A Fault Tolerant Scheduling System Based on Checkpointing for Computational Grids

Mohammed Amoon

Computer Science and Eng. Dept., Faculty of Elec. Eng., Menofia University, Egypt
King Saud University, P. O. Box: 28095-11437 Riyadh-Saudi Arabia
m_amoon74@yahoo.com

Abstract

Job checkpointing is one of the most common utilized techniques for providing fault tolerance in computational grids. The efficiency of checkpointing depends on the choice of the checkpoint interval. Inappropriate checkpointing interval can delay job execution. In this paper, a fault-tolerant job scheduling system based on checkpointing technique is presented and evaluated. When scheduling a job, the system uses both average failure time and failure rate of grid resources combined with resources response time to generate scheduling decisions. The system uses the failure rate of the assigned resources to calculate the checkpoint interval for each job. Extensive simulation experiments are conducted to quantify the performance of the proposed system. Experiments have shown that the proposed system can considerably improve throughput, turnaround time and failure tendency.

Keywords: *computational grids, checkpointing, scheduling, fault rate*

1. Introduction

Computational grids can be defined as an environment that organizes geographically distributed and heterogeneous resources in different administrative domains with different security polices into a single computing system. It enables users to use its resources for large-scale compute applications in science, engineering and commerce [1].

Since grid environments are extremely heterogeneous and dynamic, with components joining and leaving the system all the time, more faults are likely to occur in grid environments [2]. Also, the likelihood of errors occurrence is exacerbated by the fact that many grid applications will perform long tasks that may require several days of computation. This will lead to a number of new conceptual and technical challenges to fault-tolerance researchers. The most important one is the scheduling of user jobs to grid resources with meeting the user's Quality of Service (QoS) in existence of resource faults.

Fault tolerance is preserving the delivery of expected services despite the presence of fault-caused errors within the system itself. Errors are detected and corrected and permanent faults are located and removed while the system continues to deliver acceptable services [3]. In computational grids, fault tolerance is important as the dependability of grid resources may not be guaranteed. It is needed to enable the grid to continue its work when one or more resources fail. In this sense, a fault-tolerant service must be included to detect errors and recover from them and thus avoiding the failure of the grid.

The main contribution of this work is to introduce a fault-tolerant checkpointing-based system (FTCS) with a scheduling strategy. The proposed scheduling strategy depends on response time combined with both the failure rate (*FR*) and average failure time (*AFT*) of resources. FTCS uses the *FR* of resources to calculate the checkpoint interval for each job.

Through simulation, FTCS is compared with a scheduling system that depends on using resource fault index and response time.

This paper is organized as follows: Section 2 briefly explains related work for providing fault tolerance in computational grids. In Section 3, the problem is described and the scope is explained. Section 4 elaborates the proposed system. 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

Review of literature reveals that a large number of research efforts have already been devoted to tolerate faults in computational grids. Job replication and job checkpointing are the two often used techniques to accomplish fault tolerance in computational grids. Job replication is based on the assumption that the probability of a single resource failure is much higher than of a simultaneous failure of multiple resources. It avoids job recomputation by starting several copies of the same job on different resources. With redundant copies of a job, the grid can continue to provide a service in spite of failure of some grid resources carrying out job copies without affecting the performance [4]. Calculating the optimal number of job replicas represents the main challenge when using this technique.

Job checkpointing is the ability to save the state of a running job to a stable storage to reduce the fault recovery time. In case of fault, this saved state can be used to resume execution of the job from the point in computation where the check-point was last registered instead of restarting the application from its very beginning. This can reduce the execution time to a large extent. The efficiency of checkpointing mechanism is strongly dependent on the length of the checkpointing interval. The checkpointing interval is the duration between two checkpoints. Each interval starts when a checkpoint is established and ends when next checkpoint is established. A short checkpointing interval leads to a large number of redundant checkpoints, which delay job processing by consuming computational and network resources. On the other hand, when a checkpointing interval is too long, a substantial amount of work has to be redone in case of a resource failure. So, calculating the optimal length of a checkpointing interval represents the main challenge when using this checkpointing.

F. G. Khan, K. Qureshi and B. Nazir [5] presented a performance evaluation of most commonly used fault-tolerant techniques (FTTs) in grid computing. These FTTs include retrying, checkpointing, alternate resource and alternate task. The metrics used in evaluation are throughput, turnaround time, waiting time and network delay.

B. Nazir, K. Qureshi and F. G. Khan [6] presented an adaptive fault tolerant job scheduling strategy for grids called CFTGS. Their strategy is checkpointing-based. It maintains the fault index of grid resources. The scheduler makes scheduling decisions according to the value of the resources fault index and response time of resources.

In [7], M. Nandagopal and V. R. Uthariaraj combined the mechanism developed in [6] with Minimum Total Time to Release (MTTR) job scheduling algorithm. Also, when making scheduling decisions, their scheduler depends on using the fault index and the response time of resources.

J. Mehta and S. Chaudhary [8] assumed that short running jobs can be resubmitted from scratch if they failed and presented a fault tolerant scheme that should applied to long running jobs using checkpointing.

In [9], P. Domingues, J. Silva and L. Silva presented a study about the effects of sharing checkpoints on turnaround time in desktop grid systems. In [10], M. Chtepen et al provided an algorithm called MeanFailureCP. This algorithm is designed to modify a job checkpointing interval as a function of mean failure frequency of resources where the job is

being executed, and the total job execution time. In [11], they developed the MeanFailureCP+ algorithm which is a modification of the MeanFailureCP that deals with checkpointing of grid applications with execution times that are unknown a priori.

Reviewing literatures reveals that computational grids are failure prone. Also, all previous research works depend on using resource fault index when scheduling or when calculating the checkpoint interval and neglect both the failure rate and the average failure time of resources. So, there is a need of an efficient fault tolerant scheduling system to reduce the fault recovery time, reduce the grid overhead incurred and take into consideration the failure rate and the average failure time of resources.

3. Problem Definition and Scope

The main objective of computational grids is to execute the user applications or jobs. Therefore, users submit their jobs to the Grid Scheduler (GS) along with their QoS requirements. These requirements may include the deadline in which users want jobs to be executed, the type of the resources required to execute the job and the type of the platform needed. The GS of the present scheduling systems allocates each job to the most suitable resource. In case of fault free, results of executing the job are returned to the user after completion of the job. If the grid resource failed during execution of the job, the job is rescheduled on another resource which starts executing the job from scratch. This leads to more time consumed for the job than expected. Thus, the user's QoS requirements are not satisfied.

To address this problem, the job checkpointing mechanism is used. Using checkpointing, we can restore the partially completed job from the last checkpoint saved and then starting a job from scratch is avoided. The main disadvantage of checkpointing mechanism is that it performs identically regardless the stability of the resource. This inappropriate checkpointing can delay the job execution and can increase the grid load. Commonly utilized checkpointing mechanisms use resource fault index to determine checkpoint interval. In [12], it was proven that resource failure rate is more effective than resource fault index to represent the failure history of resources. So, the resource failure rate (*FR*) is used to determine the checkpoint interval and the number of checkpoints instead of using the resource fault index.

In computational grid environments, there are resources that satisfy QoS requirements but they tend to fail. The GS of the present scheduling systems [6, 7, 13, 14] select resources according to the response time combined with the resource fault index to execute the job. If the selected resource is failed and it is the only available resource that can execute the job at that time, the job must wait for that resource to join the system again and become available. This waiting time delays the job execution and reduces the throughput of the grid.

To address this problem, the average failure time (*AFT*) of the resource is taken into consideration when making scheduling decisions.

4. FTCS Scheduling

The aim of this work is to optimize the performance of the grid in the presence of faults. The performance metrics used include throughput, turnaround time and failure tendency. A fault occurs when a grid resource cannot complete its job within the given deadline [7]. The main strategy of the proposed FTCS depends on using the job checkpointing mechanism to minimize the effect of grid faults and to reduce the fault recovery time.

4.1. Components of the FTCS

The interaction between different components of the FTCS is shown in Figure 1. The FTCS restarts the execution of the failed job from the last saved checkpoint. Thus, it reduces the response time of the job by reducing the time wasted in re-executing partially completed job from the scratch.

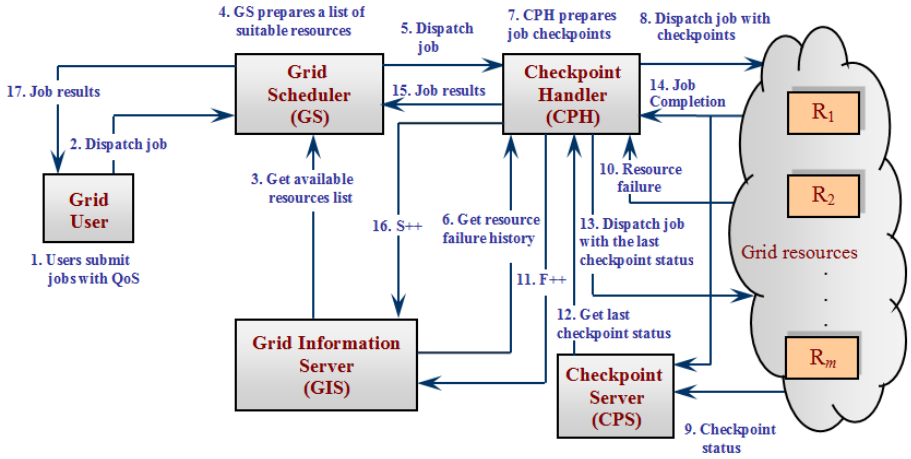

Figure 1. Architecture of the FTCS

A grid contains multiple grid resources that provide computing services to users. The main component of the FTCS is the grid scheduler (GS). It receives jobs with their information from users. Job information include job number, job type, and job size. Also, the user submits QoS requirements of each job such as the deadline to complete its execution, the number of required resources and the type of these resources.

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, the GS connects to the grid information server (GIS) to get information of available grid resources that can execute the job. Figure 2 shows the operation of the GS. The GS uses response time, resource failure rate and resource failure time to construct the list of suitable resources that can execute the job.

```
For each job j submitted by the user to the GS
{
  GS gets a list of suitable resources for j from GIS;
  GS sorts this list according to the response time*FR*AFT of each resource;
  GS dispatches the job and the list to the CPH;
}
```

Figure 2. Grid Scheduler Algorithm

GIS contains information of all available resources in the grid required by the GS. The information include resource speed, current load, resource failure rate and total failure time of each resource. The latter is the time the resource spent in the failure case before it is coming again to join grid and work properly.

Checkpoint server (CPS) receives and stores partially executed results of a job from the resource in intervals specified by the checkpoint handler (CPH). These intermediate results are called checkpoint status. For each job, there is only one record of checkpoint status. When

CPS receives a new checkpoint status it overwrites the old one. If CPS receives a job completion message from the resource it removes the record of such job.

CPH is an important component of FTCS. The main functions of CPH are determining the number of checkpoints and determining the checkpoints interval for each job. CPH receives a job with its assigned list of resources from GS. It connects to GIS to get information about the failure history of grid resources assigned to the job. Based on failure rate of the resource, the CPH determines the number of checkpoints and the checkpoint intervals for each job. Then, it submits the job to the first grid resource in the resources list. Figure 3 shows the algorithm used by CPH to calculate the number of checkpoints and the checkpoint interval for each job.

```
T_jre: The response time of job j by the resource r
FR_r: Failure rate of the resource r
CHN: The number of checkpoints
CHI: Checkpoint interval
For each job j assigned to a resource r
{
 CPH gets FR_r value from the GIS;  /* FR_r = F / ( S + F ) */
 CPH calculates the number of checkpoints CHN = ⌊FR_r * T_jre⌋;
 CPH calculates the checkpoint interval as CHI = T_jre /CHN;
}
```

Figure 3. Checkpoint Calculation Algorithm

```
F: The number of jobs did not complete due to resource failure
S: The total number of jobs completed by the resource
For each job j dispatched by the CPH
{
 IF (CPH receives a job completion message)
       Sends a message to GIS to increment S;
 IF (CPH receives a resource failure message)
       {
        Sends a message to GIS to increment F;
        Gets the last checkpoint status from CPS;
       IF (there is a checkpoint stored)
       CPH dispatches the not completed part of the job along with the checkpoint
              status to the second resource in the resources list;
       Else
       CPH dispatches the whole job along to the second resource in the resources
              list;
       }
}
```

Figure 4. GIS Algorithm

If CPH receives a job completion message for a certain job, it will notify the GIS to increase the number of successful times, *S*, of the resource. Then, it delivers results to the GS which in turn submits it to the user. On the other hand, if the CPH receives a resource failure message, it will notify the GIS to increase the number of failure times, *F*, of the resource. In this case, the CPH connects to the CPS to get the last checkpoint status of the job and resubmits it along with the job to the second resource in the resources list. Figure 4 shows decisions taken upon resource failure or job completion.

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 and fault tolerant systems. Therefore, simulation is often used. There is a number of well-known grid simulators, such as GridSim [1], SimGrid [15] and NSGrid [16]. However,

none of these simulators support the development of fault-tolerant scheduling algorithms because they have a limited modeling for dynamic grids [10, 17]. So, in order to carry out this study, we have used our implemented grid simulator [12].

The simulator supports modeling and simulation of grid resources and user applications. It enables the creation of application jobs and mapping of these jobs to resources in the grid.

In experiments, we modeled applications with size of 1000 jobs. The size of each job is selected to be 200MB. The number of resources in the grid can reach up to 1000. The percentage of faults injected is from 10% to 50%. These specifications remain the same in all experiments of measuring performance.

We have conducted different simulation experiments with variation in the total number of faults injected in the grid and measuring the throughput, turnaround time and the tendency of resources to fail. The proposed system is compared with a recent checkpointing-based scheduling system called CFTGS [6] that depends on the response time and resource fault index when scheduling jobs and uses the fault index of each resource when determining the checkpoint intervals and the number of checkpoints for each job.

6. Results and Discussions

6.1. Throughput

Throughput is one of the most important standard metrics used to measure the performance of fault tolerant systems [5]. Throughput is defined as:

$$Throughput(n) = \frac{n}{T_n}$$

where n is the total number of jobs submitted and T_n is the total amount of time necessary to complete n jobs. Throughput is used to measure the ability of the grid to accommodate jobs.

Figure 5 shows the throughput comparison of the proposed FTCS with the system in [6], named CFTGS. The comparison is done for different percentages of faults injected in the grid.

Generally, the throughput of the two systems decreases with the increase in the percentage of faults injected in the grid. This is due that the extra delay encountered by both of them to complete jobs in case of some resources failure. Figure 5 shows that the throughput of the proposed system is better than the throughput of the CFTGS. This is due to that in the proposed system both *AFT* and *FR* of grid resources are taken into consideration along with the response time during the scheduling step. On the other hand, the CFTGS considers the response time and only the fault index as the failure history of resources.

CFTGS Neglects the *FR* of resources and this leads to more waiting time when executing job on a resource and this resource is crashed. If it is the only available resource that can execute the job at that time, the job must wait for that resource to join the system again and become available. Thus, the waiting time will increase the response time of the job and then the throughput will decrease.

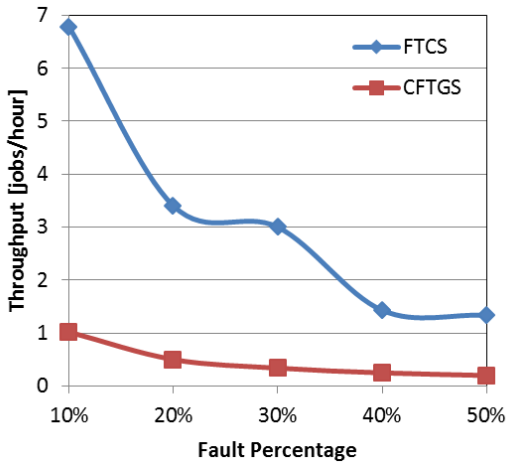

Figure 5. Throughput Comparison for Different Number of Faults Injected

Also, using *FR* by FTCS rather than using fault index during the scheduling process can lead to selecting resources that have lower probability to fail than resources selected by the CFTGS. This is because a resource can have a small fault index but it can have a high *FR* [12]. In case failures occur, this high *FR* can lead to a wasting time. This will lead to increasing the response time and then the throughput will decrease.

6.2. Average Turnaround Time

It is the most important criterion of any computational system. Turnaround is an important parameter for determining the performance of different FTTs. It is the only parameter users pay attention to. It can be defined as the interval between job submission time and job completion time. This parameter is greatly affected by the failure history of resources selected during the scheduling process and also by the number of checkpoints produced for each job. Figure 6 shows the turnaround time comparison between the proposed FTCS, and CFTGS. In general, the average turnaround time of both systems increases with increase in the percentage of faults injected. From the figure, it is shown that the turnaround time of the proposed system is better than that of the CFTGS system. This is because FTCS selects resources that have lower probability to fail than resources selected by CFTGS and this will reduce the waiting time in case of resources failure.

6.3. Failure Tendency

This metric is firstly introduced in [12]. It is the percentage of the tendency of the selected grid resources to fail and is defined as:

$$FailTendency = \frac{\sum_{j=1}^{m} p_{fj}}{m} \times 100\%$$

where m is the total number of grid resources and P_{fj} is the failure rate of resource j. Through this metric, the faulty behavior of the system can be expected.

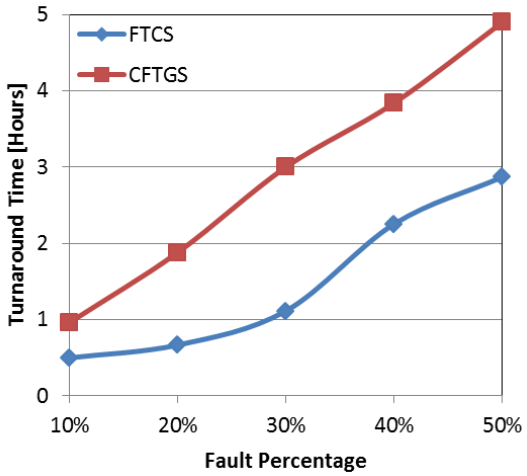

Figure 6.Turnaround Time Comparison for Different Number of Faults Injected

FailTendency metric indicates the failure prediction of the grid resources when applying a scheduling system. It reflects the extent of reliability of the grid. The value of it is greatly affected by the algorithm used during the scheduling step. The FailTendency of FTCS is compared with the FailTendency of the CFTGS. The comparison is depicted in Figure 7.

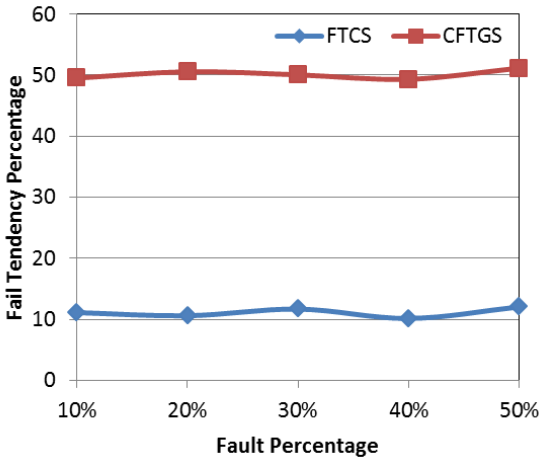

Figure 7. FailTendency Comparison for Different Number of Faults Injected

The figure shows that the proposed FTCS has a lower FailTendency than the CFTGS. In CFTGS system, it is approximately around 50% while in the proposed system it is around 10%. This is due that during the scheduling step the proposed system considers both *FR* and average *AFT* of resources are representing the failure history of resources. On the other hand, the CFTGS uses the fault index of resources only. So, the proposed system will select resources with better failure history than resources selected by the CFTGS. Therefore, the proposed FTCS has a better tendency to fail than the CFTGS.

7. Conclusions

In this paper, a checkpointing-based scheduling system for grids is proposed and presented. The proposed system depends on average failure time and failure rate of resources combined with response time when taking scheduling decisions. The checkpoint interval is calculated using resource failure rate. The performance of the system is compared with the CFTGS scheduling system that depends on the response time and the fault index of resources when scheduling resources to execute jobs and it uses the resource fault index when calculating checkpoint interval. The metrics used for evaluation are throughput, turnaround time and failure tendency.

Experimental results show that FTCS effectively schedules jobs in the presence of failures. It is observed that the throughput for the proposed system is better than CFTGS. Also, the FTCS improves the turnaround time when compared with the CFTGS. Moreover, the failure tendency for the proposed FTCS is far better than the CFTGS. Thus, it can be concluded that the proposed scheduling system provides performance superiority over the CFTGS. This shows the effectiveness of considering resource failure rate and resource failure time over considering the resource fault index

References

[1] 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, no. 13-15, **(2002)**, pp. 1175-1220.
[2] S. S. Sathya and K. S. Babu, "Survey of Fault Tolerant Techniques for Grid", Computer Science Review, vol. 4, Issue 2, **(2010)**, pp. 101-120.
[3] A. Avizienis, "The N-version Approach to Fault-Tolerant Software", IEEE Trans. Software Engineering, vol. 11, no. 12, **(1985)**, pp. 1491-1501.
[4] M. Chtepen, F. Claeys, B. Dhoedt, F. Turck, P. Vanrolleghem and P. Demeester, "Providing fault-tolerance in unreliable grid systems through adaptive checkpointing and replication", Proc. of Intl. Conf. on Computational Science 2007 (ICCS2007), Beijing, China, **(2007)**, pp. 454-461.
[5] F. G. Khan, K. Qureshi and B. Nazir, "Performance Evolution of Fault Tolerance techniques in Grid Computing System", Journal of Computers and Electrical Engineering, vol. 36, Issue 6, **(2010)**, pp. 1110-1122.
[6] B. Nazir, K. Qureshi and F. G. Khan, "Adaptive checkpointing strategy to tolerate faults in economy based grid", Journal of Supercomputing, vol. 50, **(2009)**, pp. 1-18.
[7] M. Nandagopal and V. R. Uthariaraj, "Fault Tolerant Scheduling Strategy for Computational Grid Environment", International Journal of Engineering Science and Technology, vol. 2, no. 9, **(2010)**, pp. 4361-4372.
[8] J. Mehta and S. Chaudhary, "Checkpointing and recovery mechanism in grid", Proc. of Sixteenth Intl. Conf. on Advanced Computing and Communication (ADCOM 2008), Chennai, **(2008)**, pp. 131-140.
[9] P. Domingues, J. Silva and L. Silva, "Sharing Checkpoints to Improve Turnaround Time in Desktop Grid Computing", Proc. of the 20th Intl. Conf. on Advanced Information Networking and Applications (AINA'06), Vienna, Austria, **(2006)**, pp. 301-306.
[10] 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)**, pp. 180-190.
[11] M. Chtepen, F. Claeys, B. Dhoedt, F. Turck, P. Demeester and P. Vanrolleghem, "Adaptive checkpointing in dynamic grids for uncertain job durations", Proc. of the 31st Intl. Conf. on Information Technology Interfaces (ITI), Dubrovnik, Croatia, **(2009)**, pp. 585-590.
[12] M. Amoon, "A fault-tolerant scheduling system for computational grids", Journal of Computers and Electrical Engineering, vol. 38, Issue 2, **(2012)**, pp. 399-412.
[13] S. Therasa, G. Sumathi and S. Dalya, "Dynamic Adaptation of Checkpoints and Rescheduling in Grid Computing", International Journal of Computer Applications, vol. 2, no. 3, **(2010)**, pp. 95-99.

[14] L. M. Khanli, M. E. Far and A. M. Rahmani, "RFOH: A New Fault Tolerant Job Scheduler in Grid Computing", Proc. of the 2nd Intl. Conf. on Computer Engineering and Applications, Bali Island, Indonesia, **(2010)**, pp. 422-425.
[15] 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)**, pp. 138-145.
[16] 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.
[17] M. Chtepen, B. Dhoedt, F. Cleays and P. Vanrolleghem, "Evaluation of replication and rescheduling heuristics for gird systems with varying resource availability," Proc. of 18th International Conference on Parallel and Distributed Computing Systems, Anaheim, CA, USA, **(2006)**, pp. 622-627.

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.