2011 14th Euromicro Conference on Digital System Design

HMMER Performance Model for Multicore Architectures

Sebastian Isaza, Ernst Houtgast, Georgi Gaydadjiev
Computer Engineering Laboratory
Delft University of Technology
Delft, The Netherlands
Contact author's email: s.isazaramirez@tudelft.nl

***Abstract*—Exponential growth in biological sequence data combined with the computationally intensive nature of bioinformatics applications results in a continuously rising demand for processing power. In this paper, we propose a performance model that captures the behavior and performance scalability of HMMER, a bioinformatics application that identifies similarities between protein sequences and a protein family model. With our analytical model, the optimal master-worker ratio for a user scenario can be estimated. The model is evaluated and is found accurate with less than 2% error. We applied our model to a widely used heterogeneous multicore, the Cell BE, using the PPE and SPEs as master and workers respectively. Experimental results show that for the current parallelization strategy, the I/O speed at which the database is read from disk and the inputs pre-processing are the two most limiting factors in the Cell BE case.**

***Keywords*-performance model; HMMER; multicore architectures; bioinformatics;**

I. Introduction

The rapid development of genetics in recent decades has led to an explosion of genetic information data. The genetic structure of many species has been sequenced and the resulting sheer size of such data sets makes analysis by hand impossible. In bioinformatics, computers are used to enable biological research directions that would be unfeasible otherwise.

Within bioinformatics, sequence alignment is a primary activity. Fragments of DNA or protein sequences are compared to each other in order to identify similarities between them. Due to the computational complexity of the algorithms used to process these data sets, demand for processing power is soaring. Therefore, it is critical for bioinformatics applications to be efficient and scalable in order to meet this demand. Two popular sequence analysis tools are BLAST [1] and HMMER [2]. Each has its own merits: BLAST is faster; HMMER is more sensitive and also able to find more distant relationships. The adoption of HMMER2 in the SPEC2006 and the recent HMMER3 developments show its significance.

Advancements in microprocessor technology in the past have resulted in steadily increasing computational power, through miniaturization and growing transistor budgets. However, single threaded performance improvement is stagnating because of frequency, power and memory scaling barriers. These "walls" are the reason for the current paradigm shift towards multicore architectures, in an attempt to deliver the expected performance growth. One example of such multicore architecture is Cell BE, a processor with special architectural components and organization that has opened a new path in processor design. In this paper we have used the Cell BE as a case study to validate our proposal, however, we consider that our analysis is applicable to other multicore architectures as well.

The effectiveness of the multicore paradigm for bioinformatics applications is still an open research question. In this paper, we propose an analytical model of a HMMER *master-worker* parallelization for multicore architectures and investigate its scalability behavior. Through profiling on real hardware we determine the coefficients and validate our model for the Cell BE case. The main contributions of this paper are the in-depth analysis of the HMMER parallelization and the accurate performance prediction model that is shown to be precise with error less than 2%.

The remainder of the paper is organized as follows. Section II describes the related work. Section III introduces the Cell BE processor and HMMERCELL. In Section IV we describe the methodology used and Section V presents the model. Finally, in Section VI we draw the conclusions.

II. Related Work

HMMERCELL, the Cell BE port of HMMER, is created by Lu *et al.* In [3], detailed information on the implementation and parallelization strategy is provided, along with raw performance data where it is benchmarked against commodity x86 architectures. Compared to the AMD Opteron platform (2.8 GHz, 1-4 cores) and the Intel Woodcrest platform (3.0 GHz, 1-4 cores), a single Cell BE is reported to be up to thirty times faster than a single core Intel or AMD processor. In contrast, in this paper we build a model of HMMER for a master-worker parallelization scheme and use HMMERCELL as a validation example. Besides, we evaluate HMMERCELL performance in much more detail by breaking down performance into three constituent phases. These are then modeled and profiled in order to analyze their behavior for various workloads. Finally, bottlenecks to scalability are discussed.

HMMER has been ported to various architectures. In [4], an FPGA implementation of HMMER is investigated. As in HMMERCELL, the computationally intensive kernel of the Viterbi algorithm is the main focus. Similar to HMMERCELL, the FPGA is used as a filter: the sequences with a promising score require reprocessing on the host machine. A thirty fold speedup over an AMD Athlon64 3500+ is reported. This result is comparable to the performance of HMMERCELL.

MPI-HMMER was created to take advantage of computer clusters [5]. Similar to HMMERCELL, one node is assigned a manager role and the rest of the machines are workers over which the workload is distributed. To cope with overhead from message passing, sequences are grouped in larger bundles and sent as one message. Through double buffering, communication latency is minimized. An eleven-fold speedup is reported when using sixteen machines. In [6], MPI-HMMER is analyzed and found to be scalable up to 32-64 nodes, depending on workload. PIO-HMMER is introduced, addressing I/O-related bottlenecks through the use of parallel I/O and optimized post-processing. The manager distributes an offset file with sequences to each node, worker nodes read the sequences from their local database. Furthermore, nodes only report significant results back to the manager. The resulting scaling capability is much improved, as up to 256 machines can be used effectively. Other authors have parallelized HMMER *hmmpfam* kernel for shared-memory machines [7] and for computer clusters in HSP-HMMER[8], using MPI. Although our proposed model could also be used for the mentioned HMMER versions, this paper only verifies the model against the HMMERCELL implementation. The reason why HMMERCELL scales to less cores than other implementations [5], [6] is because of the higher per-core Viterbi performance brought by the Cell's SPEs.

III. Background

In this section we briefly present the platform used in our case study, i.e., the Cell BE. Besides, we introduce HMMER along with its parallel implementation.

A. The Cell Broadband Engine

The Cell Broadband Engine [9] represents a radical departure from traditional microprocessors design. The Cell BE is a heterogeneous architecture with 9 computing cores: the Power Processing Unit (PPE), used for general purpose tasks, and 8 Synergistic Processing Elements (SPEs), designed for streaming workloads. SPEs are dual-issue in-order SIMD cores with 256KB Local Stores (LS) and 128 registers, 128-bit wide. The PPE is a 2-way Simultaneous Multithreading dual-issue in-order PowerPC processor. A circular ring of four 16B-wide unidirectional channels connects the SPEs, the PPE, the two memory controllers and the two I/O controllers. The operating system runs on the PPE and software can spawn threads to the SPEs. Data has to be explicitly copied to the SPEs LSs using Direct Memory Access (DMA) commands. The Memory Flow Controller (MFC) in each SPE takes care of these DMA transfers and it does so in parallel to the SPEs' SIMD execution unit.

B. HMMERCELL

HMMER is an open source family of tools often used in biosequence analysis [2]. It is aimed specifically at protein sequence analysis. Groups of protein sequences thought of as belonging to the same family are modeled with profile Hidden Markov Models (HMMs). This paper focuses on one tool within the HMMER suite: *hmmsearch*. With this program, an HMM can be compared to a protein sequence database to obtain a list of high scoring sequences and their alignment to the HMM. Execution time is dominated by the Viterbi decoding phase, which is performed once for each sequence in the database. Profiling shows that for all but the simplest workloads, this phase accounts for 98+% of total execution time.

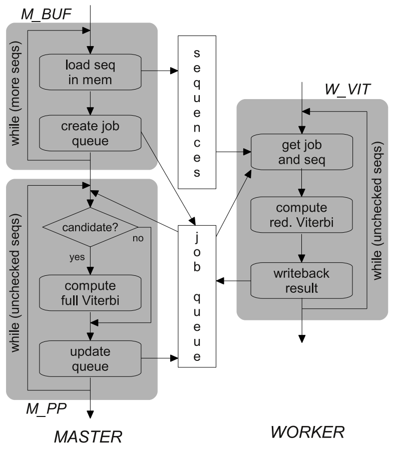

Fig. 1. HMMERCELL program phases.

HMMERCELL [3] is the port of *hmmsearch v2.3.2* to the Cell BE. Parallelism is used at two levels: coarse-grain parallelism by spawning SPE threads that run one Viterbi instance each, and fine-grain parallelism within the SPEs, by using a highly efficient SIMDized version of the Viterbi algorithm [3]. Secondly, due to the small SPE Local Store, the use of small memory footprint version of the Viterbi algorithm is required. Hence, SPEs do not provide a full alignment but only produce an alignment score. High scoring alignments are post-processed on the PPE to obtain the actual alignment.

Fig. 1 shows the internal functioning of HMMERCELL with the PPE and the SPEs acting as master and workers respectively. The three important phases are:

- **M-BUF:** the master buffers the sequences by loading them from disk to memory and creates tasks for the workers by adding entries in a job queue.
- **W-VIT:** each worker copies a protein from main memory to its LS, performs the reduced Viterbi algorithm and writes the result back to main memory.
- **M-PP:** during the post-processing phase, the master runs the full Viterbi algorithm to recover the alignment of highly scored proteins.

IV. Experimental Methodology

Experiments are performed on an IBM QS21 Blade featuring two Cell BE processors (and hence 16 SPEs) running at 3.2GHz and having 4GB of RAM. The code has been compiled with GCC4.1.1 and -O3 flag. Only one PPE was

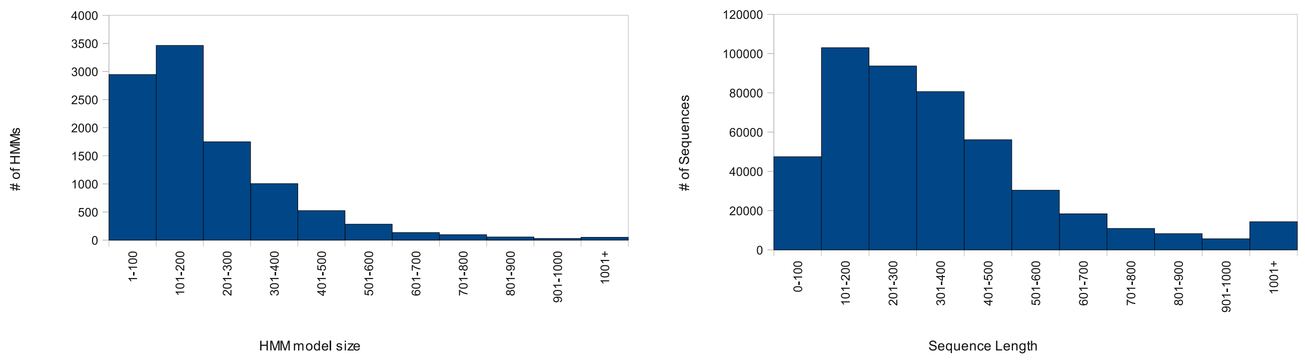

Fig. 2. Distribution of protein database element sizes.

used in our experiments as we intend to study scalability in the number of SPEs.

For the profiling and model validation tests, profile HMMs from the Pfam database [10] and sequence data sets from the UniProt database [11] were used. Fig. 2 shows the current model and sequence length distribution for Pfam and UniProt databases. Only the length of the profile HMMs was taken into account. For the sequence data set, the number of items in the set and the distribution of their lengths was relevant. Based on this information, input test sets have been chosen.

To model the application's behavior, parametric functions for each phase were created. Their dependence on the input and the choice for linear or logarithmic scaling depends on theory, profiling and inspection of code. Based on these equations, the formulas were parametrized by fitting the profiled performance data, using linear or logarithmic regression.

V. HMMER Analytical Model

Our model describes the total execution time of the parallelized version of HMMER using the master-worker paradigm. Based on theoretical expectations and code inspection, we model the required time for each program phase separately and combine these phases together. This results in an accurate model for HMMER performance on multicore platforms.

First, we start with the derivation of the different functions of the model. Then, the model is applied to our implementation platform (the Cell BE) by estimating the numeric coefficients. The analytical results are validated and used to show how the model can be used to derive the maximum effectively usable SPE count. More information on the program phases and the profiling results can be found in [12].

A. Model Derivation

The following parameters are used in our HMMER model:

- T_M, T_W: master-worker processor time;
- T_{M_BUF}, T_{M_PP}, T_{W_VIT}: execution time of phases;
- l_i: length of a specific sequence;
- $\bar{l}$: average length of sequences in the test set;
- m: length of the profile HMM H;
- n: number of sequences in the test set S;
- P_{PP}: chance for a high scoring alignment that requires post-processing on the master;
- q: number of workers used;
- $\alpha, \beta, \gamma, \delta, C_\alpha, C_\beta, C_\gamma, C_\delta$: model coefficients.

The required time (t) for each phase to process a single sequence is expressed in Eqs. 1-3 and is based upon expectations from theory and program inspection. Function I_{PP} acts as an indicator, returning 1 when an alignment between a sequence s_i and the model H is significant for a test set of size n and otherwise returning 0. Such a sequence requires post-processing on the master node, which in our case means recomputing the alignment using the full Viterbi algorithm.

$$t_{M\_BUF} = \alpha \cdot l_i + C_\alpha \tag{1}$$
$$t_{W\_VIT} = \beta \cdot m \cdot l_i + C_\beta \tag{2}$$
$$t_{M\_PP} = (\gamma \cdot m \cdot l_i + C_\gamma) \cdot I_{PP} \tag{3}$$

Aggregating these equations for individual sequences to the entire test set results in Eqs. 4-6. The indicator function I_{PP} has been replaced by the probability function P_{PP}, giving the average chance for a sequence in test set S to require post-processing. Predicting the result of indicator function I_{PP} is difficult, as it requires knowledge of the biological match between the protein model and a specific sequence. Probability P_{PP} however can be estimated based on overall *traceback* count of a test set. Also, T_{W_VIT} states the time required for the Viterbi computations of all the sequences combined.

$$T_{M\_BUF} = n \cdot (\alpha \cdot \bar{l} + C_\alpha) \tag{4}$$
$$T_{W\_VIT} = n \cdot (\beta \cdot m \cdot \bar{l} + C_\beta) \tag{5}$$
$$T_{M\_PP} = n \cdot (\gamma \cdot m \cdot \bar{l} + C_\gamma) \cdot P_{PP} \tag{6}$$

$$with \quad P_{PP} = \frac{\delta \cdot \ln n + C_\delta}{n}$$

To combine the previous equations into an integrated model of HMMER performance, the interrelation between the functions should be taken into account. The dependencies between these three functions are depicted in Fig. 3. W_VIT starts after M_BUF, as at least one sequence should be buffered before

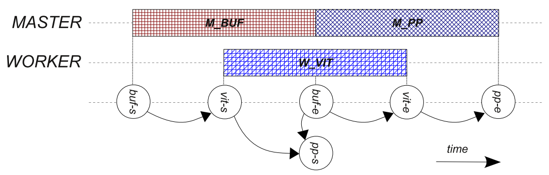

Fig. 3. Relationship of dependence between functions.

processing by the workers can commence. W_VIT ends after M_BUF, as the last sequence to be buffered must be processed as well. M_PP starts when M_BUF finishes, as both buffering and post-processing is performed on the Master node. M_PP ends after W_VIT, as the last processed sequence must be checked by the master.

When a test set contains many thousands of sequences, processing time of any individual sequence is insignificant when compared to total execution time. This observation allows for two simplifications: first, the above dependencies between functions can be approximated as follows: M_BUF and W_VIT can start at the same time, M_PP starts when M_BUF completes, and M_PP must finish after W_VIT. The model also assumes that when M_BUF finishes there are *hits* already available for it to post-process. This is reasonable considering the M_BUF long latencies and because *hits* will usually be randomly distributed in the database.

On the other hand, load balancing between workers is assumed to be perfect, as all processes will finish at approximately the same time. This approximation and hence the accuracy of the model relies on the assumption that the test set contains a large number of sequences, so that the granularity of individual sequence processing becomes very small. This is reasonable, for example a relevant workload such as the UniProt database contains around half a million sequences. Using these assumptions, execution time is modeled as per Eqs. 7-9:

$$T_M = T_{M\_BUF} + T_{M\_PP} \tag{7}$$

$$T_W = (T_{W\_VIT})/q \tag{8}$$

$$T_{TOTAL} = \max(T_M, T_W) \tag{9}$$

B. Model Parametrization

The previous section shows the generalized form of a performance model for an application parallelized using the master-worker paradigm. The coefficients α-δ, C_α-C_δ and function P_{PP} are specific to the actual implementation of HMMER. Here, we show the actual values for our implementation on the Cell BE architecture. Using linear and logarithmic regression, the parametrized values are derived from the profiling results. The extra processing time required by reprocessing high scoring sequences on the master node is incorporated in the coefficient's values.

$$T_{M\_BUF} = n \cdot (\frac{0.19}{10^3} \cdot \bar{l} + \frac{5.52}{10^3}) \tag{10}$$

$$T_{W\_VIT} = n \cdot (\frac{0.59}{10^3} \cdot \frac{m}{10^2} \cdot \bar{l} + \frac{0.88}{10^3}) \tag{11}$$

HMM Length	100	200	300	400	500
q (max worker count)	3	6	9	12	15

TABLE I
MAXIMUM EFFECTIVELY USABLE WORKERS.

$$T_{M\_PP} = n \cdot (\frac{2.25}{10^3} \cdot \frac{m}{10^2} \cdot \bar{l} + \frac{35.7}{10^3}) \cdot P_{PP} \tag{12}$$

$$with \quad P_{PP} = \frac{21.9 \cdot \ln n - 73.2}{n}$$

Combining Eqs. 7-9 and 10-12, total execution time is approximated by:

$$\max \begin{cases} T_M \Rightarrow n \cdot [(\frac{0,19}{10^3} + \frac{2,25}{10^3} \cdot \frac{m}{10^2} \cdot P_{PP}) \cdot \bar{l} \\ \quad +(\frac{5,52}{10^3} + \frac{35,7}{10^3} \cdot P_{PP})] \\ T_W \Rightarrow n \cdot (\frac{0,59}{10^3} \cdot \frac{m}{10^2} \cdot \bar{l} + \frac{0,88}{10^3})/q \end{cases} \tag{13}$$

C. Maximum Effective SPE Count

Eq. 13 can be used to derive the number of workers (or in the case of Cell BE: SPEs) that can be effectively used in scenarios that are constrained by the master's buffering performance (in our case: the PPE). In that case, the number of cores that will saturate the master's buffering capability can be estimated by setting T_{M_BUF} equal to T_W, which results in the maximum effective number of workers q:

$$q \approx \frac{T_{W\_VIT}}{T_{M\_BUF}} = \frac{0,59 \cdot \frac{m}{10^2} \cdot \bar{l} + 0,88}{0,19 \cdot \bar{l} + 5,52} \tag{14}$$

From this equation, it follows that the number of usable workers is solely dependent on HMM model size. Table I gives the maximum number of usable workers for various HMM sizes when using sequences with typical length.

D. Model Validation

To validate our model, additional tests have been performed with new randomly selected data sets of 20.000 and 40.000 sequences (the size is constrained to fit in our blade user quota, but large enough to be significant for the experiments). These are compared against four different HMMs with length 150 and 450 (two representative lengths as seen in Fig. 2). Our model was able to accurately estimate the execution time of M_BUF and W_VIT. The average deviation between result and estimation was 1.5% and 1.7% respectively. Our model for M_PP was found inaccurate, as the number of sequences that require post-processing depends on the biological fit between data set and the HMM, and because the time for post-processing varies considerably for each sequence. However, the M_PP model inaccuracy will only affect overall performance estimation if the application is constrained by the M_PP phase. This only occurs when a high fraction of sequences requires post-processing. However, as *traceback* count scales logarithmically with test set size, this fraction is marginal for realistic data sets. Furthermore, for shared memory architectures where the M_PP phase does not need to compute the full Viterbi algorithm, the significance of the phase is even less than in our case-study. Thus, M_PP contribution to total execution time is negligible. Overall, the average error of our model was 2%.

VI. Conclusions

In this paper we presented an analytical model of HMMER aimed at master-worker parallelization schemes. The model was deduced from program inspection and later compared against execution of HMMERCELL on a real Cell BE based system. The *hmmsearch* kernel was parallelized following to the master-worker approach into three stages: buffering, Viterbi processing and post-processing. The phase that performs the Viterbi calculations is the most time-consuming portion of HMMER and is primarily responsible for overall program behavior. Hence, inspection of this part and its parallelization strategy is very important. Offloading the Viterbi calculations onto the workers is effective: the workload is regular, the computation-to-communication ratio is high, and in theory the number of workers that can be efficiently used is only limited by the number of input sequences. However, the master should be able to create jobs fast enough. This implies that for any given workload a certain worker count exists that will saturate a single master. In this respect, our model shows that the HMM model size determines how many workers can be used for a given master's buffering capability. This is explained by the fact that only the HMM size has a different impact on M_BUF and W_VIT. For short HMMs for instance, worker jobs are small compared to the M_BUF phase, resulting in the master not being able to keep up with preparing jobs. Notice that by letting the workers format the input sequences themselves would improve scalability as less work needs to be done by the master in the buffering phase. More in-depth profiling revealed that 40% of the M_BUF time is spent on I/O (fetching the sequences from the disk). Most of the remaining time is spent on formatting the sequences before they can be processed by the Viterbi algorithm.

Results showed that the M_PP phase introduces uncertainty in our model. However, the M_BUF and W_VIT phases are both linearly dependent on the number of sequences and are the most influential to overall performance. Since M_PP execution time becomes less significant for larger workloads, its impact on the overall model accuracy becomes negligible for realistic data sets. Overall, the model was found to be highly accurate, with only 2% error when compared to execution on real hardware. Notice that on a shared-memory system, the M_PP influence would be reduced even further as there would be no need to re-compute the Viterbi algorithm. HMMER executions on both shared-memory machines and other scratchpad-based architectures such us GPUs could also be modeled by following the proposed methodology and determining the model coefficients.

Our model can be used to estimate the optimal ratio between PPE and SPEs for different working sets. For our case study with the Cell BE, we found that three SPEs saturate the PPE for typical HMM sizes. In general, modeling the behavioral characteristics is a valuable aid for decision-making during design space exploration as it can show the optimal ratio between job creation and consumption. The proposed model can also be used for runtime scheduling.

The findings in this paper are relevant for other bioinformatics applications as well. Most bioinformatics applications contain an abundance of coarse-grained parallelism and the master-worker approach (also known as task-centric) is a useful strategy to divide the work over multiple cores.

Although using only HMMER and the Cell BE in our experiments, the study presented in this paper has a more general scope. Our ultimate goal is to understand the interaction between bioinformatics workloads and heterogeneous multicore architectures. In future work we will analyze the new HMMER3 [13] and apply the same methodology.

Acknowledgments

This work was sponsored by the European Commission in the context of the following projects: SARC (#27648), COMCAS (CA501) and HiPEAC NoE. The authors would like to thank the Barcelona Supercomputing Center for the access to the Cell BE blades, tools and their technical support.

Sebastian Isaza is also affiliated with the Department of Electronic Engineering, University of Antioquia, Colombia.

References

[1] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman, "Basic Local Alignment Search Tool," *Journal of Molecular Biology*, vol. 215, pp. 403–410, 1990.
[2] S. R. Eddy, "Profile Hidden Markov Models," *Bioinformatics*, vol. 14, no. 9, pp. 755–763, 1998.
[3] J. Lu, M. Perrone, K. Albayraktaroglu, , and M. Franklin, "HMMer-Cell: High Performance Protein Profile Searching on the Cell/B.E. Processor," in *ISPASS '08: IEEE International Symposium on Performance Analysis of Systems and software*, 2008, pp. 223 –232.
[4] T. Oliver, L. Yeow, , and B. Schmidt, "Integrating FPGA Acceleration into HMMER," *Parallel Computing*, vol. 34, no. 11, pp. 681–691, 2008.
[5] J. Walters, B. Qudah, and V. Chaudhary, "Accelerating the HMMER Sequence Analysis Suite using Conventional Processors," in *AINA '06: International Conference on Advanced Information Networking and Applications*, 2006, p. 6 pp.
[6] J. P. Walters, R. Darole, , and V. Chaudhary, "Improving MPI-HMMER's Scalability with Parallel I/O," in *IPDPS '09: IEEE International Symposium on Parallel & Distributed Processing*, 2009, pp. 1 –11.
[7] U. Srinivasan, P.-S. Chen, Q. Diao, C.-C. Lim, E. Li, Y. Chen, R. Ju, and Y. Zhang, "Characterization and Analysis of HMMER and SVM-RFE Parallel Bioinformatics Applications," in *IEEE International Symposium on Workload Characterization*, 2005, pp. 87 – 98.
[8] B. Rekapalli, C. Halloy, and I. Zhulin, "HSP-HMMER: A Tool for Protein Domain Identification on a Large Scale," in *ACM Symposium on Applied Computing*, 2009, pp. 766–770.
[9] J. Kahle, M. Day, H. Hofstee, C. Johns, and D. Shippy, "Introduction to the Cell Multiprocessor," *IBM Systems Journal*, vol. 49, no. 4/5, pp. 589–604, 2005.
[10] E. L. Sonnhammer, S. R. Eddy, and R. Durbin, "Pfam: a Comprehensive Database of Protein Domain Families based on Seed Alignments," *Proteins*, vol. 28, no. 3, pp. 405–420, 1997.
[11] "UniProt: Universal Protein Resource," 2011, www.uniprot.org.
[12] E. J. Houtgast, *Scalability of Bioinformatics Applications for Multicore Architectures*. MSc. Thesis, Delft University of Technology, The Netherlands, November 2009.
[13] "Howard Hughes Medical Institute, HMMER Web Site," http://hmmer.janelia.org/.