High Performance Supercomputing using Infiniband based Clustered Servers

M.J. Johnson A.L.C. Barczak C.H. Messom

Institute of Information and Mathematical Sciences
Massey University
Auckland, New Zealand.

Email: m.j.johnson@massey.ac.nz, a.l.barczak@massey.ac.nz, c.h.messom@massey.ac.nz

Abstract: This paper introduces the hardware architecture for the BeSTGRID (Broadband Enabled Science and Technology GRID) supercomputer system. This system is a high performance cluster supercomputer with infiniband interconnect. The system has been benchmarked for compute and file IO and is shown to be scalable. The infiniband interconnect provides high bandwidth, low latency links between both compute nodes and storage system which gives good parallel scalability. The linpack benchmark scales linearly up to the full 208 processors. Although infiniband is not a commodity interconnect, it is shown that for both parallel compute scalability and storage IO scalability it is a significant improvement on gigabit Ethernet interconnects.

Keywords Cluster supercomputing, lustre file system, scalability, infiniband interconnect.

1. INTRODUCTION

Cluster supercomputing has been well established in New Zealand with the helix cluster supercomputer hosted in Massey University entering the top 500 supercomputer list in 2002 with a Linpack performance of 235 Gflops [1,2]. These older systems used workstation/ server commoditised system components and commoditised gigabit Ethernet for high speed interconnect. These systems had good scalability for compute tasks, but the latency associated with gigabit Ethernet ensured that most algorithms would not scale much beyond four to eight processors.

The latest quad core processors from both Intel™ and AMD™ mean that 8 core commodity systems are becoming available, with the price of memory reducing as well a server system with 8 processor cores and 16GB RAM (Random Access Memory) is within reach of most research budgets. As always with the progress in computer systems, work loads that could not have been supported five years ago on the most advanced machines are now significantly more affordable. Single systems with up to 8 processor cores, cache coherent architectures, large cache and good memory bandwidth mean that many workloads easily scale up to 8 processors. Scalability beyond 8 processor cores still depends on the performance of the networking interconnect.

Infiniband interconnects with high bandwidth low latency provides the possibility to scale beyond 8 processor cores, however this requires the use of a message passing framework to communicate between the nodes, while a shared memory approach can be adopted on a single system for workloads that require less than 8 processors.

The BeSTGRID project, a project from the Tertiary Education Commission (TEC) Innovation and Development Fund (IDF), commissioned a cluster supercomputer using the latest high speed processors and interconnects. A 208 processor node compute cluster system was developed with 16 GB RAM per server (8 processors), while a high bandwidth infiniband DDR (Double Data Rate) 4x (20 Gbps) low latency (2.7μs) interconnect was adopted. Although the line speed of the system is rated at 20 Gbps the node to node raw infiniband performance was 11Gbps. This performance gap represents the IO bottleneck within the system to and through the PCI (Peripferal Component Interface) express to the infiniband HCA (Host Channel Adaptor).

This paper discusses the design and scalability of the BeSTGRID supercomputer, presenting performance scalability results based on well known benchmarks such as Linpack, netperf, MPI ring test etc, as well as comparing these with legacy systems using gigabit Ethernet network and systems that have a shared memory single system image.

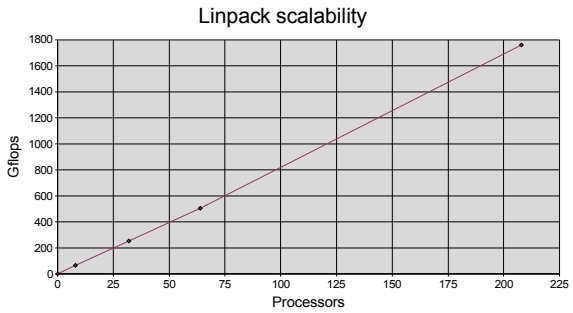

Figure 1. Linpack Scalability of the BeSTGRID cluster

2. LINPACK BENCHMARKING

2.1 Linpack benchmark

The Linpack benchmark measures the performance of a distributed memory computer while solving a dense linear system in double precision. Several factors determine the performance achieved on this benchmark, but the bandwidth and latency are a significant component. High bandwidth and low latency ensure that the communication is not a bottleneck in the algorithm. The algorithm tends to be more efficient for larger problem sizes and since the optimal performance is when the problem fits into available system memory, the larger the memory of the machine the better [3, 4].

Each of the nodes in the BeSTGRID cluster contains 16GB of RAM, so this provides a limit to the maximum problem size. Since we require 8 bytes per double precision floating point number, the maximum theoretical matrix size is 26*16GB/8 = 236,293 by 236,293. However since there is operating system overhead and infiniband buffer overhead, a more realistic size using 80% of the memory gives a figure of 190,000 by 190,000. The theoretical peak performance (Rpeak) of the BeSTGRID cluster (the maximum Gflop rating that the machine is guaranteed never to exceed) is 2.2 Teraflops, however this is not a useful measure since it will not be achieved even if every machine is working independently.

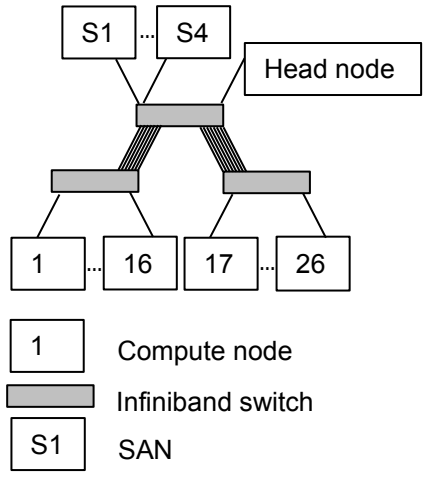

Figure 2. Fat tree structure of infiniband network

2.2 Linpack performance results

The peak Linpack performance (Rmax) using a problem size of 210,000 was 1.758 Teraflops. The performance of the BeSTGRID cluster on the Linpack benchmark vs processors (and memory) shows that the system scales almost linearly (see figure 1), this is due to the high bandwidth, low latency switching. The infiniband switches each have 24 ports and as seen in figure 2 the fat tree structure allows up to 40 nodes to be connected with only 1:2 oversubscription. This means that up to 320 processors can be used with at most three high bandwidth low latency hops between processors. The almost linear increase in performance versus number of processors for the BeSTGRID cluster indicates that the architecture is scalable and so additional nodes can be added to the BeSTGRID to increase the size of problems solved as well as the speed at which they will be solved. The fat tree structure can be further expanded to allow the system to scale to more than 320 processors, however for very large clusters a larger switch solution is recommended. Using a complex fat tree structure that has significant oversubscription will require a communication pattern be established in the algorithm and the jobs allocated to the nodes with the required bandwidth and latency between nodes with high communication.

3. LUSTRE FILE SYSTEM

3.1 Cluster File Systems

Cluster file systems have been attracting significant research interest with the ever diminishing storage IO performance of computer systems versus cpu speed. Cluster compute systems tended to have local disks that offered the potential to scale storage IO bandwidth if access could be striped over multiple disks in multiple machines rather than just the local system. Even with the rise of diskless cluster systems, the potential to have multiple SANs attached to the cluster system has raised the need for a parallel scalable cluster file system.

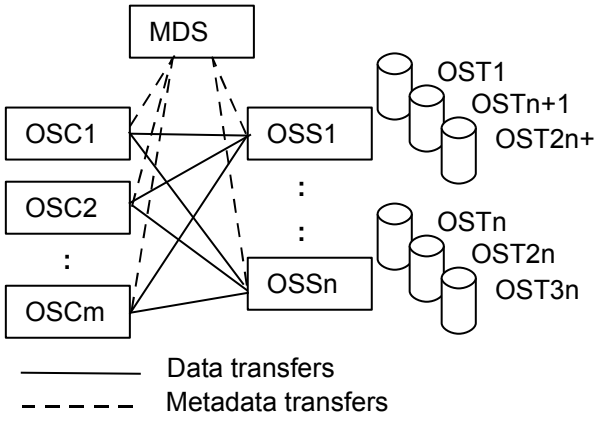

Figure 3. Lustre file system structure with Meta Data Server (MDS), Object Storage Servers (OSS), Object Storage Targets (OST) and Object Storage Client (OSC)

There are several cluster file systems that could be implemented, GFS (Global File System), OCFS (Oracle Cluster File System) etc, but Lustre has been selected for the BeSTGRID cluster [5-7]. Lustre although very tedious to implement due to the significant modifications required to the kernel to support the file system is gaining popularity due to its performance scalability. Lustre is the file system recommended by SGI™, and is also implemented by Cray™ and is used by IBM™ in the BlueGene systems.

3.2 Lustre File Systems

The Lustre file system implements a distributed cluster file system by abstracting RAID (Redundant Array of Independent Drives) enabled disk mount points as Object Storage Targets (OST) that are managed by Object Storage Servers (OSS). The clients, Object Storage Client (OSC) access the file space by interacting with a Meta Data Server (MDS) to obtain locks and locations of the data. Figure 3 illustrates the structure and data transfer topology of the lustre file system. The single MDS represents a performance bottleneck as a well as a single point of failure, multiple MDS based systems are available but only supported for shared disk hardware.

3.3 Lustre File System Performance

The lustre file system was installed over the 4 SANs for the BeSTGRID system. The SANs each have 16 500GB disks providing a raw storage capacity of 8TB each for a total of 32 TB. Each server manages 2 OSTs created by striping over 8 disks each using RAID 6, giving a 6/8 redundancy. That is the system can recover from four disks failing out of 16 disks in each server. Each SAN provides 4 Gbps raw disk IO performance, striping over all SANs using 1 client with lustre file system gives approx 8Gbps. The theoretical maximum striping over the 4 SANs are 16Gbps so multiple clients must access the SANs simultaneously to saturate the disk bandwidth.

4. PARALLEL SCALABILITY, LATENCY AND BANDWIDTH

Although the Linpack benchmark shows that the system architecture is very scalable, it does not show how other algorithms that do not have such good scalability will perform on this architecture. For this the Latency and bandwidth of the system must be investigated. Although we have the rated infiniband DDR 4x 20 Gbps bandwidth and 2.7µs latency, these figures do not reflect what can actually be achieved in most situations. As discussed in section 1 only a raw infiniband line speed of 11 Gbps has been achieved due to IO bottlenecks from the cpu to the PCI express interface to the Infiniband HCA. The latency figures do not take into account the fat tree structure which introduces additional latencies when the communication is over multiple hops.

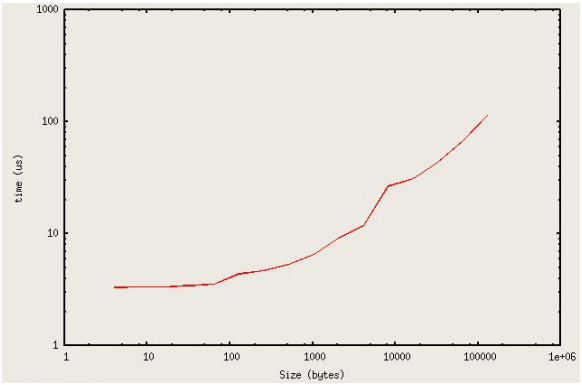

Figure 4. Bandwidth scalability versus message size.

In addition most algorithms will implement a protocol stack above the raw infiniband interconnect that increases the latency and decreases the achievable bandwidth. Most parallel algorithms will use an mpi stack which adds additional overhead to the raw infiniband performance. Figure 4 shows the network bandwidth scalability of the point to point infiniband network. Small messages take 3.2µs just over the 2.7µs latency to be delivered, while larger message sizes do not take significantly longer. Message about 1KB in size can be delivered in less than 10µs which is significantly better than gigabit Ethernet technology which has higher latencies and lower bandwidth.

4.1 MPI Scatter Performance

Some of the key components of a message passing library are the scatter, gather and barrier operations [8]. The scatter operation passes a message from a single source to the target nodes, while the gather process acts in reverse gathering the results from multiple nodes to a single node. This scatter algorithm is normally implemented as a tree communication pattern, meaning the head node sends a message to one node, then the head node and this child nodes sends a message to two other nodes (1 node each) in a growing exponential tree. With this structure scattering data to a smaller number of nodes is a faster operation. This can be seen in figure 5 as the time to complete the scatter operation increases as the size of the communication group increases, this trend exists over all message sizes.

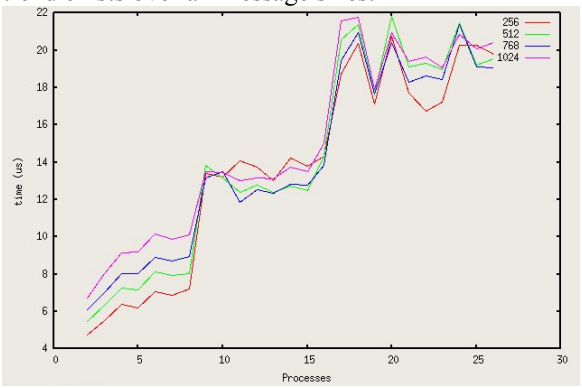

Figure 5. Scalability of the Message Passing Interface (MPI) scatter algorithm using different message sizes

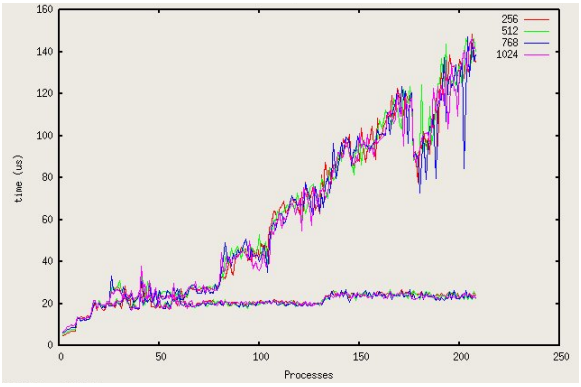

Figure 6. Scalability of the MPI scatter algorithm over large processor counts using two different processor layouts.

Figure 6 shows the performance of the MPI scatter algorithm over larger message groups as is the case when there is a communication node for each processors core rather than one for each server. The two groups of lines on the graph are due to whether the communication groups are ordered over the single machines or are scattered between the machines. When the communication groups are ordered within the machine, then a large proportion of the communication occurs within a single machine, which is faster than communicating between the machines.

4.2 Point to point latency

In general given enough bandwidth (which for small messages this is always the case), the most significant impact on system performance is the latency in the system. This can be tested with a multipoint ring test [9] that can ensure that there are no communication patterns that can introduce excessive latencies. Figure 7a shows the MPI ring test for the 26 nodes of the BeSTGRID cluster. The smooth gradient of the ring test shows that the point to point latency is almost constant between the nodes and is very low. In addition the barrier operation, where all the nodes are waiting for the barrier before continuing processing is almost vertical, again showing the inherently low latency in the system. Figure 7b shows the test for 128 processors spread across 26 nodes of the cluster, similarly the point to point latency is very consistent and the barrier implementation is very good (almost vertical line for the barrier to receive transition)

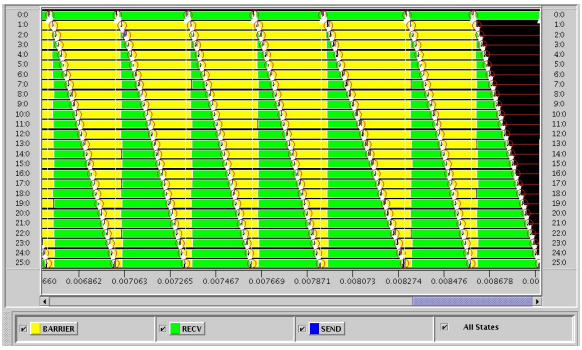

Figure 7 a. Ring test results for 26 nodes

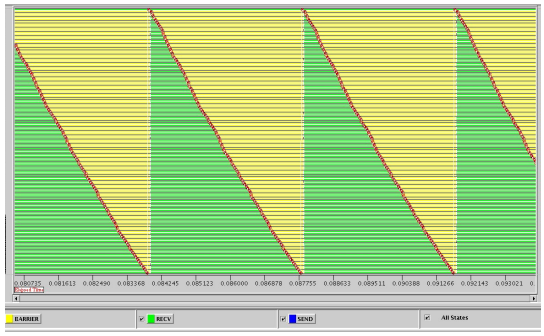

Figure 7 b. Ring test results for 128 processors spread over 26 nodes.

4.3 Point to point bandwidth

Although latency is a key issue for small message sizes, bandwidth plays a more significant role as the message size increases. Although raw bandwidth figures can be derived for communication over the infiniband network performance using the MPI communication operations are of more interest in parallel programming scenarios. Message passing libraries have blocking and non-blocking communication functions. Non-blocking calls allow the program to work on other processing tasks while the communication operating is in progress while the blocking operations force the process to wait for the communication to complete. Figures 8 a-d show the performance of the communication for non-blocking MPI functions while Figures 9 a-d show the performance for the blocking MPI functions.

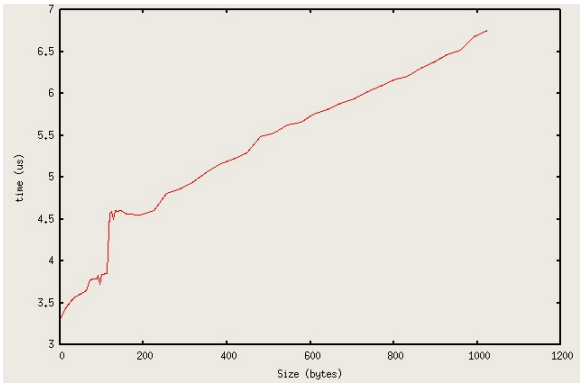

Figure 8 a. MPI Non-blocking Point to Point transfer times for Small Message Size

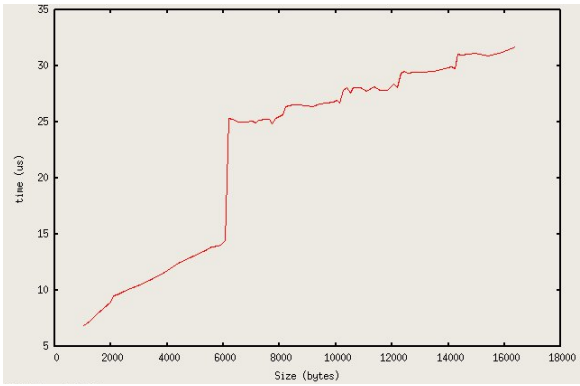

Figure 8 b. MPI Non-blocking Point to Point transfer times for Medium Message Size

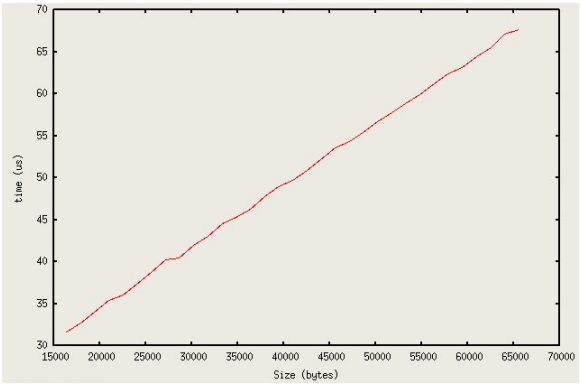

Figure 8 c. MPI Non-blocking Point to Point transfer times for Large Message Size

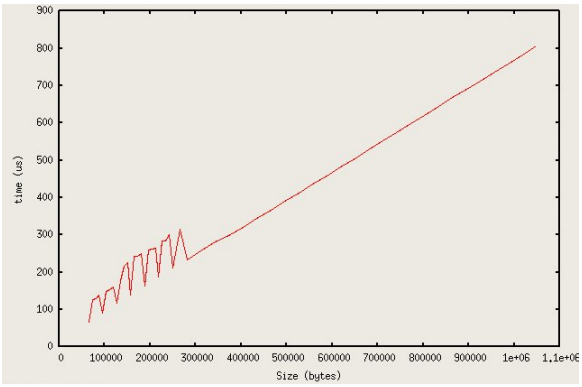

Figure 8 d. MPI Non-blocking Point to Point transfer times for Very Large Message Size

Ranging from small message sizes through to medium, large and very large message sizes, the MPI communication functions scale very well. The gradient of the graph (the inverse of the effective bandwidth) does not change as the messages become larger. For very small and medium message sizes we see some step transitions near message sizes of 128 bytes and 6000 bytes, these are related to cache and buffer size issues. The infiniband interconnect copies the message from memory to the HCA and as intermediate hardware and software buffers are exceeded performance is degraded. An additional artefact is seen for very large messages which were reliably recreated for both blocking and non-blocking calls over multiple experiments.

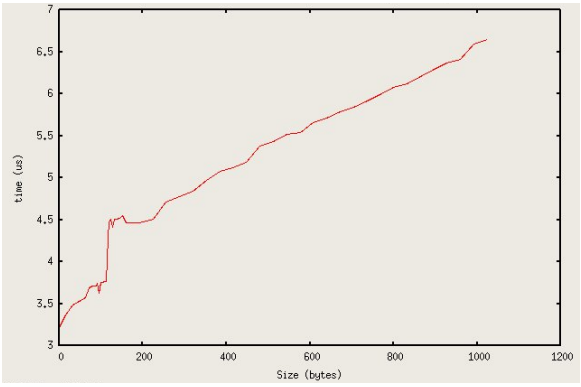

Figure 9 a. MPI Blocking Point to Point transfer times for Small Message Size

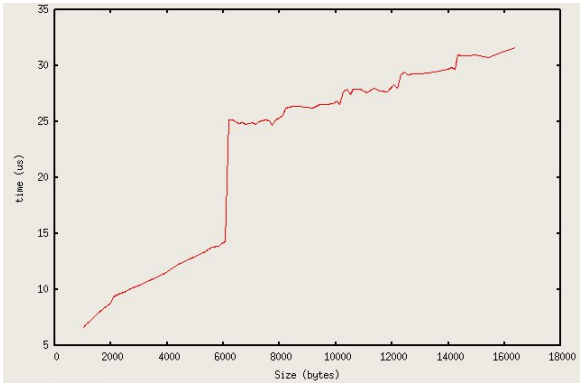

Figure 9 b. MPI Blocking Point to Point transfer times for Medium Message Size

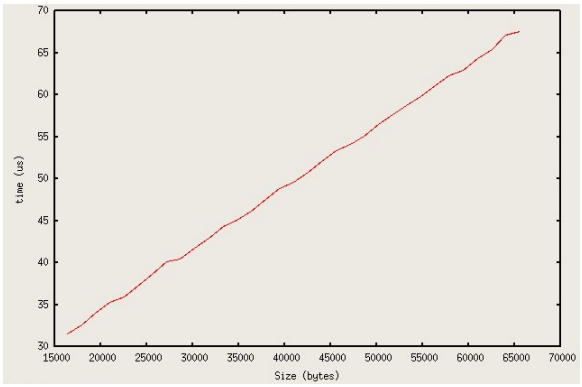

Figure 9 c. MPI Blocking Point to Point transfer times for Large Message Size

These artefacts are likely due to the message size chosen in the experiment, some message sizes providing a multiple of the internal buffers used, while other buffer size just overflowing a multiple of buffers, effectively reducing the efficiency of the MPI function over the interconnect.

Comparing the performance of the blocking and non-blocking MPI functions we can see there is little difference between them, this shows that the non-blocking implementation does not have significant overhead. A task which can execute some computation while the communication is in progress will not have any significant effect on the communication performance.

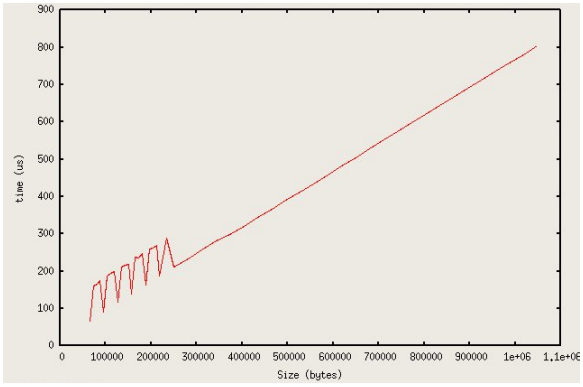

Figure 9 d. MPI Blocking Point to Point transfer times for Very Large Message Size

4.4 Bisectional bandwidth

The Bisectional bandwidth, or the effective average bandwidth between two halves of the cluster is theoretically 10Gbps. This is the case due to the 2:1 oversubscription in the fat tree switch structure reducing the 20Gbps theoretical bandwidth down to 10Gbps when the two halves of the cluster (16 nodes each) that are communicating with each-other have to communicate via the (8) inter switch links. This is clearly close to the actual 11Gbps achieved when communicating on a single switch, so the over subscription is not expected to reduce overall performance significantly.

5. CONCLUSIONS

This paper has presented the design and performance characteristics of an infinband based cluster supercomputer. The system has shown to be extremely scalable achieving 1.758 Teraflops peak performance over 208 processors, linearly scalable from 68 Gflops for 8 processors. Although the cluster does not have a single system image model it is seen that using the mpi programming model, parallel algorithms can easily implemented that can scale efficiently up to large processor counts. The lustre parallel file system provides high speed IO striping over the available storage, providing significantly higher IO performance than any local storage system. The lustre file system is also scalable in that multiple clients can simultaneously access the storage and in fact multiple clients are required to saturate the hard disk bandwidth of the SANs.

6. ACKNOWLEDGMENTS

We would like to thank the TEC IDF funded BeSTGRID project for providing the manpower and infrastructure to carry out the research reported in this paper.

7. REFERENCES

[1] L. Grosz, and A. Barczak, “Building an Inexpensive Parallel Computer”, *Lecture Notes in Computer Science*, vol 2660, (2003), pp 1050-1059, Springer-Verlag, ISSN 0302-9743.

[2] A.L.C. Barczak, C.H. Messom, M.J. Johnson, “Performance Characteristics of a Cost-Effective Medium-Sized Beowulf Cluster Supercomputer”, *Res. Lett. Inf. Math. Sci*, (5), pp 1-10, ISSN 1175-2777, 2003.

[3] Wilkinson, B and Allen, M. "Parallel Programming", Prentice Hall, New Jersey, 1999.

[4] Ridge, D., Becker, D., Merkey, P. and Sterling, T."Beowulf: Harnessing the Power of Parallelism in a Pile-of-PC", Proceedings of the IEEE Aerospace Conference, 1997.

[5] W.K. Yu, J. Vetter, R.S. Shane, S. Jiang, “Exploiting Lustre File Joining for Effective Collective IO”, *Proceedings of Seventh IEEE International Symposium on Cluster Computing and the Grid*, pp. 267-274, ISBN: 0-7695-2833-3, 2007.

[6] S.C. Simms, P.G. Pike, D. Balog, “Wide Area Filesystem Performance using Lustre on the TeraGrid”, *TERAGRID Conference*, 2007.

[7] S. Ihara, “Tokyo Tech Tsubame Grid Storage Implementation”, *Sun BluePrints™ On-Line*, Sun Microsystems, 2007

[8] Gropp, L. and Lusk, E. "Reproducible measurements of MPI performance characteristics", LNCS 1697, pp 11-18, Springer Verlag, 1999.

[9] Zaki, O., Lusk, E., Gropp, L. and Swider, D., "Toward Scalable Performance Visualisation with Jumpshot", High Performance Computing Applications, vol. 13, n. 2, pp 277-288, 1999.