Optimizing the Forward Algorithm for Hidden Markov Model on IBM Roadrunner clusters

Stefania–Iuliana SOIMAN[1], Ionela RUSU[2], Stefan-Gheorghe PENTIUC[3]
Stefan cel Mare University of Suceava, 720229, Romania
[1]*stefania.soiman@usv.ro,* [2]*ionelar@eed.usv.ro,* [3]*pentiuc@eed.usv.ro*

Abstract—**In this paper we present a parallel solution of the Forward Algorithm for Hidden Markov Models. The Forward algorithm compute a probability of a hidden state from Markov model at a certain time, this process being recursively. The whole process requires large computational resources for those models with a large number of states and long observation sequences. Our solution in order to reduce the computational time is a multilevel parallelization of Forward algorithm. Two types of cores were used in our implementation, for each level of parallelization, cores that are graved on the same chip of PowerXCell8i processor. This hybrid architecture of processors permitted us to obtain a speedup factor over 40 relative to the sequential algorithm for a model with 24 states and 25 millions of observable symbols. Experimental results showed that the parallel Forward algorithm can evaluate the probability of an observation sequence on a hidden Markov model 40 times faster than the classic one does. Based on the performance obtained, we demonstrate the applicability of this parallel implementation of Forward algorithm in complex problems such as large vocabulary speech recognition.**

Index Terms—**forward algorithm, hidden Markov models, multicore processing, parallel hybrid architectures, parallel programming, performance analysis.**

I. INTRODUCTION

Hidden Markov models are used in a variety of pattern recognition problems, as the recognition of speech, gestures, image processing and in the bioinformatics field. Initially introduced in speech recognition problems, the HMM Forward algorithm has become increasingly popular in bioinformatics. Molecular biology uses Markov models as a popular tool in the statistical description of protein families.

As the database of these proteins grows rapidly, a solution is the implementation of HMM algorithms on parallel computing platforms [1-4]. With the introduction of multicore graphic units in the development of parallel algorithms, there was introduced a series of parallel implementations of Markov Hidden models of GPUs (Graphic Processing Units). GPU optimizations of Markov models applied in speech recognition problems, the analysis of biological sequences or processing images appear in papers [5-10]. Although GPUs are increasingly used in parallel computing, achieving superior performances of the CPU, we can often see HMM implementations on parallel systems that use CPU. In 2006, IBM offers an innovative solution in the field of HPC (IBM Roadrunner cluster) whose particularity is the hybrid architecture of the processing unit.

The Cell Broadband Engine (Cell/B.E) implies two types of units/elements of processing on the same chip [11]. The challenge of this type of architecture raises in the development of parallel applications is that we can distribute the amount of data on two levels of parallelization.

Based on these considerations, there are a lot of implementations of Markov models on systems equipped with processor that are based on Cell/B.E architecture. Hence, Viterbi algorithms or Forward applications in HMM models on Cell/B.E architecture are presented in [12-15].

In this paper, we developed a HMM Forward parallel algorithm, in order to reduce the execution time by using the computing power of USV Roadrunner cluster [16]. The probability of the sequence of observations is calculated recursively with the Forward algorithm, the whole process is time consuming and of computing resources for a large number of states or for the long observation sequence.

In [13] we presented the preliminary results of the parallel Forward algorithm which was executed on a parallel machine with hybrid architecture similar to supercomputer IBM Roadrunner. The PowerXCell8i processor with Cell/B.E. architecture is composed of two types of processors: PowerPC Processor Element (PPE) and Synergistic Processing Elements (SPE) used for intensive calculations. The role of PPE processors is to run the operating system, to allocate resources and distribute tasks to SPE cores. Each SPE core has a local memory (LS) used both for storing instructions and data. These differences between processors should be considered, the programmer facing real challenges when developing applications on Cell/B.E [17-21].

Moving on, the differences between the architectures of the two processors do experience problems in data transmission between PPE processor and SPE cores. One solution would be transmitting data as a pointer to a data structure that allows each SPE to receive the effective address via communication mechanisms between the PPE and SPE. SPE can load up to 128b, so it is necessary to align variables in the space of 128b. Accessing memory is different for the two types of processors: PPE works with the main memory with load/store instructions through a register, and SPE accesses the main memory through direct memory access protocol (DMA). At a moment, only one block of 256 KB can be transferred via DMA between main memory and private local memory.

The structure of this paper is as follows: the first section is a brief introduction to Markov models and the parallel implementations of HMM Forward Algorithm (FA) on

This paper has been financially supported within the project entitled „SOCERT. Knowledge society, dynamism through research", contract number POSDRU/159/1.5/S/132406. This project is co-financed by European Social Fund through Sectoral Operational Programme for Human Resources Development 2007-2013. Investing in people!"

Digital Object Identifier 10.4316/AECE.2015.02013

various parallel machines.

In the following sections (section III and IV) we present the mathematical model of the Forward algorithm and considerations to implement this algorithm on USV Roadrunner cluster from our HPC laboratory.

The last part presents and discusses the results obtained using a multilevel parallelized implementation of FA provided by Cell/B.E. architecture.

The paper ends with final remarks, and a discussion regarding the performance comparison study.

II. BACKGROUND

A Markov model is a stochastic model in which the future state of the system depends only on the current state and not on the process of developing the current state. A Hidden Markov Model (HMM) is a Markov model influenced by two stochastic processes at the same time: one that cannot be observed directly (hidden) represented by the evolution of the system state, and an output process represented by the sequence of observations $O_1, O_2, \ldots, O_T$. A HMM can be described by the following elements

- finite set of states $S = \{s_1, s_2, \ldots s_N\}$; a T sequence of states will be referred to $Q = q_1, q_2, \ldots q_T$, where $q_i \in S$;
- distribution of observable symbols: $V = \{v_1, v_2, \ldots v_M\}$;
- distribution of transition probabilities between states $A = \{a_{ij}\}$, where: $a_{ij} = P(q_t = s_j | q_{t-1} = s_i), 1 \le i, j \le N$;
- distribution of observable symbols' probabilities for each state $B = \{b_j(k)\}$, where: $b_j(k) = P(O_t = v_k | q_t = j), 1 \le j \le N$;
- distribution of initial state probability $\Pi = \{\pi_j\}$, where: $\pi_j = P(q_1 = j), 1 \le j \le N$.

The values of observable variables O(t) depend only on the hidden states at time t. With these elements, one HMM can be described as follows:

$$\lambda = (A, B, \pi) \tag{1}$$

In a Markov model, we can observe three fundamental issues:

- assessment: having a sequence of observations and a Markov model $\lambda = (A, B, \pi)$ it is required to calculate $P(O | \lambda)$;
- recognition: being given a sequence of observations $O = O_1, O_2, \ldots, O_T$ and a Markov model $\lambda = (A, B, \pi)$ it is required to find the most probable sequence of states $Q = q_1, q_2, \ldots q_T$ that can generate the appropriate sequence of observations O;
- drive: given a sequence of observations $O = O_1, O_2, \ldots, O_T$ it is required to adjust the model parameters $\lambda = (A, B, \pi)$ so that the probability sequences of observation $P(O | \lambda)$ is maximized.

The first problem, the evaluation of a HMM, is the subject of our paper and is solved with the Forward algorithm [22]. The second problem is solved by using the Viterbi algorithm and the last by using Baum Welch algorithm.

A. Forward Algorithm

In a first approach of brute force we can enumerate all possible sequences of state q of length T and to assess probabilities $P(O | \mathrm{Q}, \lambda)$.

The probability conditioned by sequence Q of states, considering model λ is calculated as follows:

$$P(O | \lambda) = \pi_{q_1} a_{q_1 q_2} a_{q_2 q_3} \ldots a_{q_{T-1} q_T} = \pi_{q_1} \prod_{i=2}^{T} a_{q_{i-1} q_i} \tag{2}$$

Knowing that the observations are independent, the probability conditioned by the appearance of the observable sequence $O = O_1, O_2, \ldots, O_T$, considering model λ and the sequence of states q, is:

$$P(O | q, \lambda) = \prod_{t=1}^{T} P(O_t | q_t, \lambda) = \prod_{i=1}^{T} b_{q_i}(o_i) \tag{3}$$

The probability of the observable sequence O and of the sequence of states q, considering model λ is calculated as follows:

$$P(O, q | \lambda) = P(O | q, \lambda) P(q | \lambda) \tag{4}$$

where the two probabilities on the right are calculated with equations (2) and (3).

The probability of sequence O of model λ is calculated by summing up the probabilities in the equation (4) for all the possible sequences of states q

$$P(O, \lambda) = \sum_q P(O, q | \lambda) = \sum_q P(O | q, \lambda) P(q | \lambda) \tag{5}$$

and by replacing them with the relations in the equations (2) and (3) and we obtain the following:

$$P(O | \lambda) = \sum_q \pi_{q_1} b_{q_1}(o_1) a_{q_1 q_2} b_{q_2}(o_2) a_{q_2 q_3} \ldots \; b_{q_T}(o_T) a_{q_{T-1} q_T} \tag{6}$$

The calculation of probability requires a large amount of calculation, considering the fact that for a model of N states and for sequences of T observable symbols there are possible NT sequences of lengths T, so the amount will be a maximum of NT terms, and each term requires $2T$ multiplications. The result is a $O(TN^T)$ complexity.

Forward Algorithm (FA) effectively calculates the probability that the sequence of observations O to be generated by model $\lambda = (A, B, \pi)$. Having more HMM and a sequence of observations, we choose the model which generated the maximum probability.

We define the forward variable $\alpha_t(i) = P(o_1 o_2 \ldots o_t, q_t = i \mid \lambda)$ as the probability of observing sequence $o_1 o_2 \ldots o_t$, with the condition of reaching time t in state *i*. FA algorithm has three phases

i) Initialization:

$$\alpha_1(i) = \pi_i b_i(o_1), 1 \le i \le N \tag{7}$$

ii) Induction

$$a_t(j) = \left[\sum_{i=1}^{N} a_{t-1}(i) a_{ij}\right] \cdot b_j(o_t), \quad 1 \le j \le N, 2 \le t \le T \tag{8}$$

iii) End

$$P(O \mid \lambda) = \sum_{i=1}^{N} \alpha_T(i) \tag{9}$$

The first step initializes variable α_1 of state *i* with probability π_i and the probability of symbol o_1, $b_i(o_1)$. The $P(O \mid \lambda)$ probability is obtained by summing product $\alpha_{t-1}(i) a_{ij}$ for all states *i* and by multiplying with $b_j(o_t)$ for each *t*. The calculations are repeated for all states *j* (1 ≤j≤N), then we iterate for each *t* (α_2 at moment *t*=2, α_3 at moment *t*=3 α_T at moment *t*=T). The final step is to sum up variables $\alpha_T(\mathrm{i})$ and we obtain the total probability $P(O \mid \lambda)$. The calculating algorithm of Forward probability of complexity $O(TN^2)$, is similar to Viterbi algorithm, with the difference that we calculate the total probability so that the model is in a certain state, whereas Viterbi algorithm calculates the maxim probability.

The Markov models presented are discrete models. In this case the observations belong to a finite state. The continuous Markov models are models that have observable symbols which are not discrete and the probabilities of the observable symbols $b_j(k)$ cannot be used in this case.

III. Parallel Implementation Considerations

The technical specifications of the cluster used to develop the Forward parallel algorithm include triblade units consisting of LS22 servers for management and QS22 for the computational part (Fig. 1). LS22 blade servers include two AMD Opteron quad-core processors at 2.33 GHz and 16 GB of DDR2 memory, and QS22 servers have two PowerXCell 8i processors with 3.2 GHz and 8 GB of DDR2 memory. The communication between compute nodes is of high speed and is made possible through a 4xDRR interface, with a transfer rate of 2GB with a 2 microsecond's latency.

The USV Roadrunner cluster from the HPC laboratory of our University is equipped with 48 QS22 blades, a total of 96 PowerXCell 8i processors. As it can be seen in Fig. 2, the hybrid architecture of the PowerXCell 8i processor offers 8 core accelerators, called SPE, for each main core, called PPE, so the total of compute nodes is of 96 PPE processors and 768 cores of acceleration.

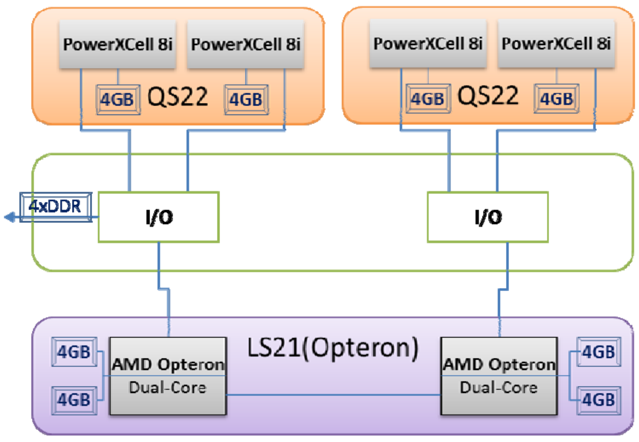

Figure 1. The structure of a triblade compute node

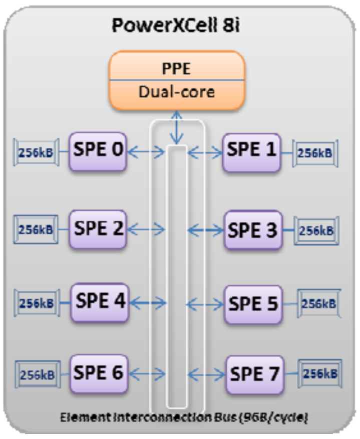

Figure 2. The structure of PowerXCell 8i processor

Based on these considerations, we developed the Forward algorithm as part of Markov model (hereafter called FWD_CBE) on two levels of parallelization, as it follows:

i) On the first FWD_CBE parallelization phase we distributed the *nSEQ* sequences of the observable symbols received as a starting point of the Markov model to the *nPPE* processors. As a consequence, on each PPE we can find nSEQ/nPPE sequences of observable symbols. The distribution of the calculation volume over the grid of PPE processors was done by using the MPI protocol. The MPI parallelization model refers to a master-slave communication, meaning there is a main (master) process that has the role of distributing the calculation volume to the other grid processes, called slave processes, and to collect the partial results, in order to build the final solution (Forward probability- α).

Hence, on each PPE core of the PowerXCell8i, in the same time run the Forward parallel algorithm with *nSEQ/nPPE* sequences of observation of Markov model. On each PPE processor we calculate the partial value α - $\alpha_{PPE\#}$, and after finalizing the calculations, the PPE master collects the partial results on each PPE slave, by applying the *MPI_AllReduce(MPI_SUM)* function, in order to build the final solution, α. In our implementation we considered the master process, after dividing the calculation volume equally between all the existing processes, including its own, to act as a slave process and, after calculating the probability α, to adopt the master processing function again.

ii) the second level of parallelization of the FWD_CBE algorithm was implemented for all the SPE accelerating cores of each PowerXCell8i processor.

Considering the advantage of the PowerXCell8i processors' architecture, by activating the SPE accelerating cores, we can increase the performance of the algorithm by approximately two orders of magnitude. Each SPE core has 256 KB of local memory used for data and also for the application. For models with a large number of states, the local memory capacity of each SPE is exceeded. Thus, it is necessary to find programming techniques and strategies, in order to use the limited capacity of the resources of SPE processors. The FWD_CBE algorithm proposes to distribute the calculation of each core SPE, so that the time of calculating the Forward probability for HMM models with a large number of states is reduced. Fig. 3 comprises the core of FWD_CBE algorithm on PPEs/ SPEs for a model of less states (*N* = 24, where N is the number of states of the model), in order to exemplify the communication and transfers between PPEs and SPEs.

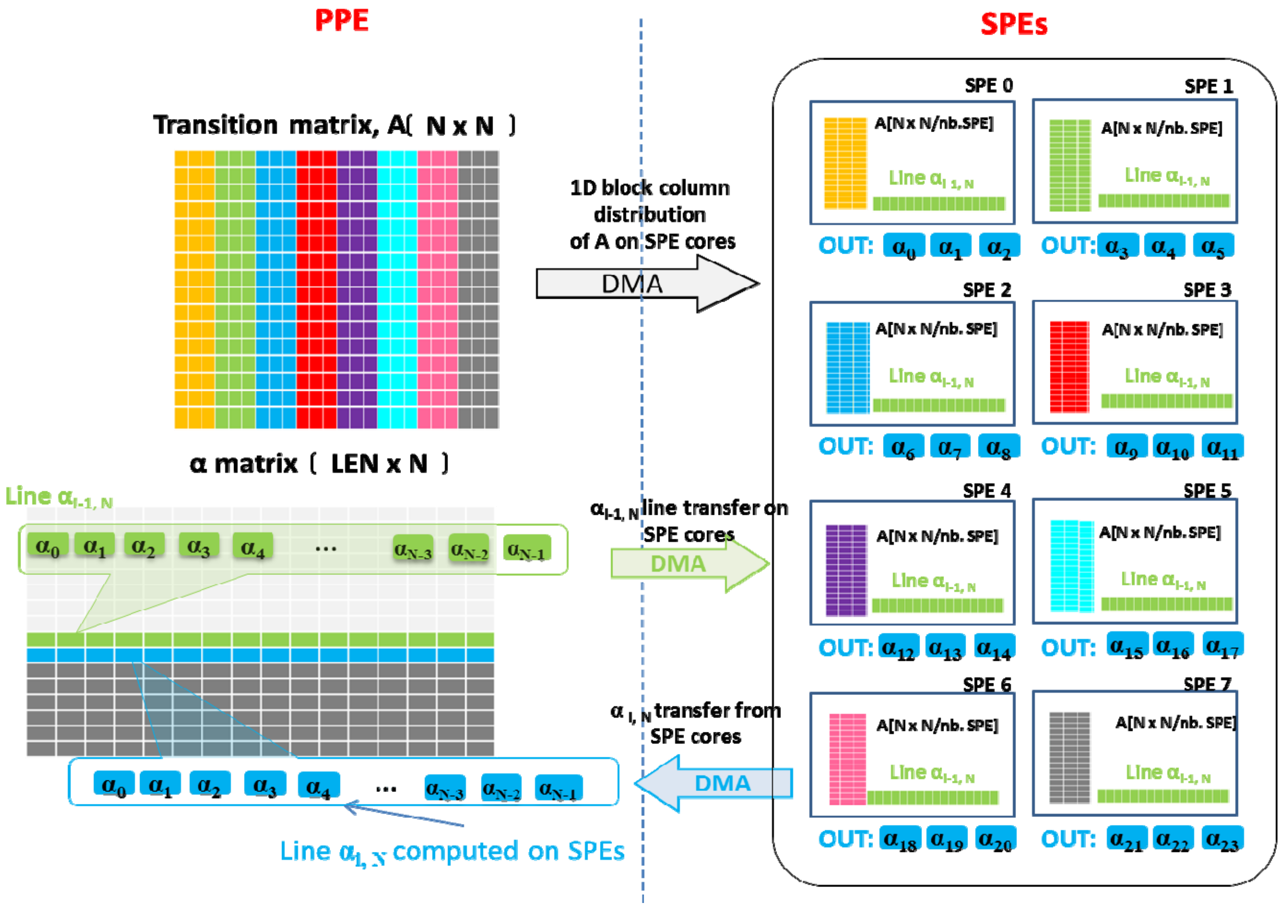
PPE
SPEs
Transition matrix, A(N x N)
α matrix (LEN x N)
Line α_{I-1, N}
1D block column distribution of A on SPE cores
DMA
α_{I-1, N} line transfer on SPE cores
DMA
α_{I, N} transfer from SPE cores
DMA
Line α_{I, N} computed on SPEs
SPE 0
SPE 1
SPE 2
SPE 3
SPE 4
SPE 5
SPE 6
SPE 7
A[N x N/nb.SPE]
Line α_{I-1, N}
OUT:

Figure 3. The kernel of FWD_CBE algorithm executed on PPE processors when the SPE cores are use

The proposed strategy or the second level of parallelization implies the forward (alpha) matrix calculation line by line, on each SPE, on *Line*$_{\alpha}$ variable. The first line of matrix α is calculated on PPE processors, based on the initial probabilities' distribution, Π, for each state, the calculation of the other *LEN-1* lines (*LEN* represents the length of sequence of the observable symbols) being performed on the 8 SPE accelerators, each linked to a PPE. In order to calculate a line from the probability matrix (*Line*$_{\alpha i}$) it is required to use the entire matrix of transition probabilities between states, *A*. In the case of HMM models with a large number of states, the local memory capacity on SPEs are exceeded by storing the entire matrix *A*.

Hence, the distribution of the transition probabilities' matrix between states, *A* is transmitted by SPE processors, through direct access memory (Direct Access Memory – DMA), using blocks of columns (1D block column distribution). For the 24-state model, exemplified in Fig. 5, each SPE processor is determined by the calculation of a block of three values of probability α (α_i $_{\alpha i+1}$ α_{i+2}), for the three columns in the matrix of transition probabilities between states, *A*, associated. Each block of columns of matrix α, *Line*$_{\alpha i,N}$ depends on the value of the previous line *Line*$_{\alpha i\text{-}1,N}$, which requires the update of the line of matrix *α* for each PPE processor, after finalizing the construction of its values on SPEs. The first line of matrix *α* is initialized with the initial probability of states, Π, on each PPE processor. For the following *LEN-1* lines, when calculating a single line we need the entire matrix of transition probabilities between states (order N, squared) – (*A*), exceeding the capacity of the local memory of each SPE unit, for a number which is larger than 128 states.

After completing the calculations of each SPE, the block of values α_i $_{\alpha i+1}$ α_{i+2} is transferred via DMA and after that the line is built on PPE. At the level of PPE processors, the line of matrix α_i (*Line*$_{\alpha i\text{-}1,N}$ is built, which will be transferred later to the SPE accelerators in order to calculate the next line (*Line*$_{\alpha i,N}$). The process continues to block the sequence of observations assigned to each PPE processor.

The data transfer between PPE and SPE, the transfer of

HMM model parameters from PPE to SPE and the transfer of the partial results of the calculation of Forward probability from SPE to PPE is done, also, via DMA.

IV. Experimental Results and Discussion

In order to obtain the experimental results of the parallel Forward algorithm, there were used artificially generated HMM models. A first step in assesing the performance of FWD_CBE algorithm was recording and calculating the index of performance for Markov models for a fixed numbers of states.

A first step in assesing the performance of FWD_CBE algorithm was recording the execution time for Markov models for a fixed numbers of states, the observable sequences being of different lengths (from 80 thousand to 25 million observable symbols). We run the tests on USV Roadrunner (IBM) with 16 QS22 blade servers. The parallel algorithm was assesed on 32 PowerXcell 8i processors, by enabling the SPE cores. To highlight the advantage of a parallel approach of the proposed algorithm, we registered the execution time of the Forward sequential algorithm on a single PowerXCell 8i processor.

The execution time obtained by using the parallel Forward algorithm (FWD_CBE) depending on the length of the sequence of observations, are shown in Fig. 4. On each MPI process we calculated the Forward local probability corresponding to the block in the observation sequence distributed to the MPI process.

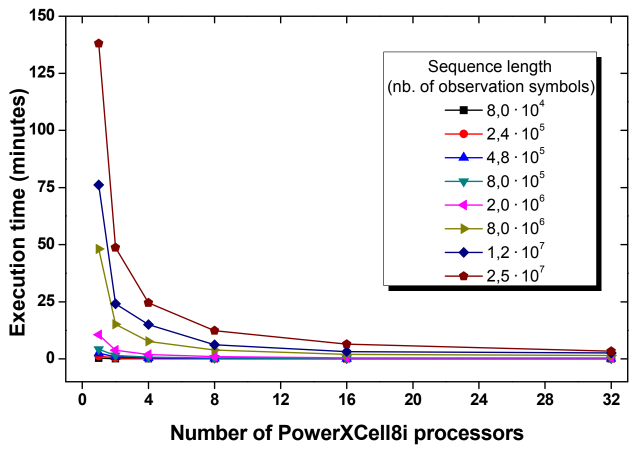

Figure 4. Running time of parallel Forward Algorithm on PowerXCell8i processors for different length of observation sequence

In order to evaluate the performances of the FWD_CBE algorithm, in terms of the number of states, there were generated Markov models with a number of 24 to 2014 states, having one single sequence of 3200 observable symbols.

The results obtained by implementing the parallel FWD_CBE algorithm on 2 - 32 PPE processors, show a better performance of the execution time that is registered on models with a number higher than 256.

The highest efficiency of the parallel algorithm is for the test consisting of 25,6 million observable symbols, the calculation of probability on a single processor lasting over 2 hours, whereas on 32 PPE processors the calculation is completed in 3 and a half minutes, which shows an acceleration factor of more than 40. The speedup obtained with the FWD_CBE algorithm is shown in Fig. 5.

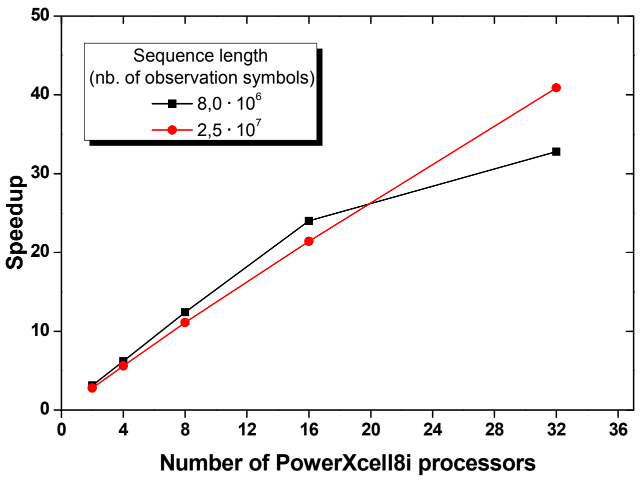

Figure 5. The acceleration gained with FWD_MPI algorithm on PowerXCell8i processors for two different observation sequences.

Due to increasing the time of communication between processors, whereas using more than 20 processors and smaller observation sequences the efficiency of the FWD_CBE algorithm begins to decline.

Table 1 shows the time scale of running the Forward parallel algorithm for 6 HMM models with a single sequence of 3200 observable symbols, using 1, 2, 4, 8, 16 and 32 PPE processors with 8 SPE cores enabled.

TABLE I. COMPUTATION TIME OF A PROBABILITY OF PARALLEL FORWARD ALGORITM ON POWERXCELL8I PROCESSORS FOR DIFFERENT NUMBER OF STATES

# of states / # of PPE	24	64	128	256	512	1024
1	1,0	7,3	29,4	145,2	694,5	2803,0
2	0,8	1,5	3,2	29,2	145,5	562,9
4	0,6	0,8	1,7	15,1	73,1	283,6
8	0,3	0,4	1,6	7,6	37,0	141,1
16	0,2	0,3	0,8	4,1	18,9	73,1
32	0,1	0,2	0,5	2,3	10,3	37,0

V. Conclusion

FWD_CBE algorithm proposed in this paper was developed by using SPE accelerating cores of the PowerXCell8i processor, which are common for the Cell/B.E architecture.

The FWD probability is calculated recursively, the whole process requiring large computational resources for the models with a large number of states and long observation sequences. FWD_CBE optimizes the performance of numerous calculations of probabilities in a multilevel parallelization way. PPE processors represent the first level of parallelization, the length of the observation sequence being divided between them. Each PPE divides the number of the states of those 8 SPE cores, thus making possible the second level of parallelization as well.

In the study of the FWD_CBE algorithm's performance we conducted a series of tests showing different parameters of the hidden Markov model (the number of states and the length of observation sequence), tests that were run on the USV Roadrunner (IBM) cluster. The tests followed the registration of both the execution time, when the PPE processors using HMM models with a reduced number of states and long observation sequences are shown, and also the execution time obtained when the accelerator cores of PowerXCell 8i processor are activated, using the HMM model with a large space of states.

Based on the data above, we sketched these diagrams, out of which we can identify the factors contributing to the increase or decrease of the algorithm's performance. The best accelerator factor obtained by using the FWD_CBE parallel algorithm was over 40 on a HMM model with 24 states and a sequence of 25 million observable symbols, on 256 SPE cores.

The results obtained showed a high level of scalability of the USV Roadrunner (IBM) cluster, in order to fix the problems involving the hidden Markov models with a large number of states and observable long sequences, used in solving complex problems, such as those that imply a large dimension vocabulary.

ACKNOWLEDGMENT

The cluster used in order to carry out the experiments was purchased within a project entitled "Grid for Developing Pattern Recognition and Distributed Artificial Intelligence Applications–GRIDNORD", 80/13.09.2007, PN II, Research Capacities Programme of the Romanian National Agency for Scientific Research.

REFERENCES

[1] T. F. Oliver, B. Schmidt, Y. Jakop, D. L. Maskell, "High Speed Biological Sequence Analysis With Hidden Markov Models on Reconfigurable Platforms", Information Technology in Biomedicine, IEEE Transactions on 13(5): 740-746. [Online] Available: doi: 10.1109/TITB.2007.904632

[2] A. Sand, Pedersen, C. N. S. Pedersen, T. Mailund, A. T. Brask, "HMMlib: A C++ Library for General Hidden Markov Models Exploiting Modern CPUs", 2010 Ninth International Workshop on Parallel and Distributed Methods in Verification, and Second International Workshop on High Performance Computational Systems Biology, IEEE 2010, pp. 126 – 134, 2010. [Online]. Available: doi: 10.1109/PDMC-HiBi.2010.24

[3] J. Nielsen, A. Sand, "Algorithms for a Parallel Implementation of Hidden Markov Models with a Small State Space", in Proc. IPDPS Workshops, IEEE 2011, pp.452-459. [Online]. Available: doi: 10.1109/IPDPS.2011.181

[4] X. Meng, Y. Ji, "Modern Computational Techniques for the HMMER Sequence Analysis", vol.2013, 13 pages, 2013. [Online]. Available: doi: 10.1155/2013/252183

[5] S. Gorgunoglu, I. M. Orak, A. Cavusoglu, M. Gok, "Examination of Speed Contribution of Parallelization for Several Fingerprint Pre-Processing Algorithms," Advances in Electrical and Computer Engineering, vol. 14, no. 2, pp. 3-8, 2014, doi:10.4316/AECE.2014.02001

[6] L. Yu, Y. Ukidave and D. Kaeli, "GPU-accelerated HMM for Speech Recognition", Workshop - Heterogeneous and Unconventional Cluster Architectures and Applications (HUCAA) September, 2014.

[7] J. Li, S. Chen, Y. Li, "The fast evaluation of hidden Markov models on GPU," *Intelligent Computing and Intelligent Systems, 2009. ICIS 2009. IEEE International Conference on* , vol.4, no., pp.426,430, 20-22 Nov. 2009. doi: 10.1109/ICICISYS.2009.5357649

[8] D. Zhihui, Y. Zhaoming, D.A. Bader, "A tile-based parallel Viterbi algorithm for biological sequence alignment on GPU with CUDA," *Parallel & Distributed Processing, Workshops and Phd Forum (IPDPSW), 2010 IEEE International Symposium on* , vol., no., pp.1,8, 19-23 April 2010. doi: 10.1109/IPDPSW.2010.5470903

[9] J.P. Walters, V. Balu, S. Kompalli, V. Chaudhary, "Evaluating the use of GPUs in liver image segmentation and HMMER database searches," *Parallel & Distributed Processing, 2009. IPDPS 2009. IEEE International Symposium on* , vol., no., pp.1,12, 23-29 May 2009. doi: 10.1109/IPDPS.2009.5161073

[10] W. Lee, J. Kim, I. Lane, "GPU Accelerated Model Combination for Robust Speech Recognition and Keyword Search", GPU Technology Conference, March 2014

[11] T. Chen, R. Raghavan, J. N. Dale, E. Iwata, "Cell Broadband Engine Architecture and its first implementation—A performance view", IBM Journal of Research and Development , vol.51, no.5, pp.559-572, 2007. [Online]. Available: doi:10.1147/rd.515.0559

[12] V. Sachdeva, M. Kistler, E. Speight, T.-H. K. Tzeng, "Exploring the viability of the Cell Broadband Engine for bioinformatics applications, " Parallel Computing, vol. 34, no. 11, pp. 616–626, 2008. [Online]. Available: http://dx.doi.org/10.1016/j.parco.2008.04.001

[13] S.–I. Soiman, I. Rusu, S.-G. Pentiuc, "A parallel accelerated approach of HMM Forward Algorithm for IBM Roadrunner clusters", Proceedings of the 12th Int. Conf. on Development and Appl. Systems, May 2014, pp. 184-188. . [Online]. Available: doi: 10.1109/DAAS.2014.6842452

[14] S.–I. Soiman, I. Rusu, S.-G. Pentiuc, " Multilevel Parallelized Forward Algorithm for Hidden Markov Models on IBM Roadrunner Cluster", Proceedings of the 20th Int. Conf. on Control Systems and Computer Science, May 2015.

[15] F. Blagojevic, A. Stamatakis, C. D. Antonopoulos, D. S. Nikolopoulos, "RAxML-Cell: Parallel Phylogenetic Tree Inference on the Cell Broadband Engine, " Parallel and Distributed Processing Symposium, IEEE International, pp. 1-10, 2007. [Online]. Available: doi: 10.1109/IPDPS.2007.370267

[16] GRIDNORD Project. High Performance Computing Laboratory of the Faculty of Electrical Engineering and Computer Science, Suceava, Romania, 2012, http://eed.usv.ro/gridnord/en/

[17] A. L. Varbanescu, H. Sips, K.A. Ross, Q. Liu, A. Natsev, J.R. Smith and L.K. Liu, "Evaluating application mapping scenarios on the Cell/B.E, " Concurrency and Computation: Practice and Experience, 21, pp. 85-100, 2009. [Online]. Available: doi: 10.1002/cpe.1335

[18] A. Arevalo, R.M. Matinata, M. Pandian, E. Peri, K. Ruby, F. Thomas, C. Almond: Programming for the Cell Broadband Engine. IBM Redbooks (2008)

[19] C. A. Tanase, V. G. Gaitan, "Threads Pipelining on the CellBE Systems", Advances in Electrical and Computer Engineering, vol. 13, no. 3, pp. 121-126, 2013. [Online]. Available: doi:10.4316/AECE.2013.03019

[20] S.-G. Pentiuc, I. Ungurean, "Multilevel Parallelization of Unsupervised Learning Algorithms in Pattern Recognition on a Roadrunner Architecture ", Intelligent Distributed Computing V, vol. 382, pp.71 – 80, 2011. [Online]. Available: doi: 10.1007/978-3-642-24013-3_8

[21] I. Ungurean, V.-G. Gaitan, N.-C. Gaitan, "Intensive computing on a large data volume with a short-vector single instruction multiple data processor," Computers & Digital Techniques, IET, vol.8, no.5, pp.219-228, 2014. [Online]. Available: doi: 10.1049/iet-cdt.2013.0149

[22] L. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition", Proceedings of IEEE, Vol. 77, pp. 257-285, 1989. [Online]. Available: http://dx.doi.org/10.1109/5.18626.