Fast Setup and Integration of Abaqus on HPC Linux Cluster and the Study of Its Scalability

Betty Huang, Jeff Williams, Richard Xu

Baker Hughes Incorporated

Abstract: High-performance computing (HPC), the massive powerhouse of IT, is now the fastest-growing sector in the industry, especially for the oil and gas industry, which has outpaced other US industries in integrating HPC into its critical business functions. HPC can offer greater capacity and flexibility to allow more advanced data analysis, which usually cannot be handled by individual workstations.

In April 2008, Baker Oil Tools installed a Linux Cluster to boost its finite element analysis and computational fluid dynamics application performance. Platform Open Cluster Stack (OCS) has been implemented as cluster and system management software and load-sharing facility for high-performance computing (LSF-HPC) as job scheduler. OCS is a pre-integrated, modular and hybrid software stack that contains open-source software and proprietary products. It is a simple and easy way to rapidly assemble and manage small- to large-scale Linux-based HPC clusters.

1. Introduction of HPC

A cluster is a group of linked computers working together closely so they form a single computer. Depending on the function, clusters can be divided into several types: high-availability (HA) cluster, load-balancing cluster, grid computing, and Beowulf cluster (computing). In this paper, we will focus on the Beowulf-type high-performance computing (HPC) cluster. Driven by more advanced simulation project needs and the capability of modern computers, the simulation world relies more and more on HPC. HPC cluster has become the dominating resource in this area. It typically comprises three layers: hardware, software, and management between them. These components integrate together to carry out the computing tasks in parallel, resulting in speeding up the whole process.

2. HPC Setup by OCS

The cost/performance ratio for cluster is very attractive, though it is still a big challenge to run commercial finite element analysis software on cluster. We chose Platform Open Cluster Stack (OCS) as cluster management/administration software and LSF (load-sharing facility) as job scheduler. It proved to be a simple and easy way to rapidly assemble and manage small- to large-scale Linux-based HPC clusters.

2.1 OCS Introduction

OCS is a comprehensive cluster and system management toolkit, Intel Cluster Ready certified software suite based on Red Hat Linux. It is built on top of the original OCS toolkit developed by San Diego Supercomputer Center (SDSC). It is a hybrid software stack, containing a blend of open-source and proprietary software technologies. Figure 1 shows the layer structure of OCS.

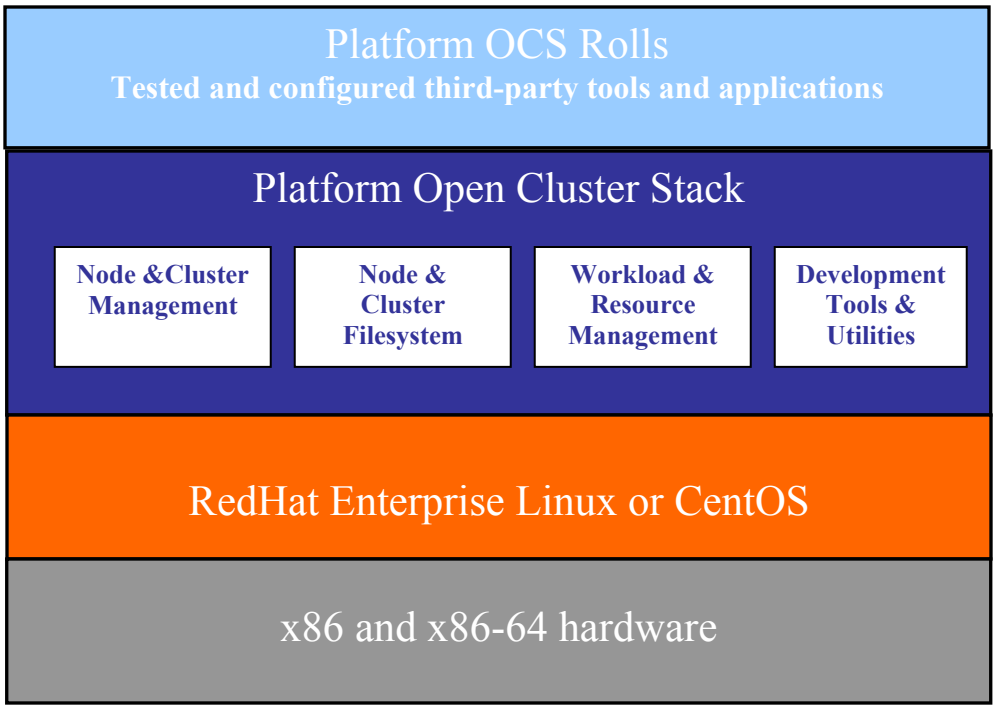

Figure 1. The Structure of Platform OCS

The bottom is the hardware layer and the top is the software/application layer. OCS is the joint of the two parts. It supports 32- and 64-bit x86 hardware and provides the core functions for a cluster: Node & Cluster management, workload & recourse management to provide functions of OS, applications provisioning. The new version OCS 5 has more options in operating systems, such as SuSE Linux, Windows, and IBM AIX.

To ensure the compatibility of the drivers, tools and applications installed on cluster, Platform OCS wraps them into rolls and develops a series of related roll manipulation commands. Roll is a self-contained ISO image that holds packages and their configuration files. It can have one or more RPMs, a set of scripts for installing the package in the post-installation step. There are required rolls and optional rolls. Required rolls are fundamental for OCS to function correctly, such as Base Roll and HPC Roll. To reduce cluster down time, you can dynamically add/remove rolls while cluster is in function.

2.2 Quickly and Easily Set up Cluster by OCS

OCS installation starts from the master node with installation DVD kit. When the installation option prompts, choose installation type as "frontend" (default is compute node). Fill in the cluster name, static IP address, roll selection, etc. and then installation proceeds automatically. In OCS 4.5, Red Hat Linux is included in OCS DVD. While in the new version OCS 5, you need to add extra OS DVD separately.

After installing OCS on the master node, you will get a functional cluster master node with all the cluster basic features included. The built-in utility insert/ethers or add-hosts can add computer nodes, switches, and customized appliances into the database. Reboot into PXE (Pre-Execution Environment), and the computer nodes will start to communicate with the master node to obtain installation and configuration files to complete OS and application provisioning.

The following schema is the Linux cluster of Baker Oil Tools set up by OCS in April 2008 (Figure 2). It contains one master node and six compute nodes. They are all DELL Power Edge servers with dual quad-core processors and 16 GB RAM (2GB RAM per core). RAID1 and RAID5 disk configuration is applied on the master node to provide redundancy.

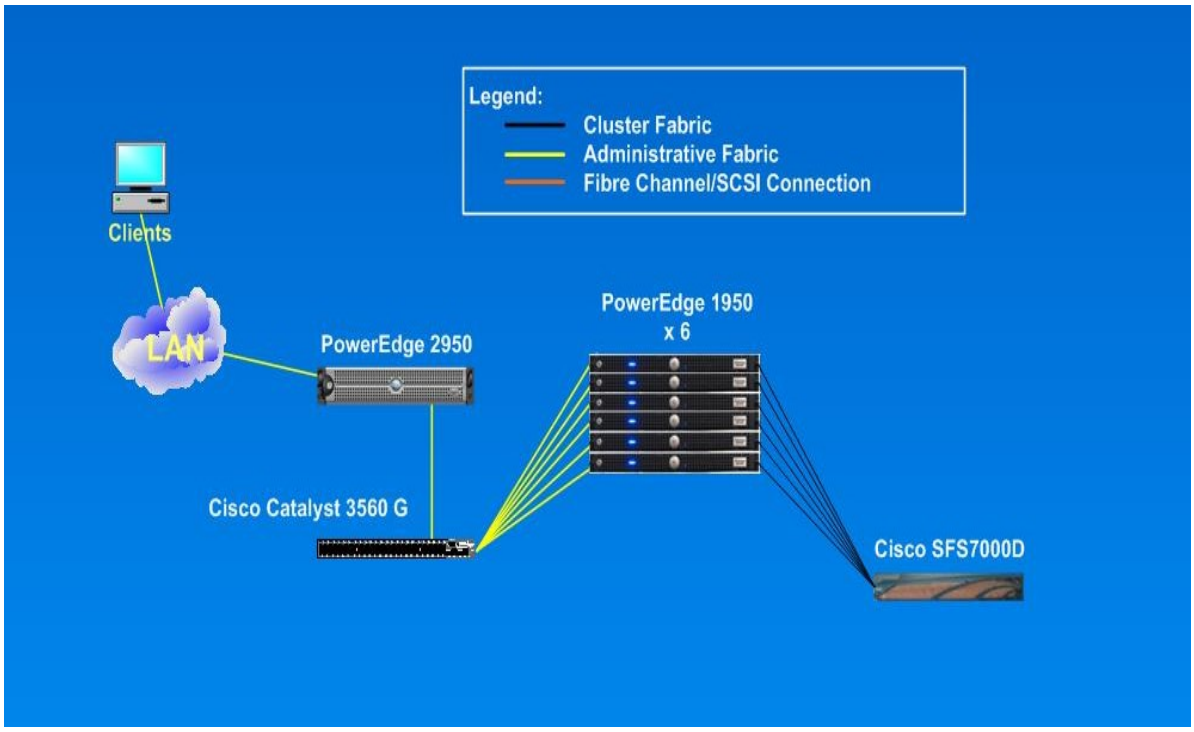

Figure 2. Schema for Linux Cluster of Baker Oil Tools

The whole cluster is isolated from public networking except master node, on which two networking cards are installed. One connects to public networking and one connects to private networking within cluster. We also used two switches: one is an Ethernet switch, providing Ethernet networking for cluster management and job submission. The other is an infiniband switch, providing message passing networking for faster computing. The cluster end user logs into the master node from a public network, and then submits jobs to compute nodes through a private network.

2.3 Live Monitoring by Ganglia

Once the cluster sets up and is running, we use graphic web monitor Ganglia (Figure 3) to monitor cluster status dynamically.

Figure 3. Screenshot of Ganglia (graphic web monitoring tool)

Ganglia provides a good high-level view of the entire cluster load and individual node load. It "pushes" the data via UDP from a gmond daemon to a gmetad daemon. The monitor gathers information for various metrics such as CPU load, free memory, disk usage, network I/O, and operating system version.

3. Abaqus Integration on Cluster by LSF

LSF is a job scheduler that configures multiple systems to perform the same function so that the workload is distributed and shared. On this cluster, we use HPC LSF. Compared to general LSF, it offers advanced HPC scheduling policies to enhance the job management capability of cluster, such as policy-based job preemption, advance reservation, memory and processor reservation, cluster-wide resource allocation limits, and user- and project-based fair share scheduling.

Platform LSF can help computer-aided engineering (CAE) users reduce the cost of manufacturing, and increase engineer productivity and the quality of results. It is integrated to work out of the box with many HPC applications, such as LSTC LS-Dyna, FLUENT, ANSYS, MSC Nastran, Gaussian, Lion Bioscience SRS and NCBI BLAST. We integrated Abaqus, Fluent and MSC.Marc with HPC LSF batch submission.

Additionally, we also use VNC remote desktop to launch Abaqus CAE, interactively submitting jobs. In this way, the user will have a shorter learning curve to using cluster. Recently, a new way was discovered to launch Abaqus CAE through bsub –I, which can submit job via GUI under LSF scheduler control.

4. Abaqus Parallel Scalability Study

Abaqus users want to get the best performance out of the software and hardware combination. So, we performed parallel scalability studies for Abaqus before and after purchasing the Red Hat Linux cluster. The information obtained in the serials of systematic study serves as the reference to make purchase decisions and optimize simulation job running.

4.1 Abaqus benchmarks on single workstation of Red Hat Linux and 2003 Windows server

In order to study the performance of Abaqus on Linux and Windows platforms, we ran both explicit and one standard model involving different-size Abaqus benchmark models and two BOT engineering jobs on identical workstations (Intel 3.6 GHz, dual-core, 4 GB memory) with Red Hat Linux and Windows 2003 server. The run time (seconds) is listed in Table1. The red fonts indicate better performance.

Abaqus Model	***Standard***				***explicit***	
	S3A	***S5***	***S6***	***Ball_Seat (non-linear)***	***E2***	***Auxetic2***
Windows 2003 Server	10042	2781	12877	764	13518	6673
Red Hat Linux 4	9089	3771	9981	513	16089	6926

Table1. Running time for Abaqus 6.7 EF-1 on identical DELL Power Edge servers with different operating systems

Due to the RAM limitation of these computers, we can't run a larger model. From the results above for small to mid-size models, it is hard to tell which OS wins.

4.2 Abaqus benchmarks on Red Hat Linux Cluster

Figure 4 shows the run time results for Abaqus S2B model running on our Linux cluster by Abaqus 6.8 implicit solver with various numbers of CPUs. It can be shown that the compute cluster approaches maximum efficiency with 16 CPUs (nodes).

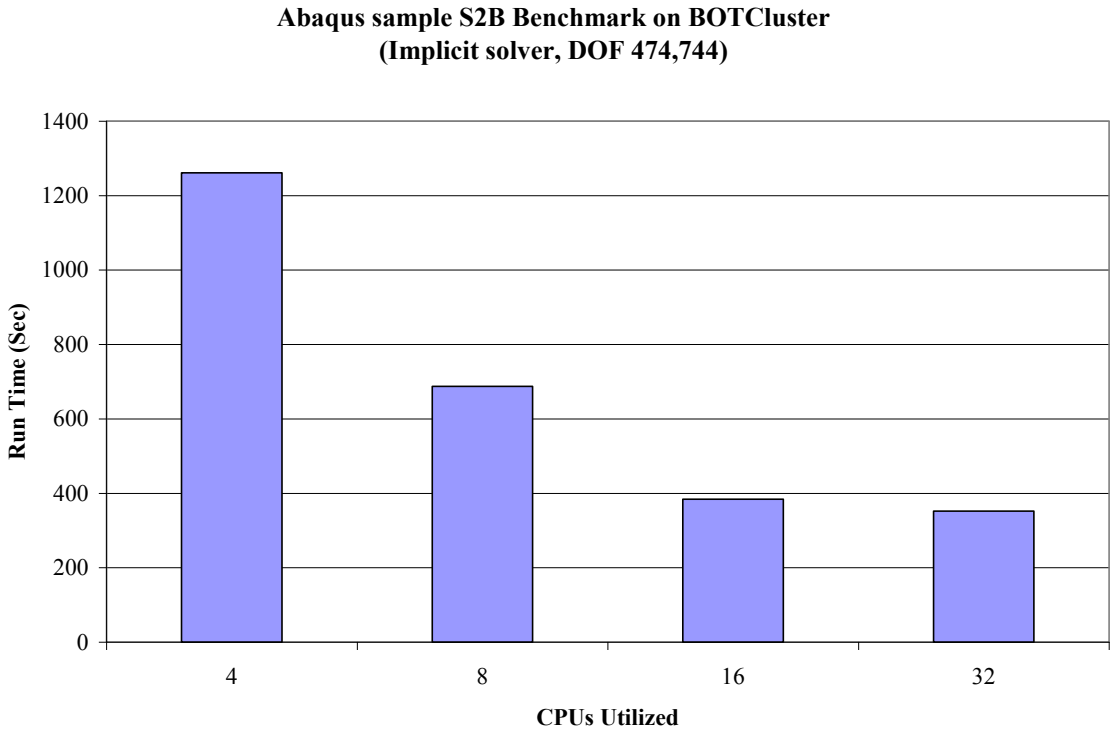

Figure 4. Abaqus benchmark of S2B on BOTCluster by Abaqus 6.8

Another study with a high number of elements was performed with the explicit solver. The model consisted of a 3D segmented cone forming inside blank pipe (Figure 5). The model contained high interface contact (.005" elements to define the surfaces) for the segments forming the metal. The segments were meshed with discrete rigid, linear quadrilateral shell elements and the pipe had 3D deformable, linear hexahedral elements.

The results (Figure 6) were scalable, but varied depending upon how many nodes per host were used. The results did not show clear parallel scalability with the number of CPUs increasing if the hosts were free to use all 8 nodes (ptile=8). Each time a new host was introduced there was a dramatic increase in run time. This was due mainly to the communication cost. As the number of CPUs increased, this influence was minimized but reached a "steady state" at approximately 24 CPUs. Although, if the hosts were limited to using 6 nodes (ptile=6), then the study showed maximum efficiency at 12 CPUs.

Still, two more studies of less complexity were analyzed with the explicit solver: Abaqus benchmark samples─E1 (with 274,632 of elements) and E3 (with 34,540 of elements). Results are shown in Figure 7 and Figure 8, respectively. These studies showed good scalability with increasing CPUs, although it should be noted that the BOT study had much more contact. Similar to the implicit study, one can gather from this statistical data that using two hosts is the most efficient setup for the cluster with a large model (whether 6+6 nodes or 8+8 nodes).

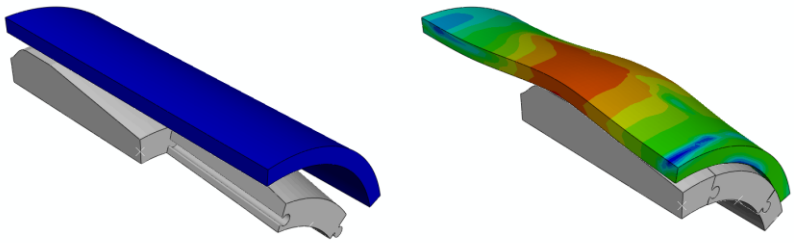

Figure 5. Expansion of segmented cone forming inside blank pipe by Abaqus 6.8EF1

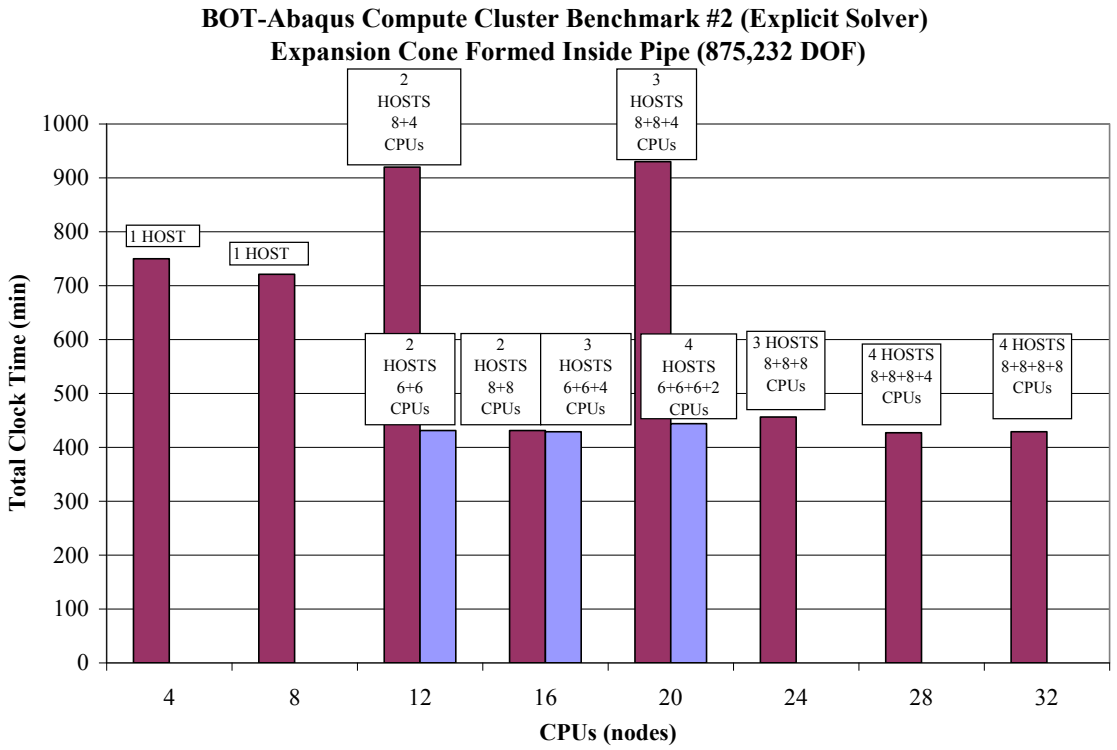

Figure 6. Benchmark of expansion cone forming inside blank pipe on BOTCluster by Abaqus 6.8EF1

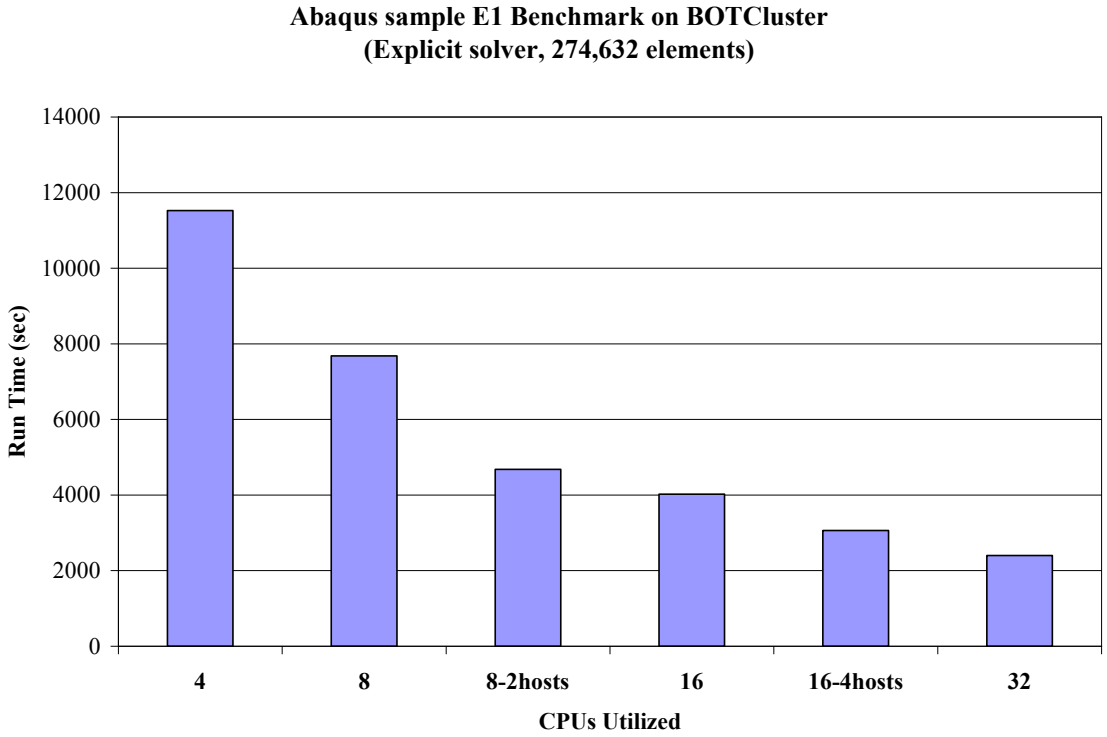

Figure 7. Abaqus benchmark of E1 on BOTCluster by Abaqus 6.8

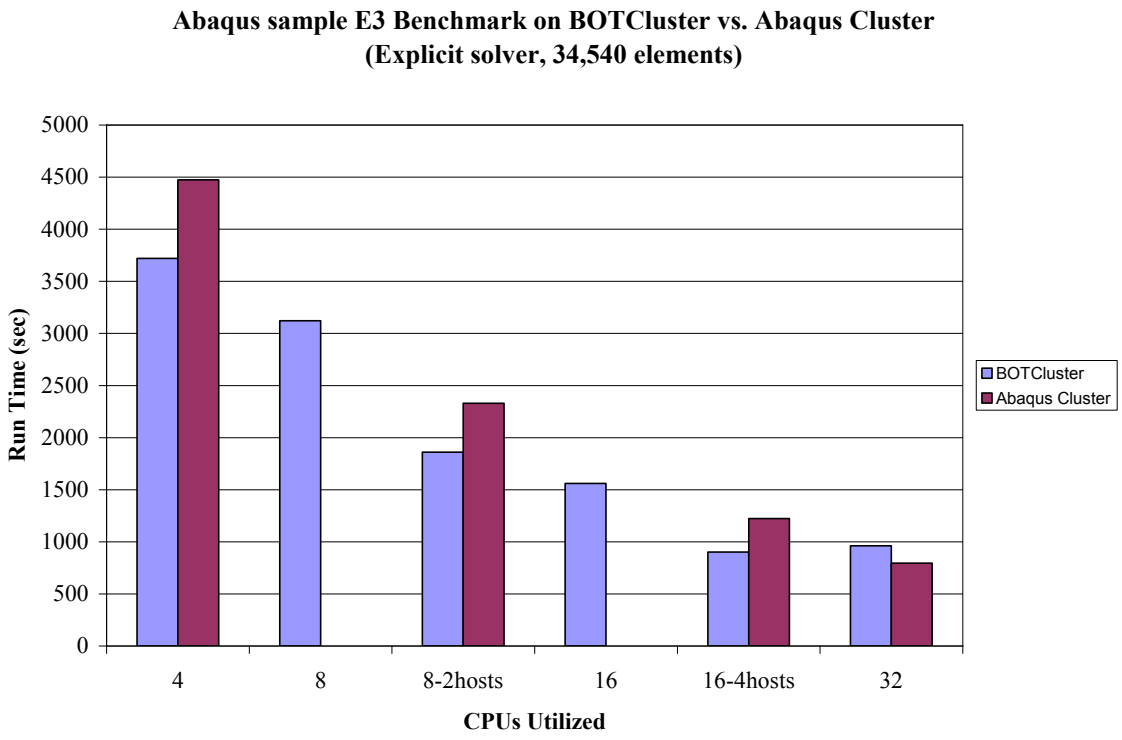

Figure 8. Abaqus benchmark of E3 on BOTCluster vs. Abaqus Cluster by Abaqus 6.8

4.3 Analysis of the benchmarks

The cost of distribution

Distributing jobs to multi-CPUs and multi-hosts comes with an added cost of more memory and CPUs to get the job done. When parallel efficiency exceeds the cost of communication, disk I/O and license usage, you will gain benefit. Otherwise, it could end up a detriment. Distribution will also cause the output database to grow (See Table 2).

Comparing E1 and E3, E3 was moderate and contains less contact. In certain ranges of CPUs, when the job was split to multiple hosts it showed better performance (e.g. comparing 8 CPUs on one host and two hosts). But with the number of CPUs increasing, the cost of communication exceeded the benefit of gaining more resources from multiple hosts. When this happens, the performance suffers (e.g. comparing CPU16 and CPU16-4hosts). When the model size grows larger and the degree of complexity also increases, the Linux cluster shows the intrinsic scalability - the more CPUs and more hosts that are implemented, higher performance is obtained.

Again, considering the BOT segmented cone model, the model size is much bigger than E1 and E3 and it runs much longer. Plus, it contained much more contact, which resulted in more communication cost when distributed to multi-hosts.

	E1	E3
4 CPUs	382982956	227445248
8 CPUs	385440672	228917432

16 CPUs	388773796	231454096
32 CPUs	395209940	236728768

Table 2. The size of .odb files generated with different CPUs utilized in bytes

Dual-core vs. Quad-core

Figure 8 compares our benchmark E3 with Abaqus' result (LIN64-29). We found at lower CPUs, BOTCluster shows better results. When CPU number climbs up, when we reach 32 CPUs, BOTCluster slows down. I think it may be due to different computing node infrastructure. BOTCluster uses dual quad-core processors, and Abaqus uses dual dual-core. In Abaqus 6.7 the element loop calculations are done using MPI. When moving to a DMP situation (more than one node) the element calculations are still done with MPI but only one thread is used per node. This is absolutely a bottle neck for using Abaqus parallel. I don't know what changed in Abaqus 6.8 for this part. The results fit the performance prediction for dual-core and quad-core processor.

Loading balance effect

Also, the result of 12 CPUs is always out of the trend curve. We studied this case in details (see Figure 9). When 12 CPUs was divided on 8 CPUs on one host and 4 CPUs on another host, comparing to equally spreading out on two hosts, it needs more running time. This may indicate some load-balance efficiency. This needs to be confirmed with more research in future.

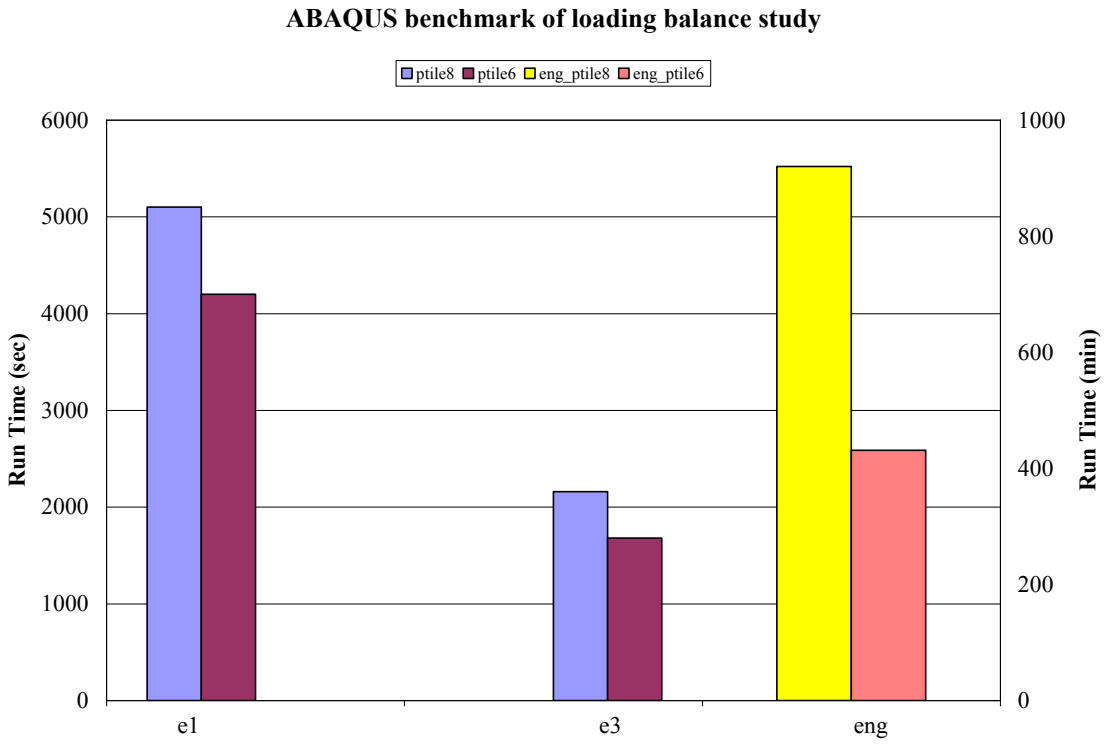

Figure 9. Abaqus benchmark of loading balance effect study

5. Conclusions

Overall, OCS is simple to rapidly set up Linux-based HPC Clusters. It can effectively minimize the cost and time spent on deploying and managing a Linux cluster. From the benchmark results above, we can see in general cases, Abaqus showed very good parallel scalability performance on Linux Cluster. And

resource requirement varies with the models. To optimize using the cluster resource, a deeper understanding of model category, the complexity and element contact level seems necessary.

6. References

1. Bernstein, J., Arend Dittmer, "Higher Returns on the Simulation Investment," ANSYS Advantage, Vol. II Issue 3, 2008.
2. Hutchings, B., "Cluster Computing with Windows CCS," ANSYS Advantage, Vol. II Issue 3, 2007.
3. Platform OCS 4.x Administration Virtual Course Training Material, 2008.

7. Acknowledgment

Special thanks to Tom Gardosik (Baker Hughes-INTEQ), Steve Hiller (Baker Hughes-BHBSS) and Rob Miller (Abaqus), who provided a lot of help in Cluster infrastructure design and installation. Also, my supervisor-Rodney Frosch assisted in determining the computing resource specifications and accelerating the purchasing process.