Uintah Hybrid Task-based Parallelism Algorithm

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

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

***Abstract*—Uintah is a software framework that provides an environment for solving large-scale science and engineering problems involving the solution of partial differential equations. Uintah uses a combination of fluid-flow solvers and particle-based methods for solids, together with adaptive meshing and asynchronous task-based approach with automated load balancing. When applying Uintah to fluid-structure interaction problems, the combination of adaptive meshing and the movement of structures through space present a formidable challenge in terms of achieving scalability on large-scale parallel computers. Adopting a model that uses MPI to communicate between nodes and a shared memory model on-node is one approach to achieve scalability on large-scale systems.This scalability challenge is addressed here for Uintah, by the development of new hybrid runtime and scheduling algorithms combined with novel lock-free data structures, making it possible for Uintah to achieve excellent scalability for a challenging fluid-structure problem with mesh refinement on as many as 256K cores.**

I. Introduction

A significant part of the challenge in moving from petascale to exascale problems is to ensure that multi-physics multi-scale codes that reflect real applications can be run in a scalable way on parallel computers with large core counts. The multi-scale challenge involved arises from the need to use approaches such as adaptive mesh refinement while the multi-physics challenge is typified by different physics being used in different parts of the domain with different computational loads in these different spatial domains. In addition it is necessary to couple these domains with the different physics and this coupling is an interesting challenge in itself as the coupling algorithm may be more complex than the algorithms used away from the interface. These challenges are further compounded by the anticipated relative memory per core potentially shrinking [1]. Furthermore the node architectures are becoming more complex with ever-increasing core counts and with the potential for the use of accelerators as part of the node [1], on machines under construction such as Titan [1] and Stampede. [2] Adopting a model that uses MPI to communicate between nodes and a shared memory model on-node is one approach to achieve scalability at full machine capacity on large-scale systems. For this approach to be successful, it is necessary to design data-structures that large numbers of cores can simultaneously access without contention. These data structures and algorithms must also be designed to avoid the overhead involved with locks and other synchronization primitives when running on large number of cores per node, as contention for acquiring locks quickly becomes untenable.

[1]Titan is a parallel computer under construction at Oak Ridge National Laboratory with about 299K CPU cores now with a large number of attached GPUs to be added in 2012.

[2]Stampede is a parallel computer under construction at Texas Advanced Computing Center with 2 petaflops of CPU performance and 8 petaflops of accelerator performance.

II. Uintah Runtime system

The simulation grid in Uintah is partitioned into patches by a highly scalable regridder and assigned to nodes by a measurement-based load-balancer [2]. In each MPI process, the Uintah runtime system will schedule the tasks on local patches by using a local task graph and data warehouse. The task graph is a directed acyclic graph (DAG) [3] which is compiled by making connections on task's required and computed variables. The Uintah scheduler uses the task graph to determine the order of execution, assign tasks to local computing resources and ensures that the correct inter-process communication is performed. The data warehouse is a directory based datastructure, which manages all of the Uintah variables. A task can use variable name and a patch id key to load and save variables into the data warehouse. The data warehouse also manages MPI message buffers and automatically garbage collects variables when no longer needed.

A new multi-threaded MPI scheduler [4] was designed to eliminate redundant MPI messages and memory copy by creating multiple worker threads on the same multi-core node. In this way, tasks running on different cores can directly access all variables on the same node. This multi-threaded MPI scheduler had one control thread and several worker threads per MPI process. The control thread processes MPI receives, manages tasks queues and assigns ready tasks to worker threads. The worker thread simply executes the task that control thread assigned to it. As Uintah variables can be accessed freely by any thread, many shared data structures, such as data warehouse and task queues were redesigned to guarantee thread-safety.

III. Improvement and Results

A potential bottleneck in the above model is that a worker thread may become idle if the control thread cannot respond to its next ready task request quickly enough. In order to response quickly, the control thread was assigned to a dedicated core.

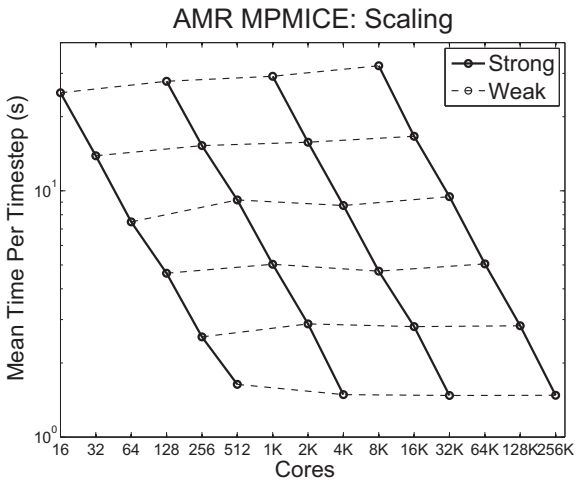

Fig. 1. AMR MPMICE hybrid approach salability

However, this led to the control thread core being under-utilized. The solution adopted in this research was designing a new de-centralized multi-threaded scheduler to allow all threads to process MPI sends and receives and/or execute tasks freely and concurrently without using a control thread [6]. Instead of asking the control thread for a ready task, the threads in the de-centralized multi-threaded scheduler are equal and directly pull tasks from the two ready queues. The decentralized scheduler is able to fully use all available cores on-node, regardless of the number of cores and outperforms the previous model.

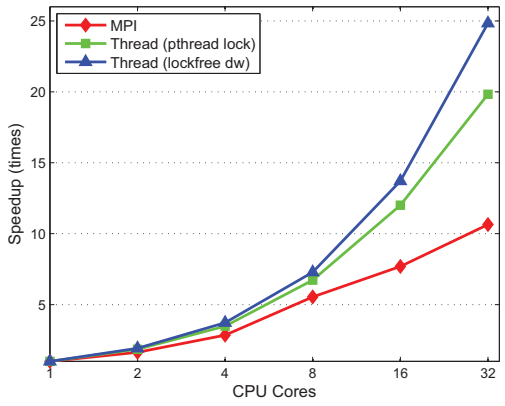

Fig. 2. Hybrid scheduler speedup

In Uintah, solid objects are represented by particle variables. During a timestep, each particle's new location co-ordinates and other physical attributes are computed. If a particle's position moves from one patch to another, it is the infrastructure's responsibility to map those particle variables back into the background grid according to their new locations. The cost of moving a particle off a node is much more expensive than that of moving a particle inside a node or core. When using a hybrid MPI multi-threaded approach, the number of particles that need to be sent is significantly reduced when using one MPI process per node as opposed to one MPI process per core, as all transfers internal to a node no longer use MPI.

By using the hybrid scheduling approach, all tasks in the same node can be executed by any idle core on that node. Moreover, the load balancer now profiles and predicts workload per node instead of per core. The accuracy of prediction is improved as changes in workload over a larger region are generally more stable than those over a small region. The average core load imbalance value was reduced from about 60% to 25% when running with nearly one patch per core at 100K cores on the Jaguar XK6 by using this approach.

The multi-threaded scheduler originally used locking to protect shared data structures. This overhead keeps increasing with the number of cores per node as contention for acquiring locks also increases. Based on our timing results on Uintah read-write locks, the data warehouse lock was the largest single source of overhead. Novel lock-free data structures and algorithms using atomic instructions were designed in this research to replace the use of high-level heavy-weight pthread locks on frequently accessed data structures [6]. Figure 1 shows Uintah achieved excellent scalability for a challenging fluid-structure problem with mesh refinement on 256K cores with 95% weak scaling efficiency and 68% strong scaling efficiency . Figure 2 shows these simulations have 2.4X speed up comparing to the MPI only scheduler. Finally a de-centralized unified MPI/Threads/GPU scheduler is currently under development [5].

ACKNOWLEDGMENT

This work was supported by the National Science Foundation under subcontracts No. OCI0721659, the NSF OCI PetaApps program, through award OCI0905068 and by DOE INCITE award CMB015 for time on Jaguar. 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.

REFERENCES

[1] Scientific Grand Challenges: Crosscutting Technologies for Computing at the Exascale Report from the Workshop Held February 2-4, 2010 http://science.energy.gov/~/media/ascr/pdf/program-documents/docs/Crosscutting_grand_challenges.pdf

[2] M. Berzins, J. Luitjens, Q.Meng, T.Harman, C.A.Wight and J. Petersonr. Uintah a scalable framework for hazard analysis *Proc. of 2010 Teragrid Conf.* July 2010, ACM.

[3] M. Berzins, Q.Meng, J.Schmidt and J. Sutherland. DAG-Based Software Frameworks for PDEs *Proc. of HPSS 2011 (Europar Bordeaux August 2011)*, Springer 2012.

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

[5] A. Humphrey, Q.Meng, M.Berzins and Todd Harman Radiation Modeling Using the Uintah Heterogeneous CPU/GPU Runtime System. *In Proceedings of the 2012 Xsede Conference: Extreme Digital Discovery*

[6] Qingyu Meng and Martin Berzins. Scalable Large-scale Fluid-structure Interaction Solvers in the Uintah Framework via Hybrid Task-based Parallelism Algorithms. *Submitted to Concurrency and Computation: Practice and Experience*. John Wiley & Sons, Ltd, July 2012.