High Performance Computing Issues for Grid Based Dynamic Data-Driven Applications *

Craig C. Douglas
Computer Science Department, University of Kentucky
773 Anderson Hall, Lexington, KY 40506-0046, USA and
Computer Science Department, Yale University
P.O. Box 208285, New Haven, CT 06520-8285, USA
Email: douglas-craig@cs.yale.edu

Gabrielle Allen
Department of Computer Science and Center for Computation and Technology, Louisiana State University
302 Johnston Hall, Louisiana State University, Baton Rouge, LA 70803, USA
Email: gallem@cct.lsu.edu

Yalchin Efendiev and Guan Qin
Institute for Scientific Computation, Texas A&M University, Address,
612 Blocker, 3404 TAMU, College Station, TX 77843-3404, USA
Email: efendiev@math.tamu.edu and guan.qin@tamu.edu

ABSTRACT

DDDAS creates a rich set of new challenges for applications, algorithms, systems software, and measurement methods. DDDAS research typically requires strong, systematic collaborations between applications domain researchers and mathematics, statistics, and computer sciences researchers, as well as researchers involved in the design and implementation of measurement methods and instruments. Consequently, most DDDAS projects involve multidisciplinary teams of researchers.

DDDAS enabled applications run in a different manner than many traditional applications. They place different strains on high performance systems and centers due to dynamic and unpredictable changes in resources that are required during long term runs. An on demand environment is required. In this paper, we will also categorize many of these differences.

Keywords: DDDAS, HPC, Dynamic scheduling, Resource allocation, Data centers, Data filters, Data assimilation.

1. INTRODUCTION

Dynamic data-driven application simulation (DDDAS) is an emerging field of science and technology [1-6]. Developing a symbiotic two way interaction between (intelligent) sensors and multi-model, multi-scale simulations provides a more accurate methodology. Implementing DDDAS, instead of the traditional take an initial guess and run a simulation for some short period of time, has impacts on high performance computing that effects the following:

1. Scheduling and allocations
2. Infrastructure (including visualization, computer environment and performance, and communication)
3. Middleware
4. Data integrity

DDDAS is an example of on demand everything. In order to actually implement DDDAS correctly and precisely, we need to be able to change all high performance computing (HPC) parts of the computation right now, again later, and we cannot define everything in advance.

When DDDAS works well, it assumes that almost everything can be modified during the course of a long term simulation. The diagram in Fig. 1 shows how a number of elements might interact with each other: All six of the components may change without resorting to a new simulation as the computation progresses. Virtually all DDDAS applications are multiscale in nature. As the scale changes, models change, which in turn, changes which numerical algorithms must be used and possibly the discretization methods. To date, almost all DDDAS applications involve a complicated time dependent, nonlinear set of coupled partial differential equations, which adds to the complexity of dynamically changing models and numeric algorithms. It also causes computational requirements to change, particularly if dynamic adaptive grid refinement or coarsening methods are used. This is not something normally supported at supercomputer centers, but is slowly being added.

2. SCHEDULING

DDDAS impacts scheduling in three ways directly:

1. Model-model coupling across wide area networks
2. The controlling computational device
3. Immediate, dynamic on demand scheduling

Model-model coupling can be quite complicated to implement. Besides mathematical modeling issues there is a HPC aspect that is subtle. Each model may require substantial computational resources to be effective. Each model may be owned by a specific entity that requires that its code must be run at a specific location. In both cases, the models may be run in different locations, separated by

* This paper was supported in part by National Science Foundation grants CNS-0540178, CNS-0540136, ACI-0324876, and CNS-0540374.

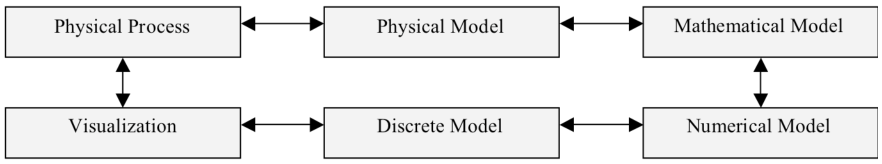

Fig. 1: DDDAS processing.

significant distances.

The network then must be scheduled if substantial amounts of data will be moved between the different locations. There is an issue of high speed, high latency versus low speed, low latency networks. Worse, there is usually no way to control how the data moves between data centers. One ancient, low speed Ethernet cable somewhere in a building can intermittently ruin high speed data transfers if a router occasionally and consistently uses it.

Co-scheduling of both computational resources becomes necessary when different computer centers are involved. Additionally, the network between the sites may have to be scheduled since there are only a maximum number of bits per second that can be transferred across the network connecting the sites. When more than two sites are involved or a regular, public network is involved (versus a research, monitored, and scheduled network), things get really interesting.

The controlling computer in a simulation is not necessarily a fixed workstation with a fixed network address. It is becoming imperative to support mobile devices (e.g., a laptop, PDA, or cell phone) as the controlling device. These devices move and must be able to reconnect to a simulation, be findable when connected to a network, and to robustly run complicated simulations with similar devices coming online and going offline randomly during a computation. Some device on the network typically acts as a broker in order to keep track of all of the sensors and computers involved as well as to provide security to the DDDAS.

Examples include a wildland fire, a burning building, or moving water contaminants. The sources of information are not fixed. The person responsible for coordinating a disaster management needs to be in the area to see what is happening (and not necessarily in a fixed location). Techniques developed in point to point (P2P) are useful here. The controller may be a separate device controlling a Grid of devices (sensors and computers). As the device moves, it may change networks, drop out of connectivity, and must be able to rejoin the DDDAS seamlessly.

The DDDAS may require dynamic and immediate on demand scheduling. Since there is a symbiotic relationship between the computational kernels and the sensors, the sensors may indicate to the DDDAS that different mathematical or physical models need to be used. The need may be immediate to continue processing. The computational components corresponding to the request may be on different machines in the Grid and have to be run on demand.

An example is a movable drone that can detect different chemicals in water. If it detects certain petroleum products in water, there are two interesting possibilities: a recent boat sinking, just a leaky fuel tank on a functional boat, or a major (possibly intentional) fuel spill. Checking for further chemicals and concentrations will determine which case exists. One of three different computational models will run after the decision and the sensor may have to be reprogrammed to continue functioning usefully as part of the DDDAS.

The computational components may not be in the same location. However, we need resources immediately and we may be dealing with batch queues. We need either fast injection or pre-injection of the application into the queue.

Fast injection into the queue means that the batch system has the ability to flush the queue quickly. It must be able to write running applications to disk very quickly, which is a serious problem disk since writing many gigabytes of data to disk is slow. In addition, most operating system kernels do not have the ability to roll in a new application while rolling out the old ones in a fast manner. This requires a kernel change, which several computer vendors are trying to implement.

Pre-injection into the queue requires starting an application and then putting it to sleep without allocating any memory for data. The application's footprint in the computer is small and as long as the batch system does not terminate the application for lack of using CPU cycles, restarting it is fast. Ideally the application uses a small amount of memory per processor (e.g., as would be done on machines like an IBM BlueGene/L or an old Intel Paragon). By using more processors, but less memory per processor, we save paging time when restarting and can, thus, be useful to the DDDAS quicker.

3. INFRASTRUCTURE IMPACT

Information about the hardware and software environments is needed to make a DDDAS run efficiently on the available resources. Standard application programming interfaces (APIs) are essential. Creating a standard DDDAS API is an ongoing research area and is beyond the scope of this paper.

The APIs are necessary so that data structures, load balancing and computational algorithms can be dynamically employed. Thus, a DDDAS can run efficiently and not waste resources or time.

Having consistent dynamically linked libraries is essential, yet one of the largest causes of failure to run applications on a Grid. Using the same variant of Linux, for example, is insufficient to guarantee that the libraries are consistent and interoperate. The lack of consistency leads to a very subtle forms of failure that are difficult to track and fix during a DDDAS that must be run during a specific period of time.

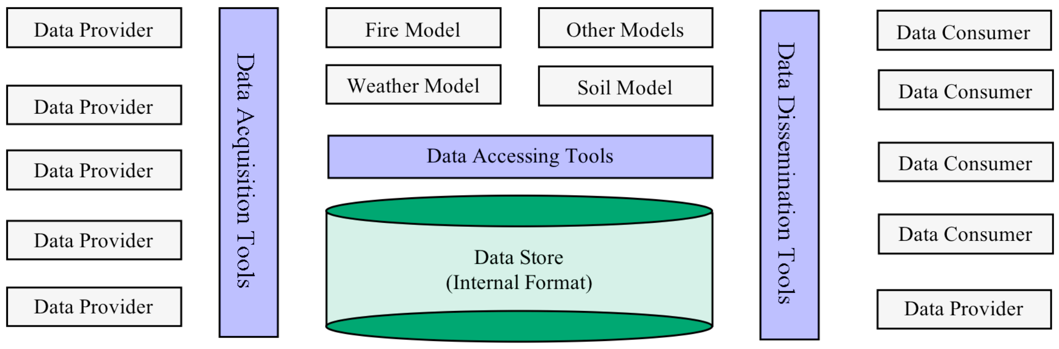

Fig. 2: Data acquisition, accessing, and dissemination software layout in a typical DDDAS project (e.g., a wildland fire DDDAS project).

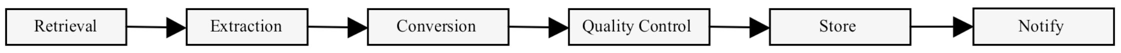

Fig. 3: A General View of Data Acquisition Tools.

The APIs need to be able to help determine when and when to stage executables. We need to know how much allocation time is remaining for specific batch queues so as to prioritize work in these queues. We need to know the queue wait times (before execution begins) to prioritize the locations for scheduling work. Finally, we need the ability to find available hardware resources that we might not normally utilize that are idle.

DDDAS also impacts performance. Using standard APIs like PAPI, the applications can dynamically optimize themselves using standard methods (though painful to write initially). Parameters can also be chosen dynamically, including the frequency of analysis of the overall application's effectiveness.

APIs are needed to negotiate increases or decreases of allocations on the computers in the Grid. The most common causes of workload changes are the following:

1. Dynamic adaptive mesh refinemen and coarsening.
2. Changing the model (e.g., character of the governing equations)

Cyber security has a major impact on DDDAS. Firewalls cause problems starting with data collection and transfers to application transfers to other computer sites on the Grid based on decisions made with respect to available queues. Firewall security levels may change dynamically at some sites based on network flow analysis, which may be effected by the DDDAS itself.

Port blocking is probably the single hardest factor to accommodate. Data may traverse long distances and multiple networks. Anywhere along the routing, a network administrator can arbitrarily block a port for no apparent reason and without notifying anyone. Suddenly the entire DDDAS stops working. Finding the source of the problem is difficult and nontrivial to get fixed.

Both cyber security issues are current research areas for the DDDAS community.

Most DDDAS need real time visualization at some point. We need to be able to provision high speed networks in order to see what the actual bandwidth is. We need flexible scheduling so that high priority visualization can occur when absolutely necessary. We also need to be able work with radically different environments at the same time, e.g., immersive three dimensional devices, a simple LCD screen, and a small screen on a PDA or mobile phone.

4. DATA INTEGRITY

To support DDDAS' requirements, data acquisition, data accessing, and data dissemination tools are typically used. Data acquisition tools are responsible for retrieving of the real-time or near real-time data, processing, and storing them into a common internal data store. Data accessing tools provide common data manipulation support, e.g., querying, storing, and searching, to upper level models. Data dissemination tools read data from the data store, format them based on requests from data consumers and deliver the formatted data to the data consumers. Fig. 2 illustrates a simplified view of a typical DDAS system.

The data used to drive a DDDAS system are retrieved periodically by a data retrieval service, extracted, converted, quality controlled, and then save to the internal data store. After the data is stored, the data accessing tools are notified which can then update the input to the simulation models. The whole process is illustrated in Fig. 3. The extraction process reads the retrieved data based on the meta data associated with them and feeds the extracted values to the conversion model whose major purpose is unit conversion, e.g., from inches to millimeters. The converted data are then analyzed for potential errors and missing values by the quality control model. This control process will ensure the correctness of the data, which is of great importance for the model simulation accuracy. The quality controlled data are then fed to the data storage model, which either saves the data to a central file system or loads them to a central database (this depends on project requirements). The data store model may also need to register the data in a metadata database so that other models can query it later.

DDDAS projects are highly multidisciplinary in nature and are developing comprehensive IT tools, mathematical models, and prototype infrastructure for disaster modeling and management. The projects will bring comprehensive information and numerical prediction where it is needed, at the disaster command center, in real time.

5. CONCLUSIONS

DDDAS is an oncoming force in computational science. The HPC community is ill prepared for it. Basic information technology research is necessary in order to accommodate DDDAS on a Grid. DDDAS will impact the HPC community through middleware to resolve new paradigms for dynamic scheduling, allocations, resource provisioning, data centers, and cyber security..

6. REFERENCES

[1] http://www.dddas.org. Includes project descriptions, many DDDAS workshop virtual proceedings, and links to DDDAS software.

[2] K. Baldridge, G. Biros, A. Chaturvedi, C. C. Douglas, M. Parashar, J. How, J. Saltz, E. Seidel, A. Sussman, January 2006 DDDAS Workshop Report, National Science Foundation, 2006, http://www.dddas.org/nsf-workshop-2006/wkshp_report.pdf.

[3] 2003 Dynamic Data-Driven Application Workshop, F. Darema, ed., in Computational Science - ICCS 2003: 3rd International Conference, Melbourne, Australia and St. Petersburg, Russia, June 2-4, 2003, Proceedings, Part IV, P.M.A. Sloot, D. Abramson, A.V. Bogdanov, J.J. Dongarra, A.Y. Zomaya, Y.E. Gorbachev (Eds.), Lecture Notes in Computer Science, Vol. 2660, Springer-Verlag Heidelberg, 2003, pp. 279-384.

[4] 2004 Dynamic Data-Driven Application Workshop, F. Darema, ed., in Computational Science - ICCS 2004: 4th International Conference, Kraków, Poland, June 6-9, 2004, Proceedings, Part III, Marian Bubak, Geert Dick van Albada, Peter M. A. Sloot, and J.J. Dongarra (eds.), Lecture Notes in Computer Science series, vol. 3038, Springer-Verlag Heidelberg, 2004, pp. 662-834.

[5] 2005 Dynamic Data-Driven Application Workshop, F. Darema, ed., in Computational Science - ICCS 2005: 5th International Conference, Atlanta, Georgia, USA, May 22-25, 2005, Proceedings, Part II, Vaidy S. Sunderam, Geert Dick van Albada, Peter M.A. Sloot, Jack J. Dongarra (eds.), Lecture Notes in Computer Science series, vol. 3515, Springer-Verlag Heidelberg, 2005, pp. 610-745.

[6] 2006 Dynamic Data-Driven Application Workshop, F. Darema, ed., in Computational Science – ICCS 2006: 6th International Conference, Reading, UK, May 28-31, 2006, Proceedings, Part III, edited by V.N. Alexandrov, G.D. van Albada, P.M.A. Sloot, and J.J. Dongarra, Lecture Notes in Computer Science 3993.

[7] C. C. Douglas, A. Deshmukh, M. Ball, R. E. Ewing, C. R. Johnson, C. Kesselman, C. Lee, W. Powell, R. Sharpley, Dynamical data driven application systems: Creating a dynamic and symbiotic coupling of application/simulations with measurements/ experiments, National Science Foundation, Arlington, VA, 2000. http://www.dddas.org/nsf-workshop-2000/worksop_report.pdf.