Using Simulation for Studying Large Scale Distributed Systems

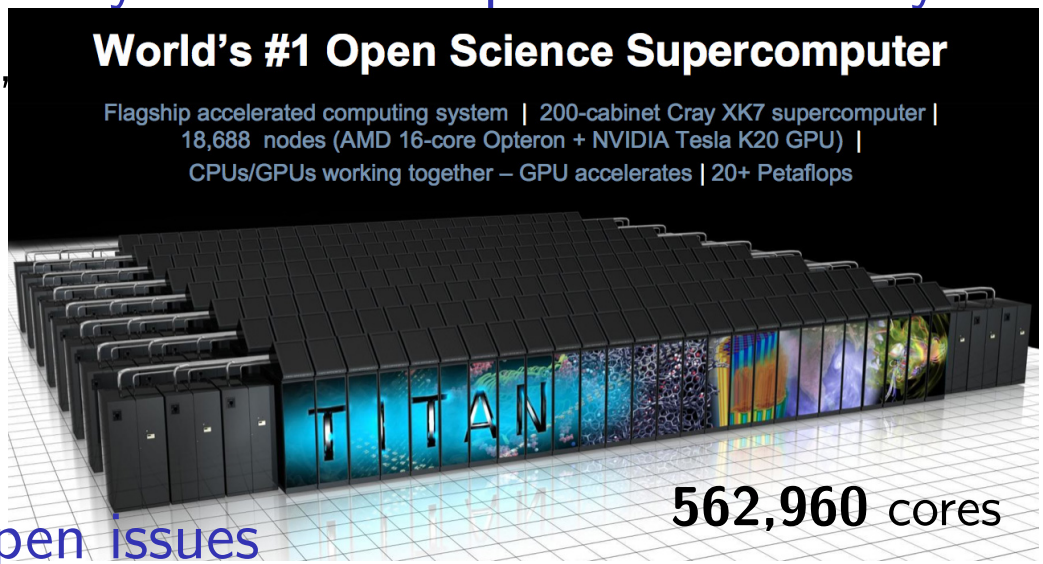
Arnaud Legrand *et Al.*

MOSIG, Performance Evaluation lecture
October 21st 2013

Large-Scale Distributed Systems Research

Large-scale parallel and distributed systems are in production today

- HPC (clusters, petascale systems, soon exascale...)
- Grid platforms
- Peer-to-peer file sharing
- Distributed volunteer computing
- Cloud Computing



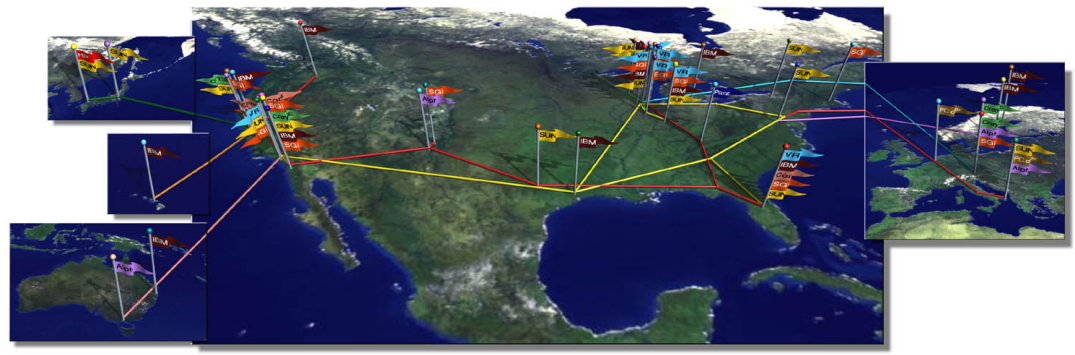
Complex platforms with many open issues

- resource discovery and monitoring
- resource & data management
- energy consumption reduction
- resource economics
- application scheduling
- fault-tolerance and availability
- scalability and performance
- decentralized algorithms

Large-Scale Distributed Systems Research

Large-scale parallel and distributed systems are in production today

- HPC (clusters, petascale systems, soon exascale...)
- Grid platforms
- Peer-to-peer file sharing
- Distributed volunteer computing
- Cloud Computing



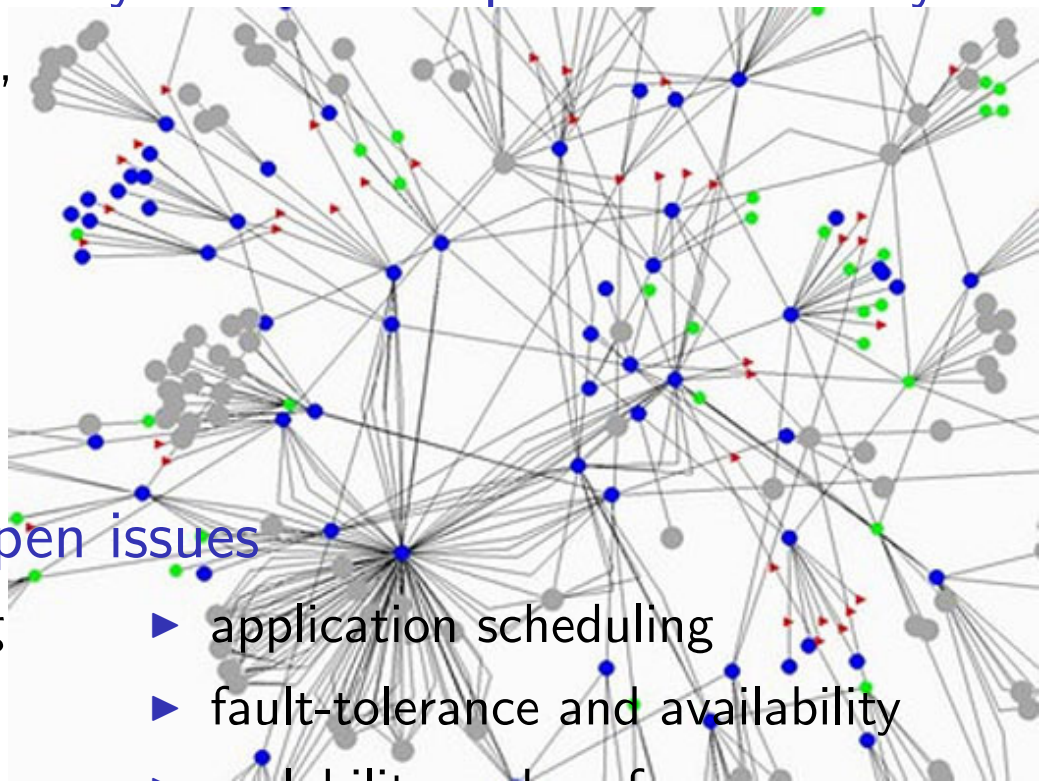
Complex platforms with many open issues

- resource discovery and monitoring
- resource & data management
- energy consumption reduction
- resource economics
- application scheduling
- fault-tolerance and availability
- scalability and performance
- decentralized algorithms

Large-Scale Distributed Systems Research

Large-scale parallel and distributed systems are in production today

- HPC (clusters, petascale systems, soon exascale...)
- Grid platforms
- Peer-to-peer file sharing
- Distributed volunteer computing
- Cloud Computing



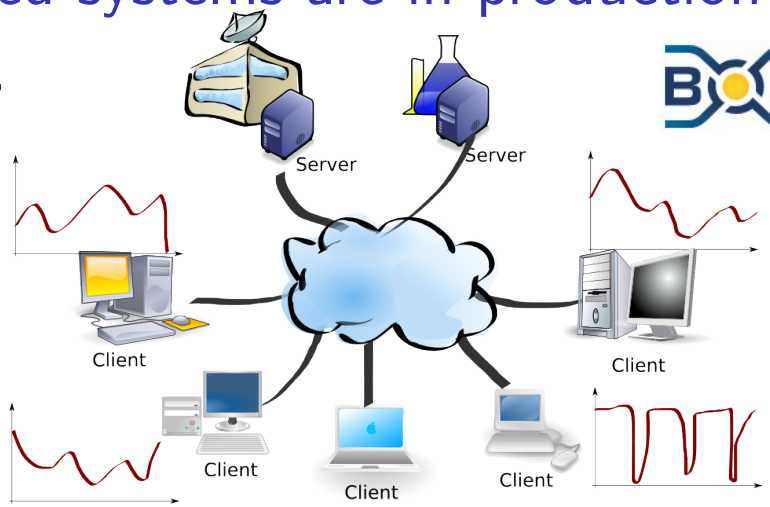
Complex platforms with many open issues

- resource discovery and monitoring
- resource & data management
- energy consumption reduction
- resource economics
- application scheduling
- fault-tolerance and availability
- scalability and performance
- decentralized algorithms

Large-Scale Distributed Systems Research

Large-scale parallel and distributed systems are in production today

- HPC (clusters, petascale systems, soon exascale...)
- Grid platforms
- Peer-to-peer file sharing
- Distributed volunteer computing
- Cloud Computing



540,000+ hosts
7.3+ PetaFLOPS

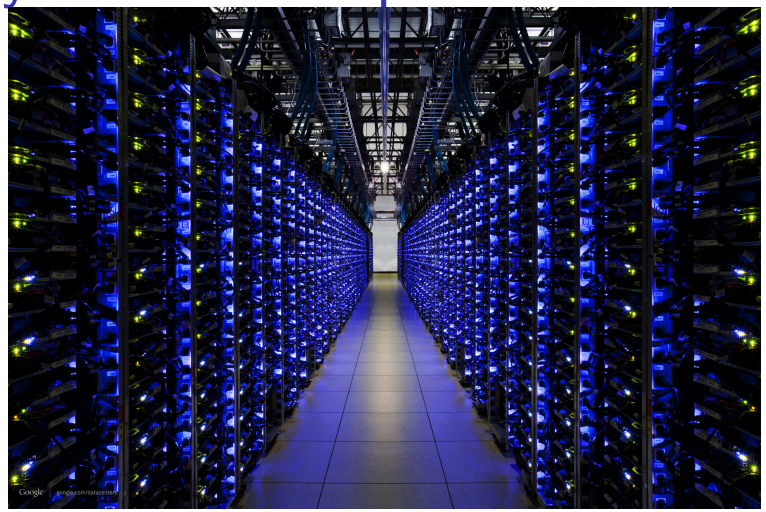
Complex platforms with many open issues

- resource discovery and monitoring
- resource & data management
- energy consumption reduction
- resource economics
- application scheduling
- fault-tolerance and availability
- scalability and performance
- decentralized algorithms

Large-Scale Distributed Systems Research

Large-scale parallel and distributed systems are in production today

- HPC (clusters, petascale systems, soon exascale...)
- Grid platforms
- Peer-to-peer file sharing
- Distributed volunteer computing
- Cloud Computing



Google Data Center

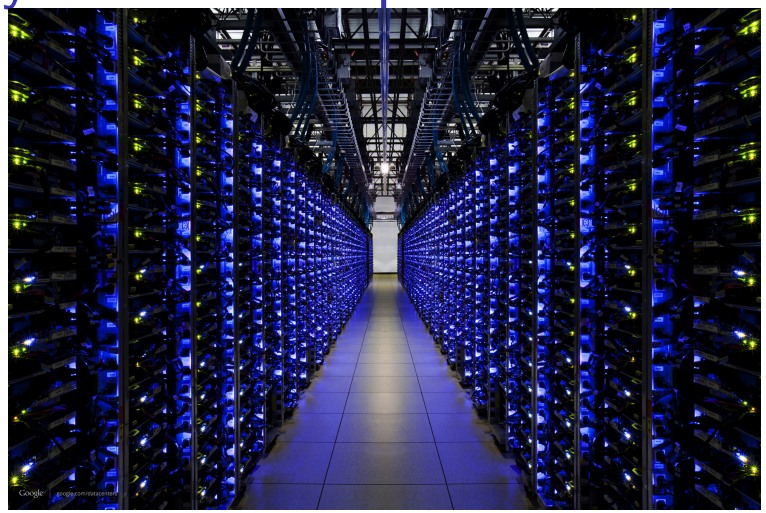
Complex platforms with many open issues

- resource discovery and monitoring
- resource & data management
- energy consumption reduction
- resource economics
- application scheduling
- fault-tolerance and availability
- scalability and performance
- decentralized algorithms

Large-Scale Distributed Systems Research

Large-scale parallel and distributed systems are in production today

- HPC (clusters, petascale systems, soon exascale...)
- Grid platforms
- Peer-to-peer file sharing
- Distributed volunteer computing
- Cloud Computing



Google Data Center

Complex platforms with many open issues

- resource discovery and monitoring
- resource & data management
- energy consumption reduction
- resource economics
- application scheduling
- fault-tolerance and availability
- scalability and performance
- decentralized algorithms

Such **applications** and **systems** deserve very advanced analysis

- Their debugging and tuning are technically difficult
- Their use induce high methodological challenges

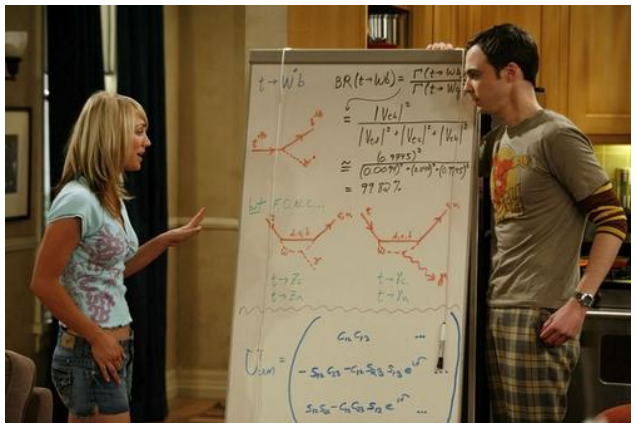
Methodological Approaches

Analytical works?

- Some purely mathematical models exist
- ☺ Allow better understanding of principles in spite of dubious applicability

  impossibility theorems, parameter influence, ...
- ☹ Theoretical results are difficult to achieve
  - Everyday practical issues (routing, scheduling) become NP-hard problems

    Most of the time, only heuristics whose performance have to be assessed are proposed
  - Models too simplistic, rely on ultimately unrealistic assumptions, fail to capture key characteristics of real systems



The Big Bang Theory

⇒ One must run experiments

⇝ Most published research in the area is experimental

- In vivo: Direct experimentation
- In vitro: Emulation
- In silico: Simulation

In vivo approach to HPC experiments (direct experiment)

☺ Eminently *believable* to demonstrate the proposed approach applicability.

In vivo approach to HPC experiments (direct experiment)

- ☺ Eminently *believable* to demonstrate the proposed approach applicability.
- ☹ Experiments can be too expensive, slow, dangerous

Large Hadron Collider

In vivo approach to HPC experiments (direct experiment)

- ☺ Eminently *believable* to demonstrate the proposed approach applicability.
- ☹ Experiments can be too expensive, slow, dangerous
- ☹ Very time and labor consuming
  - Entire application must be functional
- ☹ Choosing the right testbed is difficult
  - My own little testbed?
    - ☺ Well-behaved, controlled,stable ☹ Rarely representative of production platforms
  - Real production platforms?
    - Not everyone has access to them; CS experiments are disruptive for users
    - Experimental settings may change drastically during experiment (components fail; other users load resources; administrators change config.)
- ☹ Results remain limited to the testbed
  - Impact of testbed specificities hard to quantify ⇒ collection of testbeds...
  - Extrapolations and explorations of "what if" scenarios difficult (what if the network were different? what if we had a different workload?)
- ☹ Real experiments are often uncontrolled and unrepeatable

  No way to test alternatives back-to-back (even if disruption is part of the experiment)

Large Hadron Collider

In vivo approach to HPC experiments (direct experiment)

- ☺ Eminently *believable* to demonstrate the proposed approach applicability.
- ☹ Experiments can be too expensive, slow, dangerous
- ☹ Very time and labor consuming
  - Entire application must be functional
- ☹ Choosing the right testbed is difficult
  - My own little testbed?
    - ☺ Well-behaved, controlled,stable ☹ Rarely representative of production platforms
  - Real production platforms?
    - Not everyone has access to them; CS experiments are disruptive for users
    - Experimental settings may change drastically during experiment (components fail; other users load resources; administrators change config.)
- ☹ Results remain limited to the testbed
  - Impact of testbed specificities hard to quantify ⇒ collection of testbeds...
  - Extrapolations and explorations of "what if" scenarios difficult (what if the network were different? what if we had a different workload?)
- ☹ Real experiments are often uncontrolled and unrepeatable

  No way to test alternatives back-to-back (even if disruption is part of the experiment)

Large Hadron Collider

Difficult for others to reproduce results

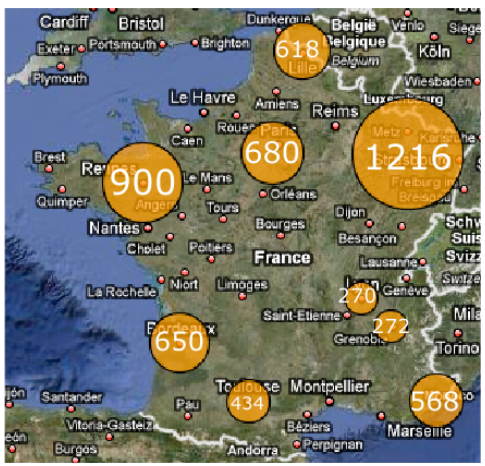
even if this is the basis for scientific advances!

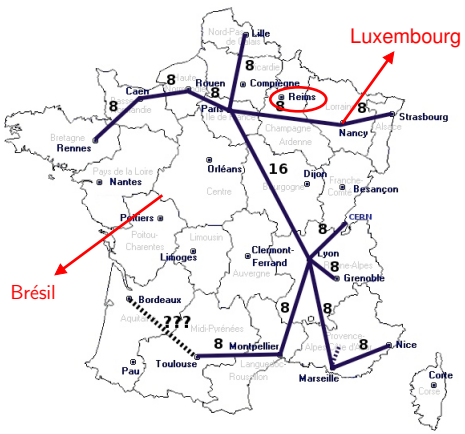
Example of Tools for Direct Experimentation

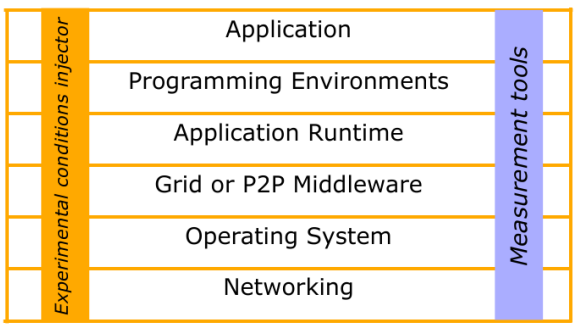
- Principle: Real applications, controlled environment
- Challenges: Hard and long. Experimental control? Reproducibility?

Grid'5000 project: a scientific instrument for the HPC

- Instrument for research in computer science (*deploy* your own OS)
- 9 sites, 1500 nodes (3000 cpus, 4000 cores); dedicated 10Gb links







Other existing platforms

- PlanetLab: No experimental control ⇒ no reproducibility
- Production Platforms (EGEE): must use provided middleware
- FutureGrid: future US experimental platform loosely inspired from Grid'5000

Emulation (in vitro) 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 (in vitro) 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

Emulation (in vitro) 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

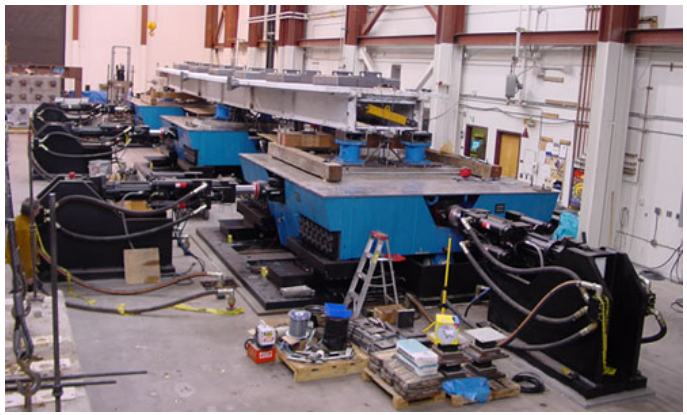
in a dry country

Why don't you just control the climate?

or tweak the car's reality?

Emulation in other Sciences

Studying earthquake effects on bridges



Studying tsunamis

Studying Coriolis effect and stratification vs. viscosity

Studying climate change effects on ecosystems

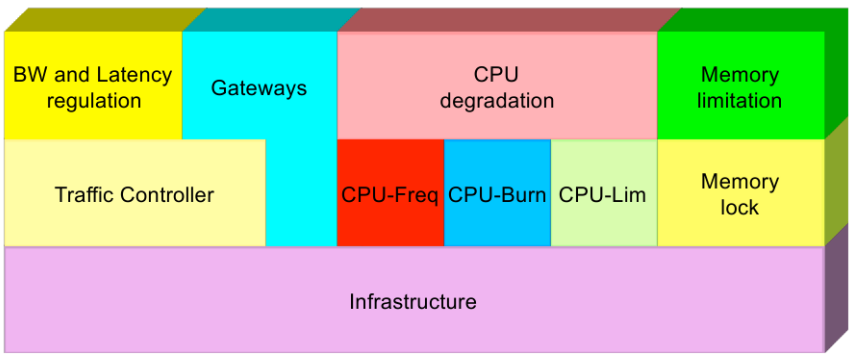
(who said that science is not fun??)

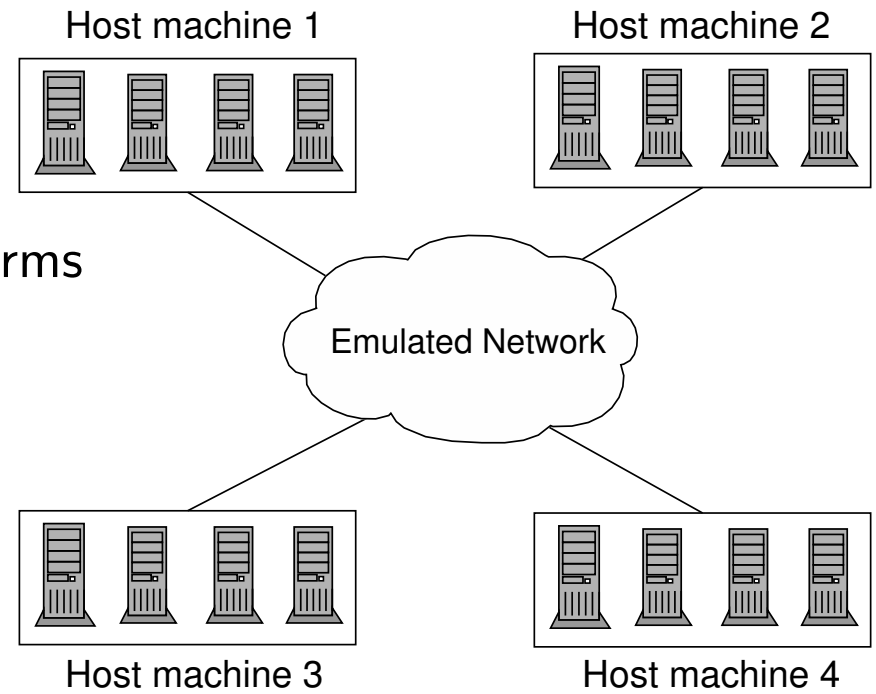
In vitro approach to HPC experiments (emulation)

- Principle: Injecting load on real systems for the experimental control
  $\approx$ *Slow platform down to put it in wanted experimental conditions*
- Challenges: Get realistic results, tool stack complex to deploy and use, control often induces bias

Wrekavoc: applicative emulator

- Emulates CPU and network
- Homogeneous or heterogeneous platforms





Nodes Virtualization

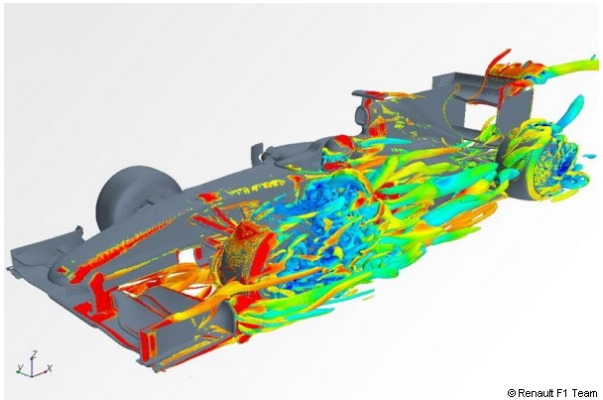
Other existing tools

- Network emulation: ModelNet, DummyNet, ...
  Tools rather mature, but limited to network
- Applicative emulation: MicroGrid, eWan, Emulab
  Rarely (never?) used outside the lab where they were created

In silico approach to HPC experiments (simulation)

Simulation solves some difficulties raised by *in vivo* experiments

- No need to build a real system, nor the full-fledged application
- Conduct controlled and repeatable experiments
- (Almost) no limits to experimental scenarios
- Possible for anybody to reproduce results



Car Mesh

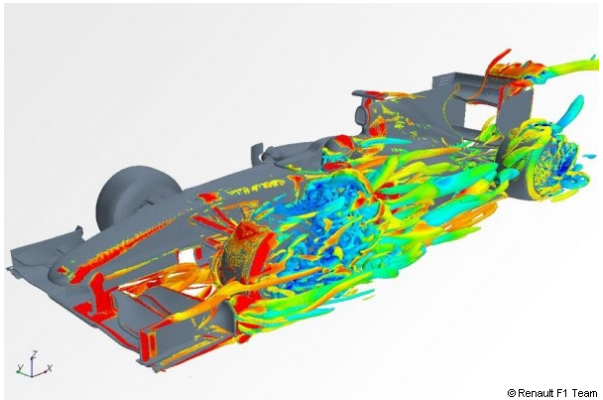
Simulation in a nutshell

Computer prediction of the behavior of a system using a (approximate) model

In silico approach to HPC experiments (simulation)

Simulation solves some difficulties raised by *in vivo* experiments

- No need to build a real system, nor the full-fledged application
- Conduct controlled and repeatable experiments
- (Almost) no limits to experimental scenarios
- Possible for anybody to reproduce results



Car Mesh

Simulation in a nutshell

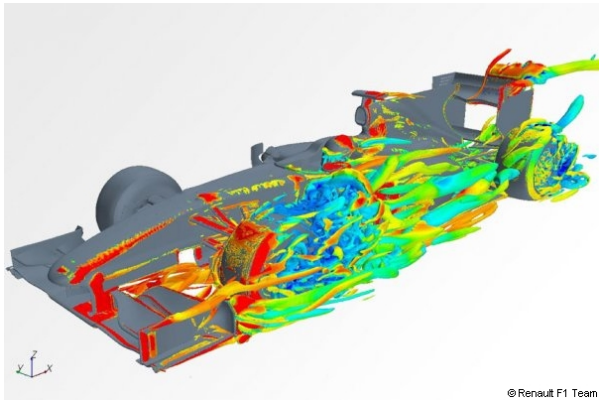
Computer prediction of the behavior of a system using a (approximate) model

- Model: Set of equations; Objects whose state evolution is governed by a set of rules; ...

In silico approach to HPC experiments (simulation)

Simulation solves some difficulties raised by *in vivo* experiments

- No need to build a real system, nor the full-fledged application
- Conduct controlled and repeatable experiments
- (Almost) no limits to experimental scenarios
- Possible for anybody to reproduce results



Car Mesh

Simulation in a nutshell

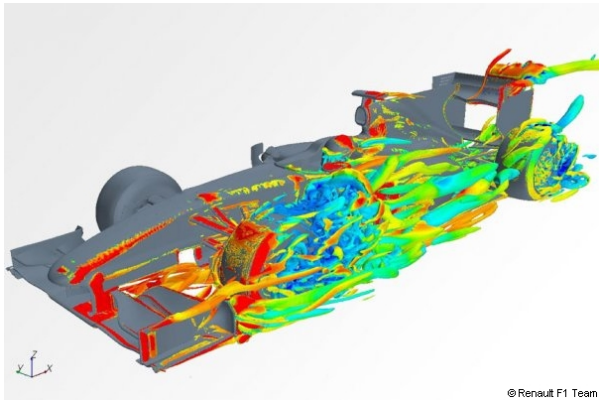
Computer prediction of the behavior of a system using a (approximate) model

- Model: Set of equations; Objects whose state evolution is governed by a set of rules; ...
- Simulator: Program solving equations or computing the evolution according to the rules

In silico approach to HPC experiments (simulation)

Simulation solves some difficulties raised by *in vivo* experiments

- No need to build a real system, nor the full-fledged application
- Conduct controlled and repeatable experiments
- (Almost) no limits to experimental scenarios
- Possible for anybody to reproduce results



Car Mesh

Simulation in a nutshell

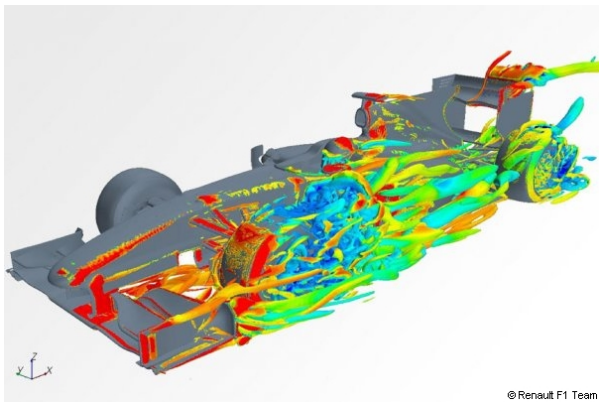
Computer prediction of the behavior of a system using a (approximate) model

- Model: Set of equations; Objects whose state evolution is governed by a set of rules; ...
- Simulator: Program solving equations or computing the evolution according to the rules
- Wanted features:
  - Accuracy: Correspondence between simulation and real-world

In silico approach to HPC experiments (simulation)

Simulation solves some difficulties raised by *in vivo* experiments

- No need to build a real system, nor the full-fledged application
- Conduct controlled and repeatable experiments
- (Almost) no limits to experimental scenarios
- Possible for anybody to reproduce results



Car Mesh

Simulation in a nutshell

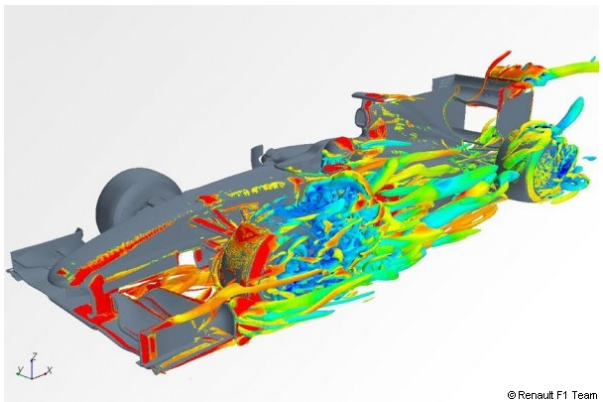
Computer prediction of the behavior of a system using a (approximate) model

- Model: Set of equations; Objects whose state evolution is governed by a set of rules; ...
- Simulator: Program solving equations or computing the evolution according to the rules
- Wanted features:
  - Accuracy: Correspondence between simulation and real-world
  - Scalability: Actually usable by computers (fast enough)

In silico approach to HPC experiments (simulation)

Simulation solves some difficulties raised by *in vivo* experiments

- No need to build a real system, nor the full-fledged application
- Conduct controlled and repeatable experiments
- (Almost) no limits to experimental scenarios
- Possible for anybody to reproduce results



Car Mesh

Simulation in a nutshell

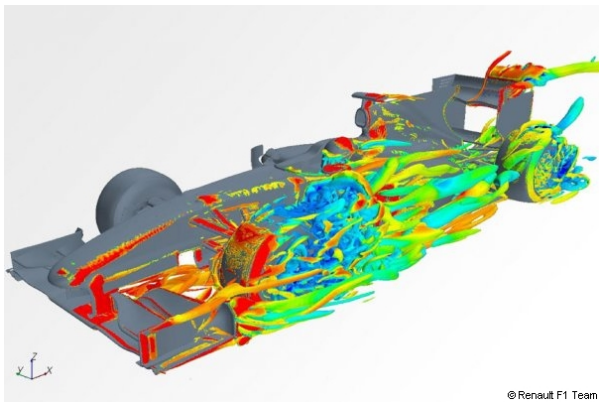
Computer prediction of the behavior of a system using a (approximate) model

- Model: Set of equations; Objects whose state evolution is governed by a set of rules; ...
- Simulator: Program solving equations or computing the evolution according to the rules
- Wanted features:
  - Accuracy: Correspondence between simulation and real-world
  - Scalability: Actually usable by computers (fast enough)
  - Tractability: Actually usable by human beings (simple enough to understand)

In silico approach to HPC experiments (simulation)

Simulation solves some difficulties raised by *in vivo* experiments

- No need to build a real system, nor the full-fledged application
- Conduct controlled and repeatable experiments
- (Almost) no limits to experimental scenarios
- Possible for anybody to reproduce results



Car Mesh

Simulation in a nutshell

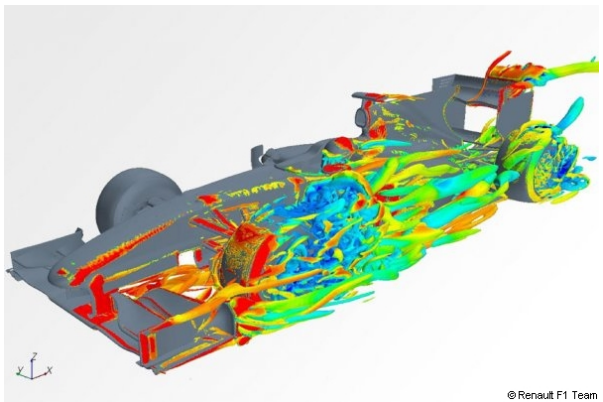
Computer prediction of the behavior of a system using a (approximate) model

- Model: Set of equations; Objects whose state evolution is governed by a set of rules; ...
- Simulator: Program solving equations or computing the evolution according to the rules
- Wanted features:
  - Accuracy: Correspondence between simulation and real-world
  - Scalability: Actually usable by computers (fast enough)
  - Tractability: Actually usable by human beings (simple enough to understand)
  - Instanciability: Can actually describe real settings (no magical parameter)

In silico approach to HPC experiments (simulation)

Simulation solves some difficulties raised by *in vivo* experiments

- No need to build a real system, nor the full-fledged application
- Conduct controlled and repeatable experiments
- (Almost) no limits to experimental scenarios
- Possible for anybody to reproduce results



Car Mesh

Simulation in a nutshell

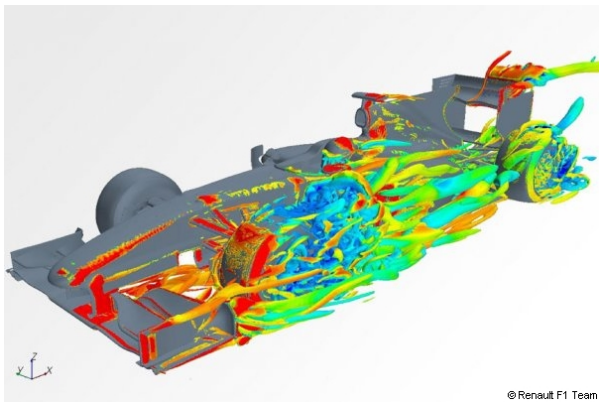
Computer prediction of the behavior of a system using a (approximate) model

- Model: Set of equations; Objects whose state evolution is governed by a set of rules; ...
- Simulator: Program solving equations or computing the evolution according to the rules
- Wanted features:
  - Accuracy: Correspondence between simulation and real-world
  - Scalability: Actually usable by computers (fast enough)
  - Tractability: Actually usable by human beings (simple enough to understand)
  - Instanciability: Can actually describe real settings (no magical parameter)
  - Relevance: Captures object of interest

In silico approach to HPC experiments (simulation)

Simulation solves some difficulties raised by *in vivo* experiments

- No need to build a real system, nor the full-fledged application
- Conduct controlled and repeatable experiments
- (Almost) no limits to experimental scenarios
- Possible for anybody to reproduce results



Car Mesh

Simulation in a nutshell

Computer prediction of the behavior of a system using a (approximate) model

- Model: Set of equations; Objects whose state evolution is governed by a set of rules; ...
- Simulator: Program solving equations or computing the evolution according to the rules
- Wanted features:
  - Accuracy: Correspondence between simulation and real-world
  - Scalability: Actually usable by computers (fast enough)
  - Tractability: Actually usable by human beings (simple enough to understand)
  - Instanciability: Can actually describe real settings (no magical parameter)
  - Relevance: Captures object of interest

Versatility

Simulation in Computer Science

Microprocessor Design

- A few standard "cycle-accurate" simulators are used extensively
  `http://www.cs.wisc.edu/~arch/www/tools.html`

⇒ Possible to reproduce simulation results

- You can read a paper,
- reproduce a subset of its results,
- improve

Workshop on Duplicating, Deconstructing, and Debunking

Networking

- A few established "packet-level" simulators: NS-2, DaSSF, OMNeT++, GTNetS
- Well-known datasets for network topologies
- Well-known generators of synthetic topologies
- SSF standard: `http://www.ssfnet.org/`

⇒ Possible to reproduce simulation results

Simulation in Distributed Systems Research

Little common methodologies and tools

- Experimental settings rarely detailed enough in literature
- No established simulator up until a few years ago
- Simulators are short-lived and rarely made available
- Most people build their own "ad-hoc" solutions

> Naicken, Stephen *et Al.*, *Towards Yet Another Peer-to-Peer Simulator*, HET-NETs'06.

From 141 P2P sim.papers, 30% use a custom tool, 50% don't report used tool

Simulation in Distributed Systems Research

Little common methodologies and tools

- Experimental settings rarely detailed enough in literature
- No established simulator up until a few years ago
- Simulators are short-lived and rarely made available
- Most people build their own "ad-hoc" solutions

> Naicken, Stephen *et Al.*, *Towards Yet Another Peer-to-Peer Simulator*, HET-NETs'06.

From 141 P2P sim.papers, 30% use a custom tool, 50% don't report used tool

Why?

- Understanding and controlling the simulator code is important.
- Researchers lack trust in a simulator developed by others...

Simulation in Distributed Systems Research

Little common methodologies and tools

- Experimental settings rarely detailed enough in literature
- No established simulator up until a few years ago
- Simulators are short-lived and rarely made available
- Most people build their own "ad-hoc" solutions

> Naicken, Stephen *et Al.*, *Towards Yet Another Peer-to-Peer Simulator*, HET-NETs'06.

From 141 P2P sim.papers, 30% use a custom tool, 50% don't report used tool

Why?

- Understanding and controlling the simulator code is important.
- Researchers lack trust in a simulator developed by others...
- ...or researchers don't care. All they want is a paper.

Consequence

Most published simulation results are impossible to reproduce by researchers other than their authors

Yet, simulation results should be easily repeatable by design!

The Specialization Excuse

But again... Why ?

- Most simulators are are domain-specific (P2P, HPC, grid, cloud, . . .).

One simulator to rule them all?

- Although many simulators claim to be generic, they were developed with a specific purpose in mind and can hardly be used beyond their initial purpose.
- Hence, simulators are developed by researchers for their own research field and these researchers are domain experts, not simulation experts.

The Specialization Excuse

But again... Why ?

- Most simulators are are domain-specific (P2P, HPC, grid, cloud, . . .).

One simulator to rule them all?

- Although many simulators claim to be generic, they were developed with a specific purpose in mind and can hardly be used beyond their initial purpose.
- Hence, simulators are developed by researchers for their own research field and these researchers are domain experts, not simulation experts.

Popular Wisdom 1

Simulators are toys that any MSc. C.S. student can write. ☺

Popular Wisdom 2

Specialization *allows for "better" simulation, i.e., simulations that achieve a desired trade-off between* **accuracy** *(low simulation error) and* **scalability** *(ability to run big and/or fast simulations).*

The SimGrid Project

SimGrid: a generic simulation framework for distributed applications

- 13 years old open-source project. Collaboration between
  - France (INRIA, CNRS, Univ. Lyon, Nancy, Grenoble, ...)
  - USA (UCSD, U. Hawaii), ...
- Started like others (unsatisfied with practice, no simulation specialists):

  Wouldn't it be possible to have both **accuracy**, **scalability** and **versatility**?
- Scalable (time and memory), modular, portable. +140 publications.

Other existing tools

- *Large* amount of existing simulator for distributed platforms:
  GridSim, ChicSim, OptorSim, GES; P2PSim, PlanetSim, PeerSim, CloudSim.
- Few are really usable: Diffusion, Software Quality Assurance, Long-term availability
- No other study the validity, the induced experimental bias

Purpose of this talk

- Present some efforts and results obtained in the SimGrid project related to improving accuracy, scalability and versatility.
- Explain how it compares to other domain-specific simulators.

Agenda

- Experiments for Large-Scale Distributed Systems Research
  - Main Methodological Approaches: In Vivo, In Silico, In Vitro
  - Bad Practices in Large-Scale Distributed Systems Research
- The SimGrid Project
  - How accurate? The Validation Quest
- Conclusions
  - Keynote Recap
  - Going Further: Experiment planning and Open Science

Outline

Simulation Validation: the FLASH example

FLASH project at Stanford

- Building large-scale shared-memory multiprocessors
- Went from conception, to design, to actual hardware (32-node)
- Used simulation heavily over 6 years

Authors compared simulation(s) to the real world

- Error is unavoidable (30% error in their case was not rare)

  Negating the impact of "we got 1.5% improvement"
- Complex simulators not ensuring better simulation results
  - Simple simulators worked better than sophisticated ones (which were unstable)
  - Simple simulators predicted trends as well as slower, sophisticated ones

  $\Rightarrow$ Should focus on simulating the important things
- Calibrating simulators on real-world settings is mandatory
- For FLASH, the simple simulator was all that was needed: Realistic $\approx$ Credible

Gibson, Kunz, Ofelt, Heinrich, *FLASH vs. (Simulated) FLASH: Closing the Simulation Loop*, Architectural Support for Programming Languages and Operating Systems, 2000

Along the same lines: Weaver and MsKee, *Are Cycle Accurate Simulations a Waste of Time?*, Proc. of the Workshop on Duplicating, Deconstruction and Debunking, 2008

Network Communication Models

Packet-level simulation Networking community has standards, many popular open-source projects (NS, GTneTS, OmNet++,...)

- full simulation of the whole protocol stack
- complex models $\rightsquigarrow$ hard to instantiate

> Flores Lucio, Paredes-Farrera, Jammeh, Fleury, Reed. *Opnet modeler and ns-2: Comparing the accuracy of network simulators for packet-level analysis using a network testbed.* WSEAS Transactions on Computers 2, no. 3 (2003)

- inherently **slow**
- beware of simplistic packet-level simulation

Network Communication Models

Packet-level simulation Networking community has standards, many popular open-source projects (NS, GTneTS, OmNet++,...)

- full simulation of the whole protocol stack
- complex models $\rightsquigarrow$ hard to instantiate

> Flores Lucio, Paredes-Farrera, Jammeh, Fleury, Reed. *Opnet modeler and ns-2: Comparing the accuracy of network simulators for packet-level analysis using a network testbed.* WSEAS Transactions on Computers 2, no. 3 (2003)

- inherently **slow**
- beware of simplistic packet-level simulation

Delay-based models The simplest ones...

- communication time = constant delay, statistical distribution, LogP
  $\rightsquigarrow$($\Theta(1)$ footprint and $O(1)$ computation)
- coordinate based systems to account for geographic proximity
  $\rightsquigarrow$($\Theta(N)$ footprint and $O(1)$ computation)

Although very scalable, these models ignore network congestion and typically assume large bisection bandwidth

Network Communication Models (cont'd)

Flow-level models

A communication is simulated as a single entity (like a flow in pipes):

$$T_{i,j}(S) = L_{i,j} + S/B_{i,j}, \text{ where } \begin{cases} S & \text{message size} \\ L_{i,j} & \text{latency between } i \text{ and } j \\ B_{i,j} & \text{bandwidth between } i \text{ and } j \end{cases}$$

Estimating $B_{i,j}$ requires to account for interactions with other flows

Network Communication Models (cont'd)

Flow-level models

A communication is simulated as a single entity (like a flow in pipes):

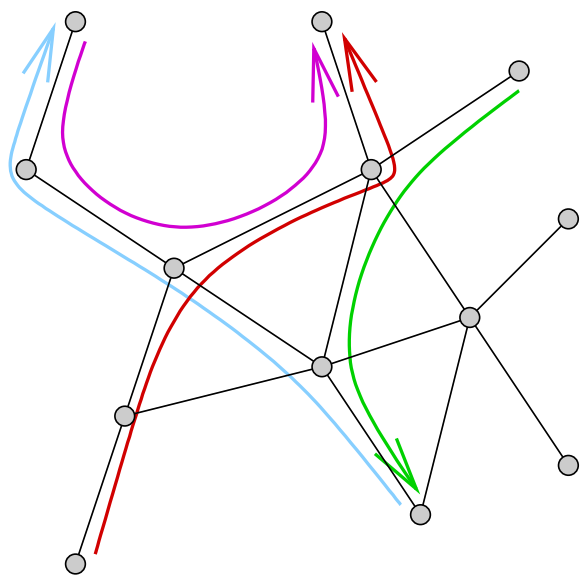
$$T_{i,j}(S) = L_{i,j} + S/B_{i,j}, \text{ where } \begin{cases} S & \text{message size} \\ L_{i,j} & \text{latency between } i \text{ and } j \\ B_{i,j} & \text{bandwidth between } i \text{ and } j \end{cases}$$

Estimating $B_{i,j}$ requires to account for interactions with other flows

Assume steady-state and **share bandwidth** every time a new flow appears or disappears

Setting a set of flows $\mathcal{F}$ and a set of links $\mathcal{L}$

Constraints For all link j: $\sum_{\text{if flow i uses link j}} \varrho_i \leqslant C_j$



Network Communication Models (cont'd)

Flow-level models

A communication is simulated as a single entity (like a flow in pipes):

$$T_{i,j}(S) = L_{i,j} + S/B_{i,j}, \text{ where } \begin{cases} S & \text{message size} \\ L_{i,j} & \text{latency between } i \text{ and } j \\ B_{i,j} & \text{bandwidth between } i \text{ and } j \end{cases}$$

Estimating $B_{i,j}$ requires to account for interactions with other flows

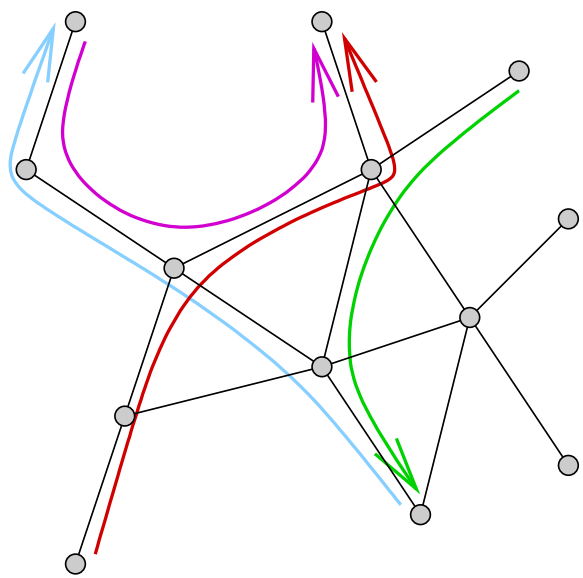
Assume steady-state and **share bandwidth** every time a new flow appears or disappears

Setting a set of flows $\mathcal{F}$ and a set of links $\mathcal{L}$

Constraints For all link j: $\sum_{\text{if flow i uses link j}} \varrho_i \leqslant C_j$

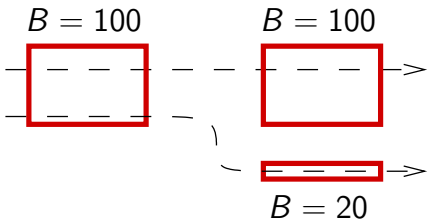
Objective function

- Max-Min $\max(\min(\varrho_i))$
- or other fancy objectives
  e.g., Reno $\sim \max(\sum \arctan(\varrho_i))$
  Vegas $\sim \max(\sum \log(\varrho_i))$



Invalidating Simulators from the Litterature

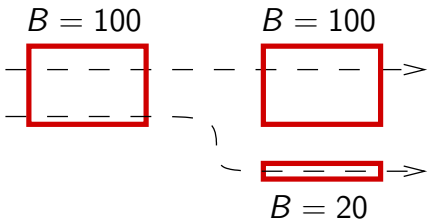
Naive flow models documented as wrong

Setting	Expected Output	Output
		

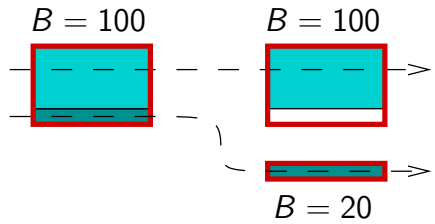
Invalidating Simulators from the Litterature

Naive flow models documented as wrong

Setting



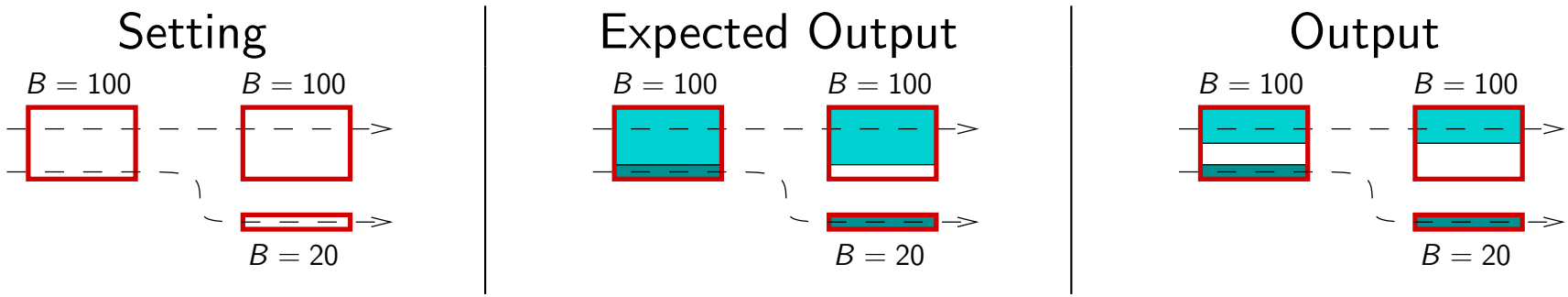
Expected Output



Output

Invalidating Simulators from the Litterature

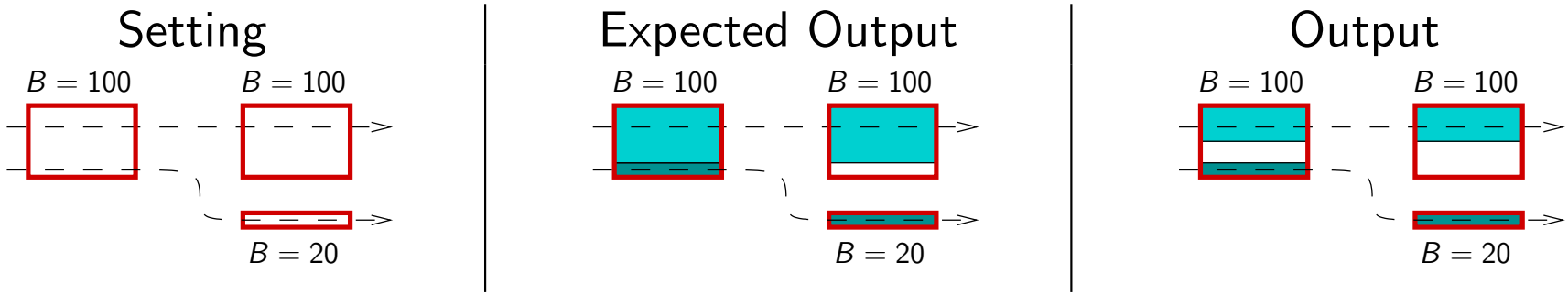
Naive flow models documented as wrong



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

Invalidating Simulators from the Litterature

Naive flow models documented as wrong

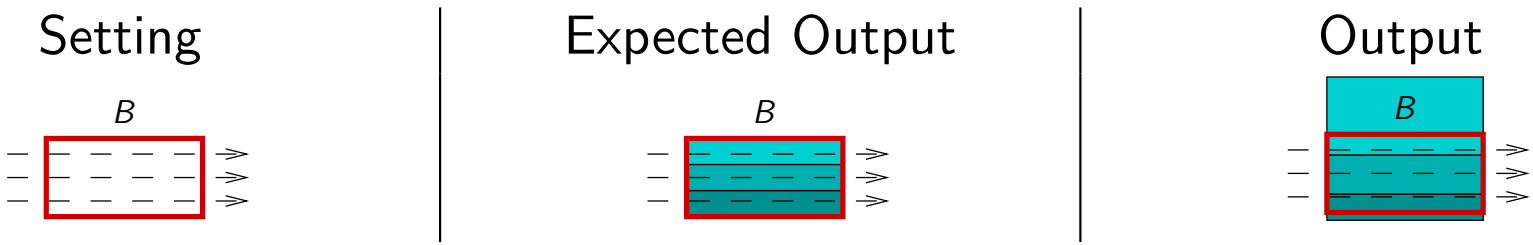


Known issue in Narses (2002), OptorSim (2003), GroudSim (2011).

Validation by general agreement

"Since SimJava and GridSim have been extensively utilized 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 (GridSim 5.2, Nov. 25, 2010). Similar issues with naive packet-level models.

Validation vs. Invalidation

Validation

- Articles full of "convincing" graphs but shallow description, unavailable or broken code
- Optimistic validation, i.e., only for a few cases in which the model is expected to work well

  $\rightsquigarrow$ merely verifies that the model implementation is correct and that its results are not completely unreasonable

Invalidation and crucial experiments

Other sciences assess the quality of a model by trying to invalidate it (Popper).

Validation vs. Invalidation

Validation

- Articles full of "convincing" graphs but shallow description, unavailable or broken code
- Optimistic validation, i.e., only for a few cases in which the model is expected to work well

  $\rightsquigarrow$ merely verifies that the model implementation is correct and that its results are not completely unreasonable

Invalidation and crucial experiments

Other sciences assess the quality of a model by trying to invalidate it (Popper).

Validation vs. Invalidation

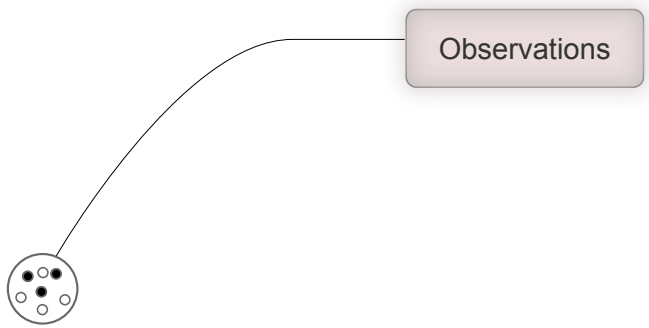
Validation

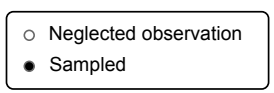
- Articles full of "convincing" graphs but shallow description, unavailable or broken code
- Optimistic validation, i.e., only for a few cases in which the model is expected to work well

  $\rightsquigarrow$ merely verifies that the model implementation is correct and that its results are not completely unreasonable

Invalidation and crucial experiments

Other sciences assess the quality of a model by trying to invalidate it (Popper).





Validation vs. Invalidation

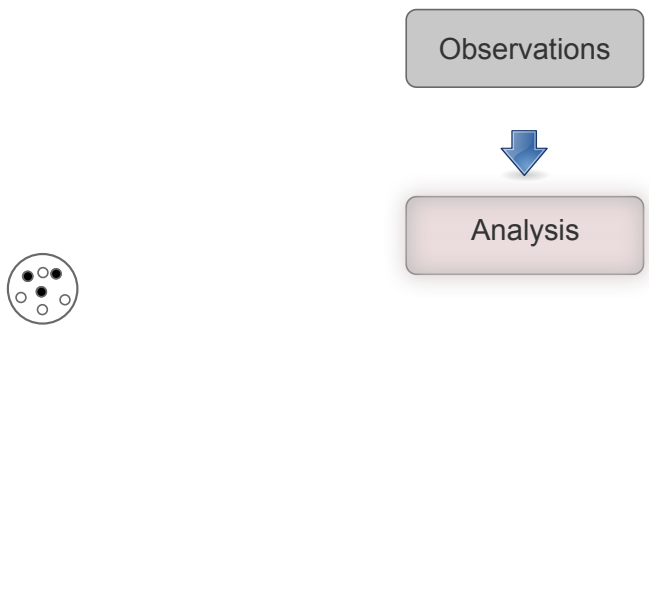
Validation

- Articles full of "convincing" graphs but shallow description, unavailable or broken code
- Optimistic validation, i.e., only for a few cases in which the model is expected to work well

  $\rightsquigarrow$ merely verifies that the model implementation is correct and that its results are not completely unreasonable

Invalidation and crucial experiments

Other sciences assess the quality of a model by trying to invalidate it (Popper).



Validation vs. Invalidation

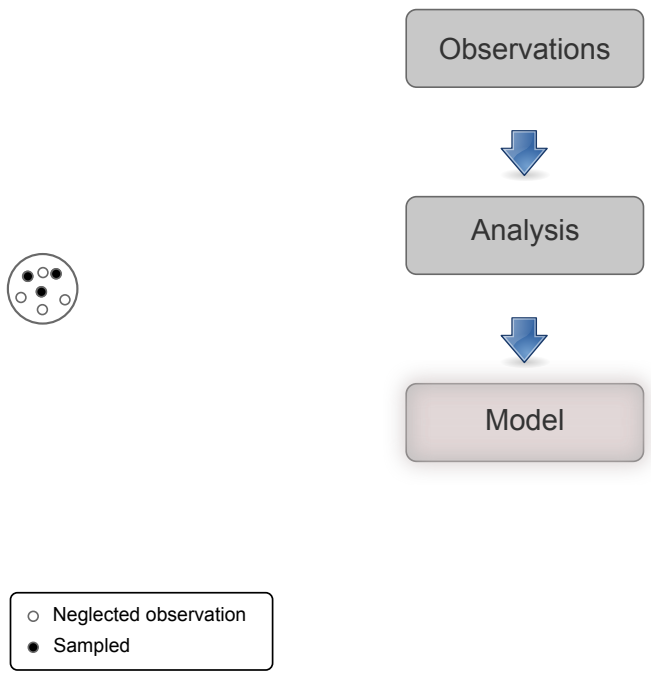
Validation

- Articles full of "convincing" graphs but shallow description, unavailable or broken code
- Optimistic validation, i.e., only for a few cases in which the model is expected to work well

  $\rightsquigarrow$ merely verifies that the model implementation is correct and that its results are not completely unreasonable

Invalidation and crucial experiments

Other sciences assess the quality of a model by trying to invalidate it (Popper).



Validation vs. Invalidation

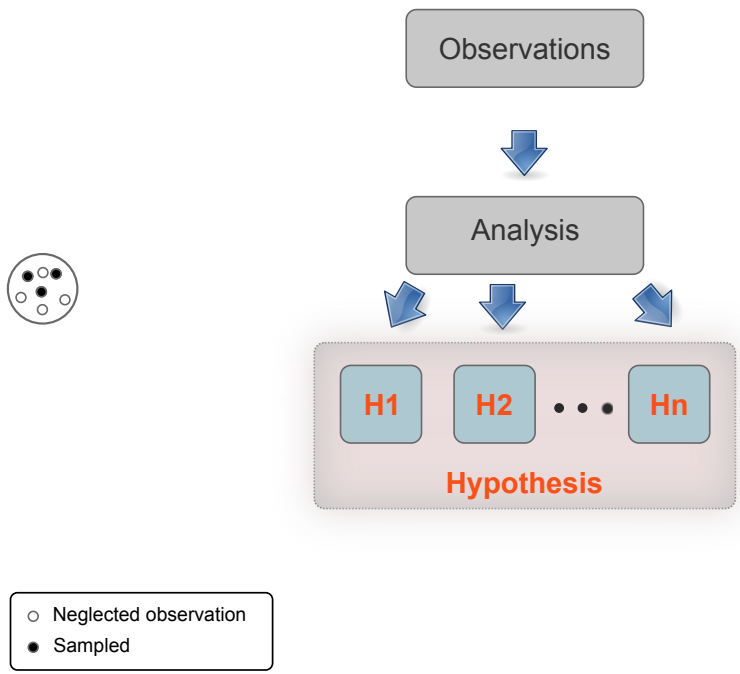
Validation

- Articles full of "convincing" graphs but shallow description, unavailable or broken code
- Optimistic validation, i.e., only for a few cases in which the model is expected to work well

  $\rightsquigarrow$ merely verifies that the model implementation is correct and that its results are not completely unreasonable

Invalidation and crucial experiments

Other sciences assess the quality of a model by trying to invalidate it (Popper).



Validation vs. Invalidation

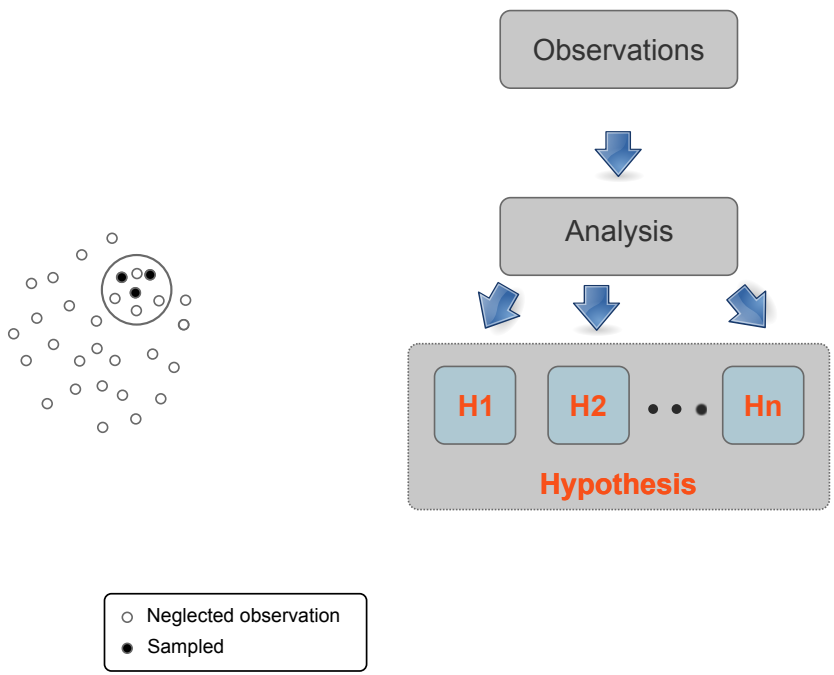
Validation

- Articles full of "convincing" graphs but shallow description, unavailable or broken code
- Optimistic validation, i.e., only for a few cases in which the model is expected to work well

  $\rightsquigarrow$ merely verifies that the model implementation is correct and that its results are not completely unreasonable

Invalidation and crucial experiments

Other sciences assess the quality of a model by trying to invalidate it (Popper).



Validation vs. Invalidation

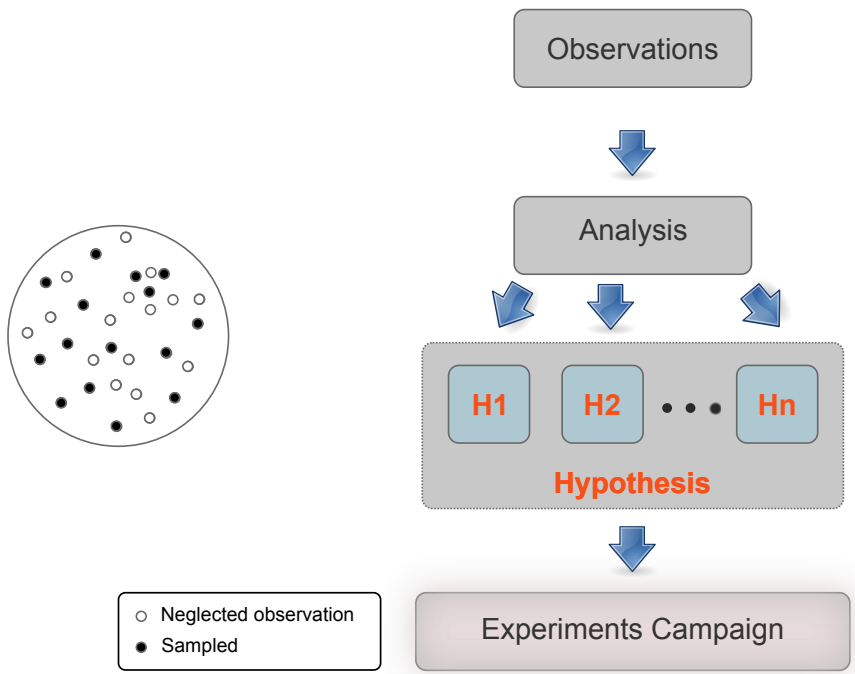
Validation

- Articles full of "convincing" graphs but shallow description, unavailable or broken code
- Optimistic validation, i.e., only for a few cases in which the model is expected to work well

  $\rightsquigarrow$ merely verifies that the model implementation is correct and that its results are not completely unreasonable

Invalidation and crucial experiments

Other sciences assess the quality of a model by trying to invalidate it (Popper).



Validation vs. Invalidation

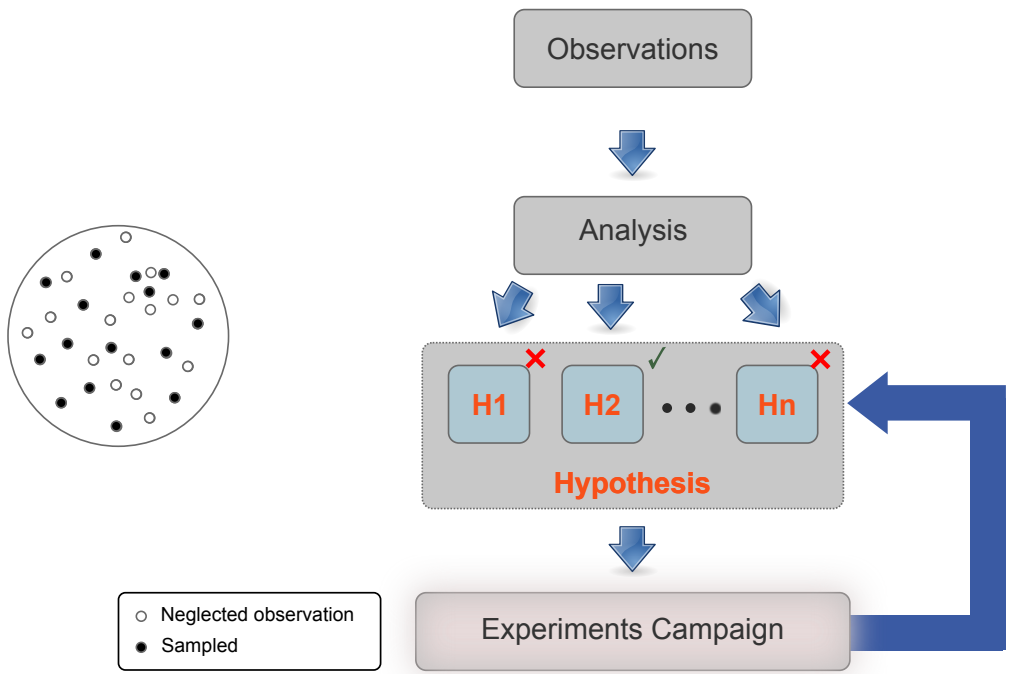
Validation

- Articles full of "convincing" graphs but shallow description, unavailable or broken code
- Optimistic validation, i.e., only for a few cases in which the model is expected to work well

  $\rightsquigarrow$ merely verifies that the model implementation is correct and that its results are not completely unreasonable

Invalidation and crucial experiments

Other sciences assess the quality of a model by trying to invalidate it (Popper).



Validation vs. Invalidation

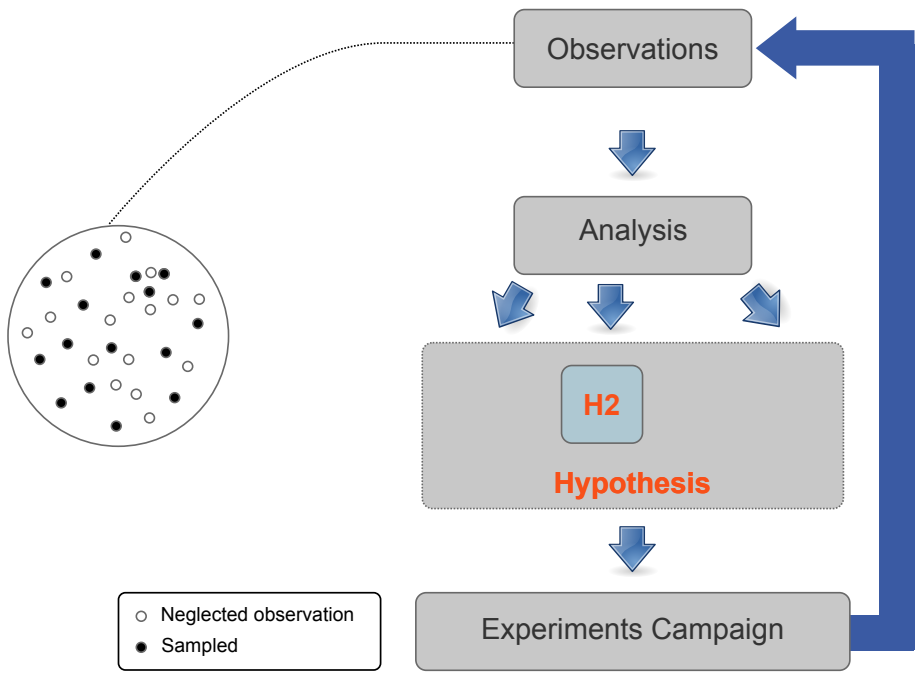
Validation

- Articles full of "convincing" graphs but shallow description, unavailable or broken code
- Optimistic validation, i.e., only for a few cases in which the model is expected to work well

  $\rightsquigarrow$ merely verifies that the model implementation is correct and that its results are not completely unreasonable

Invalidation and crucial experiments

Other sciences assess the quality of a model by trying to invalidate it (Popper).



Validation vs. Invalidation

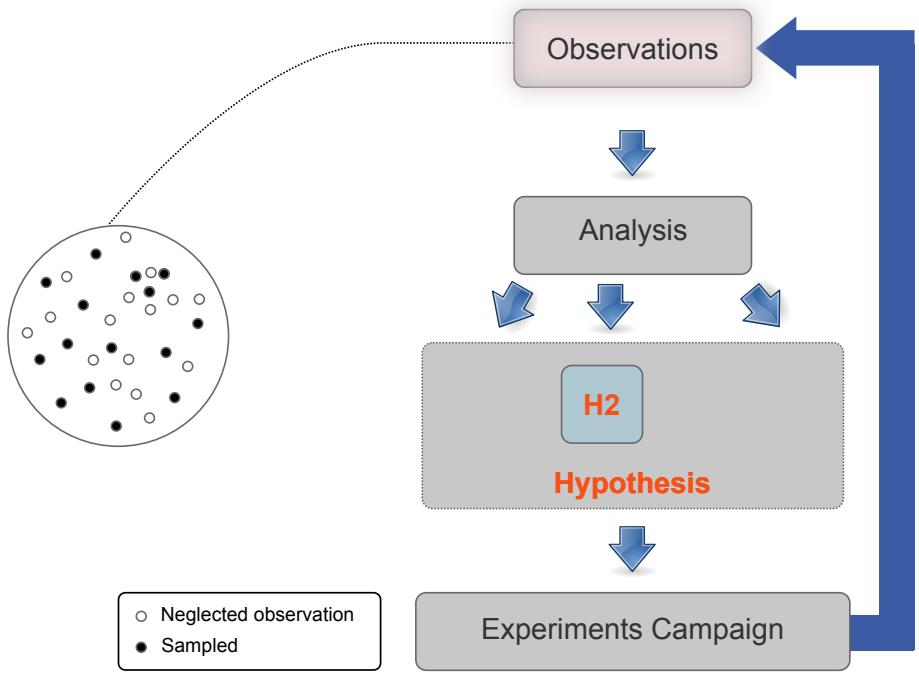
Validation

- Articles full of "convincing" graphs but shallow description, unavailable or broken code
- Optimistic validation, i.e., only for a few cases in which the model is expected to work well

  $\rightsquigarrow$ merely verifies that the model implementation is correct and that its results are not completely unreasonable

Invalidation and crucial experiments

Other sciences assess the quality of a model by trying to invalidate it (Popper).



Validation vs. Invalidation

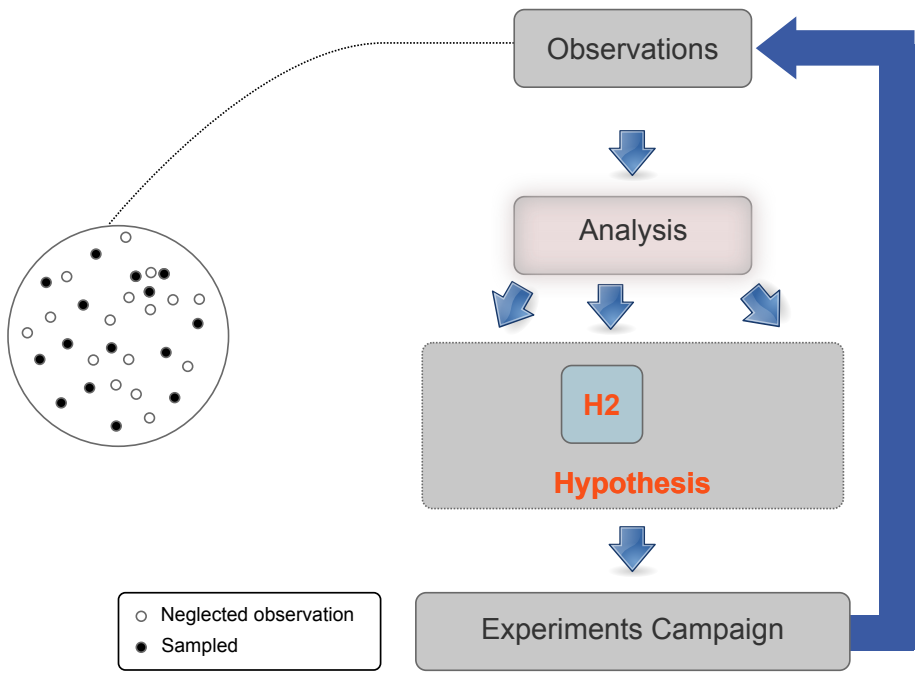
Validation

- Articles full of "convincing" graphs but shallow description, unavailable or broken code
- Optimistic validation, i.e., only for a few cases in which the model is expected to work well

  $\rightsquigarrow$ merely verifies that the model implementation is correct and that its results are not completely unreasonable

Invalidation and crucial experiments

Other sciences assess the quality of a model by trying to invalidate it (Popper).



Validation vs. Invalidation

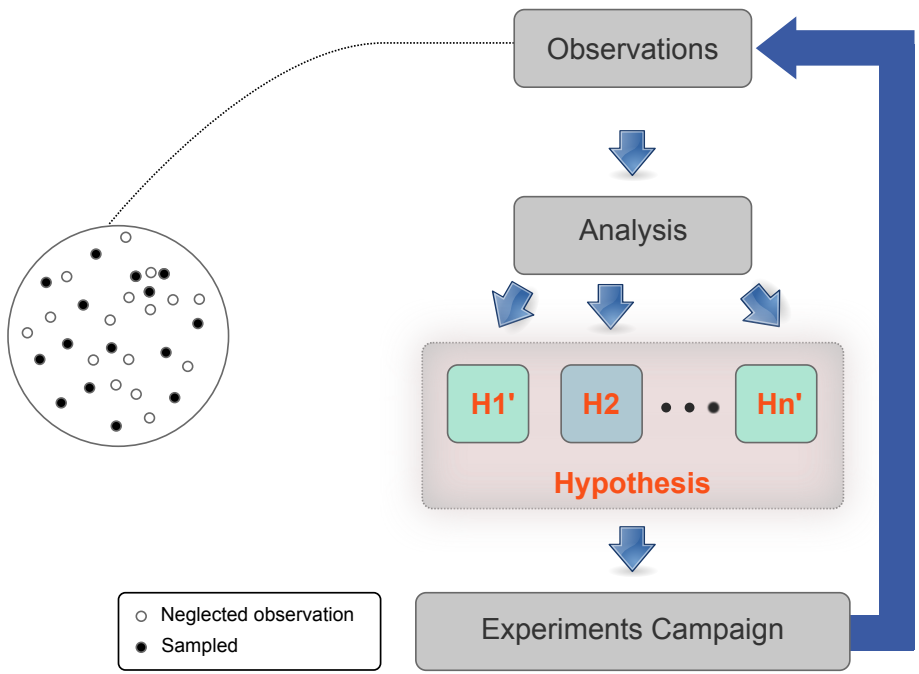
Validation

- Articles full of "convincing" graphs but shallow description, unavailable or broken code
- Optimistic validation, i.e., only for a few cases in which the model is expected to work well

  $\rightsquigarrow$ merely verifies that the model implementation is correct and that its results are not completely unreasonable

Invalidation and crucial experiments

Other sciences assess the quality of a model by trying to invalidate it (Popper).



Validation vs. Invalidation

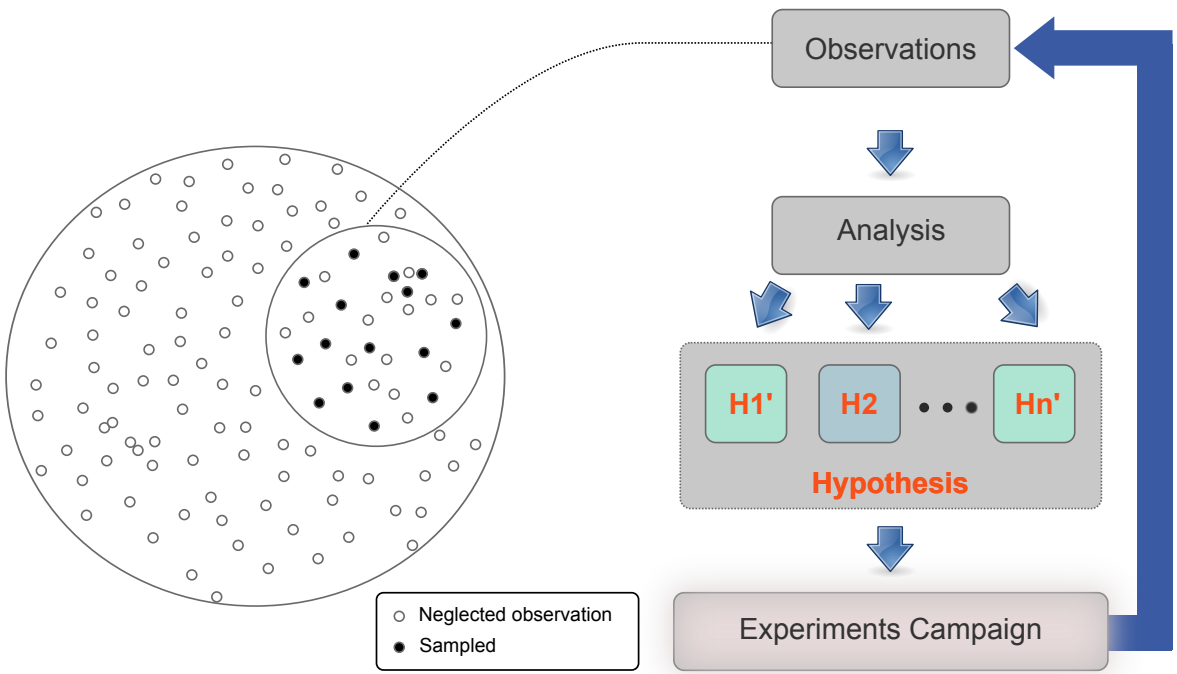
Validation

- Articles full of "convincing" graphs but shallow description, unavailable or broken code
- Optimistic validation, i.e., only for a few cases in which the model is expected to work well

  $\rightsquigarrow$ merely verifies that the model implementation is correct and that its results are not completely unreasonable

Invalidation and crucial experiments

Other sciences assess the quality of a model by trying to invalidate it (Popper).



Validation vs. Invalidation

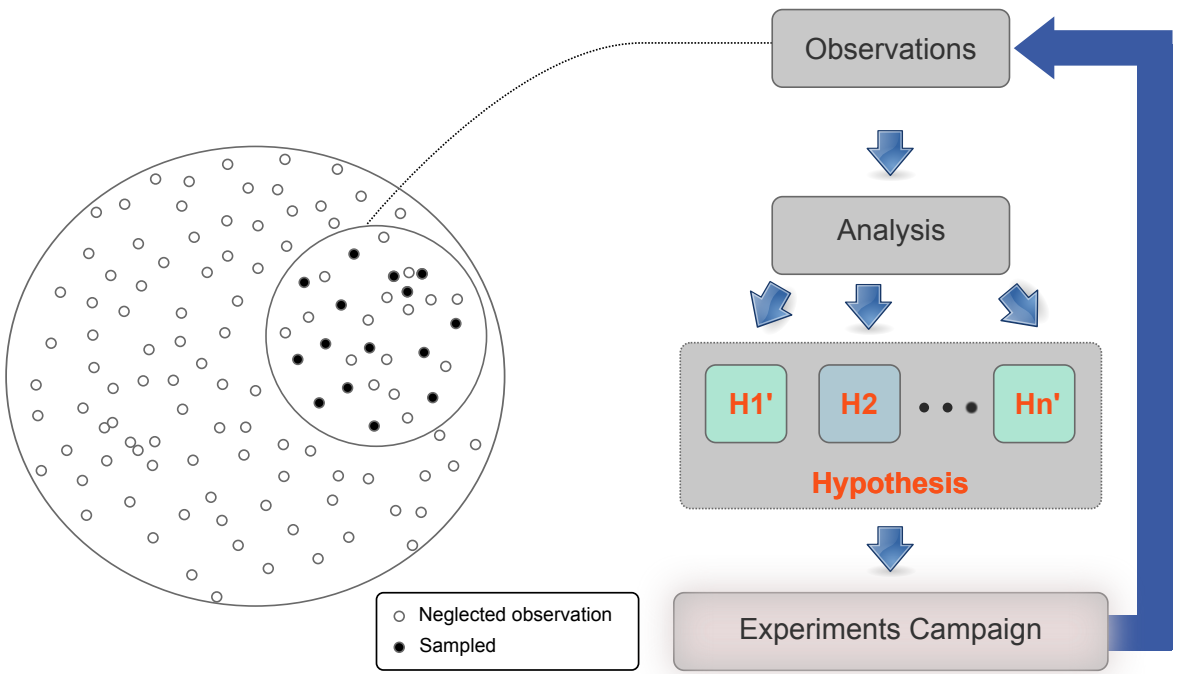
Validation

- Articles full of "convincing" graphs but shallow description, unavailable or broken code
- Optimistic validation, i.e., only for a few cases in which the model is expected to work well

  $\rightsquigarrow$ merely verifies that the model implementation is correct and that its results are not completely unreasonable

Invalidation and crucial experiments

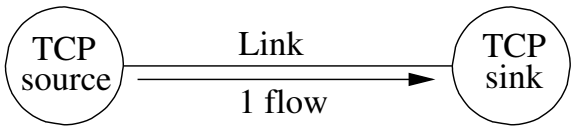
Other sciences assess the quality of a model by trying to invalidate it (Popper).



1. A cyclic process
2. Experiments should be designed to objectively prove or disprove an hypothesis
3. Rejected hypothesis provide generally much more insight than accepted ones

Wanted Feature (1): Flow Control Limitation

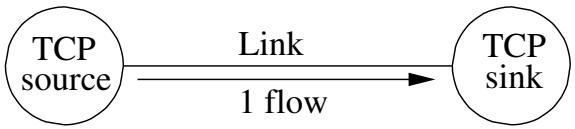
Experimental settings



- Flow throughput as function of L and B
- Fixed size (S=100MB) and window (W=20KB)

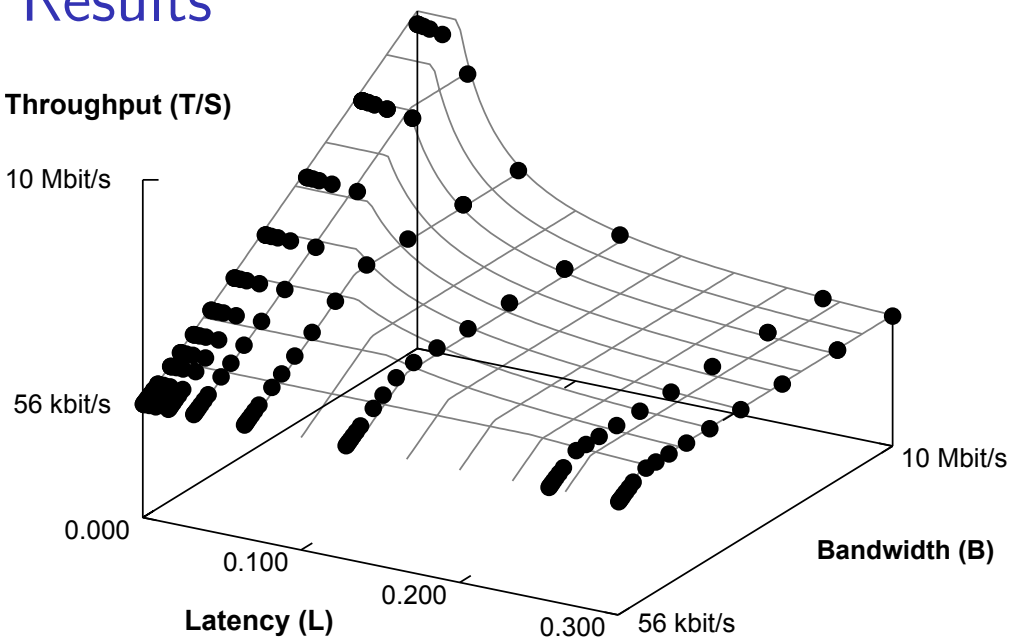
Wanted Feature (1): Flow Control Limitation

Experimental settings



- Flow throughput as function of L and B
- Fixed size (S=100MB) and window (W=20KB)

Results



Legend

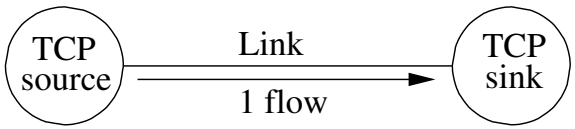
- Mesh: SimGrid results

$$\frac{S}{S/min(B, \frac{W}{2L}) + L}$$

- •: GTNetS results

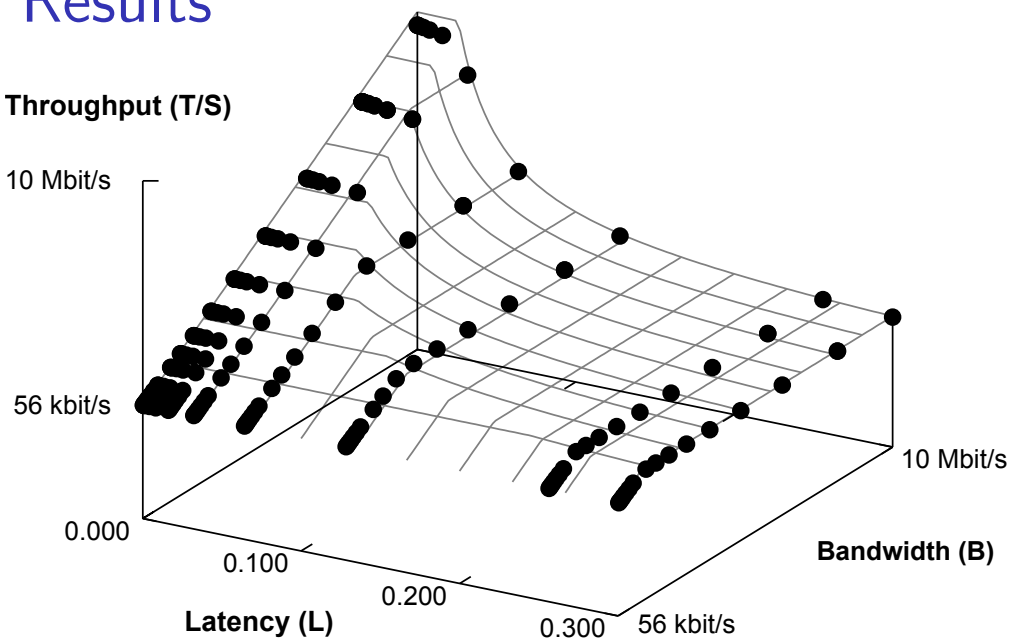
Wanted Feature (1): Flow Control Limitation

Experimental settings



- Flow throughput as function of L and B
- Fixed size (S=100MB) and window (W=20KB)

Results



Legend

- Mesh: SimGrid results

$$\frac{S}{S/min(B, \frac{W}{2L}) + L}$$

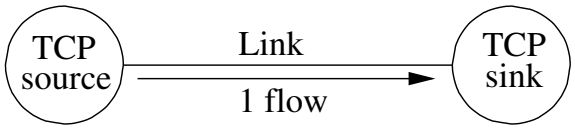
- •: GTNetS results

Conclusion

- SimGrid estimations close to packet-level simulators (when S=100MB)
  - When $B < \frac{W}{2L}$ (B=100KB/s, L=500ms), $|\varepsilon_{max}| \approx \overline{|\varepsilon|} \approx 1\%$
  - When $B > \frac{W}{2L}$ (B=100KB/s, L= 10ms), $|\varepsilon_{max}| \approx \overline{|\varepsilon|} \approx 2\%$

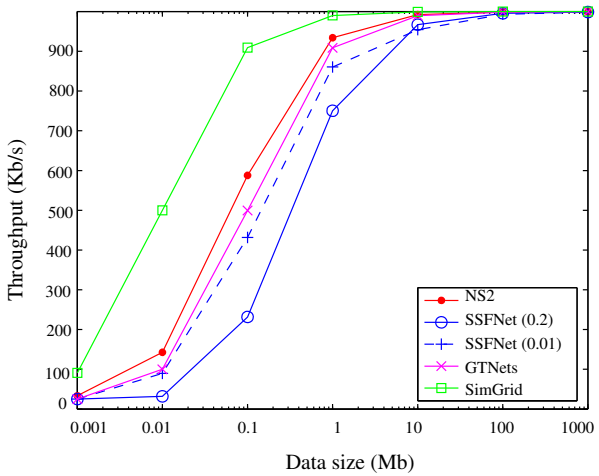
Wanted Feature (2): Slow Start

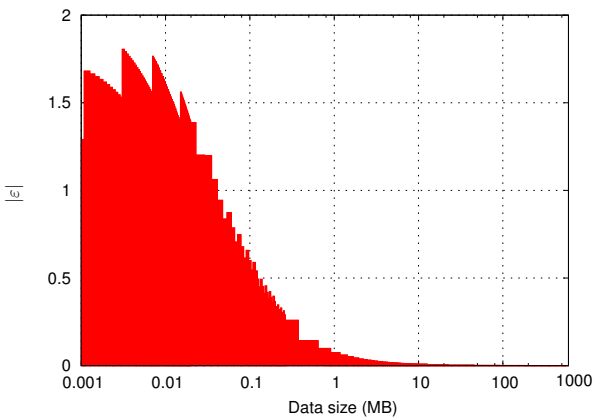
Experimental settings



- Compute achieved bandwidth as function of S
- Fixed L=10ms and B=100MB/s

Evaluation of the SimGrid fluid model

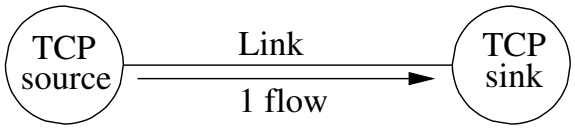




- Packet-level tools don't completely agree

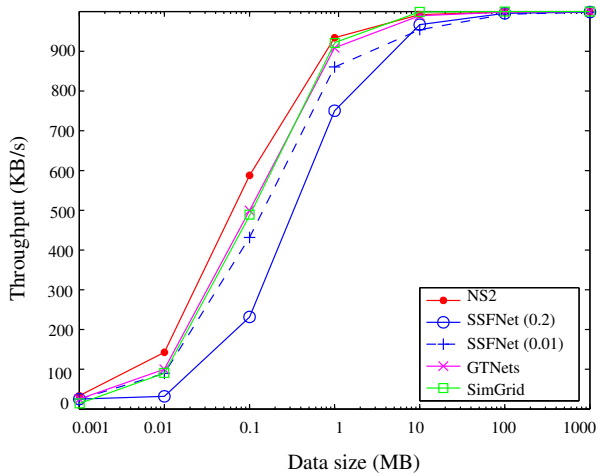
Wanted Feature (2): Slow Start

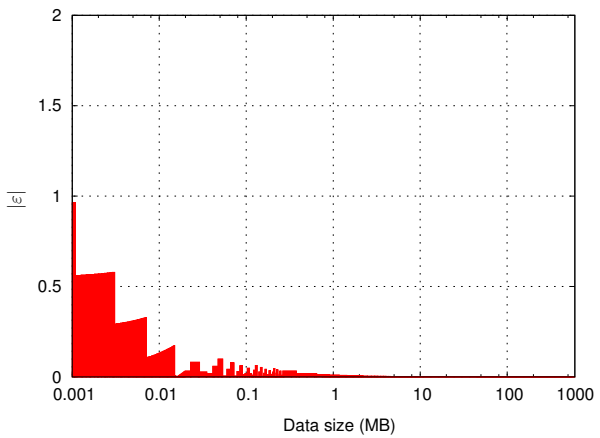
Experimental settings



- Compute achieved bandwidth as function of S
- Fixed L=10ms and B=100MB/s

Evaluation of the SimGrid fluid model





- Packet-level tools don't completely agree
- Statistical analysis of GTNetS slow-start
- Better instantiation
  - Bandwidth decreased (97%)
  - Latency changed to $13.1 \times L$
  - Hence: $Time = \frac{S}{min\left(0.97 \times B, \frac{W}{2L}\right)} + 13.1 \times L$

- This dramatically improve validity range compared to using raw L and B

S	$\overline{\lvert\varepsilon\rvert}$	$\lvert\varepsilon_{max}\rvert$
$S < 100KB$	$\approx 12\%$	$\approx 162\%$
$S > 100KB$	$\approx 1\%$	$\approx 6\%$

Wanted Feature (3): RTT-unfairness

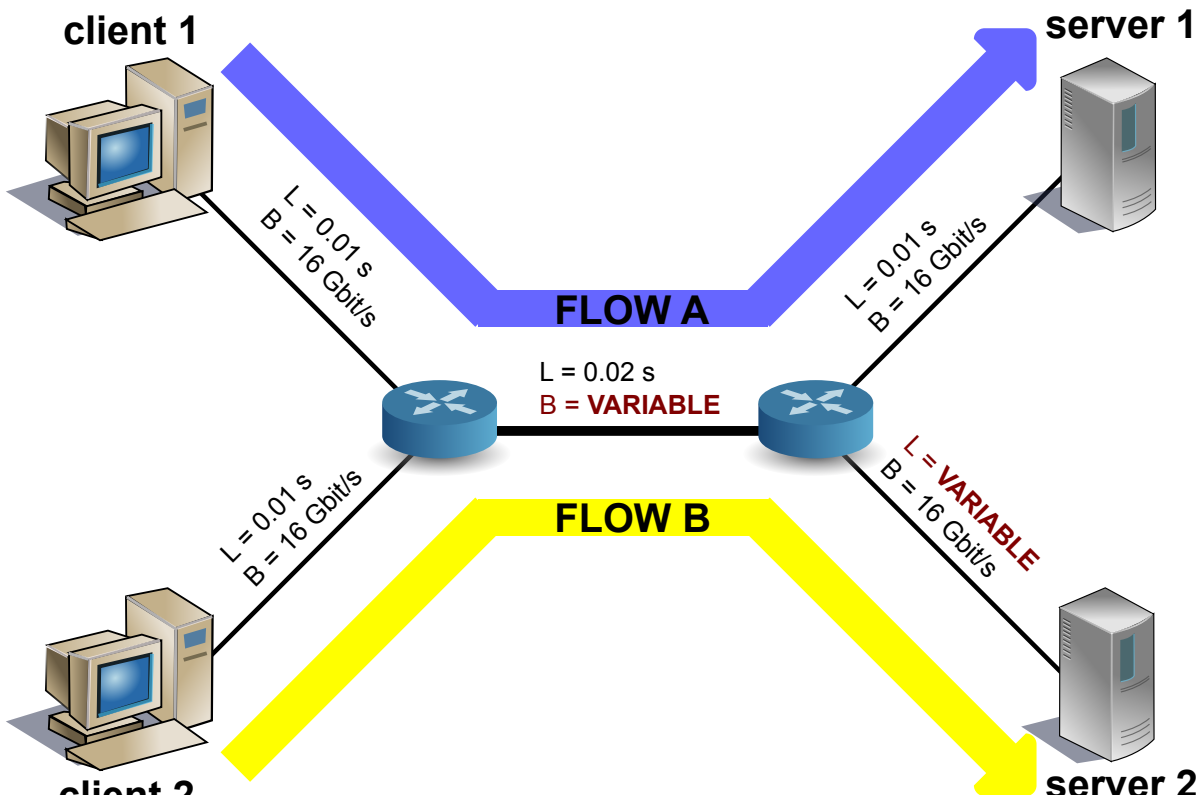
Hypothesis: **Bottleneck links are proportionally shared with respect to flow RTT**

Wanted Feature (3): RTT-unfairness

Hypothesis: **Bottleneck links are proportionally shared with respect to flow RTT**

$$RTT_A.\varrho_A = RTT_B.\varrho_B \text{ where } RTT_i \approx \sum_{\text{flow i uses link j}} (L_j) \text{ (naive model)}$$

- Longer flows (higher latency) will receive slightly less bandwidth

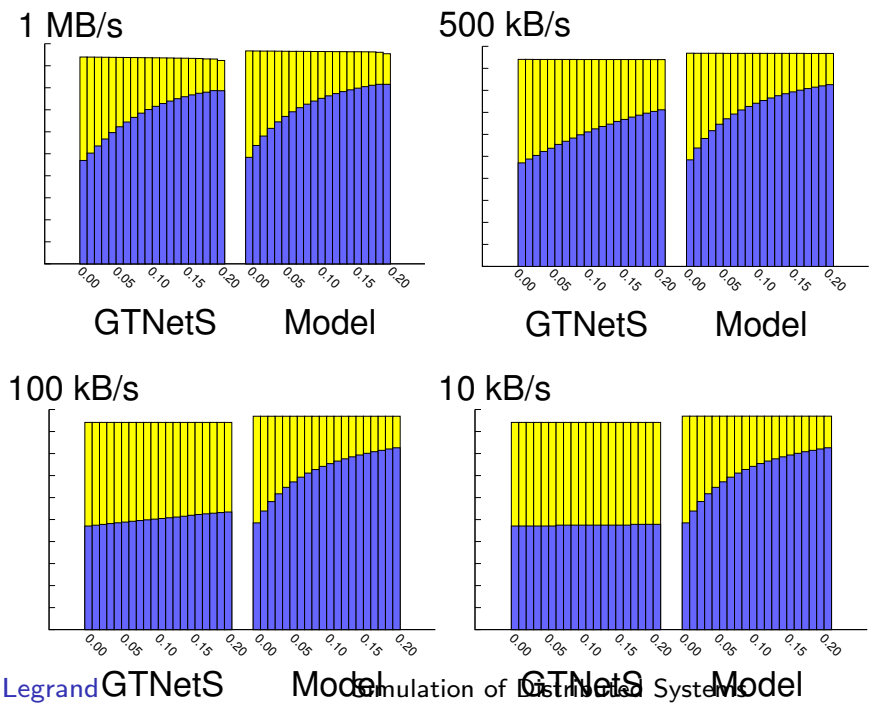


Wanted Feature (3): RTT-unfairness

Hypothesis: **Bottleneck links are proportionally shared with respect to flow RTT**

$$RTT_A.\varrho_A = RTT_B.\varrho_B \text{ where } RTT_i \approx \sum_{\text{flow i uses link j}} (L_j) \text{ (naive model)}$$

- Longer flows (higher latency) will receive slightly less bandwidth

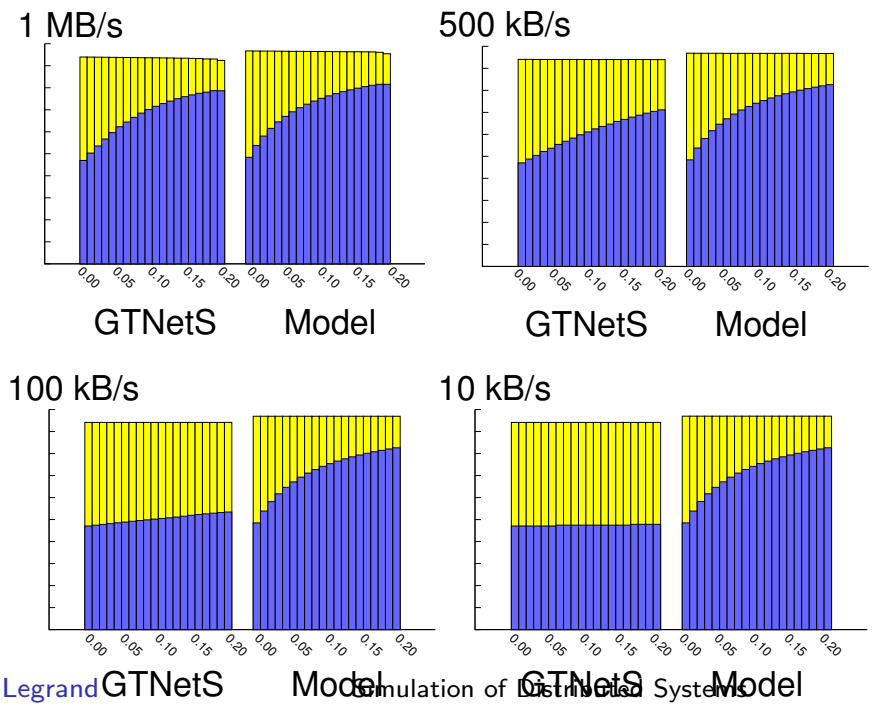


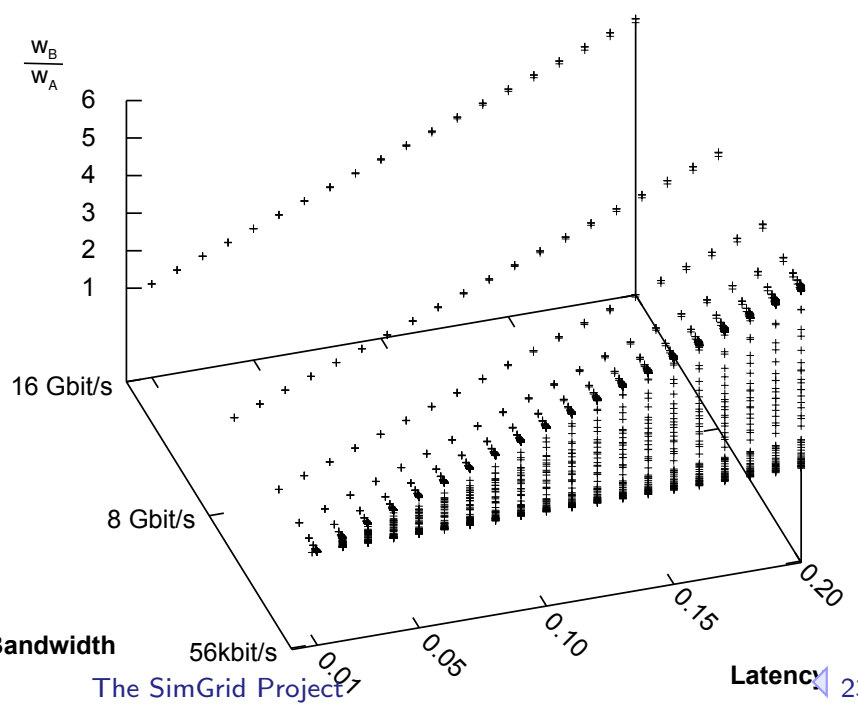
Wanted Feature (3): RTT-unfairness

Hypothesis: **Bottleneck links are proportionally shared with respect to flow RTT**

$$RTT_A.\varrho_A = RTT_B.\varrho_B \text{ where } \cancel{RTT_i \approx \sum_{\text{flow i uses link j}} (L_j)} \text{ (naive model)}$$

- Longer flows (higher latency) will receive slightly less bandwidth
- However, bandwidth also matters



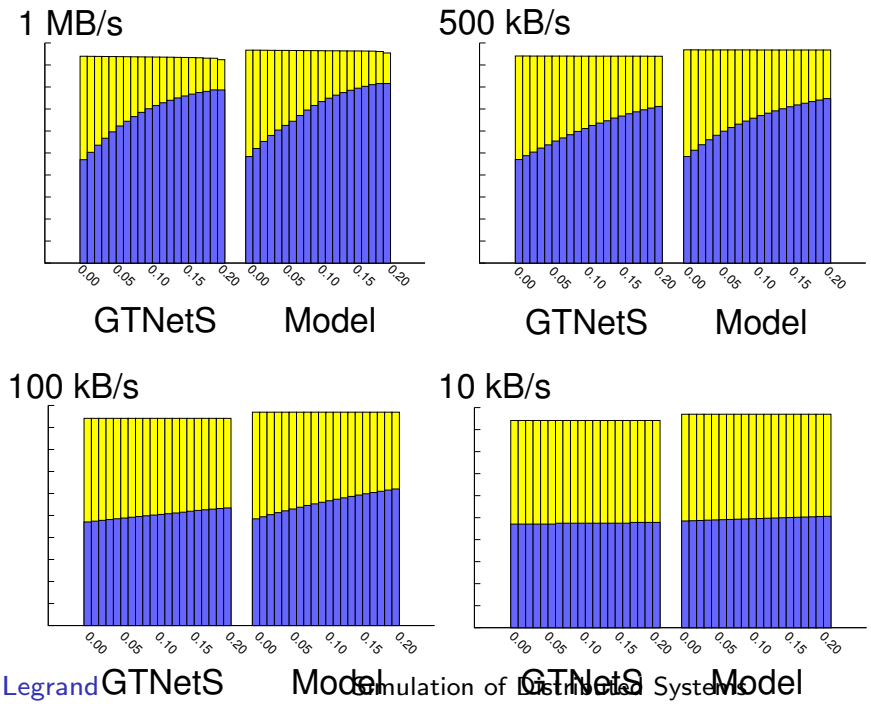


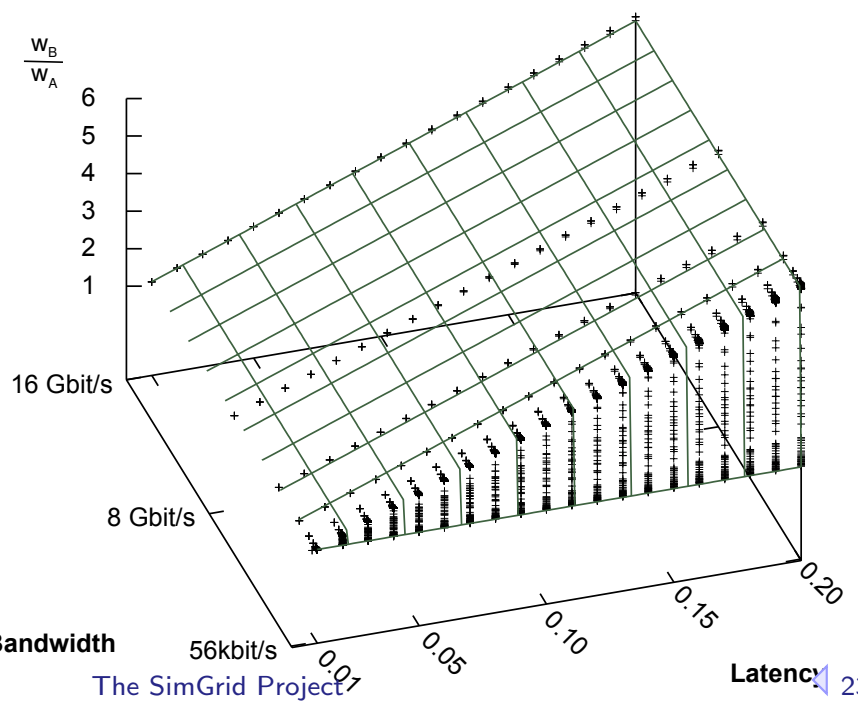
Wanted Feature (3): RTT-unfairness

Hypothesis: **Bottleneck links are proportionally shared with respect to flow RTT**

$RTT_A.\varrho_A = RTT_B.\varrho_B$ **where** ~~$RTT_i \approx \sum_{\text{flow i uses link j}} (L_j)$~~ (naive model)

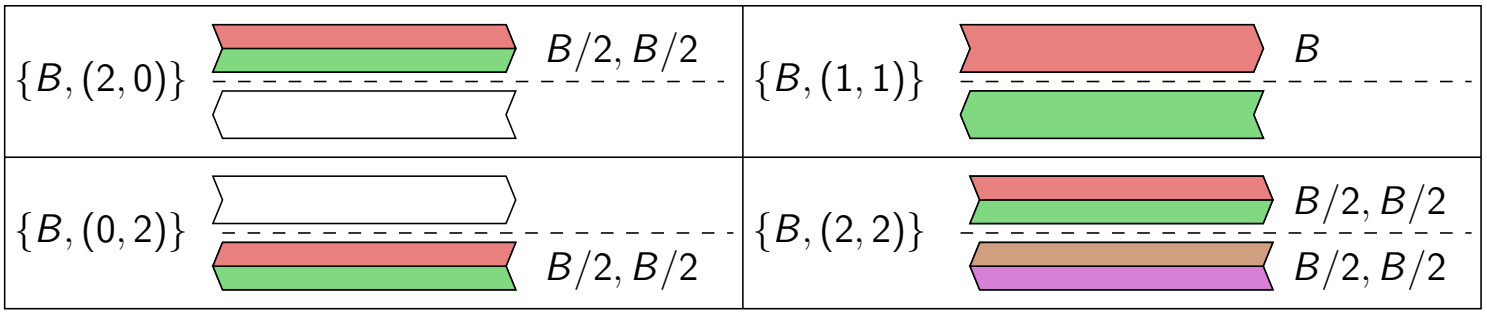
- Longer flows (higher latency) will receive slightly less bandwidth
- However, bandwidth also matters
- Again, instantiation improvement: $RTT_i \approx \sum_{\text{flow i uses link j}} \left(\frac{M}{B_j} + L_j\right)$





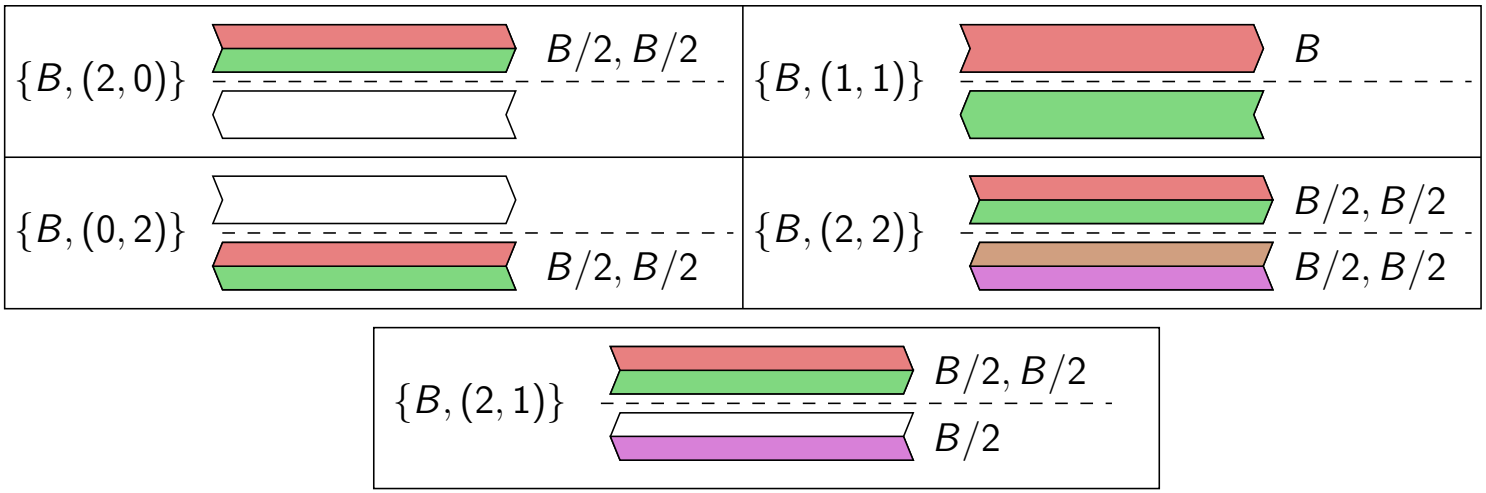
Wanted Feature (4): Cross-Traffic Interference

Take two machines connected by a full-duplex ethernet link.



Wanted Feature (4): Cross-Traffic Interference

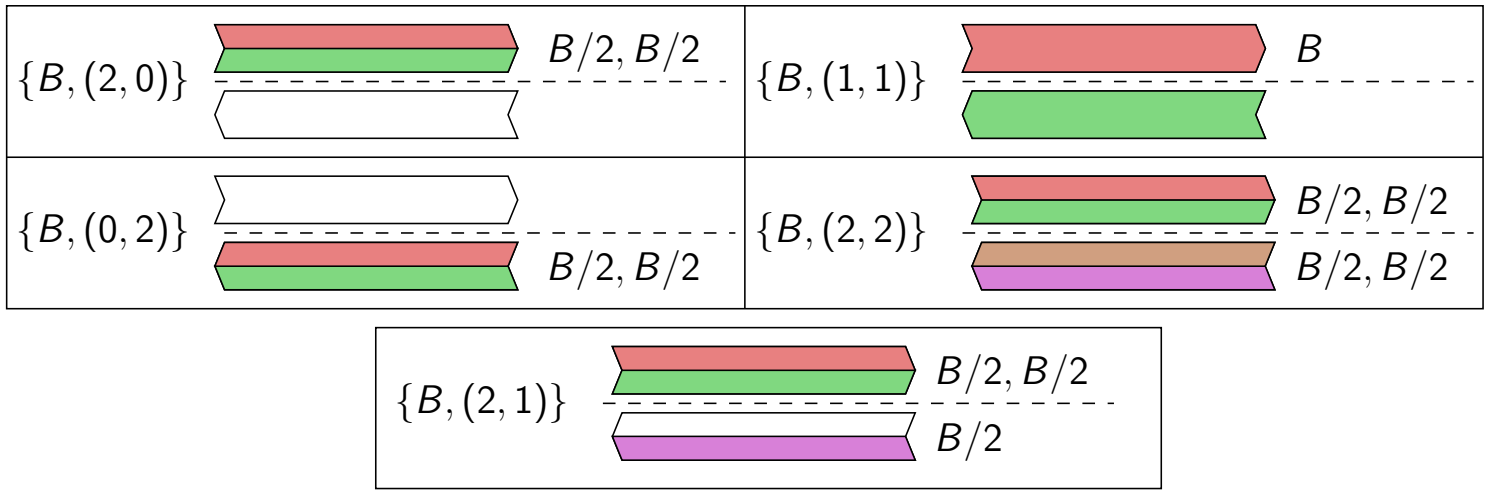
Take two machines connected by a full-duplex ethernet link.



This is a well-known phenomenon when you are using ADSL

Wanted Feature (4): Cross-Traffic Interference

Take two machines connected by a full-duplex ethernet link.



This is a well-known phenomenon when you are using ADSL

Burstiness at micro-scale severely impact macro-scale properties

Modeling such burstiness is ongoing research and resorts to complex differential algebraic equations

Tang et al., *Window Flow Control: Macroscopic Properties from Microscopic Factors*, in INFOCOM 2008

Wanted Features!

Key characteristics of TCP

- Flow-control limitation
- Slow start
- RTT-unfairness
- Cross Traffic Interference

That's messy. Have fluid models a chance ?

- Most previous models (delay, $\sum \log$, $\sum \arctan$, ...) are available in SimGrid
- When well-instantiated, max-min based model can account for all these well-known phenomenon
- The default SimGrid model is LV08: a pragmatic max-min based that is far from perfect but seems reasonnable according to our invalidation studies

Invalidation studies: an endless quest?

Wanted Features!

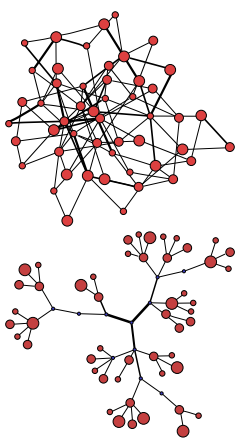
Key characteristics of TCP

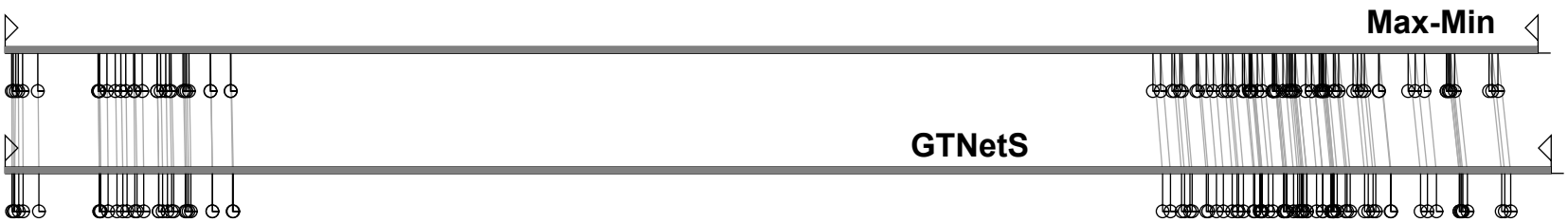
- Flow-control limitation
- Slow start
- RTT-unfairness
- Cross Traffic Interference

That's messy. Have fluid models a chance ?

- Most previous models (delay, $\sum \log$, $\sum \arctan$, ...) are available in SimGrid
- When well-instantiated, max-min based model can account for all these well-known phenomenon
- The default SimGrid model is LV08: a pragmatic max-min based that is far from perfect but seems reasonnable according to our invalidation studies

Invalidation studies: an endless quest?





Wanted Features!

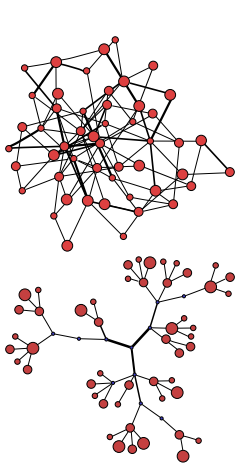
Key characteristics of TCP

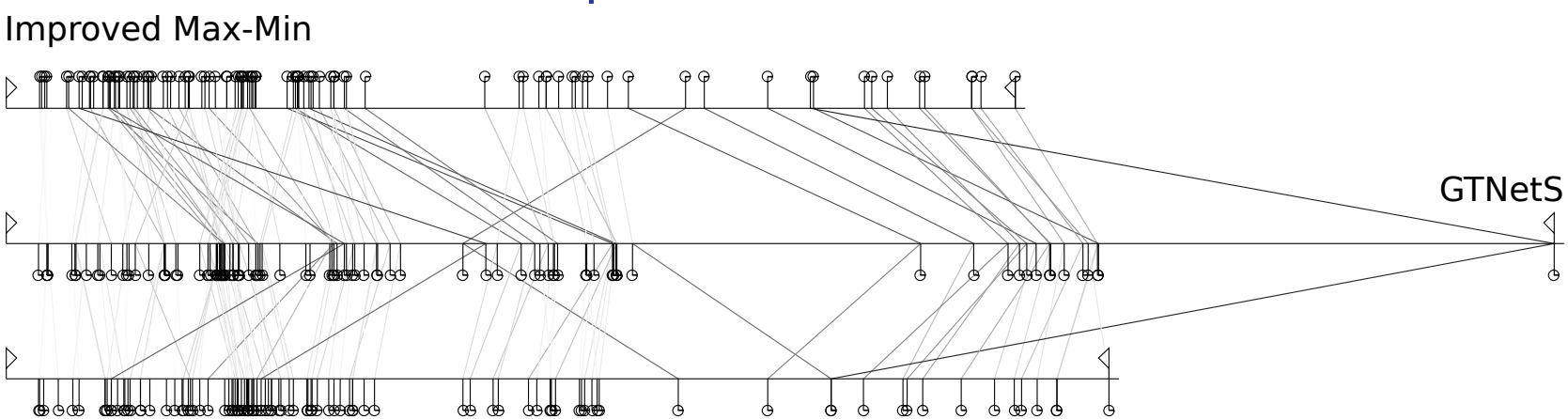
- Flow-control limitation
- Slow start
- RTT-unfairness
- Cross Traffic Interference

That's messy. Have fluid models a chance ?

- Most previous models (delay, $\sum \log$, $\sum \arctan$, ...) are available in SimGrid
- When well-instantiated, max-min based model can account for all these well-known phenomenon
- The default SimGrid model is LV08: a pragmatic max-min based that is far from perfect but seems reasonnable according to our invalidation studies

Invalidation studies: an endless quest?





Wanted Features!

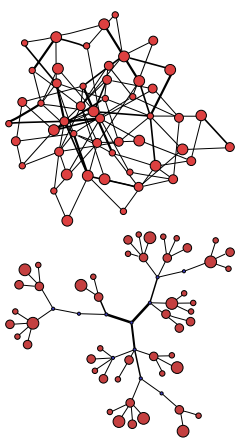
Key characteristics of TCP

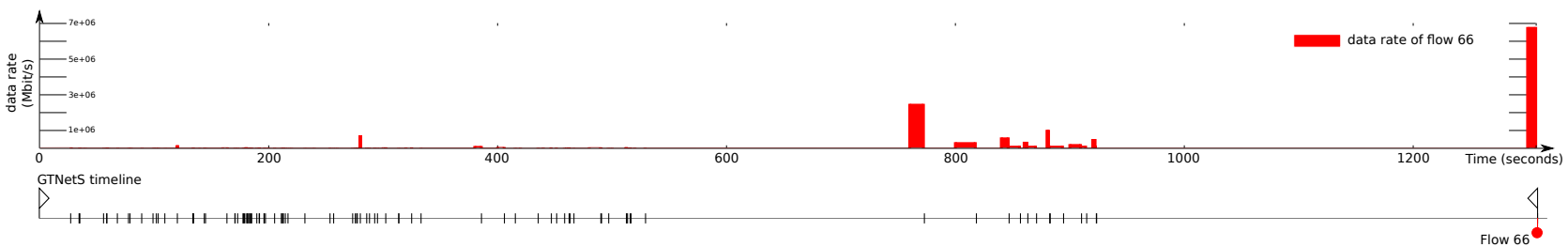
- Flow-control limitation
- Slow start
- RTT-unfairness
- Cross Traffic Interference

That's messy. Have fluid models a chance ?

- Most previous models (delay, $\sum \log$, $\sum \arctan$, ...) are available in SimGrid
- When well-instantiated, max-min based model can account for all these well-known phenomenon
- The default SimGrid model is LV08: a pragmatic max-min based that is far from perfect but seems reasonnable according to our invalidation studies

Invalidation studies: an endless quest?

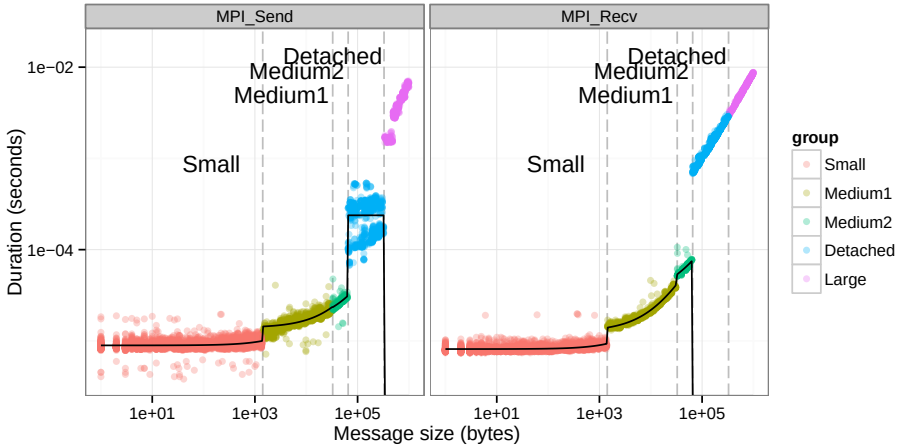




Accuracy of MPI simulation

MPI Oddities and Cluster Peculiarities

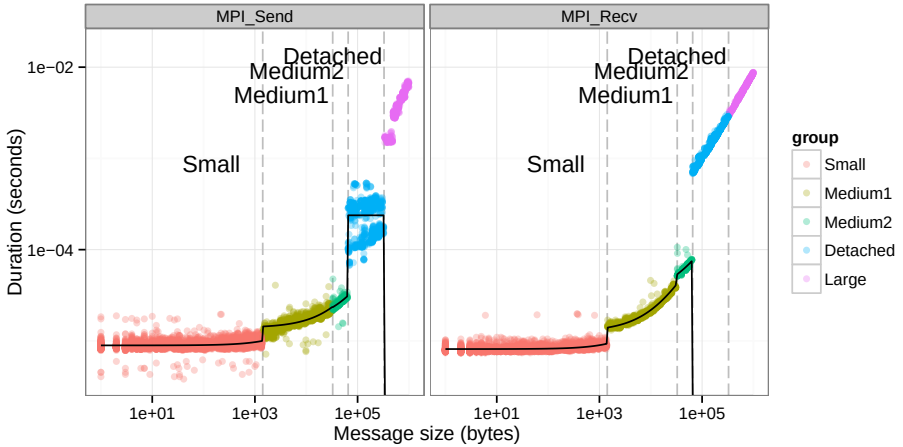
- Protocol switch (1500, 65k, 327k,...),
- Piecewise affine model gives satisfying results



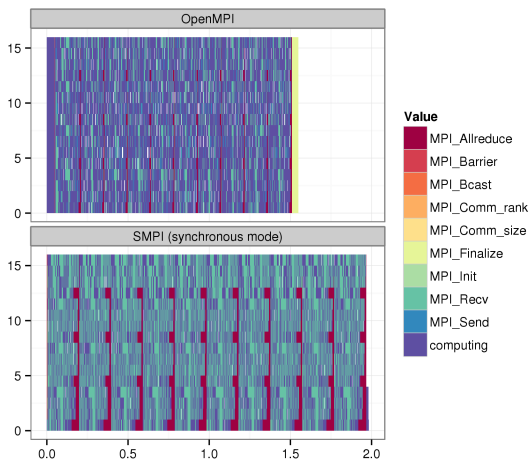
Accuracy of MPI simulation

MPI Oddities and Cluster Peculiarities

- Protocol switch (1500, 65k, 327k,...),
- Piecewise affine model gives satisfying results



Need to accurately model communication/computation overlap



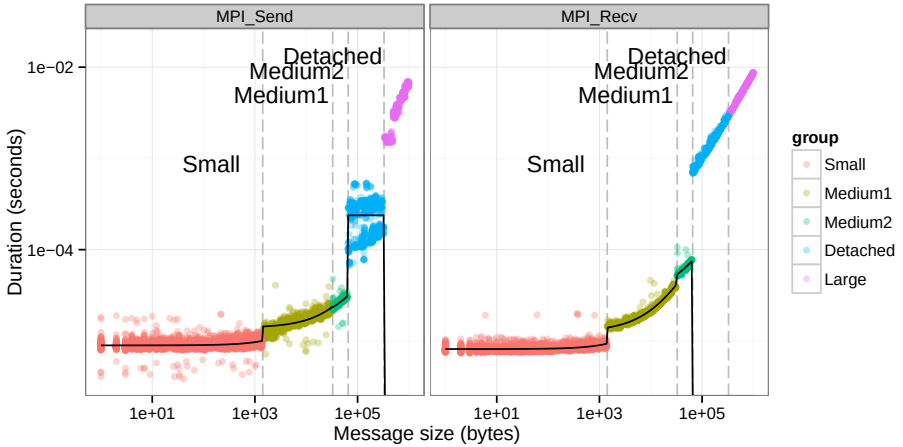
1

1. Forgot the **eager** mode!

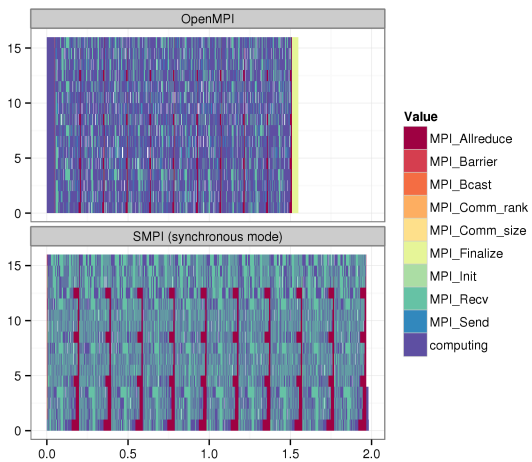
Accuracy of MPI simulatior

MPI Oddities and Cluster Peculiarities

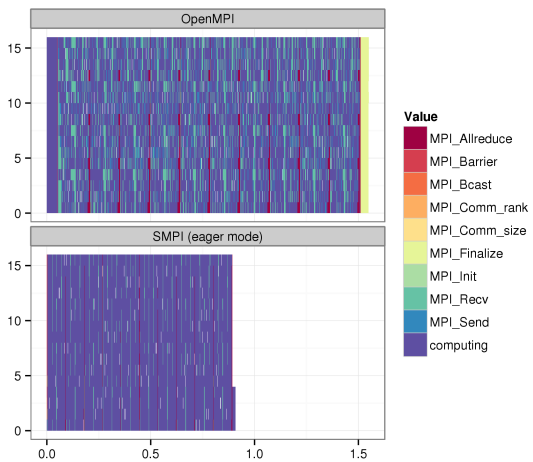
- Protocol switch (1500, 65k, 327k,...),
- Piecewise affine model gives satisfying results



Need to accurately model communication/computation overlap



1



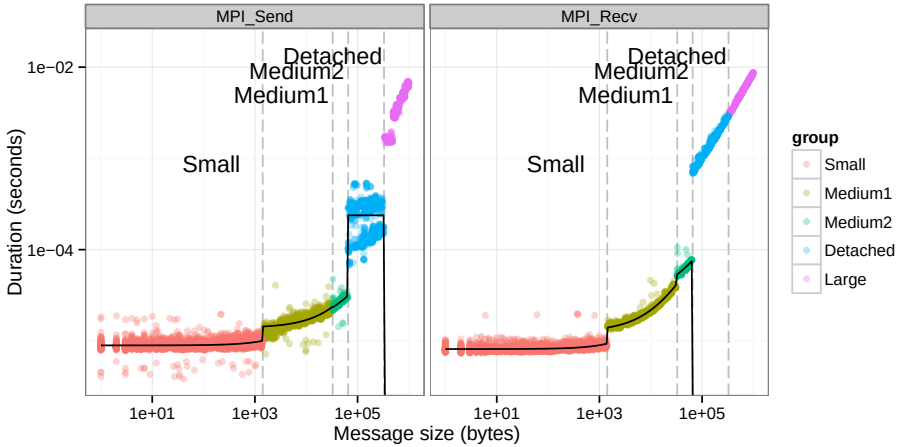
2

1. Forgot the **eager** mode!
2. What about the syscalls and memory copies **overhead** !?!

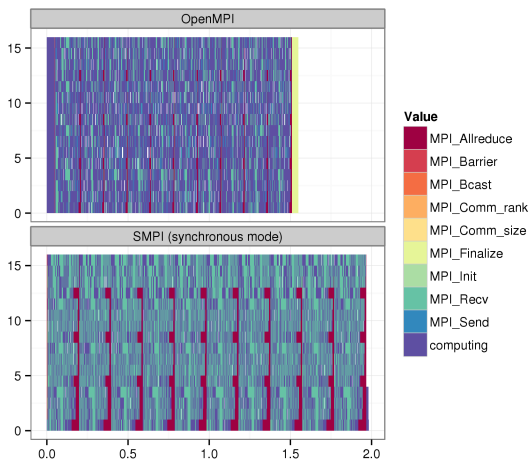
Accuracy of MPI simulation

MPI Oddities and Cluster Peculiarities

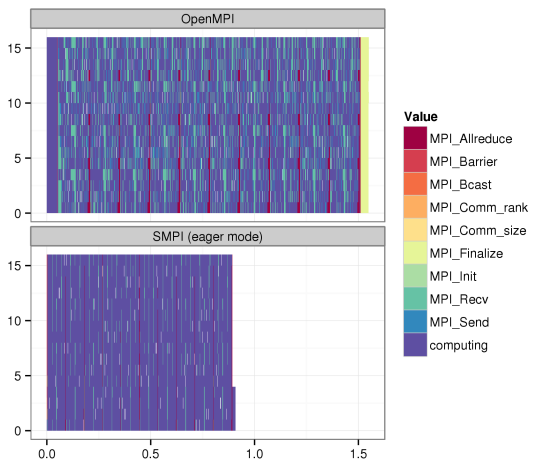
- Protocol switch (1500, 65k, 327k,...),
- Piecewise affine model gives satisfying results



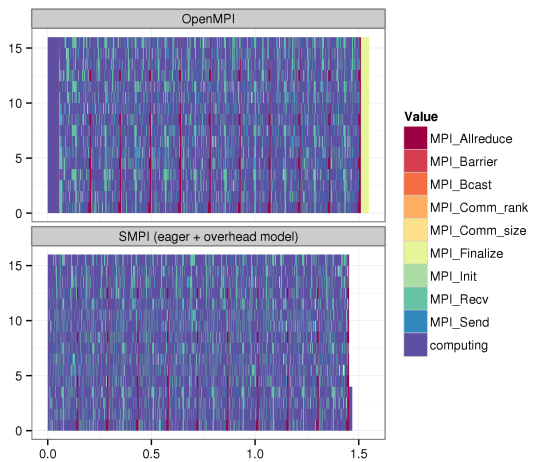
Need to accurately model communication/computation overlap



1



2



3

1. Forgot the **eager** mode!
2. What about the syscalls and memory copies **overhead** !?!
3. OK now but simple modeling error $\rightsquigarrow$ gross inaccuracies

Need to evaluate (i.e., try to invalidate) on a wide range of settings

Modeling Saturation on G5K cluster

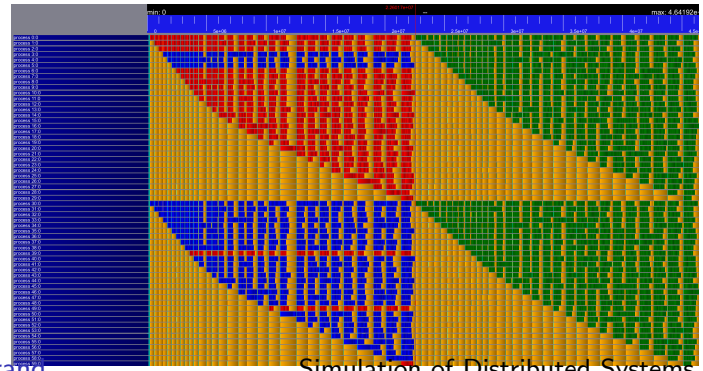
Experimental Setup

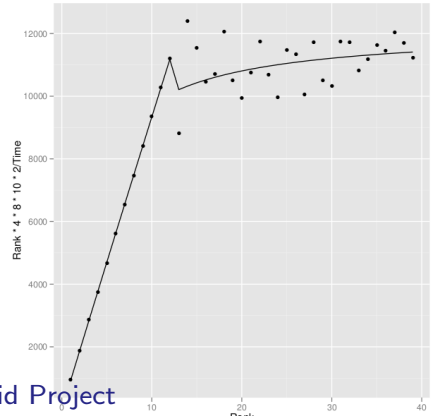
We used the graphene cluster of the Grid'5000 experimental testbed:

- 144 2.53GHz Quad-Core Intel Xeon x3440 nodes
- Four cabinets interconnected by a hierarchy of 10 Gigabit Ethernet switches

Main issue

- Simple collective operations are not too sensitive to bandwidth saturation
- AllToAll stress the network all way long
- Contention may occur within or between cabinets
- Identified issues:
  - Only 65% of max bandwidth (fullduplex $2B$) with `MPI_SendRecv`
  - No saturation within cabinets but similar limitation between cabinets
  - Nodes and cabinet interconnection have three links: up, down, limiter

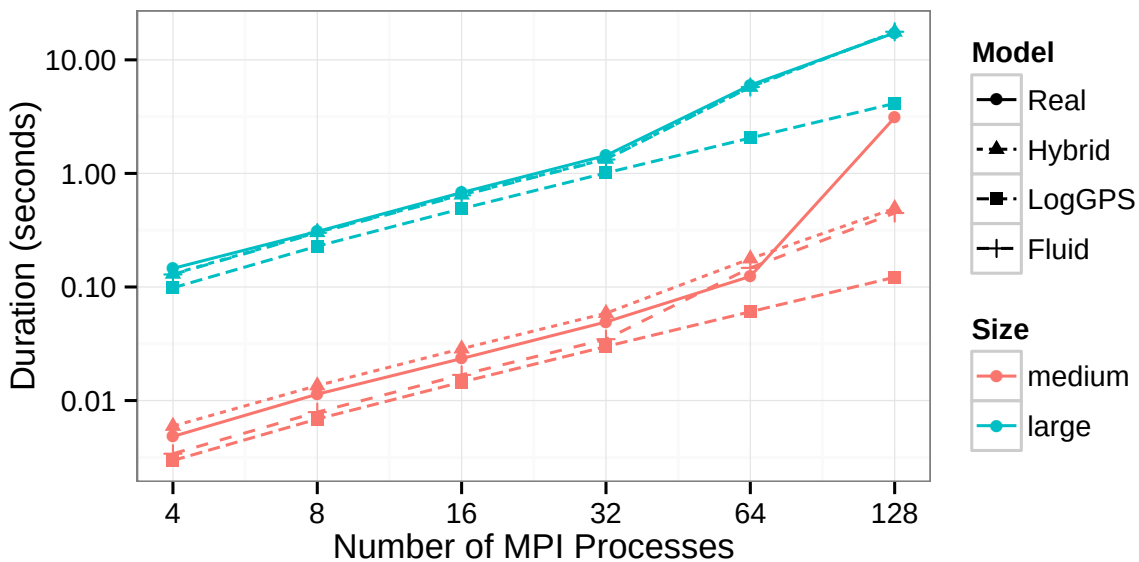




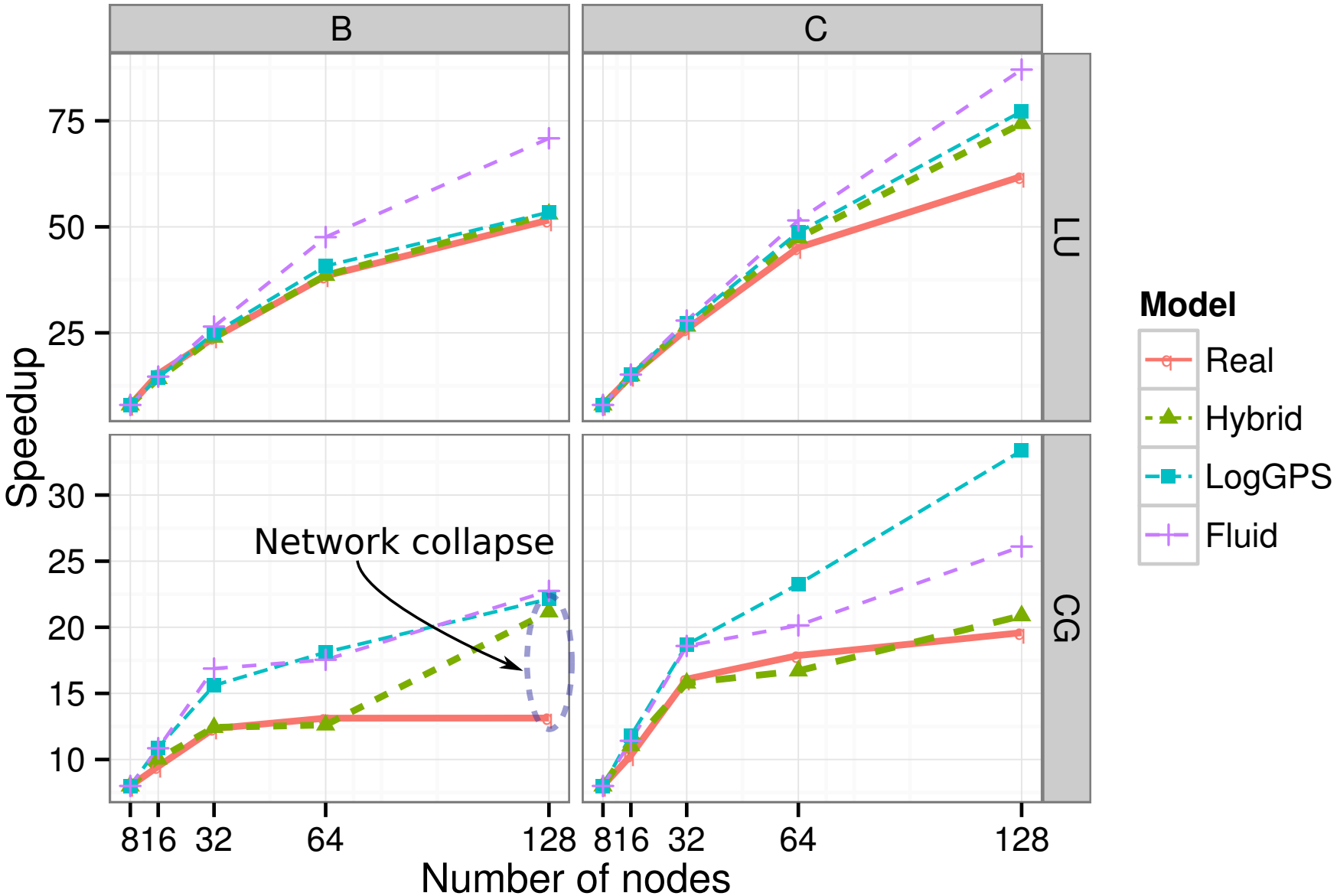
Modeling Collective Communications

Some projects propose to use simple analytic formula. This is a little naive.

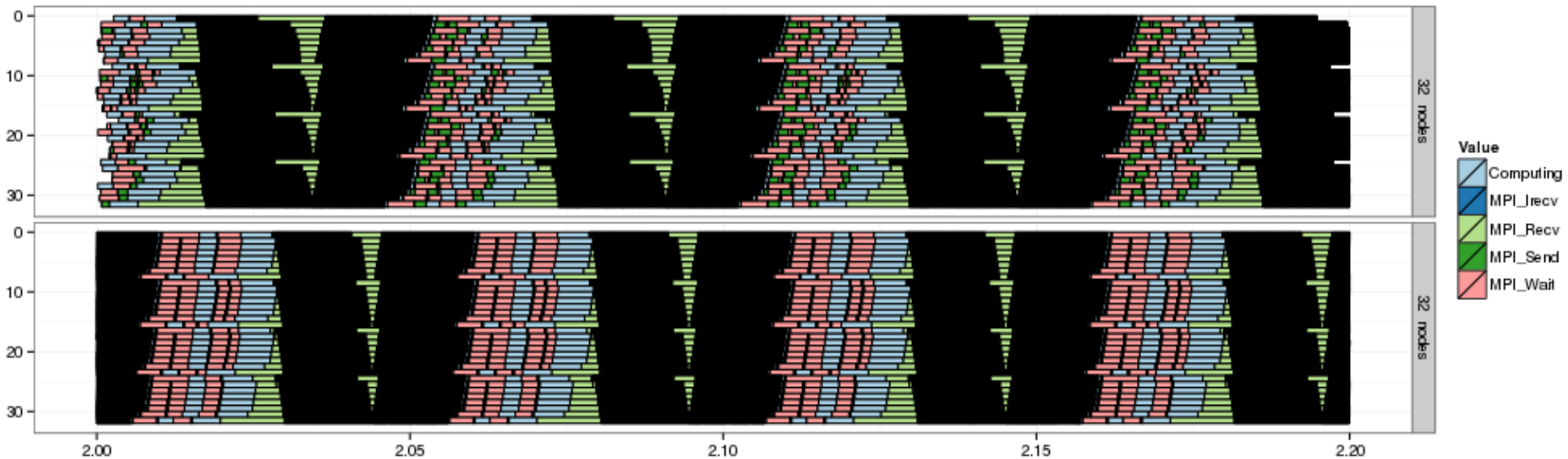
- Real MPI applications use several implementations for each collective, and select the right one at runtime
  - 2300 lines of code for the AllReduce in OpenMPI!!!
- Our initial SMPI versions had only one simple implementation for each one (except alltoall, which had 3):
  - MPICH, OpenMPI, StarMPI: large collection of implementations for collectives, adaptative selector
  - SMPI now: usStarMPI's collectives reused,
  - 100+ collective algorithms plus same selection logic as standard MPI implementations



(In)Validation of SMPI with NAS PB

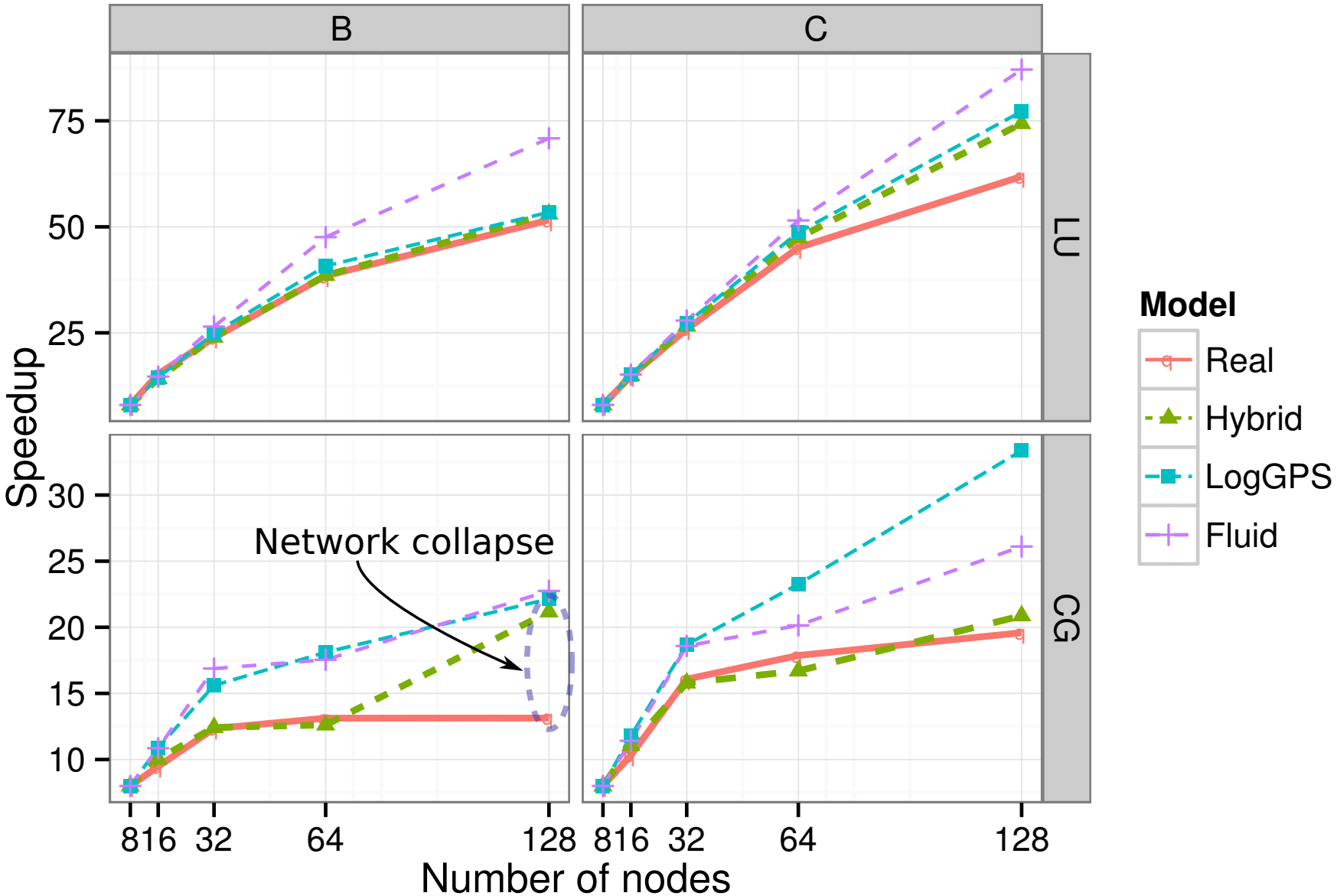


(In)Validation of SMPI with NAS PB



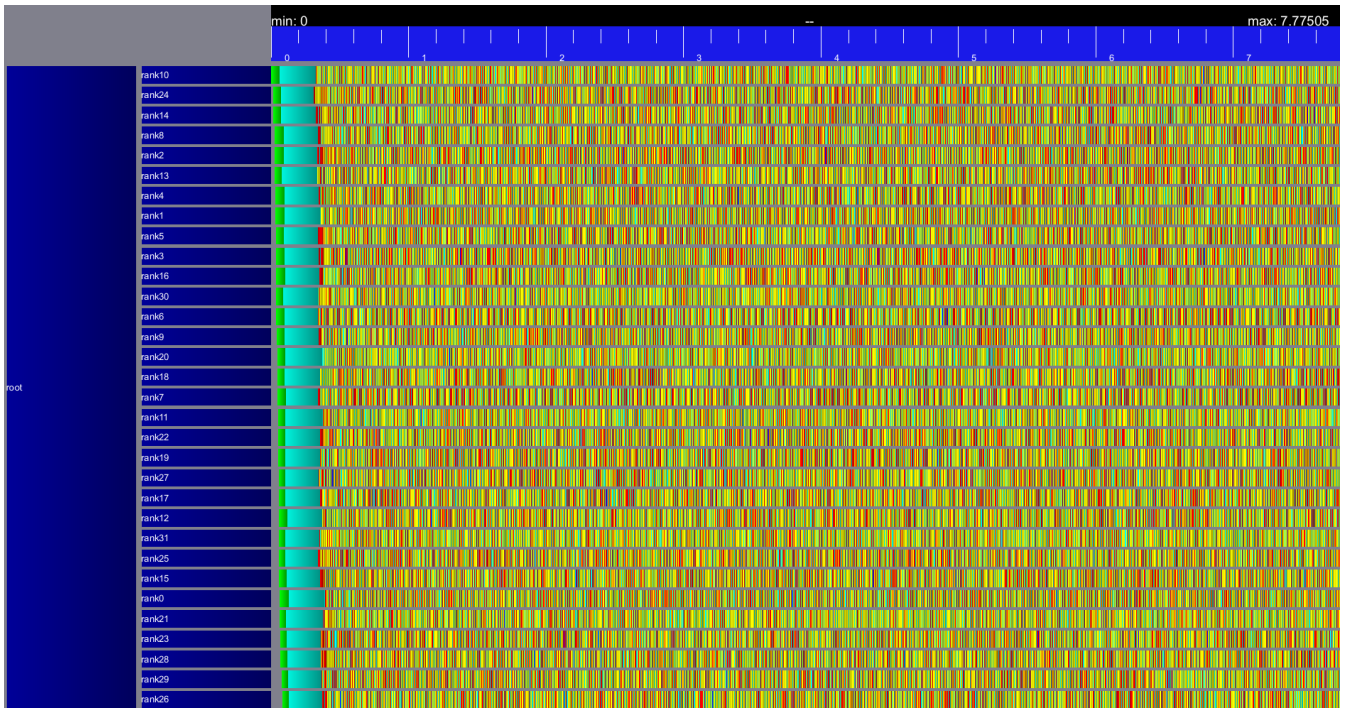
Zoom on 1 second of the LU benchmark with 32 processes: variability should be simulated as well

(In)Validation of SMPI with NAS PB



(In)Validation of Real Life with NAS PB

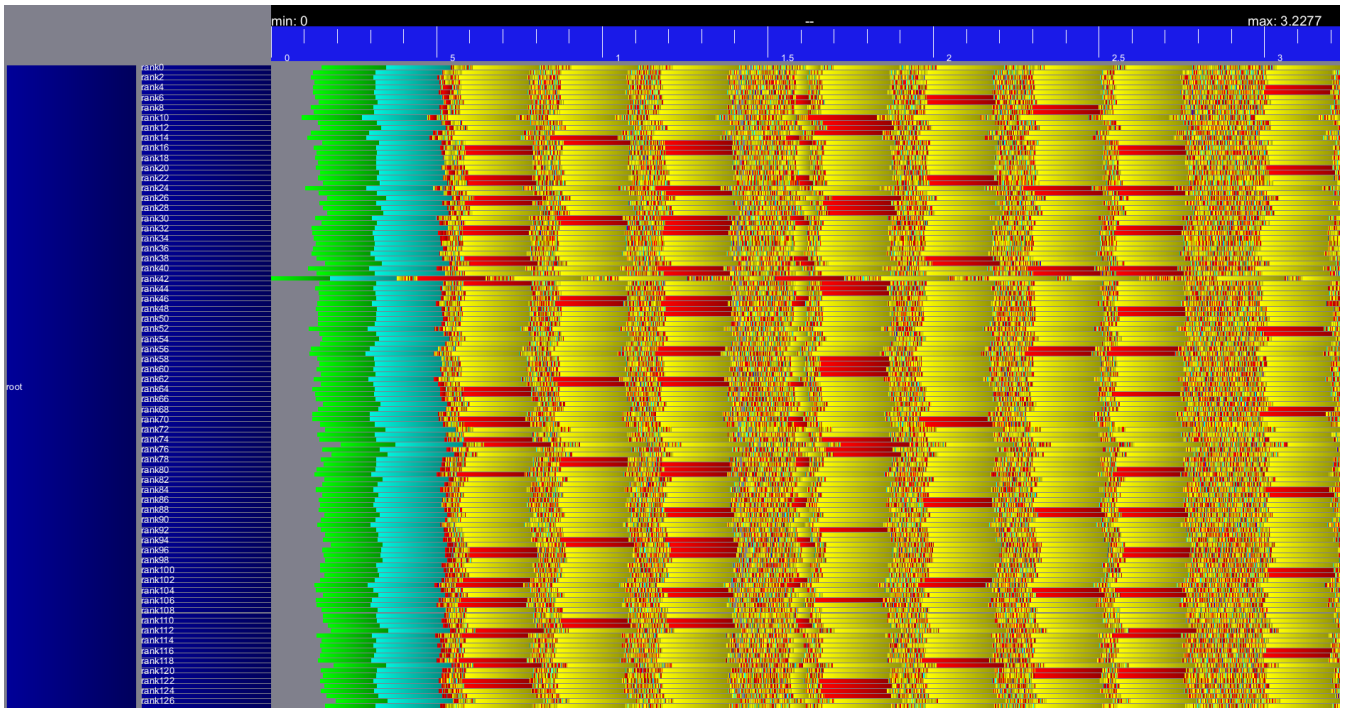
CG 32 nodes, red = send, yellow = wait



1. Communication time (32) $\approx$ a few micro seconds

(In)Validation of Real Life with NAS PB

CG 128 nodes, red = send, yellow = wait



1. Communication time (32) $\approx$ a few micro seconds
2. Communication time (128) $\approx$ sometimes 200 ms!!!
   - Occurs **24 times** leading to a delay of 4.86s out of 14.4s!!!
   - Removing it would lead to the correct estimation
   - Probably due to **TCP RTO** that also arises in the cloud context ("**TCP Incast** Throughput Collapse")

BigDFT

BigDFT in a nutshell

- Density Functional Theory (DFT) code (electronic structure simulation)
- Test application in the European Mont-Blanc project
- Heavily relies on collective operations

Online Simulation issues

- Global variables (Fortran Code, manual privatization with `openmp`), configuration files
- Get rid of computation checks (ruined by computation and memory folding)
- Use different set of collective operations depending on size, instance, ...

BigDFT

BigDFT in a nutshell

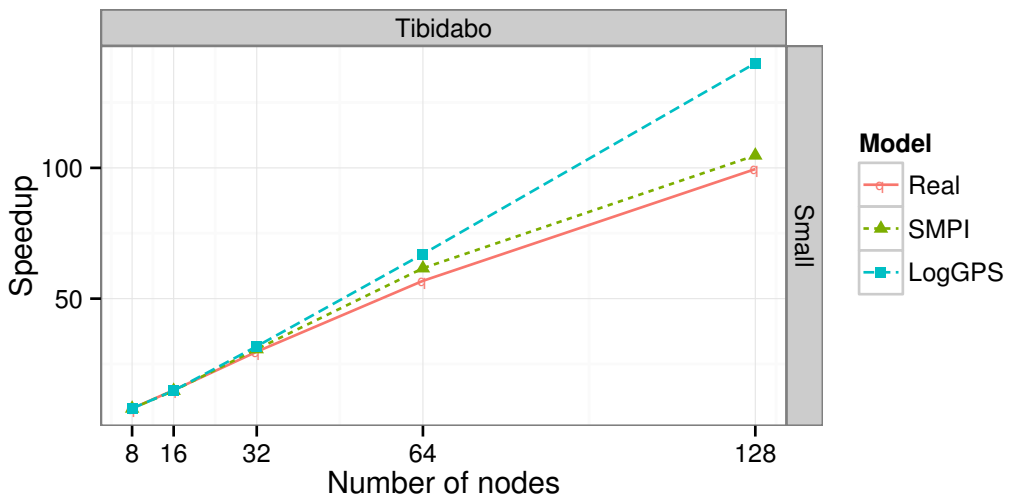
- Density Functional Theory (DFT) code (electronic structure simulation)
- Test application in the European Mont-Blanc project
- Heavily relies on collective operations

Online Simulation issues

- Global variables (Fortran Code, manual privatization with `openmp`), configuration files
- Get rid of computation checks (ruined by computation and memory folding)
- Use different set of collective operations depending on size, instance, ...

First results

- Tibidabo (Mont-Blanc ARM cluster with Ethernet 10G)



Scaling experiment

Simulation/Reality Mismatch

- Bad model $\leadsto$ fix the model if possible (scope,

Simulation/Reality Mismatch

- Bad model $\leadsto$ fix the model if possible (scope,
- Bad instanciation $\leadsto$ re-calibrate the platform

Simulation/Reality Mismatch

- Bad model $\leadsto$ fix the model if possible (scope,
- Bad instanciation $\leadsto$ re-calibrate the platform
- "Application" mismatch (e.g., not the right collective algorithm) $\leadsto$ improve the simulator

Simulation/Reality Mismatch

- Bad model $\leadsto$ fix the model if possible (scope,
- Bad instanciation $\leadsto$ re-calibrate the platform
- "Application" mismatch (e.g., not the right collective algorithm) $\leadsto$ improve the simulator
- Reality problem $\leadsto$ fix reality ?

Outline

Conclusions on Distributed Systems Research

Research on Large-Scale Distributed Systems

- Reflexion about common methodologies needed (reproductible results needed)
- Purely theoritical works limited (simplistic settings $\leadsto$ NP-complete problems)
- Real-world experiments time and labor consuming; limited representativity
- Simulation appealing, if results remain validated

Simulating Large-Scale Distributed Systems or Applications

- Packet-level simulators too slow for large scale studies
- Large amount of ad-hoc simulators, but discutable validity
- Coarse-grain modeling of TCP flows possible (cf. networking community)
- Model instantiation (platform mapping or generation) remains challenging

SimGrid provides interesting models

- Implements non-trivial coarse-grain models for resources and sharing
- Validity results encouraging with regard to packet-level simulators
- Several orders of magnitude faster than packet-level simulators
- Several models availables, ability to plug new ones or use packet-level sim.

Grid Simulation and Open Science

Requirement on Experimental Methodology (what do we want)

- Standard methodologies and tools: Grad students learn them to be operational
- Incremental knowledge: Read a paper, Reproduce its results, Improve.
- Reproducible results: Compare easily experimental scenarios
  Reviewers can reproduce result, Peers can work incrementally (even after long time)

Current practices in the field (what do we have)

- Very little common methodologies and tools; *many* home-brewed tools
- Experimental settings rarely detailed enough in literature

These issues are tackled by the SimGrid community

- Released, open-source, stable simulation framework
- Extensive optimization and validation work
- Separation of simulated application and experimental conditions
- Are we there yet? Not quite

SimGrid and Open Science

Simulations are reproducible ... provided that authors ensure that

- Need to publish source code, platform file, statistic extraction scripts ...
- Almost no one does it. I try to but ... (shame, shame). Why?

SimGrid and Open Science

Simulations are reproducible ... provided that authors ensure that

- Need to publish source code, platform file, statistic extraction scripts ...
- Almost no one does it. I try to but ... (shame, shame). Why?

Technical issues to tackle

- Archiving facilities, Versionning, Branch support, Dependencies management
- Workflows automating execution of test campaigns (`myexperiment.org`) and help sharing results (`manyeyes.alphaworks.ibm.com`)
- We already have most of them (Makefiles, Maven, debs, forges, repositories, ...)
- But still, we don't use it. Is the issue really technical?

SimGrid and Open Science

Simulations are reproducible ... provided that authors ensure that

- Need to publish source code, platform file, statistic extraction scripts ...
- Almost no one does it. I try to but ... (shame, shame). Why?

Technical issues to tackle

- Archiving facilities, Versionning, Branch support, Dependencies management
- Workflows automating execution of test campaigns (`myexperiment.org`) and help sharing results (`manyeyes.alphaworks.ibm.com`)
- We already have most of them (Makefiles, Maven, debs, forges, repositories, ...)
- But still, we don't use it. Is the issue really technical?

Sociological issues to tackle

- A while ago, simulators were simple, only filling gant charts automatically
- We don't have the culture of reproducibility:
  - "My scientific contribution is the algorithm, not the crappy demo code"
  - But your contribution cannot be assessed if it cannot be reproduced!
- I don't have any definitive answer about how to solve it