Past, Present and Future Scalability of the Uintah Software

Martin Berzins
Scientific Computing and Imaging Institute
University of Utah
Salt Lake City, UT 84112 USA
mb@cs.utah.edu

John Schmidt
Scientific Computing and Imaging Institute
University of Utah
Salt Lake City, UT 84112 USA
John.Schmidt@utah.edu

Qingyu Meng
Scientific Computing and Imaging Institute
University of Utah
Salt Lake City, UT 84112 USA
qymeng@cs.utah.edu

Alan Humphrey
Scientific Computing and Imaging Institute
University of Utah
Salt Lake City, UT 84112 USA
ahumphre@cs.utah.edu

ABSTRACT

The past, present and future scalability of the Uintah Software framework is considered with the intention of describing a successful approach to large scale parallelism and also considering how this approach may need to be extended for future architectures. Uintah allows the solution of large scale fluid-structure interaction problems through the use of fluid flow solvers coupled with particle-based solids methods. In addition Uintah uses a combustion solver to tackle a broad and challenging class of turbulent combustion problems. A unique feature of Uintah is that it uses an asynchronous task-based approach with automatic load balancing to solve complex problems using techniques such as adaptive mesh refinement. At present, Uintah is able to make full use of present-day massively parallel machines as the result of three phases of development over the past dozen years. These development phases have led to an adaptive scalable run-time system that is capable of independently scheduling tasks to multiple CPUs cores and GPUs on a node. In the case of solving incompressible low-mach number applications it is also necessary to use linear solvers and to consider the challenges of radiation problems. The approaches adopted to achieve present scalability are described and their extensions to possible future architectures is considered.

Categories and Subject Descriptors

D.1.3 [**Software**]: Concurrent Programming; G.1.8 [**Mathematics of Computing**]: Partial Differential Equations; G.4 [**Mathematics of Computing**]: Mathematical Software; J.2 [**Computer Applications**]: Physical Sciences and Engineering

Keywords

Uintah, parallelism, scalability, adaptive mesh refinement, linear equations

Blue Waters Workshop July 15-16, 2012, Chicago, IL, USA.

1. INTRODUCTION

The present consensus is that the move to multi-petaflop and eventually exascale computing will require step changes in many different computational and computer science areas in order to perform the large-scale computational science simulations successfully on as yet unknown computer architectures with multi-million-way parallelism. Computing the solution to such problems will require novel approaches for managing, processing and visualizing data sets at scales beyond which are manageable today. Despite uncertainty, it is clear that the time delay and energy cost of data movements will constitute a major bottleneck in future systems. Datasets will not be easily transferrable from the final multi-petaflop architecture to post-processing environments, and when such transfers will be possible an even greater data management challenge will be created on any post-processing infrastructure. The challenge is thus not only of running on emerging machines such as Blue Waters, Titan and Sequoia, but also in being able to run on subsequent generations of parallel machines whose characteristics [10] may include:

1. Communications will be hierarchical in nature and potentially subject to long delays, thus requiring the overlapping of communication with computation.
2. Compute nodes will have more cores and/or accelerators than at present.
3. Compute nodes will have less memory per core. Thus any global data will need to be shared on a node.
4. In order to achieve good future performance it will be necessary to tune the runtime system so as to detect and remove inefficiencies in scaling.
5. The probability of associated soft and hard hardware faults on such machines is expected to be much greater than at present and will require resilient hardware and software.
6. The need for energy efficiency requires that computational work be focused where it is needed by using techniques such as adaptive mesh refinement.
7. Careful orchestration and minimization of data movements needs to be a key design component in the software ecosystem since moving data is a major contributor to the energy

cost of a simulation. The time delays introduced by traditional bulk-data movements will introduce unacceptable utilization degradation of the overall computing resources allocated.

8. The need to solve linear systems of equations across all or parts of future systems presents a considerable challenge.

One of the main approaches suggested for the move to multi-petaflop architectures and eventually exascale is to use a graph representation of the computation to schedule work, as opposed to a bulk synchronous approach in which blocks of communication follow blocks of computation. The importance of this approach for exascale computing is expressed by [10, 17] *Exascale programming will require prioritization of critical-path and non-critical path tasks, adaptive directed acyclic graph scheduling of critical-path tasks, and adaptive rebalancing of all tasks with the freedom of not putting the rebalancing of non-critical tasks on the path itself.* This is the approach used in the Uintah framework, [24]. However achieving performance with such an approach is not guaranteed. In this paper the challenges of using such an approach are explored in the context of the Uintah software framework. This software framework was initially written as part of the CSAFE ASC center at Utah and is a sophisticated fluid-structure interaction code that is capable of scaling on large core counts. In particular we will provide a brief description of Uintah describel how its development proceeded in three distinct phases from the point of view of how scalable the code was at each stage. In the third and most recent phase the development of a new runtime system has made it possible to use both multi-core CPUs and accelerators such as GPUs. An important requirement for Uintah is to use implicit methods and so the same discussion with regard to scalability will be applied in this case too. The paper concludes with a discussion of how the software needs to develop with regard to addressing the points made by [10] above.

2. OVERVIEW OF UINTAH SOFTWARE

The Uintah Software framework originated from the University of Utah Center for the Simulation of Accidental Fires and Explosions (C-SAFE) [8, 27, 28], a Department of Energy ASC center. Uintah was designed for the simulation of multi-scale multi-physics problems, such as those arising from combustion and fluid-structure interaction applications. Uintah is open source software[1] released under the MIT open sourse model. An important feature of Uintah is the use of a component-based design that strictly enforces separation between those components and allows them to be interchanged and independently developed and tested. This inter-operability has led to the development of components that have been used to solve a wide variety of problems, see [5]. This component design has also enabled a strict and critical separation between the applications developers and the developers of the runtime system. Uintah makes use of four main discretization components. These are the ICE compressible multi-fluid/material method, the particle-based Material Point Method (MPM) for structural mechanics, the combined fluid-structure interaction algorithm MPMICE and the ARCHES combustion component. The first of these, ICE, was developed by Kashiwa and others at LANL [19] for incompressible and compressible flow regimes. The second method, the Material Point Method, is a particle method based on a particle-in-cell approach that discretizes solid materials applications involving complex geometries [14], large deformations [7] and fracture. The combination of ICE and MPM is termed MPMICE and is used to solve fluid-structure interaction problems. The general aproach used in MPMICE and the complex coupling algorithm between MPM and ICE is described in detail by [13, 15]. The fourth component is the ARCHES Large Eddy Simulation (LES) code developed by Prof. P.J. Smith and his group in Utah. This component is often used for many industrial and industrial-strength research simulations, [5] and uses a low-Mach number (Ma< 0.3), variable density formulation to simulate heat, mass, and momentum transport in reacting flows. The Large Eddy Simulation algorithm used in ARCHES solves the filtered, density-weighted, time-dependent coupled conservation equations for mass, momentum, energy, and particle moment equations in a Cartesian coordinate system [18]. ARCHES is a stencil-based p.d.e. code and so achieves scalability [29] through its use of the Uintah infrastructure. The low-Mach, pressure approach of ARCHES requires a solution of a pressure projection set of equations at every time step. Typically both the PETSc [4] and the hypre packages [2, 11] have been used to solve these systems in parallel in what can be the most computationally intensive part of the simluation. At present hypre is quite heavily used in this context.

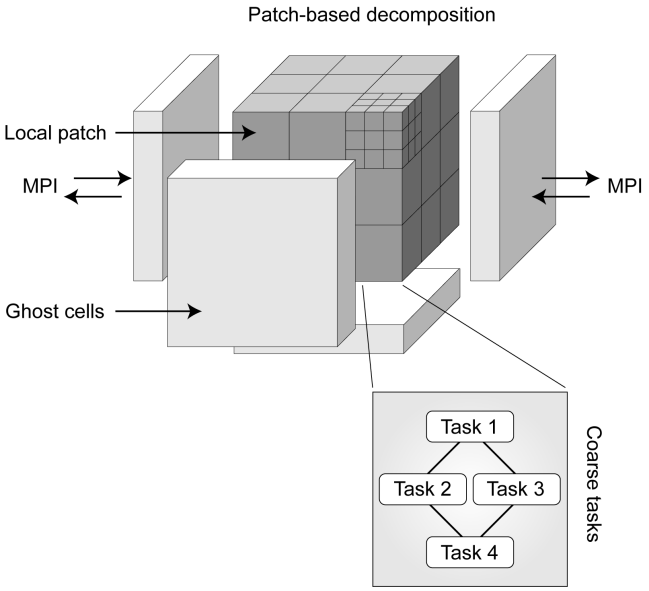

Figure 1: Nodal Patch Structure

3. UINTAH DEVELOPMENT PHASES

As mentioned above, an important feature of Uintah, is its use of a task-based paradigm, with complete isolation of the user from parallelism. The individual tasks that make up a component are viewed as part of a directed acyclic graph (DAG). While this approach provides flexibility there are also considerable challenges in ensuring that this approach works well. This challenge is reflected by the fact that the Uintah software has used three distinct and very different runtime systems to implement execution of the task graph.

3.1 Phase 1 (1998-2005)

The initial phase of the runtime system used a static task-graph approach that scaled up to a few thousand cores for fixed-mesh calculations, as is shown for example in [9]. The task graph approach enabled communication to be overlapped with computation. Global

[1] see `www.uintah.utah.edu`

data structures were relatively expensive and every MPI task had a copy of these data structures.

3.2 Phase 2 (2005-2010)

Motivated by the desire to use adaptive meshing, new load balancing methods and many improved and revised algorithms [22] were introduced, including a novel adaptive meshing approach, [21]. There was a move to dynamic execution of the task graph (including out-of-order) execution [25]. The new adaptive meshing approach is more local and together with dynamic execution of the task-graph enabled AMR to scale from about 12K cores to almost 100K cores, before data structure issues made it difficult to increase the underlying problem size further.

3.3 Phase 3 (2010-2013)

In attempting to make AMR in Uintah scale above 100K cores it was clear that there was a need to address global memory usage. Instead of each core on a node being an MPI process, one MPI process per node was used [24] with a multi-threaded scheduler being used to send tasks to cores. The underlying difference is shown in Figure 1 in which all mesh patches internal to a node no longer communicate using MPI but access data directly from the single copy of the data warehouse that Uintah uses to store variables on a node. This approach was very successful on Kraken [2] and the

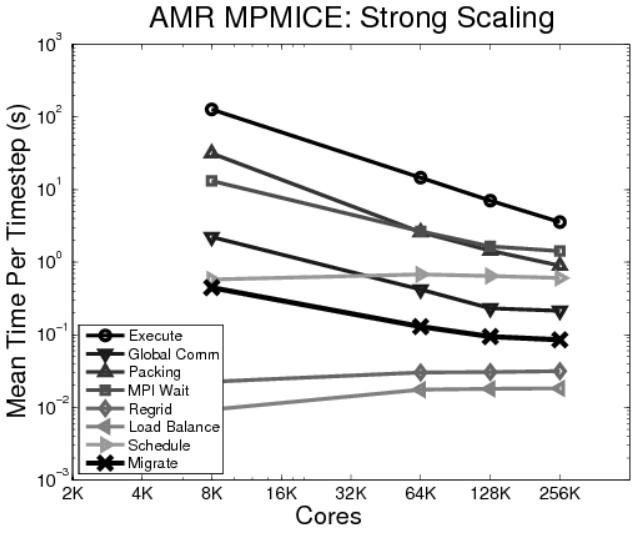

Figure 2: Analysis of scalability of Uintah components

Jaguar XT5. [3] In moving to the new form of Jaguar, the XK6, [4] the greater core count per node and the faster communications due to the new Gemini interconnect gave rise to reduced scalability. Figure 2 shows the strong scalability of a number of Uintah components, such as load balancing, scheduling, regridding, data migration etc for the AMR code with the ICE fluid-flow solver and the

[2]Kraken is an NSF supercomputer located at the University of Tennessee/ Oak Ridge National Laboratory with 112,896 cores.

[3]Jaguar is a DOE supercomputer located at the Oak Ridge National Laboratory with 224,256 cores that was in service until December 2011.

[4]Jaguar XK6 is a DOE supercomputer undergoing construction in 2012 at the Oak Ridge National Laboratory with approx 299,008 CPU cores and when a large number of attached GPUs are attached will be called Titan.

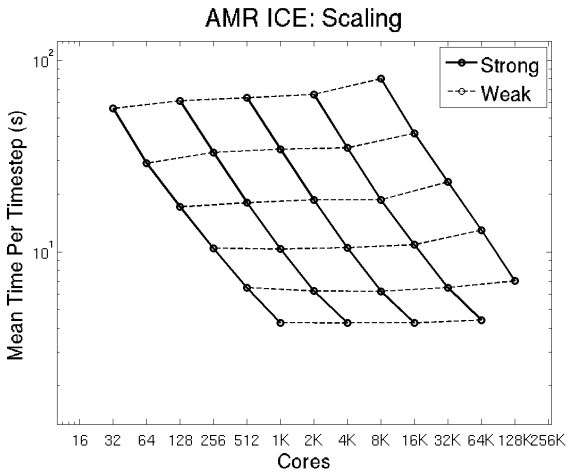

Figure 3: ICE-AMR Scaling

MPM component on the test problem in [23]. The motivating fluid-structure interaction problem used here arises from the simulation of explosion of a small steel container filled with a solid explosive. The benchmark problem used for this scalability study is the transport of a small cube of steel container inside of the PBX product gas at an initial velocity of Mach two. The simulation used an explicit formulation with the lock-step time stepping algorithm which advances all levels simultaneously. The refinement algorithm used tracked the interface between the solid and the fluid causing the simulation to regrid often while maintaining a fairly constant sized grid, which allows the scalability to be more accurately measured. This criteria led to each problem being about four times as large as the previous one. This problem exercises all of the main features of ICE, MPM and AMR and amounts to solving eight partial differential equations, along with two point-wise solves, and one iterative solve [6, 21]. There are two notable features in these results. The first is that the routine that creates the task graph schedule, listed as schedule is becoming problematic, but has not yet had an impact on scalability. The second feature is that the MPIWait time is growing in a way that destroys scalability. This was caused by the overloading of the single core being used to schedule tasks. The solution to this was to move to a distributed approach whereby each core pulls work when it needs it rather than having work centrally assigned to it. It was also necessary to take care to ensure that all the cores can access the single data structure without contention occurring. A shared memory approach that is lock-free is implemented by making use of hardware-based atomic operations on a read-only data structure and thus allows efficient access by all cores to the shared data on a node. These developments are described by [23], where it is also shown that the same calculation scales with the revised runtime system. The improved scalability may also be demonstrated when the ICE multi-material algorithm with explicit time stepping is used to simulate the transport of two fluids with a prescribed initial velocity. The fluids exchange momentum and heat through the exchange terms in the governing equations. This problem [21] exercises all of main features of ICE and amounts to solving eight P.D.E's, along with two point-wise solves, and one iterative solve. Figure 3 shows the strong and weak scaling of the AMR code with the ICE fluid-flow solver on the same standard test problem. While these results and those in [23] are very promising, it is also clear that it is important to be able to run codes such as Uintah on the accelerators of emerging machines such as Titan and

Blue Waters. In Uintah, accelerator task execution [16] on a node with both CPUs and GPUs is implemented through an extension of the runtime system that enables tasks to be executed efficiently (through pre-loading of data) on one or more accelerators per core, as has been demonstrated on the DOE TitanDev development system at ORNL and on the NSF Keeneland GPU system at NICS ORNL. In the case of the implementation of a radiation ray-tracing algorithm the GPUs on Keeneland are about 35x faster (Keeneland) or 70x faster (TitanDev) than a CPU core. For less-friendly stencil calculations the speedup is shown in Figure 4 for a computationally intensive kernel of the ICE code. In this case the superiority of the GPU emerges only for very large patch sizes. For almost all of the

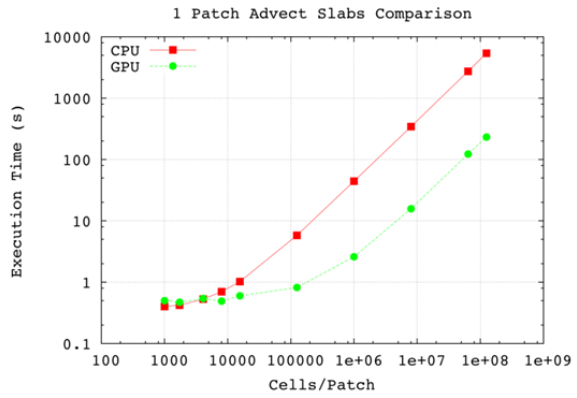

Figure 4: Performance comparison of ICE kernel on GPU and CPU

discussion above we have considered explicit calculations in which it was not necessary to solve a system of equations.

4. SCALABILITY WITH THE HYPRE LINEAR SOLVER IN UINTAH

As it is far from clear that the linear solvers available today will perform efficiently on such large core counts, as those on the Titan and Blue Waters machines, in this section the scalability of the Uintah software applied to incompressible flow problems when using the hypre software is addressed [3, 12]. This issue is described in detail in the unpublished report [30], while the emphasis here is on detailed scalability results and improved performance above that in [30].

The hypre software library [2] is designed for the iterative solution of linear systems of equations on large parallel machines. Hypre allows the use of multi-grid preconditioners, including a structured grid interface that is designed for stencil-based p.d.e. codes such as Uintah. The structured multigrid solver used here, PFMG, makes use of the mesh structure of Uintah and [1] and is a semi-coarsening multigrid method for solving scalar diffusion equations on logically rectangular grids. PFMG is used within hypre as preconditioner for iterative methods; The conjugate gradient method with a Red-Black Gauss-Seidel (RBGS) relaxation scheme inside PFMG is used in the experiments described below. It should be noted that the results in [30] use the less efficient Jacobi option than those presented here that also use the RBGS scheme.

As the setup phase of a method such as PFMG is computationally expensive, great care has to be taken to perform this step only when the underlying grid changes. Schmidt et al. [30] demonstrate the expense of setup and only perform this step once. The solution they adopt is to preserve the appropriate data structures from hype in the Data Warehouse that Uintah uses on each node.

The Data Warehouse manages data including the allocation, deallocation, reference counting, and ghost data exchanges which greatly simplifies data management for the simulation algorithm implementations [20]. Schmidt et al. [30] developed a new C++ templated variable type for Uintah that acted as a container object for the raw memory pointers from within the Data Warehouse and explain how this data structure may be used to hold the hypre data.

The test problems used by [30] were a model incompressible Taylor Green flow problem and an example taken from the modeling of Helium plumes. A complete description of these problems is given in [30]. While the Taylor-Green vortex problem provides a good model problem in this section the scalability associated with the more challenging Helium Plume problem is considered. In each case the solution of the incompressible Navier-Stokes equations given by

$$\frac{\partial \rho}{\partial t} + \nabla \cdot \rho \mathbf{u} = 0 \tag{1}$$

$$\frac{\partial \rho \mathbf{u}}{\partial t} = \mathbf{F} - \nabla p; \quad \mathbf{F} \equiv -\nabla \cdot \rho \mathbf{u}\mathbf{u} + \mu \nabla^2 \mathbf{u} + \rho g \tag{2}$$

The velocity vector $\mathbf{u} = (u_x, u_y, u_z)$ gives the speed of fluid particles in three orthogonal directions, ν is the kinematic viscosity - a fluid property that reflects its resistance to shearing forces, ρ is the fluid density, and p is the pressure.

The numerical solution of these equations requires evaluation of the pressure field while enforcing the continuity constraint given by (1). Pressure however is only implicitly defined by the above equations and so one approach is to derive an explicit pressure equation is to take the divergence of (2) and make use of (1) to act as the constraint. This approach provides a Poisson equation for the pressure:

$$\nabla^2 p = \nabla \cdot \mathbf{F} + \frac{\partial^2 \rho}{\partial t^2} \equiv R \tag{3}$$

where ρ is the density and $\mathbf{F}$ is given by [30]. Equation (3) is large and sparse and is known as the pressure-Poisson-equation (PPE) and is solved here by making use of hypre.

The Helium plume example used here is challenging as it both requires the full solution of the incompressible Navier Stokes equations and uses sub-grid models to account for any unresolved turbulence scales. The model is representative of a real fire, has experimental validation data [26], and thus is an important test problem for ARCHES and Uintah, [30].

For this simulation, the mesh was composed of multiple patches with 32^3 cells per patch. One patch per core was used and the resolution and domain sizes were increased to achieve a constant workload as the core count increased. For the smallest core count case of 192 cores, the approximately 6.2 million (192×32^3) unknowns were solved using hypre with the PFMG pre-conditioner and the Jacobi smoother. The largest core count case of 256K cores required the solution over 8.5 billion ($256K \times 32^3$) unknowns.

A model for the weak scalability of the linear solver time as a func-

tion of the number of cores (C) can be described by a simple power law: $time = a * C^m$. Taking logarithms gives

$$log(time) = log(a) + m * log(C). \quad (4)$$

and performing a linear least squares fit yields the coefficients for each problem as shown in Figure 5.

This figure shows both the raw results and the linear least squares fit of the power law equation in two cases. The first case is a run on Jaguar XK6 with a Jacobi smoother up to 256K cores. The second case is the same problem on the older slower Kraken machine up to 96K cores with the RBGS smoother. In the case of the Kraken simulation more than twice as many unknowns per patch were used, namely 75K as opposed to 32K. The results suggest that the scalability of the linear solver depends significantly on the iterative method employed. In the case when an RBGS method is used with the Helium plume the weak scalability scales roughly to the power of $\frac{1}{6}, m = 0.15$. The scalability of the Helium Plume problem with the Jacobi method used for relaxation in the preconditioner is quite different scaling approximately to the power of $\frac{1}{2}$. The power of roughly $\frac{1}{6}$ is also observed by [30] for a simpler model problem. The power of $\frac{1}{6}$ is also almost parallel to logarithmic behavior as given by the reference line showing $log(c)/30$ in the figure. The results shown used here may also appear to be suggestive of the scalability of the linear solvers for future machines. In order to confirm this hypothesis, experiments were undertaken to measure the computational cost of the individual components of hypre. The times for some of these components, PFMG, semi-restriction, point relaxation and overall Red Black Gauss Seidel are shown in Figure 6. The values of m for these routines lie in the range $[0.13, 0.16]$. As expected the actual pointwise iteration routine has a value of $m = 0.04$. Similar results exist for the conjugate gradient part of hypre. These results show that the main components of hypre exhibit the same scalability as the solver as a whole. This is a helpful result with regard to using large core counts.

5. CONCLUSIONS AND FUTURE WORK

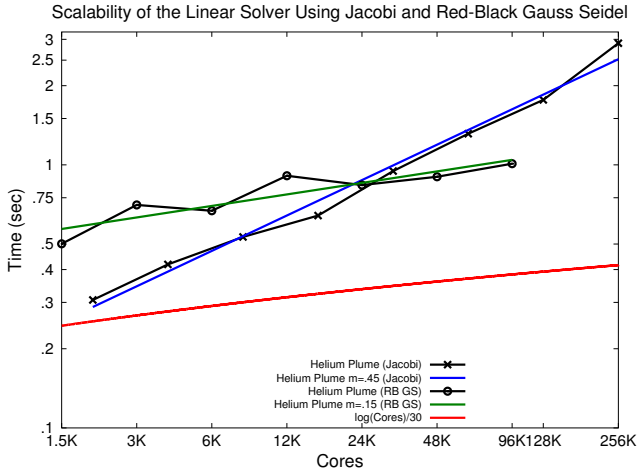

Figure 5: Scalability of the Linear Solvers with a Least Squares Fit

This paper has shown how the asynchronous task-based approach of Uintah is successful in making complex applications involving fluid-structure interaction adaptive mesh refinement and combustion scale. In particular we have also shown how the Uintah approach addresses the eight points raised in the introduction. Uintah has consistently overlapped computation and communication and through dynamic execution of its task graph has been able to offset communications delays (point 1). The new combined CPU-GPU runtime system of Uintah makes it possible to utilize large numbers of cores and GPUs while minimizing energy usage (points 2 and 3). While Uintah does not self-tune itself to achieve better scaling the detailed trace information available at every stage of the calculation makes it possible to identify scalability problems and to improve the runtime system (point 4). Uintah already uses mesh refinement at large scale (point 6) but could clearly do more to increase energy efficiency. While Uintah efforts are ongoing to use data streaming approaches, there is much to be done(point 7). Finally we have a clear understanding of how to achieve scalability with state-of-the-art linear solvers (point 8). A major challenge is to include fault tolerance into how Uintah undertakes its calculations (point 5), although major changes are needed at the operating system and MPI levels before this is possible. The approach appears to have promise in conjunction with efficient parallel linear solvers and with accelerator-based architectures. It is also clear that it is very important in this context to continue developing efficient run-time systems.

6. ACKNOWLEDGMENTS

This work was supported by the National Science Foundation under subcontracts No. OCI0721659, the NSF OCI PetaApps program, through award OCI 0905068 and by DOE INCITE awards CMB015 and CMB021 for time on Jaguar and DOE NETL for funding under NET DE-EE0004449, Uintah was written by the University of Utah's Center for the Simulation of Accidental Fires and Explosions (C-SAFE) and funded by the Department of Energy, subcontract No. B524196. We would like to thank the hypre team for offering assistance and insight into maximizing the full capabilities of hypre. We would like to thank all those previously involved with Uintah and TACC, NICS and Oak Ridge for access to large numbers of cores.

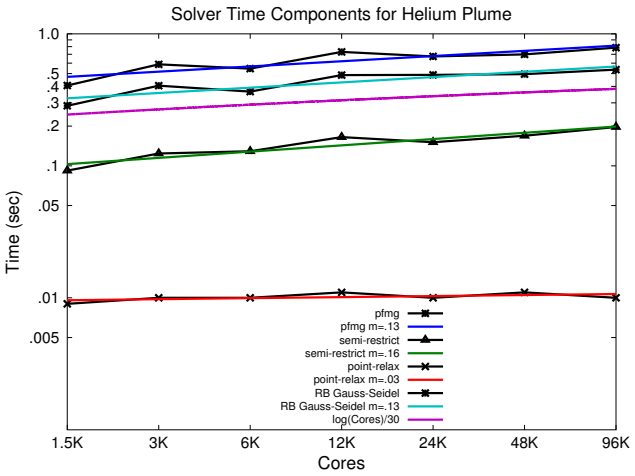

Figure 6: Scalability of the Hypre components with a Least Squares Fit

7. REFERENCES

[1] S. F. Ashby and R. D. Falgout. A parallel multigrid preconditioned conjugate gradient algorithm for groundwater flow simulations. *Nulclear Science and Engineering*, 124(1):145–159, 1996.

[2] A.H. Baker, R.D. Falgout, T. Kolev, and U.M. Yang. Scaling hypre's multigrid solvers to 100,000 cores. In M.W. Berry, K.A. Gallivan, E. Gallopoulos, A. Gram, B. Philippe, Y. Saad, and F. Saied, editors, *High-Performance Scientific Computing*, pages 261–279. Springer London, 2012.

[3] A.H. Baker, T. Gamblin, M. Schulz, and U.M. Yang. Challenges of scaling algebraic multigrid across modern multicore architectures. In *IPDPS*, pages 275–286, 2011.

[4] S. Balay, J. Brown, K. Buschelman, W.D. Gropp, D. Kaushik, M.G. Knepley, L.C. McInnes, B.F. Smith, and H. Zhang. PETSc Web page, 2011. http://www.mcs.anl.gov/petsc.

[5] M. Berzins. Status of release of Uintah computational framework. Technical Report SCI UUSCI-2012-001, SCI Institute University of Utah, 2012.

[6] M. Berzins, J. Luitjens, Q. Meng, T. Harman, C.A. Wight, and J.R. Peterson. Uintah - a scalable framework for hazard analysis. In *Proceedings of the Teragrid 2010 Conference, TG 10*, number 3, page (published online), July 2010. Awarded Best Paper in the Science Track!

[7] A. D. Brydon, S. G. Bardenhagen, E. A. Miller, and G. T. Seidler. Simulation of the densification of real open–celled foam microstructures. *J. Mech. Phys. Solids*, 53:2638–2660, 2005.

[8] J. D. de St. Germain, J. McCorquodale, S. G. Parker, and C. R. Johnson. Uintah: A massively parallel problem solving environment. In *Ninth IEEE International Symposium on High Performance and Distributed Computing*, pages 33–41. IEEE, Piscataway, NJ, November 2000.

[9] J.D. de St. Germain, A. Morris, S.G. Parker, A.D. Malony, and S. Shende. Performance analysis integration in the uintah software development cycle. *International Journal of Parallel Programming*, 31(1):35–53, 2003.

[10] D.L.Brown and P.Messina et al. Scientific grand challenges: Crosscutting technologies for computing at the exascale. Technical Report Report PNNL 20168, US Dept. of Energy Report from the Workshop on February 2-4, 2010 Washington, DC, 2011.

[11] R.D. Falgout, J.E. Jones, and U.M. Yang. The design and implementation of hypre, a library of parallel high performance preconditioners. In *Numerical Solution of Partial Differential Equations on Parallel Computers*, pages 267–294. Springer-Verlag, 2006.

[12] R.D. Falgout and U.M. Yang. hypre: A library of high performance preconditioners. In Peter M. A. Sloot, Chih Jeng Kenneth Tan, Jack Dongarra, and Alfons G. Hoekstra, editors, *International Conference on Computational Science (3)*, volume 2331 of *Lecture Notes in Computer Science*, pages 632–641. Springer, 2002.

[13] J. E. Guilkey, T. B. Harman, A. Xia, B. A Kashiwa, and P. A. McMurtry. An Eulerian-Lagrangian approach for large deformation fluid-structure interaction problems, part 1: Algorithm development. In *Fluid Structure Interaction II*, Cadiz, Spain, 2003. WIT Press.

[14] J. E. Guilkey, J. B Hoying, and J. A. Weiss. Modeling of multicellular constructs with the material point method. *Journal of Biomechanics*, 39:2074–2086, 2007.

[15] T. B. Harman, J. E. Guilkey, B. A Kashiwa, J. Schmidt, and P. A. McMurtry. An Eulerian-Lagrangian approach for large deformationfluid-structure interaction problems, part 1:multi-physics simulations within a modern computationalframework. In *Fluid Structure Interaction II*, Cadiz, Spain, 2003. WIT Press.

[16] A. Humphrey, Q. Meng, M. Berzins, and T. Harman. Radiation modeling using the uintah heterogeneous cpu/gpu runtime system. SCI Technical Report UUSCI-2012-003 (to appear in Proc. XSEDE 2012), SCI Institute, University of Utah, 2012.

[17] J.Ang and K.Evans et al. Workshop on extreme-scale solvers: Transition to future architectures. Technical Report USDept. of Energy, Office of Advanced Scientific Computing Research. Report of a meeting held on March 8-9 2012, Washington DC, 2012.

[18] J.Spinti, J. Thornock, E. Eddings, P. Smith, and A. Sarofim. Heat transfer to objects in pool fires, in transport phenomena in fires. In *Transport Phenomena in Fires*, Southampton, U.K., 2008. WIT Press.

[19] B. A. Kashiwa. A multifield model and method for fluid-structure interaction dynamics. Technical Report LA-UR-01-1136, Los Alamos National Laboratory, Los Alamos, 2001.

[20] J. Luitjens. *The Scalability of Parallel Adaptive Mesh Refinement Within Uintah*. PhD thesis, University of Utah, 2011.

[21] J. Luitjens and M. Berzins. Improving the performance of Uintah: A large-scale adaptive meshing computational framework. In *Proceedings of the 24th IEEE International Parallel and Distributed Processing Symposium (IPDPS10)*, page (accepted), 2010.

[22] J. Luitjens, M. Berzins, and T. Henderson. Parallel space-filling curve generation through sorting: Research articles. *Concurr. Comput. : Pract. Exper.*, 19(10):1387–1402, 2007.

[23] Q. Meng and M. Berzins. Scalable large-scale fluid-structure interaction solvers in the uintah framework via hybrid task-based parallelism algorithms. SCI Technical Report UUSCI-2012-004 (submitted for publication), SCI Institute, University of Utah, 2012.

[24] Q. Meng, M. Berzins, and J. Schmidt. Using hybrid parallelism to improve memory use in Uintah. In *Proceedings of the Teragrid 2011 Conference*. ACM, July 2011.

[25] Q. Meng, J. Luitjens, and M. Berzins. Dynamic task scheduling for the uintah framework. In *Proceedings of the 3rd IEEE Workshop on Many-Task Computing on Grids and Supercomputers (MTAGS10)*, 2010.

[26] J. O'Hern, E.J. Weckman, A.L. Gerharti, S.R. Tieszen, and R.W. Schefer. Experimental study of a turbulence buoyant helium plume. Technical Report SAND2004-0549, Sandia National Laboratories, 2004.

[27] S. G. Parker. A component-based architecture for parallel multi-physics PDE simulation. *Future Generation Comput. Sys.*, 22:204–216, 2006.

[28] S. G. Parker, J. Guilkey, and T. Harman. A component-based parallel infrastructure for the simulation of fluid-structure interaction. *Engineering with Computers*, 22:277–292, 2006.

[29] R.Rawat, J. Spinti, W.Yee, and P. Smith. Parallelization of a large scale hydrocarbon pool fire in the uintah pse. In *ASME 2002 International Mechanical Engineering Congress and Exposition (IMECE2002)*, pages 49–55, November 2002.

[30] J. Schmidt, M. Berzins, J. Thornock, T. Saad, and J. Sutherland. Large scale parallel solution of incompressible flow problems using uintah and hypre. SCI Technical Report UUSCI-2012-002, SCI Institute, University of Utah, 2012.