WP4: StarPU+SimGrid

Luka Stanisic    Samuel Thibault    Arnaud Legrand

CNRS/Inria/University of Grenoble, France
University of Bordeaux/Inria, France

SONGS meeting, Nancy
January 12, 2015
Envisioned Workflow: StarPU + SimGrid

Calibration

StarPU

Performance Profile

Run once!
Envisioned Workflow: StarPU+SimGrid

Calibration

StarPU

Performance Profile

Run once!

Simulation

SimGrid

StarPU

Quickly Simulate Many Times
Implementation Principles

**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
Overview of Simulation Accuracy

- 7 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>
<th>20K</th>
<th>40K</th>
<th>60K</th>
<th>80K</th>
<th>20K</th>
<th>40K</th>
<th>60K</th>
<th>80K</th>
<th>20K</th>
<th>40K</th>
<th>60K</th>
<th>80K</th>
</tr>
</thead>
<tbody>
<tr>
<td>hannibal</td>
<td>3 QuadroFX5800</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>attila</td>
<td>3 TeslaC2050</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>conan</td>
<td>3 TeslaM2075</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>frogkepler</td>
<td>2 K20</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Matrix dimension: 20K, 40K, 60K, 80K

Experimental Condition
- SimGrid
- Native
Outline

1. Extending Research for Dense Linear Algebra Kernels
2. Investigating Sparse Linear Algebra Kernels
3. Conclusion
• GPUs are much more powerful

• Simulation accuracy for CPU+GPU execution is slightly worse, but still shows clearly the trends

Figure: Illustrating simulation accuracy for Cholesky application using different resources of the Mirage machine.
Real Hybrid: Similar Traces

Comparing traces

Native

CPU0
CPU1
CPU2
CPU3
CPU4
CPU5
CPU6
CPU7
CPU8
CPU9
CPU10
CPU11
GPU1
GPU2
GPU3

SimGrid

CPU0
CPU1
CPU2
CPU3
CPU4
CPU5
CPU6
CPU7
CPU8
CPU9
CPU10
CPU11
GPU1
GPU2
GPU3

Time [ms]

0 20000 40000 60000 80000

State
- POTRF
- TRSM
- GEMM
- DriverCopy
Limits of Our Approach: NUMA Architectures

- CPUs use shared memory so no explicit data transfers
- Time to access data depends on the NUMA node
- Effective memory bandwidth depends on efficient utilization
- Very sensitive to suboptimal block size and memory strides

Figure: Illustrating the impact of deployment when using 8 cores on two NUMA nodes on the Mirage machine.
Bad Predictions for 192 Cores Machine

Figure: Simulation predictions of Cholesky application with a $32,000 \times 32,000$ matrix (block size $320 \times 320$) on large NUMA Idchire machine are precise for a small number of cores, but scale badly. The reason is that the memory is shared, while models are not taking into account various NUMA effects.
Outline

1. Extending Research for Dense Linear Algebra Kernels
2. Investigating Sparse Linear Algebra Kernels
3. Conclusion
QR_mumps + StarPU + SimGrid (1/2)

- QR_mumps: a software package for the solution of sparse, linear systems on hybrid computers
- qrm_starpu: implementation of QR_mumps using StarPU
- Developers:
  1. Alfredo Buttari, Florent Lopez - Toulouse
  2. Abdou Guermouche, Emmanuel Agullo - Bordeaux
- More challenging than dense linear algebra applications, because computations/communications are irregular
- For now working with CPU implementation, GPU coming soon

First simulation results: Makespans are matching!
- Shocking since each micro-kernel was represented as a single mean value when simulating
- Micro-kernel execution time greatly varies depending on the block size and characteristics
QR_mumps + StarPU + SimGrid (2/2)

**Explanation**
- QR_mumps is well implemented -> very little idle time
- All CPUs are constantly working -> injecting mean timings will finally give good overall execution time
- However traces are not matching at all

**Ultimate goal**
- Find precise models for each micro-kernel and use them in simulation to compute timings

**Challenges**
- Determine crucial parameters
- Not trivial retrieve parameters from execution
Illustration with tp-6 mtx on 8 cores

- Native and SimGrid makespans matching perfectly
- Zooming on execution times of different kernels from paje traces
1. Extending Research for Dense Linear Algebra Kernels

2. Investigating Sparse Linear Algebra Kernels

3. Conclusion
Conclusion

- Works great for hybrid setups with StarPU applications
- Our solution allows to:
  1. Quickly and accurately evaluate the impact of various parameters
  2. Test different scheduling alternatives
  3. Debug applications on a commodity laptop in a reproducible way
  4. Detect problems with real experiments using reliable comparison
- Some researchers in Bordeaux are already extensively using it
- Stable in terms of both performance and bugs (except MAGMA/MORSE)
- Work in progress: Coupling StarPU-MPI with SimGrid
- Published at EuroPar14, waiting for extended CCPE journal version
Real Hybrid: Paje Traces without MKL

Comparing traces

State
- POTRF
- TRSM
- GEMM
- DriverCopy

Resource

CPU0
CPU1
CPU2
CPU3
CPU4
CPU5
CPU6
CPU7
CPU8
CPU9
CPU10
CPU11
GPU1
GPU2
GPU3

Time [ms]

0
20000
GEQRT Depending on Matrix Size
GEQRT Depending on Important Parameters

Start

Duration

X1

X2

BK

SYM
Call:
```r
lm(formula = Duration ~ X1 + X2 + BK + SYM + X1:X2 + X1:SYM, 
data = dftest)
```

Coefficients:

| Term       | Estimate | Std. Error | t value | Pr(>|t|) |
|------------|----------|------------|---------|----------|
| (Intercept)| 4.905e+00| 3.561e-01  | 13.773  | <2e-16 ***|
| X1         | 4.051e-03| 1.633e-04  | 24.811  | <2e-16 ***|
| X2         | -3.036e-02| 3.038e-03 | -9.993  | <2e-16 ***|
| BK         | -8.426e-01| 4.162e-03 | -202.431| <2e-16 ***|
| SYM        | 3.463e+01| 2.232e-01  | 155.160 | <2e-16 ***|
| X1:X2      | 2.662e-05| 1.350e-06  | 19.711  | <2e-16 ***|
| X1:SYM     | -1.259e-04| 6.059e-05 | -2.078  | 0.0382 * |

Residual standard error: 0.6118 on 460 degrees of freedom
Multiple R-squared:  0.998, Adjusted R-squared:  0.998
F-statistic: 3.816e+04 on 6 and 460 DF,  p-value: < 2.2e-16
• Actual computation results irrelevant - care only about the time it takes to get them
• Execution of each kernel replaced by a virtual delay accounting for its duration
• Mean duration work fine (for now!).
Modeling Computation (2)

- Actual computation results irrelevant - care only about the time it takes to get them
- Execution of each kernel replaced by a virtual delay accounting for its duration
- Mean duration work fine (for now!).

<table>
<thead>
<tr>
<th>Problems:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Obtaining only one possible execution</td>
</tr>
<tr>
<td>Scheduling sensibility</td>
</tr>
<tr>
<td>Possible bias (e.g. found a deadlock StarPU)</td>
</tr>
</tbody>
</table>
Actual computation results irrelevant - care only about the time it takes to get them
Execution of each kernel replaced by a virtual delay accounting for its duration
Mean duration work fine (for now!).

Problems:
- Obtaining only one possible execution
- Scheduling sensibility
- Possible bias (e.g. found a deadlock StarPU)

Solution:
- Introducing some variability with histograms
- Benchmarking timings of computational kernels during the real execution and later approximating their distribution
Histogram Problem
Histogram Problem

- Default R function for histograms is not made for this
- It uses uniform bin-widths, which are very inefficient at representing details of distributions
- Information loss due to outliers, empty regions, magnitude difference
Histogram Problem

- Default R function for histograms is not made for this
- It uses uniform bin-widths, which are very inefficient at representing details of distributions
- Information loss due to outliers, empty regions, magnitude difference
Histogram Problem

- Default R function for histograms is not made for this
- It uses uniform bin-widths, which are very inefficient at representing details of distributions
- Information loss due to outliers, empty regions, magnitude difference
- Started using *dhist*
- For optimal results need careful parameter tuning for each distribution