TELKOMNIKA Indonesian Journal of Electrical Engineering
Vol. 12, No. 12, December 2014, pp. 8278 ~ 8285
DOI: 10.11591/telkomnika.v12i12.6572

Accelerating Computation of DNA Multiple Sequence Alignment in Distributed Environment

Ramdan Satra*[1,2], Wisnu Ananta Kusuma[1], Heru Sukoco[1]
[1]Department of Computer Science, Bogor Agricultural University
Kampus IPB Dermaga, Jl. Meranti, Wing 20 Level 5-6, Bogor 16680, Indonesia
Telp./Fax.: +62-251-8625584
[2]Department of Informatics Engineering, University of Muslim Indonesia, Makassar, Indonesia
*Corresponding author, e-mail: ramdanstr@gmail.com[1], ananta@ipb.ac.id[2], hsrkom@ipb.ac.id[2]

Abstract

Multiple sequence alignment (MSA) is a technique for finding similarity in many sequences. This technique is very important to support many Bioinformatics task such as identifying Single Nucleotide Polymorphism (SNP), creating phylogenetic tree, and metagenome fragments binning. The simplest algorithm in MSA is Star Algorithm. This algorithm consists of aligning all possible pairs of sequences, finding a sequence Star chosen from sequence that has maximum alignment score, and aligning all sequences refered to the sequence Star. Each of pairwise alignments is conducted using dynamic programming technique. The complexity of DNA multiple sequence alignment using dynamic programming technique is very high. The computation time is increased exponentially due to the increasing of the number and the length of DNA sequences. This research aims to accelerate computation of Star Mutiple Sequence Alignment using Message Passing Interfaces (MPI). The performance of the proposed method was evaluated by calculating speedup. Experiment was conducted using 64 sequences of 800 bp Glycine-max-chromosome-9-BBI fragments yielded by randomly cut from reference sequence of Glycine-max-chromosome-9-BBI taken from NCBI (National Center for Biotechnology Information). The results showed that the proposed technique could obtain speed up three times using five computers when aligning 64 sequences of Glycine-max-chromosome-9-BBI fragments. Moreover, the increasing of the number of computers would significantly increase speed up of the proposed method.

Keywords: *DNA multiple sequence alignment, distributed computing, star algorithm, message passing interface*

1. Introduction

Multiple Sequence Alignment (MSA) is a technique for aligning multiple biological sequences [1] to find similarities and differences in the sequence [2]. MSA is used in phylogenetic analysis to assessing the origin of species. MSA is also used to search a single nucleotide polymorphism (SNP) for molecular-based plant breeding genetics [3]. The growth of biological sequence databases makes the sequence alignment computation increase significantly [1].

The complexity of sequence alignment is O (LN), where L is the length of each sequence and N is the number of sequences [4]. The longer and the more sequences to be aligned the longer it's computational time. The optimal time of sequence alignment is increased exponentially with increaseing of the number and length of sequences [5].

Methods or tools are needed to obtain optimal computational process. Previous study to optimize the computational sequence alignment used a dynamic programming algorithm with complexity of *O (mn)* by Zhou and Chen [6]. Other studies used a linear space algorithm for pairwise sequence alignment with the complexity of *O (n)* [7, 8]. Others used parallel computing with MPI [9] and GPU-CUDA [1, 3]. Parallel computing used to propose new clssification algorithm [12] and simulate properties of polymer chain [13] in other case of computational problem.

In this research, parallel computing is utilized on clusters of computers known as Distributed Memory. This research tries to improve a previous work conducted by Sunarto which used star algorithms in parallel environment using MPI [9]. The previous research had limitation in scalability. In this research, we proposed a scallable distributed computing of DNA

Received July 4, 2014; Revised September 21, 2014; Accepted October 12, 2014

Multiple Sequence Alignment. The new approach could handle more number of sequences and was also easily extended by increasing the number of resources used to compute multiple sequence alignment.

2. Research Method

2.1. Research Data

In this research, we used the sequence of *Glycine-max-chromosome-9-BBI* obtained from NCBI (*National Center for Biotechnology Information*) [10]. Detailed research data is shown in Table 1.

Table 1. Research data

Gene Name	Length (bp)	Number
Glycine-max-chromosome-9-BBI	811-850	64

2.2. Multiple Sequence Alignment (MSA) with Star Algorithm

Algorithms for MSA are actually developed based on probabilistic or heuristic approaches such as the Star method and the Progressive method [3]. This research used the Star method by improving and extending theprevious research conducted by Sunarto AA *et al*. In the Star alignment algorithm, there are 3 stages, including (i) calculating pairwise alignment scores from all possible sequence pairs, (ii) selecting a Star sequence which has the best alignment score, and (iii) aligning the Star sequence with other sequences [3, 9]. The illustration of multiple sequence alignment using the Star algorithm can be described as follows:

For example, supposed we have DNA sequence data as follows:

Seq 1 = ATGG
Seq 2 = ATGC
Seq 3 = ATCC
Seq 4 = AGCG

The first stage is calculating pairwise similarity scores from all possible sequences pairs. This stage has $O(n^2)$ complexity where *n* is number of sequences. This stage begins by calculating the number of sequence pairs using Equation (1):

$$(n-1) + (n-2) + \ldots + 1 \text{ or } n(n-1)/2 \quad (1)$$

Where *n* is number of sequences.

The next step is creating identity matrix containing the sequences, symbolized by Kn. Figure 1 shows the identity matrix.

	Seq 1	Seq 2	Seq 3	Seq 4
Seq 1		K1	K2	K3
Seq 2	K1		K4	K5
Seq 3	K2	K4		K6
Seq 4	K3	K5	K6	

Figure 1. Pairwise sequence identity matrix

The results of the determination of DNA pairwise sequence are:

K1 = Seq 1 (ATGG) and Seq 2 (ATGC)
K2 = Seq 1 (ATGG) and Seq 3 (ATCC)
K3 = Seq 1 (ATGG) and Seq 4 (AGCG)
K4 = Seq 2 (ATGC) and Seq 3 (ATCC)
K5 = Seq 2 (ATGC) and Seq 4 (AGCG)
K6 = Seq 3 (ATCC) and Seq 4 (AGCG)

Each pair of sequences above is aligned using Needleman Wunsch algorithm. This algorithm uses dynamic programming approach to find global similarity of sequences. Needleman Wunsch algorithm consists of matrix initialization, filling the value of each cell matrix and tracing back. Similarity score of pairwise sequence is calculated using Equation (1) below:

$$A_{[mn]} = Max \begin{cases} A_{m-1,n-1} + S_{m,n} \\ A_{m,n-1} + W \\ A_{m-1,n} + W \\ 0 \end{cases} \quad (1)$$

Where:

- *m* is the row and *n* is the collumn.
- *A* is a matrix of values for each cell is expressed as ($A_{[m,n]}$)
- *S* is the score of each cell is expressed as ($S_{[m,n]}$), match score = 5 and unmatch score = -3
- *W* expressed as gap alignment with value = - 4

The calculation results of similarity scores for all pairs of sequences are shown in Figure 2.

0	Seq 1	Seq 2	Seq 3	Seq 4
Seq 1	0	12	4	7
Seq 2	12	0	12	7
Seq 3	4	12	0	4
Seq 4	7	7	4	0

Figure 2. Sequence alignment similarity scores

The second stage of star algorithm is selecting a Star sequence. A Star sequence was selected by comparing the sequence alignment similiartity scores of all sequences. A sequence with the best similarity score was chosen as a Star sequence. Next, the Star sequence was updated by aligning it to the other sequences. The complexity of this stage is *O(n)*, where *n* is the number of sequences. Illustration of this stage is shown in Figure 3.

0	Seq 1	Seq 2	Seq 3	Seq 4	Accumulated Similarity Scores
Seq 1	0	12	4	7	23
* Seq 2	12	0	12	7	31
Seq 3	4	12	0	4	20
Seq 4	7	7	4	0	18

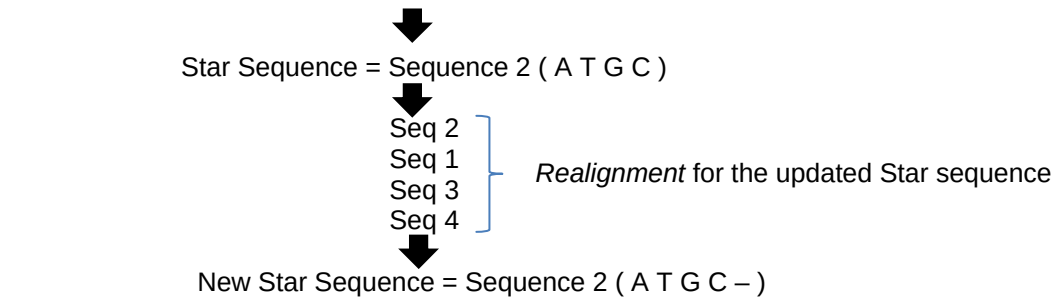

Figure 3. Selecting a Star sequence

The third stage of star algorithm is realignment. This stage is realigning the Star sequence to others sequences by using dynamic programming technique. Complexity is *O(n),* where *n* is the amount of sequences. Illustration for realignment is shown in Figure 4.

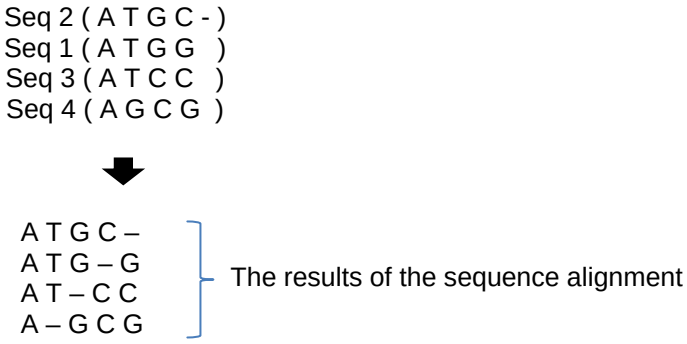

Figure 4. Realignment

2.3 Parallelization MSA with Message Passsing Interfaces (MPI)

In this research, we used Foster's Methodology for implementing parallel programming. This methodology consists of partitioning, communication, agglomeration and mapping stage [8]. Parallelization was conducted for computing pairwise similarity scores. The calculation began by dividing the sequence pairs into multiple processes (threads) symbolized by *K1, K2 ... Kn*. In the case when the amount of computers is equal to the amount of processes, each process will be allocated to each computer. If *Pk* is a computer for *k* = 1,2,...n this allocation can be defined as *P1 = K1, P2 = K2 ... Pn = Kn*. Unfortunately the number of sequence pairs is often larger than the number of computers. Sequence pairs which has not been allocated yet must wait until the previous process has been completed. Allocation of processes (threads) when the number of sequence pairs are more than the number of computers or when *Kn > Pn* can be described in Figure 5.

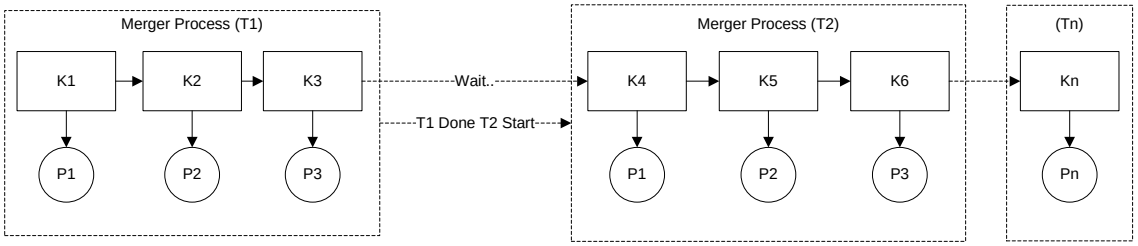

Figure 5. The distribution of processes when the number of sequence pairs are more than the number of computers or *Kn > Pn*

Illustration of data distribution of DNA sequence with MPI can be seen in Figure 6. In this research, comunication of MPI used point to point communication with a blocking send and receives operations. Parallelization scheme for MSA is to assign a computer as the data divider and other computers as data processors. A data divider called *rank 0* distributed sequence pairs using *MPI_Send()* to other computer as data processors (Figure 7). Data processors which were symbolized by the *rank 1...n* received sequence pairs using *MPI_Recv()* (Figure 8). Next, each data processors conducted pairwise sequence alignment to compute pairwise similarity scores in parallel. The calculation results of each data processor were transmitted to data divider (*rank 0)* by using *MPI_Send()*. Data divider with *rank 0* received the similarity scores using *MPI_Recv()* command and completing the MSA process by selecting a Star sequence and realigning all sequence to the Star sequence

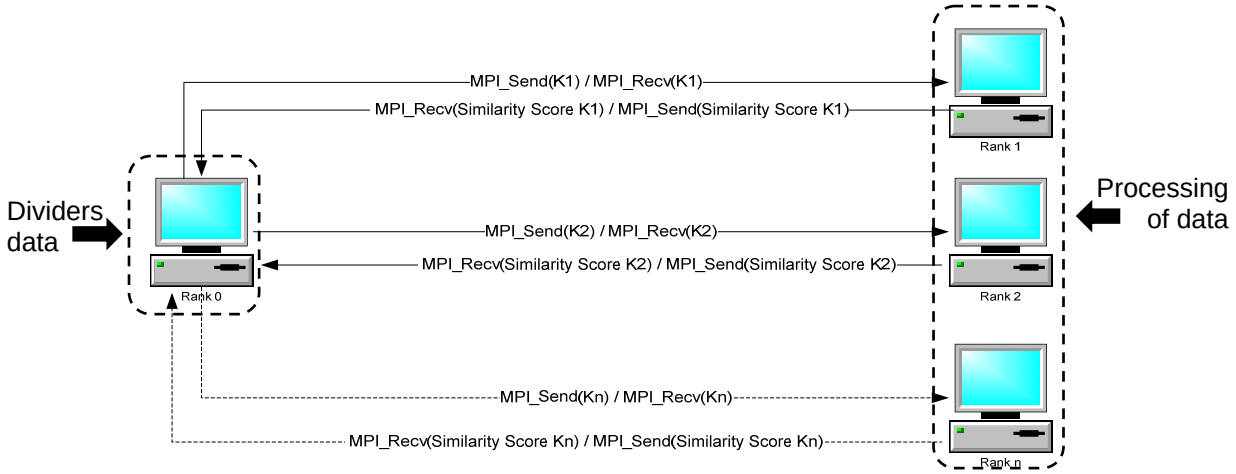

Figure 6. Illustration the distribution of sequence data with MPI

```
//MPI initialization
MPI_Init(&argc, &argv);
MPI_Comm_rank (MPI_COMM_WORLD, &rank);//Computer initialization
MPI_Comm_size (MPI_COMM_WORLD, &p_size);//Initialize the number of computers

If rank == 0 {
  c = ((js*js)-js)/2; // Calculate the many pairwise sequences
  p_bts = p_size-1; // Limit Number of Computers
  p=1; // Limit Number of computers for processing data (other than rank 0)
  for (i=1; i<=js-1; i++){
    for (j=i+1; j<=js; j++){
      us1 = i-1; // Initialization the order of sending data sequence
      us2 = j-1; // Initialization the order of sending data sequence
      MPI_Send (&c, 1, MPI_INT, p, 1, MPI_COMM_WORLD);
      MPI_Send (&SIZE, 1, MPI_INT, p, 3, MPI_COMM_WORLD);
      MPI_Send (&p_bts, 1, MPI_INT, p, 5, MPI_COMM_WORLD);
      MPI_Send (&us1, 1, MPI_INT, p, 6, MPI_COMM_WORLD);
      MPI_Send (&us2, 1, MPI_INT, p, 7, MPI_COMM_WORLD);
      MPI_Send (s[us1], SIZE+1, MPI_CHAR, p, 8, MPI_COMM_WORLD);
      MPI_Send (s[us2], SIZE+1, MPI_CHAR, p, 9, MPI_COMM_WORLD);
      p++;
if (p>p_bts){p=1;} // Looping sequence data sending besides to rank 0
     }
   }
} // Tutup rank 0
```

Figure 7. Source code data divider

```
If rank <> 0 {
  MPI_Recv(&c, 1, MPI_INT, 0, 1, MPI_COMM_WORLD, &status);
  MPI_Recv(&SIZE, 1, MPI_INT, 0, 3, MPI_COMM_WORLD, &status);
  MPI_Recv(&p_bts, 1, MPI_INT, 0, 5, MPI_COMM_WORLD, &status);
  p=1;
  for (h=0; h<=c-1; h++){
  if (rank==p){
    MPI_Recv(&us1, 1, MPI_INT, 0, 6, MPI_COMM_WORLD, &status);
    MPI_Recv(&us2, 1, MPI_INT, 0, 7, MPI_COMM_WORLD, &status);
    MPI_Recv(s[us1], SIZE+1, MPI_CHAR, 0, 8, MPI_COMM_WORLD, &status);
    MPI_Recv(s[us2], SIZE+1, MPI_CHAR, 0, 9, MPI_COMM_WORLD, &status);
  }
} //end rank <> 0
```

Figure 8. Source code data processors

3. Results and Analysis

This research used 5 computers with spesification Intel Core CPU i3-3220@3.30GHz with 4 CPU cores, 8 GB of RAM memory. A whole genome data with FASTA format of *Glycine-max-chromosome-9-BBI* was divided randomly into some numbers of sequences such as 3, 4, 8, 12, 14, 16, 20, 24, 28, 32, 36, 40, 44, 48, 52, 56, 60 and 64 sequence. Testing evaluation was conducted in 3, 4 and 5 computers.

3.1. Results

The evaluation results was described in Figure 9-11. Figure 9 showed the performance of our proposed method in term of execution time.

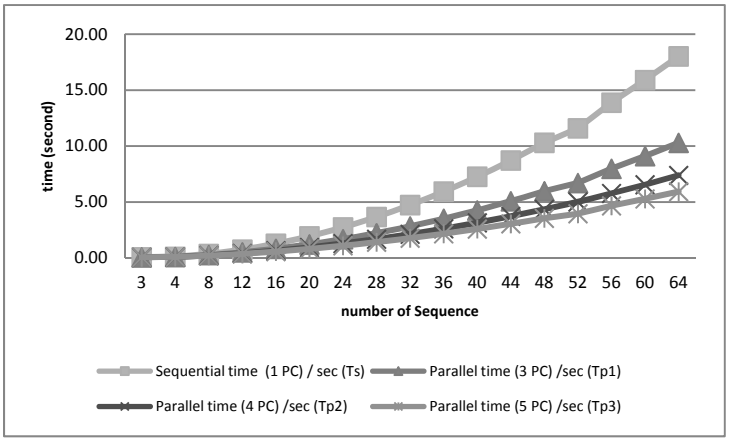

Figure 9. the execution time in various number of sequences

The result of sequence alignment computation showed that the parallel computation time was faster than the sequential computation time. Figure 9 showed the difference in computation time of data FASTA Glycine-max-chromosome-9-BBI with the length of 811-850 basepair. Using 3, 4 and 8 sequences the computational time was only slightly different. However, with the incresing of the number of sequences the computational time was significantly different.

3.2. Analysis

The performance analysis or evaluation of this proposed method was conducted by calculating speedup [11]. Speedup can be obtained by using Equation (2).

$$\text{Speedup} = \frac{Ts}{Tp} \quad (2)$$

Description:
Ts = Sequential execution time
Tp = Parallel execution time

The results of the parallel computing with 3, 4 and 5 PC (Personal Computer) showed that increasing of the number of computers (PCs) affected the parallel execution time. Parallel computational speedup increased as the number of computers was increased. Plot of parallel computing speedup for 3, 4 and 5 PCs were shown in Figure 10.

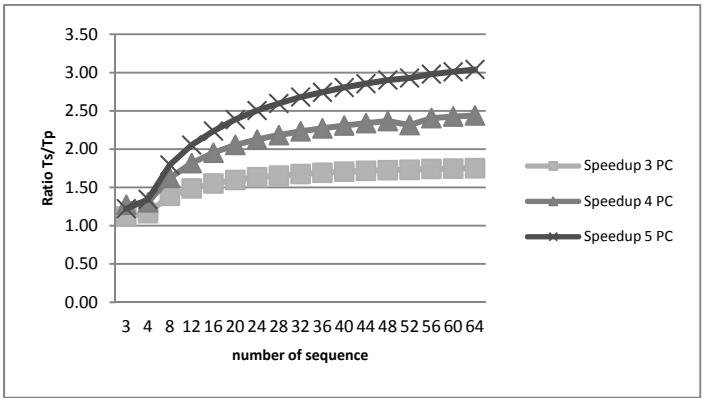

Figure 10. The speedup of parallel computing in various number of sequences

The other metric for showing the parallel performance is efficiency. Increasing the number of processors decreased the value of efficiency, and conversely an increase in the size of data will increase the efficiency [14]. The constant efficiency value indicates that the performance of a parallel system is scalable to the size of the data. Scalable parallel systems means the system is able to maintain performance with increasing number of processors [14] to process the data in a certain size. Efficiency is calculated using Equation (4). The results of efficiency calcucaltion in this study can be seen in Figure 11.

$$\text{Efficiency} = \frac{S(n)}{P} \quad (4)$$

Limitation efficiency value is 1 / p < efficiency <1

Description:
S (n) = Speedup of parallel computing
p = Many processors

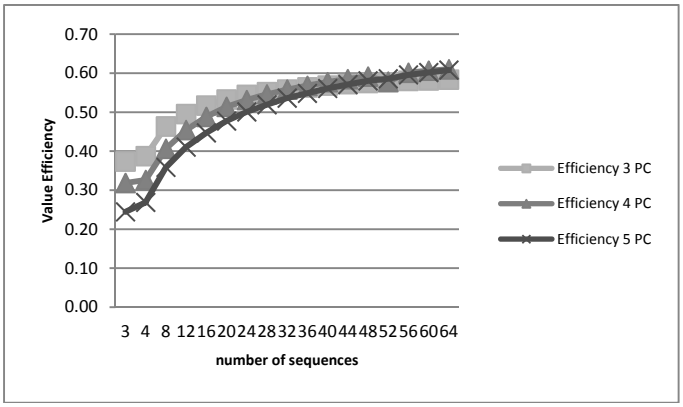

Figure 11. Efficiency of parallel computing in various number of sequences

Figure 11 showed that the use of 3, 4 and 5 processor were scalable in processing data with the length of 811-850 basepair and the number of sequence of 32-64. This efficiency will increase when the size and the length of sequence increased with increasing of the number of processor.

4. Conclusion

Parallel computing using MPI is recommended for MSA. This study show that the speed up of using MPI in distributed environment for conducting MSA was incresed by increasing the size and the number of DNA sequences aligned. The trend of efficiency was almost constant when using 56 of sequences or more. This tendency indicated that this method was scalable to the size of data.

Future research can be conducted by using the larger size of data sequence and increase the number of processors to improve the speedup and efficiency of parallel computing for MSA. In this research, pairwise sequence alignment was conducted in sequential computation. In future, we could conduct hybrid computing by conducting the paiwise sequence alignment in parallel using MPI and CUDA GPU to increase speed up significatly.

.

Acknowledgements

The authors would like to thank Indonesia Ministry of Agriculture for funding this research through the KKP3N 2014 program.

References

[1] Liu Y, Schmidt B, Maskell DL. MSA-CUDA. Multiple Sequence Alignment on Graphics Processing Units with CUDA. *IEEE International Conference on Application-specific Systems, Architectures and Processors*. 2009: 121-128.
[2] Junior SAC. Sequence Alignment Algorithms. *Department of Computer Science School of Physical Sciences & Engineering Kingís College London*. 2003.
[3] Sujiwo MAP, Kusuma WA. Multiple Sequence Alignment with Star Method in Graphical Processing Unit using CUDA. *International Seminar on Science (ISS)*. Bogor. 2013: 359-363.
[4] Lloyd GS. 2010. Parallel Multiple Sequence Alignment: An Overview.
[5] Edgar RC, Batzoglou S. Multiple sequence alignment. *Current opinion in structural biology*. 2006; 16(3): 368-373.
[6] Zhou Zm, Chen Z-w. Dynamic Programming for Protein Sequence Alignment. *International Journal of Bio-Science and Bio-Technology*. 2013; 5(2): 141-150.
[7] Myers EW, Miller W. Optimal alignments in linear space. *Oxford Univ Pres*. 1988: 1-13.
[8] Sandes EFO, de Melo ACMA. Smith-Waterman Alignment of Huge Sequences with GPU in Linear Space. *IEEE International Parallel & Distributed Processing Symposium*. 2011: 1199-1211.
[9] Sunarto AA, Kusuma WA, Sukoco H. Paralelisme Of Star Alignment. *IEEE International Conference on Instrumentation, Comunications, Information Technologi, and Biomedical Engineering*. 2013: 167-171.
[10] Zhou L, Wang H, Wang W. Parallel Implementation of Classification Algorithms Based on Cloud Computing Environment. *TELKOMNIKA Indonesian Journal of Electrical Engineering*. 2012; 10(5): 1087-1092.
[11] Li H, Gong B, Gao HB, Qian CJ. Parallel Computing Properties of Tail Copolymer Chain. *TELKOMNIKA*. 2013; 11(8): 4344-4350.
[12] "NCBI Home Page,"[Online]. Available: http://www.ncbi.nlm.nih.gov/
[13] Quinn MJ. Parallel Programing in C with MPI and OpenMP. *McGraw-Hill Companies, Inc*. 2003.
[14] Maria A Kartawidjaja. Analisis Kinerja Perkalian Matriks Paralel Menggunakan Metrik Isoefisiensi. *TESLA*. 2008; 10(2): 51-54.