A major purpose of the Technical Information Center is to provide the broadest dissemination possible of information contained in DOE's Research and Development Reports to business, industry, the academic community, and federal, state and local governments.

Although a small portion of this report is not reproducible, it is being made available to expedite the availability of information on the research discussed herein.

LA-UR--84-2609

DE84 016825

TITLE PARALLEL PROCESSING OF NUMERICAL TRANSPORT ALGORITHMS

AUTHOR(S) B. R. Wienke
R. E. Hiromoto

SUBMITTED TO Conference on Vector and Parallel Processors in Computational Science II, Oxford, England, 08/28 - 08/31/84.

DISCLAIMER

Los Alamos Los Alamos National Laboratory
Los Alamos, New Mexico 87545

PARALLEL PROCESSING OF NUMERICAL TRANSPORT ALGORITHMS

B. R. Wienke
R. E. Hiromoto

Computing and Communications Division
Los Alamos National Laboratory
Los Alamos, N.M. 87544

ABSTRACT

The multigroup, discrete ordinates representation for the linear transport equation enjoys widespread computational use and popularity. Serial solution schemes and numerical algorithms developed over the years provide a timely framework for parallel extension. On the Denelcor HEP, we investigate the parallel structure and extension of a number of standard S_n approaches. Concurrent inner sweeps, coupled acceleration techniques, synchronized inner-outer loops, and chaotic iteration are described, and results of computations are contrasted. The multigroup representation and serial iteration methods are also detailed. The basic iterative S_n method lends itself to parallel tasking, portably affording an effective medium for performing transport calculations on future architectures. This analysis represents a first attempt to extend serial S_n algorithms to parallel environments and provides good baseline estimates on ease of parallel implementation, relative algorithm efficiency, comparative speedup, and some future directions. We find basic inner-outer and chaotic iteration strategies both easily support comparably high degrees of parallelism. Both accommodate parallel rebalance and diffusion acceleration and appear as robust and viable parallel techniques for S_n production work.

PRESENTED:

CONFERENCE ON VECTOR AND PARALLEL PROCESSORS IN COMPUTATIONAL SCIENCE II

OXFORD, ENGLAND
AUGUST 28-31, 1984

PAGES AND PROOFS TO:

B.R. Wienke
LANL MS-B206
Los Alamos, N.M. 87544
pages-12 tables-2 figures-3

1. INTRODUCTION

Solutions to the linear transport equation for neutral and charged particles occupy a significant portion of computational efforts in a wide variety of problems within production and specialized code environments. Solution schemes fall into two well-known classes: probabilistic[1] (Monte Carlo) and deterministic[2] methods. Considerable effort has been expended to speed up and optimize solution algorithms on serial machines for a wide range of applications using both methods. As we approach computational limits on single processors, using optimized solution strategies, it is obvious that parallel architectures will provide a faster and more efficient means to perform both kinds of computations. Although parallel machines are in early stages of development, it is clear how parallel algorithms can now be extended from serial schemes for a broad class of transport computations. Multi-acceleration and iteration processes, mesh and material division, coupled forward-adjoint techniques, operator splitting, concurrent biasing, and importance sampling for variance reduction and batching of particle histories are strategies that lend themselves to parallel processing. In this analysis, we examine parallelization of some standard, iterative, multigroup methods for solving the linear Boltzmann equation in the discrete ordinates representation. This particular representation enjoys widespread appeal and use, forming the backbone of numerous production transport and coupled physics analysis modules. We employ the production transport code, ESN,[3] a one-dimensional, Lagrangian, time-dependent S_n module to perform all calculations.

Parallel processing of S_n iterative loops was performed on a single Denelcor HEP[4] Process Execution Module (PEM). The HEP incorporates with every 64-bit data memory word an additional bit for state information which can take one of two states: full or empty. Access to data memory may or may not use this state information. If the full/empty bit is used, memory access is prohibited if one attempts to read from data set empty or write into data set full. Memory access is allowed, on the other hand, if one attempts to read from data set full. This process not only permits read access to occur but blocks subsequent reads by setting the state bit to empty. Conversely, writing into data set empty resets the state bit to full. Enforcement of these accesses is known as "scoreboarding" and is implemented in hardware providing a very flexible, low-overhead synchronization mechanism.

Architecturally, the HEP is a pipelined machine which, combined with data level synchronization, possesses the capability to execute multiple, cooperating parallel streams of instructions. Furthermore, using minor language extensions[5] to Fortran (namely, CREATE to spawn parallel processes and $ variables to invoke scoreboarding memory access), parallel algorithms are easily developed and tested on a single PEM.

Section 2 briefly reviews the multigroup transport equation in the discrete ordinates picture, describes the nested iteration schemes used to solve the equation, and details two acceleration techniques widely employed to speed calculations. Section 3 contrasts parallel extensions of basic serial strategies and details resuts. Section 4 summarizes the analysis.

2. LINEAR TRANSPORT EQUATION AND METHODOLOGY

The multigroup, discrete ordinates equation can be written[6]

$$v_g^{-1}\frac{\partial \phi_{gm}}{\partial t} + \Omega_m \cdot \nabla \phi_{gm} + \sigma_g \phi_{gm} = \sum_{h=1}^{G}\sum_{l=1}^{M-1} \sigma_{hg}^{l} \phi_h^{l} P_l(\Omega_m) + Q_{gm}, \tag{1}$$

as a compact representation of the transport equation with ϕ the particle flux, t the time, Q the external source, v the velocity, σ the transport cross section, Ω a unit vector in the direction of particle travel, and P_l Legendre polynomials. The indices g and m are the discretization indices on the energy and angular domains, respectively, $1 \le g \le G$, $1 \le m \le M$. The multigroup angular fluxes are functions of particle position, direction, energy, and time satisfying the relationship

$$\int_{\epsilon_g}^{\epsilon_{g-1}} d\epsilon \int d\Omega \phi(r,\Omega,\epsilon,t) = 4\pi \sum_{n=1}^{M} w_n \phi_{gn} . \tag{2}$$

with r the position coordinate (one-dimensional slab, cylindrical or spherical geometry), ϵ the energy, and w_n Gaussian quadrature weights for Ω_n which integrate the azimuthally symmetric angular flux. The two quantities with superscripts in Eq. (1), ϕ_h^l and σ_{hg}^l, are the appropriate group-averaged Legendre moments of the flux and differential cross section. The differential cross sections also satisfy the normalization,

$$\sigma_g^l = \sum_{h=1}^{G} \sigma_{gh}^l . \tag{3}$$

Spatial differencing of the streaming term can be highly problem dependent, particle dependent (neutral or charged) and required to satisfy conservation laws.[2,3] Linear discontinuous, exponential, and diamond schemes[2,3,6-8] have all been used with success in applications involving neutral and charged particles. Angular differencing is usually based on the diamond approximation[2]

$$2\phi_{gm} = \phi_{gm+1/2} + \phi_{gm-1/2} , \tag{4}$$

with $m \pm 1/2$ denoting the edge values. In order to recover the exact representation of the angular part of the streaming operator[2] in the various one-dimensional geometries, we introduce recursive angular coefficients, $\alpha_{m+1/2}$, given by

$$\alpha_{m+1/2} - \alpha_{m-1/2} = -(k-1) w_m \mu_m , \tag{5}$$

with $k=1,2,3$ for plane, cylindrical, or spherical geometry. It follows[2,5] from Eq.(4) for initial values,

$$\alpha_{1/2} = \alpha_{M+1/2} = 0 \tag{6}$$

that the full streaming operator can be written generally

$$\Omega \cdot \nabla \phi_{gm} = \mu_m \frac{\partial \phi_{gm}}{\partial r} + \mu_m \gamma_m r^{-1} (\phi_{gm} - \phi_{gm\ 1/2}) \tag{7}$$

for

$$\gamma_m = (\alpha_{m+1/2} + \alpha_{m-1/2}) / w_m \mu_m \tag{8}$$

and r the appropriate spatial coordinate, as before. The time derivative is differenced in the implicit picture

$$v_g^{-1} \frac{\partial \phi_{gm}}{\partial t} = v_g^{-1} \frac{(\phi_{gm}^k - \phi_{gm}^{k-1})}{\Delta t} , \tag{9}$$

with k the time index. All other quantities in Eq. (1) carry the current value of k.

Equation (1) is an integro-differential expression that is solved by successive flux iteration on the first term of the right-hand side.[8-10] Iterations with $h=g$ are the within-group, or *inner* iterations, while iterations for $h \neq g$ represent *outer* iterations. To speed up these cycles, flow rebalance (enforcing

particle conservation locally) and diffusion acceleration (coupling diffusion solutions to kernel iterations) techniques are employed to cut computational time.[11,12] For clarity, it is convenient to rewrite the transport equation in compact operator form, where Φ is the flux,

$$\mathbf{L}\Phi + \Sigma\Phi = (\mathbf{S} + \mathbf{U} + \mathbf{D})\Phi + \mathbf{Q} \tag{10}$$

and $\mathbf{L}$, Σ, $\mathbf{S}$, $\mathbf{U}$, $\mathbf{D}$ and $\mathbf{Q}$ are the streaming, collision, self-scatter, upscatter, downscatter, and external source operators, respectively. Upscatter or downscatter underscore energy transfers between higher or lower energy groups, while self-scatter implies no energy transfer. Upscatter and downscatter couple different energy groups (*outer*), while self-scatter only couples within-group fluxes (*inner*). Denoting an iteration index, i, the outer scheme takes the symbolic form

$$\mathbf{L}\Phi^{i+1} + \Sigma\Phi^{i+1} = \mathbf{S}\Phi^{i+1} + (\mathbf{D} + \mathbf{U})\Phi^{i} + \mathbf{Q}\,. \tag{11}$$

The outer source, $(\mathbf{D} + \mathbf{U})\Phi^{i}$, is computed from the previous iterate, and the external source, $\mathbf{Q}$, is constant. For inner iterations, we define a fixed effective source, $\mathbf{QQ}$,

$$\mathbf{QQ}^{i} = (\mathbf{D} + \mathbf{U})\Phi^{i} + \mathbf{Q} \tag{12}$$

in the solution sweeps from higher to lower energy groups. Having obtained $\mathbf{QQ}^{i}$, one then solves the within-group equation in the inner strategy

$$\mathbf{L}\Phi^{i+1,j+1} + \Sigma\Phi^{i+1,j+1} = \mathbf{S}\Phi^{i+1,j} + \mathbf{QQ}^{i}, \tag{13}$$

with j denoting the inner iteration cycle, in analogy to the outer index i. In this computational cycle, Eq. (13) is iterated until convergence is met. Each outer iteration thus involves one pass through all energy groups in solving Eq. (13). At the end of each outer cycle, convergence is again tested and $\mathbf{QQ}^{i}$ is updated for the next iteration, if necessary. This dual iteration strategy is popularly called the inner-outer sweep, with spectral radius and convergence properties determined by the norm of the iteration matrix, $(\mathbf{L} + \Sigma)^{-1}\mathbf{S}$

For most applications, the nested inner-outer scheme described above converges rapidly. However, there exist problems (optically thick regions, pure scattering, little absorption and spectral radius close to 1) for which algorithms require excessive iterations for convergence. To remedy these problems, both mesh rebalance[2,12,13] and diffusion acceleration[9,11] have proved useful. Mesh rebalance effectively

enforces conservation after inner cycles in the following general way. Defining leftward and rightward flows in zone k,

$$LL_g^{k-1/2} = \sum_{n=1}^{M/2} w_n \mu_n A^{k-1/2} \phi_{gn}^{k-1/2}$$

$$RL_g^{k+1/2} = \sum_{n=M/2}^{M} w_n \mu_n A^{k+1/2} \phi_{gn}^{k+1/2}, \tag{14}$$

effective absorption,

$$AB_g^k = (\sigma_g - \sigma_{gg}) \sum_{n=1}^{M} w_n V^k \phi_{gn}^k, \tag{15}$$

and source,

$$SS_g^k = \sum_{n=1}^{M} w_n V^k Q Q_{gn}^k , \tag{16}$$

with A and V the appropriate zone transverse areas and volumes, one *rebalances* the fluxes by first solving the equation for the rebalance factors, f,

$$f_g^k (LL_g^{k-1/2} + RL_g^{k+1/2} + AB_g^k) = SS_g^k + f^{k-1} RL_g^{k-1} + f^{k+1} LL_g^{k+1} \tag{17}$$

and then applying the factors, f_g^k, multiplicatively to the angular fluxes, ϕ_{gm}^k, before the next iteration. Equation (17) represents a tridiagonal system for the rebalance factors which may be solved by forward elimination-backward substitution. At convergence, $f_g^k \rightarrow 1$.

Diffusion acceleration is similar to rebalance but relies upon solution of a corrected diffusion equation for the scalar flux to accelerate transport iterations. The corrected diffusion equation for the scalar flux, ϕ_g, is written

$$-\nabla \cdot d_g \nabla \phi_g + \lambda_g^{-1} \phi_g = QQ_g - C_g, \tag{18}$$

where

$$d_g = 1/3\sigma_g$$

$$\lambda_g^{-1} = \sigma_g - \sigma_{gg}, \tag{19}$$

and the correction term, C_g, represents the flow difference between the streaming transport and diffusion operators

$$C_g = \nabla \cdot \mathbf{J}_g + \nabla \cdot d_g \nabla \phi_g, \tag{20}$$

for

$$\mathbf{J}_g = \int d\Omega \Omega \phi_g(r,\Omega,t). \tag{21}$$

The approach used to accelerate transport solutions is to first solve the transport equation for the angular flux, ϕ_{gm}, construct the source correction, C_g, using Eq. (20), and then obtain an estimate of the diffusion corrected scalar flux from Eq. (20). This corrected estimate of the scalar flux is finally used in the scattering term on the right-hand side of Eq. (1) to accelerate iterations on the transport kernel. At convergence, the balanced transport equation (integrated over angle) results from Eqs. (18) and (20). Variants of both methods have proved useful in practice.[9-12]

3. PARALLEL ITERATION SCHEMES

The foregoing strategies have evolved over a number of years in production applications. They form a compact nucleus for developing parallel algorithms. Since their use is widespread, investigation of their parallel implementation is also timely. Before detailing their parallelization, a few comments are appropriate.

The particular inner-outer iteration scheme described above and employed in the bulk of existing S_n computational modules is well suited for parallel processing. Each group can be preassigned a parallel process for the energy sweeps in outright fashion, or self-scheduled among parallel processes as they become available. Inner sweeps are then synchronized within outer loops in natural analogy with serial methods. Convergence tests on the flux iterates can also be performed in the standard inner-then-outer fashion presently used, or the flux iterates can be chaotically updated and tested for overall convergence. Chaotic iteration methods are easily implemented inside the inner-outer loops, ranging from fully to partially chaotic depending on the number of processes updating the flux iterates.

Acceleration schemes can also be coupled to inner-outer sweeps by employing flux iterates from previous cycles to accelerate the calculation. Both rebalance and diffusion synthetic computations can be backstepped one iteration cycle and synchronized with the present transport sweep. Parallel acceleration routines can also be loaded inside loops. For chaotic iterations, acceleration techniques can be applied randomly, sequentially, or after selective convergence tests.

We examine two parallel iteration schemes, termed TPMG (multigroup) and TPCC (chaotic convergence), that take the operational forms, respectively,

$$\mathbf{L}\Phi^{l+1,j+1} + \Sigma\Phi^{l+1,j+1} = \mathbf{S}\Phi^{l+1,j} + (\mathbf{D} + \mathbf{U})\Phi^{l} + \mathbf{Q} \tag{22}$$

$$\mathbf{L}\Phi^{l+1,j+1} + \Sigma\Phi^{l+1,j+1} = \mathbf{S}\Phi^{l+1,j} + \mathbf{D}\Phi^{l+1} + (1-\gamma)\mathbf{U}\Phi^{l} + \gamma\mathbf{U}\Phi^{l+1} + \mathbf{Q}\,. \tag{23}$$

The parameter γ quantifies the degree of *chaoticity* of TPCC. It depends on the number of processes spawned per total number of energy groups in the application. With a single spawned process, $\gamma = 0$, while for spawned processes equal to the number of energy groups, $\gamma = 1$. The *chaoticity* of TPCC ranges from non-chaotic to fully chaotic as γ ranges from 0 to 1. The scheme TPMG is completely ordered, as in the serial case. On the other hand, TPCC becomes increasingly disordered as the number of processes spawned approaches the number of groups. Because TPMG is completely ordered, one expects the total iteration count at convergence to be independent of the number of parallel processes assigned to the task, and to equal the serial count. Iteration counts for TPCC are expected to vary with spawned processes.

For test purposes, two model problems were chosen. The first is a 5-group boundary source of electrons, Maxwellian distributed in energy at T=10 keV and isotropic in direction, incident on a thin slab varying in thickness from 2 to 4 mean free paths. The scattering ratio, $\sigma_s/\sigma_s+\sigma_a$, with σ_s and σ_a the scattering and absorption cross sections, is unity (no absorption). Only adjacent groups are coupled by the scattering matrix in a Fokker-Planck approximation. The second is a 16-group, isotropic boundary source of photons, Planckian distributed in energy at T=100 keV, incident on a sphere of 6 to 10 mean free paths thickness. The scattering ratio varies from 0.6 to 0.85 and all energy groups are Compton coupled. A S_4 quadrature is assigned in both test cases. ESN[3] is used for calculations with convergence cri-

teria of 10^{-3}. Both problems induce representative demands on the S_n method that are realistic and common in applications.

Our first (5-group, pure scatterer) test problem was run serially in both standard inner-outer and chaotic modes, with and without acceleration, to provide baselines. The 5 static, inner, group loops were preassigned and processed in parallel. Synchronization was enforced on the inner loops before outer convergence tests. Simple coarse mesh rebalance was employed to accelerate convergence at the beginning of each outer cycle, using the previous, or last, set of flux iterates. The execution times (sec) for various cases are compared below in Table 1, with TPMG and TPCC denoting the parallel inner-outer and chaotic iteration strategies, respectively. Iteration cycle counts are listed in Table 2.

TABLE 1. Scatterer Test Problem Execution Times (sec)

MODE	ACCELERATION	TPMG	TPCC
serial			
	none	1642.2	1208.4
	rebalance	1314.5	982.6
parallel			
	none	341.3	318.6
	rebalance	273.8	224.1

TABLE 2. Scatterer Test Problem Iteration Count

MODE	ACCELERATION	TPMG	TPCC
serial			
	none	300	240
	rebalance	240	187
parallel			
	none	300	212
	rebalance	240	150

Tables 1 and 2 exhibit some interesting features. Parallel speedup in execution time is near the theoretical maximum[15] of 5, with and without acceleration. This is not surprizing since the very large granularity of the iteration schemes overshadow overhead in executing 5 processes in parallel. Chaotic iteration schemes are certainly faster than ordered inner-outer sweeps, but by a smaller margin in the parallel case. Rebalance easily accelerates both serial and parallel computations by factors of 20% to 30%. Clearly

parallel chaotic iteration with rebalance is about 8 times faster than serial inner-outer iteration with no acceleration as the slowest case. Iteration counts for the standard inner-outer schemes are the same serially or in parallel. Again, this is to be expected since inner loops were synchronized before outer convergence tests and is, of course, the reason the parallel speedup in this test problem is close to 5. Concurrent rebalance acceleration cut both run times and iteration counts. The parallel chaotic sweeps require fewer iterations than parallel inner-outer sweeps and less time to inner convergence. The performance (speedup) of the parallel chaotic scheme results from the fact that inner loops are also passing converged flux iterates to all other loops as scattered outer sources. Both the inner-outer (TPMG) and chaotic (TPCC) iteration patterns show noteworthy parallel gain and are easily implemented in any standard S_n module.

The second test problem (16-group, absorber-scatterer) was run serially and in parallel. A dynamical scheduling technique was implemented allowing group sweeps to be executed in parallel using varying numbers of processes (1 to 16). Diffusion synthetic acceleration was employed in the inner sweeps in all calculations. Both inner-outer and chaotic iteration patterns were tested. Results are summarized in Figures 1 through 3. Figure 1 contrasts raw execution times (100 sec) for inner-outer (TPMG) and chaotic (TPCC) schemes. Figure 2 plots relative speedup over serial processing time, and Figure 3, in analogy with Table 2, contrasts total iteration counts with 1 to 16 processors.

Results for the second problem reinforce the first. Execution times for TPCC are consistently lower than TPMG for any number of processes assigned to the inner loop, except in the case of 8 or 9 processes for which execution times are roughly the same. Both iteration schemes exhibited their optimal performance with 16 processes. The overall parallel gain with increasing numbers of processes is a monotonic curve for both schemes. Although TPCC is faster than TPMG overall, the relative speedup, or gain, over single process execution time is greater for the parallelized inner-outer scheme. Both schemes show dramatic relative speedup for 16 processes, corresponding to the execution time minima at 16 processes. Iteration counts for TPMG are requisite constants, as in the preceding case. The iteration count for TPCC also has a relative minimum at 16 processes, dropping down by a factor of roughly 45% over adjacent values. Again, both the inner-outer and chaotic schemes show significant parallel gain.

4. SUMMARY

Analysis of the foregoing iteration strategies on the transport kernel indicate speedup roughly proportional to the number of processes for inner-outer and chaotic schemes with concurrent acceleration. Chaotic iteration is slightly faster than the parallelized standard inner-outer algorithm. Both schemes easily generalize to parallel environments. Concurrent rebalance or diffusion acceleration appear similarly effective in both the parallel and serial case. In both iteration strategies, major loops are parallelized to take advantage of the large computational granularity necessary for effective parallel performance. Simple parallel processing of S_n inner group sweeps has been shown to afford significant savings in computational time with an absolute minimum of reprogramming and retrofitting. In production modules, we have demonstrated that these approaches will have unusually high payoff.

These initial parallel processing efforts suggest a number of additional research activities which we are actively pursuing. Coupled inner-outer and chaotic schemes for specialized physical situations (such as upscatter predominance or pure scatter), chaotic divergence, block iteration, and optimized acceleration techniques are under investigation.

5. REFERENCES

(1) L.L. Carter and E.D. Cashwell, *Particle Transport Simulation With The Monte Carlo Method*, ERDA Critical Review Series, U.S. Energy and Research Development Administration, Oak Ridge (1975).

(2) B.G. Carlson and K.D. Lathrop, *Computing Methods In Reactor Physics*, Gordon and Breach, New York (1968).

(3) B.R. Wienke, J. Quant. Spect. Rad. Trans. 28, 311 (1982); Nuc. Sci. Eng. 81, 302 (1982).

(4) H.F. Jordan, "Experience With Pipelines Multiple Instruction Streams," to appear Proc. IEEE, January (1984).

(5) *HEP Fortran Users Guide*, Denelcor Inc., Aurora (1982).

(6) G.I. Bell and S. Glasstone, *Nuclear Reactor Theory*, Van Nostrand Reinhold, New York (1970).

(7) B.G. Carlson, Nuc. Sci. Eng. 18, 149 (1976).

(8) R.E. Alcouffe, E.W. Larsen, W.F. Miller, and B.R. Wienke, Nuc. Sci. Eng. 71, 111 (1979).

(9) W.H. Reed, Nuc Sci. Eng. 45, 245 (1971).

(10) K.D. Lathrop, J. Comp Phys. 4, 475 (1969).

(11) R.E. Alcouffe, Nuc. Sci. Eng. 64, 344 (1977).

(12) T.R. Hill, "ONETRAN: Discrete Ordinates Finite Element Code For The Solution Of The One Dimensional Muligroup Transport Equation," Los Alamos National Laboratory Report *LA-5990-MS*, June (1979).

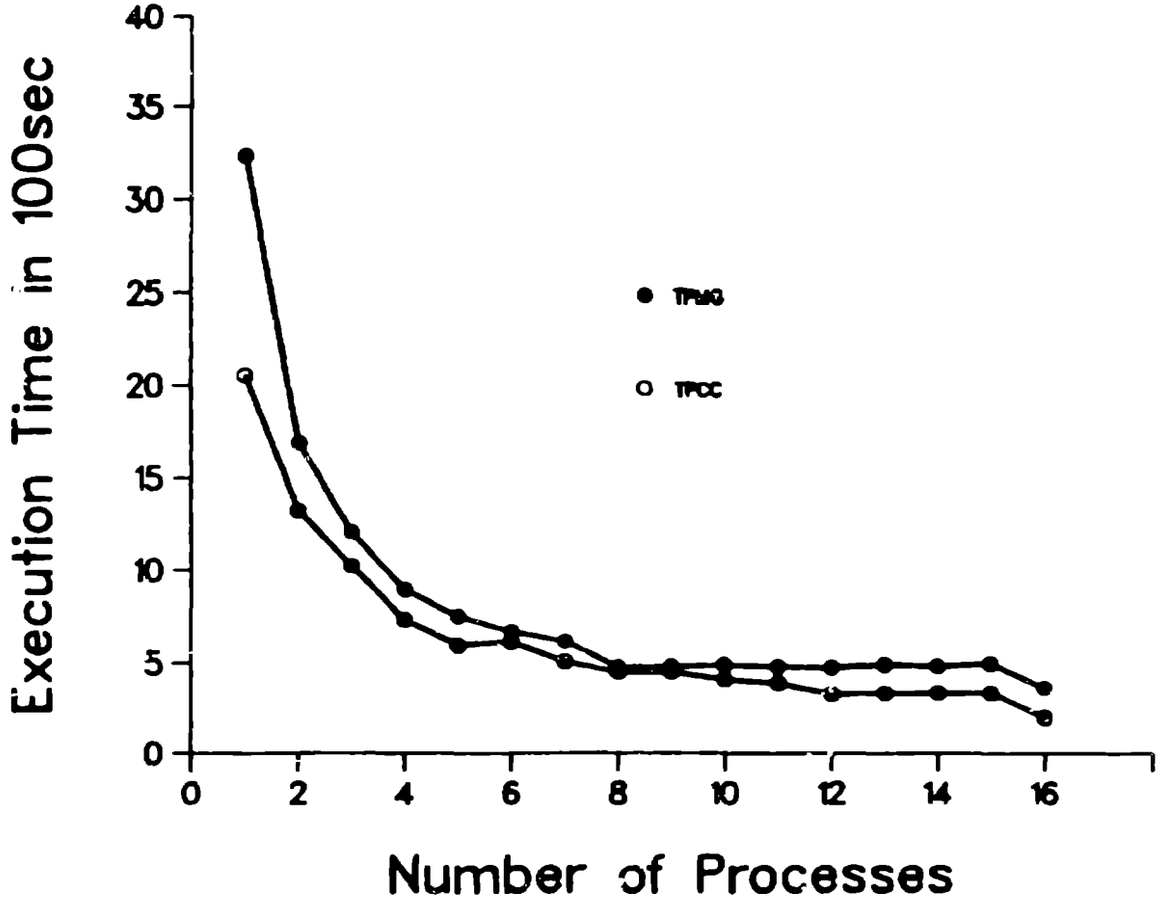
(13) S. Nakamura, Nuc. Sci. Eng. 39, 535 (1958).

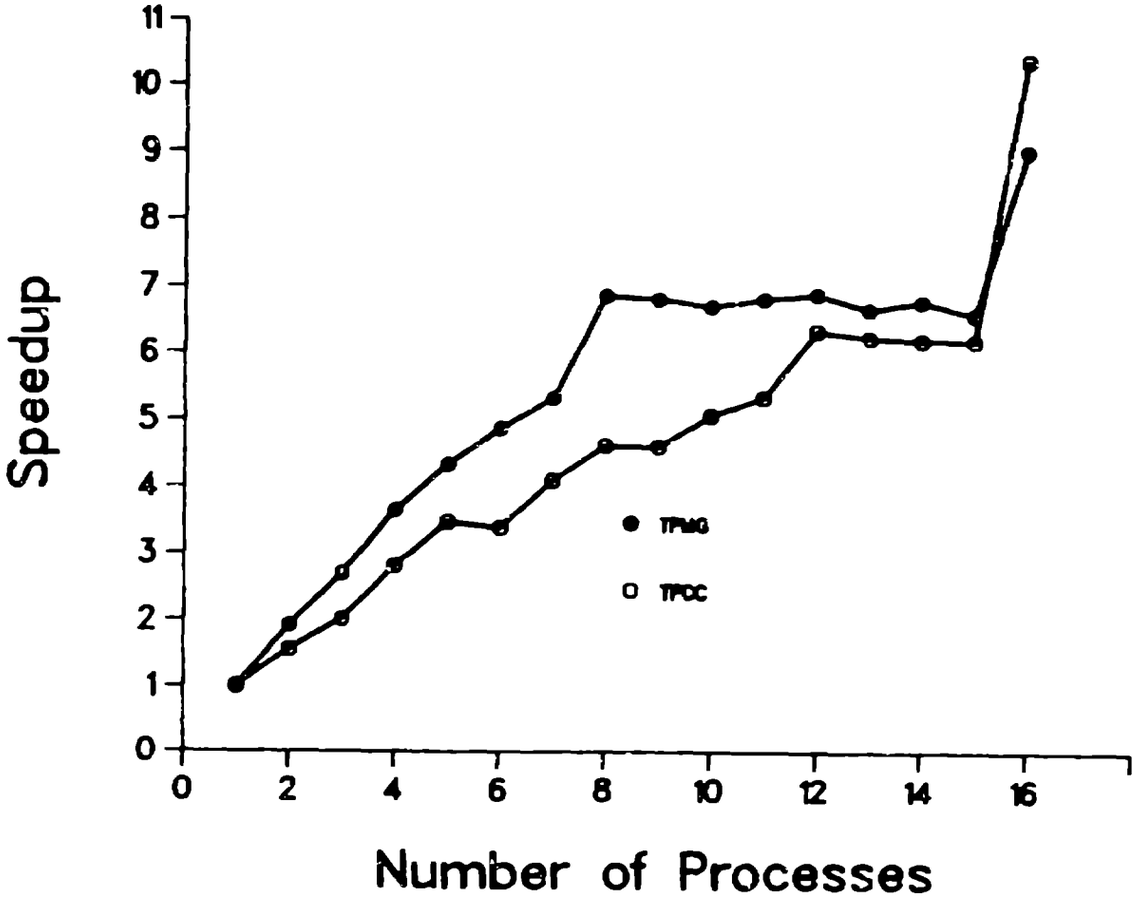
(14) B.J. Smith, *Proceedings Of The* 1978 *International Conference On Parallel Processing*, 6 Bellair (1978).

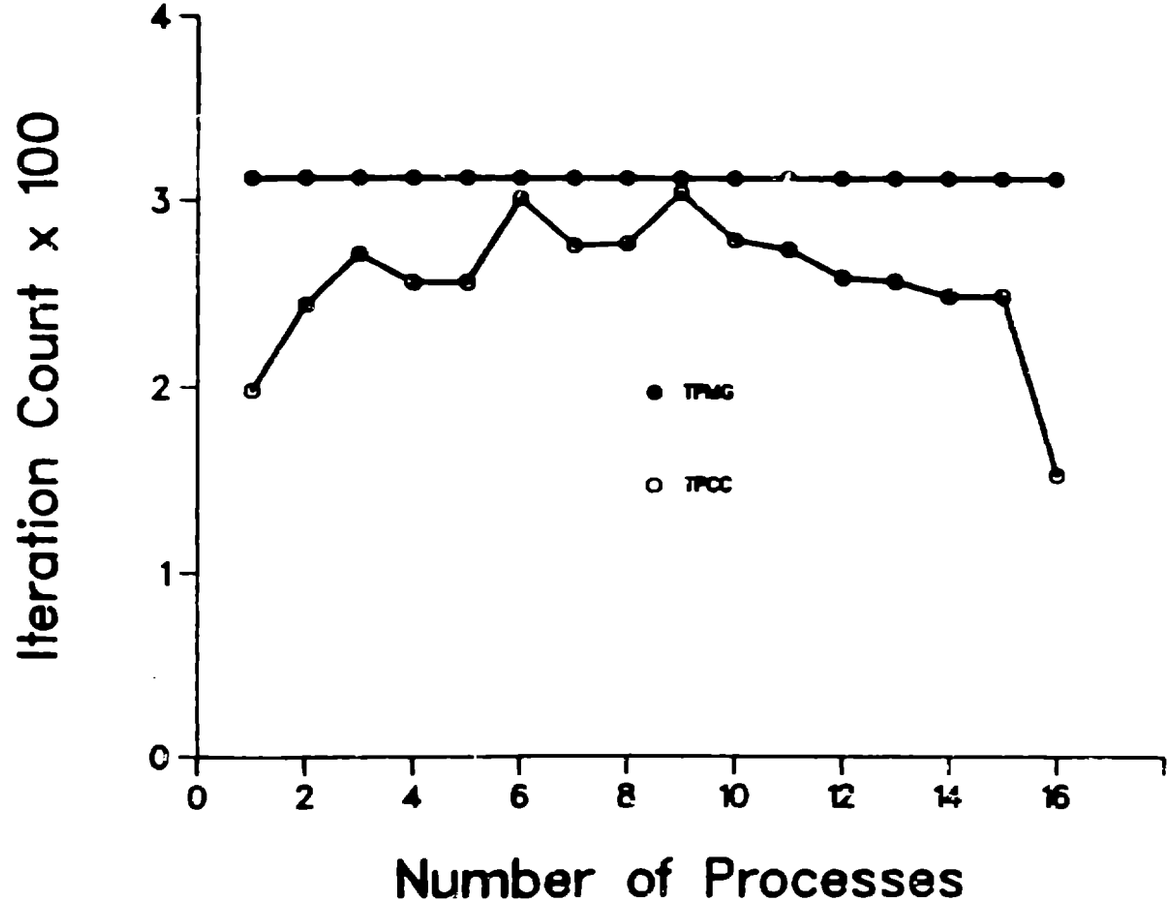
(15) B.L. Buzbee, *Los Alamos Science* 9, 71 (1984).

6. LIST OF TABLES

7. LIST OF FIGURES


Execution Time in 100sec
40
35
30
25
20
15
10
5
0
0
2
4
6
8
10
12
14
16
Number of Processes
TPMG
TPCC


Speedup
Number of Processes
TPMG
TPCC
0
1
2
3
4
5
6
7
8
9
10
11
0
2
4
6
8
10
12
14
16


Iteration Count x 100
4
3
2
1
0
0
2
4
6
8
10
12
14
16
Number of Processes
TPMG
TPCC