

Beta Barrel Transmembrane Protein Prediction using TMBHunt algorithm on the Cell Broadband Engine

Prasanna Kumar B , Ritesh M Nayak, Shrisha Rao
{*bprasanna,ritesh.m,srao*}@iiitb.ac.in
International Institute of Information Technology
Bangalore 560 100
India

Abstract

Beta-BarrelTransmembrane (BBTM) proteins perform diverse functional roles including bacterial adhesion, structural integrity of the cell wall, and material transport. These proteins are much harder to identify due to the big variation in the short transmembrane segments. There are few methods available to identify BBTM proteins by exploring properties such as sequence profiles, beta-barrel score and signal peptides, the distribution of multiple properties on protein sequences. TMB-Hunt is one such open source method that uses a k-Nearest Neighbour (k-NN) algorithm to discriminate between BBTM and Non-BBTM proteins on the basis of their amino acid composition. In this paper, we present the implementation of TMB-hunt on Cell Broadband Engine (CBE). CBE (Cell Broadband Engine) is a heterogeneous multicore processor with one Power PC Processing Element (PPE) and eight Synergistic Processing Elements (SPE). We take advantages of CBEs parallel processing to optimize the performance of TMBhunt to achieve super linear scalability. By parallelizing an otherwise serial operation, our implementation promises faster prediction of the BBTM/AHTM.

Keywords: Multicore, Computation Biology, Parallel Computing, Cell Broadband Engine, BBTM

1 Introduction

BBTM prediction helps in discovering mutation effects on bacteria and also helps in synthesis of new drugs. Prediction of this protein with amino acid composition based method has shown better improvement. This basically involves comparing the new sequences with non-redundant dataset of 32 betabarrel outer membrane proteins. For a real world set of input protein sequences, which is in the order of millions, the prediction takes a long time. Cell Broadband Engine (CBE), jointly developed by Sony Computer Entertainment, Toshiba, and IBM, an alliance known as STI, is a heterogeneous multicore

platform. The CBE is gaining a lot of attention in the multicore and parallel computing arena, thanks largely due to its massive processing power and more and more compute intensive programs are being ported to run on the CBE. It would be helpful to see how TMB-hunt method of BBTM prediction is solved using a parallel algorithm on the CBE. There are three major things to be considered before implementing. Firstly assigning of the data from PPE to SPE in a scheduled way. Secondly adapting to the memory hierarchy posed by Cell Broadband Engine. Thirdly optimizing the logic to the CBE specific code. This paper presents how the above problems can be solved.

The main contributions of this paper include:

Porting of the TMBhunt onto CBE. We present an example of how the communication and computation pattern of BBTM prediction can be implemented on CBE. The result shows that CBE is a potential architecture for parallelizing the computations involving independent large datasets especially in computation biology.

We also explore the possible methods of parallelization on the CBE . We study the effects of multigrained parallelism on the CellBE compared to a intuitive parallel approach. The rest of the paper is organized as follows. Section 2 summarizes related work on programming support for Cell and Cell BE architecture. Section 3 introduces the implementation and methods of parallelizing on the CBE. Section 4 presents the step by step implementation of TMB-hunt method into Cell BE. Section 5 presents the results and performance analysis. Section 7 concludes the paper.

2 Related Work

TMB-Hunt is a program that uses a k-Nearest Neighbour (k-NN) algorithm to discriminate between bbtm and non-bbtm proteins on the basis of their amino acid composition. A major advantage of this approach is that, because it does not rely on beta-strand detection, it does not require resolved structures and thus larger, more representative, training sets could be used. Transmembrane beta-barrel (TMB) proteins perform diverse functional roles including bacterial adhesion, structural integrity of the cell wall, and material transport. Unlike transmembrane alpha-helical proteins that can be easily identified by the long hydrophobic transmembrane regions, TMB proteins are much harder to identify due to the big variation in the short transmembrane segments. There are few methods available to identify TMB proteins by exploring properties such as sequence profiles, beta-barrel score and signal peptides, the distribution of multiple properties on protein sequences. The location of the membrane lipid bilayer relative to a transmembrane protein structure is important in protein engineering. Since it is not present on the determined structures, it is essential to automatically define the membrane embedded protein region in order to test

mutation effects or to design potential drugs.

Garrow et al. (2005)[1] developed a TMB-Hunt method to identify TMB proteins based on residue composition. There is an implementation of the TMB hunt on a webserver[6]. The web server, available at www.bioinformatics.leeds.ac.uk/betaBarrel, allows screening of up to 10,000 sequences in a single query and provides results and key statistics in a simple color coded format.

Blagojevic ,F. et al (2007) have studied the effects of multigrained parallelism the Cell Broadband Engine. They have also used the CBE for parallel phylogenetic tree construction of the RaxML-CELL, an application of the CBE for computational biology.

2.1 Cell Broadband Engine

The Cell Architecture grew from a challenge posed by Sony and Toshiba to provide power-efficient and cost-effective high-performance processing for a wide range of applications, including the most demanding consumer appliance: game consoles[2]. Cell is a heterogeneous chip multiprocessor that consists of an IBM 64-bit Power Architecture core, augmented with eight specialized co-processors based on a novel single-instruction multiple-data (SIMD) architecture called Synergistic Processor Unit (SPU), which is for data-intensive processing, like that found in cryptography, media and scientific applications. The system is integrated by a coherent on-chip bus[3]. The cell processor is capable of very high speed parallel computation. Here are some of CBEs statistics[3].

Observed clock speed: 4 GHz

Peak performance (single precision): 256 GFlops

Peak performance (double precision): 26 GFlops

Local storage size per SPU: 256KB

Cell BE can find uses in many fields which include problems that are parallel in nature. Biotechnology is one such fields that requires as much computation as it can get. Folding@Home, a project to find out the effects and cause of Protein Folding is one of the worlds largest distributed computing projects. Such computation power is still insufficient to many of the other mysteries of biology and biologists are looking towards large scale computation resources for their experiments. TMB hunt is one such problem in computational biology.

3 Parallelism and Granularity

Decision on the granularity of the threading operations was very critical in parallelizing the TMBHunt algorithm. TMBHunt uses the K -Nearest Neighbor for its calculation and there are well defined partitioning approaches to making the serial algorithm parallel. In doing so, there are many consequences that don't prove to be beneficial for a calculation like ours. CellBE encourages a truly parallel programming model in which each thread is self sufficient and doesn't need to coordinate with the other threads to finish its task. Inter SPE communication only leads to delays bound by the longest running thread. Means of communication between SPE's is not well defined and hard to implement. In addition to these factors, TMBHunt runs the nearest neighbor calculation on multiple inputs in the hopes of finding a BBTM match, which is also another level of data granularity that can be exploited for parallelism. With these considerations, we set up some experiments to figure out what granularity of threading can be utilized[7]. We found that a lot of problems with partitioning of the data for the K Nearest Neighbor algorithm due to its inter core dependency while reducing the result from parallel operations. The first approach involved having one dataset run on the PPU which will split the data into $k-1$ equal parts (K being the number of idle cores) and then use the Kth core for collating the result. This approach mandates at least two of the cores to be idle, one for partitioning and the other for collating the results, which is inefficient. The second approach worked at a higher level of threading granularity compared to the former approach. TMBHunt computes the KNN algorithm on input dataset, with many reference datasets, to find the Euclidean distance between them. instead of partitioning the dataset , a better approach would be to partition the reference files. Each core now runs the KNN algorithm independently on the reference dataset. The PPE takes the job of allocating and managing the reference files. When a core becomes idle the PPE allocates it the next reference dataset. This is the ideal scenario.

CBE Software development kit only supports pThreads implementation and due to the lack of a reliable asynchronous non blocking call to check the state of a running thread , the ideal parallel scenario as described above could not be implemented. To overcome this problem a barrier synchronization method was used. Eight threads are assigned their corresponding datasets and the PPE synchronization thread waits for all the eight threads to complete their tasks. Once all the eight threads have completed their assigned task, new dataset is allocated to the threads for computation. This process continues till the input dataset is compared with all the reference datasets(in batches of 8). The execution time for each run is bound by the longest running thread, but with each dataset taking approximately the same running time, this approach worked best for the problem.

4 Implementation

There are two programs used to achieve to the parallelism. One program is used for PPE and another program is for SPE, which performs compute intensive operations. We have taken the existing open source implementation project to predict the BBTM proteins using KNN algorithm and optimized it to adapt the parallel processing power of CellBE. We have used a perl file to create input query files by reading the genome sequence file. The input query file contains the percentage of 20 amino acids present in the protein. The GA vector and z-scores are used to make decision on BBTM existance. The memory architecture of Cell BE comes with set of limitations. The common data which can be exchanged between PPE and SPE has a size limitation of 16Kb. Each SPE contains the local memory size of 256Kb. The total size of data shared between PPE and SPE should be multiple of 16 [9]. The PPE program reads the contents of GA Vector, ZScores and Percentage files and stored in two dimensional arrays and passed to each running threads. The input query file name and output query file name are passed to SPE to be opened there and perform the calculation using KNN algorithm. The prediction is done using the two dimensional arrays passed from PPE and the result is written in output files for further analysis. Barrier synchronization method is used to synchronization of threads. As described above, the computation time for each parallel run of the algorithm is bound by the longest running thread.

4.1 Splitting of Data

Firstly a Perl file is used to create a set of input query files based on the input genome sequences file. The genome sequence file used is of FASTA format with .fa extension [10]. This file is in turn passed to the PPE program, which is written in C language, and number of query sequences is decided.

The PPE program reads the contents of GA Vector, ZScores and Percentage files and stores them in two dimensional arrays which are passed to each of the running threads. This is the major performance tune up done by reducing File I/O operations in the SPEs. In this method all the two dimensional arrays are passed to SPEs through SPE context. The names of input and output files, which should be opened at SPE, are sent to SPE. The total number of SPE threads created are 8 [3]. The query files are equally divided and sent to SPE in batches of 8. The .pc files which contain percentages of each amino acid in BBTM protein and AHTM protein are passed to SPE this way. Because of the memory limitation of 16Kb and local store size of 256Kb, only 32 percentage sequences are considered in each run.

4.2 Prediction Process

The prediction of BBTM is performed in the SPEs. Each SPE is given a specific input and output file to process. The input file for a SPE is a query file containing the percentage of amino acids. Each thread opens an output file and writes the

```

//inputFile: candidate for BBTM detection
//referenceFiles: set of files to be compared with the inputFile

//duplicate the inputFile for each SPE
inputFiles[8] = inputFile
refCount= 1;

while ( refCount < no_of_referenceFiles)
{
threadArray[8] // array corresponding to each thread
for ( threadCount = 1 to 8 )
{
//spawn SPE threads which take the duplicated inputFile, referenceFile
SPE_Create_Thread(inputFiles[threadCount],referenceFiles[refCount]
expectedOutputFile, threadArray[threadCount])
refCount ++
}
for(threadCount = 1 to 8 )
{
Barrier_Sync_Wait(threadArray[threadCount])
}
}

```

Figure 1: PPE logic for BBTM prediction

```

SPE_Program(inputFile, referenceFile,expectedOutputFile)
{
result = TMBHunt_KNN_Logic(inputFile,referenceFile)
Write_File(expectedOutputFile, result)
return
}

```

Figure 2: SPE logic for BBTM prediction

results into that. This query percentage is compared with BBTM percentage file using the Euclidean method of K-NN(K-Nearest Neighbor algorithm). The value of K used in the K-NN is set to 5. The GA vector and zscores are used to make decision on BBTM existence. The result is written in output files for further analysis. In SPE total size of variables declared and processing memory should be limited to 256KBs. So, it is essential to declare the variables considering the SPEs local stores limitation. The total run time of the program is calculated based on the number of processor cycles taken.

5 Performance Analysis

We compared the Cell BE implementation with a similar implementation on a uni processor using built pThread implementation. CellBE has shown a significant improvement in performance compared to a stand alone processor. We achieved super linear scalability using the 9 cores present in the CellBE. Since, we used a simulator to calculate the values depicted below, the values may not represent real world execution times. The execution time lost precision as we kept increasing the dataset size and after a certain point, it became impractical. We have restricted our analysis to the consistent part of the tests. The simulator behaves consistently for small datasets and even in those circumstances, on an average, we achieved a scalability factor of 20. Given that the uniprocessor is plagued with context switching delays, locks and other synchronization prob-

lems a factor of 20 is a tremendous speed up in the prediction process.

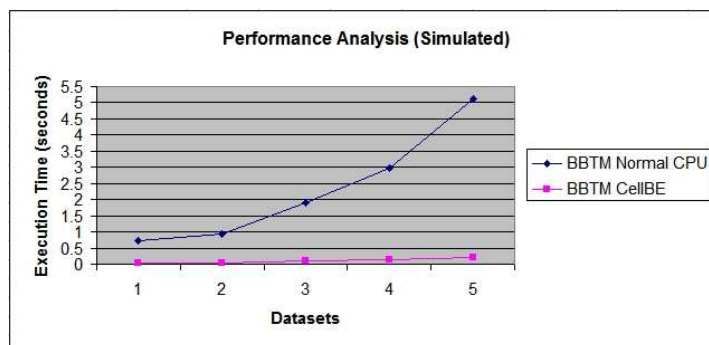


Figure 3: Comparative graph of the execution times on a Single CPU vs CellBE(Simulator).

6 Conclusion

This paper presented various approaches to parallelizing the TMBHunt method on the Cell Broadband Engine. We also explored many Cell BE SDK specific optimizations and the performance implications of these optimizations. By off-loading the most time consuming prediction function on to SPE, we implement a parallel version of BBTM Prediction with less overhead. We also discussed the pros and cons of using different levels of granularity in data while programming using the CBE. Even though our approach was sub optimal due to the limitation posed by the SDK, we were able to use our programming model to tap most of CBE's processing power. We also touched aspects of memory structuring for solving problems on the CellBE to reduce transfer of data between cores. We were able to show super linear scalability by implementing the TMB-Hunt method on the CBE. This approach will dramatically reduce the time for prediction of the BBTM proteins that could reduce time for its applications like synthesis of drugs. Computational biology is filled with problems of this genre that can be solved with significant performance improvements using the CellBE.

References

- [1] Andrew G. Garrow, Alison Agnew and David R. Westhead - TMB-Hunt: a web server to screen sequence sets for transmembrane b-barrel proteins
- [2] Liu,Q., Zhu,Y., Wang,B. and Li,Y. (2003)Identification of beta-barrel membrane proteins based on amino acid composition properties and predicted secondary structure. *Comput. Biol. Chem.*, 27, 355361
- [3] Bagos,P.G., Liakopoulos,T.D., Spyropoulos,I.C. and Hamodrakas,S.J. (2004) - PRED-TMBB: a web server for predicting the topology of betabarrel outer membrane proteins. *Nucleic Acids Res.*, 32, W400W404

- [4] Blagojevic ,F. , Stamatakis ,A. ,Antonopoulos, C., Nikolopoulos, D.S : Dynamic multigrain Parallelization on the Cell Broadband Engine (2007)
- [5] Blagojevic ,F. , Stamatakis ,A. ,Antonopoulos, C., Nikolopoulos, D.S - RAxml -CELL :parallel phylogenetic tree construction on the Cell Broadband Engine (2007)
- [6] Bellens, P., Perez, J.M., Badia, R.M., Labarta, J. :Memory-CELLS: a programming model for the Cell BE Architecture in *proceedings of the 2006 ACM/IEEE conference on supercomputing, Tampa, Florida (2006)*