An Adaptive Scheduling Approach in Distributed Systems

Alexandra Olteanu, Florin Pop, Ciprian Dobre, Valentin Cristea

Faculty of Automatic Control and Computers, University, Computer Science Department
University "*Politehnica*" of Bucharest, Romania
Emails: olteanu.alexandra@cti.pub.ro, florin.pop@cs.pub.ro, ciprian.dobre@cs.pub.ro valentin.cristea@cs.pub.ro

***Abstract*—A large number of scheduling algorithms for large scale distributed systems have been proposed, studied and compared. In spite of this, there are few studies comparing the performance of scheduling algorithms considering at the same time the distributed system structure on which we want to schedule the tasks; the type of directed acyclic graph (DAG), in which graph nodes represent tasks and graph edges represent data transfers; and the type of tasks, for example CPU-bound vs. I/O bound. This paper proposes a method for selecting, in a dynamic manner, the most appropriate scheduling algorithm for a particular distributed system, after a previous analysis. Choosing the best known scheduling algorithm can improve performance of an application if all the aspects previously enumerated are considered. The results of this paper consist in a comparison of the scheduling algorithms performance for the following scheduling algorithms: Modified Critical Path, Cluster ready Children First, Earliest Time First, Highest Level First with Estimated Times and Hybrid Remapper Minimum Partial Completion Time Static Priority. However, the main purpose of this investigation tests is to demonstrate the usefulness of the presented scheduling approach using these results.**

***Index Terms*—Grid Scheduling, Communication to Computation Ratio, Dynamic Scheduling, Adaptive Scheduling, Simulation, System Resources Taxonomy.**

I. Introduction

Due to the NP-complete nature of scheduling problems, the quality of schedule solutions produced by existing scheduling heuristics cannot be guaranteed. In distributed systems, scheduling is the process of allocating a set of resources to tasks or jobs aiming to achieve performance objectives satisfying certain constraints. When considering a system of resources, a programmer must decide which scheduling algorithm will perform best depending on the type of the system usage. Unfortunately, there is no universal best scheduling algorithm. Selecting the best known scheduling algorithm can improve performance of an application if one considers aspects such as the distributed system structure on which we want to schedule the tasks, the type of directed acyclic graph (DAG) and the type of tasks.

This paper proposes a method for selecting, in a dynamic, automated and adaptive manner, the most appropriate scheduling algorithm for a particular distributed system, after a previous analysis. This approach can bring significant improvements to the application execution, as scheduling algorithms base their decisions on different methods to calculate the allocation of resources to tasks, prioritizing in various ways the costs.

Similar approaches proposed by the research community like [1] describe different criteria for evaluation of algorithms, considering a different way to classify applications: concurrent, pipelined and parallel. In a classical approach, it is common for a single scheduling algorithm to be presented, which is compared with a set of well known scheduling algorithms, using a set of specific experimental tests, usually regarding just one type of resources, homogeneous or heterogeneous [2].

The mere fact that usually authors do not consider the whole range of types of systems that can occur within a Grid can turn to the assumption that the algorithms proposed by them cannot offer the best results in all cases. To address the aforementioned shortcomings, our approach not only provides a set of class association rules, where the class is represented by the scheduling algorithm, but also builds a decision tree based on which it manages to take a decision in a small amount of time and with minimum information.

The paper is organized in the following way: Section II presents the related work of adaptive scheduling methods in distributed systems. In Section III we describe the developing method for adaptive scheduling algorithms. Section IV presents the distributed system classification. Section V highlights the design and implementation for MONARC II and in Section VI the experimental methodology and results are presented.

II. Related Work

Zhang et al consider in [3] that finding effective scheduling algorithms for this problem is a challenging research area and many scheduling algorithms have been proposed, studied and compared on heterogeneous parallel computers but there are few studies comparing the performance of scheduling algorithms in Grid environments. They compared several scheduling algorithms that represent two classes of schedulers used for Grid computing and analyzed the results to explain how different resource environments and workflow application structures affect the performance of these algorithms. In this paper it was introduced a new measurement was introduced, called effective aggregated computing power (EACP) that could drastically improve the performance of some schedulers [4].

For scheduling parallel loops, Banicescu et al [5] proposed an adaptive weighted factoring (AWF), a method focused to relax the time-stepping requirement, a modification that allows the AWF to be used in any application with a parallel loop. The modification further allows the AWF to adapt to load imbalance that may occur during loop execution. Results of experiments to compare the performance of the modified AWF with the performance of the other loop scheduling methods in the context of three nontrivial applications reveal that the performance of the modified method is comparable to, and in some cases, superior to the performance of the most recently introduced adaptive factoring method.

Rashid et al investigate in [6] the performance of a dynamic loop scheduling with reinforcement learning (DLS-with-RL) approach to load balancing. The DLS-with-RL is most suitable for use in time-stepping scientific applications with large number of steps. The RL agent's characteristics depend on a learning rate parameter and a discount factor parameter. The advantages of integrating reinforcement learning (RL) techniques into scientific parallel time-stepping applications have been revealed in research work and the object of the integration was to automatically select the most appropriate dynamic loop scheduling (DLS) algorithm from a set of available algorithms with the purpose of improving the application performance via load balancing during the application execution.

Scheduling mechanisms for Grid environment use different methods: Random and Best of n Random, Exhaustive Search, Simulated Annealing [8], Game Theory, Ad-Hoc Greedy, Genetic Algorithms, method for task dependencies. Optimization of Distributed Scheduling is based on some theoretical methods for local or global optimization (Multidimensional Optimization Methods): Simplex Algorithm, Simulated Annealing, Genetic Algorithms, Clustering Methods, and Heuristic Methods.

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. Also, genetic algorithms are considering in [9].

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 [15][10][14]. It does not consider the task's expected execution time on that machine, which may lead to very poor maxspans [11].

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 [12]. The result can be severe load imbalance across machines, although MET gives each task to its best machine.

The Minimum Completion Time heuristic assigns each task, in arbitrary order, to the machine with the minimum expected completion time for that task [12]. The MCT combines the benefits of OLB and MET, and tries to avoid the circumstances in which OLB and MET perform poorly.

III. Adaptive Scheduling in Distributed Systems

In real-life scenarios, a scheduling algorithm needs to address a number of issues. It should exploit parallelism by identifying the task graph structure, and take into consideration task granularity (amount of computation with respect to communication), arbitrary computation and communication costs. Furthermore, we should also consider the system homogeneity computed based on standard deviation of common resources (CPU Power, memory and bandwidth), which shows, in simple terms, how much variation from the average exists. How the process of allocating tasks to resources is made should be adapted depending on a set of parameters that characterize both the resources and the application and have an important influence on the scheduling performance, which is reflected in the application execution time performance.

A. Used terms

For defining this set of parameters, next, a number of terms that will be used for analysis are introduced.

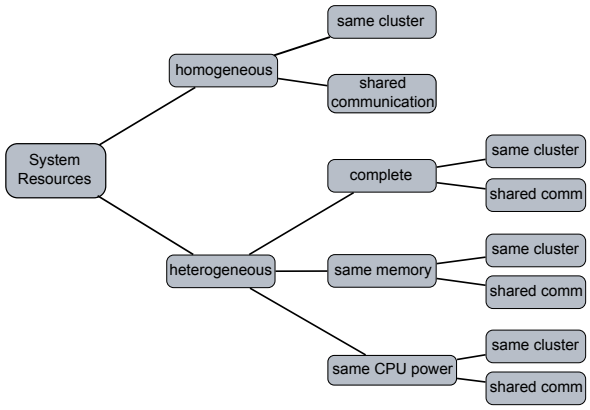

Fig. 1. Resources Taxonomy for Distributed Systems

- **Granularity**
  Depending on its granularity, which is a measure of the communication to computation ratio, a DAG can be coarse grained (the computation dominates the communication) or fine grained (the communication dominates the computation). Granularity of a DAG is defined as:

$$g(G) = min(cn(x)/max(c(x,i))), \quad (1)$$

  where $cn(x)$ - is the computation cost of node x
  $c(x,j)$ - the communication costs from node x to node j
  x, j - represents the nodes and can take values from 1 to the number of nodes

  We conclude that:
  - I/O bound term is equivalent with fine-grained
  - CPU-bound term is equivalent with coarse-grained

  Intuitively a graph is coarse-grained if the amount of computation is relatively large with respect to communication.

- **CCR(Communication to Computation Ratio)**
  Definitions found in the literature usually assume CCR defined as the average edge weight divided by the average node weight. With the help of CCR, one can judge the importance of communication in a task graph, which strongly determines the scheduling behavior. Based on CCR we classify task graphs in:
  - CCR < 1 - coarse grained graph
  - CCR $= 1$ - mixed
  - CCR > 1 - fine grained graph
- **RCCR (Resources Communication to Computation Ratio)**
  Resources may be also analyzed taking into account the RCCR (Resources Communication to Computation Ratio). RCCR is a resource system parameter similar to CCR. While CCR represents the necessary Communication to Computation Ratio, RCCR represents the available Communication to Computation Ratio.
- **Heterogeneity**
  Numerous heuristics have been proposed for scheduling DAGs both for heterogeneous and for homogeneous computing environments. Consequently, some scheduling algorithms work better on one environment type than another. We determined the system heterogeneity or homogeneity using the standard deviation function.
- **Communication medium**
  Another important aspect that should be considered when scheduling DAGs is the available amount of communication bandwidth.

The next section describes how distributed systems are classified for our further analysis using these concepts.

B. Distributed System Classification

Our approach aims to take into consideration all of the enumerated terms, and to find realistic system taxonomy based on their definitions and the brief descriptions given above. Furthermore, the term system or distributed system parameters refers to both the system resources parameters and the application parameters, because all factors that can influence scheduling performance are to be highlighted. Since the application characteristics have an important impact on scheduling performance, especially due to the dependencies existence along the tasks, our first classification groups applications by tasks type, using CCR and task granularity, as shown in Figure 2.

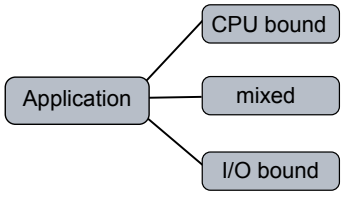

Fig. 2. Application Taxonomy for Distributed Computing

In addition, the resource characteristics should also be taken into account and Figure 1 highlights the system resources taxonomy, which considers the last two terms previously presented in Section 3. As shown, heterogeneous systems are divided into three categories described by the calculated values of standard deviation functions for CPU power and memory. If the values of standard deviation of both CPU power and memory overpass the preselected thresholds, the system is considered completely heterogeneous. On the other hand, if only one value is satisfying this condition, the system is considered to be heterogeneous due to a single parameter value, CPU power or memory. In addition, another observation can be made about the classification according to the communication environment: resources are coupled via a dedicated or shared communications medium.

Lastly, we make one more classification considering the RCCR parameter, which will be correlated in our tests with the CCR parameter (required vs. available resources communication to computation ratio). This classification also divides the system in another three categories depending on the difference from the value 1: lower, higher or approximately equal.

Taking all this into consideration our entire classification resulted in a total of 72 types of systems that can occur in a Grid.

C. System analyzer

Furthermore, we designed a method for dynamically selecting the most appropriate scheduling algorithm for a particular distributed system considering all the elements described below. Our method consists of three major steps:

- Step 1: gathering data about the system, using a system monitor, and starting system analysis
- Step 2: analyzing the system from three points of view:
  - analyze system resources structure - here we consider the RCCR parameter and the resources heterogeneity (classification is based on the resources taxonomy)
  - analyze the DAG type (application parameters) - for this we use the CCR parameter
  - analyze the DAG tasks type - this represents task granularity, but in this version we included this analysis in the DAG type analysis
- Step 3: choosing the best algorithm considering the analysis results

These steps are also presented in Figure 3.

This approach takes into account that, in general cases, scheduling algorithms are designed considering a limited number of system types, so we can assume that there is no universal best in terms of scheduling algorithms.

IV. Design and Implementation for MONARC II

MONARC II is a large scale distributed systems simulator, designed for modeling and simulation of distributed systems, aimed at predicting performance parameters of specific applications. 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

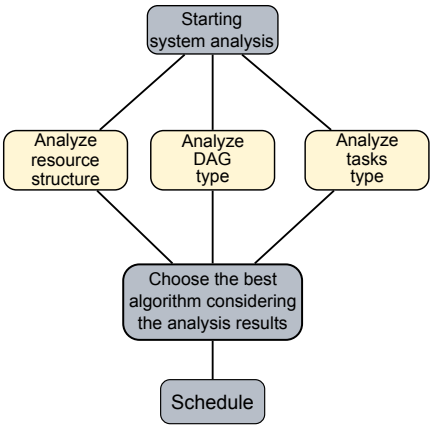

Fig. 3. System Analyzer Diagram

as all the stochastic arrival patterns, specific for such type of simulation [16].

In order to provide a realistic simulation, all the components of the system and their interactions were abstracted. The chosen model is equivalent to the simulated system in all the important aspects. A first set of components was created for describing the physical resources of the distributed system under simulation. The largest one is the regional center, which contains a farm of processing nodes (CPU units), database servers and mass storage units, as well as one or more local and wide area networks. Another set of components model the behavior of the applications and their interaction with users.

The job is another basic component, simulated with the aid of an active object, and scheduled for execution on a CPU unit by a "Job Scheduler" object. Any regional center can dynamically instantiate a set of users or activity objects, which are used to generate data processing jobs based on different simulation scenarios. Inside a regional center different job scheduling policies may be used to distribute jobs to corresponding processing nodes [17].

In order to model the new scheduling concept previously presented, the MONARC simulator had been extended with simulation components for system analysis as highlighted in Figure 4.

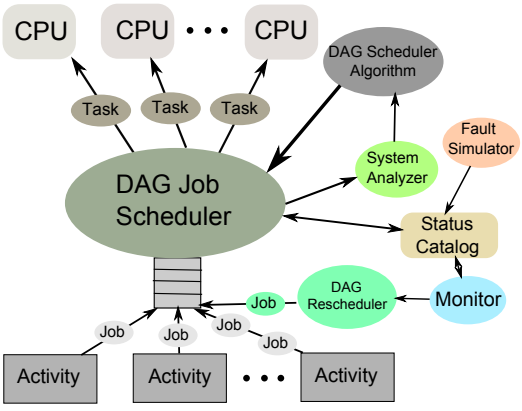

Fig. 4. Extended MONARC II Architecture

The simulation model used by MONARC allows the construction of various 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 default scheduler behavior does not consider the topology of the underlying resources for scheduling assignments or the costs of the edges as in case of the algorithms presented in Section 2. Instead, the scheduler uses a highly simplistic algorithm that first prioritizes the tasks such that to eliminate all possible loops and then schedules each task to be executed according to its own local requirements, without considering the minimization of the execution times among them. We extended the default behavior to allow for more realistic scheduling decisions that consider both the underlying topology (input parameter for the DAG job scheduler algorithm), as well as the full topology of tasks dependencies.

The default behavior of the simulator does not consider the selection of an algorithm (the most appropriate scheduling algorithm) from a set of scheduling algorithms, which are known by the system. Therefore, we extended the default behavior of the simulator so that it can simulate a selection mechanism based on a particular analysis. This analysis takes into account all the elements that we have presented in the previous section: resource structure, DAG type, task type.

The System Analyzer Component is responsible both for current system analysis and for providing an algorithm selection method. Therefore, it is composed of two parts: system monitor and system analyzer. The system monitor receives all necessary data and computes them to determine a series of parameters. It maps the first two steps of the presented method. The last step is mapped by the system analyzer which surveys the set of parameters and decides on the scheduling algorithm to be used.

V. Experimental Methodology

A. DAG Generation

For the majority of tests we used a graph generator to obtain the application DAGs. This generator builds the graph, level after level. For each level, excluding the first level, it looks for a number of parents for each node, number that is lower than a predefined threshold for links complexity, in the above level. Graph construction takes into account the parameters defined in the configuration file (number of nodes, number of task levels, processing power, links complexity, communication costs) presented below:

```
# TaskGen configuration file
# Number of tasks
dioTasksNumber = 100
# Number of task levels
dioTaskLevels = 10
# Links complexity
dioLinksComplexity = 4
# processing time
```

```
dioMinProcessingTime = 9
dioMaxProcessingTime = 90
# communication costs
dioMinCommunicationCost = 90
dioMaxCommunicationCost = 195
```

In addition, in some cases where extreme results were obtained with this type of DAG, balanced DAGs and DAGs with LU parallel algorithm pattern were also used. However, for the performance obtained for scheduling algorithms on synthetic randomly generated DAGs is given less significance due to the DAG shape and type of tasks in DAG. In general a scientific workflow application has a uniquely shaped DAG due to its design, which is made to accomplish a complex task by means of job parallelism. The DAGs of many real world workflow applications are well balanced and highly parallel. This is the motivation for choosing this kind of graphs for these special cases, but this paper purpose is to validate the presented adaptive approach, not necessarily to give a classification of scheduling algorithms.

B. Costs

For costs analysis CCR (Communication to Computation Ratio) parameter is used:

$$CCR = \frac{\sum_{x,j} c(x,j) * number\_of\_nodes}{\sum_{x} cn(x) * number\_of\_edges} \quad (2)$$

where $cn(x)$ - is the computation cost of node x, $c(x,j)$ - the communication costs from node x to node j, and x, j - represents the node's number and can take values from 1 to the number of nodes.

C. Resources availability

Furthermore, for resources analysis three important parameters were considered:

1) Heterogeneity: In order to decide if a system is homogeneous or heterogeneous the standard deviation formula for CPU power and memory is used:

$$pd = \sqrt{\frac{1}{N}\sum_{x}(p(x) - \overline{p})} \quad (3)$$

where $\overline{p}$ - average links cost, $p(x)$ - the communication costs from node x to node j, and x - represents the node's number and can take values from 1 to the number of nodes.

$$md = \sqrt{\frac{1}{N}\sum_{x}(m(x) - \overline{m})} \quad (4)$$

where $\overline{m}$ - average links cost, $m(x)$ - the communication costs from node x to node j, and x - represents the node's number and can take values from 1 to the number of nodes.

2) Communication Medium: For this parameter the standard deviation to decide if there is a shared or a dedicated communication medium is also used:

$$cd = \sqrt{\frac{1}{N}\sum_{x,j}(c(x,j) - \overline{c})} \quad (5)$$

where $\overline{c}$ - average links cost, $c(x,j)$ - the communication costs from node x to node j, and x, j - represents the node's number and can take values from 1 to the number of nodes.

3) RCCR(Resources Communication to Computation Ratio):

$$RCCR = \frac{\sum_{x,j} cr(x,j) * number\_of\_CPUs}{\sum_{x} cnr(x) * number\_of\_links} \quad (6)$$

where $cnr(x)$ - is the available computation for CPU x, $cr(x,j)$ - the available communication from CPU x to CPU j, x, j - represents the node's number and can take values from 1 to the number of available CPU.

D. Analyzed Scheduling Algorithms

I. MCP (Modified Critical Path) - algorithm based on lists with two phases: the prioritization and selection of resources. Parameter used to prioritize nodes is ALAP (As Late As Possible)

II. CCF (Cluster ready Children First) dynamic scheduling algorithm based on lists. The graph is visited in topological order, and tasks are submitted as soon as scheduling decisions are taken. The algorithm assumes that when a task is submitted for execution it is inserted into the RUNNING-QUEUE. If a task is extracted from the RUNNING-QUEUE, all its successors are inserted into the CHILDREN-QUEUE. The running ends when the two queues are empty.

III. ETF (Earliest Time First) - algorithm based on keeping the processors as busy as possible. It computes, at each step, the earliest start times of all ready nodes and selects the one with the shortest start time.

IV. HLFET (Highest Level First with Estimated Times) - uses a hybrid of the list-based and level-based strategy. The algorithm schedules a task to a processor that allows the earliest start time.

V. Hybrid Remapper PS (Hybrid Remapper Minimum Partial Completion Time Static Priority) is a dynamic list scheduling algorithm specifically designed for heterogeneous environments. The set of tasks is partitioned into blocks so that tasks in a block do not have any data dependencies among them. Subsequently the blocks are executed one by one.

VI. Experimental Results and Interpretations

In order to demonstrate the efficiency of the process of selecting the most suitable scheduling algorithm for a given system we run a number of 450 tests, that use different resource configuration types and application DAGs with different CCR values. After analyzing data gathered from the comparison of the scheduling algorithms in terms of performance we decided which algorithm is recommended for a particular distributed system. Besides this recommendations, the results also serve for demonstrating the efficiency of this adaptive scheduling

approach. All tests have been performed using the MONARC II simulator, which provides a useful, low cost testing tool compared with a real system.

In the test process a series of relevant configuration files to describe different types of system resources were designed and correlated with input files containing descriptions of DAGs with different CCR values. All these resulted in 30 configuration files and 12 input files, which were used for testing and comparing a set of 5 scheduling algorithms,presented in Section V.

Nr	Res				Comm		CCR			RCCR		
	0	1	2	3	sh	cl	$<$	$=$	$>$	$<$	$=$	$>$
1	*	-	-	-	-	*	*	-	-	-	*	-
2	*	-	-	-	-	*	-	*	-	-	*	-
3	*	-	-	-	-	*	-	-	*	-	*	-
4	*	-	-	-	*	-	-	*	-	*	-	-
5	-	*	-	-	-	*	*	-	-	*	-	-
6	-	*	-	-	-	*	-	*	-	*	-	-
7	-	*	-	-	*	-	-	*	-	*	-	-
8	-	*	-	-	*	-	*	-	-	-	-	*
9	-	*	-	-	*	-	-	*	-	-	-	*
10	-	*	-	-	*	-	-	-	*	-	-	*
11	-	-	*	-	*	-	-	*	-	*	-	-
12	-	-	*	-	-	*	-	*	-	*	-	-
13	-	*	-	-	-	*	-	*	-	-	*	-
14	-	*	-	-	*	-	-	*	-	-	*	-
15	-	*	-	-	-	*	*	-	-	-	-	*
16	-	*	-	-	-	*	-	*	-	-	-	*
17	-	-	*	-	-	*	-	*	-	*	-	-
18	-	-	*	-	*	-	-	*	-	*	-	-
19	-	-	*	-	-	*	-	*	-	-	*	-
20	-	-	*	-	*	-	-	*	-	-	*	-
21	-	-	*	-	-	*	-	*	-	-	-	*
22	-	-	*	-	-	*	*	-	-	-	-	*
23	-	-	*	-	*	-	-	*	-	-	-	*
24	-	-	*	-	*	-	*	-	-	-	-	*
25	-	-	-	*	-	*	*	-	-	*	-	-
26	-	-	-	*	-	*	-	*	-	*	-	-
27	-	-	-	*	*	-	*	-	-	*	-	-
28	-	-	-	*	*	-	-	*	-	*	-	-
29	-	-	-	*	-	*	*	-	-	-	*	-
30	-	-	-	*	-	*	-	*	-	-	*	-
31	-	-	-	*	*	-	*	-	-	-	*	-
32	-	-	-	*	*	-	-	*	-	-	*	-
33	-	-	-	*	-	*	*	-	-	-	-	*
34	-	-	-	*	-	*	-	*	-	-	-	*
35	-	-	-	*	*	-	*	-	-	-	-	*
36	-	-	-	*	*	-	-	*	-	-	-	*
35	*	-	-	-	-	*	-	*	-	*	-	-
35	*	-	-	-	*	-	-	*	-	-	*	-
35	*	-	-	-	-	*	-	*	-	-	-	*
35	*	-	-	-	*	-	-	*	-	-	-	*

TABLE I
EXPERIMENTAL RESULTS FOR CCF (CLUSTER READY CHILDREN FIRST) ALGORITHM

The results obtained on leveled DAGs with a preselected maximum link complexity, presented in the three tables below, highlight which scheduling algorithm performs best for each system type, in the case of the tests that have been run. Each one of the tables is correlated with one of the scheduling algorithms that have proven to offer the lowest execution time for at least one system type. When determining the system patterns for which an algorithm offers the best results, these patterns can be analyzed as belonging to a class, which is defined by the algorithm itself.

Therefore, TABLE I is for the CCF algorithm, TABLE II for the HLFET algorithm and TABLE III for the Hybrid Remapper algorithm. Furthermore, the tables have been analyzed in order to find the set of association rules for each class, algorithm. One must take into account that the results presented in the tables were obtained only for leveled DAGs with preselected maximum link complexity.

The following notations were used:

- Res: 0 - homogeneous system; 1 - completely heterogeneous; 2 - heterogeneous with the same memory; 3 - heterogeneous with the same CPU power;
- Comm: sh - shared communication medium; cl - same cluster (dedicated communication medium);
- CCR - Communication to Computation Ratio;
- RCCR - Resources Communication to Computation Ratio;
- "*" means that the system satisfies the value of the parameter;
- "-" the system does not satisfy the value of the parameter.

It can be noted that for most system types that our tests have covered, the CCF algorithm obtained the best results out of the set of available algorithms. Several system patterns that determine the use of the CCF scheduling algorithm, can be observed in TABLE I:

- homogeneous, $CCR \approx 1$
- completely heterogeneous, $CCR \approx 1$
- heterogeneous with the same CPU power, $CCR \approx 1$ or $CCR < 1$
- heterogeneous with the same CPU power, $CCR \approx 1$

The second scheduling algorithm, for which our results have proven that it offers the best results for seven system types, is the HLFET algorithm. Two of the system patterns which establish the use of this algorithm can be observed in TABLE II:

- homogeneous, $CCR > 1$, $RCCR < 1$
- completely heterogeneous, $CCR > 1$, $RCCR < 1$

Nr	Res				Comm		CCR			RCCR		
	0	1	2	3	sh	cl	$<$	$=$	$>$	$<$	$=$	$>$
1	*	-	-	-	*	-	-	-	*	*	-	-
2	-	*	-	-	-	*	-	-	*	*	-	-
3	-	*	-	-	*	-	-	-	*	*	-	-
4	*	-	-	-	-	*	-	-	*	*	-	-
5	*	-	-	-	-	*	-	-	*	-	*	-
6	*	-	-	-	-	*	-	-	*	*	-	-
7	*	-	-	-	*	-	-	-	*	-	*	-

TABLE II
EXPERIMENTAL RESULTS FOR HLFET (HIGHEST LEVEL FIRST WITH ESTIMATED TIMES) ALGORITHM

Hybrid Remmaper algorithm obtained the best results for the system types presented in TABLE III. Some of the obtained system patterns are:

- shared communication medium, $CCR < 1$, $RCCR < 1$
- homogeneous, $CCR < 1$
- heterogeneous with the same CPU power, $CCR > 1$

Nr	Res				Comm		CCR			RCCR		
	0	1	2	3	sh	cl	$<$	$=$	$>$	$<$	$=$	$>$
1	*	-	-	-	*	-	*	-	-	*	-	-
2	-	*	-	-	*	-	*	-	-	*	-	-
3	-	-	*	-	*	-	*	-	-	*	-	-
4	-	-	*	-	*	-	-	-	*	*	-	-
5	-	-	*	-	-	*	*	-	-	*	-	-
6	-	-	*	-	-	*	-	-	*	*	-	-
7	-	*	-	-	-	*	*	-	-	-	*	-
8	-	*	-	-	-	*	-	-	*	-	*	-
9	-	*	-	-	*	-	*	-	-	-	*	-
10	-	*	-	-	*	-	-	-	*	-	*	-
11	-	*	-	-	-	*	-	-	*	-	-	*
12	-	-	*	-	-	*	-	-	*	-	*	-
13	-	-	*	-	*	-	-	-	*	-	*	-
14	-	-	*	-	-	*	-	-	*	-	-	*
15	-	-	*	-	*	-	-	-	*	-	-	*
16	-	-	-	*	-	*	-	-	*	*	-	-
17	-	-	-	*	*	-	-	-	*	*	-	-
18	-	-	-	*	-	*	-	-	*	-	*	-
19	-	-	-	*	*	-	-	-	*	-	*	-
20	-	-	-	*	-	*	-	-	*	-	-	*
21	-	-	-	*	*	-	-	-	*	-	-	*
22	*	-	-	-	-	*	*	-	-	-	-	*
23	*	-	-	-	*	-	*	-	-	-	-	*
24	*	-	-	-	-	*	*	-	-	-	-	*
25	*	-	-	-	-	*	-	-	*	-	-	*
26	*	-	-	-	*	-	*	-	-	-	-	*
27	*	-	-	-	*	-	-	-	*	-	-	*

TABLE III
EXPERIMENTAL RESULTS FOR HYBRID REMAPPER PS (HYBRID REMAPPER MINIMUM PARTIAL COMPLETION TIME STATIC PRIORITY) ALGORITHM

There are two more cases for which ETF obtained the best results and their pattern is:

- heterogeneous with the same memory, $CCR < 1$, $RCCR \approx 1$

After comparing the resulted patterns we discovered the most powerful class association rules that help us to decide in a small amount of time and with minimum information what class, scheduling algorithm, should be chosen for one case or another:

- CCF algorithm - $CCR \approx 1$
- CCF algorithm - heterogeneous with the same CPU power, $CCR < 1$
- HLFET algorithm - $CCR > 1$, $RCCR < 1$, homogeneous or completely heterogeneous
- Hybrid Remapper algorithm - homogeneous, $CCR < 1$
- ETF algorithm - heterogeneous with the same memory, $CCR < 1$, $RCCR \approx 1$

The results obtained and presented within this paper are credited to the classifications which have been made. Please keep in mind that these results were obtained on leveled DAGs with a preselected maximum link complexity. Using these results, faster execution times of up to 30% can be achieved.

The charts in Figures 5 and 6 present the performance improvements between the scheduling algorithm that obtained the best time and the one that obtained the worst time, respectively the one that obtained the second worst time. The second chart was made to strengthen the conclusions about the effectiveness of this approach.

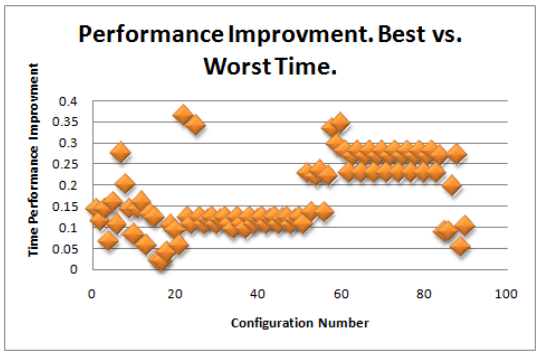

Fig. 5. Performance Improvment. Best vs. Worst time for each system type that was tested

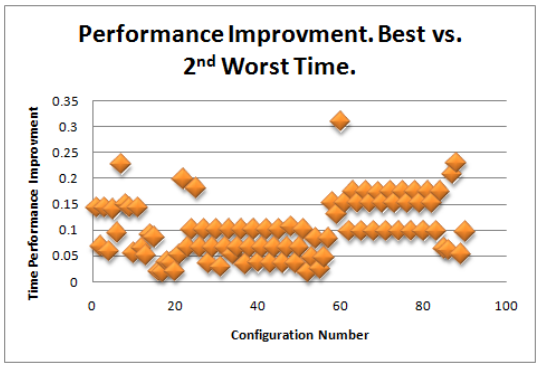

Fig. 6. Performance Improvment. Best vs. 2^{nd} Worst time for each system type that was tested

Can be easily observed that the majority of values are found between 0.05 and 0.3 in the first chart, Figure 5, and between 0.0 and 0.2 in the second chart, Figure 6. This means that the average performance improvement that can be obtained is found in these intervals. Moreover for the extreme value obtained in chart shown in Figure 6 for configuration number 60, we performed a series of tests for which we used balanced DAGs and DAGs generated using LU parallel algorithm pattern, besides leveled DAGs with a preselected maximum link complexity . The results of this tests, for which the configuration file describe a heterogeneous resource system with dedicated communication medium and the input files describe different DAGs with a CCR value closed to 1, are shown in the Table IV. In this table we made the notations: DAG type 1 - leveled DAGs with a preselected maximum link complexity; DAG type 2 - balanced DAGs; DAG type 3 - DAGs generated using LU parallel algorithm pattern; BTSA - the scheduling algorithm that obtained the best time for the current test.

Nr	DAG type	CCR	BTSA	2^{nd} BTSA
1	1	1.24	Hybrid Remapper	HLFET
2	1	1.09	Hybrid Remapper	HLFET
3	2	1.09	CCF	HLFET
4	2	1.31	CCF	HLFET
5	3	1.09	HLFET	Hybrid Remapper
6	3	1.08	HLFET	Hybrid Remapper

TABLE IV
PERFORMANCE RESULTS FOR TESTS ON DIFFERENT KINDS OF DAGS STRUCTURES

This proves that the DAGs structure is also important when choosing an appropriate scheduling algorithm for a system, since we obtained different results depending on the DAG structure.

VII. CONCLUSION AND FUTURE WORK

In this work, we have compared the performance of several algorithms that represent alternative approaches to scheduling a large set of different Grid environments and applications. Our experiments show how the performance of the scheduling algorithms can be affected by different factors in a Grid computing environment. Furthermore, we found some system patterns, different sets of factors that indicate which algorithm should use for a given system configuration. This confirmed our expectations that since scheduling algorithms are designed considering a limited number of system types they will provide good results only in those cases.

Our investigation covers all types of systems that we have obtained through the presented classification, but we observed that the DAG's structure should be also considered when selecting a scheduling algorithm. Therefore our future work will focus on how DAG's structure influence the performance of scheduling algorithms. We also intend to use an adequate data mining heuristic, a supervised learning algorithm, to generate the class association rules, the proper algorithm for each set of system characteristics, which satisfy a minimum support and a minimum confidence. Using this we will obtain a better performance for the scheduling algorithm selection component. This should have an important impact on the implementation of this approach for rescheduling, even in terms of error recovering or performance improvement. Therefore, we plan to use this approach also for rescheduling.

To summarize, we have proved the efficiency of our proposed adaptive scheduling approach for large scale distributed systems and obtained some interesting results such as the behavior of the Hybrid Remapper algorithm. It offered the best results for homogeneous environments although it was designed especially for heterogeneous environments.

ACKNOWLEDGMENTS

The research presented in this paper is supported by national project "DEPSYS - Models and Techniques for ensuring reliability, safety, availability and security of Large Scale Distributes Systems", Project "CNCSIS-IDEI" ID: 1710.

REFERENCES

[1] Ravi Chidansh Hema, *A Grid system with different scheduling strategies and dynamically choosing an appropriate scheduling strategy*, Topics in Grid Computing, September 16, 2004.

[2] Kwok, Y. and Ahmad, I. 1999. Benchmarking and comparison of the task graph scheduling algorithms. *J. Parallel Distrib. Comput. 59*, 3 (Dec. 1999), 381-422. DOI=http://dx.doi.org/10.1006/jpdc.1999.1578

[3] Zhang, Y., Koelbel, C., and Kennedy, K. 2007. Relative Performance of Scheduling Algorithms in Grid Environments. In *Proceedings of the Seventh IEEE international Symposium on Cluster Computing and the Grid (May 14 - 17, 2007)*. CCGRID. IEEE Computer Society, Washington, DC, 521-528. DOI= http://dx.doi.org/10.1109/CCGRID.2007.94

[4] Radulescu, A. and van Gemund, A. J. 1999. On the complexity of list scheduling algorithms for distributed-memory systems. In *Proceedings of the 13th international Conference on Supercomputing (Rhodes, Greece, June 20 - 25, 1999)*. ICS '99. ACM, New York, NY, 68-75. DOI= http://doi.acm.org/10.1145/305138.305162

[5] Carino, R. L. and Banicescu, I. 2008. Dynamic load balancing with adaptive factoring methods in scientific applications. *J. Supercomput. 44*, 1 (Apr. 2008), 41-63. DOI=http://dx.doi.org/10.1007/s11227-007-0148-y

[6] Rashid, M., Banicescu, I., and Carino, R. L. 2008. Investigating a Dynamic Loop Scheduling with Reinforcement Learning Approach to Load Balancing in Scientific Applications. In *Proceedings of the 2008 international Symposium on Parallel and Distributed Computing (July 01 - 05, 2008)*. ISPDC. IEEE Computer Society, Washington, DC, 123-130. DOI= http://dx.doi.org/10.1109/ISPDC.2008.25.

[7] Wieczorek, M., Hoheisel, A., and Prodan, R. 2009. Towards a general model of the multi-criteria workflow scheduling on the grid. *Future Gener. Comput. Syst. 25*, 3 (Mar. 2009), 237-256. DOI = http://dx.doi.org/10.1016/j.future.2008.09.002.

[8] Aziz, A. and El-Rewini, H. 2008. On the use of meta-heuristics to increase the efficiency of online grid workflow scheduling algorithms. *Cluster Computing 11, 4 (Dec. 2008)*, 373-390. DOI=http://dx.doi.org/10.1007/s10586-008-0062-y

[9] F. Pop, Communication model for decentralized meta-scheduling in Grid Environments, Proceedings of The Second International Conference on Complex, Intelligent and Software Intensive System, Second International Workshop on P2P, Parallel, Grid and Internet computing - 3PGIC-2008 (CISIS'08), pp. 315-320, March 4-7, 2008, Barcelona, Spain, Published by IEEE Computer Society, ISBN: 0-7695-3109-1.

[10] Braun, T. D., Siegel, H. J., Maciejewski, A. A., and Hong, Y. 2008. Static resource allocation for heterogeneous computing environments with tasks having dependencies, priorities, deadlines, and multiple versions. *J. Parallel Distrib. Comput.* 68, 11 (Nov. 2008), 1504-1516. DOI= http://dx.doi.org/10.1016/j.jpdc.2008.06.006

[11] Tao Wang; Xing-she Zhou; Qiu-rang Liu; Zhi-yi Yang; Yun-lan Wang, An Adaptive Resource Scheduling Algorithm for Computational Grid, Services Computing, 2006. APSCC apos;06. IEEE Asia-Pacific Conference, Dec. 2006 Page(s):447 - 450

[12] Foster, I., Karonis, N. T., Kesselman, C., and Tuecke, S. 1998. Managing security in high-performance distributed computations. *Cluster Computing* 1, 1 (Jan. 1998), 95-107.

[13] Kim, K. W., Yun, Y., Yoon, J. M., Gen, M., and Yamazaki, G. 2005. Hybrid genetic algorithm with adaptive abilities for resource-constrained multiple project scheduling. *Comput. Ind.* 56, 2 (Feb. 2005), 143-160. DOI= http://dx.doi.org/10.1016/j.compind.2004.06.006

[14] Stensland, H. K., Griwodz, C., and Halvorsen, P. 2008. Evaluation of multi-core scheduling mechanisms for heterogeneous processing architectures. In *Proceedings of the 18th international Workshop on Network and Operating Systems Support For Digital Audio and Video (Braunschweig, Germany, May 28 - 30, 2008)*. NOSSDAV '08. ACM, New York, NY, 33-38. DOI= http://doi.acm.org/10.1145/1496046.1496054

[15] Iordache, G. V., Boboila, M. S., Pop, F., Stratan, C., and Cristea, V. 2007. A decentralized strategy for genetic scheduling in heterogeneous environments. *Multiagent Grid Syst.* 3, 4 (Dec. 2007), 355-367.

[16] Dobre, C.M., C. Stratan, "MONARC Simulation Framework", Proc. of the 3rd Edition of RoEduNet International Conference, Timisoara, Romania, 2004.

[17] Dobre, C.M., V. Cristea, "A Simulation Model for Large Scale Distributed Systems", Proc. of the 4th International Conference on Innovations in Information Technology, Dubai, United Arab Emirates, November 2007.