An adaptive scheduling approach in distributed systems

Olteanu Alexandra
University of "Polytechnics" Bucharest
olteanu.alexandra@gmail.com

***Abstract*—Choosing a good scheduling approach is an important issue for the performance of an application launched onto a distributed systems environment. Many scheduling algorithms have been proposed, studied and compared, but 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. Choosing the best known scheduling algorithm can improve performance of an application if all the aspects previous enumerated are considered. This paper wants to propose a method for choosing, in a dynamic manner, the most appropriate scheduling algorithm for a particular distributed system, which is analyzed.**

***Index Terms*—Grid scheduling; Communication to Computation Ratio; scheduling algorithms; adaptive scheduling; 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. Scheduling is the process of allocating a set of resources to tasks or jobs to achieve certain performance objectives satisfying certain constraints. When considering a system, a programmer must decide which scheduling algorithm will perform best for the use system is going to see. Unfortunately, there is no universal best scheduling algorithm. Choosing the best known scheduling algorithm can improve performance of an application if aspects like the distributed system structure on which we want to schedule the tasks, the type of directed acyclic graph (DAG) and the type of tasks are considered.

This paper proposes a method for choosing, in a dynamic and automatic manner, the most appropriate scheduling algorithm for a particular distributed system, which is analyzed. This approach can bring significant improvements for the application execution, because scheduling algorithms take decisions considering different methods to calculate the allocation of resources to tasks, prioritizing in various ways the costs.

Similar approaches proposed by research community like [3] describe different criteria for evaluation of algorithms, considering a different way to classify applications: concurrent, pipelined and parallel. In classic approaches, usually is presented only one new scheduling algorithm, 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.

II. A new scheduling approach

In realistic cases, a scheduling algorithm needs to address a number of issues. It should exploit the 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 which can be determinate using the standard deviation of resources CPU power, memory and bandwidth (in simple terms, it shows how much variation there is from the "average") and system resources structure.

A. Used terms

Therefore, we will introduce some terms that will help us in our analysis:

- **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 will classify graph tasks considering:
  - CCR < 1 - coarse grained graph
  - CCR $= 1$ - mixed
  - CCR > 1 - fine grained graph
- **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:

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

  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

  So we can conclude:
  - I/O bound term is equivalent with fine-grained
  - CPU-bound term is equivalent with coarse-grained

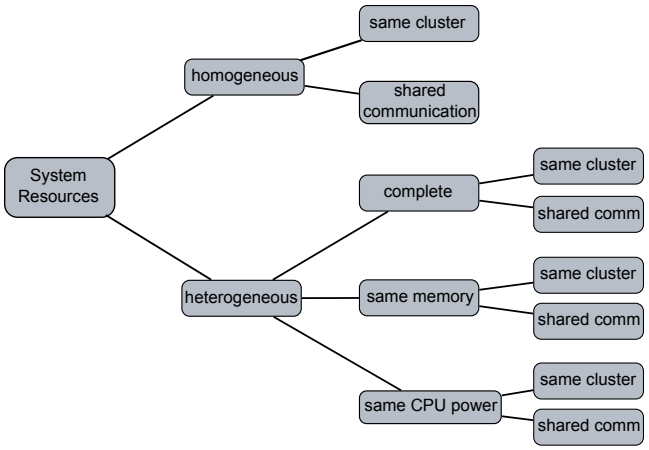

Fig. 1. System resources taxonomy

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

- **RCCR (Resources Communication to Computation Ratio)**
  Resources can be also analyzed considering a parameter very similar with CCR, but while CCR represent the needed Communication to Computation Ratio, this parameter represent the available Communication to Computation Ratio.
- **Heterogeneity**
  Numerous heuristics have been proposed for scheduling DAGs both for heterogeneous and for homogenous computing environment. Consequently, some scheduling algorithm 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.

B. System classification

Our approach aim is to take into consideration all of the enumerated terms, and based on their definitions and the brief descriptions given above to find a realistic system taxonomy. We first classified the applications based on tasks type, using CCR and task granularity. This classification can be seen below in Fig.2:

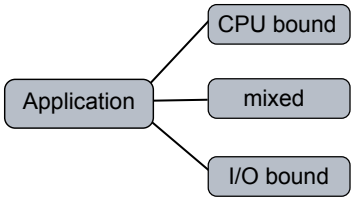

Fig. 2. Application taxonomy

Fig.1, highlights the system resources taxonomy, which consider the last two terms previously presented. It can be observed that heterogeneous systems are divided into three categories described by the calculated values of standard deviation functions for CPU power and memory. If both values are quite high compared to zero, the system is considered complete heterogeneous. On the other hand, if only one value is satisfying this consideration, the system is considered to be heterogeneous because of a single value, CPU power or memory. In addition, another observation can be made about the classification according to communication environment: resources are coupled via a dedicated or shared communications medium.

Lastly we should make one more classification considering RCCR parameter, which will be correlated in our tests with CCR parameter (needed 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 equal.

C. System analyzer

Furthermore, we designed a method for choosing, in a dynamic manner, the most appropriate scheduling algorithm for a particular distributed system considering all this elements. Our method is composed from three major steps:

- Step 1: gathering the data about the system, using a system monitor, and starting the system analysis
- Step 2: analyzing the system from three point of view:
  - analyze system resources structure - here we consider RCCR parameter and the resources heterogeneity (classification is based on the resources taxonomy)
  - analyze DAG type (application parameters) - for this we use CCR parameter
  - analyze DAG tasks type - this represent the 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

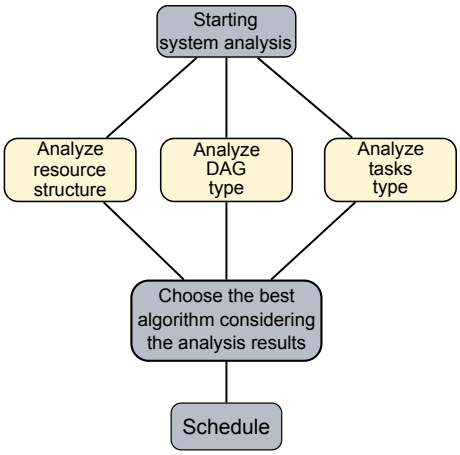

Fig. 3. System analyzer diagram

These steps are also presented in the below figure:

This approach take 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.

III. Design and Implementation for MONARC II

The MONARC II is a simulator for large scale distributed systems, having as a purpose the modeling and simulation of distributed systems, with the goal of predicting general performances of the applications running on these systems [10]. MONARC is built based on a process oriented approach for discrete event simulation, which is well suited to describe concurrent running programs, network traffic [8].

For modeling the new scheduling concept previously presented, the MONARC simulator had been extended with simulation components for analyzing the system, and this is highlighted in the figure Fig.4. First, 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 an analysis, which 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 implemented not only for analyzing the current system, but also to provide 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 it is mapped by the system analyzer which analyze the set of parameters and decide what scheduling algorithm should be used.

IV. Experimental Methodology

A. DAG generation

To obtain the application DAGs we use a graph generator. This generator builds the graph, level after level. For each level, excluding level 0, it is looking for a number of parents for each node, in the above level.In graph construction are taken 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

B. Costs

For the costs analysis we use CCR (Communication to Computation Ratio) parameter:

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

where $cn(x)$ - is the computation cost of node x

$c(x,j)$ - the communication costs from node x to node j

x, j - represents the nodes number and can take values from 1 to the number of nodes

C. Resources availability

For resources analysis we consider three important parameters:

1) Heterogeneity: To decide if a system is homogeneous or heterogeneous we use the standard deviation formula for CPU power and memory:

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

where $\overline{p}$ - average links cost

$p(x)$ - the communication costs from node x to node j

x - represents the nodes 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

x - represents the nodes number and can take values from 1 to the number of nodes

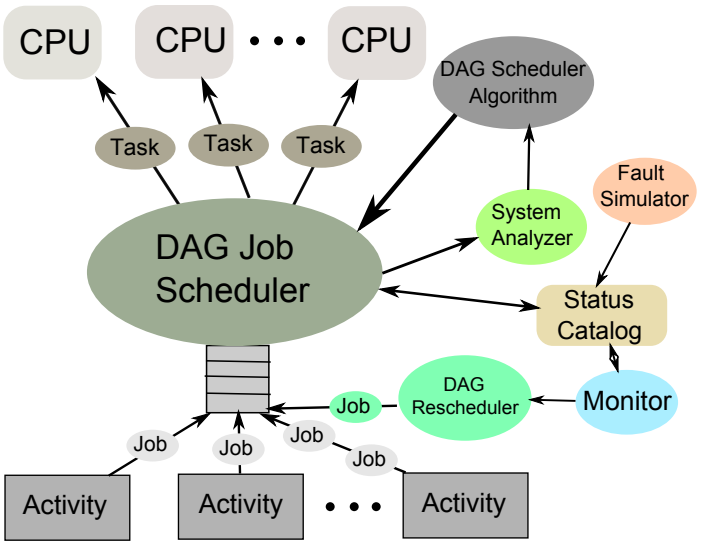

Fig. 4. MONARC II arhitecture

2) Communication medium: For this parameter we also use the standard deviation to decide if we have a shared or a dedicated communication medium:

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

where $\bar{c}$ - average links cost

$c(x,j)$ - the communication costs from node x to node j

x, j - represents the nodes number and can take values from 1 to the number of nodes

3) RCCR(Resources Communication to Computation Ratio):

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

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 nodes number and can take values from 1 to the number of available CPU

D. Used 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. In this algorithm the graph is visited in topological order, and tasks are submitted as soon as scheduling decisions are taken. The algorithm considers 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 having the smallest start time.

IV. HLFET (Highest Level First with Estimated Times) - use 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 the tasks in a block do not have any data dependencies among them. Then the blocks are executed one by one

V. Experimental results. Interpretation

For taking the best decision about choosing the best scheduling algorithm for a given system we made a number of 165 tests. After analyzing the data gathered from the comparison of the scheduling algorithms performance we decided which algorithm from our scheduling algorithms set is recommended for a particular distributed system. All our tests were done only using the MONARC II simulator. It provides a low cost testing tool compared with a real system. Therefore, it is a very useful tool.

For testing we tried to design a series of relevant configuration files to describe different types of system resources. Correlated with this, we also create input files with different CCR values. All this resulted in 7 configuration files and 7 input files, used for testing and comparing all known algorithms.

Our decisions are based on the results presented in three tables that can be seen below. Each table is correlated with one of the algorithms that have proven to offer the best scheduling

for at least one system type. As a result TABLE I is for CCF algorithm, TABLE II for HLFET algorithm and TABLE III for Hybrid Remapper algorithm. Therefore, we analyze the tables and try to find a system pattern for each algorithm.

For the tables we made the following notations:

- Res: 0 - homogeneous system; 1 - complete 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

Can be noted that for most system types that our tests have covered, from the set of available algorithms, CCF algorithm obtained the best results. The system patterns, that we have observed in TABLE I, are:

- homogeneous, same cluster, $RCCR \approx 1$
- complete heterogeneous, shared communication medium, $RCCR > 1$
- heterogeneous with the same memory, $CCR \approx 1$, $RCCR < 1$
- complete heterogeneous, $CCR \approx 1$, $RCCR < 1$
- shared communication medium, $CCR \approx 1$, $RCCR < 1$
- complete heterogeneous, same cluster, $RCCR < 1$

Nr	Res				Comm		CCR			RCCR		
	0	1	2	3	sh	cl	$<$	$=$	$>$	$<$	$=$	$>$
1	*	-	-	-	-	*	*	-	-	-	*	-
2	*	-	-	-	-	*	-	*	-	-	*	-
3	*	-	-	-	-	*	-	-	*	-	*	-
4	*	-	-	-	*	-	-	*	-	*	-	-
5	-	*	-	-	-	*	*	-	-	*	-	-
6	-	*	-	-	-	*	-	*	-	*	-	-
7	-	*	-	-	*	-	-	*	-	*	-	-
8	-	*	-	-	*	-	*	-	-	-	-	*
9	-	*	-	-	*	-	-	*	-	-	-	*
10	-	*	-	-	*	-	-	-	*	-	-	*
11	-	-	*	-	*	-	-	*	-	*	-	-
12	-	-	*	-	-	*	-	*	-	*	-	-

TABLE I

The second scheduling algorithm, for which our results have proven that it offers the best results for three system types, is HLFET algorithm. The system pattern that we have observed in TABLE II is:

- $CCR > 1$, $RCCR < 1$

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

TABLE II

Last, but not least, is Hybrid Remmaper algorithm, which obtained the best results for six system types, TABLE III. The obtained system patterns are:

- shared communication medium, $CCR < 1$, $RCCR < 1$
- heterogeneous with the same memory, same cluster, $RCCR < 1$
- shared communication medium, heterogeneous with the same memory, $RCCR < 1$

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

TABLE III

After we have compared the obtained patterns we discovered the most powerful parameters associations that decide what scheduling algorithm should be used for one case or another:

- CCF algorithm - $CCR \approx 1$, $RCCR < 1$
- CCF algorithm - heterogeneous with the same memory, $CCR \approx 1$, $RCCR < 1$
- HLFET algorithm - $CCR > 1$, $RCCR < 1$
- Hybrid Remapper algorithm - heterogeneous with the same memory, $CCR > 1$, $RCCR < 1$
- Hybrid Remapper algorithm - heterogeneous with the same memory, $CCR < 1$, $RCCR < 1$

Using these results we obtained faster execution times by up to 12%.

VI. Conclusion and Future Work

In this work, we have compared the performance of several algorithms that represent alternative major approaches to scheduling a large set of different Grid environments and applications. Our experiments also show how the performance of the scheduling algorithms can be affected by different factors in a Grid computing environment.

We find some system patterns, different sets of factors that indicate us which algorithm we should use. This confirmed our expectations, because scheduling algorithms are designed considering a limited number of system types they will provide good results just on these cases.

This research covers only a part of the types of system that we have obtained through our classification. Therefore our future work will cover all cases that remained untested. After finishing the analysis on all system types, we intend to implement this new approach on real systems.

As a last conclusion, we demonstrated the efficiency of our proposed adaptive scheduling approach for distributed systems and obtained some interesting results such as the behavior of Hybrid Remapper algorithm, which gave the best results for homogeneous environments although it was designed especially for heterogeneous environments.

REFERENCES

[1] Yu-Kwong Kwok and Ishfaq Ahmad, *Benchmarking and Comparison of the Task Graph Scheduling Algorithms*,Journal of Parallel and Distributed Computing, March 17, 1999.

[2] Yang Zhang, Charles Koelbel and Ken Kennedy, *Relative Performance of Scheduling Algorithms in Grid Environments*, Houston, Rice University.

[3] Ravi Chidansh Hema, *A Grid system with different scheduling strategies and dynamically choosing an appropriate scheduling strategy* , 6 September 2004.

[4] Andrei Radulescu and Arjan J.C. van Gemund, *On the Complexity of List Scheduling Algorithms for Distributed-Memory Systems*,The Netherlands: Delft University of Technology.

[5] Zhifeng Yu and Weisong Shi, *An Adaptive Rescheduling Strategy for GridWorkflow Applications*, Wayne State University.

[6] Ishfaq Ahmad, Yu-Kwong Kwok, Min-You Wu and Wei Shu, *Automatic Parallelization and Scheduling of Programs on Multiprocessors using CASCH*.

[7] Alberto Forti, *DAG Scheduling for Grid Computing systems*, PhD thesis, University of Udine - Italy, 2006.

[8] Florin Pop, *Optimization of Decentralized Schedulind Strategies in Grid Environ- ments*, PhD thesis, University of "Polytechnics" Bucharest, 2008.

[9] Alexandra Olteanu, *Reschedulind and Error Recovering Algorithm for Grid Environments*, License thesis, University of "Polytechnics" Bucharest, 2009.

[10] MONARC II, *http://monarc.cacr.caltech.edu*, Accessed 30 June, 2009.