Rendering Grid Heterogeneity Harmless

Marco Aldinucci
ISTI - National Research Council
Pisa - Italy

with S. Campa, M. Danelutto, C. Zoccolo
Rendering Grid *(performance)*
Heterogeneity *(mostly)*
Harmless

*Marco Aldinucci*

ISTI - National Research Council
Pisa - Italy

with S. Campa, M. Danelutto, C. Zoccolo
Motivation

Grid as collection of heterogeneous resources
- Presenting experimental results
- A simple, even simplistic model
- Defining the asymptotic performance

Detect Grid current status and react

Re-distributing work & load through WS
Researchers in the Grid community hardly agree
- programming model (and either if it should exists)
- components (and either if they are an useful vehicle)
- legacy code existence ...

but them all agree
- THE GRID IS A HIGHLY HETEROGENEOUS, HIGHLY DYNAMIC EXECUTION ENVIRONMENT
However ...

<table>
<thead>
<tr>
<th></th>
<th>Experimental results on &gt;2 PEs</th>
<th>Performance figures for &gt;2 heterogeneous PEs</th>
</tr>
</thead>
<tbody>
<tr>
<td>EuroPar 04 (Grid &amp; P2P) LNCS 3149</td>
<td>2–4/20</td>
<td>1? (as far I known)</td>
</tr>
<tr>
<td>Grid Computing 04 LNCS 3165</td>
<td>3/30</td>
<td>No (as far I known)</td>
</tr>
<tr>
<td>Grid &amp; Cooperative Computing 04 LNCS 3251</td>
<td>6/150</td>
<td>No (as far I known)</td>
</tr>
</tbody>
</table>

- How many platforms GTx supports?
- Java seems to be the panacea for heterogeneity:
  - Maybe we relying too much on Sun’s researchers
- Look at conference proceedings:
  - few of them present experimental results
  - very few of them present result for heterogeneous environments
- we agreed on heterogeneity, thought
Testbed:

\texttt{wait(4*365*24*60*60);} 
\texttt{unfortunately();}
blur() it
These can be parallel as well.
Why speedup is important

- We would like to deploy HPC applications on Grid
  - not just seti@home
  - they may have time/performance/memory/... critical requirements
- Known in advance what I can expect from my run, at least as asymptotically optimal curve
  - speedup for example (widely used in COW)
  - any measure able to give information on the quality of the algorithm, implementation, configuration, ...
Experimental env: a home-made Grid
BogoPower:
- Models machine power on (tasks/sec) on a single PE
- Neglect net performance
- What speedup means in this scenario?
- Another metric is needed ...

P2@233MHz
P3@1.1GHz
G4@800MHz
G4@867MHz
P4@1.7GHz
P4@2.8GHz
2xP3@550MHz
2xP4@800MHz
2xG5@2GHz
4xP4@2.8GHz
Two experiments

\[ \pm 1400\% \]
Two experiments

±1400%
Speedup ... ?

REJECTED

Ideal

Ideal
As simple as speedup

- Speedup does not give any information
  - does not provide any reference curve, i.e. an upper bound for algorithm and implementation quality
- It can be replaced with another simple measure
  - with the same features in order to keep the intuition
  - suitable for heterogeneous (in power) envs
- BogoPower can be used (sometime)
Two experiments revised
Naive scheduling (and not)

\[ n_1 = n_2 = n_3 = n_4 = 7 \]

\[ T_1 = T_2 = 3 \quad T_3 = 2 \quad T_4 = 1 \]

\[ n_1 = 5 \quad n_2 = 5 \quad n_3 = 7 \quad n_4 = 12 \]

\[ T_1 = T_2 = 3 \quad T_3 = 2 \quad T_4 = 1 \]
Describing sub-optimal Performance

Suppose to have an idea of the performance $T$ (time) of a given task $T$ on a given platform

- i.e. platform BogoPower - it maybe figured out from any suitable measure of performance, e.g. GridBench, GGF BenchGroup, ...
- if task haven’t constant time consider the average of a bulk of tasks
- dynamically adapt knowledge through monitoring, adjusted by current load

compute a scheduling, miming on-demand policy

- that is sub-optimal, but easy to compute, to understand and to present as “ideal” performance in a paper

$$n_i = \frac{N H(T_1, \ldots, T_n)}{n T_i}$$

- $N = \#$ of tasks
- $H = \text{Harmonic Mean}$
- $T_i = \text{Time for 1 task on } PE_i$
- $n_i = \text{optimal number of tasks for } PE_i$
Motivation

Grid as collection of heterogeneous resources

Detect Grid current status and react
- The ASSIST framework
- A service to find them, a GTx to bring them all and in the darkness bind them, a model to rule them all ...

Re-distributing work & load through WS

It is not a joke, it is e-fantasy!
The ASSIST way

from Danelutto’s yesterday talk
Managers may interact through non-functional interfaces realizing a global, distributed control for the application. I.e. the Application Manager (AM)

Manager's use-provides RPC-style dependencies

Messages may flow through different media: native, WS/SOAP, Corba streams (compiler provides the full support)
Parmod component

non functional  events  streams

VP  VP  VP

uses  provides

Shared global persistent state

GAM
Run & Monitor
Check the strategy
Possibly interact with other Parmod Managers
Parmod component

non functional events streams

Run & Monitor
Check the strategy
Possibly interact with other Parmod Managers
Make a decision (local or global)
Reconfigure the Parmod
Parmod component

1. Run & Monitor
2. Check the strategy
3. Possibly interact with other Parmod Managers
4. Make a decision (local or global)
5. Reconfigure the Parmod
6. Parmod reconfigured!

This maybe is e-autonomous-computing (even if since yesterday I did not known it)
Two key issues

2 Check the strategy
- simulate possible scenarios by using the suitable model
  - respect the performance contract: service time, resources, ...
  - e.g. the one I’ve presented seems quite efficient for HPF “do parallel” or BSP style computations
- we already working to support other paradigms

5 Reconfigure the parmod
- keep the shared state in a “storage component” that is distributed, persistent, WS accessible, high-performance
- E.g. HOC / WS-HOC already available as part of ASSIST
... and it work
Motivation

Grid as collection of heterogeneous resources

Detect Grid current status and react

Re-distributing work & load through WS
  - decouple management of data e computation
  - the “storage component” idea
Redistribute data

non functional events streams

Monitor Actuator Strategy AM cooperation

Parmod Manager

uses provides

WS Native Interface

Storage Component (distributed, HOC-based)
HOC (Herd of Object Caches)

- A very basic storage facility
  - No hardwired policies for deployment, allocation, data coherence, ...
  - pluggable into different, third-party applications/frameworks
- proving **data management** as external service for applications
  - implemented as high-throughput distributed server
- decoupling computational and storage management in (distributed) application design
  - enforcing a structured development
- and exploiting persistency, scalability, re-configurability
Permanent, shared storage facility

*a facility (distributed server)* providing permanent, shared storage to apps (clients)

*clients may dynamically join/leave the storage facility*

*HOC set may be hotly enlarged/reduced on need - storage room change accordingly*

*interaction with HOCs may be delegated to application-specific protocol*
Why using HOC

- is efficient (because essential)
  - HOC provide few primitives and no policies for data integrity (e.g. coherence, consistency, ...)
  - these are application specific and may be deployed upon HOC (at the protocol level)

- is a basic building block for broad class of applications
  - may be considered a storage component
  - massive storage, out-of-core applications, high-throughput data servers, shared memory support
  - extendible with application-specific primitives

- enhances both memory size and throughput by means of parallelism
... using HOC

- Protocol enforces application requirements on data integrity acting as mediator between the application and HOC.
- It is linked to the application and uses HOC API.
- E.g., Apache module.

Protocol may actually be a distributed application (e.g., reaching consensus, cache invalidation, ...)

HOC API may also be easily extended (provided some knowledge of HOC internals).
HOC internals

- C++, single-threaded, manage concurrent connections using non-blocking I/O based services (each of them being a state machine managing a single connection)
  - supporting both level-triggered (select, poll, ...) and edge-triggered (RTsignal, kqueue, ...) I/O events
- object storage may be managed either as a memory or a cache, remote objects may be cached in a separate write-through cache. Policies are configurable.
- tested on Linux, MacOS X, and heterogeneous cluster of them
Why does the web work so well?
A language with few verbs (get, put, post) ... Gannon said ... (Europar04, invited talk)

We also believe on such philosophy. As matter of a fact HOC have a four operations API

- get, put, remove arbitrary length objects. Each object is identified by a key and a home node

- execute(key, op, data) remotely execute method op with parameter data on object identified by key
## Performance figures (1PE)

<table>
<thead>
<tr>
<th>Arch/Net/OS</th>
<th>concurrent connections</th>
<th>Msg size (Bytes)</th>
<th>Replies/Sec</th>
<th>net throughput (Bytes/Sec)</th>
<th>net throughput w.r.t. ideal</th>
</tr>
</thead>
<tbody>
<tr>
<td>P4@2GHz Mem 512MB GigaEth Linux ker. 2.4.22</td>
<td>2048</td>
<td>1 M</td>
<td>91</td>
<td>91 M</td>
<td>96%</td>
</tr>
<tr>
<td>P3@800MHz Mem 1GB FastEth Linux ker. 2.4.18</td>
<td>1024</td>
<td>8 K</td>
<td>1429</td>
<td>11.2 M</td>
<td>90%</td>
</tr>
</tbody>
</table>
Speedup (Hit per sec VS N. servers)

- 8K objects (perfect)
- 8K objects
- 16K objects (perfect)
- 16K objects
Sustained aggregate throughput

Processing Elements

Aggregate throughput (MB/sec)

Net Asymptotic throughput
8K objects
16K objects

90% 90%
88% 88%
88% 88%
85% 86%
88% 88%
88% 88%
85% 85%
75% 85%
HOC is a building block for storage-oriented components:
- distributed caches, distributed memories, parallel repositories
- configurable, hot-pluggable,

very good performances:
- close-to-ideal net throughput over thousands of concurrent connections
- close-to-ideal speedup
Conclusions

- A simple model able to describe what we can expect from our Grid applications
  - Usable as “Ideal performance” slope in papers
- A first effort toward a serious AM
  - A very ongoing work, as asked by Alexander
- Exploit the potentiality of ASSIST+WS+StorageComponent
THANK YOU!
QUESTIONS?
Some references

  ASSIST as a research framework for high-performance Grid programming environments. 

  Components for high-performance Grid programming in the Grid.it project. 

  Targeting interoperability and heterogeneous architectures in ASSIST. 

  The Implementation of ASSIST, an Environment for Parallel and Distributed Programming. 

- M. Aldinucci, M. Torquati. 
  Accelerating Apache farms through ad-HOC distributed scalable objects repository. 

- M. Aldinucci, M. Danelutto, J. Dünnweber, S. Gorlatch. 
  Optimization techniques for implementing parallel skeletons in Grid. 