Distributed Applications' Study:
Experimentation, Simulation and Formal methods

Martin Quinson (team AlGorille)

Université Henri Poincaré – Nancy I

April 15, 2011

Context: Distributed Systems Taxonomy

Cloud Computing

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

High Performance Computing and Exascale

- Have the world's biggest computer, to lead CS and IT world's research
- Main issues: do the biggest possible numerical simulations [justify investment]

Grid Computing

- Infrastructure for *computational science:* lot of sequential simulation jobs
- Main issues: compatibility, virtual organizations (trust and accountability mgmt)

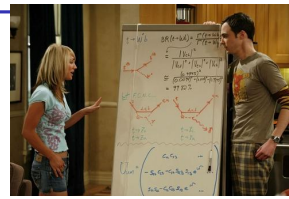
Peer-to-peer Systems (P2P)

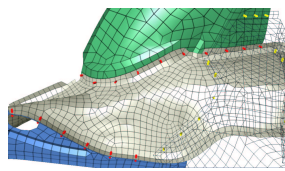
- Exploit resources at network edges (storage, CPU, human presence)
- Main issues: Intermittent connectivity (churn); Network locality; Anonymity

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





Classical Scientific Pillars Apply

- Theoretical Approach: Mathematical study of algorithms
- Experimental Science: Study applications on scientific instrument
- Computational Science: Simulation of a system model

Performance Study $\rightsquigarrow$ Experimentation

- Theory still mandatory, but everything's NP-hard
- 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	☹	☹☹	☺☺

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

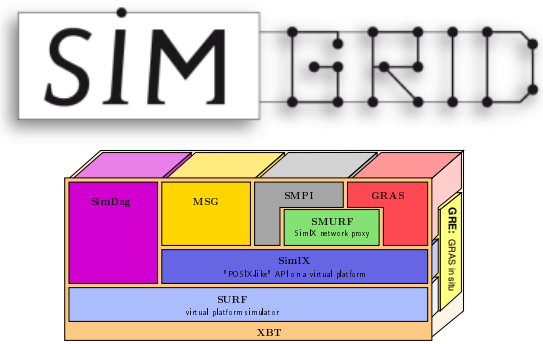
- Facilities: Experience plans limited, by abilities or by time
- Simulation: How to decide if coverage is sufficient?
- Proof assistants: semi-automated proof demonstration (tedious for users)
- Model checking: Exhaustive state space exploration, search counter examples

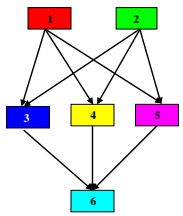
	Experimental Facilities	Emulation	Simulation	Proofs	Model Checking
Performance Assessment	☺☺	☺☺	☺☺	☹☹	☹☹
Experimental Bias	☺☺	☺	☹	(n/a)	(n/a)
Experimental Control	☹☹	☺	☺☺	(n/a)	(n/a)
Ease of Use	☹	☹☹	☺☺	☹☹	☺
Correction Assessment	☹☹	☹	☹	☺☺	☺
Result if failed	(n/a)	(n/a)	(n/a)	☹	☺☺

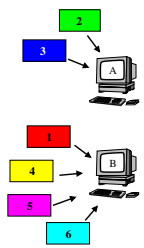
Contributions to the simulation of dist apps (in silico)

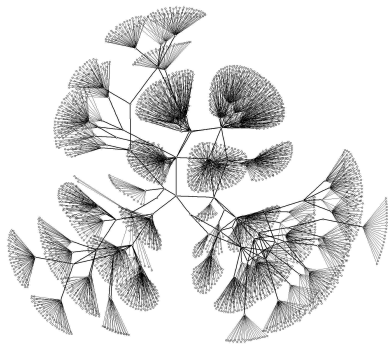
SimGrid: simulator of applications' prototype on models of platforms

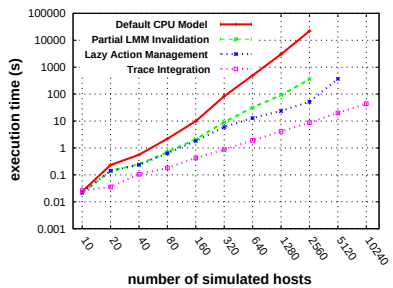
- Scalable (big enough, fast enough); Modular (TCP and others); Portable
- +70 papers; 100 members on simgrid-user@











Personal Contributions (collaboration with A. Legrand and F. Suter)

- Leader of the federating ANR project (7 labs 19 partners 800k€: P2P scalability)
- Main Software Architect Parallel Simulation, Applicative validation, MPI, energy
- Collaborations: CERN, IBM, U. Anvers; Dissemination: User Days; Tutorials

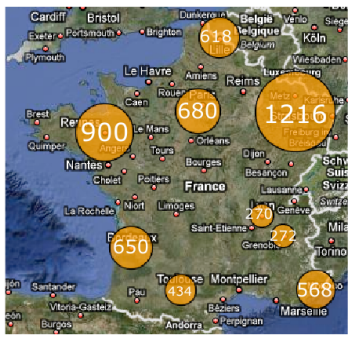
Project: Simulation Of Next Generation Systems (SONGS)

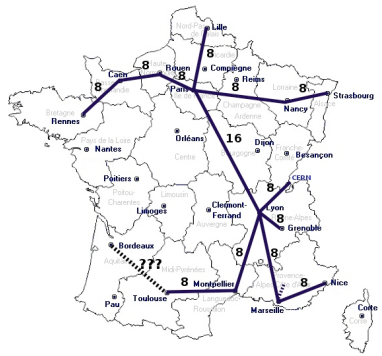
- Federate Grids, P2P, Clouds & HPC simulation within the same framework
- Leader of ANR project under evaluation (7 labs, 21 scientists, 1.8M€)

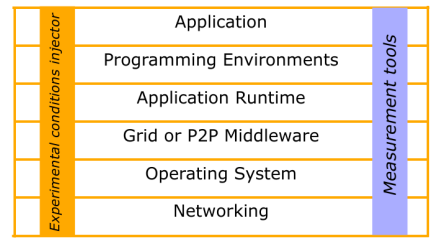
Contributions to Experimental Facilities (in vivo)

Grid'5000 Project: world leading scientific instrument for dist. apps

- Instrument for research in computer science (*deployment* of customized OSes)
  1500 nodes (2800 cpus, 7200 cores), 9 sites; dedicated 10Gb network







Personal Contributions

- National steering committee; Local project co-leader (CPER, Aladdin, Hemera)
- Scientific animation, event co-organization: Nancy is a leading site
- Collaboration: Production grids (IdG), CEA, Arcelor-Mittal

Project: Experimentation Process Industrialization (with L. Nussbaum)

- Open science: ensure that experiments can be shared, reviewed, improved
- Convergence of simulation and direct execution
- Methodological framework and practical tools (+administrative duties)

Other Contributions

Model-Checking (collaboration with S. Merz and C. Rosa)

- **Goal:** democratize Formal Methods to non specialists through SimGrid
- Achievements:
  - Model-checking mode in SimGrid; Generic modeling of communications API
  - DPOR implementation fighting combinatorial explosion regardless of used API
- Projects:
  - Integrate Liveness Properties; Automatically bridge code $\leftrightarrow$ model variables
  - Long Term: *semantic* debugger of distributed applications within SimGrid
  - Very Long term: Performance checking (time discrete at best in MC)

Simulated MPI (collaboration with S. Genaud, H. Casanova, F. Suter, P.N. Clauss)

- **Goal:** study real applications based on MPI within SimGrid
- Achievements: Partial implem of MPI; Assessment of LAN models
- Projects: Modeling collectives' *Semantic* ($\rightsquigarrow$ MPI-3); Trace based simulation

Study of Real Applications: SimTerpose (collaboration with L. Nussbaum)

- **Goal:** intercept every actions of the application, and study them online
- Achievements: Prototype of interceptor; Projects: TBC, and used

+ PlusCal (MC$\rightarrow$Sim with Lamport); GRAS, Alnem; Energy, DistSim; JLM, CLE

Conclusion: Une délégation pour une HDR

IMHO, writting an HDR is about making 3 points

- Ability to do research yourself
  - 1 book chapter, 4 journals, 17 confs&wshops, 7 tutorials & invited talks
  - Pragmatic approach leading to several tools, with large user bases
- Ability to collaborate with peers; Integration in scientific community
  - Academic collabs: Nancy, Grenoble, Lyon, Bdx, Nice, Nantes, Strasbrg, Hawai'i
  - Industrial collabs: CERN, adhoc intl, IBM, (arcelor-mittal)
  - Project leader of several large scale projects (1,5M€ since 2005)
  - Member of Steering Committees, Program Committees and PhD defense juries
- Ability to advise young researchers
  - 3 postdocs, 2 ongoing PhDs, 4 M2R Loria, 6 other M2R; 3 engineers (IJD)
  - (Coordinator of first year curriculum at ESIAL and of 7 teaching modules)

Why asking a délégation now?

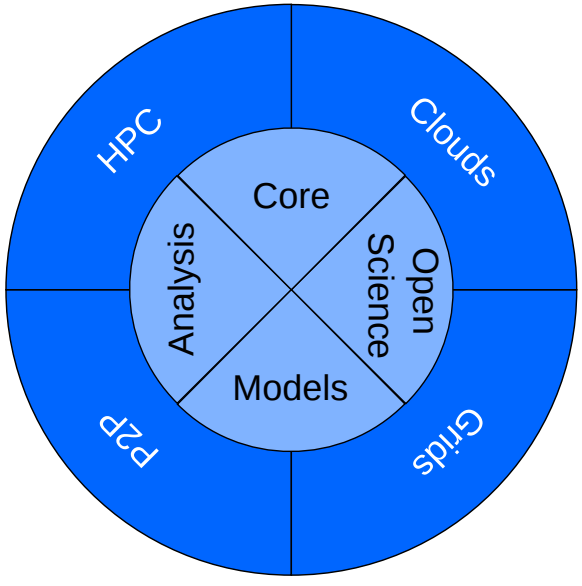
- I need to finish what's ongoing, and get published the ideas that emerged
  Publication file may not really reflect my production yet
- I need to write the manuscript
- I'm short on time with 200-250 hours of teaching duty per year

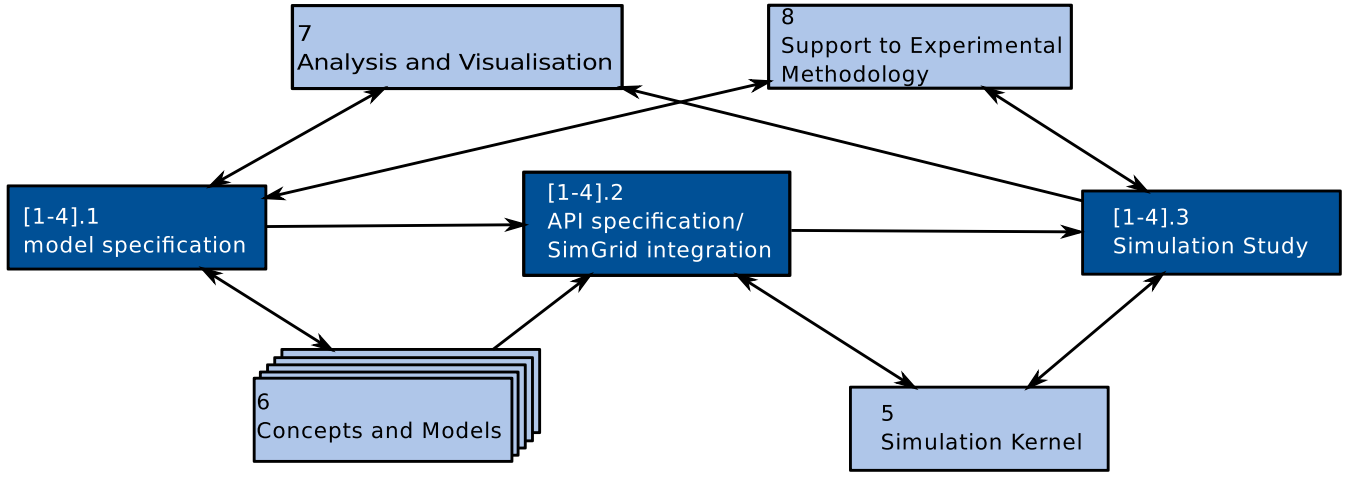
Agenda

- Context
  - Distributed Systems Taxonomy
  - Assessing Distributed Applications Performance
  - Assessing Distributed Applications Correction
- Contributions: Methodologies for the study of Distributed Applications
  - Simulating Distributed Applications
  - Experimental Facilities
  - Other Contributions
- Annexes
  - SONGS: Simulation Of Next Generation Systems
  - Emulation
    - Emulation through Degradation or through Simulation
    - How to intercept application actions?

Simulating Next Generation Systems

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





Emulation as an Experimental Methodology

Execute your application in a perfectly controlled environment

- Real platforms are not controllable, so how to achieve this?
- Let's look at what engineers do in other fields

Emulation as an Experimental Methodology

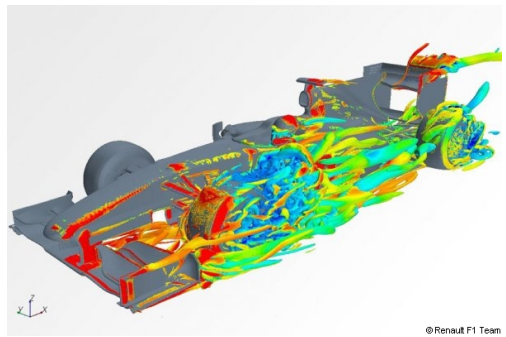
Execute your application in a perfectly controlled environment

- Real platforms are not controllable, so how to achieve this?
- Let's look at what engineers do in other fields

When you want to build a race car...

...adapted to wet tracks

...in a dry country ...



...you can simulate it.

Courtesy of Lucas Nussbaum

But then, you have

- To assess your models
- Technical burden
- No real car

Why don't you... just control the climate?

Emulation as an Experimental Methodology

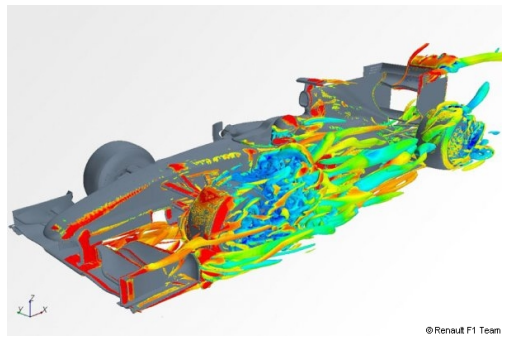
Execute your application in a perfectly controlled environment

- Real platforms are not controllable, so how to achieve this?
- Let's look at what engineers do in other fields

When you want to build a race car...

...adapted to wet tracks

...in a dry country ...



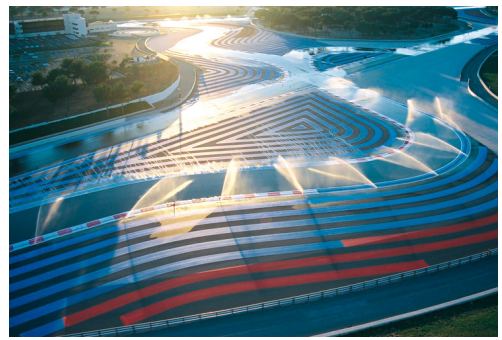
...you can simulate it.

But then, you have

- To assess your models
- Technical burden
- No real car

Why don't you...

just control the climate?



That's Emulation

Courtesy of Lucas Nussbaum

Emulating Distributed Systems

Such *Emulation through Degradation* is quite classical

- Degrade the performance of the host platform (CPU burners, Network capping)
- WrekAvoc (LORIA – Lucas Nussbaum *et Al.*) works this way

☺ Real application, controlled environment

☹ Complex technologies, heavy infrastructures, tedious tool assessment

☹ Reeeeally hard to emulate faster/larger platforms than host platform

Emulating Distributed Systems

Such *Emulation through Degradation* is quite classical

- Degrade the performance of the host platform (CPU burners, Network capping)
- WrekAvoc (LORIA – Lucas Nussbaum *et Al.*) works this way

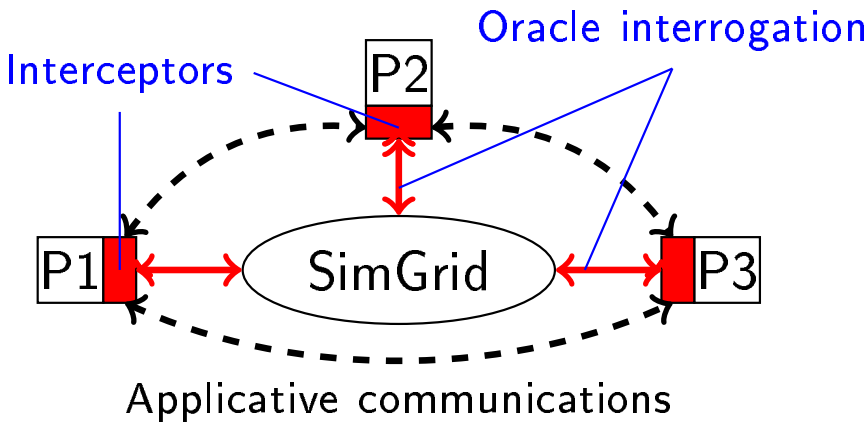
☺ Real application, controlled environment

☹ Complex technologies, heavy infrastructures, tedious tool assessment

☹ Reeeeally hard to emulate faster/larger platforms than host platform

Another approach: Emulation through Simulation

- Intercept what the application does, Compute answer by simulation, Apply it



This is SimTerpose

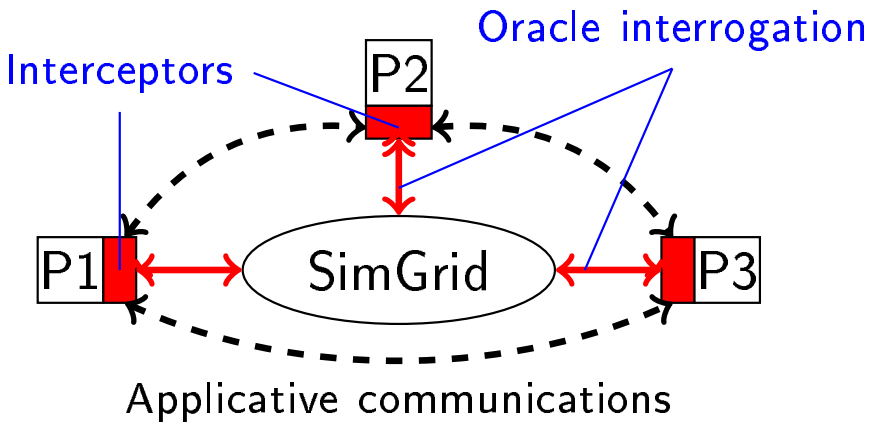
Emulating Distributed Systems

Such *Emulation through Degradation* is quite classical

- Degrade the performance of the host platform (CPU burners, Network capping)
- WrekAvoc (LORIA – Lucas Nussbaum *et Al.*) works this way
- ☺ Real application, controlled environment
- ☹ Complex technologies, heavy infrastructures, tedious tool assessment
- ☹ Reeeeally hard to emulate faster/larger platforms than host platform

Another approach: Emulation through Simulation

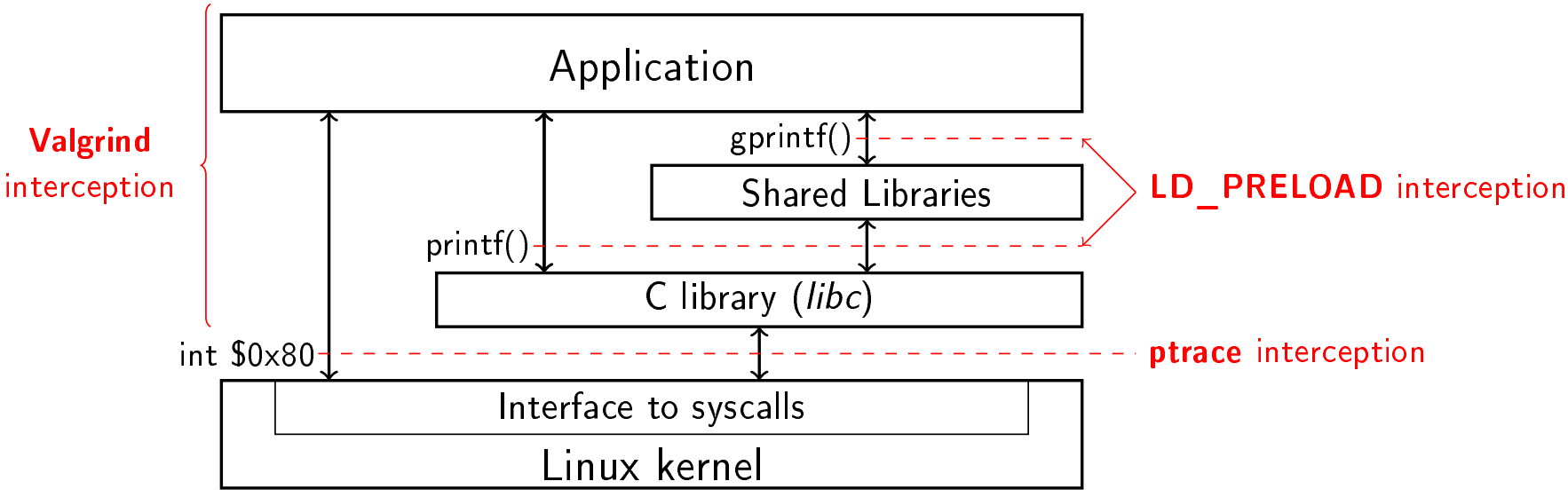
- Intercept what the application does, Compute answer by simulation, Apply it
- This is what you do to assess the breaking system of your car



This is SimTerpose

How to intercept application actions?

Several approaches exist



- Valgrind: Binary rewrite before execution
  Very slow!
- LD_PRELOAD: Dynamic loader trick ($\approx$ DLL injection)
  Library calls only (lot of them can be used to communicate)
- ptrace: SysCall trapping by kernel (approach used by gdb or strace)
  Quite tricky to setup correctly – but possible