ACCELERATING ABAQUS COMPUTATIONS USING NVIDIA GPUS

WP-07843-001_v03 | August 2016

White Paper

TABLE OF CONTENTS

LIST OF FIGURES

INTRODUCTION

The Abaqus Unified Finite Element Analysis (FEA) tool offers powerful and complete solutions for both routine and sophisticated engineering problems covering a vast spectrum of industrial applications. Users run high fidelity simulations with the help of High Performance Computing (HPC) workstations and/or clusters, and are constantly demanding more parallel efficiency at lower costs and short turnaround time because of the ever-growing model sizes, more complex physics, and exploration of large design-space to find optimal designs. These are the key factors behind engineering decisions to develop efficient products in a short time.

HPC is available on-demand to solve complex compute-intensive problems, and there is a current move towards using GPUs with massively parallel processor architectures. These GPUs achieve faster performance through the extraction of a high degree of fine-grained parallelism from software applications, providing memory bandwidth and floating-point performance that are several factors faster than the latest CPUs. NVIDIA and SIMULIA — the Dassault Systèmes brand for Realistic Simulation — have collaborated to deliver the power of GPU computing for Abaqus users.

In this whitepaper, you will learn about supported Abaqus features for GPU acceleration, GPU activation steps, software licensing tokens, and how simulation computations can be sped-up significantly with reduced energy consumption. These benefits can help transform the product development processes, improving workflow productivity and efficiency.

ACCELERATED GPU COMPUTING

Parallel applications increasingly use GPUs to accelerate CPU computations. In this heterogeneous computing model, the GPU serves as a co-processor to the CPU. Several studies have demonstrated the performance gains that are possible by using GPUs. Independent Software Vendors (ISVs) have been able to demonstrate two-fold to three-fold computational gains over multi-core CPUs. This limit is to the result of the current focus on linear equation solvers for GPUs as opposed to complete GPU implementations. Roughly 50% of the total computation time of typical simulations can be spent on linear solvers. With each release, Abaqus is improving GPU performance and implementing new features.

Abaqus supernode-based direct sparse solver for implicit scheme simulations can be computationally intense, and computing requirements grow with increasing model size (degrees of freedom). In a supernodal solver, an elimination tree is built for the sparse matrix. The nodes in the tree are dense matrices. The factorization of the sparse matrix is based on the factorization of those dense matrices (fronts) and their assembly into super-nodes. Most of the runtime of a direct sparse solver is spent in the factorization of the frontal matrices and their assembly.

The typical strategy for developing a direct sparse solver for the GPU is to offload those dense matrix operations to the GPU. This strategy can typically be very efficient for a massively parallel GPU architecture. Other solver operations such as the matrix assembly, tree traversal, and forward-backward solve, can all stay on the CPU as they typically require only a small fraction of the total solution time.

Because matrix fronts are copied to the GPU one-by-one for computation and then copied back to the CPU afterwards, there are two potential performance bottlenecks:

- For finite element models with many “holes,” thin shells, or both, the fraction of the runtime for small fronts will increase. Generally, a GPU requires a dense matrix to be larger than 1K to reach peak performance. For many small matrices, this could require a significant fraction of the factorization phase and the overall performance

could be well below the peak GPU performance. Based on tests, we have seen computational speedups on models as small as 500, 000 degrees of freedom (DOF).

- Models that are relatively small may not gain speedup from GPUs because of communication overheads incurred in transferring matrices to and from CPUs. However, speedup is significant for models that contain more than a million DOF because the overhead is relatively small compared to the computing time in the solver.

The concurrent kernel feature in the GPUs provides a solution to a more general direct solver GPU approach that could benefit a broader range of model geometries. Because the matrix fronts on the same assembly tree level can be processed independently, their GPU kernels can run concurrently on the GPU, thereby fully using the hardware for peak performance.

ABAQUS GPU COMPUTING

Support for NVIDIA GPUs in the Abaqus Unified Finite Element Analysis (FEA) tool was added in Abaqus 6.11 and enhanced in later releases.

- **Abaqus 6.11:** SIMULIA began supporting NVIDIA GPUs, starting with a single GPU direct sparse solver for Abaqus/Standard computations.
- **Abaqus 6.12:** This release featured support for distributed memory parallel (DMP), multiple GPUs, and the flexibility to run Abaqus simulation jobs on specific GPUs.
- **Abaqus 6.13:** More GPU enhancements were made available with Unsymmetric sparse solver support.
- **Abaqus 6.14:**The DMP Split feature and Automatic Multi-level Substructuring (AMS) solver were added. AMS Eigensolver offers a performance boost for natural frequency extraction in full vehicle models that have a very large number of modes.
- **Abaqus 2016**: This release features GPU support for an enhanced AMS Eigensolver and a frequency response solver. Frequency response simulation is used to better understand the operation of a system or a vehicle under continuous harmonic loading. It uses results from a natural frequency analysis performed using a solver such as the AMS.

SUPPORTED ABAQUS FEATURES FOR GPU ACCELERATION

Abaqus/Standard employs solution technology ideal for static events and low-speed dynamic events where highly accurate stress solutions are critical. The following Abaqus/Standard features are supported and recommended on NVIDIA GPUs in Abaqus 2016:

- General Linear and Nonlinear Analyses
  - Static Stresses and Displacement
  - Dynamic Stress and Displacement
  - Steady State and Transient Heat Transfer
  - Multi-physics Analysis
    - Thermo-electrical-structural
    - Pore-fluid-flow-mechanical-thermal
- Linear Perturbation Analysis
  - Static Stress and Displacement
    - Linear static
  - Dynamic Stress and Displacement
    - Steady state dynamics (direct)
- Solution Techniques
  - Parallel execution on both shared memory and distributed memory parallel cluster systems
  - Parallel direct sparse solver with dynamic load balancing
  - Accelerated DMP Execution mode (also called DMP Split) — an optional feature that delivers good performance
  - Parallel AMS Eigensolver — an optional add-on for Abaqus/Standard that has a significant performance advantage over Lanczos

ACTIVATING THE GPU FEATURE

The `gpus`, `threads`, and `mp_host_split` flags

To run the Abaqus Simulation on GPUs, the `-gpus` flag must be included in the command line. For example:

```
$Abaqus_2016 -interactive -job jobname -input inputfile.inp -cpus
8 -gpus 1 >& outputfile.log
```

From Release 6.14 onwards, the DMP split feature (DMP and SMP) can be combined with GPU acceleration by adding the `-threads` flag or the `-mp_host_split` flag with the `-gpus` flag.

- If the `-threads` flag is used, the Abaqus driver takes `-cpus N` `-threads M` to create *N*/*M* DMP process. For example:

  ```
  $Abaqus_2016 -interactive -job jobname -input inputfile.inp -cpus 32
  -threads 16 –gpus 1 >& outputfile.log
  ```

  In this example, *N*=32 and *M*=16. Therefore, the Abaqus driver creates two DMP processes, and each DMP process has 16 threads.

  The `-gpus N` parameter is per MPI (or DMP) process. Therefore, `-gpus 1` means that each MPI DMP process will run on 1 GPU to give a total of 2 GPUs for a job that has 2 DMP processes.

- If the `-mp_host_split` flag is used instead of the `-threads` flag, the number of DMP processes per node is specified directly as an argument to the `-mp_host_split` flag. For example:

  ```
  $Abaqus_2016 -interactive -job jobname -input inputfile.inp -cpus 32
  -mp_host_split 2 –gpus 1 >& outputfile.log
  ```

GPUs in exclusive mode

In Abaqus version 2016, it is no longer necessary to set the GPUs in exclusive mode.

However, it is always a good practice to check if GPUs are over-subscribed when multiple Abaqus jobs are running. If so, set the GPUs in exclusive mode for the DMP processes to go to separate GPUs. The GPUs are set in exclusive mode by running the following `nvidia-smi` command:

```
$nvidia-smi -c 3
```

Recommended configuration to run Abaqus jobs on GPUs

With a two-CPU socket machine, create two DMP processes and use two GPUs —one GPU for each DMP process. In addition, place a local `abaqus_v6.env` file with the following contents in the `project/run` directory to override and specify some commands that are specific to DMP and GPUs.

```
# Comment out if not running DMP Split per host
memory='40%'
ask_delete=OFF
# Modify the Host List based on the number of Compute Nodes Used and
specify the CPU cores per node accordingly
mp_host_list = [['NODENAME_1',16],['NODENAME_1',16]]
```

BENEFITS OF USING NVIDIA GPUS WITH ABAQUS

ABAQUS LICENSING TOKENS — COST SAVINGS

Abaqus licensing is token-based so that it is flexible enough for users to run a variety of Abaqus analyses, and use Abaqus/CAE for building simulation models or for looking at results. The number of tokens is calculated from the number of CPU cores used for the simulation run. Abaqus uses the following decaying function formula to determine the number of tokens. In this formula, N is the number of CPU cores.

$$Number\ of\ Tokens\ =\ INT\ (5 * N\text{^}0.422)$$

The graph in Figure 1 illustrates this equation, showing how the number of Abaqus licensing tokens increases as the number of cores increases.

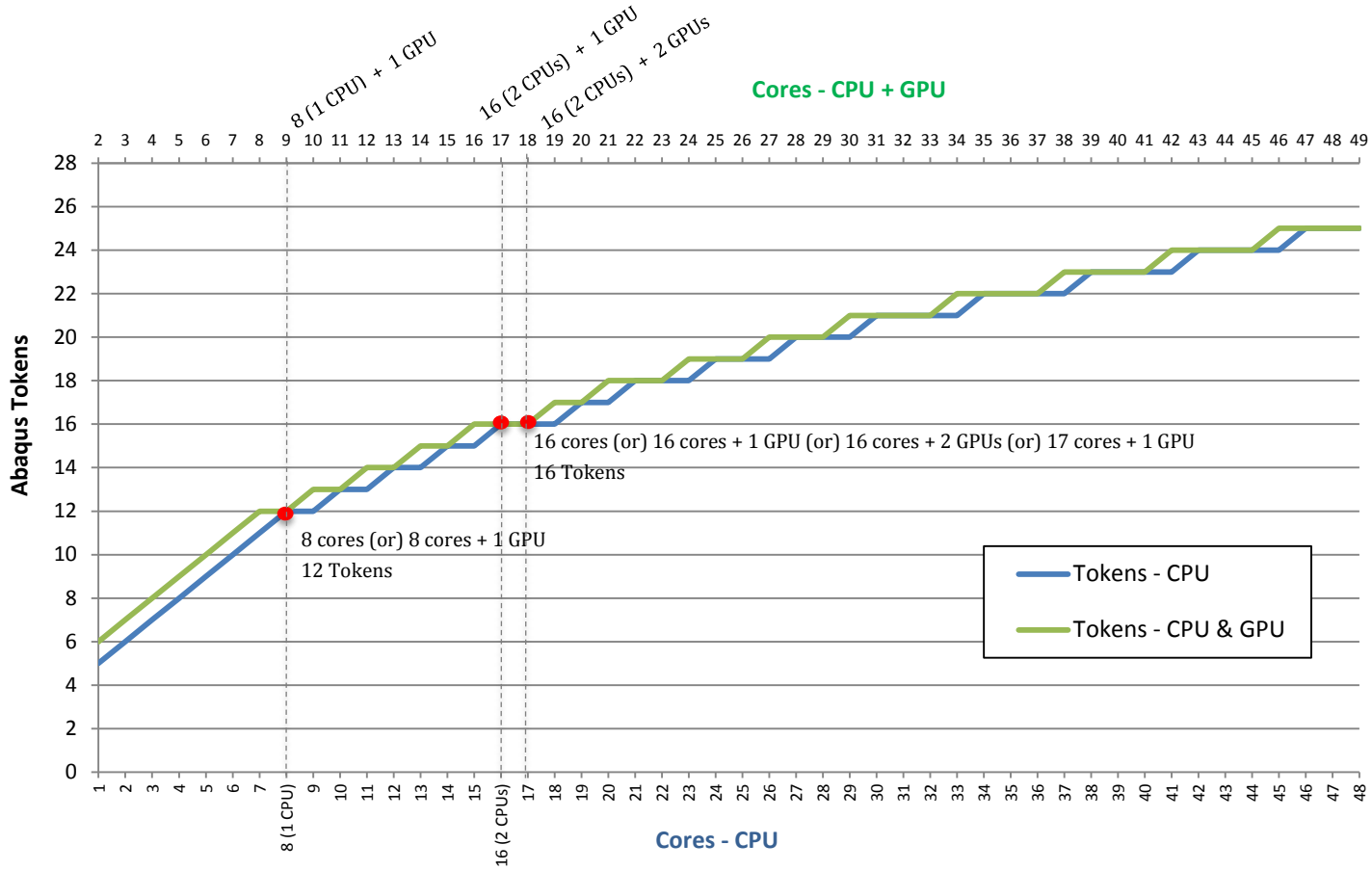


Figure 1 Cores against Abaqus Licensing Tokens for CPUs and GPUs

The blue curve represents the CPU-only case, while the green curve represents the CPU and GPU case. The staircase pattern in these curves shows how the increase in the required number of tokens decays as the number of CPU cores increases.

When a GPU is included in the simulation, it is counted as a single CPU core for the purposes of calculating the number of required tokens. This way of counting a GPU is represented by the green staircase-patterned curve with the corresponding CPU+GPU core count in the secondary X-axis on the top.

The cost benefit of using GPUs for simulations is illustrated by the two sets of computing configurations indicated by the dotted lines in Figure :

- The first dotted line shown at the 8-core mark on the primary X-axis indicates that for 8 CPU cores, 12 tokens are required.

  If a GPU is included in the simulation run, the CPU core count is 9 but the number of tokens remains at 12, as shown by the single red dot.

- The second dotted line shown at the 16-core mark on the primary X-axis indicates that for 16 CPU cores, 16 tokens are required.

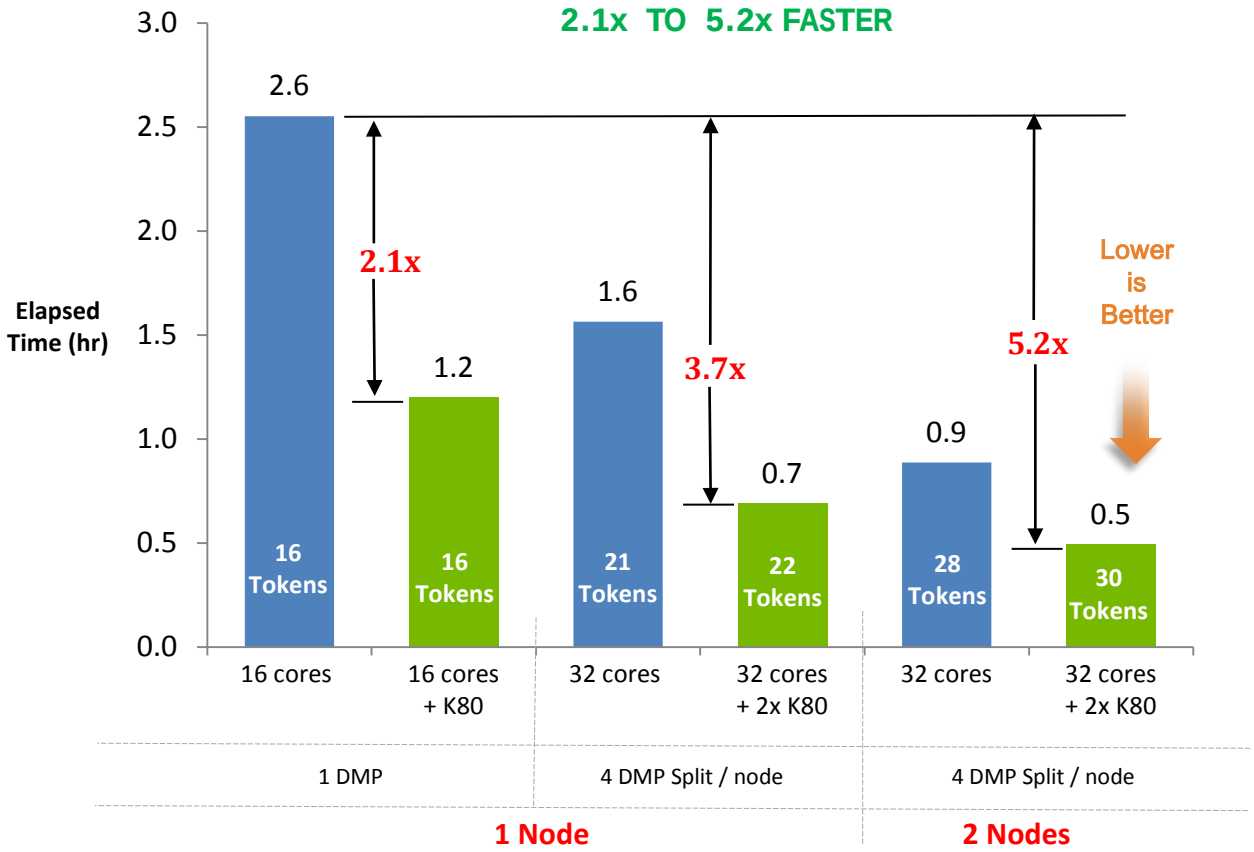
  Adding 1 or 2 GPUs to 16 CPU cores increases the CPU core count to 17 or 18 respectively, but the number of tokens would remain at 16, as shown by the pair of red dots.

The staircase pattern shows wider steps with increasing cores for both curves, highlighting the fact that it is cost-effective when more CPU cores are used. The benefit is much higher when GPUs are used instead of additional CPUs.

ABAQUS GPU PERFORMANCE — MORE DESIGN SCENARIOS AND OPTIMIZATION

To study the performance of NVIDIA GPUs, a variety of computing configurations were used on a typical industry model representing about 77 TFLOPs (Floating Point Operations). The results of the study are shown in Figure 2, Figure 3, Figure 4, and Figure 5.

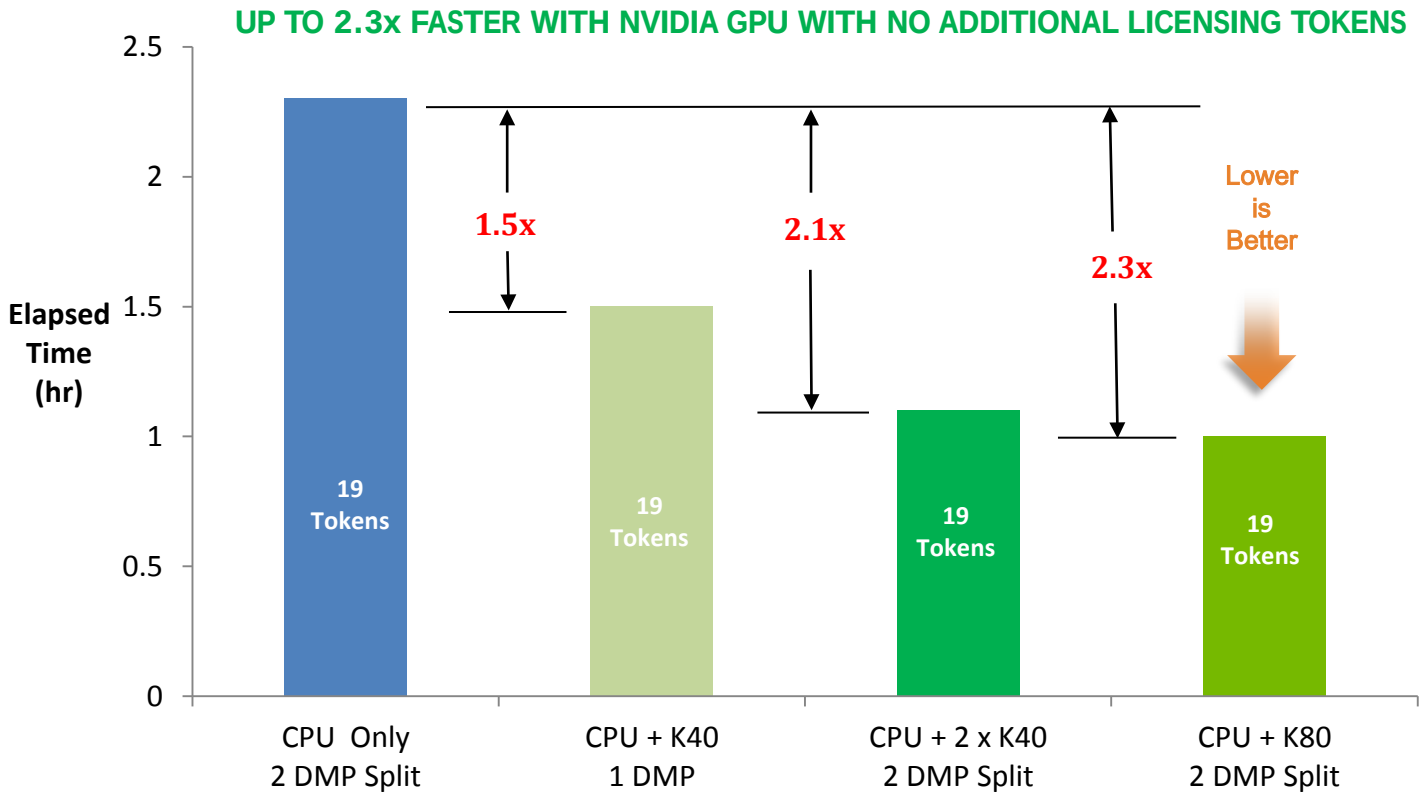
The bar chart in Figure 2 shows the benefit of using GPUs that makes use of the DMP split enhancement for Abaqus computations. The configuration on the left shows that almost four hours can be saved in an eight-hour day when running three similar jobs with GPUs without consuming extra licenses, while the configuration on the middle and right shows that the computing time can be reduced further when two or four GPU boards are used with just a few additional tokens.



~77 TFLOPs, 4.71M DOF, Nonlinear Static, Direct Sparse Solver (Model courtesy: Rolls-Royce)
Abaqus 2016 with 2 x Intel Xeon E5-2698v3, 2.30 GHz CPU, 32 CPU cores, 256GB memory, Tesla K80 Boards

Figure 2 Abaqus Acceleration on NVIDIA GPUs with DMP Split

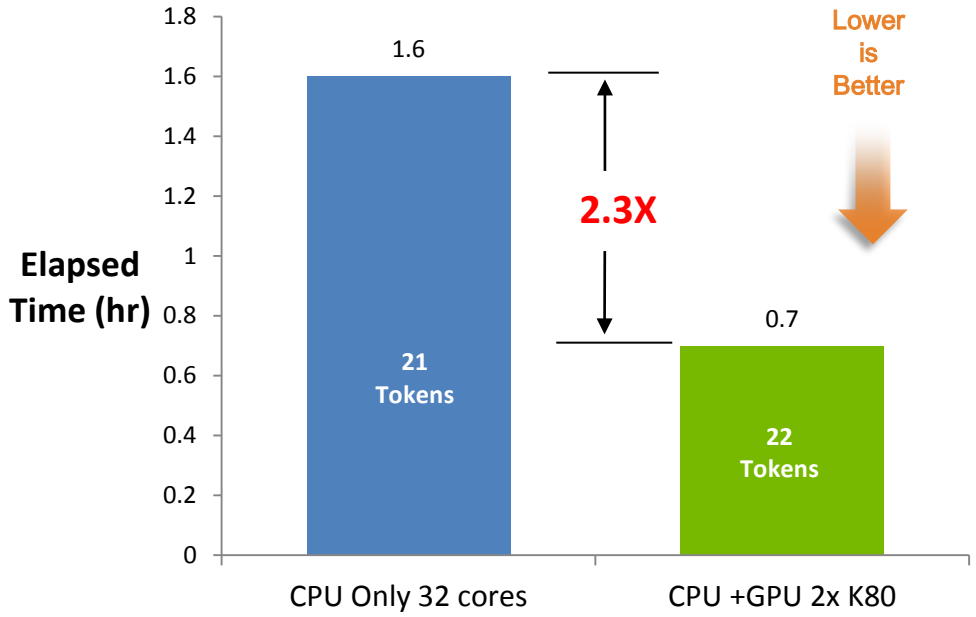
Figure 3 shows the value of Tesla K80 over the Tesla K40 when compared with the CPU-only configuration. In all scenarios, good speedups were observed without the need for additional licensing tokens. Also, K80 offers more speedups than other configurations. The best-case scenario was taken into account for the CPU-only run by having the two DMP split option.



~77 TFLOPs, 4.71M DOF, Nonlinear Static, Direct Sparse Solver (Model courtesy: Rolls-Royce)
Abaqus 6.14-2 with 2 x Intel Xeon E5-2697v2, 2.70 GHz CPU, 24 CPU cores, 128 GB memory, Tesla Boards

Figure 3 NVIDIA Tesla K40s vs. Tesla K80 over CPU

Figure 4 shows that adding two NVIDIA Tesla K80 GPU boards to a dual-socket Haswell system accelerates computations by factor of 2.3 while consuming just one additional licensing token.



~77 TFLOPs, 4.71M DOF, Nonlinear Static, Direct Sparse Solver, 4 DMP – Split (Model courtesy: Rolls-Royce)
Abaqus 2016 with 2 x Intel Xeon E5-2698v3, 2.30 GHz CPU, 32 CPU cores, 256GB memory, Tesla K80 Boards

Figure 4 NVIDIA Tesla K80 Acceleration on a Dual-Socket Haswell System

Figure 5 shows that the additional investment in GPUs and software licensing provides a five-fold increase in the benefit-cost ratio, indicating a high return on investment (ROI). The CPU-only solution cost is approximated and includes both hardware and paid-up software license costs. Productivity is based on the number of completed Abaqus jobs per day.

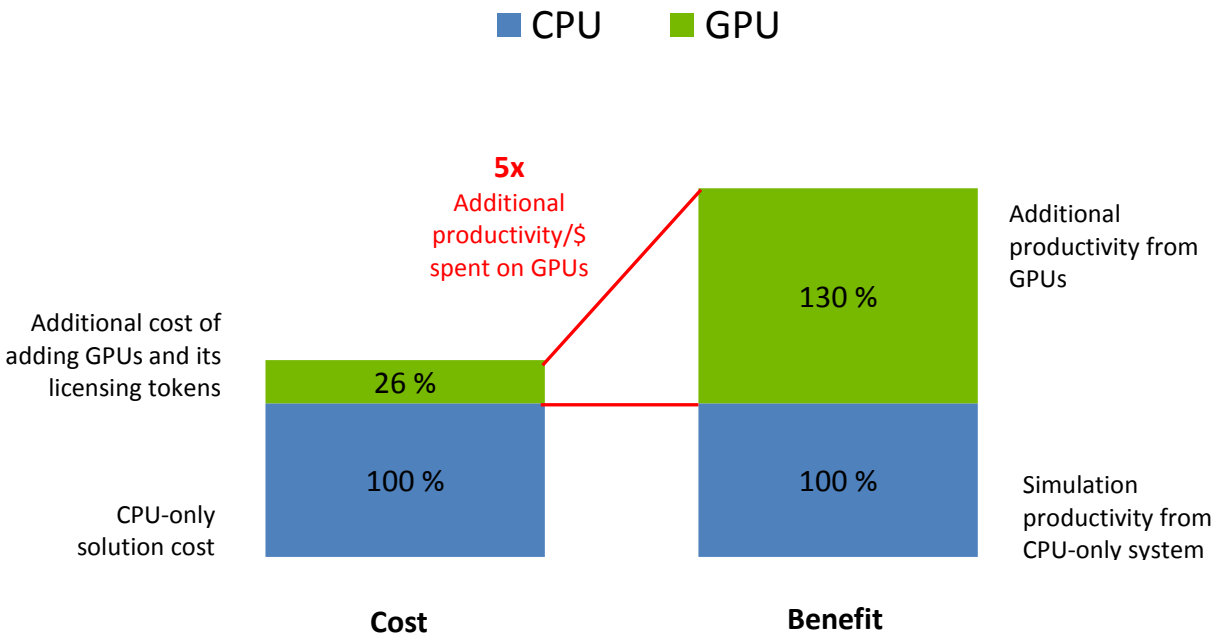
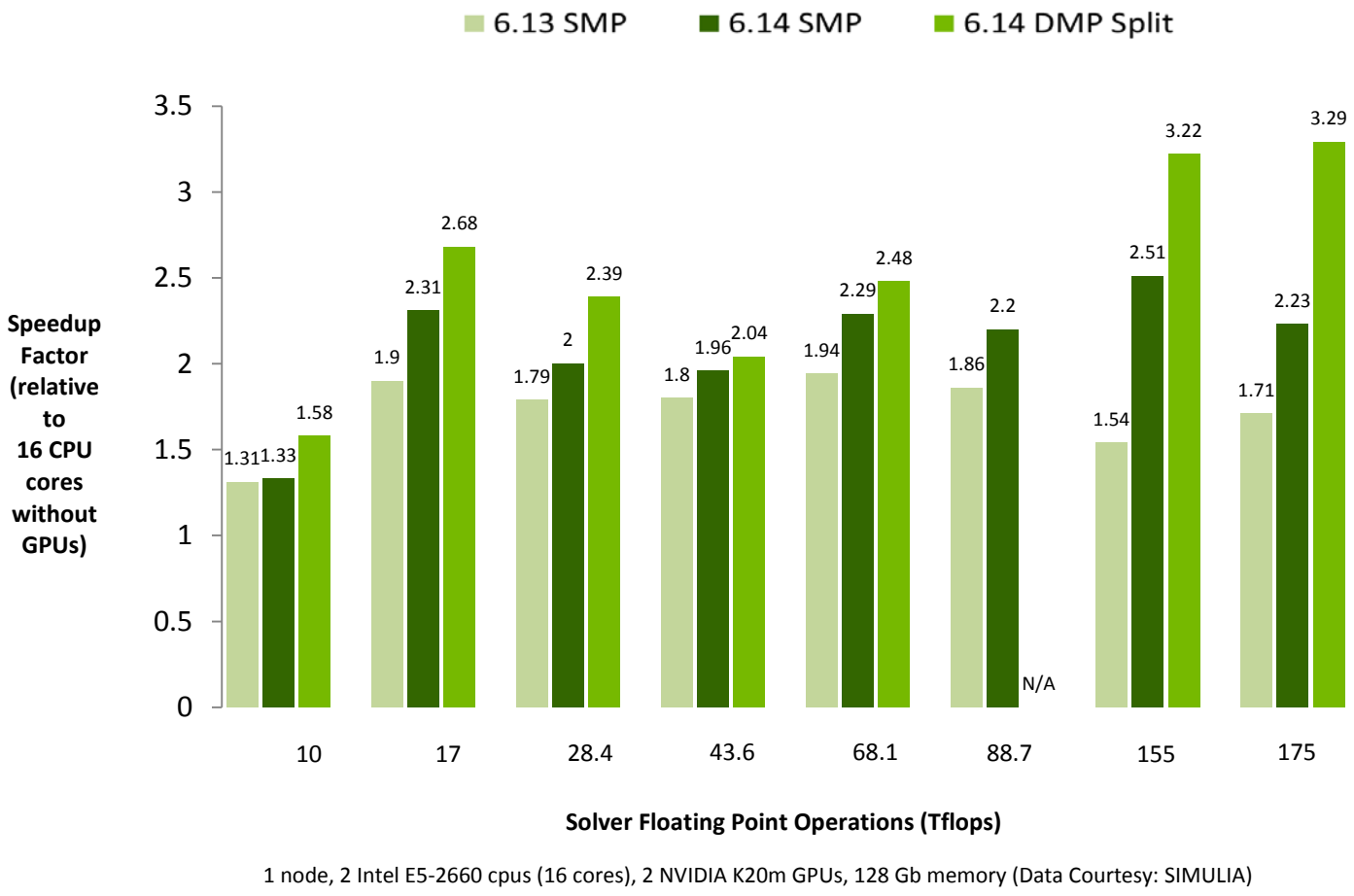


Figure 5 Productivity/Dollar ROI on NVIDIA GPUs

The preceding scenarios demonstrate that GPUs provide better cost-benefit ratios as they drastically reduce computation time while consuming either no extra licenses or just one additional licensing token.

Figure 6 shows the GPU speed improvements for the Abaqus 6.14 Release with the DMP split option compared with the 6.13 Release and 6.14 Release with SMP Option for a broad range of industry models whose sizes are specified in solver floating point operations (TFLOPs).

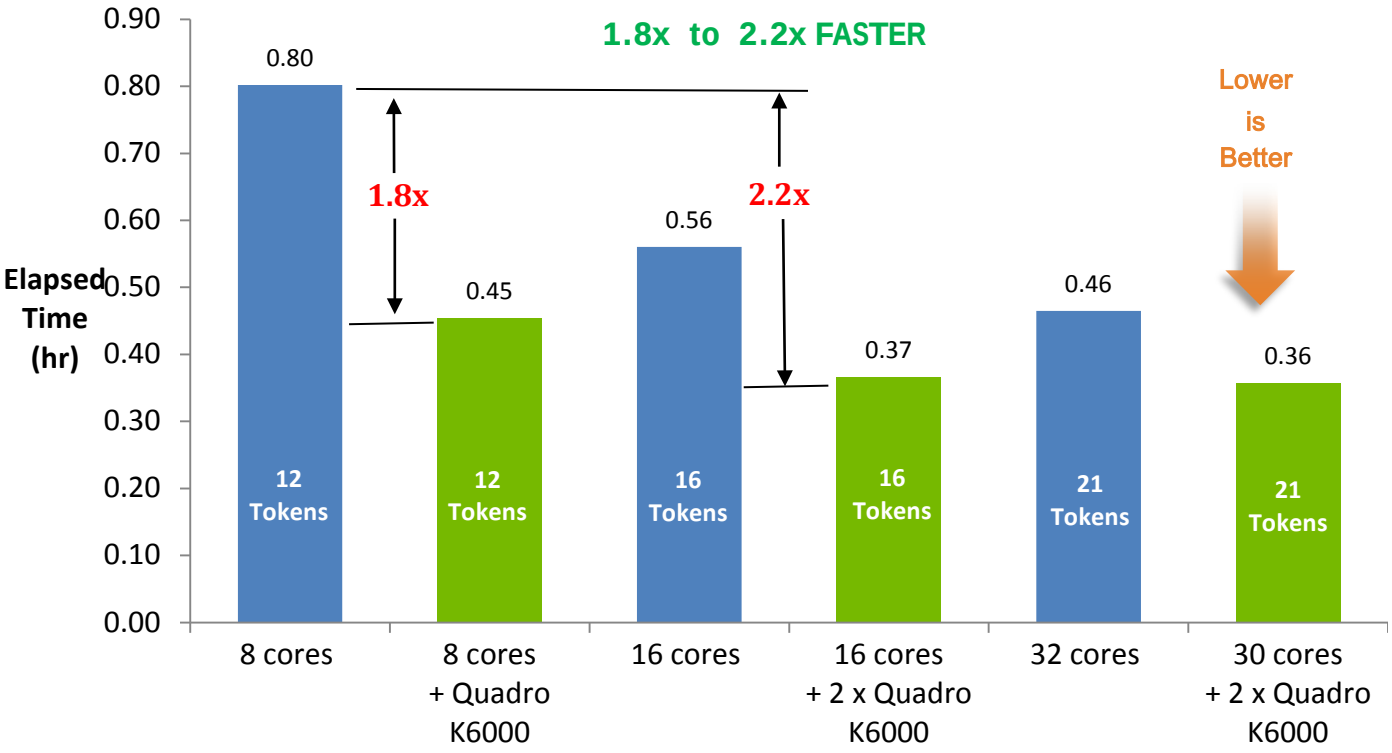
The bars on the Y-axis denote the GPU speedup factor relative to the 16-core CPU-only run. The 175 TFLOPs model example shows the relative performances where the Abaqus 6.14 Release with DMP option has a significant performance advantage over the previous release.



1 node, 2 Intel E5-2660 cpus (16 cores), 2 NVIDIA K20m GPUs, 128 Gb memory (Data Courtesy: SIMULIA)

Figure 6 NVIDIA GPU Speedup Improvements in Release 6.14

Figure 7 shows the Quadro K6000 GPU speedup of the 2016 Abaqus/AMS Eigensolver.



8.2M DOF, AMS Eigensolver for Structural Domain (Model: S3E)
Abaqus 2016 with 2 x Intel Xeon E5-2698v3, 2.30 GHz CPU, 32 CPU cores, 256GB memory, Quadro K6000 boards

Figure 7 NVIDIA Quadro K6000 GPU Performance on Abaqus/AMS 2016

Figure 8 shows the performance of NVIDIA GPUs on six compute nodes on a very large 43M DOF automotive model. Accelerated DMP execution mode with GPUs offers increased speedups. In this case, two message passing interface (MPI) processes are running on each node. Each process goes to a GPU.

Adding 12 Tesla K20m GPUs to a setup with 96 CPUs increases speed by a factor of 1.44 while consuming only two additional Abaqus license tokens. Additional benchmarks have shown that a GPU-based six-node compute cluster completes the simulation run in the same time as a CPU-only cluster of 12-nodes.

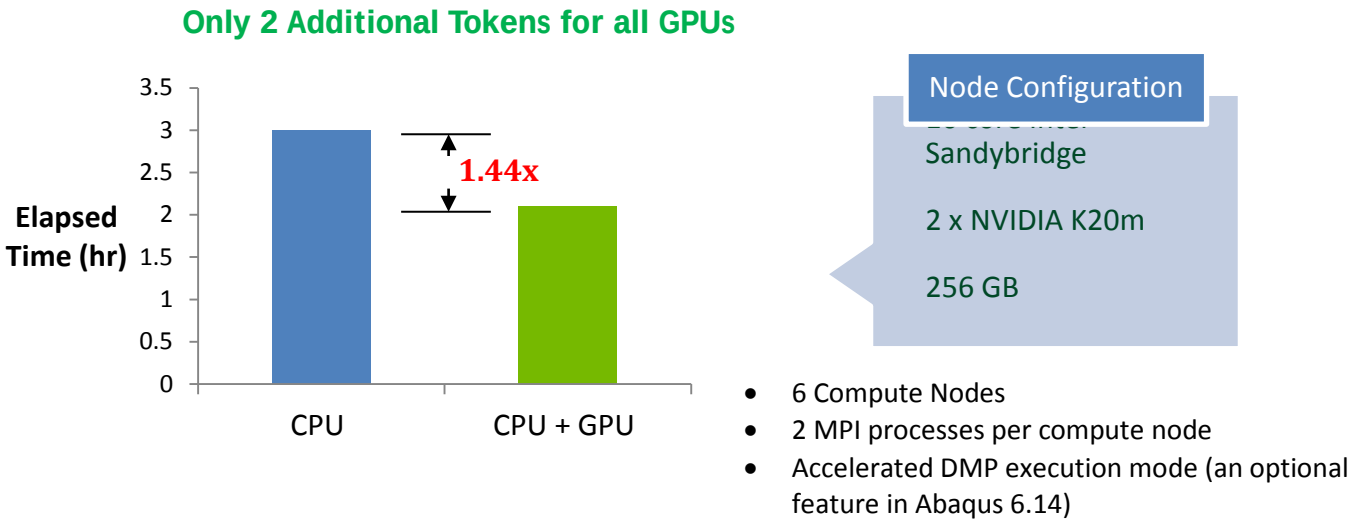


Figure 8 NVIDIA GPU Scalability Performance on 6 Compute Nodes

ABAQUS ENERGY CONSUMPTION — ENERGY COMPLIANCE & COST SAVINGS

Enterprises running simulations in large clusters want to drive down energy consumption to reduce costs and meet broader corporate sustainability initiatives. At the same time, researchers and engineers demand high levels of computing power to model complex simulations and explore large design spaces. NVIDIA GPUs can satisfy these apparently conflicting requirements because they are optimized for higher throughput and performance per watt. In fact, large installations of GPUs are typically included in supercomputers to manage energy costs.

Figure 9 and Figure 10 show the instantaneous power consumption and energy consumption for the Rolls- Royce engine model. The instantaneous power drawn by a 24-core CPU-only system was compared with a CPU + GPU system doing the same job. The CPU-only system drew 411 watts on average over a period of 8200 seconds, totaling 953 watt-hours. Although the CPU + GPU system drew an average of 513 watts, the job was completed in just 3901 seconds because of acceleration. Therefore, the CPU + GPU system consumed only 555 watt-hours in total. Compared with the CPU-only system, the GPU+CPU system resulted in a 42 percent savings in energy use. This reduction in energy use is of paramount importance for enterprises looking for energy compliance and cost savings.

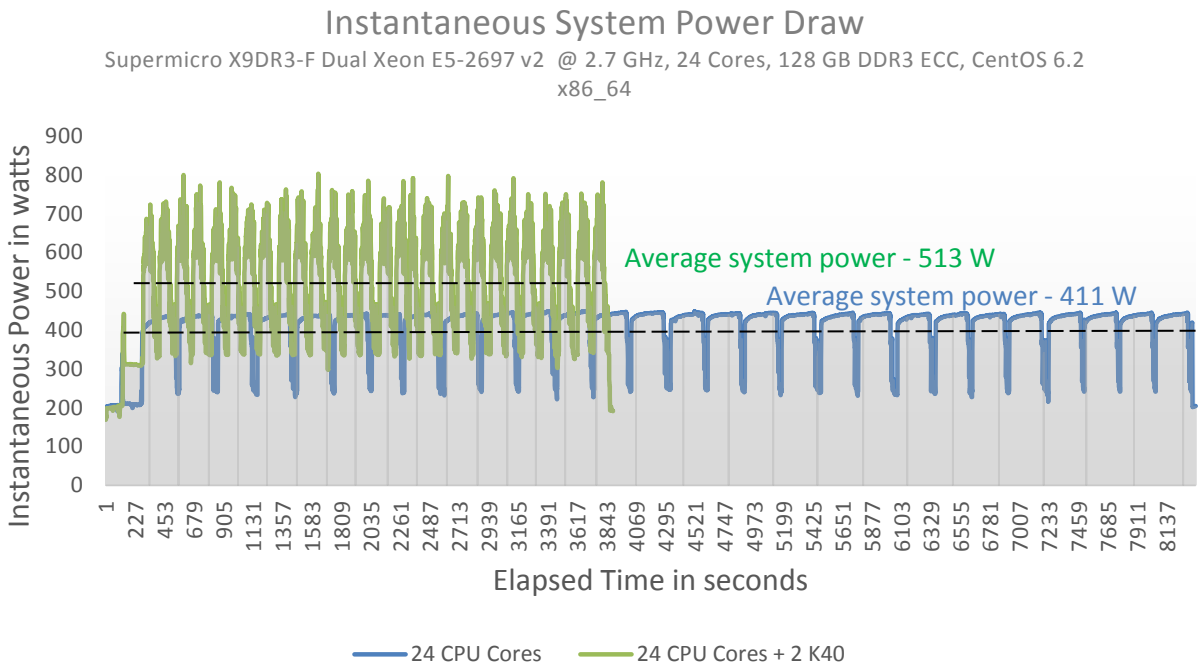


Figure 9 Instantaneous System Power Draw for CPU vs. CPU + Tesla K40 GPUs

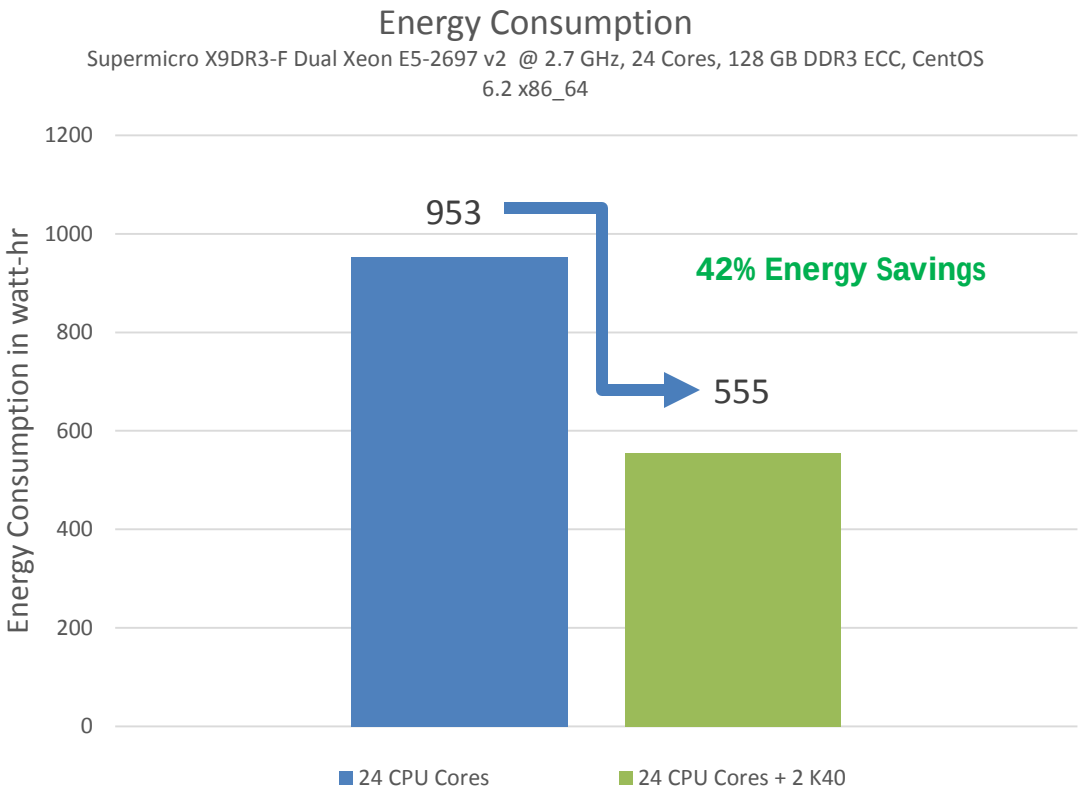


Figure 10 System Energy Consumption for CPU vs. CPU + Tesla K40 GPUs

HARDWARE REQUIREMENTS

The hardware requirements for accelerating Abaqus computations using NVIDIA GPUs are as follows:

- Two (2) socket CPU system with enough RAM to perform in-core simulations
- NVIDIA Tesla GPUs for servers and workstations
- NVIDIA Quadro GPUs for workstations

Recommendations:

- Configure Tesla GPUs such as the Tesla K20, K40 or K80 or a high-end Quadro K6000.
- Use GPU boards with 12 GB to 24 GB of memory per GPU and high double-precision capacity.
- Avoid using GeForce® GPUs and gaming class cards for accelerating Abaqus computations.

Table 1 lists the specifications for the NVIDIA GPUs that can be used for Abaqus computations.

Table 1 NVIDIA GPU Specifications

Features	Tesla K80	Tesla K40	Tesla K20	Tesla K20X	Quadro K6000
GPU	2× Kepler GK210	1 Kepler GK110B	1 Kepler GK110	1 Kepler GK110	1 Kepler GK110B
Peak double-precision floating point performance	2.91 Tflops (Boost Clocks) 1.87 Tflops (Base Clocks)	1.66 Tflops (Boost Clocks) 1.43 Tflops (Base Clocks)	1.17 Tflops	1.31 Tflops	1.7 Tflops
Peak single-precision floating point performance	8.74 Tflops (Boost Clocks) 5.6 Tflops (Base Clocks)	5 Tflops (Boost Clocks) 4.29 Tflops (Base Clocks)	3.52 Tflops	3.95 Tflops	5.2 Tflops
Memory bandwidth	480 GB/sec (240 GB/sec per GPU)	288 GB/sec	208 GB/sec	250 GB/sec	288 GB/sec
Memory size (GDDR5)	24 GB (12GB per GPU)	12 GB	5 GB	6 GB	12 GB
CUDA cores	4992 (2496 per GPU)	2880	2496	2688	2880

CONCLUSION

The performance numbers in the examples presented come from benchmark tests conducted by NVIDIA, or SIMULIA, or both in conjunction with our customers. Although the performance is dependent on the particular problem at hand, we have often attained two-fold to three-fold GPU acceleration, either with no additional Abaqus licensing tokens or with just one or two additional tokens, while saving energy consumption by about 40%.

In summary:

- GPU capability helps engineers to perform more high fidelity simulations in a day, providing more job throughput.
- The Abaqus GPU licensing model, when coupled with NVIDIA GPUs, offers more productivity per dollar investment.
- Use of GPUs in a system with CPUs leads to less energy consumption for running simulations.

With products becoming increasingly complex, manufacturers can rely on NVIDIA solutions to create innovative, higher quality products, and get to market faster.

REFERENCES

1. V. Belsky, et. al., Accelerating Commercial Linear Dynamic And Nonlinear Implicit FEA Software Through High-Performance Computing, Proceeding of NAFEMS 2011, Boston, USA, May 2011.
2. S. Posey and P. Wang, GPU Progress in Sparse Matrix Solvers for Applications in Computational Mechanics, Proceedings of 50th Aerospace Sciences Meeting, AIAA, Nashville, TN, January 2012.
3. G. Shyu, G. Deng, and S. Kodiyalam, GPU Computing with Abaqus 6.11 and its Application in Oil & Gas Industry, SIMULIA Community Conference 2012 Proceedings, Dassault Systemes SIMULIA, Providence, RI, April 2012.

Notice

Trademarks

Copyright

WP-07843-001_v03