International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:6, No:1, 2012

Heuristics Analysis for Distributed Scheduling using MONARC Simulation Tool

Florin Pop

***Abstract*—**Simulation is a very powerful method used for high-performance and high-quality design in distributed system, and now maybe the only one, considering the heterogeneity, complexity and cost of distributed systems. In Grid environments, foe example, it is hard and even impossible to perform scheduler performance evaluation in a repeatable and controllable manner as resources and users are distributed across multiple organizations with their own policies. In addition, Grid test-beds are limited and creating an adequately-sized test-bed is expensive and time consuming. Scalability, reliability and fault-tolerance become important requirements for distributed systems in order to support distributed computation. A distributed system with such characteristics is called dependable. Large environments, like Cloud, offer unique advantages, such as low cost, dependability and satisfy QoS for all users. Resource management in large environments address performant scheduling algorithm guided by QoS constrains. This paper presents the performance evaluation of scheduling heuristics guided by different optimization criteria. The algorithms for distributed scheduling are analyzed in order to satisfy users constrains considering in the same time independent capabilities of resources. This analysis acts like a profiling step for algorithm calibration. The performance evaluation is based on simulation. The simulator is MONARC, a powerful tool for large scale distributed systems simulation. The novelty of this paper consists in synthetic analysis results that offer guidelines for scheduler service configuration and sustain the empirical-based decision. The results could be used in decisions regarding optimizations to existing Grid DAG Scheduling and for selecting the proper algorithm for DAG scheduling in various actual situations.

***Keywords*—**Scheduling, Simulation, Performance Evaluation, QoS, Distributed Systems, MONARC

I. Introduction

THE heuristics for distributed scheduling have the main goal optimization of resource allocation at local level (for a limited number of homogeneous resources) and at global level (in heterogeneous environments). The actual increasing interest in scheduling for heterogeneous distributed systems is due to the dimensions of some large scale applications, which makes inadequate a single parallel architecture to cover their needs for parallelism. When dealing with a computational Grid for parallel and distributed computing we have to efficiently exploit the computational power. In many practical cases, heterogeneous systems have proved to produce higher performance at lower cost than a single high performance computing machine [3]. Scheduling applications on wide-area distributed systems is useful for obtaining quick and reliable results in an efficient manner. Optimized scheduling algorithms for multi-criteria constraints are fundamentally important in order to achieve advanced resources utilization and considering scalability limits on hierarchical platforms [9]. The modern applications address many fields of activity like satellite data processing, medicine, and others. An essential requirement for a global Data Grid is to support the parallel and distributed processing of huge data. Therefore, it requires a scheduling system. The system needs to access and process the satellite image archives; the job manager should assign the jobs to available resources, basically by splitting the image in sub-images and process each sub-image on a different node. Understanding the timing behavior and users constrains of distributed applications gets more and more important because new real-time (like multimedia and health) applications require defined upper bounds for runtime performance, called deadlines, in order to provide application to application quality of service (QoS) [5]. In this context, the scheduling algorithms for distributed systems can be divided in two major categories, best effort based and QoS constraint based scheduling [7].

Best effort algorithm can be chosen according to performance and the tasks that need to be scheduled from the following: hybrid heuristic, adaptive generalized scheduler, adaptive scheduling algorithm, heterogeneous earliest finish time, greedy randomized adaptive search procedure, simulated annealing algorithm, genetic algorithm, task duplication based scheduling algorithm for heterogeneous systems, dynamical critical path, fast critical path [6]. QoS algorithms consist of different classes: back-tracking algorithms, approximation algorithms, loss and gain algorithm, and Bio-inspired algorithms (genetic, immune, ant colony system algorithms) [4]. The field of simulation was long-time seen as a viable alternative to develop new algorithms and technologies for distributed systems. Simulation represents a powerful support to enable the development of large-scale distributed systems, where analytical validations are prohibited by the nature of the encountered problems. The use of discrete-event simulators in the design and development of large scale distributed systems is appealing due to their efficiency and scalability. Their core abstractions of process and event map neatly to the components and interactions of modern-day distributed systems and allow the design of realistic scenarios. Compared with the alternative of implementing the new technology directly in real-world to demonstrate its viability, the simulation of distributed systems is a far better alternative because it achieves faster validation results, minimizing the costs involved by the deployment process [2]. This is an extension of [1] and presents a useful approach for analyzing the performance of scheduling algorithms for tasks with dependencies. Finding the optimal procedures for scheduling in Grid systems is important especially in large scale distributed computing systems and complex applications for

Florin Pop is with the University POLITEHNICA of Bucharest, Faculty of Automatic Control and Computers, Department of Computer Science, Spl. Independentei, 313, Bucharest 060042, Romania, Personal web page: http://florinpop.ro, E-mail: florin.pop@cs.pub.ro.

different research areas.The paper is organized as follows: next section presents the background provided by MONARC simulator for scheduling, and then the scheduling in distributed systems and scheduling heuristics are presented. The papers end with test scenarios, experimental results, synthetic analysis and a brief overview of related work. Finally, we will conclude and will identify future works.).

II. Simulation Model For Scheduling

MONARC is built based on a process oriented approach for discrete event simulation, which is well suited to describe concurrent running programs, network traffic as well as all the stochastic arrival patterns, specific for such type of simulation [17]. Threaded objects or "Active Objects" (having an execution thread, program counter, stack, etc.) allow a natural way to map the specific behavior of distributed data processing into the simulation program. With the MONARC simulation model, users can define various types of jobs to model common types of actions that can occur in any distributed systems.

TABLE I
MONARC Configurations File For Scheduling Experiments

```
# queue type to be used for storing events
queue_type = snoopy
# the maximum number of running threads in one burst
max_simultaneous_threads = 1000
# just one regional center
regional0 = testbed

# "testbed" regional center section
[testbed]
latitude = 44.25
longitude = 26.60
# the class name for the job scheduler
job_scheduler = JobSchedulerSimple
# the number of active job threads to be created initially
initial_pool_size = 0
lan0 = testbed_LAN
# maximum available bandwidth (Mbps)
lan0_max_speed = 10.0
lan0_connect = testbed_WAN
wan0 = testbed_WAN
# maximum available bandwidth (Mbps)
wan0_max_speed = 100.0
wan0_connect = testbed_Router
router0 = testbed_Router
# (seconds per package)
router0_latency = 10.0
# The name of the section for the cpu units
                                         cpu_unit0 = CPUtestbed
# just one activity, defined in the "activityTestbed" section
activity0 = activityTestbed

[CPUtestbed]
from = 1  # the id of the first CPU
to = 8    # the id of the last CPU
# (SI95) maximum available cpu power of one cpu unit
cpu_power = 112.0
# (MB) The maximum available memory
memory = 512.0
# The address of the first cpu unit, the others will have 100.100.100.101
link_node = 10.0.1.12
# (Mbps) The maximum available bandwidth of the link port attached
link_node_max_speed = 10.0
# connect all cpu units to the LAN named testbed_LAN
link_node_connect = testbed_LAN

[activityTestbed]
# The name of the activity class
class_name = ActivityScheduling
```

The MONARC simulation model is not limited to specific activities, the user having the possibility to easily incorporate new advanced job behaviors, as specified in the simulation scenario being executed. A simulated regional center also uses the services of a job scheduler. In order to schedule a job for execution the scheduler executes an appropriate scheduling algorithm. An example used for experiments in this paper is presented in Table I.

The modeled job object contains a number of parameters that are used to estimate the time needed for execution. The time needed by a job to complete a CPU-intensive operation is estimated based on a number of attributes such as the CPU power, memory and the processing time needed to complete. For the data processing jobs, these attributes depend on the type of data that the job works with (in the configuration file, the user can set this parameters for each data type used in the simulation). Once the CPU-intensive job starts processing the time needed to complete its operation is pre-computed. If another job starts executing on the same processing unit before the first one completes, then an interrupt mechanism is used to handle the re-estimation of the time needed for both jobs. The time needed for an I/O intensive job (for example, a data transfer handling type of job) is based on the mechanism provided by the network model. In this case again an interrupt mechanism is used to simulate the competition for bandwidth usage of data transfer jobs.

Within the job model the user can define new jobs starting from the basic behavior provided. It can even combine several behaviors in one single composite job type. This aspect can be used to simulate a job that transfer some data, then processes it and in the end transfers back the obtained results. This behavior represents a composition between the processing and data transfer types of jobs and can be modeled using only five lines of codes. But, in the same type, the user can include a job which does all the data processing according to some advanced algorithm, extending in this case one method provided by the processing data type of job.One interesting aspect is the job decompositions being offered by the job model. The user can specify a situation where a job requests some data, and then split itself in several parallel tasks, each one processing a particular chunk of data, and in the end the obtained results are reassembled and sent back. This fork-join programming paradigm can be modeled with the dependence mechanisms being offered.Any job cans instantiate new jobs. This means that, for example, one job receives the data, splits it into chunks and instantiate processing jobs, each one supplied with one specific chunk of data. It then specify the dependence, meaning it specifies what jobs must be executed after it finishes its own execution. The dependency between jobs can be specified in the form of a DAG structure [4].

The simulation model also allows the evaluation of advanced scheduling algorithms such as the ones we were particularly interested in, evaluating DAG Grid strategies. However, in order to accommodate the modeling of the DAG scheduling algorithms we had to extend this default behavior of the job model in MONARC.

The simulation model is presented in Figure 1 [8].

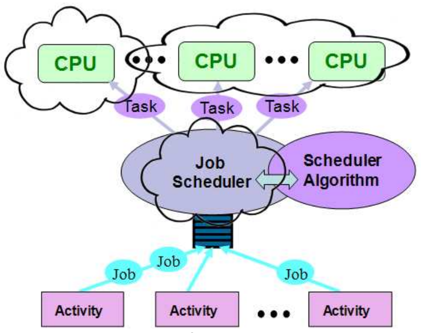

Fig. 1 Simulation Model for meta-scheduling in Clouds

III. Scheduling For Distributed Computing

We present a flexible approach for analyzing the performance of meta-scheduling technics and algorithms for tasks with or without dependencies in Cloud environments. Finding the optimal procedures for meta-scheduling is important especially in large scale distributed computing systems and complex applications for different research areas.
The main scope of the experimental scenarios is to evaluate scheduling algorithms using a powerful simulation tool, an approach suitable for different scheduling algorithms using bag-of-tasks model and various task dependencies and considering a wide range of systems architectures. Our proposed solution is based on MONARC, a generic simulation framework designed for modeling large scale distributed systems.

This section presents the extension of the simulation platform to accommodate various scheduling procedures and, as a case study, and offers a critical analysis of four well known scheduling strategies: Federated Grid Algorithm and Community-Aware Scheduling Algorithm. We consider static objective and dynamic objective. The comparisons of different algorithms for tasks with dependencies are also important. We consider CCF (Cluster ready Children First), ETF (Earliest Time First), HLFET (Highest Level First with Estimated Times) and Hybrid Remapper. The obtained results confirm that the proposed solution is a very good model for performance evaluation in a wide range of DAG scheduling algorithms and a large scale distributed system architectures.

The evaluation model for scheduling algorithms stats with presumption that it is quite difficult to make a comparison among different scheduling systems, since each of them is suitable for different situations.

For different scheduling systems, the class of targeted applications and resource configurations may differ significantly. The evaluation criteria for scheduling systems are: Application Performance Promotion (involves reviewing how well the applications can benefit from the deployment of the scheduling system), System Performance Promotion (concerns how well the whole system can benefit, like the utilization of resources is increased by, and how much the overall throughput gains), Scheduling Efficiency (the scheduling system should introduce additional overhead as low as possible and the overhead introduced by the scheduling system may exist in the information collection, the mapping process, and the resources allocation), Reliability (level of fault-tolerance for large collection of loosely-coupled resources considering that the scheduler should handle such frequent resource failures), Scalability (a scalable scheduling infrastructure should maintain good performance with not only increasing number of applications, but also increasing number of participating resources with heterogeneity) [4]. The scheduling objective is represented in Figure 2.

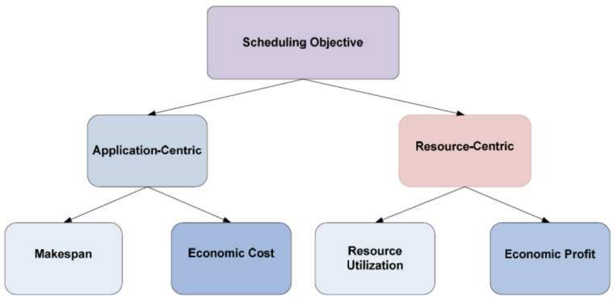

Fig. 2 Scheduling Objectives for Large Scale Environments Simulation Model

The distributed approach of simulation model is presented in Figure 3.

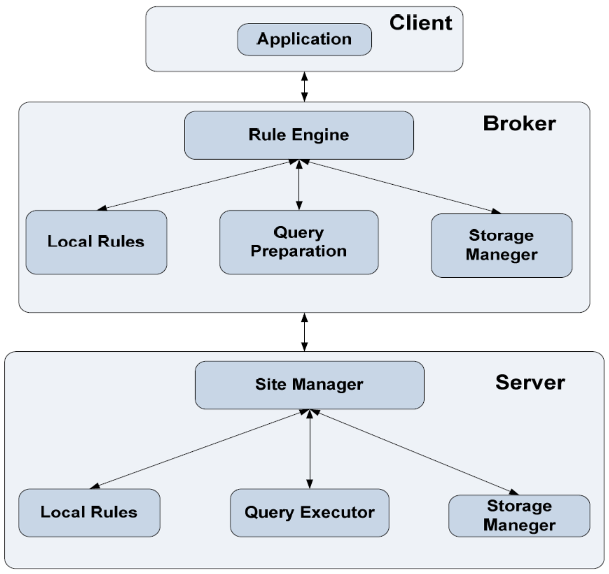

Fig. 3 Distributed approach of simulation model

When designing the scheduling infrastructure of a Grid system, these criteria are expected to receive careful consideration. Emphasis may be laid on different concerns among these evaluation criteria according to practical needs in real situations. There are some objective functions that could be used in order to optimize the process of scheduling. These functions could be a key order to satisfy the QoS constrains. There are *bottleneck functions* (for instance the *makespan* and the *maximum lateness*) and *sum/mean functions*. The latter ones may also appear in: mean/sum of completion time, mean/sum of weighted completion time, mean/sum of flow time, mean/sum of weighted flow time, and mean/sum of tardiness, number of late tasks and total weight of late tasks.

For task scheduling problem in distributed computing, considering dependencies for tasks, the model is bag-of-task with dependencies called DAG (Directed Acyclic Graph). In a DAG, a vertex (node) is the task and an arc is the communication constrain between two tasks. In a distributed system, the communication cost will be ignored if both tasks are run on the same processor. A schedule is an assignment of tasks (in the required order) on each processor. The goal is minimizing the makespan (or other function that is mention as a measure for QoS) of the schedule. Makespan represents the time elapsed between the start of the first task and the end of the last task and it is a good measure for QoS for scheduling problem.

The following algorithms give a suboptimal solution to the task scheduling problem. The trade-offs considered are minimizing makespan, running time of algorithm, number of processors and task communication costs [10]. These algorithms were chosen because they are much closed to DAG scheduling.

Wave Front Method (WFM): The wave fronts of the graph are determined according to the level of the vertices in a breadth-first-search traversal of the DAG. The vertices in each wave front are independent from each other, and are all assigned to different processors [11].

Critical Path Merge (CPM): A critical path in a DAG is a maximum weight root to leaf path (the path weight is the summation of all vertex and edge weights on the path). CPM computes the critical path, clusters all tasks in it, assigns them to the same processor, and removes them from the graph. This process is iterated until all tasks are scheduled. This logic behind this algorithm is that getting all the nodes on the current critical path on the same processor removes the edge costs and the duration of the critical path itself. Also the duration of an infinitely parallelizable dependency graph will still be equal to the duration of the critical path so reducing it is a necessary condition for optimality [12].

Heavy Edge Merge (HEM): Heavy Edge Merge works by iteratively clustering vertices (tasks) along edges with non-increasing weights. During an initialization stage, the edges are sorted in non-increasing order by edge weight, one task is assigned to each (virtual) processor, and the makespan of this assignment is computed. Then, all edges are processed in sorted order. For each edge, the makespan resulting from merging the tasks associated with the endpoints (perhaps clusters themselves) is computed. If the makespan increases, then the merge is not performed.

Min-min heuristic: Min-min heuristic uses minimum completion time (MCT) as a metric, means that the task which can be completed earliest is given priority. This heuristic considers a graph of tasks (G) and begins with the set U of all unmapped tasks. Then the set of minimum completion times tasks: $M=\{min\_compl\_time(M_j, T_i))|i,j \text{ in } G\}$ is found. M consists of one entry for each unmapped task. Next, the task with the overall minimum completion time from M is selected and assigned to the corresponding machine. Then the workload of the selected machine will be updated and finally the newly mapped task is removed from U. This process repeats until all tasks are mapped.

IV. Test Scenarios, Experimental Results and Synthetic Analysis

Using MONARC's extensions we proceeded to evaluate the scheduling algorithms in order to satisfy the QoS constrains discussed in this paper. We were particularly interested in analyzing their performance considering the use of two realistic scenarios. For this reason the modeling experiment considered the use of two and eight connected processors and a set of tasks with dependencies. The communication costs were considered between 0 and 20 and the tasks execution time were considered between 0 and 40.

The evaluated scheduling strategies were: Wave Front Method (WFM), Critical Path Merge (CPM), Heavy Edge Merge (HEM) and Min-min heuristic (MIN), discussed in this paper.

In our experiments, in order to satisfy the QoS constrains and to have the value 1 for satisfy operator for all assignments, we exclude all un-matching possibility. The makespan (as maximum execution time provided by scheduler) and logarithmic runtime (measured after tasks execution) were considered in order to compare the performances op evaluated scheduling strategies.

The results for presented scenarios are represented in Figures 4-7. The quantitative analysis of these graphics shows that CPM, MIN and WFM are simple and efficient. WFM has good results because the input graph was artificially built. HEM gives good schedules but takes a lot of time. Depending of the particularities of the input graph, each algorithm behaves strongly or weakly. As a direct observation, the site manager of a cluster must adapt algorithms (set-up) to the problem and this paper gives you the necessary insight.

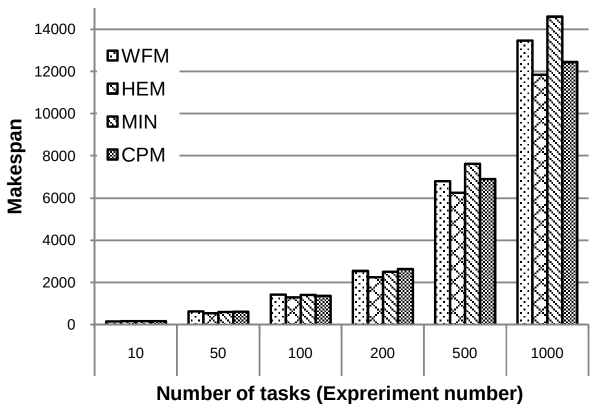

Fig. 4 Makespan Comparison for nprocs = 2

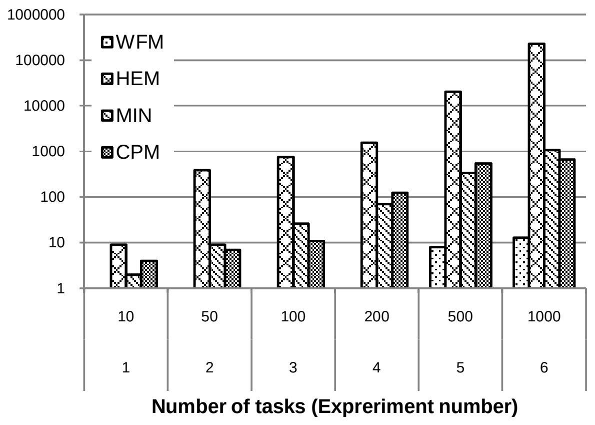

Fig. 5 Logarithmic Runtime for nprocs=2

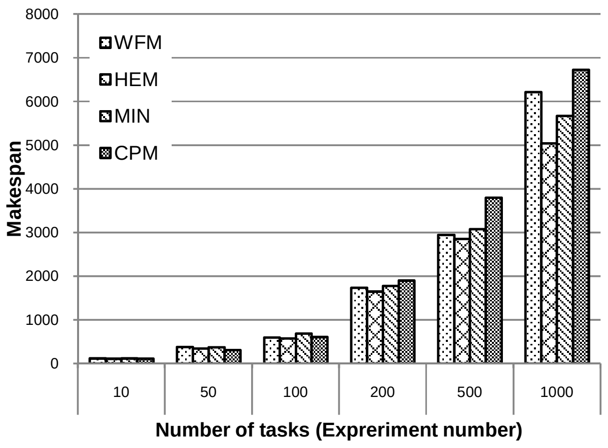

Fig. 6 Makespan Comparison for nprocs = 8

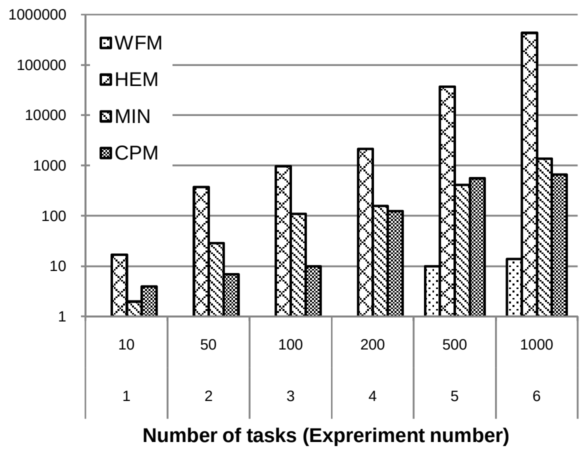

Fig. 7 Logarithmic Runtime for nprocs=8

The Figures 8 to 13 presents the simulation results for 50 and 100 set of tasks and for 3 different algorithms: FCFS Scheduling algorithm (queuing model), Shortest job first Scheduler (a model oriented to task execution cost evaluation) and Earliest deadline first Scheduler (a model oriented to balancing resource utilization). The conclusion of this tests show that the increasing of task number submitted to scheduling consist in a decreasing of CPU utilization in time, so it si good to calibrate this number in order to have the good performance.

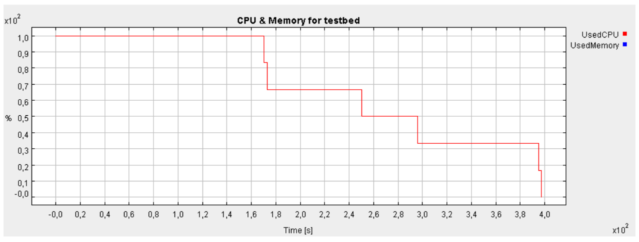

Fig. 8 Simple FCFS Scheduler (50 tasks)

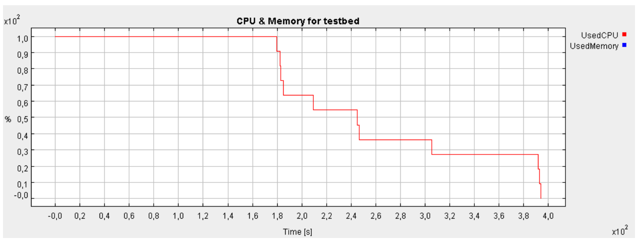

Fig. 9 Simple FCFS Scheduler (100 jobs)

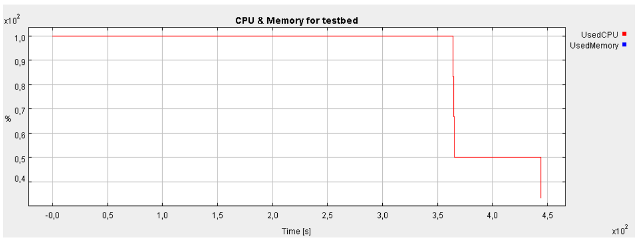

Fig. 10 Shortest job first Scheduler (50 tasks)

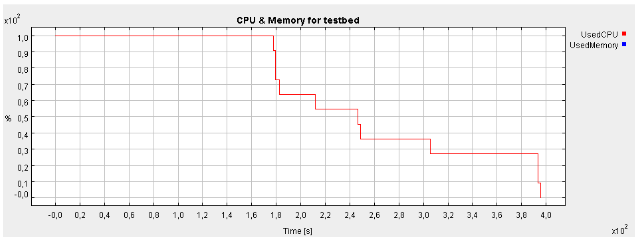

Fig. 11 Shortest job first Scheduler (100 jobs)

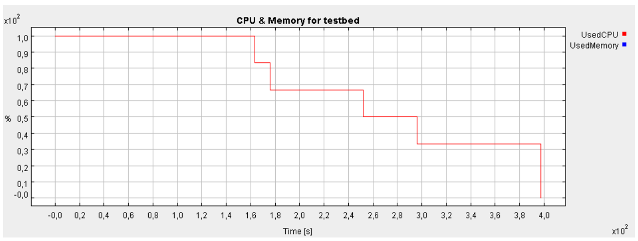

Fig. 12 Earliest deadline first Scheduler (50 tasks)

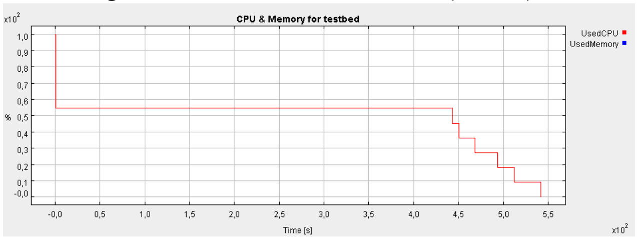

Fig. 13 Earliest deadline first Scheduler (100 tasks)

V. Related Work

Regarding simulation as a tool for scheduling evaluation there are multiple research projects and papers in the last ten years. Alea simulation is based on the GridSim simulation toolkit which was extended to provide a simulation environment that supports simulation of varying Grid scheduling problems. Alea demonstrates the features of the GridSim environment implementing an experimental centralized Grid scheduler which uses advanced scheduling techniques for schedule generation [13]. Li and Buyya [14] present statistical models that are able to reproduce various autocorrelation structures, including pseudo-periodicity and long range dependence. By conducting model-based simulation they quantitatively evaluate the performance impacts of workload autocorrelations in Grid scheduling. The results obtained indicate that autocorrelations result in system performance degradation, both at the local and the Grid level. Few years ago, Phatanapherom er al. [15] sustain that to develop grid scheduling algorithms, a high performance simulator is necessary since grid is an uncontrollable and unrepeatable environment. They propose a discrete event simulation library called HyperSim is used as extensible building blocks for grid scheduling simulator. The use of event graph model for the grid simulation is proposed. This model is well supported by HyperSim which yields a very high performance simulation. Fu and Fan [16] focus in their paper on how to schedule a system with distributed resources for multiple task execution. They explore the dynamic scheduling method for the parallel tasks with dependencies in distributed environments.

VI. Conclusion

Simulation is a very powerful tool, and now maybe the only one, considering the complexity (and cost!) of Grid systems. In Grid environments, it is hard and even impossible to perform scheduler performance evaluation in a repeatable and controllable manner as resources and users are distributed across multiple organizations with their own policies. In addition, Grid test-beds are limited and creating an adequately-sized test-bed is expensive and time consuming.

The aim of the experiments was to evaluate a few scheduling algorithms (for task without and with dependencies) in order to measure a QoS constrains (like makespan). It is very hard to compare these algorithms because there are many different assumptions and conditions from which some of the scheduling algorithms start. Tasks with DAG dependencies are frequent in case of Grid applications and they require advanced scheduling procedures that must consider QoS requirements. In this paper was proposed a simulation-based solution to evaluate the performances of Grid scheduling algorithms.

The results could be used in decisions regarding optimizations to existing Grid DAG Scheduling and for selecting the proper algorithm for DAG scheduling in various actual situations. The main contribution of the presented research consists in the development of the simulation layer in MONARC that is appropriate for DAG scheduling algorithms evaluation. It was introduced a set of recent algorithms and presented the solution to evaluate DAG scheduling algorithms using a generic simulator for large scale distributed systems guided by QoS constrains.

In this field, future work will include, among other things: the analysis of a wider set of scheduling algorithms currently used in Grid systems; the establishment of relevant performance measures and an improved simulation model; the evaluation of the current scheduling algorithms, using the chosen model. We will consider new scheduling algorithms for real-time scenarios, solutions for backup and recovery from error (re-scheduling) and solving the problem of co-scheduling and multi-criteria constraints scheduling.

ACKNOWLEDGMENT

The research presented in this paper is supported by national project "SORMSYS - Resource Management Optimization in Self-Organizing Large Scale Distributes Systems", Contract No. 5/28.07.2010, Project CNCSIS-PN-II-RU-PD ID: 201. The work has been co-funded by the Sectorial Operational Program Human Resources Development 2007-2013 of the Romanian Ministry of Labor, Family and Social Protection through the Financial Project and Agreement POSDRU/89/1.5/S/62557.

REFERENCES

[1] Florin Pop. 2011, *Simulation based Evaluation of QoS-guided Scheduling for Distributed Computing in Large Environments*, In Proc. of ESM 2011, The 2011 European Simulation and Moelling Confernce, October 24-26, University of Minho, Portugal, pp: 84-90.

[2] Ciprian Dobre, Florin Pop, and Valentin Cristea. 2009. New Trends in Large Scale Distributed Systems Simulation. In *Proceedings of the 2009 International Conference on Parallel Processing Workshops* (ICPPW '09). IEEE Computer Society, Washington, DC, USA, 182-189.

[3] Fatos Xhafa and Ajith Abraham. 2010. Computational models and heuristic methods for Grid scheduling problems. *Future Gener. Comput. Syst.* 26, 4 (April 2010), 608-621.

[4] Florin Pop and Valentin Cristea. 2009. Decentralised meta-scheduling strategy in Grid environments. *Int. J. Grid Util. Comput.* 1, 3 (August 2009), 185-193.

[5] Li Chunlin and Li Layuan. 2007. Optimization decomposition approach for layered QoS scheduling in grid computing. *J. Syst. Archit.* 53, 11 (November 2007), 816-832.

[6] Peng Li and Binoy Ravindran. 2004. Fast, Best-Effort Real-Time Scheduling Algorithms. *IEEE Trans. Comput.* 53, 9 (September 2004), 1159-1175.

[7] Zhiang Wu, Junzhou Luo, and Fang Dong. 2006. Measurement model of grid QoS and multi-dimensional QoS scheduling. In *Proceedings of the 10th international conference on Computer supported cooperative work in design III* (CSCWD'06), Weiming Shen, Junzhou Luo, Zongkai Lin, Jean-Paul A. Barthos, and Qi Hao (Eds.). Springer-Verlag, Berlin, Heidelberg, 509-519.

[8] Florin Pop, Ciprian Dobre, and Valentin Cristea. 2008. Performance Analysis of Grid DAG Scheduling Algorithms using MONARC Simulation Tool. In *Proceedings of the 2008 International Symposium on Parallel and Distributed Computing* (ISPDC '08).

[9] Fabricio A. B. da Silva and Hermes Senger. 2011. Scalability limits of Bag-of-Tasks applications running on hierarchical platforms. *J. Parallel Distrib. Comput.* 71, 6 (June 2011), 788-801.

[10] N.M. Amato, P. An. 2000. *Task scheduling and parallel mesh-sweeps in transport computations*, Technical Report TR00-009, Department of Computer Science, Texas A&M University, January 2000.

[11] Hamed Nooraliei and Amir Nooraliei. 2009. Path Planning Using Wave Front's Improvement Methods. In *Proceedings of the 2009 International Conference on Computer Technology and Development - Volume 01* (ICCTD '09), Vol. 1. IEEE Computer Society, Washington, DC, USA, 259-264.

[12] Marek Wieczorek, Andreas Hoheisel, and Radu Prodan. 2009. Towards a general model of the multi-criteria workflow scheduling on the grid. *Future Gener. Comput. Syst.* 25, 3 (March 2009), 237-256.

[13] Dalibor Klusáček, Luděk Matyska, and Hana Rudová. 2007. Alea: grid scheduling simulation environment. In *Proceedings of the 7th international conference on Parallel processing and applied mathematics* (PPAM'07), Roman Wyrzykowski, Konrad Karczewski, Jack Dongarra, and Jerzy Wasniewski (Eds.). Springer-Verlag, Berlin, Heidelberg, 1029-1038.

[14] Hui Li and Rajkumar Buyya. 2009. Model-based simulation and performance evaluation of grid scheduling strategies. *Future Gener. Comput. Syst.* 25, 4 (April 2009), 460-465.

[15] Sugree Phatanapherom, Putchong Uthayopas, and Voratas Kachitvichyanukul. 2003. Dynamic scheduling II: fast simulation model for grid scheduling using HyperSim. In *Proceedings of the 35th conference on Winter simulation: driving innovation* (WSC '03). Winter Simulation Conference 1494-1500.

[16] Yanfang Fu and Yan Fan. 2010. Research of the Simulation Grid System Based on Multi-agent Dynamic Scheduling. In *Proceedings of the 2010 Second International Conference on Computer Modeling and Simulation - Volume 03* (ICCMS '10), Vol. 3. IEEE Computer Society, Washington, DC, USA, 392-395.

[17] Ciprian Dobre, Corina Stratan, and Valentin Cristea. 2008. Realistic Simulation of Large Scale Distributed Systems using Monitoring. In *Proceedings of the 2008 International Symposium on Parallel and Distributed Computing* (ISPDC '08). IEEE Computer Society, Washington, DC, USA, 434-438

Florin Pop, PhD, is lecturer with the Computer Science Department of the University Politehnica of Bucharest. His research interests are oriented to: scheduling in Grid environments (his PhD research), distributed system, parallel computation, communication protocols and numerical methods. He received his PhD in Computer Science in 2008 with Magna cum laudae distinction. He is member of RoGrid consortium and participates in several research projects, in collaboration with other universities and research centers from Romania and from abroad developer. E-mail: florin.pop@cs.pub.ro
Web-page: http://florinpop.ro