NBody Simulation Multi-Device Version

Sample User's Guide

Intel® SDK for OpenCL* Applications - Samples

Document Number: 328221-004US

Contents

Contents	2
Legal Information	3
About Multi-Device Version of the NBody Simulation Sample	
Introduction	4
Motivation	4
Hardware Considerations	4
Right Way to Study the Sample	5
Understanding OpenCL* Performance Characteristics	
Future Work and Potential Enhancements	
Reference (Native) Implementation	6
Controlling the Sample	
	7

Legal Information

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 NEGLIGENT 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. Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined". Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The information here is subject to change without notice. Do not finalize a design with this information.

The products described in this document 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.

Contact your local Intel sales office or your distributor to obtain the latest specifications and before placing your product order.

Copies of documents which have an order number and are referenced in this document, or other Intel literature, may be obtained by calling 1-800-548-4725, or go to:

http://www.intel.com/design/literature.htm.

Intel processor numbers are not a measure of performance. Processor numbers differentiate features within each processor family, not across different processor families. Go to: http://www.intel.com/products/processor_number/.

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.

Intel, Intel logo, Intel Core, VTune, Xeon are trademarks of Intel Corporation in the U.S. and other countries.

* Other names and brands may be claimed as the property of others.

OpenCL and the OpenCL logo are trademarks of Apple Inc. used by permission from Khronos.

Microsoft product screen shot(s) reprinted with permission from Microsoft Corporation.

Copyright © 2010-2013 Intel Corporation. All rights reserved.

Optimization Notice

Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice.

Notice revision #20110804

About Multi-Device Version of the NBody Simulation Sample

NBody algorithm, used in this sample, is straightforward each-to-each approach. This version of demo, demonstrated at Siggraph 2012, targets both Intel CPU and Intel® HD Graphics OpenCL* devices and also enables utilization of both devices with load-balancing.

The NBody sample demonstrates advantages of heterogeneous execution, through the example of simultaneous utilization of CPU and Intel HD Graphics OpenCL devices.

The automatic load-balancing approach is exactly the same as in multi-device version of the Tone Mapping SDK sample. Specifically it relies on the coarse-grain job splitting that tracks some short history to adapt to the changing speed of devices, for example, due to Turbo frequency changes. Also quite similar optimizations to resource sharing were applied to this sample.

The GUI also enables to perform manual load-balancing and compare results with "native" implementations of the same NBody algorithm, for example OpenCL version and the version in openmp+intrinisics.

Introduction

This sample demonstrates simultaneous utilization of Intel CPU and Intel HD Graphics OpenCL devices through the example of NBody algorithm. Intel OpenCL implementation enables interacting with Intel HD Graphics and CPU devices by use of Common Runtime. Refer to the Optimization guide for more information.

You can also create a "shared" context for both devices, to provide resource sharing between devices. Since the OpenCL specification 1.2 does not assume any "shared queue", the sample logic manages the job distribution for the devices.

Motivation

Simultaneous execution on both Intel CPU and Intel HD Graphics devices requires proper synchronization, for example for operations with shared resources. It is also important to follow certain alignment recommendations to get true resource sharing between devices. The sample demonstrates basic approaches to both tasks.

Static work assignment for the devices might result in lower overall performance. The sample demonstrates a strategy for adaptive work partition between the devices (command queues).

General optimization strategy for a heterogeneous application is to maximize the utilization of the underlying devices by keeping them busy most of the time. It is important to avoid transfer latencies. The sample implements basic flip-flop technique when output buffer from the first frame becomes the input buffer for the next frame, and so on, resulting in eliminating any copies between frames.

Similarly the memory objects are also efficiently shared between CPU and Intel HD Graphics OpenCL devices. Refer to the Optimization Guide for more information.

Hardware Considerations

The goal of the sample is to demonstrate heterogeneous execution on both CPU and Intel HD Graphics devices. Thus to be able to use Intel® HD Graphics device separately or in heterogeneous CPU+GPU mode, run the sample on the 3^{rd} Generation Intel® Core[™] Processors.

On lower versions of Intel Core Processors supported by the SDK, only CPU OpenCL device is available. See the User's Guide and the Release Notes for the list of supported platforms.

The sample does not work on other vendors CPUs.

You can evaluate performance of ATI* OpenCL implementation on a system with Intel CPU. To evaluate GPU performance of ATI OpenCL implementation, plug in a discrete ATI graphics card into a system with Intel CPU. You can also evaluate heterogeneous CPU+GPU performance for this particular setup. But this hardware configuration is not optimized and may result in lower performance.

Right Way to Study the Sample

Running single device cases is conceptually identical, while the case of a shared context is different. Consider stepping through an application in debugger to understand its basic logic, including creation of shared context, resources and initialization routine, also refer to the iNBody::Setup() function. For splitting the job between the devices, firing the tasks and synchronization for results refer to the ExecuteNBodyKernel function, which is good starting point.

You can study "native" implementations of the same algorithm and capture the resulted performance to compare the results before walking through the code of Execute_NBody_C_Parallel or Execute_NBody_SSE_Parallel functions.

Also try manual load-balancing and various visualization modes for better understanding of the sample algorithm

Refer to the multi-device version of the Tone-Mapping sample and Optimization Guide for more information.

Refer to the "Controlling the Sample" section to study single- and multi-device cases.

Understanding OpenCL* Performance Characteristics

The actual benefit of using both devices depends on several factors. First is the relative speed of underlying devices: comparative numbers for different processors in and relative benefits of using both CPU and Intel HD Graphics devices simultaneously. Generally, the more similar the performance of the devices, the more gains you get when using both of them. Another important factor is application characteristics like fraction of parallel work, data dependencies and requirements for synchronization.

The sample outputs resulting fps, which includes *rendering* and *synchronizations*. To understand whether *computations* scale well with respect to multiple device usage, use tools or refer to the OpenCL* profiling events, associated with kernel execution. Alternatively, increase number of bodies to simulate by use of a slider in UI. The higher is the number of bodies, the closer the speed of heterogeneous mode to the plain sum of the devices. As rendering and various synchronizations do not scale, so larger amount of computations amortizes them better; and the more computations, the better is the scalability with respect to using both devices.

To understand performance characteristics, consider the following experiments:

- Disable simulation and check what the rendering contribution to a frame cost breakdown is. For more information refer to the Controlling the Sample section.
- Make the number of simulated bodies small in either mode (CPU-only, GPU-only or CPU+GPU), so that execution takes less time. What is important is the difference between fps for the smallest number of bodies and fps for the case when simulation is completely disabled, so OpenCL code pass is completely eliminated. This difference exactly counts overheads associated with OpenCL. These overheads drag down the speedup of heterogeneous mode.
- For CPU+GPU case, increase the number of simulated bodies to improve amortization of overheads, and check how close you can get to the theoretical peak of ideal performance aggregation of the devices.
- Track the overall time like the sample does by measuring FPS. The overall time includes all overheads, transfers and rendering but not the speedup of the kernel itself, which is tracked separately, for example, for computing the job splitting ratio. Difference between "pure kernel" speedup and overall speedup might indicate any inefficiency, related, for example, to additional synchronization and/or resource sharing.
- Try manual load-balancing and compare results to automatic load-balancer.
- For your convenience the application reports current FPS and also tracks maximum value achieved. You can reset the latter value through GUI.

Future Work and Potential Enhancements

Current job splitting strategy keeps the devices loaded by keeping execution time for each device approximately the same. It is also important to minimize the *overall* execution time. For example, an advanced load-balancer would move the entire job to the faster device if heterogeneous mode is slower than single-device case. Then the load-balancer would perform some probing to check if it is beneficial to switch back to heterogeneous mode.

You may also want to implement power-aware load-balancing approach.

Even though the major pieces of the processed data are shared between CPU and Intel HD Graphics devices, still part of memory (bodies positions) are copied to DirectX* structures for rendering. You can avoid this overhead by sharing this data between DirectX and OpenCL.

You can try different load-balancing schemes, like finer-grain, which is splitting job into more than two pieces. Refer to the Optimization Guide for details.

You can also try multi-threading approach, when you dedicate a physical CPU core for scheduling. The device fission extension reserves a core for scheduling GPU tasks. This might prevent GPU starvation in certain cases.

Reference (Native) Implementation

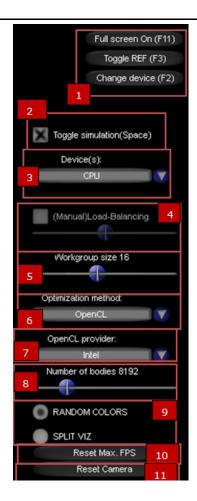
Multiple reference implementations, which range between single-threated code and multi-threaded code in SSE, are available in $Execute_NBody_XXX()$ routines of NBody.cpp file. Also there is a validation routine in $Execute_NBody_Validate$, which you can trigger by use of command-line option.

Controlling the Sample

The command-line argument that triggers kernel results validation routine is "-v".

Following are available GUI controls in groups (refer to the numbers in Figure 1):

- 1. General DirectX application controls:
 - Full-screen mode
 - Rendering device settings
 - Device selection (for rendering)
- 2. **Simulation** checkbox that stops NBody simulation, but does not affect rendering.
- Drop-down list of **Devices** available for the selected platform.
- Enabling manual Load-Balancing, the mode when you specify the ratio to split the jobs between devices with the slider.
- Workgroup (local) size selection slider, ranging between 1 and 512.
- 6. Optimization method:
 - "native" implementations that run on CPU-only (C and SSF)
 - OpenCL for which you can select the device(s).
- 7. **OpenCL platform** (provider) selection. You can select between Intel and other vendors' implementations installed on your system.
- 8. Number of bodies selection slider.
- 9. Visualization mode:
 - Random coloring of the particles
 - Split visualization mode, when particles colors depend on the device, which simulates them.



- 10. Application tracks current and maximum fps. You can reset maximum FPS by clicking Reset Max. FPS button. It might be convenient in case you increase the number of simulated bodies.
- 11. To reset camera position, click the **Reset** button.

References

- http://www.khronos.org/assets/uploads/developers/library/2010_siggraph_bof_opencl/OpenC L-BOF-Intel-SIGGRAPH-Jul10.pdf
- "The Future of OpenCL for Graphics & Film Applications on Intel Platforms". Tech talk at Siggraph 2012. http://software.intel.com/file/45639
- http://software.intel.com/en-us/articles/vcsource-samples-hdr-tone-mapping-multi-device/
- Intel SDK for OpenCL Applications Optimization Guide. Intel SDK for OpenCL Applications User Guide.