Market Risk Analytics Enhanced: Empowering Finance with NVIDIA GPUs on AWS

Abstract:

The financial sector, especially in the context of market risk assessment, heavily relies on computational applications. However, many of these applications are traditionally built on outdated CPU technology, leading to performance limitations. To address these limitations, financial institutions often resort to purchasing licenses for third-party grid applications like SGE (Sun Grid Engine) and Data Synapse. Additionally, high-performance computing (HPC) applications in this domain are typically developed using tools like the GCC (GNU Compiler Collection) or Windows C#, which lack support for advanced hardware technologies such as GPUs (Graphics Processing Units) or TPUs (Tensor Processing Units).

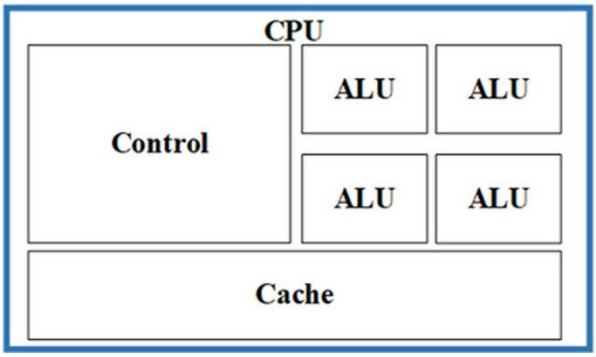
This study explores the challenges and opportunities in optimizing high-compute applications in financial computing. It investigates the utilization of NVIDIA CUDA on AWS GPU instances to overcome these challenges and improve performance significantly. The study focuses on performance benchmark comparisons between native CPU compilation and the substantial performance gains achieved through the utilization of Amazon Cloud (NVIDIA GPU) infrastructure.

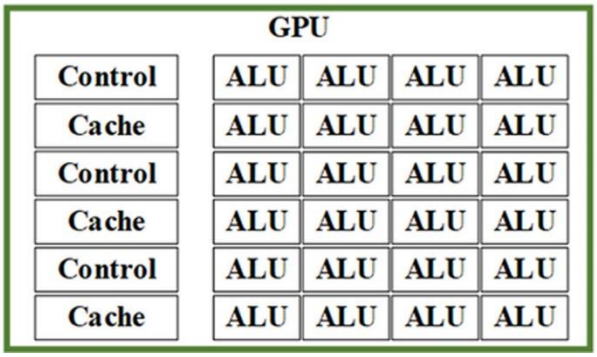
GPU Architecture Overview:

GPUs excel in throughput, with their SIMT architecture, hundreds of stream processors (SPs) per stream multiprocessor (SM), and shared resources. This design, emphasizing execution units, results in high data throughput and energy efficiency. In contrast, CPUs prioritize low-latency calculations and rely on complex control units and large caches, leading to fewer execution units and smaller data throughput. CPU's design requires mechanisms for cache hit rate and data consistency.

Number of Cores: Modern GPUs are designed with a massive number of cores, which are individual processing units responsible for executing tasks in parallel. These cores are organized into streaming multiprocessors (SMs) or compute units. For example, NVIDIA's high-end GPUs can have thousands of CUDA cores within multiple SMs.

CPU and GPU Archicture:





Why GPU :

Parallelism: The abundance of cores in a GPU enables massive parallelism. Each core can perform its calculations independently, allowing the GPU to handle thousands of parallel threads simultaneously. This parallelism is particularly advantageous for tasks like matrix multiplication, where many calculations can be executed concurrently, leading to significant speedup.

Memory Hierarchy: The memory hierarchy ensures efficient data access. Global memory provides a large storage capacity for data, while shared memory and caches reduce memory latency. This hierarchy minimizes the time cores spend waiting for data and maximizes computational throughput.

Data-Parallel Processing: GPUs are optimized for data-parallel processing, where the same operation is applied to multiple data elements simultaneously. This makes them well-suited for tasks that involve applying the same mathematical operations to large arrays or matrices, such as those encountered in financial computations. **Memory Hierarchy:** GPU architecture includes multiple levels of memory hierarchy to efficiently manage data access.

Global Memory: This is the primary memory pool accessible by all cores. It stores data used by the entire GPU and has a relatively large capacity.

Shared Memory: Shared memory is a small, high-speed memory pool that can be accessed by threads within the same thread block. It is used for fast data sharing between threads

L1 and L2 Cache: GPUs often have cache levels (L1 and L2) to reduce memory latency. These caches store frequently accessed data and instructions, accelerating memory access.

Specialized Hardware Units: Modern GPUs may include specialized hardware units like Tensor Cores for accelerating specific types of computations, such as tensor operations commonly used in deep learning. These units further enhance computational power for specific workloads.

High Memory Bandwidth: The high memory bandwidth ensures that data can be fetched and processed quickly, reducing the time cores spend idle due to memory bottlenecks. This is crucial for tasks involving extensive data manipulation, such as matrix multiplications and complex financial calculations.

GPU-based matrix multiplication implementation :

(i) CUDA C, an extension of the C language, leverages shared memory for GPU programming, improving data read and write speeds in many-core processors.

(ii) CuBLAS, an NVIDIA library, offers high-performance matrix multiplication via functions like cublasSgemm.

Python itself does not directly support CUDA or cuBLAS, as these are low-level libraries for parallel computing and GPU acceleration primarily used in languages like C and C++.

However, there are Python libraries and frameworks that provide bindings or interfaces to CUDA and cuBLAS to enable GPU acceleration in Python-based applications. Some of these libraries include: PyCUDA, CuPy, TensorFlow and PyTorch

Experimental Results – Matrix Multiplication Benchmarks:

AWS Instance : g4dn.xlarge

AWS G4dn instances featuring NVIDIA T4 GPUs and custom Intel Cascade Lake CPUs, and are optimized for HPC. These instances libraries such as CUDA, CuDNN, and NVENC. The g4dn.xlarge instance is in the gpu instance family with 4 vCPUs, 16.0 GiB of memory and up to 25 Gibps of bandwidth.

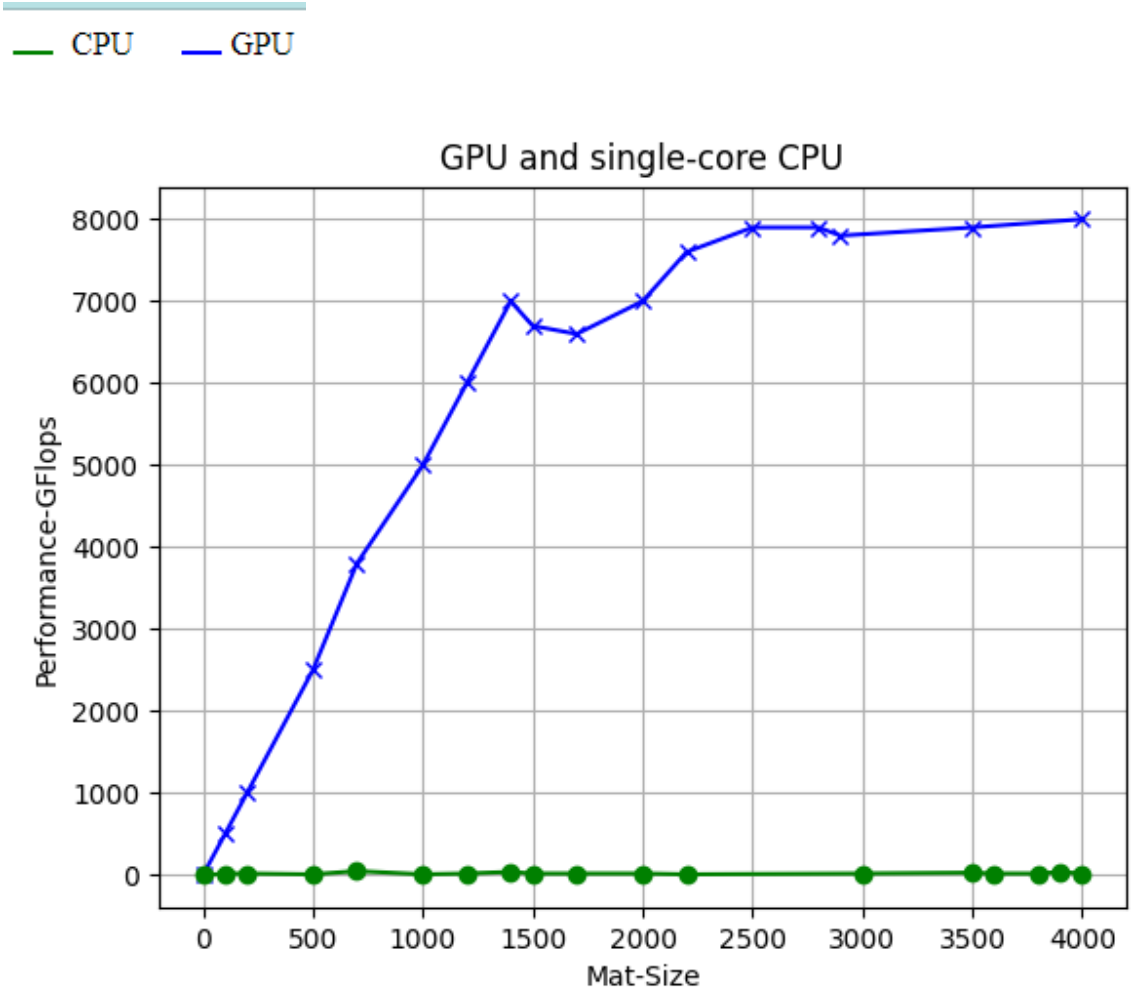
Server information:

Compute	Value
vCPUs	4
Memory (GiB)	16.0
Memory per vCPU (GiB)	4.0
Physical Processor	Intel Xeon Family
Clock Speed (GHz)	2.5
CPU Architecture	x86_64
GPU	1
GPU Architecture	nvidia t4 tensor core
Video Memory (GiB)	16
GPU Compute Capability (?)	7.5
FPGA	0

Matrix multiplication is a fundamental operation in many financial computations. Given two dense matrices A and B of dimensions $M \times K$ and $K \times N$, respectively, the goal is to compute their dot product $C = A.B$, also known as matmul. The dot product is defined as:

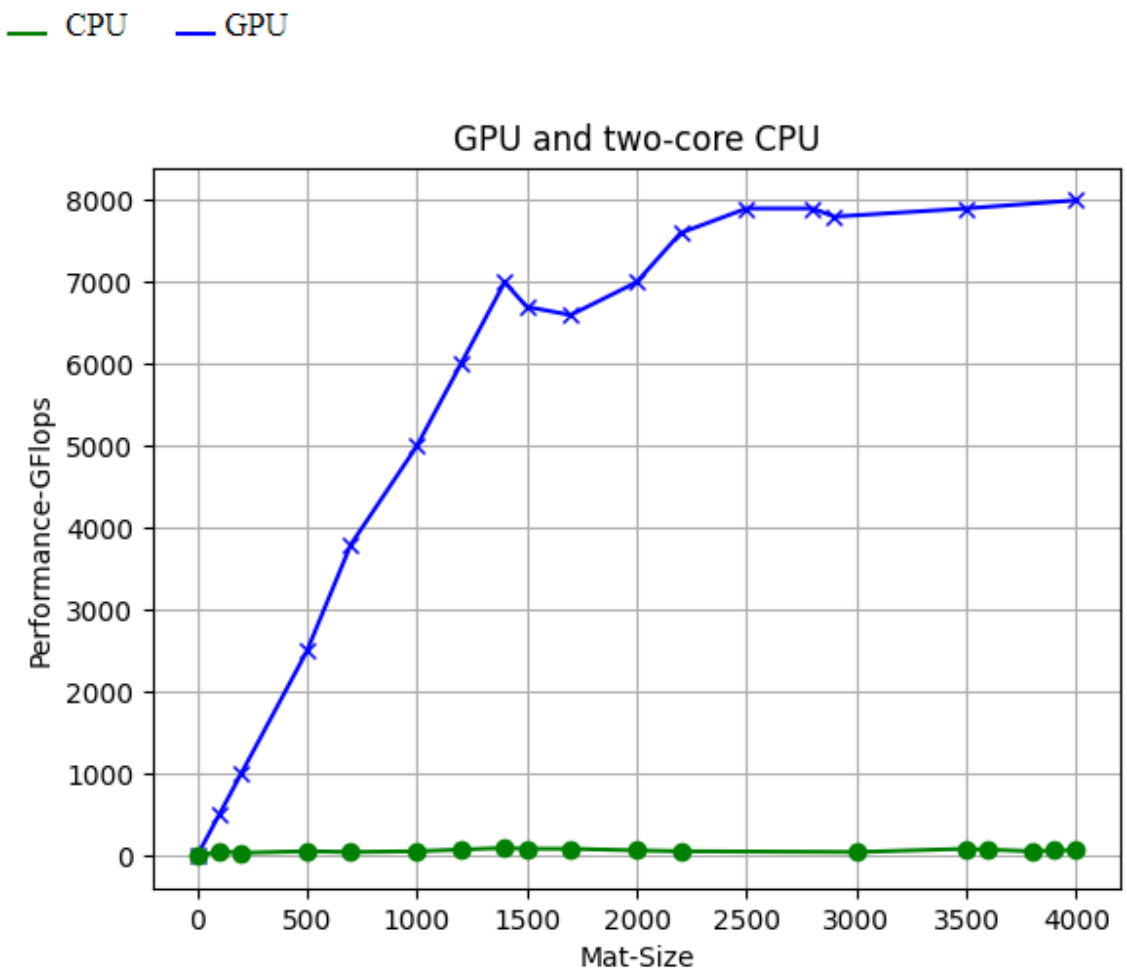
$$Ci,j += \sum k \in [0 \cdots K) Ai,kBk,j$$

Single-core CPU and GPU:



Most of the cores are idle showing low parallelism. Since the whole chip is equipped with many cores, the resources of each core are less than the single core of the CPU, so performance is poor and the overall computing time is long. With the increase of data size, as is shown in Fig. 13 the degree of parallelism in data processing in the GPU is increased and the GPU efficiency has a distinct advantage over the CPU.

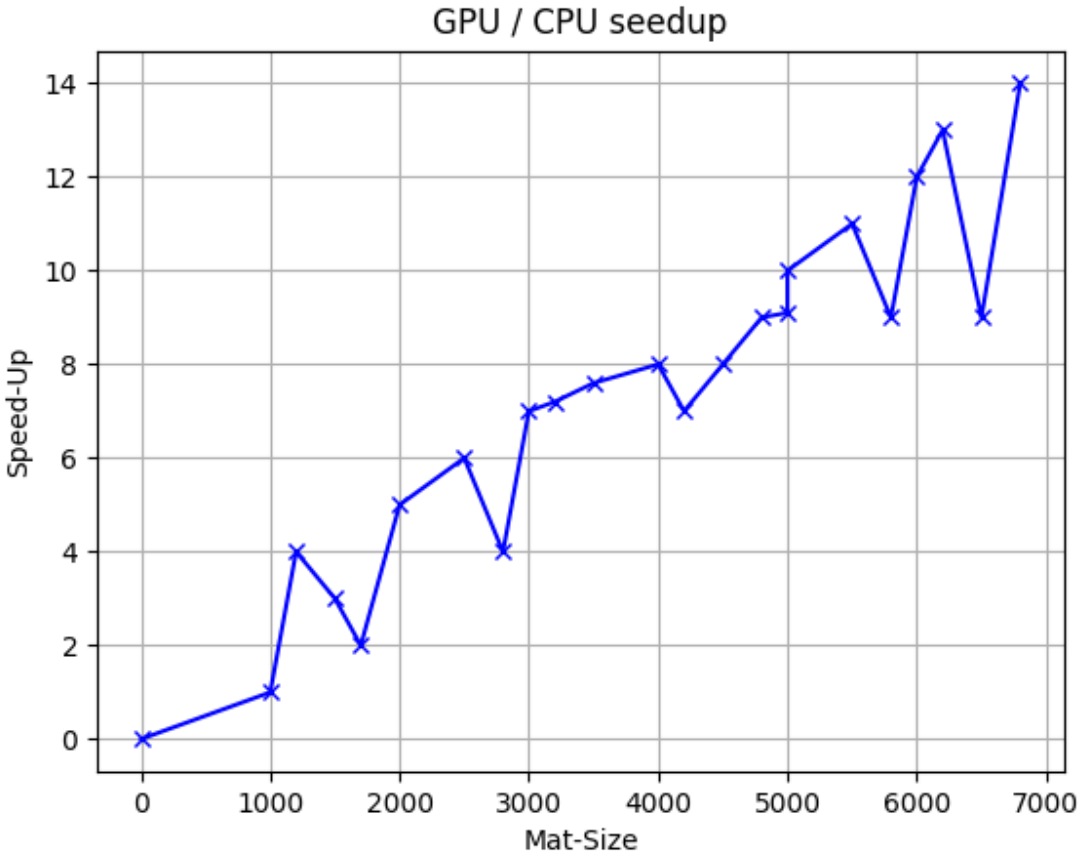
Multi-core CPU and GPU :



As the size of the matrix increases, the computing efficiency of the multi-core GPU increases rapidly while the CPU maintains a relatively low computing efficiency.

GPU-CPU speed up (TF GPU and TF CPU) :

Following study compares Tensorflow's performance testing on CPU and GPU computing platforms, for matrix multiplication of the Tensorflow CPU version and the matrix multiplication accelerated ratio curve of the GPU version.



The GPU-to-CPU speedup ratio increases with the increase of the matrix calculation scale.

Other Banking Use Cases where NVIDIA GPU and CUDA can be used :

- NVIDIA GPU and CUDA offer transformative benefits in various banking HPC applications.
- They enhance VaR and CVaR calculations, portfolio optimization, and real-time risk assessment.
- Stress testing and scenario analysis benefit from efficient simulations.
- Wealth management gains from GPU-accelerated portfolio adjustments and risk assessment.
- Fraud detection benefits from rapid data analysis, while algorithmic trading gains speed.
- Credit risk assessment sees improvements in data processing, scoring model development, and real-time decision support.
- Regulatory compliance demands are met efficiently with NVIDIA GPU and CUDA, elevating banking computational prowess.

Conclusion:

This study highlights the pressing need for optimizing high-compute applications in financial computing, given the industry's heavy reliance on computational tools. Traditional applications built on outdated CPU technology have posed performance limitations, leading to the adoption of third-party grid applications. Unfortunately, many high-performance computing (HPC) applications in this sector lack support for advanced hardware technologies like GPUs and TPUs.

The study demonstrates the potential of Python, NVIDIA CUDA on AWS GPU instances to revolutionize financial computing. Modern GPUs offer an abundance of cores, an efficient memory hierarchy, and high memory bandwidth, contributing significantly to computational power. They excel in parallelism, data-parallel processing, and specialized hardware units, making them ideal for financial tasks.

Experimental results showcase the remarkable performance gains achieved through GPU acceleration, especially in matrix multiplication—a fundamental operation in financial computations. GPUs outperform CPUs in parallel processing, with increasing efficiency as data size grows.

Furthermore, the study outlines various banking use cases where Python, NVIDIA GPU and CUDA prove invaluable. From risk assessment to portfolio optimization, fraud detection to credit risk assessment, and regulatory compliance, these technologies empower financial institutions to elevate their computational prowess and maintain a competitive edge.

In conclusion, the adoption of GPUs and CUDA in financial computing represents a pivotal shift towards efficient, high-performance solutions. Embracing these technologies promises not only substantial performance improvements but also opens doors to innovative approaches in addressing the computational challenges of the financial sector.

Acknowledgments :

The authors of this technical paper would like to acknowledge the invaluable support and insights provided by the research and development teams at Amazon Corporation. Their contributions were instrumental in conducting experiments and gathering data for this study.

References :

Catanzaro B., Sundaram N., Keutzer K.: 'Fast support vector machine training and classification on graphics processors'. Int. Conf. on Machine Learning, Helsinki, Finland

Nurvitadhi E., Mishra A., Marr D.: 'A sparse matrix vector multiply accelerator for support vector machine'. IEEE Int. Conf. on Compilers, Architecture and Synthesis for Embedded Systems, Amsterdam, Netherlands

Osawa K., Sekiya A., Naganuma H. et al.: 'Accelerating matrix multiplication in deep learning by using low-rank approximation'. IEEE Int. Conf. on High Performance Computing & Simulation, Genoa, Italy

Očkay M.: 'General-purpose computing on GPU: pixel processing'. IEEE Communication and Information Technologies (KIT)

Suda R., Aoki T., Hirasawa S. et al.: 'Aspects of GPU for general purpose high performance computing'. IEEE Asia and South Pacific Design Automation Conf

Amazon G4dn Instances. Available online: Amazon G4dn Instances

Contact Information :

For inquiries related to this study/POC paper, please contact:

[Sandeep Kanao] [Sandeep.Kanao@gmail.com]

[Salabh Kumar Saxena] [Salabh.Saxena@gmail.com]

Note: The full contact information of the authors is provided but has been omitted for this public document.

Disclaimer: The experimental results presented in this technical paper are based on specific test scenarios and configurations. Actual performance may vary depending on the specific hardware, software, and workload conditions.