On the Characterization of OpenCL Dwarfs on Fixed and Reconfigurable Platforms

K. Krommydas, W. Feng
Dept. of Computer Science, Virginia Tech

M. Owaida, Christos D. Antonopoulos, N. Bellas
Dept. of Electrical and Computer Engineering, University of Thessaly
Talk Outline

• Motivation
• OpenDwarfs Benchmark Suite
• FPGAs and SOpenCL
• Experimental Setup
  – Hardware/Software
• Results/Characterization
  – GEM/NW/SRAD/BFS
• Future Work
• Conclusions
Motivation (1/4)

• Parallel programming revolution delivered parallel computing to the masses

• Many *different* computing platforms
Motivation (2/4)

How am I going to program my application for all these?
Motivation (3/4)

How will I evaluate my architecture and compare it against the others?
Motivation (4/4)

How will I develop back-end optimizations and know they work well?
Contributions

• Introduce the latest version of OpenDwarfs, a benchmark suite, based on the concept of the dwarfs, that is characterized by:

  - Portability
  - Fairness
  - Usability
  - Extensibility

• Showcase the usefulness of OpenDwarfs by characterizing four representative applications on a wide range of parallel architectures, including an FPGA…

• …and provide insight on the interplay between dwarfs’ computation patterns & underlying architectures
OpenDwarfs Benchmark Suite: Idea

• Benchmarks should be related to *scientific paradigms*
  – What the important problems in a scientific domain are
  – What the set of accepted solutions is

• Tangential to the concept of the *dwarfs*
  • Algorithmic methods that capture patterns of computation and communication that are important for science and engineering

---

**Dwarfs**

- Dense linear algebra
- Sparse linear algebra
- Spectral methods
- N-body methods
- Structured grids
- Graphical models
- Finite state machines
- Unstructured grids
- Combinational logic
- Graph traversal
- Dynamic programming
- Backtracking & Branch and bound
- MapReduce
OpenDwarfs Benchmark Suite: Recent Developments (1/3)

- Continuous bug fixes geared towards a first stable release

- Updated implementations for previously brittle dwarfs:
  - Finite state machine dwarf (TDM – Temporal Data Mining)
  - Backtrack & Branch and bound (NQ - N-Queens Solver)
  - Graphical models (HMM – Hidden Markov Models)
OpenDwarfs Benchmark Suite: Recent Developments (2/3)

• Usability and readability/code uniformity, extensibility:
  – Uniform error checking routines and error messages
  – OpenCL initialization routines
  – Run-time device selection (including FPGA support via SOpenCL and Altera OpenCL SDK)
  – Extended set of options (multiple kernel runs, variable work-group sizes, etc.)
Enforcing “de-optimized” implementation: No dwarf contains optimizations that favor one architecture over another

- Does not imply the lack of need for separate optimized implementations per device

Eventually, choice depends on end users’ needs:

- Hardware vendors: optimized
- Compiler writers: unoptimized, optimized
- Independent parts/organizations: unoptimized
FPGAs and SOpenCL: Background

• FPGA (Field Programmable Gate Array):
  ✓ Maximum flexibility in tailoring the architecture
  ✓ Power efficiency
  X Low level programming requirement (VHDL, Verilog)

• Automatic generation of hardware accelerators for Xilinx FPGAs
FPGAs and SOpenCL – Architectural Template

**Sin Kernel:**
- $\text{ind} = \text{phi}[0, \text{preh}], [\text{i2}, \text{body}]$
- $i_0 = \text{add} a_0, \text{ind}$
- $i_2 = \text{add} \text{ind}, 1$
- $i_3 = \text{add} a_0, i_2$
- $\text{gep0} = \text{getelementptr} i_8* x_0, i_0$
- $\text{gep1} = \text{getelementptr} i_8* x_0, i_3$
- $i_7 = \text{load} i_8^* \text{gep0}$
- $i_{10} = \text{load} i_8^* \text{gep1}$

**Computational Kernel:**
- $i_{46} = \text{phi}[\text{true}, \text{preh}], [i_{41}, \text{body}]$
- $\text{ind} = \text{phi}[0, \text{preh}], [i_2, \text{body}]$
- $i_2 = \text{add} \text{ind}, 1$
- $i_7 = \text{pop} i_8^* \text{gep0}$
- $i_9 = \text{mul} i_7, a_3$
- $i_{10} = \text{pop} i_8^* \text{gep1}$
- $i_{12} = \text{mul} i_{10}, a_4$
- $i_{19} = \text{add} i_9, 32$
- $i_{20} = \text{add} i_{19}, i_{12}$
- $i_{23} = \text{ashr} i_{22}, 6$
- $\text{push} i_{23}, i_8^* \text{gep4}$
- $i_{40} = \text{icmp}\text{eq} i_2, 8$
- $i_{41} = \text{xor} i_{40}, \text{true}$
- $\text{br} i_{40}, \text{exit}, \text{body}$

**Sout Kernel:**
- $\text{ind} = \text{phi}[0, \text{preh}], [i_2, \text{body}]$
- $i_2 = \text{add} \text{ind}, 1$
- $i_6 = \text{add} a_2, \text{ind}$
- $\text{gep4} = \text{getelementptr} i_8^* x_1, i_6$
- $\text{store} i_{23}, i_8^* \text{gep4}$
Experimental Setup: Hardware

**Multicore CPU:**
- AMD Opteron 6272

**Intel Many Integrated Core (MIC) Co-processor:**
- Intel Xeon Phi P1750

**Accelerated Processing Units (APUs):**

a) AMD Llano A8-3850 (Llano)

- **Multicore CPU:** AMD Llano A8-3850
- **Integrated GPU (iGPU):** AMD Radeon HD 6550D

b) AMD Llano A10-5800K (Trinity)

- **Multicore CPU:** AMD Llano A10-5800K
- **Integrated GPU (iGPU):** AMD Radeon HD 7660D

**GPU (discrete):**
- AMD Radeon HD 7970

**Field Programmable Gate Array (FPGA):**
- Xilinx Virtex-6 LX760, 1 GB DRAM, PCIx connectivity

For detailed specs refer to the paper!
Experimental Setup: Software

**Benchmarks**

**OpenDwarfs**
- GEM (N-body methods)
- NW (Dynamic programming)
- SRAD (Structured grids)
- BFS (Graph traversal)

**Host systems**

**AMD CPU/GPU/APU:** 64-bit Debian Linux 7.0 (Kernel 2.6.37), AMD APP SDK 2.8, AMD Catalyst 13.1

**Intel Xeon Phi:** CentOS 6.3, Intel OpenCL SDK XE 2013

**FPGA:** Ubuntu 12.04, Xilinx ISE 12.4

**Profiling tools**

**AMD CPU/GPU/APU:** AMD CodeXL 1.3

**Intel Xeon Phi:** Intel Vtune Amplifier XE 2013
Characterization: GEM

Data
- x, y, z coordinates
- electric charge

Surface vertices
- 0
- 1
- ...
- m

Atoms
- 0
- 1
- ...
- n
Characterization: NW

### Work-group-level granularity

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>3</th>
<th>7</th>
<th>11</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Thread-level granularity

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>m=0 tx=0</td>
<td>m=1 tx=1</td>
<td>m=2 tx=2</td>
<td>m=3 tx=3</td>
</tr>
<tr>
<td>1</td>
<td>m=1 tx=0</td>
<td>m=2 tx=1</td>
<td>m=3 tx=2</td>
<td>n=2 tx=2</td>
</tr>
<tr>
<td>2</td>
<td>m=2 tx=0</td>
<td>m=3 tx=1</td>
<td>n=2 tx=1</td>
<td>n=1 tx=1</td>
</tr>
<tr>
<td>3</td>
<td>m=3 tx=0</td>
<td>n=2 tx=0</td>
<td>n=1 tx=0</td>
<td>n=0 tx=0</td>
</tr>
</tbody>
</table>

$m$: for loop index  
$tx$: thread id
Characterization: NW

Execution time (in msec.)

- caterpillarCPU
- llanoCPU
- tnsi2CPU
- llanoGPU
- tnsi2GPU
- trinityGPU
- mic
- FPGA_C1
- FPGA_C2

- nw2_kernel
- nw1_kernel
- Data transfer
Characterization: SRAD

Thread-level granularity

Work-group-level granularity

0 3 7 11 15

3

7

11

15
Characterization: SRAD

Execution time (in msec.)

- srad_kernel2
- srad_kernel1
- Data transfer

Devices:
- caterpillarCPU
- llanoCPU
- tnsi2CPU
- llanoGPU
- tnsi2GPU
- trinityGPU
- mic
- FPGA_C1
- FPGA_C2
Characterization: BFS

![Graph Diagram]

<table>
<thead>
<tr>
<th>Iter.</th>
<th>Kernel</th>
<th>Thread id (tid)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>0   1  2  3  4  5  6</td>
</tr>
<tr>
<td>1</td>
<td>kernel1</td>
<td>✔   X  X  X  X  X  X</td>
</tr>
<tr>
<td>1</td>
<td>kernel2</td>
<td>X   ✔  ✔  ✔  X  X  X</td>
</tr>
<tr>
<td>2</td>
<td>kernel1</td>
<td>X   ✔  ✔  ✔  X  X  X</td>
</tr>
<tr>
<td>2</td>
<td>kernel2</td>
<td>X   X  X  X  ✔  ✔  ✔</td>
</tr>
<tr>
<td>3</td>
<td>kernel1</td>
<td>X   X  X  X  ✔  ✔  ✔</td>
</tr>
<tr>
<td>4</td>
<td>kernel2</td>
<td>X   X  X  X  ✔  ✔  ✔</td>
</tr>
</tbody>
</table>
Characterization: BFS

![Bar chart showing execution time (in msec.) for different nodes and kernels.](chart.png)

- **caterpillarCPU**
- **llanoCPU**
- **tns2CPU**
- **llanoGPU**
- **tns2GPU**
- **trinityGPU**
- **mic**
- **FPGA_C1**

**Execution Time (in msec.):**
- **bfs_kernel2**
- **bfs_kernel1**
- **Data transfer**
Future Work

• Further **development** of OpenDwarf benchmark suite:
  – Input data-set generation
  – Automated result verification
  – OpenACC/OpenMP implementations

• **Conception** and **development** of more “generic” OpenDwarfs

• Further **characterization** of more architectures and OpenCL SDKs:
  – Intel Xeon Phi co-processor (Knight’s Landing + NEW Intel OpenCL SDK)
  – Altera FPGAs (Stratix V + Altera OpenCL SDK)
  – Memory-coherent APUs
Conclusions

- The concept of the **dwarfs** embraced by OpenDwarfs is important for:
  - Understanding the **suitability** of different devices/architectures for different workloads
  - Using such representative workloads in order to **guide** future architectural innovation

---

**OpenDwarfs**

- **Portability** (OpenCL/SOpenCL/Altera OpenCL)
- **Fairness** (baseline unoptimized set of dwarfs)
- **Usability** (simple build system, detailed timing infrastructure)
- **Extensibility** (common utility functions)
For more details, check our paper!

**On the Characterization of OpenCL Dwarfs on Fixed and Reconfigurable Platforms**

K. Krommydas, W. Feng, M. Owaida, C. D. Antonopoulos, N. Bellas