BioPerf: A Benchmark Suite to Evaluate High-Performance Computer Architecture on Bioinformatics Applications

David A. Bader
College of Computing
Georgia Institute of Technology
Atlanta, GA 30332
bader@cc.gatech.edu

Yue Li
Department of ECE
University of Florida
Gainesville, FL 32611
yli@ecel.ufl.edu

Tao Li
Department of ECE
University of Florida
Gainesville, FL 32611
taoli@ece.ufl.edu

Vipin Sachdeva
Department of ECE
University of New Mexico
Albuquerque, NM 87131
vipin@ece.unm.edu

Abstract

The exponential growth in the amount of genomic data has spurred growing interest in large scale analysis of genetic information. Bioinformatics applications, which explore computational methods to allow researchers to sift through the massive biological data and extract useful information, are becoming increasingly important computer workloads. This paper presents BioPerf, a benchmark suite of representative bioinformatics applications to facilitate the design and evaluation of high-performance computer architectures for these emerging workloads. Currently, the BioPerf suite contains codes from 10 highly popular bioinformatics packages and covers the major fields of study in computational biology such as sequence comparison, phylogenetic reconstruction, protein structure prediction, and sequence homology & gene finding. We demonstrate the use of BioPerf by providing simulation points of pre-compiled Alpha binaries and with a performance study on IBM Power using IBM Mambo simulations cross-compared with Apple G5 executions.

The BioPerf suite (available from **www.bioperf.org***) includes benchmark source code, input datasets of various sizes, and information for compiling and using the benchmarks. Our benchmark suite includes parallel codes where available.*

1. Introduction

In the 50 years since the discovery of the structure of DNA, and with new techniques for sequencing the entire genome of organisms, biology is rapidly moving towards a data-intensive, computational science. Computational biology has been aided by recent advances in both technology and algorithms; for instance, the ability to sequence short contiguous strings of DNA and from these reconstruct the whole genome [1, 2, 3] and the proliferation of high-speed micro array, gene, and protein chips [4] for the study of gene expression and function determination. These high-throughput techniques have led to an exponential growth of available genomic data. For example, the National Center for Biotechnology Information (NCBI) GenBank, an annotated collection of all publicly available DNA sequences, has been growing at an exponential rate. There are 51,674,486,881 bases from 46,947,388 sequences in GenBank as of 15 August 2005 [5].

As genomics moves forward, having accessible computational methods with which to extract, view, and analyze genomic information, becomes essential. Bioinformatics allows researchers to sift through the massive biological data and identify information of interest. Today, biologists are in search of bio-molecular sequence data, for its comparison with other genomes, and because its structure determines function and leads to the understanding of biochemical pathways, disease prevention and cure, and the mechanisms of life itself.

Bioinformatics has become an industry and has gained popularity among numerous markets including pharmaceutical, (industrial, agricultural and environmental) biotechnology and homeland security. A number of recent market research reports estimate the size of the bioinformatics market as $176 billion and the market is projected to grow to $243 billion within the next 5 years [6]. In August 2000, IBM announced an initial $100 million investment to spur business development in the life sciences, assuring its prominence as one of the emerging computing markets.

Algorithms and applications in this new computational field of biology are now one of the largest consumers of computational power. Many problems use polynomial time algorithms (e.g., all-to-all comparisons) but have long running times due to the large data volume to process; for example, the assembly of an entire genome or the all-to-all comparison of gene sequence data. Other problems are compute-intensive due to their inherent algorithmic complexity, such as protein folding and reconstructing evolutionary histories from molecular data, that are known to be NP-hard (or harder) and often require approximations that are also complex [13].

Clearly, computer systems that can cost-effectively deliver high-performance on computational biology applications play a vital role in the future growth of the bioinformatics market. Since current high-end architectures have not been primarily designed with these applications in mind, we are interested in finding architectural trade-offs that can benefit computational biology without degrading performance of other workloads. In order to apply a quantitative approach in computer architecture design, optimization and performance evaluation, researchers need to identify representative workloads from this emerging application domain first.

This paper presents *BioPerf*, a benchmark suite representing a wide variety of bioinformatics applications. The codes span the heterogeneity of algorithms and biological problems. Only freely available open-source codes are included in this suite to ease its portability to new architectures and systems and its dissemination. Currently, the *BioPerf* suite contains 10 packages and covers the major fields of study in computational biology such as sequence comparison, phylogenetic reconstruction, protein structure prediction, and sequence homology & gene finding. Sequence comparison finds similarities between two or more DNA or protein sequences. Phylogeny explores the ancestral relationships among a set of genes or organisms. Protein structure analysis (a) finds the similarities between three-dimensional protein structures and (b) predicts the shape of a protein (e.g., primary, secondary, and tertiary structure) given its amino acid sequence. Gene-finding identifies DNA segments that encode proteins.

Table 1. BioPerf benchmark suite.

#	Package	Executable Codes
1	**Blast**	*blastp*, *blastn*
2	**FASTA**	*fasta34*, *ssearch*
3	**Clustalw**	*clustalw*, *clustalw_smp*
4	**Hmmer**	*hmmsearch*, *hmmpfam*
5	**T-Coffee**	*tcoffee*
6	**Glimmer**	*glimmer2*
7	**Phylip**	*dnapenny*, *promlk*
8	**Grappa**	*grappa*
9	**CE**	*ce*
10	**Predator**	*predator*

For each of these codes listed in Table 1, we have assembled benchmark source code, varying sizes of input datasets, and information for compiling and using the benchmarks. As algorithms for solving problems from computational biology often require parallel processing techniques, we provide parallel versions of 4 benchmarks. To facilitate computer architecture researchers to run the *BioPerf* suite on several popular execution driven simulators, we also provide little-endian Alpha ISA binaries and generated simulation points [7] and PowerPC binaries [11] from the *BioPerf* web page, **www.bioperf.org**.

The rest of this paper is organized as follows. Section 2 discusses previous work on benchmark collection. Section 3 provides an introductory background on biology and a brief review of bioinformatics study areas. Section 4 describes the *BioPerf* suite, including benchmark functionality, input datasets and execution. Section 5 presents the pre-compiled Alpha binaries, and the generated simulation points. Section 6 discusses our performance study on IBM Power using IBM Mambo simulations cross-compared with Apple G5 executions. Section 7 concludes the paper and outlines our future work.

2. Previous Work

One of the most successful attempts to create standardized benchmark suites is SPEC (Standard Performance Evaluation Corporation), which started initially as an effort to deliver better benchmarks for workstations. Over the years, SPEC evolved to cover different application classes, such as the SPECSFS for the NFS performance and the SPECWeb for performance of Web servers.

Other examples of domain-specific benchmarks include transaction-processing benchmarks TPC, benchmarks for embedded processors such as the EEMBC benchmarks and many others. One of the important benchmark suites in the scientific research community is the SPLASH (Stanford Parallel Applications for Shared Memory) suite, later updated to SPLASH-2 [8]. SPLASH-2 includes mostly codes from linear algebra and computational physics, and is designed to measure the performance of these applications on centralized and distributed memory-machines.

Few comprehensive suites of computationally-intensive life science applications are available to the computer architecture community. The closest effort to ours is the development of BioBench [9]. Compared with BioBench, our independent work covers many more bioinformatics tools in terms of quantity and diversity. Also, our work includes parallel codes where available.

3. Background

To help readers understand the *BioPerf* benchmarks better, we first provide an introductory background on biology and illustrate the major areas of bioinformatics.

3.1 Introduction: DNA, Genes and Proteins

One of the fundamental principles of biology is that within each cell, DNA that comprises the genes encodes RNA which in turn produces the proteins that regulate all of the biological processes within an organism.

DNA is a double chain of simpler molecules called *nucleotides*, tied together in a double helix helical structure. The nucleotides are distinguished by a nitrogen base that can be of four kinds: *adenine* (A), *cytosine* (C), *guanine* (G) and *thymine* (T). Adenine (A) always bonds to thymine (T) whereas cytosine (C) always bonds to guanine (G), forming base pairs. DNA can be specified uniquely by listing its sequence of nucleotides, or base pairs. Proteins are molecules that accomplish most of the functions of a living cell, determining its shape and structure. A protein is a linear sequence of molecules called *amino acids*. Twenty different amino acids are commonly found in proteins. Similar to DNA, proteins are conveniently represented as a string of letters expressing their sequence of amino acids. A gene is a contiguous stretch of genetic code along the DNA that encodes a protein. Not all parts of a DNA molecule encode genes; some segments, called *introns*, have no influence on protein synthesis.

As a protein is produced, it folds into a three-dimensional shape. For example, figure 1 shows the 3-D structure of human *foetal deoxyhaemoglobin*. The positions of the central atoms, called *carbon-alpha* ($C_{\propto}$), of the amino acids of a protein define its primary structure. If a contiguous subsequence of $C_{\propto}$ atoms follows some predefined pattern, they are classified as a secondary structure, such as alpha-helix or beta-sheet. The relative positioning of the secondary structures define the tertiary structure. The overall shape of all chains of a protein then defines the quaternary structure.

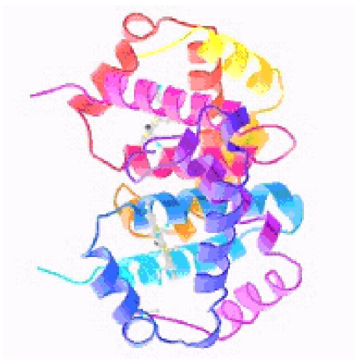

Figure 1. Three dimensional structure of human *foetal deoxyhaemoglobin* (PDB id = 1FDH, produced from [12]).

As a protein is produced, it folds into a three-dimensional shape. For example, figure 1 shows the 3-D structure of human *foetal deoxyhaemoglobin*. The positions of the central atoms, called *carbon-alpha* ($C_{\propto}$), of the amino acids of a protein define its primary structure. If a contiguous subsequence of $C_{\propto}$ atoms follows some predefined pattern, they are classified as a secondary structure, such as alpha-helix or beta-sheet. The relative positioning of the secondary structures define the tertiary structure. The overall shape of all chains of a protein then defines the quaternary structure.

3.2 Bioinformatics Problems

In this section, we illustrate the major problems in bioinformatics, including sequence analysis, phylogeny, sequence homology & gene finding, and protein structure analysis/prediction.

3.2.1 Sequence Analysis

Sequence analysis is perhaps the most commonly performed task in bioinformatics. Sequence analysis can be defined as the problem of finding which parts of the sequences (nucleotide or amino acid sequences) are similar and which parts are different. By comparing sequences, researchers can gain crucial understanding of their significance and functionality: high sequence similarity usually implies significant functional or structural similarity while sequence differences hold the key information regarding diversity and evolution.

The most commonly used sequence analysis technique is pairwise sequence comparison. A sequence can be transformed to another sequence with the help of three edit operations. Each edit operation can insert a new letter, delete an existing letter, or replace an existing letter with a new one. The alignment of two sequences is defined by the edit operations that transform one into the other. This is usually represented by writing one on top of the other. Insertions and deletions (i.e., gaps) are represented by the dash symbol ("-"). The following example illustrates an alignment between the sequences S_1= "**GAATTCAGTA**" and S_2= "**GGATCGTTA**". The objective is to match identical subsequences as best as possible (or equivalently use as few edit operations as possible). In the example, the aligned sequences match in seven positions.

```
Sequence S1 GAATTCAGT-A
             |R|D||D||I|
Sequence S2 GGA-TC-GTTA
```

Figure 2. Alignment of two sequences that match in seven positions. One replace, two delete, and one insert operations, shown by letters R, D, and I, are used.

Alignment of sequences is considered in two different but related classes: If the entire sequences are aligned, then it is called a *global alignment*. If subsequences of two sequences are aligned, then it is called a *local alignment*. Multiple sequence alignment compares more than two sequences: all sequences are aligned on top of each other. Each column is the alignment of one letter from each sequence. The following example illustrates a multiple alignment among the sequences S_3= "**AGGTCAGTCTAGGAC**", S_4= "**GGACTGAGGTC**", and S_5="**GAGGACTGGCTACGGAC**".

Sequence S_3 **`-AGGTCAGTCTA-GGAC`**
Sequence S_4 **`--GGACTGA----GGTC`**
Sequence S_5 **`GAGGACTGGCTACGGAC`**

Figure 3. Multiple alignment of three DNA sequences S_3, S_4, **and** S_5.

3.2.2 Molecular Phylogeny Analysis

Molecular phylogeny infers lines of ancestry of genes or organisms. Phylogeny analysis provides crucial understanding about the origins of life and the homology of various species on earth. Phylogenetic trees are composed of nodes and branches. Each leaf node corresponds to a gene or an organism. Internal nodes represent inferred ancestors. The evolutionary distance between two genes or organisms is computed as a function of the length of the branches between their nodes and their common ancestors.

3.2.3 Sequence Homology and Gene Finding

Portions of genomes could be seen as genomic entities spawned through some dynamic changes in content and order of the ancestral genome. Certain regions, through selection, are conserved over time. Such genomic portions that are related due to their derivation from the same element in a common ancestral genome are termed homolog. Sequence homology study aims to infer genome organization and structure, as well as the evolutionary mechanisms that shaped present day genomes.

The sizes of biological sequence databases are usually very large. Not all the sequences are coding, namely are a template for a protein. For example, in the human genome only 3%-5% of the sequences are coding. Due to the size of the database, manual searching of genes who do code for proteins is not practical. Gene-findings aim to provide computational methods to automatically identify genes that encode proteins.

3.2.4 Protein Structure Analysis

Two protein substructures are called similar if their C∝ atoms can be mapped to close-by points after translation and rotation of one of the proteins. This can also be considered as a one-to-one mapping of amino acids. Usually, structural similarity requires that the amino acid pairs that are considered similar have the same secondary structure type. Structural similarities among proteins provide insight regarding their functional relationship. Figure 4 presents the structural similarity of two proteins.

Three-dimensional structures of only a small subset of proteins are known as it requires expensive wet-lab experimentation. Computationally determining the structure of proteins is an important problem as it accelerates the experimentation step and reduces expert analysis. Usually, the relationship among chemical components of proteins (i.e. their amino acid sequences) is used in determining their unique three-dimensional native structures.

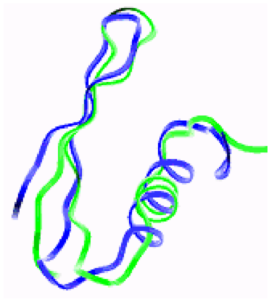

Figure 4. The structural similarity between two proteins. (produced from http://cl.sdsc.edu/**)**

Three-dimensional structures of only a small subset of proteins are known as it requires expensive wet-lab experimentation. Computationally determining the structure of proteins is an important problem as it accelerates the experimentation step and reduces expert analysis. Usually, the relationship among chemical components of proteins (i.e. their amino acid sequences) is used in determining their unique three-dimensional native structures.

3.3 Bioinformatics Databases

A bioinformatics database is an organized body of persistent data (e.g. nucleotide and amino acid sequences, three-dimensional structure). Thanks to the human genome project, there has been a growing interest both in the public and private sectors towards creating bioinformatics databases. At the end of 2002, there were more than 300 molecular biology databases available worldwide. This section provides a brief overview of several popular and publicly available bioinformatics databases.

An important class of bioinformatics databases is the sequence database. The largest sequence database is the NCBI/*GenBank* [5] which collects all known nucleotide and protein sequences. Other major data sources are *EMBL* (European Molecular Biology Lab) [14] and *DDBJ* (DNA Data Bank of Japan) [15]. Two major sources of protein sequences and structures are *PDB* (Protein Data Bank) [12], and *SWISS-PROT* [16]. *PDB* contains the protein structures determined by NMR and X-ray crystallography techniques. *SWISS-PROT* is a curated protein sequence database which provides a high level of annotation such as description of protein function, its domain structure, post-translational modification and other useful information.

4. The BioPerf Benchmark Suite

To allow computer architecture researchers to explore and evaluate their designs on these emerging applications, we developed *BioPerf*, a suite of representative applications assembled from the computational biology community, where the codes are carefully selected to span a breadth of algorithms and performance characteristics.

Bioinformatics is a field for which the problems themselves are not thoroughly categorized, and many of the computational problems are NP-hard. This has led to the development of heuristics to solve the problems, giving sub-optimal results within a reasonable degree of accuracy quickly. Thus, the field is still in its infancy with problems, algorithms, applications, and even system architecture requirements, changing frequently. The present suite of tools should therefore be treated as a starting point. As the field evolves, we expect the included codes and inputs to evolve to encompass important emerging trends. Our endeavor is to provide a representative set of codes and sample data that encompass the field of bioinformatics in terms of the problems it represents and the solutions which are devised for those problems. We use this set of bioinformatics applications to drive changes in computer architecture for high-performance computing systems specifically targeted towards the computational biology applications.

The packages used in *BioPerf* are handpicked from the following broad problems identified by the biological community of interest to computer designers: sequence alignment (pairwise and multiple), phylogeny reconstruction, protein structure prediction, and sequence homology & gene-finding. For each of these codes, we have assembled input datasets with varying sizes which can be used in conjunction with the applications included in this suite. Due to space limitations, this paper details a moderate-sized class of input that allows each benchmark code to run for tens of minutes. Other class sizes are available from the *BioPerf* web site for smaller and larger runs. Detailed, quantitative workload characterization of the *BioPerf* benchmarks on different platforms can be found in [10, 11, 29, 30, 35, 36]. This section describes the selected programs, which can be classified using the categories we introduced in Section 3.2. Table 2 provides a summary of *BioPerf* executables and inputs.

4.1 Sequence Analysis Benchmarks

Blast: The *Blast* (Basic Local Alignment Search Tool) programs [17] are a set of heuristic methods that are used to search sequence databases for local alignments to a query sequence. The *Blast* programs are written in C. *BlastP* and *BlastN* are the versions of *Blast* for searching protein and nucleotide sequences respectively. The query file is the file which includes the nucleotide or protein sequence for search. The database file is the database which will be searched. The blast implementation provided by NCBI is multithreaded and contains a parameter to set the number of threads.

ClustalW: *ClustalW* [19] is a multiple sequence alignment program for nucleotides or amino acids. It first finds a phylogenetic tree for the underlying sequences. It then progressively aligns them one by one based on their ancestral relationships. *ClustalW* is programmed in C and takes as input multiple DNA or protein sequences and output the results after alignment. *Clustalw_smp* is a symmetric multiprocessor implementation of *ClustalW*.

Hmmer: *Hmmer* [20] employs hidden Markov models (profile HMMs) for aligning multiple sequences. Profile HMMs are statistical models of multiple sequence alignments. They capture position-specific information about how conserved is each column of the alignment, and which residues are likely. *Hmmer* is programmed in the C language. It includes several applications such as *hmmbuild, hmmcalibrate* and *hmmsearch*. Among these applications, the *hmmsearch* is widely used to search a sequence database for matches to an HMM. The benchmark input is the example HMM built from the alignment file of 50 aligned globin sequences and a FASTA file of brine shrimp globin, which contains nine tandemly repeated globin domains. *Hmmpfam* [20] another program of the same family, compares one or more sequences to a database of profile hidden Markov models, such as the Pfam library, in order to identify known domains within a sequence, using either the Viterbi or the forward algorithm. *Hmmpfam* is a multithreaded program. The dataset we have provided for *hmmpfam* performs the search of a transcriptional regulatory protein of about 8800 residues against the PFAM database.

T-Coffee: *T-Coffee* [21] is a sequential multiple sequence alignment similar to *ClustalW*, but which has been proven to be more accurate than *ClustalW*, though with a higher time complexity. *T-Coffee* enhances the progressive alignment of *ClustalW* with an internal library creation, and uses both scores from aligning every sequence with other sequences and the library for the alignment. The parameters we provide for running *T-Coffee* set the dynamic programming mode to Myers and Miller, which has linear space and quadratic time complexity. The option *-in* specifies methods used for library making (*lalign_id_pair* is the local alignment using FASTA function and *clustalw_pair* is the global alignment using the Smith-Waterman algorithm). The option *tree mode = slow* implies that similarity matrix construction is performed by using dynamic programming mode. The input file is *1yge_1byt* (50 sequences of average length 850) extracted from the Prefab database.

Table 2 Summary of *BioPerf* executables and inputs

Package	Executable Codes	Execution Commands	Input Instances
Blast	blastp blastn	blastall –p blastp -i <query file> -d <database file> -o <output file> -a <number of threads> blastall –p blastn -i <query file> -d <database file> -o <output file> -a <number of threads>	**Input Sequences**: target.txt is the homo sapiens hereditary haemochromatosis protein **Database**: non-redundant protein sequence database NCBI **Input Sequences**: Input dataset of 20 sequences each of about 7000 residues. **Database**: Swiss-Prot
FASTA	fasta34 ssearch	fasta34 <query file> <database file> ssearch34 t -a <Show entire length in alignment> -b 20 <number of high scores to display> -q -O <Output Alignment File> < Input Sequence >	**Input Sequence**: qrhuld.aa is a query file that contains the human LDL receptor precursor protein. **Database**: The nr is the same database mentioned above. **Input Sequences**: Two genomes NC_003903 (Streptomyces_coelicolor) and NC_005824 (Leptospira chromosome II) each of about 360,000 residues as the input dataset
Clustalw	clustalw clustalw_smp	clustalw -batch -infile= <input file> -outfile= <output file> clustalw_smp –infile=<input file> -outfile=<output file>	**Input Sequences**: input.ext is a query file that includes 317 Ureaplasma's gene sequences from the NCBI Bacteria genomes database **Input Sequences**: 6000.seq (318 sequences with average length of about 1450 residues) included with the executable is used as input.
Hmmer	hmmsearch hmmpfam	hmmsearch <input file> <database file> hmmpfam < HMM database > < Input sequence >	**Database**: globin.hmm is the example HMM of 50 aligned globin sequences **Input**: Artemia.fa is a FASTA file of brine shrimp globin, **Database**: Pfam **Input:** transcriptional regulatory protein of about 8800 residues against the PFAM database
T-Coffee	tcoffee	tcoffee < Input sequences file> -dp_mode = myers_miller_pair_wise -in=lalign_id_pair, clustalw_pair –tree_mode =slow	**Input**: The input file is 1yge_1byt (50 sequences of average length 850) extracted from the Prefab database.
Glimmer	glimmer2 run_glimmer2	glimmer2 <input sequence> <model file> run-glimmer2 <input sequence>	**Input Sequence**: NC_000907.fna is a kind of bacterium whose name is Haemophilus_influenzae **Model File**: glimmer.icm is the collection file of Markov models **Input Sequence**: Bacteria Bradyrhizobium japonicum genome NC_004463.fna consisting of about 9200 kilobase pairs
Phylip	dnapenny promlk	Execution Script provided with the benchmark	**Input**: Aligned 92 cyclophilins and cyclophilin-related proteins from eukaryotes of average length 220
Grappa	grappa	grappa -f < Input file > -o < Output file > -m, where m specifies tighten circular lower bound	**Input**: Input file is 12 sequences of the bluebell flower species Campanulaceae
CE	CE	CE - <protein sequence 1 in PDB> - <protein sequence 2 in PDB> - <path of directory for storing temporary results of execution>	**Input**: 1hba.pdb and 4hhb.pdb are different types of hemoglobin which is used to transport oxygen
Predator	predator	predator -a -l <sequence> -f<output>	**Input**: eukaryota_100.seq includes 100 Eukaryote protein sequences from NCBI genomes database. The additional dataset to run Predator is 19 sequences extracted from SWISS-PROT each of almost 7000 residues

FASTA: Similar to *Blast*, *FASTA* [18] is a collection of local similarity search programs for sequence databases. While *FASTA* and *Blast* both do pairwise local alignment, their underlying algorithms are different. The query and database files for *FASTA* have the same meaning as those of *Blast*. With the provided dataset, the *FASTA* benchmark performs a query that contains the *human LDL receptor precursor* protein. Another program that we have included from the FASTA package is the *ssearch* which does an exact Smith-Waterman alignment on a pair of sequences.

4.2 Sequence Homology and Gene Finding

Glimmer: *Glimmer* (Gene Locator and Interpolated Markov Modeler) [22] finds genes in microbial DNA. Its uses interpolated Markov models (IMMs) to identify coding and noncoding regions in the DNA. The program consists of essentially two steps: the first step trains the IMM from an input set of sequences, the second step uses this trained IMM for finding putative genes in the input genome. *Glimmer* can be used for gene annotation by inputting its predictions into BLAST and FASTA. The input we provide for *Glimmer* is a kind of bacterium whose name is *Haemophilus_influenzae* and *glimmer.icm* is the collection file of Markov models. Run-glimmer2 is a script included in the program, which runs programs *long-orfs* and *extract* (extracts all non-overlapping open reading frames), *build-icm* (build an interpolated context model) and finally runs *glimmer2* for a particular sequence. The input sequence used for this script is *Bacteria Bradyrhizobium japonicum* genome consisting of about 9200 kilobase pairs.

4.3 Molecular Phylogeny Analysis Benchmarks

Phylip: *Phylip* (PHYLogeny Inference Package) [23] is a package of programs for inferring phylogenies (evolutionary trees). Methods that are available in the package include parsimony, distance matrix, maximum likelihood, bootstrapping, and consensus trees. Data types that can be handled include molecular sequences, gene frequencies, restriction sites and fragments, distance matrices, and discrete characters. *Dnapenny* and *promlk* are the typical applications in *Phylip*. *Dnapenny* is a program that finds all of the most parsimonious trees of the input data. *Promlk* implements the maximum likelihood method for protein amino acid sequences. They both can run in command line method or interactive method. To provide deterministic execution, we provide execution script to invoke the two benchmarks. The additional dataset available to run *promlk* is the aligned 92 *cyclophilins* and *cyclophilin*-related proteins from eukaryotes of average length 220.

GRAPPA: *GRAPPA* (Genome Rearrangements Analysis under Parsimony and other Phylogenetic Algorithms) is a program for phylogeny reconstruction [24]. To date, almost every model of speciation and genomic evolution used in phylogenetic reconstruction has given rise to NP-hard optimization problems. GRAPPA is a reimplementation of the breakpoint analysis [25] developed by Blanchette and Sankoff, and also provides the first linear-time implementation of inversion distances improving upon Hannenhalli and Pevzner's polynomial time approach [26]. Currently, GRAPPA also handles inversion phylogeny and unequal gene content. The input file is 12 sequences of the bluebell flower species *Campanulaceae*.

4.4 Protein Structure Analysis Benchmarks

CE: *CE* (Combinatorial Extension) [27] finds structural similarities between the primary structures of pairs of proteins. CE first aligns small fragments from two proteins. Later, these fragments are combined and extended to find larger similar substructures. The inputs we provide for *CE* are different types of hemoglobin used to transport oxygen.

Predator: *Predator* [28] is a tool for finding protein structures, and is based on the calculated propensities of every 400 amino-acid pairs to interact inside an α -helix or one upon three types of β -bridges. It then incorporates non-local interaction statistics. *Predator* uses propensities for α -helix, β -strand and coil derived form a nearest neighbor approach. Our input for *Predator* includes 100 *Eukaryote* protein sequences from NCBI genomes database and results of the secondary structure prediction. The additional dataset to run *Predator* is 19 sequences extracted from *SWISS-PROT* each of almost 7000 residues.

5. Pre-Compiled Binaries and Simulation Points

5.1 *BioPerf* Alpha Binaries for Simulation based Studies

To allow computer architecture researchers to simulate the *BioPerf* benchmarks using execution-driven and cycle-accurate architecture research frameworks (such as SimpleScalar [32], SimAlpha [33], and M5 [34]), we made an extra effort to produce the Alpha binaries of the majority of *BioPerf* benchmarks. We have tested all pre-compiled Alpha binaries (with static link option) using the Simplescalar sim-outorder simulator. The pre-compiled binaries are available in the *BioPerf* package.

5.2 Simulation Points of *BioPerf* Workloads

Our earlier study [10, 11, 30] shows that bioinformatics applications can execution billions of instructions before completion. Therefore, it is infeasible to simulate entire benchmark execution using detailed cycle-accurate simulators. Recently, the computer architecture research community has widely adopted SimPoint [7] methodology as an efficient way to simulate the representative workload execution phases. We used the SimPoint framework developed by Calder et al. to generate the simulation points of the *BioPerf* benchmarks listed in the Table 2. Since the total number of instructions of different benchmarks varies significantly, we used the criteria suggested in [34] to determine the size of interval for each individual benchmark. Table 2 lists the interval size as well as the simulation points for each benchmark.

Table 3. Pre-compiled Alpha binaries (all binaries have been successfully tested on Simplescalar Sim-outorder simulator) and their simulation points

Benchmark	Input Dataset	Interval (M)	Simulation Points
fasta34	human LDL receptor precursor protein, NCBI *nr* database	400	412,698,242,326,810,961,503,487,354,1051,932,8,832,792,459,988,54,107,482,808,136,554,996,588
clustalw	317 *Ureaplasma*'s gene sequences from the NCBI *Bacteria genomes* database	850	24,918,906,701,702,674,793,395,252,845,719,883,857,585,858,651,381,817
hmmsearch	a profile HMM built from the alignment of 50 *globin* sequences, uniprot_sprot.dat from *SWISS-PROT*	680	340,674,330,695,711,75,619,599,54,677,551,672,794,40,404,682,618,370,457,951
glimmer2	18 bacteria complete genomes from the NCBI genomes database	20	581,1,45,70,32,13,2,17,1567,8,404,26,84,1572,6,21,494,1109,40,1240
diffseq	nucleic acid database *EMBL*	35	705,680,444,597,442,255,662,10,3,977,288,443,1006,827,990,1004,343,927,707,256,98,1054,689,964,780,958,824,942
megamerger	nucleic acid database *EMBL*	35	703,559,254,596,679,443,377,825,259,1067,3,442,901,255,310,593,94,773,976,758,889,781,1058,818,5,560
shuffleseq	nucleic acid database *EMBL*	300	318,1,718,981,280,115,18,355,261,365,1019,457,194,1018,196,43,254,406,226,775,842,454,894,986,776
dnapenny	ribosomal RNAs from *bacteria* and *mitochondria*	140	95,1048,1043,225,672,1,61,413,185,627,51,1014,1056,2,110,911,200,1003,777,970,639,1053,40,438,576,597
promlk	protein amino acid sequences of 17 species ranging from a deep branching bacterium to humans	320	21,911,255,1,320,548,876,713,176,319,813,332,818,932,445,807,909,773,580,854,224,719,700,969,472,715,1008,661,375,210
predator	100 *Eukaryote* protein sequences from NCBI genomes database	700	272,247,758,195,1143,536,535,166,585,81,233,296,482,640,429,406,88,343,203,403,479,955,37,971

6. Performance Analysis on PowerPC G5 and Mambo

We have performed an exhaustive analysis of *BioPerf* codes on the cycle-accurate IBM Mambo simulator [37] and the Apple G5 (IBM PowerPC 970) workstation [38] (see Figure 5). In addition to reporting the aggregate performance characteristics at the end of each run, we also analyze the "live-graph" performance data which show the variation of the performance exhibited during the execution of the application. We are using these analyses to propose architectural features in the design of next generation computer architectures to optimize their performance for computational biology workloads.

An advantage of the Mambo and MONster tools is the "live graph" capabilities that report accumulated data not only at the end of each run, but also at chosen regular sampling intervals. Using such data, we correlate the performance with phases of the application, and suggest optimizations targeted at separate regions of the application.

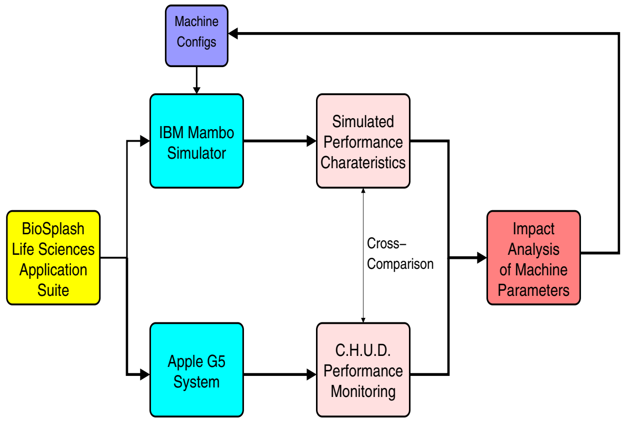
Machine Configs
IBM Mambo Simulator
Simulated Performance Charateristics
BioSplash Life Sciences Application Suite
Cross–Comparison
Impact Analysis of Machine Parameters
Apple G5 System
C.H.U.D. Performance Monitoring

Figure 5. Dual-platform analysis reveals how cache and machine organization parameters affect the performance of life sciences applications.

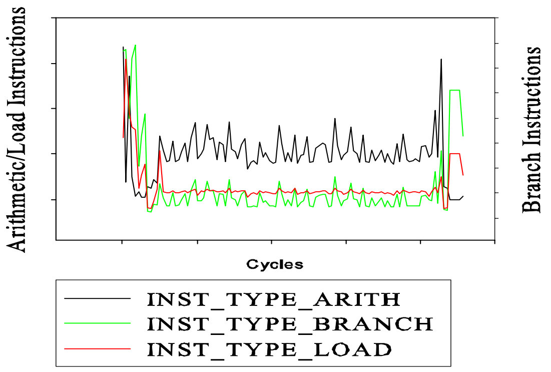

Figure 6. Instruction profile of *Blast.*

As an example, we include two live-graphs in this paper, one from the Mambo simulator runs and the other from Apple G5 runs. In the first plot (Figure 6), we have the instruction profiles for *Blast* from Mambo, and the second plot (Figure 7) shows the variation in instructions per cycle (IPC) with the L1 data miss rate. For cumulative performance analysis, we use the data generated by the MONster tool, since the Apple G5 codes may be run for significantly larger input data sets. We summarize the hardware performance counters for *BioPerf* runs we used on the Apple G5. These include:

- Instruction-level analysis: Instructions dispatched, instructions completed (including and excluding I/O and load/store), branch mispredictions due to condition register value and target address predict were recorded as part of this analysis.
- Memory-level analysis: L1 and L2 cache loads and stores, L1 and L2 cache load and store misses, TLB and SLB misses, read/write request bytes, number of memory transactions and Load Miss Queue (LMQ) full events were included in this analysis.

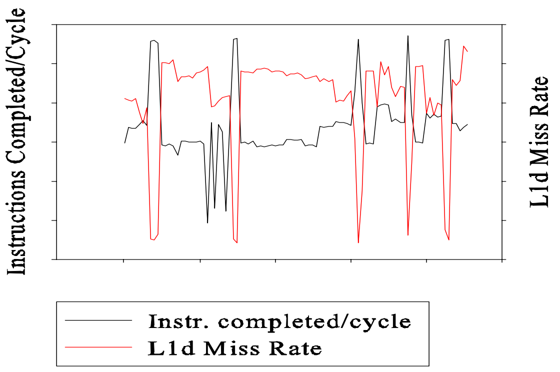

Figure 7. Variation in IPC and L1 data cache miss rate

In our research studies, we also compare these codes on dual-core systems to quantify to what extent computational biology applications can utilize and benefit from commodity computer architecture enhancements such as multi-core processing chips. We are not able to include all the live-graphs, cumulative data and the analysis in this document due to space constraints; fully detailed analysis and all the live-graphs are included in [11].

7. Conclusions

Bioinformatics applications represent increasingly important computer workloads. In order to apply a quantitative approach in computer architecture design and performance evaluation, there is a clear need to develop a benchmark suite of representative bioinformatics applications. This paper presents a group of programs representative of bioinformatics software. These programs include popular tools used for sequence alignments, molecular phylogeny analysis, protein structure prediction, and gene finding. The *BioPerf* benchmark suite is freely available from the web site **www.bioperf.org**. As the field of bioinformatics evolves, we will extend *BioPerf* to encompass important emerging trends. In the future, we will also explore integrated software/hardware techniques to optimize the performance of bioinformatics applications.

8. Acknowledgements

Bader's work was supported in part by DARPA Contract NBCH30390004; and NSF Grants CAREER ACI-00-93039, NSF DBI-0420513, ITR ACI-00-81404, DEB-99-10123, ITR EIA-01-21377, Biocomplexity DEB-01-20709, and ITR EF/BIO 03-31654. We would like to thank Ram Rajamony of IBM Austin Research Laboratories for his help with the Mambo simulator and the CHUD performance tools and performance metrics. We also would like to thank Virat Agarwal and Gaurav Goel for performing several of the simulation runs included in the paper, Tamer Kahveci and Abhishek Narain Singh for helping with the selection of the *BioPerf* codes.

References

[1] J. L. Weber and E. W. Myers, Human Whole Genome Shotgun Sequencing, Genome Research, 7(5): 401-409, 1997.

[2] E. Anson and E. W. Myers, Algorithms for Whole Genome Shotgun Sequencing, In Proceedings of 3rd Annual International Conference on Computational Molecular Biology, 1999.

[3] J. C. Venter et al., The Sequence of the Human Genome, Science, 291(5507):1304-1351, 2001.

[4] M. Schena, D. Shalon, R.W. Davis, and P.O. Brown. Quantitative Monitoring of Gene Expression Patterns with a Complementary DNA Microarray, Science, 270(5235):467-470, 1995.

[5] http://www.ncbi.nlm.nih.gov/Genbank/genbankstats.html

[6] Bioinformation Market Study for Washington Technology Center, Alta Biomedical Group LLC, www.altabiomedical.com, June 2003.

[7] T. Sherwood, E. Perelman, G. Hamerly and B. Calder, Automatically Characterizing Large Scale Program Behavior, In Proceedings of the International Conference on Architectural Support for Programming Languages and Operating Systems, 2002.

[8] S. C. Woo, M. Ohara, E. Torrie, J. P. Singh, and A. Gupta, The SPLASH-2 Programs: Characterization and Methodological Considerations, In Proceedings of the International Symposium on Computer Architecture, 1995.

[9] K. Albayraktaroglu, A. Jaleel, X. Wu, M. Franklin, B. Jacob, C.-W. Tseng, and D. Yeung, BioBench: A Benchmark Suite of Bioinformatics Applications, In Proceedings of the International Symposium on Performance Analysis of Software and Systems, 2005.

[10] Y. Li, T. Li, T. Kahveci and J. Fortes, Workload Characterization of Bioinformatics Applications on Pentium 4 Architecture, In Proceedings of the International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, 2005.

[11] D. A. Bader, V. Sachdeva, V. Agarwal, G. Goel, A. N. Singh, BioSPLASH: A Sample Workload for Bioinformatics and Computational Biology for Optimizing Next-Generation Performance Computer Systems, Technical Report, University of New Mexico, April 2005.

[12] The RCSB Protein Data Bank, http://www.rcsb.org/pdb/

[13] D. A. Bader, Computational Biology and High-Performance Computing, Special Issue on Bioinformatics, Communications of the ACM, 47(11):34-41, 2004.

[14] European Molecular Biology Laboratory, http://www.embl-heidelberg.de

[15] DNA Data Bank of Japan, http://www.ddbj.nig.ac.jp/

[16] The UniProt/Swiss-Prot Database, http://www.ebi.ac.uk/swissprot/

[17] S. 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, pages 403-410, 1990.

[18] W. R. Pearson and D. J. Lipman, Improved Tools for Biological Sequence Comparison, Proc. Natl. Acad. Sci., 85 (1988), 3244-3248.

[19] 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, no. 22, pages 4673-4680, 1994.

[20] S. R. Eddy, Profile Hidden Markov Models, Bioinformatics Review, vol. 14, no. 9, page 755-763, 1998.

[21] C. Notredame, D. Higgins, and J. Heringa. T-Coffee: A Novel Method for Multiple Sequence Alignments, J. Molecular Biology, 302:205− 217, 2000.

[22] S. Salzberg, A. Delcher, S. Kasif, and O. White, Microbial Gene Identification using Interpolated Markov Models, Nucleic Acids Research, vol. 26, no. 2, page 544-548, 1998.

[23] J. Felsenstein, PHYLIP - Phylogeny Inference Package (version 3.2), Cladistics, 5: 164-166, 1989.

[24] B. M. E. Moret, D. A. Bader, T. Warnow, S. K. Wyman, and M. Yan, GRAPPA: A High Performance Computational Tool for Phylogeny Reconstruction from Gene-Order Data, In Proc. Botany, Albuquerque, NM, August 2001.

[25] M. Blanchette, G. Bourque, and D. Sankoff, Breakpoint Phylogenies, In S. Miyano and T. Takagi, editors, Genome Informatics, pages 25-34. University Academy Press, Tokyo, Japan, 1997.

[26] D. A. Bader, B. M. E. Moret, and M. Yan, A Linear-Time Algorithm for Computing Inversion Distance between Signed Permutations with an Experimental Study, Journal of Computational Biology, 8(5):483-491, 2001.

[27] I. N. Shindyalov, and P. E. Bourne, Protein Structure Alignment by Incremental Combinatorial Extension (CE) of the Optimal Path, Protein Engineering, vol. 11, no. 99, page 739-747, 1998.

[28] D. Frishman, and P. Argos, 75% Accuracy in Protein Secondary Structure Prediction, Proteins, vol. 27, page 329-335, 1997.

[29] Y. Li and T. Li, BioInfoMark: A Bioinformatics Benchmark Suite for Computer Architecture Research, Technical Report, IDEAL Research, ECE Dept., University of Florida, January 2005.

[30] T. Kahveci, V. Ramaswamy, H. Tao and T. Li, Approximate Global Alignment of Sequences, IEEE Symposium on Bioinformatics and Bioengineering, 2005.

[31] Simplescalar, http://www.simplescalar.com/

[32] R. Desikan, D. Burger, and S. W. Keckler. Measuring Experimental Error in Microprocessor Simulation, In Proceedings of the International Symposium on Computer Architecture, 2001.

[33] The M5 Simulator System, http://m5.eecs.umich.edu/

[34] G. Hamerly, E. Perelman and B. Calder, How to Use SimPoint to Pick Simulation Points, ACM SIGMETRICS Performance Evaluation Review, 2004.

[35] D. A. Bader, V. Sachdeva, A. Trehan, V. Agarwal, G. Gupta, and A.N. Singh, BioSPLASH: A Sample Workload from Bioinformatics and Computational Biology for Optimizing Next-Generation High-Performance Computer Systems, (Poster Session), 13th Annual International Conference on Intelligent Systems for Molecular Biology (ISMB 2005).

[36] D. A. Bader, V. Sachdeva, BioSPLASH: Incorporating Life Sciences Applications in the Architectural Optimizations of Next-Generation Petaflop-System, (Poster Session), The 4th IEEE Computational Systems 4th IEEE Computational Systems Bioinformatics Conference (CSB 2005).

[37] P. Bohrer, M. Elnozahy, A. Gheith, C. Lefurgy, T. Nakra, J. Peterson, R. Rajamony, R. Rockhold, H. Shafi, R. Simpson, E. Speight, K. Sudeep, E. Van Hensbergen, and L. Zhang. Mambo - A Full System Simulator for the PowerPC Architecture, ACM SIGMETRICS Performance Evaluation Review, 31(4):8-12, 2004.

[38] Apple Computer, Inc. PowerPC G5: White Paper, images.apple.com/powermac/pdf/PowerPCG5 WP 06092004.pdf.