PBB: A Parallel Bioinformatics Benchmark Suite for Shared Memory Multiprocessors

Wenguang Chen, Chuntao Hong
Dept. of Computer Science and Technology
Tsinghua University
Beijing, China
cwg@tsinghua.edu.cn

ABSTRACT

Bioinformatics applications are widely used in life science and medicine industry where the demand on high performance computing system is increasing rapidly. It is very important to understand the characteristics of bioinformatics applications for both end-users and computer system designers. With the advent of multicore processors, it is also critical to know how well can multicore processors speedup bioinformatics applications. Although some bioinformatics benchmarks have been proposed, they cover limited application domains and most of them are serial codes. As a result, the analysis based on these benchmarks needs further examination with broader range of bioinformatics applications and it is indispensable to have a parallel bioinformatics benchmark set for shared memory multi-processors.

This paper presents PBB, a Parallel Bioinformatics Benchmark suite for shared memory processors. The benchmark suite is a collection of seven bioinformatics applications, which covers seven most important bioinformatics domains, such as sequence alignment, phylogenetic analysis, gene regulatory network learning, single nucleotide polymorphisms study and protein structure prediction. All of the applications have been parallelized with OpenMP except the NCBI *BLAST*, which was parallelized with the Pthread library.

We characterize the PBB on a real system and compare it with SPEC CPU2000 INT and FP. The results confirm and disprove some previous conclusion on bioinformatics workloads. Especially we disprove the claim in previous literature that "floating point operations are negligible".

Performance results on several shared memory multiprocessors with PBB are also presented and analyzed. Six out of seven PBB applications show satisfactory speedup up to 16 threads. HyperThreading techniques could provide modest speed up on PBB. Overall, the results of characterization and analysis of PBB suggest that multi-core processors could be used to support parallel bioinformatics workloads

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

ATIP 3rd China HPC Workshop, November 11, 2007, Reno, NV
Copyright 2007 ACM ISBN 978-1-59593-903-6/11/07...\$5.00

effectively.

1. INTRODUCTION

Bioinformatics applications are widely used in life science and medicine industry where the demand on high performance computing system is increasing rapidly. Although there are some general purpose benchmarks to evaluate the system performance, such as SPEC CPU2000 FP and INT [5], they may not match the characteristics of bioinformatics workloads well. The importance to have a bioinformatics benchmark suite is of two folds: 1) It enables users of bioinformatics applications to choose computer systems more wisely; 2) It allows the computer system researcher to understand the characteristics of bioinformatics applications to guide their system design.

Another trend recently is that the multicore processor is becoming the mainstream solutions in the market. To fully leverage the compute-power provided by multicore processors, it is critical for applications to support thread-level parallelism. Some people may argue that multiprogramming is good enough to use the multicore processors which could support high *throughput*. However, we believe that thread-level parallelization can enable applications to get much better *responsetime* than their serial versions which are also very important in many scenarios. For example, in a shot-gun whole genome assembly work flow, the gene assembly application are executed iteratively for several rounds to get satisfactory results. These rounds can not be executed in parallel because the parameter tuning must be directed by the result of previous executions. It takes days or weeks for a serial shot-gun whole genome assembly code to run once, so the total time for a work flow could easily cost months of time. With a parallel version, the time to solution could be significantly reduced which is critical for both researchers and enterprises. Thus, it is unquestionable that parallel bioinformatics applications are necessary to fully use the multicore processors and there is a clear need to a parallel bioinformatics benchmark suite for shared memory multiprocessors.

Although there have been some efforts recently on bioinformatics benchmarks, such as the BioBench[6] and the BioPerf[7], they either only cover limited application fields or only provide sequential benchmarks.

This paper presents PBB, a Parallel Bioinformatics Benchmark suite for shared memory multiprocessors(In this paper,

we use *SMP* to indicate shared memory multiprocessors regardless of whether the multiple processors are on one chip or on multiple chips). The benchmark suite is a collection of seven bioinformatics applications, which cover seven of the most important domains in bioinformatics, namely

- pairwise sequence alignment,
- global alignment,
- multiple sequences alignment,
- protein 3D structure prediction,
- phylogenetic tree reconstruction,
- gene regulatory network learning,
- pattern study of single nucleotide polymorphisms.

We believe the diversity in PBB helps the community to understand the characteristics of bioinformatics applications more thoroughly. Among these seven applications in PBB, six are parallelized with OpenMP, the de facto standard of shared memory programming, and the remaining NCBI BLAST is parallelized with Pthread library. So all of the seven applications support thread-level parallelization, making PBB a good tool in evaluating the performance of shared memory machines.

We collected instruction profile of PBB on a real SMP systems and compare it with the SPEC CPU2000 INT and FP. The results confirms some analysis result in [6] such as high percentage of load/store instructions and low CPI, but also contradicts the conclusion that "floating point operations are negligible in bioinformatics applications". In our study, we find that bioinformatics applications have non-negligible floating point operations, though still significant less than SPEC CPU 2000 FP.

In summary, the analysis of PBB applications on several different systems reveals that:

1. bioinformatics applications have high percentage of load/store instruction,
2. floating point operations are not negligible,
3. CPI is low,
4. HyperThreading technology generally provides modest speedup for the applications,
5. all the applications except SNP scale well to 16 threads, and
6. memory bandwidth could be the bottleneck for better scalability for some applications.

2. RELATED WORK

2.1 Bioinformatics Benchmark Suites

BioBench[6] is a bioinformatics benchmark suite which includes seven applications. BioBench emphasizes on mature genomics applications, thus does not include parallel code, and does not cover some important domains such as protein structure analysis, microarray analysis and mass spectrometry analysis.

BioPerf[7] is another bioinformatics benchmark suite which includes ten bioinformatics applications. Compared with BioBench, it includes five different applications which cover a few more domains such as protein structure analysis and gene finding. *ClustalW*, one benchmark in BioPerf includes a parallel implementation *Clustalw_smp*. But the majority of BioPerf are still serial code and could not be used to evaluate multi-processor machines effectively.

BioParallel is the first parallel bioinformatics benchmark suite which includes five parallel bioinformatics applications. It is a previous work of authors of this paper[1].

Compared with BioParallel, PBB includes three more applications to cover three important domains that are missed in BioParallel: *Rosetta* for 3-D protein prediction, *Muscle* for multiple sequence alignment and *BLAST - P* for pair-wise sequence alignment, whose serial versions are very popular in the bioinformatics community. Besides, the gene regulatory network learning benchmark *GeneNet* is replaced with a more advanced application *ModuleNet*. With more applications in PBB, we believe it is more representative than BioParallel.

2.2 Bioinformatics workload characterization with Bioinformatics Benchmark Suites

BioBench is used to characterize bioinformatics workloads[6]. It demonstrates distinct characteristics compared with SPEC CPU 2000 INT and SPEC CPU 2000 FP:

1. high percentage of load/store instructions,
2. almost negligible floating point operation, and
3. high IPC.

In this paper, we adopt the same analysis methodology to characterize PBB. Our results confirm the above characteristic 1 and 3 of bioinformatics workloads, but disprove the characteristic 2. The different results come from the diversity of applications in PBB.

The BioParallel is used in research on the last level cache performance in [18]. It is an in-depth analysis of parallel benchmarks and we plan to perform similar analysis with PBB in the future.

3. THE PBB BENCHMARK SUITE

3.1 Benchmark Selection and Construction

A good benchmark suite for workload characterization and system evaluation should be representative of the domain and portable to many platforms. The source code should be

available for the purpose of both porting and further study. As a parallel benchmark suite, PBB also requires that the application has a parallel version.

The benchmark selection process of PBB follows several stages:

1. Several important bioinformatics application domains are identified within the bioinformatics community.
2. For each domain, one or two representative applications are selected. For mature application domains, such as pair-wise gene sequence alignment, we chose the most popular applications (i.e. NCBI BLAST). For less mature application domains, we chose the most advanced applications or algorithms at the time of selection. The preference on more advanced algorithms is because they have better chance of gaining high popularity. If the implementation of an algorithm was not available yet, we would implement the algorithm by ourselves.
3. Benchmark optimization and parallelization. For each selected application, its serial version was optimized first. Methods of optimization include compiling with Intel compiler's highest level optimization options and architecture oriented optimizations, manual data re-organization to improve cache locality, and tricks to avoid large penalty data access and code execution patterns.

Because there're not many parallel bioinformatics applications or algorithms available, we have to parallelize most of applications by ourselves. In this version of PBB, there are seven applications among which only one application (the NCBI BLAST) has a parallel version. We parallelized all the remaining six applications ourselves.

Bioinformatics is a very diverse field and we don't claim that our benchmark suite covers all its important domains. For example, we don't include any mass spectrometry analysis code in PBB. However, we believe that the initial version of PBB covers some important domains and could help the understanding on the characteristics of the bioinformatics workloads, especially the parallel workloads.

3.2 Benchmark Applications in PBB

A brief description of the PBB applications is shown in Table 1, which includes the language, the number of lines of code, the residence memory size and the virtual memory size of each application.

3.2.1 BLAST: Pairwise Sequence Alignment

Sequence alignment, which analyzes the similarities of biological sequences to understand the structures and functions of gene or protein, is the most important and fundamental problem in Bioinformatics. Pairwise sequence alignment is the basic alignment problem which often serves as a basis for multiple sequence alignment illustrated in 3.2.3.

BLAST[2] is the most popular pairwise sequence alignment tool in the world. The NCBI BLAST includes a set of tools for the alignment of DNA and protein sequences.

PBB includes the BLAST-P program which aligns protein sequences from the NCBI BLAST page. The BLAST-P program have a parallel version to multiple threads execution.

The query data is part of human protein with length 39706 and the database is part of Uniprot[29].

3.2.2 PLSA: Global Alignment

Global alignment is specifically used for very long sequences such as whole genome data[11]. Traditional global alignment requires huge compute power and memory space. For example, aligning sequences with several mega lengths would lead to a memory requirement of several Terabytes.

PLSA is a global alignment application developed by us. Its algorithm can efficiently decompose the whole problem into several smaller independent subproblems, and then be solved in parallel. The detailed presentation of PLSA algorithm can be found in [23].

The data are chosen from a test suite suggested by the bioinformatics group at Penn Stat University [3] with sequence length of 30K.

3.2.3 MUSCLE: Multiple Sequences Alignment

Multiple sequence alignment is a fundamental and challenging problem in computational molecular biology. It can be used to find conserved regions in biomolecular sequences, predict the protein structure, and help constructing phylogenetic tree, etc.

MUSCLE is a multiple sequence alignment application developed by R. C. Edgar[13]. It is more accurate and fast than either ClustalW or T-Coffee which are widely used tools in multiple sequence alignment.

We included the MUSCLE in PBB to represent the multiple sequence alignment applications and [12] presented the parallelization of MUSCLE.

The experimental data are from EMBL-EBI, containing 100 sequences with average length of 330.

3.2.4 Rosetta: Protein 3D structure prediction

Prediction of protein 3D structures is one of the most important problems in Molecular Biology, which can be simply stated as: given the sequence of amino acids of a protein, what is the three dimensional structure?

Prediction of protein 3D structures directly from their amino acid sequences is referred to as "ab initio" method. Among the ab initio tools, Rosetta is one of the best in the international competition of Critical Assessment of Techniques for Protein Structure Prediction (CASP). Rosetta is developed by the Baker laboratory of University of Washington, which uses simulated annealing optimization algorithm (SA) to find optimum protein tertiary structures[9].

We chose Rosetta as a benchmark of PBB and parallelized it. The parallel algorithm and data structures used can be found in [24].

Application	Lang.	KLOC	Residence Size(in KB)	Virtual Size(in KB)	Description
BLAST	C	1350	88056	150880	Pairwise sequences alignment
Plsa	C++	6.35	47420	169192	Global alignment
Muscle	C++	16.1	774372	792952	Multiple sequences alignment
Rosetta	F77	34.6	260936	317996	Protein 3D structure prediction
Semphy	C++	19.7	63564	74980	Phylogenetic tree reconstruction
Modulenet	C++	0.75	142232	142232	Gene regulatory network learning
Snp	C++	0.34	28600	35268	Single Nucleotide Polymorphisms(SNP)

Table 1: Applications in the PBB benchmarks suite

The test data set is named *2ptl* which contains 60 amino acids[21].

3.2.5 *SEMPHY: Phylogenetic Tree Reconstruction*

Phylogenetic tree represents the phylogenetic relationship of species by a tree where closely related species are placed in nearby branches. For DNA/protein sequences from different species, a phylogenetic relationship among them can be inferred to reflect the course of evolution. The phylogeny of genes or species is broadly used in gene family classification, species divergence time estimation and disease associated mutation identification, etc.[17]

There are many approaches to construct the phylogenetic tree, such as the Neighbor Joining, Maximum Parsimony, etc [14], where the maximal likelihood (ML) approach is one of the most accurate methods [20]. However, the ML method is limited in relative small data set for its huge computational intensity, even after incorporating heuristic search techniques.

Nir Friedman et.al proposed a new algorithm SEMPHY for learning ML trees [16]. The main idea of SEMPHY is to use the structural expectation-maximization algorithm [15, 16], a variation of the EM algorithm for structure learning, to efficiently search for maximum likelihood phylogenetic trees. This algorithm is dramatically faster than other ML approaches while reserving comparable accuracy.

We included the SEMPHY in PBB and parallelized it with OpenMP. Detailed information on the parallelized code could be found in [22].

Data are obtained from the Pfam database[8], containing 53 protein sequences.

3.2.6 *ModuleNet: Gene Regulatory Network Learning*

Gene regulatory network learning is to discover the relationship of genes from the gene expression data generated by microarrays. The traditional algorithm is to use the Bayesian networks.

In real world applications, it is often the case that the problem domain is so complex that it contains a mass of variables while there are only a few instances available due to expensive cost of acquiring data. In these cases, the amount of data is insufficient to learn the underlying distribution: statistical noises tend to lead spurious dependencies that significantly overfit the data.

Module network can be viewed as a Bayesian network in which variables in the same module share the same parents and conditional probability distribution. A real world example corresponding to this model is that many genes in a cell are co-regulated by the same factors and exhibit the same expression pattern. Module network introduces the clustering concept into regular Bayesian network and make up its aforementioned defect. Currently it has been successfully used in Gene regulatory analysis[27].

We implemented the Module Network algorithm with the Intel PNL library[4], and parallelized it with both MPI [25] and OpenMP[19]. The OpenMP version is included in the PBB.

The test dataset is of 8880 nodes which will be learned in 296 modules.

3.2.7 *SNP: Pattern Study of Single Nucleotide Polymorphisms*

Single Nucleotide Polymorphisms (SNPs) is subtle variation in a genomic DNA sequence of individuals of the same species. It plays a key role in the pharmaceutical industry to understand variations in drug treatment responses between individuals at the molecular level. Discovering patterns around SNPs loci is very important for better understanding the possible origin of SNPs in evolution.

Bayesian network has been applied to this problem and got promising results. We implemented a SNP pattern study application with Intel PNL library, which includes the routines for Bayesian network learning[26]. The code is also parallelized and characterized on a SMP machine[28].

We have totally 616,179 SNPs sequences as the train dataset, with each has length 50. The dataset is from HGBASE (Human Genic Bi-Allelic Sequences)[10].

4. BENCHMARK CHARACTERISTICS

4.1 Characterization Methodology

In order to understand the characteristics of bioinformatics workloads, we ran PBB on a real system and used the hardware counter to collect the data required for characterization. Table 2 lists the systems we used in this research for both workload characterization and performance measuring.

Workload characterization was only performed on one system, the *QP001* which has 4 physical processors. When HyperThreading is enabled, it has 8 logical processors.

We collected the instruction profile, IPC, cache miss rate for all memory hierarchies and the FSB utilization data for both PBB and SPEC CPU 2000 INT/FP. Intel VTune 8.0 was used for data collection. Because only 3 hardware counters could be tracked at the same time in Pentium 4 processors, data are collected in multiple runs.

4.2 Characterization Results

4.2.1 Instruction Profiles

Figure 1 shows the instruction profiles of both PBB and SPEC CPU 2000 INT/FP. The data was collected with hyperthreading(HT) disabled.

Figure 1: Instruction profiles of PBB applications

Instruction profile of the PBB applications are collected with the number of thread set to 1. PBB_1P_AVG is the arithmetic average of all the 7 benchmarks in PBB when they are executed with only 1 threads. PBB_4P_AVG is the result of PBB executes with 4 threads. It can be seen that the instruction profile of PBB is almost identical, no matter the number of thread is set to 1 or 4. So we only compare the one thread PBB result with SPEC CPU 2000 INT/FP.

From the instruction profile, we can see that the average ratio of load/store instructions is higher than both SPEC CPU 2000 INT and FP.

This result confirms the conclusion of BioBench[6], although we use quite different workloads. However, two applications, *Rosetta* and *Muscle*, exhibits significantly lower percentage of load/store instructions than others, which reflects their compute-intensive nature and the diversity of bioinformatics workloads.

And we can also see that the floating point operations is not negligible, though still significant lower than SPEC CPU 2000 FP.

This result contradicts the conclusion made in BioBench[6] which claims that bioinformatics workloads have almost negligible floating point operations. We believe that this difference is mainly caused by the difference in our selected workloads.

Among the seven benchmarks in PBB, only two of them, BLAST and PLSA, have negligible floating point operations. The result of BLAST, which is also included in BioBench Benchmark Suite, confirms that of the BioBench paper[6]. However, all of the other five applications have non-negligible FP operations. This is caused by the heavy use of probability calculation and score rating in the algorithms.

Nevertheless, the average floating point operation ratio of PBB is still significantly lower than SPEC CPU 2000 FP, which could be viewed as a distinctive characteristics of bioinformatics applications.

Another thing to notice is that *MUSCLE* has the highest percentage of FP operations in PBB, while its counterpart *clustalw* exhibits negligible FP operations. This fact indicates that even for the same application domain, different algorithms may demonstrate completely different characteristics.

4.2.2 CPI

Figure 2: CPI of PBB applications

Figure 2 shows the CPI data of PBB with 1, 4(HT off) and 8(HT on) threads. The data are also compared with SPEC 2000 CPU INT/FP.

It can be observed that CPI of PBB applications, except that of SNP, is significantly lower than SPEC 2000 CPU INT/FP, which is also a characteristic of BioBench. Lower CPI value indicates better instruction level parallelism and usually better performance.

We can see from the figure that CPI changes marginally from 1 threads to 4 threads for most applications. When executed with 8 threads (HT enabled), all applications show a dramatic increase in CPI. However, the CPI of PBB applications of 8 threads is still comparable with the SPEC CPU 2000 FP/INT, and obviously lower than twice the CPI of 4 threads, which suggests the potential of using hyperthreading to improve performance of multi-threaded bioinformatics workloads.

4.2.3 Cache Miss Ratio and Memory Bandwidth

	RefSys	QP001	DC001	Hydra	Osprey	Unisys-Es700
CPU Name	PIII Xeon	XEON	Dual-Core Xeon	Itanium 2	Itanium 2	XEON
CPU Freq.	700MHz	2.8GHz	3.2GHz	1.3GHz	1.5GHz	3.0GHz
L1 D-cache	16KB 4-way	8KB 4-way	16KB 8-way	16KB	16KB	8KB
L2 Cache	1MB 8-way	512KB 8-way	4MB 8-way	256KB	256KB	512KB
L3 Cache	-	2MB 8-way	-	3MB	6MB	4MB
L4 Cache	-	-	-	-	-	32MB
# of Chips	4	4	2	4	4	16
HT Support	N	Y	Y	N	N	Y
Interconnect	FSB	FSB	FSB	FSB	FSB	FSB and Crossbar
Memory	1GB	4GB	4GB	4GB	16GB	8GB

Table 2: Systems used for workload characterization and performance measuring

Figure 3: Cache Miss Ratio of PBB applications

One of the major challenges to the future computer system is the bandwidth demand of multi-thread applications. Figure3 demonstrates the cache miss ratio for all the three levels of cache in the system and Figure4 presents the memory bus bandwidth utilization of PBB applications at 1, 2, 4(HT off) and 8(HT on) threads.

The cache miss ratio statistics show that *MUSCLE* and *SNP* have higher cache miss ratio than other applications. This accords to the bandwidth data in that they demand significantly more bandwidth than other PBB applications.

It can be seen from Figure4 that all the other applications except *MUSCLE* and *SNP* have low memory band-

Figure 4: Memory bandwidth Usage of PBB applications

width demands than the SPEC CPU applications. Even at 8 threads, these five applications only consume less than 400MB/s, comparable to SPEC CPU INT, and much less than SPEC CPU FP.

However, *MUSCLE* and *SNP* require much higher memory bandwidth than the others. The maximum bandwidth usage for them are 1460MB/s and 1260MB/s, respectively. However, the system memory on QP001 has a theoretical peak bandwidth of only 2100MB/s, which usually discounts by about 30% due to bus and DRAM servicing transactions in multi-processor environment. So *MUSCLE* and *SNP* are actually approaching the upper bound of memory bandwidth.

The cause of high memory bandwidth requirement for *MUSCLE* is mainly due to the large size dataset. The dataset on which *MUSCLE* is executed is a sparse matrix at a size of 6MB, which is larger than the L3 cache used in this machine. So most of the L1 and L2 cache misses goes straight to main memory, which can be seen in the high L2/L3 cache miss rate, forming a high demand for memory bandwidth. As for *SNP*, the threads in *SNP* exchange their data frequently, which incurs large amount of FSB transactions, which is measured as memory accessing operations in our experiments. Either way, the memory system could be a bottleneck when these two programs are run on a machine with more processors.

5. PERFORMANCE RESULTS

5.1 Benchmark scores

We followed the paradigm of SPEC CPU 2000 and defined the score of PBB. We set a reference system which is a server with four PIII Xeon processors. The parameters of this machine is illustrated in Table 2 as the "RefSys".

Although we have seven benchmarks in PBB, only six benchmarks are calculated for scores. *Rosetta* is excluded for this purpose because it is a random algorithm and its result varies from time to time. It is very difficult to define sensible performance metrics for *Rosetta*. We plan to solve this problem in our future work. Currently, we keep it in the benchmark for workload characterization only.

Each PBB application was executed on this reference system with 1, 2 and 4 threads. We chose the shortest time of several runs as the reference time for this application regardless of the number of threads. Table 3 shows the PBB performance on the reference system and the time with the asterisk was chosen as the reference time for each application respectively (denoted as r_i for application i , where $1 \leq i \leq 6$).

Then we defined the score of a system by measuring execution time of each application on the system with various number of threads and the shortest time is chosen for that application (denoted as m_i for application i , where $1 \leq i \leq 6$). Then the PBB score could be calculated as

$$PBB_Score = \sqrt[6]{\prod_{1 \leq i \leq 6} \frac{r_i}{m_i}} \times 100$$

We measured PBB on several real systems, including a server with Intel's dual-core Xeon processor and a 16-processor SMP machine. The parameters of these systems are shown in Table 2. Their PBB scores are shown in Table 4.

The two IA64 systems don't have high scores mainly because that they have poor performance on *SNP* and *ModuleNet*. These two applications all use the *PNL* library which has been optimized for X86 architecture. We expect optimization on *PNL* library for IA64 to be able to improve the performance of PBB on IA64 systems.

Comparing the two Xeon systems, *QP001* and the *DC001*, we could see that *DC001* is 30% better than *QP001* although the frequency of processors is only 14% better. This performance gain could be contributed by larger L2 cache (4MB vs 512 KB) and higher FSB frequency (1066MHz vs 400 MHz). Recalling that *SNP* and *MUSCLE* have the greatest memory bus bandwidth demand in PBB, it is easy to explain why they get the most performance improvement on *DC001*.

5.2 Parallel Speedup

The speedup of PBB on a 16-P SMP server (Unisys-Es700) is presented in Figure 5. The result shows that five out of six benchmarks get satisfactory speedup. *SEMPHY* and *MUSCLE* even get almost linear speedup. However, the *BLAST - P* seems to be left behind. The main cause for the poor performance is that the total instructions executed by *BLAST - P* increases dramatically with the number of threads (from 190 billion for 1 threads to 566 billion for 4 threads). This increase in instruction count may indicate a

design fault in the implementation.

Overall, this result is encouraging because it indicates that future multi-core processors could provide significant performance gain for parallel bioinformatics workloads.

Figure 5: Speedup of PBB on Unisys-Es700

5.3 HyperThreading Effects

Hyperthreading (HT) techniques are supported in Xeon processors, which aims at hiding memory access latencies by providing two logical processors with one physical processor. We evaluated the effects of HyperThreading with PBB on the *DC001* and the results are demonstrated in Figure 6. The 8p data are obtained with HT enabled and all other data are obtained with HT disabled.

It can be seen from Figure 6 that HyperThreading provides around 10% performance improvements for 4 out of 6 benchmarks. For *SEMPHY*, the speedup is more than 20%. *PLSA* is the only one application in PBB that suffers from performance loss with HT.

The speedup provided by HT can be explained by hiding the memory access latencies. A Xeon processor with HT technology provides two logical processors, which shares the same execution units. When a logical processor stalls on memory access, the other can perform memory-free operations, thus hiding the memory access latencies with computation. However, due to the shared execution units, this technology is not expected to benefit programs with rare memory access. And if the two threads executed on two logical processors both access memory too frequently, they won't benefit from HT, either.

From Figure 4 we can see that most of our applications have moderate memory access frequency, which makes HT beneficial to our programs.

6. CONCLUSION

This paper presents a bioinformatics benchmark suite PBB, a parallel benchmark suite focusing on bioinformatics applications. It covers several important application domains that are missed in previous benchmarks.

We performed characterization on PBB and compared it with SPEC CPU INT/FP. Several distinct features of bioinformatics workloads are identified.

# of threads	plsa	muscle	semphy	blastp	pnl	snp
4	114.9*	370.6*	113.5*	308.3*	69.00*	108.1*
2	224.8	555.4	210.0	466.0	133.7	187.3
1	445.7	1002	402.4	775.0	259.0	288.5

Table 3: The reference time of PBB applications(in seconds)

App.	Ref	QP001	DC001	Hydra	Osprey	Unisys-Es700
Plsa	114.9	45.4	32.0	55.6	48.0	19.3
Muscle	370.6	75.0	53.0	114	93.0	21.0
Semphy	114	25.2	21.5	49.8	32.2	22.3
Modulenet	68.1	18.2	15.1	62.0	54.8	7.58
BLAST-P	308.3	80.0	75.0	87.5	76.0	54.7
SNP	108	27.9	14.7	77.9	68.8	15.7
PBB Scores	100	383	503	208.5	254	756

Table 4: Scores of PBB on several real systems

Figure 6: HyperThreading Effects on DC001

formatics workloads were identified: (1)High percentage of load/store instructions; (2)Non-negligible floating point operations, but still significant less than SPEC CPU 2000 FP; (3)Low CPI and (4)Low memory bandwidth demand for most applications.

We defined a scoring method for PBB and presented PBB scores for several real systems, including a server with Intel's dual-core Xeon processor and a 16-processor SMP machine. Most of the applications in PBB shows satisfactory speedup up to 16 threads. The effect of HyperThreading technique is evaluated. The conclusion is that HyperThreading could provide modest performance improvement for most PBB applications.

Overall, the results of characterization and analysis of PBB suggest that multi-core processors could be used to support parallel bioinformatics workloads effectively.

In the future, we plan to add a few more applications into the next version of PBB. We are finalizing the copyright issues to make PBB freely available to the community. Besides, we will perform more in-depth analysis on these workloads.

7. REFERENCES

- [1] Bioparallel: A suite of parallel bioinformatics workloads. <http://www.ece.umd.edu/biobench/biobench.html#parallel>.
- [2] Ncbi blast. <http://www.ncbi.nih.gov/BLAST/>.
- [3] Penn state university center for comparative genomics and bioinformatics. http://www.bx.psu.edu/miller_lab/.
- [4] Pnl: Probabilistic network library. <http://www.intel.com/technology/computing/pnl/>.
- [5] Standard performance evaluation corporation. <http://www.spec.org>.
- [6] K. Albayraktaroglu, A. Jaleel, X. Wu, M. Franklin, B. Jacob, C.-W. Tseng, and D. Yeung. Biobench: A benchmark suite of bioinformatics applications. In *ISPASS 2005*, pages 2–9, March 2005.
- [7] D. Bader, Y. Li, T. Li, and V. Sachdeva. Bioperf: A benchmark suite to evaluate high-performance computer architecture on bioinformatics applications. In *The IEEE International Symposium on Workload Characterization (IISWC 2005)*, October 2005.
- [8] A. Bateman, E. Birney, R. Durbin, S. R. Eddy, K. L. Howe, and E. L. L. Sonnhammer. The pfam protein families database. *Nucleic Acids Research*, 28(1):263–266, 2000.
- [9] R. Bonneau, J. Tsai, I. Ruczinski, D. Chivian, C. Rohl, C. Strauss, and D. Baker. Rosetta in casp4: progress in ab initio protein structure prediction. *Proteins Suppl*, 5:119–126, 2001.
- [10] A. J. Brookes, H. Lehtväslaiho, M. Siegfried, J. G. Boehm, Y. P. Yuan, C. M. Sarkar, P. Bork, and J. F. R. Ortigao. HGBASE: a database of SNPs and other variations in and around human genes. *Nucleic Acids Research*, 28(1):356–360, 2000.
- [11] A. L. Delcher, S. Kasif, R. D. Fleischmann, J. Peterson, O. White, and S. L. Salzberg. Alignment of whole genomes. *Nucl. Acids. Res.*, 27(11):2369–2376, 1999.
- [12] X. Deng, E. Li, J. Shan, and W. Chen. Parallel implementation and performance characterization of muscle.
- [13] R. C. Edgar. Muscle: multiple sequence alignment with high accuracy and high throughput. *Nucleic Acids Research*, 32(5):1792–1797, Mar. 2004.

- [14] J. Felsenstein. *Inferring Phylogenies*. Sinauer, 2001.
- [15] N. Friedman. Learning belief networks in the presence of missing values and hidden variables. In *Proc. 14th International Conference on Machine Learning*, pages 125–133. Morgan Kaufmann, 1997.
- [16] N. Friedman, M. Ninio, I. Péér, and T. Pupko. A structural EM algorithm for phylogenetic inference. In *the Fifth International Conference on Computational Biology (RECOMB-01)*, pages 132–140, New York, Apr. 22–25 2001. ACM Press.
- [17] M. Holder and P. Lewis. Phylogeny estimation: Traditional and bayesian approaches. *Nature Reviews Genetics*, 4:275–284, 2003.
- [18] A. Jaleel, M. Mattina, and B. Jacob. Last-level cache (llc) performance of data-mining workloads on a cmp—a case study of parallel bioinformatics workloads. In *12th International Symposium on High Performance Computer Architecture*, February 2006.
- [19] H. Jiang, C. Lai, W. Chen, Y. Chen, W. Hu, W. Zheng, and Y. Zhang. Parallelization of module network structure learning and performance tuning on smp.
- [20] W. JJ and S. MR. Phylogenetic analysis and intraspecific variation: performance of parsimony, likelihood, and distance methods. *Syst. Biol.*, 47:228–253, 1998.
- [21] S. KT, R. Kooperberg, C. Fox, B. Bystro, and C. Baker. Improved recognition of nativelike protein structures using a combination of sequence-dependent and sequence-independent features of proteins, 1999.
- [22] E. Li, Z. Ouyang, X. Deng, Y. Zhang, and W. Chen. Parallel implementation of semphy—a structural em algorithm for phylogenetic reconstruction, February 2005.
- [23] E. Li, C. Xu, T. Wang, L. Jin, and Y. Zhang. Parallel linear space algorithm for large-scale sequence alignment. In *Euro-Par*, volume 3648, pages 1207–1216. Springer, 2005.
- [24] W. Li, T. Wang, E. Li, D. Baker, L. Jin, S. Ge, Y. Chen, and Y. Zhang. Parallelization and performance characterization of protein 3d structure prediction of rosetta.
- [25] L. Liu et al. An implement of parallel module network learning algorithm on distributed memory multiprocessors. In *The 7th International Workshop on High Performance Scientific and Engineering Computing (HPSEC-05) (34th ICPP’2005)*, pages 129–134, Oslo, Norway, June 2005. IEEE Computer Society. Tsinghua University, Beijing, PRC.
- [26] X. Ma, J. Cai, W. Hu, Y. Zhang, Y. Li, and X. Zhang. Discovering possible context dependences around snp sites in human genes with bayesian network learning. In *ICARCV*, volume 2, pages 1315–1319, 2004.
- [27] E. Segal, D. Pe’er, A. Regev, D. Koller, and N. Friedman. Learning module networks. *Journal of Machine Learning Research*, 6:557–588, 2005.
- [28] J. Song, E. Li, W. Hu, S. Ge, C. Lai, Y. Zhang, X. Zhang, W. Chen, and W. Zheng. Parallelization of bayesian network based snps pattern analysis and performance characterization on smp/ht. In *ICPADS*, pages 315–322, 2004.
- [29] C. H. Wu, R. Apweiler, A. Bairoch, D. A. Natale, W. C. Barker, B. Boeckmann, S. Ferro, E. Gasteiger, H. Huang, R. Lopez, M. Magrane, M. J. Martin, R. Mazumder, C. O’Donovan, N. Redaschi, and B. Suzek. The universal protein resource (uniprot): an expanding universe of protein information. *Nucleic Acids Res*, 34, 2006.