ISSN 0333-3590

A New Algorithm for the SLALOM Benchmark[1]

Petter E. Bjørstad   Erik Boman

REPORT NO 55   MAY 1991

Abstract

We apply a preconditioned conjugate gradient algorithm to the SLALOM benchmark, proving that the solution phase of the benchmark is reduced to $O(N^2)$. The algorithm therefore performs well over the full range of input parameters specified by the benchmark. This has a dramatic impact on the benchmark for all computers. We illustrate this with new implementations on an 8192 processor MP-1 system and a Cray X/MP.

Keywords: Computer benchmarks, Conjugate gradients, Radiosity.

[1]Institutt for Informatikk, University of Bergen, Høyteknologisenteret, N-5020 Bergen, Norway. This work was supported in part, by NAVF, grant 413.89/024 and in part by NTNF, contract IT0228.20484

1 Description of the problem

Recently, the SLALOM benchmark [5, 6] has received considerable interest [7], as a way of providing a scalable (super) computer benchmark. It is a fixed time benchmark, i.e. the challenge is to solve the largest possible problem in one minute elapsed time.

The SLALOM benchmark is based on solving a radiosity problem. Radiosity has over the last few years become a popular technique in computer graphics to generate realistic images. It is a global illumination model for environments with diffuse surfaces [4].

The specific SLALOM problem is to find the equilibrium radiation inside a box made of diffuse colored surfaces. The faces are divided into smaller regions (patches). The global coupling matrix for the patches is set up and the radiosity equations are solved for red, green, and blue spectral components, respectively.

Reading the geometry and the other parameters describing the box (reflectivity and emissivity) and storing the final results to disk are also parts of the benchmark. However, the computationally most expensive parts are the setup phase which requires $O(N^2)$ flops and the solver which requires $O(N^3)$ flops (using a direct solver). Here N denotes the number of patches.

2 The proposed algorithm

An important part of the SLALOM philosophy is that only the problem to be solved is specified. There are no restrictions on which algorithms that may be employed. Any method is accepted as long as it produces correct answers within a certain accuracy for all input data within a given range.

Since for large problems, the most time consuming part of the benchmark consists of solving a nearly dense symmetric, positive definite system of equations, we decided to investigate the possible benefit of replacing the Cholesky solver with an iterative solver. A logical approach was to use the conjugate gradient algorithm [3]. Knowing that the radiosity matrix often will have significantly larger diagonal entries than off-diagonal entries, we chose to use the diagonal as a preconditioning matrix. This was also particularly simple to implement, and requires very few extra operations.

In each CG iteration we do a matrix-vector multiply and several vector-vector operations. The most expensive step is the matrix-vector multiply, requiring $2N^2$ flops. If the maximum number of iterations required for convergence remains constant for large values of N, we have an $O(N^2)$ algorithm.

Our numerical experiments suggest that this is indeed the case. We have used the convergence test $\|r\|_1/(\max \mid S_{ij} \mid \|x\|_\infty) \leq 0.5 \times 10^{-8}$, where S is the matrix for the SLALOM equations, x is the current solution and r is the residual. This is consistent with the residual test used for verification in the SLALOM benchmark codes. Note that this a stricter condition than the criterion stated in [5], where the infinity norm is applied to the residual also. It is not clear which test is the official one, but is seems reasonable to use the same norm on all vectors.

A number of refinements to the approach described here can be implemented. A near optimal implementation of more sophisticated preconditioners as well as the basic iterative algorithm, becomes more dependent on the computer architecture than is the case for direct solvers. A number of these issues will be discussed in a forthcoming report.

3 A proof of optimal convergence properties

It is well known that the numbers of iterations required for the CG algorithm to converge to any specified accuracy is proportional to the square root of the condition number of the iteration matrix [3]. In order to find an upper bound for the condition number, we must have some knowledge of the radiosity equations. Starting with the equations as they appeared in the original paper on radiosity by Goral et. al. [4], we have:

$$b_i = e_i + \rho_i \sum_{j=1}^{N} b_j F_{ij}, \qquad i = 1, \ldots, N \tag{1}$$

where

a_i = the area of surface i,
b_i = the radiosity of surface i
(the total rate of energy leaving the surface),
e_i = the rate of energy (light) emitted from surface i,
ρ_i = the reflectivity of surface i
(the fraction of incident light which is reflected back into the environment),
F_{ji} = the form-factor from j to i
(the fraction of energy leaving surface j which lands on surface i).

In matrix notation,

$$Ab = e, \qquad A = I - \mathrm{diag}(\rho) F \tag{2}$$

The form factors corresponding to any surface sum to unity, $\sum_{j=1}^{N} F_{ij} = 1$. Since we have non-concave patches, a patch cannot reflect light upon itself so $F_{ii} = 0$ for all i.

By applying Gerschgorin's circle theorem [3] we get that the eigenvalue corresponding to the i'th row of A lies within a disk with radius $\rho_i \sum_{j \neq i} F_{ij} = \rho_i$ and center at 1. It follows that

$$1 - \rho_{max} \leq \lambda(A) \leq 1 + \rho_{max}. \tag{3}$$

The reflectivities ρ_i always take values between 0 and 1. In the SLALOM benchmark, it is given that $0.001 \leq \rho_i \leq 0.999$ for all i. Hence, any real eigenvalue must lie between 0.001 and 1.999.

Now, suppose that we apply the conjugate gradient method on the SLALOM equations. We solve the symmetric system $Sb = De$, where $S = DA$ and $D = \text{diag}(a)\text{diag}(\rho)^{-1}$. Using the diagonal as a preconditioner, the iteration matrix is $\tilde{A} = D^{1/2} A D^{-1/2}$. $\tilde{A}$ and A are similar and therefore have the same eigenvalues. Thus, the condition number

$$\mathcal{K}(\tilde{A}) = \frac{\lambda_{max}(A)}{\lambda_{min(A)}} \leq \frac{1.999}{0.001} = 1999. \tag{4}$$

This upper bound depends only on the reflection coefficients and not on the number of patches. So for any fixed set of input parameters, the condition number is bounded by a constant. We will now show that the SLALOM accuracy request stated in [5] is, up to a constant, equivalent to measuring the error in $\tilde{A}$-norm. Then we can conclude that the number of CG iterations will also be bounded by a constant (independent of N).

$$\frac{\|r\|_\infty}{\max \mid S_{ij} \mid \|x\|_\infty} \leq \frac{\|S\|_\infty \|e\|_\infty}{\max \mid S_{ij} \mid \|x\|_\infty} \leq \frac{2\|e\|_\infty}{\|x\|_\infty} \leq \frac{2\lambda_{min}^{-1/2} \|e\|_{\tilde{A}}}{\|x\|_\infty} \leq c\|e\|_{\tilde{A}} \tag{5}$$

Here x, r, e denote the current solution, residual and error respectively and c is a constant. We have used the facts that

$$\|S\|_\infty \leq 2 \max \mid S_{ij} \mid$$

because S is diagonally dominant, and

$$\|e\|_\infty \leq \|e\|_2 \leq \lambda_{min}^{-1/2} \|e\|_{\tilde{A}}$$

because $\lambda_{min} \leq \frac{e^T \tilde{A} e}{e^T e}$ for any vector $e \not\equiv 0$.

Thus, the usual convergence theory for CG applies and we have an $O(N^2)$ algorithm which is obviously of optimal order. For large N, the CG algorithm will always perform better than an $O(N^3)$ direct solver.

4 The parallel implementation

The SLALOM group at Ames Laboratory has implemented the SLALOM benchmark on the MasPar MP-1 family of SIMD computers. The code is written in MPL (parallel C) and is heavily optimized. A number of clever tricks are used, including doing output formatting in parallel and overlapping the solver routine on the DPU with disk I/O on the front-end.

We have used this code, only replacing the solver. With $O(N)$ processors, the vector operations on the MP-1 can each be done in $O(1)$ time. On our 8K processor MP-1 we will in fact have more processors than vector elements. We do the matrix-vector product by distributing the vector elements over the processor grid, multiplying with the matrix elements, and summing up in $O(\log N)$ steps.

5 Machine configuration

The MasPar MP-1 system is a massively parallel SIMD computer system. The system consists of a high performance UNIX workstation (FE) and a data parallel unit (DPU). The DPU consist of at least 1024 processor elements (PEs) each with 16Kb of memory and 192 bytes of register space. The experiments to be reported here, are all performed on a 8192 processor system. We define the average time of a floating point instruction, $\alpha = \frac{1}{2}(Mult + Add)$. Measured in units of α, the floating point performance of the MP-1, corresponds to a peak speed of 0.0355 Mflops in 64 bit arithmetic per processor, or 290 Mflops for a machine having 8192 processors. On the MasPar MP-1, a floating point multiplication takes approximately three times the time for a corresponding floating point addition, in 64 bit precision. A more detailed general description of the MasPar MP-1 computer can be found in [1, 2, 8].

6 Results

We have carefully timed the implementation. All MasPar times are wall clock elapsed times.

The SLALOM benchmark supports a set of input data. We refer to this as (the) *Standard* (case) in Table 1 and Table 2. For our iterative solver, we have tried to find a difficult case, that is a problem with a very unfavorable distribution of eigenvalues as well as a right hand side that depends on this distribution. We believe that the worst case input data is a highly eccentric

patches	Cholesky		CG *Standard*		CG *Difficult*	
	Time (s)	%solver	Time (s)	%solver	Time (s)	%solver
600	7.170	63.7	5.670	56.3	6.570	54.2
1200	24.080	73.3	12.330	49.1	17.080	61.4
2400	112.150	82.2	32.240	36.9	49.320	60.4

Table 1: MasPar MP-1208

patches	Cholesky		CG *Standard*		CG *Difficult*	
	Time (s)	%solver	Time (s)	%solver	Time (s)	%solver
600	7.235	39.1	4.629	2.9	4.922	24.9
1200	37.264	53.6	18.238	3.0	20.335	30.2
2400	209.338	66.9	71.237	3.4	81.296	30.6

Table 2: Cray X-MP

box with maximal reflectivity. Results for the most difficult input parameters we could find are given in the last column of the table and referred to as (the) *Difficult* (case).

In Table 1, we show the performance of our iterative solver as a function of the number of patches. We list the total time for the benchmark and the percentage of the time spent in the solver. Notice how the percentage increases for the Cholesky, decreases for the *Standard* case, and stays about constant for the *Difficult* case. In Table 2 we give similar figures from the 'public domain' Cray code that we have been able to obtain. Note that the numbers for Cray are based on CPU time, which can only serve as an upper bound (on the number of patches), since we were unable to run the Cray stand alone. We observe a similar behavior and note that the differences between 'easy' and more 'difficult' cases can be quite significant.

In the last 4 tables, we list the standard output that SLALOM produces for the cases that just run within the one minute limit. We are interested in the relative performance, as substantial tuning could be applied to all versions. The benchmark improves from 1807 to 2688 patches for the MP-1 and from 1440 to 2048 patches for the Cray X-MP. The iterative case is based on the most difficult case we have identified. As a curiosity, this result moves the MP-1 up ahead of the figure reported for a one processor Cray Y-MP in [7] and gives the solver a 'Mflop number' equal to 539. Of course, the replacing of the Cholesky solver by an iterative solver will have a similar positive effect on all computers that can run a sufficiently large problem.

Task	Seconds	Operations	MFLOPS	% of Time
Reader	0.170	258	0.001518	0.3 %
Region	0.090	2809	0.031211	0.2 %
SetUp1	3.020	151041190	50.013639	5.0 %
SetUp2	7.830	393990456	50.318066	13.1 %
SetUp3	0.090	2713759	30.152907	0.2 %
Solver	47.220	5303582037	112.316434	78.8 %
Storer	1.520	86736	0.057063	2.5 %
TOTALS	59.940	5851417245	97.621242	100.0 %

Table 3: MasPar MP-1208 using Cholesky factorization on 1807 patches

Task	Seconds	Operations	MFLOPS	% of Time
Reader	0.160	258	0.001612	0.3 %
Region	0.120	3994	0.033283	0.2 %
SetUp1	15.320	924329790	60.334843	26.2 %
SetUp2	5.040	50988264	10.116719	8.6 %
SetUp3	0.220	3774684	17.157652	0.4 %
Solver	36.120	19474668906	539.165806	61.7 %
Storer	1.520	129024	0.084884	2.6 %
TOTALS	58.500	20453894920	349.639229	100.0 %

Table 4: MasPar MP-1208 using an iterative solver on a difficult case having 2688 patches.

Task	Seconds	Operations	MFLOPS	% of Time
Reader	0.001	258.	0.244625	0.0 %
Region	0.000	2346.	6.746395	0.0 %
SetUp1	6.865	151408323.	22.054343	11.9 %
SetUp2	17.725	290810720.	16.406962	30.7 %
SetUp3	0.028	1726136.	61.225475	0.0 %
Solver	32.687	2687379255.	82.215399	56.7 %
Storer	0.361	69120.	0.191232	0.6 %
TOTAL	57.668	3131396158.	54.300252	100.0 %

Table 5: CRAY X-MP using Cholesky factorization on 1440 patches

Task	Seconds	Operations	MFLOPS	% of Time
Reader	0.001	258.	0.272971	0.0 %
Region	0.000	3226.	9.517925	0.0 %
SetUp1	36.966	846901443.	22.910034	63.3 %
SetUp2	2.146	34709600.	16.174796	3.7 %
SetUp3	0.054	2196376.	41.033244	0.1 %
Solver	18.721	8620847115.	460.486630	32.0 %
Storer	0.529	98304.	0.185954	0.9 %
TOTAL	58.417	9504756322.	162.705508	100.0 %

Table 6: CRAY X-MP using an iterative solver on a difficult case having 2048 patches.

7 Conclusions

We have shown how to solve the SLALOM benchmark by an iterative method. This approach eliminates the $O(N^3)$ operations from the benchmark and therefore completely changes the nature of the test. We have further demonstrated that the MP-1 is a very cost effective computer, performing significantly better than a Cray X-MP in the conjugate gradient iteration. As a 'side effect' we have (very temporarily!) 'moved' the MasPar MP-1 computer having 8192 processors, up ahead of the Cray Y-MP on the benchmark ranking.

References

[1] T. Blank, *The MasPar MP-1 architecture*, in Proceedings of IEEE Compcon Spring 1990, IEEE, February 1990.

[2] P. Christy, *Software to support massively parallel computing on the MasPar MP-1*, in Proceedings of IEEE Compcon Spring 1990, IEEE, February 1990.

[3] G. H. Golub and C. F.Van Loan, *Matrix Computations*, Johns Hopkins Univ. Press, 1989. Second Edition.

[4] C. M. Goral, K. E. Torrance, D. P. Greenberg, and B. Battile, *Modeling the interaction of light between diffuse surfaces*, Computer Graphics, 18 (1984), pp. 213–222. (In Proceedings of SIGGRAPH 84).

[5] J. Gustafson, D. Rover, S. Elbert, and M. Carter, *The design of a scalable, fixed-time computer benchmark*, Tech. Report IS-5049/UC-32, Ames Laboratory, Iowa State University, 1990.

[6] ——, *SLALOM: The first scalable supercomputer benchmark*, Supercomputing Review, (1990), pp. 56–61.

[7] ——, *SLALOM*, Parallellogram, (1991), pp. 10–13.

[8] J. Nickolls, *The design of the MasPar MP-1, a cost effective massively parallel computer*, in Proceedings of IEEE Compcon Spring 1990, IEEE, February 1990.