LA-UR- 10-08284

Approved for public release; distribution is unlimited.

Title:

Implementation of a Cell-Wise Block-Gauss-Seidel Iterative Method for SN Transport on a Hybrid Parallel Computer Architecture

Author(s):

Massimiliano Rosa, James S. Warsa
Computational Physics (CCS-2)

Michael Perks
IBM Corporation

Intended for:

American Nuclear Society
M&C 2011 Topical Meeting
May 8-12, 2011
Rio de Janeiro, RJ, Brazil

International Conference on Mathematics and Computational Methods Applied to Nuclear Science and Engineering (M&C 2011)
Rio de Janeiro, RJ, Brazil, May 8-12, 2011, on CD-ROM, Latin American Section (LAS) / American Nuclear Society (ANS)
ISBN 978-85-63688-00-2

IMPLEMENTATION OF A CELL-WISE BLOCK-GAUSS-SEIDEL ITERATIVE METHOD FOR SN TRANSPORT ON A HYBRID PARALLEL COMPUTER ARCHITECTURE

Massimiliano Rosa and James S. Warsa
Los Alamos National Laboratory
P.O. Box 1663, MS K784, Los Alamos, NM 87545, USA
maxrosa@lanl.gov; warsa@lanl.gov

Michael Perks
IBM Corporation
11501 Burnet Rd, Austin, TX 78758, USA
mperks@us.ibm.com

ABSTRACT

We have implemented a cell-wise, block-Gauss-Seidel (bGS) iterative algorithm, for the solution of the S_n transport equations on the Roadrunner hybrid, parallel computer architecture. A compute node of this massively parallel machine comprises AMD Opteron cores that are linked to a Cell Broadband Engine^{TM} (Cell/B.E.)[1]. LAPACK routines have been ported to the Cell/B.E. in order to make use of its parallel Synergistic Processing Elements (SPEs). The bGS algorithm is based on the LU factorization and solution of a linear system that couples the fluxes for all S_n angles and energy groups on a mesh cell. For every cell of a mesh that has been parallel decomposed on the higher-level Opteron processors, a linear system is transferred to the Cell/B.E. and the parallel LAPACK routines are used to compute a solution, which is then transferred back to the Opteron, where the rest of the computations for the S_n transport problem take place. Compared to standard parallel machines, a hundred-fold speedup of the bGS was observed on the hybrid Roadrunner architecture. Numerical experiments with strong and weak parallel scaling demonstrate the bGS method is viable and compares favorably to full parallel sweeps (FPS) on two-dimensional, unstructured meshes when it is applied to optically thick, multi-material problems. As expected, however, it is not as efficient as FPS in optically thin problems.

Key Words: neutral particle transport, hybrid parallel computer architecture.

1. INTRODUCTION

We recently analyzed the stability and convergence of the one-group, cell-wise block-Jacobi (bBJ) and cell-wise block-Gauss-Seidel (bGS) algorithms for solving the S_n transport equations in two dimensions [1]. These iterative solution methods were specifically targeted for implementation on the Roadrunner [2] class of hybrid parallel computers, several of which have been assembled at Los Alamos National Laboratory (LANL) [3]. Fourier analysis indicates that both algorithms are rapidly convergent in problems with optically thick mesh cells and bGS approaches the asymptotic, thick-cell regime at convergence rates higher than bBJ.

We have implemented the bGS iteration on the Roadrunner machines at LANL with the expectation that the combination of fast factorization algorithms available on the Cell/B.E. and rapid convergence rate associated with the bGS algorithm will make the bGS iteration competitive with the full parallel transport sweeps (FPS) that is typically used in parallel S_n transport applications [4,5]. Specifically, we investigate the range of problems for which the bGS algorithm is more efficient than FPS on unstructured meshes.

The remainder of the paper is organized as follows. In Sec. 2 we discuss how bGS is applied to the multi-group S_n equations. Single- and dual-threaded strategies for the numerical implementation of multi-group bGS on Roadrunner are illustrated in Sec. 3. Numerical results are discussed in Sec. 4. Finally, in Sec. 5 we present a summary of our main findings and conclusions.

2. MULTI-GROUP CELL-WISE BLOCK-GAUSS-SEIDEL (bGS) ALGORITHM

2.1. Review of one-group formulation of bGS

The bGS algorithm for the steady-state, one-group, non-multiplying transport equation with isotropic scattering is based on an operator splitting of

$$(L-SD)\psi = q\,, \tag{1}$$

where L, S and D represent the "streaming plus total interaction" operator, the scattering operator, and the "discrete-to-moment" operator (which represents integration over all S_n angles,) respectively; ψ is the angular flux and q is the fixed source. The cell-wise bGS splitting of the L operator is $L = L_c^* + L_b^*$, where L_c^* acts on the angular fluxes within a mesh cell and L_b^* acts on all the angular fluxes whose directions are oriented such that they are incoming relative to the faces of neighboring mesh cells [1]. Within this splitting, Eq. (1) is written in the form

$$\left[I + \left(L_c^* - SD\right)^{-1} L_b^*\right]\psi = \left(L_c^* - SD\right)^{-1} q\,, \tag{2}$$

where I is the identity operator. This corresponds to a linear system for all S_n angles and all spatial degrees of freedom (we use a Discontinuous Finite Element Method (DFEM) on triangles [6].) Equation (2) is solved iteratively with restarted Generalized Minimum Residual (GMRES) such that, for each iteration, the action of the $\left(L_c^* - SD\right)^{-1}$ operator is computed using a LU decomposition for any mesh cell on a parallel sub-domain. Specifically, all angular fluxes for all the spatial degrees of freedom on a given mesh cell are computed simultaneously, one mesh cell at a time. The ordering in which this "bGS-sweep" takes place depends only on the numbering of the mesh. The distinction between the bBJ and bGS algorithms is that for bGS the most recently computed angular fluxes are used when the term involving L_b^* is constructed for a given mesh cell, while for bBJ this term uses angular fluxes that are "lagged" from the previous iteration [1].

2.2. Multi-group formulation of bGS

We consider the steady-state, multi-group transport problem using two energy groups to illustrate the concepts presented during this discussion. Later, results with more groups will be presented. The two-group homogeneous transport problem is, in operator notation,

$$\begin{cases} \hat{\Omega}\cdot\nabla\psi_1 + \sigma_1\psi_1 = \sigma_{S11}\Phi_1 + \sigma_{S12}\Phi_2 \\ \hat{\Omega}\cdot\nabla\psi_2 + \sigma_2\psi_2 = \sigma_{S21}\Phi_1 + \sigma_{S22}\Phi_2 \end{cases}. \tag{3}$$

The terms on the left side in Eq. (3) are the "streaming plus total interaction" operators for each group and the scalar fluxes on the right side of Eq. (3) are the integral of the group angular fluxes over all angles, performed via the "discrete-to-moment" operator for each group. The σ_{Sij} cross-sections represent isotropic scattering from group *j* into group *i*.

The coupled system in Eq. (3) is often solved with source iteration (SI) which is written for iteration ℓ as

$$\begin{pmatrix} \psi_1^{(\ell+1)} \\ \psi_2^{(\ell+1)} \end{pmatrix} = \begin{bmatrix} L_1 & 0 \\ 0 & L_2 \end{bmatrix}^{-1} \begin{bmatrix} \sigma_{S11} & \sigma_{S12} \\ \sigma_{S21} & \sigma_{S22} \end{bmatrix} \begin{bmatrix} D & 0 \\ 0 & D \end{bmatrix} \begin{pmatrix} \psi_1^{(\ell)} \\ \psi_2^{(\ell)} \end{pmatrix}. \tag{4}$$

On a parallel computer, it is the block inverses on the right side of Eq. (4) that are computed with FPS. The FPS forms the bulk of the computational effort in a S_n transport calculation. Efficiency of the FPS degrades as the number of processors increases. Note that this version of SI does not use nested, within-group iterations typically used in transport codes.

The two-group bGS algorithm employs the same spatial splitting of the group "streaming plus total interaction" operators, as described in Sec. 2.1, for each of the two groups. In this case, at iteration ℓ, the bGS algorithm is:

$$\begin{pmatrix} \psi_1^{(\ell+1)} \\ \psi_2^{(\ell+1)} \end{pmatrix} = -\left\{ \begin{bmatrix} L_{c1}^* & 0 \\ 0 & L_{c2}^* \end{bmatrix} - \begin{bmatrix} \sigma_{S11} & \sigma_{S12} \\ \sigma_{S21} & \sigma_{S22} \end{bmatrix} \begin{bmatrix} D & 0 \\ 0 & D \end{bmatrix} \right\}^{-1} \begin{bmatrix} L_{b1}^* & 0 \\ 0 & L_{b2}^* \end{bmatrix} \begin{pmatrix} \psi_1^{(\ell)} \\ \psi_2^{(\ell)} \end{pmatrix}. \tag{5}$$

Again, the bGS variation on bBJ uses a mix of updated fluxes $\psi^{(\ell+1)}$ and lagged fluxes $\psi^{(\ell)}$ operated on by the L_b^* operators, which depends on the numbering of the mesh cells. The combination of operations on the right side of Eq. (5) is the bGS-sweep.

The convergence rate of SI is the spectral radius of the matrix on the right side of Eq. (4). Fourier analysis indicates that the flat error mode is the dominant eigenvalue. Because that error mode is constant in space, particle streaming can be ignored and we can compute the spectral radius of the iteration matrix on the right side of

$$\begin{pmatrix}\Phi_1^{(\ell+1)}\\ \Phi_2^{(\ell+1)}\end{pmatrix}=\begin{bmatrix}\sigma_1 & 0\\ 0 & \sigma_2\end{bmatrix}^{-1}\begin{bmatrix}\sigma_{S11} & \sigma_{S12}\\ \sigma_{S21} & \sigma_{S22}\end{bmatrix}\begin{pmatrix}\Phi_1^{(\ell)}\\ \Phi_2^{(\ell)}\end{pmatrix}, \tag{6}$$

to investigate the asymptotic convergence rate of SI. That is, if we let

$$\mathbf{T}_{SI2g}=\left(\mathbf{\Sigma}_T\right)^{-1}\mathbf{\Sigma}_S=\begin{bmatrix}\sigma_1 & 0\\ 0 & \sigma_2\end{bmatrix}^{-1}\begin{bmatrix}\sigma_{S11} & \sigma_{S12}\\ \sigma_{S21} & \sigma_{S22}\end{bmatrix}, \tag{7}$$

the two-group spectral radius is

$$\rho_{SI2g}=\max\left|Eig\left(\mathbf{T}_{SI2g}\right)\right|. \tag{8}$$

The scattering cross-sections in Eq. (7) must satisfy the inequalities:

$$\begin{cases}\sigma_{S11}+\sigma_{S21}\le\sigma_1\\ \sigma_{S12}+\sigma_{S22}\le\sigma_2\end{cases}. \tag{9}$$

If equality holds in Eq. (9), then the spectral radius of the $\mathbf{T}_{SI2g}$ matrix is exactly equal to 1, regardless of the numerical values of the cross-sections. This result applies for any number of energy groups. For example, we devise a problem for which ρ for two-group SI is 0.9999 by enforcing the constraint:

$$\begin{cases}\sigma_{S11}+\sigma_{S21}=0.9999\sigma_1\\ \sigma_{S12}+\sigma_{S22}=0.9999\sigma_2\end{cases}\Rightarrow\rho_{SI2g}=0.9999. \tag{10}$$

In this case, source iteration would converge slowly.

On the other hand, the convergence of GMRES depends on the shape of the eigenvalue spectrum and not just the size of the dominant eigenvalue [7]. Random scattering matrices can be used to generate an iteration matrix whose spectrum is widely distributed in the complex plane, with eigenvalues that are small relative to the size of the spectrum, a situation in which GMRES would converge slowly. Therefore, we will devise problems for which both SI and GMRES converge slowly by generating a dense, random, scattering matrix and compute the total cross-sections from the constraints that fix $\rho\sim 1$.

We begin constructing such a multi-group problem with G energy groups by creating a random $G\times G$ matrix $\mathbf{R}$, with entries $r_{gg'}$ in the range $[0,1]$, and re-normalize the elements on each row of matrix $\mathbf{R}$ so that their sum equals 1. We then use the re-nomarlized matrix elements to construct the scattering cross-section matrix

$$\sigma_{Sgg'} = \frac{r_{gg'}}{\sum_{g'=1}^{G} r_{gg'}}, \quad g, g' = 1,..,G. \tag{11}$$

Note that re-normalization is not strictly necessary to obtain a desired ρ for multi-group SI. We do so only to ensure the total cross-sections obtained in each group are not far from unity. If we take the total cross-sections to be proportional to the sums of the entries on the columns of the scattering matrix

$$\sigma_{g'} = \left(1+\alpha_{g'}\right)\sum_{g=1}^{G}\sigma_{Sgg'}, \quad g' = 1,..,G, \tag{12}$$

then coefficients $\alpha_{g'}$ are obtained as follows:

$$\alpha_{g'} = l + \left(u-l\right) r_{g'}, \quad g' = 1,..,G, \tag{13}$$

where $r_{g'}$ is a random value on [0,1], such that the α coefficients have random values on [*l*,*u*]. For example, by selecting $l = 10^{-5}$ and $u = 10^{-4}$, we obtain ρ for multi-group SI comprised between 0.9999 and 0.99999. If $l = u = 10^{-4}$, then ρ is exactly equal to 0.9999. Physically, this means that each α coefficient introduces a small neutron absorption in its corresponding energy group.

To verify this procedure for multi-group SI, we compare the value of ρ predicted via Eq. (8) with Fourier analysis for two-group SI, using cross-sections obtained by selecting $l = 10^{-5}$ and $u = 10^{-4}$. This leads to scattering cross-sections $\sigma_{S11} = 6.1789\times10^{-1}$, $\sigma_{S12} = 3.8211\times10^{-1}$, $\sigma_{S21} = 9.2747\times10^{-1}$ and $\sigma_{S22} = 7.2534\times10^{-2}$, and total cross-sections $\sigma_1 = 1.5454$ and $\sigma_2 = 4.5468\times10^{-1}$, in cm^{-1}. Using these values in Eq. (8) leads to $\rho_{SI2g} = 0.99995$ for this model problem. Fourier analysis on a four-triangle, $1cm \times 1cm$ square with level-symmetric S_4 quadrature gives the same result. We have verified, as well, that the maximum eigenvalue is found at the Fourier wave numbers $\lambda_x = \lambda_y = 0$, namely, it is the flat error mode which is most slowly attenuated by multi-group SI.

Fourier analysis for two-group SI was also compared with numerical results obtained from a multi-group FPS implementation in 2D Cartesian coordinates. Measured estimates for ρ obtained for the model problem on an $I \times J$ domain are compared with the Fourier analysis in Table I for a sequence of meshes of squares with sides of length dx = dy. Every Cartesian element in the mesh is subdivided into four triangular cells. The difference in the results presented in Table I is explained because the Fourier analysis is over an infinite medium whereas the actual spectrum incorporates the effect of particle leakage at the boundaries of the problem, for which vacuum boundary conditions were specified. As the mesh size is increased, for a given element width, the effect of leakage becomes less significant. Overall the spectral properties of multi-group SI appear to be similar to the well known spectral properties of one-group SI.

Table I. Theoretical and computed spectral radius of two-group SI

dx = dy	10^{-1}	10^{0}	10^{+1}	10^{+2}	10^{+3}	10^{+4}	10^{+5}	10^{+6}
1×1	0.06336	0.41485	0.91905	0.99831	0.99993	0.99995	0.99995	0.99995
2×2	0.11910	0.60015	0.97183	0.99954	0.99994	0.99995	0.99995	0.99995
4×4	0.21469	0.77284	0.99186	0.99986	0.99995	0.99995	0.99995	0.99995
8×8	0.35891	0.89521	0.99778	0.99992	0.99995	0.99995	0.99995	0.99995
16×16	0.54046	0.96118	0.99939	0.99994	0.99995	0.99995	0.99995	0.99995
32×32	0.72195	0.98798	0.99981	0.99995	0.99995	0.99995	0.99995	0.99995
64×64	0.86236	0.99664	0.99991	0.99995	0.99995	0.99995	0.99995	0.99995
Fourier	**0.99995**	**0.99995**	**0.99995**	**0.99995**	**0.99995**	**0.99995**	**0.99995**	**0.99995**

Table II compares Fourier analysis of two-group bGS to a numerical implementation for the same problem as in Table I. Again, the results for multi-group bGS are similar to those for one group [1], that is, multi-group bGS is rapidly convergent in problems with optically thick cells, even for the highly scattering model problem, while convergence degrades for problems containing optically thin cells. This is the reverse of SI, which converges more quickly when cells are optically thin and more slowly when cells are thick.

Table II. Theoretical and computed spectral radius of two-group bGS

dx = dy	10^{-1}	10^{0}	10^{+1}	10^{+2}	10^{+3}	10^{+4}	10^{+5}	10^{+6}
1×1	0.02209	0.13220	0.52027	0.90705	0.96422	0.78241	0.22617	0.01563
2×2	0.22834	0.49514	0.83448	0.97496	0.97528	0.84453	0.30419	0.01916
4×4	0.54655	0.77117	0.94952	0.99128	0.97780	0.85753	0.32751	0.02111
8×8	0.78134	0.91477	0.98602	0.99553	0.97889	0.86289	0.33598	0.02178
16×16	0.90729	0.97268	0.99614	0.99660	0.97915	0.86437	0.33831	0.02197
32×32	0.96410	0.99211	0.99879	0.99687	0.97918	0.86477	0.33891	0.02203
64×64	0.98728	0.99785	0.99947	0.99694	0.97927	0.86487	0.33911	0.02204
Fourier	**0.99999**	**0.99997**	**0.99970**	**0.99696**	**0.97932**	**0.86487**	**0.33911**	**0.02204**

3. IMPLEMENTATION OF MULTI-GROUP bGS ON ROADRUNNER

The LANL *Roadrunner* machine has a massively parallel, hybrid computing architecture separated into several layers of communication pathways and processor types. While this idea is not completely new, Roadrunner was the first supercomputer to break the Petaflop "barrier". The smallest logical building block is commonly referred to as a "TriBlade" which consists of an IBM LS21 blade connected to two QS22 blades using a special expansion card for the quad PCIe 8x interconnect; see Fig. 1. Each of the cores in the LS21 two dual-core Opterons is connected to a Cell/B.E. CPU using a PCIe x8 bus. Each of the two QS22 blades has two IBM PowerXCell 8i Cell/B.E CPUs. Each PowerXCell 8i CPU consists of a dual threaded 64-bit PowerPC core (Power Processing Element - PPE) connected to 8 Synergistic Processing Elements (SPE). The SPEs are designed for running compute-intensive applications but depend on the PPE to run the operating system. In turn, the PPE depends on the SPEs to provide the bulk of the compute power.

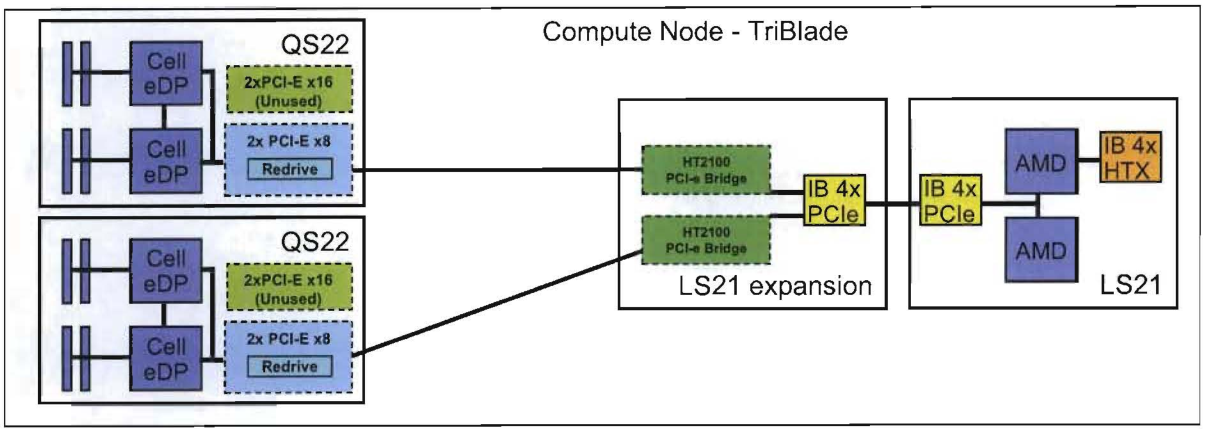


Figure 1. Compute node of Roadrunner hybrid architecture.

Each TriBlade can either be configured as four MPI nodes consisting of one Opteron core connected to a PPE and eight SPEs or two MPI nodes consisting of two Opteron cores connected to a single PPE and sixteen SPEs. Overall each TriBlade has over 400 GFlops double-precision, 8 GB Opteron memory, and 8 GB of Cell/B.E. memory.

The TriBlades can be clustered together using first-level infiniband (IB) switches into connected units (CUs), each of which has up to 180 TriBlades (or up to 720 MPI ranks). Redundant second level IB switches are used to connect the CUs together. Several machine configurations are available at LANL with double-precision computation rates of up to 1 Petaflop.

3.1. Single-Threaded Hybrid Implementation of bGS

The bGS-sweep can be executed on the cores on the Opteron blades, without taking advantage of the Cell/B.E. processors. Numerical experiments shown later illustrate that the bGS algorithm is not viable in this case. This is because the times for the LU factorization and solution on which the algorithm is based increase cubically with the size of the linear system. For multi-group S_n transport, this means the algorithm can only be applied for a very small number of energy groups and S_n quadratures. The Cell/B.E. implementation of the LU decomposition in LAPACK [8], which exploits up to 16 SPEs, scales much better with the size of the linear system. A hybrid implementation of the bGS algorithm restores the viability of the method because of the lower computational cost of the LU factorizations for typical numbers of energy groups and S_n quadrature orders.

The basic bGS-sweep is written in pseudo-code in Fig. 2. For a given mesh cell, the size N is computed, which is the product of the number of S_n angles, the spatial degrees of freedom on the mesh cell (three for DFEM on triangles,) and the number of coupled energy groups. The matrix A and right hand side b are constructed and a LU factorization and solution of matrix A computed. Finally, we store the cell angular fluxes in the global solution vector of angular fluxes before moving to the next cell. The rest of the S_n transport computations take place outside the bGS-sweep code.

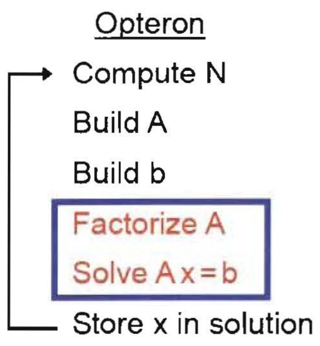


Figure 2. bGS computational kernel: traditional Opteron-only implementation.

The most straightforward way to implement the above procedure on the Roadrunner architecture, and the one which requires the least modification of the original code, is to move the LU factorization, circled in Fig. 2, to the PPE as shown in Fig. 3. In this case, the Opteron still computes N and builds A and b. The Opteron signals the PPE that a linear system is ready to be solved at that point by writing N into the PPE mailbox. The mailbox is a communication mechanism whereby an *unsigned integer* can rapidly be transferred from the Opteron to the PPE. Since A and b can be large, the most efficient way for the Opteron to make them available to the PPE is to "share" memory for A and b with the PPE. This gives the PPE unidirectional access to the data so that the PPE then gets the data for A and b and proceeds with the LU factorization and solution via the Cell/B.E. implementation of the LAPACK routines.

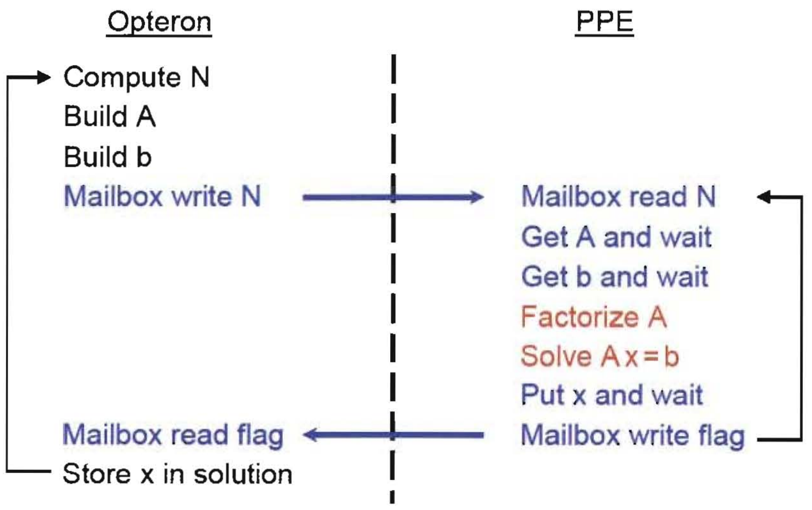


Figure 3. bGS computational kernel: single-threaded hybrid implementation.

The PPE puts the solution into the Opteron memory space and sends a mailbox signal to the Opteron indicating that the solution for this computational cell is available for retrieval, at which point it is stored in the global solution vector of angular fluxes.

3.2. Dual-Threaded Hybrid Implementation of bGS

The single-threaded hybrid implementation discussed in Sec. 3.1 is straightforward but it has a drawback, namely the Opteron is idle while the PPE is working, and *vice versa.* Computational efficiency of the bGS-sweep is improved by overlapping communication and computation using multithreading techniques [9], as shown in Fig. 4.

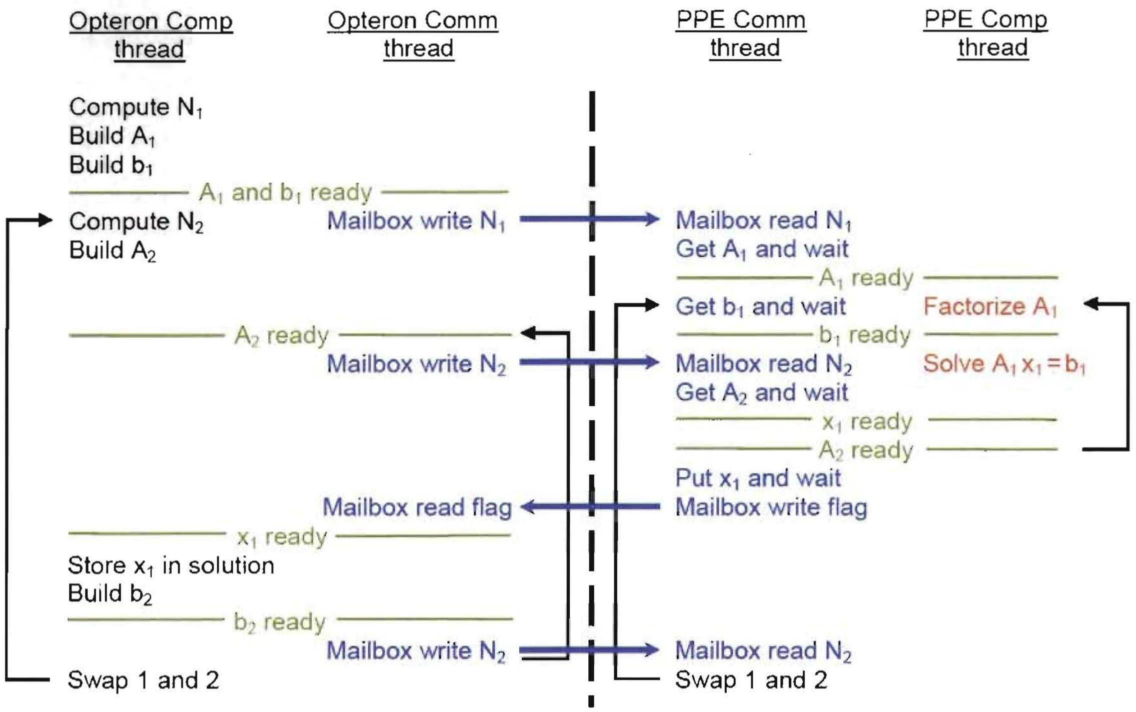


Figure 4. bGS computational kernel: dual-threaded hybrid implementation.

Communication (Comm) and computation (Comp) threads take place simultaneously on both the Opteron and the PPE, using two buffers denoted with the 1 and 2 subscripts in Fig 4. A Comp thread on the Opteron starts by computing N_1 and building A_1 and b_1. The Opteron Comm thread then writes N_1 in the PPE mailbox, but the Comp thread is not idle; it starts computing N_2 and building the next matrix A_2. Concurrently, the PPE Comm thread receives N_1 and is ready to get A_1. As soon as A_1 is received, the PPE Comp thread performs a LU factorization. Once b_1 is available, it computes the solution x_1. Meanwhile the PPE Comm thread receives N_2 and then A_2, and so on. Since b_2 depends on x_1 for bGS, the Opteron builds b_2 after storing x_1 and re-uses N_2 to flag the PPE that b_2 is ready. Overlapping of the threads takes place by swapping the buffers, illustrated by the loops in Fig. 4. Race conditions between threads are avoided using semaphores indicated by the "ready" labeled horizontal lines spanning the Comp and Comm threads. Appropriate conditions (not shown) ensure that the loops are exited once a bGS-sweep is completed.

3.3. Speedup of the Hybrid Implementation of bGS

To validate the speedup provided by the single- and dual-threaded hybrid implementations of bGS, we used a 8×8 grid of squares, each of which is $10^{+4}cm$ on a side, and was further subdivided into four triangular cells for every square - a total of 256 mesh cells. Scattering ratio $c = 0.99$, $\sigma = 1cm^{-1}$, a uniform fixed source of $1m^{-3}s^{-1}$ and vacuum boundary conditions were specified. Square Chebyshev-Legendre (SCL) quadratures vary from order $n = 2$ to $n = 40$, such that the linear systems sizes vary as $N = 3n^2$.

Figure 5 shows the execution times for the GMRES(20) solution, to a relative convergence tolerance of 10^{-5} using different numbers of processors, Np = 4, 8, 16, for the same problem. The Opteron-only implementation of bGS is labeled GSOP, while GSST and GSDT refer to the single- and dual-threaded hybrid implementations of bGS, respectively, using 16 SPEs for every process.

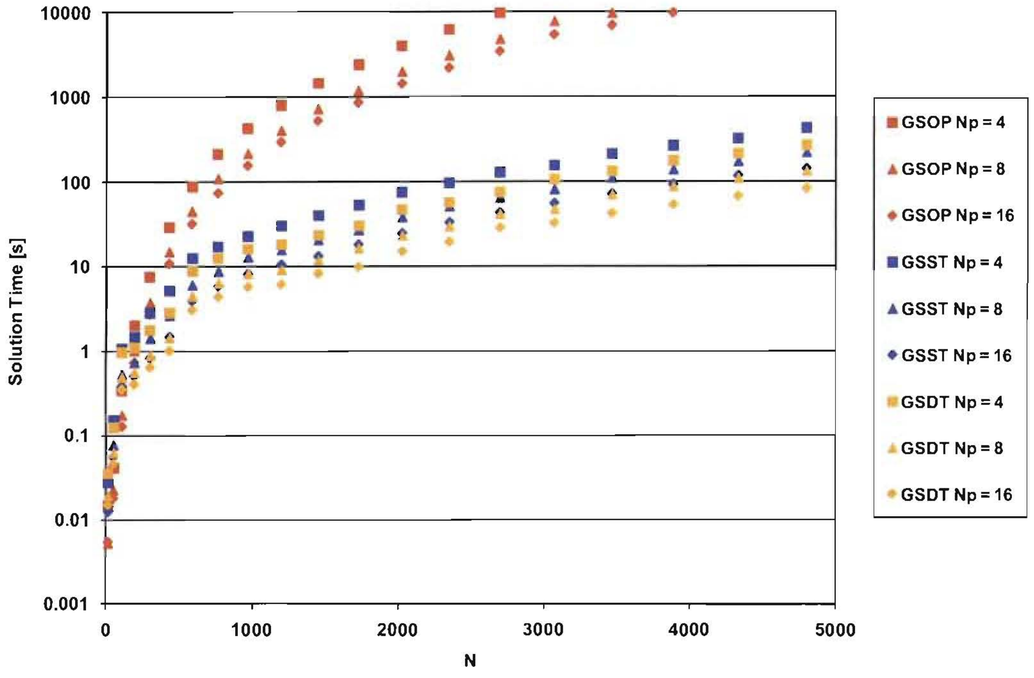
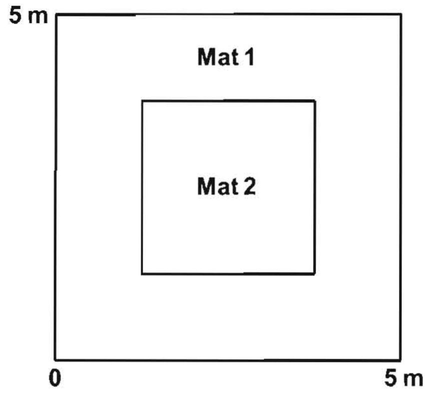


Figure 5. Speedup of bGS on the hybrid architecture.

The dual-threaded, double-buffered version of bGS is almost twice as fast as single-threaded version because the communication and computation threads are completely overlapped. When N is less than about 300, the hybrid implementations are slower than the Opteron-only implementation, because of the overhead associated with moving data between the Opteron and the PPE. When the linear system size increases, the LU decomposition time on the Opteron-only implementation begins to dominate, while the hybrid solution times increase at a much slower rate, such that the hybrid implementations are up 100 times faster.

4. NUMERICAL RESULTS

We now compare our bGS-sweep implementation to FPS for a more complex problem which comprises two square regions, an outer region $5m$ wide surrounding an inner region $2.5m$ wide. The outer region contains material 1 and the inner region material 2. Figure 6 shows the configurations that we used to construct four problems of increasing material heterogeneity. Because bGS performs well in optically thick problems, Mat 1 is chosen to be optically thick and characterized by a scattering ratio of 0.9999. The properties of Mat 2 are then varied in such a way that the convergence rate of GMRES for the FPS implementation is adversely affected. In the figures and tables that follow, the label Hom stands for a homogeneous problem while the label Het identifies the heterogeneous problems. The first digit in the homogeneous problems is used to denote the exponent in the total cross-section, while the first and second digits of the heterogeneous problems indicates the exponents in the cross-sections for the heterogeneous problems in Mat 1 and Mat 2, respectively.



- Mat 1 (g = 1): $\sigma_1 = 1.0001e{+}6\ m^{-1}$
  $\sigma_{S1} = 1.e{+}6\ m^{-1}$
- Mat 2 (g = 1):
  - Hom66: $\sigma_2 = \sigma_1$
    $\sigma_{S2} = \sigma_{S1}$
  - Het66: $\sigma_2 = 1.e{+}6\ m^{-1}$
    $\sigma_{S2} = 0.9999e{+}6\ m^{-1}$
  - Het65: $\sigma_2 = 1.e{+}5\ m^{-1}$
    $\sigma_{S2} = 0.999e{+}5\ m^{-1}$
  - Het64: $\sigma_2 = 1.e{+}4\ m^{-1}$
    $\sigma_{S2} = 0.99e{+}4\ m^{-1}$

Figure 6. Two-region configuration.

As long as the inner region is not too optically thin, we expect that the bGS convergence will be fairly insensitive to material discontinuities. The spatially discretized problem is solved on an unstructured triangular mesh, comprising 10454 mesh cells and we vary the number of energy groups used in each of the four problems: 1, 2, 10, and 20. We use an S_4 level symmetric angular quadrature, which has 12 discrete ordinates. Thus, the dimension of linear systems increases with increasing number of groups. For the multi-group extensions of these four problems we construct the scattering and total cross-sections for a material as described in Sec. 2.2 in such a way as to ensure that the eigenvalue of maximum magnitude for the multi-group SI iteration is equal to the scattering ratio selected for the material in the one-group cases. We assume vacuum boundary conditions and a fixed uniform source of strength $1cm^{-3}s^{-1}$, throughout both material regions. Solutions are computed using GMRES(50) to a relative convergence tolerance of 10^{-5}.

4.1. Strong Scaling of bGS and FPS

Strong scaling is measured by increasing the number of processors employed in the parallel computation, from 32 to 512 in powers of 2. The results of the strong scaling are reported in Fig. 7, showing the solution times and number of iterations versus the number of processors, plotted separately for each of the various number of energy groups. Solid markers connected with solid lines refer to the dual-threaded implementation of bGS (GSDT), and non-filled markers with dashed lines refer to the (Opteron-only) implementation of FPS (PSWP).

For $g = 1$, solution times with the bGS implementation are fairly constant for the four types of problems, while the FPS implementation solution times increase with increasing material heterogeneity. For a given problem and number of groups, the number of iterations for FPS is constant with respect to Np. The slight increase in the number of iterations for bGS, as Np is increased, is a consequence of the effect of processor interfaces on the spectral properties of bGS [1] because the terms involving L_b^* on the parallel-decomposed mesh boundaries are always lagged. The bGS implementation also displays better scaling properties than the FPS implementation, for which the dependencies between processors result in a sweep schedule that scales poorly as Np becomes large [4,5]. This becomes evident on more than 256 processors.

Similar results are observed for the $g = 2$ and $g = 10$ calculations. For $g = 20$, the bGS implementation slows down relative to the FPS implementation due to the increasing size of the linear systems for a larger numbers of groups. Even though the execution times are slower, the scaling of the bGS implementation is better than the FPS implementation because bGS requires fewer iterations compared to FPS. Even for a homogeneous configuration, convergence of the multi-group problems slows as a consequence of the eigenvalue distribution arising out of the randomly generated multi-group scattering operator; this effect is more pronounced for the FPS implementation than for the bGS implementation.

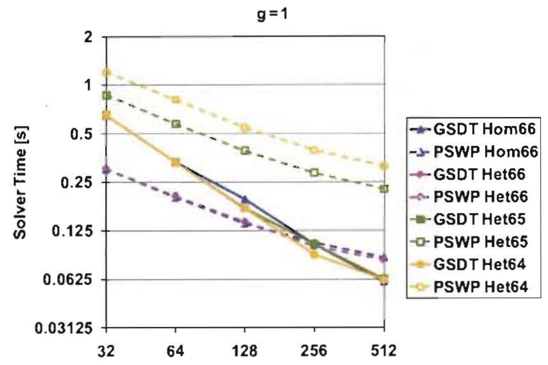
4.2. Weak Scaling of bGS and FPS

We compared weak scaling by considering the strong scaling unstructured triangular mesh, comprising 10454 cells, that was used for the initial number of processors, Np = 32. We generated a sequence of meshes such that the average number of cells per processor is roughly the same as for the initial mesh, that is, 326 mesh cells per processor. In other words, as the number of processors is increased for the fixed-sized problem domain, the mesh is increasingly refined and the spatial discretization becomes increasingly resolved.

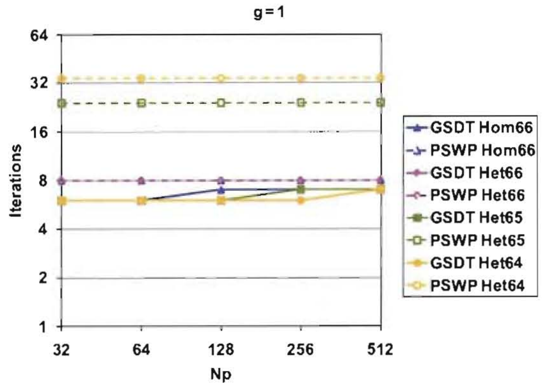
Results in Fig. 8 show that the bGS implementation on the hybrid architecture displays better weak scaling than the FPS implementation and can be competitive with FPS, once the number of processors becomes large enough. This is again due to scheduling dependencies among parallel sub-domains associated with the FPS implementation and to the resilience of the GMRES solution using the bGS splitting in the presence of material discontinuities. The increase in GMRES iterations for bGS is due to the triangles becoming progressively optically thinner as the meshes are refined and the fact that bGS spectral properties degrade as the cells become thinner.

5. CONCLUSIONS

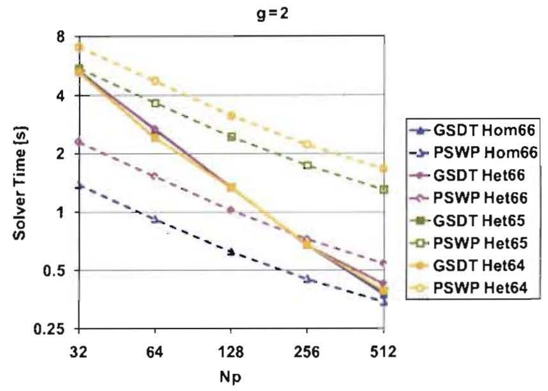
We have implemented a multi-group dual-threaded, double-buffered version of the bGS-sweep on the Roadrunner hybrid computer architecture. When combined with the Cell/B.E. implementation of a LU decomposition and solution, we have shown that the bGS splitting is a viable solution algorithm, when solved with GMRES on the Opteron level of this architecture.



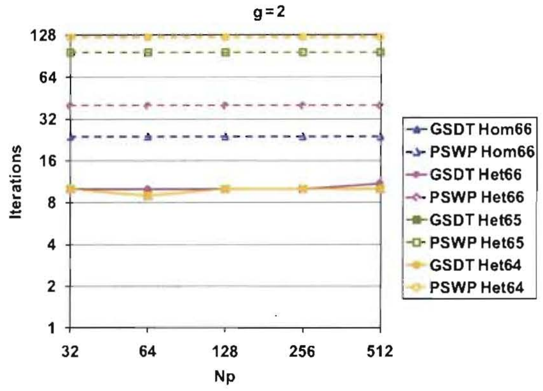
(a)



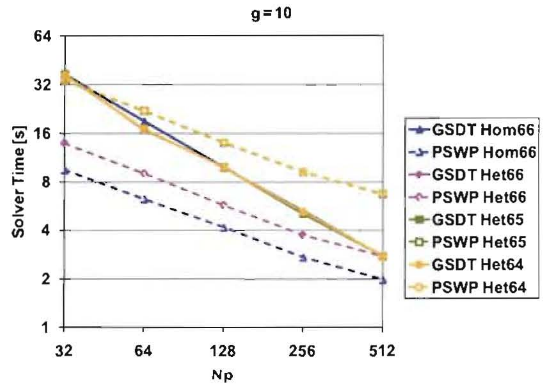
(e)



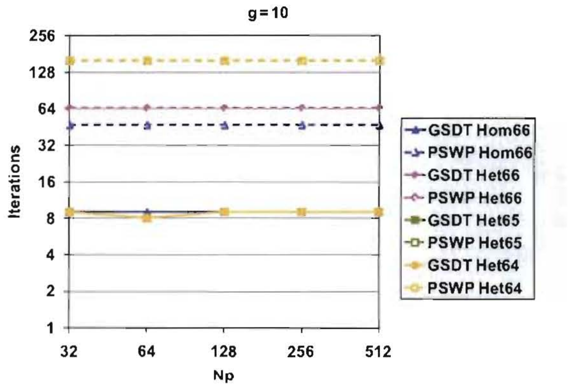
(b)



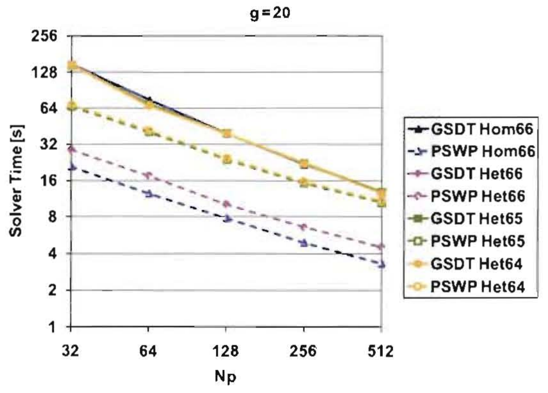
(f)



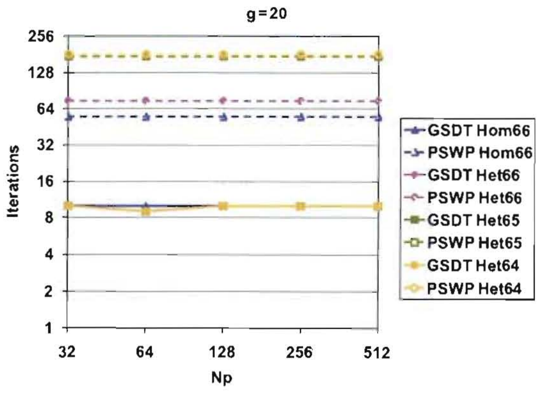
(c)



(g)

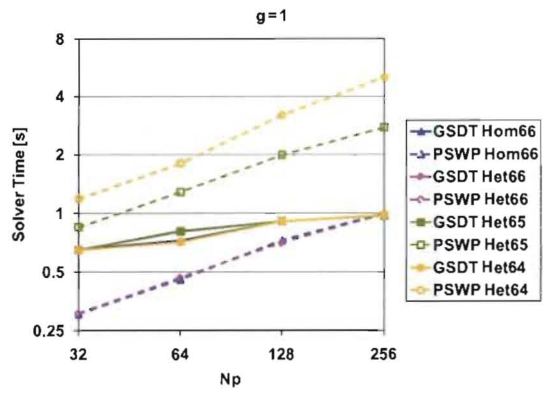


(d)

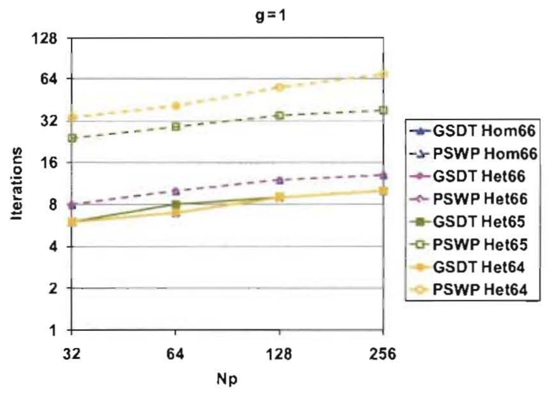


(h)

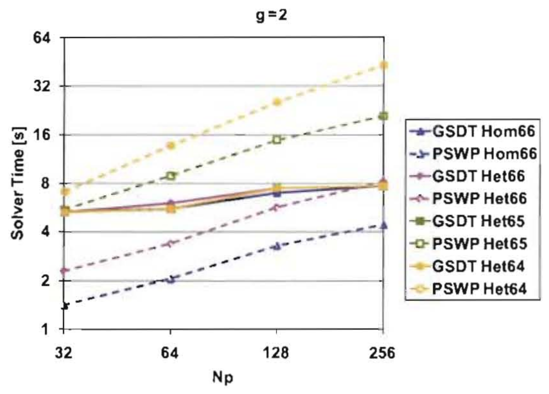
Figure 7. Strong scaling of bGS compared to FPS.



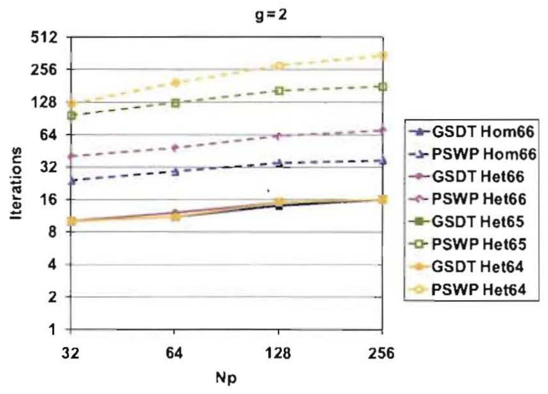
(a)



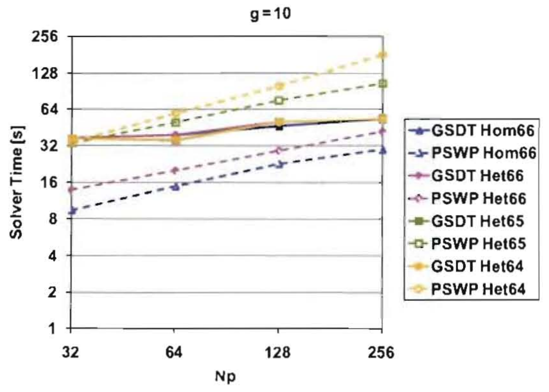
(e)



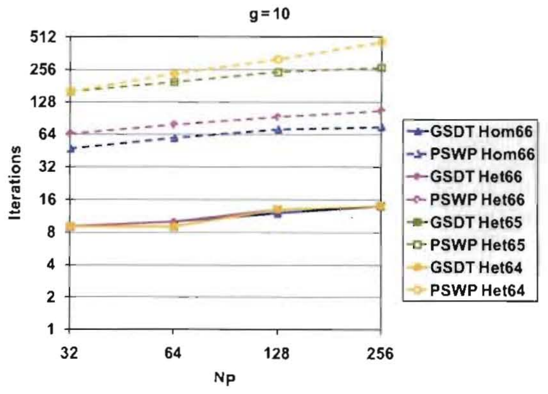
(b)



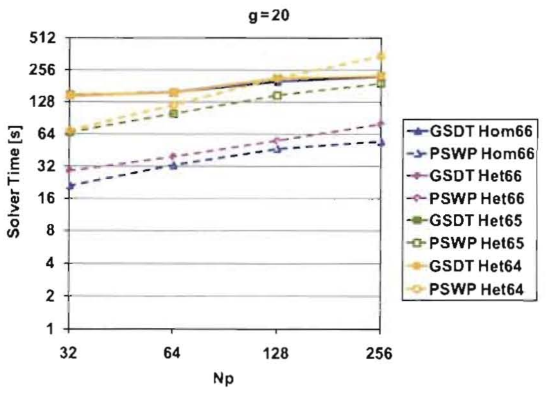
(f)



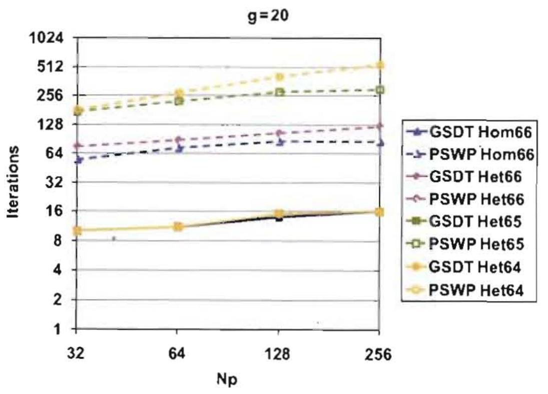
(c)



(g)



(d)



(h)

Figure 8. Weak scaling of bGS compared to FPS.

We observed that the bGS implementation outperformed the traditional FPS implementation on large numbers of processors by virtue of superior strong and weak scaling. While bGS holds the potential of being an efficient algorithm for massively parallel transport calculations in the model problems we devised for our numerical experiments, further refinements to the algorithm involving energy-splitting strategies will be investigated in order to solve problems for applications with realistic data and large numbers of energy groups.

The speed-up and scalings we observed, while impressive, are not sufficient to make bGS competitive with the FPS algorithms under all circumstances. Other hybrid architectures using even faster accelerators, such as those based on General Purpose Graphics Processing Units (GPGPUs) or Field-Programmable Gate Arrays (FPGAs,) hold the potential to further speed-up a LU factorization on which the bGS implementation is based. The overlapped dual-threaded approach to the bGS-sweep could be applied to other hybrid architectures.

ACKNOWLEDGMENTS

This information has been authored by employees of the Los Alamos National Security, LLC (LANS) operator of Los Alamos National Laboratory under contract No DE-AC52-06NA25396 with the U.S. Department of Energy.

[1] Cell Broadband Engine^{TM} and Cell/B.E. are trademarks of Sony Computer Entertainment, Inc.

REFERENCES

1. M. Rosa, J. S. Warsa, T. M. Kelley, "Fourier Analysis of Cell-Wise Block-Jacobi Splitting in Two-Dimensional Geometry," *Proceedings of the International Conference on Mathematics, Computational Methods & Reactor Physics (M&C 2009)*, Saratoga Springs, NY, May 3-7, 2009, on CD-ROM, ANS, LaGrange Park, IL (2009).
2. D. Grice et al., "Breaking the Petaflops Barrier," *IBM Journal of Research and Development*, **53**, no. 5, pp. 1:1-1:16 (2009)
3. C. H. Crawford, P. Henning, M. Kistler, C. Wright, "Accelerating Computing with the Cell Broadband Engine Processor," *Proceedings of the 2008 Conference on Computing Frontiers*, Ischia, Italy, May 5-7, 2008, pp. 3-11 (2008).
4. S. D. Pautz, "An Algorithm for Parallel S_n Sweeps on Unstructured Meshes," *Nucl. Sci. Eng.*, **140**, pp. 111-136 (2002)
5. S. J. Plimpton, B. Hendrickson, S. P. Burns, W. Mclendon III, L. Rauchwerger, "Parallel S_n Sweeps on Unstructured Grids: Algorithms for Prioritization, Grid Partitioning, and Cycle Detection," *Nucl. Sci. Eng.*, **150**, pp. 267-283 (2005)
6. J. S. Warsa, "A Continuous Finite Element-Based, Discontinuous Finite Element Method for S_N Transport," *Nucl. Sci. Eng.*, **160**, pp. 385-400 (2008)
7. Y. Saad, *Iterative Methods for Sparse Linear Systems*, PWS Publishing Company, Boston (1996)
8. E. Anderson et al., *LAPACK User's Guide*, SIAM (1999)
9. B. Lewis, D. J. Berg, *Multithreaded programming with pthreads,* Sun Microsystems Press, (1998)