AlGorille Team
Axis on “Experimentation of Distributed Systems”

Martin Quinson & Lucas Nussbaum

ALGORILLE team

24 octobre 2011
Our Scientific Objects: Distributed Systems

Scientific Computing: High Performance Computing / Computational Grids
- Infrastructure underlying *Computational science*: Massive / Federated systems
- Main issues: Have the world’s biggest one / compatibility, trust, accountability

Cloud Computing
- Large infrastructures underlying commercial Internet (eBay, Amazon, Google)
- Main issues: Optimize costs; Keep up with the load (flash crowds)

P2P Systems
- Exploit resources at network edges (storage, CPU, human presence)
- Main issues: Intermittent connectivity (churn); Network locality; Anonymity

Systems already in use, but characteristics hard to assess
- Performance: everyone want to maximize it, but definition differs
- Correction: absence of crash, race conditions, deadlocks and other defects
Assessing Distributed Applications

Performance Study $\sim$ Experimentation

- **Maths**: these artificial artifacts contain what we’ve put in it. But complex, dynamic, heterogeneous, scale $\sim$ beyond our capacities

- **Experimental Facilities**: Real applications on Real platform *(in vivo)*

- **Emulation**: Real applications on Synthetic platforms *(in vitro)*

- **Simulation**: Prototypes of applications on system’s Models *(in silico)*

<table>
<thead>
<tr>
<th>Experimental Facilities</th>
<th>Emulation</th>
<th>Simulation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Experimental Bias</td>
<td>😒�</td>
<td>😒</td>
</tr>
<tr>
<td>Experimental Control</td>
<td>😒�</td>
<td>😒</td>
</tr>
<tr>
<td>Ease of Use</td>
<td>😒</td>
<td>😒�</td>
</tr>
</tbody>
</table>

Correction Study $\sim$ Formal Methods *(model-checking, proof, . . . )*
Simulating Distributed Systems

Idea to test + System Model + Experimental setup → Scientific results

Main challenges
- **Validity**: Get realistic results (controlled experimental bias)
- **Scalability**: Simulate *fast enough* problems *big enough*
- **Usability**: Associated Tools; Ease of use; Applicability to context of interest

The SimGrid Framework as a Scientific Instrument
- Validated, Scalable, Usable; Modular; Portable
- Grounded +100 papers; 100 members on simgrid-user@; Open Source
- Simulates real programs, not models (C, Java, Lua, Ruby)
SimGrid Validity

10 years of efforts

- ≈ 10 network models: NS3 bindings, fast precise model, fast simplistic model
- Fluid models with contention, TCP congestion avoidance, piggybacking, ...

Results in HPC: accuracy of MPI simulations

![Graph showing communication time vs message size for different models and MPI implementations.]

Result in WAN: seeking for worst case scenario

- SimGrid and packet-level simulators only diverge in strange cases

![Simulation network topology diagram.]

M. Quinson – L. Nussbaum AlGorille team
Other SimGrid Qualities

Scalability
Millions of small processes

Chord simulation

Vizualization tools

Dozen of huge processes

Binominal scatter with 16 processes

Global View

Project A > 70%
Project B > Project A
Project A
~52.30 %
Project B
~47.70 %

Project A > 70%

SMPI (Simulation time) | SMPI (Simulated execution time) | OpenMPI

Memory limit

Time (in s)

Message size (in MiB)

0 4 8 16 32 64 128 224

0 10 20 30 40 50 60 70 80 90 100

0 500000 1e+06 1.5e+06 2e+06

Running time in seconds

Number of nodes

Oversim (simple underlay)
Constant network, sequential
Constant network, parallel
Precise network, sequential
Precise network, parallel

SMPI (Simulation time) | SMPI (Simulated execution time) | OpenMPI

Memory limit

Time (in s)

Message size (in MiB)

0 4 8 16 32 64 128 224

0 10 20 30 40 50 60 70 80 90 100

0 500000 1e+06 1.5e+06 2e+06

Running time in seconds

Number of nodes

Oversim (simple underlay)
Constant network, sequential
Constant network, parallel
Precise network, sequential
Precise network, parallel

Chord simulation

Vizualization tools

Project A > 70%
Project B > Project A
Project A
~52.30 %
Project B
~47.70 %

Global View

start end
time frame
whole simulated time
SONGS (Simulation Of Next Generation Systems)

Making SimGrid usable in 2 more domains

▶ Task 1: [Data]Grid (informal collab with CERN’s LHC)
  ▶ Challenge: Distributed Data mgnt for LHC; Hierarchical Storage System
▶ Task 2: Peer-to-Peer and Volunteer Computing
  ▶ Challenges: Replica Placement in VOD; Affinities in VC
▶ Task 3: IaaS Clouds (informal collab with IBM France and Haifa)
  ▶ Challenges: Study from client or provider POV; other metrics (energy)
▶ Task 4: High Performance Computing
  ▶ Challenges: exascale; memory & energy models

<table>
<thead>
<tr>
<th>Network</th>
<th>CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>quantitative</td>
<td>qualitative</td>
</tr>
<tr>
<td>quantitative</td>
<td>qualitative</td>
</tr>
<tr>
<td>“Old grids”</td>
<td>dozen</td>
</tr>
<tr>
<td>New grids</td>
<td>large scale</td>
</tr>
<tr>
<td>P2P</td>
<td>large scale</td>
</tr>
<tr>
<td>Clouds</td>
<td>large scale</td>
</tr>
<tr>
<td>HPC</td>
<td>large scale</td>
</tr>
</tbody>
</table>

(plus, work on validity, scalability, visualization, and support for Open Science)
**Grid’5000**

**Experimental Facility**
- Research on parallel, large-scale or distributed computing and networking
- 1700 machines (7400 CPU cores) in 26 clusters and 11 sites

**Technologies to support diverse experiments**
- CPUs from one to twelve cores
- High Performance networks: Infiniband & Myrinet
- Dedicated 10 Gb inter-site network (RENATER)

**Key feature: reconfigurable by users** (*Hardware-as-a-Service Cloud*)
- Installation of other operating systems on nodes: experiments on any level of the software stack – with Kadeploy
- Network isolation: allows the deployment of intrusive or security-sensitive protocols and applications – with KaVLAN
Grid’5000 – example results

Cloud: Sky computing on FutureGrid and Grid’5000
- Nimbus cloud deployed on 450+ nodes
- Grid’5000 and FutureGrid connected using ViNe

HPC / Cryptography: breaking RSA-768
- Feasibility study: prove that it can be done
- Different hardware \(\sim\) understand the performance characteristics of the algorithms

Grid: evaluation of the gLite grid middleware
- Fully automated deployment and configuration on 1000 nodes (9 sites, 17 clusters) in 3 hours
distem – distributed systems emulator

When Grid’5000 is too perfect... 

- Alter the platform so that it matches the experimental conditions you need
  - Introduce heterogeneity in an homogeneous cluster
  - Emulate a complex network topology
- You can still run real applications (in lightweight containers)

Key features:

- Uses modern Linux technology to steal resources from applications
- Easy to install and to use
- Scalable: 10 000-vnodes in a single experiment
Future: ScaLab (industrializing experimentation)

Experimental methodology: experiment design & planning; description of scenarios, of experimental conditions; definition of metrics; laboratory journal; analysis and visualization of results

Orchestration of experiments: organize the execution of complex and large-scale experiments; run experiments unattended and efficiently; handles failures; compose experiments

Basic services: common tools required by most experiments

- Interact w/ testbed
  - find/reserve/configure resources
- Check resources before using them
- Control a large number of nodes
- Manage data
- Change experimental conditions
- Instrument and monitor; extract data

Experimental testbed (Grid’5000): provides reconfigurable hardware and network, isolation, some instrumentation and monitoring
Conclusion

Computer Science is just like other Sciences

- Experimental facilities are mandatory (even if somehow rigid)
- Emulators are the ultimate scientific instruments (even if very complex)
- Computational Science is extremely powerful (even if tedious to get right)
- All available research methodologies must be combined and leveraged
- Grid’5000 and SimGrid are world leading tools

<table>
<thead>
<tr>
<th>Whiteboard</th>
<th>Simulation</th>
<th>Experimental Facilities</th>
<th>Emulation</th>
<th>Production Platforms</th>
</tr>
</thead>
<tbody>
<tr>
<td>Idea</td>
<td>😊😊</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Algorithm</td>
<td>😊</td>
<td>😊😊</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Prototype</td>
<td>😊</td>
<td>😊😊</td>
<td>😊😊</td>
<td></td>
</tr>
<tr>
<td>Application</td>
<td>😊😊</td>
<td></td>
<td>😊</td>
<td></td>
</tr>
<tr>
<td>Product</td>
<td></td>
<td></td>
<td></td>
<td>😊</td>
</tr>
</tbody>
</table>

One could determine the age of a science by the technic of its measurement instruments
– Gaston Bachelard, La formation de l’esprit scientifique.
Question slides
Studying Computer Systems

Computers are eminently artificial artifacts

- Humans built them completely, they contain only what we’ve put in it
  ⇒ Theoretical (maths) methodology to study it

Computer systems complexity getting tremendous

- Heterogeneity of components (hosts, links)
  - Quantitative: CPU clock, link bandwidth and latency
  - Qualitative: ethernet vs myrinet vs quadrics; amd64 vs ARM vs GPU

- Dynamicity
  - Quantitative: resource sharing ↗ availability variation
  - Qualitative: resource come and go (churn, failures)

- Complexity
  - Hierarchical systems: grids of clusters of multi-processors being multi-cores
  - Deep software stacks: Middleware, Web Services, mashups
  - Multi-hop nets, high latencies; Interference comput./comm. (disk/memory)

Computer Systems as Natural Objects

- The complexity is so high that we cannot understand them fully anymore
- Frankenstein effect, but allows to use computers to understand computers
Assessing Distributed Applications Correction

- Absence of crash / data corruption (like always)
- Absence of race condition / deadlocks / livelocks (classic in multi-entities)
- Feal with lack of central time and central memory (specific to distributed)

Correction Assessment $\sim$ Formal Methods

- **Facilities:** Experience plans limited, by abilities or by time
- **Simulation:** How to decide if coverage is sufficient?
- **Proof assistants:** semi-automated proof demonstration (tedious for users)
- **Model checking:** Exhaustive state space exploration, search counter examples

<table>
<thead>
<tr>
<th></th>
<th>Experimental Facilities</th>
<th>Emulation</th>
<th>Simulation</th>
<th>Proofs</th>
<th>Model Checking</th>
</tr>
</thead>
<tbody>
<tr>
<td>Performance Assessment</td>
<td>☺☺</td>
<td>☺☺</td>
<td>☻☺</td>
<td>☻☺</td>
<td>☻☺</td>
</tr>
<tr>
<td>Experimental Bias</td>
<td>☻☺</td>
<td>☻</td>
<td>☻</td>
<td>(n/a)</td>
<td>(n/a)</td>
</tr>
<tr>
<td>Experimental Control</td>
<td>☻☺</td>
<td>☻</td>
<td>☻☺</td>
<td>(n/a)</td>
<td>(n/a)</td>
</tr>
<tr>
<td>Ease of Use</td>
<td>☻</td>
<td>☻☺</td>
<td>☻☺</td>
<td>☻☺</td>
<td>☻</td>
</tr>
<tr>
<td>Correction Assessment</td>
<td>☻☺</td>
<td>☻</td>
<td>☻</td>
<td>☻☺</td>
<td>☻</td>
</tr>
<tr>
<td>Result if failed</td>
<td>(n/a)</td>
<td>(n/a)</td>
<td>(n/a)</td>
<td>☻</td>
<td>☻</td>
</tr>
</tbody>
</table>
System Administration Challenges

Goals and means

▶ Automating to factorize
  From 12 to 6 people
▶ Unique domain
  Intervention range unlimited
▶ Receipts in a central git
  ▶ Puppet for servers
  ▶ Chef for images
▶ Capistrano to push configs

Results

▶ Most know how encoded in receipts
  (young engineers-friendly)
▶ Hard to handle HPC hosts this way