Implementation of Recursive Brute Force for Solving Motif Finding Problem on Multi-Core

Marwa A. Radad, Department of Computer Science and Engineering, Faculty of Electronic Engineering, Menoufia University, Menouf, Egypt

Nawal A. El-Fishawy, Department of Computer Science and Engineering, Faculty of Electronic Engineering, Menoufia University, Menouf, Egypt

Hossam M. Faheem, Department of Computer Systems, Faculty of Computer and Information Sciences, Ain Shams University, Cairo, Egypt

ABSTRACT

Motif is an over-represented pattern in biological sequence. Motif discovery is a major challenge in bioinformatics. Pattern mismatches phenomena makes motif mining very difficult. Brute Force approaches take exponential time with motif length to solve this problem. In this paper, the authors discuss a Recursive-Brute Force algorithm. Its average case time complexity is exponential with the allowed mutations instead of the motif length. Modern Multi-Core architecture revolution encourages us to parallelize our algorithm. We implement the algorithm using two different approaches. A multi-threaded version (OMP-RBF) is implemented using OpenMP. OMP-RBF suffers from a serious performance degradation due to the heap contention problem. The authors have investigated different solutions to solve the heap contention problem. The second implementation is based on MPI that is called MPI-RBF. The efficient handling of the data locality boost the scalability of the MPI-RBF. The authors prove that MPI approach outperforms OpenMP in such computationally-intensive, memory-intensive, and communication-less problem.

Keywords: Bioinformatics, Heap Contention, Message Passing Interface (MPI), Motif Finding, Open Multi Processing (OpenMP), Parallel Programming

INTRODUCTION

Biologists are interested in understanding gene interaction and gene regulation. They have determined that special proteins known as transcription factors are controlled when genes are turned on and off. Transcription factors bind to relatively short sequences in the region surrounding a gene which are called transcription factors binding sites (TFBS) or simply motifs. As a result, finding these motifs is a very important task for decoding a genome (Gopal, Haake, Jones & Tymann, 2009).

The literature on the topic of DNA motif finding is extensive. Specialized surveys for existing motif finders are documented (Das & Dai, 2007; Sandve & Drabløs, 2006). In brief words, the algorithms fall into two major groups: word-based algorithms which represent the motif as string, and probabilistic model algo-
rithms witch represent the motif as a Position Weight Matrix (PWM). Word-based algorithms include enumeration methods like Brute Force (Jones & Pevzner, 2004), FLAME (Floratou, Tata & Patel, 2010) (that uses suffix tree), and MITRA (Eskin & Pevzner, 2002) (that uses mismatch tree). Graph algorithms are among other implementation of word-based approach, WINNOWER (Pevzner & Sze, 2000) is an example. Many algorithms of this approach can guarantee finding the optimal solution and are considered exact algorithms. On the other hand, most of the probabilistic model algorithms fall under the approximate category because they employ local search techniques. Probabilistic model algorithms include expectation maximization like MEME (Bailey, Williams, Misleh, & Li, 2006), and Gibbs Sampling like Gibbs Sampler (Lawrence et al., 1993).

The goal of motif finding is to find an unknown motif sequence with approximate occurrences at unknown positions in a sample of DNA sequences. This makes motif finding considered as a computationally intensive problem. Parallel computation reduces the execution time by running multiple processes concurrently. Many of motif finding algorithms are parallelized using different approaches. Special hardware devices like VHDL, FPGAs, GPUs, and COPACOBANAs are efficiently utilized to solve the motif finding problem (Faheem, 2010(AIA4); Schröder, Wienbrandt, Pfeiffer & Schimmker, 2008; Yu & Xu, 2009). In fact, these solutions require special hardware and specialists who are able to design such systems.

Now, the era of Multi-core architecture has begun. Parallel programming is introduced to solve computationally intensive problem like motif finding by running multiple threads concurrently on the available cores. ParaMEME (Grundy, Bailey & Elkan, 1996) is a parallel version of MEME algorithm that provides efficient performance when it is ported to San Diego Supercomputer Center’s 400-node Intel Paragon XP/S and Pittsburgh Supercomputer Center’s T3D. This architecture development encourages us to parallelize our efficient exact algorithm called Recursive-Brute Force (R-BF) in order to benefit from this technology.

While R-BF does not enhance the worst case of the brute force approach, the average case complexity reflects the performance enhancement that achieved by the R-BF. It requires time that exponential with the allowed mutations (d) rather than the motif length (l) like the previous versions of the Brute Force. We consider two different paradigms for multi-core programming, OpenMP (http://www.openmp.org) and MPI (http://www.mpi-forum.org). While MPI is normally used on large distributed computing systems, it can be also used on a multicore architecture. Many researches try to answer the question: How does the performance of MPI compare to OpenMP on a multicore system? (Eadline, 2007; Jost, Jin, an Mey & Hatay, 2003; Mallón et al., 2009).

In this paper, we implement two parallel versions of Recursive-Brute Force algorithm. We call the first OpenMP-Recursive-Brute-Force (OMP-RBF). The second version is called MPI-Recursive-Brute-Force (MPI-RBF). The advantage of OpenMP is that an existing code can be easily parallelized by placing OpenMP directives around time consuming loops which do not contain data dependences, leaving the source code unchanged. However, OMP-RBF faces the heap contention problem because its threads concurrently invoke the memory allocator to allocate and free large numbers of objects. The result is that instead of the expected linear performance increase, adding CPUs leads to performance degradation. Two solutions to this problem are presented. The first is to create local heap for each thread. The second is using an efficient multi-heap memory allocator Hoard (Berger, McKinley, Blumofe & Wilson, 2000). On the other hand, MPI-RBF scales well with the number of processors. However, to write an MPI code we need to work on the whole code, distributing the data and the work completely.

The rest of this paper is organized as follows: In the next section we describe the motif finding problem and the Brute Force algorithm. Then, a brief explanation of the Recursive-Brute Force algorithm and its complexity analysis are