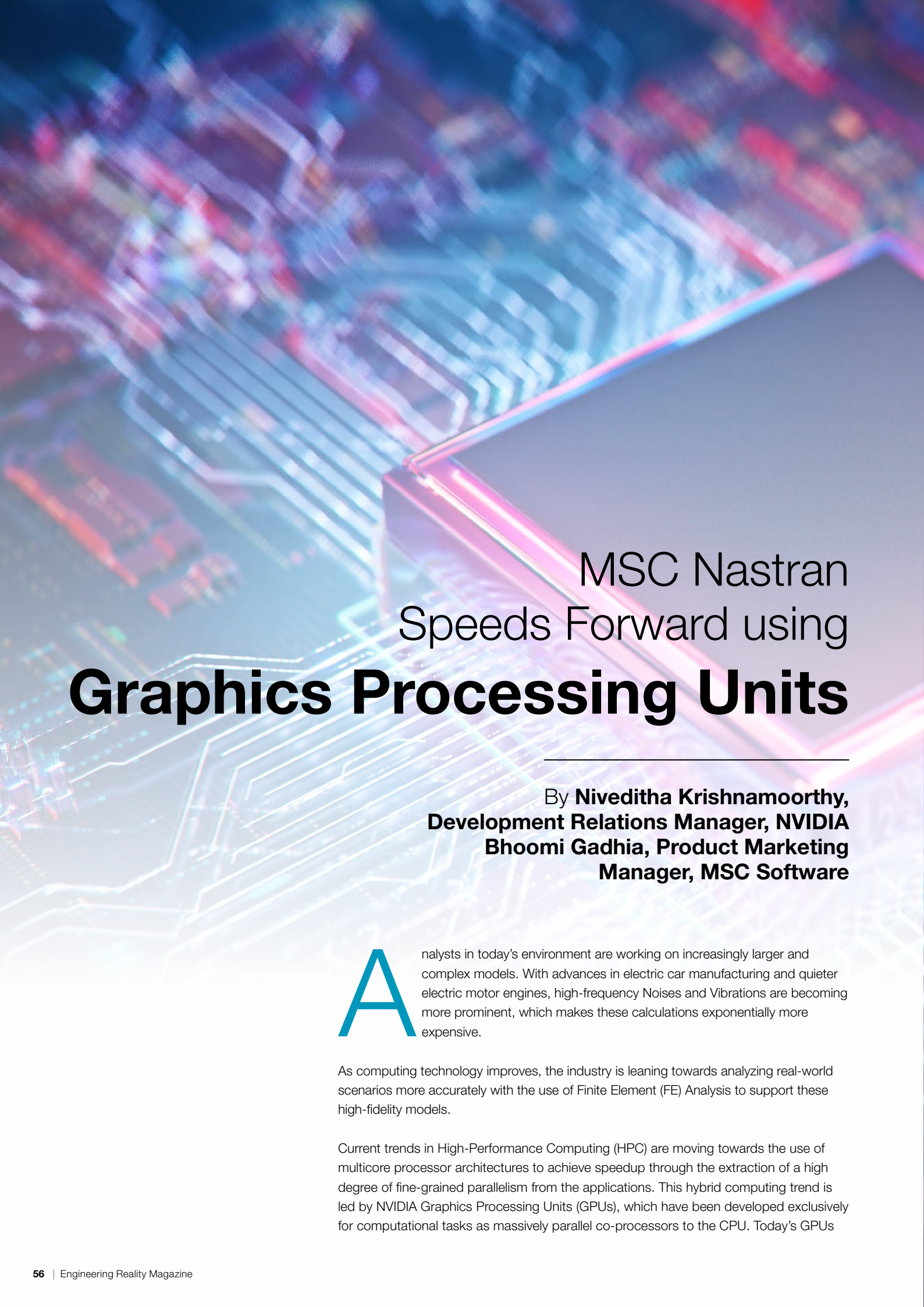


MSC Nastran Speeds Forward using **Graphics Processing Units**

By **Niveditha Krishnamoorthy, Development Relations Manager, NVIDIA**
Bhoomi Gadhia, Product Marketing Manager, MSC Software

Analysts in today's environment are working on increasingly larger and complex models. With advances in electric car manufacturing and quieter electric motor engines, high-frequency Noises and Vibrations are becoming more prominent, which makes these calculations exponentially more expensive.

As computing technology improves, the industry is leaning towards analyzing real-world scenarios more accurately with the use of Finite Element (FE) Analysis to support these high-fidelity models.

Current trends in High-Performance Computing (HPC) are moving towards the use of multicore processor architectures to achieve speedup through the extraction of a high degree of fine-grained parallelism from the applications. This hybrid computing trend is led by NVIDIA Graphics Processing Units (GPUs), which have been developed exclusively for computational tasks as massively parallel co-processors to the CPU. Today's GPUs

can provide memory bandwidth and floating-point performance that are several factors faster than the latest CPUs. To exploit this hybrid computing model and the massively parallel GPU architecture, application software will need to be redesigned.

MSC Software and NVIDIA engineers have been working together on the use of GPUs to accelerate the sparse direct solvers in MSC Nastran for the last several years. We will dive deeper into the recent GPU computing developments in MSC Nastran in this article, including the support of NVH solutions with Fast Frequency Response (FASTFR), and Matrix Multiply and Add (MPYAD) modules development. Representative industry examples will be presented to demonstrate the performance speedup resulting from GPU acceleration.

The resulting rapid CAE (Computer-Aided Engineering) simulation capability from GPUs has the potential to transform current practices in engineering analysis and design optimization procedures, enabling manufacturing OEMs to deliver their products to market at a much faster rate than is currently possible with CPU-based solutions.

Working jointly with NVIDIA provides MSC Software a comprehensive portfolio of enterprise-grade, high-performance systems, and software, with high-value services to help manufacturers implement MSC Nastran throughout the value chain and product lifecycle. Many of these systems are already certified by MSC Software and deliver unparalleled productivity, performance, and flexibility.

GPUs have been a significant part of HPC for decades, and MSC Nastran has been on the market for more than 50 years. There had been attempts to utilize GPUs in MSC Nastran in the past, which have not led to much success. One of the reasons was the architecture of MSC Nastran, which relied heavily on the disc storage of the data and having a minimum memory footprint. Another reason was the memory limitations and the overhead caused by the data transfer between the host and the GPUs. In the last few years, GPUs have accelerated performance – NVIDIA has added more cores and memory specifically for the FE applications. This has helped specifically with NVH simulations.

To accommodate the GPU usage, MSC Nastran has evolved to maximize memory usage, and the GPUs have become increasingly faster, and their on-board memory has reached the tens of gigabytes mark, they are finally ready for successful symbiosis.

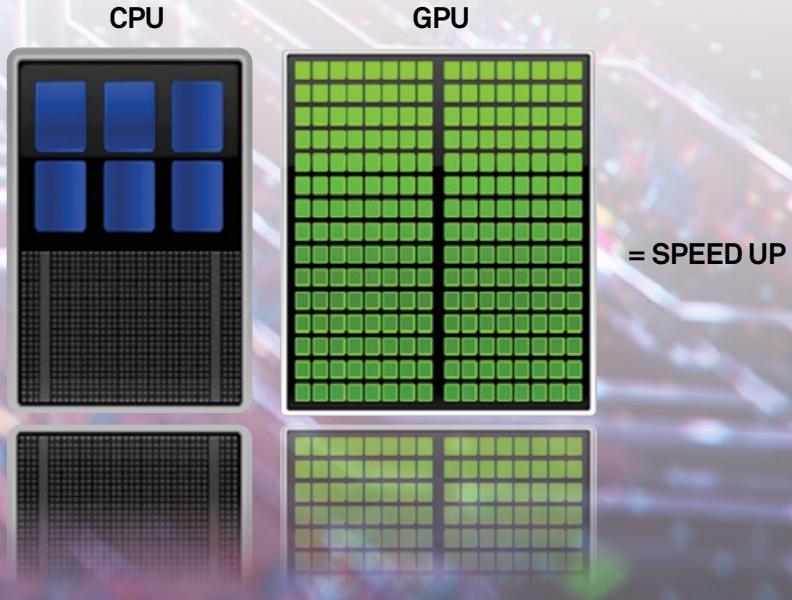
Why Use GPU?

Most customers ask why they should buy a GPU. It is generally beneficial when models are mainly composed of solid elements, and the simulation is dominated by matrix factorizations, not like in Normal Modes Analysis (SOL 103), where there are lots of modes, and FBS dominates. Furthermore, each user should evaluate whether multiple CPU cores can give them the same benefit as a GPU while keeping in mind that GPU usage can free up some of the CPU cores for use by other applications/users.

A primary use of GPU is to offload massive computational tasks from CPU. GPUs will not be useful for small models that operate on a small amount of data, do a large number of I/O operations, or iteratively solve small sets of equations. Every time a GPU is invoked, it comes at a fixed cost of, typically, a fraction of a second, 0.1 – 1 sec. So, if you are trying to use a GPU to compute a product of two 100 x 100 matrices, which takes 0.01 seconds on the CPU, you will pay a 0.1 – 1-second penalty for just using the GPU before you even start transferring the data between the host memory and the GPU. This will not benefit your simulation.

However, if you are dealing with a large model that requires operations on

Leveraging the GPUs for dynamic analysis using MSC Nastran 2019 at Volvo opens for substantial savings in run time (up to 50% faster) for the runs extending to high frequency.



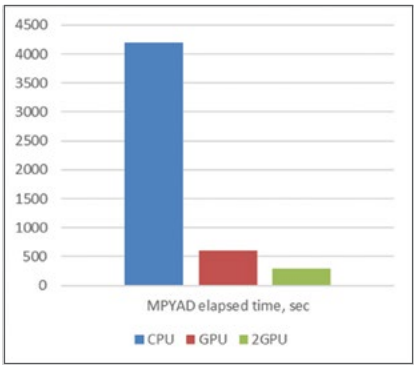


Figure 1

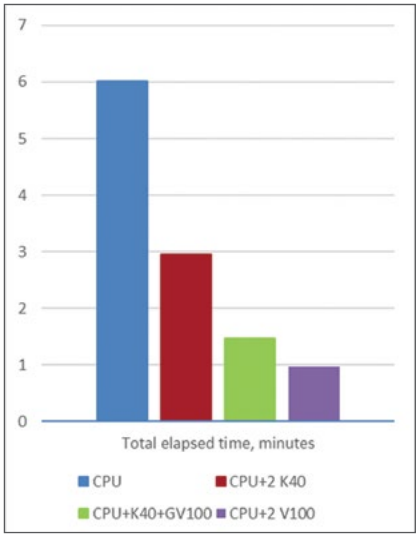


Figure 2

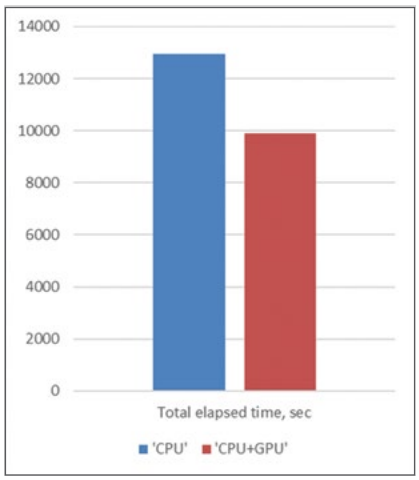


Figure 3

matrices with dimensions of the order of 10000 x 10000 or more, then the GPU will come in very handy. In fact, for dense matrix operations, every additional GPU is roughly equivalent to having an extra CPU with 20 cores. By splitting the workload between the CPU and one GPU, you can achieve approximately a factor 2x speedup. When splitting the workload between the CPU and two GPUs – factor 3x speedup, and so forth. For sparse matrix operations, every GPU is roughly ten times faster than the CPU, so all sparse BLAS operations should be handled by GPUs alone.

Similar to the CPUs, GPUs have multiple cores that can be used for parallel computations, and they also have parallel "streams" that can perform parallel data transfers between the memory of the host machine and the GPU memory. Every time a GPU is invoked in a computation, all the input matrices need to be copied from the host memory into the GPU memory, and the resulting matrix needs to be copied from the GPU back to the host. By using multiple streams in parallel, the cost of these data transfers can be significantly reduced. However, it would be best if you considered that each GPU stream could only communicate to a dedicated CPU core. So, when you are performing some computations involving dense matrices, precise balance must be reached between how many CPU cores are included in calculations, and how many are dedicated to communicating with the GPU streams.

Rule of the thumb is: the number of CPU threads/cores should always be larger than the number of GPUs you are trying to use. For example, in a scenario where you try to compute something using two GPUs and just four CPU cores, the workload will be divided between the GPUs because the number of the CPU cores is too small to compete with the GPUs. In a scenario where you try to use two GPUs and 20 cores, four cores will be assigned to each GPU for parallel streaming of data, and the remaining 12 cores will be used for the computation.

MSC Nastran GPU acceleration is user-transparent, meaning jobs launch and complete without additional user steps. Also, the schematic of a CPU with an attached GPU accelerator is such that CPU begins/ends job, GPU manages heavy computations. This gives unparalleled productivity and performance with NVIDIA GPUs.

Examples

When you deal with sparse matrices and sparse BLAS operations, the work partitioning logic is simpler, since all work will be handled by GPUs only. In case you are using one GPU, a typical large sparse BLAS operation will be about 3x – 10x times faster. Since most of these operations are scalable, the more GPUs you use, the greater the speedup. With two GPUs, you get 6x – 20x speedup, and so on. Specific speedup values are strongly dependent on the hardware, on the size of the input matrices, and the type of operations.

Figure 1 represents an example of performance improvement for dense-sparse matrix multiplication by adding one or two GPUs. Input matrix parameters: matrix A 56,000 x 192,000, 100% dense; matrix B 92,000 x 192,000, 1.5% dense. Blue bar: 20-core Intel Xeon CPU; red bar: single NVIDIA Tesla K40 GPU; green bar: two NVIDIA Tesla K40 GPUs.

Figure 2 represents performance improvement for a dense matrix multiplication benchmark: 50,000 x 50,000 dense matrices. Blue bar: 20-core Intel Xeon CPU; red bar: 20-core Intel Xeon CPU + two NVIDIA Tesla K40 GPU; green bar: 20-core Intel Xeon CPU + NVIDIA Tesla K40 + Quadro GV100 GPU; purple bar: 20-core Intel Xeon CPU + two Nvidia Tesla V100 GPU.

It is possible to use several different GPU models on the same host system, with varying sizes of memory and efficiencies. Every time Nastran uses GPUs for BLAS operations, it will check the elapsed times on each GPU and CPU and balance the workload in a way that would minimize the elapsed time across all devices. This way, the fastest GPU/CPU gets most workload, while the slowest GPU/CPU gets the least workload.

The current implementation of GPU support is using CUDA Toolkit by NVIDIA and only works with NVIDIA GPU devices. The threshold matrix size is set to 4000 for dense matrices and 500 for sparse matrices. Matrices are partitioned among the CPU and GPUs and, depending on the memory availability, may be processed in one or multiple passes on each device.

Below, we shall consider several examples from the automotive industry to highlight the usage of GPU accelerate the solution. All models are SOL111 with varying number of degrees-of-freedom (DOF), frequency range, number of eigenmodes, and load cases. Note how the performance improves with an increasing number of frequencies and eigenmodes. The reason for this is that most of the computationally intensive operations are proportional to the number of eigenmodes to the third power and the number of frequencies. This is where the GPU makes the impact. In all the simulations shown below, the GPUs were actually used only in a couple of modules. Yet they managed to cut down the entire simulation elapsed times by 15-35%, or between 0.4 – 4 hours.

Figure 3 shows an example of SOL111 performance improvement by adding a single GPU. The model: 20.5M DOF, 2050 frequencies, 30,000 eigenmodes, 4 right-hand sides. Blue bar: 20-core Intel Xeon CPU; red bar: 20-core Intel Xeon CPU + NVIDIA Tesla K40 GPU.

Figure 4 represents the car-body model with 43.5M DOF provided by Volvo Car Corporation used for benchmarking.

Andrzej Pietrzyk, Method Development for NVH CAE at Volvo Car Corporation, says, “Leveraging the GPUs for dynamic analysis using MSC Nastran 2019 at Volvo opens for substantial savings in run time (up to 50% faster) for the runs extending to high frequency. This is definitely a highly promising development path for MSC Nastran.”

Figure 5. Performance improvement for model from Fig.4 (1500 frequencies, 50,000 eigenmodes) by adding one or two GPUs. Blue bar: 20-core Intel Xeon CPU; red bar: 20-core Intel Xeon CPU + NVIDIA Tesla K40 GPU; green bar: 20-core Intel Xeon CPU + two NVIDIA Tesla K40 GPUs.

Figure 6. Performance improvement for model from Fig.4 (2000 frequencies, 73,600 eigenmodes) by adding two GPUs of different types. Blue bar: 20-core Intel Xeon CPU; red bar: 20-core Intel Xeon CPU + two NVIDIA Tesla K40 GPU; green bar: 20-core Intel Xeon CPU + two NVIDIA Tesla V100 GPU.

Key Takeaways

GPUs allow speeding up large computationally intensive MSC Nastran operations by factors between 2 and 20, compared to the elapsed time on CPU alone. Depending on the percentage of these operations in the total elapsed time of the entire simulation, the results obtained can be several times faster than a simulation that is run only on CPUs.

The larger the simulation, the larger the speedup. For the same model, the frequency response analysis over the frequency range of 1500 Hz and 50,000 modes improves from 11.3 hours to 7.2 hours by adding 2 GPUs. With the frequency range increased to 2000 Hz and 73,000 modes, the elapsed times on CPU alone and with two Tesla K40 GPUS are 30.3 and 18.1 hours, respectively.

Newer GPU models are several times faster than older models. For dense BLAS operations 5-year old Tesla K40 model is roughly equivalent to adding an extra CPU to the host machine. Adding a newer Quadro GV100 or Tesla V100 model is approximately equal to adding CPU that is 2-3 times faster. It is possible to split the workload dynamically between different GPU models and get the best performance out of all available devices. For sparse BLAS operations adding a Tesla K40 GPU leads to a speedup of 10x on average. Adding a Tesla V100 leads to a speedup of ~20x – 30x compared to the elapsed times for the same operations handled by Intel MKL kernels on CPU.

Acknowlegdgement: We would like to thank Illia Silin who contributed greatly towards this article.

Figure 4

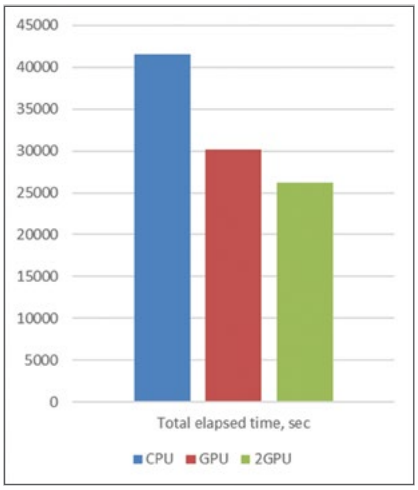


Figure 5

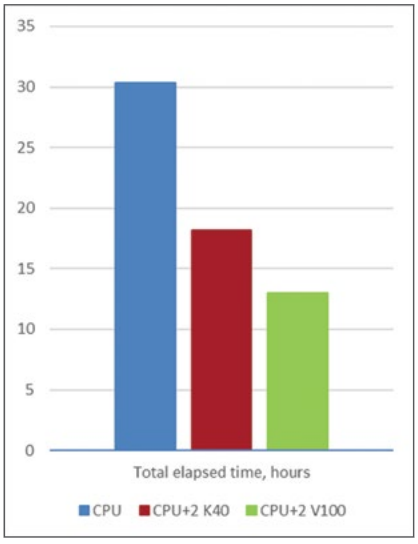


Figure 6