Introduction to InfiniBand™ for End Users

Industry-Standard Value and Performance for High Performance Computing and the Enterprise

Paul Grun
InfiniBand® Trade Association

Contents

Acknowledgements

Special thanks to Gilad Shainer at Mellanox for his work and content related to Chapter 7 - InfiniBand Performance Capabilities and Examples.

Thanks to the following individuals for their contributions to this publication: David Southwell, Ariel Cohen, Rupert Dance, Jay Diepenbrock, Jim Ryan, Cheri Winterberg and Brian Sparks.

About the Author

Paul Grun has been working on I/O for mainframes and servers for more than 30 years and has been involved in high performance networks and RDMA technology since before the inception of InfiniBand. As a member of the InfiniBand® Trade Association, Paul served on the Link Working Group during the development of the InfiniBand Architecture; he is a member of the IBTA Steering Committee and is a past chair of the Technical Working Group. He recently served as chair of the IBXoE Working Group, charged with developing the new RDMA over Converged Ethernet (RoCE) specification. He is currently chief scientist for System Fabric Works, Inc., a consulting and professional services company dedicated to delivering RDMA and storage solutions for high performance computing, commercial enterprise and cloud computing systems.

Introduction

InfiniBand™ is not complex. Despite its reputation as an exotic technology, the concepts behind it are surprisingly straightforward. One purpose of this book is to clearly describe the basic concepts behind the InfiniBand Architecture.

Understanding the basic concepts is all well and good, but not very useful unless those concepts deliver a meaningful and significant value to the user of the technology. And that is the real purpose of this book: to draw straight-line connections between the basic concepts behind the InfiniBand Architecture and how those concepts deliver real value to the enterprise data center and to high performance computing (HPC).

The readers of this book are exquisitely knowledgeable in their respective fields of endeavor, whether those endeavors are installing and maintaining a large HPC cluster, overseeing the design and development of an enterprise data center, deploying servers and networks, or building devices/appliances targeted at solving a specific problem requiring computers. But most readers of this book are unlikely to be experts in the arcane details of networking architecture. Although the working title for this book was "InfiniBand for Dummies," its readers are anything but dummies. This book is designed to give its readers, in one short sitting, a view into the InfiniBand Architecture that you probably could not get by reading the specification itself.

The point is that you should not have to hold an advanced degree in obscure networking technologies in order to understand how InfiniBand technology can bring benefits to your chosen field of endeavor and to understand what those benefits are. This minibook is written for you. At the end of the next hour you will be able to clearly see how InfiniBand can solve problems you are facing today.

By the same token, this book is not a detailed technical treatise of the underlying theory, nor does it provide a tutorial on deploying the InfiniBand Architecture. All we can hope for in this short book is to bring a level of enlightenment about this exciting technology. The best measure of our success is if you, the reader, feel motivated after reading this book to learn more about how to deploy the InfiniBand Architecture in your computing environment.

We will avoid the low-level bits and bytes of the elegant architecture (which are complex) and focus on the concepts and applications that are visible to the user of InfiniBand technology. When you decide to move forward with deploying InfiniBand, there is ample assistance available in the market to help you deploy it, just as there is for any other networking technology like traditional TCP/IP/Ethernet. The InfiniBand® Trade Association and its website (www.infinibandta.org) are great sources

of access to those with deep experience in developing, deploying and using the InfiniBand Architecture.

The InfiniBand Architecture emerged in 1999 as the joining of two competing proposals known as Next Generation I/O and Future I/O. These proposals, and the InfiniBand Architecture that resulted from their merger, are all rooted in the Virtual Interface Architecture, VIA. The Virtual Interface Architecture is based on two synergistic concepts: direct access to a network interface (e.g. a NIC) straight from application space, and an ability for applications to exchange data directly between their respective virtual buffers across a network, all without involving the operating system directly in the address translation and networking processes needed to do so. This is the notion of "Channel I/O" – the creation of a "virtual channel" directly connecting two applications that exist in entirely separate address spaces. We will come back to a detailed description of these key concepts in the next few chapters.

InfiniBand is often compared, and not improperly, to a traditional network such as TCP/IP/Ethernet. In some respects it is a fair comparison, but in many other respects the comparison is wildly off the mark. It is true that InfiniBand is based on networking concepts and includes the usual "layers" found in a traditional network, but beyond that there are more differences between InfiniBand and an IP network than there are similarities. The key is that InfiniBand provides a messaging service that applications can access directly. The messaging service can be used for storage, for InterProcess Communication (IPC) or for a host of other purposes, anything that requires an application to communicate with others in its environment. The key benefits that InfiniBand delivers accrue from the way that the InfiniBand messaging service is presented to the application, and the underlying technology used to transport and deliver those messages. This is much different from TCP/IP/Ethernet, which is a byte-stream oriented transport for conducting bytes of information between sockets applications.

The first chapter gives a high-level view of the InfiniBand Architecture and reviews the basic concepts on which it is founded. The next two chapters relate those basic concepts to value propositions which are important to users in the HPC community and in the commercial enterprise. Each of those communities has unique needs, but both benefit from the advantages InfiniBand can bring. Chapter 4 addresses InfiniBand's software architecture and describes the open source software stacks that are available to ease and simplify the deployment of InfiniBand. Chapter 5 delves slightly deeper into the nuances of some of the key elements of the architecture. Chapter 6 describes the efforts undertaken by both the InfiniBand Trade Association and the OpenFabrics Alliance (www.openfabrics.org) to ensure that solutions from a range of vendors will interoperate and that they will behave in conformance with the specification. Chapter 7 describes some comparative performance proof points, and finally Chapter 8 briefly reviews the ways in which the InfiniBand Trade Association ensures that the InfiniBand Architecture continues to move ahead into the future.

Chapter 1 – Basic Concepts

Networks are often thought of as the set of routers, switches and cables that are plugged into servers and storage devices. When asked, most people would probably say that a network is used to connect servers to other servers, storage and a network backbone. It does seem that traditional networking generally starts with a "bottoms up" view, with much attention focused on the underlying wires and switches. This is a very "network centric" view of the world; the way that an application communicates is driven partly by the nature of the traffic. This is what has given rise to the multiple dedicated fabrics found in many data centers today.

A server generally provides a selection of communication services to the applications it supports; one for storage, one for networking and frequently a third for specialized IPC traffic. This complement of communications stacks (storage, networking, etc.) and the device adapters accompanying each one comprise a shared resource. This means that the operating system "owns" these resources and makes them available to applications on an as-needed basis. An application, in turn, relies on the operating system to provide it with the communication services it needs. To use one of these services, the application uses some sort of interface or API to make a request to the operating system, which conducts the transaction on behalf of the application.

InfiniBand, on the other hand begins with a distinctly "application centric" view by asking a profoundly simple question: How to make application accesses to other applications and to storage as simple, efficient and direct as possible? If one begins to look at the I/O problem from this application centric perspective one comes up with a much different approach to network architecture.

The basic idea behind InfiniBand is simple; it provides applications with an easy-to-use *messaging service*. This service can be used to communicate with other applications or processes or to access storage. That's the whole idea. Instead of making a request to the operating system for access to one of the server's communication resources, an application accesses the InfiniBand messaging service directly. Since the service provided is a straightforward messaging service, the complex dance between an application and a traditional

network is eliminated. Since storage traffic can be viewed as consisting of control messages and data messages, the same messaging service is equally at home for use in storage applications.

A key to this approach is that InfiniBand Architecture gives every application direct access to the messaging service. Direct access means that an application need not rely on the operating system to transfer messages. This idea is in contrast to a standard network environment where the shared network resources (e.g. the TCP/IP network stack and associated NICs) are solely owned by the operating system and cannot be accessed by the user application. This means that an application cannot have direct access to the network and instead must rely on the involvement of the operating system to move data from the application's virtual buffer space, through the network stack and out onto the wire. There is an identical involvement of the operating system at the other end of the connection. InfiniBand avoids this through a technique known as stack bypass. How it does this, while ensuring the same degree of application isolation and protection that an operating system would provide, is the one key piece of secret sauce that underlines the InfiniBand Architecture.

InfiniBand provides the messaging service by creating a *channel* connecting an application to any other application or service with which the application needs to communicate. The applications using the service can be either user space applications or kernel applications such as a file system. This application-centric approach to the computing problem is the key differentiator between InfiniBand and traditional networks. You could think of it as a "tops down" approach to architecting a networking solution. Everything else in the InfiniBand Architecture is there to support this one simple goal: provide a message service to be used by an application to communicate *directly* with another application or with storage.

The challenge for InfiniBand's designers was to create these channels between virtual address spaces which would be capable of carrying messages of varying sizes, and to ensure that the channels are isolated and protected. These channels would need to serve as pipes or connections between entirely disjoint virtual address spaces. In fact, the two virtual spaces might even be located in entirely disjoint *physical* address spaces – in other words, hosted by different servers… even over a distance.

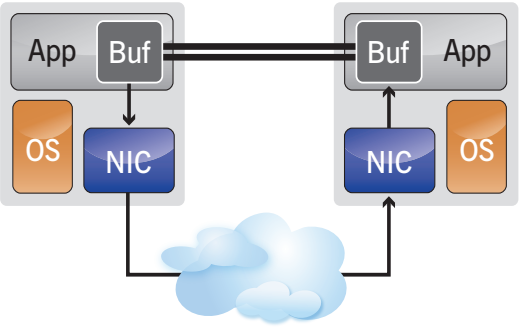


Figure 1: *InfiniBand creates a channel directly connecting an application in its virtual address space to an application in another virtual address space. The two applications can be in disjoint physical address spaces – hosted by different servers.*

It is convenient to give a name to the endpoints of the channel – we'll call them Queue Pairs (QPs); each QP consists of a Send Queue and a Receive Queue, and each QP represents one end of a channel. If an application requires more than one connection, more QPs are created. The QPs are the structure by which an application accesses InfiniBand's messaging service. In order to avoid involving the OS, the applications at each end of the channel must have direct access to these QPs. This is accomplished by mapping the QPs directly into each application's virtual address space. Thus, the application at each end of the connection has direct, virtual access to the channel connecting it to the application (or storage) at the other end of the channel. This is the notion of *Channel I/O*.

Taken altogether then, this is the essence of the InfiniBand Architecture. It creates private, protected channels between two disjoint virtual address spaces, it provides a channel endpoint, called a QP, to the applications at each end of the channel, and it provides a means for a local application to transfer messages directly between applications residing in those disjoint virtual address spaces. Channel I/O in a nutshell.

Having established a channel, and having created a virtual endpoint to the channel, there is one further architectural nuance needed to complete the channel I/O picture and that is the actual method for transferring a message. InfiniBand provides two transfer semantics; a channel semantic sometimes called SEND/RECEIVE and a pair of memory semantics called RDMA READ and RDMA WRITE. When using the channel semantic, the message is received in a data structure provided by the application on the receiving side. This data structure was pre-posted on its receive queue. Thus, the sending side does not have visibility into the buffers or data structures on the receiving side; instead, it simply SENDS the message and the receiving application RECEIVES the message.

The memory semantic is somewhat different; in this case the receiving side application registers a buffer in its virtual memory space. It passes control of that buffer to the sending side which then uses RDMA READ or RDMA WRITE operations to either read or write the data in that buffer.

A typical storage operation may illustrate the difference. The "initiator" wishes to store a block of data. To do so, it places the block of data in a buffer in its virtual address space and uses a SEND operation to send a storage request to the "target" (e.g. the storage device). The target, in turn, uses RDMA READ operations to fetch the block of data from the initiator's virtual buffer. Once it has completed the operation, the initiator uses a SEND operation to return ending status to the initiator. Notice that the initiator, having requested service from the target, was free to go about its other business while the target asynchronously completed the storage operation, notifying the initiator on completion. The data transfer phase and the storage operation, of course, comprise the bulk of the operation.

Included as part of the InfiniBand Architecture and located right above the transport layer is the *software transport interface*. The software transport interface contains the QPs; keep in mind that the queue pairs are the structure by which the RDMA message transport service is accessed. The software transport interface also defines all the methods and mechanisms that an application needs to take full advantage of the RDMA message transport service. For example, the software transport interface describes the methods that applications use to establish a channel between them. An implementation of the software transport interface includes the APIs and libraries needed by an application to create and control the channel and to use the QPs in order to transfer messages.

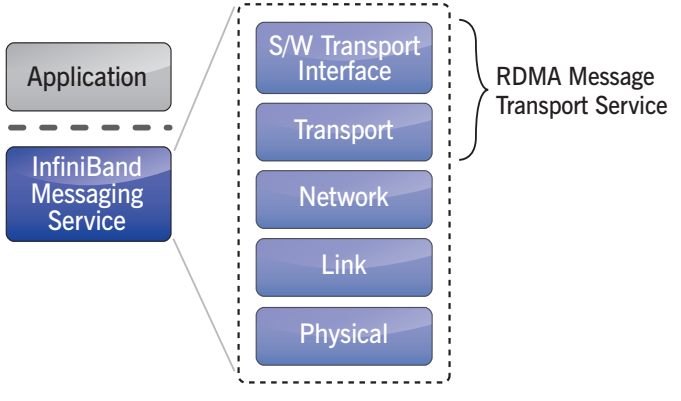


Figure 2: *The InfiniBand Architecture provides an easy-to-use messaging service to applications. The messaging service includes a communications stack similar to the familiar OSI reference model.*

Underneath the covers of the messaging service all this still requires a complete network stack just as you would find in any traditional network. It includes the InfiniBand transport layer to provide reliability and delivery guarantees (similar to the TCP transport in an IP network), a network layer (like the IP layer in a traditional network) and link and physical layers (wires and switches). But it's a special kind of a network stack because it has features that make it simple to transport messages directly between applications' virtual memory, even if the applications are "remote" with respect to each other. Hence, the combination of InfiniBand's transport layer together with the software transport interface is better thought of as a *Remote Direct Memory Access (RDMA) message transport service*. The entire stack taken together, including the software transport interface, comprise the InfiniBand messaging service.

How does the application actually transfer a message? The InfiniBand Architecture provides simple mechanisms defined in the software transport interface for placing a request to perform a message transfer on a queue. This queue is the QP, representing the channel endpoint. The request is called a Work Request (WR) and represents a single quantum of work that the application wants to perform. A typical WR, for example, describes a message that the application wishes to have transported to another application.

The notion of a WR as representing a message is another of the key distinctions between InfiniBand and a traditional network; the InfiniBand Architecture is said to be *message-oriented*. A message can be any size ranging up to 2**31 bytes in size. This means that the entire architecture is oriented around moving messages. This messaging service is a distinctly different service than is provided by other traditional networks such as TCP/IP, which are adept at moving strings of bytes from the operating system in one node to the operating system in another node. As the name suggests, a byte stream-oriented network presents a stream of bytes to an application. Each time an Ethernet packet arrives, the server NIC hardware places the bytes comprising the packet into an anonymous buffer in main memory belonging to the operating system. Once the stream of bytes has been received into the operating system's buffer, the TCP/IP network stack signals the application to request a buffer from the application into which the bytes can be placed. This process is repeated each time a packet arrives until the entire message is eventually received.

InfiniBand on the other hand, delivers a complete message to an application. Once an application has requested transport of a message, the InfiniBand hardware automatically segments the outbound message into a number of packets; the packet size is chosen to optimize the available network bandwidth. Packets are transmitted through the network by the InfiniBand hardware and at the receiving end they are delivered directly into the receiving application's virtual buffer where they are re-assembled into a complete message. Once the entire

message has been received the receiving application is notified. Neither the sending nor the receiving application is involved until the complete message is delivered to the receiving application's virtual buffer.

How, exactly, does a sending application, for example send a message? The InfiniBand software transport interface specification defines the concept of a verb. The word "verb" was chosen since a verb describes action, and this is exactly how an application requests action from the messaging service; it uses a verb. The set of verbs, taken together, are simply a semantic description of the methods the application uses to request service from the RDMA message transport service. For example, "Post Send Request" is a commonly used verb to request transmission of a message to another application.

The verbs are fully defined by the specification, and they are the basis for specifying the APIs that an application uses. But the InfiniBand Architecture specification doesn't define the actual APIs; that is left to other organizations such as the OpenFabrics Alliance, which provides a complete set of open source APIs and software which implements the verbs and works seamlessly with the InfiniBand hardware. A richer description of the concept of "verbs," and how one designs a system using InfiniBand and the verbs, is contained in a later chapter.

The InfiniBand Architecture defines a full set of hardware components necessary to deploy the architecture. Those components are:

- **HCA – Host Channel Adapter.** An HCA is the point at which an InfiniBand end node, such as a server or storage device, connects to the InfiniBand network. InfiniBand's architects went to significant lengths to ensure that the architecture would support a range of possible implementations, thus the specification does not require that particular HCA functions be implemented in hardware, firmware or software. Regardless, the collection of hardware, software and firmware that represents the HCA's functions provides the applications with full access to the network resources. The HCA contains address translation mechanisms under the control of the operating system that allow an application to access the HCA directly. In effect, the Queue Pairs that the application uses to access the InfiniBand hardware appear directly in the application's virtual address space. The same address translation mechanism is the means by which an HCA accesses memory on behalf of a user level application. The application, as usual, refers to virtual addresses; the HCA has the ability to translate these into physical addresses in order to effect the actual message transfer.
- **TCA – Target Channel Adapter.** This is a specialized version of a channel adapter intended for use in an embedded environment such as a storage appliance. Such an appliance may be based on an embedded operating system, or may be entirely based on state machine logic and therefore may not require a standard interface for applications. The concept of a TCA is

not widely deployed today; instead most I/O devices are implemented using standard server motherboards controlled by standard operating systems.

- **Switches** – An InfiniBand Architecture switch is conceptually similar to any other standard networking switch, but molded to meet InfiniBand's performance and cost targets. For example, InfiniBand switches are designed to be "cut through" for performance and cost reasons and they implement InfiniBand's link layer flow control protocol to avoid dropped packets. This is a subtle, but key element of InfiniBand since it means that packets are never dropped in the network during normal operation. This "no drop" behavior is central to the operation of InfiniBand's highly efficient transport protocol.
- **Routers** – Although not currently in wide deployment, an InfiniBand router is intended to be used to segment a very large network into smaller subnets connected together by an InfiniBand router. Since InfiniBand's management architecture is defined on a per subnet basis, using an InfiniBand router allows a large network to be partitioned into a number of smaller subnets thus enabling the deployment of InfiniBand networks that can be scaled to very large sizes, without the adverse performance impacts due to the need to route management traffic throughout the entire network. In addition to this traditional use, specialized InfiniBand routers have also been deployed in a unique way as range extenders to interconnect two segments of an InfiniBand subnet that is distributed across wide geographic distances.
- **Cables and Connectors** – Volume 2 of the Architecture Specification is devoted to the physical and electrical characteristics of InfiniBand and defines, among other things, the characteristics and specifications for InfiniBand cables, both copper and electrical. This has enabled vendors to develop and offer for sale a wide range of both copper and optical cables in a broad range of widths (4x, 12x) and speed grades (SDR, DDR, QDR).

That's it for the basic concepts underlying InfiniBand. The InfiniBand Architecture brings a broad range of value to its users, ranging from ultra-low latency for clustering to extreme flexibility in application deployment in the data center to dramatic reductions in energy consumption and all at very low price/performance points. Figuring out which value propositions are relevant to which situations depends entirely on the destined use.

The next two chapters describe how these basic concepts map onto particular value propositions in two important market segments. Chapter 2 describes how InfiniBand relates to High Performance Computing (HPC) clusters, and Chapter 3 does the same thing for enterprise data centers. As we will see, InfiniBand's basic concepts can help solve a range of problems, regardless whether you are trying to solve a sticky performance problem or if your data center is suffering from severe space, power or cooling constraints. A wise choice of an appropriate interconnect can profoundly impact the solution.

Chapter 2 – InfiniBand for HPC

In this chapter we will explore how InfiniBand's fundamental characteristics map onto addressing the problems being faced today in high performance computing environments.

High performance computing is not a monolithic field; it covers the gamut from the largest "Top 500" class supercomputers down to small desktop clusters. For our purposes, we will generally categorize HPC as being that class of systems where nearly all of the available compute capacity is devoted to solving a single large problem for substantial periods of time. Looked at another way, HPC systems generally don't run traditional enterprise applications such as mail, accounting or productivity applications.

Some examples of HPC applications include atmospheric modeling, genomics research, automotive crash test simulations, oil and gas extraction models and fluid dynamics.

HPC systems rely on a combination of high-performance storage and low-latency InterProcess Communication (IPC) to deliver performance and scalability to scientific applications. This chapter focuses on how the InfiniBand Architecture's characteristics make it uniquely suited for HPC. With respect to high performance computing, InfiniBand's low-latency/high-bandwidth performance and the nature of a channel architecture are crucially important.

Ultra-low latency for:

- Scalability
- Cluster performance

Channel I/O delivers:

- Scalable storage bandwidth performance
- Support for shared disk cluster file systems and parallel file systems

HPC clusters are dependent to some extent on low latency IPC for scalability and application performance. Since the processes comprising a cluster application are distributed across a number of cores, processors and servers, it is clear that a low-latency IPC interconnect is an important determinant of cluster scalability and performance.

In addition to demand for low-latency IPC, HPC systems frequently place

enormous demands on storage bandwidth, performance (measured in I/Os per second), scalability and capacity. As we will demonstrate in a later section, InfiniBand's channel I/O architecture is well suited to high-performance storage and especially parallel file systems.

InfiniBand support for HPC ranges from upper layer protocols tailored to support the Message Passing Interface (MPI) in HPC to support for large scale high-performance parallel file systems such as Lustre. MPI is the predominant messaging middleware found in HPC systems.

InfiniBand MPI Clusters

MPI is the de facto standard and dominant model in use today for communication in a parallel system. Although it is possible to run MPI on a shared memory system, the more common deployment is as the communication layer connecting the nodes of a cluster. Cluster performance and scalability are closely related to the rate at which messages can be exchanged between cluster nodes; the classic low latency requirement. This is native InfiniBand territory and its clear stock in trade.

MPI is sometimes described as communication middleware, but perhaps a better, more functional description would be to say that it provides a *communication service* to the distributed processes comprising the HPC application.

To be effective, the MPI communication service relies on an underlying messaging service to conduct the actual communication messages from one node to another. The InfiniBand Architecture, together with its accompanying application interfaces, provides an RDMA messaging service to the MPI layer.

In the first chapter, we discussed direct application access to the messaging service as being one of the fundamental principles behind the InfiniBand Architecture; applications use this service to transfer messages between them. In the case of HPC, one can think of the processes comprising the HPC application as being the clients of the MPI communication service, and MPI, in turn as the client of the underlying RDMA message transport service. Viewed this way, one immediately grasps the significance of InfiniBand's channel-based message passing architecture; its ability to enable its client, the MPI communications middleware in this case, to communicate among the nodes comprising the cluster without involvement of any CPUs in the cluster. InfiniBand's copy avoidance architecture and stack bypass deliver ultra-low application-to-application latency and high bandwidth coupled with extremely low CPU utilization.

MPI is easily implemented on InfiniBand by deploying one of a number of implementations which take full advantage of the InfiniBand Architecture.

Storage and HPC

An HPC application comprises a number of processes running in parallel. In an HPC cluster, these processes are distributed; the distribution could be across the cores comprising a processor chip, or across a number of processor chips comprising a server, or they could be distributed across a number of physical servers. This last model is the one most often thought of when thinking about HPC clusters.

Depending on the configuration and purpose for the cluster, it is often the case that each of the distributed processes comprising the application requires access to storage. As the cluster is scaled up (in terms of the number of processes comprising the application) the demand for storage bandwidth increases. In this section we'll explore two common HPC storage architectures, shared disk cluster file systems and parallel file systems, and discuss the role channel I/O plays. Other storage architectures, such as network attached storage, are also possible.

A shared disk cluster file system is exactly as its name implies and is illustrated in the diagram below; it permits distributed processes comprising the cluster to share access to a common pool of storage. Typically in a shared disk file system, the file system is distributed among the servers hosting the distributed application with an instance of the file system running on each processor and a side band communication channel between file system instances used for distributed lock management. By creating a series of independent channels, each instance of the distributed file system is given direct access to the shared disk storage system. The file system places a certain load on the storage system which is equal to the sum of the loads placed by each instance of the file system. By distributing this load across a series of unique channels, the channel I/O architecture provides benefits in terms of parallel accesses and bandwidth allocation.

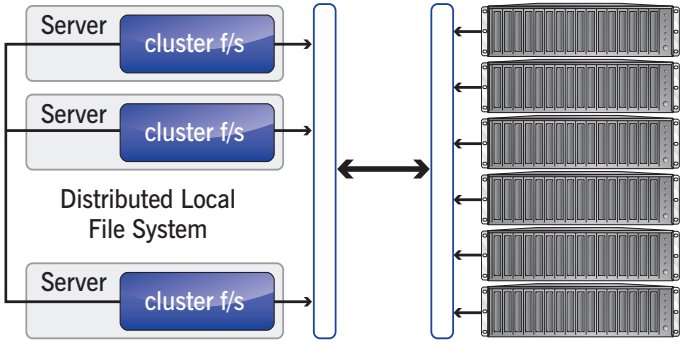


Figure 3: ***Shared disk cluster file system.*** *Each instance of the distributed file system has a unique connection to the elements of the storage system.*

One final point; the performance of a shared disk file system and hence the overall system is dependent on the speed with which locks can be exchanged between file system instances. InfiniBand's ultra low latency characteristic allows the cluster to extract maximum performance from the storage system.

A parallel file system such as the Lustre file system is similar in some ways to a shared disk file system in that it is designed to satisfy storage demand from a number of clients in parallel, but with several distinct differences. Once again, the processes comprising the application are distributed across a number of servers. The file system however, is sited together with the storage devices rather than being distributed. Each process comprising the distributed application is served by a relatively thin file system client.

Within the storage system, the control plane (file system metadata), is often separated from the data plane (object servers) containing user data. This promotes parallelism by reducing bottlenecks associated with accessing file system metadata, and it permits user data to be distributed across a number of storage servers which are devoted to strictly serving user data.

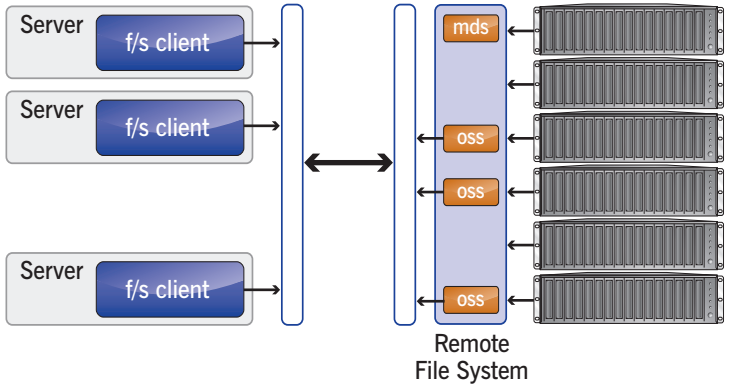


Figure 4: ***A parallel file system.*** *A thin file system client co-located with each instance of the distributed application creates parallel connections to the metadata server(s) and the object servers. The large arrow represents the physical InfiniBand network.*

Each file system client creates a unique channel to the file system metadata. This accelerates the metadata look-up process and allows the metadata server to service a fairly large number of accesses in parallel. In addition, each file system client creates unique channels to the object storage servers where user data is actually stored.

By creating unique connections between each file system client and the storage subsystem, a parallel file system is perfectly positioned to reap maximum advantage from InfiniBand's channel I/O architecture. The unique and persistent low latency connections to the metadata server(s) allow a high degree of scalability by eliminating the serialization of accesses sometimes

experienced in a distributed lock system. By distributing the data across an array of object servers and by providing each file system client with a unique connection to the object servers, a high degree of parallelism is created as multiple file system clients attempt to access objects. By creating unique and persistent low-latency connections between each file system client and the metadata servers, and between each file system client and the object storage servers, the Lustre file system delivers scalable performance.

Upper Layer Protocols for HPC

The concept of upper layer protocols (ULPs) is described in some detail in Chapter 4 – Designing with InfiniBand. For now, it is sufficient to say that open source software to support the InfiniBand Architecture and specifically focused on HPC is readily available, for example, from the OpenFabrics Alliance. There are ULPs available to support file system accesses such as the Lustre file system, MPI implementations and a range of other APIs necessary to support a complete HPC system are also provided.

Chapter 3 – InfiniBand for the Enterprise

The previous chapter addressed applications of InfiniBand in HPC and focused on the values an RDMA messaging service brings to that space – ultra-low, application-to-application latency and extremely high bandwidth all delivered at a cost of almost zero CPU utilization. As we will see in this chapter, InfiniBand brings an entirely different set of values to the enterprise. But those values are rooted in precisely the same set of underlying principles.

Of great interest to the enterprise are values such as risk reduction, investment protection, continuity of operations, ease of management, flexible use of resources, reduced capital and operating expenses and reduced power, cooling and floor space costs. Naturally, there are no hard rules here; there are certainly areas in the enterprise space where InfiniBand's performance (latency, bandwidth, CPU utilization) is of extreme importance. For example, certain applications in the financial services industry (such as arbitrage trading) demand the lowest possible application-to-application latencies at nearly any cost. Our purpose in this chapter is to help the enterprise user understand more clearly how the InfiniBand Architecture applies in this environment.

Reducing expenses associated with purchasing and operating a data center means being able to do more with less – less servers, less switches, fewer routers and fewer cables. Reducing the number of pieces of equipment in a data center leads to reductions in floor space and commensurate reductions in demand for power and cooling. In some environments, severe limitations on available floor space, power and cooling effectively throttle the data center's ability to meet the needs of a growing enterprise, so overcoming these obstacles can be crucial to an enterprise's growth. Even in environments where this is not the case, each piece of equipment in a data center demands a certain amount of maintenance and operating cost, so reducing the number of pieces of equipment leads to reductions in operating expenses. Aggregated across an enterprise, these savings are substantial. For all these reasons, enterprise data center owners are incented to focus on the efficient use of IT resources. As we will see shortly, appropriate application of InfiniBand can advance each of these goals.

Simply reducing the amount of equipment is a noble goal unto itself, but in an environment where workloads continue to increase, it becomes critical to do with less. This is where the InfiniBand Architecture can help; it enables the enterprise data center to serve more clients with a broader range of applications with faster response times while reducing the number of servers, cables and switches required. Sounds like magic.

Server virtualization is a strategy being successfully deployed now in an attempt to reduce the cost of server hardware in the data center. It does so by taking advantage of the fact that many data center servers today operate at very low levels of CPU utilization; it is not uncommon for data center servers to consume less than 15-20 percent of the available CPU cycles. By hosting a number of Virtual Machines (VMs) on each physical server, each of which hosts one or more instances of an enterprise application, virtualization can cleverly consume CPU cycles that had previously gone unused. This means that fewer physical servers are needed to support approximately the same application workload.

Even though server virtualization is an effective strategy for driving up consumption of previously unused CPU cycles, it does little to improve the *effective utilization (efficiency)* of the existing processor/memory complex. CPU cycles and memory bandwidth are still devoted to executing the network stack at the expense of application performance. Effective utilization, or efficiency, is the ratio of a server resource, such as CPU cycles or memory bandwidth that is devoted directly to executing an application program compared to the use of server resources devoted to housekeeping tasks. Housekeeping tasks principally include services provided to the application by the operating system devoted to conducting routine network communication or storage activities on behalf of an application. The trick is to apply technologies that can fundamentally improve the server's overall effective utilization, without assuming unnecessary risk in deploying them. If successful, the data center owner will achieve much, much higher usage of the data center servers, switches and storage that were purchased. That is the subject of this chapter.

Devoting Server Resources to Application Processing

Earlier we described how an application using InfiniBand's RDMA messaging service avoids the use of the operating system (using operating system bypass) in order to transfer messages between a sender and a receiver. This was described as a performance improvement since avoiding operating system calls and unnecessary buffer copies significantly reduces latency. But operating system bypass produces another effect of much greater value to the enterprise data center.

Server architecture and design is an optimization process. At a given server price point, the server architect can afford to provide only a certain amount

of main memory; the important corollary is that there is a finite amount of memory bandwidth available to support application processing at that price point. As it turns out, main *memory bandwidth* is a key determinant of server performance. Here's why.

The precious memory bandwidth provided on a given server motherboard must be allocated between application processing and I/O processing. This is true because traditional I/O, such as TCP/IP networking, relies on main memory to create an "anonymous buffer pool," which is integral to the operation of the network stack. Each inbound packet must be placed into an anonymous buffer and then copied out of the anonymous buffer and delivered to a user buffer in the application's virtual memory space. Thus, each inbound packet is copied at least twice. A single 10Gb/s Ethernet link, for example, can consume between 20Gb/s and 30Gb/s of memory bandwidth to support inbound network traffic.

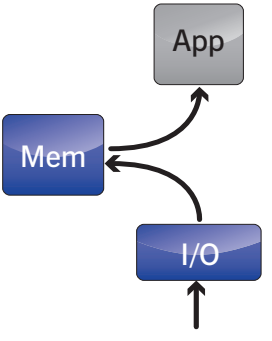


Figure 5: *Each inbound network packet consumes from two to three times the memory bandwidth as the inbound packet is placed into an anonymous buffer by the NIC and then subsequently copied from the anonymous buffer and into the application's virtual buffer space.*

The load offered to the memory by network traffic is largely dependent on how much aggregate network traffic is created by the applications hosted on the server. As the number of processor sockets on the motherboard increases, and as the number of cores per socket increases, and as the number of execution threads per core increases, the offered network load increases approximately proportionally. And each increment of offered network load consumes two to three times as much memory bandwidth as the network's underlying signaling rate. For example, If an application consumes 10Gb/s of inbound network traffic (which translates into 20Gb/s of memory bandwidth consumed) then multiple instances of an application running on a multicore CPU will consume proportionally more network bandwidth and twice the memory bandwidth. And, of course, executing the TCP/IP network protocol stack consumes CPU processor cycles. The cycles consumed in executing the network protocol

and copying data out of the anonymous buffer pool and into the application's virtual buffer space are cycles that cannot be devoted to application processing; they are purely overhead required in order to provide the application with the data it requested.

Consider: When one contemplates the purchase of a server such as a server blade, the key factors that are typically weighed against the cost of the server are power consumption and capacity. Processor speed, number of cores, number of threads per core and memory capacity are the major contributors to both power consumption and server performance. So if you could reduce the number of CPUs required or the amount of memory bandwidth consumed, but without reducing the ability to execute user applications, significant savings can be realized in both capital expense and operating expense. If using the available CPU cycles and memory bandwidth more efficiently enables the data center to reduce the number of servers required, then capital and operating expense improvements multiple dramatically.

And this is exactly what the InfiniBand Architecture delivers; it dramatically improves the server's "effective utilization" (its efficiency) by dramatically reducing demand on the CPU to execute the network stack or copy data out of main memory and it dramatically reduces I/O demand on memory bandwidth, making more memory bandwidth available for application processing. This allows the user to direct precious CPU and memory resources to useful application processing and allows the user to reap the full capacity of the server for which the user has paid.

Again, server virtualization increases the number of applications that a server can host by "time slicing" the server hardware among the hosted virtual machines. Channel I/O decreases the amount of time the server devotes to housekeeping tasks and therefore increases the amount of time that the server can devote to application processing. It does this by lifting from the CPU and memory the burden of executing I/O and copying data.

A Flexible Server Architecture

Most modern servers touch at least two different I/O fabrics – an Ethernet network and a storage fabric, commonly Fibre Channel. In some cases there is a third fabric dedicated to IPC. For performance and scalability reasons, a server often touches several instances of each type of fabric. Each point of contact between a server and one of the associated fabrics is implemented with a piece of hardware – either a Network Interface Card (NIC) for networking or a Host Bus Adapter (HBA) for storage. The server's total I/O bandwidth is the sum of the bandwidths available through these different fabrics. Each of these fabrics places a certain load both on memory bandwidth and CPU resources.

The allocation of the total available I/O bandwidth between storage and networking is established at the time the server is provisioned with I/O hard-

ware. This process begins during motherboard manufacture in the case of NICs soldered directly to the motherboard (LAN on Motherboard – "LOM") and continues when the server is provisioned with a complement of PCI Express-attached Host Bus Adapters (HBAs, used for storage) and NICs. Even though the complement of PCI Express adapters can be changed, the overall costs of doing so are prohibitive; so by and large the allocation of I/O bandwidth between storage and networking is essentially fixed by the time the server has been installed.

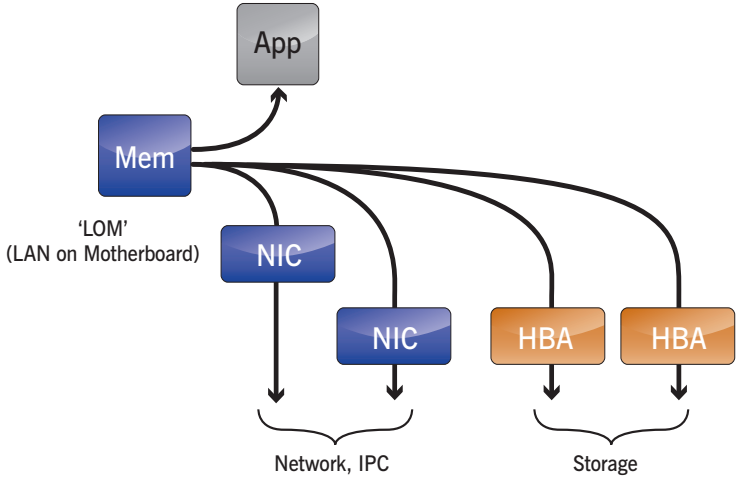


Figure 6: *Memory bandwidth is divided between application processing and I/O. I/O bandwidth allocation between storage, networking and IPC is predetermined by the physical hardware, the NICs and HBAs, in the server.*

The ideal ratio of storage bandwidth to networking bandwidth is solely a function of the application(s) being hosted by the server… each different enterprise application will present a different characteristic workload on the I/O subsystem. For example, a database application may create a high disk storage workload and a relatively modest networking workload as it receives queries from its clients and returns results. Hence, if the mix of enterprise applications hosted by a given server changes, its ideal allocation of I/O bandwidth between networking and storage also changes. This is especially true in virtualized servers, where each physical server hosts a number of application containers (Virtual Machines – VMs), each of which may be presenting a different characteristic workload to the I/O subsystem.

This creates a bit of a conundrum for the IT manager. Does the IT manager over-provision each server with a maximum complement of storage and networking devices, and in what proportions? Does the IT manager lock a given application, or application mix, down to a given server and provision that server

with the idealized I/O complement of storage and networking resources? What happens to the server as the workload it is hosting is changed? Will it still have proportional access to both networking and storage fabrics?

In a later chapter we introduce the notion of a "unified fabric" as being a fabric that is optimal for storage, networking and IPC. An InfiniBand fabric has the characteristics that make it well suited for use as a unified fabric. For example, its message orientation and its ability to efficiently handle the small message sizes that are often found in network applications, as well as the large message sizes that are common in storage. In addition, the InfiniBand Architecture contains features such as a virtual lane architecture specifically designed to support various types of traffic simultaneously.

Carrying storage and networking traffic efficiently over a single InfiniBand network using the RDMA message transport service means that the total number of I/O adapters required (HBAs and NICs) can be radically reduced. This is a significant saving, and even more significant when one factors in the 40Gb/s data rates of commonly available commodity InfiniBand components versus leading edge 10Gb/s Ethernet. For our purposes here, we are ignoring the approximately 4x improvement in price/performance provided by InfiniBand.

But far more important than the simple cost savings associated with the reduction in the number of required adapter types is this: by using an InfiniBand network, a single type of resource (HCAs, buffers, queue pairs and so on) is used for both networking and storage. Thus the balancing act between networking and storage is eliminated, meaning that the server will always have the correct proportion of networking and storage resources available to it – even as the mix of applications changes and the characteristic workload offered by the applications changes!

This is a profound observation; it means that it is no longer necessary to overprovision a server with networking and storage resources in anticipation of an unknown application workload; the server will always be naturally provisioned with I/O resources.

Investment Protection

Enterprise data centers typically represent a significant investment in equipment and software accumulated over time. Only rarely does an opportunity arise to install a new physical plant and accompanying control and management software. Accordingly, valuable new technologies must be introduced to the data center in such a way as to provide maximum investment protection and to minimize the risk of disruption to existing applications. For example, a data center may contain a large investment in SANs for storage which it would be impractical to simply replace. In this section we explore some methods to allow the data center's servers to take advantage of InfiniBand's messaging service while protecting existing investments.

Consider the case of a rack of servers, or a chassis of bladed servers. As usual, each server in the rack requires connection to up to three types of fabrics, and in many cases, multiple instances of each type of fabric. There may be one Ethernet network dedicated to migrating virtual machines, another Ethernet network dedicated to performing backups and yet another one for connecting to the internet. Connecting to all these networks requires multiple NICs, HBAs, cables and switch ports.

Earlier we described the advantages that accrue in the data center by reducing the types of I/O adapters in any given server. These included reductions in the number and types of cables, switches and routers and improvements in the server's ability to host changing workloads because it is no longer limited by the particular I/O adapters that happen to be installed.

Nevertheless, a server may still require access to an existing centrally managed pool of storage such as a Fibre Channel SAN which requires that each server be equipped with a Fibre Channel HBA. On the face of it, this is at odds with the notion of reducing the number of I/O adapters. The solution is to replace the FC HBA with a virtual HBA driver. Instead of using its own dedicated FC HBA, the server uses a virtual device adapter to access a physical Fibre Channel adapter located in an external I/O chassis. The I/O chassis provides a pool of HBAs which are shared among the servers. The servers access the pool of HBAs over an InfiniBand network.

The set of physical adapters thus eliminated from each server are replaced by a single physical InfiniBand HCA and a corresponding set of virtual device adapters (VNICs and VHBAs). By so doing, the servers gain the flexibility afforded by the one fat pipe, benefit from the reduction in HBAs and NICs, and retain the ability to access the existing infrastructure, thus preserving the investment.

A typical server rack contains a so-called "top of rack" switch which is used to aggregate all of a given class of traffic from within the rack. This is an ideal place to site an I/O chassis, which thus provides the entire rack with virtualized access to data center storage resources and with access to the data center network and with access to an IPC service for communicating with servers located in other racks.

Cloud Computing – An Emerging Data Center Model

Cloud computing has emerged as an interesting variant on enterprise computing. Although widespread adoption has only recently occurred, the fundamental concepts beneath the idea of a "cloud" have been under discussion for quite a long period under different names such as utility computing, grid computing and agile or flexible computing. Although cloud computing is marketed and expressed in a number of ways, at its heart it is an expression of the idea of flexible allocation of compute resources (including processing, network and

storage) to support an ever-shifting pool of applications. In its fullest realization, the "cloud" represents a pool of resources capable of hosting a vast array of applications, the composition of which is regularly shifting. This is dramatically different from the classical enterprise data center in which the applications hosted by the data center are often closely associated with, and in some cases statically assigned to, a specific set of hardware and software resources – a server. For example, the classical three-tier data center is an expression of this relatively static allocation of servers to applications.

Given the above discussion concerning the flexible allocation of I/O resources to applications, it is immediately clear that InfiniBand's ability to transport messages on behalf of any application is a strong contributor to the idea that applications must not be enslaved to the specific I/O architecture of the underlying server on which the application is hosted. Indeed, the cloud computing model is the very embodiment of the idea of flexible computing; the ability to flexibly allocate processors, memory, networking and storage resources to support application execution anywhere within the cloud. By using the set of Upper Layer Protocols described briefly in the first chapter, the application is provided with a simple way to use the InfiniBand messaging service. By doing so, the application's dependence on a particular mix of storage and networking I/O is effectively eliminated. Thus, the operator of a cloud infrastructure gains an unprecedented level of flexibility in hosting applications capable of executing on any available compute resource in the cloud.

One last point should be made before we leave behind the topic of cloud computing. Cloud infrastructures, as they are being deployed today, represent some of the highest densities yet seen in the computing industry. Cloud systems are often delivered in container-like boxes that push the limits of compute density, storage density, power and cooling density in a confined physical space. In that way, they share the same set of obstacles found in some of the highest density commercial data centers, such as those found in the financial services industry in Manhattan. These data centers face many of the same stifling constraints on available power, cooling and floor space.

We mentioned in an earlier section that enterprise data centers are often constrained in their ability to adopt relevant new technologies since they must balance the advantages gained by deploying a new technology against requirements to protect investments in existing technologies. In that way, many of the cloud computing infrastructures now coming online are very different from their brethren in other commercial enterprises. Because of the modular way that modern cloud infrastructures are assembled and deployed, the opportunity does exist for simple adoption of a vastly improved I/O infrastructure like InfiniBand. For example, a typical cloud infrastructure might be composed of a series of self-contained containers resembling the familiar shipping container seen on trucks. This container, sometimes called a POD, contains racks,

servers, switches, networks and infrastructure. They are designed to be rolled in, connected to a power and cooling source and be operational right away. A POD might comprise a complete, self-contained data center or it may represent a granular unit of expansion in a larger cloud installation.

By deploying InfiniBand as the network internal to the container, and by taking full advantage of its messaging service, the container manufacturer can deliver to his or her client the densest possible configuration presenting the highest effective utilization of compute, power and cooling resources possible. Such a container takes full advantage of InfiniBand's complete set of value propositions:

- Its low latency, and extremely high bandwidth removes bottlenecks that prevent applications from delivering maximum achievable performance.
- The ease and simplicity with which an application directly uses InfiniBand's messaging service means that precious CPU cycles and, more importantly, memory bandwidth, are devoted to application processing and not to housekeeping I/O chores. This drives up the effective utilization of available resources. Thus, the overall return on investment is higher, and the delivered performance per cubic foot of floor space and per watt of power consumption, is higher.
- By providing each server with "one fat pipe," the server platform gains an intrinsic guarantee that its mixture of storage I/O, networking I/O and IPC will always be perfectly balanced.
- InfiniBand's efficient management of the network topology offers excellent utilization of the available network connectivity. Because InfiniBand deploys a centralized subnet manager, switch forwarding tables are programmed in such a way that routes are balanced and available links are well used. This efficient topology handling scales to thousands of nodes and thus satisfies the needs of large cloud installations.
- InfiniBand's native, commodity bandwidths (currently 40Gb/s) are nearly four times higher than 10Gb/s Ethernet, at prices that have historically been extremely competitive compared to commodity 1Gb/s or 10Gb/s Ethernet solutions. Since each InfiniBand cable carries nearly four times the bandwidth of a corresponding 10Gb/s Ethernet cable, the number of connectors, cables, switches and so on that are required is dramatically reduced.
- And finally, the simplified communication service provided by an InfiniBand network inside the container simplifies the deployment of applications on the pool of computing, storage and networking resources provided by the container and reduces the burden of migrating applications onto available resources.

InfiniBand in the Distributed Enterprise

Certain classes of enterprises are facing a long-standing question; how to gather and store large amounts of data from a globally distributed enterprise and make it available to a broad field of geographically dispersed consumers? There are many specific examples of this class of enterprise; consider the globally distributed trading firm which for performance and logistical reasons must maintain the bulk of its infrastructure in New York City, and yet support a robust trading office in London. Historically, various forms of sophisticated data replication have been deployed in an attempt to address problems like these. This is in addition to the common "disaster recovery" problem – how does one maintain a reliable, online backup of essential data and infrastructure to insure against loss of continuous operation due to regional disaster?

A few more examples of important applications conducted over longer distances include:

- **Data center replication** – with ever-growing storage capacities, data centers require a means of replicating this data over considerable distances for disaster tolerance reasons. Should a data center recovery be required, the time to restore from remote backup depends on moving very large data sets over WAN connections. With data center storage growing exponentially, it is critical to extract the most throughput from often expensive telecommunications network links. InfiniBand over the WAN brings near-local efficiency to networks of arbitrary length.
- **Latency-sensitive messaging** – high-frequency trading applications must crunch market feed data into trade orders in as short a time as possible – microseconds matter in this space. When trading algorithms require input from geographically distributed exchanges the network's latency, jitter and link determinism are critical components of system performance. InfiniBand connections over the WAN are a perfect match for these applications, and interface seamlessly with InfiniBand-based server farms on each side.
- **Virtualization, cloud computing** – migrating very large data sets to and from cloud data centers is extremely bandwidth-intensive, as is migrating virtual machine images between data centers. Long-distance RDMA provides an ideal mechanism for internal data flows such as these.

For all these reasons, InfiniBand over the WAN has emerged as an interesting concept. Experiments with the idea of extending RDMA over the WAN have been underway for several years, but recent developments have encouraged and driven widespread interest in the technology.

Consider an example: Two or more regional data centers, each of which contains compute resources (workstations or application servers) and large scale storage. From the perspective of a local user accessing data located on the remote site, the remote files do not become available until the data has been

physically copied to the local site. This is because enterprises rely on a file transfer protocol to replicate data from site to site. The time delay to accomplish the transfer can often be measured in minutes or hours depending on file sizes, the capacity of the underlying network and the performance of the storage system. This poses a difficult problem in synchronizing storage between sites.

Suppose, instead of copying files from the remote file system to the local file system, that the remote file system could simply be mounted to the local user's workstation in exactly the same way that the local file system is mounted? This is exactly the model enabled by executing InfiniBand over the WAN. Over long haul connections with relatively long flight times, the InfiniBand protocol offers extremely high bandwidth efficiency across distances that cause TCP/IP to stumble.

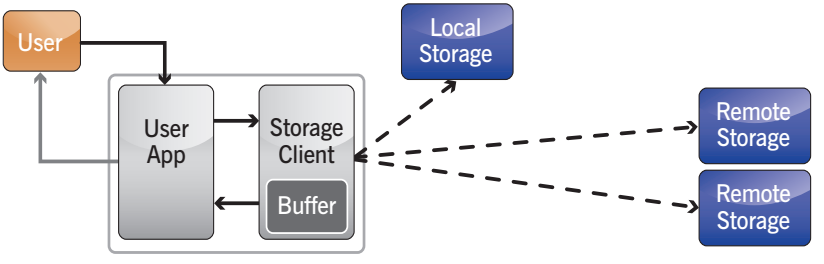


Figure 7.
An enterprise application reads data through a storage client
The storage client connects to each storage server via RDMA
Connections to remote storage servers are made via the WAN
Thus, the user has direct access to all data stored anywhere on the system

It turns out that the performance of any windowing protocol is a function of the so-called bandwidth-delay product; this product represents the maximum amount of data that can be "stored" in transit on the WAN at any given moment. Once the limit of the congestion window has been reached, the transport must stop transmitting until some of the bytes in the pipe can be drained at the far end. In order to achieve maximum utilization of the available WAN bandwidth, the transport's congestion window must be sufficiently large to keep the wire continuously full. As the wire length increases, or as the bandwidth on the wire increases, so does the bandwidth-delay product. And hence, the "congestion window" that is basic to any congestion managed network such as InfiniBand or TCP/IP must grow very large in order to support the large bandwidth-delay products commonly found on high speed WAN links. Because of the fundamental architecture of the InfiniBand transport, it is easily able to support these exceedingly large bandwidth-delay products.

On the other hand, lossy flow control protocols such as TCP degrade rapidly as the bandwidth-delay product increases, causing turbulent rather

than smooth packet flow. These effects cause a rapid roll-off in achieved bandwidth performance as link distance or wire bandwidth increases. These effects are already pronounced at 10Gbits/s, and become much worse at 40 or 100Gbits/s and beyond.

Because the InfiniBand transport, when run over the WAN is able to effectively manage the bandwidth-delay product of these long links, it becomes possible to actually mount the remote file system directly to the local user's workstation and hence to deliver performance approximately equivalent to that which the local user would experience when accessing a local file system. Of course speed of light delays are an unavoidable byproduct of access over the WAN, but for many applications the improvement in response time from minutes or hours to milliseconds or seconds represents an easy tradeoff.

Long distance InfiniBand links may span Campus or Metro Area Networks or even Global WANs. Over shorter optical networks, applications directly benefit from InfiniBand's low latency, low jitter, low CPU loading and high throughput. Examples of such applications include:

- **Data Center Expansion** – spill an InfiniBand network from a primary data center to nearby buildings using multiple optical InfiniBand connections. This approach causes no disruption to data center traffic, is transparent to applications and sufficiently low latency as to be ignorable to operations.
- **Campus System Area Networks** – InfiniBand makes an excellent System Area Network, carrying storage/ IPC and Internet traffic over its unified fabric. Campus Area InfiniBand links allow islands of compute, storage and I/O devices such as high-resolution display walls to be federated and shared across departments. Location agnostic access performance greatly assists in the efficient utilization of organization-wide resources.

Chapter 4 – Designing with InfiniBand

As with any network architecture, the InfiniBand Architecture provides a service; in the case of InfiniBand it provides a messaging service. That message service can be used to transport IPC messages, or to transport control and data messages for storage, or to transport messages associated with any other of a range of usages. This makes InfiniBand different from a traditional TCP/IP/Ethernet network which provides a "byte stream" transport service, or Fibre Channel which provides a transport service specific to the Fibre Channel wire protocol. By "application," we mean, for example, a user application using the sockets interface to conduct IPC, or a kernel level file system application executing SCSI block level commands to a remote block storage device.

The key feature that sets InfiniBand apart from other network technologies is that InfiniBand is designed to provide message transport services directly to the application layer, whether it is a user application or a kernel application. This is very different from a traditional network which requires the application to solicit assistance from the operating system to transfer a stream of bytes. At first blush, one might think that such application level access requires the application to fully understand the networking and I/O protocols provided by the underlying network. Nothing could be further from the truth.

The top layer of the InfiniBand Architecture defines a *software transport interface*; this section of the specification defines the methods that an application uses to access the complete set of services provided by InfiniBand.

An application accesses InfiniBand's message transport service by posting a Work Request (WR) to a work queue. As described earlier, the work queue is part of a structure, called a Queue Pair (QP) representing the endpoint of a channel connecting the application with another application. A QP contains two work queues; a Send Queue and a Receive Queue, hence the expression QP. A WR, once placed on the work queue, is in effect an instruction to the InfiniBand RDMA message transport to transmit a message on the channel, or perform some sort of control or housekeeping function on the channel.

The methods used by an application to interact with the InfiniBand transport are called verbs and are defined in the software transport interface section of the specification. An application uses a POST SEND verb, for example, to

request that the InfiniBand transport send a message on the given channel. This is what is meant when we say that the InfiniBand network stack "goes all the way up to the application layer." It provides methods that are used directly by an application to request service from the InfiniBand transport. This is very different from a traditional network, which requires the application to solicit assistance from the operating system in order to access the server's network services.

Application ("Consumer of InfiniBand message services"):
– posts 'work requests' to a queue
– each work request represents a message…a unit of work

Channel interface (verbs):
– allows the consumer to request services

InfiniBand Channel interface provider:
– Maintains work queues
– Manages address translation
– Provides completion and event mechanisms

IB Transport:
– Packetizes messages
– Provides transport service
  - reliable/unreliable, connected/unconnected, datagram
– Implements RDMA protocol
  - send/receive, RDMA r/w, Atomics
– Implements end-to-end reliability
– Assures reliable delivery

Figure 8.

Building on Top of the Verbs

The verbs, as defined, are sufficient to enable an application developer to design and code the application for use directly on top of an InfiniBand network. But if the application developer were to do so, he or she would need to create a set of customized application interface code, at least one set for each operating system to be supported. In a world of specialized applications, such as high performance computing, where the last ounce of performance is important, such an approach makes a great deal of sense – the range of applications in question is fairly well defined and the variety of systems on which such applications are deployed is similarly constrained. But in a world of more broadly distributed applications and operating systems it may be prohibitive for the application or hardware developer and the profusion of application interfaces may be an obstacle inhibiting an IT manager from adopting a given application or technology.

In more broadly deployed systems, an application accesses system services

via a set of APIs often supplied as part of the operating system.

InfiniBand is an industry-standard interconnect and is not tied to any given operating system or server platform. For this reason, it does not specify any APIs directly. Instead, the InfiniBand verbs fully specify the required behaviors of the APIs. In effect, the verbs represent a semantic definition of the application interface by specifying the methods and mechanisms that an application uses to control the network and to access its services. Since the verbs specify a set of required behaviors, they are sufficient to support the development of OS-specific APIs and thus ensure interoperability between an application executing, for example, on a Windows platform which is communicating with an application executing on a Linux platform. The verbs ensure cross-platform and cross-vendor interoperability. APIs can be tested both for conformance to the verbs specification and for interoperability allowing different vendors of InfiniBand solutions to assure their customers that the devices and software will operate in conformance to the specification and will interoperate with solutions provided by other vendors.

However, a "semantic description of a required behavior" is of little actual use without an accompanying set of APIs to implement the required behavior. Fortunately, other organizations do provide APIs, even if the InfiniBand Trade Association does not. For example, the OpenFabrics Alliance and other commercial providers supply a complete set of software stacks which are both necessary and sufficient to deploy an InfiniBand-based system. The software stack available from OFA, as the name suggests, is open source and freely downloadable under both BSD and GNU licenses. The OFA is broadly supported by providers of InfiniBand hardware, by major operating system suppliers and suppliers of middleware and by the Linux open source community. All of these organizations, and many others, contribute open source code to the OFA repository where it undergoes testing and refinement and is made generally available to the community at large.

But even supplying APIs is not, by itself, sufficient to deploy InfiniBand. Broadly speaking, the software elements required to implement InfiniBand fall into three major areas; there is the set of Upper Layer Protocols (ULPs) and associated libraries, there is a set of mid-layer functions that are helpful in configuring and managing the underlying InfiniBand fabric and that provide needed services to the upper layer protocols, and there is a set of hardware specific device drivers. Some of the mid-layer functions are described in a little more detail in the chapter describing the InfiniBand Management Architecture. All of these components are freely available through open source for easy download, installation and use. They can be downloaded directly from the OpenFabrics Alliance, or, in the case of Linux, they can be acquired as part of the standard distributions provided by a number of Linux distributions. The complete set of software components provided by the OpenFabrics Alliance is formally known

as the Open Fabrics Enterprise Distribution – OFED.

There is an important point associated with the ULPs that bears further mention. This chapter began by describing InfiniBand as a message transport service. As described above, in some circumstances it is entirely practical to design an application to access the message transport service directly by coding the application to conform to the InfiniBand verbs. Purpose-built HPC applications, for example, often do this in order to gain the full measure of performance. But doing so requires the application, and the application developer to be RDMA aware. For example, the application would need to understand the sequence of verbs to be used to create a queue pair, to establish a connection, to send and receive messages over the connection and so forth. However, most commercial applications are designed to take advantage of a standard networking or storage subsystem by leveraging existing, industry standard APIs. But that does not mean that such an application cannot benefit from the performance and flexibility advantages provided by InfiniBand. The set of ULPs provided through OFED allows an existing application to take easy advantage of InfiniBand. Each ULP can be thought of as having two interfaces; an upward-facing interface presented to the application, and a downward-facing interface connecting to the underlying InfiniBand messaging service via the queue pairs defined in the software transport interface. At its upward-facing interface, the ULP presents a standard interface easily recognizable by the application. For example, the SCSI RDMA Protocol ULP (SRP) presents a standard SCSI class driver interface to the file system above it. By providing the advantages of an RDMA network, the ULP allows a SCSI file system to reap the full benefits of InfiniBand. A rich set of ULPs are provided by the Open Fabrics Alliance including:

- **SDP – Sockets Direct Protocol.** This ULP allows a sockets application to take advantage of an InfiniBand network with no change to the application.
- **SRP – SCSI RDMA Protocol.** This allows a SCSI file system to directly connect to a remote block storage chassis using RDMA semantics. Again, there is no impact to the file system itself.
- **iSER – iSCSI Extensions for RDMA.** iSCSI is a protocol allowing a block storage file system to access a block storage device over a generic network. iSER allows the user to operate the iSCSI protocol over an RDMA capable network.
- **IPoIB – IP over InfiniBand.** This important part of the suite of ULPs allows an application hosted in, for example, an InfiniBand-based network to communicate with other sources outside the InfiniBand network using standard IP semantics. Although often used to transport TCP/IP over an InfiniBand network, the IPoIB ULP can be used to transport any of the suite of IP protocols including UDP, SCTP and others.

- **NFS-RDMA – Network File System over RDMA.** NFS is a well-known and widely-deployed file system providing file level I/O (as opposed to block level I/O) over a conventional TCP/IP network. This enables easy file sharing. NFS-RDMA extends the protocol and enables it to take full advantage of the high bandwidth and parallelism provided naturally by InfiniBand.
- **Lustre support –** Lustre is a parallel file system enabling, for example, a set of clients hosted on a number of servers to access the data store in parallel. It does this by taking advantage of InfiniBand's Channel I/O architecture, allowing each client to establish an independent, protected channel between itself and the Lustre Metadata Servers (MDS) and associated Object Storage Servers and Targets (OSS, OST).
- **RDS – Reliable Datagram Sockets** offers a Berkeley sockets API allowing messages to be sent to multiple destinations from a single socket. This ULP, originally developed by Oracle, is ideally designed to allow database systems to take full advantage of the parallelism and low latency characteristics of InfiniBand.
- **MPI –** The MPI ULP for HPC clusters provides full support for MPI function calls.

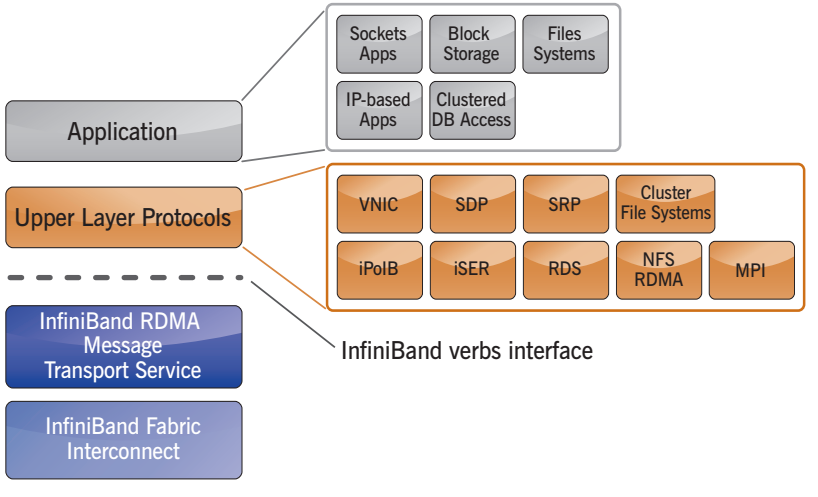


Figure 9.

There is one more important corollary to this. As described above, a traditional API provides access to a particular kind of service, such as a network service or a storage service. These services are usually provided over a specific type of network or interconnect. The sockets API, for example, gives an application access to a network service and is closely coupled with an underlying TCP/IP network. Similarly, a block level file system is closely coupled with a

purpose-built storage interconnect such as Fibre Channel. This is why data center networks today are commonly thought of as consisting of as many as three separate interconnects – one for networking, one for storage and possibly a third for IPC. Each type of interconnect consists of its own physical plant (cables, switches, NICs or HBAs and so on), its own protocol stack, and most importantly, its own set of application level APIs.

This is where the InfiniBand Architecture is different: The set of ULPs provide a series of interfaces that an application can use to access a particular kind of service (storage service, networking service, IPC service or others). However, each of those services is provided using a *single underlying network*. At their lower interfaces, the ULPs are bound to a single InfiniBand network. There is no need for multiple underlying networks coupled to each type of application interface. In effect, the set of ULPs gives an application access to a single unified fabric, completely hiding the details associated with a unified fabric, while avoiding the need for multiple data center networks.

Chapter 5 – InfiniBand Architecture and Features

Chapter 1 provided a high-level overview of the key concepts underpinning InfiniBand. Some of these concepts include channel I/O (described as a connection between two virtual address spaces), a channel endpoint called a queue pair (QP), and direct application level access to a QP via virtual to physical memory address mapping. A detailed description of the architecture and its accompanying features is the province of an entire multi-volume book and is well outside the scope of this minibook. This chapter is designed to deepen your understanding of the architecture with an eye toward understanding how InfiniBand's key concepts are achieved and how they are applied.

Volume 1 of the InfiniBand Architecture specification covers four main topic areas. It specifies the hardware architecture (roughly analogous to the link, network and transport layers of the OSI reference model); it specifies the software transport interface comprising the methods an application uses to access the InfiniBand messaging service, it specifies a complete management architecture, and it specifies the required architectural characteristics of InfiniBand devices (HCAs, TCAs, switches and routers). Volume 2 of the specification is devoted to the physical layers and includes the specifications necessary to design InfiniBand cables and connectors and the low level electrical characteristics of all InfiniBand devices.

In this chapter we focus our attention on developing a broad understanding of how the architecture works. The software transport interface was discussed in the last chapter entitled "Designing with InfiniBand." Also discussed in that chapter was a brief overview of the software architecture and a discussion of the upper layer protocols (ULPs) useful to an application in accessing InfiniBand's message transport service. This chapter lightly covers InfiniBand's transport layer, describes some of the key features of the InfiniBand link layer and introduces the InfiniBand Management Architecture. After reading this chapter you should have a basic understanding of the network services provided by InfiniBand, the key technological features that enable those services, and the completeness and deployability of the architecture.

Address Translation

Perhaps more than any others, the two major concepts driving InfiniBand are:

1. Direct application level access to the InfiniBand messaging service
2. The use of the message service to transfer messages directly applications residing in disjoint virtual address spaces

These are the key concepts that give InfiniBand the characteristics that are most readily apparent to an application.

These (related) features are enabled through address translation mechanisms. Hence, address translation is one of the key technologies that underpin InfiniBand's features. Note that the address translation being described here is the virtual-to-physical address translation that occurs within a server's virtual memory system; it is not the same as one would find in Network Address Translation (NAT).

The InfiniBand Architecture is rooted in concepts expressed as part of the Virtual Interface Architecture (VIA - http://en.wikipedia.org/wiki/Virtual_Interface_Architecture). A key element of VIA, and hence of InfiniBand, is the way that a "virtual interface" is created between the application and the underlying messaging service. The notion of a queue pair (QP) as a logical representation of a channel endpoint was introduced earlier. By using address translation, a QP is mapped directly into the application's virtual address space, thus giving the application direct access to an I/O channel.

The physical I/O channel is actually created by the underlying HCA. The QP, which appears to the application as the logical endpoint of that channel, serves as the physical connection between the application and the underlying InfiniBand HCA, thus a QP can be thought of as an interface to a physical InfiniBand HCA. This is the notion of a "virtual interface;" the QP is mapped into the application's virtual address space and provides an interface to the underlying HCA. The HCA in turn creates the physical communication channel. So when an application posts a work request (WR) to a Send Queue, it is actually exercising control over the physical HCA. The HCA, in turn, carries out the requested operation, such as transmitting the requested message to the appropriate remote destination.

Since an HCA occupies a physical address range in the PCI Express address map, a virtual-to-physical (and vice versa) address translation is required. The application requests the necessary address translations through a memory registration process after which the required virtual to physical address translations are conducted by the HCA using address translation tables. Because the operating system retains ownership of the address translation tables, it continues to ensure the isolation and protection of the virtual address space allocated to each application, even though it does not directly participate in the interactions between an application and an HCA. This provides the same

(or greater) level of address protection and isolation that is found in a standard PCI Express I/O device and ensures that only the registered application can gain access to a given HCA QP. Since an HCA can support many thousands of QPs (up to 2**24 per HCA) and since each QP can be registered to a different application, it is obvious that a single HCA can provide service to a large number of applications simultaneously. Looked at from the application's perspective, the HCA appears to the application to have been virtualized; it is completely unaware that other applications are using services provided by the same HCA. So by providing access to the InfiniBand hardware via a QP which is mapped into the application's virtual address space, the HCA is effectively virtualized; each application is given one or more "virtual interfaces" to the same shared hardware.

The second critical form of address translation enables an application residing in its own virtual address space to exchange messages directly with an application residing in a separate virtual address space. In most cases, the second application ("remote application") is physically hosted by an entirely separate server.

The InfiniBand transport protocol enables this sort of "cross-server" direct communication by providing for the exchange of a set of virtual addresses and keys between a local application and a remote application. This combination of a virtual address and a key, which is provided by the remote (responding) application, is used by the local (initiating) application to perform an RDMA Read or RDMA Write operation directly into or out of the remote application's virtual buffer. The remote application, at the time it passes the virtual addresses and keys to the local application, is effectively passing control of that buffer to the local application, which returns control of the buffer to the remote application after the data transfer is completed. This transfer of control of the remote application's virtual buffer via the passing of a virtual address and key combination ensures that no third party, such as another application, can intrude on the channel connecting the local and remote applications. Channel I/O delivers private, protected channels for the exclusive use of the applications connected by that channel.

The InfiniBand Transport

When an application places a work request (WR) on one of its QPs, it is actually requesting that the InfiniBand transport perform a service on its behalf. In addition to the basic message transport service we have been describing, there is also a number of support services needed to give the application control over the messaging service. We won't describe the support services, such as management and configuration services, except in passing in the section below on the Management Architecture in order to make you aware that such services exist. Our focus in this minibook is instead on delivering a conceptual

understanding of the overall InfiniBand Architecture. In this section, we focus our attention on the message transport services which are at the heart of InfiniBand's ability to facilitate message passing between applications. The provided message transport services include:

- A reliable and an unreliable SEND/RECEIVE service, conceptually similar to the reliable and unreliable services provided on an IP network by TCP (reliable) and UDP (unreliable)
- Two unique forms of an RDMA Read and RDMA write service; more on these momentarily
- Several special purpose services called Atomic Operations
- Multicast services

Despite their importance in some applications, we won't describe either Atomics or Multicast to any degree here.

So although the InfiniBand transport layer occupies approximately the same position in the well-known OSI network stack reference model compared to TCP and UDP, it is actually a superset of the corresponding IP-based transports because it provides a broader range of services.

Like any transport, the InfiniBand transport ensures that packets are transmitted correctly. If one of the reliable services is being used, it further ensures correct delivery of the message to the application, exactly once and in order, in the absence of errors, and it returns a delivery notification to the sender of the message. If an error occurs the transport's reliability protocols are invoked to recover from the error. If recovery is not possible the transport ensures that a higher authority such as the application itself is notified; these are much the same reliability guarantees provided by any reliable transport such as TCP. "Delivery" means that the receiving application is notified of the arrival of the message; with InfiniBand, an application is not disturbed until such time as the complete message has been placed in the appropriate buffer in the application's virtual address space.

However there are two important and related factors to distinguish between InfiniBand's reliable SEND/RECEIVE service and TCP. One factor is that InfiniBand is "message oriented" versus TCP's byte stream orientation. This is important because it vastly simplifies the application's interactions with the network stack, because it dramatically simplifies the hardware and software design and because it simplifies the transport layer's reliability protocols. The second factor is InfiniBand's ability to place an inbound message directly into an application's virtual buffer space without the need for receiving an inbound byte stream into an intermediate "anonymous buffer pool" and then copying it into the application's destination buffer as is required by TCP. This is the feature sometimes described as "zero copy" and is provided courtesy of InfiniBand's built in address translation mechanisms.

A SEND/RECEIVE operation works as follows. A remote (receiving) application uses a Post Receive Request verb to build one or more WRs (work requests) which are placed on a receive queue. Each RECEIVE WR represents a receive buffer in the application's virtual address space. A local (sending) application uses a Post Send Request verb to place one or more WRs on a send queue. Each WR on the send queue represents a message that the applications wishes to be sent to the application at the other end of the channel; a SEND operation targets a buffer represented by a WR on the receive queue at the opposite end of the channel. The sending application does not know the virtual address of the buffer to which the SEND is targeted; instead the SEND is targeted to the receiver QP's receive queue, which contains a pointer to the application's buffer. This rather complex sounding operation is sometimes referred to as a "Channel Semantic." SEND/RECEIVE operations are frequently used, for example to transfer short control messages.

Beyond the channel semantic operations (SEND/RECEIVE), InfiniBand provides a unique service called RDMA. There is no equivalent transport service available on a TCP/IP network. An RDMA operation is sometimes also referred to as a "Memory semantic." This is because the local application is able to directly read or write a virtual memory location (e.g. a buffer) provided by the remote application. Before the local application can execute an RDMA READ or WRITE, it must be in possession of a key and the virtual address of the target buffer provided by the remote application. The actual manipulation of the remote application's virtual buffer is performed by its HCA, using the virtual address and key provided as part of the RDMA operation launched by the local application. RDMA operations are often used for bulk data transfers.

Consider the example of a storage access. Assume that a file system (which we will call the local application), wishes to write a block of data to a block storage device (which we will call the remote application). Assume that an appropriate "channel" (i.e. association between the QPs at the local and remote application ends) has already been created. To initiate the block write, the file system places the data to be written into a data buffer in its local virtual address space. The data buffer had been previously registered with the HCA, which in turn provided a "key" to the file system. Next, the file system constructs a SEND operation. The SEND operation contains a protocol specific command (e.g. SCSI block write), an extent and a virtual address and key. This virtual address and key together represent the buffer at the file system end containing the data to be written. In effect, the file system has passed control of that buffer to the storage device. At the same time, the file system also places a WR on its receive queue, because it knows that the storage device will soon be returning ending status.

The storage device, in turn, uses the virtual address and key to construct an RDMA Read operation in order to read the data from the file system's virtual

data buffer. Once the RDMA Read is complete and the data has been written to disk, the storage device uses a SEND operation to send ending status back to the file system. The receipt of the ending status message by the file system completes the operation.

InfiniBand Link Layer Considerations

InfiniBand practices link layer flow control, which is to say that the hardware dynamically tracks receive buffer usage and never transmits data unless there is space for it to land in. Maintaining perfect knowledge of these states allows InfiniBand to be lossless because flow control happens before data transmission, preempting buffer overflows.

In contrast, TCP/IP uses lossy flow control, in which data is speculatively transmitted without any certainty as to whether downstream elements can accommodate it. If a switch, router or target server is overloaded, data is simply dropped, triggering an elaborate sequence of end-to-end messaging and speed adjustments to recover from the data loss. TCP/IP's flow control happens after data has been dropped, to recover from the inevitable buffer overflows. Indeed, TCP/IP's packet dropping behavior is part of the signaling mechanism it uses to indicate the state of downstream buffers.

Lossless flow control endows InfiniBand with deterministic traffic flows, leading to very efficient use of bandwidth within the data center. This efficient use of bandwidth and deterministic behavior are two of the features that drove the development of InfiniBand's transport protocols.

It turns out that those same "no drop" properties, combined with its congestion windowing scheme also make InfiniBand highly suitable for very long range communication between data centers when used with suitable InfiniBand range extender devices. Range extenders stretch the lossless traffic control semantics over extremely long links encapsulated within standard Wide Area Network (WAN) transports such as 10GbEthernet or SONET.

Management and Services

All the services described above and provided by the InfiniBand transport require an underlying infrastructure consisting of HCAs, switches and routers and cables. Managing this underlying network such that a channel between two applications can be created requires a number of pieces. This management software includes mechanisms for discovering and configuring the switches and HCAs (or TCAs) comprising the fabric, providing routing information between one application and another, establishing channels between applications and so on. The InfiniBand Architecture describes and specifies a complete set of management services and facilities to fully support applications as they use the fabric to communicate. There are implementations of all these services readily available in the marketplace from commercial vendors. The Open Fab-

rics Alliance also provides a complete and robust set of management services and tools via open source. These are freely available for download for Linux or Windows systems, or they can be acquired as complete packages through many commercial Linux distributors. The set of management classes specified by InfiniBand includes:

- **Subnet Management (SM) and Subnet Administration (SA).** This class of managers works together to discover, initialize and maintain an InfiniBand fabric. The SA provides a centralized repository that other management entities can use to obtain information about the fabric configuration and operation.
- **Communication management.** This class is responsible for associating pairs of QPs together in order to create a communication channel connecting two applications together.
- **Performance management.** This class specifies a set of facilities for tracking important performance characteristics of a fabric.
- **Device management.** This class specifies the means for determining the kind and locations of various types of devices on a fabric.
- **Baseboard management.** This class specifies in-band mechanisms by which a system management function can access the low level management features typically included as part of a server.
- **SNMP tunneling.** This class provides mechanisms to tunnel SNMP operations in-band through the InfiniBand fabric.
- **Vendor-specific class.** This class is provided to support management communications that are vendor unique.
- **Application specific classes.** These classes are provided to support application unique management communication needs outside the scope of the InfiniBand Architecture.

The point behind describing this litany of management classes is to drive home that point that the InfiniBand Architecture is completely defined in every important respect and fully deployable based on readily available open source software. Consistent with the notion of open source, there is a vibrant community of vendors offering support and assistance.

InfiniBand Management Architecture

We have already highlighted a number of significant differences between a TCP/IP/Ethernet network and an InfiniBand network. Some of those key differences included InfiniBand's message oriented transport, its support for direct user level access to the network, its ability to support direct communications between two applications and others. There is one other significant difference in management architecture.

The InfiniBand Architecture specifies a centralized management scheme, the major elements, or management classes, of which were described in the previous session. Such a centralized view of the complete fabric makes it simple to establish efficient communications between applications, it makes it possible to select fault tolerant redundant paths through the fabric, it means that routing paths can be discovered very quickly, and it provides an easily accessible database (the subnet administrator) providing a central repository for a great deal of information about the topology of the network. The notion of a centralized management scheme should not be taken to imply a single point of failure. The InfiniBand management architecture goes to some lengths to provide a mechanism for failing over a failed subnet manager to a series of alternate managers.

Quite by contrast, each switching element in an Ethernet fabric is autonomous and makes independent decisions about packet forwarding. There are, naturally, several advantages to such an approach; it is inherently fault tolerant since each switch is responsible for discovering its neighbors, and it may not exhibit a bottleneck in the process of discovery and configuration. But at the same time, there may be several pitfalls associated with a distributed management approach: it may take longer for the fabric to converge on a set of possible routes through the fabric; there is no practical way to identify alternate routes through the fabric; the eventual path chosen through the fabric may or may not be optimal; and it may make it hard to perform certain specialized functions that depend on a “bird’s eye view” of the entire fabric.

At the heart of InfiniBand’s centralized management architecture is an entity known as the Subnet Manager (SM) and an associated repository of information concerning the physical topology of the system. The repository is known as the Subnet Administrator (SA). Together, these entities provide an unprecedented level of performance in control in discovering and configuring the network and provide for advanced features such as support for automatic path failover and the ability to avoid single points of failure. As with the other application components comprising a complete InfiniBand system, these elements are made available via open source by the OpenFabrics Alliance.

Chapter 6 – Achieving an Interoperable Solution

As with most modern computer technologies, the InfiniBand Architecture specifies a rich set of hardware devices, drivers, applications, software libraries and APIs. Making sure it all works together seamlessly is a non-trivial task. Recognizing that software and hardware must interoperate, an interesting combination of strategies have emerged with the single goal of providing an unprecedented level of assurance to adopters that the overall deployed solution will operate correctly, that the required elements will interoperate, and that broad support will be available.

An Interoperable Software Solution

In many hardware-based systems, the software necessary to the correct operation of the system has traditionally been viewed as either a necessary but burdensome cost to be born by the hardware or system vendor, or as an opportunity for vendor differentiation. Recognizing its importance to the widespread adoption of the InfiniBand technology, the members of the InfiniBand community took an unusual step; the OpenFabrics Alliance (www.OpenFabrics.org) was formed to foster the development of a broadly available set of software necessary to the adoption and deployment of the InfiniBand Architecture. The available software includes the Upper Level Protocols, APIs, Applications and Libraries need to instantiate a complete InfiniBand solution, including both application level access to the messaging service provided, drivers for specific hardware components as well as a complete fabric management solution. Since then, the scope of the OpenFabrics Alliance has been expanded to include software support for alternative RDMA-based solutions. But each of those solutions has in common the verbs defined in the InfiniBand Architecture specification. The verbs provide a semantic specification of the required behaviors of the underlying transport.

Consistent with the values espoused by the open source community, software (including libraries and APIs) developed for use with InfiniBand networks is made freely available under a dual license scheme. The breadth of the participating community helps ensure that each piece of submitted code undergoes a broad review which contributes greatly the generally high quality of the software necessary to InfiniBand's correct operation.

In addition to the intrinsic nature of open source software, the OpenFabrics Alliance operates an extensive compliance and interoperability testing focused on ensuring that software comprising the open source repository. Testing is focused on the Upper Level Protocols, Application programs and Application Programming Interfaces provided as part of the repository, and naturally includes thorough testing of switches, routers, channel adapters and systems implementing the InfiniBand Architecture.

Ensuring Hardware Interoperability

The InfiniBand ecosystem contains a rich field of vendors providing an array of components, adapters and switches and cables. Therefore, deploying an InfiniBand solution will usually entail installing equipment and associated software provided by more than one vendor. From its inception, the member companies comprising the InfiniBand Trade Association recognized the importance of assuring correct operation of any given vendor's offerings and established the Compliance and Interoperability Working Group (CIWG). As its name suggests, the CIWG was chartered both with ensuring that any given component can be tested to ensure its compliance with the InfiniBand Architecture specification and with ensuring that products from different vendors would interoperate correctly. This rich and mature program remains in practice today with a regular series of 'plug fests' supported by a marketing program known and the InfiniBand Integrators' List. The Integrators' List (http://members.infinibandta.org/kwspub/Integrators_List_Brochure1.pdf) is a labeling program administered by the InfiniBand Trade Association designed to provide adopters of InfiniBand products with a clear assurance that the products they purchase have been properly tested by the IBTA.

The work of the CIWG actually began before the first page of the InfiniBand specification had been published. The actual text of the specification contains literally thousands of 'compliance statements' which are formal declarations of required or optional behavior that any given component must exhibit. This set of compliance statements is the based against which any device can be tested to verify that it exhibits the behaviors required by the specification; as such, the compliance tests are the basis on which an extensive set of compliance tests have been created over time.

The IBTA holds Plugfests, typically twice per year, at which both devices and interconnecting cables are tested for conformance to the specification. Run by the CIWG, Plugfests are normally held at the University of New Hampshire's Interoperability Lab (IOL). Suppliers submit their products for testing by IOL personnel under the supervision of the CIWG. The test suites performed at the Plugfest consists of two basic types: compliance and interoperability.

To test InfiniBand devices, such as HCAs and switches, testing is done using a "golden" set of cables to ensure that a link can be established. This test

checks the devices for conformance to the link training protocol and ability to negotiate link speed with another device. At some Plugfests transmitter physical layer testing is also performed. If the device passes all the relevant tests, it is added to the Integrators' List indicating conformance.

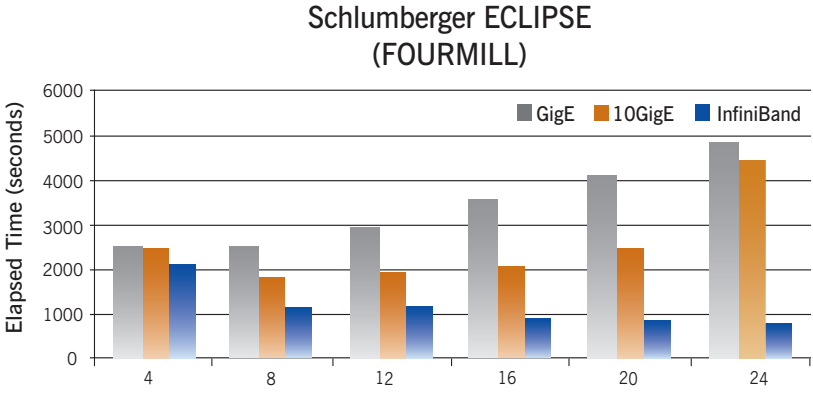
Cable compliance tests in turn are done in two phases. The first is a high-level screen test that is done to ensure that an error-free link can be established using the previously-tested devices and the cable under test. If a successful link can be established, the cable is then tested for a number of physical performance parameters called out in the IBTA specification. The tests are dependent upon the type of cable, but typically include insertion and return loss, crosstalk, eye opening, skew, and jitter. The new Waveform Dispersion Penalty (WDP) technique used by a number of other industry standards is also being explored. If all requirements of the specification are met, the cable is added to the Integrators' List indicating conformance.

Chapter 7 – InfiniBand Performance Capabilities and Examples

By providing low latency, high bandwidth and extremely low CPU overhead with very high price/performance ratios, InfiniBand has become the most deployed high-speed interconnect, replacing proprietary or low-performance solutions. The InfiniBand Architecture is designed to provide the needed scalability for tens of thousands of nodes and multiple CPU cores per server platform and to provide efficient utilization of compute processing resources. Currently, InfiniBand provides up to 40 Gb/s server to server and 120Gb/s switch to switch throughput. This high-performance bandwidth is matched with ultra-low application latencies of 1μsec, and switch latencies of 100ns to150ns, and these performance numbers are delivered at nearly zero cost in terms of CPU utilization. These low latencies and high bandwidths, coupled with the savings in CPU cycles and in memory bandwidth enable efficient scale out of compute systems.

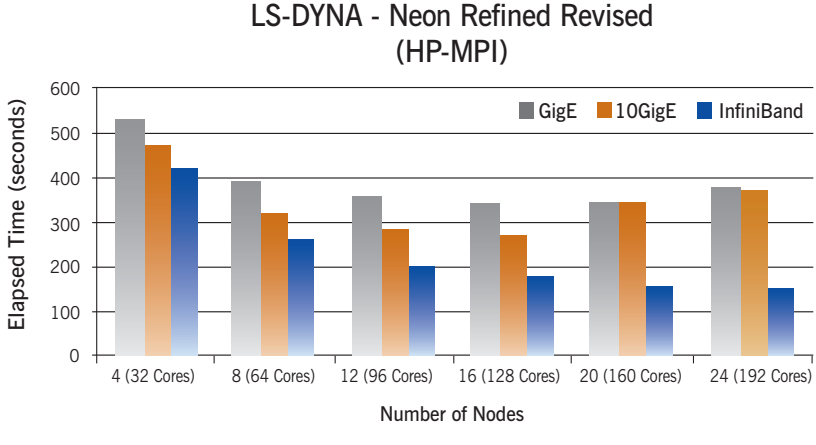
To really understand the significance of these performance capabilities, it is helpful to review real application performance results. The following application cases are taken from the HPC Advisory Council (http://www.hpcadvisory-council.com).

The first example is Eclipse. Eclipse is a commercial product of Schlumberger and is an oil and gas reservoir simulation application. It is extremely popular in the oil and gas industry. The graph below shows the application performance for GigE, 10GigE, and InfiniBand in elapsed time (seconds) as the number of nodes is varied. In every server node size, InfiniBand delivers higher performance. The chart also shows how InfiniBand's latency, CPU utilization and bandwidth characteristics produce excellent scaling as the number of nodes is varied.



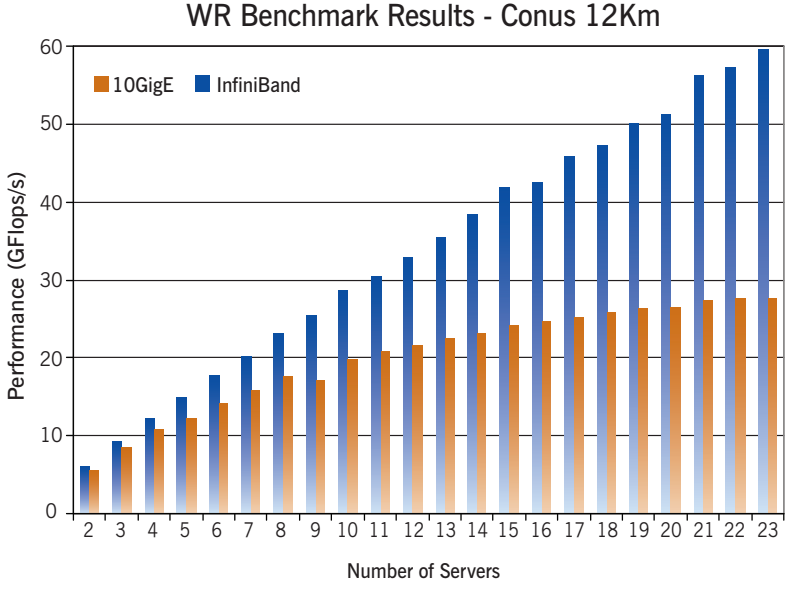
The second example is LS-DYNA, one of the premier impact analysis applications. It is used in many industries; for example, aerospace companies use it to examine the impact of weapons on a fighter or the impact of a bird on wind screens. The car industry uses it to test car impact. Proctor & Gamble uses it to design detergent containers so that they don't break open if dropped at home or in the store.

The figure below presents the performance results of a Neon Refined Revised benchmark from LSTC. The results show the superior capability of InfiniBand to deliver higher performance, or faster simulation run times up to 60 percent faster than Ethernet. As in the previous example, the performance gap improves with the system size.

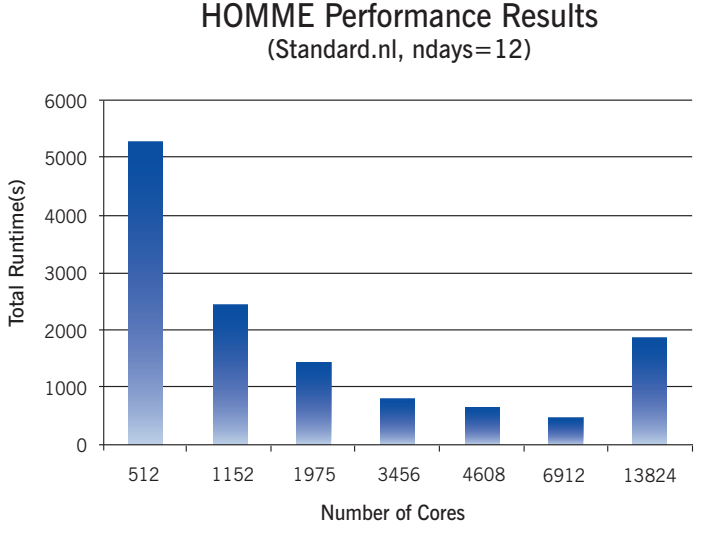


The third example is WRF, a very common weather modeling application. It is also part of the SpecMPI benchmark suite. The benchmark results shown in the graph below are for the 12km CONUS Case which is a 48-hour, 12km resolution case over the Continental U.S. (CONUS) domain October 24,

2001, using the Eulerian Mass (EM) dynamics. The computational cost for this domain is about 28.5 billion floating point operations per average time step (72 seconds). To measure cost over a fully spun-up (all moisture fields and processes fully activated) forecast interval, the test is run over 27 hours; the benchmark period occurs during the last three hours (hours 25-27) starting from a restart file taken at the end of hour 24. When using InfiniBand as the cluster interconnect, WRF showed linear scalability, i.e. with the addition of more server nodes, the total performance of WRF increased accordingly. When using Ethernet as the cluster interconnect, a degradation was measured in the WRF capabilities in every cluster size as compared to InfiniBand. InfiniBand outperformed Ethernet, achieving from 8 percent higher performance with a 2-node cluster to 115 percent higher performance with a 24-node cluster. Moreover, Gigabit Ethernet failed to show linear scalability and it limited performance gains beyond 20 nodes. A 10-node InfiniBand cluster met the performance level achieved by a 24-node gigabit Ethernet cluster running the particular workload. Therefore using InfiniBand gives the cluster administrator the flexibility to either save energy costs by using 2.4x fewer nodes to achieve a given performance level, or to utilize a larger cluster to increase productivity. Looking at the asymptotic behavior of the gigabit Ethernet based results, one could deduct that adding more nodes to the 24-node Ethernet based configuration would not result in significant increased performance.



The last example is HOMME. HOMME is the High-Order Method Modeling Environment developed by the Computational and Information Systems Laboratory at the National Center for Atmospheric Research (NCAR) in partnership with the Computational Science Center at the University of Colorado at Boulder (CU). The following graph illustrates InfiniBand's capacity to scale out to tens of thousands of cores. The performance results were achieved on the JUROPA (Jülich Research on Petaflop Architectures – http://www.fz-juelich.de/jsc/juropa) system. The results shows scalability of nearly 85 percent at the pick tested system size.



Chapter 8 – Into the Future

To be considered viable, any industry standard must have a clearly discernible path into the future. InfiniBand has consistently led the industry by providing a realistic performance roadmap. It continues to do so by providing a roadmap into the future which is one to three years ahead of competing network technologies. The current performance roadmap can be found on the InfiniBand Trade Association website at *www.infinibandta.org* in the Technology tab on the home page.

The InfiniBand Trade Association continues to actively support the architecture through regular maintenance of the specifications and a measured program of consistent feature improvements and technical advancements. Significant new advancements are delivered through a series of annexes to the base specification.

Coincident with the release of this book, for example, the IBTA has announced the release of Annex A16: RDMA over Converged Ethernet (RoCE). By taking advantage of InfiniBand's layered architecture, the annex makes it possible to layer InfiniBand's RDMA protocols and complete messaging service directly on top of a standard Ethernet layer 2 switched fabric. This makes it entirely practical to harvest the full benefits of InfiniBand's RDMA architecture in markets which are deeply invested in Ethernet technologies.

InfiniBand Trade Association Administration
3855 SW 153rd Drive
Beaverton, OR 97006
503.619.0565
administration@infinibandta.org
www.infinibandta.org