An Open Benchmark Suite for Evaluating Computer Architecture on Bioinformatics and Life Science Applications

David A. Bader
College of Computing
Georgia Institute of Technology
Atlanta, GA 30332 USA

Vipin Sachdeva
Electrical and Computer Engineering
University of New Mexico
Albuquerque, NM 87131

***Abstract*— In this paper, we propose BIOPERF, a definitive benchmark suite of representative applications from the biology and life sciences community, where the codes are carefully selected to span a breadth of algorithms and performance characteristics. The BIOPERF suite is available from `www.bioperf.org` and includes benchmark source code, input datasets of various sizes, and information for compiling and using the benchmarks. We include parallel codes where available.**

I. 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 [30], [3], [29] and the proliferation of high-speed micro array, gene, and protein chips [23] 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 with 56,037,734,462 bases from 52,016,762 sequences as of 15 December 2005 [18]. 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.

Algorithms and applications in this new computational field of biology are now one of the largest consumers of computational power for research and industry in pharmaceuticals, biotechnology, and homeland security. 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 [4]. 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.

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. In this paper, we propose BIOPERF, a definitive benchmark suite of representative applications that we have assembled from the biology and life sciences community, where the codes are carefully selected to span a breadth of algorithms and performance characteristics. 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. 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.

We are actively vetting the **BioPerf** suite with the computational biology and life sciences community, for instance, with a technical presentation at the 2005 Intelligent Systems of Molecular Biology (ISMB) [9] and the 2005 IEEE Computational Systems Bioinformatics (CSB) [7] conferences, and also a recent presentation at the IEEE International Symposium on Workload Characterization (IISWC 2005) [5].

II. METHODOLOGY AND SELECTION OF CODES

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 and gene finding. Only freely-available open-source codes are included in this suite to ease its portability to new architectures and systems and its dissemination.

The selection of codes follows these guiding principles:

- **Coverage:** The packages must span the heterogeneity of algorithms and biological and life science problems important today as well as (in our view) increasing in importance over the next 5-10 years.
- **Popularity:** Codes with larger numbers of users are preferred because these packages represent a greater percentage of the aggregate workloads used in this domain.
- **Open Source:** Open source code allows the scientific study of the application performance, the ability to place hooks into the code, and eases porting to new architectures.
- **Licensing:** Only packages for which their licensing allows free redistribution as open source are included. This requirement eliminated several popular packages, but was kept as a strict requirement to encourage the broadest use of this suite.
- **Portability:** Preference was given to packages that used standard programming languages and could easily be ported to new systems (both in sequential and parallel languages).
- **Performance:** We gave slight preference to packages whose performance is well-characterized in other studies. In addition, we strived for computationally-demanding packages and included parallel versions where available.

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.

Number	Package	Executable Codes
1	BLAST	blastn, blastp
2	FASTA	fasta34, ssearch34
3	CLUSTALW	clustalw, clustalw_smp
4	HMMER	hmmsearch, hmmpfam
5	T-COFFEE	tcoffee
6	GLIMMER	glimmer2, glimmer-package
7	PHYLIP	dnapenny, promlk
8	GRAPPA	grappa
9	CE	ce
10	PREDATOR	predator

TABLE I

BIOPERF BENCHMARK SUITE

For each of these codes listed in Table I, 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 four 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 [24] and PowerPC binaries [8] from the BIOPERF web page, `www.bioperf.org`. The rest of this paper is organized as follows. Section III discusses previous work on benchmark collection. Section IV provides an introductory background on biology and a brief review of bioinformatics study areas. Section V describes the BIOPERF suite, including benchmark functionality, input datasets and execution. Section VI discusses the packaging of the suite, and Section VII describes the code arguments used to run the benchmark instance. Finally, section VIII concludes the paper.

III. 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 [31]. 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 [1]. 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.

IV. BACKGROUND

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

A. 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, Fig. 1 shows the 3-D structure of human foetal deoxyhaemoglobin. The positions of the central atoms, called carbon-alpha (C_α), of the amino acids of a protein define its primary structure. If a contiguous subsequence of (C_α) 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.

Fig. 1. Three dimensional structure of human foetal deoxyhaemoglobin (PDB id = 1FDH, produced from [21])

B. Bioinformatics Problems

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

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
```

Fig. 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 S3="AGGTCAGTCTAGGAC", S4="GGACTGAGGTC", and S5="GAGGACTGGCTACGGAC".

```
Sequence S3   -AGGTCAGTCTA-GGAC
Sequence S4   --GGACTGA----GGTC
Sequence S5   GAGGACTGGCTACGGAC
```

Fig. 3. Multiple alignment of DNA sequences S_3, S_4 and S_5

2) 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) 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.

4) Protein Structure Analysis: Two protein substructures are called similar if their C_α 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. Fig. 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.

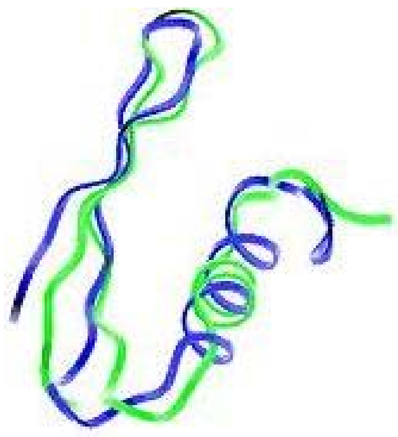
Fig. 4. The structural similarity between two proteins

C. 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 [18] which collects all known nucleotide and protein sequences. Other major data sources are EMBL (European Molecular Biology Lab) [13] and DDBJ (DNA Data Bank of Japan) [11]. Two major sources of protein sequences and structures are PDB (Protein Data Bank) [21], and SWISS-PROT [26]. 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.

V. 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 a 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 and 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 allow 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 [7], [8], [9], [16].

A. Sequence Analysis Benchmarks

- **Blast**: The Blast (Basic Local Alignment Search Tool) programs [2] 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.
- **FASTA**: Similar to Blast, FASTA [20] is a collection of local similarity search programs for sequence databases. FASTA is a two step algorithm. The first step is a search

for highly similar segments in the two sequences. In this search a word with a specific word size is used to find regions in a two-dimensional table table similar to that shown for the Smith-Waterman algorithm. These regions are a diagonal or a few closely spaced diagonals in the table which have a high number of identical word matches between the sequences. The second step is a Smith-Waterman alignment centered on the diagonals that correspond to the alignment of the highly similar sequence segments. 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.

- **ClustalW**: ClustalW [27] 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 combines several heuristics for improved accuracy: weights are given to the sequences implying their importance in the alignment. Duplicate sequences are underweight while divergent sequences are overweight. Amino acid substitution values are changed in every step of the alignment. Improved gap-handling algorithms especially during the early stage of the algorithm also improved the accuracy of the algorithm in general. ClustalX is a windows interface for the alignments created by ClustalW [28]. 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 [12] 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. Basically, you give HMMER a multiple sequence alignment as input; it builds a statistical model called a "hidden Markov model" which you can then use as a query into a sequence database to find (and/or align) additional homologues of the sequence family. 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 [12] 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 [19] 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. T-Coffee uses the familiar progressive approach to multiple sequence alignment. It uses time consuming pre-processing steps to guide through the alignment. As the authors point, intermediate alignments are based not only on the sequences to be aligned next, but also on the how all of the sequences align with each other. In this approach, they use a combination of local and global pairwise alignments to guide the alignments. The authors claim that the method is reliable irrespective of the phylogenetic background of the sequences to be aligned. 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.

B. *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 Glimmer system consists of two main programs. The first of these is the training program, build-imm. This program takes an input set of sequences and builds and outputs the IMM for them. These sequences can be complete genes or just partial orfs. For a new genome, this training data can consist of those genes with strong database hits as well as very long open reading frames that are statistically almost certain to be genes. The second program is glimmer, which uses this IMM to identify putative genes in an entire genome. Glimmer automatically resolves conflicts between most overlapping genes by choosing one of them. It also identifies genes that are suspected to truly overlap, and flags these for closer inspection by the user. These "suspect" gene candidates have been a very small percentage of the total for all the genomes analyzed thus far. Glimmer's predictions as input to the

BLAST and FASTA programs is thereby used for gene annotation. 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.

C. Molecular Phylogeny Analysis Benchmarks

- **Phylip**: Phylip (PHYLogeny Inference Package) [14] 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 [17]. 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 [10] 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 [6]. Currently, GRAPPA also handles inversion phylogeny and unequal gene content. The input file is 12 sequences of the bluebell flower species Campanulaceae.

D. Protein Structure Analysis Benchmarks

- **CE**: CE (Combinatorial Extension) [25] 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 input we provide for CE are different types of hemoglobin used to transport oxygen.
- **Predator**: Predator [15] 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.

VI. BioPerf package

The installation directory of BioPerf contain the following directories:

- `Binaries`: Directory containing pre-compiled x86, PowerPC and Alpha binaries, these binaries are contained in separate subdirectories, `Alpha-binaries`, `x86-binaries` (Linux) and `PowerPC-binaries` (Mac OS). x86 and the PowerPC binaries are included for all the executables, while Alpha binaries are not fully included for all the codes. The subdirectories for each of the platform further have directories for each of the packages.
- `Inputs`: Directory containing all the inputs for each of the executables. There are subdirectories for each of the packages, and the inputs for each executable are further categorized into class-A, class-B and class-C based on the sizes of the inputs. Some of the input directories have only one class of input, in which case there are no further subdirectories. The larger databases Swissprot (71MB), NR (1.46 GB) and Pfam (633 MB) are not included in the Inputs directory and have to be separately downloaded from the BioPerf website. In case an attempt is made to run a script for a executable which uses any of these databases, the scripts will look for the databases on the host machine in a directory represented by the environment variable `$DATABASES`.
- `Outputs`: All the scripts in BioPerf are set up to store their output in this directory. The `Outputs` directory itself has subdirectories by the name of each of the packages.
- `Scripts`: This directory contains all the scripts used in BioPerf except the `use-bioperf.sh` which is wrapper for all these scripts. It has further subdirectories that include the scripts for running each of the codes for small, medium and large datasets, scripts for compiling each of the codes, running the BioPerf suite and installing BioPerf on the target architecture in case the architecture's binaries are not part of the BioPerf package. The script has separate subdirectories for each of the tasks, but the naming itself is fairly intuitive.
- `Simpoints`: This directory contains the simulation points using the Simpoint methodology to simulate the representative workload execution phases.

- `Source-codes`: This directory contains the source-codes for each of the codes in BIOPERF. The directory structure is same with subdirectories for each package, and further with the executable for the packages having more than one executable.

The following scripts are included in BIOPERF in the `Scripts` directory:

- `use-bioperf.sh`: The wrapper for scripts `install-codes.sh`, `display-versions.sh`, `run-bioperf.sh` and `CleanOutputs.sh`. These scripts are explained below.
- `install-BioPerf.sh`: installs the BIOPERF into the directory specified by the user (explained in INSTALLATION section).
- `install-codes.sh`: compiles the codes for an architecture. In case the architecture is not x86 with Linux OS, PowerPC with Mac OS, or Alpha, you can use this script to compile the codes. This script picks up the makefiles of the source codes from the `Source-codes` directory, tries to do a make for each of the codes and installs the compiled codes into a subdirectory called `$HOSTNAME/Binaries`. It will also create `$HOSTNAME-scripts` subdirectory inside the directory `Scripts`, which can then be used to run the newly compiled executables. In case the codes cannot be compiled on the target architecture, the script will output an error message telling the user which code failed to compile.
- `display-versions.sh`: This script outputs the versions of all the installed codes in BIOPERF.
- `run-bioperf.sh`: This script is the basic run script for BIOPERF.

A. *How to Use* BIOPERF

On successful installation, run the script `use-bioperf.sh` located in the main directory of BIOPERF suite to run all the supported asks in BIOPERF. Note that `$BRUN` refers to the directory `$BIOPERF/Scripts/Run-scripts`. The following choices are available:

- **[R]** Run BIOPERF
- **[I]** Install BIOPERF on the user architecture
- **[C]** Clean Outputs in the user's `$BIOPERF/Outputs`
- **[D]** Display versions of all installed codes

These choices are explained as follows.

- **Run BIOPERF**: If the user selects the option [R], the BIOPERF suite is run through the script `$BRUN/run-bioperf.sh`. The script prompts the user for choosing either the platform, if the platform is x86, ppc or alpha, it then gives two modes of running BIOPERF: either the user can run all the codes one-by-one, or the user can choose codes which would be run. The user can then add packages to be run, with a prompt for every package. After the packages have been selected, the user is prompted for the size of the input datasets tp use. When BIOPERF is subsequently run, it outputs time of running for each executable and also the complete execution time for running the suite with the selected packages.
- **Install BIOPERF on the user architecture**: If the user selects the option [I], the architecture's binaries are compiled and installed through the script `$BRUN/install-bioperf.sh`. All the codes are attempted to be compiled, and the script fails in case any of the codes fails to compile. The script also tries to first delete the previous installation if detected before trying to proceed with the new installation. Assuming a successful compilation for each of the packages, the same subdirectory structure is maintained as in the x86 and the PowerPC subdirectories also. This allows the main run-bioperf.sh to use these executables just as they use x86 and the PowerPC executables.
- **Clean Outputs**: If the user selects the option [C], the outputs in `$BIOPERF/Outputs` are deleted through the script `$BRUN/CleanOutputs.sh`. All the scripts in BIOPERF store the outputs generated in running the executables in the `Outputs` directory, which has subdirectories for each of the packages also. This increases the size of the `Outputs` directory, and hence the script deletes all the outputs previously generated.
- **Display Versions**: If the user selects the option [D], the versions of all the software in the BIOPERF suite are displayed through the script `$BRUN/display-versions.sh`.

VII. BIOPERF PROGRAM OPTIONS FOR RUNNING

For each of the ten packages in BIOPERF, in Table II we provide the executable name, a brief summary of the code's use, and the arguments we use in our benchmark suite.

Code	**Summary**	**Running Options**
BLAST *(blastp)*	blastp searches for the homologues of an input amino acid sequence against a database of amino acid	*./blastall -p blastp -i Drosoph.txt -d Drosoph/drosoph.aa -o outdd* *-p -i* input query file *-d* Database *-o* Output file
(blastn)	blastn searches for homologues of input DNA sequences against a	
FASTA (*ssearch*)	ssearch does an exact Smith-Watermann of an input sequence with every sequence of an input library printing the results	*./ssearch34_t -a -b 20 -q -O <Output Alignment File> < Input Sequence > < Input Library File >* *-a* Show entire length in alignment *-b* Number of high scores to display *-q* Quiet
(*fasta34*)	fasta does a heuristic alignment of sequence with database file	*fasta34 < sequence file > < database file >*
ClustalW (*clustalw*)	Clustalw makes a multiple sequence alignment of the unaligned sequences given	*./clustalw < Unaligned sequences file >*
clustalw smp	Shared memory Clustalw	*clustalw_smp <unaligned sequence file>*
T-Coffee	T-Coffee makes a multiple sequence alignment of unaligned sequences	*./tcoffee < Input sequences file > -dp_mode = myers_miller_pair_wise -in = lalign_id_pair, clustalw_pair -tree_mode = slow* *-dp_mode* = Dynamic programming mode is Myers and Miller, linear space and quadratic time complexity *-in* = methods used for library making, lalign_id_pair is the local alignment using FASTA function, clustalw_pair is the global alignment using the Smith-Watermann. *tree_mode* = slow, similarity matrix construction done using dynamic programming mode.
HMMER (*hmmbuild*)	hmmbuild makes a profile hidden markov model from aligned sequences	*./hmmbuild <Output HMM file > < Input aligned sequences file >*
(*hmmpfam*)	hmmpfam searches for a sequence in a database of profile HMM's database of profile HMM's	*./hmmpfam < HMM database > < Input sequence > < Input sequence >*
Glimmer (*glimmer-package*)	Finds genes in microbial DNA especially bacteria and archea	*./run-glimmer2 < genome file >*
Glimmer (*glimmer*)	Finds genes in DNA given model file	*glimmer2 < input sequence > < model file >*
Grappa	Tool for phylogeny reconstruction	*./grappa -f < Input file > -o < Output file > -m* *-m Tighten circular lower bound*
Phylip (*dnapenny*)	Finds trees by parsimony	*./dnapenny < scriptdnapenny > < output >*
(*Promlk*)	Protein Maximum Likelihood Program with molecular clock	*./promlk < scriptproml > < output >*
CE	Predicts structure by aligning 3D structures	*ce < PDB sequence 1 > < PDB sequence 2 >*
Predator	Predicts the 3D structure of a protein taking an amino acid sequence as input	*./predator < Input Sequence > -u -h -a -f< Ouput file >* *-u Do not copy assignment directly from the PDB database if query sequence is found in PDB* *-h Indicate progress by dots* *-a Make prediction for all sequences in input file*

TABLE II
BIOPERF PROGRAMS WITH RUNNING OPTIONS

VIII. 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 and life science 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.

IX. Acknowledgments

This 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, Gaurav Goel, Abhishek Narain Singh, Tao Li, and Yue Li, for their help with the selection of the codes.

References

[1] K. Albayraktaroglu, A. Jaleel, X. Wu, M. Franklin, B. Jacob, C.-W. Tseng, and D. Yeung. BioBench: A benchmark suite of bioinformatics applications. In *Proc. of the 5th Int'l Symp. on Performance Analysis of Software and Systems (ISPASS)*, Austin, TX, March 2005.

[2] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman. Basic local alignment search tool. *J. Molecular Biology*, 215:403–410, 1990.

[3] E. Anson and E.W. Myers. Algorithms for whole genome shotgun sequencing. In *Proc. 3rd Ann. Int'l Conf. on Computational Molecular Biology (RECOMB99)*, Lyon, France, April 1999. ACM.

[4] D.A. Bader. Computational biology and high-performance computing. *Communications of the ACM*, 47(11):34–41, 2004.

[5] D.A. Bader, Y. Li, T. Li, and V. Sachdeva. BioPerf: A benchmark suite to evaluate high-performance computer architecture on bioinformatics applications. In *Proc. IEEE Int'l Symp. on Workload Characterization (IISWC*, Austin, TX, October 2005.

[6] 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.

[7] D.A. Bader and V. Sachdeva. Incorporating life sciences applications in the architectural optimizations of next-generation petaflop-system. In *Proc. 4th IEEE Computational Systems Bioinformatics Conference (CSB 2005)*, Stanford University, CA, August 2005. Poster Session.

[8] D.A. Bader, V. Sachdeva, V. Agarwal, G. Goel, and A.N. Singh. BioSPLASH: A sample workload for bioinformatics and computational biology for optimizing next-generation high-performance computer systems. Technical report, University of New Mexico, Albuquerque, NM, April 2005.

[9] 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. In *Proc. 13th Int'l Conf. on Intel. Sys. for Mol. Bio. (ISMB 2005)*, Detroit, MI, June 2005. Poster Session.

[10] 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.

[11] DNA Data Bank of Japan. DDBJ website. `http://www.ddbj.nig.ac.jp/`.

[12] S. R. Eddy. Profile hidden Markov models. *Bioinformatics*, 25:755–763, 1998.

[13] European Molecular Biology Laboratory. EMBL website. `http://www.embl-heidelberg.de/`.

[14] J. Felsenstein. PHYLIP – phylogeny inference package (version 3.2). *Cladistics*, 5:164–166, 1989.

[15] D. Frishman and P. Argos. 75% accuracy in protein secondary structure prediction. *Proteins*, 27:329–335, 1997.

[16] Y. Li, T. Li, T. Kahveci, and J. Fortes. Workload characterization of bioinformatics applications on Pentium 4 architecture. In *Proc. of the 13th IEEE Int'l Symp. on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*, Atlanta, GA, September 2005.

[17] 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.

[18] National Center for Biotechnology Information (NCBI). GenBank statistics. `http://www.ncbi.nlm.nih.gov/Genbank/genbankstats.html`, December 2005.

[19] C. Notredame, D. Higgins, and J. Heringa. T-Coffee: A novel method for multiple sequence alignments. *J. Molecular Biology*, 302:205–217, 2000.

[20] W. R. Pearson and D. J. Lipman. Improved tools for biological sequence comparison. *Proceedings of the National Academy of Sciences USA*, 85:2444–2448, 1988.

[21] Protein Data Bank. PDB RCSB protein data bank. `http://www.rcsb.org/pdb/`.

[22] S. Salzberg, A. Delcher, S. Kasif, and O. White. Microbial gene identification using interpolated Markov models. *Nucleic Acids Res.*, 26(2):544–548, 1998.

[23] 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.

[24] T. Sherwood, E. Perelman, G. Hamerly, and B. Calder. Automatically characterizing large scale program behavior. In *Proc. of the 10th Int'l Conf. on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, pages 45–57, San Jose, CA, October 2002.

[25] I.N. Shindyalov and P.E. Bourne. Protein structure alignment by incremental combinatorial extension (CE) of the optimal path. *Protein Engineering*, 11(99):739–747, 1998.

[26] SwissProt. The UniProt/Swiss-Prot database. `http://www.ebi.ac.uk/swissprot/`.

[27] J. D. Thompson, D. G. Higgins, and T. J. Gibson. CLUSTALW: improving the senstivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucleic Acids Res.*, 22:4673–4680, 1994.

[28] J.D. Thompson, T.J. Gibson, F. Plewniak, F. Jeanmougin, and D.G. Higgins. The ClustalX windows interface: flexible strategies for multiple sequence alignment aided by quality analysis tools. *Nucleic Acids Res.*, 24:4876–4882, 1997.

[29] J.C. Venter and *et al.* The sequence of the human genome. *Science*, 291(5507):1304–1351, 2001.

[30] J.L. Weber and E.W. Myers. Human whole-genome shotgun sequencing. *Genome Research*, 7(5):401–409, 1997.

[31] S. C. Woo, M. Ohara, E. Torrie, J. P. Singh, and A. Gupta. The SPLASH-2 programs: Characterization and methodological considerations. In *Proc. 22nd Ann. Int'l Symp. Computer Architecture*, pages 24–36, June 1995.