the
BioTeam

White Paper

Bioinformatics Benchmarks on the SGI Altix XE Cluster

Prepared by:
Christopher Dwan
Principal Investigator

Primary Contact:
Stan Gloss
Managing Director

The BioTeam, Inc.
Cambridge, MA
http://www.bioteam.net

This report was prepared under contract to SGI Inc.

Table of Contents

1.0 Executive Summary 1
2.0 Test Methodology 1
  2.1 Evaluation Hardware 1
    2.1.1 Portal 1
    2.1.2 Compute Nodes 1
    2.1.3 Network 1
    2.1.4 Cluster Configuration 2
    2.1.5 Software Versions 2
3.0 Test Scripts 2
  3.1 Measurement Error 2
  3.2 Test suites available 2
4.0 Test 2
  4.1 BLAST 2
    4.1.1 BLAST Test: 1,000 proteins, BLASTP vs. NR 2
  4.2 CLUSTALW MPI 3
    4.2.1 CLUSTALW MPI Test 1: Nucleic Sequence Alignment 4
    4.2.2 CLUSTALW MPI Test 2: Protein Sequence Alignment 4
5.0 MRBAYES 5
  5.1 MRBAYES Test: 5
6.0 Summary and Conclusion 6
7.0 References and Further Reading: 6
  7.1 NCBI and BLAST 6
  7.2 CLUSTALW 7
  7.3 MRBAYES: 7
  7.4 Other Software 7

1.0 Executive Summary

This report presents a set of bioinformatics benchmark results on an SGI Altix XE cluster. Three common algorithms were used to exercise the systems: BLAST, ClustalW-MPI, and MrBayes. The performance of each tool scaled smoothly across the system, all of which showed excellent scaling performance.

Benchmarks in bioinformatics are complex, primarily because no two labs are doing precisely the same task at any given time. Traditional metrics of system performance, including clock speed, memory and IO bandwidth, are useful as broad quantifiers. Performance as perceived by any particular end-user is intimately tied to the architecture and design of the system as a whole, as well as to the specific use case. Computing systems for use in genomic biology must be usable across a broad range of skill sets. They must support users with a wide variety of skill sets. Perhaps most important, they must support both relatively static "production" workloads, as well as smaller research and development tests.

Any cluster is composed of a set of individual systems. The performance of the system as a whole will be effected by two primary factors: The performance of the algorithms in question on a single one of those systems, and the parallelization used to spread those algorithms across the cluster. In this set of benchmarks, we focused primarily on the scaling characteristics of the three algorithms.

2.0 Test Methodology

2.1 Evaluation Hardware

The evaluation system was an SGI "Altix XE Cluster." It consists of eight independent Linux systems: 1 portal (XE 240) and 7 compute nodes (XE 210). The system arrived pre-wired in a half height rolling rack, and we transferred the servers and network switches to the installed racks in our collocation facility for the duration of the tests, re-connecting all wiring to match the original configuration.

2.1.1 Portal

Model:	SGI Altix XE 240
Form Factor:	2U rack-mount server
CPU:	2x Intel dual core Xeon 5160 - 3.0GHz
RAM:	16GB (expandable to 32GB)
Network:	– Dual Gigabit Ethernet – Infiniband
Disk:	5x disk drives configured as a RAID 5 with nearly 0.5TB of usable space.

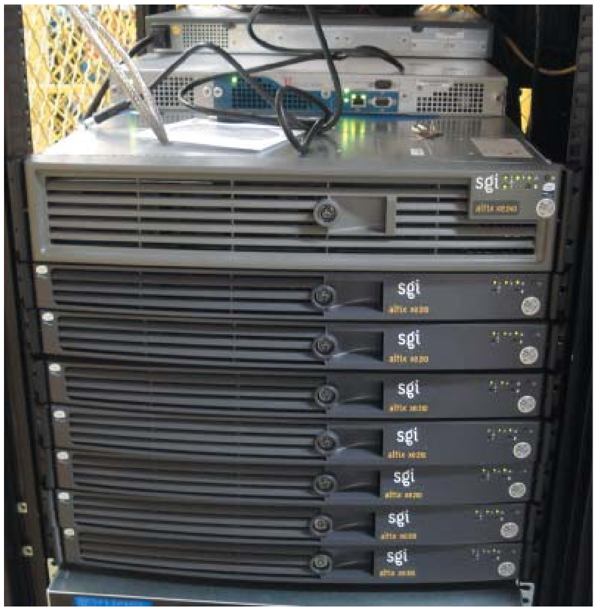


Figure 1: Altix XE Cluster

2.1.2 Compute Nodes

Model:	SGI Altix XE 210
Form Factor:	1U high density rack-mount server
CPU:	2x Intel Xeon 5160 - 3.0GHz
RAM:	8GB (expandable to 32GB)
Network:	– Dual gigabit Ethernet – Infiniband
Disk:	1 disk drive with 200GB of usable space.

2.1.3 Network

The systems were configured with two private networks, both of which are entirely dedicated to internal cluster traffic. The Gigabit Ethernet network used an SMC "TigerConnect" switch, while the high speed, low latency Infiniband network used a Voltaire 9024 switch.

The gigabit ports on the Altix servers implement a network management protocol named "IPMI," which stands for the "Intelligent Platform Management Interface." IPMI provides utilities for monitoring and simple manipulation of the system independent of the operating system. Specifically, it is possible to power the chassis on and off without any additional serial or management connections. This proved remarkably useful for remotely powering the compute nodes down when they were not in use.

2.1.4 Cluster Configuration

Operating System: The sytems were pre-configured with SUSE Linux version 9. In addition, SGI provided the SCALI cluster management software for remote management of the systems. In performing these benchmarks we used the pre-installed software to the extent possible.

Distributed Resource Management: Sun Grid Engine version 6.0u9. Because these benchmarks were run by hand, by a single user, with the cluster in an un-loaded condition, there would be no appreciable advantage to be found in one scheduler or queuing system over another. It was simply a way to run the jobs on the machines.

Shared Filesystems: The primary RAID partition on the portal (/data1) was exported via NFS over the ethernet network.

The cluster was pre-configured using the cluster configuration tools from Scali. These proved quite useful, since pre-built versions of the MPI and Infiniband tools were available simply by changing environment variables. In addition, succinct hardcopy system documentation including critical passwords and network addresses was provided, which made it simple to get the cluster up and running. The system was operational literally minutes after we completed re-wiring it. This will provide a great deal of value in environments where system administration expertise is in short supply.

2.1.5 Software Versions

All software was built from source, using the default configuration parameters.

- BLAST, version 2.2.15 obtained from the NCBI toolkit
- CLUSTALW_MPI: version 0.13, obtained from the author
- MrBayes: version 3.1.1, obtained from the project website
- MPICH 2; version 1.0.3, pre-installed on the system.

3.0 Test Scripts

System tests were run via a PERL script that uses a system fork to start and monitor process execution. Wall clock time for completion is the primary metric used in this report.

All tests were run with the systems in a near idle, multi-user configuration. While tests were in progress, no other processing was performed except for that required for monitoring progress through a terminal session.

3.1 Measurement Error

Each experimental test was executed at least twice, and some individual tests were executed many. In every case, there was a high level of agreement (within 0.1%) between the execution runtimes of equivalent tests. The benchmarks reported in this document are averages of the observed runtimes.

3.2 Test Suites Available

The code and data used to execute these tests is available for download from the BioTeam web server: http://bioteam.net/sgi_benchmarks.

4.0 Tests

4.1 BLAST

BLAST (Basic Local Alignment Search Tool), from the National Center for Biological Information (NCBI) is the most popular piece of software ever developed for genetic sequence analysis. Biologists use BLAST to analyze DNA or protein sequence data. BLAST performs a similarity search in which a dataset of genome or protein sequence (the “target” set) is scanned for sequences similar to some other set of sequences provided by the user (the “query” set). BLAST searches are screening tests for sequence similarity. These similarities are a starting point for understanding potential biological significance.

BLAST is trivially parallelizable, assuming that there are many searches to run. One simply runs each search independent of the others, on a totally separate machine, if possible. Bioteam uses a script named “btbatchblast” to accomplish this parallelization. There is no optimization or parallel communication in the script. It counts the number of query sequences, divides by the desired number of chunks, and submits the appropriate number of subordinate tasks. The scheduler decided which system should run any particular unit of work, until all the cores were occupied. On this particular system, when less than 28 tasks available, some CPU cores sat idle. When more than 28 tasks were in the system, the extras waited in the queue until earlier ones completed, and were run in turn. There was no oversubscription of CPUs in this test. Instead, a queuing system was used to manage up to 28 concurrent tasks on the CPUs.

4.1.1 BLAST Test: 1,000 Proteins, BLASTP vs. NR

For the BLAST test, we selected the first 1,000 protein sequences from the E. Coli dataset from NCBI as the query, and the “NR” dataset from NCBI as the target. The NR dataset consists of approximately 2GB of amino acid sequences, and should fit easily into the RAM of the compute nodes. Since the search is protein vs. protein, we used the “BLASTP” algorithm. BLASTP provides the most interesting balance of IO and CPU load. The BLAST variants which compare in DNA “space” tend to be more limited by RAM and data IO than anything else. To prevent the common bottleneck at the NFS shared filesystem, the BLAST target files were copied to the local disks directly attached to the compute nodes.

Figure 1 shows both the time and speedup from running this test, varying the number of "task" subdivisions to be used. It shows clearly that both the number of available CPUs and the configuration of the test itself are important to the performance of a computing system on a particular job. Performance increases smoothly until the number of tasks equals the number of CPU cores available to do the work. After that point, we see a discontinuity. This is not due to oversubscription or contention for resources, since the scheduler prevented more than four tasks starting on any particular chassis at any time. The simplest explanation for the discontinuity is that, when the number of tasks equals the number of CPUs, each CPU is allotted one task. Adding a small number of tasks reduces the size of each task, but means that some small number of tasks will remain to be done, after the first 28 complete. This means that we incur a cost of the time required to run a whole extra task. The fact that the CPUs that finish first had the shortest original piece of work mitigates this somewhat. In Figure 1, we see that the two best choices for number of tasks are 28 and 56: One and two times the number of CPUs, respectively. Continuing to increase the number of tasks past 56 again incurs a penalty, but a smaller one. This is because the individual units of work are smaller, with more subdivisions. This speaks well to the ability of the system to be further tuned for high throughput bioinformatics environments.

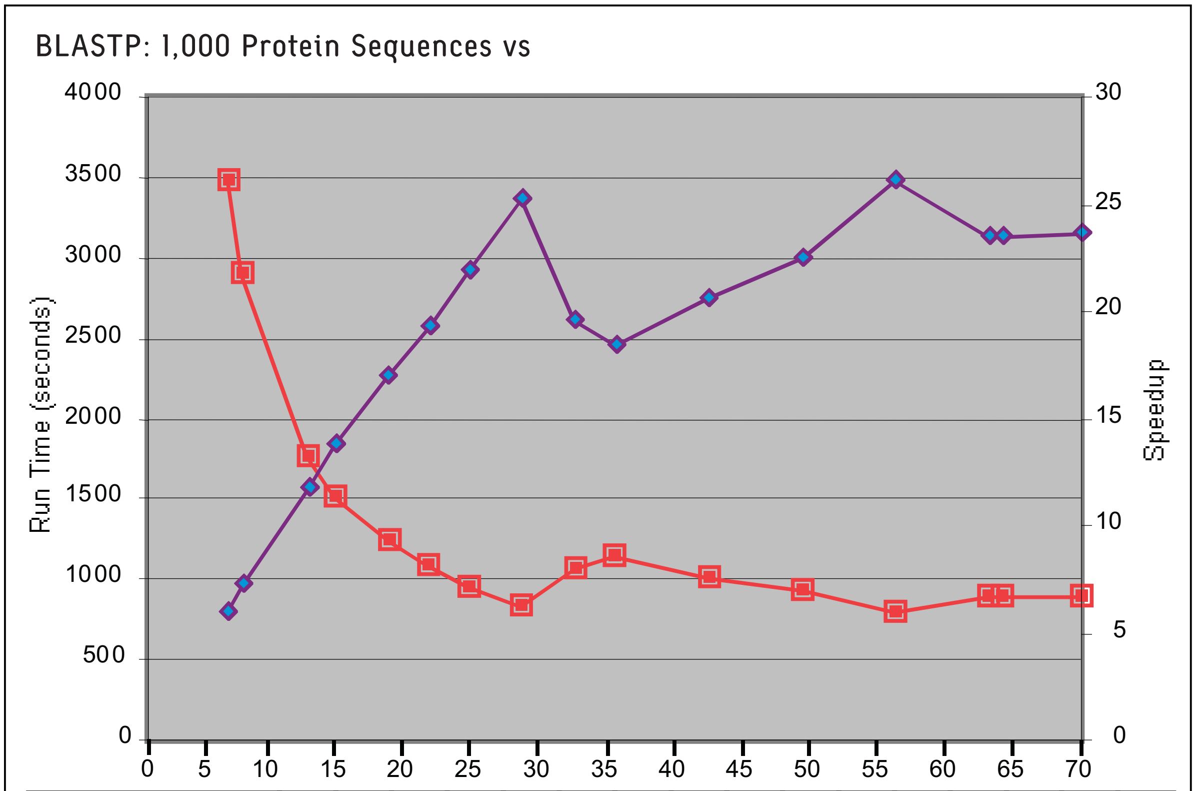
The following definitions apply in all of the tables and plots.

Runtime:	The runtime in seconds
Speedup:	Serial runtime divided by parallel runtime

4.2 CLUSTALW MPI

CLUSTALW is a program that performs alignments of multiple sequence alignments. Biologists use CLUSTALW to find the portions of a set of related sequences that have been conserved during evolution, and also to determine subtle patterns of similarity among a set of sequences. These patterns might not be obvious from the pairwise similarities returned by BLAST.

Unlike BLAST, CLUSTALW does not search a large set of data. Instead, CLUSTALW evaluates a huge space of possible gapped alignments based on a relatively small set of input data. It finds the best answer out of a large set of possible answers. Computational jobs requiring hours, days, or weeks of CPU time can be defined fairly easily based on small groups of sequences.



# of Subtasks	6	7	12	14	16	21	24	28	32	35	42	49	56	63	64	70
▣ Run Time	3470	2901	1766	1515	1228	1077	950	826	1067	1135	1010	928	797	888	888	881
◆ Speedup	6	7.2	11.8	13.7	17	19.3	21.9	25.2	19.5	18.3	20.6	22.4	26.1	23.4	23.4	23.6

Figure 1

This means that the performance characteristics should be determined more by CPU characteristics than by memory or I/O.

The parallel / MPI implementation of CLUSTALW was developed at the Bioinformatics Institute in Singapore (http://www.bii.a-star.edu.sg/). It is a re-write of the original code to parallelize and accelerate processing.

The CLUSTALW source code includes performance tests to verify that the code is functional, and to provide performance benchmarks. We used two of these for our tests, a moderate sized alignment in both DNA and protein.

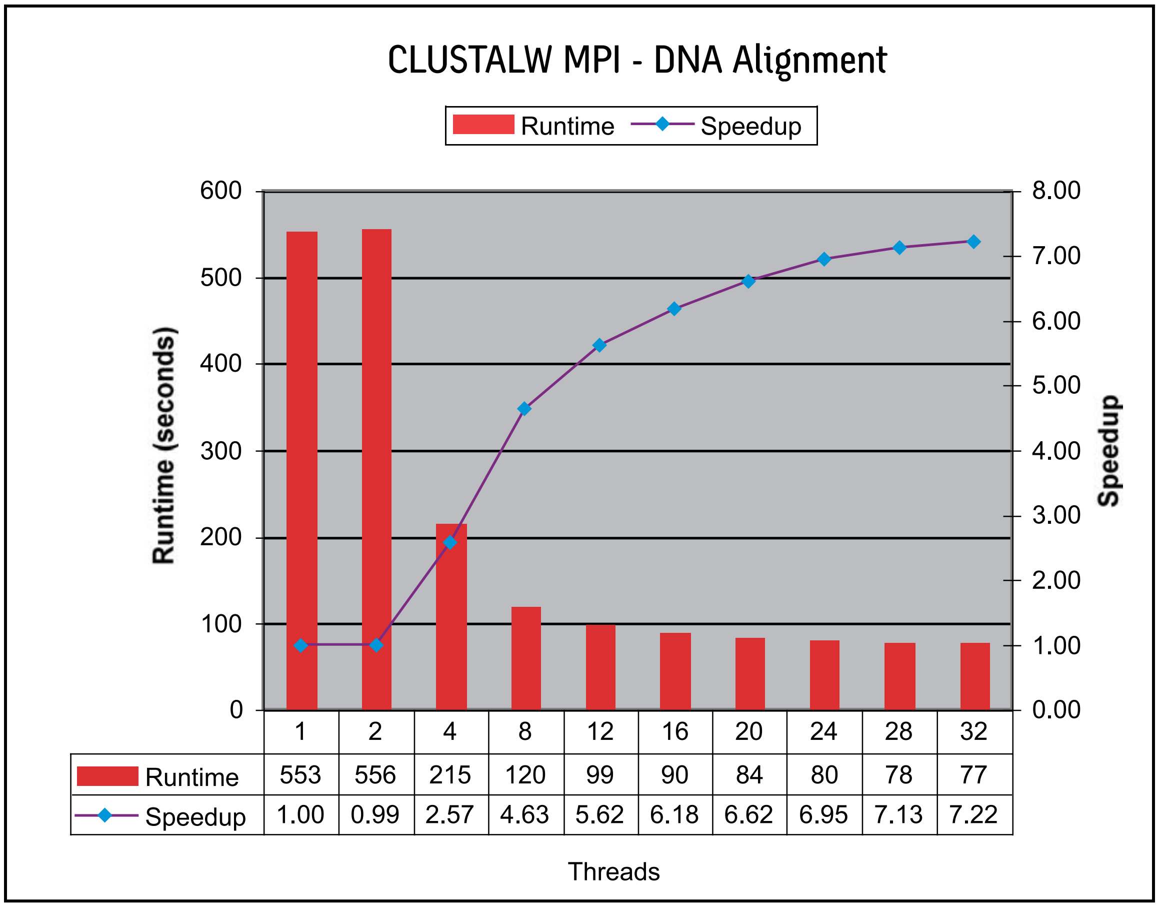
4.2.1 CLUSTALW MPI Test 1: Nucleic Sequence Alignment

The first performance benchmark provided with the CLUSTALW software distribution is a DNA alignment of 17 sequences of different variants of HIV. This is the sort of analysis that might be performed by a researcher trying to determine if a set of infections has a common source, or to track the difference between lethal and non-lethal variants of a disease.

Figure 2 shows the results of 10 runs of CLUSTALW, using up to 32 processes spread across the machines in the cluster. The diminishing returns at the higher numbers of CPUs are to be expected, since with only 17 input sequences there is little that increased parallelism can accomplish. The task requires the same amount of time to complete with two threads as with one. This is also true in the other CLUSTALW test, shown in figure 3 and has been observed with MPI-CLUSTALW running on other systems. It is not clear why this is the case, but it appears to be application dependent.

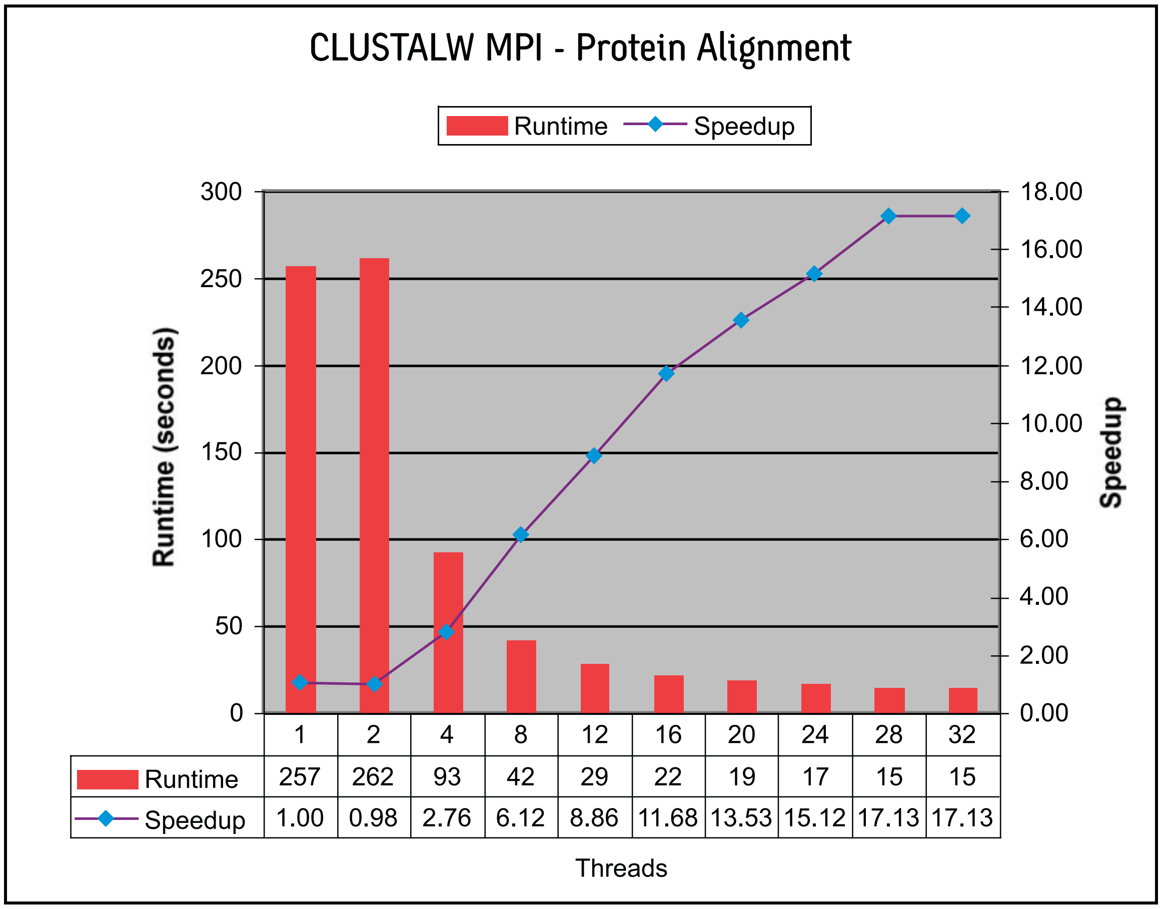
4.2.2 CLUSTALW MPI Test 2: Protein Sequence Alignment

The second test provided with the CLUSTALW software is a protein alignment of 469 protein sequences from the PFam database. These are the variants of a protein called Peptidase. The number of sequences is much higher, but each individual sequence is much shorter: The sequences average 280 letters per sequence compared with 10,000 for the entire virus in the previous test. This is the sort of analysis that a biologist might perform in order to better understand the functional



	1	2	4	8	12	16	20	24	28	32
🟥 Runtime	553	556	215	120	99	90	84	80	78	77
◆ Speedup	1.00	0.99	2.57	4.63	5.62	6.18	6.62	6.95	7.13	7.22

Figure 2



Threads	1	2	4	8	12	16	20	24	28	32
Runtime	257	262	93	42	29	22	19	17	15	15
Speedup	1.00	0.98	2.76	6.12	8.86	11.68	13.53	15.12	17.13	17.13

Figure 3

relationships within a family of similar molecules. Any features which are preserved across more than 400 proteins are most likely of interest.

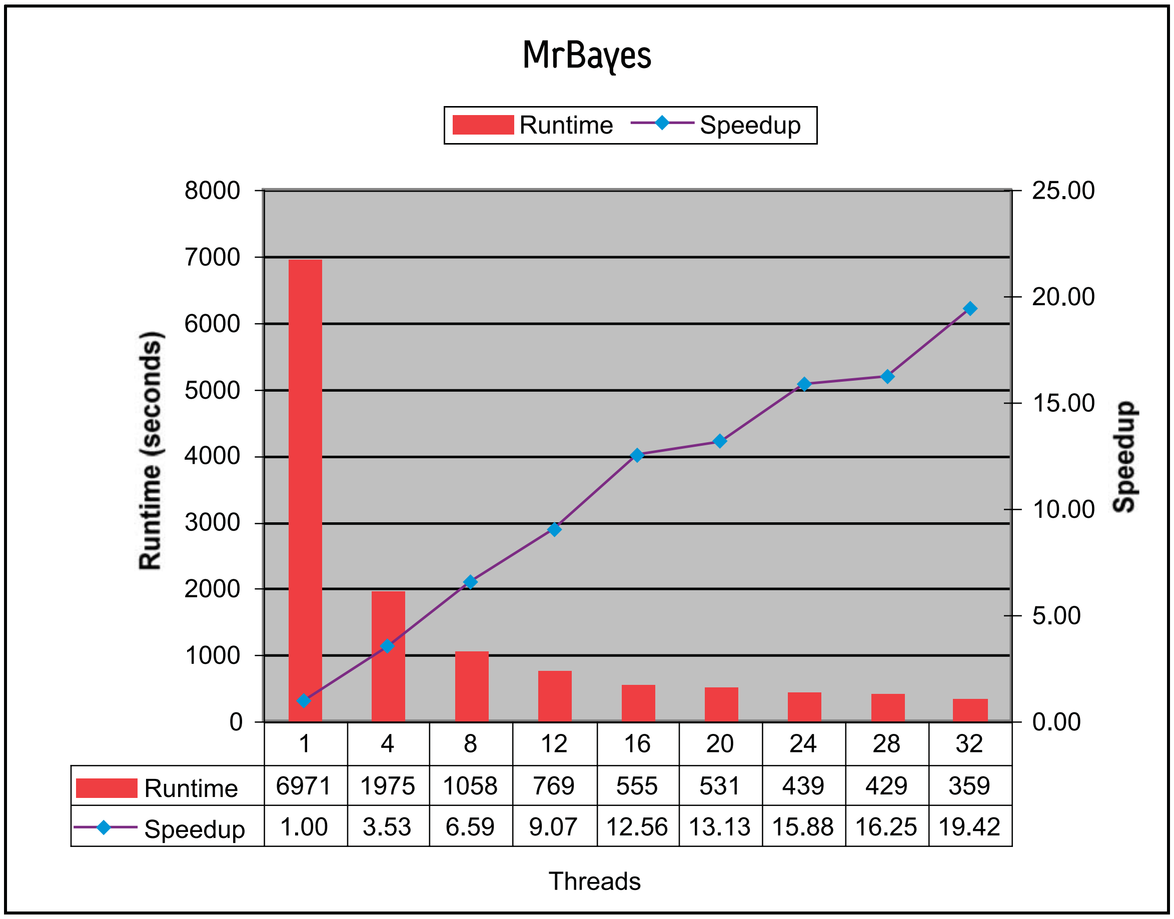
Figure 3 shows the timing results when this CLUSTALW job was run with up to 32 parallel processes on the cluster. We see steady performance gains until the number of processes equals the number of CPU cores in the cluster, at which point no further improvement is to be had. While the wall clock difference between the performance with 24 and 28 threads is not great, there is clearly still room within this data set for further parallelization.

5.0 MRBAYES

MrBayes performs Bayesian inference of phylogeny. This means that it computes the most probable evolutionary history that could have led to a set of observed sequences. It uses a technique called “Markov Chain Monte Carlo.” MrBayes was explicitly designed for use in parallel environments, using the MPI standard for inter-process communication. Understanding the most likely relationships between a set of samples is incredibly valuable in understanding both the differences and the similarities in that data.

5.1 MRBAYES Test:

The MrBayes test is a phylogenetic analysis of 243 different DNA sequeneces., each approximately 800 base pairs in length. The sequences are derived from African vegetables, including sweet potato, cassava, bean, and others. The analysis performed is the “MCMC” algorithm, running for 5,000 generations, and using 32 “chains” to ensure that the number of threads of work to be done equals or exceeds the number of CPUs doing the work. In practice, scientists run this sort of algorithm until it “converges” on a solution, or until they run out of time on the computing resource. The generation and chain counts were selected to show a spread of runtimes on the available machines.



Threads	1	4	8	12	16	20	24	28	32
🟥 Runtime	6971	1975	1058	769	555	531	439	429	359
◆ Speedup	1.00	3.53	6.59	9.07	12.56	13.13	15.88	16.25	19.42

Figure 4

Figure 4 shows that performance continued to improve with each added set of CPUs, all the way to utilization of the entire system. The performance difference between 28 and 32 threads is small in terms of raw seconds, but we clearly had not reached the point of diminishing returns in terms of the scaling of the problem.

6.0 Summary and Conclusion

This paper explores the scaling performance of three popular algorithms in bioinformatics on a cluster of SGE Altix XE systems. The cluster was extremely easy to set up and get running in our lab, and showed good performance across the breadth of sequence based bioinformatics tasks.

7.0 References and Further Reading:

7.1 NCBI and BLAST

• NCBI:	http://www.ncbi.nih.gov
• BLAST Tutorial:	http://www.ncbi.nlm.nih.gov/blast/producttable.shtml
• NCBI FTP Site:	ftp://ftp.ncbi.nih.gov/pub/blast
• NCBI Toolkit Archive:	ftp://ftp.ncbi.nih.gov/toolbox/old/20050828

- Altschul, S.F., Gish, W., Miller, W., Myers, E.W. & Lipman, D.J. (1990) “Basic local alignment search tool.” J. Mol. Biol. 215:403-410.
- Altschul, S.F., Madden, T.L., Schäffer, A.A., Zhang, J., Zhang, Z., Miller, W. & Lipman, D.J. (1997) “Gapped BLAST and PSI-BLAST: a new generation of protein database search programs.” Nucleic Acids Res. 25:3389-3402.

7.2 CLUSTALW

• CLUSTALW at the EBI:	http://www.ebi.ac.uk/clustalw/
• CLUSTALW MPI:	http://web.bii.a-star.edu.sg/~kuobin/clustalg/

- Thompson JD, Higgins DG, Gibson TJ: "CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice". Nucleic Acids Res 1994, 22:4673-4680.
- Kuo-Bin Li, "ClustalW-MPI: ClustalW Analysis Using Distributed and Parallel Computing", Bioinformatics, 2003, 19(12), 1585--1586.

7.3 MRBAYES:

• MrBayes Project:	http://mrbayes.csit.fsu.edu/index.php
• Sourceforge Page:	http://sourceforge.net/projects/mrbayes

- Huelsenbeck, J. P., F. Ronquist, R. Nielsen and J. P. Bollback. 2001. "Bayesian inference of phylogeny and its impact on evolutionary biology". Science 294: 2310-2314.
- Ronquist, F. and J. P. Huelsenbeck. 2003. "MRBAYES 3: Bayesian phylogenetic inference under mixed models". Bioinformatics 19:1572-1574.

7.4 Other Software

• GCC:	http://gcc.gnu.org
• Intel Compilers:	http://www.intel.com/cd/software/products/asmo-na/eng/compilers/index.htm
• IPMI:	http://en.wikipedia.org/wiki/Intelligent_Platform_Management_Interface
• Fedora Linux:	http://fedora.redhat.com
• MPI:	http://www.mpi-forum.org
• MPICH:	http://www-unix.mcs.anl.gov/mpi/mpich/
• SGE:	http://gridengine.sunsource.net/
• SUSE Linux:	http://www.novell.com/linux/suse/

the BioTeam

The BioTeam, Inc.
7 Derosier Drive,
Middleton,
MA 01949
(978) 304-1222