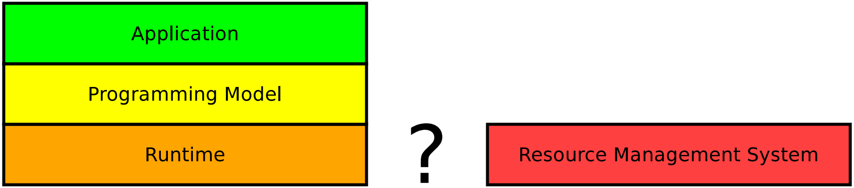
Cooperative Resource Management for Parallel and Distributed Systems

Cristian KLEIN, Christian PÉREZ

INRIA, Lyon, France

JLPC Workshop, November 19, 2012

High-Performance Computing



HPC Resource Management

Resource Management System (RMS)

- Multiplexes computing nodes among multiple users
- Aims at isolating them for security and **improved performance**

Current Practice

Dynamic allocations (à la Cloud)

[1] http://blog.cyclecomputing.com/2012/04/cyclecloud-50000-core-utility-supercomputing.html

[2] http://blog.cyclecomputing.com/2011/03/cyclecloud-4096-core-cluster.html

Current Practice

Dynamic allocations (à la Cloud)

- Clouds
  "The illusion of *infinite* computing resources available on demand"

[1] http://blog.cyclecomputing.com/2012/04/cyclecloud-50000-core-utility-supercomputing.html

[2] http://blog.cyclecomputing.com/2011/03/cyclecloud-4096-core-cluster.html

Current Practice

Dynamic allocations (à la Cloud)

- Clouds
  "The illusion of *infinite* computing resources available on demand"
  - Infinite? Actually up to 20 nodes

[1] http://blog.cyclecomputing.com/2012/04/cyclecloud-50000-core-utility-supercomputing.html

[2] http://blog.cyclecomputing.com/2011/03/cyclecloud-4096-core-cluster.html

Current Practice

Dynamic allocations (à la Cloud)

- Clouds
  "The illusion of *infinite* computing resources available on demand"
  - Infinite? Actually up to 20 nodes
  - HPC Supercomputer of 50,000 cores[1]
  - **Out of capacity** errors[2]

[1] http://blog.cyclecomputing.com/2012/04/cyclecloud-50000-core-utility-supercomputing.html

[2] http://blog.cyclecomputing.com/2011/03/cyclecloud-4096-core-cluster.html

Current Practice

Dynamic allocations (à la Cloud)

- Clouds
  "The illusion of *infinite* computing resources available on demand"
  - Infinite? Actually up to 20 nodes
  - HPC Supercomputer of 50,000 cores[1]
  - **Out of capacity** errors[2]

Static allocations (à la batch schedulers)

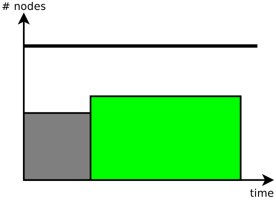
- a.k.a. rigid jobs (node-count times duration)
- **Misses** opportunities for improvement (next slide)

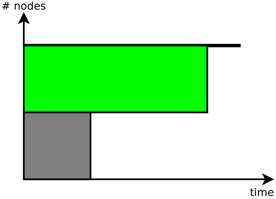
[1] http://blog.cyclecomputing.com/2012/04/cyclecloud-50000-core-utility-supercomputing.html

[2] http://blog.cyclecomputing.com/2011/03/cyclecloud-4096-core-cluster.html

Opportunities for Improvement

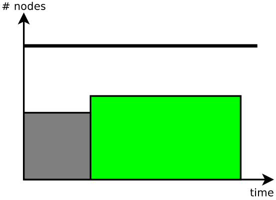
Moldability

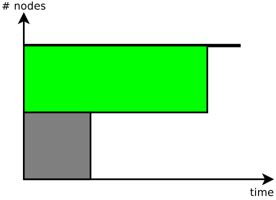




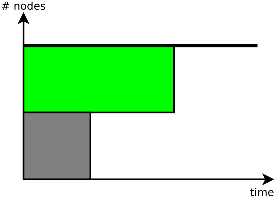
Opportunities for Improvement

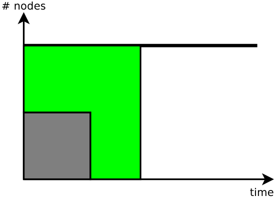
Moldability





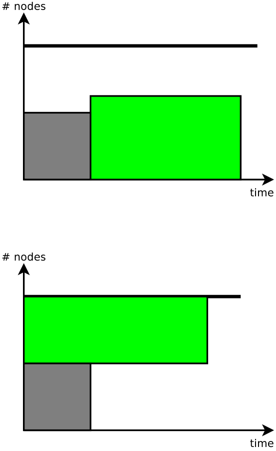
Malleability



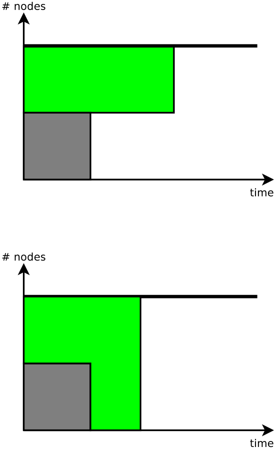


Opportunities for Improvement

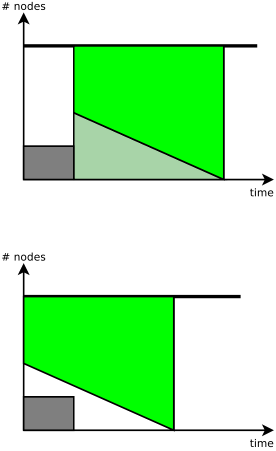
Moldability



Malleability

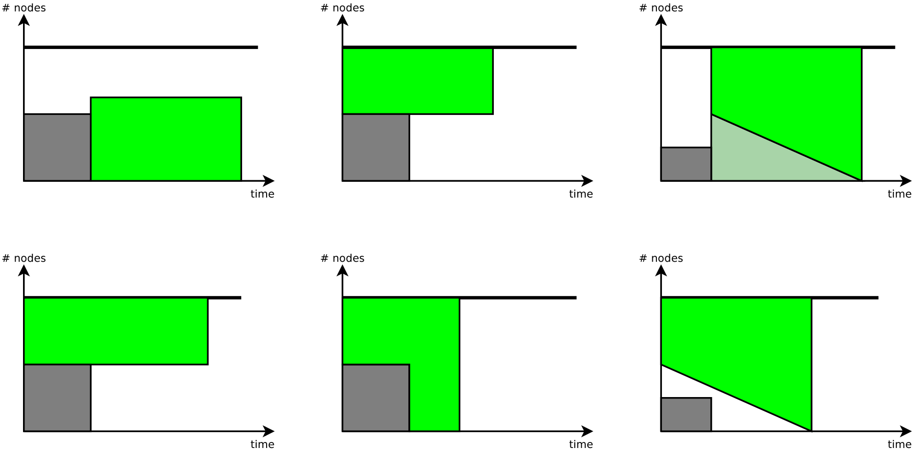


Evolution



Opportunities for Improvement

Moldability | Malleability | Evolution



Problem: Insufficiently supported in the state-of-the art.

CooRM: **Coo**perative **R**esource **M**anagement

Goal: Improve resource management

- Resource utilization
- Application completion time
- Energy consumption

How?

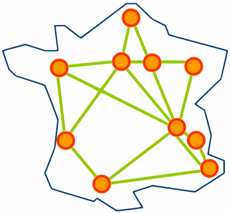
- Resource management architecture
- Support moldability, malleability, evolution **without workarounds**
- **Cooperates** with applications

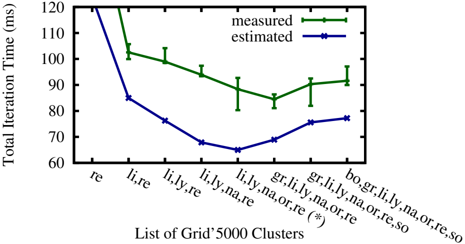
1. Introduction
2. CooRMv1: Moldability
   - Computational Electromagnetics Application
   - RMS Description
   - Evaluation
3. CooRMv2: Malleability, Evolution
   - Adaptive Mesh Refinement Application
   - RMS Description
   - Evaluation
4. Conclusions and Perspectives

Multi-cluster CEM Application

Computational ElectroMagnetics (CEM)

- Traditionally executed on a single cluster
- Huge mesh → launch on multiple clusters

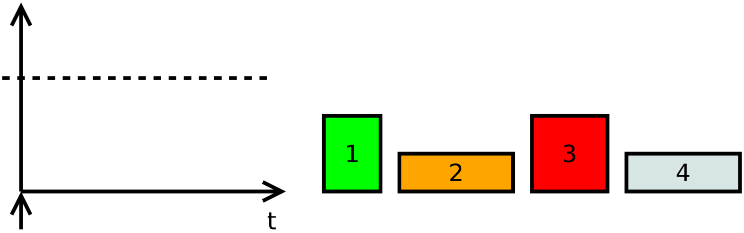




- Devised a **custom** resource selection **algorithm**

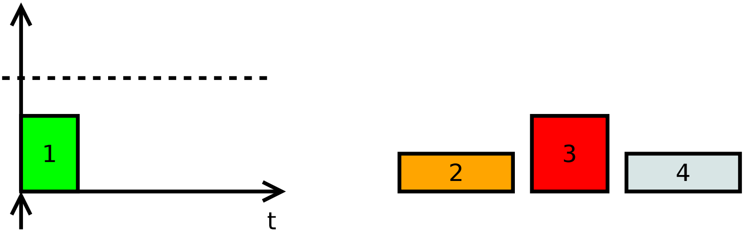
Towards Moldability

- No moldability (rigid jobs): fix node-count and duration



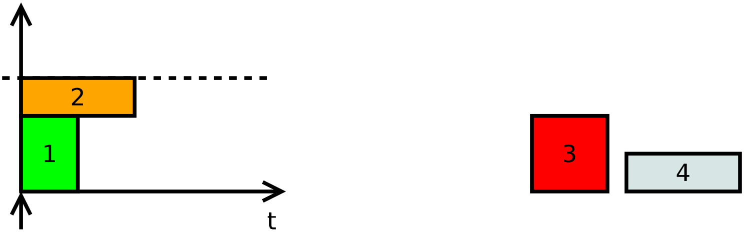
Towards Moldability

- No moldability (rigid jobs): fix node-count and duration



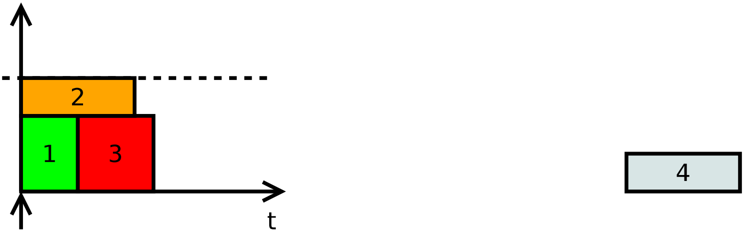
Towards Moldability

- No moldability (rigid jobs): fix node-count and duration



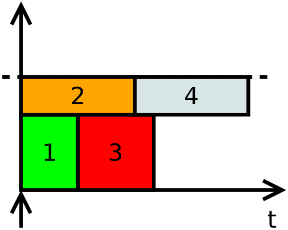
Towards Moldability

- No moldability (rigid jobs): fix node-count and duration



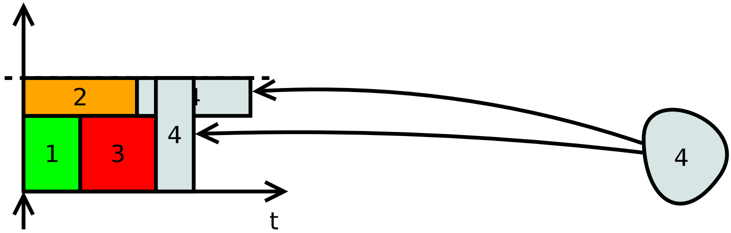
Towards Moldability

- No moldability (rigid jobs): fix node-count and duration



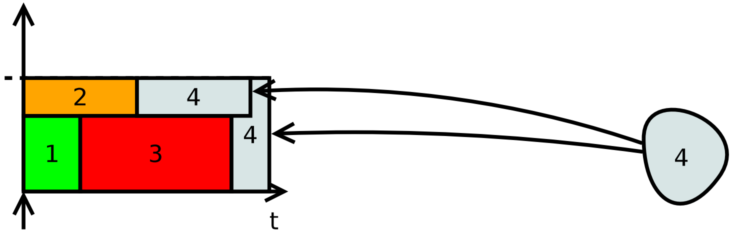
Towards Moldability

- No moldability (rigid jobs): fix node-count and duration



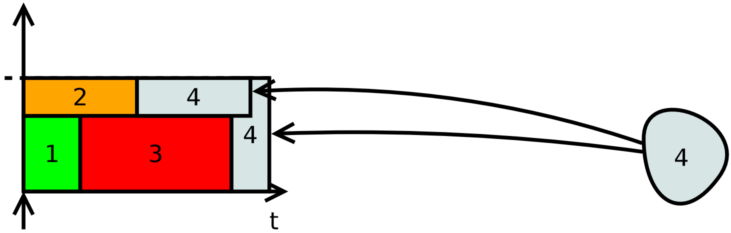
Towards Moldability

- No moldability (rigid jobs): fix node-count and duration



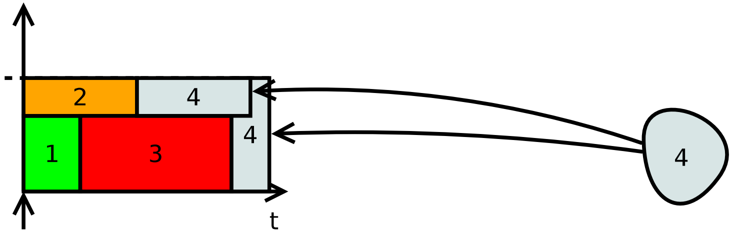
Towards Moldability

- No moldability (rigid jobs): fix node-count and duration
- **Limited** moldability: range of node-counts and a **single** duration
  - e.g., SLURM



Towards Moldability

- No moldability (rigid jobs): fix node-count and duration
- **Limited** moldability: range of node-counts and a **single** duration
  - e.g., SLURM
- Moldable configurations: list of node-count, durations
  - 8 nodes $\times$ 2 hours *OR* 16 nodes $\times$ 1 hour *OR* . . .
  - e.g., OAR, Moab
  - **Impractical**: large number of configurations (next slide)



Number of Configurations

For a multi-cluster system:

- e.g., number of nodes on each cluster
- # configurations is large (exponential)

# clusters:	C
# nodes per clusters:	N
# configurations:	$(N+1)^{\mathbf{C}} - 1$

For a supercomputer:

- number of CPU nodes
- number of CPU+GPU nodes
- network topology
- # configurations is large (potentially exponential)

Number of Configurations

For a multi-cluster system:

- e.g., number of nodes on each cluster
- # configurations is large (exponential)

# clusters:	C
# nodes per clusters:	N
# configurations:	$(N+1)^{\mathbf{C}} - 1$

For a supercomputer:

- number of CPU nodes
- number of CPU+GPU nodes
- network topology
- # configurations is large (potentially exponential)

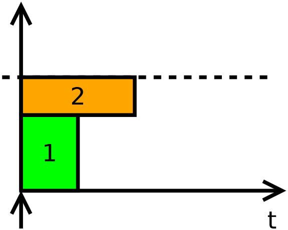
Problem

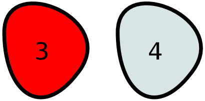
What interface should the RMS expose to allow moldable applications to effectively select resources?

Rationale

How CooRM Should Work

- Applications should take a more active role in the scheduling

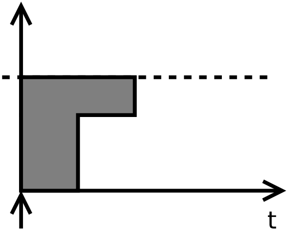


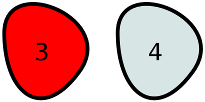


Rationale

How CooRM Should Work

- Applications should take a more active role in the scheduling
- RMS gives application the resource occupation (we call this a **view**)

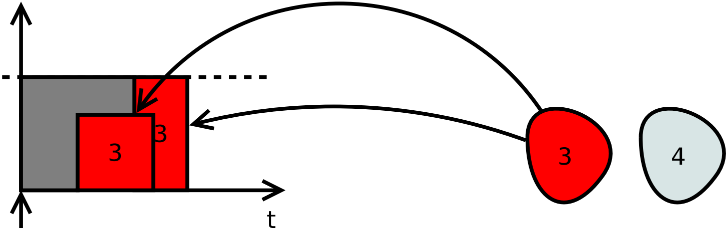




Rationale

How CooRM Should Work

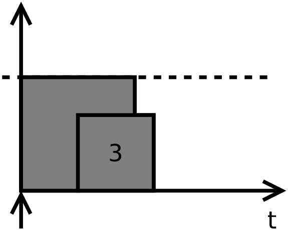
- Applications should take a more active role in the scheduling
- RMS gives application the resource occupation (we call this a **view**)
- Applications send a **resource requests**

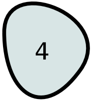


Rationale

How CooRM Should Work

- Applications should take a more active role in the scheduling
- RMS gives application the resource occupation (we call this a **view**)
- Applications send a **resource requests**

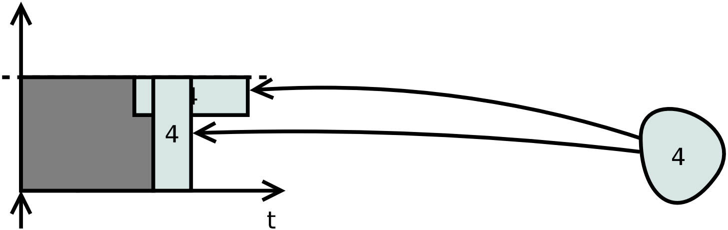




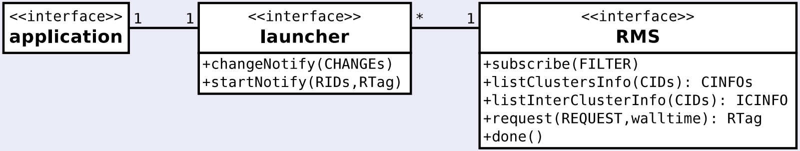
Rationale

How CooRM Should Work

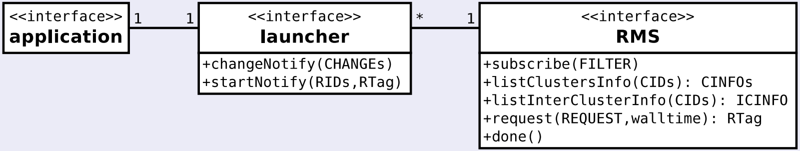
- Applications should take a more active role in the scheduling
- RMS gives application the resource occupation (we call this a **view**)
- Applications send a **resource requests**

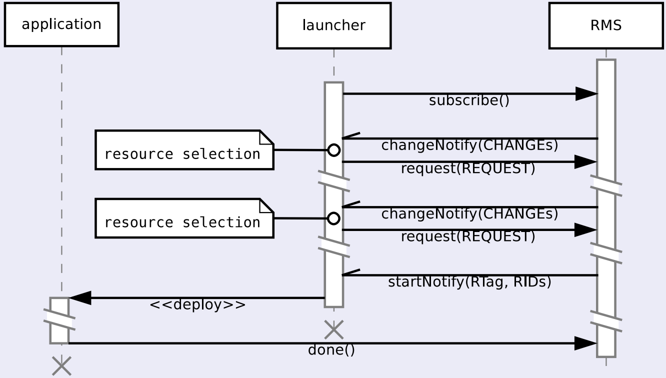


Architecture



Architecture





Evaluation

Setup

- Application model based on Parallel Workload Archive
- CooRM vs. listing configurations (à la OAR)

Results (scenarios previously practical)

☹ More network traffic (< 200KB per application)

☺ CPU usage on frontend **reduced** (≈ 3 times less)

☺ $\#$ configurations significantly **reduced** (≈ 10 times less)

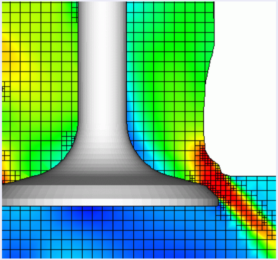
Results (scenarios previously impractical)

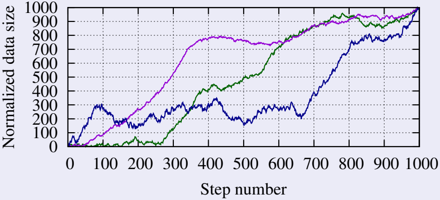
☺ Moldability **practical** in all cases
$\approx 10^3$ configurations vs. $\approx 10^{17}$

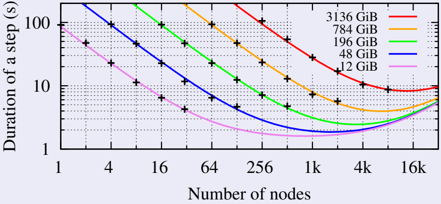
C. Klein, C. Pérez, *An RMS Architecture for Efficiently Supporting Complex-Moldable Application*, HPCC, 2011

Adaptive Mesh Refinement Applications (AMR)

- Mesh is dynamically refined / coarsened as required by numerical precision
  - Memory requirements increase / decrease
  - Amount of parallelism increases / decreases
- Generally **evolves non-predictably**







End-user's Goal: maintain a given target efficiency

Problem and Goal

Problem

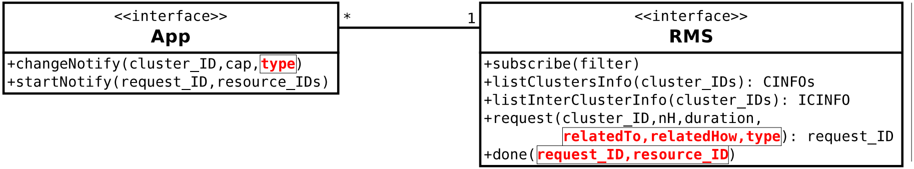
- Static allocations → **inefficient** resource utilisation
- Dynamic allocations → **out of capacity**

Goal

An RMS which allows non-predictably evolving applications

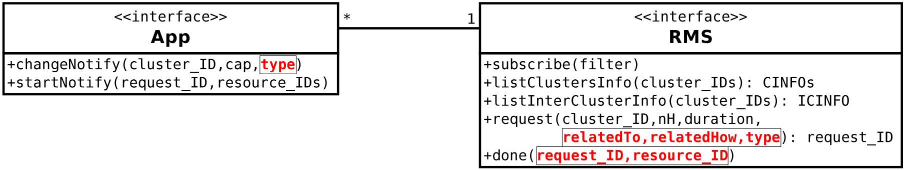
- To use resources **efficiently**
- **Guarantee** the availability of resources

CooRMv2: Resource Requests



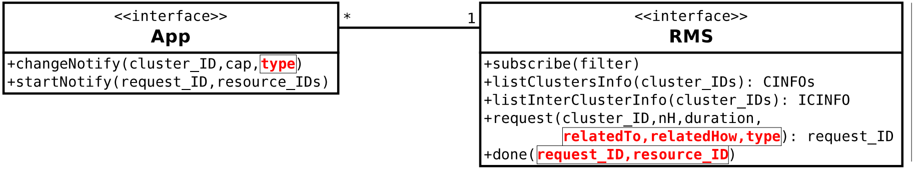
- number of nodes, duration
- RMS chooses start time $\rightarrow$ node IDs are allocated to the application

CooRMv2: Resource Requests



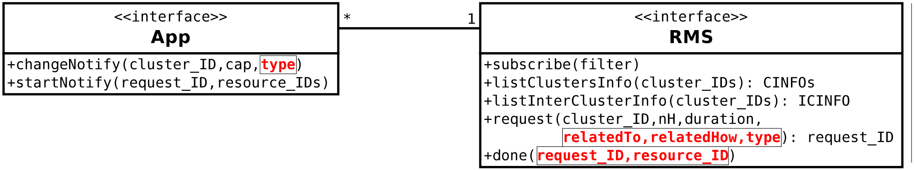
- number of nodes, duration
- RMS chooses start time → node IDs are allocated to the application
- Type
  - Non-preemptible (default in major RMSs, i.e., are not taken away)

CooRMv2: Resource Requests



- number of nodes, duration
- RMS chooses start time → node IDs are allocated to the application
- Type
  - Non-preemptible (default in major RMSs, i.e., are not taken away)
  - Preemptible (i.e., can be taken away at any time)

CooRMv2: Resource Requests



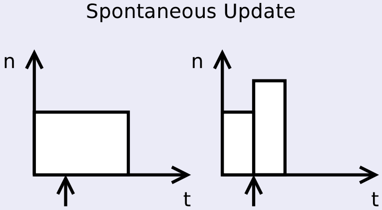
- number of nodes, duration
- RMS chooses start time → node IDs are allocated to the application
- Type
  - Non-preemptible (default in major RMSs, i.e., are not taken away)
  - Preemptible (i.e., can be taken away at any time)
  - Pre-allocation
    "I do not currently need these resources, but make sure I can get them immediately if I need them."

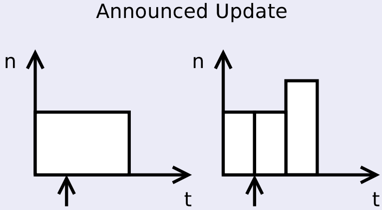
High-level Operations

Low-level Operations

- Add, update, delete requests

High-level Operations

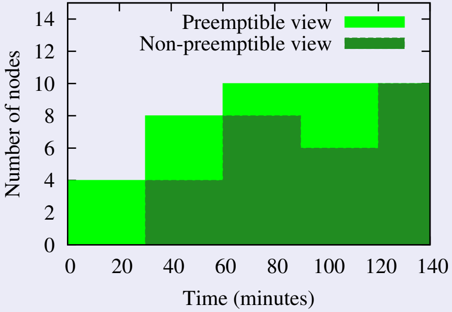




- An update is guaranteed to succeed only inside a pre-allocation

Views

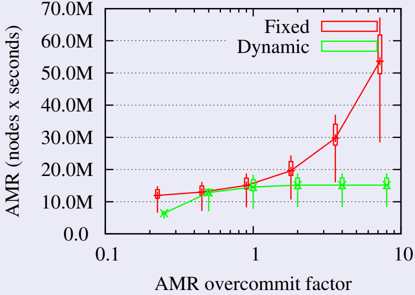
- Each app is presented with two views: **non-preemptible**, preemptible
- Preemptible view informs when resources need to be preempted

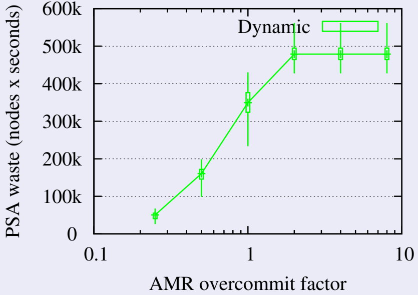


Scheduling with **Spontaneous** Updates

Experimental Setup

- Apps: 1xAMR (target eff. = 75%), 1xPSA (task duration = 600 s)
- Resources: number of nodes just enough to fit the AMR
- AMR uses fixed / dynamic allocations



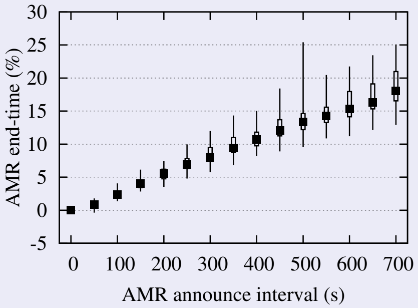


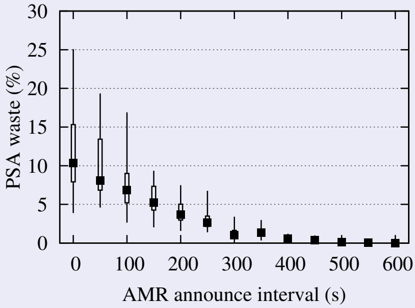
C. Klein, C. Pérez, *An RMS for Non-predictably Evolving Applications*, Cluster, 2011

Scheduling with **Announced** Updates

Experimental Setup

- Apps: 1xAMR (target eff. = 75%), 1xPSA (task duration = 600 s)
- Resources: number of nodes just enough to fit the AMR
- AMR uses announced updates (*announce interval*)





C. Klein, C. Pérez, *An RMS for Non-predictably Evolving Applications*, Cluster, 2011

Conclusions and Perspectives

Conclusions

- Improving the resource management on HPC resources
- **CooRM**: Cooperative resource management system
- Moldable: new cases become **practical** $(10^3$ vs. $10^{17})$
- Evolving: effective resource usage improved up to **7.2 times**
- Validated through prototype implementations

Conclusions and Perspectives

Conclusions

- Improving the resource management on HPC resources
- CooRM: Cooperative resource management system
- Moldable: new cases become **practical** (10^3 vs. 10^{17})
- Evolving: effective resource usage improved up to **7.2 times**
- Validated through prototype implementations

Perspectives

- **Topology** inside a supercomputer/cluster
  - Allow pre-launch topology optimization
- Interact with **runtime**: Charm++