Journal of Environmental Protection and Ecology 15, No 1, 287–292 (2014)

Computer applications on environmental information system

PARALLEL IMPLEMENTATON OF THE WEIBULL DISTRIBUTION PARAMETERS ESTIMATOR

S. DULI[a]*, B. KRSTAJIC[b]

[a]*Department of Computer Sciences, Faculty of Natural Sciences, University of Shkoder, Shokdra, Albania*
E-mail: *siditaduli@yahoo.com*
[b]*Department of Electrical Engineering, University of Montenegro, Podgorica, Montenegro*

Abstract. The Weibull distribution is a statistical tool used to model the wind speed. Estimating the Weibull parameters, when the sample of data is given, requires some mathematically complex calculations. In case when the sample contains a big amount of data, making these calculations to require longer time. A solution for a better speed up of processing might be to perform these calculations in a parallel computing system. Among different parallelisation strategies, in this research there are implemented two of them: the MPI (message passing interface) and the Pthread. The aim of this article is to verify the increased speed-up of the parallel version, and to compare the techniques of paralleling the code, the MPI and the POSIX, in estimating the Weibull parameters of large datasets.

Keywords: parallel programming, the Weibull distribution, MPI, Pthread.

AIMS AND BACKGROUND

The analysis of the wind speed is useful in many fields of industry, in agriculture and in meteorology. Generating energy from the wind power is a mature technology and economically competitive with most fossil fuel applications. Europe is a leader of wind energy using in the world[1]. The wind energy power is the energy that has been making the fastest increase in the world[2]. In agriculture, irrigation of an agricultural field above the level of a dam or pond by a pump working by wind energy is much more economical than the pumps working with fuel and electrical energy[3]. In the field of wind erosion of soil is highlighted that strong winds cause a number of unfavourable conditions and processes in the microclimate and the ecological production environment[4]. All these highlight the important role of statistics of the wind speed in a specific location.

The Weibull distribution is a statistical tool used to model the wind speed. The wind speed data collected in a weather station tend to have the form of an Weibull distribution. The shape of the graphics depends on the parameters of the distribution, which are estimated by the set of wind speed data. Once the Weibull

* For correspondence.

distribution can be used to calculate the probability of a particular wind speed at a particular location, it can be used to work out the number of hours per year that certain wind speeds are likely to record and therefore the likely total power output of a wind turbine per year[5].

The aim of this research is to perform the estimation of the Weibull parameters in a distributed computer system. The principle of parallel programming is to divide the problem in smaller parts and to distribute the calculations between many processors. In this way can be archieved a faster result than at the same code executed in one processor. There are different ways of paralleling the code of this application. Two of them are the MPI and Pthread.

EXPERIMENTAL

Two main methods that can be used to estimate the Weibull parameters are the maximum likelihood MLE, the least squares, and the Bayesian estimation approach. In this research it is applied the maximum likelihood method, as it is usually considered to be more robust and produces more accurate results. MLE is a commonly used procedure because it has very desirable properties[6]. It is used to find global optimum parameters for a certain function to fit a given data set. In this case, MLE is used to find the optimum parameters for the Weibull distribution.

Among the methods of paralleling the code, are chosen the MPI and Pthreads. MPI provides source-code portability of message-passing programs written in C or FORTRAN across a variety of architectures. A good point of using the message-passing is its wide portability. The programs using MPI libraries may run on distributed-memory multicomputer, shared-memory multiprocessors, networks of workstations, and combinations of all of these. MPI is implemented on a great variety of machines, including those 'machines' consisting of collections of other machines, parallel or not, connected by a communication network[7].

Meanwhile the Pthread library provides an interface to generate and interact with separate threads of execution within a program. This standard is defined by the Institute of Electrical and Electronics Engineers (IEEE) and is available across nearly all variants of the UNIX by IEEE operating systems[8].

The algorithm is written in C programming language. In the MPI version it is included the mpi.h library and in the Pthread version is included the Pthread.h library. The system where it is executed is a grid cluster, a Linux-based system and portable enough to run with consistent result any implementation of message passing model.

The application takes a censored set of data, which might be ordered or unordered, taken from a sample of *N* data. It is given the location parameter of the Weibull distribution. The data sample and the location parameter are read from a file by each process that is created. The algorithm makes an estimation of the scale parameter and shape parameter of this distribution.

Also for the density function of the Weibull distribution is calculated the mode, the mean and the variance for the given set of data. This procedure is repeated *R* times for different set of data, taken from this *N* data sample.

In tests performed, the data samples were not taken for a specified location. As the aim of the tests were the to compare the performance and speed-up of these two techniques, and not the wind statistics for any region, the samples were taken for a random wind speed values in-between 1 and 20 km/h. Of course, this implementation could be used for any sample data to any region.

RESULTS AND DISCUSSION

Tests are performed with data samples of 10 000 000 values and 5 000 000 values. The Weibull parameters are calculated for 100 such samples, taken randomly from a bigger amount of data, from 15 000 000. In the case when the sample contains wind speed data, these samples might be chosen for calculating 100 estimations of 100 different periods of time, seasons or weeks that might be of interest to research. For each of these samples is calculated also the mode, the mean and the variance of the distribution. The performance is tested for different number of processors, beginning with the serial version with only one processor running and doing the calculations, up to twelve parallel processors doing the calculations. The same tests are done with both techniques, MPI and Pthread. In the Pthread version, there are created parallel threads instead of parallel processes. Both MPI and Pthread scale well up to 12 cores.

Results are compared with the related work in the Stamatakis, parallel programming, implementing the MPI and Pthread in the field of bioinformatics, and paralleling phylogenetic likelihood function[9,10].

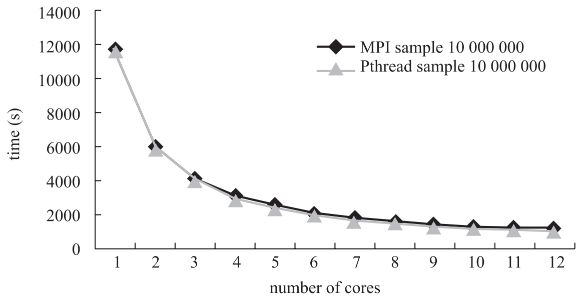

Fig. 1. Comparison of time performance of MPI and Pthreads for a sample containing 10 000 000 data

Figure 1 shows the time spent to estimate the Weibull parameters of a dataset containing 10 000 000 data. Graphics are built for both versions, MPI and Pthreads, and for a number of cores increasing up to 12 cores. For large dataset, both versions reduce the time spent to estimate the result. Executing the serial program, with a large dataset, it is needed 11 718 s to get the parameters of the Weibull distribution for this sample. There results show how parallel programming helps in reducing this time. Both techniques, MPI and Pthread, get the result 1.96 times faster as soon as a second processor is added. The time is reduced up to 1220 s when is executed the MPI version in 12 cores. Meanwhile time spent for the Pthread version using 12 threads is 1022 s.

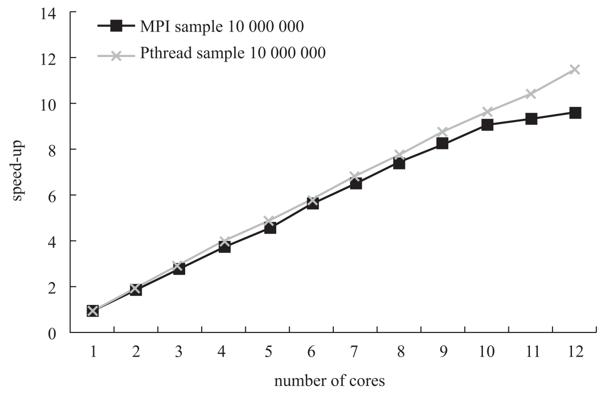

Fig. 2. Comparison of speed-up of MPI and Pthreads for a sample containing 10 000 000 data

Figure 2 shows the speed-up graphic for these performance tests. This speed-up is the number of times that the parallel implementation is faster than the serial implementation. The tests show that in both cases the speed-up increases in a linear form. According to the Gustavson law[11], the linear graphic depends also on an alpha coeficient which is the non-parallelisable fraction of any parallel process. The efficiency shows how well-utilised are the processors in the parallel implementation. It is a value between zero and one. A parallel efficiency of one corresponds to ideal, linear speed-up. In this case, the alpha coeficient is small, because the implementation is made with a small fraction of non-pallelised.

In a similar comparison between Pthreads and MPI, but in a different case of study, Stamatakis notes that: 'On the larger dataset scalability of OpenMP and Pthreads are similar, while the MPI-based version yields slightly sub-linear speed-ups on the Opteron cluster for 7 and 15 worker processes'[10]. Similar results are shown in this research. On a larger dataset the Pthread has a speed-up almost linear, showing almost an ideal performance. Meanwhile the performance of MPI gets down when running the application in 9 or more cores.

The cost of manage of a new process requires more system resources than managing a new thread with shared memory with older ones. Making comparisons in tests for both MPI and Pthread version, for a sample of 10 000 000 data, for smaller number of threads/processors the speed up is nearly the same.

The same implementation is executed applying different sample containing 5 000 000 data.

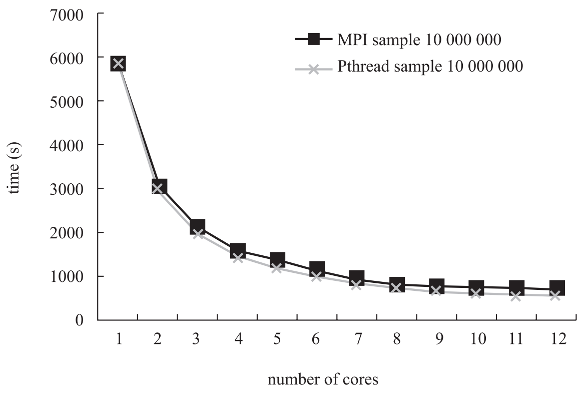

Fig. 3. Comparison of time performance of MPI and Pthreads for a sample containing 5 000 000 data

In Fig. 3 there are presented the grapics of time spent for estimating the Weibull parameters for a sample containing 5 000 000 data, for both versions, MPI and Pthread. Results show that while executing the serial program, the result is obtained after 5857 s. Again, for a different sample, the parallel program shows its efficency in time, as soon as we add a second processor in the MPI version, or as soon as its created a new thread, in the Pthread version. The minimum time spent, for these tests, is during the execution of the parallel versions in 12 cores. In the MPI case time spent is 712 s, and in Pthread, it is needed only 560 s to have the result.

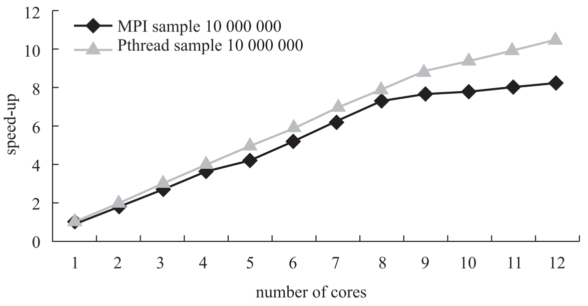

Fig. 4. Comparison of speed-up of MPI and Pthreads for a sample containing 5 000 000 data

Figure 4 shows the curves of speed-up for both MPI and Pthread versions, applying the calculations on a sample containing 5 000 000 data. Results show that, with a smaller sample, both speed-ups of MPI and Pthread are reduced. The speed-up of Pthread is no more almost linear, reaching a value at 10.45 while running 12 parallel threads. The speedup of MPI version is also lower than the speedup shown in Fig. 2.

CONCLUSIONS

• This paper introduced the benefits of the parallel programms applied in mathematical and statistical calculations. The serial version is far slower in generating the result compared with the parallel version. That shows that the parallel program presented in this research will be very useful in industry or meteorology, or in every other field where is spent a lot of time analysing wind speed of a specific location.

• For more than 9 threads/ processors, the speed-up of the Pthread version is higher than the speed-up of MPI version. Also, if we compare in time, and that is the scope, having a faster execution, less time is spent during the execution of the Pthread version. This results for both samples used.

• During the process of building the parallel versions of the application, it is noted that the cost of implementation of the parallel program is the complexity of the code and the knowledge of the programmer about the mpi.h and Pthread.h library. The programmer should care about how the messages will pass without causing any deadlock between sending and receiving new messages. Also the programmer should know the implementation and syncronisation of threads.

REFERENCES

1. K. STEPANOVA: Renewable Energy in Tourism: Opportunities and Benefits. J Environ Prot Ecol, **10** (2), 468 (2009).
2. F. CATOVIC, M. BEHMEN, E. ZLOMUSICA: Trends in the Development of the Electric Power Systems Based on Wind Energy in World and in Bosnia and Herzegovina. J Environ Prot Ecol, **5** (4), 836 (2004).
3. I. BECENEN, H. INCE K. KARACAVUS: Water Storing for Irrigation of Agricultural Fields from Ponds or Dams by Using Wind Energy in the Thrace Region. J Environ Prot Ecol, **8** (4), 849 (2007)
4. B. PEEV, D. IVANOVA: Climatic and Agroecological Preconditions for Wind Erosion in Bulgaria. J Environ Prot Ecol, **2** (3), 649 (2001).
5. http://www.reuk.co.uk/Wind-Speed-Distribution-Weibull.htm.
6. H. RINNE: The Weibull Distribution. A Handbook. Chapman & Hall, CRC, 2009.
7. Y. AOYAMA, J. NAKANO: Practical MPI Programming. International Technical Support Organization, 1999.
8. D. R. BUTENHOF: Programming with POSIX thread. Addison-Wesley, 1997.
9. W. PFEIFFER, A. STAMATAKIS: Hybrid MPI/Pthreads Parallelization of the RAxML Phylogenetics Code. In: IEEE International Symposium on Parallel & Distributed Processing, Workshops and Phd Forum, 2010, 1–8.
10. A. STAMATAKIS, M. OTT: Exploiting Fine-grained Parallelism in the Phylogenetic Likelihood Function with MPI, Pthreads, and OpenMP: A Performance Study. Lecture Notes in Comp Sci, **5265**, 424 (2008).
11. T. RAUBER, G. RUNGER: Parallel Programming for Multicore and Cluster Systems. Springer, 2011.

Received 9 January 2014
Revised 7 February 2014