# APPLICATIONS OF ALGEBRAIC MULTIGRID TO LARGE-SCALE FINITE ELEMENT ANALYSIS OF WHOLE BONE MICRO-MECHANICS ON THE IBM SP

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

**Abstract.** Accurate micro-finite element analyses of whole bones require the solution of large sets of algebraic equations. Multigrid has proven to be an effective approach to the design of highly scalable linear solvers for solid mechanics problems. We present some of the first applications of scalable linear solvers, on massively parallel computers, to whole vertebral body structural analysis. We analyze the performance of our algebraic multigrid (AMG) methods on problems with over 237 million degrees of freedom on IBM SP parallel computers. We demonstrate excellent parallel scalability, both in the algorithms and the implementations, and analyze the nodal performance of the important AMG kernels on the IBM Power3 and Power4 architectures.

**Key words.** multigrid, trabecular bone, human vertebral body, finite element method, massively parallel computing.

**1. Introduction.** This paper presents applications of optimal linear solver methods to large-scale trabecular bone finite element (FE) modeling problems on massively parallel computers. 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 very complicated 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).

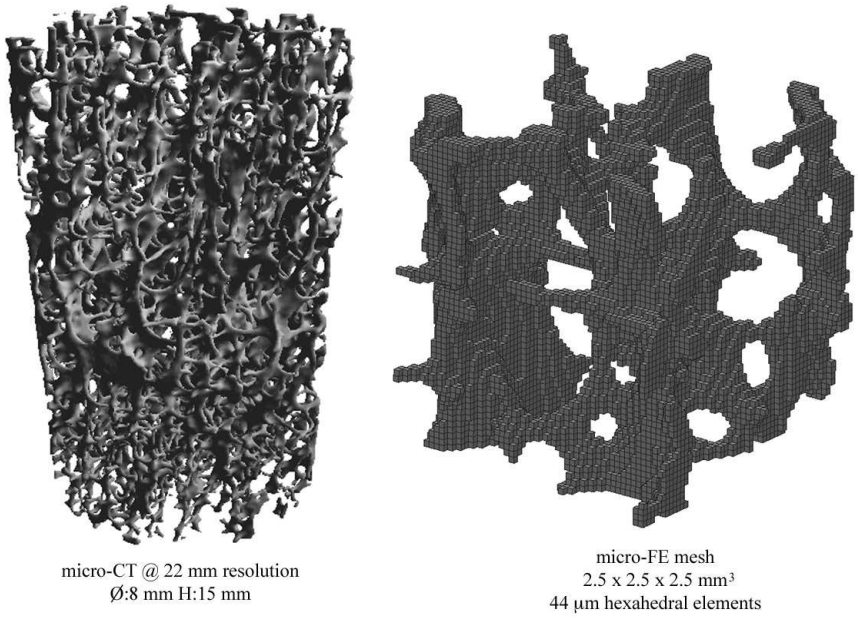


FIGURE 1.1. *Rendering of a human vertebral bone cylindrical specimen and detail showing voxel finite elements*

[1]Sandia National Laboratories, PO Box 969, MS 9217, Livermore, CA 94551 (mfadams@ca.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



[0]SC'03, November 15-21, 2003, Phoenix, Arizona, USA, Copyright 2003 ACM 1-58113-695-1/03/0011...$5.00

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 57 million elements with a resolution of 40 microns.

High-resolution finite element models of trabecular bone have traditionally been solved with the one level element-by-element (EBE) preconditioned conjugate gradient (CG) method [20, 39, 38, 26, 8]. While the EBE-PCG solver is very memory efficient (the stiffness matrix need not be explicitly constructed), and the work per iteration, 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.

Multigrid methods are well known to be theoretically scalable for $H^1$-elliptic operators, both scalar problems like Poisson's equation and systems of PDEs like displacement finite element discretizations of elasticity [9], and are hence a natural choice for solving these micro-FE bone problems. Typically, one multigrid cycle is used as a preconditioner for a conjugate gradient solver. Multigrid has been applied to structured grid problems for decades [10]. In the past ten years, multigrid methods have been developed for unstructured problems as well. One such method, that has proven to be well suited to elasticity problems, is smoothed aggregation [41, 2, 29]. This study investigates the effectiveness and the performance of smoothed aggregation on large-scale bone modeling problems.

The biomechanics component of our research primarily involves nonlinear micro-FE analyses [26, 8, 5] that can require a total of over 100 linear solves in a time-stepping Newton scheme, but this study focuses on the performance of a single linear solve. That is the solution of

$$Ax = b$$

where $A$ is the FE stiffness matrix, $x$ is the solution vector to be solved for, and $b$ is the FE residual vector. Iterative solvers generate approximate solutions $\hat{x}$ and, in this study, convergence is declared when the residual $r = b - A\hat{x}$ satisfies $\|r\|_2/\|b\|_2 < 10^{-6}$.

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 algebraic multigrid in §3. In §4 our project's computational architecture for scalable high resolution bone modeling is described. In §5 the performance of large scale 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 (e.g., during a fall). Therefore, understanding the mechanisms that lead to failure in bones is of great clinical importance. Characterization of bone mechanical properties [13] is a challenging problem, since they vary between anatomic sites, across individuals, over time and with disease. Our main area of research in orthopaedic biomehanics is the mechanical behavior of trabecular bone (for a review, see Keaveny [22]) 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) [31] scans to obtain geometrically accurate FE models of specimens (Figure 1.1). While these voxel meshes with non-smooth surfaces have initaially raised concerns among researchers, the accuracy of these models has been investigated in detail [37, 18, 27]. A general rule for convergence of stresses at the tissue and apparent levels is to ensure a resolution such the that three to four elements exist across the thickness of a beam or plate (trabecula) in the trabecular bone structure [18, 27]. 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 [39] and yield properties [8] have been determined using specimen-specific experimental data. (b) At the continuum level, trabecular bone orthotropic elastic properties [36], effects of geometrically nonlinear deformations [5], and yield behavior under multiaxial loads [25, 7] have been investigated. (c) At the whole bone level, load transfer characteristics [6] and stress distributions in healthy and osteoporotic bones [38] have been studied. Current research using the micro-FE approach is primarily focused on problems with

geometric and material nonlinearities (e.g., 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 [20]. 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 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 [38]. 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 [38]. While the memory requirement is small, the slow convergence (5 weeks of wall-clock time) to reduce the residual by three orders of magnitude indicates the need for a highly scalable and fast solver for such problems. We have used the smoothed aggregation multigrid solver discussed in the present study to perform a linear analysis on one human vertebral body micro-FE model with 57 million elements. We use a convergence tolerance of $10^{-6}$ and this analysis currently takes under 9 minutes wall clock time (144 CPU hours) on 960 processors (64 nodes) of an IBM SP3 parallel computer (RS6000-375MHz 16GB memory/node) at the Lawrence Livermore National Laboratory (LLNL).

**3. Multigrid introduction.** Multigrid methods are among the most efficient iterative algorithms for solving the linear algebraic systems associated with elliptic partial differential equations [19, 34, 11]. The crux of multigrid methods is the use of multiple resolutions of the problem—to uniformly reduce the entire spectrum of the residual—in the iterative scheme. Multigrid is motivated by the fact that, for Poisson's equation, high-energy (or oscillatory) components of the error are effectively reduced with simple iterative methods, and low-energy (or smooth) components can be resolved with an auxiliary lower-resolution discretization of the problem. This processes is applied recursively to provide a highly scalable method (i.e., the computational complexity of the solution per degree of freedom has little or no dependence on the size of the problem). Thus, the use of inexpensive iterative methods (smoothers), to reduce the region of the spectrum in which they are most efficient, results in low constants in the complexity of the solve. The success of multigrid is due to its utilization of these smoothers, in concert, to uniformly reduce the entire spectrum of the error. Multigrid is popular because smoothers and corresponding coarse grid spaces have been developed for many discretizations of PDEs of interest.

Multigrid requires three types of algebraic operators: 1) the grid transfer operators (i.e., the *restriction* and *prolongation* operators, which are implemented with a rectangular matrix $R$ and $P = R^T$, respectively); 2) the operator $A_{coarse}$ on each coarse grid (the fine grid operator $A_{fine}$ is generated by the finite element application); and 3) inexpensive iterative solvers that effectively reduce high frequency error (more precisely, error that is not represented well on the coarse grid). The coarse grid operators can be formed in one of two ways, either algebraically with a Galerkin process to form variational coarse grids ($A_{coarse} \leftarrow RA_{fine}P$) or, by creating a finite element mesh, on each coarse grid, and using a matrix generated by the finite element method.

Only algebraic coarse grids are considered in this study so as to maintain a narrow interface with the finite element application. Algebraic coarse grids are also inherently more robust because material nonlinearities are incorporated in the Galerkin process whereas, for instance, regions of stress concentration may not form on a coarse version of a problem. Additionally, boundary conditions are naturally incorporated in Galerkin coarse grid operators.

Figure 3.1 shows the standard multigrid *V-cycle* with: 1) a smoother $x \leftarrow S(A, b, x_0)$ that takes an operator $A$, the right hand side vector $b$, an initial guess $x_0$ and returns an updated solution $x$, 2) a restriction operator $R_{i+1}$, on each coarse grid $i+1$, that maps residuals from the fine grid space $i$ (the rows of $R_{i+1}$ are the discrete representation, on the fine grid, of the coarse grid function space), and 3) the operator $A_i$ on each grid $i$. One multigrid cycle is generally used as a preconditioner to CG, i.e., the CG preconditioner is $MGV(A_0, r)$ with the fine grid operator $A_0$ and the CG residual $r$.

**function** $MGV(A_i, r_i)$
  **if** $i$ **is the coarsest grid**
    $x_i \leftarrow A_i^{-1} r_i$ - - direct solve of coarsest grid
  **else**
    $\bar{x}_i \leftarrow S(A_i, r_i, 0)$ - - pre-smooth without an initial guess
    $\hat{r}_i \leftarrow r_i - A\bar{x}_i$
    $r_{i+1} \leftarrow R_{i+1}(\hat{r}_i)$ - - restriction of residual to coarse grid
    $x_{i+1} \leftarrow MGV(R_{i+1} A_i R_{i+1}^T, r_{i+1})$ - - the recursive application of multigrid
    $\hat{x}_i \leftarrow \bar{x}_i + R_{i+1}^T(x_{i+1})$ - - prolongation of coarse grid correction
    $x_i \leftarrow S(A_i, r_i, \hat{x}_i)$

  **return** $x_i$

FIGURE 3.1. *Multigrid V-cycle*

A multigrid method is defined by the method's coarse grid function space (see [32]). This study uses the "smoothed aggregation" method of Vanek, Mandel and Brezina because it has superior theoretical convergence characteristics to plain aggregation methods [41]. Aggregation methods start with the kernel of the operator without any Dirichlet boundary conditions. For static 3D finite element formulations in solid mechanics there are six such kernel vectors—the *rigid body modes*. The rigid body modes can, in principle, be constructed from the stiffness matrix (without Dirichlet boundary conditions imposed) [14], however, these spaces are easily constructed from the nodal coordinates for elasticity problems. Additionally, these rigid body modes are a good basis even if there is no kernel for the operator, as with dynamic problems where a mass matrix is added to the stiffness matrix.

Many rigid body mode methods have been developed [12, 14, 16]; the addition of smoothing provides significant improvement to these methods [40, 24], especially on more challenging problems. Additionally, smoothing provides nearly $h$ independent convergence. Smoothed aggregation, as the name suggests, begins by aggregating fine grid nodes into "strongly connected" aggregates (see [40, 1]). The kernel vectors (of the operator without any Dirichlet boundary conditions) are simply injected into these aggregates to construct a tentative prolongator $P_0$ (a block diagonal matrix). It is well known from multigrid theory that high energy coarse grid spaces deteriorate multigrid convergence [42]; smoothed aggregation reduces the energy of the coarse grid functions by applying an iterative solver (i.e., a smoother) to the coarse grid functions (columns of $P_0$). These smoothed kernel vectors are factored via a QR decomposition, with "Q" used in the prolongator $P$ in a standard multigrid framework and "R" used in the kernel vectors of the next grid, thus allowing the method to be applied recursively. The fact that this smoothing is done without additional input from the user is the key to the practicality of this algorithm. This process is applied recursively until the top grid is small enough to solve quickly with an available accurate solver.

**4. Parallel Finite Element Architecture.** This section describes the computational architecture in this project. There are three major components: the bone micro-FE modeling process that creates the meshes and post-processes the simulation results (§4.1), a parallel finite element implementation (§4.2), and a parallel multigrid linear solver (§4.3).

**4.1. Micro-FE Modeling of Bone (BOBCAT).** 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 (Figure 4.1, top):

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 [27]. 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 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 implementation of the finite element application Athena/FEAP.

**4.2. Parallel finite element software (Athena/FEAP).** The parallel finite element system *Athena* is built on the serial finite element code FEAP [15], and the parallel graph partitioner ParMetis [21]. Figure 4.1 shows a schematic representation of this system. The finite element input file, generated by BOBCAT, 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 partition. 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 processor. 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 two small global text files with the material properties and solution script, FEAP runs on each processor much as it would in serial mode, in fact FEAP itself has no parallel constructs, but only interfaces with our parallel solver Prometheus (below).

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. Clusters of SMPs are accommodated by first partitioning the problem onto the SMPs and then each local problem is partitioned onto each processor as depicted in Figure 4.1. This approach implicitly takes advantage of any increase in communication performance within the SMP, though the numerical kernels (in PETSc [4]) are "flat" MPI codes.

**4.3. Parallel multigrid linear solver (Prometheus).** The largest portion of the simulation time for our bone micro-FE problems is typically spent in the linear solve. The solver package *Prometheus* is used for this study (available at [30]). Prometheus has three multigrid algorithms, additive and (true) parallel multiplicative smoother preconditioners (both general block and nodal block versions) and several smoother iterators for the additive preconditioners (Chebyshev polynomials are used in this study).

Prometheus repartitions the coarse grids to maintain load balance and reduces the number of active processors on the coarsest grids 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 parallel construction of the coarse grid spaces to have the exact serial semantics in the regions between processors (e.g., 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 faster if fewer processors are used.

**5. Numerical studies.** This section investigates the performance of our micro-mechanical modeling of bone with particular attention to the performance of our linear solver. The runtime cost of our bone micro-FE analyses can be segregated into five primary sections:

1. The Athena parallel mesh setup.
2. The FEAP element residual, tangent, and stress calculation.
3. The linear solver *mesh* setup (construction of the coarse grids).
4. The linear solver *matrix* setup (coarse grid operator construction and subdomain factorizations).
5. The solve for the solution.

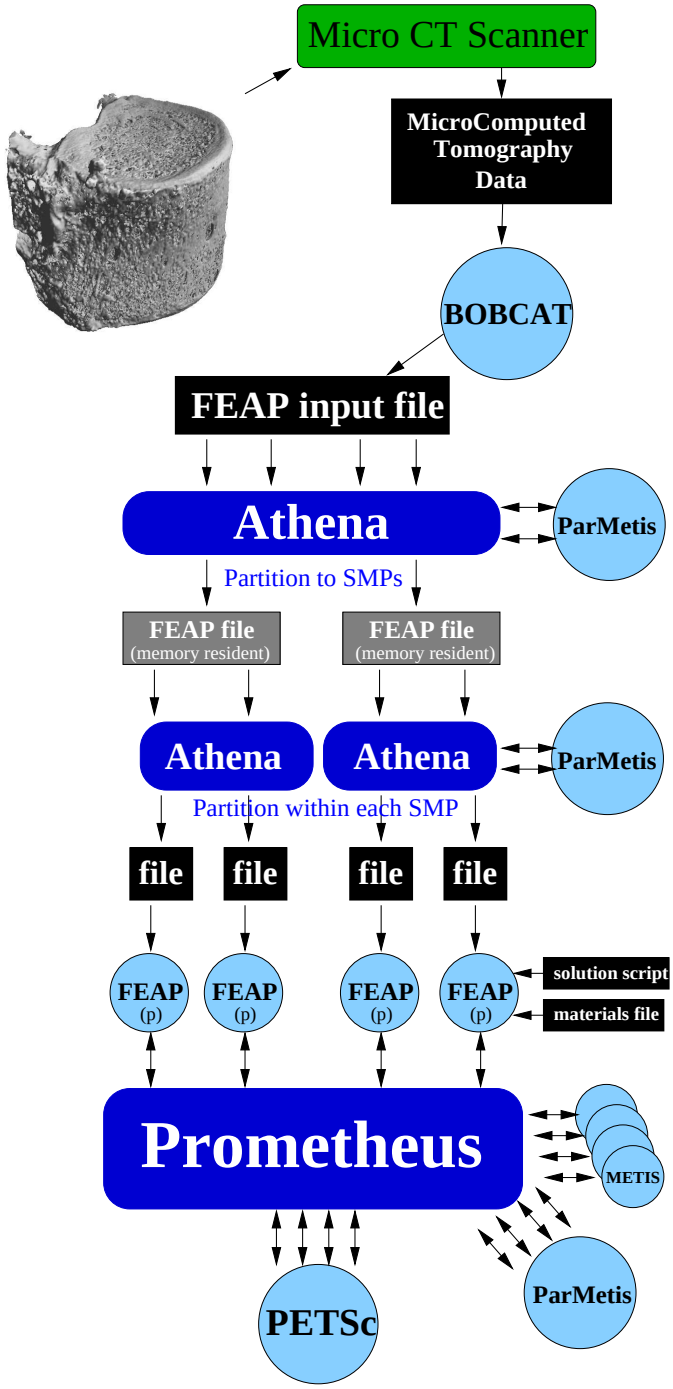


FIGURE 4.1. *Voxel based mesh generation and parallel finite element implementation.*

Although the initial setup in Athena (1) and to a lesser extent the mesh setup cost in Prometheus (3), represents a significant portion of the time in the numerical experiments, because only one linear system is solved, these costs are amortized in nonlinear simulations which constitute the predominant use of our finite element application [8]. Thus, the performance of Athena is discussed only briefly in this study. The finite element work in FEAP (2) requires little communication and represents a modest percentage of the run times and is not considered in the performance analysis.

The mesh setup costs in Prometheus (3) 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. Analyzing the performance of the mesh setup is difficult because there are many types of operations and there are no obvious kernels as with the numerically intensive parts of the solve. Generally, much of the mesh setup time is spent iterating over the complex graphs used in these AMG algorithms. Specifically, this study uses the square of the matrix graph (i.e., the graph of $A \cdot A$ [35]) in the aggregation—this accounts for about 30% of the mesh setup time in the experiments.

The primary focus of this performance analysis is on the matrix setup computations (4), and on the solution phase (5). Most of the floating point work is in these last two components of the simulation, and this is the only work that is not amortized in a full Newton nonlinear simulation. The performance analysis is separated into two parts: 5.2) a scalability study on two IBM parallel computers to assess the parallel efficiency of the solver algorithms and implementations, and 5.3) a performance analysis of the important multigrid kernels on one compute node of an IBM Power3 and Power4 to assess serial and nodal performance.

**5.1. Micro-FE Problem.** We consider finite element models of a thoracic vertebral body (T-10) at different 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 ($\mu$CT80, SCANCO Medical AG, Bassersdorf, Switzerland). After endplates are removed, uniform displacement boundary conditions are applied to simulate a 1% strain compression (Figure 5.1 left). The practical purpose of such a simulation is to obtain stress distributions in the human vertebral body [6] as well comparing the linear elastic properties of the micro-FE model with laboratory mechanical testing data. Figure 5.1 (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.

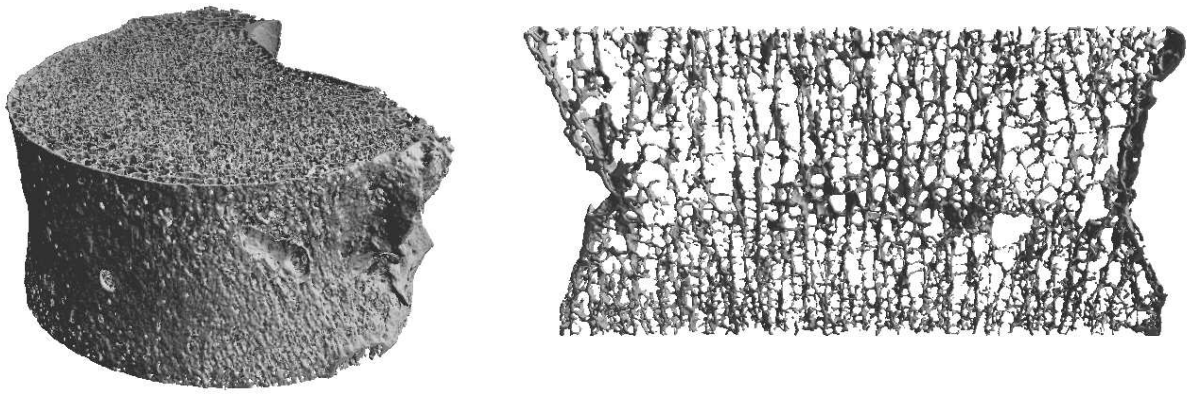

FIGURE 5.1. *Vertebral body with endplates removed (left). one millimeter thick cross-section of vertebral body at 30 $\mu m$ resolution shows the complexity of the micro-FE mesh (right).*

| $h$ ($\mu m$) | 150 | 120 | 80 | 60 | 40 |
|---|---|---|---|---|---|
| # of equations ($10^6$) | 7.4 | 13.5 | 37.6 | 77.9 | 237 |
| # of elements ($10^6$) | 1.08 | 2.12 | 7.12 | 16.9 | 57.0 |
| # of compute nodes (IBM LLNL) | 2 | 3 | 9 | 20 | 64 |
| # of compute nodes (IBM SDSC) | 13 | 24 | 67 | 140 | NA |

TABLE 5.1
*Scaled vertebral body mesh quantities.*

Five meshes of this vertebral body with discretization scales ($h$) ranging from 150 microns to 40 microns, and pertinent mesh quantities are shown in table 5.1. The bone tissue is modeled as an isotropic linear elastic material with a Young's modulus of 1 GPa and a Poisson's ratio of 0.3. All elements are geometrically identical trilinear hexahedra (8-node bricks).

**5.2. Scalability study.** 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, in our case, by simply starting with the micro-computed tomography data (30 micron resolution) and decreasing the mesh resolution (e.g., from 40 microns to 150 microns). Performance is measured on numerical experiments with the number of processors selected so as to place approximately the same number of equations per processor.

Scaled speedup is a measure of parallel efficiency, and is in our opinion the best method of assessing the scalability characteristics of a parallel finite element application for several reasons. First, scaled speedup maintains approximately the same pressure on the cache, in all experiments, so that parallel inefficiencies are not hidden by cache effects—as can be the case with unscaled speedup. 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 turn around time from a particular application—providing relatively accurate data about the time costs of the application on a particular machine. 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 for each resolution.

Two parallel computers are used in this study:

- IBM SP at Lawrence Livermore National Laboratory (LLNL): 64 compute nodes, 16 375Mz Power3 processors per node, 16GB memory per node, with 24 Gflop/sec theoretical peak flop rate per node §5.2.1.
- IBM SP at San Diego SuperComputing Center (SDSC): 144 compute nodes, 8 375MHz Power3 processors per node, 4GB memory per node, with 12 Gflop/sec theoretical peak flop rate per node §5.2.2.

In this study a conjugate gradient solver is used with one multigrid V-cycle as the preconditioner. The smoothed aggregation multigrid method, described in §3, is used with three multigrid smoothers: 1) first order Chebyshev [3], 2) fourth order Chebyshev, and 3) one sweep of parallel symmetric Gauss–Seidel in the pre- and post-smoothing steps [1].

**5.2.1. IBM Power3 (LLNL).** The IBM SP at LLNL has 16 Power3 processors per node, of which only 15 are used. The number of compute nodes is selected to place about 265K degrees of freedom (dof) per processor. Each node has 16GB of main memory and they are connected with a double Colony switch. Figure 5.2 shows the iteration counts (left) and the solve times (right) for all five meshes and three choices of smoothers.

The flop rates are the most critical measure of parallel efficiency, given that the multigrid method provides approximately constant iteration counts and the amount of work per dof quickly asymptotes to a constant—the only remaining source of inefficiency is in the flop rate. Figure 5.3 shows the average per processor flop rate (left) and the total solution time (right), the latter being segregated into the three components of a linear solve.

This parallel efficiency in the flop rate for the solve phase is nearly perfect (98% in Figure 5.3 left), with the first order Chebyshev smoother from 2 to 64 nodes. Figure 5.3 (left) also shows that the flop rate in the solve phase with the parallel Gauss–Seidel smoother is 91% parallel efficienct—this is excellent for fully parallel Gauss–Seidel and can be attributed to, among other things, the large processor subdomains used in this study. Figure 5.2 (right) shows a parallel efficiency for the solve phase of 79%, with the first order Chebyshev smoother. This inefficiency is apparently due to more flops being performed per dof as the problems get larger, but the solve time does quickly asymptote to a constant. Note, the iteration counts are not deterministic because ParMetis and Prometheus are not deterministic and we generally see fluctuations in the iteration counts of about 10%, but the overall trend is of a constant number of iterations.

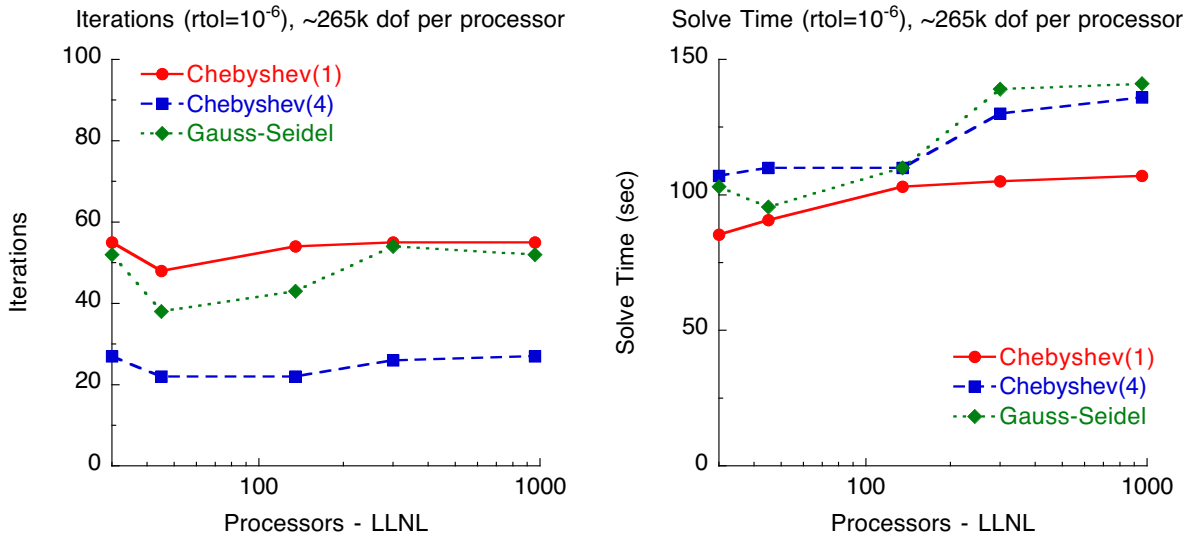


FIGURE 5.2. *IBM SP (LLNL) iteration counts (left) and solve times (right).*

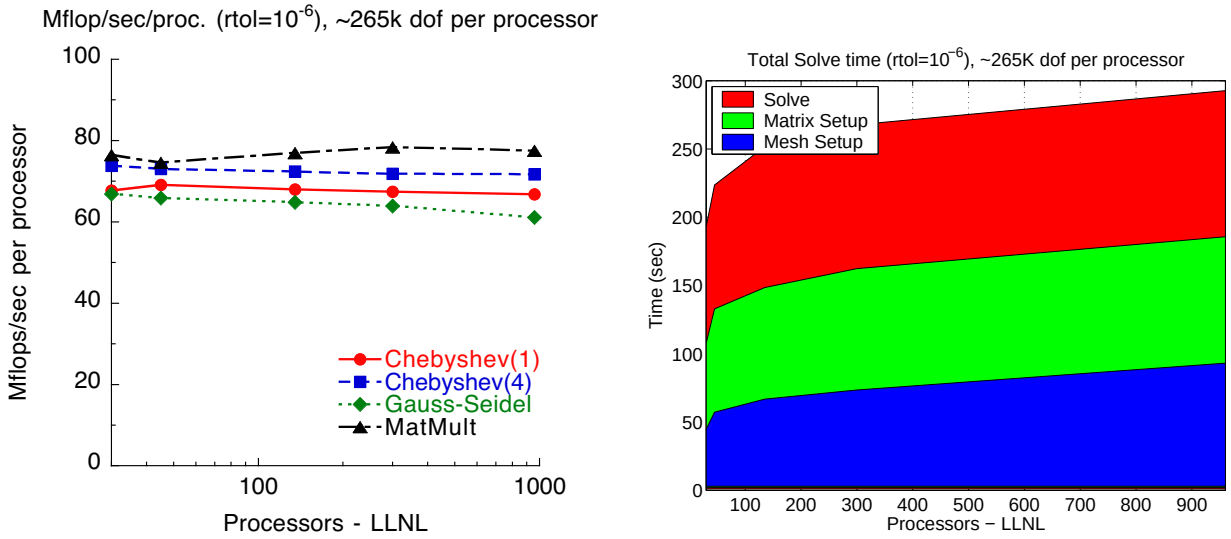


FIGURE 5.3. *IBM SP (LLNL) Mflop rates (left) and total linear solve times (right).*

Figure 5.3 (right) shows that the matrix setup (middle band) is scaling well—this is dominated by the Galerkin coarse grid operator construction. The Galerkin coarse grid operator construction requires a sparse matrix triple product $A_{i+1} \leftarrow R \cdot A_i \cdot P$ (RAP), and is computationally intensive. There is, however, much data reuse in the RAP (the RAP is analogous to a BLAS4 operation, because there are three matrices and $R = P^T$), but the RAP requires communication because some matrix entries are computed non-locally. This communication is exacerbated because Prometheus repartitions the coarse grids for load balancing and thus the coarse grids are not as nested as they otherwise would be.

Figure 5.4 shows the total wall-clock time for the three smoothers. This shows that the total time—including the distribution of the mesh—for just one linear solve increases by less than a factor of two from 2 to 64 compute nodes.

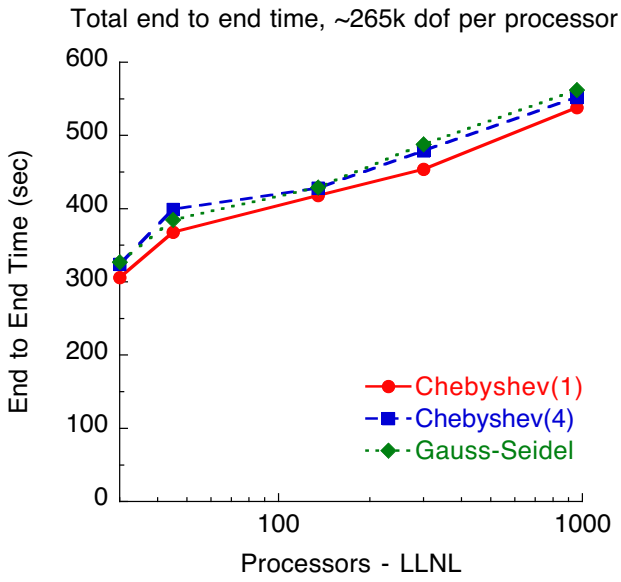


FIGURE 5.4. *IBM SP (LLNL) end to end time.*

**5.2.2. IBM Power3 (SDSC).** The IBM SP at SDSC has 8 Power3 processors, 4GB of main memory per node and the nodes are connected with a single Colony switch. The number of compute nodes is selected to place about 70K degrees of freedom (dof) per processor. Figure 5.5 shows the iteration counts (left) and the solve times (right).

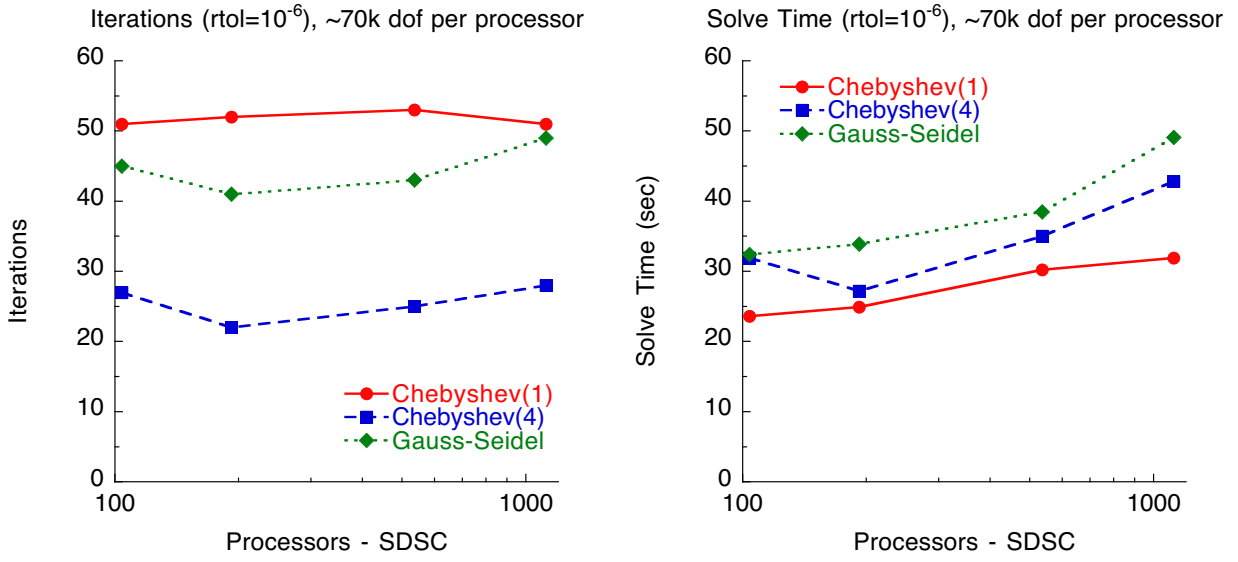


FIGURE 5.5. *IBM SP (SDSC) iteration counts (left) and solve times (right).*

Figure 5.6 shows the average per processor flop rate (left) and the total solution time for the first order Chebyshev smoother (right), segregated into the three components of a linear solve.

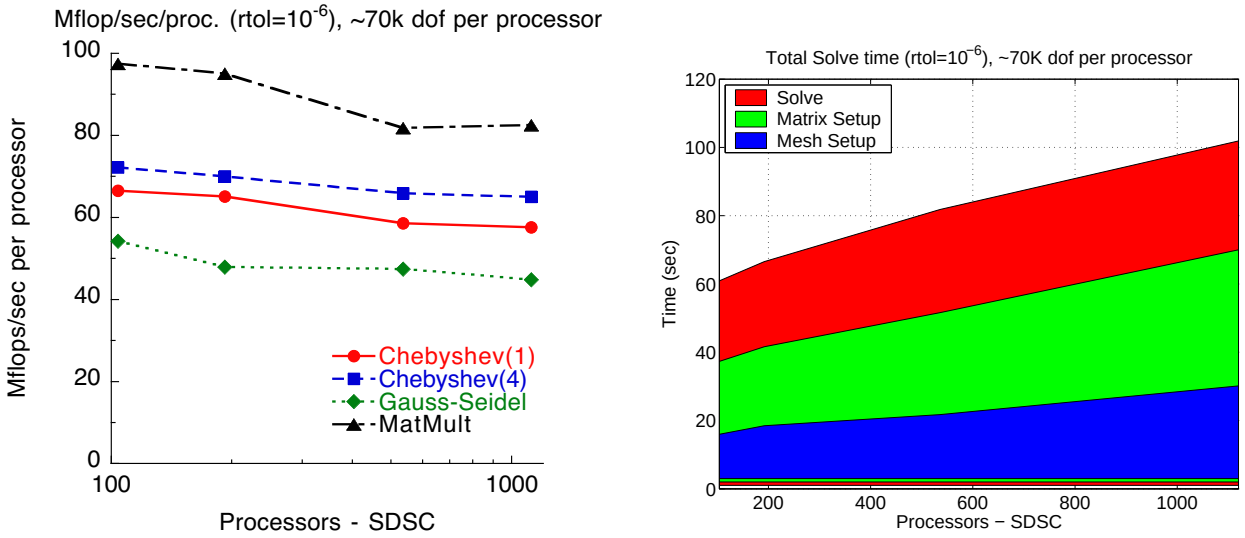


FIGURE 5.6. *IBM SP (SDSC) Mflop rates (left) and total linear solve times (right).*

The solves are generally scaling well. The mesh setup phase is scaling very well and as well as the LLNL data in Figure 5.3. The matrix setup phase times are increasing; this could be due to the smaller processor subdomain sizes in these studies.

Note, the LLNL IBM has a double Colony switch and the nodes at SDSC are half the size of those at LLNL, thus, to a first order approximation both machines have one switch per 8 processors. These different configurations could effect the performance characteristics of the setup phases. Some of this inefficiency, compared to LLNL, is due to the increased communication to computation ratio due to the smaller processor subdomains used at SDSC. Additionally, the LLNL studies use only 15 of the 16 processors on each node—the one idle processor per node could provide critical processing of MPI messages without interrupting the primary compute processes.

The SDSC Mflop/sec/proc data is qualitatively different from the LLNL data (Figure 5.3 left) in that 1) the matrix vector product flop rate is higher, 2) the Chebyshev smoother is about the same and 3) the Gauss–Seidel smoother is slower. This is because the SDSC IBM seems to have a higher effective memory bandwidth, given that the matrix vector products are faster, however, the smaller processor subdomains used in the SDSC study reduces the parallel efficiency. The relatively poor performance of the Gauss–Seidel smoother, compared to the LLNL data, is due to the fact that parallel Gauss–Seidel requires more (and more complex) communication—resulting in more degradation in parallel efficiency with reduced processor subdomain size.

Figure 5.7 shows the total wall clock time for this series. This shows that the total wall clock time is increasing by a less than a factor of two on 104 to 1120 processors for the first order Chebyshev and Gauss–Seidel smoothers.

This scalability study has demonstrated parallel efficient solutions of these geometrically complex problems in that the cost per degree of freedom scales very well with increased problem size. Additionally, the constants in these solution times are modest in comparison with other published results for linear solves of unstructured elasticity problems of this scale [23].

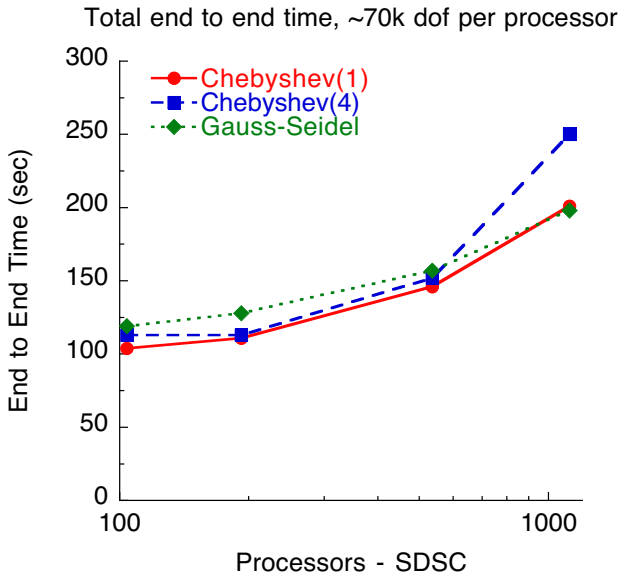


FIGURE 5.7. *IBM SP (SDSC) end to end time.*

**5.3. Nodal performance on the IBM Power3 and Power4.** The preceding section demonstrated the scalability of our linear solver algorithms and implementations. This section investigates the nodal performance of the IBM Power3 and Power4 architectures, that is, the parallel efficiency from one to 16 processors on the Power3 and from one to 32 processors on the Power4. Two components of the solution process are analyzed: 1) the RAP and 2) the actual solve or the time spent in the CG iterations. The matrix setup is dominated by the matrix triple product (RAP) in the construction of the Galerkin coarse grids and the cost of each CG iteration is dominated by the cost of the matrix vector product on the fine grid. We therefore look carefully at these two numerical kernels.

Figure 5.8 shows the aggregate flop rates (in contrast to the per processor flop rate reported in the previous section) for the RAP and the matrix vector product on the fine grid.

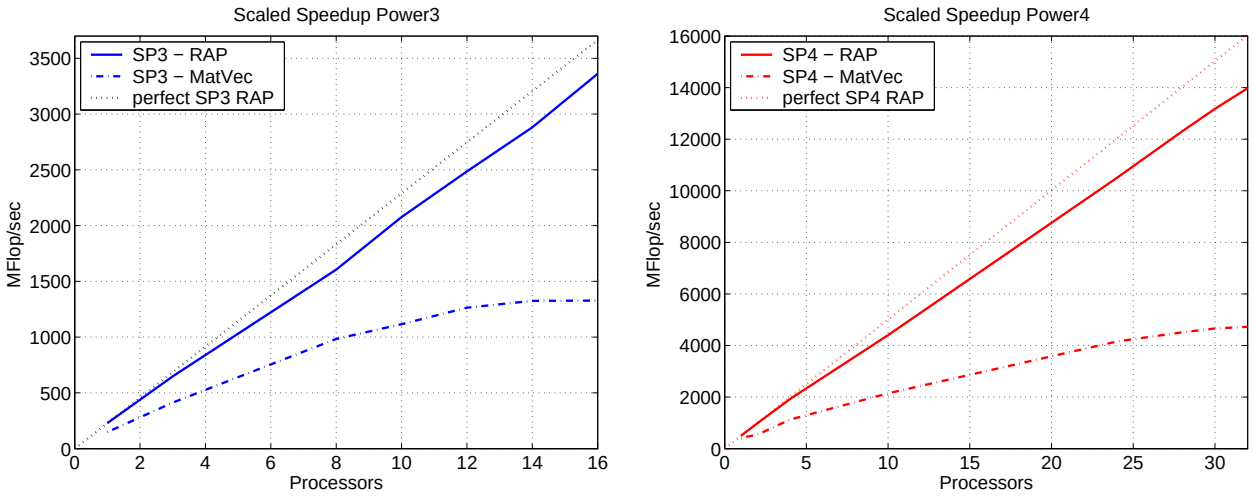


FIGURE 5.8. *Aggregate flop rate of Primary numerical Kernels in an AMG solver for the IBM SP3 (left) and the IBM SP4 (right).*

Several finite element meshes are used in this data to place between 150K and 300K dof per processor so as to limit cache effects from polluting the data. Note, FE meshes from a simple 3D problem, with trilinear hexahedra, are used in these studies, due to a lack of available trabecular bone meshes of an appropriate size. The meshes used here are from a continuum model which has more non-zeros per row than other meshes in this paper and therefore has slightly better flop rates in the RAP.

This data shows that, first, the RAP is scaling well with about 90% of perfect speedup from 1 to 32 processors on the Power4 and about 95% of perfect speedup from 1 to 16 processors on the Power3. This relatively high parallel efficiency of the RAP is due to the extensive data reuse (e.g., each entry $A_{ij}$ in the fine grid is used in approximately 80 flops). These flop rates are however significantly below the theoretical peaks: about 14% for the Power3 and 8% for the Power4.

Figure 5.8 also shows that the matrix vector products do not speedup as well as the RAP. This is due to the relatively little data reuse in a sparse matrix vector product (approximates two flops per value retrieved from main memory). It can be deduced that we are memory bandwidth limited by observing that the flop rate for the matrix vector product peaks at about 12 processors on the Power3. For instance, we gain a speedup of 8.5 on 12 processors and 8.9 on 16 processors. This dramatic plateau is attributed to the simple memory architecture of the Power3 (relative to the Power4), which has a well balanced memory bus architecture (i.e., almost all stages in the memory bus have a hardware bandwidth limit of 16.0 GBytes/sec). This hardware limit is, however, not achievable in practice and a more realistic bound on the bandwidth is the memory bandwidth performance of the STREAMS benchmark, which is about 8.5 GB/sec. The IBM hardware performance monitors were used to measure about 6.4 GB/sec bandwidth performance in the Power3 experiments in Figure 5.8. This is about 75% of the bandwidth performance of the STREAMS benchmark, which is reasonably good considering that a sparse matrix vector product has more complex memory access patterns than the STREAMS benchmark. Thus, we conclude that the matrix vector product is bound by the memory bandwidth on the Power3.

The Power4 architecture is far more complex than the Power3 and Figure 5.8 shows that the flop rate does not plateau as dramatically as the Power3, but the speedup is only 11.3 with 32 processors on the fine grid matrix vector product. And, like the Power3, the Power4 demonstrates very good speedup in the RAP.

This analysis shows that, though the flop rates are increasing with each generation of the Power series, the percentage of peak is decreasing noticeably. To some extent this is an unavoidable result of the extraordinary rate at which processors speeds are increasing and the inability of the communication architecture to keep up. This growing gap between processor speeds and memory bandwidth provides an opportunity to decrease the solve times by increasing the flop rate of—in particular—the matrix vector products. We see two classes of techniques that can take advantage of this opportunity. One, use a highly optimized matrix free residual computation in the finite element method for the matrix vector products on the fine grid. And two, use higher order smoothers (e.g. fourth order Chebyshev or several Gauss–Seidel iterations [33]) and implement special purpose, highly optimized, kernels that fold several iterations of the smoother together in such a way as to maximize data reuse.

**6. Conclusions.** This study demonstrates that a general algebraic multigrid method, smoothed aggregation, is effective on finite element analysis of high-resolution bone models with complex geometries on problems with up to 237 million degrees of freedom. We have demonstrated the first, to our knowledge, application of optimal linear solver algorithms to bone micro-mechanics problems. Additionally, we have demonstrated the first, again to our knowledge, utilization of large-scale parallel computers with over 1000 processors for this class of problems.

While a single linear solve was considered in this study, our results validate the use of the smoothed AMG solver in geometrically and/or materially nonlinear problems—which is the goal in the orthopaedic biomechanics component of our research. Our preliminary analyses which include both geometrical and material nonlinearities indicate that over 2000 linear solves are necessary for simulations that reach 3% apparent level strain which is beyond the point of stability (ultimate stress). While these preliminary analyses are run using a 5-mm cubic sample of vertebral trabecular bone with a finite elasto-plasticity material constitution [28], more realistic material models which include trabecular tissue strength asymmetry [8] and damage [17] behavior are being implemented.

This study has shown that smoothed aggregation is highly scalable in that we are able to solve problems with up to 237 million degrees of freedom in near constant time with a constant number of equations per

processor and that this time is about three and one half minutes on IBM SP3s with 265K degrees of freedom per processor for a single linear solve phase in a full Newton iteration scheme (i.e., matrix setup and solve). These are the largest analyses on unstructured elasticity problems on massively parallel computers that we are aware of. Additionally, this is significant in that no special purpose algorithms or implementations were required to achieve a highly-scalable performance on common parallel computers with these extremely geometrically complex finite element models.

**Acknowledgments.** We would like to thank 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 services. We would like to thank Jeff Fier of IBM at LLNL for performance tuning advice and help with performance monitoring tools. 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). Supercomputing resources at San Diego Supercomputer Center (SDSC) were available through the National Partnership for Computational Infrastructure (NPACI Grant UCB-254). 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 SC01: High Performance Networking and Computing, 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, M. Brezina, J. J. Hu, and R. S. Tuminaro, *Parallel multigrid smoothing: polynomial versus Gauss–Seidel*, Journal of Computational Physics, 188 (2003), pp. 593–610.

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

[5] 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.

[6] 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.

[7] H. H. Bayraktar, A. Gupta, R. Y. Kwon, and T. M. Keaveny, *Multiaxial failure behavior of human femoral trabecular bone*, in Transactions of the Orthopaedic Research Society, vol. 28, New Orleans, 2003, p. 202.

[8] H. H. Bayraktar, E. F. Morgan, G. L. Niebur, G. E. Morris, E. K. Wong, and T. M. Keaveny, *Comparison of the elastic and yield properties of human femoral trabecular and cortical bone tissue*, Journal of Biomechanics, (2003). In Press.

[9] 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.

[10] A. Brandt, *Multi-level adaptive solutions to boundary value problems*, Math. Comput., 31 (1977), pp. 333–390.

[11] W. L. Briggs, V. E. Henson, and S. McCormick, *A multigrid tutorial, Second Edition*, SIAM, Philadelphia, 2000.

[12] 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.

[13] S. C. Cowin, ed., *Bone Mechanics Handbook*, CRC Press, Boca Raton, 2 ed., 2001.

[14] C. Farhat and F.-X. Roux, *A method of finite element tearing and interconnecting and its parallel solution algorithm*, International Journal for Numerical Methods in Engineering, 32 (1991), pp. 1205–1227.

[15] *FEAP*. www.ce.berkeley.edu/∼rlt.

[16] J. Fish and V. Belsky, *Generalized aggregation multilevel solver*, International Journal for Numerical Methods in Engineering, 40 (1997), pp. 4341–4361.

[17] D. P. Fyhrie and M. B. Schaffler, *Failure mechanisms in human vertebral cancellous bone*, Bone, 15 (1994), pp. 105–9.

[18] 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.

[19] W. Hackbusch, *Multigrid methods and applications*, vol. 4 of Computational Mathematics, Springer–Verlag, Berlin, 1985.

[20] 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.

[21] G. Karypis and V. Kumar, *Parallel multilevel k-way partitioning scheme for irregular graphs*, ACM/IEEE Proceedings of SC96: High Performance Networking and Computing, (1996).

[22] T. M. Keaveny, *Bone Mechanics Handbook*, CRC Press, Boca Raton, 2 ed., 2001, ch. 16, pp. 1–42.
[23] 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 SC02: High Performance Networking and Computing. Gordon Bell Award.
[24] J. Mandel, M. Brezina, and P. Vaněk, *Energy optimization of algebraic multigrid bases*, Tech. Report 125, UCD/CCM, February 1998.
[25] 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.
[26] 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.
[27] 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.
[28] P. Papadopoulos and J. Lu, *A general framework for the numerical solution of problems in finite elasto-plasticity*, Computer Methods in Applied Mechanics and Engineering, 159 (1998), pp. 1–18.
[29] G. Poole, L. Y.C., and M. J., *Advancing analysis capabilities in ansys through solver technology*, Electronic Transactions in Numerical Analysis, 15 (2003), pp. 106–121.
[30] *Prometheus*. www.cs.berkeley.edu/∼madams.
[31] 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.
[32] B. Smith, P. Bjorstad, and W. Gropp, *Domain Decomposition*, Cambridge University Press, 1996.
[33] M. M. Strout, L. Carter, J. Ferrante, J. Freeman, and B. Kreaseck, *Combining performance aspects of irregular Gauss–Seidel via sparse tiling*, in 15th Workshop on Languages and Compilers for Parallel Computing (LCPC), College Park, Maryland, 2002.
[34] U. Trottenberg, C. Oosterlee, and A. Schüller, *Multigrid*, Academic Press, London, 2001.
[35] R. Tuminaro and C. Tong, *Parallel smoothed aggregation multigrid: Aggregation strategies on massively parallel machines*, in SuperComputing 2000 Proceedings, J. Donnelley, ed., 2000.
[36] 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.
[37] 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.
[38] B. Van Rietbergen, R. Huiskes, F. Eckstein, and P. Rüegsegger, *Trabecular bone tissue strains in the healthy and osteoporotic human femur*, Journal of Bone and Mineral Research, (2003). In press.
[39] 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.
[40] 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.
[41] P. Vaněk, J. Mandel, and M. Brezina, *Convergence of algebraic multigrid based on smoothed aggregation*, Numerische Mathematik, 88 (2001), pp. 559–579.
[42] W. L. Wan, *An energy minimizing interpolation for multigrid methods*, Tech. Report 97-18, UCLA, April 1997. Department of Mathematics.