Design of Unstructured Adaptive (UA) NAS Parallel Benchmark featuring irregular, dynamic memory accesses

Huiyu Feng*, Rob F. Van der Wijngaart†, Rupak Biswas‡
{fhy,wijngaar,rbiswas}@nas.nasa.gov
NAS Technical Report NAS-01-012
December 2001
NASA Ames Research Center, M/S T27A-1
Moffett Field, CA 94035, USA

Abstract

We describe the design of a new method for the measurement of the performance of modern computer systems when solving scientific problems featuring irregular, dynamic memory accesses. The method involves the solution of a stylized heat transfer problem on an unstructured, adaptive grid. A Spectral Element Method (SEM) with an adaptive, non-conforming mesh is selected to discretize the transport equation. The relatively high order of the SEM lowers the fraction of wall clock time spent on inter-processor communication, which eases the load balancing task and allows us to concentrate on the memory accesses. The benchmark is designed to be three-dimensional. Parallelization and load balance issues of a reference implementation will be described in detail in future reports.

1 Introduction

The NAS Parallel Benchmarks (NPB) [1] were originally formulated to measure the performance of modern computer architectures—especially parallel machines—when applied to computational problems of importance to the scientific community in general, and to NASA in particular. Despite the relatively limited scope of the eight problems that made up NPB, the benchmarks became quite popular and have been widely used by researchers, computer vendors and buyers, and software tool developers alike. Since their initial release as paper-and-pencil specifications in 1991, and as source code implementations (in MPI) in 1995 [2], it has become increasingly clear that the NPB problems lack in the area of irregular and dynamically changing memory accesses. Such accesses may defeat the hardware and software support for memory traffic in modern computer architectures. The original NPB featured static applications with (mostly) fixed-stride memory access, which can be exploited by compilers and specialized hardware to reduce the cost of memory traffic. Not all applications of importance to NASA are static, however, and it was found by Oliker and Biswas [9] that some strongly dynamic applications fare poorly when implemented on certain widely used modern parallel computers. This situation is exacerbated when the programming environment selected offers limited control over data placement and granularity of data traffic, as is often the case with distributed shared memory systems.

*Mechanical and Aerospace Engineering Department, George Washington University, Washington, DC, 20052
†Computer Sciences Corporation
‡NASA Advanced Supercomputing Division

The current work, to be incorporated in the NPB suite, means to provide a standardized method for gauging the performance of computer systems when running scientific applications whose memory access patterns are irregular and unpredictable. Here we present an outline of the application selected, as well as a motivation for the specific design choices that led to the problem specification. In a subsequent report we provide a full paper-and-pencil description of the entire application.

Computing solutions to systems of unsteady partial differential equations (PDEs) on uniform meshes within an a priori specified error tolerance can be achieved by employing consistent discretization schemes and a sufficiently high spatial resolution, assuming numerical stability. However, for problems with localized sources of error, such as weather forecasting or advancing front problems, uniform meshes are often not efficient. Computational methods based on the use of adaptively constructed nonuniform meshes have the potential to reduce the amount of computation and storage necessary to perform many scientific calculations. Parallel implementation of such computations based on adaptive meshes may provide further performance improvement, but raises many additional issues, such as data distribution, load balance, inter-processor communication, data dependence, and false and true data sharing.

The class of unstructured, adaptive grid problems is identified as a good candidate on which to base our new benchmark. Several points shall be considered in constructing the specific benchmark problem:

1. It should be representative of a class of problems relevant to a significant part of the scientific computing community.

2. In order to keep the benchmark size and complexity under control, and to make sure that little time is spent in modules that constitute parallel overhead that would not be incurred by machines like the Cray (formerly Tera) Multithreaded Architecture (MTA), the problem should be as simple as possible without sacrificing credibility and effectiveness.

3. Any load imbalance resulting from grid adaptation must be taken care of by data (re)partitioning (for distributed-memory systems) and task (re)assignment among processors. It is not useful to measure performance of a problem with substantial load imbalance, so we must ensure that the problem can be load balanced for a whole range of numbers of processors, while the repartitioning and data remapping itself may consume only little time.

To satisfy the above requirements, we select a stylized heat transfer problem on a cubical domain with Dirichlet boundary conditions, which makes this benchmark similar in structure to the existing NPB structured-grid benchmarks BT, SP, and LU. The grid consists of hexahedral elements which will be nonconforming in the case of nonuniform refinement/coarsening (see Figure 1a). A Spectral Element Method (SEM) is selected for the spatial discretization, and a time-split method composed of fourth-order Runge-Kutta and preconditioned Conjugate Gradient is used to solved the discrete equations. We use an SEM of a fairly high order, which increases the amount of work per point in the grid, and reduces the fraction of execution time spent on inter-processor communication in the case of distributed-memory implementation. In our reference implementation a tree structure is used to keep track of the grid refinement, which is used later when coarsening is needed.

2 Problem description

The mathematical model for the heat transfer problem is as follows:

$$T_t + \mathbf{v} \cdot \vec{\nabla} T = \epsilon \nabla^2 T + S(\mathbf{x}, t)\,, \tag{1}$$

where the source term is defined by

$$S(\mathbf{x}, t) = \begin{cases} \frac{1}{2}[\cos(\pi||\mathbf{x} - \mathbf{x_0} - \mathbf{v}t||) + 1] & \text{if} \quad ||\mathbf{x} - \mathbf{x_0} - \mathbf{v}t|| \leq 1 \\ 0 & \text{if} \quad ||\mathbf{x} - \mathbf{x_0} - \mathbf{v}t|| > 1 \end{cases} \tag{2}$$

Here $||\cdot||$ signifies the Euclidean norm. The location vector $\mathbf{x}$ is defined by (x, y, z), and $\mathbf{x_0}$ is the initial location of the center of the source. The initial temperature distribution on the domain, consisting of the unit cube $[0, 1]^3$, is zero, and the Dirichlet boundary conditions are fixed at zero as well. The prescribed velocity field $\mathbf{v} = (u, v, w)$ is uniform and constant, and equals the speed of the source. Hence, it is known explicitly when and where mesh refinement and coarsening are required. While solution-adaptive schemes based on local error estimates are more common in practical engineering computations, our prescribed feature motion makes the adaptation more easily controlled and limits the complexity and numerical sensitivity of the benchmark. High resolutions are applied where large temperature gradients exists, i.e. near the traveling source. After the source has passed a certain region, high resolutions are no longer needed there, and grid coarsening is applied. This heat transfer problem is a good candidate for mesh adaptation, as both refinement and coarsening are clear and obvious. Even though the elements are rectangular, the circular shape of the source results in a significantly nonuniform grid.

2.1 Spectral Element Method

The spatial discretization method selected is the Spectral Element Method (SEM) [10], which is a high order weighted residual technique that combines the geometrical flexibility of the Finite Element Method and the high accuracy of spectral methods. The SEM provides exponential spatial convergence for problems with smooth solutions. The reason for using this method is that the high order of the SEM involves proportionally many arithmetic operations per data element, which renders the time spent doing communication and load balancing on distributed memory systems relatively insignificant.

In the SEM, the solution domain is broken up into macroelements. We discretize the dependent variable as follows,

$$T^K(\mathbf{x}) = \sum_{i=0}^{N} \sum_{j=0}^{N} \sum_{k=0}^{N} T^K_{ijk}\; h_i(r) h_j(s) h_k(t)\,, \tag{3}$$

where the h functions are the Lagrangian interpolants based on orthogonal sets of Legendre polynomials of high degree N, and r, s, t are the local coordinates within each element K ($(r, s, t) \in [-1, 1]^2$). T^K_{ijk} is the value of the temperature at the collocation point with indices (i, j, k) within element K. Performing Gauss-Lobatto quadrature on the discrete variational form of the equation to be solved and summing contributions from adjacent elements, we obtain a system of ordinary differential equations (ODEs). Details may be found in many references (e.g. [4, 7, 8, 10]).

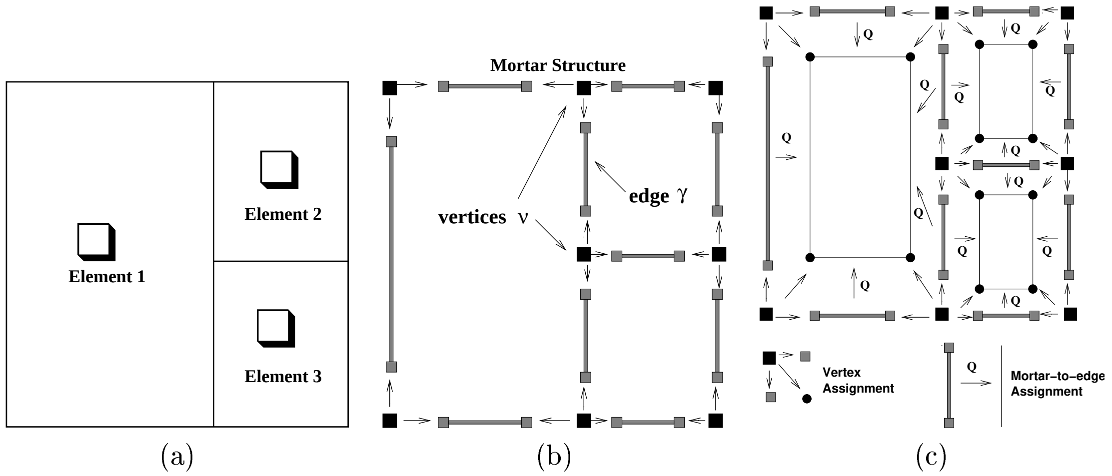

Figure 1: (a) A typical nonconforming mesh, (b) the structure of its corresponding mortar, (c) projection operation Q on nonconforming mesh.

A two-step time splitting scheme is used to solve the system of ODEs:

$$\hat{T}^{n+1} = \mathrm{RK}_{4,\Delta t}(-\mathbf{v} \cdot \vec{\nabla} T^n + S(\mathbf{x}, t)) \tag{4}$$

$$\frac{T^{n+1} - \hat{T}^{n+1}}{\Delta t} = \epsilon \nabla^2 T^{n+1} \tag{5}$$

The convection and source terms in Eq. (4) are advanced in time by an explicit, fourth-order Runge-Kutta method (RK_4). The stability region of RK_4 includes part of the imaginary axis, which is important, because the anti-symmetric contribution of the convective term to the right hand side discretization matrix produces imaginary eigenvalues. Stability is guaranteed by selecting the time step Δt such that the *Courant-Friedrichs-Lewy* condition is satisfied. Explicit time integration of the diffusion term would lead to an overly restrictive time step constraint (stiffness). Instead, we use the Euler Implicit method for time advancement of the diffusion term, as shown in Eq. (5). The variational form of this equation is

$$\left(\frac{T^{n+1} - \hat{T}^{n+1}}{\Delta t}, v\right) = (\epsilon \nabla T^{n+1}, \nabla v) \tag{6}$$

where (.,.) represents the L^2 inner product, and v is the test function.

A fully implicit scheme is applied to solve this equation, employing an iterative Conjugate Gradient method. A diagonal preconditioner is used to accelerate convergence.

2.2 Mortar Method

A nonconforming mesh—see Figure 1(a)—is used to provide geometrical flexibility, which is essential to grid adaptation. A *mortar method* is introduced to allow the matches between nonconforming elements [3, 5]. The configuration of the discretization using a nonconforming mesh is illustrated in Figure 1(b) where each element is surrounded by a mortar structure γ with vertices ν at each corner. All the neighboring elements with a matching edge (2D) or face (3D) share the same mortar.

This method is based on the introduction of an auxiliary space (mortar), which is defined at the boundaries of individual elements. It preserves local structure while decoupling local internal residual evaluations from transmission of the continuity conditions and imposition of boundary conditions. It is this clean decoupling which allows for the efficiency of the method when used in conjunction with fast iterative solvers, especially in the context of parallel computations. It allows for intensive per element/subdomain calculations, with a minimum of communication between elements. A mortar element is effectively an interpolation polynomial whose support consists of the smaller of the faces on either side of the interface between grid elements. The coefficients of the polynomial are defined such that the jump between values of the dependent variables across the interface is minimal in an integral sense [3].

The mapping of discretization coefficients of neighboring grid elements to the mortar interface element takes place through a transformation matrix Q (see Figure 1(c)), which reduces to the identity operator at those grid interfaces where the mesh is conforming. Values of the dependent variable on the element boundaries are transferred to the mortar by applying Q^T. The mortar is updated by summing all the transferred values of the residual from neighboring elements. The updated mortar value of the residual is then transferred back to the element edges/faces using Q. However, when mapping the dependent variable between adaptively updated meshes, to avoid a discontinuity across the nonconforming edge/face, the mortar value is copied from the shortest edge/face of the element on one side of the mortar without summation.

3 Mesh adaptation

The goal of the UA benchmark is to gauge the performance of modern computer architectures on applications featuring irregular and dynamically changing memory accesses. Time- and location-dependent adaptation of the topology of unstructured meshes provides both features. Mesh adaptation includes grid refinement and coarsening. It efficiently provides high resolution in areas of the domain where large or rapidly varying physical changes exist, while saving unnecessary computation where the solution is smooth or where error-inducing physical phenomena are absent. An adaptation algorithm based on globally unstructured, locally structured nonconforming hexahedral meshes is being developed and can be described as follows.

Create a uniform coarse grid. Based on the distance between the element and the center of the source, perform mesh adaptation to refine the grid. The level of refinement increases in steps of unit size as the distance between element and source decreases. It is forbidden to have a difference in refinement level of more than one between adjacent elements. Begin the time integration with the refined initial grid. Perform mesh adaptation (refinement and coarsening) at certain prescribed intervals of physical time, depending on how fast the source travels. This adaptation is again determined by the distance between each element and the source. After each mesh adaptation, map the solution to the newly updated mesh, and continue to the next time step computation. Both mesh refinement and coarsening are isotropic: when an element needs to be refined, it is always divided evenly in all three coordinate directions, producing eight child elements. And when coarsening the grid, the reverse procedure is performed: the eight children are combined into a single parent element. Coarsening is not allowed beyond the initial coarse grid created at the start of the computation.

As nonconforming elements are allowed in the grid, an element may match either one or four elements at a particular face. In our serial reference implementation a tree data structure is used to trace the geometrical relations between elements. Each element has a code which is combination of 0's and 1's. Each parent element has eight child elements whose suffix codes are 000, 001, 010, 011, 100, 101, 110 and 111, see Figure 2a. A complete element code consists of the parent's code, concatenated with the child code, as indicated in Figure 2b.

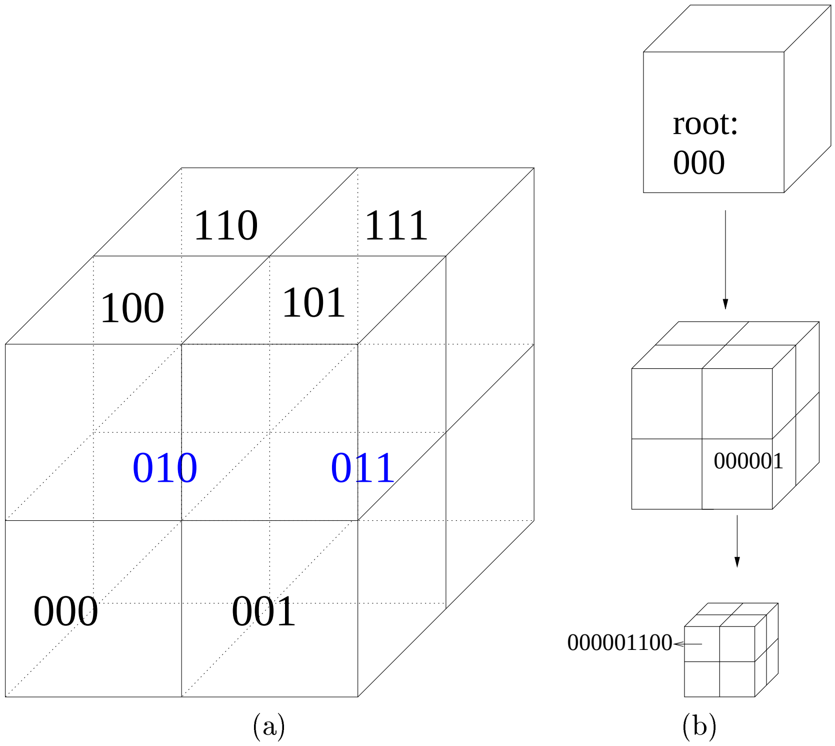

Figure 2: (a) Codes for eight child elements of a parent element (b) Tree structure of element refinement

One imposed limitation on the grid refinement is that one element can only match four elements on one face, so when doing the coarsening, this restriction must be taken into consideration to avoid the creation of forbidden meshes. Our procedure for coarsening which produces the required levels of (de)refinement is as follows. First mark all the elements that are candidates for coarsening. Then check whether all eight child elements of a parent are marked, If so, check whether the coarsening will result in one element matching more than four elements on any face. If all these conditions are satisfied, then the eight child elements will be combined into the parent element. Checking whether eight child elements are from the same parent is quite simple to implement, using the element codes; these may only differ in the last three bits.

To illustrate how the mesh adaption works, Figure 3 shows the mesh (2D) adaptively refined in the area where large temperature gradients exist, i.e. near the source. The shaded disk indicates the source, and the solid lines indicate grid element boundaries.

4 Parallelization Issues

While the problem specification allows any type of implementation that conforms to the numerics prescribed, we outline a useful strategy for parallel implementation on a distributed-memory system.

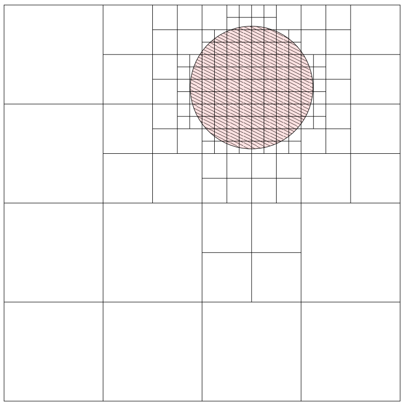

Figure 3: Mesh adaptively refined to assign more elements in area where large temperature gradients exist (2D cut of 3D mesh).

4.1 Load balance

Keeping the order of the SEM fixed throughout the computational domain, the amount of work per element is constant. Consequently, assigning the same number of elements to each processor results in a good load balance. A useful restriction is to partition the mesh such that a baseline element (one belonging to the coarse initial mesh) is never split across multiple processors. That way the creation of parent elements by coarsening remains straightforward.

4.2 Communications

While partitioning to obtain a good load balance can be achieved efficiently in numerous ways, it is very hard to create the optimal partitioning that minimizes the amount of data communication across partition boundaries. In a computation with a low computation to communication ratio per interface element, it is essential to minimize the number of interface elements. Solving the minimization problem can be quite costly in terms of CPU time, but this is not what we want to measure. Consequently, we design our benchmark problem such that optimal partitioning with respect to communication volume is not critical. This is achieved by choosing a fairly high order of the SEM (as compared to the traditional finite-difference methods used in the other NAS Parallel Benchmarks), which increases the ratio of computation to communication for interface elements. This allows us to employ the rather simple partitioning strategy of Recursive Coordinate Bisection without a severe penalty. The method is adapted to make sure no split occurs between child elements of the same parent.

References

[1] Bailey, D.H., Barton J., Lasinski T., and Simon, H. (Eds.) (1991), The NAS Parallel Benchmarks, *NAS Technical Report RNR-91-002*, NASA Ames Research Center, Moffett Field, CA

[2] Bailey, D.H., Harris, T., Saphir, W.C., Van der Wijngaart R.F., Woo, A.C., Yarrow, M. (1995), The NAS Parallel Benchmarks 2.0., *NAS Technical Report NAS-95-020*, NASA Ames Research Center, Moffett Field, CA

[3] Bernardi, C. Maday, Y. and Patera, A. T. (1994), A new nonconforming approach to domain decompositions: The mortar element method. *Pitman Research Notes in Mathematics*, series 1,No. 299:13.

[4] Henderson, R. D. (1999), Adaptive Spectral Elements Methods for Turbulence and Transition, in *High-Order Methods for Computational Physics*, T.J. Barth and H. Deconinck (eds.), Springer, 225-324.

[5] Maday, Y., Mavriplis, C. and Patera, A.T.(1989), Non-Conforming Mortar Element Methods: Application to Spectral Discretizations, in *Domain Decomposition Methods*, pp. 392-418, SIAM, also ICASE Report 88-59.

[6] Mavriplis, C. (1990), *A Posteriori* Error Estimators for Adaptive Spectral Element Techniques, *Notes on Numerical Methods in Fluid Mechanics*, 29:333, Vieweg.

[7] Mavriplis, C. (1994), Adaptive Mesh Strategies for the Spectral Element Method, *Computer Methods in Applied Mechanics and Engineering*, 116: 77-86.

[8] Mavriplis, C. and Hsu, L.-C. (1997), A Two-Dimensional Adaptive Spectral Element Method, *Proceedings of the 13th AIAA Computational Fluid Dynamics Conference*, Snowmass.

[9] Oliker, L., Biswas, R. (2000), Parallelization of a Dynamic Unstructured Algorithm Using Three Leading Programming Paradigms, *IEEE Transactions on Parallel and Distributed Systems*, Vol. 11, No. 9, pages 931-940

[10] Patera, A. T. (1984), A Spectral Element Method for Fluid Dynamics: Laminar Flow in a Channel Expansion, *Journal of Computational Physics*, 54(4):468–488.