SimGrid for HPC

A. Degomme, A. Legrand, J.-F. Méhaut B. Videau,
and the whole SimGrid team

CNRS / Inria / Universities of Cranfield, Grenoble, Lyon, Nancy, Strasbourg

JLPC workshop

Toward Exascale ? A little bif of context

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 (provide sound baseline)
3. Capacity planning (can we save on components? what-if analysis)

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

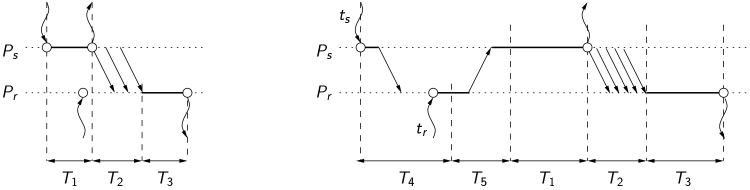
> Flores Lucio, Paredes-Farrera, Jammeh, Fleury, Reed. *Opnet modeler and ns-2: Comparing the accuracy of network simulators for packet-level analysis using a network testbed.* WSEAS Transactions on Computers 2, no. 3 (2003)

- 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 ($k > S$)

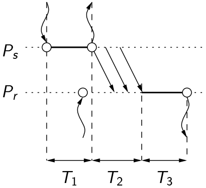
The T_i's are basically continuous linear functions.

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

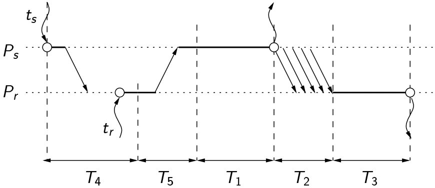
$$T_3 = o + kO_r \qquad T_4 = \max(L+o, t_r - t_s) + o \qquad T_5 = 2o + L$$

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



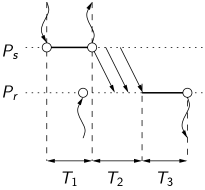
Rendez-vous mode ($k > S$)

The T_i's are basically continuous linear functions.

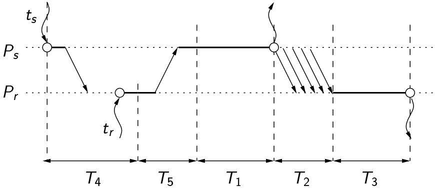
Routine	Condition	Cost
MPI_Send	$k \leqslant S$	T_1
	$k > S$	$T_4 + T_5 + T_1$
MPI_Recv	$k \leqslant S$	$\max(T_1 + T_2 - (t_r - t_s), 0) + T_3$
	$k > S$	$\max(o + L - (t_r - t_s), 0) + o +$ $T_5 + T_1 + T_2 + T_3$
MPI_Isend		o
MPI_Irecv		o

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



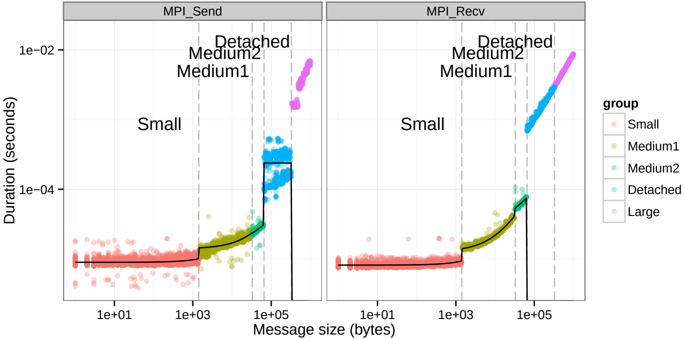
Rendez-vous mode ($k > S$)

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
- I'm not an HPC expert but I doubt this kind of behavior is specific to my *crappy* experimental setup. . .

SimGrid

- Last year, Sanjay mentioned having something intermediate between flit/packet-level simulation and simplistic delay models
- SimGrid in a nutshell:
  - A 13 years old project. Collaboration between France (Inria, CNRS, Univ. Lyon, Nancy, Grenoble, ...), USA (UCSD, U. Hawaii), Great Britain (Cranfield), Austria (Vienna)...
  - Open source/science: we put a lot of effort into making it usable thanks to the support of Inria and ANR
  - Initially focused on Grid settings, we argue that the same tool/techniques can be used for P2P, HPC and more recently cloud
- SimGrid relies on flow-level models that take topology into account.
  - Many naive flow-level models implemented in other simulators are *documented as wrong*
  - Some tools are *validated by general agreement*
  - Some tools present convincing graphs, which are *hardly reproducible*
  - Some tools are *optimistically validated*
  - We have tried hard to invalidate and improve our models for years

Flow-level models

A communication is simulated as a single entity (like a flow in pipes):

$$T_{i,j}(S) = L_{i,j} + S/B_{i,j}, \text{ where } \begin{cases} S & \text{message size} \\ L_{i,j} & \text{latency between } i \text{ and } j \\ B_{i,j} & \text{bandwidth between } i \text{ and } j \end{cases}$$

Estimating $B_{i,j}$ requires to account for interactions with other flows

Flow-level models

A communication is simulated as a single entity (like a flow in pipes):

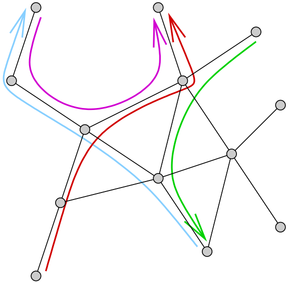
$$T_{i,j}(S) = L_{i,j} + S/B_{i,j}, \text{ where } \begin{cases} S & \text{message size} \\ L_{i,j} & \text{latency between } i \text{ and } j \\ B_{i,j} & \text{bandwidth between } i \text{ and } j \end{cases}$$

Estimating $B_{i,j}$ requires to account for interactions with other flows

Assume steady-state and **share bandwidth** every time a new flow appears or disappears

Setting a set of flows $\mathcal{F}$ and a set of links $\mathcal{L}$

Constraints For all link j: $\displaystyle\sum_{\text{flow } i \text{ using link } j} \rho_i \leqslant C_j$



Flow-level models

A communication is simulated as a single entity (like a flow in pipes):

$$T_{i,j}(S) = L_{i,j} + S/B_{i,j}, \text{ where } \begin{cases} S & \text{message size} \\ L_{i,j} & \text{latency between } i \text{ and } j \\ B_{i,j} & \text{bandwidth between } i \text{ and } j \end{cases}$$

Estimating $B_{i,j}$ requires to account for interactions with other flows

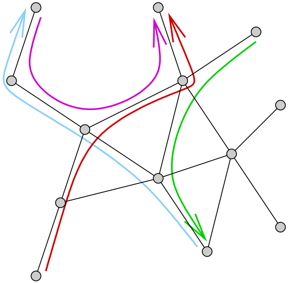
Assume steady-state and **share bandwidth** every time a new flow appears or disappears

Setting a set of flows $\mathcal{F}$ and a set of links $\mathcal{L}$

Constraints For all link j: $\displaystyle\sum_{\textbf{flow } i \textbf{ using link } j} \rho_i \leqslant C_j$

Objective function Maximize $\min_i(\rho_i)$

Other possible objectives ($\sum \log(\rho_i)$, $\sum \arctan(\rho_i)$) but does not help much in practice



Flow-level Models Facts

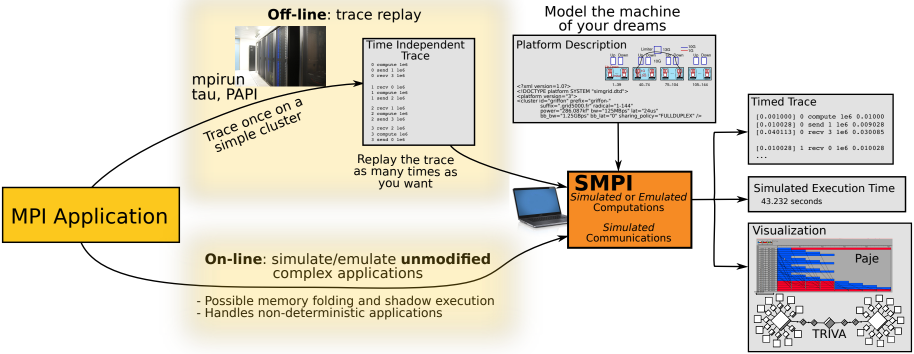
Many different sharing methods can be used and have been evaluated in the context of SimGrid

- **Pros**:
  - rather flexible (add linear limiters whenever you need one)
  - account for network topology
  - account for many non-trivial phenomena (e.g., RTT-unfairness of TCP and even reverse-traffic interferences to some extent)
- **Cons**:
  - ignores protocol oscillations, TCP slow start
  - ignores all transient phases
  - does not model well very unstable situations
  - does not model computation/communication overlap

Most people assume they cannot scale so they're ruled out in this context

Yet, when correctly implemented and optimized, it's better than commonly found implementations of delay-based models

SMPI – Offline vs. Online Simulation

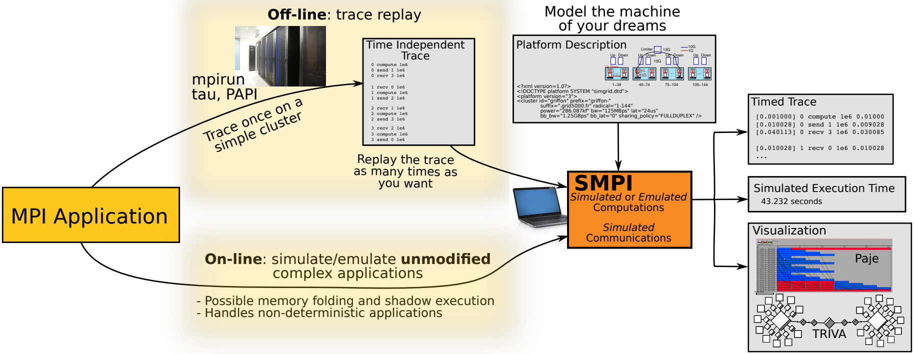


Offline simulation

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

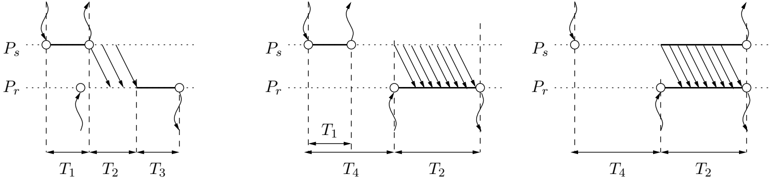


Online simulation

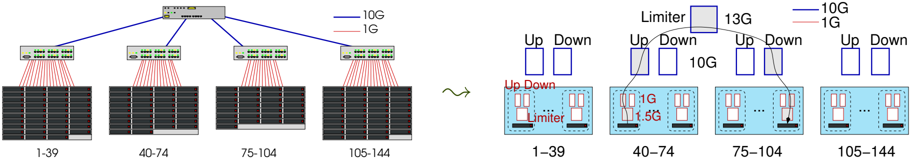
- Directly run the code on top of SimGrid
- Possible memory sharing between simulated processes (reduces memory footprint) and kernel sampling (reduces simulation time)
- Complies with most of the MPICH3 testsuite, compatible with many C F77 and F90 codes (NAS, LinPACK, Sweep3D, BigDFT, SpecFEM3D)

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: 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:

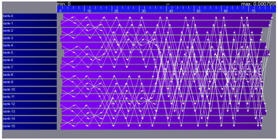
- use simple analytical formulas
- benchmark everything and inject corresponding timing
- trace communication pattern and replay

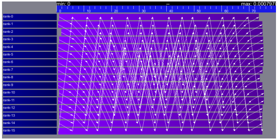
Real MPI implementations have several implementations for each collective and select the right one at runtime

- 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



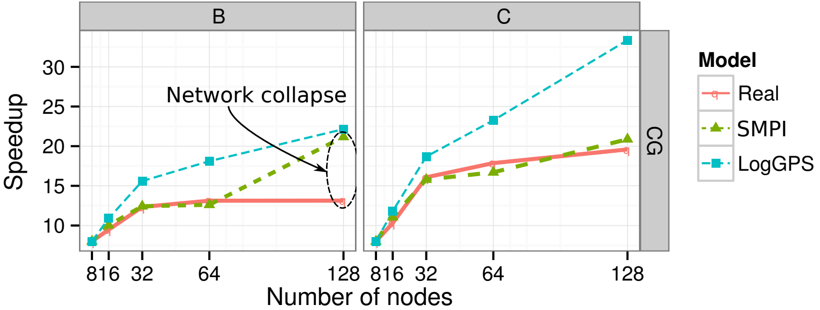


Such accurate modeling is actually critical to obtain decent predictions

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)

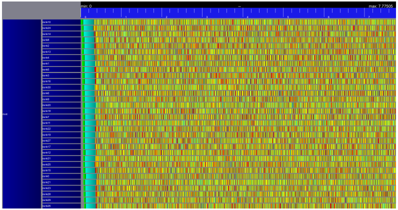


Validation: Non-trivial Application Scaling (1)

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...

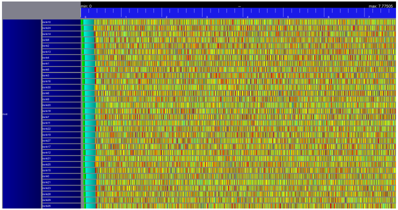


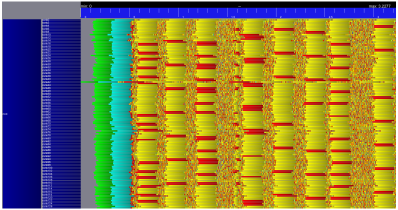
Validation: Non-trivial Application Scaling (1)

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...



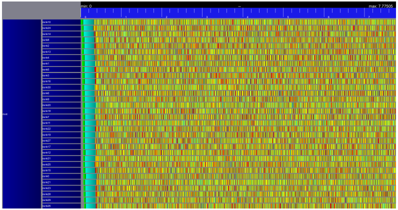


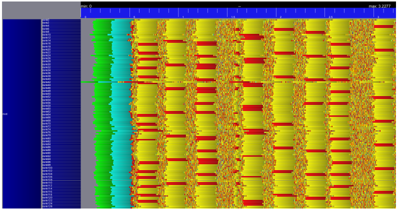
Validation: Non-trivial Application Scaling (1)

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!

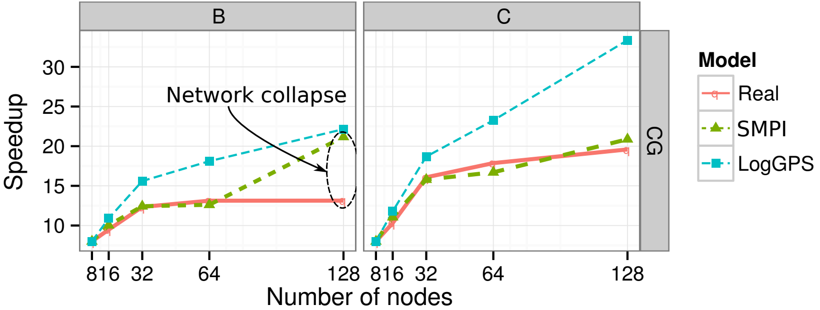
This is a software issue that needs to be fixed (not modeled) in reality

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. . .

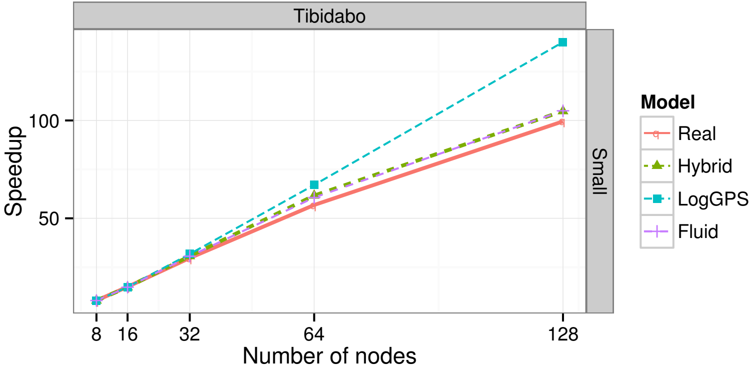


This is a software issue that needs to be fixed (not modeled) in reality

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)

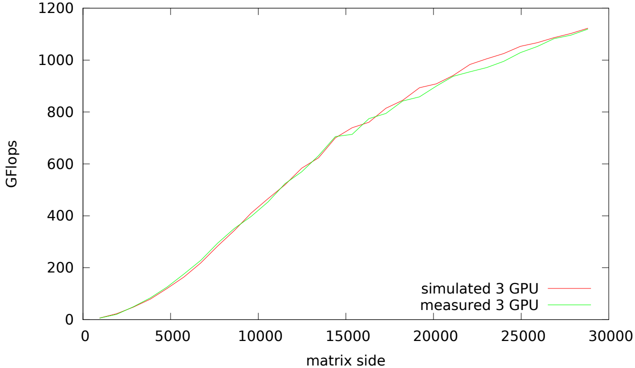


SMPI Conclusion

- 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
- 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
- Hope it will be useful to
  - the Mont-Blanc project
  - colleagues from TACC
  - the BigDFT developers
  - you?...
- Will work with Eric and Sanjay to see whether it can be used to understand application performance. Maybe reuse Torsten's student work on torus routing in SimGrid

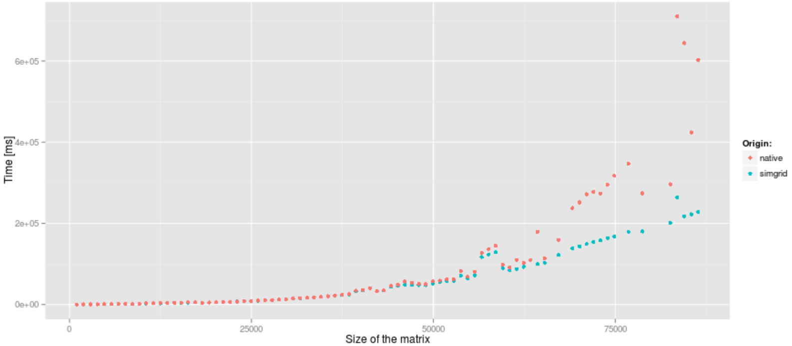
Related Ongoing Work: StarPU (1/2)

- Last year, at SC, S. Thibault (RunTime/Bordeaux) and E. Agullo (HiePac Bordeaux) expressed their interest in using simulation to
  - save experimental time
  - extrapolate performances
  - allow better back-to-back comparisons
- Within a day, S. Thibault ported StarPU on top of Simgrid and within two days had early very promising comparisons of *cholesky* scalability on a 3 GPU node



Related Ongoing Work: StarPU (1/2)

- Last year, at SC, S. Thibault (RunTime/Bordeaux) and E. Agullo (HiePac Bordeaux) expressed their interest in using simulation to
  - save experimental time
  - extrapolate performances
  - allow better back-to-back comparisons
- Within a day, S. Thibault ported StarPU on top of Simgrid and within two days had early ~~very promising~~ comparisons of *cholesky* scalability on a 3 GPU node

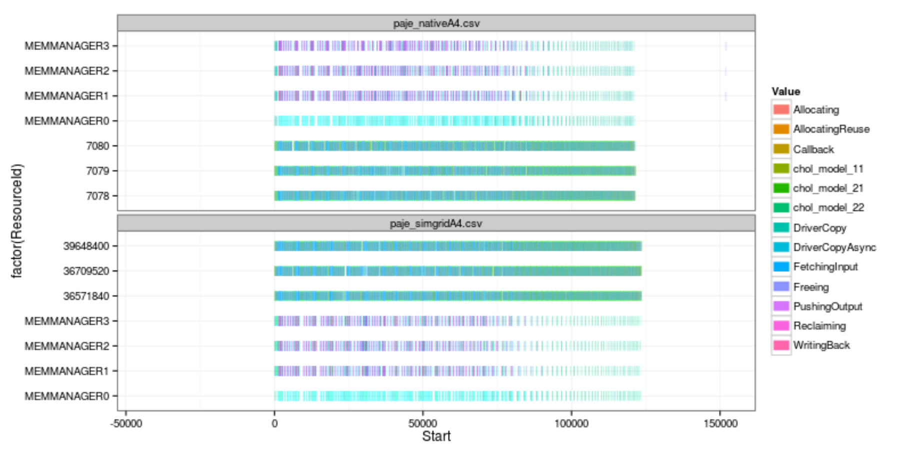


Related Ongoing Work: StarPU (2/2)

L. Stanisic (Mescal/Grenoble) investigated this:

- Serialize transfers between RAM and GPU
- Distinguish performances of RAM $\leftrightarrow$ GPU, GPU0 $\leftrightarrow$ GPU1
- Implement mutual exclusions of transfers
- Account for allocation/de-allocation time
- Use same, carefully chosen values for CUDA memory limit
- Compute better histograms of kernel performances
- Avoid Quadro FX 5800 GPU: it is bogus for large `memcopy2D` transfers

Related Ongoing Work: StarPU (2/2)



Now we need to validate this on wide variety of machines with MAGMA workload (later QRMUMPS).

Related Ongoing Work: Model Checking

M. Quinson and M. Guthmuller (Algorille/Nancy) were crazy enough to try to model-check real C code

- Relies on libunwind and DWARF
- State equality is based on stack inspection and get rid of irrelevant differences (e.g., padding bytes, freed variables, . . .)

They now manage to model-check MPI programs and tried the MPICH 3.04 test suite:

- found several deadlocks in particular in collectives examples ($\text{MPI}_{\text{Allreduce}}$, $\text{MPI}_{\text{Gather}}$, $\text{MPI}_{\text{Allgather}}$ or $\text{MPI}_{\text{Alltoall}}$)
- may be small issues in the test themselves or real issues in the collective algorithms

This work is motivated by Franck's work on communication determinism in parallel HPC applications