COMPUTING ANGULAR CORRELATION FUNCTION USING MESSAGE PASSING INTERFACE

Fareed Ahmed Jokhio[*,**], Umair Ali Khan[*], Intesab Hussain Sadhayo[*]

ABSTRACT

There are many fields of science where the Two- Point Angular Correlation Function (TPACF) is used, such as in statistics, and astronomy. In cosmology, it is used to calculate the distribution of galaxies. The number of galaxies in the Universe is more than 170 billion in a limited region of 13.8 billion light years distance. To compute the distribution of such a large number of galaxies requires huge computational resources. This paper adopts a parallel processing approach to compute angular correlation function. The distribution of galaxies using the TPACF is computed by using a message passing programming model in a cluster based parallel processing environment. To get a better speedup, we adopt a coarse grain approach in which the computation is split among different processes. Each process gets a fixed number of data lines and performs the required computation. The results indicate that the parallel processing of the TPACF provides a significant speedup in terms of execution time.

Keywords: *Parallel processing, TPACF computation, Message Passing Interface, cluster computing, galaxies distribution in the Universe.*

1. INTRODUCTION

The earth is a very small planet and it is a part of a solar system. This solar system is a member of a galaxy having 300 to 400 billion stars [1][2]. At the present time, only a fraction of the Universe is known. According to some estimates the number of galaxies in the known Universe is more than 170 billions [3]. The galaxies have significantly different sizes in terms of the number of stars which count hundreds of billions in each galaxy. To determine at what extent galaxies are distributed, we use the Two-Point Angular Correlation Function (TPACF) [4], which is the relative probability to find the angular distance of galaxies [5] . To understand the structure of the Universe, there exists a theoretical model known as "Labmbda Cold Dark Matter" (^CDM) [6][7]. According to this model, there are two components in the Universe: (i) dark energy or dark matter which is about 96% of the total matter energy in the Universe and (ii) a normal baryonic matter which is 4%. The dark matter-energy cannot be directly observed. However, we can observe its effects on the other objects in terms of gravitational force. The search for dark matter is a continuous process. However, there is no significant breakthrough. Scientists have already performed several experiments with different kinds of equipments. On the basis of theoretical assumptions, it is believed that the dark matter takes the form of some kind of particles named as "Weakly Interacting Massive Particles" [8], or WIMPs. These particles are present everywhere. But, it is not possible for us to have a trace of them without very sensitive equipments. To hunt the dark matter, scientists use three possible ways. The first way is to explore the space. Scientists use the Alpha Magnetic Spectrometer (AMS) detector which searches for the scattered remains of the dark matter which are formed after the collision of the dark matter particles. In the second method scientists believe that the dark matter can be created by smashing particles together with the Large Hadron Collider. In the third possible method the hunt of dark matter is performed underground by using highly sensitive detectors such as the Large Underground Xenon (LUX).

As a very large number of galaxies in the Universe exists. The distribution of the galaxies in the Universe is determined by the two-point angular correlation function. The knowledge about the distribution of galaxies is particularly important in the sense that it may provide some theoretical basis about the presence of the dark matter. As the number of the galaxies is very large, computing the angular correlation of the galaxies requires substantial computing time. The main focus of this work is to compute the angular correlation of the galaxies using a parallel processing approach.

The remaining parts of this paper are organized as follows: In section II we provide the problem statement. Section III discusses the parallelization of the sequential implementation of the program which computes the distribution of the galaxies. Results are discussed in section IV. Some related works are given in section V, and conclusion is provided in section VI.

* Department of Computer Systems Engineering, Quaid-e-Awam University of Engineering, Science & Technology, Nawabshah, Pakistan

** Department of Information Technologies, Åbo Akademi University, Finland

Email:{fajokhio, umair.khan, intesab}@quest.edu.pk

2. PROBLEM STATEMENT

The main goal of this work is to parallelize the computation of the TPACF, which is used to calculate the distribution of the galaxies in the Universe. The following subsections provide more details about the problem

A. Input data

In order to determine the statistical properties of the distribution of the galaxies, the input datasets are available on the MPA Numerical Cosmology Data Archive. In this work, it is assumed that the earth is the central point of the Universe as shown in Figure 1.

The input data used in this work consists of positions of galaxies between 0.345 and 0.355 redshift. The angular separation between two galaxies is used to determine the distance between them. Input data consists of a list of positions of the galaxies in the equatorial coordinate system (Ra, Dec), where the position is a pair of right ascension and declination. The equatorial coordinates are converted to cartesian coordinate system with x, y and z axis.

B. Computing angles

In this work, the angle between two galaxies is used to determine how much they are separated

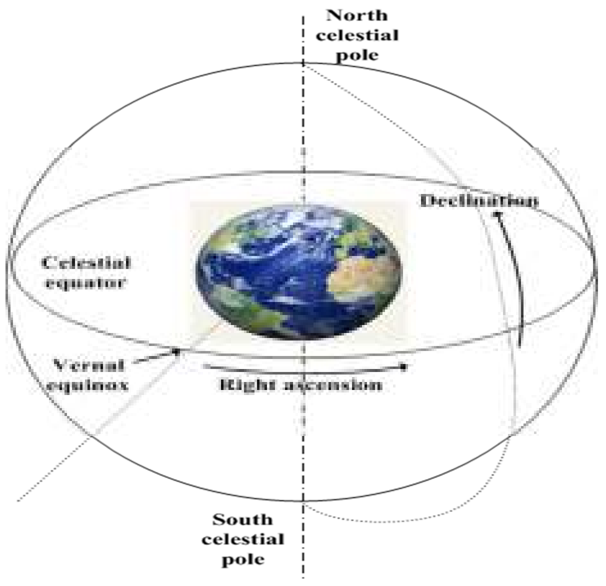

Fig.1. The Universe

from each other. The angle between two galaxies located at point ‘p’ and ‘q’ is calculated as,

$$\theta = arcos(p.q) = arcos(x_p x_q + y_p y_q + z_p z_q) \quad (1)$$

Where (x_p, y_p, z_p) are the coordinates of a galaxy located at an observation ‘p’. The angle between pairs of galaxies is computed in two sets of observations. A histogram is used to record the distribution of the angles between galaxies. It shows the number of galaxies in a certain range of angular separation.

C. Datasets

In the experiments, two different datasets are used. The first dataset is the real data ‘D’ which consists of the real observations of galaxies coordinates in the Universe. The other dataset ‘R’ is a randomly generated set of galaxy coordinates. The idea of using both real and random data is to see if the real galaxies are in an indiscriminate group in space than in a random distribution. If the galaxies are lumped together, then it indicates that there is a gravitational force which is caused by cold dark matter. This gravitational force causes more forces than the actual visible mass of the galaxies. If the galaxies are not lumped together, then it indicates the absence of the dark matter.

D. Constructing histograms

The program calculates the histogram distribution of angular separations between 0 and 64 degrees with a resolution of 0.25 degrees. The number of bins in the histograms is 4*64 = 256. Three histograms are calculated:

(i) HistogramDD(θ) is the number of coordinate pairs (p,q) with an angular separation ‘θ’, where both observations ‘p’ and ‘q’ are from the set of real observations.

(ii) HistogramDR(θ) is the number of pairs (p,q) with an angular separation ‘θ’, where observations ‘p’ are from the set of real observations and ‘q’ are from the randomly generated set.

(iii) HistogramRR(θ) is the number of pairs (p,q) with an angular separation ‘θ’, where both observations ‘p’ and ‘q’ are from the set of randomly generated coordinates.

The histogram computations have a complexity of $O(N^2)$ where ‘N’ is the number of observations, we compute the angle for all coordinate pairs. It means that the larger datasets require very high computation.

E. Calculating the correlation

The two-point angular correlations ‘ω’ for angle ‘θ’ is defined as:

$$\omega(\theta) = 1 + \frac{N^2_{random} \times DD(\theta)}{N^2_{real} \times RR(\theta)} - \frac{2N_{random} \times DR(\theta)}{N_{real} \times RR(\theta)} \quad (2)$$

Where ‘N_{real}’ and ‘N_{random}’ are the number of galaxies in the real and random datasets, respectively. DD(θ), DR(θ) and RR(θ) are the histogram values for the angle ‘θ’. A positive value of ω(θ) indicates that there are more galaxies with an angular separation ‘θ’ than expected in a random distribution. A negative value of ω(θ) indicates that there are less galaxies with an angular separation ‘θ’ than expected in a random distribution. If ω(θ) = 0 the distribution of galaxies is random.

3. PARALLELIZATION

In order to parallelize the sequential implementation of the program, we first profiled the code using gprof [9] so that hotspots or critical regions of the program can be identified. The program was profiled with a small dataset and the profiling results indicate that 87.02% of the execution time is consumed in calculating histogram and remaining 12.98% of the execution time is in the main function. If we analyze the code carefully, we will identify that the critical region, i.e., the histogram calculation is performed for histogramDD, histogramDR, and histogramRR. For histogramDD(real, real) and histogramRR(rand, rand) data points is a triangle of dots. After calculating (i, j) points DD and RR histograms are multiplied by 2, since (j, i) points are not calculated. The histogram calculations for histogramDR(real, rand) is a square (rectangle) of dots.

To get a better speedup, we split the histogram calculations among different processing units. Each processing unit gets a fixed number of data lines from the set of data points and performs the histogram computation. The histogramDR has a square (rectangular) shape. Therefore, it is possible to divide the equivalent amount of work among the processing units. In histogramDD and histogramRR the lines in the lower section have less points. We also divided the equivalent number of lines among processing units regardless of the number of points in the lines. Dynamic load balancing schemes can also be a good solution in a master/slave arrangement by using MPI. The dynamic load balancing schemes are good when the computation requirements are changing. However, in the use case of TPACF, the number of data points is already known, Hence, this paper used the static scheme based on equal number data lines for load balancing.

There are several models of computations available for parallel programming such as shared memory programming model [10], message passing [11], dataflow programming model [12], actor model [13] etc. Detailed description of all those programming models and different programming languages is out of scope here. The sequential source code is modified and re-written using the Message Passing Interface (MPI) [14]. There exist some other programming languages to write parallel programs such as Intel Cilk Plus and OpenMP. However, these programming languages are not suitable to write programs which can execute in a cluster based environment with a distributed memory system. In contrast an MPI based implementation has more portability and can execute on a multicore machine, in a cluster of computing nodes or even in a cloud computing environment.

To execute the parallel version of the code, at least two processes are created and mapped on different processors or processing cores. The one process serves as a master while the second as a slave. The number of slaves can be larger as well. The master performs the task of data distribution to slaves, while slaves perform the actual computation. Each process has a unique “rank” which is an integer starting from 0,1,2 up to N-1.

Figure 2 provides the flow chart of an MPI based program. Here the master process tasks include data partitioning, sending data to the slaves, receiving results from the slaves and finally producing the final output of the program. Each slave obtains data from the master and performs the histogram computation on that part of data and sends the results to the master. Figure 3 shows the part of the code which is used to compute the histogram and is executed at slave side. To compute different histograms such as HistDD, HistDR, and HistRR add_histogram function is invoked with different parameters. Here ‘xd_real’,’ yd_real’,’ zd_real’ indicates the ‘x’, ‘y’, ‘z’ coordinates of a point from the real dataset, while ‘xd_sim’, ‘yd_sim’, ‘zd_sim’ indicates the ‘x’, ‘y’, ‘z’ coordinates of a point from the randomly generated dataset. Both ’i’ and ‘j’ indicates the first and second points respectively. The code for add_histrogram is shown in Figure 4. The histogram computation code is exactly the same for all types of histograms. The only difference is in the arguments sent to perform the computation. Each slave gets a starting data line number and ending data line number to perform the computation.

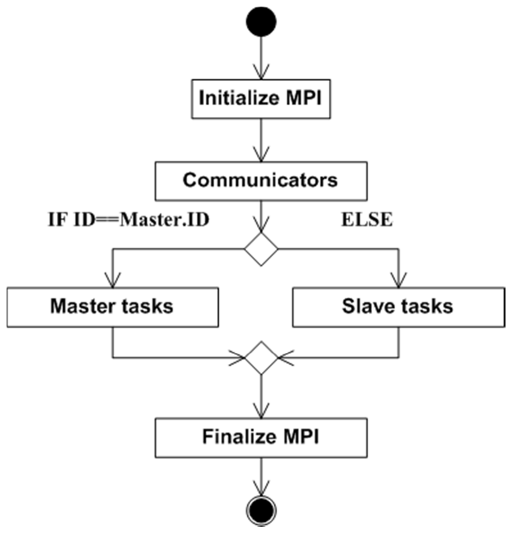

Fig. 2. MPI based program

4. RESULTS AND DISCUSSION

To perform the computation, we use a cluster with 8 compute nodes Each node has 2 Intel Xeon X5650 processors. Each processor has 6 cores with hyperthreading support. Each node has a 24 GB (Gigabytes) of RAM (Random Access Memory). For communication among different nodes, a 4xQDR (Quad Data Rate) infiniband network is used. The cluster also has a 24 TB (Terabytes) disk server.

We test the MPI framework with two datasets, namely small and medium. The small dataset has 20000 coordinates. The medium dataset has 100000 coordinates. The main purpose of this work is to compute the two-point angular correlation function using parallel processing. Our MPI based solution can work on multicore systems, clusters, and in a cloud computing environment without changing the code.

```
/* Initialize the histograms to zero */
 for ( i = 0; i <= nr_of_bins; ++i )
   {
     histogramDD[i] = 0L;
     histogramDR[i] = 0L;
     histogramRR[i] = 0L;
   }

  int x,start,end;
  x = Nooflines_Real/np;
  start = me*x;
  end = x + start;
  if(me==(np-1))
  end = Nooflines_Real;
  for(i=start;i<end;++i)
    {

     for ( j = i+1; j < Nooflines_Real; ++j )
       {

/* Histogram DD compuation */
           dd_histogram (xd_real[i], yd_real[i], zd_real[i], xd_real[j], yd_real[j], zd_real[j], histogramDD, pi, costotaldegrees);


       }
   }
 for(i=start;i<end;++i)
    {
     for ( j = 0; j < Nooflines_Sim; ++j )
       {
        /*Histogram DR compuation */


           add_histogram (xd_real[i], yd_real[i], zd_real[i], xd_sim[j], yd_sim[j], zd_sim[j], histogramDR, pi,
costotaldegrees);
```

```
      }
    }
   for(i=start;i<end;++i)

    {
      for ( j = i+1; j < Nooflines_Sim; ++j )
        {
          /* Histogram RR computation */
             add_histogram (xd_sim[i], yd_sim[i], zd_sim[i], xd_sim[j], yd_sim[j], zd_sim[j], histogramRR, pi, costotaldegrees);

        }
    }
```

Fig. 3. Computation performed by slaves

```
void add_histogram (
float px, float py, float pz, float qx,
float qy, float qz, long int *histogram,
const float pi, const float costotaldegrees)
{
  float theta;
  float degreefactor = 180.0/pi*binsperdegree;
  int bin;
  theta = px*qx + py*qy + pz*qz;
  if ( theta >= costotaldegrees ) {
  /* Skip if theta < costotaldegrees */
    if ( theta > 1.0 ) theta = 1.0;
    /* Calculate which bin to increment */
    bin = (int)(acosf(theta)*degreefactor);
    histogram[bin]++;
  }
}
```

Fig. 4 Histogram Computation

In order to compute the TPACF and determine the distribution of galaxies, two different datasets are used in the experiments. First dataset is the real data 'D' which consists of the real coordinates of the galaxies in the Universe, while the other dataset is a randomly generated set of galaxies coordinates. The real data 'D' basically indicates a subset of the real observations of galaxies coordinates in the known Universe. To determine the presence of dark matter, we plot the histograms for both real and randomly generated data. If the real data has a random distribution of galaxies, it indicates that there is no gravitational force among them and the dark matter does not exist. If the galaxies are in an indiscreimate group in space and they are not randomly distributed, then it indicates the presence of gravitational forces which are caused by cold dark matter. Those gravitational forces make it possible for galaxies to remain closer to each other. To represent the distribution of the galaxies in the Universe, we plot a bar chart graphical representation that organizes the galaxies into logical ranges. This grouping of data points or galaxies coordinates is termed as a histogram. Figures 5 and 6 indicate the histogram results of the two-point angular correlation for a small and medium datasets. HistDD indicates the histogram of the two-point angular correlation of the "real, real" coordinates of the galaxies in the Universe. HistRR indicates the histogram of the two-point correlation of the randomly generated coordinates "rand, rand". HistDR indicates the histogram of the two-point angular correlation for "real, rand" coordinates.

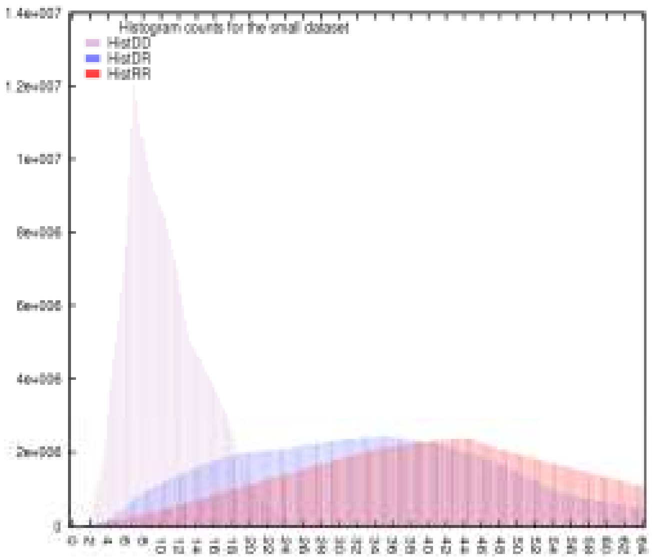

Fig. 5. Histogram for a small dataset

Figure 5 shows that the real galaxies which are indicated in histDD are more lumped together as compared with the random distribution. In the small dataset the distribution of galaxies is mostly between 2 degrees to 25 degrees, though it clearly indicates the presence of dark matter.

To justify the claim, a medium dataset histogram result is indicated in Figure 6. The distribution of galaxies in the medium dataset also depicts the similar pattern as it was in the small dataset. Here, histDD indicates the galaxies are lumped together.

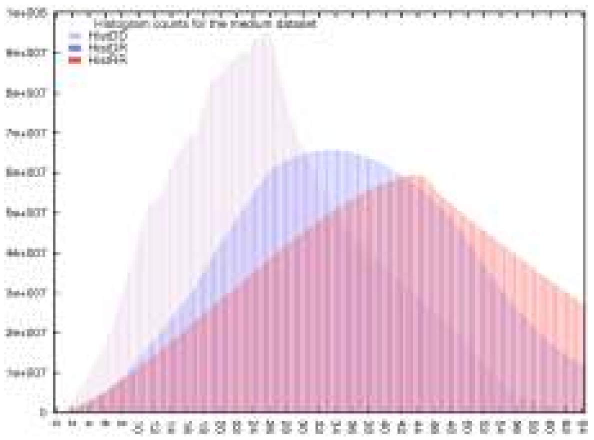

Fig.6. Histogram for a medium dataset

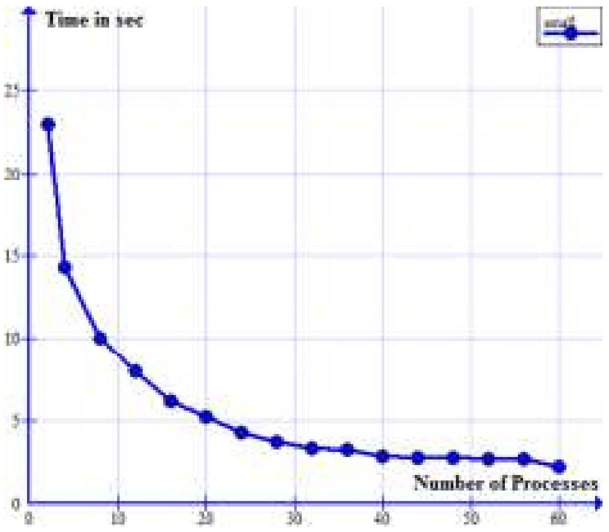

Fig.7. Results for small dataset

Figure 7 shows the execution times for the small dataset with different number of processes. Here we can clearly see that as the number of processes increases, we get better performance. Here the processes are mapped on different cores of the computing nodes on our cluster. In order to evaluate our approach, we also tested the same program on a dataset which requires more computing power as compared with the small dataset.

Figure 8 shows the results for medium dataset and it has more data points as compared with the small dataset. Here we can also see a speedup in execution time. Since the cluster used for doing experiments has total 08 computing nodes. Each node has 02 processors and each processor has 06 cores. Hence, this system is basically a cluster of multicore machines. In a multicore system, usually there is a shared system bus and shared cache among different cores. Therefore, memory intensive applications do have resource contention and do not scale well [15] . Therefore, the figure 8 which represents the computation time for a medium dataset does not have similar cure as compared with the curve in figure 7.

In our implementation, the first slave process gets the maximum work load and the last slave process gets the shortest work load. It will be more cost efficient to map the master process and the last slave process on the same physical core or processor. In this implementation, the master process mainly partitions the data and send it to the slaves and finally waits to receive the computing results from the slaves. Therefore, mapping the master and the last slave processes on the same physical core will not have any significant effect on the performance.

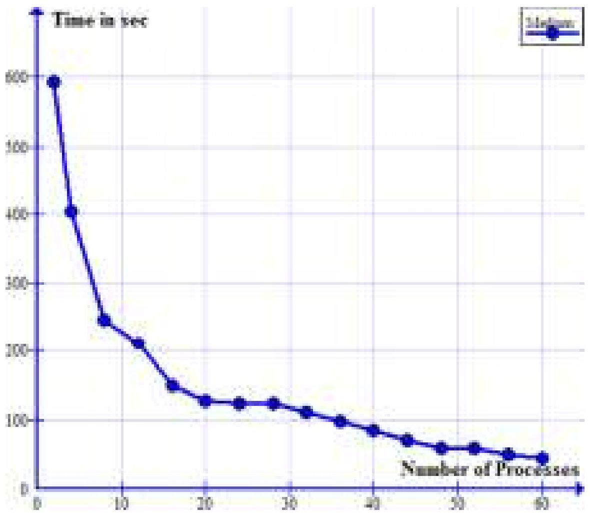

Fig. 8. Results for medium dataset

The results in Figure 7 and 8 indicate that the parallel processing approach to compute the TPACF is also scalable, and can be used for larger datasets and thousands of computing nodes with several processing cores can be used to perform the computation.

5. RELATED WORKS

There exist several works on accelerating TPACF on different platforms such as GPUs, FPGAs and Clusters of computing nodes. In addition to that there also exist some works, which attempt to optimize the algorithm of computing TPACF in less time. Landy et. al. [5] presented a method to quickly calculate the variance and bias for the angular correlation function ω(θ), based on the number of pairs of galaxies (DD), random distribution (RR), and on the cross-correlation between the galaxies and random distribution (DR). Although, their method can be used to calculate the bias and variance in shorter time. However, their work is not based on the parallel programming. The work in this paper provides a parallel processing approach to perform the TPCAF computation. Dolence at al.[16] developed an algorithm to perform the computation of two point correlation functions. The parallel algorithm is based on a dual-tree and it requires a very huge amount of memory. To overcome the memory requirements, each data set is broken into smaller sets. This breaking of dataset introduces its own overheads and entire data set is frequently accessed. Kindratenko et al. [17] presented the implementation of TPACF computation on FPGAs. In the experiments 97K dataset was used and two different FPGA designs were evaluated. The obtained speedup was 3.3x and 6.7x as compared with the reference implementation running on an AMD Opteron machine with 2.4 GHz microprocessor. In a similar work using FPGAs Kindratenko et al. [18] claimed a speedup of more than 90x as compared with their sequential implementation running on a single CPU. FPGAs could provide some speedup for 97K dataset. However, if the size of the data set will be in the millions then the current technology of FPGAs will not be able to perform the computation. To compare the performance of a CPU and FPGA, Jatin et al. [19] optimized the TPACF algorithm and executed it on a single node. Their optimized implementation for the 97K particles dataset is 80x faster as compared with the FPGAs performance. There also exist some implementation GPUs [20] which gives better performance as compared with CPU based implementations. Jatin et al. [19] worked on scaling the TPACF computation on a very large number of nodes while using very huge datasets. They claim that the obtained speedup is 35x to 37x as compared with other state-of-the-art parallelization techniques to compute TPACF. It is not possible to cover all related work. However, most works are either platform specific or use optimizations. In contrast, this paper proposes a generic parallel processing approach to computing TPACF in a parallel computing environment.

6. CONCLUSION

This paper presented a parallel implementation of the two-point angular correlation function by using an MPI programming model. The approach is evaluated using a master/slave communication model. The most compute intensive operation of the correlation functions is the computation of different histograms. To distribute the data for the histogram computation among different slaves, an equivalent number of data lines are taken into account. The results indicate that the parallelization approach gets a higher speedup in terms of execution time as the number of processes are increased. A potential future work of this work is a comparison between shared memory models and distributed memory models for multi-core systems for this application.

REFERENCES

[1] R. E. Nancy and W. Mark, "The Mass of the Andromeda Galaxy," Royal Astronomical Society, vol. 316, no. 1, pp. 1-16, 13 April 2000.

[2] X. X. Xiang, W. R. Hans, b. Z. Gong, R. F. Paola, N. Thorsten, S. Matthias, V. d. B. Frank C, C. B. Timothy, W. L. Young, F. B. Eric, M. R. Constance, Y. Brian, J. N. Heidi, W. Ronald, K. Xi, W. L. S. Matthew and P. S. Donald, "The Milky Way's Circular Velocity Curve to 60 kpc and an Estimate of the Dark Matter Halo Mass from Kinematics of ~2400 SDSS Blue Horizontal Branch Stars," The Astrophysical Journal, vol. 684, no. 2, pp. 1-42, 28 May 2008.

[3] D. Deutsch, The Fabric of Reality, UK: Penguin Books Limited, 14 April 2011.

[4] X. Chen and J. Hakkila, "The Two-Point Angular Correlation Function and BATSE Sky Exposure," Dec 1997.

[5] S. D. Landy and A. S. Szalay, "Bias and Variance of Angular Correlation Functions," Astrophysical Journal, vol. 412, pp. 64-71, 1993.

[6] B. Luc and L. T. Alexandre, "Model of dark matter and dark energy based on gravitational polarization," p. 4, 02 July 2008.

[7] D. Aharon, K. David and L. Yoav, "Cold dark matter from dark energy," p. 4, 29 November 2001.

[8] G. S. Alex and M. K. Sawas, "Exclusion of canonical WIMPs by the joint analysis of Milky Way dwarfs with Fermi," p. 5, August 2011.

[9] S. L. Graham, P. B. Kessler and M. K. McKusick, "gprof: a call graph execution profiler," ACM SIGPLAN Notices, vol. 39, no. 4, pp. 49--57, April 2004.

[10] A. R. Kay and R. Steven, UNIX Systems Programming: Communication, Concurrency, and Threads, Prentice Hall Professional, 2003.

[11] M. M. John and C. W. David, "Some considerations for a high performance message-based interprocess communication system," in Proceedings of the 1975 ACM SIGCOMM/SIGOPS workshop on Interprocess communications, New York, NY, USA , 1975.

[12] M. J. Wesley, R. P. H. J and J. M. Richard, "Advances in Dataflow Programming Languages," ACM Computing Surveys, vol. 36, no. 1, pp. 1-34, 2004.

[13] G. A. Agha, "ACTORS: A Model of Concurrent Computation in Distributed Systems," MIT Libraries, Cambridge, United States, 1985.

[14] "MPI: A Message-Passing Interface," University of Tennessee, Knoxville, Tennessee, May 5, 1994.

[15] H. Robert, J. Haoqiang, M. Piyush, C. Johnny, D. Jahed, G. Sharad, J. Dennis, T. Kenichi and B. Rupak, "Performance impact of resource contention in multicore systems," in Parallel & Distributed Processing (IPDPS), 2010 IEEE International Symposium, Atlanta, Georgia, April 2010.

[16] D. Joshua and J. B. Robert, "Fast Two-Point Correlations of Extremely Large Data Sets," in The 9th LCI International Conference on High-Performance Clustered Computing, Pittsburgh, Pennsylvania, USA, 2008.

[17] V. K. Volodymyr and J. B. Robert, "Accelerating Cosmological Data Analysis with FPGAs," in 17th IEEE Symposium on Field Programmable Custom Computing Machines, Napa, California, 2009.

[18] V. K. Volodymyr, D. M. Adam and J. B. Robert, "Implementation of the two-point angular correlation function on a high-performance reconfigurable computer.," Scientific Programming, vol. 17, no. 3, pp. 247-259, 2009.

[19] C. Jatin, K. Changkyu, S. Hemant and P. Jongsoo, "Billion-Particle SIMD-Friendly Two-Point Correlation on Large-Scale HPC Cluster Systems," in SC '12 Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis, Utah, USA, 2012.

[20] W. R. Dylan, V. K. Volodymyr and J. B. Robert, "Accelerating cosmological data analysis with graphics processors," in Proceedings of 2nd Workshop on General Purpose Processing on Graphics Processing Units , Washington, DC, USA, 2009.