# Parallel Strategy of Implementing Composite Newton-Cotes Rules Using Message Passing on Parallel Computing Systems

**Festus Omonigho Iyuke[a], Abdul Rahman Abdullah[b] and Bahari Idrus[c]**

[a]*Faculty of Information Science and Technology*
*Universiti Kebangsaan Malaysia, 43600, Bang, Selangor Darul Ehsan*
*Tel : 012-3893765, Fax : 03-89256732 , E-mail : fesiyuke@yahoo.com*

[b]*Faculty of Information Science and Technology*
*Universiti Kebangsaan Malaysia, 43600, Bang, Selangor Darul Ehsan*
*Tel : 03-89489900, Fax : 03-89458128 , E-mail : ara@mmsc.com.my*

[c]*Faculty of Information Science and Technology*
*Universiti Kebangsaan Malaysia, 43600, Bang, Selangor Darul Ehsan*
*Tel : 03-89216180, Fax : 03-89256732 , E-mail : bahari@ftsm.ukm.my*

**ABSTRACT**

*The paper describes the parallel implementation of composite Newton-Cotes rules (Trapezoidal and Simpson's ⅓ rules) under PVM-based environment for approximating one-dimensional definite integral on parallel and distributed computing systems. The parallelism is realized by master-slave relationship where the master process decomposes the interval of integration into n subintervals, then distribute to the slave processes. Thereby initiating work pool technique to ensure perfect workload balanced state to avoid unnecessary communication overheads among the various contending processors. The effectiveness of the approach used in connection with the novel workload management scheme is demonstrated in the good quality results and the global load optimization for the tested applied application problem.*

**Keywords**

*Newton-Cotes rules, Distributed, parallel computing system, message passing, load balancing, and work pool.*

## 1.0 INTRODUCTION

The composite Newton-Cotes integration formulas are set of integration rules corresponding to the varying degree of interpolating polynomials. Newton-Cotes rules have been studied and developed to increase the accuracy of integral approximations in parallel computing environments. These rules are some of the numerical integration techniques for approximating definite integral in one-dimension as in Equation (1), and/or in multiple dimensions to give absolute accuracy.

$$I = \int_a^b f(x)dx \qquad (1)$$

We restrict our discussion to one-dimensional case. Its approximation of integral function I, takes the form of a weighted sum of integrand evaluation as specified in Equation (2),

$$I = \int_a^b f(x)dx \approx \sum_{i=0}^{n} w_i f(x_i) \qquad (2)$$

where $w_i$ $i=0,1,\ldots,n$ are called the weighting coefficients, and $x_i$, $i=0,1,\ldots,n$ nodes of the composite rules. The evaluation of the integral function is done by subdividing the integration interval [a,b] into *n* subintervals $a=x_0<x_1<x_2<\ldots<x_n=b$. Then, the specified composite rule is applied separately on each of these subintervals. The accuracy of these rules depends largely on the range of *n* equal subintervals used. For an accurate approximation of a specified integral function will required an infinite number of subintervals using these composite rules (Cheney and Kincaid, 2002; Steven and Raymond, 2002).

Parallelism is realized by approximating the integral functions by *p* processors through domain decomposition. Decomposition of the integral interval into *n* subintervals involves master-slave relationship. This process involves creating a pool of tasks with the *n* subintervals in the master process.

From this pool subintervals are distributed to the various slave processors until completion. When a slave processor completes its subinterval computational process, the partially approximated results are returned to the master processors, while simultaneously requesting for further subintervals from the master processor. When all subintervals have been taken, the master processor sent a completion signal to all slave processors indicating end of the integral evaluation. So, it summed up all partially approximated results to give the final computed integral values. A vivid description of the composite Newton-cotes rules is given in Section 2 of this paper. While the work pool technique involving the master-slave relationship is described in Section 3. Then, Section 4 discusses experimental results of the parallel composite Newton-Cotes rules on parallel and distributed network-based of PVM Linux workstations.

## 2.0 DESCRIPTION OF THE COMPOSITE NEWTON-COTES METHODS

Newton-Cotes formulas are most commonly used numerical integrations methods, but unstable over large integration intervals due to oscillatory nature of high degree polynomials. In its evaluation, the integral function to be approximated is replaced by straight-line and parabolic segments. The area under these segments is then approximated as the estimated integral values connecting both limits of integration. The rules are derived by approximating *f(x)* with Lagrange interpolating polynomials. For example, given a set of nodes $x_i$, *i=0,1…n in [a,b],* the Lagrange interpolating could then be defined as

$$\ell_i(x) = \prod_{\substack{j=0 \\ j\neq i}}^{n} \frac{(x - x_j)}{(x_i - x_j)} \qquad (0 \le i \le n) \qquad (3)$$

Therefore, Equation (3) becomes the fundamental polynomial of interpolation for all numerical integration techniques. While the polynomial of degree at most *n* that interpolates *f* at the nodes is defined as

$$p(x) = \sum_{i=0}^{n} f(x_i)\ell_i(x) \qquad (4)$$

Therefore,

$$\int_a^b f(x)dx \approx \int_a^b p(x)dx = \sum_{i=0}^{n} f(x_i)\int_a^b \ell_i(x)dx \qquad (5)$$

We obtained a formula that can be used in evaluating any integral function *f* as shown in Equation (6)

$$\int_a^b f(x)dx \approx \sum_{i=0}^{n} w_i f(x_i) \qquad (6)$$

where

$$w_i = \int_a^b \ell_i(x)dx.$$

So, a formula of the form in Equation (6) is called *closed* Newton-Cotes formula (*Trapezoidal and Simpson's ⅓ rules*) if the nodes are equally spaced (Burden and Faires, 2000).

For improved accuracy, the integration interval [a,b] is subdivided into number of subintervals for Trapezoidal rule and/or subintervals of equal width for Simpson's ⅓ rule. Thus, applying these rules piecewise on the consecutive pair of the *n* subintervals $(x_{i-1}, x_i)$, $i = 1,2,\ldots,n$, where $a = x_0 < x_1 < x_2 < \ldots < x_n = b$. Gives the areas of the individual subintervals, added to yield the approximated integral values for the entire interval. The application of the Newton-Cotes rules on each subinterval with uniform spacing $h = (b- a)/n$ and $x_i = a + ih$, for $i = 0,1,\ldots,n$ gave Equation(7) as the composite Trapezoidal rule and Equation(8) as the composite Simpson's ⅓ rule with the associated error terms (Rojiana, 1996) respectively:

$$\int_a^b f(x)dx = \frac{h}{2}\left(f(a) + 2\sum_{i=1}^{n-1} f(x_i) + f(b)\right) - \frac{(b-a)}{12}h^2 f^{(2)}(\xi) \qquad (7)$$

where $\xi \in (b, a)$. The $f^{(2)}$ in the error term, represents the average of $f^{(2)}(\xi)$ over the *n* subintervals. So the global formula truncation error is of the order $O(h^2)$, because the *n* local error terms have been summed with $h = \frac{(b-a)}{n}$.

However, for the Composite Simpson's ⅓ rule, an even number of subintervals is usually used,

$$\int_a^b f(x)dx = \frac{h}{3}\left(f(a) + 2\sum_{i=1}^{n/2} f(x_i) + 4\sum_{i=1}^{n-1} f(x_i) + f(b)\right) - \frac{(b-a)}{180}h^4 f^{(4)}(\xi) \qquad (8)$$

for some $\xi \in (a, b)$. The $f^{(4)}$ in the error term, represents the average of $f^{(4)}(\xi)$ over the $n/2$ pairs of subintervals. Also, the global formula truncation error is of the order $O(h^4)$, because the $n/2$ local truncation error terms $O(h^4)$ means reducing *h* by a

factor of two reduces the error by a factor of *16* (Schilling and Harris, 2000).

## 3.0 DESCRIPTION OF THE WORK POOL TECHNIQUE

We discuss the implementation of the parallel Composite Newton-Cotes algorithm with work *pool technique* involving master-slave relationship. The potential parallelism within the work pool algorithm is entirely dependent on the nature of the particular composite Newton-Cotes rule and the integral functions being evaluated.

The work pool technique consists of two entities: master processor and several slave processors. In the work pool technique implementation, the master identifies a set of interval, which could be profitably decomposed to give enough new subinterval(s) to keep *p* processors busy. The algorithm is applied on each of these subintervals in the various processors using the particular Composite rules. Every subinterval is associated with a quantity of data, and the work needed to process it, stored in a common data structure, called the *work pool or pool of task.* This centralized master uses some distributed strategies (sender-initiated, receiver-initiated and symmetrically-initiated) to transfer subintervals to the various slave processors (Finkel and Manber, 1987; Luling and Monien, 1993). To ensure a workload balanced system and equally allows the master processor to make workload placement decisions. The master processor also uses such distributing strategies as workload measurements, state information exchange, transfer initiation and workload placement mechanism in the transfer of subintervals (Chao-Yang et al., 2001; Zaki et al., 1997).

Then the parallelism is realized by the distribution these *n* subintervals on the available slave processors in the virtual configuration. The slave processors perform integration evaluation in a cyclical format: request a subinterval, process it, and returned the partially approximated result to master processor. While simultaneously requesting further task from the master processor in order to reduce the communication overhead involved. The slave processors receive subintervals on a continual basis from the work pool in the master processor until the termination is reached.

When all subintervals have been taken, the master send a completion signal to all slave processors indicating end of the computation. The master processor summed up the partially approximated results from the dedicated slave processors to give the final approximated result of the integral computation.

Basically, the communication is between the master and the various slave processors in the Virtual Machine (host of computers) as depicted in Figure 1, (Wilkinson and Allien, 1999; Foster, 1994). A perfectly workload balanced is attained, when the master processor have processed all of the subintervals and no processors remained idle during the integral function approximation.

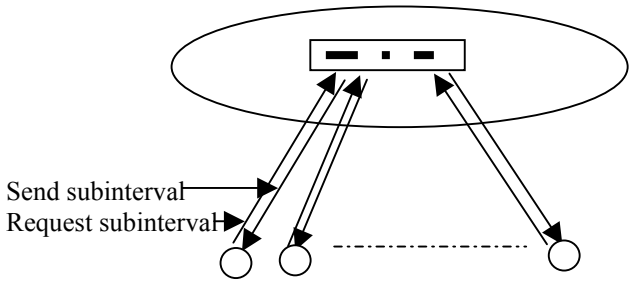


*Figure 1: Work Pool Technique*

### 3.1 Work Pool Technique Algorithm

In the algorithm implementation, there is no need to distinguish between Composite Trapezoidal rule and Composite Simpson's rule – the algorithm is applicable to either case, though, one may work better than other. However, the structure of the problem permits a data parallelism. Suppose the number of processor is given by *nproc* that computes one subinterval at a time. Therefore, the work pool holds the subintervals rather than the individual intervals (nodes) as explained in Section 3. The program code of the work pool technique algorithm is given below.

**Master**

```
count=0;
subinterval = 0;
for (i=0,i<nproc;i++)
  {
    send(&subinterval, P_i ,msg_tag);
    count++;
    subinterval++;
  }
do {
     recv (&integral, P_any result_tag);
     count--;
     if(subinterval < count)
       {
```

```
        send (subinterval, P_slave, msg_tag);
        subinterval++;
        count++;
       }else
      send(subinterval, P_slave, terminator_tag);
      integral_recv++;
      display(integration summation);
   }
  }while(subinterval >count);
```

***Slave***

```
  recv(subinterval, P_master, msg_tag);
  for(i=0,i<nproc;i++)
    {
     integral = 0.0;
     send(integral, P_master, result_tag);
    }
  recv(subinterval, m, P_master, msg_tag);
  }
```

## 4.0 RESULTS AND DISCUSSIONS

The performance of an algorithm on parallel computing systems is dependent not only on the problem characteristics and the number of processors. It does depend on how processors interact with each other, as determined both by a physical architecture in hardware and virtual architecture in software.

So the physical architecture used in the experiment described is a completely connected network-based system (topology). In a completely connected network-based system, each processor has a direct communication link to every other processor in the network. This network is ideal because a processor can send a message to other processor in a single step. So our architecture consists of 17 homogeneous Red Hat Linux 7.2 workstations, Intel Pentium IV processors, 20 GB HDD, CPU speed of 1.6 MHz, 256Mbytes of memory connected by a Ethernet (10/100 Mbps) network on PVM-based parallel programming software.

From the results obtained for the Composite Newton-Cotes techniques, both gave an impressive sublinear speedups and high efficiencies of (91% for Composite Trapezoidal rule and 96% for Simpson's ⅓ rule) with the same interval of integration. However, this competitiveness in terms of the performance evaluation or accuracy was due to subdivision of integration interval into number of segments of equal widths. One can generalized that the Composite Simpson's ⅓ rule gave a better accuracy than the composite Trapezoidal rule under the same condition of evaluation.

These performances obtained indicate the usefulness of PVM-based application in approximating integral functions problems. It also showed the degree of utilization of individual workstations (processors) in the parallel computing systems. However, these results were achieved because each processor gets equal number of subintervals to compute, showing the workload was evenly distributed on the available processors (workload balanced). If the workloads among the processors were not balanced, poor speedup and efficiency values would have be obtained. This is because some processors gets less workload (underloaded) than other processors with much workload (overloaded), as a result the underloaded processors were held at synchronizing points waiting for the other processors to get done. Thus, the load management system would not be capable of balancing the workload among processors to achieve a high quality results because it adds an almost constant overhead to all scheduling or mapping strategies.

As a result, this Centralized technique using master-slave relationship creditably exhibits sublinear speedup and parallel efficiency curves, as shown in Figures 1 and 2. It tends to decrease as the number of processors increases due to Amdahl's law analogy (Foster, 1994). Centralized algorithm using work pool technique should be preferred in the evaluation of load management schemes in parallel applications in network of workstations.

Besides, the Newton-Cotes numerical techniques have some useful industrial applications in the areas of science and Engineering. These techniques are used in the evaluation of a force exerted on a dam constructed across a river to generate hydroelectic power by a water as successfully applied in (Schilling and Harris, 2000). Also, the techniques could equally be used in the approximation of flow rate/seconds of fluid through a circular pipe with some mathematical simplifications as in (Rojiani, 1999).

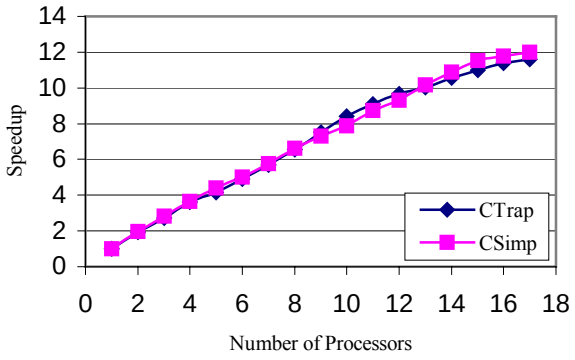


*Figure 2: Comparison of Composite Trapezoidal and Simpson's ⅓ rules: speedup vs. number of processors*

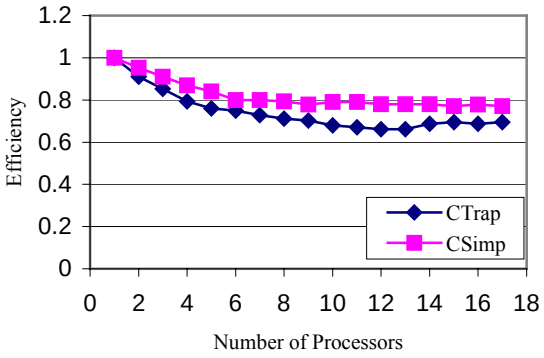


*Figure 3: Comparison of Composite Trapezoidal and Simpson's ⅓ rules: efficiency vs. number of processors*

## 5.0 CONCLUSION

We described how efficient PVM-based parallel application involving load management techniques could be used to effectively solve the integral functions problems on the network of workstations. However, the applications running this approach is essentially master-slave structured, which has been described as a valid cooperation paradigm for parallel and distributed applications. The centralized technique operates on global subintervals, which is distributed among the processing units. The overlap communication and computation by the use of synchronous and asynchronous blocking communication functions provided by PVM. Enables the load-adjusting scheme to substantially reduce out-of-work idle states in the various processors while reducing the communication needs in the integral evaluation. Also helps achieve an even workload balance, thereby obtaining a high speedup and efficiency for the composite rules considered.

It also works towards maintaining non-empty local processor and evenly balanced global workload distribution. As such the centralized technique provides a large reduction in network communication requirements, thus reducing communication bottlenecks and load imbalances that would have be apparent in the evaluation approach.

**Acknowledgements**

The research paper is supported by the Universiti Kebangsaan Malaysia under Project Grant **IRPA 04-02-02-0009-EA009.**

## 6.0 REFERENCES

Burden, R.L. and Faires, D.J. (2001). *Numerical Analysis*, California.: Brooks/Cole.

Chao-Yang, Gua and Mark, A. Stadtherr (2001). *Parallel Interval-Newton using Message Passing: Dynamic Load Balancing Strategies.*

Cheney, W. and Kincaid, D. (2002). *Numerical Analysis: Mathematics of Scientific Computing,* USA.: Brooks/Cole.

Finkel, Raphael and Udi, Manber (1987). *A Distributed Implementation of Backtracking. ACM Transactions On Programming Languages and Systems* 9**(2)**: 235-256.

Foster, I. (1994). *Designing and Building Parallel Programs: Concepts and Tools for Parallel Software Engineering,* U.S.A.: Addison Wesley.

Lüling, R. and Monien, B. (1993). *A Dynamic Distributed Load Sharing Algorithm with Provable Good Performance.* In Proceedings of the 5th Annual ACM Symposium on Parallel Algorithms and Architectures. *164-172. Velen, German.*

Rojiani, K.B. (1996). *Programming in C with Numerical Methods for Engineers,* New Jersey.: Prentice-Hall.

Schilling, R. and Harris, S. (2000). *Applied Numerical Methods for Engineers Using Matlab and C,* California.: Brooks/Cole.

Steve, C.C. and Raymond, P.C. (2002). *Numerical Methods for Engineers with Software programming Applications*, New York.: McGraw-Hill.

Wilkinson, B. and Alien, M. (1999). *Parallel Programming: Techniques and Applications using Networked Workstations and Parallel Computers*, New Jersey.: Prentice Hall.

Zaki, Mohammed Javeed, Li, Wei and Parthasarathy, Srinivasan. (1997). *Customized Dynamic Load Balancing for a Network of Workstations. Journal of Parallel and Distributed Computing* **43**(2): 156-162.