Development of the Uintah Gateway for Fluid-Structure-Interaction Problems

John A. Schmidt
School of Computing
University of Utah
Salt Lake City, UT
john.schmidt@utah.edu

Martin Berzins
School of Computing
University of Utah
Salt Lake City, UT
mb@cs.utah.edu

ABSTRACT

The Uintah Gateway was designed to allow users to create complex multi-physics Uintah simulations with ease and to run these on large parallel computers. We used the Django web application framework to develop the Uintah Gateway for fluid-structure-interaction problems. We describe using the Gateway from input file creation to data management. We also describe two use cases: one involving a complex fluid-structure interaction problem with multiple simulations and multiple restarts and the other involving novice users just getting started with Uintah. Preliminary results suggest that both novice and advanced productivity increased dramatically using the Gateway.

Keywords

Uintah, Globus, Django, Fluid Structure Interaction, Heat Transfer

1. INTRODUCTION

The Uintah Software Framework arose from the University of Utah's Center for the Simulation of Accidental Fires and Explosions (C-SAFE) [10], a Department of Energy ASC center, that focused on providing state-of-the-art, science-based tools [3] for the numerical simulation of accidental fires and explosions. The benchmark C-SAFE problem was a multi-physics, large deformation, fluid-structure problem; a small cylindrical steel container filled with a plastic bonded explosive (PBX9501) subjected to convective and radiative heat fluxes from a fire. The incident heat flux caused the PBX to rapidly decompose into a gas above a critical temperature. The solid-to-gas reaction pressurized the interior of the steel container causing the shell to rapidly expand and eventually rupture. The gaseous products of reaction formed a blast wave that expanded outward along with pieces of the container and unreacted PBX.

With NSF SDCI funding, C-SAFE's science based simulation capability was transformed into the open-source release of the Uintah Software Framework (`www.uintah.utah.edu`) for complex multi-scale multi-physics problems [18]. Uintah makes use of a component design that has also allowed it to excel as a research platform. Components can be swapped in and out, allowing them to be developed and tested within the entire framework, without affecting other components. This has led to a highly flexible simulation package that has been able to simulate a wide variety of problems including shape charges, stage-separation in rockets, the biomechanics of microvessels [6], the properties of foam under large deformation [1], and the evolution of large pool fires caused by transportation accidents [14], in addition to the exploding container described above. Uintah has been used on single processor workstations all the way up to the largest TeraGrid resources (simulations using 99,072 cores).

Uintah can be viewed at two basic levels, an underlying infrastructure support role that defines the notion of particles, grids, time stepping, parallelization, load balancing and a layered component system that embodies the physical models and equations that define each component. At its most basic level, each Uintah component solves partial differential equations on block structured adaptive meshes. A team of computer scientists and engineers worked together to design a generic and robust system that partitioned the problem space into orthogonal components consisting of application specific code, i.e. algorithms for PDEs and infrastructure code. The primary goal of this two phased approach was to provide a system where the scientist/engineer would only be concerned with implementing algorithms to solve governing equations using a combination of particle based methods, and structured grid techniques. The component writer did not have to implement explicit message passing calls, nor be concerned with any aspect of parallelization or load balancing instead focusing entirely on efficient algorithm development and implementation. Whereas computer scientists working on the underlying infrastructure were focused primarily on scalability concerns including load balancing [17], partitioning, message passing, and data output. The powerful and efficient decoupling of infrastructure from application components have allowed for new applications to be developed and implemented quickly. Developers immediately benefit from any improvements [15] in the underlying infrastructure without requiring any changes to the individual component code. When advances were made in the infrastructure that allowed Uintah to scale to over 50K cores on Ranger[1] and 99,072 cores on Kraken[2], each computation component immediately scaled and ran at these core counts [15].

Uintah is one of the few large scale software systems for structured AMR multi-phyics simulations that scales well on the largest publicly available supercomputers in the TeraGrid. However, for the new user, installing Uintah can be difficult especially on large scale platforms with unique environments. Once Uintah is installed,

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, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
TeraGrid '10 Pittsburgh, Pennsylvania USA
Copyright 20XX ACM X-XXXXX-XX-X/XX/XX ...$10.00.

[1]Ranger is a NSF supercomputer located at the University of Texas with 62976 cores

[2]Kraken is a supercomputer located at the University of Tennessee with 99,072 cores

the new user must then navigate the batch system that is often different from one platform to the next. And when a user gains experience with Uintah and the general TeraGrid experience, the user then experiences data management issues. Instead of doing more science, users often spend more time administering, organizing, and managing data.

Our vision of the Uintah Gateway was to improve the user experience for both novice and advanced users that would offer solutions to the problems that confront each user population. As we looked to expand the scope and reach of Uintah and encourage the broad adoption to the scientific and engineering community the necessary step was to implement a Gateway that would provide new users a way to quickly get up to speed using Uintah while allowing our most advanced users a collection of tools that would streamline the data management issues that result from any large simulation study. The Uintah Gateway offers new and experienced users the opportunity to conduct some of the most powerful multi-physics on some of the most powerful supercomputers in the world from the convenience of a web browser offers a unique opportunity to transform the way computational science is performed.

The format of the rest of the paper is to discuss our previous end user experience with Uintah, discussion of the actual Gateway development using the Django web front-end based on user requirements, followed by case studies of advanced usage and novice usage, and concluding with an overview of how the Gateway is actually used.

2. REFLECTIONS FROM USING UINTAH ON LARGE SCALE PLATFORMS

Based on our previous thirteen years of using Uintah in supercomputing environments, we have encountered some challenges that surround large scale computing. First and foremost, long running simulations requiring multiple restarts generating hundreds of gigabytes of data present real challenges to even the most seasoned users. Data management tasks often dominant the time spent performing simulations. These tasks include categorizing, organizing, migrating data from scratch space to long term storage, or migrating a subset of the data to a local machine for further analysis. Compound this activity with multiple simulations that arise from any kind of parameter study and the task of managing this data glut often dominants the user experience.

Our users often want to visualize the simulation either during the course of a long running simulation or at the end. Invariably, the user must download a subset of the data to ensure that the simulation is proceeding as expected. With data sets that range from Gigabytes to Terabytes downloading even a fraction of the data presents difficulties that include issues such as data storage on the end user's machine, speed of data transfer, etc. Ideally, data movement should never take place. Instead users should rely on the visualization capabilities of the remote resource, but in practice our users would routinely seek to migrate data back from various supercomputing resources including the TeraGrid to do visualization locally. One of the design goals of the Uintah Gateway was to encourage our users to do simple remote visualizations by providing an interface to the command line options for VisIT while still providing the ability to migrate data back to a local resource for visualization purposes.

From the new user perspective, we typically found that new users experienced problems installing Uintah and running Uintah from the command line through the various batch systems. For those that come from the point and click world of graphical user interfaces, the command line world of Uintah and supercomputing in general presented challenges to the user that required a paradigm shift in the way they conducted their work. If the new user wanted to try out Uintah, they first had to install it, then figure out the batch system which usually varied from one supercomputer site to another and finally, they had to learn new data exploration tools. The users that did make it past this stage usually required many months of mentoring and dedicated efforts on their part before they got to the point where they were productively carrying out simulation science.

The overarching goal of the Uintah Gateway is to bring a world class multi-physics simulation environment running at petascale levels to new users while helping advanced users manage the ensuing data. Eliminating installation difficulties and hiding batch submission behind a web based front end allows new users to quickly submit and manage jobs on multiple processes. Although the new user must still understand the input file format (XML based input file) and various options used to describe a simulation, the proficiency level is enhanced by the elimination of the installation problem and the batch submission problem.

As we moved from Uintah software development and testing to actually use for very large scale simulation studies (using as many as 99,072 cores on Kraken) for a variety of problem cases, it became apparent that scientists and engineers needed some kind of interface to the TeraGrid to improve their productivity. The Uintah Gateway is unique in the sense that very long running simulations encompassing multiple restarts, management of hundreds of Gigabytes of data, and remote scientific visualization are being managed by the gateway on the largest machines on the TeraGrid (Ranger and Kraken). The ability for a user to deploy these jobs, visit a web site (the Gateway) to check on the status of the jobs, track where the data is stored, the archival status, and keep notes, documents, and figures associated with the simulation enhancing productivity.

3. ENABLING TECHNOLOGIES

3.1 Software

Several key technologies have come about over the past decade which makes the realization of a gateway interface possible. First and foremost the decision of the TeraGrid to provide a Globus GRAM environment for which applications can be launched outside of the typical batch submission process. The second important enabling technology is the proliferation of robust web application frameworks, i.e. Django that combine database technologies with web technologies that allow for the rapid development of a front-end to an existing command line driven application.

Specifically, Django combines three principle technologies that make web application development straightforward based on the concept of Models, Views, and Templates [11]. Firstly, Django's models provide an elegant interface to the underlying SQL database that is used to manage the entire simulation process. The database holds key information about the user and all simulations both past and present that have run using the Gateway. Key simulation data that is stored includes such things as source code revision, input file parameters, location of data sets, archival history of the data sets, visualizations generated, post-processing scripts, graphical and textual analysis and bookkeeping, etc. Secondly, Django's views define the processes and procedures that are called when different actions are taken when the user submits a job or operates on an existing or currently running job. These processes and procedures are embedded python code, shell scripts, and Globus RSL scripts that interact with both the TeraGrid resources and the machine hosting the Gateway. Thirdly, Django's templates define the look and feel of each web pages that comprises the web application. The combination of these three orthogonal components allows the gateway developer to write code for defining and accessing the data

(the model) that is separate from the workhorse scripts that interact with the TeraGrid and Gateway machine (the views), which in turn is distinct from the user interface (the templates). The sole purpose for choosing Django was to facilitate gateway design and development. Each individual piece could be tested in isolation and then integrated into the system. We did not have the luxury of bringing in a team of developers with expertise in database design, graphics, or human computer interactions, etc. but instead a single individual developed the Gateway relying on the rapid prototyping features of Django and its conceptual model for a web application. User feedback was instrumental in providing an interface for each view that was straightforward and intuitive. Each view could be evolved independently and the Django's templating system allowed for aesthetic design decisions to be incorporated as the system was developed. The ease of this development process was due primarily to the decoupled nature of Django. The end user is unaware that the web application was developed using Django.

Gateway development was simplified by implementing simple Globus RSL (Resource Specification Language) scripts [5] that could be run from the command line. The RSL scripts were only used on a very limited number of machines due to our limited access to other resources. We have not tried porting the RSL scripts to other platforms, but are starting to explore the feasibility of using these scripts on other non-TeraGrid resources. These scripts implemented job submission, building unique versions of the code, launching batch mode visualization, data archiving, etc. Once the scripts were thoroughly vetted, they were added to the Django web application in an iterative manner. Each separate view of the web application provided input to the Globus scripts and any output (other than the data sets generated from Uintah) were stored in the Django database.

The advent of powerful visualization software, i.e. VisIT [2], that runs both interactively and in batch modes on dedicated TeraGrid resources, i.e. Spur[3], has been essential in providing an integrated end user experience where a simulation is started, monitored, and the results stored and visualized in a seamless way all via the Gateway interface. The Gateway provides an interface for setting up some basic visualization tasks that our early adopters sought. There is a visualization view with a listing of the simulation variables and several common visualization tasks, i.e. contour plots, slices, etc. that can be selected. Once the user selects a task, a Python script is sent up to the remote visualization host via RSL and initiates the visualization task. This aspect of the Gateway is still in its early stages and is undergoing continuous development and refinements.

In addition to the software described above, general TeraGrid policies enhance the end user experience. First and foremost the community account offerings for TeraGrid allocations provide a mechanism for new users to try out Uintah without requiring dedicated TeraGrid accounts. This is a significant advantage for outreach education using Gateways. With community accounts, the responsibility for security and compliance of all TeraGrid resources was left to the gateway system. The trade-off for dealing with security at the gateway level was offset by the significant opportunities we have to expand our visibility and user base by offering TeraGrid access via our Gateway.

3.2 Hardware

The combination of several different platforms that allow smaller simulation runs on as few as tens of processors to thousands of processors on Ranger to the very large, data intensive, long running simulations requiring full machine capacity offered our user community unique opportunities to ramp up their simulation studies. We could offer our user community opportunities to get to know Uintah at the small scale using Ranger and as they developed a feel for the power of Uintah at the very large scale, Kraken could be used to do ground breaking simulation science with minimal overhead.

The primary computational resources included using Ranger (Sun Constellation Cluster with 3936 nodes and 62,976 cores) for smaller jobs, hundreds of cores, to several thousand cores. Kraken (Cray XT5 with 8,256 compute nodes and 99,072 cores) was used for very large core count runs – 99,0729 cores. Finally Spur and Ranger (Sun Visualization Cluster with 8 nodes and 128 cores) were used as a visualization resource to launch VisIT (parallel visualization system) [2] in batch mode during a running simulation as well as being used as a post processing tool for more comprehensive data exploration. Ranch (Sun Storage Tek mass Storage Facility with a capacity of 1 Petabye) archival system was used for storage. Ranch was just used as an archival resource with the data staying local for a period of time. Kraken data was not migrated to Ranch. We are still investigating how to improve the management for large scale Kraken runs.

4. UINTAH GATEWAY REQUIREMENTS

The Uintah Gateway was conceived to be a tool that would target two disparate sets of users: the experienced/advanced user and the novice. This disparate group presented a unique challenge to the gateway development. While trying to balance the competing and often contrasting requirements between these two extremes, we felt that it was important to prioritize the development process. During the conceptual stage of the Gateway, we sat down with both very experienced users and several novice users to better understand their needs. What evolved in each user community was a homogeneous set of requirements or must haves followed by individual requests for certain unique features. The universal requests were prioritized and emphasized, and the unique feature requests are being considered for future versions of the Gateway.

4.1 Large Scale Computing Requirements

From our advanced set of users, several common scenarios or wishes were repeated. The primary requirement was to manage the end to end computation or work flow for a simulation. This essentially included the ability to quickly create in an input file either from scratch or modify an existing input file, launch the simulation from the web interface, provide ongoing feedback to the state of the simulation via both email updates and visualization using the batch mode capabilities of VisIT, and finally provide some mechanism for managing the large volume of data that results.

Our advanced users estimated that they spend upwards to 70% of their time managing the runs and the remaining time performing the actual science. Management tasks include keeping track of code versions, input files, data management including moving data from scratch file systems to archival or bringing back portions of the data to local workstations for analysis and data reduction. Our advanced users still had issues with building and running Uintah in batch mode on the various TeraGrid resources due to differences in OS, compilers, libraries, batch submission scripts, etc. They were very receptive to using a system that would eliminate the burden of maintaining individual versions of executables.

To help our advanced users, we designed the Uintah Gateway database to store the following pieces of information for each simulation: user, date, problem description notes, TeraGrid resources, input files, code version including any code differences, compiler version, library versions (petsc, hypre, mpi), length of run, num-

[3]Spur is a Sun Visualization Cluster with 8 nodes and 128 cores housed at the University of Texas

ber of SUs, checkpoint/restart information,post-processing scripts, location of the data (is it archived, if so where, when was it last accessed), archived state, data movement (is it still in storage or was it moved to a non TeraGrid resource), post processing output including any data reduction as well as images generated during and after the run using VisIT and any other visualization tools. We used the Postgresql database is the back-end SQL database accessed via the Django API.

4.2 New User Requirements

The primary new user requirement is a simplified graphical user interface for creating input files for the Uintah executable, *sus*[4]. The input files are written in XML and embody all the parameters and options used to specify a complete simulation. A second requirement was an easier way to install the Uintah software. And finally, the last requested feature was a way to try out Uintah on large scale resources.

The main Uintah executable requires a conforming XML file that details all facets of the simulation including such things as

simulation component ICE, MPM, MPMICE, Arches, etc.
time maximum simulation time, time-step size, etc.
boundary conditions Variables, i.e. velocity, temperature, etc, and type, i.e. Neumann, Dirichlet, etc.,
data archiver variables stored and how often data is saved to disk
material properties material models used and any parameters required to characterize each model including the geometry specification.

Each material model requires a number of parameters to be specified Uintah includes a built in validator which ensures that for a given model all inputs have been specified. However, the range of values for each input is not verified and the user must ensure all inputs have reasonable values.

The current version of the Gateway does not include any kind of GUI input for problem specification. Instead the user must upload an input file to the Gateway server. Experienced and new users have requested a graphical user interface for input file creation that also helps ensure that reasonable values are specified. Future versions of the Gateway will incorporate a GUI input file creator.

In order to expand the Uintah user base, we plan on implementing training workshops for Uintah and the Gateway will provide an excellent resource for introducing new users to the system. In addition, we are encouraging educators to consider using the Uintah Gateway in the classroom as part of courses for numerical analysis, structural and fluid mechanics, and large scale computing.

5. GATEWAY USE CASES

5.1 Case Study: Large Scale Computing

Three of our Utah colleagues including one Co-PI with NSF PetaApps funding have an ongoing project using the TeraGrid to investigate micro-scale fluid structure interaction problems. Specifically, this work is the examination of the heat flux from array of micro-scale flexible pin fins (μFPF) placed in a cross flow. This configuration is interesting both scientifically and for its potential practical applications in CPU cooling. One proposed configuration for an array of pins is shown in Figure 1, in which the deformation of the fins is caused by the asymmetric oscillating vortices on the downstream side of the μFPFs at Reynolds numbers $Re > 47$, producing a time-varying pressure differential. When the frequency of vortex oscillations is near that of the μFPF's fundamental frequency, a harmonic flexure and contraction of the fin results.

[4]StandAlone Uintah Simulation, see section 9

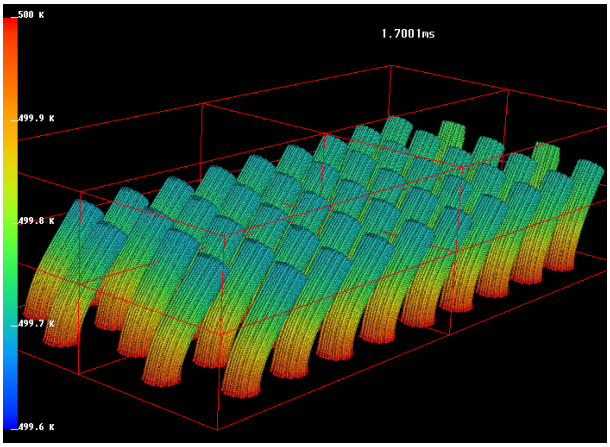

Figure 1: Micro flexible pin fins in a cross flow.

The fin motion increases local heat transfer due to enhanced mixing and the increased relative velocity of the flow over the solid surface. The actual arrangement of μFPFs will be optimized to maximize local heat transfer conditions for the expected bulk flow rates. The anticipated μFPF diameter will be on the order of $10 - 50\mu m$ with an aspect ratio AR will be approximately $5 - 16$ resulting in flows in the slip flow regime. Initial calculations indicate that the heat transfer enhancement (ratio of fin heat transfer rate for a rigid pin fin to the heat transfer rate without a fin) may be as high as 60 in the range $50 < Re < 500$ for copper pin fins with $5 \leq AR \leq 16$. For the same conditions, the μFPF oscillations could increase the enhancement by as much as a factor of 3, producing an overall enhancement for a single μFPF as high as 180.

Preliminary simulations have been conducted to verify the modifications made to the momentum and energy exchange models of the MPM-ICE (a fluid structure interaction algorithm-FSI) Uintah component [9], such that the equivalent of first-order slip velocity and temperature jump boundary conditions are achieved at fluid-solid boundaries, which may move and deform arbitrarily with time. The MPM-ICE algorithm is a three-dimensional, unsteady, continuum based Eulerian-Lagrangian methodology in which fluids, modeled using ICE (implicit, continuous fluid, Eulerian) and solid materials, modeled with MPM (the material-point-method), may be modeled either independently or simultaneously. ICE is a finite volume, cell-centered, multi-material, compressible, computational fluid dynamics (CFD) algorithm that originated at Los Alamos National Laboratory [4, 13]. The development and documentation of the MPM-ICE implementation currently used is given in [7–9, 12]. The MPM-ICE FSI algorithm utilizes a statistically averaged, or âĂŸmulti-field,âĂŹ approach, where, each material is continuously defined $(\rho, \mathbf{u}, e, T, v, \theta, \sigma, P)$, with some probability, over the entire computational domain. This approach differs from the, perhaps more common, separate domain methodology, in which, fluid and solid materials are defined separately, with only one material at each point, and interaction only occurring at material boundaries. The multi-field approach is advantageous for the current application, because it tightly couples fluid-structure-interactions through the conservation equations, rather than explicitly though specified boundary conditions, which allows arbitrary distortion of material and material surfaces without explicit surface tracking, passing of boundary conditions, and excessive stability and convergence issues. Use of the MPM-ICE algorithm to evaluate rarefaction with FSI is further merited, as rarefaction effects

have already been successfully studied utilizing the independent CFD (ICE) portion of the algorithm, with slip boundary conditions implemented at the computational domain boundaries [19–21].

The multi-material governing conservation equations used by the MPM-ICE algorithm, without effects that are not considered in the present research (chemical reactions, turbulence, multiphase Reynolds stress, gravity, etc.), are given in Equations 1-âĂŞ4 [8].

$$\frac{\partial \rho_r}{\partial t} + \nabla \cdot (\rho \mathbf{u})_\mathbf{r} = \mathbf{0} \quad (1)$$

$$\frac{\partial (\rho \mathbf{u})_\mathbf{r}}{\partial t} + \nabla \cdot (\rho \mathbf{u}\mathbf{u})_\mathbf{r} = -\theta_\mathbf{r} \nabla \mathbf{P} + \nabla \cdot (\theta \tau)_\mathbf{r} + \sum_{\mathbf{s=1}}^{\mathbf{N}} \mathbf{f_{rs}} \quad (2)$$

$$\frac{(\rho e)_r}{\partial t} + \nabla \cdot (\rho e \mathbf{u})_\mathbf{r} = -\frac{\mathbf{P}\theta_\mathbf{r}}{\mathbf{v_r}} \frac{\mathbf{dv_r}}{\mathbf{dt}} + (\theta \tau)_\mathbf{r} \quad (3)$$

$$\nabla \mathbf{u_r} + \nabla \cdot (\theta \mathbf{k} \nabla \mathbf{T})_\mathbf{r} + \sum_{\mathbf{s=1}}^{\mathbf{N}} \mathbf{q_{rs}} \quad (4)$$

Equations 1–4 are the ensemble average, r material, conservation of mass, momentum, and energy equations respectively, where there are N materials, θ_r is the râĂŚmaterial volume fraction, and and are models for the momentum and energy exchange between materials. Equations 1–4, along with individual material constitutive models, and equations of states models form a complete system of equations. The detailed numerical solution strategy utilized by the MPM-ICE algorithm to solve this system of equations is presented in [8]. Briefly, the numerical approach involves operator splitting. For each time-step, the quantities on the right-hand-side of Equations 1–4 are computed first - this is the Lagrangian phase of the time-step. The conserved quantities, that is, mass, momentum, and energy, for fluid materials are accounted for at the cell centers; while, the conserved quantities for solid materials are accounted for at the material particles. Consequently, during the Lagrangian phase, which is executed primarily within the cell-centered ICE framework, the solid materials are dually represented, both, at the particles, and at the cell centers, where the solid material conservation quantities are interpolated. In the second phase of the time-step, the Eulerian phase, the contribution due to advection, that is, the second term on the left of Equations 1–4, is added to the Lagrangian phase values, where the advected contributions are computed for fluid materials by ICE, and for solid materials by MPM. As such, during the Lagrangian phase, models for both the momentum and energy exchange between materials, are used, while during the Eulerian phase, only the momentum exchange model is used to determine the advecting velocity.

To achieve first-order slip velocity and temperature jump boundary conditions are achieved at fluid-solid surfaces for a rarefied gas in the slip flow regime, the momentum and energy exchange coefficients, which result in tangential slip velocity and temperature jump values that correspond to values predicted by the standard first-order slip boundary conditions [16, 22], are derived as a function of the level of rarefaction. Then, because the slip flow momentum exchange coefficient is only applicable in the fluid-solid surface tangential direction, while a no-slip momentum exchange coefficient must still be applied in the fluid-solid surface normal direction, the momentum exchange coefficient can no longer be treated as a scalar quantity. The momentum exchange between materials must be calculated in fluid-solid surface normal and tangential coordinate directions, rather than the arbitrary global coordinate directions. Following the development and implementation of the slip flow momentum and energy exchange models, several basic configurations were considered and compared to established data to verify the resulting algorithmâĂŹs capabilities. These verifications include:

1. velocity profiles of a rarefied gas between parallel plates
2. temperature profiles of a rarefied gas between parallel plates
3. drag coefficients, C_D, and Nusselt numbers, Nu, for low Reynolds number rarefied flow around an infinite cylinder
4. the transient, thermal/structural response of a damped-oscillatory three-dimensional finite cylinder subject to an impulsively started uniform, rarefied flow

The preliminary simulations are used as a verification procedure to quantify and understand the modifications made to the momentum exchange and energy exchange models necessary to understand more realistic μPF configurations. The realistic geometry configurations that will be used to study the flow (pressure drop) and heat transfer characteristics include:

1. a single fin as a function of flow velocity, geometric, and pin material parameters
2. two μFPFs aligned in the stream-wise direction as a function of the same parameters and fin spacing
3. a μFPF array subject to different μFPF arrangements.

For each configuration, the drag coefficient (dimensionless quantity that is used to quantify the resistance of the μFPF to the air flow), and the Nusselt number (ratio of convective to conductive heat transfer across the μFPF surfaces within the air)) are computed [20, 21, 23]. These simulations provide valuable data concerning the applicability of the novel μFPF heat exchanger surface that would be difficult to obtain otherwise, either analytically, experimentally, or by a commercial CFD code.

5.1.1 Preliminary Results

Initial simulations looked at three different configurations to assess the modifications to momentum and energy exchange to represent slip flow velocity and slip flow temperature boundary conditions. The first configuration was ideal gas flow between parallel plates and steady state thermal conduction between two parallel plates. Comparison to analytic solutions indicates the modified algorithm solutions converge to the same order of accuracy and exchanged momentum and energy quantities are conserved. Velocity and temperature profile comparisons between numerical and analytic solutions show good agreement [23].

Flow around an infinite circular cylinder was evaluated with the flow behavior, drag coefficient, C_D and Nusselt number, Nu both numerically and analytical. The numerical evaluation of the infinite cylinder required a large computational domain along with AMR. Good agreement for both C_D and Nu were obtained for various Reynolds numbers and reported in [23].

The third configuration included the steady state thermal and hydrodynamic interaction of a rarefied gas with a stationary solid (pin fin). The initial displacement of the solid was zero, and as the impulsively started uniform rarefied flow was initiated causing the displacement and the damped oscillations while simultaneously transferring heat to the fluid. The Euler-Bernoulli equation for beam vibration with a Stokes drag force were used to compute the analytic results. Good numerical agreement was obtained with the analytic predicted displacement and temperature solutions [23].

This particular study is an attractive case study for looking at the impact of the Gateway on scientific productivity. For each par-

ticular configuration, two numbers are computed, the drag coefficient and the Nusselt number. The data is generated on the TeraGrid, post processing scripts are used to compute the two numbers. Within this type of scenario the data management and post processing can be automated. In addition, the various configurations can be automatically set up and launched from within the Gateway and managed via the web interface. Small time-steps are required in the explicit algorithm due to large domain areas and small cells requiring 100's of thousand time-steps and multiple restarts to achieve a steady state solution. Automating the post processing steps, restart configurations and data management presents opportunities for increasing scientific productivity.

5.2 Case Study: New User

Specifically we looked at the following common scenario, no experience using Uintah on the TeraGrid. The metric we used is the total time required to get a four processor MPM example (using an input file from our distribution tarball) running on the TeraGrid. Using an existing input file on a small processor count permitted for the simulation to be executed in near real time. The first scenario required the user to download and install Uintah on Ranger using our extensive documentation and wiki instructions (see `www.uintah.utah.edu`) and then writing a simple batch script and submitting the job to the batch queuing system. This was compared to the user logging onto the Gateway for the very first and launching a small simulation to the TeraGrid. A comparison summary of these two different use cases is described in Table 1.

Table 1: User Submission Times (minutes)

User Type	Install Time	Batch Sub-mission	Gateway Login	Gateway Sub-mission
Non-Gateway	120	20	N/A	N/A
Gateway	N/A	N/A	5	10

We had several new graduate students with no prior experience using Uintah on the TeraGrid and compared their ability to initiate jobs, manage the queuing system and process the simulation data versus graduate students that used the Gateway to manage the job submission and data processing. We found that for simple scenarios, the Gateway users were able to get simulation results in approximately half the time versus those new users that managed the entire process manually. In addition, we found that the gateway users were able to spend more time doing actual science rather than managing the simulation runs.

6. USING THE UINTAH GATEWAY

The Uintah Gateway consists of a web interface with multiple views depending on what features the user is performing. The following main views are available to the user:

1. Initial login screen
2. Simulation history
3. Simulation submission screen
4. Detailed simulation view
5. Simulation status

The user logs into the Gateway via the initial login screen available from the main Uintah web page (`www.uintah.utah.edu/gateway.html`). At this stage, the user is presented with a list of simulations (**Simulation History**) that have been run or have been submitted via the Gateway interface. If the simulation history is empty, the user is automatically taken to the simulation submission view. (Note: only simulations submitted through the Gateway are stored in the database and available via the web interface.) However, if this is a returning user, each simulation submitted via the Gateway is listed based on the date submitted and a short descriptive title along with the simulation status, i.e. completed, pending, or in progress. Beside each simulation is the option to select one of the listed simulations for a detailed view. In addition to the list of simulations, there is also a menu list showing the available main views including the **Simulation Submission** option.

The process of submitting a simulation begins with **Simulation Submission** view. Within this view are forms for specifying the simulation title (used in the simulation history view), a form for uploading the input file from the user's home file-system. The user must create a conformant input file outside of the Gateway. (Future Gateway enhancements will include a graphical user interface for input file creation.) The uploaded input will be validated on the Gateway machine prior to submission to the TeraGrid. Also within this view are boxes for specifying the number of processors to use and time requested and the machine to use (at this point, Ranger is our primary TeraGrid resource for Gateway runs). There is no automatic mechanism for estimating the amount of time required for a given run. If the user underestimates the amount of time, the restart capabilities of Uintah can be used to continue with the simulation until the simulation is completed. At the bottom of the view is a Submit button that must be pressed before any action takes place on the Gateway machine. Once the user submits the input file, the input file is validated using the validate feature of *sus*. The validation procedure ensures that the input file is correctly specified, i.e. no missing input tags. If the input file fails to validate, the user immediately is notified of the problem. At this point, the simulation is no longer valid and is removed from the submission history for the user. If the input validates, then Globus RSL scripts are automatically created and submitted to the TeraGrid machine via the Gateway server using a combination of python and shell scripts.

Once the user submits the simulation, the main **Simulation History** view is displayed showing the just completed simulation at the top of the page with status identifier *Pending* next to the simulation title. This identifier signifies that the job has been submitted but is still waiting to be run. At this point, the user may log off from the Gateway and wait for email notification from the Gateway that the user's job has started and when it has been completed.

When the user has received email notification, they may log back into the Gateway returning to the **Simulation History** view and noting the status change from *Pending* to *Completed*. At this point, the user may select the simulation's **Detailed Simulation View** and is presented with a view similar to Figure 2. This view presents information the user added during the submission process, such as input file, date, number of processors, data-set location, standard output from the simulation, etc. In addition, to this information, options are available to upload images and text notes for storage into the database. This centralized database encourages users to archive all relevant post processing images and notes conducted as part of the analysis phase of the simulation data.

For long running simulations, there is a **Simulation Status** view that presents an updated state of the simulation including time remaining and standard error and standard output generated from the *sus* executable. Querying a running simulation allows the user to catch any problems or to download selected time-steps for further

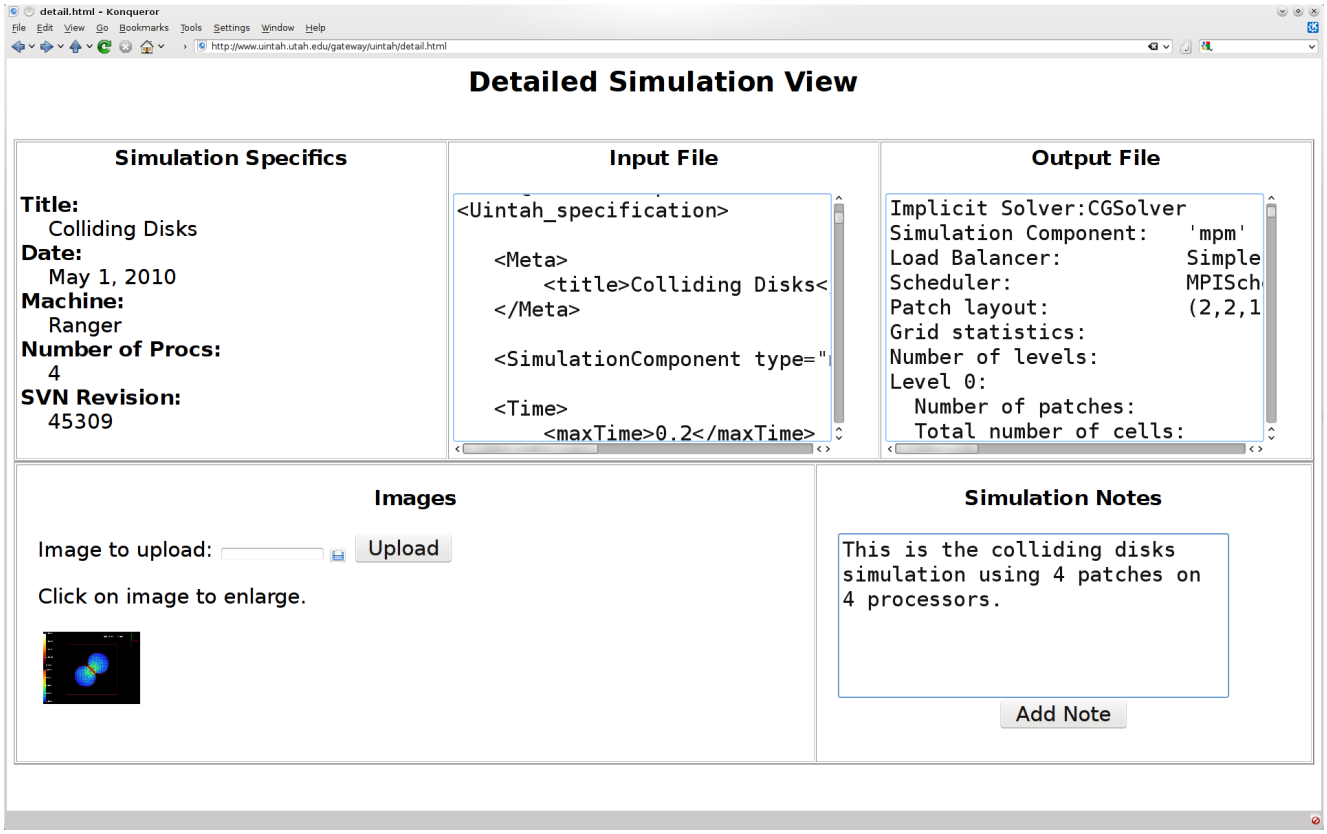

Figure 2: Detailed simulation view

analysis. Uintah has automatic checkpointing that writes out the complete state of the simulation at intervals specified by the input file. Typically two checkpointed datasets are always written out and overwritten with latter versions as the simulation advances in time. The user has the option of restarting from either the latest checkpointed data or from an earlier version. Hardware or software failures that would cause a job to abort can be restarted from the checkpointed data. At most the only simulation data that is lost is from the time of the last checkpointed data output to the time-step that occurs during the time of failure. However, a restart from the last checkpoint can recreate this lost data in a subsequent run. Email notification is sent to the user indicating that the job aborted prematurely. The user can then log back into the Gateway to restart the aborted job. This is primarily useful for those instance where hardware failure causes a job to abort. For instances of software failure within Uintah, restarting a job usually causes the job to fail/abort in the same time-step indicating an underlying problem in the algorithm, models, or assumptions in the simulation itself. There is no automatic mechanism built into the Gateway to restart aborted jobs, instead the responsibility is left to the user via the Gateway interface.

Within the **Data Management** view are options to display the location of the simulation's data-set, where and when it was archived, and an option to copy the data-set to another machine. There is also the option to copy selected time-steps.

7. SUMMARY AND FUTURE DEVELOPMENT

The Uintah Gateway offers new and experienced users a way to easily use the largest machines in the country to solve complicated fluid structure interaction problems. The Uintah Gateway was developed using the Django web application infrastructure which allowed for rapid and iterative development. Two use cases were presented to show the gains made using the Uintah Gateway for simulating complex fluid structure interaction problems on the TeraGrid. The Uintah Gateway's primary views were described including the simulation history, management interface, detailed simulation view with an overview of how the user would use the Gateway.

Preliminary results suggest dramatic improvements in scientific productivity for both new users and experienced users for very large scale problems. For all user demographics, we observed significant reduction in time to solution as much as 50%. The increase in productivity allowed our new users to rapidly explore and launch more sophisticated simulations in a fraction of the time it took a non-gateway user to accomplish the same task. For our experienced users, we found that they spent more time doing science than they did managing the simulations. Fewer errors were made by the scientist, since much of the record keeping and automation was performed by the gateway system.

Future development of the Gateway will include a GUI interface for problem setup that automatically generates a conforming input file. The ongoing development of data management features addressing the needs of our advanced user community will also be a focus of our future work. The Uintah Gateway is an ongoing project that seeks to bring the most powerful supercomputers available to solve important complex multi-physics problems to a broad set of users in multiple disciplines.

8. ACKNOWLEDGMENTS

This work was supported by the National Science Foundation under subcontract No. OCI0721659 and the University of Utah's Center for the Simulation of Accidental Fires and Explosions (C-SAFE) by the the Department of Energy under subcontract No. B524196. We would like to thank Jennifer Van Rij, Todd Harman and Tim Ameel funded under National Science Foundation award CBET-0933574. Jennifer was our our initial large scale gateway user. We would like to thank TACC and NICS for computer access along with the C-SAFE team. We would like to thank Matthew Woitaszek from UCAR for many helpful discussions and insights into gateway development.

9. APPENDIX – UINTAH SOFTWARE REPOSITORY, GATEWAY LOCATION AND USAGE INFORMATION

The Uintah project (see `http://wwww.uintah.utah.edu`) and the C-SAFE projects (see `http://www.csafe.utah.edu`) are extensively documented. From the homepage of the Uintah project are links to the source code, installation instructions, usage information, example input files. In addition, the main Gateway link (`http://www.uintah.utah.edu/gateway.html`) allows new users to get started using the Uintah Gateway.

The main Uintah executable is called `sus` which stands for *StandAlone Uintah Simulation*. Input files describing the main algorithmic components, boundary conditions, grid layout, etc. are laid out using XML and passed to the `sus` executable from the command line.

10. REFERENCES

[1] A. Brydon, S. Bardenhagen, E. Miller, and G. Seidler. Simulation of the densification of real open-celled foam microstructures. *Journal of Mechanics Physics and Solids*, 53(2638-2660), 2005.

[2] H. Childs, E. Brugger, K. Bonnell, J. Meredith, M. Miller, B. Whitlock, and N. Max. A contract-based system for large data visualization. In *Proceedings of IEEE Visualization 2005*, pages 190–198, 2005.

[3] J. de St. Germain, S. Parker, J. McCorquodale, and C. Johnson. Uintah: A massively parallel problem solving environment. In *HPDC*, pages 33–42, 2000.

[4] F.Harlow and A. Amsden. Numerical calculation of almost incompressible flow. *Journal of Computational Physics*, 3:80–93, 1968.

[5] I. Foster and C. Kesselman. Globus: A metacomputing infrastructure toolkit. *International Journal of Supercomputer Applications*, 11(2):115–128, 1997.

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

[7] J. E. Guilkey, T. Harman, A. Xia, B. Kashiwa, and P. McMurty. An eulerian-lagrangiana approach for large deformation fluid structure interaction problems, part 1: Algorithm development. In S. Chakrabarti, editor, *Proceedings of the Second Internation Conference on Fluid Structure Interaction*, pages 143–156. WIT Press, Boston, 2003.

[8] J. E. Guilkey, T. B. Harman, and B. Banerjee. An eulerian-lagrantian approach for simulating explosions of energetic deivces. *Computational Structures*, pages 660–674, 2007.

[9] T. Harman, J. E. Guilkey, B. Kashiwa, J. Schmidt, and P. McMurty. An eulerian-lagrangiana approach for large deformation fluid structure interaction problems, part 2: Multi-physics simulations within a modern computational framework. In S. Chakrabarti, editor, *Proceedings of the Second Internation Conference on Fluid Structure Interaction*, pages 157–166. WIT Press, Boston, 2003.

[10] T. Henderson, P. McMurtry, P. Smith, G. Voth, C. Wight, and D. Pershing. Simulating accidental fires and explosions. *Comp. Sci. Eng.*, 2:64–76, 1994.

[11] A. Holovaty and J. Kaplan-Moss. *The Definitive Guide to Django: Web development Done Right*. Apress Publishing, 2007.

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

[13] B. A. Kashiwa, N. T. Padial, R. Rauenzahn, and W. VanderHeyden. A cell-centered ice method for multiphase flow simulations. Technical Report LA-UR-93-3922, Los Alamos National Laboratory, 1993.

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

[15] J. Luitjens and M. Berzins. Improving the performance of uintah: A large-scale adaptive meshing computational framework. In *IPDPS '10: Proceedings of the 2010 IEEE International Symposium on Parallel&Distributed Processing*, Washington, DC, USA, 2010. IEEE Computer Society.

[16] J. Maxwell. On stresses in rarified gases arising from inequalities of temperature. *Philosophical Transactions Royal Society of London*, 170:231–256, 1879.

[17] Q. Meng, J. Luitjens, and M. Berzins. A comparison of load balancing algorithms for amr in uintah. SCI Technical Report UUSCI-2008-006, University of Utah, 2008.

[18] S. Parker. A component-based architecture for parallel multi-physics pde simulation. *Future Gener. Comput. Syst.*, 22(1):204–216, 2006.

[19] J. V. Rij, T. Ameel, and T. Harman. The effect of creep flow on two-dimensional isoflux microchannels. *International Journal of Thermal Sciences*, 46:1095–1103, 2007.

[20] J. V. Rij, T. Ameel, and T. Harman. The effect of viscous dissipation and rarefaction on rectangular microchannel convective heat transfer. *International Journal of Thermal Sciences*, 48:271–281, 2009.

[21] J. V. Rij, T. Ameel, and T. Harman. An evaluation of secondary effects on microchannel frictional and convective heat transfer characteristics. *International Journal of Heat and Mass Transfer*, 52:2792–2801, 2009.

[22] M. Smoluchowski. Ueber wärmeleitung in verdünnten gasen. *Annals of Physical Chemistry*, 64:101–130, 1898.

[23] J. van Rij, T. Harman, and T. Ameel. Model development for fluid structure interaction in the slip regime. In ASME, editor, *Proceedings of ASME 2010 3rd Joint US-European Fluids Engineering Summer Meeting and 8th International Conference on Nanochannels, Microchannel, and Minchannels*, Montreal, Canada, August 2010. ASME.