

Characterization of Bioinformatics Applications on Multimedia Processor

N. Z. Azeemi¹, A. Sultan²

¹Christian Doppler Laboratory for Design Methodology of Signal Processing Algorithms, Institute of Communications and Radio Frequency Engineering, University of Technology Vienna Austria

²Division of Molecular Cell Biology, Max F. Perutz Laboratories, Vienna Biocenter Austria
Emails: nzafar@nt.tuwien.ac.at, aneesa.sultan@univie.ac.at

Abstract

The explosive growth in size of genomic data and increase in computational complexity of bioinformatics applications has fueled the issue of appropriate underlying hardware architecture choice. Another impediment is the latest trend towards handheld/mobile bio-computational devices. One of the most critical factors holding back mobile computing is the perfect match between hardware architecture and application expression profile (AEP). This paper analyzes AEP for several bioinformatics application on a high end processor. We found that the synergy of bio-applications and hardware architecture is hampered by many debilitating factors, such as application algorithm, input sequences, architecture constraints, to name just a few. The result of the proposed approach are measured at our profile monitor framework that characterize static and dynamic behavior of bioinformatics applications based on instruction profile, scheduling factor, slot utilization, and on-chip/off-chip memory usage. Experimental results show very high application-architecture correlation between our target platform (Nexperia PNX1302) and well known applications such as NetPlanGene (87%), Manfinder (92%), Fgene (82%), GlimmerHMM (68%) and Mummer (75%). Impact of different bio-computing algorithms on cache performance is also studied. The result is important for developing general methodology for highly efficient handheld bioinformatics devices.

Keywords:

Bioinformatics, High-end Processors, Application Expression Profile, Workload Characterization.

1 Introduction

Bioinformatics applications are expected to form a large part of the workload on a growing number of systems, including future handheld computers, mobile devices and desktop systems. This paper analyzes the application behavior at instruction level granularity on a high end multimedia processor. Although current high end processor architectures are perceived to deliver an optimal performance for bioinformatics applications but there is little quantitative data to support [1, 2, 3, 6, 7].

Systems are software running on hardware. The ideal performance of a computer system demands a perfect match between machine

capability and program behavior. Machine performance varies from program to program; this makes peak performance an impossible target to achieve in real-life applications. The mapping of algorithmic and data structures onto the machine architecture include computational power, scheduling, memory maps, etc., these activities are usually architecture dependent. Aim of this work is to find basic characteristics of coded application and its runtime profile. We call the static behavior of application as static application expression profile (sAEP) and runtime profile as dynamic application expression profile (dAEP).

This work is motivated by two broad set of implications, first the perception that the implementation of aggressive architectural features such as branch prediction, caches and instruction issue mechanism is increasing execution time unpredictability and making them undesirable for bioinformatics applications [2,6,7]. Typical bioinformatics applications periodically process a set of data, and it important to be able to predict the execution time for the application. If execution time is unpredictable, then it is difficult to know how much processing power to schedule to guarantee a desired input sequence rate, or, conversely, what sequence is sustainable with a given amount of processing power. The second motivation for our work concerns the highly streaming nature of genomic data, which allow us to represent an application in terms of metrics based on their sAEP and dAEP at the underlying architecture.

Compared with traditional application characterization methodologies [1, 2, 3], our scheme has following advantages.

- **A novel approach.** Application behavior is measured for capturing and correlating application-architecture performance problems
- **A comprehensive benchmark.** Characterization is done at a set of 16 bioinformatics applications from 11 diversified bio-computing domains.
- **Profiling framework is portable across popular commercial platforms.**

The structure of this paper is as follows. Section 2 presents an overview of bioinformatics applications and our workload. The profile monitor framework is described in Section 3. Section 4 presents detailed case studies demonstrating how our framework can be used to better exploit the architectural benefits. Section 5 concludes the paper with future work.

¹ This work has been funded by the Christian Doppler Laboratory for Design Methodology of Signal Processing Algorithms.

2 Bioinformatics Benchmarks

This section presents an introduction to bioinformatics and research areas. Section 2.1 describes the basics of bioinformatics applications. Section 2.2 presents the benchmarks.

2.1 Bioinformatics Applications

Bioinformatics and computational biology applications use mathematical tools to extract useful information from data produced by high throughput biological techniques such as genome sequencing. The assembly of high quality genome sequences from fragmentary DNA sequencing has always been challenging. DNA sequencing is the process of determining the nucleotide order for a given DNA fragment, called the DNA sequence. Nucleotides are the structural units of RNA, DNA, and several cofactors - CoA, FAD, FMN, NAD, and NADP. They play important role in cell for e.g., energy production, metabolism, and signaling. A DNA sequence or genetic sequence is a succession of letters representing the primary structure of a real or hypothetical DNA molecule or strand, with the capacity to carry information. Bioinformatics deals with algorithm, computational and statistical techniques to address the information extraction issue from a genetic sequence. Most common application domains are:

- Sequence analysis
- Gene expression analysis
- Regulation Analysis
- Protein expression analysis
- Protein structure prediction
- Genomic Divergence
- Biological System Modelling
- Evolutionary computational biology
- Biodiversity Measurement
- Cancer mutation analysis
- High throughput image analysis

2.2 Benchmark Description

We chose applications for their importance in real systems and to be representative enough to make the inferences in this study – and is summarized in Table 1. We obtained codes for these applications from various public domain sources. For lack of space, we only report their underlying algorithm; details may be found in [7, 8, 11]. The input databases are obtained from NIH genetic sequence database ‘GenBank’, NCBI assembly archive ‘Genome Assembly Archive’, Homologous structure alignment database ‘HOMSTRAD’, the NIMH-NCI protein-disease database ‘PDD’ and ‘The Lens’ [9, 10, 11].

3 Profiling Framework and Methodology

This section describes our infrastructure. Section 3.1 introduces the simplified view of our target hardware architecture. Section 3.2 describes the profiling framework. Section 3.3 presents the implementation of the methodology.

3.1 Architecture Studied

Figure 3.1 shows a simplified VLIW (very long instruction word) architecture composed of CPU core, instruction and data cache units, memory management units and address/ data highway.

3.2 Profile Monitor Framework

The methodology framework is shown in [5,12], briefly, the source code is processed successively for static code analysis, post compiler analysis and finally for scheduling analysis. In each step it generates a list of parameter to be used in our transformation cost model, explained in [5]. These extracted features are software and architectural dependent, attributed to executing application and target processor. Intermediate trace files are generated during code processing flow (i.e. pre/post compiler, scheduler etc.) to produce size, number of cycle, number of hits (for both caches), cache bank conflicts, cache access overlaps, highway activity factor, scheduling index, issue slot index for dynamic instructions and number of live registers. A VLIW descriptor file is used to provide architecture information to compiler, scheduler and finally to the machine code generator. This feature has made our scheme architecture independent. Iteratively after each cycle, all these parameters are recorded again and are compared to preset cost functions.

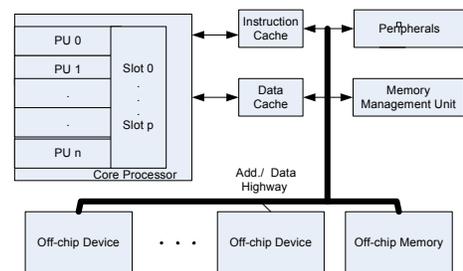


Figure 3.1: General architecture of VLIW processor

3.3 Experimental Setup

The benchmarks were compiled for the Nexperia DSP PNX1302 platform on 200MHz board running pSoS operating system, using the TriMedi C compiler cc and the optimization level -O3 -G. We generated the suitable input files using the nucleotide/ protein databases as mentioned in Section 2.2. The output file sizes were selected to yield at least several tens of billions of instruction executions for all of our computational biology benchmarks. For our bio-benchmarks, we skipped the first 100 million instructions and collected data for the next 1 billion instructions or until completion. All analyzed benchmarks were validated against precompiled binaries provided in the original distributions of the benchmark suites. We use the TriMedia sim [4] simulator v2.2.2 of tcs2.0007 to run the benchmarks and collected data. The benchmark profiles were obtained using the TriMedia improf profiling simulator, and the performance data was collected using our framework [5].

4 Results

We analyze the extent and the nature of the variability in the sAEP. Such static operations involve load/store operations, conditional/non-conditional branches, int/floating operations and the exceptions handling procedures. The results are shown in Figure 2. At first

Table 1. Bioinformatics Applications Workload Description		
Package/ Application	Deliverable	Features
GLIMMER	Coding regions in microbial DNA	Interpolated Markov Models (IMM)
GENEID	Find genes in eukaryotes	Hierarchical rule-based system
GENESPLICER	Detect splice sites in the genomic DNA	High accuracy and computationally efficient
TIGRSCAN	DNA modeling	Generalized Hidden Markov Model (GHMM), HMM
TRANSTERMIS	Rho-independent transcriptional terminators	Statistical estimation techniques
GENSCAN	Predict complete gene structure	Search algorithms
MUMMER	Genome Sequence alignment	Tree algorithms
BAMBUS	Orders and orients contigs into scaffolds	
GLIMMERHMM	Find gene sequence in eukaryotes	IMM, Splice site models, Maximal dependence decomposition
GENIE	Gene finder in vertebrate and human DNA	GHMM, Neural Networks
FGENE	Find splice sites, genes, promoters	Linear discriminant analysis
GRAIL	Analysis of DNA sequence	Automated computation
GENEMARK	Find genes in bacterial DNA sequence	Markov chains
MarFinder	Process differential gene expression, DNA replication	Matrix Association Regions, Statistics pattern matching.
NetPlaneGene	Sequence analysis	Neural network
TESS	Transcription factor for binding sites	Data base application

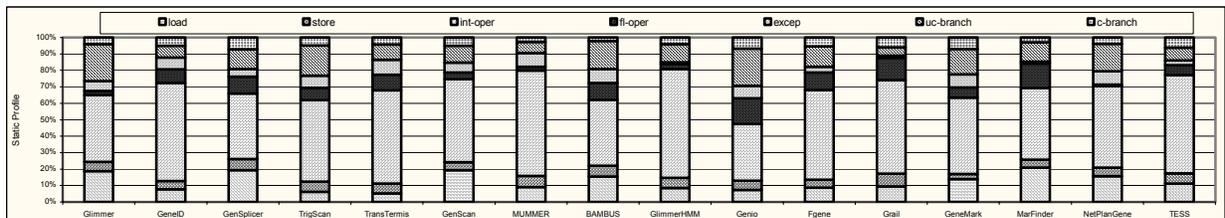


Figure 2: Static Application Expression Profile (sAEP)

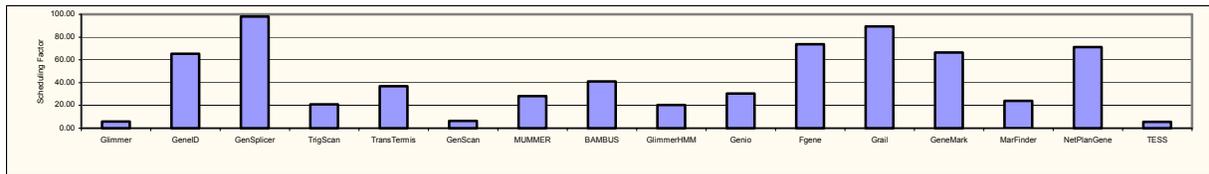


Figure 3: Level of Architectural Constraints offered to Bioinformatics Applications Benchmarks

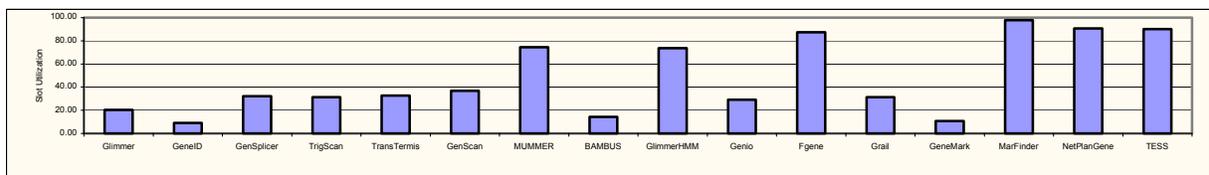


Figure 4: Level of Parallelism in Bioinformatics Applications Benchmarks

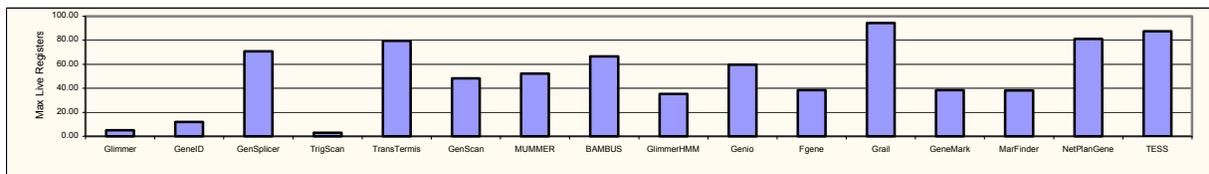


Figure 5: Compactness of Code in Term of Maximum Live Onchip Register

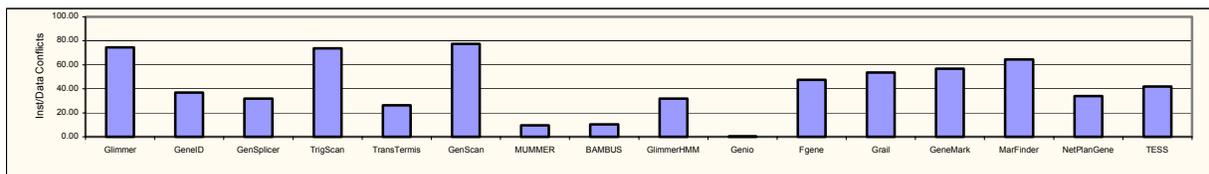


Figure 6: Instruction/ Data Conflicts in Bioinformatics Applications Benchmarks

glance, the interesting outcome is the dominating integer operations in benchmark. GlimmerHMM (67%) has the large number of integer operations because it utilize interpolated Markov Model as well as the maximal dependence decomposition technique for improving specificity in splice site identification. Being adapted from Glimmer, one can observe the next higher number of instructions is static branching. While Genio, Grail and MarFinder have large number of floating point operation (13%-20%), illustrating the relatively compute intensive nature of these applications. The later uses statistical patterns to deduce the matrix association regions in DNA sequence to further facilitate differential gene expression and DNA replication. Moreover, load/store operation in most applications contribute 15% to 27% of all operations. A careful observation in their source codes reveals that these applications use mostly search algorithms, rule based schemes, linear discriminant analysis and database applications. All these techniques are based on heavy integer manipulation.

Our Target architecture offers high parallelism [4], Figure 3 and Figure 4 show the fact how much a bioinformatics application can exploit such parallelism. Figure 3, shows the scheduling factor, which is a ratio as if some operations are performed at a machine with infinite number of processing elements to the finite processing element machine. It gives an important measure as if an application is suitable for a low speed parallel machine or a high speed uniprocessor machine. GeneSplicer (97%), based on Interpolated Markov Model and is a good candidate for uniprocessor machines. NetPlanGene (87%), Manfinder (92%), Fgene (82%), GlimmerHM (68%) and Mummer (75%) are showing a high degree of parallelism in their implementation in Figure 4. Mummer uses suffix tree and can run in parallel the sequence alignments containing millions of nucleotides. Figure 5 reveals a correlation between the scheduling factor and maximum internal register usage for GenSplicer (97%, 69%) and same is true for Grail (89%, 94%).

Figure 6 shows the instruction to data cache conflicts, it happens mainly because of cache data bank conflicts, but Geneid shows overall a good tendency to utilize our target architecture, it has very low instruction/ data cache conflict, primarily due to very high inherent data locality. This allows it to map all local variables in code to internal register and saves off-chip traffic that leads to high execution efficiency and low energy consumption.

Our profiling framework yields another interesting result about TigrScan bioinformatics application. Generally it is known as highly time-efficient and space-efficient, due to use of queues and propagators. But our profile shows a completely different picture, this is the poorest application for a parallel architecture like VLIW processors. The exact reason for such behavior is not know to us yet.

5 Conclusions

This paper introduces application expression profile, an approach for understanding performance related issue in mapping modern bioinformatics applications to a high end processor. This approach monitors behavioral information about application at all layers and correlates them with underlying architecture. We have built a profile monitor framework to capture sAEP and dAEP, and finally to correlate with architecture capabilities. We found the widespread parallelism and regular communication patterns in different bio-computational applications that further can be exploited to architecture-based optimization both for cycle and energy efficiency on handheld devices.

Our results are important for developing a general methodology for energy-aware bioinformatics handheld/ mobile devices. In addition, we expect to be able to model such applications conformance to the target architecture based on their expression profile in terms of energy consumption, execution time and architecture usage.

6 References

- [1] K. Albayraktaroglu, M. Franklin "Workload characterization of Modern Computational Biology Applications," Proc. of ICISIP-2005, pp. 435-440.
- [2] Y. Li, T. Li, , "Workload Characterization of Bioinformatics Applications," Proc. of the Int. Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems 2005, pp. 15-22, Sep 2005.
- [3] N. Zafar, M. Rupp, "Energy-aware source-to-source transformations for a VLIW DSP processor," Proc. of the 17th IEEE ICM 2005, pp. 133-138, Dec. 2005.
- [4] TM1300 Data Book, Philips Electronic, North America Corporation, pp. 3.1-3.16, Oct 1999.
- [5] N. Zafar Azeemi, "Power Aware Framework for Dense Matrix Operations in Multimedia Processors," Proc. of the 9th IEEE International Multi-topic Conference, Dec. 2005.
- [6] The Apple Workgroup Cluster for Bioinformatics, http://images.apple.com/xserve/cluster/pdf/Workgroup_Cluster_PO_021104.pdf
- [7] BioSpace, <http://www.biospace.com/>
- [8] Genomeweb Daily News, <http://www.genomeweb.com/>
- [9] NCBI, <http://www.ncbi.nlm.nih.gov/>
- [10] The UniProt/Swiss-Prot Database, <http://www.ebi.ac.uk/swissprot/>
- [11] The NCBI Bacteria genomes database <ftp://ftp.ncbi.nih.gov/genomes/Bacteria/>
- [12] N. Zafar Azeemi, "A Framework for Architecture Based Energy-Aware Code Transformations in VLIW Processors," Proc. Of the IEEE International Symposium on Telecommunications (IST 2005), pp.393-398. Sep. 2005.