Elastic desktop grids with (J)OCaml

William Le Ferrand

April 16, 2010
Naclgrid-ec2, an elastic desktop grid engine

1. Introduction
2. Motivations
3. Distribution process
4. Implementation
5. Deployment
6. Conclusion
Context

Desktop grid

grid built with heterogenous resources
Context

**Desktop grid**
grid built with heterogenous resources

**Success cases**
Folding@home, Seti@home : BOINC-based grids.
Naclgrid-ec2, an elastic desktop grid engine

1. Introduction
2. Motivations
3. Distribution process
4. Implementation
5. Deployment
6. Conclusion
Aggregate various powersources

- Dedicated servers
Aggregate various powersources

- Dedicated servers
- Local computers (from the administration, etc etc)
Aggregate various powersources

- Dedicated servers
- Local computers (from the administration, etc etc)
- Extra servers rented/started on the fly
New constraints need new software

- flexibility: we have to be able to add nodes in the node on the fly, during the job
New constraints need new software

- flexibility: we have to be able to add nodes in the node on the fly, during the job
- security: computers might not be dedicated to HPC, we should be able to use their power without risk
New constraints need new software

- flexibility: we have to be able to add nodes in the node on the fly, during the job
- security: computers might not be dedicated to HPC, we should be able to use their power without risk
- heterogeneity: computers are not clones, nodes might be connected through the internet, loose connection
New constraints need new software

- flexibility: we have to be able to add nodes in the node on the fly, during the job
- security: computers might not be dedicated to HPC, we should be able to use their power without risk
- heterogeneity: computers are not clones, nodes might be connected through the internet, loose connection
- simplicity: scalability, easy to port existing software
What are we going to discuss?

- Distribution process
What are we going to discuss?

- Distribution process
- Overall architecture
What are we going to discuss?

- Distribution process
- Overall architecture
- An application of naclgrid-ec2: the corefarm
Naclgrid-ec2, an elastic desktop grid engine

1. Introduction
2. Motivations
3. Distribution process
4. Implementation
5. Deployment
6. Conclusion
Workflow

- Each node can start a job

```ocaml
1 | type task = { application: string; entry: string; input: string }
```
Workflow

- Each node can start a job
- Nodes can drop off tasks on the server and retrieve results

```ocaml
type task = { application: string; entry: string; input: string }
```
Workflow

- Each node can start a job
- Nodes can drop off tasks on the server and retrieve results
- Nodes (when idle) can fetch tasks and return results to the server

```ocaml
1 | type task = { application: string; entry: string; input: string }
```
Workflow

- Each node can start a job
- Nodes can drop off tasks on the server and retrieve results
- Nodes (when idle) can fetch tasks and return results to the server

1 | type task = { application: string; entry: string; input: string }
Workflow

- Each node can start a job
- Nodes can drop off tasks on the server and retrieve results
- Nodes (when idle) can fetch tasks and return results to the server

```ocaml
1 | type task = { application: string; entry: string; input: string }
```

pros

- Simple
- More flexible than MapReduce
- Fits the constraints
Workflow

- Each node can start a job
- Nodes can drop off tasks on the server and retrieve results
- Nodes (when idle) can fetch tasks and return results to the server

```ocaml
1 | type task = { application: string; entry: string; input: string }
```

<table>
<thead>
<tr>
<th>pros</th>
<th>cons</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simple</td>
<td>Risk of deadlock</td>
</tr>
<tr>
<td>More flexible than MapReduce</td>
<td>Inertness</td>
</tr>
<tr>
<td>Fits the constraints</td>
<td></td>
</tr>
</tbody>
</table>
## Life and death of a job

<table>
<thead>
<tr>
<th>Nacgrid server</th>
<th>Node A</th>
<th>Node B</th>
<th>Node C</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 job</td>
<td>idle</td>
<td>idle</td>
<td>idle</td>
</tr>
<tr>
<td>0 job</td>
<td>initiate a job</td>
<td>idle</td>
<td>idle</td>
</tr>
<tr>
<td>1 job with 0 tasks and 0 results</td>
<td>compute alone</td>
<td>idle</td>
<td>idle</td>
</tr>
<tr>
<td>1 job with 0 tasks and 0 results</td>
<td>fork 2 tasks</td>
<td>idle</td>
<td>idle</td>
</tr>
<tr>
<td>1 job with 2 tasks and 0 results</td>
<td>compute</td>
<td>process a task</td>
<td>process a task</td>
</tr>
<tr>
<td>1 job with 2 results</td>
<td>fetch results</td>
<td>idle</td>
<td>idle</td>
</tr>
<tr>
<td>1 job</td>
<td>closing job</td>
<td>idle</td>
<td>idle</td>
</tr>
<tr>
<td>0 job</td>
<td>idle</td>
<td>idle</td>
<td>idle</td>
</tr>
</tbody>
</table>
### Naclgrid REST API

<table>
<thead>
<tr>
<th>Uri</th>
<th>GET params</th>
<th>POST params</th>
<th>result</th>
</tr>
</thead>
<tbody>
<tr>
<td>new_job</td>
<td></td>
<td>application</td>
<td>job_id</td>
</tr>
<tr>
<td>drop_file</td>
<td>job_id</td>
<td>file</td>
<td></td>
</tr>
<tr>
<td>start_job</td>
<td>job_id</td>
<td>input</td>
<td></td>
</tr>
<tr>
<td>fork</td>
<td>job_id</td>
<td>entry, input</td>
<td></td>
</tr>
<tr>
<td>fetch</td>
<td>job_id</td>
<td>slot</td>
<td></td>
</tr>
<tr>
<td>stop_job</td>
<td>job_id</td>
<td>result</td>
<td></td>
</tr>
<tr>
<td>pop</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ack</td>
<td>job_id</td>
<td>slot, result</td>
<td></td>
</tr>
</tbody>
</table>

Additionnally, each connector is protected by credentials. Of course developers don’t directly use this API but the C / C++ / OCaml APIs.
What can you do with this pattern?

- Data mining
- 3D rendering

<table>
<thead>
<tr>
<th>Time</th>
<th>Master node</th>
<th>Agents</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>preprocessing</td>
<td>idle</td>
</tr>
<tr>
<td>2</td>
<td>build kd-tree</td>
<td>idle</td>
</tr>
<tr>
<td>3</td>
<td>split image in tiles</td>
<td>idle</td>
</tr>
<tr>
<td>4</td>
<td>process tiles</td>
<td>process tiles</td>
</tr>
<tr>
<td>5</td>
<td>assemble results</td>
<td>idle</td>
</tr>
</tbody>
</table>
Naclgrid-ec2, an elastic desktop grid engine

1. Introduction
2. Motivations
3. Distribution process
4. Implementation
5. Deployment
6. Conclusion
Components

User management
Authentication - Billing - Notifications

Resources Manager
Volunteers
EC2 instances

Job manager
Running job queue - Pending job queue

Tracker
Failover - Fraud control - Logging

Web server (Oxygen)

Networking layer
x86 Sandbox (Google Native Client)
Generic agent (volunteers)

Specialized agent (ec2 instance)
Host application (ex: Yafaray)

Host application (ex: Yafaray)

Networking layer
Client side

Current client is Naclys (http://code.google.com/p/naclys/)

- Sandboxing with the Google Native client (http://code.google.com/p/nativeclient/)
- C++ code to get contributions (QT gui)
Server side

Server side is implemented as a module for Ocsigen (www.ocssigen.org).
JOCaml implementation. JOCaml (http://jocaml.inria.fr/) is OCaml + join calculus.

- concurrent features to organize job queues
- distribution features enforce the stability
Naclgrid-ec2, an elastic desktop grid engine

1. Introduction
2. Motivations
3. Distribution process
4. Implementation
5. Deployment
6. Conclusion
OCaml in the cloud
The collaborative rendering farm

Corefarm

Cooperative rendering farm for distributed ray tracing

New: Corefarm port for the Wosoggie

An experimental port of Corefarm is available for the Wosoggie. The Wosoggie is a new desktop grid powered by Tempestfactory. Check it out.

What is Corefarm?

Corefarm is a place where you can either submit your Yafaray rendering job or participate to other users' rendering tasks. Corefarm splits your job in several small pieces and distributes them across the internet, leading to impressive rendering times. Learn more.

Submit your job to the grid

Anyone can submit his Yafaray rendering job to the grid. All you have to do is to upload your files and let the grid dispatch the load. Show me how.

Participate to the grid

Your PC is idle? Participate to the grid safely simply by browsing a web page. Thanks to the technology used, there is no risk for your computer and no installation is required. Tell me more.
Several frontends for naclgrid-ec2

The collaborative rendering farm

Rendering desktop grid, based on Yafaray
Several frontends for naclgrid-ec2

The collaborative rendering farm

- Rendering desktop grid, based on Yafaray
- 700+ registered users
Several frontends for naclgrid-ec2

The collaborative rendering farm

- Rendering desktop grid, based on Yafaray
- 700+ registered users
- 30M seconds exchanged
Several frontends for naclgrid-ec2

The collaborative rendering farm

- Rendering desktop grid, based on Yafaray
- 700+ registered users
- 30M seconds exchanged

Your application?

Almost anyone can develop for the grid and take advantage to the elastic power plant. The developer website is not ready yet, but we can already get in touch!
Naclgrid-ec2, an elastic desktop grid engine

1. Introduction
2. Motivations
3. Distribution process
4. Implementation
5. Deployment
6. Conclusion
Why OCaml?

- Development time is shorter (compared to C++)
- Clean software, easy to explain to non-developers

But it’s difficult to find contributors.. (and to hire?)
Thanks to Gallium and PPS for those amazing tools!
Questions

william@corefarm.org - www.corefarm.org