Towards Performance-Portable, Scalable, and Convenient Linear Algebra

Philippe Tillet\textsuperscript{1,3}, Karl Rupp\textsuperscript{2}, Siegfried Selberherr\textsuperscript{1}, Chin-Teng Lin\textsuperscript{3}

\textsuperscript{1}Institut für Mikroelektronik, TU Wien, Austria
\textsuperscript{2}MCS Division, Argonne National Laboratory, USA
\textsuperscript{3}National Chiao Tung University, Taiwan

HotPar’13, June 24th, 2013
GPUs: Library Aspects

Challenge: The architectural barrier

Challenge: The memory barrier

Challenge: The usability barrier
OpenCL - A unified programming model for parallel devices

Includes:
- a low-level API
- a low-level programming language

Compiled Just-In-Time

Write once, run everywhere

Currently available for CPUs, GPUs, MICs...
Soon available on FPGAs, DSPs...
The architectural barrier
The architecture strikes back

The good old matrix-matrix multiplication problem

We obtain same performance as CuBLAS 5.0
What if we use this kernel on AMD Hardware?
The architectural barrier
The architecture strikes back

The good old matrix-matrix multiplication problem

We get 60% of the clAmdBlas performance!
What if we use this kernel on a CPU?
The good old matrix-matrix multiplication problem

We get 20% of the MKL performance!
Code portability does not imply performance portability
The architectural barrier
Breaking the barrier

The manual approach

Learn about the architecture
Write, debug, profile, optimize your code
Repeat the process whenever a new architecture is released

The automatic approach

Build a compute kernel generator
Build parameterized compute kernels for your algorithms
Run an automatic tuning procedure
The architectural barrier
The auto-tuning procedure

Potentially huge parameter space

30k × 4 layouts × 4 transpositions = 480k kernels

Big area of future work

The parameter space is still architecture-dependent
The brute-force approach executes in 10 × 4 × 4 hours...
The good old matrix-matrix multiplication problem - revisited

What about single precision?
The architectural barrier

Breaking the barrier

The good old matrix-matrix multiplication problem - revisited

What about less compute-intensive tasks?
The architectural barrier
Breaking the barrier

The Matrix-Vector multiplication problem

OpenCL code *can* be performance-portable
How to run

BLAS Level 1 operations

```cpp
using namespace viennacl;
vector<double> x, y;
scalar<double> beta;
/* Fill x, y here */
custom_operation op;
op.add(beta = inner_prod(x, 2*x + y));
op.execute();
```

Multi-Matrix multiplication

```cpp
using namespace viennacl;
add_all_available_devices(CL_DEVICE_TYPE_GPU);
multi_matrix<float> C, A, B;
/* Fill A, B here */
C = prod(A + B, A - B);
finish();
```
What you get
BLAS Level 1 operations

![Graph showing performance comparison between different hardware and software options.](image-url)
What you get
Distributed SGEMM

![Graph showing performance metrics for different configurations of ViennaCL on different hardware. The x-axis represents Matrix Rows/Columns in thousands, and the y-axis represents GFLOP/s and Memory Usage in GB. The graph compares various configurations including ViennaCL on GTX 470, Tesla C2050, GTX 470 + Tesla C2050, and Ideal configurations.]
Summary

Auto-Tuning Framework

- Adapts the underlying hardware
- The whole BLAS Standard is supported
- At least 75% of vendor-tuned libraries performance
- Supports multiple GPUs

Future Work

- Improve the auto-tuning procedure
- Fully transparent use

Open-Source

- Will be released in ViennaCL 1.5.0
  http://viennacl.sourceforge.net
Global Structure

Scheduler

Task 1
Task 2
Vector Addition

Task 3
Task 4
Matrix Product

RAM

Device Dv1,1
Vendor 1
Task 1
Compute Kernel
Generator

Device Dv1,2 Device Dv2,1

Compute Kernel Generator

Vendor 1
Device Dv1,1

Vendor 2
Device Dv2,1

Device Dv1,2
Generator Scheme

Device 1

Auto-Tuning Environment

Dynamic Optimizations

Dynamic Optimizations

Template Engine

Static Optimizations

Compute Kernels Segmentation

Device N

Auto-Tuning Environment

Dynamic Optimizations

Dynamic Optimizations

Dynamic Optimizations