Exploiting Multi-level Parallelism for Homology Search using General Purpose Processors

Xiandong Meng
Electrical and Computer Engineering
Wayne State University
MI, Detroit 48202
meng@ece.eng.wayne.edu

Vipin Chaudhary
Institute for Scientific Computing
Wayne State University
Detroit, MI 48202
vipin@wayne.edu

Abstract

New biological experimental techniques are continuing to generate large amounts of data using DNA, RNA, human genome and protein sequences. The quantity and quality of data from these experiments makes analyses of their results very time consuming, expensive and impractical. Searching on DNA and protein databases using sequence comparison algorithms has become one of the most powerful techniques to better understand the functionality of particular DNA, RNA, genome, or protein sequence. This paper presents a technique to effectively combine fine and coarse grain parallelism using general purpose processors for sequence homology database searches. The results show that the classic Smith Waterman sequence alignment algorithm achieves super linear performance with proper scheduling and multi level parallel computing at no additional cost.

1. Introduction

High-throughput technologies in the field of biology have led to an exponential growth in the amount of data generated over the past several years that has far exceeded the growth in processor performance. Sequence comparison algorithms [3] based on the dynamic programming method such as the Needleman-Wunsch [16] and Smith-Waterman algorithms [20], provide optimal solutions. However, they are very computationally expensive. For this reason, heuristics based algorithms, such as BLAST [2], FASTA [17] etc., although sub-optimal, are widely used. The biology community would rather use the accurate methods provided there existed cheap and fast accurate solutions. A number of different designs for special-purpose hardware [8][12][14][19] for performing sequence alignments and database searching have been proposed and implemented. Their advantage over general-purpose computers is that they can be tailored specifically to perform sequence comparisons at a high speed, while the disadvantage is high cost.

The first widely used program for database similarity searching was FASTA [17]. FASTA stands for FAST-All, reflecting the fact that it can be used for a fast protein comparison or a fast nucleotide comparison between a query sequence and a large database of known sequences. This program achieves a high level of sensitivity for similarity searching at high speed. OSEARCH and SSEARCH [18] are two Smith-Waterman implementations in FASTA programs. OSEARCH is straightforward Smith-waterman implementation. SSEARCH [18] is an optimized implementation of Smith-Waterman algorithm that is approximately twice as fast as OSEARCH. However, OSEARCH is more sensitive and accurate.

Traditional approaches to sequence homology searches using Smith-waterman algorithm on general-purpose processor have proven to be too slow to keep up with the current increasing rate of sequence database. Therefore, many approaches [10] [11] [17] to parallelizing the FASTA and SSEARCH have been investigated. In this paper a method to combine fine grain and coarse grain parallism among cluster nodes is presented.

This paper is structured as follows. Section 2 discusses the sequence homology search relevant details on Smith-Waterman algorithm, fine and coarse grain parallelism, and scheduling. In Sections 3 and 4 we describe the experimental infrastructure and our implementation methodology, respectively. In Section 5 we present the performance results. Section 6 discusses the related work. We end with conclusions in section 7.

2. Background

In this section we present a brief background on Smith-Waterman algorithm and the various levels of parallelism that can be exploited in its implementation.

Authorized licensed use limited to: SUNY Buffalo. Downloaded on October 22, 2008 at 15:36 from IEEE Xplore. Restrictions apply.

2.1 Smith-waterman algorithm

The Smith-Waterman algorithm [20] is perhaps the most widely used local similarity algorithm for biological sequence pairwise alignment. It was enhanced by Gotoh [7]. Pairwise alignment is the alignment of two sequences. In Smith-Waterman database searches, the dynamic programming method is used to compare every database sequence to the query sequence and assign a score to each result. The dynamic programming method checks every possible alignment between two given sequences. The two sequences define a matrix in which every cell represents the alignment of two specific positions in the two sequences. The value of each cell depends on the residues located in these positions.

Scores in the first row and column are defined as zero. Entries $L(i, j)$ in all other cells of the matrix are defined as the score of the best alignment ending in the position matching x_i and y_j, and are calculated using the following recurrences:

$$L(i, j) = max\{E(i,j), L(i-1, j-1)+s(x_i, y_j), F(i, j), 0\};$$

where

$$E(i, j)=max\{ L(i, j-1)+a, E(i, j-1)+b\}$$
$$F(i, j)=max\{ L(i-1, j)+a, F(i-1, j)+b\}$$

where $s(x_i, y_j)$ is the score of a match or mismatch between x_i and y_j. In the above equations, a is the opened gap penalty and b is the extended gap penalty.

2.2 Fine grain, coarse grain parallelism and scheduling

To exploit fine-grain parallelism [21], the processors work together to compute the above $L(i, j)$ matrix, cell by cell. This fine-grain computation would require a very large number of processors if a long sequence were to be considered. The availability of SIMD vector instructions in common processors such as the MMX SSE2 from Intel, MMX 3Dnow! from AMD, and VIS from SUN allows fine grain parallelism to be exploited for a single pairwise alignment.

For coarse grain parallelism, the database is divided into blocks of sequences. These blocks can be assigned to the processors (workers). Coarse grain in this context means each processor performs a selected number of comparisons.

The objective of scheduling is to find a policy for assigning processors to tasks so that the overall execution time for database searching is minimal. However, any parallel strategy represents a trade-off between reducing communication time and improving the computational load balance.

3. Hardware and software platform

The hardware used for the experiments is based on Intel Xeon architecture running a distribution of NPACI Rocks (v3.1.0) Linux cluster. The MPI layer is based on MPICH v1.2.6 [1] with the ch_p4 device. The cluster consists of a master node and 16 compute nodes with 100Mb Ethernet connections between the nodes. Each node is an Intel Pentium IV Xeon System, which has dual 2.66 GHz processors with 2.5GB of RAM. Each node has two processes mapped to it. All cluster nodes are identical and have no other applications running at the time of the experiment.

FASTA was ported onto the above Linux cluster using the parallel MPI version (release 3.4) of SSEARCH program, which is an open resource software and can be download from EBI web site [5].

4. Implementation

4.1 Multi-level parallelism

The designed implementation takes advantages of combinations of fine and coarse grain parallelism paradigm within a parallel architecture, using general-purpose processors. The structure of the sequence homology searches and dynamic programming lead to many opportunities for parallel computation. At the lowest level, micro-parallelism techniques have been used to take advantage of specific features of parallel architecture, e.g., SIMD vector instructions. At higher level, the searching of large database leads itself to an implementation with multiple searches distributed across separate processors.

The computing requirements for these similarity search problems can only be tackled by using the computing resources efficiently. The SSEARCH and OSEARCH programs are first analyzed by Intel Vtune® performance analyzer. The performance reports indicate that the core of sequence alignment algorithm normally takes more than 90% of total execution time when performing a big database search. Parallel and distributed computing at the cluster level along with vector computing at the instruction level has to be utilized to create an effective solution to overcome this computation bottleneck.

4.2 Fine grain parallelism

Intel's Streaming SIMD Extensions 2 (SSE2) instruction set [9] enhances the SIMD instructions previously delivered with MMX and SSE technology. The key benefits of SSE2 are the added support for 64-bit double-precision floating point and for 64, 32, 16 and 8-bit integer operations on the eight 128-bit XMM registers first introduced with SSE.

Authorized licensed use limited to: SUNY Buffalo. Downloaded on October 22, 2008 at 15:36 from IEEE Xplore. Restrictions apply.

In this paper, our implementation exploits SSE2 instructions to effectively harness the power of the Pentium IV processor. SSE2 was introduced into the Intel's IA-32 architecture in the Pentium IV and Intel Xeon processors. These extensions were introduced to enhance the performance for advanced 3-D graphics, video decoding encoding, and speech recognition.

We wrote the FASTA program using SSE2 intrinsics library and compiled with Intel 8.1 C compiler on Red Hat v8.0 Linux cluster. The instrinsics library provides a C programming language interface to the SSE2 instructions. All the SSE2 instructions have a corresponding C function in the intrinsics library.

4.2.1 Vector computing along row with the query sequence The sequentially calculated values, e.g. *L, E* and *F*, have been grouped into a vector. The computation flows from top row to the bottom row. Due to the aforementioned data dependency, the vectors have to follow this order and the internal dependency has to be solved before moving to the next vector.

Since the computation always needs the data from previous calculated row, there are two arrays to hold the *L* and *E* values, whose equations are listed in section 2. The arrays are dynamically allocated and the size of the array is equal to the length of the query sequence. In other words, the vectors have to save their intermediate results to these two arrays for the next row reference.

4.2.2 Data padding and alignment Query sequence partition explores maximum parallelism along the computation flow. Normally the size of query sequence is not multiples of eight. If these marginal data are ignored, the final alignment scores may not be accurate and may miss some important biological information. One way is to create a clean loop to compute the rest of several cells, but it returns to the slow sequential computing. Considering a big database with millions of residues, we still want to solve the marginal data with parallel method. Therefore data padding is applied in our technique to eliminate the margin interference, which overcomes the problem of vector partition. Therefore, our technique can be applied to any query sequences of any size. The experiments show that query sequence partition with data padding can achieve significant performance gain.

In order to access the data fast, all used memories including arrays and score matrix are aligned on the 16-byte boundaries. Then SSE2 instructions are able to store or load 128-bit memory as fast as possible. The aligned memory plays critical roles on the speedup of SSE2 implementation.

4.2.3 Eight-way parallel processing with 16-bit values Using the SSE2 instructions, a 128-bit wide integer register of a CPU can be divided into sixteen smaller 8-bit units or eight 16-bit units, where the same arithmetic or logical operation can be performed simultaneously and independently on the data in each of the individual units. In this way, it allows fine grain parallelism to be exploited for a single pair wise alignment. If we choose to divide the XMM registers into as many as possible, i.e., sixteen 8-bit units, it increases the amount of parallelism, but limiting the maximum alignment score to 255. For medium, larger-sized and high similarity sequences, this limit is too restrictive. Therefore, we apply eight 16-bit units rather than sixteen 8-bit units, which increase the data accuracy in the range of -32768 to +32767. We believe this range is enough for high similarity homology searches.

4.3 Coarse grain parallelism – cluster computing

A cluster of workstations connected by an Ethernet network, is a very appropriate platform for sequence database searches, because of the independence between the different sequences in the database. The parallel OSEARCH, SSERACH and our SSE2 implementations were implemented using the MPI library. These parallel programs are data parallel implementations with the master worker approach in which one processor acts as master and the other processors as workers. The master processor reads the query, the database, and the number of workers, *n*, and splits the database into *n-1* parts, which is distributed among the *n-1* workers so as to search the database in parallel. After searching the database, the workers send the calculated scores to the master, which further sorts the scores and displays the alignments. The scalability, which is the ability to yield good performance with an increasing number of workers, is analyzed in the next section.

5. Performance evaluation

The performance of our parallel SSE2 implementation was evaluated using various query sequence lengths and various database sizes. All test amino acids sequences and databases were downloaded from the NCBI FTP site [15]. A set of amino acid query sequences (P07305 194aa, NP_001008227 398 aa, NP_033963 724 aa, NP_116653 1760 aa) is used to test the effect of various query sequence sizes. Scanning a large database is time consuming and the most heavily used bioinformatic application. We select the following three databases to verify the performance of parallel scalability on various database sizes. Month.gss is 214 MB having 169486546 residues in 298731 sequences. Nr is 891 MB having 593787265 residues in 1798171 sequences. And patnt is 1.55 GB having 1370828404 residues in 2365892 sequences.

The parallel Smith-Waterman's running time can be decomposed into five primary components [4]: MPI

Authorized licensed use limited to: SUNY Buffalo. Downloaded on October 22, 2008 at 15:36 from IEEE Xplore. Restrictions apply.

initialization, database fragment copying time, Smith-Waterman search time, communication time, and resulting merging and printing time. Since we optimized Smith-Waterman database searching part without any changes on other parts, the performance evaluation and comparison are based on this part only.

5.1 Experiment results

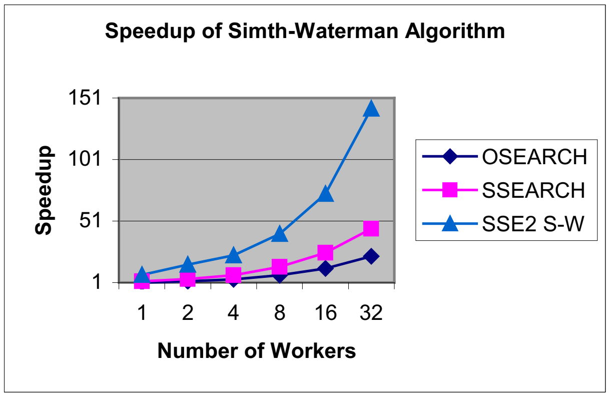

Figure 1: Speedup of SSE2 Smith-waterman on 32-processor Linux cluster

We present a series of results that detail the homology searches parallel speedups vs. the number of processes used by the three programs. We first begin with the analysis of the OSEARCH, which sets a baseline for the analysis of the performance in the other programs. The speedups of SSEARCH and our parallel SSE2 Smith-Waterman are compared to OSEARCH. Overall, we see that short query and small database cause the performance to degrade. Performance is better overall, when the long query sequence and large sequence database are used. Our parallel SSE2 implementation can obtain additional three times performance gain compared to SSEARCH, and six times as compared to OSEARCH.

Figure 1 speedup curves show the performance of a single query sequence of length 1760 amino acids searched against a 214MB database. The workers (processors) range from 1 up to 32. Compared to OSEARCH and SSEARCH, the SSE2 implementation has very good scalability on 32-processor Linux cluster. A speedup of 143 using 32 workers compared to the sequential OSEARCH program is obtained. The total searching time is 38 seconds using 32 workers.

Figure 2 shows the performance of query sequences with various lengths of amino acids searched against a 214MB database using 32 workers. The speedups of our SSE2 implementation obtained are 43, 61, 80 and 143 for the queries of lengths 194, 398, 724, and 1760 amino acids, respectively. The results clearly illustrate that the

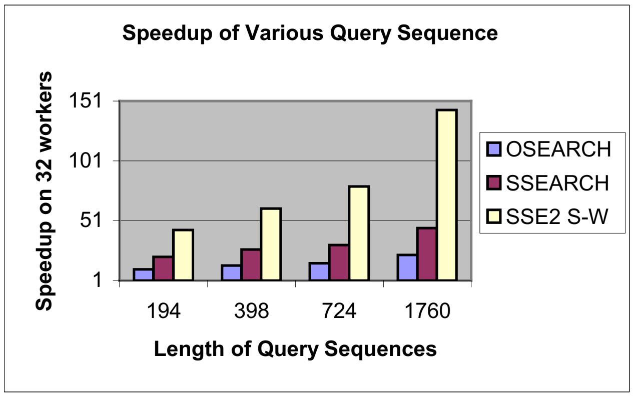

Figure 2: Speedup of various query sequence length on 32-processor Linux cluster

long sequences could be more effectively searched using the parallel SSE2 Smith-Waterman algorithm on 32 workers.

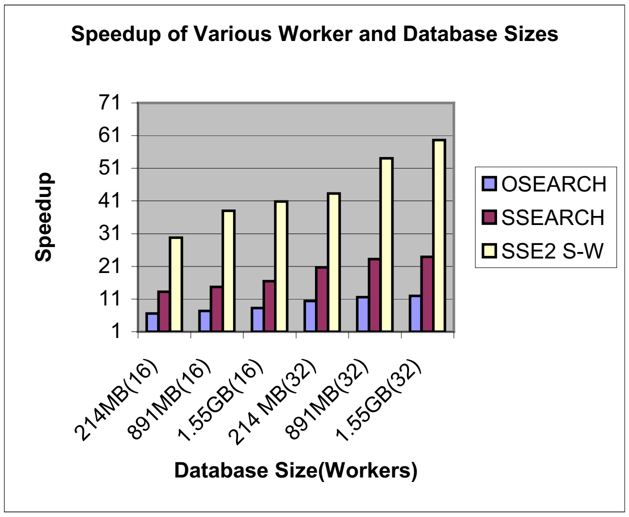

Figure 3: Speedup of various database sizes and workers on 32-processor Linux cluster

Query sequence of length 194 amino acids was searched against a 214MB, 891MB and 1.55 GB database with 16 and 32 workers. The above performance bar chart shows that speedup increases from 30 to 41 on 16 workers and from 43 to 60 on 32 workers with the database increase. While searching a specific query sequence against size varied databases using parallel SSE2 Smith-Waterman, better speedup was observed with larger database size.

Our parallel SSE2 Smith-Waterman implementation produces exactly the same outputs as those computed by OSEARCH, which is higher quality than SSEARCH, with big time saving and no additional hardware cost.

IEEE COMPUTER SOCIETY
Authorized licensed use limited to: SUNY Buffalo. Downloaded on October 22, 2008 at 15:36 from IEEE Xplore. Restrictions apply.

6. Related work on parallel sequence alignment

A group of fine-grained parallel designs on special-purpose hardware for performing sequence alignments and database searching have been proposed and implemented. estrel [12] is a 512-element linear parallel processor with 8-bit, SIMD processing elements. The single-board system was designed and built with particular emphasis on efficient high-throughput sequence analysis. Performance of the Smith-Waterman was reported at being 20 times faster than 433 MHz DEC Alpha series 21164. Yamaguchi and Maruyama [22] achieved high speed homology search using FPGA (Field Programmable Gate Array). They reported a time of 34 seconds for comparing a query of 2,048 elements with a database of 64 million, which is 330 times faster than a desktop with a 1 GHz Pentium III. Schmidt *et.al.* [19], use two new massively parallel architectures. The first architecture is built around a PC-cluster linked by network and fine-grained parallel Systola 1024 processor boards connected to each node. The second architecture is the Fuzion 150, a new parallel computer with a linear SIMD single array of 1535 processing elements on a single chip. Both architectures provide high throughput solutions at a good price performance ratio. Six-fold speedup of Smith-Waterman sequence database search using parallel processing on common microprocessors using MMX was reported [11]. Lander *et. al.* [13] implemented the algorithm on a CM-2 SIMD machine.

Parallel FASTA [17], parallel SSEARCH [18] are examples of coarse grain parallelism.

7. Conclusion and future work

We presented a new implementation of the Smith-Waterman algorithm that combines fine grain and coarse grain parallelism and multi-level scheduling and achieved a speedup of 143 on a cluster of 16 dual-CPU Pentium IV Xeons. This was six times faster than the currently available parallel implementations on a general purpose CPU cluster.

Our future work includes the impact of using faster networking technologies and different multi-level scheduling schemes on the speed up of the Smith-Waterman algorithm. We are also looking at implementations across multiple grids, making scheduling more important in efficient implementations.

References

[1] Argonne National Laboratory. MPICH – A portable implementation of MPI. http: wwwunix.mcs.anl.gov mpi mpich.

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

[3] Chaudhary, V., F. Liu, X. Meng, V. Matta, A. Nambiar, G. Yadav, and L. T. Yang, "Parallel Implementations of Local Sequence Alignment: Hardware and Software", in Parallel Computing in Bioinformatics and Computational Biology, Editor: Albert Zomaya, John Wiley and Sons, 2005 (to appear).

[4] Darling, A. E., Carey L., Feng W., "The Design, Implementation, and Evaluation of mpiBLAST", ClusterWorld Conference&Expo in conjunction with the 4th International Conference on Linux Clusters: The HPC Revolution 2003, June 2003.

[5] EBI FASTA Programs download ftp: ftp.ebi.ac.uk pub software unix fasta

[6] Grate, L., Diekhan, M., Dahle, D. and Hughey, H. Sequence Analysis With the estrel SIMD parallel Processor.Pacific Symposium on Biocomputing 2001 pp.263-74

[7] Gotoh, O. An improved algorithm for matching biological sequences J. Mol.Biol.(1982) 162, 705-708

[8] Hughey, R. Parallel hardware for sequence comparison and alignment. (1996) Comput. Appl. Biosci. 12, 473-479

[9] IA-32 Intel® Architecture Software Developer's Manual Volume 1: Basic Architecture

[10] Janaki, C. and Joshi, R. R., Accelerating comparative genomics using parallel computing in Silico Biology 3, 0036 (2003); ©2003, Bioinformation Systems e.V.

[11] Rogens, T. and Seeberg, E., Six-fold speed-up of Smith-Waterman sequence database searches using parallel processing on common microprocessors. Bioinformatics, 16, (2000), 699-706.

[12] Grate, L., Diekhan, M., Dahle, D. and Hughey, H. Sequence Analysis With the estrel SIMD parallel Processor.Pacific Symposium on Biocomputing 2001 pp.263-74

[13] Lander, E., Mesirov, J.P. and Taylor W., Protein sequence comparison on a data parallel computer. Proc. Of 1988 International conference on parallel processing pp. 257-263

[14] Meng, X. and Chaudhary, V., Bio-Sequence Analysis with Cradle's 3SoC Software Scalable System on Chip In Proceedings of the ACM Symposium on Applied Computing (SAC) SAC'04, March 14-17, 2004, Nicosia, Cyprus.

[15] NCBI DATABASE Download http: www.ncbi.nlm.nih.gov Ftp index.html

[16] Needleman, S. and Wunsch, C. A general method applicable to the search for similarities in the amino acid sequence of two sequences. . J. Mol. Biol., 48(3), (1970), 443-453

[17] Pearson, W. R., Rapid and sensitive sequence comparison with FASTP and FASTA, Methods Enzymol. 183: 63-98, 1990.

[18] Pearson, W.R. Searching protein sequence libraries: comparison of the sensitivity and selectivity of the Smith-waterman and FASTA algorithms. Genomics, 11, 635-650

[19] Schmidt, B., Schroder, H. and Schimmler, M. Massively Parallel Solutions for Molecular Sequence Analysis, International Parallel and Distributed Processing Symposium: IPDPS Workshops (2002), p. 0186

[20] Smith, T.F. and Waterman, M.S. Identification of common molecular subsequences. J. Mol. Biol., 147, (1981), 195-197

[21] Trelles, O. On the Parallelization of Bioinformatic Application. Briefings in Bioinformatics (2001), vol.2, 2

[22] Yamaguchi Y., Maruyama, T. High Speed Homology Search with FPGA. Pacific Symposium on Biocomputing 2002

Authorized licensed use limited to: SUNY Buffalo. Downloaded on October 22, 2008 at 15:36 from IEEE Xplore. Restrictions apply.