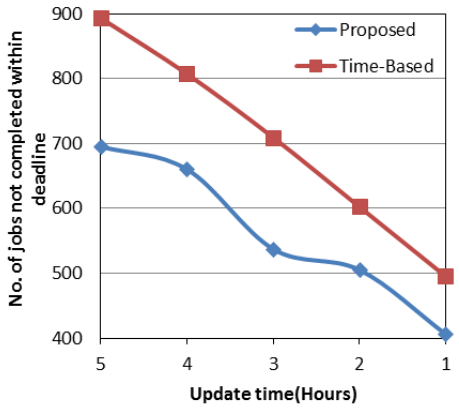

Figure 4. The Effect of the Update Time

Figure 4 shows the number of not completed jobs for different update periods. It is observed that for both strategies the number of not completed jobs is decreased when the update period becomes more close to the scheduling time. Also, it is shown that the number of not completed jobs of the proposed strategy is less than that of the time-based strategy during all update periods. This is because as the update period of a resource increases the resource will have a high probability to be moved or overloaded with more jobs. Also, as the update period of a resource decreases the resource will have a low probability to be moved or overloaded and then the number of not completed jobs will decrease as the updating period become more close to the scheduling time.

6.2 Makespan Comparison

Makespan is an important performance metric of scheduling strategies in computing systems. It can be defined as the maximum completion time of application jobs executed on grid resources. The less the makespan of a scheduling strategy the better it works [17, 18]. In this section, the proposed strategy is compared with the time-based strategy in terms of makespan.

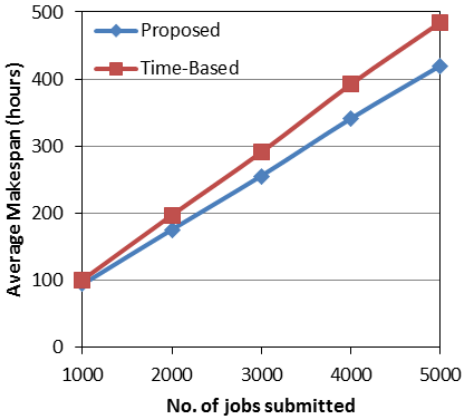

Figure 5. The Makespan Comparison

Figure 5 shows the comparison for different number of jobs submitted. In general, the makespan for both strategies increases as the number of submitted jobs increases. It is shown

that the proposed strategy has a lower makespan than the time-based strategy. This means that the proposed strategy consumes less time than the time-based strategy in processing user applications. This is due to that the proposed strategy tries to minimize the processing time of jobs by avoiding the selection of resources that have out of date status and considering the most recently updated resources. This can reduce the possible time needed for waiting and rescheduling of jobs in case of time-based strategy. Figure 6 shows the average more processing time needed for each job to be completed in both strategies. The more time is due to the dynamism of the grid. It is shown that the proposed strategy has a lower more time than the time-based strategy. For the proposed strategy, it is roughly 0.21 hour for each job while it is 0.24 hour for each job in case of time-based strategy.

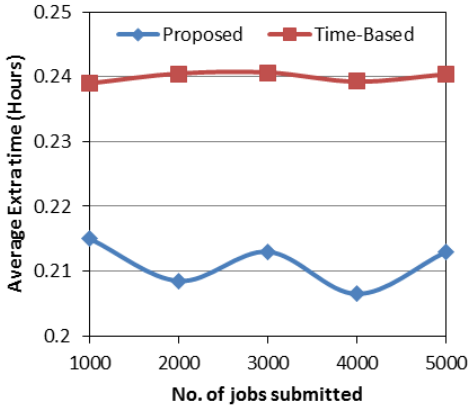

Figure 6. The Extra Time Comparison

6.3 The Number of Not Completed Jobs

This section compares the proposed strategy with the time-based strategy [10] in terms of the number of not completed jobs within the deadline assigned by the user for each job. Figure 7 shows the comparison for different number of jobs submitted. Generally, the number of not completed jobs within the deadline increases as the number of submitted jobs increases.

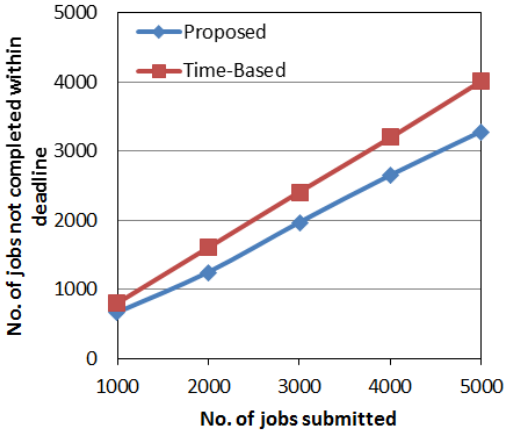

Figure 7. The Not Completed Jobs within Deadline Comparison

The figure shows that the number in case of the proposed strategy is better than in case of the time-based strategy. This is due to that a resource may have a lower response time but it may have status that is out of date. This resource may add more time to the job processing time in the grid and then the deadline of the job will be exceeded. The proposed strategy considers the most recently updated resources in addition to the response time when selecting resources. So, the effect of the outdated resources is alleviated. On the other hand, time-based strategy considers only the response time. So, some of the selected resources may have out of date status and after the last update, these resources may be becoming occupied by a large number of jobs to be executed in their waiting queues or may be moved to another location in the grid. Therefore, jobs assigned to these resources will not be completed in the expected deadline.

7. Conclusions

In this paper, a strategy for dynamic scheduling of jobs in computational grids is proposed and presented. The strategy selects resources for jobs depending on the update time of their status combined with their response time. The performance of the strategy is compared with a scheduling strategy in [10] that depends on the response time when scheduling jobs.

Experimental results show that the proposed strategy effectively schedules jobs in dynamic grids. It is observed that the performance of the proposed strategy outperforms the performance of the time-based strategy. This shows the effectiveness of considering most recently updated resources when scheduling jobs in computational grids.

References

[1] L. Lu and S. Yang, "DIRSS-G: An Intelligent Resource Scheduling System for Grid Environment Based on Dynamic Pricing", International Journal of Information Technology, vol. 12, no. 4, **(2006)**, pp. 120-127.
[2] D. Amalarethinam and F. Selvi, "A Minimum Makespan Grid Workflow Scheduling Algorithm", International Conference on Computer Communication and Informatics, **(2012)** January 10-12, pp. 1-6.
[3] L. Chunlin, Z. J. Xiu and L. Layuan, "Resource Scheduling with Conflicting Objectives in Grid Environments: Model and Evaluation", Journal of Network and Computer Applications, vol. 32, Issue 3, **(2009)**.
[4] M. Chtepen, "Dynamic scheduling in grids system", Sixth Firw PhD Symposium, Faculty of Engineering, Ghent University, **(2005)**, pp.110.
[5] K. Ranganathan and I. Foster, "Simulation Studies of Computation and Data Scheduling Algorithms for Data Grids", J. Grid Computing, vol. 1, **(2003)**, pp. 53–62.
[6] W. Meihong, "A Comparison of Four Popular Heuristics for Task Scheduling Problem in Computational Grid", 6th International Conference on Wireless Communications Networking and Mobile Computing, IEEE, **(2010)** September 23-25, pp. 1-4.
[7] M. Mishra, R. Sharma, V. K. Soni, B. R. Parida and R. K. Das, "A Memory-Aware Dynamic Job Scheduling Model in Grid Computing", International Conference on Computer Design and Applications, IEEE, vol. 1, **(2010)**.
[8] N. Keat and A. Fong, "Scheduling Framework for Bandwidth-Aware Job Grouping-Based Scheduling In Grid Computing", Malaysian Journal of Computer Science, vol. 19, No. 2, **(2006)**, pp. 117-126.
[9] R. Chang, J. Chang and P. Lin, "An Ant Algorithm for Balanced Job Scheduling in Grids", J. Future Generation Computer Systems, vol. 25, **(2009)**, pp. 20–27.
[10] K. Ruhana, K. Mahamud and H. Abdul Nasir, "Ant colony algorithm for job scheduling in grid computing", In proceeding of Fourth Asia International Conference on Mathematical/Analytical Modelling and Computer simulation, **(2010)** May 26-28, pp. 40-45.
[11] T. Altameem and M. Amoon, "An Agent-Based approach for dynamic adjustment of scheduled Jobs in Computational Grids", Journal of Computer and Systems Sciences International, vol. 49, no. 5, **(2010)** October, pp. 765–772.
[12] R. Buyya and M. Murshed, "GridSim: A Toolkit for the Modeling and Simulation of Distributed Resource Management and Scheduling for Grid Computing", J. Concurrency and Computation: Practice and Experience, vol. 14, nos. 13-15, Wiley, **(2002)** Nov.-Dec.

[13] A. Legrand, L. Marchal, and H. Casanova, "Scheduling Distributed Applications: The SimGrid Simulation Framework", Proc. Third Int'l Symp. Cluster Computing and the Grid (CCGrid '03), **(2003)** May.
[14] P. Thysebaert, B. Volckaert, F. De Turck, B. Dhoedt and P. Demeester, "Evaluation of Grid Scheduling Strategies through NSGrid: A Network-Aware Grid Simulator," J. Neural, Parallel and Scientific Computations, special issue on grid computing, vol. 12, no. 3, **(2004)**, pp. 353-378.
[15] M. Chtepen et. al., "Adaptive Task Checkpointing and Replication: Toward Efficient Fault-Tolerant Grids", IEEE Trans. Parallel and Distributed Systems, vol. 20, no. 2, **(2009)** February, pp. 180-190.
[16] M. Amoon, "A fault-tolerant scheduling system for computational grids," Journal of Computers and Electrical Engineering", vol. 38, Issue. 2, **(2012)**, pp. 399-412.
[17] K. Etminani and M. Naghibzadeh, "A Min-Min Max-Min Selective Algorihtm for Grid Task Scheduling", Proc. of the 3rd IEEE/IFIP International Conference in Central Asia, **(2007)** September 26-28, pp. 1-7.
[18] A. Chandak, B. Sahoo and A. Turuk, "An Overview of Task Scheduling and Performance Metrics in Grid Computing", Proc. of 2nd National Conference-Computing, Communication and Sensor Network, **(2011)** October 29-30, pp. 30-33.

Authors

M. Amoon

He received his B.S. in Electronic Engineering in 1996 and M.Sc. and PhD degrees in Computer Science and Engineering from Menofia University in 2001 and 2006, respectively. His research interest includes distributed computing, grid computing, Agent-based systems, and cloud computing.

H. M. Faheem

He received PhD degrees in Computer Science and Engineering from Ain Shams University in 2000. At the moment, he is a professor at the department of computer systems, Ain Shams University. His research interest includes grid computing, agent-based systems, network security, computer vision and pattern recognition.