PARALLEL COMPUTING OF WEIBULL DISTRIBUTION PARAMETERS

Sidita Duli, Faculty of Natural Sciences, University of Shkoder, Albania
Božo Krstajić, Department of Electrical Engineering, University of Montenegro

Abstract: *The statistical calculations might have complex formulas to compute. In cases when the sample contains a big amount of data, the time of execution increases. One of these is calculating the parameters of Weibull distribution when the sample data is given. The parameters are estimated by using the maximum likelihood method, which requires many mathematical computations. For a better speed of processing the result, a solution might be to make this calculation in a parallel computing system, and by using a parallel algorithm. One technique is by using the Pthread library in C application. The threads created are used to compute simultaneously the calculations in complex statistical formulas. This article shows how the Pthread code added in the application helps to perform faster.*

1. INTRODUCTION

The weibull distribution is an important distribution especially in maintainability analysis and reliability. Furthermore it is widely used in biomedical engineering and to model variations in wind speed. This distribution exists in two main forms: the two-parameter and three-parameter Weibull distribution.
The two parameter Weibull Distribution has the following density and distribution functions:

$$f(x)=\left(\frac{b}{a}\right)\left(\frac{x}{a}\right)^{b-1} e^{-\left(\frac{x}{a}\right)^{b}} \quad (1)$$

$$F(x)=1- e^{-\left(\frac{x}{a}\right)^{b}} \quad (2)$$

where a is the scale parameter and b is the shape parameter.
The three parameter Weibull distribution has the following formula of density and distribution:

$$f(x)=\left(\frac{b}{a}\right)\left(\frac{x-c}{a}\right)^{b-1} e^{-\left(\frac{x-c}{a}\right)^{b}} \quad (3)$$

$$F(x)=1-e^{-\left(\frac{x-c}{a}\right)^{b}} \quad (4)$$

where a is the scale parameter, b is the shape parameter and c is the location parameter.
The application makes an estimation of the parameters of the Weibull distribution function. Two main methods that can be used are the maximum likelihood and the least squares. In the application it is used the maximum likelihood method, as it is usually considered to be more robust and produces more accurate results.
The method of maximum likelihood [2] is a commonly used procedure because it has very desirable properties. Let x_1 , x_2 , ..., x_n be a random sample of size n drawn from a probability density function f_x (x;θ) where θ is an unknown parameter. The likelihood function of this random sample is the joint density of the n random variables and is a function of the unknown parameter. Thus

$$L= \prod_{i=1}^{n} fxi(xi,\theta) \quad (5)$$

is the likelihood function. The maximum likelihood estimator (MLE) of θ , say $\hat{\theta}$, is the value of θ that maximizes L or, equivalently, the logarithm of L . Often, but not always, the MLE of θ is a solution of

$$\frac{d\log L}{d\theta} =0 \quad (6)$$

where solutions that are not functions of the sample values x_1, x_2, ..., x_n are not admissible, nor are solutions which are not in the parameter space.
The problem with the MLE lies in obtaining initial estimates for x and θ**.** MLE does not supply the values for x and θ, those values are supplied by a parameter estimation technique. The most common technique for parameter estimation of Weibull distributed reliability data is the Newton-Raphson algorithm (NRA).
The Newton-Raphson algorithm uses a Taylor Series expansion about the predefined function to estimate the parameters' values. However, the main problem with the NRA algorithm is the selection process for the initial parameter values. The selection of the initial parameter values is critical, because of the inability of the NRA to avoid

convergence to local optima. f(x; θ) is the function which is trying to be optimized. In this research the function f(x; θ), is the Two-Parameter Weibull Distribution probability density function.
These calculations in the serial version requires lots of time when the data sample is too large and when the calculations are done for a large number of samples. The parallel version executed in parallel computers systems makes these estimations in less time. There are different ways of parallelization the code of this application, but below is presented an applied method by using the POSIX library.

2 METHODS

Using the technique of parallelizing the code by two different processors that use the same memory space, it makes possible that complex tasks might be computed in a shorter time as they were computed by a single processor architecture.
Pthreads are a set of types and procedure calls in C. They are implemented in a *pthread.h* file, and a thread library. [1]
The POSIX thread library provides an interface to generate and interact with separate threads of execution within a program. This standard is defined by the IEEE and is available across nearly all variants of the UNIX operating systems.
The processor creates threads that are used to compute a part of calculations of the whole application. They are running concurrently, so the total time to compute the result is smaller and is depending on the number of threaded created by the processor.
The primary motivation for using threads is to realize potential program performance gains. Comparing to processes, creating a thread requires fewer operations, and to manage a thread requires fewer system resources.
Another advantage is related to the software portability. Applications that use threads can be developed on serial machines and run on parallel machines without changing anything. This portability is very significant advantage of threaded APIs.
The program takes a censored set of data, which might be ordered or unordered, taken from a sample of N data. The location parameter A is given, and the algorithm makes an estimation of the scale parameter B and shape parameter C of the equation (4) of the Weibull distribution. The method used for this estimation is the maximum likelihood estimation. To calculate the root of (6) where the L is the likelihood function and is computed via a Newton-Raphson iteration. The Newton-Raphson method in one variable is implemented as follows: Given a function f defined over the reals x and its derivative f', we begin with a first guess x_0 for a root of the function f [3]. The function is reasonably well-behaved a better approximation x_1 is

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}\,. \qquad (7)$$

In our case, the value of x_0 is given, and if it is zero is it taken a guess value of x.
Also for the density function is calculated the mode, the mean and the variance for the given set of data. This procedure is repeated for R times for different set of data, taken from this N data sample.
The algorithm is written in C, including the Pthread library. The system where it is executed is a grid cluster, a Linux based system and portable enough to run with consistent result on any POSIX system. The data sample and the location parameter is read from a file by each thread that is created. These parallel threads take the set of data M, from the sample N, and it is repeated R/nr_threads times.

3.RESULTS AND DISCUSSIONS

Tests are performed with data samples of 1000000 values, and 2000000 values. The Weibull parameters are calculated for 100 such samples, and for each of them is calculated also the mode, the mean and the variance of the distribution. The project is tested for different number of threads, beggining with the serial version with only one thread running a doing the calculations, to eight threads in parallel doing the calculations.
In the table 1and table 2 are given the results for the time in seconds and for the speed-up for different number of threads used.

Table 1: Performance in time and speed up of the application with a sample of 1 000 000 data.

Threads	Time in Seconds	Speed-up
1	1214	1
2	420	2.8
3	269	4.5
4	204	5.9
5	165	7.3
6	135	8.9
7	111	10.9
8	97	12.5

Table 2: Performance in time and speed up of the application with a sample of 2 000 000 data.

Threads	Time in Seconds	Speed-up
1	2436	1
2	816	2.9
3	537	4.5
4	408	5.9
5	325	7.4
6	260	9.3
7	225	10.8
8	187	13.0

The figure 1 and figure 2 show a graphical presentations for the time spent for different number of threads. Time in reduced immediately as the second thread is added, increasing in this way the performance of all the parallel executions of the application.

Figure 1 : Graphical presentation of the performance in time of sample with 1000000 data.

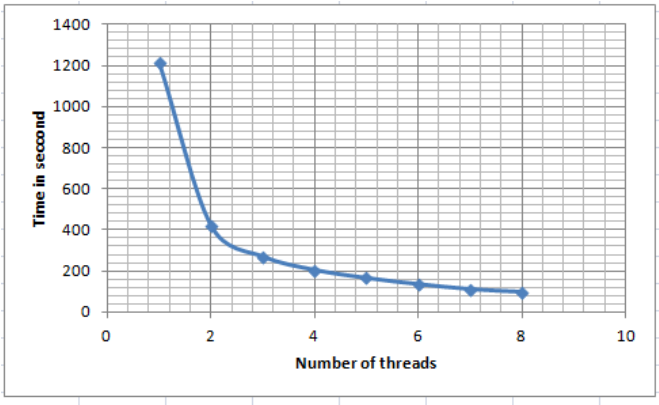
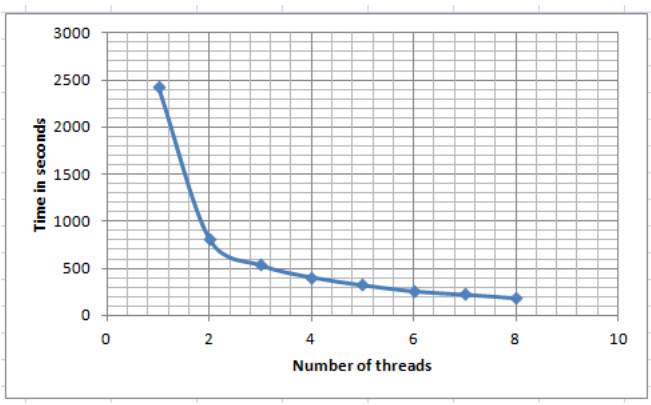
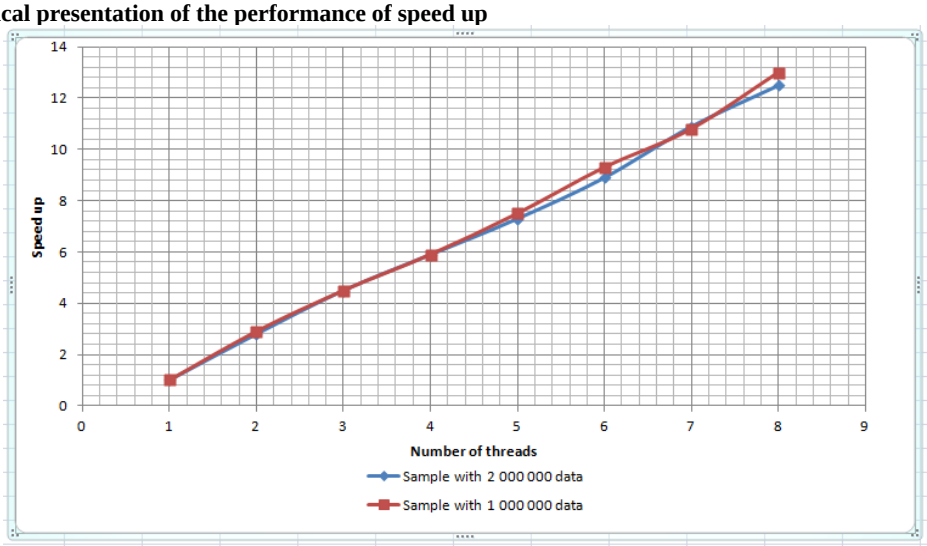


Figure 2 : Graphical presentation of the performance in time of sample with 2000000 data.



The figure 3 shows the speed-up graphic for these performance tests. The tests shows that in both cases the speed up increases in a linear form.

Figure 3:Graphical presentation of the performance of speed up



5.CONCLUSIONS

This paper introduced the benefits of using the Pthread library as a function of highlighting the advantaged of the parallel versions in mathematical and statistical calculations. The Pthread has the ability to create concurrent excecutions instances on a single processor. It also allows to share global memory and maximizes the machine's resources. In the performance tests it is shown that adding a second thread iy helps a lot by increasing the speed up till 6 times faster. Also it is noticed that after a certain number of thread, the speed up is not increased so fast as it does for a couple threads added.

Also it is worth noting the difficulty that is required to turn the serial code into a parallel code using the threads. The programmer has to make this explicitly in the code, by using the Pthreads library functions of creating and managing them. It is required more work to the programmer to know how the threads work and how they will share the memory between them. The programmer should know how to manage the access of a shared variable in a simultaneously way. Also the programmer should be careful with the grobal variables. Threads should not make changes to them, but only by using the mutex semaphore.

The cost of creating threads, instead of using only one thread in the serial version, is the complexity of the code, and the effort of the programmer to make it work without deadlocks. However, compared to the cost of creating and managing a process, a thread can be created with much less operating system overhead. Managing threads requires fewer system resources than managing processes.

REFERENCES

[1] B. Lewis, D.J. Berg, (2008), Pthreads Primer.

[2] Harter, H. L. and Moore, A. H. (1965). Point and Interval Estimation, Based on m-order Statistics, for the scale parameter of a Wiebull Population with Known Shape parameter.