BioBench: A Benchmark Suite of Bioinformatics Applications

Kursad Albayraktaroglu, Aamer Jaleel, Xue Wu†, Manoj Franklin, Bruce Jacob, Chau-Wen Tseng† and Donald Yeung

Dept. of Electrical & Computer Engineering
University of Maryland, College Park
{kursad,ajaleel,manoj,blj,yeung}@eng.umd.edu

†Dept. of Computer Science,
University of Maryland, College Park
{wu,tseng}@cs.umd.edu

Abstract

Recent advances in bioinformatics and the significant increase in computational power available to researchers have made it possible to make better use of the vast amounts of genetic data that has been collected over the last two decades. As the uses of genetic data expand to include drug discovery and development of gene-based therapies, bioinformatics is destined to take its place in the forefront of scientific computing application domains. Despite the clear importance of this field, common bioinformatics applications and their implication on microarchitectural design have received scant attention from the computer architecture community so far.

The availability of a common set of bioinformatics benchmarks could be the first step to motivate further research in this crucial area. To this end, this paper presents BioBench, a benchmark suite that represents a diverse set of bioinformatics applications. The first version of BioBench includes applications from different application domains, with a particular emphasis on mature genomics applications. The applications in the benchmark are described briefly, and basic execution characteristics obtained on a real processor are presented.

Compared to SPEC INT and SPEC FP benchmarks, applications in BioBench display a higher percentage of load/store instructions, almost negligible floating point operation content, and higher IPC than either SPEC INT and SPEC FP applications. Our evaluation suggests that bioinformatics applications have distinctly different characteristics from the applications in both of the mentioned SPEC suites; and our findings indicate that bioinformatics workloads can benefit from architectural improvements to memory bandwidth and techniques that exploit their high levels of ILP.

The entire BioBench suite and accompanying reference data will be made freely available to researchers.

1. Introduction

The success of genome sequencing efforts and developments in bioinformatics resulted in a vast amount of data over the last two decades. As of early 2004, the number of genetic sequences in the GenBank gene sequence repository of The National Center for Biotechnology Information (NCBI) is 30.9 million genetic sequences and increasing rapidly: this number has increased more than tenfold between 1998 and 2003. Among the many remaining questions in biotechnology researchers' minds is the question of how this mountain of data can be put to use in the efforts for understanding life and developing new cures for life threatening health problems.

While the results of these efforts may be some years away, it is certain that success of future bioinformatics depends on high-performance computers to a great extent. As bioinformatics emerges as an important class of scientific computing applications, it is becoming more evident that further advances will render this field even more important for pharmaceutical research, protein structure prediction, and development of gene-based therapies. We expect that the performance of bioinformatics applications will therefore become an important factor in defining future high performance computing systems. While the design and analysis of faster algorithms for bioinformatics applications is a very active field of research, very little has been published in the literature on general performance characteristics of these applications and the implications on system or processor design. Most of the published work in this field seems to have focused on incremental improvements to bioinformatics application suites or certain algorithms. One reason behind the current disconnect between computational biology and computer architecture communities could be the lack of a standard benchmark suite of bioinformatics applications. We see a clear need for such a set of well-defined benchmark applications drawn from bioinformatics codes in common use, which will be an important step towards motivating further research on the characteristics of such applications and their implication on computer systems engineering.

This paper presents BioBench, a set of benchmark applications chosen to reflect the diversity of bioinformatics codes in common use. The applications in BioBench and the reference data sets were selected with input from the bioinformatics community, and we expect BioBench to evolve in response to future developments and comments from both bioinformatics and computer architecture communities. While the initial BioBench suite aims to provide tools to evaluate bioinformatics applications on uniprocessor systems, a parallel version is also planned for common multiprocessor architectures.

In addition to providing a benchmark for evaluating the performance of computer systems running common bioinformatics applications, a secondary goal of the BioBench suite is to establish bioinformatics applications as a distinctly different class of applications than the commonly accepted framework of scientific applications. In contrast to these scientific applications which typically are floating-point intensive, many bioinformatics applications operate with textual representations of biological sequence data. The straightforward encoding of this data can mean that many bioinformatics codes are primarily fast string search or pattern matching applications; and we have reason to expect distinctly different execution behavior for these benchmarks than traditional scientific application benchmarks, particularly with respect to the importance of floating point versus integer operations and branch behavior.

In the benchmark characterization part of our work, we obtain basic execution characteristics for the applications present in the BioBench suite, and compare these characteristics to those of SPEC 2000 benchmarks to test the validity of our expectations.

The rest of the paper is organized as follows: Section 2 discusses some previous work done in this field. Section 3 briefly describes major bioinformatics application domains represented in BioBench, and describes the applications. Section 4 describes our experimental methodology and tools used to obtain performance data. Section 5 presents this data, and compares the characteristics of BioBench applications to those of SPEC 2000 benchmarks. Finally, we present some concluding remarks in Section 6.

2. Previous Work

Many examples of application domain-specific benchmark suites have been proposed and some were widely accepted, following the example of the SPEC [13] suite for evaluation of integer and floating-point performance of computer systems. Among the most notable are the TPC benchmarks [17] for database/transaction processing, and more recently benchmark suites like MediaBench [10] or CommBench [18].

To the best of our knowledge, a comprehensive set of bioinformatics benchmarks has not been compiled and studied prior to our study. In contrast, studies on performance of individual algorithms or tools are abundant in literature. Most of the published work on performance studies of bioinformatics workloads involves either performance optimization of established algorithms, or analysis of the performance of such algorithms on parallel systems. Yap et al. [19] present a detailed study of parallel sequence searching. Catalyurek et al. [5] analyze performance of specific applications on a centralized-server, multi-client environment.

While we could not find any comprehensive academic study of multiple bioinformatics workloads, we noticed at least one publication on the subject from the industry: The SGI Bioinformatics Performance Report [1] includes several studies of uniprocessor and multiprocessor bioinformatics applications.

3. BioBench Suite Applications

An important goal of the BioBench effort was to define a representative set of bioinformatics application domains. We first identified several important application classes and selected commonly used applications from these classes. While a diverse set of benchmark applications was desirable, we limited the scope of this initial release of BioBench to

Table 1. BioBench benchmarks and instruction counts

Benchmark	Description	Instruction Count
blastn	DNA sequence searching	215,131,057,029
blastp	Protein sequence searching	514,628,929,894
clustalw	Multiple sequence alignment	2,150,900,967,391
fasta_dna	DNA sequence searching	1,001,512,078,272
fasta_prot	Protein sequence searching	1,149,078,024,873
hmmer	Sequence profile searching	1,573,753,830,214
mummer	Genome-level alignment	106,703,486,044
protpars	Phylogenetic analysis	1,730,029,486,107
tigr	Sequence assembly	862,484,000,000

relatively mature application classes that found widespread usage in academia and industry. In addition to widespread use, another important criterion in choosing benchmarks was the availability of source code for use in our studies, and in some cases a relatively less known application suite had to be chosen instead of a popular commercial suite. Equally important was the selection of input data that is representative of real-world computational biology problems. Problem sizes were determined in collaboration with members of the bioinformatics community, and our execution-based methodology allowed us to use complete copies of major protein and DNA databases instead of smaller data sets which would not be representative of real-world problems and could have skewed the results. As an example, the BLAST workload in BioBench was evaluated using NCBI's NT database, containing 11GB of data that represented all DNA sequences discovered to date.

We recognize that bioinformatics is a very diverse field, and the initial version of BioBench does not cover some important application domains like microarray analysis, protein structure prediction, protein docking and mass spectrometry analysis. In its initial version, the choice of BioBench applications reflects an emphasis on mature genomics tools. Future versions will address a much wider variety of bioinformatics application domains. As new application domains emerge, we plan to update BioBench with new benchmarks.

The application classes and the individual BioBench benchmarks selected to represent them are listed below. The applications are also listed in Table 1.

3.1. Sequence Similarity Searching

Sequence similarity searching applications are typically used to identify similarities between DNA or protein sequences, or to search for certain subsequences in large sequence databases. The similarity between two sequences (or the lack of it) can often reveal important clues about structural or functional relationships between them, and in some cases can provide important clues about common evolutionary roots of organisms. BioBench contains programs from both BLAST [3] and FASTA [12] suites for sequence similarity searching.

- **BLAST**: The most commonly used sequence searching application is represented by two programs, BLASTN and BLASTP, in the BioBench suite. These programs are used for DNA and protein sequence searching, respectively. We used the freely available version 1.3 of the BLAST suite. The DNA and protein databases used were NCBI's NT (11GB) and NR (945MB) databases containing the full set of non-redundant DNA and protein sequences submitted to NCBI.
- **FASTA:** BioBench includes the main search utility from University of Virginia's FASTA suite v3.4t21, the other important sequence searching suite [12]. To reflect the difference between protein and nucleotide (DNA) searches, our test cases use the FASTA application for searching against a DNA database and a protein database with suitable search sequences. The DNA database used in our study is a daily update file to the NCBI GenBank data repository (190MB), and the protein database used is the entire SwissPROT protein database (70MB).

3.2. Phylogenetic Analysis

Phylogenetic analysis aims to discover how a group of related protein sequences were derived from common origins during the process of evolution. This information is frequently displayed as a hierarchical diagram called a phylogenetic tree. The discovery and visualization of such relationships between proteins offers important clues on how certain traits were passed from species to species.

- **PHYLIP:** To represent phylogenetic analysis applications, we chose benchmarks from the PHYLIP suite [8], version 3.5c. PHYLIP is the most widely used phylogenetic analysis package, and contains several programs to conduct different types of phylogenetic analysis. We chose PROTPARS, a protein parsimony computation application.

3.3. Multiple Sequence Alignment

Multiple sequence alignment is the process of aligning more than two sequences to find regions of similarity. This kind of analysis is used to have a deeper understanding of similarity patterns that might suggest common origins between the proteins they code.

- **CLUSTAL W:** For our representative multiple-alignment benchmark, we chose the CLUSTAL W multiple sequence alignment application. CLUSTAL W [16] builds on the CLUSTAL package described in [9], and is currently the most commonly used multiple sequence alignment application.

3.4. Sequence Profile Searching

When an evolutionary diverse set of proteins are under investigation to find remotely related proteins, searching a sequence database for the consensus of a sequence family (a common signature of the family) can be more effective than searching the same database for individual sequences. This analysis approach is called sequence profile searching.

- **HMMER:** We chose the sequence profile searching package HMMER [7] to represent this class of applications in BioBench. HMMER uses profiles based on hidden Markov models to conduct searches against protein databases. We used HMMER v2.3 to search the SwissPROT protein database against the consensus of a small selection of protein sequences.

3.5. Genome-level Alignment

Genome-level alignment algorithms and tools are used to align complete genomes of related species. Due to the sheer number of nucleotides in a complete genome, multi-sequence alignment algorithms and tools (which are more geared toward aligning single proteins or simple nucleotide sequences) can not be used effectively for this task. Genome-level alignment tools employ algorithms specifically developed for the purpose of pairwise alignment of very large nucleotide sequences.

- **MUMMER:** MUMMER [6] is a genome-level alignment tool that has been used to assemble complete genomes. We chose MUMMER v3.14 for inclusion in BioBench.

3.6. Sequence Assembly

Sequence assembly tools are used to generate sequence data from many small overlapping partial sequences obtained by DNA sequencing hardware. Sequence assembly is a crucial step for using shotgun sequencing to obtain complete sequence data from physical DNA sequences.

- **TIGR:** The class of sequence assembly applications is represented by the TIGR Assembler [15] suite in BioBench. The version we used in BioBench was TIGR Assembler v2.

4. Methodology and Tools

Bioinformatics applications such as sequence similarity searching and multiple alignment are typically used in conjunction with very large databases, resulting in large execution times that are impractical for a simulator-based study. In order to meet our goal of collecting data on the entire execution of bioinformatics applications with meaningful input sizes, we chose to use the hardware performance counters built into modern microprocessors instead of a simulator.

Many modern microprocessors include special-purpose counters that can be used to count occurrences of different events and registers to access these counters. Among the many different events that can be counted are cache misses, branch mispredictions, and others that are useful measures of application performance. A particular drawback of hardware performance counters is their limited number: there were only 2 on the Intel Pentium 3 CPU used in our study. One workaround for this limitation is multiplexing, which uses time-sharing to use the counters to measure different events at different time slices, and extrapolates the result. For long running applications (which is typical for bioinformatics application workloads), the multiplexing method yields reasonably accurate measurements [11]. We used the PAPI hardware performance counter access library [4] that uses the *perfctr* Linux kernel patch for counter multiplexing. To facilitate data collection and analysis, we used the PerfSuite [2] utilities.

Using these software to utilize CPU performance counters, we were able to run unmodified BioBench applications with large input sizes characteristic of their typical use. We used a commodity workstation based on an Intel Pentium 3 CPU running Linux kernel 2.4.22; and PAPI v3.0. All BioBench programs were compiled using *gcc* version 2.95 on the same computer system used for data collection, at the *-O4* optimization level. To collect some low-level hardware performance counter data not collected by PerfSuite/PAPI, we also used the *brink/abyss* [14] toolset.

To provide a comparison to the SPEC benchmark suite, applications from the SPEC 2000 suite were also compiled using the same compiler and system using the default parameters. We collected execution characteristics using complete reference data input sets from the SPEC distribution, to be used for comparison against the BioBench benchmark applications.

Our execution-based methodology allowed us to collect precise performance characteristics on a real commodity processor for entire workloads that took up to 2.1 trillion instructions. The number of instructions for each benchmark in the BioBench suite is presented in Table 1. Some pertinent parameters of the Intel Pentium 3-based system used for our study are given in Table 2.

Table 2. Parameters of the system used in the study

Processor	Intel Pentium III
Clock Speed	700 MHz
Main Memory	512MB
L1 data cache	16KB, 4-way set assoc.
L1 instr. cache	16KB, 4-way set assoc.
L2 cache	256KB, 8-way set assoc.
Cache Line Size	32B

5. Benchmark Characteristics

To characterize the BioBench suite, we collected data on instruction profiles, basic block lengths, IPC, L1 and L2 data cache miss rates, and branch prediction accuracy. The same set of data was collected for the SPEC 2000 benchmarks for comparison. In the first phase of the evaluation, hardware performance counters were used to provide a count of instructions belonging to different major instruction classes in the x86 architecture. Instruction profiles for BioBench applications are given in Figure 1.

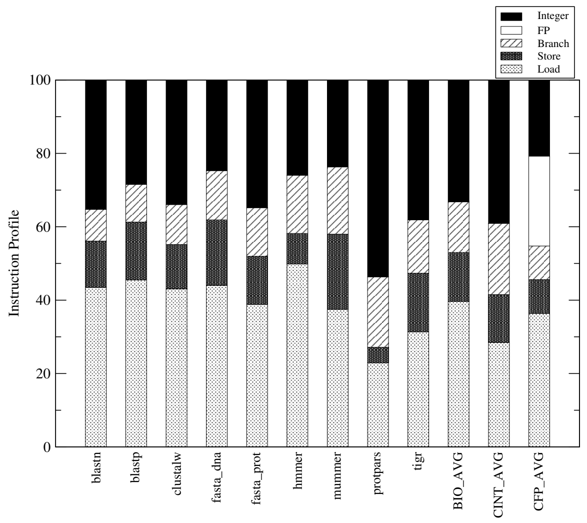

Figure 1. Instruction profiles for all BioBench benchmarks

For comparison, the average instruction class percentages for SPEC INT and FP benchmarks are also shown.

We observed that the floating point operation content of almost all BioBench applications are negligible. This finding reflects the intuition that bioinformatics applications are inherently different from mainstream scientific codes due to the representation of data they operate on. None of the BioBench workloads contained a floating point instruction content of more than 0.09 % of all instructions executed. While operating on primarily string data, most of the benchmarks do rely on some floating point computation for calculating statistics and likelihood values as part of their main algorithms, but this does not seem to constitute a significant part of the overall instruction count.

The average share of load instructions in BioBench applications has a marked difference from that of SPEC integer benchmarks, and these instructions constitute a larger portion of the instructions in BioBench than both classes of SPEC benchmarks. This implies that the amount of computation per datum is relatively small, a typical characteristic of search algorithms. Many of the BioBench components search through large input files and databases. The BioBench component with the lowest share of loads, *protpars*, was also the one benchmark with the smallest input file size in the benchmarks. (It is the second longest-running workload in BioBench, however.) *Protpars* essentially is less of a database search application than many of the BioBench components are, since its main function computes a tree-like hierarchy for related species using relatively shorter sections of sequences. This benchmark also differed from the rest of BioBench components with its larger share of integer ALU instructions, these instructions accounting for more than half of the instruction count. Similarly high share of load instructions was observed in one other non-search component, namely *mummer* which was found to be highly memory-bound with its dependence on very large suffix-tree data structures created in memory. The higher share of load/store operations in BioBench suggests that bioinformatics applications are likely to benefit from future architectures with higher memory bandwidth and prefetching.

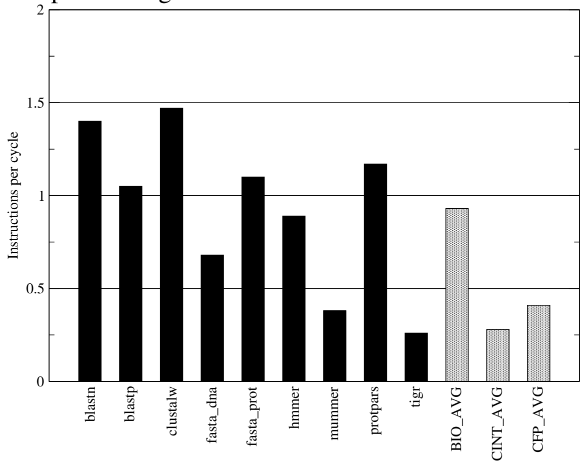

Figure 2. IPC values for all BioBench benchmarks

Figure 2 shows the IPC numbers for the applications in the BioBench suite. The significantly higher average IPC of the BioBench benchmarks hints at higher levels of ILP (instruction level parallelism) in the BioBench applications than the SPEC INT and FP benchmarks. This finding is encouraging, and along with our earlier finding of almost negligible floating point content in BioBench suggests that bioinformatics applications will benefit greatly from wider superscalars of the future with highly optimized fast integer cores. While we anticipated high levels of ILP in bioinformatics codes due to the often mentioned "*embarrassingly parallel*" nature of these programs, we did not expect to see this level of difference between BioBench and SPEC suite. We noticed considerable variation in the IPC values for the individual applications in BioBench, and our future work on BioBench will include a detailed analysis of performance differences between applications that are very similar in function and usage, a clear example being *blastn* and *fasta_dna* which execute essentially the same kind of search using two different algorithms.

The basic block length for BioBench applications is shown in Figure 3.

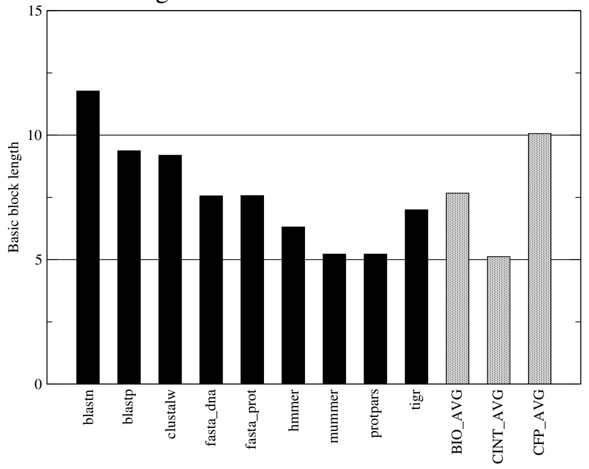

Figure 3. Basic block lengths for all BioBench benchmarks

On average, BioBench applications have a basic block length that lies roughly between those of the SPEC INT and SPEC FP averages; all individual BioBench benchmarks having higher basic block lengths than the SPEC INT average. The higher basic block length for applications in BioBench characterize bioinformatics applications as being closer to scientific workloads than in terms of the distribution of control transfer instructions.

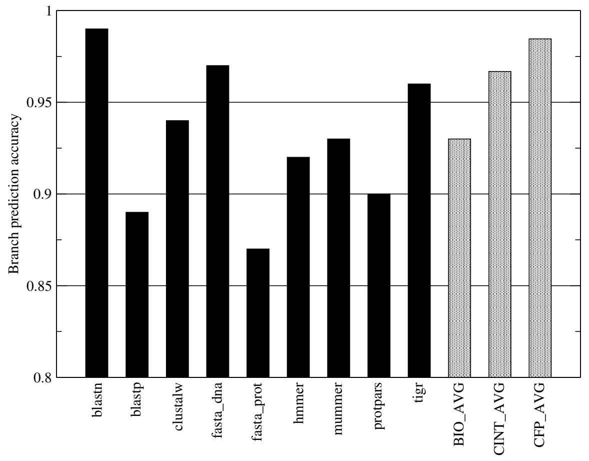

Figure 4. Branch prediction accuracy for all BioBench benchmarks

Figure 4 shows the branch prediction accuracy for the benchmarks. While the branch prediction accuracy for BioBench benchmarks is somewhat lower than that for SPEC benchmarks, the difference is not significant considering the very high prediction accuracy available with modern branch predictors.

L1 and L2 data cache miss rates are shown in Figures 5 and 6, respectively, and highlight differences in memory usage patterns of different BioBench components. The genome level alignment program *mummer* and the sequence assembly program *tigr* have higher L1 data cache miss rates than the rest of the applications in BioBench, a characteristic mirrored by their L2 data cache miss behaviors. These two applications had very high levels of memory utilization that eventually led us to scale the problem size for *mummer* down to be able to run it to completion on our test system with 512MB of main memory. In contrast, the multiple alignment component *clustalw* displayed very low L1 and L2 data cache miss rates. The component with largest duration of execution in our studies, *clustalw* displayed high IPC and fairly high average basic block length in addition to its low memory footprint. To our knowledge *clustalw* is one of the few commonly-used computational biology applications that had not been implemented in hardware before, and we believe its characteristics warrant a closer look at this benchmark as part of our future work.

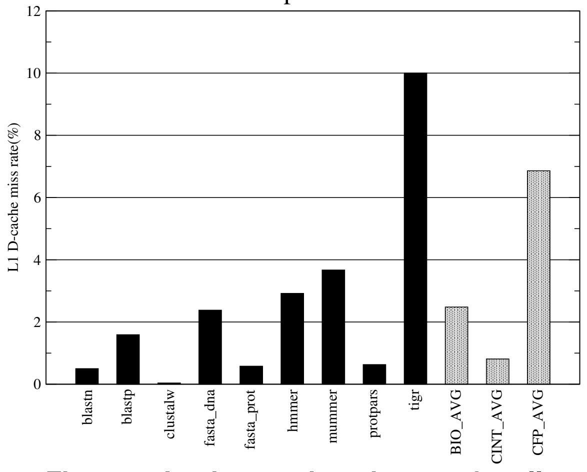

Figure 5. L1 data cache miss rate for all BioBench benchmarks

6. Concluding Remarks

In this paper, we identified and described important computational biology application categories and proposed BioBench, a benchmark of bioinformatics applications that represents relatively mature application classes with reference data that closely parallels real usage. BioBench applications and reference input data will be made available to researchers to allow them to evaluate their architectures using bioinformatics applications. We believe BioBench fills an imminent need for a well-defined set of benchmarks covering an important emerging class of applications.

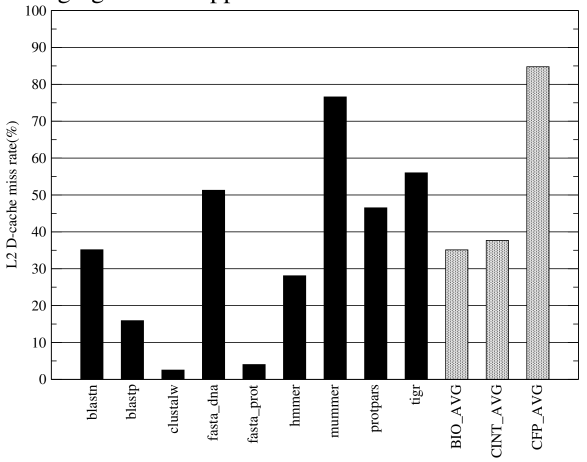

Figure 6. L2 data cache miss rate for all BioBench benchmarks

Our evaluation of BioBench components validated our intuition that bioinformatics applications have characteristics that distinguish them from traditional scientific computing applications characterized by SPEC FP benchmarks. Bioinformatics applications evaluated in this study displayed almost no significant floating point instructions and higher ILP while having basic block lengths closer to SPEC FP benchmarks than SPEC INT, implying similar regularity in distribution of branches. These findings indicate that bioinformatics applications stand to benefit from future architectural features such as increased memory bandwidth, memory prefetching and wider superscalars to exploit their high ILP.

Looking ahead, we plan to expand BioBench with benchmarks from several other emerging bioinformatics application domains in its next revision. Considering the parallelism available in bioinformatics workloads, a parallel version of BioBench would be a very valuable tool for studying the characteristics of these codes on multiprocessor systems and clusters, and such a version of BioBench is among our plans for future work in this field. In addition, we will be conducting studies on different levels of parallelism available in bioinformatics applications by studying BioBench components in detail to evaluate how such applications can be accelerated using thread-level parallelism techniques.

7. Acknowledgements

The authors would like to thank Charles F. Delwiche for his help during the early stages of this work. Manoj Franklin and Kursad Albayraktaroglu were supported in part by a Faculty Award from IBM Corporation. Donald Yeung was supported in part by NSF CAREER Award CCR-0000988.

10. References

[1] SGI Bioinformatics Performance Report. http://www.sgi.com/industries/sciences/chembio/pdf/bioperf01.pdf

[2] http://perfsuite.ncsa.uiuc.edu

[3] S. F. Altschul, W. Gish, W. Miller, E. W. Meyers, and D. J. Lipman. "Basic local alignment search tool", Journal of Molecular Biology, vol. 215, no.3, pp.403-410, October 1990.

[4] S. Browne, J. Dongarra, N. Garner, K. London, and P. Mucci. "A scalable cross-platform infrastructure for application performance tuning using hardware counters" in Proceedings of the 2000 ACM/IEEE Conference on Supercomputing, page 42, 2000.

[5] U. Catalyurek, E. Stahlberg, R. Ferreira, T. Kurc, and Joel Saltz, "Improving performance of multiple sequence alignment analysis in multi-client environments" in Online Proceedings of the 1st International Workshop on High Performance Computational Biology (HICOMB 2002).

[6] A.L. Delcher, S. Kasif, R.D. Fleischmann, J. Peterson, O. White, and S.L. Salzberg, "Alignment of whole genomes", Nucleic Acids Research, vol. 27, no. 11, pp. 2369-2376, 1999.

[7] S. R. Eddy, "Profile hidden markov models", Bioinformatics, vol. 14, no. 9, pp. 755-763, 1998.

[8] J. Felsenstein, "PHYLIP-phylogeny inference package (version 3.2)", Cladistics, vol. 5, pp.164-166, 1989.

[9] D. G. Higgins and P. M. Sharp, "CLUSTAL:a package for performing multiple sequence alignment on a microcomputer", Gene, vol.73, pp. 237-244,1988.

[10] C. Lee, M. Potkonjak, and W. H. Mangione-Smith, "Mediabench: A tool for evaluating and synthesizing multimedia and communicatons systems", in International Symposium on Microarchitecture (MICRO), pp. 330-335, 1997.

[11] J. M. May, "MPX: Software for multiplexing hardware performance counters in multithreaded programs" in Proceedings of the 15th International Parallel & Distributed Processing Systems Symposium (IPDPS), 2001.

[12] W. R. Pearson and D. J. Lipman. Improved tools for biological sequence comparison. Proc. Natl. Acad. Sci. USA, (8):2444?2448, April 1988.

[13] SPEC. SPEC Benchmark Suite Release 1.0. SPEC, 1989.

[14] B. Sprunt, "Managing the complexity of performance monitoring hardware: The brink and abyss approach", http://www.eg.bucknell.edu/~bsprunt/emon/brink_abyss/brink_abyss.shtm

[15] G. G. Sutton, O. White, M.D. Adams, and A.R. Kerlavage, "TIGR assembler: A new tool for assembling large shotgun sequencing projects", Genome Science and Technology, vol. 1, no. 2, pp. 9-19, 1995.

[16] J. D. Thompson, D. G. Higgins, and T.J. Gibson, "CLUSTAL W: improving the sensitivity of progressive multiple-sequence alignment through sequence weighting positions-specific gap penalties and weight matrix choice", Nucleic Acids Research, vol. 22, pp. 4673-4680, June 1994.

[17] TPC. TPC Benchmark A. Itom International Co., 1989.

[18] T. Wolf and M. Franklin, "Commbench - a telecommunications benchmark for network processors" in Proceedings of IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS), pp. 154-162, 2000.

[19] T. K. Yap, O. Frieder, and R. L. Martino, "Parallel computation in biological sequence analysis", IEEE Transactions on Parallel and Distributed Systems, vol. 9, no. 3, pp. 283-294, 1998.