StarPU/SimGrid: How does it Really Work?

Arnaud Legrand    Jean-François Méhaut    Luka Stanisic    Brice Videau
Emmanuel Agullo    Abdou Guermouche    Samuel Thibault
Alfredo Buttari    Florent Lopez

CNRS/Inria/University of Grenoble, France
University of Bordeaux/Inria, France
CNRS/University Paul Sabatier, Toulouse, France

SOLHAR meeting, Lyon
June 12, 2015
Developing Applications and Runtime for Hybrid Machines

Application and Runtime Developers Wishes

1. Controled and replicable results
2. Access to many different platforms
3. Shorter time to conduct large experimentation campaigns

Possible Answer: Simulation

1. Reproducible executions (performance, bugs)
2. Predictions on unavailable architectures
3. Cheaper than running on real supercomputers ~ richer experimental design possible
StarPU and SimGrid

StarPU

- Dynamic runtime for hybrid architectures
- Opportunistic scheduling of a task graph
- Guided by resource performance models

SimGrid

- Simulation framework for distributed systems
- Performance oriented and scalable
- Modeling with threads rather than states and transitions \(\leadsto\) ability to simulate dynamic applications
- Portable, open source and easily extendable

StarPU was ported on top of SimGrid by Samuel
Envisioned Workflow: StarPU + SimGrid

Calibration

StarPU

Performance Profile

Run once!
Envisioned Workflow: StarPU+SimGrid

Calibration

Performance Profile

Run once!

Simulation

Quickly Simulate Many Times

StarPU

SimGrid
Outline

1. Coupling StarPU and SimGrid
2. Evaluating Dense Linear Algebra Applications
3. Evaluating Sparse Linear Algebra Applications
4. Conclusion and Perspectives
Emulation: executing real applications in a synthetic environment
Simulation: representing process as sequence of events separated by delays

- 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
Modeling Runtime

Increasing Simulation Time

1. Process synchronizations
2. Memory allocations of CPU or GPU
3. Submission of data transfer requests

Example from `malloc.c` in StarPU

```c
... ndef STARPU_SIMGRID
    MSG_process_sleep((float) dim * 0.000650 / 1048576.);
#else
    if (_starpu_can_submit_cuda_task()) {
        cudaError_t cures;
        cures = cudaHostAlloc(A, dim, cudaHostAllocPortable);
        ...
Components of hybrid platforms have differing characteristics
Correctly modeling their communication is of primary importance

Flow-Based Contention Model (TCP/IP)

Basic and flexible contention model

(a) Crude modeling
(b) More elaborated modeling
Modeling Computation

Actual Computation Results Irrelevant
- Workflow is not function of the results
- Computation results can be discarded
- Only time to compute is important

Execution of Kernels Replaced by Delays
- Mean duration works just fine for dense linear algebra kernels
- Additionally implemented histogram sampling (accounts for possible variability)
Outline

1. Coupling StarPU and SimGrid
2. Evaluating Dense Linear Algebra Applications
3. Evaluating Sparse Linear Algebra Applications
4. Conclusion and Perspectives
Dense Linear Algebra Applications

- Started with more regular dense kernels
- Used two different matrix decomposition algorithms:
  1. Cholesky
  2. LU
- Efficiently implemented over StarPU
- Extensively used by StarPU developers and users to:
  1. Develop StarPU
  2. Test StarPU performance
  3. Test scheduling policies
  4. Use as templates for new algorithm implementations
### Experimental Setup

<table>
<thead>
<tr>
<th>Name</th>
<th>Processor</th>
<th>#Cores</th>
<th>Memory</th>
<th>GPUs</th>
</tr>
</thead>
<tbody>
<tr>
<td>hannibal</td>
<td>X5550</td>
<td>$2 \times 4$</td>
<td>$2 \times 24\text{GB}$</td>
<td>$3 \times \text{QuadroFX5800}$</td>
</tr>
<tr>
<td>attila</td>
<td>X5650</td>
<td>$2 \times 6$</td>
<td>$2 \times 24\text{GB}$</td>
<td>$3 \times \text{TeslaC2050}$</td>
</tr>
<tr>
<td>mirage</td>
<td>X5650</td>
<td>$2 \times 6$</td>
<td>$2 \times 18\text{GB}$</td>
<td>$3 \times \text{TeslaM2070}$</td>
</tr>
<tr>
<td>conan</td>
<td>E5-2650</td>
<td>$2 \times 8$</td>
<td>$2 \times 32\text{GB}$</td>
<td>$3 \times \text{TeslaM2075}$</td>
</tr>
<tr>
<td>frogkepler</td>
<td>E5-2670</td>
<td>$2 \times 8$</td>
<td>$2 \times 16\text{GB}$</td>
<td>$2 \times \text{K20}$</td>
</tr>
<tr>
<td>pilipili2</td>
<td>E5-2630</td>
<td>$2 \times 6$</td>
<td>$2 \times 32\text{GB}$</td>
<td>$2 \times \text{K40}$</td>
</tr>
<tr>
<td>idgraf</td>
<td>X5650</td>
<td>$2 \times 6$</td>
<td>$2 \times 36\text{GB}$</td>
<td>$8 \times \text{TeslaC2050}$</td>
</tr>
<tr>
<td>idchire</td>
<td>E5-4640</td>
<td>$24 \times 8$</td>
<td>$24 \times 31\text{GB}$</td>
<td>/</td>
</tr>
</tbody>
</table>

**Table**: Machines used for the dense linear algebra experiments.
Most results are satisfactory
Some need improved models
For Attila and Hannibal, SimGrid is too optimistic => why?
Improving GPU Memory Models

- Taking GPU size into account (attila and hannibal have 3GB)
- Using improved network model
- Performance drop for large matrices on hannibal => why?
Improving QuadroFX5800 Transfer Model

- Transfer time of 3.6 MB using cudaMemcpy2D differs
- Problem with older GPUs when copying with large strides
Overview of Simulation Accuracy

- 4 different platforms
- 2 different algorithms
- Memory footprint ranges from 3.6 MB to 27.8 GB

Checking predictive capability of the simulation

<table>
<thead>
<tr>
<th>Platform</th>
<th>GPUs</th>
</tr>
</thead>
<tbody>
<tr>
<td>hannibal</td>
<td>3 QuadroFX5800</td>
</tr>
<tr>
<td>attila</td>
<td>3 TeslaC2050</td>
</tr>
<tr>
<td>conan</td>
<td>3 TeslaM2075</td>
</tr>
<tr>
<td>frogkepler</td>
<td>2 K20</td>
</tr>
</tbody>
</table>

Matrix dimension vs. GFlop/s

- Cholesky
- LU
Comparing Different Schedulers

- Differences between schedulers performances are faithfully predicted by SimGrid
- DMDAR: locality aware scheduler $\Rightarrow$ less transfer

![Graph comparing schedulers' performances](image-url)
- GFlops are a limited metric
- Verifying that simulation traces are representative
Outline

1. Coupling StarPU and SimGrid
2. Evaluating Dense Linear Algebra Applications
3. Evaluating Sparse Linear Algebra Applications
4. Conclusion and Perspectives
Simulating Sparse Solvers

In collaboration with Emmanuel Agullo, Alfredo Buttari, Abdou Guermouche and Florent Lopez

<table>
<thead>
<tr>
<th>qrm_starpu</th>
</tr>
</thead>
<tbody>
<tr>
<td>• QR factorization on top of StarPU runtime</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Porting qrm_starpu on top of SimGrid</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Changing main for the subroutine</td>
</tr>
<tr>
<td>• Changing compilation process</td>
</tr>
<tr>
<td>• Advanced kernel modeling</td>
</tr>
</tbody>
</table>
Dense kernels (POTRF, GEMM, ...) during single experiment are always executed with the same block size \(\leadsto\) duration is very stable

Sparse kernel durations depend on their input parameters \(\leadsto\) more variability

Cannot model sparse kernels with simple mean values
Example for Modeling Kernels: GEQRT

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

- It is a linear model, depends only on few matrix parameters
- We can do a linear regression based on adhoc calibration

<table>
<thead>
<tr>
<th>GEQRT Duration</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>(NB^3)</td>
<td>1.50 \times 10^{-5} (1.30 \times 10^{-5}, 1.70 \times 10^{-5}) ***</td>
</tr>
<tr>
<td>(NB^2 \times MB)</td>
<td>5.49 \times 10^{-7} (5.46 \times 10^{-7}, 5.51 \times 10^{-7}) ***</td>
</tr>
<tr>
<td>(NB^3 \times BK)</td>
<td>-5.52 \times 10^{-7} (-5.57 \times 10^{-7}, -5.48 \times 10^{-7}) ***</td>
</tr>
<tr>
<td>Constant</td>
<td>-2.49 \times 10^1 (-2.83 \times 10^1, -2.14 \times 10^1) ***</td>
</tr>
</tbody>
</table>

Observations 493
\(R^2\) 0.999

Note: *p<0.1; **p<0.05; ***p<0.01
### Matching Kernel Duration Distributions

<table>
<thead>
<tr>
<th></th>
<th>Do_subtree</th>
<th>INIT</th>
<th>GEQRT</th>
<th>GEMQRT</th>
<th>ASM</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. #Flops</td>
<td></td>
<td></td>
<td>NB</td>
<td>NB</td>
<td>#Coeff</td>
</tr>
<tr>
<td>2. #Nodes</td>
<td></td>
<td></td>
<td>MB</td>
<td>MB</td>
<td>/</td>
</tr>
<tr>
<td>3. #Assemble</td>
<td></td>
<td></td>
<td>BK</td>
<td>BK</td>
<td>/</td>
</tr>
<tr>
<td>$R^2$</td>
<td>0.99</td>
<td>0.99</td>
<td>0.99</td>
<td>0.99</td>
<td>0.86</td>
</tr>
</tbody>
</table>

#### Kernel Duration Distributions

![Graphs showing kernel duration distributions for different kernels and implementations](image)

#### Kernel Details
- **Do_subtree**
- **INIT**
- **GEQRT**
- **GEMQRT**
- **ASM**
- **CLEAN**
Overview of Simulation Accuracy

- Most results are satisfactory
- With bigger and architecturally more complex machines error is increasing
Minimizing memory footprint is very important for qrm_starpu.
Scheduling is dynamic so even consecutive Native experiments have different output.
Minimizing memory footprint is very important for qrm_starpu
Scheduling is dynamic so even consecutive Native experiments have different output
Extrapolating to Larger Machines

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

1. Coupling StarPU and SimGrid
2. Evaluating Dense Linear Algebra Applications
3. Evaluating Sparse Linear Algebra Applications
4. Conclusion and Perspectives
Works great for hybrid setups with both dense and sparse linear algebra StarPU applications

Our solution allows to:

1. Quickly and accurately evaluate the impact of various parameters
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
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
- Kernel interferences (cache contention)
- Predicting performance of next generation machines

Analysis and Visualization Perspectives

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

Open Science

- Anyone can check and try to reproduce this work

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