SimGrid 101

Getting Started to the SimGrid Project

Da SimGrid Team

January 21, 2015

About this Presentation

Goals and Contents

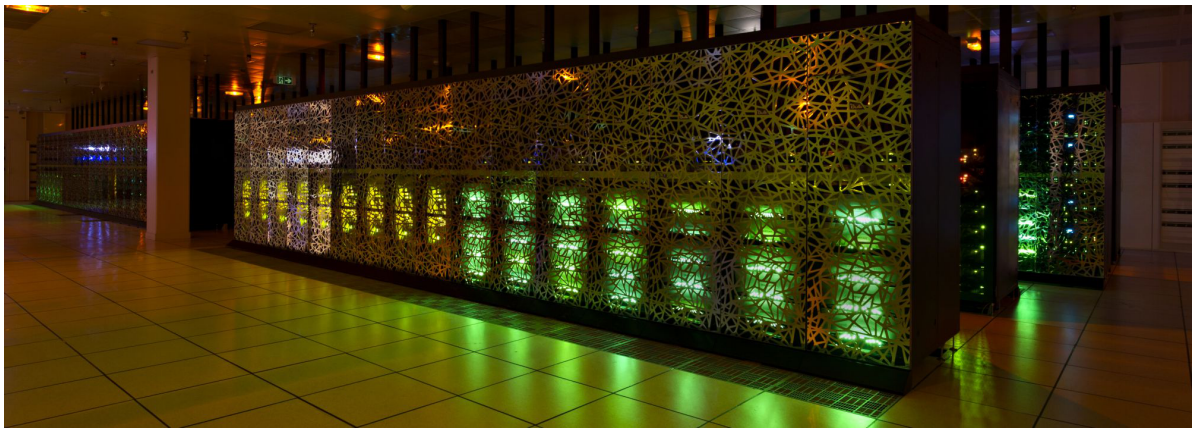
- Scientific Context: experimenting with distributed applications
- Context of SimGrid: Historical considerations; State of the Art
- Key Features of the Framework: bigger, faster, smarter, better

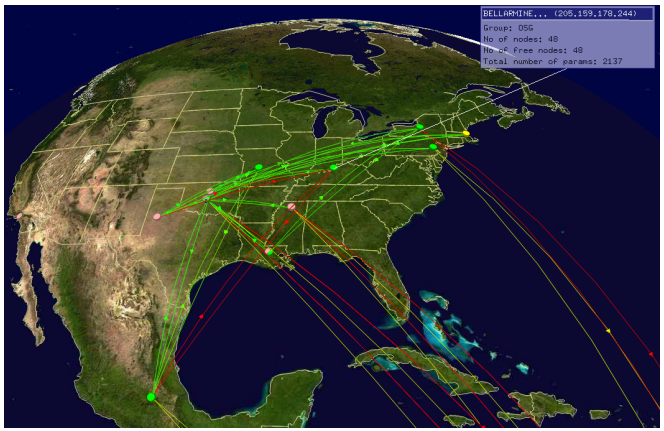
The SimGrid 101 serie

- 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
- SimGrid Contributer. Giving back to the community
- Get them from `http://simgrid.gforge.inria.fr/documentation.html`

Our Scientific Objects: Distributed Systems

Clusters, supercomputers, peer-to-peer systems, grids, clouds, ...





- HPC central to computational science; Clouds externalize everything
- Hierarchical, complex and heterogeneous; Very large and dynamic systems
- Better systems $\rightsquigarrow$ strategical advantage, reduced costs

Challenge #1: Conceive and optimize such systems

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

  Google computers dissipate 300 MW, Tianhae-2 dissipates 18 MW!!!

Assessing Distributed Applications

Correction Study $\rightsquigarrow$ Formal Methods

- Tests: Unable to provide definitive answers

Performance Study $\rightsquigarrow$ Experimentation

- Maths: Often not sufficient to fully understand these systems

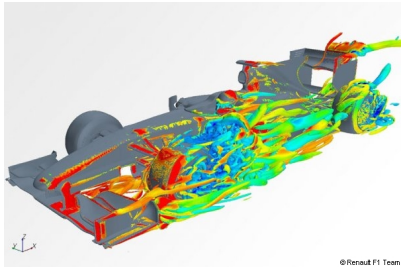
Assessing Distributed Applications

Correction Study $\rightsquigarrow$ Formal Methods

- Tests: Unable to provide definitive answers
- Model-Checking: Exhaustive and automated exploration of state space

Performance Study $\rightsquigarrow$ Experimentation

- Maths: Often not sufficient to fully understand these systems



- Experimental Facilities: Real applications on Real platform *(in vivo)*
- Simulation: Prototypes of applications on system's Models *(in silico)*

Assessing Distributed Applications

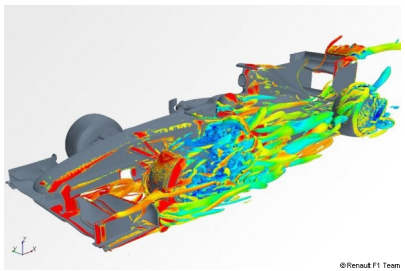
Correction Study $\rightsquigarrow$ Formal Methods

- Tests: Unable to provide definitive answers
- Model-Checking: Exhaustive and automated exploration of state space

Performance Study $\rightsquigarrow$ Experimentation

- Maths: Often not sufficient to fully understand these systems



- 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)*

Why Simulating Distributed Systems??

Challenge #0: Access to such systems?

These systems are like teenage sex: everyone talks about it,
nobody really knows how to do it, everyone thinks everyone else is doing it,
so everyone claims they are doing it... *– Dan Ariely (Duke University)*

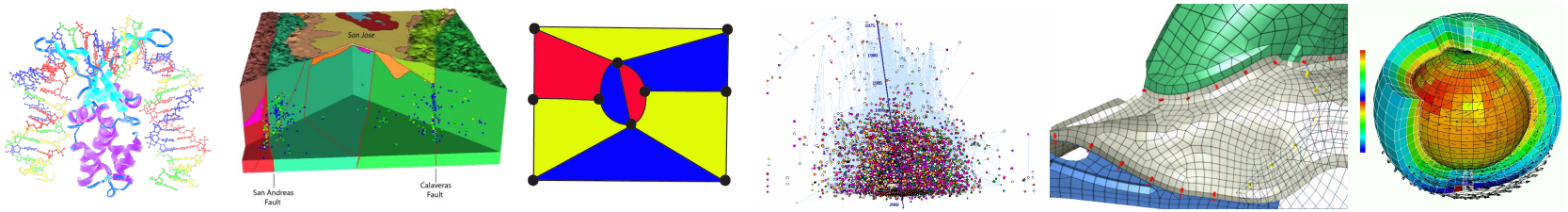
Why are Theoretical Studies not enough?

- These artificial artifacts present an unpreceded complexity
  - Heterogeneous components, Dynamic and Complex platforms
  - Numerous: milions of cores expected within the decade (ExaScale)
  - Large: Linux kernel only is 15M lines – over 10 times Encyclopedia Britanica

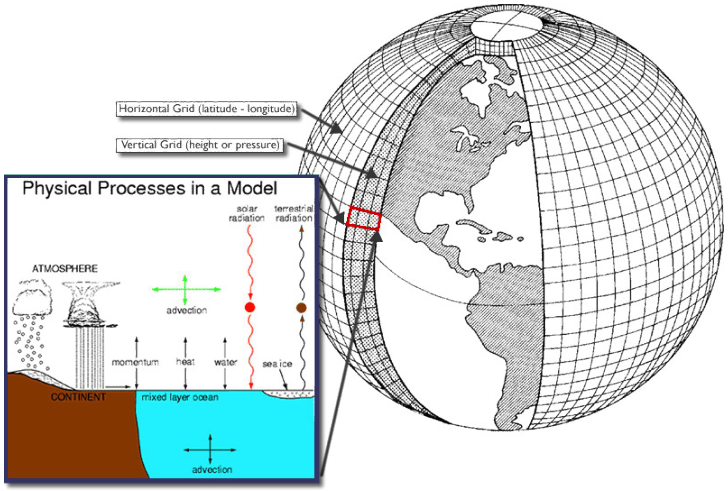
Toward a Computational Science of Distributed Computer Systems

- Empirically consider Distributed Systems as "Natural" Objects
- Other sciences routinely use computers to understand complex systems

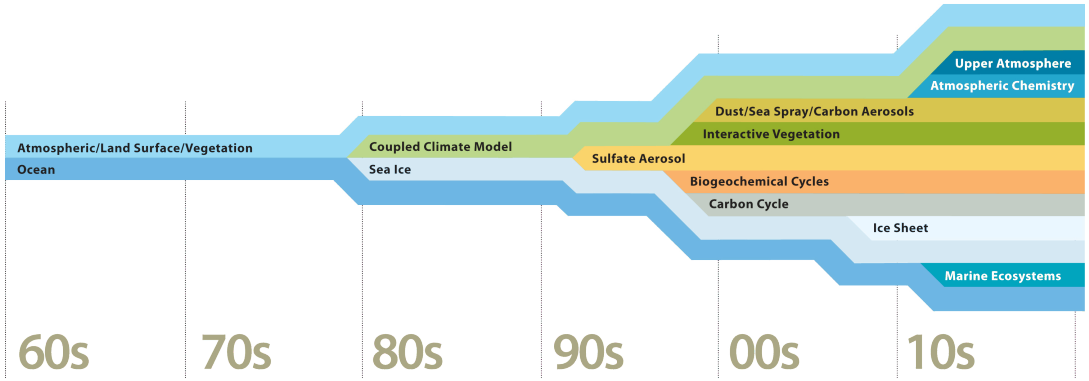
Computational Science



- Understanding the Climate Change with Predictions



- Models complexity grows



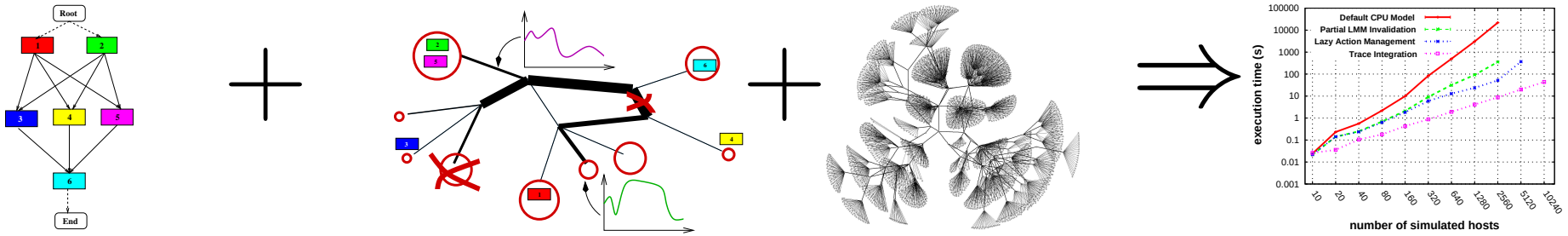
This requires **large** computers

UPSCALE project:
15,000 computing-years in 2012!

Simulating Distributed Systems

Big Idea: Simulation is the fastest path from idea to data

Idea to test | Experimental setup | Simulation Model | Scientific results



Comfortable for users

- Get preliminary results from partial implementations
- Experimental campaign with thousands of runs within the week
- Test your scientific idea, don't fiddle with technical subtleties (yet)

Challenges for the tools

- 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 the user, it's void

Computational Science of Distributed Systems?

Requirements for a Scientific Approach

- Reproducible results: read a paper, reproduce the results and improve
- Standard tools that Grad students can learn quickly

Current practice in the field is quite different

- Experimental settings not detailed enough in literature
- Many short-lived simulators; few sound and established tools
  - Grid/Cloud: OptorSim, GridSim, GroudSim, CloudSim, iCanCloud, ...
  - Volunteer Computing: SimBA, EmBOINC, SimBOINC, ...
  - P2P: PeerSim, P2PSim, OverSim, ...
  - HPC: PSINS, LogGOPSim, BigSim, MPI-SIM, ...
  - ...

SimGrid: Versatile Simulator of Distributed Apps

Toward **Computational Science** of Large-Scale Distributed Systems

Scientific Instrument

- Versatile: Grid, P2P, HPC, Volunteer Computing and others
- Sound: Validated, Scalable, Usable; Modular; Portable
- Open: Grounded +100 papers; 100 members on simgrid-user@; LGPL

Scientific Object (and lab)

- Allows comparison of network models on non-trivial applications
- Experimental Model-Checker; full Emulator under way

Scientific Project since 15 years

- Initially a toolbox to factorize code between PhD students, in 1999
- Soon a collaboration Loria / Inria Rhône-Alpes / CCI-N2P3 / U. Hawaii
- Funding and support from INRIA since 2002
- Funding from French ANR: USS SimGrid (08-11) and SONGS (12-16)

SimGrid History

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

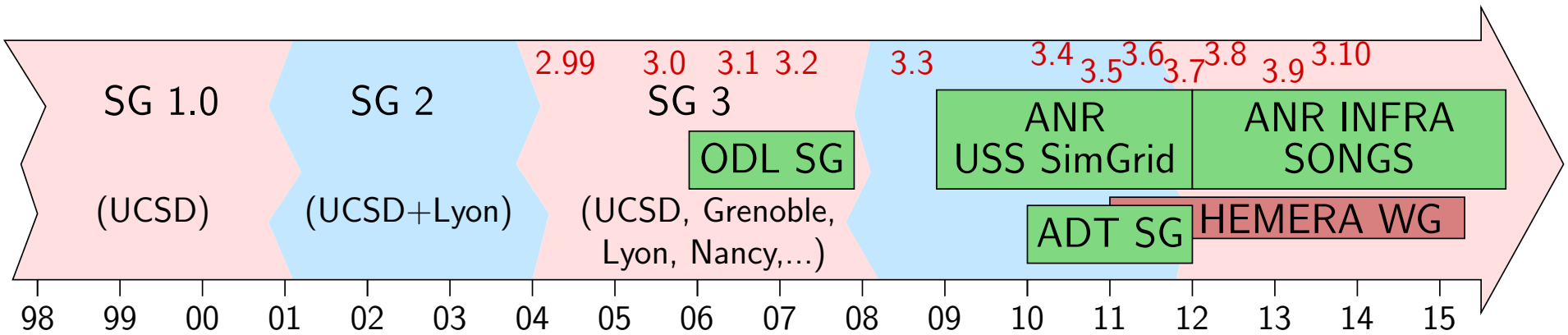
2001-2003 Infancy: CSP and improved models

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

2008-2011 Maturation: Scope increase to P2P; visualization

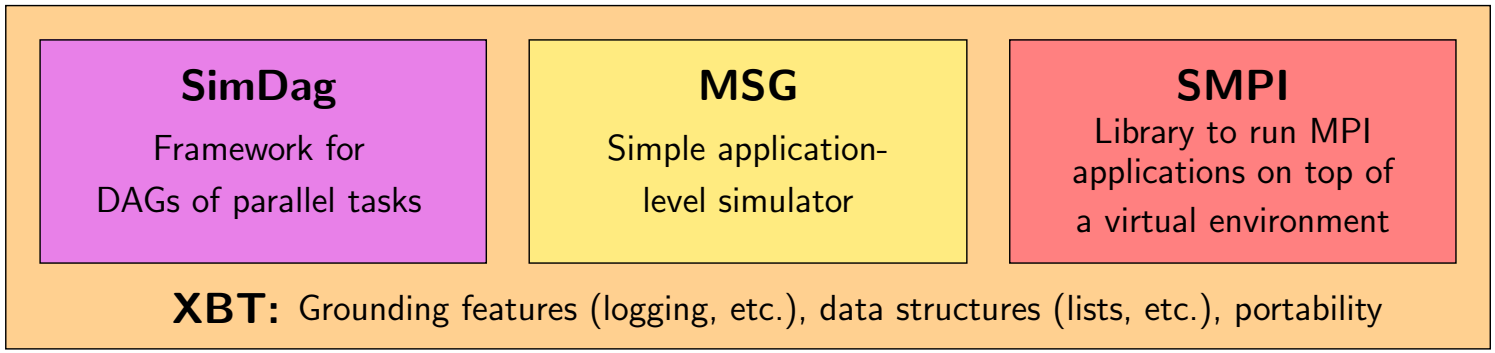
2012-: Taking the world over :)

- Further scope increase to HPC and Cloud
- Added metholologies: emulation, verification
- Mature ecosystem and community



Quick Overview of Internals Organization

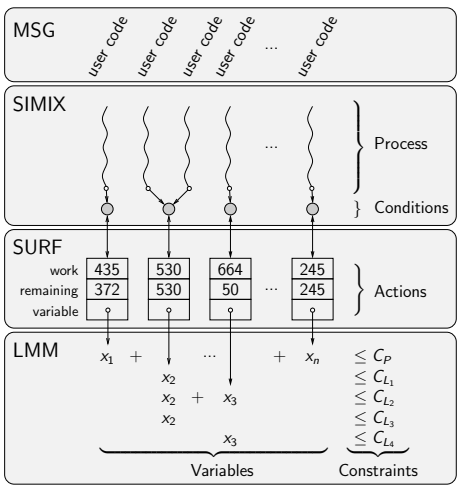
User-visible SimGrid Components



- MSG: heuristics as Concurrent Sequential Processes; Java bindings (+lua)
- SimDag: heuristics as DAG of (parallel) tasks
- SMPI: simulate real applications written using MPI

SimGrid is Strictly Layered internaly

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



Simulation Validity

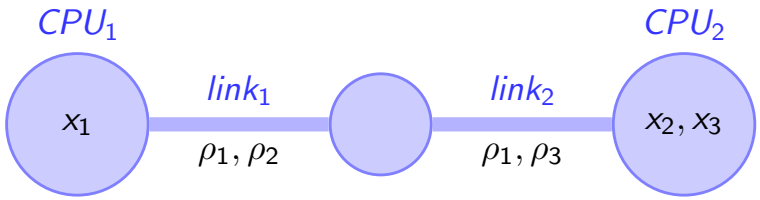
SotA: Models in most simulators are either simplistic, wrong or not assessed

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

SimGrid provides several network models

- Fast flow-based model, toward realism and speed (by default)
  Accounts for Contention, Slow-start, TCP congestion, Cross-traffic effects
- Constant time: A bit faster, but no hope of realism
- Coordinate-based: Easier to instantiate in P2P scenarios
- Packet-level: NS3 bindings
- Controlled by command line switches (exact comparison on a given application)
- See SURF 101 for details on the models and their validity

Max-Min Fairness between Network Flows



$$x_1 \leq Power\_CPU_1 \quad (1a)$$
$$x_2 + x_3 \leq Power\_CPU_2 \quad (1b)$$
$$\rho_1 + \rho_2 \leq Power\_link_1 \quad (1c)$$
$$\rho_1 + \rho_3 \leq Power\_link_2 \quad (1d)$$

Computing the sharing between flows

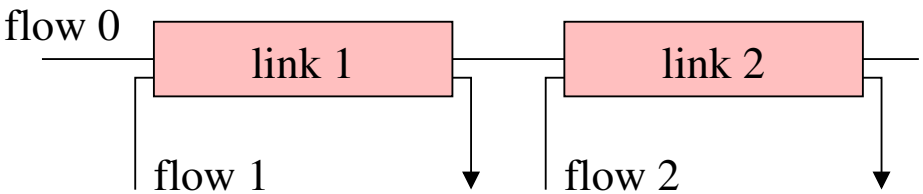
- Objective function: maximize $\min_{f \in \mathcal{F}}(\rho_f)$ [Massoulié & Roberts 2003]
- Equilibrium: increasing any ρ_f decreases a ρ'_f (with $\rho_f > \rho'_f$)
- (actually, that's a simplification of our real objective function)

Efficient Algorithm

1. Search for the bottleneck link l so that: $\frac{C_l}{n_l} = min\left\{\frac{C_k}{n_k},\ k \in \mathcal{L}\right\}$
2. This determines any flow f on this link: $\rho_f = \frac{C_l}{n_l}$
3. Update all n_k and C_k to remove these flows; Loop until all ρ_f are fixed

Max-Min Fairness Example

Homogeneous Linear Network



$C_1 = C \qquad n_1 = 2$
$C_2 = C \qquad n_2 = 2$

$\rho_0 =$
$\rho_1 =$
$\rho_2 =$

- All links have the same capacity C
- Each of them is limiting. Let's choose link 1

$\Rightarrow$ $\rho_0 = C/2$ and $\rho_1 = C/2$

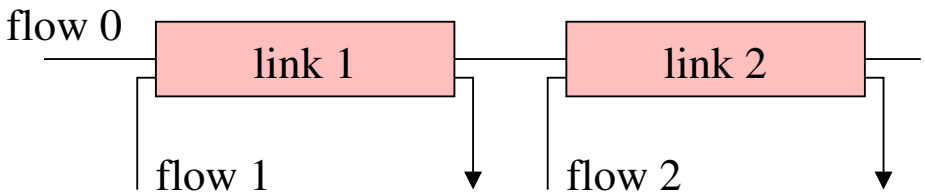
- Remove flows 0 and 1; Update links' capacity
- Link 2 sets $\rho_1 = C/2$.
- We are done computing the bandwidths ρ_i

SimGrid Implementation is **efficient**

- Lazy updates, Trace integration, preserving Cache locality

Max-Min Fairness Example

Homogeneous Linear Network



$C_1 = C \qquad n_1 = 2$
$C_2 = C \qquad n_2 = 2$

$\rho_0 = C/2$
$\rho_1 = C/2$
$\rho_2 =$

- All links have the same capacity C
- Each of them is limiting. Let's choose link 1

$\Rightarrow$ $\rho_0 = C/2$ and $\rho_1 = C/2$

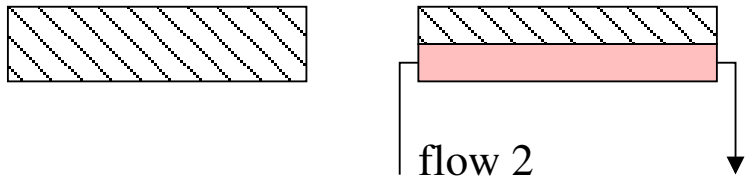
- Remove flows 0 and 1; Update links' capacity
- Link 2 sets $\rho_1 = C/2$.
- We are done computing the bandwidths ρ_i

SimGrid Implementation is **efficient**

- Lazy updates, Trace integration, preserving Cache locality

Max-Min Fairness Example

Homogeneous Linear Network



$C_1 = 0$ $\quad n_1 = 0$
$C_2 = C/2$ $\quad n_2 = 1$

$\rho_0 = C/2$
$\rho_1 = C/2$
$\rho_2 =$

- All links have the same capacity C
- Each of them is limiting. Let's choose link 1

$\Rightarrow$ $\rho_0 = C/2$ and $\rho_1 = C/2$

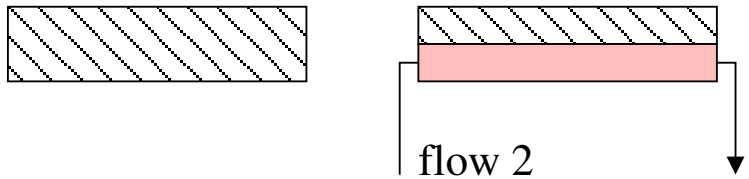
- Remove flows 0 and 1; Update links' capacity
- Link 2 sets $\rho_1 = C/2$.
- We are done computing the bandwidths ρ_i

SimGrid Implementation is **efficient**

- Lazy updates, Trace integration, preserving Cache locality

Max-Min Fairness Example

Homogeneous Linear Network



$C_1 = 0$ $\quad n_1 = 0$
$C_2 = 0$ $\quad n_2 = 0$

$\rho_0 = C/2$
$\rho_1 = C/2$
$\rho_2 = C/2$

- All links have the same capacity C
- Each of them is limiting. Let's choose link 1

$\Rightarrow$ $\rho_0 = C/2$ and $\rho_1 = C/2$

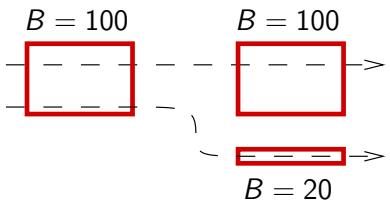
- Remove flows 0 and 1; Update links' capacity
- Link 2 sets $\rho_1 = C/2$.
- We are done computing the bandwidths ρ_i

SimGrid Implementation is **efficient**

- Lazy updates, Trace integration, preserving Cache locality

Simulating grids or clouds? Experts wanted!

Naive flow models documented as wrong

Setting	Expected Output	Output
		

Known issue in Narses (2002), OptorSim (2003), GroudSim (2011).

Validation by general agreement

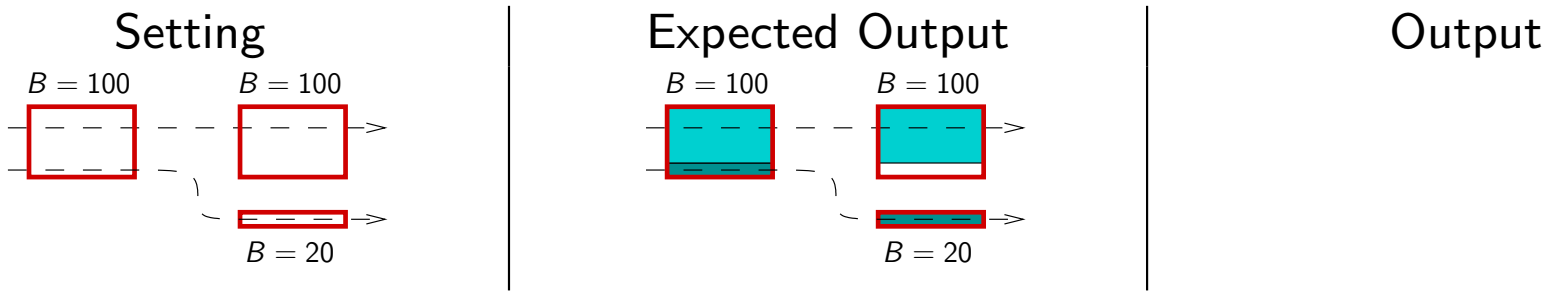
"Since SimJava and GridSim have been extensively utilized in conducting cutting edge research in Grid resource management by several researchers, bugs that may compromise the validity of the simulation have been already detected and fixed." – CloudSim, ICPP'09

Setting	Expected Output	Output

Buggy flow model (GS 5.2, 11/2010). Similar issues with naive packet-level models

Simulating grids or clouds? Experts wanted!

Naive flow models documented as wrong



Known issue in Narses (2002), OptorSim (2003), GroudSim (2011).

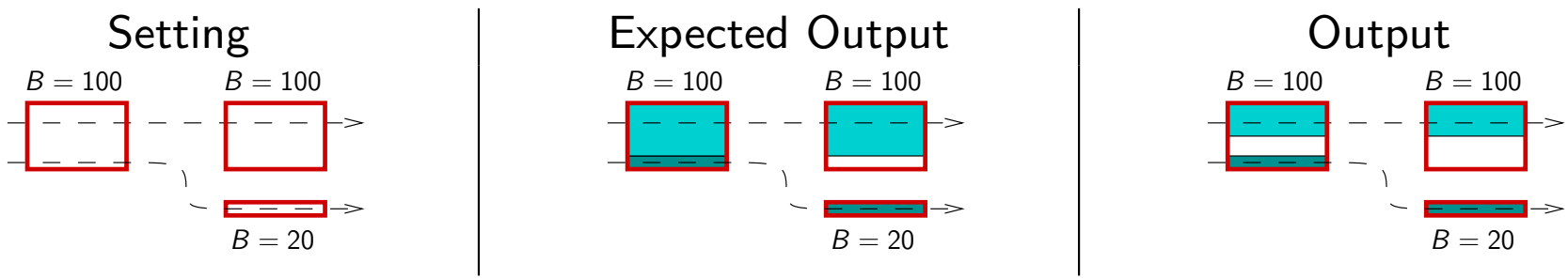
Validation by general agreement

"Since SimJava and GridSim have been extensively utilized in conducting cutting edge research in Grid resource management by several researchers, bugs that may compromise the validity of the simulation have been already detected and fixed." – CloudSim, ICPP'09

Buggy flow model (GS 5.2, 11/2010). Similar issues with naive packet-level models

Simulating grids or clouds? Experts wanted!

Naive flow models documented as wrong



Known issue in Narses (2002), OptorSim (2003), GroudSim (2011).

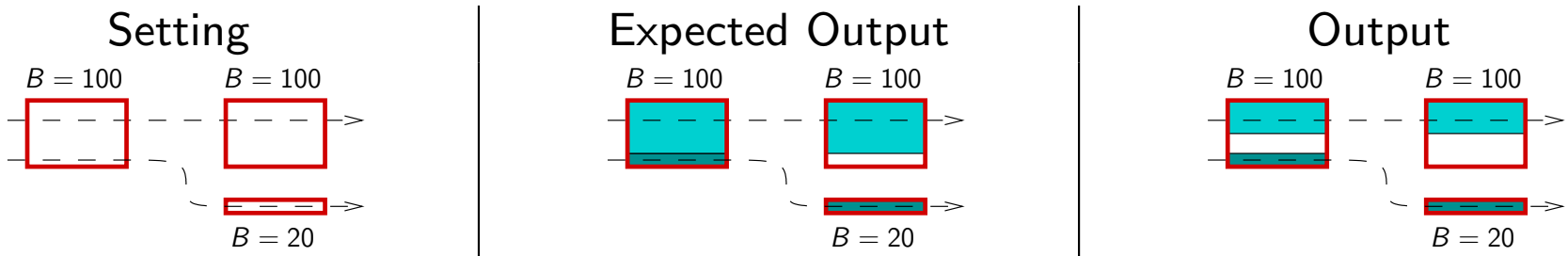
Validation by general agreement

"Since SimJava and GridSim have been extensively utilized in conducting cutting edge research in Grid resource management by several researchers, bugs that may compromise the validity of the simulation have been already detected and fixed." – CloudSim, ICPP'09

Buggy flow model (GS 5.2, 11/2010). Similar issues with naive packet-level models

Simulating grids or clouds? Experts wanted!

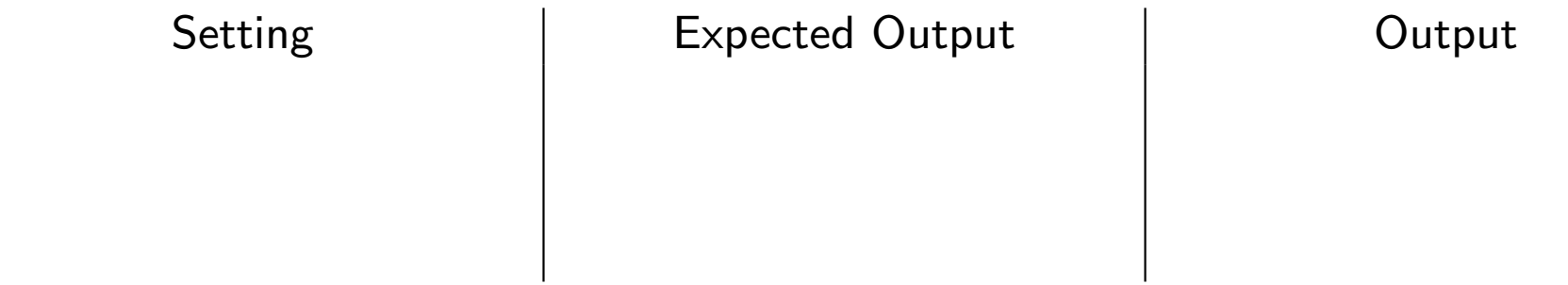
Naive flow models documented as wrong



Known issue in Narses (2002), OptorSim (2003), GroudSim (2011).

Validation by general agreement

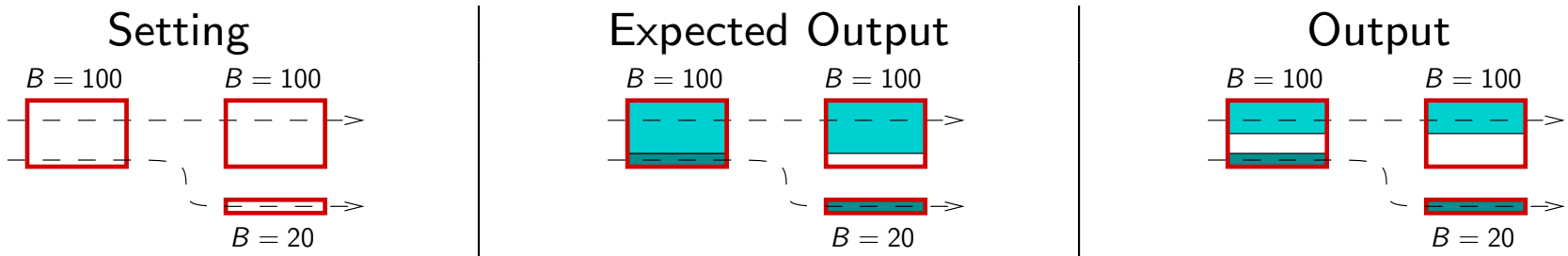
"Since SimJava and GridSim have been extensively utilized in conducting cutting edge research in Grid resource management by several researchers, bugs that may compromise the validity of the simulation have been already detected and fixed." – CloudSim, ICPP'09



Buggy flow model (GS 5.2, 11/2010). Similar issues with naive packet-level models

Simulating grids or clouds? Experts wanted!

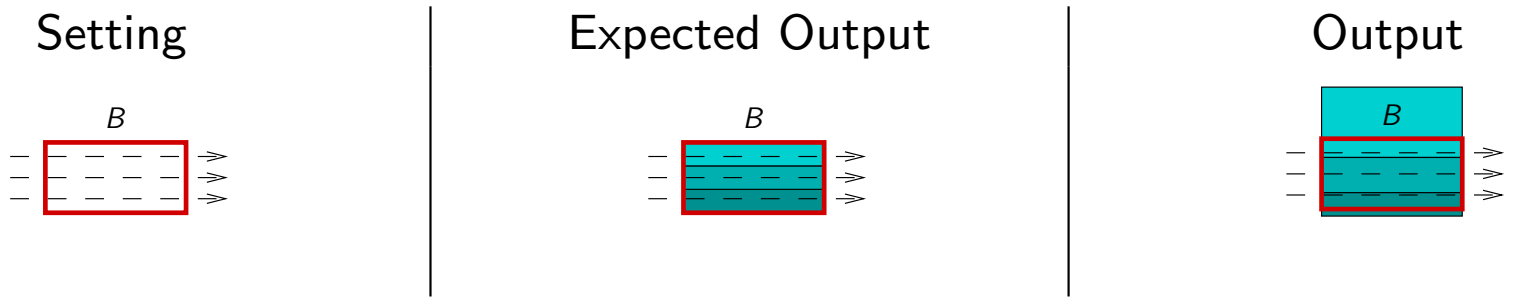
Naive flow models documented as wrong



Known issue in Narses (2002), OptorSim (2003), GroudSim (2011).

Validation by general agreement

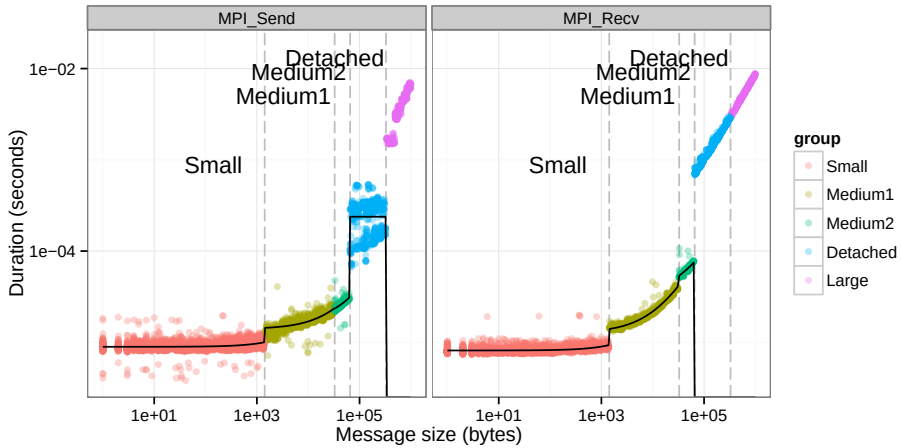
"Since SimJava and GridSim have been extensively utilized in conducting cutting edge research in Grid resource management by several researchers, bugs that may compromise the validity of the simulation have been already detected and fixed." – CloudSim, ICPP'09



Buggy flow model (GS 5.2, 11/2010). Similar issues with naive packet-level models

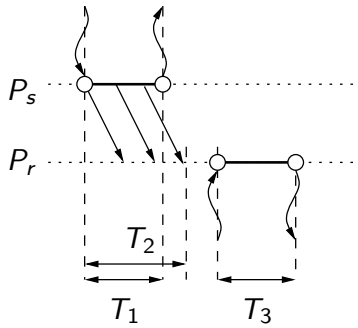
SimGrid Network Models

Measurements

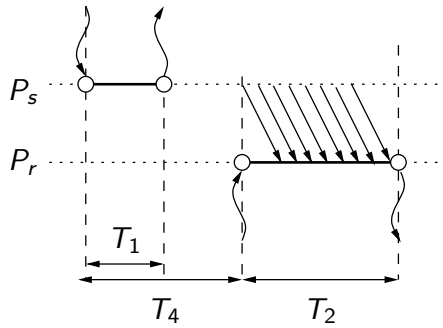


Hybrid Model

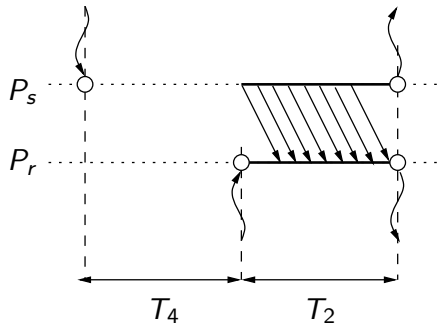
Asynchronous ($k \leq S_a$)



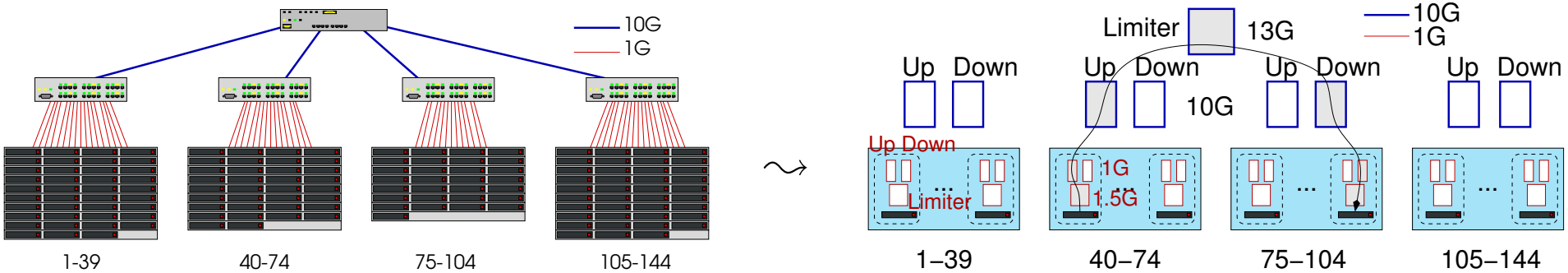
Detached ($S_a < k \leq S_d$)



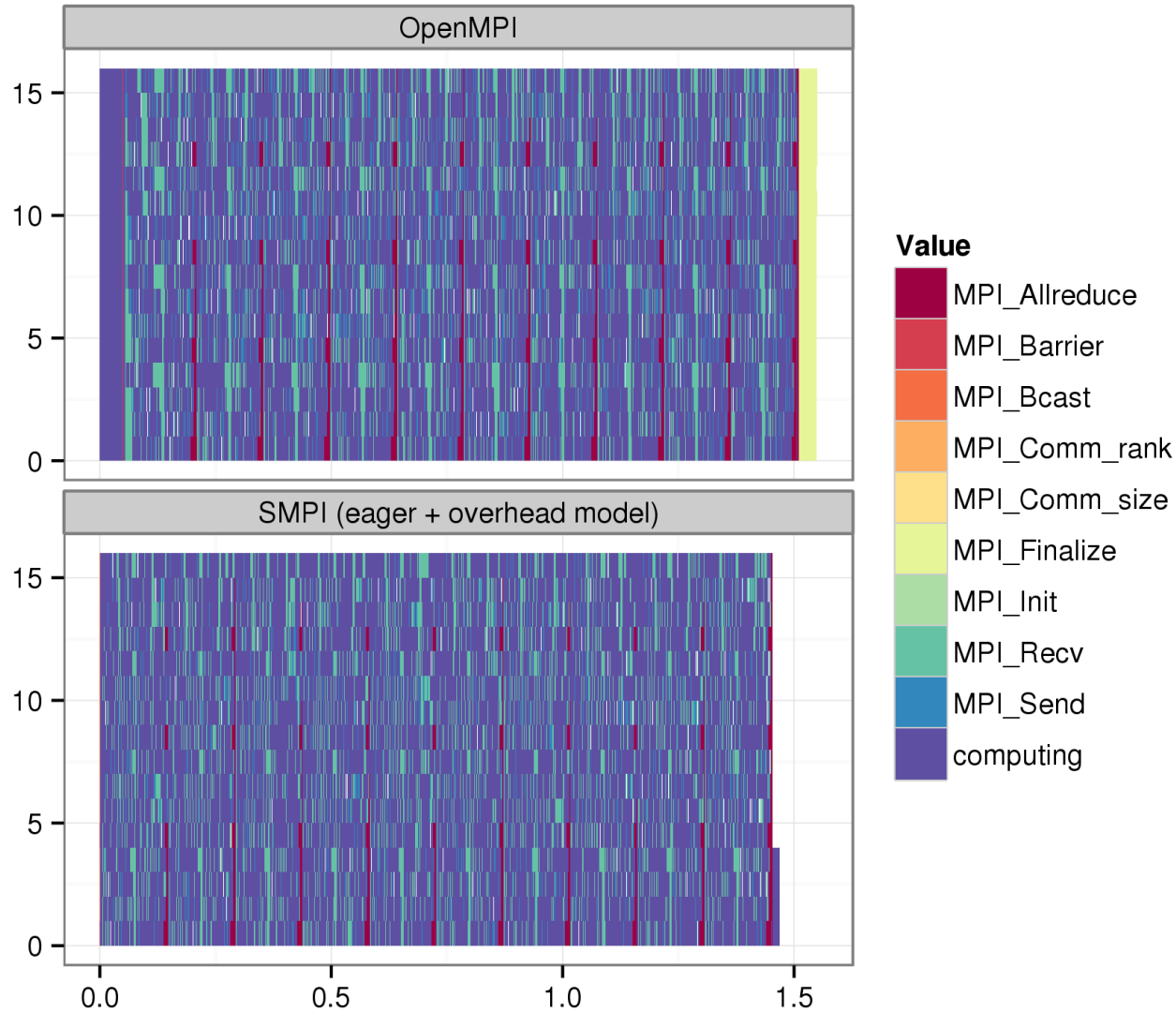
Synchronous ($k > S_d$)



Fluid model: account for contention and network topology



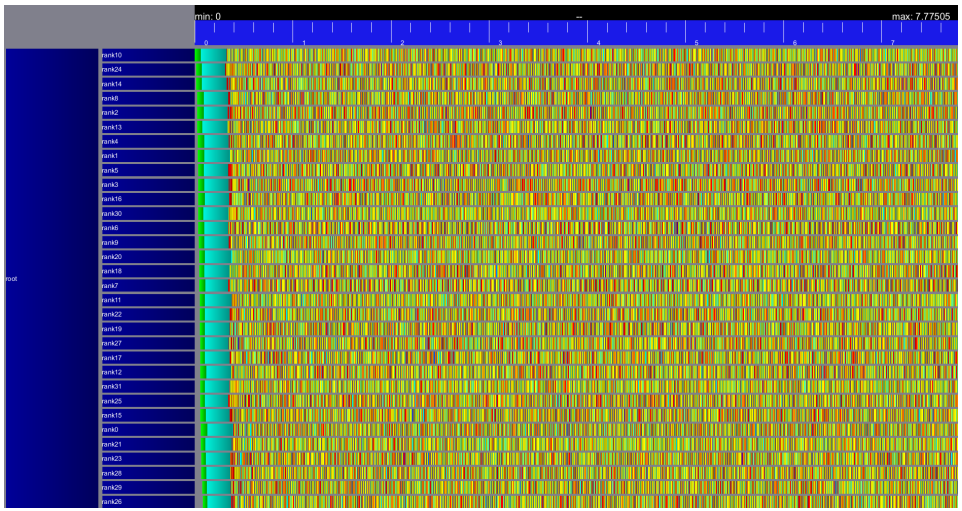
And this actually works!

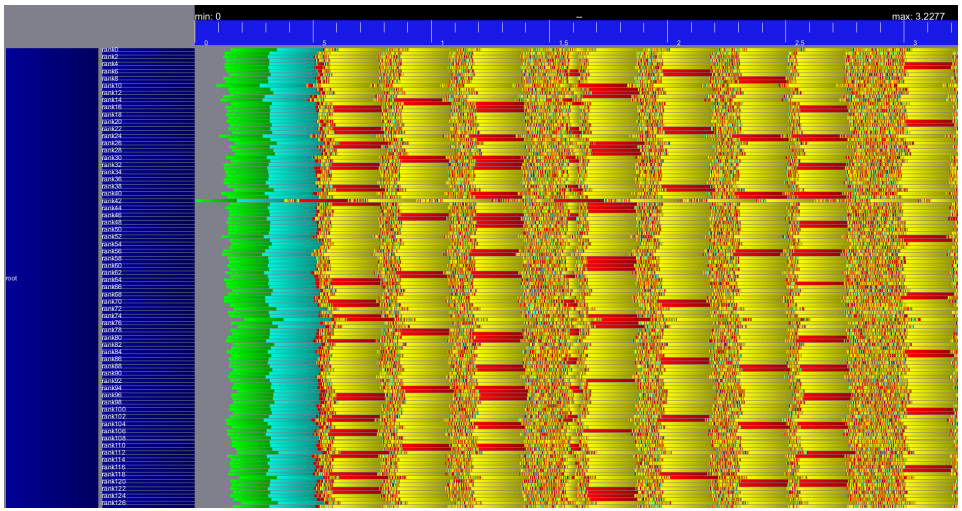


- Sweep3D: Simple Application (but not trivial) predicted in all details
- Graphene (16 procs), OpenMPI, TCP, Gigabit Ethernet achieved without overfiting :)

Reality often . . . surprizing

TCP sucks





- NAS CG on Graphene, with 128 processes
- Highly congested $\rightsquigarrow$ TCP reduce the emissions
- When speed reaches 0, it timeouts after 200ms, resets, and start over
- (TCP_RTO should help alleviating this bug – but doesn't)

We could model these effects

- **but actually, you want to fix reality**
- We wanted to understand the systems with models, what a success!

SimGrid Scalability

Simulation Versatility should not hinder Scalability

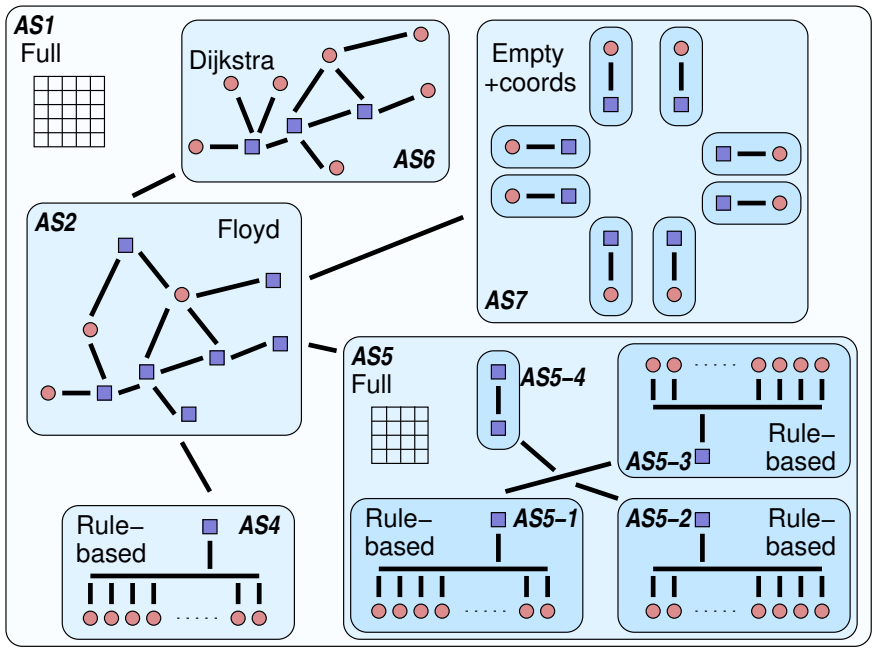
- Two aspects: Big enough (large platforms) $\oplus$ Fast enough (large workload)

Versatile yet Scalable Platform Descriptions

- Hierarchical organization in ASes
  $\rightsquigarrow$ cuts down complexity
  $\rightsquigarrow$ recursive routing
- Efficient on each classical structures
  Flat, Floyd, Star, Coordinate-based
- Allow bypass at any level

$\rightsquigarrow$ Grid'5000 platform in 22KiB
(10 sites, 40 clusters, 1500 nodes)

$\rightsquigarrow$ King's dataset in 290KiB
(2500 nodes, coordinate-based)



How big and how fast? (1/3 – Grid and VC)

Comparison to GridSim

A master distributes 500,000 fixed size jobs to 2,000 workers (round robin)

	GridSim	SimGrid
Network model	delay-based model	flow model
Topology	none	Grid5000
Time	1h	14s
Memory	4.4GB	165MB

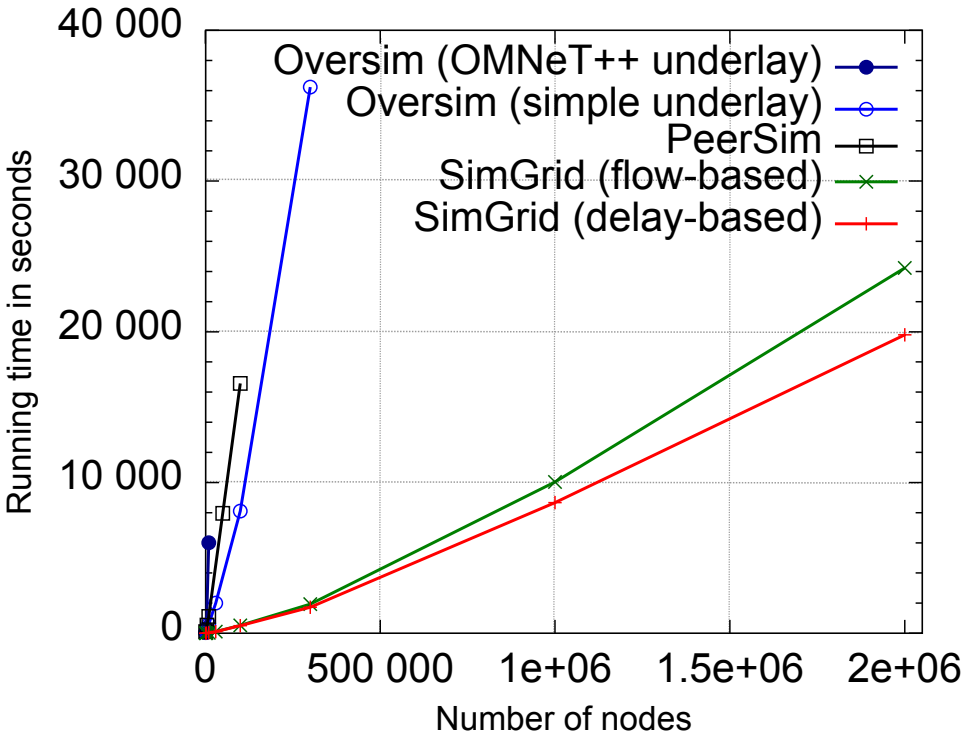
Volunteer Computing settings

- Loosely coupled scenario as in Boinc
- SimGrid: full modeling (clients and servers), precise network model
- SimBA: Servers only, descisions based on simplistic markov modeling

$\rightsquigarrow$ SimGrid shown 25 times faster

How big and how fast? (2/3 – P2P)

- Scenario: Initialize Chord, and simulate 1000 seconds of protocol
- Arbitrary Time Limit: 12 hours (kill simulation afterward)



Largest simulated scenario

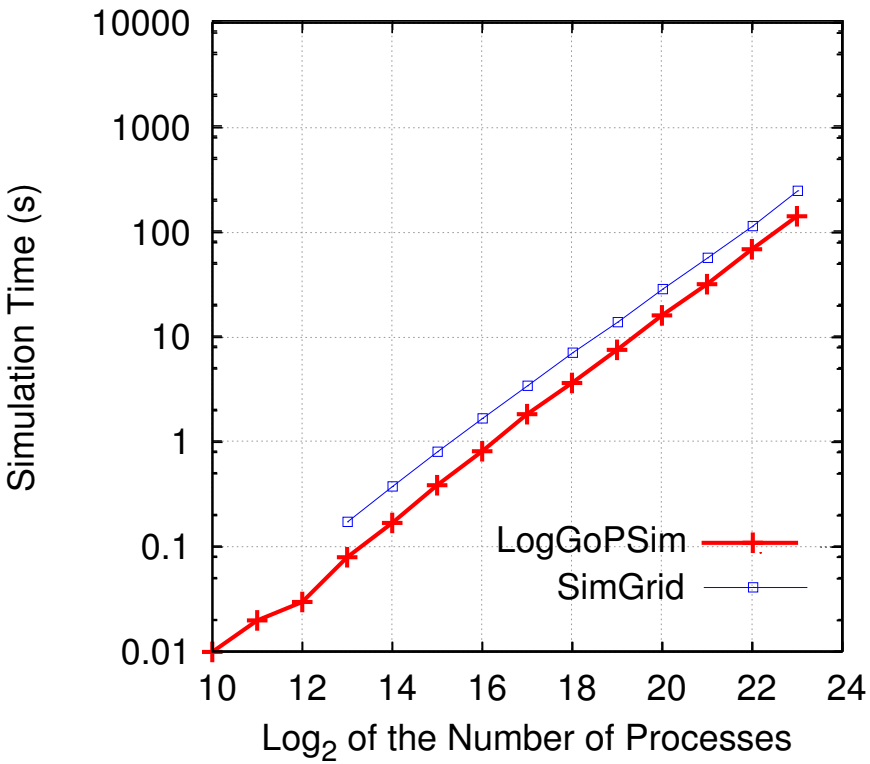
Simulator	size	time
OverSim (OMNeT++)	10k	1h40
OverSim (simple)	300k	10h
PeerSim	100k	4h36
SG (flow-based)	10k	130s
	300k	32mn
	2M*	6h23
SG (delay-based)	2M	5h30

∗ 36GB = 18kB/ process (16kB for the stack)

- Orders of magnitude more scalable than state-of-the-art P2P simulators
- Precise model incurs a $\approx 20\%$ slowdown, but accuracy is not comparable
- Also, parallel simulation (faster simulation at scale); Distributed sim. ongoing

How big and how fast? (3/3 – HPC)

Simulating a binomial broadcast



Model:

- SimGrid: contention + cabinets hierarchy
- LogGOPSim: simple delay-based model

Results:

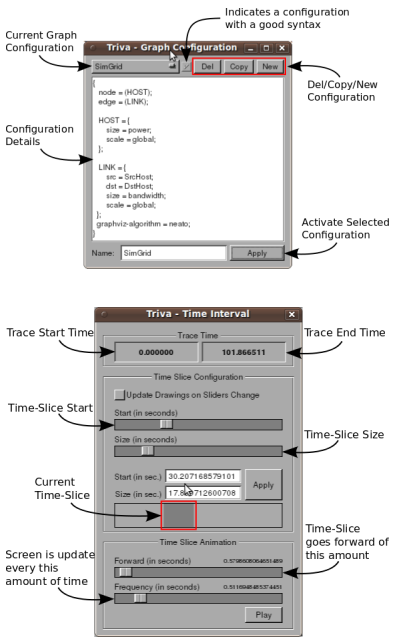
- SimGrid is roughly 75% slower
- SimGrid is about 20% more fat (15GB required for 2^{23} processors)

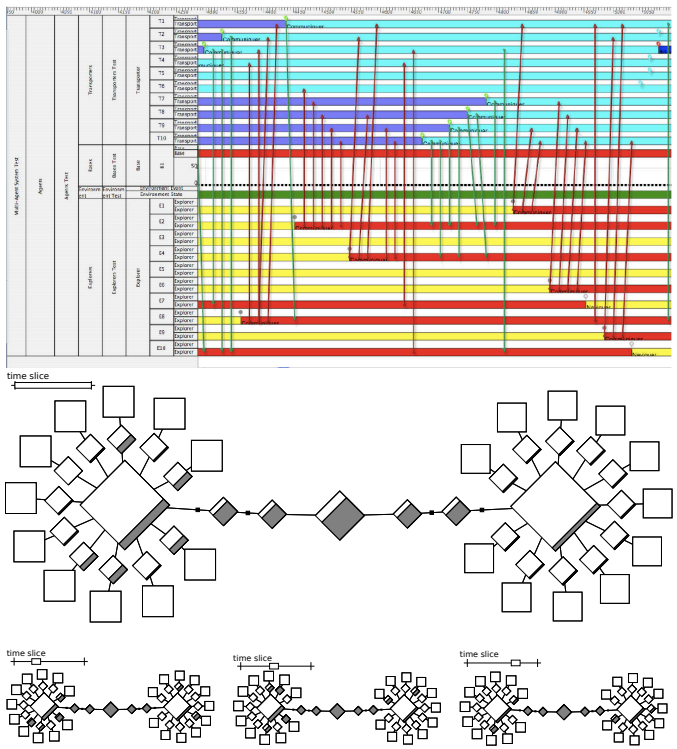
The genericity of SimGrid data structures comes at the cost of a slight overhead

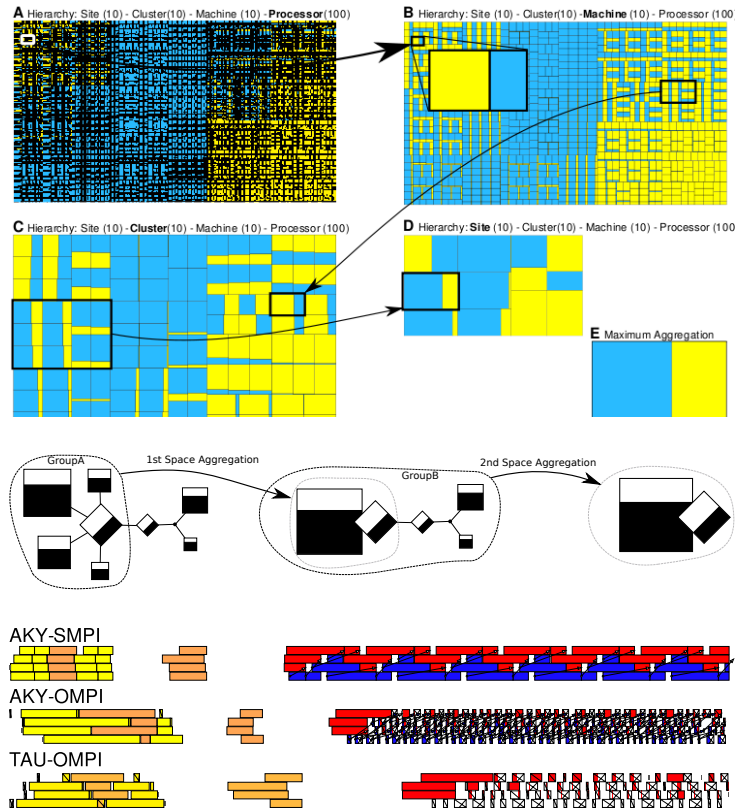
BUT scalability does not necessarily comes at the price of realism

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)



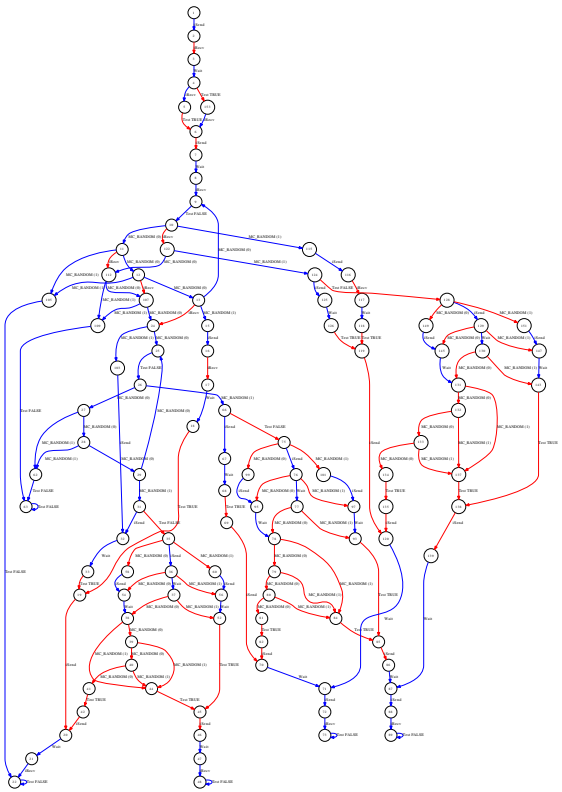




Dynamic Verification with SimGrid

Verifying safety and liveness properties

- Works on real C code, using Dwarf to introspect state
- Explicitely explores the execution graph
- DPOR-based reduction techniques (safety only) or State equality reduction
- Mostly suited for bug finding (no certification)



Current state

- Usable in MSG and SMPI (in C)
- Found *wild* bugs in medium-sized programs (Chord protocol)
- Verified collectives of MPICH (in minutes)

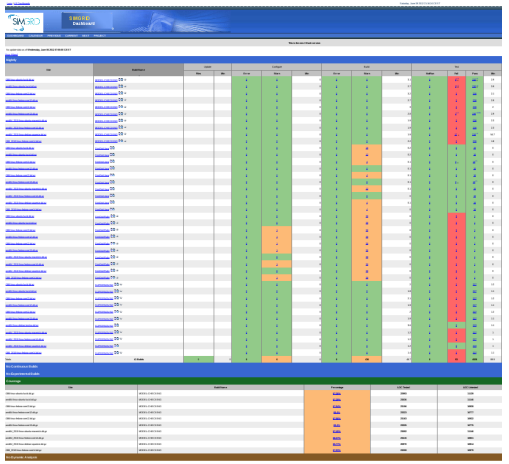
Ongoing work

- Verify larger applications
- Ensure send determinism (for checkpointing)

Practical Trust onto SimGrid?

- Internal code base rather complex because of hacks for versatile efficiency

Continuous Integration



- Current version tested every night
- 450 integration tests; 10,000 unit tests; 70% coverage
- 2 SimGrid configurations on 10 Linux versions
- Performance regression testing soon operational

Release tests

- Windows and Mac considered as additional release goals
- Actually works on all Debian arch.: hurd, kfreebsd, mips, arm, ppc, s390 ;)

This is free software anyway

- The code base is currently LGPL (probably soon GPL)
- Come, check it out and participate! (5 of 25 commiters not affiliated to us)

Take Away Messages

SimGrid will prove helpful to your research

- Versatile: Used in several communities (scheduling, Grids, HPC, P2P, Clouds)
- Accurate: Model limits known thanks to validation studies
- Sound: Easy to use, extensible, fast to execute, scalable to death, well tested
- Open: LGPL; User-community much larger than contributors group
- Around since 15 years, and ready for at least 15 more years

Welcome to the Age of (Sound) Computational Science

- Discover: `http://simgrid.gforge.inria.fr/`
- Learn: other 101 tutorials, user manual and examples
- Join: user mailing list, #simgrid on irc.debian.org
  We even have some open positions ;)