Regular Paper

You Don't Really Need Big Fat Switches Anymore — Almost

Satoshi Matsuoka[††,†]

Although commodity cluster computing based on very fast and inexpensive commodity processors are proliferating today, one of the prohibitive factors towards its large-scale deployment is the high cost of the network switching fabric in order to retain properly high bandwidth. We argue that, except for the most demanding applications, appropriate aggregation of inexpensive switches, with collective communication algorithms that utilize the characteristics of such networks, will accommodate a bulk of parallel applications, even those with substantial communication requirements. We present 3 techniques for implementing high-bandwidth collective communications in such a setting, and provide preliminary performance measurements that hint the effectiveness of our proposal. The technique can be extended to interconnect a set of clusters for implementing high-bandwidth Grid interconnect as well as replacing SAN for high-bandwidth I/O.

1. Introduction

Commodity cluster computing based on very fast and inexpensive commodity processors are proliferating today. The most recently announced "The 21_{st} Top 500 List"[6] shows that 29.8% of the systems are clusters, and the MCR Linux Cluster at the Lawrence Livermore National Laboratory, USA, is ranked 3_{rd} on the list with the RMAX of 7634.00. Note that this performances greatly surpasses those of the latest supercomputers with the same or even greater number of processors (2304) and higher RPEAK, for example the Fujitsu PRIMEPOWER HPC2500, newly ranked 7_{th} with 2304 processors.

With increasing size of the clusters, however the prohibiting factors tend not to be individual processor performance, or general scalability in terms of aggregating the number of nodes physically (except for very dense clusters). Rather, the important factors are those of reliability/fault tolerance, and network scalability. The former is being addressed by various work, including our JST CREST "MegaScale" project which aims to address cluster scaling issues in terms of dependability up to 100,000 processors, with various techniques such as fault tolerant MPI[7),9)], and multi-path networking[8)], and low power processors.

The latter problem is typically addressed by merely "throwing money at the problem", i.e., using faster, more expensive network interconnects. The cost issues here are twofold—the cost of individual links to each node and it's associated network interface, versus the cost of the switches. Individual link and interface cost often takes dramatic price dives with commoditization; as an example, consider the proliferation of 1000Base-T Ethernet. Such commoditization in turn may drive down the prices of competing, higher-priced solutions such as the Myrinet[5)] and Infiniband[3)].

On the other hand, large-scale switches tend to remain expensive, even for commodity networks, and especially for high-bandwidth networks. It would be either that 1) large crossbar switches would be required (which would be very expensive due to very large backplane switching speed), or more often for higher-speed networks, 2) staged $O(n \log n)$ full bandwidth network such as the CLOS (fattree) network, would by employed, increasing the number of switches by significant factors when n becomes large. In such cases, the total price of the network fabric may well exceed the aggregated price of the nodes themselves. Indeed, for our Presto III Cluster (512 Athlon MP processors / 256 dual nodes with Myrinet 2K, ranked 86_{th} in the 21_{st} Top 500) in our laboratory, the price of the network occupies the largest cost, and not the nodes themselves. Although the network hardware prices may come down, the situation may not change because the increase in processor performance driven by Moore's law will require higher-performance networks, while for ordinary desktop and multimedia applications such fast networks may not be necessary, slowing down the commodatization of faster network standards such as 10Gbps Ethernet.

The aim of this paper is to demonstrate that, by combining various techniques using inexpensive switches with fast commodity networks as cluster network interconnects, in most practical cases expensive networks, especially those caused by the high expenses in large, high-bandwidth switches, would not often be necessary. That is to say with a combination of

† Global Scientific Information and Computing Center (GSIC)
†† Tokyo Institute of Technology

cheap, inexpensive switches (such as commodity 24-port 1000Base-T Layer 2 switches, whose per port cost is below \$100 as of writing of this paper), our scheme should provide sufficient network interconnect performance except for the most network demanding applications; moreover, in such a case scalability and high machine efficiency would be difficult to achieve in any case. The three techniques used are (1) *Pipeline communication* through a *cyclic ring communicator*, (2) *clique network switch topology* with *small network switches*, and (3) *multipath routing* with *active node forwarding.* Although the work is still very preliminary, we will discuss and/or demonstrate the feasibility of each technique, and discuss the viability of Grid-like, multi-cluster configuration as future HPC platforms based on mainly commodity networking.

2. Network and Application Preliminaries and Assumptions

We will assume that each cluster node ($\text{Node}_1 \ldots \text{Node}_C$, totalling C nodes) has a single network interface of bandwidth L (such as a 1000Base-T interface where L = 1Gbps). Each network switch is assumed to have N network ports of the same speed as the node network interface, where N being significantly smaller number than the number of nodes in a cluster C. and all switches are identical, and numbered in an ordinal fashion (SW_1, SW_2, $\ldots$ SW_M), where $M \geq \lfloor C/N \rfloor$ For example, for $C = 100$ and $N = 24$, then $M \geq 5$. Within each switch we assume full duplex communication, and there is sufficient backplane bandwidth to support full bisection communication load. So in the $N = 24$ case above the switching speed must be at least 48Gbps (full duplex). We also assume that network switches are statically configurable, so that any loops in the network can be resolved by static reconfiguration, but a given network paths are not divisible along different network paths between switches, e.g., suppose Node_1 and Node_2 are connected to SW_1, while Node_3 and Node_4 are connected to SW_2; then, communication between Node_1 and Node_3, as well as that between Node_2 and Node_4, must take the same network paths through the switches.

The latter requirement may seem strong, but usually is the property of most Ethernet switches, although some switches may allow for port-by-port trunked routing; here we assume that the switches are inexpensive such that such high-end features are not supported or hinder performance greatly.

As for applications, for simplicity we assume that they are parallel, SPMD applications that do not share memory on clusters and communicate solely by MPI, although our results are more general. Where possible collective communication is employed, instead of point-to-point communication. This is in order to take advantage of the underlying efficient collective communication algorithm tailored for our systems as described below, specifiable by designating an appropriate MPI communicator. Collective communication can be categorized as follows, in accordance with[4].

- broadcast from one process to all
- gather data from all to one
- scatter data from one to all
- allgather: like a gather, followed by a broadcast of the gather output
- alltoall: like a set of gathers in which each process receives a distinct result
- global reduction operations such as sum, max, min, and user-defined functions
- scan (or prefix) across processes

Given the above assumptions, one may easily conclude that fabricating a large cluster out of such commodity parts would pose significant communication overhead for many applications. Indeed, a naive topology, MPI and/or the user application being unaware of the underlying communication topology, poor inter-switch bandwidth, as well as heavy network contention, may reduce overall network performance by an order of magnitude. For example, if $N - 1$ nodes are each fully connected to SW_1 and SW_2, and there is only a single link between SW_1 and SW_2, then a naive all-to-all communication cost would increase to $O(N^2)$, as the effective bandwidth to half of the nodes reduce to $O(1/N)$. It is thus very important to devise strategies to best exploit the available bandwidth in a strategic fashion so as to come close to having a full bandwidth network as much as possible.

Global communication cost of a collective communication is the time from which the collective communication is initiated on the first node of the cluster, until the entire operation is completed and acknowledged on all the nodes (and thus the barrier is complete). We say that a collective communication algorithm is *node-link-bound* when the global communication cost to send X bytes is largely a function of X and L only, and not dependent on C or N. The node-link-bound property denotes optimal collective communication performance in a sense that performance of the communication will not improve even if a large switch with full bisection bandwidth is employed. We say that a collective communication is *switch-bound* when the global communication cost also depends on N and C. Finally, we note that tree-based col-

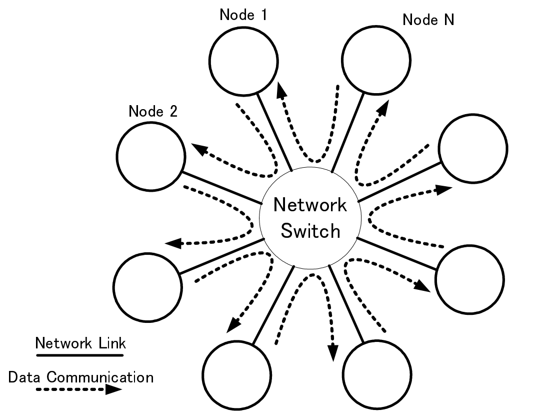

Fig. 1 A ring communication structure with pipelined communication

lective communication algorithms are *not* link-bound since the communication time is dependent on $logN$, but could be almost considered to be so if the tree is very shallow.

3. Pipeline communication through a cyclic ring communicator

We present the first technique, which aims to utilize the maximum bisection bandwidth available in individual switches, while sacrificing latency. Thus the algorithm is appropriate for collective communications involving large data sets, a case where naive communication scheme will result in prohibitive global communication costs (for small message sizes, switch bandwidth is not important and efficiency is largely a matter of latency. Thus, it would be formidable in practice to create a library where collective communication algorithms are altered depending on message size, network topology, etc., by estimating the global communication cost.)

For simplicity, we consider a case with a single switch with N ports ($= N$ nodes). Upon start of a collective communication, a single node (suppose Node_1 = rank 0 in MPI) takes charge of formulating a ring communication structure as depicted in Figure 1, based on the node membership of the communicator. Here, Node_1 communicates with Node_2, Node_2 with Node_3, ... and finally Node_N communicates with Node_1 to complete the ring. Note that even when all the nodes communicate with each other, we obtain full bandwidth on all the rings based on the network switch backplane assumption we made in Section 2.

Let us start with a simple example of broadcast from Node_1 (rank 0). Once the communication ring is formed, the nodes communicate in a *pipeline* fashion, i.e., given Node_I, as the bytes are read from Node_{I-1}, it is stored into the communication buffer as specified by

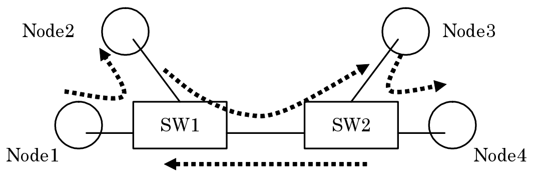

Fig. 2 Extension to Multiple Network Switches

`MPI_Bcast()`, and forwarded immediately to Node_{I+1}. This is effectively the broadcast algorithm as being used by our Dolly+[2)] file data distributor for clusters. Given the latency δ for communication inbetween two nodes, including the overhead for forwarding, it is easy to see that the total time required $\text{TB}(X)$ to send X bytes equals $X/L + N\delta$. Thus if the first term is large enough, then the global communication cost is link-only-bound.

We can generalize this algorithm in two ways. The first is to have multiple switches interconnected by one or more links, as shown in Figure 2. In this case the extension is straightforward—the ring is extended so that there is communication between Node_N of the first switch, and Node_{N+1} of the second switch, and communication between Node_{2N} and Node_1. The only added overhead is the bidirectional latency of the interconnecting link, namely 2δ.

The second extension is combined broadcast, by having multiple broadcast initiation points along the ring, and assumes that the broadcast commences at the same time. This will lead to situation where the communication will have to be buffered along the ring, as multiple data transmission among the ring will directly contend with each other. We can easily see that by delaying the start of each broadcast so that they are performed in succession, we achieve the *additive property*, i.e., When X_1, X_2, ..., X_K bytes are being sent on a combined, then $\text{TB}(X_1, X_2, \ldots, X_K) = \text{TB}(X1) + \text{TB}(X_2) + \ldots + \text{TB}(X_K)$. In fact, we can easily prove the following:

Theorem 1 For any buffering strategy employed for combined broadcast for sending $X_1, \ldots, X_K$ bytes from nodes $Y_1, \ldots Y_K$, the optimal buffering strategy at best yields the additive property, i.e., $\text{TB}(X_1, X_2, \ldots, X_K) = \text{TB}(X_1) + \text{TB}(X_2) + \ldots + \text{TB}(XK)$, and is node-link-bound.

The above results allow other collective communication algorithms to be utilized on the cyclic ring communicator, and be shown to exhibit link-only-bound property. In particular, scatter and gather algorithms can be easily derived from the combined broadcast. For example, for allgather operation, combined broadcast for all the nodes will exactly yield the result, and would be optimal.

Also, where latency becomes excessive such that the node-link-bound property will no longer hold, one could employ a hybrid scheme where we will have a hierarchical ring structure. We will measure the latency and the scalability thereof of the scheme in Section 6.

Still, there are collective communication operations that are not quite fit for the cyclic ring communicator scheme. One is the reduction operation, where communication time can be reduced to $O(X \log(N))$, in which case it can be shown that it is more efficient than cyclic ring communicator where it would be $O(XN)$. The other is all-to-all communication, such as matrix transpose, where again it will not be of help either. In fact, for all-to-all communication, there is no really clever algorithmic tricks one can play to improve on the all-to-all communication; one has to provide a switch structure to facilitate as much bisection bandwidth as possible.

4. Clique network topology with small network switches

As discussed above, for facilitating all-to-all the question boils down to: "How could one build an efficient switch structure that does not lose bisection bandwidth, while staying within the constraints of inexpensive switches as defined in Section 2?" Although there are various ways possible, the proposal here is to create a clique network. Specifically, we structure the network as follows: given a switch with N ports, we connect $N/2$ ports to the nodes, and connect the other $N/2$ ports to $N/2$ switches. Figure 3 shows the network configuration for $N = 8$; note that we essentially create a clique involving $N/2 + 1$ switches as clique nodes. The number of nodes interconnected by the network is thus $N(N + 2)/4$, so for $N = 8$, it interconnects 20 nodes, for $N = 16$ it is 72 nodes, for $N = 24$ it is 156 nodes, for $N = 32$ it is 272 nodes, for $N = 48$ it is 600 nodes, and so on. A two-level tree-structure without trunking with the same number of switches will allow $N(N-1)/2$ nodes to be connected, so we are losing a little less than 1/2 of connectivity.

The benefit of course is in attaining scalable bisection bandwidth. In particular, the following can be proven easily:

Theorem 2 The lower bound on the bisection bandwidth of the above network is 1/2, i.e., For N nodes connected to the network it is $LN/2$.

A simple proof here would be to construct a bipartite graph, and merely counting the crossing edge, and the lower bound is reached

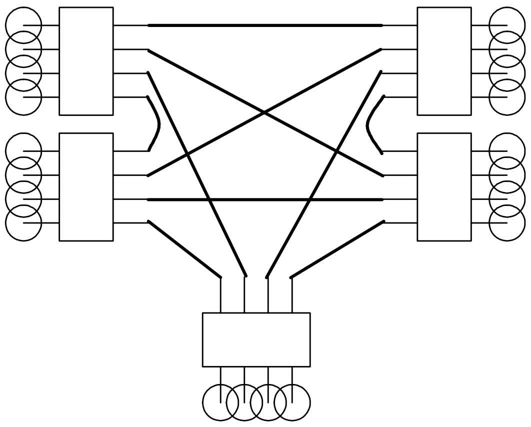
Fig. 3 Clique Network with Switch $N = 8$

asymptotically when $N \to \infty$. So for example for a 24-port Gigabit Ethernet Switch, the attainable bisection bandwidth is 78 Gbps or over 8 GBytes/s. While this still is not great, it is far superior to hierarchical trunking techniques. For example, for a 4-trunked network of similar size using 24-port switches, the bisection bandwidth will be only 32 Gbps by comparison.

Note that the network is being constructed while satisfying the requirements set forth in Section 2. For example, routing between switches can be completely static, as there always is a direct path between switches being a clique, and any alternative routes need not be taken.

5. Multi-path routing with active node forwarding

Although the network in Section 4 exhibits good bisection bandwidth when the entire machine is used, when parts of a machine are used for high-bandwidth communication, the trouble is that the 1/2 asymptotic bandwidth property will persist, while other parts of the network could be idle and unused. For example, our Presto III cluster employs a variant of this topology as shown in Figure 4 where multiple paths could be actively taken to exploit idle inter-switch networks, but this is only possible because of Myrinet network configuration and switch property, while not being possible or difficult with the current crop of inexpensive Ethernet switches.

The alternative solution we propose is to implement multi-path routing by *using nodes that reside in switches whose inter-switch link bandwidth is underutilized.* This is a technique similar to Condor Diskrouter[1] but the difference is that we will implement it so that the active forwarding node merely "loopbacks" the com-

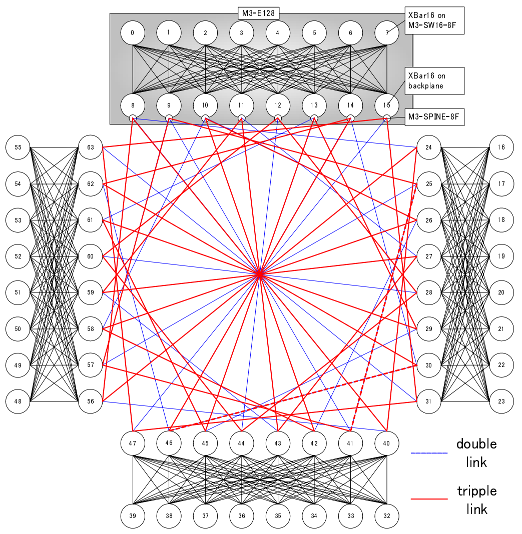

Fig. 4 Presto III Network Topology — Double Link and Triple Link indicates the trunking factor. Each circle indicates a 8 × 8 crossbar switch.

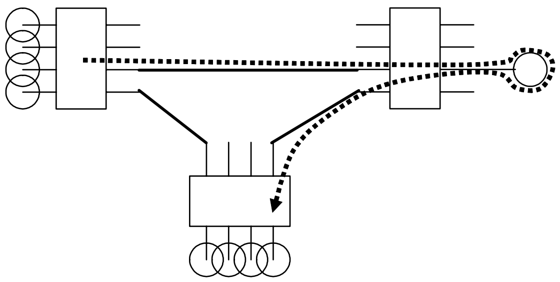

Fig. 5 Multi-path routing with active forwarding. Although multi-path communication (indirect trunking) solely by switches is not possible, by involving an active reflector node it becomes possible. Note that we are not losing bandwidth here assuming that the network activities in the network paths and the active reflector node are nearly idle.

munication data back to the switch for routing (Figure 5). We will also facilitate automated selection of appropriate idle inter-switch links and nodes. The "detour" route cannot have direct inter-switch hops except for a peer-to-peer connection case, but by sending the data first to a loopback node which effectively "deflects" it and send it off to the next switch destination serves to alleviate this problem.

As an example, consider the case of a simple blocked SOR; for any communication between the block tiles located within the same switch, one obtains full bandwidth, but for nodes on other switches the bandwidth reduces to almost half—and if the program is written in SPMD style, the slower bandwidth will dominate. Multi-route with active forwarding may (at least partially) solve this problem by effectively fattening the inter-switch bandwidth by multi-route message paths. The penalty is when there are other communications going on in the system that may share the detour route, or when the loopback node is communicating on its own.

6. Preliminary Benchmark Results.

The work is still in progress and we have not implanted software substrate that would serve directly as a messaging layer for MPI. Still, we have conducted some preliminary tests using Dolly+. One drawback with Dolly+ is that, since it is tuned solely for very large file transfers (over 100MBytes), it may exhibit some inefficiencies for small data sizes, both in terms of latency and bandwidth. Still, we could observe trends which may validate our proposals.

We conduct two experiments, (1) latency of the pipeline communication, and (2) effectiveness of the multi-path routing with active node forwarding. The benchmarks were conducted under the following setting:

- CPU: Dual Athlon MP 1900+ (Appro 1124)
- RAM: 768MB DDR
- PCI: 32bit/66MHz
- Linux Kernel 2.4.18
- Network: Switched 100base-T, Planex GX-222M (64 ports)

Figure 6 shows the result of latency measurement of cyclic ring (pipelined) communication. Dolly+ was configured to send 0-byte data, and the latency measured for the entire communication to complete, excluding any setup and cleanup times. We observe nearly linear increase in latency as expected, with approximately 400μseconds per node. So for a 256 node cluster the aggregate latency will be approximately 10 milliseconds, or about 100KBytes of transfer on a single link. Even under this setting, for a relatively large transfer the algorithm scales well. We could likely further reduce this overhead by possibly orders of magnitude by a) tuning the algorithm for shorter latency, and b) making the ring hierarchical at some point, achiveving usability for scatter-gather of 10-100KByte range, even likely for 1000Base-T networks.

Figure 7 illustrates the result of multi-path routing. Without going into details, the light square denotes the case where there is only one inter-switch link (no active node forwarding), and the black triangle when there are three links with two intermediate active forwarding node. The cross indicates the aggregated bandwidth. As can be seen, we obtajn almost double inter-switch bandwidth thanks to multi-path routing.

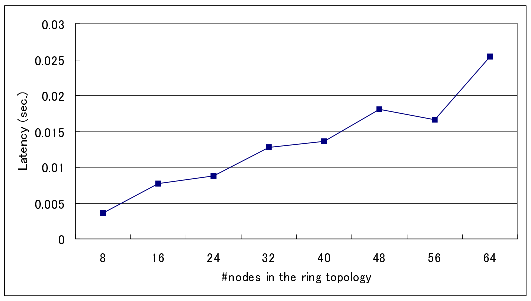

Fig. 6 Latency Measurement of Cyclic Ring Communication on Dolly+

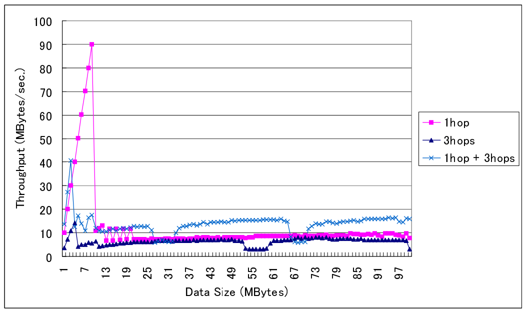

Fig. 7 Bandwidth of Multi-path Routing with Active Node Forwarding

In future work we will implement the appropriate communication layer for collective communications for MPICH, and present the results using real benchmarks.

7. Discussions and Conclusions

We have discussed the basis for using inexpensive switches to accommodate most forms of collective communication in commodity clusters, for all but the most communication demanding applications. Combined with further techniques, such as using tree-based communication structure for short messages when latency is more important, we should be able to achieve most of what very expensive and proprietary communication infrastructures will offer at a fraction of the cost.

The work is extensible to a group of clusters in the following manner. We allocate parts of the internal network interconnect of each of the clusters so that they form a clique network topology as described in Section 4, and employ multi-path active forwarding as described in Section 5. Barring the extreme case that the inter-cluster switch and the "border nodes" (the nodes that directly connect to the inter-cluster switches) are all busy, there is good chance that the full inter-cluster link bandwidth could be utilized. By employing inexpensive interconnect technology, we could do away with most SANs or expensive inter-machine networking technologies such as HIPPI.

As an example, suppose we have 5 128 node clusters configured using $N = 24$ Gigabit Ethernet switches as described above, with one of the clusters used mostly as a storage server; then, 28 links would be usable as inter-cluster networking, giving us inter-cluster bandwidth of almost 3 GByte/s. Assuming 20% I/O and inter-cluster bandwidth usage, we obtain almost full 3 GByte/s bandwidth, which would be adequate for most applications. In the case of Infiniband 4x the bandwidth will be 23GB/s, outpacing I/O bandwidth of fastest supercomputers today.

Future work will include the actual implementation on large clusters as well as the strategies for determining the loopback node in an effective fashion.

8. Acknowledgements

This work is partially supported by the JST CREST "Megascale Computing" project and NEDO "Dependable Cluster" project grants. We also greatly thank Yasuhito Takamiya and Atsushi Manabe for conducting the actual Dolly+ benchmarks.

References

1) Diskrouter. `http://www.cs.wisc.edu/condor/diskrouter/`.
2) Dolly+ home page. `http://corvus.kek.jp/~manabe/pcf/dolly/index.htm`.
3) Infiniband trade association: Home. `http://www.infinibandta.org/`.
4) Mpi - the message passing interface standard. `http://www-unix.mcs.anl.gov/mpi/`.
5) Myricom home page. `http://www.myri.com/`.
6) Top500 supercomputer sites. `http://www.top500.org/`.
7) Graham Fagg, Jack Dongarra In J. Dongarra, P. Kacsuk, and N. Podhorszki (Eds.). Ft-mpi: Fault tolerant mpi, supporting dynamic applications in a dynamic world. In *Lecture Notes in Computer Science 1908*, page 346. Springer Verlag, September 2000.
8) Shin'ichi Miura, Taisuke Boku, Mitsuhisa Sato, and Daisuke Takahashi. Ri2n - interconnection network system for clusters with wide-bandwidth and fault-tolerancy based on multiple links. *IPSJ SIGNotes High Performance Computing*, 093, 2003.
9) Yasuhito Takamiya and Satoshi Matsuoka. Towards mpi with user-transparent fault tolerance. In *Proceedings of JSPP2002*, pages 217–224, 2002.