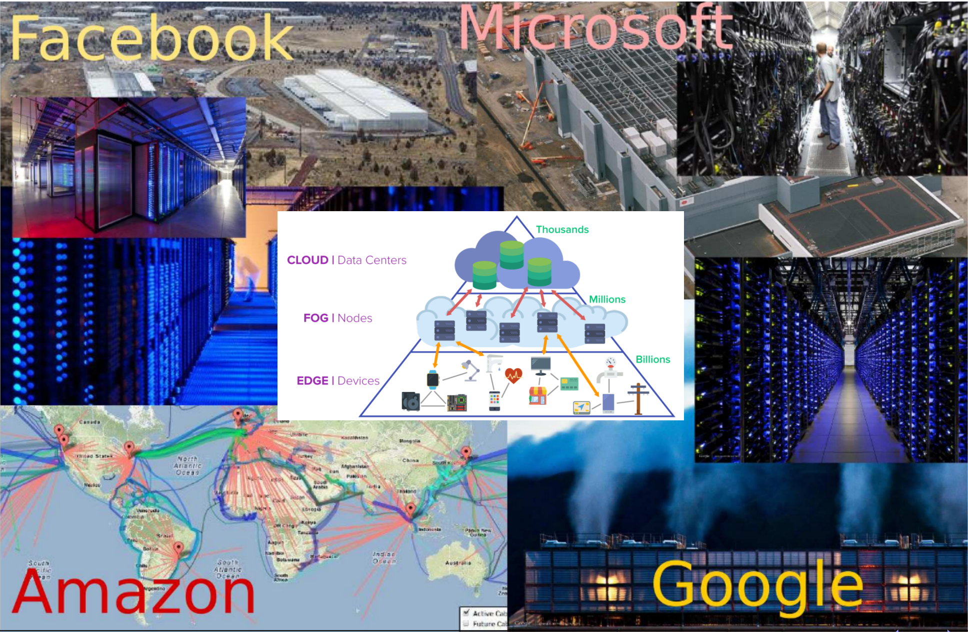
Experimenting with Modern IT Systems

Myriads Seminar, Pornichet

October 12., 2018

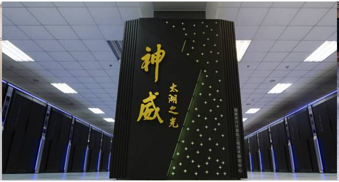
Modern IT Systems: Huge, Complex, Dynamic



Even HPC is getting Heterogeneous and Dynamic!

Huge Heterogenous Systems

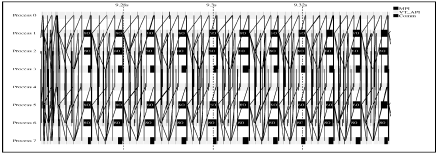
#1 Summit
2,282,544 cores
4608×(2×22-cores + 6GPU)
122 Tflops, 9MW



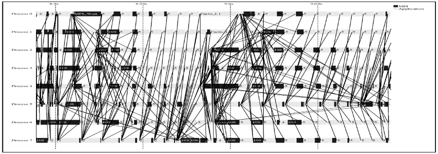
#2 Taihu Light
10,649,600 cores
40 960× 260-cores RISCs
93 Tflops, 15MW

#3 Sierra
1,572,480 cores
4300×(2×22-cores + 4GPU)
71 Tflops, 12MW

Complex Dynamic Applications



Rigid, Regular, Hand-tuned Comm Patterns



Dynamic, Irregular (task-based?)

Assessing Distributed Applications

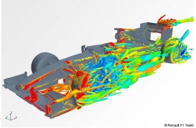
How do we study these beasts?

- Maths: System too complex to compute realistic models manually
- Experimental Facilities:
- Emulation:
- Simulation:

Assessing Distributed Applications

How do we study these beasts?

- Maths: System too complex to compute realistic models manually



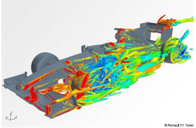
By L. Nussbaum

- Experimental Facilities: Real applications on Real platform *(in vivo)*
- Emulation: Real applications on System Models *(in vitro)*
- Simulation: Prototypes of applications on System Models *(in silico)*

Assessing Distributed Applications

How do we study these beasts?

- Maths: System too complex to compute realistic models manually



By L. Nussbaum

Myriads contributes to Experimental Methodologies and Instruments

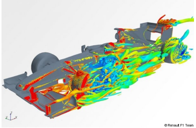
- Experimental Facilities: Real applications on Real platform *(in vivo)*
- Emulation: Real applications on System Models *(in vitro)*
- Simulation: Prototypes of applications on System Models *(in silico)*

Assessing Distributed Applications

How do we study these beasts?

- Maths: System too complex to compute realistic models manually





By L. Nussbaum

Myriads contributes to Experimental Methodologies and Instruments

- Experimental Facilities: Real applications on Real platform *(in vivo)*
- Emulation: Real applications on System Models *(in vitro)*
- Simulation: Prototypes of applications on System Models *(in silico)*

Our approach: We are Physicists

- Empirically consider large IT Systems as natural objects
  - Eminently artificial artifacts, but complexity reaches "natural" levels
  - Other sciences routinely use computers to understand complex systems
- Advanced usage of carefully designed Scientific Instruments (Grid'5000)

Remaining of this Session

This talk

- Designing and Assessing Platform Models
- What does it take to build an IT System Simulator
- (tool builder perspective)

Benjamin (after lunch)

- How to choose the simulator you need
- (power user perspective)

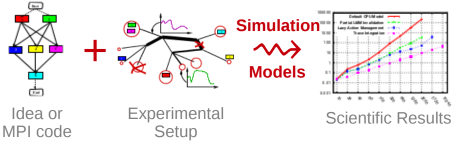
Matthieu

- How to fully use Grid'5000
- (power user perspective)

Simulating Distributed Systems

Simulation: Fastest Path from Idea to Data

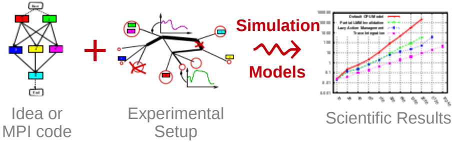
- Test your scientific idea with a fast and confortable scientific instrument



Simulating Distributed Systems

Simulation: Fastest Path from Idea to Data

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



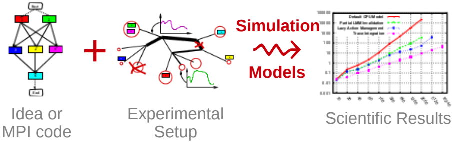
Simulation: Easiest Way to Study Real Distributed Systems

- Centralized and reproducible setup. Don't waste resources to debug and test
- No Heisenbug, full Clairevoyance, High Reproducibility, *What if* studies

Simulating Distributed Systems

Simulation: Fastest Path from Idea to Data

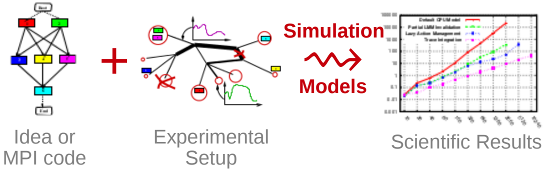
- Test your scientific idea with a fast and confortable scientific instrument

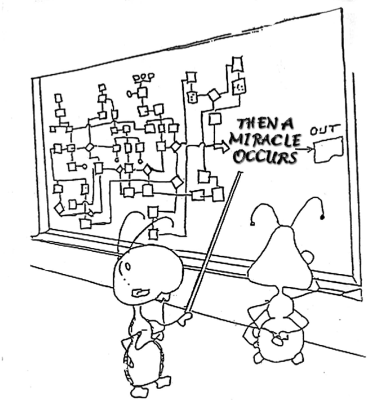


Simulation: Easiest Way to Study Real Distributed Systems

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

Methodological Challenges raised





Challenges

- Validity: Realistic results
- Scalability: Fast enough; Big enough
- Right Focus: Aligned with users concerns

Flourishing State of the Art

- Each group / student build its own tool
  - Short lived, Narrow focus, Improvable
- Some very good domain-specific tools (HPC)

SimGrid: Versatile Simulator of Distributed Apps

Install a Scientific Instrument on your Laptop

Computational Science of Computer Science

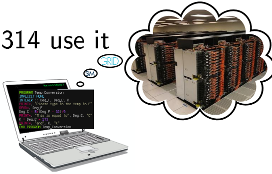
- Joint Project since 1998, mostly from French institutions
- Open Project, contributors in the USA (UHawaii, ISI, NEU), UK, Austria, Cern

Key Strengths

- Performance Models validated with Open Science ⇝ Predictive Power
- Architectured as an OS ⇝ Efficiency; Performance & Correction co-evaluation
- Usability: Fast, Reliable, User-oriented APIs, Visualization
- Versatility: Advances in HPC modeling reused by Cloud users

Community

- Scientists: 500+ publications only cite it, 58 extend it, 314 use it
- Apps/Model co-dev : StarPU, BigDFT, TomP2P
- Some industrial users on internal projects (Intel, Bull)
- Open Source: external Power Users (fixes & models)



SimGrid: Versatile Simulator of Distributed Apps

Install a Scientific Instrument on your Laptop

Computational Science of Computer Science

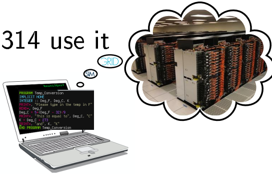
- Joint Project since 1998, mostly from French institutions
- Open Project, contributors in the USA (UHawaii, ISI, NEU), UK, Austria, Cern

Key Strengths

- Performance Models validated with Open Science $\rightsquigarrow$ Predictive Power
- Architectured as an OS $\rightsquigarrow$ Efficiency; Performance & Correction co-evaluation
- Usability: Fast, Reliable, User-oriented APIs, Visualization
- Versatility: Advances in HPC modeling reused by Cloud users

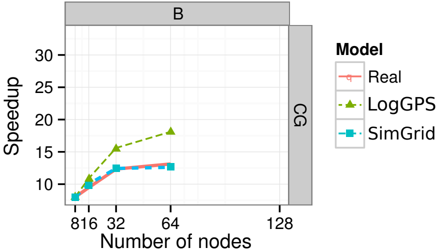
Community

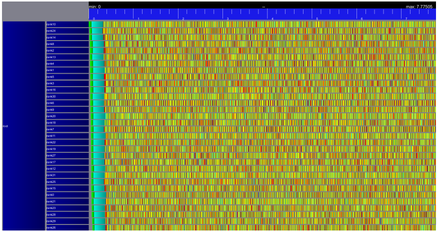
- Scientists: 500+ publications only cite it, 58 extend it, 314 use it
- Apps/Model co-dev : StarPU, BigDFT, TomP2P
- Some industrial users on internal projects (Intel, Bull)
- Open Source: external Power Users (fixes & models)



Validity Success Stories

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



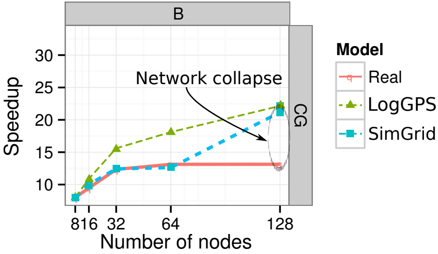


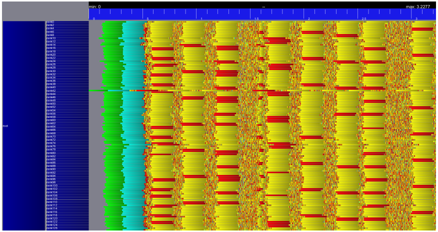
Key aspects to obtain this result

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

Validity Success Stories

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





Discrepency between Simulation and Real Experiment. Why?

- Massive switch packet drops lead to 200ms timeouts in TCP!
- Tightly coupled: the whole application hangs until timeout
- Noise easy to model in the simulator, but useless for that very study
- Discrepancy between simulated and real-world is actually a real-world problem

Have we reached the Perfect Model yet?

What is the Perfect Model anyway?

- Detailed enough to be realistic
- Efficient enough for ultra fast simulations
- Abstracted enough so that I can reason about
- In short, that's the one I could give to my students and forget about

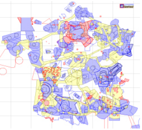
Perfect Model of France would be Perfect Map

Maps (and models) are abstractions

- Quality depends on what your usage
- More detailled $\neq$ better (not always)
- **No One True Map fitting all needs**
- Myriads of carefully adapted maps

Perfect Model of France would be Perfect Map



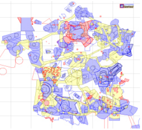



Maps (and models) are abstractions

- Quality depends on what your usage
- More detailled $\neq$ better (not always)
- **No One True Map fitting all needs**
- Myriads of carefully adapted maps

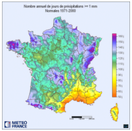
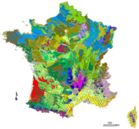
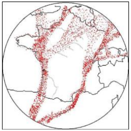
Perfect Model of France would be Perfect Map



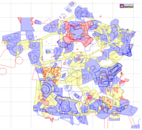
Maps (and models) are abstractions

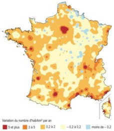
- Quality depends on what your usage
- More detailled $\neq$ better (not always)
- **No One True Map fitting all needs**
- Myriads of carefully adapted maps

Perfect Model of France would be Perfect Map

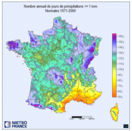


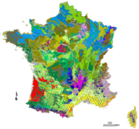
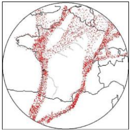



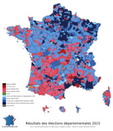
Maps (and models) are abstractions

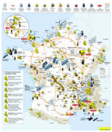
- Quality depends on what your usage
- More detailled $\neq$ better (not always)
- **No One True Map fitting all needs**
- Myriads of carefully adapted maps

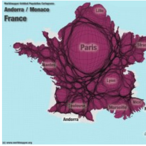






Which SimGrid Model should you use?

the one making your Study sound

If you study a theoretical P2P algorithm

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

If your study is a MPI application

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

If you work on a TCP variant

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

If your study WAN-interconnected Set Top Boxes

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

Why should you Trust SimGrid Results?

Why should you Trust SimGrid Results?

You should not.

Why should you Trust SimGrid Results?

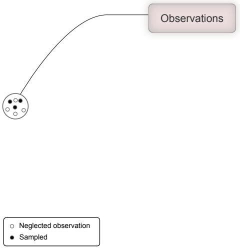
You should not.

Validating your results is not enough

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

Try to invalidate them through *crucial experiments!*

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



- Observe the System you Study
- Build hypothesis from Observations

Why should you Trust SimGrid Results?

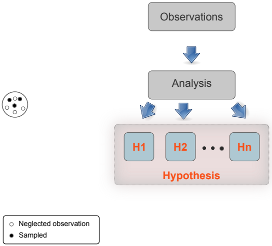
You should not.

Validating your results is not enough

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

Try to invalidate them through *crucial experiments!*

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



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

Why should you Trust SimGrid Results?

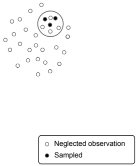
You should not.

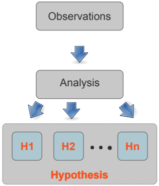
Validating your results is not enough

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

Try to invalidate them through *crucial experiments!*

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





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

Why should you Trust SimGrid Results?

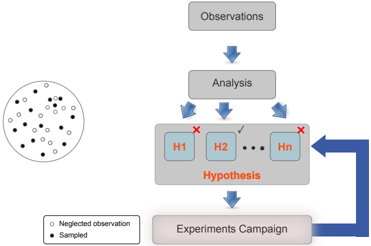
You should not.

Validating your results is not enough

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

Try to invalidate them through *crucial experiments!*

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



Cyclic Process

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

Why should you Trust SimGrid Results?

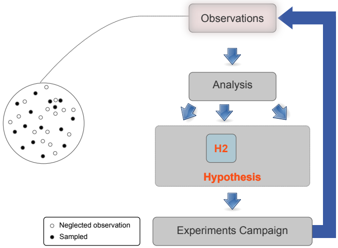
You should not.

Validating your results is not enough

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

Try to invalidate them through *crucial experiments!*

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



Cyclic Process

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

Why should you Trust SimGrid Results?

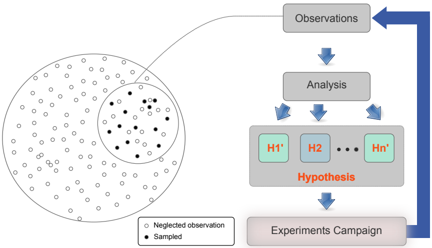
You should not.

Validating your results is not enough

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

Try to invalidate them through *crucial experiments!*

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



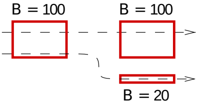
Cyclic Process

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

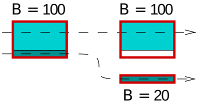
We do so in SimGrid since decades

Some Crucial Experiments

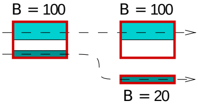
Model Limit: Heterogeneity (Narses, OptorSim, GroudSim)



Setting



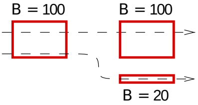
Expected Result



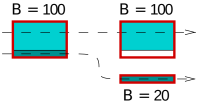
Actual Result

Some Crucial Experiments

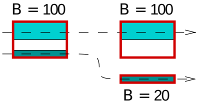
Model Limit: Heterogeneity (Narses, OptorSim, GroudSim)



Setting

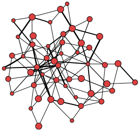


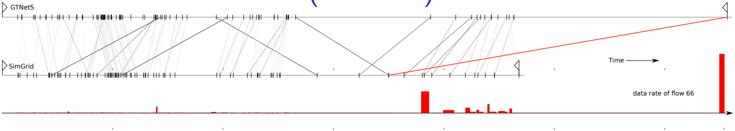
Expected Result



Actual Result

Model Limit: Platform Stress (SimGrid)

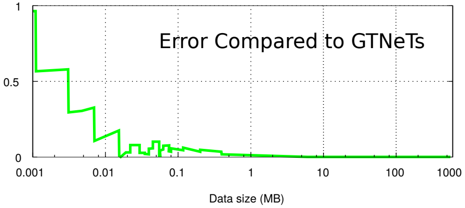




- Flow 66 terminates too early in SimGrid; seems stuck until timeout on GTNetS

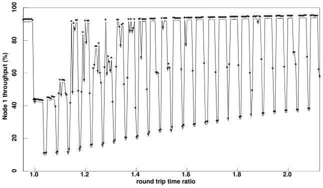
More Crucial Experiments

Model Limit: Slow Start (SimGrid without SMPI)



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

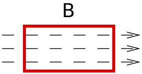
Model Limit: Phase effect (packet-level tools: NS2, NS3)



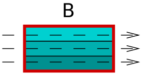
- Periodic and deterministic traffic may resonate
- First shown by Floyd and Jacobson in 1991

So, what can you Expect from SimGrid??

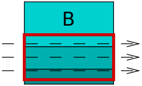
Implementation Limit: Bugs (GridSim)



Setting



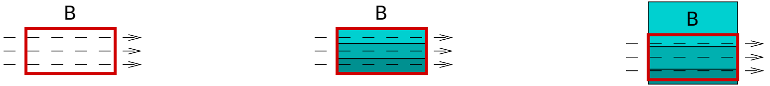
Expected Result



Actual Result

So, what can you Expect from SimGrid??

Implementation Limit: Bugs (GridSim)



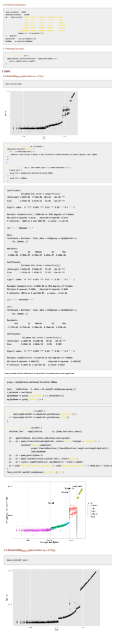
Setting Expected Result Actual Result

- Issue reported since ages, but no answer from authors
- If you (really) want to use CloudSim, prefer CloudSimPlus (better quality)

SimGrid is well Tested

- 740 integration tests, 10k units (coverage: 80%)
- Each commit: 22 configurations (4 OS, 3 compilers, 2 archs; 3 providers)
- Nightly: 2 dynamic + 2 static analyzers; StarPU, BigDFT and Proxy Apps
- Still expect bugs, but our community strive to fix them if you provide a MWE

Open Science



Tedious Experiments must be Reproducible

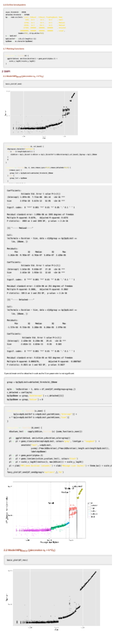
Devel in the details *vs.* Reproducibility Grail

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

But there is Hope!

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

Open Science



Tedious Experiments must be Reproducible

Devel in the details *vs.* Reproducibility Grail

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

But there is Hope!

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

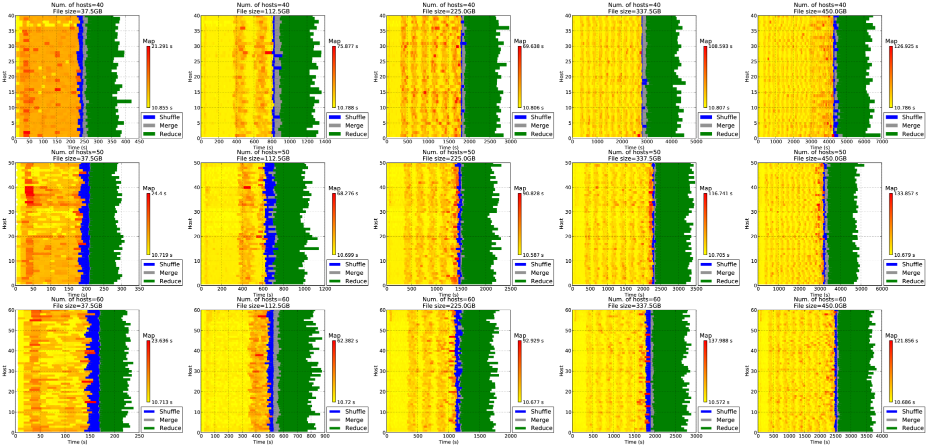
Grumpy Reviewer #3 is not convinced.

- *I found the results section of this paper to be pretty weak: previous simulators can simulate 100,000+ procs*
- Sociological problem take time to solve

A Nice Scientific Journey

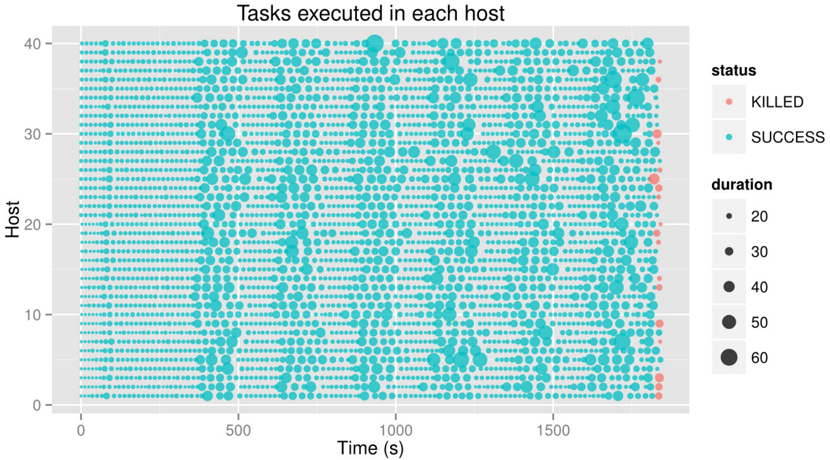
Modeling MapReduce for SimGrid from G5K experiments (2010)

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



Unexpected, annoying slowdown waves

Closer Look at this Unexpected Behavior



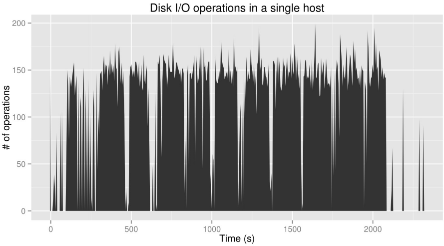
- Duration of all tasks should be roughly the same
- Many map tasks slowed down, synchronously in different hosts!

Next student next year reproducibility issues

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

Next student next year reproducibility issues

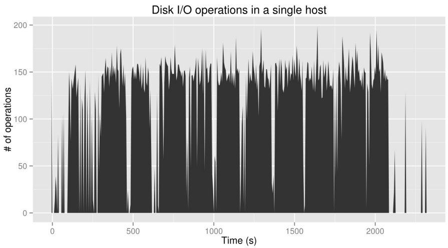
- Orsay cluster retired, impossible to rerun there
- No slowdown wave at Nancy for months. Even when changing the application
- Finally reproduced in Sophia (on similar hardware). So it's hardware.

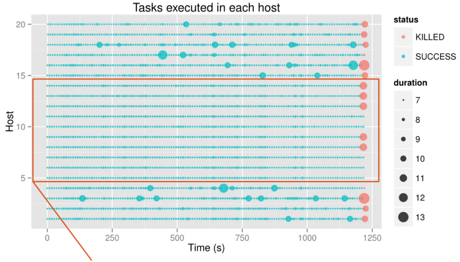


- SATA disks get saturated

Next student next year reproducibility issues

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

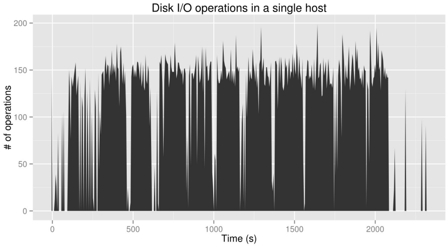


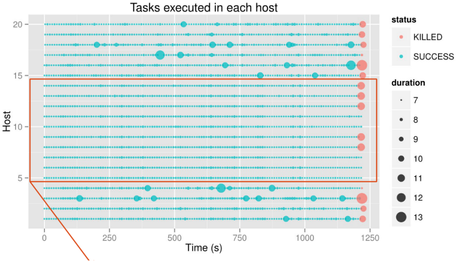


- SATA disks get saturated by Reduce, killing the machine, explaining wave
- Could fix the application or Model the phenomenon, as usual

Next student next year reproducibility issues

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





- SATA disks get saturated by Reduce, killing the machine, explaining wave
- Could fix the application or Model the phenomenon, as usual

Open Science is absolutely mandatory

- You need to righteously trust your results when the reality deceives you
- Feed your LabBook with Love, Know your tools, Ask for help from experts

SimGrid: Versatile Simulator of Distributed Apps

Install a Scientific Instrument on your Laptop

Computational Science of Computer Science

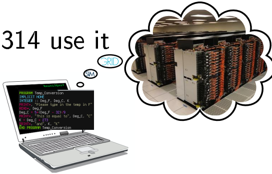
- Joint Project since 1998, mostly from French institutions
- Open Project, contributors in the USA (UHawaii, ISI, NEU), UK, Austria, Cern

Key Strengths

- Performance Models validated with Open Science ⇝ Predictive Power
- Architectured as an OS ⇝ Efficiency; Performance & Correction co-evaluation
- Usability: Fast, Reliable, User-oriented APIs, Visualization
- Versatility: Advances in HPC modeling reused by Cloud users

Community

- Scientists: 500+ publications only cite it, 58 extend it, 314 use it
- Apps/Model co-dev : StarPU, BigDFT, TomP2P
- Some industrial users on internal projects (Intel, Bull)
- Open Source: external Power Users (fixes & models)



The Many Interfaces of SimGrid

S4U: Future interface for algorithms

- Combines the power of SimGrid with the power of C++11
- Currently under development toward SimGrid 4, already usable (C++ or C)
- That's actually documented, with examples and a nice tutorial

Legacy Interfaces

- MSG: legacy interface for Concurent Sequential Processes
  - Used to be our main interface, now frozen (no further dev)
- SimDAG: legacy interface to study of centralized algorithms

SMPI: Reimplementation of MPI on top of SimGrid

- Complex in C/C++/Fortran applications *emulated* out of the box

Remote SimGrid (RSG): Toolbox to emulate your own application

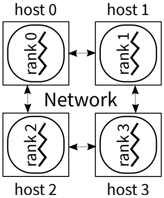
- Ongoing effort to emulate the real OpenMPI
- Ongoing effort to emulate Ceph

Virtualizing MPI Applications with SimGrid

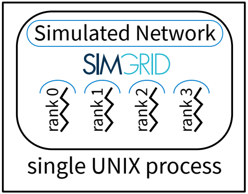
SMPI: Reimplementation of MPI on top of SimGrid

- Computations emulated; Communications simulated
- Complex C/C++/F77/F90 apps run out of the box
  - 23 out of 30 Exascale Project's proxy apps supported (others: 5 extra deps, 2 unsupported MPI calls)
- MPI 2.2 partially covered ($\approx$ 160 primitives supported)
  - No MPI-IO, MPI3 collectives, spawning ranks, ...
  - Monothreaded applications, no pthread nor OpenMP

MPI Applications are *folded into* a single process



Real Settings

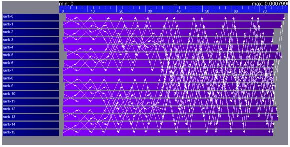
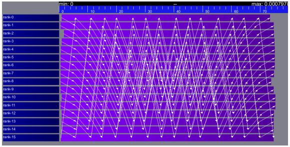
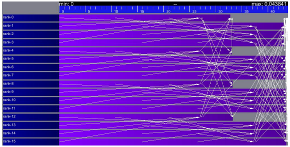


SimGrid Simulation

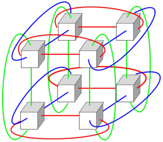
SimGrid Modeling of MPI

MPI Collectives

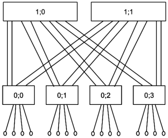
- SimGrid implements more than 120 algorithms for the 10 main MPI collectives
- Selection logic from OpenMPI, MPICH can be reproduced

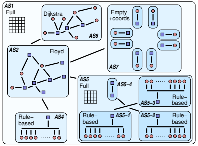
HPC Topologies



Torus



Fat-trees



Hierarchies of ASes

But also

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

From SMPI to Cloud Applications Emulation

SimGrid executes unmodified HPC applications

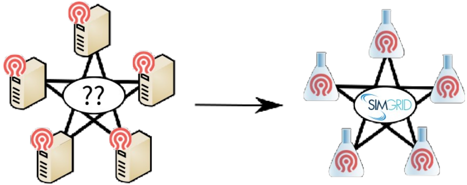
- Most of HPC applications are written with MPI (and OpenMP)
- Qualitative: reimplement MPI API on top of SimGrid
- Quantitative: 10 years of (in)validation of IT perf models for this usecase

From SMPI to Cloud Applications Emulation

SimGrid executes unmodified HPC applications

- Most of HPC applications are written with MPI (and OpenMP)
- Qualitative: reimplement MPI API on top of SimGrid
- Quantitative: 10 years of (in)validation of IT perf models for this usecase

How to do the same for Clouds?

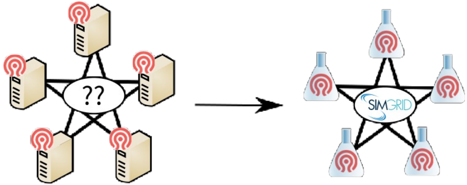


From SMPI to Cloud Applications Emulation

SimGrid executes unmodified HPC applications

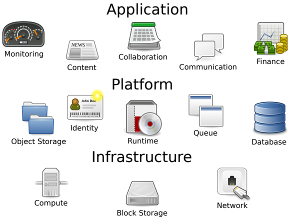
- Most of HPC applications are written with MPI (and OpenMP)
- Qualitative: reimplement MPI API on top of SimGrid
- Quantitative: 10 years of (in)validation of IT perf models for this usecase

How to do the same for Clouds?



No nice interception point

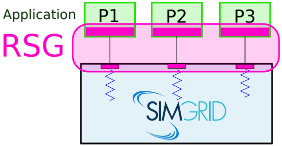
- Communications using BSD socket and HTTP
  - Low-level Interception ⇒ tedious predictions
- Many different apps and runtimes
  - Manual modifs both tedious and unavoidable?



Remote SimGrid (RSG)

In a Nutshell

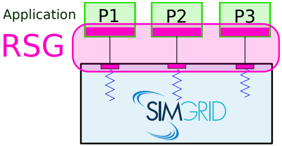
- Library simplifying the interception of applications
- Code in app + Code in SimGrid + Comms
- Require to modify the application



Remote SimGrid (RSG)

In a Nutshell

- Library simplifying the interception of applications
- Code in app + Code in SimGrid + Comms
- Require to modify the application

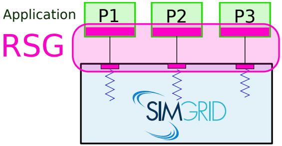


Does it work?

- 2015: Internship to intercept Cassandra (Aspect Oriented Prog in Java)
- 2016: Internship to intercept Slurm (niveau BSD)
- 2017-2018: Postdoc to intercept OpenMPI (specific driver)

Remote SimGrid (RSG)

In a Nutshell



- Library simplifying the interception of applications
- Code in app + Code in SimGrid + Comms
- Require to modify the application

Does it work?

- 2015: Internship to intercept Cassandra (Aspect Oriented Prog in Java)
- 2016: Internship to intercept Slurm (niveau BSD)
- 2017-2018: Postdoc to intercept OpenMPI (specific driver)

Ongoing: Interception of the Ceph application

- C++ Project, nice architecture for this modification (by accident)
- Highly visible project in Cloud world, huge community
- Many debugging and profiling concerns, in both R&D
- Later on: validating performance models on this kind of workload

SimGrid: Versatile Simulator of Distributed Apps

Install a Scientific Instrument on your Laptop

Computational Science of Computer Science

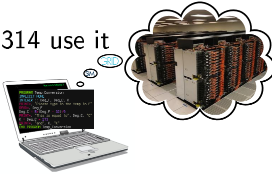
- Joint Project since 1998, mostly from French institutions
- Open Project, contributors in the USA (UHawaii, ISI, NEU), UK, Austria, Cern

Key Strengths

- Performance Models validated with Open Science $\rightsquigarrow$ Predictive Power
- Architectured as an OS $\rightsquigarrow$ Efficiency; Performance & Correction co-evaluation
- Usability: Fast, Reliable, User-oriented APIs, Visualization
- Versatility: Advances in HPC modeling reused by Cloud users

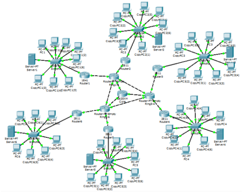
Community

- Scientists: 500+ publications only cite it, 58 extend it, 314 use it
- Apps/Model co-dev : StarPU, BigDFT, TomP2P
- Some industrial users on internal projects (Intel, Bull)
- Open Source: external Power Users (fixes & models)

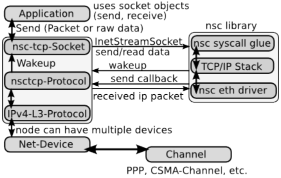


Classical Network Models: Hands and Feets

Packet-level Simulators

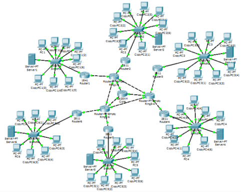


- ▶ Full network stack
- ☺ Very detailled
- ☹ Hard to instanciate
- ☹ Very slow
- ☹ Hard to reason about

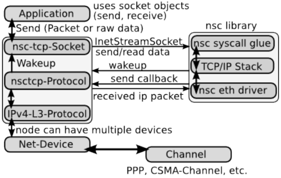


Classical Network Models: Hands and Feets

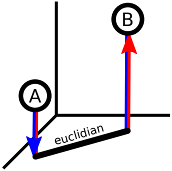
Packet-level Simulators



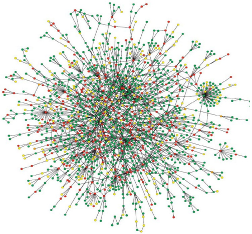
- Full network stack
- ☺ Very detailled
- ☹ Hard to instanciate
- ☹ Very slow
- ☹ Hard to reason about



Simplistic Models

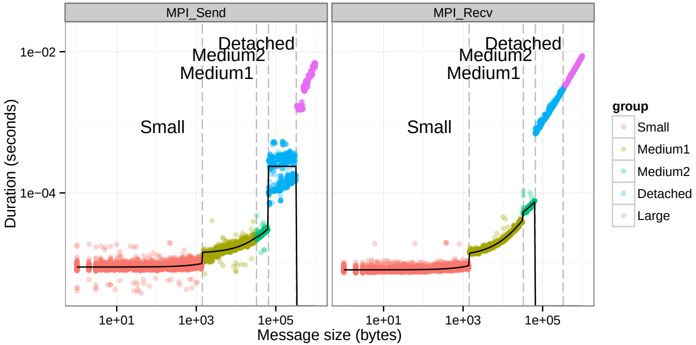


- Constant/Random delay
- N-d coordinates
- ☺ Very scalable
- ☹ No topology
- ☹ *No network congestion*



MPI Point-to-Point Model

Real Measurements (OpenMPI/TCP/Eth1GB)



- Important variability $\rightsquigarrow$ tedious, randomized experiments

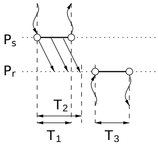
SMPI Model

- Piece-wise linear model (4 discontinuous modes)
- Automatic Calibration $\rightsquigarrow$ Vivid Research (collab with Grenoble)

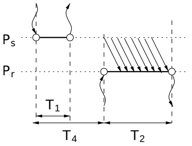
SimGrid Hybrid Network Model

LogP (small message sizes)

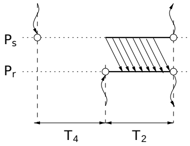
- Accounts for delay, communication modes and protocol switches



Asynchronous ($k \leqslant S_a$)



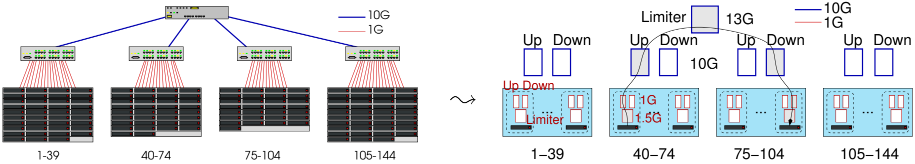
Detached ($S_a < k \leqslant S_d$)



Synchronous ($k > S_d$)

Fluid Model (large sizes)

- Accounts for contention and network topology



SimGrid: Versatile Simulator of Distributed Apps

Install a Scientific Instrument on your Laptop

Computational Science of Computer Science

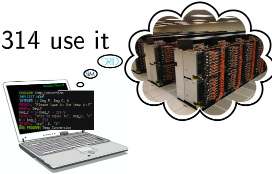
- Joint Project since 1998, mostly from French institutions
- Open Project, contributors in the USA (UHawaii, ISI, NEU), UK, Austria, Cern

Key Strengths

- Performance Models validated with Open Science $\rightsquigarrow$ Predictive Power
- Architectured as an OS $\rightsquigarrow$ Efficiency; Performance & Correction co-evaluation
- Usability: Fast, Reliable, User-oriented APIs, Visualization
- Versatility: Advances in HPC modeling reused by Cloud users

Community

- Scientists: 500+ publications only cite it, 58 extend it, 314 use it
- Apps/Model co-dev : StarPU, BigDFT, TomP2P
- Some industrial users on internal projects (Intel, Bull)
- Open Source: external Power Users (fixes & models)



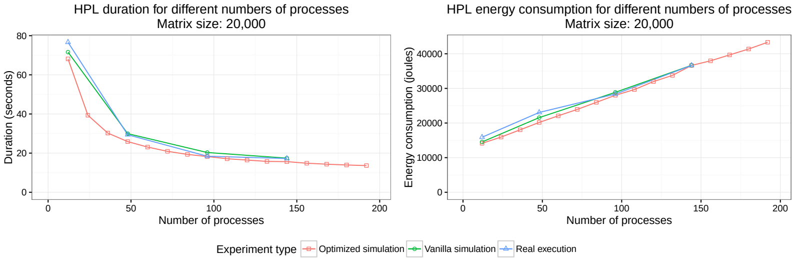
HPL and the Top500

Context

- Real execution (qualification benchmark)
  - Matrix of rank 3,875,000: ≈ 120 Terabytes
  - 6,006 MPI processes for 2 hours: 500 CPU-days
- Simulation/Emulation with SMPI
  - 1 Xeon E5-2620 server (Nova, Grid'5000)
  - ≈ 47 hours and 16GB
  - Modified HPL (abstract compute kernels, factorize malloc)

Stampede, U.S.A., #20 with ≈ 5 Pflops
56 Gbit/s FDR InfiniBand Fat tree topology
6,400 × (8 cores + 1 Xeon Phi)

Accuracy (Evaluation on Taurus (Grid'5000))



Mismatch with the Stampede qualification run (Intel HPL vs. Open-Source HPL)

Perspective Capacity planning, Tune real applications, Co-Design, ...

SimGrid: Versatile Simulator of Distributed Apps

Install a Scientific Instrument on your Laptop

Computational Science of Computer Science

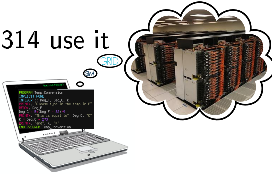
- Joint Project since 1998, mostly from French institutions
- Open Project, contributors in the USA (UHawaii, ISI, NEU), UK, Austria, Cern

Key Strengths

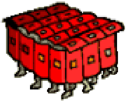
- Performance Models validated with Open Science ⇝ Predictive Power
- Architectured as an OS ⇝ Efficiency; Performance & Correction co-evaluation
- Usability: Fast, Reliable, User-oriented APIs, Visualization
- Versatility: Advances in HPC modeling reused by Cloud users

Community

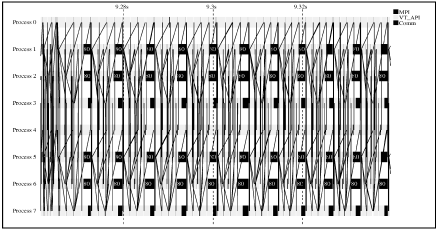
- Scientists: 500+ publications only cite it, 58 extend it, 314 use it
- Apps/Model co-dev : StarPU, BigDFT, TomP2P
- Some industrial users on internal projects (Intel, Bull)
- Open Source: external Power Users (fixes & models)



Writting Correct Distributed Applications

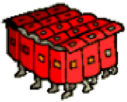


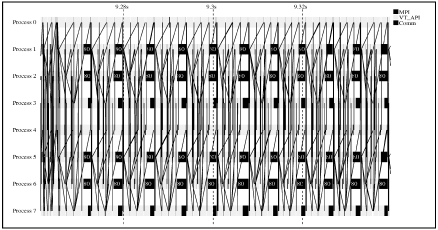
- Classical Solution: Proof of algorithms
- Pessimistic Solution: Lower performance expectations
- Optimistic Solution: Eventually Consistent
- HPC Solution: Rigid, Regular, Hand-tuned Communication Patterns

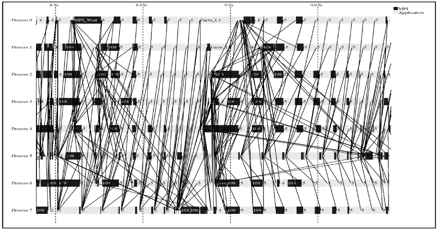


Writting Correct Distributed Applications

- Classical Solution: Proof of algorithms
- Pessimistic Solution: Lower performance expectations
- Optimistic Solution: Eventually Consistent
- HPC Solution: Rigid, Regular, Hand-tuned Communication Patterns
- Large-Scale Hybrid Machines: Dynamic, Irregular (task-based?)

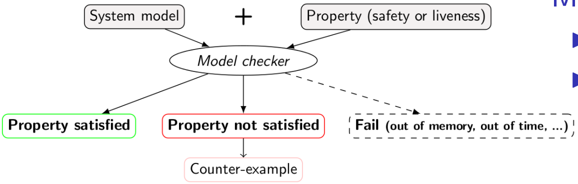






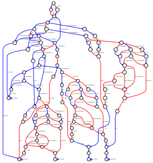
Verification: must explore **all possible execution paths**

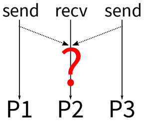
Formal Methods in Mc SimGrid



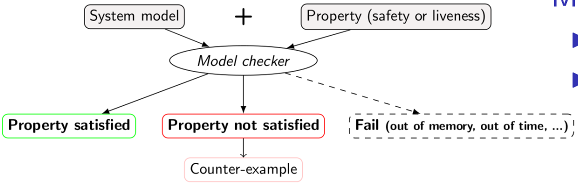
Model Checking

- Exhaustively search for faults
- Requires an accurate model





Formal Methods in Mc SimGrid

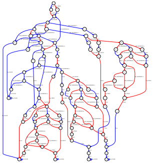


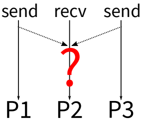
Model Checking

- Exhaustively search for faults
- Requires an accurate model

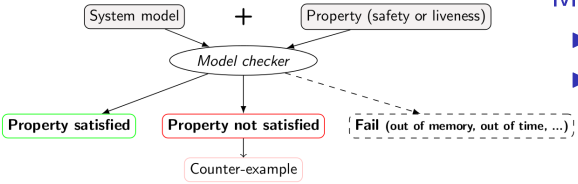
Dynamic Verification: similar idea, applied to source code

- Mc SimGrid: Live, virtualized execution
  No static analysis (yet), no symbolic execution
- On Indecision Points: checkpoint, explore, rollback





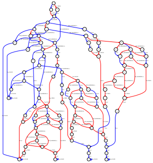
Formal Methods in Mc SimGrid



Model Checking

- Exhaustively search for faults
- Requires an accurate model

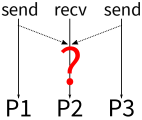
Dynamic Verification: similar idea, applied to source code



- Mc SimGrid: Live, virtualized execution
  No static analysis (yet), no symbolic execution
- On Indecision Points: checkpoint, explore, rollback

Execution Model in Mc SimGrid

- Mono-threaded MPI applications (CSP)
- Point-to-Point semantic: Configurable (paranoid / permissive)
- Collective semantic: Implementations of MPICH3, OpenMPI



Mc SimGrid Overview

Mc SimGrid: Dynamic Verification of MPI applications

- Unmodified C/C++/Fortran MPI applications
- Early stage, but already functional: Safety, Liveness, Send-determinism
- Reductions: DPOR and State Equality
- Scale to a few processes only, but exhaustive testing

State of the Art

- Many testing tools (MUST): not exhaustive nor sound
- Symbolic execution (TASS, CIVL): complementary to our work
- Dynamic verification (ISP, DAMPI at U. Utah)
  - PMPI proxy at runtime to delay communications to guide execution
  - Works for safety, but not applicable to liveness (state equality)

Ongoing Works

- Improve DPOR by using Event Unfolding structures (IPL PhD)
- Convert checkpoints taken on OpenMPI into SimGrid runs (IPL Post-doc)
- Static Analysis to improve Dynamic State Equality Detection (IPL collab)

SimGrid: Versatile Simulator of Distributed Apps

Install a Scientific Instrument on your Laptop

Computational Science of Computer Science

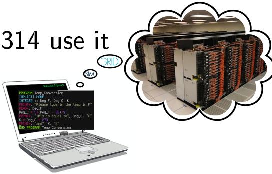
- Joint Project since 1998, mostly from French institutions
- Open Project, contributors in the USA (UHawaii, ISI, NEU), UK, Austria, Cern

Key Strengths

- Performance Models validated with Open Science $\leadsto$ Predictive Power
- Architectured as an OS $\leadsto$ Efficiency; Performance & Correction co-evaluation
- Usability: Fast, Reliable, User-oriented APIs, Visualization
- Versatility: Advances in HPC modeling reused by Cloud users

Community

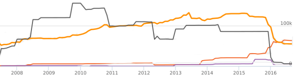
- Scientists: 500+ publications only cite it, 58 extend it, 314 use it
- Apps/Model co-dev : StarPU, BigDFT, TomP2P
- Some industrial users on internal projects (Intel, Bull)
- Open Source: external Power Users (fixes & models)

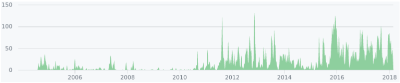


Technical Considerations

Complex and Dynamic Code Base

- Only 100k sloc, but complex due to versatile efficiency + formal verification
- Implemented in C++/C (+ assembly); Bindings: Java, Lua and Fortran
- Active project: commits every day by ≈ 6 commiters, 4 releases a year
- Ongoing full rewrite in C++ along with *Release soon, Release often*

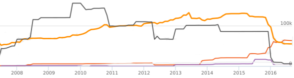


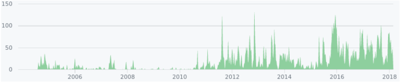


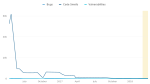
Technical Considerations

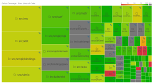
Complex and Dynamic Code Base

- Only 100k sloc, but complex due to versatile efficiency + formal verification
- Implemented in C++/C (+ assembly); Bindings: Java, Lua and Fortran
- Active project: commits every day by $\approx$ 6 commiters, 4 releases a year
- Ongoing full rewrite in C++ along with *Release soon, Release often*









Well Tested

- 740 integration tests, 10k units (coverage: 80%)
- Each commit: 22 configurations (4 OS, 3 compilers, 2 archs; 3 providers)
- Nightly: 2 dynamic + 2 static analyzers; StarPU, BigDFT and Proxy Apps

- We cultivate our garden: simplify to grow further

The SimGrid Community

http://simgrid.org simgrid-user@lists.gforge.inria.fr

Communication and Animation

- SimGrid User Days: Welcome newcomers & Take feedback since 2010
- 500 cite 300 use 60 extend; 30 mails/month; 5 bugs/month; Stack Overflow

Preliminary Industrial Contacts

- CERN: test the LHC DataGrid before production (since years)
- Intel: internal project to address a call from KAUST on co-design
- Octo: dimensionning Ceph infrastructures for their clients (attempt)
- Bull: sometimes used internally, but not officially yet :)

Training and User Support in Computing Centers

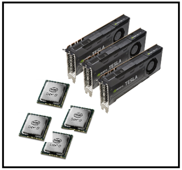
- Training @TACC: Victor Eijkhout is porting his book to SMPI
- @MPI Computing & Data Facilities: Profile some apps with SMPI

Toward Education

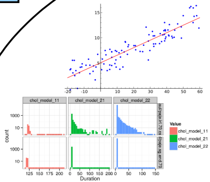
- Teach now the researchers and engineers of tomorrow to SimGrid
- Done: SMPI CourseWare, PeerSimGrid; Ongoing: Cloud, Wrench and more?

StarPU-Simgrid Overview

Calibration



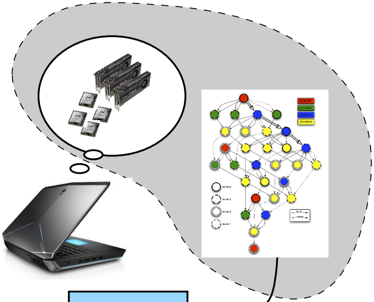
StarPU



Performance Profile

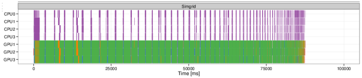
Run once!

Simulation



StarPU

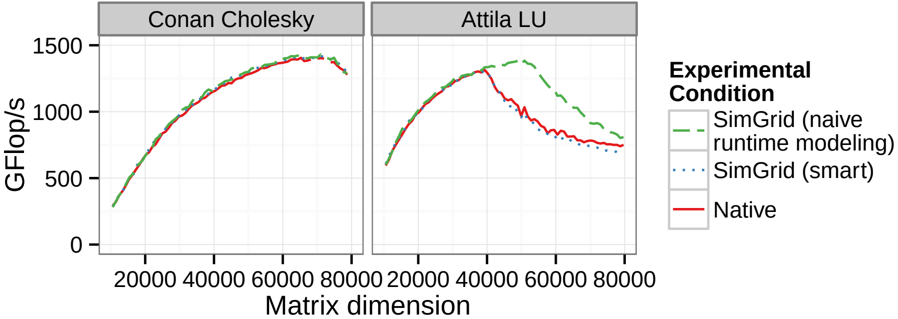
SimGrid



Quickly Simulate Many Times

StarPU-Simgrid on dense linear algebra

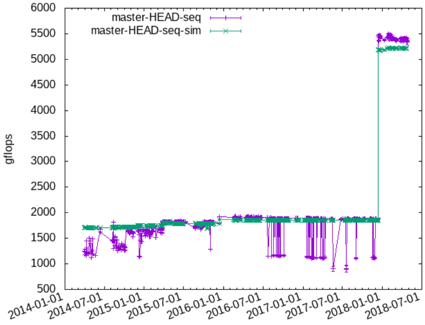
- Accurate simulated time results
- Already required a lot of care
- Extensively used for scheduling research



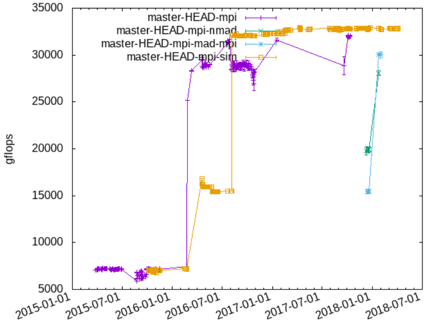
Continuous Integration of StarPU using SimGrid

Nightly build since several years

- Compare native and simulated execution as a CI process
  - Runs on sirocco nodes on Grid'5000: 1 CPU (12 cores) + 3 GPUs (K40M)
- Very successful
  - Satisfying prediction (even on HW upgrade), at least gets the trends
  - Real executions noisy and hard to deal with



Sequential (CPU+GPU)



MPI (4 nodes)

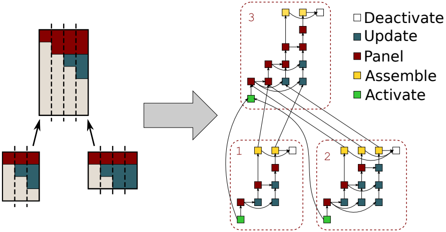
StarPU Visualization

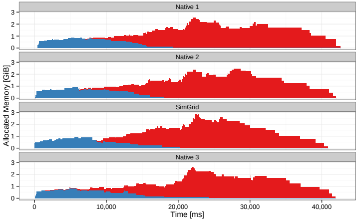
Get data without Heisenbug, analyze it with R

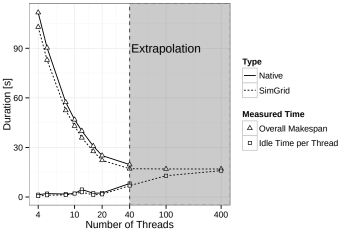
StarPU QR-Mumps

QR-MUMPS multi-frontal sparse factorization on top of StarPU

- Tree parallelism
- Node parallelism
- Variable matrix geometry
- Fully dynamic scheduling w. StarPU





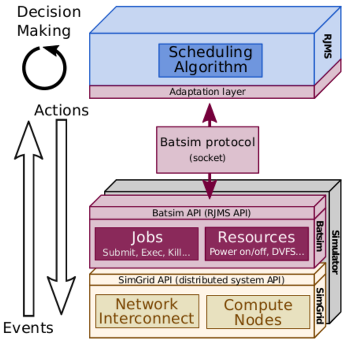


Perspective Tune app. and scheduler, capacity (memory) planning

BatSim

A Job and Resource Management System Simulator

- A key component in HPC systems
- Decouple the decision making from the simulation
- Uses SimGrid as a backend



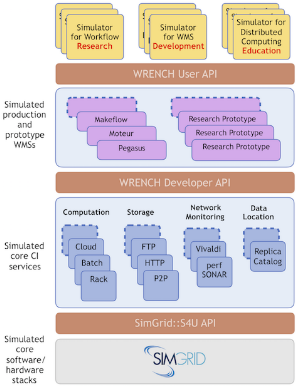
- Developed in the Datamove team (Grenoble)
- `https://github.com/oar-team/batsim`

Wrench

A Workflow Management System Simulation Workbench

- Objective
  - Provide high-level building blocks for developing custom simulators
- Targets:
  - Scientists: make quick and informed choices when executing workflows
  - Software developers: implement more efficient software infrastructures to support workflows
  - Researchers: Develop novel efficient algorithms
- Coupled with BatSim
- `http://wrench-project.org`
  - Collaboration with ISI/USC and UH Manoa
  - Funded by the NSF (grants number 1642369 and 1642335) and CNRS (PICS 7239)

WRENCH



SimGrid with TomP2P

- TomP2P is a Java-based DHT that stores key-value pairs
- Goals of using SimGrid
  - Difficult to run more than ~5K peers
  - Difficult to simulate network, TomP2P peers run on same machine

  → The goal to use SimGrid was to have an easy way to simulate many peers in a network scenario
- What to expect from SimGrid
  - Not having to implement a simulation framework
  - Faster verification if new algorithm works in a large-scale network
- Feedback from using SimGrid and the Java-bindings

  \- Threading is done by SimGrid (needed rework in TomP2P)

  \+ Good documentation and examples

  \+ Active community

ifi

(Courtesy of Thomas Bocek)

SimGrid: Versatile Simulator of Distributed Apps

Install a Scientific Instrument on your Laptop

Computational Science of Computer Science

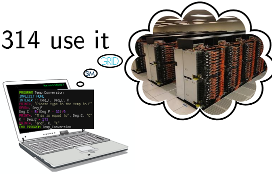
- Joint Project since 1998, mostly from French institutions
- Open Project, contributors in the USA (UHawaii, ISI, NEU), UK, Austria, Cern

Key Strengths

- Performance Models validated with Open Science $\rightsquigarrow$ Predictive Power
- Architectured as an OS $\rightsquigarrow$ Efficiency; Performance & Correction co-evaluation
- Usability: Fast, Reliable, User-oriented APIs, Visualization
- Versatility: Advances in HPC modeling reused by Cloud users

Community

- Scientists: 500+ publications only cite it, 58 extend it, 314 use it
- Apps/Model co-dev : StarPU, BigDFT, TomP2P
- Some industrial users on internal projects (Intel, Bull)
- Open Source: external Power Users (fixes & models)



Future Research Directions for SimGrid

Better Interfaces and Tooling

- Domain-specific API for the Cloud and IoT platforms
- Semulation: Study real arbitrary applications with SimGrid
- Switching between Execution, Simulation and Verification within a run
- Online Simulation of Distributed Infrastructures

Better Models

- Co-simulation of Smart Grids: IT and energy

Formal Verification

- More usecases (larger ones) for both Safety and Liveness
- Domain-specific exploration and reduction technics (Star-PU)
- Domain-specific properties (QoS as a fairness?)

Build a Sustainable Community

- Production ready, toward Industry and Education (for engineers)

IPL HAC-SPECIS (2016-2020)

Inria Project Lab $\approx$ 1 postdoc and 1 PhD student per year for 3-4 years

Project Partners

8 Inria Teams (verification^{+}, performance evaluation$^{\triangle}$, HPC*) + CEA*

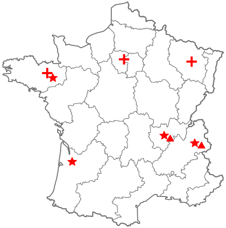
Rhône Alpes: AVALON$^{\star\ \triangle}$, <u>POLARIS</u>$^{\star\ \triangle}$ + CEA*

Rennes: MYRIADS$^{\star\ +}$, SUMO^{+}

Bordeaux: HIEPACS*, STORM*

Paris: MEXICO^{+}

Nancy VERIDIS^{+}



Context and Objectives

- Rigid communication patterns are not scalable enough:
- HPC apps become adaptive, lock-free, with complex optimizations/scheduling

- **Research Question:** Joint Study of Performance **AND** Correctness
- Goal: *bridge the gap between communities*

What Kind of Properties can be Verified?

Safety Properties: "A given bad behavior never occurs"

- e.g.: any assertion (x != 0, no deadlock)
- Verified on each state separately
- Counter example: a faulty state

Liveness Properties: "An expected behavior will happen in all cases"

- e.g.: Any request will eventually be fulfilled; No non-progression cycle
- Verified on a full execution path
- Counter example: a cycling execution path that violates the property

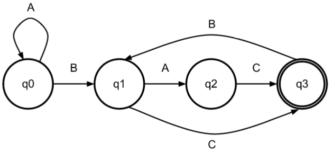
Comm Patterns: "It exists a pattern that is the same for all exec paths"

- e.g.: send-deterministic (local sending order is always the same)
- Work on all execution paths
- Counter examples: two paths exhibiting differing communication patterns

Checking Liveness Properties

Enforce property ϕ

- Search for a counter-example, ie a run of the system satisfying $\neg\phi$
- Counter examples are infinite $\rightsquigarrow$ Build the Büchi Automaton of $\neg\phi$



- Ensure that Application $\times$ Bucchi($\neg\phi$) is empty (no accepted run)
- State Equality is crucial to detect cycles

Current state in Mc SimGrid

- Working in our tests (although fragile: equality is based on heuristics)
- We are looking for more domain-specific interesting properties

Verification of Protocol-wide Properties

Motivation

- Clever checkpoint algorithms exist, provided that the application is nice enough
- *On communication determinism in parallel HPC applications*, F. Cappello, A. Guermouche and M. Snir (2010)
  - Manual inspection of 27 HPC applications, seeking for such properties

Protocol-wide properties

- deterministic: On each node, send and receive events are always in same order
- send deterministic: ∀ node, send are always the same, no matter the recv order
- Not liveness, not even LTL: quantifies for all execution paths within property

Status report: we can verify such properties in Mc SimGrid

- Explore one path to learn the communication order, deduce the property
- Enforce that this order holds on all other execution path
- We reproduced the conclusions of previous paper on several benchmarks
  - NAS Parallel Benchmarks NPB 3.3 (5 kernels)
  - CORAL Benchmark codes
  - NERSC-8/Trinity Benchmarks

Mitigating the State Space Explosion

The exploration process often fails to complete

- Too many states to explore, not enough time and/or memory
- Mc SimGrid provides two reductions techniques

Dynamic Partial Ordering Reduction (DPOR)

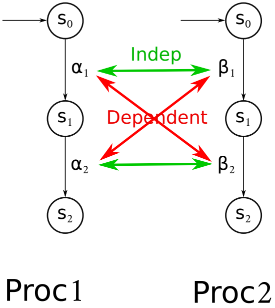
- Avoid re-exploring equivalent interleavings
- Don't explore all interleavings of local executions: they are equivalent

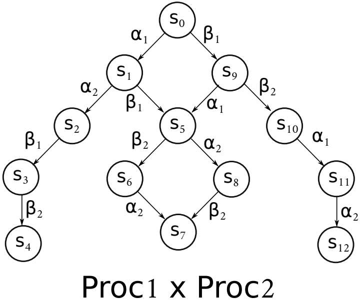
System-Level State Equality

- Detect when a given state was previously explored

Partial Ordering Reduction (DPOR)

- Avoid re-exploring Mazurkiewicz traces (don't permute independent events)





- McSimGrid: iSend and iRecv are independent, etc.
- Dynamic Partial Ordering Reductions take advantage of runtime knowledge
- Many techniques (sleep sets, ample sets) are hard to understand & get right
- Ongoing work: reimplement our DPOR using Event Unfolding Structures

But what are the transitions in Mc SimGrid?

Transition = atomic block of code between Indecision Points

- Test all interleavings of the shared state (mem+network) modifications
- Transition = (some local code +) **one** shared state's change

But what are the transitions in Mc SimGrid?

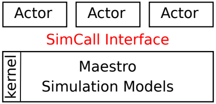
Transition = atomic block of code between Indecision Points

- Test all interleavings of the shared state (mem+network) modifications
- Transition = (some local code +) **one** shared state's change

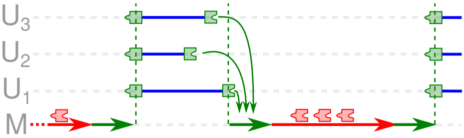
Implementation: SimGrid is an Operating System

- Actors must use simcalls to modify the shared state
- First introduced for parallel simulation, but crucial to dynamic verification

Functional View

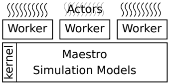


Temporal View

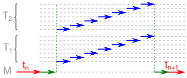


Going parallel

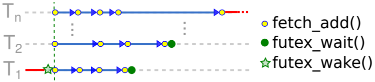
- More actors than cores $\rightsquigarrow$ Worker Threads that execute co-routines



Functional View



Temporal View



Ideal Algorithm

Mitigating the State Space Explosion

The exploration process often fails to complete

- Too many states to explore, not enough time and/or memory
- Mc SimGrid provides two reductions techniques

Dynamic Partial Ordering Reduction (DPOR)

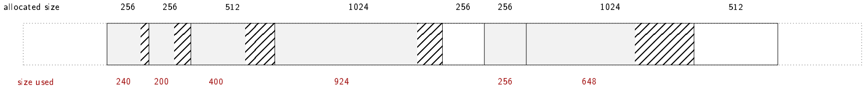
- Avoid re-exploring equivalent interleavings
- Don't explore all interleavings of local executions: they are equivalent

System-Level State Equality

- Detect when a given state was previously explored
- Introspect the application state similarly to gdb
- Also with Memory Compaction

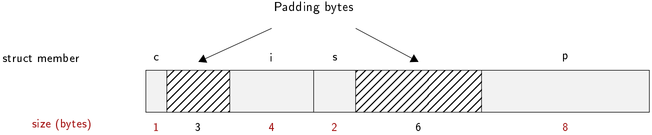
OS-level State Equality Detection

- Memory over-provisioning

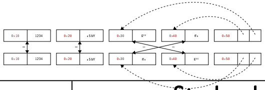


- Padding bytes: Data structure alignment

```
struct foo {
char c;
int i;
short s;
void *p;
}
```



- Irrelevant differences: system-level PID, fd, ...
- Syntactic differences / semantic equalities:



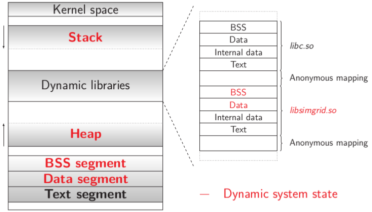
Solutions

Issue	Heap solution	Stack solution
Overprovisioning	`memset` 0 (customized `mmalloc`)	Stack pointer detection
Padding bytes	`memset` 0 (customized `mmalloc`)	DWARF + libunwind
Irrelevant differences	Ignore explicit areas	DWARF + libunwind + ignore
Syntactic differences	Heuristic for semantic comparison	N/A (sequential access)

Applicative State in Mc SimGrid

We work at system level

- Target = legacy MPI apps
- Stack: where maestro lives
- Heap: shared between actors + actors stacks
- BSS+Data: private copy for each actor
- Network state is within libsimgrid data

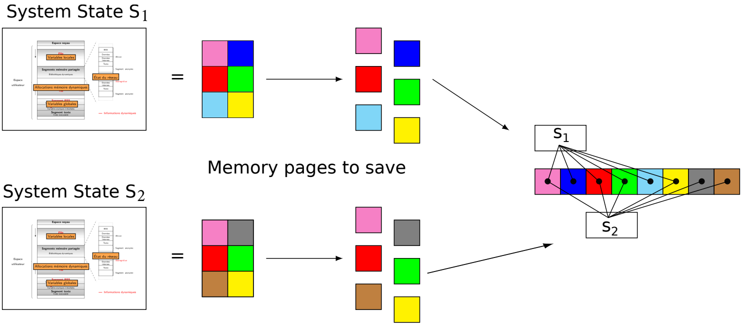


How to privatize the BSS+data

- (this is required to fold MPI processes anyway)
- Source-to-Source: turn globals into arrays of locals
- Compiler's pass: move globals into TLS area
  changes toolchain (no `icc`) $\leadsto$ alters SEBs (as any previous solution)
- GOT injection: rewrite the ELF symbol table when switching contextes
  static variables are not part of the GOT unfortunately
- mmap of bss+data segments: preserves SEBs but forces sequential exec
- dlopen tricks: compile app with -fPIE, dlopen() it many times

Memory Compactions

We save literally thousands of states



- Very few modification between states in practice
- First fast hash function to distinguish new pages, then byte-wise equality
- Combines nicely with State Equality Detection (but complex implementation)

Evaluation

Verified small applications

- MPI2 collectives, MPICH3 test suite, Benchmarks (NAS, CORAL, NERSC)
- Safety, Liveness (no non-progressive cycle), Send-determinism

Results

- Without reduction, only scales up to 2 to 6 processes in 24h
- Reductions (when usable) and Memory Compaction goes a bit further
- Not exactly ExaScale, but exhaustively at small size already useful

Found bugs

- The one we intentionally added to the code
- Our own implementation of the Chord protocol (not in MPI)
- But no wild bugs in MPI yet :(

Verification of some MPICH3 unit tests

- Looking for assertion failures, deadlocks and non-progressive cycles
- Exhaustive exploration, but no error found
- $\approx$ 1300 LOCs (per test) – State snapshot size: $\approx$ 4MB

Application	**#P**	**Stateless exploration**		**Stateful exploration**		
		# States	**Time**	**# States**	**Time**	**Memory**
sendrecv2	2	> 55 millions	> 6h	936	13s	2GB
	5	-	-	2 284	43s	5.4GB
	10	-	-	3 882	2m	11GB
bcastzerotype	5	> 12 millions	> 1h	2 474	41s	3.1GB
	6	-	-	17 525	5m	19GB
coll4	4	> 100 millions	> 24h	29 973	20m	38GB
	5	-	-	> 150 000	> 4h	> 200GB
groupcreate	5	> 10 millions	> 1h30	2 217	38s	2.8GB
	7	-	-	71 280	24m	62GB
dup	4	> 57 millions	> 5h	4 827	1m20	6.5GB
	5	-	-	75 570	49m	87GB

- We verified several MPI2 collectives too: all good so far ☹