SAND2008-2082C

Adaptive Partitioning in High-Performance Computing for Scientific Simulations using Adaptive Mesh Techniques

Johan Steensland

Advanced Software Research and Development, Sandia National Laboratories,
P.O. Box 969, Livermore, CA 94550-9915, USA,
jsteens@ca.sandia.gov

Abstract. There are numerous partitioning techniques available within high-performance computing for simulating physical phenomena with adaptive mesh techniques. However, because of the dynamics inherent in these applications and computer systems, manually selecting an optimal technique for a new simulation can be a significant problem. Adaptive partitioning refers to repeatedly selecting, configuring and invoking the optimal partitioning technique at run-time, based on the current state of the computer and application. We investigate whether a relatively simple implementation could automatically improve application performance as compared to routinely using a seemingly good static technique in a dynamic context. We use six real-life adaptive applications from different scientific domains, five complementing partitioning techniques, and a large set of parameters corresponding to a wide spectrum of computing environments. The results show that even a simple implementation of adaptive partitioning can improve performance. Thus, it eliminates the problem of selecting partitioning technique for new simulations.

Key words: AMR, load balancing, adaptive partitioning

1 Introduction

Adaptive mesh refinement (AMR) is widely used for numerical simulations of physical phenomena [6, 5]. When dynamic and localized solution features require higher mesh resolution for obtaining sufficient numerical accuracy, the mesh adapts dynamically. Parallel implementations of AMR potentially lead to realistic three-dimensional models. However, it also presents significant challenges in dynamic resource allocation. The parallel efficiency is limited by the effectiveness of the partitioner to partition and distribute the underlying mesh to expose all inherent parallelism, minimize communication and synchronization overheads, and balance load. Because application adaptation results in repeated modifications of the mesh, it is necessary to repeatedly repartition the mesh to maintain efficient use of resources. The specific requirements for an effective partitioning change with the mesh. By explicitly considering these dynamic conditions, the scalability for large, realistic simulations could possibly be significantly improved. We introduce *adaptive partitioning*, meaning dynamic and automatic switching of partitioning techniques, based on the current run-time state.

We investigate whether a relatively simple implementation of adaptive partitioning could automatically improve application performance as compared to routinely using a seemingly good static technique in a dynamic context.

2 Experimentation

Setup We use five partitioning strategies: recursive coordinate bisection (RCB) [2], RCB plus re-mapping (RCB+M) [2], AdaptiveRepart [4] (PM) from ParMetis [3], the HyperGraph algorithm (HG) from Zoltan [1], and an implementation the Hilbert space-filling curve (HSFC) [2]. We use six real-world unstructured adaptive mesh applications from a variety of scientific domains, with different sizes, geometries, structures, and refinement patterns (see Table 1).

Application	Element	Dim	Timesteps	Avg #elmts	Max #elmts
Quake	Tetra	3D	6	$308K$	$1.6M$
MachStem	Quad	2D	109	$8.2K$	$12.5K$
LaserRaster	Cube	3D	65	$4,4K$	$10.3K$
Convection	Tetra	3D	73	$87K$	$94K$
Spheres	Cube	3D	70	$1.4M$	$1.8M$
ShockTurb	Cube	3D	100	$750K$	$1.3M$

Table 1. Table of applications

To investigate the *impact* a given sequence of partitioners has on the overall parallel efficiency, we use the cost function [6]:

$$\text{Cost} = \text{CCR} \times \text{loadimb.} + \text{ITR} \times \text{edgecut} + \text{migr.}.$$

By applying the cost function to two different sets of data (*average* and *max*), we capture a wider gamut of conditions. To achieve a fictitious application and computing environment that is unbiased and does not favor a certain metric or partitioner, we used the settings CCR $\in \{0.25, 0.5, 1.0\}$ and ITR $\in \{0.01, 0.25, 0.5, 1.0\}$.

To better estimate the data migration cost associated with switching from one partitioner to another, we use the penalty function [5], $F(f)$:

$$F(f) = \begin{cases} f \text{ for } P_i^I \rightarrow P^S \\ 1 \text{ otherwise.} \end{cases}$$

The data migration component in the cost function is then multiplied by $F(f)$.

For each application, we apply both the average and max data sets to the cost function with all permutations of f, CCR, and ITR. For all applications except Quake, the number of partitions tested is 8 and 16. For Quake, we use 32 and 64.

Results Table 2 shows a small, but typical subset of a performance comparison of the best adaptive partitioning and the best static algorithm for the particular application.

ITR=HG_MULT=0.25

App.	Max, 0.25	Avg, 0.25	Max, 0.5	Avg, 0.5	Max, 1.0	Avg, 1.0
Quake32	(1,4)=96	(1,4)=100	(1,4)=93	(1,4)=100	(1,4)=64	(2,4)=100
Quake64	(1,4)=100	(1,4)=100	(1,4)=100	(1,4)=100	(1,4)=100	(1,4)=97
Mach8	(8,3)=95	(8,3)=102	(1,3)=94	(2,3)=101	(4,3)=88	(2,3)=87
Mach16	(2,3)=97	(2,3)=99	(2,3)=97	(4,3)=103	(4,3)=98	(8,3)=99
L-R8	(4,3)=93	(1,3)=104	(1,3)=92	(1,3)=102	(1,3)=88	(1,3)=92
L-R16	(2,3)=85	(2,3)=93	(2,3)=86	(4,3)=95	(4,3)=86	(8,3)=98
Conv8	(2,4)=96	(2,4)=96	(4,4)=93	(2,4)=87	(2,4)=84	(4,4)=69
Conv16	(2,4)=95	(2,4)=94	(2,4)=91	(2,4)=91	(4,4)=96	(4,4)=86
Sphe8	(2,3)=99	(4,3)=102	(2,3)=99	(8,3)=102	(4,3)=97	(1,3)=80
Sphe16	(2,3)=96	(4,3)=96	(2,3)=96	(4,3)=97	(2,3)=96	(1,3)=89
Shock8	(4,3)=100	(8,3)=100	(4,3)=100	(8,3)=100	(4,3)=102	(1,3)=85
Shock16	(2,3)=94	(8,3)=100	(2,3)=95	(8,3)=100	(4,3)=95	(2,3)=95

Table 2. Notation: For $(X,Y) = Z$, X is the best penalty, Y is the index of the best static algorithm, where 1=RCB, 2=RCB+MAP, 3=ParMetis, 4=HyperGraph, 5=HSFC, and Z is the cost ratio of the best adaptive and the best static algorithm.

Figure 1 shows a typical performance comparison of all partitioners for a small subset of the parameters.

Discussion The results show that even a simple implementation of adaptive partitioning can improve performance. Thus, it eliminates the problem of selecting partitioning technique for new simulations.

3 Acknowledgments

Thanks to John Peterson at CFDLab, University of Texas, Richard Drake, James Overfelt, and Karen Devine at Sandia, Charles Norton at Jet Propulsion Lab, NASA, and Ralf Deiterding at Oakridge National Laboratories. Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy under contract DE-AC04-94-AL85000.

References

1. U.V. Catalyurek, E.G. Boman, K.D. Devine, D. Bozdag, R.T. Heaphy, and L.A. Riesen. Hypergraph-based dynamic load balancing for adaptive scientific computations. In *Proc. of 21st International Parallel and Distributed Processing Symposium (IPDPS'07)*. IEEE, 2007. Best Algorithms Paper Award.
2. Karen Devine, Erik Boman, Robert Heaphy, Bruce Hendrickson, and Courtenay Vaughan. Zoltan data management services for parallel dynamic applications. *Computing in Science and Engineering*, 4(2):90–97, 2002.
3. G. Karypis, K. Schloegel, and V. Kumar. PARMETIS - parallel graph partitioning and sparse matrix ordering library, version 2.0. Univ. of Minnesota, Minneapolis, MN, 1998.
4. K. Schloegel, G. Karypis, and V. Kumar. A unified algorithm for load-balancing adaptive scientific simulations. In *Proceedings of Supercomputing 2000*, 2000.

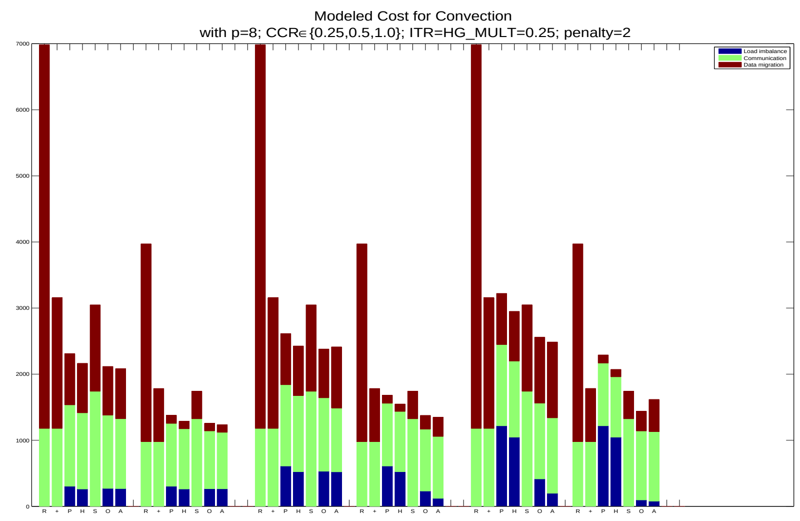

Fig. 1. Performance for all partitioners including the theoretical optimal adaptive partitioning (O) and the actual adaptive partitioning (A).

5. Johan Steensland. Reducing data migration in the context of adaptive partitioning for amr. In *Proceedings of The 19th IASTED International Conference on Parallel and distributed computing and systems PDCS07*, volume n, pages X–X. ACTA PRESS, 2007.
6. Johan Steensland and John Peterson. A study of dynamically adaptive partitioning for AMR. In *Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'07)*. To appear.