Compare Benefits of CPUs, GPUs, and FPGAs for Different oneAPI Compute Workloads

ID 757681

Updated 11/9/2022

Version Latest

Public

Article at a Glance

- oneAPI is an open, unified programming model designed to simplify development and deployment of data-centric workloads across central processing units (CPUs), graphics processing units (GPUs), field-programmable gate arrays (FPGAs), and other accelerators.
- In a heterogeneous compute environment, developers need to understand the capabilities and limitations of each compute architecture to effectively match the appropriate workload to each compute device.

This article:

- Compares the architectural differences between CPUs, GPUs, and FPGAs
- Shows how SYCL* constructs are mapped to each architecture
- Examines library support differences
- Discusses characteristics of applications best suited for each architecture

Compute Architecture Comparison of CPUs, GPUs, and FPGAs

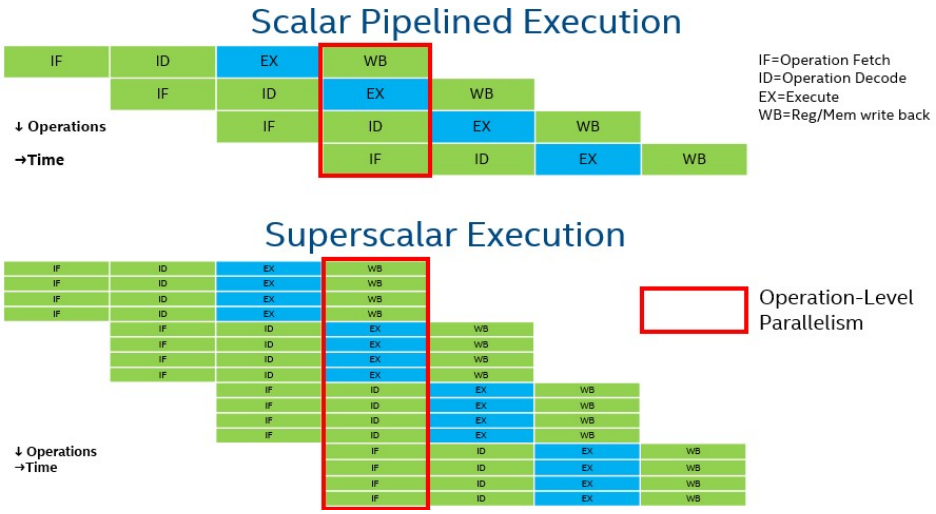
CPU Architecture

Among the compute architectures discussed here, CPUs, with over a half-century of history, are the most well-known and ubiquitous. CPU architecture is sometimes referred to as scalar architecture because it is designed to process serial instructions efficiently. CPUs, through many techniques available, are optimized to increase instruction-level parallelism (ILP) so that serial programs can be executed as fast possible.

A scalar pipelined CPU core can execute instructions, divided into stages, at the rate of up to one instruction per clock cycle (IPC) when there are no dependencies.

To boost performance, modern CPU cores are multi-thread, superscalar processors with sophisticated mechanisms that are used to find instruction-level parallelism and execute multiple out-of-order instructions per clock cycle. They fetch many instructions at once, find the dependency-graph of those instructions, utilize sophisticated branch-prediction mechanisms, and execute those instructions in parallel (typically at 10x the performance of scalar processors in terms of IPC).

The diagram below shows the simplified execution of a scalar-pipelined CPU and a superscalar CPU.



Advantages

There are many advantages to using a CPU for compute compared to offloading to a coprocessor, such as a GPU or an FPGA.

Why?

Because data does not need to be offloaded. Latency is reduced with minimal data transfer overhead.

Since high-frequency CPUs are tuned to optimize scalar execution and most software algorithms are serial in nature, high performance can easily be achieved on modern CPUs.

For parts of the algorithm that can be vector-parallelized, modern CPUs support single instruction, multiple-data (SIMD) instructions like the Intel® Advanced Vector Extensions 512 and Intel® Advanced Matrix Extensions. As a result, CPUs are suited to a wide variety of workloads. Even for massively parallel workloads, CPUs can outperform accelerators for algorithms with high branch divergence or high instruction-level parallelism, especially where the data size to compute ratio is high.

CPU advantages:

- Out-of-order superscalar execution
- Sophisticated control to extract tremendous instruction level-parallelism
- Accurate branch prediction
- Automatic parallelism on sequential code
- Large number of supported instructions
- Lower latency when compared to offload acceleration
- Sequential code execution results in ease-of-development

GPU Architecture

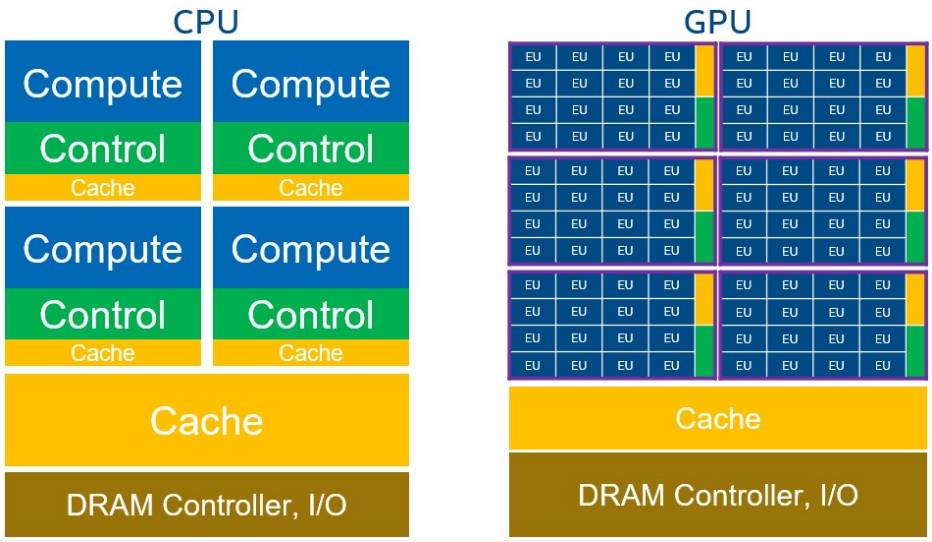
GPUs are processors made of massively parallel, smaller, and more specialized cores than those generally found in high-performance CPUs. GPU architecture:

- Is optimized for aggregate throughput across all cores, deemphasizing individual thread latency and performance.
- Efficiently processes vector data (an array of numbers) and is often referred to as vector architecture.
- Dedicates more silicon space to compute and less to cache and control.

As a result, GPU hardware explores less instruction-level parallelism and relies on software-given parallelism to achieve performance and efficiency.

GPUs are in-order processors and do not support sophisticated branch prediction. Instead, they have a plethora of arithmetic logic units (ALUs) and deep pipelines. Performance is achieved through multithreaded execution of large and independent data, which amortizes the cost of simpler control and smaller caches.

Finally, GPUs employ a single instruction, multiple threads (SIMT) execution model where multithreading and SIMD are leveraged together. In the SIMT model, multiple threads (work-items or a sequence of SIMD lane operations) are processed in lockstep in the same SIMD instruction stream. Multiple SIMD instruction streams are mapped to a single execution unit (EU) or vector engine where the GPU can context-switch among those SIMD instruction streams when one stream is stalled.



The above diagram shows the difference between the CPU and GPU.

EUs or vector engines are the basic unit of processing on a GPU. Each EU can process multiple SIMD instruction streams. In the same silicon space, GPUs have more compute logic than CPUs. GPUs are organized hierarchically. Multiple EUs or vector engines combine to form a compute unit with shared local memory and synchronization mechanisms (aka Xe-core, sub-slice or streaming multiprocessor, outlined in purple). The compute units combine to form the GPU.

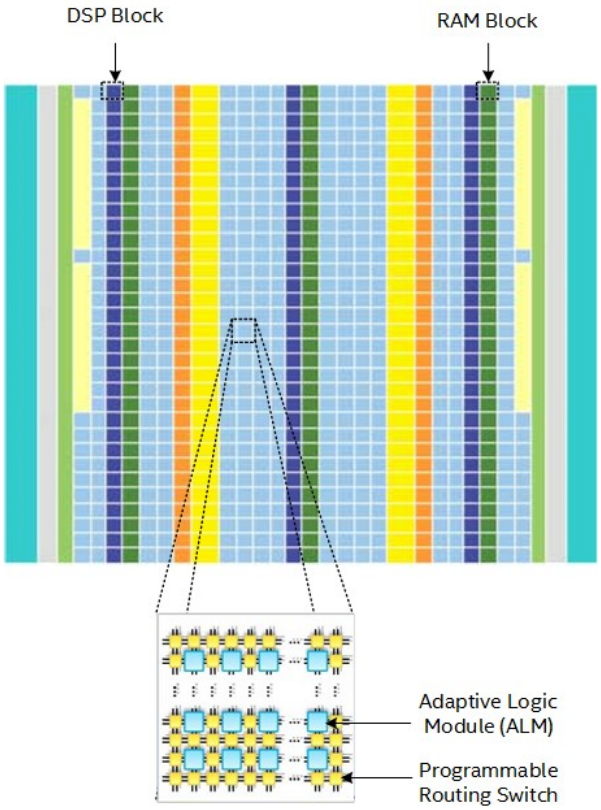
GPU Advantages:

- Massively parallel, up to thousands of small and efficient SIMD cores/EUs
- Efficient execution of data-parallel code
- High dynamic random-access memory (DRAM) bandwidth

FPGA Architecture

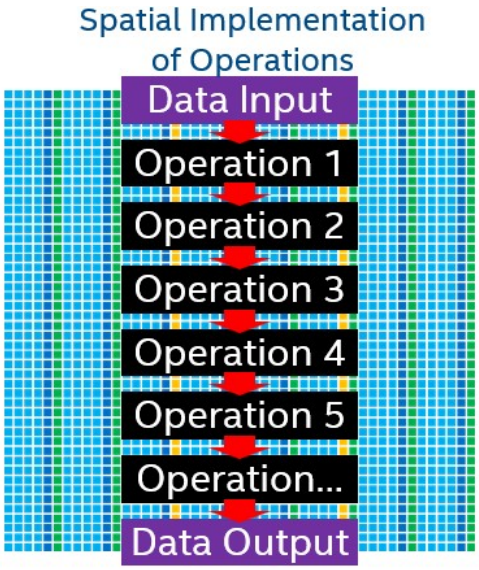
Unlike CPUs and GPUs, which are software-programmable fixed architectures, FPGAs are reconfigurable, and their compute engines are defined by the user. When writing software targeting an FPGA, compiled instructions become hardware components that are laid out on the FPGA fabric in space, and those components can all execute in parallel. Because of this, FPGA architecture is sometimes referred to as a spatial architecture.

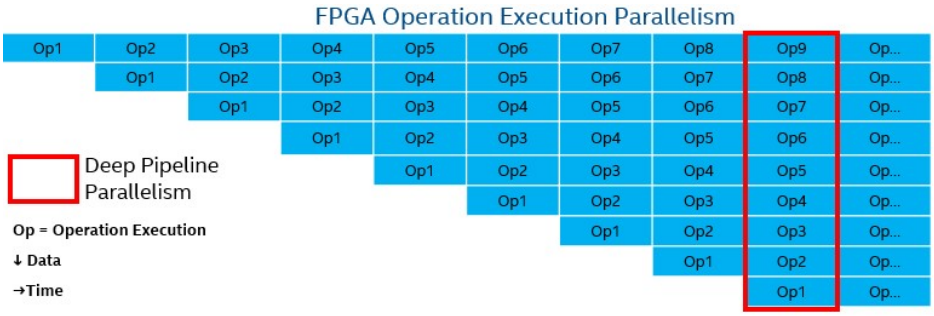
An FPGA is a massive array of small processing units consisting of up to millions of programmable 1-bit Adaptive Logic Modules (each can function like a one-bit ALU), up to tens of thousands of configurable memory blocks, and tens of thousands of math engines, known as digital signal processing (DSP) blocks, that support variable precision floating-point and fixed-point operations. All these resources are connected by a mesh of programmable wires that can be activated on an as-needed basis.



When software is “executed” on the FPGA, it is not executing in the same sense that compiled and assembled instructions execute on CPUs and GPUs. Instead, data flows through customized deep pipelines on the FPGA that match the operations expressed in the software. Because the dataflow pipeline hardware matches the software, control overhead is eliminated, which results in improved performance and efficiency.

With CPUs and GPUs, instruction stages are pipelined, and new instructions start executing every clock cycle. With FPGAs, operations are pipelined so new instruction streams operating on different data start executing every clock cycle.





While pipeline parallelism is the primary form of parallelism for FPGAs, it can be combined with other types of parallelism. For example, data parallelism (SIMD), task parallelism (multiple pipelines), and superscalar execution (multiple independent instructions executing in parallel) can be utilized with pipeline parallelism to achieve maximum performance.

FPGA Advantages:

- Efficiency: Data processing pipeline tuned exactly to the needs of software. No need for control units, instruction fetch units, register writeback, and other execution overhead.
- Custom Instructions: Instructions not natively supported by CPUs/GPUs can be easily implemented and efficiently executed on FPGAs (e.g., bit manipulations).
- Data Dependencies across Parallel Work can be resolved without stalls to the pipeline.
- Flexibility: FPGAs can be reconfigured to accommodate different functions and data types, including non-standard data types.
- Custom On-Chip Memory Topology Tuned to Algorithm: Large-bandwidth, on-chip memory built to accommodate access pattern, thereby minimizing or eliminating stalls.
- Rich I/O: FPGA core can interact directly with various network, memory, and custom interfaces and protocols resulting in low and deterministic latency solutions.

Mapping of oneAPI and SYCL* to CPUs, GPUs, and FPGAs

Now that the basics of each architecture have been described, this section examines how oneAPI and SYCL execution map to the execution units of CPUs, GPUs, and FPGAs.

To learn more about SYCL, see the references at the end of this article.

NDRange Kernel

```
auto total = range{N};
auto wg_size = range{256};

queue{}.submit([&](handler& h) {
   accessor out{input_buf, h};
   accessor in{output_buf, h};

   h.parallel_for(nd_range{total, wg_size},
                              [=](auto i) {
      // kernel
      out[i] = process(in[i]);
   });
});
```

Singel Task Kernel

```
queue{}.submit([&](hanc
   accessor out{input_b
   accessor in{output_b
   h.single_task( [=] {
      // kernel
      for (int i=0; i<N
         out = process(
      }
   });
});
```

Above are two types of SYCL kernels: NDRange and single task. This section examines how these two kernels and their parallelism are executed on the different architectures.

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.