Exploring the GPU Architecture

VMware AI/ML

Table of contents

Exploring the GPU Architecture 3
Overview 3
Latency vs Throughput 4
Differences and Similarities 5
Exploring the GPU Architecture 6
To Conclude 8

Exploring the GPU Architecture

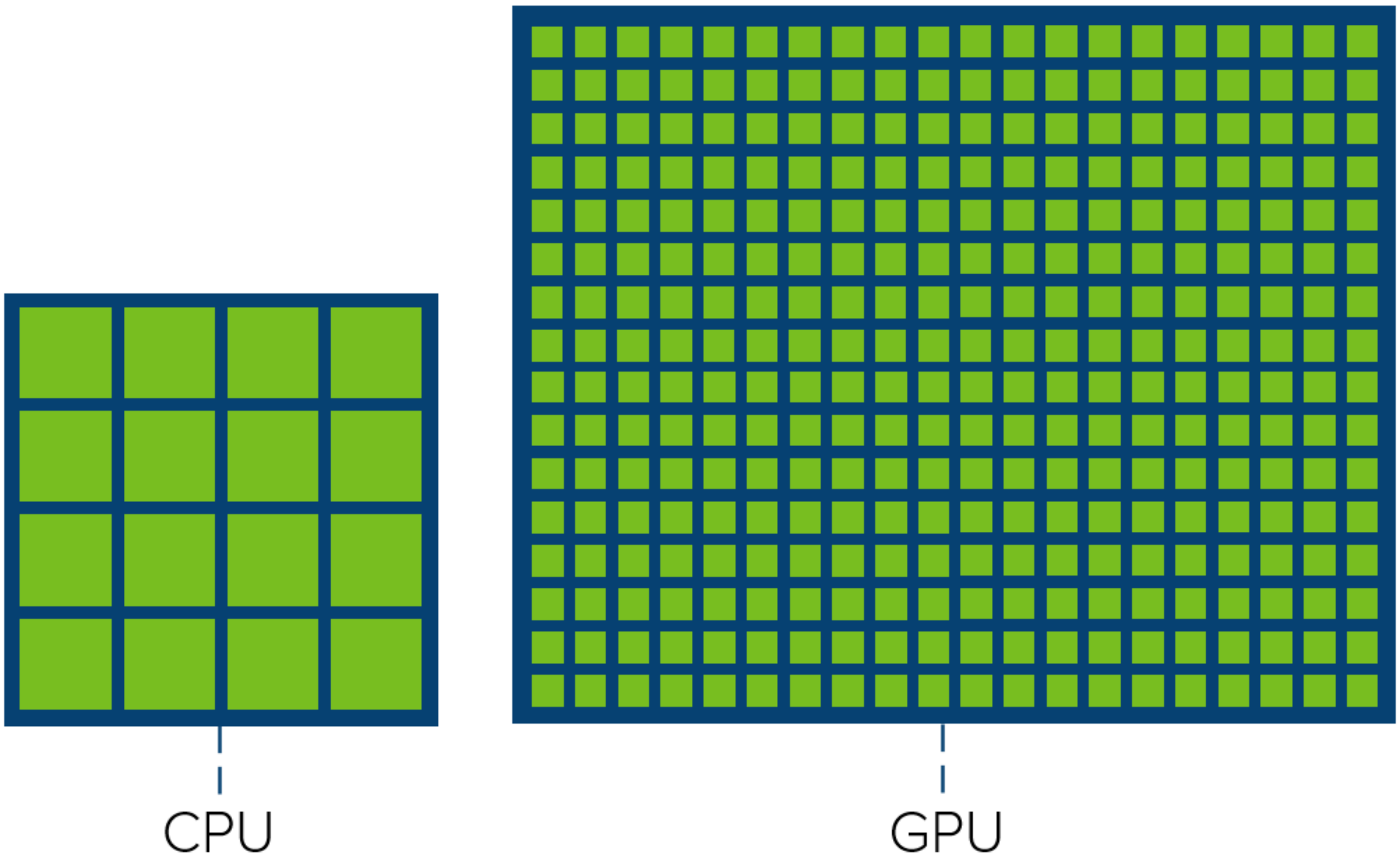
Overview

A Graphics Processor Unit (GPU) is mostly known for the hardware device used when running applications that weigh heavy on graphics, i.e. 3D modeling software or VDI infrastructures. In the consumer market, a GPU is mostly used to accelerate gaming graphics. Today, GPGPU's (General Purpose GPU) are the choice of hardware to accelerate computational workloads in modern High Performance Computing (HPC) landscapes.

HPC in itself is the platform serving workloads like Machine Learning (ML), Deep Learning (DL), and Artificial Intelligence (AI). Using a GPGPU is not only about ML computations that require image recognition anymore. Calculations on tabular data is also a common exercise in i.e. healthcare, insurance and financial industry verticals. But why do we need a GPU for these types of all these workloads? This blogpost will go into the GPU architecture and why they are a good fit for HPC workloads running on vSphere ESXi.

Latency vs Throughput

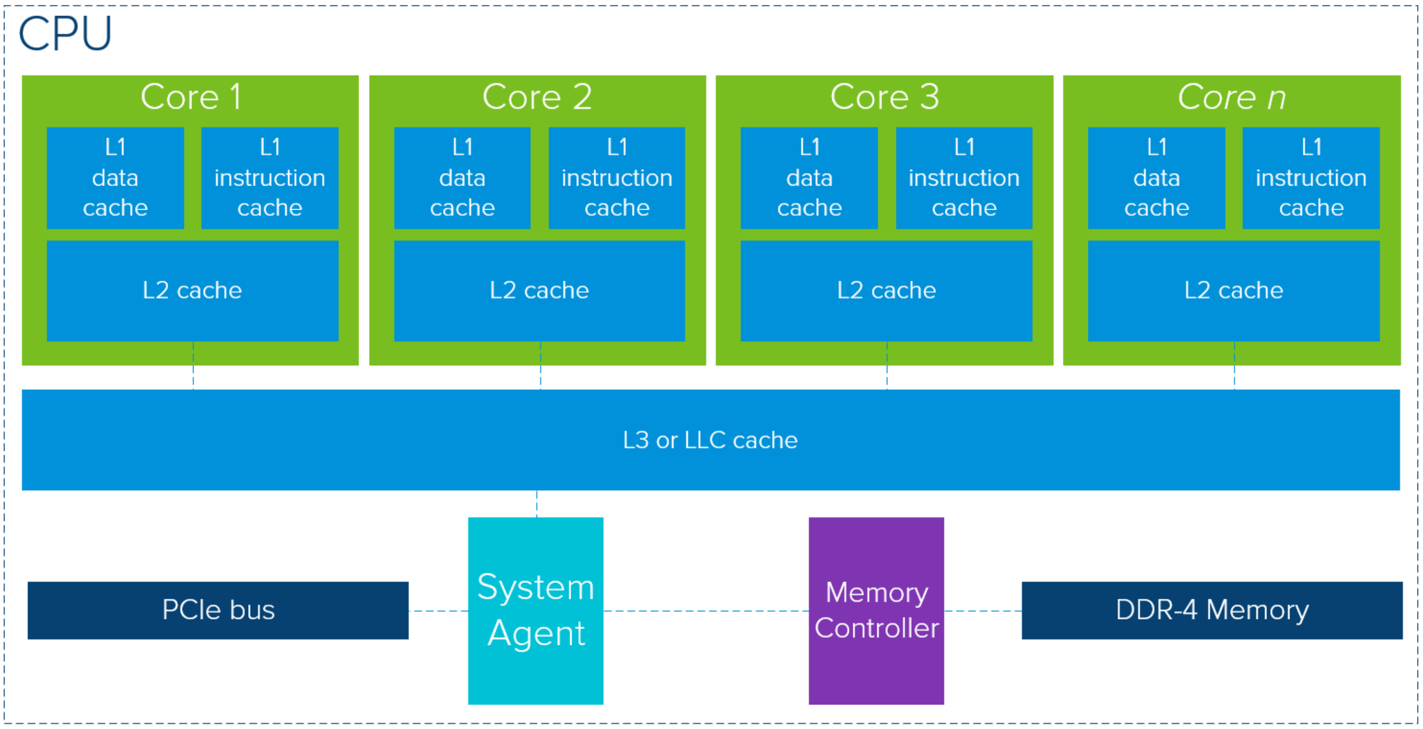
Let’s first take a look at the main differences between a Central Processing Unit (CPU) and a GPU. A common CPU is optimized to be as quick as possible to finish a task at a as low as possible latency, while keeping the ability to quickly switch between operations. It’s nature is all about processing tasks in a serialized way. A GPU is all about throughput optimization, allowing to push as many as possible tasks through is internals at once. It does so by being able to parallel process a task. The following exemplary diagram shows the ‘core’ count of a CPU and GPU. It emphasizes that the main contrast between both is that a GPU has a lot more cores to process a task.



Differences and Similarities

However, it is not only about the number of cores. And when we speak of cores in a NVIDIA GPU, we refer to CUDA cores that consists of ALU's (Arithmetic Logic Unit). Terminology may vary between vendors.

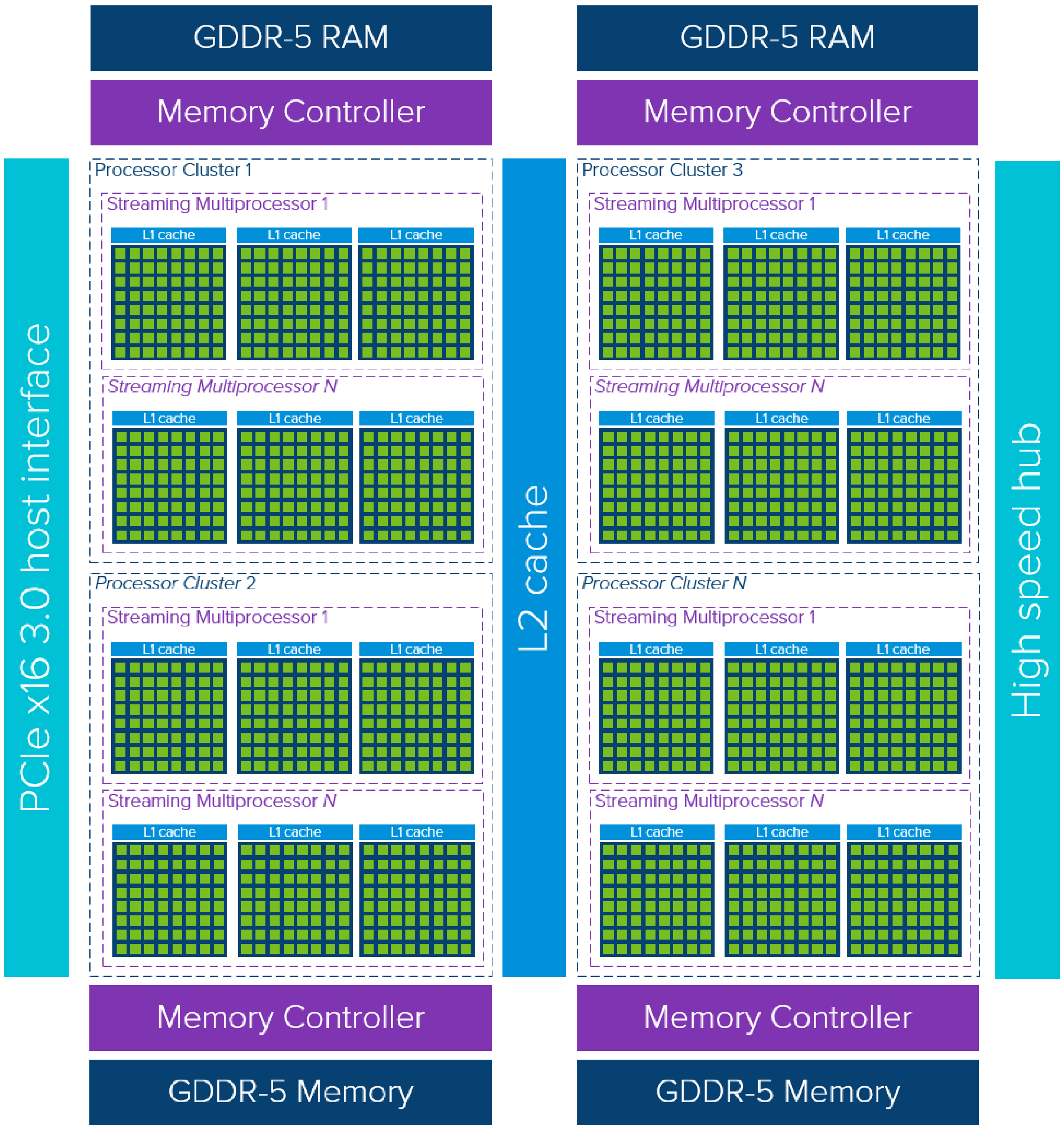
Looking at the overall architecture of a CPU and GPU, we can see a lot of similarities between the two. Both use the memory constructs of cache layers, memory controller and global memory. A high-level overview of modern CPU architectures indicates it is all about low latency memory access by using significant cache memory layers. Let's first take a look at a diagram that shows an generic, memory focussed, modern CPU package (**note**: the precise lay-out strongly depends on vendor/model).



A single CPU package consists of cores that contains separate data and instruction layer-1 caches, supported by the layer-2 cache. The layer-3 cache, or last level cache, is shared across multiple cores. If data is not residing in the cache layers, it will fetch the data from the global DDR-4 memory. The numbers of cores per CPU can go up to 28 or 32 that run up to 2.5 GHz or 3.8 GHz with Turbo mode, depending on make and model. Caches sizes range up to 2MB L2 cache per core.

Exploring the GPU Architecture

If we inspect the high-level architecture overview of a GPU (again, strongly depended on make/model), it looks like the nature of a GPU is all about putting available cores to work and it's less focussed on low latency cache memory access.



A single GPU device consists of multiple Processor Clusters (PC) that contain multiple Streaming Multiprocessors (SM). Each SM accommodates a layer-1 instruction cache layer with its associated cores. Typically, one SM uses a dedicated layer-1 cache and a shared layer-2 cache before pulling data from global GDDR-5 (or GDDR-6 in newer GPU models) memory. Its architecture is tolerant of memory latency.

Compared to a CPU, a GPU works with fewer, and relatively small, memory cache layers. Reason being is that a GPU has more transistors dedicated to computation meaning it cares less how long it takes the retrieve data from memory. The potential memory access ‘latency’ is masked as long as the GPU has enough computations at hand, keeping it busy.

A GPU is optimized for data parallel throughput computations.

Looking at the numbers of cores it quickly shows you the possibilities on parallelism that is it is capable of. When examining the 2019 NVIDIA flagship offering, the Tesla V100, one device contains 80 SM’s, each containing 64 cores making a total of 5120 cores! Tasks aren’t scheduled to individual cores, but to processor clusters and SM’s. That’s how it’s able to process in parallel. Now combine this powerful hardware device with a programming framework so applications can fully utilize the computing power of a GPU.

To Conclude

> High Performance Computing (HPC) is the use of parallel processing for running advanced application programs efficiently, reliably and quickly.

This is exactly why GPU's are a perfect fit for HPC workloads. Workloads can greatly benefit from using GPU's as it enables them to have massive increases in throughput. A HPC platform using GPU's will become much more versatile, flexible and efficient when running it on top of the VMware vSphere ESXi hypervisor. It allows for GPU-based workloads to allocate GPU resources in a very flexible and dynamic way.