

Elastic desktop grids with (J)OCaml

William Le Ferrand

April 16, 2010

Naclgrid-ec2, an elastic desktop grid engine I

- 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 I

- 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 I

- 1 Introduction
- 2 Motivations
- 3 Distribution process**
- 4 Implementation
- 5 Deployment
- 6 Conclusion

Workflow

- Each node can start a job

```
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

```
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

```
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 }
```

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

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

pros

- Simple
- More flexible than MapReduce
- Fits the constraints

cons

- Risk of deadlock
- Inertness

Life and death of a job

| Nacgrid server | Node A | Node B | Node C |
|----------------------------------|----------------|----------------|----------------|
| 0 job | idle | idle | idle |
| 0 job | initiate a job | idle | idle |
| 1 job with 0 tasks and 0 results | compute alone | idle | idle |
| 1 job with 0 tasks and 0 results | fork 2 tasks | idle | idle |
| 1 job with 2 tasks and 0 results | compute | process a task | process a task |
| 1 job with 2 results | fetch results | idle | idle |
| 1 job | closing job | idle | idle |
| 0 job | idle | idle | idle |

Naclgrid REST API

| Uri | GET params | POST params | result |
|-----------|------------|--------------|--------|
| new_job | | application | job_id |
| drop_file | job_id | file | |
| start_job | job_id | input | |
| fork | job_id | entry, input | |
| fetch | job_id | slot | result |
| stop_job | job_id | | |
| pop | | | task |
| ack | job_id | slot, result | |

Additionally, 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?

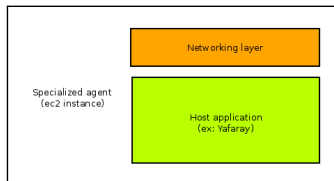
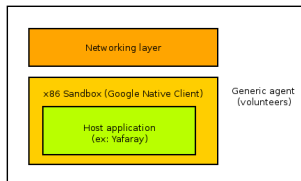
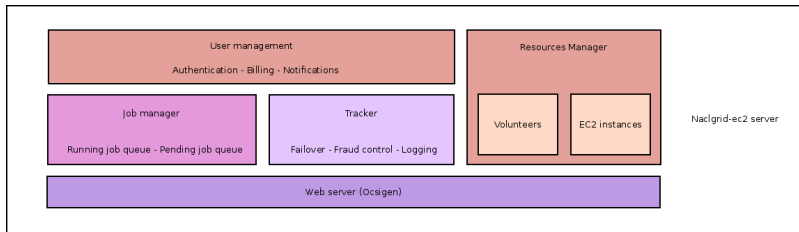
- Data mining
- 3D rendering

| Time | Master node | Agents |
|------|----------------------|---------------|
| 1 | preprocessing | idle |
| 2 | build kd-tree | idle |
| 3 | split image in tiles | idle |
| 4 | process tiles | process tiles |
| 5 | assemble results | idle |

Naclgrid-ec2, an elastic desktop grid engine I

- 1 Introduction
- 2 Motivations
- 3 Distribution process
- 4 Implementation**
- 5 Deployment
- 6 Conclusion

Components



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.ocsigen.org).

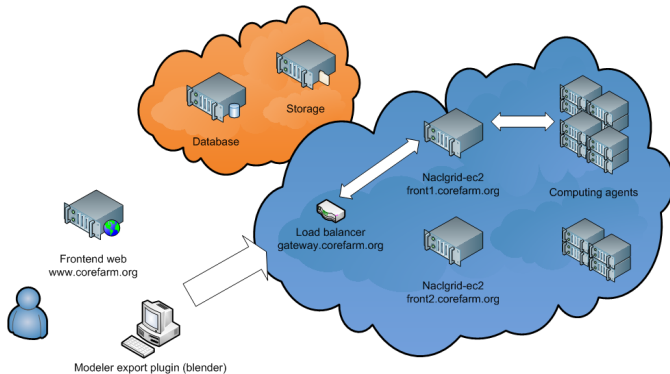
JOCaml implementation. JOCaml (<http://jocaml.inria.fr/>) is OCaml + join calculus.

- concurrent features to organize job queues
- distribution features enforce the stability

Naclgrid-ec2, an elastic desktop grid engine I

- 1 Introduction
- 2 Motivations
- 3 Distribution process
- 4 Implementation
- 5 Deployment**
- 6 Conclusion

OCaml in the cloud



The collaborative rendering farm



The screenshot shows the Corefarm website interface. At the top, there is a header with the Corefarm logo (a stylized orange 'C' with a cow head) and the text 'Alpha Cooperative rendering farm for distributed ray tracing'. Below the header is a navigation bar with links: Home, Download, Forum, Status, Feedback, Join, and Login. A central box highlights a 'New: Corefarm port for the Wosoggie' announcement, stating that an experimental part of Corefarm is available for the Wosoggie, a new desktop grid powered by Thimpfactory, and provides a link to 'Check it out.' Below this, there are three main sections: 'What is Corefarm?' which explains that users can submit Yafaray rendering jobs or participate in other users' tasks, and that the farm splits jobs into small pieces distributed across the internet; 'Submit your job to the grid' which states that anyone can submit a Yafaray rendering job to the grid by uploading files and letting the grid dispatch the load, with a link to 'Show me how.'; and 'Participate to the grid' which encourages users to participate if their PC is idle, noting that no installation is required and providing a link to 'Tell me more.'

Corefarm.Alpha
Cooperative rendering farm for distributed ray tracing

Home Download Forum Status Feedback Join Login

New: Corefarm port for the Wosoggie

An experimental part of Corefarm is available for the Wosoggie. The Wosoggie is a new desktop grid powered by Thimpfactory. [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 I

- 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

Thanks to Gallium and PPS for those amazing tools!

Questions



william@corefarm.org - www.corefarm.org