Center for Plasma Edge Simulation (CPES) -- Rutgers University Final Report (03/06/14)

The CPES scientific simulations run at scale on leadership class machines, collaborate at runtime and produce and exchange large data sizes, which present multiple I/O and data management challenges. During the CPES project, the Rutgers team worked with the rest of the CPES team to address these challenges at different levels, and specifically (1) at the data transport and communication level through the DART framework, and (2) at the data management and services level through the DataSpaces and ActiveSpaces frameworks. These frameworks and their impact are briefly described below.

DART (Decoupled and Asynchronous Remote Data Transfers): Large-scale CPES simulations generate massive amounts of data. This data must be extracted from the system and transported in a timely manner, to remote consumers for online processing, analysis and visualization, monitoring, and decision-making. However, managing and transporting this data is becoming a significant bottleneck, imposing considerable overheads on the applications and leading to inefficient resource utilization and frequent QoS violations. Advanced interconnect architectures and innovative communication protocols, such as for example, customized high speed interconnection buses, one-sided remote direct memory access with zero-copy and OS and application bypass data transfers, have been introduced to address these challenges. Nevertheless, these advances have significantly increased the complexity exposed to the applications, and applications must be adapted and managed at runtime to effectively use these capabilities.

As part of the CPES effort, we have developed DART, an autonomic data management and transport substrate that builds on communication technologies such as RDMA, to provide applications with high-throughput and low-overhead data extraction and transport capabilities. The key objectives of DART are (1) to offload the expensive I/O operations from the nodes running the simulation to dedicated I/O nodes, and thus to enable an application to do useful computational work, (2) to minimize the impact of the I/O operations on the application, (3) to maximize data throughput from the application, and (4) to minimize data transfer latency. DART provides the application layer with a simple and asynchronous API. These mechanisms autonomically adapt to heterogeneous and dynamic data types, data volumes, data rates and application loads. DART has three key components (1) a DART Client that runs in-line with the application and provides communication calls similar to file operations for ease of use, (2) DART Server that runs as a service independent of the application and coordinates, schedules and extracts data from the application, and (3) a DART Receiver that transports data to a remote location.

We implemented DART on the CRAY XT5 machine at Oak Ridge National Laboratory using the Portals RDMA library. We demonstrated that DART can efficiently extract large volumes of data from a live simulation and stream it to remote (downstream consumers), or save it to the local storage system, achieving effective throughputs of over 1TB per hour from 2048 computing cores. Evaluations used testing applications as well as different scientific applications simulating complex plasma phenomena. Simulations running on 1024 and 2048 nodes respectively produced 500GB and 1TB of data. The I/O overhead on the applications in these experiments was only 0.4% and 0.6%. These results show that DART is a viable transfer method.

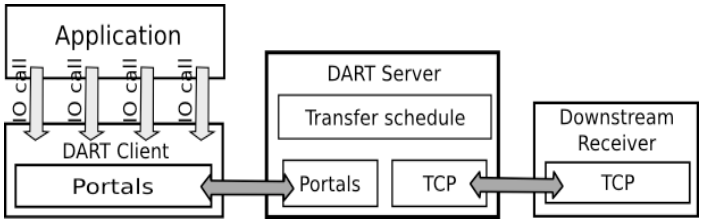

Figure 1 - Architectural overview of DART.

DataSpaces: DataSpaces is a coordination and data-sharing framework that enables dynamic and asynchronous applications interactions. It provides the abstraction of a virtual semantically specialized

shared space that can be associatively and asynchronously accessed using simple yet powerful and flexible operators (e.g., put() and get()) with appropriate data selectors or filters. These operators are agnostic of the location, e.g., source/destination, as well as the data distribution and decomposition of the interacting application components. It also provides a runtime system for "in-the-space" data manipulation and/or reduction, using predefined or customized and user-defined functions, which can be dynamically downloaded and executed at runtime while the data is in-transit through the space. DataSpaces has an extensible architecture and can provide new data services, e.g., data subscription and notification.

The DataSpaces framework provides flexible, decoupled and asynchronous data sharing semantics that enables interactions between multiple distributed application services. It easily integrates into the data pipeline of data workflow engines to complement or replace the more traditional file-based approaches. It alleviates the performance penalties associated with these approaches, e.g., latency, variability, by providing transparent and memory-to-memory data sharing.

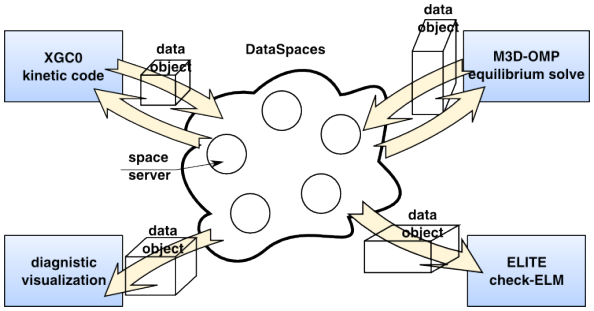

Figure 2 – Code coupling using DataSpaces

The key idea underlying the design of DataSpaces is an efficient addressing/lookup mechanism that uses an indexing and addressing scheme that is semantically meaningful to the applications. This addressing scheme is used to index into a dynamic distributed hash table, which is implemented across the nodes of the space, i.e., a relatively small and dynamic partition called the staging area. It automatically load-balances data inserted into the space, as well as the data look-up operations. DataSpaces has also been extended to support homogeneous as well as heterogeneous distributed data types and have analyzed its behavior while coupling heterogeneous applications. It has been evaluated using multiple application scenarios in both high-performance parallel environments as well as using distributed resources. The results obtained demonstrated overall framework scalability, as well as its ability to sustain high data volume traffic for large-scale applications, for example, the data redistribution in a scenario where one application running on 1024 processors is coupled with a different application running on 8192 processors and the applications exchanged 256GB of data per application iteration for 100 iterations.

ActiveSpaces: Data-intensive application workflows typically transform data they manage and exchange, and often reduce it before the data can be processed by consumer applications or services. For example, coupled application may only require subsets of data, which is sorted and processed before coupling. One approach to address this requirement, which we initially explored in this effort, consists of embedding pre-defined data transformation operations in the staging area to better utilize CPU resources, and to transform the data before it is shipped to the consumer. This approach however requires a priori knowledge of the processing, the data structures and data representation, which may not always be feasible.

ActiveSpaces builds on DataSpaces and explores an alternate paradigm -- it allows application developers to programmatically define data-processing routines, and to dynamically deploy and execute them at runtime, in the staging area where the data resides, rather that moving the data to these processing routines. The ActiveSpaces framework provides (1) programming support for defining the data processing routines, called data kernels, to be deployed and executed on the staging area, and (2) run-time mechanisms for transporting the binary codes associated with these data kernels to the staging area and executing them in parallel on the staging nodes. The programming abstractions allow an application developer to define and implement the data kernels using all constructs of the native programming language (e.g., C). The run-time mechanism enables code offloading and remote execution at the data

source for HPC applications.

The ActiveSpaces architecture consists of two main components: an ActiveSpaces server and an ActiveSpaces client component. The ActiveSpaces server is a stand-alone component, which runs on the staging area and provides data services to user applications. The ActiveSpaces client integrates with user applications and runs on the computing nodes. These components implement the programming API, which is exposed at the application level, and the run-time system, which executes the user-defined data kernels. ActiveSpaces extends the DataSpaces framework and implements new services to apply transformations to the data on the space or to the results of a data request. These services are provided by the run-time execution system. (Rexec).

Data kernels are implemented within an application and have direct knowledge of the structure of the data used in the application. Once deployed on the space, these kernels can access the data directly and manipulate it without additional support from the space, such as parameter marshalling or data decoding. After executing a data kernel, the Rexec layer from the server component returns the results back to the application.

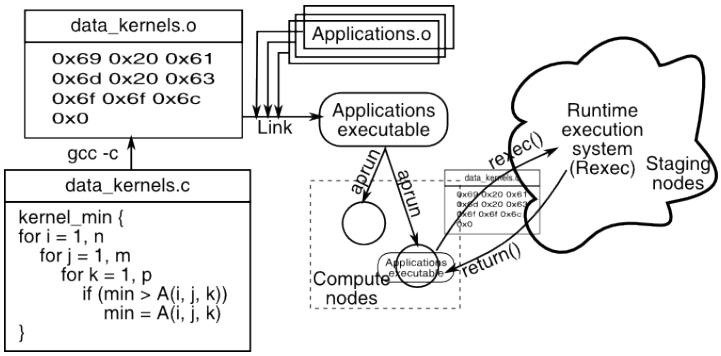

Figure 3 – Dynamic code deployment in ActiveSpaces.

Multiple applications collaborating at runtime can insert data in the space, and can retrieve raw or pre-processed data of interest using the data kernels processing routines. ActiveSpaces can reduce the amount of data that needs to be transferred over the network for data reduction operations. It can also reduce an application's computation time by offloading computations, such as interpolation, redistribution, reformatting, etc., which can be asynchronously executed in parallel on the staging area nodes. ActiveSpaces offers some benefits even in more constrained cases where execution of

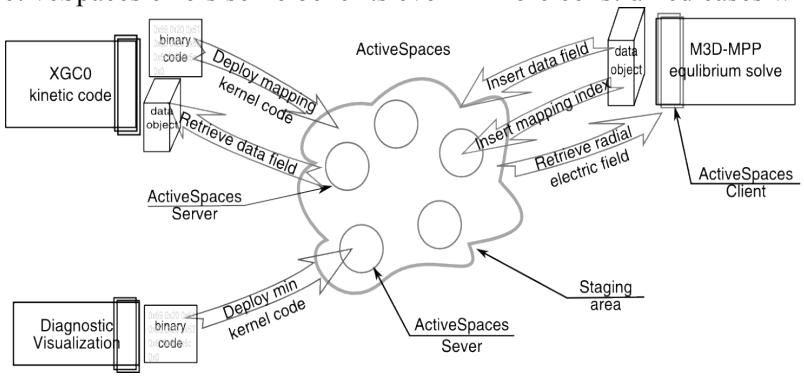

Figure 4 – ActiveSpaces operation in a coupled simulations scenario.

the data kernels is synchronous, because it can better exploit data locality within the staging nodes, because the number of nodes hosting the staging area is much smaller than the number of nodes running the application.

DataSpaces with Fusion Application: As a case study, we used the kinetic XGC0-M3D-OMP code coupling in the Full-ELM workflow for memory-to-memory coupling using DataSpaces. The two component simulations, XGC0 and M3D-OMP run on separate set of compute nodes, progress independently and at different speeds. The data coupling is two-way, XGC0 produces and shares plasma profile data, which is used by M3D-OMP, and M3D-OMP in turn produces and shares equilibrium data, which is used by XGC0 in subsequent steps.

XGC0 and M3D-OMP coordinate and exchange data by creating and sharing in-memory data objects through DataSpaces shared space abstraction. In this case, XGC0 creates the plasma profile data objects named "m3d.in" during a single step, and insert the object into DataSpaces using the put() operator. M3D-OMP can retrieve one or more of "m3d.in" data objects from DataSpaces using the get() operator. Similarly, M3D-OMP creates and inserts equilibrium data objects "g-eqdsk" into DataSpaces, which is consumed by XGC0.

The memory-to-memory coupling is asynchronous and flexible, and it allows the simulation codes to create and share multiple versions and multiple copies of data objects. The number of coupling rounds and synchronizations are driven by the applications at runtime and are not pre-orchestrated.

Relevant Publications:

1. F. Zhang, S. Lasluisa, T. Jin, I. Rodero, H. Bui, M. Parashar, "In-situ Feature-based Objects Tracking for Large-Scale Scientific Simulations", In International Workshop on Data-Intensive Scalable Computing Systems (DISCS) in conjunction with the 2012 ACM/IEEE Supercomputing Conference (SC'12), Salt Lake City, UT, USA, November 2012.
2. J. C. Bennett, H. Abbasi, P. Bremer, R. W. Grout, A. Gyulassy, T. Jin, S. Klasky, H. Kolla, M. Parashar, V. Pascucci, P. Pbay, D. Thompson, H. Yu, F. Zhang, J. Chen, "Combining In-Situ and In-Transit Processing to Enable Extreme-Scale Scientific Analysis", In ACM/IEEE International Conference for High Performance Computing, Networking, Storage, and Analysis (SC'12) , Salt Lake City, Utah, U.S.A., November, 2012.
3. T. Jin, F. Zhang, M. Parashar, S. Klasky, N. Podhorszki, H. Abbasi, "A Scalable Messaging System for Accelerating Discovery from Large Scale Scientific Simulations", 19th IEEE International Conference on High Performance Computing (HiPC'12) , Pune, India, December, 2012.
4. F. Zhang, C. Docan, M. Parashar, S. Klasky, N. Podhorszki and H. Abbasi, "Enabling In-situ Execution of Coupled Scientific Workflow on Multi-core Platform",26th IEEE International Parallel and Distributed Processing Symposium (IPDPS'12) , Shanghai, China, May 2012.
5. F. Zhang, C. Docan, M. Parashar and S. Kalasky, "Enabling Multi-Physics Coupled Simulations within the PGAS Programming Framework", 11th IEEE/ACM International Symposium On Cluster, Cloud and Grid Computing (CCGrid'11), Newport Beach, California, May 2011.
6. C. Docan, M. Parashar, J. Cummings and S. Klasky, "Moving the Code to the Data -- Dynamic Code Deployment using ActiveSpaces", 25th IEEE International Parallel and Distributed Processing Symposium (IPDPS'11), Anchorage, Alaska, May 2011.
7. C. Docan, F. Zhang, M. Parashar, and S. Klasky, "Coupling Scientific Fusion Simulations at Extreme Scale", 4th IEEE International Scalable Computing Challenge (SCALE 2011).

8. C. Docan, M. Parashar and S. Klasky. “Enabling High Speed Asynchronous Data Extraction and Transfer Using DART”. Concurrency and Computation: Practice and Experience, 22(9):1181-1204, 2010.
9. F. Zhang, C. Docan, M. Parashar and S. Kalasky, “DADS: A Dynamic and Adaptive Data Space for Interacting Parallel Applications”, 22nd International Conference on Parallel and Distributed Computing and Systems (PDCS'10), Marina del Rey, California, November 2010.
10. C. Docan, M. Parashar and S. Klasky, “DataSpaces: An Interaction and Coordination Framework for Coupled Simulation Workflows”, 19th ACM International Symposium on High Performance and Distributed Computing (HPDC'10), Chicago, Illinois, June 2010.
11. C. Docan, F. Zhang, M. Parashar, J. Cummings, N. Podhorszki and S. Klasky, “Experiments with Memory-to-Memory Coupling for End-to-End Fusion Simulation Workflows”, 10th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid'10), Melbourne, Australia, May 2010.
12. F. Zheng, H. Abbasi, C. Docan, J. Lofstead, S. Klasky, Q. Liu, M. Parashar, N. Podhorszki, K. Schwan and M. Wolf, “PreDatA - Preparatory Data Analytics on Peta-Scale Machines”, 24th IEEE International Parallel and Distributed Processing Symposium (IPDPS'10), Atlanta, Georgia, April 2010.
13. J. Cummings, A. Sim, A. Shoshani, J. Lofstead, K. Schwan, C. Docan, M. Parashar, S. Klasky, N. Podhorszki and R. Barreto, “EFFIS: an End-to-end Framework for Fusion Integrated Simulation.”, 18th Euromicro International Conference on Parallel, Distributed and Network-Based Computing (PDP'10), Pisa, Italy, February 2010.
14. N. Podhorszki, S. Klasky, Q. Liu, C. Docan, M. Parashar, H. Abbasi, J. F. Lofstead, K. Schwan, M. Wolf, F. Zheng and J. C. Cummings “Plasma Fusion Code Coupling using Scalable I/O Services and Scientific Workflows”, 4th Workshop on Workflows in Support of Large-Scale Science (WORKS'09), Portland, Oregon, November 2009.
15. C. Docan, M. Parashar and S. Klasky, “Enabling High Speed Asynchronous Data Extraction and Transfer Using DART”, Concurrency and Computation: Practice and Experience, 22(9):1181-1204, 2010.
16. C. Docan, S. Klasky, and M. Parashar. “Enabling High Speed Asynchronous Data Extraction and Transfer Using DART”, 17th ACM International Symposium on High Performance and Distributed Computing (HPDC'08), Boston, Massachussets, June 2008.
17. C. Docan, S. Klasky, and M. Parashar. “High Speed Asynchronous Data Transfers on the Cray XT3”, Technical Report TR-284, May 2007.
18. V. Bhat, M. Parashar, H. Liu, M. Khandekar, N. Kandasamy, S. Klasky, and S. Abdelwahed, “An Self-Managing Wide-Area Data Streaming Service”, Cluster Computing: The Journal of Networks, Software Tools, and Applications, Special Issue on Autonomic Computing, 10(7):365-383, 2007.
19. V. Bhat, M. Parashar, and S. Klasky, “Experiments with In-Transit Processing for Data Intensive Grid workflows”, 8th IEEE International Conference on Grid Computing, Texas, Austin, September, 2007.
20. V. Bhat, M. Parashar, M. Khandekar, N. Kandasamy, and S. Abdelwahed, “Enabling Self-Managing Applications using Model-based Online Control Strategies” 3rd IEEE International Conference on Autonomic Computing, Dublin, Ireland, June 2006.

21. L. Zhang and M. Parashar. Seine: A Dynamic Geometry-based Shared Space Interaction Framework for Parallel Scientific Applications. Concurrency and Computations: Practice and Experience. John Wiley and Sons. 2006.
22. L. Zhang and M. Parashar. Enabling Efficient and Flexible Coupling of Parallel Scientific Applications. In the Proceeding of the 20th IEEE International Parallel and Distributed Processing Symposium, Rhodes Island, Greece, 2006.
23. L. Zhang and M. Parashar, E. Gallicchio and R.M. Levy. Salsa: Scalable Asynchronous Replica Exchange for Parallel Molecular Dynamics Applications. In the proceedings of the 2006 International Conference on Parallel Processing. Columbus, Ohio, USA, August 14-18, 2006.
24. L. Zhang, M. Parashar, and Scott Klasky. Experiment with Wide Area Data Coupling Using the Seine Framework. In the proceeding of the IEEE International Conference on High Performance Computing. Bangalore, India. Dec. 2006.