InfiniBand Clustering

Delivering Better Price/Performance than Ethernet

1.0 Introduction

High performance computing clusters typically utilize Clos networks, more commonly known as “Fat Tree” or Constant Bisectional Bandwidth (CBB) networks to construct large node count non-blocking switch configurations. A Clos network is a switch topology in which integrated non-blocking switch elements (crossbars) with a relatively low number of ports are used to build a non-blocking switch topology supporting a much larger number of endpoints.

There are two important aspects of constructing CBB networks: i. topology ii. link level forwarding. Topology defines how the network is physically connected together. From a purely physical perspective, point to point networks are virtually identical and any topology can be constructed using similar corssbar switch elements. Forwarding deals with the way individual packets are actually routed through the physical topology and thus effectively define a logical topology that exists on top of the physical one.

While all point to point technologies can implement the same physical topologies, this is not the case with the logical topology created by different forwarding algorithms. In fact, unlike InfiniBand, bridged Ethernet imposes severe constraints on what types of logical topologies are implementable and typically can not make efficient use of the underlying physical topology. In order to overcome this limitation of layer 2 Ethernet switches, it is necessary to use layer 3 (and higher) IP switches. Using layer 3 and above switches severely impact both, the price and performance of the base Ethernet offering.

2.0 Topology

2.1 Clos, CBB, and Fat Tree Networks

The concept of a CBB or Fat-Tree topology is based on the seminal 1953 work on non-blocking switch networks by Charles Clos[1]. This paper describes how a non-blocking switch network can be constructed using minimal number of basic switch building blocks.

In order to be strictly non-blocking, several key requirements must be met. The most basic requirement is that the cross bar switches which make up the network, must be configured appropriately in order to offer non-blocking forwarding for the particular traffic pattern being switched. This requirement is easily met in the case of the public switched telephone network for which Clos networks were originally designed, since voice conversations are relatively long lived compared to the time required to set up the crossbar switch forwarding tables.

This basic requirement is not met by packet based networks since every packet is routed independently through the network of switches. Nonetheless, for a very large set of typical networking applications, there exist traffic patterns which closely resemble long lived circuit switched connections. Because of this fact, Clos networks have been successfully used in packet based networks, and to first order, provide non-blocking switch configurations. Leiserson[1] was the first to recognize this and extended the concept of a Clos network from telecommunications to high performance computing communications based on “fat tree” networks.

Figure 1 shows both a simple binary tree as well as a fat tree topology. As can be seen in the simple binary tree, the number of links and thus the aggregate bandwidth is reduced by half at each stage of the network. This can result in serious congestion “hot spots”’ towards the root as all traffic funnels into a single link. The fat tree topology remedies this situation by maintaining identical bandwidth at each level of the network. Another way to look at the fat tree is to imagine a line that bi-sects the network into an equal number of endpoints. In a fat tree topology any such bi-section line would cut an equal number of links - hence the name Constant Bi-Sectional Bandwidth. Note that the diagram shows simple switch elements with only a few ports, however the CBB topology is readily extended to higher port count crossbar switch building blocks.

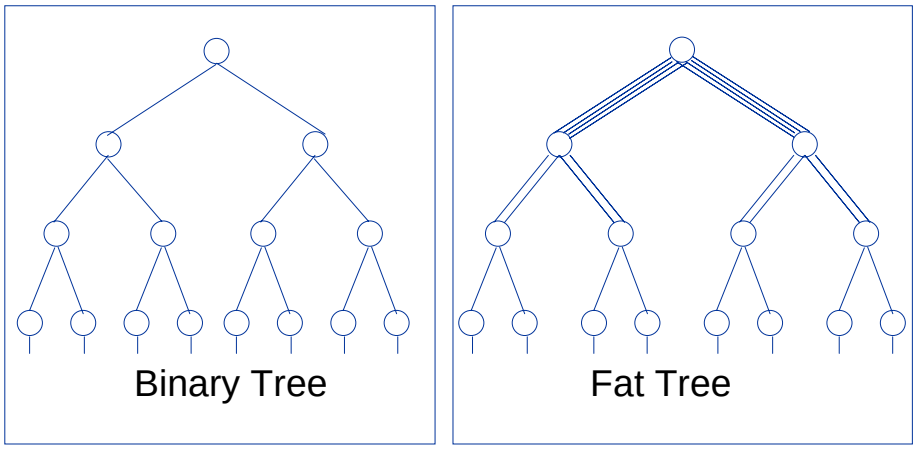

Figure 1. Binary and Fat Tree Topologies

In general, the Clos, CBB, and Fat Tree networks describe identical switch topologies, however, in high performance compute clusters the CBB nomenclature is most common and thus will be used from here on.

So what is the point of building a CBB network? Given a basic switch building block with N ports and the requirement to build a switch network to connect to M end nodes (where M > N), a CBB network defines an efficient topology using multiple levels of basic switch elements providing a non-blocking switch configuration supporting M endpoints. In such multi-level switch architectures, some of the ports of each basic switch element are connected to end nodes, while other ports are connected internally to additional levels of basic switch elements which do not connect to endpoints.

1. C. Clos, *A study of non-blocking switching networks*, Bell System Technical Journal, Vol. 32, 1953, pp. 406-424.
1. Charles E. Leiserson: Fat-Trees: Universal Networks for Hardware-Efficient Supercomputing., IEEE Transactions on Computers, Vol 34, October 1985, pp 892-901

As previously described, the key attribute of the CBB topology is that for any given level of the switch topology, the amount of bandwidth connected to the downstream end nodes is identical to the amount of bandwidth connected to the upstream path used for interconnection. Note that the links are bidirectional so the notion of upstream and downstream describes the direction of the interconnect topology towards or away from the shortest path to an end node, rather than the actual data flow.

2.2 Two Level CBB Network Example

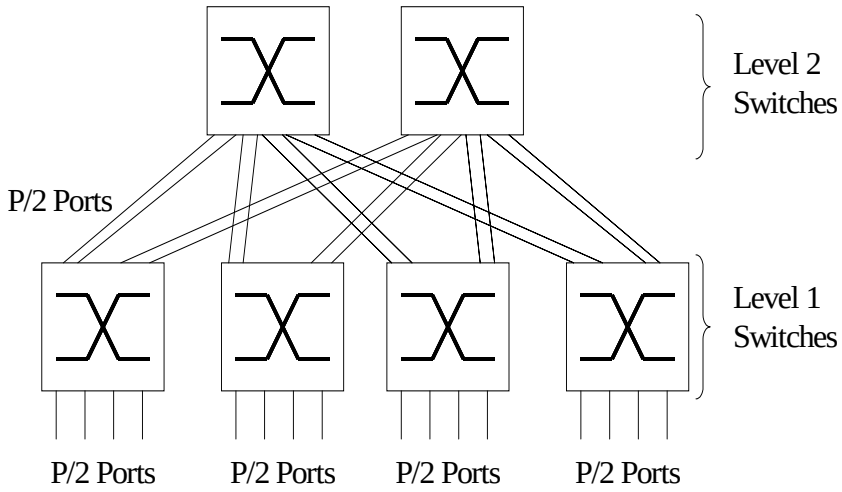

Figure 2. Example of a simple two level CBB switch topology

An example to make this clearer. Consider the two level switch topology shown in Figure 2 constructed with basic switch building blocks having P ports. Suppose there are N1 level-1 switches, each of which has P/2 ports connected downstream to end nodes, and a identical number connected upstream to the second level switch. The total number of end points connected is then: N1 * P/2. Suppose for example each basic switch building block has eight ports and a two level network is constructed with four ports connected both to end points and level 2 switches as shown. There are four level-1 switches so: P=8 and N1=4, hence using an 8 port building block this topology builds a 16 port non-blocking switch. It is possible to connect more total endpoints by increasing the number of level-1 switches and even further by increasing the number of levels in the switch topology. The key to making the configurations non-blocking is to always preserve identical bandwidth (upstream and downstream) between any two levels of the multi-level switch topology.

Closer examination of the example yields the observations that for two level CBB networks, the total number of basic switch element ports used is three times the total number of end nodes being connected. This is so because for each end node being connected there is another level-1 switch port which connects to yet a third port on a second level switch element. Thus the 16 end nodes supported in the example use six separate eight port switches requiring a total of 48 ports.

3.0 Link Level Forwarding, Loops, and Logical Topologies

While point to point networks are nearly identical at the topology (physical) level, there are profound differences at the logical level. Two key issues related to link level forwarding are: i. spanning (the ability of every node to communicate with every other node) and ii. loop breaking. Many different algorithms are available to address these issues, but these algorithms have a profound impact on network performance.

Of the two, providing an algorithm that spans is fairly straightforward, whereas, loop breaking is the more critical issue. Loop breaking is required since the physical network topology is human administered and therefore it can not be guaranteed that the network is physically connected so as to be free of loops. Network loops are problematic since it is possible that a packet forwarded from the output port of a crossbar

switch, may ultimately arrive to an input port of the same crossbar switch. This crossbar element will, of course, forward the packet out from the same output port originally used to transmit it, thereby creating a transmission loop. Under these circumstances and without any other mechanisms to resolve the situation, the packet will flow indefinitely in a loop consuming bandwidth and ultimately creating a fully congested network unable to forward any other traffic. In short, loops are a potentially catastrophic aspect of switched networks that must be algorithmically addressed by any useful forwarding algorithm.

3.1 Ethernet Forwarding and the Spanning Tree Algorithm

Ethernet adopted a simple but elegant algorithm to deal with loops known as the spanning tree algorithm. Every layer 2 Ethernet switch is required to run this algorithm during initialization, and periodically to be able to accommodate changes to the physical network. The algorithm can be briefly explained as follows:

- Every layer 2 switch in the network starts by broadcasting a 2-Tuple consisting of: i. its own unique NodeID and ii. a cost of 0.
- Every layer 2 switch in the network continues to broadcast a 2-Tuple consisting of i. the best (lowest) NodeID received via any port and ii. the best cost (in terms of number of hops) required to reach this best ID. If the switch receives a better {NodeID, cost} it will increment the cost by 1 and start sending this new {NodeID,cost}2-Tuple.
- Ultimately, the switch with the best NodeID is recognized throughout the network as the root node.
- Every switch will determine its own root port which provides the lowest cost to the root node. Ports with ties in the cost metric will be arbitrarily decided by the port number and a single root port chosen. In addition, every switch will also determine ports for which “self” has been identified as the best forwarding bridge. All other ports will be logically turned off (no packets forwarded out these ports).

More or less this is it - elegant and simple at breaking loops but unfortunately not ideal for CBB networks as will be shown.

3.1.1 Problems with Spanning Tree for Implementing CBB Clusters

This description overly simplifies the spanning tree algorithm which also includes enhancements such as switches periodically scanning the network and discarding information to accommodate network changes and device failure. Nonetheless, the description captures the essence of the algorithm and is adequate to understand the algorithm’s limitations for construction of CBB networks for clustering. As an example, let’s apply the spanning tree algorithm to the network as shown. For simplicity assume the unique IDs of the various switches range from 1 to 6 as shown in Figure 3. Clearly, as the spanning tree algorithm progresses, the switch on the upper left with the ID of 1 will emerge as root node.

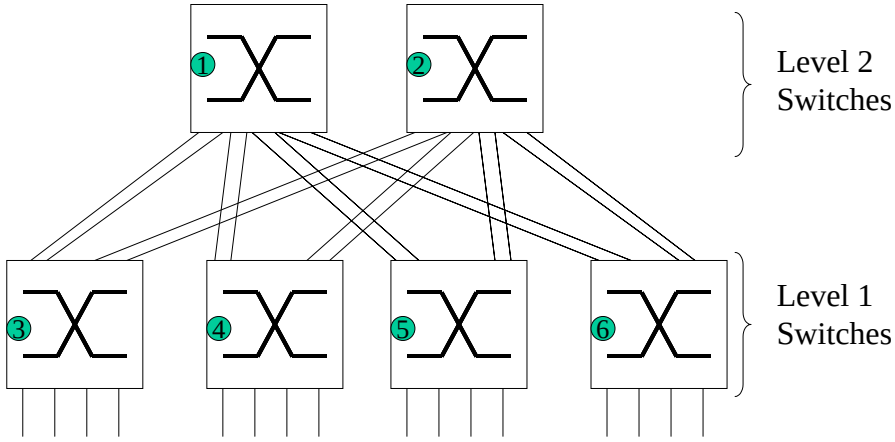

Figure 3. Fat tree topology with switches labeled with arbitrary unique IDs 1-6

Now, since the shortest path to the root node is directly to switch #1, the spanning tree algorithm will disable all level 1 switch ports that connect to the second level switch labeled #2. Furthermore, one of the two ports from each of the level 1 switch connecting to switch #1 will be disabled, leaving only one active port. So the physical CBB network reduces to the logical network as shown in Figure 4 with disabled ports and switches shown with red dotted lines.

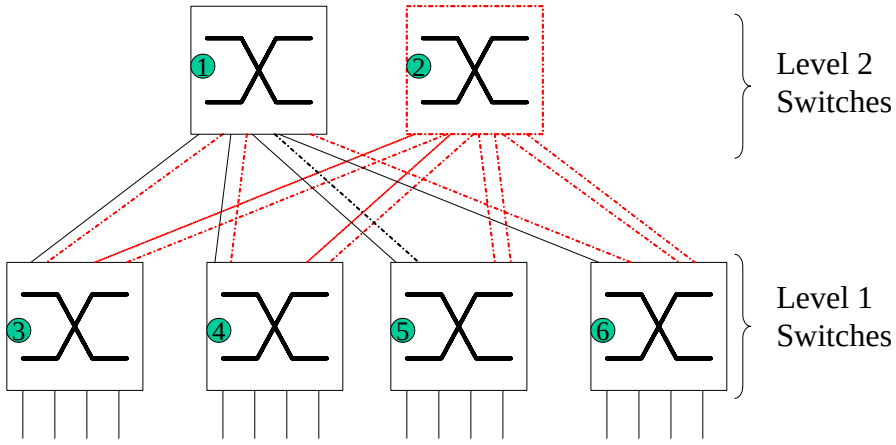

Figure 4. Logical Network After Spanning Tree Algorithm Disables ports to Eliminate Loops (Disabled elements shown with dotted lines)

In essence, switch #2 is logically removed from the network. Furthermore only half of the links available between the level 1 switches and switch #1 are now active. The reason the spanning tree algorithm behaves this way is that its goal is to provide a loop free logical network that spans to every node (i.e. every node in the network is reachable by every other node). Thus, only connectivity is considered and preserving balanced bandwidth is not. Consequently, switch #2 is logically removed as it does not provide any additional spanning connectivity that is not provided by switch #1. Similarly the redundant inter-level links, though allowing for balanced bandwidth connections, are discarded as superfluous by the spanning tree algorithm.

In short, the spanning tree algorithm does not allow the use of layer 2 Ethernet switches to build CBB cluster networks. Instead higher level (layer 3 and above) Ethernet switches must be used. These switches in turn have different routing algorithms available, but again, these are not designed with CBB clustering networks in mind. Furthermore, IP routing requires sophisticated functionality such as longest pre-fix matching, sub-netting, and super-netting. This functionality requires advanced processing which requires store as well as forward, and thus precludes the use of "cut-through" switches. The requirement for store and forward and advanced processing introduces significant switch latency and greatly increases the cost of these switches. Ultimately, the promised "low cost" Ethernet switches in fact become very high priced, high latency, IP routers.

3.2 InfiniBand Forwarding

By contrast, InfiniBand does not dictate the use of the spanning tree algorithm and, in fact, does not mandate any specific algorithm at all. Instead, InfiniBand defines a set of forwarding and loop breaking mechanisms on top of which any destination routed algorithm may be implemented. The choice of algorithm is a function of the Subnet Manager (SM) which is responsible to discover the physical topology and set up the logical topology by configuring the forwarding tables in each switch.

A single master SM is required to manage an InfiniBand subnet which may reside on any node (typically running on a switch or host server). The SM is a software process and thus configuration policies are completely flexible and can implement appropriate high performance forwarding algorithms. In the case of a CBB network, as shown, a weighted, shortest mean path algorithm is typically implemented so that traffic is evenly distributed across the available connectivity and loops are prevented. Using these more sophisticated algorithms, InfiniBand can address the issue of loop breaking, without sacrificing the ability to recognize logical topologies that take full advantage of the physical topologies.

In fact, other even more advanced load balancing and dynamic algorithms are possible which can take advantage of InfiniBand's multi-path, virtual lane, and automatic path migration algorithms. Multi-pathing extends InfiniBand's destination routing capabilities such that traffic flows to a given endpoint can actually traverse different paths through the network. This allows load balancing software to monitor the network, detect hot spots, and choose different routes through the network to avoid traffic and avoid congestion. Similarly, automatic path migration capabilities which are useful to implement failover can also be used by sophisticated routing algorithms.

Furthermore, with InfiniBand, the SM has mechanisms other than port disabling to break potential loops. For example, InfiniBand offers service level to virtual lane mapping functionality which allows packets erroneously received to be dropped without actually completely disabling a port. These more powerful mechanisms allow InfiniBand fabrics to take full advantage of the physical topology and distribute traffic across links to make best use of all available bandwidth.

The user is shielded from all this sophistication as the SM automatically and transparently discovers and initializes the fabric. For users who do want direct control of configuring the logical topology, the SM provides a software interface to allow user programmable algorithms to be implemented. Thus, InfiniBand architecture provides the best combination of automatic provisioning with the flexibility to implement optional user defined algorithms.

At an even lower level, InfiniBand Architecture includes mechanisms that avoid dropped packets and guarantee in-order packet delivery. InfiniBand defines link level, credit based flow control so that packet data is not transmitted unless there is guaranteed buffering at the receiving end of the link. This means that, unlike Ethernet, the InfiniBand link level fabric does not drop packets due to buffer overruns and thus provides guaranteed in-order delivery. By addressing such issues at the lowest possible level, InfiniBand

transport level processing only needs to deal with extremely rare packet corruption issues. This greatly simplifies the requirements and makes transport offload efficient to implement in hardware. The net result is higher performance, lower latency, and reduced CPU utilization at the network and application level.

4.0 Practical Considerations of Cluster Price/Performance

It is certainly interesting to better understand the limitations and impact of various clustering fabrics, however, it is always important to ask whether any of this really matters. In fact, underlying technology issues are not the most important factor, but rather application level performance and the price required to achieve it, together, define the ultimate metric that determines the value of a technology. Ultimately, it is price/performance that determines which technology is the most valuable.

Now with Ethernet technology, there are two options: i. use cheap layer 2 switches and hope poor latency, congestion, dropped and out-of-order packets and other issues don't impact performance, ii. use expensive layer 3 high port count switches.

Let's look at both these options and compare to InfiniBand. For the first option, the November 2003 Top 500 list of the highest performing computers in the world (www.top500.org) provides a ready comparison. The comparison in Figure 5 makes it clear that the clustering technology indeed does make a pronounced difference in achieved performance. Here, the InfiniBand cluster ranked as the #111 computer in the world has one fourth the number of CPUs, yet still outperforms the Gigabit Ethernet based cluster. That's right: ***the InfiniBand cluster with only one fourth the processors outperforms the Gigabit Ethernet cluster***. In this case, the price/performance equation is simple since InfiniBand switch ports, at less than $300 per port, are far more cost effective than buying servers costing several thousand dollars or more.

Clustering Technology	Gigabit Ethernet	InfiniBand
CPU	Xeon	Xeon
CPU Clock Frequency	3.0 Ghz	3.0 Ghz
Number of Processors	1024	256
Performance	1.068 Tflops	1.076 Tflops
Place on Top500 List	112	111

Figure 5. Comparison between Gigabit Ethernet and InfiniBand Clusters from Nov2003 Top500 list of Computers (www.top500.org)

Now, in fact, the poor performance of the Gigabit Ethernet cluster is not solely related to networks problems caused by Ethernet spanning tree algorithm. In reality, InfiniBand has many advantages as a clustering interconnect over Ethernet including 10Gb/sec performance including transport offload in hardware, kernel bypass, and remote direct memory access capabilities (RDMA). In addition, the Ethernet cluster levies a heavy TCP/IP tax on the CPU compared to InfiniBand, which delivers reliable transport connections in hardware with minimal CPU utilization. So it is not just Ethernet inability to make effective use of CBB networks, but rather all of these factors together, that combine to result in the huge performance advantage shown by InfiniBand. Nonetheless, the price/performance comparison is valid and clearly the InfiniBand cluster delivers more compute power per dollar.

So now let's consider the second option - build a CBB network from layer 3 Ethernet switches. Here we'll compare InfiniBand with both 10 Gigabit/s and 1 Gigabit/s Ethernet switches (Figure 6). It is somewhat surprising that though offering only about 1/10th the bandwidth, the Gigabit Ethernet switch actually costs about 40% more per port than the 10Gb/s InfiniBand switches! The 10Gigabit Ethernet switch price per port is actually about 1590% greater than the 10Gb/sec InfiniBand switch price per port. That right - 10GE costs about 16 times as much as 10Gb/s InfiniBand! Thus the so called "cheap" Ethernet story completely breaks down when layer 3 or 10GE switches are used. Furthermore, InfiniBand substantially outperforms both Gigabit and 10Gigabit Ethernet technology for clustered applications because of all of its other attributes.

	Gigabit Ethernet	**10 Gigabit Ethernet**	**InfiniBand 10Gb/s**
Vendor	Extreme Networks	Foundry	Voltaire
Product	Summit 7i	Fes-X	ISR9024
Price	$16,495	$12,500	$8,850
Number of Ports	32	2	24
Price per Port	$515	$6,250	$369

Figure 6. Comparison of InfiniBand and GE / 10GE Switches

5.0 Summary

Constant Bi-Sectional Bandwidth (CBB) or Fat Tree networks have emerged as a key ingredient to deliver non-blocking bandwidth for high performance computing and other large scale compute clusters. In addition to wiring the network to provide a physical CBB network topology, system architects also need to pay close attention to the underlying networking technology. Indeed, the spanning tree algorithm required by Ethernet layer 2 switches is not able to exploit physical CBB fat tree topologies. InfiniBand on the other hand combines automatic configuration and flexibility of the forwarding algorithm, to be able to fully take advantage of the underlying CBB network. The only solution for Ethernet is to move to expensive layer 3 switches. Unfortunately, despite offering only 1/10th the bandwidth, high port count Gigabit Ethernet layer 3 switches are actually more expensive than the equivalent InfiniBand 10Gb/sec switches. Thus from a price performance perspective InfiniBand offers significant advantages over Ethernet. The InfiniBand advantage is due not just to more flexible and efficient forwarding algorithms, but also because of 10Gb/sec performance, transport offload in hardware, kernel bypass and remote direct memory access (RDMA).