Grid-based Image Registration

William Gropp[1], Eldad Haber[2], Stefen Heldmann[2], David Keyes[3], Neill Miller[1], Jennifer Schopf[1], and Tianzhi Yang[3]

[1] Computation Institute, University of Chicago, Chicago, IL 60637
{gropp,neillm,schopf}@mcs.anl.gov
[2] Mathematics & Computer Science, Emory University, Atlanta, GA USA 30322
{haber,heldmann}@mathcs.emory.edu
[3] Applied Physics & Applied Mathematics, Columbia University
New York, NY USA 10027 {david.keyes,ty2109}@columbia.edu

We introduce and discuss preliminary experience with an application that has vast potential to exploit the Grid for social benefit and offers interesting resource assessment and allocation challenges, having real-time aspects: image registration. Image registration is generally formulated as an optimization problem that satisfies constraints, such as coordinate displacements that are affine or volume-preserving or that obey the laws of elasticity. Three-dimensional registration of high-resolution images is computationally complex and justifies parallel implementation. In turn, ensembles of registration tasks exploit concurrency in the simpler sense of job farming.

Registration is an elementary example of a much larger class of large-scale mesh-based computations that are in principle amenable to execution on the Grid, but are sensitive to workload-to-capability balance at synchronization points. While better resource assessment and allocation tools lift all such applications, reducing sensitivity to synchronization within an individual application is a complementary and equally important objective. We therefore examine the potential for weakening the synchronization sensitivity of general mesh-based bulk synchronous computations through less restrictive programming paradigms.

Keywords: Medical image registration, asynchronous numerical algorithms, Grid-based processing, MPI-based parallelization

Introduction

Imaging is exploding and, with it, so are the needs and opportunities for image registration. A recent catalog of books and journals on imaging from a single publisher [26] lists 5 pages of imaging journals, 13 pages of books on imaging techniques (CT, fMRI, ultrasound, etc.), 29 pages of books on diagnostic imaging of specialized anatomical domains, and 6 pages of books on radiotherapy and image-based intervention. Telling as well, with respect to clinical applications, is the formation of a new *Journal of Real-time Image Processing*.

Imaging applications are ripe for the Grid. The Globus-based MEDICUS system has already "broken the medical image communication barrier" [17], in the sense that raw images can be shared with unprecedented speed and transparency. The communication breakthroughs of the Grid create opportunities in Grid-enabled processing, as well, by opening up vast possibilities for registration of images that were previously not simultaneously available.

Numerical computing on the Grid, of which image registration is an example, is, in turn, only one of several categories of Grid-based applications, and has not so far been a significant driver of the Grid overall. The Grid is yet to be exploited by most computational scientists, though it is potentially very useful for applications requiring real-time solution or exceptional amounts of memory. This potential will be realized after two independent trends, each with its own inertia, converge: better tools for understanding and harnessing the dynamic performance capabilities of the Grid, and better asynchronous algorithms implemented in consistency-relaxed parallel programming models.

A basic problem that motivates this work can be posed as follows: given a number of images that require registration, an MPI-based program to perform the registration, and a collection of Grid-enabled compute resources, compute the registrations by a specified time or estimate for the user what portion of the registrations can be completed before a given time. A currently available affine linear registration code has complexity roughly linear in the voxel volume. The target images are 1024^3. Our test images of 128^3 voxels require approximately 5 minutes of processing on a commodity cluster of 8 dual nodes. The targets should therefore take approximately two days to run on such a cluster. The kernel that dominates CPU cycles is multivariate interpolation. The task is easily partitioned into arbitrary working-set distributions by apportioning subdomains (subvolumes) of the image space to processors. The processing that needs to be done to back out the parameters that specify the three-dimensional affine map is negligible in comparison to the easily partitioned interpolation work. However, there is synchronization at regular intervals between interpolation steps. Although our demonstration is confined to affine registration, the techniques are extensible with the same computational issues to more general registrations of clinical importance.

In Section 1, we consider the motivation for real-time registration, some registration algorithms, an example of registration, and an initial feasibility demonstration of registration recently conducted on the Grid. Section 2 backs up from registration as a particular application and examines advances in asynchronous algorithms more generally, in terms both of algorithms and software infrastructure. We conclude in Section 3 with some speculations.

The philosophy of this presentation is that algorithms must be adapted or created to bridge to "hostile" architectures to support applications, taking both the applications and the architectures as givens. The interplay of applications and architectures with algorithms is a two-way street, generally. Knowledge of algorithms can influence the way applications are formmulated and the way architectures are constructed. More often than otherwise, however, it is the

algorithms that must adapt to inflexible architecture and nonnegotiable formulations of the application.

1 Image registration

Real-time registration has numerous applications. In medicine, alone, registration arises in many contexts. Diagnosis and surgery planning make use of patient-to-reference anatomy registration. The evaluation of fitness for transplants makes use of patient-to-patient registration. In addition, patient-to-self registration of images taken at different times may be useful in monitoring progress. Besides clinical applications, real-time registration arises in automated surveillance, in which the goal is to recognize people whose images are stored in a database. In the area of robotics, manipulation of and navigation in the environment depends upon recognition of objects, which can include real-time registration aspects. Finally, we mention as a motivation for the importance of performing image registration tasks quickly the novel technique of super-resolution [9], which relies upon repeated registrations of an object that is moving with respect to the viewer. As its name implies, super-resolution is able to produce high-resolution images by comparing a series of low-resolution images of the same object. Intuitively, information at sub-pixel resolution that is "lost" in any one image can be recovered from staggered images in which the underlying information is captured in pixels that are displaced by a fraction of a pixel cell. This powerful technique allows a tradeoff between the quality of the instrument capturing the image in digital form and the capability of the computer system processing the resulting data.

1.1 Mathematical setting of registration

The typical mathematical setting of registration is optimization. One posits an objective function whose minimization is designed to minimize mismatch between a pair of images and seeks a transformation of one image into the other. This sounds simple in principle, but the simplicity can be deceptive. One must be careful, for instance, not to allow uncorrelated pixel-to-pixel matches from a list of pixels in one image to those of another, or else continguity of the transformation could be lost. Constraints may be imposed to preserve contiguity, to preserve volume, to map certain key points, etc. Since the number of constraints is generally vastly smaller than the number of parameters to be determined in a deformation map of one image into another, regularization is almost always required to remove ill-posedness.

Many optimization problems that arise in registration, as in other fields, require the solution of discretizations of elliptic partial differential equations; hence numerical ill-conditioning is often present. The data sets may be large-scale, with multiple billion-voxel images (thousand-fold resolution in each of

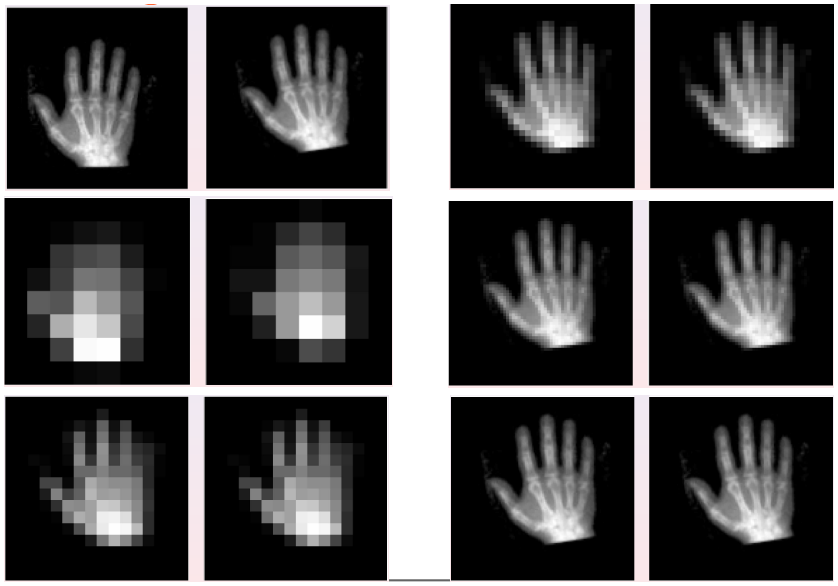

Fig. 1. Registration of hand images by a multilevel technique. The original pair of images is in the upper left. The next four pairs of images show successive stages of the registration process at successively finer resolutions, ending with the registered pair in the lower right.

three dimensions) coming on line, requiring parallel computing for each registration, in addition to distribution of registration of multiple image pairs over the Grid.

The field of image registration is mathematically rich, with both new theory and new heuristics playing roles in moving it forward.

Finally, in production mode, registration presents challenges in terms of managing the computational workflow, with issues of transparent archiving, remote access, and security requirements arising in conjunction with large data sets.

Mathematical descriptions of registration The problem of image registration can be posed intuitively as follows [22]: "given a template image and a reference image, find a transformation of the template image such that it becomes similar to the reference image." Images may be considered as fields over domains, in which a template image $T(\mathbf{x}) = T(x_1, x_2, x_3)$ and a reference image $R(\mathbf{x}) = R(x_1, x_2, x_3)$ are given and a transformation $\mathbf{u}(\mathbf{x}) = [u_1(\mathbf{x}), u_2(\mathbf{x}), u_3(\mathbf{x})]$ is sought such that $T(\mathbf{x} + \mathbf{u}(\mathbf{x})) \approx R(\mathbf{x})$. An example of a pair of images to be registered is given in Figure 2.

The generality of the flow $\mathbf{u}(\mathbf{x})$ can be controlled by its parameterization. A general affine map in three dimensions consists of just twelve parameters, independent of the number of pixels in the image:

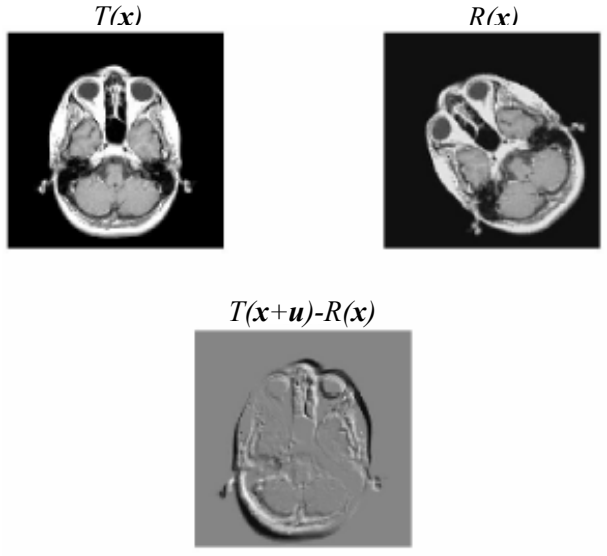

Fig. 2. $T(\mathbf{x})$ for a two-dimensional slice through the midplane of a head. $R(\mathbf{x})$, a reference image for the head. $T(\mathbf{x}+\mathbf{u})-R(\mathbf{x})$ after construction of a coordinate flow $\mathbf{u}$.

$$\begin{pmatrix} u_1 \\ u_2 \\ u_3 \end{pmatrix} = \begin{pmatrix} c_1 \\ c_2 \\ c_3 \end{pmatrix} + \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}. \tag{1}$$

Determination of the parameters is through minimization in some norm of an objective function. A typical example is

$$\min \frac{1}{2}||T(\mathbf{x}+\mathbf{u})-R(\mathbf{x})||^2 + \alpha S(\mathbf{u}), \tag{2}$$

where S is a regularization term and α is a weighting parameter that strikes a compromise between similarity and regularity. The regularization may, for instance, penalize lack of smoothness in $\mathbf{u}$. Registration based on the minimization of distance between two images, alone, is in general ill-posed. The parameterization of the registration is also a form of regularization through the choice of basis. A subspace regularization may be employed to restrict the generality of $\mathbf{u}$, for instance

$$\min \frac{1}{2}||T(\mathbf{x}+Q\mathbf{z})-R(\mathbf{x})||^2. \tag{3}$$

Besides minimizing the distance between images, one can minimize the entropy-related concept of "mutual information" [29], or normal gradient fields [15],

which is particularly useful for registration of images from two different modalities, such as CT and MRI, or a number of other objectives. Though the problem statements are simple, the solutions may not be unique.

Multilevel algorithms [14] seek to overcome the problem of multiple minima by successively registering the same pair of images at resolutions ranging from coarse to fine. While the mathematics is formal, the motivation is very intuitive and has been exploited in many areas of optimization that are plagued by multiple minima. Projections of the problem into coarse spaces obscures the local minima, which can only be differentiated at finer resolutions. As the fine structure is revealed, the registration settles into a particular local minimum, which is hopefully the global one. Figure 1 illustrates. If one imagines a properly rotationally oriented template hand being moved continuously over the reference hand, one can envision nine or more local minima, as first one, then two, then successively more fingers overlap, with the best minimum obtained in the middle when all five appendages overlap.

There are many types of registration objectives, including those based on colocating landmarks, aligning principal axes, correlating image intensities, minimizing elastic deformation, and conserving volume. While the variety is large, and the motivations that drive selecting among the choices beyond the scope of this overview, the optimization setting provides a mathematical framework that leads to the computational data structures and the types of operations that must be executed over them.

1.2 Computational characteristics of registration

Typical contemporary registration problems may be two- or three-dimensional, and may vary widely in discrete size from tens of KB to many GB per individual pixelated or voxelated image. Inheriting contemporary tools from computational optimization, the algorithmic building blocks of importance are Newton-like nonlinear solvers [19], Krylov linear solvers [13], multilevel [3] and other linear preconditioners [25] to solve the discretized problem, and multivariate spline interpolation [2] to allow the images to be compared at chosen sets of points. Large, sparse matrix methods predominate. Domain decomposition leads to advantageous surface-to-volume communication-to-computation ratios that permit weak scaling in an implementation sense [20]; see the next section for details. Multilevel preconditioners are capable of preserving weak scaling in a convergence sense [27].

1.3 Grid-based illustration of registration

To illustrate the potential for real-time registration on the Grid in a very preliminary way that is representative of, but not pushing, the state of the art, we consider a data set of 20 three-dimensional images of pig's heart, each $128 \times 128 \times 64$ 8-bit voxels (resolved on a gray scale from 0 to 255) or 8 MB per image. The images comprise a time series. One is taken as the reference, and 19 pairwise

registrations are performed to study the deformations of the heart. An SPMD code implemented in C and MPI to perform affine registration was provided by co-author Heldmann. Its input is a pair of images, the reference and one template, and the output of each execution is very compact – a set of 12 scalars describing the map. Such a compact output enables instant verification of the correct execution of the code in distributed environments.

The facility employed for the test is the Skynet cluster at ISI, a collection (at the time of writing) of 96 dual processor Intel P3 nodes, with a range of clock ratings from 800 to 1200 MHz. The Web Services Grid Resources Allocation and Management (WS-GRAM) [30] component of the Globus [12] toolkit is used to distribute the registration tasks. The GRAM service provides a single interface for requesting and using remote system resources for the execution of "jobs." The most common use of GRAM is remote job submission and control. It is designed to provide a uniform, flexible interface to job scheduling systems. GRAM does not provide scheduling or resource brokering capabilities. A wide variety of metaschedulers and resource brokers that leverage GRAM mechanisms have been developed by other projects. WS-GRAM was released in July 2006 and it is expected that the various metaschedulers and resource brokers of Globus will be integrated into it.

WS-GRAM provides rich job description and resource provisioning capabilities. Among other objectives, such as optimizing throughput by matching requirements to resources, WS-GRAM can allocate jobs so as to minimize expected overall latency, given an existing pool of resources. This is the sense in which we apply it here. Our assumption is that in medical applications, information derived from registration has time-value. If it does not return in time to enter into a physician's decision-making process, it may have little value at all. This suggests other problem formulations, which we will examine in the future, such as ordering the jobs so that the partial results of greatest value are returned first. This project (in progress) provides a preliminary demonstration of the registration application together with the WS-GRAM harness.

In the test reported here, each registration job was launched with 16-way parallelism at the MPI level, on 8 dual-procesor nodes. Up to 11 such parallel jobs could be executed at once, as mediated by the underlying PBS scheduler [23]. In the suboptimal, preliminary test reported on here, all jobs in a given batch must report complete before a subsequent batch is launched.

The result was the reduction in latency of a "sequential" processing of 19 MPI-parallel jobs that required 51.77 minutes to 6.95 minutes in the Grid environment, a speedup of 7.5.

Our near-term plans call for expansion of this registration test in many dimensions. The entire TeraGrid will be used as a resource pool. Our next data sets will be made up of images of size $512 \times 512 \times 512$, or 1GB each. Each pairwise registration job (as currently) will be run with tightly coupled SPMD parallelism. Jobs will be launched individually from a queue based on resource monitoring. A strategy will be developed for gaining the medically most relevant information first. Finally, in view of the size of the data sets, a strategy will be

developed for overcoming transmission latency that is synergistic with the multilevel character of the registration algorithm, as discussed above. Specifically, coarse-grid representations of the problem can be sent before the full fine-grid representations, so that processing can begin while the fine-grid representations are still en route.

Our longer term objective includes development of parallel algorithms for registration that are sufficiently asynchronous that the Grid environment can be employed even within a single registration. This is not yet necessary for medical images of contemporary sizes, but it is a technology driver. Such algorithms would be useful for many problems besides real-time registration, which is the subject of the second part of this paper.

2 The challenge of asynchronous algorithms for Grid hosting of PDE-based systems

The image registration challenge in the first part of this paper is addressed under the relevant, but restricted paradigm of SPMD bulk synchronous processing. There are numerous applications in addition to image registration in which it is desirable to exploit the Grid to occasionally grab exceptional amounts of memory for highly resolved simulations, or in which it is desired to take over exceptional levels of processing power. As an instance of the first, an exceptional simulation might use a close to first-principles model to calibrate a multiscale model, which would then be used for the majority of production at a center. As instances of the second, flood, firespread, or pathogen transport prediction models might need to be run ahead of real time in emergencies. Or real-time control of a massive experimental device, such as ITER [18], might be allowed to take over Grid-availed resources for the exceptional experimental "shot." In these contexts, we must consider individual large jobs, not a large ensemble of small jobs. Historically, the Grid has supported very few claims for success in this realm. PDE-based simulations are naturally implemented in bulk synchronous mode, in which the work load for each processing element is carefully matched to its capabilities, so that idleness is minimized at synchronization points.

Many issues must be addressed to make PDE-simulations a reality for the dynamic environment of the Grid. One of them is fault tolerance. However, in the limited scope of this contribution, we assume that processors and networks are reliable and seek algorithmic tolerance of dynamic performance, or actual synergisms between algorithms and the dynamic performance Grid environment.

2.1 Concurrency through domain decomposition

PDE-based codes are nearly universally parallelized with domain decomposition – applying a serial algorithm that (approximately or exactly) solves a PDE

on a given domain so that it solves for a subdomain's worth of data and nesting this inside of an iterative process that adjusts the values on the interfaces or overlaps between the subdomains to consistency. Such a decomposition preserves the volume(work)-to-surface(communication) ratio in the weak scaling limit as work per processor remains fixed and both the work and the number of processors are scaled in proportion. For domain decomposition methods to be weak-scalable, it is necessary that the individual problems on subdomains be well load balanced and that the number of outer iterations be bounded, independently of problem resolution and processor granularity (one implies the other) in the limit of weak scaling. Theory constructively showing how to obtain nearly resolution- and granularity-independent convergence rates is well developed for many problems [27] and freely available software exists that delivers this performance in practice [16, 24, 28]. The price of convergence-rate optimality is often the solution of auxiliary problems in reduced dimensional spaces. These auxiliary problems are not as scalable as the original union of independent fine-scale problems, and their implementation requires extreme care. Nevertheless, a families of multilevel iterative methods whose parameters can be tuned to application and architecture exist, which are mature in their analysis and software aspects. All such methods, however, presume predictable load per process since synchronization points are relatively frequent.

For nonlinear problems, a popular family of methods is Newton-Krylov-Schwarz (NKS) [5]. This is a triply-nested loop domain-decomposed algorithm, with an outer Newton loop, which evaluates a nonlinear residual vector at a given solution iterate and solves a linear system with the nonlinear residual vector as the right-hand side and the Jacobian of the residual as the system matrix. A multiple of the resulting solution is added to the current iterate. Krylov iteration is employed to solve the linear system with the Jacobian. Each Krylov loop begins with an iterate to which it applies a Jacobian-vector product and then computes some inner products, which determine coefficients with which to update the linear iterate. Both the outer Newton and inner Krylov loops are synchronous and essentially update a vector defined over the domain with AXPY operations. Inside of the Krylov loop, subspace iterations of Schwarz type are employed to precondition the linear system. The Schwarz iterations are generally a mixture of multiplicative and additive projections into subspaces, with the bulk of the work being done concurrently on each processor within subdomains. To summarize the NKS technique, within each level of the triply nested loop there is a decomposition into concurrent tasks by domain. Typically, one subdomain's worth of work is assigned to each process. Processes must communicate thin regions near their boundaries with neighbors, and they must cooperate globally in performing inner products (`AllReduce` commutatives) and in solving reduced-dimensional problems. Typically, one process is mapped to each processor, and processors synchronize at the `AllReduce` points.

2.2 Algorithmic adaptation

No computer system is well-balanced for all computational tasks since different tasks (such as concurrent neighbor communication, global reduction, concurrent local residual evaluation, concurrent local recurrences, etc.) stress different parts of the processor-memory-network system. By being aware of which algorithmic phases are limited by which aspect of hardware or system software, one can adapt algorithms to take advantage of the strengths or mitigate the weaknesses of given hardware. Detailed phase-by-phase performance analysis of an unstructured PDE-based code for aerodynamics led to the inspiration of the Gordon Bell "Special Prize," a share of which was first awarded to a subset of the co-authors in 1999 [1]. While many performance optimizations were studied in that and other papers for the massively parallel solution of PDE-based problems, one parasitic loss of performance that was noted in the strong scaling limit, but not addressed, was idleness at synchronization bottlenecks. This idleness was due to the difficulty of load balancing increasingly smaller and also increasingly less homogeneous partitions. For instance, as subdomains get smaller, the distribution of ratios of boundary and interface nodes to interior nodes gets broader. In the context of the Grid, these same synchronization points will lead to idleness for other reasons, even in weak scaling.

Historically, there have been a number of noteworthy adaptations to high latency and low bandwidth in parallel systems. Reduction of communication or replacement of communication with extra work will generally be useful adaptations in the Grid environment.

Garbey and Tromeur-Dervout [11] introduced the $C(p, j, q)$ schemes in an attempt to hide interprocessor latency by extrapolating data messages from neighboring processes in a time integration process, rather than waiting for the messages to appear before computing locally with their data as inputs. Rollbacks were used if the data upon arrival proved to be too inconsistent with the extrapolated values. Intuitively, this radical procedure has a chance of being successful, since for the problems considered, the extrapolations have to be correct in the low-spectrum eigenmodes only. Error in the the high-spectrum eigenmodes decays rapidly. For a given accuracy-work tradeoff, the technique has a payoff region.

Cai and Sarkis [6] introduced an algorithm called restricted additive Schwarz (RAS), which was discovered accidentally by turning off certain communications while debugging. Observing that not only efficiency per iteration but also convergence rate improved, the researchers were able to prove why, for many problems, the updates provided by the turned-off communication were unnecessary and even detrimental. RAS is now the default form of Schwarz preconditioning in PETSc [24].

Additive versions of algorithms are often available where multiplicative versions are the default. The additive versions may converge more slowly, but can sometimes be virtually as good as their mathematically more pedigreed cousins. The AFACx version of the asynchronous fast adaptive composite grid method

is shown in [21] to be nearly as good as AFAC in practice, for instance. Such algorithms have received a bad rap historically due to theory which can be pessimistic in the worst case [8].

Cai and Keyes [7] introduced a nonlinear form of Schwarz preconditioning, called Additive Schwarz Preconditioned Inexact Newton (ASPIN), which was motivated by nonlinear convergence theory, but turns out to relax the frequency of global synchronization in Newton-Krylov methods rather dramatically, while simultaneously unbalancing domain-decomposed work. Essentially, the method introduces process-scale Newton iterations inside of the outer Newton iteration. Most of the work of the nonlinear convergence shifts from the global outer iteration to the local inner iterations, with the result that the outer iteration converges in very few Newton steps, and therefore few global synchronizations. There is often a substantial reduction of work overall, but more importantly, the work that remains occurs in local subsets of communicating processors. However, unlike the traditional Newton-Krylov-Schwarz method, whose work-per-processor can be well load-balanced up to the limit of too small the average size of the subdomains on each processor, ASPIN has an unpredictable and potentially poorly balanced distribution of subdomain work. This is because the different nonlinear systems that are iterated to convergence on each subdomain may take different numbers of internal iterations. Fortunately, these internal iterations are not synchronized with those of other subdomains. The processors governing different subdomains synchronize only at outer loops. On a tightly coupled parallel machine, ASPIN is difficult to recommend, because of this feature of unbalanced inner loop work. However, the structure of the computation lends itself well to the Grid. In the spirit of Eisenhower's maxim that "one way to solve a big problem is to make it bigger," ASPIN folds its uncertain load imbalance into the uncertain performance guarantee of the processors in a Grid-based computation, and harvests cycles when available. The tools created as part of the WS-GRAM infrastructure to monitor and predict availability and so allocate work can be combined with tools that predict the work in ASPIN processes based on recent history so that a collection of ASPIN processes can in principle share resources synergistically with other Grid-enabled jobs.

Transcending particular algorithmic inventions, we also propose an asynchronous programming style that loans itself to many bulk synchronous algorithms that must confront inefficiency through idleness at synchronization points, whether due to internal work imbalance or external dynamic availability. The synchronization in many scientific simulation codes, including PDE-based codes, is artifactual. At a synchronization point there is often lots of work ready to perform whose data needs are completely local; however conventional programming styles do not allow an independent user thread from the same overall process to begin executing while the synchronizing thread is blocked.

The critical path in a Newton-Krylov code, abstracted to a sufficiently high level is: ..., `linear_solve`, `bound_step`, `update`, `linear_solve`, `bound_step`, `update`, ... We often insert into the critical path tasks that could be performed less synchronously, on which the tasks above do not critically depend, such as

refreshing the Jacobian with which the linear solver is performed, or refreshing the preconditioner for that Jacobian. It is well studied, theoretically and in practice, that Newton and Krylov methods are robust with respect to less frequent updates to these linear operators than once per step, under many circumstances. Convergence degrades if refreshing is long postponed, but for the sake of synchronization, either of these updates could be invested in to varying degrees. We also frequently insert global convergence testing and parameter adaptation on the critical path above more frequently than necessary. They can be partially completed on free cycles and thrown onto the critical path less frequently than is generally done today. Other tasks such as I/O, data compression or archiving, data visualization, or data mining, which must be or can be performed directly on the parallel cluster, represent useful work that is, to a significant degree, not tightly synchronized with respect to the solution loop above.

To take full advantage of such asynchronous algorithms, we need to develop greater expressiveness in scientific programming, by creating threads with relaxed data requirements and dynamically adjusting the relative priority of threads. This will require "associative communication" models such as the one recently addressed in [4].

3 Convergence of Grid computing and scientific computing

As illustrated in the context of medical image registration, there are numerous scientific applications today that can exploit the Grid in all of its heterogeneity without ill effect and to advantage, even in a hybrid model of nearly independent batched jobs each of which is a tightly coupled application. Tools such as WS-GRAM make it increasingly practical to schedule the independent jobs in such a way as to meet real time applications requirements, or at least to know when it is physically impractical to meet them, so that cycles are not wasted and alternatives not deferred. Many of the agendae of large-scale simulation share workflow characteristics with the image registration task considered herein. Computational science is not about individual large-scale analyses, done fast and "thrown over the wall." Both results and their sensitivities are desired; often multiple operation points to be simulated are known *a priori*, rather than sequentially. Sensitivities may be fed back into optimization process. Full PDE analyses may also be inner iterations in a multidisciplinary computation. In such contexts, "petaflop/s" may mean 1,000 analyses running somewhat asynchronously with respect to each other, each at 1 Tflop/s – clearly a less daunting challenge and one that has better synchronization properties for exploiting "The Grid" – than one analysis running at 1 Pflop/s.

However, even perfect knowledge of resource capabilities at every moment and perfect load balancers will not redeem the Grid for all SPMD implementations of PDE simulations. The cost of rebalancing is frequently too large to do

on the short intervals required in the dynamic environment of the Grid, and the Amdahl penalty for failing to rebalance is fatal. A combination of better Grid monitoring and allocation tools and less synchronous algorithms is required for the greatest ultimate success.

Less synchronous algorithms for traditionally tightly coupled PDE-based simulations are highly desirable for reasons independent of the Grid. For the petascale machines of which we expect to take delivery in 2008 and beyond, consisting of 10^5 and more processors, it will be highly desirable to have such methods.

Natural forces with both the Grid community and the PDE simulation community are converging independently towards a rendezvous that it is already practical at some scale. We are cautiously optimistic about a much more significant rendezvous ahead.

Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant Nos. CCF-0427464 to Columbia University, CCF-0427904 to Emory University, and CCF-0427912 to the University of Chicago.

References

1. Anderson, W. K, Gropp, W. D., Kaushik, D. K., Keyes, D. E. and Smith, B. F. (1999). *Achieving High Sustained Performance in an Unstructured Mesh CFD Application*, in "Proceedings of SC'99," IEEE Computer Society, Los Alamitos.
2. Bojanov, B., Hakopian, H. and Sahakian, B. (1993). *Spline Functions and Multivariate Interpolation*, Springer, New York.
3. Briggs, W. L., Henson, V. E. and McCormick, S. F. (2000). *A Multigrid Tutorial*, SIAM, Philadelphia.
4. Browne, J. C., Yalamanchi, M., Kane, K. and Sankaralingam, K. (2004). *General Parallel Computations on Desktop Grid and P2P Systems*, in "Proceedings of LCR 2004: Seventh Workshop on Languages, Compilers & Runtime Support for Scalable Systems," University of Houston, Houston.
5. Cai, X.-C., Gropp, W. D., Keyes, D. E. and Tidriri, M. D. (1993). *Newton-Krylov-Schwarz Methods in CFD*, in "Numerical Methods for the Navier-Stokes Equations" (F.-K. Hebeker et al., eds.), Vieweg, Braunschweig.
6. Cai, X.-C. and Sarkis, M. (1999). *A Restricted Additive Schwarz Preconditioner for General Sparse Linear Systems*, SIAM J. Sci. Comput. **21**:792–797.
7. Cai, X. C. and Keyes, D. E. (2002). *Nonlinearly Preconditioned Inexact Newton Algorithms*, SIAM J. Sci. Comp. **24**:183-200.
8. Chazan, D. and Miranker, W. (1969). *Chaotic Relaxation*, Linear Alg. Applics. **2**:199–222.
9. Chung, J., Haber, E. and Nagy, J. (2006). *Numerical Methods for Coupled Super-Resolution* Inverse Problems **22**:1261-1272.

10. Elad, M. and Feuer, A. (1997). *Numerical Methods for Coupled Super-Resolution* Inverse Problems **22**:1261-1272.
11. Garbey, M. and Tromeur-Dervout, D. (2000). *A Parallel Adaptive Coupling Algorithm for Systems of Differential Equations*, J. Comput. Phys. **161**: 401-427.
12. Globus (2006). http://www.globus.org/toolkit/.
13. Greenbaum, A. (1997). *Iterative Methods for Solving Linear Systems*, SIAM, Philadelphia.
14. Haber, E. and Modersitzki, J. (2006). *A Multilevel Method for Image Registration*, SIAM J. Sci. Comput. **27**:1594–1607.
15. Haber, E. and Modersitzki, J. (2006). *Intensity gradient based registration and fusion of multi-modal images*, Medical Image Computing and Computer-Assisted Intervention - MICCAI (2), pp. 726-733.
16. hypre (2006). http://www.llnl.gov/CASC/linear_solvers/.
17. ISI, Information Sciences Institute, University of Southern California (2006). *Breaking the Medical Image Communication Barrier*, http://www.isi.edu/news/news.php?story=152.
18. ITER (2006). http://www.iter.org/.
19. Kelley, C. T. (1995). *Iterative Methods for Linear and Nonlinear Equations*, SIAM, Philadelphia.
20. Keyes, D. E. (1998). *How Scalable is Domain Decomposition in Practice?*, in "Proceedings of the 11th Intl. Conf. on Domain Decomposition Methods" (C. H. Lai, et al., eds.), pp. 286–297, DDM.ORG, New York.
21. Lee, B., McCormick, S. F., Philip, B. and Quinlan, D. J. (2003). *Asynchronous Fast Adaptive Composite-Grid Methods: Numerical Results*, SIAM J. Sci. Comput. **25**:682–700.
22. Modersitzki, J. (2004). *Numerical Methods for Image Registration*, Oxford University Press, Oxford.
23. OpenPBS (2003). http://www.openpbs.org/.
24. PETSc (2006). http://www-unix.mcs.anl.gov/petsc/.
25. Saad, Y. (1996). *Iterative Methods for Sparse Linear Systems*, PWS Publishing, Boston.
26. Springer Verlag (2006). *Journals and New Books in Imaging*, 56 pp.
27. Toselli, A. and Widlund O. (2005). *Domain Decomposition Methods – Algorithms and Theory*, Springer, New York.
28. Trilinos (2006). http://software.sandia.gov/trilinos/.
29. Viola, P. (1995). *Alignment by Maximization of Mutual Information*, Ph.D. thesis, Massachusetts Institute of Technology.
30. WS-GRAM (2006). http://www.globus.org/toolkit/docs/3.2/gram/key/.