Performance Prediction of Task-Based Runtimes

1: A. Legrand J.-F. Méhaut L. Stanisic B. Videau
2: E. Agullo A. Guermouche S. Thibault
3: A. Buttari F. Lopez

1: CNRS/Inria/University of Grenoble, France

2: University of Bordeaux/Inria, France

3: CNRS/University Paul Sabatier, Toulouse, France

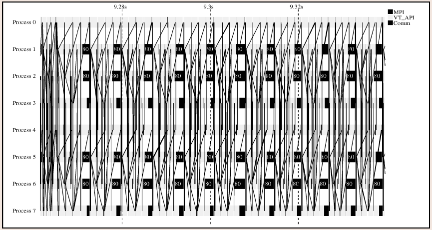
JLESC, Barcelona
June 29, 2015

Context

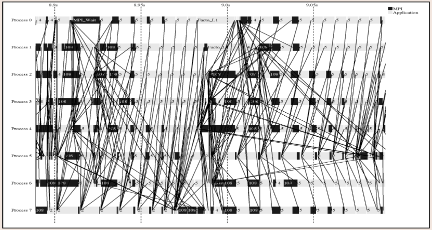
Larger and larger scale hybrid machines

$\leadsto$ Different programming approaches (e.g., in linear algebra applications)

"Rigid, hand tuned" | Task-based and Dynamic



SuperLU



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.

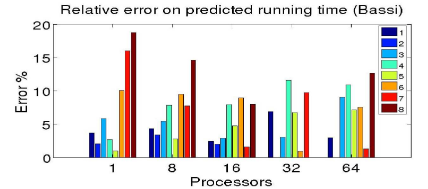
Deep need for performance prediction through simulation

- Save experimental time, baseline comparison, reproducibility, extrapolation
- Rigid (deterministic control flow) applications $\leadsto$ trace replay...
- ... but for dynamic applications, the scheduling has to be emulated

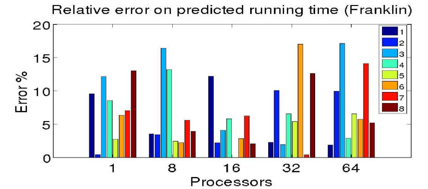
Close Related Work

Sparse linear algebra (LBNL + UCSD)

IBM Power 5



Cray TX4



Performance Modeling Tools for Parallel Sparse Linear Algebra Computations. Cicotti, Li, Baden. ParCo **2009**

- Distributed setting (SuperLU/MPI), ad hoc model of SuperLU
- Fine/Coarse grain simulation (memory, cpu, comm), linear interpolations
- Error difficult to control; Difficult evolution (no recent result AFAIK)

Dense linear algebra (UTK)

Parallel simulation of superscalar scheduling, Haugen, Kurzak, YarKhan, Dongarra. ICPP **2014**.

- Ad hoc simulator, works for OmpSs, StarPU, and QUARK
- Good results for a homogeneous machine with no communication
- Quite difficult to evolve beyond this study (IMHO)

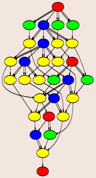
Close Related Work

Sparse linear algebra (LBNL + UCSD)

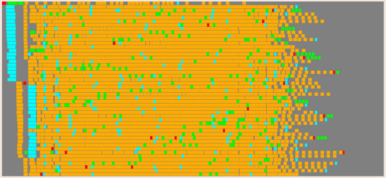
Performance Modeling Tools for Parallel Sparse Linear Algebra Computations. Cicotti, Li, Baden. ParCo **2009**

- Distributed setting (SuperLU/MPI), ad hoc model of SuperLU
- Fine/Coarse grain simulation (memory, cpu, comm), linear interpolations
- Error difficult to control; Difficult evolution (no recent result AFAIK)

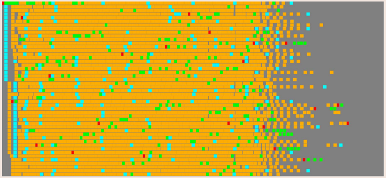
Dense linear algebra (UTK)



Real life



Simulation



QR factorization, 3960 $\times$ 3960, AMD Opteron 6180SE (4 $\times$ 12 Cores)

Parallel simulation of superscalar scheduling, Haugen, Kurzak, YarKhan, Dongarra. ICPP **2014**.

- Ad hoc simulator, works for OmpSs, StarPU, and QUARK
- Good results for a homogeneous machine with no communication
- Quite difficult to evolve beyond this study (IMHO)

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 (Inria Grenoble, Lyon, Nancy ...)

- 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

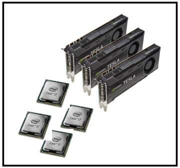
StarPU was ported on top of SimGrid by S. Thibault in 1 day:

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

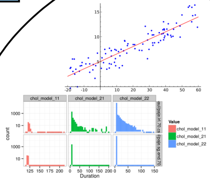
The same approach should be applicable to any task-based runtime

Envisioned Workflow: StarPU+SimGrid

Calibration



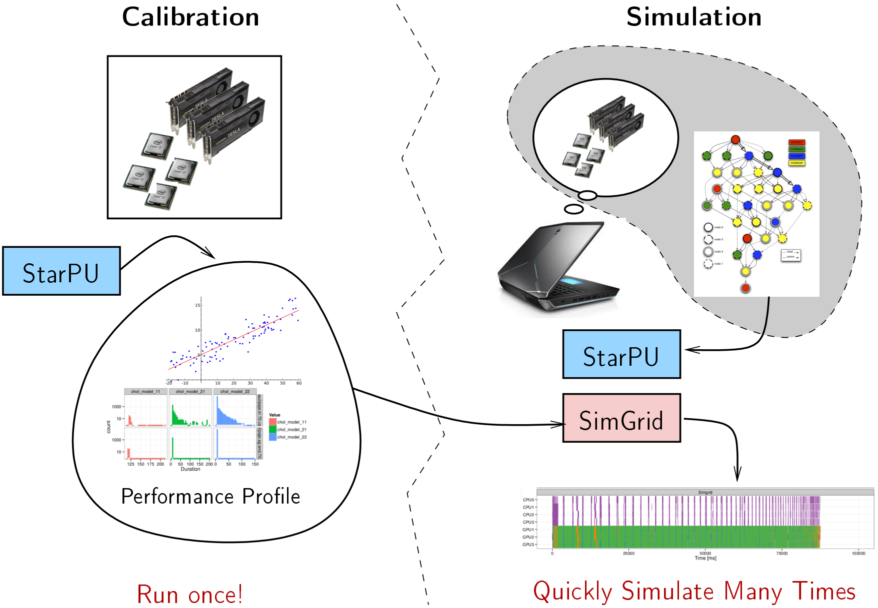
StarPU



Performance Profile

Run once!

Envisioned Workflow: StarPU+SimGrid



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

Outline

1. Evaluating Dense Linear Algebra Applications
2. Evaluating Sparse Linear Algebra Applications
3. Conclusion and Perspectives

Outline

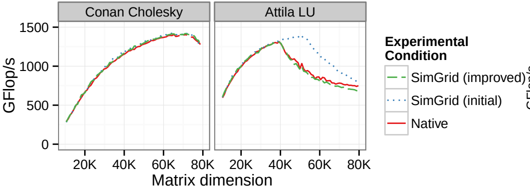
Dense Linear Algebra Applications

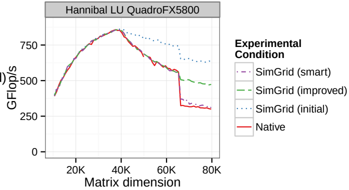
- Started with regular dense kernels and a fixed tile size
- Used two different matrix decomposition algorithms:
  1. Cholesky
  2. LU
- Used a wide diversity of machines

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

Table: Machines used for the dense linear algebra experiments.

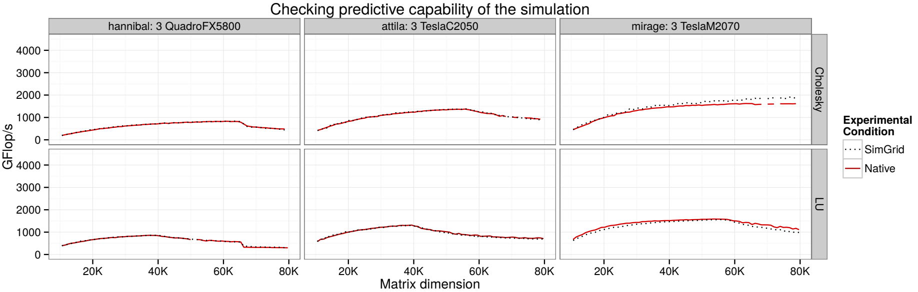
The path to reliable predictions

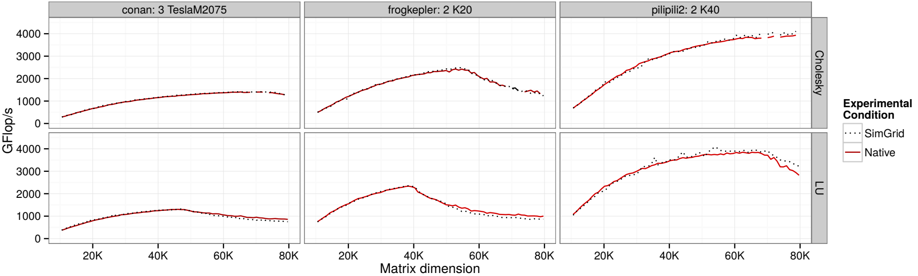




- Getting excellent results (e.g., Cholesky on Conan) sometimes do not requires much efforts
- But modeling communication heterogeneity, contention, memory operation (and even sometimes hardware/driver peculiarity) is essential
- Try to be as exhaustive as possible...

Overview of Simulation Accuracy

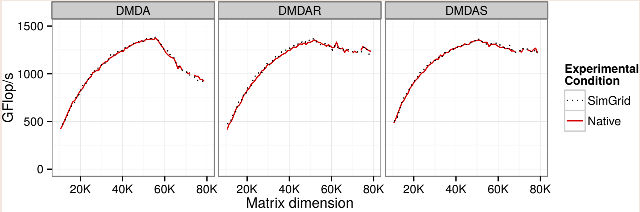




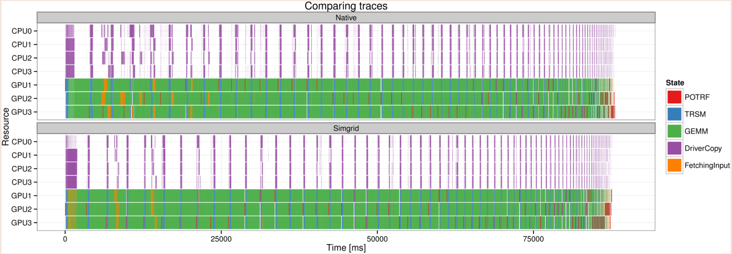
Beyond Simple Graphs

Comparing Different Schedulers

Cholesky on Attila



Investigating Details

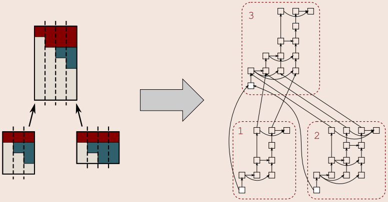


Outline

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:

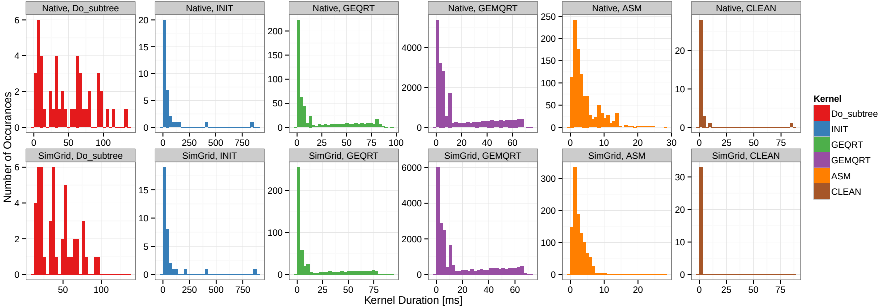
$$T_{\text{GEQRT}} = a + 2b(NB^2 \times MB) - 2c(NB^3 \times BK) + \frac{4d}{3}NB^3$$

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

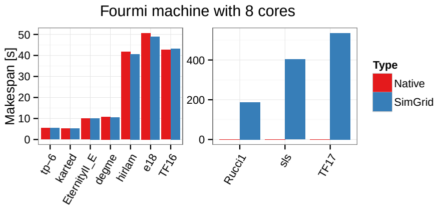
	GEQRT Duration	
NB^3	1.50×10^{-5} $(1.30 \times 10^{-5}, 1.70 \times 10^{-5})$	***
$NB^2 * MB$	5.49×10^{-7} $(5.46 \times 10^{-7}, 5.51 \times 10^{-7})$	***
$NB^3 * BK$	-5.52×10^{-7} $(-5.57 \times 10^{-7}, -5.48 \times 10^{-7})$	***
Constant	-2.49×10^{1} $(-2.83 \times 10^{1}, -2.14 \times 10^{1})$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

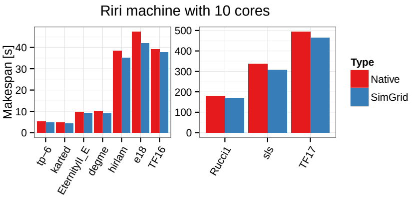
Comparing Kernel Duration Distributions

	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



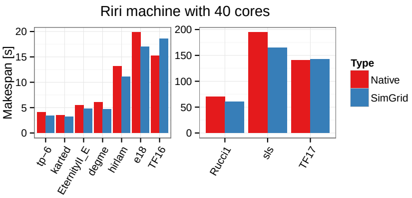
Overview of Simulation Accuracy





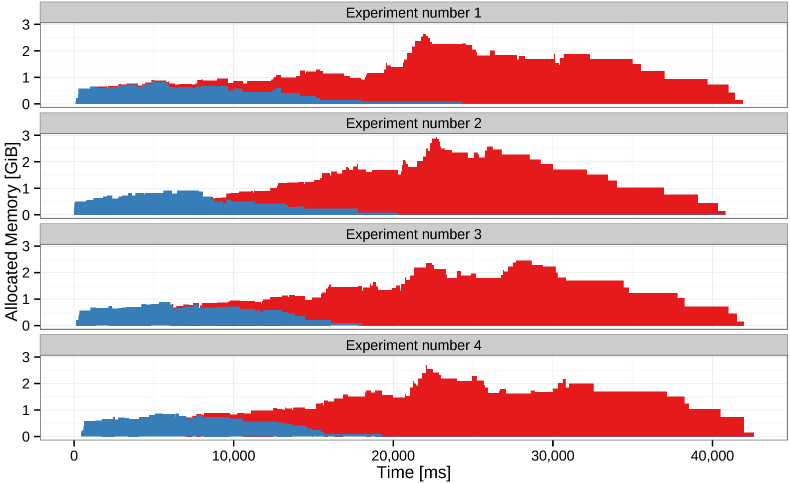
Results in a nutshell

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



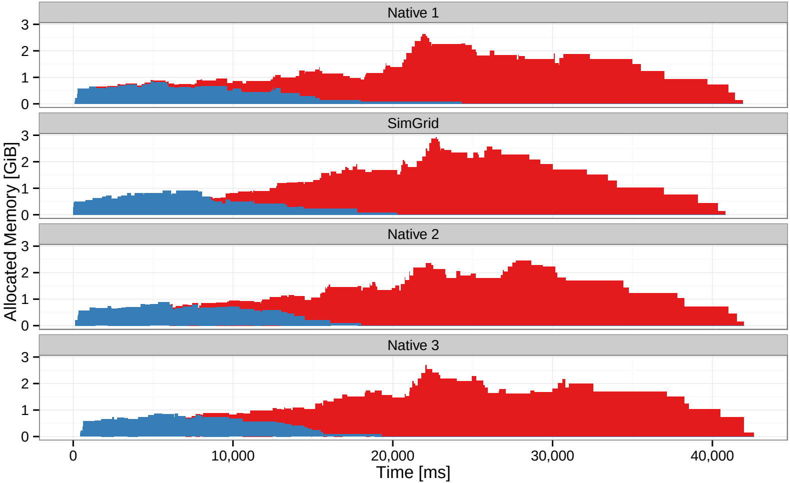
Studying Memory Consumption

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



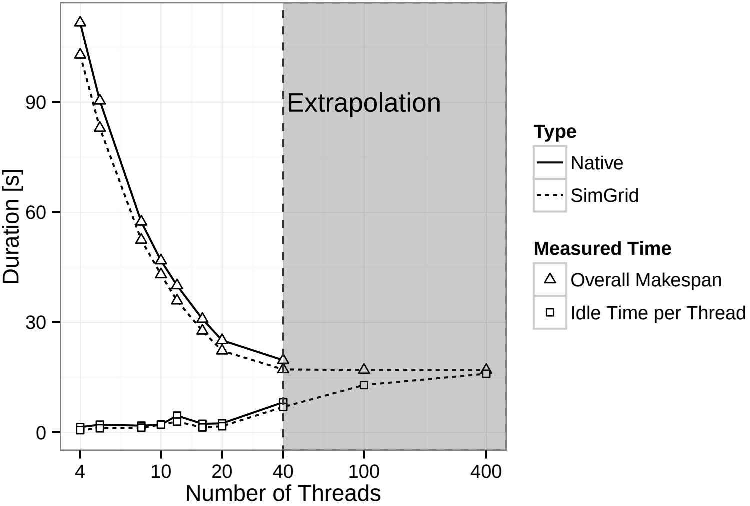
Studying Memory Consumption

- 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



Outline

Achievements

- Works great for hybrid setups with both dense and sparse linear algebra StarPU applications
- The simulator is used to investigate scheduling aspects that could not be studied with a classical approach
- Anyone can check and try to reproduce this work

  http://starpu-simgrid.gforge.inria.fr/

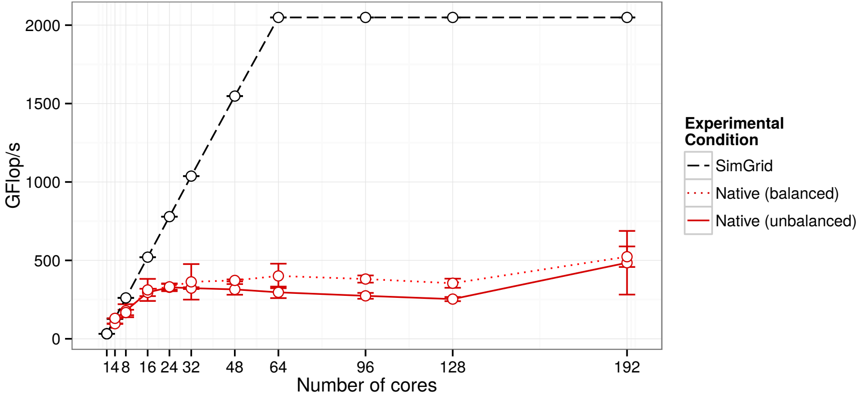
- This approach allows to:
  1. Quickly and accurately evaluate the impact of various scheduling/application parameters:

QR TF17 on riri (40 cores)	RAM	Time
RL	58.0GB	157.0s
Simulation	1.5GB	57.0s

  2. Test different scheduling alternatives
  3. Evaluate memory footprint
  4. Debug applications on a commodity laptop in a reproducible way
  5. Detect problems with real experiments using reliable comparison

There Are Situations Where We're Completely Wrong

- Some are due to bad behavior of the application/runtime in RL
- Some are due to a bad modeling of the platform (e.g., large NUMA)



Ongoing Work and Perspectives

Ongoing Work

- Simulate StarPU-MPI applications
- Simulate advanced implementations of qrm_starpu using:
  - 2D partitioning and memory aware scheduling
  - GPUs for executing tasks

Modeling and Simulation Perspectives

- Large NUMA architectures (with StarPU-MPI?)
- Kernel interferences (cache/memory contention)
- Predicting performance of next generation machines

Analysis and Visualization Perspectives

- Trace comparison
- Applicative/spatial/temporal aggregation
- Building on application models