Achieving Higher Productivity on Abaqus Simulations with HPC Clustering Technologies

Pak Lui, Gilad Shainer, David Cho, Scot Schultz, Richard Graham

HPC Advisory Council

***Abstract**: Engineers from wide ranges of industries face an ever increasing need to run virtual tests to simulate complex models for improving reliability as well as reducing product development time and costs. Abaqus Unified FEA is designed to tackle these challenges by producing high-quality realistic simulation solutions, while delivering high performance in productivity by utilizing efficient use of modern compute cluster available in High Performance Computing (HPC). Many organizations can now deploy with HPC cluster to process such complex analyses on Abaqus with computer cluster to render such high-quality models, while reducing simulation time from days to just hours. Behind this type of computational and clustering technologies that enables Abaqus to perform, it involves complex calculations and data exchanges among computational systems. The HPC Advisory Council has performed a deep investigation on Abaqus to evaluate its performance and productivity capabilities and to explore potential optimizations. This study presents the techniques and profiling results to further understand Abaqus dependencies on the CPUs, network, and the other underlying hardware and software components. The paper will review the effects by comparing various components using different simulation models on Abaqus.*

***Keywords**: Abaqus/Standard, Abaqus/Explicit, clusters, High Performance Computing, HPC, Benchmarking, Ethernet, Tuning, CPU, Power, network, InfiniBand, Turbo Mode, MPI Profiling, IO Profiling*

1. Introduction

The HPC Advisory Council has performed a deep investigation on Abaqus Unified FEA to evaluate its performance and productivity capabilities and to explore potential optimizations. This study presents the techniques and profiling results to further understand Abaqus dependencies on the CPUs, network, IO, and the other underlying hardware and software components. The paper will review the effects by comparing various components using different simulation models on Abaqus.

1.1 Abaqus Unified FEA

Abaqus Unified FEA product suite offers powerful and complete solutions for both routine and sophisticated engineering problems covering a vast spectrum of industrial applications. The Abaqus analysis product provides tools for simulating nonlinear finite element analysis (FEA), advanced linear and dynamics application problems. Abaqus/Standard is a General-purpose FEA

that includes broad range of analysis capabilities. Abaqus/Explicit is a nonlinear, transient, dynamic analysis of solids and structures using explicit time integration.

1.2 High Performance Computing

High-performance computing (HPC) is a crucial tool for engineering design and manufacturing. It is used for computer-aided engineering (CAE) from component-level to full vehicle analyses: crash simulations, structure integrity, thermal management, climate control, engine modeling, exhaust, acoustics and much more.
HPC helps drive faster time-to-market, significant cost reductions, and tremendous flexibility. The strength in HPC is the ability to achieve best sustained performance by driving the CPU performance towards its limits. The motivation for high-performance computing in the automotive industry has long been its tremendous cost savings and product improvements – the cost of a high-performance compute cluster can be just a fraction of the price of a single crash test, and the same cluster can serve as the platform for every test simulation going forward.

The recent trends in cluster environments, such as multi-core CPUs, GPUs, and new interconnect speeds and offloading capabilities are changing the dynamics of clustered-based simulations. Software applications are being reshaped for higher parallelism and multi-threading, and hardware is being configured for solving the new emerging bottlenecks, in order to maintain high scalability and efficiency.

1.3 HPC Cluster

Abaqus simulations can be carried out on a single server system or a cluster of compute servers known as high-performance computing (HPC) cluster. The HPC cluster is based on industry-standard hardware connected by a private high-speed network. The main benefits of clusters are affordability, flexibility, availability, high-performance and scalability. A cluster uses the aggregated power of compute server nodes to form a high-performance solution for parallel applications such as Abaqus. When more compute power is needed, it can sometimes be achieved simply by adding more server nodes to the cluster.

The areas which HPC clusters are architected have a huge influence on the overall application performance and productivity are the number of CPUs, usage of GPUs, the storage solution and the cluster interconnect.

In the subsequent sections of this paper, we are going to explore each of those components of a cluster that can enable higher performance.

2. Base Configuration

The configuration that will be used to demonstrate the performance differences are as follows, unless otherwise stated.

2.1 Hardware Configuration

The current study was conducted at the HPC Advisory Council Cluster Center [1] on the cluster comprised of HP ProLiant SL230s Gen8 4-node cluster, each node with Dual Socket Intel® Xeon® 10-core CPUs E5-2680 v2 at 2.80 GHz, Mellanox Connect-IB 56Gb/s FDR InfiniBand adapter, and with 64GB of 1600MHz DDR3 memory. The nodes were connected into a network

using a Mellanox SwitchX® SX6036 36-Port VPI switch which supports 10Gb/s and 40Gb/s Ethernet and 56Gb/s FDR InfiniBand. The Operating System used was RHEL6.2, the InfiniBand driver version was OFED 2.1-1.0.0, and the File System is shared over NFS, which provides 1TB SATA hard drive. The MPI library used was Platform MPI 9.1.

References to the previous performance studies [4][5][6][9][10] on Abaqus Unified FEA by the HPC Advisory Council can be found at the Best Practices [8] section of the HPC Advisory Council web site [1].

The Abaqus version was 6.13-2. The Abaqus/Standard benchmark workload was the test simulation S2A that has 474,744 DoF. This description of the benchmark is provided on the Abaqus Performance Data webpage [3]. The benchmark utilizes the nonlinear static analysis of a flywheel with centrifugal loading. The flywheel is meshed using 1st order hexahedral elements of type C3D8R and uses an isotropic hardening Mises plasticity material model. The nonlinearity in this problem arises from localized yielding in the vicinity of the bolt holes.

2.2 Benchmark Cases

The Abaqus/Explicit benchmark workload used is the test simulation E6. The E6 is the benchmark that consists of many concentric spheres with some spacing between each sphere. The Concentric Spheres with 244,124 elements, as shown in Figure 1. The spheres are put into a general domain and forces are exerted from the outer sphere that creates some complex interactions between the spheres from within. The E6 model is an example of a memory bandwidth bound problem.

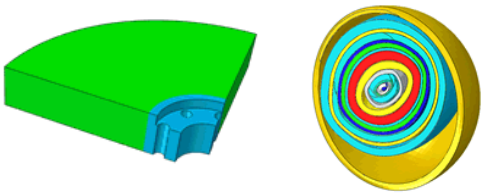

Figure 1: Abaqus/Standard Benchmark workloads: S2 Flywheel (left) and Abaqus/Explicit workload E6 Concentric Spheres (right)

2.3 Performance Rating

The performance rating is the metric that will be shown; the value of performance rating is calculated based on the amount of jobs can be run in a day from the actual runtime achieved.

We measure the total runtime of each dataset that is accumulated with the time in seconds for the input file preprocessor, solver phases. Then we take the total runtime and convert it into a value that represents the performance metric rating based on the number of analyses run that can be achieved during a 24 hour period. The metric we display is effectively becomes the number of jobs can be run per day. This rating system provides a much clearer representation when observing the scaling progression in a multi-core and multi-node environment because it can clearly shows the additional amount of workload can be achieved by increasing the number of systems involved in the simulation. It also becomes useful when comparing environment of similar system components.

It is also to note that in the traditional HPC environment, it is a common practice to utilize all of the CPU cores and hardware resources available on the compute nodes to run exclusively for a single workload. By having dedicated compute nodes for a particular parallel job, the underlying hardware would not run into resource contention or interference with other processes, it is regarded as a good approach for achieving highest performance for a parallel job. Thus, the comparisons made in the subsequent sections will be on a per-node basis, enough though the actual CPU cores available on a node may vary.

3. Analyses on Turbo Mode and CPU Frequencies

3.1 MSR Tools and CPU Governor

To show the cost and benefit of turbo mode and higher CPU clock frequency, we are relying on the software tools which allow us to boost and throttle the CPU clock frequency and toggling the turbo mode dynamically in the OS. The software tools that we used for modifying turbo mode dynamically in the OS is a kernel tools called "msr-tools"[7]. MSR stands for Machine Specific Register.

Another method to dynamically change the CPU clock relies on the kernel module called CPU governor [2] which is available in the OS. This tools facilitates our benchmark testing with the different CPU speeds as we would not need to acquire the actual CPU processors that runs that particular clock rates.

3.2 Effects of CPU Frequency

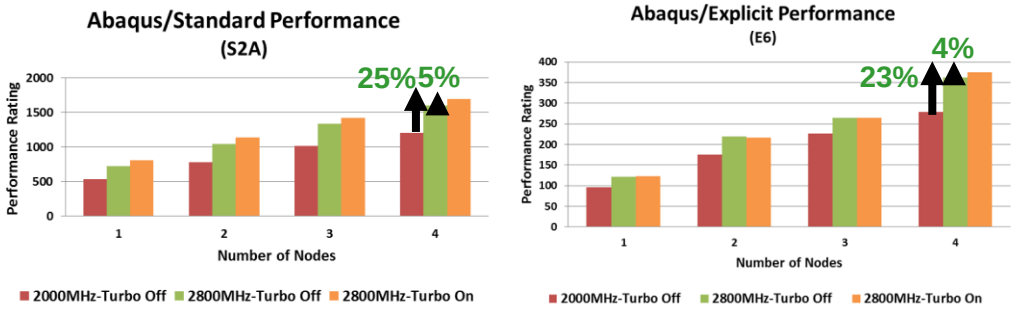

Figure 2: Turbo Mode Effect on Abaqus/Explicit Performance

The data labeled with "2800MHz-Turbo Off" is our performance baseline since it is the type of processors that we have installed on all the compute nodes in the HPC cluster. The data labeled "2000MHz-Turbo Off" is the comparison run that is conducted by lowering the CPU clock speed using the CPU governor available in the OS. The reason that we are interested in scaling down to 2800MHz to 2000MHz is to observe the performance difference between the 2, in order to judge whether it makes sense to use a CPU SKU with the lower clock frequency.

We observe that there is some modest dependency on CPU frequency by showing the performance difference using the Abaqus/Explicit benchmark case, as shown on Figure 2. There is a 23% of the performance difference between 2000MHz vs 2800MHz. However, the tradeoff of having a higher

clock frequency that delivers the performance is the power consumption. There is about 51% of the power additional power needed to run from 2GHz to 2.8GHz CPUs. This implies that there is actually some benefit of using higher clock rate CPUs on Abaqus workload, as generally there are CPU models available at higher clock rate, but at a higher price point premium.

The similar behavior on CPU frequency and Turbo mode have exhibit with the Abaqus/Standard test case on Figure 3. While there is about 25% of the improvement at 51% of higher power utilization by enabling higher clock frequency from 2000MHz to 2800MHz.

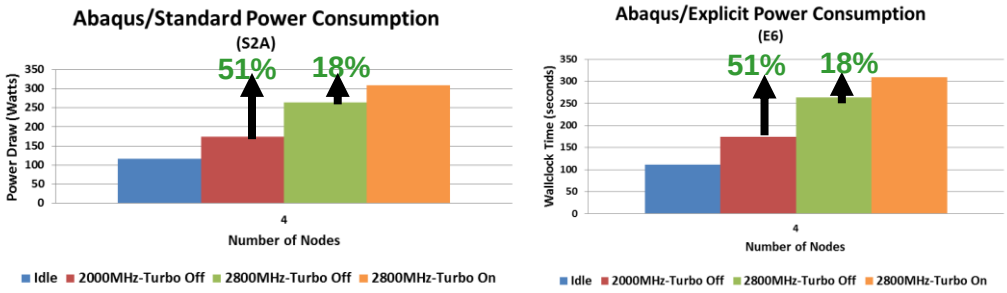

Figure 3: Power Consumption of Enabling CPU Turbo Mode on Abaqus/Standard and Abaqus/Explicit

3.3 Effects of Turbo Mode

On the other hand, enabling Turbo Mode can only provide very marginal difference of performance for Abaqus/Explicit. Enabling Turbo mode can provide approximately 4% of performance gain for CPU cores running at 2800 MHz, which resulted in 18% of higher power usage as a tradeoff for the additional performance. It is clear that turbo mode only provide marginal additional benefit for Abaqus/Explicit, while the power consumption of turbo mode overweighs the benefit.

Similarly for Abaqus/Standard, it is also observed a 5% gain in performance when using Turbo Mode, which results in a tradeoff of higher power consumption by 18%.

4. Impact of Interconnect on Abaqus Performance

The cluster interconnect is very critical for efficiency and performance of the application in the multi-core era. When more CPU cores are present, the overall cluster productivity increases only by the presence of a high-speed interconnect. We have compared the elapsed time with Abaqus using 1Gb/s, 10Gb/s and 40Gb/s Ethernet, and 56Gb/s FDR InfiniBand.

Abaqus FEA software utilizes Message Passing Interface (or MPI) for cluster or node-to-node communications. MPI is the de-facto messaging library for high performance clusters. It relies on fast server and storage interconnects in order to provide low latency and high messaging rate. Performance demands from the cluster interconnect increase dramatically as the simulation requires more complexity to properly simulate the physical model behavior.

4.1 Ethernet

The 10GbE and 40GbE are the newer standards used in the data center environment. The Ethernet interconnects are typically used in the data center to provide a cost-effective solutions for transferring data between the compute nodes and storage systems. While these data rates provides faster throughput than 1GbE that is typically available on server systems, the 10GbE and 40GbE are not able to deliver good performance in the HPC environment, as majority of the communication that takes place in the HPC run are not point-to-point communication, but rather one-to-many and many-to-many group communications that may overwhelm the Ethernet network even at small scale.

4.2 InfiniBand

This network interconnect has a scalable architecture for delivery performance on large high-performance clusters environment, specifically designed for use by the largest supercomputers in the world. By providing low-latency, high-bandwidth and extremely low CPU overhead, InfiniBand has become the most deployed high-speed interconnect for HPC clusters, replacing proprietary or low-performance solutions. The InfiniBand Architecture (IBA) is an industry-standard fabric designed to provide high-bandwidth, low-latency computing, scalability for tens of thousands of nodes and multiple CPU cores per server platform and efficient utilization of compute processing resources.

Figure 4 shows the Performance Rating for the InfiniBand and Ethernet interconnects for a range of node counts for the benchmark cases on the same set of the aforementioned cluster of servers.

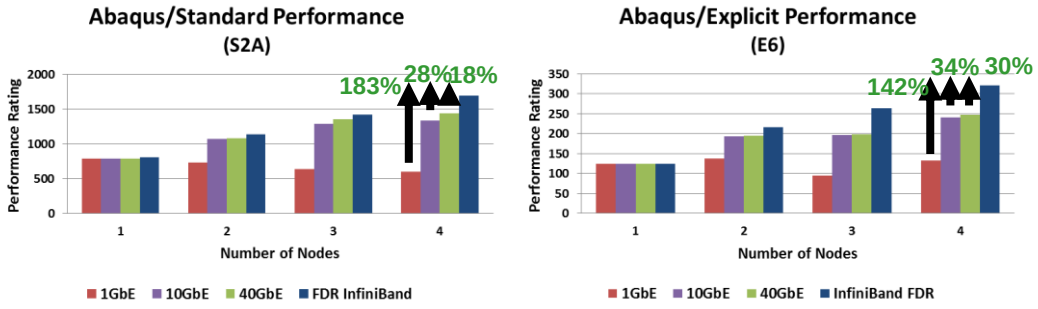

Figure 4: Interconnect Comparisons between 1/10/40Gb Ethernet and FDR InfiniBand

4.3 Performance Comparisons on the Network Interconnects

The result shows that InfiniBand delivered better scalability in application performance, which results in faster run time which translates into higher performance rating, which means more jobs per day. The 56Gb/s FDR InfiniBand-based simulation performance was 142% higher than 1GbE, over 34% higher than 10GbE and over 30% higher than 40GbE on a Abaqus simulation which runs on 4 nodes or 80 MPI processes.

Likewise for the Abaqus/Explicit case, FDR InfiniBand enables higher cluster productivity among the network interconnects being tested. InfiniBand reduces the runtime by 183% versus 1GbE, and delivers higher performance by 28% versus 10GbE, and 18% higher performance versus 40GbE.

While 1Gb, 10Gb, and 40Gb Ethernet showed some loss of performance or increase in run time beyond 2 nodes or 40 MPI processes, InfiniBand demonstrated good scalability throughout the various tested configurations.

4.4 Time Ratio on Different Network Interconnects

InfiniBand can speed up the communication time, which is not only due to the throughput available, but it provides a much simpler and less congested communications that is more suitable for the HPC environment. By conducting the MPI profiles among the runs from different network interconnect, we captured the amount of time that different network interconnect would use to run the same workload. We found that FDR InfiniBand consumes about 37% of total runtime at 4 nodes/80 MPI processes, while Ethernet solutions consume from 53% to 72% at 4 nodes. This is shown on Figure 5.

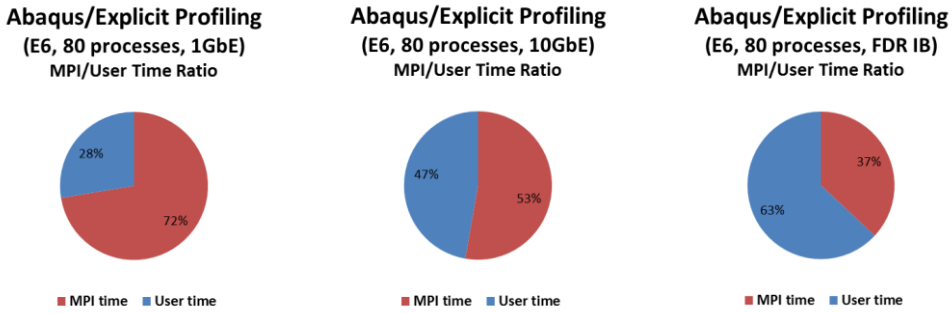

Figure 5: Time Ratio between Computational (User) Time and Communication (MPI) Time

5. MPI Profiling

By inspecting the output from the MPI profiler, it is observed that the majority of the MPI messaging appears to be concentrated on the small message buffer sizes. We can then understand that the underlying communication patterns that make Abaqus scalable on the low latency interconnect.

5.1 Time Spent on MPI Communications

By profiling the communication, we can observe in the type of MPI calls that are involved. We can see the type of communication that Abaqus engages the MPI processes. In Figure 6, the MPI profiler shows FDR InfiniBand spends approximately 54% of the time during the job run in the MPI_Gather, 9% time MPI_Allreduce, and approximately 9% of the time in MPI_Scatterv on a 4-node, 80 processes job. The MPI profile shows Abaqus/Explicit would majority of time in MPI collective operations. The gather algorithm would gather values from a group of MPI processes, while scatter would do the reverse. The MPI_Allreduce would do a combination of both Scatter and Gather, with a preset operation, such as the sum operation. This clearly shows that many of the processes are engaged in a group area of communication.

Similarly, when taking a look at the communication pattern on a slower, less efficient network interconnect, we can compare the two and identify in the areas that would help by the higher

performance interconnect. The time spends on the 1GbE network is about 43% for MPI_Gather, 13% for MPI_Irecv, 14% for MPI_Scatterv.

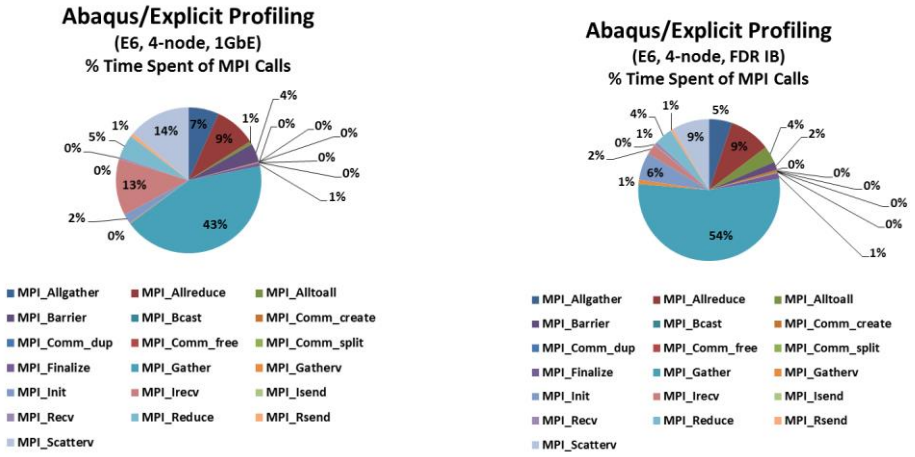

Figure 6: Profiling Differences of MPI Communications between 1GbE and FDR InfiniBand

There are difference in communication patterns between Abaqus/Standard and Abaqus/Explicit. Abaqus/Standard shows high usage for testing non-blocking messages, for instance, MPI_Gather dominates the MPI communication time in Abaqus/Explicit. For Abaqus/Standard, majority of the time is spent on MPI_Test. This is shown on Figure 7.

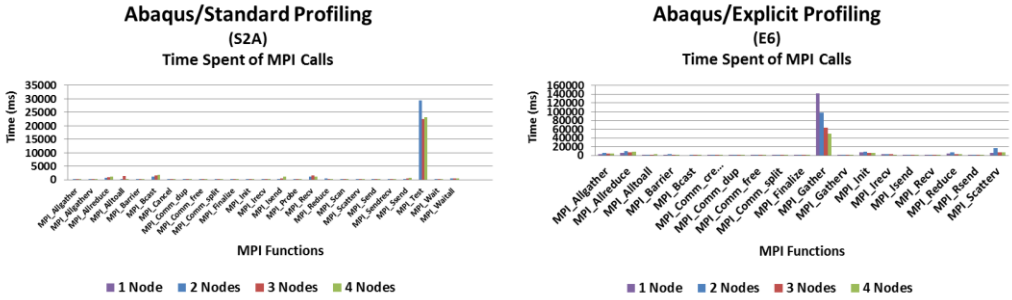

Figure 7: Communications Profiled on Abaqus/Standard and Abaqus/Explicit

In the MPI profiling, it shows that Abaqus uses MPI library for communications between the application and the networking layer, it requires a scalable and efficient send-receive semantics as well as good scalable collectives operations.
InfiniBand provides an effective method for communication called Remote Direct Memory Access, or RDMA. With RDMA, InfiniBand can write directly to the pinned system memory of the remote systems, which reduces the CPU involvement in handling of the networking traffic. By freeing up the network traffic from CPU, can focus on computation which allow better performance in computation. On the other hand, the collision in the network in the TCP protocol

in the Ethernet network would lead more overheads for CPU, which can translate into higher network latency, reducing the cluster efficiency and scalability.

5.2 MPI Message Sizes

Profiling the MPI communications that is taking place during the job run can show the application uses MPI to communicate between the MPI processes. By knowing the MPI message sizes and communication patterns, we can estimate the benefits of using a network interconnect. The profile shows us that Abaqus/Standard uses small and medium MPI message sizes for communication.

We see that most of the MPI messages are appeared in the small message sizes under 256 bytes. For the most time consuming MPI calls, MPI_Recv messages are concentrated 4KB. Majority of the MPI_Bcast messages are in buffers that are less than 16B. For MPI_Allreduce, most messages are less than 256B.With the MPI profiling tool, we have conducted some profiling samples that include the MPI message size distribution in Figure 8.

We analyzed that majority of the MPI messages fall into a 0 to 256 Byte range and the total of all MPI message sizes fall in the 0-64KB range which are small message sizes. The overall MPI communications collected was approximately 30GBs from all MPI ranks for about 50 seconds by all 80 MPI processes. This amount of MPI traffic is low when InfiniBand can handle about 6.4GB/sec with FDR InfiniBand throughput. With Ethernet (1GbE) we can easily saturate the network when running two or more datasets since one analysis consumes about 57% of the total bandwidth of the GigE network. Knowing the MPI message size can be useful when evaluating new interconnect technologies.

The small messages are typically used for group communications for coordinating the processes. Most of those MPI messages fall between 0B to 64B, and 65B to 256B. There is some concentration of medium size messages in the range of 64KB to 256KB. On the other hand, Abaqus/Explicit shows a wide distribution of small message sizes. The small messages peak in the range from 65B to 256B.

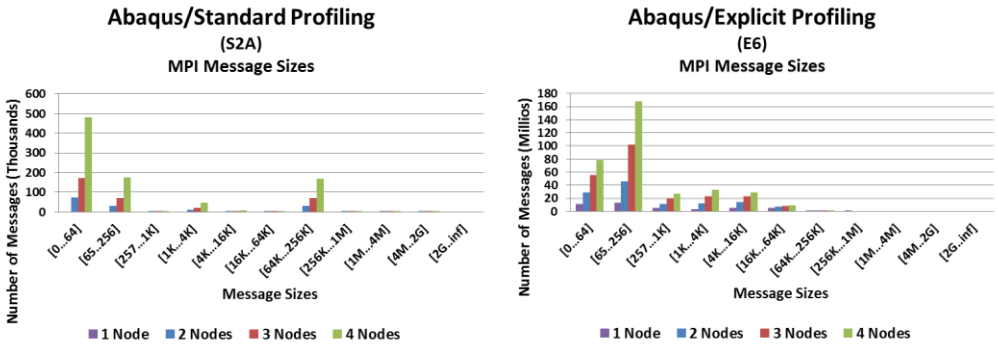

Figure 8: Message Sizes used for MPI communications during Abaqus/Standard and Abaqus/Explicit job run

Many HPC applications are based on communications patterns that use many small messages between parallel processes within the job. It is critical that the network interconnect used to transport these messages provides low latency and high message rate capabilities to assure that

there are no bottlenecks to the application. The InfiniBand architecture can deliver over much more messages per second to the network than the Ethernet solutions. This increase assures that there are no message rate limitations for applications and that the multiple cores on the server will communicate to other machines as fast as the cores are capable, without any slowdown from the network interface.

6. Observations on File I/O

The file I/O can also be an important element in Abaqus performance.

For Abaqus/Standard, it is observed on that approximately 27 to 46% of the total time on throughput. There is a mixture of I/O that is accessed to the local file system for scratch, or the system shared memory space like /dev/shm, and some are accessed over the central file system, such as NFS or parallel file systems, such as Lustre, PVFS, FhGFS and others.

All of the processes are involved in the disk I/O activity. Most of the I/O activities are write activities, which accounts for approximately 83% are in the range of 1KB to 8KB sizes. There is also a reasonable amount of system calls that are made for I/O reads, approximately 72.5% are 1KB to 8KB sizes. It also appears from the I/O profiling that a period of time there it has no I/O activities, it is assume that the time period is used for running the solver. Figure 9 shows the write and read activities for the Abaqus/Standard cases. Therefore, a decent file storage that provides good IOPS and throughput is essential for Abaqus/Standard performance. Additional findings were reported on the previous HPC Advisory Council Best Practices [5,8]

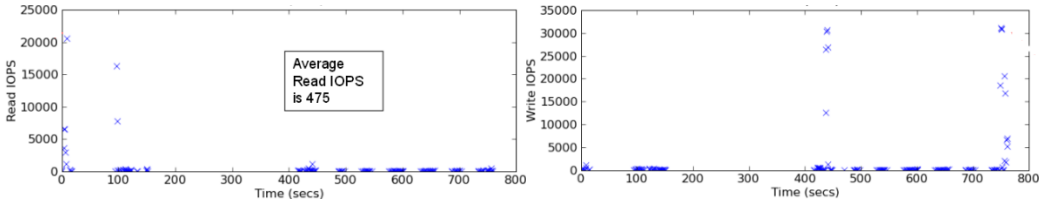

Figure 9: Write and Read Throughput History for an Abaqus/Standard test case

On the hand, Abaqus/Explicit shows little time for I/O activities compared to the overall run time, as depicted on Figure 10. The previous HPC Advisory Council Best Practices studies [8] on Abaqus/Explicit [6], it is found that the first MPI rank, also known as rank 0, handles most of the file I/O activities. Only approximately 134MB is written and 10MB is read by the rank 0 process. The rest of the MPI processes are having even less disk I/O activities; the activities are generally used for loading the shared libraries for Abaqus/Explicit.
Given the IO profiling it shows the benchmark dataset only exhibited very limited IO activities. It is not clear whether or not the lack of disk activity is due to the nature of the benchmark being performed for Abaqus/Explicit.

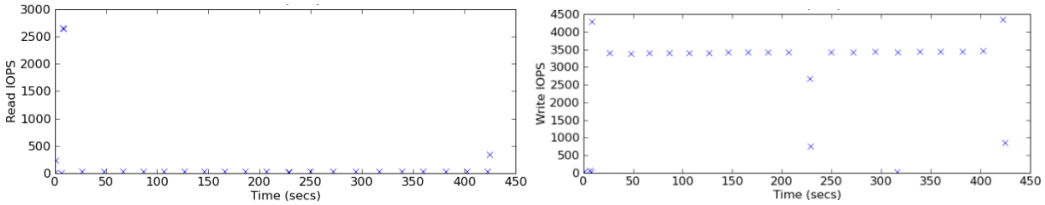

Figure 10: Write and Read Throughput History for an Abaqus/Explicit test case

7. System Architecture

HPC systems typically get periodic update by the CPU refresh cycle. When the new generation of CPU architecture is introduced, there the performance is expected to be increased. We have conducted a study on the performance improvement by the different system architecture due to the CPU generations for the over time.

7.1 Descriptions of System Architecture Compared

To conduct the performance comparison tests, the following system configurations were used:

- **WSM**: Each "Westmere" system used the HP ProLiant SL2x170z systems, with a dual-socket 6-core Intel Xeon X5670 running at 2.93GHz, 1333MHz DIMMs, and Mellanox ConnectX-2 QDR InfiniBand. [10]
- **SNB**: Each "Sandy Bridge" system used the HP ProLiant SL230s Gen8, each with a dual-socket 8-core Intel Xeon E5-2680 running at 2.7GHz, 1600MHz DIMMs, and Mellanox Connect-IB FDR InfiniBand.[9]
- **IVB**: Each "Ivy Bridge" system used the aforementioned HP ProLiant SL230s Gen8, each with a dual-socket 12-core Intel Xeon E5-2680 v2 running at 2.8GHz, 1600MHz DIMMs, and Mellanox Connect-IB FDR InfiniBand [4]
- **HSW**: Each "Haswell" system, each with a dual-socket 14-core Intel Xeon E5-2697v3 running at 2.6GHz, 2133MHz DIMMs, and Mellanox Connect-IB FDR InfiniBand [11]

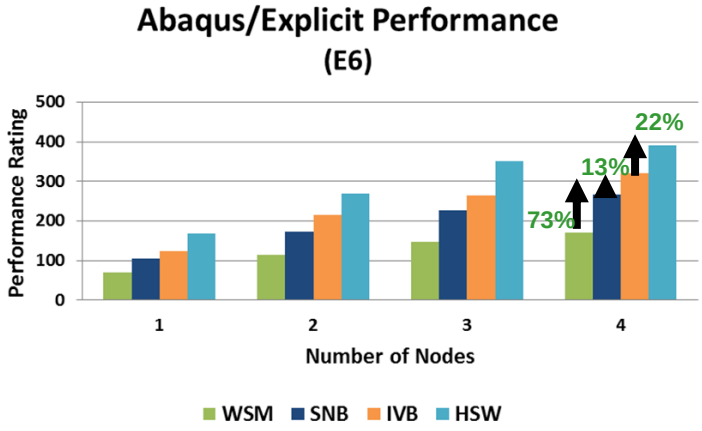

Figure 11: Abaqus Performance on system architecture based on the CPU technology over time

7.2 Differences among System Architecture

By comparing to system architecture of different generations, it becomes clear that the latest generation of system based on Intel Xeon E5-2697 v3 (Haswell) cluster can deliver gain of up to 22% higher performance over the older Intel Xeon E5-2680 v2 (Ivy Bridge) at 4 nodes. Likewise, the Intel Xeon E5-2680 v2 (Ivy Bridge) cluster outperforms the Intel Xeon E5-2680 (Sandy Bridge) cluster by up to 13%, it also outperforms Intel Xeon X5670 (Westmere) cluster by up to 73%. There is a wide gap in performance improvement between Westmere and Sandy Bridge platforms is most likely because of the improvement in the system architecture beginning in Sandy Bridge. Notably, the Ivy Bridge architecture enables the PCIe device to run at its maximum throughput and lowest latency. Thus hardware such as InfiniBand can operate in the PCIe Gen3 standard, which is able to provide higher throughput over the old PCIe Gen 2 standard. Among the rest of the changes, there are also an increase in CPU core processing, memory bandwidth which provide the necessary network throughput for the network. The results are shown in Figure 11.

8. Software Releases

Another interesting aspect of the component that make a difference in performance is the version of Abaqus FEA software that is used. Figure 12 shows that Abaqus/Explicit would perform faster than with the 6.13-2 over the previous version of 6.12-2 by 8% at 4 nodes / 80 CPU cores. However, the difference in Abaqus/Standard with the version 6.13-2 over the older version of 6.12-2 is not as clear. Marginal gain can be seen on the dataset tested.

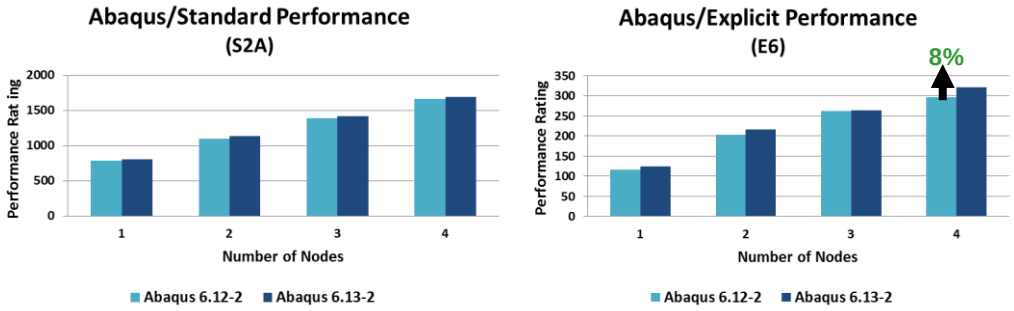

Figure 12: Performance Differences in Software Versions between Abaqus 6.12-2 and 6.13-2

9. Conclusion

HPC clustering technology allows Abaqus Simulations to achieve higher productivity by minimize time-to-solution by deploying high performance systems to run simulation in parallel. The components in the servers such as CPU, network, and IO can make a great influence on the Abaqus performance. We have demonstrated that the performance improvement is providing

through over the different CPU generations which also introduce some system architecture. By adjusting the different frequencies and toggling the Turbo mode setting, we have demonstrated the benefit of increased Abaqus productivity with higher CPU clock and the tradeoff of increase power consumption for running the simulation.

HPC cluster environments impose high demands for connectivity throughput, low-latency, low CPU overhead, network flexibility and high-efficiency in order to maintain a balanced system in order to achieve high application performance and scaling. Low-performance interconnect solutions, or lack of interconnect hardware capabilities will result in application performance and result in an extended time-to-market process.

It is to note that these two benchmark workload obtained from the Abaqus Performance Benchmarking web site shown in this paper are relatively small compared to the typical problems that are employed in real world Abaqus simulations which would require the use of an HPC cluster. Therefore, when evaluating the system performance of HPC cluster, it would worth the time and effort to use benchmark workload that has larger problem size to run with Abaqus on similar computer system hardware.

10. References

1. HPC Advisory Council – http://www.hpcadvisorycouncil.com
2. IPM profile - http://ipm-hpc.sourceforge.net/
3. Abaqus performance Data http://www.3ds.com/support/certified-hardware/simulia-system-information/abaqus-613/performance-data/
4. Abaqus 6.13-2 Performance Benchmarking and Profiling (Ivy Bridge) – http://hpcadvisorycouncil.com/pdf/Abaqus_Analysis_and_Profiling_H-IVB.pdf
5. HPC Advisory Council – Abaqus/Standard I/O Profiling – http://hpcadvisorycouncil.com/pdf/Abaqus%20Implicit%20IO%20Performance%20Analysis.pdf
6. HPC Advisory Council – Abaqus/Explicit I/O Profiling – http://hpcadvisorycouncil.com/pdf/Abaqus%20IO%20Performance%20Analysis.pdf
7. MSR Tools - https://github.com/01org/msr-tools
8. HPC Advisory Council Best Practices - http://hpcadvisorycouncil.com/best_practices.php
9. Abaqus 6.12 Performance Benchmarking and Profiling (Sandy Bridge) – http://hpcadvisorycouncil.com/pdf/Abaqus_Performance_Analysis_Intel_E5_2680.pdf
10. Abaqus Performance Benchmarking and Profiling (Westmere) – http://hpcadvisorycouncil.com/pdf/Abaqus%20Performance%20Analysis_Intel_x5670.pdf
11. Lui, P., "Achieving Higher Productivity on Abaqus Simulations for Small and Mid-size Enterprises with HPC and Clustering Technologies". SIMULIA West Regional User Meeting, San Jose, CA, 2014. (http://hpcadvisorycouncil.com/pdf/141029a%20SIMULIA%20West%20RUM%20-%20Pak%20Lui.pdf)