School of Computer Science & Engineering
Shri Mata Vaishno Devi University

Katra J&K

[Anmol Rana]

its_dev@outlook.com

Using a Single Codebase Across Hardware Architectures To Accelerate Sorting in Big Data Workloads

A Parallelized 3-W Quicksort Algorithm Using the Intel® oneAPI HPCT Runs Across Multiple Architectures

Challenge

Well-known sorting functions based on the quicksort algorithm, like **qsort()** (ANSI C) and **std::sort()** (ISO/IEC 14882E), are generally inefficient due to their highly serial nature. Sorting performance on large datasets, such as in High Performance Data Analytics (HPDA) and High Performance Computing (HPC), suffers because of this.

Solution

As a developer with quite a expreience in algorithms and data structures used in system designing and kernal programming, i have solved these performance challenges using the oneAPI programming model and Intel® oneAPI toolkits(Beta). The innovative redesign of a stable parallel sort algorithm significantly speeds up sorting using a single code base running on a range of hardware architectures—CPUs, GPUs, and FPGAs. The optimized algorithms and code, now available (on request) , offers a unique solution that addresses the big data sorting challenge. The solution will speed up time to insight and time to solution for big dataset workloads.

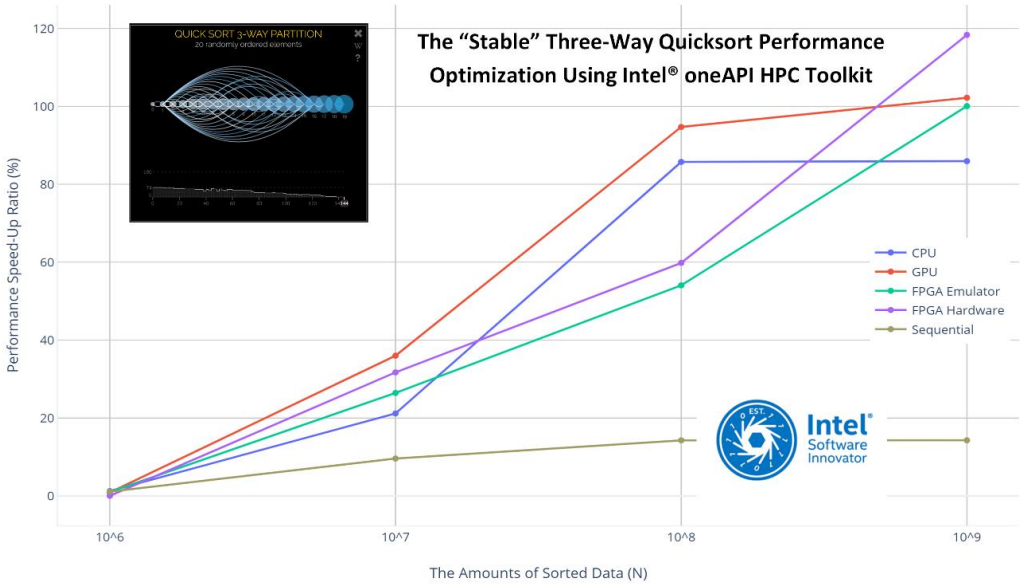

Figure 1. Developed with oneAPI and Intel® oneAPI toolkits, parallelized sorting function shows significant speedup when run across multiple architectures.[1]

History

"In modern computing , an entire class of data mining and AI machine learning algorithms becomes inefficient while processing large datasets due to their enormously high computational complexity. I focus on improving efficiency of big data mining and machine learning through algorithm-level optimizations and modern parallel code implementation of those algorithms. I use specific Intel HPC libraries and tools to improve the performance of big data processing."

Multicore, distributed computing is an ideal environment for big data processing. However, sorting operations greatly influence overall performance, and with largely serial execution **qsort(…)** and **std::sort(…)** functions typically utilize only a single core for sorting tasks. Thus, they do not scale well on today's multicore CPUs and can slow down time to insight and time to solution.

Parallelizing these functions is non-trivial. The code is overly complicated in implementation of the introspective sort and other auxiliary sorting algorithms. Additionally, these functions cannot perform "stable" sorting of large datasets that contain complex user-defined abstract datatypes (ADT). To speed up sorting of big datasets, the algorithm needed a different approach that would increase efficiency of the classical quicksort algorithm. Faster sorting can have an enormous impact on overall performance of large dataset processing in HPDA and HPC.

Developing the Solution

A new approach meant redesigning the classical three-way quicksort algorithm as parallel processes that could be run on Intel multicore CPUs.

The first step was to formulate an efficient sorting algorithm and then implement the specific code in C++ to complete the actual sorting. To parallelize the code, Intel® Parallel Studio XE is used to create a modern parallel code in C++11. The implementation of a parallel "stable" three-way quicksort function is done using OpenMP* 4.5 and Intel® Threading Building Blocks (Intel® TBB) libraries for Linux and Windows platforms.

Finally, to provide additional performance speed up potential, refining of the application for heterogeneous computing across CPUs, GPUs, and FPGAs is done using the oneAPI programming model. The code delegates particular sorting workloads to various hardware acceleration targets by:

- Implementing a specific code in Data Parallel C++ using the Intel oneAPI HPC toolkit and DPC++ standard library
- Using Parallel STL containers such as **std::remove_if(…)** with DPSTD execution context to offload execution to the acceleration targets
- Using Unified Shared Memory (USM) to allocate buffers in shared memory

Performance evaluation of the complete ready-to-use solution is done using the Intel® DevCloud.

Table 1. Speedup of parallelized quicksort function using oneAPI and Intel® oneAPI toolkits[1]

Hardware	**Speedup**
Intel® Xeon® Gold processor 6128 @ 3.47 GHZ / 192 GB RAM	2X to 11X
Intel® FPGA Emulator Application	4X to 6X
Intel® Gen9 Graphics NEO	4X to 7X
Intel® PAC Platform / Intel® ARRIA 10 GX	3X to 8X

" Intel HPC technologies are choosen because they enables one to achieve specific performance optimization goals while delivering a complete project solution , Working with Intel offers an opportunity to learn about heterogeneous computing using the oneAPI programming model and to use many HPC libraries and development tools, including the Intel oneAPI toolkits."

Parallel code implementation can be used as a part of any project to improve performance of an entire solution.

As a part of his project, Mr. Anmol developed an application for Windows and Linux platforms that demonstrates his parallel "stable" 3-W function running in a heterogeneous hardware environment. The application compares his algorithm's performance to the performance of the serial **qsort(…)** and **std::sort(…)** functions.

Using the oneAPI programming model and Intel oneAPI HPC toolkit, my implementation speeds up sorting and runs across multiple computing architectures, including CPUs, GPUs, and FPGAs.

Enabling Technologies

Following tools are being used to help develop the solution.

Intel Development Tools:

- Intel® Parallel Studio XE Cluster Edition
- Intel® oneAPI Threading Building Blocks (oneTBB)
- Intel® MPI Library
- Intel® oneAPI Base Toolkit 2021.1.beta07.1506
- Intel® oneAPI HPC Toolkit
- Intel® Parallel Studio
- Intel® VTune™ Profiler(Beta)

Intel Hardware:

- Intel® Core™ processors
- Intel® Xeon® processors
- Intel® FPGA
- Intel® UHD Graphics 9Gen NEO
- Intel® Programmable Acceleration Card with Intel® ARRIA® 10 GX FPGA
- Intel® DevCloud

Recommendations and Resources

If you are starting out, here are some recommendations.

- Sorting is a critical function in many applications. It deserves exploration for further improvements in performance and efficiency.
- Initially, developers should concentrate on the formulation of an efficient algorithm that solves a specific problem before attempting to create a modern parallel code that does the actual data processing.

Resources

- Sedgewick, Robert, with Kevin Wayne. Algorithms, Addison-Wesley Professional; 4th Edition (March 19, 2011)
- OpenMP Books: https://www.openmp.org/resources/openmp-books
- Intel® Threading Building Blocks Documentation
- Intel oneAPI HPC Toolkit Documentation

Technical Articles and Tutorials

- A Parallel Stable Sort Using C++11 for TBB, Cilk Plus, and OpenMP
- An Efficient Parallel Three-Way Quicksort Using Intel C++ Compiler And OpenMP 4.5 Library
- How To Implement A Parallel "Stable" Three-Way Quicksort Using Intel C++ Compiler and OpenMP 4.5 library
- How To Optimize A Parallel Stable Sort Performance Using The Revolutionary Intel® oneAPI HPC Toolkit
- How To Implement The Parallel "Stable" Sort Using Intel® MPI Library And Deploy It To A Multi-Node Computational Cluster

- oneAPI Single Programming Model to Deliver Cross-Architecture Performance

[1] Performance testing done by Mr. Anmol