OPTIMIZATION OF SCHEDULIG PROCESS IN GRID ENVIRONMENTS

Florin POP[1], Valentin CRISTEA[2]

Acest articol prezintă metodele de optimizare a planificării în Grid. Sunt descrise modelele de planificare şi algoritmi de planificare folosiți în medii Grid. Este propusă, de asemenea, o taxonomie a algoritmilor de planificare, bazată pe nivele funcționale, care oferă o referință pentru proiectarea unei soluții complete de planificare în Grid. Articolul prezintă şi analizează posibilitățile de optimizare cu mai multe crierii. Analiză critică a algoritmilor de planificare evidențiază aspectele importante nerezolvate în acest domeniu şi constituie suportul metodelor de optimizare. Comparația instrumentelor de planificare existente în Grid evidențiază evoluția sistemelor de planificare. Rezultatele experimentale prezentate demonstrează o îmbunătățire pentru echilibrarea încărcării şi pentru timpul de execuție în cazul algoritmului genetic folosit pentru optimizare.

This article presents of the optimization of Grid scheduling problem. The scheduling models of Grid scheduling algorithms are presented. The paper also proposes a taxonomy of scheduling algorithms, based on functional levels, which offers a reference for designing a complete solution for Grid scheduling. The analyses of possibilities for multi-criteria optimization are presented. A critical analysis of scheduling algorithms describes the open issues in this field and represents the support of optimization methods design. The comparison of existing Grid scheduling tools highlights the state of the art for scheduling in Grids. We present the evaluation of some scheduling mechanism. The experimental results demonstrate a very good improvement in load-balancing and execution time for scheduling algorithm used for optimization.

Keywords: Grid Scheduling, Multi-criteria optimization, Genetic Algorithms

1. Introduction

Grid computing became a very important model for resource sharing in Virtual Organizations (VOs). Grid systems allow the use of temporarily available resources, the execution of large tasks that require high computing power and large memory volumes. On the other side, resource sharing in grid systems (generally, in very large distributed systems) is more complex and asks for more complicated management policies and techniques. An important management

[1] PhD Student, Faculty of Automatics and Computer Science, University POLITEHNICA of Bucharest, Romania, email: florin.pop@cs.pub.ro

[2] Professor, Faculty of Automatics and Computer Science, University POLITEHNICA of Bucharest, Romania, email: valentin.cristea@cs.pub.ro

function is task scheduling. In the case of Grid systems, task scheduling has two objectives. One objective targets the efficient use of resources, similar with schedulers found in traditional operating systems. The second objective, not less important, is related to the VO concept and aims to respond to the requirements stated by the users in a VO concerning the performance of tasks execution, such as the response time. This is why the scheduling function has been distributed to two components: one which is closer to the resources (the local scheduler), and a second (the meta-scheduler) closer to the application. Scheduling in distributed systems has been significantly improved due to innovations proposed in Grid systems and VO management. The scheduling algorithms for large scale distributed systems (LSDS), such as the Grid systems, are subject to recent research. Also, Grid system scheduling tools evolved according to users' requirements and system constraints to cope with important characteristics of LSDS like heterogeneity, dynamicity, process distribution, and data distribution.

The scheduling in Grid systems is very complicated. The resource heterogeneity, the size and number of tasks, the variety of policies, and the high number of constraints are some of the main characteristics that contribute to this complexity. The necessity of scheduling in Grid is sustained by the increasing of number of users and applications. The design of scheduling algorithms for a heterogeneous computing system interconnected with an arbitrary communication network is one of the actual concerns in distributed system research.

2. Related Work

In this section it is presented a brief description of some Grid scheduling systems, and then a comparison of existing (and presented) grid scheduling tools is realized.

Condor is a specialized resource management system (RMS) developed at the University of Wisconsin-Madison for compute-intensive tasks. Like other full featured batch systems, Condor provides a task queuing mechanism, scheduling policy, priority scheme, resource monitoring, and resource management. Users submit their serial or parallel tasks to Condor, Condor places them into a queue, chooses when and where to run the tasks based upon a policy, carefully monitors their progress, and ultimately informs the user upon completion. Condor can be used to manage a cluster of dedicated compute nodes. In addition, unique mechanisms enable Condor to effectively harness wasted CPU power from otherwise idle desktop workstations.

Condor-G [20] represents the link between two technologies: Condor and the Globus toolkit [9]. With Condor-G it is possible to use Condor inside a Grid environment. Condor-G can be used as a reliable submission and task management service for one or more sites, Condor as the fabric management

service and finally the Globus toolkit can be used as the bridge between them [4]. Another service of Condor is the Directed Acyclic Graph Manager (DAGMan) for executing multiple tasks with dependencies described as DAGs. Each task is a node in the graph and the edges identify their dependencies. DAGMan does not support automatic intermediate data movement, so users have to specify data movement transfer through pre-processing and post-processing commands associated with processing task. The DAGMan meta-scheduler processes the DAG dynamically, by sending to the condor scheduler the tasks as soon as their dependencies are satisfied and they become ready to execute.

Another resource management project is **PBS**, which is available as open source (TORQUE version) and provides control over batch tasks and distributed compute nodes. It is a community effort based on the original PBS project and, with more than 1,200 patches, has incorporated significant advances in the areas of scalability, fault tolerance, and feature extensions contributed by NCSA, OSC, USC, the U.S. Dept of Energy, Sandia, PNNL, U of Buffalo, TeraGrid, and many other leading edge HPC organizations. Torque is a centralized system, in which a controller is responsible for the system-wide decision-making and for estimating the state of the system. The controller mediates access to distributed resources by discovering suitable data sources for a given analysis scenario, suitable computational resources, optimally mapping analysis tasks to resources, deploying and monitoring task execution on selected resources, accessing data from local or remote data source during task execution and collating and presenting results [12].

In **Moab**, a cluster management solution that integrates scheduling, managing, monitoring and reporting of cluster workloads, we find a simplified and unified management across one or multiple hardware, operating system, storage, network, license and resource manager environments. "Its task-oriented graphical management and flexible policy capabilities provide an intelligent management layer that guarantees service levels, speeds task processing and easily accommodates additional resources". The precursor to Moab is Maui, "an optimized, configurable tool capable of supporting an array of scheduling policies, dynamic priorities, extensive reservations, and fairs-hare capabilities". Features like virtual private clusters, basic trigger support, graphical administration tools, and a Web-based user portal are present both in Moab and Maui.

Grids offer a dramatic increase in the number of available processing and storing resources that can be delivered to applications. However, efficient task submission and management continue being far from accessible to ordinary scientists and engineers due to their dynamic and complex nature.

The aim of the **GridWay** project is to do the research and to develop the technology required to automatically perform all the submission steps and also to provide the runtime mechanisms needed for dynamically adapting the application

execution. The GridWay framework has been developed to reduce the gap between Grid middleware and application developers. The GridWay framework is a component for meta-scheduling in the Grid Ecosystem intended for end users and grid application developers. GridWay is a workload manager that performs task execution management and resource brokering on a grid consisting of distinct computing platforms managed by Globus services. GridWay allows unattended, reliable, and efficient execution of single, array, or complex tasks on heterogeneous and dynamic grids. GridWay performs all the task scheduling and submission steps transparently to the end user and adapts task execution to changing grid conditions by providing fault recovery mechanisms, dynamic scheduling, migration on-request and opportunistic migration. GridWay on Globus provides decoupling between applications and the underlying local management systems [13].

AppLeS (Application Level Schedulers) is a hierarchical scheduler, using predictive heuristics, online rescheduling, and fixed application oriented policy. It has an agent based methodology for application level scheduling. AppLeS agents are based on the application level scheduling paradigm, i.e. everything about the system is evaluated in terms of its impact on the application. Each application has its own AppLeS and each AppLeS combines both static and dynamic information to determine a customized application specific schedule and implement that schedule on the distributed resources [11]. The goal of the AppLeS project is to develop software to assist and enhance the scheduling activities of the application developer on a distributed meta-computing system.

3. General Characteristics of Grid Scheduling

Due to Grid systems' characteristics, the main issue for Grid scheduling is to develop a Meta-Scheduling architecture that encompasses heterogeneous and dynamic clusters. This architecture is a decentralized one and represents the solution for the scheduling problem at a Global Grid level. At this level, the QoS (quality of service) constraint is very important. The scheduling methods for decentralized heterogeneous environment are based on heuristics that consider complex applications. The tasks that compose these applications can have different dimensions and can be based on diverse data and control patterns.

The optimization of scheduling process for Grid systems tries to provide better solutions for the selection and allocation of resources to current tasks. The scheduling optimization is very important because the scheduling is a main building block for making Grids more available to user communities. The optimization methods for Grid scheduling are the main subject of this paper. The scheduling problem is NP-Complete. Consequently, approximation algorithms are

considered, which are expected to quickly offer a solution, even if it is only near-to-optimal.

QoS is a requirement for many Grid applications. QoS might refer to the response time, the necessary memory, etc. It might happen that these requirements are satisfied only by specific resources, so that they only these resources can be assigned for that application. Situations might become more complex when there are more tasks having QoS requirements, and several resources exist which satisfy them. The resource allocation under QoS constrains is another subject for the optimization process.

Scheduling in Grid computing must also take into account additional issues such as the resource owners' requirements, the need to continuously adapt to changes in the availability of resources, and so on. In these cases, a number of challenging issues need to be addressed: maximization of system throughput and user satisfaction, the sites' autonomy (the Grid is composed of resources owned by different users, which retain control over them), and scalability.

The fault tolerance is also important in Grid. The fault tolerant solutions for Grid Scheduling are based on error recovery and re-scheduling. Two of the problems related to re-scheduling are the high cost and the lack of coping with dependent tasks. For computational intensive tasks, re-scheduling the original schedule can improve the performance. But, re-scheduling is usually costly, especially in Directed Acyclic Graphs (DAGs) where there are extra data dependencies among tasks. Current research on DAG rescheduling leaves a wide open area on optimization for the scheduling algorithms.

Performance prediction is also used in optimizing the scheduling algorithms. Existing scheduling algorithms only consider an instant value of the performance at the scheduling time, and assume this value remains constant during the task execution. A more accurate model should consider that performance changes during the execution of the application.

In many cases, the data must be transported to the place where tasks will be executed. Consequently, scheduling algorithms should consider not only the task execution time, but also the data transfer time for finding a more realistic mapping of tasks [6]. Only a handful of current research efforts consider the simultaneous optimization of computation and data transfer scheduling.

According with all these presented aspects, many research activities are being conducted to develop a good scheduling approach for distributed nodes. The activities vary widely in a number of characteristics, e.g. support for heterogeneous resources, objective function(s), scalability, co-scheduling methods, and assumptions about system characteristics.

In compliance with the new techniques in application development, it is more natural to consider schedulers closer to Grid applications. They are responsible for the management of tasks, such as allocating resources, managing

the tasks for parallel execution, managing of data transfers, and correlating the events. To provide their functions, a scheduler needs information coming from monitoring services available in the platform.

4. Multi-Criteria Optimization for Grid Scheduling

A hierarchical taxonomy for scheduling algorithms in parallel and distributed systems was made for the first time by Casavant in [5] (see Fig. 1). Scheduling algorithms in Grid fall into a sub-category of this taxonomy. This general taxonomy described scheduling models. A part of these models was described in the previous section.

Basically, we have local and global scheduling. A *Local scheduler* considers a single CPU (a single machine). *Global scheduling* is dedicated to multiple resources. Scheduling for distributed systems such as the Grid is part of the global scheduling class.

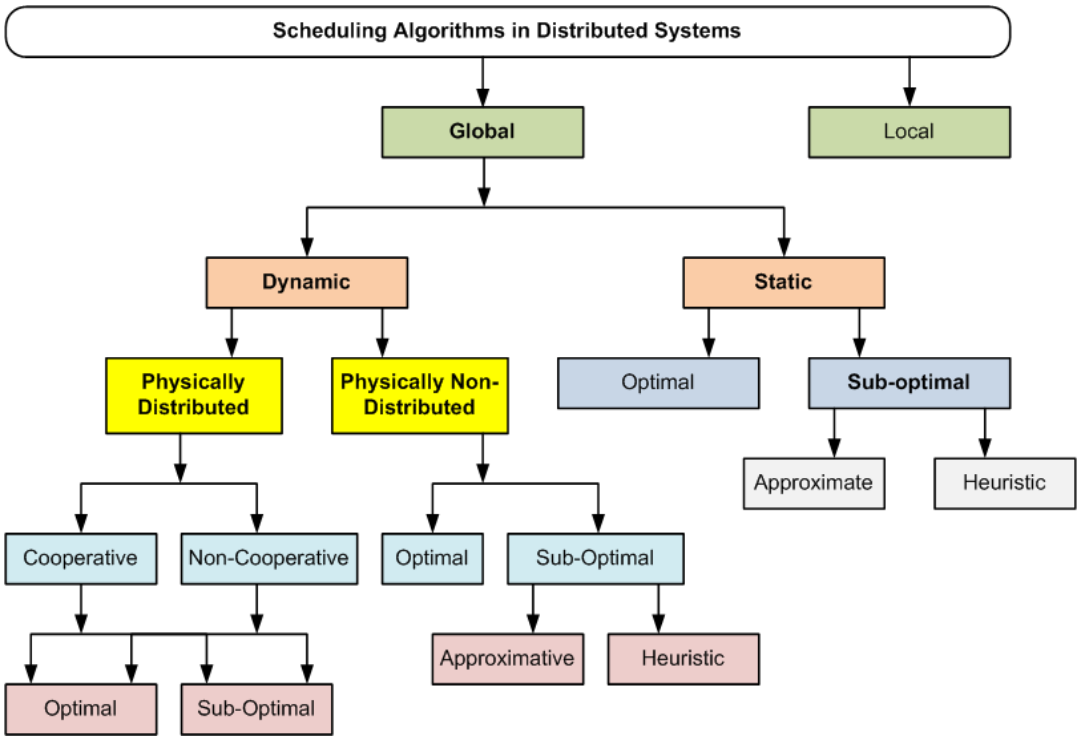

Fig. 1. Taxonomy of scheduling algorithms (a hierarchical approach)

For global scheduling there are two way to allocate the resources for tasks: static or dynamic. In the *static scheduling* model, every task is assigned only once to a resource. A realistic prediction of the cost of the computation can be made

before to the actual execution. The static model adopts a "global view" of tasks and computational costs. One of the major benefits is the ease of implementation. On the other hand, static strategies cannot be applied in a scenario where tasks appear a-periodically, and the environment undergoes various state changes. Cost estimate does not adapt to situations in which one of the nodes selected to perform a computation fails, becomes isolated from the system due to network failures, is so heavily loaded with tasks that its response time becomes longer than expected, or a new computing node enters the system. These changes are possible in Grids [3]. In *dynamic scheduling* techniques tasks are allocated dynamically at their arrival. Dynamic scheduling is usually applied when it is difficult to estimate the cost of applications, or tasks are coming online dynamically (in this case, it is also called online scheduling). Dynamic task scheduling has two major components: one for system state estimation (other than cost estimation in static scheduling) and one for decision making. System state estimation involves collecting state information through Grid monitoring and constructing an estimate. On this basis, decisions are made to assign tasks to selected resources. Since the cost for an assignment is not always available, a natural way to keep the whole system healthy is by balancing the loads of all resources [7].

The dynamic scheduling could be done in a *physically distributed* environment (grids) or in a *physically non-distributed* system (cluster). Sabin et al [17] propose a centralized scheduler which uses backfill to schedule parallel tasks in multiple heterogeneous sites. Arora et al [2] present a completely decentralized, dynamic and sender-initiated scheduling and load balancing algorithm for the Grid environment.

In distributed scheduling (global) the involved nodes could working cooperatively or independently (non-cooperatively). In the *non-cooperative scheduling*, individual schedulers act alone as autonomous entities and arrive at decisions regarding their own optimum objects independent of the effects of the decision on the rest of system. In *cooperative scheduling* each Grid scheduler has the responsibility to carry out its own portion of the scheduling task [18]. If all information about the state of resources and the tasks is known, an *optimal* assignment could, considering an objective function. But due to the NP-Complete nature of scheduling algorithms *sub-optimal* algorithms for scheduling represent good solutions.

The sub-optimal algorithm can be further divided into the following two general categories: approximate and heuristic. The *approximate algorithms* use formal computational models and are satisfied when a solution that is sufficiently "good" is found. If a metric is available for evaluating a solution, this technique can be used to decrease the time taken to find an acceptable schedule. The *heuristic algorithms* make the most realistic assumptions about a priori knowledge concerning process and system loading characteristics. The heuristic algorithms

are the solutions to the scheduling problem which cannot give optimal answers but require amount of cost and other system resources to perform their function.

The scheduling process, in sub-optimal case, could be conducted by objective functions. Objective functions can be classified into two categories: application-centric and resource-centric.

Application-centric function in scheduling tries to optimize the performance of each individual application [19]. Most of current Grid applications' concerns are about time, for example the makespan, which is the time spent from the beginning of the first task in a task to the end of the last task of the task. On the other hand, the economic cost that an application needs to pay for resources utilization becomes a concern of some of Grid users [4].

Resource-centric function in scheduling tries to optimize the performance of the resources. They are usually related to resource utilization, for example, throughput (ability of a resource to process a certain number of tasks), utilization (which the percentage of time a resource) [21]. As economic models are introduced into Grid computing, economic profit (which is the economic benefits resource providers can get by attracting Grid users to submit applications to their resources) also comes under the purview of resource management policies.

Adaptive Scheduling is used to make scheduling decisions change dynamically according to the previous, current and/or future resource status [5]. In Grid, adaptive scheduling could be done considering tree criteria: the heterogeneity of candidate resources, the dynamism of resource performance, and the diversity of applications. Relations between tasks divide scheduling algorithms in two classes: independent task scheduling and DAG scheduling (workflow scheduling). Dependency means there are precedence orders existing in tasks, that is, a task cannot start until all its parent are done.

Some applications involve parallel tasks that access and generate large data sets. Data sets in this scale require specialized storage and management systems and data Grid projects are carried out to harness geographically distributed resources for such data-intensive problems by providing remote data set storage, access management, replication services, and data transfer protocols [1]. Data Scheduling could be done without replication or with replication, ensuring in this case the fault tolerance. If replication is considered, there are possible two cases: decoupled computation [15] and data scheduling or integrated computation and data scheduling [16].

The Grid is a large number of autonomous resources, which could be used concurrently, changing dynamically, interacting with different VOs (virtual organizations). In human society and in nature there are systems having the similar characteristics. The Grid Economy approaches and other heuristics inspired by natural phenomena were studied in recent years to address the challenges of Grid computing.

Considering scheduling strategies treating performance dynamism, some of the schedulers provide a rescheduling mechanism, which determines when the current schedule is re-examined and the task executions reordered. The rescheduling taxonomy divides this mechanism in two conceptual mechanisms: periodic/batch and event-driven on line. Periodic or batch mechanism approaches group resource request and system events which are then processed at intervals that may me periodic triggered by certain system events. The other mechanism performs the rescheduling as soon the system receives the resource request. The scheduling policy can be fixed or extensible. The fixed policies are system oriented or application oriented. The extensible policies are ad-hoc or structured. In a fixed approach, the policy implemented by the resource manager is predetermined. Extensible scheduling policy schemes allow external entities the ability to change the scheduling policy [8].

5. Critical Analysis of Near-Optimal Scheduling Algorithms

Optimization methods for decentralized scheduling in Grid environment use heuristic (multi-objective) approaches. We present in this section opportunistic load balancing heuristics, methods that are based on minimum execution time, minimum completion time, min-min, max-min, duplex, genetic algorithms, simulating annealing, A*.

Opportunistic Load Balancing (OLB). The Opportunistic Load Balancing heuristic picks one task arbitrarily from the group of tasks and assigns it to the next machine that is expected to be available. It does not consider the task's expected execution time on that machine, which may lead to very poor maxspans. The advantages of this heuristic are: its simplicity and the intention of keeping all the machines as busy as possible, this meaning high processor utilization. In tasks that come one at a time, rather than in groups of tasks, the Opportunistic Load Balancing heuristic is also named First Come First Served.

Minimum Execution Time (MET). The Minimum Execution Time heuristic assigns each task picked arbitrarily to the machine with the least expected execution time for that task, and is not concerned with the time the machine becomes available [15]. The result can be severe load imbalance across machines, although MET gives each task to its best machine.

Minimum Completion Time (MCT). The Minimum Completion Time heuristic assigns each task, in arbitrary order, to the machine with the minimum expected completion time for that task [16]. The MCT combines the benefits of OLB and MET, and tries to avoid the circumstances in which OLB and MET perform poorly.

Min-Min heuristic begins with the set T of all unmapped tasks. The task with the minimum possible execution time is then assigned on the respective

processor, after which the process continues in the same way with the remaining unmapped tasks. The major difference between Min-min and MCT is that Min-min considers all unmapped tasks during each mapping decision and MCT only considers one task at a time. The machine that finishes the earliest is also the machine that executes the task the fastest. The percentage of tasks assigned to their first choice (on the basis of execution time) is likely to be very high, and therefore a smaller maxspan can be obtained.

Max-Min heuristic is very similar to Min-min. The Max-min heuristic also begins with the set *T* of all unmapped tasks. Then, the set *C* of minimum completion times is found. The difference from Min-min comes at the next step, when the task with the overall maximum completion time from *C* is selected and assigned to the corresponding machine. Last, the newly mapped task is removed from *C*, and the process repeats until C is empty.

Max-min tries to perform tasks with longer execution times first, which usually leads to a better balanced allocation of tasks, and prevents that some processors stay idle for a long time, while others are overloaded.

Duplex heuristic is a combination of the Min-min and Max-min heuristics. The Duplex heuristic performs both of the Min-min and Max-min heuristics and then uses the better solution. Duplex exploits the conditions in which either Min-min or Max-min performs better.

Genetic Algorithms (GA) is technique used for searching large solution spaces. Multiple possible mappings of the meta-task are computed, which are considered chromosomes in the population. Each chromosome has a fitness value, which is the result of an objective function designed in accordance with the performance criteria of the problem (for example makespan). At each iteration, all of the chromosomes in the population are evaluated based on their fitness value, and only the best of them survive in the next population, where new allocations are generated based on crossover and mutation operators. The algorithm usually stops when a predefined number of steps is performed, or all chromosomes converge to the same mapping.

Simulated Annealing (SA) is an iterative technique that considers only one possible solution (mapping) for each meta-task at a time. This solution uses the same representation as the chromosome for the GA. SA uses a procedure that probabilistically allows poorer solutions to be accepted to attempt to obtain a better search of the solution space. This probability is based on a system temperature that decreases for each iteration. As the system temperature decreases, poorer solutions are less likely to be accepted. The initial temperature of the system is the maxspan of the initial mapping, which is randomly determined. At each iteration, the mapping is transformed in the same manner as the GA, and the new maxspan is evaluated. If the new maxspan is better (lower), the new mapping replaces the old one.

A* heuristic is a search technique based on a tree, which has been applied in various task allocation problems. The A* heuristic begins at a root node that is a null solution. As the tree grows, nodes represent partial mappings (a subset of tasks is assigned to machines). With each child added, a new task *T* is mapped. This process continues until a complete mapping is reached.

6. Experimental Comparison of Presented Optimization Scheduling Algorithms

In this section, a comparison between the performances of the scheduling algorithms is presented. Below we present the graphics obtained from running 50 jobs on 6 CPUs.

As we are not interested in memory usage, the trends lines for this do not appear in the graphics. Just by looking at the graphics, we can see that we obtain the best results for both the throughput and the load-balancing when using the genetic algorithm. Average load-balancing (cpu load) can be computed from these graphics using the formula:

$$AVG = \frac{\sum_{k=1}^{N}(t_k - t_{k-1}) \cdot l_k}{\sum_{j=1}^{N}(t_k - t_{k-1})} = \frac{\sum_{k=1}^{N}(t_k - t_{k-1}) \cdot l_k}{t_N - t_0}$$

where l_k is the the CPU load in the time interval $[t_k, t_{k+1}]$.

Table 1

Schedulers comparison (50 jobs)

Performance parameter / Scheduler	Throughput [seconds]	Load-balancing [%]
Simple scheduler	397.1052	70.58
Shortest job first scheduler (SJF)	443.8935	91.10
Earliest deadline first scheduler (EDF)	397.2369	70,55
Genetic scheduler	296.0156	94,68

We can clearly see now, that whether we choose throughput or load-balancing as criterion for performance, the genetic algorithm is the winner:

- Throughput: genetic, simple & EDF (almost a tie), SJF
- Load balancing: genetic, SJF, simple & EDF (again, almost equal).

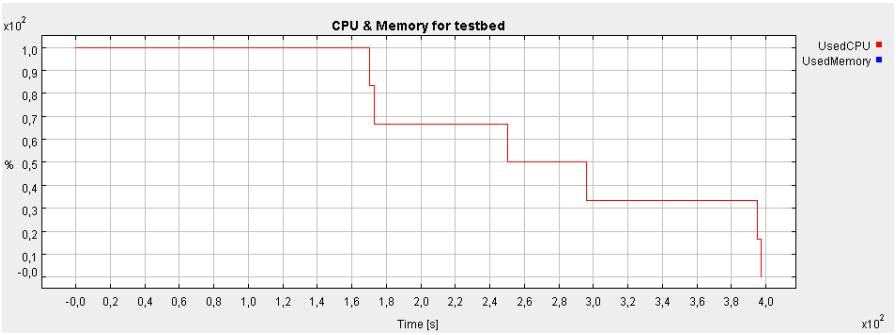

Fig. 2. Simple scheduler (50 jobs)

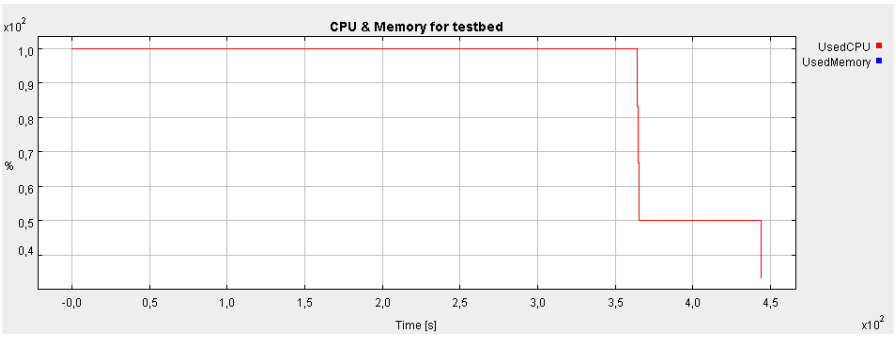

Fig. 3. SJF scheduler (50 jobs)

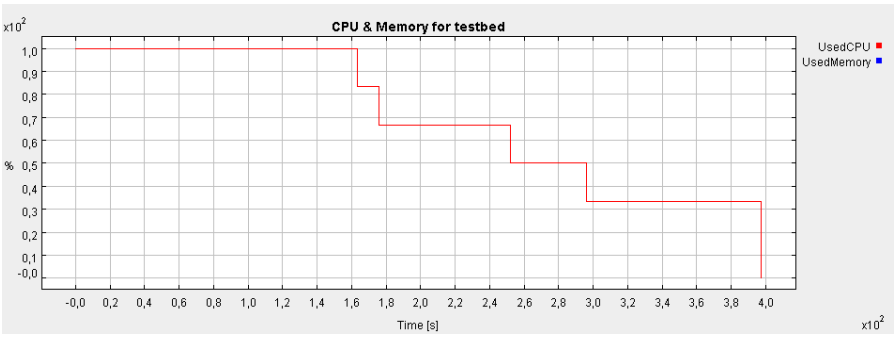

Fig. 4. EDF scheduler (50 jobs)

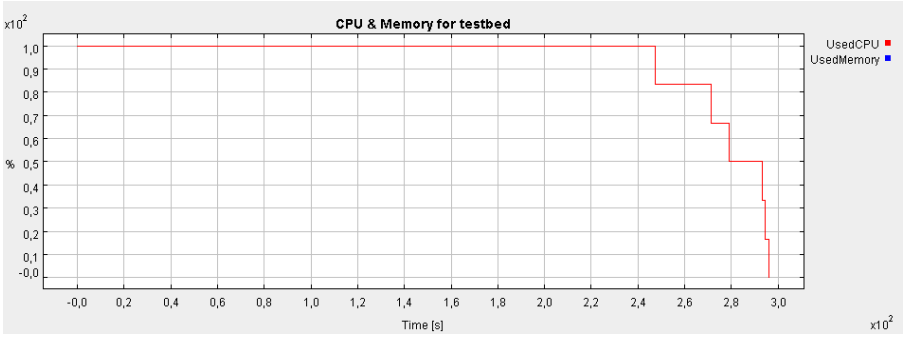

Fig. 5. Genetic scheduler (50 jobs)

7. Conclusions

We present in this paper the process of optimization for decentralized scheduling strategies in Grid environments. The objective was focused on the determination of near-optimal strategies for application scheduling in decentralized environments based on existing strategies.

We elaborated an evaluation model for scheduling algorithms using simulation. The MONARC simulator was extended to support tasks with dependencies and multiple scheduling algorithms. MONARC offers a realistic simulation, so the evaluation model was used to compare different scheduling algorithm for independent and dependent task and to select the best algorithm.

According with experimental results we propose a solution for decentralized scheduling architecture.

Future investigation would involve the extension of the scheduling algorithms towards classes of dependent tasks, as well as the incorporation of new features into the existing frameworks with new optimization criteria. It is possible to extend the scheduling algorithms for task with dependencies.

R E F E R E N C E S

[1] *W. Allcock, J. Bresnahan, R. Kettimuthu, M. Link, C. Dumitrescu, I. Raicu, I. Foster*, The Globus striped gridftp framework and server. In Proceedings of the 2005 ACM/IEEE conference on Supercomputing, Seattle, Washington USA, pages 54-64, 2005

[2] *M. Arora, S.K. Das, R. Biswas,* A decentralized scheduling and load balancing algorithm for heterogeneous grid environments. In Proceedings of International Conference on Parallel Processing Workshops (ICPPW'02), Vancouver, British Columbia Canada, pages 499-505, 2002

[3] *R. Braun, H. Siegel, N. Beck, L. Boloni, M. Maheswaran, A. Reuther, J. Robertson, M. Theys, B. Yao, D. Hensgen, R. Freund*, A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems, Parallel and Distributed Computing, 61(6):810-837, 2001

[4] *R. Buyya, D. Abramson, J. Giddy, H. Stockinger,* Economic models for resource management and scheduling in grid computing. In The Journal of Concurrency and Computation: Practice and Experience (CCPE), Wiley Press, pages 44-53, 2002

[5] *T. Casavant, J. Kuhl,* A taxonomy of scheduling in general-purpose distributed. Computing Systems, in IEEE Transactions on Software Engineering, 14(2):141-154, 1988

[6] *Cătălin Cîrstoiu, Nicolae Ţăpuş*, Framework for data-intensive applications optimization in large-scale distributed Systems, U. P. B. Sci. Bull., Series C, **vol. 71**, Iss. 3, 2009, ISSN 1454-234x

[7] *H. Chen, M. Maheswaran,* Distributed dynamic scheduling of composite tasks on grid computing systems. In Proceedings of the 16th International Parallel and Distributed Processing Symposium (IPDPS 2002), Fort Lauderdale, Florida, USA, pages 88-97, 2002

[8] *E. Deelman, J. Blythe, Y. Gil, C.l Kesselman, G. Mehta, K. Vahi, A. Lazzarini, A. Arbree, R. Cavanaugh, S. Koranda,* Mapping abstract complex work owes onto grid environments. Journal of Grid Computing, 1(1): 9-23, 2003

[9] *I. Foster,* Globus toolkit version 4: Software for service-oriented systems. In IFIP International Conference on Network and Parallel Computing, Springer-Verlag LNCS 3779, pages 2-13, 2005

[10] *James Frey, Todd Tannenbaum, Ian Foster, Miron Livny, Steve Tuecke*, Condor-g: A computation management agent for multi-institutional grids. Cluster Computing, 2006

[11] *Francine Berman, Richard Wolski,* The Apples project - a status report, Technical report, Department of Computer Science and Engineering, University of California, San Diego, 1997

[12] *R.L. Henderson,* Job scheduling under the portable batch system, Proceedings of the JSSPP'95, Lecture Notes in Computer Science, 949:279-294, 1995

[13] *E. Huedo, R. S. Montero, I. M. Llorente*, A Framework for Adaptive Execution in Grids, Software - Practice and Experience 34 (7): 631-651, 2004

[14] *E. Huedo, R.S. Montero, I.M. Llorente,* Experiences on adaptive grid scheduling of parameter sweep applications. In Proceedings of the 12th Euromicro Conference on Parallel, Distributed and Network-Based Processing (PDP'04), pages 28-33, 2004

[15] *K. Ranganathan, I. Foster*, Identifying dynamic replication strategies for a high performance data grid. Proceedings of the 2nd of International Workshop on Grid Computing, Lecture Notes In Computer Science, 2242(1):75-86, 2001

[16] *K. Ranganathan I. Foster*. Decoupling computation and data scheduling in distributed data-intensive applications. In Proceedings of 11th IEEE International Symposium on High Performance Distributed Computing (HPDC-11), pages 352-361, 2002

[17] *G. Sabin, R. Kettimuthu, A. Rajan, P. Sadayappan,* Scheduling of parallel jobs in a heterogeneous multi-site environment. Proceedings of the 9th International Workshop on Job Scheduling Strategies for Parallel Processing, Lecture Notes In Computer Science, 2862, 2003

[18] *H. Shan, L. Oliker, R. Biswas, W. Smith*, Scheduling in heterogeneous grid environments: The effects of data migration. Proceedings of ADCOM2004: International Conference on Advanced Computing and Communication, Ahmedabad Gujarat, India, 2004

[19] *Corina Stratan, Valentin Cristea*, A Framework for Performance Prediction in Distributed Systems, U. P. B. Sci. Bull., Series C, **vol. 71**, Iss. 3, 2009, ISSN 1454-234x

[20] *D. Thain, T. Tannenbaum, M. Livny,* Condor and the grid. Grid Computing: Making The Global Infrastructure a Reality, Fran Berman, Anthony J.G. Hey, Georgey Fox, John Wiley, 2003

[21] *D. Wright,* Cheap cycles from the desktop to the dedicated cluster: Combining opportunistic and dedicated scheduling with condor. Proceedings of Conference on Linux Clusters: the HPC Revolution, Champaign Urbana, IL USA, 2001.