# Resource Management Systems for Complex & Non-Predictably Evolving Applications

Cristian Klein & Christian Perez

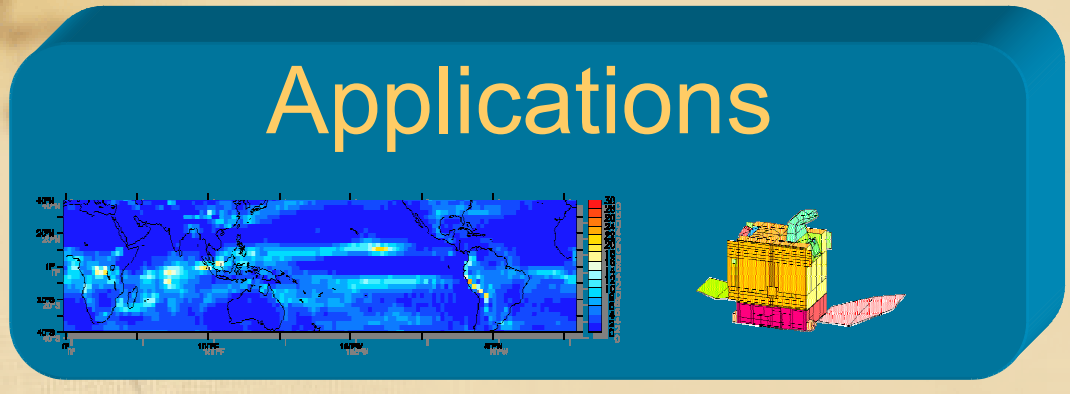
AVALON Team, INRIA/LIP, ENS Lyon, France

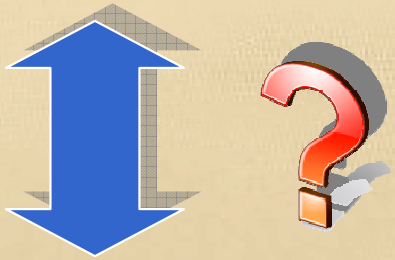
Intl. Research Workshop Advanced HPC Systems

Cetraro, Italy, June 28th, 2011

INRIA CNRS UNIVERSITÉ DE LYON UCBL LIP

# Avalon Research Topics

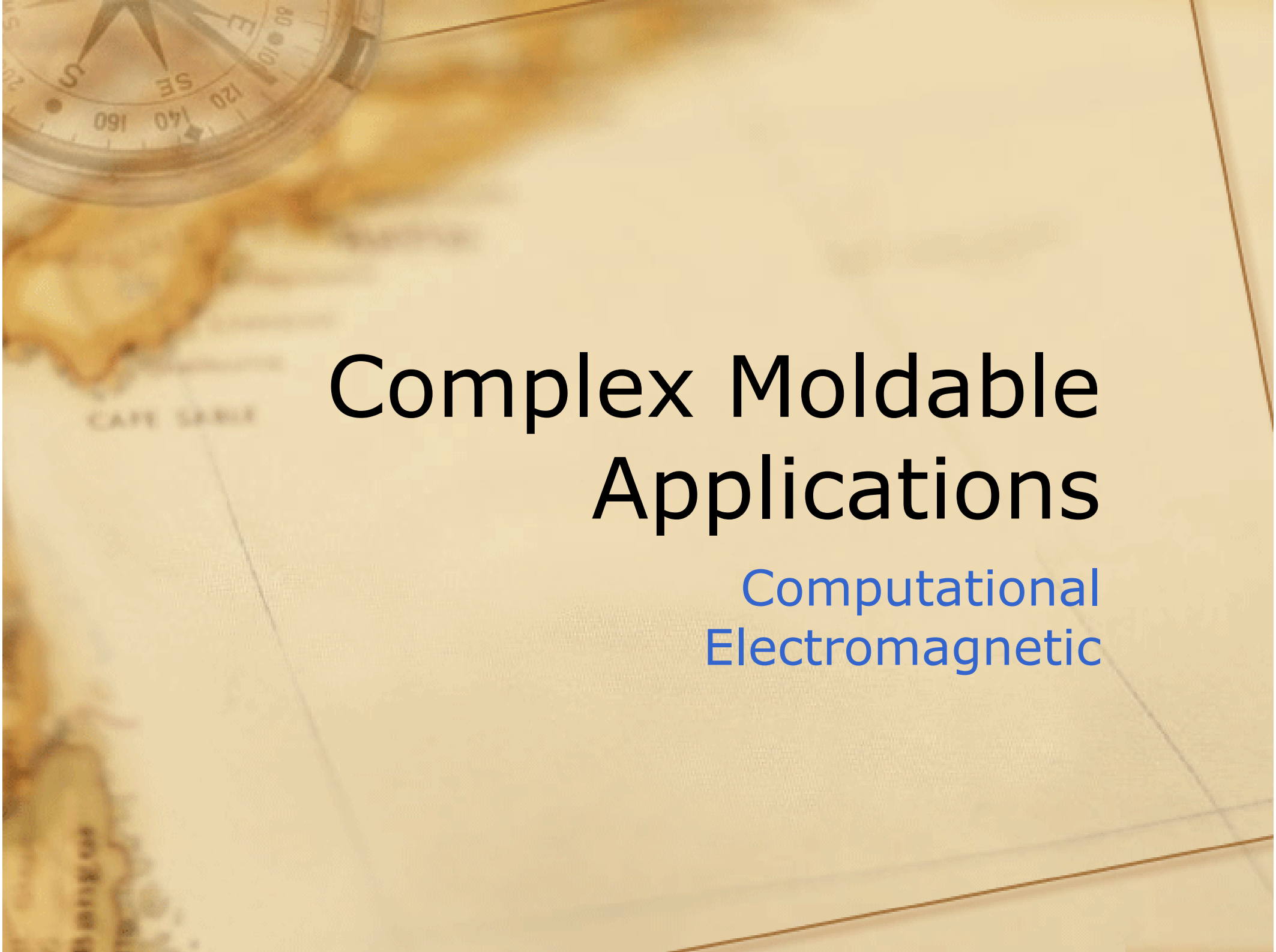



- Scientific applications
  - Complexity
    - Code coupling
  - Code and language heterogeneity
  - Performance needs
    - Computation
    - Data
- Objectives
  - Expressiveness simplicity
  - Application portability
  - Resource specific optimizations
  - Elastic resource management

# Content

- *Context*
- Complex moldable applications
  - Example (Computational Electromagnetic)
  - Limitations of existing RMS
  - CooRMv1 – Model & Experiments
- Non predictably evolving applications
  - Example (AMR)
  - Limitations of existing RMS
  - CooRMv2 – Model & Experiments
- Conclusion and future works

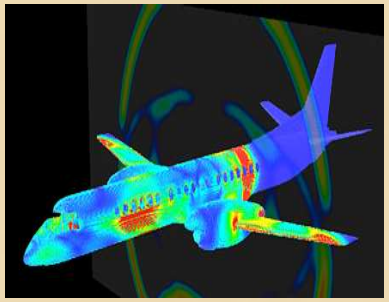


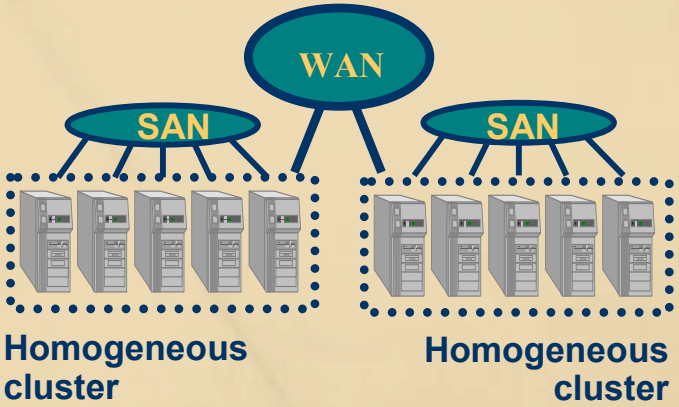
# Complex Moldable Applications

## Computational Electromagnetic

# Computational Electromagnetic

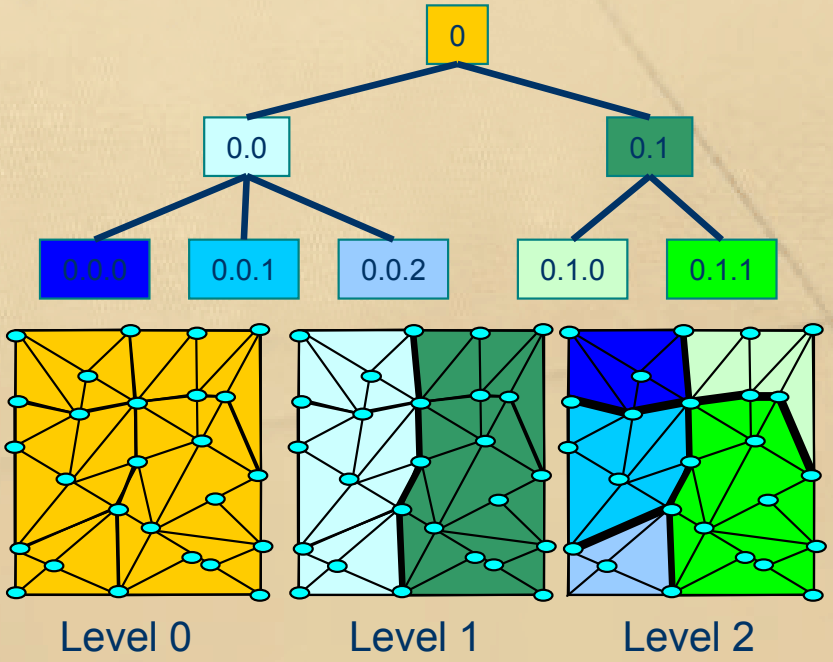
- From ANR DISCOGRID
  - ANR-05-CIGC-005






- Resource model
  - Hierarchical machines
    - Federation of clusters

- Application model
  - Set of MPI-based codes
    - How many groups?
    - Size of each group?



# Application Performance Model

- Inputs
  - Number of tetrahedra: $n_t$
  - Clusters: $n_H^i$ , $B_H^i$ , $B_C^i$ , $\alpha_j^i$
  - Maximum inter-cluster (WAN) latency: $l_{WAN}$
- One iteration

$$n_{tl}^{(i)} = n_t \cdot \frac{\sum_j 1/\alpha_j^{(i)}}{\sum_{k \in C} \left( n_H^{(k)} \cdot \sum_j 1/\alpha_j^{(k)} \right)}$$

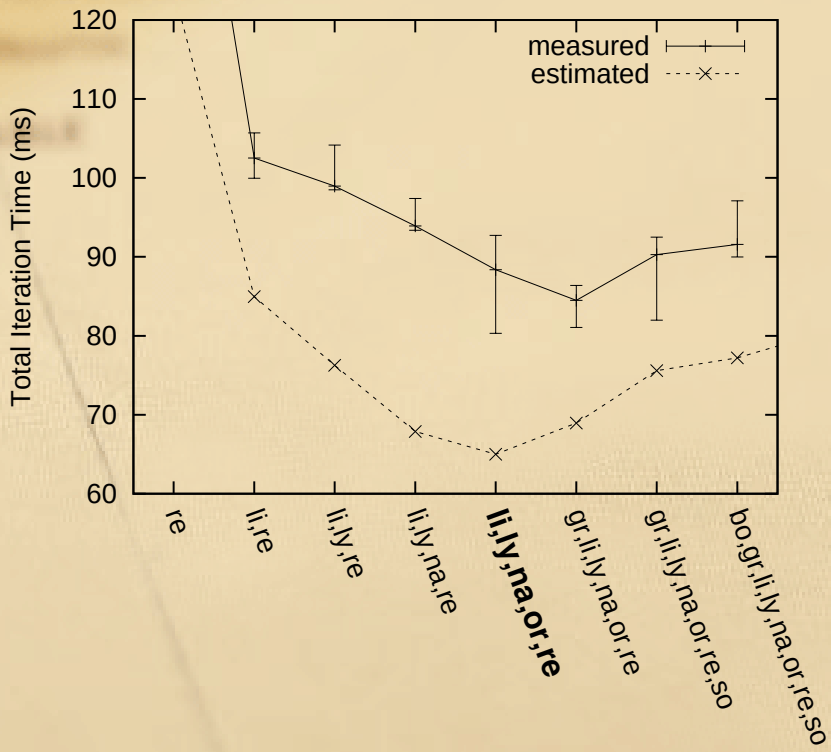
$$t = \sum_{j \in \text{non-overlap}} \max_{i \in C} \left( \alpha_j^{(i)} \cdot n_{tl}^{(i)} \right)$$

$$+ \sum_{j \in \text{overlap}} \max_{i \in C} \left( \alpha_j^{(i)} \cdot n_{tl}^{(i)}, l_{WAN} + \frac{s_U}{min \left( B_C^{(j)}, n_H^{(j)} \cdot B_H^{(j)} \right)} \cdot \beta_C \cdot \left( \frac{n_t}{n_C} \right)^{2/3} + \frac{s_U}{B_H^{(i)}} \cdot \beta_H \cdot \left( \frac{n_t}{\sum_{i \in C} n_H^{(i)}} \right)^{2/3} \right)$$

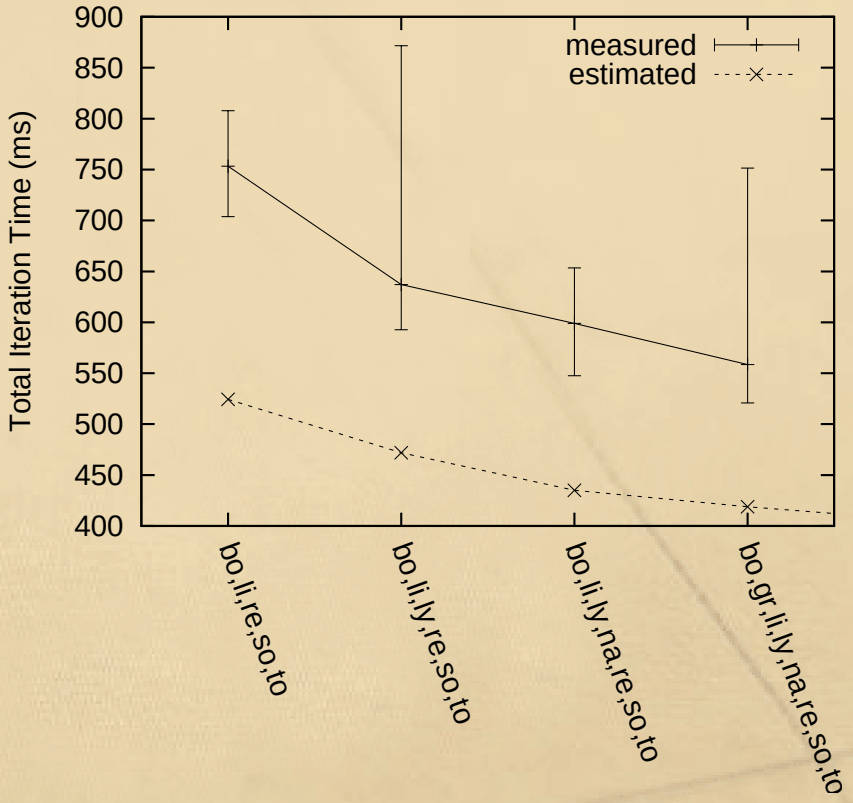
$$+ n_{AR} \cdot l_{WAN}$$

# Experiments on Grid'5000

**588, 245 tetrahedra**

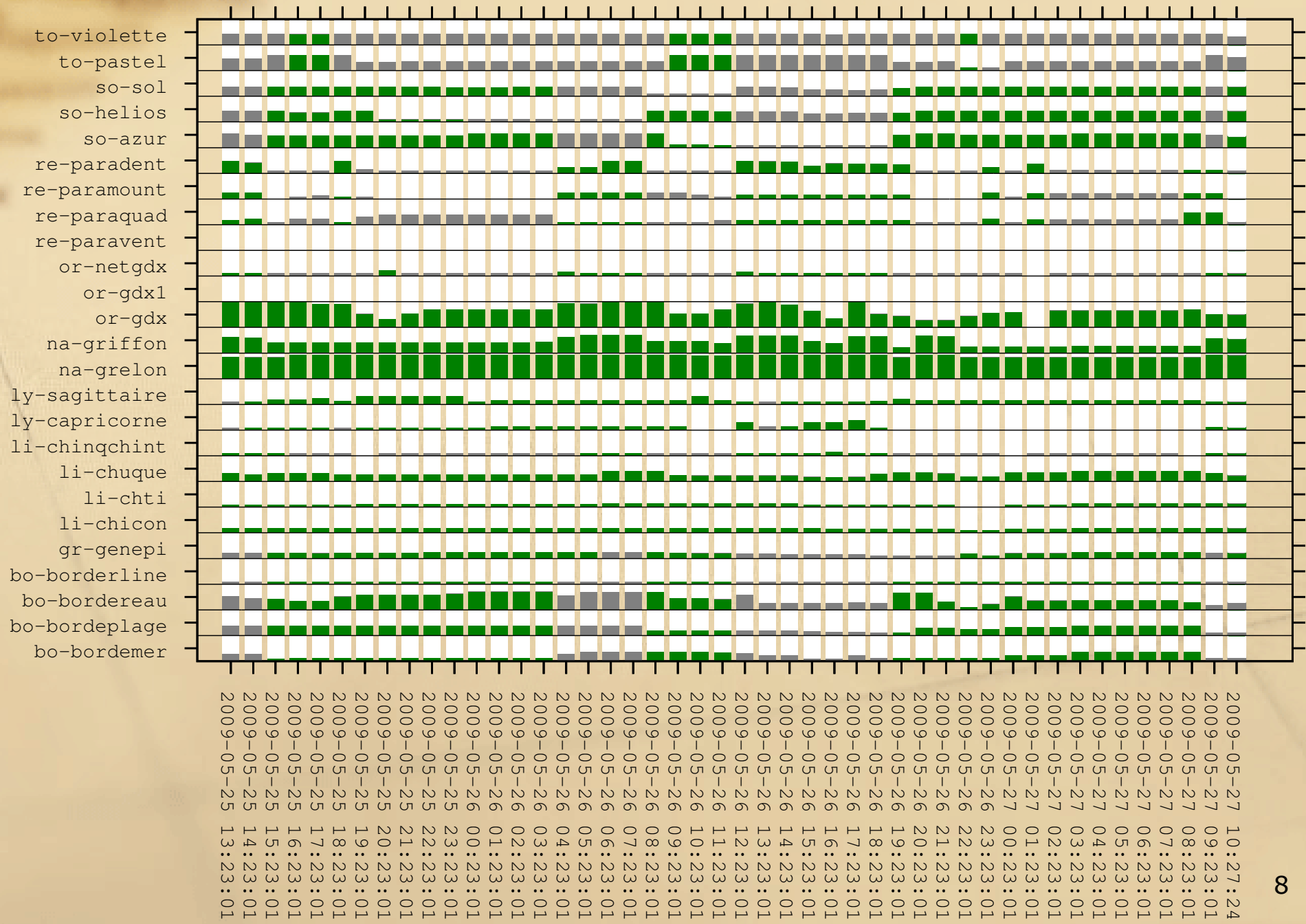



**9, 659, 894 tetrahedra**




- Expected optimal solution (all algorithms returned the same)
  - Added fast clusters
  - Removed slow clusters
- Actual optimal deployment slightly different

# CEM Moldability & G5K Dynamicity



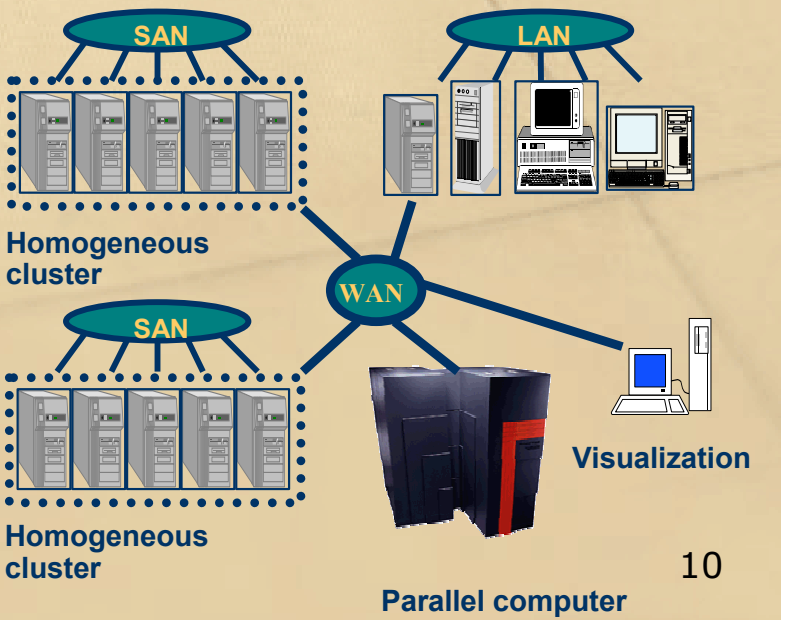
# Problem Statement

- How to automatically select the best resources for executing a moldable application given that the objective function to minimize if run-specific
  - Minimize makespan
  - Minimize cost
  - Minimize energy
  - Finish before a given deadline
  - Etc..
- And that
  - resources are efficiently used and
  - the availability of resources is guaranteed

# Limitations of Existing RMS

- Traditional RMS (Cluster, Supercomputers, …)
  - 1 queue ⇔ 1 known set of machines
    - **Fixed sized jobs**
      - Eg. 1024 nodes of quad-core processors
    - **Rigid jobs (JSDL)**
      - User selected number of processor
    - **Moldable jobs (SLURM, OAR, …)**
      - User selected range of number of processor
        - 1 global walltime (e.g., SLURM)
        - A set of rigid jobs (e.g., OAR)

- Federation of clusters / Grids
  - Finding adequate resource
    - Constraint or match-making language
- Clouds
  - On demand VMs
    - No guarantee on the availability
- Externally building a model of RMS
- Multiple submissions

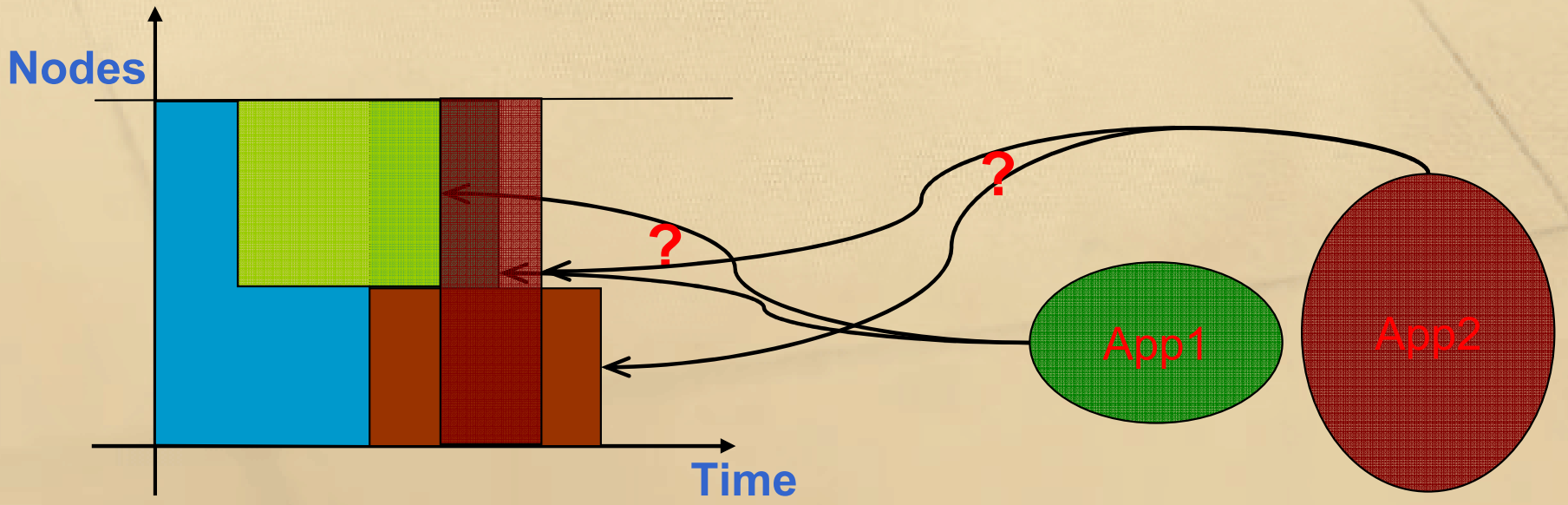


# CooRMv1

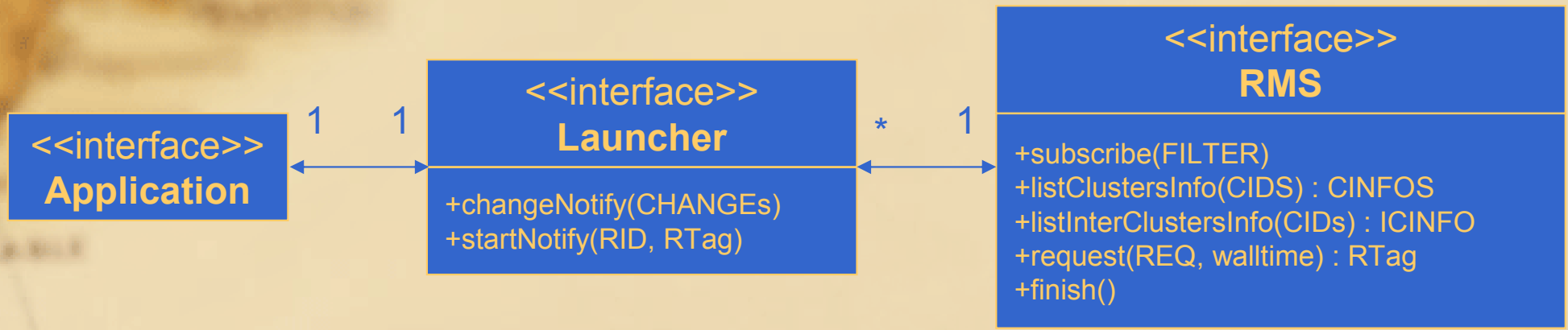
## Supporting Complex Moldable Applications

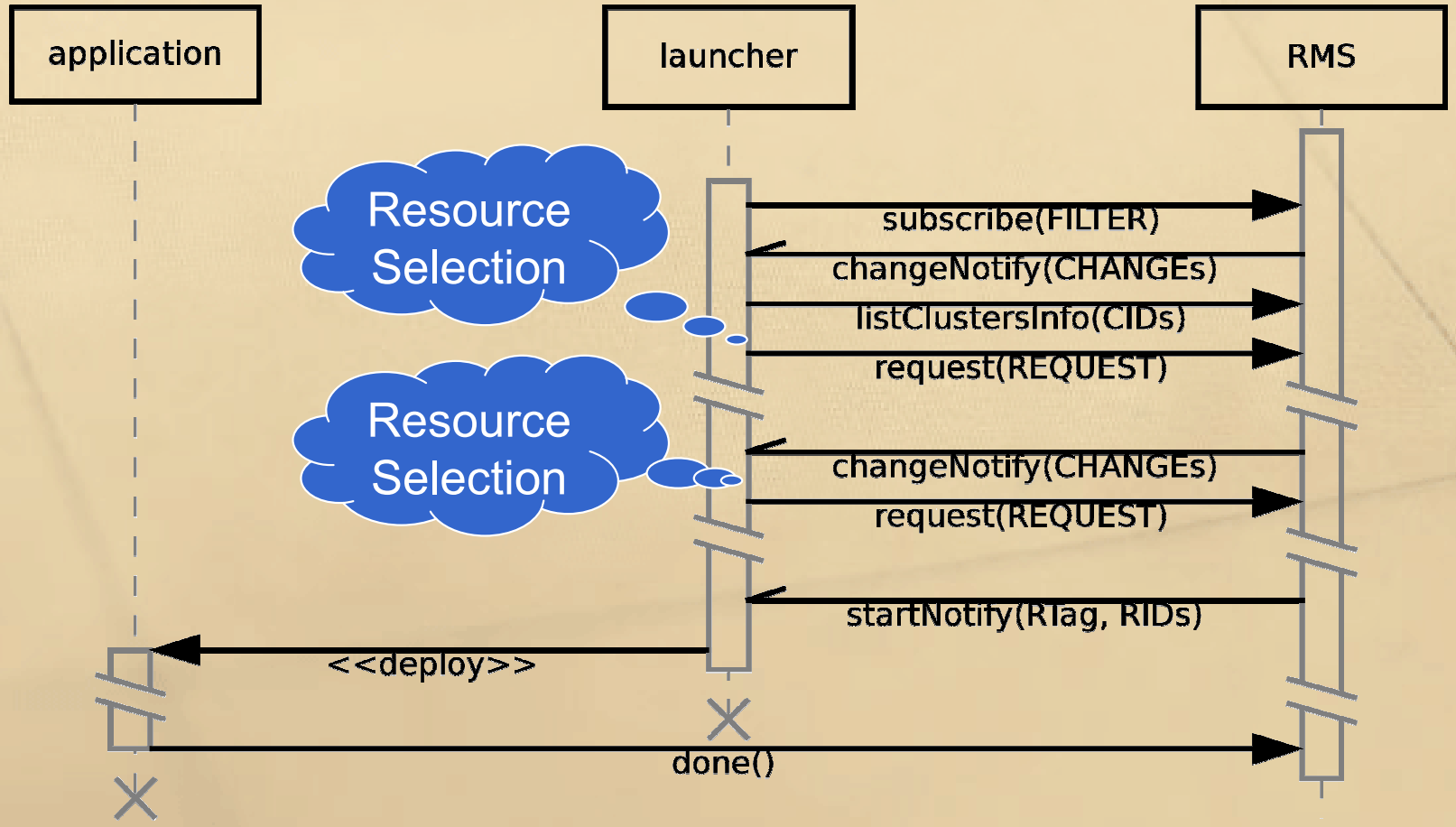
# Untying RMS from Moldable Application Scheduling

- Applications should take a more active role in the scheduling
- RMS gives application the resource occupation (a view)
- Application decides what resources to request



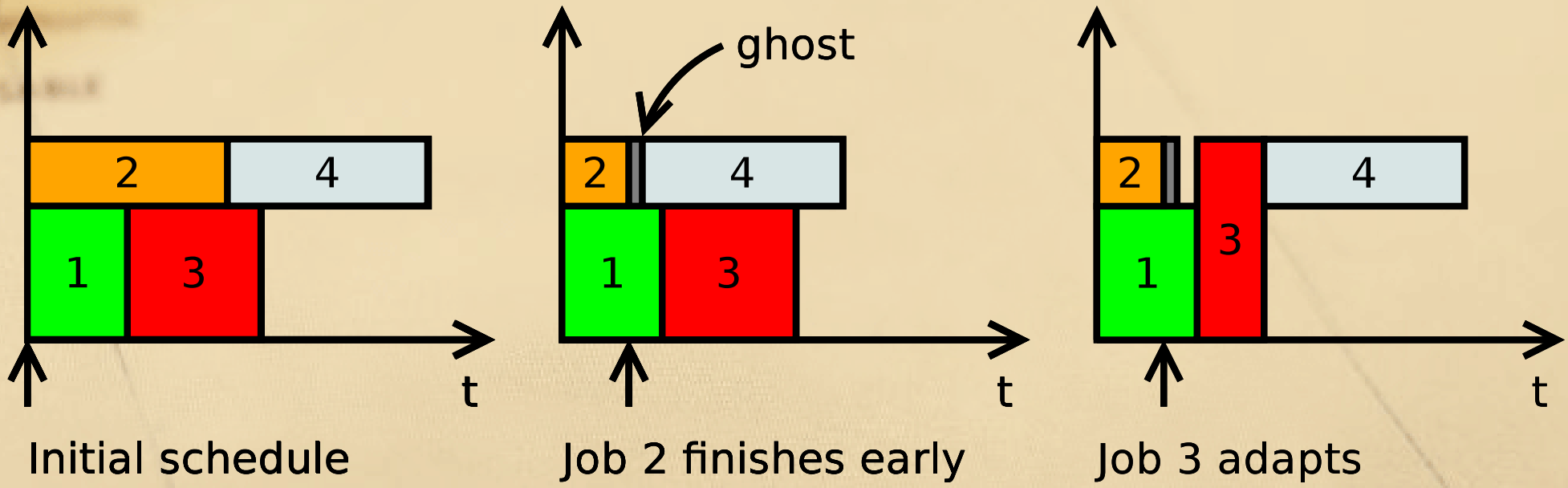
# CooRMv1: Overview

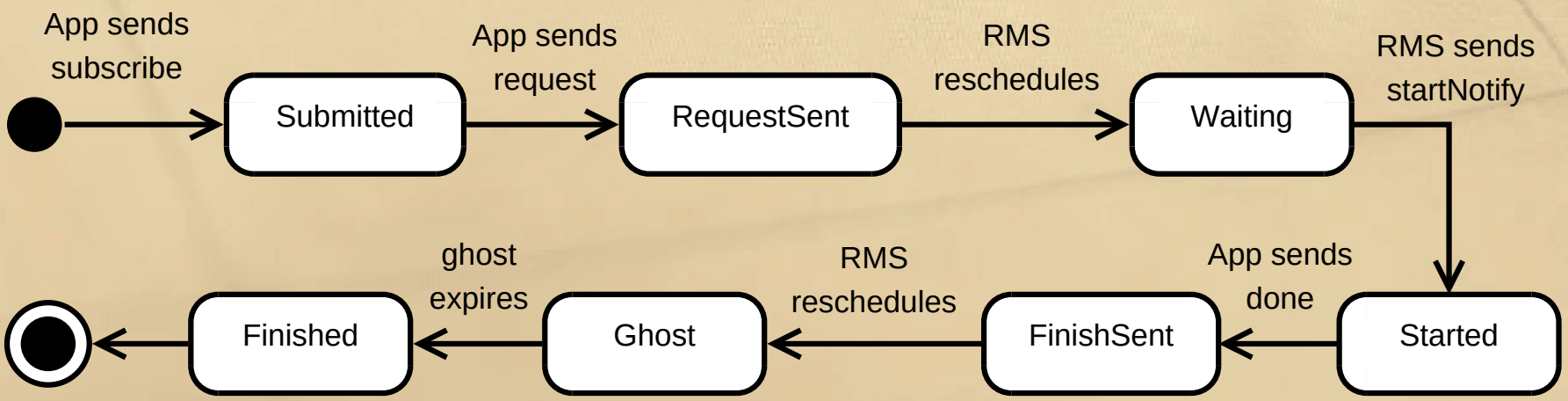


# RMS-side Scheduling

- Fair-start Delay and Ghost Zones

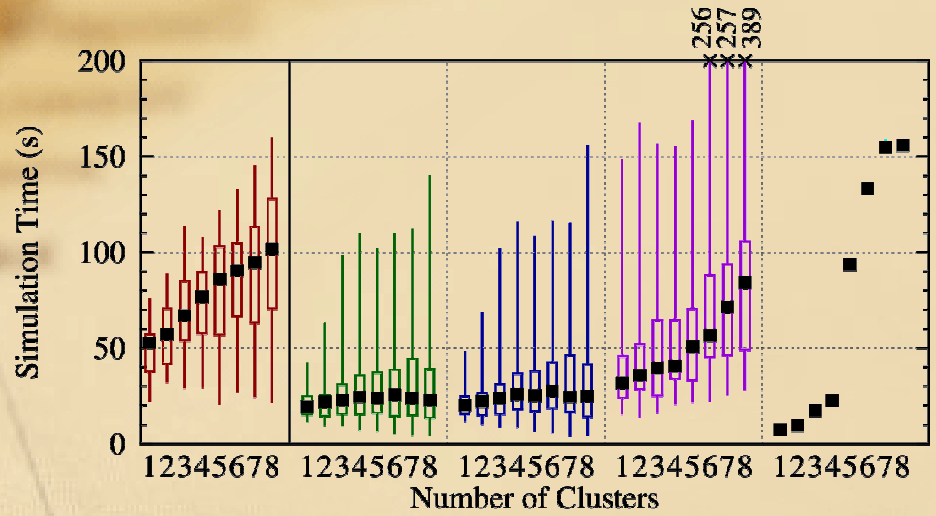


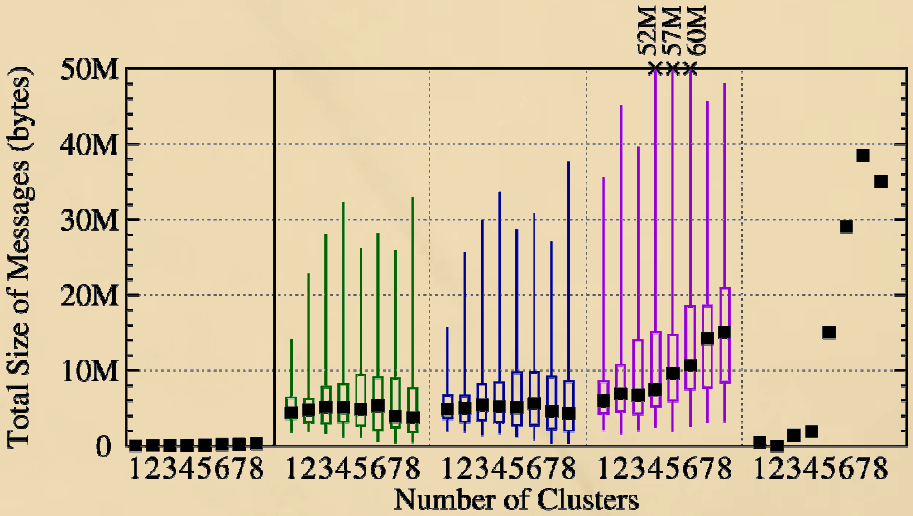



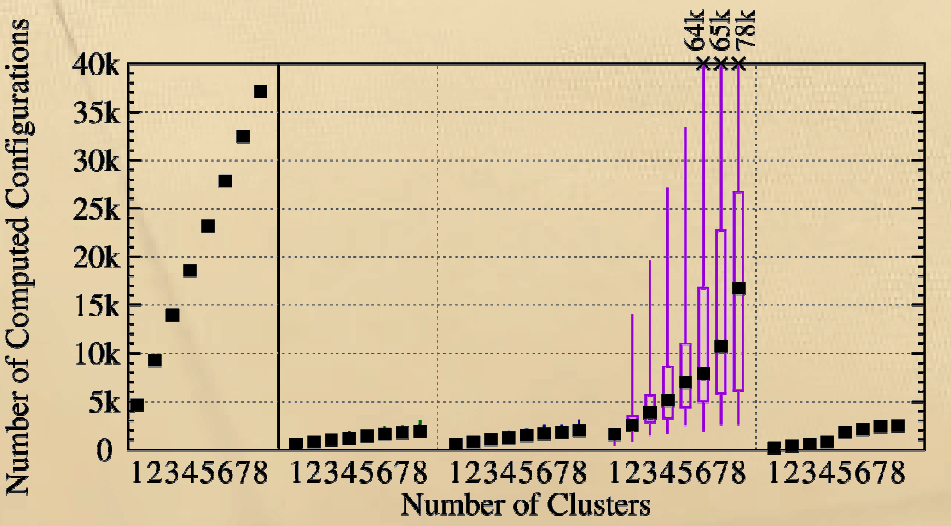
# RMS Scheduling Algorithm

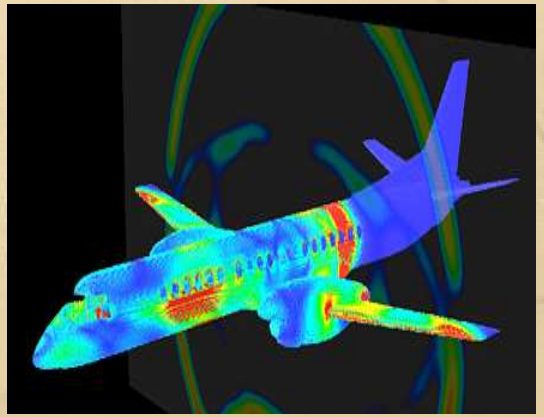
1. Mark as **Ghost** applications which sent **Done**
2. Mark as **Finished** ghosts which have expired
3. Generate the initial last view by adding the **Started** applications and **Ghosts** to it.
4. For all Request Sent and Waiting applications, taken in the order of their submission time:
    1. Set the application's view to the last view;
    2. Compute the application's estimated start time;
    3. Add its request to the last view;
    4. Mark the application as Waiting.
5. Send **changeNotify** messages to applications whose view has changed because of the previous steps.

# CooRM – Scalability



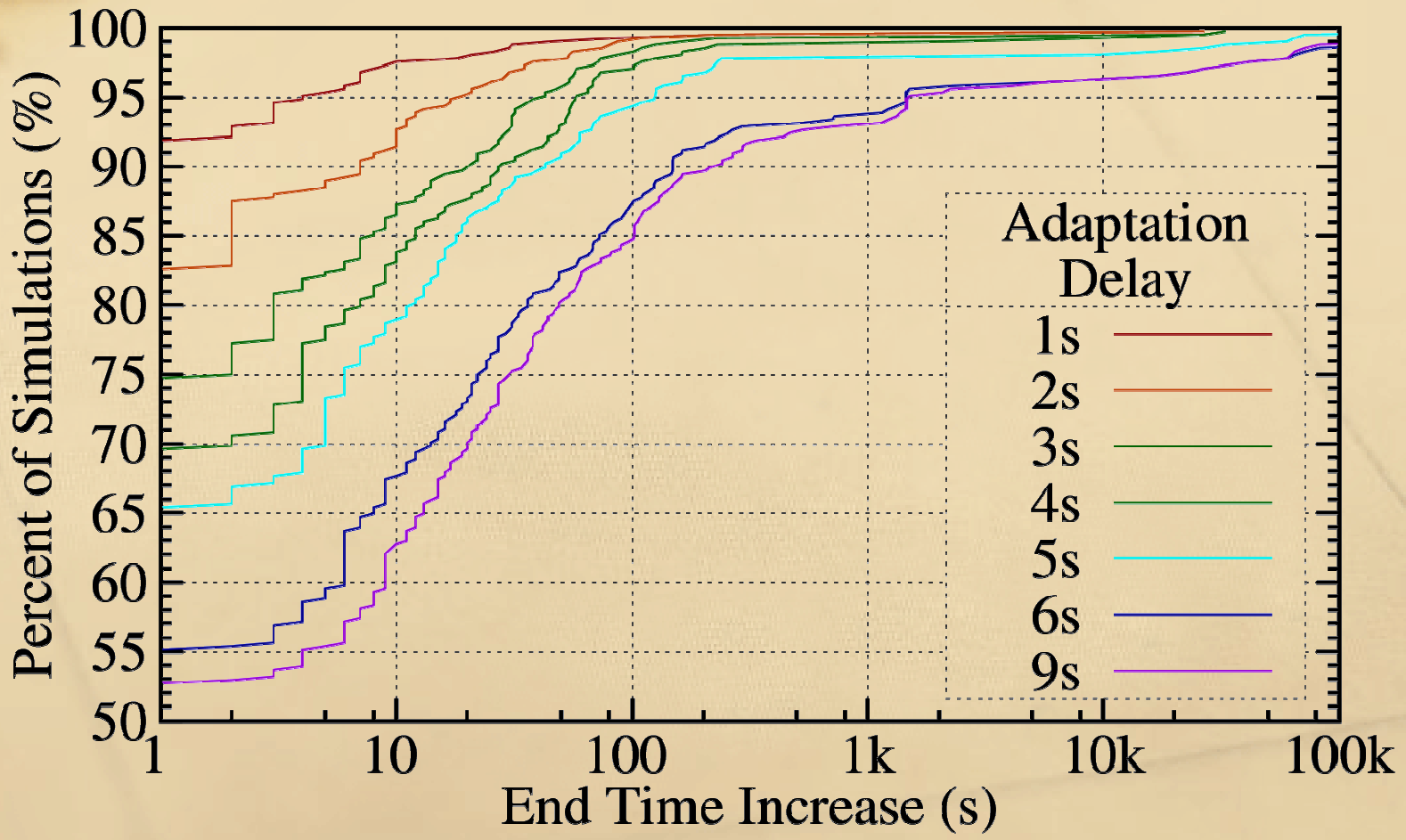



**Legend**

OAR : 0% CEM

CooRM: 0% CEM, 0.5% CEM, 50% CEM, 100% CEM

# CooRM – Fairness



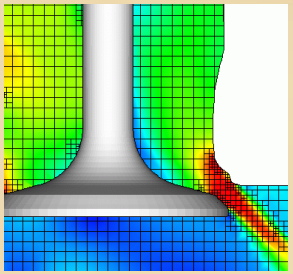
# Content

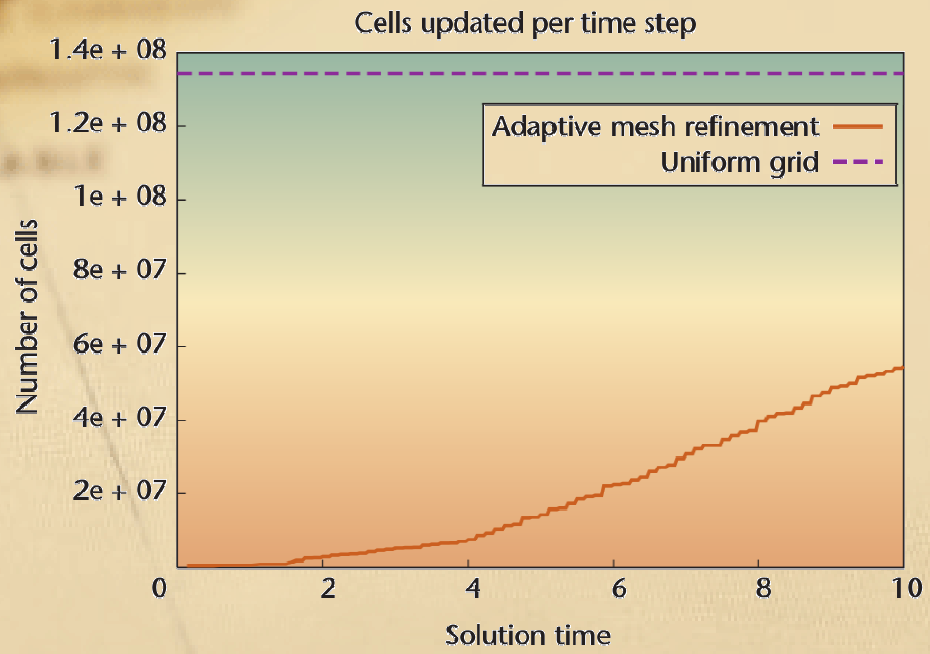
- *Context*
- *Complex moldable applications*
  - *Example (Computational Electromagnetic)*
  - *Limitations of existing RMS*
  - *CooRMv1 – Model & Experiments*
- Non predictably evolving applications
  - Example (AMR)
  - Limitations of existing RMS
  - CooRMv2 – Model & Experiments
- Conclusion and future works

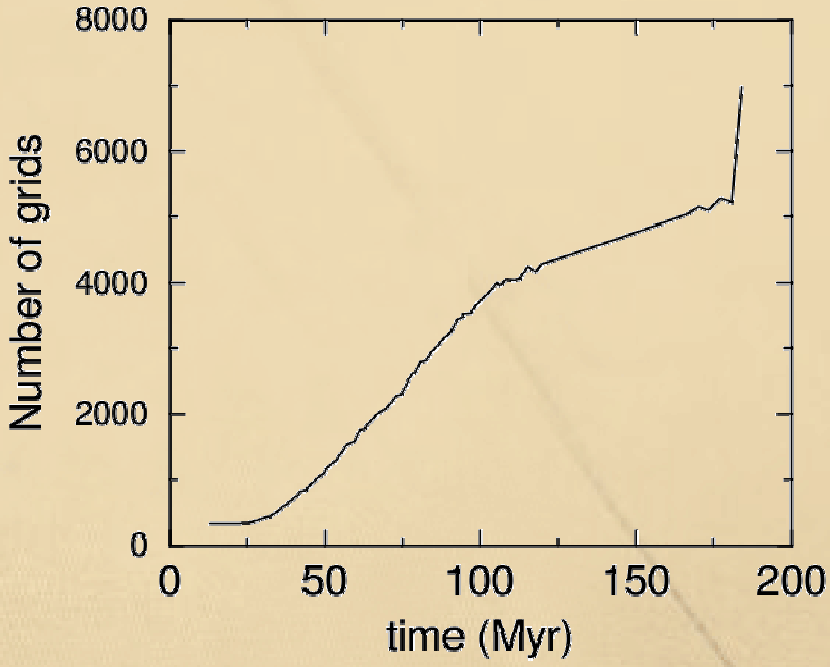
# Non-Predictably Evolving Applications

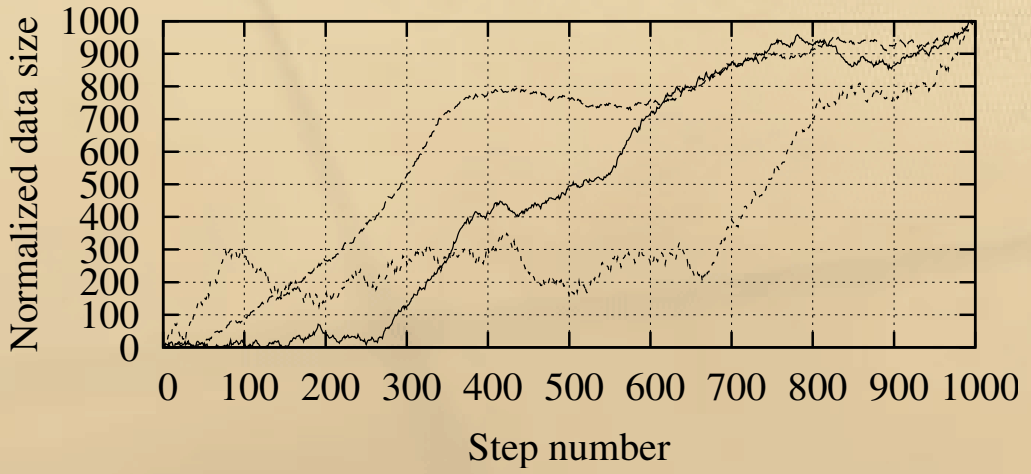
## Adaptive Mesh Refinement

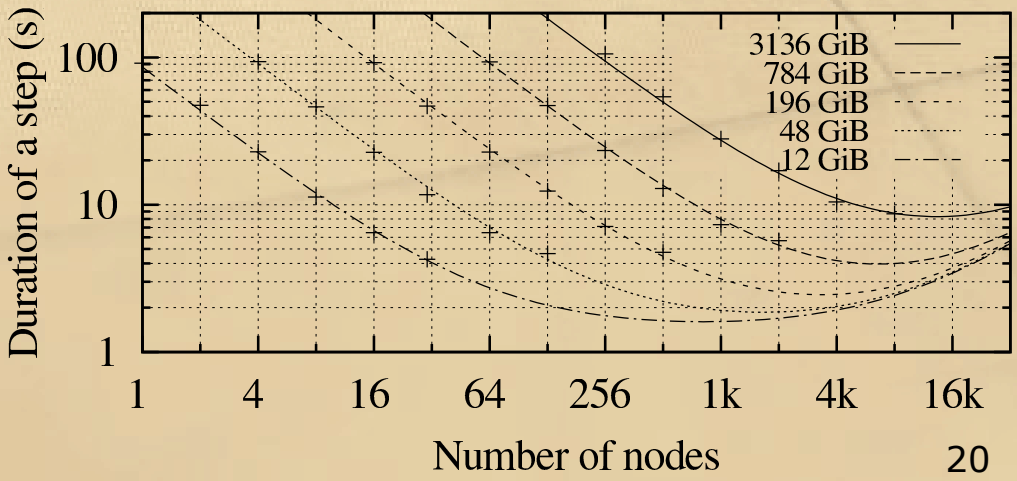
# AMR Applications





# Problem Statement Revisited

- How to automatically select the best resources for executing a moldable **and evolving** application given that the objective function to minimize if run-specific
  - Minimize makespan
  - Minimize cost
  - Minimize energy
  - Finish before a given deadline
  - Etc..
- And that
  - resources are efficiently used and
  - the availability of resources is guaranteed

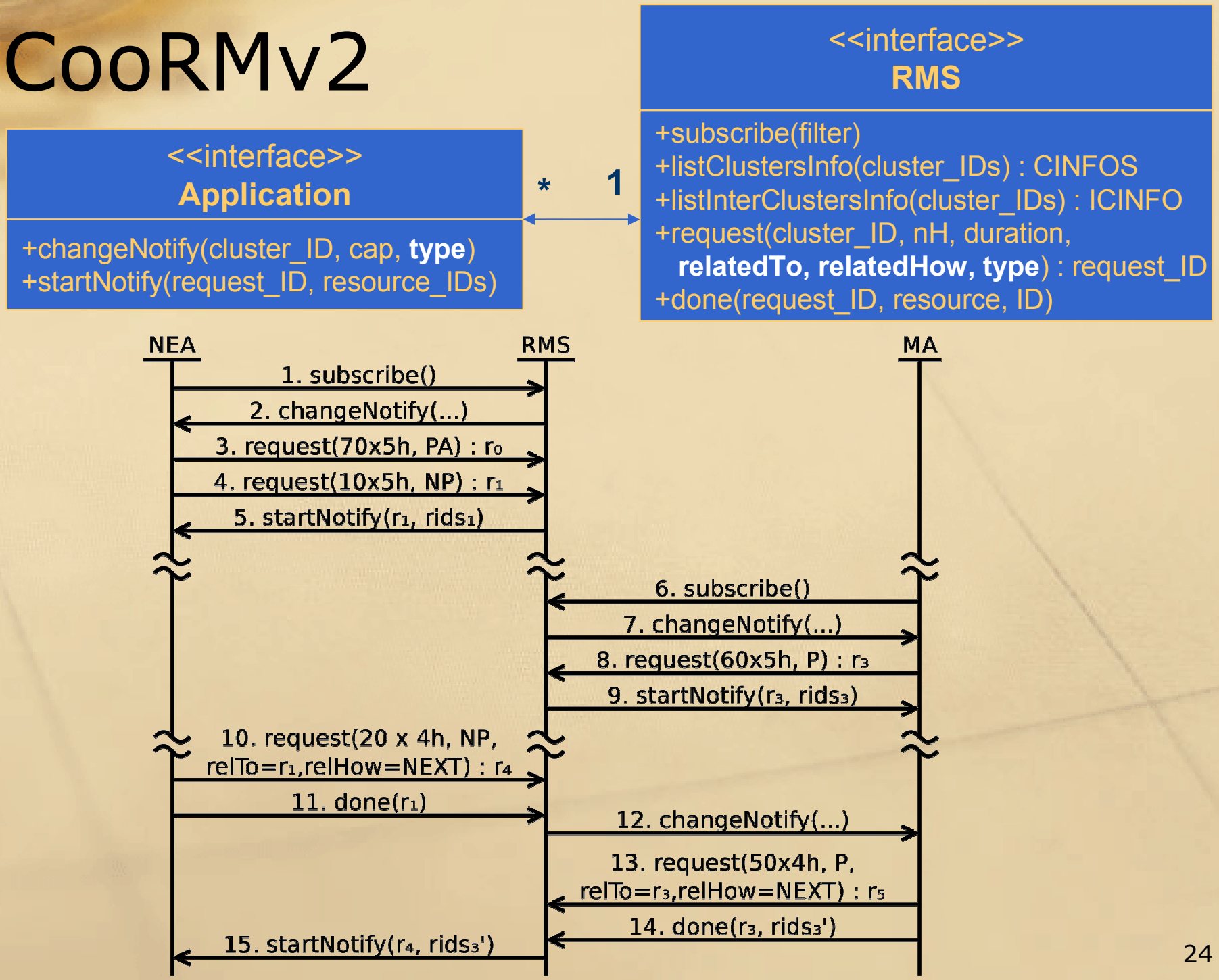
# Limitations of Current RMS

- Malleable jobs
  - RMS tells applications to grow/shrink
- Clouds
  - The illusion of infinite computing resources available on demand”
  - Infinite? Actually up to 20
  - Even without this limit: “Out of capacity” errors
    - Application may run out-of-memory
- Ideally, RMS guarantees the availability of resources to an AMR application?

# CooRMv2

## Supporting Non-Predictably Evolving Applications

# CooRMv2

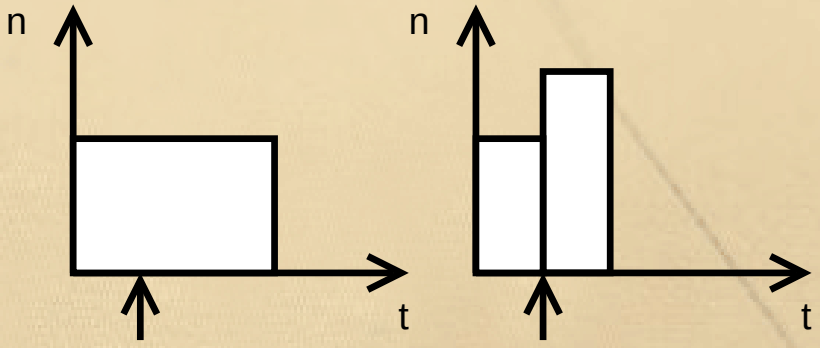

<<interface>>
Application
+changeNotify(cluster_ID, cap, type)
+startNotify(request_ID, resource_IDs)
*
1
<<interface>>
RMS
+subscribe(filter)
+listClustersInfo(cluster_IDs) : CINFOS
+listInterClustersInfo(cluster_IDs) : ICINFO
+request(cluster_ID, nH, duration, relatedTo, relatedHow, type) : request_ID
+done(request_ID, resource, ID)
NEA
RMS
MA
1. subscribe()
2. changeNotify(...)
3. request(70x5h, PA) : r0
4. request(10x5h, NP) : r1
5. startNotify(r1, rids1)
6. subscribe()
7. changeNotify(...)
8. request(60x5h, P) : r3
9. startNotify(r3, rids3)
10. request(20 x 4h, NP, relTo=r1,relHow=NEXT) : r4
11. done(r1)
12. changeNotify(...)
13. request(50x4h, P, relTo=r3,relHow=NEXT) : r5
14. done(r3, rids3')
15. startNotify(r4, rids3')

# Resource Request Model

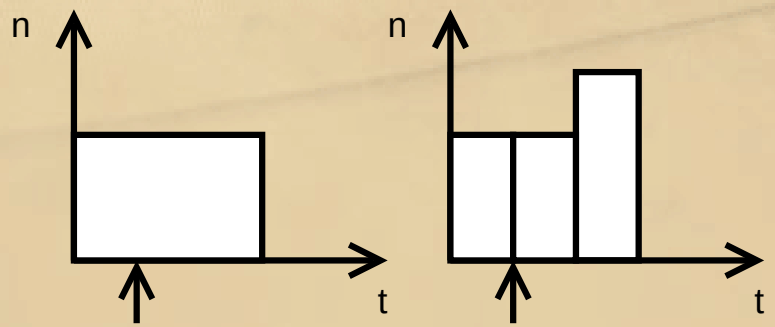
- CooRMv1 extension
- New request parameters
  - Cluster ID, number of nodes, duration, **type**
  - RMS chooses start time
    - Node IDs are allocated to the application
- Type
  - Non-preemptible (default in major RMSs)
  - Preemptible (think OAR best-effort jobs)
  - Pre-allocation
    - "I do not currently need these resources, but make sure I can get them immediately if I need them."

# High Level Operations

- Low level operations
  - Not very expressive but simplify scheduling
- High level operations
  - Spontaneous update*



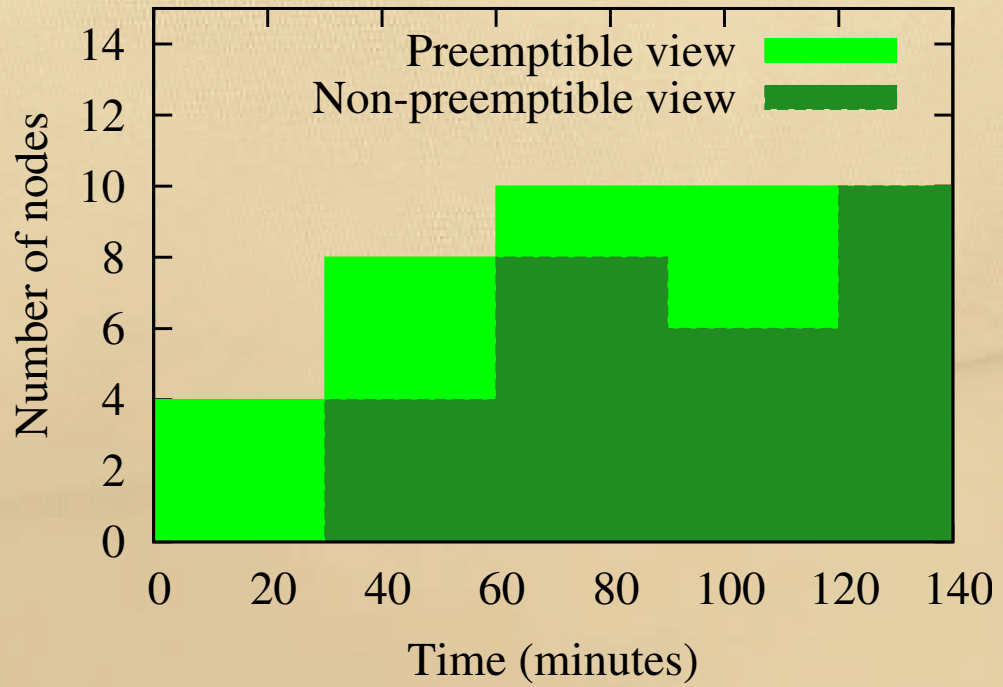

  - Announced updates*




*: Update is only guaranteed within a pre-allocation request.

# Views

- Apps need to adapt their requests to the availability of the resources
- Each app is presented with two views
  - non-preemptible
  - preemptible
- Preemptible view informs when resources need to be preempted

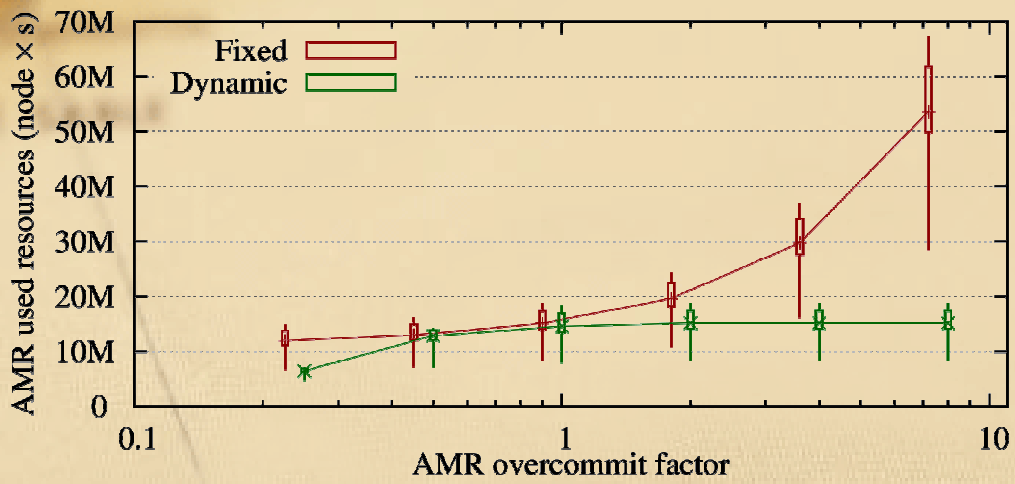


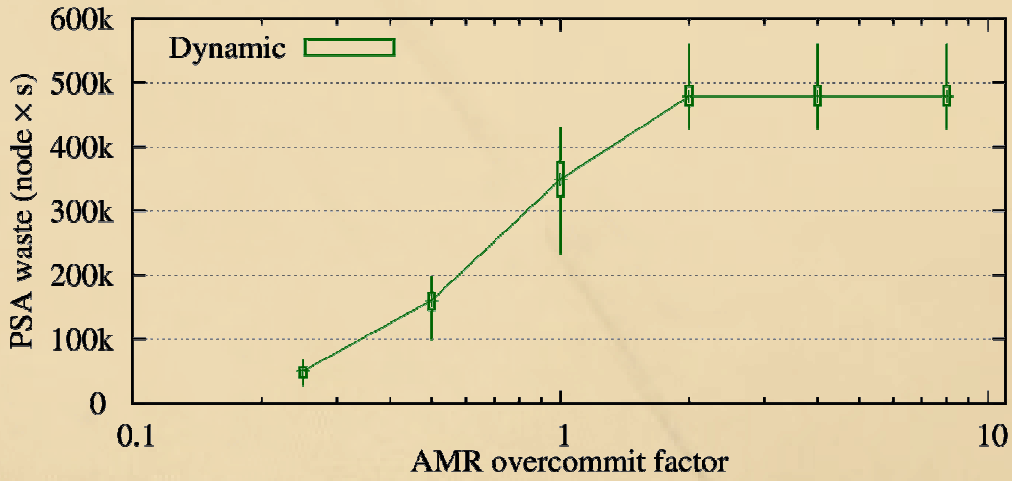
# Scheduling Algorithm

- Pre-allocations and non-preemptible requests
  - Conservative Back-Filling (CBF)
- Preemptible requests
  - Equi-partitioning

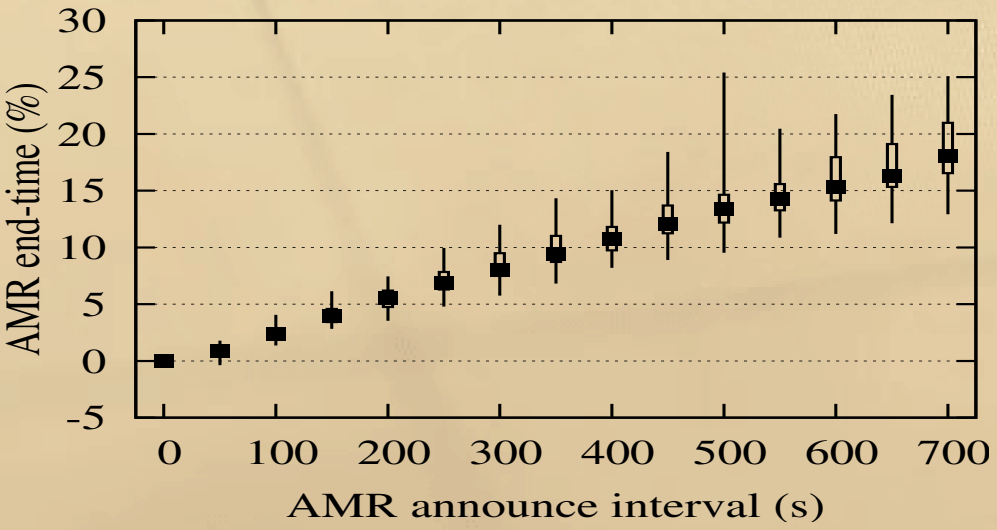
# CooRMv2: AMR & PSA

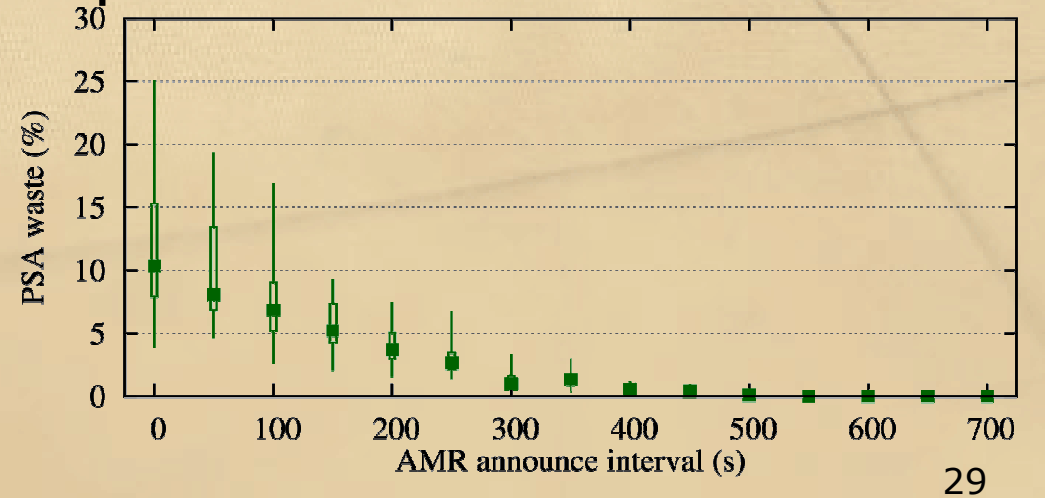
## Spontaneous Updates




## Announced Updates



# Conclusion

# CooRMv1/2

- Simple & Generic RMS–Application Interface
  - Application specific resource selection algorithm
    - Application specific optimization (time, cost, …)
  - Applications with dynamic resource requirements
    - Malleable and evolving
  - Proof of concept implementation
    - Good performance
- Future Work
  - Add (intra-cluster) network topology support
  - Develop (generic) resource selection algorithms for more applications
  - Integrate CooRM concept into existing RMS
    - OAR is planned
  - Is CooRM suitable for distributed systems?
    - DIET? XtreemOS?
  - Is CooRM suitable for Clouds?