Applying OpenCL

IWOCL, May 2017

Andrew Richards
The next generation of software will not be built on CPUs
“On a 100 millimetre-squared chip, Google needs something like 50 teraflops of performance”

- Daniel Rosenband (Google’s self-driving car project) at HotChips 2016
Performance trends

GFLOPS

Year of introduction

Desktop GPU (200W+)
Integrated Desktop GPU (~15W)
Smartphone GPU (~2W)
Desktop CPU
Mobile CPU
Google target
Desktop GPU
Integrated GPU
Smartphone GPU
Smartphone CPU
Desktop CPU
The rise of the AI processor

- Parallelism of GPU
- Power efficiency of GPU
- Remove graphics

AI processor
How do we connect tomorrow’s software to tomorrow’s silicon?
OpenCL: Our targets for 2017 and beyond

1. Make it fast
2. Make it safe
3. Make it ubiquitous
How do we get performance on accelerators?

- Hand-optimized operations
- Kernel fusion
- Custom operations
In this example, we perform 3 image processing operations on an accelerator and compare 3 systems when executing individual nodes, or a whole graph.

The system is an AMD APU and the operations are: RGB->HSV, channel masking, HSV->RGB.

Halide and SYCL use kernel fusion, whereas OpenCV does not. For all 3 systems, the performance of the whole graph is significantly better than individual nodes executed on their own.
Applying fusion to TensorFlow Eigen

This is how TensorFlow uses Eigen to achieve kernel-fusion.

CUDA does this for NVIDIA GPUs, SYCL is used here for AMD GPUs.

Unfused performance improvement: AMD GPU vs multi-core Intel CPU

Total performance improvement delivered by SYCL is both of these graphs combined.
How do we combine our requirements?

Hand-optimized operations + Kernel fusion + Custom operations = ?
How do we fuse custom and hand-coded kernels?

Custom operations written in fusable form

Hand-coded operations written in fusable form

Kernel fusion system
• We need a language and compiler that:

1. Lets users easily write custom operations

2. Lets hardware experts drill-down and write device-specific optimized code

3. Allows code to be efficiently fused
We need a language and compiler that:

1. Lets users easily write custom operations
   - C++ is a well-understood programming language that programmers can use

2. Lets hardware experts drill-down and write device-specific optimized code
   - C++ allows expert programmers to write low-level device-specific optimized code

3. Allows code to be efficiently fused
   - C++ single-source lets us fuse kernels
   - ... and is already used
We need a language and compiler that:

1. Lets users easily write custom operations

2. Lets hardware experts drill-down and write device-specific optimized code

3. Allows code to be efficiently fused

- But now we have SPIR/SPIR-V, you could write your own compiler to solve this
Make it safe
• Our challenges:
  • We need all the tools to follow standard safety-critical processes
  • We need predictable timing
  • We need to test the OpenCL implementations thoroughly
  • We need to test OpenCL code in extreme situations
  • We need to be able to handle highly parallel errors and recovery
OpenCL SC: We need to work together

• Each challenge is a massive challenge
• We need to come together to solve these challenges
  • Academics and industry
  • Parallelism, safety, automotive, medical, formal methods, testing...
Make it ubiquitous
Making OpenCL ubiquitous

Jon Peddie Research: Feb 2017 report on the VPU market

Figure 1: Population of companies making VPUs over time
Why OpenCL for new AI/vision processors?

• It’s royalty-free
• It’s programmable
• It’s very widely supported already
• Providing OpenCL brings in a wide ecosystem of software:
  • OpenCV, Halide, SYCL, OpenVX, clBLAS/clBLAST, TensorFlow, Caffe, ViennaCL, Boost.compute, ....
But, what do we need to solve?

• We need to bring OpenCL to devices that are not GPUs
  • And so we need to focus on adding non-GPU features
  • And removing GPU features

• While also still supporting the capabilities of GPUs
  • and CPUs, FPGAs, DSPs...
• We need to build out the ecosystem
  • Make it easier to bring OpenCL to new devices
  • Make it easier to test OpenCL devices
  • Make it easier to find all the existing OpenCL software
• SYCL  http://sycl.tech
• OpenCL  https://www.khronos.org/opencl/
• OpenVX  https://www.khronos.org/openvx/
• OpenCV  http://opencv.org/
• Halide  http://halide-lang.org/
• VisionCpp  https://github.com/codeplaysoftware/visioncpp
• OpenCL org  http://opencl.org/
• CLsmith  http://multicore.doc.ic.ac.uk/tools/CLsmith/clsmith.php
• TensorFlow OpenCL  http://ci.tensorflow.org/view/OpenCL/
What do you want to accelerate?

@codeplaysoft  info@codeplay.com  codeplay.com