Computational Science of Computer Systems

Méthodologies d'expérimentation pour l'informatique distribuée à large échelle

Martin Quinson

ENS Rennes

With the SimGrid Team: Arnaud Legrand, Frédéric Suter, H. Casanova, S Merz, Anne-Cécile Orgerie, G. Corona, A. Degomme, and *many* others.

24 mai 2016

E3 RSD

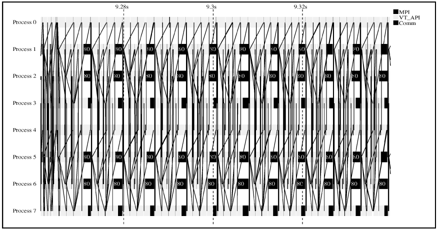
Super Computers

- A worldwide ranking (Top500) $\rightsquigarrow$ a worldwide competition
- 100,000–1,000,000 cores, accelerators (GPU, Xeon Phi) + fast interconnect
- Among the most complex artefacts ever built

Complex Applications

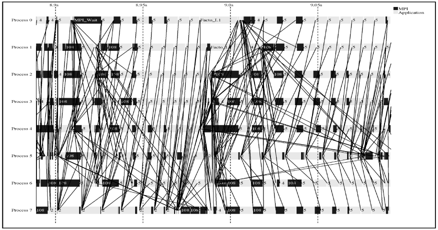
Large scale hybrid machines $\rightsquigarrow$ Novel programming approaches

Rigid, hand tuned



SuperLU

Task-based and dynamic



MUMPS

Huge Societal Impact

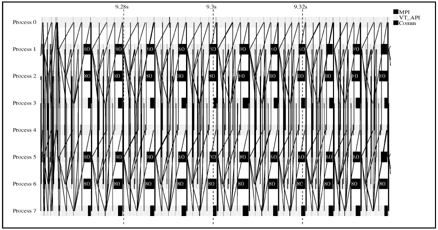
- Computational Science fuel *every branches* of Science and Technologies
- Cloud Computing virtualizes IT in always larger data centers
- Google disspates 300MW, BotNets control millions of zombies computers

How to study these beasts?

Complex Applications

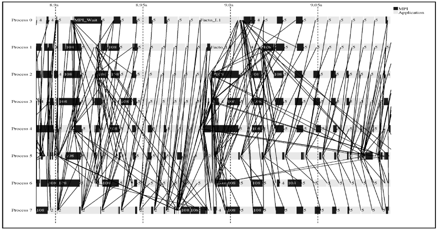
Large scale hybrid machines ⇝ Novel programming approaches

Rigid, hand tuned



SuperLU

Task-based and dynamic



MUMPS

Huge Societal Impact

- Computational Science fuel *every branches* of Science and Technologies
- Cloud Computing virtualizes IT in always larger data centers
- Google disspates 300MW, BotNets control millions of zombies computers

How to study these beasts?

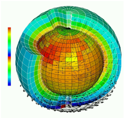
- Proposal: Computational Science of Computer Systems

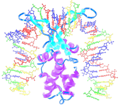
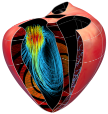
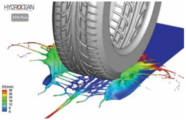
Computational Science

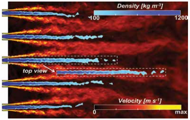
In a Nutshell

- Mathematical models of phenomena
- Simulation on super-computers
- *In*validation: predictions vs. observations
  Experimental evaluation of Theories
- Then results without running experiments

Computational Science



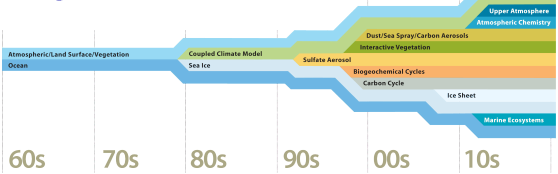






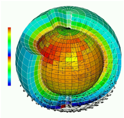
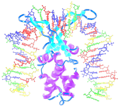
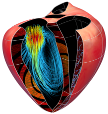
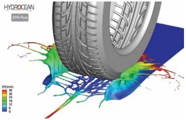
In a Nutshell

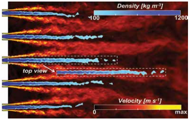
- Mathematical models of phenomena
- Simulation on super-computers
- *In*validation: predictions vs. observations
  Experimental evaluation of Theories
- Then results without running experiments


Larger Models for better predictions



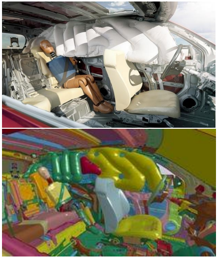
Computational Science

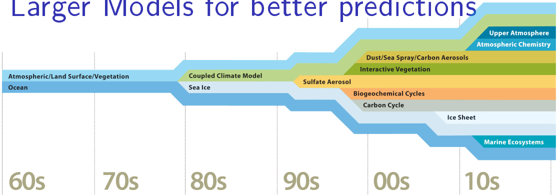


In a Nutshell

- Mathematical models of phenomena
- Simulation on super-computers
- *In*validation: predictions vs. observations
  Experimental evaluation of Theories
- Then results without running experiments



Larger Models for better predictions



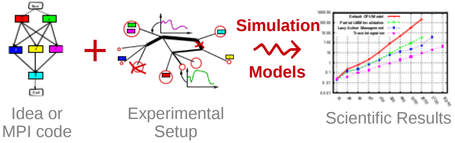
Our proposal

Apply this idea to Computer Systems

Simulating Distributed Systems

Simulation: Fastest Path from Idea to Data

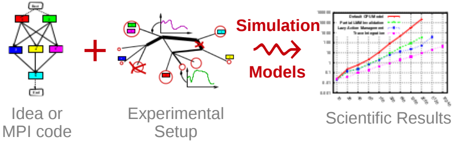
- Test your scientific idea with a fast and confortable scientific instrument



Simulating Distributed Systems

Simulation: Fastest Path from Idea to Data

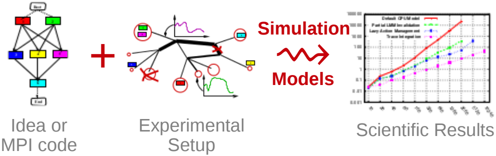
- Test your scientific idea with a fast and confortable scientific instrument

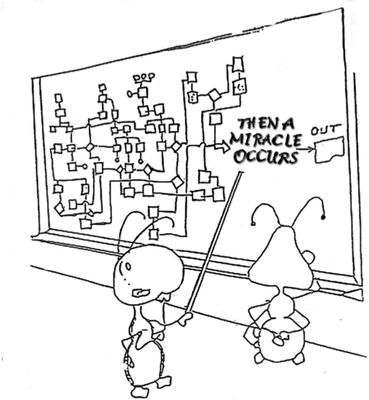


Simulation: Easiest Way to Study Real Distributed Systems

- Centralized and reproducible setup. Don't waste resources to debug and test
- No Heisenbug, full Clairevoyance, High Reproducibility, *What if* studies
- Also software/hardware co-design, capacity planning or hardware qualification

Simulation Challenges





Challenges for the Tool Makers

- Validity: Get realistic results (controlled experimental bias)
- Scalability: *Fast enough* and *Big enough*
- Open Science: Integrated lab notes, runner, post-processing (data provenance)

Simulation of Parallel/Distributed Systems

Network Protocols: Standards emerged: GTNetS, DaSSF, OmNet++, NS3
Huge amount of non-standard tools in other domains:

- Grid Computing — OptorSim, ChicagoSim, GridSim, JFreeSim ...
- Peer-to-peer — P2Psim, SimP2P, PeerSim, OverSim ...
- Volunteer Computing — SimBA, EmBOINC, SimBOINC ...
- HPC/MPI — Dimemas, PSinS, BigSim, LogGoPSim, XSim, SST ...
- Cloud Computing — CloudSim, GroudSim, iCanCloud, GreenCloud ...

This raises severe methodological/reproducibility issues:

- Short-lived, badly supported (software QA), sparse validity assessment

Simulation of Parallel/Distributed Systems

Network Protocols: Standards emerged: GTNetS, DaSSF, OmNet++, NS3
Huge amount of non-standard tools in other domains:

- Grid Computing — OptorSim, ChicagoSim, GridSim, JFreeSim …
- Peer-to-peer — P2Psim, SimP2P, PeerSim, OverSim …
- Volunteer Computing — SimBA, EmBOINC, SimBOINC …
- HPC/MPI — Dimemas, PSinS, BigSim, LogGoPSim, XSim, SST …
- Cloud Computing — CloudSim, GroudSim, iCanCloud, GreenCloud …

This raises severe methodological/reproducibility issues:

- Short-lived, badly supported (software QA), sparse validity assessment

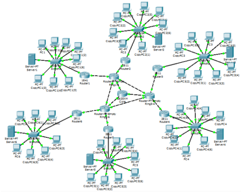
SimGrid: a 15 years old joint project

- Versatile: Grid, P2P, Clouds, HPC, Volunteer, etc
- Collaborative: (ANR, CNRS, Univ., Inria) Open Source: active community
- Widely used: 150 publications by 120 individuals, 30 contributors

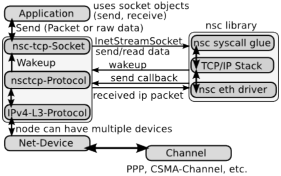
http://simgrid.org/

Classical Network Models: Hands and Feets

Packet-level Simulators

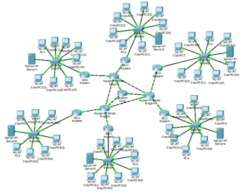


- ▶ Full network stack
- ☺ Very detailled
- ☹ Hard to instanciate
- ☹ Very slow
- ☹ Hard to reason about

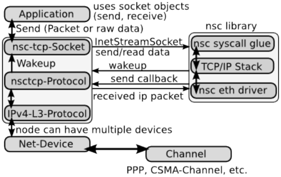


Classical Network Models: Hands and Feets

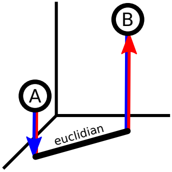
Packet-level Simulators



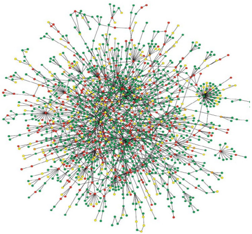
- ▶ Full network stack
- ☺ Very detailled
- ☹ Hard to instanciate
- ☹ Very slow
- ☹ Hard to reason about



Simplistic Models



- ▶ Constant/Random delay
- ▶ N-d coordinates
- ☺ Very scalable
- ☹ No topology
- ☹ *No network congestion*

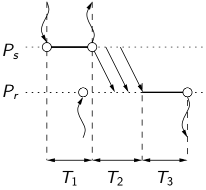


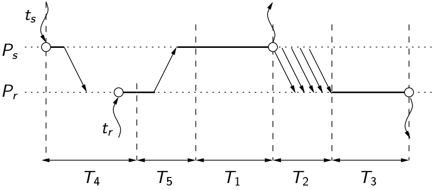
is there a third way?

The LogP Model

- First Goal: complexity analysis and algorithm design
- Accounts for delays and protocol switch

Asynchronous mode ($k \leqslant \boxed{S}$) Rendez-vous mode ($k > S$)





$$T_1 = o + kO_s \quad T_2 = \begin{cases} L + kg & \text{if } k < \boxed{s} \\ L + sg + (k-s)G & \text{otherwise} \end{cases} \quad T_3 = o + kO_r \qquad \dots$$

MPI_Send	$k \leqslant S$	T_1
	$k > S$	$T_4 + T_5 + T_1$
MPI_Recv	$k \leqslant S$	$\max(T_1 + T_2 - (t_r - t_s), 0) + T_3$
	$k > S$	$\max(o + L - (t_r - t_s), 0) + o + T_5 + T_1 + T_2 + T_3$

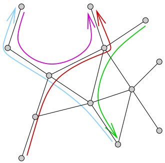
- Nice approximated model of perfect machine; Ignores contention, topology, etc

Fluid Network Model

In a Nutshell

- Assume that data = water ; network link = pipe
- Delay $= Lat_{i,j} + Size/Bw_{i,j}$
- Compute bandwidth sharing on macroscopic events (assumming steady state)

  Bandwidth sharing as an optimization problem



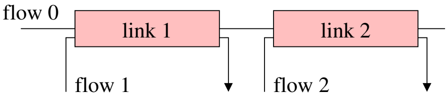
$$\sum_{\text{if flow i uses link j}} \rho_i \leqslant C_j$$

  - Max-Min objective function: $\max(\min(\rho_i))$

Results

- Actually models congestion very well in steady state
- Can be enriched to model slow start and cross traffic congestion
  Corrective coefficients, Other objective functions
- Can be implemented very efficiently

Max-Min Fairness, Homogeneous Linear Network

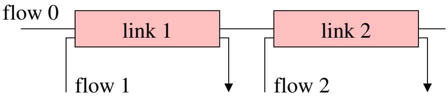


$C_1 = C \qquad n_1 = 2$
$C_2 = C \qquad n_2 = 2$

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

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

Max-Min Fairness, Homogeneous Linear Network



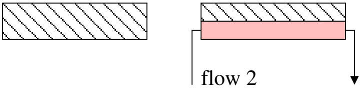
$$C_1 = C \qquad n_1 = 2$$
$$C_2 = C \qquad n_2 = 2$$

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

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

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

Max-Min Fairness, Homogeneous Linear Network



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

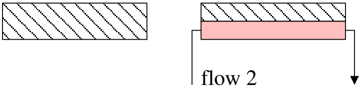
$\rho_0 = C/2$
$\rho_1 = C/2$
$\rho_2 =$

- 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

Max-Min Fairness, Homogeneous Linear Network



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

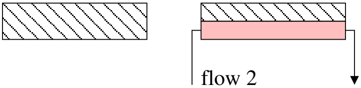
$\rho_0 = C/2$
$\rho_1 = C/2$
$\rho_2 = C/2$

- 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 $\rho_1 = C/2$

Max-Min Fairness, Homogeneous Linear Network



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

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

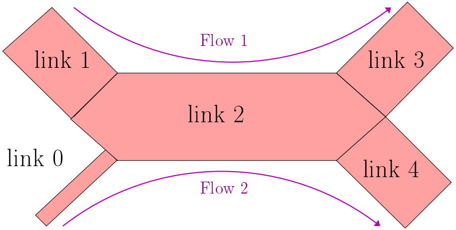
- All links have the same capacity C
- Each of them is limiting. Let's choose link 1

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

- Remove flows 0 and 1; Update links' capacity
- Link 2 sets $\rho_1 = C/2$

We're done computing the bandwidth allocated to each flow

Max-Min Fairness, Backbone

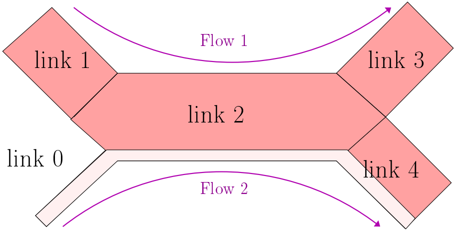


$C_0 = 1$ $n_0 = 1$
$C_1 = 1000$ $n_1 = 1$
$C_2 = 1000$ $n_2 = 2$
$C_3 = 1000$ $n_3 = 1$
$C_4 = 1000$ $n_4 = 1$

$\rho_1 =$
$\rho_2 =$

- The limiting link is link 0 $\left(\text{since } \frac{1}{1} = min\left(\frac{1}{1}, \frac{1000}{1}, \frac{1000}{2}, \frac{1000}{1}, \frac{1000}{1}\right)\right)$

Max-Min Fairness, Backbone

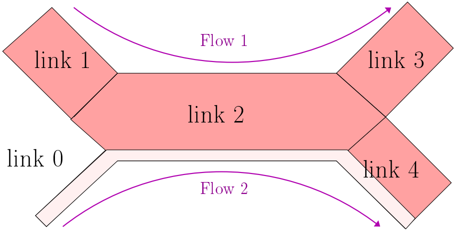


$C_0 = 0 \quad n_0 = 0$
$C_1 = 1000 \quad n_1 = 1$
$C_2 = 999 \quad n_2 = 1$
$C_3 = 1000 \quad n_3 = 1$
$C_4 = 999 \quad n_4 = 0$

$\rho_1 =$
$\rho_2 = 1$

- The limiting link is link 0 $\left(\text{since } \frac{1}{1} = min\left(\frac{1}{1}, \frac{1000}{1}, \frac{1000}{2}, \frac{1000}{1}, \frac{1000}{1}\right)\right)$
- This fixes $\rho_2 = 1$. Update the links

Max-Min Fairness, Backbone

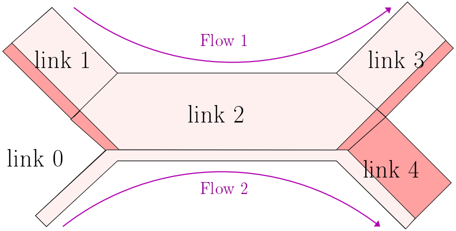


$$\begin{array}{ll} C_0 = 0 & n_0 = 0 \\ C_1 = 1000 & n_1 = 1 \\ C_2 = 999 & n_2 = 1 \\ C_3 = 1000 & n_3 = 1 \\ C_4 = 999 & n_4 = 0 \end{array}$$

$$\rho_1 =$$
$$\rho_2 = 1$$

- The limiting link is link 0 $\left(\text{since } \frac{1}{1} = min\left(\frac{1}{1}, \frac{1000}{1}, \frac{1000}{2}, \frac{1000}{1}, \frac{1000}{1}\right)\right)$
- This fixes $\rho_2 = 1$. Update the links
- The limiting link is link 2

Max-Min Fairness, Backbone

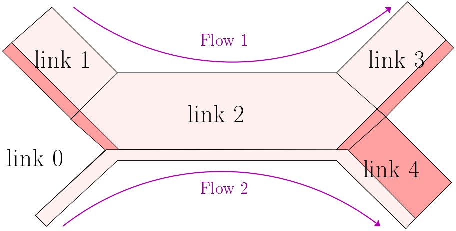


$$C_0 = 0 \quad n_0 = 0$$
$$C_1 = 1 \quad n_1 = 0$$
$$C_2 = 0 \quad n_2 = 0$$
$$C_3 = 1 \quad n_3 = 0$$
$$C_4 = 999 \quad n_4 = 0$$

$$\rho_1 = 999$$
$$\rho_2 = 1$$

- The limiting link is link 0 $\left(\text{since } \frac{1}{1} = min\left(\frac{1}{1}, \frac{1000}{1}, \frac{1000}{2}, \frac{1000}{1}, \frac{1000}{1}\right)\right)$
- This fixes $\rho_2 = 1$. Update the links
- The limiting link is link 2
- This fixes $\rho_1 = 999$

Max-Min Fairness, Backbone



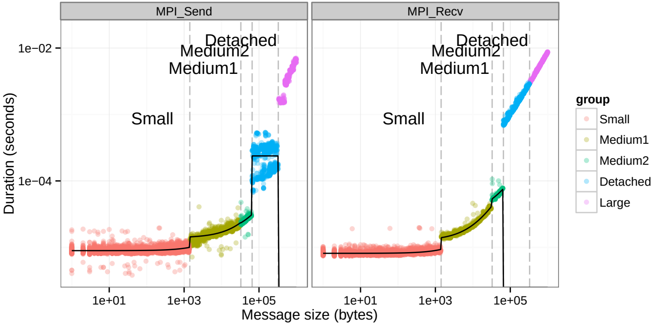
$$C_0 = 0 \quad n_0 = 0$$
$$C_1 = 1 \quad n_1 = 0$$
$$C_2 = 0 \quad n_2 = 0$$
$$C_3 = 1 \quad n_3 = 0$$
$$C_4 = 999 \quad n_4 = 0$$

$$\rho_1 = 999$$
$$\rho_2 = 1$$

- The limiting link is link 0 $\left(\text{since } \frac{1}{1} = min\left(\frac{1}{1}, \frac{1000}{1}, \frac{1000}{2}, \frac{1000}{1}, \frac{1000}{1}\right)\right)$
- This fixes $\rho_2 = 1$. Update the links
- The limiting link is link 2
- This fixes $\rho_1 = 999$
- Done. We know ρ_1 and ρ_2

MPI Point-to-Point Communication

- Performance characterization: Randomized Measurements (OpenMPI/TCP/Eth1GB)



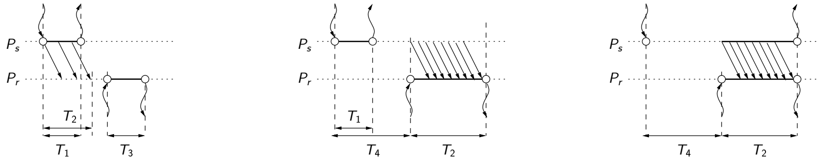
- There is a quite important variability
- There are at least 4 different modes
- It is piece-wise linear and discontinuous

Neither LogP nor Fluid seem to match

SimGrid Hybrid Network Model

LogP (small message sizes)

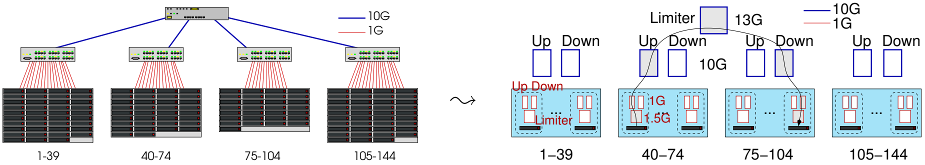
- Accounts for delay, communication modes and protocol switches



Asynchronous ($k \leqslant S_a$) Detached ($S_a < k \leqslant S_d$) Synchronous ($k > S_d$)

Fluid Model (large sizes)

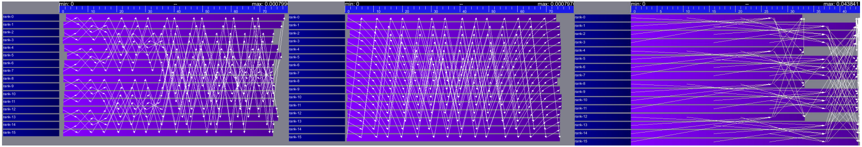
- Accounts for contention and network topology



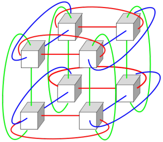
Applicative Model of MPI

MPI Collectives

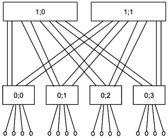
- SimGrid implements more than 120 algorithms for the 10 main MPI collectives
- Selection logic from OpenMPI, MPICH can be reproduced



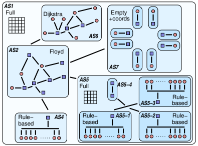
HPC Topologies



Torus



Fat-trees



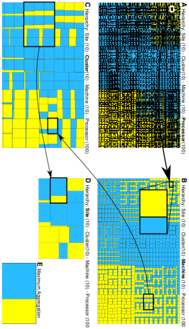
Hierarchies of ASes

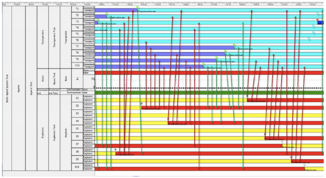
But also

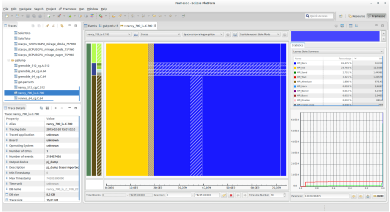
- External load (availability changes), Host and link failures, Energy (DVFS)
- Virtual Machines, that can be migrated; Random platform generators

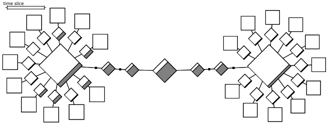
Simulation Outcomes: Visualizing Results

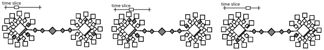
- Visualization scriptable: easy but powerful configuration; Scalable tools
- Right Information: both platform and applicative visualizations
- Right Representation: gantt charts, spatial representations, tree-graphs
- Easy navigation in space and time: selection, aggregation, animation
- Easy trace comparison: Trace diffing (not automated)

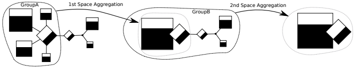


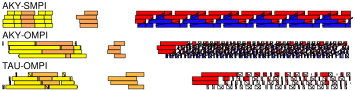






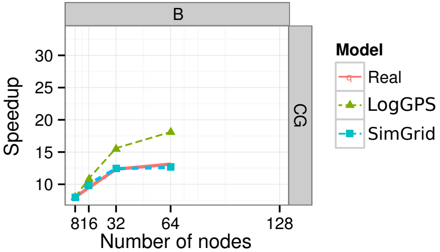


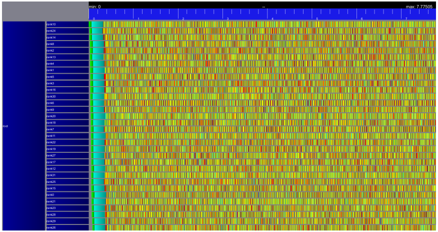




Validity Success Stories

unmodified NAS CG on a TCP/Ethernet cluster (Grid'5000)



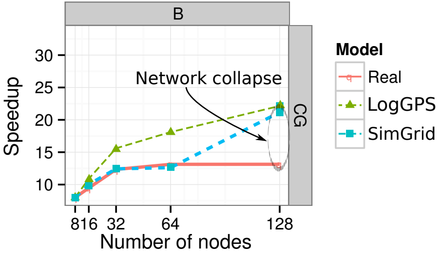


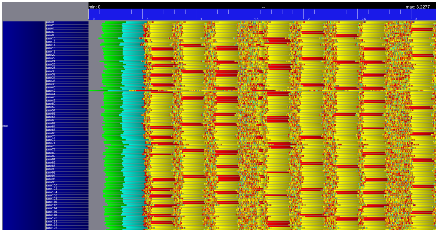
Key aspects to obtain this result

- Network Topology: Contention (large msg) and Synchronization (small msg)
- Applicative (collective) operations (stolen from real implementations)
- Instantiate Platform models (matching effects, not docs)
- All included in SimGrid but the instantiation (remains manual for now)

Validity Success Stories

unmodified NAS CG on a TCP/Ethernet cluster (Grid'5000)





Discrepency between Simulation and Real Experiment. Why?

- Massive switch packet drops lead to 200ms timeouts in TCP!
- Tightly coupled: the whole application hangs until timeout
- Noise easy to model in the simulator, but useless for that very study
- Our prediction performance is more interesting to detect the real issue

Do we got the Perfect Model yet?

Do we got the Perfect Model yet?

Perfect Model: the one making your Study sound

If you study a theoretical P2P algorithm

- You could probably go for a super-fast constant-time model

If your study is a MPI application

- with TCP LAN, SMPI should do the trick (with correct instanciatiation)
- with InfiniBand and/or GPUs, you need our still ongoing models

If you work on a TCP variant

- then you need a packet-level simulator such as NS3

If your study WAN-interconnected Set Top Boxes

- SMPI model not suited! Impossible to instanciate, validated only for MPI
- Vivaldi model intended for that kind of studies

In any case, assess the validity & soundness

Validation vs. Invalidation

Validation

- Articles with nice graphs but shallow description and no working code
- Optimistic validations on few simple cases (merely tests the implementation)

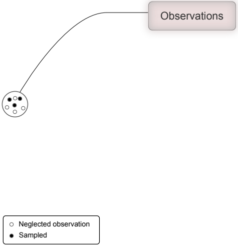
Validation vs. Invalidation

Validation

- Articles with nice graphs but shallow description and no working code
- Optimistic validations on few simple cases (merely tests the implementation)

Invalidation and *crucial experiments*

- Other sciences assess the quality of a model by trying to invalidate it [Popper]



- Observe the System you Study

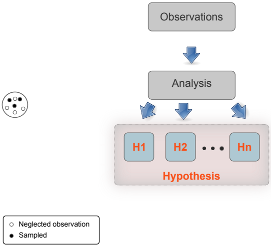
Validation vs. Invalidation

Validation

- Articles with nice graphs but shallow description and no working code
- Optimistic validations on few simple cases (merely tests the implementation)

Invalidation and *crucial experiments*

- Other sciences assess the quality of a model by trying to invalidate it [Popper]



- Observe the System you Study
- Build hypothesis from Observations

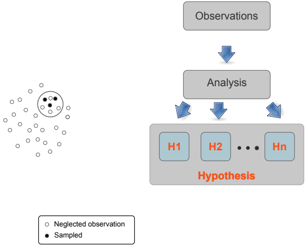
Validation vs. Invalidation

Validation

- Articles with nice graphs but shallow description and no working code
- Optimistic validations on few simple cases (merely tests the implementation)

Invalidation and *crucial experiments*

- Other sciences assess the quality of a model by trying to invalidate it [Popper]



- Observe the System you Study
- Build hypothesis from Observations
- Think of Crucial Experiments

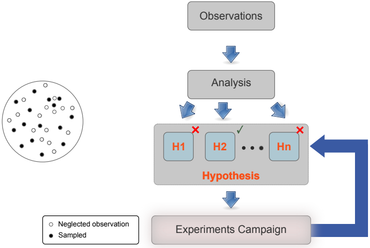
Validation vs. Invalidation

Validation

- Articles with nice graphs but shallow description and no working code
- Optimistic validations on few simple cases (merely tests the implementation)

Invalidation and *crucial experiments*

- Other sciences assess the quality of a model by trying to invalidate it [Popper]



- Observe the System you Study
- Build hypothesis from Observations
- Think of Crucial Experiments
- Experiment (dis-)prove hypothesis
- Rejected hypothesis $\rightsquigarrow$ more insight

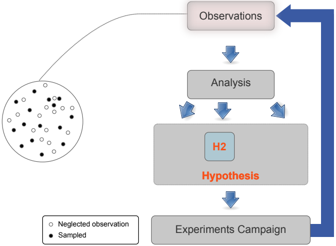
Validation vs. Invalidation

Validation

- Articles with nice graphs but shallow description and no working code
- Optimistic validations on few simple cases (merely tests the implementation)

Invalidation and *crucial experiments*

- Other sciences assess the quality of a model by trying to invalidate it [Popper]



Cyclic Process

- Observe the System you Study
- Build hypothesis from Observations
- Think of Crucial Experiments
- Experiment (dis-)prove hypothesis
- Rejected hypothesis $\rightsquigarrow$ more insight

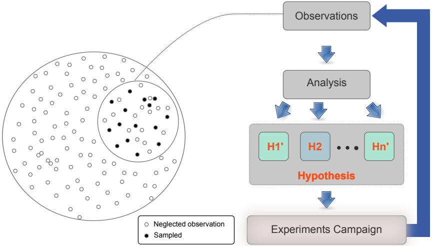
Validation vs. Invalidation

Validation

- Articles with nice graphs but shallow description and no working code
- Optimistic validations on few simple cases (merely tests the implementation)

Invalidation and *crucial experiments*

- Other sciences assess the quality of a model by trying to invalidate it [Popper]



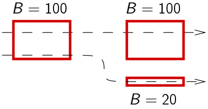
Cyclic Process

- Observe the System you Study
- Build hypothesis from Observations
- Think of Crucial Experiments
- Experiment (dis-)prove hypothesis
- Rejected hypothesis $\rightsquigarrow$ more insight

Don't trust your models and tools, always (in-)validate them!

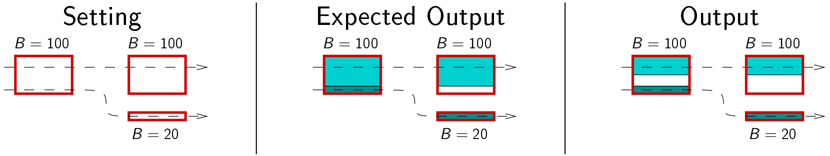
Invalidating Simulators from the Litterature

Naive flow models documented as wrong

Setting	Expected Output	Output
		

Invalidating Simulators from the Litterature

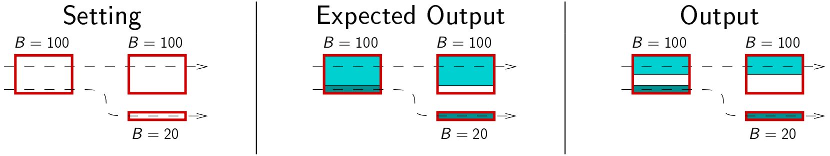
Naive flow models documented as wrong



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

Invalidating Simulators from the Litterature

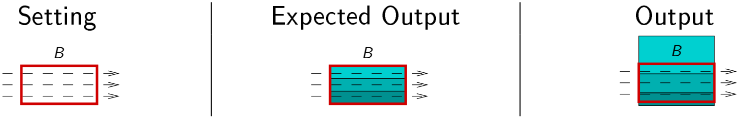
Naive flow models documented as wrong



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

Validation by general agreement

> *"Since SimJava and GridSim have been extensively utilized [...] by several researchers, bugs that may compromise the validity of the simulation have been already detected and fixed."* CloudSim, ICPP'09

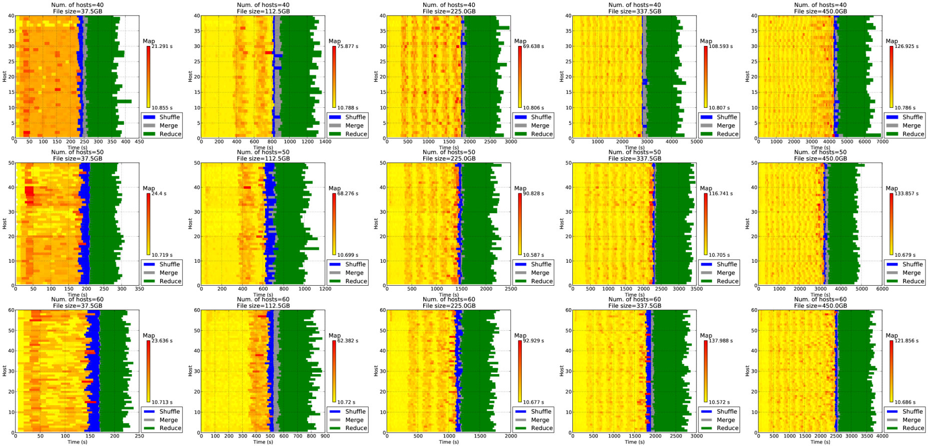


Buggy flow model in GridSim 5.2 (reported years ago, never fixed)

Building Models ⇝ Better Understanding

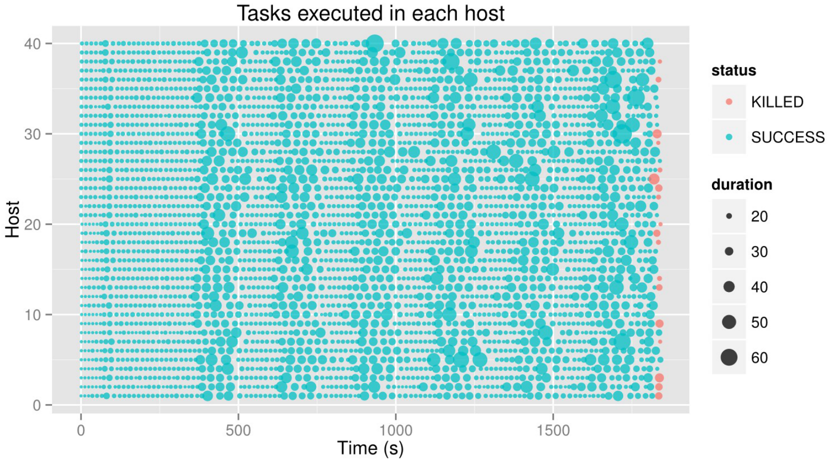
Modeling MapReduce for SimGrid from G5K experiments

- Settings: gdx@g5k; 1 Map + 1 Reduce per host; 1 replicat
- Workload: TeraSort. #hosts: 40, 50, 60; File size: from 37.5GB to 450GB



Unexpected, annoying slowdown waves

Closer Look at this Unexpected Behavior



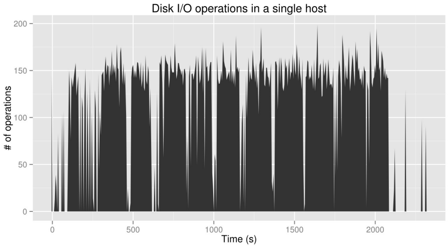
- Duration of all tasks should be roughly the same
- Many map tasks slowed down, synchronously in different hosts!

Next student next year reproducibility issues

- Orsay cluster retired, impossible to rerun there
- No slowdown wave at Nancy for months. Even when changing the application
- Finally reproduced in Sophia (on similar hardware). So it's hardware.

Next student next year reproducibility issues

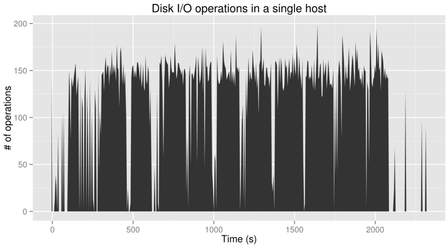
- Orsay cluster retired, impossible to rerun there
- No slowdown wave at Nancy for months. Even when changing the application
- Finally reproduced in Sophia (on similar hardware). So it's hardware.

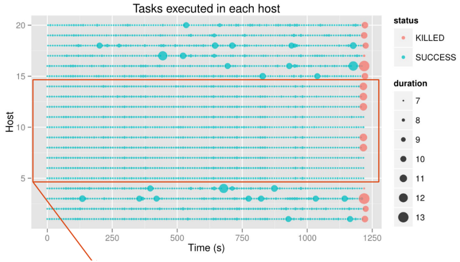


- SATA disks get saturated

Next student next year reproducibility issues

- Orsay cluster retired, impossible to rerun there
- No slowdown wave at Nancy for months. Even when changing the application
- Finally reproduced in Sophia (on similar hardware). So it's hardware.

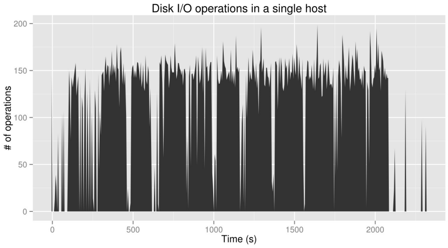


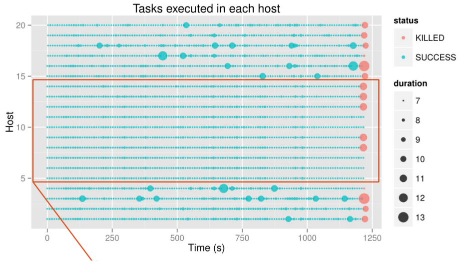


- SATA disks get saturated by Reduce, killing the machine, explaining wave

Next student next year reproducibility issues

- Orsay cluster retired, impossible to rerun there
- No slowdown wave at Nancy for months. Even when changing the application
- Finally reproduced in Sophia (on similar hardware). So it's hardware.



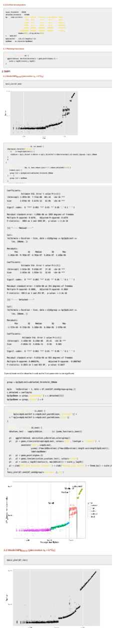


- SATA disks get saturated by Reduce, killing the machine, explaining wave
- Could fix the application or Model the phenomenon, not sure

but this was discovered by invalidating models

Understanding can be seen as a model based form of data compression.
– Gregory Chaitin

Open Science



Tedious Experiments must be Reproducible

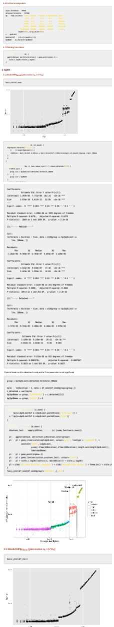
Devel in the details *vs.* Reproducibility Grail

- Describe experiments (material & method): data deluge
- Very sensible experiments: macro impact of micro errors
- Statistical Analysis gets more complex

But there is Hope!

- Grid'5000 very precious: hardware but also expertise
- Our tools (YMMV): git + org-mode + R
- *Computational scientists* already use them elsewhere

Open Science



Tedious Experiments must be Reproducible

Devel in the details *vs.* Reproducibility Grail

- Describe experiments (material & method): data deluge
- Very sensible experiments: macro impact of micro errors
- Statistical Analysis gets more complex

But there is Hope!

- Grid'5000 very precious: hardware but also expertise
- Our tools (YMMV): git + org-mode + R
- *Computational scientists* already use them elsewhere

Grumpy Reviewer #3 is not convinced.

- *I found the results section of this paper to be pretty weak: previous simulators can simulate 100,000+ procs*
- This sociological should be soon solved

Science Today: Incomplete Publications

- Publications are just the tip of the iceberg
  - Scientific record is incomplete--- to large to fit in a paper
  - Large volumes of data
  - Complex processes
- Can’t (easily) reproduce results

Science Today: Incomplete Publications

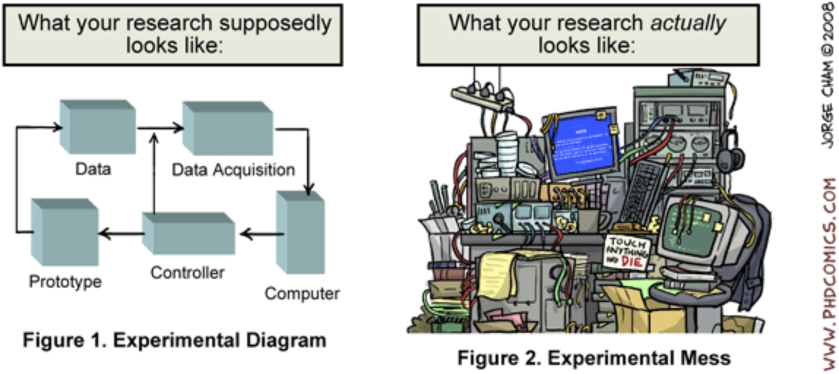
- Publications are just the tip of the iceberg
  - Scientific record is incomplete--- to large to fit in a paper
  - Large volumes of data
  - Complex processes
- Can't (easily) reproduce results

"It's impossible to verify most of the results that computational scientists present at conference and in papers." [Donoho et al., 2009]

"Scientific and mathematical journals are filled with pretty pictures of computational experiments that the reader has no hope of repeating." [LeVeque, 2009]

"Published documents are merely the advertisement of scholarship whereas the computer programs, input data, parameter values, etc. embody the scholarship itself." [Schwab et al., 2007]

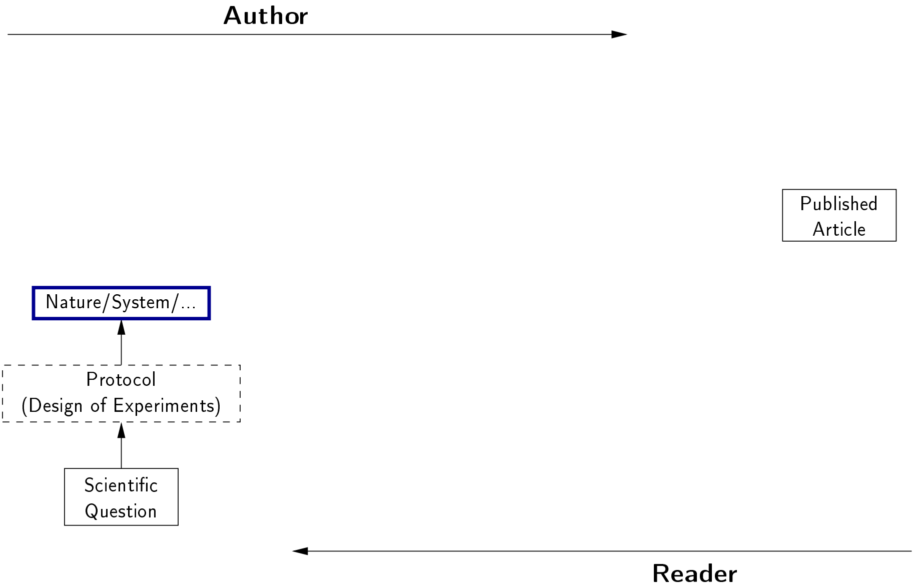
Experimenting in the Wild



Experiments in Distributed Systems: Even Worse!

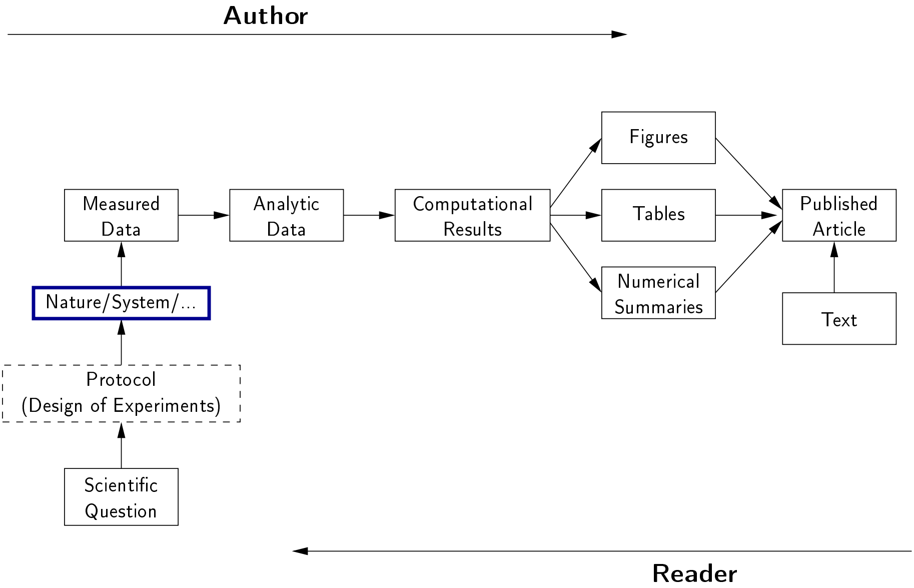
- Rely on large, distributed, hybrid, prototype hardware/software
- Measure execution times (makespans, traces, . . .)
- Many parameters, very costly and hard to *reproduce*

Reproducible Research: Trying to Bridge the Gap



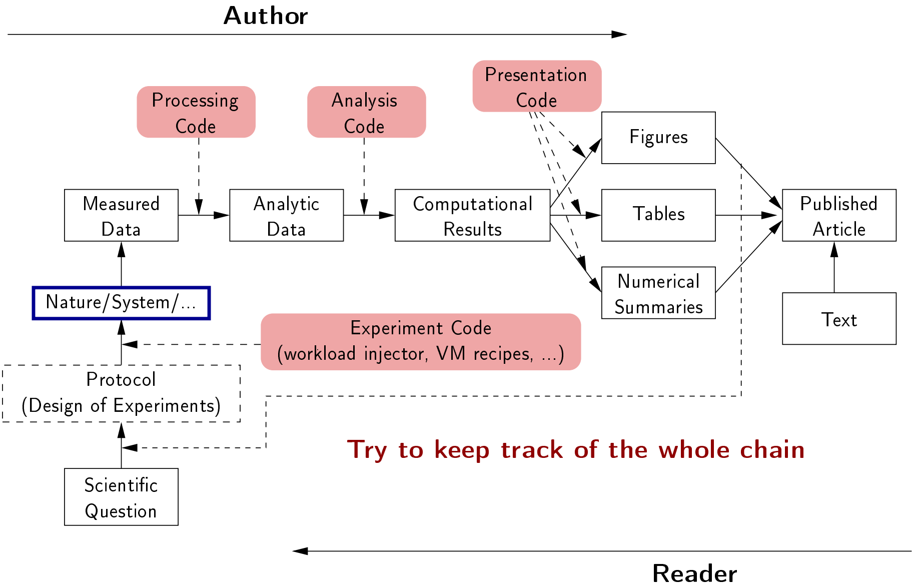
A. Legrand, Inspired by Roger D. Peng's lecture on reproducible research, May 2014

Reproducible Research: Trying to Bridge the Gap



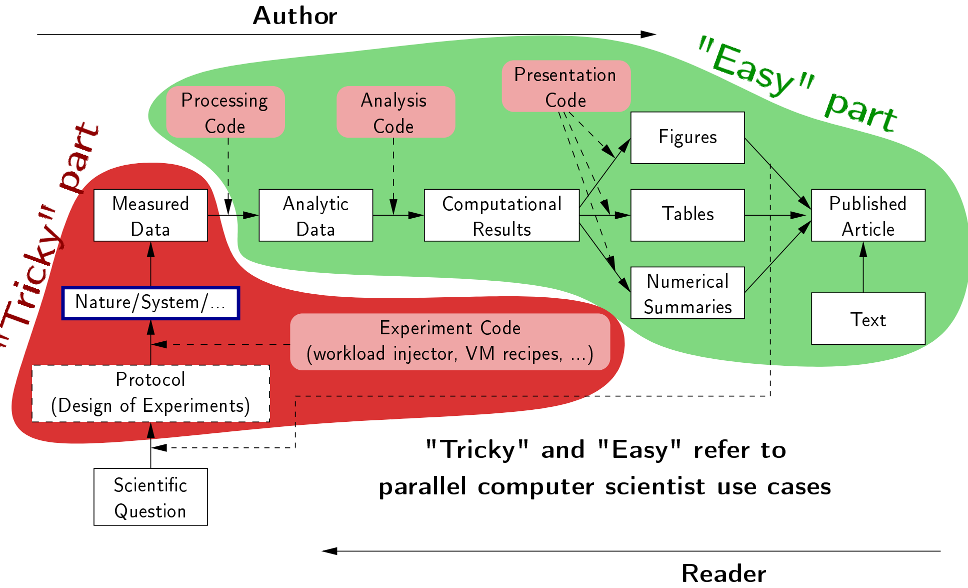
A. Legrand, Inspired by Roger D. Peng's lecture on reproducible research, May 2014

Reproducible Research: Trying to Bridge the Gap



A. Legrand, Inspired by Roger D. Peng's lecture on reproducible research, May 2014

Reproducible Research: Trying to Bridge the Gap



"Tricky" and "Easy" refer to parallel computer scientist use cases

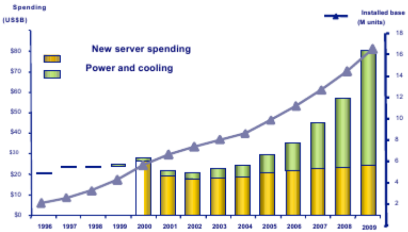
A. Legrand, Inspired by Roger D. Peng's lecture on reproducible research, May 2014

Agenda

- Introduction
- Computational Science of Computer Systems (CS^2)
- Simulation Models
- Models
- Open Science
- Energy
- Conclusion

Electric Power becomes THE problem

- IT industry dissipate 1% of world wide electric production
- 1Mw/h is 1M$ per year, and data centers dissipate hundreds of
- Microsoft's DataCenter in Chicago: 198Mw (Nuclear Power Plant: 1000-1500Mw)
- Power becomes more expensive than servers!



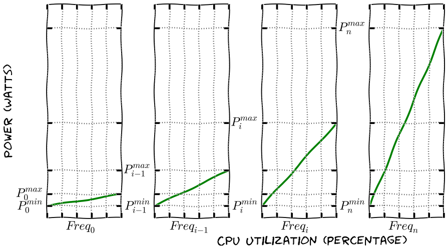
Can soon put more transistors on chip than can afford to turn on. — Patterson'07

we must model the Energy in Distributed Systems!

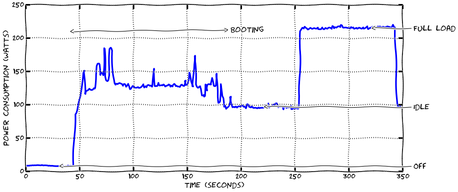
Energy in SimGrid

- Models for Speed vs. Power depending on the pstate
- Modeling of the On/Off power switches

DVFS Model



On/Off Models

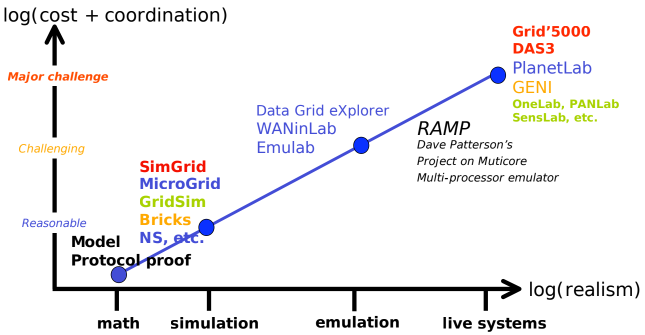


Agenda

- Introduction
- Computational Science of Computer Systems (CS^2)
- Simulation Models
- Models
- Open Science
- Energy
- Conclusion

Take Away Message

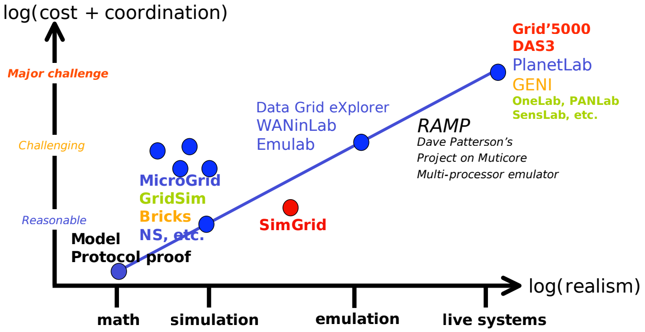
- Common Belief in 2008: Simulation as a toy methodology in CS



Courtesy of Franck Cappello (Gri5000 keynote @ EGEE, Feb 2008 :)

Take Away Message

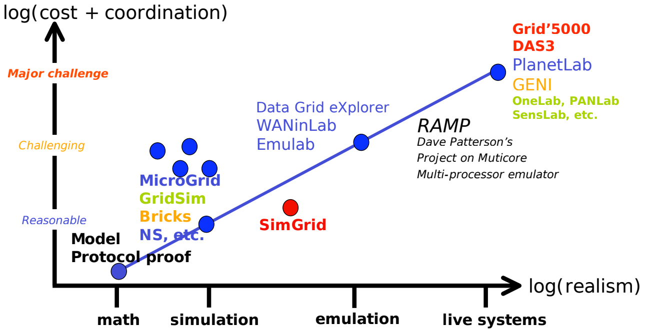
- Common Belief in 2008: Simulation as a toy methodology in CS
- Consensus in 2016: SimGrid as a scientific instrument (w/ Grid'5000)



Simulation turned into a reliable scientific instrument!

Take Away Message

- Common Belief in 2008: Simulation as a toy methodology in CS
- Consensus in 2016: SimGrid as a scientific instrument (w/ Grid'5000)



Simulation turned into a reliable scientific instrument!

- Consensus in 2025? We were naïve in 2015, but it works better now
- There is still a long way to go! **Now go, and do Good Science!**