WHITE PAPER

ANSYS® FLUENT on Advanced SGI® Architectures

By Dr. Mark Kremenetsky*

Abstract

ANSYS FLUENT software contains the broad physical modeling capabilities needed to model flow, turbulence, heat transfer, and reactions for industrial applications ranging from air flow over an aircraft wing to combustion in a furnace, from blood flow to semiconductor manufacturing, and from clean room design to wastewater treatment plants. Special models that give the software the ability to model in-cylinder combustion, aeroacoustics, turbomachinery, and multiphase systems have served to broaden its reach. ANSYS FLUENT solutions provide volume parallel capabilities to support multiple users, with complete flexibility to deploy and use the software wherever there are distributed resources.

Nevertheless execution of solutions requires extreme care in allocating hardware resources. The topic is even more important given hardware variety today, ranging from single node multi-core workstations through multiple nodes clusters to single image many-core systems addressing very large memory space. This paper will explore MPI performance, core frequency, choice of a network topology, memory speed and use of hyper-threading of such solutions to establish guidelines for running ANSYS FLUENT on advanced SGI computer hardware systems.

TABLE OF CONTENTS

1.0 About the SGI Systems

1.1 SGI Altix® ICE 8400

The SGI Altix® ICE integrated blade cluster was designed for today's data intensive problems. This innovative new platform from SGI raises the efficiency bar, easily scaling to meet virtually any processing requirements without compromising ease of use, manageability or price/performance. SGI Altix ICE delivers unsurpassed customer value, with breakthrough density, efficiency, reliability and manageability.

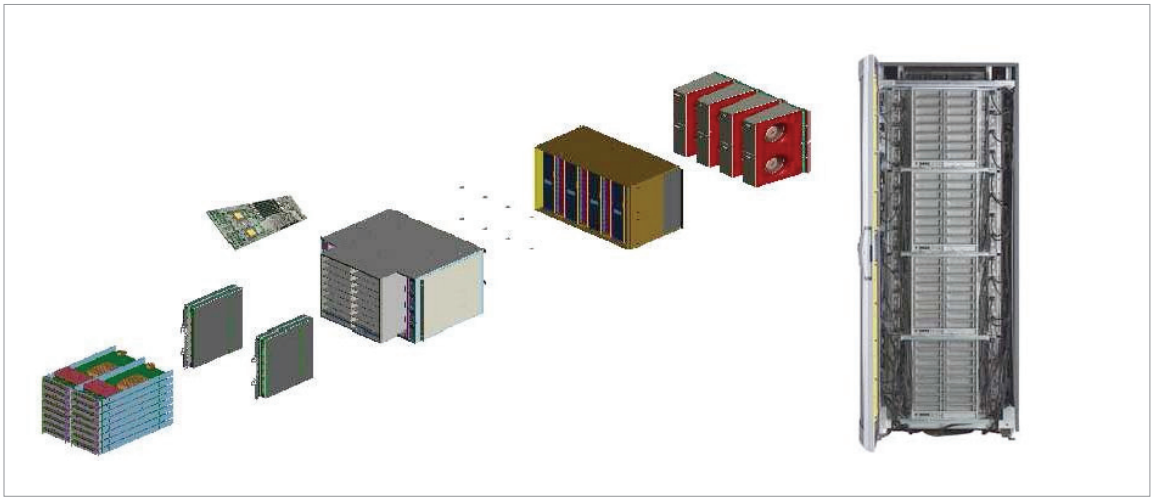


Figure 1: SGI Altix ICE cluster and IRU

SGI Altix ICE 8400 used in the benchmark (Intel)

- 256 dual-socket nodes, each socket with 6-core Intel® Xeon® Processor X5680 (12MB) Cache, 3.33 GHz, 6.40 GT/s Intel® QPI, total 3072 cores
- Total Memory: 3TB Speed: 1333 MHz (0.8 ns)
- SGI Tempo Admin Node 2.2
- SLES11SP1 OS, SGI ProPack 7SP1
- Hyper-threading enabled
- TurboBoost was enabled

1.2 SGI Altix® UV 1000

SGI Altix® UV scales to extraordinary levels-up to 256 sockets (2,048 cores, 4096 threads) with architectural support to 262,144 cores (32,768 sockets). Support for up to 16TB of global shared memory in a single system image, enables Altix UV to remain highly efficient at scale for applications ranging from in-memory databases, to a diverse set of data and compute-intensive HPC applications. With this platform, it is simpler for the user to access huge resources for programming via a familiar OS, without the need for rewriting their software to include complex communication algorithms.

Figure 2: SGI Altix UV 100, UV 1000 SMP

SGI Altix UV 1000

- 32 dual-socket nodes, each socket with 6-core Intel® Xeon® Processor X7542 (18MB Cache, 2.66 GHz, 5.86 GT/s Intel® QPI), total 384 cores
- Total Memory: 2TB Speed: 1067 MHz (0.9 ns)
- SLES11SP0 OS, SGI ProPack 7SP1
- Hyper-threading not available
- TurboBoost was used

To ensure that SGI Altix ICE and Altix UV users obtain the best possible performance out of their systems, SGI® ProPack is available with every SGI system. SGI ProPack 7 provides functionality in four areas: accelerating applications, enabling parallel programming, supporting real-time performance and analyzing system resources.

2.0 ANSYS FLUENT Overview

ANSYS FLUENT software is a leader in the engineering simulation industry in the number of complex physical models offered for solution on unstructured meshes. Combinations of elements in a variety of shapes are permitted, such as quadrilaterals and triangles for 2-D simulations and hexahedra, tetrahedra, polyhedra, prisms and pyramids for 3-D simulations. Meshes can be created using ANSYS or third-party meshing products and, in the case of polyhedra, via automatic cell agglomeration directly within ANSYS FLUENT. Meshes containing many cells, even over a billion, can quickly be automatically partitioned when read into ANSYS FLUENT software running on a compute cluster. Additional built-in tools can be used to further manipulate meshes.

Inside ANSYS FLUENT, sophisticated numerics and solvers — including a pressure-based coupled solver, a fully segregated pressure-based solver and two density-based solver formulations — help to ensure robust and accurate results for a nearly limitless range of flows. Advanced parallel processing numerics can efficiently utilize numerous multi-core processors in a single machine and in various machines on a network. Dynamic load balancing automatically detects and analyzes parallel performance and adjusts the distribution of computational cells among the processors so that a balanced load is shared by the CPUs even when complex physical models are in use. ANSYS FLUENT is available on Windows®, Linux® and UNIX® platforms.

2.1 Parallel Processing Capabilities of ANSYS FLUENT

Parallelism in computer systems exists in two paradigms:

- Distributed Memory Parallelism (DMP) uses the MPI Application Programming Interface (API) which focuses on an explicit physical domain decomposition. The resulting reduced size geometric partitions have lesser processing resource requirements, resulting in increased efficiency, but the size of the common boundary should be kept minimal to decrease inter-process communication.
- Shared Memory Parallelism (SMP) uses various kind of shared threads (e.g.OpenMP, pthreads, etc).

These two paradigms can simultaneously map themselves on two different system hardware levels:

- Inter-node or cluster parallelism (memory local to each node)–DMP only.
- Intra-node or multi-core parallelism (memory shared by all cores of each node).

The ANSYS FLUENT 13.0 features that use the above hybrid approach are the AMG solver, particle tracking, ray tracing and architecture-aware partitioning.

2.2 Distributed Parallel Capabilities in ANSYS FLUENT

FLUENT parallel simulations always begins with a Geometry Domain Decomposition. This technique partitions the geometry model and distributes the partitions among the cores on each processor socket. Care must be taken to minimize the boundary sizes between partitions to decrease inter-process communication. Load balancing is just as important as minimizing the communication costs. The workload for each MPI process is balanced so that each process does roughly the same number of computations during the solution and therefore finishes at the same time. Allocation of the total number of MPI processes over the nodes may be made by filling up each node's cores designated for processing first ('rank' allocation) or by distributing them in a round-robin fashion across all the nodes. Practically all domain decompositions are transparent to the user, but he/she can choose among available domain decomposition techniques (Principal Axes, Metis, etc).

2.3 Distributed Execution Control

2.3.1 Submittal Procedure

Submittal procedure must ensure:

- Placement of processes and threads across nodes and sockets within nodes.
- Control of process memory allocation to stay within node capacity.

Batch schedulers/resource managers dispatch jobs from a front-end login node to be executed on one or more compute nodes. To achieve the best runtime in a batch environment disk access to input and output files should be placed on the high performance shared parallel file system. The high performance file system could be an in-memory file system (/dev/shm), a Direct (DAS) or Network (NAS) Attached Storage file system. In a diskless computing environments in-memory file system or network attached storage are the only options. This file system nomenclature is illustrated in Figure 3.

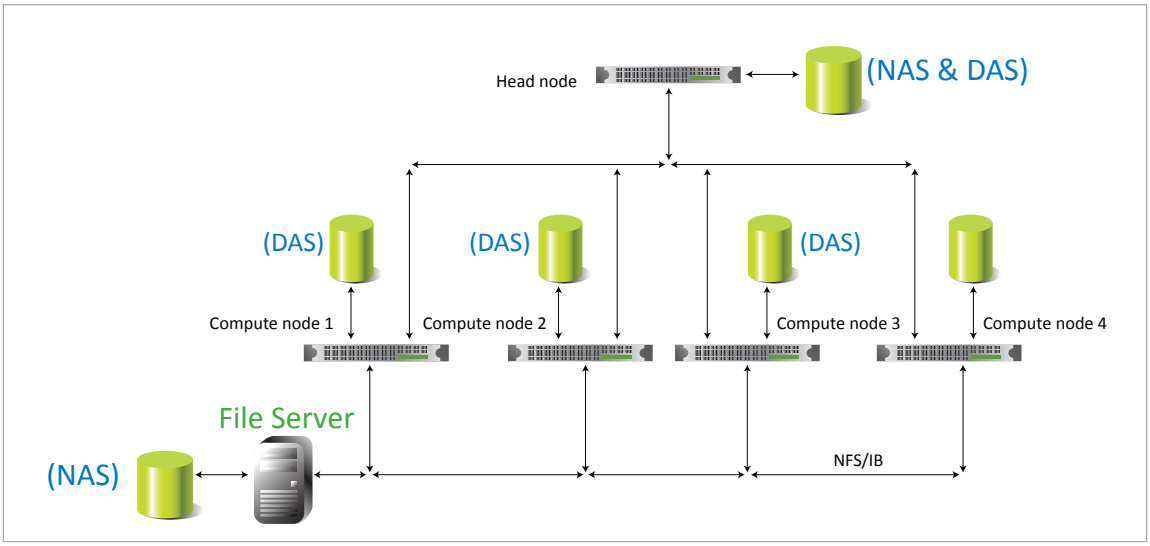


Figure 3: Example file systems for scratch space

Following is the synopsis of a job submission script.

1. Change directory to the work directory on the first compute node allocated by the batch scheduler.
2. Copy all input files over to this directory.
3. Launch application on the first compute node. The executable may itself carry out propagation and collection of various files between launch and the other nodes at start and end of the main analysis execution.

FLUENT v12 and FLUENT v13 employ both serial and parallel I/O. The later one relies on parallel MPI I/O functions of MPI. The choice of serial or parallel I/O is controlled by an extension of a stored solution file: xx.dat for a serial I/O and xx.pdat for a parallel I/O.

2.3.2 Submittal Command

The following keywords were used for the ANSYS FLUENT execution command:

FLUENT <precision> –<rshel> –p<network> –cnf =./hosts –t<nprocs> -mpi =<MPI > -l <journal> <GUI>

- precision: single (3d) or double (3ddp) precision solver.
- rshel: remote shell (rsh or ssh).
- network: type of internode network- TCP for GigE or ib for InfiniBand.
- hosts: list of nodes used in a Distributed Memory Parallel job.
- mem: size in words of explicit memory allocation for in-core processing for each MPI process.
- MPI: MPI implementation version (default hp).
- journal: file defining the application control flow.
- -g: nNo GUI.

2.3.3 Software Environment

FLUENT v13.0.0 was used.

3.0 Results

3.1 Benchmark Examples

The benchmark used is a car body model similar to Figure (4). It has approximately 14,000,000 Compute Elements (cells) and 111,000,000 Compute Elements (cells). Segregated (pressure) single precision solver, k – eps turbulence model.

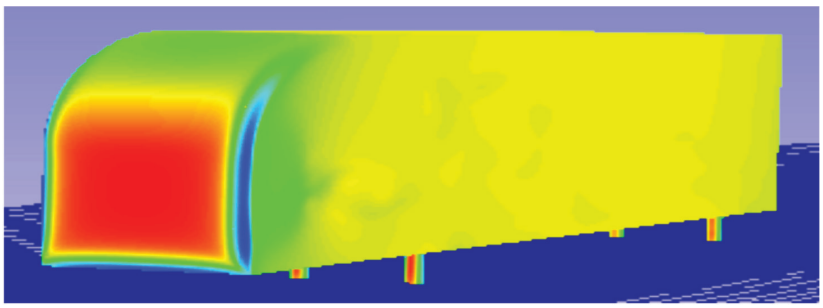
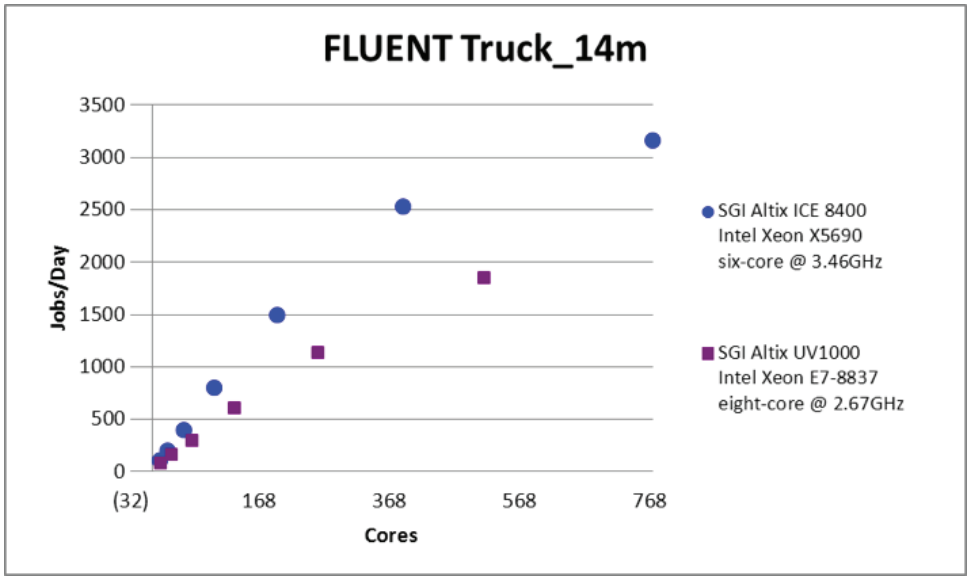
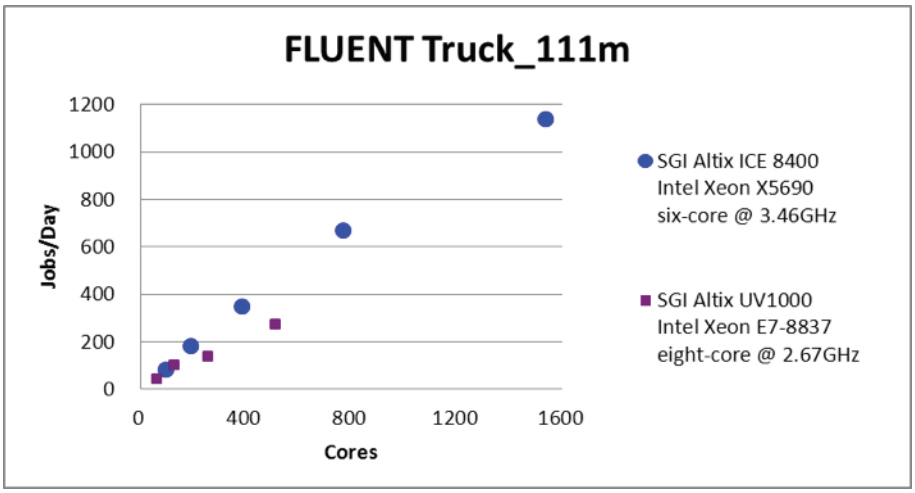


Figure 4: Truck body model

3.2 Benchmark Results



Note: the solver rating on the y-axis is the number of benchmark runs that can be performed in a 24 hour period.



Through the extensive collaborative efforts of ANSYS and SGI engineering, FLUENT has become one of the most scalable HPC applications. The above two graphs demonstrate the excellent parallel scalability achieved on SGI systems, that covering various architectures (SMP and DMP) as well as processors types.

3.2.1 Advantages of the SGI MPI Library Through SGI PerfBoost

An MPI library capability to bind an MPI rank to a processor core is key to control performance because of the multiple node/socket/core environments. Platform MPI™ currently provides CPU-affinity and core-placement capabilities to bind an MPI rank to a core in the processor from which the MPI rank is issued. Children threads, including SMP threads, can also be bound to a core in the same processor, but not to a different processor; additionally, core placement for SMP threads is by system default and cannot be explicitly controlled by users.

In contrast, SGI MPI, through the *omplace* command uniquely provides convenient placement of hybrid MPI/OpenMP processes and threads within each node. This MPI library is linklessly available through the PerfBoost facility bundled with SGI ProPack. PerfBoost provides a Platform MPI, Intel® MPI, Open MPI, HP-MPI ABI-compatible interface to SGI's MPI[1].

For FLUENT the major advantage of PerfBoost is its ability to work on a very high number of processors. SGI was able to scale one of the large FLUENT benchmarks (truck_111m) up to 3072 cores which is currently a world record in parallel scalability of ANSYS CFD solvers. To date, no other cluster has reported ANSYS FLUENT benchmark results above 2,048 cores (which was also posted by SGI).

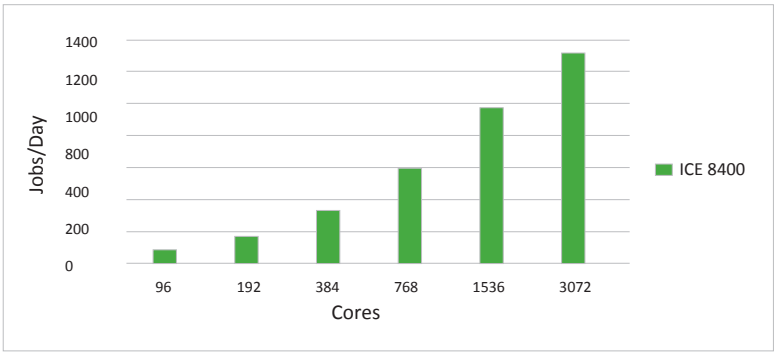


Figure 5: FLUENT Scalability World Record

[1]SGI will support customers with any issues resulting in running ANSYS using PerfBoost. ANSYS certifies its software on specific MPI libraries; SGI MPI and SGI PerfBoost technologies are not certified by ANSYS, but assumed to be compatible. ANSYS supports customers on any issues that can be reproduced without the use of PerfBoost.

There is another compelling observation in favor of SGI Perfboost. Using FLUENT built in affinity setting on a large SMP system in conjunction with a batch system can lead to unexpected threat allocations since this setting doesn't have a "cpuset" feature. Cpusets constrain the CPU and memory placement of tasks to only the resources within a tasks current cpuset. They form a nested hierarchy visible in a virtual file system, usually mounted at /dev/cpuset. Cpusets provides an essential mechanism for managing dynamic job placement on large systems. Without cpusets requests are scattered across the system thus impacting the runtime performance. In contrast, SGI® MPI uses this feature by default so all threads will be CPU-bound in a precise manner (default allocation is the user defined allocation).

There is another interesting SGI MPI feature that can be realized only on SMP systems. This feature is MPI-2 extensions and in particular the so-called one sided MPI primitives (put and get). The use of one-sided primitives can lead to significant performance improvements due to considerably lower communication cost in comparison with traditional two-sided communication primitives (send, receive, etc). This improvement comes from two sources:

- Significantly lower communication latencies.
- Reduced number of synchronization barriers.

To justify this statement we ran one FLUENT experiment with the case called aircraft_2m which is a simulation of external flow around a full body airplane configuration. Since we needed only relative evaluation the whole test was constructed only for 10 steps.

Metrics: Total wall-clock time in seconds (lower numbers are better).

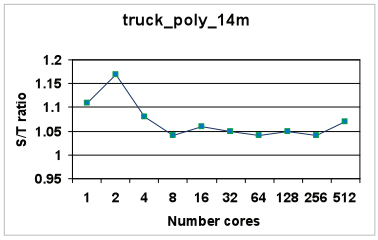
AIRCRAFT_2M			
Number of Cores	**1-sided**	**2-sided (Blocked Communication)**	**Speedup (2s/1s)**
12	40.065	42.303	1.06
24	21.242	23.916	1.13
48	12.138	14.824	1.22
96	8.164	11.467	1.41

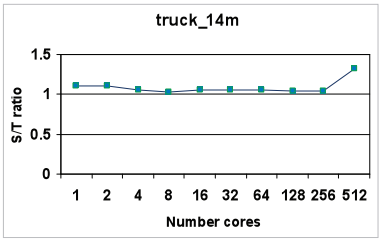
Since communication cost becomes a more significant part of the "total timing cost" of the run as one increases the cores, the use of MPI-2 primitives, as shown above, can bring a significant performance boost on Altix UV systems.

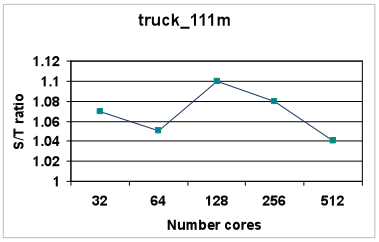
3.2.2 Effect of Core Frequency and Intel® Turbo Boost Technology

An additional data point is the added performance gained through use Intel® Turbo Boost. Turbo Boost is a feature first introduced in the Intel® Xeon® 5500 series for increasing performance by raising the core operating frequency within controlled limits depending on the sockets' thermal envelope. The mode of activation is a function of how many cores are active at a given moment which may be the case when OpenMP threads or MPI processes are idle under their running parent. For example, for a base frequency of 2.93GHz, with 1-2 cores active, their running frequencies will be throttled up to 3.3GHz, but with 3-4 cores active only to 3.2 GHz. For most computations, utilizing Turbo Boost technology can result in improved runtimes, but the overall benefit may be mitigated by the presence of performance bottlenecks other than pure arithmetic processing. The performance gain is predictably higher for the serial run (3%) than for the 8-way parallel run (.5%).

FLUENT Examples: Three standard benchmarks, Standard/Turbo Boost ratio







3.2.3 Choice of Network Technology for a Cluster

Quite often during the configuration phase of architecting a cluster based system the following question will arise; what network technology should one choose? And usually it comes down to the following two types of network technologies: InfiniBand (IB) or Gigabit Ethernet (GigE). It is known that InfiniBand is faster but relatively expensive and GigE is cheaper but slower. So the choice becomes quite complex and the outcome very often depends on the performance of the application of interest on a particular network type. So we ran a few experiments with IB and GigE on SGI Rackable system with Xeon® X5675 processors. The case used in this experiment was one of FLUENT Standard benchmarks named truck _14m.

Metrics: Rating=Jobs/Day (higher numbers are better)

truck _14m				
Number of Cores	Nodes	IB	GigE	Speedup (IB/GigE)
12	1	98.4	98.4	1
24	2	183.5	166.3	1.1
48	4	351.5	86	4.08
96	8	710.5	74	9.6

It is easy to see that if your cluster is going to be larger than 2 nodes, and performance is the main criteria, then one should consider an InfiniBand network.

3.2.4 Effect of Memory Speed

Another performance factor worth examining is the effect of memory speed on applications performance. Perhaps the best way to examine this effect is through measuring single compute node performance, thus avoiding other effects such as between nodes communication and message passing. Recall that the Intel processors can be configured to run with 1066 MHz and 1333 MHz memory speeds. For FLUENT our tests resulted in quite complex performance data. Three table entries were obtained as ratios with respect to this value. All tests used 12 cores.

Processor Speed	Memory Speed 1066 MHz	Memory Speed 1333 MHz
2.67 GHz	1.00	1.03
2.8 GHz	1.04	1.06
2.93 GHz	1.08	1.10

Table 1a: *Memory speed vs. clock rate using FL5M3 FLUENT model*

Processor Speed	Memory Speed 1066 MHz	Memory Speed 1333 MHz
2.67 GHz	1.00	1.03
2.8 GHz	1.03	1.06
2.93 GHz	1.06	1.08

Table 1b: *Memory speed vs. clock rate using FL5L3 FLUENT model*

Processor Speed	Memory Speed 1066 MHz	Memory Speed 1333 MHz
2.67 GHz	1.00	1.11
2.8 GHz	1.04	1.12
2.93 GHz	1.10	1.14

Table 1c: *Memory speed vs. clock rate using SEDAN_4M FLUENT model*

Tables 1a-c shows that the memory speed appears to affect the FLUENT models in different ratios. Furthermore it showed that the memory speed effect seemed to increase with processor clock speed.

3.2.5 Use of Hyper-Threading Technology

Intel® Hyper-Threading Technology (Intel® HT Technology) is an Intel patented technique for simultaneous multithreading (SMT). In this technique some execution elements are duplicated, specifically elements that store the executional state, where as elements that actually do the execution are not duplicated. This means that only one processor is physically present but the operating system sees two virtual processors, and shares the workload between them. Note that units such as L1 and L2 cache, and the execution engine itself are shared between the two competing threads. Hyper-threading requires both operating system and CPU support. Intel's Xeon® 5600 series processor reintroduced a form of Intel® HT Technology where 6 physical cores effectively scale to 12 virtual cores thus executing 12 threads. Thus in a Xeon® 5600 based compute node, there is a total of 24 virtual cores. In practice, an executing thread may occasionally be idle waiting for data from main memory or the completion of an I/O or system operation. A processor may stall due to a cache miss, branch misprediction, data dependency or the completion of an I/O operation. This allows another thread to execute concurrently on the same core taking advantage of such idle periods. In general, some applications can benefit as much as 30% from hyper-threading providing a very economical way in gaining extra performance. However, different software can have diverse execution profiles, thus one will expect that hyper-threading to have a variable effect on different applications. For CFD software, hyper-threading benefits will not only depend on the fact that software has different coding techniques, but also different CFD capabilities and also the different options and features of various CFD input models. Generally we may classify a software implementation of hyper-threading in the following two approaches:

- Single job in a hyper-threading environment: This implies that a job's threads execute on all 24 virtual cores of the compute nodes it requires.
- Throughput jobs in a hyper-threading environment: In this case more than one job may share the 24 virtual cores of each compute node. That is multiple jobs run in hyper-threading mode simultaneously. This may also apply on different applications running simultaneously by sharing the hyper-threaded virtual cores. The benefit of this method depends on how differently the multiple jobs or the different applications make use of the physical cores resources.

In either of the above two approaches, a performance criteria may be devised as follows, Let 'm' and 'n' be two integers such that n>0 and m>n. Thus let

'S_n' denote the elapsed time for the n-thread job in standard non hyper-thread mode of operation,

And let

'H_m' denote the elapsed time for an m-thread job under hyper-thread mode of operation.

A performance criteria for an application to benefit from Intel® HT Technology must be such that

$$H_m < S_n \quad ; \text{for } n > 0,\ m > n.$$

A typical sequence of 'm' and 'n' is where 'm=2n' and 'n' being a multiple of '8'. Note that '8' is the number of physical cores in an Intel "Nehalem" compute node. Thus a performance sequence may be based on the values of the following ratios,

$$H_{16} / S_8,\ H_{32} / S_{16},\ H_{64} / S_{32},\ H_{128} / S_{64},\ \ldots$$

Clearly each of the above ratios need be < 1 if the application is to gain any performance due to the Intel® HT Technology feature.

The second approach may become pretty complex to analyze if two (or more) different applications were to share the compute nodes under hyper-thread mode. For example one may analyze hyper-thread performance when a CFD application and a structural engineering application run simultaneously under hyper-thread mode, where the two applications execution profiles are known to be inherently different in the way system resources are utilized. Due to complexities involved in the second approach, we will focus on analyzing performance using the first approach. Hence consider the CFD application ANSYS-FLUENT where we again refer to the set of FLUENT standard benchmarks. For the first hyper-threading approach, table 2, presents solver elapsed times H2n and Sn (in seconds) for each case using hyper-threading and the standard, thread pin to physical core method, respectively, on an Altix ICE system.

n-core	sedan_2m		truck_14m	
	S_n	H_{2n}	S_n	H_{2n}
12	92	80	211	181
24	48	42	111	96
48	25	23	59	57
96	15	14-	36	35-

Lower is better

Table 2: *Hyper-threading performance vs. number of cores*

As you can see above, the influence of hyper-threading running FLUENT decreases with the use of higher number of cores. So depending on the size of the problem, hyper-threading might only be useful on one or two nodes. Because of the highly synchronized nature of computationally intensive HPC parallel codes the usefulness of hyper-threading can be very limited. It should also be noted that with commercial CFD applications a user should weigh the technical and performance benefits of hyper-threading against the possible additional licensing costs that might be incurred.

4.0 Conclusions

Computational Fluid Dynamics is a time-consuming and performance-intensive activity. The SGI Altix UV and ICE product lines delivers the scalability and performance required of complex physical modeling that includes multi-phase flows, chemistry, combustion, and spray among other physical phenomenon. This study showed how the effect on performance of core frequency, Turbo Boost, memory speed and hyper-threading can be gauged for a given dataset, in particular one observed:

- Great parallel scaling;
- Hyper-threading does not help because of communication costs beyond 8 processes;
- Infiniband can be twice as fast as GigE interconnect;
- Use of SGI MPI (through PerfBoost) compares favorably to other MPI's for high number of threads;
- Effect of Frequency and Turbo Boost are weak as this is not the only limiting factor for performance;
- DIMM speed effect is negligible in the particular case which was studied.

All these effects are definitely dependent on the dataset and solution methods used. Procurement of the right mix of resources should therefore be tailored to the range of datasets envisaged. Moreover, the performance metrics discussed here could be only one of many, others of which may include turnaround time, throughput or cost-itself comprised of acquisition cost, licenses, energy, facilities and services.

Global Sales and Support: sgi.com/global