Parameterized Characterization of Bioinfomatics Workload on SIMD Architecture

Naeem Z. Azeemi
Christian Doppler Laboratory for Design Methodology of Signal Processing Algorithms, Institute of Communications and Radio Frequency Engineering, University of Technology Vienna Austria
Email: nzafar@nt.tuwien.ac.at

A. Sultan
Division of Molecular Cell Biology, Max F. Perutz Laboratories, Vienna Biocenter Austria
Email: aneesa.sultan@univie.ac.at

A Arshad Muhammad
Centre of Biomolecular medicine and Pharmacology, Medical Univ. of Wien
Lazarettgasse 19, 1090 Wien, Austria

***Abstract*—Bioinformatics applications expression profile is a critical performance metric in high end genomic data processing. These profiles are compute intensive and offers a wide range of computation pattern ranging from data base searching applications to highly irregular phylogenetic trees. Such debilating factors lead to poor architecture-application correlation. A prior knowledge of software application would be very useful for an efficient bioinformatics embedded system design. This paper presents an integrated approach for parameterized workload characterization and trace driven simulation at a high end multimedia processor in bioinformatics applications. A set of sixteen widely used bioinformatics applications is selected as benchmark. Software monitoring techniques are used to collect execution traces. Based on the measured results, we investigate both the computation and communication behavior of these applications, including execution time, functional unit utilization, scheduling factors and cache misses. The temporal and spatial localities in gnomic data are also discussed. Experimental results are measured at a high end SIMD (single instruction stream over multiple data stream) processor.**

***Keywords*—Bioinformatics Applications, SIMD, Workload Characterization.**

I. INTRODUCTION

High-end multimedia processors have recently offered cost-effective and feasible approach to compute-data intensive applications both for mobile devices and set-top boxes. Despite the proliferation of small, inexpensive embedded devices that reflects a new era of mobile computing, new mobile architecture are assumed to meet stringent performance requirements [1, 2, 3]. In the same vein, the exponential growth in genomic data and entailed computational complexity [2, 4] has reduced the modern architectural gains for fast and efficient computation. Little work has been done to explore the bioinformatics applications behavior and their correlation with the underlying architecture. Major break through in shrinking the integrated circuits geometry have brought the ubiquitous computing on handheld devices. These devices offer variety of bioinformatics applications with embedded OS E.g., Windows Mobile, Symbian, Palm OS, and embedded Linux etc. Such hardware malleability permits portability, flexibility in application installation and quick availability in fields [6, 7, 15, 16, 17].

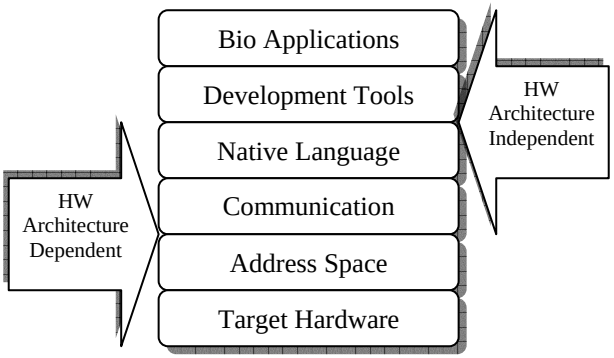

Figure 1. Bioinformatics System Development Layers

Moreover, the performance of these devices strictly depends on their data processing speed and the battery backup time. The lifetime of battery is determined by the computational workload, and thus, workload characterization has emerged as an important design metric. On the other hand the independent software vendors (ISV) provide interface independent applications [2]. The real time nature of such software further complicates the performance issues. A typical bioinformatics application development cycle is shown in Figure 1. 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, and memory maps. These activities are usually

architecture dependent. The motivation behind 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 and runtime profile as dynamic application expression profile.

Several research groups have focused upon an application level workload characterization [2, 4]. They characterize the bioinformatics applications to be collection of group of instructions whose combined operation provides a useful service and discuss the impact of dynamic performance on customized processor. However, they focus on application level expression profile; the architecture correlation is completely ignored in results. There are several system level parameters that need to be measured during application build flow as well as application run time profile. For example, the cyclomatic complexity can reduce the benefits offered by the underlying parallel architecture. Similarly algorithmic complexity can add scheduling overhead at the cost of cache misses; eventually lead to increase CPU cycles and energy. In this work we measure electrical parameters of application run time profile. The aim of these measurements was to validate the assumption typically made by researchers in the bio-application modeling area [1, 4, 14], that the implementation of high profile architectural features such as instruction issue slots, branch prediction and caches is increasing unpredictability in process completion time and making them undesirable for bioinformatics applications. There are several application level factors that need to be evaluated E.g., it is important to be able to predict the execution time for the application in addition to the conventional software optimization. These measurements guarantee the desired input sequence rate and hence make sure the optimal system performance.

The contribution of this work is as follows:

- A comprehensive application behavior is measured for capturing and correlating application-architecture performance problems
- Case studies consist of a set of 16 bioinformatics applications.
- Results are reported based on real measurement at SIMD processor

Rest of the paper is organized 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.

II. BIOINFORMATICS AND SEQUENCE ANALYSIS

This section introduces an overview of bio computation in Section A. Section B describes the benchmark applications that are selected from widely use websites.

A. Overview of Computational Biology Applications

Bioinformatics is the application of computer technology to the management and analysis of biological data. The umbrella activities are recording, annotation, storage, analysis, and searching/ retrieval of nucleic acid sequence (genes and RNAs), protein sequences and structural information.

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. The genes are located on the DNA which is found in the nucleus of the cell. Protein synthesis happens outside the nucleus in the cytoplasm on the ribosome. The processes that are involved in making proteins from our genes are called transcription and translation and the molecules that are involved in theses processes are called DNA, mRNA, tRNA and proteins. The order of information transfer is DNA to RNA to protein is shown in. Figure 2.1 depicts how genes encode proteins.

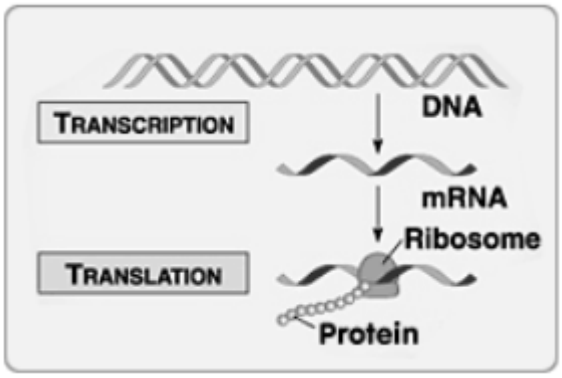

Figure 2.1, Schematic of Protein Encoding (© 1999, Addison Weseley Loman Inc.)

The exponential growth in number of genetic sequences at GenBank, EMBL [5, 7, 8] repository give rise to a deluge of genomic data generation and management. Bioinformatics is an interdisciplinary research area that helps to produce 'sensible' and 'useful' information from the wealth of data that has been produced by the genome sequencing projects. We categorize the basic functionality offered by all bioinformatics tools into four groups, they are:

1. **Algorithm for pattern recognition**, probability formulae are used to determine the statistical similarity in given two or more than two sequences
2. **Rule-bases analysis**, which define how a mathematical or statistical technique can be applied. Different sets are defined with a membership, and set of rules are also created to elaborate the associativity. Basic set theory is used to fire a rule.
3. **Biological data bases** are uniformly and efficiently maintained archives of consistent data that contain information and annotation of DNA and protein sequences, DNA and protein structures and DNA and protein expression profile. An important feature of these databases is their simplicity in access and query

management. In addition some website [9, 10, 11] provides visualization tools to aid biological interpretation.

4. **Biological taxonomy** records the differences in sequences across different classes, it helps further to reduce the similarity errors.

Few umbrella activities are briefly described below:

- **Sequence Searching**: Similarity between the DNA or protein sequences in a large database is the main objectives in such applications. BLAST and FASTA are two widely use tools used to perform such analysis.
- **Region Finding (RF)**: This is achieved by aligning multiple sequences to find coherent regions. It helps to identify the evolution tree structure. CLUSTAL W is used to locate such protein patterns.
- **Common Signature Locator (CSL):** These tools identify the similarity across the database from diverse domain E.g., across the mice and human genome. It helps to identify same protein pattern in different origins.
- **Evolutionary Computational Biology**: This digs deeper the evolutionary pattern and perform phylogenetic analysis to locate common origin.
- Other application domains are gene expression analysis, regulation analysis, and mutation analysis, to name a few.

B. Benchmark Description

As mentioned above the gap between bioinformatics applications and underlying hardware architecture is primarily due to the lack of research on application expression profile. In order to evaluate performance of bioinformatics applications, we make it necessary to choose bioinformatics applications from diversified domain and must be most popular in use. Our benchmark suite is tabulated in Table 1. We use pseudonym in our results for the sake of clarity and are mentioned as A01-A12 in Table 1. The input database is obtained from public domain sources [7, 8, 9, 10, 11, 12, 13]. Further detail about the benchmark suite can be found in [13].

III. BASLINE SYSTEM

A layered development of typical embedded system is shown in Figure 1, based on a recent classification [2]. The technology of parallel processing is the outgrowth of rapid progress in hardware technology and the increased feasibility of building a new generation of embedded systems adopting parallel processing. The growing trend towards ISV, developers are more inclined to develop machine independent application programs and programming environments. From a programmer's viewpoint, SIMD (single instruction stream over multiple data stream) architectures are lucrative choice for exploiting spatial parallelism rather than temporal parallelism. SIMD computing use an array of processing elements. Despite the architectural freedom, the major barrier preventing parallel processing being widely adaptation is the software and application side.

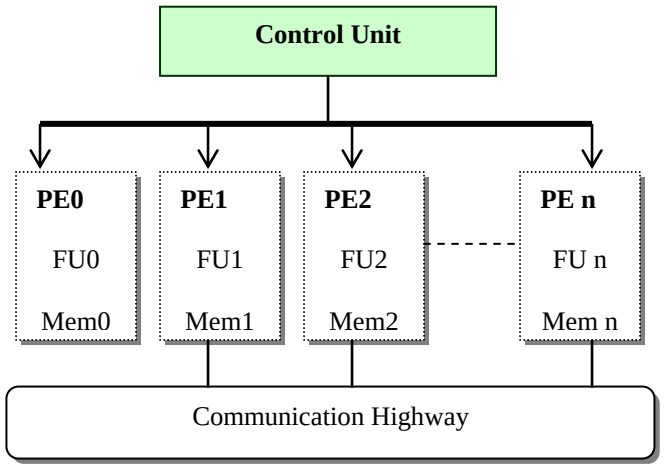

Figure 2.2. A Simplified View of SIMD Architecture

A. SIMD Architectures

Hardware parallelism is defined by the machine architecture and multiplicity and is often a function of cost and performance tradeoffs. One way to characterize the parallelism is a processor is by the number of instruction issue per machine cycle. If a processor issues k instructions per machine cycle, then it is called a k-issue processor. Conventional processor takes one or more machine cycles to issue a single instruction. These types of processors are called one-issue machines, with a single instruction on pipeline in the processor. Our target hardware plate form discussed in [14] can handle 5 instructions per one machine cycle.

Software parallelism is a function of algorithm, programming style and compiler optimization. The program flow graph displays the patterns of simultaneously executable operations and parallelism varies during the execution periods. Therefore for an optimal embedded system design a synergy of architecture and applications is very critical. A simplified model of our SIMD basline architecture is shown in Figure 2.2. Control unit steers the functional unit depending on the n-issue instructions. The operational model of an SIMD computer is specified by a 5-tuple: M= <N, C, I, M, R>, Where N is the number of processing elements (PE) in the machine E.g., in our case; our target platform has 27 PEs. . C is the set of instruction s directly executed by the control units including scalar and program flow control instructions., I is the set of instruction broadcast by the CU to all PEs, M is the set of masking schemes, and R is the set of data routing function. The architecture of SIMD machine can be define by specifying different parameters in M. Our target platform is discussed in [2]. The abbreviations uses in our results are:

load = number of load instructions, **store** = number of store instruction, **ialu** = number of integer operations, **falu** = number of floating point operations, **pbr** = number of exceptions handling instructions, **branch** = number of unconditional branch instructions, **cmpp** = number of conditional branch instructions.

Table 1. Bio Computation Applications Benchmark			
Application	**Pseudonym**	**Features**	**Algorithms**
GENESPLICER	A01	Detect splice sites in the genomic DNA	High accuracy and computationally efficient
TIGRSCAN	A02	DNA modeling	Generalized Hidden Markov Model (GHMM), HMM
TRANSTERMIS	A03	Rho-independent transcriptional terminators	Statistical estimation techniques
GENSCAN	A04	Predict complete gene structure	Search algorithms
MUMMER	A05	Genome Sequence alignment	Tree algorithms
GLIMMERHMM	A06	Find gene sequence in eukaryotes	IMM, Splice site models, Maximal dependence decomposition techniques
GENIE	A07	Gene finder in vertebrate and human DNA	GHMM, Neural Networks
FGENE	A08	Find splice sites, genes, promoters	Linear discriminant analysis
GRAIL	A09	Analysis of DNA sequence	Automated computation
GENEMARK	A10	Find genes in bacterial DNA sequence	Markov chains
NetPlaneGene	A11	Sequence analysis	Neural network
GLIMMER	A12	Coding regions in microbial DNA	Interpolated Markov Models (IMM)

IV. RESULTS AND DISCUSSION

The performance of the selected benchmark application is analyzed in terms of static and run time expression profiles. These profiles contain type of instructions, static affinity (sAffinity), maximum register mapping, spatial access, and number of operations per cycle (OPC), parallelism affinity and cache affinity. The affinity is defined with respect to the correlation between the applied software and underlying hardware architecture. Since affinity in a candidate application with a pre-defined genome data stream is determined probabilistically, there is always the chance that input data stream may not reflect the same response as the one under test. One way to enhance the architecture affinity is to reduce the application variability and/ or to increase the parallelism in software coding. However, it substantially sacrifices the uniqueness of bio applications and thus the portability. In addition, as the increase in parallelism, (paffinity), is increased for higher operations per cycle, the computational overhead and storage requirement in each application also increase. However, the first step in the workload characterization is to analyze them at the instruction level followed by measuring their unique response, which guarantees the optimal application affinity for underlying hardware architecture.

Figure 4.1 depicts the scatter plot matrix, to display the relationship between varieties of instruction types, common in bioinformatics applications. The plot matrix consists of plot cells containing little scatter plots formed from a pair of variables, (load, store), (load, cmpp), (load, branch), (load, pbr), (load, falu), (load, ialu). E.g., the parameters are represented by the horizontal axis and vertical axis of each plot cell. The measure values on the two parameters are represented by points in the little scatter plot. Each point represents the values for one application in two measurements. The scatter plot showing greatest density in the middle as (load, pbr), (cmpp, falu) are normally distributed variable, they can be identified roughly by their elliptical shape and small outliers. There exist five linear relations in Figure 4.1, (load, branch), (load, pbr), (load, ialu), (store, ialu), (branch, ialu), (falu, ialu).

Each application in the proximity of linear relations is concluded as having a negative or positive gradient along with other parameter. E.g., branch and 'ialu' has inverse relation ship, while store and 'ialu' has direct relationship. Such scattering plot can be used to identify potential relations between the measurements made on applications executed at same hardware platform.

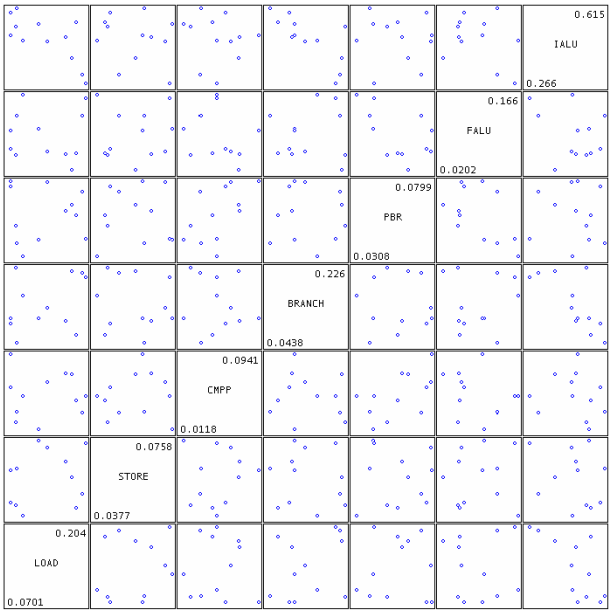

Figure 4.1, Scatter Plot Matrix for Static Application Expression Profile.

The dots displayed in Figure 4.2, are applications located vertically on the vertical statistical central tendency scale. Scale is made on the box plot, reflecting arithmetic mean, quartile, median and mode. The applications at the outliers are the carrying an unusual expression profile for our hardware platform. The load operations are dominating at only 1Q (first quartile) and 3Q (third quartile). Exception handling is

uniformly distributed in our set of bioinformatics applications but this is strictly a hardware feature, solely supported by the processor micro architecture. The integer operation is the most dominating operation and is easy to conclude form 'ialu' box plot shown in Figure 4.2. Apparently, interpolated Markov Model, maximization techniques involve higher integer operations, and are dominated in most of the bio applications. Conditional branch operations are dominating in bioinformatics applications, mostly they deal with search algorithm and classifications.

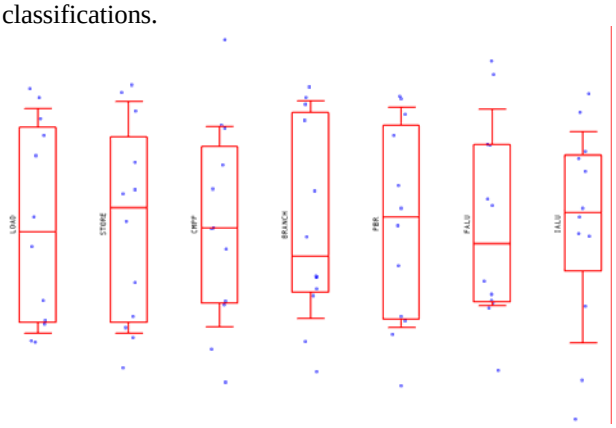

Figure 4.2, Central Tendency Plot for Static Application Expression Profile.

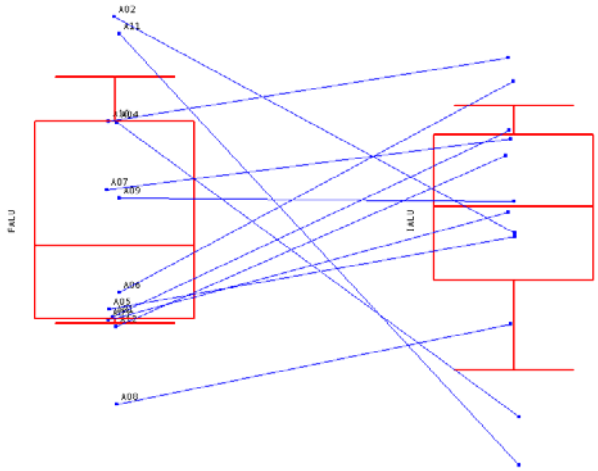

Figure 4.3, FALU and IALU Connected Plot for Benchmark Applications

The second compute intensity diagram is shown in Figure 4.3, where ALU operations are connected for integer and floating point operations in different bio applications marked at the 'falu' plot. The 'Tirgerscan' is known widely as highly computationally efficient, where being located at the top of 'falu' schematic, it appears to be computational intensive, while extending its connection to 'ialu', it lies in the mean value, it means it involved low integer operations. Similarly 'Netplanegene' has lowest integer operations but large number of floating point operations, eventually lead to most CPU consuming. Once again, it is important to note that laying at lowest 'falu', does not necessarily guarantee the minimum number of 'ialu'. The small number of floating operations and large integer operations eventually bring 'GlimmerHMM' and Mummer as the popular choice for gene sequence finding in eukaryotes and genomic sequence alignment. The effectiveness of Mummer is hidden in its implicitly implemented dependence decomposition techniques for improving splice site identification

We ran several test cases using our profiling framework [14], with the number of applications and different data bases, and architecture affinity as the base metric. The sAffinity in Figure 4.4, show the fact how much an application exploit the architectural benefits. 'Mummer' 19%, 'Genscan' 17%, 'FGene' 18%, 'NetPlaneGene' 17%, are showing high affinity to our target platform. 'Mummer' is used to identify gene sequence and is dominated with tree operations, while 'GenScan' predicts complete gene structure using search algorithms. The neural network is used in 'Fgene' and 'NetplaneGene' to analyze sequence genome sequence alignment. Both are suitable for our target platform.

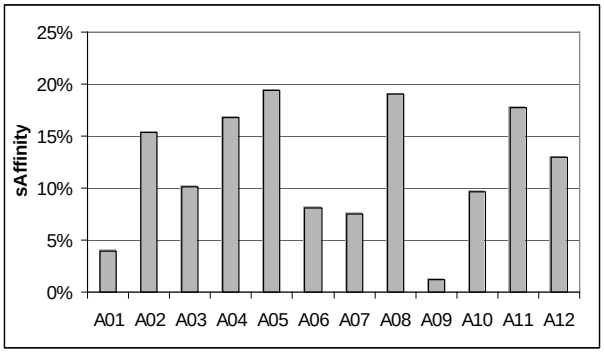

Figure 4.4, Relative Architectural Affinity for Benchmark Applications.

We optimize the codes for our target platform, Figure 4.5 show results for spatial access in improved code. We can see that 'GeneSplicer', 'Trasntermis', 'Genie', 'NetplanGene' are dropped in performance due to excessive branch oriented coding and large floating point operations.

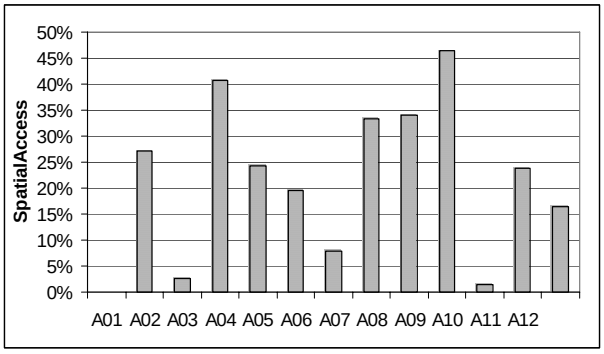

Figure 4.5, Percentage Spatial Access for SIMD Architecture.

In summary, the above results show that bio applications are better for SIMD architectures when they are dominated with integer operations and when operations are involved for changing the link trees between on and off is critical. However, the search algorithm and input database impact can not be avoided since it adds multiple exit edges and hence more computation.

V. CONCLUSIONS AND POTENTIAL TRENDS

In this work we emphasized on parameterized workload characterization-- a first step to design handheld bioinformatics embedded device. An application characterization at early stage opens a gateway to utilize off-the-shelf and state-of-the-art hardware architecture in efficient and reliable way.

We have also presented in detail static and runtime profile for popular bioinformatics applications. Multivariate statistical techniques are used to prune the potential relationship between the measure parameters. The experimental results demonstrated the importance of application-architecture correlation.

Our future plan includes extending the characterization to the optimization both in term of battery life and processor execution efficiency. We would like also to study approaches for modeling behavior of bio applications and the impact of databases organization techniques to design mobile bio computation devices.

ACKNOWLEDGMENT

This work is supported by ÖAD-Pakistan scholarship program initiated by Prof. Dr. Atta-ur-Rahman chairman HEC and Federal Minister Pakistan. The authors would like to thank Prof. Dr. Markus Rupp, Prof. Dr. Arpad Scholtz and Christian Doppler Laboratory at Institute of Communication and Radio-Frequency Engineering, Vienna University of Technology for their support and kind input during this work.

REFERENCES

[1] K. Albayraktaroglu, M. Franklin "Workload characterization of Modern Computational Biology Applications," Proc. of ICISIP-2005, pp. 435-440.

[2] Advance Computer Architecture, McGrawHill Publishing Company Limited, ISBN 0-07-053070-X

[3] N. Zafar Azeemi, "Probabilistic Iterative Compilation for Source Optimization of Embedded Programs," in the Proc. of IEEE International SoC Design Conference ISOCC 06, pages 323 - 328, Seoul, Korea, October, 2006.

[4] Y. Li, T. Li, , "Workload Characterization of Bioinformatics Applications," Proc. of the Int. Symp. on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems 2005, pp. 15-22, Sep 2005.

[5] The NCBI Bacteria genomes database ftp://ftp.ncbi.nih.gov/genomes/Bacteria/

[6] 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.

[7] NCBI, http://www.ncbi.nlm.nih.gov/

[8] www.bis.med.jhmi.edu

[9] www.gdb.org

[10] www.sequenceanalyses.org

[11] www.nar.oupjournal.org

[12] www.grail.lsd.ornl.gov

[13] BioSpace, http://www.biospace.com/

[14] N. Zafar Azeemi, "Power Aware Framework for Dense Matrix Operations in Multimedia Processors," Proc. of the 9th IEEE International Multi-topic Conference, Dec. 2005.

[15] The Apple Workgroup Cluster for Bioinformatics, http://images.apple.com/xserve/cluster/pdf/Workgroup_Cluster_PO_021104.pdf

[16] Genomeweb Daily News, http://www.genomeweb.com/

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