Overview of Scheduling Tasks in Distributed Computing Systems

O. M. Elzeki, M. Z. Rashad, M. A. Elsoud

Abstract*: Distributed System is large scale computing environment that includes many subscribed resources to perform tasks more rapidly, stability, accuracy and availability. Nowadays, grid computing and cloud computing are widely common available distributed environment. In these computing there is many tasks requires to be executed by the available resources to achieve best performance, minimal total time for completion, shortest response time, utilization of resource usage and etc. Because of these different intentions and high performance of computing environment, we need to design, develop, propose a scheduling algorithm to outperform appropriate allocation map of tasks due to different factors. In this paper we present a package of reviews based on different factors which affect scheduling process such as communication cost and execution time. Here we have studied many different homogenous algorithms and state of art algorithms that can be applied in grid, cloud or both. These algorithms have different perspectives, working principles, domains and others.

***Keywords*: Cloud, Distributed, Grid, Scheduling.**

I. INTRODUCTION

Distributed Systems are very powerful and helpful computer systems that are known to solve tasks and problems in a feasible and fast way. A distributed system can be defined as a computing system in which services are provided by a pool of loosely coupled computers collaborating over a network working for a common goal [1]. Grid computing systems are distributed system that aim to utilize ideal CPU cycles and storage of millions of computer systems across a worldwide network. Grid computing enables sharing power of computer and data storage capacity over the internet [1]. Cloud computing is a parallel distributed system that consists of large scale of interconnected and virtualized computers based on service-level agreements which is established through negotiation between service provider and consumers [18].

It is clear from definition that traditional distributed computing is a subset of distributed computing either as Grid or Cloud. In grid computing concerned with how to efficiently module pool of heterogeneous systems with optimal workload management which is known as distributed computing. Besides; grid computing focuses on ability of supporting computation across multiple administrative domains. On the other hand, cloud, is concerned with the ability of dynamically providing the amount of required resources for satisfying various applications demands either scaling up or down. This scaling can be predictable due to access pattern observed during day and night occurrence or unexpected due to subtle increase in the popularity of application services. This feature of clouds is useful for elastic applications, such as web hosting, social network and etc [19]. This fact contributes to setting grid and cloud computing apart from traditional distributed computing [1].

The basic idea of distributed computing is paralleling computational requirements of large number of current and rising applications so that it has an evolutionary aspect about job scheduling and task scheduling in available runtime environment.

Georage Amalarethinam, and Muthulakshmi have defined Scheduling as the way processors are assigned to run on the available resources [2]. They have also defined job scheduling as a process of establishing queue to run a sequence of programs over a period of time [2]. Besides, they have defined task scheduling as the mapping of tasks to a selected group of resources which may be distributed in multiple administration domains [2]. So scheduling is a decision making process about assigning which task will be executed by which resource.

This paper discusses issues associated with developing job scheduler (JS) that behaves highly dynamic rather than static on its classified categories.

II. BACKGROUND

Grid is a technology that manages two contrasting services sharing and trust. Let’s assume there is a virtual organization as a group of individuals or institutions defined by a set of shared rules; then grid computing can be considered to coordinate resource sharing and problem solving in dynamic, multi-institutional virtual organizations, according to Robert [3]. Briefly there are many functional and non-functional challenges face grid computing such as trust, sharing applications and data, interpretability as functional, plus reliability and robustness, Quality of Service (QoS) as non-functional and others. Whereas Cloud is a platform that can support elastic applications in order to manage limited virtual machines and computing servers to application services at a given instance of time. Hence, scaling application's capacity becomes complicated, at which, number of requests overshoots the cloud's capacity. Exactly, there are many challenges associated with cloud computing such as monitoring,

Manuscript Received on June 29, 2012

O. M. Elzeki, Computer Science, Mansoura University, Mansoura, Egypt.
M. Z. Reshad, Computer Science, Mansoura University, Mansoura, Egypt.
M. A. Elsoud, Computer Science, Mansoura University, Mansoura, Egypt.

virtual machine deployment, load balancing, QoS and others.

A. Grid Organization

Grid computing architecture or organization is described by an hour-glass as shown in Fig 1. The thin center represents few standards; whereas the wide top represents many high-level behaviors that can be allocated; whereas the wide bottom represents underlying technologies and systems [3].

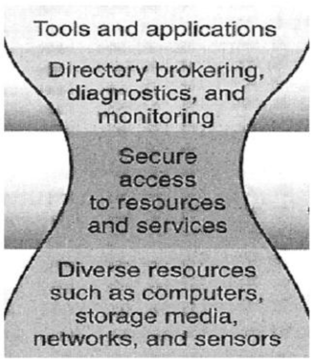
Tools and applications
Directory brokering, diagnostics, and monitoring
Secure access to resources and services
Diverse resources such as computers, storage media, networks, and sensors

Fig 1 Hour-glass [3]

The organization of grid as shown in Fig 2 is the infrastructure which involves [2], [3]:

- Fabric / Foundation Layer, that is responsible for interface local control between physical resources such as computers, storage media, network and sensors.
- Middleware /Resources and Connectivity Protocols, working on providing secure access to resources and services.
- Collective Services, helpful for diagnosing and monitoring of coordinated shared multiple resources.
- User Applications, sits for business solutions, operational support or other decision.

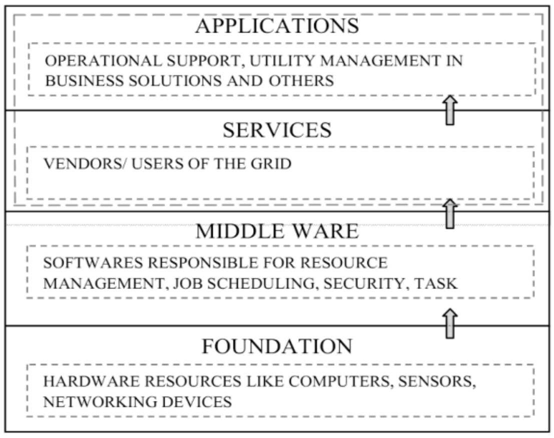
APPLICATIONS
OPERATIONAL SUPPORT, UTILITY MANAGEMENT IN BUSINESS SOLUTIONS AND OTHERS
SERVICES
VENDORS/ USERS OF THE GRID
MIDDLE WARE
SOFTWARES RESPONSIBLE FOR RESOURCE MANAGEMENT, JOB SCHEDULING, SECURITY, TASK
FOUNDATION
HARDWARE RESOURCES LIKE COMPUTERS, SENSORS, NETWORKING DEVICES

Fig 2 Organization of Grid [2]

B. Types of Grids

Different types of grids are designed and developed depending on many factors including structure of the organization, available resources for gird usage, services powered and provided by grid… etc.

Officially there are many grids developed for different purposes [2]. Data Grids, for example are tuned for data intensity and oriented for data operations. Compute Grids like Smart Grid works only for providing access usage to computational resources. Global Grids like White Rose Grid are available over internet; Utility/Resources Grids provide access to resources. Departmental Grids like Folding@Home are concerned with specific domain problems solving. Extraprise Grids like Amazon are interconnecting companies, customers, etc.

C. Cloud Organization

In Fig 3, Cloud computing architecture is presented as layered model. Cloud layers are logically divided into three layers, Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS) respectively from bottom to the top. From Fig 3, physical cloud resources (System Level) and middleware capabilities form the basis provider of delivering IaaS and PaaS in the form of a collection of transparently data centers and runtime environment for hosting and managing user-level application services. While user-level provides the programming environment and composition tools which ease the creation, deployment, and execution process of applications in the cloud. Finally, Cloud Application includes the applications available directly to the end users consuming SaaS services based on subscription model or pay-per-use basis [19].

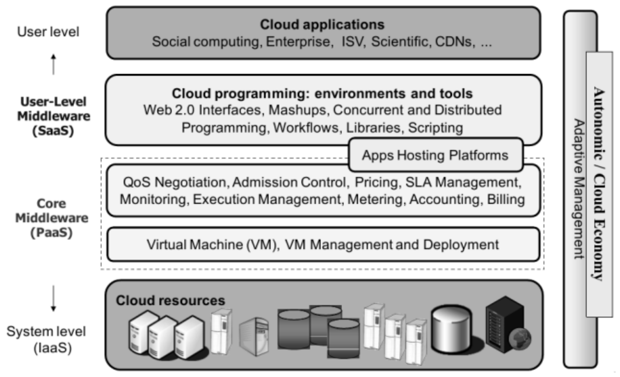
User level
Cloud applications
Social computing, Enterprise, ISV, Scientific, CDNs, ...
User-Level Middleware (SaaS)
Cloud programming: environments and tools
Web 2.0 Interfaces, Mashups, Concurrent and Distributed Programming, Workflows, Libraries, Scripting
Apps Hosting Platforms
Core Middleware (PaaS)
QoS Negotiation, Admission Control, Pricing, SLA Management, Monitoring, Execution Management, Metering, Accounting, Billing
Virtual Machine (VM), VM Management and Deployment
System level (IaaS)
Cloud resources
Autonomic / Cloud Economy
Adaptive Management

Fig 3 Cloud Layered Organization [19]

D. Types of Clouds

A Cloud can be public, private, community or hybrid cloud. For public cloud, the infrastructure of cloud is available for general public or a large industry group. Public cloud always is owned by cloud services seller. Where, private operates for a single organization. However, Community Cloud is shared by several organizations and supports a specific community that has shared concerns. It may be managed by the organization or a third party organization. Last type, Hybrid, is a cloud whose infrastructure is a composition of two or more clouds private, community, or public. Hybrid computing is bound together by standardized technology which enables data and application portability [31].

III. GUIDELINES OF SCHEDULING

Different types of scheduling are based on different criteria, such as static vs. dynamic environment, centralized vs. distributed etc. Different distributed computing scheduling criteria are listed as follows [4], [5] and [6], although they could be overlapped and not clearly distinct of each other:

1) ***Static Scheduling***: pre-schedule jobs, all information about available resources and tasks in application must be known and further more a task is assigned once to a resource, so that it's easier to adapt based on scheduler's perspective [4].
2) ***Dynamic Scheduling***: it is more flexible than static scheduling where jobs are dynamically available for scheduling over time by the scheduler with no issues, to be able of determining run time in advance. It is highly critical to include load balance as main factor to obtain stable and efficient scheduler algorithm [4].
3) ***Centralized Scheduling***: as mentioned in dynamic scheduling, a centralized / distributed scheduler is responsible for making global decision. By using centralized scheduling; ease of implementation, efficiency and more control and monitoring on resources are gained benefits. On the other hand; such scheduler lacks scalability, fault tolerance and efficient performance so it's not recommended for large-scale grids [4].
4) ***Distributed / Decentralized Scheduling***: such type of scheduling is more realistic for real grids despite of its weak efficiency compared to centralized scheduling, in which there are local schedulers' requests to manage and maintain state of jobs queue as there is no more central control entity [5].
5) ***Co-operative Scheduling***: in this case, system have already many schedulers, each one is responsible for performing certain activity in scheduling process towards common system wide range based on the cooperation of procedures, given rules and current system users [4].
6) ***Preemptive Scheduling***: This scheduling criterion allows each job to be interrupted during execution and a job can be migrated to another resource leaving its originally allocated resource unused to be available for other jobs. It is more helpful if there are constraints as priority to be considered [6].
7) ***Non Preemptive Scheduling***: in which resources aren't being allowed to be re-allocated until the running and scheduled job finished its execution [6].
8) ***Immediate/ Online Mode***: in which scheduler schedules any recently arriving job as soon as it arrives with no waiting for next time interval on available resources at that moment [6].
9) ***Batch/ Offline Mode:*** the scheduler holds arriving jobs as group of problems to be solved over successive time intervals, so that it is better to map a job for suitable resources depending on its characteristics [6].

IV. COMMON OF SCHEDULING IN DISTRIBUTED SYSTEM

A. First Come First Serve

It is an abstracted way of organizing and allocating of resources to jobs over time, it serves as a principle of a queue processing or demands' servicing by ordering that means what comes in first is allocated first, what comes in next waits until the first is finished [7], U. Schwiegelshohn and Yahyapour present (FCFS) through parallel processing aiming at tuning resource allocation time with the selected task from the incoming tasks [8]. This is known as Opportunistic Load Balancing (OLB) [9] or myopic algorithm. OLB works on assigning each task in a queue, in arbitrary order, to the next expected resource to be available, irrespective of the task's expected execution time using resources [2], [10], [11] and [12]. It is obviously clear that (OLB) works on keeping all machines as busy as possible.

B. Round Robin

Round-robin (RR) is a simple scheduling algorithm, based on time sharing among jobs in equal slice / quantum and in circular queue without priority so it is simple and easy to implement, but it has a starvation problem [13]. So it focuses on fairness between jobs, as Ruay-Shinung Chang claimed [14]. The advantage of RR is that no job has to wait for another one to be completed as FCFS and others. However, this algorithm is not a good choice for jobs that are largely varies in their size and requirements, by means, a job is never been satisfied which in turn leads to starvation or indefinite blocking.

C. Minimum Completion Time

R. F. Freund et al. proposed Minimum Completion Time (MCT) [11], in which allocates tasks arbitrary to be executed on a resource with minimum completion time [10]. Based on these criteria of scheduling causes some tasks to be assigned for resources that haven't the minimum execution time in turn poorly consume some resources.

D. Minimum Execution Time

In contrast to Opportunistic Load Balance (OLB), R. F. Freund et al., investigate the Minimum Execution Time (MET) algorithm. MET based on choose the task to be run on a resource having minimum execution time regardless of the resource availability. MET is not applicable in high computing environments because of its load imbalance across machines [11] and [20].

E. Min-Min

Min-Min scheduling is based on Minimum Completion Time (MCT); that is used to arbitrary assigning tasks to resources having minimum expected completion time for the task. Initially a scheduler takes a set of unmapped/ unscheduled tasks and a set of available resources starting by task with minimum MCT to be mapped to next resources; this process is repeated after removing job from the map until the unmapped set becomes empty [10]. Briefly, expectation time produces a smaller makespan, which is a measure of throughput of heterogeneous computing systems like computational grids [15] and [16]. If more tasks can be obtained, they can be mapped to resources to complete them earliest and executing them faster.

In Fig 4, the expected time of resource R_j is the time to become ready to execute a task after finishing the execution of all tasks assigned to it which is denoted by r_j. Also E_{ij} is the estimated execution time of task T_i on resource R_j whereas C_{ij} is the Expected Completion Time that is the estimated execution time and ready time together [17].

```
1. for all tasks $T_i$ in meta-task $M_v$
2.         for all resources $R_j$
3.         $C_{ij} = E_{ij} + r_j$
4. do until all tasks in $M_v$ are mapped
5. for each task in $M_v$ find the earliest completion
      time and the resource that obtains it.
6.         find the task $T_k$ with the minimum earliest
      completion time.
7.         assign task $T_k$ to resource  that gives the
      earliest completion time.
8.         delete task $T_k$ from $M_v$ .
9.         update $r_j$
10.        update $C_{ij}$ for all i.
11. end do.
```

Fig 4 Min-Min Algorithm [2]

F. Max-Min

Similar to Min-min, a scheduler schedules tasks by expecting the Execution Time of the tasks and allocation of resources. Instead of selecting the minimum MCT, the maximum MCT is selected, that is why it is named Max-min. It focuses on giving priority to large tasks over others small. The Max-min algorithm is typical to the min-min algorithm, except for being different in line (6) of Fig 4; the word "minimum" would be replaced by "maximum" [15]. Officially Max-min algorithm does better than Min-min algorithm in cases when the number of short tasks is much more than the long ones. For example, if there is only one long task, the Max-min algorithm executes many short tasks concurrently with the long one [17].

In order to avoid the main drawbacks of the Max-min and Min-min, the two schedulers can be executed alternatively to each other for assigning tasks to appropriate resource, eliminating each other drawback. Such methodology, called Resource Awareness Scheduling Algorithm (RASA) which was a new grid task scheduling algorithm [17].

G. RASA

This algorithm was proposed by Saeed Parsa and Reza Entezari-Maleki [17]. Fig 5 illustrates RASA by applying alternatively Max-min and Min-min over scheduling process rounds/ iterations. For example, if the first task is assigned to a resource by Min-min strategy, in the next round the task will be assigned by Min-min and so on.

Based on experimental results, if the number of available resources in grid system is odd it is highly preferred to start by Min-min algorithm in first round otherwise recommended starting by Max-min algorithm [17]. For next rounds just assign resources to task using a strategy different from last round ignoring waiting time of the small tasks in Max-min algorithm and the waiting time of the large tasks in Min-min algorithm.

RASA has no time consuming; it has time complexity like Max-min and Min-min, $O\ (mn^2)$ where m is the total number of resources and n is the number of tasks.

```
1. for all tasks $T_i$ in meta-task $M_v$
2.       for all resources $R_j$
3.          $C_{ij} = E_{ij} + r_j$
4. do until all tasks $M_v$ in  are mapped
5.    if the number of resources is even then
6.       for each task in $M_v$ find the earliest
      completion time and the resource that obtains it.
7.          find the task $T_k$ with the maximum earliest
      completion time.
8.          assign task $T_k$ to resource  that gives the
      earliest completion time.
9.          delete task $T_k$ from .
10.         update $r_j$
11.         update $C_{ij}$ for all i.
12. else
13.         for each task in $M_v$ find the earliest
      completion time and the resource that obtains it.
14.            find the task $T_k$ with the minimum
      earliest completion time.
15.            assign task $T_k$ to resource  that gives
      the earliest completion time.
16.            delete task $T_k$ from .
17.         update $r_j$
18.         update $C_{ij}$ for all i.
19. end do.
```

Fig 5 RASA Algorithm [2]

V. RECENT WORK OF SCHEDULING IN DISTRIBUTED COMPUTING

He. X, X-He Sun, and Laszewski. G.V [21], proposed QoS guided Min-min as new algorithm based on original Min-min. It schedules tasks with high bandwidth requirements before the others. Experiments show that QoS guided Min-min is better than original Min-min when submitted tasks varies highly in their bandwidth otherwise they have the same outperform.

Kamalam et. al. [22], designs a new algorithm called Min-mean heuristic scheduling for static meta-tasks. The proposed algorithm applies original Min-min then estimates the mean makespan of all the resources finally reschedules tasks. While submitted tasks are heterogeneity increase, the algorithm performs better than the Min-min.

Singh. M and Suri. P.K present two heuristics algorithms called QoS Guided Weighted Mean Time-Min (QWMTM) and QoS Guided Weighted Mean Time Min-min Max-min Selective (QWMTS) [23]. These algorithms schedule independent batch tasks and considering network bandwidth as QoS parameter.

Sameer Singh Chauhan,R. Joshi. C proposed algorithm that is called QoS based predictive Max-min, Min-min Switcher for scheduling jobs [24]. Based on heuristics, appropriate selection among QoS based Min-min and QoS based Max-min is made. The algorithm uses historical information about the execution time of jobs to predict the performance.

T. Kokilavani and Dr. D.I. George Amalarethinam introduce Load Balanced Min-min algorithm (LBMM) to schedule meta-tasks [25]. LBMM executes Min-min algorithm for the first round. Then, LBMM identifies the resources with heavy load by choosing the resource with high makespan in the schedule produced by Min-min algorithm.

Hai Zhong, Kun Tao, Xuejie Zhang proposed an optimized resource scheduling algorithm that focuses on achieving the optimization or partial optimization for cloud scheduling outperforms [26]. This algorithm uses an improved genetic algorithm for automation scheduling policy to improve utilization rate of resources and speed.

Y. Yang, K. Liu, J. Chen, X. Liu, D. Yuan and H. Jin introduce an improved cost-based algorithm to improve the computation / communication rate in cloud computing [27]. This algorithm makes an efficient mapping of tasks through measurement of resource cost and computation performance.

A compromised time cost scheduling algorithm is presented by Suraj Pandey, LinlinWu, Siddeswara Mayura Guru and Rajkumar Buyya [28]. It considers the set of cloud computing characteristics for accommodating instance intensive cost constraint by compromising both execution time and cost with user input enabled.

A version of Particle Swarm Optimization (PSO) is presented by Cui Lin and Shiyong Lu [29], which based on heuristics to schedule tasks in cloud environment. It uses computation cost and data transmission cost for scheduling application workflow.

Heterogeneous-Earliest-Finish-Time algorithm (HEFT) is presented by W. Chen, J. Zhang, [30]. This algorithm first calculates average execution time of each task and average communication time between resources of two successive tasks. Next tasks are ordered due to non-increasing a rank function. Higher priority is assigned for the task with higher rank value while resource selection phase tasks are scheduled due to their order based on their priorities and each task is assigned to the resource that can complete the task at the earliest time.

Next, in Table 1, we compare between listed algorithms as possible as based on different aspect of scheduling policies such as complexity; that represents cost or time consumption, scalability; that considers if tasks can increase or not during scheduling process, reliability which marks algorithm to be stable and applicable, availability as marker of algorithm existence in working environment and dependency between tasks. Each cell has a mark (√); means algorithm satisfy feature or (χ); means feature is missing for it.

Table 1: Comparison of listed scheduling algorithms based on different factors

Algorithm			Complexity		Dependency		Focus on			Scalability		Reliability	Availability	Recommended In
Name	**Method**	**Factor**	**Cost**	**Time**	**Dependent**	**Independent**	**Makespan**	**Resource utilization**	**Speed**	**Dynamic**	**Static**			
QoS guided Min-min [21]	Batch Mode	Bandwidth of Tasks	χ	χ	χ	√	√	χ	χ	χ	√	χ	χ	Grid
Min-Mean heuritics [22]	Batch Mode	----	χ	χ	χ	χ	√	χ	χ	χ	√	χ	χ	Grid
QoS guided weighted Mean Time-min [23]	Batch Mode	QoS and Network Bandwidth	χ	χ	χ	√	χ	χ	χ	χ	√	χ	χ	Grid
Qos based predictive Max-min, Min-min switcher [24]	Batch Mode	Hueritics	χ	χ	χ	√	√	χ	√	χ	√	χ	χ	Grid
Load Balanced Min-min [25]	Batch Mode	List of heavy load resources	χ	χ	χ	√	√	√	χ	χ	√	χ	χ	Grid
Optimized Resource Scheduling [26]	Cooperative	Allocation Request	χ	χ	χ	χ	χ	√	√	√	χ	χ	χ	Cloud
Improved Cost Based [27]	Batch Mode	Meta Tasks	√	χ	χ	χ	χ	χ	χ	χ	√	χ	χ	Cloud
A Compromised Time Cost [28]	Batch Mode	Count of Workflow instances	√	√	χ	χ	χ	χ	χ	χ	√	χ	χ	Cloud
HEFT Workflow [30]	Dependent	Highest Rank	χ	χ	√	χ	√	χ	χ	√	χ	χ	χ	Grid

VI. CONCLUSION

This study is only concerned with various numbers of available scheduling algorithms in distributed system and their basis criteria for task allocation. In addition to these algorithms, there are many studies related to and based on them that are searching for improvements, load balance, optimization and etc. There are many aspects could be considered as topics of research to introduce more accurate and improved algorithms rather that those introduced here such as the arriving rate of the tasks, cost of the task execution on each of the resource, cost of the communication, … etc. Also, applying these proposed algorithms on actual desired interested distributed system environment for practical evaluation can be other open problem in scheduling area. As seen from this paper the Max-min algorithm maps submitted tasks based on calculating the expected complete time and the expected execution time matrixes, then select tasks with overall maximum complete time to be executed on a resource with minimum overall execution time. We proposed an algorithm, improved Max-min, which is a research paper under reviewing for being published. The improved algorithm maps the task with overall maximum execution time to be executed by a resource that has minimum completion time.

The proposed algorithm increases opportunity of load balancing, resource utilization, tasks waiting times (delays) rather than the original algorithm. Also, it achieves makespan that is optimal rather than achieved by the original Max-min.

ACKNOWLEDGMENT

The Authors would to thank Dr. M. Alrahmawy for his pleased and valuable assist. Also, O. M. Elzeki would like to express his great ordinal thanks to Miss Ghada. H. El-Khawaga for her exclusive valuable assist.

REFERENCES

1. Journal of Theoretical and Applied Information Technology. (2011, April 9). [Online]. Available: http://www.jatit.org/distributed-computing/grid-vs-distributed.htm.
2. Dr.D.I.Georage Amalarethinam, P.Muthulakshmi, "An Overview of the Scheduling Policies and Algorithms in Grid Computing", " International Journal of Research and Review in Computer Science (IJRRCS)", vol. 2,no. 2, April 2011, pp. 280-294.
3. Prof. Robert van Engelen, "Concepts and Architecture of Grid Computing", "Advanced Topics Spring 2008, HPC II", spring 2008.
4. T. Casavant and J. Kuhl, "A Taxonomy of Scheduling in General Pupose Distributed Computing Systems", "IEEE Trans. on Software Engineering", vol. 14, no. 2, February 1988, pp. 141-154.
5. M. Arora, S. K. Das, R. Biswas, "A Decentralized Scheduling and Load Balancing Algorithm for Heterogeneous Grid Environments", "Proc. Of International Conference on Parallel Processing Workshops (ICPPW'02)", Vancouver, British Columbia Canada, August 2002, pp. 499-505.
6. Fatos Xhafa, Ajith Abraham, "Computational models and heuristic methods for Grid scheduling problems", "Future Generation Computer Systems 26", 2010, pp. 608-621.
7. El-Rewini, H., Ali, H.H., Lewis, T. Task scheduling in multiprocessing systems, IEEE Journal, December 1995, vol. 28, pp. 27-37.
8. U. Schwiegelshohn, R. Yahyapour, "Analysis of First-Come-First-Serve parallel job scheduling", " Proceedings of the 9th SIAM Symposium on Discrete Algorithms", 1998, pp. 629-638.
9. M. Maheswaran, S. Ali, H. J. Siegel, D. Hensgen and R. F. Freund, "Dynamic Matching and Scheduling of a Class of Independent Tasks onto Heterogeneous Computing Systems", "J. of Parallel and Distributed Computing", vol. 59, no. 2, November 1999, pp.107-131.
10. R. Armstrong, D. Hensgen, and T. Kidd, "The relative performance of various mapping algorithms is independent of sizable variances in run-time predictions", "7th IEEE Heterogeneous Computing Workshop (HCW '98)", 1998, pp. 79-87.
11. R. F. Freund, M. Gherrity, S. Ambrosius, M. Campbell, M. Halderman, D. Hensgen, E. Keith,T.Kidd, M. Kussow, J. D. Lima, F. Mirabile, L. Moore, B. Rust, and H. J. Siegel, "Scheduling resources in multi-user, heterogeneous, computing environments with SmartNet", "7th IEEE Heterogeneous Computing Workshop (HCW '98)", 1998, pp. 184-199.
12. R. F. Freund and H. J. Siegel, "Heterogeneous processing", IEEE Comput. 26, June 1993.
13. Rasmus V. Rasmussen, Michael A. Trick . Round Robin scheduling – a survey, European Journal of Operational Research, vol. 188, Issue 3, August 2008, pp. 617–636.
14. Ruay-Shiung Chang, Jih-Sheng Chang, Po-Sheng Lin, "An ant algorithm for balanced job scheduling in grids", "Future Generation Computer Systems 25", 2009, pp. 20–27.
15. M. Maheswaran, Sh. Ali, H. Jay Siegel, D. Hensgen, R. F. Freund, "Dynamic Mapping of a Class of Independent Tasks onto Heterogeneous Computing Systems", "Journal of Parallel and Distributed Computing", vol. 59, 1999, pp. 107-131.
16. T. D. Braun, H. Jay Siegel, N. Beck, L. L. Boloni, M. Maheswaran, A. I. Reuther, J. P. Robertson, M. D. Theys, and B. Yao, "A Comparison of Eleven Static Heuristics for Mapping a Class of Independent Tasks onto Heterogeneous Distributed Computing Systems", in "Journal of Parallel and Distributed Computing", vol. 61, 2001, pp. 810-837.
17. Saeed Parsa, Reza Entezari-Maleki, "RASA: A New Grid Task Scheduling Algorithm", "International Journal of Digital Content Technology and its Applications", vol. 3, no. 4, December 2009, pp. 91-99.
18. R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic. "Cloud Computing and Emerging IT Platforms: Vision, Hype, and Reality for Delivering Computing as the 5th Utility", "Future Generation Computer Systems, 25", June 2009, pp. 599-616.
19. Rodrigo N. Calheiros, Rajiv Ranjan, Anton Beloglazov, César A. F. De Rose, and Rajkumar Buyya**,** "CloudSim: A Toolkit for Modeling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms", unpublished.
20. R. Braun, H. Siegel, N. Beck, L. Boloni, M. Maheswaran, A. Reuther, J. Robertson, M. Theys, B. Yao, D. Hensgen and R. Freund, "A Comparison of Eleven Static Heuristics for Mapping a Class of Independent Tasks onto Heterogeneous Distributed Computing System", Journal of Parallel and Distributed Computing Systems, vol. 61, no. 6, 2001, pp.810-837.
21. He. X, X-He Sun, and Laszewski. G.V, "QoS Guided Min-min Heuristic for Grid Task Scheduling," Journal of Computer Science and Technology, vol. 18, 2003, pp. 442-451.
22. Kamalam.G.K and Muralibhaskaran.V, "A New Heuristic Approach:Min-Mean Algorithm For Scheduling Meta-Tasks On Heterogenous Computing Systems", "International Journal of Computer Science and Network Security", vol.10, no.1, January 2010.
23. Sameer Singh Chauhan,R. Joshi. C, "QoS Guided Heuristic Algorithms for Grid Task Scheduling", "International Journal of Computer Applications", vol. 2, no.9, June 2010, pp 24-31.
24. Singh. M and Suri. P.K, "QPS A QoS Based Predictive Max-Min, Min-Min Switcher Algorithm for Job Scheduling in a Grid", "Information Technology Journal", vol. 7, Issue. 8, 2008, pp. 1176-1181.
25. T. Kokilavani, Dr. D.I. George Amalarethinam, "Load Balanced Min-Min Algorithm for Static Meta-Task Scheduling in Grid Computing", "International Journal of Computer Applications", vol. 20, no. 2, April 2012, pp. 43-49.
26. Hai Zhong, Kun Tao, Xuejie Zhang, "An Approach to Optimized Resource Scheduling Algorithm for Open-source Cloud Systems ", 5th Annu. Conf. China Grid Conference, China, 2010.
27. Y. Yang, K. Liu, J. Chen, X. Liu, D. Yuan and H. Jin, "An Algorithm in SwinDeW-C for Scheduling Transaction-Intensive Cost-Constrained Cloud Workflows", 4th IEEE International Conference on e-Science, 374-375, Indianapolis, USA, December 2008.
28. Suraj Pandey, LinlinWu, Siddeswara Mayura Guru, Rajkumar Buyya, "A Particle Swarm Optimization-based Heuristic for Scheduling Workflow Applications in Cloud Computing Environments", unpublished
29. Cui Lin, Shiyong Lu," Scheduling ScientificWorkflows Elastically for Cloud Computing", 4th International Conf. IEEE on Cloud Computing, 2011.
30. W. Chen, J. Zhang, "An Ant Colony Optimization Approach to a Grid Workflow Scheduling Problem With Various QoS Requirements", "IEEE Transactions on Systems, Man, and Cybernetics - Part C: Applications and Reviews", vol. 39, no. 1, January 2009.
31. R.Madhubala, "An Illustrative study on Cloud Computing", "International Journal of Soft Computing and Engineering", vol. 1, issue. 6, January 2012, pp. 286-290.

AUTHORS PROFILE

O. M. Elzeki, lecture assistant, computer science at Faculty of Computer & Information System., Mansoura University, Egypt, Has 7 years of experience in IT technologies, works on Cloud Computing.

M. Z. Reshad, associate professor, computer science at Faculty of Computer & Information System., Mansoura University, Egypt, Has 10 years of experience in scientific research, cloud, grid, distributed computing and others.

M. A. Elsoud, associate professor, computer science at Faculty of Computer & Information System., Mansoura University, Egypt, Has 10 years of experience in scientific research, distributed computing, image processing, computer engineering and others.