# Scheduling for Large Scale Distributed Computing Systems: Approaches and Performance Evaluation Issues


Arnaud Legrand

CNRS
University of Grenoble


November 2, 2015

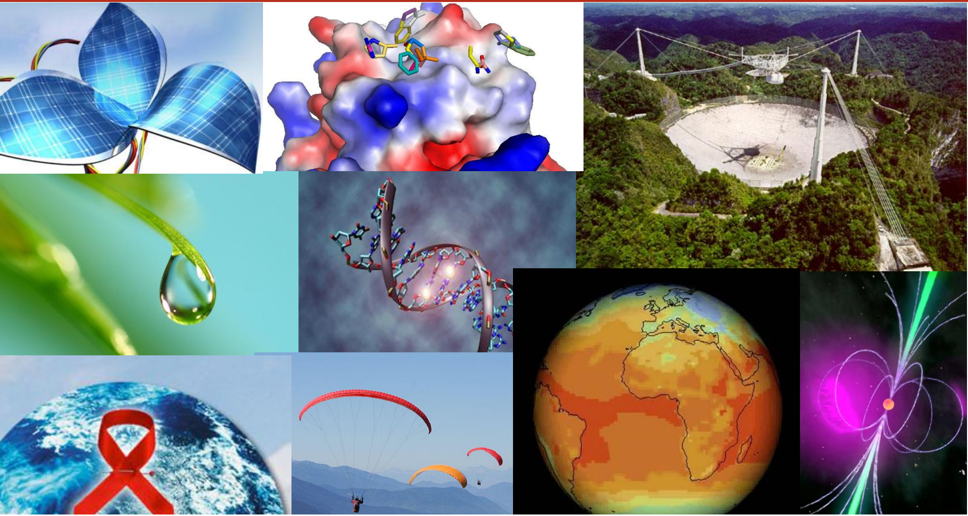
Examiners and President:

- Petra Berenbrink, SFU
- Frédéric Desprez, Inria
- Yves Robert, ENS-Lyon

Reviewers:

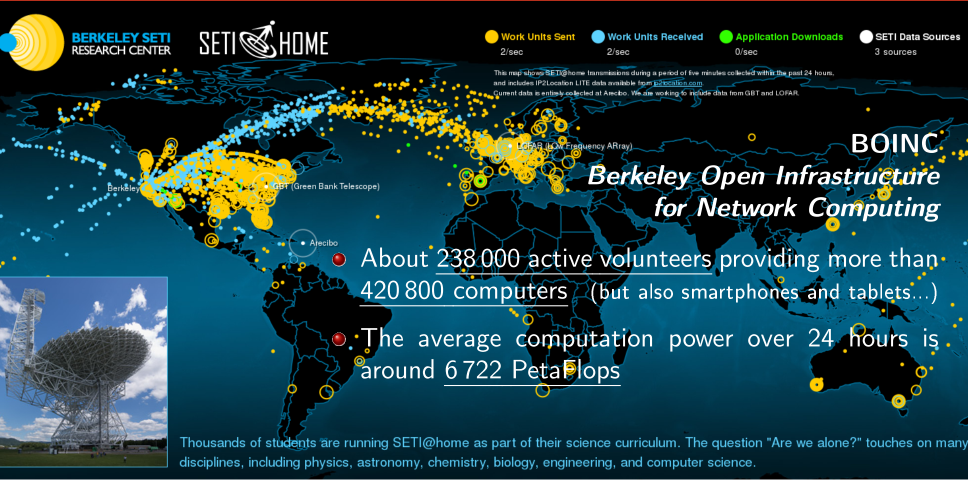
- David Abramson, UQ
- Evripidis Bampis, UPMC
- Marc Snir, UIUC/ANL

# What do... have in common?



*Today the computer is just as important a tool for chemists as the test tube. Simulations are so realistic that they predict the outcome of traditional experiments*

– Nobel committee (chemistry), 2013




**BOINC**

***Berkeley Open Infrastructure for Network Computing***

- About 238 000 active volunteers providing more than 420 800 computers (but also smartphones and tablets...)
- The average computation power over 24 hours is around 6 722 PetaFlops

**Scheduling**: *Where and when should move data and run computations?*

# Challenges for Exploiting Such Platforms

**Scheduling**: *Where and when should move data and run computations?*

Key Features Irregular and large scale

- Heterogeneous
- Complex network topology
- Evolving with technology
- Dynamic
- Shared by several users

Contribution Understand how to

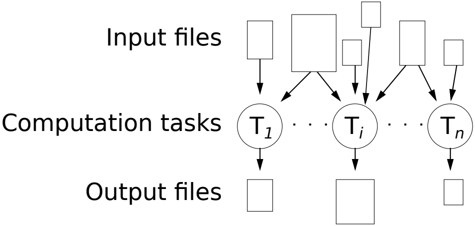
- **Optimize** their exploitation
- **Evaluate** their performance

Approach Try to use **adequate model** or point of view

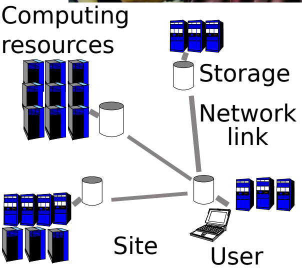
# Outline

# 1999: The APST Experience

Scheduling Parameter Sweep Applications



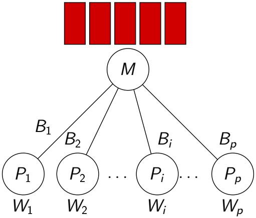

Application structure




Platform model

- NP-hard (many difficulties inside)
  $\rightsquigarrow$ simple heuristics, evaluation with a custom simulator
- **Open problems**:
  - Really understand
  - Truly handle dynamicity
  - More complex topologies
  - Handle several users

# 2000: A Simpler Problem

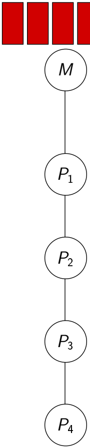

Let's assume all tasks are identical and independent (and have negligible output)

Polynomial! but. . .

- No real intuition
- Polynomial in the number of tasks $n$
- Polynomial in simple cases but NP-hard for non-trivial topologies[Dutot03]

**Probably not the right metric. . .**

# 2000: A Simpler Problem



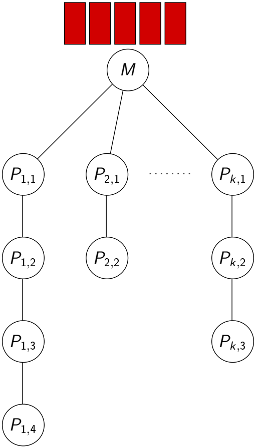

Let's assume all tasks are identical and independent (and have negligible output)

Polynomial! ☺ but...

- No real intuition ☹
- Polynomial in the number of tasks $n$ ☹
- Polynomial in simple cases but NP-hard for non-trivial topologies[Dutot03] ☹

**Probably not the right metric...**

# 2000: A Simpler Problem



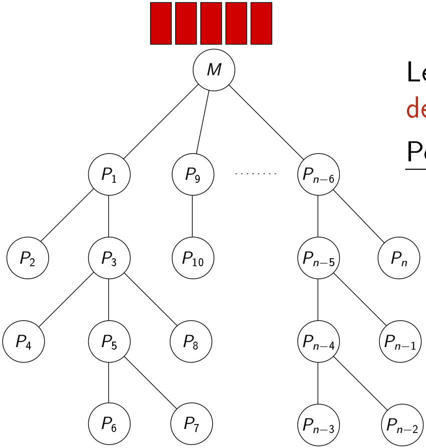

Let's assume all tasks are identical and independent (and have negligible output)

Polynomial! ☺ but...

- No real intuition ☹
- Polynomial in the number of tasks $n$ ☹
- Polynomial in simple cases but NP-hard for non-trivial topologies[Dutot03] ☹

**Probably not the right metric...**

# 2000: A Simpler Problem



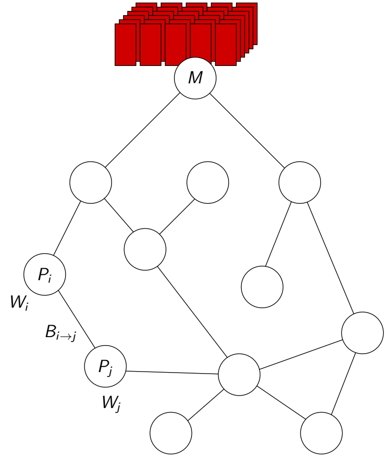

Let's assume all tasks are identical and independent (and have negligible output)

Polynomial! ☺ but...

- No real intuition ☹
- Polynomial in the number of tasks $n$ ☹
- Polynomial in simple cases but NP-hard for non-trivial topologies[Dutot03] ☹

**Probably not the right metric...**

# 2001-2003: Steady-state Throughput




Let's optimize steady-state throughput instead of makespan

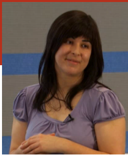
Polynomial! ☺

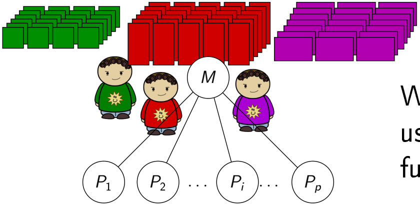
- Equivalent to linear programming or network flow (under some conditions)
- Sometimes provides intuition
- Very flexible formulation

**Remaining issues in 2003**:

- Account for multiple users/applications
- Intuitive distributed solution in the general case

# Non-Cooperative Optimization





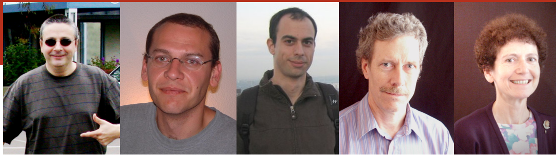

We know the optimal solution for a single user. Is non-cooperative optimization harmful?

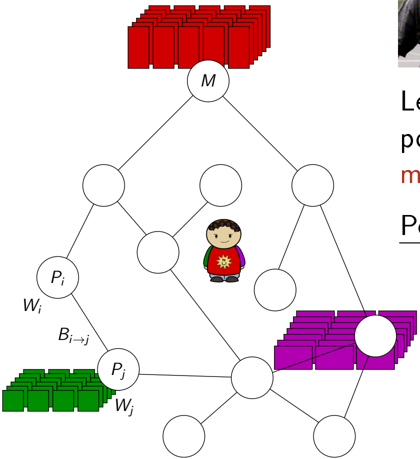
Unique Nash Equilibrium with a closed form formula! ☺

- Characterization of Pareto-optimality
- Inefficiency up to 2 ☹
- No Braess paradox ☺ but resource augmentation results in non-intuitive sharing ☹

**Enforcing cooperation seems worth the effort...** 😉

# Centralized Optimization



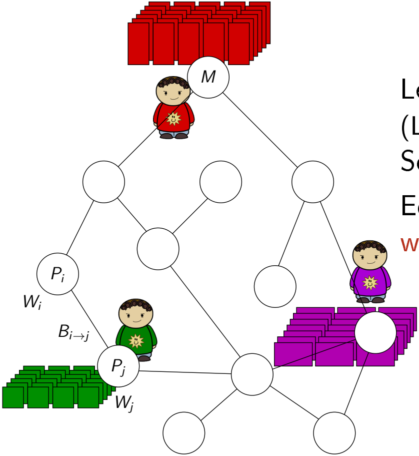



Let's assume we want to be as "fair" as possible between all applications: optimize max-min fairness

Polynomial again! ☺

- Equivalent to linear programming
- Limited intuition in simple settings ☹
- Centralized and static ☹
  - Can guide a dynamic scheduler ☺
- Inadequate fairness ☹

# Cooperative Optimization




Let's use proportional fairness instead!
(Let's also assume a tree deployment per user)
Scary because non linear anymore...

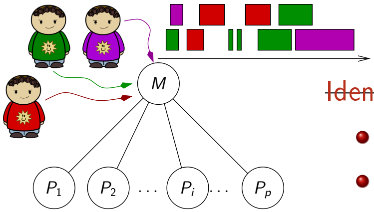
Equivalent to flow control in multi-path networks:

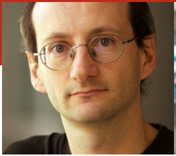
- Lagrangian optimization and distributed gradient descent ⇝ fully distributed and adaptive solution ☺
- Even provides an intuition (shadow prices) ☺

- Adaptation to our context was however non trivial at all...
  - Earlier studies on toy scenarios only
  - Both theoretical and practical convergence issues
  - Finding robust and efficient step sizes was difficult.

**But OK in the end!** ☺

# Online Setups



~~Identical tasks~~ assumption:

- Online arrival
- Divisible, uniform restricted availabilities
- Negligible communication cost

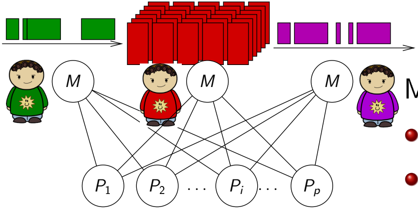
Optimize Stretch of jobs

- Sum vs. Max stretch (L.P.)
- Many competitiveness results
- Practical heuristics avoiding starvation but efficiently exploiting resources

**Remaining key modeling difficulty:**

- Per user (instead of per job) fairness

# Non-Cooperative Optimization




Modeling BOINC:

- Throughput optimization by default
- Need for response time optimization too
- Study which parameters have influence

What happens in case of non-cooperative optimization ?

- Simulation study
- Could reach some N.E.
- Pareto inefficient ($\approx$ 20%)
- Probably not so important...

**Remaining key difficulties:**

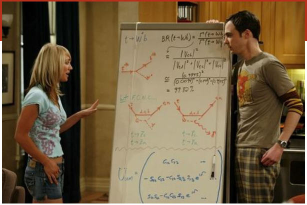
- Response time optimization in the wild
- Managing time varying demand in a sound way

# Outline

# Context and Motivation

- These systems are so **complex** that solely evaluating through equations has become impossible
- Performing experiments on such infrastructures is costly and sometimes not even possible

**We should study them as *Natural* objects**



The Big Bang Theory

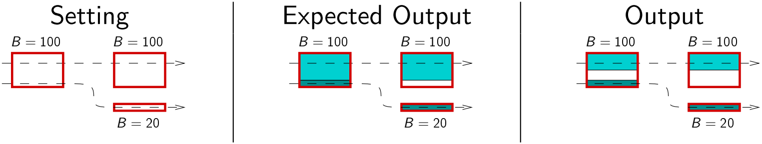
Large Hadron Collider

Other sciences experiment with real systems but also routinely use computers to understand complex systems

**How to faithfully evaluate the performance of such systems through simulation?**

# The practice in the field is. . . disappointing

- Experimental settings are rarely detailed enough in literature
- Many short-lived simulators; few sound and established tools
  - Grid/Cloud: OptorSim GridSim GroudSim CloudSim iCanCloud
  - Volunteer Computing: SimBA EmBOINC SimBOINC . . .
  - P2P: PeerSim P2PSim OverSim . . .
  - HPC: Dimemas PSINS LogGOPSim BigSim MPI-SIM . . .
  - . . .
- Simulating grids or clouds? Experts wanted!




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

People keep reinventing the wheel in a bad way

# A Collaborative Project

- 1999-2000: SimGrid 1.0 by Henri Casanova
- 2001-2003: Needed for my own research and my office-mates liked it
  - SimGrid 2.0 (A. Legrand, M. Quinson)
- 2004: Major rewriting (A. Legrand, M. Quinson, F. Suter)
  - Getting ready for SimGrid 3.*
- 2005-2008: We realized SG was also a research object
- 2009-2012: ANR USS-SimGrid (+ A. Giersh, L. Schnorr, . . . ).
  - P2P, early devs for HPC.
- 2012-2015: ANR SONGS (+ A. Lèbre, A.C. Orgerie, L. Eyraud, . . . )
  - HPC, Cloud infrastructures

More than 1260 citations. At least 162 publications on or using SimGrid.

An open and mature project with an endless quest for
Scalability and Validity

# Validity 1/2

## 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

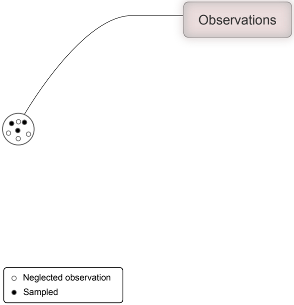
# Validity 1/2

## 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
  $\leadsto$ 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



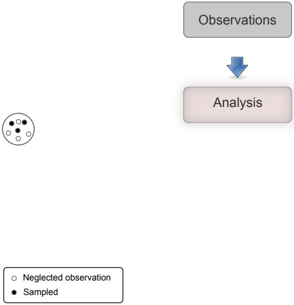
# Validity 1/2

## 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



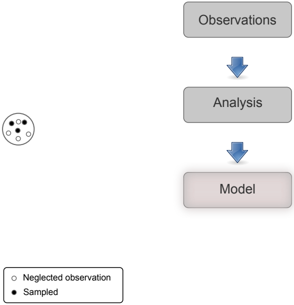
# Validity 1/2

## 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



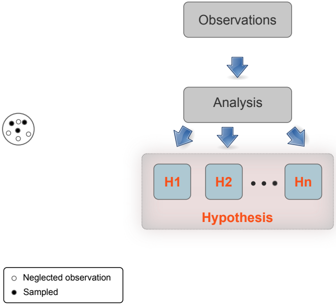
# Validity 1/2

## 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
  $\leadsto$ 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



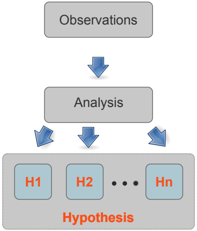
# Validity 1/2

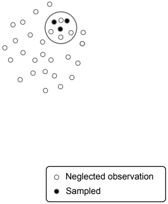
## Validation

- 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



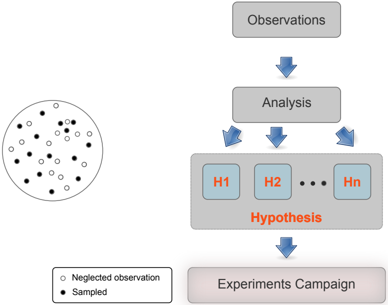
# Validity 1/2

## 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
  
  $\leadsto$ 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



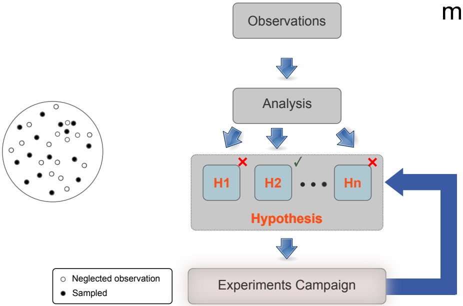
# Validity 1/2

## 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
  ⇝ 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



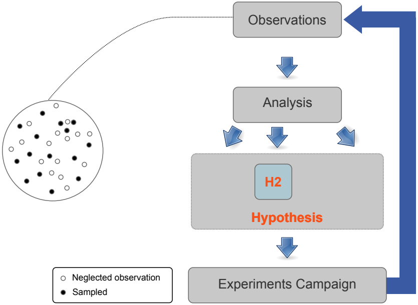
# Validity 1/2

## 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



# Validity 1/2

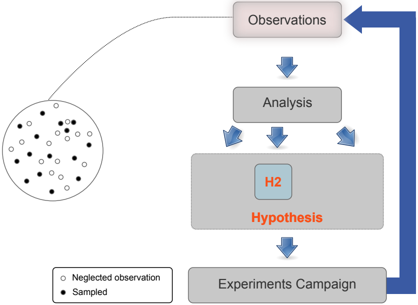
## Validation

- 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

  $\leadsto$ 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



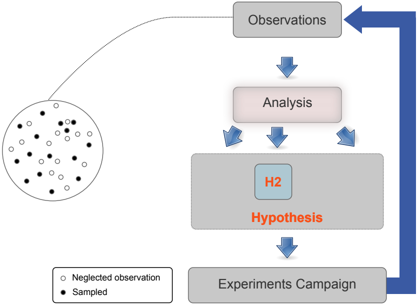
# Validity 1/2

## 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



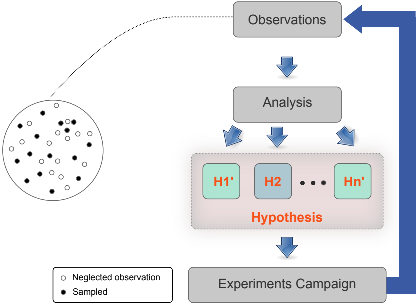
# Validity 1/2

## 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



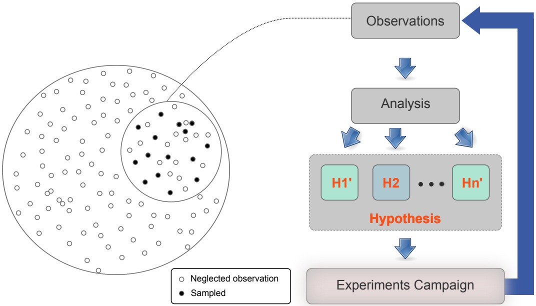
# Validity 1/2

## 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

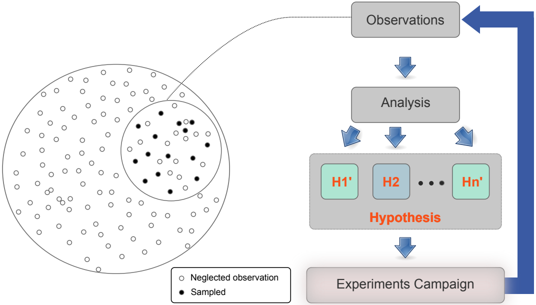


# Validity 1/2

## 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
  ⇝ 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

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

# Validity 2/2

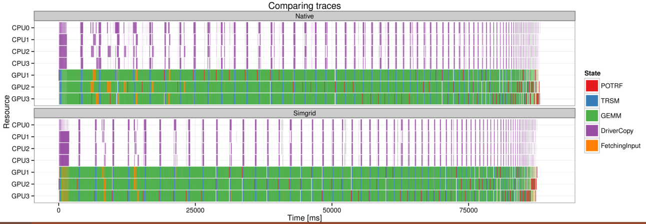
We followed this approach in P. Velho's and L. Stanisic's PhD and with A. Degomme.

- SimGrid uses a flow-level model (assume steady-state and **share bandwidth** every time a new flow appears or disappears)
  - Many bandwidth sharing mechanisms are possible (max-min fairness, proportional fairness, $\sum \arctan$, ...)

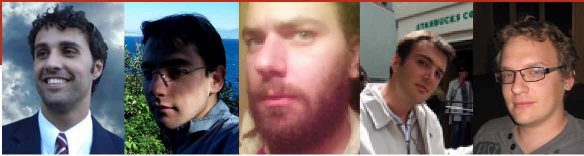
# Validity 2/2

We followed this approach in P. Velho's and L. Stanisic's PhD and with A. Degomme.

- SimGrid uses a flow-level model (assume steady-state and **share bandwidth** every time a new flow appears or disappears)
  - Many bandwidth sharing mechanisms are possible (max-min fairness, proportional fairness, $\sum$ arctan, ...)
- Invalidation with critical experiments
  - Extensive comparison with packet-level simulations and with real life
  - Bandwidth sharing models previously proposed rely on **excessive hypothesis**. Important phenomenon **not accounted for** (e.g., reverse traffic)
  - We managed to debug our models and propose reasonable ones



# Scalability



Coarse grain flow-level models are the key but they raise non classical issues:

Bandwidth sharing:

- Sparse data structures to have minimal complexity
- Cache oblivious implementation
- Partial invalidation and lazy updates
- Trace integration when possible

Platform representation:

- Hierarchical routing
- Optimized representations

Efficient Process representation: we often *emulate* code (key to validity ☺)

- Pthreads for portability but ucontexts for performance

# Future Works

**Simulation**: Shift to the HPC context

- SimGrid can be used to actually predict performances of real applications on actual platforms (SMPI/BigDFT, StarPU, ...)
- Can help capacity planning, platform qualification, runtime tuning, ...

**Visualization/Aggregation**: Meaningful visualization, comparing two traces can be particularly challenging even at small scale

- At large scale, everything remains to be invented; The knowledge obtained for simulating should help

**Reproducible Research**: Invested a lot on design of experiments, conduct of experiments, and provenance tracking


- Laboratory notebooks, literate programming
- The last articles we have published have gradually improved in term of quality ($\rightsquigarrow$ reproducible)

Thank you!