

Cluster Interconnects: The Whole Shebang

It is what makes a cluster "a cluster"

An often asked question from both "clusters newbies" and experienced cluster people ask is, "what kind of interconnects are available?" The question is important for two reasons. First, the price of interconnects can range from as little as \$32 per node to as much as \$3,500 per node, yet the choice of an interconnect can have a huge impact on the performance of the codes and the scalability of the codes. And second, many users are not aware of all the possibilities. People new to clusters may not know of the interconnection options and, sometimes, experienced people choose an interconnect and become fixated on it, ignoring all of the alternatives. The interconnect is an important choice and ultimately the choice depends upon on your code, requirements, and budget.

In this review we provide an overview of the available technologies and conclude with two tables summarizing key features and pricing for various size clusters.

Introduction

This article will focus on interconnects that aren't tied to vendor specific node hardware, but can work in a variety of cluster nodes. While determining *which* interconnect to use is beyond the scope of this article, I can present what is available and make some basic comparisons. I'll present information that I have obtained from the vendors websites, from information people have posted to the [Beowulf mailing list](#), the vendors, and various other places. I won't make any judgments or conclusions about the various options because, simply, I can't. The choice of an interconnect depends on your situation and there is no universal solution. I also intend to stay "vendor neutral" but will make observations where appropriate. Finally, I have created a table that presents various performance aspects of the interconnects. There is also a table with list prices for 8 nodes, 24 nodes, and 128 nodes to give you an idea of costs.

I'm going to examine seven interconnects. These interconnects are:

- Gigabit Ethernet
- Gigabit Ethernet with **Level 5** NICs
- 10 Gigabit Ethernet
- **Infiniband**
- **Infinipath**
- **Myrinet**
- **QsNet** (Quadrics)
- **SCI** (Dolphin)

In addition, I'll talk about some of the technologies that some of the interconnects use to improve performance. Namely, TCP bypass (OS bypass), TCP Off-load Engine (TOE), RDMA (Remote Direct Memory Access), zero-copy networking, and interrupt mitigation. I'll also discuss some alternatives to plain Gigabit Ethernet such as

- **GAMMA**
- **Scali MPI**
- **ParTec**

Gigabit Ethernet

Ethernet has an enduring history. It was developed principally by Bob Metcalfe and David Boggs at Xerox Palo Alto Research Center in 1973 as a new bus topology LAN (Local Area Network). It was designed to be a set of protocols at the datalink and physical layers. In 1976, carrier sensing was added. A 10 Mbps (Mega-bits per second) standard was promoted by DEC, IBM, and Xerox in the 1980's. Novell adopted it and used in their widely successful Netware. It finally became an IEEE standard in 1983 (802.3, 10Base5).

The initial speed of Ethernet was only 10 Mbps. As more systems were added to networks, a faster network connection was needed. Development of Fast Ethernet (802.3u), which runs at 100 Mbps or 10 times faster than 10 Mbps, was started in 1992 and was adopted as an IEEE standard in 1995. It is really just the same 10 Mbps standard but run at a higher speed.

With the explosion of the Internet and with the need to move larger and larger amounts of data, even Fast Ethernet was not going to be fast enough. The development of Gigabit Ethernet (GigE), which runs as 1 Gbps (Giga-bits per second) or 10 times faster than Fast Ethernet, was started and became an IEEE standard (802.3z) that was adopted in 1998. GigE uses the same protocol as 10 Mbps Ethernet and Fast Ethernet. This makes it easy to upgrade networks without having to upgrade every piece of software. Plus GigE can use the same category 5 copper cables that are used in Fast Ethernet (If you are buying new cables for GigE choose cat 5e or better).

Currently, there are a large number of manufacturers of GigE Network Interface Cards (NIC) on the market. Most of the GigE NICs are supported at some level in the Linux Kernel. Many of NIC manufacturers use the same basic GigE chipsets (e.g. Intel, Broadcom, Realtek, National Semiconductor), sometimes slightly modifying how they are implemented. This makes it difficult to write "open" drivers without vendor support. There are some good articles available on the web that describe the lack of vendor support for GigE NICs, especially Nvidia (although Nvidia has some **closed drivers** that allow for the use of the Nvidia audio and network devices). In recent years, most major vendors have released source code drivers for their GigE chipsets or at least provided documentation on the chipsets. The Linux kernel has driver support for most common types of GigE chipsets. There are also Fast Ethernet drivers, original 10 Mbps Ethernet drivers, and fairly recently, 10 Gigabit Ethernet (10 GigE) drivers.

Many of these Ethernet drivers have parameters that can be adjusted. Later in this article I will mention some of the things that can be adjusted. You can adjust the various parameters for any number of reasons and you find that cluster users often experiment with drivers to determine which is the best combination for their cluster and work load.

One of the problems with GigE is that it is based on the standard Ethernet **MTU** (Maximum Transmission Unit) size of 1500 Bytes. As each Ethernet packet must be serviced by the processor (as an interrupt) moving data at Gigabit speeds can tax even the fastest systems (80,000 interrupts a second). One of the ways to solve this problem is the use of "Jumbo Frames" or basically a larger MTU. Most GigE NICs support

MTU sizes up to 9000 Bytes (reducing the number of interrupts by up to a factor of 6). If you want to use Jumbo Frames, make sure your switch supports these sizes (see the subsection on GigE switches entitled "Switching the Ether").

GigE NICs come in various sizes and flavors over a range of prices. Sixty-four-bit PCI "server" GigE NICs are still slightly expensive ("desktop" NICs, which are essentially built from the same components, are very inexpensive, but are limited by the 32-bit PCI bus). For server class motherboards as well as normal desktop motherboards, you will find GigE built into the motherboard at a lower cost. If your motherboard doesn't have built-in GigE, then regular 32-bit PCI NICs start at about \$3-\$4 with decent ones, such as the DLINK DFE_530TX starting at about \$13 or the popular Intel PCI GigE NIC starting at about \$30. Sixty-four-bit GigE NICs start at about \$33 (when this article was written), but PCI-X NICs are more expensive. PCI-X GigE NICs with copper connectors start at about \$100 and PCI-X GigE NICs with Fiber Optic connectors start at about \$260. (*Note: These prices were determined at the time the article was posted. They will vary.*) Also, be aware that the really low cost NICs may not perform as expected. Some basic testing will help qualify NICs.

GigE is a switched network. A typical GigE network consists of a number of nodes connected to a switch or set of switches. You can also connect the switches to each other to create a tiered network. Other network topologies such as rings, 2D and 3D torus, meshes, etc. can also be constructed without switches. However, some of these networks may require multiple NICs per node and can be complicated to configure, but they do eliminate the need for switches.

While it is beyond the scope of this article, you can also use new network topologies such as **Flat Neighborhood Networks** (FNNs) or **Sparse Flat Neighborhood Networks** (SFNNS) that offer potential benefits for particular applications. There are on-line tools to help you design FNNs. As shown in some of the literature, these types of networks can offer very good price/performance ratios for certain codes.

Pushing GigE

Despite the rise in Ethernet speeds, it is possible for a single 32-bit PCI based NIC to saturate the entire PCI bus. Consequently, it's easy to see that the CPU can spend a big part of it's time handling interrupt requests and processing network traffic.

The situation has led vendors to develop TOE (TCP Off-Load Engines) and RDMA (Remote Direct Memory Access) NICs for GigE. The goal of these cards is to reduce latency and to reduce the load on the CPU when handling network data. Several companies started marketing these type of NICs, but really only one remains, **Level 5 Networks**.

Level 5 Networks makes a 32/64 bit PCI GigE NIC that is not a RDMA NIC but uses a new technology called **Etherfabric**. In a conventional network, a user application converts the sockets or MPI calls into system calls that use the kernel TCP/IP stack. With Etherfabric, each application or thread gets its own user space TCP/IP stack. The processed data is sent directly to the virtual hardware interface that was assigned to the stack and then sent out over the NIC. This process eliminates context switching (mentioned below) to improve latency.

The interesting thing about the Level 5 NIC and Etherfabric, is that you can use existing TCP/IP networks and any routers because it doesn't use a different packet protocol. In addition, the NIC appears as a conventional GigE NIC so it can send and receive conventional traffic as well as Etherfabric traffic. It also means that you don't have to use an Etherfabric specific MPI implementation. Consequently, you don't have to recompile and can use your binaries immediately. This overcomes an Achilles heal of past high-speed GigE NICS because you can use all of your ISV applications.

GAMMA or Genoa Active Message Machine is a project maintained by Giuseppe Ciaccio of Dipartimento di Informatica e Scienze dell'Informazione in Italy. It is a project to develop an Ethernet kernel bypass capability for existing Ethernet NICs as well as its own packet protocol (doesn't use TCP packets). Since it bypasses the Linux kernel and doesn't use TCP packets it has a much lower latency than standard Ethernet.

Right now, GAMMA only supports the Intel GigE NIC and the Linux 2.6.12 kernel (Broadcom chipsets (tg3) are said to be under development). Also, since it uses something other than TCP the packets are not routable over a WAN (not likely to be a problem for clusters, but could be a problem for Grids). GAMMA can share TCP traffic over the interfaces, but it takes exclusive control of the NIC when transmitting or sending data. That is, all TCP/IP traffic is stopped. Normally this is not a problem as most clusters have multiple networks available to them.

There is also a port of MPICH1 to run using GAMMA. **MPI/GAMMA** is based on MPICH 1.1.2 and supports SPMD/MIMD parallel processing and is compatible with standard network protocols and services.

Figure One is a plot of the throughput of MPI/GAMMA compared to LAM/MPI and MPICH1. *Figure Two* is a comparison of the latencies for the same three MPI's.

Netpipe MPI - Throughput vs. Blocksize
Gigabit Ethernet

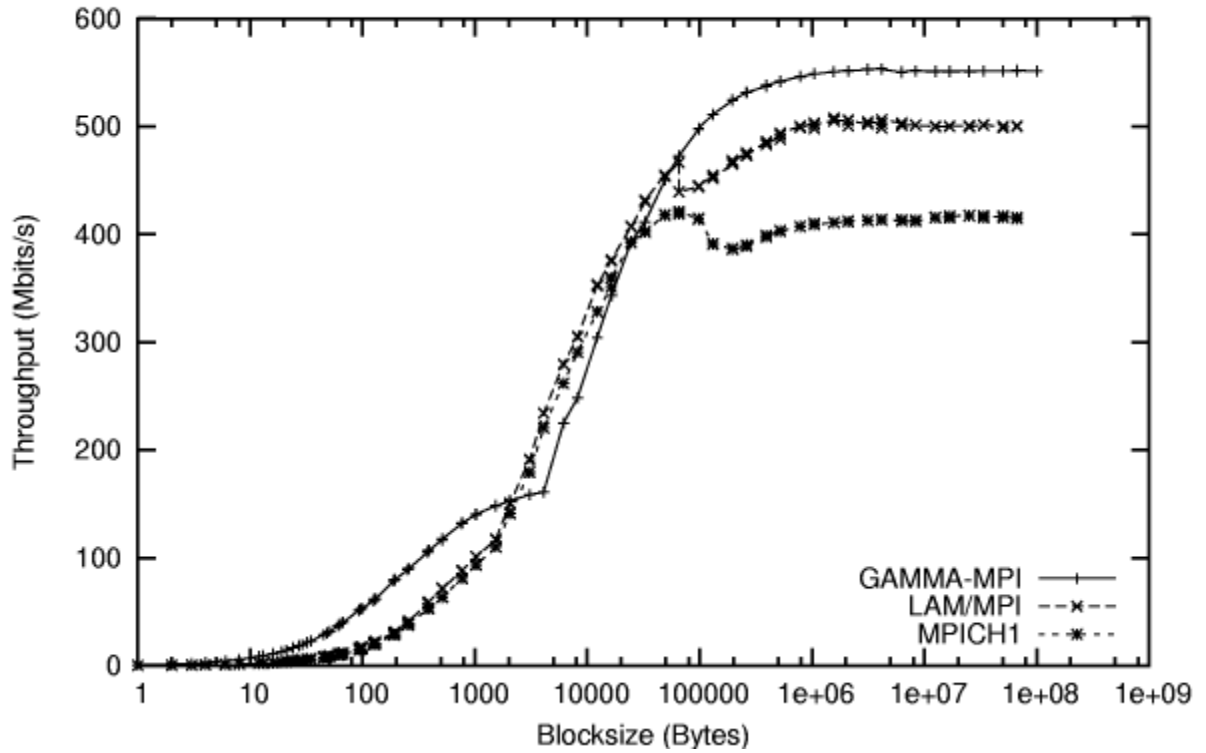


Figure One: Throughput Comparison for GAMMA, LAM, and MPICH

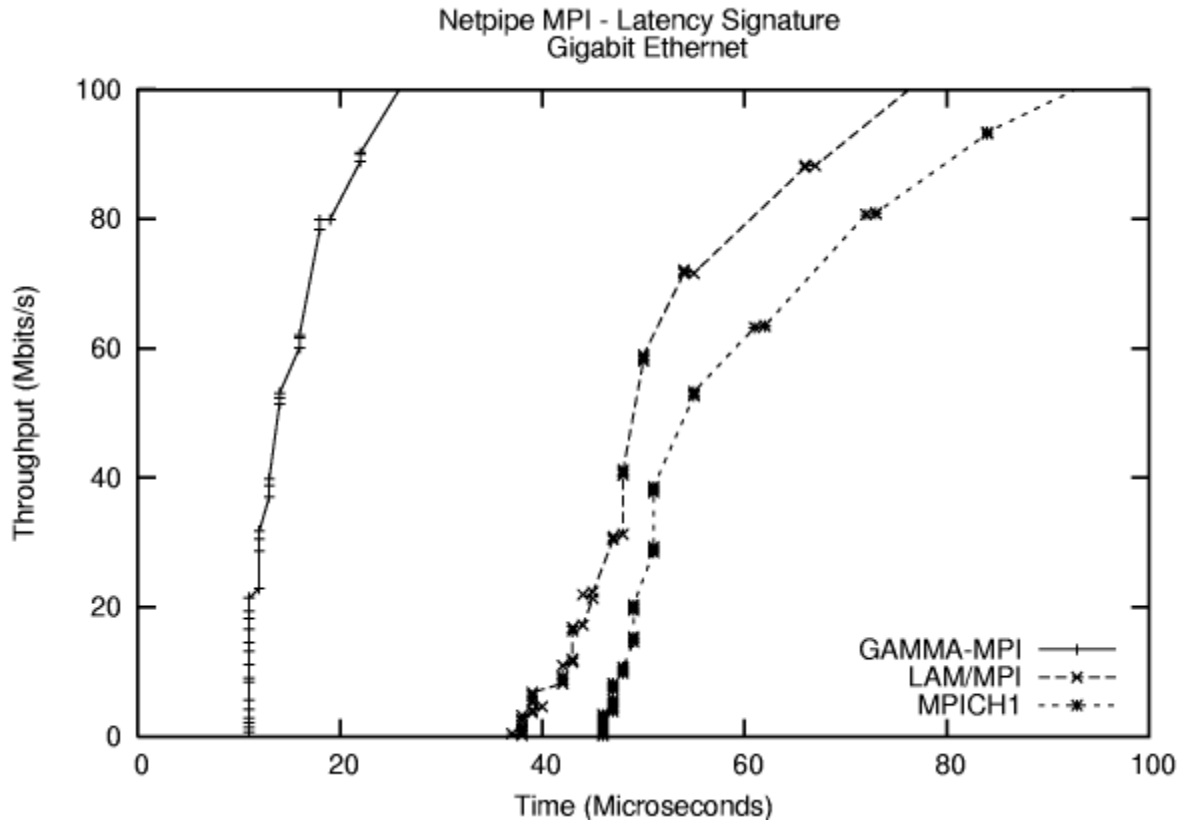


Figure Two: Latency Comparison for GAMMA, LAM, and MPICH

Notice that the latency is only **9.54** microseconds and this includes the latency of the switch! This is probably the lowest latency I've ever seen for GigE. Also, the throughput for MPI/GAMMA is very good as well, peaking at about 72 MByte/s or 560 Mbits/s. Not shabby for inexpensive Intel 32 bit PCI GigE card.

In addition to GAMMA, there are other network solutions that don't use TCP. For example **Scali MPI Connect** has a module called "Direct Ethernet" or DET that runs over any communication interface that has a MAC address. In other words, it uses a different protocol than TCP and by-passes the kernel (sometimes called TCP-bypass). Moreover, Scali says that using DET allows you to aggregate multiple GigE networks together. Scali says that you can use independent inexpensive unmanaged layer 2 switches for this aggregation.

Another example of an MPI implementation that doesn't use TCP as the packet protocol is **Parastation** from Partec. It currently is up to Version 4 and claims an MPI latency of about 11 microseconds. It is based on MPICH 1.2.5.

I Feel the Need for Speed

I have been using various terms such as TCP-bypass, without adequately explaining them. I want to take some time to explain some of the techniques that have been used to improve the performance of interconnects. In particular, I'm going to cover:

- TCP-Bypass
- TCP Offload Engine
- Kernel Bypass (OS Bypass)
- RDMA
- Zero-copy Networking
- Interrupt Mitigation

TCP Bypass

I previously mentioned TCP-Bypass when I discussed Scali MPI Connect. TCP-bypass really means that you are using a different packet protocol than **TCP** or **UDP**. The application uses its own packet definition, builds the packets on the sending side and sends them directly over the **link layer** where they are decoded and the data processed. There are a number of products that do this, such as, **Scali MPI Connect**, Parastation, **HyperSCSI** (iSCSI like storage), and **Coraid ATA-over-Ethernet**. The term "TCP-bypass" is used because you are "bypassing" the TCP packet protocol and using a different protocol right on top of the link layer.

The idea behind using a different protocol is that you reduce the overhead of having to process **TCP** packets. Theoretically this allows you to process data packets faster (less overhead) and fit more data into a given packet size (better data bandwidth). So you can increase the bandwidth and reduce the latency. However, there are some gotchas with TCP-bypass. The code has to perform all of checks that the TCP protocol performs, particularly the retransmission of dropped packets. In addition, because the packets are not TCP, they are not routable across Ethernet. So you can't route these packets to other networks. This is not likely to be a problem for clusters that reside on a single network, but Grid applications will very likely have this problem because they need to route packets to remote systems.

TCP Offload Engine

TCP Offload Engines (TOE) are a somewhat controversial concept in the Linux world. Normally, the CPU processes the TCP packets, which can require extreme processing power. For example, it has been reported that a single GigE connection to a node can saturate a single 2.4 GHz Pentium IV processor. The problem of processing TCP packets is worse for small packets. To process the packets, the CPU has to be interrupted from what it's doing to process the packet. Consequently, if there are a number of small packets, the CPU can end up processing just the packets and not doing any computing.

The **TOE** was developed to remove the TCP packet processing from the CPU and put it on a dedicated processor. In most cases, the TOE is put on a NIC. It handles the TCP packet processing and then passes the data to the kernel, most likely over the PCI bus. For small data packets, the PCI bus is not very efficient. Consequently, the TOE can collect from a series of small packets and then send a larger combined packet across the PCI bus. This design increases latency, but may reduce the impact on the node processing. This feature is more likely to be appropriate for enterprise computing than HPC.

Kernel Bypass

Kernel Bypass, also called OS bypass, is a concept to improve the network performance, by going "around" the kernel or OS. Hence the term, "bypass." In a typical system, the kernel decodes the network packet, most likely TCP, and passes the data from the kernel space to user space by copying it. This process means the user space process context data must be saved and the kernel context data must be loaded. This step of saving the user process information and then loading the kernel process information is known as a context switch. According to this [article](#), application context switching constitutes about 40% of the network overhead. So, it would seem that to improve bandwidth and latency of an interconnect, it would be good to eliminate the context switching.

In Kernel bypass, the user space applications communicate with the I/O library that has been modified to communicate directly with the user space application. This process takes the kernel out of the path of communication between the user space process and the I/O subsystem that handles the network communication. This change eliminates the context switching and potentially the copy from the kernel space to user space (it depends upon how the I/O library is designed). However, people are arguing that the overhead in the kernel associated with a context switch has shrunk. Combined with faster processors the impact of a context switch has lessened.

RDMA

Remote Direct Memory Access (**RDMA**) is a concept that allows NICs to place data directly into the memory of another system. The NICs have to be RDMA enabled on both the send and receive ends of the communication. RDMA is useful for clusters because it allows the CPU to continue to compute while the **RDMA** enabled NICs are passing data. This can help improve compute/communication overlap, which helps improve code scalability.

The process begins with the sending RDMA NIC establishes a connection with the receiving RDMA NIC. Then the data is transferred from the sending NIC to the receiving NIC. The receiving NIC then copies the data directly to the application memory bypassing the data buffers in the OS. RDMA is most commonly used in Infiniband implementations, but other high-speed interconnects use it as well. Recently 10 GigE NICs started using RDMA for TCP traffic to improve the performance. There has been some discussions lately that RDMA may have outlived its usefulness for MPI codes. The argument is that most messages in HPC codes are small to medium in size and that using memory copies to move the data from kernel space to user space is faster than having a RDMA NIC to it. Reducing the amount of time the kernel takes to do the copy and improving processors speeds are two of the reasons that a memory copy could be faster than RDMA.

There is a **RDMA Consortium** that helps organize and promote RDMA efforts. They develop specifications and standards for RDMA implementations so the various NICs can communicate with each other. Their recent efforts have resulted in the development of an RDMA set of specifications for TCP/IP over Ethernet.

Zero-Copy Networking

Zero-Copy networking is a technique where the CPU does not perform the data copy from the kernel space to user space (the application memory). This trick can be done for both send operations and receive operations. This can be accomplished in a number of ways including using **DMA** (Direct Memory Access) copying or memory mapping using a **MMU** (Memory Management Unit) equipped system. Zero-copy

networking has been in the Linux kernel for some time, since the 2.4 series. Here is an [article](#) that discusses how the developers went about accomplishing it. The article gives some details at a high level about how one accomplishes this. It also points out that zero-copy networking requires extra memory and a fast system to perform the operations. If you would like more information, this [article](#) can give you even more detail.

With every new idea there are always seems to differing opinions. [Here](#) is an argument that zero-copy may not be worth the trouble. Rather the authors argue that a Network Processor (kind of a programmable, intelligent NIC) would be a better idea.

Interrupt Mitigation

Interrupt Mitigation also called Interrupt Coalescence, is a another trick to reduce the load on the CPU resulting from interrupts to process packets. As mentioned earlier, every time a packet gets sent to the NIC, the kernel must be interrupted to at least look at the packet header to determine of the data is destined for that NIC, and if it is, process the data. Consequently, it is very simple to create a Denial-of-Service (DOS) attack by flooding the network with a huge number of packets forcing the CPU to process everyone of them. Interrupt Mitigation is a driver level implementation that collects packets for a certain amount of time or a certain total size, and then interrupts the CPU for processing. The idea is that this reduces the overall load on the CPU and allows it to at least do some computational work rather than just decode network packets. However, this can increase latency by holding data before allowing the CPU to process it.

Interrupt Coalescence has been implemented in Linux through NAPI (New API) rewrites of the network drivers. The rewrites include interrupt limiting capabilities on both the receive side (Rx) as well as the transmit side (Tx). Fortunately, the people who wrote the drivers allow the various parameters to be adjusted. For example, in this [article](#), Doug Eadline (Head Monkey) experimented with various interrupt throttling options (interrupt mitigation). Using the stock settings with the driver the latency was 64 microseconds. After turning off interrupt throttling, the latency was reduced to 29 microseconds. Of course, we assume the CPU load was higher, but we didn't measure that.

There are two good articles that discuss "tweaking" GigE, TCP NIC drivers. The [first](#) describes some parameters and what they do for the drivers. The [second](#) describes the same thing but with more of a cluster focus. Both are useful for helping you understand what to tweak and why, and what the impacts are.

Switching the Ether

GigE switches have now become less expensive. Switches can be purchased in two varieties: managed and unmanaged. Small switches are almost always unmanaged switches. That is, they are plug-and-play with no option to provide any type of configuration. A small 5-port switch starts at about \$32 with an 8-port switch costing starting at about \$52 and a 16-port switch starting at \$233. These switches typically do not support jumbo frames (MTU up to 9000).

SMC has been selling a very inexpensive line of unmanaged small switches for several years. While these switches are very good in their own right, one of the best features of these switches is that they are capable of jumbo frames. (*Note: The prices mentioned below were determined at the time the article was posted.*)

The **SMC 8505T** 5-port switch costs about \$70, the **SMC 8508T** 8-port switch costs about \$100, and a 16-port version (8516T) starts at about \$300. There is also a 24-port version of this new switch called the SMC 8524T that can be found for about \$400. The 8516T and 8524T switches are becoming more difficult to find but there is a new line of 16 and 24 port unmanaged switches that are capable of jumbo frames. The **SMC GS16-Smart** 16-port switch costs about \$240. A 24-port version of this switch, the **SMC GS24-Smart** can be found for about \$300.

There are a number of switch manufacturers that target the cluster market by providing large high density/performance managed switches. Managed switches allow users to monitor performance or change settings. Companies such as **Foundry**, **Force10**, **Extreme Networks**, **HP**, **SMC**, and **Cisco** offer large GigE switching solutions.

For clusters with Ethernet, having a single high performance backplane (i.e. one large switch) is important for good performance. A single backplane provides good "cross-sectional" bandwidth where all nodes can achieve maximum throughput. An alternative to a single backplane is to cascade smaller and usually less expensive switches to provide the requisite number of ports for your cluster. This is usually done in a common topology such as a fat-tree topology. Depending upon the topology, cascading switches can reduce the effective cross-sectional bandwidth and almost always increases the latency. Several companies have examples of large single backplane switches. Foundry has a switch (**BigIron RX-16**) that can have up to 768 GigE ports in a single switch. Extreme Networks has a 480-port GigE switch (**Black Diamond 10808**) and Force10 has a monster 1260-port GigE switch (**Terascale E-series**).

There are some interesting topologies you can use with Ethernet to get higher performance and lower cost networks. The two most prominent ones are **FNN** (Flat Neighborhood Networks), and **SFNN** (Spare Flat Neighborhood Networks).

A FNN is a network topology that guarantees single-switch latency and full bandwidth between per processing element for a variety of communication patterns. The way this is achieved is that each node has more than one NIC that are used to plug into smaller switches in such a way that there is only a single link between certain pairs of nodes. If you know how your code communicates you can then use this knowledge to design a low-cost, high performance FNN that uses inexpensive small switches to achieve the same performance that you get from larger much more expensive switches. The website even has an on-line tool for designing FNNs.

Sparse FNN's are a variation of FNNs but allow you to select which communication patterns you want to ensure have single switch latency and full bandwidth. They allow you to design a network for very specific codes to get the best performance at the lowest cost.

GigE Message-Passing

Communication over GigE is usually done by using the kernel TCP services. The result of this standard means a large number of MPI implementations are available for GigE. Virtually every MPI implementation available, either open-source or commercial, supports TCP. There are a number of open-source implementations such as **MPICH**, **MPICH2**, **LAM-MPI**, **Open-MPI**, **FT-MPI**, **LA-MPI**, **PACX-MPI**, **MVICH**, **GAMMA**, **OOMPI** (C++ only), **MP-MPICH**, and **MP_Lite** (useful subset of MPI).

There are also some major commercial MPI vendors. Verari Systems Software (previously MPI Software Technologies) has **MPI/Pro** and Scali has **Scali MPI Connect**, Critical Software has **WMPI**, HP has **HP-MPI**, and Intel has **Intel-MPI**.

Also, since this is just plain TCP, we can also use PVM and other message passing schemes may be used over GigE. In addition, many storage protocols support TCP and in some cases native Ethernet. For example, **iSCSI**, **HyperSCSI**, **Lustre**, **ATA over Ethernet**, **PVFS**, **GPFS**, can be run over GigE.

Beyond GigE: 10 Gigabit Ethernet

It was quickly realized that even GigE would not give enough throughput for our data-hungry world. Consequently, the development of the next level, 10 Giga-bits per second, or **10 GigE**, was started. One of the key tenants of the development was to retain backwards compatibility with previous Ethernet standards. This new IEEE standard is 802.3ea.

It was also realized that there would have to be some changes to the existing way Ethernet functioned to get the required performance. For example, the IEEE standard has altered the way the MAC layer interprets signaling. Now the signals are interpreted in parallel to speed up the processing. However, nothing has been done to the standard to limit backward compatibility.

For most 10 GigE installation, fiber optic cables are used to maintain a 10 Gbps speed. However, copper is becoming increasingly popular. There are NICs and switches that support 10GBASE-CX4, which are copper cables that use Infiniband 4x connectors and CX4 cabling. These cables are currently limited to 15m in length.

However, there are new 10 GigE cables being developed. In approximately August 2006, there should be 10GBASE-T cables available. They use unshielded twisted-pair wiring, the same as cat-5e or cat-6 cables. This is the proverbial "Holy Grail" of 10 GigE. Having just the cables will be somewhat pointless though if the NICs and the switches are not available as well. So we should see a flurry of new product announcements during 2006.

10 GigE NIC Vendors

There are several vendors currently developing and selling 10 GigE NICs for clusters. Currently there are **Chelsio** and **Neterion**, formally S2IO, and **Intel** providing 10 GigE NICs.

Chelsio markets two PCI-X intelligent 10 GigE NICs that include RDMA support that are reasonable to use for the HPC market. The **T210** uses fiber optic cables and comes in both an SR and LR version (single fiber mode and multi fiber mode) for 64-bit PCI-X. This NIC includes both a TOE capability (TCP Off-Load Engine) and a RDMA (Remote Direct Memory Access) capability to improve the performance of the NIC as much as possible. Since the NIC uses TCP as the network protocol, any MPI implementation that uses TCP, which virtually all of them do, will work with these NICs without change. More over, the Chelsio driver was the first 10 GigE driver to be included in the Linux kernel.

Chelsio also has a copper connector version, the **T210-CX**. It is a 64-bit PCI-X NIC that uses CX4 copper cables. It has the same features of the T210 including TOE and RDMA capability.

Chelsio also sell a "dumb" 10 GigE NIC, **N210**, that that does not include RDMA nor TOE capabilities. It uses fiber optic cables, both SR and LR, for connecting to switches or to other NICs. It is cheaper than the T210 series, but likely have the same level of performance for HPC codes.

Neterion has a 10 GigE NIC, called the **Xframe**. The NIC is a 64-bit PCI-X NIC and has RDMA and some TOE capability and uses fiber optical cabling. Neterion has also announced a new 10 GigE NIC called **Xframe II**. This NIC has a 64-bit PCI-X 2.0 interface that should allow a bus speed of 266 MHz instead of the usual 133 MHz. According to the company this NIC should be capable of hitting 10 Gbps wire-speed (PCI-X currently limits 10 GigE NICs to about 6.5-7.5 Gbps) and achieve a 14 Gbps bi-directional bandwidth.

Currently both the Xframe and Xframe II NICs use optical fiber connectors, presumably both SR and LR. However, with the desire for copper connectors, it is entirely possible they have a CX4 version of the NIC. Neither NIC is directly sold to the public but is sold to OEM's. Recently IBM has announced that they will use Neterion NICs in their xSeries servers that use Intel processors.

Intel has developed and is selling three 10 GigE NICs: the Intel Pro/10GbE CX4 NIC, Intel Pro/10GbE SR NIC, and the Intel Pro/10 GbE LR adapter. The CX4 version is a PCI-X NIC that uses copper cabling. When this article was written, the best pricing for it was about \$872. The SR version is also a PCI-X NIC that uses multi-mode fiber cables for connectivity. It is intended primarily for connecting enterprise systems and not for HPC. Currently it costs about \$3,000 per NIC. Finally, the LR version of the NIC, which is also a PCI-X NIC, is for long-range connectivity (up to 10 km) using single-mode fiber cables. As with the LR NIC, it is not really designed for HPC and its price is about \$5,000.

It is likely that other vendors will be developing 10 GigE products in the near future. Level 5 and other RDMA GigE manufacturers are rumored to be developing a 10 GigE product.

10 GigE MPI

Since 10 GigE is still Ethernet and TCP, you can use just about any MPI implementation, commercial or open-source, as long as it supports TCP or Ethernet. This means that you can run existing binaries without any source changes. This reason is a why people are seriously considering 10 GigE as the upcoming interconnect for HPC.

10 GigE Switches

There are several 10 GigE switch manufacturers. The typical HPC switch vendors such as Foundry, Force 10, and Extreme all make 10 GigE line cards for their existing switch chassis. They have been developing these line cards primarily for the enterprise market, but the now realize that as the costs come down on the line cards and the NICs, that they may have a product line suitable for the HPC market.

Foundry has a new large chassis (14U) switch, called the **RX-16**. It can accommodate up to 16 line cards. Foundry currently has a 4-port 10 GigE line card. They have a fiber optic version of this line card and,

presumably, a copper version using CX4 cables. All ports on the switch run at full line rate and have an approximate per port cost of \$4,000.

A company that has focused on 10 GigE for some time is **Extreme Networks**. They have a **BlackDiamond** series of switches that focus on high performance, including HPC. Their largest switch, the **BlackDiamond 10808**, can accommodate up to 48 ports of 10 GigE, presumably both fiber and copper.

Force10 has probably been the leader in the 10 GigE market. They have a large single chassis switch, called the **E Series** that can accommodate up to 224 10 GigE ports. To reach this port count, they have a new line card that can accommodate up to 16 ports of 10 GigE (a total of 14 line cards). However, these new line cards are not **full line rate cards**. To get full line rate on each port, they have an 4-port 10 GigE line card, resulting in 56 total 10 GigE ports. An interesting difference between the cards, beside the performance, is the price. The 16-port line cards result in a per port cost of about \$2,700, while the 4-port line cards result in a per port cost of about \$7,500. Both line cards have plug-able XFP optics allowing SR, LR, ER, and ZR optics to be used.

The switches from Extreme, Force10, and Foundry focus on the high end of 10 GigE with large port counts. However, other companies are focusing on lower port count 10 GigE switches that deliver good performance. **Fujitsu** has a single-chip 10 GigE switch called the **XG700** that has 12 ports in a compact form factor. Also, the switch can be configured for SR, LR, and CX4, connections. It has a very low latency of 450 ns and has a reasonably low cost of about \$1,200 per port.

SMC has long been a favorite of cluster builders for small to medium clusters. Their switches have very good performance and they have a wide range of unmanaged and managed switches. Recently they brought out an inexpensive 8-port switch, the **SMC 8708L2**. It is an 8-port single backplane switch that use XFP connectors that support SR, LR, and ER XFP. It is a managed switch, but at press time one of these switches was about \$6,300. That comes out to less than \$800/port. This is the price/performance leader for small 10 GigE fiber switches.

At the Supercomputer 05 conference some new 10 GigE switch products were announced. For example, **Quadrics** introduce a new 10 GigE switch that uses the Fujitsu single-chip 10 GigE solution. They showed a **new 10 GigE switch** that fits into an 8U chassis. It has 12 slots for 10 GigE line cards. Each line card has 8-ports for 10 GigE connections using CX4 connectors. The remaining four ports for each line card are used to internally connect the line cards in a fat tree configuration. This means that the network is 2:1 oversubscribed but looks to have very good performance. If all line card slots are populated, then the switch can have 96 ports. It is due to come out in Q1 of 2006. No prices have been announced, but the rumors are that the price should be below \$2,000 a port. Quadrics also stated in their press release that follow-on products will increase the port count to 160 and then to 1,600.

Even more exciting is a new company, **Fulcrum Micro**, that is developing a new **10 GigE switch ASIC**. It has great performance with a latency of about 200 ns and uses cut-through rather than store-and-forward for improved latency and throughput. It can accommodate up to 24 ports and should be available in Jan. 2006 for about \$20/port. Fulcrum has a paper that talks about how to take the 24-port 10 GigE switches that use their ASIC and construct a 288-port fat-tree topology with full bandwidth to each port and a latency of only 400 ns. According to Fulcrum, a number of companies are looking at using their ASICs to build HPC-centric 10 GigE switches.

Infiniband

Infiniband was created as an open standard to support a high-performance I/O architecture that is scalable, reliable and efficient. It was created in 1999 by the merging of two projects: Future I/O supported by Compaq, IBM, and HP, and Next Generation I/O supported by Dell, Intel, and Sun.

The reason for the drive to a new high-performance I/O systems was that the existing PCI bus had become a bottleneck in the computing process. It was hoped that updating PCI to something new would allow the bottleneck to be removed.

Much like other standards, IB is a standard that can be implemented by anyone. This freedom has the potential to allow for greater competition. Today there are four major IB companies: Mellanox, Topspin (acquired by Cisco), Silver Storm (was Infinicon), and Voltaire. However, Mellanox is the main manufacturer of Infiniband ASIC.

The Infiniband specification that was finally ratified, provides for a number of features that improve latency and bandwidth for interconnects. One of these is that IB is a bidirectional serial bus. This reduces cost and can improve latency. The specification also provides for the NICs (usually called a HCA - Host Channel Adapter) to use RDMA. This greatly improves latency. Equally important, the specification provides an upgrade path for faster interconnect speeds.

As with other high-speed interconnects, IB does not use IP packets. Rather, it has its own packet definition. However, some of the IB companies have developed an 'IP-over-IB' software stack, allowing anything written to use IP to run over IB albeit with a performance penalty compared to native IB.

The specification starts IB at a 1X speed which allows for an IB link to carry 2.5 Gbps (giga-bits per second) in both directions. The next speed is called 4X. It specifies that data can travel at 10 Gbps (however PCI-X limits this speed to about 6.25 Gbps). The next level up is 12X which provides for a data transmission rate of 30 Gbps. There are also standards that allow for Double Data Rate (DDR) transmissions which transfer twice the same amount of data per clock cycle, and for Quad Data Rate (QDR) transmissions that transfer 4 times the amount of data per clock cycle. For example, a 4X DDR NIC will transfer 20 Gbps and a 4X QDR NIC will transmit 40 Gbps.

Like many other networks, IB is a switched network. That is, the HCAs, connect to switches that are used to transmit the data to the other HCAs. A single chassis switch can be used or the switches can be connected in some topology. Today there are a wide variety of switches from the major IB companies.

Voltaire is a privately held company focusing on IB for HPC in addition to other areas in need of high-speed networking. They were the first of the IB companies to market a large **288-port IB switch**. The switch uses 14U of rack space and provides full 4X SDR (Single Data Rate) bandwidth to each switch port. Alternatively, this switch can accommodate up to 96 ports of 12X Infiniband. Voltaire also ships a small (1U) IB switch, the **9024** that provides up to 24 ports of 4X Infiniband (DDR capable). They also have a cool product, the **ISR 6000** that allows Infiniband based networks, like those in clusters, to be connected to Fibre Channel or TCP networks.

Silver Storm (previously called Infinicon) was founded in 2000 and privately held, Silver Storm sells an **HCA** in both PCI-X and PCI-Express form factors, IB switches, and all of the support infrastructure for IB. They currently sell **IB switches** as large as 288 ports in a single chassis. Silver Storm also uses their own IB software stack, called **Quick Silver** that has a reputation for very good performance, reliability, and easy of use.

Silver Storm specializes in multi-protocol switches that allow different types of connections, such as 4X IB, GigE, and Fibre Channel, to all use the same switch. They are also focusing on Virtual I/O that allow you to aggregate traffic from SAN, LAN, and server interconnects into a single pipe. This allows you to take 3 different networks and combine them into a single network connection.

Topspin, which is now part of Cisco, is also pursuing the high-performance computing market as well as other markets that can use the high performance IB interconnect including Grid Computing and database servers. Topspin is producing IB products to combine CPU communication as well as IO communication. They are also shrinking the size of IB switches. They have a 1U switch, called the **Topspin 90**, that has up to 12 ports of 4X Infiniband and also up to two 2 Gbps Fibre Channel ports and six GigE ports to allow the network to be connected to a range of other networks, such as storage networks. They also sell a PCI-X and PCI-Express **HCA**s that have two IB ports.

Mellanox was founded in 1999 and develops and markets IB ASIC's (chips), IB HCA's, switches, and all of the software for controlling an IB fabric. Their first product was delivered in 2001 shortly after the IB specification came out. Mellanox does not sell directly to the public. Rather they sell to other IB companies and to vendors such as cluster vendors.

Currently, Mellanox has a wide range of **HCA** cards. The **Infinihost III Ex** cards fit into a PCI-Express x8 slot and come in SDR (10 Gbps) and DDR (20 Gbps) versions. It comes in two versions, one that has memory on the HCA, and one that does not (called MemFree) that uses the host memory. There is also an HCA, the **Infinihost III Lx** that only uses the MemFree capability that uses the host memory instead of HCA memory. It also comes in 4X SDR (10 Gbps) and DDR (20 Gbps) versions and uses a PCI-Express interface, either as x4 (SDR) or x8 (DDR). Mellanox has also been an active participant in the development of the OpenIB Alliance software project. The OpenIB Alliance software stack has been included in the 2.6.11 kernel. They are also participating in the development of what is called "OpenIB Gen 2," which is the next generation IB stack for the Linux kernel.

There are a number of commercial MPI implementations that support IB. Infinicon ships an MPI library with their hardware. Scali MPI connect, Critical Software MPI, Intel's MPI-2 and Verari System Software (previously MPI Software Technologies) MPI/Pro works with IB hardware.

There are also a number of open-source MPI implementations that now support IB. For example LAM-MPI, Open-MPI, MPICH2-CH3, MVAPICH, and LA-MPI all support various IB stacks.

Infinipath

Infinipath is a new interconnect technology developed by Pathscale. It is a variation on IB taking advantage of the HyperTransport bus in Opteron systems.

Pathscale was founded in 2001 by a team of experienced engineers and scientists from Sun, SGI, and HP. The focus of Pathscale is on software and hardware solutions that enable Linux clusters to achieve new levels of performance and efficiency. Pathscale has recently announced a new technology called "Infinipath." An Infinipath NIC plugs directly into the HyperTransport bus of Opteron systems via the connector HTX connector. The NICs are then connected using standard IB switches and cables.

This combination is unique for several reasons. First it uses standard IB switches that reduces costs compared to developing a new switch. Second, plugging directly into the HyperTransport fabric results in a factor of 2 to 3 times lower latency than other IB NICs. And third, as you increase the speed of the Opteron processors, the latency of the Infinipath NIC will continue to decrease.

At this time, Infinipath is only available for HTX equipped motherboards such as the Iwill DK8-HTX. However, a version is underway that will support PCI-Express for Intel's EM64T processors. While the performance will be good for this NIC, it will not have the same characteristics as the HTX NIC. Pathscale does make the Infinipath ASIC available for OEM's to integrate into their products.

Pathscale ships an MPI for Infinipath. It is based on MPICH 1.2.6 with performance enhancements for Infinipath. An MPICH2 based MPI implementation will follow at a later date. Other open-source MPI implementations are looking to support Infinipath. Pathscale also has an IP over Infinipath capability. Thus any IP based protocol package should work (PVM or TCP/IP based MPIs), albeit with reduced performance.

As you will see in the performance table at the end of this article, Infinipath is the fastest performing interconnect available for clusters today. It has the lowest latency and the smallest N/2 (defined later). Consequently, its performance on many HPC codes is very good.

Myrinet

Myrinet was one of the first high-speed networks developed with clustering in mind. Myricom is a privately held company started in 1994 by Charles L. Seitz. Some of the initial technology was developed under two ARPA sponsored research projects. Myrinet became an IEEE standard in 1998 and is now one of the most popular high-speed cluster interconnects, being the dominant non-Ethernet interconnect on the 26th Top500 list. Myrinet is used by one of the fastest clusters in the world, including MareNostrum in Barcelona Spain.

Myricom has two product lines. Their existing product line, also called the 2G line, has been in successful production use for a number of years. Their new product line, called 10G, which is faster than 2G, is slated to ship at the end of 2005.

Myrinet - 2G

Myricom currently has three NICs in the 2G line. All of them use fiber optic connections. The 'D' card is the lowest price card with a 225 MHz RISC processor and a single fiber port. The newer 'F' card uses a 333 MHz RISC processor with a single fiber port. The 'E' card has two NIC ports and uses a 333 MHz RISC processor. All three NICs are 64-bit PCI-X based cards that are "short"cards (not full length), and are low-profile.

Myrinet 2G is a switched network. The network is based on a Clos design that uses small switch elements to build larger switches. Myrinet currently uses 16-port chips as the basic building block. Myrinet 2G network switches come in several sizes. They have a 2U switch chassis that can accommodate 8-port and 16-port models. For medium size networks, they have a 3U switch chassis for 32-ports, a 5U switch chassis for up to 64-ports, and a 9U switch for up to 128 ports. For larger networks they have a single 14U switch with up to 256 host ports. For even larger networks, you can connect the switches using 'spine' cards creating a Clos network giving full bisection bandwidth to each port. This feature gives a great deal of flexibility when designing a network topology and allows extremely large networks to be constructed.

Since Myrinet is focused on clusters, it has taken advantage of the fact that it is not tied to compatibility with general purpose networks such as Ethernet, TCP, and IP. Consequently, it has made changes to improve networking performance, specifically using a different protocol than TCP. Current protocols for the Myrinet 2G line include GM and MX. They use simpler packets than TCP, resulting in better usage of the packets (less overhead). These packets can be of any length so they can contain packets of other types, such as IP packets. In addition, the MX protocol has been redesigned so that it has about half the zero-packet latency of the GM protocol. To further improve performance, Myrinet also uses an OS-bypass like interface to help latency and reduce CPU overhead.

The data packets on Myrinet are source routed which means that each host must know the route to all of the other hosts through the switch fabric. The result is that the NICs do most of the work and the switches can be very simple. Since the switches are not doing much of the work, each NIC must know the full network topology to route the data and the topology must be fairly static.

There are several MPI's available for Myrinet. Myricom uses an open-source MPI called MPICH-GM for the GM protocol and MPICH-MX for the MX protocol. Both are based on MPICH. Open-MPI will support GM and MX. LAM-MPI support GM. Also since Myrinet can run TCP over GM, you can use any MPI that uses TCP with Myrinet. There is a small performance penalty for running in this manner.

There are several commercial MPI implementations that support Myrinet with GM. Verari System Software (previously MPI Software Technologies) has MPI/Pro and Scali has Scali MPI Connect.

Myrinet - 10G

The Myrinet 10G product is a new product line from Myricom. It takes the bandwidth up to 10 Gbps, the same as 4X Infiniband and 10 GigE. However, the 10G NIC is different than most NICs because it can take on different personalities depending upon which type switch is used. If you plug it into a Myricom switch, it will use MX as the protocol. If you plug it into a 10 GigE switch, it will talk TCP.

The 10G NIC is PCI-Express only (sorry PCI-X world). Right now it uses 8 lanes (x8 in PCI-Express talk). As with the 2G NICs, it includes a processor and firmware. It does the network protocol processing and also uses OS-bypass to improve performance.

The NICs are 10Gbase-CX4, 10GBase-R, or XAUI over ribbon fiber. The copper 10Gbase-CX4 cables can go up to 15m in length and the 10Gbase-R serial fiber cables are good for 10 GigE.

Since the Myrinet design philosophy puts the intelligence in the NIC, they can use the same basic switch concepts for 10G. So they will have Clos networks with full bisection bandwidth to each port. Initially 16-port, 128-port switches. In 2006, they will have 256-port switch.

Quadrics

Quadrics is another company that has focused on very high performance interconnects for clusters. The Quadrics R&D team grew out of the Meiko supercomputer company. The company was incorporated in 1996 in Europe as part of the Finmeccanica Group. It is headquartered in Britain with regional offices in Italy and the US. It's networking technology, **QsNet** and the newest version, QsNetII, are focused on providing the lowest latency possible with the largest bandwidth.

In many ways, Quadrics interconnect technology is similar to other high-speed networking technologies but Quadrics is one of the best performing cluster interconnects on the market. The intelligent Quadrics NICs use an OS-bypass capability to help reduce latency. The NICs also use RDMA concepts to reduce copying. This feature helps both latency and bandwidth.

Quadrics currently is selling two NICs and several switches. The QM400 NIC is a 64-bit 66 MHz PCI NIC and the QM500 is a 64-bit 133 MHz PCI-X NIC for QsNetII. The QM500 is a new card that has faster on-board processors and uses less power than the QM400. However the QM400 card is still one of the fastest cards for a simple PCI bus.

Quadrics is a switched network like most of the networks in this survey. The basic topology of Quadrics (QsNetII) is to connect the NICs with copper cables (up to 10m in length) to 8 port switch chips that are arranged in a fat-tree configuration. They have a 128-port switch and a 16-port switch for QsNet. These switches can be linked together to form large switching networks in a fat-tree topology so each port has full bisection bandwidth. Quadrics also has both a 64-port and an 8-port QsNetII switch. They have an E-series network with up to 128 ports and an R-series network of federated switches scalable to greater than 4096 ports.

Quadrics provides an MPI implementation for QsNet and QsNetII. The MPI is based on MPICH from Argonne Labs and is built in top of the low latency libraries written by Quadrics. It also has shmemp capability for SMP nodes. Verari System Software (previously MPI Software Technologies) has a commercial MPI, ChaMPlon/Pro that supports Quadrics.

Like other vendors, Quadrics provides TCP/IP capability on their networks allowing TCP based MPIs and PVM to work as well. The TCP/IP layer has also been used for Lustre and the PVFS2 file systems. Finally, Quadrics offers the Quadrics Resource Management System (RMS) which provides a high level of parallel supercomputing facilities for UNIX and Linux working on top of QsNet.

Dolphin

Dolphin Interconnect Solutions develops, markets, and supports SCI (Scalable Coherent Interface) networking for clusters. Dolphin began as a group of visionary engineers at the Norwegian computer maker

Norsk Data. Together with engineers from a number of US computer makers, the SCI (Scalable Coherent Interface) standard was developed and made an official IEEE/ANSI standard in the spring of 1992.

Dolphin Interconnect Solutions was formed in the autumn of 1991 to focus on SCI markets and products. During the period from 1992 to 1996, the core SCI technology was developed and clusters using SCI began to appear. Dolphin Interconnect Solutions is based in Oslo, Norway with sales representation around the world. Product and technology development is conducted by the technical staff in Oslo, in close collaboration with a network of partners, consultants, universities and research institutes.

Dolphin ships single, dual, and triple-port SCI cards in PCI form factors and a dual-port PCI-Express card. SCI is a switch less network. The NIC's are connected to one another in some network topology. For example, you can make a simple ring topology, or a 2D torus topology, or even a 3D topology (i.e. by using multiple ports on the NICs, you create one dimension, two dimensional, and three dimensional topologies). For a switched network, you have to worry about how many open ports you have. If you need more, you have to buy another switch or a bigger switch. To expand an existing SCI system, it usually just involves plugging in the NIC into the network in the appropriate fashion. However, a downside is that if there is a downed node in the network, the links to the node must be routed, thus impacting messaging in the remaining system. On the other hand, it avoids a single point of failure of typical switched networks.

The chips on the NICs handle all of the routing so that a packet, if it is not intended for that node, has no impact on the CPU at all. Consequently, the NICs are very efficient at routing packets to the neighboring nodes.

The flexibility of network design is a big plus for SCI. To design a SCI network, you must choose the layout carefully to avoid saturation. Typically, 8-10 nodes will saturate a simple SCI ring, 64-100 nodes will saturate a 2D SCI torus, and 640-1000 nodes will saturate a 3D SCI torus.

There are several MPI implementations available for SCI. There is an open-source version based on MPI 1.2 from MPICH called SCI-MPICH. There is also an open-source MPI based on MPICH2 called SCI-MPICH2.

Dolphin also sells a package called Dolphin Sockets. It provides the necessary glue underneath MPI and PVM implementations. It has been tested with MPICH2, LAM-MPI, PVM, and even Lustre, GFS, and iSCSI. Scali also supplies a commercial MPI implementation that works with SCI.

There Is More, But

There are other network technologies we did not mention, but decided to cover the "mainstream" products and technology that are used by the HPC cluster market. Please **contact** us if you think there is something we missed.

Summary

It's always nice to have a table of data to compare various things. I've created a table to compare the various interconnects I've talked about. In particular, the first table lists latency in microseconds, bandwidth

in Mega-bits per second (Mbps), and the N/2 packet size. The second table lists the cost for 8 nodes, 24 nodes, and 128 nodes.

The N/2 packet size is the size of the packets in bytes that reach half the bandwidth of the interconnect. It is important because it tells if small packets get good bandwidth performance. While latency is important for some codes, other codes depend heavily on bandwidth. Having good bandwidth for small packet sizes is good for these types of codes.

I have made every effort to provide accurate numbers. The information in *Table One* has either been provided by the vendors or obtained from references on the web.

I want to emphasize that *Table One* is designed to give you a "ball park" comparison of each technology. You will never be able to predict how well your application(s) work from examining the table. We recommend that you contact the vendors or integrators and discuss your needs with them. *Table One* can be considered a guide to your discussions. Indeed, the latency and bandwidth were not likely obtained in the exact same manner from each vendor. You should be careful in comparing them to one another.

In particular, **using this table to select an interconnect is not a good idea**. The table is intended as a 50,000 foot look at cluster interconnect technologies. The performance (and the devil) is in the details. The final selection of an interconnect should be done by testing your codes or a set of benchmark codes that you can correlate to your codes.

Table One - Interconnect Summary: Performance Metrics

interconnect	Latency (microseconds)	Bandwidth (MBps)	N/2 (Bytes)
GigE	~29-120	~125	~8,000
GigE: GAMMA	~9.5 (MPI)	~125	~9,000
GigE with Jumbo Frames	29-120	~125	~8,000
GigE: Level 5	15	104.7	NA
10 GigE: Chelsio (Copper)	9.6	~862	~100,000+
Infiniband (4X SDR): Voltaire	~4.0	~875	~2,000
Infinipath	1.29	954	385
Myrinet D (gm)	~7.0	~493	~1,000
Myrinet F (gm)	~5.2	~493	~1,000
Myrinet E (gm)	~5.4	~493	~1,000
Myrinet D (mx)	3.5	~493	~1,000
Myrinet F (mx)	2.6	~493	~1,000
Myrinet E (mx)	2.7	~493	~1,000

Myrinet 10G	2.0	1,200	~1,000
Quadrics	1.7	~875-910	~1,500
Dolphin	4.2	457.5	~800

Table Two is a cost comparison for various cluster sizes and interconnect technologies. Aside from direct price comparisons, it was also created to illustrate how cost scales with cluster size. Please consult the foot notes for the assumptions. Finally, price is often related to performance (i.e. low price may equal low performance). Contact the companies listed in the article for more information about your HPC needs.

Table Two - Interconnect Summary: Pricing

Interconnect	8 Node Cost	24 Node Cost	128 Node Cost
GigE ¹	\$258.00	\$944.00	\$27,328.00
GigE: GAMMA ²	\$258.00	\$944.00	\$27,328.00
GigE with Jumbo Frames ³	\$308.00	\$944.00	\$27,328.00
GigE: Level 5 ⁴	\$4,060	\$12,200	\$83,360.00
10 GigE: Chelsio (Copper) ⁵	\$15,960.00	\$62,280.00	\$447,360.00
Infiniband: Voltaire ⁶	\$11,877.00	\$23,084.00	\$182,083.00
Infinipath ⁷	\$13,810.00	\$26,530.00	\$207,860.00
Myrinet D (gm/mx) ⁸	\$6,345.00	\$6,345.00	\$124,800.00
Myrinet F (gm/mx) ⁹	\$10,345.00	\$10,345.00	\$137,600.00
Myrinet E (gm/mx) ¹⁰	\$14,080.00	\$14,080.00	\$202,240.00
Myrinet 10G ¹¹	\$9,600.00	\$28,800.00	\$153,600.00
Quadrics ¹²	\$19,393.00	\$19,393.00	\$257,478.00
Dolphin ¹³	\$7,800.00	\$7,800.00	\$140,160.00

Notes:

¹ This assumes \$26/NIC using the INTEL 32-bit PCI NICs, a basic 8-port GigE switch at \$50, \$320 for a 24-port switch (SMC GS16-Smart), and for 128-ports a Force10 switch that costs approximately \$24,000.

² This assumes \$26/NIC using the INTEL 32-bit PCI NICs, a basic 8-port GigE switch at \$50, \$320 for a 24-port switch (SMC GS16-Smart), and for 128-ports a Force10 switch that costs approximately \$24,000.

³ This assumes \$26/NIC using the INTEL 32-bit PCI NICs, a SMC8508T 8-port GigE switch at \$100, \$320 for a 24-port switch (SMC GS16-Smart), and for 128-ports a Force10 switch that costs approximately \$24,000.

⁴ This assumes \$495/NIC (list price), a SMC8508T 8-port GigE switch at \$100, \$320 for a 24-port switch (SMC GS16-Smart), and for 128-ports a Force10 switch that costs approximately \$24,000.

⁵ This assumes \$795/NIC (list price). For an 8-port switch, I used an approximate price of \$1,200/port (assuming a Fujitsu switch). For 24 ports, I assumed a price of \$1,800/port using the new Quadrics 10 GigE switch. For 128-ports, I assumed a per port switch cost of \$2,700 using the Force10 E1200 even though it's not a full line rate line card.

⁶ List prices obtained from Voltaire.

⁷ The list prices for the Infinipath NIC is \$795. It uses IB switches to connect the NICs. For 8-port and 24-port pricing, the Voltaire 9024 switch was used with an approximate list price of \$7,450. For 128 ports, a Voltaire ISR 9288 switch was used with an approximate list price of \$106,100.

⁸ List prices obtained from Myricom.

⁹ List prices obtained from Myricom.

¹⁰ List prices obtained from Myricom.

¹¹ List prices for Myrinet 10G are not available at the time of this writing. The price per port is an approximation assuming about \$800 per NIC and \$400 per switch port.

¹² List prices obtained from Quadrics on-line configurator.

¹³ List prices obtained from Dolphin.