Proceedings of the 32nd Hawaii International Conference on System Sciences - 1999

The P-Mesh—A Commodity-based Scalable Network Architecture for Clusters

Bill Nitzberg, Chris Kuszmaul, Ian Stockdale, Jeff Becker, John Jiang, Parkson Wong

MRJ Technology Solutions
Numerical Aerospace Simulation
NASA Ames Research Center, M/S 258-6
Moffett Field, CA 94035-1000

nitzberg@nas.nasa.gov

Abstract

We designed a new network architecture, the P-Mesh, which combines the scalability and fault resilience of a torus with the performance of a switch. We compare the scalability, performance, and cost of the hub, switch, torus, tree, and P-Mesh architectures. The latter three are capable of scaling to thousands of nodes, however, the torus has severe performance limitations with that many processors. The tree and P-Mesh have similar latency, bandwidth, and bisection bandwidth, but the P-Mesh outperforms the switch architecture (a lower bound for tree performance) on 16-node NAS Parallel Benchmark tests by up to 23%, and costs 40% less. Further, the P-Mesh has better fault resilience characteristics. The P-Mesh architecture trades increased management overhead for lower cost, and is a good bridging technology while the price of tree uplinks is expensive.

1.0. Commodity Supercomputing: The Beowulf Approach

A fundamental shift is taking place in the computer marketplace: high-end scientific computing is disappearing. Computer vendors are merging (SGI and Cray, HP and Convex), and re-focusing on more lucrative mass markets (e.g., business, Wall Street, databases). These markets do not benefit from large systems, hence the motivation for vendors to build these systems is rapidly disappearing.

The Whitney project has two goals: to ensure NASA can maintain a high-performance scientific computing capability in the face of this fundamental market shift, and to push the limits of system and application scalability to enable petaflops computing. Achieving these goals requires new technology harnessing the power of tens of thousands of mass-market commodity off-the-shelf (M^2COTS) components to build high-end systems targeted at aerospace applications.

The Beowulf approach [8] provides a framework for building M^2COTS clusters. It couples commodity PCs and LANs with a publicly available operating system and application software, to produce distributed memory multiprocessing systems with excellent price/performance. To date, systems up to a few hundred nodes in size have been built. We are using this methodology to design Whitney-500, a 500 node system for running scientific workloads, with a goal of scaling up to several thousands of nodes.

The design of Whitney-500 is unique in that it combines a "back-end computing" design philosophy with mass-market technology. The design was directed by price/performance on a small set of representative benchmarks (the NAS Parallel Benchmarks [2]) much the way a back-end array processor might be designed—tuning for the exact application which it will support. In this case, Whitney-500 was designed to support the computational fluid dynamics workload at NASA Ames Research Center.

The main focus of the Whitney project is in three areas:

- price-based performance modeling,
- extremely scalable systems software research and development,
- architecture design, prototyping, and evaluation.

The focus on price/performance on a specific set of benchmarks rather than peak performance or general system balance allowed us to make design choices which, at first glance, appear counter to common wisdom, such as using a relatively slow network despite cries that everyone's application "needs" a fast network.

Both analytical models and empirical studies [3,5] show that Fast Ethernet (100baseT) is the network of choice for Whitney. It isn't the fastest, but it is very inexpensive, and well balanced to the processing power of the Intel Pentium processors.

The Whitney project is named after Eli Whitney, who, in addition to inventing the cotton gin, was the first to use interchangeable parts in manufacturing---he invented "commodity" components.

1.1. Whitney Prototype Hardware

All performance results were run on the prototype Whitney system. Each node of the system was running an identical copy of Red Hat Linux [6], release 4.1 using the 2.0.30 version of the Linux kernel, and the tulip (v0.76 4/22/97) driver. The kernel was configured with IP forwarding on. During our analysis, newer versions of OS software became available. However, to ensure consistency of results, no upgrades were performed.

The nodes of the Whitney prototype consist of:

- Intel Pentium Pro 200 MHz processor with 256K cache
- ASUS P/I-P65UP5 motherboard w/P6ND CPU board
- 128 MB 60ns DRAM
- 2.5 GB Western Digital AC2250 IDE disk
- 1-4 Cogent/Adaptec ANA-6911/TX Ethernet cards
- Trident ISA graphics card (for diagnostic purposes only)

The network tests were performed using

- 3 Bay Networks Netgear FE 516 (16-port) hubs
- 2 Bay Networks 350F-HD (24-port) switches

The Bay switches have a backplane bandwidth of 1.2 Gb/s (150 MB/s). A fully nonblocking 24-port switch would need a 2.4 Gb/s (300 MB/s) fabric.

2.0. Network Characteristics

Our design goal for the Whitney project is a system with thousands of nodes. A system of this magnitude has unique scalability requirements on the network architecture:

- Price/performance—the ultimate goal of M^2COTS clusters
- Simple packaging—short, robust cables, no spaghetti
- Easy software configuration and management—simple setup, easy problem diagnosis, ability to expand on-the-fly, etc.
- Fault resilience—no single points of failure

2.1. Ethernet Technology

The building blocks of a commodity Ethernet network consist of network interface cards (NICs), cables, hubs, and switches. Further, commodity Ethernet is now available in three speeds: Ethernet (1.25 MB/s), Fast Ethernet (12.5 MB/s) and Gigabit Ethernet (125 MB/s). Note that we always report bandwidth in megabytes per second (MB/s). The peak speed of Fast Ethernet is 100 Mb/s (megabits per second); 100 Mb/s divided by 8 bits per byte equals 12.5 MB/s. Fast Ethernet is currently the leader in price/performance, delivering adequate performance for approximately 1/10th the cost of a typical cluster node (see Table 1 for rough pricing as of Spring 1998). As the cost of a cable is negligible (about $5), we have omitted it from our analysis.

These prices are averages. Prices vary depending on vendor, actual performance of the component, and expandability. For example, 24-port Fast Ethernet switches are particularly inexpensive as of the writing of this paper (less than $100 per port).

TABLE 1. Ethernet technology prices (Spring 1998)

Item		Cost ($)	Available Sizes
Fast (100Mb) NIC	$\$_{NIC}$ =	30	-
Fast hub port	$\$_{hub}$ =	50	4 - 100
Fast switch port	$\$_{sw}$ =	150	16 - 100
Gigabit NIC	$\$_{G\text{-}NIC}$=	1200	-
Gigabit switch port	$\$_{G\text{-}sw}$=	2500	6 - 32

Finally, design considerations limit the maximum size of Fast Ethernet hubs and switches to about 100 ports; and the largest Gigabit Ethernet switches currently available have about 30 ports.

3.0. Network Architecture

Network architecture defines how the building blocks are assembled. Figure 1 shows several possible architectures. The simplest network architecture consists of two nodes, two NICs (one in each node), and one cable connecting the two NICs (Figure 1 (a)). Obviously, this architecture has severe limitations (scaling to at most 2 nodes), but it is useful for illustration purposes.

There are three network characteristics which, when taken together, have proved to be a valuable measure of network performance:

- point-to-point latency—the minimum amount of time to pass a message between two processes
- point-to-point bandwidth—the maximum data rate that can be achieved transferring messages between two processes
- bisection bandwidth—the worst-case aggregate bandwidth among all bisections of the nodes

We used the following techniques to measure these quantities.

Message latency is measured by sending M messages from A to B and back (ping-pong) and dividing the total time by $2 \cdot M$. This ping-pong test is repeated for a variety of message sizes. The minimum time is reported as latency. We call this the base latency or L. The measured latency of this architecture (using the Whitney prototype) was 175 microseconds.

Bandwidth is reported similarly. For each message size, we computed the message size divided by the time as a candidate bandwidth. The base bandwidth B. is the largest candidate bandwidth. The base bandwidth was 8.5 MB/s.

As bisection bandwidth is somewhat more difficult to measure, we report "peak" bisection bandwidth (a property of the architecture itself). Bisection bandwidth is the sum of rated bandwidths along the minimum network cut which separates the compute nodes into two equal sized groups. In the example above, the bisection bandwidth would be $2B$ (each node can send data at a rate of B MB/s to the other. Using Fast Ethernet, the bisection bandwidth works out to 25 MB/s, or 12.5 MB/s per node.

3.1. Hub

A hub, Figure 1 (d), is a generalization of this example architecture, providing the logical equivalent of many nodes sharing a single cable. A hub has a cost per node of ($\$_{NIC}$ + $\$_{hub}$), or about $80.

The performance of a hub is nearly identical to that of directly connected nodes (a latency of L and a bandwidth of B). However, bisection bandwidth is particularly poor, as all nodes logically share a single network cable. The per node bisection bandwidth for N nodes is $\frac{2B}{N}$. In other words, as N grows, per node bisection bandwidth shrinks dramatically. A Fast Ethernet hub with 24 nodes would have a bisection bandwidth of about 1 MB/s per node.

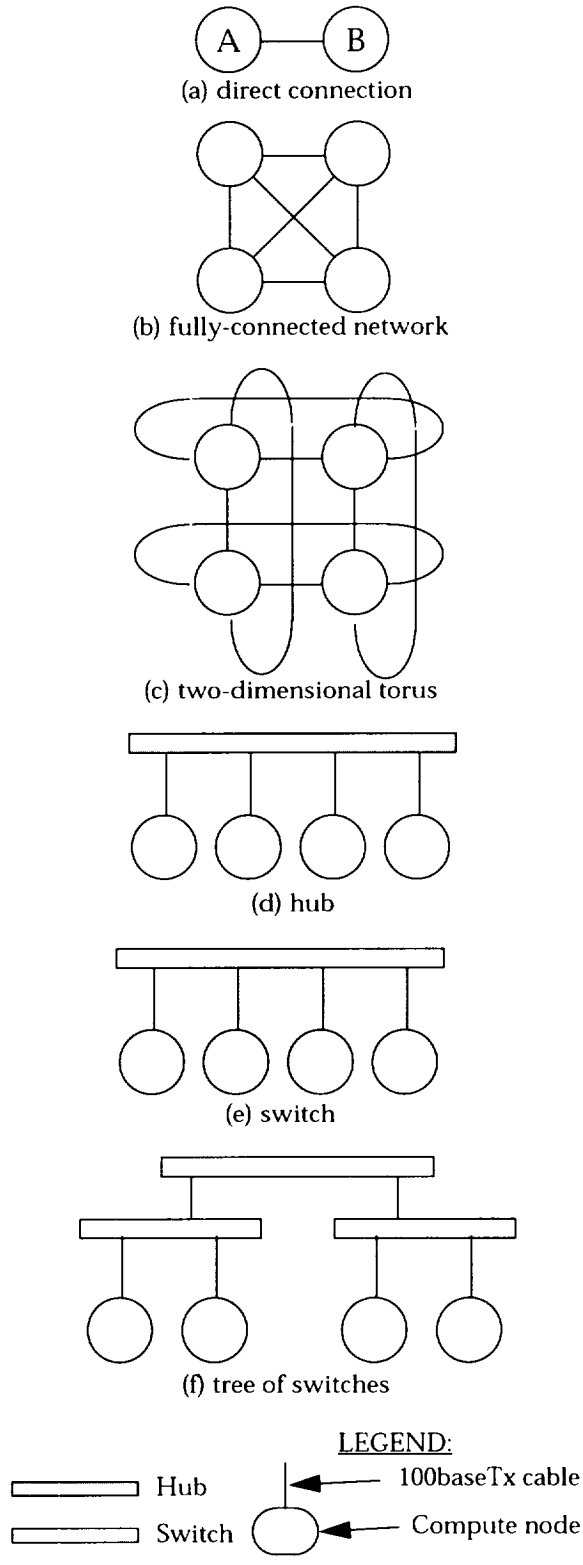

FIGURE 1. Network topologies

3.2. Switch

A switch, Figure 1 (e), goes further, and provides the logical equivalent of (N/2) cables which automatically connect pairs of nodes when they want to send messages. Another way to look at a switch is as a "fully connected network", Figure 1 (b), where each node only has a single cable connected to the network fabric.

The defacto Ethernet network architecture for cluster computing is to connect all nodes to a single Fast Ethernet switch. Few larger clusters have been proposed, hence little work has been done evaluating alternatives to this de-facto standard.

The switch architecture has three main advantages:

- Switches are mass-market commodity off-the-shelf items, and as such benefit from brutal cost-competitiveness.
- Switches deliver good network performance (latency, bandwidth, and bisection bandwidth).
- Switch architectures are easy to build (all wires can be short) and simple to configure (just plug nodes in and go).

However, this architecture is not without drawbacks:

- Switches don't scale—mass-market pressures only force down the price of "small" switches (supporting about 24 ports), and "large" switches (greater than about 100 ports) don't exist.
- Switch architectures lack hardware fault resilience— a single point of failure (the switch) can bring down the entire system.

A Fast Ethernet switch is relatively inexpensive, with a per node cost of ($\$_{NIC}$ + $\$_{sw}$), or about $180. Gigabit Ethernet, however, is quite expensive, with a per node cost of about $3700 (20 times more expensive for minimally better latency and 10 times better bandwidth). With the cost of a typical cluster node around $2000, Gigabit Ethernet is not yet in the M^{2}COTS commodity range. However, Gigabit technology is on the commodity curve, and we expect it to supplant Fast Ethernet as the network of choice in the next few years. For now, Gigabit Ethernet is a useful bridging technology for aggregating Fast Ethernet devices (for the tree architecture below).

Switches perform nearly as well as directly connected nodes, but do add a small amount of latency (which we call L_{+sw}), and slow bandwidth by a small amount (which we call B_{-sw}). The Bay 350F-HD added L_{+sw} of about 5 microseconds, and reduced B_{-sw} by about 0.5 MB/s.

Our requirement to design a system capable of scaling to thousands of nodes led us to investigate alternative network architectures. Two architectures stood out as obvious candidates: the torus and the tree, see Figure 1 (c) and (f).

3.3. Torus

The torus architecture uses the direct connect approach, eliminating the need for using switches (which have limited hardware scalability), but adding the need to route messages (up to $\sqrt[d]{N}$ hops for a d-dimensional torus of N nodes). The torus is interesting because it is quite inexpensive, costing (2d $\$_{NIC}$) per node, or about $120 per node for a 2-dimensional torus. Further, the torus has particularly good hardware fault resilience. Up to $d-1$ failed cables or NICs can be tolerated (by re-routing messages), and, if all d links to a node fail, only that node becomes inaccessible.

This expanded scalability comes at a high price in performance. All messages except nearest neighbor communication must be routed by the nodes. Every hop a message travels adds routing latency (L_{+r}) and reduces bandwidth (B_{-r}). For a medium sized torus, this adds up quickly. In a 2-dimensional, N-node torus, the worst case route between nodes traverses through $\sqrt{N}-1$ router nodes, resulting in a worst-case latency of $L+(\sqrt{N}-1)L_{+r}$ and a worst case bandwidth of $B-(\sqrt{N}-1)B_{+r}$. Our measurements show L_{+r} is about 40 microseconds and B_{-r} is about 1.5 MB/s. The bisection bandwidth, however, is quite good, maintaining B MB/s per node independent of dimension or size.

Finally, the torus requires a non-trivial amount of setup and configuration management to connect all the necessary cables and create all the necessary routing tables.

All in all, what is gained in architectural scalability is lost in performance and management scalability. For a small number of nodes, this architecture may be O.K.—but for a small number of nodes, a switch does not have a scalability problem, and requires less management.

3.4. Tree

The tree architecture is the most natural design from the LAN perspective. (In fact, all vendors to which we outlined our project suggested this architecture.) The tree architecture is constructed by connecting switches together using a higher performance network (e.g., connecting Fast Ethernet together with Gigabit uplinks). In order to maintain bisection bandwidth, the higher levels in the tree use either more links or higher bandwidth links and switches.

The bisection bandwidth for this architecture is a maximum of 12.5 MB/s per node (as each node has a single bidirectional 12.5 MB/s link connecting it to the network). In practice, however, the bisection bandwidth will be limited further by: Fast Ethernet switch fabric capacity, the

number of gigabit links connecting Fast Ethernet switches to the gigabit switch, or the gigabit switch fabric capacity.

In this architecture, base latency and bandwidth are affected little. For a two-level tree, the worst case latency would be $L + 3L_{+sw}$, and the worst-case bandwidth would be $B - 3B_{-sw}$. Given our measurement of additional switch latency and reduced bandwidth, we conjecture that this would yield a latency of 190 microseconds and a bandwidth of 7 MB/s. However, we believe these numbers are quite conservative, and would expect better performance (certainly in terms of bandwidth) due to the pipelined nature of the way large messages are sent. We were unable to test a tree architecture, as Gigabit switches were just becoming available at the time of this writing.

Using currently available Ethernet technology, the tree architecture trades bisection bandwidth for scalability beyond 100 nodes. (Cost is also directly correlated with bisection bandwidth.) Most Fast Ethernet switches with gigabit ports and gigabit switches have sufficient switch fabric capacity to supply non-blocking service. For this reason, the gigabit links connecting switches are likely to be the bottleneck. A system with 768 nodes can be constructed using 24-port Fast Ethernet switches, each with two Gigabit uplinks, all connected to 2 32-port Gigabit switches. This would give a per node bisection bandwidth of $2B_{GBIT}/24$, or 250/24 = 10.4 MB/s. Cost, however, is significantly greater than for the switch, costing $\$_{NIC}+\$_{sw}+(4/24)\$_{G\text{-}sw}$, or about $597 per node. Reducing bisection bandwidth to 5.2 MB/s by using a single Gigabit uplink per 24 nodes reduces the cost to $\$_{NIC}+\$_{sw}+(2/24)\$_{G\text{-}sw}$, or about $388 per node.

Although the tree architecture is currently not scalable to tens of thousands of nodes (our target), we expect technology improvements in both switch density and backplane bandwidth. However, even the tree architecture suffers from a lack of failure resilience—the failure of the top level Gigabit switch would render the entire system useless.

4.0. P-Mesh Architecture

Our detailed network comparison led us to design a hybrid architecture, the P-Mesh, with the architecture scalability and hardware fault resilience of the torus, but without the loss of performance scalability. We replace the direct connections along a line of processors in a given dimension of the torus with a switch. Meanwhile, nodes continue to route messages that traverse more than one dimension.(see Figure 3).

In general, a d-dimensional P-Mesh with M_i nodes in dimension i has $N = \prod_{i=1}^{d} M_i$ nodes and requires dN network interfaces, dN network cables, and $\sum_{i=1}^{d} M_i$ switches. In particular, the j-th dimension requires $\prod_{(i=1)(i\neq j)}^{d} M_i$ switches, each capable of handling M_i ports. An N-node "square" P-Mesh is denoted as n^d P-Mesh (each dimension is of size n and $n^d = N$).

The cost of a P-Mesh is more than a single switch, but less than the tree. Each P-Mesh node costs $d(\$_{NIC}+\$_{sw})$, or about $360 per node for a 2-dimensional P-Mesh.

The network characteristics of the P-Mesh are a combination of switch characteristics and torus characteristics. Assuming switches are non-blocking (i.e., bisection bandwidth equal to 12.5 MB/s per node), then a P-Mesh has worst-case latency of $L+(d-1)L_{+r}+dL_{+sw}$, and worst-case bandwidth of $B-(d-1)B_{-r}-dB_{-sw}$. For a 2-dimensional P-Mesh, the worst-case routing would add a single hop, and should result in about 225 microseconds latency (we measured 230 microseconds), and about 6 MB/s bandwidth (exactly as measured).

The P-Mesh inherits the good bisection bandwidth of the torus, delivering a constant B MB/s per node independent of the number of nodes or dimension (assuming non-blocking switches). If the individual switches provide less than 12.5 MB/s per node bisection bandwidth then the bisection bandwidth of the whole system will be this lesser amount (e.g., 9.4 MB/s per node for a 16^2 P-Mesh based on the Bay 350 series which has a backplane bandwidth of 1.2 Gb/s).

However, as the dimension of the P-Mesh increases, the latency gets worse. For the system sizes we are interested in (approximately 500 nodes) the degenerate one-dimensional P-Mesh (a single switch) is not available, but a 2-dimensional P-Mesh can be built using 24-port M^2COTS parts supporting up to 576 nodes.

The fault resilience of the P-Mesh is similar to that of the torus. A single switch failure can be tolerated by updating the routing tables, and two switch failures will, at most, result in the loss of a single node.

The P-Mesh does suffer from increased packaging complexity (a number of cables must be "long"), as well as increased configuration and management overhead (to set up routing tables). The regular nature of the P-Mesh architecture, however, allows most of the configuration and management to be easily handled by a single setup script.

A slight modification of the P-Mesh architecture allows switches to cover multiple dimensions (e.g., one switch for every two rows rather than one switch per row). This can reduce the parts count by allowing higher density (more ports) switches to be used at a cost of reduced bisection bandwidth.

4.1. P-Mesh IPv4 Addressing

Although the P-Mesh architecture is general, and can be used with any underlying protocol, the most common, hence, most commodity, protocol is the Internet Protocol (IP) [9]. IP version 4 (IPv4) uses 32-bit addresses which are broken up into a network part, a subnet part, and a host part. P-Mesh IPv4 addressing further breaks up the subnet portion of the address into dimension and P-Mesh "subnet vector". (IP version 6 will allow multiple levels of subnet, making the addressing and routing in a P-Mesh somewhat less complicated—however, IPv6 is still under development [4]).

The design of the P-Mesh IPv4 addressing scheme had three goals:

- Compact encoding: a minimal number of address bits are required to encode a P-Mesh of a particular size
- Ease of growth: the addressing for an n^d P-Mesh can be used to assign addresses for all smaller m^d P-meshes (m<n). In other words, once an upper bound is chosen for the size, individual nodes do not have to have their addresses changed to grow the system.
- Ease of routing: a simple script can be used to generate all routes for any size P-Mesh

The method chosen is to break up bits into: **d**imension, subnet vector, and **h**ost (written "**d**.[**n**].**h**").

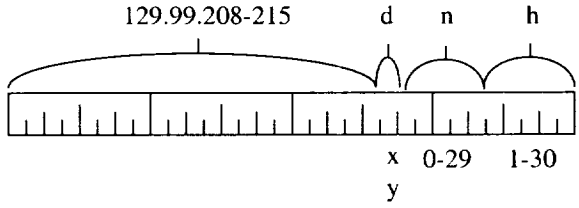

FIGURE 2. Example 30^2 P-Mesh addressing for Whitney design. Multiple subnets on the NAS IPv4 class B network support the proposed 12 x 24 P-Mesh. This addressing scheme would support up to a 30 x 30 P-Mesh.

Given $N = n^d$ nodes, an n^d P-Mesh has:

- $\lceil \log(d) \rceil$ dimension bits
- $(d-1)\lceil \log(n) \rceil$ subnet bits
- $\lceil \log(n+2) \rceil$ host bits

A two-dimensional P-Mesh uses one bit for dimension (X vs. Y), $\lceil \log(n) \rceil$ bits for the subnet number in that dimension, and $\lceil \log(n+2) \rceil$ bits for the host number on that subnet. The largest two-dimensional P-Mesh that can be constructed using IPv4 addressing (using a class A address such as 10.0.0.0) would support up to a 2048^2 P-Mesh. However, given the M^{2}COTS technology with the best price/performance in May '98 (24-port Fast Ethernet Switches are under $100 per port), the most reasonable architecture would be a 24^2 P-Mesh. This size requires 11 bits for IPv4 addressing: 1 dimension bit, 5 subnet bits (representing 0 - 23), and 5 host bits (representing 1 - 24).

In a d-dimensional P-Mesh, each node has d addresses. For example, node $\vec{m} = (m_0, m_1, \ldots, m_{d-1})$ would have one IP address for each dimension:

$$0.[m_{d-1}\ldots m_k \ldots m_2 m_1].m_0$$

$$1.[m_{d-1}\ldots m_k \ldots m_2 m_0].m_1$$

$$\ldots$$

$$k.[m_{d-1}\ldots m_{k+1} m_{k-1} \ldots m_1 m_0].m_k$$

$$\ldots$$

$$d-1.[m_{d-2}\ldots m_k \ldots m_1 m_0].m_{d-1}$$

where the notation $[abc]$ represents the concatenation of the $\lceil \log(n) \rceil$ bits representing $a+1$ followed by those representing $b+1$ then $c+1$.

Note that P-Mesh subnet address are numbered starting from 0, while the P-Mesh host portion is numbered starting at 1. This saves one bit per dimension in the IP address for an n^d P-Mesh, where n is a power of 2—if n is not a power of 2, the subnets can be numbered more sensibly starting at 1. The "[]" notation hides this for convenience.

4.2. P-Mesh IPv4 Routing

We decided to use X-Y routing due to its simplicity and semantic properties. In X-Y routing, a message from one node to another first travels along dimension 0, then dimension 1, etc. In a 2-dimensional mesh, a message would first travel along the X dimension, then the Y dimension (hence the name X-Y routing). Commercial mesh-based highly parallel systems (e.g., the Intel Paragon and Cray T3E) also use X-Y routing. However, the commercial systems use proprietary (non-commodity) technology which also allows them to implement fancy routing techniques such as worm-hole and cut-through routing.

We designed a routing scheme using IPv4 subnet routes. In an n^d P-Mesh, each node has dn^{d-1} routing entries. In

particular, the routing table for node $\vec{m} = (m_0, m_1, \ldots, m_{d-1})$ could be generated by the following pseudocode:

```
for k = 0..d-1
  for s_0 = 1..n, s_1 = 1..n, ...,
                                  s_{d-2} = 1..n
    for i = 0..d-2
      if (m_i ≠ s_i) {
        add route to net k.[s_0...s_{d-2}].0
            through node
                i.[m_{d-1}...m_{i+1}m_{i-1}...m_1m_0].s_i
        break
      }
```

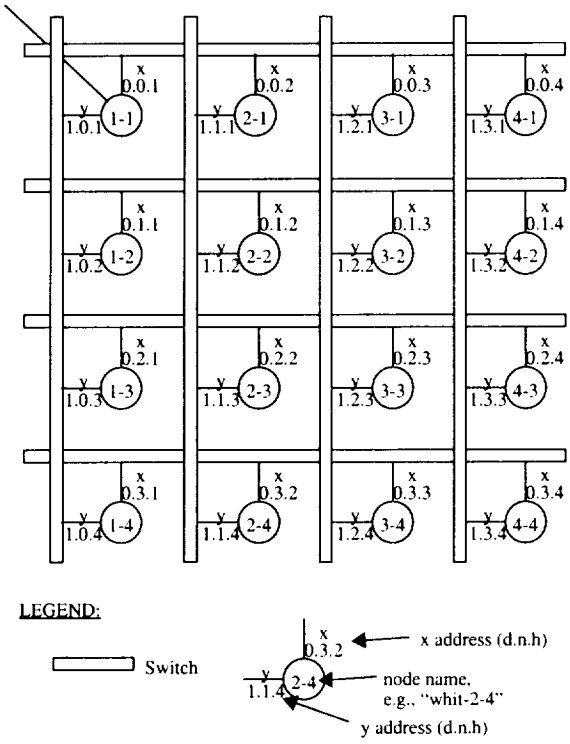

FIGURE 3. Generic 4^2 P-Mesh

For example, on our proposed Whitney system, a 24^2 P-Mesh. Node (x, y) has two network interfaces, addressed 0.[y].x and 1.[x].y. The routing table would be constructed using a simplified version of the above:

```
for k = 0..1
  for s = 1..24
   if (x != s)
      add route to k.[s].0
            through 0.[y].s
   else if (y != s)
      /* interface 1.[x].y is used */
   else
      /* no route needed to self! */
```

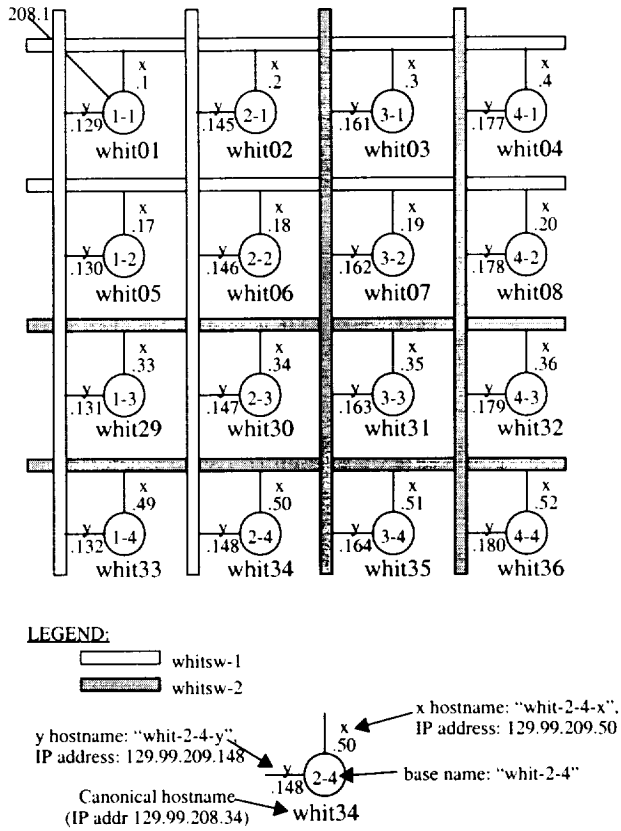

FIGURE 4. Whitney 4^2 P-Mesh test configuration (two Bay 350F-HD 24-port switches were used to create 8 logical 4-port 100baseT switches.)

5.0. Architecture Summary

Of the architectures we evaluated, only the torus, tree, and P-Mesh are capable of scaling to thousands of nodes. Although the torus hardware is scalable, the performance degrades proportional to the number of processors by a significant amount. For this reason, we do not consider the torus a viable architecture for use with more than a few dozen processors—and at this small size, a switch is clearly the winner.

The worst case latency and bandwidth of both the tree and P-Mesh are fixed. The tree is slightly better in both regards, as performance loss due to a node router is generally greater than that due to a switch. However, the P-Mesh gives better bisection bandwidth (especially if only a single uplink is used to construct the tree to save money).

The P-Mesh has better fault resilience characteristics, as the tree cannot sustain a single point of failure, while the P-Mesh is highly redundant. However, the P-Mesh requires substantially more packing and configuration management overhead.

Note that our estimates of tree performance are probably pessimistic. We expect actual performance to be closer to that of a single "big" switch. For this reason, in our analysis, we compare the P-Mesh results to the single switch results.

TABLE 2. Network architecture characteristics

Architecture	Scalability (max nodes)	Cost ($ per node)	Latency (usecs)	Bandwidth (MB/s)	Bisection Bandwidth (MB/s/node)
Direct Connect	2	$\$_{NIC}$	L	B	B
Hub	100	$\$_{NIC}+\$_{hub}$	L	B	2B/N
Fully Connected	∞	$(N-1)\$_{NIC}$	L	B	(BN)/4
Switch	100	$\$_{NIC}+\$_{sw}$	$L+L_{+sw}$	$B-B_{-sw}$	B
2D Torus	∞	$4\$_{NIC}$	$L+(\sqrt{N}-1)L_{+r}$	$B-(\sqrt{N}-1)B_{-r}$	B
Tree[a]	768	$\$_{NIC}+\$_{sw}+(4/24)\$_{G-sw}$	$L+3L_{+sw}$	$B-3B_{-sw}$	(20/24)B
2D P-Mesh[b]	10,000	$2(\$_{NIC}+\$_{sw})$	$L+2L_{+sw}+L_{+r}$	$B-2B_{-sw}-B_{-r}$	B

a. 32 24-port Fast Ethernet Switches each with 2 uplinks to 2 32-port Gigabit switches

b. 100^2 P-Mesh (maximum sized Fast Ethernet switches are about 100 ports)

Comparing the cost of the Tree and 2D P-Mesh, and solving for $\$_{G-sw}$, shows that the tree architecture will cost less as soon as a Gigabit Ethernet switch port costs less than $(24/4)(\$_{NIC}+\$_{sw})$, or, if we ignore the cost of the NIC, the two will be equivalent when $6\$_{NIC} = \$_{G-sw}$. At current prices ($\$_{NIC}+\$_{sw} = \$180$) the tree would cost less than the P-Mesh, if $\$_{G-sw}$ is less than $1080.

TABLE 3. Network comparison for 16 nodes (Spring 1998)

Network Architecture	Cost ($ per node)	Latency (usec)		Bandwidth (MB/s)		Bisection Bandwidth (MB/s per node)
		best-case	worst-case	best-case	worst-case	
Direct	30	175		8.5		12.5
Hub	80	175		8.5		1.6
Switch	180	180		8.0		9.4
2D Torus	120	175	295	8.5	4.0	12.5
Tree	597	190[a]		7.0[a]		10.4
2D P-Mesh	360	180	230	8.0	6.0	9.4

a. Tree latency and bandwidth performance are predicted (not measured) using architectural characteristics.

A summary of measured network characteristics appears in Table 3. The predicted latency of the P-Mesh, $L+2L_{+sw}+L_{+r}$, works out to 175 + 2(5) + 40 = 225 (we measured 230). Similarly, the P-Mesh bandwidth was predicted to be $B-2B_{-sw}-B_{-r}$, or 8.5 - 2(0.5) - 1.5 = 6.0 (as measured).

5.1. NAS Parallel Benchmarks

We use the application benchmarks LU, BT and SP from the NAS Parallel benchmark suite [2], version 2.2 in order to compare network performance. These code are written using the MPI (message passing interface) library, and are representative of the computation fluid dynamics workload typically run at our facility. All three benchmarks compute the solution of a system of five partial differential equations. LU uses the symmetric successive over-relaxation (SSOR) algorithm, while BT and SP respectively implement block tridiagonal, and scalar pentadiagonal variants of an alternating direction implicit (ADI) solver.The benchmarks have 3 different classes: A (smallest), B and C (largest), depending on the problem size (number of grid points) and number of iterations.

TABLE 4. NAS Parallel Benchmarks class A performance on 16 nodes

Network Architecture	NPB-A 2.2 (MFLOPS)		
	LU	BT	SP
Hub	210.53	235.28	105.54
Switch	338.55	228.14	121.89
Torus	355.37	255.79	133.11
P-Mesh	398.56	247.11	117.16

TABLE 5. NAS Parallel Benchmarks class B performance on 16 nodes

Network Architecture	NPB-B 2.2 (MFLOPS)		
	LU	BT	SP
Hub	328.1	282.25	152.36
Switch	327.22	256.46	162.89
Torus	402.22	323.31	191.82
P-Mesh	404.20	309.96	174.93

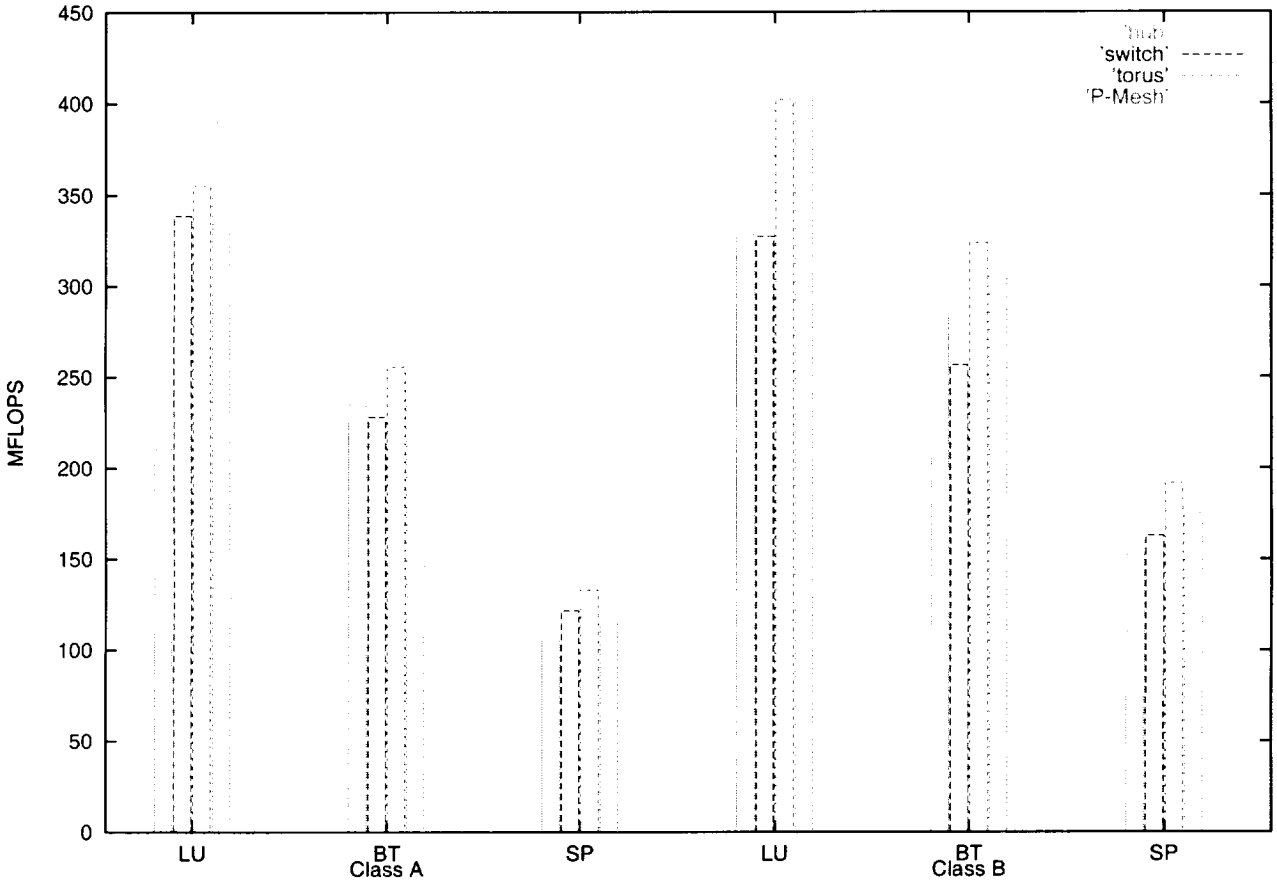

FIGURE 5. NAS Parallel Benchmark performance (version 2.2) comparing various Fast Ethernet architectures on 16 nodes

Tables 4 and 5 and Figure 5 show the results of class A and B performance for the hub, switch, torus, and P-Mesh architectures on 16 nodes (a 4 by 4 mesh is used for the torus, and P-Mesh tests). The hub and torus numbers are taken from [5]. Surprisingly, the switch is better than the hub architecture in only half the cases. The torus is better than both the switch and the hub, as is the P-mesh (except for SP class A).The P-mesh is better than the torus on LU class A and B, and is 3-12% slower on BT and SP.

6.0. Related Work

The main contribution of this work is two fold: the analytical and empirical evaluation of networking topologies

for M²COTS clusters, and the design of a new topology, the P-Mesh. Some of the early work on the Beowulf project also focused on evaluation of networking topologies [7]. The Beowulf project experimented with hubs as the interconnect for each row and column of a four by four mesh of PCs—at the time, switching technology was not a "commodity". The idea of using bridging hardware to interconnect rows and columns of a mesh of processors dates back as far as the mid-1980s, with the DAP [1], which used buses for the interconnect mechanism.

7.0. Summary

We designed a new network architecture for M²COTS clusters: the P-Mesh. The P-Mesh combines the scalability and fault resilience of a torus with the performance of a switch.

We compared the scalability, performance, and cost of the hub, switch, torus, tree, and P-Mesh architectures. The latter three are capable of scaling to thousands of nodes, however, the torus has severe performance limitations with that many processors. The tree and P-Mesh have similar latency, bandwidth, and bisection bandwidth, but the P-Mesh outperforms the switch architecture (a lower bound for tree performance) on 16-node NAS Parallel Benchmark tests by up to 23%, and costs 40% less. Further, the P-Mesh has better fault resilience characteristics, able to tolerate multiple failures. The P-Mesh architecture trades increased management overhead for lower cost, and is a good bridging technology while the price of tree uplinks is expensive.

8.0. Acknowledgments

This work was performed under NASA contract NAS2-14303.

Kevin Pedretti enabled us to gather last minute P-Mesh results by judicious hacking on the Whitney prototype, and, along with Sam Fineberg, gathered an immense amount of data on 100 Mb Ethernet performance which we were grateful to be able to draw on.

9.0. References

[1] Active Memory Technology, "DAP Series Technical Overview", 1988.

[2] Bailey, D., T. Harris, W. Saphir, R. Van der Wijngaart, A. Woo, and M. Yarrow, "The NAS Parallel Benchmarks 2.0", Applied Research Branch Report RNR-95-020, NASA Ames Research Center, Moffett Field, CA 94035-1000, 1995.

[3] Becker, Jeffrey C., Bill Nitzberg, and Rob F. Van der Wijngaart, "Predicting Cost/Performance Trade-offs For Whitney: A Commodity Computing Cluster", HICSS '98, the Hawaii International Conference on Systems Sciences, Hawaii, January 1998.

[4] Deering, Stephen E. and, Robert M. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 1883, December 1995.

[5] Fineberg, Samuel A. and Kevin T. Pedretti, "Analysis of 100Mb/s Ethernet for the Whitney Commodity Computing Testbed", Technical Report NAS-97-025, NASA Ames Research Center, Moffett Field, CA 94035-1000, October 1997.

[6] Red Hat Linux, Red Hat Software, Inc., available via www.redhat.com.

[7] Reschke, Chance, Thomas Sterling, Daniel Ridge, Daniel Savarese, Donald Becker, Phillip Merkey, "A Design Study of Alternative Network Topologies for the Beowulf Parallel Workstation", Proceedings, High Performance and Distributed Computing, 1996

[8] Ridge, Daniel, Donald Becker, Phillip Merkey, and Thomas Sterling, "Beowulf: Harnessing the Power of Parallelism in a Pile-of-PCs", Proceedings, IEEE Aerospace, 1997

[9] Stevens, Richard W., "TCP/IP Illustrated Vol. 1: The Protocols", Addison Wesley, 1994.