



# Intel<sup>®</sup> Technology Journal

Compute-Intensive, Highly Parallel Applications and Uses

## Performance Scalability of Data-Mining Workloads in Bioinformatics

# Performance Scalability of Data-Mining Workloads in Bioinformatics

Yurong Chen, Corporate Technology Group, Intel Corporation  
Qian Diao, Corporate Technology Group, Intel Corporation  
Carole Dulong, Corporate Technology Group, Intel Corporation  
Chunrong Lai, Corporate Technology Group, Intel Corporation  
Wei Hu, Corporate Technology Group, Intel Corporation  
Eric Li, Corporate Technology Group, Intel Corporation  
Wenlong Li, Corporate Technology Group, Intel Corporation  
Tao Wang, Corporate Technology Group, Intel Corporation  
Yimin Zhang, Corporate Technology Group, Intel Corporation

Index words: data mining, bioinformatics, performance scalability analysis

## ABSTRACT

Data mining is the extraction of hidden predictive information from large data bases. Emerging data-mining applications are important factors to drive the architecture of future microprocessors. This paper analyzes the performance scalability on parallel architectures of such applications to understand how to best architect the next generation of microprocessors that will have many CPU cores on chip.

Bioinformatics is one of the most active research areas in computer science, and it relies heavily on many types of data-mining techniques. In this paper, we report on the performance scalability analysis of six bioinformatics applications on a 16-way SMP based on Intel® Xeon™ microprocessor system. These applications are very compute intensive, and they manipulate very large data sets; many of them are freely accessible. Bioinformatics is a good proxy for workload analysis of general data-mining applications. Our experiments show that these applications exhibit good parallel behaviors after some algorithm-level reformulations, or careful parallelism selection. Most of them scale well with increased numbers of processors, with a speed-up of up to 14.4X on 16 processors.

We start with an introduction to data mining. The data-mining techniques studied are briefly described, and the selected workloads using these techniques are listed. We then provide a brief description of the methodology used for the studies. We present the scalability analysis of three workloads related to Bayesian Network (BN) structure, two workloads relevant to recognition, and one workload related to optimization. We conclude with the key lessons of the study. These workloads are compute intensive and data parallel. They manipulate large amounts of data that stress the cache hierarchy. Techniques optimizing the use of caches are key to ensure performance scalability of these workloads on parallel architectures.

## DATA MINING: A DEFINITION

Databases today can hold terabytes of data that hide a lot of information. *Data mining* is the technology that draws meaningful conclusions, extracts knowledge, and acquires models from these data.

The potential returns of data mining are large. Innovative organizations worldwide use it to locate and appeal to higher-value customers, to reconfigure their product offerings to increase sales, and to minimize losses due to error or fraud. Data mining has been widely used in various domains such as retail, telecommunication, medical diagnosis, and financial services.

---

® Intel and Xeon are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

## Bioinformatics Application Classification

Broadly speaking, bioinformatics is the recording, annotation, storage, analysis, and search/retrieval of nucleic acid sequences (genes and RNAs), protein sequences, and structural information. Currently, bioinformatics mainly includes databases of sequences and structural information, as well as methods to access, search, analyze, visualize, and retrieve the information. Bioinformatics applications can be categorized as follows:

- *Sequencing*: gene sequence assembly
- *Sequence alignment and search*: pair-wise and multiple sequence alignment, database search.
- *Sequence analysis*: gene finding, Single Nucleotide Polymorphisms (SNP) pattern analysis, etc.
- *Structure analysis and structural genomics*: protein secondary/tertiary structure prediction, protein folding.
- *Comparative genomics*: whole genome alignment, phylogenetic tree reconstruction.
- *Functional genomics/proteomics and system biology*: function prediction of non-coding sequences, gene expression clustering, and gene regulatory networks.
- *Clinical field* (gene expression classification, etc.)

Bioinformatics relies heavily on many types of data-mining techniques. For the purposes of our study, we describe several categories of data-mining techniques, and corresponding workloads.

## DATA-MINING TECHNIQUES STUDIED

Data mining uses a variety of data analysis tools to discover patterns and relationships in data that may be used to make valid predictions. It takes advantage of advances in the fields of Artificial Intelligence (AI) and statistics. Algorithms that are employed in many areas such as pattern recognition, machine learning, decision-making support, and statistical modeling can be used in

data mining. Following, we briefly introduce some of the techniques and algorithms.

## Bayesian Networks

A Bayesian Network (BN) is a probabilistic model that encodes probabilistic relationships between variables of interest. Over the last decade or so, BNs have been widely used in statistics, machine learning, pattern recognition, engineering, diagnostics, decision making, and so on.

Learning the structure of a BN from data is the most important task of BN applications [1]. The goal is to identify the statistic relationship between variables, and usually at the same time the conditional probability distribution of each variable can also be determined. BN structure learning has become an active research area in recent years [2, 3, 4].

The most popular approach to structure learning is to turn it into an optimization exercise. We first introduced a scoring function to evaluate the network with respect to the training data and to output a value that reflects how well the network scores, relative to the available data. We then search through possible network structures for the best scored network and take this as the network learned from the data. In general, the search problem is NP-hard [5]. Most algorithms use heuristic search methods, such as the Markov Chain Monte Carlo (MCMC) sampling [1, 6], K2 [1], simulated annealing [2], etc., of which the greedy hill-climbing algorithm is the most efficient and popular approach.

We have studied three applications using BNs: SNPs [7], GeneNet [8], and SEMPHY [10]. All are a variation on the hill-climbing concept. In SNPs and GeneNet applications, all the training data are observed (i.e., there are no missing data), and a standard hill-climbing search algorithm is employed. In a SEMPHY application, only a part of the data is observed (i.e., there are some missing data), and the hill-climbing learning procedure is used with an EM parameter-learning procedure. The total algorithm is called a Structural EM Algorithm [11, 12]. We look at these three applications next.

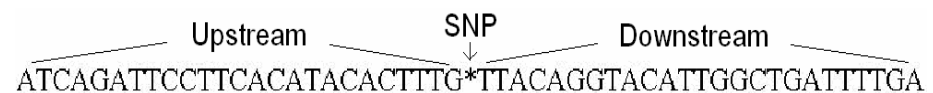


Figure 1(a): An instance of the SNP sequences (the symbol \* denotes the SNP site)

## Workloads Studied Using Bayesian Networks

SNPs are DNA sequence variations that occur when a single nucleotide (A, T, C, or G) is altered at certain loci in the genome sequence (shown in Figure 1(a)). These

variations are major sources of individual diversity. Understanding the importance of the recently identified SNPs in human genes has become a goal of human genetics [13]. A common understanding of the cause of SNPs is nucleotide substitution. A number of studies have shown that the substitution process can be context

dependent, that is, neighboring base composition can influence the substitution bias at a particular site. Substitution patterns at polymorphic sites and bias patterns in nucleotides neighboring polymorphic sites are important for understanding molecular mechanisms of mutation and genome evolution [14, 15].

This research suggests the existence of context dependencies near SNP sites. However, by employing BN structure learning, not only the dependencies around the SNPs loci can be confirmed, but also the dependencies model and influence strength for each loci neighboring the SNPs, can be discovered. The task can be formulated as follows: each locus on the sequence segment is represented as a discrete random variable of BN, with integer value ranges from 0 to 3 (each corresponds to A, C, G, or T), so the possible relations among these loci can be represented by the BN structure.

DNA microarray experiments measure all the genes of an organism, providing a “genomic” viewpoint on gene expression. Most of the analysis tools currently used are based on *clustering* algorithms. These algorithms attempt to locate groups of genes that have similar expression patterns over a set of experiments. A more ambitious goal for analysis is revealing the structure of the transcriptional regulation process. Thus, BN provides a natural approach to model the regulatory relationship between genes.

By representing each gene as a variable of the BN, the gene expression data analysis problem can be formulated as a BN structure-learning problem. The GeneNet application uses the same serial hill-climbing algorithm as the SNPs problem, but its input set has different characteristics: for SNPs, the BN contains only tens of variables (<100), but a large number of training data (typically 50K – 500K), while in GeneNet, there are

many variables (1K – 10K typically), but only hundreds of training cases.

Unlike the previous applications on BN structure learning, the SEMPHY application uses the Structural Expectation Maximization (SEM) algorithm. SEMPHY differs from traditional BN applications in two aspects: it searches from a bifurcating tree space rather than the DAG space; and it can find the optimal solution based on missing data.

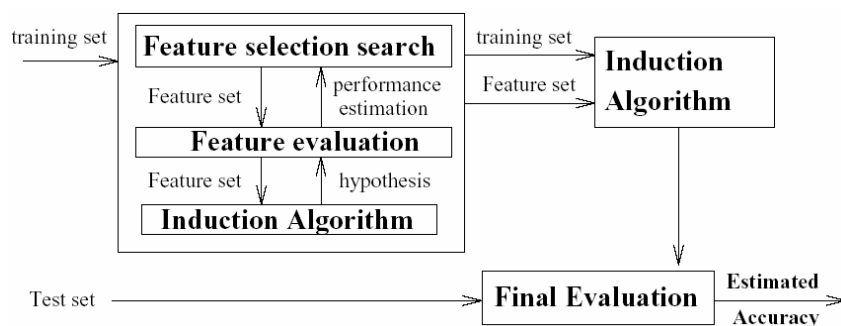
## Classification and Prediction

To classify an object is to put it into a pre-defined class or category, or to assign it a label. Prediction can be viewed as the construction and use of a model to classify an unlabeled sample. Classification and prediction have numerous applications including credit approval, medical diagnosis, performance prediction, and selective marketing.

For example, the Support Vector Machine (SVM) has been considered a state-of-the-art classification technique since the 1990s, and we have used it in disease gene finding, based on the Support Vector Machines Recursive Feature Elimination (SVM-RFE) method.

## Workloads Studied Using Classification Techniques

SVM-RFE [17] is a feature selection method to refine the optimum feature set by using SVM in a wrapper approach (shown in Figure 1(b)). It selects or omits dimensions of the data, depending on a performance measurement of the SVM classifier. It is much more robust to data overfitting than other methods, including combinatorial search. (In SVM-RFE, the induction algorithm used is SVM.)



**Figure 1(b): Overview of wrapper method for feature selection**

In bioinformatics, SVM-RFE has been used for the task of microarray data analysis, particularly in disease gene finding. It eliminates gene redundancy automatically and yields better and more compact gene subsets. The

selection is obtained by a recursive feature elimination process: at each RFE step, a gene is discarded from the active variables of an SVM classification model. The features are eliminated according to a criterion related to

their support for the discrimination function, and the SVM is re-trained at each step.

We now describe the second workload studied in this report using a classification technique: the Cocke-Younger-Kasami (CYK) algorithm. This technique uses a basic parsing algorithm for any context-free language, and it is used in RSEARCH during an RNA secondary structure homolog search. RSEARCH [13] uses Stochastic Context-Free Grammar (SCFG) to take a single RNA sequence with its secondary structure, and it utilizes the CYK algorithm to search a database for homologous RNAs through local alignment. RSEARCH has better performance in accuracy for RNA homolog search than other sequence search programs, such as BLAST and SSEARCH, and it is also capable of finding significant remote RNA structure homologies.

SCFG and its decoding algorithm CYK used in RSEARCH can also be applied in other areas, such as language modeling for speech recognition [14], language parsing for natural language processing [15], multitasked activities recognition for computer vision [16], and so on.

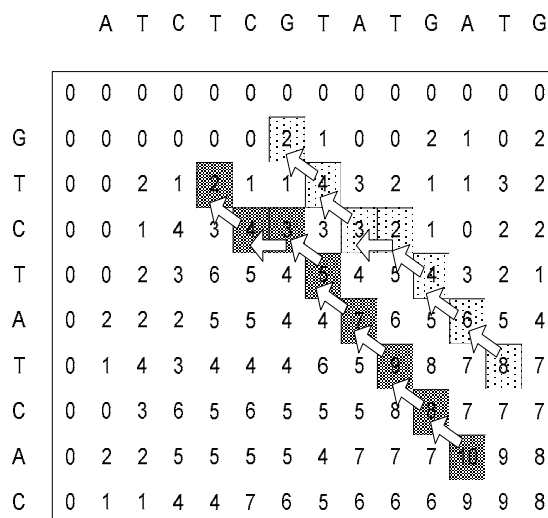
## Optimization

Dynamic Programming (DP) is an approach developed to solve sequential, or multi-stage, decision problems. This approach is also applicable to decision problems where sequential property is induced solely for computational convenience. DP is widely used in combinatorial optimization, speech recognition, sequence alignment, time series data processing, etc. Even when it does not solve a problem completely, it can be useful as part of an overall approach. In particular, DP plays an important role in solving similarity problems of some major data-mining tasks, such as Association Rule Mining (ARM), similar time sequence mining, similar image mining, and so on. ARM is a matter of looking for association rules in data. An association rule is an expression  $X \text{ IMPLIES } Y$ , where  $X$  and  $Y$  are sets of items. The intuitive meaning of such a rule is that transactions of the database that contain  $X$  tend to contain  $Y$ .

## Workloads Studied Using Optimization Techniques

Sequence alignment is an important tool in bioinformatics, text, acoustic signal, and image processing. It is capable of identifying the similar and diverged regions between two sequences, e.g., biological DNA/protein sequences or text strings. From a biological point of view, matches may turn out to be similar functions, e.g., homology pairs and conserved regions, while mismatches may detect functional differences, e.g., SNP.

With DP, Needleman and Wunsch presented the first global alignment algorithm in 1970 [18]. Smith and Waterman improved this algorithm for the local alignment to find the longest common substring [19] (shown in Figure 2). In this paper, we study an efficient Parallel Linear Space Alignment (PLSA) for large-scale sequence alignment. By introducing the novel grid cache, global/local start points, the algorithm reduces the re-computations of the trace-back period dramatically, and it provides more parallelism than other methods do. Besides the algorithms mentioned above, there are many other techniques or algorithms that have been widely used in data mining, such as clustering, statistic modeling, association rules mining, Naïve Bayes classifiers, neural networks, memory-based reasoning, evolutionary programming, regression, decision trees, etc.



Similarity matrix of the alignment of sequence ATCTCGTATGATG and GTCTATCA.  $k=2$ , substitution cost of +2 if the two characters are identical and -1 otherwise. Both the first and extension gap penalty are -1. Two traceback paths deliver the non-intersecting optimal and near-optimal local alignments.

**Figure 2: Smith Waterman sequence alignment**

## SUMMARY OF APPLICATIONS STUDIED

Table 1 summarizes the six applications studied in this paper. We list the type of algorithm they use, how the parallelism can be exploited, and we show what types of applications they are representative of.

We have used applications developed in universities. For example, we have used SEMPHY (from the Hebrew University in Jerusalem) for reconstruction of phylogenetic trees; and RSEARCH (from Washington University in St. Louis) for RNA secondary structure homolog search. We have also developed some workloads. For instance, we have used a BN structured learning to solve the discovering of patterns in SNPs, and to analyze gene expression data in DNA microarrays in GeneNet. In PLSA, we have developed a novel large-scale alignment algorithm for the whole genome alignment.

This section describes the methodology used to parallelize, optimize, and analyze the workloads. The first step is to profile the workloads to identify the hot spots. The Intel® VTune™ Performance Analyzer was used for function-level profiling, and for correlation of the hardware performance events with the source code.

Parallelization was done with the Open MP programming model that is well suited to exploit the data parallelism of these algorithms. For example, in RSEARCH, the whole RNA sequence database is scanned with a three-dimensional dynamic programming algorithm.

Generally, the query RNA sequence is far shorter than the sequences in the database. RSEARCH first defines a value “D\_scale,” representing the largest ratio between the match part of the sequence and the query sequence. The database sequence is segmented into different subsets that overlap so that the D\_scale can be a multiplier of the query length. The search through the different subsets can be done in parallel by different processors.

The costs of synchronization, locks, and barriers were measured using the Intel VTune Performance Analyzer thread profiler for OMP applications. The experiments show that for most studied applications these costs are very low, which is expected for data-parallel applications with very little synchronization between threads.

After having characterized the communication overhead, synchronizations (explicit or implicit), and load-balancing performance (dynamic or static partitioning methods), we measured the performance increase on up to 16 processors, and we characterized the memory hierarchy behavior.

® Intel and VTune are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

**Table 1: Bioinformatics workloads, algorithms, applicability Workload Analysis Methodology**

Category	Workload	Algorithm	Parallelism	Applicability
Bayesian Network/Structure Learning	SNPs (Single Nucleotide Polymorphisms)	Structure learning Hill-climbing	Data parallelism	<ul style="list-style-type: none"> <li>• Pattern recognition</li> <li>• Speech recognition</li> <li>• Optimization</li> <li>• Text mining</li> <li>• Game</li> <li>• Decision Making</li> </ul>
	GeneNet (Gene Expression analysis in microarrays)		(instance data, node, tree)	
	SEMPHY	Structural EM		
Classification and Prediction	RSEARCH (homologous RNA sequence)	Stochastic Context Free Grammar: CYK Local alignment	Data base Segmentation	<ul style="list-style-type: none"> <li>• Recognition</li> <li>• Classification</li> <li>• Prediction</li> <li>• Speech recognition, language parsing</li> </ul>
	SVM-RFE (Disease Gene Finding in Microarrays)	SVM based feature selection	Data blocking matrix/vector multiply	<ul style="list-style-type: none"> <li>• Pattern recognition</li> <li>• Classification</li> <li>• optimization</li> </ul>
Optimization	PLSA (Parallel Linear Space Alignment)	Dynamic Programming	Data blocking Wave-front parallelism	<ul style="list-style-type: none"> <li>• Pattern recognition,</li> <li>• Text mining</li> <li>• Association Rule Mining</li> <li>• Combinatorial optimization</li> </ul>

To measure the performance of our workloads, we use a 16-way SMP based on Intel Xeon microprocessor system interconnected with a crossbar. The configuration of the 16-way system is described in Table 2. The machine is running the SUSE ES Linux\* operating system environment for all the experiments. All applications were compiled with the Intel Compiler v8.0, at the highest level of optimization.

## PERFORMANCE SCALABILITY ANALYSIS

All the workloads studied in this paper use data parallelism, where all processors are executing the same code on different data. Figure 3 shows most of the workloads scale well with increased numbers of processors, and two workloads exhibit linear speed-up performance (SEMPHY, PLSA).

\* Other brands and names are the property of their respective owners.

**Table 2: Configuration of the 16-way SMP based on Intel® Xeon™ microprocessor system**

Processor Speed	3.0 GHz
L1 Data Cache	8 KB, hit latency: 2 cycles
L2 Unified Cache	512 KB, hit latency: ~10 cycles
L3 Unified Cache	4 MB, hit latency: 30+ cycles
L4 on-board Unified Cache	32 MB, hit latency: 300+ cycles
Interconnection	Crossbar
System Bus Speed	400 MHz
Front Side Bus Bandwidth	3.2 GB/s
Memory Size	8 GB, dual-channel DDR 400

To understand the performance-limiting factors, we have quantified the parallelism overhead such as synchronizations penalties, load imbalance, and sequential sections. They are not significant, especially for large data sets typical in current, and future, data-mining workloads. Figure 4 shows these metrics for two selected workloads, where the sequential area and the load imbalance penalties are diminishing when the data set size increases. These results are typical of all the workloads studied in this report. Very few synchronizations are needed between threads, and load balancing between threads is not an issue. In SEMPHY, for example, computations are distributed in four kernels that are nested loops, with no dependency between loop iterations. The basic “Parallel For” pragma is used to parallelize these loops. In one of the kernels, the data decomposition is constrained by the relation between the number of leaves in the tree, and the length of the DNA or protein sequence. This creates a small load imbalance for small data sets. The balancing issue goes away for large data sets, where the constraints become insignificant.

To identify the performance bottlenecks, we characterize the memory hierarchy behavior by measuring the cache miss rates and the Front Side Bus (FSB) bandwidth. In Figure 5, it is interesting to see that the L2 cache miss rates vary very little with the number of processors. The data sets are large enough not to fit in L2, even when problems are divided among 16 processors. We can

observe SVM-RFE has very high L2 and L3 cache miss rates. The function profile of SVM-RFE shows that the SVM training is the most time-consuming kernel. It consists of a large number of vector-vector multiplications. We have used the Intel Math Kernel Library (MKL) for these operations to take advantage of its highly optimized routines. But there is no data reuse in vector-vector multiply operations, and this explains the high cache miss rates. With increasing numbers of processors, the shared system bus sees high memory traffic, resulting in high memory latencies. This limits the speed-up performance for the SVM-RFE workload. In the case of SNP, and GeneNet, the L3 miss rates are modest, but they increase with the number of processors, when there are more than four processors. These applications have at least one large data structure shared between the thread. This data structure is shared efficiently between four processors in the L4 cache of the 16-way system. But sharing is done through the main memory interconnect for more than four processors, and this limits the scalability of these workloads.

Figure 6 shows the cache miss per instruction in L2 and L3 and the FSB bandwidth used by these workloads running a single thread.

We have verified that these miss per instructions in L2 and L3 remain about constant per workload as the number of threads increases from 1 to 16.



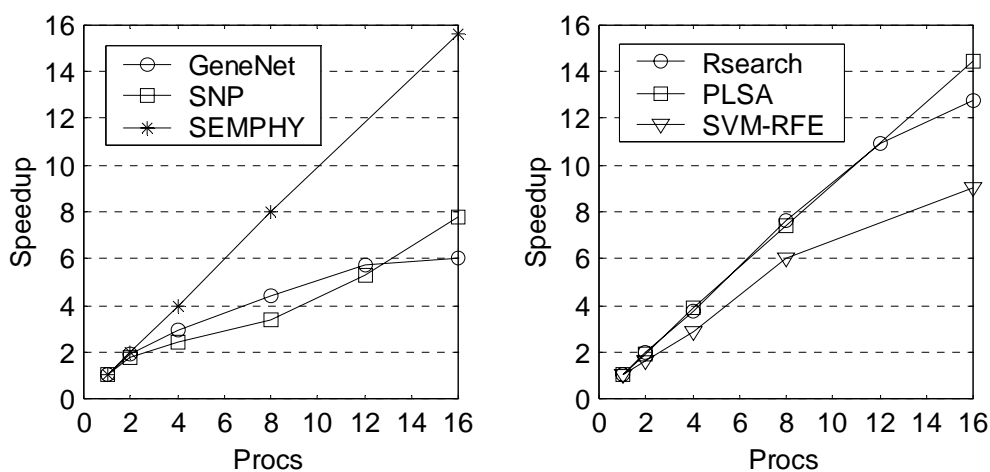


Figure 3: Performance speed-up as a function of the number of processors

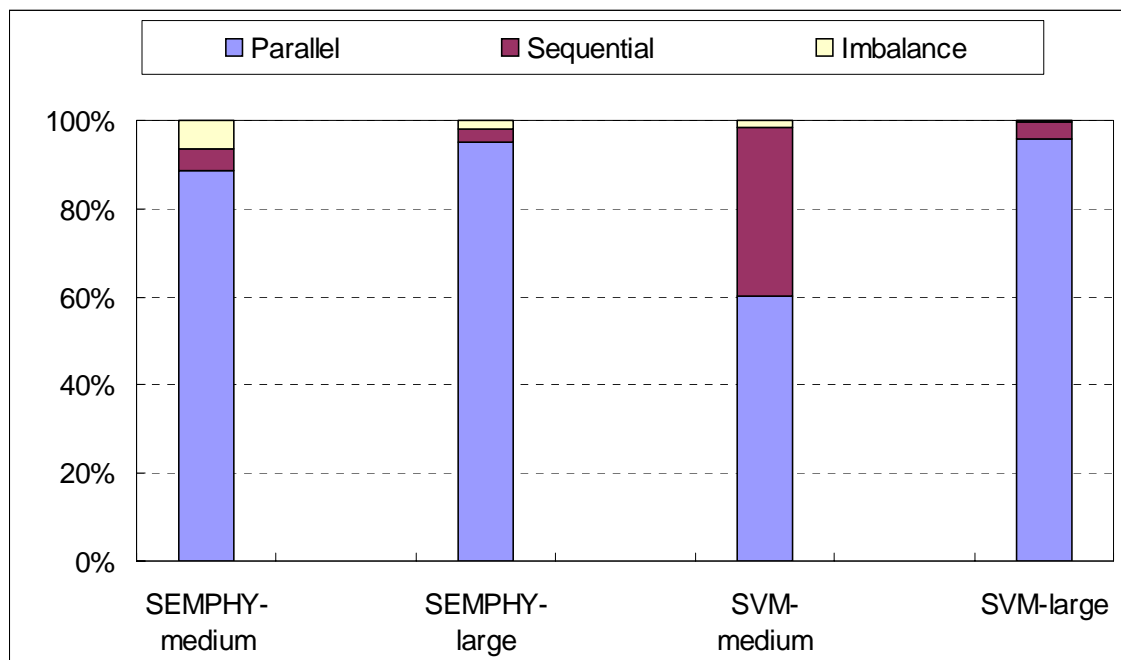


Figure 4: Distribution of time spent in parallel, and sequential, code

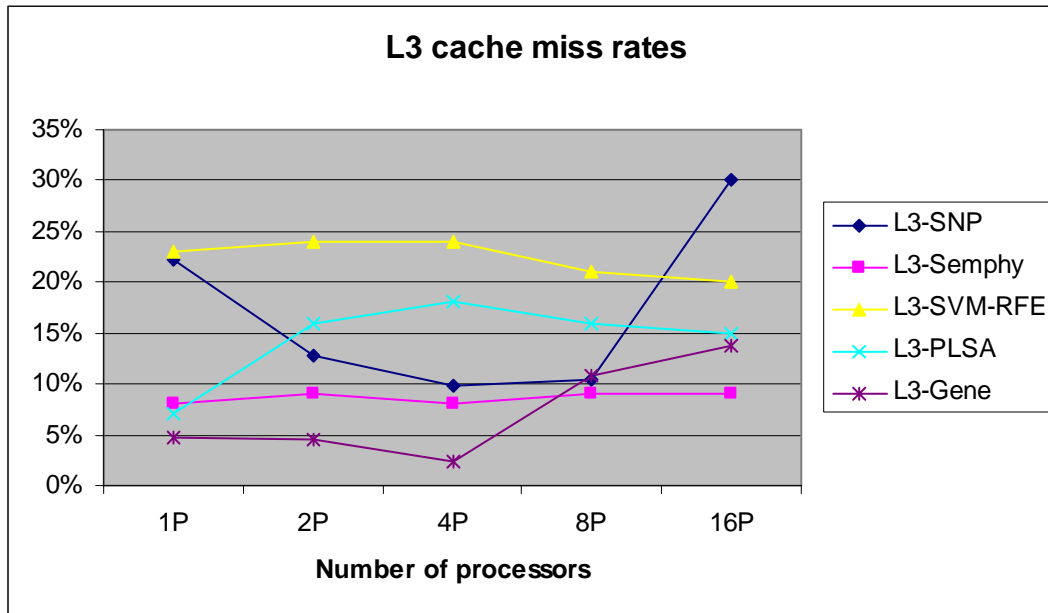


Figure 5: L2/L3 cache miss rates as a function of the number of processors

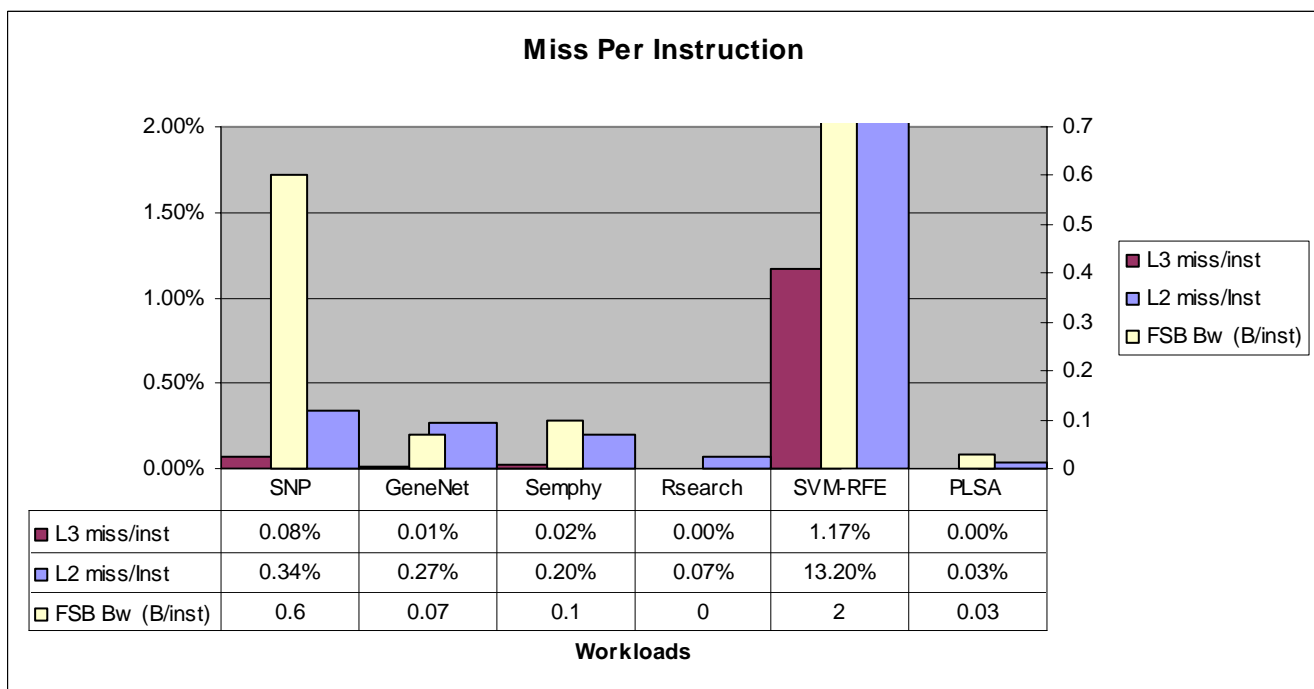


Figure 6: L2 and L3 miss per instruction for the one processor case

Most of these workloads are integer workloads, and they use very few floating-point operations. So the bandwidth per instruction shown in Figure 6 is in byte-per-integer operation. The three workloads that have the lowest L3 miss per instruction are the ones whose performance scales almost linearly with the number of processors: SEMPHY, RSEARCH, and PLSA. The high miss per

instruction in both L2, and L3 in SVM-RFE, has already been explained by the vector-vector multiply operations used most of the time in SVM-RFE, and the fact that there is no data reuse. Note that the Intel Xeon processor prefetchers are highly effective for such operations working on contiguous data, and they explain why the L3 miss per instruction is not higher than it is. For SNP and

GeneNet, the cache misses per instruction are higher than for the workloads scaling very well, but much lower than for SVM-RFE. Yet these two workloads do not scale as well as SVM-RFE. Caches are not the limiting factors for these workloads. Threads spend a significant part of the time waiting for each other at the end of parallel sections, because of load balancing. In SNP for example, parallel threads perform the hill-climbing algorithm on different input data. The number of computations depends on the data structure, and the difference in computation requirements between threads explains the load imbalance.

Bus bandwidth utilization varies widely between these six workloads. They go all the way from virtually 0 for

RSEARCH that is highly compute intensive, with very high reuse of data in cache, to 2 Bytes per instruction for SVM-RFE whose main computation is a vector-vector multiply that has very little data reuse.

Figure 7 shows how the bus bandwidth utilization varies with the number of processors. It uses a log scale on both axis. The bandwidth used by the different workloads varies widely between workloads, which is explained by the very different miss rates the workloads get in the L2 and L3 caches. But for all workloads, the FSB bandwidth varies linearly with the number of processors. This indicates that the bus bandwidth is not a limiting factor on this system.

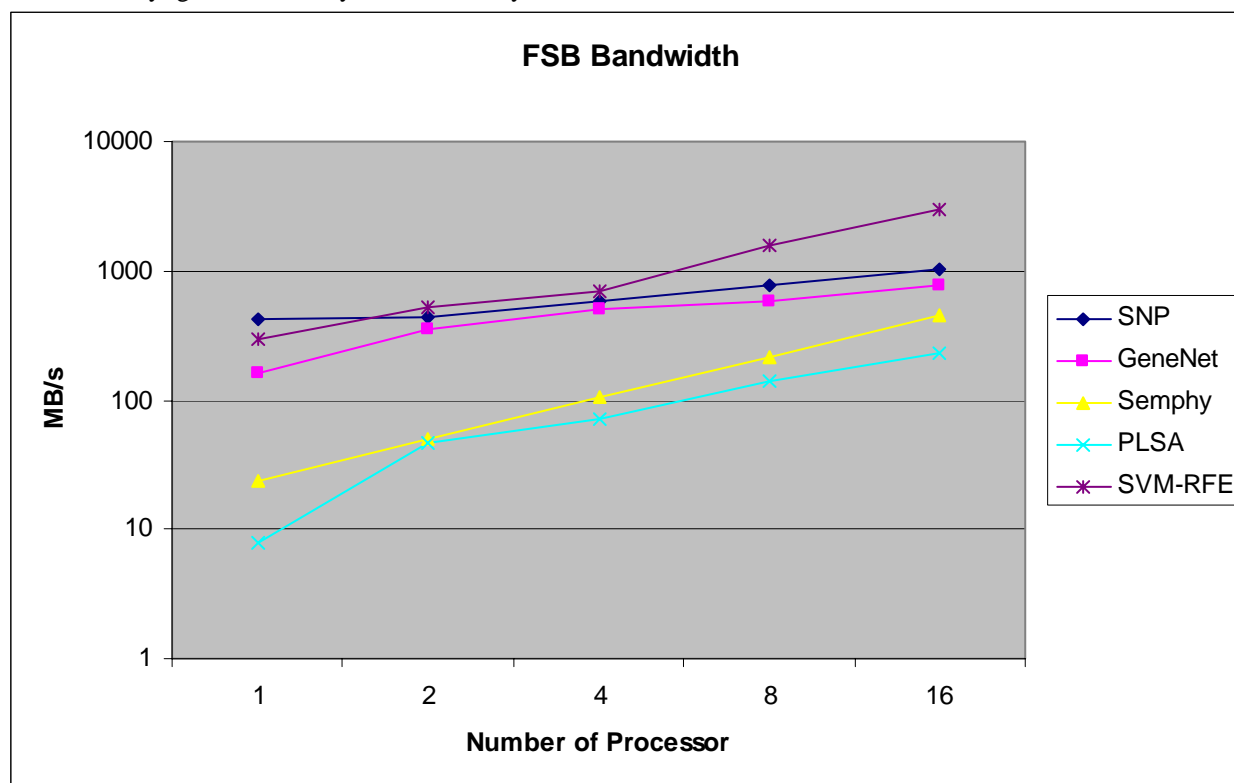


Figure 7: FSB bandwidth as a function of the number of processors

## CONCLUSION

The bioinformatics workloads studied in this paper are representative of general-purpose data-mining techniques. They use BNs, SVMs, and DP, among other methods. They are data parallel workloads with large data sets and are very compute intensive.

Performance scalability up to 16 processors is very good for some workloads such as SEMPHY, RSEARCH, and PLSA that exhibit almost linear speed-up. It is not as

good, but quite respectable, for SNP, GeneNet, and SVM-RFE.

The FSB bandwidth is not a limiting factor for performance scalability (for the system used in the study). The FSB utilization grows linearly with the number of active processors for all workloads. Caches are too small, which leads to the high miss rates of 5% to 30% in the L3 cache. The large latencies to the L4 cache and to main memory DRAM are limiting scaling for workloads like SVM-RFE and SNP that access large and complex data structures.

## ACKNOWLEDGMENTS

We acknowledge the encouragement and help that we have received from Bob Liang. Additional thanks go out to individuals who have reviewed the paper and provided valuable feedback: Chu-cheow Lim, Gideon Gerzon, and Chuck Yount.

## REFERENCES

- [1] Kevin Murphy, "The Bayes Net Toolbox for Matlab," Computing Science and Statistics, vol. 33, 2001.
- [2] D. Heckerman et al., "Learning Bayesian networks: the combination of knowledge and statistical data," Technical Report MSR-TR-09-09, Microsoft Research, 1994.
- [3] D. Heckerman, "A Tutorial on Learning with Bayesian Networks," in Learning in Graphical Models, M. Jordan, ed. MIT Press, Cambridge, MA, 1999.
- [4] W. Buntine, "A guide to the literature on learning probabilistic networks from data," IEEE Trans. On Knowledge and Data Engineering, 8:195-210, 1996.
- [5] Chickering, D. M., "Learning Bayesian networks is NP-Complete," in D. Fisher and H. Lenz (Eds.), Learning from data: Artificial intelligence and statistics v, pp. 121-130.
- [6] P. Guidici et al., "Markov Chain Monte Carlo methods for probabilistic network model determination," Journal of the Italian Statistical Society, 7, pp. 171-183.
- [7] X. Ma et al., "Discovering Possible Context Dependencies around SNP Sites in Human Genes with Bayesian Network Learning," Eighth International Conference on Control, Automation, Robotics and Vision (ICARCV), 2004.
- [8] N. Friedman et al., "Using Bayesian Networks to Analyze Expression Data," Journal of Computational Biology, 7:601-620, 2000.
- [9] Bateman, A. et al., "The Pfam Protein Families Database," Nucleic Acids Research Database, Issue 32:D138-D141. 2004.
- [10] N. Friedman, "A Structural EM algorithm for Phylogenetic Inference," Journal of Computational Biology, 9:331-353, 2002.
- [11] N. Friedman, "Learning belief networks in the presence of missing values and hidden variables," Fisher, D. ed., Proceedings of the Fourteenth International Conference on Machine Learning, pp. 125-133, Morgan Kaufman, San Francisco, (1997).
- [12] "The Bayesian structure EM algorithm," in Fourteenth Conf. on Uncertainty in Artificial Intelligence (UAI), 1998.
- [13] Klein, R.J. and Eddy, S.R., "RSEARCH: Finding homologs of single structured RNA sequences," BMC Bioinformatics, 2003, 4:44.
- [14] Jurafsky, D., Wooters, C., Segal, J., Stolcke, A., Fosler, E., Tajchman, G., Morgan, N., "Using a Stochastic Context-Free Grammar as a Language Model for Speech Recognition," in Proc. ICASSP'95, 189-192.
- [15] Fujisaki, T.; Jelinek, F.; Cocke, J.; Black, E.; and Nishino, T., "A probabilistic parsing method for sentence disambiguation," in Current Issues in Parsing Technology, edited by Masaru Tomita, Kluwer Academic Publishers, 1991, pp. 139-152.
- [16] Darnell Moore, Irfan Essa, "Recognizing Multitasked Activities using Stochastic Context-Free Grammar," in Proceedings of Workshop on Models versus Exemplars in Computer Vision," held in Conjunction with IEEE CVPR 2001, Kauai, Hawaii, December 2001.
- [17] E. Osuna, R. Freund, and F. Girosi, "Training support vector machines: An application to face detection," in Proceedings of CVPR'97, pages 130-136, New York, NY, 1997b.
- [18] Saul B. Needleman and Christian D. Wunsch, "A General Method Applicable to the Search for Similarities in the amino acid Sequence of Two Sequences," Journal of Molecular Biology, 48:443-453, 1970.
- [19] Temple F. Smith and Michael S. Waterman, "Identification of Common Molecular Subsequences," Journal of Molecular Biology, 147:195-197, 1981.

## AUTHORS' BIOGRAPHIES

**Yurong Chen** is a researcher at Microprocessor Technology Lab, Beijing. Currently he conducts research on emerging computing paradigms, parallel algorithms, and scalable workload development and analysis. He joined Intel in 2004. Before that he spent two years doing postdoctoral research work in computer science in the Institute of Software, Chinese Academy of Sciences. He received his BS degree in Applied Mathematics in 1998 and MS and PhD degrees in computational mathematics in 2002, all from Tsinghua University, China. His e-mail is yurong.chen at intel.com.

**Qian Diao** is a researcher in the Microprocessor Technology Lab, Beijing. Currently she works on various

data-mining techniques. In Intel, she has been involved in several projects related to information retrieval, speech recognition, visual tracking, bioinformatics, and statistical computing. She got her Ph.D. from Shanghai Jiao Tong University in 2000 and joined Intel in 2000. Her e-mail is qian.diao at intel.com.

**Carole Dulong** is a senior researcher and Computer Architect in the Microprocessor Technology Lab. She leads a team of researchers working on various data-mining techniques. She joined Intel in 1990. She was a member of the IPF architecture definition team and contributed to the IPF compiler design. She graduated from Institut Supérieur d'Electronique de Paris (France). Her e-mail is carole.dulong at intel.com.

**Wei Hu** is a researcher in the Microprocessor Technology Lab, Beijing. Currently he works on various data-mining techniques. At Intel, he has been involved in several projects related to natural language processing and speech recognition, and statistic computing. He got his Ph.D from the Institute of Computing Technology, China Academy of Sciences (CAS/ICT) in 1998, and he joined Intel in 2000. His e-mail is wei.hu at intel.com.

**Chunrong Lai** is a researcher in the Microprocessor Technology Lab, Beijing. He is currently working on data-mining workload scalability and related architecture research within the Scalable Statistical Computing Group. His projects in Intel include various applications analysis, parallelization and optimization, performance simulation, and compiler re-target. He received his M.S. degree from the Chinese Academy of Sciences in 2000 and joined Intel as his first job. His e-mail is chunrong.lai at intel.com.

**Eric Li** is a researcher in the Microprocessor Technology Lab, Beijing. Currently he is working on algorithmic and workload analysis on data-mining applications. Prior to this position, he worked on the workload optimization, parallelization, analysis, and related algorithm methodology development. He received his M.S. degree from Tsinghua University in 2002, and joined Intel that same year. His e-mail is eric.q.li at intel.com.

**Wenlong Li** is a researcher in the Microprocessor Technology Lab, Beijing. Currently he is working on algorithmic and workload analysis on data-mining applications. Before this, he did research in loop compilation techniques for IPF architecture. He received his Ph.D degree from Tsinghua University in 2005 and joined Intel that same year. His e-mail is wenlong.li at intel.com.

**Tao Wang** is a researcher in the Microprocessor Technology Lab, Beijing. Currently he conducts research on data mining, machine learning, and computer vision

techniques. At Intel, he has been involved in several projects related to visual tracking, bioinformatics, and content-based image/video retrieval. He received his Ph.D. degree from Tsinghua University in 2003 and joined Intel that same year. His e-mail is tao.wang at intel.com.

**Yimin Zhang** is a researcher in the Microprocessor Technology Lab, Beijing. He leads a team of researchers working on various statistical computing techniques and their scalability analysis. He joined Intel in 2000. At Intel, he has been involved in several projects related to natural language processing and speech recognition, especially focusing on Chinese-named entity extraction and DBN-based speech recognition. He received his B.A. degree from Fudan University in 1993, his M.S. degree from Shanghai Maritime University in 1996, and his Ph.D. degree from Shanghai Jiao Tong University in 1999, all in Computer Science. His e-mail is yimin.zhang at intel.com.

Copyright © Intel Corporation 2005. This publication was downloaded from <http://developer.intel.com/>.

Legal notices at <http://www.intel.com/sites/corporate/tradmarx.htm>.

For further information visit:

[developer.intel.com/technology/itj/index.htm](http://developer.intel.com/technology/itj/index.htm)