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
**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

**Applications**

- Cluster (GPU/MC/...)
- Grids (EGEE)
- Super-computers (Exascale)
- Clouds

**Elasticity**

**Heterogeneity**
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: $n_H^i$, $B_H^i$, $B_C^i$, $a_j^i$
  - Maximum inter-cluster (WAN) latency: $l_{WAN}$

- **One iteration**

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

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

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

\[+ n_{AR} \cdot l_{WAN}\]
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
| to-violette | to-pastel | so-sol | so-helios | so-azur | re-paradent | re-paramount | re-paraquad | re-paravent | or-netgdx | or-gdxl | or-gdx | na-griffon | na-grelon | ly-sagittaire | ly-capricorne | li-chingchint | li-chuque | li-cht1 | li-chicon | gr-genevi | bo-borderline | bo-bordereau | bo-bordeplage | bo-bordemer |
|-------------|-----------|-------|----------|---------|-------------|-------------|------------|------------|-----------|---------|--------|----------|----------|---------------|---------------|---------------|-----------|--------|--------|----------|------------|--------------|--------------|--------------|-----------|

2009-05-25 13:23:01
2009-05-25 14:23:01
2009-05-25 15:23:01
2009-05-25 16:23:01
2009-05-25 17:23:01
2009-05-25 18:23:01
2009-05-25 19:23:01
2009-05-25 20:23:01
2009-05-25 21:23:01
2009-05-25 22:23:01
2009-05-26 00:23:01
2009-05-26 01:23:01
2009-05-26 02:23:01
2009-05-26 03:23:01
2009-05-26 04:23:01
2009-05-26 05:23:01
2009-05-26 06:23:01
2009-05-26 07:23:01
2009-05-26 08:23:01
2009-05-26 09:23:01
2009-05-26 10:27:24
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 $\Leftrightarrow$ 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

Application

Launcher

+changeNotify(CHANGEs)
+startNotify(RID, RTag)

RMS

+subscribe(FILTER)
+listClustersInfo(CIDs) : CINFOS
+listInterClustersInfo(CIDs) : ICINFO
+request(REQ, walltime) : RTag
+finish()

Resource Selection

Resource Selection

<<deploy>>
done()
RMS-side Scheduling

- Fair-start Delay and Ghost Zones

Initial schedule

Job 2 finishes early

Job 3 adapts

App sends subscribe

Submitted

RequestSent

Waiting

Finished

Ghost

FinishSent

Started

App sends request

RMS reschedules

RMS sends startNotify

ghost expires

RMS reschedules

App sends done
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

- Cells updated per time step
- Number of cells vs. Solution time
- Normalized data size vs. Step number
- Duration of a step vs. Number of nodes
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)

<<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)
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

![Graph showing number of nodes over time for preemptible and non-preemptible views.](image)
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?