Comparison of OpenCL performance on different platforms using VexCL and Blaze

Hammad Mazhar
Dan Negrut

January 28, 2015
Abstract

This technical report provides performance numbers for several benchmark problems running on several different hardware platforms. The goal of this report is twofold. First, it helps us better understand how the performance of OpenCL changes on different platforms. Second, it provides a OpenCL-OpenMP comparison for a sparse matrix-vector multiplication operation. The VexCL library will be used for the OpenCL portion of this comparison and the Blaze C++ library will be used for the OpenMP portion.
Contents

1 Blaze 3

2 VexCL 3

3 Hardware Platforms 3
   3.1 AMD Kaveri 3
   3.2 Intel Haswell 4
   3.3 Intel Haswell-E 4
   3.4 Intel Xeon 5
   3.5 AMD Opteron 5

4 Benchmark Results 6

5 Chrono SpMV Benchmark 7
1  Blaze

Blaze [1] is an open source headers-only library for performing linear algebra operations using dense and sparse data structures. Blaze was designed so that mathematical expressions can be written intuitively with the library transparently handling type conversion and optimization. By default, Blaze uses OpenMP for parallelism but it can be configured to use C++11 threads, Boost threads and also execute serially. Additionally bindings for generic BLAS libraries, such as ATLAS, which will be transparently used for certain linear algebra operations such as Matrix-Matrix multiplication.

In this context Blaze was used to compare the performance of OpenCL to OpenMP on platforms that supported it.

2  VexCL

VexCL [2] is an open source headers-only expression template library for both OpenCL and CUDA. Similar to Thrust [3], its purpose is to reduce boilerplate code required to develop applications for GPUs and other accelerators. VexCL provides many different functions that deal with reduction, linear algebra, sorting etc. In terms of this technical report the synthetic benchmark results are provided by the VexCL benchmark example [4]. For the real world examples VexCL’s SpMV function is used.

For all tests the latest version of VexCL was used from the GitHub repository [5].

3  Hardware Platforms

Thirteen different hardware setups were used for benchmarking, this section will describe each setup.

3.1  AMD Kaveri

This CPU is based on AMDs new Kaveri architecture, the die features 4 x86-64 cores based on AMDs Steamroller architecture and an 8 core Radeon R7 class GPU. Specifically, the performance in the 7850K matches that of the Radeon HD 7750. The GPU does not have dedicated memory and instead relies on the system memory. The upside of this is that the total memory available to the GPU is equal to the system ram, the downside is that generally system memory is much slower than ordinary on-board GPU memory.

<table>
<thead>
<tr>
<th>Model</th>
<th>AMD A10-7850K</th>
</tr>
</thead>
<tbody>
<tr>
<td>Architecture</td>
<td>Steamroller</td>
</tr>
<tr>
<td>Clock (Turbo)</td>
<td>3.7 GHz (4.0 GHz)</td>
</tr>
<tr>
<td>Cores</td>
<td>4</td>
</tr>
<tr>
<td>Threads</td>
<td>4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>L1 Cache</th>
</tr>
</thead>
<tbody>
<tr>
<td>2x96 KB 3-way Instruction</td>
</tr>
<tr>
<td>4x16 KB 4-way Data</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>L2 Cache</th>
</tr>
</thead>
<tbody>
<tr>
<td>2x2 MB 16-way</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>L3 Cache</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
</tr>
</tbody>
</table>
Memory 16GB
Memory Interface Dual Channel DDR3
OS Arch Linux

Compiler GCC 4.9.2
Compiler Flags -O3
OpenCL AMD APP

Accelerator

Type AMD Radeon R7 series
Compute Units 8

Reference: [6, 7]

3.2 Intel Haswell

Model i7-4770K
Architecture Haswell
Clock (Turbo) 3.5 GHz (3.9 GHz)
Cores 4
Threads 8
L1 Cache
   4x32 KB 8-way Instruction
   4x32 KB 8-way Data

L2 Cache 4x256 KB 8-way
L3 Cache 8 MB 16-way
Memory 32GB
Memory Interface Dual Channel DDR3
OS Arch Linux
Compiler GCC 4.9.2
Compiler Flags -O3
OpenCL Intel(R) OpenCL

Accelerator 1

Model NVIDIA GTX 680
Architecture GK104 - Kepler
Compute Units 8

Cores 1536
Clock (Boost) 1006 MHz (1058 MHz)
Memory 2GB 256-bit GDDR5

Accelerator 2

Model NVIDIA K20c
Architecture GK110 - Kepler
Compute Units 13

Cores 2496
Clock 706 MHz
Memory 5GB 320-bit GDDR5

Reference: [8–11]

3.3 Intel Haswell-E

Model i7-5960X
Architecture Haswell-E
CPU Clock (Turbo) 3.0 GHz (3.5 GHz)
CPU Cores 8
CPU Threads 16

L1 Cache
   8x32 KB Instruction
   8x32 KB Data
L2 Cache 8x256 KB
L3 Cache 20 MB
### Accelerator

**Model** NVIDIA GTX 770  
**Architecture** GK104 - Kepler  
**Compute Units** 8

Reference: [12, 13]

#### 3.4 Intel Xeon

**Model** E5-2690 V2  
**Architecture** Ivy Bridge-EP  
**Sockets** 2  
**CPU Clock (Turbo)** 3.0 GHz (3.6 GHz)  
**CPU Cores** 10  
**CPU Threads** 20  
**L1 Cache**
   - 10x32 KB 8-way Instruction  
   - 10x32 KB 8-way Data

**L2 Cache** 10x256 KB 8-way  
**L3 Cache** 25 MB 20-way  
**Memory** 64GB  
**Memory Interface** Quad Channel DDR3  
**OS** Arch Linux  
**Compiler** GCC 4.9.2  
**Compiler Flags** -O3  
**OpenCL** Intel(R) OpenCL

### Accelerator 1,2,3

**Model** NVIDIA K20x  
**Architecture** GK110 - Kepler  
**Compute Units** 15

**Cores** 2688  
**Clock** 732 MHz  
**Memory** 6GB 384-bit GDDR5

### Accelerator 4

**Model** Intel Xeon Phi 5110P  
**Architecture** Knights Corner  
**Cores** 60  
**Threads** 240  
**L1 Cache**
   - 60x32 KB 8-way Instruction  
   - 60x32 KB 8-way Data  

**L2 Cache** 60x512 KB 8-way  
**Clock** 1053 MHz  
**Memory** 8GB GDDR5

Reference: [14–16]

### 3.5 AMD Opteron
Model 6274
Architecture Bulldozer
Sockets 4
Clock (Turbo) 2.2 GHz (3.1 GHz)
Cores 16
Threads 16
L1 Cache
8x64 KB 2-way Instruction
16x16KB 4-way Data

L2 Cache 8x2 MB 16-way
L3 Cache 2x8 MB up to 64-way
Memory 128GB
Memory Interface Quad Channel DDR3
OS Centos 6
Compiler GCC 4.9.2
Compiler Flags -O3
OpenCL AMD APP

Reference: [17]

4 Benchmark Results

Using the VexCL library a benchmark was performed on each platform. Several different tests were used to gauge performance including sort, reduce, and scan operations along with vector-vector operations such as add and matrix-vector operations such as SPMV.

![Benchmark Results Chart]
5 Chrono SpMV Benchmark

Along with the synthetic benchmarks performed using the VexCL library, actual matrices from a simulation were used to gauge real world performance. The simulation setup consisted of a kinematically driven vehicle that fords a river comprised of one million rigid, frictionless spheres.

Specifically 8 different sets of matrices will be compared on each platform. The problem being solved is $D^T M^{-1} D x$ which is split into two matrix vector multiplications, first $temp = M^{-1} D x$ and then $Result = D^T temp$. Note that $M$ is a diagonal matrix and $x$ is a vector. The figures below show the simulation output, jacobian matrix $D$ and the results for FLOP rate for the computation.

Fig. 8, and Fig. 9 provide the same data as above in a different format, data is grouped by device name with each bar representing one of the 7 tests that were performed. The results
demonstrate how different sparsity patterns affect the speed that the SpMV operation is performed at. Fig. 10, and Fig. 11 show the results using the Blaze C++ library along with the speedup of VexCL vs Blaze. In most cases Blaze was slightly faster than VexCL.
Figure 8: Combined plots for the CPUs using VexCL
Figure 9: Combined plots for the accelerators using VexCL
Figure 10: Combined plots for the CPUs using Blaze

Figure 11: Speedup for VexCL compared to Blaze for different matrices. A speedup of less than one means that VexCL was slower than Blaze.
References


