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

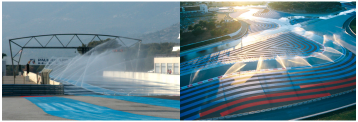
- Performance: everyone want to maximize it, but definition differs
- Correction: absence of crash, race conditions, deadlocks and other defects

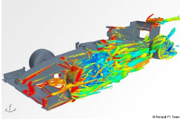
Assessing Distributed Applications

Performance Study ⇝ Experimentation

- Maths: these artificial artifacts contain what we've put in it
  But complex, dynamic, heterogeneous, scale ⇝ 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)*

	Experimental Facilities	Emulation	Simulation
Experimental Bias	☺☺	☺	☹
Experimental Control	☹☹	☺	☺☺
Ease of Use	☹	☹☹	☺☺

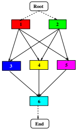




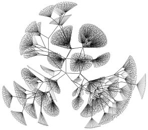
Correction Study ⇝ Formal Methods (model-checking, proof, ...)

Simulating Distributed Systems

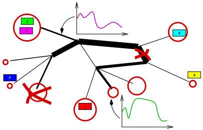
Idea to test



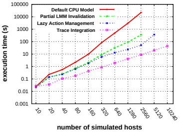
\+ System Model



\+ 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

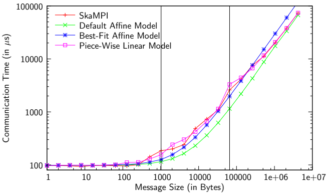
- Validated, Scalable, Usable; Modular; Portable
- Grounded +100 papers; 100 members on simgrid-user@; Open Source
- Simulates real programs, not models (C, Java, Lua, Ruby)

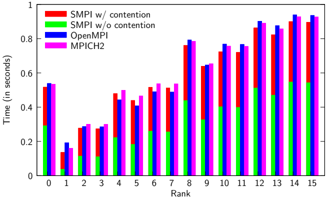
SimGrid Validity

10 years of efforts

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

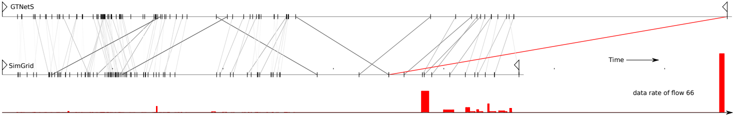
Results in HPC: accuracy of MPI simulations





Result in WAN: seeking for worst case scenario

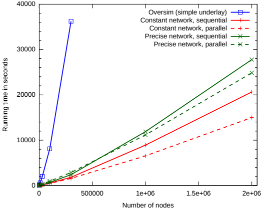
- SimGrid and packet-level simulators only diverge in strange cases



Other SimGrid Qualities

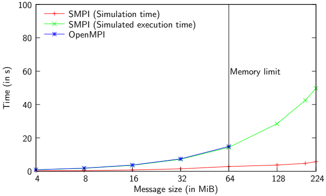
Scalability

Millions of small processes



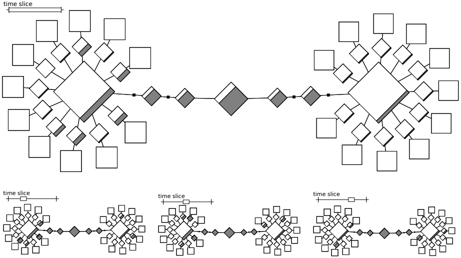
Chord simulation

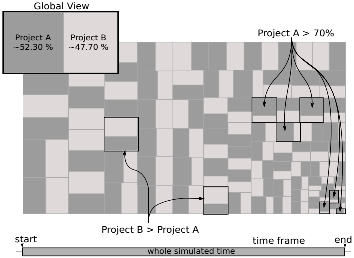
Dozen of huge processes



Binomial scatter with 16 processes

Vizualization tools

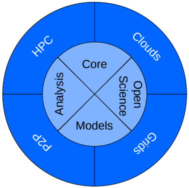




SONGS (Simulation Of Next Generation Systems)

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



	Network		CPU	
	quantitative	qualitative	quantitative	qualitative
"Old grids"	dozen	NREN	hundreds	spare
New grids	large scale	NREN	hundreds	clusters
P2P	large scale	Internet	large scale	spare
Clouds	large scale	Internet	hundreds	clusters
HPC	large scale	LAN	thousands	clusters

(plus, work on validity, scalability, vizualization, and support for Open Science)

Grid'5000

Experimental Facility

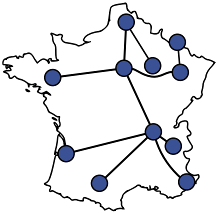
- Research on parallel, large-scale or distributed computing and networking
- 1700 machines (7400 CPU cores) in 26 clusters and 11 sites

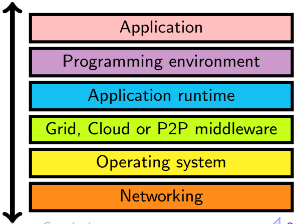
Technologies to support diverse experiments

- 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)

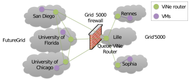
- Installation of other operating systems on nodes: experiments on any level of the software stack – with Kadeploy
- Network isolation: allows the deployment of intrusive or security-sensitive protocols and applications – with KaVLAN





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 $\rightsquigarrow$ 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

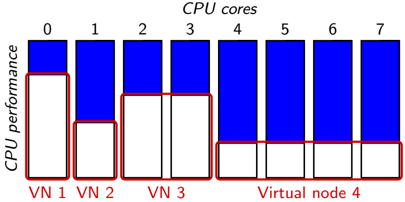
distem – distributed systems emulator

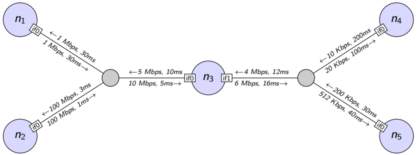
When Grid'5000 is too perfect...

- 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)

Layer 3

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

Layer 2

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

Layer 1

Basic services: common tools required by most experiments

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

Layer 0

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

	Whiteboard	Simulation	Experimental Facilities	Emulation	Production Platforms
Idea	☺☺				
Algorithm	☺	☺☺			
Prototype		☺	☺☺		
Application			☺☺	☺	
Product					☺

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

$\Rightarrow$ 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 $\rightsquigarrow$ 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

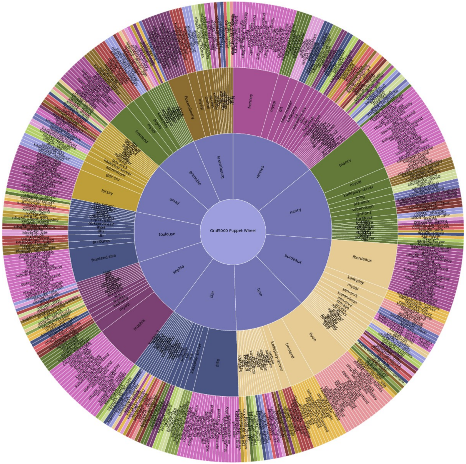
- Absence of crash / data corruption (like always)
- Absence of race condition / deadlocks / livelocks (classic in multi-entities)
- Feal with lack of central time and central memory (specific to distributed)

Correction Assessment $\rightsquigarrow$ 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

	Experimental Facilities	Emulation	Simulation	Proofs	Model Checking
Performance Assessment	☺☺	☺☺	☺☺	☹☹	☹☹
Experimental Bias	☺☺	☺	☹	(n/a)	(n/a)
Experimental Control	☹☹	☺	☺☺	(n/a)	(n/a)
Ease of Use	☹	☹☹	☺☺	☹☹	☺
Correction Assessment	☹☹	☹	☹	☺☺	☺
Result if failed	(n/a)	(n/a)	(n/a)	☹	☺☺

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