Achieving Performance with OpenCL 2.0 on Intel® Processor Graphics

Robert Ioffe, Sonal Sharma, Michael Stoner
INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

A "Mission Critical Application" is any application in which failure of the Intel Product could result, directly or indirectly, in personal injury or death. SHOULD YOU PURCHASE OR USE INTEL'S PRODUCTS FOR ANY SUCH MISSION CRITICAL APPLICATION, YOU SHALL INDEMNIFY AND HOLD INTEL AND ITS SUBSIDIARIES, SUBCONTRACTORS AND AFFILIATES, AND THE DIRECTORS, OFFICERS, AND EMPLOYEES OF EACH, HARMLESS AGAINST ALL CLAIMS COSTS, DAMAGES, AND EXPENSES AND REASONABLE ATTORNEYS' FEES ARISING OUT OF, DIRECTLY OR INDIRECTLY, ANY CLAIM OF PRODUCT LIABILITY, PERSONAL INJURY, OR DEATH ARISING IN ANY WAY OUT OF SUCH MISSION CRITICAL APPLICATION, WHETHER OR NOT INTEL OR ITS SUBCONTRACTOR WAS NEGLIGENCE IN THE DESIGN, MANUFACTURE, OR WARNING OF THE INTEL PRODUCT OR ANY OF ITS PARTS.

Intel may make changes to specifications and product descriptions at any time, without notice.

All products, dates, and figures specified are preliminary based on current expectations, and are subject to change without notice.

Intel processors, chipsets, and desktop boards may contain design defects or errors known as errata, which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Any code names featured are used internally within Intel to identify products that are in development and not yet publicly announced for release. Customers, licensees and other third parties are not authorized by Intel to use code names in advertising, promotion or marketing of any product or services and any such use of Intel's internal code names is at the sole risk of the user.

Intel product plans in this presentation do not constitute Intel plan of record product roadmaps. Please contact your Intel representative to obtain Intel’s current plan of record product roadmaps.

Performance claims: Software and workloads used in performance tests may have been optimized for performance only on Intel® microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. For more information go to http://www.Intel.com/performance

Intel, Intel Inside, the Intel logo, Centrino, Intel Core, Intel Atom, Pentium, and Ultrabook are trademarks of Intel Corporation in the United States and other countries.
One more short message from our lawyers 😊

FTC Disclaimer

Software and workloads used in performance tests may have been optimized for performance only on Intel® microprocessors. Performance tests are measured using specific computer systems, components, software, operations, and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products.
Acknowledgements

Thanks to Stephen Junkins, Aaron Kunze, Allen Hux, Adam Lake, Michał Mrozek, Ayal Zaks, Deepti Joshi, Ben Ashbaugh and Yuri Kulakov for input and feedback on this presentation and GPU-Quicksort sample!
Agenda

Shared Virtual Memory in OpenCL 2.0
  • Crowd Simulation algorithm
  • Border pixel processing
  • Cyberlink PowerDirector usage

Device Side Enqueue and Work-group Scan Functions in OpenCL 2.0
  • Usage and Benefits
  • Sierpinski Carpet Example
  • GPU-Quicksort Example
Shared Virtual Memory

Allows de-referencing of host-allocated virtual memory pointers directly on the GPU.

Enables GPU offload of pointer-oriented algorithms (e.g. using trees or linked lists).
Crowd Simulation Example

What type of algorithm can actually benefit from SVM?
Crowd Simulation Example

What type of algorithm can actually benefit from SVM?
What type of algorithm can actually benefit from SVM?
Crowd Simulation Example

What type of algorithm can actually benefit from SVM?
Crowd Simulation Example

What type of algorithm can actually benefit from SVM?
What type of algorithm can actually benefit from SVM?
`orcaLines_` – pointer to SVM

- Pointer contained in “Agent” struct
- No need for `clCreateBuffer()`, but must use `clSetKernelExecInfo()`

```c
__kernel
void computeNewVelocity(__global PAgent* agents, …)
{
  __global Agent* agent =
  agents[get_global_id(0)].value;
  …
  agent->orcaLines_[agent->numOrcaLines_++] = line;
}
```
Further gains possible with heterogeneous mode, issuing OpenCL threads concurrently on GPU and CPU.
SVM ISV usage – Cyberlink Photo Director

CPU/GPU border pixel processing strategy
- Remove border bounds-checks from GPU kernel, process border pixels on the CPU in parallel

Fine-grain SVM buffer

5% gain vs. serial execution
Cyberlink PowerDirector 13 Effects Performance

![Graph showing performance comparison between different media effects using CPU, OpenCL 1.2, OpenCL 2.0, and OpenCL 2.0/SVM configurations. The x-axis represents media effects, and the y-axis represents production time in seconds. The graph also indicates the percentage increase in performance.]

* Permission to share Information from Cyberlink Corp.
SVM Summary

Coarse-grain SVM available on Intel 5th Generation Processors with HD Graphics 5300+

- Supports virtual-memory pointer access from GPU kernels
- No longer need to marshal buffers into ‘cl_mem’ constructs
- No alignment or size restrictions to achieve zero-copy buffer sharing

Fine-grain SVM available in Intel 5th Generation Processors w/ HD Graphics 5500+

SVM samples available on Intel® Developer Zone

- SVM Basic sample
- CrowdSim coming soon!
Part II:

Device Side Enqueue and Work-group Scan Functions in OpenCL 2.0
Device Side Enqueue

Device kernels can enqueue kernels to the same device with no host interaction, enabling flexible work scheduling paradigms and avoiding the need to transfer execution control and data between the device and host, often significantly offloading host processor bottlenecks*

Introduced to OpenCL 2.0 to express recursive and iterative algorithms

We are going to use Sierpiński Carpet as a simple example to show all the building blocks of the device side enqueue

*Khronos Finalizes OpenCL 2.0 Specification for Heterogeneous Computing
The **Sierpiński carpet** is a plane fractal first described by [Wacław Sierpiński](https://en.wikipedia.org/wiki/Wac%C5%82aw_Sierpi%C5%84ski) in 1916.

Start with a white square.

Divide the square into 9 sub-squares in a 3-by-3 grid.

Paint the central sub-square black.

Apply the same procedure recursively to the remaining 8 sub-squares.

And so on …


Sierpiński Carpet Kernel in OpenCL 2.0

```c
__kernel void sierpinski(__global char* src, int width, int offsetx, int offsety)
{
    int x = get_global_id(0);
    int y = get_global_id(1);
    queue_t q = get_default_queue();

    int one_third = get_global_size(0) / 3;
    int two thirds = 2 * one_third;

    if (x >= one_third && x < two thirds && y >= one third && y < two thirds)
    {
        src[(y+offsety)*width+(x+offsetx)] = BLACK;
    }
    else
    {
        src[(y+offsety)*width+(x+offsetx)] = WHITE;

        if (one_third > 1 && x % one_third == 0 && y % one_third == 0)
        {
            const size_t grid[2] = {one_third, one_third};
            enqueue_kernel(q, 0, ndrange_2D(grid), ^{ sierpinski(src, width, x+offsetx, y+offsety); });
        }
    }
}
```

Easy to translate recursive algorithm to implementation
Sierpiński Carpet - Result

2187x2187 image: $8^6 = 299592$ enqueue_kernel calls!
How Recursive Version Compares to Iterative?

Recursive: 2050 ms 😞
Iterative: 11 ms

Solution: combine the two, start w/ recursive, switch to iterative for 243x243 tiles
Mixed version: 10.45 ms 😊 - speedup of 1.05X for 2187 by 2187 image
Speedup improves for larger image sizes: 1.15X for 6561 by 6561 image
1.23X for 19683 by 19683 image

enqueue_kernel will improve performance when used properly!
GPU-Quicksort - Overview

Invented by Daniel Cederman and Phillipas Tsigas

- Student and Professor pair
- At Chalmers University of Technology
- Invented in 2007, written in CUDA
- Improves on the work of Shubhabrata Sengupta, Mark Harris, Yao Zhang, and John D. Owens

First phase:

- Workgroups work on different parts of the same sequence
- Each workgroup partitions a block assigned to it around the pivot
- The partitioned blocks are merged
- The last workgroup writes one or more pivot values between the sequences
- We repeat the first phase until each subsequence is short enough to be sorted by one workgroup

Second phase:

- Each workgroup is assigned its own subsequence of $\leq$ QUICKSORT_BLOCK_SIZE (e.g. 1536) elements
- Use explicit work stack to simulate quicksort recursive calls within the kernel
- Use Bitonic sort when the number of elements in subsequence is $\leq$ SORT_THRESHOLD (e.g. 512)

* Photos of Daniel Cederman and Prof. Phillipas Tsigas from their research group’s home page
GPU-Quicksort in OpenCL 2.0

Switch to work group scan functions

- `work_group_scan_exclusive_add` in `gqsort_kernel`
- `work_group_scan_exclusive_add` and `work_group_scan_inclusive_add` in `lqsort_kernel`
- Performance gain of 8% compared to Blelloch algorithm in 1.2
- Gain in code conciseness, maintainability and clarity
  - code size reduced almost 3X

Take advantage of `enqueue_kernel` function

- Move the logic that calculates block and parent records, sorts the records after each `gqsort_kernel` run, and launches either `gqsort_kernel` or `lqsort_kernel` to GPU
- Dramatically simplify launch from CPU: just `relauncher_kernel`
- Simplified `gqsort_kernel` and `lqsort_kernel` CPU wrapper code: we use `blocks` on GPU

GPU-Quicksort is 44% to 62% faster on Intel HD Graphics 5500 when implemented in OpenCL 2.0 vs OpenCL 1.2!
GPU-Quicksort in OpenCL 2.0 Performance

HD Graphics 5500: Speedup vs std::sort
X axis - input size, Y axis - speedup

HD Graphics 6000: Speedup vs std::sort
X axis - input size, Y axis - speedup

Parallel CPU-Quicksort
GPU-Quicksort in OpenCL 1.2
GPU-Quicksort in OpenCL 2.0
Conclusions

Device side enqueue is a powerful addition to the OpenCL programmer toolbox

- Provides the ability to port recursive and iterative algorithms to OpenCL
- Might dramatically improve performance when used properly
- Syntactically comparable to CUDA's dynamic parallelism feature

enqueue_kernel and more available in Intel's OpenCL 2.0 driver

GPU-Quicksort for OpenCL 1.2 runs very well on Intel® Processor Graphics

GPU-Quicksort for OpenCL 2.0 runs even better

- Due to optimized work group scan functions
- Due to enqueue_kernel functions, which avoid round trips to the CPU

**Intel HD Graphics 5500 with Intel's OpenCL 2.0 driver is a powerful platform for writing high performance algorithms!**
Key Takeaways

SVM

• Use it for all your shared pointy data structure needs
• Coarse grain and Fine grain flavors w/ atomics are available on Intel’s 5th Generation Processors

Device side enqueue

• Use it to implement and port high-performance recursive and iterative algorithms
• Avoid round-trips to the host

New work group functions

• Simplify you code for scan, reduce and other common group ops
• Use high-performance implementations optimized for Intel hardware
GPU-Quicksort Bibliography


