ULTRASCALABLE IMPLICIT FINITE ELEMENT ANALYSES IN SOLID MECHANICS WITH OVER A HALF A BILLION DEGREES OF FREEDOM

MARK F. ADAMS[1] , HARUN H. BAYRAKTAR[2,6] , TONY M. KEAVENY[2,3,4] , AND PANAYIOTIS PAPADOPOULOS[3,5]

Abstract. The solution of elliptic diffusion operators is the computational bottleneck in many simulations in a wide range of engineering and scientific disciplines. We present a truly scalable—ultrascalable—linear solver for the diffusion operator in unstructured elasticity problems. Scalability is demonstrated with speedup studies of a non-linear analyses of a vertebral body with over a half of a billion degrees of freedom on up to 4088 processors on the ACSI White machine. This work is significant because in the domain of unstructured implicit finite element analysis in solid mechanics with complex geometry, this is the first demonstration of a highly parallel, and efficient, application of a mathematically optimal linear solution method on a common large scale computing platform—the IBM SP Power3.

1. Introduction. The availability of large high performance computers is providing scientists and engineers with the opportunity to simulate a variety of complex physical systems with ever more accuracy and thereby exploit the advantages of computer simulations over laboratory experiments. The finite element (FE) method is widely used for these simulations. The finite element method requires that one or several linearized systems of sparse unstructured algebraic equations (the stiffness matrix) be solved for static analyses, or at each time step when implicit time integration is used. These linear system solves are the computational bottleneck (once the simulation has been setup and before the results are interpreted) as the scale of problems increases. Direct solvers (eg, LU decomposition) and one level solvers (eg, diagonally preconditioned CG) have been popular in the past but, with the ever decreasing cost of computing, the size of problems that are easily tractable for many scientists and engineers today renders these methods impractical. Thus, the use of mathematically optimal linear solution methods has become an important and pressing issue for simulations in many areas of science and engineering.

We investigate the application of mathematically and computationally optimal solver methods to the diffusion operator from displacement finite element formulations of elasticity. This work is significant in that in the domain of unstructured implicit finite element analysis in solid mechanics with complex geometry, this is the first demonstration of a highly parallel and efficient application of a mathematically optimal linear solution method on a common large scale computing platform.

Solvers for the elliptic diffusion operator are used in almost all unstructured implicit analyses in the computational sciences and thus the technology presented here has the potential to be applied to a wide variety of problems and have significant impact. Multigrid methods are well known to be theoretically optimal for both scalar problems like Poisson's equation and systems of PDEs like displacement finite element discretizations of elasticity [11]. Multigrid has been applied to structured grid problems for decades [13]; and in the past ten years, algebraic multigrid (AMG) methods have been developed for unstructured problems. One such method, that has proven to be well suited to elasticity problems is smoothed aggregation [2, 33, 42].

This paper demonstrations the effectiveness of smoothed aggregation AMG on large-scale elasticity problems from trabecular bone finite element modeling. Trabecular bone is the primary load-bearing biological structure in the human spine as well as at the end of long bones such as the femur. It has a complex structure with typical porosity values exceeding 80% in most anatomic sites. A common method to study the structural properties of trabecular bone is to use specimen-specific high-resolution finite element models obtained from 3D micro-computed tomography (micro-CT) images (Figure 1.1). This process converts image voxels into a finite element mesh of hexahedral elements. These voxel meshes have the advantage of being able to capture complex geometries intrinsically but require many elements to accurately model the mechanics. For example, a convergent linear analysis of one human vertebral body requires over 135 million elements with

[1]Sandia National Laboratories, PO Box 969, MS 9217, Livermore, CA 94551 (mfadams@sandia.gov)
[2]Orthopaedic Biomechanics Laboratory, University of California, Berkeley, CA
[3]Department of Mechanical Engineering, University of California, Berkeley, CA
[4]Department of Bioengineering, University of California, Berkeley, CA
[5]Computational Solid Mechanics Laboratory, University of California, Berkeley, CA
[6]ABAQUS, Inc., Pawtucket, RI
[7]SC'04, November 6-12, 2004, Pittsburgh, PA, USA

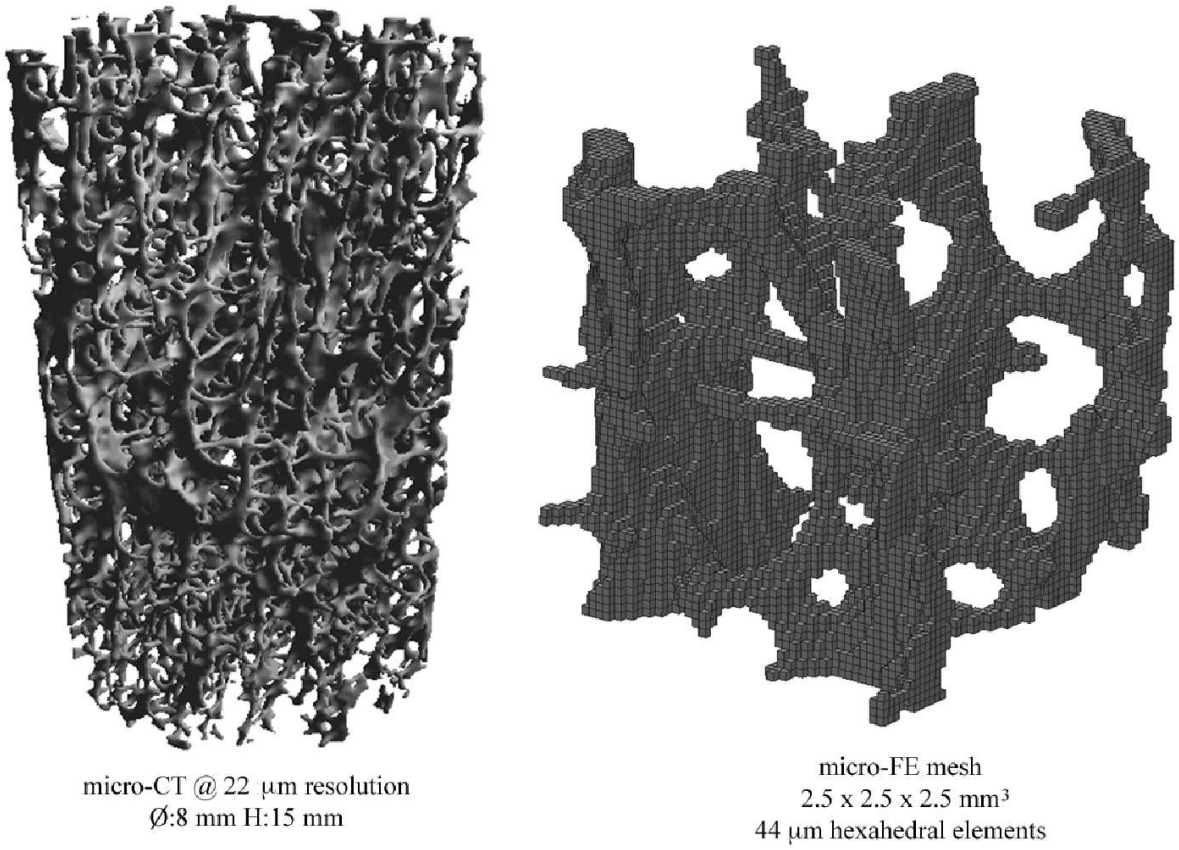

FIGURE 1.1. *Rendering of a human vertebral bone cylindrical specimen and detail showing voxel finite elements*

a resolution of 30 microns.

The biomechanics component of our research primarily involves nonlinear micro-FE analyses [31, 29, 10, 9], for which an inexact Newton method is used. Newton's method requires the solution of a linearized set of equations in an iterative process. That is the solution of

$$A(x)x = b$$

where A is the FE stiffness matrix, x is the solution vector to be solved for, and b is the residual vector. It is useful to solve the system inexactly when an iterative method is used in the Newton iterations, particularly when the solution is far from the solution and hence the linearization is not generally accurate.

High-resolution finite element models of trabecular bone have traditionally been solved with element-by-element (EBE) preconditioned conjugate gradient (CG) method [24, 41, 40, 31, 10]. While the EBE-PCG solver is very memory efficient (the stiffness matrix need not be explicitly constructed), and the work per iteration, as well as per degree of freedom, is constant, the number of iterations required to reduce the residual by a constant fraction rises dramatically as the problem size increases.

This paper proceeds by first introducing the current state of the art in high-resolution finite element modeling of bones in §2, followed by an introduction to AMG and complexity analysis in §3. §4 discusses our project's computational architecture. In §5 the performance of large scale nonlinear whole bone modeling problems is investigated, and we conclude in §6.

2. Micro-FE Modeling of Bone. Osteoporotic fractures in the hip and spine are increasingly common. In the US alone, osteoporosis affects an estimated 44 million Americans and generates health care costs in excess of $17 billion annually. The two types of bone—cortical and trabecular—have the functional

task of withstanding stresses that arise during daily activities, as well as those arising from non-habitual loading (eg, during a fall). Therefore, understanding the mechanisms that lead to failure in bones is of great clinical importance. Characterization of bone mechanical properties [16] is a challenging problem, since they vary between anatomic sites, across individuals, over time, and with disease. Our main area of research in orthopaedic biomechanics is the mechanical behavior of trabecular bone (for a review, see Keaveny [26]) which is highly porous and has a microstructure that adapts to its mechanical loading environment through remodeling.

A widely used tool to investigate the mechanical behavior of trabecular bone is the micro-finite element (micro-FE) method (Figure 1.1), which uses high-resolution micro-computed tomography (micro-CT) [34] scans to obtain geometrically accurate FE models of specimens (Figure 1.1). While these voxel meshes with non-smooth surfaces have initially raised concerns among researchers, the accuracy of these models has been investigated in detail [39, 21, 32]. A general rule for convergence of stresses at the tissue and apparent levels is to ensure a resolution such that three to four elements exist across the thickness of a beam or plate (trabecula) in the trabecular bone structure [21, 32]. To date, the micro-FE approach has been used successfully in studying bone mechanical properties at three different levels. (a) At the trabecular tissue level, elastic [41] and yield properties [10] have been determined using specimen-specific experimental data. (b) At the continuum level, trabecular bone orthotropic elastic properties [38], effects of geometrically nonlinear deformations [7, 35], and yield behavior under multiaxial loads [30, 9] have been investigated. (c) At the whole bone level, load transfer characteristics [8] and stress distributions in healthy and osteoporotic bones [40, 23] have been studied. Current research using the micro-FE approach is primarily focused on problems with geometric and material nonlinearities (eg, plasticity, damage, etc.) and whole bone analysis to gain insight into the structure-function relationships in bone.

The current standard linear equation solver in almost all bone micro-FE analyses is the one level element-by-element preconditioned conjugate gradient (EBE-PCG) method [24]. The main advantage of the EBE-PCG solver is that it is very memory efficient due to its global stiffness matrix-free algorithm which only requires a matrix-vector product. This solver also takes advantage of the fact that all elements in a voxel based mesh have the same element stiffness matrix. Currently, such linear elastic finite element analysis capabilities with the EBE-PCG solver are incorporated into micro-CT scanner software (SCANCO Medical AG, Bassersdorf, Switzerland) increasing the popularity of the voxel based models. While this method is attractive for linear elastic analysis of problems with under 1 million elements, the slow convergence of EBE-PCG and relatively poor scalability makes it unattractive for whole bone micro-FE models with over 10 million elements, as well as problems with geometric and/or material nonlinearities.

Recently, with the increasing availability of parallel computers, as well as micro-CT scanners that can scan whole bones at resolutions that capture the trabecular architecture in detail, high-resolution finite element analyses of models with tens of millions of elements has become possible [3, 8, 40, 23]. However, the orthopaedic biomechanics community is quickly reaching the limit of the usability of one level solvers—for instance, a linear elastic analysis of a high-resolution finite element model of the proximal femur with 96 million elements using EBE-PCG with a convergence tolerance of 10^{-3} took 25,000 CPU hours on 30 processors of an SGI-Origin2000 computer with 250MHz-R10000 processors using 17GB of memory [40]. Similarly, using the same EBE-PCG solver, linear micro-FE analysis of a human vertebral body with 33 million elements, required about 20,000 CPU hours on 16 processors of an SGI-Origin3800 computer. While the memory requirement is small for the EBE-PCG solver, the slow convergence (8 weeks of wall-clock time for the vertebral body model) to reduce the residual by three orders of magnitude is indicative of the need for scalable solvers for such problems.

3. Multigrid introduction. Multigrid is well known to be an optimal solution method for H^1-elliptic problems on structured meshes with a serial complexity of $\mathcal{O}(n)$ (with n degrees of freedom) and $\mathcal{O}(\log n)$ in parallel [13, 22, 17, 12]. Multigrid is motivated, first, by the fact that inexpensive iterative methods, such as Gauss-Seidel, are effective at reducing high frequency or high energy error. These solvers are called *smoothers* because they render the error geometrically smooth by reducing the high frequency content of the error. Smoothers are, however, ineffectual at reducing low energy error. The second observation that motivates multigrid methods is that this low energy error can be represented effectively with a coarse version of the problem and that this coarse problem can be "solved" recursively in a multigrid process. That is,

the solution can be projected to a smaller space to provide a coarse grid correction, in much the same way that the finite element method computes an approximate solution by projecting the infinite dimensional solution onto a finite dimensional subspace. Multigrid is practical because this projection can be prepared and applied with reasonable and scalable cost.

Over the past decade algebraic multigrid (AMG) methods have been developed for unstructured problems. AMG methods are, by the broadest definition, methods that construct the coarse grid operators "algebraically" (ie, internally), usually via a Galerkin process. More specifically, AMG methods refer to multigrid methods that construct the coarse grid spaces and operators from the matrix alone. We investigate one such method — smoothed aggregation [43]. Smoothed aggregation starts with the kernel vectors of the fine grid problem without essential boundary conditions (ie, the six rigid body mode vectors for 3D elasticity) and constructs nodal aggregates of strongly connected subdomains which are used to inject the kernel vectors into a block diagonal rectangular matrix to form a so called tentative prolongator P_0.

This tentative prolongator can be used in a standard algebraic multigrid framework to provide a "plain" aggregation algorithm [15]. Note, a multigrid prolongator transfers coarse grid corrections to the fine grid and is all that is required to define the multigrid projection process, given that the restriction operator, which maps fine grid residuals to the coarse grid, is defined as $R = P^T$ and the coarse grids are constructed with a Galerkin process: $A_{i+1} \leftarrow RA_iP$. Smoothed aggregation can be motivated by multigrid theory, in which the energy of the coarse grid spaces degrades the bound on the convergence rate of a multigrid method. The energy of the coarse grid functions of plain aggregation (ie, the columns of P_0) can be reduced with a damped Jacobi process: $P \leftarrow (I - \frac{1}{\omega}D^{-1}A)P_0$. This, along with a smoother for each coarse grid and an accurate solver for the coarsest grid, provides all of the operators required for a standard multigrid process [22, 37, 14].

3.1. Multigrid complexity. Multigrid is an optimal solution method with polylogarithmic parallel complexity—$\mathcal{O}(\log n)$—in the PRAM complexity model [20]. Theory can not provide hard bounds on the convergence rate of multilevel methods on unstructured grid problems, as it can for structured problems, but can provide expressions for bounds on the condition number of the preconditioned system in terms of coarse grid quantities (eg, characteristic subdomain sizes H and characteristic discretization size h, mesh quality metrics, and so on), as well as assumptions about the underlying operator and discretization. Typical condition number bounds for multilevel methods are of the form

$$\kappa = \mathcal{O}(1 + \log^m \frac{H}{h}), \qquad m \leq 3 \tag{3.1}$$

where $\frac{H}{h}$ reflects the coarsening rate between grids and m depends details of the multigrid method and discretization.

The expected complexity of using most iterative methods can be estimated by combining the expected number of iterations (assumed proportional to the condition number of the preconditioned system) with the complexity of each iteration in terms of the number of unknowns n, or characteristic discretization size h, and relevant quantities used in the condition number bound (eg, subdomain size H). The advantage of employing multilevel methods is that a (small) constant $\frac{H}{h}$ (eg, 2 or 3) in Equation 3.1 can be used with as many levels as is required to allow the coarse grid solve—which must be accurate—to have a complexity of $\mathcal{O}(1)$. By limiting the number of levels (ie, by using a two level method) one must either increase $\frac{H}{h}$ to keep the cost of the coarse grid solve affordable and accept degradation in convergence, or keep $\frac{H}{h}$ a constant and accept growth in complexity of the coarse grid solve. Multigrid can be optimal because: 1) the condition number of a system preconditioned with multigrid is bounded by a constant (ie, is not a function of h or the number of levels) and 2) the cost of each iteration (eg, number of floating point operations) is proportional to n in the limit. Thus, multigrid applied to some model problems has provably $\mathcal{O}(n)$ serial complexity, but has polylogarithmic parallel complexity because, in the limit, some processors remain idle on the coarsest grids. Smoothed aggregation provides a condition number that is provably $\mathcal{O}(1+\log n)$ for elasticity problems with regularity and $\mathcal{O}(1 + \log^3 n)$ for problems without regularity [42]. Note, truly optimal convergence requires $\mathcal{O}(1)$ iterations, or work per unknown, to attain accuracy to the *discretization* error of the problem. Full multigrid attains this complexity on model problems [12, 6], but we will assume that all problems are being solved with a relative residual tolerance defined by the inexact Newton method described in §5.1.

4. Parallel Finite Element Architecture. This project is composed of three major components, which are described in the following sections: 1) the bone micro-FE modeling process that creates the meshes and post-processes the simulation results (§4.1), 2) the parallel finite element implementation—Olympus (§4.2), and 3) the parallel multigrid linear solver—Prometheus (§4.3).

4.1. Micro-FE Modeling of Bone. The first step of our simulation process is to construct the finite element meshes. Voxel based high-resolution finite element meshes of bone specimens are created in three steps:

1. The bone specimen is scanned using a micro-CT scanner at a resolution that varies between 13 and 90 microns depending on the anatomic site and size of the specimen. The resulting image is a three dimensional array of voxels with 1-byte grayscale values (0-black, 255-white).
2. The micro-CT data is loaded into an image-processing software (IDL v5.6, Research Systems Inc., Boulder, CO). This image is coarsened to an optimal resolution for convergence of element stresses using regional averaging to reduce the number of bone voxels to the desired level of discretization [32]. The image is thresholded to extract the bone phase resulting in a binary data set with bone (white) and space (black).
3. Finally, our custom code BOBCAT (Berkeley Orthopaedic Biomechanics Computational Analysis of Trabecular Bone) uses this binary image to generate an 8-node hexahedral finite element mesh in FEAP format (below), with displacement or force boundary conditions prescribed by the user. The final mesh file does not contain any material properties—which are supplied separately—and therefore allow the same input file to be used for various material models or types of analyses.

The resulting voxel based mesh is used with our parallel finite element application Olympus.

4.2. Olympus: parallel finite element application. The parallel finite element framework *Olympus* is composed of a parallelizing finite element code *Athena*. Athena uses a parallel graph partitioner (ParMetis [25]) to construct a sub-problem on each processor, for a serial finite element code *FEAP* [19]. Olympus uses a parallel finite element object *pFEAP*, which is a thin parallelizing layer for FEAP, that primarily maps vector and matrix quantities from between local FEAP problem and the global Olympus operator. Olympus uses the parallel algebraic multigrid linear solver *Prometheus*, which is built on the parallel numerical kernels library *PETSc* [5]. Olympus controls the solution process including an inexact Newton method and manages the database output (SILO from LLNL) to be read by a visualization application (eg, VISIT from LLNL). Figure 4.1 shows a schematic representation of this system.

The finite element input file is read in parallel by Athena, which uses ParMetis to partition the finite element graph. Athena generates a complete finite element problem on each processor from this partitioning. The processor sub-problems are designed so that each processor computes all rows of the stiffness matrix and entries of the residual vector associated with vertices that have been partitioned to the processors. This eliminates the need for communication in the finite element operator evaluation at the expense of a small amount of redundant computational work. Given this sub-problem and a small global text file with the material properties, FEAP runs on each processor much as it would in serial mode; in fact FEAP itself has no parallel constructs, but only interfaces with Olympus through the pFEAP layer.

Explicit message passing (MPI) is used for performance and portability and all parts of the algorithm have been parallelized for scalability. Clusters of symmetric multi-processors (SMPs) are the target platforms for this project. Faster communication within an SMP is implicitly exploited by first having Athena partition the problem onto the SMP compute nodes, and then recursively calling Athena to construct the processor subdomain problem on each node. This approach implicitly takes advantage of any increase in communication performance within the SMP, though the numerical kernels (in PETSc) are pure MPI.

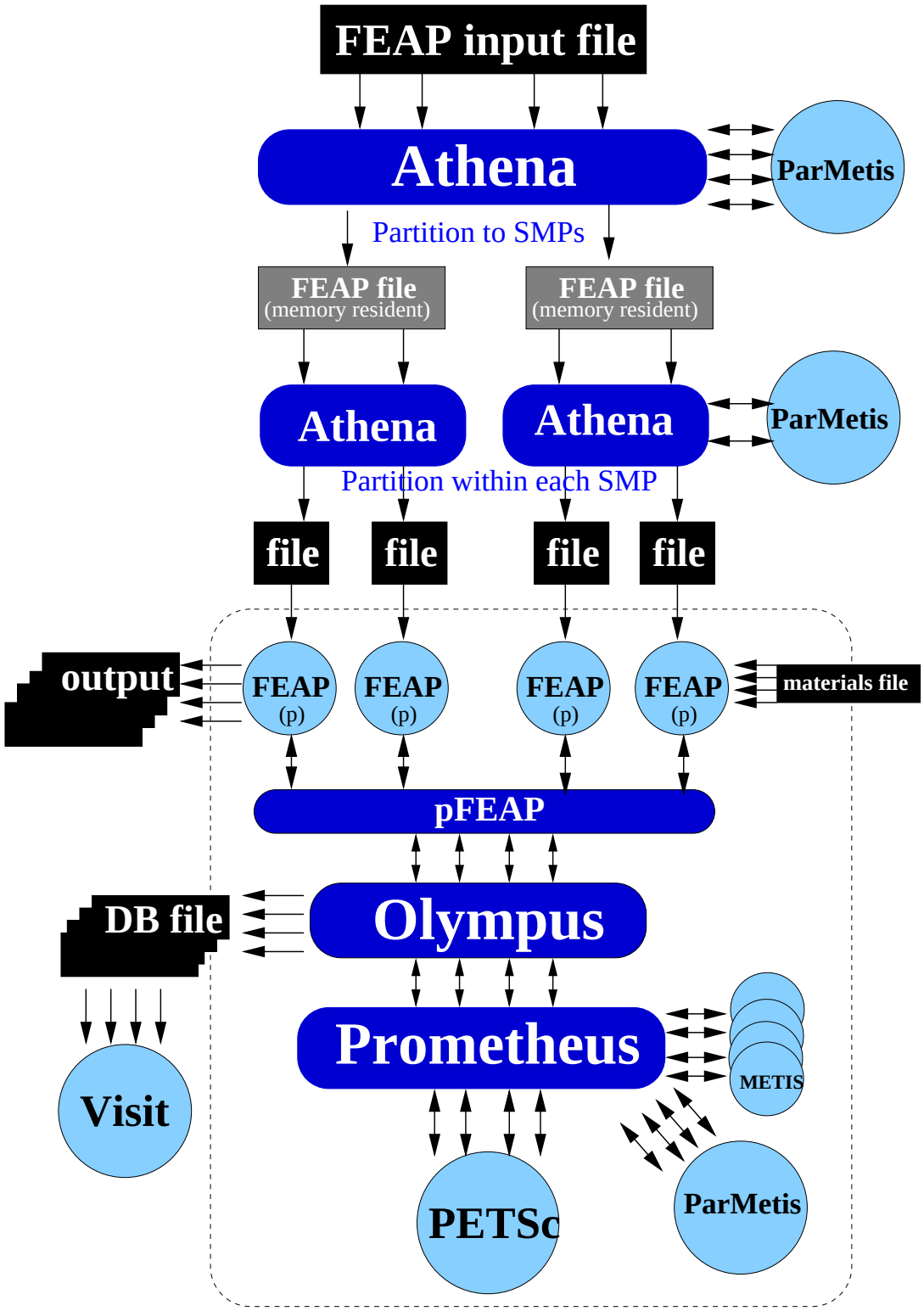

FIGURE 4.1. *Olympus parallel finite element application.*

4.3. Prometheus: parallel multigrid linear solver. The largest portion of the simulation time for our micro-FE bone problems is typically spent in the linear solve. We use the solver package *Prometheus* for this study. Prometheus is equipped with three multigrid algorithms, has both additive and (true) parallel multiplicative smoother preconditioners (general block and nodal block versions) [1], and several smoother iterators for the additive preconditioners (Chebyshev polynomials are used in this study [4]).

Prometheus has been optimized for ultrascalability, including two important features for complex problems using many processors. Prometheus repartitions the coarse grids to maintain load balance and the number of active processors, on the coarsest grids, is reduced to keep a minimum of about 500 equations per processor. The reasons for reducing the number of active processors are two fold: 1) it is difficult to implement the construction of the coarse grid spaces, in parallel, to have the exact serial semantics in the regions between processors (eg, it is difficult to implement the aggregation algorithms exactly in parallel) and 2) the computational time on the grids with very few equations per processor is dominated by communication and the coarsest grids can be solved as fast—or even faster—if fewer processors are used. Reducing the number of active processors on coarse grids is all but necessary for complex problems when using several thousand processors, and is even useful on a few hundred processors. Repartitioning the coarse grids is important for highly heterogeneous topologies because of severe load imbalances that can result from different coarsening rates in different domains of the problem. Repartitioning is also important on the coarsest grids when the number of processors are reduced because, in general, the processor domains become fragmented.

5. Numerical studies. This section investigates the performance of our project with geometrically nonlinear micro-FE bone analyses. The runtime cost of our analyses can be segregated into five primary sections, (listed with a typical percentage of the total run time in these experiments):

1. Athena parallel FE mesh partitioner (4%)
2. Linear solver *mesh* setup (construction of the coarse grids) (1%)
3. FEAP element residual, tangent, and stress calculations (13%)
4. Solver *matrix* setup (coarse grid operator construction) (43%)
5. The solve for the solution (37%)

Additionally, about 2% of the total run time is not directly measured and is added to the FEAP time (3), in Figure 5.3 (right). This unmeasured time is primarily accounted for by the time for FEAP to read the input file on each processor and other ancillary work in FEAP.

Although the initial setup in Athena (1), and to a lesser extent the mesh setup cost in Prometheus (2), represents a significant portion of the time for one linear solve, these costs are amortized in nonlinear simulations—but they must be fully parallel for the degree of parallelism required for this project. The mesh setup costs in Prometheus (2) can be significant on problems that converge quickly and remeshing prevents the cost from being amortized, but is often an insignificant part of the simulation time. The matrix setup (4) costs are significant because they can not be amortized in a full Newton solution scheme. But these matrix setup costs can be amortized in a secant Newton method. An inexact full Newton method is used for the nonlinear analysis herein; this results in the matrix setup costs (4) and solve phase (5), and to a lesser extent the element residual and tangent calculation (3), dominating the solution time.

5.1. Nonlinear solution algorithm. An inexact Newton method is a generalization of Newton's method for solving $F(x) = 0$.

Figure 5.1, sketches our inexact Newton method. Where τ_r the given convergence criteria.

given $x_0 = 0$
for $k = 1, 2, \ldots$
  Find **some** $\eta_k \in [0, 1)$
  Solve $F'(x_{k-1})s_k = -F(x_{k-1})$ for s_k such that $\|F(x_{k-1}) + F'(x_{k-1})s_k\|_2 < \eta_k \|F(x_{k-1})\|_2$
  $x_k \leftarrow x_{k-1} + s_k$
  if $\sqrt{s_k^T F(x_{k-1})} < \tau_r \sqrt{s_1^T F(x_0)}$ **return** x_k

FIGURE 5.1. *Inexact Newton iterations*

The linear solve for each Newton iteration is performed using our parallel AMG solver Prometheus with

a tolerance of η_k. When an iterative solver is used for solving the linearized system $F'(x_{k-1})s_k = -F(x_{k-1})$ in the k^{th} iteration of Newton's algorithm, the tolerance, or "forcing term" with which the system is solved, is critical in optimizing the cost of the analysis. The selection of η_k in governed by several factors. This discussion is adapted from Eisenstat and Walker [18], and Kelly [27].

First, it is not efficient to solve the system to machine precision, especially when the linearization is poor. This *over-solve* of the problem can be ameliorated with the use of $\eta_k^A = \gamma \|F(x_k)\|_2 / \|F(x_{k-1})\|_2$, where $\gamma \in (0, 1]$ is a parameter. To insure that some solution is generated at each Newton step (ie, $\eta_k < 1.0$), η_{max} is defined as the upper bound on the linear solver relative tolerance. If the residual drops far in an early iteration but the residual is still high, then η_k^A can result in a over-solve of the system: $\eta_k^B = \gamma \eta_{k-1}^2$ is used to prevent this. Finally, if the nonlinear residual is closer to the solution (ie, termination of the Newton iteration) than η_k, then the solution may be too accurate (ie, an over-solve resulting in a more accurate and hence more expensive solution than requested). Given τ_r, the relative tolerance used in the convergence criteria for the Newton iteration, $\eta_k^C = .5\tau_r \sqrt{s_{k-1} F(x_k)}$ is used for a lower bound on η_k. Note, this form of η_k^C, derived from Kelly (pg. 105, [27]), is not consistent because our convergence criteria (in Figure 5.1) is not the same as that used by Kelly ($\|F(x_k)\|_2 < \tau_r \|F(x_0)\|_2$), but we have observed that it works reasonably well at preventing over-solves. Finally, the number of linear solve iterations is limited by a constant M, because the cost of reforming the matrix—FEAP (3) and the matrix setup (4)—is small enough that is better to relinearize than spend too much effort solving the linearized system far from the point of linearization.

We compute a relative linear solver tolerance, or forcing term η_k in step k of the inexact Newton process, by starting with $\eta_0 = \eta_{max}$. For $k = 1, 2, \ldots$, if $\gamma \eta_{k-1}^2 > .1$, then set $\eta_k = max(\eta_k^A, \gamma \eta_{k-1}^2)$, otherwise set $\eta_k = \eta_k^A$. If this η_k is lager than η_{max} then η_k is set to η_{max}. Likewise, if this η_k is less than η_k^C then η_k is set to η_k^C. This study uses $\gamma = .9$, $\eta_{max} = 1 \cdot 10^{-2}$, $\tau_r = 1 \cdot 10^{-6}$ and $M = 70$.

5.2. Micro-FE Problem. We consider finite element models of a thoracic vertebral body (T-10) at several resolutions. This vertebral body was taken from an 82-year-old female cadaver and was scanned at 30 micron spatial resolution using a micro-CT scanner (μCT80, SCANCO Medical AG, Bassersdorf, Switzerland). After endplates are removed, uniform displacement boundary conditions are applied (Figure 5.2 left). The material model is a large deformation neo-Hookean elastic material. The purpose of such a simulation is to obtain stress distributions in the human vertebral body [8] as well as to compare the elastic properties of the micro-FE model with laboratory mechanical testing data. Figure 5.2 (right) shows a 1 mm vertical slice of this vertebral body, with over 85% porosity, and illustrates the complexity of the domains of these micro-FE models.

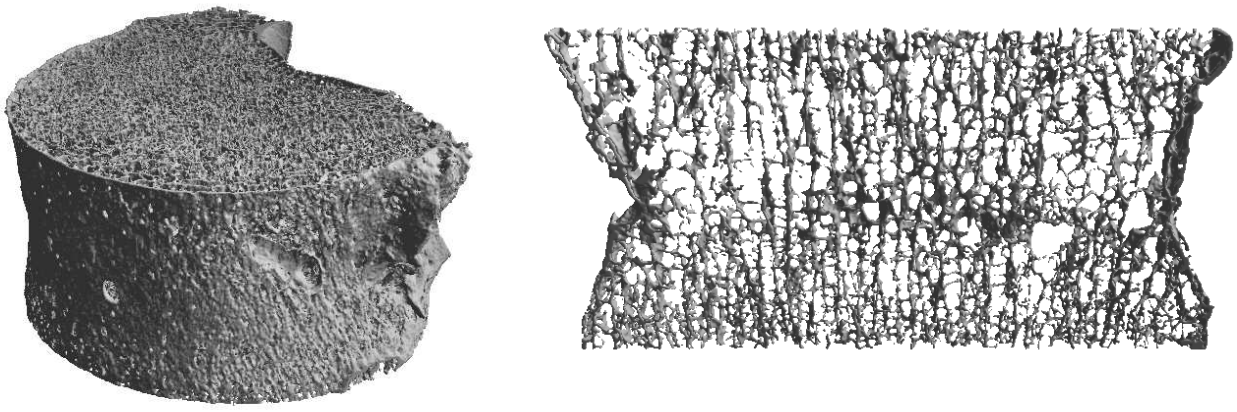

FIGURE 5.2. *Vertebral body with endplates removed (left). One millimeter thick cross-section of vertebral body at 30 μm resolution shows the complexity of the micro-FE mesh (right).*

5.3. Experimental procedures. Understanding the scalability characteristics of a highly optimal, unstructured, implicit parallel finite element code, such as our project, is complex. Our multigrid solver and parallel finite element code are not deterministic (ie, Prometheus and ParMetis are not deterministic). Additionally, the IBM SPs exhibit non-deterministic performance behavior—this is well understood in the IBM user community. Ideal scientific experiments are all but impossible under these circumstances, but we attempt to conduct valid and useful numerical experiments by running each case several times and selecting the best case. This is not ideal, but a careful statistical approach is not practical and limited machine resources make the largest problem difficult to run often, and it was in fact run only once for each study below. However these largest experiments were run during dedicated access time on the ASCI White machine at Lawrence Livermore National Laboratory (LLNL). Additionally, we were careful to run the smallest version of this problem many times to try to insure that our base case is not artificially poor (and thus making the parallel efficiency look deceptively good).

5.4. Scaled speedup. This section examines the scalability of the solver via scaled speedup studies. Scaled speedup is the measure of pertinent performance quantities such as solve time, flop rate and iteration counts, on several discretizations of a given problem with increasing resolution. These meshes can be constructed by uniformly refining an initial mesh or, as in our case, by generating meshes at varying resolutions, starting with the micro-computed tomography data at 30 micron resolution. Performance is measured in numerical experiments with the number of processors selected so as to place approximately the same number of equations per processor.

There are several advantages to using scaled speedup. First, scaled speedup maintains approximately the same pressure on the cache—in all experiments—so that parallel inefficiencies are not hidden by cache effects. Second, the (constant) subdomain size per processor can be selected to use the machine efficiently, with respect to memory requirements, communication performance characteristics and constraints of turnaround time, to provide useful data about the time costs of the application on a particular machine. See Sun for a discussion of models for parallel speedup [36]. Note, that construction of good scaled speedup experiments can be difficult; in our case, the domain of the PDE changes with different resolutions and thus, the mathematical problems are not identical at each resolution.

This study uses a preconditioned conjugate gradient solver—preconditioned with one multigrid V-cycle. The smoothed aggregation multigrid method, described in §3, is used with second order Chebyshev smoother [4].

All computational experiments were conducted on the ASCI White IBM SP at LLNL, with 16 375Mz Power3 processors per node, 16GB memory per node, and a theoretical peak flop rate of 24 Gflop/sec per node. This paper focuses on scalability, but we achieve about 7.5% of this theoretical peak — see Adams [3], for a detailed investigation of the performance of one linear solve on one node.

Six meshes of this vertebral body are used with discretization scales (h) ranging from 150 microns to 30 microns. Pertinent mesh quantities are shown in Table 5.1. The bone tissue is modeled as a finite deformation (neo-Hookean) isotropic elastic material with a Poisson's ratio of 0.3. All elements are geometrically identical trilinear hexahedra (8-node bricks). The models are analyzed with displacement control, in six load ("time") steps, to a strain of 3% in compression.

h (μm)	150	120	80	60	40	30
# degrees of freedom (10^6)	7.4	13.5	37.6	77.9	237	537
# of elements (10^6)	1.08	2.12	7.12	16.9	57	135
# of compute nodes used (§5.4.1)	4	7	21	43	130	292
# of compute nodes used (§5.4.3)	3	5	15	32	96	217

TABLE 5.1
Scaled vertebral body mesh quantities.

5.4.1. Scalability study with 131K degrees of freedom per processor. This section presents a scaled speedup study that reflects a usage of the machine to minimize turnaround time—that is, use as much of the machine as possible on the largest problem. Only 14 of the 16 processors on each node are used

because our application runs slower if all 16 processors are used on the larger test cases (due to a known bug in the IBM operating system). The number of compute nodes is selected to place about 131K degrees of freedom (dof) per processor.

Table 5.2 shows the iteration counts for each linear solve, in the six load steps, for the smallest and largest version of the vertebral body. This data show that the linear solver is scaling perfectly in that the

	smallest version (7.4M dof)					largest version (537M dof)					
Newton step	1	2	3	4	5	1	2	3	4	5	6
Load step 1	5	14	20	21	18	5	11	35	25	70	2
Load step 2	5	14	20	21	20	5	11	36	26	70	2
Load step 3	5	14	20	22	19	5	11	36	26	70	2
Load step 4	5	14	20	22	19	5	11	36	26	70	2
Load step 5	5	14	20	22	19	5	11	36	26	70	2
Load step 6	5	14	20	22	19	5	11	36	25	70	2

TABLE 5.2
Linear solver iterations.

number of iterations for the first linear solve in each Newton loop is a constant (5), but that the nonlinear problems are becoming more challenging as the mesh is refined. The number of Newton iterations required to solve the nonlinear problems is increasing—this indicates that the nonlinear problem is getting more challenging as the mesh is refined.

Figure 5.3 (left) shows the time costs that are amortized in this study—the partitioning of Athena (1) and the mesh setup (2), and are hence insignificant in this study but could be significant under other circumstances (eg, if only one linear solve was performed). The Athena time does grow as more processors are used, but the partitioning process in Athena is difficult to scale perfectly because it is a purely parallel process (ie, there is no real work done in a one processor run) and, in fact, very few applications partition the mesh in parallel (and thus have no parallel efficiency whatsoever).

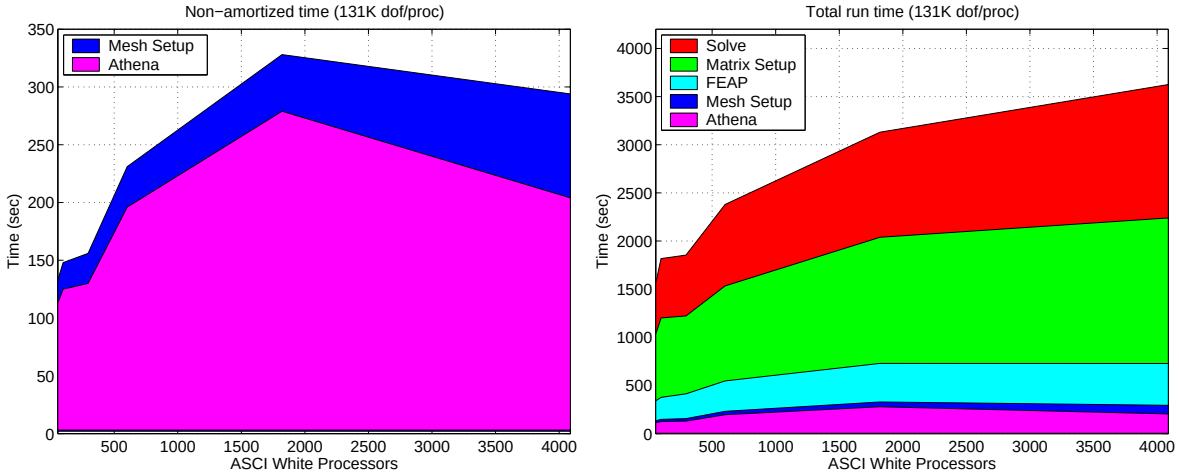

FIGURE 5.3. *Total end-to-end solution times - 131K dof per processor*

Figure 5.3 (right) shows the total end-to-end run times, as well as the times for the major components of the solution time as listed above. The growth in the solution phases is due to four main factors: 1) increase the number of Newton iterations, 2) increase in the average number of linear solve per Newton iteration, 3) the matrix triple product (RAP), and 4) growth in the number of flops per iteration. These first two causes are due to the nonlinear problems becoming more challenging as the mesh is refined, and hence requiring more Newton iterations and linear solve iterations. There are at least two potential sources of this growth: 1) the models do get softer as the mesh is refined, leading to potentially more geometric nonlinearities, and

2) the micro-FE modeling process does not produce perfect meshes and the larger meshes could have locally somewhat unstable structures within them. These difficulties could potentially be ameliorated with grid sequencing (ie, using the solution from the coarser problem as an initial guess), but we have not investigated this. The last two causes of growth in the solution phases, the RAP and the increase in flops per iteration per node are discussed below.

Figure 5.4 (left) shows the sustained average flop rate per processor for the entire analysis and the for the matrix setup and solve phases. The global flop rates are measured with the IBM Power3 hardware performance monitors, accessed with the "hpm_count" utility, and the flop rates for the coarse grid construction (RAP) and solve phase are measured with PETSc's timing functions which in turn use the "rs6000_time" function. The aggregate flop rate for the largest problem reaches a sustained rate of .47 Teraflops/sec on 4088 processors. Perfect parallel efficiency results in flat plots for this data (Figure 5.4 left) and the data in Figure 5.3.

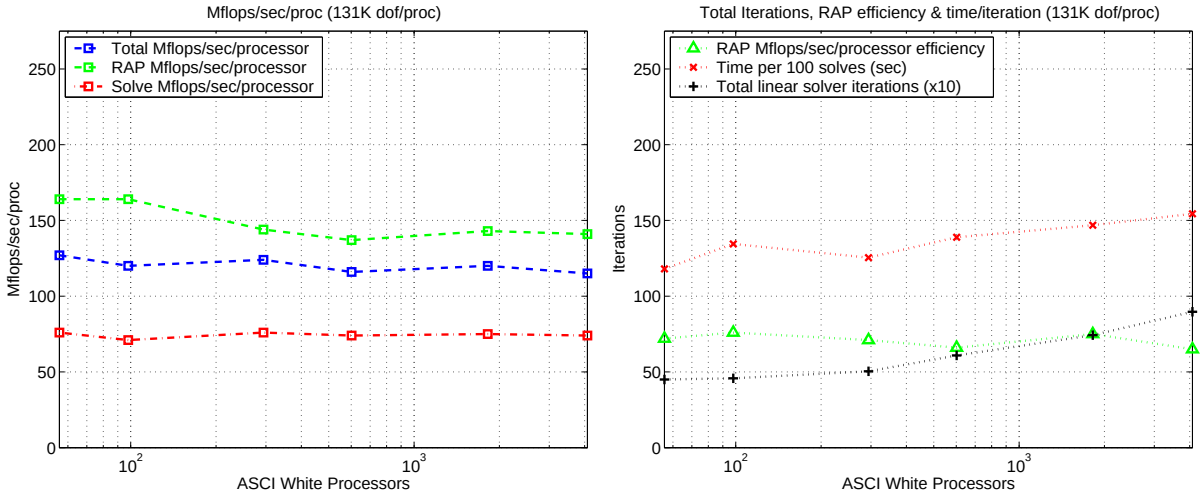

FIGURE 5.4. *Solver iteration counts, average megaflop rates per processor and RAP flop rate efficiency.*

Figure 5.4 (right) shows the total number of iteration in the linear solver, the time per linear solver iteration and the flop rate load imbalance of the RAP (defined as the percentage of the average processor flop rate of the processor with the minimum flop rate). The total number of linear solver iterations does increase, due to the nonlinear problem becoming more challenging, as discussed above, but the average flop rate per processor is scaling reasonably well, with a total parallel flop rate efficiency of about 90%. Additionally, the parallel flop rate efficiency for the solve phase is essentially 100%. Thus, all of the flop rate inefficiency is due to the matrix triple product (RAP).

The RAP flop rate load imbalance, shown in Figure 5.4 (right), is about 75% — in this set of data it is relatively constant, though more degradation is observed due to better efficiency on the smallest problem when there are more degrees of freedom per processor (§5.4.3). There are two primary sources of this inefficiency: 1) the fine grids are well balanced by Athena for matrix vector products, and the coarse grids are rebalanced by Prometheus, but the matrices are not explicitly balanced for the RAP, and 2) from the relatively complex communication patterns of the coarse grid matrix entries (probably exacerbated by the coarse grid repartitioning). Load balancing for the RAP is challenging problem for which we do not have an ideal solution.

The final source of growth in the solution time on larger problems is an increase in the number of flops per iteration per fine grid node. Figure 5.4 (left) show perfect flop rate efficiency for the solve phase, but an increase in the time per iteration of about 20% (Figure 5.4 right). This is due to an increase in the average number non-zeros per row which in turn is due to the large ratio of volume to surface area in the vertebral body problems. These vertebral bodies have a large amount of surface area and, thus, the low resolution mesh (150 μm) has a large ratio of surface nodes to interior nodes. As the mesh is refined the ratio of interior nodes to surface nodes increases, resulting in more non-zeros per row—from 50 on the smallest version to 68

on the largest. Additionally, the complexity of the construction of the coarse grids in smoothed aggregation (the RAP) has the tendency to increase in fully 3D problems. Thus, as this problem is refined, the meshes become more fully 3D in nature—resulting in grids with more non-zeros per row and higher complexities in the RAP—this would not be the case with a fully 3D problem like a cube.

This observation suggests that a better design for scalability studies would be to place the same number of non-zeros per processor instead of the same number of equations—this would increase the number of processors used on the largest problems (or commensurately, reduce the number of processors used on the smaller versions), make the time plots more flat. We, however, use the definition of scaled speedup that is traditional in the linear solver community. Note, the computational mechanics community generally uses the number of elements as the bases for scaled speedup studies—the average number of elements per processor in this study goes from about 19.3K on the smallest problem to about 33.0K on the largest.

5.4.2. Speedup study. This section investigates a (non-scaled) speedup study of the smallest—7.4M degrees of freedom—version of the vertebral body from the previous section. Non-scaled speedup measures the performance of one problem run on many sets of processors and provides means of finding the limit of scalability of the software and hardware. However, effects from pressure on the memory hierarchy, such as paging, in the base case can lead to inflated scalability results (ie, super-linear speedups are possible and an upper bound on scalability is not well defined). Note, this 7.4M dof problem avoids paging on one node but does use essentially all of the 16GB of main memory of the Power3 node. 15 of the 16 processors on the Power3 nodes of the ASCI Frost and White machines at LLNL are used. One linear solve with a relative residual tolerance of $1.0 \cdot 10^{-6}$ is performed on one to 128 nodes. Table 5.3 shows the number of iterations in this linear solve for all cases.

Power3 Nodes	1	2	4	8	16	32	64	128
Number of iterations	39	40	38	45	46	42	36	40

TABLE 5.3
Linear solve iterations.

This shows some variance in the number of iterations, but is reasonably constant throughout this range of processors and, thus, the parallel implementation has effectively maintained smoothed aggregation's semantics. Figure 5.5 (left) shows the total flop rate for the solve phase with an efficiency of 92% on 32 nodes and 78% on 64 nodes, but only 41% efficiency on 128 nodes. See Adams [3], for a more detailed investigation of the performance of one linear solve on one node (ie, efficiency from one to 16 processors).

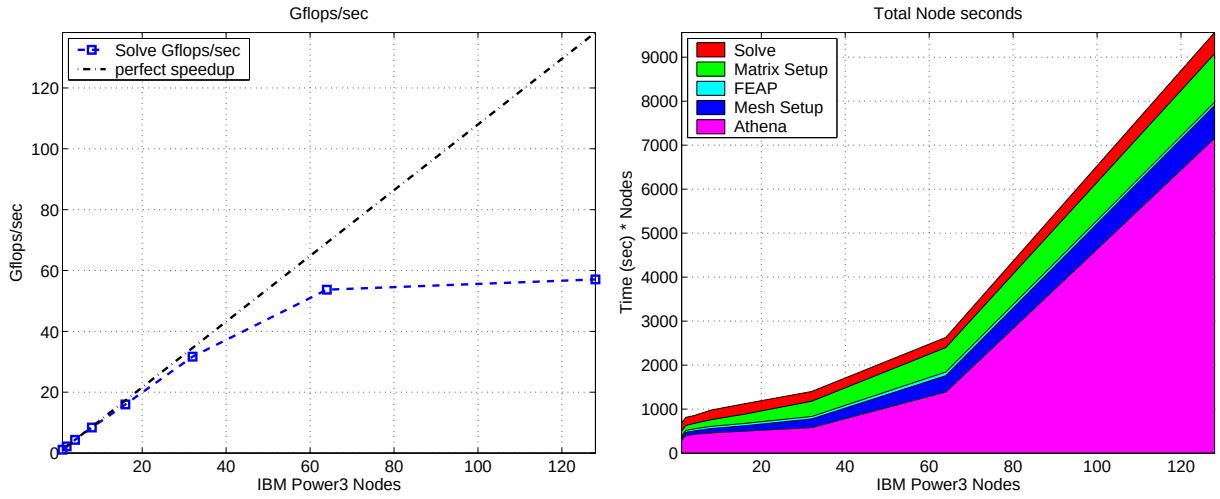

FIGURE 5.5. *Speedup with 7.4M degrees of freedom problem*

Figure 5.5 (right) shows the total end-to-end run time costs (Power3 node seconds), as well as the times

for the major components of the solution time as listed above. This show that most of the inefficiency arises from the initial partitioning phase and the whole system is very efficient up to 32 or 64 nodes. This is another demonstration of outstanding scalability.

5.4.3. Scaled speedup with 164K degrees of freedom per processor. This section presents a second scaled speedup study using less compute nodes and hence more degrees of freedom per processor—about 164K dof per processor. This study represents a usage model that is more optimal for the efficient use of the machine, as opposed to minimizing the turn around time as in the previous section. Using fewer processors results in better parallel efficiency, due to the larger processor subdomains and hence more available parallelism. These experiments uses 15 of the 16 processors per node of the ASCI White IBM SP Power3 at LLNL. Three nodes are used for the base case and 217 nodes are used for the largest problem.

Figure 5.6 show that total solve time, decompose into the primary categories discussed above.

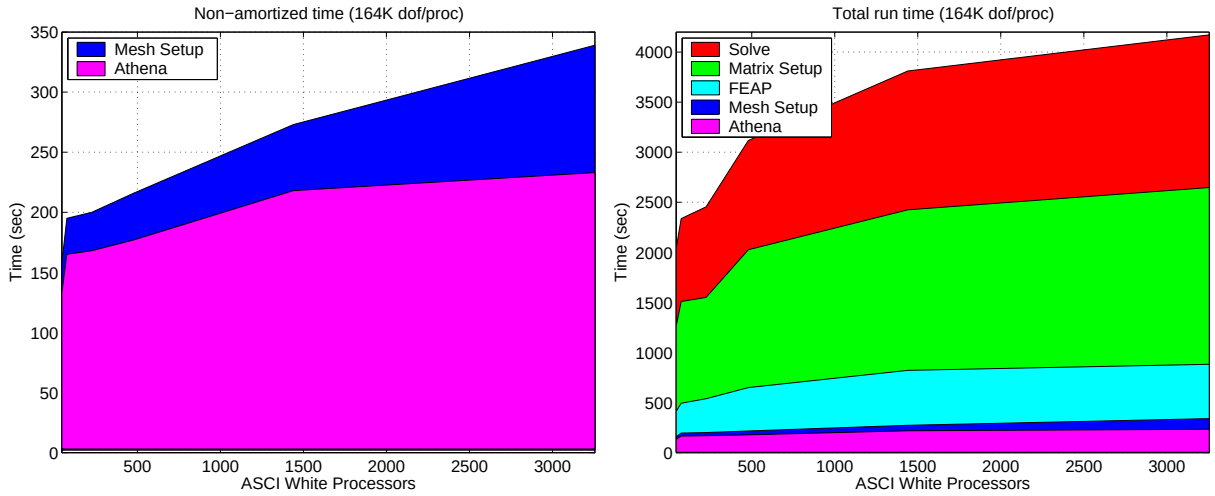

FIGURE 5.6. *Total end-to-end solution times - 164K dof per processor*

This data shows similar results as those in §5.4.1, with slightly better parallel efficiencies—due to the larger processor subdomains and fewer processors.

Figure 5.7 shows various efficiency quantities (like Figure 5.4).

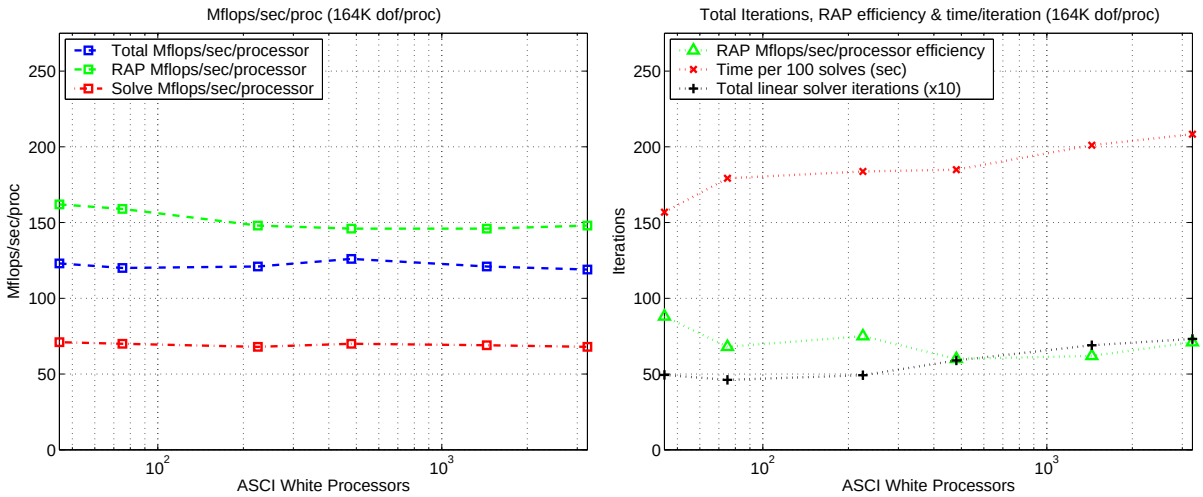

FIGURE 5.7. *Solver iteration counts, average megaflop rates per processor and RAP flop rate efficiency.*

Again, this data shows similar results as those in §5.4.1, with slightly better parallel efficiency, especially in the RAP efficiency in the smallest case, due to the larger processor subdomains.

6. Conclusions. We have demonstrated that a mathematically optimal algebraic multigrid method (smoothed aggregation) is computationally effective for large deformation finite element analysis of solid mechanics problems with up to 537 million degrees of freedom. We have achieved a sustained flop rate of almost one half a Teraflop/sec on 4088 IBM Power3 processors (ASCI White). These are the largest analyses of unstructured elasticity problems with complex geometry that we are aware of, with an average time per linear solve of about 1 and a half minutes. This compares favorably to the closest published results to this work—that of a 115 million degree of freedom linear elasticity solve in 7 minutes (which also used ASCI White) [28]. Additionally, this work is significant in that no special purpose algorithms or implementations were required to achieve a highly scalable performance on a common parallel computer. The only limits on the size of problems that we can solve with Olympus are the construction of the FE meshes and the size of available computers—there are no limits in the foreseeable future of this software on parallel machines with performance characteristics similar to that of the IBM SPs.

This project demonstrates absolute state-of-the-art linear solver technology for unstructured solid mechanics problems—*sui generis*. This project also possesses a fully parallel finite element partitioner that provides access to enough memory (in parallel) to use a high quality mesh partitioner (ParMETIS). We, however, by no means have a complete ultrascalable finite element system. Future work on Olympus will entail pushing the ultrascalability back to the mesh generation and forward into the data processing and visualization, and broaden the domain of finite element problems that can be addressed. Future work on Prometheus involves the application of this core linear solver technology to a wider variety of engineering and science applications.

Acknowledgments. We would like to acknowledge the many people that have contributed libraries to this work: R.L. Taylor for providing FEAP, the PETSc team for providing PETSc, George Karypis for providing ParMetis. We would also like to thank Livermore National Laboratory for providing access to its computing systems and to the staff of Livermore Computing for their support. We would like to thank Jeff Fier for help with performance advise on the IBM SPs. This research was supported in part by the Department of Energy under DOE Contract No. W-7405-ENG-48 and DOE Grant Nos. DE-FG03-94ER25217 and DE-FC02-01ER25479 and the National Institutes of Health, Grants NIH-AR49570 and NIH-AR43784. Human cadaveric tissue was obtained through the National Disease Research Interchange (NDRI). We would also like to thank Dr. Michael Liebschner for specimen imaging. Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy under contract DE-AC04-94AL85000.

REFERENCES

[1] M. F. Adams, *A distributed memory unstructured Gauss–Seidel algorithm for multigrid smoothers*, in ACM/IEEE Proceedings of SC2001: High Performance Networking and Computing, Denver, Colorado, November 2001.

[2] ———, *Evaluation of three unstructured multigrid methods on 3D finite element problems in solid mechanics*, International Journal for Numerical Methods in Engineering, 55 (2002), pp. 519–534.

[3] M. F. Adams, H. Bayraktar, T. Keaveny, and P. Papadopoulos, *Applications of algebraic multigrid to large-scale finite element analysis of whole bone micro-mechanics on the IBM SP*, in ACM/IEEE Proceedings of SC2003: High Performance Networking and Computing, 2003.

[4] M. F. Adams, M. Brezina, J. J. Hu, and R. S. Tuminaro, *Parallel multigrid smoothing: polynomial versus Gauss–Seidel*, J. Comp. Phys., 188 (2003), pp. 593–610.

[5] S. Balay, W. D. Gropp, L. C. McInnes, and B. F. Smith, *PETSc 2.0 users manual*, tech. report, Argonne National Laboratory, 1996.

[6] R. Bank and T. Dupont, *An optimal order process for solving finite element equations*, Math. Comp., 36 (1981), pp. 35–51.

[7] H. H. Bayraktar, M. F. Adams, A. Gupta, P. Papadopoulos, and T. M. Keaveny, *The role of large deformations in trabecular bone mechanical behavior*, in ASME Bioengineering Conference, Key Biscayne, FL, 2003, pp. 31–32.

[8] H. H. Bayraktar, J. M. Buckley, M. F. Adams, A. Gupta, P. F. Hoffmann, D. C. Lee, P. Papadopoulos, and T. M. Keaveny, *Cortical shell thickness and its contribution to vertebral body stiffness*, in ASME Bioengineering Conference, Key Biscayne, FL, 2003, pp. 115–116.

[9] H. H. Bayraktar, A. Gupta, R. Kwon, P. Papadopoulos, and T. M. Keaveny, *The modified super-ellipsoid yield criterion for human trabecular bone*, Journal of Biomechanical Engineering, In Press (2004).

[10] H. H. Bayraktar, E. F. Morgan, G. L. Niebur, G. Morris, E. Wong, and T. M. Keaveny, *Comparison of the elastic and yield properties of human femoral trabecular and cortical bone tissue*, Journal of Biomechanics, 37 (2004), pp. 27–35.
[11] D. Braess, *On the combination of the multigrid method and conjugate gradients*, in Multigrid Methods II, W. Hackbusch and U. Trottenberg, eds., Berlin, 1986, Springer–Verlag, pp. 52–64.
[12] J. Bramble, *Multigrid methods*, Longman Scientific and Technical, 1993.
[13] A. Brandt, *Multi-level adaptive solutions to boundary value problems*, Math. Comput., 31 (1977), pp. 333–390.
[14] W. L. Briggs, V. E. Henson, and S. McCormick, *A multigrid tutorial, Second Edition*, SIAM, Philadelphia, 2000.
[15] V. E. Bulgakov and G. Kuhn, *High-performance multilevel iterative aggregation solver for large finite-element structural analysis problems*, International Journal for Numerical Methods in Engineering, 38 (1995), pp. 3529–3544.
[16] S. C. Cowin, ed., *Bone Mechanics Handbook*, CRC Press, Boca Raton, 2 ed., 2001.
[17] C. C. Douglas, *Multi–grid algorithms with applications to elliptic boundary–value problems*, SIAM J. Numer. Anal., 21 (1984), pp. 236–254.
[18] S. C. Eisenstat and H. F. Walker, *Choosing the forcing terms in an inexact Newton method*, SIAM J. Sci. Comput., 17 (1996), pp. 16–32.
[19] *FEAP*. www.ce.berkeley.edu/∼rlt.
[20] S. Fortune and J. Wyllie, *Parallelism in random access machines*, in ACM Symp. on Theory of Computing, 1978, pp. 114–118.
[21] R. Guldberg, S. Hollister, and G. Charras, *The accuracy of digital image-based finite element models.*, Journal of Biomechanical Engineering, 120 (1998), pp. 289–295.
[22] W. Hackbusch, *Multigrid methods and applications*, vol. 4 of Computational Mathematics, Springer–Verlag, Berlin, 1985.
[23] J. Homminga, B. Van-Rietbergen, E. M. Lochmuller, H. Weinans, F. Eckstein, and R. Huiskes, *The osteoporotic vertebral structure is well adapted to the loads of daily life, but not to infrequent "error" loads*, Bone, 34 (2004), pp. 510–516.
[24] T. J. R. Hughes, R. M. Ferencz, and J. O. Hallquist, *Large-scale vectorized implicit calculation in solid mechanics on a cray x-mp/48 utilizing ebe preconditioned conjugate gradients*, Computer Methods in Applied Mechanics and Engineering, 61 (1987), pp. 215–248.
[25] G. Karypis and V. Kumar, *Parallel multilevel k-way partitioning scheme for irregular graphs*, ACM/IEEE Proceedings of SC1996: High Performance Networking and Computing, (1996).
[26] T. M. Keaveny, *Bone Mechanics Handbook*, CRC Press, Boca Raton, 2 ed., 2001, ch. 16, pp. 1–42.
[27] C. Kelly, *Iterative Methods for Linear and Nonlinear Equations*, Frontiers in Applied Mathematics, SIAM, Philadelphia, 1995.
[28] M. M. Bhardwaj, K. Pierson, G. Reese, T. Walsh, D. Day, K. Alvin, and J. Peery, *Salinas: A scalable software for high-performance structural and solid mechanics simulation*, in ACM/IEEE Proceedings of SC2002: High Performance Networking and Computing, 2002. Gordon Bell Award.
[29] E. F. Morgan, H. H. Bayraktar, O. C. Yeh, S. Majumdar, A. Burghardt, and T. M. Keaveny, *Contribution of inter-site variations in architecture to trabecular bone apparent yield strain*, Journal of Biomechanics, In Press (2004).
[30] G. L. Niebur, M. J. Feldstein, and T. M. Keaveny, *Biaxial failure behavior of bovine tibial trabecular bone*, Journal of Biomechanical Engineering, 124 (2002), pp. 699–705.
[31] G. L. Niebur, M. J. Feldstein, J. C. Yuen, T. J. Chen, and T. M. Keaveny, *High-resolution finite element models with tissue strength asymmetry accurately predict failure of trabecular bone*, Journal of Biomechanics, 33 (2000), pp. 1575–1583.
[32] G. L. Niebur, J. C. Yuen, A. C. Hsia, and T. M. Keaveny, *Convergence behavior of high-resolution finite element models of trabecular bone*, Journal of Biomechanical Engineering, 121 (1999), pp. 629–635.
[33] G. Poole, Y. Liu, and J. Mandel, *Advancing analysis capabilities in ANSYS through solver technology*, Electronic Transactions in Numerical Analysis, 15 (2003), pp. 106–121.
[34] P. Rüegsegger, B. Koller, and R. Müller, *A microtomographic system for the nondestructive evaluation of bone architecture*, Calcified Tissue International, 58 (1996), pp. 24–29.
[35] J. S. Stolken and J. H. Kinney, *On the importance of geometric nonlinearity in finite-element simulations of trabecular bone failure*, Bone, 33 (2003), pp. 494–504.
[36] X. Sun and L. Hi, *Scalable problems and memory-bounded speedup*, J. of Parallel and Distibuted Comp., 19 (1993), pp. 27–37.
[37] U. Trottenberg, C. Oosterlee, and A. Schüller, *Multigrid*, Academic Press, London, 2001.
[38] D. Ulrich, B. Van Rietbergen, A. Laib, and P. Rüegsegger, *The ability of three-dimensional structural indices to reflect mechanical aspects of trabecular bone.*, Bone, 25 (1999), pp. 55–60.
[39] D. Ulrich, B. Van Rietbergen, H. Weinans, and P. Ruegsegger, *Finite element analysis of trabecular bone structure: a comparison of image-based meshing techniques*, Journal of Biomechanics, 31 (1998), pp. 1187–1192.
[40] B. Van Rietbergen, R. Huiskes, F. Eckstein, and P. Ruegsegger, *Trabecular bone tissue strains in the healthy and osteoporotic human femur*, Journal of Bone Mineral Research, 18 (2003), pp. 1781–1788.
[41] B. Van Rietbergen, H. Weinans, R. Huiskes, and A. Odgaard, *A new method to determine trabecular bone elastic properties and loading using micromechanical finite element models*, Journal of Biomechanics, 28 (1995), pp. 69–81.
[42] P. Vaněk, M. Brezina, and J. Mandel, *Convergence of algebraic multigrid based on smoothed aggregation*, Numerische Mathematik, 88 (2001), pp. 559–579.
[43] P. Vaněk, J. Mandel, and M. Brezina, *Algebraic multigrid by smoothed aggregation for second and fourth order elliptic problems*, in 7th Copper Mountain Conference on Multigrid Methods, 1995.