Emulation Through Simulation

The SimTerpose Project

Martin Quinson
(AlGorille – LORIA)

March 7 2011

Context: Real large-scale distributed systems

Cloud Computing

- Large infrastructures underlying the commercial Internet (eBay, Amazon, Google)
- Main issue: keep up with the load, even when facing flash crowd effects

Grid Computing

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

High Performance Computing and Exascale

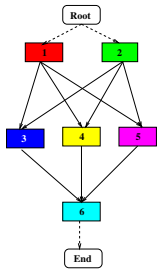
- Have the world's biggest computer, to lead CS and IT world's research
- Main issues: do the biggest possible parallel simulations [justify the investment]

These systems are in use today, but badly understood

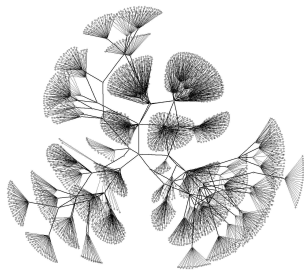
- They deserve a thorough scientific analysis
- Classical experimental methodologies apply: theory, experiment, simulation
- Most of the published work rely on simulation

Simulating Distributed Systems

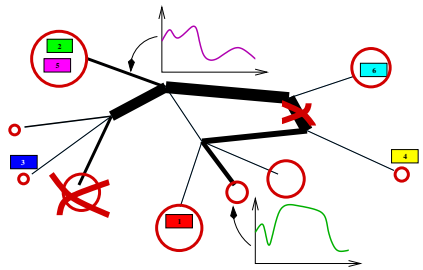
Idea to test



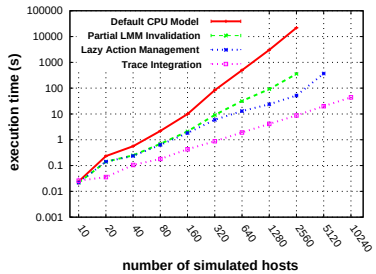
System Model



Experimental setup



Scientific results



Advantages

- Better experimental control than on production systems ($\rightsquigarrow$ reproducible)
- Easier and quicker experiments compared to experimental platforms
- Plus: What If analysis
- SimGrid project [LORIA *et Al.*]: one of the world-leading application simulator

Drawbacks

- Models must be assessed to limit experimental bias
- Tools must be efficient enough to study big enough fast enough
- Require to use the interfaces of the simulator
  $\Rightarrow$ Impossible to study an existing application this way

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

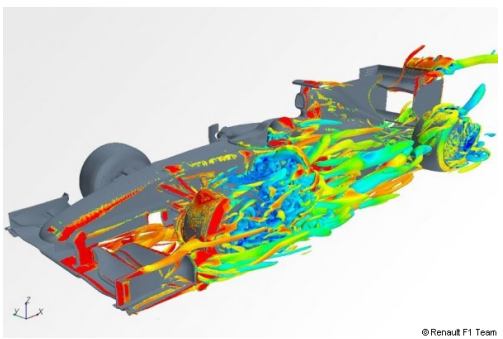
Execute your application in a perfectly controlled environment

- 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

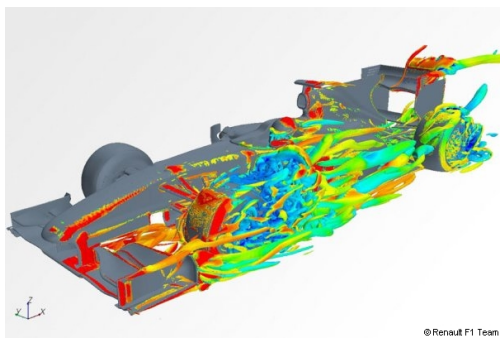
Execute your application in a perfectly controlled environment

- 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 ...



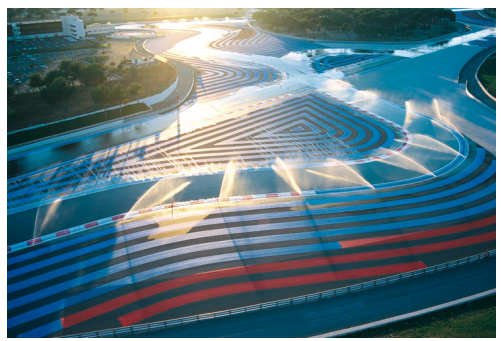
...you can simulate it.

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

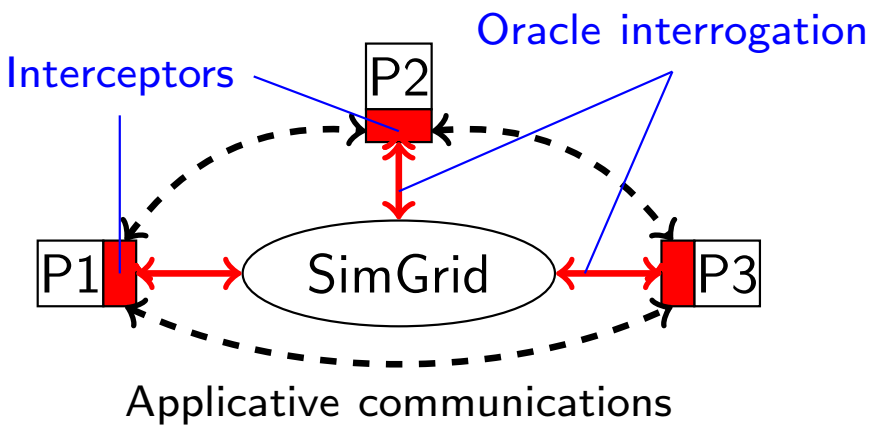
Emulating Distributed Systems

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

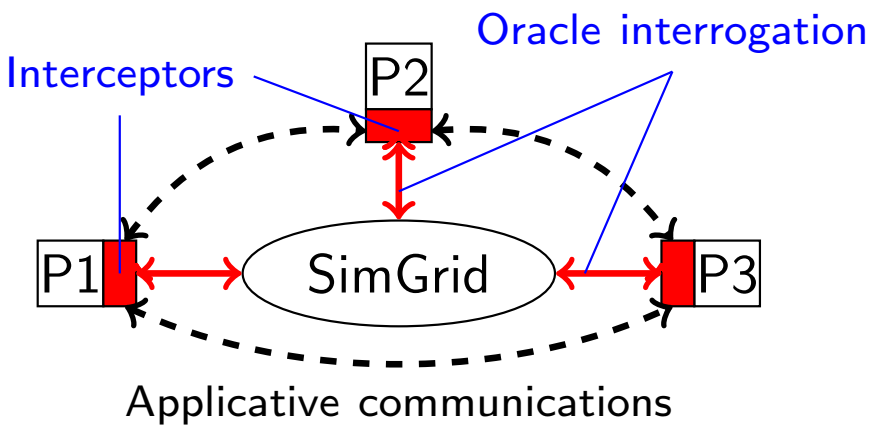
Emulating Distributed Systems

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

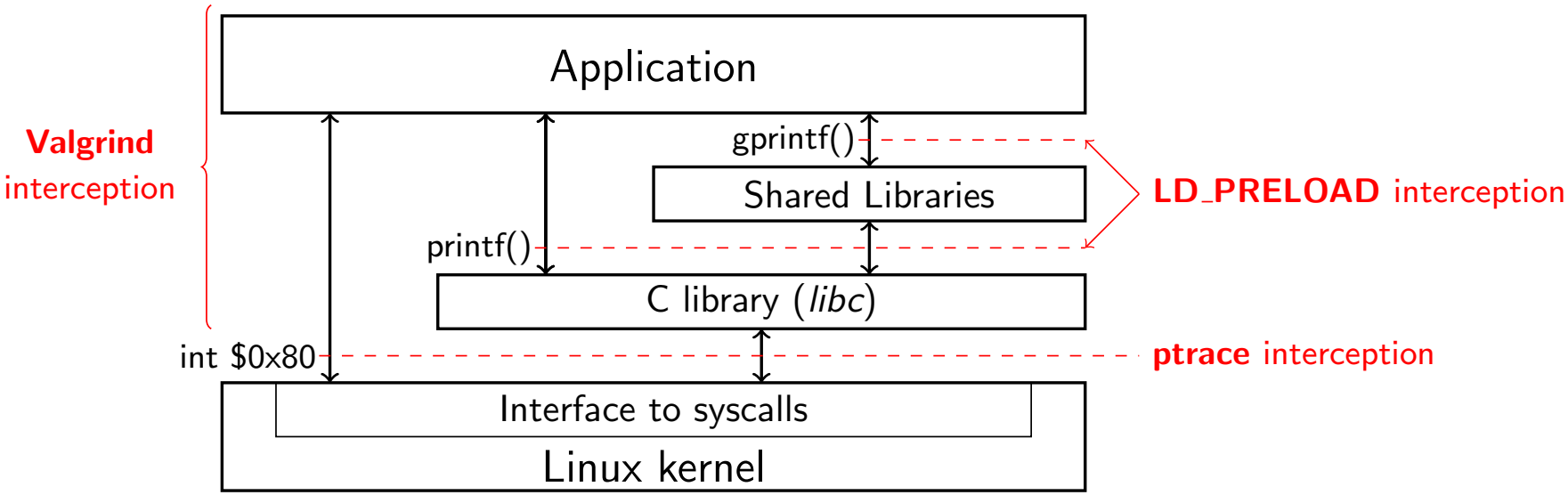
- Intercept what the application does, Compute answer by simulation, Apply it
- This is what you do to assess the breaking system of your car



This is SimTerpose

How to intercept application actions?

Several approaches exist



- Valgrind: Binary rewrite before execution
  Very slow!
- LD_PRELOAD: Dynamic loader trick (≈ 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

Conclusion

State of the Art and Current Status

- MicroGrid: Project from UCSD using LD_PRELOAD (very complex; now dead)
- Our interception mechanism (ptrace based) kinda works for comms and comput

Project

- Plan: Complete the prototype; Validate & Enjoy
  - Task 1: Time and DNS mandate novel interception schema
  - Task 2: Feedback from simulator is to be applied (delay app, or trick time)
  - Task 3: Validate tool, Compare to MicroGrid and Publish results
  - Task 4: Use it to better understand and model MPI runtimes
- Have: A candidate; Some amount outstanding after other funding
- Need: Funding for one year of post-doc to do that work

Expected Benefits

- **Scientific:** Killer methodology using simulation for better studies of real apps
- **The candidate:** Expertise in HPC experimentations (+publication[s])
  All big corporate need data centers, this is a wanted ability
- **The region:** Reinforce local expertise in experimentation methodologies
  Fully matches the agenda of EDGE project from CPER MISN