JOURNAL OF CIVIL ENGINEERING AND MANAGEMENT
ISSN 1392-3730 / eISSN 1822-3605
2014 Volume 20(4): 477–484
doi:10.3846/13923730.2013.801917

PARALLEL FRAMEWORK FOR EARTHQUAKE INDUCED RESPONSE COMPUTATION OF THE SDOF STRUCTURE

Sarfraz MUNIR[a], Raja Rizwan HUSSAIN[b], A. B. M. Saiful ISLAM[c]

[a]*The Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong*
[b]*Department of Civil Engineering, College of Engineering, King Saud University, Saudi Arabia*
[c]*Department of Civil Engineering, University of Malaya, Kuala Lumpur, Malaysia*

Received 11 Mar 2012; accepted 20 Apr 2012

Abstract. Parallel computing briskly diminishes computation time through simultaneous use of multiple computing resources. In this research, parallel computing techniques have been developed to parallelize a program for obtaining a response of single degree of freedom (SDOF) structure under earthquake loading. The study uses Distributed Memory Processors (DMP) hardware architecture and Message Passing Interface (MPI) compilers directives to parallelize the program. The program is made parallel by domain decomposition. Concurrency in the program is created by dividing the program into two parts to run on different computers, calculating forced response and free response of the first half and the second half. Parallel framework successfully creates concurrency and finds structural responses in significant lesser time than sequential programs.

Keywords: parallel computing, SDOF, parallel framework, earthquake response, parallelization, computation cost.

Reference to this paper should be made as follows: Munir, S.; Hussain, R. R.; Islam, A. B. M. S. 2014. Parallel framework for earthquake induced response computation of the SDOF structure, *Journal of Civil Engineering and Management* 20(4): 477–484. http://dx.doi.org/10.3846/13923730.2013.801917

Introduction

Parallel computing is simultaneous use of multiple computing resources to solve a computing problem in a reduced computation time. Such parallel computing is the best way to increase the computation speed and reduce run time of a program. Its application is rapidly increasing in the field of scientific and engineering computations (Chan *et al.* 2005; Polychronopoulos 1988; Wilkinson, Allen 2005). The approach surprisingly makes the calculations and simulations time efficient based on multi-core processors in addition to clusters. All major applications in different fields of science and technology are employing parallel computing for high speed computations and getting results in lesser time. Most desktop and laptop systems now ship with dual-core microprocessors or quad-core processors, which are best suited for parallel computing. Parallel programming is different from sequential computing, in which a problem/program is divided into a number of small instructions or tasks called threads, which are executed in the processor one by one.

To make many processors simultaneously work on a single program, the program must be divided into smaller independent chunks so that each processor can work on separate chunks of the problem (Wilkinson, Allen 2005). Distributed Memory Processors (DMP) is the computer hardware architecture, which is used in the research. DSM systems require a communication network to connect inter-processor memory (ordinary computers). To parallelize programs on DMP, the Message Passing Interface (MPI) compilers directives have to be used. In this case first find or create concurrency in the program and then by using MPI commands send concurrent parts of the program from the master computer to different computers; different computers perform operations on it and send back the results to the master computer, which is arranged and written by master computer (Polychronopoulos 1988).

The specialty of this research is that the master computer acts as a worker during the computation time and then it also collects the results from the worker, arranges the data and at the end displays the results. Parallel computing techniques (Fig. 1) have been implemented to parallelize a program for the response of the single degree of freedom structure against typical earthquake force. The program aimed at finding a response of the SDOF structures is created concurrently in the program by dividing the program into two parts, finding forced response and free response of the first half and the other part. Each part is executed on a different computer; and as SDOF Structure is linear and homogeneous, results can be added. In this way, the program for finding a response of the SDOF structure is parallelized. Time reduction in parallel programs can be seen over sequential programs in the presented results. Using multiple computing devices

Corresponding author: A. B. M. Saiful Islam
E-mail: abm.saiful@gmail.com

successfully created the concurrency, found the response of structure against earthquake force in lesser time than that of in case of sequential programs.

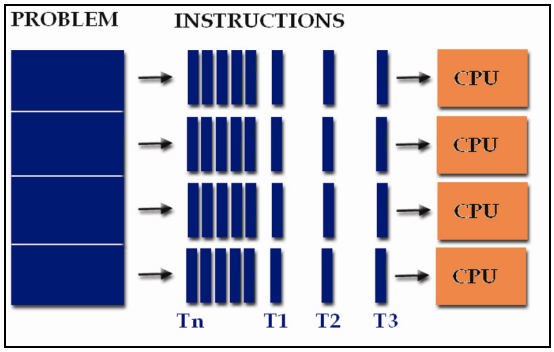

Fig. 1. Schematic representation of Parallel Computing

1. Parallel computing in earthquake engineering

Starting in the late 80's, clusters came to compete and eventually displace the concept of supercomputers for many applications. A cluster is a type of parallel computer built from large numbers of off-the-shelf computers connected by an off-the-shelf network. Today, clusters are the workhorse of scientific computing and are the dominant architecture in the data centres that power the modern information age (MPICH2 2006). Parallel and distributed computations can be efficaciously implemented in structural mechanics (Bittnar *et al.* 2001).

Frequent earthquakes that occur all over the world cause gigantic threat on structures (Islam *et al.* 2012a, b; Jameel *et al.* 2012a). Structural responses substantially vary with the occurrence of ground excitation induced from severe earthquake (Islam *et al.* 2012c; Jameel *et al.* 2012b). Therefore, the computation of seismic responses of structures is of utmost important (Islam *et al.* 2013; Lin *et al.* 2011). Ólafsson, Sigbjörnsson (2011) presented digital filters for simulation of seismic ground motion and structural response. Uma *et al.* (2010) developed probabilistic framework for performance-based seismic assessment of structures considering residual deformations. The equivalent force control method for real-time testing of nonlinear structures has been proposed by Wu *et al.* (2011) and Stochastic Modeling by Yazdani, Abdi (2011).

Goda *et al.* (2009) studied probabilistic characteristics of seismic ductility demand of SDOF systems with Bouc-Wen hysteretic behaviour. Grand challenges are the computational problems, which can't run on one computer because of memory restraint or problems that take too much time to execute on a single computer, such as applied fluid dynamics, macro-scale environmental modelling, ecosystem simulations, weather forecast, biomedical imaging and biomechanics, earth quake & plate tectonics, molecular design and process optimization, nuclear power and weapons simulations, and strong artificial intelligence (Janies, Wheeler 2001; Čiegis *et al.* 2006). For such computations, parallel computing is readily used by, for example, the Earth Simulator Center (Bruck *et al.* 1997).

Parallel computing is used in the field of earthquake engineering as well (Zhong *et al.* 2003). It is worth mentioning that the computer aided analysis of complex structural model poses economic solution in time cost in advanced structural mechanics (Jameel *et al.* 2012c, d; Šliseris, Rocēns 2010; Vaidogas, Šakėnaitė 2011). Stability and ductility of structures are vital issues to be analysed in an accurate fashion with the rapid solution approach (Kvedaras 2010; Rasiulis, Gurkšnys 2010). Parallel computing is a competent tool used in optimization of earthquake response, selection of equivalent earthquake wave and optimization of damaged response index. This solution technique is used in getting the response of structure, which is rather difficult and is still in under progress (Gropp *et al.* 2007).

In the present study, parallel computing has been used to get the response of the simplest structure, i.e. the single degree of freedom structure against earthquake force. For the program of finding response of the SDOF structures concurrency in the program is created by dividing the program into two parts finding the forced response and the free response of the first half and the other part. Each part is executed on a different computer. As the SDOF Structure is linear and homogeneous, results can be added. In this approach, the program used for obtaining the responses of the SDOF structure is parallelized.

Parallel computing provides an effective utilization of multi-core processors as well as old computers, so its usage is becoming a more popular computational technique at present (Hossain *et al.* 2002). Chip manufacturers have begun to increase overall processing performance by adding additional CPU cores. The reason is that increasing performance through parallel processing can be far more energy-efficient than increasing microprocessor clock frequencies. In the world, which is increasingly more mobile and energy conscious, this has become essential (Quinn 2004).

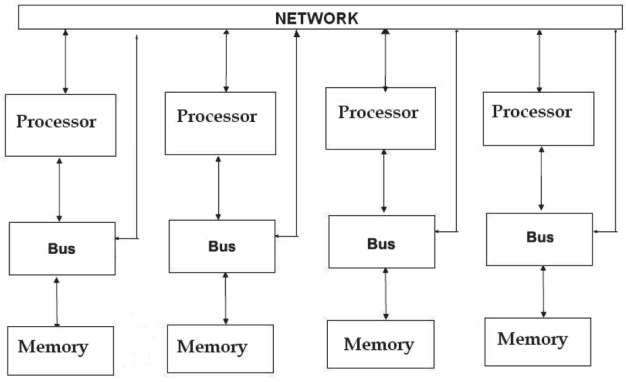

Fig. 2. Schematic representation of Distributed Memory Computers

2. Technique of parallel computing

To execute programs in parallel, there are certain requirements, such as parallel computers/processors, an operating system (Linux preferred), a high level programming language, such as C or Fortran, the parallel application programming interface (MPI), and above all –

a parallel algorithm. The author had used two computers connected by a high speed network, i.e. ether net, with LINUX 9 on each computer, programming language used was Fortran 77 with the latest MPICH version, in which one computer acted like the Master computer and the other one was a worker. It is to mention here that the Master computer also worked as a worker after distribution of the tasks to workers and then it also got the results from the Worker and displayed them after organization (Strout *et al.* 2006; MPI 1997).

MPI is parallel application programming interface (API), it is the sets of routines, data structures, object classes and/or protocols provided by libraries in order to support building of a parallel application. To make many processors simultaneously work on a single program, the program was divided into smaller independent chunks so that each processor could work on separate chunks of the problem. This is the foremost theme of parallel programming (MPI 1997). This way decomposed program is a parallel program and then it can be run on parallel architecture computers.

Parallel computing techniques depend on the computer hardware architecture, nature of the program and the number of available computers/processors (Kurc, Ozmen 2008). In this research, the main emphasis of parallel computing techniques is the distributed memory architecture (Fig. 2). There are several ways to develop such architectures like making clusters, or by installing different programs on multiple computers, which supports parallel programming. It is also within the scope of the study to create a parallel computing system from old ordinary computer with the help of the Local Area Network (LAN). A computer cluster is a group of linked computers, working together closely, so that in many respects they form a single computer. The components of a cluster are commonly, but not always, connected to each other through fast local area networks. These are usually used for a computational purpose.

Another very important trait of such computers is that these computers sustain their Identities. So, those computers are ordinary computers most of the time and when anyone wants to run any parallel application, those computers act as a single computer at that time. The idea of parallelization using the MPI is that we divide the problem into independent parts and data, which is required to operate that part independently, then it is shared and communicated with the help of MPI subroutines to different computers. So, the computers operate on different parts of a program independently and send back the results to the master computer, which assembles and displays the results (Pacheco 1998).

The main part of parallel computing is to exploit or to create the concurrency in the program, share it to the other computers/processors using the parallel application programming interface (API), such as the MPI, so that other computers can act upon the data (Peterson, Arbenz 2004). Some programs have inherent concurrency, which is easy to parallelize like some loops having different variables to operate upon; while, in some programs we can create concurrency, such as matrix operations; but some programs cannot be parallelized, such as the program to find Fibonacci series (Wittwer 2006).

The parallel program accomplishes the following code:

```
        program hello
include 'mpif.h'
integer rank, size,
ierr,tag,status(MPI_STATUS_SIZE)
call MPI_INIT(ierr)
call MPI_COMM_SIZE(MPI_COMM_WORLD,
size, ierr)
call MPI_COMM_RANK(MPI_COMM_WORLD,
rank, ierr)
print*, 'Hello world from process', rank, 'of ', size
call MPI_FINALIZE(ierr)
end
```

In this code, MPI subroutines will initiate the parallel environment, get the size (Numbers) of parallel computers, assign the numbers to parallel computers starting from 0 for masters and size – 1 to workers and then terminates the parallel environment respectively. The Master computer then displays the print statement, which is in case of two computers in a parallel cluster (Kurc, Will 2006):

```
Hello world from process 0 of 2
Hello world from process 1 of 2
```

Following next are important points, which have been followed to develop the parallel algorithm for parallel programming:

1) Design, implementation, and tuning of the parallel algorithm should be such that it takes full advantage of parallel computing systems;
2) Partitioning the overall problem into separate tasks and allocating tasks to processors should uphold all computers busy equally;
3) The parallel algorithm can apply to the problems that are inherently parallelizable, mostly without data dependence;
4) Communication time should be as small as possible;
5) Communication always implies overhead, so try to reduce the communication by creating course granular partition of data;
6) All tasks should be kept busy, data should be evenly distributed;
7) Primary inhibitors to parallelism are loops, which usually the most common target of parallelization effort, so the first loop dependency should be analysed and then tried to parallelize that, if possible;
8) If work involves much I/O it is not suitable for parallelizing because I/O operations take nearly 10X more time than any operation;
9) Input and output should be displays only on master computer, i.e. computer, which has ID 0.

3. Parallelization of the SDOF program

The program has successfully been parallelized for response of the SDOF structure against earthquake. The numerical method, which the author used to calculate response of the SDOF structure is the Runge–Kutta Method. The Runge–Kutta Method is the initial value problem and it doesn't have any inherent parallelism. But due to the linear property of the SDOF structure, we can induce parallelism in the program. The main idea to parallelize such program is that we can find the free response as well as the forced response of the SDOF structure separately and then according to the principle of the super position we can add them up.

To make this parallel computing technique more general, effective and applicable to most programs, the author tried to parallelize the program using subroutines. As mostly programming for structural analysis and responses developed using different subroutines. We use several subroutines in a complicated problem and in the main program we just call these subroutines. The author found the technique to effectively call the same subroutine in the parallel program from different computers. We can apply the same method to parallelize a similar program.

These are the steps, which should be followed in making a parallel program for the response of the SDOF structure:

1) Split the input ground acceleration history into two equal parts as the author wanted to run that parallel program on two computers;
2) Input data is divided in such a way that we have some free response of the first part exactly after half of the computations, which can be added to forced response of the second computer to get the exact result. Normally, this extra computation for the free response in the first half is kept at 200 units;
3) Separate the computations from the subroutines, which required executing only once;
4) Send the subroutine parameters from computer 1 to computers 2;
5) Call the subroutine in the main program simultaneously from both computers;
6) Watch the processor history so that we can observe that both computers are busy in computations;
7) Format the calculations in such a way that we have acceleration, velocity and displacement responses easily in the form of graphs;
8) Compare results with that of the single computer calculations.

4. Framework for the parallel program

The author successfully ran the parallel program for the structure having typical values of damping, natural circular frequency, and input of the sine wave. The calculations from Computer 1 and Computer 2 from the same parallel program is obtained and shown in Figures 3 and 4, respectively. Then these results added to get the complete response of the SDOF structure, which is also shown in Figure 5.

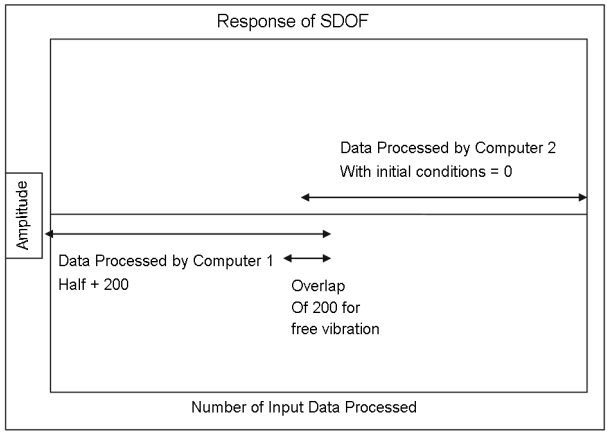

Fig. 3. Parallel framework for response of SDOF structure

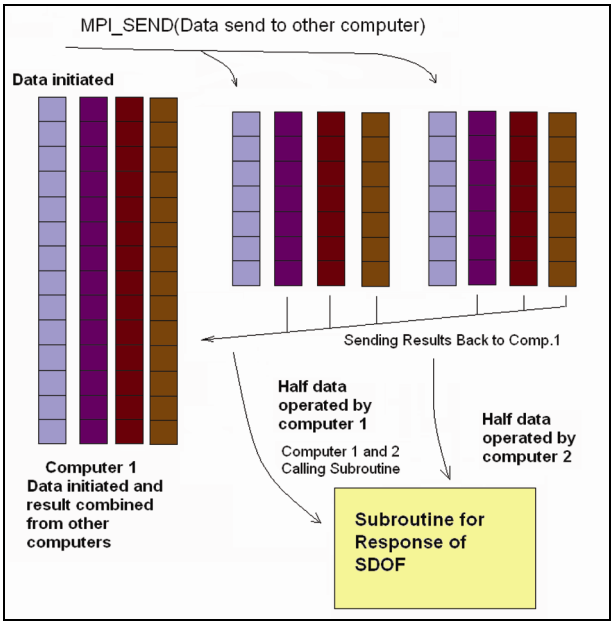

Fig. 4. Schematic representation of Parallel Program for response of SDOF structure

The response, which is received by parallel computing is then compared with the results from the equivalent sequential program, which is shown in Figure 6, and the difference between the results from the parallel program and the equivalent sequential program is shown in Figure 7. The difference between the results from the parallel program and the equivalent sequential program is negligible, which can be observed easily. The schematic representation of the parallel program for the response of the SDOF has been displayed in Figure 8.

This was one way to parallelize the program for the response of the SDOF structure by dividing all the computations into half, which is executed of different computer and later joined the results. There is another method of parallel programming, in which we divide the calculations. For example, a loop first calculates the sum of both variables in it and then calculates the square of resultants. Suppose we are able to calculate the sum on one computer and then square it on the other computer. This is another way to parallelize the problem but it is very difficult.

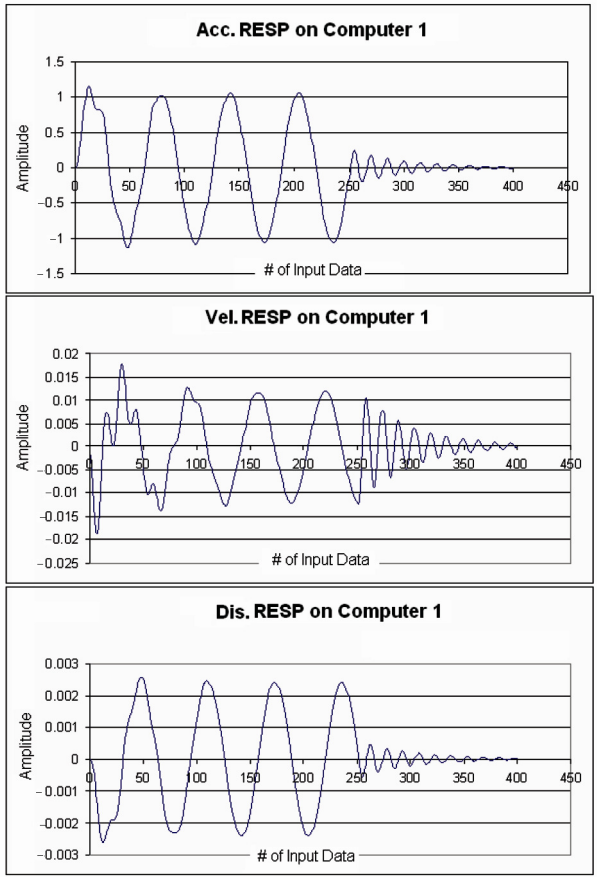

Fig. 5. Response of SDOF Structure from Computer 1

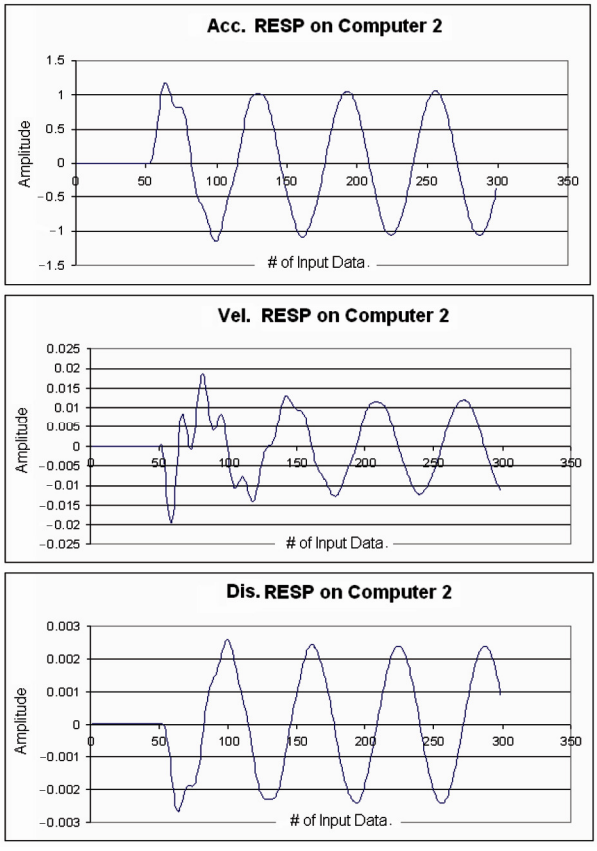

Fig. 6. Response of SDOF Structure from Computer 2

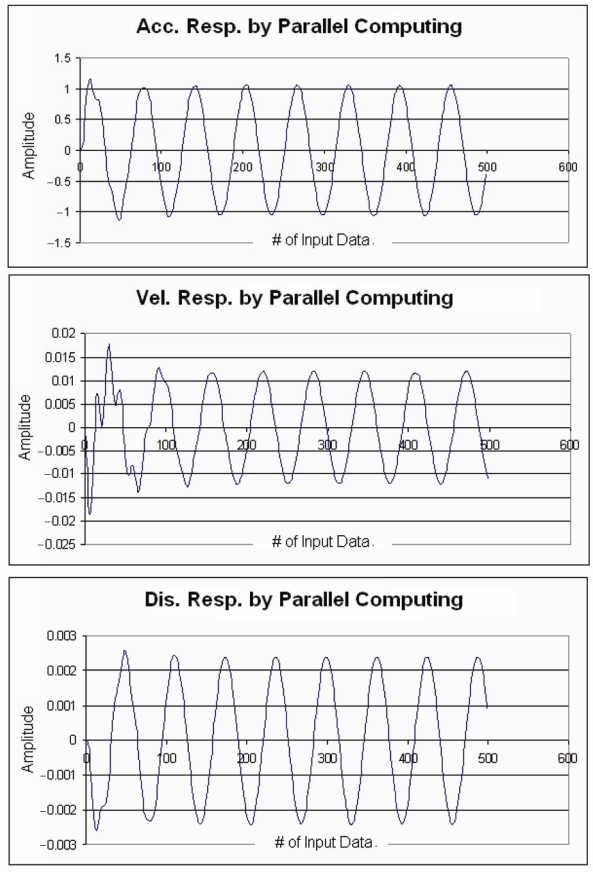

Fig. 7. Response of SDOF Structure from Parallel Program

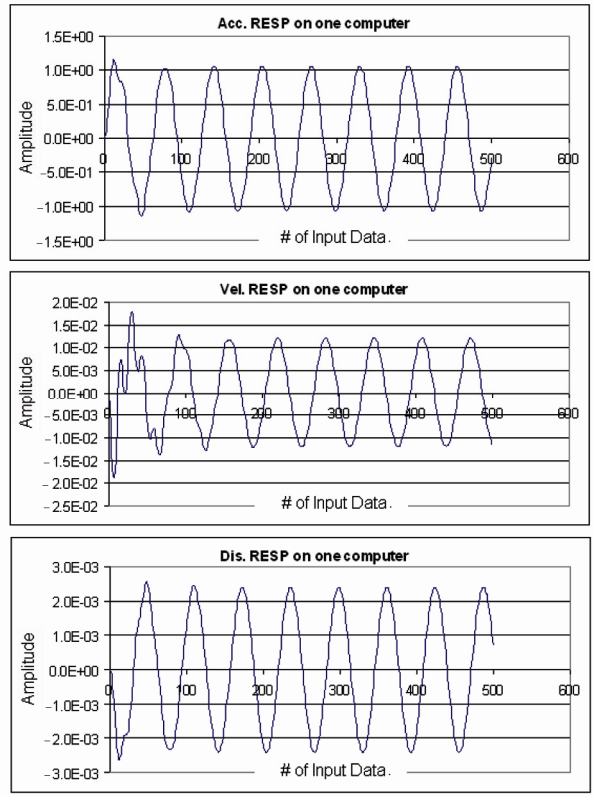

Fig. 8. Response of SDOF Structure from Sequential Program

First of all, such computations are highly dependent, which is very difficult to parallelize; secondly in such parallelization, one computer may have little computational effort compared to the other, so effective parallelism can be very difficult to achieve.

5. Results and discussions

The program for the response of the SDOF structure ran on two computers in parallel and reduced the run time of the program. Another important point in this research is that the Master computer also worked as a worker. Actually, in most of the parallel programs, the main computer acts as the Master computer and only distributes the data to be parallelized and then waits for the results, but in this research, the main computer distributes the data and then works on its part of data, then receives results and displays them. So, if we have small number of computers in the parallel computer cluster then the Master computer can also work as worker.

Moreover, the parallel computing technique used in this research was also very convenient and workable. In the program of finding response of the SDOF structures, concurrency in the program has been created by dividing the program into two parts finding the forced response and the free response of the first half and the other part. Schematic representation of the parallel computing technique is shown in Figures 3 and 4. Each part is executed on a different computer and as the SDOF structure is linear and homogeneous, the results can be added. In this way, the program for finding the response of the SDOF structure is parallelized. Time reduction in parallel programs is observed over sequential programs in the obtained results.

The use of processors of both computers has been checked during the runtime and processors use of both computers showed 100% use of processors by the program. The runtime is compared for the parallel program with similar sequential programs for the response of the SDOF structure. And the runtime of the parallel program reduced to nearly 2/3 of that of the sequential program. The reasons for this reduction of the runtime instead of making it half then that of the sequential program are: the coarseness or finesse of the parallel data, communication time between the parallel computers and computational efforts. There can be little drawbacks of the parallel computing like; latency in the network communications, steep learning curve, some programs have no concurrency like Fibonacci series, fineness in concurrent data. It is mostly useless to parallelize the program with high fineness.

The courser the data will be, the more reduces the runtime of the program will be. And if the data division is fine then the runtime of the program will be less reduced. Course data means that a large amount of data can send in single communication; and fine data means a small amount of parallel data send to other parallel computer. The 2nd factor, which affects the results, is the communication time between the parallel computers. Data communicates between the computers via the Local Area Networks (LAN). The maximum communication speed is determined by the Switch we are using in the parallel computing. So we can have a 100 MB or 1 GB switch but even those have very small speed as compared to processor speed available nowadays. Generally, the data communication takes nearly 10 times more delay than that of processing. If data is needed to be sent to different computers in the cluster many times, then the runtime will also be affected. So, if we have to send small amounts of data many times in a parallel program, then such parallel program can take more time to show the results compared to sequential programming.

Computational effort is also made nearly equal in this research. So, the efforts have made course data parallel program, minimum communication between the data (only for distribution of the data and receiving the results) and nearly equal computational efforts. For comparison of a parallel and a sequential program, the computations should be reasonable, because for small computational efforts programs, the run time of parallel as well as sequential programs are nearly equal even in some cases the runtime of a parallel program can be more than that of a sequential program because of data communication between computers.

The study uses a reasonable amount of earthquake input data so that they can easily compare the results from the sequential as well as the parallel program. So, to make appropriate computational efforts, the author enlarged the data to be processed by the processors. The author introduced large computations of about 200,000 iterations of the loop. This then ran on parallel computers as well as equivalent sequential program, which also ran on a single computer.

The results from Computer 1 and Computer 2 were presented in Figures 5 and 6, respectively; and the sum of these two results is presented in Figure 7. Then, it is compared with the results from the sequential program shown in Figure 8. Those results are compared. The difference of calculations between the parallel as well as the sequential programming is very small and can be easily neglect as it is in the range of exponent –6 to exponent –9, as shown in Figure 9. Runtime comparison is shown in Figure 10. Time reduction in parallel programs can be seen over sequential programs in the presented results.

Conclusion

Parallel computing is the technology of the future; we can convert our computations from sequential computing to parallel computing for time efficiency of the program. As modern computing systems are multi-core computer systems, to fully utilize all the processors of modern computers we will have to use parallel programming and parallel computing. Moreover, parallel computing requires using multiple ordinary computers as a super computer.

Runtime of these parallel programs has been compared with equivalent sequential programs. But one important thing is that reduction of time by parallel computing is dominant in case of large computational efforts. For example, the run time for the parallel program for response of the SDOF structure is equal to equivalent sequential program, which is negligible in both cases. But when a large data is to be processed by the same parallel program and equivalent sequential program, then runtime

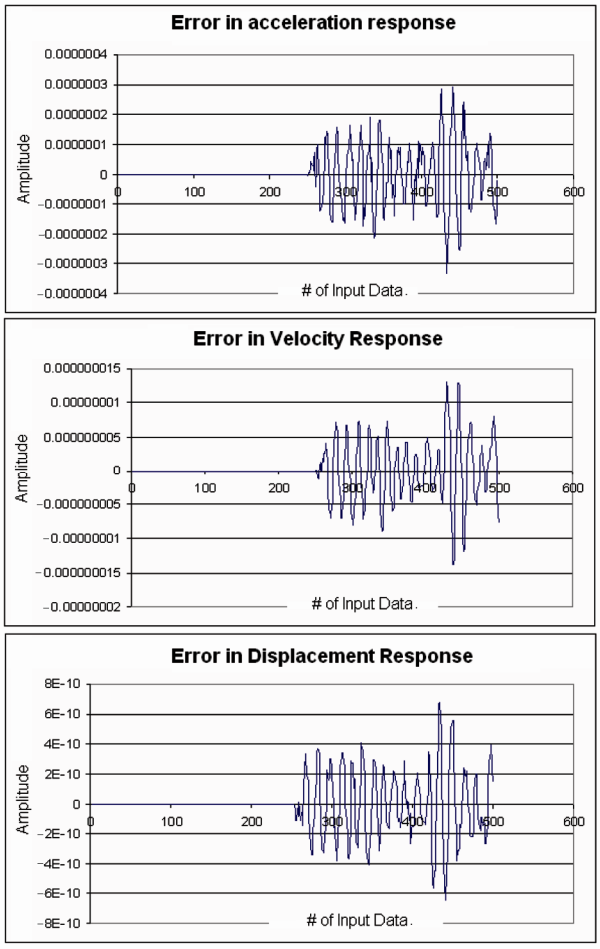

Fig. 9. Difference of results from Parallel Program & Sequential Program

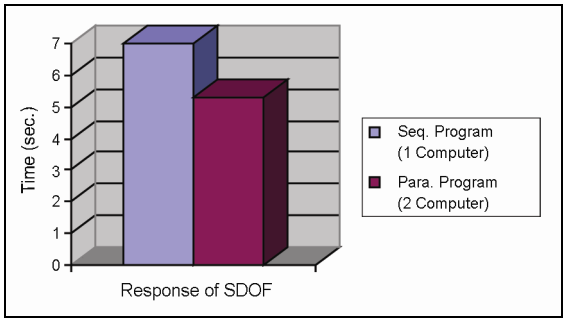

Fig. 10. Runtime comparison of sequential and parallel program

for the parallel program is nearly 2/3 than that of the sequential program.

The cheapest way of parallel computing is by using Distributed Memory Processors like a multiple computer as in my case where a parallel computer and then using the MPI libraries on Linux OS. Because it consists of cheap computers connected together and all software is available either absolutely free or at a low cost. It is fruitful to parallelize those programs having large computational efforts. As you can see, for ordinary programs or programs with little computational efforts, it takes equal time in sequential as well as in parallel programs. To make a program parallel, problem should have inherent concurrency. Some problems, which do not have concurrency cannot be parallelized like the program used to calculate Fibonacci series.

Reference

Bittnar, Z.; Kruis, J.; Němeček, J.; Patzák, B.; Rypl, D. 2001. Parallel and distributed computations for structural mechanics: a review, in Topping, B. H. V. (Ed.). *Civil and Structural Engineering Computing: 2001*. Saxe-Coburg Publications, Stirlingshire, UK, Chapter 9, 211–233. http://dx.doi.org/10.4203/csets.5.9

Bruck, J.; Dolev, D.; Ho, C.-T.; Roşu, M.-C.; Strong, R. 1997. Efficient Message Passing Interface (MPI) for parallel computing on clusters of workstations, *Journal of Parallel and Distributed Computing* 40(1): 19–34. http://dx.doi.org/10.1006/jpdc.1996.1267

Chan, A.; Gropp, W.; Lusk, E. 2005. *User's Guide for MPI Programs*. University of Chicago, Illinois: Aragon national laboratory. 34 p.

Čiegis, R.; Baravykaitė, M.; Belevičius, R. 2006. Parallel global optimization of foundation schemes in civil engineering, *Applied Parallel Computing. State of the Art in Scientific Computing* 3732: 305–312. http://dx.doi.org/10.1007/11558958_36

Goda, K.; Hong, H. P.; Lee, C. S. 2009. Probabilistic characteristics of seismic ductility demand of SDOF systems with Bouc-Wen hysteretic behavior, *Journal of Earthquake Engineering* 13(5): 600–622. http://dx.doi.org/10.1080/13632460802645098

Gropp, W.; Lusk, E.; Ashton, D.; Buntinas, D.; Butler, R.; Chan, A.; Ross, R.; Thakur, R.; Toonen, B. 2007. *MPICH2. User's Guide Manual*. University of Chicago, Illinois: Aragon national laboratory. 44 p.

Hossain, M. A.; Kabir, U.; Tokhi, M. O. 2002. Impact of data dependencies in real-time high performance computing, *Microprocessors and Microsystems* 26(6): 253–261. http://dx.doi.org/10.1016/S0141-9331(02)00027-3

Islam, A. B. M. S.; Hussain, R. R.; Jumaat, M. Z.; Rahman, M. A. 2013. Nonlinear dynamically automated excursions for rubber-steel bearing isolation in multi-storey construction, *Automation in Construction* 30: 265–275. http://dx.doi.org/10.1016/j.autcon.2012.11.010

Islam, A. B. M. S.; Jumaat, M. Z.; Hussain, R. R.; Alam, M. A. 2012a. Incorporation of rubber-steel bearing isolation in multi-storey building, *Journal of Civil Engineering and Management* 19(Supplement 1): S33–S49. http://dx.doi.org/10.3846/13923730.2013.801904

Islam, A. B. M. S.; Hussain, R. R.; Jameel, M.; Jumaat, M. Z. 2012b. Non-linear time domain analysis of base isolated multi-storey building under site specific bi-directional seismic loading, *Automation in Construction* 22: 554–566. http://dx.doi.org/10.1016/j.autcon.2011.11.017

Islam, A. B. M. S.; Ahmad, S. I.; Jumaat, M. Z.; Hussain, R. R. 2012c. Efficient design in building construction with rubber bearing in medium risk seismicity: case study & assessment, *Journal of Civil Engineering and Management* (in Press). http://dx.doi.org/10.3846/13923730.2013.801910

Jameel, M.; Islam, A. B. M. S.; Khaleel, M.; Amirahmad, A. 2012a. Efficient three dimensional modeling of high-rise building structures, *Journal of Civil Engineering and Management* 19(6): 811–822. http://dx.doi.org/10.3846/13923730.2013.799096

Jameel, M.; Islam, A. B. M. S.; Hussain, R. R.; Khaleel, M.; Zaheer, M. M. 2012b. Optimum structural modelling for tall buildings, *The Structural Design of Tall and Special Buildings* 22(15): 1173–1185. http://dx.doi.org/10.1002/tal.1004

Jameel, M.; Islam, A. B. M. S.; Khaleel, M.; Jumaat, M. Z. 2012c. Nonlinear finite element analysis of spar platform, *Advanced Science Letters* 13: 723–726. http://dx.doi.org/10.1166/asl.2012.3851

Jameel, M.; Ahmad, S.; Islam, A. B. M. S.; Jumaat, M. Z. 2012d. Nonlinear dynamic analysis of coupled spar platform, *Journal of Civil Engineering and Management* 19(4): 476–491. http://dx.doi.org/10.3846/13923730.2013.768546

Janies, D. A.; Wheeler, W. C. 2001. Efficiency of parallel direct optimization, *Cladistics* 17(1): S71–S82. http://dx.doi.org/10.1111/j.1096-0031.2001.tb00106.x

Kurc, O.; Ozmen, S. 2008. An efficient parallel solution framework for the linear analysis of large structures on PC Clusters, *Tsinghua Science and Technology* 13(SI): 65–70.

Kvedaras, A. K. 2010. Stability and ductility of structures: editorial, *Journal of Civil Engineering and Management* 16(2): 155–158. http://dx.doi.org/10.3846/jcem.2010.15

Lin, L.-k.; Chang, C.-c.; Lin, Y.-c. 2011. Structure development and performance evaluation of construction knowledge management system, *Journal of Civil Engineering and Management* 17(2): 184–196. http://dx.doi.org/10.3846/13923730.2011.576833

MPI 1997. *A message passing interface standard from message passing interface forum*. University of Tennessee, Knoxville, Tennessee. 245 p.

MPICH2 Installation and Configuration Guide 2006.

Ólafsson, S.; Sigbjörnsson, R. 2011. Digital filters for simulation of seismic ground motion and structural response, *Journal of Earthquake Engineering* 15(8): 1212–1237. http://dx.doi.org/10.1080/13632469.2011.565862

Pacheco, P. S. 1998. *A user's guide to MPI*. San Francisco. 51 p.

Peterson, W. P.; Arbenz, P. 2004. *Introduction to parallel computing – a practical guide with examples in C*. New York: Oxford University Press. 288 p.

Polychronopoulos, C. 1988. *Advanced loop optimizations for parallel computers, supercomputing*. Lecture Notes in Computer Science. Springer Berlin/Heidelberg, 255–277.

Quinn, M. J. 2004. *Parallel programming in C with MPI and OpenMP*. New York: McGraw Hill. 516 p.

Rasiulis, K.; Gurkšnys, K. 2010. Analyses of the stress intensity of the cylindrical tank wall at the place of the geometrical defect, *Journal of Civil Engineering and Management* 16(2): 209–215. http://dx.doi.org/10.3846/jcem.2010.23

Šliseris, J.; Rocēns, K. 2010. Curvature analysis for composite with orthogonal, asymmetrical multi-layer structure, *Journal of Civil Engineering and Management* 16(2): 242–248. http://dx.doi.org/10.3846/jcem.2010.28

Strout, M. M.; Kreaseck, B.; Hovland, P. D. 2006. Data-flow analysis for MPI programs, in *International Conference on Parallel Processing (ICPP 2006)*, 14–18 August, 2006, Columbus, Ohio, 175–184. http://dx.doi.org/10.1109/ICPP.2006.32

Uma, S. R.; Pampanin, S.; Christopoulos, C. 2010. Development of probabilistic framework for performance-based seismic assessment of structures considering residual deformations, *Journal of Earthquake Engineering* 14(7): 1092–1111. http://dx.doi.org/10.1080/13632460903556509

Vaidogas, E. R.; Šakėnaitė, J. 2011. A brief look at data on the reliability of sprinklers used in conventional buildings, *Journal of Civil Engineering and Management* 17(1): 115–125. http://dx.doi.org/10.3846/13923730.2011.559908

Wilkinson, B.; Allen, M. 2005. *Parallel programming techniques and applications using networked workstations and parallel computers*. New York: Pearson Prentice Hall. 496 p.

Wittwer, T. 2006. *An introduction to parallel programming*. The Netherlands: VSSD Delft. 63 p.

Wu, B.; Xu, G.; Shing, P. B. 2011. Equivalent force control method for real-time testing of nonlinear structures, *Journal of Earthquake Engineering* 15(1): 143–164. http://dx.doi.org/10.1080/13632461003681171

Yazdani, A.; Abdi, M. S. 2011. Stochastic modeling of earthquake scenarios in Greater Tehran, *Journal of Earthquake Engineering* 15(2): 321–337. http://dx.doi.org/10.1080/13632469.2010.498906

Zhong, H.; Lin, G.; Li, J. 2003. Application of parallel computing to seismic damage process simulation of an arch dam, in *IOP Conference Series: Materials Science and Engineering* 10(1): 1–9. http://dx.doi.org/10.1088/1757-899X/10/1/012003

Sarfraz MUNIR is a PhD Student, in The Hong Kong Polytechnic University Hong Kong. He obtained Master's degree in Engineering from Saitama University, Japan. He has a BSc in Civil Engineering from University of Engineering & Technology Lahore, Pakistan. He served as a Research Assistant at CoE-CRT, KSU, KSA, Assistant Professor at University of Lahore, Lahore, Pakistan, and as a Lecturer at University of Engineering & Technology Lahore, Pakistan.

Raja Rizwan HUSSAIN is an Associate Professor in CoE-CRT, Department of Civil Engineering, College of Engineering, King Saud University, Riyadh, Saudi Arabia. He received his PhD and MSc in Civil Engineering from the University of Tokyo, Japan for which he was ranked outstanding and was awarded best research thesis award from the University of Tokyo. He received his PhD in record short period of just two years. He has authored more than 75 publications in less than 5 years of his post PhD tenure and has received several awards, prizes and distinctions throughout his research and academic career.

A. B. M. Saiful ISLAM is a Research Fellow after receiving his PhD at the Department of Civil Engineering, University of Malaya, Malaysia. He completed his BSc in Civil Engineering and MSc in Structural Engineering from Bangladesh University of Engineering and Technology (BUET), Bangladesh. He is a member of Institution of Engineers, Bangladesh and American Society of Civil Engineers (ASCE). His research interests include offshore structures, nonlinear dynamics, finite element modelling, seismic protection, base isolation, pounding and special tall buildings.