2010 IEEE INTERNATIONAL CONFERENCE ON INTELLIGENT COMPUTER COMMUNICATION AND PROCESSING

An Adaptive Scheduling Approach in Distributed Systems

Alexandra Olteanu, Florin Pop, Ciprian Dobre, Valentin Cristea

Emails: olteanu.alexandra@cti.pub.ro, florin.pop@cs.pub.ro, ciprian.dobre@cs.pub.ro valentin.cristea@cs.pub.ro

University "*Politehnica*" of Bucharest
Faculty of Automatic Control and Computer Science

Computer Science & Engineering Department

26.08.2010

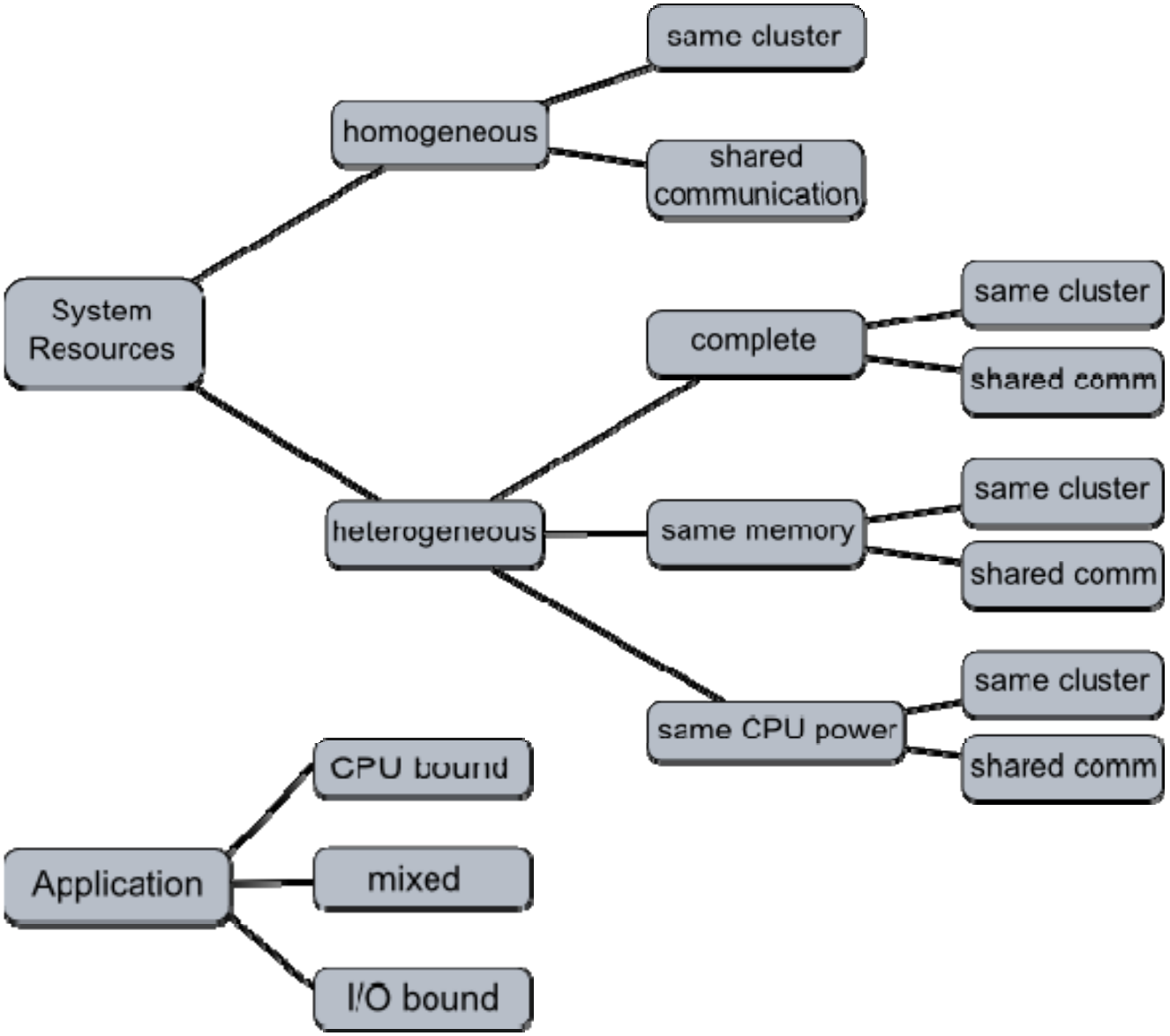
Scheduling problem, Environment, Objective

- Scheduling Problem
  - Allocate a group of different tasks on resources
  - NP-Complete (algorithm)
  - Dynamic and adaptive approach
- Description of Scheduling Environment
  - Computational GRID, Dependent set of tasks (DAG)
  - Heterogeneous and Public
- Objectives
  - Successful completion of tasks
  - Improve performance of an application
- Exploit parallelism by identifying:
  - the task graph structure, task granularity (amount of computation with respect to communication), arbitrary computation and communication costs, resources parameters

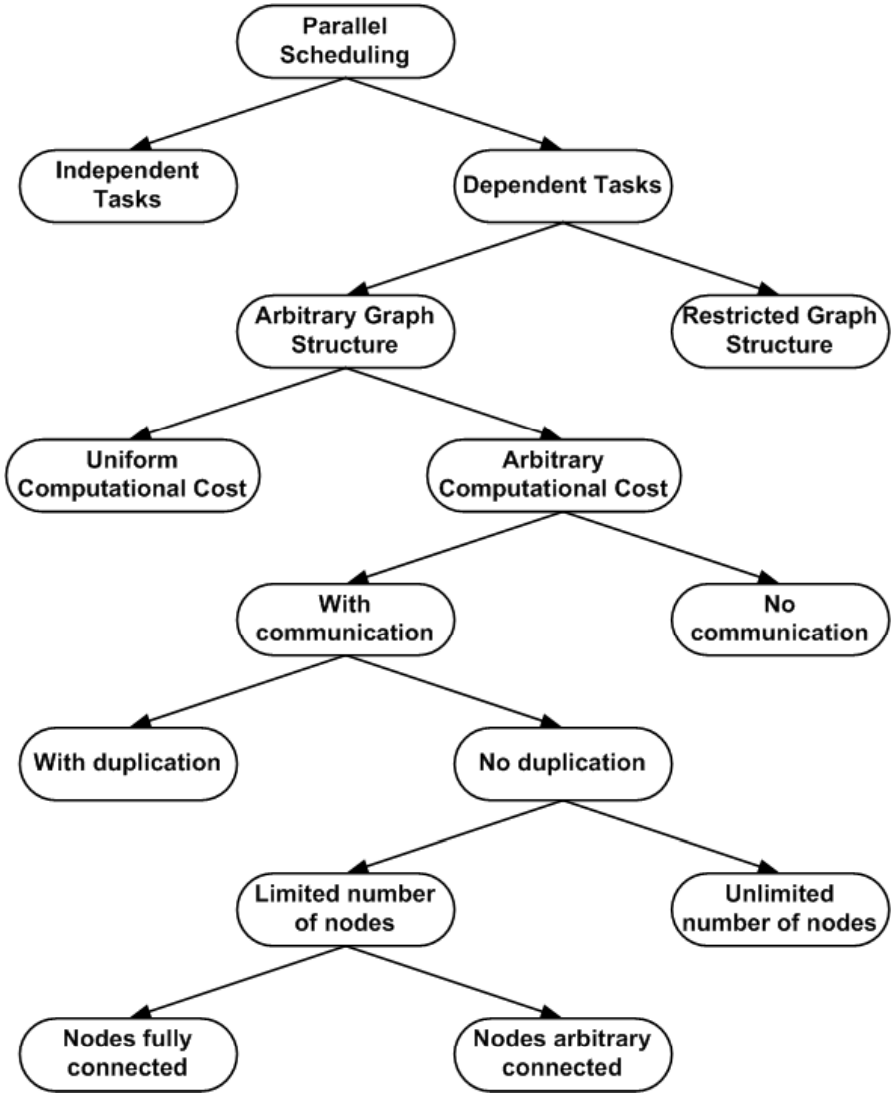
Outline

- Resources and Applications Taxonomy for Distributed Systems
- DAG Tasks Scheduling
  - Example of DAG Task
  - Metrics for Adaptive Scheduling in Distributed Systems
- Proposed System Analyzer
- Extended MONARC II Architecture
- Experimental methodology
  - DAG Generation
  - Costs
  - Resources availability
  - Analyzed Scheduling Algorithms
- Conclusion and Future Work

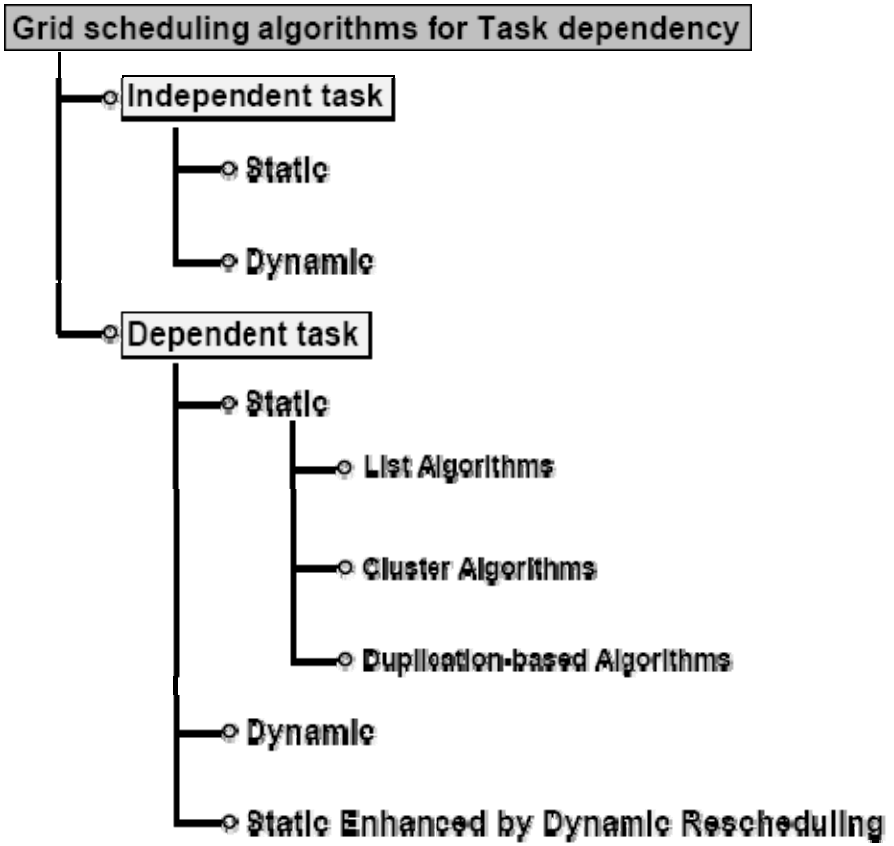
Resources and Applications Taxonomy



DAG Tasks Scheduling



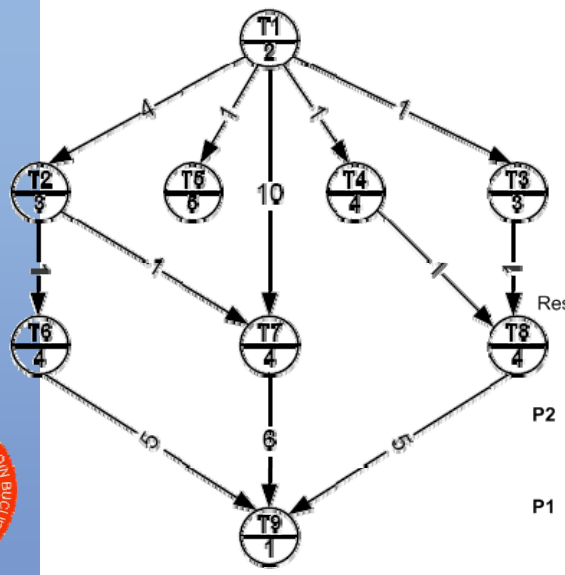
A taxonomy of the DAG scheduling problem

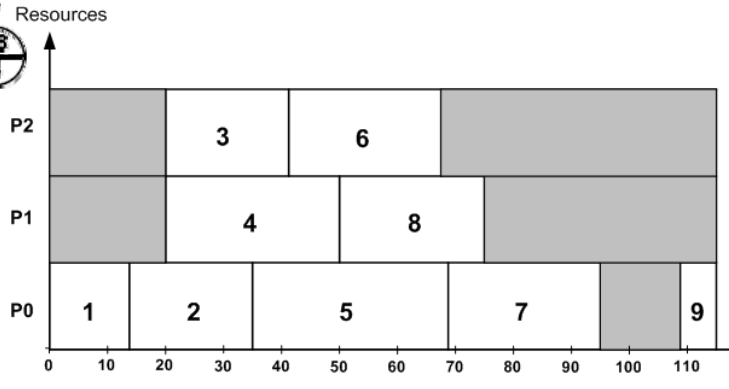


Taxonomy of task dependency scheduling algorithms in Grid environments

DAG Tasks Scheduling (Example)

- Scheduling algorithms based on priorities:
  - *tlevel* (top-level) - the weight of the longest path from the source node (the first scheduled task) to a node *u*
  - *blevel* (bottom-level) - the weight of the longest path from a node *u* to an exit node (the latest scheduled task).
- Critical path (*CP*) - the longest path in a graph.
- *ALAP* (As Late As Possible): $ALAP(u) = CP - blevel(u)$





Node	tlevel	blevel	ALAP
0	0	23	0
1	0	23	0
2	6	15	8
3	3	14	9
4	3	15	8
5	3	5	18
6	10	10	13
7	12	11	12
8	8	10	13
9	22	1	22

Metrics for Adaptive Scheduling

- **Granularity** is a measure of the communication to computation ratio

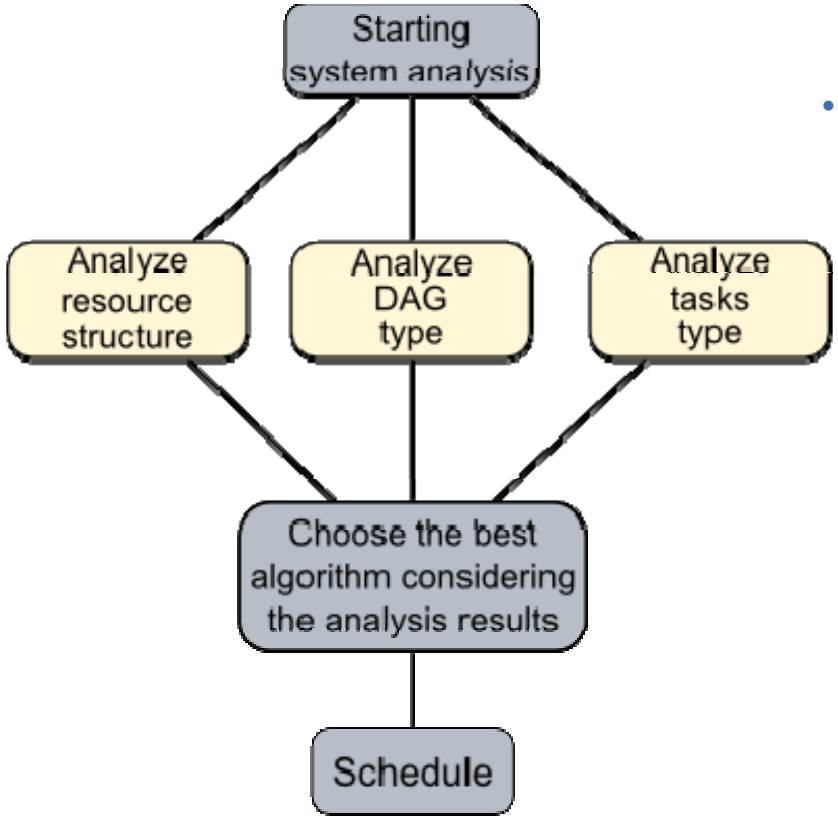
$$g(G) = \frac{\min\{cn(x)\}}{\max\{c(x,i)\}}$$

- DAG can be:
  - *coarse grained* (the computation dominates the communication)
  - *fine grained* (the communication dominates the computation)

- **CCR (Communication to Computation Ratio)**
  - CCR < 1 - coarse grained graph
  - CCR = 1 - mixed
  - CCR > 1 - fine grained graph

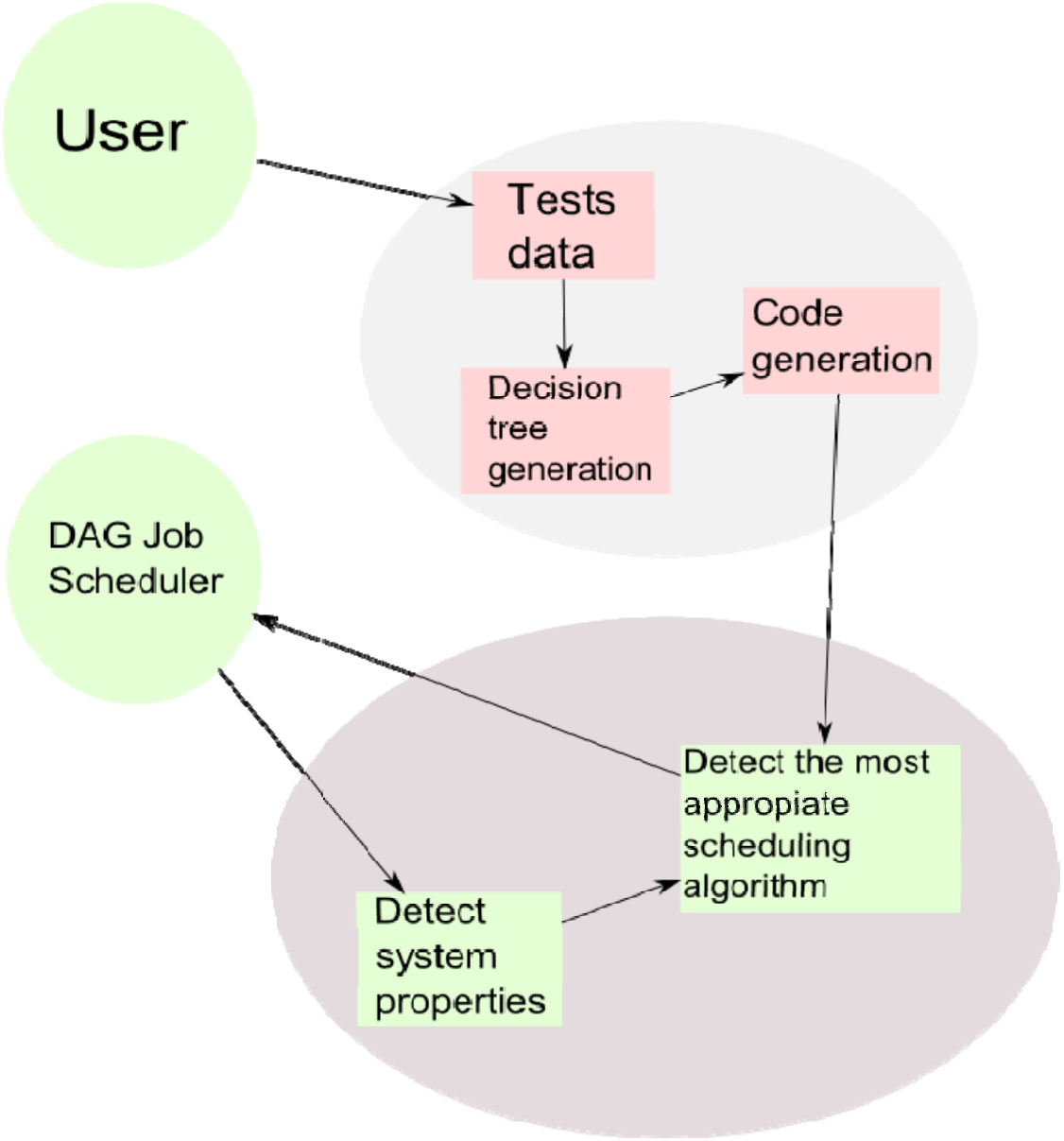
- **RCCR (Resources Communication to Computation Ratio)**
  - CCR represents the necessary ratio for communication to computation
  - RCCR represents the available ration for communication to computation

Proposed System Analyzer

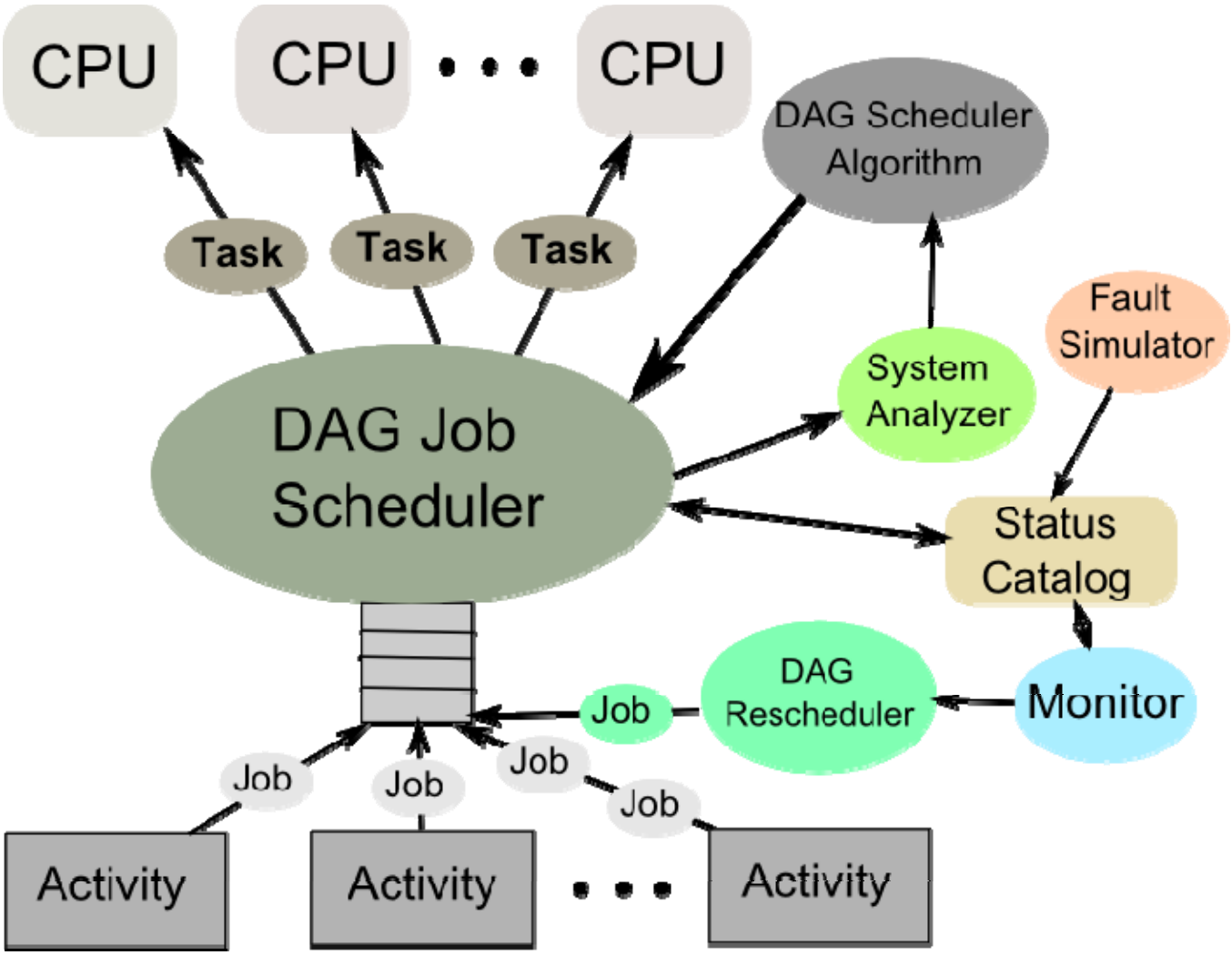


- **Step 1**: gathering data about the system, using a system monitor, and starting system analysis
- **Step 2**: analyzing the system from three points of view:
  - analyze system resources structure - here we consider the RCCR parameter and the resources heterogeneity (classification is based on the 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

System Analyzer Architecture



Extended MONARC II Architecture



Experimental Methodology

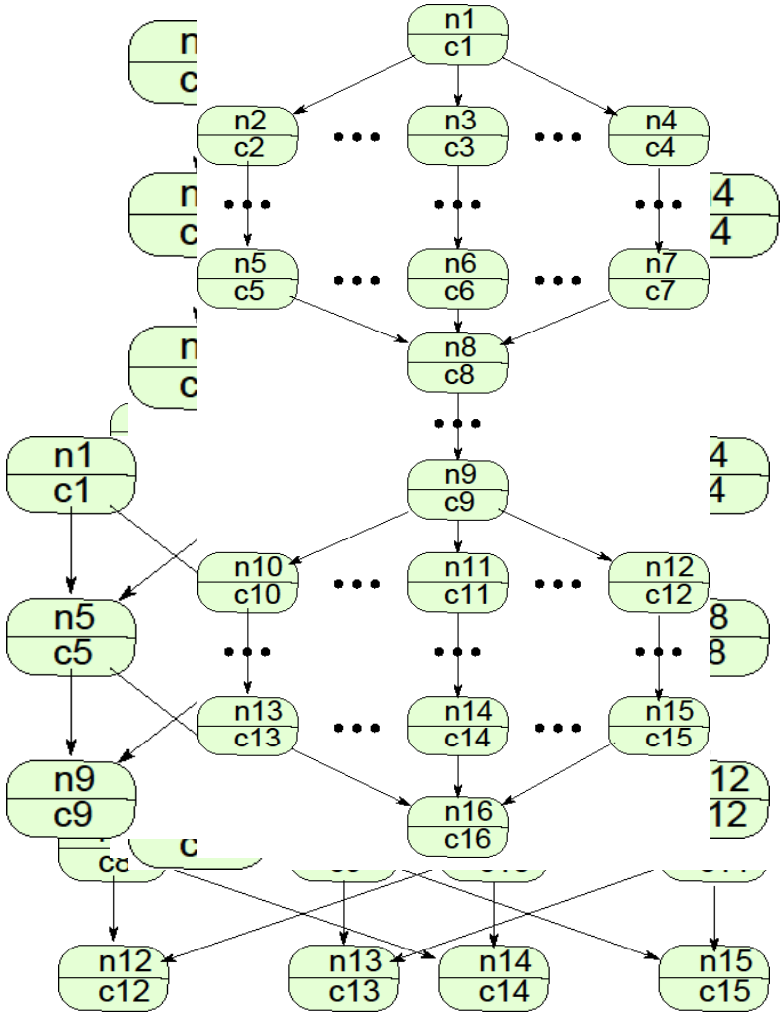
- **A. DAG Generation**
  - DAGs graph generator.
  - This generator builds the graph, level after level. For each level, excluding level 0, it is looking for a number of parents for each node, in the above level.

Configuration file for DAGs graph generator

```
# 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
```

DAGs generation

- **Random**
- **Leveled** with link complexity
- **Balanced** → DAGs of many real world workflow applications
- **Laplace** → differential equation solving
- **Stencil** → linear and non-linear image processing, seismic reconstruction
- **FFT** → signal and image processing, electro-acoustic music and audio signal processing, medical imaging, image processing, pattern recognition, computational chemistry
- **LU** → solving linear equations, systems, matrix inversion, FPGA programming



Experimental Methodology

- **B. Costs**
- For the costs analysis we use CCR (Communication to Computation Ratio) parameter:

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

- 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.

Experimental Methodology

- **C. Resources availability**

- 1) Heterogeneity

$$pd = \sqrt{\frac{1}{N}\sum_{x}(p(x) - \overline{p})^2}$$

$$md = \sqrt{\frac{1}{N}\sum_{x}(m(x) - \overline{m})^2}$$

- *pd* – standard deviation for CPU power
- *md* – standard deviation for CPU memory

- 2) Communication Medium

$$cd = \sqrt{\frac{1}{N}\sum_{x,j}(c(x,j) - \overline{c})^2}$$

- *cd* – standard deviation for communication costs

Experimental Methodology

- **C. Resources availability**

- 3) RCCR (Resources Communication to Computation Ratio)

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

- 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.

Experimental Methodology

- **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. 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.

Experimental Methodology

- **D. Analyzed Scheduling Algorithms**

- 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

Experimental Results

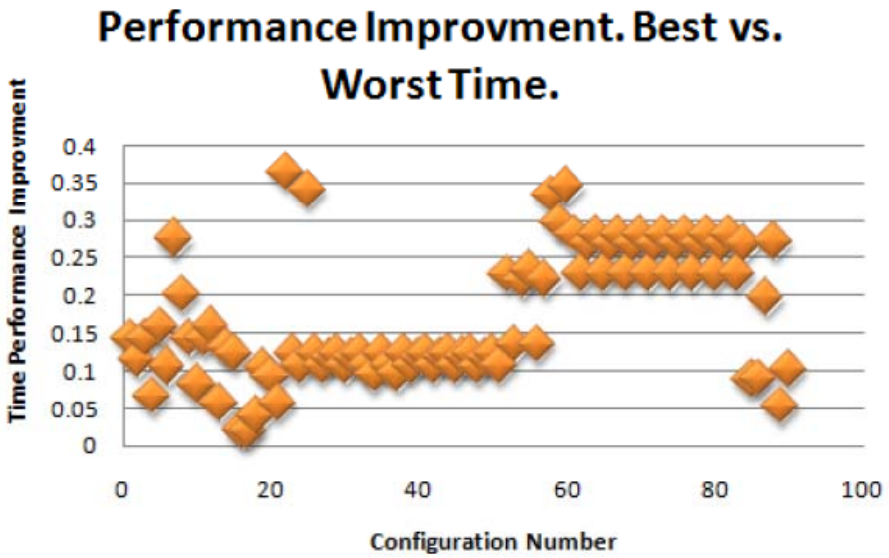
- **Simulation**
- We used
  - 30 configuration files
  - 12 input files
  - 5 scheduling algorithms

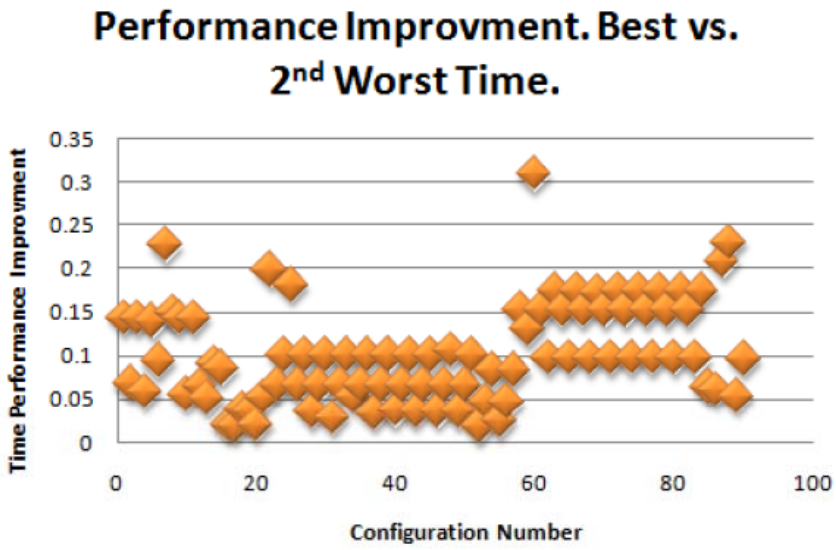
- => This means that we run 450 tests

Experimental Results

- The charts 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.





Discovered System Patterns

- **CCF algorithm** - CCR ≈ 1
- **CCF algorithm** - heterogeneous with the same CPU power, CCR < 1
- **HLFET algorithm** - CCR > 1, RCCR < 1, homogeneous or completely heterogeneous
- **Hybrid Remapper PS algorithm** - homogeneous, CCR < 1
- **ETF algorithm** - heterogeneous with the same memory, CCR < 1, RCCR ≈ 1
- s

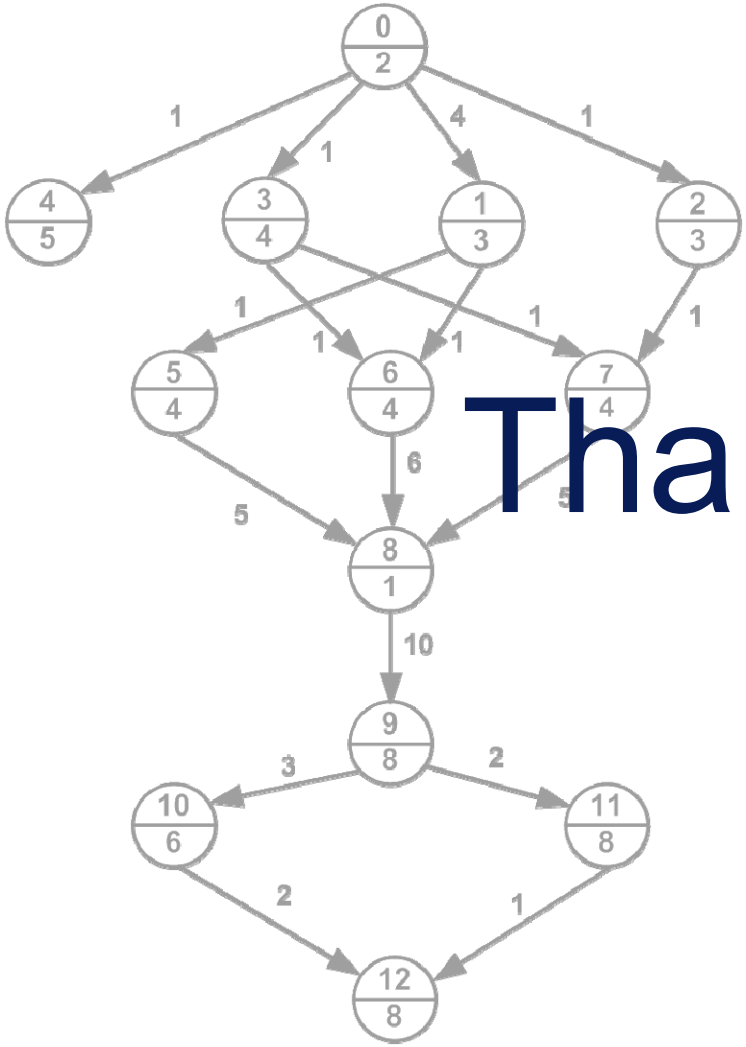
Conclusions

- We have compared the performance of several algorithms that represent alternative major approaches to scheduling a large set of different Grid environments and applications.
- The experiments 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.
- We demonstrated the efficiency of our proposed adaptive scheduling approach 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.

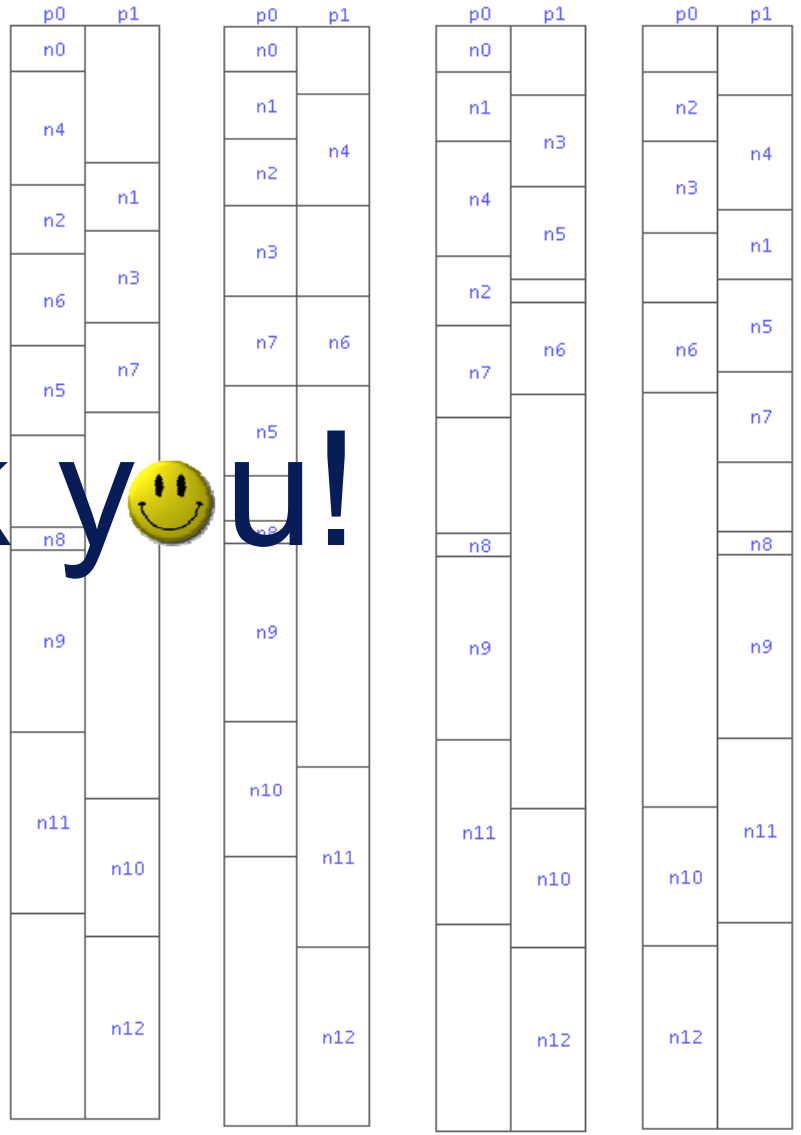
Future work

- Our future work will cover the following scenario:
  - The use of data mining heuristic, an supervised learning algorithm, to generate the class association rules, the algorithm needed for a set of system characteristics, that 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 if we talk about the case of error recovering or the case of performance improvement.

Q&A


0
2
4
5
3
4
1
3
2
3
5
4
6
4
7
4
8
1
9
8
10
6
11
8
12
8
1
4
1
1
1
1
1
1
1
6
5
5
10
3
2
2
1

Thank you!


p0
p1
n0
n4
n2
n6
n5
n8
n9
n11
n1
n3
n7
n10
n12