Pure appl. geophys. 165 (2008) 749–759
0033–4553/08/030749–11
DOI 10.1007/s00024-008-0313-0

Pure and Applied Geophysics

Fast Computing for LURR of Earthquake Prediction

YANGDE FENG,[1] XUEBIN CHI,[1] WU WANG,[1] JIANG CHEN,[2] and XIANGCHU YIN[3]

Abstract—The LURR theory is a new approach for earthquake prediction, which achieves good results in earthquake prediction within the China mainland and regions in America, Japan and Australia. However, the expansion of the prediction region leads to the refinement of its longitude and latitude, and the increase of the time period. This requires increasingly more computations, and the volume of data reaches the order of GB, which will be very difficult for a single CPU. In this paper, a new method was introduced to solve this problem. Adopting the technology of domain decomposition and parallelizing using MPI, we developed a new parallel tempo-spatial scanning program.

Key words: Load-unload response ratio (LURR); temporal-spatial scanning, optimization schemes.

1. Introduction

The Load-Unload Response Ratio (LURR) Method, which was invented by Prof. Yin (YIN, 1987), has achieved successful results in earthquake prediction within the China mainland as well as other regions in America, Japan and Australia (ZHANG *et al*., 2004; YIN *et al*., 1904; MORA *et al*.).

The main idea of the LURR earthquake prediction approach is that when a system is in a stable state, its response to a small loading is nearly the same as that to unloading, but when the system is near failure or in an unstable state, the response to loading and unloading becomes quite different (YIN *et al*., 1995, 2000). LURR is defined according to this difference. P and R are the load and response of a system respectively, if P has a small change ΔP resulting in a small change to R of ΔR, then we can define X as

$$X = \lim_{\Delta P \to 0} \frac{\Delta R}{\Delta P}, \tag{1}$$

[1] Supercomputing Center Computer Network Information Center Chinese Academy of Sciences, P.O. Box 349, Beijing 100080, China. E-mail: ydfeng@sccas.cn
[2] Software Engineer, Intel Asia-Pacific Research and Development Ltd., Shanghai, China.
[3] LNM, Institute of Mechanics, Chinese Academy of Sciences, Beijing 100080, China.

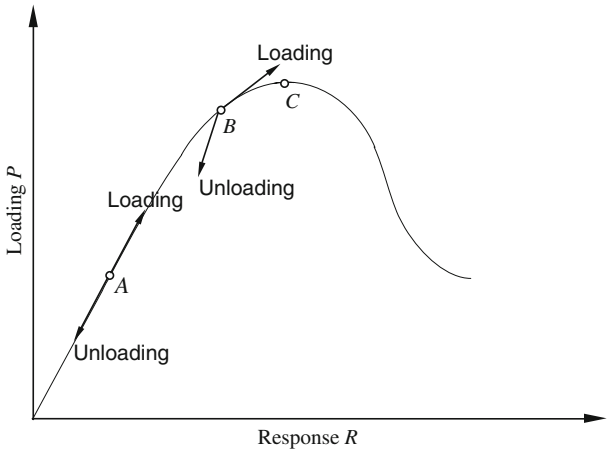

Figure 1
Constitutive relation of systems.

This is thus called the response ratio, and LURR is defined as

$$Y = \frac{X^+}{X^-}, \tag{2}$$

where X^+ and X^- are response ratio during loading and unloading. As shown in Figure 1, when a system is in a stable or linear state, $X^+ \approx X^-$ then $Y \approx 1$. When a system lies beyond the linear state, $X^+ > X^-$ and $Y > 1$. Hence, Y can be used as a criterion to judge the degree of stability of a system.

In LURR theory, Y is defined directly by the seismic energy as follows:

$$Y = \lim_{\Delta p \to 0} \frac{E^+}{E^-},$$

where E^+ and E^- are seismic energy during loading and unloading (ZHANG *et al*, 2000). Since the preparation and occurrence process of earthquakes are controlled not only by deterministic dynamical law but also affected by stochastic or disorder factors, ZHUANG and YIN (1999) studied the influence of random factors on LURR in order to judge whether a high Y value can be considered an earthquake precursor at a specified confidence level. They gave the critical value of LURR Y_c that depends on the number of earthquakes at different specified confidence levels. For instance, at the confidence level of 90%, Y_c equals 3.18 if the number of earthquakes in the time and space window is 20, which means that Y should be equal to or greater than 3.18 when the number of earthquakes is 20. For the confidence level of 99%, Y_c is 7.69 if the number of earthquakes in the specific time and space window is 20. The greater the earthquake number is, the lower the Y_c (critical LURR).

In this paper we give the critical region of LURR by Y/Y_c instead of Y at a confidence level of 99%.

With the expansion of the prediction region, the refinement of its longitude and latitude, and the increase of the time period, the computation overburden will be very high, and the volume of data reaches the order of GB, which will be very difficult for a single CPU to deal with. In this paper, a new method was introduced to solve this problem. Adopting the technology of domain decomposition and parallelizing using MPI, we developed a new parallel tempo-spatial scanning program based on Yin's previous work.

2. Optimization and Testing of LURR

We analyzed the algorithm as well as the source code of the tempo-spatial scanning program of LURR and optimized it. It was developed in four aspects: Eliminating redundant computation, eliminating unnecessary computation, improving algorithm formula and parallelizing. The code sample data set provided by Prof. Yin is used to measure the optimization of each aspect.

In this sample, the spatial region is from longitude 70.00°E to 135.00°E, and latitude 20.00°N to 55.00°N, earthquake data are from April 1, 2004 to March 31, 2005, and the earthquake spatial scale is R = 200 km, while the spatial precision for scanning is 0.1^{o}.

2.1. Eliminating Redundant Computation

The reason for eliminating redundant computation is that some values used in calculations are fixed during a calculation. This recomputation wastes massive time. Therefore, we redesigned the program by letting these values be computed in advance and be stored in an array. When needed, they will be recalled from the array directly instead of being recomputed.

According to the source code analysis and the program performance profiling, it is known that calculating the distance between a location and epicenter takes the longest time in the whole procedure. This distance is two points' distance of great circles on the spherical surface. The following formula can be used in the procedure to calculate the distance:

$$d = R_e \times \sqrt{\cos^2(lat_z) + \cos^2(lat_c) - 2\cos(lat_z)\cos(lat_c)\cos(lon_z - lon_c) + (\sin(lat_z) - \sin(lat_c))^2}, \quad (3)$$

where R_e is the Earth's radius, lon_c and lat_c are longitude and latitude of the epicenter, and lon_z and lat_z are longitude and latitude of the assigned location, respectively. The latitude and longitude are transformed from the angle value to the radian value. We discover that sine and cosine of latitude or longitude are independent between the

assigned location and the epicenter mutually. Thus these trigonometric functions can be calculated in advance.

In the example, the program will compute 4.25 billion times of d value in total, and each computation will call trigonometrical function 5 times, so that there will be 21.25 billion times of trigonometrical functions calls. However, the assigned location only has 651 longitudes and 351 latitudes; and there are 18,604 historical epicenters, thus 18,604 latitude values and 18,604 longitude values in total. Therefore, only 76,420 times of trigonometrical function calls are needed for all 38,210 of the sine and the cosine values of latitude and longitude values, which is just 1/2,780,000 of original calls.

In the procedure, another long-time costing operation is the calculation of the assigned location's loading or the unloading response in the earthquake regions. The following equation is used,

$$E = \sqrt{10^{(11.8+1.5M)}} = 10^{(11.8+1.5M)/2} = 10^{5.9+0.75M}, \tag{4}$$

where M is Richter magnitude scale. E must be computed once for each location in the earthquake spatial regions. In the present procedure, the LURR is only related to the Richter magnitude scale, namely, LURR of an earthquake is a constant and needs to be calculated once only. In the example, LURRs need to be computed 22,826,006 times, but after optimization only 18,604 times are needed.

2.2. *Eliminating Unnecessary Computation*

In each point, when the LURR of tempo-spatial scanning is computed, the program will judge this point whether in the spatial scope of the earthquake first, and then call formula (3) to calculate the distance between this point and the epicenter. But an earthquake spatial scale is very small compared to the entire scanning region. Therefore, a place will be judged whether it is in the spatial scope of the earthquake by comparing its latitude and longitude to the epicenter's. If it is not in the scope, its distance will not be computed. Obviously, this will save computing time.

In Figure 2, the pink region is the spatial scope of the earthquake. The latitude and longitude of earthquake spatial scope which is the green region that is obtained by the earthquake spatial scale R and the latitude and longitude of the epicenter; it is circumscribed quadrilateral of the earthquake spatial scope. Because it is on the spherical surface, the pink region is not circularity but the upper part is wider than the lower part, and the green region is not a square. In order to guarantee that the green region can contain the entire pink region, a toleration is given while computing. For a point, the program will test the latitude and longitude to judge this point as to whether it is in the green region first. If not, d will not be computed, as 'Spot 2' shown in Figure 2. If in the green region, the distance d between the point and epicenter can be calculated. Then the program will judge the point as to whether it is in the pink region based on d or R. If it is in, the LURR is calculated, as 'Spot 1' shown in Figure 2. This can save massive time to compute the spherical surface distance. In the example, there are more than 4.25 trillion times of

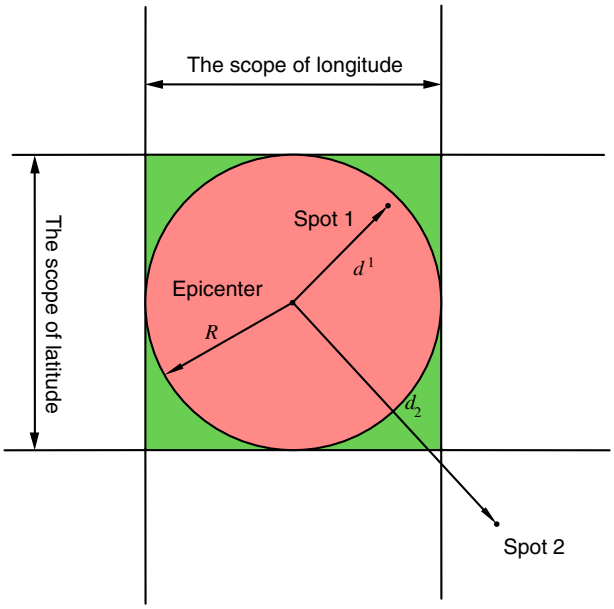

Figure 2
Estimating some point in the earthquake region. The quadrilateral is the spatial scope of being scanned; the pin region is the spatial scope of earthquake.

judging the point as to whether it is in the earthquake spatial scope, after optimization, the times of spherical surfaces distance computation is 32,092,365, which is just 1/132 of the original times.

2.3. *Improving Algorithm Formula*

In the procedure, to calculate the distance between two points on the spherical surface will require a lot of time. Thus, it is possible to make some improvement to obtain the same or the approximate result by using less operation. The spherical surface is approximated to the plane in the original formula to compute the distance d. The times of calling trigonometrical functions are 5, the times of calling the extraction of square root are one, and the times of floating point multiplications are 7, the times of floating point plus-minus operations are 5. The exact formula of the distance d of two points on the spherical surface is given as the following:

$$d = R_e \times \arccos\left(\sin(lat_z)\sin(lat_c) + \cos(lat_z)\cos(lat_c)\cos(lon_z - lon_c)\right). \quad (5)$$

In this formula, the times of calling trigonometrical functions are 5, the times of call anti-trigonometric functions are one, and the times of floating point multiplications are 4, the times of floating point plus-minus operations are 2.

After the approximate formula is replaced by equation (5), the times of floating point multiplications decrease 3 and the times of floating point plus-minus operations decrease 3, as well as the extraction of the square root which is replaced by inverse cosine function; the operating times are reduced. The LURR effects a slight change, whereas the, results of earthquake prediction cannot be affected.

2.4. Parallel Computing

Based on MPI (Message Passing Interface) library, we parallelized computing of Temporal and Spatial Scanning of LURR. The actual spatial region is divided into small spatial regions, each of which is computed by one processor. There are many different methods by which to partition the actual spatial region, such as, block distribution, cyclic distribution, block-cyclic distribution, and continue.

The computation volume of each point in the spatial region is different; in particular, after the program is optimized by several steps above, the computation mainly concentrates in the earthquake spatial scope. According to the block distribution, there are different volumes of computation in each small region which can cause the load imbalance between processors, and decrease the parallel performance of the procedure. If the cyclic distribution is adopted, two neighboring longitudes and latitudes can be computed by the different processor, and it makes the load balanced and boosts the procedure performance. The nonoverlapped domain decomposition method is used; this is because the partitioned small domains do not tie up each other. The communication amount is decreased on the boundary.

Because the data of the each step of the longitude and the latitude scanning to be computed are recorded, considerable data need high frequency storage which will require large amounts of communication time dealing with the data reduction. In order to solve this problem, many temporary files are opened to record data by each processor separately. Once all processes are finished, the output document will be written by one main processor.

3. Performance Testing

We have tested the original program and the optimized procedure in two kinds of different platforms, respectively. One platform is that CPU is Pentium 4 E 2.8 GHz, with level 1 cache of 32 KB. Level 2 cache of 1 MB, and memory of 384 MB. The operating system is Windows XP, and the compiler is Intel Visual FORTRAN Compiler 9.0 for Windows. We used optimization compiling option "/Ox".

Another platform is DeepComp 6800 Cluster which has 265 nodes, and each of them is four Itanium II 1.3 GHz CPU, with level 1 cache of 32 KB, level 2 cache of 256 KB, level 3 cache of 3 MB, and memory of 8 GB or 16 GB, Network is QSnet. The bandwidth of point-to-point communication is bigger than 300 MB/s, the latency is less

Table 1

Running times of eliminating before and after redundant computation on different platforms

Platform	Before	After
Pentium 4 E 2.8 GHz	1133.766 s	78.625 s
Itanium II 1.3 GHz	1328.989 s	146.037 s

than 7 microseconds. The operating system is Red hat Linux Advanced the Server 2.1 for IA-64, and the compiler is Intel FORTRAN Compiler 7.1 for IA-64. The optimization compiling option "-O3" is used.

3.1. Eliminating Redundant Computation

Table 1 shows running time before and after eliminating redundant computation on different platforms.

After the first step of optimization, on the Pentium 4 platform, the procedure running speed is 14.4 times faster than the procedure without optimization. On the Itanium II platform, the procedure running speed boosting is slightly lower, which is 9.1 times faster than without optimization. This indicates that the floating point performance of the Itanium II processor is extremely formidable. Therefore the speed boosting by the eliminating redundant floating point computing is small.

3.2. Eliminating Unnecessary Computation

Table 2 shows running time before and after eliminating unnecessary computation on different platforms.

After the second step of optimization, the procedure running speed is further enhanced, on the Pentium 4 platform; of the procedure running speed which is 2.64 times faster than the procedure of first step of optimization, is 38 times faster than the procedure without optimization, On the Itanium II platform, the procedure running speed which is 20.2 times faster than the procedure without optimization is 2.22 times faster than the procedure of the first step of optimization.

Table 2

Running times of before and after eliminating unnecessary computation on different platforms

Platform	Before	After
Pentium 4 E 2.8 GHz	78.625 s	29.828 s
Itanium II 1.3 GHz	146.037 s	65.908 s

Table 3

Running times of before and after improving the algorithm formula on different platforms

Platform	Before	After
Pentium 4 E 2.8 GHz	29.828 s	27.172 s
Itanium II 1.3 GHz	65.908 s	65.535 s

3.3. Improving Algorithm Formula

Table 3 shows running time before and after improvement of the algorithm formula on different platforms.

After the third step of the optimization, running speed of the procedure is enhanced slightly. On the Pentium 4 platforms, the running speed of the procedure is 41.7 times faster than the procedure without optimization. It is 9.77% faster than the previous step of using the approximate method procedure. On the Itanium II platform, the running speed is 20.3 times faster than the procedure without optimization, and it is 0.57% faster than the previous step of using the approximate method procedure. Because floating point operations are decreased similarly in this step, and decreased modestly, therefore, considering the floating point processing performance formidable Itanium II, the performance enhancement might be ignored.

3.4. Parallel Performance Testing

Performance testing of the parallel program is run on the DeepComp 6800 cluster. The running time on 1, 2, 4, 8, 16, 32 and 64 processors is tested. Four types of data distribution are used, including 1D block, 2D block, 1D cyclic and 2D cyclic. From Table 4, the performances using cyclic distributions are observably higher than those using block distribution.

Parallel speedup can be computed by the following formula:

Table 4

Running times of using 4 types of data distribution

Number of Processors	Running Time (sec)			
	1D block	2D block	1D cyclic	2D cyclic
1	66.960	66.960	66.960	66.960
2	38.764	38.764	34.612	34.612
4	22.119	20.504	18.368	18.438
8	12.665	12.264	10.373	10.286
16	7.804	7.743	6.266	6.258
32	5.143	5.410	4.400	4.436
64	3.885	4.501	3.834	3.637

Table 5

Speedups using 4 types of data distribution

Number of Processors	Speedup			
	1D block	2D block	1D cyclic	2D cyclic
1	1	1	1	1
2	1.727	1.727	1.935	1.935
4	3.027	3.266	3.645	3.632
8	5.287	5.460	6.455	6.510
16	8.580	8.648	10.686	10.700
32	13.020	12.377	15.218	15.095
64	17.236	14.877	17.465	18.411

$$\text{Speedup}(N) = \frac{T_1}{T_N}, \tag{6}$$

where T_1 is the running time on 1 processor, T_N is the running time on N processors. The speedups using 4 types of data distributions are listed in Table 5.

From this table, the speedups are much lower than the number of processors, especially when the number of processors is large. According to AMDAHL's law (1967), a program consists of 2 parts: the serial part and the parallel part. Assume in a program, the ratio of the serial part is α, the running time of the serial part is αT_1, the running time of the paralleled part is $(1-\alpha)\ \alpha T_1/N$, then the speedup formula can be expressed as:

$$Speedup(N) = \frac{T_1}{\alpha T_1 + \frac{(1-\alpha)T_1}{N}} = \frac{1}{\alpha + \frac{1-\alpha}{N}}. \tag{7}$$

From this formula we can see that the ratio of the serial part is larger, speedup is smaller, when the number of processors $N \to \infty$, the limitation of speedup is $1/\alpha$. In this program, the ratio of the serial part increases after optimization. According to the test on DeepComp 6800, the ratio of the measurable serial part is about 3%, thus the theoretical speedups are listed in Table 6.

The premise of these theoretical speedups is full load-balancing and omitting parallel overhead, such as synchronized and communication overhead. In Figure 4, speedup curves using 4 types of data distribution, ideal speedup and theoretical speedup are drawn.

From Figure 4, speedups using cyclic data distributions are close to the theoretical values. This shows that their load-balancing is better than block distributions; and the speedups using 1D cyclic distribution and 2D distributions have no significant difference,

Table 6

Theoretical speedups by Amdahl's law

Number of Processors	1	2	4	8	16	32	64
Theoretical speedup	1	1.942	3.670	6.612	11.034	16.580	22.145

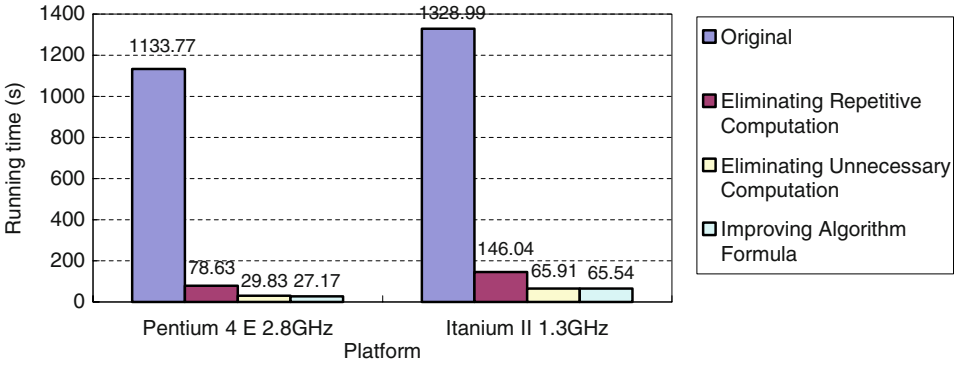

Figure 3
Running time of each step by the optimized procedure in two kinds of platforms.

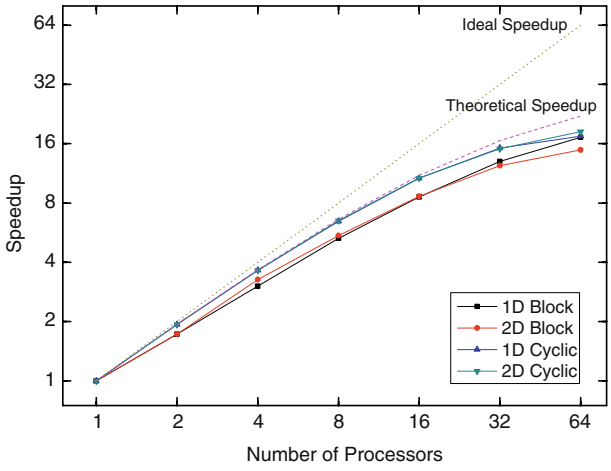

Figure 4
The testing result of parallel speedups.

when the number of processors is more than 32, 2D cyclic distribution is better. In 2 types of block distribution, when the number of processors is not more than 16, 2D block distribution's load-balancing is better than 1D block distribution's, but if the number of processors is more than 32, 1D block distribution's load-balancing is better.

4. *Conclusion*

After the parallel tempo-spatial scanning program is optimized, its running speed increases 20 to 40 times. With the optimization and parallel implementation of LURR, a wider range, longer time and higher accuracy earthquake prediction are possible. In

addition, this program requires less computation power, which makes it less dependent on high performance computers and decreases running cost.

Acknowledgements

This research was funded by the Ministry of Science and Technology of China (Grant No. 2005DKA64000-2), National Natural Science Foundation of China (Grant No. 60533020) and Research on high performance scientific computation (973 programs, Grant No. 2005CB321702)

References

Amdahl, G.M. Validity of the single-processor approach to achieving large scale computing capabilities. In *AFIPS Conf. Proc. 30* (Atlantic City, N.J., Apr. 18–20). (AFIPS Press, Reston, Va., 1967), pp. 483–485.

Mora, P., Place, D., Wang, Y., Yin, X., Peng, K., Weatherley, D., (2000), *Earthquake Forecasting: Retrospective Studies in Australia — the Newcastle and Burra Earthquakes and Numerical Simulation of the Physical Process, AEES* (Australian Earthquake Engineering Society) Annual Meeting, November, 2000, Hobart, Australia.

Yin, X.C., *A new approach to earthquake prediction*, Earth. Res. in China *3*, 1–7, (1987).

Yin, X.C., Chen, X.Z., Song, Z.P., and Yin, C. (1995), *A new approach to earthquake prediction: the Load/Unload Response Ratio (LURR) theory*, Pure Appl. Geophys. *145*, 701–715.

Yin, X.C., Wang, Y.C., Peng, K.Y., and Bai, Y.L. (2000), Development of a new approach to earthquake prediction: Load/Unload Response Ratio (LURR) theory, Pure Appl. Geophys. *157*, 2365–2383.

Zeyao Mo and Yuan Guoxing, *Parallel Programming Environment MPI (in Chinese)*. ISBN 7-03-009805-6 (Science Press 2001).

Zhang Yongxian, Yin, Xiangchu, and Peng Keyin, (2004), *Spatial and temporal variation of LURR and its implication for the tendency of earthquake occurrence in southern California*, *Pure Appl. Geophys 161*, 1–9.

Yin, Xiangchu *et al*. (1996) *The temporal variation of LURR in Kanto and other regions in Japan and its application to earthquake prediction*, *Earthq. Res. in China 10*(4), 381–385.

Zhuang, J.C. and Yin X.C. (1999), *Random distribution of the Load/Unload Response Ration(LURR) under Assumptions of Poisson Model*, Earthq. Res. in China *15*, 128–138. .

(Received March 21, 2007; revised March 21, 2007, accepted June 28, 2007)
Published Online First: April 2, 2008