SimGrid 101

Getting Started to the SimGrid Project

Da SimGrid Team

June 13, 2012

About this Presentation

Goals and Contents

- Scientific Context of SimGrid: experimenting with distributed applications
- Historical Context of SimGrid: where we come from
- Key Features of the Framework: bigger, faster, smarter, better

The SimGrid 101 serie

- This is part of a serie of presentations introducing various aspects of SimGrid
- SimGrid 101. Introduction to the SimGrid Scientific Project
- SimGrid User 101. Practical introduction to SimGrid and MSG
- SimGrid User::Platform 101. Defining platforms and experiments in SimGrid
- SimGrid User::SimDag 101. Practical introduction to the use of SimDag
- SimGrid User::Visualization 101. Visualization of SimGrid simulation results
- SimGrid User::SMPI 101. Simulation MPI applications in practice
- SimGrid User::Model-checking 101. Formal Verification of SimGrid programs
- SimGrid Internal::Models. The Platform Models underlying SimGrid
- SimGrid Internal::Kernel. Under the Hood of SimGrid
- Retrieve them from `http://simgrid.gforge.inria.fr/101`

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

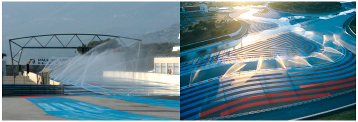
- Performance: makespan, economics, energy, ... ← main context of SimGrid
- Correction: absence of crash, race conditions, deadlocks and other defects

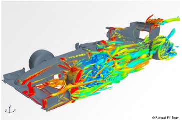
Assessing Distributed Applications

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, . . .)

Computer Systems to Study 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

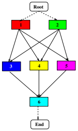
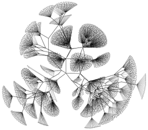
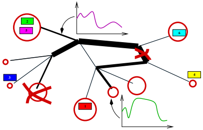
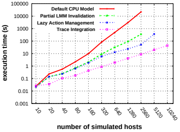
Unpreceeded Computer Systems complexity

- Heterogeneity of components (hosts, links); Dynamicity; Complexity
- (both Quantitative and Qualitative)

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
- Simply applies In Silico approach to our science

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*
- Associated tools: campaign mgmt, result analysis, settings generation, ...
- Applicability: If it doesn't simulate what is important to you, it's void

Simulation's Advantages

- Less simplistic than proposed theoretical models (which are useful too)
- Better XP control ($\rightsquigarrow$ reproducible) than production systems (+ not disruptive)
- Not as tedious, time/labor consuming than experimental platforms
- Plus: Lower technical burden; Quick and easy experiments; What if analysis

Large-Scale Distributed Systems Science?

Requirements for a Scientific Approach

- Reproducible results: read a paper, reproduce the results and improve
- Standard tools and methodologies
  - Grad students can learn their use and become operational quickly
  - Experimental scenario can be compared accurately

Current practice in the field: quite different

- Experimental settings rarely detailed enough in literature
- Very little common methodologies and tools, large load of (ad-hoc) tools
  - Few are really usable: Diffusion, Software Quality Assurance, Long-term availability
  - Most rely on straightforward models with no validity assessment

Large-Scale Distributed Systems Science?

Requirements for a Scientific Approach

- Reproducible results: read a paper, reproduce the results and improve
- Standard tools and methodologies
  - Grad students can learn their use and become operational quickly
  - Experimental scenario can be compared accurately

Current practice in the field: quite different

- Experimental settings rarely detailed enough in literature
- Very little common methodologies and tools, large load of (ad-hoc) tools
  - Few are really usable: Diffusion, Software Quality Assurance, Long-term availability
  - Most rely on straightforward models with no validity assessment

	Dom	CPU	Disk	Network	Application	Scale
ns2/3	Networking	None	None	Packet-level	Coarse d.e.s.	1,000
GTNetS	Networking	None	None	Packet-Level	Coarse d.e.s.	100,000
OptorSim	(Data)Grid	Analytic	Amount.	(buggy) Analytic	Coarse d.e.s.	1,000
GridSim CloudSim	Grid Cloud	Analytic	Analytic	Wormhole (buggy) Analytic	Coarse d.e.s.	1,000
PlanetSim	P2P	None	None	Constant time	Coarse d.e.s.	100,000
PeerSim	P2P	None	None	None	State machine	1,000,000
SimGrid	Grid, VC, P2P, HPC, cloud, ...	Analytic	Analytic	Analytic or Packet-level (NS3)	Coarse d.e.s. or emulation	1,000,000

SimGrid: Versatile Simulator of Distributed Apps

- Allow a scientific approach of Large-Scale Distributed Systems simulation
- Propose ready to use tools enforcing methodological best practices
- Methodological outcomes not limited to simulation (Grid'5000 and DistEm)

Scientific Instrument

- Allows studies of Grid, P2P, HPC, Volunteer Computing and others
- Validated, Scalable, Usable; Modular; Portable
- Grounded +100 papers; 100 members on simgrid-user@; Open Source
- Simulates real programs (using specific API), not models; C, Java, Lua, Ruby

Scientific Object (and lab)

- Allows comparison of network and middleware performance models
- Experimental (but on par with SotA) Model Checker; Soon an emulator

Scientific Project since 12 years

- Collaboration Loria / Inria Rhône-Alpes / CCIN2P3 / U. Hawaii
- ANR's USS SimGrid and SONGS; INRIA fundings and support

SimGrid History

1998-2001 Baby steps: Factorize some code between PhD students in scheduling

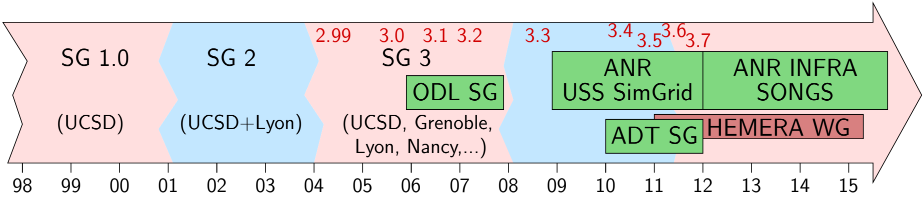
- Features: Platform heterogeneity and dynamicity (traces)
- Done by H. Casanova; Scope: limited to scheduling at UCSD

2001-2003 Infancy:

2003-2008 Teenage:

2008-2011 Maturation:

2012-: Taking over the world ;-)



SimGrid History

1998-2001 Baby steps: Factorize some code between PhD students in scheduling

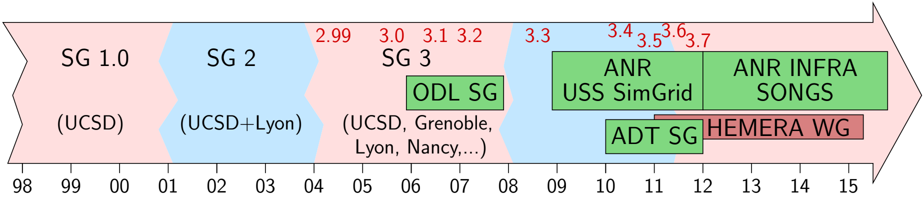
2001-2003 Infancy:

- Allow distributed scheduling: A. Legrand added CSPs on top of existing
- Model validity questionable: L. Marchal implemented analytical models
- Scope: UCSD + ENS-Lyon and few others

2003-2008 Teenage:

2008-2011 Maturation:

2012-: Taking over the world ;-)



SimGrid History

1998-2001 Baby steps: Factorize some code between PhD students in scheduling

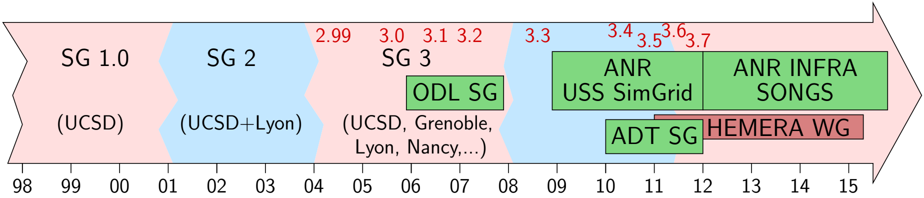
2001-2003 Infancy: CSP and improved models

2003-2008 Teenage:

- Performance improvement: rewrite kernel from scratch (SURF)
- Validity quest: Hunt (and fix) worst case scenarios
- Other interfaces: GRAS, SimDag, SMPI

2008-2011 Maturation:

2012-: Taking over the world ;-)



SimGrid History

1998-2001 Baby steps: Factorize some code between PhD students in scheduling

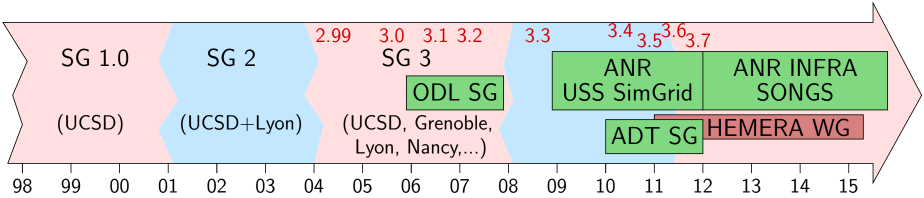
2001-2003 Infancy: CSP and improved models

2003-2008 Teenage: Performance, validity, multi-APIs

2008-2011 Maturation:

- Scope extension from Grid to P2P and VC: scalable to death, peer models
- Associated tools: visualization, campaign management
- ANR USS SimGrid + support from INRIA (ADT)

2012-: Taking over the world ;-)



SimGrid History

1998-2001 Baby steps: Factorize some code between PhD students in scheduling

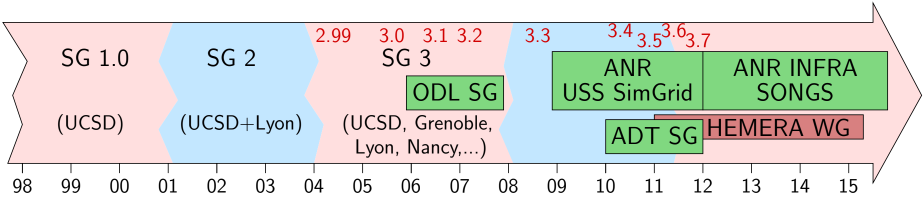
2001-2003 Infancy: CSP and improved models

2003-2008 Teenage: Performance, validity, multi-APIs

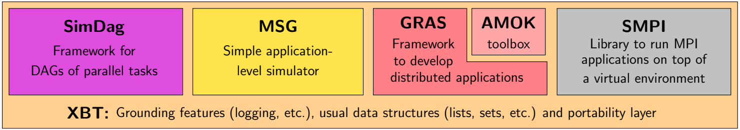
2008-2011 Maturation: Scope increase to P2P, visualization

2012-: Taking over the world ;-)

- Scope extension to HPC, Cloud
- Associated tools: visualization, campaign management
- ANR INFRA SONGS



User-visible SimGrid Components



SimGrid user APIs

- SimDag: heuristics as DAG of (parallel) tasks
- MSG: heuristics as Concurrent Sequential Processes (Java/Ruby/Lua bindings)
- GRAS: develop real applications that are debugged in simulator (don't use)
- SMPI: simulate real applications written using MPI

Which API should I choose?

- Your application is a DAG $\rightsquigarrow$ SimDag; you have a MPI code $\rightsquigarrow$ SMPI
- You study distributed applications $\rightsquigarrow$ MSG; you like dangerous living $\rightsquigarrow$ GRAS

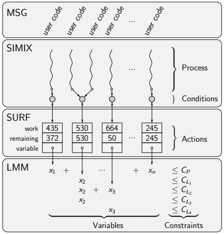
Argh! Do I really have to code in C?!

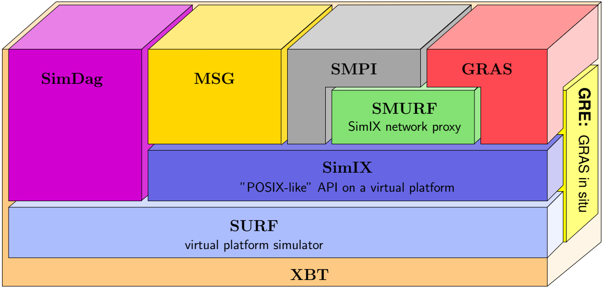
- Not necessary. Java and lua bindings of MSG: $\approx 5\times$ slower only (+ruby)
- But it helps for performance; non-MSG modules are C-only

Quick glance under the SimGrid Hood

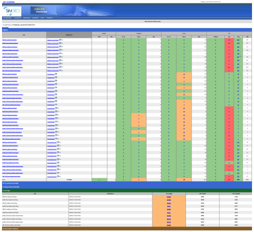
SimGrid Functional Organization

- MSG: User-friendly syntaxic sugar
- Simix: Processes, synchro (SimPOSIX)
- SURF: Resources usage interface
- Models: Action completion computation





Practical Trust onto SimGrid?



Nightly tests

- git version gets tested every night
- 230 integration tests; 10,000 unit tests; coverage ≈70%
- 2 SimGrid configs, 10 Linux version
- If something goes wrong, we'll notice

Release tests

- Windows and Mac considered as release goals too (but manually tested only)
  (actually works on all Debian arch.: Hurd, kfreebsd, mips, arm, ppc, s390 ;)
- No, not quite "just another simulator", things can go wrong

Performance regression testing: coming soon

This is free software anyway

- The code is currently LGPL, probably soon GPL
- Come, check it out and participate!

SimGrid Validity

SotA: Models in most simulators are either simplistic, wrong or not assessed

- PeerSim: discrete time, application as automaton;
- GridSim: naive packet level or buggy flow sharing
- OptorSim, GroudSim: documented as wrong on heterogeneous platforms

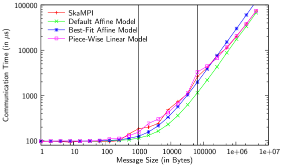
SimGrid provides 3 models

- NS3 bindings; Fast precise model (preferred); Fast simplistic model
- *Fluid models* with Contention, TCP congestion avoidance, Cross-traffic, ...
- See SURF 101 for details on the models and their validity

Quality assessment by comparing to packet level simulators

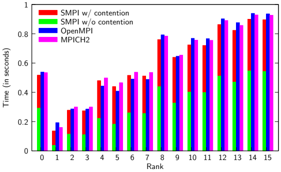
- Hunting (and fixing) worst case scenarios since 10 years
- Sharing mechanism from theoretical literature experimentally proved wrong
- SimGrid and packet-level simulators now mostly diverge in *extreme* cases

Accuracy of MPI simulations



Timings of each communication

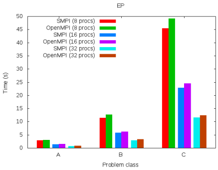
- $\lambda + size \times \tau$ not sufficient (TCP congestion)
- No affine fonction can match for all message sizes
- A 3-parts piecewise affine gives satisfying results



Taking resource sharing into account

- Cannot ignore contention (as other simulators do)
- Our “error” $\approx$ difference between runtimes
- This is only one collective

Still a work in progress for complete MPI applications

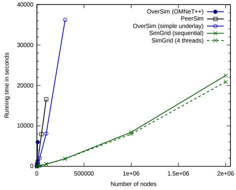


- Performance prediction not correct
- Trashing particularly challenging

SimGrid Scalability

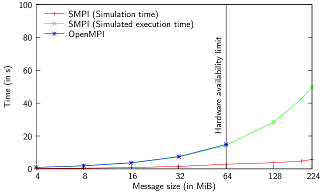
Two aspects: Big enough (large platforms) ⊕ Fast enough (large workload)

Millions of small processes



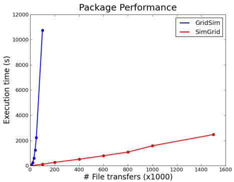
Chord simulation

Dozen of huge processes



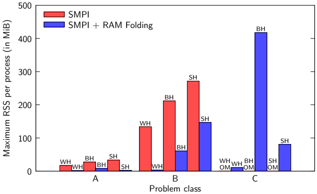
Binomial scatter with 16 processes

Large Workload



Simulation from CERN users

Hundreds of large processes

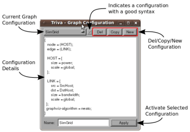


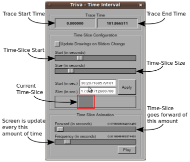
DT with up to 448 processes

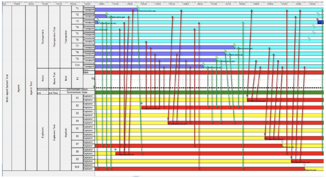
SimGrid can run in parallel, but only on one node (no distributed simulation yet)

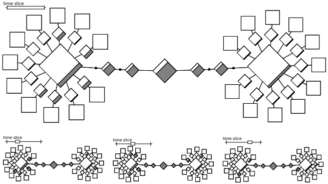
Visualizing SimGrid Simulations

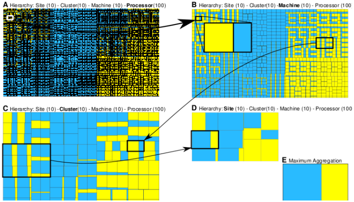
- Visualization scriptable: easy but powerful configuration; Scalable tools
- Right Information: both platform and applicative visualizations
- Right Representation: gantt charts, spatial representations, tree-graphs
- Easy navigation in space and time: selection, aggregation, animation
- Easy trace comparison: Trace diffing (still partial ATM)

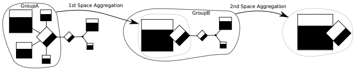


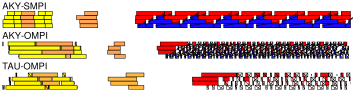










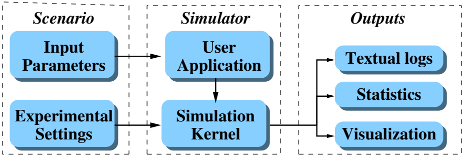


Other Associated Tools

Workflow to any Experiments through Simulation

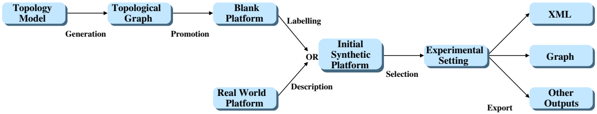
1. Prepare the experimental scenarios
2. Launch thousands of simulations
3. Post-processing and result analysis

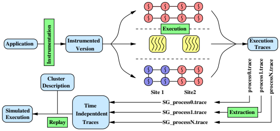
⇝ Each simulation is only a brick



Workload Generation

- Platforms: Simulacrum (generation), PDA (archive) and MintCAR (mapping)
- Applicative Workload: Tau-based trace collection + replay





Conclusion

SimGrid will prove helpful to your research

- User-community much larger than contributors group (which is large too)
- Used in several communities (scheduling, GridRPC, HPC infrastructure, P2P)
  More to come: Clouds and HPC are the topic of the SONGS project
- Model limits known thanks to validation studies
- Easy to use, extensible, fast to execute, scalable to death
- Around since over 10 years, and ready for at least 10 more years

We only scratched the surface. Check the other 101 presentations

- SimGrid User 101. Practical introduction to SimGrid and MSG
- SimGrid User::Platform 101. Defining platforms and experiments in SimGrid
- SimGrid User::SimDag 101. Practical introduction to the use of SimDag
- SimGrid User::Visualization 101. Visualization of SimGrid simulation results
- SimGrid User::SMPI 101. Simulation MPI applications in practice
- SimGrid User::Model-checking 101. Formal Verification of SimGrid programs
- SimGrid Internal::Models. The Platform Models underlying SimGrid
- SimGrid Internal::Kernel. Under the Hood of SimGrid