The Center for Plasma Edge Simulation Workflow Requirements

Scott A. Klasky[1] Klasky@ornl.gov
Bertram Ludaescher[2] Ludaesch@ucdavis.edu
Manish Parashar[3] Parashar@caip.rutgers.edu

Abstract

The Center for Plasma Edge Simulation (CPES) is a recently funded prototype Fusion Simulation Project, which is part of the DOE SciDAC program. Our center is developing a novel integrated predictive plasma edge simulation framework, which is applicable to existing magnetic fusion facilities (D3D, NSTX, CMOD) and next generation burning plasma experiments, e.g. ITER. The success of this project will be in developing and understanding new models for the plasma edge in a kinetic regime with complex geometry.

Because of the multi-scale nature of the problem, we will study the neoclassical physics time scale kinetically, and the fast and larger scale MHD modes via a fluid code. Our approach is to couple these codes via a scientific workflow system, Kepler-HPC. Kepler-HPC will enhance Kepler with capabilities such as code coupling and data redistribution, high volume data transfers and interactive (and autonomic) monitoring, steering and debugging, which will be necessary for scientific progress in this project.

1. Introduction

We are starting to develop new integrated predictive plasma edge simulation code package, applicable for the plasma edge region relevant to both existing magnetic fusion facilities and next-generation burning plasma experiments, such as the International Thermonuclear Experimental Reactor (ITER). Timely progress in this formidable scientific challenge demands a well-coordinated effort involving experts in the plasma science, computer science, and applied mathematics areas – a research approach at the heart of the SciDAC Program [SciDAC].

The plasma edge includes the region from the top of the pedestal to the scrape-off layer and divertor region bounded by a material wall, see Figure 1. A multitude of non-equilibrium physical processes on different spatio-temporal scales present in the edge region demands a *large scale integrated simulation*. The low collisionality of the pedestal plasma, magnetic X-point geometry, spatially sensitive velocity-hole boundary, non-Maxwellian nature of the particle distribution function, and particle source from neutrals, combine to require the development of a special kinetic transport code, XGC-NT, for kinetic transport physics, using a particle-in-cell (PIC) approach on a massively parallel computing platform. For the study of large scale MHD phenomena, we will use the M3D code, which is a code used in the CEMM SciDAC project [M3D]. However, the kinetic and MHD codes must be integrated together for a self-consistent simulation as a whole.

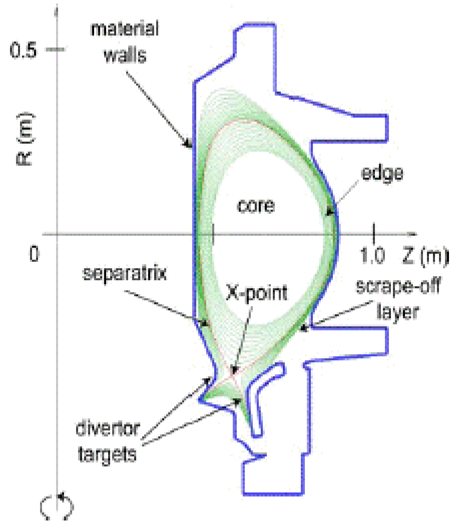

Figure 1. The plasma edge

The management of scientific data and information in a fusion simulation project will be an essential factor for this project. We are developing Kepler-HPC, a set of specialized HPC extensions for the Kepler scientific workflow system [LAB+06], and use it to couple the kinetic code to the fluid code, and to manage the data between these codes, along with the monitoring system which we are building. The workflow system includes provisions of services for exploring, analyzing and visualizing data and extracting information and features.

Achieving efficient, flexible and scalable coupling of physics models and parallel application codes investigated in this project presents significant algorithmic, numerical and computational challenges. From the computational point of view, the coupled simulations, each typically running on a distinct parallel system or set of processors with independent (and possibly dynamic) distributions, need to periodically exchange information

Specifically, in this project, the kinetic code, XGC-NT, runs efficiently on thousands of processors [XGC-1]. This code will be weakly coupled to an MHD code, M3D, which for the type of problems investigated in this project, will run on 64-128 processors. Because of the nature of the weak coupling, the workflow system must support hybrid execution combining coarse grain parallelism (running M3D and XGC-NT simultaneously) and fine grain parallelism (running each code on MPP's). When these codes are coupled, they will pass several 2D-3D variables from one code to another. The information exchange will take place about every 600 seconds, but the exchange should take less than one second. This constraint is due to the fact that the codes running on the separate platforms must be "in-synch" with one another.

The kinetic code in this project will run on 2048+ processors on the Cray XT3 at ORNL. This system is connected to a 40Gbit link to a 160 processor PIV infiniband cluster, as shown in Figure 4. A challenging issue with using this platform is that the XT3 computational nodes do not allow users to run sockets or threads. This constraint can be addressed by first sending data from the computational nodes to the I/O nodes using PORTALS [PORTALS]. The I/O nodes can then communicate to the infiniband cluster.

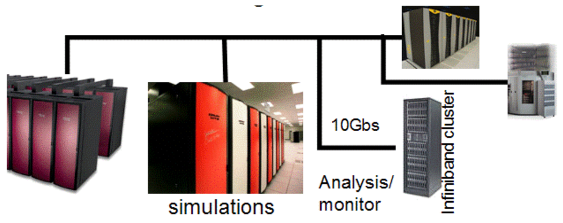

Figure 2. Network connectivity between the ORNL Cray's and our workflow cluster.

Given these constraints, typically used file-based data movement is likely not a feasible solution and memory to memory data movement must be investigated. Key requirements include: (1) interaction/communication schedules between individual processors executing each of the coupled simulation codes need to be computed efficiently, locally, and on-the-fly, without requiring synchronizations or gathering global information, and without incurring significant overheads on the simulations themselves; and (2) data transfers should also be efficient and should happen directly between the individual processors of each simulations via the intermediate IO nodes. Furthermore, specifying these coupling behaviors between the simulations codes using popular message-passing abstractions can be cumbersome and often inefficient, as these systems require matching sends and receives to be explicitly defined for each interaction. As the individual simulations become larger, more dynamic and heterogeneous and their couplings more complex, implementations using message passing abstractions can quickly become unmanageable. Clearly, realizing coupled simulations requires an efficient, flexible and scalable coupling framework and simple high-level programming abstractions.

The primary goal of our effort is to simplify and automate the scientific investigation processes for large scale parallel codes. The core research issues addressed include (1) dynamic coupling of constituent models and simulation codes, (2) adaptive and automated application workflows, (3) efficient and transparent access to and transport of distributed data, (4) data analysis and visualization, and (5) runtime monitoring and interactive and autonomic control. This research leverages technologies developed at the Fusion SciDAC's, PPPL, the Center for Advanced Information Processing (CAIP), and the Scientific Data Management (SDM) Center [SDM] to develop and deploy a framework for adaptive and automated workflows and integrated data analysis and visualization. In this paper we describe some aspects of the underlying scientific workflow system that will be used in this project and provide an overview of the code coupling, data redistribution and data transfer components that are being developed in this project.

2. The Kepler-HPC Workflow System

Kepler [ABB+05, Kep05, LAB+06] is a scientific workflow system extending the underlying Ptolemy II system [BLL+04] for heterogeneous modeling and design with a number of extensions that are necessary in many scientific applications. Extensions often take the form of components called *actors* that are independent of each other and typically execute as independent threads or processes. Kepler actors support access to and transport of distributed data via SRB (Storage Resource Broker), SRM (Storage Resource Manager), gridFTP, Sabul, BCC, and local and remote execution of legacy applications (e.g. via a command-line/shell actor or via web services). Moreover Kepler features a novel *hybrid type* system [BL05] that combines structural and semantic type information to facilitate scientific workflow design via controlled vocabularies and community ontologies: whenever datasets and computational components have corresponding rich metadata, Kepler can exploit this information during design and static analysis of a workflow.

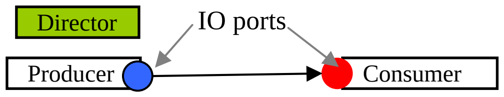

Figure 3. Actor communication in Kepler-HPC.

Kepler-HPC is a set of specialized extensions to Kepler to enable high-performance computing workflows, in particular, for interactive and autonomic monitoring, steering and debugging, of remote HPC applications. Like other scientific workflow systems, Kepler workflows naturally support task parallelism (parallel branches are executed concurrently), and data parallelism (provided the individual jobs or codes "inside" of actors exploit data parallelism). Unlike most other systems, Kepler also comes with "built-in" support for data streaming, due to the underlying workflow execution model for dataflow process networks [LP95]. This feature makes Kepler well-suited for high-end applications where file-based data transport might not be feasible. Actors are independent from one another, so that a data-consuming actor is decoupled from a data-producing one (cf. Figure 2). The receiver is placed and controlled by an overall system component called "director" which can make use of specialized low-level communication channels such as sockets. Another feature of the Kepler-HPC architecture is that actors can be used to set-up, monitor, and steer remote HPC jobs. Below, we show part of the workflow which we have been working on. The major codes (XGC-NT, XGC-SOL) run on thousands of processors, and the M3D code will likely run on about 100+ processors.

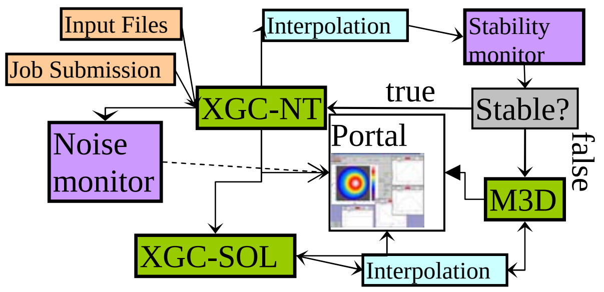

Figure 4. The XGC-M3D workflow

3. Code Coupling

In this project we are developing a coupling framework. The framework is based on the Seine geometry-based interaction model [ZP04], which is motivated by two observations about the targeted applications: (a) formulations of these scientific and engineering applications are based on multi-dimensional geometric discretizations of the problem domain (e.g., grid or mesh) and (b) couplings and interactions in these applications can be defined based on geometric relations in this discretization (e.g., intersecting or adjacent regions). Seine provides a geometry-based virtual shared space interaction abstraction. This abstraction derives from the tuple space model.

However, instead of implementing a general and global interactions space (as in the tuple model), Seine presents an abstraction of transient geometry-based interaction spaces, each of which is localized to a sub-region of the overall geometric domain. This allows the abstraction to be efficiently and scalability implemented and allows interactions to be decoupled at the application level. A Seine interaction space is defined to cover a closed region of the application domain described by an interval of coordinates in each dimension, and can be identified by any set of coordinates contained in the region.

The architecture of the Seine geometry-based coupling framework is illustrated in Figure 5. It differs from existing approaches in several ways. First, it provides a simple but powerful abstraction for interaction and coupling in the form of the virtual geometry-based shared space. Processes register geometric regions of interest, and associatively read and write data associated with the registered region from/to the space in a decoupled manner. Second, it supports efficient local computation of communication schedules using lookups into directory implemented as a distributed hash table. The index space of the hash table is directly constructed from the geometry of the application using Hilbert space filling curves. Processes register their regions of interest with the directory layer, and the directory layer automatically computes communications schedules based on overlaps between the registered geometric regions. Registering processes do not need to know of or explicitly synchronize with other processes during registration and the computation of communication schedules. Finally, it supports efficient and low-overhead processor-to-processor socket-based data streaming and adaptive buffer management.

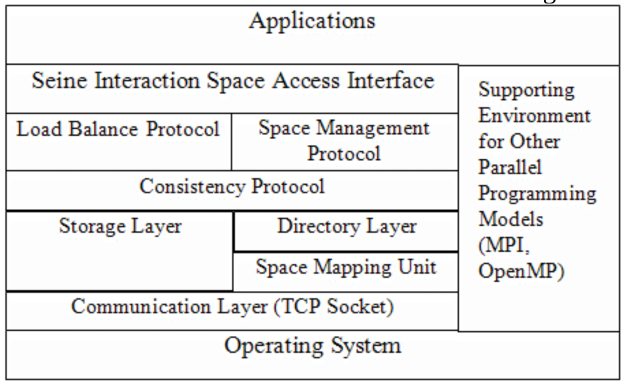
Applications
Seine Interaction Space Access Interface
Load Balance Protocol
Space Management Protocol
Consistency Protocol
Storage Layer
Directory Layer
Space Mapping Unit
Communication Layer (TCP Socket)
Supporting Environment for Other Parallel Programming Models (MPI, OpenMP)
Operating System

Figure 5. Architecture of the Seine geometry based coupling/interaction framework.

Data coupling in memory can be problematic in cases where the volume of data to be shared is too large. In such cases, data has to be moved partially to disk, and effective caching methods have to be deployed. A second, more compute intensive aspect of data coupling is the transformation required when output and input data formats do not match.

In cases where the coupled component codes are not on the same system, data has to be moved from one system to another perhaps over a wide area network [Bhat04]. For example, the XGC and M3D codes currently require data volumes on the order of megabytes per minute to be streamed from the memory of one machine to the memory of another. As computational speeds increase, these volumes will increase significantly.

The Seine model and the Seine-based coupling framework is designed to complement existing parallel programming models and can work in tandem with systems such as MPI, PVM and OpenMP. The design, implementation and experimental evaluation of a prototype implementation of the Seine based coupling framework based on the DoE Common Component Architecture (CCA) and enabling coupling within and across CCA-based simulations are presented in [ZP06].

4. Runtime Monitoring via Workflows

The scale, complexity and dynamism of these simulations coupled with similar scale and complexity of emerging parallel/distributed execution environments requires that these applications be accessed, monitored and controlled during their execution. This is necessary for us to ensure the correct and efficient execution of the simulations. Here, simulation component behaviors and their compositions can no longer be statically defined. Further, their performance characteristics can no longer be derived from a small synthetic run, as they depend on the state of the simulations and the underlying system. Algorithms that worked well at the beginning of the simulation may become suboptimal as the solution deviates from the space the algorithm was optimized for or as the execution context changes. This requirement presents a new set of deployment and runtime management challenges. Further, as these simulations are long running and some, XGC-NT, will run as batch jobs, the monitoring and control activities must be automated based on user defined policies.

We are investigating programming and runtime management solutions to support the development and deployment of applications that can be externally monitored and interactively or *autonomically* controlled. In particular, we are looking into programming and runtime systems that can support efficient and scalable implementations of our simulations. We are starting to design control networks to allow computational elements to be accessed and managed externally, both interactively and using automated policies, to support runtime monitoring, dynamic data injection and simulation control. Here, we are building on our current and prior research efforts and software projects including Accord [Liu06], and Discover [Liu05].

6. References

[ABB+05] I. Altintas, et al., A Framework for the Design and Reuse of Grid Workflows, *Intl. Workshop on Scientific Applications on Grid Computing*, Springer LNCS 3458, 2005.

[Bhat04] V. Bhat, *et al*, "High Performance Threaded Data Streaming for Large Scale Simulations," in Proceedings of 5th International Grid Computing Workshop, IEEE Computer Society Press, 243-250, November (2004).

[BL05] Actor-Oriented Design of Scientific Workflows, S. Bowers, B. Ludaescher, *Intl. Conf. on Conceptual Modeling (ER)*, Klagenfurt, Austria, LNCS, 2005.

[BLL+04] C. Brooks, E. A. Lee, X. Liu, S. Neuendorffer, Y. Zhao, and H. Zheng. Heterogeneous Concurrent Modeling and Design in Java (Vol 1-3). Technical report, Dept. of EECS, University of California, Berkeley, 2004. Technical Memoranda UCB/ERL M04/27, M04/16, M04/17.

[Kep05] Kepler Scientific Workflow System and Project. http://kepler-project.org, 2005

[LAB+06] B. Ludaescher, *et al*. "Scientific Workflow Management and the Kepler System", *Concurrency and Computation: Practice & Experience*, to appear 2006

[LP95] E. Lee, T. Park, Dataflow Process Networks, *Proceedings of the IEEE*, volume 83, pp.773–799, 1995.

[SciDAC] Scientific Discovery through Advanced Computing http://www.osti.gov/scidac/

[ZP04] L. Zhang, M. Parashar, A Dynamic Geometry-Based Shared Space Interaction Framework for Parallel Scientific Applications, 11th Intl. Conf. on High Performance Computing (HiPC), Bangalore, Springer LNCS, 2004.

[ZP06] L. Zhang, M. Parashar, Enabling Efficient and Flexible Coupling of Parallel Scientific Applications, IEEE Intl. Parallel and Distributed Processing Symposium, Rhodes Island, Greece, 2006.

[M3D] http://w3.pppl.gov/~jardin/CEMM/

[XGC-NT] C. S. Chang, Seung-Hoe Ku, and H. Weitzner, "Numerical study of neoclassical plasma pedestal in a tokamak geometry", Phys. Plasmas, 11, 2649 (2004).

[PORTALS] R. Brightwell, *et al.*, "Portals 3.0: Protocol Building Blocks for Low Overhead Communication," Proc. Workshop on Communication Architecture for Clusters, 2002, pp. 164-73.

[Liu06] H. Liu, M. Parashar, "Accord: A Programming Framework for Autonomic Applications", *IEEE Transactions on Systems, Man and Cybernetics, Special Issue on Engineering Autonomic Systems*, 2006.

[Liu05] H. Liu, M. Parashar, "Rule-based Monitoring and Steering of Distributed Scientific Applications", *International Journal of High Performance Computing and Networking*, Vol. 3, No. 4, pp. 272–282, 2005.

[SDM] Scientific Data Management Center. http://sdmcenter.lbl.gov/