Tunghai Science Vol. : 125−141
July, 2003

Apply Parallel Bioinformatics Applications on Linux PC Clusters

Yu-Lun Kuo and Chao-Tung Yang*

Abstract

In addition to the traditional massively parallel computers, distributed workstation clusters now play an important role in scientific computing perhaps due to the advent of commodity high performance processors, low-latency/high-band width networks and powerful development tools. As we know, bioinformatics tools can speed up the analysis of large-scale sequence data, especially about sequence alignment. To fully utilize the relatively inexpensive CPU cycles available to today's scientists, a PC cluster consists of one master node and seven slave nodes (16 processors totally), is proposed and built for bioinformatics applications. We use the mpiBLAST and HMMer on parallel computer to speed up the process for sequence alignment. The mpiBLAST software uses a message-passing library called MPI (Message Passing Interface) and the HMMer software uses a software package called PVM (Parallel Virtual Machine), respectively. The system architecture and performances of the cluster are also presented in this paper.

Keywords: Parallel computing, Bioinformatics, BLAST, HMMer, PC Clusters, Speedup.

1. Introduction

Extraordinary technological improvements over the past few years in areas such as microprocessors, memory, buses, networks, and software have made it possible to assemble groups of inexpensive personal computers and/or workstations into a cost effective system that functions in concert and posses tremendous processing power. Cluster computing is not new, but in company with other technical capabilities, particularly in the area of networking, this class of machines is becoming a high-performance platform for parallel and distributed applications [1, 2, 11, 12, 13, 14, 15, 16, 17].

Scalable computing clusters, ranging from a cluster of (homogeneous or heterogeneous) PCs or workstations to SMP (Symmetric MultiProcessors), are rapidly becoming the standard platforms for high-performance and large-scale computing. A cluster is a group of independent computer systems and thus forms a loosely coupled multiprocessor system as show in Figure 1.

A network is used to provide inter-processor communications. Applications that are distributed across the processors of the cluster use either message passing or network shared memory for communication. A cluster computing system is a compromise between a massively parallel processing system and a distributed system. An MPP (Massively Parallel Processors) system node typically cannot serve as a standalone computer; a cluster node usually contains its own disk and equipped with a complete operating systems, and therefore, it also can handle interactive jobs. In a distributed system, each node can function only as an individual resource while a cluster system presents itself as a single system to the user.

Since a Beowulf cluster is a parallel computer system, it suits applications that can be partitioned into tasks, which can then be executed concurrently by a number of processors. These applications range from high-end, floating-point intensive scientific and engineering problems to commercial data-intensive tasks. Uses of these applications include ocean and climate modeling for prediction of temperature and precipitation, seismic analysis for oil exploration, aerodynamic simulation for motor and aircraft design, and molecular modeling for biomedical research.

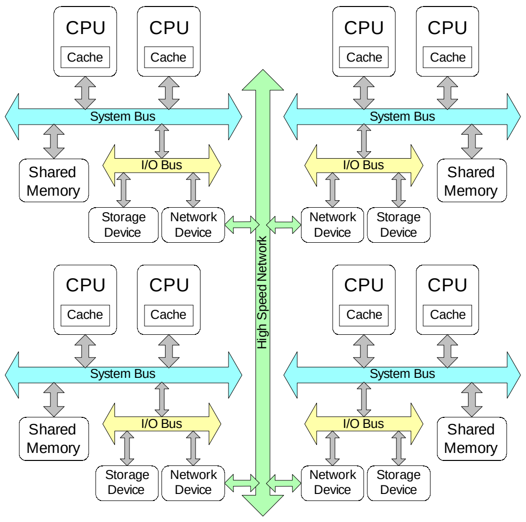


Figure 1: A cluster system by connecting four SMPs.

Inexpensive systems such as Beowulf clusters have become increasingly popular in both the commercial and academic sectors of the bioinformatics community. Clusters typically consist of a master node that distributes the bioinformatics application amongst the other nodes (slave nodes). There exist some parallel version bioinformatics application [22] which can be installed

[*] Department of Computer Science and Information Engineering, Tunghai University, Taichung 407, TAIWAN

and conducted on a PC cluster [5, 6, 7], for example, HMMer, FASTA, mpiBLAST, PARACEL BLAST [27], ClustalW-MPI [35], Wrapping up BLAST [34], and TREE-PUZZLE [25] et al. Using these parallel programs on sequence alignment can always save much time and cost. Especially for some companies, the PC clusters can be used to replace mainframe systems or supercomputers and save much hardware cost. According to efficiency and cost, the use of parallel version software and cluster system is a good way and it will become more and more popular [11] in the near feature. As we know, bioinformatics tools can speed up the analysis of large-scale sequence data, especially for sequence alignment. To fully utilize the relatively inexpensive CPU cycles available to today's scientists, a PC cluster consists of one master node and seven slave nodes (16 processors totally), is proposed and built for bioinformatics applications. We use the mpiBLAST and HMMER on parallel computer to speed up the process of sequence alignment [23, 24, 28]. The system architecture and performance of the cluster are also presented in this paper.

2. Parallel Bioinformatics Applications

2.1 BLAST and mpiBLAST

2.1.1 BLAST

The most popular tool for searching sequence databases is a program called BLAST (Basic Local Alignment Search Tool) [29]. BLAST compares two sequences by trying to align them, and is also used to search sequences in a database. The algorithm starts by looking for exact matches, and then expands the aligned regions by allowing for mismatches. It performs pairwise comparisons of sequences, seeking regions of local similarity, rather than optimal global alignments between whole sequences. Actually the NCBI-BLAST implementation is the de facto standard for biological sequence comparison and search in computational biology.

Here are the four main executable programs in the BLAST distribution [30, 31]:

- [*blastall*]

  Performs BLAST searches using one of five BLAST programs: *blastn*, *blastp*, *blastx*, *tblastn*, or *tblastx*

  The following table summarizes the query, database sequence, and alignment types for the various BLAST commands.

Program	Query sequence type	Database sequence type	Alignment sequence type
blastn	nucleotide	Nucleotide	nucleotide
blastp	protein	Protein	protein
blastx	nucleotide	Protein	protein
tblastn	protein	nucleotide	protein
tblastx	nucleotide	nucleotide	protein

- [*blastpgp*]

  Performs searches in PSI-BLAST or PHI-BLAST mode. *blastpgp* performs gapped *blastp* searches and can be used to perform iterative searches in psi-blast and phi-blast mode.
- [*bl2seq*]

  Performs a local alignment of two sequences. *bl2seq* allows the comparison of two known sequences using *blastp* or *blastn* programs. Most of the command-line options for *bl2seq* are similar to those for *blastall*.
- [*formatdb*]

  formatdb is used to format protein or nucleotide source database. It converts a FASTA-format flat file sequence database into a BLAST database

2.1.2 mpiBLAST

BLAST is a set of programs to find similarity between a query protein or DNA sequence and a sequence database. A scheme for efficiently running a large number of sequence files against a variety of BLAST databases has been implemented on pc clusters. The BLAST search algorithms consume the bulk of compute cycles for most bioinformatics researchers. The algorithms search for similarities between a short query sequence and a large, unchanging database of DNA or amino acid sequence. As BLAST is both computationally intensive and embarrassingly parallel, several paradigms exist for parallelizing BLAST including multithreading, query segmentation, and database segmentation. Virtually all parallel implementations use multithreading and query segmentation. Using the ubiquitous parallel-programming library called MPI: Message Passing Interface, mpiBLAST [32] segments a database into several fragments such that each node in a computational cluster searches a unique portion of the database. Database segmentation offers two primary advantages over existing parallel BLAST algorithms. First, the current size of sequence databases is larger than core memory on most computers, forcing BLAST searches to use disk I/O. Segmenting the database permits each node to search a smaller portion of the database, eliminating disk I/O and vastly improving BLAST performance. Second, because database segmentation does not create heavy interprocessor communication demands, it allows BLAST users to take advantage of power-efficient, space-efficient, low-cost clusters. The mpiBLAST is a freely available open source parallelization of NCBI BLAST. The mpiBLAST segments the BLAST database and distributes it across cluster nodes, permitting BLAST queries to be processed on many nodes simultaneously. mpiBLAST is based on MPI. The current release runs under Linux and Windows, and will probably work on other varieties of UNIX as well.

2.2 HMMer

Profile hidden Markov models (profile HMMs) [20, 21] can be used to do sensitive database searching using statistical descriptions of a sequence family's consensus. HMMer uses profile HMMs for several types of homology searches. HMMer is a software package which is an implementation of profile hidden Markov model (HMM) methods for sensitive database searches using multiple sequence alignment as queries. About HMMer’s sequence file format, it attempts to read most common biological sequence file formats. The programs automatically detect what format the file is in and whether the sequences are DNA, RNA, or protein.. Unaligned sequence files may be in FASTA, SWISS-PROT, EMBL, GenBank, PIR, or GCG format. And aligned sequence files (multiple sequence alignments) may be in CLUSTALW, SELEX, or GCG MSF format. HMMer’s main functionality is located in the hmmbuild program and the hmmcalibrate program. The first program creates profile HMMs from sequence alignment, and the second program calibrates search statistics for the HMMs. The HMMs software packages also contain many useful programs such like hmmpfam, hmmindex, hmmsearch. List some HMMer tools here:

- [hmmpfam] Searches a profile HMM database with a query sequence, trying to annotate an unknown sequence.
- [hmmindex] Create a binary SSI index for HMM database.
- [hmmsearch] Searches a sequence database with a profile HMM, looking for more instances of a pattern in a sequence database.
- [hmmalign] Align multiple sequences to a profile HMM.
- [hmmbuild] Builds a profile HMM from a multiple sequence alignment.
- [hmmcalibrate] Reads an HMM and calibrates its search statistics.
- [hmmconvert] Converts an HMM into other profile formats.

- [hmmemit] Generates sequences probabilistically based on a profile HMM. It can also generate a consensus sequence.

  [hmmfetch] Retrieves a profile HMM from HMM database.

2.2.1 Hmm Database-Pfam Database

Pfam [19] is a database of alignments of protein domain families and a database of profile Hidden Markov Models. Pfam includes two sub-dataset: Pfam-A and Pfam-B. Pfam-A contains over 2700 gapped profiles, and most of them cover whole protein domains; Pfam-B entries are generated automatically by applying a clustering method to the sequences left over from the creation of Pfam-A. Pfam-A entries being with a "seed alignment", it is a biologically meaningful multiple sequence alignment and sometimes may involve some manual editing. From each seed alignment, a profile hidden Markov model is constructed and used to search more protein sequences available. The process can be iterated to make the family domain more complete. The latest Pfam version is 10.0 (July 2003), which contains alignments and models for 6190 protein families, based on the SWISS-PROT 41.10 and SP-TrEMBL 23.15 protein sequence databases.

3 System Environment and Applications Installation

3.1 Our System Environment

We used dual-processor motherboards to reduce the number of boxes to eight, thus minimizing the space needed for storage (and the footprint of the cluster). This structure impacts performance because two processors share the memory bus (which causes bus contention but reduces the hardware cost) since only one case, motherboard, hard drive, etc., are needed for two processors. We ruled out the option of rack-mounting the nodes, essentially to reduce cost, but chose to use standard mid-tower cases on shelves. This approach is sometimes given the name LOBOS ("lots of boxes on shelves").

Our SMP cluster is a low cost Beowulf-type class supercomputer that utilizes multi-computer architecture for parallel computations. It consists of eight PC-based symmetric multiprocessors (SMP) connected by one 24-port 100Mbps Ethernet switches with Fast Ethernet interface. There are one server node and seven computing nodes. The server node has two AMD ATHLON MP 2000+ processors and 1GBytes of shared local memory. Each AMD ATHLON processor has 128K on-chip instruction and data caches (L1 cache), a 256K on-chip four-way second-level cache with full speed of CPU. Each computing node has dual AMD ATHLON MP 1800+ with 512MB shared-memory.

The Redhat 8.0 Linux distribution has been installed on each node. The idea of the Linux cluster is to maximize the performance-to-cost ratio of computing by using low-cost commodity components and free-source Linux and GNU software to assemble a parallel and distributed computing system. Software support includes the standard Linux/GNU environment, including compilers, debuggers, editors, and standard numerical libraries. Coordination and communication among the processing nodes is a key requirement of parallel-processing clusters. In order to accommodate this coordination, developers have created software to carry out the coordination and hardware to send and receive the coordinating messages. Messaging architectures such as MPI or Message Passing Interface, and PVM or Parallel Virtual Machine, allow the programmer to ensure that control and data messages take place as needed during operation.

MPI is a message-passing library standard that was published in May 1994. The “standard” of MPI is based on the consensus of the participants in the MPI Forum, organized by over 40 organizations. Participants included vendors, researchers, academics, software library developers and users. MPI offers portability, standardization, performance, and functionality. The advantage for the user is that MPI is standardized on many levels. For example, since the syntax is standardized, you can rely on your MPI code to execute under any MPI implementation running on your architecture. Since the functional behavior of MPI calls is also standardized, your MPI calls should behave the same regardless of the implementation. This guarantees the portability of your parallel programs. Performance, however, may vary between different implementations. MPI includes point-to-point message passing and collective (global) operations. These are all scoped to a user-specified group of processes. MPI provides a substantial set of libraries for the writing, debugging, and performance testing of distributed programs. Our system currently uses LAM/MPI, a portable implementation of the MPI standard developed cooperatively by Notre Dame University. LAM (Local Area Multicomputer) is an MPI programming environment and development system and includes a visualization tool that allows a user to examine the state of the machine allocated to their job as well as provides a means of studying message flows between nodes.

PVM, or Parallel Virtual Machine, started out as a project at the Oak Ridge National Laboratory and was developed further at the University of Tennessee. PVM is a complete distributed computing system, allowing programs to span several machines across a network. PVM utilizes a Message Passing model that allows developers to distribute programs across a variety of machine architectures and across several data formats. PVM essentially collects the network's workstations into a single virtual machine. PVM allows a network of heterogeneous computers to be used as a single computational resource called the parallel virtual machine. As we have seen, PVM is a very flexible parallel processing environment. It therefore supports almost all models of parallel programming, including the commonly used all-peers and master-slave paradigms. A typical PVM consists of a (possibly heterogeneous) mix of machines on the network, one being the "master" host and the rest being "worker" or "slave" hosts. These various hosts communicate by message passing. The PVM is started at the command line of the master which in turn can spawn workers to achieve the desired configuration of hosts for the PVM. This configuration can be established initially via a configuration file. Alternatively, the virtual machine can be configured from the PVM command line (master's console) or during run time from within the application program.

3.2 mpiBLAST Installation

mpiBLAST is a parallelization of NCBI BLAST. mpiBLAST is a pair of programs that replace *formatdb* and *blastall* with versions that execute BLAST jobs in parallel on a cluster of computers with MPI installed. There are two primary advantages to using mpiBLAST versus traditional BLAST. First, mpiBLAST splits the database across each node in the cluster. Because each node's segment of the database is smaller it can usually reside in the buffer-cache, yielding a significant speedup due to the elimination of disk I/O. Second, it allows BLAST users to take advantage of efficient, low-cost Beowulf clusters because interprocessor communication demands are low. The installation steps are descried as below [32].

1. mpiBLAST requires that an MPI [4] implementation is installed.
2. mpiBLAST also requires that the computers have some shared storage directory. This can be an NFS mount, samba share, AFS, or any other type of shared network file system. The location of the shared directory must be specified in the *mpiblast.conf* config file.
3. mpiBLAST requires the NCBI libraries and the formatdb executable. So to build mpiBLAST from source you will also need to compile the NCBI Toolbox [33].
   - Take the sources of NCBI toolkit and put the archive to the new empty directory. This directory be called toolkit.
   - cd toolkit
   - download ncbi.tar.Z and compress –d < ncbi.tar.Z | tar xvf
   - run the command: env CC = gcc ./ncbi/make/makedis.csh - |& tee out.makedis.txt
4. From the mpiblast directory:
   - > ./configure
   - > make

- And (optionally as root):
- > make install

5. Useful options for 'configure':
   - **--prefix=/path/to/install/directory** Specifies the location where mpiBLAST should be installed
   - **--with-ncbi=/path/to/ncbi/** Specifies the path to the NCBI Toolbox
   - **--enable-many-fragments** Causes mpiBLAST too look for 3 digit fragment identifiers instead of 2, permitting mpiBLAST to use up to 1000 database fragments. The NCBI Toolbox must be patched for this to work.
   - **--enable-mpe** Causes mpiBLAST to use MPE logging to measure the running time of its components
   - **--enable-debug** Causes mpiBLAST to be compiled with debug options
6. Creating the config file. This config called *mpiblast.conf*. And the config file contains three lines, and looks like this:

   ```
   /path/to/shared/storage
   /path/to/local/storage
   /path/to/NCBI/BLAST/binaries
   ```

   The first line designates the location of the shared storage that will be used for communication of database fragments, queries, and BLAST results. The second line designates the location of a directory on the node's local hard drive where that node's database fragments can be stored. The third line designates the path where the NCBI BLAST programs can be found.
7. Creating a file named "a.ncbirc". And the contents is such like below:

   ```
   [NCBI]
   Data=/path/toolkit/ncbi/data
   ```

   - This file is used to calculate the value by BLOSUM45 matrix.
8. Formatting the database. And we use the yeast.nt database. This yeast.nt database's size near about 12 MB.
   - ./mpiformatdb -f /path/to/mpiblast.conf -N 8 -i yeast.nt -o T -p F
   - Because our system includes 8 computing nodes. So we use mpiformatdb format yeast.nt into 8 fragments.
   - Option: -i means "Input file for formatting. And input file is yeast.nt database.
   - Option: -p means the type of file: (1) T = Protein (2) F = Nucleotide

9. Then the multiple alignment tool can use now. Type a command to execution mpiBLAST software such like below.
   - mpirun -np 8 mpiblast --config-file=/path/to/mpiblast.conf -p blastn -d yeast.nt -i blast_query.fas -o blast_results.txt
   - Option: -d means the database name
   - Option: -i means the query file
   - Option: -o means the BLAST report output file

3.3 HMMer Installation

The Washington University HMMer homepage [18] provide much information and software version about HMMer software package. The latest HMMer version is HMMer 2.2g. It provides many platform version, such like FreeBSD, OpenBSD, Solaris (Intel x86 and Sun SPARC), IBM AIX, Compaq Alpha, and Intel Linux. In this paper, Intel Linux version is used to install and conduct experiment. The installation steps are descried as below.

1. Download the HMMer package and select one version which you want use.
   - hmmer-2.2g.bin.intel-linux.tar
   - uncompress the tar file and will bring to a new directory which called hmmer-2.2g.bin.intel-linux
2. Go into the hmmer-2.2g.bin.intel-linux directory and type two command:
   - env CC = gcc CFLAGS="-06" ./configure
   - ./configure –with-pvm
   - The first command execute, if you want to change the choice of compilation flags (CFLAGS) or the compiler (CC). And the second command is order to include the optional PVM support.
3. Then copy all file which exist in the directory hmmer-2.2g.bin.intel-linux/binaries into /usr/local/bin, and then HMMer can be executing at any different directory.
4. At the master node, it must install full HMMer program. But at the other slave nodes, just need to copy hmmcalibrate-pvm, hmmsearch-pvm, hmmpfam-pvm into /usr/local/bin directory. It is not necessary to install complete HMMer programs on each slave node.
5. Use hmmindex to create a binary SSI index for HMM database Pfam_fs. Type a command like below: (At server node)
   - hmmindex /home/ct/pack/Pfam_fs
   - Hint: The query sequence Artemia.fa is store in /home/ct/pack and the HMMer database Pfam_fs is store in /home/ct/pack/pfam

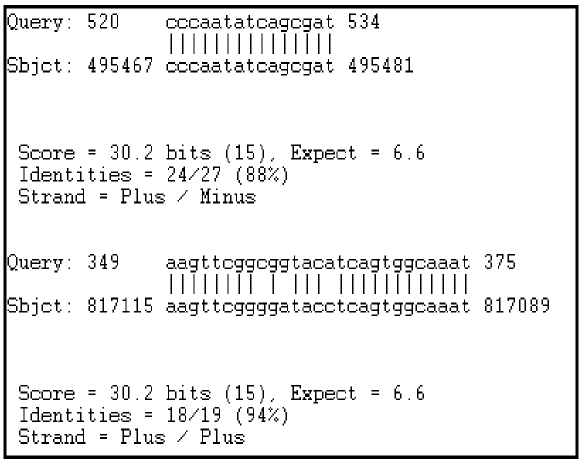
6. Set execution path way (At all nodes)
   - cd /usr/share/pvm3/bin/LINUX
   - And type one command ln –s /home/ct/hmmpfam-pvm
7. Then the multiple alignment tool can use now. Type a command to execution HMMer software such like below.
   - hmmpfam --pvm /home/ct/pack/pfam/Pfam_fs /home/ct/pack/Artemia.fa

   And we can get some message about result.

4 Experimental Results

4.1 The Performance of mpiBLAST

The mpiBLAST sequence alignment result message is such like below. The result messages are showed the alignment score and E-value, which is calculating by BLOSUM45 matrix. And the sequence alignment results express which base pair is match, which base pair is unmatched, and the inserted gap. According these messages, we can understand the degree of similarity about the sequences as shown in the following:



```
Query: 520    cccaatatcagcgat 534
              |||||||||||||||
Sbjct: 495467 cccaatatcagcgat 495481



 Score = 30.2 bits (15), Expect = 6.6
 Identities = 24/27 (88%)
 Strand = Plus / Minus


Query: 349    aagttcggcggtacatcagtggcaaat 375
              |||||||| | ||| ||||||||||||
Sbjct: 817115 aagttcggggatacctcagtggcaaat 817089



 Score = 30.2 bits (15), Expect = 6.6
 Identities = 18/19 (94%)
 Strand = Plus / Plus
```

nodes	2	4	6	8
1	1478	1109	873	807
2	1471	1166	900	799
3	1485	1195	860	809
4	1498	1100	869	803

5	1494	1167	866	808
Avg(ms)	1485. 2	1147. 4	873.6	805.2

Table 1: The execution time of mpiBLAST using nodes from 2 to 8.

The Table 1 is the mpiBLAST software experimental result. The software is execution by 2, 4, 6, and 8 nodes to compare the time. In order to aspire get more accurate data, we execute 5 times per experiment and calculate the average time. And By the Table 1 and the Figure 2, we can easily to nose out the parallel system can save more time to do the sequence alignment. According to the Table 1 and Figure 2, we can easily to discover the speedup is near two times as we used the fore nodes to executing the mpiBLAST.

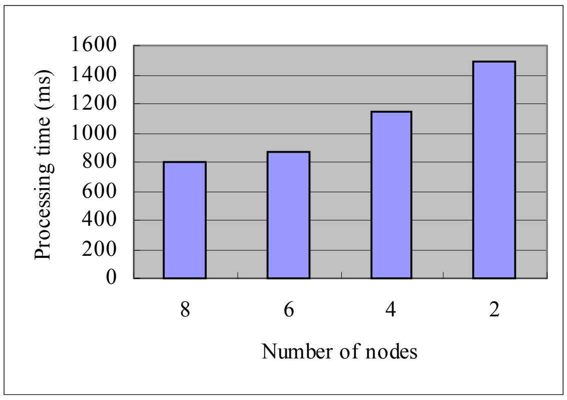


Figure 2: The average execution time of mpiBLAST using node from 2 to 8.

4.2 The Performance of HMMer

The HMMer sequence alignment result messages:

```
TLC: domain 1 of 1, from 1184 to 1207: score -0.6, E = 2.6
                   *->tskeekffsKlraiiWPIeryELk<-*
                      t    e+ +f +++ ++ P+ ++ Lk
      S13421    1184       TNVERRHFQAFSNALIPVMQHDLK       1207

Adeno_Penton_B: domain 1 of 1, from 1277 to 1288: score -0.5, E = 1.1
                   *->tDHGtlPLkNsL<-*
                      tDHG +PL+++L
      S13421    1277       TDHGYQPLFSNL       1288

Macscav_rec: domain 2 of 2, from 1309 to 1323: score 0.4, E = 10
                   *->VLLQLnsLissvqeh<-*
                      V+ QL +L+ s+q
      S13421    1309       VMAQLDTLVGSLQNS       1323

FliD: domain 1 of 1, from 1310 to 1322: score 1.5, E = 0.47
                   *->FtamDtlmgkmne<-*
                      + ++Dtl+g++++
      S13421    1310       MAQLDTLVGSLQN       1322
```

The result messages are showed the alignment score and E-value. And the sequence alignment results express which base pair is match, which base pair is unmatched, and the inserted gap. According these messages, we can understand the degree of similarity about the sequences.

According the time measurement, we can get some conclusion about the HMMers execution on PC cluster.

Processors number	**2**	**4**	**6**	**8**	**10**	**12**	**14**	**16**
1	3m30.749	1m51.007	1m15.694	57.512	46.344	39.215	33.961	30.808
2	3m32.231	1m51.541	1m15.602	57.144	46.973	39.065	33.315	29.876
3	3m32.071	1m50.748	1m15.754	58.309	46.936	38.579	32.824	29.371
4	3m31.890	1m50.687	1m15.366	57.817	46.267	38.542	32.973	29.806
5	3m31.260	1m51.292	1m16.140	57.898	45.891	38.921	33.566	29.237
Avg (ms)	211640.2	111055	75711.2	57736	46482.2	38864.4	33327.8	29819.6

Table 2: The execution time of HMMer using processor from 2 to 16.

The Table 2 is the experimental result about 2, 4, 6, 8, 10, 12, 14, 16-processor's execution time. And every experiment is executing five times and calculating its average. By Table 2, it is get more speed up and save more time to do multiple sequences alignments. Let us look the time of 2 processors and the time of 16 processors, it saves near 3 minutes. And it does not spend much money to construct this PC cluster system.

And according to this experimental data, draw a bar chart. By this bar chart, we can easier to find out the time cost change. When we use 4 processors to executing the software, it saved about a half time. So the speedup is near two degree which compare with 2 processors. And it produced near two degree speedup as we used 8 processors which compares with 4 processors. The Graph is draw below (Figure 3).

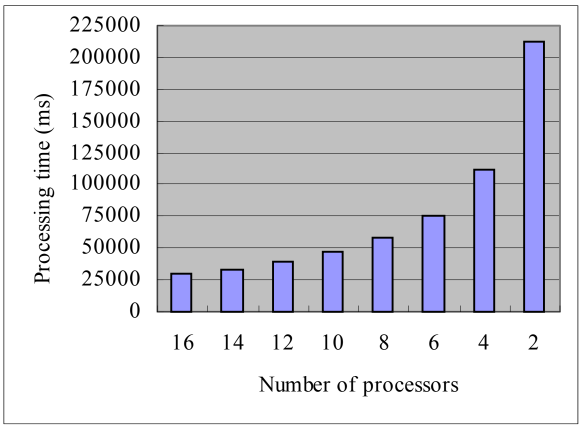


Figure 3: The average execution time of HMMer using processor from 2 to 16.

5 Conclusion and Future Work

In this paper, we have shown a great improvement about multiple sequences alignment time costs. But we know that the speed up is not enough, because we don't have so much PC can build a more high performance PC Cluster. The present day, biology data is increasing quickly. According this reason, bioinformatics requires high performance computing facilities for homology search, molecular simulation, cell simulation et al. Grid computing has a potential for expansion in computing performance by connecting a large number of computers or PC clusters with high performance networks. So Grid system can help us speedup the experimentation time. In the future, we want to integrate the Grid technology and bioinformatics to implement all kind of sequence alignment applications or bioinformatics tools [26]. And this hybrid system sometimes called BioGrid. BioGrid is a large-scale distributed computing environment, including couple of computers, storage systems, and other devices. By this BioGrid computing system, we can speed up the sequence alignment time by many computers of the grid system. And BioGrid system can improve performance more than parallel computing on PC Clusters. In this paper, the mpiBLAST bioinformatics application software is used on MPI, and we can re-compile it by MPICH-G2. And the mpiBLAST can execution on Grid System. So construct the BioGrid System is our future work and hope to accelerate the sequence alignment time by it.

References

[1] R. Buyya (1999), *High Performance Cluster Computing: System and Architectures*, Vol. **1**, Prentice Hall PTR, NJ.

[2] R. Buyya (1999), *High Performance Cluster Computing: Programming and Applications*, Vol. **2**, Prentice Hall PTR, NJ.

[3] http://www.netlib.org/benchmark/hpl, HPL – A Portable Implementation of the High-Performance Linpack Benchmark for Distributed-Memory Computers.

[4] http://www.lam-mpi.org, LAM/MPI Parallel Computing.

[5] http://www.haveland.com/povbench, POVBENCH – The Official Home Page.

[6] http://parlweb.parl.clemson.edu/pvfs, Parallel Virtual File System.

[7] http://www.epm.ornl.gov/pvm, PVM – Parallel Virtual Machine.

[8] Lie, W. N. (2001), *Distributed Computing Systems for Satellite Image Processing, Technical Report*, EE, National Chung Cheng University.

[9] Lillesand, Thomas M. and Kiefer, Ralph W. (1994), *Remote Sensing and Image Interpretation*, Third Edition, John Wiley & Sons.

[10] Richards, John A. (1999), *Remote Sensing Digital Image Analysis: An Introduction*, Springer-Verlag.

[11] T. L. Sterling, J. Salmon, D. J. Backer, and D. F. Savarese (1999), *How to Build a Beowulf: A Guide to the Implementation and Application of PC Clusters*, 2nd Printing, MIT Press, Cambridge, Massachusetts, USA.

[12] B. Wilkinson and M. Allen (1999), *Parallel Programming: Techniques and Applications Using Networked Workstations and Parallel Computers*, Prentice Hall PTR, NJ.

[13] M. Wolfe (1996), *High-Performance Compilers for Parallel Computing*, Addison-Wesley Publishing, NY.

[14] C. T. Yang, S. S. Tseng, M. C. Hsiao, and S. H. Kao (1999), “A Portable parallelizing compiler with loop partitioning,” Proc. of the NSC ROC(A), Vol. **23**, No. 6, pp. 751-765.

[15] Chao-Tung Yang, Shian-Shyong Tseng, Yun-Woei Fan, Ting-Ku Tsai, Ming-Hui Hsieh, and Cheng-Tien Wu (2001), “Using Knowledge-based Systems for research on portable parallelizing compilers,” Concurrency and Computation: Practice and Experience, vol. **13**, pp. 181-208.

[16] Chao-Tung Yang, Chi-Chu Hung, and Chia-Cheng Soong (July 2001), “Parallel Computing on Low-Cost PC-Based SMPs Clusters,” Proc. of the 2001 International Conference on Parallel and Distributed Computing, Applications, and Techniques (PDCAT 2001), Taipei, Taiwan, pp 149-156.

[17] Chao-Tung Yang and Chi-Chu Hung (Dec. 2001), “High-Performance Computing on Low-Cost PC-Based SMPs Clusters,” Proc. of the 2001 National Computer Symposium (NCS 2001), Taipei, Taiwan, pp 149-156.

[18] http://hmmer.wustl.edu/ Washington University in St.Louis

[19] http://pfam.wustl.edu/ Pfam Database Home

[20] PIERRE BALDI, YVES CHAUVIN, TIM HUNKAPILLER, and MARCELLA A. McCLURE. *Hidden Markov models of biological primary sequence information.*

[21] Lior Pachter, Marina Alexandersson, and Simon Cawley. *Applications of Generalized Pair Hidden Markov Models to Alignment and Gene Finding Problems.*

[22] Oswaldo Trelles. *On the Parallelization of Bioinformatic Applications.*

[23] Jens Kleinjung, Nigel Douglas and Jaap Heringa. *Parallelized multiple alignment.*

[24] Thomas Royce and Rance Necaise. *A parallel Algorithm for DNA Alignment*

[25] Heiko A. Schmidt, Korbinian Strimmer, Martin Vingron and Arndt von Haeseler. *TREE-PUZZLE: maximum likelihood phylogenetic analysis using quartets and parallel computing*.

[26] Michael Karo, Christopher Dwan, John Freeman, Jon Weissman, and Miron Livny, *Ernest Retzel. Applying Grid Technologies to Bioinformatics*.

[27] PARACEL BLAST-Accelerated BLAST software optimized for Linux clusters.

[28] Dmitry Korkin. A *New Dominant Point-Based Parallel Algorithm for Multiple Longest Common Subsequence Problem*.

[29] Altschul, S.F., W. Miller, E. W. Myers, and D. J. Lipman. (1990). *Basic local alignment search tool*. J. Mol. Biol. 215:403-410.

[30] http://www.ncbi.nlm.nih.gov/BLAST/, NCBI BLAST main page

[31] http://www.ncbi.nlm.nih.gov/Education/BLASTinfo/information3.html, NCBI BLAST information guide

[32] http://mpiblast.lanl.gov/index.html, mpiBLAST main page

[33] ftp://ftp.ncbi.nih.gov/toolbox/ncbi_tools/, NCBI Toolbox download site

[34] Karsten Hokamp, Denis C. Shields, Kenneth H. Wolfe and Daniel R. Caffrey. *Wrapping up BLAST and other applications for use on Unix clusters*.

[35] Kuo-Bin Li, and ClustalW-MPI: *ClustalW Analysis Using Distributed and Parallel Computing*

平行生物資訊軟體在個人電腦叢集上之應用

楊朝棟[*]　　郭育倫

摘　　要

除了傳統的大型平行電腦之外，由於許多高效能處理器的誕生以及擁有高速傳輸頻寬的網路和許多有用發展工具的出現，使得分散式電腦叢集在現今的科學計算領域中扮演了非常重要的角色。如我們所知，生物資訊領域的相關軟體可以加速巨量序列資料的分析，尤其是針對序列的分析比對。因此我們利用了目前處理器價格低廉的特點，利用八台雙處理器的電腦建構一組主從式架構的個人電腦叢集以作為生物資訊之計算平台。在此篇論文中，利用 mpiBLAST 與 HMMer 兩種平行版本的生物資訊應用軟體，並且紀錄與比較其序列比對所需的時間與效能。而論文中也會提及關於此系統之架構以及叢集電腦的效能分析等。

關鍵詞：平行計算、生物資訊、BLAST、HMMer、個人電腦叢集、加速

[*] 高效能計算實驗室，東海大學資訊工程與科學系