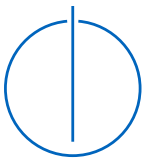


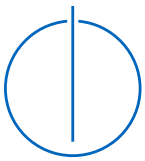
DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Bachelor's Thesis in Informatics: Game Engineering

Thesis title

Ruilin Qi



DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Bachelor's Thesis in Informatics: Game Engineering

Thesis title

Titel der Abschlussarbeit

Author:	Ruilin Qi
Supervisor:	Supervisor
Advisor:	Stepan Vanecek
Submission Date:	15th March 2022

I confirm that this bachelor's thesis in informatics: game engineering is my own work and I have documented all sources and material used.

Munich, 15th March 2022 Ruilin Qi

Acknowledgments

Abstract

Contents

Acknowledgments **iii**

Abstract **iv**

1 Introduction **1**
1.1 Background and Motivation 1
1.2 Goals and Aims 1
1.3 Delimitation 1
1.4 Structure and Approach 1

2 Background **2**
2.1 HPC 2
2.2 PCI-Express 2
2.2.1 Key Features 2
2.2.2 Functionality 3
2.2.3 Topology and Communication 5
2.2.4 Revisions and Further Specifications 7
2.3 Graphics Processing Units 8
2.3.1 What are GPUs 8
2.3.2 Uses of GPUs 8
2.3.3 GPU Memory 8
2.4 CUDA 9
2.4.1 Kernels and Scalability 9
2.4.2 Memory Management 9

3 Bandwidth Benchmark **10**
3.1 Concept 10
3.2 Implementation 10
3.3 Results 10
3.4 Discussion 10
3.4.1 successes 10
3.4.2 shortcomings 10

4 NVML Counters **11**
4.1 Concept . . . 11
4.2 Implementation . . . 11
4.3 Results . . . 11
4.4 Discussion . . . 11
4.4.1 successes . . . 11
4.4.2 shortcomings . . . 11

5 Link Saturation **12**
5.1 Concept . . . 12
5.2 Implementation . . . 12
5.3 Results . . . 12
5.4 Discussion . . . 12
5.4.1 successes . . . 12
5.4.2 shortcomings . . . 12

6 Summary **13**

List of Figures **14**

List of Tables **15**

Bibliography **16**

1 Introduction

1.1 Background and Motivation

1.2 Goals and Aims

- develop lightweight tool that has few requirements to gain insight to data movements in a PCIe link

1.3 Delimitation

- Heterogeneous systems have many different interconnects - Focus on PCIE-interface in this thesis - (why)? not sure yet, figure something out

1.4 Structure and Approach

- three approaches / benchmarks - bandwidth: for measuring the raw bandwidth capacity of the system - nvml: for measuring pcie link activity - copy: for measuring pcie link activity

2 Background

2.1 HPC

High-performance computing, abbreviated as HPC, leverages the compute capacity of supercomputers or computer clusters to solve problems that are highly complex in nature. [1] A computer cluster consists of many different computers (nodes) that are interconnected with high-speed, low-latency interconnects. Each node contains the same primary components a desktop or laptop PC would contain, such as a CPU, RAM, and storage. [2] It should be noted that modern CPUs, especially ones utilized in HPC applications, usually contain multiple physical cores and multiple threads to enable some amount of parallel processing. [example maybe?] Some nodes, just like some PCs, also have a dedicated GPU to accelerate certain types of workloads. [2]

2.2 PCI-Express

PCIe, or PCI-Express, shorthand for Peripheral Component Interconnect Express, is a "general-purpose serial I/O interconnect". [3] PCIe, as an interface, allows the CPU to connect with, as the name suggests, peripherals and components. [4] Common components and peripherals include, but are not limited to: Graphics cards, sound cards, video capture cards, WiFi cards, and storage. PCIe is designed to replace the ageing PCI (Peripheral Component Interconnect), PCI-X (Peripheral Component Interconnect Extended), and AGP (Accelerated Graphics Port) standards. [5] These standards are developed, defined, and maintained by the PCI-SIG group, which is a nonprofit organization with 800+ member companies based in Beaverton, Oregon. [6] This chapter will briefly introduce the key features and functionality of PCI-Express.

2.2.1 Key Features

PCI-Express is, at its core, a serialized, point-to-point connection that is designed to be processor agnostic, scalable, and backwards compatible with PCI.[3], [7], [8] PCI-Express utilizes a dual-simplex connection to facilitate sending and receiving information concurrently. Additionally, to ensure backwards compatibility with PCI, PCI-Express shares the same memory configuration as PCI, which will be elaborated

Link Width	x1	x2	x4	x8	x12	x16	x32
Gen1 Bandwidth (GB /s)	0.5	1	2	4	6	8	16
Gen2 Bandwidth (GB/s)	1	2	4	8	12	16	32
Gen3 Bandwidth (GB/s)	2	4	8	16	24	32	64

Table 2.1: PCI Express aggregate bandwidths by generation and link width [9]

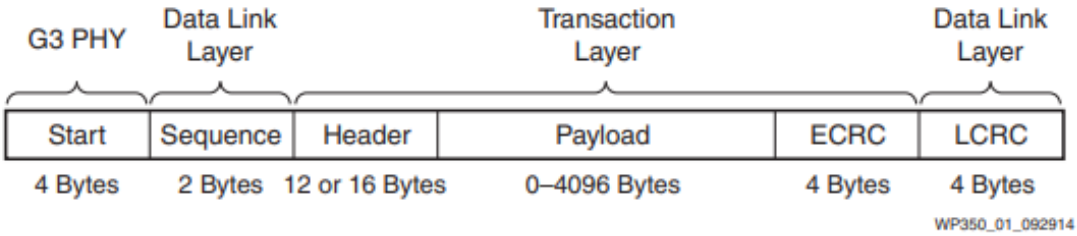


Figure 2.1: An example of a PCI-Express packet [7]

further upon in section 2.2.3. Further key features include better error handling and data integrity capabilities. [5] To future-proof the standard, current and future generations of PCIe are to be designed to be compatible with current PCIe standards. [8] So far, each generation of PCIe doubled the previous generation's theoretical maximum bandwidth, as seen in Table 2.1.

2.2.2 Functionality

packet

PCIe, similar to IPv4 or IPv6, utilizes packets to communicate between the host - the CPU - and the device. As shown in figure 2.1, the packet consists of a few different elements, which will be further expanded upon below.

- Start: this is the start component which signals the begin of a packet to the physical layer.
- Sequence: This two-byte sequence is used by the Data Link Layer to determine the sequence of the packets.

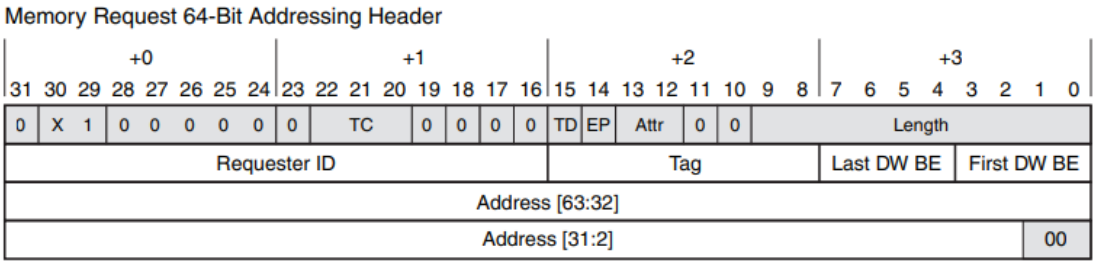


Figure 2.2: An example of a memory request header [7]

- Header: The 12 to 16 Byte header will be discussed in further detail in subsection 2.2.2. This component belongs to the Transaction layer.
- Payload: The PCIe payload. This is optional, however any memory transferred via memory copy operations will have the memory as payload. This also is a part of the Transaction layer.
- ECRC: a CRC code for error-checking purposes used by the transaction layer.
- LCRC: a CRC code for error-checking purposes used by the Data Link Layer.

[question: do i need a source for each of the bullet points? additionally, the source doesn't outright state that 'this is what this is for', in a degree, but it is inferred knowledge. what should I do about this?]

header

As with IPv4 or IPv6, PCI-Express uses headers to determine the purpose and target of each TLP (Transaction Layer Packet) However, instead of using IP-addresses, stored in the header, to determine the sender and the receiver, PCIe uses the Requester ID to determine the sender. The Address determines the receiver of the intended packet, as the device memory is memory-mapped into the host address domain to enable the processor's native load or store instructions to work with PCIe devices. [10] As seen in Figure 2.2, the header has a fixed format, similar to an IPv4 or v6 header. The fields and their uses are briefly explained below.

- TC: Traffic Class: this denotes the priority of the packet. A larger value represents a higher priority. [9]
- TD: The TLP Digest field. If TD is set to 1, it indicates that there is additional CRC data in the TLP data. [cite xillybus]

- Length: more or less self-explanatory: length denotes the length of the payload in Double Words. [cite xillybus]
- Requester ID: self-explanatory: the ID of the device that requested or sent the packet. [cite xillybus]
- Tag: The Tag field has the function of a tracking number, as for read requests, the device must copy this value to its response. All outstanding tags must be unique to ensure data integrity. [cite xillybus]
- DW BE fields: DW BE stands for Double-Word Byte Enable. This denotes which of the bytes in the first / last DWs are valid. [cite xillybus]
- Address: self-explanatory: The Address to which this packet is addressed, as explained above. Additionally, for read and write requests, this denotes the starting address of the read or write. [cite xillybus]
- The EP and Attr fields are not further elaborated upon as they are rarely used by PCIe endpoint devices. [cite xillybus]

[todo: update and verify with book]

2.2.3 Topology and Communication

Topology

There are four significant components to be mentioned when discussing the topology of a PCI-Express based system. PCIe endpoints, switches, bridges, and a root complex. The communication between CPU cores and memory controllers to the PCIe endpoint is handled by the PCIe root complex. This communication can be routed through (but does not require) PCIe switches. PCIe switches allow for cascading connections, however do not benefit the total bandwidth, which is limited by the PCIe root complex in a CPU. [11] Bridges are used to connect legacy PCI and PCI-X devices with the PCIe root complex. [3] Figure 2.3 shows an example PCIe configuration of an Intel-based processor.

Memory Management

The PCIe memory structure is, due to compatibility reasons, the same as the memory structure found in the older PCI standard. This divides the device memory into three major parts for addressing:[9]

- Configuration

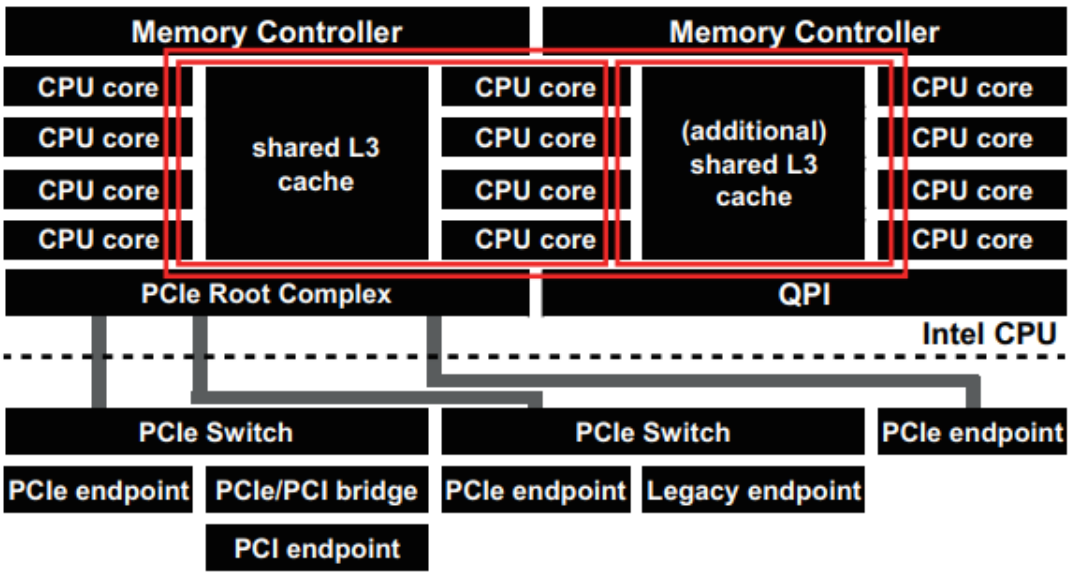


Figure 2.3: PCIe configuration on an Intel-based system[11]

- Memory
- IO

The configuration address space enables software to both identify and correctly configure the device, and is defined by its physical bus and device number.[10]

The memory address space is where the storage and registers of a PCIe device is mapped.[9] This memory space is also memory-mapped to the host address domain for ease of access by the CPU.[10]

The IO address space is a place dedicated to accessing the internal registers / storage of a PCIe or PCI device. However, this is mostly deprecated in PCIe as the internal registers and storage of said devices are simply mapped into the memory address space instead. It is now common practice to map the same set of registers in both memory and IO address space for backwards compatibility purposes. The PCIe specification discourages use of the IO space, which indicates that it remains solely for legacy support purposes.[9]

Links and Lanes

A connection between the two PCIe devices is called a link, which is made up of lanes. [9] A PCIe device, in this case, can be the CPU's PCIe root complex, bridges,

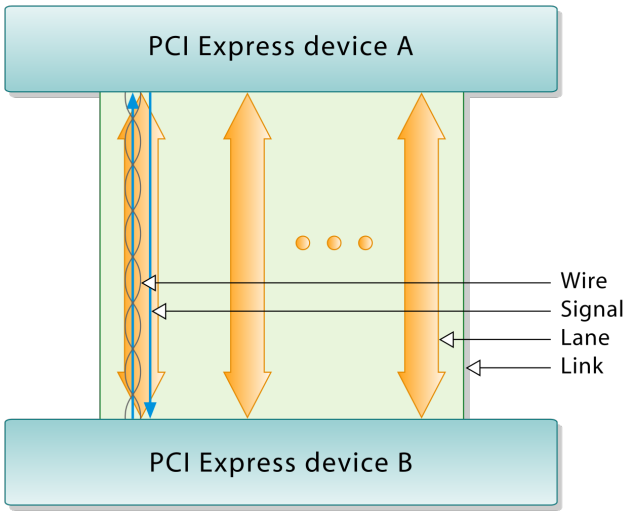


Figure 2.4: An example PCIe link between two devices [12]

switches, or a PCIe end point. A lane, on the hardware level, is a set of four copper wires, two for each signal direction.[9] Due to the scalability of PCIe, the amount of lanes in a link is variable, from 1 up to 32, and is represented by a x in front of the lane width, e.g. PCIe x16, which indicates that the PCIe link has 16 lanes. A wider link means higher bandwidths and transmit capabilities, however it also means higher power consumption, space, and cost. [9] Figure 2.4 illustrates an example PCIe link with several lanes.

2.2.4 Revisions and Further Specifications

PCI-Express was first introduced in 2003, and has received a new revision once every three to four years on average. Whilst most current hardware uses PCIe 3.0 and 4.0, introduced in 2010 and 2017 respectively [does this need a source?], PCI-SIG has already published their specifications for the PCIe 5.0 and 6.0 standards. These, again, double the bandwidths of the previous generation, enabling theoretical transfer speeds of up to 128GB/sec in both directions on a PCIe 6.0 x16 link.[13] However, it is to be expected that these high-speed interconnect standards will take a few years to become widely available and adopted, as Intel only released their first PCIe 5.0-capable CPUs around the end of 2021. [14] Additionally, other manufacturers and companies have developed their own protocols and standards to extend the feature-set of PCIe, such as Intel's Thunderbolt, which enables PCIe devices to be connected externally with only a small

loss of performance. [15] Another example is the NVMe standard, a PCIe-compatible interface specifically devised and optimized for high-bandwidth, low-latency storage solutions. [16]

2.3 Graphics Processing Units

2.3.1 What are GPUs

To begin with, it should be noted that the term graphics processing unit (GPU) does not equate to a graphics card. A GPU is a specialized processing unit primarily designed for parallel processing and accelerating workloads that require parallel processing. [15] A graphics card, on the other hand, is the add-in card that features a PCI-Express link to facilitate communication between CPU and GPU, dedicated memory and power delivery for the GPU, and the GPU itself. It should also be noted that the graphics card usually has its own dedicated PCB. [does this need citing?] There are also integrated GPUs, which can be embedded alongside the CPU. These integrated GPUs are usually less powerful compared to discrete GPUs. [15]

2.3.2 Uses of GPUs

GPUs originally began as, as their names suggest, dedicated graphics accelerators optimized for floating-point operations, which are essential to 3D graphics rendering. They were initially developed as a hardware pipeline with fixed functionality, namely to render graphics. Over the years, GPU architecture has evolved from essentially being an integrated frame buffer into a set of general-purpose, highly parallel, programmable processing cores, enabling more general-purpose computation. [17] Today, a GPU is more of an accelerator for many different use-cases and workloads. Examples include, for personal use, gaming, video editing, and content creation. [15] On the scientific side, GPUs are frequently used to accelerate workloads that require parallel computing, such as machine learning, fluid dynamics, and data science. [18]

2.3.3 GPU Memory

Addressing GPU memory, assuming that the GPU is connected via PCI-Express and has its own dedicated video memory, works in the same way as addressing memory in other PCIe devices, which was briefly touched upon in section 2.2.3. However, GPU memory usually has higher throughput bandwidths compared to conventional RAM of a similar period. As example, current top of the line graphics cards from Nvidia are equipped with GDDR6X memory, which has a theoretical maximum system bandwith

of one terabyte per second. [19], [20] On the other hand, state of the art main memory, currently DDR4, is limited to a bandwidth of about 35 gigabytes per second. [20]

TODO: DDR5 standard, however: numbers hard to find. what to put?

2.4 CUDA

CUDA is a closed source API developed and maintained by Nvidia for general-purpose GPU computing for their GPUs and graphics cards. It is designed to work with C++ and Fortran and comes with a set of GPU-accelerated libraries, optimization tools, debugging tools, and a C++ compiler. [18] Some sample libraries include: linear algebra, signal processing, and image processing. [21] For this thesis, only the C++ version of CUDA is discussed in further detail.

2.4.1 Kernels and Scalability

CUDA uses kernels, which are an extension to standard C++ functions. Kernels, when called, are executed N times by N different threads, and enable these threads to run on the GPU. This enables heterogeneous programming, which allows serial code to run on the host - the CPU - and parallel code, the kernels, to run on the GPU, thereby leveraging the GPU's increased capabilities for parallel computing to accelerate the workload. The kernel is executed on a thread, many of which make up a block, many of which, in return, make up a grid, the dimensions of which are defined upon calling the kernel. Different blocks can be executed in parallel, or in sequence, in any order, on any of the multiprocessors of a GPU, which enables automatic scalability as the compiled program can run irrespective of the amount of multiprocessors present on the GPU. [22]

2.4.2 Memory Management

CUDA assumes that the CPU and GPU maintain separate memory spaces, and is able to manage both host and device memory. CUDA can both manage shared memory, which is visible and accessible for both the CPU and GPU, and dedicated device memory, which is not accessible by the CPU. CPU memory is, unless overridden by CUDA, managed natively by C++. Memory management includes allocation, deallocation, and data transfer. [22]

[maybe extend this section a bit? Not sure if i covered everything] [add sources] [maybe add graphics]

3 Bandwidth Benchmark

3.1 Concept

- Measures raw theoretical maximum bandwidth of pcie link by measuring the duration of memory copies of various chunk sizes - Checks at which chunk sizes the bandwidth of the link is fully saturated

3.2 Implementation

- Compensates for delay - 1 packet with 4B measured as delay - Pageable and pinned memory benchmarks measured - Warmup-feature: first transfer usually has some sort of longer delay, compensates for that (windows-finding, verify on p6000)

3.3 Results

- Transfer durations don't really increase until 8kb - Due to the nature of the PCI-E packet having a max payload of 4kb - First transfer usually has a bit longer delay (warmup?) - On windows: not executable that calls functions, but rather nvcuda64.dll - requires compiling on windows and then using a profiling tool like AMDUprof to look at the program

3.4 Discussion

3.4.1 successes

- gives accurate reading of pcie bandwidths - non-linear scaling of packet transfer durations (2 packets does not equal double the duration of one packet)

3.4.2 shortcomings

- does not really compensate for other bottlenecks, as seen on time-x with gen4 link bandwidth speeds

4 NVML Counters

4.1 Concept

- nvidia has hardware counters, accessible via nvml library - counters measure average bandwidth over the last 20ms, in kb/sec - Transmit and Receive have seperate counters

4.2 Implementation

- method call to read counters takes about 20ms - to increase data granularity and measurement consistency, measuring of TX and RX was done in parallel

4.3 Results

- graphs

4.4 Discussion

4.4.1 successes

- measures bandwidths accurately to some degree - introduces little overhead (probably, still to be measured)

4.4.2 shortcomings

- granularity of method calls prevent more accurate readings -> short memory transfers may be not detected - black box approach of nvidia's source code doesn't allow for proper sanity-checking

5 Link Saturation

5.1 Concept

- if bandwidth is saturated, copy operations should slow down - full duplex, so HtoD and DtoH both need measuring

5.2 Implementation

- Started as a thread that just continuously monitored the counters for set duration of time and printed output into console - Added wrapper and file output in subsequent versions to simplify data-gathering - Added chunk size options for the buffer copy chunks to get as little overhead as possible while getting most consistent data gathering - Delays are measured, no clear correlation between delay and bandwidth - Overhead yet to be properly assessed, however, introduces somewhat significant overhead. TODO: ASSESS OVERHEAD - Measure transmit and receive in the same thread, sequentially

5.3 Results

- graphs - descriptors of graphs

5.4 Discussion

5.4.1 successes

- gives somewhat detailed going-on about PCIe link activities

5.4.2 shortcomings

- introduces some overhead due to occupying PCIe link - Delay compensation sometimes leads to negative values due to delay inconsistencies

6 Summary

Outlook

List of Figures

2.1 An example of a PCI-Express packet [7] 3
2.2 An example of a memory request header [7] 4
2.3 PCIe configuration on an Intel-based system[11] 6
2.4 An example PCIe link between two devices [12] 7

List of Tables

2.1 PCI Express aggregate bandwidths by generation and link width [9] . . 3

Bibliography

[1] IBM. "What is HPC? introduction to high-performance computing | IBM." (), [Online]. Available: https://www.ibm.com/topics/hpc (visited on 02/20/2022).

[2] I. S. University. "What is an HPC cluster | high performance computing." (), [Online]. Available: https://www.hpc.iastate.edu/guides/introduction-to-hpc-clusters/what-is-an-hpc-cluster (visited on 02/20/2022).

[3] PCI-SIG, *PCI express architecture frequently asked questions*, Sep. 17, 2011.

[4] PCMAG. "Definition of PCI express," PCMAG. (), [Online]. Available: https://www.pcmag.com/encyclopedia/term/pci-express (visited on 02/17/2022).

[5] A. Verma and P. Dahiya, "PCIe BUS: A state-of-the-art-review," *IOSR Journal of VLSI and Signal Processing (IOSR-JVSP)*, vol. 7, pp. 24–28, Jul. 12, 2017. DOI: 10.9790/4200-0704012428.

[6] PCI-SIG. "Contact us | PCI-SIG." (), [Online]. Available: https://pcisig.com/membership/contact-us (visited on 02/14/2022).

[7] J. Lawley, "Understanding performance of PCI express systems," p. 16, 2014.

[8] PCI-SIG. "Membership | PCI-SIG." (), [Online]. Available: https://pcisig.com/membership (visited on 02/14/2022).

[9] M. Jackson, R. Budruk, J. Winkles, and D. Anderson, *PCI Express technology: comprehensive guide to generations 1.x, 2.x, 3.0* (MindShare technology series), 1st ed. Monument, Colo.: MindShare, 2012, 986 pp., OCLC: ocn824814290, ISBN: 978-0-9770878-6-0.

[10] I. Oracle. "PCI address domain - oracle documentation." (), [Online]. Available: https://docs.oracle.com/cd/E19253-01/816-4854/hwovr-25/index.html (visited on 02/14/2022).

[11] H. Nakamura, H. Takayama, Y. Yamaguchi, and T. Boku, "Thorough analysis of PCIe gen3 communication," in *2017 International Conference on ReConFigurable Computing and FPGAs (ReConFig)*, Dec. 2017, pp. 1–6. DOI: 10.1109/RECONFIG.2017.8279824.

[12] R. Budruk, "PCI express basics," Jul. 15, 2014.

[13] D. D. Sharma, "PCI express® 6.0 specification at 64.0 GT/s with PAM-4 signaling: A low latency, high bandwidth, high reliability and cost-effective interconnect," in *2020 IEEE Symposium on High-Performance Interconnects (HOTI)*, ISSN: 2332-5569, Aug. 2020, pp. 1–8. DOI: 10.1109/HOTI51249.2020.00016.

[14] Intel. "Product specifications - i9 12900k." (), [Online]. Available: https://www.intel.com/content/www/us/en/products/sku/134599/intel-core-i912900k-processor-30m-cache-up-to-5-20-ghz.html (visited on 02/20/2022).

[15] Intel. "What is a GPU? graphics processing units defined," Intel. (), [Online]. Available: https://www.intel.com/content/www/us/en/products/docs/processors/what-is-a-gpu.html (visited on 02/15/2022).

[16] Kingston. "Understanding SSD technology: NVMe, SATA, m.2 - kingston technology," Kingston Technology Company. (), [Online]. Available: https://www.kingston.com/germany/en/community/articledetail/articleid/48543 (visited on 02/20/2022).

[17] C. McClanahan, "History and evolution of GPU architecture," p. 7, 2010.

[18] Nvidia. "CUDA zone," NVIDIA Developer. (Jul. 18, 2017), [Online]. Available: https://developer.nvidia.com/cuda-zone (visited on 02/19/2022).

[19] Nvidia. "NVIDIA GeForce RTX 3090 graphics card." (), [Online]. Available: https://www.nvidia.com/en-us/geforce/graphics-cards/30-series/rtx-3090/ (visited on 02/19/2022).

[20] I. Micron Technology. "RAM memory speeds & compatibility | crucial.com," Crucial. (), [Online]. Available: https://www.crucial.com/support/memory-speeds-compatability (visited on 02/19/2022).

[21] Nvidia. "CUDA 11 features revealed," NVIDIA Developer Blog. (May 14, 2020), [Online]. Available: https://developer.nvidia.com/blog/cuda-11-features-revealed/ (visited on 02/15/2022).

[22] Nvidia. "CUDA c++ programming guide." Archive Location: Programming Guides. (), [Online]. Available: https://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html (visited on 02/22/2022).