Better utilization of a diverse computing environment

Todd Hansen *

June 9, 2000

Abstract

Supercomputer designs are diverging further every day. Originally we had a single vector CPU that was super fast. Now we have competing architectures of distributed processors in thousands of different configurations. In this paper I am going to discuss the tradeoffs between a distributed multiprocessor architecture (the Blue Horizon at SDSC) and a traditional vector architecture (the Cray T90 at SDSC). The end result of this analysis is that despite the Blue Horizons horrible memory bus bottle neck, it can still beat the Cray for many applications. This occurs for three reasons: Blue Horizon is cheaper, more scalable and has a larger number of CPUs.

1 Introduction

Today a supercomputer is no longer just a machine developed by Cray Research Inc. Instead there are many competing companies proposing vastly different architectures. The original supercomputers developed by Cray Research fit into a classification termed as vector computers. They have super fast processors and I/O systems capable of doing calculations at extremely high speed. One of these computers is the Cray T-90. An alternative architecture is called massively parallel processor systems or a cache machine. Basically, the idea is to put a bunch of reasonably fast processors together in the attempt to get better performance at a cheaper price. And this seems to be working. One of these systems is the IBM Blue Horizon, which was recently installed at the San Diego Supercomputer Center.

In this paper I will analyze the Cray T-90 and Blue Horizon architectures, in an attempt to categorize the requirements for optimal applications in each environment. I will then compare and contrast these two systems to show the tradeoffs between the two systems. And finally, I will discuss the application of various algorithms to these two systems.

*Todd Hansen wrote this for CSE 190 at the University of California, San Diego

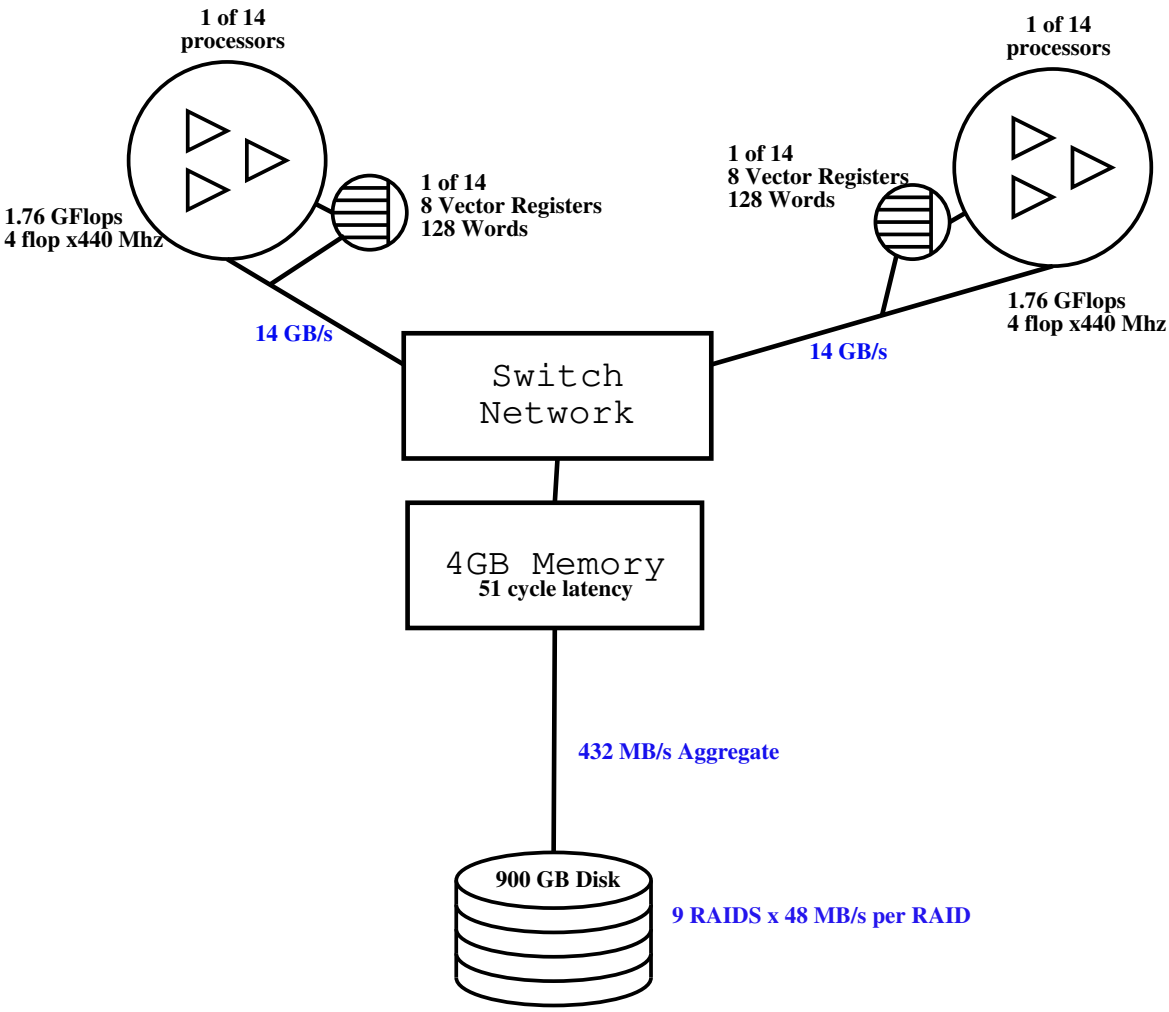
2 The Cray T-90

The Cray T-90 supercomputer developed by Cray Research Inc., is a vector supercomputer with a small number (14) of very fast processors. The system boasts a peak calculation rate of 24.6 GigaFlops. This system has eight vector registers per processor, which allows each processor to sustain a high calculation rate while working on large amounts of data. Each processor has a large bandwidth to memory, but it still must contend with other CPUs for access to a specific port or bank of memory. In most cases it is like each CPU has its own bus to memory. But in about 2% if cases this contention affects performance. Another benefit is the memory bandwidth. Each processor can transfer three words (eight bytes each) to and from memory per cycle. Since the processors can do four FLOP's per cycle (assuming three words/flop), this means that an optimized program should be able to attain a processing speed of over 25% of the peak rate (in reality, with variable reuse it does better than that).

The next information provides more detail on the actual performance of each subsystem. For a picture of how it all fits together see figure 1 on page 3. Much of this information is based on info gathered from reference [1]. Each of the 14 processors on this computer has a peak calculation rate of 1.76 GFlops. In order to achieve this speed each processor has to sustain four floating point operations per cycle. In order to achieve this the Cray computer uses a 128 word (eight Bytes / word) vector register to store the data. There are actually eight vector registers for each processor. You can think of these registers as a sibling to the L1 cache on modern day microprocessors. In the Cray T-90 the data is explicitly loaded into this register while the system does calculations on previously loaded data. This allows the processors to run at full speed as long as data can be kept in the vector register. The T-90 was designed with data flow in mind and can handle transferring three words every cycle. This works great because you can be using one vector register for results and two for input data and in the meantime be loading other registers with data for up coming calculations. In theory you should be able to keep the vector register filled with data and the CPU's humming for as long as you need to. However, there is an initial startup time as you draw the first byte from main memory, this is 51 clock periods or 102 ns. There are 14 processors on the T-90 and they all share the same 4 GB of memory. This is done in such a way that each processor has its own "bus" to the memory. So we have a usable memory bandwidth of 12 GB/s per CPU [3].

So given that there are no major bottlenecks in the memory I/O system, the disk system is

The Cray T-90



Statistics:

Peak: 24.6 GFlops
Word Size: 8 Bytes

Notes: 4 of the 14 processors are used for Disk I/O operations.

Figure 1 - The Cray T-90 Data Flow Diagram

the next area of interest. Four of the T-90 CPUs support disk I/O. They have access to 9 separate RAID arrays at an aggregate rate of 432 MB/s or 48 MB/s per RAID. As you would expect the disk I/O system is considerably slower than the memory I/O system. That is ok though because in most applications the calculations are done in memory. The top speed for the disk subsystem is approximately 8.15 cycles per word written. This assumes that only one CPU is writing to disk. If ten CPUs are creating data for the other four to write to disk, then they can only write one word every 81.5 cycles. However, disk accesses are bursty and so this limit would not have a harsh effect on CPU processing power. Plus if you are doing calculations that fast, you will run out of disk space in just over half an hour (double that time if you are reading data from disk as well).

As an additional side note, the Cray T-90 can support an external network interface at OC-3 line rates or 155 Mbps. At SDSC we have a HIPPI network operating at 122 MB/s. For comparison this is 8 times faster connectivity than most institutions have to their Internet backbone providers. However, this sort of connectivity could be very useful when you are transferring large data sets locally within the same building.

3 The IBM RS/6000 - SP (Blue Horizon)

The IBM Blue Horizon is not at all like the Cray T-90. It was developed by IBM using the same hardware that is used in servers at many large companies. This means that the hardware is inexpensive and standard. The manufacturer claims a peak computing rate of 1.023 TFlops. Some of the key features of this system are the number of processors (1152 total) and the overall distributed nature of the system. Each node has eight processors all competing for the same memory bandwidth (2.2 GB/s total) and some local hard drive space. Any inter-node communication must go over an interconnect to the other nodes. Additionally, there are 12 large disk arrays setup that must be accessed through 12 server nodes. The blue horizon is termed a cache machine because its performance relies heavily on the caching system. For example, instead of writing all the data to disk immediately, put some of it in the extra memory space and we may discard it before we need to write it to disk [2]. It is a very interesting architecture. The main benefit of this architecture is that it is significantly cheaper than the Cray T-90 when compared by sustained Gigaflops / dollar [2]. Another benefit of the IBM architecture is that due to its distributed nature it is quite scalable.

A more detailed description follows, along

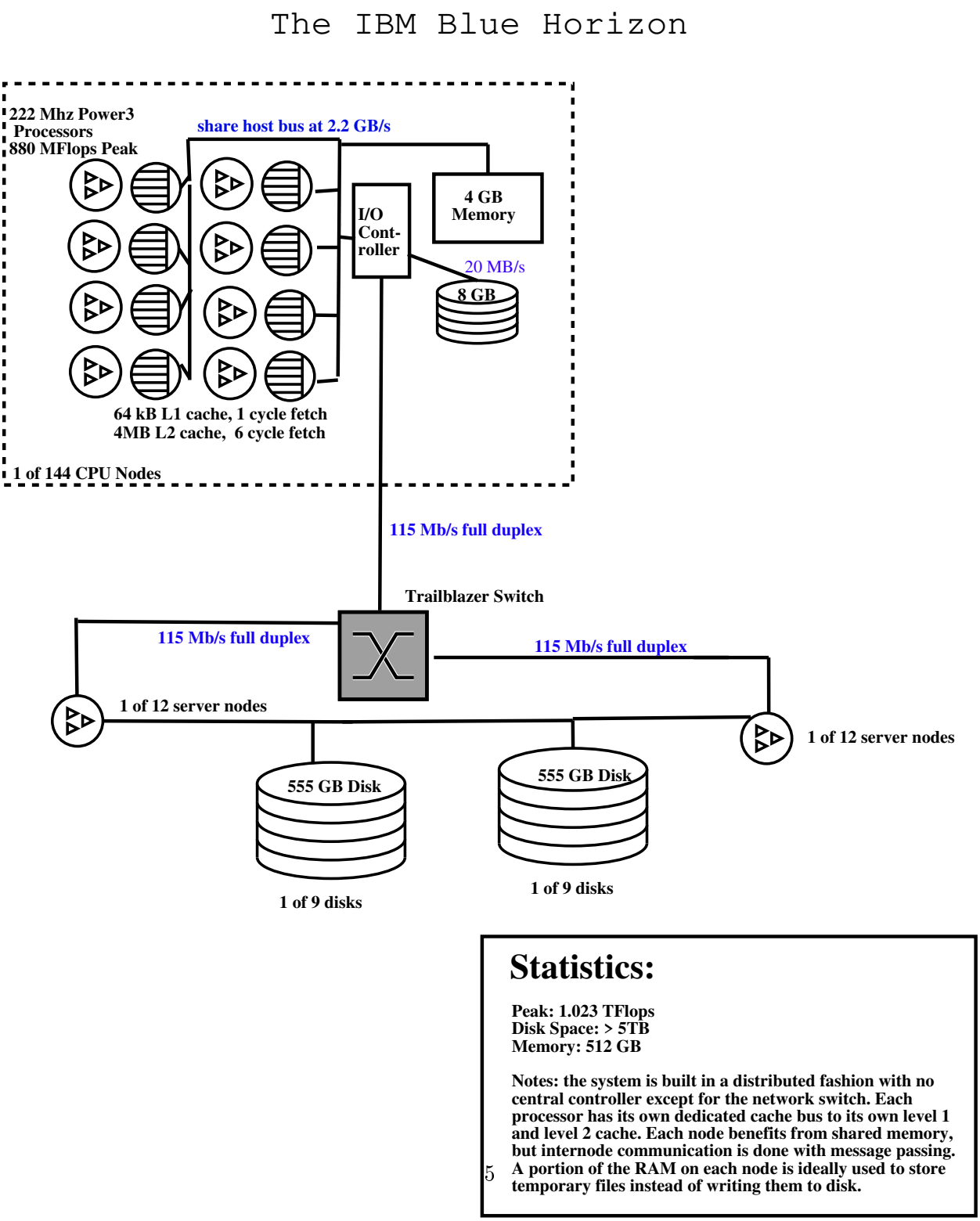


Figure 1: The IBM Blue Horizon Data Flow Diagram

with a data flow diagram (figure 2 on page 5). Most of the information was gathered from [1]. Each of the 1152 CPUs has a peak rate of 880 MFlops. The theory behind this system is to take a large number of reasonably fast processors and combine them into one large computing system. There are numerous scalability issues as well as I/O performance issues that are largely ignored by commodity microprocessor developers. Eight of the processors are grouped into a shared memory node and connected to a central switch. Each node has 4 GB of memory and 8 GB of disk space on a SCSI bus. The shared memory is accessed through a shared bus, so memory accesses will slow down other processors using the same bus. As with the Cray each processor is capable of four floating point operations per cycle. However, instead of a vector register like the T-90, this system has both an L1 and an L2 cache. The first level of cache is 64 kB in size and has an access time equivalent to a single cycle. The level two cache is 4 MB in size and has a delay of 6 cycles per fetch. I don't see any particularly bad bottle necks here, but the shared memory bus is another issue.

The shared memory bus runs at 2.2 GB/s [2]. This is 275 MB/s per processor or 1.238 bytes per cycle. If the CPUs are supposed to run at full pace they need to do approximately 19.4 cycles per read and two writes (an eight byte word). Remember how the T-90 required no cycles between each new read/write set (after it had started the pipeline). There is additional overhead if the system has to gather data from other nodes. Hopefully, this is a rare occurrence because the network is only capable of transferring 115 MB/s in each direction. In addition to that there is a significant latency for transferring data. This is exceptionally bad if you are treating the system as a shared memory system and your working set does not fit into one nodes memory. If there is no packet overhead to the network, you can only send and receive a byte (two bytes total) over the network every 15.2 cycles (if each processor is sending data). This does not take into account the CPU time used to create a data packet, the latency of the response or the use of the memory bus for packet data.

The disk system also uses the network as it is accessed through 12 server nodes. The system houses 5 TB of disk space, but all of the data must first cross the network. If we assume that the 12 server nodes all have a separate network connection, we have a peak throughput of 1380 MB/s (assuming disk writes are instantaneous). If each system is writing to the disk arrays, each CPU can only write 1.190 MB/s. However, the disk subsystem will probably overload before that. Additionally, the 5TB of disk space is broken down into 12 disk arrays, each is

connected to two server nodes.

4 Comparisons

This section discusses the benefits of various subsystems of T90 and Blue Horizon and how they compare to those on the other system.

4.1 Vector Registers vs L1/L2 Cache

The Cray T-90 has eight 128-word vector registers that can be loaded and used explicitly. On the Blue Horizon, there are two levels of caching L1 and L2. L1 cache is 64 KBytes in size and is able to be accessed within 1 CPU cycle or 1.1 ns. This cache is allocated and manipulated by the hardware. Individual programs have no control over its use. The L2 cache is a lot larger and a little bit slower. It is 4 MB in size and takes 6 cycles to access. Both of these caches are connected directly to the CPUs so there is no bus contention between processors. The L2 cache is also allocated and manipulated by hardware with no software control.

On the Cray T-90, a good compiler can take advantage of the vector registers to speed the computations by explicitly loading the data into two of the registers and storing the results in another. Given the extra registers the software can be loading one set of vector registers while it is busy doing computations on the other registers. This allows us to compute at a faster sustained rate.

The cache on the Blue Horizon is controlled by hardware and as a result may not perform in the most optimal manner. Also, the L1 cache is bigger and as a result a good portion of the current data set may be stored closer to the CPU. This means less latency, if we were able to take full benefit of it. Of course, without being able to pull data from memory to the cache as fast as we are consuming it in the processor, we will eventually stall waiting for data from main memory (particularly for data intensive applications). This of course depends on the complexity of the computation. However, with any reasonably simple computation you will not get enough throughput on the bus to sustain the computation. So both the L1 and L2 caches are a big help, but they do not solve the problem of memory I/O. However, under certain applications (where there is a lot of computation on individual data elements) it will work great. This does not mean that the Blue Horizon loses in the battle of the supercomputers. For sufficiently large problems, it has many more CPUs and makes up for this inefficiency by doing more of the work in parallel.

4.2 Memory Size

The Cray T-90 only has 4 GB of memory, where as the IBM Blue Horizon has 4 GB of memory per compute node (576 GB of memory total). This allows the Blue Horizon to store more of the problem set in memory.

On the Cray T-90 there were problems where they did not have enough memory to maintain the working set of the current processes in memory and had to swap to disk [4]. If you recall the working set is the memory that is currently being referenced by each process to do its current computations. When this doesn't fit into memory, the program must be written to explicitly read data into memory as it is needed and to free it when it is no longer useful. While this data is being brought in from disk at least one other memory region of the current working set must be written to disk, so you get into a vicious cycle. By adding more memory to the T-90, they were able to overcome this obstacle. But for sufficiently large problem sizes you still may not be able to store all of the data set in memory.

According to [5] the Blue Horizon has more memory than it will need at its current peak TFlops rating. This allows us to use the extra memory to store more of the data set or even to store some of the results. The benefit of this is that temporary results can be stored in the excess memory and reused, without ever having to be written to disk. This acts like a very large disk cache (350 GB) that will hopefully allow the Blue Horizon to overcome its limitations on disk speed. The data flow model suggests that the Blue Horizon needs 2.5 GB/sec of bandwidth to the disk subsystem. Unfortunately, even with setting up hierarchical raid arrays we could not achieve this performance. So the idea is to remove some of the throughput demands by storing the temporary data in memory. If all works out well, according to the data model for a tera flop system developed in class, we would need throughput to archive disk to be over 50 MB/sec which is definitely achievable. This is all of course based on theory and we will have to wait for some long term use patterns before we can determine the real bottlenecks and performance requirements. For more information on the data flow model for a prototype Tera Flops system, see Figure 3.

4.3 Interprocessor Communication

Interprocessor communication is a problem where any complex parallel algorithm must access data stored by its neighbors and its neighbors neighbors (and so on). If you are trying to decrypt an encode string without the key like the group at distributed.net, then you have very little interprocess communication (IPC), as you are just sending your completion status to a cen-

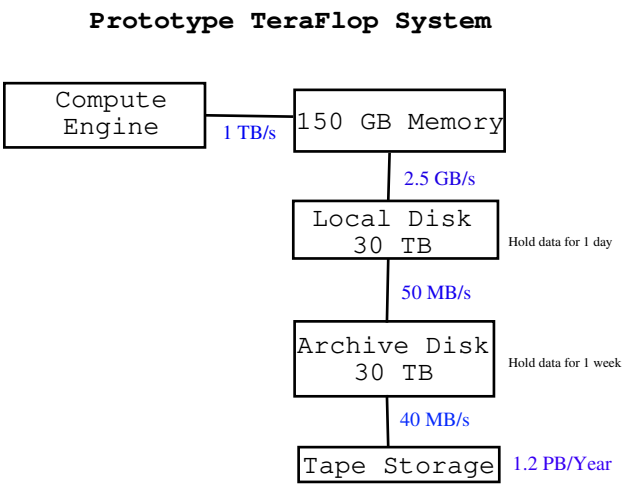


Figure 2: Prototype Teraflop System Data Flow Diagram

tral server. However, when you are doing code like matrix multiplies or Fourier transforms you have a large amount of interprocess communication. These are the type of codes that most people run on these systems. As a result you need to have a pretty advanced IO system to handle this communication.

There are largely two IO systems in use today to handle interprocess communication, one is called shared memory. The other is message passing. In shared memory, all of the variables are accessible on each processor and it is a simple matter of reading the data from memory. Usually some sort of memory locking will take place.

Message passing on the other hand involves sending data/requests over network connections and waiting for the response. This can be quite difficult to implement efficiently and does not have the easy interface that shared memory programming does.

However, it has been proven that a shared memory system can be programmed to act like a message passing system and vice versa. In the Cray T-90, we benefit from having shared memory between all of the processors with each processor having its full bandwidth to memory. This allows us to have a seemless interprocessor communication with only having to wait for whatever locking mechanisms are implemented by the user.

On the Blue Horizon the problem is somewhat different as it is a heterogenous environment of both shared memory on intra-node communication and message passing on internode communication. This means that programming is even more challenging as the programmer must combine the two environments into an efficient program. On could just use a message passing interface and treat the shared memory that way, but then you don't get the benefit of the shared memory speed. Another approach is to derive a shared memory system from Munin [6] that takes advantage of the benefits of intra-node shared memory and optimizes the internode traffic to get as good a message passing performance as possible. If this can be done invisible to the user,

we will benefit two fold. The first is the programmer can worry about the application and spend less time dealing with memory issues. The second benefit is that if a user has less to worry about, they tend to develop more optimal code and as a result they may even get better performance than if they had tried to master both interfaces [7].

The interprocess communication facilities on the Blue Horizon are severely limited by the latency and bandwidth of the node interconnect which runs at 115 MB/s. This is incredibly slow if there is a lot of interprocess communication, like in Fourier transforms. However, because of the number of processors, and the fact that 8 of them have shared memory (where as all 14 have shared memory on the Cray), it will still win out over the Cray in most cases.

4.4 Disk Space

The Cray T-90 has 900 GB's of disk space on nine separate RAID arrays. Four of the Cray's 14 processors support the disk I/O. The aggregate disk bandwidth is 432 MB/s. This means that you must compute 8.15 cycles per word per CPU for each word of data written to disk. So that is only 32.6 cycles per word of data if all of the CPUs are writing to disk. That will be a very difficult bottle neck to break.

The IBM Blue Horizon, as I mentioned above was intended to store most of the temporary data in memory so as to reduce the disk accesses. However, it still has an 8 GB local disk for each node (with a bandwidth of 20 MB/sec per node). In addition to this there are 12 server nodes to nine 555 GB disk arrays connected to the 115 MB/s switch. This means the aggregate network bandwidth that can be dedicated to the central disk is 1380 MB/s. Unfortunately, there has been a lack of disk performance statistics on the actual disk configuration.

Additionally, each of the systems can take the data from their disks and back it up to the tape archive in the same room. As you can see from figure 3, the disk space on the Blue Horizon is not even close to that predicted by the data flow model. However, this should not be an issue as other bottle necks will slow the amount of data generated that must be stored on disk.

4.5 Crash Recovery

Since the Cray T-90 is a shared memory architecture, when one of the CPUs crashes the whole system must be taken down and all of the jobs started again on the working CPUs. On the Blue Horizon a crash is isolated to the node that it occurred on and any processes that were using that node. The rest of the processes and unaffected nodes continue to run while the other node is brought up and the affected jobs are started

again. This is a tribute to the distributed nature of the Blue Horizon design. This information was taken from [8].

5 Performance Estimates for Certain Algorithms

In this section I would like to discuss my performance analysis for certain basic algorithms that one might need to run on these systems. The algorithms I choose were: a matrix multiple and a normalizing algorithm. The normalizing algorithm will be discussed in more detail below.

5.1 Matrix Multiply

In this algorithm I assume that the data is distributed between the processors in a checkerboard manner and is already loaded into memory. I also assume that the data fits in the memory of both the T-90 and the Blue Horizon, which may not be the case depending on the problem size. I also assume the data is evenly dividable on to any number of processors.

On the Cray T90 since all the processors share memory, they can simply do their calculations of the matrix multiplication without sending any messages, except for maybe a semaphore to say they have all finished. The algorithm takes 2 flops per element of the loop (an add and a multiply) [9]. It runs through the loop n^3 times [9] and the code looks highly vectorizable. So if n is the size of one dimension of the square matrix we get the runtime equation of:

$$T_c(seconds) = \frac{2 * n^3}{14 * (1.76 * 10^9)}$$

Since a Blue Horizon node has an equal memory size to the T90's, we can store the entire set of matrices in memory. So all we have to do is combine our changes at the end after the calculation. As a result we get the benefit of up to 144 nodes that can benefit from the same shared memory as the T90, just at a slower rate. The resulting equation for the Blue Horizon is:

$$T_{bh}(seconds) = \frac{2 * n^3}{num_nodes * \left(\frac{2.2*10^9}{2*8}\right)}$$

This is because the Blue Horizon's processing performance in limited by the bus and we are sending two words across the bus for every FLOP, so we get a useful FLOP rate of $\frac{bus_speed}{data_transfered*8}$. The key to the Blue Horizons performance benefits over the T90s is that you can throw a whole lot of nodes at it, where as no one can really afford to throw more than one T90 at any problem. In order for the Blue Horizon to beat the Cray doing matrix multiplication hands down it would have to use 179.2 compute nodes. This is only about 30 more than it currently has, but is still reasonably tractable. The value was determined by the T_{bh} and T_c equations. This is one case where the Cray T90 is actually faster

than the current implementation. But my guess is that any substantially difficult matrix multiply will deal with matrices bigger than 1.33 GBs in size and the T90 will be slowed by disk accesses. So for small enough problems the Cray does actually win.

5.2 2D Normalizing

This algorithm is designed around the idea of normalizing all of the elements of a 2D matrix. The traditional normalizing algorithm is to take the average of all of the elements and to divide the value of each element by that average. The result is then stored in the same location. After this is done the matrix will have a magnitude of one. If effect normalizing is simply removing all of the magnitude data, while keeping the directional data.

The first step in this process is to distribute all of the data from disk to the elements. Doing this on the Cray T-90 will take a minimum of:

$$T_{r_c}(seconds) = \frac{numelements * 8}{4.52 * 10^8}$$

This assumes that each data element is one word in size. On the IBM Blue Horizon reading the data from local disks runs at 20 MB/s per node. So it will take:

$$T_{r_bh}(seconds) = \frac{\frac{numelements}{numnodes} * 8}{20 * 10^6}$$

As you might guess with 144 compute nodes on the IBM Blue Horizon you can throw almost 2900 MB/s at it. It is interesting to see how with one of the most data flow intensive applications I could come up with, the Cray T-90 is already falling behind. I think it is a tribute to the distributed nature of the Blue Horizon.

The next step in the process is calculating the average of all the elements in the local 2D array. The T90 does this at:

$$T_{avg_c}(seconds) = \frac{numelements}{14 * (1.76 * 10^9)}$$

On the Blue Horizon this is:

$$T_{avg_bh}(seconds) = \frac{numelements}{num_nodes * (2.2 * 10^9)}$$

Notice how on the Blue Horizon the performance of this algorithm is dependent on the data flowing across the shared memory bus and not the processor speed. However, even given that the processors spend a good portion of their time waiting for data, if you have a large enough problem and put it on enough processors the Blue Horizon will still win.

The next step is to calculate the average of all the averages (assuming each processor has the same number of elements to calculate). On either of the systems this takes a very short amount of time dependent on the number of nodes involved. I will not calculate this as it is effectively running in a short constant time.

The next step is to normalize each element in the 2D array. On the Cray T-90 we can use the same equation for performance as we did above.

On the IBM Blue Horizon however, we have to take half of the previous equation because we are transferring twice as much data across the shared memory bus.

$$T_{normalize_bh}(seconds) = \frac{2 * numelements}{num_nodes * (2.2 * 10^9)}$$

Finally, something the T90 is good at. Despite having to transfer a lot of data to/from memory, we still get the full performance of the processors. Sadly, in order for the Blue Horizon to beat the T90 all it has to do is use twice as many nodes as it did before. So in this case, we find that for significantly large problems the Blue Horizon still wins. By using the two previous equations ($T_{normalize_bh}$ and T_{avg_c}) for performance, I find that it will take only 22.4 nodes for the Blue Horizon to beat the Cray T-90.

6 Conclusions

Unfortunately, I think the conclusion is obvious. The Blue Horizon is cheaper and faster. It definitely feels wrong because the Blue Horizon's CPUs will spend a long time doing nothing as they wait for data, but by putting so many computers together in parallel, it is only a matter of problem size before the Cray T90 loses. This is unfortunate because the Cray has much faster CPUs that can run at their full rate, but they still can't compete with 1152 processors. Additionally, the memory size is a big issue. On the Cray we can only fit a problem $\frac{1}{144}$ of the size into memory, so we must access the disk a lot more. This is something I didn't take account in the above analysis and I think it will be a very large factor in the future. It is still interesting that the matrix multiply, as configured above, would be faster on the Cray T90 over the existing Blue Horizon system. But as we can see it would only take about 30 extra compute nodes for the performance to be comparable. I am sure that there is more intensive analysis that could be done on this subject, but I am comfortable with the idea that the end result would be the same.

7 References

[1] NPACI System Information Pages: http://hotpage.npaci.edu/

[2] CSE 190 Lecture Notes on the IBM SP

[3] Reagan Moore, a professor at the Univ. Of California at San Diego

[4] CSE 190 Lecture Notes on the Cray T-90

[5] CSE 190 Lecture Notes on Performance Analysis (lecture #1)

[6] Research paper on Munin: http://www.cs.ucsd.edu/classes/sp00/cse121/munin.pdf

[7] CSE 121 Lecture Notes on Munin

[8] Holly Dail's submission to hpcompl@ucsd.edu regarding node failures, the origin of the information contained in this submission was Regean Moore.

[9] Pacheco, Peter S.; Parallel Programming with MPI; Morgan Kaufmann Publishers, Inc.; 1997