Modeling and Simulation of Hybrid Systems

Luka Stanisic Samuel Thibault Arnaud Legrand Brice Videau
Jean-François Méhaut

CNRS/Inria/University of Grenoble, France

University of Bordeaux/Inria, France

JointLab Workshop, Sophia-Antipolis
June 9, 2014

Context

- Hybrid machines with both multi-core CPUs and GPUs are now commonplace and need to be efficiently exploited.
- Obtaining portable performances across architectures is extremely challenging $\leadsto$ adaptive task-based runtime (StarPU, StarSs, QUARK, DAGuE, KAAPI, ...)

Context

- Hybrid machines with both multi-core CPUs and GPUs are now commonplace and need to be efficiently exploited.
- Obtaining portable performances across architectures is extremely challenging $\leadsto$ adaptive task-based runtime (StarPU, StarSs, QUARK, DAGuE, KAAPI, ...)

Typical Performance Evaluation Issues

- Checking the impact of a parameter/algorithm modification (granularity, scheduling, application structure, ...) is time consumming and wastes precious resources
- Debugging is even worse, errors are generally hard to reproduce
- Machine misconfiguration can be difficult to detect
- Making sure new feature work on a wide variety of setups
- Extrapolate behavior on larger/unavailable machines

Context

- Hybrid machines with both multi-core CPUs and GPUs are now commonplace and need to be efficiently exploited.
- Obtaining portable performances across architectures is extremely challenging $\rightsquigarrow$ adaptive task-based runtime (StarPU, StarSs, QUARK, DAGuE, KAAPI, ...)

Typical Performance Evaluation Issues

- Checking the impact of a parameter/algorithm modification (granularity, scheduling, application structure, ...) is time consumming and wastes precious resources
- Debugging is even worse, errors are generally hard to reproduce
- Machine misconfiguration can be difficult to detect
- Making sure new feature work on a wide variety of setups
- Extrapolate behavior on larger/unavailable machines

Possible solution: Simulation

StarPU + SimGrid

StarPU

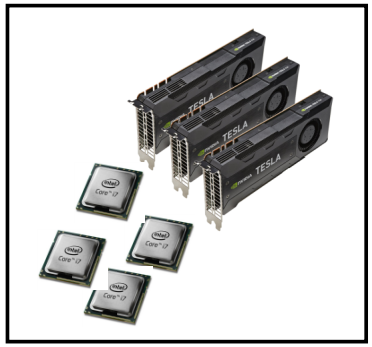
Dynamic runtime for hybrid architectures: opportunistic scheduling of a task graph guided by resource performance models

SimGrid

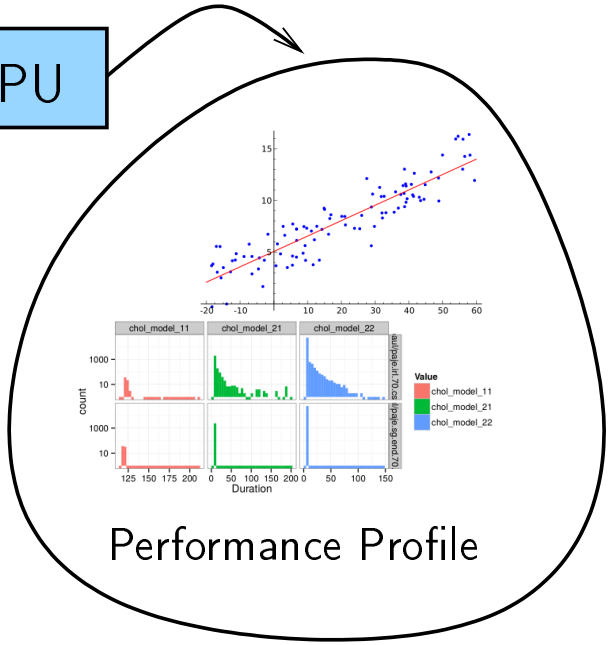
Versatile simulator of distributed systems

Workflow

Calibration



StarPU

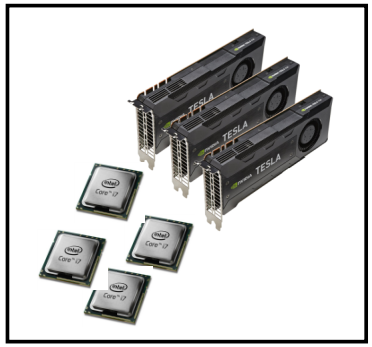


Performance Profile

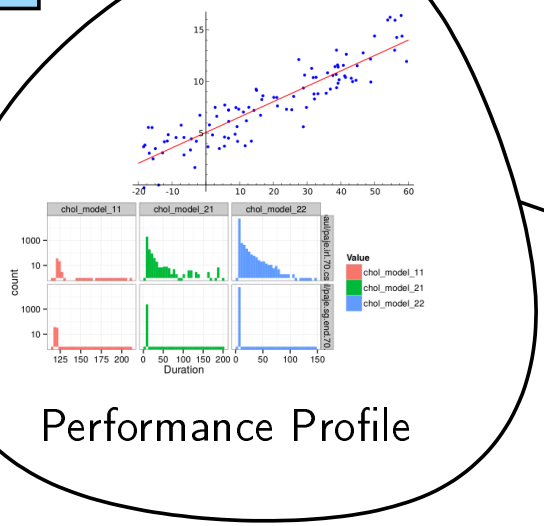
Run once!

Workflow

Calibration



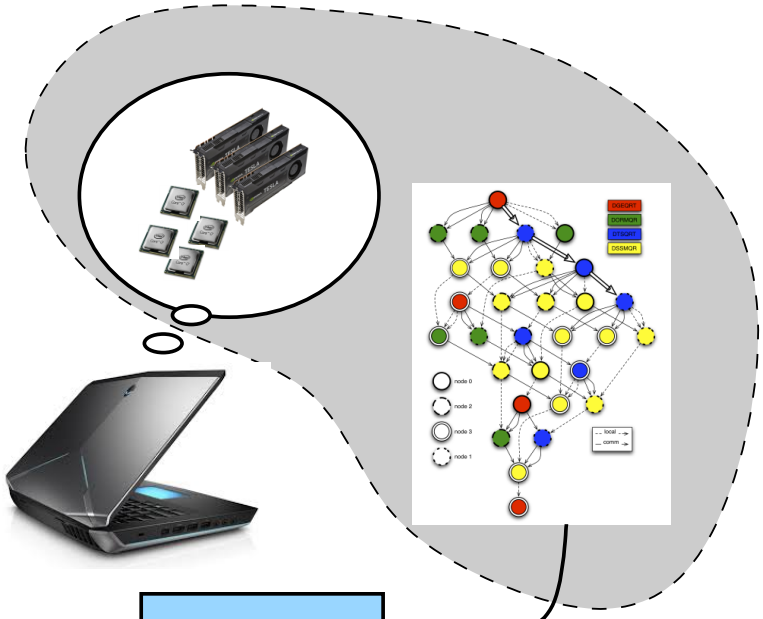
StarPU



Performance Profile

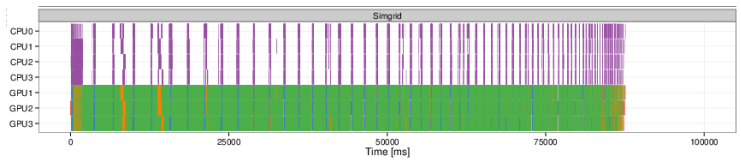
Run once!

Simulation



StarPU

SimGrid



Quickly Simulate Many Times

StarPU + SimGrid Implementation

StarPU

Dynamic runtime for hybrid architectures: opportunistic scheduling of a task graph guided by resource performance models

SimGrid

Versatile simulator of distributed systems

Implementation:

- StarPU applications and runtime are *emulated*
- All operations related to thread synchronization, actual computations and data transfer are *simulated*
- Control part of StarPU is modified to dynamically inject computation and communication tasks into the simulator
- StarPU calibration and platform description is used by SimGrid

(In)Validation Experimental Protocol

Machines:

Name	Processor	Number of Cores	GPUs
hannibal	Intel Xeon X5550	2×4	3×QuadroFX5800
attila	Intel Xeon X5650	2×6	3×TeslaC2050
conan	Intel Xeon E5-2650	2×8	3×TeslaM2075
frogkepler	Intel Xeon E5-2670	2×8	2×K20
mirage	Intel Xeon X5650	2×6	3×TeslaM2070
froggy	-	-	K40

Applications: Cholesky and LU implementations from StarPU

Protocol:

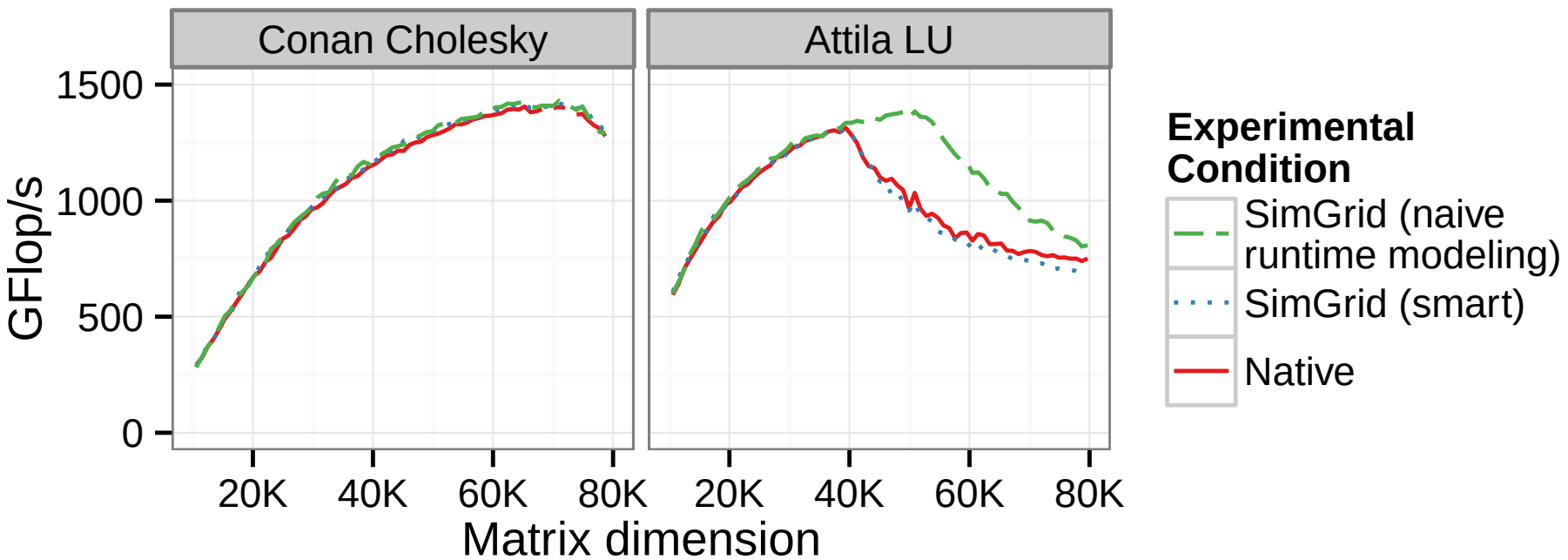
1. Calibrate model on target machine
2. Run StarPU over SimGrid on laptop
3. Run StarPU on target machine
4. Compare results (3) with predictions (2)

Modeling Runtime

- Inserting delays for:
  1. Process synchronizations
  2. Memory allocations of CPU or GPU
  3. Submission of data transfer requests

Modeling Runtime

- Inserting delays for:
  1. Process synchronizations
  2. Memory allocations of CPU or GPU
  3. Submission of data transfer requests
- Taking GPU memory size into account is crucial
- Conan: TeslaM2075 4GB Attila: TeslaC2050 3GB

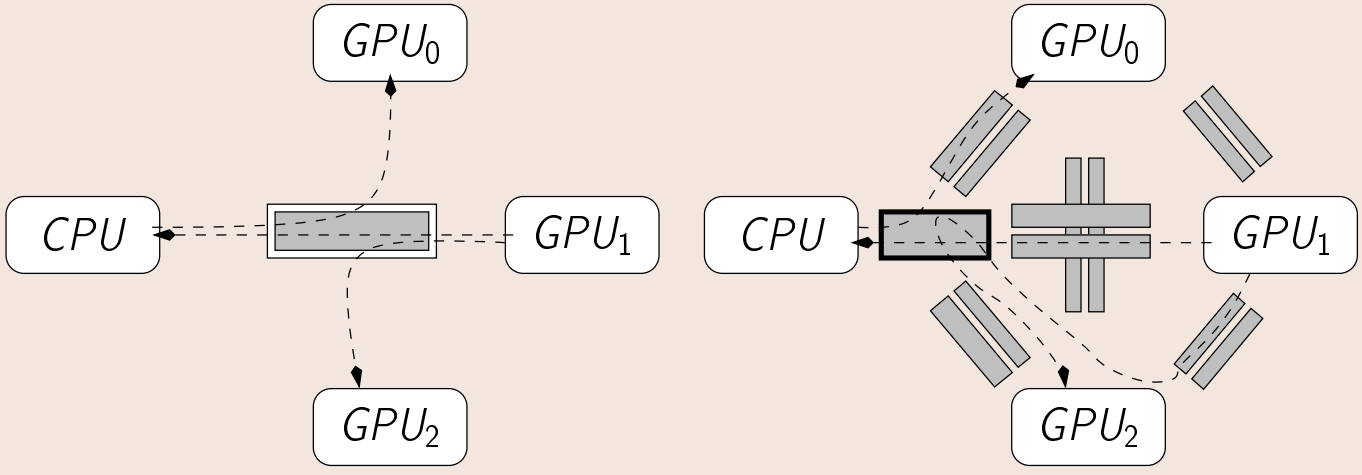


Modeling Communication

- Due to the relatively low bandwidth of the PCI bus, applications spend a lot of time transferring data
- Components of hybrid platforms have differing characteristics
- Correctly modeling communication is of primary importance

Why SimGrid?

1. Modeling with threads rather than states and transitions
2. Basic and flexible contention model



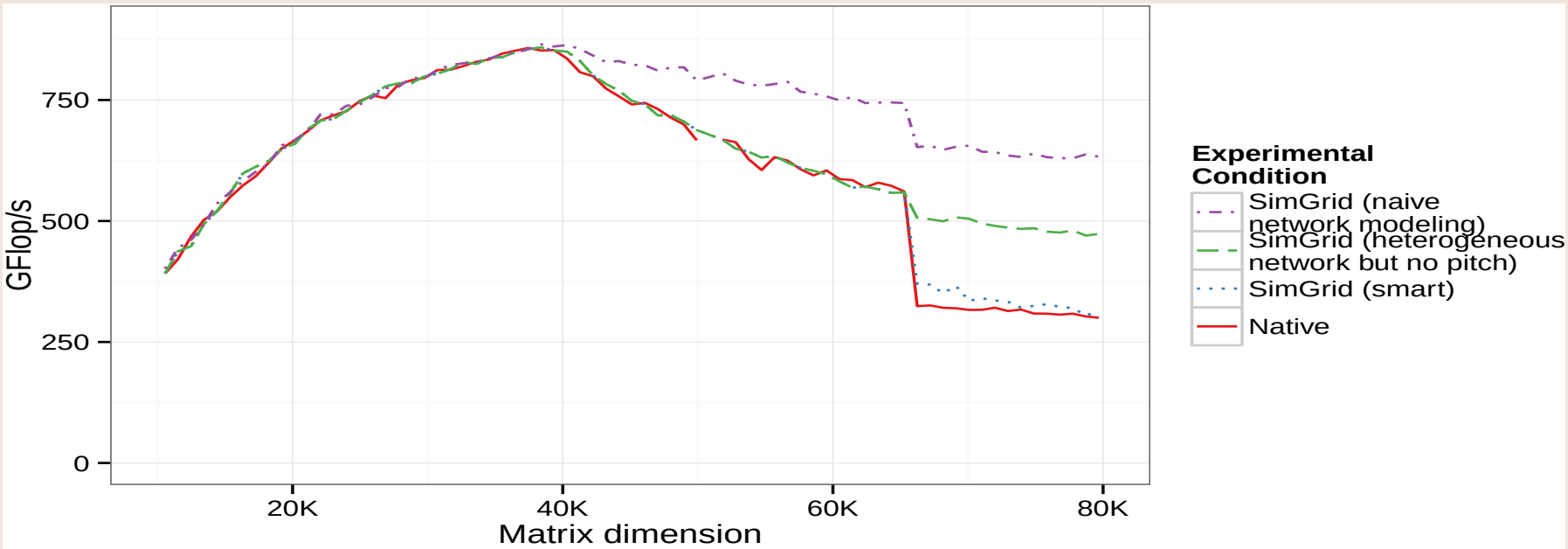
(a) Crude modeling (b) More elaborated modeling

Modeling Communication

- Due to the relatively low bandwidth of the PCI bus, applications spend a lot of time transferring data
- Components of hybrid platforms have differing characteristics
- Correctly modeling communication is of primary importance

Why SimGrid?

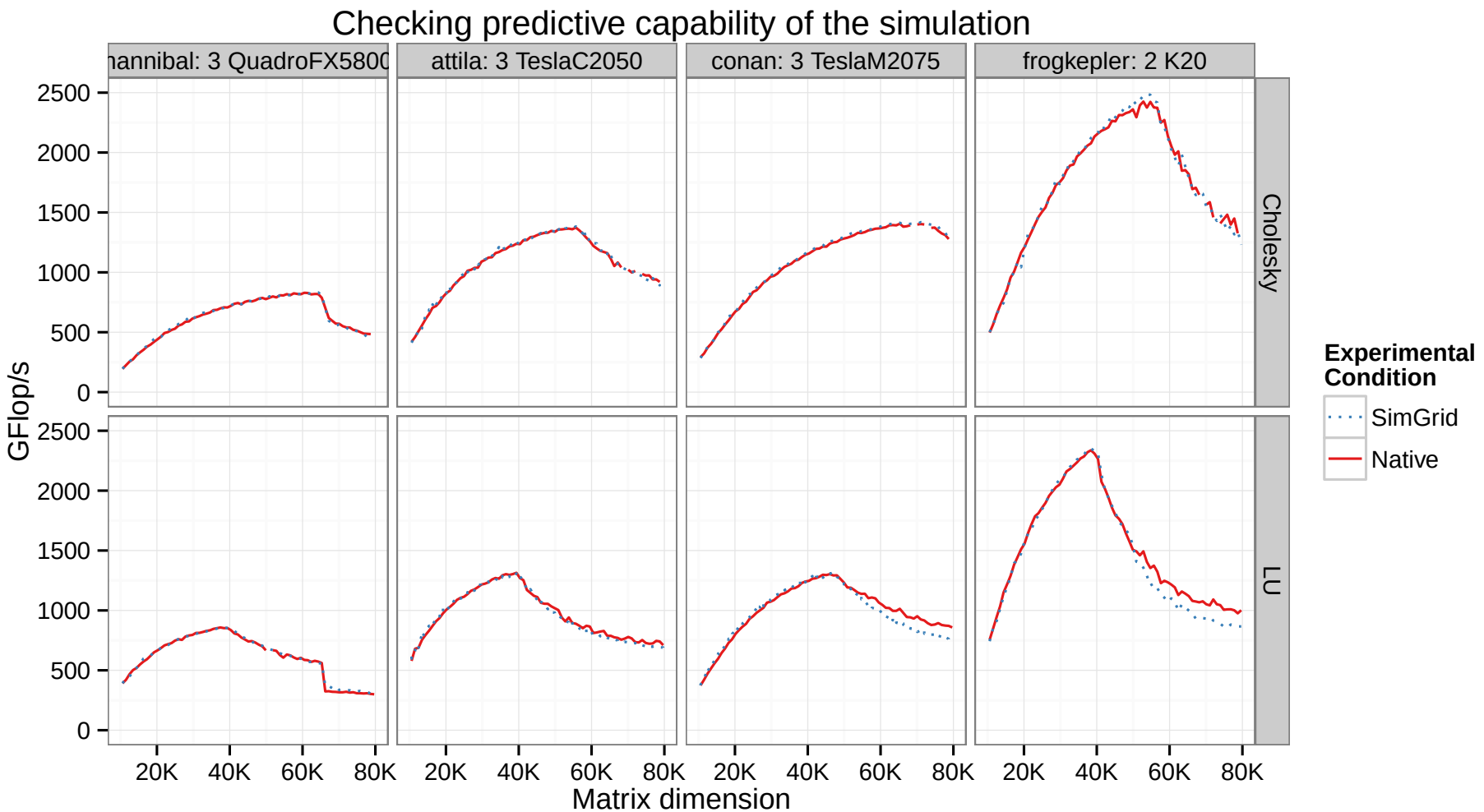
1. Modeling with threads rather than states and transitions
2. Basic and flexible contention model



Modeling Computation

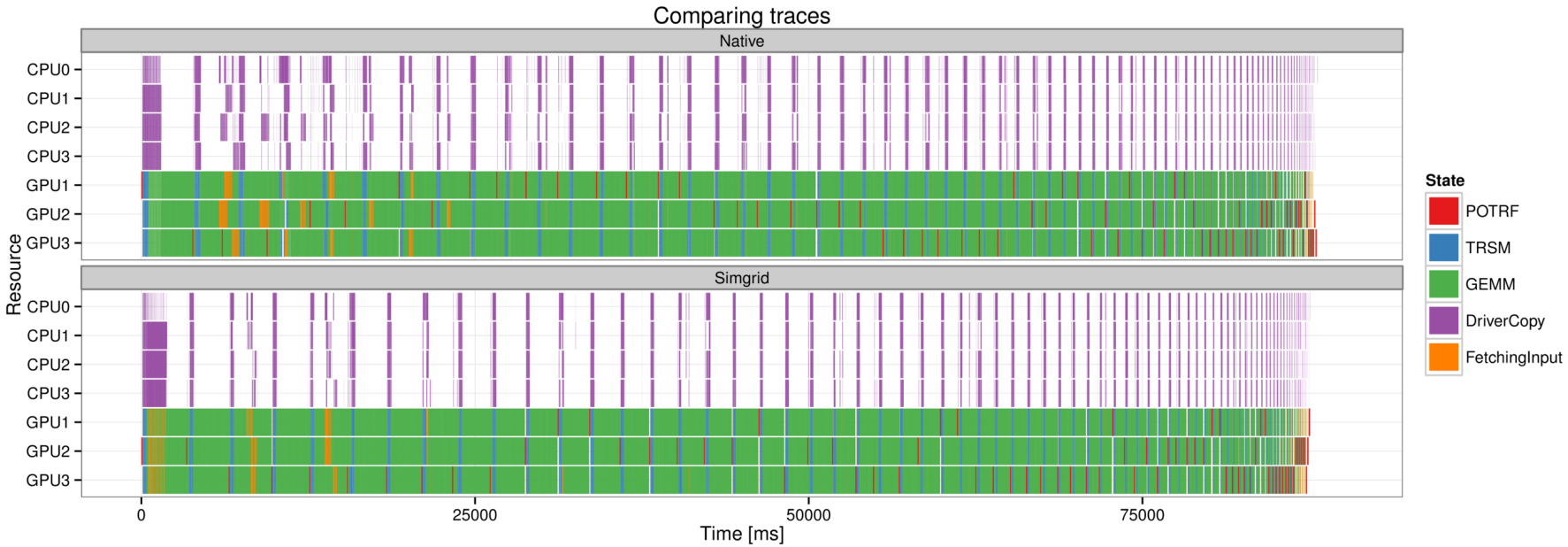
- Actual computation results are irrelevant
  - We only care about the time it takes to produce them
- Execution of each kernel is replaced by a delay accounting for its duration
- Mean duration works just fine for dense linear algebra kernels
- We also implemented histogram sampling (accounts for possible variability)

Recap


Checking predictive capability of the simulation
hannibal: 3 QuadroFX5800
attila: 3 TeslaC2050
conan: 3 TeslaM2075
frogkepler: 2 K20
Cholesky
LU
GFlop/s
0
500
1000
1500
2000
2500
20K
40K
60K
80K
Matrix dimension
Experimental Condition
SimGrid
Native

Analyzing Traces

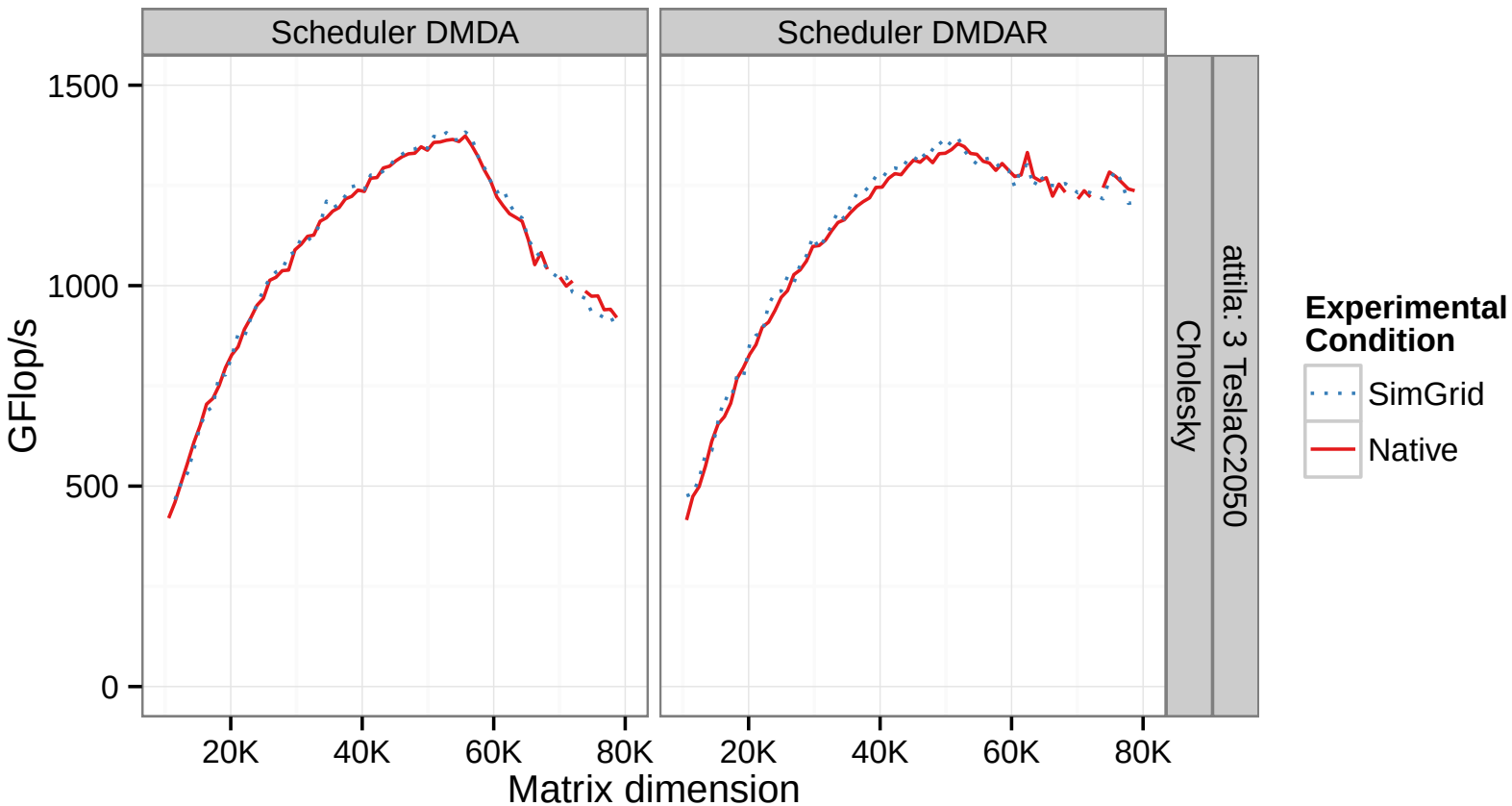
- Focusing on GFlops may hide a lot of things so we also looked at the details
- Comparing non-deterministic executions is tricky
- But global application behavior is perfectly modeled



Comparing Different Schedulers

Simulation is precise enough to explain performance issues and to faithfully compare different alternatives

- In the former cases, the DMDA does not balance memory usage between GPUs, which causes "swapping" for large matrices



Conclusion

- Works great for hybrid setups with StarPU implementation of Cholesky and LU
- Our solution allows to:
  1. Quickly and accurately evaluate the impact of various parameters or scheduling alternatives
  2. Tune and debug applications on a commodity laptop in a reproducible way
  3. Obtain reliable comparison of performance estimations that may allow to detect problems with real experiments
- Unlikely to work so well on machines with important NUMA factor
- But we have other challenges in mind for a near future

Ongoing Work: MPI

With Samuel Thibault and Augustin Degomme

- StarPU MPI can use MPI to leverage larger machines
- SimGrid MPI works well
- Could we combine these two approaches to faithfully simulate large scale hybrid architectures?

- *In Theory:* Should work fine as well
- *In Practice:* Many technical issues to address
  1. ☒ Intercepting StarPU calls by SimGrid
  2. ☒ Variable privatization
  3. ☒ SimGrid initialization
  4. ☐ Handling several `main` functions
  5. ☐ Topology modeling with private network parameters

- Expecting first experiment results in the next months

Ongoing Work: MAGMA/MORSE

With Samuel Thibault, Suraj Kumar, Emmanuel Agullo, Lionel Eyraud, ...

- MAGMA/MORSE is an extension to the MAGMA library that relies on StarPU to handle hybrid architectures
- MAGMA/MORSE/StarPU developers are starting to benefit from fast and reliable simulations and users will follow
- Initial simulation of the MORSE implementation of Cholesky was overestimating performance
  1. MORSE was actually not properly using CUDA streams
  2. Error has been corrected, improving real execution
- Now results match with a difference around 6%
- Plan to (in)validate simulation of all other MORSE applications in a near future

Ongoing Work: QR_mumps

With Abdou Guermouche, Emmanuel Agullo, Alfredo Buttari, Florent Lopez

- QR_mumps: a software package to solve sparse linear systems on hybrid computers
- qrm_starpu: implementation of QR_mumps using StarPU

- More challenging than dense linear algebra applications, because computations/communications are extremely irregular
- Already have a working prototype
- Initial investigations are promising but reveal that kernel calibration will be more challenging than in the dense case