Jurnal Teknologi Maklumat & Multimedia **2**(2005): 165-178

Evaluation of Gauss Rules with a Centralised Dynamic Load Balancing Technique in Parallel Computing Systems

FESTUS OMONIGHO IYUKE & BAHARI IDRUS

ABSTRACT

Evaluation of Gauss rules with a centralised dynamic load balancing technique under PVM-based environment in approximating one-dimensional definite integrals on parallel computing systems is described. Gauss rules are normally applied in pairs, so that both an approximation to the integral and an estimate of the error in the approximation can be evaluated together. It is usual to subdivide the range of integration into n subintervals, and these rules are applied separately to each of these subintervals. The aim is to satisfy the accuracy requirement (this is assured if the sum of the estimated absolute errors across the n subintervals is less than absolute accuracy, $\in$) whilst keeping the number of evaluations of the integrand to a minimum. The load balancing operation is realised by initialising a centralised pool of task from which a workload (subintervals) to be performed is distributed to the various contending slave processors. Besides, the centralised pool of task technique involved master-slave relationship, where a master processor engaged in interval decomposition into n subintervals. Subsequently, these subintervals are distributed to the slave processors to ensure a workload balanced state is attained. Whenever a slave processor completes its subinterval computation, the partially approximated results are returned to the master. By way of reducing the communication overhead, that would have been associated with the integral evaluation process. The effectiveness of the approach used in connection with the novel workload management scheme is demonstrated in the result obtained and the global workload optimisation for the tested application problems.

ABSTRAK

Kertas ini membincangkan tentang Petua Gauss dengan teknik pengimbangan muatan dinamik terpusat bagi persekitaran berasaskan PVM dalam menganggarkan pengamiran tentu satu dimensi pada sistem pengkomputeran selari. Petua Gauss pada kebiasaannya dilaksanakan secara berpasangan supaya penganggaran kamiran serta ralat pada penganggaran tersebut

dapat dinilai bersama. Julat pengamiran dibahagikan kepada n bilangan sub-selang dengan setiap peraturan dalam petua ini akan dilaksanakan secara berasingan bagi setiap sub-selang ini. Matlamatnya adalah bagi memastikan keperluan kejituan dipenuhi (ini dapat ditentukan sekiranya jumlah anggaran ralat mutlak bagi n sub-selang kurang daripada kejituan mutlak, $\in$), dalam masa yang sama memastikan bilangan penilaian kamiran berada pada tahap minimum. Operasi pengimbangan muatan dilakukan dengan memusatkan kumpulan kerja yang mana setiap beban tugas sub-selang diagihkan kepada pemproses-pemproses kecil (slave processors). Selain daripada itu, teknik kumpulan kerja berpusat turut melibatkan hubungan di antara pemproses utama dan beberapa pemproses kecil (master-slave relationship), yang mana pemproses utama dihuraikan kepada beberapa n sub-selang. Kemudian, kesemua sub-selang ini akan diagihkan pula kepada pemproses-pemproses kecil sehingga tahap beban yang seimbang dicapai. Apabila pemproses kecil selesai melaksanakan pengiraan sub-selang, sebahagian hasil anggaran dihantar ke pemproses utama. Melalui pengurangan overhed komunikasi, ini telah diambil kira dalam proses penilaian pengamiran. Keberkesanan pendekatan yang digunakan bersama dengan skema pengurusan muatan ditunjukkan dalam dapatan kajian serta keputusan yang diperolehi melalui beban tugas yang optimum bagi masalah aplikasi yang diuji.

INTRODUCTION

Gauss rules (Legendre, Laguerre and Hermite) are numerical integration methods that use the roots of their individual polynomials. It is generally not suitable for evaluating a function given in a tabular form with equally spaced intervals. As these points are not evenly spaced, but are chosen to achieve accuracy in the integral approximations. Gauss rules are differ from the other numerical integration methods (i.e. Newton-Cotes) in that, the n points are set to the roots of the Legendre polynomial $Pn(x) = 0$, where the $Pn(x)$ is the Legendre polynomial of order n. The advantage of using Gauss' rules is to obtain a higher accuracy than the Newton-Cotes integration methods. Gauss rules have been studied and developed to increase the accuracy of integral approximations in parallel computing environments. It is among the numerical integration techniques for approximating definite integral in one-dimension as in Equation (1), and/or in multiple dimensions to give absolute accuracy, $\in$.

$$I = \int_a^b f(x)dx \tag{1}$$

The conventional approach to numerical integration is to use a quadrature rule (or a sequence of quadrature rules) to approximate I, that takes the form of a weighted sum of integrand evaluation as specified in Equation (2),

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

where w_i, xi, for all i=0,1,...,n are respectively, the weights and the abscissas of the Gauss rules. Gauss rules are normally applied in pairs, so that both an approximation to the integral and an estimate of the error in the approximation can be calculated. It is usual to subdivide the range of integration [a,b] into n subintervals, $a = x_0 < x_1 < x_2 < ... < x_n = b$, and apply the Gauss separately to each of these subintervals. The aim is to satisfy the accuracy requirement (this is assured if the sum of the estimated absolute errors across the n subintervals is less than ϵ) whilst keeping the number of evaluations of the integrand to a minimum. However, the accuracy of this rule depends largely on the range of n equal subintervals used. For an accurate approximation of a specified integral function would require an infinite number of subintervals using these Gauss rules.

Parallelism is realised by approximating the integral functions by p processors through domain decomposition. Decomposing the integral interval into n subintervals involves master-slave relationship. To achieved improve performance for the integral evaluation under this parallel application, an efficient load balancing technique is needed to distribute subinterval in a judicious manner among various processors. Thus ensures no processors are overloaded while other processors are idle or lightly loaded. We utilised a process involving the creation of pool of task 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 processor, 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, then accumulates all partially approximated results to give the final computed integral values (Iyuke et al. 2004a; Antonis et al. 2004).

Since an effective load balancing scheme requires the knowledge of the global system state (e.g. workload distribution), but in a parallel computing system, the global state is swiftly and dynamically changing and it is very difficult to accurately model the system analytically. Thus, in order to tackle the load balancing problem in such an environment where state uncertainty

is unavoidable, we employ a centralised dynamic load balancing approach to model those state variables that could cause uncertainty in global states (Kwok & Cheung 2004). The paper is organised as follows. In the next section, we describe the overview of Gauss rules. The following section contains the centralised work pool technique involving the master-slave relationship while the final section discusses the experimental results of the parallel Gauss rules on parallel and distributed network-based of PVM Linux workstations.

OVERVIEW OF GAUSSIAN RULES

Gauss rules are based on the individual roots (Legendre, Laguerre and Hermite) of polynomials that choose its point of evaluation in an optimal, rather than equally spaced interval like the Newton-Cotes rules (Nakamura 1991). However, these weighting coefficients w_i, for all i = 1,2,...,n in the approximation formula are arbitrary, and the nodes x_i for all i = 1,2,...,n are restricted by the fact that they must lie in [a,b] interval of integration. Gauss rules use the properties of orthogonal polynomials (i.e. functions whose scalar product is zero) to achieve their accuracy. It is usually evaluated over an interval -1 and +1; which is exact when applied to any polynomials of higher degree. Then, the set of Legendre polynomials, a collection {$P_0(x)$, $P_1(x)$,...,$P_n(x)$,...,} has the following properties:

1. For each n, $P_n(x)$ is a polynomial of degree n.
2. $\int_{-1}^{1} P(x)P_n(x)dx = 0,$ whenever $P(x)$ is a polynomial of degree less than n.

However, the roots of these polynomials are distinct, since they lie in the interval [-1, 1] having symmetry with respect to the origin. Therefore, the nodes x_i, for all i = 1,2,...,n is needed to produce an integral approximation formula that gives exact results for any polynomials of degree less than 2n are the roots of the nth-degree Legendre polynomial as established by the following.

Suppose x_i for all i = 1,2,...,n are the roots of the n_{th} Legendre polynomial $P_n(x)$ and for each i=1,2,...,n, the weighting coefficients w_i are defined by

$$w_i = \int_{-1}^{1} \prod_{\substack{j=1 \\ j\neq 1}}^{n} \frac{(x - x_j)}{(x_i - x_j)} \tag{3}$$

If $P(x)$ is any polynomial of degree less than $2n$, then

$$\int_{-1}^{1} P(x)dx = \sum_{i=1}^{n} w_i p(x_i) \tag{4}$$

For Equation (4) provides polynomials p of degree at most ≤ n that interpolates *f* at the nodes; that is, $p(xi) = f(xi)$ for i=0,1,2,...,n. Then, p will be a good approximation to f, and will be a good approximation to.

$$\int_{-1}^{1} f(x)dx \approx \int_{-1}^{1} p(x)dx = \sum_{i=0}^{n-1} f(x_i)\int_{-1}^{1} \ell_i(x)dx = \sum_{i=0}^{n} w_i f(x_i) \tag{5}$$

where $w_i = \int_{-1}^{1} \ell_i(x)dx$.

Therefore, *wi* needed for the Gauss rules approximation can be generated from Equation (5) to give correct values for the integral of every polynomial of degree at most *n* (Burden & Faires 2001), but these constants and the roots of the Legendre, Laguerre and Hermite polynomials are extensively tabulated in Abramowitz & Stegun (1972). Though, the first two-points (*n* = 1) of Gauss quadrature formula can be calculated using Equation (6),

$$\text{I} = \int_{-1}^{1} f(x)dx = w_0 f(x_0) + w_1 f(x_1) \tag{6}$$

where x_0 and x_1 are the unknowns two nodes, while w_0 and w_1 are the corresponding weighting coefficients. Then, applying Equation (6) successively to a constant ($f(x) = 1$), line ($f(x) = x$), quadratic ($f(x) = x^2$), and cubic function ($f(x) = x^3$) would yield the two point Gauss-Legendre formula, whose results are exact for the integral of any polynomial of degree 3 or less. However, a higher-point formula can be obtained using Equation (7),

$$\text{I} = \int_{-1}^{1} f(x)dx \approx \sum_{i=0}^{n} w_i f(x_i) \tag{7}$$

where x_i are nodes, w_i the associated weighting coefficients and *n*, the number points in the Gauss rules. The basic problem is to determine x_i and the corresponding weighting functions, w_i, so as to make the integration formula exact for $f(x)$ a polynomial of as large a degree as possible.

However, Gauss-Laguerre's rule is meant for numerical integration over semi-infinite intervals, (i.e. one limit finite and the other infinite), it is convenient to use a weight function, w_i for all i = 0,1,2,...,*n*, to ensure

convergence of the integral $wi\ f(x_i)$. An integral equation with this semi-infinite interval is evaluated using the Gauss-Laguerre's rule with a weight function given by this expression $w(x) = e^{-x}$. Accordingly, the Gauss quadrature rule associated with integration process in the range of $(0,\infty)$ is given as in Equation (8),

$$\int_0^{\infty} e^{-x} f(x)dx \approx \sum_{i=0}^{n} w_i f(x_i) \tag{8}$$

Therefore, when the semi-infinite integral has an arbitrary lower limit, a, the variable in Equation (8) could be changed to $x + a$, such that Equation (8) becomes,

$$\int_0^{\infty} e^{-x} f(x)dx \approx e^{-a} \sum_{i=0}^{n} w_i f(x_i + a).$$

For numerical integration over infinite intervals, it is convenient to use a weight function w_i, for all $i = 0,1,2,...,n$, which guarantees the convergence of the integral $w_i\ f(x_i)$. Therefore, an integral function with this infinite interval is evaluated using the Gauss-Hermite's rule with a weight function given by $w(x) = e^{-x2}$. This method is similar to Gauss-Laguerre integration process except its weighting coefficients (w_i) and abscissas (x_i) are derived from Hermite polynomials $H_n(x)$ (Ralston 1964). The Gauss quadrature formula associated with integration in this range $(-\infty, +\infty)$ is given as,

$$\int_{-\infty}^{+\infty} e^{-x^2} f(x)dx \approx \sum_{i=1}^{n} w_i f(x_i)$$

The best numerical estimates of the integral function are obtained by picking optimal abscissas, x_i at which to evaluate the function $f(x)$. The Gauss quadrature is usually optimal because it fits all polynomials up to degree $2n$ exactly, except for Gauss-Laguerre's rule, which is slightly less optimal fits. Actually, the accuracy of the Gauss integration techniques depends on the number of the abscissas (points) at which the ordinates (x_i) are evaluated, and the nature of the integrand. Since these nodes and weighting functions are mostly irrational numbers, it is used in algorithms for automatic computations to give a better accuracy with fewer functions evaluations.

Gauss rules are somewhat more difficult to apply than Newton-Cotes rules and Romberg method due to weights and nodes normally derived for

specific interval of integration [-1,1]. To apply this rule to a given interval of integration [a,b], it becomes imperative to transform these variables into standard interval for which nodes and weights have been tabulated. For example, an integral to be evaluated is given by Equation (9),

$$\int_a^b f(y)dy \tag{9}$$

Therefore, a change in variable is accomplished by making

$$x = \frac{2y-(a+b)}{b-a} \tag{10}$$

or $$y = \frac{(b-a)x+(a+b)}{2} \tag{11}$$

and $dy = \frac{(b-a)}{2}$

This permits Gaussian rule to be applied to any interval [a,b], since

$$\int_a^b f(y)dy = \int_{-1}^{1} f\left(\frac{(b-a)x+(a+b)}{2}\right)\left(\frac{(b-a)}{2}\right)dx \tag{12}$$

It could as well be written as

$$I = \frac{(b-a)}{2}\int_{-1}^{1} f\left(\frac{(b-a)x+(a+b)}{2}\right)dx \tag{13}$$

Thus,

$$I = \frac{(b-a)}{2}\sum_{i=0}^{n} w_i f\left(\frac{(b-a)x+(a+b)}{2}\right) \tag{14}$$

Then, applying Equation (14), we obtain the integral of a function between the limits $x = a$ and $x = b$ (Iyuke 2004; Kim et al. 2003; Burden & Faires 2001; Cheney & Kincaid 2002; Rojiani 1996).

Recent developments in networking have turned computer networks into attractive platforms for parallel computing bringing in a new concept of networked distributed computing (Clark & Skarmeas 1996). Distributed computing allows the network to be viewed as a multi-processor (virtual) parallel computer with the obvious benefits of its scalable cumulative power, more efficient use of existing resources, and more effective system management. These parallel computing frameworks are, as a rule, oriented towards execution of a single job on a network of workstations in parallel using Parallel Virtual Machine (PVM).

PVM is a software package that permits a heterogeneous collection of computers hooked together by a network to be used as a single large parallel computer. However, it is more oriented towards networks workstations. It provides an abstraction of a processor pool. In fact, each processor in the pool is a separate network node. This PVM system provides more high level support for the programmer. Thus, the PVM routines are used to start/stop a process or send messages between processes. The parallel-based implementation of PVM consists of a PVM daemon, responsible for managing processes on a network node, and a library, containing PVM communication and process management functions. PVM concealed mapping between processes and processors, perform implicit synchronisation or communication between parallel processes (Iyuke et al. 2005). Such frameworks consist of a programming language support and a runtime support. Programming language support usually extends an existing programming language with constructs required to write parallel programs, e.g. parallel loops, spawning of parallel processes, synchronisation and communication functions. Runtime support is responsible for parallel execution of a program on a computer network. It deals with the distribution of workloads between nodes, faults of separate machines, migration of processes, runtime communication and synchronisation.

Therefore, the implementation of these parallel Gauss algorithms with a centralised dynamic load balancing involves a work pool technique under master-slave relationship. The potential parallelism within the work pool algorithm is entirely dependent on the particular Gauss rules and integral functions being evaluated. Therefore, the centralised load balancing system addresses the issue of approximating the integral of a continuous function using the Gauss rules. The approximation of the integral value is done by subdividing the interval a to b into a fixed number of subintervals, then the area of each subinterval is approximated using the Gauss rules. However, the master processor employs coordinator process and any of the independent worker (slave) processes accessible to its own local memory. Thus, the master process implements a pool of task technique, where each worker (slave) is given a task (subinterval) to compute its area, then collects and detects termination. Each worker first receives these subintervals from the

master processor (coordinator). The worker then computes its area using the parallel Gauss rule and sends it back to the coordinator. Thus, the master processor initiates the computation and waits to receive the area from every worker and prints the results. The send and receive statements used by the coordinator and worker are message-passing primitives. A send statement packages up a message and transmits it to another process, while a receive statement waits for a message passing from process, and then stores it in a local memory for onward processing (Elbaz & Elkihel 2005; Liu et al. 2005; Cybenko 1989).

This centralised master uses some distributed strategies (sender-initiated, receiver-initiated and symmetrically-initiated) to transfer subintervals to the various slave processors (Finkel & Manber 1987; Lüling & Monien 1993). To ensure a workload balanced system and equally allows the master processor to make workload placement decisions using the above-mentioned strategies, policies like location, information, transfer and selection are also used to transfer the subintervals (Chao-Yang & Mark 2001; Zaki et al. 1997; Krueger & Shivaratri 1994; Kunz 1991). These policies, specifically the location and information policies play vital roles in the transferring of messages as discussed below.

- Location policy: The objective of this policy is to find a suitable transfer partner for a processor, once the transfer policy has decided that the processor is a heavily-loaded state or lightly-loaded one. Common location policies include random selection, dynamic selection, and state polling.
- Information policy: This policy determines when the information about the state of other processors should be collected, from where it has to be collected, and what information is to be collected. Common approaches are no exchange of states, state probing (or demand-driven) in process of load balancing, periodic exchange (information gathered periodically), state-change broadcasting, and conditional and limited multicasting (Kwok & Cheung 2004).
- Transfer policy: A transfer policy determines whether a machine is in a suitable state to participate in a task transfer, either as a sender or a receiver. For example, a heavily loaded processor could try to start process migration when its load index exceeds a certain threshold.
- Selection policy: This policy determines which task should be transferred. Once the transfer policy decides that a processor is in a heavily loaded state, the selection policy selects a task for transferring. Selection policies can be categorised into two policies; preemptive and non-preemptive. A preemptive policy selects a partially executed task. As such, a preemptive policy should also transfer the task state that can be very large or complex. Thus, transferring operation is expensive. A non-preemptive policy selects only tasks that have not begun execution. Hence, it does not require transferring the state of task.

However, the distribution of these n subintervals on the available slave processors in the virtual configuration by the master processor helps realised the load balancing operations. The slave processors perform load balancing/ integration evaluation in a cyclical format; request a subinterval, process it, and returned the partially approximated result to the 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 sends a completion signal to all slave processors indicating end of the computation. The master processor sums 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). A workload balanced is attained; when the master processor has processed all of the subintervals and no processors remained idle during the integral function approximation (Stadtherr 2004; Charcranoon et al. 2004; Wilkinson & Allen 1999; Foster 1994).

RESULTS AND DISCUSSIONS

The performance of an algorithm on parallel computing systems is not dependent 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. The physical architecture used in the experiment described is a completely connected network-based system (mesh topology). The parallel architecture used in the experimentation consists of 17 homogeneous Red Hat Linux 7.2 workstations, Intel Pentium 4 processors, 20Gb HDD, CPU speed of 1.6 MHz, 256Mb of memory, connected by Fast Ethernet (10/100 Mbps) network using PVM-based parallel programming software.

The performance of the approach evaluation is given in terms of speedup and efficiency. Therefore, speedup is defined as the ratio of the sequential execution time to the parallel execution time. While the efficiency is defined as the ratio of the parallel speedup to the number of processors used. As performance evaluation of a centralised dynamic load balancing strategy is assessed with respect to speedup. The speedup is preferred performance objective in comparing a strategies performance across several sets of parallel computers. Therefore, the results obtained for the Gauss rule technique, gave an impressive sublinear speedup when compared to the ideal case and high efficiency values of 91% for Gauss-Legendre, 89% for Gauss-Laguerre and 85% for the Gauss-Hermite rules respectively. Although, the Gauss-Legendre's

rule performance is due to the closed intervals of integration as compared to Gauss-Laguerre and Gauss-Hermite rules whose intervals of integration are infinite at one or both ends with 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 generalised that the Gauss-Legendre's rule gave an appreciable accuracy than the Gauss-Laguerre and Gauss-Hermite rules.

These performances obtained indicate the usefulness of PVM-based application in approximating integral functions problems. It also showed the degree of utilisation of individual workstations (processors) in the parallel computing systems. Besides, these results were achieved because each processor gets an equal number of subintervals to compute, showing the workload was evenly distributed on the available processors (workload balanced).

The workloads among the processors were balanced; as a result poor speedup and efficiency values were obtained. This is because some processors receive less workload (underloaded) while other processors receive too much workload (overloaded), as a result of the underloaded processors being held at synchronising points waiting for the other processors to complete evaluation. 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.

Consequently, this centralised 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, according 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.

The Gauss numerical integration techniques have some useful industrial applications in the areas of Science and Engineering. This technique may be used in the evaluation of the force exerted on a dam constructed across a river to generate hydroelectric power (Schilling & Harris 2000). Also, the technique could equally be used in the approximation of flow rate/seconds of fluid through a circular pipe with some mathematical simplifications (Rojiani 1999).

CONCLUSION

PVM-based parallel application involving load management techniques have been 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 centralised technique

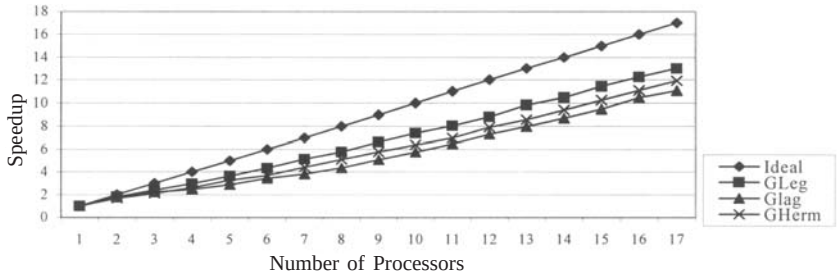

FIGURE 1. Speedup curves for Gauss rules

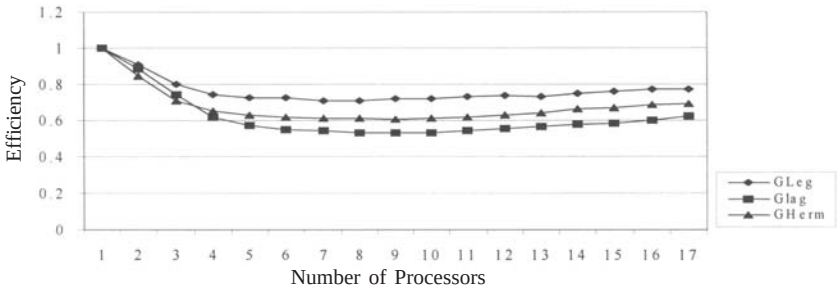

FIGURE 2. Efficiency curves for Gauss rules

operates on global subintervals distributed on all processing units. The overlap communication and computation by the use of synchronous and asynchronous blocking communication functions provided by PVM. This enables the load-adjusting scheme to substantially reduced out-of-work idle states in the various processors while reducing the communication needs in the integral evaluation. It helps achieve an even workload balance, thereby obtaining a high speedup and efficiency values. Equally work towards maintaining non-empty local processor and well-balanced global workload distribution. Thus, the centralised technique provides a large reduction in network communication requirements, reduction in communication bottlenecks and load imbalances that would have be apparent in the evaluation approach.

REFERENCES

Abramowitz, M. & Stegun, A. 1972. *Handbook of mathematical functions*. New York, USA: Dover.

Antonis, K., Garofalakis, J., Mourfos, I. & Spirakis, P. 2004. A hierarchical adaptive distributedalgorithm for load balancing. *Journal of Parallel and Distributed Computing* 64(1): 151-162.

Burden, R. L. & Faires, D. J. 2001. Numerical analysis. 7th Ed. Pacific Grove, USA: Brooks/Cole.

Chao-Yang, G. & Mark, A. S. 2001. Parallel interval-Newton using message passing: dynamic load balancing strategies. *Proceedings of the ACM/IEEE Conference on Supercomputing*, 10-16 November. Colorado, USA, 23.

Charcranoon, S., Robertazzi, T. G. & Luryi, S. 2004. Load sequencing for a parallel processingutility. *Journal of Parallel and Distributed Computing* 64(1): 29-35.

Cheney, W. & Kincaid, D. 2002. *Numerical analysis: mathematics of scientific computing*. 3rd Ed. Pacific Grove, USA: Brooks/Cole.

Clark, K. L & Skarmeas, N. 1996. Process oriented programming for agent based network management. (online). http://citeseer.ist.psu.edu/skarmeas96process.html (25 April 2005).

Cybenko, G. 1989. Dynamic load balancing for distributed memory multiprocessors. *Journal ofParallel and Distributed Computing* 7(2): 279-301.

Elbaz, D. & Elkihel, M. 2005. Load balancing methods and parallel dynamic programming algorithm using dominance technique applied to the 0-1 knapsack problem. *Journal of Parallel and Distributed Computing* 65(1): 78-84.

Finkel, R. & Manber, U. 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*. Reading, USA: Addison Wesley.

Iyuke, F. O. 2004. Implementation of dynamic load balancing for numerical integration methods on parallel computing systems. Masters Thesis. Universiti Kebangsaan Malaysia, Bangi, Malaysia.

Iyuke, F. O., Abdul, R. A. & Bahari, I. 2004. Parallel strategy of implementing composite Newton-Cotes rules using message passing on parallel computing systems. *Proceedings of the 2004 Knowledge Management International Conference & Exhibition*, 14-15 February. Penang, Malaysia, 486-491.

Iyuke, F. O., Abdul, R. A. & Bahari, I. 2005. Comparison evaluation of Romberg and Legendre's integration methods with a centralized dynamic load balancing technique in parallel computing systems. *Proceedings of the National Seminar on Computational & Experimental Mechanics*, 17-18 May. Bangi, Malaysia, 37-44.

Kim, K., Cools, R. & Ixaru, L. G. 2003. Extended quadrature rules for oscillatory integrand. *Applied Numerical Mathematics* 46(1): 59-73.

Krueger, P. & Shivaratri, N. G. 1994. Adaptive location policies for global scheduling. IEEE Transactions on Software Engineering 20(6): 432-444.

Kunz, T. 1991. The influence of different workload descriptions on a heuristic load balancing scheme. *IEEE Transactions on Software Engineering* 17(7): 725-730.

Kwok, Y. & Cheung, L. 2004. A new fuzzy-decision based load balancing system for distributedobject computing. *Journal of Parallel and Distributed Computing* 64(2): 238-253.

Liu, G. Q., Poh, K. L. & Xie, M. 2005. Iterative list scheduling for heterogeneous computing. *Journal of Parallel and Distributed Computing* 65(5): 654-665.

Lüling, R. & Monien, B. 1993. A dynamic distributed load sharing algorithm with provable good performance. *Proceedings of the 5th Annual ACM Symposium on Parallel Algorithms and Architectures*, 30 June - 2 July. Velen, Germany, 164-173.

Nakamura, S. 1991. *Applied numerical methods with software*. New Jersey, USA: Prentice Hall.

Ralston, A. 1964. *A first course in numerical analysis*. Japan: McGraw Hill.

Rojiani, K. B. 1996. *Programming in C with numerical methods for engineers*. New Jersey: Prentice-Hall.

Schilling, R. & Harris, S. 2000. *Applied numerical methods for engineers using matlab and C*. Pacific Grove, USA: Brooks/Cole.

Stadtherr, H. 2004. Scheduling interval orders with communication delays in parallel. *Journal ofParallel and Distributed Computing* 64(1): 1-15.

Wilkinson, B. & Allen, M. 1999. *Parallel programming: techniques and applications using networked workstations and parallel computers*. New Jersey, USA: Prentice Hall.

Zaki, M. J., Li, W. & Parthasarathy, S. 1997. Customized Dynamic Load Balancing for a Network of Workstations. *Journal of Parallel and Distributed Computing* 43(2): 156-162.

Festus Omonigho Iyuke
Faculty of Information Technology
Universiti Tun Abdul Razak (UNITAR)
47301 Kelana Jaya
Selangor Darul Ehsan
e-mail: festus@unitar.edu.my

Bahari Idrus
Faculty of Information Science and Technology
Universiti Kebangsaan Malaysia
43600 UKM Bangi
Selangor Darul Ehsan
e-mail: bahari@ftsm.ukm.my