

OpenCL SC13 BOF Report

Approximately 75 people attended the OpenCL Birds of a feather (BOF) which was held during the Wednesday evening BOF time slot (5:30 to 7:00 PM). The attendees were vocal, engaged, and enthusiastic.

We passed out reference cards for OpenCL 2.0 and the C++ wrapper. These reference cards are professionally produced, laminated summaries of the OpenCL specifications. They are extremely useful for anyone wanting to work with OpenCL. We normally sell them on Amazon for 13 dollars each ... so this is a valuable “free benefit” we offer to the OpenCL community at SC13.

This year we added a new feature to the BOF. We passed out a survey to collect feedback from attendees. Survey questions and a summary of the results are presented later in this report.

This report was edited by Tim Mattson of Intel. Feedback was provided by Tim Mattson, Simon McIntosh-Smith, and James Price (both from Bristol University).

OpenCL 2.0 overview

Tim Mattson presented a brief overview of OpenCL 2.0; the latest version of OpenCL released on Monday November 18. He just focused on the high level features:

- SVM
- The new memory model and C++ atomics
- Nested parallelism

We also briefly discussed SPIR which should be released soon.

News from the world of OpenCL

Simon McIntosh-Smith described news from the world of OpenCL. He spoke of IWOCL this spring in the UK and more.

OpenCL for FPGA

John Freeman and Michael Kinsner spoke about the use of OpenCL with FPGAs. As with all the talks at the BOF, they kept things simple and to the point. They briefly presented an overview of the FPGA value proposition. Then outlined the features and benefits of using OpenCL with FPGAs. The key value proposition relative to a C-to-HW tool chain is that with OpenCL, you generate code for the entire platform and automatic timing closure. Literally, a single OpenCL program can be used across the full range of platforms from CPU to GPU to FPGA.

OpenCL Panel/Discussion

Question: So what the hell is up with HSA?

Jeff - OpenCL on top of HSA, application developers won't know HSA is there. Other vendors on board too. Fairly transparent.

Simon - OpenCL should be the only way to use HSA, not direct programming.

Jeff - Want to give app devs OpenCL and low level devs (libraries etc) HSAIL

Tim - The messaging around HSA has confused the market. I've heard high level HSA-people from some OpenCL member companies state that HSA is better than OpenCL since it is so much easier for programmers to use. This is a very damaging message to use. And as we release SPIR it will be even worse. HSA has done more damage than good and I wish it would just go away.

Question: What do you love about OpenCL?

Altera - Only need to learn one thing to use all platforms

Simon - Portability and performance portability (with a small amount of work)

Question: What do you hate about OpenCL?

Altera - Not performance portable

Tim - I am so tired of hearing people moan about performance portability in OpenCL. There is no such thing as performance portability ... even with serial programs written with C. We get 5 to 10 percent of peak performance from a typical C program. If you shoot for a major fraction of peak performance, you must modify source to the platform. That is the same with C and serial programming. And it is the same with OpenCL.

Altera - Still very tuned to a GPU

Simon - Vendor support still very flaky

Jeff - Hard to leverage multiple devices effectively

Question: Difficult to manage memory address spaces and host-device transfers

Tim - OpenCL now has SVM, although fine-grained only optional

Simon - C++ wrapper makes this easier

Question: Why can't you make the C API suck less?

Tim - Have to stick to lowest common denominator. Functional portability across very different platforms means limitations and verbosity

Question: Latest C++ API better, but what if you're not a C++ programmer?

Tim - CLU project on GitHub, makes C API a lot easier

Question: Can you pass structs with pointers in as arguments?

Tim - Yes with coarse grained SVM

Comment: I'd like to see more interoperability with other standards (e.g. with OpenMP buffers).

OpenCL BOF survey

We gave the attendees of the BOF a survey to help us gather direct information about where we should take OpenCL in the future. The following are the survey questions and a summary of the responses.

Describe yourself

<p>1. Which label best fits your organization?</p> <ul style="list-style-type: none">a. Commercial, software vendorb. Commercial, hardware vendorc. Commercial, otherd. Academice. Government labf. Government funding agency	A:7 B:3 C:5 d:15 d:4 f:1
<p>2. Which label best characterizes your perspective when it comes to HPC software development?</p> <ul style="list-style-type: none">a. An application programmerb. A runtime system, library, tool or language developerc. A scientist who uses HPC systems as an end-user, not a programmerd. A marketer, manager, planner or other non-programming capacitye. IT or user supportf. Other	A:13 b:12 c:2 d:4 e:2 f:3
<p>3. How much OpenCL programming do you do?</p> <ul style="list-style-type: none">a. I am a proficient OpenCL programmer writing OpenCL code on a regular basis.b. I experiment with OpenCL but I don't do serious software development with the language.c. I am interested in OpenCL but have never written any code.d. I am a parallel programmer but have no plans to use OpenCLe. I am not a programmer and don't write OpenCL or any other parallel code.	A:13 B:7 C:8 d:2 e:2

Future developments in OpenCL

<p>4. Rank the following least from most to least important:</p> <ul style="list-style-type: none"> a. Support for some version of a C++ kernel programming language. b. A high level language that masks the need to manage low level details of OpenCL. c. Concurrent execution guarantees between kernels or other “units of execution” to support a richer range of algorithms d. A set of high level algorithms such as pipeline parallelism or data-flow-graphs. 	<p>Results all over the map. Most common: BDCA: 8 BACD: 4 Top ranked items: A:7 B:15 c:5 d:2 Second ranked items A:5 B:5 C:3 D:15</p>
<p>5. Select which statement you most agree with</p> <ul style="list-style-type: none"> a. C++ as a kernel programming language isn’t important at all so just focus on making the C kernel programming language as great as possible. b. IF you support C++, support as much of the language as possible including dynamic features (i.e. virtual functions, etc.), even if this makes it difficult for some devices to support. c. If you support C++, it’s OK to only support a simplified (static) subset that all devices can support. d. You’ll never be able to keep up with the kernel programming languages people need. Just get SPIR right and work to enable a rich set of languages that map onto SPIR 	<p>A: 7 B: 4 C:9 D:14</p>
<p>6. Which statement comes closest to capturing your attitude about future developments of OpenCL?</p> <ul style="list-style-type: none"> a. We need new features to expand the range of applicability of OpenCL. b. We need feature parity across platforms so we can write high quality, portable code. c. We need performance and reliability on a single platform. 	<p>A:4 b:16 C:7</p>

Additional comments ... in a few simple words so we can easily consolidate your feedback.

7. What (if anything) is stopping you from using OpenCL today?
There was mention of a lack of C++ kernel support, lack of portability, lagging support for the latest HW trends and even in one case a preference for CUDA. But the most common reply was that OpenCL (especially the host API) is just too complex (4 people). The second most common response was to complain about poor support of OpenCL by NVIDIA (3). The other comments (two people each) were a lack of software development tools and inconsistent support for the latest OpenCL standard between vendors.
8. What do you like best about OpenCL?
The near universal response was "portability". People get the message that OpenCL is the standard for cross-vendor support of heterogeneous platforms.
9. If you could add just one new feature to OpenCL, what would that be?
Responses were diverse ranging from support for java to a call for load balancing between devices. Two people each asked for better tool support and C++. But to follow the discussion on this question, you really need to look at the responses in the spread sheet.
10. What are your comments or feedback about the reference cards or online reference pages?
Most people did not respond. Of the few who did respond, the feedback was positive.
11. Is there anything else you'd like to say to the OpenCL language committee (either positive or negative)?
 - *Profiles for different classes of devices (GPU, CPU, FPGA, etc)*
 - *Better cooperation between major HW vendors.*
 - *Make OpenCL so compelling, NVIDIA will have to "come on board".*
 - *Make OpenCL a superset of CUDA.*
 - *SPIR will only have a significant impact if it is required, not optional*
 - *Portability is important, but only if we can get 2/3 of performance you'd get from a native API such as CUDA or SSE.*