*dist*CooRM: Distributed Resource Management for Moldable Applications

Cristian KLEIN[1], Christian PÉREZ[1], Yann RADENAC[2]

[1] Avalon, INRIA/LIP, ENS de Lyon, France
[2] Myriads, INRIA/IRISA, France

SimGrid User Days
13–15 Juin 2012
Écully, France

1. Research
    - Problem
    - Solution
    - Results
2. SimGrid
    - Background
    - Attempted Solutions
    - Encountered Problems
    - Workarounds
3. Conclusions and Opinions

1. Research
   - Problem
   - Solution
   - Results
2. SimGrid
   - Background
   - Attempted Solutions
   - Encountered Problems
   - Workarounds
3. Conclusions and Opinions

Context

HPC Resources are Complex

- Data centers feature multiple clusters
  (IN2P3 has 4 clusters)
- Supercomputers feature multiple types of nodes
  (BlueWaters Blueprint: some CPU-only, some CPU+GPU nodes)
- Share among multiple users → availability changes

Context

HPC Resources are Complex

- Data centers feature multiple clusters
  (IN2P3 has 4 clusters)
- Supercomputers feature multiple types of nodes
  (BlueWaters Blueprint: some CPU-only, some CPU+GPU nodes)
- Share among multiple users → availability changes

HPC Applications are Complex

- Moldable
- They feature multiple codes (e.g., fluid, solid)
- Leading to complex performance models

Context

HPC Resources are Complex

- Data centers feature multiple clusters
  (IN2P3 has 4 clusters)
- Supercomputers feature multiple types of nodes
  (BlueWaters Blueprint: some CPU-only, some CPU+GPU nodes)
- Share among multiple users → availability changes

HPC Applications are Complex

- Moldable
- They feature multiple codes (e.g., fluid, solid)
- Leading to complex performance models

How to launch such applications to minimize completion time?

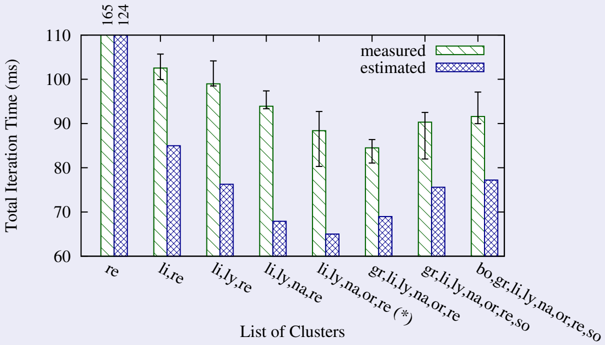
Example: Computational ElectroMagnetics Application

- Ported a CEM application for a multi-cluster execution
- To ensure good performance:
  - need to employ a **custom resource selection algorithms**

→ E. CARON, C. KLEIN, C. PÉREZ, *Efficient Grid Resource Selection for a CEM Application*, RenPar'19, 2009

Example: Computational ElectroMagnetics Application

- Ported a CEM application for a multi-cluster execution
- To ensure good performance:
  - need to employ a **custom resource selection algorithms**



→ E. CARON, C. KLEIN, C. PÉREZ, *Efficient Grid Resource Selection for a CEM Application*, RenPar'19, 2009

Problem

Resource Management Systems **do not provide efficient interfaces**

Problem

Resource Management Systems **do not provide efficient interfaces**

- Moldable jobs (à la OAR)
  - Specify a list of nodes, maximum execution times
  - OAR choses the one than minimizes completion time

Problem

Resource Management Systems **do not provide efficient interfaces**

- Moldable jobs (à la OAR)
  - Specify a list of nodes, maximum execution times
  - OAR choses the one than minimizes completion time
  - For c cluster and n nodes on each cluster ... $n^c - 1$ configurations

Problem

Resource Management Systems **do not provide efficient interfaces**

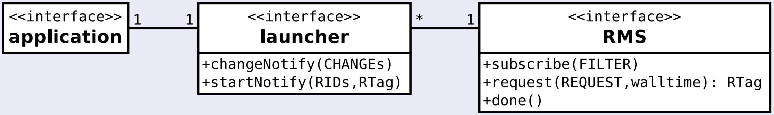
- Moldable jobs (à la OAR)
  - Specify a list of nodes, maximum execution times
  - OAR choses the one than minimizes completion time
  - For *c* cluster and *n* nodes on each cluster ... $n^c - 1$ configurations
- Application-level scheduling
  - Look at currently (or future) available resources
  - Run resource selection algorithm
  - Send resource request (job)

Problem

Resource Management Systems **do not provide efficient interfaces**

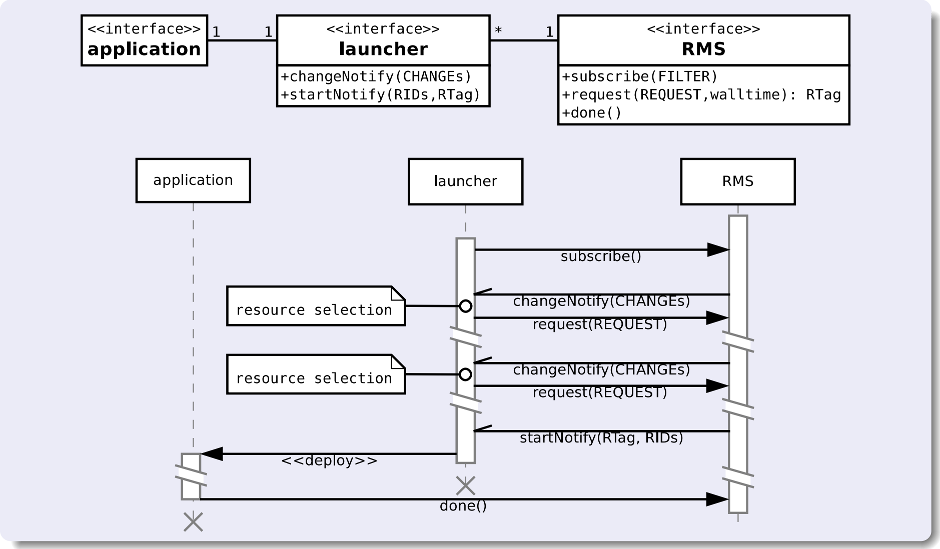
- Moldable jobs (à la OAR)
  - Specify a list of nodes, maximum execution times
  - OAR choses the one than minimizes completion time
  - For c cluster and n nodes on each cluster ... $n^c - 1$ configurations
- Application-level scheduling
  - Look at currently (or future) available resources
  - Run resource selection algorithm
  - Send resource request (job)
  - Need to simulate RMS's scheduling algorithm
  - Cannot update resource request

Centralized Solution: CooRM Architecture



→ C. KLEIN, C. PÉREZ, *An RMS Architecture for Efficiently Supporting Complex-Moldable Applications*, HPCC, 2011

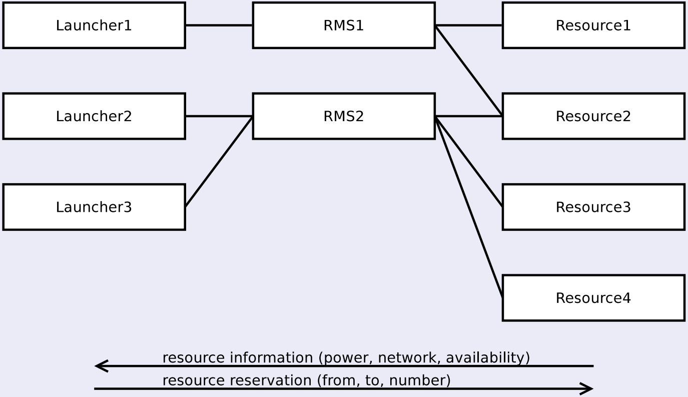
Centralized Solution: CooRM Architecture



→ C. KLEIN, C. PÉREZ, *An RMS Architecture for Efficiently Supporting Complex-Moldable Applications*, HPCC, 2011

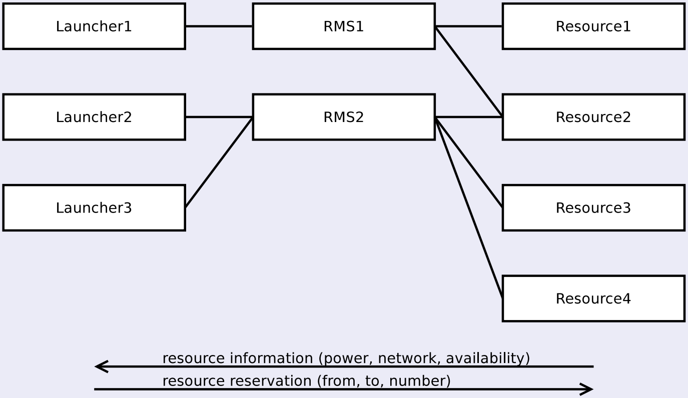
Challenge: Distributed Solution

distCooRM Architecture


Launcher1
RMS1
Resource1
Launcher2
RMS2
Resource2
Launcher3
Resource3
Resource4
resource information (power, network, availability)
resource reservation (from, to, number)

Challenge: Distributed Solution

distCooRM Architecture



Launcher1
RMS1
Resource1
Launcher2
RMS2
Resource2
Launcher3
Resource3
Resource4
resource information (power, network, availability)
resource reservation (from, to, number)

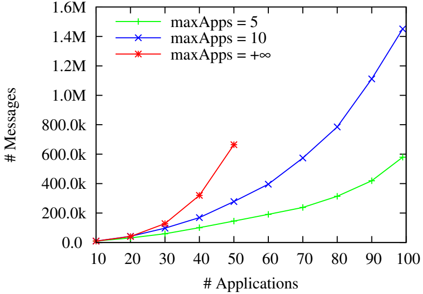
Data structures can be quite complex

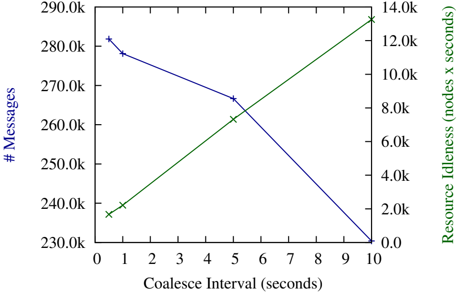
- basic types: int, float, string
- complex types: dictionary of list of pairs of float and int

Evaluation of distCooRM

- Need to check
  - That the system actually works
  - That it scales
  - That it behaves well
- Parameters to explore
  - Platform parameters: timeouts, limits
  - Number of resources
  - Number of applications
- Metrics to measure
  - Number of messages
  - Number of bytes
  - Utilisation of resources

Results


maxApps = 5
maxApps = 10
maxApps = +∞
# Messages
1.6M
1.4M
1.2M
1.0M
800.0k
600.0k
400.0k
200.0k
0.0
10
20
30
40
50
60
70
80
90
100
# Applications


# Messages
290.0k
280.0k
270.0k
260.0k
250.0k
240.0k
230.0k
14.0k
12.0k
10.0k
8.0k
6.0k
4.0k
2.0k
0.0
Resource Idleness (nodes x seconds)
0
1
2
3
4
5
6
7
8
9
10
Coalesce Interval (seconds)

1. Research
    - Problem
    - Solution
    - Results

2. SimGrid
    - Background
    - Attempted Solutions
    - Encountered Problems
    - Workarounds

3. Conclusions and Opinions

Background

Requirements

- Reusable code: simulator and real implementation
- Event-driven, events are message arrived, timeouts
- RPC-like interaction between agents
- 1000 agents, **each agents talks to 1–1000 other agents**
- Simulator
  - Host model: sleep
  - Network model: Vivaldi latencies (messages are less than 100 bytes)
  - Allow measuring network traffic
- Real implementation
  - Efficient serialization (like CORBA)

Background

Requirements

- Reusable code: simulator and real implementation
- Event-driven, events are message arrived, timeouts
- RPC-like interaction between agents
- 1000 agents, **each agents talks to 1–1000 other agents**
- Simulator
  - Host model: sleep
  - Network model: Vivaldi latencies (messages are less than 100 bytes)
  - Allow measuring network traffic
- Real implementation
  - Efficient serialization (like CORBA)

Development Environment

- C++
- sockets, MPI, CORBA

Attempted Solutions (1/2)

XBT

Attempted Solutions (1/2)

XBT

- ☹ Useless, cumbersome (when compared to STL, Boost)
- ☹ Some APIs force the user into using XBT

Attempted Solutions (1/2)

XBT

- ☹ Useless, cumbersome (when compared to STL, Boost)
- ☹ Some APIs force the user into using XBT

GRAS

- ☺ Code once, use twice
- ☹ Difficult to use for complex structures (“IDL” parser is buggy)
- ☹ Cannot measure network traffic

Attempted Solutions (2/2)

MSG

- ☺ Easy to create processes
- ☺ Includes P2P network model: Vivaldi coordinates
- 😐 Only useful for simulator

Attempted Solutions (2/2)

MSG

- ☺ Easy to create processes
- ☺ Includes P2P network model: Vivaldi coordinates
- 😐 Only useful for simulator
- ☹ Principle of Least Astonishment
    - What does one send and receive?

Attempted Solutions (2/2)

MSG

- ☺ Easy to create processes
- ☺ Includes P2P network model: Vivaldi coordinates
- 😐 Only useful for simulator
- ☹ Principle of Least Astonishment
  - What does one send and receive? **tasks**

Attempted Solutions (2/2)

MSG

- ☺ Easy to create processes
- ☺ Includes P2P network model: Vivaldi coordinates
- 😐 Only useful for simulator
- ☹ Principle of Least Astonishment
  - What does one send and receive? **tasks**
  - What does **isend** do?
  - What about **dsend**?
  - Is it `MSG_*`, `m_*`? **m_task_t**, **MSG_task_isend**, **MSG_error_t**.

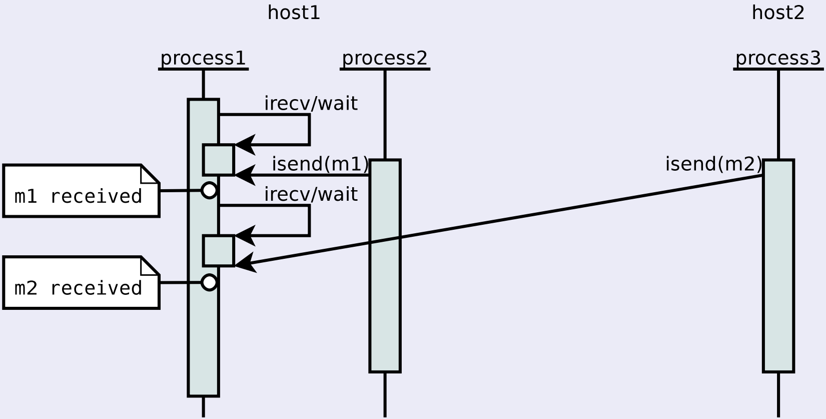
Attempted Solutions (2/2)

MSG

- ☺ Easy to create processes
- ☺ Includes P2P network model: Vivaldi coordinates
- 😐 Only useful for simulator
- ☹ Principle of Least Astonishment
  - What does one send and receive? **tasks**
  - What does **isend** do?
  - What about **dsend**?
  - Is it `MSG_*`, `m_*`? **m_task_t**, **MSG_task_isend**, **MSG_error_t**.
- ☹ Need to create separate network process which handles queue (due to `MSG_comm_wait`)
- ☹ Network "semantics" make it unusable

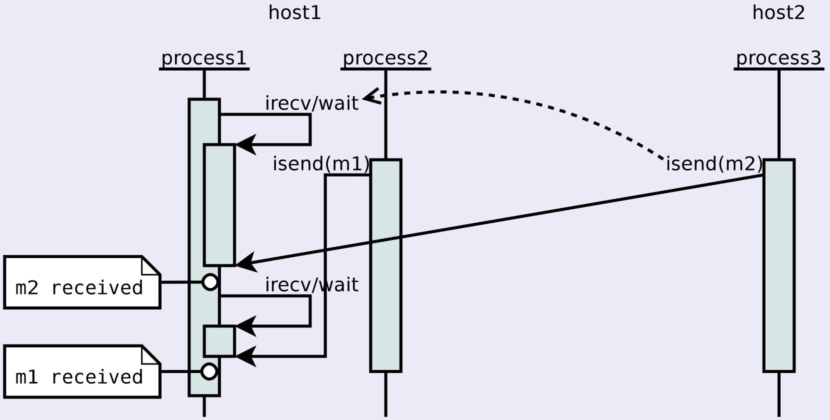
MSG Communication Semantics

Expected



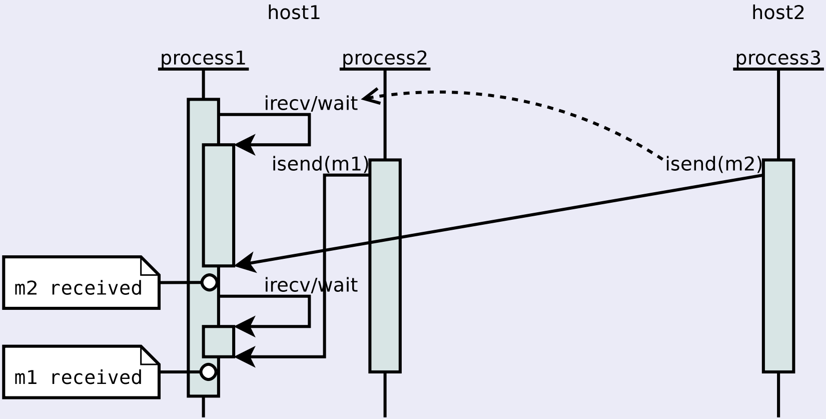
MSG Communication Semantics

Actual



MSG Communication Semantics

Actual



- Gives incorrect transmission times

Workarounds

Post multiple `irecvs`

- 1000 agents × 1000 possible sources = 1,000,000 `irecvs`
- Simulation significantly slowed down

Own Implementations

- Custom `irecv`/`isend`
- Uses `xbt_mutex_t`, `xbt_cond_t` and global variables
- Hackish, but works ☺

1. Research
   - Problem
   - Solution
   - Results

2. SimGrid
   - Background
   - Attempted Solutions
   - Encountered Problems
   - Workarounds

3. Conclusions and Opinions

Conclusions and Opinions

distCooRM

- HPC applications require specialized resource selection algorithms
- RMS do not provide adequate interfaces
- distCooRM (will) be a distributed solution

Conclusions and Opinions

distCooRM

- HPC applications require specialized resource selection algorithms
- RMS do not provide adequate interfaces
- distCooRM (will) be a distributed solution

SimGrid

- ☹ Network model was found to be error-prone and not scalable
- 😐 SimGrid did not prove as useful as expected, but eventually did its job
- ☺ Model checking seems promising
- ☺ The community provided invaluable support

Conclusions and Opinions

distCooRM

- HPC applications require specialized resource selection algorithms
- RMS do not provide adequate interfaces
- distCooRM (will) be a distributed solution

SimGrid

☹ Network model was found to be error-prone and not scalable

😐 SimGrid did not prove as useful as expected, but eventually did its job

☺ Model checking seems promising

☺ The community provided invaluable support

Opinions

- Do not force user to use XBT
- Intuitive communication abstractions:
  datagram (UDP-like), stream (TCP-like), remote call (CORBA-like)