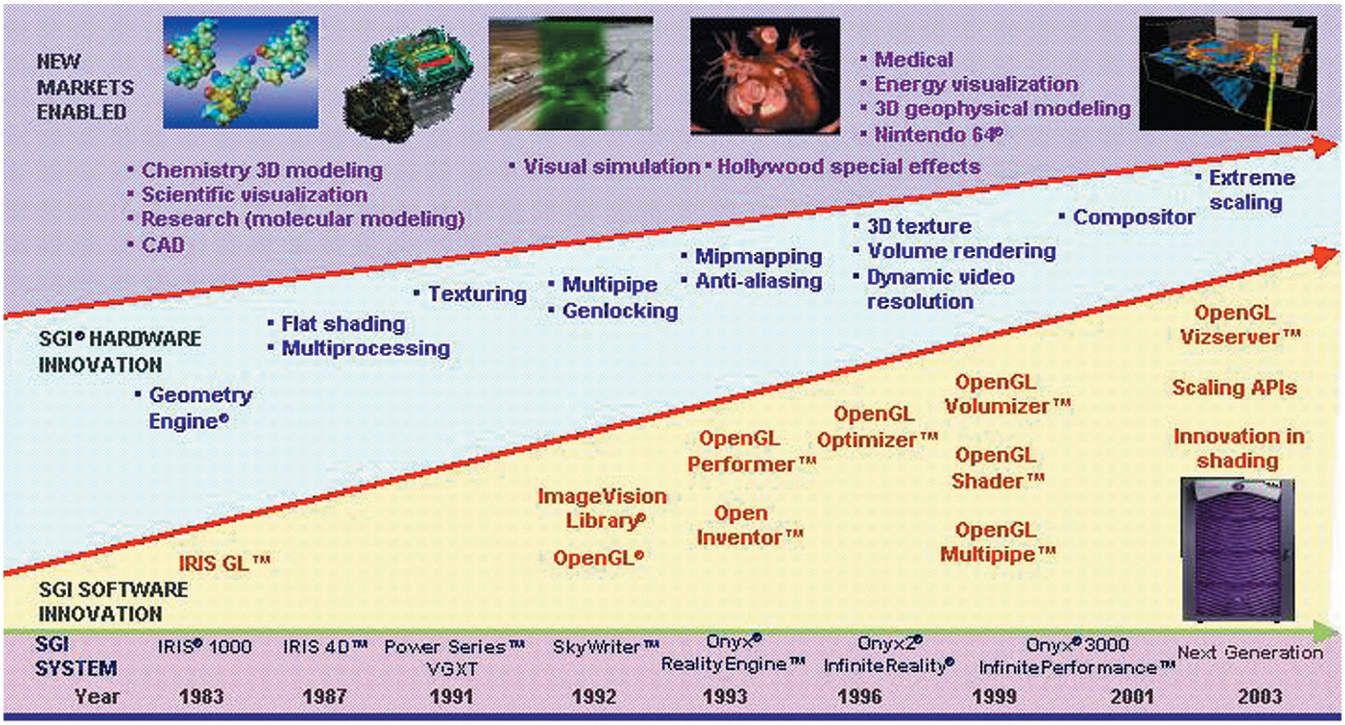
SILICON GRAPHICS | The Source of Innovation and Discovery™

White Paper

The Power of Scalable Visualization

Table of Contents



Molecules courtesy of Tripos Inc; Engine courtesy of PSA, Aechelon Technology; Heart courtesy of University Hospital of Rotterdam and Duke University; Seismic image courtesy of Magic Earth, LLC.

Scalability: Past, Present, and Future

SGI has been producing scalable compute and graphics systems for over 20 years, most of that time based on the MIPS® CPU and IRIX® operating system. From the beginning, SGI designed scalable systems to work in a Single System Image (SSI) environment, meaning that there's a single copy of the operating system running and all memory and devices are available to all processes. Initial system scaling took systems from 1 to 4 CPUs, then allowed graphics scaling in SSI to 2 graphics pipes.

With the traditional Symmetric Multi-Processing (SMP) architecture, scaling maxed out at 24 CPUs and 3 graphics pipes, which rapidly became an unreasonable constraint. In late 1995, SGI introduced the Origin® and Onyx2® families, which were the first commercial computer systems with a Cache Coherent Non-Uniform Memory Access (ccNUMA) architecture. The revolutionary ccNUMA architecture enabled systems to scale efficiently, well beyond the traditional bus-based architecture systems, by scaling system bandwidth as CPUs and memory are added. This provides balanced resources even for very large systems.

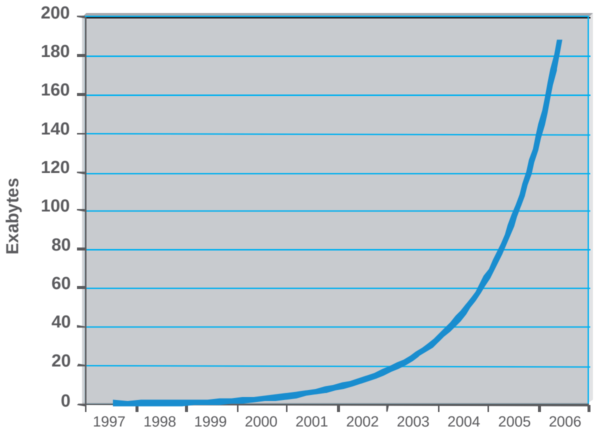
Today, Origin supercomputers scale to systems with over 512 processors and 1 TB of shared memory. In addition, the recently introduced SGI® Altix™ family, which is based on Intel® Itanium® 2 processors and standard Linux®, scales to 128 CPUs in a single instance of the operating system and up to 512 CPUs in a shared memory super-cluster. The new Silicon Graphics® Onyx4™ UltimateVision™ system enables over 32 industry standard graphics cards that can be accessed in a SSI environment, 16 times more than in any other system.

Looking forward, as data continues to grow exponentially, scalable visualization becomes much more imperative. SGI will continue to innovate to scale further in memory sizes, CPUs and graphics pipes on both IRIX and future Linux visualization environments in order to meet these ever-increasing demands in data sizes.

Why Do I Need Scalability?

In today's rapidly changing world, one constant is that problems are getting more and more complex. This phenomenon typically manifests itself as a growth in the amount of data being created and the size of a typical data set or model. Another important growth phenomenon is called Moore's law, which is the growth rate of computing power. Historically, computational components that are based on semiconductor technologies are doubling in performance every 18 months.

World Wide Production of Information



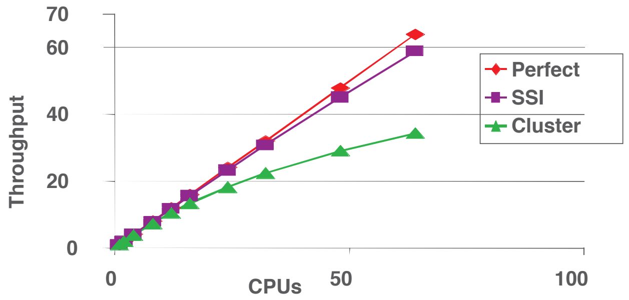
Source: Gartner

Unfortunately, in almost every industry and application segment, the growth in data far exceeds the growth in computing power. Waiting for the latest computational component to solve the increasing data problem will not work because the problems are typically growing faster than the ability of computational components to handle them. The ability to scale the fundamental components of computing to meet the ever-increasing demands becomes the most cost-effective way to resolve this discrepancy. Combining the performance growth of individual components with the ability to scale to larger and larger systems provides the best way to beat the growth in problem size and complexity.

The Dawn of GPU Scalability

SGI has been building scalable computer systems for the past 20 years. Over this time, we've seen systems grow from uni-processors to small SMP machines to ccNUMA based systems with 10s or 100s of CPUs and gigabytes or even terabytes of memory.

In a perfect world, as more and more resources are added to systems, the time to solution should decrease as a function of the amount of resources added. In the real world, however, as resources are scaled, internal bottlenecks between components can quickly dominate the time to solution. For years, SGI has built systems designed to minimize the internal bottlenecks to the point where real-world performance is comparable to the theoretical performance of the sum of the components.



Source: SGI

In the current implementation of ccNUMA, NUMAflex™ provides the fundamental system architecture that minimizes these bottlenecks. Real-world systems built using NUMAflex running real-world applications are achieving sustained performance numbers that are in line with theoretical numbers up to 1024 processors. In some cases, super-linear results have been achieved as various synergies in the architecture enable the components to run more efficiently than their single component performance.

Software is readily available for these machines that will decompose a computational problem over all the available CPUs in a system. If a problem cannot be solved fast enough on 64 or 128 CPUs, then the number of CPUs can be doubled or more to get to the solution faster. For ultimate performance, the number of CPUs needed is determined by fitting the problem set into the available CPU cache memories.

SGI is applying the same scaling expertise to graphics components. Problems that were once not feasible to solve become easy when the power of 10s or eventually 100s of graphics processing units (GPUs, also known as graphics pipes) are applied. Again, the key to scaling in GPUs is the same as CPUs. Minimizing the bottlenecks is critical. The combination of SGI's NUMAflex architecture with global shared memory, and scalable graphics architecture with multiple OpenGL® streams and flexible compositing, powers efficient GPU scaling.

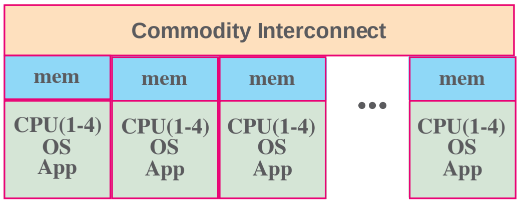
Software that once only worked with a single GPU can be modified to scale across all the CPU and GPU resources available in a system. As GPUs are added, larger and larger problems can be addressed and solved. This new paradigm of scalability is the future of visualization.

Scalability the Hard Way

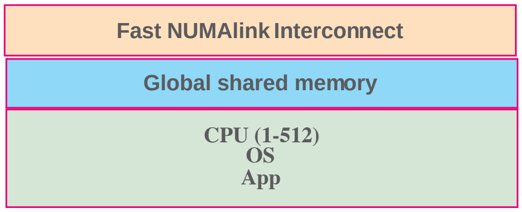
Some of the scalability that SGI provides can be done using traditional architectures by clustering multiple SSI machines together. Of course, using clusters is the hard way to achieve scalability. The difficulties start with the physical configuration of the nodes, disks, operating systems, network infrastructure, rack space, power distribution, and all the other components that SGI ships ready to plug in, boot up, and get to work. The day to day issues of managing a cluster, like nodes crashing and needing to be rebooted, processes hanging and needing to be reset, or disks failing or filling up and needing to be maintained, make clusters a significant long term challenge. In addition, technology updates on clusters are pretty much complete fork-lift upgrades because of the complications of using components with unsynchronized life cycles. A Single System Image, with a single file system, single operating system, and single view of all the processes running on the system makes working with even the largest system straightforward.

A cluster runs as many operating systems and instances of the application as there are nodes. When a problem can be easily sliced into smaller chunks that do not require much communication, this approach works well. Unfortunately, for most applications that want to take advantage of the power of visualization, typical graphics problems are extremely difficult to split up into small enough chunks that have minimal communication between them to make clusters feasible. Global shared memory and support for multiple OpenGL streams from one application remove the complications and performance issues of communicating between multiple instances of an application and an operating system.

Traditional Clusters



NUMAflex Architecture



Developing applications in a SSI based environment is very similar to developing on a single CPU and single GPU based system. A single executable can access all data in memory at once and render to all GPUs at the same time with multiple draw threads. A single debugger can attach to the executable to view the whole application state. This dramatically differs from code development on a cluster where each executable is a different process on a different node which needs a separate debugger to determine program faults. In this case, resolving inter-process timing issues, which are a challenge on a shared memory machine, becomes next to impossible to fix on distributed systems. The Heisenberg uncertainty principle is much more of a challenge in a multi-CPU, multi-GPU, multi-OS, multi-RAM environment than in a SSI environment.

What is GPU Scalability?

GPU scalability starts at the system level. SGI® machines are based on small building blocks known as bricks. A brick can be CPUs with memory, only memory, CPUs with memory and graphics cards, only graphics cards, I/O, routers, and other possible types of components. NUMAlink™ interconnect technology enables these bricks to be a single system, in contrast to connecting cluster components with a standard interconnect. The sum of all of the bricks and their connections creates a high performance memory fabric. This fabric provides the SSI capabilities – a single OS, a single large pool of memory, and a single process space, with all CPUs and GPUs accessible from each other CPU and GPU. A system can be as small as 2 processors and 2 graphics cards or as large as 100s of processors and 10s of graphics cards. Scalability allows an application to drive a single GPU with a small dataset or 10 GPUs with a 10X larger data set. Memory bandwidth, disk I/O, bus bandwidth, efficient kernel locking, and many other factors go into creating this scalability.

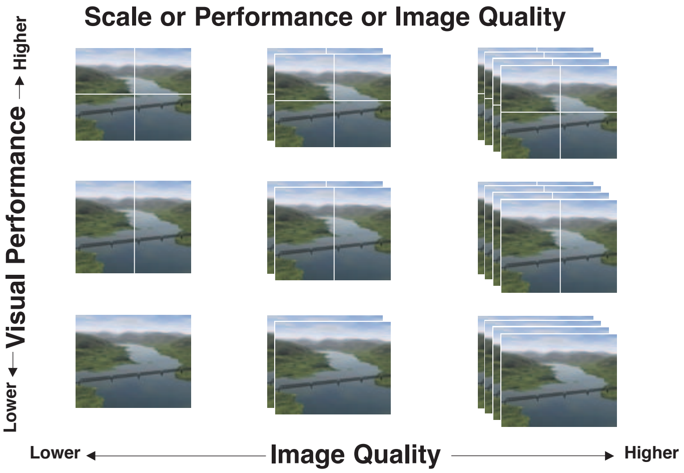


Image courtesy of British Aerospace

The ability to add GPUs into the system allows a visualization application to scale in many dimensions.

- Performance Scaling
- Resolution
- Data size

Performance Scaling:
The dimension most apparent is performance. More GPUs directly translate into more polygon transform and fill performance, allowing larger data sets to be viewed at interactive rates. As data sets grow, more graphics bricks can be added with the SGI systems growing along with the data. In addition to polygon transform and fill performance, scalability also includes pixel scalability to provide large scale display surfaces for theatres, CAVEs™, or just a single wall of a conference room. Equally, more GPUs adds increased pixel shader and vertex shader performance. Over time, this will become the most relevant and important measure of performance.

Resolution:
Pixel scalability provides the capability of moving beyond 1M - 9M pixel desktop displays to 10s or 100s of Mpixel display walls proving high resolution across almost any size display surface. Scalability also doesn't have a limit to a single dimension. A single display surface might need polygon and fill

scalability with multiple GPUs to achieve the desired framerate. At the same time, the number of pixels can be scaled to make a 4x4 display wall, where each of the 16 display surfaces on the wall have multiple GPUs driving it.

Data size:
By adding GPUs, GPU local memory is scaled. This can allow larger 3D textures to be cached in a distributed fashion, and re-assembled after rendering (monster mode). Likewise, it allows more polygons to be cached in the GPU where performance is highest. Adding GPUs also adds bandwidth, which enables the GPU cached data to be updated more rapidly overall. This paradigm of viewing GPU memory as a local cache is going to become increasingly important and widely used, particularly as vertex and pixel shaders allow more complex and higher level algorithms to be applied to the data.

Scalability Goes Beyond the GPU

Scalability reaches beyond rendering to system accessibility and flexibility. SGI provides hardware compositing solutions to combine multiple GPU output streams together into a single output stream that can be sent to a display device. SGI also provides many system level tools to help configure and manage systems with large numbers of GPUs and CPUs. SGI has been working with scalable graphics longer than any other company and has built much of its knowledge into toolkits and documentation. SGI provides the knowledge needed to write high performance OpenGL applications that explicitly take advantage of large CPU and GPU count systems. Conversely, software is also available that will allow any application to scale seamlessly across the GPUs available in a system without need to write any explicit parallel CPU or GPU code. SGI also provides toolkits that fall between the two extremes, which allow an application to choose the level of scalability assistance needed.

As SGI developed expertise in building scalable compute and visualization architectures, it was clear that tools to enable developers to build applications easily leveraging the power of scalability were critical. Over the past 10 years, SGI has been developing and improving on these toolkits to the point that there is a full range of toolkits that enable scalable applications. The choice of which toolkit to use depends on many factors including:

- the application architecture
- the target platform
- level of scalability desired
- the amount of control you are willing to give to the toolkit

At the simplest level, SGI offers its OpenGL Multipipe™ software, which is designed to run a traditional single pipe application across multiple pipes' outputs. This level of API gives you moderate scalability with very little impact to the design and architecture of the existing application. Any single pipe application will see moderate scalability when run with OpenGL Multipipe. To increase scalability, SGI has some simple guidelines to follow that allow OpenGL Multipipe to further increase application scalability. These guidelines entail some OpenGL coding idioms that generally do not affect single pipe performance, but allow OpenGL Multipipe to run more efficiently. For example, using smaller localized display lists allows OpenGL Multipipe to cull out many of the display lists on a pipe that will not be visible on that pipe. See the OpenGL Multipipe Web site (www.sgi.com/software/multipipe) for more details on additional scalability optimizations.

A step beyond OpenGL Multipipe is to have your application directly address all the graphics pipes in the system rather than leaving this up to OpenGL Multipipe. This will provide a boost in performance and scalability by sending multiple streams of OpenGL to the graphics pipes rather than a single OpenGL stream that is characteristic of a single pipe application. The OpenGL Multipipe SDK toolkit provides a framework for you to manage multiple draw threads rendering to multiple pipes or GPUs. The OpenGL Multipipe SDK framework determines when and which GPU to render to while handling all of the details necessary to manage graphics contexts and matrix stacks. The application is still responsible for all rendering, but many the complexities of multipipe rendering are offloaded to OpenGL Multipipe SDK. See the OpenGL Multipipe SDK Web site (www.sgi.com/software/multipipe/sdk) for more details on scaling applications with OpenGL Multipipe SDK.

While OpenGL Multipipe SDK can allow an application to linearly scale over many GPUs, there are still many details that need to be taken into account to achieve the highest level of scalability. OpenGL Performer™, which is a scene graph-based API, is designed to enable this ultimate level of scalability and performance. OpenGL Performer offloads all rendering responsibility from the application. An application describes the scene to be visualized to OpenGL Performer, and OpenGL Performer decides the most efficient way to render that scene across all the available CPUs and GPUs in the system. See the OpenGL Performer Web site (www.sgi.com/software/performer) for more details on scaling applications with OpenGL Performer.

There are many approaches available to scale visual computing, each having different benefits for different problem domains. The most common ways to combine GPUs are :

• 2D tiling	screen space
• 3D tiling (also called depth compositing)	world space
• 4D tiling (also called DPLEX)	time based
• Eye based decomposition	used for stereo
• Data tiling	database
• Task division	

It should also be noted that each of these can be combined; for example, 4 sets of 8 GPUs could be grouped in a 2D tile of 4 quadrants with each quadrant using data sub-division, and two sets of 32 GPUs could be used with 32 for the left eye and 32 for the right eye!!

2D tiling:

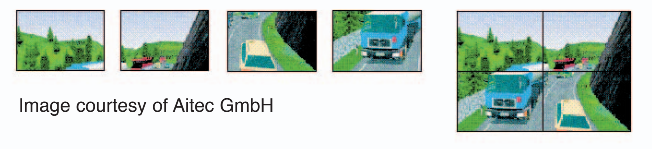
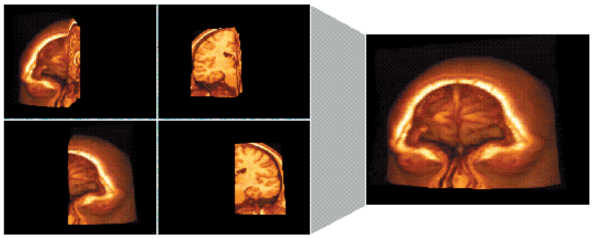


Image courtesy of Aitec GmbH

Imaging applications that primarily work in 2D can easily distribute the data to the available CPUs in a "tiled" fashion, with each GPU focusing on a part of the screen. This is a division of labor by screen space. As more and more GPUs are added, the tiles get smaller and smaller, and the rendering performance goes faster and faster.

This same approach works well in many 3D rendering situations as well, particularly if the problem is "fill limited," or if the polygon data can be easily culled on each pipe.

3D tiling:



Visible Human public data set

A similar approach can apply to 3D division of labor by world/data space division. This requires more culling, but works well where there is a lot of depth complexity, and particularly well where 3D textures are used.

4D tiling:

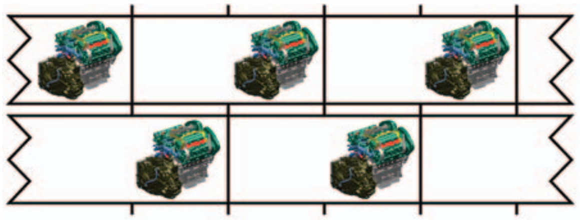
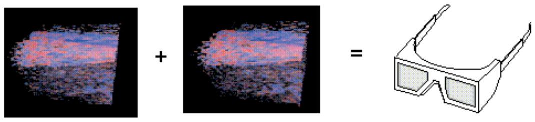


Image courtesy of PSA

Here, each GPU produces a different frame/image all by itself. This is more complex to program, since application state has to be handled carefully to ensure that each GPU can render its view of the scene at the same time.

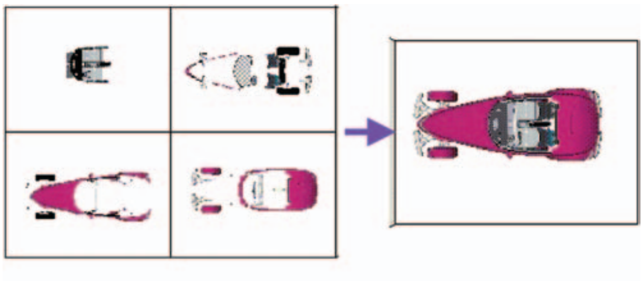
The advantage is that scaling of performance is linear. 20 GPUs make for 20x faster performance. Unfortunately, latency doesn't improve as GPUs are added, so this technique is best used where latency is less important (e.g., viewing a set animation with simple interaction).

Eye based decomposition:



Here, one pipe (or group of pipes) renders for one eye, and one pipe (or group of pipes) renders for the other eye. These two images are then either sent to separate projectors for passive stereo, or are combined into an active stereo signal by the Silicon Graphics® Compositor technology.

Data tiling:



Data courtesy of DaimlerChrysler

Here, each GPU takes a part of the data set to render. For example, one could take the wheels of a car, another all glass parts, another all metallic parts, and a final one all remaining parts. This can allow very good load balancing and scaling, and allow very specialized shaders to be efficiently applied. Care must be taken in the re-assembly of the GPU outputs though. In particular, if full-scene anti-aliasing is used then sub-samples will often be combined into pixel results without being blended against all the samples that would have been in that pixel had the entire scene been rendered on a single pipe. For this reason, rendering at high resolution and down-sampling later is the recommended approach.

Task division:
Here, specific GPUs are assigned key tasks. For example, one GPU could be preparing a cube-map for environment mapping while another is producing image-based rendering proxies for objects too complex to be rendered interactively. These results would then become pages into the GPU/GPUs being used to render the final scene at interactive rates. GPUs can be assigned specific rendering tasks like culling, texture generation, compositing, or other operations that lead to the final visualization. These tasks can be load balanced over time to provide optimal rendering performance. During intensive geometry operations, more GPUs can be dedicated to culling and geometry rendering, whereas in lighter loaded visual scenes, some GPUs can be dedicated to rendering higher quality textures.

The ability to scale the resources in a computer system is one of the most powerful techniques available to meet the ever-increasing demands in data size and problem complexity. SGI pioneered scaling for high-performance computing and continues to innovate and lead the way for the industry with breakthrough technology, including the largest SSI system ever built and the largest Linux node ever built. With the introduction of the Onyx4, SGI brings the same balanced scaling capabilities into the world of visualization. SGI's scalable, shared memory graphics architecture provides capabilities such that GPUs can be scaled seamlessly and efficiently, like CPUs. That ability to independently scale resources, like CPUs, GPUs, and memory, to meet the demands of a given problem, takes visualization to the next level, where data becomes knowledge and solutions.

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

North America +1 800.800.7441
Latin America +55 11.5509.1455
Europe +44 118.925.7500
Japan +81 3.5488.1811
Asia Pacific +1 650.933.3000