# Supercomputing in the South Pacific: performance of a parallel cluster using existing USP facilities

**Imtiyaz Hussein[1], William J Blanke[2]**
[1]*Department of Maths and Computing Sciences, University of the South Pacific, Suva, Fiji*
*hussein_i@usp.ac.fj*
[2]*Department of Maths and Computing Sciences, University of the South Pacific, Suva, Fiji*
*blanke_w@usp.ac.fj*

**ABTRACT**
*This paper presents the details of a parallel computing cluster built using existing computing resources at the University of the South Pacific. Benchmarking tests using the High Performance Linpack Benchmark were done in order to measure the gigaflops (billions of floating point operations per second) ratings for solving large systems of linear equations while varying the number of computers and Ethernet switches used. These tests provided an overall maximum gigaflops rating which allowed comparison of USP's cluster with leading edge clusters from around the world. Efficiency results also provided insight in how improving the existing network infrastructure might improve the performance of USP's cluster and increase its gigaflops rating. Further tests revealed that the number of Ethernet switches used in USP's current network layout is a definite contributor to the low efficiency of the system as a whole.*
***Keywords:*** *Beowulf cluster, parallel computing, Linpack, benchmarking, network topologies*

# 1 INTRODUCTION

In order to do computationally intensive research in any field, whether it be chemistry, physics, mathematics, or any of the other sciences, a powerful computer is needed. Unfortunately such powerful computers typically have large price tags and for that reason high performance computing facilities are not available to researchers locally in the South Pacific region. Purchasing computing time on systems elsewhere in the world is possible, but it is expensive and, in the case of the South Pacific, impractical, given the low bandwidth Internet connections available to communicate with such systems located offshore.

For these reasons, computationally intensive research is difficult to do in the region, even though many locally important issues, such as climate change modeling, could benefit significantly from it. To remedy this problem, given the budgetary and location constraints
outlined previously, this paper details the performance of a high performance computing environment constructed solely from existing, unused (and therefore free) computer resources at the University of the South Pacific. In its present form, it is able to solve a wide range of computationally intensive problems in a multitude of fields and disciplines and will hopefully foster further parallel computing research at the University.

# 2 METHODS

To build the computing cluster with a zero dollar budget, already existing, but underutilized, high performance desktop computers at the University are teamed with several free open source software packages. The University of the South Pacific currently has 44 state of the art desktop computers donated by the Japanese government which are being utilized in labs on the Laucala (Suva, Fiji) campus at the Department of Maths and Computing Sciences. Each computer is equipped with a 2 gigahertz Pentium 4 processor, 256 megabytes of RAM, and two 40 gigabyte hard drives.

Currently, these computers are turned off and the labs are locked during night time hours (8:00 PM till 8:00 AM) primarily due to security and safety concerns and are completely unavailable for student use. This project makes use of these 44 machines and their 12 hours per day of unused time-in essence 528 hours of computing time per day to perform research computations by using parallel processing.

Parallel processing means linking together two or more computers to jointly solve a computational problem. Originally such computers were expensive and custom built. However, an increasing trend since the early 1990s has seen the movement from expensive and specialized proprietary parallel supercomputers towards networks of PCs or workstations (Becker *et al*. 1995).

Clusters of homogeneous or heterogeneous PCs or workstations working together to solve a problem are rapidly becoming the standard platforms for high-performance and large-scale computing. Known as Beowulf clusters (Sterling, 2001) these systems are built using very affordable, low-cost, commodity hardware such as Pentium PCs, fast Local Area Networks (LANs), and standard free open source software components such as the UNIX operating system (Hoffman and Hargrove, 1999), MPI, and PVM parallel programming environments.

A common approach to parallel programming is to use a message passing library, where a process uses the library calls to exchange messages (information) with another process. This message passing (Pacheco, 1996) allows processes running on multiple processors to cooperate in solving problems. When used correctly, it allows the group of computers to work as one.

MPI (Gropp *et al*. 1996) is one such message passing interface standard. It is available on a wide variety of platforms, ranging from custom built massively parallel systems to networks of workstations. There are several implementations of MPI in existence. MPICH, used in this paper, is freely available from Argonne National Laboratory and Mississippi State University.

A job management system is the software component that ensures the balanced use of cluster resources. It maximizes the delivery of resources to different computing

jobs, given competing user requirements and local policy restrictions. The software monitors the state of the cluster, schedules work, enforces policy and tracks usage. For this paper, the free open source Maui job management system and PBS scheduler (Feitelson and Rudolph, 1995) were used.

For measuring the performance (benchmarking) of the parallel computing cluster, the High Performance Linpack Benchmark (Petitet *et al*. 2000) was used. Linpack (Dongarra *et al*. 1986) is a general purpose library for solving dense systems of linear equations in double-precision (64 bit) arithmetic using the Gaussian elimination method. In addition to its value for solving systems of linear equations, Linpack has also found use in measuring the performance of supercomputers worldwide. The High Performance Linpack Benchmark keeps track of execution time and then divides this into the number of floating point operations that it performs to get a gigaflops (billions of floating point operations per second) rating. The gigaflops rating is the basis for the performance graphs in this paper.

# 3 RESULTS

The theoretical peak performance (P) of any parallel system is calculated by equation 1:

$$P = SCFH \qquad (1)$$

In this equation, S is the number of computing nodes, C is the number of CPUs per computing node, F is the number of floating point operations per clock cycle, and H is the clock rate. Thus for 44 nodes using single 2 gigahertz processors, and assuming one floating point operation per cycle, the peak performance would be 88 gigaflops. Theoretically, the USP cluster should be able to reach that number. However, in practice this is never achieved because of efficiency issues.

The percentage of time that a group of computers spend doing actual work on the problem (in contrast to the time they spend idle waiting for messages to arrive or for other computers to finish a required computation) in comparison to the theoretical peak performance is known as efficiency. The efficiency of a cluster at any point in time is the measured gigaflops rating $R_{max}$, obtained by running the Linpack benchmark, divided by the theoretical peak performance as shown by equation 2.

$$E = R_{max} / P \qquad (2)$$

As an example, if a computer system has twice as many nodes as another system but overall it is only half as efficient, both systems will be equivalent in their gigaflops rating. Thus efficiency is extremely important to consider.

There are many things that can cause efficiency issues. If an algorithm is poorly designed, increasing the number of computers involved in a calculation will usually result in much lower efficiency. Such an algorithm would be said not to be scalable. A highly scalable algorithm should show little reduction. Linpack is very well designed and considered highly scalable (Dongarra *et al*. 1986).

Communication problems can also cause efficiency reductions. In a parallel computing cluster, the way that the network connections are constructed and the speed and latency of those connections bear heavily on how well the entire system performs. If messages take especially long to travel from one computer to another that will likely impact the performance of the entire group as a whole.

The existing computer laboratories used at USP consisted of 44 computing nodes and three 100 megabit Fast Ethernet switches located in two different rooms, the MaCS "Small Lab" and the MaCS "Large Lab". Figure 1 shows the setup of the entire cluster. In essence three sub-clusters, each linked by a single switch exist within this group of computers. The first switch connects 16 computing nodes which is located in the MaCS "Small Lab". The second switch connects twelve computing nodes which are located in the MaCS "Large Lab" and finally the third switch connects the remaining sixteen computing nodes in the MaCS "Large Lab". Since this configuration cannot be changed, a looming question is how these different network topologies will affect the overall efficiency and thus the overall performance of the system.

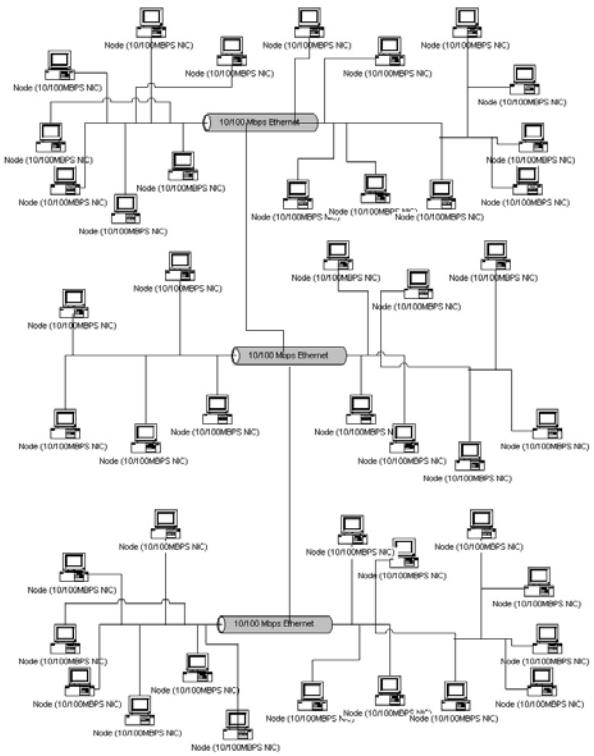


**Figure 1.** Existing USP MaCS computer lab arrangement

Given that the Linpack algorithm is considered highly scalable, to determine the efficiency impact of increasing the numbers of computing nodes and also the numbers of Ethernet switches, several High Performance Linpack Benchmark runs were made with varying numbers of computing nodes (and therefore involving varying numbers of Ethernet switches). As the benchmark dictates, the highest performing problem set solved is quoted for the gigaflops rating.

**Table 1.** Linpack Benchmark Results

| Nodes | Switches | N(1/2) | $N_{max}$ | $R_{max}$ | Eff. |
|---|---|---|---|---|---|
| 4 | 1 | 4000 | 14000 | 4.891 | 61% |
| 8 | 1 | 5000 | 20000 | 8.97 | 55% |
| 12 | 1 | 8000 | 25000 | 12.41 | 52% |
| 16 | 1 | 9000 | 28000 | 16.24 | 51% |
| 28 | 2 | 14000 | 36000 | 20.22 | 36% |
| 44 | 3 | 22000 | 48000 | 24.20 | 28% |

Using the 16 computing node "Small Lab" cluster, runs were done of 4, 8, 12, and 16 computing nodes using one Ethernet switch. The results are shown in table 1. N represents the size of the linear equation system that was solved for the given peak gigaflops rating. N(1/2) represents the size of the linear equation system that was solved for half this rating. $R_{max}$ is the maximum gigaflops rating found for a series of problem sizes. For example, the 16 computing node run solved a 28,000 by 28,000 number matrix averaging 16.24 billion floating point operations per second with an efficiency of 51%. Essentially it spends half the time working on the problem and the other half communicating or waiting for data. As per N(1/2), it solved a 9,000 by 9,000 number matrix averaging 8.12 billion floating point operations per second.

One interesting feature to note on this chart is that as the number of computing nodes are increased the gigaflops rating also increases. This would be expected with more computers you can calculate more numbers. Another interesting feature is that the efficiency is highest using 4 computing nodes at 61% but then drops off when using 8 computing nodes to 55%. It levels off at 52% and 51% using 12 and 16 nodes respectively. Communication delays are relatively low with only 4 computers involved. As the numbers of machines increases, delays do as well and efficiency suffers. However, this is tempered by the scalability of the algorithm with the higher computing node counts.

In order to use more than 16 computing nodes at the USP computer labs, another switch would need to be employed. For the next runs, the two MaCS "Large Lab" sub clusters were joined together to create a cluster consisting of 28 computing nodes and two switches. The results of the benchmark is shown in table 1. Gigaflops numbers have certainly increased-up to 20.22 billion floating point operations per second for a matrix of size 36,000 by 36,000. However, efficiency has definitely suffered at 36%. Two thirds of the time, the cluster is not doing productive work.

To go beyond 28 computing nodes, all three sub clusters, the 2 MaCS "Large Lab" sub clusters and the MaCS "Small Lab" sub cluster, will need to be joined together. This will create a cluster of 44 machines across 3 Ethernet switches. The results from benchmarking this cluster are shown in table 1. Again, the gigaflops rating rises to 24.20 but much more slowly than before. As shown in figure 2 even though we have significantly increased the number of computing nodes the gigaflops performance has not risen linearly. This is mainly due to the paltry 28% efficiency of the system. Almost three quarters of the time, the cluster is not doing productive work.

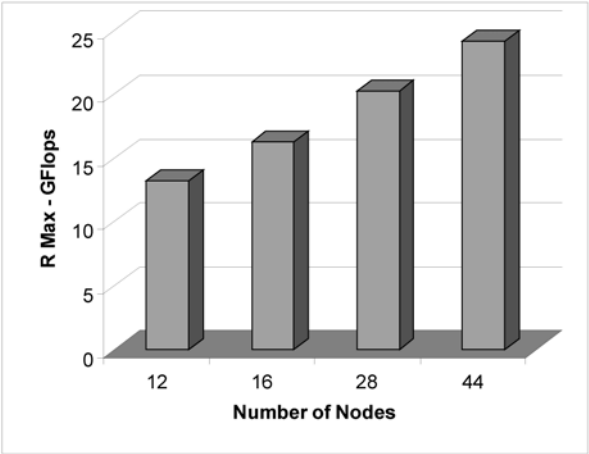


**Figure 2.** Comparison of cluster performance against different number of nodes.

Now that we have the peak gigaflops rating of 24.20 over 44 computing nodes, we can compare the performance of the cluster at USP with other supercomputers from around the world. The Top 500 project (Meuer *et al*. 2004) was started in 1993 to provide a reliable basis for tracking and detecting trends in high-performance computing. A list of the sites operating the 500 most powerful computer systems is assembled and released twice a year. The best performance on the Linpack benchmark is used as a performance measure for ranking computer systems.

The third fastest computer in the world (the two fastest are not clusters in the same sense as USP's system) according to the Top 500 is a cluster consisting of 1,100 Apple Computer Power Mac G5 desktops at Virginia Polytechnic Institute and State University (Showerman and Enos, 2004). Each node on the cluster is a Power Mac computer with dual 2 gigahertz PowerPC 970 processors made by IBM, 4 gigabytes of memory and 160 gigabytes of storage. They are connected together using Infiniband interconnect technology and Gigabit Ethernet switches. This Apple cluster gained its third place with a 10.3 teraflops ranking, and became only the third computer in the world to achieve a performance of more than 10 teraflops.

10.3 teraflops is 10.3 trillion floating point operations per second which means Virginia Tech's system is roughly 425 times faster at solving systems of linear equations than USP's cluster. However, it uses 50 times more CPUs, has much more hard disk space and memory, and utilizes a much better networking infrastructure. Better networking interconnections should be able to improve USP's performance results as well. If the efficiency of the cluster can be improved then the performance of the cluster will increase even if no additional CPUs are added to the system.

Because Linpack is highly scalable, the network and its topology of multiple switches seem to be causing the efficiency problem of the previous USP results. To prove this experiment was devised in order to compare efficiency. Three runs of 12 computing nodes each were benchmarked. The only difference between them was the

number of switches used. One run used 12 computing nodes all on one switch. Another used 6 computing nodes each on two switches for 12 in all. Finally the third used four computing nodes each on three switches again for 12 in all.

In figure 3 we see a drop in the efficiency of the twelve-node cluster as these nodes are distributed over switches. First a efficiency of 52% over 1 switch is achieved, which is consistent with results from table 1. Then when these twelve nodes are distributed between 2 switches (six nodes each), the efficiency decreases from above the 50% mark to about 42%. When comparing this result to results from table 1 in which the efficiency of 28 nodes was measured with two switches, results are quite similar. The minor difference of a few percentage points could be said to be due to the increased number of nodes (Luo *et al*. 2002). Finally when the same twelve nodes were distributed among the three switches, a 34% efficiency was measured. This again is quite comparable to results from table 1 where 44 nodes were distributed among the 3 switches.

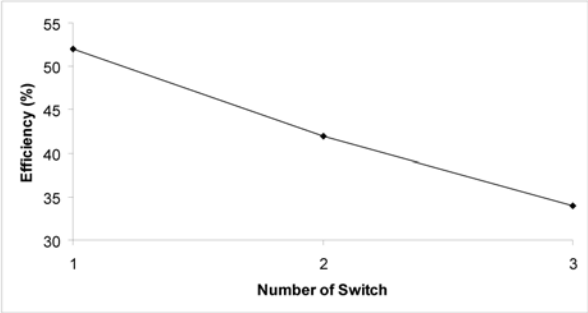


**Figure 3.** This graph shows the efficiency of the cluster when number of nodes are fixed but the number of interconnecting switch changes.

# 4 CONCLUSION

This paper presents a 44 node parallel computing environment with a peak performance rating of 24.20 billion floating point operations per second (gigaflops). It has been constructed at zero cost to the University using unused computing time from already existing USP computing resources.

From the benchmark results presented, it can now be said that inter-switch communication plays a vital role in the performance of a parallel cluster such as the one created at USP. Improving networking hardware to reduce the switch count and increase switch performance should increase the efficiency of the system and thus increase the overall gigaflops rating. Research also shows that moving from Fast Ethernet to Myrinet (Boden, 1995) or Gigabit Ethernet (Innocente *et al*. 2000) provides faster performance results due to data throughput increases and communication latency decreases. However, as USP is lacking this infrastructure it was not possible to test these cases in this paper.

By opening the cluster to projects from all departments in the University, it is hoped that a large number of students and faculty will gain experience in running the computing cluster. This critical mass of students, and the involvement of numerous other faculty in the projects of those students, will ensure that the cluster will continue to operate at peak performance. Also, since the parallel computing cluster can be controlled exclusively via a network connection, other campuses of the University of the South Pacific are able to easily have access to the facilities via USPNet, the satellite network linking other island countries to USP's Laucala campus in Fiji.

# ACKNOWLEDGMENTS

The authors are very grateful to Nitesh Nand at the USP Department of Maths and Computing Sciences and to John Isles and Simon Greaves at USP Information Technology Services for their technical assistance.

# REFRENCES

1. Becker, D.J., Sterling, T., Savarese, D., Dorband, J.E., Ranawak, U.A. and Packer, C.V. 1995. Beowulf : A parallel workstation for scientific computation. In *Proceedings, International Conference on Parallel Processing*.
2. Boden, N.J. 1995. Myrinet - a gigabit-per-second local-area network. *IEEE-Micro*, **15(1)**, 29-36.
3. Dongarra, J. J., Bunch, J., Moler, C. and Stewart, G.W. 1986. *LINPACK User's Guide*. SIAM, Philadelphia, PA.
4. Feitelson, D. and Rudolph, L. 1995. Parallel job scheduling : Issues and approaches. In *Proceedings of the 1st Workshop on Job Scheduling Strategies for Parallel Processing*.
5. Gropp, W., Lusk, E., Doss, N. and Skjellum, A. 1996. A high-performance, portable implementation of the message passing interface standard. *Parallel Computing*, **22(6)**, 789-828.
6. Hoffman, F.M. and Hargrove, W.W. 1999. Cluster computing: Linux taken to the extreme. *Linux Magazine*, **1(1)**, 56-59.
7. Innocente, R., Corbatto, M. and Cozzini, S. 2000. A PC cluster with high speed network interconnects. In *CAPI Calcolo ad Alte Prestazioni in Italia CILEA*.
8. Luo, X.O., Gregory, E., Xi, H., Yang, J., Wang, Y. Lin, Y. and Ying, H. 2002. High performance beowulf computer for lattice qcd.
9. Meuer, H., Strohmaier, E., Dongarra, J.J. and Simon, H.D. 2003. Current top 500 list.
10. Pacheco, P. 1996. *Parallel Programming with MPI*. Morgan Kaufmann Publishers, San Francisco, CA.
11. Petitet, A., Whaley, R., Dongarra, J.J. and Clearly, A. 2000. Hpl - a portable implementation of the high performance linpack benchmark for distributed-memory computers.
12. Showerman, M. and Enos, J. 2004. Terascale cluster.
13. Sterling, T. 2001. *Beowulf Cluster Computing with Linux*.