Maximizing Occupancy of GPU for Fast Scanning Biological Database Using Sequence Alignment

Mohamed Issa, Ahmed Helmi, Hitham Abo Bakr, Ibrahim Ziedan, Ahmed Alzohairy

Computer and Systems Engineering Department, Faculty of Engineering, Zagazig University, Zagazig, Egypt.
Genetics Department, Faculty of Agriculture, Zagazig University, Zagazig, Egypt.

Received 28 February 2017; Accepted 2 June 2017; Available online 10 July 2017

ABSTRACT
Background: Sequence Alignment is a main task in a lot of bioinformatics applications and it is used for aligning residues/nucleotides (for protein/DNA) to measure the similarity between two biological sequences and locates the regions of matching. The alignment operation consumes huge execution time especially for long sequences and so parallelization of it is mandatory. Objectives: Maximization of occupancy of Graphical Processing Unit is an important issue for increase performance efficiency. In this paper effect of maximizing occupancy was tested through sequence alignment application for searching biological databases for finding similar sequences. Results: The application that was used for testing is scanning biological databases (SWISS-PORT) contains around 300000 proteins. The objective of the application is finding the sequence that share longest common substring with a query sequence based on local sequence alignment. The application was implemented on Graphical Processing Unit for parallel execution. A comparison between the serial execution time and parallel execution time was proposed which shows the huge speed up of using Graphical Processing Unit over serial execution time by maximum speed up around 140. The occupancy of Graphical Processing Unit was tested against modification number of threads used per block in the programming model. The tests shows that increasing the number of threads per block will maximize the occupancy and so decreasing the execution time. Conclusion: Maximization of occupancy of Graphical Processing Unit will lead to increase in speed up of application execution. Hence it is important to choose the values of parameters that maximize the occupancy such as number of threads per block, number of registers and shared memory sizes allocated for each thread.

KEYWORDS: Sequence Alignment; Graphical Processing Unit; Occupancy;

INTRODUCTION

The new high-throughput biotechnologies are providing us with unique opportunities to study the complex evolutionary histories of large sets of genes. For the first time, it will be possible to compare sequences from hundreds of diverse organisms, both present and extinct, in order to reconstruct the dynamics of evolutionary change at the molecular level. In this context, a number of studies have focused on the origin of new genes with novel functions, which are thought to be a major contributor to the evolution of lineage specific phenotypes and
adaptive innovation (reviewed in [1]). Such studies have revealed the importance of specific molecular mechanisms, including gene duplication, exon shuffling, retro-transposition, mobile elements, lateral gene transfer, gene fusion/fission, as well as de novo origination. A major task is exploring the origin of modified DNA or protein sequences in the genomes of various organisms [2]. Sequence alignment (SA) is the operation of measuring the functional and structure similarity between biological sequences (DNA, RNA and proteins) [3,4]. Hence, the main usage of SA is finding the origin of unknown biological sequences or modified ones by finding the most similar sequence in biological databases. In addition, SA is used in a lot of biological operations such as phylogenetic trees construction [5], protein secondary structure prediction and analysis [6] and DNA fragment assembly [7]. SA has quadratic time complexity so it consumes huge execution time for longer lengths or scanning big biological databases. A lot of parallelization techniques were used for speed up execution of SA using CPU Multiprocessors [8], FPGA [9] and Graphical Processing Unit (GPU) [2, 10-11].

GPU nowadays is available in most machines and mobiles so development of applications on GPU is a vital process. Hence this paper tends to develop a speed application of scanning biological databases for finding a sequence that has longest common subsequence with a query sequence. This operation mainly depend on sequence alignment algorithm and the development of an application on a GPU depend on maximizing occupancy of multiprocessors of GPU. Occupancy is a measurable key for keeping multiprocessors busy as possible and increasing the occupancy yields reducing the execution time.

The rest of paper is organized as follow: Section 2 explains the sequence alignment operation, section 3 gives introduction to GPU. Section 4 explain the main parameters affects on occupancy of GPU and section 5 shows the results of increasing occupancy on the speeding up. Finally, conclusion summarizes the main objectives and results of the paper.

2. Sequence Alignment:

The operation of sequence alignment aimed to align the nucleotides/ amino acids of DNA/proteins by inserting a gaps that shift the matching bases to increase the similarity between two sequences. As shown in Fig.1, the first two rows are the aligned sequences and the third row represents columns that are similar. The overall similarity is measured based on alignment score between each two bases which is based on matching, un-matching and gaps scores. A simple scoring scheme is a positive score for matching and negative one for un-matching while gaps are penalized with negative scores. This scheme for gaps penalty is called constant gap penalty. Furthermore, it penalized each gap separately which decreases the overall alignment scheme. Another gap penalty is called affine gap [12] which is penalize the contiguous gaps with a score smaller than the separated ones. So, for a contiguous gaps with length \( I \) the gap penalty is \( g_a + (I-1) \cdot g_e \), where \( g_a \) is the open gap penalty for first gap and \( g_e \) is the extended gap penalty for each gap after the first one. Table 1 shows various gap penalty schemes. Another scoring schemes for bases comparisons are used for proteins such as BLOcked SUBstitution Matrix (BLOSUM) [13] and Point Accepted Mutation (PAM) [14]. Sequence alignment is used to measure the similarity between two biological sequences and in this case is called pairwise alignment while for similarity between more than two sequences is called multiple sequence alignment [15]. Global alignment and local alignment are the basic types of sequence alignment [15]. Global alignment is used for aligning the entire lengths of sequences while local alignment is used to find the matching region between sequences.

\[
\begin{array}{cccccccc}
A & T & T & C & T & G & A & T \\
C & T & T & A & C & C & G & A & T & G
\end{array}
\]

Fig.1: Sequence alignment between two biological sequences.

<table>
<thead>
<tr>
<th>Gap penalty</th>
<th>Formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear gap penalty</td>
<td>( P = a^k )</td>
</tr>
<tr>
<td>Affine gap penalty</td>
<td>( P = a^k + b \cdot \log k + c )</td>
</tr>
<tr>
<td>Logarithmic gap penalty</td>
<td>( P = b \cdot \log k + c )</td>
</tr>
<tr>
<td>Logarithmic-affine gap penalty</td>
<td>( P = a^k + b^k \cdot \log k + c )</td>
</tr>
</tbody>
</table>

Where \( k \) is the number of successive gaps and \( a, b \) and \( c \) are constants. Two basic algorithms were developed for global and local alignment are based on dynamic programming (DP) approach were Needleman-Wunsch alignment [16] and Smith-Waterman alignment [17] respectively. Smith-Waterman alignment algorithm was focused in this work due to it mainly is used for finding the longest common substring between two biological sequences.

Smith-Waterman alignment based on constructing a two dimension matrix (Score) column for one of sequences and the rows for the other. DP alignment algorithm tries to find all possible scores of alignment between two sequences based on the suitable scoring scheme. The score of each possible alignment \( f(x, y) \)
computed using Eqn.1 based on similarity between the aligned residues. In addition linear gap penalty is used for penalize consecutive gaps instead of separated gaps which decreases the alignment score.

\[
\text{Score}(i, j) = \max \begin{cases} 
\text{Score}(i - 1, j - 1) + \text{Similarity}(\text{Seq}_A(i), \text{Seq}_B(j)) \\
\max_{k=1}^{i-1} (\text{Score}(i, k) + g_0 + kg_g) \\
\max_{j=1}^{j-1} (\text{Score}(k, j) + g_0 + kg_g) \\
0 
\end{cases}
\]

(1)

where SeqA and SeqB are the sequences to be aligned, i and j denote the row and column indices respectively, 1 < i < length(SeqA) and 1 < j < length(SeqB). Similarity ( ) is a function that computes the similarity between two bases according to the used scoring scheme. The optimal diagonal path for tracing back is starting from the cell contains maximum score until the first cell contain zero score [18-19]. The time and space complexity of DP alignment is \(O(n^3)\) where \(n\) is the maximum of lengths of the sequences to be aligned. So, the DP alignment algorithm executes huge time for sequences with longer length.

The application to be speeded up is scanning large biological databases to find the DNA/protein sequence that share the longest common substring with an input sequence using local sequence alignment algorithm [17]. That application helps in finding origin of unknown sequences and determining common features between sequences. Simply, the procedure of operation by aligning the query sequence with each sequence in database. The time complexity of serial execution of this application is \(O(m*n*z)\) where \(n\) is the length of query sequence, \(m\) is the length of maximum sequence length in database and \(z\) is the count of sequences in database. This complexity indicate huge increasing of execution time for longer lengths of sequences or increasing number of sequences in databases.

3- Graphical Processing Unit (GPU):

Originally, GPU was designed to speed up the computation of Graphical operations. The key idea of GPU is using hundred or thousands of arithmetic logic units (ALUs) and large space memory with high bandwidth. The difference between CPU and GPU is the number of computation ALU of GPU over CPU with small memory sizes than CPU as shown in Fig.2. In addition, Fig.3 shows the evolution of FLoating-point Operations Per Second (FLOPS) where Number of floating point per operation in GPUs is higher than that in CPUs. GPUs were originally designed to accelerate computer Graphics algorithms. However, their high computational capabilities and their highly parallel structure opened up to them a wide a range of other fields, like scientific computing [20], computational geometry [21] and bioinformatics [22].

GPU execution model consists of a grid of blocks and each block consists of a group of threads that is as shown in Fig.4. The thread executes functions (kernels) on many processors in parallel. GPU’s threads are light weight than that of CPU and the scheduling or launching a core on the GPU is faster also. On GPU a hierarchy of memories are available for programmers to be used such as registers, local memory, shared memory, global memory, constant memory and texture memory. Shared memory and registers are accessible by the threads directly and are the fastest access memories but with limited sizes. Other memories are exist on the main random access memory of the GPU. Fig.5 shows the layout architecture of the memory with number of cycles for access by multiprocessors.

Fig.2: CPU organization versus GPU one [23]
Fig. 3: FLOPS for CPU and GPU [23]

Fig. 4: GPU programming structure

Fig. 5: Memory Architecture Layout
4- Maximizing GPU Occupancy:

The massive multi-threading of GPUs delivers high performance via single instruction multiple thread execution model. As shown in Fig. 4 each block consists of a group of threads and the block is executed on one of the streaming multiprocessors (SM) of GPU. During the execution the block is divided into warps where it is a group of threads typically 32 threads and it is the smallest execution unit. Warp is active as at least one of its threads is run on SM and there is a maximum of concurrent run threads on the same multiprocessors as shown in Fig. 6. So, occupancy is defined as the ratio of number of active warps per SM to the maximum allowable warps on SM. The active warps is computed as multiplication of number of active blocks per SM and number of active warps per SM. The number of active blocks per SM is determined as minimum of the following:

- Physical limit of blocks per SM (depend on GPU model).
- Number of blocks per SM determined by number of registers allocated to a block divided by number of registers allocated to SM.
- Number of blocks per SM determined by shared memory size allocated by a block divided by size of shared memory allocated to SM.

Number of active warps per SM is determined as division of number of threads per block on the warp size which is determined based on GPU compute capability [23]. So, using the minimum amount of registers and shared memory will increase number of active blocks per SM. In addition, increasing the number of threads per blocks will increase number of active warps per SM which increases the occupancy of SM.

![Fig. 6: Representation of division threads into warps for execution on Multi-Processors](image)

5- Results:

This section show the effect of increasing the occupancy by increasing number of threads per block on speed of scanning biological database for measuring similarity using local sequence alignment. The parallel scheme of this application was performed in two levels: (1) parallelization of scanning many sequences at the same time. (2) is parallelization of pairwise alignment between input sequence and each sequence in database by computing the cells of computation matrix in the same diagonal in parallel. That is due to these cells in the same diagonal independent on each other. The procedure of maximizing occupancy of GPU occurs by adjusting the number of registers and size of shared memory allocated to block to be minimum as possible. The number of threads per block was adjusted and the corresponding occupancy was measured and its effect noted through measuring execution time. Swiss-port protein database containing 3000000 sequences was used in the test. The test was done on different length of query input to show its effect on the execution time against varying the length of query sequence with 64 threads allocated to each block. The test was implemented on a GPU Tesla C2075 with 448 CUDA cores (Each core is 1.15 GHZ) and total memory 6 GByte. The CPU is core i3 multiprocessor each core is 2.27 GHZ and with main memory is 4 GByte. The serial and parallel execution time are compared as shown in Table 2. The occupancy of SM for allocating 64 threads per block is 33 % for query sequence with length 1024 amino acids. The effect of increasing number of threads per block and fixing the number of registers and shared memory size per block on occupancy is as shown in Table 2. The increasing of threads must be 32 threads which is the warp size allocated by Nvidia company [23]. As shown increasing number of threads per block increasing the occupancy and so decreasing the execution time.
Table 2: Serial execution time versus Parallel execution time

<table>
<thead>
<tr>
<th>Query Length</th>
<th>GPU Time (Sec)</th>
<th>CPU Time (Sec)</th>
<th>Speed up Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>1.5</td>
<td>120</td>
<td>80</td>
</tr>
<tr>
<td>16</td>
<td>3</td>
<td>240</td>
<td>80</td>
</tr>
<tr>
<td>32</td>
<td>6</td>
<td>540</td>
<td>90</td>
</tr>
<tr>
<td>64</td>
<td>13</td>
<td>1050</td>
<td>80.7</td>
</tr>
<tr>
<td>128</td>
<td>24</td>
<td>2820</td>
<td>117.5</td>
</tr>
<tr>
<td>256</td>
<td>47</td>
<td>6570</td>
<td>140</td>
</tr>
<tr>
<td>512</td>
<td>97.5</td>
<td>10800</td>
<td>111</td>
</tr>
<tr>
<td>768</td>
<td>145.5</td>
<td>17400</td>
<td>119.5</td>
</tr>
<tr>
<td>1024</td>
<td>189</td>
<td>23040</td>
<td>118</td>
</tr>
</tbody>
</table>

Table 3: Effect of number of threads on occupancy

<table>
<thead>
<tr>
<th>Number of threads per block</th>
<th>Occupancy</th>
<th>Time (Sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>64</td>
<td>33%</td>
<td>189</td>
</tr>
<tr>
<td>96</td>
<td>50%</td>
<td>185</td>
</tr>
<tr>
<td>128</td>
<td>67%</td>
<td>177</td>
</tr>
<tr>
<td>160</td>
<td>83%</td>
<td>174</td>
</tr>
<tr>
<td>192</td>
<td>100%</td>
<td>172</td>
</tr>
</tbody>
</table>

Conclusion and Future work:

This paper proposed analysis of the effect of increasing occupancy on speed up execution of applications on GPU. The occupancy of GPU is affected by three parameters: number of threads per block, number of registered allocated by block and size of shared memory per block. This paper focus on the effect of increasing occupancy by increasing number of threads per block on the performance efficiency on real application. The application that was scanning biological database for finding most similar sequence that share longest common substring with a query sequence using local sequence alignment algorithm. Local sequence alignment has a huge time complexity that is proportional with cubic of length of aligned sequences. A comparison between serial and parallel executions of the application was tested. The occupancy was maximized by increasing the number of threads and execution time of application was measured. The summarization of test is increasing the number of threads per block will lead to increasing the occupancy and decreasing the execution time.

In the future, metaheuristics algorithms that based on using strategy mimics its behavior from nature will be used to choose the best values of the parameters that affect on maximizing occupancy. These algorithms will be used since it is difficult to choose the best values of parameters using try and error.

REFERENCES