Experiments with Wide Area Data Coupling Using the Seine Coupling Framework*

Li Zhang[1], Manish Parashar[1], and Scott Klasky[2]

[1] TASSL, Rutgers University, 94 Brett Rd. Piscataway, NJ 08854, USA
[2] Oak Ridge National Laboratory, P.O. Box 2008, Oak Ridge, TN, 37831

Abstract. Emerging scientific and engineering simulations often require the coupling of multiple physics models and associated parallel codes that execute independently and in a distributed manner. Realizing these simulations in distributed environments presents several challenges. This paper describes experiences with wide-area coupling for a coupled fusion simulation using the Seine coupling framework. Seine provides a dynamic geometry-based virtual shared space abstraction and supports flexible, efficient and scalable coupling, data redistribution and data streaming. The design and implementation of the coupled fusion simulation using Seine, and an evaluation of its performance and overheads in a wide-area environment are presented.

1 Introduction

Scientific and engineering simulations are becoming increasingly sophisticated as they strive to achieve more accurate solutions to realistic models of complex phenomena. A key aspect of these emerging simulations is the modeling of multiple interacting physical processes that comprise the phenomena being modeled. This leads to challenging requirements for coupling between multiple physical models and associated parallel codes that execute independently and in a distributed manner. Coupled systems provide the individual models with a more realistic simulation environment, allowing them to be interdependent on and interact with other physics models in the coupled system and to react to dynamically changing boundary conditions. For example, in plasma science, an integrated predictive plasma edge simulation couples an edge turbulence code with a core turbulence code through common grids at the spatial interface [11].

However, achieving efficient, flexible and scalable coupling of physics models and parallel application codes 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. This requires that: (1) communication schedules between individual processors executing each of the coupled simulations are computed efficiently, locally, and on-the-fly, without

* The research presented in this paper is supported in part by National Science Foundation via grants numbers ACI 9984357, EIA 0103674, EIA 0120934, ANI 0335244, CNS 0305495, CNS 0426354 and IIS 0430826, and by Department of Energy via the grant number DE-FG02-06ER54857.

Y. Roberts et al. (Eds.): HiPC 2006, LNCS 4297, pp. 229–241, 2006.

requiring synchronization or gathering global information, and without incurring significant overheads on the simulations; and (2) data transfers are efficient and happen directly between the individual processors of each simulation. Furthermore, specifying these coupling behaviors between the simulation codes using popular message-passing abstractions can be cumbersome and often inefficient, as they 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.

This paper presents experiences with wide-area coupling for a coupled fusion simulation using the Seine [6] coupling framework. The objective of this paper is to evaluate the ability of Seine to support the coupling requirements of the recent CPES [1] DoE SciDAC Fusion Simulation Project. Seine provides a semantically specialized virtual shared space coupling abstraction and efficient, flexible and scalable mechanisms for data coupling, redistribution and transfer [6]. The Seine shared space abstraction is derived from the tuple space model. It presents an abstraction of a transient interaction space that is semantically specialized to the application domain. The specialization is based on the observation that interactions in the target applications can be specified on an abstract spatial domain that is shared by the interacting entities, such as a multi-dimensional geometric discretizations of the problem domain (e.g., grid or mesh). Further, the interactions are local in this domain (e.g., intersecting or adjacent regions). The shared spaces provided by Seine are localized to these regions of interaction, which are sub-regions of the overall abstract domain. This allows the Seine abstraction to be efficiently and scalably implemented and allows interactions to be decoupled at the application level [6].

The Seine coupling framework differs from existing approaches in several ways. First, it provides a simple but powerful abstraction for interaction and coupling in the form of a virtual semantically-specialized shared space. This may be the geometric discretization of the application domain or an abstract multi-dimensional domain defined exclusively for coupling purposes. Processes register regions of interest, and associatively read and write data associated with the registered region from/to the space in a decoupled manner. Registering processes do not need to know of, or explicitly synchronize with, other processes during registration and computation of communication schedules. Second, it supports efficient local computation of communication schedules using lookups into a directory, which is implemented as a distributed hash table. Finally, it supports efficient and low-overhead processor-to-processor socket-based data streaming and adaptive buffer management. The Seine model and the Seine-based coupling framework complement existing parallel programming models and can work in tandem with systems such as MPI, PVM and OpenMP.

This paper presents the coupling, data redistribution and data transfer requirements of the coupled fusion simulations, and describes a prototype implementation of these simulations using Seine. The paper then describes experiments with wide-area coupling and demonstrates that the Seine-based implementation can potentially meet the

[1] Center for Plasma Edge Simulation.

data coupling requirements of the project. The experiments investigate the behavior and performance of Seine-based coupling between simulations running at Oak Ridge National Laboratory (ORNL) in TN, and Rutgers University (RU) in NJ, and measure the time required for data redistribution and streaming as well as throughputs achieved for different distribution patterns and data sizes. These experiments are intended to be a proof-of-concept to demonstrate the feasibility of using the Seine coupling framework to support data coupling in real Fusion simulations.

The rest of the paper is organized as follows. Section 2 presents related work. Section 3 presents an overview of the Seine coupling framework. Section 4 describes the coupled fusion simulations. Section 5 presents the Seine-based prototype implementation and experimental evaluation of the simulations. Section 6 presents conclusions and outlines future directions.

2 Background and Related Work

Parallel data redistribution (also termed as the MxN problem) is a key aspect of the coupling problem. It addresses the problem of transferring data from a parallel program running on M processors to another parallel program running on N processors. Different aspects of this problem have been addressed by recent projects such as MCT [4], InterComm [3], PAWS [2], CUMULVS [1], DCA [5], DDB [9] etc., with different foci and approaches. These systems differ in the approaches they use to compute communication schedules, the data redistribution patterns that they support, and the abstractions they provide to the application developer. Most of these existing systems gather distribution information from all the coupled models at each processor and then locally compute data redistribution schedules. This implies a collective communication and possible global synchronization across the coupled systems, which can be expensive and limit scalability. Further, abstractions provided by these systems are based on message passing, which requires explicit definition of matching of sends and receives and synchronized data transfers. Moreover, expressing very general redistribution patterns using message passing type abstractions can be quite cumbersome.

The Seine geometry-based coupling framework provides a simple but powerful high-level abstraction, based on a virtual associative shared space, to the application developer. Communication schedules are computed locally and in a decentralized manner using a distributed directory layer. All interactions are completely decoupled and data transfer is socket-based and processor-to-processor, and can be synchronous or asynchronous. The Seine framework is introduced below.

3 The Seine Geometry-Based Coupling Framework

Seine is a dynamic geometry-based interaction/coupling framework for parallel scientific and engineering applications. Note that the geometry may be based on the geometric discretization of the application domain or an abstract multi-dimensional domain defined exclusively for coupling purposes. Seine spaces can be dynamically created and destroyed. They complement existing parallel programming models and can co-exist with them during program execution.

3.1 The Seine Geometry-Based Coupling Model

Conceptually, the Seine coupling/interaction model is based on the tuple space model where entities interact with each other by sharing objects in a logically shared space. However there are key differences between the Seine model and the general tuple space model. In the general tuple space model, the tuple space is global, spans the entire application domain, can be accessed by all the nodes in computing environments, and supports a very generic tuple-matching scheme. These characteristics of the general tuple model have presented several implementation challenges. In contrast, Seine defines a virtual dynamic shared space that spans a specific geometric region, which is a subset of the entire application domain, and is only accessible by the dynamic subset of nodes to which the geometric region is mapped. Further, objects in the Seine space are geometry-based, i.e. each object has a geometric descriptor that specifies the region in the application domain that the object is associated with. Applications use these geometric descriptors to associatively *put* and *get* objects to/from a Seine space. These interactions are naturally decoupled.

The Seine API consists of a small set of simple primitives as listed in Table 1. The *register* operation allows a process to dynamically register a region of interest, which causes it to join an appropriate existing space or create a new space if one does not exist. The *put* operator is used to write an object into the space, while the *get* operator

Table 1. Primitives provided by the Seine framework

Primitives	**Description**
init(bootstrap-server-IP)	Uses a bootstrap mechanism to initialize the Seine runtime system.
register(object-geometric-descriptor)	Registers a region with Seine.
put(object-geometric-descriptor, object)	Inserts a geometric object into Seine.
get(object-geometric-descriptor, object)	Retrieves and removes a geometric object from Seine. This call will block until a matching object is *put*.
deregister(object-geometric-descriptor)	De-registers a region from Seine.

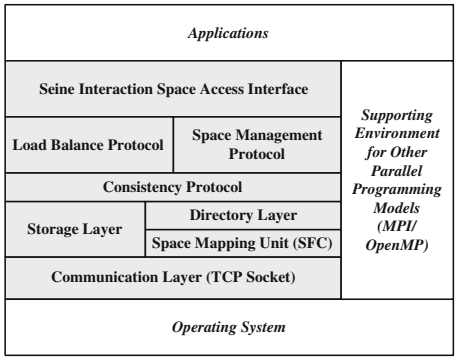

Fig. 1. Architectural overview of the Seine framework

retrieves a matching object from the space, if one exists. If no matching object exists, it will block until a matching object is *put* into the space. The *deregister* operation allows a processor to de-register a previously registered region.

3.2 Design of the Seine Geometry-Based Coupling Framework

A schematic overview of the Seine architecture is presented in Figure 1. The framework consists of three key components: a directory layer, a storage layer, and a communication layer. The distributed directory layer enables the registration of spaces and the efficient lookup of objects using their geometric descriptors. It detects geometric relationships between shared geometry-based objects and manages the creation of shared spaces based on the geometric relationship detected, the lifetime of shared spaces including merging or splitting, and the destruction of shared spaces. The storage layer consists of the local storage associated with registered shared spaces. The storage for a shared space is distributed across the processors that have registered the space. The communication layer provides efficient data transfer between processors. Since coupling and parallel data redistribution for scientific application typically involves communicating relatively large amounts of data, efficient communication and buffer management are critical. Further, this communication has to be directly between the individual processors. Currently Seine maintains the communication buffers at each processors as a queue, and multiple sends are overlapped to better utilize available bandwidth. Adaptive buffer management strategies are being integrated.

To share an object in the space, the geometric region of the object first needs to be *register*ed with Seine. During registration, the Seine runtime system first maps the region defined in the n-dimensional application space to a set of intervals in a 1-dimensional index space using the Hilbert Space Filling Curve (SFC) [8]. The index intervals are then used to index into the Seine directory to locate the processor(s) to which the region is mapped. Note that the mapping is efficient and only requires local computation.

A new registration request is compared with existing spaces. If overlapping regions exist, a union of these regions is computed and the existing shared spaces are updated to cover the union. Note that this might cause previously separate spaces to be merged. If no overlapping regions exist, a new space is created. After registration, objects can be *put*/*get* to/from the shared space. When an object is *put* into the space, the update has to be reflected to all processors with objects whose geometric regions overlap with that of the object being inserted. This is achieved by propagating the object or possibly corresponding parts of the object (if the data associated with the region is decomposable based on sub-regions, such as multi-dimensional arrays) to the processors that have registered overlapping geometric regions. As each shared space only spans a local communication region, it typically maps to a small number of processors and as a result update propagation does not result in significant overheads. Further, unique tags are used to enable multiple distinct objects to be associated with the same geometric region. Note that Seine does not impose any restrictions on the type of application data structures used. However, the current implementation is optimized for multi-dimensional arrays. The *get* operation is simply a local memory copy from Seine's buffer to the application's buffer. Further details of Seine can be found in [6,7].

3.3 Coupling Parallel Scientific Applications Using Seine

Developing coupled simulations using the Seine abstraction consists of the following steps. First, the coupled simulations register their regions of interests, either in the geometric discretization of the application domain or in an abstract n-dimensional domain defined exclusively for coupling purposes. The registration phase detects geometric relationships between registered regions and results in the dynamic creation of a virtual shared space localized to the region and the derivation of associated communication schedules. Coupling between the simulations consists of one simulation writing data into the space and the other simulation independently reading data from the space. The actual data transfer is point-to-point between the corresponding source and destination processors of the respective applications.

4 Coupling Requirements in the CPES SciDAC Fusion Simulation Project

4.1 An Overview of the CPES Fusion Simulation Project

The CPES DoE SciDAC Fusion project is developing a new integrated predictive plasma edge simulation code package that is applicable to 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) [10]. The plasma edge includes the region from the top of the pedestal to the scape-off layer and the divertor region bounded by a material wall. A multitude of non-equilibrium physical processes on different spatio-temporal scales present in the edge region demand 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, massively parallel kinetic transport code for kinetic transport physics using a particle-in-cell (PIC) [12] approach. However, a fluid code is more efficient in terms of computing time, for studying the large scale MHD phenomena, such as Edge Localized Modes (ELMs) [12]. Furthermore, such an event is separable since its time scale is much shorter than that of the transport. The kinetic and MHD codes must however be integrated together for a self-consistent simulation as a whole. Consequently, the edge turbulence PIC code (i.e., XGC [13]) will be connected with the microscopic MHD code (i.e., M3D) using common grids at the spatial interface to study the dynamical pedestal-ELM cycle.

4.2 Data Coupling in the CPES Fusion Simulation Project

The coupled parallel simulation codes, XGC and M3D, will be run on different numbers of processors on different platforms. The overall workflow illustrating the coupling between XGC and M3D code is shown in Figure 2. The coupling begins with the generation of a common spatial grid. XGC then calculates two dimensional density,

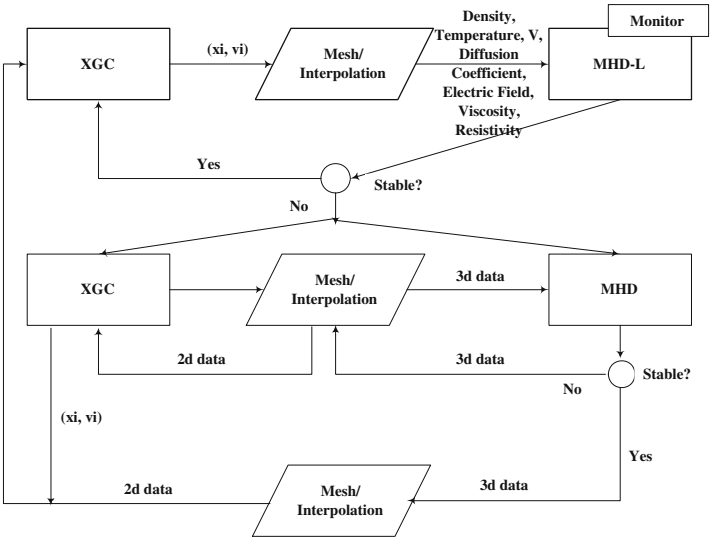

Fig. 2. Workflow illustrating the coupling between XGC and M3D

temperature, bootstrap current, and viscosity profiles in accordance with neoclassical and turbulent transport, and sends these to M3D. The input pressure tensor and current information are used by M3D to evolve the equilibrium magnetic field configuration, which it then sends back to XGC to enable it to update its magnetic equilibrium and to check for stability. During and after the ELM crash, the pressure, density, magnetic field and current will be toroidally averaged and sent to XGC. During the ELM calculation, XGC will evaluate the kinetic closure information and kinetic E_r evolution and send them to M3D for a more consistent simulation of ELM dynamics. The XGC and MHD codes [12] use different formulations and domain configurations and decompositions. As a result, a mesh interpolation module (referred to as MI) is needed to translate between the mesh/data used in the two codes.

Challenges and Requirements. In the CPES project, XGC will be running on a large number of processors while M3D will typically run on 128 or fewer processors. As a result, coupling these codes will require data redistribution. Note that in this case, the redistribution is actually MxPxN, where the XGC code runs on M processors, the interpolation module (MI) runs on P processors, and the M3D code runs on N processors.

The fusion simulation application imposes strict constraints on the performance and overheads of data redistribution and transfer between the codes. Since the integrated system is constructed so as to overlap the execution of XGC with stability check by M3D, it is essential that the result of the stability check is available by the time it is needed by XGC, otherwise the large number (1000s) of processors running XGC will remain idle offsetting any benefit of a coupled simulation. Another constraint is the overhead that the coupling and data transfer imposes on the simulations.

4.3 A Prototype Coupled Fusion Simulation Using Seine

Since the CPES project is at a very early stage, the scientists involved in the project are still investigating the underlying physics and numerics, and the XGC and M3D codes are still under development. However, the overall coupling behaviors of the codes are reasonably understood. As a result, this paper uses synthetic codes, which emulate the coupling behaviors of the actual codes but perform dummy computations, to develop and evaluate the coupling framework. The goal is to have the coupling framework ready when the project moves to production runs. The configuration of the mock simulation using the synthetic codes is shown in Figure 3. In the figure, the coupling consists of two parts, the coupling between XGC and MI and the coupling between MI and M3D.

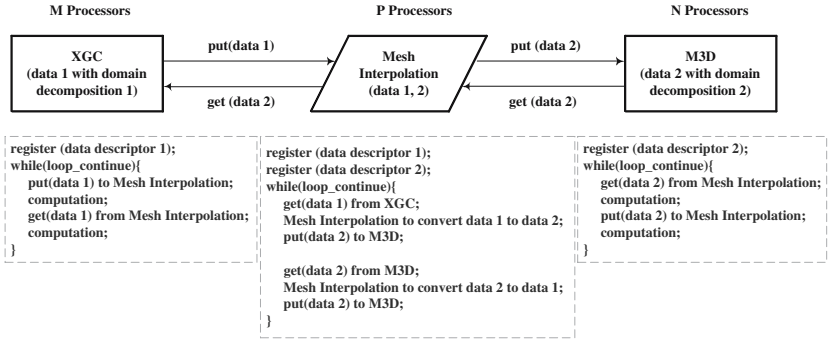

Fig. 3. Configuration of the mock coupled fusion simulation

Domain decompositions: The entire problem domain in the coupled fusion simulation is a 3D toroidal ring. The 3D toroidal ring is then sliced to get a number of 2D poloidal planes as the computation domains. Each plane contains a large number of particles, each of which is described by its physical location using coordinates and a set of physics variables. Each 2D poloidal plane is assigned to and replicated on a group of processors. Since XGC and M3D use different domain decompositions, the numbers of planes in the two codes are different, and MI is used to map the XGC domain decomposition to the M3D domain decomposition.

Coupled fusion simulations using Seine Shared Spaces: Recall that coupling in Seine is based on a spatial domain that is shared between the entities that are coupled. This may be the geometric discretization of the application domain or may be an abstract multi-dimensional domain defined exclusively for coupling purposes. The prototype described here uses the latter.

Given that the first phase of coupling between XGC and M3D is essentially based on the 2D poloidal plane, a 3D abstract domain can be constructed as follows: The X-axis represents particles on a plane and is the dimension that is distributed across the processors. The Y-axis represents the plane id. Each processor has exactly one plane and may have some or all the particles in that plane. The Z-axis represents application variables associated with each particle. Each processor has all the variables associated

with each particle that is mapped to it. Using this abstract domain, Seine-based coupling is achieved as follows. Each XGC processor registers a region in the 3D abstract domain based on the 2D poloidal plane and the particles assigned to it, and the variables associated with each particle. The registered region is specified as a 6-field tuple and represents a 2D plane in the 3D abstract domain, since each processor is assigned particles on only one poloidal plane. Each processor running MI similarly registers its corresponding region in the 3D abstract domain. Note that since MI acts as the "coupler" between XGC and M3D, these processors register regions twice - once corresponding to the XGC domain decomposition and the second time corresponding to M3D domain decomposition. Once the registration is complete, the simulations can use the operators provided by Seine, i.e., *put* and *get*, to achieve coupling.

5 Prototype Implementation and Performance Evaluation

The schematic in Figure 4 illustrates a prototype implementation of a Seine-based coupled simulation. Note that, while the figure illustrates a MxN coupled simulation, the configuration for a coupled MxPxN simulation is similar. The Seine implementation requires a Seine-proxy, which is a local daemon process that resides on each processor using Seine. The Seine distributed directory layer deterministically maps the shared abstract domain onto the Seine infrastructure processors. The Seine distributed directory runs on X processors, which may or may not overlap with the M, P and N processors running XGC, MI and M3D respectively. The Seine-proxy at each processor is initialized by the *init* call within the application code. Once the Seine-proxy is initialized, it handles all the processor interaction with Seine including *register*, *put* and *get* operations.

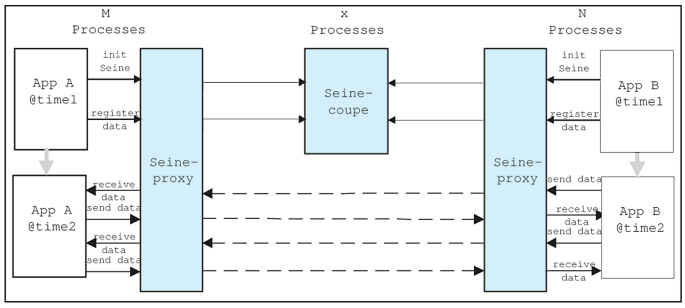

Fig. 4. A prototype schematic of coupling and data redistribution using the Seine framework

5.1 Experiments with Wide-Area Coupling Using the Prototype Seine-Based Fusion Simulation

The experiments presented in this section were conducted between two sites: a 80 nodes cluster with 2 processors per node at Oak Ridge National Laboratory (ORNL) in TN, and 64 node cluster at the CAIP Center at Rutgers University in NJ. The synthetic XGC code ran on the ORNL cluster and the MI module and the synthetic M3D code ran on

the CAIP cluster. That is, in the MxPxN coupling, site M was at ORNL and sites P and N were at CAIP. The two clusters had different processors, memory and interconnects. Due to security restrictions at ORNL, these experiments were only able to evaluate the performance of data transfers from ORNL to CAIP, i.e., XGC pushing data to the MI module, which then pushes the data to M3D.

In the experiments below, the XGC domain was decomposed into 8 2D poloidal planes, while the M3D problem domain was decomposed into 6 2D poloidal planes. The number of particles in each plane was varied in the different experiments. Each particle is associated with 9 variables. Since the *get* operation in Seine is local and does not involve data communication, the evaluations presented below focus on the *put* operation, which pushes data over the network. The experiments evaluate the operation cost and throughput achieved by the *put* operation.

Cost of the put *operation:* In this experiment, 7,200 particles were used in each poloidal plane resulting in an abstract domain of size 7,200x8x9 between XGC and MI and 7,200x6x9 between MI and M3D. The number of processors at site M, which ran the XGC code, was varied. As the number of processors at site M increased, the absolute time for the *register* and *put* operations decreased since operation costs are directly affected by the size of the region. The decrease in absolute time cost is because the size of the entire abstract domain is fixed and as the number of processor increases, each processor registers/puts a smaller portion of this domain, resulting in a decrease in the absolute operation cost. Since the size of the region varies in the above metric, a normalized cost for the operations is calculated by dividing the absolute cost of an operation by the size of region involved. The normalized cost increases as the system size increases. Several factors contribute to this increase, including blocked-waiting time within a *register* operation, and message and data transfer costs associated with a *register* or *put* operation. A detailed analysis of this behavior can be found in [7].

Throughput achieved: The goal of this experiment is to measure the per processor throughput that can be achieved during wide-area data coupling for different system and abstract domain sizes. In the experiment, the number of particles per poloidal plane was varied to be 7,200, 14,400, and 28,800, and the number of processors running XGC at site M were varied to be 8, 16, 32, 64 and 128. Throughput per processor in this experiment was calculated as the ratio of the average data size used by a *put* operation to the average cost of a *put* operation. Note that data transfers from the processors at site M occur in parallel and the effective application level throughput is much higher. The per processor throughput at site M is plotted in Figure 5(a), and the estimated effective system throughputs computed assuming different levels of concurrency for the data transfer are plotted in Figure 5(b) and (c). Two observations can be made from Figure 5(a). First, the per processor throughput at site M decreases with the number of processors used at site M for all the abstract domain sizes tested. This is because the wide-area link is shared and when the number of processors increases the bandwidth available to each processor decreases, resulting in a lower throughput on each processor. Second, for the same number of processors at site M, in most cases, the per processor throughput for smaller abstract domain sizes is higher than the throughput for larger abstract domain sizes. This is because, for larger abstract domain sizes, the size of data to be redistributed is correspondingly larger, resulting in a more congested network.

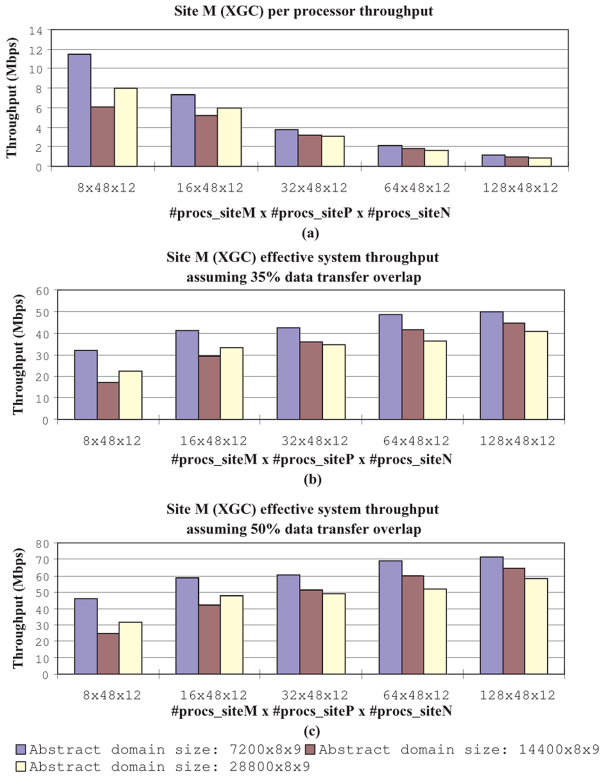

Fig. 5. (a) Per-processor throughput for XGC at site M; (b) Estimated effective system throughput assuming a data transfer overlap of 35%; (c) Estimated effective system throughput assuming a data transfer overlap of 50%

Further, the processors at site P are connected to the processors at both site M and site N. Consequently, site M processors have to compete with site N for connections with site P, which further causes the throughput to decrease for larger abstract domain sizes.

In the CPES Fusion project, site M (running XGC) throughput is a key requirement that must be met by the coupling framework. An initial estimate for the transfer rate from XGC to MI is 120Mbps. The estimated effective system throughput, based on the per processor bandwidth measured above and assuming 35% and 50% overlap in the per processor data transfer respectively, are plotted in Figure 5(b) and (c). Assuming that the system running XGC has 32 IO nodes, as seen from these plots, the estimated effective system throughputs are 34 - 42Mbps assuming a 35% overlap and 50 - 60Mbps assuming a 50% overlap. While these figures are still not close to the Fusion throughput requirement, we believe that these are conservative numbers and that Seine can support the required throughput when used in a real production scenario. This is because (1) these experiments assumed an extreme case where data was continuously generated by XGC, which is not realistic, and (2) these experiments use the Internet for the wide-area data-transfers while a real production run would use a dedicated and customized high-speed interconnect.

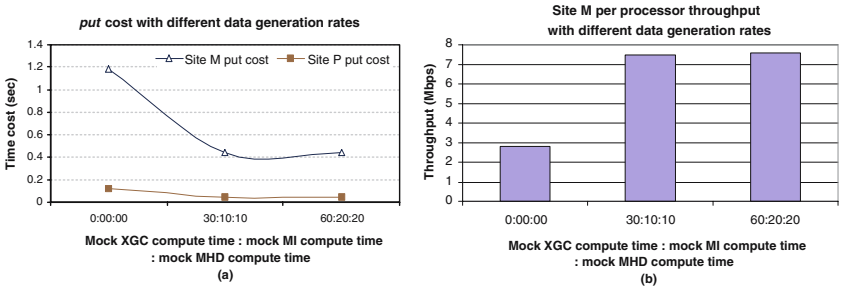

Fig. 6. (a) *put* operation cost at site M and P for different data generation rates; (b) Per processor throughput at site M for different data generation rates

Effect of data generation rates: This experiment evaluated the effect of varying the rate at which data was generated by XGC at site M. In this experiment, XGC generated data at regular intervals, between which, it performed computations. It is estimated by the physicists that on average, XGC requires 3 times the computation as compared to MI and M3D. As a result, the experiment used three sets of computes times for XGC, MI and M3D of (1) 0, 0 and 0 seconds (corresponding to the previous experiments), (2) 30, 10 and 10 seconds, and (3) 60, 20 and 20 seconds respectively. The results are plotted in Figure 6. The plots show that, as expected, the cost of the *put* operation and the throughput per processor improves as the data generation rate reduces.

6 Conclusion

The paper presented experiments and experiences with wide area coupling for a fusion simulation using the Seine coupling framework. The goal of these experiments is to evaluate the ability of Seine to support the coupling requirements of the ongoing CPES DoE SciDAC Fusion Simulation Project. Seine presents a high-level semantically specialized shared space abstraction to application and provides efficient and scalable data coupling, data redistribution and data transfer services. The experimental results using a prototype coupled fusion simulation scenario demonstrate the performance, throughput and low simulation overheads achieved by Seine.

Note that the experiments presented here are a proof-of-concept and demonstrate feasibility. As the project progresses and real codes and more detailed requirements become available, the Seine framework and abstractions will have to be tuned to ensure that it can support true production runs.

References

1. J.A. Kohl, G.A. Geist. Monitoring and steering of large-scale distributed simulations, IASTED Intl Conf on Appl. Modl. and Sim., Cairns, Queensland, Aus., Sept. 1999.
2. K. Keahey, P. Fasel, S. Mniszewski. PAWS: Collective interactions and data transfers. High Perf. Dist. Comp. Conf, San Francisco, CA, Aug. 2001.
3. J.Y. Lee and A. Sussman. High performance communication between parallel programs. 2005 Joint Wksp on High-Performance Grid Computing and High Level parallel Programming Models. IEEE Computer Society Press. Apr. 2005.

4. J.W. Larson, R.L. Jacob, I.T. Foster, and J. Guo. The Model Coupling Toolkit. Intl Conf on Comp. Sci. 2001, vol. 2073 of Lec. Nt. in Comp. Sci., pg 185-194, Berlin, 2001. S.-V.
5. F. Bertrand and R. Bramley. DCA: A distributed CCA framework based on MPI. The 9th Intl Wksp on High-Level Par. Prog. Mdls and Sup. Env., Santa Fe, NM, Apr. 2004. IEEE Press.
6. L. Zhang and M. Parashar. Enabling Efficient and Flexible Coupling of Parallel Scientific Applications. The 20th IEEE Intl Par. and Dist. Proc. Symp. Rhodes Island, Greece. IEEE Comp. Soc. Press.
7. L. Zhang and M. Parashar. Seine: A Dynamic Geometry-based Shared Space Interaction Framework for Parallel Scientific Applications. Con. and Comp.: Prac. and Exp. John Wiley and Sons. 2006.
8. T. Bially. A class of dimension changing mapping and its application to bandwidth compression. PhD thesis, Poly. Inst. of Brklyn., Jun. 1967.
9. L.A. Drummond, J. Demmel, C.R. Mechoso, H. Robinson, K. Sklower, and J.A. Spahr. A data broker for distributed computing environments. Intl Conf on Comp. Sc., pg 31-40, 2001.
10. ITER. http://www.iter.org.
11. I. Manuilskiy and W.W. Lee. Phys. Plasmas 7, 1381 (2000).
12. W.W. Lee, S. Ethier, W.X. Wang, W.M. Tang, and S. Klasky. Gyrokinetic Particle Simulation of Fusion Plasmas: Path to Petascale Computing. SciDAC 2006. June 25-29. Denver.
13. C. Chang, S. Ku, and H. Weitzner, Numerical Study of Neoclassical Plasma Pedestal in a Tokamak Geometry. Phys. Plasmas 11, 2649 2667 (2004).