SILICON GRAPHICS | The Source of Innovation and Discovery™

White Paper

A Framework for Evaluating High Performance Computation Solutions:
Small-node Clusters and the SGI® Altix™ 3000

Table of Contents

1.0 Introduction

Managers building new high performance computing resources face a complex set of choices. To begin with, improvements in data collection capabilities are outpacing our ability to assimilate the resulting terabytes of information, so workflow design is complicated. And the market offers a plethora of architectural options, ranging from small-node commodity clusters to custom supercomputers, each of which is optimized around specific classes of applications. The challenge is to to create an environment that can store, manage, and operate on data that might double in size every year, and efficiently run a portfolio of applications that will probably change over time.

Until recently, the two major classes of architectures, commodity clusters and large shared memory systems, involved two different operating environments. Linux® OS-based clusters brought commodity economics and the dynamic developments in the open-source community into data centers, while the proprietary operating systems running on large shared-memory systems offered mature, highly scalable, full-featured environments for high-performance computing. Both environments worked well for their target applications, but deploying both meant staffing to handle a very complex, potentially incompatible operating environment.

With the introduction of the SGI Altix family of servers and superclusters in 2003, Linux was deployed for the first time on a highly scalable large shared memory architecture, leveraging many of the commodity cost advantages. Now, research or IT managers can deploy multiple architectures to suit the full breadth of their application portfolios, all while maintaining a single industry-standard Linux operating environment. Most importantly, this choice within one unified Linux operating environment spans across many different vendors, eliminating the “hostage technology” dilemma that has fallen on many who have purchased proprietary solutions in the past.

The purpose of this article is to provide a framework to aid technical users and budget managers in assessing their computational “workflow” requirements and how they might be met by the available options, considering the architectural characteristics, total cost of ownership, and applicability to the users’ workflows. To illustrate this framework, we will consider two specific architectural approaches: small-node clusters with commodity interconnects, and the SGI Altix, a next-generation cluster combining commodity processors, memory, and Linux with large-scale shared memory through larger nodes and a dramatically faster interconnect. Both architectures take advantage of the explosion of interest in the open-source Linux operating system and its ability to enable users to create cost-effective, scalable industry standard solutions.

2.0 Workflow Analysis

Given the complexities in today’s platform market, how can managers ensure that their new deployments will drive the full productivity gains they’re expecting? Often, purchase decisions are made by evaluating peak theoretical performance numbers, single job benchmarks, or other performance measures that may not accurately depict the real compute environment or the problem to be solved.

Ultimately, an IT manager wants to create a high-productivity environment that allows users to achieve optimal results in the shortest amount of time. In some situations, this will involve maximizing productivity over a wide range of applications; in others, it will mean speeding up completion of a single critical process. A deeper analysis of the specific bottlenecks to be addressed will increase the chances of seeing real improvements in workflow efficiency.

The most important task for those making architectural choices is to first understand the real bottlenecks in the computational workflow and the computational resources required to accelerate them. In other words, how does the workflow map to different computing resources in processing, memory, and I/O? For our purposes, “workflow” is defined as the mix of applications, and flow of jobs and data within those applications, that users need to get computational results throughout their scientific methodology and discovery process. Often, a critical factor for users is time—time to solution. This is the critical bottleneck in the most important workflow—the human workflow—that surrounds the compute workflow. For a pharmaceutical company, this is literally the delay in finding and bringing its next drug to market. For the university research lab, this is the time before the next paper can be written and new grants can be funded in the “publish or perish” culture of modern research. For an automobile manufacturer, this is the time before the next car can be designed, virtually crash-tested, and brought to market. Understanding compute workflows and the value of time to solution is critical.

It is difficult to generalize, but such workflows usually fall into one of three categories (and can migrate between these categories as data sets and programming models evolve):

1) Capability workflows: these are for big, single compute jobs that require significant amounts of processing, memory, and/or I/O to complete. One example is weather forecasting, where huge data sets need to reside in main memory, accessible to all processors, for the forecast to be completed in time at a useful resolution. Another is computational fluid dynamic simulation of cars or airplanes, where results can be obtained many times faster by having all the processing work together on a shared memory data set and I/O. Such capability workflows today remain primarily in the realm of proprietary SMP and vector supercomputing systems.

2) Homogenous throughput workflows: these are for single application environments where all the jobs can be broken down into small chunks (for memory and processing) and are generally of similar size and time sensitivity. Such homogenous throughput workflows are wonderfully suited for small-node clusters, either 32-bit (with maximum addressable memory of 2-3GB per system) or 64-bit (with multi-terabyte addressable memories). The homogeneity of the job sizes and resource requirements make a small-node cluster easily tunable and a superior price-performance solution.

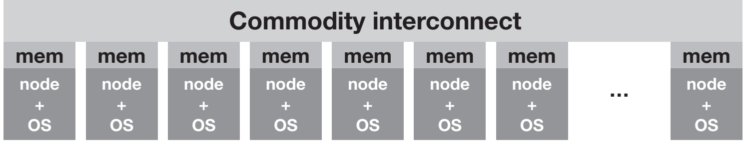
3) Heterogeneous throughput workflows: these are the users who have a mix of applications, job sizes, and time sensitivities in their compute mission. One system may be shared by a more volatile mix of applications that use very different levels of processor, memory, and I/O resources, and are thus more difficult to tune the system around. Often, such workflows are too diverse for small-node clusters to manage productively, but they do not need the largest SMP supercomputers, but they benefit greatly from medium-node sizes in a cluster deployment. An 8 to 16 processor node size has a larger pool of resources to handle the varying mix of jobs and applications on the system.

The thorough decision-maker will work with the user community directly, assessing their resource bottlenecks and asking what they could do better if they had more memory, more processing scalability, or more I/O to make their next breakthrough.

Now let's explore some of the architectural options and how they map against these workflows.

3.0 The Architectural Options

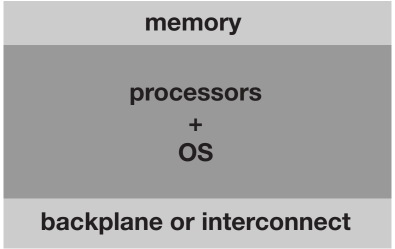
Small-node clusters can involve dozens of combinations of CPU, interconnect, and operating system technology. In this paper, we'll consider the cluster in its most common implementation: a collection of two- or four-processor systems that are loosely coupled with commodity interconnects, based on open-source Linux. Clusters' low acquisition costs and interchangeable parts have made them a growing favorite among research institutions and universities who seek the most processor capacity for their budgets. They work best on problems that are "embarrassingly parallel" – those that can be broken up into independent processes without significant internode communication – and tend to be less effective on applications requiring very large data sets or intensive I/O.



The most common compute nodes utilized in these systems are 2- or 4-p systems based on 32-bit processors such as the Intel® Xeon™ or Intel® Pentium® or, less frequently but increasingly, on 64-bit processors such as the Intel® Itanium® 2 where much larger (4-128GB per processor) memory addressing is possible. Commodity interconnects vary too: the least expensive clusters are implemented around fast Ethernet connections, but data center managers often find that higher-performance interconnects like Infiniband®, Myrinet®, or Quadrix® are required to support their applications. In these cases, the interconnect can make up 30% or more of the acquisition cost of the cluster.

Small-node clusters work well on certain classes of applications and generally offer the lowest hardware costs. They are less well suited for applications involving large data sets and inter-node communication, often have weaker overall stability and can be difficult to optimize for a varied mix of applications without overprovisioning memory and I/O.

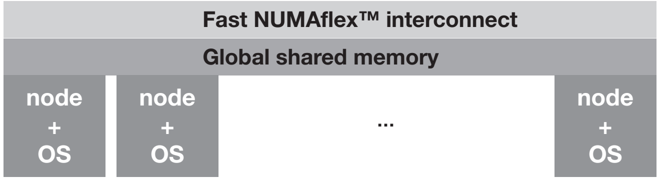
Proprietary Symmetrical Multiprocessing (SMP) Systems tend to be good at many of the things commodity Linux clusters cannot do. Their implementations are robust and mature, they support a wide variety of software and peripherals, and their architectures are well-tested in a variety of demanding production environments.



Proprietary SMP systems have limited processor scalability, but their large shared memory architectures make them well-suited for applications with large data sets. And the negatives associated with proprietary architectures have a flip side: most have cadres of loyal users who ensure that a wealth of applications in their target markets is ported and tuned to run well. Those loyal users will pay more for their proprietary solution than they would for one based on standard components, but they'll enjoy the fact that their systems run their applications well without extensive configuring and tuning, at the highest levels of production stability, and are simple to administer.

SGI Altix Superclusters combine the industry-standard components of clusters with shared memory, a supercomputer-class interconnect fabric, and true production environment capabilities. They consist of large (up to 128p) shared-memory nodes based on the Intel Itanium 2 processor, tightly coupled with the high-speed NUMAlink™ interconnect. In addition, the Altix system uniquely supports global shared memory across nodes, so that even large clusters can operate on a single shared data set in memory.

These systems support industry-standard Linux with extensions that facilitate scalability, efficient resource utilization, high-performance I/O, and excellent stability for mission-critical deployments. They are designed to combine high-performance computing with excellent big-data capabilities, and they work well on both compute-intensive and I/O-intensive codes. SGI Altix tends to have a hardware acquisition cost that is substantially less than proprietary SMP solutions, but somewhat more than commodity small-node clusters. It is well-suited for a wide variety of demanding applications, and can offer a lower total cost of ownership since real user productivity is often much higher and overall costs for power, space, system administration, and the like will often be much lower than those costs for a similarly performing small-node cluster.



4.0 Architectural Characteristics and Performance

Given these options, how does a manager determine which architectures to deploy? Whether a system supports 32 or 64 bit addressing, shared memory, a high-speed interconnect, and high-performance I/O will have an enormous impact on the types of applications that will run well. Ideally, the manager wants to see improvements in productivity and maximum utilization of resources at a reasonable cost. It may be that one architecture works best for the entire mix of applications. Alternatively, the manager with a broad application mix might choose to deploy both small-node clusters and systems with large shared-memory nodes, each running the applications best suited for them in different compute workflows.

Comparing Proprietary SMPs, Small Node Clusters, and Altix

	Proprietary SMP	Small-node cluster	Altix
Scalability - scaling out	fixed	unbounded	unbounded
Scalability - scaling out	64p	4p	128p
Max memory	varies	2GB per thread (32-bit systems)	8TB per system
Memory architecture	shared memory	distributed memory	distributed shared memory: large shared memory SSI nodes; shared memory across clusters
Latency (MPI send/receive)	varies	varies: Gbit Ethernet 50-100us Myrinet 6.3us Quadrix 5us	1.3us
Interconnect bandwidth (through MPI)	varies	varies: Gbit Ethernet 5-10MB/sec Myrinet 248MB/sec Quadrix 340MB/sec	1.5GB/sec
Programming models (through MPI)	shared memory models	MPI	all major programming models
Ideal for	tuned applications mix of applications	embarrassingly parallel apps; single application	shared memory and I/O intensive apps; mix of apps
System manageability	varies	difficult	easy
Production stability	varies	weak	strong
Cost	expensive to acquire and run	cheaper to acquire; more expensive to program and run	moderately expensive to acquire; cheaper to program and run

Processor architecture- 32-bit or 64-bit

Most small node clusters are based on 32-bit processors like Intel's Pentium, which support a maximum memory of 2GB per thread. Of course, since these systems can have tens or hundreds of nodes, they can be configured with virtually infinite amounts of memory, but data will ultimately need to be distributed across the nodes in 2GB chunks.

With 64-bit computing, the theoretical maximum memory is three orders of magnitude larger at 4 exabytes. Altix nodes can have substantially higher processor counts than other cluster nodes based on Intel , so they don't compare directly; but they currently support up to 4TB per node.

Memory architecture

Historically one of the central problems of supercomputer design, memory architecture can make an enormous difference to overall performance and real user productivity. As described above, memory in small-node clusters is distributed across all of the nodes in the clusters. Even if a small-node cluster is based on a 64-bit processor such as Itanium 2, and therefore isn't subject to the 2GB limitation of 32-bit architectures, economics will likely drive a low memory-per-node configuration, especially on clusters with lots of nodes. Where large data sets are involved, the application must spend time swapping subsets of the data on and off disk.

A large-node system like the Altix™ 3700 system solves this problem efficiently by allowing all of the memory in the system to be accessed by any processor. Multi-terabyte data sets can be loaded entirely into memory and operated upon by all processors directly, without inefficient waiting for I/O calls and disk-swapping. In addition, the large available memory can be used as a high-performance I/O buffer, and applications can write scratch files to memory instead of disk, resulting in dramatically improved application performance.

Memory bandwidth makes a significant difference as well, especially in application environments where the data is decomposed into small chunks that must be paged in and out. A well-known drug discovery and genomic information solution provider's experience illustrates how dramatic an effect higher bandwidth can have. This company began buying Pentium processor-based cluster nodes over time beginning four years ago. As they began seeking improvements in search productivity, they decided to benchmark a Beowulf cluster of 2.2GHGz Xeon processors against an Altix 3000 with 1GHz processors. They found that the application ran seven times faster on the Altix system, largely due to its superior memory bandwidth. In the future, they will benefit further by using the large shared memory capabilities of Altix to eliminate the need for data swapping altogether.

Interconnect

Along with memory usage, the interconnect can have the most dramatic impact on overall application performance, since it influences both bandwidth and latency.

In a small-node cluster, both applications and data are divided into chunks small enough to fit within the CPU and memory limitations of individual nodes. Some applications have relatively small data files and independent processes; for example, rendering often falls into this category. Such codes, often referred to as "embarrassingly parallel" applications, run cost-effectively and well on small-node clusters.

Other problems have more demanding communication requirements, and may require moving data and instructions across the cluster in order to send results where they are needed or facilitate load balancing. In applications that are difficult to parallelize, excessive inter-node communication can rapidly become an application productivity bottleneck, particularly if the interconnect used is performance-limited.

With MPI bandwidth of 1.5GB/second, and MPI send/receive latency of 1.3us, the NUMAflex interconnect implemented in Altix offers several times the memory throughput and latency than the interconnects available for small-node clusters. These capabilities have a dramatic impact on application performance.

Programming models and development tools

Small-node clusters support distributed memory programming models such as MPI. Since more and more scientific and technical applications are being written in MPI, this provides clusters with good application availability.

The unique SGI Altix architecture, which combines a ccNUMA shared memory architecture with cluster capabilities, supports all major programming models associated with either distributed or shared memory architectures, including MPI, OpenMP™, SHMEM, and Pthreads. The market for 64-bit cluster development tools continues to evolve, and some of the available hardware platforms lack mature compilers and performance tools. Altix supports a rich and robust set of compilers and tools from Intel, Gnu, and third-party suppliers. Also, Altix supports a set of resource allocation, tuning and I/O libraries tuned to help developers take maximum advantage of the Altix architecture.

5.0 Total Cost of Ownership

Managers purchasing high-performance computing solutions will usually perform some form of price/performance analysis. Of course, mileage varies on these analyses, since performance can mean any number of things (peak GFLOPS, single job benchmarks, and application mix benchmarks) and cost can include only hardware acquisition costs or additional charges related to floor space, power, system administration, and other elements of the overall cost of ownership.

While small node clusters can offer attractively low acquisition costs, the cost of systems must be viewed in a larger scope than simply looking at the cost-per-processor allows. The balance of compute economics is changing: as hardware costs fall, software and staffing costs are rising as systems grow larger and more complex. Per-node costs for staffing (installation, tuning, and system administration), facilities (rack space and power) and software represent an increasing proportion of the overall cost of ownership. This suggests that machines with fewer nodes can be managed more efficiently and cost-effectively.

The following table summarizes a few of the variables that must be considered.

Elements of the TCO

Acquisition	Small-node clusters	SGI Altix 3000
System	PC-class servers offer lowest cost	Higher-end architecture; higher per-processor cost
Interconnect	Can add 30-50% over processor costs	Included with system; no additional costs
SW	OS and other open-source tools are free, but other applications may be priced per node, driving higher SW costs for large systems	Fewer nodes may be required; driving lower SW costs
Installation	Racking and SW installation takes longer for systems with many nodes	Fewer nodes mean lower installation costs

Operation	Small-node clusters	SGI Altix 3000
Data center	Larger systems may be required to achieve specific performance	Smaller, more integrated system requires less datacenter space
Power	Power is lower per processor, but processor count can be much higher	Smaller system requires less power
System administration and programming	Larger, loosely coupled cluster requires more administration	Smaller, shared memory system is easier to program and maintain

Hardware acquisition costs

Per-processor costs for PC-class 2- or 4-processor nodes are attractively inexpensive. To create a small-node cluster, the configuration must also include an interconnect fabric, cabling, ports, and a rack. In the past, these all had to be purchased separately and integrated on-site; now, major vendors like Dell and IBM offer a variety of pre-configured cluster solutions. A comparatively high-performance interconnect such as Myrinet will make up as much as 30% of the cluster's total acquisition cost.

SGI Altix will generally have a higher per-processor cost. The high-speed NUMAflex interconnect and shared-memory capabilities are integral parts of the system, and are included in the cost.

Software acquisition costs

Software costs are highly variable, and may be minimal with new hardware acquisitions. The Linux OS and some system software are open source and freely available. Also, large institutions may have site licenses for their applications, eliminating the need to purchase new software licenses when acquiring new hardware. When evaluating overall acquisition costs, it is important to consider that any critical middleware or application software will generally need to be installed on each node. Where software costs are incurred, this will drive costs significantly higher on small-node clusters with many nodes.

Installation

Installation costs for a small-node cluster will depend on whether a user purchases a pre-configured cluster system or decides to build his or her own. If the user takes the latter route, this will involve installing interconnect cards in each node, racking and cabling all of the nodes, and installing drivers and software on each node. If the small node cluster is based on lower-performing 32-bit processors, a similarly performing Altix might have only a single node, resulting in lower installation costs.

Software application installation involves similar economies. On Altix, applications are loaded and tuned once on a large shared-memory system. Applications must be loaded across multiple nodes on the cluster, thus increasing the time required and the associated cost. Furthermore, tuning is often more difficult on a loosely coupled cluster, where default memory and CPU allocation choices may not provide optimal performance for a given application mix. Consequently, application installation and tuning will generally require a larger effort, and therefore higher cost, on the small-node cluster.

Data center costs

Next, we look at the operational costs, including data center space, power consumption, and system administration over a three-year period. Data center space is usually allocated and charged per rack unit, whether by commercial co-locaters or by internal IT departments. Currently, a 32-processor Altix fits into a 40U rack. Since a typical Pentium processor-based cluster requires 1U for each 2P node, the same space will hold a 32P small-node cluster. But since Altix achieves similar performance with far fewer CPUs than the small node cluster, its overall space requirements are frequently smaller, particularly when compared to clusters based on 32-bit processors like Pentium.

Power consumption

Per-CPU power consumption runs higher for Altix at 188 watts than for a Xeon cluster at 92 watts. Again, if the cluster in question involves lots of lower performing processors, total power consumption could be significantly less on Altix. Clusters based on higher-performing 64-bit processors like Itanium 2 or Opteron® will be more comparable in both processor count and per-CPU power consumption, so it's unlikely that there would be much difference in total power cost between those and Altix.

System administration

System administration costs will vary significantly depending upon the applications to be run and the overall system environment. In general, a shared memory system will have lower system administration costs since it has fewer nodes, and it can manage volatile mixes of applications and jobs without complex load balancing, distributed data, and other challenges specific to clusters.

6.0 Real-World Case Studies

Making the best architectural choice requires a clear understanding of workflow needs and architectural trade-offs in meeting those needs. Two brief case studies illustrate this point. First, we'll examine a task-oriented environment in which fast single-process execution was critical to achieve faster insights on a particular problem. Next, we'll look at a throughput-oriented environment in which the goal was to optimize performance over an entire data center's mix of applications.

Workflow case study: University Materials Research Team

Our first example concerns a materials research department in a major university. Their goal is to develop new materials using molecular simulations to identify compounds worthy of additional study, then perform more detailed molecular dynamics analyses on interesting candidates.

The customer initially decided to purchase a 96-processor cluster of 2P Pentium processor-based nodes, then decided to evaluate the newly-available SGI Altix 3700. They considered a 16-processor model since they could fit that configuration into their acquisition budget.

To begin their analysis, they looked at how their major application, Amber, performed and scaled on both configurations. To support their comparisons, they used data from the published benchmarks on Amber's web site (http://www.amber.ucsf.edu/amber/amber7.bench4.html). The table below shows the relevant subset of those results.

Node Name	CPU	Compiler	Npcu	Time-per step
Small-node Cluster	2.4 GHz Pentium® 4	ifc 7	1	1.09
			2	0.56
			4	0.35
			8	0.20
			16	0.18
SGI Altix	1 GHz Itanium 2	Efc 7	1	0.68
			2	0.34
			4	0.18
			8	0.10
			16	0.07

Upon analyzing these results, the evaluation team made the following observations:

- The application runs well on clusters and does not require much communication between processes.
- The application scales relatively efficiently up to 16 processors; there is little benefit gained by running the application on more than 16 processors.
- When both machines are benchmarked with 16 processors, the Altix system runs the application more than 2.5 times faster.
- A 4-processor Altix runs the application in the same time as the 16-processor Pentium cluster.

As a result of this analysis, the team assumed that they would run the application using 16-processor subsets of the Pentium cluster and 4-processor subsets of the Altix system. Given the total processor counts of the two machines, 96p for the cluster and 16p for Altix, this means that in throughput mode the cluster can run six Amber jobs in the time that it takes Altix to run four.

Throughput Analysis

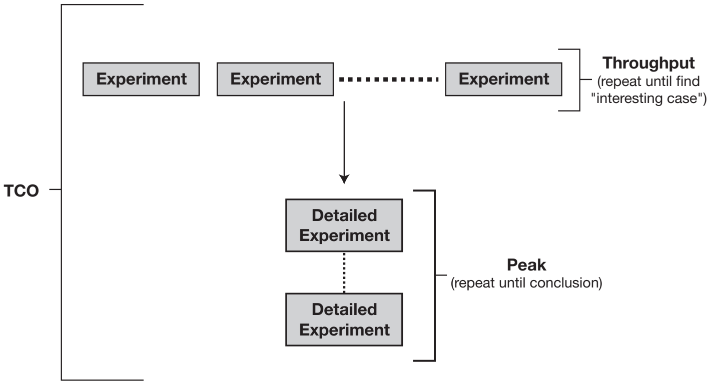
	No CPUs	Cpus Job	Time	# of jobs
96P cluster	96	16	0.18	6
16P Altix	16	4	0.18	4

At this point the cluster offers a better price-per-simulation according to the benchmarks. Most commodity cluster advocates stop their analyses here. By focusing only on overall throughput, some miss the opportunity to affect more dramatic productivity improvements by analyzing the workflow more deeply.

In this case, the team decided to analyze each component of the workflow to determine where improvements could benefit them the most. They found that the process had two distinct phases with different goals and characteristics.

During the first phase, the researcher is usually finding good candidates for further study. The researchers run multiple simulations or experiments in order to narrow down their search for a good molecule to study in more detail. This phase makes use of the target machine's throughput capabilities, and the ability to run many simulations quickly is valuable.

Once an interesting candidate has been found, the second phase begins. The research team structures and runs a very detailed simulation, waits for the results, analyzes them, fine-tunes the experiment and sends a new simulation. These complex analyses can take weeks to complete, and each step in the analysis must wait for the results of the preceding step. Thus, at this point the workflow changes dramatically and the cycle becomes sequential, rather than throughput-oriented.



Since this second phase of the workflow is the most time-consuming, and the most critical to making new discoveries, it actually drives the team's compute requirements. Their objective is to complete these detailed, iterative second-phase simulations as quickly as possible, so that they can gain insights faster.

So the team's interest shifted from throughput to peak performance. Their next step was to analyze how long it would take to run these complex sequential simulations and how many of them they could get through in a year.

Since these simulations had to be run sequentially, productivity was limited by the maximum scalability of the application to 16 processors. The team was able to fully utilize the 16-processor Altix, but resource usage on the cluster was much less efficient due to the scalability limitations of the application. They found that the simulation would take approximately three weeks on the cluster, whereas Altix could run the analysis in 8 days, dramatically increasing the number of experiments per year. Assuming that the systems run continuously, this would mean that the cluster could run 17 sequential simulations in a year of research, whereas Altix could run 45.

Task-oriented analysis

	# CPUs	Cpus/Job	Time	Experiments/yr.
Pentium cluster	96	96	3 weeks	17
16P Altix	16	16	8 days	45

At this point, the team had sufficient data to make a decision; in other cases, additional analyses might be done to evaluate the average numbers of experiments required for particular types of conclusions. The Altix solution proved in this case to be a better match for the customer's workflow, and it will allow the team to achieve faster time-to-insight for a lower overall cost of ownership.

Workflow case study: University Centralized Research Computing Center

In our second example, the customers were responsible for a research computing center supporting a wide variety of applications in physics, chemistry, and all facets of engineering. Given the broad application mix, analysis of a single workflow was of marginal use in determining the best solution. Instead, the team had to determine how best to maximize productivity across a variable workload.

The team analyzed the application portfolio and determined that some applications had minimal communication and memory requirements and ran well on clusters, and others required the large shared memory and fast interconnect offered by Altix, and a few could work reasonably well in either environment. They elected to invest their $500,000 budget into a flexible network of systems that could run all of the applications in their portfolio well. The solution included a 32-processor Altix, two 32-processor Pentium clusters, and a terabyte of storage for their data sets. The SGI professional services team worked with the customer to structure a flexible, easy-to-use scheduling process that allows each application to run on the architecture best suited for it.

There were several factors that made this mixed environment particularly successful:

- The mix of architectures can efficiently run all of the applications in the research center's portfolio. Demanding applications with large data sets run on Altix, while embarrassingly parallel throughput-oriented applications can run on the Pentium clusters.
- The decision to use both architectures allowed the customer to optimize both for the applications they run best and to limit hardware expense on the cluster. For example, since Altix is available for high-communication, large data set applications, the customer was able to avoid overprovisioning the cluster with extra memory and high-speed interconnect hardware. Instead, they were able to afford extra CPUs.
- Altix was able to fit seamlessly into the customer's mixed environment. In fact, the customer worked with SGI Professional Services to construct a centralized scheduling process that chooses appropriate target hardware based on customer-defined criteria such as memory requirements or application type. This process was simple to set up and is flexible and user-transparent.

7.0 Conclusion

To create real productivity improvements in their organizations, managers evaluating computational solutions for research centers and engineering departments will need to look beyond the usual standard benchmark and acquisition cost comparisons. As computing problems grow increasingly complex and data-intensive, the real challenge is not how to buy the most processors for the budget, but how to best map the numerous architectural options to the users' specific problems. By taking a careful look at the overall workflows driven by the organization, managers can evaluate their requirements much more accurately, and choose a solution that will drive productivity improvements where they're needed most. And most importantly, for the first time ever, there is one unified open-standards based operating environment—the extraordinary combination of Linux and Intel processors—that covers the full spectrum of architectures and the many vendors that offer them.

Corporate Office
1500 Crittendon Lane
Mountain View, CA 94043
(650) 960-1980
www.sgi.com

North America 1 (800) 800.7441
Latin America (52)5267.1300
Europe (44)118.925.75.00
Japan (81)3.5488.1811
Asia Pacific (67)6771.0290