CONF-901121--6

ON THE ADEQUACY OF MESSAGE-PASSING PARALLEL SUPERCOMPUTERS FOR SOLVING NEUTRON TRANSPORT PROBLEMS*

CONF-901121--6

DE90 016041

Y. Y. Azmy

Engineering Physics and Mathematics Division

Oak Ridge National Laboratory
P.O. Box 2008, Bldg. 6025
Oak Ridge, TN 37831-6363
(615) 574-8069

To be published in the Proceedings of SUPERCOMPUTING '90
New York City, New York, November 12-16, 1990.

*Research sponsored by Office of High Energy and Nuclear Physics, U.S. Department of Energy under Contract No. DE-AC05-84OR21400 with Martin Marietta Energy Systems, Inc. This research was supported in part by the Office of Laboratory Computing of Oak Ridge National Laboratory.

ON THE ADEQUACY OF MESSAGE-PASSING PARALLEL SUPERCOMPUTERS FOR SOLVING NEUTRON TRANSPORT PROBLEMS*

Y. Y. Azmy

Engineering Physics and Mathematics Division
Oak Ridge National Laboratory
P.O. Box 2008, Oak Ridge, TN 37831-6363

ABSTRACT

A coarse-grained, static-scheduling parallelization of the standard iterative scheme used for solving the discrete-ordinates approximation of the neutron transport equation is described. The parallel algorithm is based on a decomposition of the angular domain along the discrete ordinates, thus naturally producing a set of completely uncoupled systems of equations in each iteration. Implementation of the parallel code on Intel's iPSC/2 hypercube, and solutions to test problems are presented as evidence of the high speedup and efficiency of the parallel code. The performance of the parallel code on the iPSC/2 is analyzed, and a model for the CPU time as a function of the problem size (order of angular quadrature) and the number of participating processors is developed and validated against measured CPU times. The performance model is used to speculate on the potential of massively parallel computers for significantly speeding up real-life transport calculations at acceptable efficiencies. We conclude that parallel computers with a few hundred processors are capable of producing large speedups at very high efficiencies in very large three-dimensional problems.

I. INTRODUCTION

The neutron transport equation is a special case of the general Boltzmann equation in which the highly improbable collisions between neutrons are neglected, thus rendering the Boltzmann collision term linear. Solutions for neutron transport problems are sought in many practical applications such as the design and optimization of nuclear reactor cores, shield design, calculation of heating rates in various reactor components, etc. In most such situations the problem is too complicated to be solved analytically, and approximate methods are inevitable. Indeed, over the years many approximation methods, algorithms, and computer codes have been developed, implemented, and used to obtain numerical solutions to the neutron transport equation,[1] providing a wide spectrum of approaches each having its own range of physical problems for which it is most suitable.

A variety of parallel algorithms have been developed recently for solving neutron transport problems each based on a specific domain decomposition that is most suitable for a certain class of problems.[2-6] Decomposition of the neutron energy variable along the multigroup structure commonly employed in nuclear applications has been shown to be particularly suitable for problems involving upscattering (i.e. scattering from low to high energies).[2] Also chaotic iterative schemes have been reported for such problems, and found to possess several interesting features.[2] Decomposition of the spatial domain has been considered in an attempt to provide a large number of concurrent processes offering the potential of very high speedup.[5] The third alternative for realizing a parallel algorithm is to decompose the angular domain along the set of discrete ordinates employed in S_n calculations. This has been done for two-dimensional Cartesian geometry problems,[4] and for one-dimensional spherical geometry.[6]

As will be discussed later, the angular domain decomposition in Cartesian geometry occurs in a natural way, unlike most other possible decompositions mentioned above. That is to say, in Cartesian geometry the solution algorithm is comprised of operations in each discrete direction (i.e. angle) that are completely and naturally independent of all other discrete directions. Hence, the original and decomposed algorithms are identical in that they perform the same set of operations, on the same set of initial and intermediate data, and therefore produce identical intermediate and final results every step of the way.[4] This is not true in the other cases where the decomposition is artificially introduced into the solution algorithm, and often requires a larger number of iterations to achieve convergence compared to the undecomposed case.[5,6] Obviously this disadvantages the parallel algorithm because the total amount of computations performed (which is proportional to the number of iterations) becomes larger in the decomposed algorithm, so that speedups with respect to the undecomposed algorithm (i.e. sequential) that are proportional to the number of processors are practically impossible.

In the last fifteen years, nodal methods have been developed, implemented, verified against conventional methods, and heavily utilized in the solution of neutron transport problems in various technical settings.[7-10] These methods have been shown to possess very high accuracies, thus permitting the use of relatively coarse meshes, which eventually translates into high computational efficiencies.[7-10] Furthermore, it has been shown recently that general high-order versions of one particular variety, the nodal integral method, can be written in a simple weighted difference form,[10] making it easy to implement, or backfit, into existing weighted difference production codes. The demonstrated high computational efficiency of nodal methods (i.e. short CPU time for a given accuracy as compared to conventional methods) made it the method of choice in calculations for real-life applications.

*Research sponsored by Office of High Energy and Nuclear Physics, U.S. Department of Energy under Contract No. DE-AC05-84OR21400 with Martin Marietta Energy Systems, Inc. This research was supported in part by the Office of Laboratory Computing of Oak Ridge National Laboratory.

In this paper we explore the potential for achieving high performance on massively parallel supercomputers in solving multidimensional neutron transport problems. This is accomplished by establishing a model for the performance of the parallel nodal transport code, P-NT,[4] as a function of the problem size and the number of processors. In Sect. II we discuss the standard iterative scheme used in solving neutron transport problems, and we describe its implementation on Intel's iPSC/2 hypercube. Two test problems that have been solved by P-NT on the iPSC/2 are presented in Sect. III, with particular emphasis on the parallel performance of the code. The parallel performance model is then developed in Sect. IV, and is validated against the measured results described in Sect. III. We conclude with a summary of the work and the most pertinent conclusions.

II. ANGULAR DOMAIN DECOMPOSITION OF THE STANDARD ITERATIVE SCHEME

In order to illustrate the issues involved in developing a parallel algorithm for solving the nodal transport equations without being overwhelmed by algebraic details, we restrict our discussion here to the lowest (zero) order nodal method. Our parallel code P-NT has a first order (i.e. linear-linear) capability which is used later to measure and model the performance of first order methods. The reader interested in parallel high-order methods will find it straightforward to apply the concepts presented here to the general-order weighted difference equations presented in Ref. 10, since the angular domain decomposition is independent of the approximation order of the nodal method.

Like the general Boltzmann equation, the steady state neutron transport equation in two-dimensional Cartesian geometry has five independent variables: two spatial, two angular, and one energy, variables. In mostly all numerical applications these variables are discretised, and discrete values representing the angular flux (the dependent variable) are defined with respect to the discrete independent variables, and solved for algebraically. The equations for the lowest order nodal method in two-dimensional Cartesian geometry with monoenergetic neutrons are given by,

$$\frac{\mu_k}{2a_{ij}}\left[{}^{m-y}_{\psi_k}(+a_{ij}) - {}^{m-y}_{\psi_k}(-a_{ij})\right] + \frac{\eta_k}{2b_{ij}}\left[{}^{m-x}_{\psi_k}(+b_{ij}) - {}^{m-x}_{\psi_k}(-b_{ij})\right] + \sigma_{ij}^T\, {}^{m=}_{\psi_{ijk}} = \sigma_{ij}^s\, {}^{m-1=}_{\phi_{ij}} + S_{ij} \quad , \tag{1}$$

$${}^{m=}_{\psi_{ijk}} = {}^{m-y}_{\psi_k}(+a_{ij})\,[1+\alpha_{ijk}]/2 + {}^{m-y}_{\psi_k}(-a_{ij})\,[1-\alpha_{ijk}]/2 \quad , \tag{2}$$

$${}^{m=}_{\psi_{ijk}} = {}^{m-x}_{\psi_k}(+b_{ij})\,[1+\beta_{ijk}]/2 + {}^{m-x}_{\psi_k}(-b_{ij})\,[1-\beta_{ijk}]/2 \quad , \quad k=1,\ldots,N. \tag{3}$$

where (μ_k, η_k) are the x and y angle cosines for the k-th discrete direction, $k=1,\ldots,n(n+2)/8$, in an S_n order quadrature, $(2a_{ij},2b_{ij})$ are the x and y dimensions of the ij-th computational cell, σ_{ij}^T and σ_{ij}^s are the total and scattering cross sections in the ij-th cell respectively, and S_{ij} is a fixed neutron source. The spatial weights in Eqs. (2) and (3) are derived consistently via the nodal integral method, and are given by,[10]

$$\alpha_{ijk} = \coth(\sigma_{ij}^T a_{ij}/\mu_k) - \mu_k/\sigma_{ij}^T a_{ij} \quad , \tag{4}$$

$$\beta_{ijk} = \coth(\sigma_{ij}^T b_{ij}/\eta_k) - \eta_k/\sigma_{ij}^T b_{ij} \quad . \tag{5}$$

The discretised dependent variable is represented by three quantities: ${}^{m=}_{\psi_{ijk}}$ is the ij-th cell-averaged angular flux, ${}^{m-x}_{\psi_k}(\pm b_{ij})$ is the x-averaged angular flux evaluated at $y = \pm b_{ij}$, and analogously ${}^{m-y}_{\psi_k}(\pm a_{ij})$, where m is the iteration index. The angular flux is considered continuous across node boundaries, so that ${}^{m-x}_{\psi_k}(+b_{ij}) = {}^{m-x}_{\psi_k}(-b_{ij+1})$, and ${}^{m-y}_{\psi_k}(+a_{ij}) = {}^{m-y}_{\psi_k}(-a_{i+1,j})$. The relationship among the dependent discrete-variables is depicted in Fig. 1 on a typical computational cell, for $\mu_k, \eta_k > 0$; for $\mu_k < 0$, and $\eta_k < 0$ the sense of the horizontal, and vertical arrows is reversed, respectively. At a given cell, normally the incoming angular flux (in the sense of the arrow) is known from the neighboring cell, or from global boundary conditions if the given cell is interior or adjacent to an external boundary, respectively. Then Eqs. (1-3) provide three linearly independent equations that are solved for the cell-averaged flux and the two outgoing fluxes on x and y = constant surfaces, which by continuity of the angular flux, are incoming fluxes to the adjacent cells in the x, and y directions, respectively. The process is repeated in these adjacent cells until the entire mesh is covered; this completes one "mesh sweep" in the k-th angular direction, to be followed by sweeps in all other angular directions thus completing one iteration. The quantity iterated upon is the scalar flux ${}^{m=}_{\phi_{ij}}$, which is updated at the conclusion of the m-th iteration using,

$${}^{m=}_{\phi_{ij}} = \sum_{k=1}^{n(n+2)/8} \omega_k\, {}^{m=}_{\psi_{ijk}} \quad , \tag{6}$$

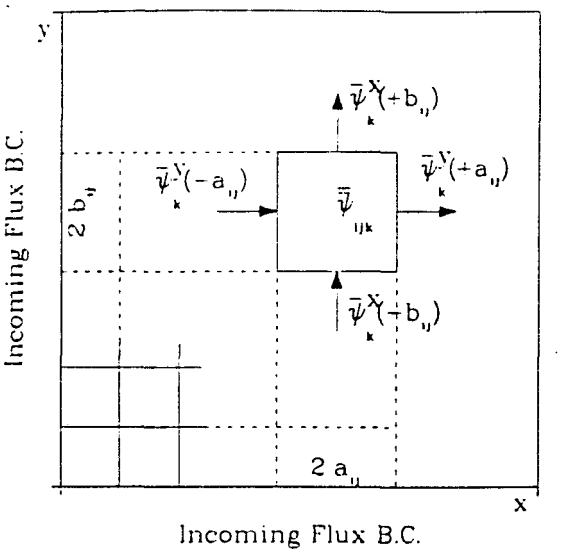

Fig. 1. Relationship among dependent variables for a typical computational cell; arrows indicate incoming and outgoing angular directions for the case $\mu_k, \eta_k > 0$.

where ω_k is an angular weight defined consistently with the k-th discrete direction. It is worthwhile mentioning that several generalizations of the method described above can be achieved by straightforward modifications; see Ref. 10.

The solution of Eqs. (1-3) plus the global boundary conditions constitutes the bulk of discrete ordinates transport calculations, as it involves a very large number of unknowns to be evaluated every iteration. For example, an S_8 calculation on a 20 x 20 mesh, involves 12,000 cell and surface averaged angular fluxes to be calculated per iteration. In the standard iterative scheme Eqs. (1-3) are solved via the successive "sweeps" described above. The lack of dependence between the angular flux in each direction k and the angular fluxes in all other directions represents a naturally occurring decomposition in the angular domain which we exploit in developing the parallel solution algorithm presented below.

The Mathematical Sciences Section of the Engineering Physics and Mathematics Division at Oak Ridge National Laboratory owns and operates an Intel's iPSC/2 hypercube computer which has 64 nodes (processors), each with 4 Mbytes of memory, and connected to the other nodes via a six-dimensional hypercube scheme. Information is shared among nodes via explicit message passing introduced into the code through extensions to FORTRAN. The nodes communicate to external i/o devices mainly through a host computer; hence a host program reads the input data and passes it to all participating nodes, and at the conclusion of the calculation, collects the solution from the nodes and prints it out. The standard iterative scheme is performed exclusively by the nodes. After receiving the input data from the host, each node program determines the set of angular directions it is to solve (static scheduling), and immediately starts sweeping the mesh in these directions. Clearly the larger the number of nodes participating in the calculation, the smaller the work load for each node, and the faster the computation. At the conclusion of the mesh sweep the global operator GDSUM is called to perform the summation represented by Eq. (6), yielding the new iterate of the cell averaged scalar flux on each of the participating processors. Next, all nodes compare the pointwise relative difference between the old and new scalar flux iterates to a user specified convergence criterion. Upon convergence, node 0 sends the host the converged solution, otherwise all nodes start a new iteration by again sweeping the mesh each in the discrete directions previously assigned to it.

III. PERFORMANCE OF P-NT ON THE iPSC/2

The parallel algorithm described above has some important features that bear heavily on its performance. The parallelization realized by the angular domain decomposition is very coarse grained; this implies that number of data exchanges between the participating nodes is very small, and this contributes positively to the high efficiency of P-NT. On the other hand, the grain coarseness sets a strict limit on the number of independent processes available for concurrent execution, thus limiting the potential for extremely large speedups on very massive parallel computers. More specifically, the number of independent processes available in a two dimensional problem with vacuum boundary conditions on all global boundaries, in a calculation employing an S_{10} quadrature set is 60; in three-dimensional problems this number is 120. In extreme cases where very high quadrature orders are necessary, e.g., S_{20}, two and three dimensional problems offer 220 and 440 independent processes, respectively, well below the full potential of massively parallel supercomputers capable of supporting thousands of processors. Hence, it seems that for present applications the algorithm presented here is suitable for high performance on low to medium size parallel supercomputers.

Another difficulty resulting from the grain coarseness of the parallel algorithm concerns load balance. If the number of participating processors does not divide the number of independent processes, some nodes will remain idle in each iteration until all directions are calculated. This is extremely penalizing when a large number of processors is used, so that the share of each processor is only a few directions, because in this case the idle time will be of the same order as the busy time, thus reducing the efficiency. Dynamic scheduling would not repair this problem either because all independent processes have almost the same length. This disadvantage is highlighted in a hypercube connection scheme where the number of processors available to the user must be a power of two, which limits the cases in which perfect load balance is achievable.

In the performance measurements presented in this section, and the performance model presented in Sect. IV, we side-step this disadvantage by assuming that arbitrary node numbers are available to the user. This is not as bad an assumption as it may appear, because it essentially extends the applicability of our conclusions and performance model to other connection schemes (e.g. grids) that do not restrict the choice of the number of participating processors, as long as the CPUs and communication speed are comparable to those of the iPSC/2. Hence in the performance measurements presented in this section, we present results obtained on cubes that do not utilize all nodes in that cube, and in such cases we calculate the efficiency based only on the number of participating nodes, not all those included in the attached cube.

The parallel code P-NT with zero and first order spatial approximations has been successfully implemented on the iPSC/2. Because the angular domain decomposition occurs naturally as discussed before, P-NT requires exactly the same number of iterations as the equivalent sequential code , GONT,[10] and converges to an identical solution (to within roundoff). This fact has been checked to be true in all cases executed to verify the correctness of the parallelization procedure. P-NT was used to solve two test problems whose geometry and material composition are presented in Fig. 2. The spatial discretization employed was a uniform 16 x 16 mesh, and the two problems solved were an S_8, zero order case, and an S_{16}, first order case. In order to evaluate the performance of the parallel code, we monitored the CPU time required for convergence as a function of the number of participating processors, and used this data to evaluate the performance of P-NT. We used two quantities to represent a quantitative measure of the parallel performance of P-NT: the speedup, S(P) = CPU time required by one processor/CPU time required by P processors to solve the same problem, and the efficiency, E(P) = 100 x S(P) / P. The speedup and efficiency vs the number of participating processors for the two test problems are shown in Figs. 3 and 4 respectively. As expected the second test problem yields higher speedup and better efficiency for the same number of participating processors. Two factors contribute to this result. First, the first order method requires more computations than the zero order method, because in the former four flux spatial moments are calculated per computational cell, while in the latter only one spatial moment is calculated. It should be clear also that in the first order case all four spatial moments have to be summed globally via GDSUM, so that the net improvement in performance implies that this additional burden is more than compensated for by the increase in computation time. Second, the higher order quadrature provides a larger pool of independent processes to be performed concurrently, thus reducing the relative effect of the communication penalty compared to the useful computation time. Since three dimensional applications would produce similar effects, namely an increase in the number of calculated flux spatial moments per cell, and a two fold increase in the number of independent processes for the same quadrature order, we conjecture that even higher speedup and efficiency should be achievable. On the other hand, it has been observed that increasing the number of computational cells produces only marginal improvements to the speedup and efficiency.[4]

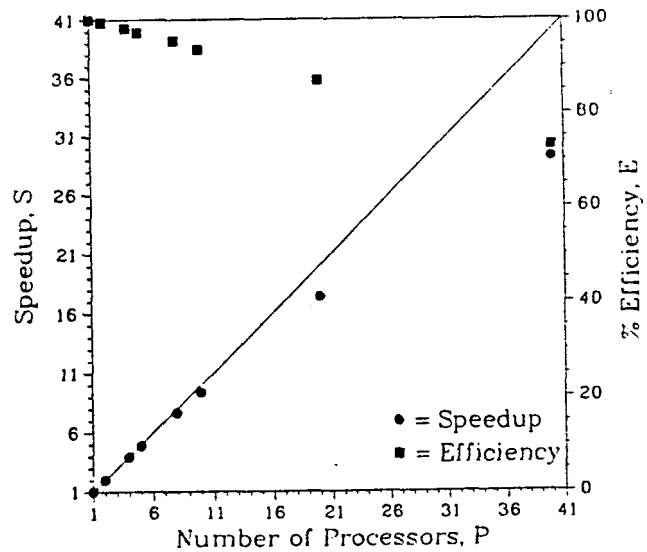

Fig. 3. Speedup and efficiency vs the number of processors for the S_8, zero order method test problem.

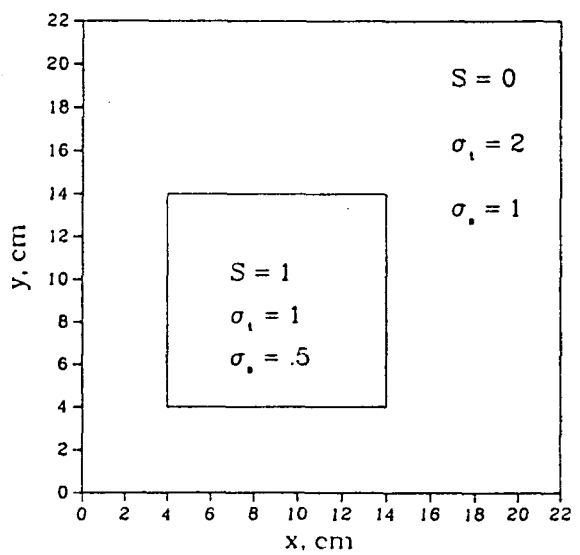

Fig. 2. Geometry and nuclear properties of the test problem with vacuum boundary conditions, i.e., zero incoming angular flux, on all four external boundaries.

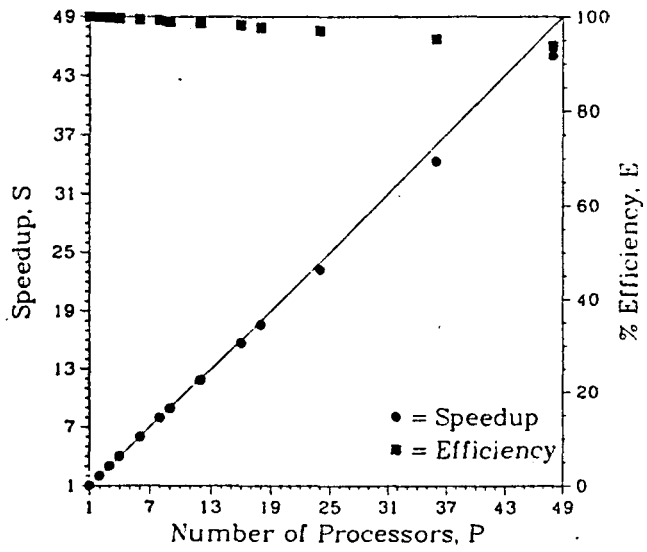

Fig. 4. Speedup and efficiency vs the number of processors for the S_{16}, first order method test problem.

IV. PERFORMANCE MODEL FOR P-NT ON THE iPSC/2

The full potential of parallel codes for high performance can not be determined based on a few test cases alone. For this purpose it is desirable to develop mathematical models that describe the performance of the parallel computer for a given algorithm as a function of relevant parameters. In general this process can be extremely complicated and it is often necessary to use statistical methods to fit simplified models to measured performance data, or to use very crude models, e.g. Amdahl's law. In contrast, the parallel algorithm described in Sect. II is for the most part "clean and simple" enough to permit the development of a fully mathematical model that describes its performance as a function of problem size, represented by n, the angular quadrature order, and the number of processors participating in the calculation, P. This fact is a direct consequence of the coarse grained and static scheduling features of the parallel algorithm, which makes predictable the exact sequence of operations performed on each processor. The only exception to this predictability is the global operation, GDSUM, which is truly statistical in nature, as discussed below.

There are three main components of the CPU time that add up to the total computation time: a serial component that is independent of the number of processors; a parallel component that is inversely proportional to the number of processors; and a "global summation" component which is dominated by communication and therefore is directly proportional to the dimension of the attached cube. Determining the full dependence of each of these components on the spatial approximation order, the mesh size, and the angular quadrature order is difficult. The dependence on the approximation order is easily accounted for parametrically, i.e. we develop separate models for the zero order and first order methods. The dependence on the mesh size is currently being developed and is complicated by the dependence of the number of iterations on the number of computational cells. As noted previously,[4] only minor gains in performance are achievable by refining the mesh; so for the time being we develop the performance model for a constant 16 x 16 mesh. In contrast the number of iterations required for convergence depends very weakly on the angular quadrature order. Hence it is reasonable to approximate the parallel component dependence on n as directly proportional to n(n+2)/2, because for a given mesh, one mesh sweep takes the same amount of time regardless of the quadrature order. Also, the results presented in Sect. III indicate that the serial and global summation time components are much smaller than the parallel component, and should grow much slower with the quadrature order.

Let T_{so} be the serial time component for the order o method, o = 0, 1; T_{po} the time required to sweep the mesh once for order o method; and T_{go} the time required to perform the global summation for order o method. According to the simplifying argument presented in the previous paragraph T_{so}, T_{po}, and T_{go} are independent of n and P on a given mesh. The total CPU time immediately follows,

$$T_o(n,P) = T_{so} + T_{po}\left\lceil\frac{n(n+2)}{2P}\right\rceil + T_{go}\left\lceil\frac{\log P}{2}\right\rceil \quad , o=0,\ 1, \tag{7}$$

where [.] is the ceiling function, and the last term on the RHS is equal to the attached cube dimension. The parallel component of the model is capable of modeling poorly balanced situations, even though we have intentionally avoided such cases in the test problems described in Sect. III.

In order to validate the model, Eq. (7), we installed clocks at several locations in P-NT to monitor the various time components for the zero and first order methods as a function of the number of processors. The serial and parallel time components behaved very consistently with the model, i.e., T_{so} and T_{po} were practically constant for all values of P considered. Because of the global nature of the summation operation, it is not fully deterministic, so that repeating the same run can give slightly different measurements of the consumed CPU time. However, the measured data seemed to closely follow a linear dependence on the cube dimension as logically expected in a hypercube connection scheme. Therefore, we calculated T_{go} as the ratio of the measured global summation time to the attached cube dimension averaged over the various choices of P. The agreement between measured and model data for this component is reasonable. The measured and model time components for the S_8, zero order, and S_{16}, first order test problems are compared in Figs. 5 and 6, respectively; the observed good agreement establishes the validity of the model.

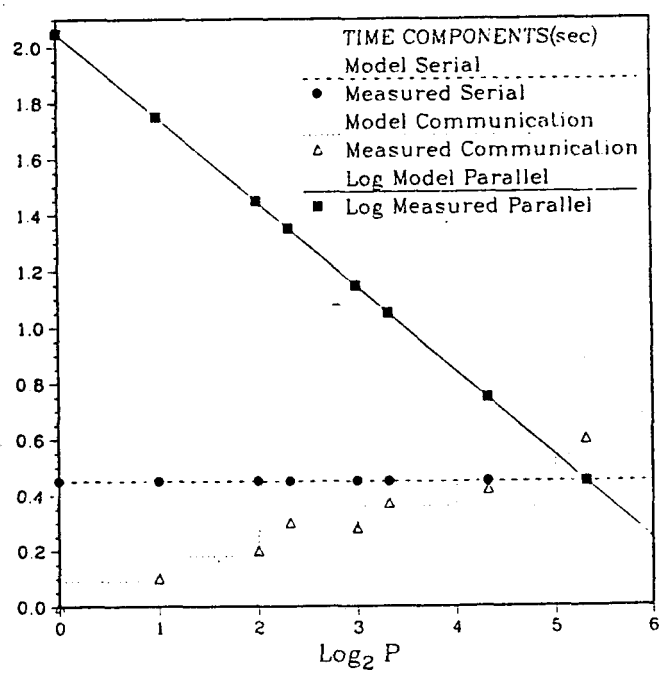

Fig. 5. Comparison between experimental and measured time components for the S_8, zero order method test problem.

To explore the full potential for high efficiency parallel performance by the parallel algorithm presented here on a two-dimensional 16 x 16 Cartesian mesh for the zero and first order methods we use the model, Eq. (7), to evaluate the efficiency as a function of n and P. Figures 7 and 8 depict, with pattern codes, the regions in the (n,P) plane at which high efficiencies are predicted by the model for the zero and first order methods, respectively. As expected, regions of high efficiency for

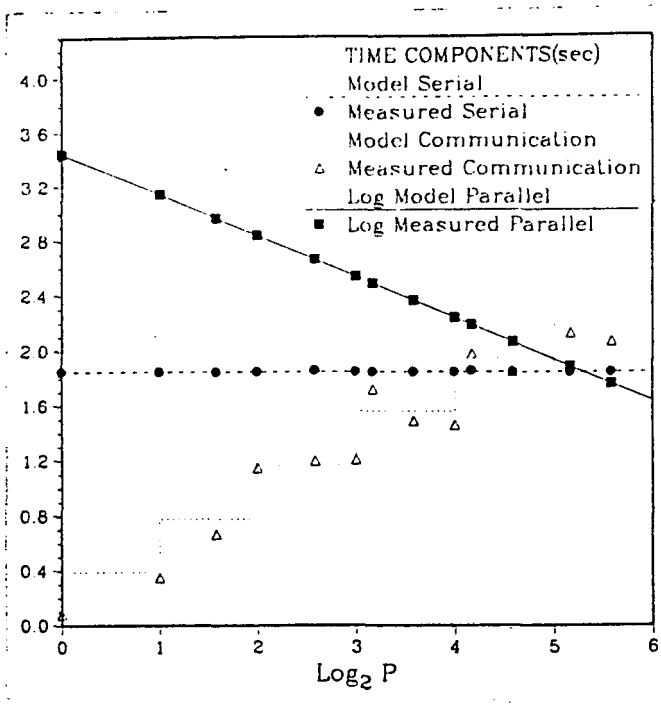

Fig. 6. Comparison between experimental and measured time components for the S_{16}, first order method test problem.

the first order method cover a larger region in the (n,P) plane than does the zero order method because of the heavier computational load in the former method. "Islands" of high efficiency appear at locations in the (n,P) plane where the computational load is well balanced at a few isolated points, and is poorly balanced at surrounding points. One important observation to be made from Figs. 7 and 8 is that load balance is extremely crucial for achieving very high efficiencies in general, but that for very large problems reasonably good efficiencies are still achievable on several tens of processors. The utility of performance models, such as Eq. (7), is evident from Figs. 7 and 8; a user solving a given S_n problem and interested in high efficiency performance, i.e. lower computation cost, would normally select P in a region surrounded by the darkest pattern on a constant n line. In contrast, a user interested in the highest speedup, i.e. shortest computation time, would select the largest possible P, especially if it exists on an island of high efficiency.

V. SUMMARY AND CONCLUSIONS

We presented an angular domain decomposition of the standard iterative scheme commonly used in solving neutron transport problems that yields a coarse grained parallel algorithm. We described our application of the parallel algorithm to the highly accurate nodal method, and its implementation in the parallel nodal transport code, P-NT. We demonstrated the very high speedup and efficiency that the parallel code is capable of achieving on Intel's iPSC/2 hypercube using two test problems. Also we developed and validated a model for the performance of the parallel code on the iPSC/2, and we used the model to explore

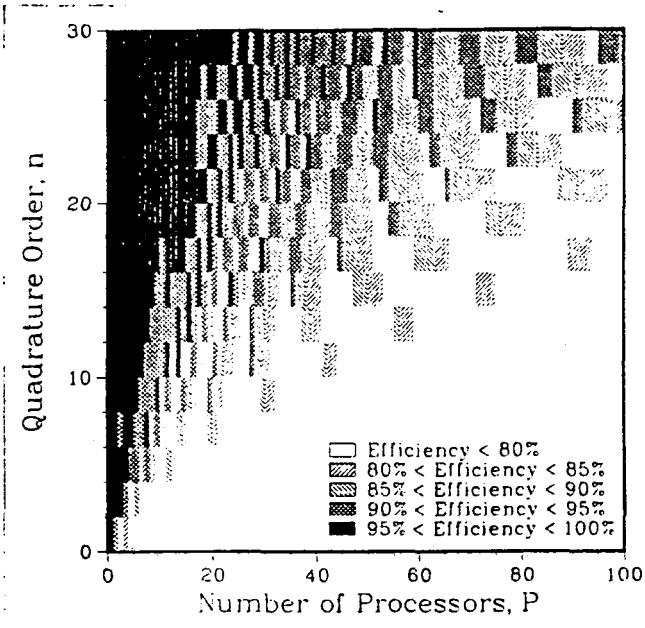

Fig. 7. Pattern-coded map of the efficiency for the zero-order method in the (n, P) plane.

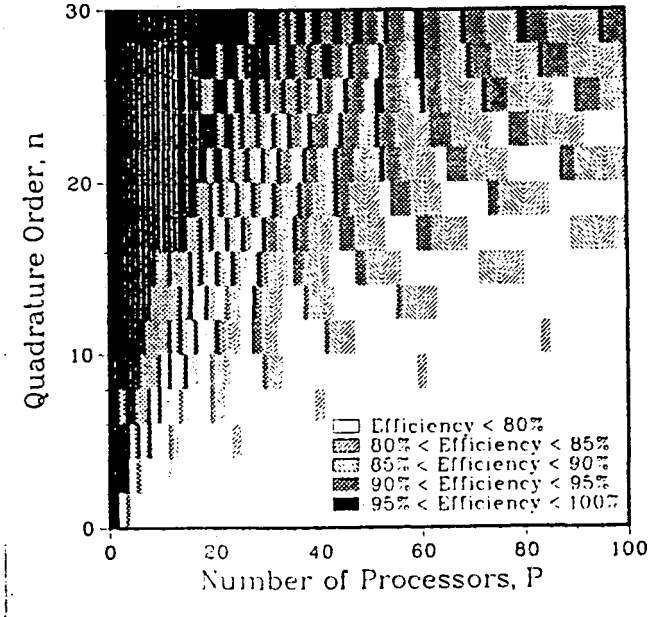

Fig. 8. Pattern-coded map of the efficiency for the first-order method in the (n, P) plane.

the potential for high performance in even larger problems than used here.

Our test problems and performance model suggest that the limited number of processes that can be executed simultaneously in this algorithm (dictated by its coarse grain) limits the size of parallel computers that can be used to a few hundred processors. Medium grained algorithms based on alternative domain decompositions, combined with well established acceleration schemes may provide a larger number of concurrent processes at a relatively low iteration penalty. This should utilize to a fuller extent massively parallel computers, with thousands of CPUs. Also, alternative architectures, such as shared memory machines, seem to be very well suited to take full advantage of the present parallel scheme because of the smaller number of processors they normally support. Finally, futuristic applications may arise in science and technology which require using a very high order angular quadrature in three dimensional geometry, in which case the present algorithm will perform extremely well on massively parallel computers.

REFERENCES

1. E. E. Lewis and W. F. Miller, "Computational Methods of Neutron Transport," John Wiley and Sons, New York (1984).

2. B. R. Wienke and R. E. Hiromoto, "Parallel Sn Iteration Schemes," Nuclear Science & Engineering, 90, 116 (1985).

3. W. A. Rhoades and R. L. Childs, "Sn Transport Calculations on Vector and Parallel Processors," Transactions of the American Nuclear Society, 55, 320 (1987).

4. Y. Y. Azmy, "Multidimensional Nodal Transport Methods for Multiple-Instruction Multiple-Data, Distributed Memory Machines," Transactions of the American Nuclear Society, 56, 292 (1988).

5. M. Yavuz and E. W. Larsen, "Spatial Domain Decomposition Methods for Discrete-Ordinates Problems," Proc. ANS Topical Meeting on Advances in Nuclear Engineering Computation and Radiation Shielding, Santa Fe, New Mexico, April 9-13, 1989, No. 12, (1989).

6. Alireza Haghighat and Ronald Mattis, "Parallel/Vector Algorithms for the Spherical Sn Transport Theory Method," to appear in the Proc. Int. Conf. on Supercomputing in Nuclear Applications, Mito City, Ibaraki, Japan, March 12-19, 1990.

7. J. J. Dorning, "Nodal Transport Methods After Five Years," Proc. Topical Mtg. Advances in Nuclear Engineering Computational Methods, Knoxville, Tennessee, April 9-11, 1985, Vol. 2, 412 (1985).

8. R. D. Lawrence, "Progress in Nodal Methods for the Solution of the Neutron Diffusion and Transport Equations," Prog. Nuclear Energy, 17, 271 (1986).

9. W. A. Rhoades and R. L. Childs, "The DORT Two-Dimensional Discrete Ordinates Transport Code," Nuclear Science & Engineering, 99, 88 (1988).

10. Y. Y. Azmy, "The Weighted Diamond Difference Form of Nodal Transport Methods," Nuclear Science & Engineering, 98, 29 (1988).