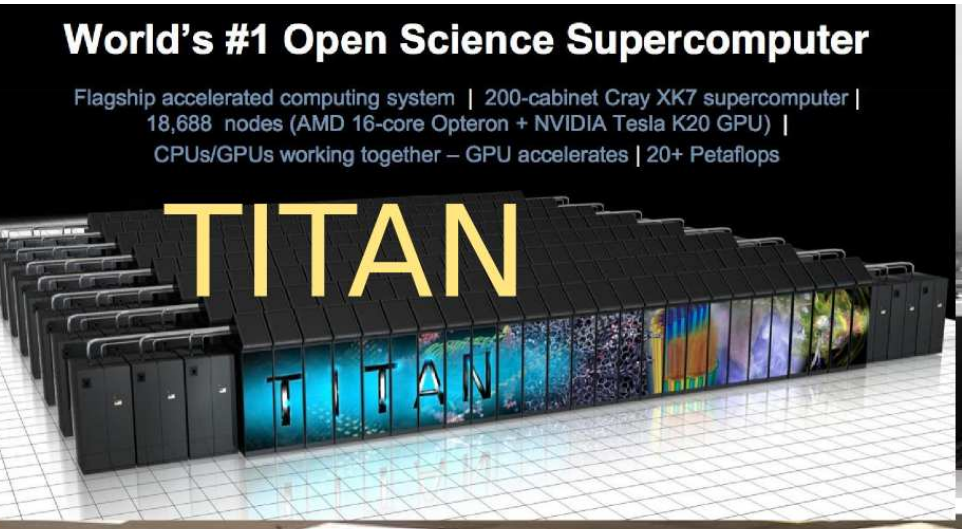
Performance Prediction of HPC Applications
The SimGrid Project

A. Legrand and the SimGrid team
(M. Quinson, F. Suter, A. Degomme, L. Stanisic, B. Videau,
L. Genovese, S. Thibault, E. Aggulo, . . .)

CNRS/Inria/University of Grenoble

Maison de la simulation
May 10, 2016

Supercomputers

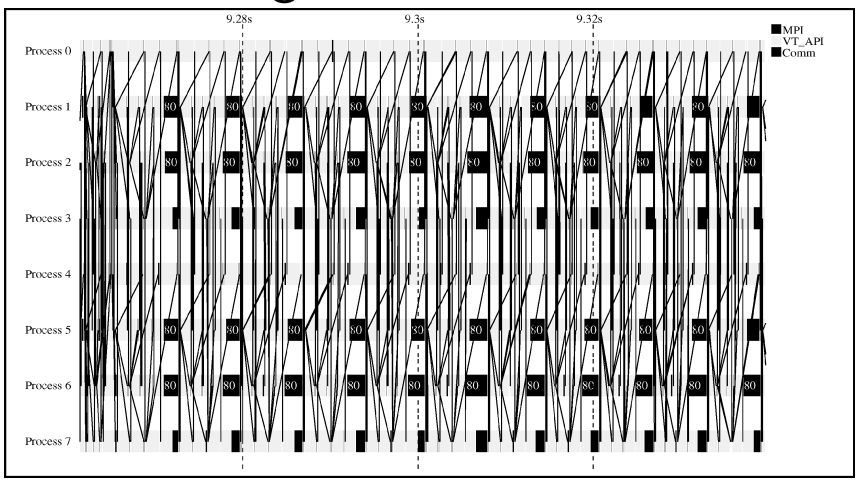


- 100,000 to 1,000,000 cores, accelerators (GPU, Xeon Phi) and a complex high speed interconnect
- A worldwide competition (Top500)

Complex Applications

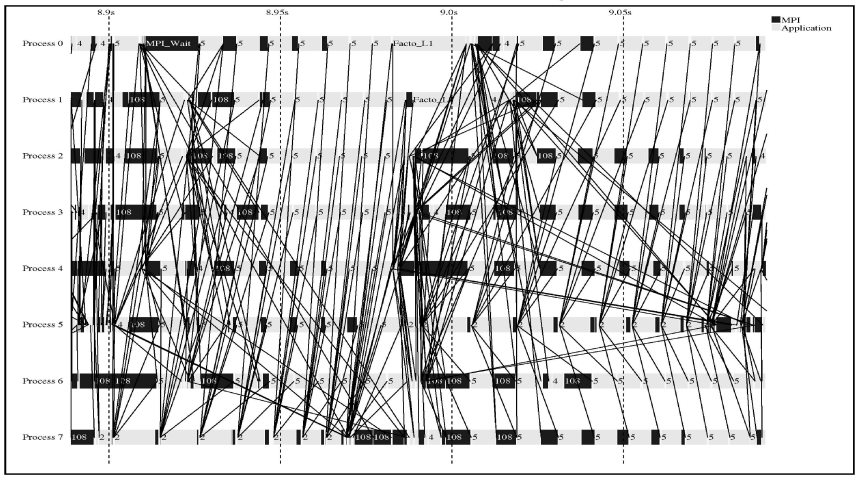
Larger and larger scale hybrid machines $\rightsquigarrow$ Different programming approaches
(e.g., in linear algebra applications)

"Rigid, hand tuned"



SuperLU

Task-based and Dynamic



MUMPS

Analysis and Comparison of Two Distributed Memory Sparse Solvers
Amestoy, Duff, L'excellent, Li. ACM Trans. on Math. Software, Vol. 27, No. 4, 2001.

Multitude of technologies

MPI | OpenMP | Cilk | CUDA | OpenCL | Charm++
KAAPI | StarPU | QUARK | ParSEC | OmpSs | ...

$\rightsquigarrow$ need for performance evaluation on a regular basis

Toward Exascale ?

Already insanely complex platforms and applications with Peta-scale systems. Do we have a chance to understand exascale systems ?

- European approach to Exascale: Mont-Blanc; low-power commodity hardware s.a. ARM+GPUs+Ethernet
- Need for application performance prediction and capacity planning

MPI simulation: what for ?

1. Helping application developers
   - Non-intrusive tracing and repeatable execution
   - Classical debugging tools (gdb, valgrind) can be used
   - Save computing resources (runs on your laptop if possible)
2. Helping application users
   - How much resources should I ask for? (scaling)
   - Configure MPI collective operations
   - Provide baseline
3. Capacity planning (can we save on components? what-if analysis)

Flourishing state of the Art

There are many different projects:

- Dimemas (BSC, probably one of the earliest)
- PSINS (SDSC, used to rely on Dimemas)
- BigSim (UIUC): BigNetSim or BigFastSim
- LogGopSim (UIUC/ETHZ)
- SST (Sandia Nat. Lab.): Micro or Macro
- ...

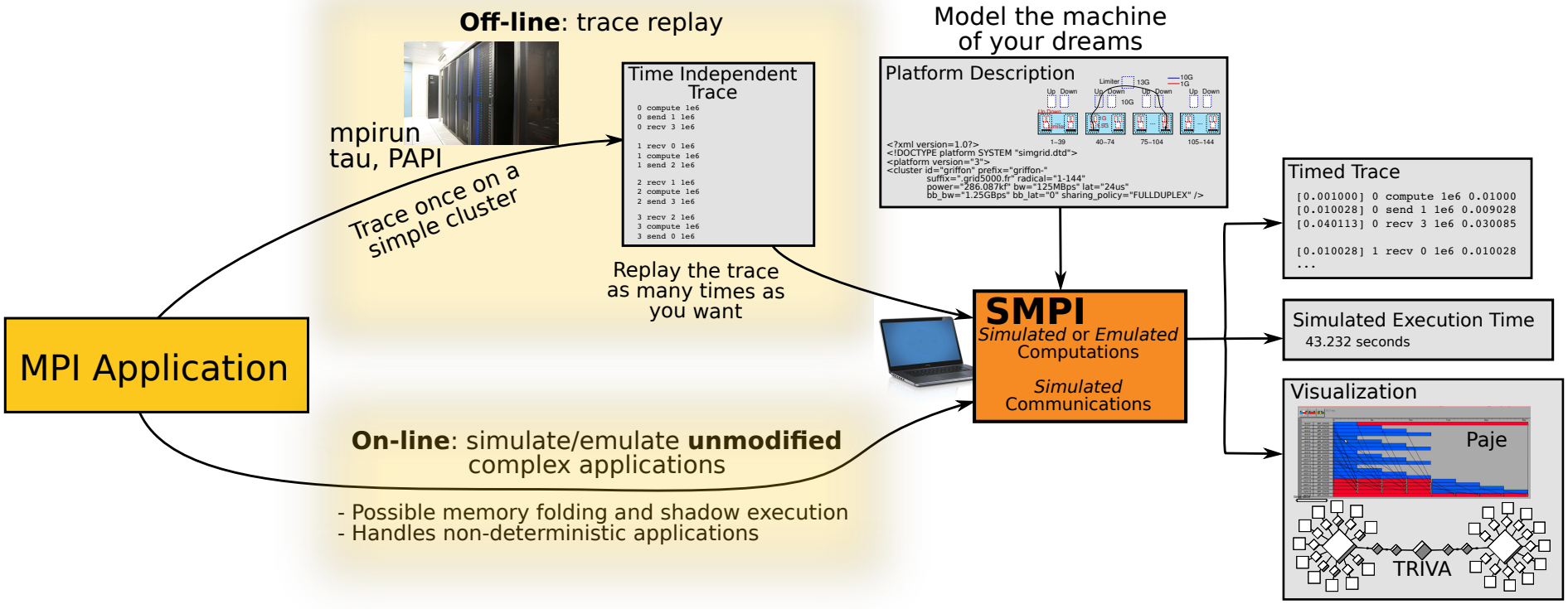
SimGrid: A 16 years old open-source project. Collaboration between France (INRIA, CNRS, Univ. Lyon, Nancy, Grenoble, ...), USA (UCSD, U. Hawaii), UK, Austria (Vienna)...

http://simgrid.gforge.inria.fr

- Initially focused on Grid settings, we argue that the same tool/techniques can be used for P2P, HPC and cloud

Outline

SMPI – Offline vs. Online Simulation

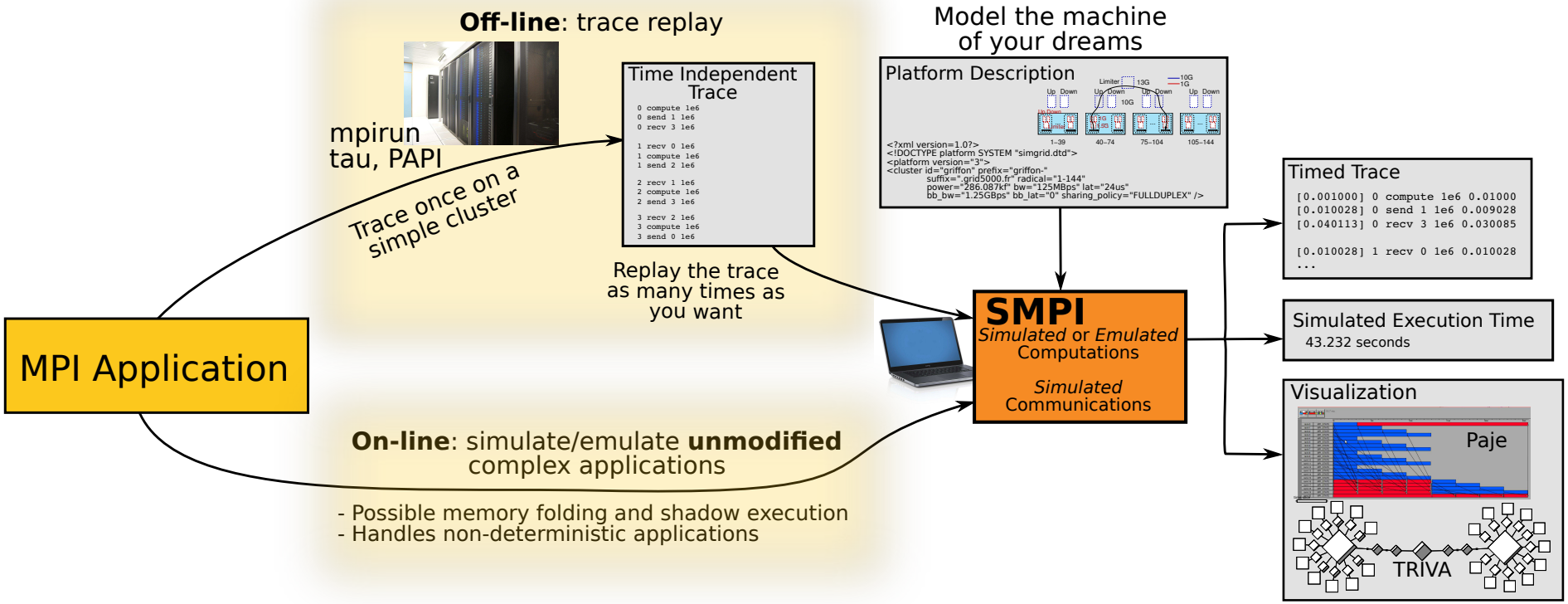


Offline simulation

1. Obtain a time independent trace
2. Replay it on top of SimGrid as often as desired
3. Analyze with the comfort of a simulator

Fast, but requires extrapolation and limited to non-adaptive codes

SMPI – Offline vs. Online Simulation



Online simulation

- Directly run the code on top of SimGrid: ~~source-to-source (Photran, coccinelle, f2c)~~, ~~GOT injection~~, ~~compiler pass for TLS~~, mmap the data segment
- Possible memory sharing between simulated processes (reduces memory footprint) and kernel sampling (reduces simulation time)
- Complies with most of the MPICH3 testsuite, compatible with many C F77 and F90 codes (NAS, LinPACK, Sweep3D, BigDFT, SpecFEM3D)

Performance Evaluation Through Fine Grain Simulation

Packet-level models full simulation of the whole protocol stack so hopefully perfect, but

Performance Evaluation Through Fine Grain Simulation

Packet-level models full simulation of the whole protocol stack so hopefully perfect, but

- complex models $\rightsquigarrow$ hard to instantiate and unstable

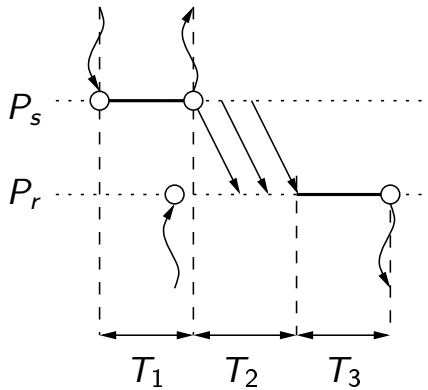
Flores Lucio, Paredes-Farrera, Jammeh, Fleury, Reed. *Opnet modeler and ns-2: Comparing the accuracy of network simulators for packet-level analysis using a network testbed.* WSEAS Transactions on Computers 2, no. 3 (2003)

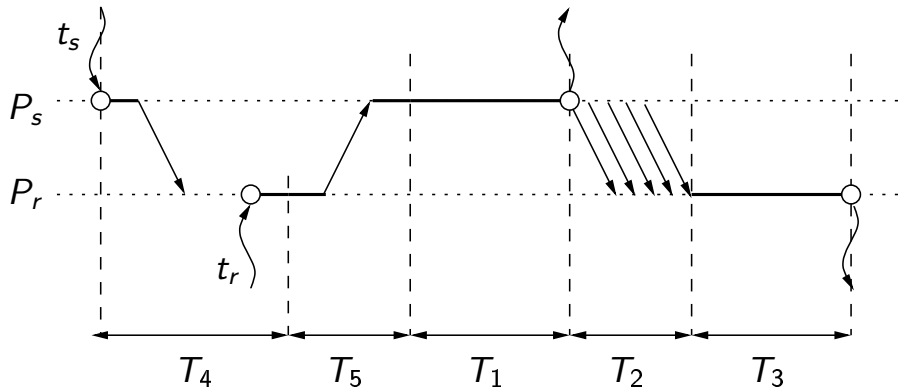
- inherently slow (parallelism won't save you here!)
- sometimes wrongly implemented
- who can understand the macroscopic behavior of the application ?

When working at the application level, there is a need for something more high level that reflects the macroscopic characteristics of the machines

LogGPS in a Nutshell

The LogP model was initially designed for complexity analysis and algorithm design. Many variations available to account for protocol switch





Asynchronous mode ($k \leqslant \boxed{S}$) Rendez-vous mode ($k > S$)

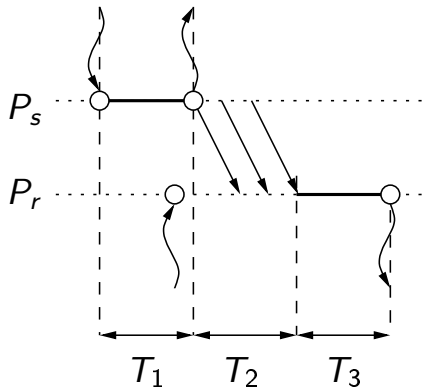
The T_i's are basically continuous linear functions.

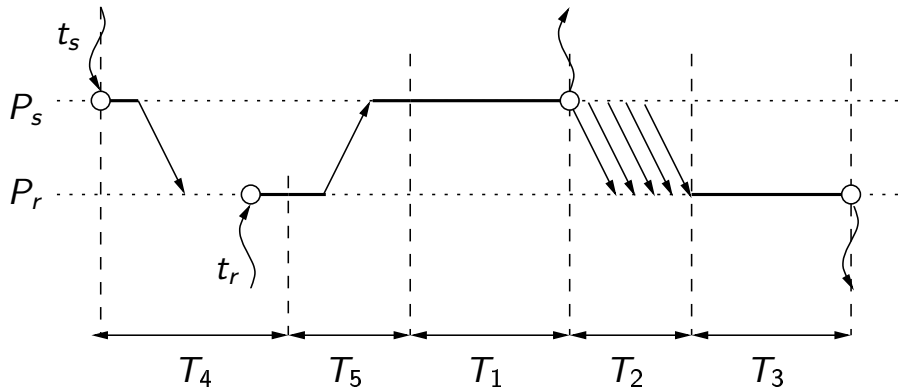
$$T_1 = o + kO_s \qquad T_2 = \begin{cases} L + kg & \text{if } k < \boxed{s} \\ L + sg + (k-s)G & \text{otherwise} \end{cases}$$

$$T_3 = o + kO_r \qquad T_4 = \max(L+o, t_r - t_s) + o \qquad T_5 = 2o + L$$

LogGPS in a Nutshell

The LogP model was initially designed for complexity analysis and algorithm design. Many variations available to account for protocol switch





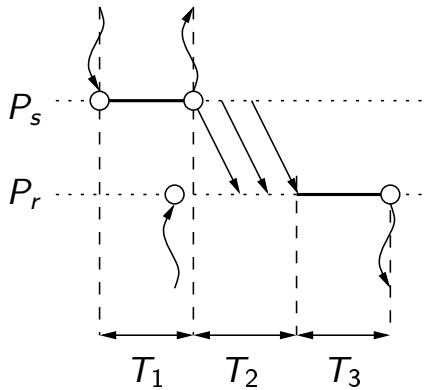
Asynchronous mode ($k \leqslant \boxed{S}$) Rendez-vous mode ($k > S$)

The T_i's are basically continuous linear functions.

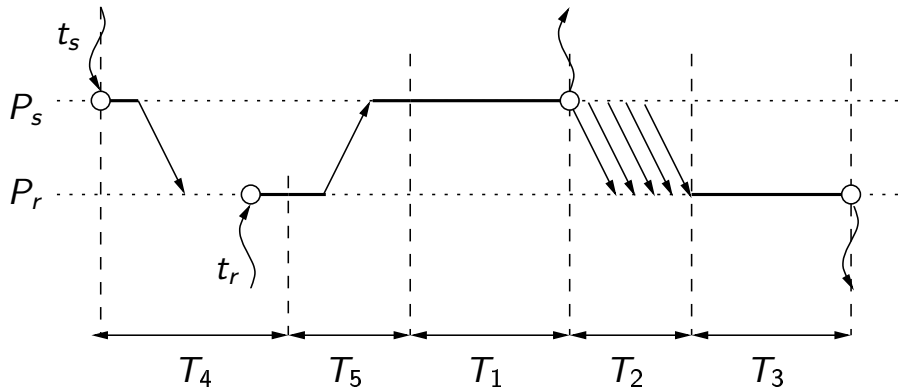
Routine	Condition	Cost
`MPI_Send`	$k \leqslant S$	T_1
	$k > S$	$T_4 + T_5 + T_1$
`MPI_Recv`	$k \leqslant S$	$\max(T_1 + T_2 - (t_r - t_s), 0) + T_3$
	$k > S$	$\max(o + L - (t_r - t_s), 0) + o +$ $T_5 + T_1 + T_2 + T_3$
`MPI_Isend`		o
`MPI_Irecv`		o

LogGPS in a Nutshell

The LogP model was initially designed for complexity analysis and algorithm design. Many variations available to account for protocol switch



Asynchronous mode ($k \leqslant \boxed{S}$)



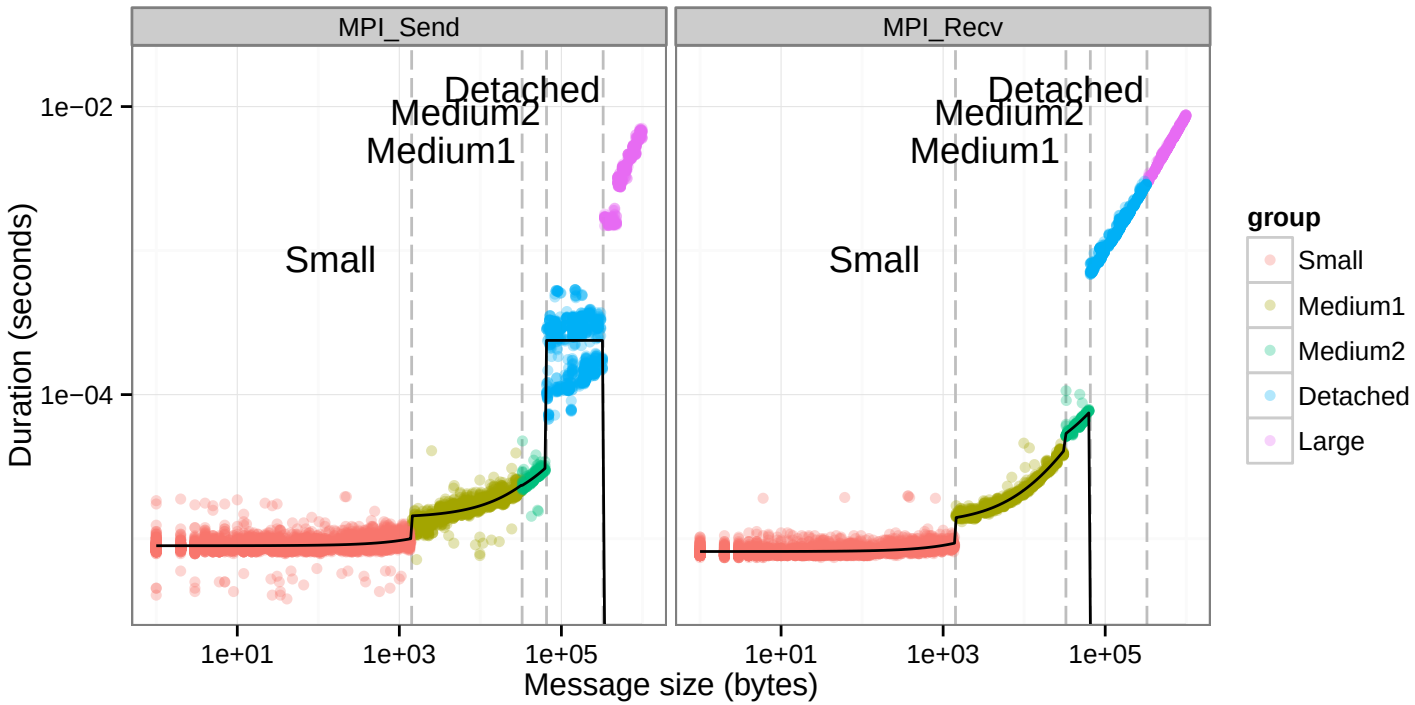
Rendez-vous mode ($k > S$)

The T_i's are basically continuous linear functions.

- May reflect the operation of specialized HPC networks from the early 1990s...
- Ignores potentially confounding factors present in modern-day systems (e.g., contention, topology, complex protocol stack, ...)
- Unless you have a well-tuned high-end machine, such model is unlikely to provide accurate estimations or useful baseline comparisons

MPI Point-to-Point Communication

Randomized measurements (OpenMPI/TCP/Eth1GB) since we are not interested in peak performances but in performance characterization

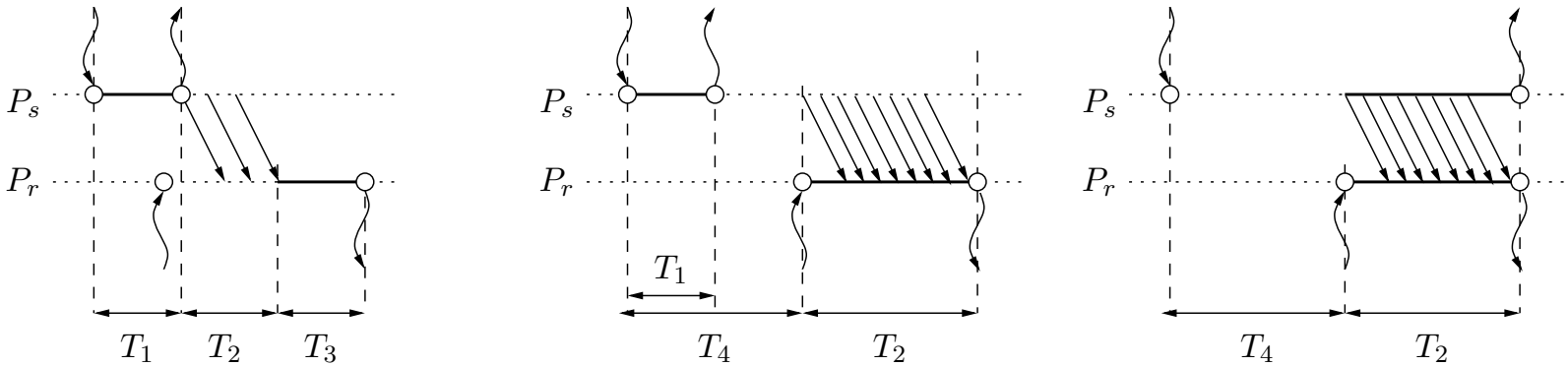


- There is a quite important variability
- There are at least 4 different modes
- It is piece-wise linear and discontinuous

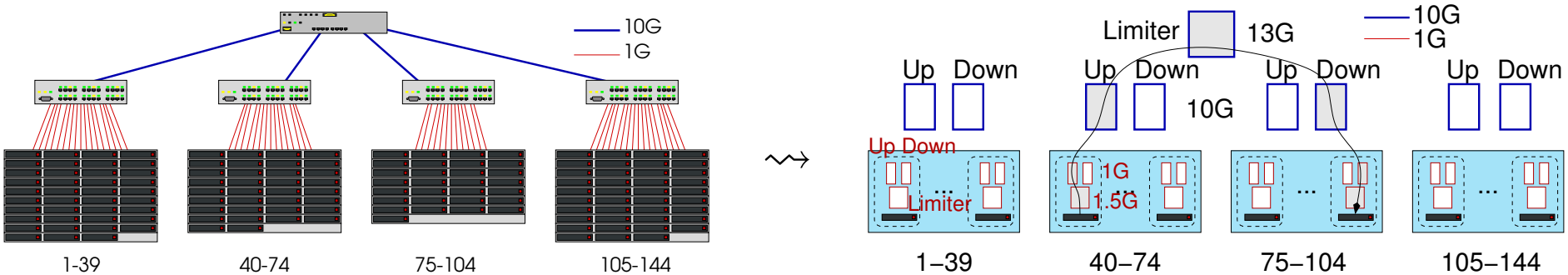
SMPI – Hybrid Model

SMPI combines accurate description of the platform, with both fluid and LogP family models:

- LogP: measure on real nodes to accurately model pt2pt performance (discontinuities) and communication modes (asynchronous, detached, synchronous)



- Fluid model (bandwidth sharing): account for contention and network topology



Collective Communications

Classical approaches:

- use simple analytical formulas
- benchmark everything and inject corresponding timing
- trace communication pattern and replay

Real MPI implementations have several implementations for each collective and select the right one at runtime

- 2300 lines of code for the AllReduce in OpenMPI!!!

Collective Communications

Classical approaches:

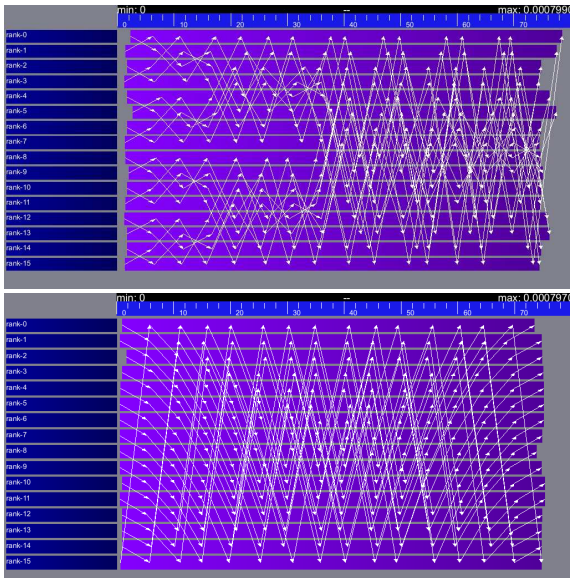
- use simple analytical formulas
- benchmark everything and inject corresponding timing
- trace communication pattern and replay

Real MPI implementations have several implementations for each collective and select the right one at runtime

- 2300 lines of code for the AllReduce in OpenMPI!!!

SMPI now uses

- more than 100 collective algorithms from three existing implementations (MPICH, OpenMPI, STAR-MPI) can be selected
- the same selection logic as MPICH or OpenMPI to accurately simulate their behavior

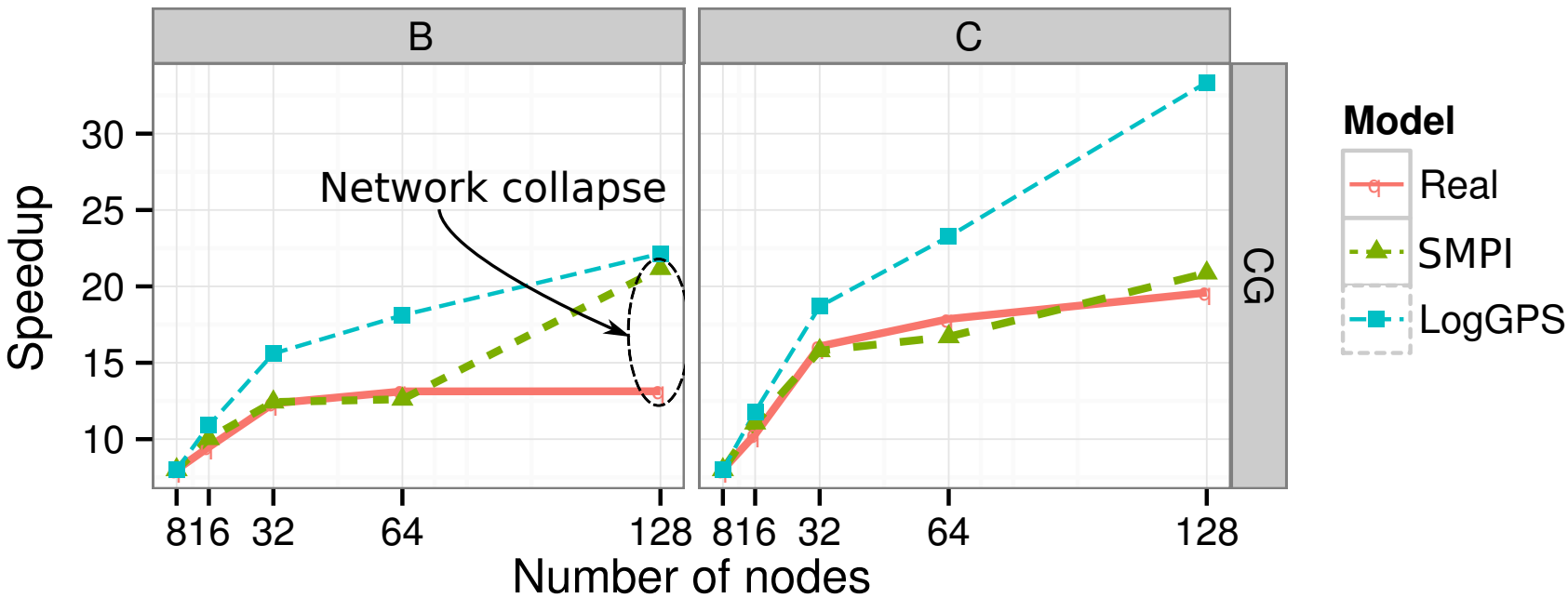


Such accurate modeling is actually critical to obtain decent predictions

Validation: Non-trivial Application Scaling (1)

Experiments run with several NAS parallel benchmarks to (in)validate the model for TCP platform

- Non trivial scaling
- Very good accuracy (especially compared to LogP)

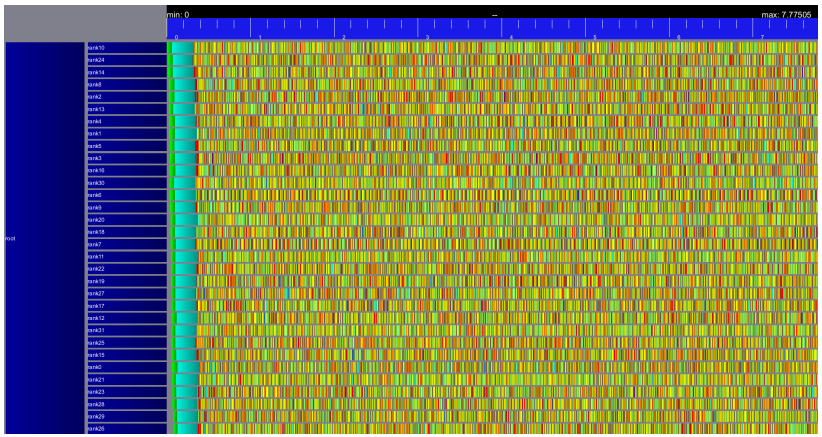


Validation: Non-trivial Application Scaling (1)

Experiments run with several NAS parallel benchmarks to (in)validate the model for TCP platform

- Non trivial scaling
- Very good accuracy (especially compared to LogP)

unless contention drives TCP in a crazy state...

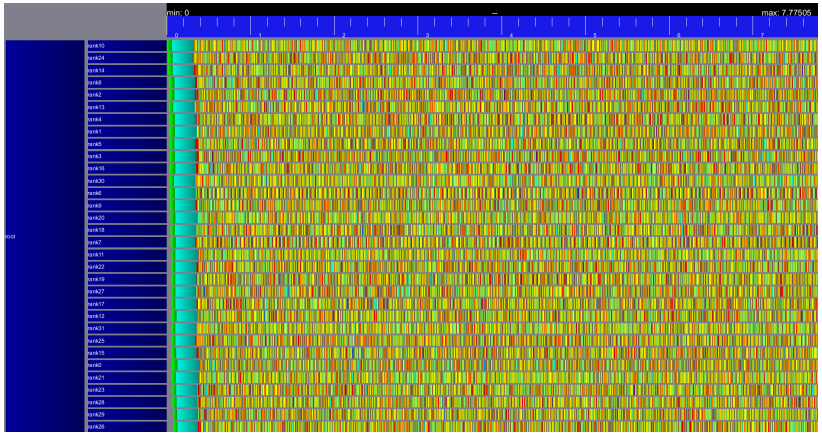


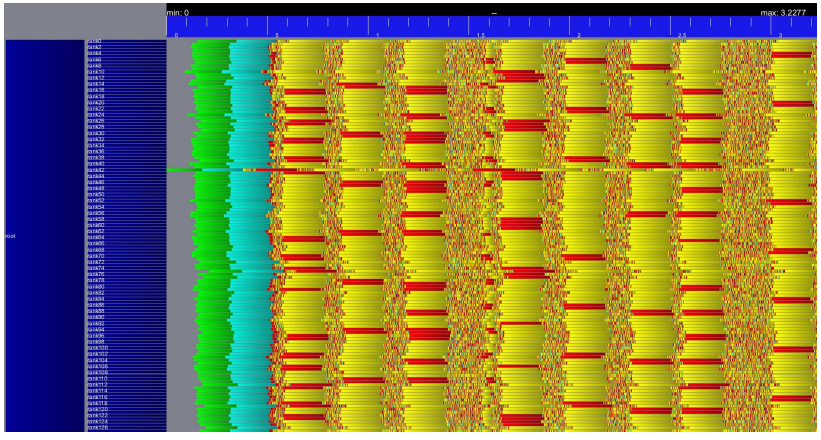
Validation: Non-trivial Application Scaling (1)

Experiments run with several NAS parallel benchmarks to (in)validate the model for TCP platform

- Non trivial scaling
- Very good accuracy (especially compared to LogP)

unless contention drives TCP in a crazy state...



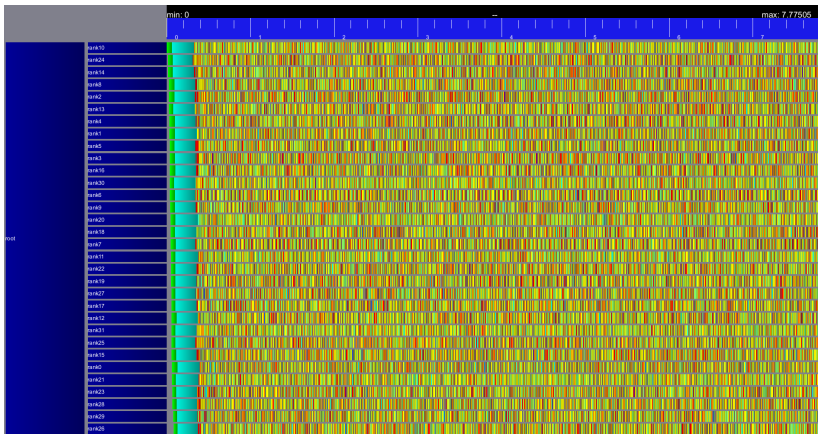


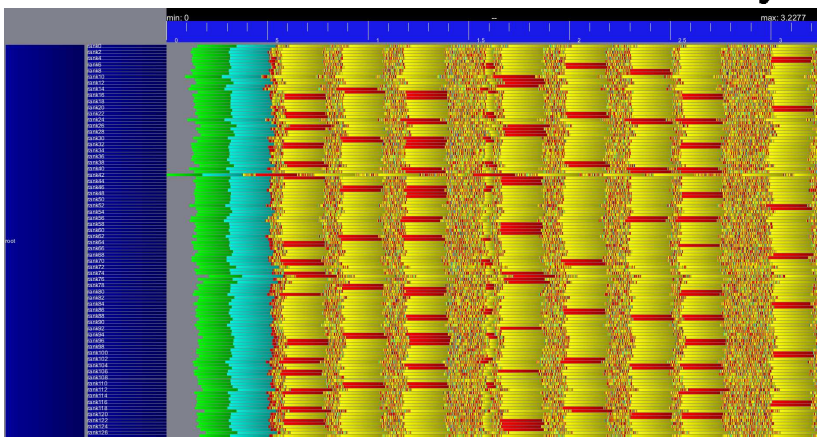
Validation: Non-trivial Application Scaling (1)

Experiments run with several NAS parallel benchmarks to (in)validate the model for TCP platform

- Non trivial scaling
- Very good accuracy (especially compared to LogP)

unless contention drives TCP in a crazy state...





Massive switch packet drops lead to 200ms timeouts in TCP!

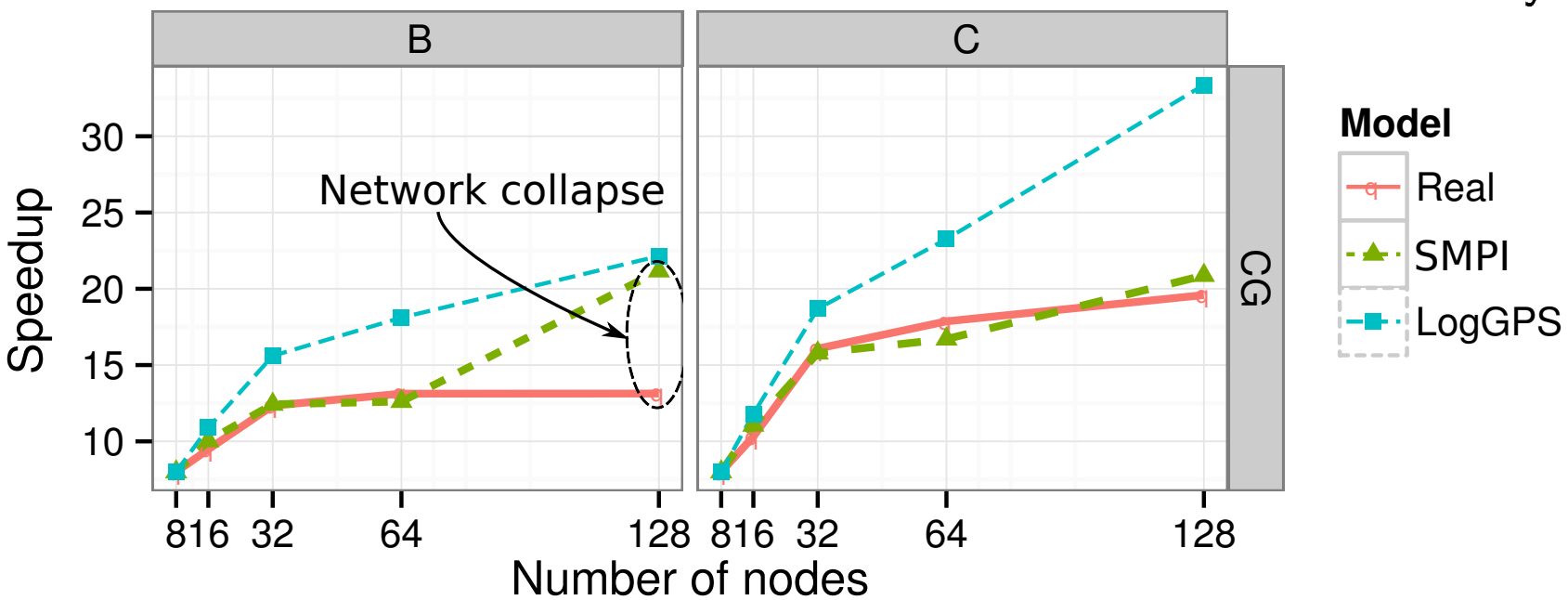
This is a software issue that needs to be fixed (not modeled) in reality

Validation: Non-trivial Application Scaling (1)

Experiments run with several NAS parallel benchmarks to (in)validate the model for TCP platform

- Non trivial scaling
- Very good accuracy (especially compared to LogP)

unless contention drives TCP in a crazy state...

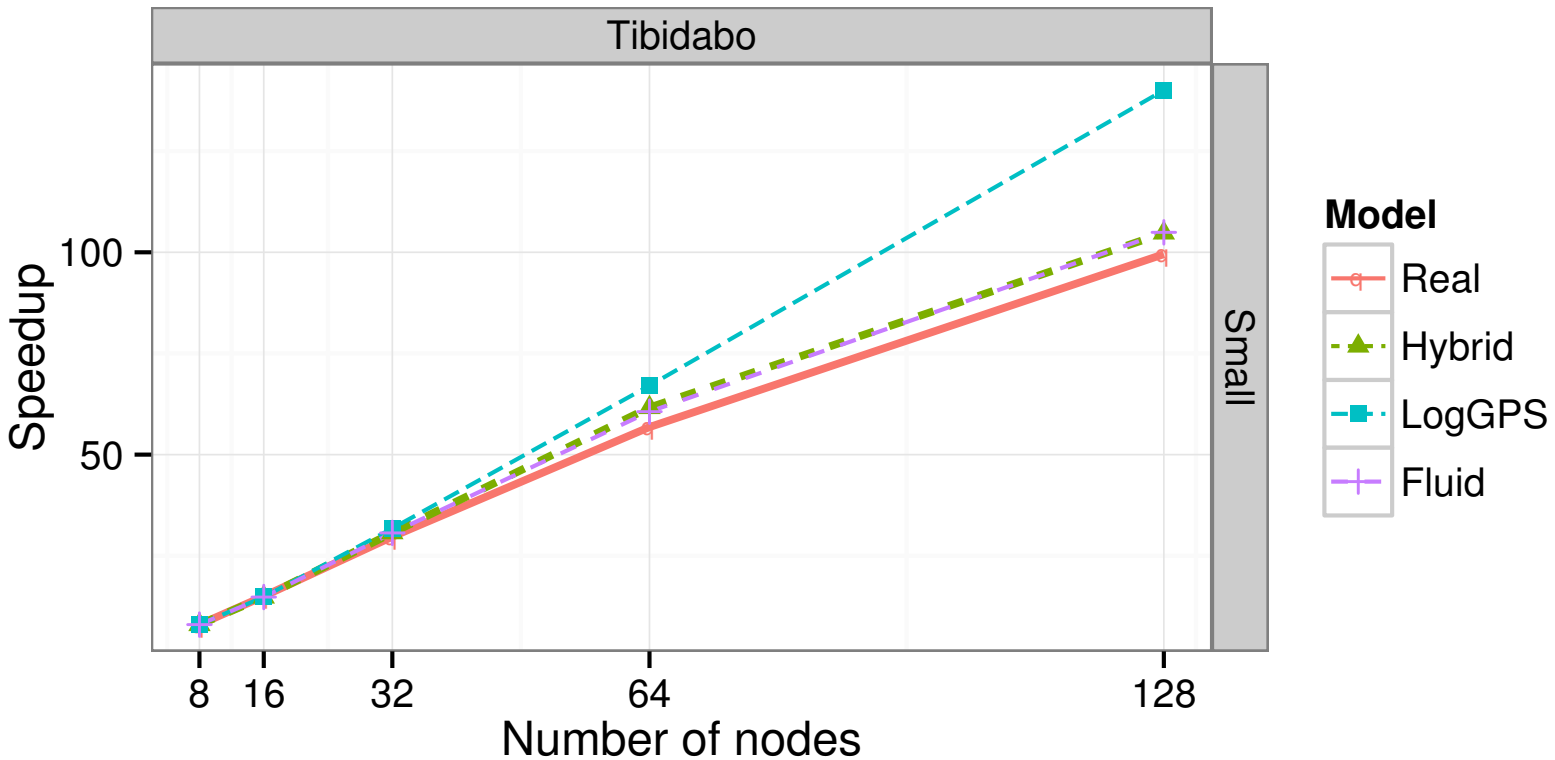


This is a software issue that needs to be fixed (not modeled) in reality

Validation: Non-trivial Application Scaling (2)

Experiments also run using real Physics code (BigDFT, SPECFEM3D) on Tibidabo (ARM cluster prototype)

- The set of collective operations may completely change depending on the instance, hence the need to use online simulation
- Very good accuracy (especially compared to LogP)

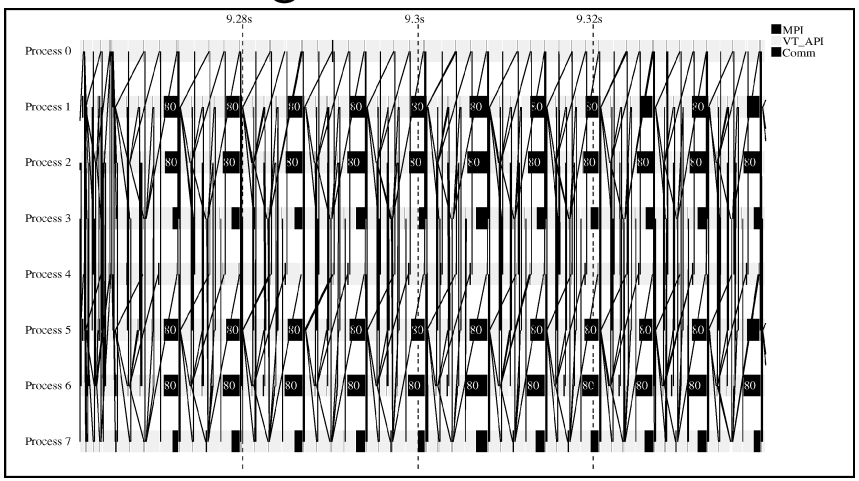


Outline

Remember the Complex Applications ?

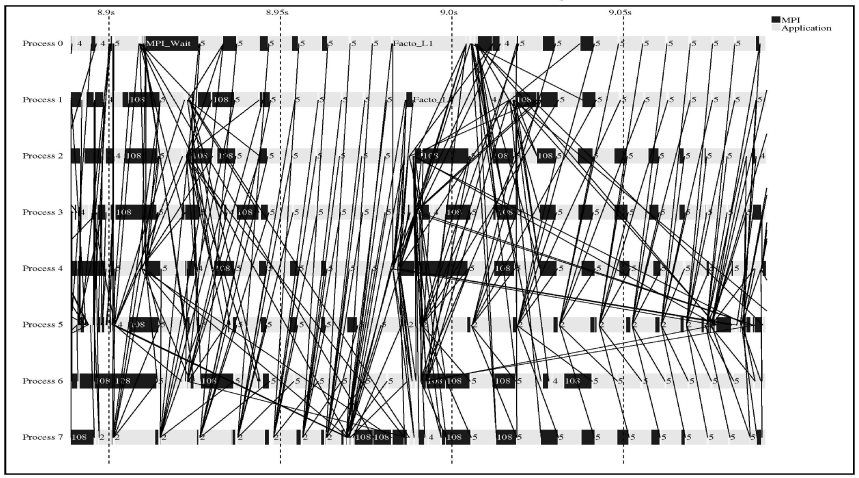
Larger and larger scale hybrid machines $\rightsquigarrow$ Different programming approaches
(e.g., in linear algebra applications)

"Rigid, hand tuned"



SuperLU

Task-based and Dynamic



MUMPS

Analysis and Comparison of Two Distributed Memory Sparse Solvers
Amestoy, Duff, L'excellent, Li. ACM Trans. on Math. Software, Vol. 27, No. 4, 2001.

Multitude of technologies

MPI | OpenMP | Cilk | CUDA | OpenCL | Charm++
KAAPI | StarPU | QUARK | ParSEC | OmpSs | ...

$\rightsquigarrow$ need for a portable **and** efficient way to exploit such architectures

StarPU and SimGrid

StarPU (Inria Bordeaux)

- Dynamic runtime for hybrid architectures (CPU, GPU, MPI)
- Opportunistic scheduling of a task graph guided by resource performance models
- Features both dense and sparse applications. FMM ongoing.

SimGrid

- Scalable Simulation framework for distributed systems
- Sound fluid network models accounting for heterogeneity and contention
- Modeling with threads rather than only trace replay $\rightsquigarrow$ ability to simulate dynamic applications
- Portable, open source and easily extendable

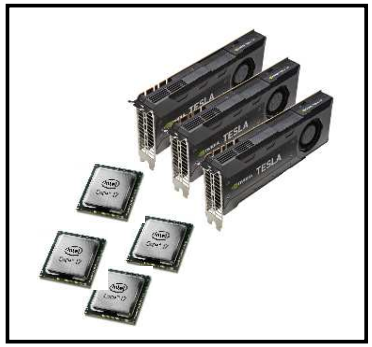
StarPU was ported on top of SimGrid by S. Thibault in 1 day:

- Replace synchronization and thread creation by SimGrid's ones
- Very crude platform model

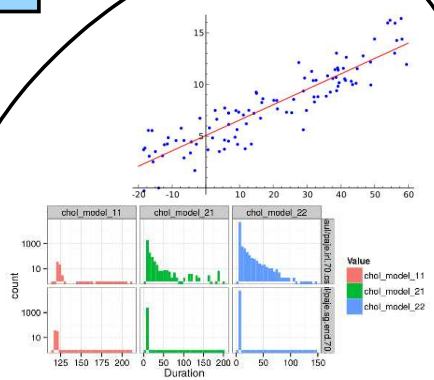
The same approach should be applicable to any task-based runtime

Envisioned Workflow: StarPU+SimGrid

Calibration



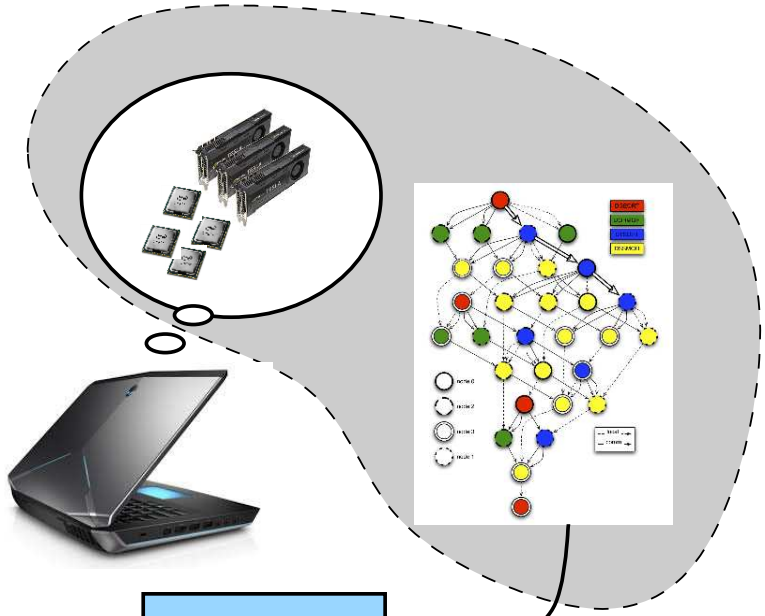
StarPU



Performance Profile

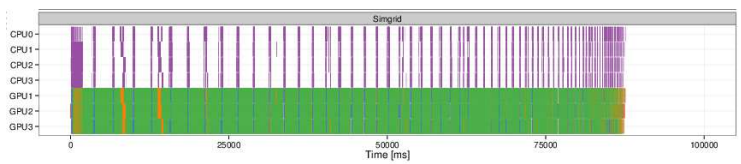
Run once!

Simulation



StarPU

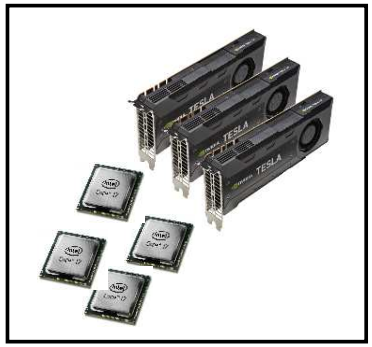
SimGrid



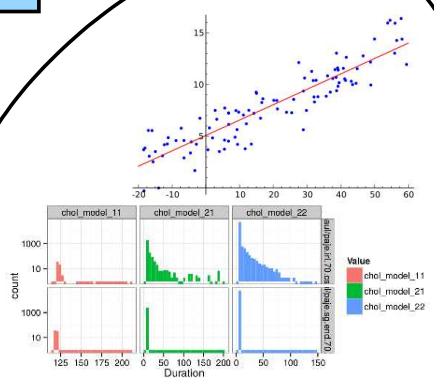
Quickly Simulate Many Times

Envisioned Workflow: StarPU+SimGrid

Calibration



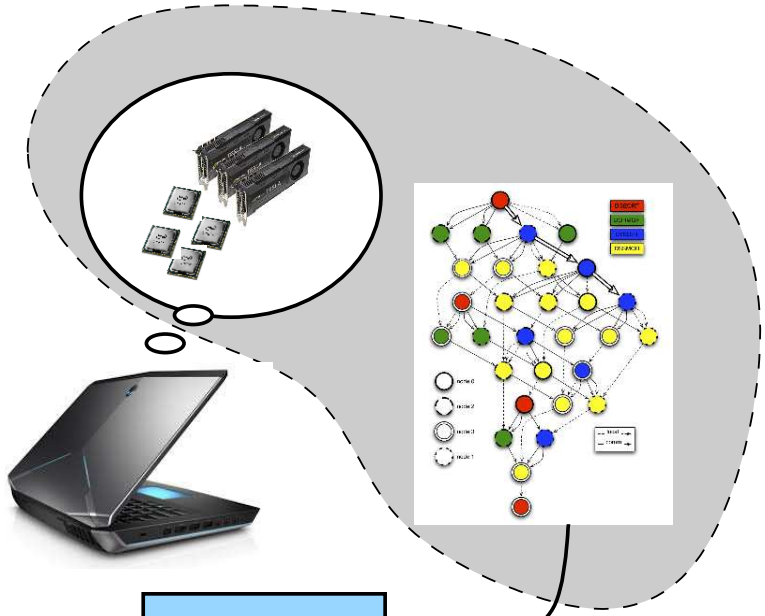
StarPU



Performance Profile

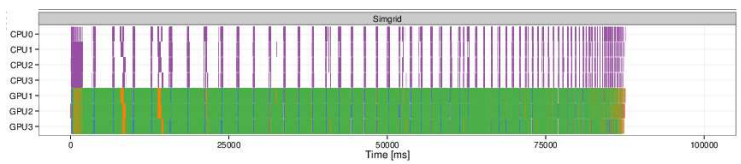
Run once!

Simulation



StarPU

SimGrid



Quickly Simulate Many Times

Implementation Principles

Emulation executing real applications in a synthetic environment, generally slowing down the whole code

Simulation use a performance model to determine how much time a process should wait

- StarPU applications and runtime are *emulated* (real scheduler and dynamic decision guided on StarPU calibration)
- All operations related to thread synchronization, actual computations, memory allocation and data transfer are *simulated* (need for a good kernel and communication model) and *faked*
  - Actual computation results are irrelevant and have no impact on the control flow. Only time matters
  - In SimGrid, all threads run in mutual exclusion (~~polling~~)
- The control part of StarPU is modified to dynamically inject computation and communication tasks into the simulator

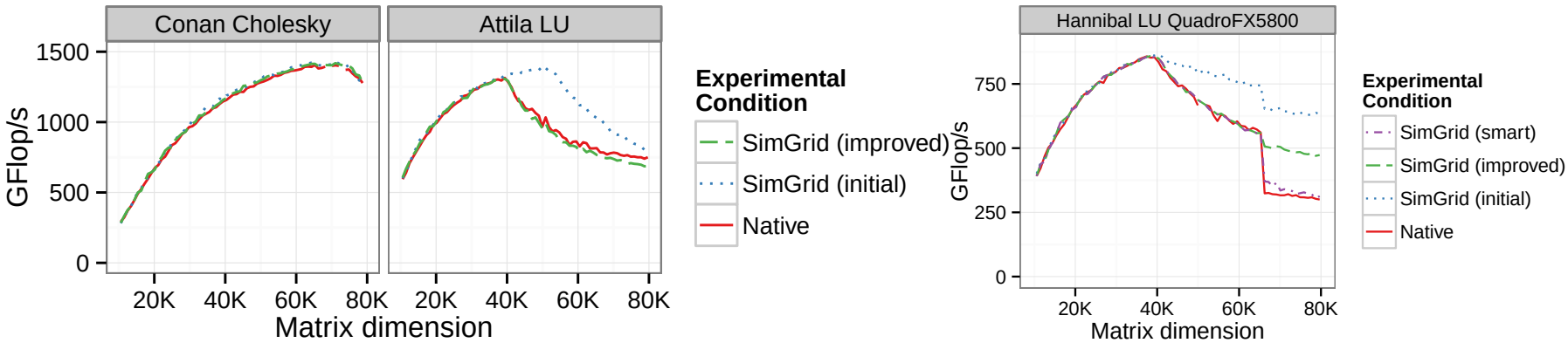
Dense Linear Algebra Applications

- Started with regular dense kernels and a fixed tile size
- Used two different matrix decomposition algorithms:
  1. Cholesky
  2. LU
- Used a wide diversity of machines

Name	Processor	#Cores	Memory	GPUs
hannibal	X5550	2×4	2×24GB	3×QuadroFX5800
attila	X5650	2×6	2×24GB	3×TeslaC2050
mirage	X5650	2×6	2×18GB	3×TeslaM2070
conan	E5-2650	2×8	2×32GB	3×TeslaM2075
frogkepler	E5-2670	2×8	2×16GB	2×K20
pilipili2	E5-2630	2×6	2×32GB	2×K40
idgraf	X5650	2×6	2×36GB	8×TeslaC2050
idchire	E5-4640	24×8	24×31GB	/

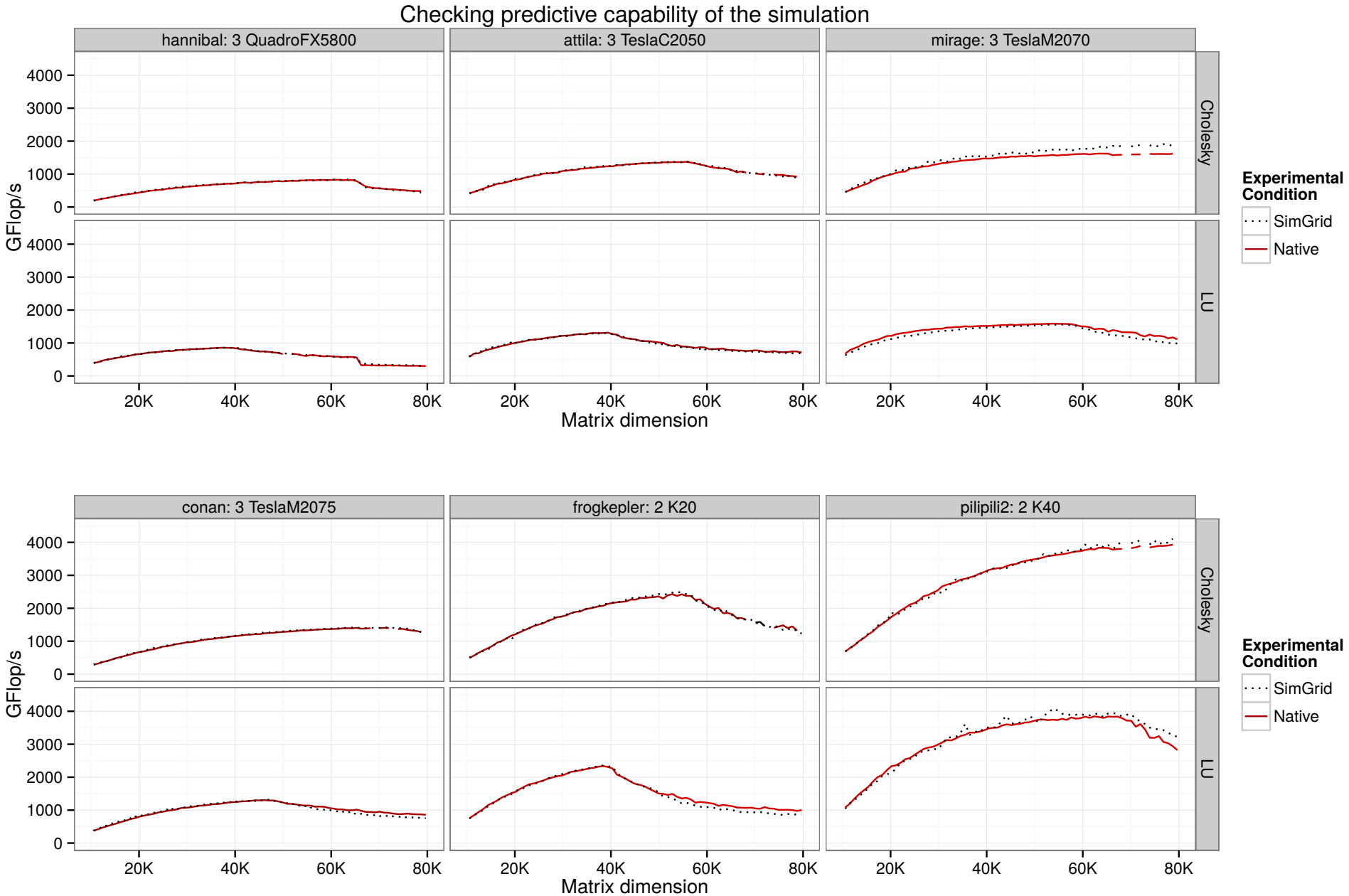
Table: Machines used for the dense linear algebra experiments.

The path to reliable predictions



- Getting excellent results (e.g., Cholesky on Conan) sometimes do not requires much efforts
- But modeling communication heterogeneity, contention, memory operation (and even sometimes hardware/driver peculiarity) is essential
- Try to be as exhaustive as possible...

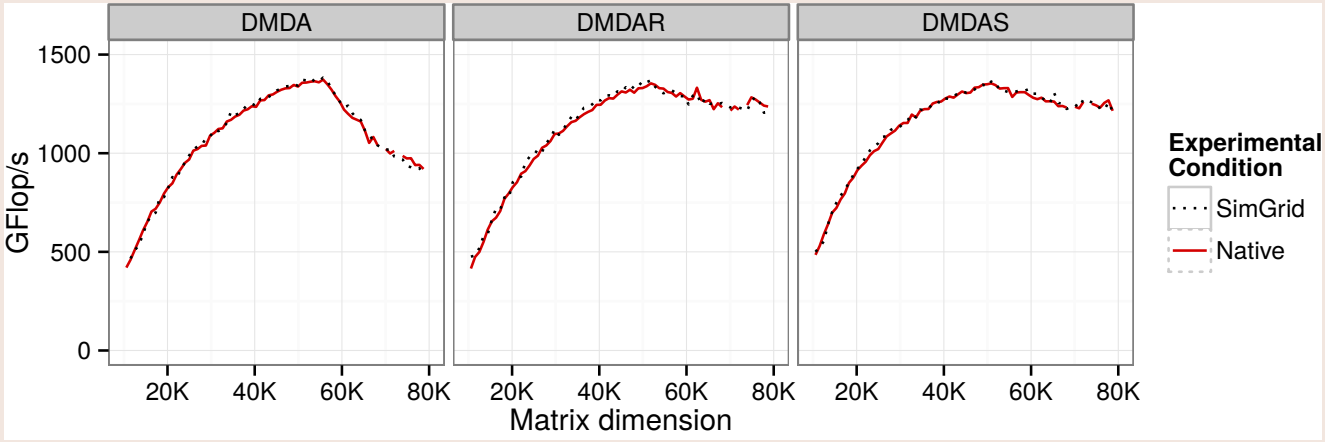
Overview of Simulation Accuracy



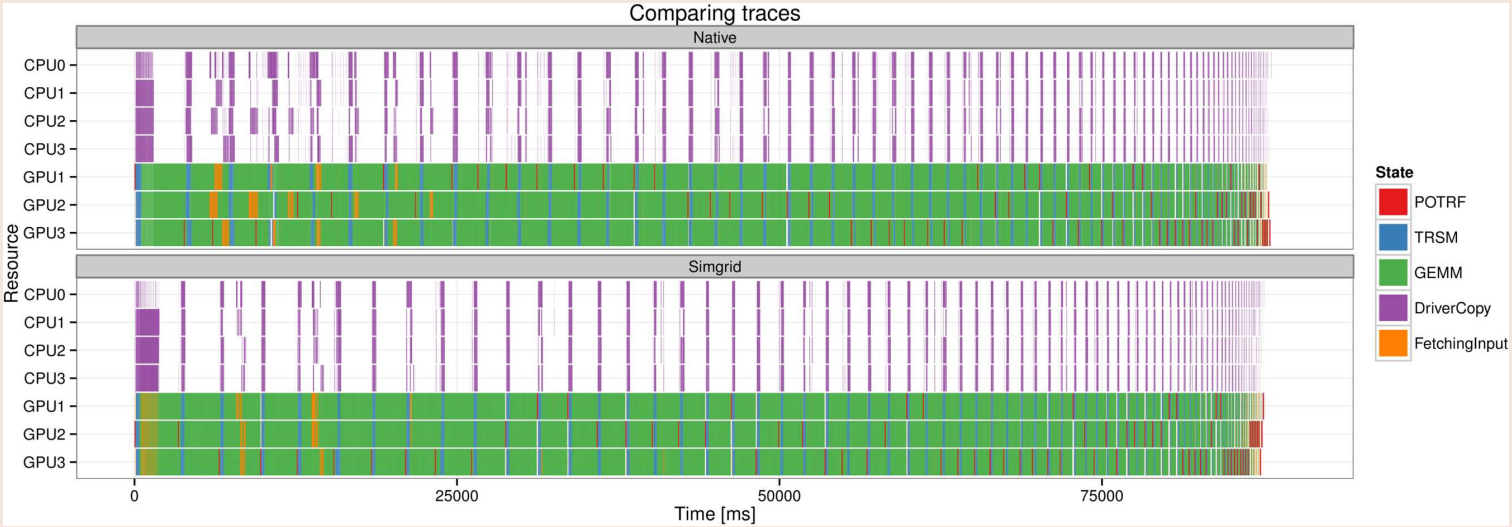
Beyond Simple Graphs

Comparing Different Schedulers

Cholesky on Attila



Investigating Details

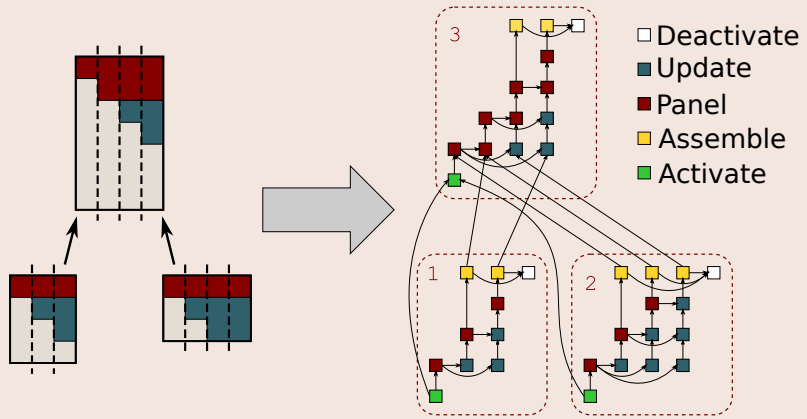


Outline

Simulating Sparse Solvers

qrm_starpu

- QR MUMPS multi-frontal factorization on top of StarPU
  - Tree parallelism: nodes in separate branches can be treated independently
  - Node parallelism: large nodes can be treated by multiple process



- No GPU support (ongoing) in this study, only multi-core

Porting qrm_starpu on top of SimGrid

- Changing `main` for the subroutine
- Changing compilation process
- Careful kernel modeling as matrix dimension keeps changing

Example for Modeling Kernels: GEQRT

- GEQRT(Panel) duration:

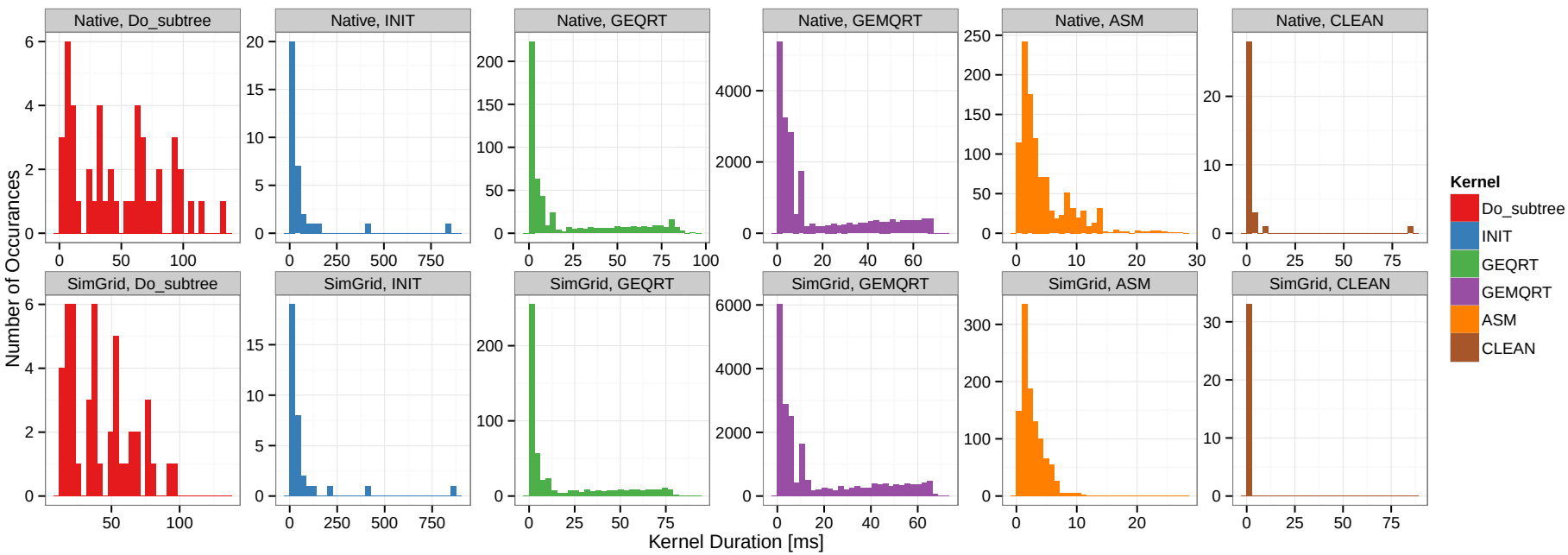
$$T_{\text{GEQRT}} = a + 2b(NB^2 \times MB) - 2c(NB^3 \times BK) + \frac{4d}{3}NB^3$$

- We can do a linear regression based on ad hoc calibration

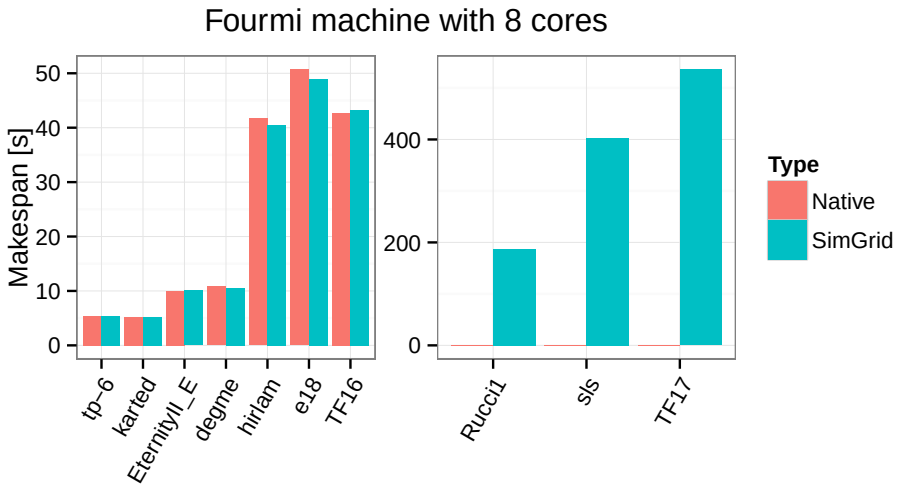
	GEQRT Duration	
NB^3	1.50×10^{-5} $(1.30 \times 10^{-5}, 1.70 \times 10^{-5})$	***
$NB^2 * MB$	5.49×10^{-7} $(5.46 \times 10^{-7}, 5.51 \times 10^{-7})$	***
$NB^3 * BK$	-5.52×10^{-7} $(-5.57 \times 10^{-7}, -5.48 \times 10^{-7})$	***
Constant	-2.49×10^{1} $(-2.83 \times 10^{1}, -2.14 \times 10^{1})$	***
Observations	493	
R^2	0.999	
Note:	*p<0.1; **p<0.05; ***p<0.01	

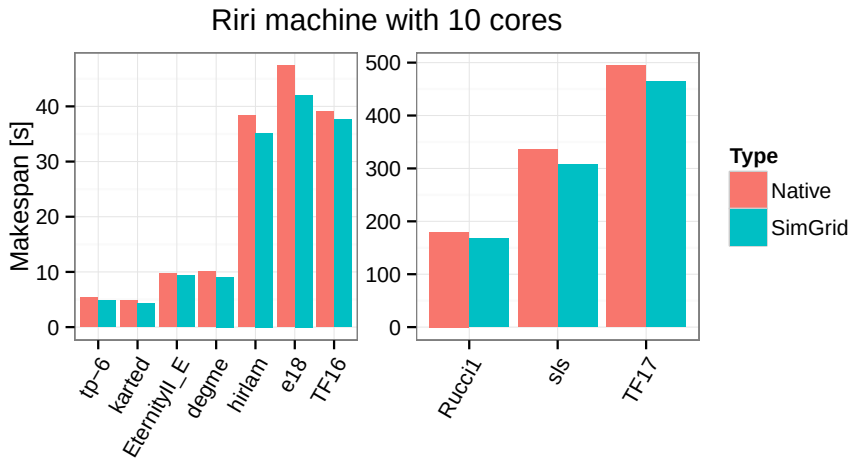
Comparing Kernel Duration Distributions

	Do_subtree	INIT	GEQRT	GEMQRT	ASM
1.	#Flops	#Zeros	*NB*	*NB*	#Coeff
2.	#Nodes	#Assemble	*MB*	*MB*	/
3.	/	/	*BK*	*BK*	/
R^2	0.99	0.99	0.99	0.99	0.86



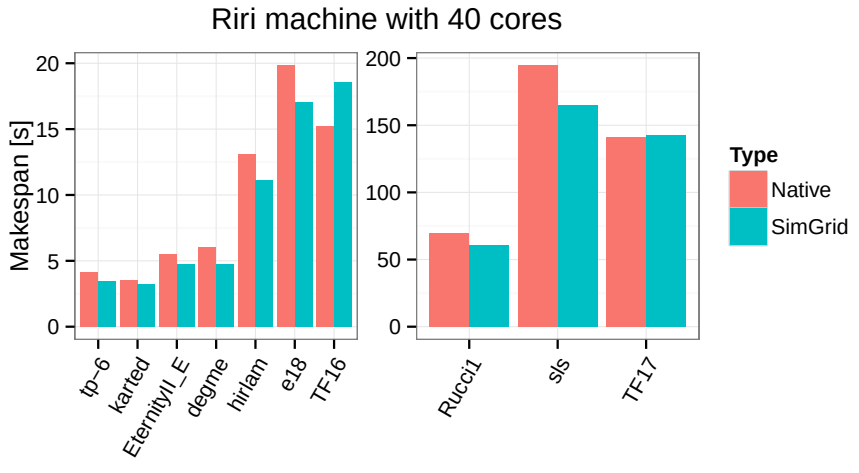
Overview of Simulation Accuracy





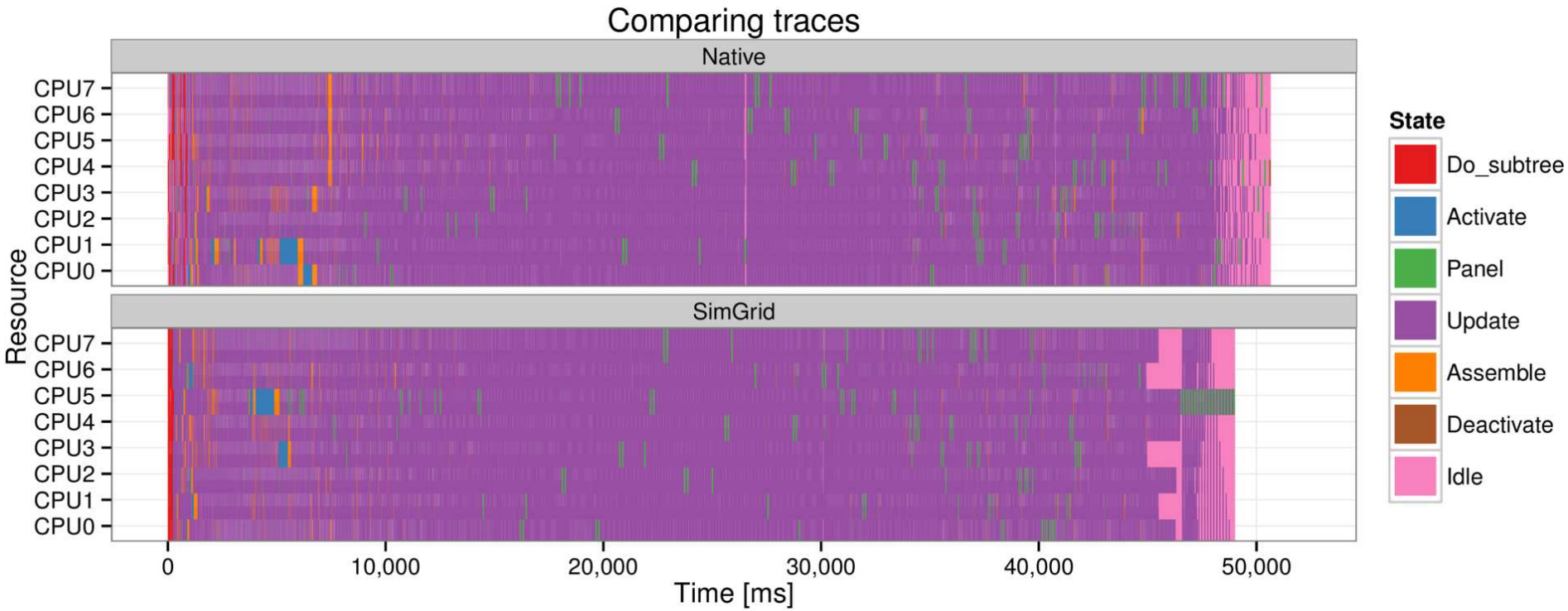
Results in a nutshell

- Most of the time, simulation is slightly optimistic
- With bigger and architecturally more complex machines, error increases



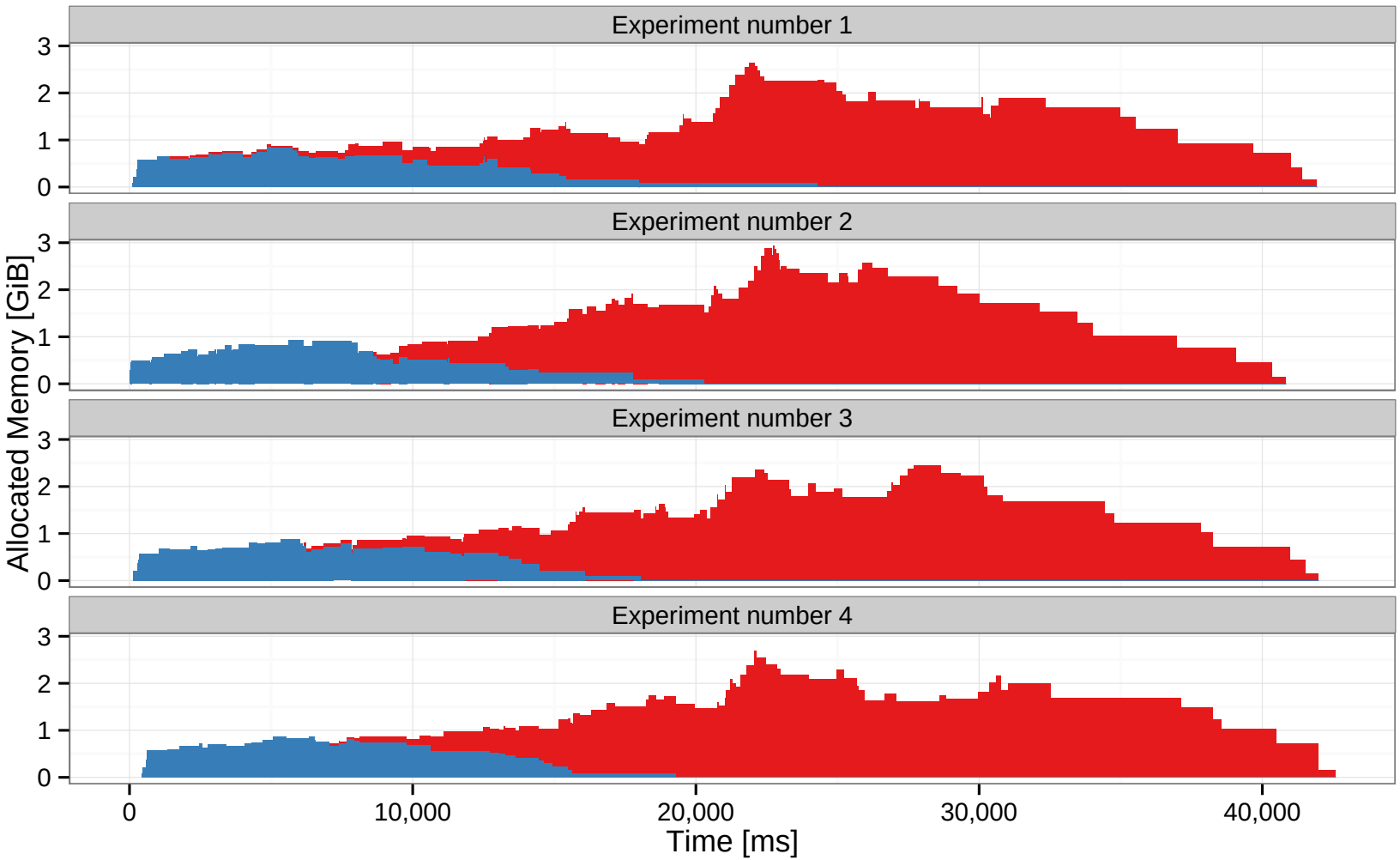
Visual Trace Comparison

- Traces extremely "*close*"
- Simulated makespan slightly optimistic
- Different scheduling at the beginning and at the end



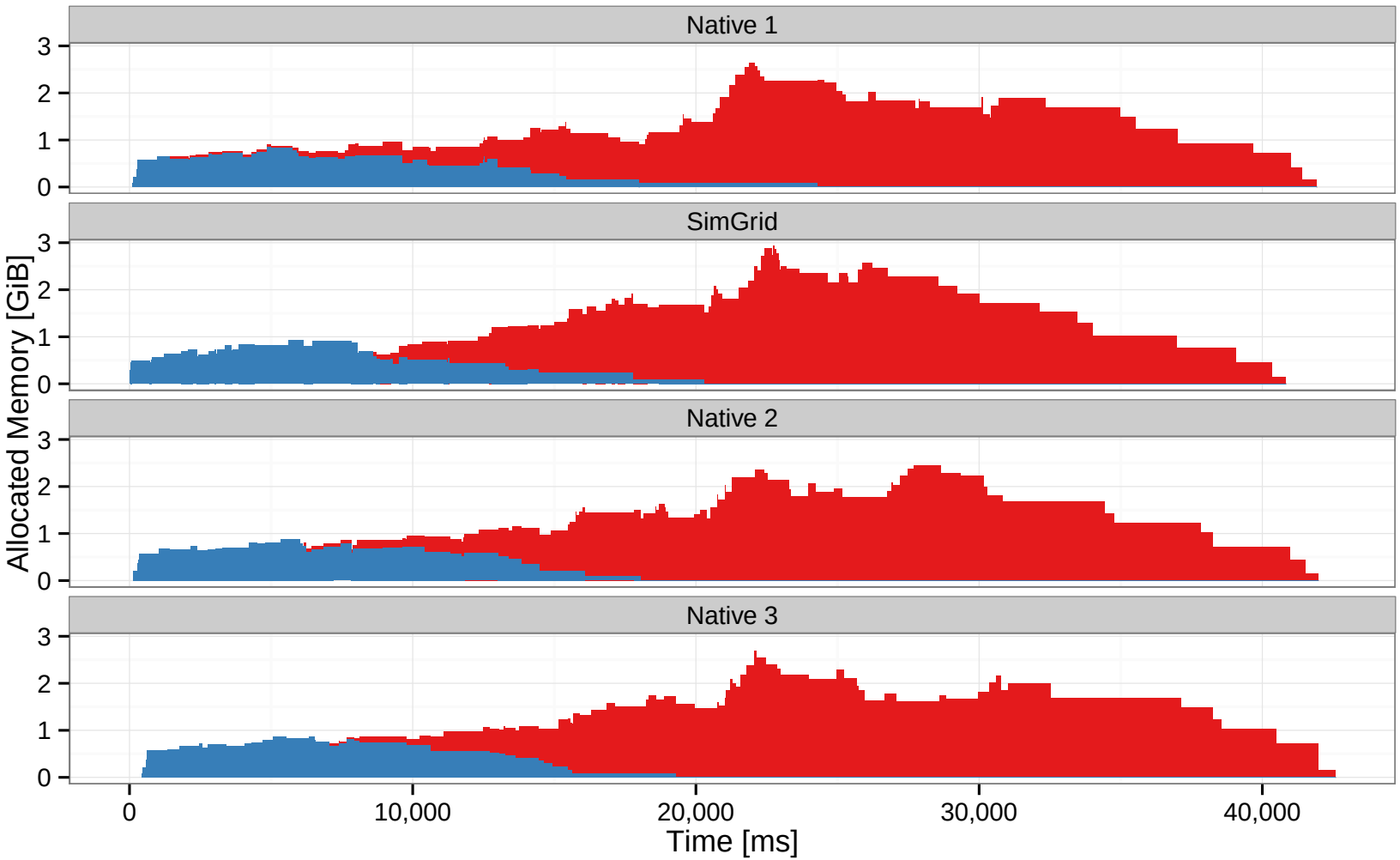
Studying Memory Consumption

- Minimizing memory footprint is very important for such applications
- Remember scheduling is dynamic so consecutive Native experiments have different output



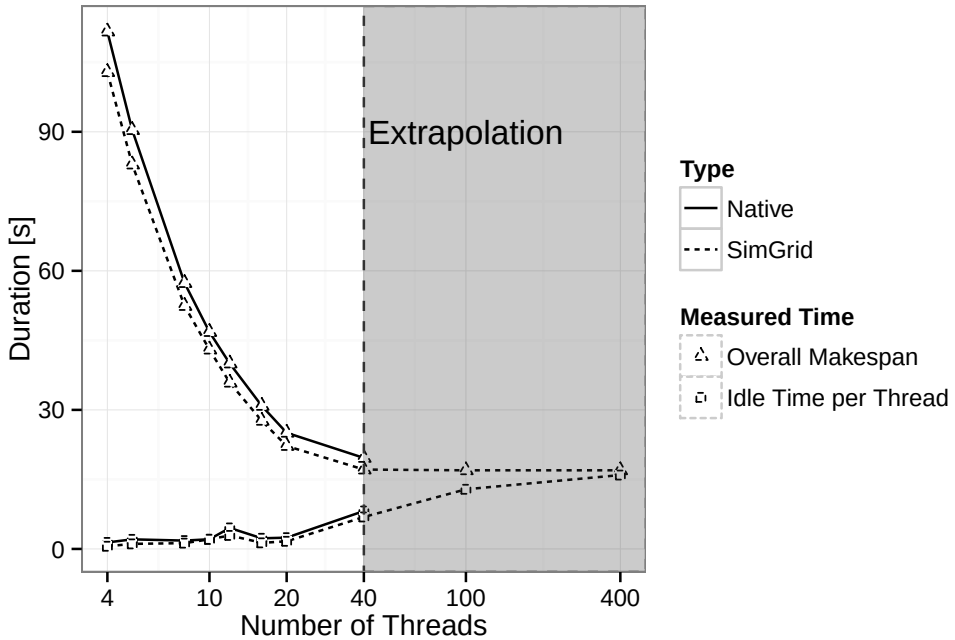
Studying Memory Consumption

- Minimizing memory footprint is very important for such applications
- Remember scheduling is dynamic so consecutive Native experiments have different output



Extrapolating to Larger Machines

- Predicting performance in idealized context
- Studying the parallelization limits of the problem



Regarding simulation speed:

- Quickly and accurately evaluate the impact of various scheduling/application parameters:

qrm_starpu	Cores	RAM	Evaluation	Makespan
Native	40	58.0GiB	157s	141s
SimGrid	1	1.5GiB	57s	142s

Conclusion

- We have now accurate baselines to compare with $\rightsquigarrow$ whenever there is a mismatch, we can question simulation as well as experimental setup:
  - TCP RTO issue
  - Inaccurate platform specifications
  - Flawed MPI optimization
- Hope it will be useful to
  - the Mont-Blanc project
  - you?...
  - BigDFT or Ondes3D developers
- Need to validate this approach on larger platforms, with other network types and topologies (e.g., Infiniband, torus)
- Communication through shared memory is ok, but modeling the interference between memory-bound kernels is really hard
- SMPI and StarPU/SimGrid are open source/science: we put a lot of effort into making it usable

http://simgrid.gforge.inria.fr

Ongoing Work

- ☒ Seamless emulation (`mmap` approach works great)
- ⊟ Modeling IB networks, torus/fat tree topologies
- ⊟ Modeling energy (with A.C. Orgerie)
- ☒ Runtimes for hybrid (CPU+GPU) platforms (StarPU, with S. Thibault) on top of SimGrid
  - Works great for both dense (MAGMA/MORSE) and sparse (QR-MUMPS) linear algebra. Ongoing work for FMM.
  - Large scale hybrid platforms (StarPU-MPI)
- ⊟ Formal verification by exhaustive state space exploration
- ⊟ Computation/communication/memory access interferences
- ☐ OpenMP/pthreads
- ⊟ IOs