Proposed Algorithm for Scheduling in Computational Grid using Backfilling and Optimization Techniques

Omar Dakkak, Shahrudin Awang Nor, Suki Arif
InterNetWorks Research Laboratory, School of Computing, Universiti Utara Malaysia, 06010 Kedah Malaysia.
oaldakkak@gmail.com

***Abstract*—In recent years, the fast evolution in the industry of computer hardware such as the processors, has led the application developers to design advanced software's that require massive computational power. Thus, grid computing has emerged in order to handle the computational power demands requested by the applications. Quality of service (QoS) in grid is highly required in order to provide a high service level to the users of Grid. Several interactions events are involved in determining the QoS level in grid such as; allocating the resources for the jobs, monitoring the performance of the selected resources and the computing capability of the available resources. To allocate the suitable resources for the incoming jobs, a scheduling algorithm has to manage this process. In this paper, we provide a critical review the recent mechanisms in "grid computing" environment. In addition, we propose a new scheduling algorithm to minimize the delay for the end user, Gap Filling policy will be applied to improve the performance of the priority algorithm. Then, an optimization algorithm will perform in order to further enhance the initial result for that obtained from backfilling mechanism. The main aim of the proposed scheduling mechanism is to improve the QoS for the end user in a real grid computing environment.**

***Index Terms*—Grid Computing; Scheduling; Backfilling; Meta-Heuristic.**

I. INTRODUCTION

Grid computing is computational technology, which aims to get the maximum benefits from idle resources, these resources could be CPU cycles, memory, bandwidth, storage, and so on [1]. The main idea behind this technology is to connect these idle resources together into one virtual network, thus a virtual system will be created and will share and manage the resources dynamically during operating time. Through the Grid, the grid system can supply sophisticated quality level services and access to a massive number of remote recourses to any user anytime. Unlike the web, which uses Internet Protocol (IP) to gain access to any content on the internet via Uniform Resource Identifier (URL), grid computing needs to have access to computational resources always [1, 2].

Users are enabled to use the resources like: database, hardware resources for many various devices that diffused everywhere, via very massive virtual network, in this case this network known as "Grid Computing ". For instance, suppose we have 20 computers available, half of these computers are busy, while the rests of them are idle. Therefore, the key idea is to use the CPU cycle for these idle machines in order to handle a huge task. In addition, there is a possibility to use some or all the of other pc's busy CPU, in case that these PC's are not using the whole cycle of their CPUs, and unify all the aggregate of processing power to handle such a huge task.

Based on Arora, Das and Biswas in [3], the grid is categorized into four main classes, which are: computational grid, access Grid, data grid and data-centric Grid. Computational grid concerns about providing the user with high computational power to process high computational power tasks. The resources in grid computing could be supercomputers, [4-7]. Access grid [8, 9], provides limited resources for a certain period of time. Data grid [4, 10-12] concerns about intensive and big data. This type of grid provides the service to save massive amounts of data that can be accessed or transferred. Whereas the main difference between data grid and data-centric grid [9, 13] is that data-centric, grid moves massive computations to the data rather than processing massive data to the computations.

Resource allocation in grid consists of four main steps, which are: scheduling, code transfer, data transmission and monitoring. The scheduling step consists of three main phases which are resource discovery, resource selection and job execution. Resource discovery is interest in searching and discovering the available resources, whereas resource selection chooses the best resource option to achieve better quality of service (QoS). In job execution phase, the submission of tasks (jobs) to the chosen resources is carried out [14].

Code transfer in charge of moving the codes that belong to individual tasks to the allocated resources to execute these codes. Data transmission concerns about transferring data from the task for execution. Finally, the monitoring is responsible for examining if the resources are available and the availability of the resources during job execution as well [14].

Classical scheduling mechanisms cannot meet the requirements for the end when the number of the jobs increased massively in a grid computing environment. To meet the requirements for non-trivial applications. Hence, this paper proposes a new mechanism that performs multi-level scheduling to avoid the flaws of the classical mechanisms. Therefore, backfilling technique becomes highly required due to its efficiency in exploiting the resources by filling the gaps that exist in the scheduler in short jobs.

The rest of the paper is organized as the following: Section II reviews some of the mechanisms in grid computing and provides a critical analysis and comparison of the reviewed mechanisms, Section III presents the new proposed mechanism. The paper is concluded in Section IV.

II. Scheduling Algorithms and Strategies in Grid Computing

Scheduling algorithms have a significant role in the quality of service that user of grid is requested for. The scheduling in resource allocation refers to the mapping process between the application and resources. The scheduling algorithms could be static, dynamic or adaptive. In this section, we review "some of many" scheduling algorithms.

A.Dynamic Objective with Advanced Scheduling

Leal et al. proposed performance-based Scheduling Strategies in [15]. This mechanism is very suitable for applications that require high throughput computational performance. It implements four strategies which are: Static Objective (SO), Dynamic Objective (DO), Static Objective with Advanced Scheduling (SO-AS) and Dynamic Objective with Advanced Scheduling (DO-AS). All previous techniques have shown less makespan, (better throughput).

However, DO-AS outperformed the other three techniques by offering better distribution. (Total number of jobs that were completed) measured the performance of this mechanism. This mechanism was simulated through GridWaySim Testbed. DO-AS approach starts by determining the performance of the system by applying linear equation $(n(t) = r\infty\, t - n1/2)$. Then, based on the results obtained from the previous step, the number of jobs that allocated to internal or external resources will be determined. DO-AS maps to the next job immediately, in order to take the advantage of the free slots in the scheduler. Therefore, another check for the available resources will be applied, in case that job is external, the job will move to internal resources (to avoid the situation of receiving the same job that already has been submitted). Figure 1 shows the flow chart steps for DO-AS.

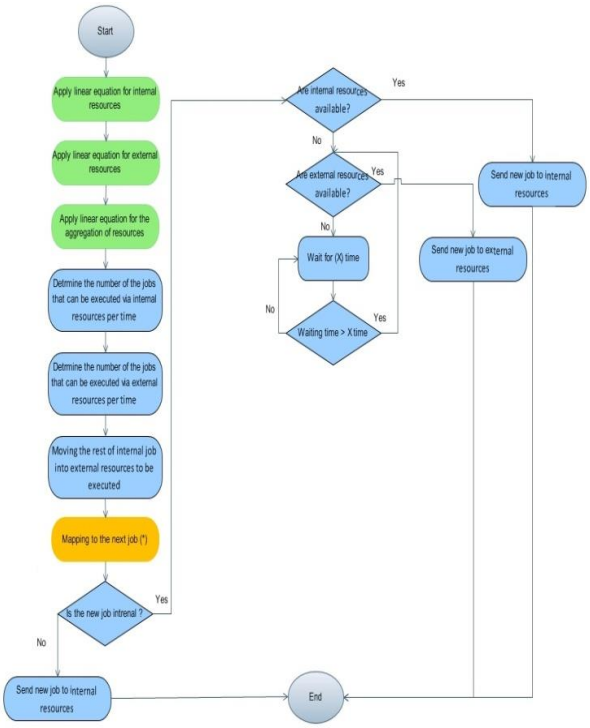

Figure 1: Dynamic objective with advanced scheduling

B. Swift Scheduling Mechanism

Scheduling Mechanism was proposed by Somasundaram and Radhakrishnan in [16]. This mechanism is suitable for distributed environments, when the tasks in the application are indivisible. This mechanism is integrating Shortest Job First (SJF) with a Heuristic Search algorithm. This approach reduces waiting average time by combining between informed search and uniformed search.

The analytical results showed that Swift Scheduling is overcoming Shortest (SJF), First Come First Serve (FCFS) and Simple Fair Task Order (SFTO). From Figure 2 below, it can be noticed that (SJF) is applied in order to schedule the job queue, whereas heuristic approach is used to match the resources with the scheduler. Figure 2 shows the concept of swift scheduling widths, line spaces, and text fonts are prescribed; please do not alter them. You may note peculiarities. For example, the head margin in this template measures proportionately more than is customary. This measurement and others are deliberate, using specifications that anticipate your paper as one part of the entire proceedings, and not as an independent document. Please do not revise any of the current designations.

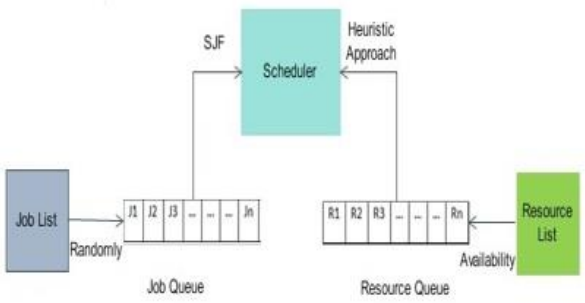

Figure 2: Swift scheduling concept [16]

C. Request Forwarding Approach

This mechanism was proposed by Iamnitchi and Foster in [17]. This approach concerns about the "Node" that user request has to go to (forwarded to). The user sends a request to the node "A". Node "A" replies back to the user, this replay contains information about node "A" resources. If the resources satisfy the user requirement, node "A" will be selected to serve the user. If not, the user will try to communicate with node "B" (another node) and so on. This procedure will be repeated if the user is not satisfied with the resource node, or when Time to Live (TTL) is over.

The request is applied by using one of four approaches, which are; Random Walk, Learning Based, Best Neighbor and Hybrid Learning Based Approach [18]. From the previous methods, Random Walk is the best in terms of reducing overhead since this approach does not need an extra memory to register the request. In addition, there is no need to store the history of the answers for the requests that reach the nodes. On the other hand, Random Walk suffers from integrity of choosing the best available resources due to TTL. Figure 3 shows the steps for this mechanism by combining between informed search and uniformed search.

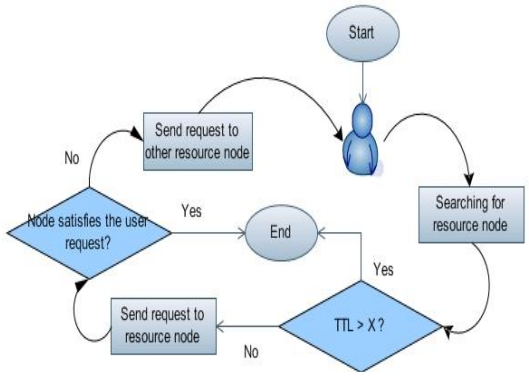

Figure 3: Request forwarding approach

D. Routing Transferring Model Based

In [19], Li et al. proposed Routing Transferring Model Based (RTM) was proposed. This mechanism consists of three main components, which are Resource Requester, Resource Router and finally Resource Provider. Topology and distributed type are very important factors to determine the complexity of this mechanism. As well as, the distribution of the resources has a very important role in the performance.

This mechanism works as the following: When the Resource Router got the request from the Resource Requester; it forwards it to the routing table. The routing table chooses the shortest path (if any). In case that there is no short path, the request is moved to another Resource Router. When the shortest path is located, the request will be forwarded to Resource Provider. If there is more than one available neighbor that can provide the resources, the request will be forwarded to the nearest one. This approach works well for resource discovery, due to the time for locating the resources is reduced because of resource replication. But this approach will consume a long time to check the resources info from the table (especially when the number of resources is big). Consequently, the performance of the scheduling will be slow. Figure 4 shows the flowchart for this mechanism.

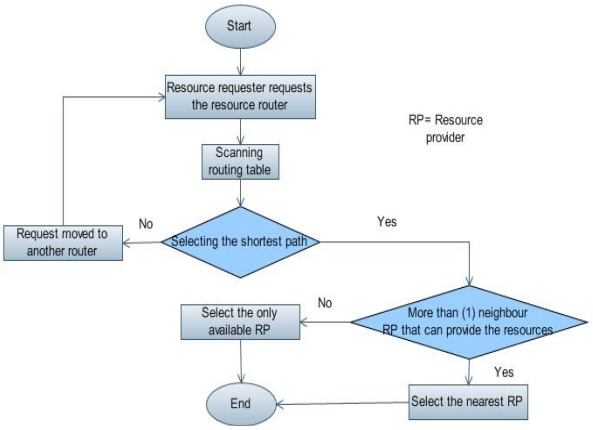

Figure 4: Routing transferring model based flowchart

E. The Parameter Based Mechanism

The Parameter Based mechanism proposed by [20], is based on the operating rate of the node (such as; CPU and memory). This mechanism uses Data Dissemination Algorithm as a searching mechanism. When a user inquires for the resources, resources status info will communicate with the node. Then a validation process will start, the validation process could be based on one of three strategies which are; Total Awareness, Neighbor Awareness and Distinctive Awareness.

When the validation process is completed, the request will be processed to the suitable resource. This mechanism reduces the overhead, but in case that Total Awareness strategy is used, the complexity will be increased as well. This is due to all dissemination messages go to the all nodes, whereas Neighbor Awareness and Distinctive Awareness reduce the overhead and the possibility of collision in the network. Figure 5 shows the flowchart for this mechanism.

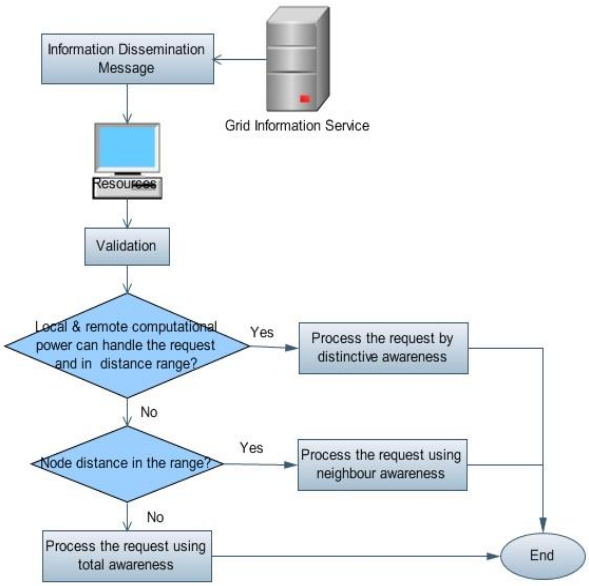

Figure 5: The parameter based mechanism

F. Peer to Peer Approach

Peer to peer approach is implemented for huge distribution network scales, p2p reduces the administrative overhead. In addition, this approach enables to seed and leach the data among the resources independently (centralized server to control the traffic is not required).

Unified Peer to Peer Database Framework (UPDF) [21] is one of the mechanisms basis on P2P approach. In order to achieve the scalability and manageability, UPDF uses graph-theoretic method. To overcome the local processing, Time To Live (TTL) is utilized in this mechanism. UPDF is scalable when the network has many resources, but the availability of information becomes tedious when the nodes are leaving and joining the network frequently.

G. Peer to Peer Approach

Volunteer Resource Allocation was proposed by Krawczyk and Bubendorfer in [22]. The idea behind this approach is; the idle resources will be donated by the volunteers. Volunteers will not get any reward for that. First, the user will send request to the broker; the broker will select the appropriate resources for the requested job. The resources will notify the broker that they are ready to serve the request. So the broker starts spreading the work to these resources via local scheduler. This approach can perform well only for a limited number of users.

In case that many users are requesting services from the broker, the executing time will take long time to complete. Figure 6 shows resource polling steps.

Volunteer Resource Allocation was proposed in [22]. The idea behind this approach is; the idle resources will be donated by the volunteers. Volunteers will not get any reward for that. First, the user will send request to the broker; the broker will select the appropriate resources for the requested job. The resources will notify the broker that they are ready to serve the request. So the broker starts spreading the work to these resources via local scheduler. This approach can perform well only for a limited number of users. In case that many users are requesting services from the broker, the executing time will take long time to complete. Figure 6 presents resource polling steps.

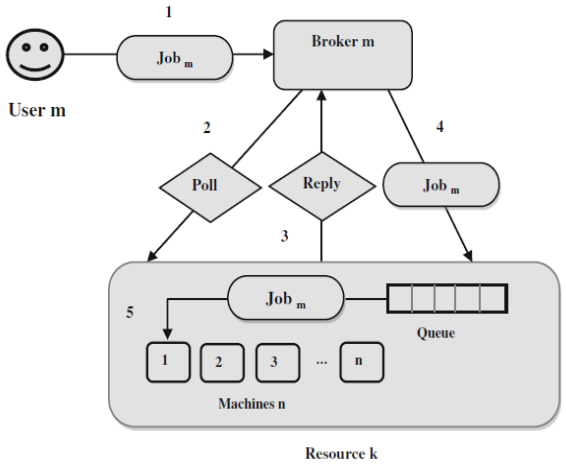

Figure 6: Volunteer Resource Allocation concept [22]

H. Economic, Market and Coalition Mechanism

The Economic mechanism was proposed in [23] by Buyya. Bartering and pricing are two main concepts in this approach. Five protocols can be used for bartering. These protocols are English-Dutch-Sealed-Bid and Vickrey. The budget of the user takes important role about selecting the resource besides the quality of the resource. This mechanism performs well when there is no deadline for executing the user's job, and when resources are distributed in a local place. While the performance will be worse in case that resources are allocated in global places.

Market Mechanism was proposed in [24]. Agents and facilitator are main components in this mechanism. The facilitator tells the agents about the price info. The agents can determine what is the optimal or near optimal request that can agree with facilitator price. Then the facilitator modifies the price and propagates the info. This mechanism suffers from the bottleneck in facilitator when the numbers of resources become bigger.

Wu, Ye and Zhangin in [24] proposed Coalition Formation Mechanism. This mechanism concerns about saving the cost through coordinating activities among the agents. Two models are used in this mechanism. These models are; Complementary Based and Utility Based Coalition. In Complementary Based, each party sequel the skill to make it easy for the agents, while Utility Based Coalition tries to distribute the profits amongst coalition members. The main disadvantage of this mechanism is the high overhead to form the coalition, which could affect the throughput badly as well as the cost is considered high comparing to other mechanisms that concern about the price. Table 1 below summarizes the key points for the reviewed mechanism.

Table 1
Resource Allocation Mechanisms in Computational Grid

Mechanism	Type	Strength	Weakness
DO-AS	Dynamic	Provides high throughput	Mapping to next job will take a long time in other grid environment(s)
Swift scheduling	Dynamic	Provides minimum (time, cost), maximum resource utilization.	Makespan is high.
Request forwarding approach	Static	Resourcing Discovery time is reduced	Suffers from high overhead
Routing transferring model based	Static	Provides good CPU power	Long time to scan resources info (in case of many resources)
Volunteer Resource Allocation)	Static	works fine with single user in the Grid	Slow execution time (for more than one user)
The parameter based mechanism	Static	Reduces overhead and congestion in the network	Complexity (Total awareness approach)
Peer to peer approach	Adaptive	Scalable	Availability of Information is required
Economic Mechanism	Static	Performs well when resources are in local place	Performance is affected for remote resources
Market Mechanism	Static	Provides travel arrangement, electric power network, Traffic flow network	Suffers from the bottle neck when resources are increased
Coalition Mechanism	Static	Agent info is not required	Overhead (in case the number of the users is huge).

III. PROPOSED SCHEDULING ALGORITHM

Due to the dynamicity of grid computing, scheduling the jobs becomes challenging, particularly when the number of the jobs increases. The traditional scheduling mechanism suffers from lack of flexibility when it allocates the jobs to the available resources.

For instance, Shortest Job First (SJF), which gives the privilege to the short jobs at the expense of the long ones. Even though most of the jobs in High Performance Computing application workloads (HPC) are very short ones [25]. Still SJF performance is questionable when this mechanism runs in real grid system.

In a real grid system for HPC workloads, when the system is running for months or years, we need to process the small jobs faster, but not at the expense of the long jobs. Even though 10 % only of the workloads are long jobs, but this ratio cannot be neglected. Moreover, traditional scheduling mechanisms cannot deal with the fragmentations that created in the queue due to different arrival times for the jobs. These fragmentations are a

CPU idle time which can be exploited if a proper policy is applied. Thus and when the number of the gaps (fragmentations) becomes high, traditional mechanisms cause inefficiency due to lack of exploiting the resources fully [26].

As a result, backfilling policy becomes required for such a system. The backfilling policy has no order or fixed rule to schedule the jobs, it simply backfills the short jobs in the gaps in order to reduce the waiting time for the whole jobs in the scheduler. The backfilling was described as something for nothing, a benefit without a tradeoff [26, 27].

To extremely utilize the resources to reduce the waiting time for the jobs to be processed, schedule-based approach has to be considered. In the queue-based system, the scheduling is executed blindly. The queue-system doesn't require any information about the incoming jobs; this can lead to a delay for the rest of the jobs ahead in the queue when backfilling is applied. This is can be justified due to the unawareness of the jobs execution time, the available resources and the size of the jobs. When such important information is missing, applying the backfilling would have repercussions as mentioned above [28].

The new proposed algorithm consists of two main parts; the first part will generate an initial solution, while the second will optimize the initial solution that generated from the first part. To extend the proposed algorithm to the dynamic mode, a gap filling policy will be applied. The gap filling policy will find the best suitable gaps (while the jobs are arriving), to fill these gaps. The jobs that cannot fit in any gap, they will be scheduled based on First Come First Serve (FCFS).

In dynamic Grid, the new arrival jobs could be short and have to wait in the schedule. This will waste the power of existing resources. To improve the utilization of the resources, smaller job size can be filled in suitable gaps without affecting the other jobs, which they are in the top of the queue. If there is no suitable gap, the gap filling policy will not be applied, and the traditional algorithm only will be practiced. The second part of the proposed algorithm will optimize the initial solution that obtained from the first part. This will be conducted by applying meta-heuristic algorithm. Meta-Heuristic algorithm will search for approximate and non-deterministic solution. Thus, mining for better solution will be targeted always without reaching for final best solution [28, 29].

The applied Local-Search Optimization Mechanism, will use a short memory. The memory will guide the search and offer the experience of the current and next search. While long memory could trap the algorithm in loops. The meta-heuristic approach will periodically enhance the initial solution based on the objective function. Thus, the utilization of the machine will be extra optimized and the waiting jobs in the queue will be less without affecting the other job in the queue since the job preemption is not supported. The generated solution from the first part of the algorithm is essential; since the second part applies meta-heuristic approach, which in turn belongs to the local search based type. Figure 7 presents the flowchart of the proposed algorithm, followed by pseudo code.

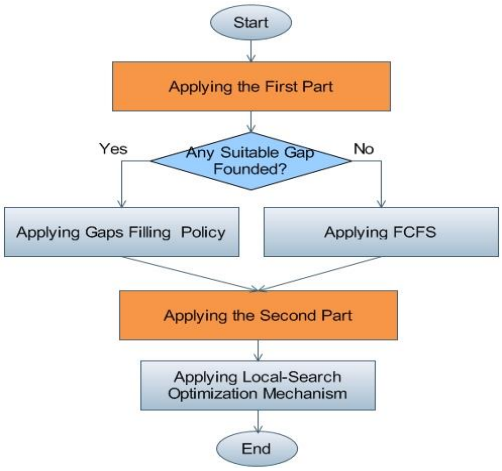

Figure 7: Proposed algorithm flowchart

```
Proposed Algorithm
1.produce NewScheduler();
2.start:
3.create resource-queue(Rn),job queue(Jn),
4.add jobs(j)to job queue(Jn);
5.add resources(r)to resource-queue(Rn);
6.finish;
7.generating initial result:
8.check for the gaps;
9.if found_gaps= true;
10.apply gap filling policy;
11.else
12.schedule the jobs based on arrival time;
13.end else
14.end if
15.apply local-search optimizing mechanism to find the fastest processing
time
16.allocate (j,r);
17.loop execution for(Jn,Rn);
18.end
```

Figure 8: Proposed Algorithm Pseudo Code

IV. Conclusion and Future Work

This papers has presented a critical review related to well-known mechanisms in scheduling for grid computing environment. Moreover, this paper proposed a new scheduling mechanism based on the multi-level scheduling approach. First, a backfilling mechanism is applied followed by optimization mechanism. The optimization is applied to further enhance the obtained solution from the first stage. The main aim of this mechanism, is to deal with real grid computing dynamic environment.

For future work, we will implement our proposed mechanism using real workloads. The main scope of the proposed mechanism will cover the HPC applications, since this is the vital implementation for grid computing.

Acknowledgment

This work is funded by UNIVERSITI UTARA MALAYSIA (UUM) S/O Code: 15861.

REFERENCES

[1] K. Czajkowski, S. Fitzgerald, I. Foster, and C. Kesselman, "Grid information services for distributed resource sharing," in *High Performance Distributed Computing, 2001. Proceedings. 10th IEEE International Symposium on*, 2001, pp. 181-194.

[2] R. van Engelen, M. Govindaraju, N. Koziris, and K. Psarris, "Distributed systems and grid computing(DSGC)," in *Symposium on Applied Computing: Proceedings of the 2006 ACM symposium on Applied computing*, 2006.

[3] M. Arora, S. K. Das, and R. Biswas, "A de-centralized scheduling and load balancing algorithm for heterogeneous grid environments," in *Parallel Processing Workshops, 2002. Proceedings. International Conference on*, 2002, pp. 499-505.

[4] R. Ranjan, A. Harwood, and R. Buyya, "Peer-to-peer-based resource discovery in global grids: a tutorial," *Communications Surveys & Tutorials, IEEE*, vol. 10, pp. 6-33, 2008.

[5] S. U. Khan and I. Ahmad, "A cooperative game theoretical technique for joint optimization of energy consumption and response time in computational grids," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 20, pp. 346-360, 2009.

[6] S. Khan and C. Ardil, "A Game Theoretical Energy Efficient Resource Allocation Technique for Large Distributed Computing Systems," in *PDPTA*, 2009, pp. 48-54.

[7] S. U. Khan, "A goal programming approach for the joint optimization of energy consumption and response time in computational grids," in *Performance Computing and Communications Conference (IPCCC), 2009 IEEE 28th International*, 2009, pp. 410-417.

[8] R. Ranjan, A. Harwood, and R. Buyya, "A taxonomy of peer-to-peer based complex queries: a grid perspective," *arXiv preprint cs/0610163*, 2006.

[9] D. B. Skillicorn, "Motivating computational grids," in *Cluster Computing and the Grid, 2002. 2nd IEEE/ACM International Symposium on*, 2002, pp. 401-401.

[10] J. Kołodziej and S. U. Khan, "Data scheduling in data grids and data centers: a short taxonomy of problems and intelligent resolution techniques," in *Transactions on Computational Collective Intelligence X*, ed: Springer, 2013, pp. 103-119.

[11] L. Wang, J. Tao, H. Marten, A. Streit, S. U. Khan, J. Kołodziej, *et al.*, "MapReduce across distributed clusters for data-intensive applications," in *Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW), 2012 IEEE 26th International*, 2012, pp. 2004-2011.

[12] R. Sharma, V. K. Soni, M. K. Mishra, and P. Bhuyan, "A survey of job scheduling and resource management in grid computing," *world academy of science, engineering and technology*, vol. 64, pp. 461-466, 2010.

[13] D. M. Batista and N. L. Da Fonseca, "A brief survey on resource allocation in service oriented grids," in *Proc. of Globecom Workshops*, 2007.

[14] M. B. Qureshi, M. M. Dehnavi, N. Min-Allah, M. S. Qureshi, H. Hussain, I. Rentifis, *et al.*, "Survey on grid resource allocation mechanisms," *Journal of Grid Computing*, vol. 12, pp. 399-441, 2014.

[15] K. Leal, E. Huedo, and I. M. Llorente, "A decentralized model for scheduling independent tasks in federated grids," *Future Generation Computer Systems*, vol. 25, pp. 840-852, 2009.

[16] K. Somasundaram and S. Radhakrishnan, "Task resource allocation in grid using swift scheduler," *International Journal of Computers, Communications & Control*, vol. 42, pp. 158-166, 2009.

[17] A. Iamnitchi and I. Foster, "On fully decentralized resource discovery in grid environments," in *Grid Computing—GRID 2001*, ed: Springer, 2001, pp. 51-62.

[18] A. Iamnitchi, I. Foster, and D. Nurmi, "A peer-to-peer approach to resource discovery in grid environments," in *IEEE High Performance Distributed Computing*, 2002.

[19] W. Li, Z. Xu, F. Dong, and J. Zhang, "Grid resource discovery based on a routing-transferring model," in *Grid Computing—GRID 2002*, ed: Springer, 2002, pp. 145-156.

[20] M. Maheswaran and K. Krauter, "A parameter-based approach to resource discovery in Grid computing systems," in *Grid Computing—GRID 2000*, ed: Springer, 2000, pp. 181-190.

[21] M. Marzolla, M. Mordacchini, and S. Orlando, "Peer-to-peer systems for discovering resources in a dynamic grid," *Parallel Computing*, vol. 33, pp. 339-358, 2007.

[22] S. Krawczyk and K. Bubendorfer, "Grid resource allocation: allocation mechanisms and utilisation patterns," in *Proceedings of the sixth Australasian workshop on Grid computing and e-research-Volume 82*, 2008, pp. 73-81.

[23] R. Buyya, "Economic-based distributed resource management and scheduling for grid computing," *arXiv preprint cs/0204048*, 2002.

[24] T. Wu*, N. Ye, and D. Zhang, "Comparison of distributed methods for resource allocation," *International Journal of Production Research*, vol. 43, pp. 515-536, 2005.

[25] D. Zotkin and P. J. Keleher, "Job-length estimation and performance in backfilling schedulers," in *High Performance Distributed Computing, 1999. Proceedings. The Eighth International Symposium on*, 1999, pp. 236-243.

[26] C. B. Lee, "On the user-scheduler relationship in high-performance computing," 2009.

[27] A. W. Mu'alem and D. G. Feitelson, "Utilization, predictability, workloads, and user runtime estimates in scheduling the IBM SP2 with backfilling," *IEEE Transactions on Parallel and Distributed Systems*, vol. 12, pp. 529-543, 2001.

[28] D. Klusacek and H. Rudová, "Improving QoS in computational Grids through schedule-based approach," in *Scheduling and Planning Applications Workshop at the Eighteenth International Conference on Automated Planning and Scheduling (ICAPS 2008), Sydney, Australia*, 2008.

[29] D. Klusacek, "Dealing with uncertainties in grids through the event-based scheduling approach," in *Fourth Doctoral Workshop on Mathematical and Engineering Methods in Computer Science (MEMICS 2008)*, 2008, pp. 978-980.