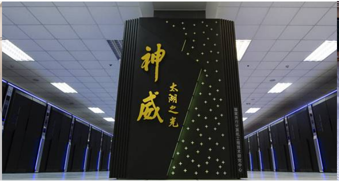
The SimGrid Framework
Computationale Science of IT Systems

Joint work of the SimGrid team over 20 years.

June 27., 2018

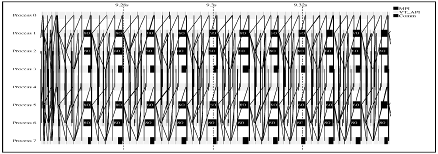
Modern Large Scale Distributed Systems

Huge Systems

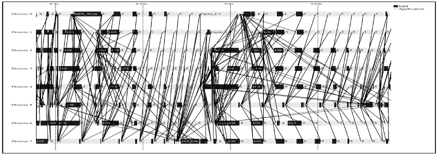


#1 Summit	#2 Taihu Light	#3 Sierra
2,282,544 cores	10,649,600 cores	1,572,480 cores
4608×(2×22-cores + 6GPU)	40 960× 260-cores RISCs	4300×(2×22-cores + 4GPU)
122 Tflops, 9MW	93 Tflops, 15MW	71 Tflops, 12MW

Complex Applications



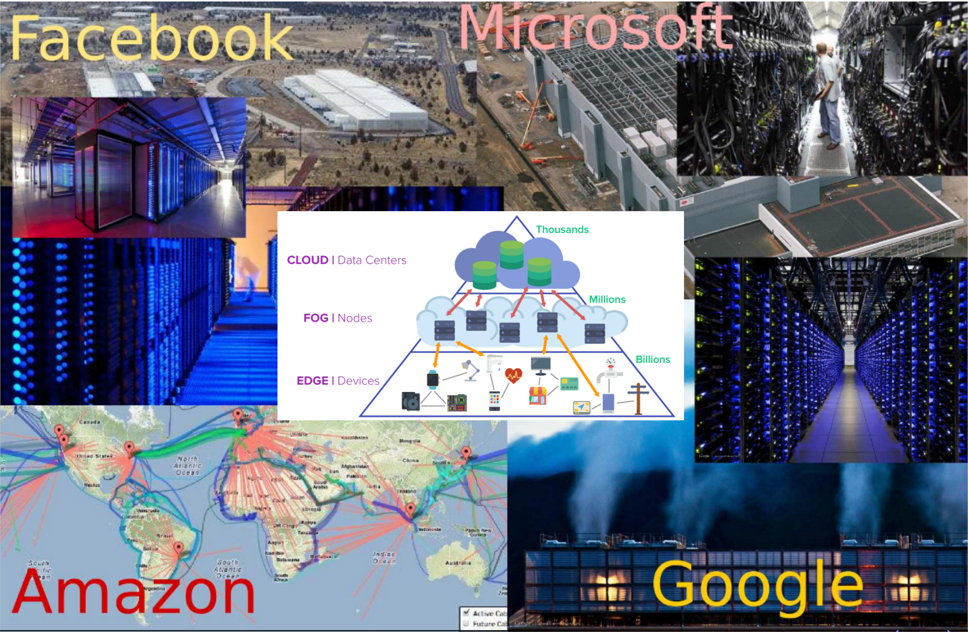
Rigid, Regular, Hand-tuned Comm Patterns



Dynamic, Irregular (task-based?)

How do we study these beasts?

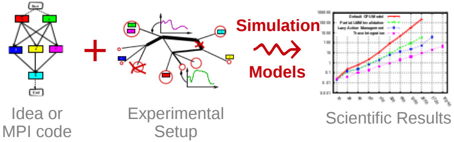
Cloud Computing, Internet of Things, Fog



Simulating Distributed Systems

Simulation: Fastest Path from Idea to Data

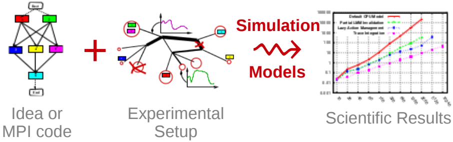
- Test your scientific idea with a fast and confortable scientific instrument



Simulating Distributed Systems

Simulation: Fastest Path from Idea to Data

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



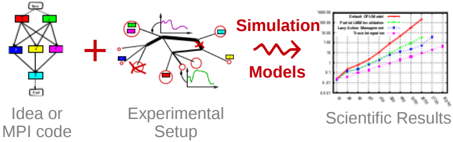
Simulation: Easiest Way to Study Real Distributed Systems

- 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

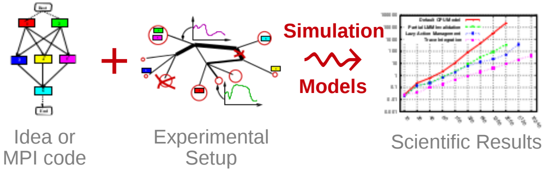
- Test your scientific idea with a fast and confortable scientific instrument

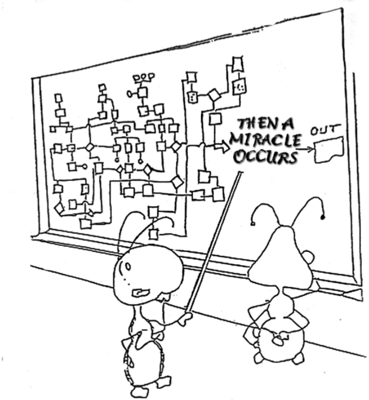


Simulation: Easiest Way to Study Real Distributed Systems

- 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

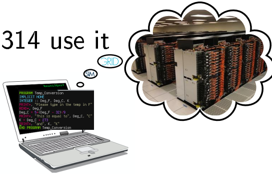
- Joint Project since 1998, mostly from French institutions
- Open Project, contributors in the USA (UHawaii, ISI, NEU), UK, Austria, Cern

Key Strengths

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

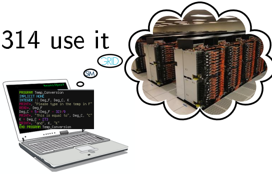
- Joint Project since 1998, mostly from French institutions
- Open Project, contributors in the USA (UHawaii, ISI, NEU), UK, Austria, Cern

Key Strengths

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

SMPI: Reimplementation of MPI on top of SimGrid

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

MSG: legacy interface for Concurent Sequential Processes

- Goal: ease the study of distributed algorithms (C or Java)
- Initially for distributed scheduling, used in many other contexts since 2005
- **Our main interface** is slowly getting crippled (backward compat when possible)

SimDag: legacy interface for DAG scheduling

- Goal: ease the study of centralized algorithms, since 1998

S4U: Future interface for algorithms

- Currently under development toward SimGrid 4, already usable (C++ or C)

BYOS: Build Your Own Simulator

- PSG Project: PeerSim interface implemented on top of SimGrid
- Wrench Project: Workflow Management System Simulation Workbench

SimGrid: Versatile Simulator of Distributed Apps

Install a Scientific Instrument on your Laptop

Computational Science of Computer Science

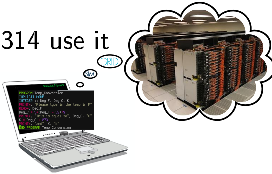
- Joint Project since 1998, mostly from French institutions
- Open Project, contributors in the USA (UHawaii, ISI, NEU), UK, Austria, Cern

Key Strengths

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

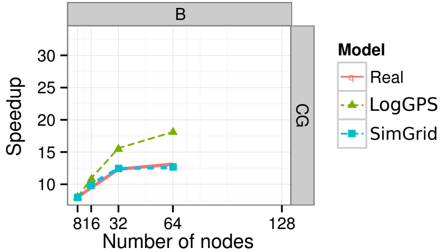
Community

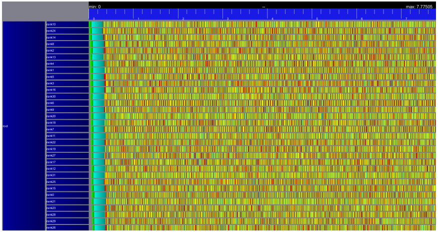
- Scientists: 500+ publications only cite it, 58 extend it, 314 use it
- Apps/Model co-dev : StarPU, BigDFT, TomP2P
- Some industrial users on internal projects (Intel, Bull)
- Open Source: external Power Users (fixes & models)



Validity Success Stories

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



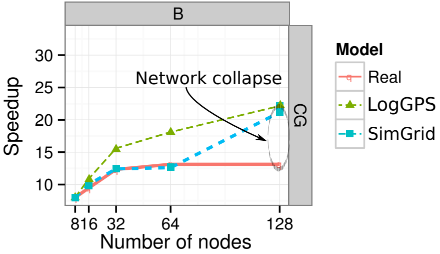


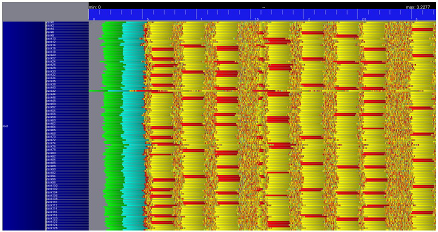
Key aspects to obtain this result

- 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

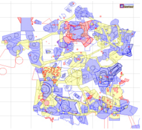
Perfect Model of France would be Perfect Map

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



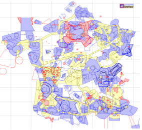



Maps (and models) are abstractions

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

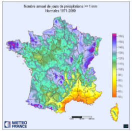
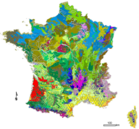
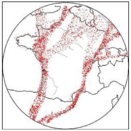
Perfect Model of France would be Perfect Map



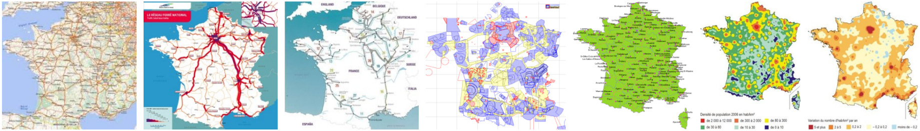



Maps (and models) are abstractions

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

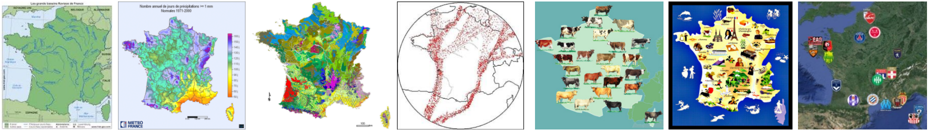





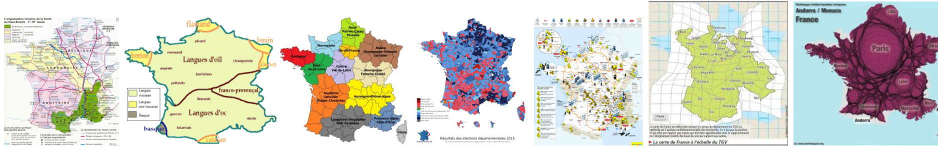
Perfect Model of France would be Perfect Map



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





Perfect Model of Distributed Systems?

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

In any case, assess the validity & soundness

SimGrid: Versatile Simulator of Distributed Apps

Install a Scientific Instrument on your Laptop

Computational Science of Computer Science

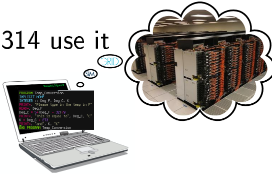
- Joint Project since 1998, mostly from French institutions
- Open Project, contributors in the USA (UHawaii, ISI, NEU), UK, Austria, Cern

Key Strengths

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



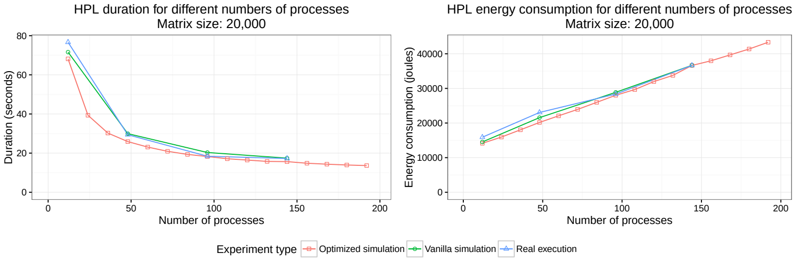
HPL and the Top500

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

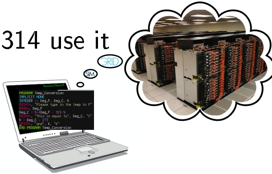
- Joint Project since 1998, mostly from French institutions
- Open Project, contributors in the USA (UHawaii, ISI, NEU), UK, Austria, Cern

Key Strengths

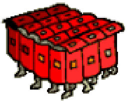
- Usability: Fast, Reliable, User-oriented APIs, Visualization
- Performance Models validated with Open Science $\rightsquigarrow$ Predictive Power
- Architectured as an OS $\rightsquigarrow$ Efficiency; Performance & Correction co-evaluation
- Versatility: Advances in Clouds modeling reused by DataGrid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

Community

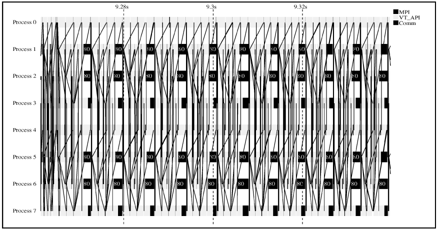
- Scientists: 500+ publications only cite it, 58 extend it, 314 use it
- Apps/Model co-dev : StarPU, BigDFT, TomP2P
- Some industrial users on internal projects (Intel, Bull)
- Open Source: external Power Users (fixes & models)



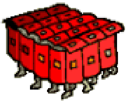
Writting Correct Distributed Applications



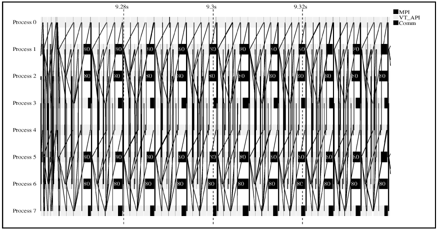
- Classical Solution: Proof of algorithms
- Pessimistic Solution: Lower performance expectations
- Optimistic Solution: Eventually Consistent
- HPC Solution: Rigid, Regular, Hand-tuned Communication Patterns

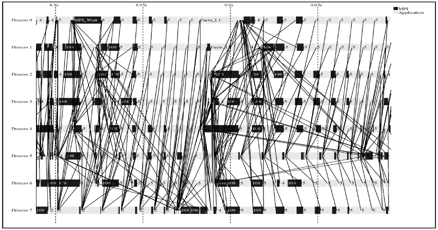


Writting Correct Distributed Applications



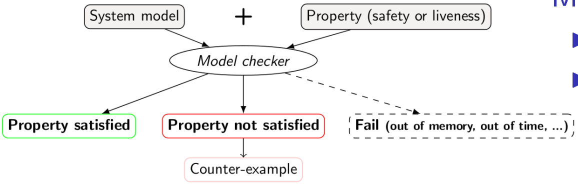
- Classical Solution: Proof of algorithms
- Pessimistic Solution: Lower performance expectations
- Optimistic Solution: Eventually Consistent
- HPC Solution: Rigid, Regular, Hand-tuned Communication Patterns
- Large-Scale Hybrid Machines: Dynamic, Irregular (task-based?)





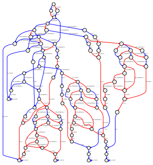
Verification: must explore **all possible execution paths**

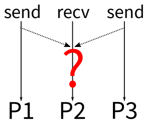
Formal Methods in Mc SimGrid



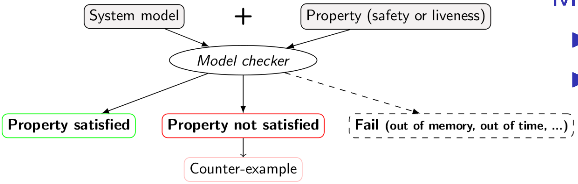
Model Checking

- Exhaustively search for faults
- Requires an accurate model





Formal Methods in Mc SimGrid

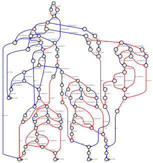


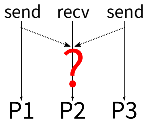
Model Checking

- Exhaustively search for faults
- Requires an accurate model

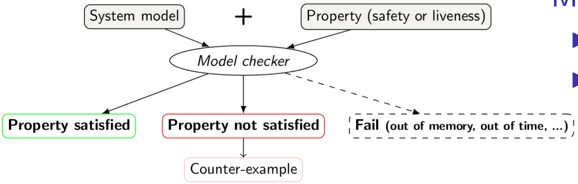
Dynamic Verification: similar idea, applied to source code

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





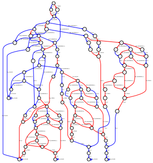
Formal Methods in Mc SimGrid



Model Checking

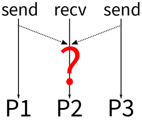
- Exhaustively search for faults
- Requires an accurate model

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

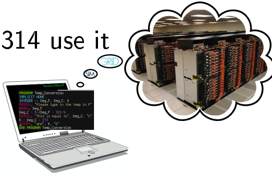
- Joint Project since 1998, mostly from French institutions
- Open Project, contributors in the USA (UHawaii, ISI, NEU), UK, Austria, Cern

Key Strengths

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

Community

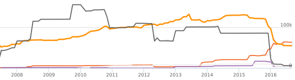
- Scientists: 500+ publications only cite it, 58 extend it, 314 use it
- Apps/Model co-dev : StarPU, BigDFT, TomP2P
- Some industrial users on internal projects (Intel, Bull)
- Open Source: external Power Users (fixes & models)

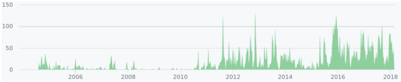


Technical Considerations

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 $\approx$ 6 commiters, 4 releases a year
- Ongoing full rewrite in C++ along with *Release soon, Release often*

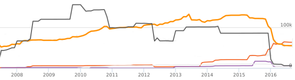


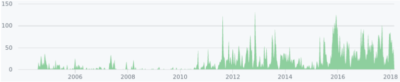


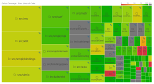
Technical Considerations

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 :)

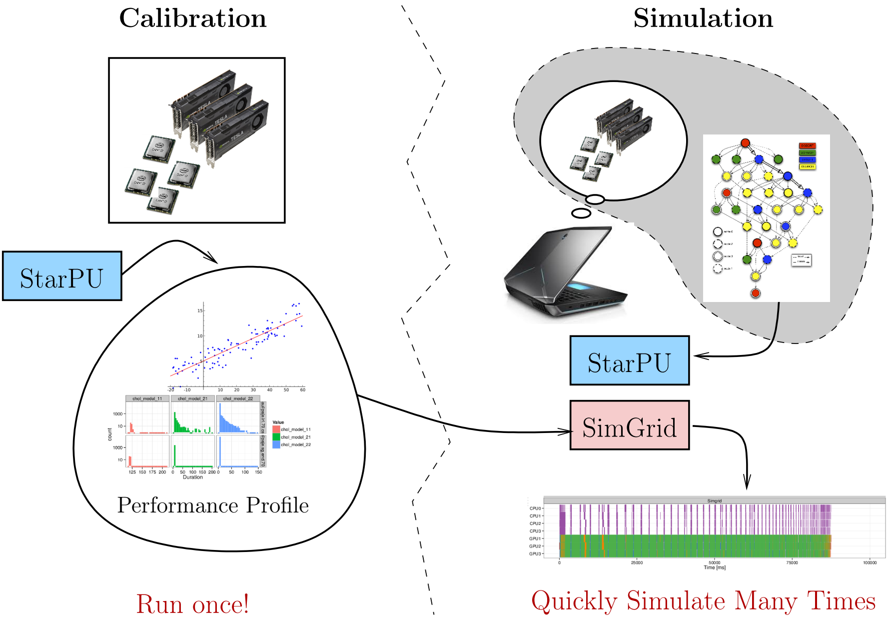
Training and User Support in Computing Centers

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

Toward Education

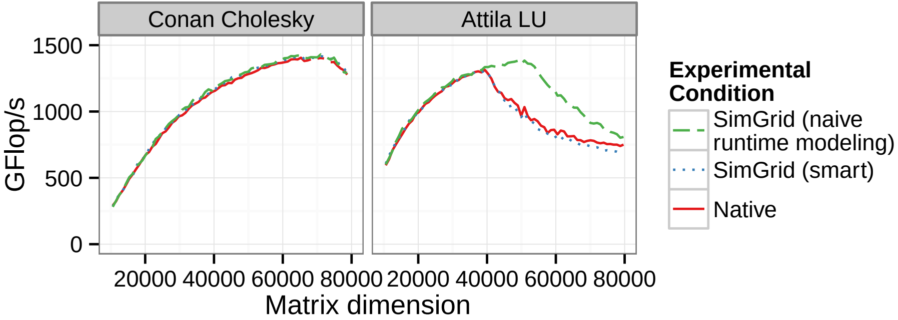
- Teach now the researchers and engineers of tomorrow to SimGrid
- Done: SMPI CourseWare, PeerSimGrid; Ongoing: Cloud, Wrench and more?

StarPU-Simgrid Overview



StarPU-Simgrid on dense linear algebra

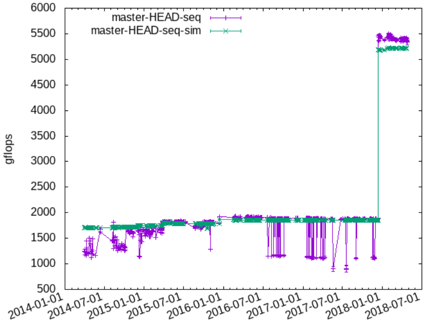
- Accurate simulated time results
- Already required a lot of care
- Extensively used for scheduling research



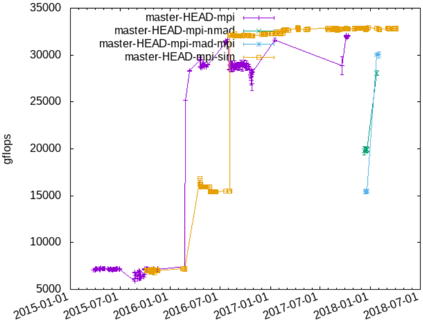
Continuous Integration of StarPU using SimGrid

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)

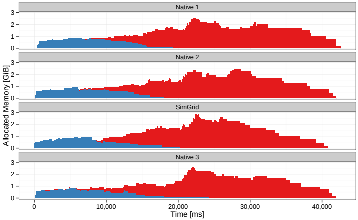
StarPU Visualization

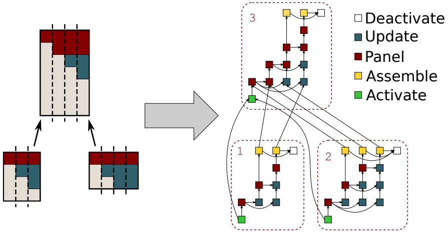
Get data without Heisenbug, analyze it with R

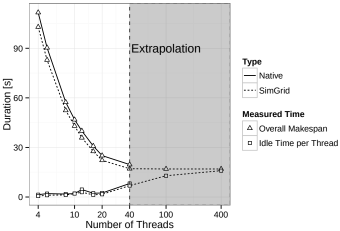
StarPU QR-Mumps

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

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





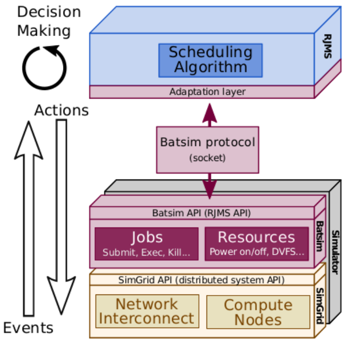


Perspective Tune app. and scheduler, capacity (memory) planning

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`

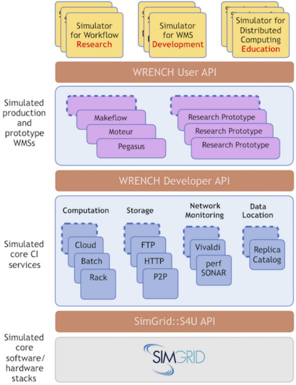
Wrench

A Workflow Management System Simulation Workbench

- Objective
  - Provide high-level building blocks for developing custom simulators

WRENCH

- 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)

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

(Courtesy of Thomas Bocek)

SimGrid: Versatile Simulator of Distributed Apps

Install a Scientific Instrument on your Laptop

Computational Science of Computer Science

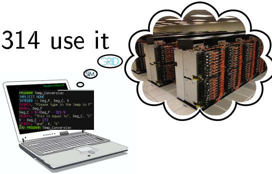
- Joint Project since 1998, mostly from French institutions
- Open Project, contributors in the USA (UHawaii, ISI, NEU), UK, Austria, Cern

Key Strengths

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



Question slides

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)

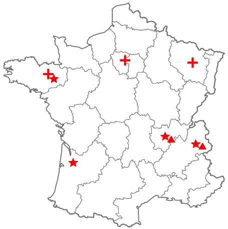
IPL HAC-SPECIS (2016-2020)

Inria Project Lab ≈ 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

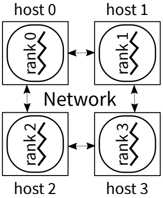
- **Research Question:** Joint Study of Performance **AND** Correctness
- Goal: *bridge the gap between communities*

Virtualizing MPI Applications with SimGrid

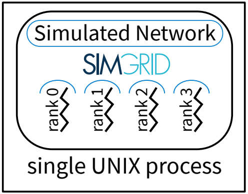
SMPI: Reimplementation of MPI on top of SimGrid

- 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

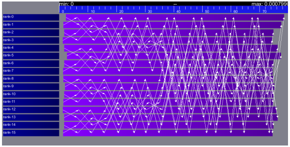


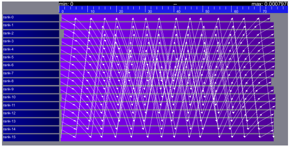
SimGrid Simulation

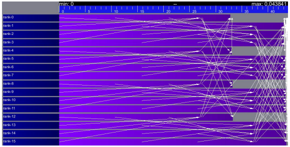
SimGrid Modeling of MPI

MPI Collectives

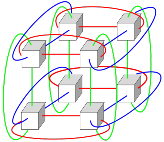
- SimGrid implements more than 120 algorithms for the 10 main MPI collectives
- Selection logic from OpenMPI, MPICH can be reproduced



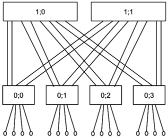




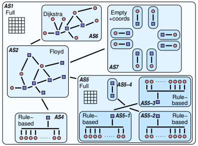
HPC Topologies



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

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

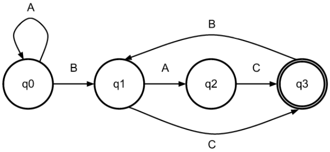
Comm Patterns: "It exists a pattern that is the same for all exec paths"

- 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: $\forall$ 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)

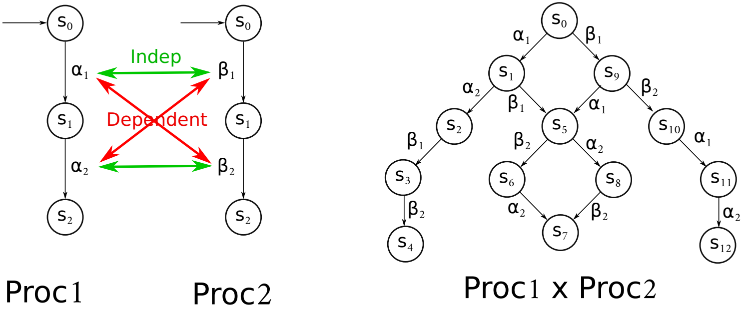
- Avoid re-exploring equivalent interleavings
- Don't explore all interleavings of local executions: they are equivalent

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?

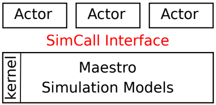
Transition = atomic block of code between Indecision Points

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

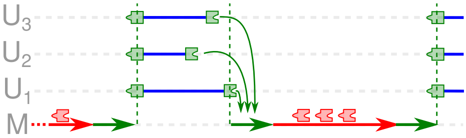
Implementation: SimGrid is an Operating System

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

Functional View

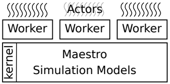


Temporal View

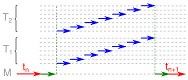


Going parallel

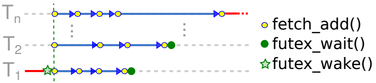
- More actors than cores ⇝ Worker Threads that execute co-routines



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)

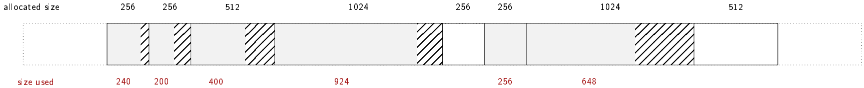
- Avoid re-exploring equivalent interleavings
- Don't explore all interleavings of local executions: they are equivalent

System-Level State Equality

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

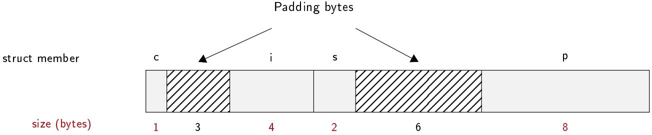
OS-level State Equality Detection

- Memory over-provisioning

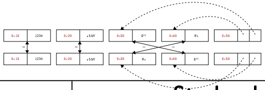


- Padding bytes: Data structure alignment

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



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



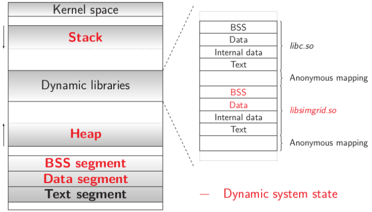
Solutions

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

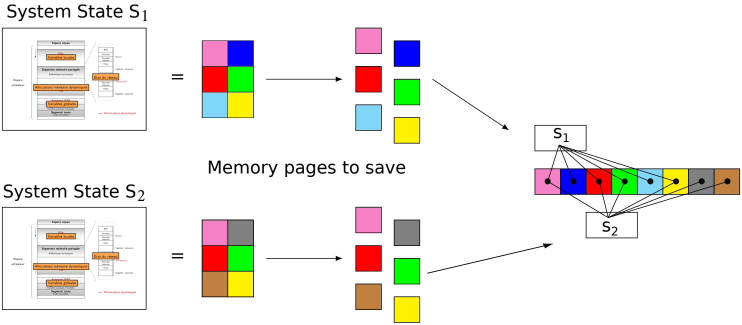


How to privatize the BSS+data

- (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 ☹