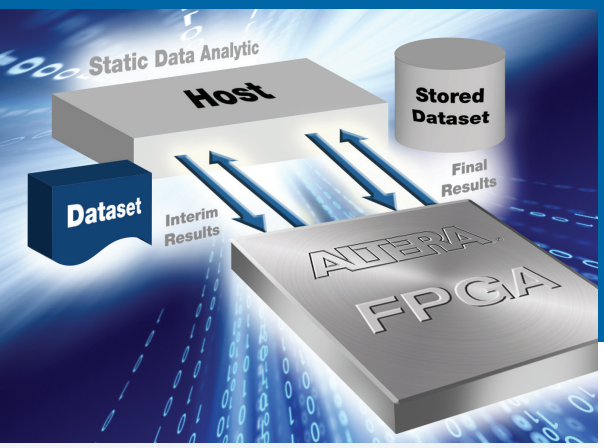


Hardware Acceleration for Map/Reduce Analysis of Streaming Data Using OpenCL

*By **Jim Costabile**, CEO/Founder, Syncopated Engineering Inc.*

syncopated ENGINEERING

The Design Team:

Syncopated Engineering is a creative solution provider of software applications and embedded systems for wireless communications, signal processing and data analytics.

Challenge:

Map/Reduce applications, which break a huge static data set into small pieces, map those pieces to independent processors, and then analyze the data on each processor, are well-supported by industry-standard frameworks such as Hadoop. But there has been no easy way for developers to move code from a static Map/Reduce environment into an environment in which the data streams through the application. This change often required a complete recoding of the application and the use of hardware accelerators to keep up with the data stream.

Solution:

By coding the Map/Reduce functions as OpenCL kernels, developers can move from a static-data model on CPUs to a streaming model using hardware acceleration with virtually no application code changes, exploiting new streaming extensions to the underlying OpenCL platform.

The Project

Typically analysts and analytic developers will pursue the solution to a complex problem by operating on static data to explore, refine, and optimize the analytic prior to deploying it as an operational analytic. This is often called a data-at-rest approach. Many times this development and discovery process produces a solution that would provide significant value if it operated on real-time, dynamic data in a streaming fashion—a data-in-motion implementation.

The Design Challenge

The challenge is to migrate these analytics from a static data analysis architecture to a dynamic streaming architecture. Today, the underlying computing infrastructure, analytic framework, and programming languages for static and dynamic architectures are different and non-compatible.

The analytic developer needs the ability to develop and verify a static data analytic, and then in a simple and straightforward manner migrate this analytic to a streaming analytic architecture. Ideally, this migration process would produce a functionally equivalent analytic capable of operating with high throughput and low latency to keep up with the streaming data rates.

The current state-of-the-art for static data analytics is the Hadoop Map/Reduce software framework. Hadoop Map/Reduce implements the Map/Reduce parallel processing model on large clusters of commodity computers. Map/Reduce jobs typically include a user defined map function that operates on input key/value pairs producing intermediate key/value pairs, and a user defined reduce function that merges and operates on all intermediate values associated with a specific intermediate key to produce a reduced output result. This framework lends itself to data-parallel processing where the input data is partitioned and processed in parallel with a large number of concurrent and identical map functions followed by a smaller number of concurrent and identical reduce functions.

Hadoop is a static data analytic framework written in Java and deployed across a large cluster of commodity machines. This cluster necessarily has a large size, weight, and power (SWAP) footprint. If you need more performance you have to add more commodity machines, at the expense of an even larger SWAP footprint. If you want to use the same analytic on streaming data, you have two choices: store the data first--effectively turning the data-in-motion scenario into a data-at-rest scenario--or re-write the analytic in a streaming analytic framework.

The Design Solution

Our solution is an implementation of the map/reduce parallel processing model using the OpenCL heterogeneous parallel programing framework. Analytic developers write custom map and reduce functions as OpenCL kernels and leverage Syncopated Engineering's map/reduce implementation to quickly deploy map/reduce applications as either static or streaming data analytics on commodity CPU architectures as well as hardware accelerated heterogeneous computing architectures using FPGAs. In order to demonstrate our OpenCL map/reduce framework, we developed a document filtering application that takes a collection of documents as the input, determines the term or word frequency count per document and then uses topic profiles to score the documents to determine the most relevant documents to the topic. We used a collection of technical papers as the input document set, and used the abstracts to create specific topic profiles.

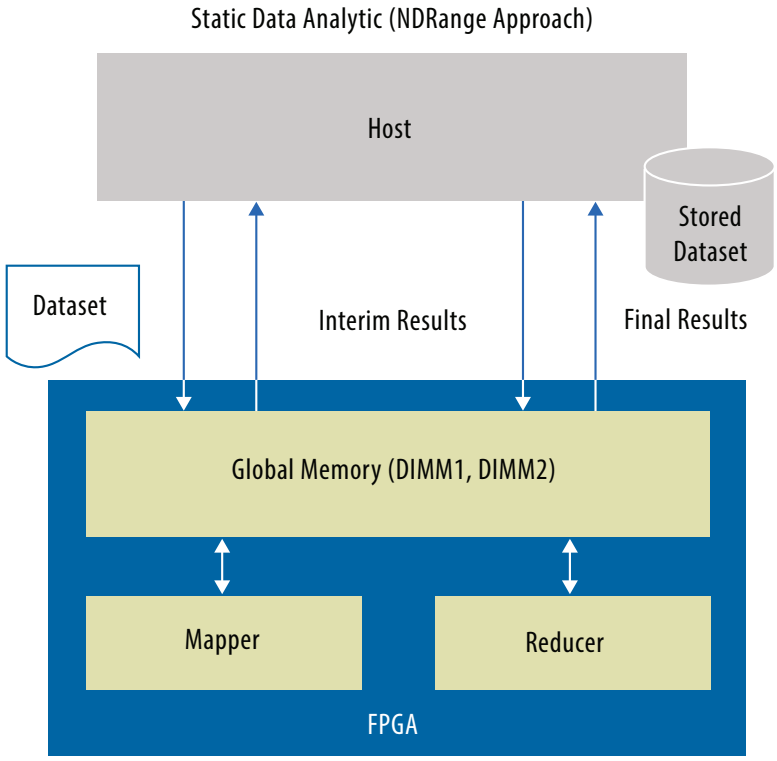
Theory of Operation

The document filtering application includes a pre-initialization process that transforms the input document set into a collection of <term, frequency> pairs per document, where the term is a word in the document and the frequency is the frequency of occurrence of that word in the document. The topic profiler performs the same transformation on one of the abstracts from the document set. Topic profiles are stored as <term, weight> pairs, where the weight represents the relevance of that term to the profile.

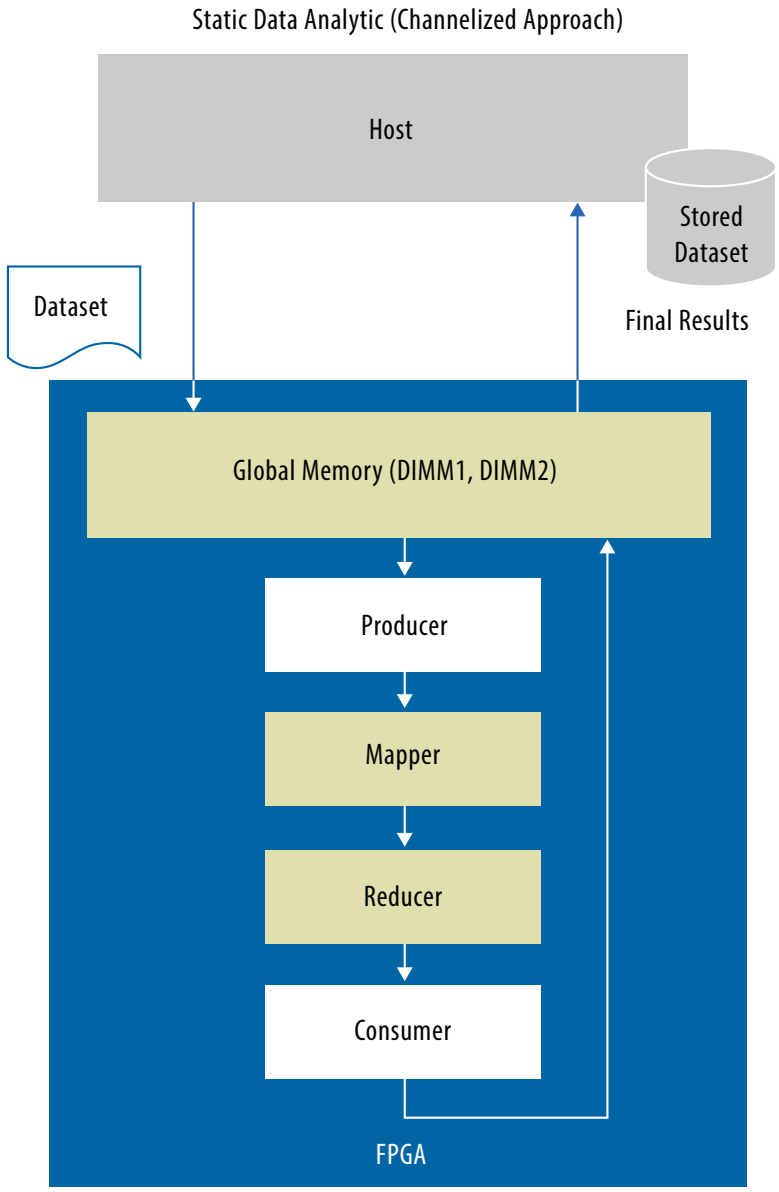
The application compares the terms in the document to the terms in the profile and computes a weighted score for each document indicating the relevance of the document to the topic profile. In order to improve memory access performance, a Bloom filter is used to efficiently determine if a document term is in the topic profile. The Bloom filter is initialized by computing hash functions on all terms in the topic profile and using the hash results to set bits in the Bloom filter. The application computes the same hash functions of the each term in the document, and compares to the Bloom filter. If the Bloom Filter returns a zero, the term is guaranteed to not be in the topic profile and therefore no memory access is required. If the Bloom filter returns a one, the weight for that term in the topic profile is retrieved. False positives are possible, which result in an extra memory access but do not affect the overall accuracy of the results.

We implemented this document filtering application using a map/reduce parallel processing approach where the mapper and reducer kernels were implemented in OpenCL and executed on the FPGA. The mapper kernel calculates the hash functions for each term in the input document dataset, compares it against the Bloom Filter and accesses the profile weights stored in global memory. The reducer kernel uses the document term frequencies and profile weights to calculate the document score. We implemented the map/reduce document filtering application as a static analytic using an NDrange approach, static analytic using a channelized approach, and streaming analytic using a channelized approach.

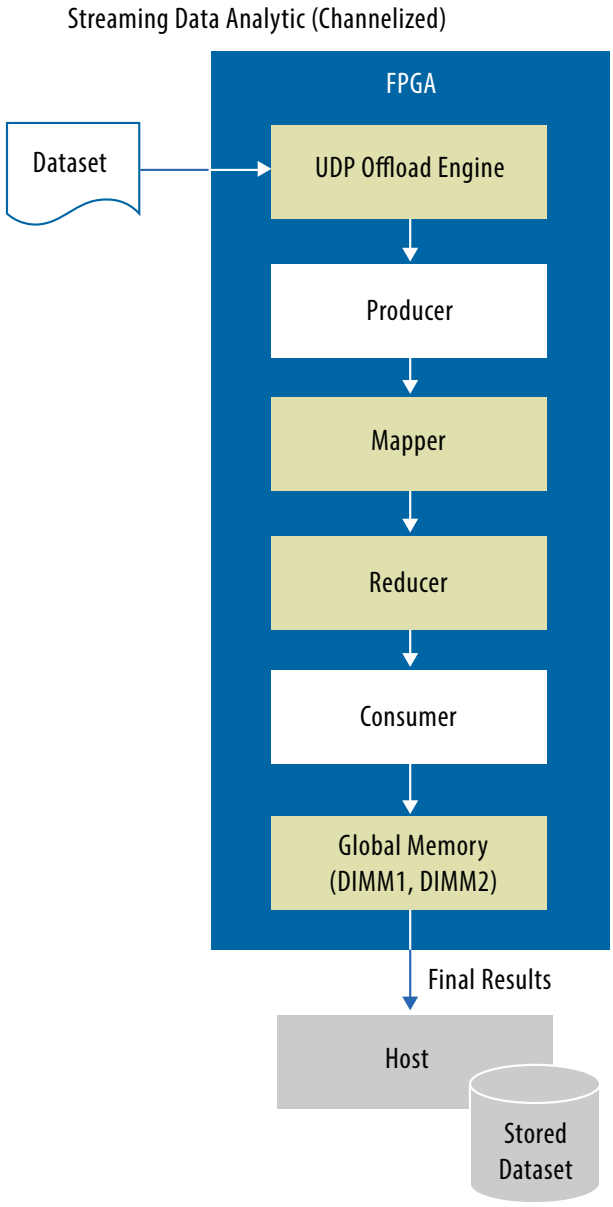
The NDRange implementation partitions the OpenCL kernel into independent and asynchronous work groups consisting of a number of work items performing the same operation in parallel. In this approach, the host loads the dataset into the device's global memory. The mapper kernel reads data from memory, processes the data and returns the intermediate results back to the host. The host then sends the intermediate results back to device global memory, so the reducer kernel can process the intermediate results. The final results processed by the reducer kernel are then written back to global memory for transfer back to the host. This is the standard approach to writing OpenCL kernels, and maps very well to the massively parallel SIMD (Single Instruction Multiple Data) compute unit architecture typical in modern GPUs. The FPGA architecture accommodates this SIMD programming approach also, but does not mandate it, allowing more flexibility in the parallel processing implementation.



The channelized approach takes advantage of Altera's channel concept to pass data directly between kernels eliminating costly host/device memory transfers. Channels are a superset to the pipe concept written into the latest OpenCL specification. Altera channels are implemented as FIFOs allowing direct data transfer between kernels, as well as between kernels and external I/O. In the channelized implementation, the code is written as a simple single-threaded application. Parallelism is exploited by the use of vector data types (data parallelism) as implemented by the developer and by unrolling loops (programming parallelism) as provided primarily by the compiler. Therefore, the channelized approach is a more straight forward implementation for the developer than the NDRange approach. Additional helper functions (producer and consumer kernels) are used for writing and reading to external memory.

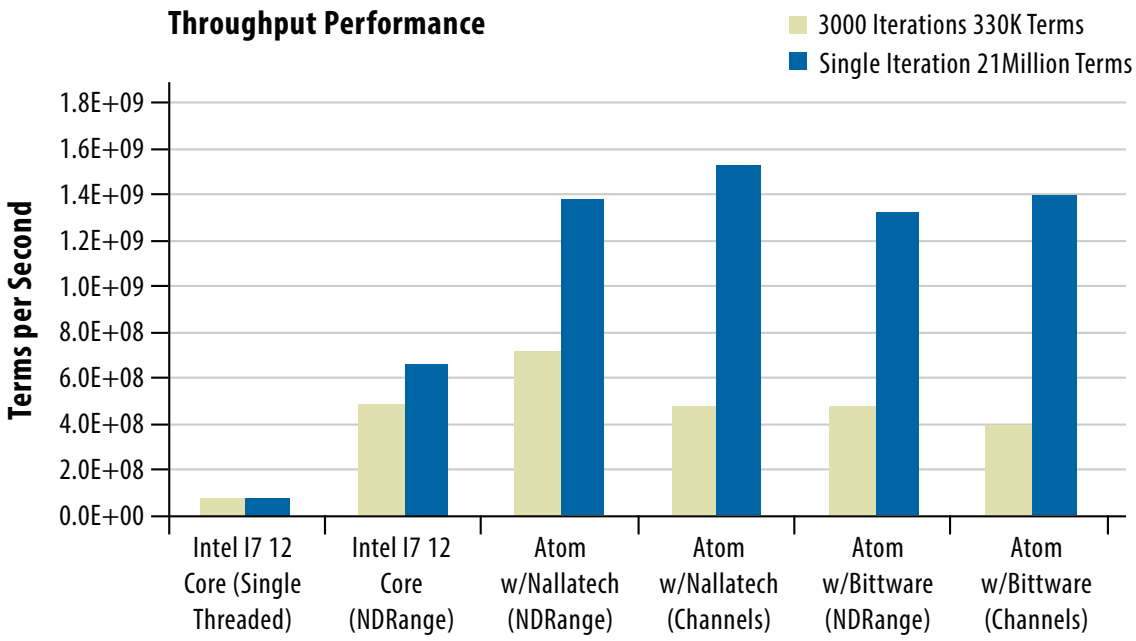


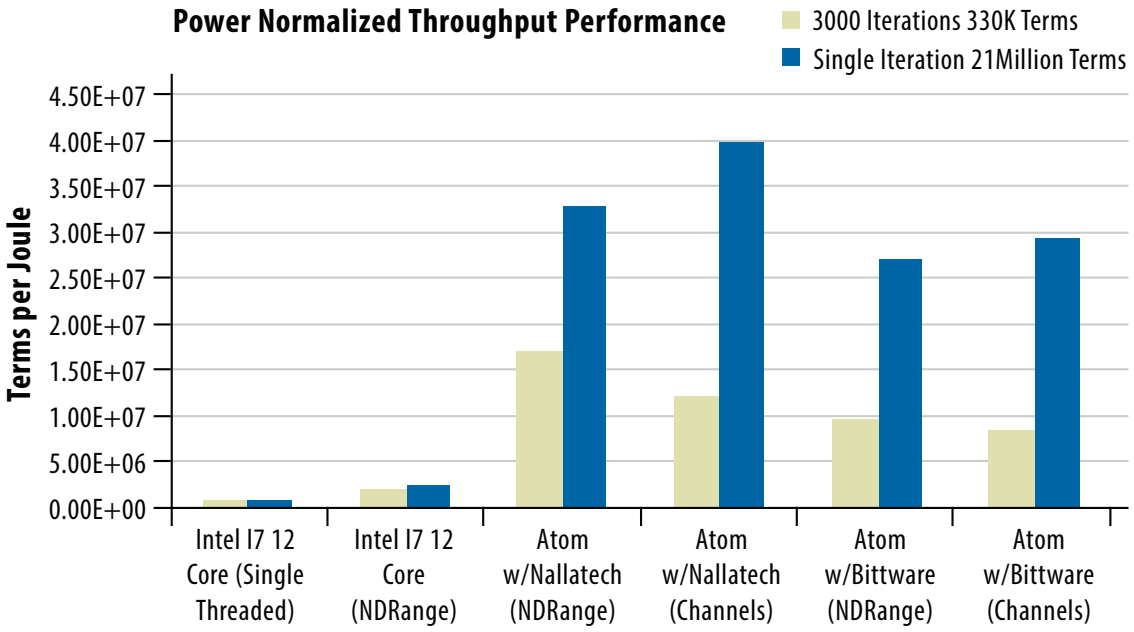
Finally, we also implemented the document filtering application as a streaming map/reduce analytic using the channelized approach to stream data directly into the FPGA. The FPGA includes a UDP offload engine that receives and processes the document data set via an external network connection providing the document <term, frequency> pairs to the rest of the processing kernels. The amount of parallelism achievable using this approach is less than that achieved in the previous approaches due to the inclusion of the UDP offload engine within the FPGA, reducing the resources available for user-defined kernels. However, this approach eliminates the collection and storage of the dataset on the host, and subsequent transfer of the data set to the device for processing. The final results are transferred back to the host for storage, which is a significant data reduction improving both data transfer time as well as required host storage capacity.



Results

We conducted performance (execution time) and power consumption benchmark testing of the static data analytic document filtering application using both the NDRange and channelized implementations. Each implementation was executed on both the Nallatech PCI385N and Bittware S5PH-Q FPGA platforms using their respective High Performance Computer (HPC) Board Support Platform (BSP). The host was a 1U micro-server with an Atom C2850 processor. We performed the benchmark testing using two separate methods. The first method conducted multiple iterations over a smaller data set re-deploying the kernel to the device at the beginning of each iteration. The second method used a much larger data set and deployed the kernel to the device only once at the beginning of the test. The performance was compared to CPU-only execution using a single-threaded implementation executed on both the 1U Atom server and an i7 3930K 12 core processor, as well as the NDRange OpenCL implementation executed on the i7 processor using AMD's OpenCL SDK.





Our research has clearly shown the benefits of implementing compute intensive algorithms using FPGAs within the OpenCL framework. Our OpenCL FPGA implementation resulted in significant performance improvements over single threaded CPU implementations as well as OpenCL implementations executing on a CPU only. The greatest performance realizations occur for very large data sets using the channelized approach, since it reduces the dependency on costly host/device memory transfers.

In addition to the raw performance gains that are achievable, the real potential for this technology is related to its overwhelming power savings. When performance is analyzed in terms of throughput per Joule, FPGAs are able to realize order-of-magnitude gains. Our research focused on implementing a document filtering application using the Map/Reduce parallel processing model. While this approach aligned well to this technology, we see no reason to conclude that the performance benefits are limited to this type of computing method. In fact most parallel algorithms would likely benefit from employing FPGAs and OpenCL.

Altera Corporation
101 Innovation Drive
San Jose, CA 95134
USA
Telephone: (408) 544 7000
www.altera.com

Altera European Headquarters
Holmers Farm Way
High Wycombe
Buckinghamshire
HP12 4XF
United Kingdom
Telephone: (44) 1 494 602 000

Altera European Trading Company Ltd.
Building 2100
Cork Airport Business Park,
Cork, Republic of Ireland
Telephone: +353 21 454 7500

Altera Japan Ltd.
Shinjuku i-Land Tower 32F
6-5-1, Nishi Shinjuku
Shinjuku-ku, Tokyo 163-1332
Japan
Telephone: (81) 3 3340 9480
www.altera.co.jp

Altera International Ltd.
Unit 11- 18, 9/F
Millennium City 1, Tower 1
388 Kwun Tong Road
Kwun Tong
Kowloon, Hong Kong
Telephone: (852) 2945 7000
www.altera.com.cn

Altera Corporation Technology Center
Plot 6, Bayan Lepas Technoplex
Medan Bayan Lepas
11900 Bayan Lepas
Penang, Malaysia
Telephone: 604 636 6100