SCI INSTITUTE
TECHNICAL REPORT

Large Scale Parallel Solution of Incompressible Flow Problems using Uintah and hypre

John Schmidt, Martin Berzins, Jeremy Thornock, Tony Saad, James Sutherland

UUSCI-2012-002

Scientific Computing and Imaging Institute
University of Utah
Salt Lake City, UT 84112 USA

May 4, 2012

Abstract:

The Uintah Software framework was developed to provide an environment for solving fluid-structure interaction problems on structured adaptive grids on large-scale, long-running, data-intensive problems. Uintah uses a combination of fluid-flow solvers and particle-based methods for solids together with a novel asynchronous task-based approach with fully automated load balancing. As Uintah is often used to solve compressible, low-Mach combustion applications, it is important to have a scalable linear solver. While there are many such solvers available, the scalability of those codes varies greatly. The hypre software offers a range of solvers and pre-conditioners for different types of grids. The weak scalability of Uintah and hypre is addressed for particular examples when applied to an incompressible flow problem relevant to combustion applications. After careful software engineering to reduce start-up costs, much better than expected weak scalability is seen for up to 100K cores on NSFs Kraken architecture and up to 200K+ cores, on DOEs new Titan machine.

THE UNIVERSITY OF UTAH

Large Scale Parallel Solution of Incompressible Flow Problems using Uintah and hypre

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

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

Jeremy Thornock
Institute for Clean and Secure Energy
University of Utah
Salt lake City, UT 84112 USA
Jeremy.Thornock@utah.edu

Tony Saad
Institute for Clean and Secure Energy
University of Utah
Salt lake City, UT 84112 USA
Tony.Saad@utah.edu

James Sutherland
Institute for Clean and Secure Energy
University of Utah
Salt lake City, UT 84112 USA
James.Sutherland@utah.edu

ABSTRACT

The Uintah Software framework was developed to provide an environment for solving fluid-structure interaction problems on structured adaptive grids on large-scale, long-running, data-intensive problems. Uintah uses a combination of fluid-flow solvers and particle-based methods for solids together with a novel asynchronous task-based approach with fully automated load balancing. As Uintah is often used to solve compressible, low-Mach combustion applications, it is important to have a scalable linear solver. While there are many such solvers available, the scalability of those codes varies greatly. The hypre software offers a range of solvers and pre-conditioners for different types of grids. The weak scalability of Uintah and hypre is addressed for particular examples when applied to an incompressible flow problem relevant to combustion applications. After careful software engineering to reduce start-up costs, much better than expected weak scalability is seen for up to 100K cores on NSF's Kraken architecture and up to 200K+ cores, on DOE's new Titan machine.

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, hypre, parallelism, scalability, linear equations

1. INTRODUCTION

In this paper we consider solving a pressure projection formulation of the Navier-Stokes equations within Uintah [8, 12, 29, 30], an open-source software framework (`www.uintah.utah.edu`). In recent previous work we showed that Uintah scales well to about 200K cores for some applications, [8, 24], including a sympathetic explosion modeling problem, funded by the NSF PetaApps Program, that is one of our main applications driving examples. In these cases the solution techniques were all fully explicit and did not require the solution of a system of equations. In many of the Uintah applications the flow is compressible and chemically reacting [17, 18] with the formulation of the governing equations requiring the use of a linear solver at each timestep [13]. Such problems arise at Institute for Clean & Secure Energy Program, Utah, (led by Professor P.Smith) whose mission is to perform research to utilize the energy stored in our domestic resources and do so in a manner that will capture CO_2. This Uintah-based research is organized around the theme of validation and uncertainty quantification through tightly coupled simulation and experimental designs. Typical problems consist of the prediction of the performance and stability of oxygen-fired burners in boilers and industrial furnaces for CO_2 capture or determining flare combustion efficiency and VOC content of flares through combination of online measurements and high-fidelity codes, [17, 33]. In tackling such problems, as we approach problem sizes requiring greater than 100K cores on machines such as Kraken [1] and Jaguar, [2] it is far from clear that the linear solvers available today will perform efficiently on such large core counts, never mind on architectures such as the proposed Titan machine. [3] In this paper we will address the scalability of the Uintah software applied to incompressible flow problems when using one particular class of solvers from the hypre software. [2, 3, 14]. The test problems used will be a model incompressible flow problem and on a challenging example taken from the modeling of Helium plumes. The rest of this paper is organized as follows. The Uintah software, including the ARCHES and Wasatch software used for incompressible flow combustion prob-

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
XSEDE'12 July 16-20, 2012, Chicago, IL, USA.
Copyright 2012 ACM 978-1-4503-0888-5/11/07...$10.00.

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

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

[3] Titan is a DOE supercomputer undergoing construction in 2012 at the Oak Ridge National Laboratory with approx 299,008 cpu cores and a large number of attached GPUs.

lems, is described in Section 2. Section 3 describes the hypre software and how it is used in conjunction with Uintah including the improved performance obtained by avoiding start-up costs by preserving data structures. In Section 4 the results of numerical experiments are given that show the weak scaling obtained on two test problems. The first problem is a straightforward test problem while the second is a more challenging example that is representative of real applications. Section 5 presents a model for the weak scalability of the linear solver. Section 6 provides conclusions and described future work in extending the ideas here to other applications and Uintah components.

2. OVERVIEW OF UINTAH SOFTWARE

The Uintah Software was originally a product of the University of Utah Center for the Simulation of Accidental Fires and Explosions (C-SAFE) [12]. C-SAFE, a Department of Energy ASC center. The aim of Uintah was to be able to solve complex multi-scale multi-physics problems, with a particular emphasis on combustion and fluid-structure interaction problems. Uintah is available in open source software[4] and extended to run on about 200K cores [8, 24], Uintah makes use of a component design that enforces separation between large entities of software and can be swapped in and out, allowing them to be independently developed and tested within the entire framework. This has led to a very flexible simulation package that has been able to simulate a wide variety of problems, see [7]. A novel feature of Uintah, and one that directly leads to its scalability is its use of a task-based paradigm, with complete isolation of the user from parallelism. The individual tasks are viewed as part of a directed acyclic graph (DAG) and are executed adaptively, asynchronously and now often out of order [25]. Uintah uses a novel adaptive meshing approach, [23], as well as a particle solution methods, however neither of these components is used here.

Uintah currently contains four main simulation algorithms, or components, the ICE compressible multi-material CFD formulation, the particle-based Material Point Method (MPM) for structural mechanics, the combined fluid-structure interaction algorithm MPMICE [15] and the ARCHES combustion component. ICE is a "multi-material" CFD algorithm that was originally developed by Kashiwa and others at LANL [19] for incompressible and compressible flow regimes. The Material Point Method is a particle method that is used to evolve the equations of motion for the solid materials applications involving complex geometries [16], large deformations [11], and fracture.

The focus in this paper is on the fixed-mesh ARCHES and Wasatch components designed for simulation of compressible turbulent reacting flows with participating media radiation [26, 28, 34]

2.1 The ARCHES Combustion Component

ARCHES is a three-dimensional, Large Eddy Simulation (LES) code developed by Prof. P.J. Smith and his group in Utah. The ARCHES code is currently used for a broad class of industrial and industrial-strength research simulations, [7]. The ARCHES component 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, 20].

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

This set of filtered equations is discretized in space and time and solved on a staggered, finite volume mesh. The staggering scheme consists of four offset grids, one for storing scalar quantities and three for each component of the velocity vector. Spatial discretization is handled with central differencing where appropriate for energy conservation or flux limiters (e.g., scalar mixture fractions) to maintain realizability. The low-Mach, pressure formulation requires a solution of an implicit pressure projection at every time sub-step. The solution of these equations has been tackled with a number of different solvers. Most recently both the PETSc [4–6] and the hypre packages [13] have been used.

ARCHES uses a dynamic, large eddy turbulence closure model for the momentum and species transport equations is used. The dynamic model accounts for sub-grid velocity and species fluctuations. Various combustion models exist within ARCHES for doing gas phase and particle phase combustion chemistry, including sub-grid turbulence and chemistry interactions. For gas phase combustion, the dimensionality of the problem is reduced by parameterizing the thermo-chemical state space $(\rho, T, x_1, x_2, ..., x_{n-1})$, where x_i represents a chemical species (from set n), into a small set of variables which are tracked on the resolved mesh. These parameters then map back the state space as a function of space in time through the computational mesh. Presumed PDF methods are used to describe sub-grid heterogeneity that may exist. The combustion chemistry is pre-processed in a tabular form for dynamic table look-up during the course of the LES simulation. The energy balance includes the effect of radiative heat-loss/gains in the IR spectra by solving the radiative intensity equation using a discrete-ordinance solver [20]. The solution procedure solves the intensity equation over a discrete set of ordinances and, like the pressure equation, is formulated as a linear system that is solved using a linear solver package such as hypre. Solid particulate fuel phases are represented using the direct quadrature method of moments (DQMOM) This Eulerian reference frame approach solves a discrete set of moments of the number density equation using numerical quadrature. The number density function is described by the solid fuel properties including particle number, size, and fuel properties (e.g., velocity, volatile mass fraction, fixed fuel mass fraction, energy content, size). Various physical models for coal combustion are implemented to include the effect gas devolatilization, char oxidation, particle drag, and size changes, which in turn effect the coal number density function. The DQMOM description of the coal phase is completely coupled with the gas phase description to produce a completely coupled, gas/solid phase description of the flow with closed mass, momentum, and energy balances.

The ARCHES code is essentially a stencil-based p.d.e. code and so achieves good weak and strong scalability for its discretization by making use of the Uintah infrastructure [31, 32]. The potential bottleneck to scalability arises from the use of parallel solvers like hypre [2, 13] and PETSc, [21], as this is where most of the compute time is spent on incompressible flow calculations.

2.2 Wasatch Component

The extension to ARCHES is an approach, named Wasatch, proposed by one of the authors (Sutherland), see [9] that is a Domain Specific Language (DSL) for the very large and complex p.d.e. systems that arise is certain combustion applications. The complexity of the combustion problems to which simulation is applied naturally increases with available computing power. For example, turbulent combustion simulation of typical fuels involves perhaps hundreds of species and quite possibly thousands of reactions. Solving

such large sets of highly coupled, nonlinear PDEs may be problematic. In such highly dynamic, multi-physics systems, one may not be able to determine *a priori* the most appropriate models. One possible solution to this is to make use of programming models which allow significant flexibility in the complex couplings that may occur for different model sets in multi-physics applications.

In the approach proposed by Sutherland, [27], the programmer writes pieces of code that calculate various mathematical expressions, explicitly identifying what data the code requires and produces/calculates. To create an algorithm, the programmer selects one or more expressions to be evaluated and the dependencies are recursively "discovered" resulting in a dependency graph. The dependency graph may be inverted to obtain the execution graph, which may be traversed in parallel if desired. Wasatch uses an operator approach over strongly typed fields to form a domain specific language to achieve abstraction of field operations, including application of discrete operators such as interpolants, gradients, etc. This level of abstraction allows the programmer to work with fields in a MATLAB (vectorized) style while maintaining full compile-time type safety, ensuring that only valid field-field and operator-field operations can be performed. Furthermore, this level of abstraction also allows vectorized field operations to be automatically dispatched in parallel transparently to the programmer.

The major advantage is that Wasatch provides a potential extra level of parallelism at the level of the Uintah task being executed. While the Uintah run-time system has the capability of dealing with the appropriate scheduling of tasks, the Wasatch component can make sure that each task executes as efficiently as possible on each core or accelerator. In this work Wasatch was used to automatically generate code for the discretization of the first examples used to test the Uintah interface to hypre.

3. USING HYPRE WITH UINTAH

The hypre software [2, 3, 14] is a library of high performance preconditioners and solvers for the solution of large linear systems of equations on massively parallel machines. The hypre library has an emphasis on multi-grid preconditioners, including algebraic multigrid. While hypre has three conceptual interfaces [2] the interface that is most applicable here is the Structured Grid Interface (Struct). This interface is designed for stencil-based p.d.e. codes that use grids, such as those in Uintah, that consist of unions of logically rectangular (sub)grids. The specialized structured grid multigrid solvers in hypre make use of the conceptual interface to introduce problem-specific algorithmic aspects by taking advantage of the the structure of the problem and are thus the most scalable part of hypre [2]. In particular the structured multigrid solver used here, PFMG, adopts this approach.

PFMG [1] is a semi-coarsening multigrid method for solving scalar diffusion equations on logically rectangular grids discretized with up to 9-point stencils in 2D and up to 27-point stencils in 3D, [2] with anisotropies that are uniform and grid-aligned throughout the domain. The solver is designed to deal with anisotropies and so attempts to automatically determines the *best* direction of semi-coarsening. PFMG interpolation is determined algebraically. The coarse-grid operators are also formed algebraically, either by Galerkin or by the alternative method [1] for 5-point (2D) and 7-point (3D) problems. PFMG has many options to deal with solution anisotropies. Baker et al. [2] report that various version of PFMG are between 2.5 and 7 times faster than the equivalent algebraic multigrid (AMG) options inside hypre because they are able to take account of structure.

In general a multigrid method such as PFMG has a setup phase and a solve phase. The setup phase consists of defining the coarse grids and the interpolation and coarse grid operators. This phase can be computationally very intensive as will be demonstrated below. In contrast the solve phase which performs the multigrid cycles is often less expensive.

Note that in the use of hypre described here multigrid methods are not used as linear solvers but are used as as a preconditioners for Krylov or other iterative methods. For this reason the hypre library provides a number of common general-purpose iterative solvers. In the cases considered here, the Conjugate Gradient (CG) method was used with the PFMG preconditioner based upon a Jacobi relaxation method inside the structured multigrid approach.

3.1 hypre Setup Costs

As mentioned above, hypre setup (preconditioning and communication) time is significant at large core counts upwards to 20X that of the solve phase. The setup phase is slower than the solve phase, due primarily to the assumed partition and global partition components of the code. The global partition requires $O(PlogP)$ communications in the setup phase, while the current implementation of the assumed partition requires $O((logP)^2)$ communications, [2] also reported setup phase scales quite poorly, likely due to the $O((logP)^3)$ number of coarse grids. It is important to configure hypre to not use a global partition especially for very large core count runs.

Tables 1 and 2 show the start-up costs for a simple Poisson equation solve in two space dimensions as described in Section 4.1. We see a considerable growth in start-up costs in this weak scaling case in which each core has roughly the same computational load in terms of equations and unknowns. Experiments were performed on both the old XT5 and new XK6 configurations of Jaguar/Titan. At the time of the experiments this system consists of approximately 262,144 AMD Interlagos cores arranged so that there are 16 cores per node and connected through the new Cray Gemini interconnect. It is interesting to note that the set-up costs are greater for the slower interconnect found on the older XT5 configuration. The set-up overhead is a significant component of the overall solution time for each linear solve.

Cores	2K	4K	8K	16K	32K	64K	128K
Start-Up	0.02	0.13	0.25	0.25	0.50	0.75	3.89

Table 1: hypre start-up times for 2,048 to 131,072 Titan cores

Cores	3K	6K	12K	24K	48K	96K	192K
Start-Up	1.54	1.10	1.22	2.04	2.4	2.9	22.11

Table 2: hypre start-up times for 3,072 to 196,608 Jaguar cores

The initial use of hypre with Wasatch and ARCHES involved doing matrix allocation and setup at every timestep. The high setup costs on each solve made both weak and strong scalability impossible to achieve. As a result work was undertaken to ensure that the interface for the pressure solve to the external hypre library is handled efficiently. This has lead to code speed-up by removing redundant operations.

3.2 Avoiding hypre Setup Costs Using Uintah's Data Warehouse

The solution that was first implemented was to note that for the case with an incompressible formulation for the pressure solve found in both ARCHES and Wasatch, the matrix did not change between timesteps and so the setup only needed to be done as often as the mesh changed. In the case of the fixed mesh calculations reported here the setup only needed to occur at the beginning of the simulation.

Originally, hypre was used as a black box where coefficients for the matrix and right hand side vector were passed to hypre. Hypre then used these coefficients to create the global matrix and partitioning graph for all the processors. Once the solve was completed and the solution vector passed back to the Uintah component, the global matrix and partitioning was thrown away only to be created again on the subsequent timestep. For modest size core counts, this setup phase was not a significant component of the overall solution time. However, as the core counts increased, the setup time became the dominant factor.

While allowing hypre to completely control the memory allocation allowed for a certain simplicity in the initial implementation of the integration of hypre with Uintah, ultimately a more cooperative memory management strategy needed to be implemented. Within Uintah, the DataWarehouse is the key component for storing data for all calculations. The DataWarehouse is essentially a large hash table that stores pointers to allocated blocks of memory. From a application point of view, the vast majority of data are represented by one of four types of grid variables, i.e. node, face, cell, and particles. The node, face, and cell grid variables present a high level interface to the underlying I,J,K block of raw memory that represents the patch/patches on a given processor. The DataWarehouse manages the complete lifetime of data including the allocation, deallocation, reference counting, and ghost data exchanges which greatly simplifies data management for the simulation algorithm implementations [22].

While this simplifies native Uintah application developments, it does complicate the interaction between external third party software packages such as hypre. To provide a more cooperative memory management strategy, we developed a new C++ templated variable type (SoleVariable<Type>) for Uintah that acted as a container object for the raw memory pointers from within the DataWarehouse. Unlike the typical Uintah variable, the SoleVariable<T> was not dependent on the underlying grid/patch layout, instead it represented data that encompassed the entire grid.

For maximal memory efficiency and reuse, the following hypre data structure needed to be allocated with its lifetime intelligently managed by the DataWarehouse. At the beginning of the simulation and then reused on all subsequent linear system solves: solver, preconditioner, A matrix, B and X vectors. These five data types can be combined into a straightforward reference counted C++ structure:

```
struct hypre_solver_struct {
  HYPRE_StructSolver solver;
  HYPRE_StructSolver precond_solver;
  HYPRE_StructMatrix* HA;
  HYPRE_StructVector* HB;
  HYPRE_StructVector* HX;
};
```

From within the Uintah, this data structure would be the template argument to the SoleVariable as in:

```
SoleVariable<struct hypre_solver_struct>
```

The pointers to the matrix and vectors are then used in the native hypre APIs to do the initial memory management and the set-up of the communication patterns. Once the various hypre data structures were created, the SoleVariable container object managed the lifetime and reuse. For the computational experiments presented below, the setup phase and data structure creation phase were done prior to the first timestep. However, some of simulation components do create matrices that have different coefficient matrices and/or the sparsity patterns that differ from timestep to timestep. In these instances, the user can specify how often the set-up phase should occur. As will be shown below, finer grain control of memory and key hypre data structures allowed for improved weak scaling for very large core counts.

In the case of combustion problems involving radiation, the external hypre solves for the pressure and radiative intensity equations represent the bulk of the overall solution time.

4. COMPUTATIONAL EXPERIMENTS

In this section we describe two computational experiments that illustrate the weak scalability that is possible when using hypre as the linear solver for incompressible flow calculations. For each experiment, the solution of the Navier-Stokes equations is required where the Navier-Stokes equations describe the spatio-temporal motion of a fluid and are given by

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

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

Here, $\mathbf{u} = (u_x, u_y, u_z)$ is the velocity vector describing 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 the Navier-Stokes equations requires evaluation of the pressure field while enforcing the continuity constraint given by (1). The key difficulty that every CFD code must address is the implicit specification of the pressure in the Navier-Stokes equations. One approach to derive an explicit equation for the pressure is to take the divergence of (2) and make use of (1) to act as the constraint. At the outset, one obtains a Poisson equation for the pressure

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

Equation (3) is known as the pressure-Poisson-equation (PPE). Its solution requires the use of a solver such as hypre for large sparse systems of equations.

4.1 Taylor Green Vortex Example

The Taylor-Green vortex is an excellent benchmark for the verification of CFD codes [10]. It simplifies the system by assuming constant density so that the continuity equation reduces to

$$\nabla \cdot u = 0. \tag{4}$$

The initial condition is a divergence-free (solenoidal) velocity field that evolves in time according to the Navier-Stokes equations.

To test this formulation, we call upon the 3D Taylor-Green vortex. The initial condition is given by the following specification

$$u_x(\mathbf{x}, t=0) = \frac{2}{\sqrt{3}} \sin(\theta + \tfrac{2}{3}\pi) \sin x \cos y \cos z \quad (5)$$

$$u_y(\mathbf{x}, t=0) = \frac{2}{\sqrt{3}} \sin(\theta - \tfrac{2}{3}\pi) \cos x \sin y \cos z \quad (6)$$

$$u_z(\mathbf{x}, t=0) = \frac{2}{\sqrt{3}} \sin\theta \cos x \cos y \sin z \quad (7)$$

where θ is an arbitrary phase angle. If the PPE equation is correctly implemented, then the evolution of the flow from this solenoidal initial condition should remain solenoidal for all subsequent times.

In the current study, we set $\theta = 0$ and solve the Navier-Stokes equations on the domain $0 \leq (x, y, z) \leq 2\pi$ with periodic conditions. We used a first-order forward-Euler method for time integration. This may be summarized as follows

$$\mathbf{u}^{n+1} = \mathbf{u}^n + \mathbf{F}^n - \frac{1}{\rho}\nabla p^n \quad (8)$$

with the following PPE

$$\nabla^2 p^n = \rho \nabla \cdot F^n \quad (9)$$

For spatial discretization, we used second order central differencing on a staggered grid. A staggered grid is one on which the velocity components are located on staggered cells while all scalars (including pressure) are located at cell centroids. Furthermore, all fluxes are stored at the faces of scalar or staggered volumes.

For each 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, 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 192K cores required the solution over 6.4 billion ($196,068 \times 32^3$) unknowns.

In the experiments that follow not only was the new Titan machine used but the calculation was also done but also the older Jaguar machine at Oak ridge before it was upgraded to Titan. For the older XT5 configuration, the core count increased as multiples of 12 reflecting the 12 cores per node. Results for core counts ranging from 192 cores up to 196,068 cores were obtained prior to the transition from the XT5 configuration to the XK6 configuration. The new Titan machine did not have its GPUs yet and so has only 16 cores per node and thus the nodes may be viewed as XE6 nodes with only half the cores or XK6 nodes without the GPUs. For simulations running on the new XK6 configuration, the core count increased by multiples of 16 reflecting the 16 cores per node up to a maximum of 131,072 cores that were available at the time of the experiments. Figure 1 shows the weak scalability of Wasatch on a test incompressible flow problem - The Taylor-Green Vortex problem as described by equations (1-3).

Not only is the new machine faster per core but the improved communications network results in better weak scaling (constant workload per core) than on the original Jaguar. The top line shows the total average cpu time per timestep on Jaguar with a roughly constant load per core. The amount of time spent in the hypre solver is the next line down. The bottom two lines correspond to the same times on the new XK6 type nodes (as mentioned above without GPUs) in the Titan machine.

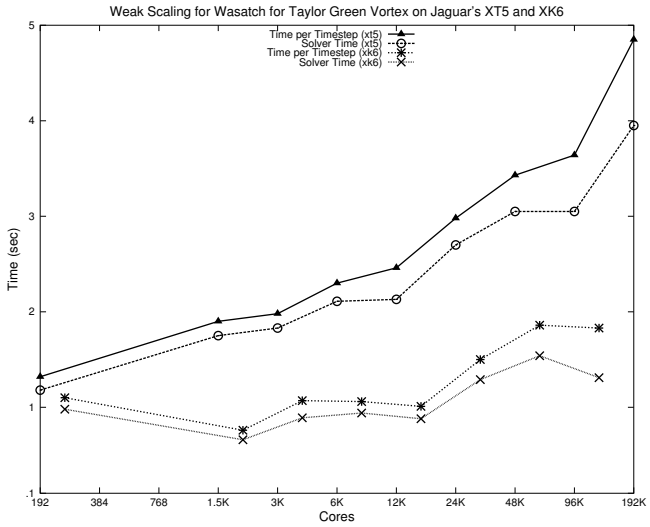

Figure 1: Weak scalability of Wasatch with hypre using the the PFMG pre-conditioner for the Taylor-Green Problem

4.2 Helium Plume problem

While the Taylor-Green vortex is an excellent illustration of a simple incompressible flow problem, it specifically neglects the effects of density variation which are paramount in combustion applications. With this in mind the second example is a more demanding Helium plume problem, which requires the full solution of Equations 1 and 2 including the density variations. In addition, various sub-models must be activated to account for any unresolved turbulence scales that are not directly resolved by the computational mesh.

The helium plume represents the essential characteristics of a real fire without introducing the complexities of combustion and thus serves as an important validation problem for the ARCHES code. Well-characterized experimental data collected by O'Hern et al. [28] at the Sandia FLAME facility in Albuquerque, NM serve as the experimental measurements. ARCHES simulations are validated against these data and serve to evaluate the various algorithmic procedures and turbulence closure models that are used within ARCHES.

The computational scenario consists of a $3m^3$ domain with a $1m$ opening that introduces the helium into a quiescent atmosphere of air with a co-flow of air. Velocity and density conditions at these boundaries are known. The sides and top of the computational cube are modeled using pressure and outlet boundary conditions respectively. As the name suggests, the outlet boundary condition encourages flow to leave the domain while the pressure conditions encourage air flow to enter (as driven by the buoyancy forces) through the side of the domain. Both the outlet and pressure conditions are driven by the resulting pressure field solution. The CFD solution procedure exercises major components of the overall algorithm, including the modeling of small, sub-grid turbulence scales. Additionally, the coupled problem combines the effects of fluid flow and turbulent scalar mixing for a full spectrum of length and time scales without introducing the complications of combustion reactions. CFD results are compared to time-averaged velocity of both

vertical and horizontal components, mixture fraction variables, and turbulence statistics. For this more complex problem the XK6 does not show such an advantage over the XT5 per core unless we factor in the smaller number of floating point units per core on the XK6.

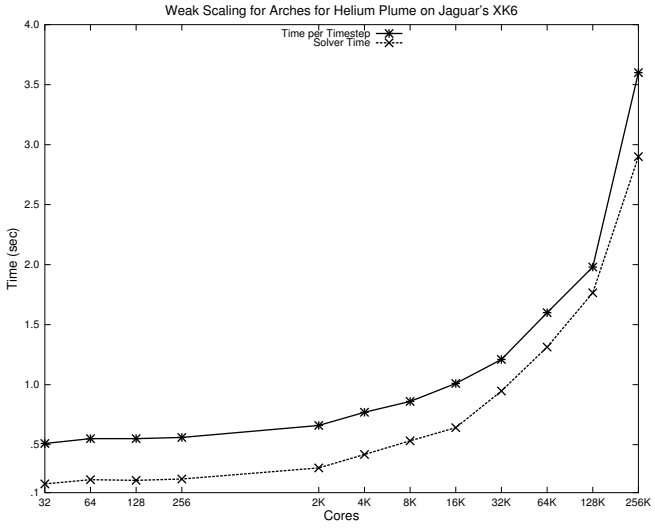

Figure 2: Weak scaling for ARCHES using hypre with the PFMG pre-conditioner for the Helium Plume Problem

The first results using hypre on Oak Ridge's new successor to Jaguar are shown in Figure 2. While the weak scaling is not perfect it is acceptable for large scale runs and amenable to improvement.

5. LINEAR SOLVER WEAK SCALABILITY MODEL

A model for the weak scalability of the linear solver time as a function of the number of cores (C) can be described by a simple power law:

$$time = a * C^b \tag{10}$$

taking the logarithm of each side

$$\log(time) = log(a) + b * log(C). \tag{11}$$

and performing a linear least squares fit yields the coefficients for each problem shown in Table 3.

Problem	a	b
Taylor-Green (xt5)	0.496	0.165
Taylor-Green (xk6)	0.182	0.179
Helium Plume (xk6)	0.0095	0.447

Table 3: Least Squares coefficients for a and b in (10)

Figure 3 shows the linear least squares fit of equation 11. The results suggest that the scalability of the linear solver depends significantly on the system of equations that are being solved for. What is interesting to note is that while the network infrastructure is quite different from the XT5 and the XK6 machines, the scalability of the linear solver for the Taylor-Green case is quite similar and scales roughly to the power of $\frac{1}{6}$. The scalability of the Helium Plume problem is quite different scaling approximately to the power of $\frac{1}{2}$. The difference in power coefficients is related to the fact that a 2D stencil defines the linear equation in the Taylor-Green case while a 3D stencil defines the matrix in the helium plume case. The difference in the work done by the PFMG/Jacobi method used in hype dictates the differences in timing.

The power law relationship for linear solver scalability demonstrates a remarkably accurate predictive model over a very wide distribution of core counts. It appears to be suggestive of the scalability of the linear solvers for future machines.

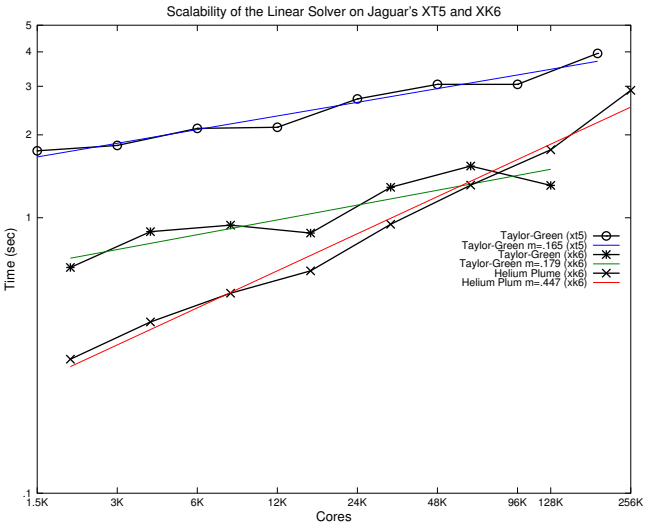

Figure 3: Scalability of the Linear Solvers with a Least Squares Fit for (11)

6. CONCLUSIONS AND FUTURE WORK

The overall conclusion of this paper is that it is possible to get quite reasonable weak scaling at large core counts when using hypre with Uintah. The only proviso is that care has to be taken with the start-up costs of hypre. In the case of Uintah this involved some careful software engineering to ensure continuity of the hypre work spaces. The results show that effective use of the hypre linear solver package has led to better-than-expected weak scalability on 100K+ cores of Uintah on incompressible flow problems that range from a simple example problem to an industrial-strength helium plume problem.

In addition, a simple power law relationship for the weak scalability of the linear solver is developed and shows a good range of applicability from the smallest core counts to the full machine capacity. Further work is needed to determine the applicability of the results to the scalability of the linear solvers for future machines and offers insight into the upper limits of current algorithms.

The future work in this area for the Uintah code is to extend this approach to the ICE component. This is quite challenging as in the

implicit form of ICE the matrix that arises depends on the density and so will have entries that vary from timestep to timestep. The sparsity pattern of the matrix will not change unless adaptive mesh refinement is used. In this case, one would want to only do the set-up phase as infrequently as possible. The convergence rate for the linear solver is determined by the quality of the preconditioner. Using preconditioners derived from previous timesteps with coefficient matrices that are too far *different* from the current matrix may require more many more iterations to converge to an acceptable solution than if a set-up phase calculation were to be performed. A possible algorithmic approach would be to collect data on the number of iterations as a function of the setup frequency. Then extrapolate from this data when the time to solution with a high iteration count would exceed the time taken to do the set-up phase and solve. Estimating this cross-over point would yield the upper bound for number of timesteps to take before a set-up phase calculation is performed. As the simulation advanced, the data collection and set-up phase frequency estimate would have to be continually monitored and updated to reflect the dynamic nature of the simulation.

7. 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 award CMB015 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.

8. 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, V. Eijkhout, W.D. Gropp, D. Kaushik, M.G. Knepley, L.C. McInnes, B.F. Smith, and H. Zhang. PETSc users manual. Technical Report ANL-95/11 - Revision 3.2, Argonne National Laboratory, 2011.

[5] 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.

[6] S. Balay, W.D. Gropp, L.C. McInnes, and B.F. Smith. Efficient management of parallelism in object oriented numerical software libraries. In E. Arge, A. M. Bruaset, and H. P. Langtangen, editors, *Modern Software Tools in Scientific Computing*, pages 163–202. Birkhäuser Press, 1997.

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

[8] M. Berzins, J. Luitjens, Q. Meng, T. Harman, C.A. Wight, and J.R. Peterson. Uintah - a scalable framework for hazard analysis. In *TG '10: Proceedings of the 2010 TeraGrid Conference*, New York, NY, USA, 2010. ACM.

[9] M. Berzins, Q. Meng, J. Schmidt, and J.C. Sutherland. DAG-based software frameworks for PDEs. In M. Alexander et al., editor, *Proceedings of Euro-Par 2011 Workshops, Part I*, Lecture Notes in Computer Science (LNCS) 7155, pages 324–333. Springer-Verlag Berlin Heidelberg, August 2012.

[10] M.E. Brachet, D.I. Meiron, S.A. Orszag, BG Nickel, R.H. Morf, and U. Frisch. Small-scale structure of the Taylor-Green vortex. *Journal Fluid Mechanics*, 130(41):1452, 1983.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] J. E. Guilkey, T. B. Harman, and B. Banerjee. An Eulerian-Lagrangian approach for simulating explosions of energetic devices. *Computers and Structures*, 85:660–674, 2007.

[16] 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.

[17] P. J.Smith, M. Hradisky, J. Thornock, J. Spinti, and D. Nguyen. Large eddy simulation of a turbulent buoyant helium plume. In *Proceedings of the 2011 companion on High Performance Computing Networking, Storage and Analysis Companion*, SC '11 Companion, pages 135–136, New York, NY, USA, 2011. ACM.

[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] G. Krishnamoorthy, S. Borodai, R. Rawat, J. P. Spinti, and P. J. Smith. Numerical modeling of radiative heat transfer in pool fire simulations. In *ASME 2005 International Mechanical Engineering Congress (IMECE2005)*, November 2005.

[21] G. Krishnamoorthy, R. Rawat, and P. Smith. Parallelization of the p-1 radiation model. *Numerical Heat Transfer Part B: Fundamentals*, 49(1):1–17, 2006.

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

[23] 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.

[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] P. Moin, K. Squires, and S. Lee. A dynamic subgrid-scale model for compressible turbulence and scalar transport. *Phys. Fluids*, 3(11):2746–2757, 1991.

[27] P.K. Notz, R.P. Pawlowski, and J. C. Sutherland. Graph-based software design for managing complexity and enabling concurrency in multiphysics pde software. ACM Transactions on Mathematical Software (accepted).

[28] 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.

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

[30] 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.

[31] 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.

[32] P. Smith, R. Rawat, J. Spinti, S. Kumar, S. Borodai, and A. Violi. Large eddy simulation of accidental fires using massively parallel computers. In *AIAA-2003-3697, 18th AIAA Computational Fluid Dynamics Conference*, June 2003.

[33] P. J. Smith, J. Thornock, D. Hinckley, and M. Hradisky. Large eddy simulation of industrial flares. In *Proceedings of the 2011 companion on High Performance Computing Networking, Storage and Analysis Companion*, SC '11 Companion, pages 137–138, New York, NY, USA, 2011. ACM.

[34] J. Spinti, J. Thornock, E. Eddings, P. Smith, , and A. Sarofim. *Transport Phenomena in Fires*, chapter Heat Transfer to Objects in Pool Fires. Wit Press, 2008.