

Optimizing Linux Clusters for ANSYS 11

By: Joshua Bernstein and Arend Dittmer

Introduction

In many organizations design engineers perform FEA (Finite Element Analysis) based simulations on personal desktop systems. Even though increasing hardware performance has enabled the solution of complex problems on desktop systems, this approach has limitations. Interactive design work on the desktop is interrupted by compute intensive simulation runs, negatively affecting productivity. Moreover this approach requires high-powered desktop systems that are not shared with other users and therefore only fully utilized for relatively short periods of time. An approach, where simple simulations are run on desktop systems and more complex problems are solved on shared 'back-end' compute systems is more efficient. Due to their excellent price/performance ratio, Linux based clusters of commodity systems have become the dominating platform for these 'back-end' computations. While Linux-based clusters are a cost effective way to address the always increasing demand for compute cycles, the concept of achieving high performance through interconnected systems introduces performance and manageability challenges. This paper introduces how the choice of a cluster architecture and the selection of hardware components can impact cluster manageability and ANSYS application performance.

Manageability Challenges

One of the biggest obstacles to a quick return on a cluster investment is getting through the initial installation and configuration of the operating system and cluster management software. After the initial deployment the system configuration on all compute nodes needs to be kept consistent. Even minor misconfigurations, as for example a library mismatch or a missed driver update on a single system can be hard to identify in a cluster. In case of a system failure on a compute node or a cluster upgrade, the new compute nodes need to be (re-)provisioned. During the (re-)provisioning process, the cluster is typically unavailable. Another common challenge is process control. In a cluster environment it is more difficult to identify and control processes than on a single system. The implication is that administrators spend more time taking care of rogue 'runaway' processes that generally are more common when running distributed applications.

Cluster Configuration

The benchmark results presented in the following were obtained on an Intel® Cluster Ready certified cluster running Scyld ClusterWare 4.20. Scyld ClusterWare is a cluster management solution from Penguin Computing that is fully compatible with RedHat Enterprise Linux and CentOS. Scyld ClusterWare implements a lightweight provisioning model. Compute nodes boot over the network from a master node, which acts as a single point of control. Avoiding a local OS installation on every compute node ensures

configuration consistency and allows for easy node replacement and cluster scalability. Moreover Scyld ClusterWare provides a unified process space across the entire cluster, simplifying application and process control.

The servers used for the benchmarks were four Intel Xeon based servers from Penguin Computing's Relion 1600 series. Each server was equipped with two dual core Intel Xeon 5160 CPUs, running at a clock speed of 3.06 Ghz. The standard ANSYS benchmark suite was used. A description of the models in the benchmark suite that are run in non-distributed mode can be found on the ANSYS web site (<http://www.ansys.com/services/hardware-support-db.htm>). A description of the models used for ANSYS running in distributed mode is shown in table 1.

Name	Description	Performance Notes
BMD-1	400k DOF DSPARSE solver, static analysis	Medium sized job, should run in-core on all systems
BMD-2	1 MDOF iterative solver job; Uses JCG simple preconditioner.	Shows good scaling due to simple preconditioner
BMD-3	2 MDOF Static analysis, uses PCG iterative solver, SOLID92 elements. Good Workbench representative problem	Shows good parallel performance for iterative solver; uses MSAVE,on feature, cache friendly
BMD-4	3 MDOF larger DSPARSE solver job	Tricky job for DSPARSE when memory is limited; shows I/O as well as CPU performance; shows benefit of large memory
BMD-5	5.8 MDOF large PCG solver job	Shows good parallel performance for iterative solver on a larger job; MSAVE,on feature; cache friendly
BMD-6	1 MDOF lanpcg solver modal analysis (new iterative modal solver)	Uses assembled matrix with PCG preconditioner chosen to maximize speedups
BMD-7	5 MDOF static analysis, uses SOLID45 elements which are NOT MSAVE,on compatible elements	Best test of memory bandwidth performance; lower mflop rate is expected because of sparse matrix/vector kernel

Table 1: Benchmark Models used for distributed ANSYS runs

Hardware Components and their Impact on ANSYS Performance

Memory

Before an ANSYS model can be solved, it must be decomposed. The decomposition step factors a sparse matrix required for a solution. This decomposition cannot be executed in parallel and the performance of the decomposition phase greatly impacts the performance of the entire simulation run. The size of the sparse matrix can vary drastically depending on the number of degrees of freedom in a model. Ideally it can be stored entirely in RAM (Random Access Memory), as RAM access is orders of

magnitude faster than access to local disk. If the RAM installed on a system can accommodate the entire sparse matrix the solver is typically run in *in-core* (IC) mode. If the amount of available memory is not sufficient to accommodate the entire sparse matrix, ANSYS is forced to write the sparse matrix to a file, passing through the disk I/O subsystem. In this case the run is best run in *out-of-core* (OOC) mode. The decomposition consists of only one addition and two multiplication instructions per matrix element. Modern CPUs are capable of performing several of these instructions per clock, making write I/O performance the gating factor.

While an optimal hardware configuration would allow for the decomposition of all models *in-core*, the cost of the required memory configuration may exceed the budget allotted for a compute cluster. In this case write I/O speed to local scratch space, greatly impacts solver performance.

Figure 1 shows benchmark results for ANSYS' SMP (single system) benchmarks for memory configurations of 8GB, 16GB and 32GB. A single SATA drive was used for local scratch space.

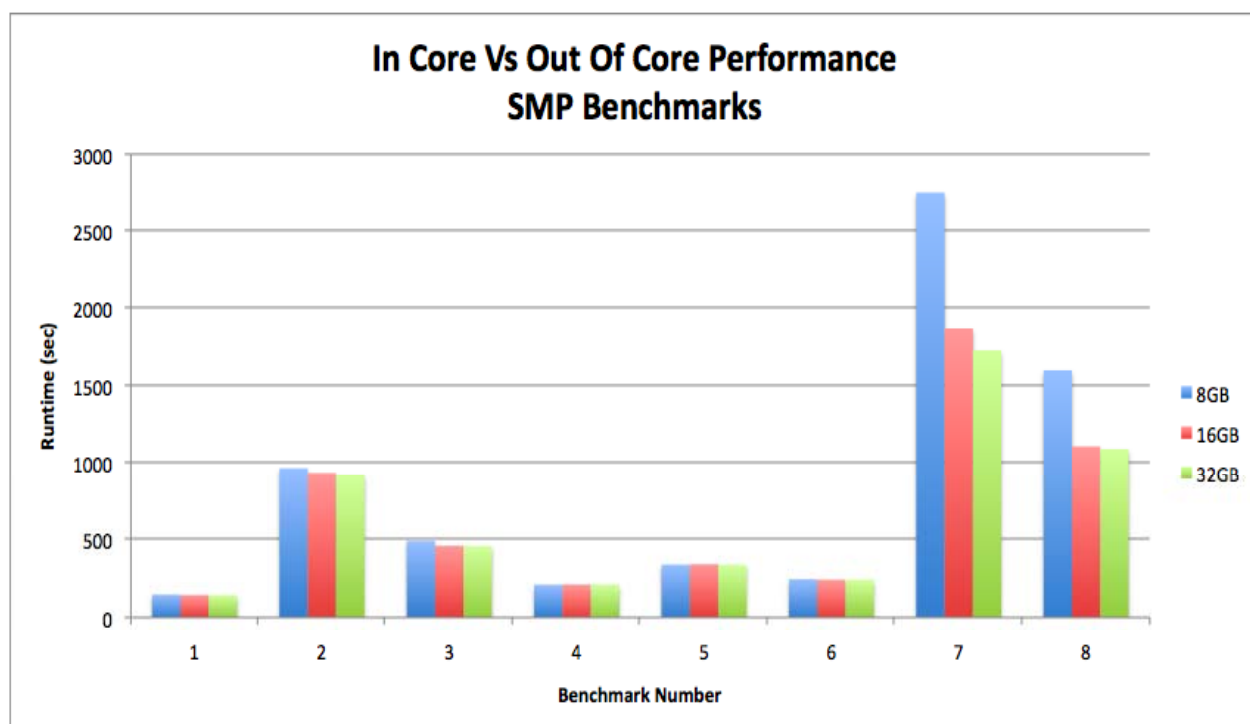


Figure 1: Impact of System Memory on Performance

As expected, the memory configuration had the biggest impact on performance for the 'larger' models BM-7 and BM-8. The drop in runtime for these models can be attributed to the fact that these models were solved *out-of-core* for the smaller 8GB memory configuration. Moving from an 8GB to a 16GB configuration resulted in a 32% performance increase as the problem was now solved 'in-core'. Adding

memory when the memory configuration already supported an *in-core* solution only resulted in a marginal performance gain.

Disk Performance

To illustrate the performance impact of different local storage configurations for scratch space for a model that runs *out-of-core*, the BM-7 benchmark was run on a node configured with 8GB of RAM.

The results of three SATA (Serial Advanced Technology Attachment) based disk configurations are shown in table 2. SATA is a bus designed for the transfer of data to and from hard disks. It is the most common bus used in low-cost commodity systems. RAID0 (Redundant Array of Independent Disks, level 0) is a disk configuration consisting of multiple disks forming a *disk array*. To the OS the *disk array* looks like a single disk. The disks in the *disk array* are accessed in parallel for increased I/O performance. The capacity of the array is the sum of the capacities of the individual drives. The downside of a RAID0 configuration is that a drive failure results in complete data loss. For scratch disks, this is generally acceptable because scratch space is only used for temporary results. The I/O operations to the RAID array are either controlled at the hardware level through an I/O controller or by means of a software configuration. For the benchmark runs presented in table 2 a low cost I/O controller card was used. With the exception of the last set of benchmarks all storage volumes were formatted with the ext2 file system. Since most 1U rack mount servers can support up to four drives, statistics for configurations of one, two, and four disk drives with a capacity of 160GB each are provided. In addition to the wall clock time, the table also shows statistics for the effective I/O rate. This metric can be obtained from ANSYS and allows for the correlation of the elapsed wall clock time with the respective disk configuration.

A good SATA hard disk is able to deliver a raw I/O performance of 50 - 80 MB/s. The observed I/O rates suggest that the disks are performing well. However, it is important to keep in mind that the effective I/O rate measured by the application is not the actual speed at which data is written to or retrieved from disk. The Linux I/O subsystem attempts to cache frequently accessed sections of files in memory in order to speed up I/O performance. As this caching mechanism is transparent to applications, ANSYS is not able to measure raw I/O speed to disk. In situations where a model is run completely *in-core*, ANSYS reports very high effective I/O rates, indicating that for *out-of-core* runs the I/O rate reported by ANSYS is not related to the performance of any read/write operations to disk. Generally, the I/O rates reported by ANSYS are slightly faster than the 'real' I/O rates for I/O operations to hard disk.

Configuration	Wall Clock Time (sec)	Effective I/O Rate (MB/s)
Single SATA	2746	84
Dual SATA	2244	157
Quad SATA	2157	184

Table 2: Benchmark BM-7 with SATA drives and a hardware controlled RAID0 configuration

Table 2 shows that a RAID0 configuration consisting of two disks increased application performance by 18% and I/O performance by 87%, compared to a single disk configuration. Considering the fact that SATA disks are cheap, moving from a single drive to a RAID0 array of two SATA drives is a very cost effective way for improving performance for *out-of-core* runs. The RAID0 configuration consisting of four disks only yields another 4% increase in application performance and a 17% increase in I/O performance.

An alternative to SATA based hard disks, are hard disks that are connected through a SAS (Serial Attached SCSI) bus. SAS drives spin with 15,500 RPMs, significantly faster than SATA drives that spin with 7200 RPMs. While SAS disks deliver significantly higher I/O performance than SATA disks, they are more expensive and have smaller capacities. SAS drives with a capacity of 73GB were used for the benchmarks. Typically this amount is sufficient for scratch space. The same hardware controller card that was used for the SATA drives was used to control the RAID0 configuration for SAS drives.

Configuration	Wall Clock Time (sec)	Effective I/O Rate (MB/s)
Single SAS	2257	150
Dual SAS	2050	247
Quad SAS	2043	249

Table 3: Benchmark BM-7 with SAS drives and a hardware controlled RAID0 configuration

Table 3 shows that a disk configuration with a single SAS drive delivers almost the same performance as a RAID0 array that consists of two SATA drives. Moving to a RAID0 configuration with two SAS drives increases application performance by 9% and I/O performance by 65%. A RAID0 configuration consisting of four SAS disks does not show any significant performance increase. This is counterintuitive as the I/O performance of RAID0 configurations typically scales up to more than four disk drives. Moreover, the application is very I/O bound, making it highly unlikely that the observed saturation of the I/O rate is related to the data output rate of the application. The SAS bus that connects the system to the I/O controller delivers a performance of 3Gb/s (384MB/s) and can therefore also be ruled out as a potential bottleneck.

The reason for the limited I/O scalability was the RAID controller, used for the benchmarks. The specification for the controller states that the card is rated at 250 MB/s, right in line with the observed limit. The benchmarks were rerun with a software based RAID0 configuration, taking advantage of the RAID capabilities available on Linux systems. Not only is this option much cheaper than the addition of an expensive RAID card to every node; it also allows for an easier, more centralized volume management.

Configuration	Wall Clock Time (sec)	Effective I/O Rate (MB/s)
Single SAS	2257	150
Dual SAS	2023	265
Quad SAS	1921	403

Table 4 Benchmark BM-7 with SAS drives and a software controlled RAID0 configuration

Table 4 shows results that were obtained with a software based RAID0 configuration. The I/O rate for the dual drive configuration exceeds the limit of the I/O rate observed with the hardware controlled RAID by 7%. The benchmark on the RAID0 array with four SAS drives shows an I/O rate improvement of 52% and a performance improvement of an additional 5%, over the software RAID0 configuration with two SAS disks.

The benchmark for four SAS drives in a RAID0 configuration was rerun with a RAID0 volume that was formatted with the XFS file system. As expected, the measured effective I/O rate was 403 MB/s, as observed with the ext2 file system. The elapsed wall-clock time for the application run was 1822 seconds, a 5% performance increase over the same configuration with an ext2 file system.

Solver Scalability

Clock rates on modern CPUs are not increasing as quickly as they did in previous years. Now CPU manufacturers are focusing on adding an increasing number of cores to their CPUs. To fully take advantage of a commodity cluster, applications must scale across CPUs/cores within a single node, as well as across many CPUs/cores on multiple nodes. ANSYS is an application that scales on a single as well as across multiple nodes. It is therefore well suited to be run in a cluster.

Solver Scalability on a Single System

Solver scalability on one system was tested on a node with 8GB of memory and 4 SAS disks configured in a RAID0 array. The RAID was software controlled and the RAID volume was formatted with the XFS file system. The scalability varies with the solver, which in turn depends on the model that is simulated. Tables 5 and 6 illustrate the scalability of the *sparse* solver for benchmarks BM-1 and BM-7.

Number of Processes	Wall Clock Time (sec)	Solver Rate (MB/s)
1	153	4865
2	122	6812
4	105	8021

Table 5: Benchmark BM-1 (Sparse Solver)

Number of Processes	Wall Clock Time (sec)	Solver Rate (MB/s)
1	1822	5699
2	1112	10057
4	803	14897

Table 6: Benchmark BM-7 (Sparse Solver)

Solver Scalability on Multiple Systems

The ultimate goal when running an application on a cluster is to achieve scalability across multiple nodes. For this purpose ANSYS can be run in a mode that distributes the computational workload of one solver run across multiple systems. In this mode, messages are exchanged over a network fabric to allow processes to communicate with each other. Obviously interconnect performance can impact the performance of ANSYS solvers that are run in distributed mode. When one thread of an application needs to communicate with a thread running on another node, it generally cannot make progress before receiving the information requested from the other thread. Latency and bandwidth of the interconnect fabric are the factors that influence application performance. The most common network fabrics used in HPC clusters today are Ethernet and Infiniband. While Ethernet is more common and delivers reasonable performance at low cost, Infiniband offers superior latency and bandwidth.

At a source code level, inter-process communication (IPC) of distributed processes is implemented by means of the Message Passing Interface (MPI). While MPI itself is only a specification, ISVs can choose from multiple MPI implementations. ANSYS 11.0 incorporates HP's MPI implementation. Besides delivering good performance, HP-MPI provides several interesting features, for example the ability to easily switch between network fabrics or the ability to use a secondary network fabric as a fallback.

For benchmarking solver scalability the benchmark BMD-4 was chosen. BMD-4 is a relatively large model (about 3 Million DOFs) that makes use of the distributed *dsparse* solver. Each node in the cluster was configured with 8GB of RAM. The nodes were using a software controlled RAID0 configuration consisting of four SAS drives.

Parallel Scaling of ANSYS Using Forced OCC Mode

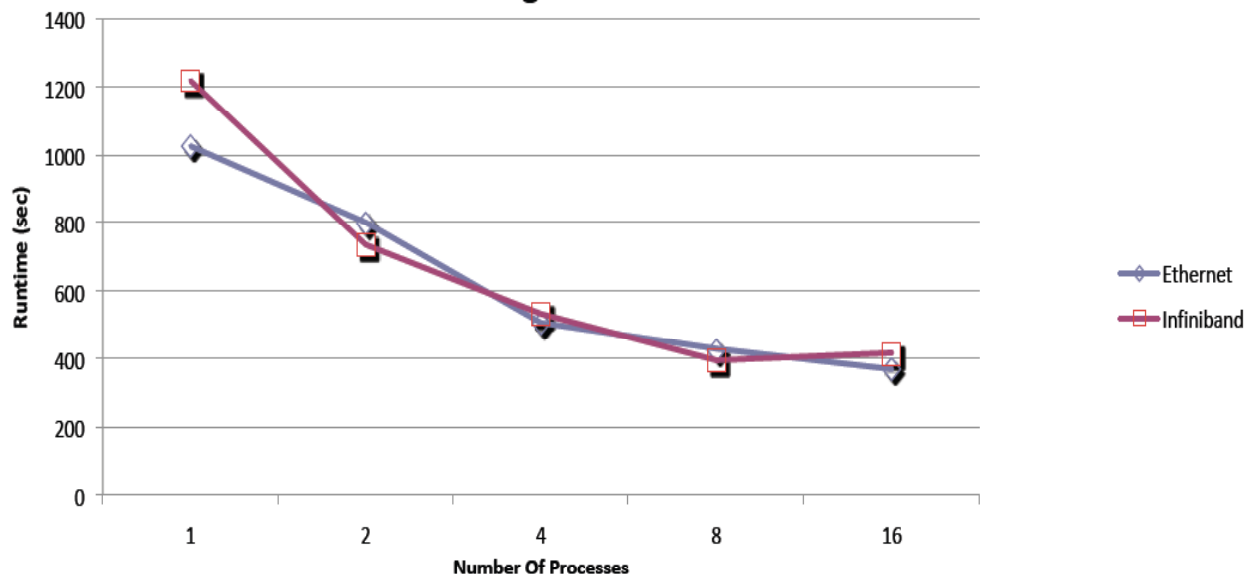


Figure 2: Distributed Solver Scalability

Figure 2 illustrates the scalability of the BMD-4 benchmark. The job scales well on both fabrics. The biggest performance gain is obtained moving from one to four distributed processes. After that the performance gained by adding additional cores diminishes. The overhead of inter-node communication reduces the performance gained through parallel processing. For this benchmark the performance gained by using an Infiniband interconnect is not significant. It is also important to note that the cores used for this set of benchmark runs were allocated *round-robin*. With the *round robin* method each process is launched on a core on a different system. After four cores on four systems have been allocated the algorithm ‘wraps around’ and allocates the next core on the first node in the set and so forth. With the round robin method the amount of memory available to each process is maximized.

In a second set of tests a *node packing* method for distributing processes onto cores was investigated. When scaling across an increasing number of cores, up to four processes (one process per core) were started on one node before additional processes were placed on another node. Using as many cores as possible on one system allows for taking advantage of shared memory optimizations in HP’s MPI implementation, avoiding unnecessary inter-process communication over the network. On the other hand less memory is available on each node if multiple processes are launched on one system. Fig. 3 shows that the latter effect outweighs the benefit of the more effective inter-process communication. Better performance is achieved with the *round-robin* allocation method.

Wallclock Time Relative to Packing Method over Infiniband

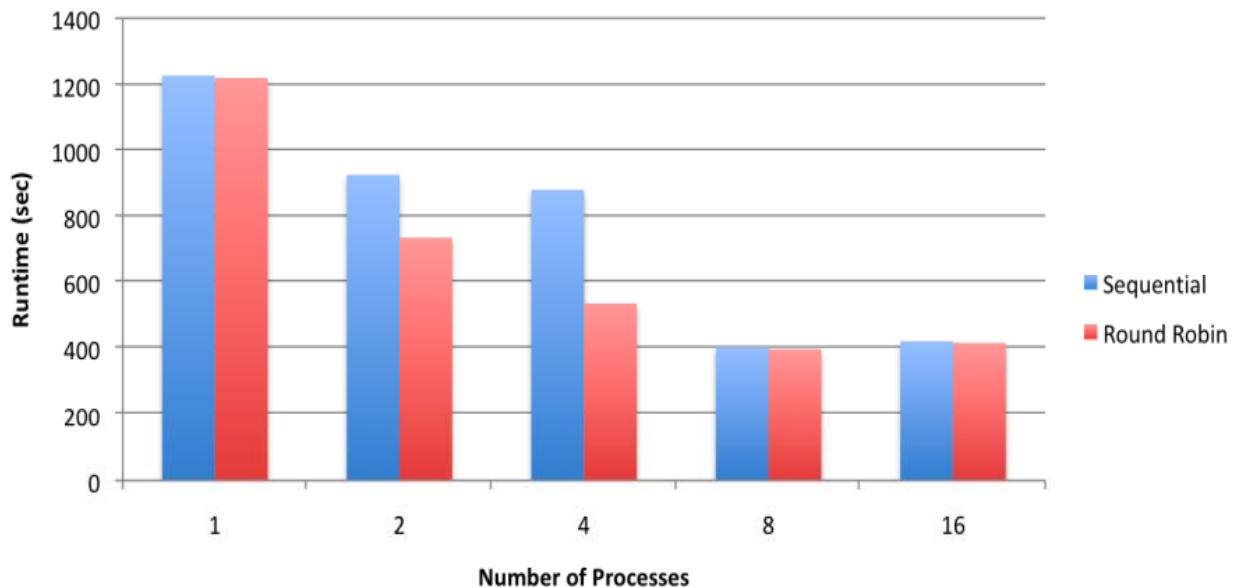


Figure 3: Performance Impact of Round Robin vs. Sequential Core Allocation

Throughput

The presented benchmark results are useful for determining the optimal set of conditions for running individual ANSYS jobs. They do not consider the more 'real-world' case of many simultaneous ANSYS runs. In the example given, optimal performance for an individual distributed ANSYS run over a gigabit ethernet fabric is achieved by using all cores in the cluster. On the other hand, sequentially running four jobs, with each job utilizing sixteen cores, takes significantly longer than a concurrent run of four jobs, where each job uses four cores. As this example illustrates, it typically makes more sense for real-world scenarios to optimize for throughput rather than for performance of individual jobs. When optimizing for throughput the performance of individual jobs has to be balanced with the number of simultaneous jobs running on the cluster.

Summary

To find an optimal hardware configuration for ANSYS, a good understanding of the models representing the typical ANSYS workload is required. It is particularly important to understand a model's memory requirements, as an *in-core* solution of a model always delivers significantly better performance than an *out-of-core* solution. If a cluster cannot be configured with enough memory for solving models *in-core*, disk I/O performance is a key factor impacting ANSYS performance. An additional critical performance

factor for distributed ANSYS runs is the distribution scheme of processes. For 'real-world' performance considerations it is important to keep in mind that job throughput may be more important than performance of individual jobs. To optimize a configuration for throughput the optimal balance between the runtime of an individual job and the number of concurrent jobs in the cluster has to be found.