SYCL Benchmarks on AWS

Giuseppe D'Ambrosio, Computer Science Ph.D. Student at Università degli Studi di Salerno, Italy.

Introduction

SYCL

SYCL is an open standard, single source, high-level, standard C++ programming model published by the Khronos Group that can target a range of heterogeneous platforms. SYCL combines the portability of OpenCL with modern C++, it does not extend the C++ language, but a SYCL program is always a valid C++ program.
SYCL implementation requires a dedicated SYCL compiler that identifies kernels, extracts them, and compiles them into an intermediate representation for the target hardware:

- SPIR and SPIR-V both 32 and 64 bits;
- PTX 64 bits.

ComputeCpp

ComputeCpp is a heterogeneous parallel programming platform that provides a conformant implementation of SYCL 1.2.1 Khronos specification. ComputeCpp supports any platform with OpenCL SPIR 1.2, SPIR-V, or PTX support, with some limitations.

Operating System	OpenCL Platform	Device	Supported
Ubuntu 16.04 64bit	AMD®	GPU	Not Tested
Ubuntu 16.04 64bit	Intel®	CPU	Yes
Ubuntu 16.04 64bit	Intel®	GPU	Yes
Any OS	NVIDIA®	GPU	No
Ubuntu 18.04 64bit	Intel®	CPU	Yes
Ubuntu 18.04 64bit	Intel®	GPU	Yes
CentOS 7.2 64 bit	Intel®	CPU	Yes
CentOS 7.2 64 bit	Intel®	GPU	Yes
Windows 7	AMD®	Cedar	Yes
Windows 7	Intel®	CPU	Yes

Table 1: ComputeCpp Supported Platforms.

Benchmarks

Microbenchmarks A set of architectural microbenchmarks with different patterns stressing different hardware subsystems. They have been designed to emphasize performance characterization on GPU devices.

- DRAM - measures the achievable device memory bandwidth by copying single and double-precision floating-point values between two buffers;
- arith - exercises the device's main arithmetic units;
- sf - exercises the device's special function units;
- HostDeviceBandwidth - measures the transfer bandwidth between the host and device memory by copying large and contiguous chunks of one, two, and three-dimensional buffers.

Applications/Kernel Real-world applications and kernels from various domains such as linear algebra, image processing, molecular dynamics. The main goal is to test the performance of different devices for real-world code patterns.

- KMeans
- LinearRegression Coefficient
- LinearRegression Error
- MatMulChain
- MedianFilter
- MolecularDynamics
- 2DConvolution
- 2mm
- 3mm
- Atax
- Bicg
- Covariance
- Fdtd2d
- Gesummv
- Gramshmidt
- Syr2k
- Syrk
- SegmentedReduction
- ScalarProduct
- Sobel3
- Sobel5
- Sobel7
- VectorAddition

SYCL Runtime Benchmarks Designed to stress the SYCL runtime. These benchmarks include multiple-kernels that stress different aspects of the SYCL runtime.

- DAG Task Throughput - measures the time from the submission to the completion of N kernels trying to access the same buffer in read-write mode. Because more than one kernel accesses the same memory object, a read-write conflict arises that forces the SYCL runtime to process the kernels sequentially. Since the kernel itself is trivial, this benchmark is dominated by the scheduling latency of the ComputeCpp implementation. The benchmark contains various mechanisms in SYCL to submit kernels: single task, basic parallel for, ndrange parallel for, hierarchical parallel for.

- Blocked Transform - divides an input array into chunks of configurable size and submits a kernel for each chunk that requests read/write access only to its chunk. Each kernel performs a tunable number of Mandelbrot iterations on the input data to extend the kernel runtime. The focus is to test the ability to automatically overlap the data transfers needed to copy the chunk data to the device and the kernels operating on each chunk. Because the kernels are independent multiple kernels concurrently can be executed if the hardware supports this.

Experimental Settings

Software

Ubuntu 16.04
ComputeCpp 2.3
NVIDIA OpenCL 1.2
Intel® CPU Runtime for OpenCL™ Applications 18.1

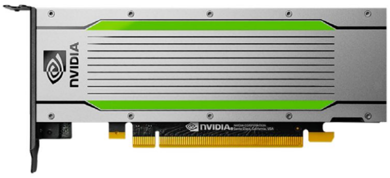
Hardware

All the experiments have been performed on AWS Amazon EC2 instances.

Name	Instance	Processor	vCPU	Mem	GPU	GPU Mem
Config 1	g4dn.large	Intel Xeon Platinum 8259CL 2.50GHz	4	16GB	NVIDIA T4 Tensor Core	16GB
Config 2	g3s.xlarge	Intel Xeon E5-2686 2.3/2.7 GHz	4	30.5GB	NVIDIA Tesla M60	8GB
Config 3	p2.xlarge	Intel Xeon E5-2686 2.3/2.7 GHz	4	61GB	NVIDIA K80 GPU	12GB
Config 4	p3.2xlarge	Intel Xeon E5-2686 2.3/2.7 GHz	8	61GB	NVIDIA Tesla V100	16GB
Config 5	a1.xlarge	AWS Graviton 64-bit Arm Neoverse cores	4	8GB	-	-

Table 2: Experimental Settings.

Tesla T4



SPECIFICATIONS

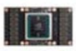
GPU Architecture	**NVIDIA Turing**
NVIDIA Turing Tensor Cores	**320**
NVIDIA CUDA® Cores	**2,560**
Single-Precision	**8.1 TFLOPS**
Mixed-Precision (FP16/FP32)	**65 TFLOPS**
INT8	**130 TOPS**
INT4	**260 TOPS**
GPU Memory	**16 GB GDDR6 300 GB/sec**
ECC	**Yes**
Interconnect Bandwidth	**32 GB/sec**
System Interface	**x16 PCIe Gen3**
Form Factor	**Low-Profile PCIe**
Thermal Solution	**Passive**
Compute APIs	**CUDA, NVIDIA TensorRT™, ONNX**

Tesla M60

SPECIFICATIONS

Virtualization Use Case	**Performance-Optimized Graphics Virtualization**
GPU Architecture	**NVIDIA Maxwell™**
GPUs per Board	**2**
Max User per Board	**32 (16 per GPU)**
NVIDIA CUDA® Cores	**4096 NVIDIA CUDA Cores (2048 per GPU)**
GPU Memory	**16 GB of GDDR5 Memory (8 per GPU)**
H.264 1080p30 streams	**36**
Max Power Consumption	**300 W**
Thermal Solution	**Active/Passive**
Form Factor	**PCIe 3.0 Dual Slot**

Tesla V100



	Tesla V100 PCIe	Tesla V100 SXM2
GPU Architecture	**NVIDIA Volta**	
NVIDIA Tensor Cores	**640**	
NVIDIA CUDA® Cores	**5,120**	
Double-Precision Performance	**7 TFLOPS**	**7.5 TFLOPS**
Single-Precision Performance	**14 TFLOPS**	**15 TFLOPS**
Tensor Performance	**112 TFLOPS**	**120 TFLOPS**
GPU Memory	**16 GB HBM2**	
Memory Bandwidth	**900 GB/sec**	
ECC	**Yes**	
Interconnect Bandwidth*	**32 GB/sec**	**300 GB/sec**
System Interface	**PCIe Gen3**	**NVIDIA NVLink**
Form Factor	**PCIe Full Height/Length**	**SXM2**
Max Power Comsumption	**250 W**	**300 W**
Thermal Solution	**Passive**	
Compute APIs	**CUDA, DirectCompute, OpenCL™, OpenACC**	

Tesla K80

SPECIFICATIONS

GPU Architecture	**NVIDIA Kepler**
Memory clock	**2.5 GHz**
NVIDIA CUDA® Cores	**4992**
Single-Precision	**8.74 TFLOPS**
Double-Precision	**2.91 TFLOPS**
Memory Bandwidth	**480 GB/sec**
GPU Memory	**24GB GDDR5 (12 GB per GPU)**
ECC	**Yes**
Interconnect Bandwidth	**32 GB/sec**
System Interface	**x16 PCIe Gen3**
Form Factor	**Dual Slot, Full Height**
Thermal Solution	**Passive**
Processor core clock	**562 - 875 MHz**

Figure 1: NVIDIA GPUs specifications.

GPU	Pipelines	Core clock speed	Transistor count
Tesla T4	2560	1590 MHz	13,600 million
Tesla M60	2048	1178 MHz	5,200 million
Tesla V100	5120	1370 MHz	21,100 million
Tesla K80	2496	562 MHz	7,100 million

Table 3: NVIDIA GPUs specification comparison.

Intel Xeon E5-2686

# of Cores	22
# of Threads	44
Processor Base Frequency	2.20 GHz
Max Turbo Frequency	3.60 GHz
Cache	55 MB Intel® Smart Cache
Bus Speed	9.6 GT/s
# of QPI Links	2
Intel® Turbo Boost Technology 2.0 Frequency‡	3.60 GHz
TDP	145 W
VID Voltage Range	0

Intel Xeon Platinum 8259CL

# of Cores	24
# of Threads	48
Processor Base Frequency	2.40 GHz
Max Turbo Frequency	3.90 GHz
Cache	35.75 MB
# of UPI Links	3
TDP	165 W

Figure 2: Intel CPUs specifications.

Failed configuration

An additional configuration was included but not tested, specifically using the AWS EC2 g4ad.4xlarge instance built with AMD Radeon Pro V520 GPU. The impossibility to test derives from the lack of support with ComputeCpp, specifically AMD driver are available only for Ubuntu 18.04 and above, while ComputeCpp supports AMD only with Ubuntu 16.04.

GPU Benchmarks

In order to target NVIDIA GPUs with ComputeCpp, it is necessary to use the experimental ComputeCpp PTX backend.

Configurations used: Config 1, Config 2, Config 3, Config 4.

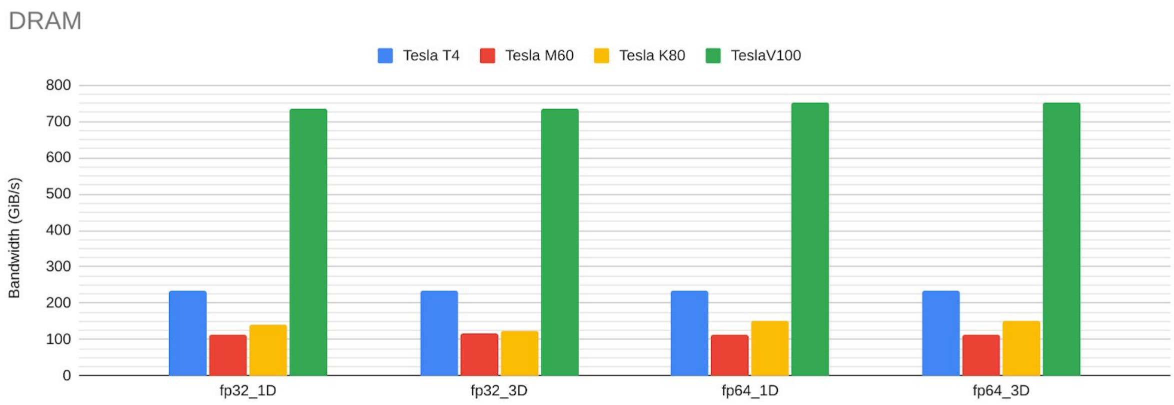


Figure 3: GPU DRAM Benchmark.

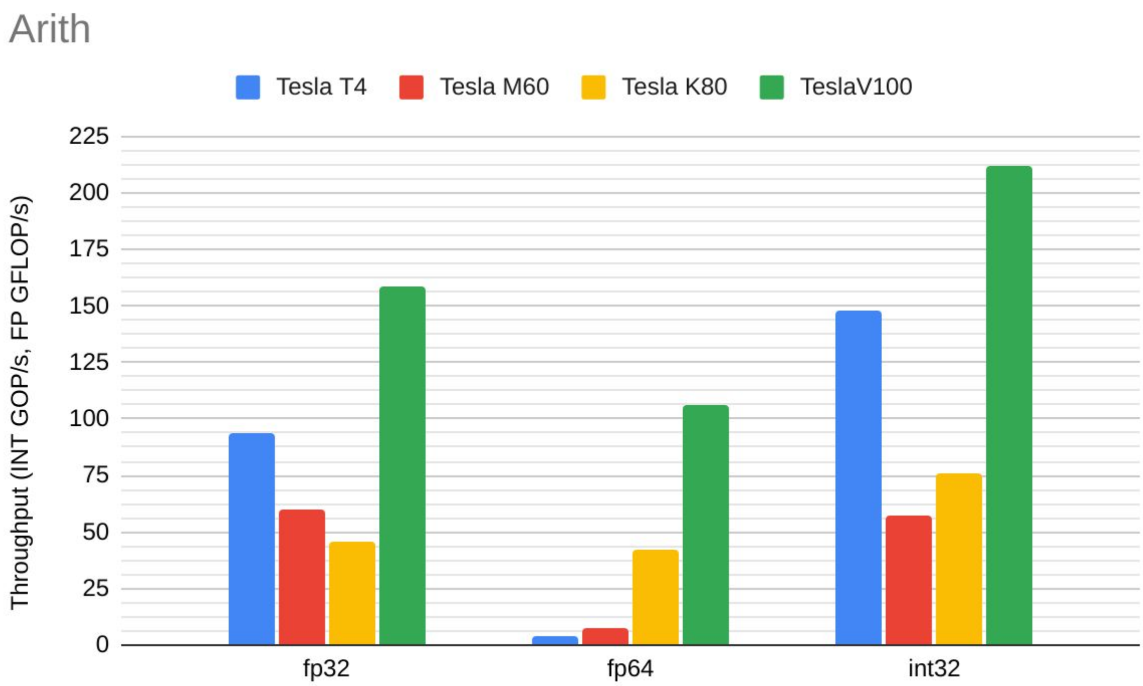


Figure 4: GPU arith Benchmark.

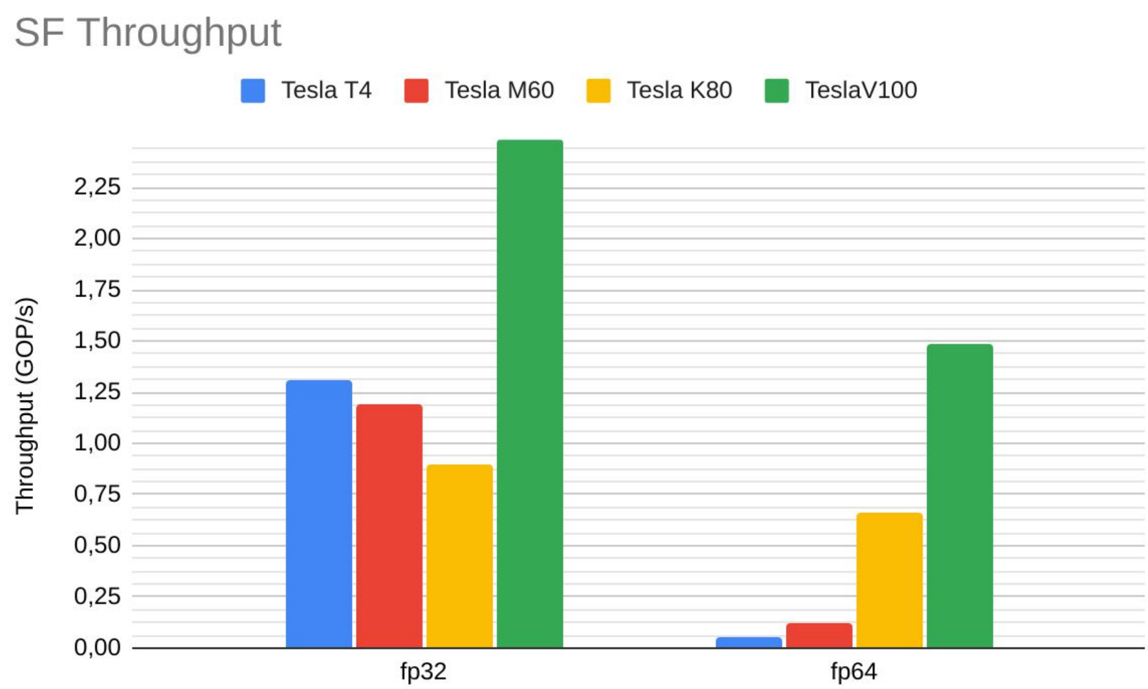


Figure 5: GPU sf Benchmark.

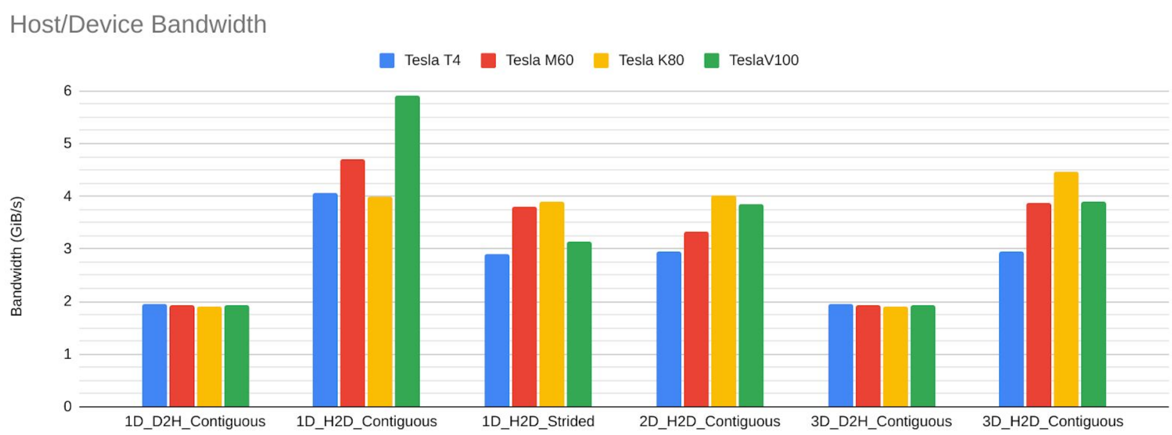


Figure 6: GPU HostDeviceBandwidth Benchmark.

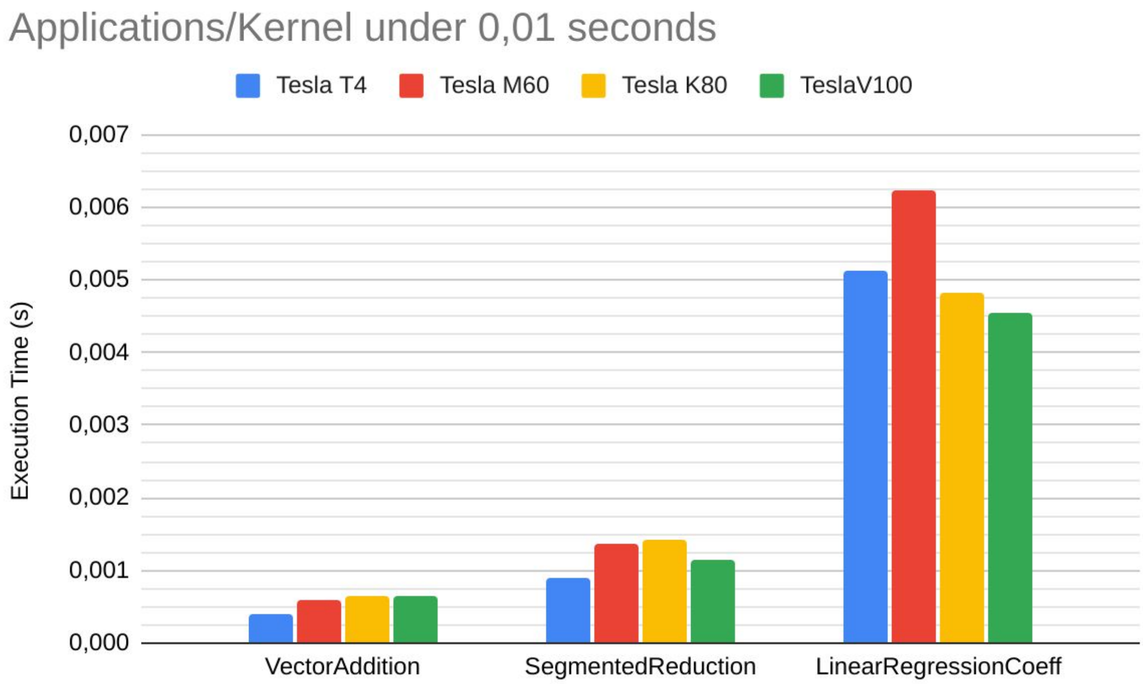


Figure 7: GPU Applications/Kernel Benchmark with Execution Time less than 0,001 seconds.

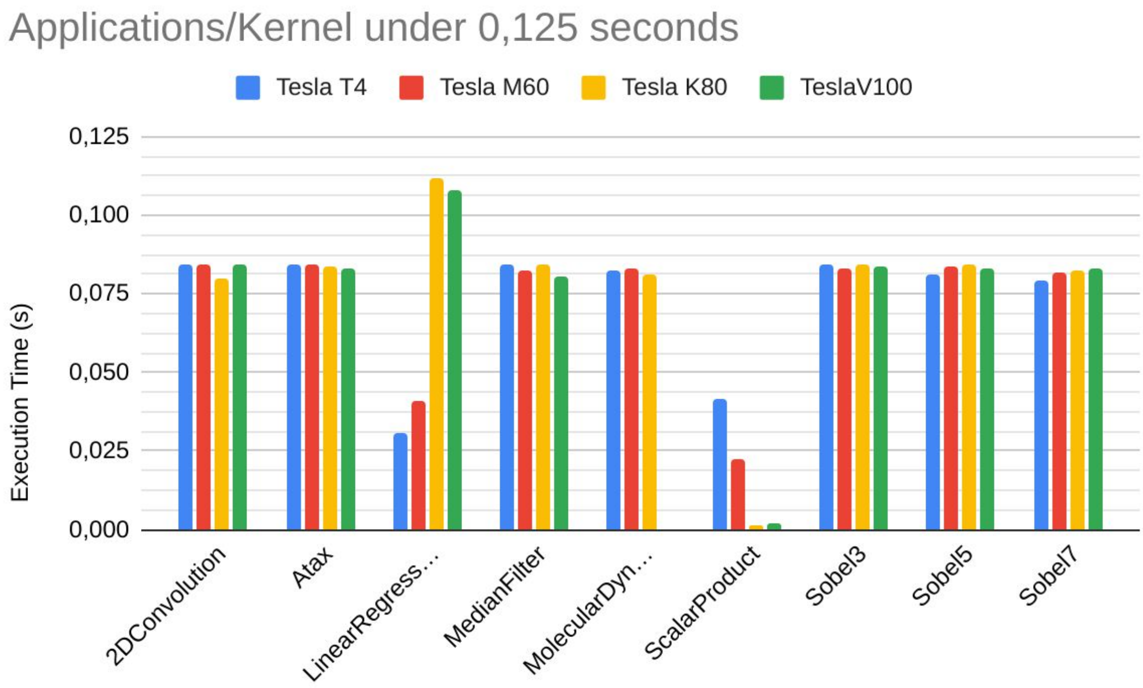


Figure 8: GPU Applications/Kernel Benchmark with Execution Time less than 0,125 seconds.

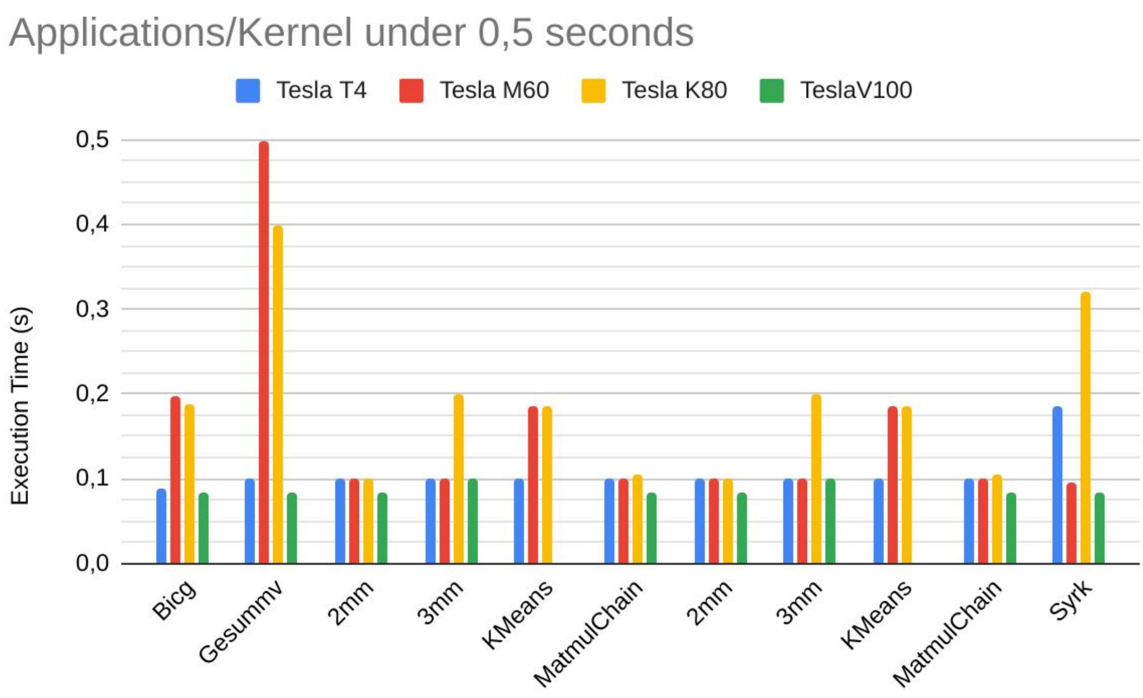


Figure 9: GPU Applications/Kernel Benchmark with Execution Time less than 0,5 seconds.

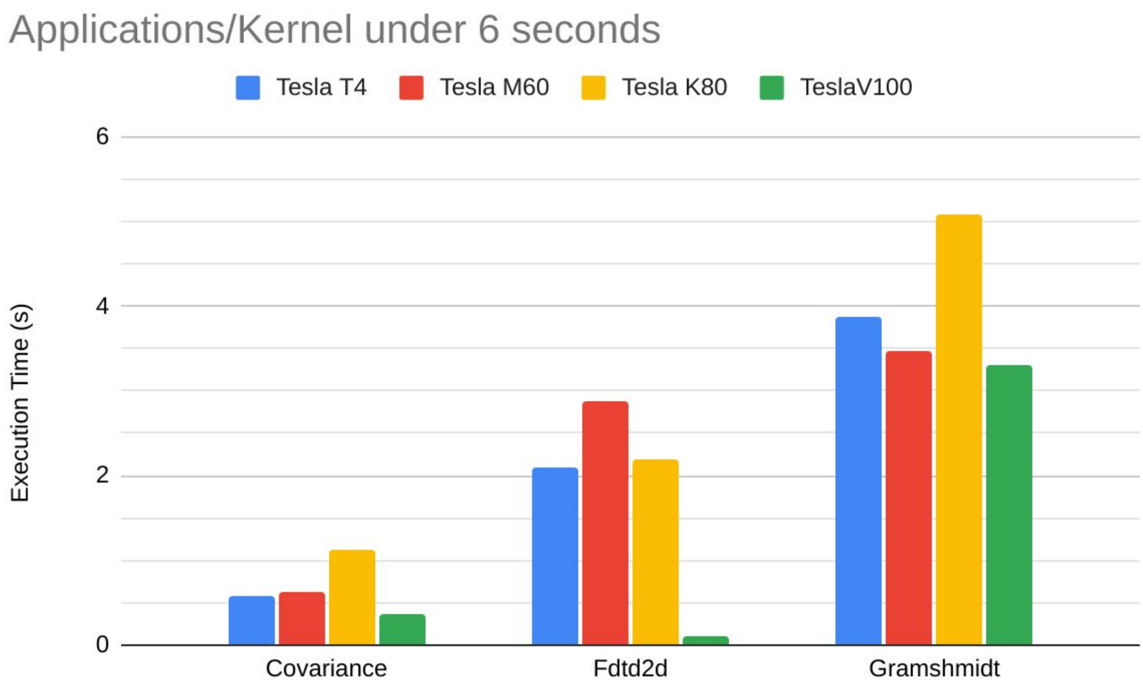


Figure 10: GPU Applications/Kernel Benchmark with Execution Time less than 6 seconds.

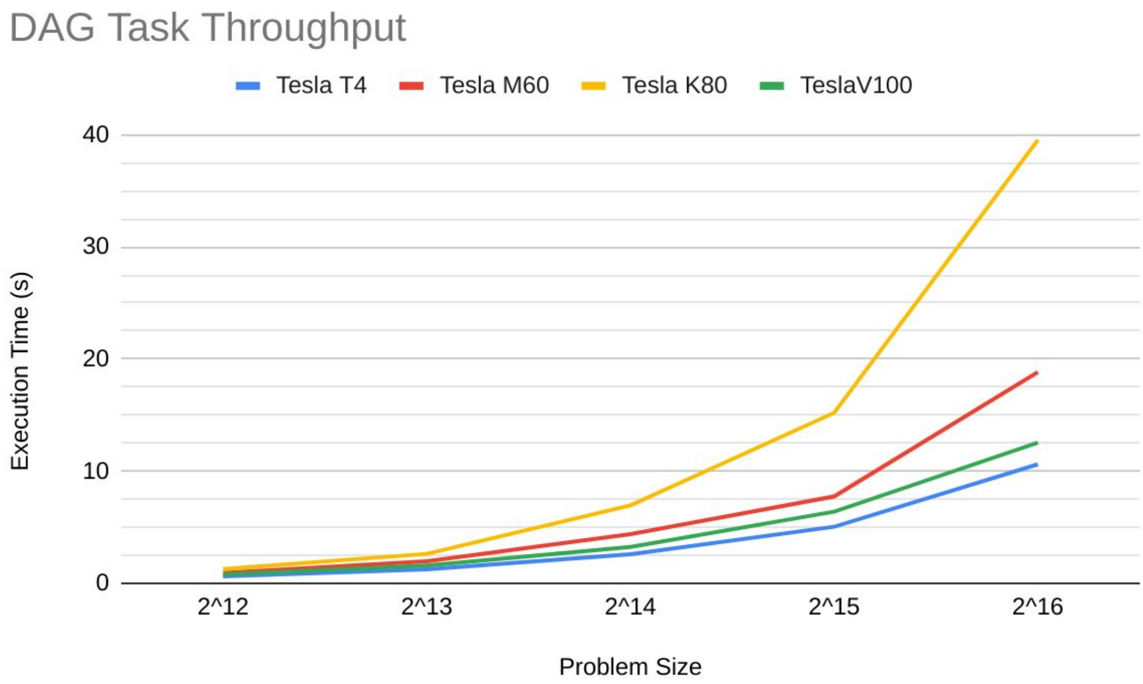


Figure 11: GPU DAGTaskThroughput Benchmark.

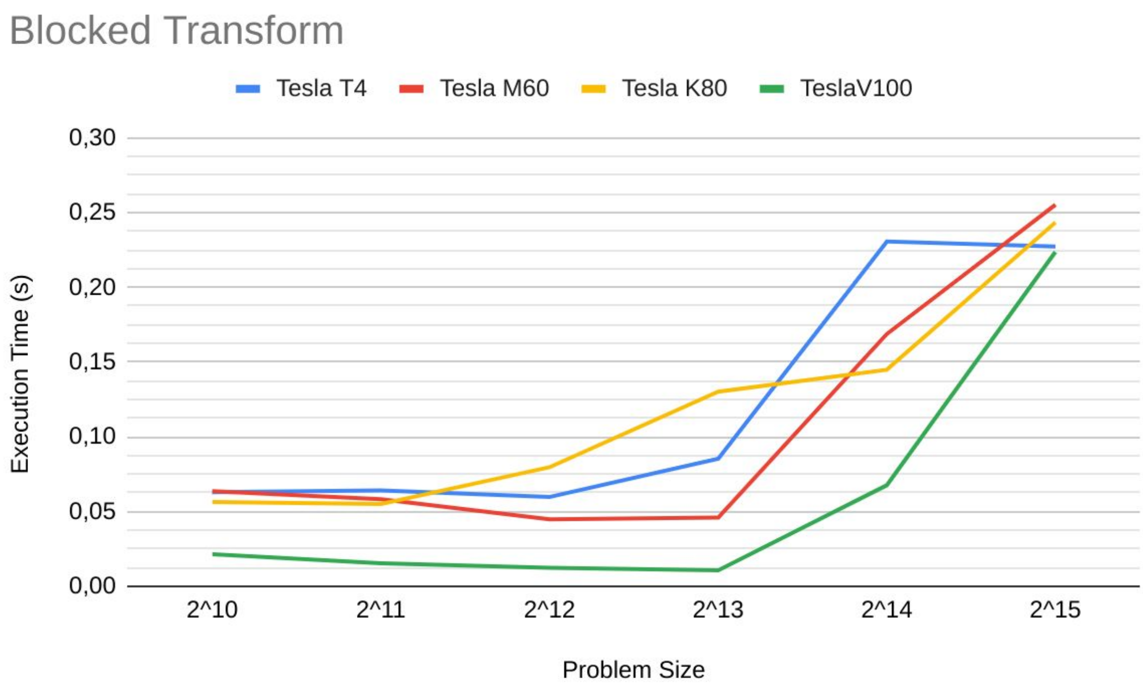


Figure 12: GPU BlockedTransform Benchmark.

CPU Benchmarks

Configurations used: Config 1, Config 3, Config 5.

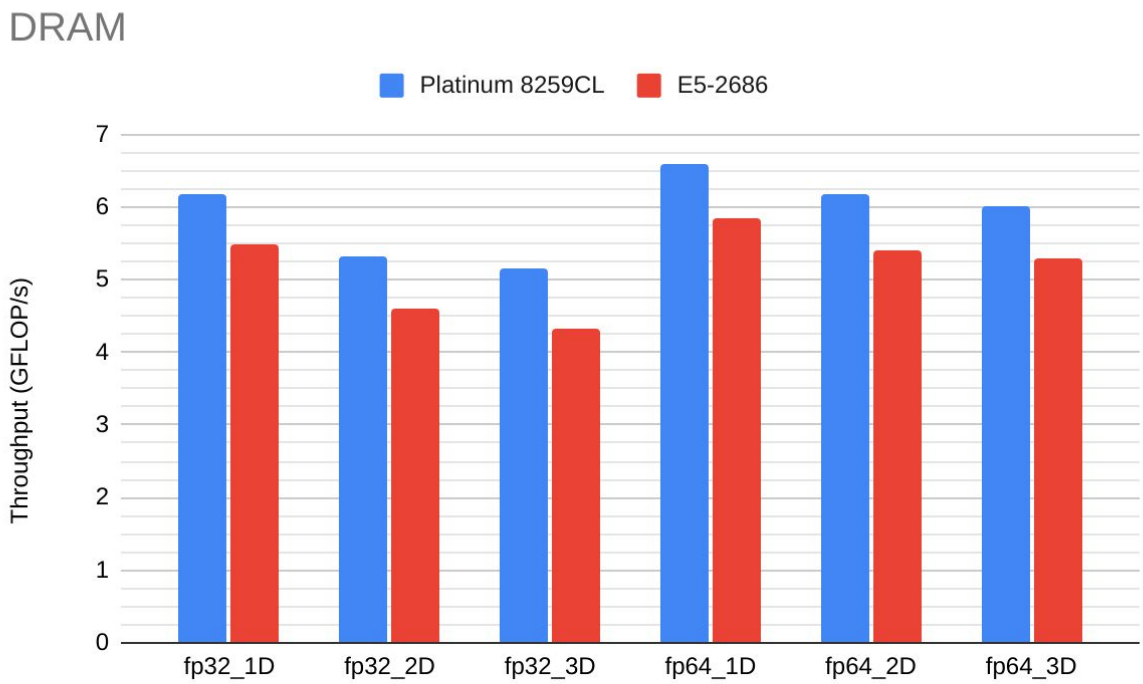


Figure 13: CPU DRAM Benchmark.

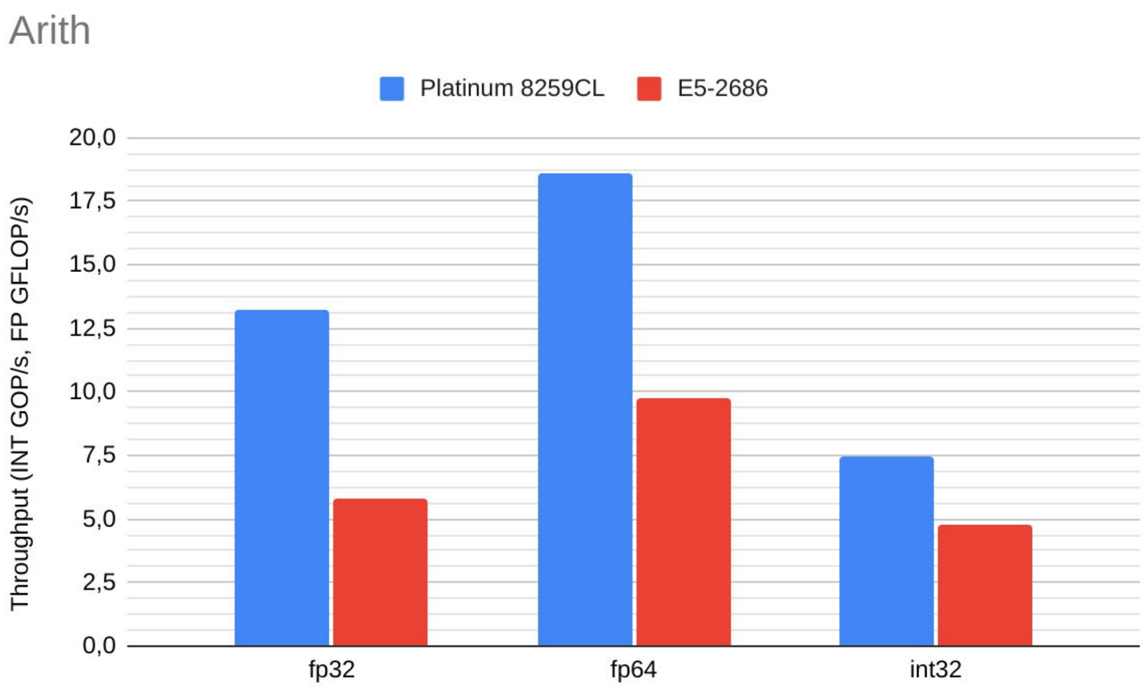


Figure 14: CPU arith Benchmark.

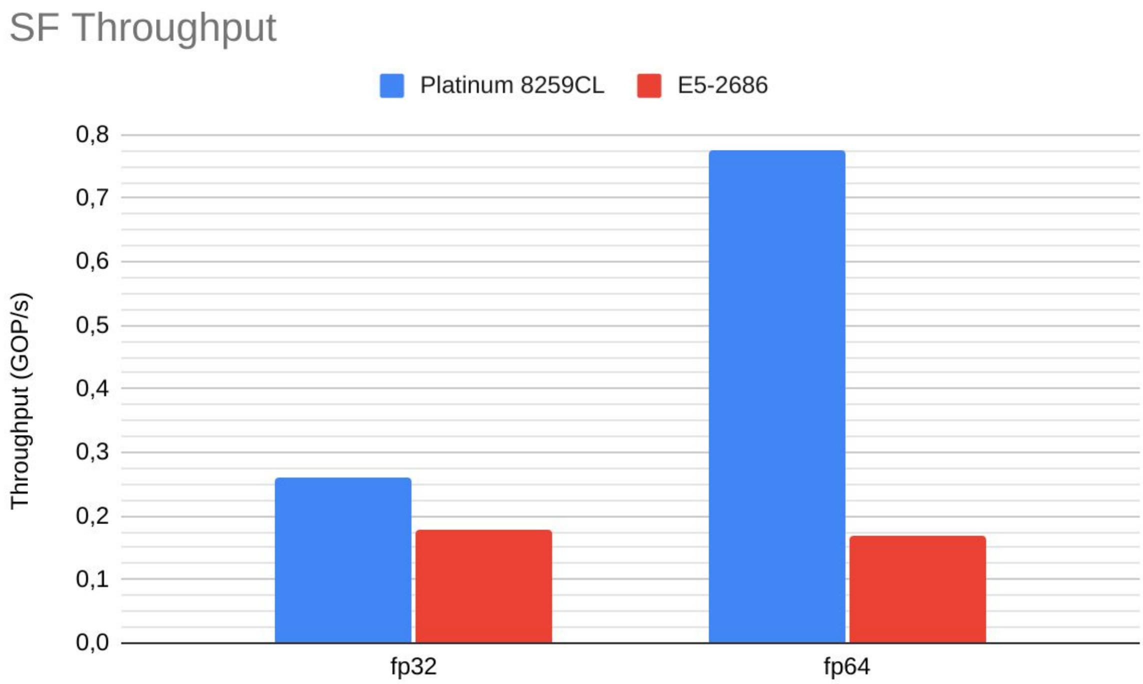


Figure 15: CPU sf Benchmark.

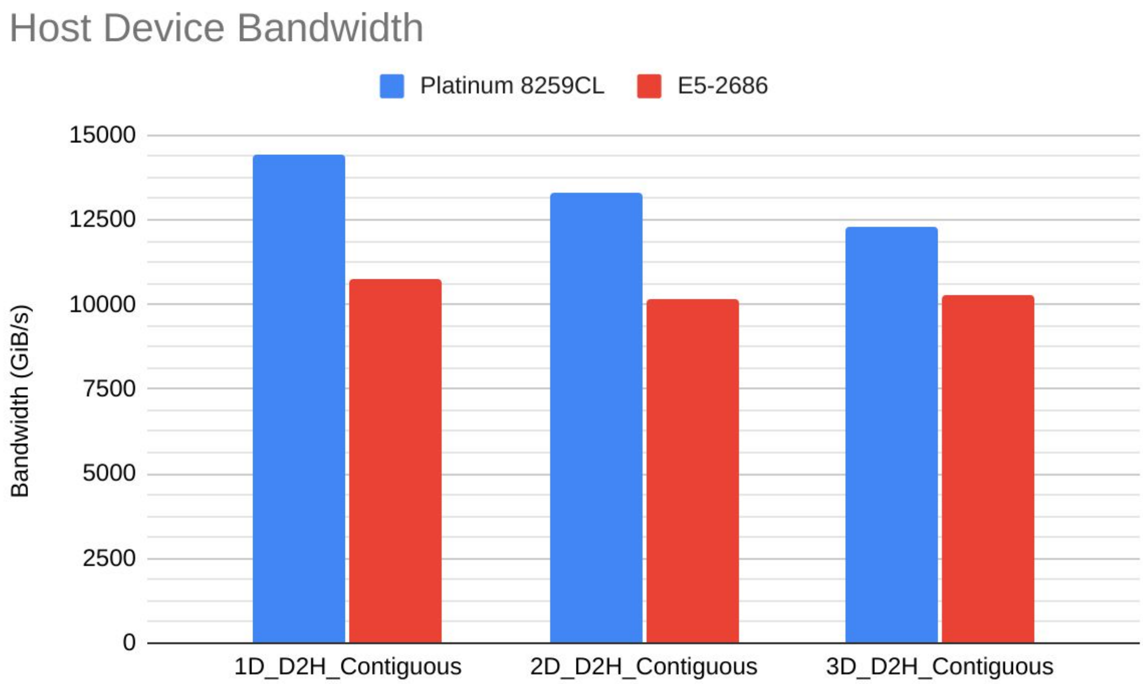


Figure 16: CPU HostDeviceBandwidth Benchmark.

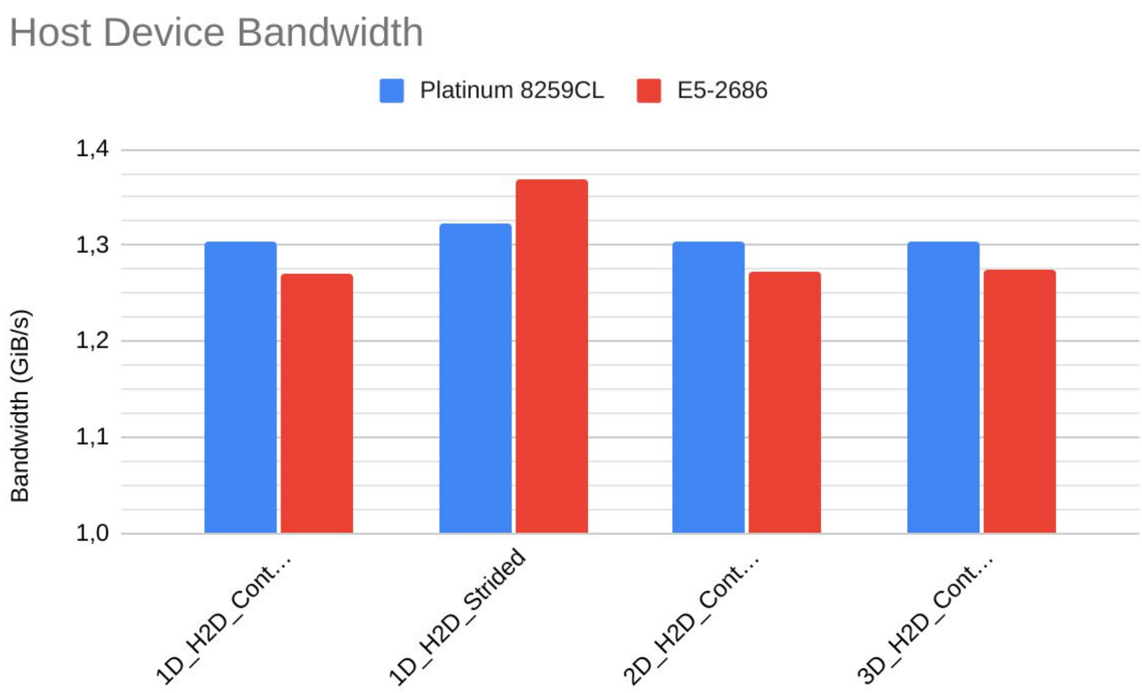


Figure 17: CPU HostDeviceBandwidth Benchmark.

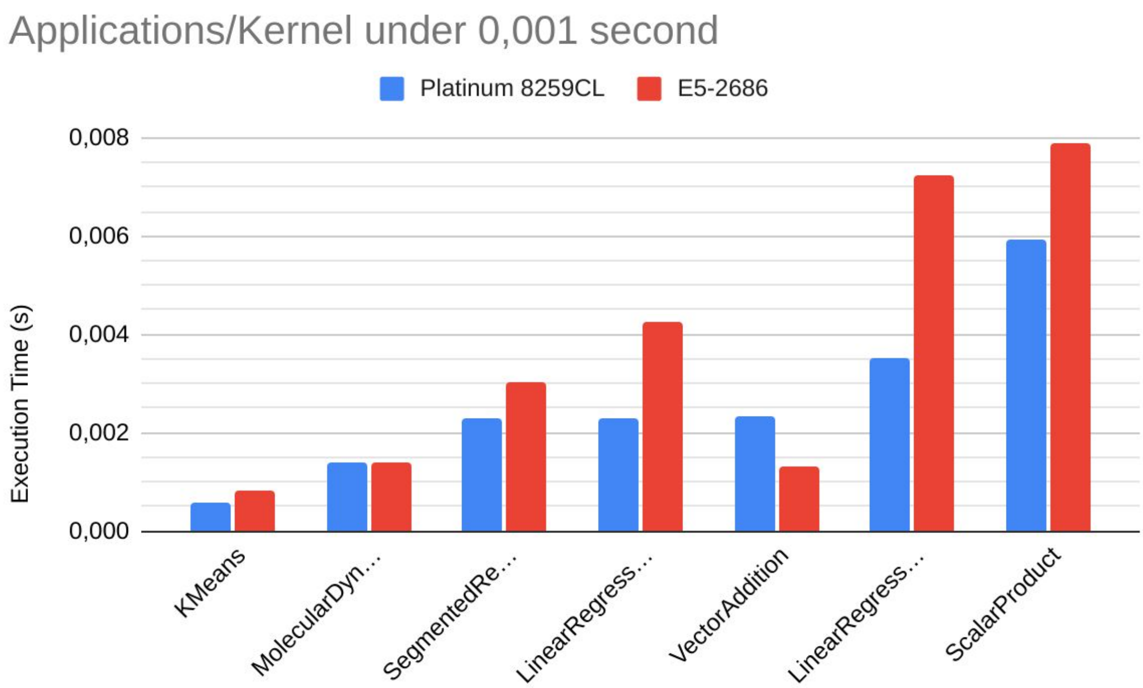


Figure 18: CPU Applications/Kernel Benchmark with Execution Time less than 0,001 seconds.

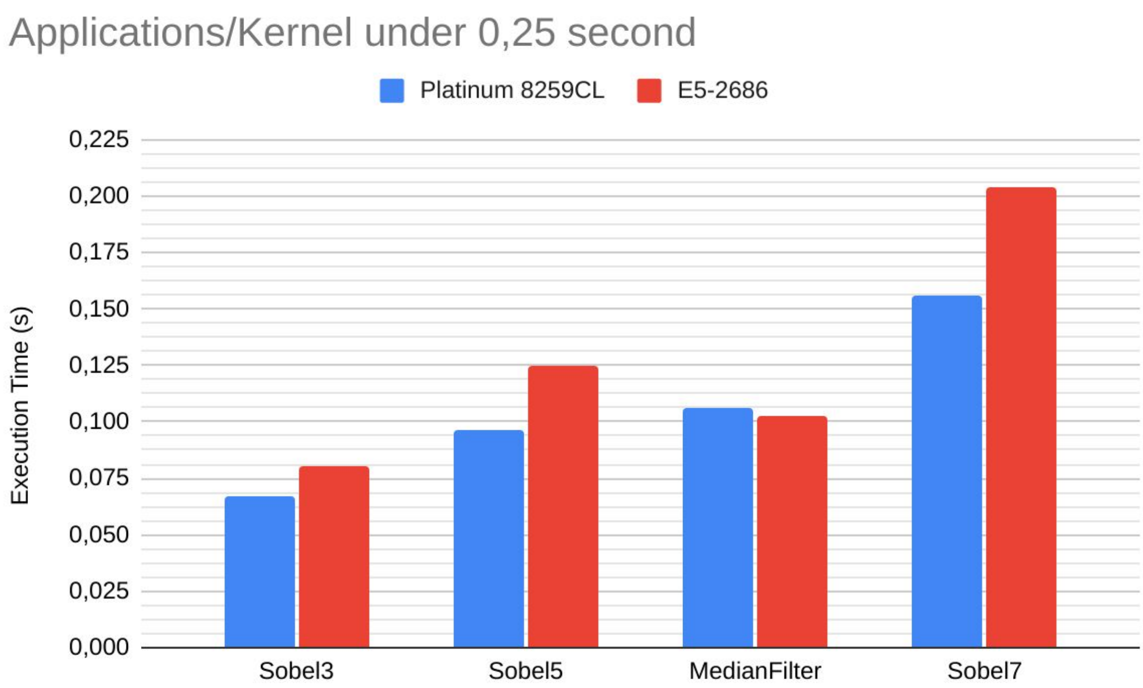


Figure 19: CPU Applications/Kernel Benchmark with Execution Time less than 0,25 seconds.

Benchmark	Platinum 8259CL	E5-2686
3mm	0,001385	0,839408
Correlation	0,001385	0,530623
Gesummv	0,001385	0,388532
Syrk	0,001385	0,302702
Gramshmidt	0,891614	7,558545
2DConvolution	0,891614	0,022069
Atax	0,891614	0,018378
2mm	0,106416	0,560319
Bicg	0,106416	0,438715
Fdtd2d	0,106416	1,029915
Syr2k	0,106416	1,108092
MatmulChain	0,891614	0,941177
Covariance	0,891614	0,533204

Table 4: CPU Applications/Kernel Benchmark with significant differences in Execution Time.

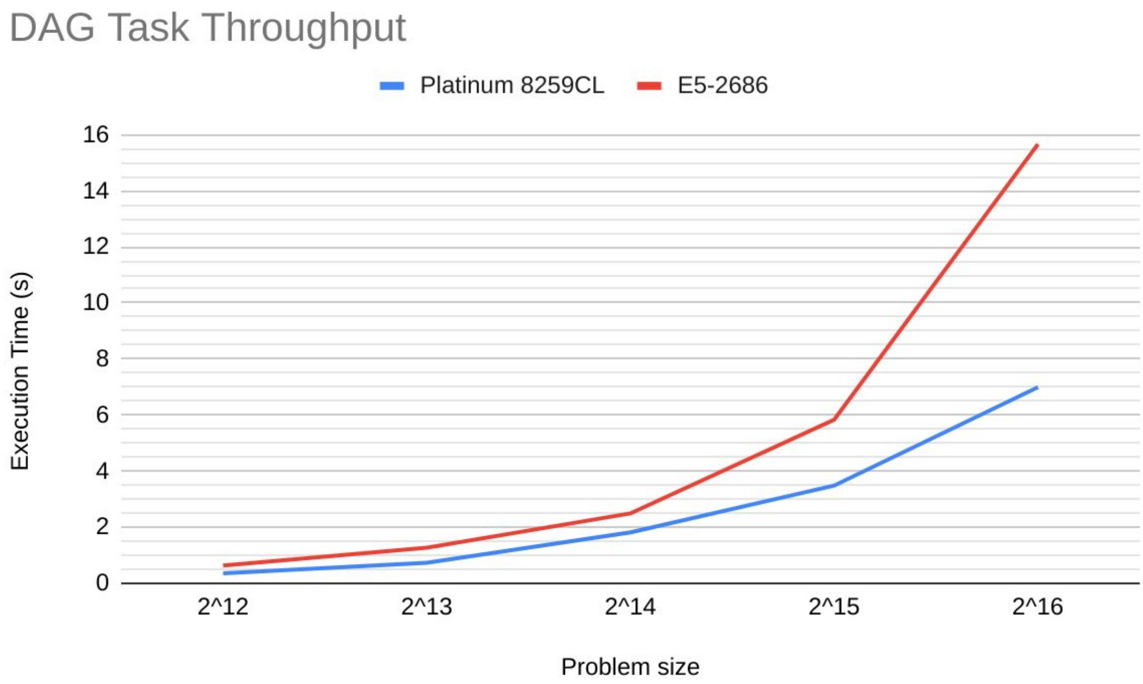


Figure 20: CPU DAGTaskThroughput Benchmark.

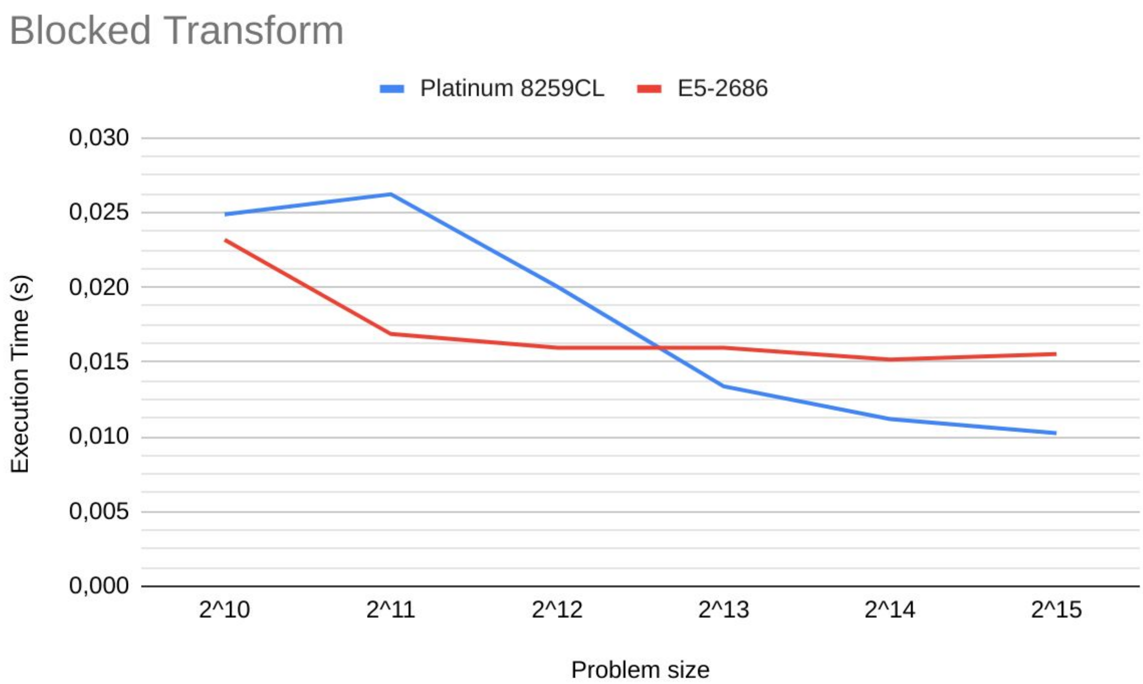


Figure 21: CPU BlockedTransform Benchmark.

Benchmark	Platinum 8259CL	E5-2686	ARM
KMeans	0,000586	0,000826	0,058203
MolecularDynamics	0,001385	0,001388	0,925868
LinearRegression	0,002312	0,004261	1,047198

Table 5: CPU Benchmarks with ARM CPU involved.

References

This project is based on the work found in

- Lal, Sohan & Alpay, Aksel & Salzmann, Philip & Cosenza, Biagio & Hirsch, Alexander & Stawinoga, Nicolai & Thoman, Peter & Fahringer, Thomas & Heuveline, Vincent. (2020). SYCL-Bench: A Versatile Cross-Platform Benchmark Suite for Heterogeneous Computing.

The code and the raw results can be found in

- https://github.com/peppekristen/SyclBench