Journal of Science and Technology 12(June 2014): 35-40

EFFECTIVE AND UNINTERRUPTED EXECUTION OF NUMERICAL INTEGRATION THROUGH COMPUTATIONAL GRID

A.K. Mandal[1*] and M.D. Hossain[2]

[1]Department of Computer Engineering, [2]Department of Computer Science and Information Technology Hajee Mohammad Danesh Science and Technology University, Dinajpur 5200, Bangladesh

Received 23 March 2014, revised 26 May 2014, accepted 16 June 2014

ABSTRACT

Numerical integration is essential for virtually all fields of scientific research. Traditionally, numerical integration is carried out sequentially in a single computer where trade-off has to be made between computing time and precision of numerical value. For reduction of computing time and betterment of precision, numerical integration problems may require a significant amount of computational effort; in this situation, using distributed computing to share task might be a solution. Therefore, this project set up a computational grid based on the Alchemi middleware and use multiple Simpson's 1/3 rules with a sample numerical integral. For uninterrupted execution, fault-tolerance method is also implemented. Results show that this approach reduces computing of numerical integration a significant amount with ensuring better precession.

Key words: Alchemi middleware, fault-tolerance, grid computing, numerical integration, Simpson's 1/3 rules

INTRODUCTION

For many engineering problems require the evaluation of the integral $I = \int_a^b f(x)$ frequently, where $f(x)$ is called the integrand, *a*= lower limit of integration, *b*= upper limit of integration. However, the function *f(x)* may be a complicated continuous function that is difficult or impossible to integrate in the closed form. In some cases, *f(x)* may not have a known analytical form; it may be known only in a tabular form (Neumaier 2001). The limits of integration may be infinite or the function *f(x)* may be discontinuous or the function may become infinite at certain points in the interval *a* to *b*. In all these cases, the integral can be evaluated only numerically. To solve the problems numerically, there are many algorithms. In this project, the Multiple-Application Simpson's 1/3 rule (Chapra and Canale 2001) is used as this equation is effective in many cases, though it requires a lot of calculation. Besides, when scientific research demands exact result in many significant digit, number of segments of the Multiple-Application Simpson's 1/3 rule have to be increased in many times. This indicates much more calculation, and employment of one processing unit takes huge time to complete the task. This computation time can be reduced significantly if distributed computing resources are used.

Grid comuting is a model of distributed computing to create the illusion of a simple yet large and powerful self managing virutal computer out of a large collection of connected heterogenous systems sharing various combinations of resources (Broberg *et.al* 2008). Another key technology in the development of grid networks is the set of middleware applications that programming environment required to construct desktop grids and develop grid applications. Among them, Alchemi (Luther *et al.* 2005) is a .NET-based framework that provides the runtime machinery for building a grid. In our application we use this framework in windows platform. A particular numerical integration is distributed and executed parallely by the help of manager and executors of this framework respectively. Since any problems of manager or executors might hamper the performance of numerical integration, we suggest folt tolerance approach so as to uninterruted execution of numerical integration can be ensured.

MATERIALS AND METHODS

Alchemi: Alchemi is a .NET-based framework that provides the runtime machinery and programming

**Corresponding author: Ashis Kumar Mandal , Department of CEN, Hajee Mohammad Danesh Science and Technology University, Dinajpur 5200, Bangladesh , Cell Phone: 88-01912136021, E-mail:ashis@hstu.ac.bd*

environment required to construct desktop grids and develop grid applications (Broberg et.al 2008). There are four types of distributed components (nodes) involved in the construction of Alchemi grids including manager, executor, owner, and cross platform manager. Owner node generates application and creates threads which are submitted in manager node. Manager node provides services such as storing thread in a pool and distributes threads to the executor. Executor accepts threads from the manager and executes them and sends result back to the manager node. After finishing all threads, the manager aggregates them and sends ultimate result to the owner node.

We choose Alchemi as the framework for building the computational grid because it is open source that can be viewed as Internet-based clustering of windows-based desktop computer. Besides, users can develop, execute and monitor grid applications using the .NET API and tools which are part of the Alchemi SDK. Alchemi offers a powerful grid thread programming model which makes it very easy to develop grid applications.

Simpson's 1/3 rule numerical integration:

Simpson's 1/3 rule (Chapra and Canale 2001) is used to estimate the value of a definite integral. Here, the interval [a, b] is broken into 2 segments, the segment width $h = \frac{b-a}{2}$. The formula for the Simpson's 1/3 rule is displayed in following equation 1.

$$I \cong \frac{h}{3}\left[f(x_0) + 4f(x_1) + f(x_2)\right]$$

……. (1)

Simpson's rule is improved by dividing the integration interval into a number of segments of equal width which is called multiple-Application Simpson's 1/3 rule. So, overall result of integration is as follows (Chapra and Canale 2001):

$h = \frac{b-a}{n}$, where n=segments number and

$$I = \frac{h}{3}\left(f(x_0) + 4\sum_{i=1,3,5\ldots}^{n-1} f(x_i) + 2\sum_{j=2,4,6\ldots}^{n-2} f(x_j) + f(x_n) \right)$$

…….. (2)

This numerical integration equation is certainly a subject that can be executed in distributed computing. That is, this equation lends itself to parallel computation because it can be easily divided into smaller problems, such that these sub-problems are independent of each other's results.

Related works: Many integration packages have been implemented based on different algorithms and computer architectures. Some are sequential (Yuasa et al. 2006) programs for standalone computers; the others are parallel programs that are executed by multiple processing elements such as those of a cluster (Cools and Laurie 1997). One work, such as distributed numerical integration, is proposed in (Doncker and Gupta 1994) where a software module named ParInt is developed for multivariate numerical integration. Similarly, Kaugars et al. (2003) describes a distributed numerical integration paradigm using web service where they propose an XML data format for the exchange of integration related data among different computing unit.

Application modeling: Before implementing our numerical integration application in grid, a grid system have to be constructed using Alchemi.Net grid framework. Now procedure to model and implimentation are described as follows:

- Specify the integration for which solution have to be calculated
- Now divide the total integral into sub-integrals of equal range
- Implement Multiple-Application Simson's 1/3 Rules for all sub-integrals
- Execute each sub-integral in particular executor of Alchemi.Net grid system
- Finally, total integral is calculated by taking all the results of sub-integrals in manager of Alchemi.Net grid

This test application basically use the concept of multithreading where the large size job is divided into some small sub-jobs. Here these sub-jobs are known as sub-integrals. Manager basically creates a thread for each sub-integral and migrates these threads into various resources (executors). Each executor accepts threads as input and computes its assigned part properly and submits the result toward the manager. Manager receives all results as input and provides appropriate result. Manager has a scheduler to control overall threads. The following figure 1 is simple flow diagram for a manager and two executors.

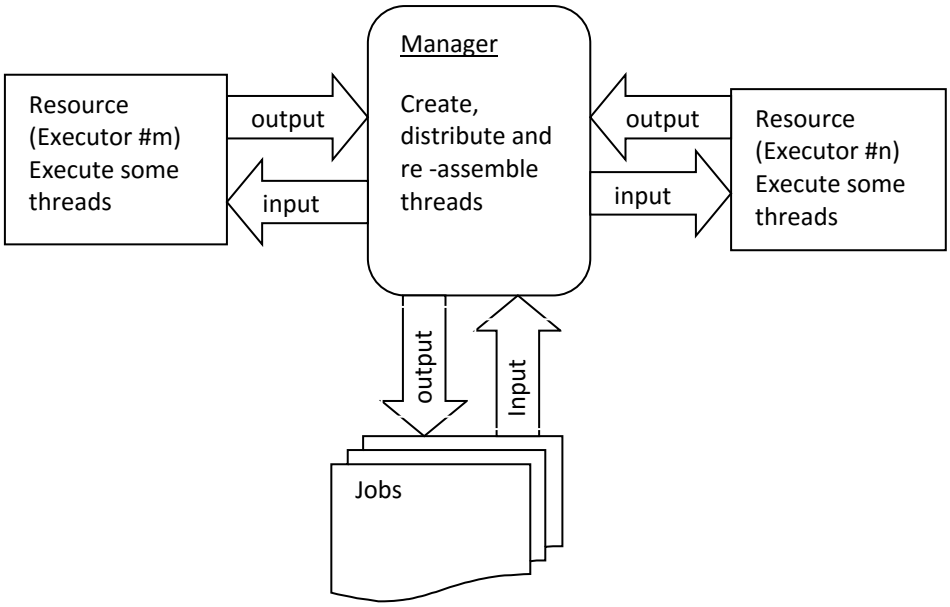

Figure 1. A flow diagram of application execution.

During the distributed execution of numerical integration, some failures may happen in the grid environment such as failure of the executor node, failure of the grid manager nodes or both. For uninterrupted execution, following methods also be used.

Failure of the executor and recovery method: If a thread is scheduled on an executor and due to some reasons, the executor crashes, the thread running on this executor also crashes. In such case, the manager reschedules this thread on another executor but it does not keep any track of the percentage of the work that has already been done. For betterment of computing, a file based grid thread was implemented to overcome this executor failure problem (Sharma et al. 2011). Here the file keeps track of the last resultant values of the thread and the executor is responsible for saving these values into the file that is on manager. Whenever a crashed thread is rescheduled on different executor, the manager node will extract the values form the file and will pass it to the thread so that it can resume its operations.

Failure of the manager and recovery method: Failure of the manager may happen if the manager gets disconnected from the network or if the manager gets crashed for any reason. Under this circumstance, all the executors registered with the failed manger will stop executing, and the whole system will come to halt. For uninterrupted execution, we need backup manager. This can be achieved by secondary manager at one of the executor (Bikas et al. 2008). Usually, an executor can register itself only with one manager. If the manager fails, secondary manager notify all executors to reregister with the new manager. This new Alchemi manager can continue functioning from the point of failure providing previous manager stores all required information in a file and this file needs to be replicated to that secondary manger. Periodic updating of file is required so as to maintain the consistency of the system.

RESULTS AND DISCUSSION

Evaluation test setup: The test bed of this Alchemi grid computing for numeric integration consisted of five Executors of Pentium (R) 4 CPU 2.26GHz desktop computers with 512MB physical memory running Windows XP Professional operating system. As for the designated Alchemi manager is set on desktop computer of Pentium (R) 4 CPU 2.26GHz machines with 512MB physical memory also running Windows XP Professional. All the six computers are linked together with a switch to form Local Area Network as grid environment. With this evaluation set up, one of the main assumptions is that every connected computer (executors) will have equal chance of receiving a thread task from manager.

We used the following integral to evaluate the procedures described above

$\int_0^1 \sqrt{x^2+1}dx$. We compute it with $\varepsilon<0.00001$

There are two parts in evaluating the performance of numerical integration in grid environment. System using above numerical integration application and execution time was measured at the elapsed clock time for the integral processing to complete on the

manager node. This application of the Alchemi grid thread model system is based on different number of executors performing the task with different number of segments and different number of split threads set.

Increasing executors with different segment size: First part of the test is to calculate the integration with increasing number of executors at different segments maintaining at 10 threads to compare and observed the affect execution time. Figure 2 shows the time taken to perform 10 threads with different amount of segments size against different number of executors.

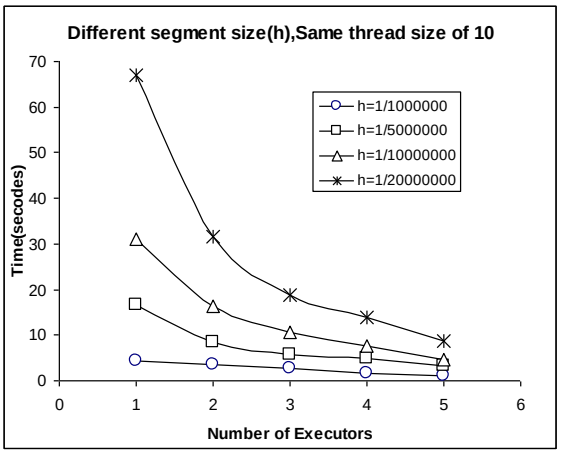

Figure 2. Performances of varying numbers of executors enabled with different segment size with same thread size.

Evaluation on different segment size and its observation result: Based on the evaluation outcome from Figure 2 on different size of segments performance on the Alchemi grid system, it is observed in showing significance improvement in execution time with increasing number of executors enabled for the task. The amount of processing time has improved by almost half when using 2 executors as compare to 1 executor. On the whole evaluation on segment number with 5 executors can shorten execution time by up to above 80% as compare to single computer processing as shown in Table 1.

Increasing executors at different thread size: The second part of the evaluation test involves in increasing up to 5 executors performing with different workloads being sliced into thread size of 10, 25, 45, 80 and 100. In order to observe the affect on execution time, same segment number of 2000000 is maintained. Figure 3 shows the time taken to complete the various numbers of threads size against varying numbers of executors enabled.

Evaluation on different thread size and its observation result: Figure 3 shows the processing time of different workloads of thread size over the grid system with increase number of executors. The resulting trend for this is similar to Figure 2 but Figure 3 shows obvious improvement in the

Table 1. Comparison for number of segments executions time efficiency improvement

No. of segments (n)	Stand alone computer execution time (s)	2 Executors grid system		5 Executors grid system	
		Execution time (s)	Percentage improve (%)	Execution time (s)	Percentage improve (%)
h=1/1000000	4.421	3.563	19.40	1.02	77
h=1/5000000	16.703	8.546	49	3.25	81
h=1/10000000	30.923	16.395	47	4.73	85
h=1/20000000	67	31.609	53	8.751	87

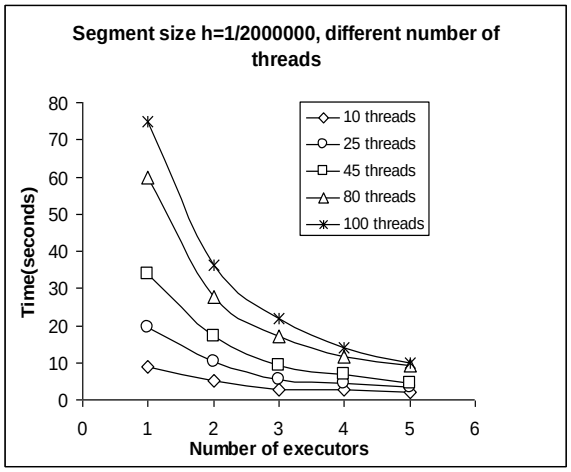

Figure 3. Performances of varying numbers of executors enabled with same number of segment and different thread size.

rendering processing time over the grid network. At low workload of 10 threads for this case, there is little difference between the execution time with different number of executors. This is so as the total overhead involved in managing the distribution of the thread task is proportion to the actual computation time. Therefore, if the number of executors continues to increase, there will have no further significant decrease in the computational time on the workload.

On the other hand with increasing thread size, significant different in the execution time can be observed with more executors enable to handle the grid job. Of course, looking at the first two-Alchemi executors, there will be obvious decrease in computational time of up to 50% with the increase in workloads thread size. Table 2 shows the improvement in the execution time for 2 executors is 50% and with 5 executors efficiency can improve up to above 80% as compare to one executor over the grid system.

Table 2. Comparison for threads size executions time efficiency improvement

Thread size	Standalone computer execution time(s)	2 Executors grid system		5 Executors grid system	
		Execution time(s)	Percentage improve (%)	Execution time(s)	Percentage improve (%)
10	8.781	5.01	42	1.89	78.4
25	19.52	10.19	48	3.4	83
45	33.94	17.23	49	4.56	87
80	60	27.55	54	9.25	85
100	75	36.4	51	10.02	87

Some general observations: While conducting the test some general observations are observed and as followed:

(a) Different processor speed desktop computer attached to the LAN network will affect the efficiency of numerical integration. Even the LAN network might cause some delay in transferring task between computers.

(b) The Alchemi scheduler is able to randomly slice and send threads task to all connected grid computers, which mean it does not required to calculate the numerical integration in any order sequence.

(c) The efficiency of the grid system depends greatly on the complexity of the task and the threads size allocated for the integral process. Of course, more slices of the workload mean more threads required to be transmitted over the network with more overheads attached to each thread to and flow the grid system; hence, this method increase the processing time.

CONCLUSION

Numerical integration is used extensively in engineering, sciences, economics, technology and other fields. As intensive computing is required for executing this integration while ensuring better precession, we present one-dimensional numerical integration scheme in computational grid to reduce executing times. In this paper we develop this grid environment by Alchemi framework. At the same time, we consider fault-tolerance methods for uninterrupted execution of numerical integration. Further enhancement could be achieved by establishment of cross platform global grid together with multidimensional numerical integration.

REFERENCES

Bikas A, Hussain A, Shoeb M, Hasan M and Rabbi MF. 2008. File based GRID thread implementation in the. NET-based Alchemi framework. 12th International Multitopic Conference, Pakistan . pp. 468-472.

Broberg J, Venugopal S and Buyya R. 2008. Market-oriented grids and utility computing: The state-of-the-art and future directions. Journal of Grid Computing. 6:255-276.

Chapra SC and Canale R. 2001. Numerical Methods for Engineers: With Software and Programming Applications. McGraw-Hill Higher Education.

Cools R, Pluym L and Laurie D. 1997. Algorithm 764: Cubpack++: a C++ package for automatic two-dimensional cubature. ACM Trans. Math. Softw. 23:1-15.

Doncker E and Gupta A. 1994. Distributed adaptive integration, algorithms and analysis. Proceedings of Transputers. pp. 266-277.

DICE. Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment. 559:306-309.

Kaugars K and Doncker E. 2003. Massive scale distributed numerical integration using web service. The Hawaii International Conference on Computer Sciences.

Luther A, Buyya R, Ranjan R and Venugopal S. 2005. Alchemi: A. NET-based Enterprise Grid Computing System. International Conference on Internet Computing. pp. 269-278.

Neumaier A. 2001. Introduction to Numerical Analysis. Cambridge University Press.

Sharma V, Vardhan M, Mishra S and Singh KD. 2011. A generalized approach for fault tolerance and load based scheduling of threads in Alchemi. Net. The Second International Conference on Cloud Computing, GRIDs, and Virtualization. pp. 211-216.

Yuasa F, Tobimatsu K and Kawabata S. 2006. Recent developments in parallelization of the multidimensional integration package