A Roofline Performance Analysis of an Algebraic Multigrid PDE Solver

Alex Druinsky, Brian Austin, Xiaoye S. Li, Osni Marques, Eric Roman, Samuel Williams
Lawrence Berkeley National Laboratory, Berkeley, California 94720, USA

***Abstract*—We present a performance analysis of a novel element-based algebraic multigrid (AMGe) method combined with a robust coarse-grid solution technique based on HSS low-rank sparse factorization. Our test datasets come from the SPE Comparative Solution Project for oil reservoir simulations. The current performance study focuses on one multicore node and on bound analysis using the roofline technique. We found that keeping a small value of spectral tolerance is most critical to achieve the best AMG solver performance. Our roofline bound estimate is within 23% accuracy compared to the actual runtime on a Cray XC30 12-cores processor.**

I. The SA-ρAMGe algorithm

The *Smoothed Aggregation Spectral Element Agglomeration AMG* method was originally introduced by Brezina and Vassilevski [1] and implemented by Kalchev [2], [3]. It is a two-level algorithm that forms the prolongation operator P by first computing the eigenvectors of the low-end eigenvalues of the local stiffness matrices associated with the agglomerates and then multiplying by a matrix polynomial smoother (see [1] for the details of this step). The coarse-grid matrix is computed as $A_c = P^T AP$, where A is the finite-element stiffness matrix.

The AMG solution cycle proceeds by alternating between the two levels. Specifically, cycle k transforms the current iterate x_k into x_{k+1} using the following steps:

1) Pre-smoothing: $y_k \leftarrow x_k + M^{-1}(b - Ax_k)$
2) Restriction: $r_c \leftarrow P^T(b - Ay_k)$
3) Coarse solution: $x_c \leftarrow A_c^{-1} r_c$
4) Interpolation: $z_k \leftarrow y_k + Px_c$
5) Post-smoothing: $x_{k+1} \leftarrow z_k + M^{-1}(b - Az_k)$

In the above, the matrix M corresponds to a polynomial smoother such that $M^{-1} = (I - p_\nu(D^{-1}A))A^{-1}$, where $p_\nu(t)$ is a polynomial and D is a diagonal matrix that can be efficiently computed from A. The polynomial is of degree $3\nu+1$, where ν is a user-specified parameter, and has the form

$$p_\nu(t) = \left(1 - \frac{t}{\tau_1}\right)\left(1 - \frac{t}{\tau_2}\right)\cdots\left(1 - \frac{t}{\tau_{3\nu+1}}\right) ,$$

where $\tau_1, \tau_2, \ldots \tau_{3\nu+1}$ are the roots of $p_\nu(t)$, defined using an easily-evaluated formula.

Although the definition of M^{-1} involves the inverse of A, the matrix M^{-1} can be applied to any vector w using a straightforward procedure. We compute $M^{-1}w$ by evaluating the sequence

$$w^{(k)} \leftarrow w^{(k-1)} + \frac{1}{\tau_k} D^{-1}(b - Aw^{(k-1)})$$

for $k = 1, 2, \ldots, 3\nu+1$, starting with $w^{(0)} = w$ and ultimately yielding $w^{(3\nu+1)} = M^{-1}w$.

A. Coarse-grid solvers

Accuracy in the coarse step of the algorithm $x_c \leftarrow A_c^{-1} r_c$ is crucial to ensure the convergence of the AMG cycles. Here we consider two solvers: the conjugate gradients algorithm, preconditioned with a single step of the Jacobi or Gauss-Seidel methods (PCG/J and PCG/GS for short); or the FGMRES iterative solver [4], preconditioned with a sparse matrix factorization algorithm called StruMF [5], that was recently developed by Napov and Li (denoted FGMRES/StruMF).

StruMF is a multifrontal, nested-dissection based sparse factorization algorithm. The novelty is to represent the dense frontal matrix corresponding to a node of the separator tree as a hierarchically-semiseparable (HSS) matrix [6]. The HSS structure exploits the *data-sparseness* of the dense matrix using SVD-like compression. Furthermore, the *hierarchical partitioning* of the matrix blocks and the use of *nested bases* lead to factorization and solve algorithms that are asymptotically faster than the classical ones and use less memory. The solve procedure in StruMF is more expensive than a PCG iteration, but it is more reliable for numerically challenging PDEs.

II. Evaluation with the SPE10 Benchmark

We evaluate our solver using SPE10, a large-scale geophysics benchmark model problem [7]. The fine-grid matrix has dimension about 1M and approximately 30.5M nonzero elements. The average number of nonzeros per row is 26.4. The matrix has two variants: isotropic and anisotropic; its sparsity structure is the same in both cases.

We solve the problem using one socket (12 cores) of a Cray XC30 whose nodes are equipped with Intel Xeon E5-2695 v2 CPUs and 64 GB of memory each.

We explored the parameter space of the solver and examined the impact on performance. The parameters are: the number of elements per agglomerate; the parameter ν that determines the degree of the polynomial smoother; the spectral tolerance θ, which determines how many eigenvectors of each local stiffness matrix are represented in the prolongation operator. In addition to this, we also compared the performance of the two aforementioned coarse-grid solvers.

The results of the isotropic problem are shown in Table I. The rows *hierarchy* and *cycles* show the time in seconds required by the AMG algorithm to form the coarse-grid matrix and the number of cycles required for convergence respectively. The row *cycles time* indicates the time required by the AMG cycles, and *hss init* indicates the time required to compute the factorization using StruMF. The row *pcgit* indicates the average number of iterations required by PCG/GS

TABLE I. RESULTS FOR THE ISOTROPIC SPE10 PROBLEM.

parameters						
elements/agglomerate	400	400	400	1000	1000	1000
ν	3	3	3	3	5	5
θ	0.001	0.003	0.005	0.005	0.003	0.005
coarse grid						
n	7,782	16,279	25,119	19,570	12,137	19,570
nnz	1.4M	6.1M	14.3M	14.0M	9.5M	24.1M
nnz/n	181.55	372.8	567.8	714.5	779.1	1,231.1
hierarchy	414.0	414.2	418.5	810.9	821.7	828.1
cycles	68	52	50	19	12	10
PCG/GS						
cycles time	89.8	161.4	309.3	147.5	99.8	197.6
pcgit	78.9	92.6	99.7	100.8	80.6	82.2
total	503.8	575.7	727.7	958.3	921.5	1,025.8
FGMRES/StruMF						
hss init	2.6	18.6	61.5	49.9	28.1	126.6
cycles time	69.5	70.6	90.9	62.8	67.8	124.2
total	486.2	503.4	570.4	923.0	917.3	1,078.7

for convergence. The number of iterations required by FGMRES/StruMF is always 3.

Our results show that the crucial performance-influencing parameter is the spectral tolerance θ. Smaller θ leads to a much sparser A_c and to much faster coarse-grid solvers. The sparsity of A_c is especially beneficial for the StruMF solver.

III. ROOFLINE ANALYSIS OF PERFORMANCE BOUND

We develop a bounds-and-bottlenecks model using the roofline approach [8] and validate the model by comparing it to the actual performance of our code. We focus on the AMG solution cycles and on the PCG/J variant of the coarse-grid solver; analyzing FGMRES/StruMF is a work in progress.

The inputs to our analysis are: the machine peak flop rate and its sustainable memory bandwidth; the dimensions of A, P and A_c; and the number of AMG cycles and the average number of PCG iterations per AMG cycle. We use these parameters to express the memory traffic volume and the flop count of the computation, as shown in Table II,[1,2] and then translate the resulting counts into two performance upper bounds: one corresponding to memory bandwidth being the bottleneck, and the other to the machine's flops rate.

The memory bandwidth that we use to derive the memory-traffic bound depends on the dimensions of the matrix. The fine-grid matrix is too large to fit in cache and therefore we use the machine's DRAM–L3 bandwidth; for the coarse step, the appropriate bandwidth is L3–L2. The bounds are expressed in terms of seconds of runtime, as plotted in Fig. 1.

For the AMG solution cycles, when only 1 or 2 cores are used, our bound based on flop count is within 1% and 7% of the actual runtime respectively. As the number of cores is increased, memory bandwidth becomes the bottleneck. For 4, 8 and 12 cores, our bound based on memory bandwidth is within 23% of the actual runtime. Furthermore, the bound in these cases is dominated by the L3–DRAM traffic; the coarse grid accounts for less than 25% of the bound.

[1]The numbers of nonzeros of A, A_c and P are denoted by nza, nzc and nzp respectively. The dimensions of A and A_c are denoted by n and n_c. Real numbers are represented in 8-byte double-precision and integers require 4 bytes.

[2]We used the following combination of parameters: elements/agglomerate was equal to 400, $\nu = 3$ and $\theta = 0.001$.

TABLE II. THE COSTS ASSOCIATED WITH EACH AMG CYCLE.

stage	bytes	flops
pre- and post-smooth	$(3\nu+1)(12\,\mathtt{nza} + 3\cdot 8n)$	$2(3\nu+1)(\mathtt{nza}+2n)$
restriction	$12\,\mathtt{nza} + 12\,\mathtt{nzp} + 3\cdot 8n$	$2(\mathtt{nza}+\mathtt{nzp})$
one coarse solve (PCG/J)		
multiply by A_c	$12\,\mathtt{nzc}$	$2\,\mathtt{nzc}$
preconditioner	$2\cdot 8n_c$	n_c
vector operations	$5\cdot 8n_c$	$2\cdot 5n_c$
interpolation	$12\,\mathtt{nzp} + 8n$	$2\,\mathtt{nzp}$
termination criterion	$12\,\mathtt{nza} + 4\cdot 8n$	$2(\mathtt{nza}+n)$

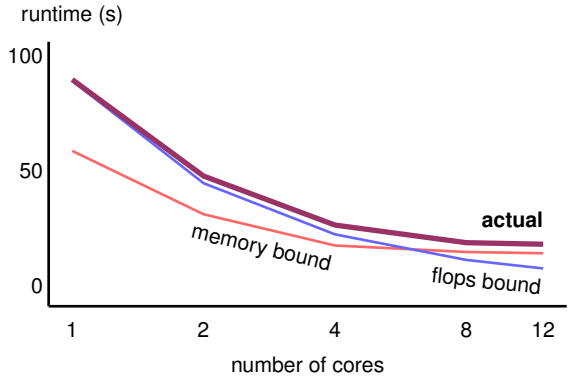

Fig. 1. Runtime bounds and actual runtime (seconds) for the isotropic model.

ACKNOWLEDGMENT

We thank Panayot Vassilevski, Delyan Kalchev, Andrew Barker and Umberto Villa for providing their code and helping us understand it. This material is based upon work supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research. This research used resources of the National Energy Research Scientific Computing Center, which is a DOE Office of Science User Facility.

REFERENCES

[1] M. Brezina and P. S. Vassilevski, "Smoothed aggregation spectral element agglomeration AMG: SA-ρAMGe," in *Large-Scale Scientific Computing*, ser. LNCS, I. Lirkov, S. Margenov, and J. Waśniewski, Eds. Springer Berlin Heidelberg, 2012, vol. 7116, pp. 3–15.

[2] D. Kalchev, "Adaptive algebraic multigrid for finite element elliptic equations with random coefficients," Master's thesis, Sofia University, Bulgaria, 2012.

[3] D. Kalchev, C. Ketelsen, and P. S. Vassilevski, "Two-level adaptive algebraic multigrid for a sequence of problems with slowly varying random coefficients," *SIAM J. Sci Comput.*, vol. 35, no. 6, pp. B1215–B1234, 2013.

[4] Y. Saad, "A flexible inner-outer preconditioned GMRES algorithm," *SIAM J. Sci Comput.*, vol. 14, no. 2, pp. 461–469, 1993.

[5] A. Napov and X. S. Li, "An algebraic multifrontal preconditioner that exploits the low-rank property," Université Libre de Bruxelles, Tech. Rep., 2014. [Online]. Available: http://homepages.ulb.ac.be/~anapov/pub/strumf.pdf

[6] J. Xia, S. Chandrasekaran, M. Gu, and X. S. Li, "Fast algorithms for hierarchically semiseparable matrices," *Numer. Linear Algebra Appl.*, vol. 17, no. 6, pp. 953–976, 2010.

[7] M. A. Christie and M. J. Blunt, "Tenth SPE comparative solution project: Comparison of upscaling techniques," *SPE Reserv. Eval. Eng.*, vol. 4, no. 4, pp. 308–317, 2001.

[8] S. Williams, A. Waterman, and D. Patterson, "Roofline: An insightful visual performance model for multicore architectures," *Commun. ACM*, vol. 52, no. 4, pp. 65–76, 2009.