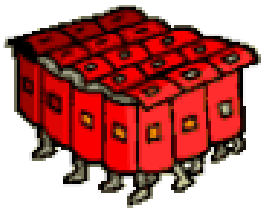
Verifying MPI Applications with Mc SimGrid

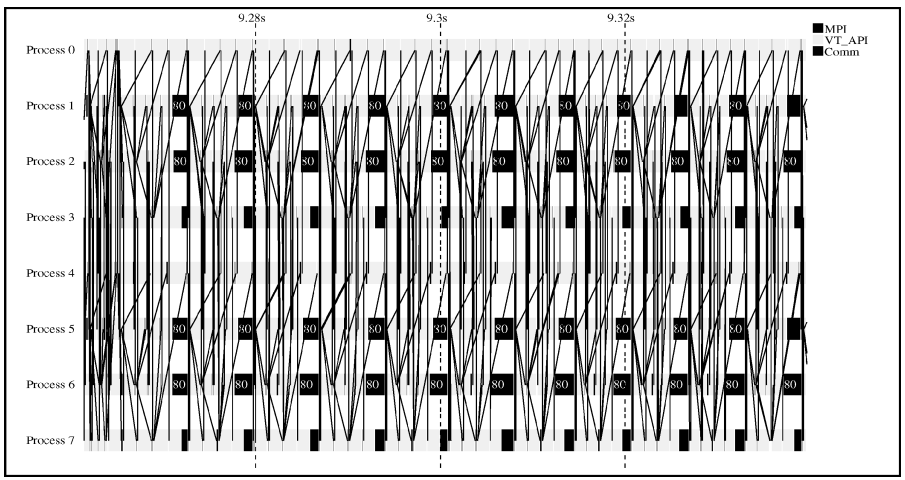
The Anh Pham, Thierry Jéron, Martin Quinson
Univ. Rennes – Inria – CNRS – IRISA (France)

Correctness 2017
Denver, Colorado

Writting Correct Distributed Applications

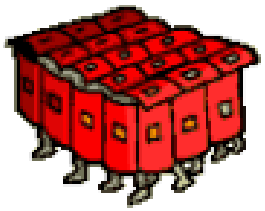
- Classical Solution: Proof of algorithms
- Pessimistic Solution: Lower performance expectations
- Optimistic Solution: Eventually Consistent
- HPC Solution: Rigid, Regular, Hand-tuned Communication Patterns

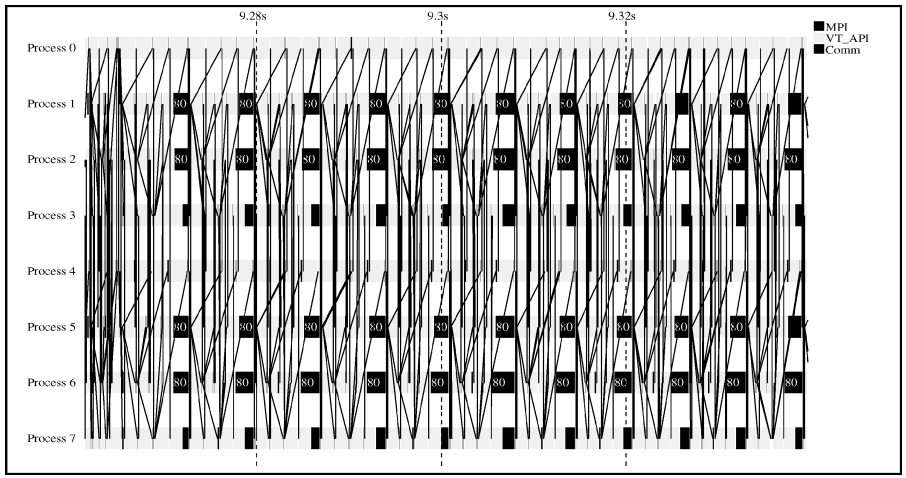


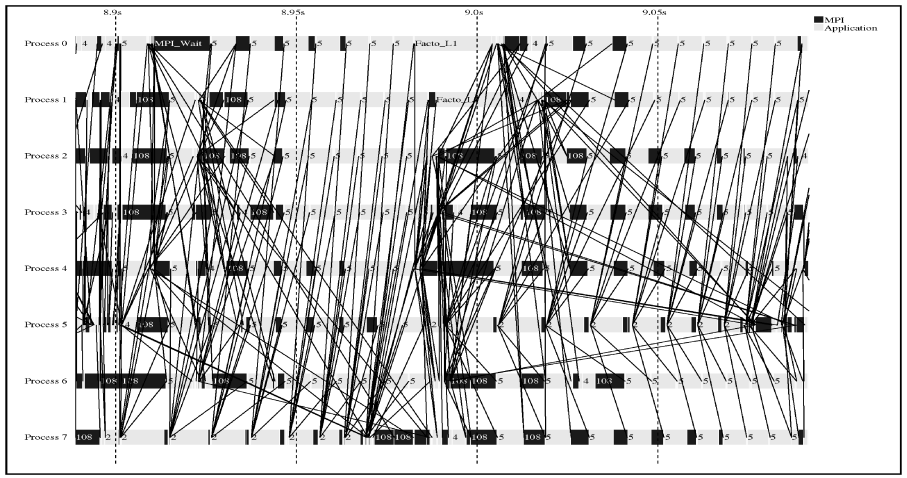


Writting Correct Distributed Applications

- 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**

Virtualizing MPI Applications with SimGrid

SimGrid: Simulate Performance of Distrib. Apps

- Versatile: HPC, Clouds, and others
- Validated Predictive Power, Highly Scalable
- Grounded 150 publications by 120 ppl, 30 contributors

`http://simgrid.org`

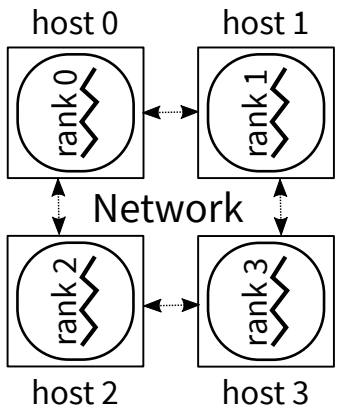
Virtualizing MPI Applications with SimGrid

SimGrid: Simulate Performance of Distrib. Apps

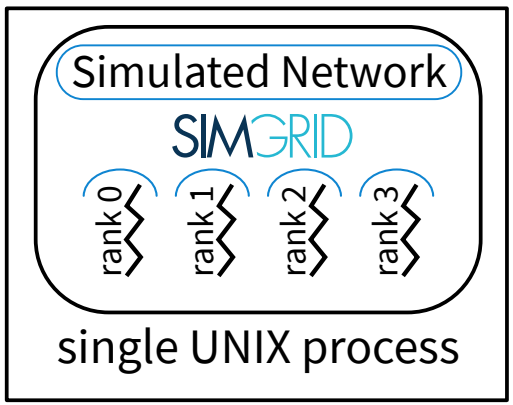
- Versatile: HPC, Clouds, and others
- Validated Predictive Power, Highly Scalable
- Grounded 150 publications by 120 ppl, 30 contributors

`http://simgrid.org`

MPI Applications are *folded into* a single process



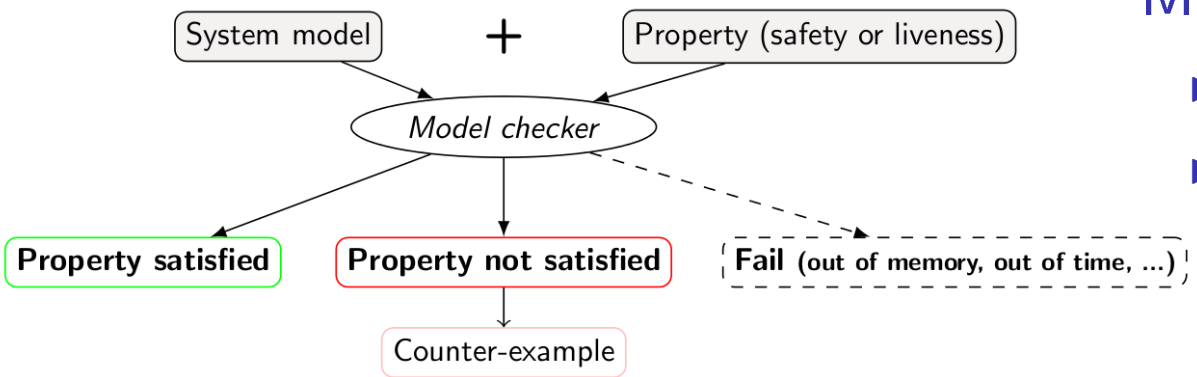
Real Settings



SimGrid Simulation

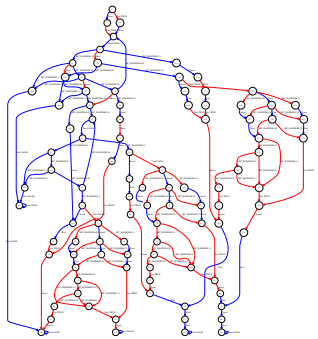
McSimGrid builds upon SimGrid to verify MPI applications

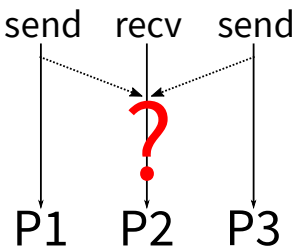
Formal Methods in Mc SimGrid



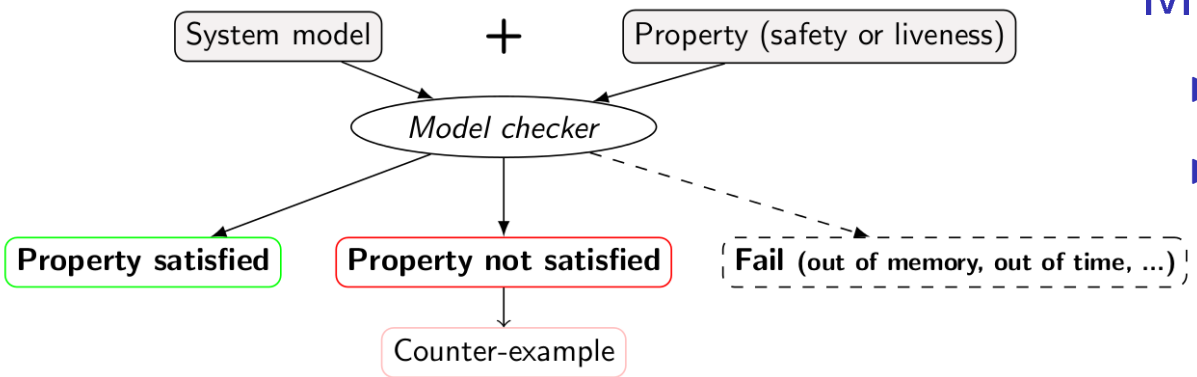
Model Checking

- Exhaustively search for faults
- Requires an accurate model





Formal Methods in Mc SimGrid

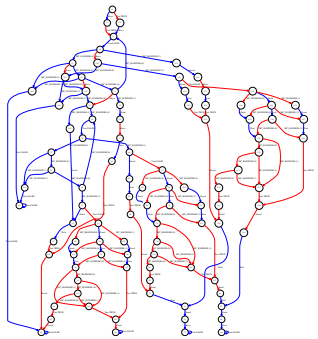


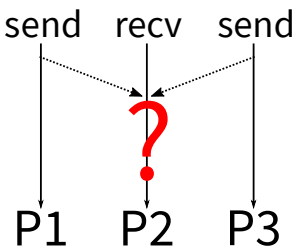
Model Checking

- Exhaustively search for faults
- Requires an accurate model

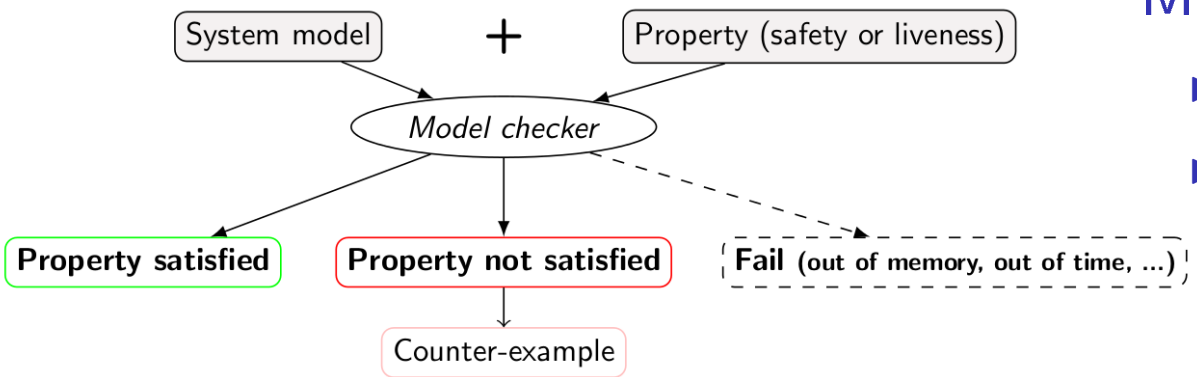
Dynamic Verification: similar idea, applied to source code

- Mc SimGrid: no static analysis, virtualized execution
- On Indecision Points: checkpoint, explore, rollback



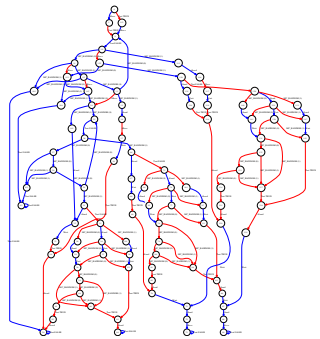


Formal Methods in Mc SimGrid



Model Checking

- Exhaustively search for faults
- Requires an accurate model

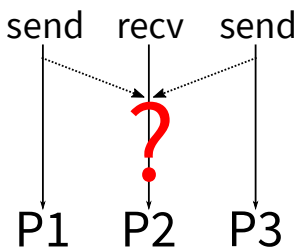


Dynamic Verification: similar idea, applied to source code

- Mc SimGrid: no static analysis, virtualized 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



Mitigating the State Space Explosion

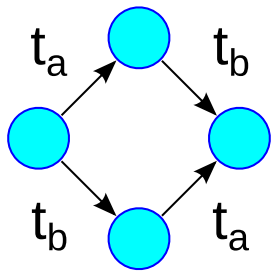
Dynamic Partial Ordering Reduction (DPOR)

System-Level State Equality

Mitigating the State Space Explosion

Dynamic Partial Ordering Reduction (DPOR)

- Avoid re-exploring Mazurkiewicz traces (don't commute independent events)
- iSend+iSend are dependent, ...
- Adapted to safety, not to liveness (cycles)

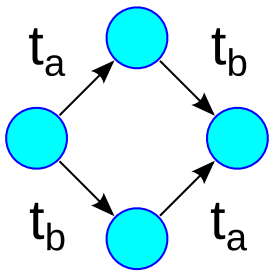


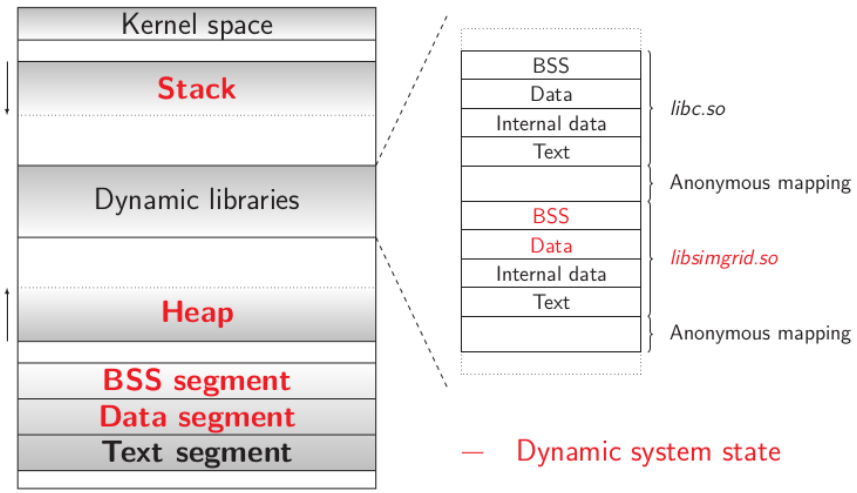
System-Level State Equality

Mitigating the State Space Explosion

Dynamic Partial Ordering Reduction (DPOR)

- Avoid re-exploring Mazurkiewicz traces (don't commute independent events)
- iSend+iSend are dependent, ...
- Adapted to safety, not to liveness (cycles)





System-Level State Equality

- Detect when a given state was previously explored
- Introspect the applications similarly to gdb
- Heuristic for both safety and liveness
- Also with Memory Compaction

Use Cases: Kind of Properties

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

Conclusion

Mc SimGrid: Dynamic Verification of MPI applications

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

Future Works

- Improve DPOR by using Event Unfolding structures
- State equality detection more configurable
- More semantics subtleties for Point-to-Point communications
- Multi-threaded MPI: by instrumenting LLVM bytecode?
- Apply to more code (starting with student projects) Gather user's feedback

What [liveness] property would be interesting on your code?

More on 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 ☹

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

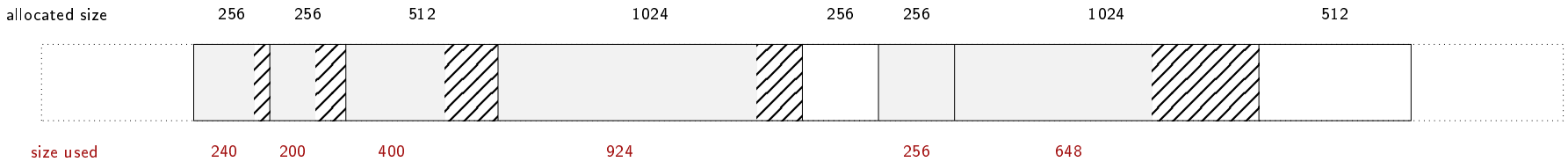
- deterministic: On each node, send and receive events are always in same order
- send deterministic: $\forall$ node, send are always the same, no matter the recv order
- Not liveness, not even LTL: quantifies for all execution paths within property

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

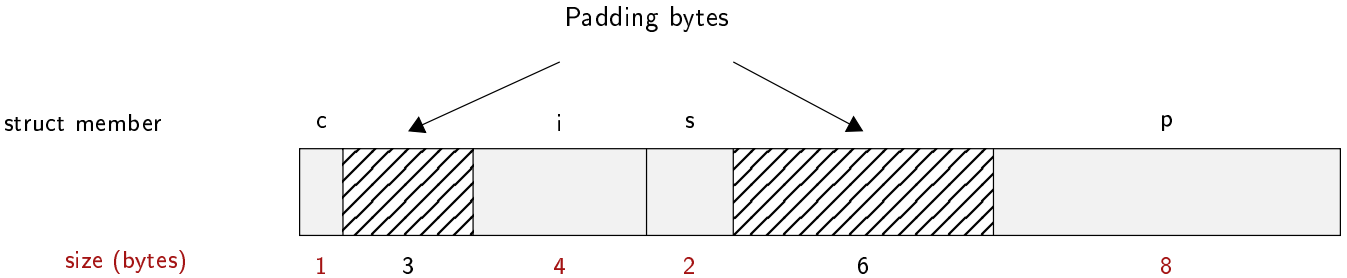
OS-level Challenges of State Equality Detection

- Memory over-provisioning

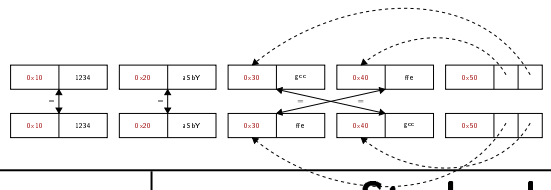


- Padding bytes: Data structure alignment

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



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

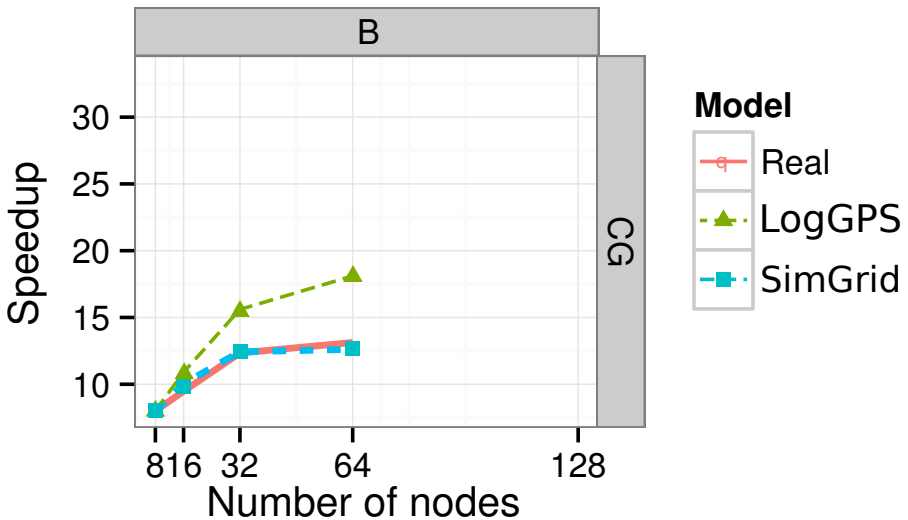


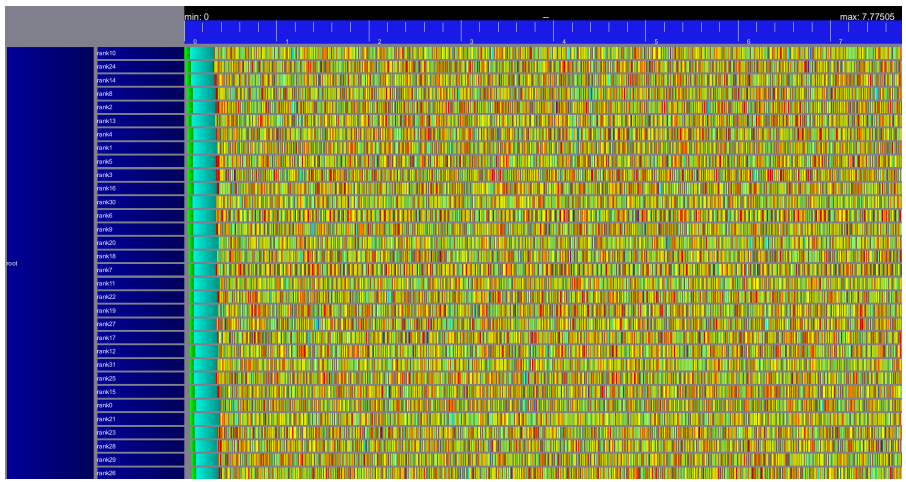
Solutions

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)

Validity Success Stories

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



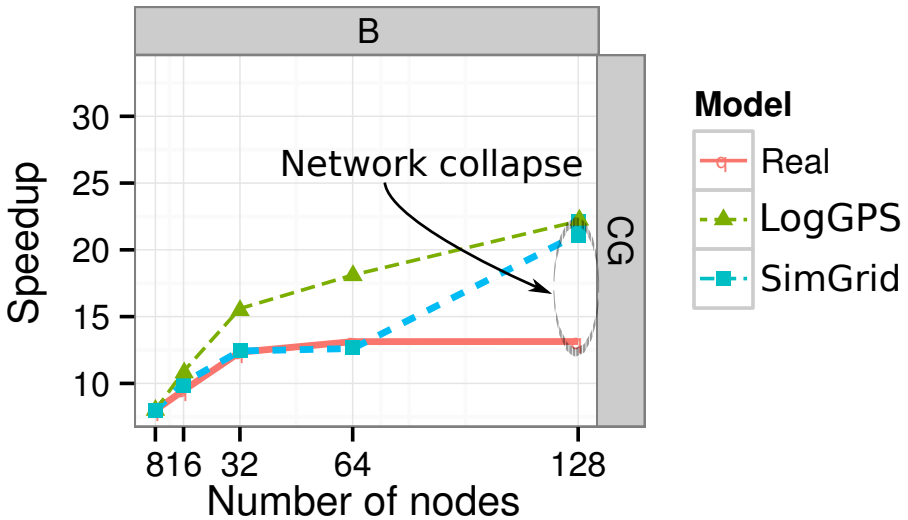


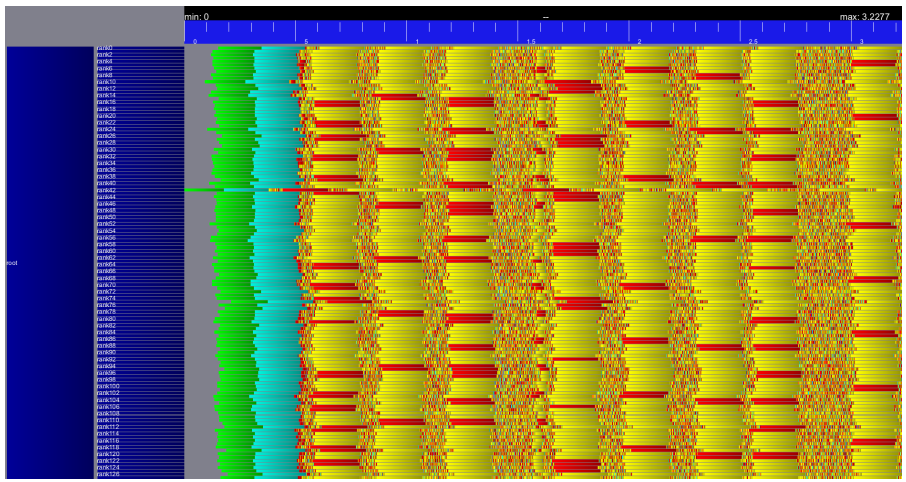
Key aspects to obtain this result

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

Validity Success Stories

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





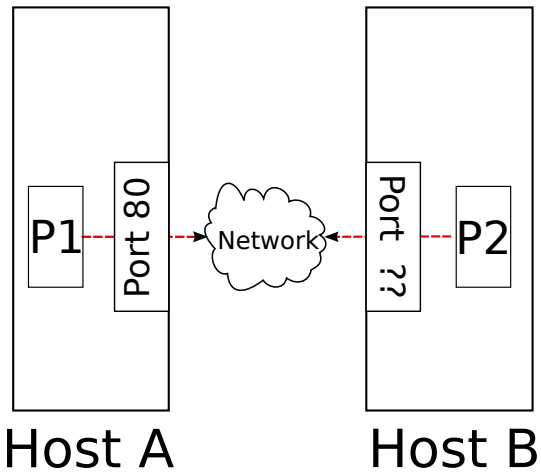
Discrepency between Simulation and Real Experiment. Why?

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

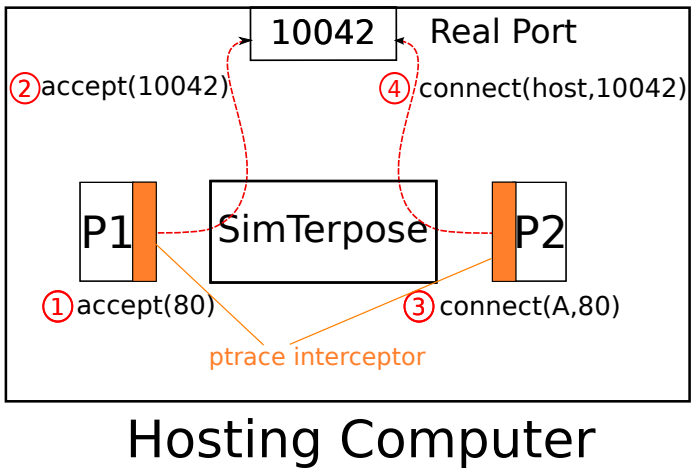
SimTerpose Project

Dream: Simulate any applications on top of SimGrid

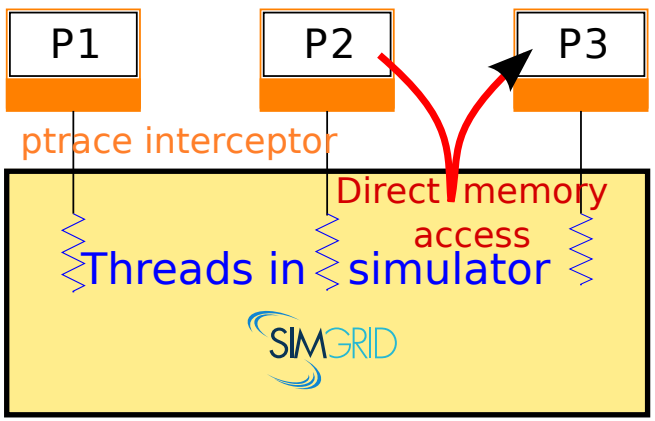
Simulated Setup



Take 1: ptrace plumbering



Take 2: Full Emulation



Current State

- Functional POC: send/recv exchange
- Need to handle the other 200 syscalls
  - Intercept, store metadata
  - Inform simulator, report effect on procs
- Time and DNS need love at link time
- We are redeveloping a libC! (in strange way ;)