

Analysis of Tasks Reallocation in a Dedicated Grid Environment

Yves Caniou, Ghislain Charrier, Frédéric Desprez

LIP - INRIA - ÉNS Lyon

Cluster 2010

September 23, 2010

Presentation outline

- Context & problematic
 - Context
 - Problematic
- Experimental framework
- Simulations results
- Conclusions and perspectives

Context

- ▶ Computational Grid
- ▶ Heterogeneous clusters
- ▶ Local Resource Management Systems (LRMS)
 - ▶ Resource allocation
 - ▶ Local scheduling



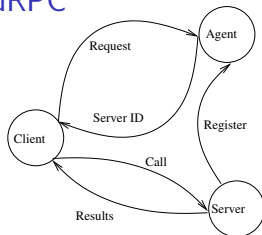
Context

- ▶ Computational Grid
- ▶ Heterogeneous clusters
- ▶ Local Resource Management Systems (LRMS)
 - ▶ Resource allocation
 - ▶ Local scheduling

- ▶ Middleware
 - ▶ Interconnects the clusters
 - ▶ Single entry point for users
 - ▶ GridRPC
 - ▶ Meta-scheduling
 - ▶ Servers submit jobs to LRMS



GridRPC



Local Resource Management Systems (LRMS)

Basics

- ▶ Jobs represented by a number of processors and a walltime
- ▶ Jobs killed once walltime is reached \Rightarrow Over evaluation of the walltime
- ▶ If jobs complete before completion: rescheduling

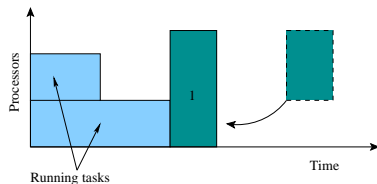
Local Resource Management Systems (LRMS)

Basics

- ▶ Jobs represented by a number of processors and a walltime
- ▶ Jobs killed once walltime is reached \Rightarrow Over evaluation of the walltime
- ▶ If jobs complete before completion: rescheduling

FCFS - First Come First Served

- ▶ Adds new jobs at the end of the queue



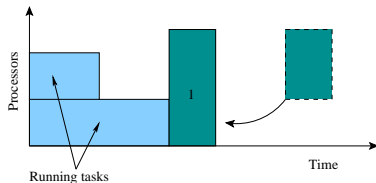
Local Resource Management Systems (LRMS)

Basics

- ▶ Jobs represented by a number of processors and a walltime
- ▶ Jobs killed once walltime is reached \Rightarrow Over evaluation of the walltime
- ▶ If jobs complete before completion: rescheduling

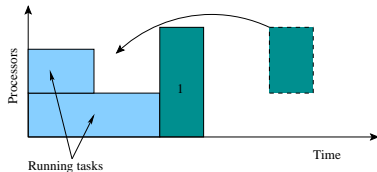
FCFS - First Come First Served

- ▶ Adds new jobs at the end of the queue



CBF - Conservative BackFilling

- ▶ Looks if there is space in the queue



Problematic

Problems

- ▶ Scheduling done at two levels: Meta-scheduler + LRMS
- ▶ Local scheduling errors due to walltime over-estimation impact the global scheduling

⇒ The middleware must take this into account

Problematic

Problems

- ▶ Scheduling done at two levels: Meta-scheduler + LRMS
- ▶ Local scheduling errors due to walltime over-estimation impact the global scheduling

⇒ The middleware must take this into account

Reallocation mechanism

- ▶ Automatically move *waiting* jobs between clusters

Problematic

Problems

- ▶ Scheduling done at two levels: Meta-scheduler + LRMS
- ▶ Local scheduling errors due to walltime over-estimation impact the global scheduling

⇒ The middleware must take this into account

Reallocation mechanism

- ▶ Automatically move *waiting* jobs between clusters

Objectives regarding reallocation

- ▶ Generic solution in a GridRPC middleware
- ▶ Easily deployable on existing architectures
- ▶ Propose different reallocation mechanisms and scheduling heuristics
- ▶ Compare the different algorithms on different metrics
- ▶ Quantify the benefits of reallocation
- ▶ Choose best heuristic to implement

Presentation outline

- Context & problematic
- **Experimental framework**
 - Simulator architecture
 - Algorithms
 - Simulated data
- Simulations results
- Conclusions and perspectives

Simulator architecture

- ▶ Developed with Simgrid and SimBatch (MSG API)
- ▶ 3 main components (cf GridRPC):

Server Runs on the cluster frontal.

- ▶ Provides middleware services
- ▶ Submits jobs to the LRMS
- ▶ Returns an estimated completion time of a job
- ▶ Cancels a waiting job in the system

Agent

- ▶ Meta-scheduler
- ▶ Manages reallocation

Client

- ▶ Asks the agent for a server for each request
- ▶ Sends requests to the servers

Reallocation algorithms

- ▶ Two reallocation algorithms
- ▶ Executed by the meta-scheduler
- ▶ Triggered periodically
- ▶ Jobs order chosen by a scheduling heuristic

Reallocation algorithms

- ▶ Two reallocation algorithms
- ▶ Executed by the meta-scheduler
- ▶ Triggered periodically
- ▶ Jobs order chosen by a scheduling heuristic

Regular algorithm

- 1: The meta-scheduler gets the list of all waiting jobs
- 2: For each job
- 3: Ask an estimation of completion time on all clusters
- 4: If $\text{newECT} + 60 < \text{oldECT}$
- 5: Move job

Reallocation algorithms

- ▶ Two reallocation algorithms
- ▶ Executed by the meta-scheduler
- ▶ Triggered periodically
- ▶ Jobs order chosen by a scheduling heuristic

Regular algorithm

- 1: The meta-scheduler gets the list of all waiting jobs
- 2: For each job
- 3: Ask an estimation of completion time on all clusters
- 4: If $\text{newECT} + 60 < \text{oldECT}$
- 5: Move job

All-cancellation algorithm

- 1: The meta-scheduler gets and cancels all waiting jobs
- 2: For each job
- 3: Ask an estimation of completion time on all clusters
- 4: Submit job to the cluster with minimum ECT

Scheduling heuristics

Minimum Completion Time (MCT)

- ▶ *Online* algorithm
- ▶ Chooses the server able to complete the job first

MinMin / MaxMin / MaxGain / MaxRelGain / Sufferage

- ▶ *Offline* algorithms
- ▶ Compute MCT for each job
- ▶ Chooses the job with the minMCT / maxMCT / maxGain / maxRelGain / maxSufferage
- ▶ Starts again on the remaining jobs

$gain = currentMCT - newMCT$

$relGain = gain / nbProcs$

$sufferageValue = secondBestMCT - bestMCT$

Simulated platforms

Platforms

- ▶ Grid'5000
 - ▶ Bordeaux (650 cores): slowest cluster
 - ▶ Lyon (170 cores): 20% faster than Bordeaux
 - ▶ Toulouse (434 cores): 40% faster than Bordeaux
- ▶ Parallel Workload Archive + Grid'5000
 - ▶ Bordeaux (650 cores): slowest cluster
 - ▶ CTC-SP2 (430 cores): 20% faster than Bordeaux
 - ▶ SDSC-SP2 (128 cores): 40% faster than Bordeaux

Jobs

- ▶ Traces of the simulated platforms

Simulated jobs

- ▶ Traces of real platforms
- ▶ No advanced reservations
- ▶ No data transfers
- ▶ 7 scenarios:

Month	Bordeaux	Lyon	Toulouse	Total
January	13084	583	488	14155
February	5822	2695	1123	9640
March	11673	8315	949	20937
April	33250	1330	1461	36041
May	6765	2179	1573	10517
June	4094	3540	1548	9182

Cluster	Bordeaux	CTC-SP2	SDSC-SP2	Total
nbJobs	74647	42873	15615	133135

Presentation outline

- Context & problematic
- Experimental framework
- **Simulations results**
 - Metrics
 - Results
 - Remarks on results
- Conclusions and perspectives

Things we Compare

Setups

- ▶ All submitted jobs pass through the meta-scheduler (MCT)
- ▶ Heterogeneous clusters
- ▶ All LRMS use the CBF scheduling policy

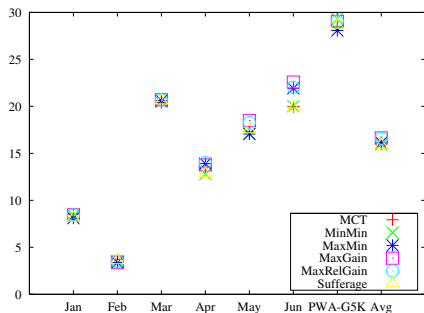
Metrics

Comparison with executions without reallocations

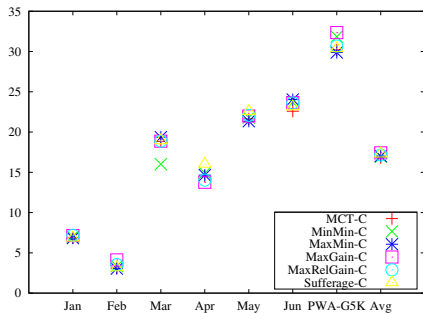
- ▶ System centered metric
 - ▶ Percentage of jobs impacted by reallocation
 - ▶ Number of reallocations relative to the total number of jobs
- ▶ User centered metrics
 - ▶ Percentage of jobs finishing earlier
 - ▶ Gain on average job response time

Percentage of jobs impacted

Regular



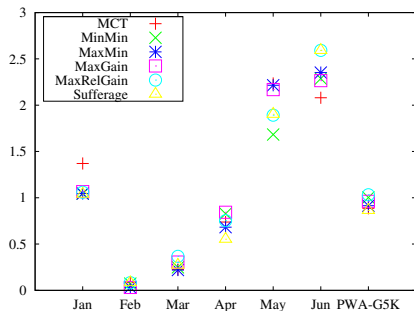
All-cancellation



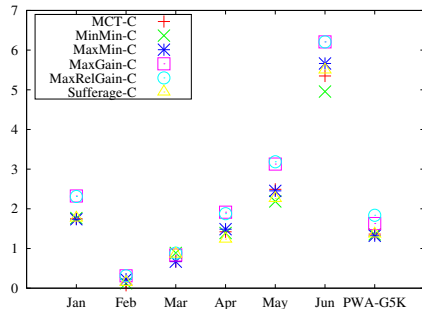
► Jobs impacted depend on trace, not on algorithm

Number of reallocations

Regular



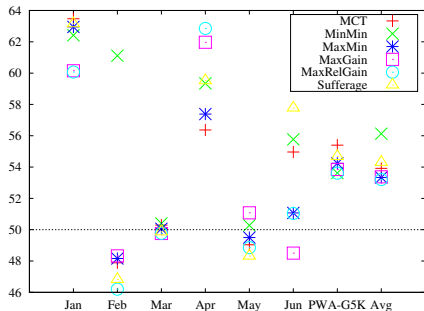
All-cancellation



- ▶ Few reallocations
- ▶ All-cancellation reallocates twice as much as regular
- ▶ Each reallocation may impact more than one job

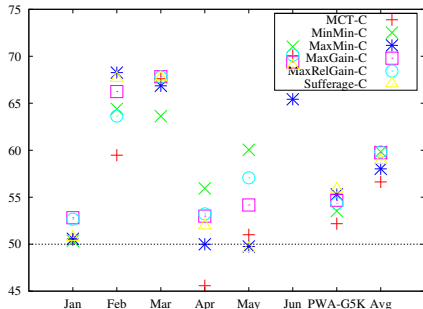
Percentage of jobs early

Regular



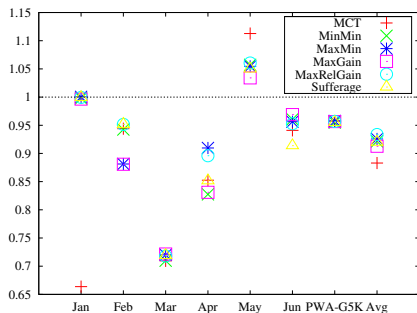
- ▶ More jobs early than late
- ▶ MinMin best on average

All-cancellation

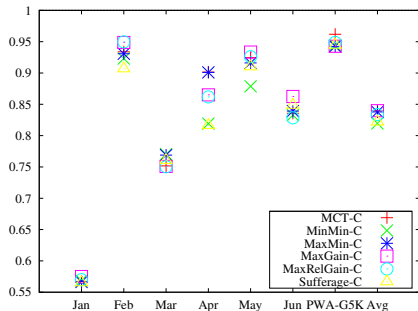


Relative average response time

Regular



All-cancellation



- ▶ On average: all scheduling heuristics give similar results
- ▶ Average gain of 10-15%
- ▶ Almost a factor 2 improvement in January for all-cancellation

Remarks on results

Algorithms

- ▶ All-cancellation better on users metrics
- ▶ All-cancellation reallocates more
- ▶ All-cancellation can cause starvation
- ▶ MCT is the choice for implementation: simple and efficient

Reallocation

- ▶ If perfect walltime, reallocation is useless
- ▶ Correct errors appearing after meta-scheduling, doesn't replace it
- ▶ Needs upper bound on the completion time of jobs
- ▶ Takes resources volatility into account

Other results

Using FCFS

- ▶ Queues are longer: less efficient algorithm than CBF
- ▶ More jobs impacted
- ▶ Twice more reallocations
- ▶ Better results on user metrics

On homogeneous platforms

- ▶ Waiting queues slower (choice of implementation)
- ▶ More jobs impacted / reallocations
- ▶ Better results on response time and jobs earlier
- ▶ Best result on average response time: 4 times faster

Presentation outline

- Context & problematic
- Experimental framework
- Simulations results
- Conclusions and perspectives

Conclusions and perspectives

Conclusions

- ▶ Reallocation mechanism in a GridRPC middleware
- ▶ Different reallocation mechanisms with heuristics
- ▶ All scheduling heuristics give close results
- ▶ Reallocation improves the users metrics
- ▶ Users can expect a 10-15% improvement on their average job response time

Perspectives

- ▶ Moldable jobs
- ▶ Implementation in DIET
- ▶ Jobs with deadlines
- ▶ Non dedicated platforms
- ▶ Jobs with no runtime prediction possible

Questions?

Thank you for your attention!