Introducing OpenSHMEM

SHMEM for the PGAS Community

Barbara Chapman
Computer Science Department,
University of Houston
bchapman@uh.edu

Tony Curtis
Computer Science Department,
University of Houston
arcurtis@mail.uh.edu

Swaroop Pophale
Computer Science Department,
University of Houston
spophale@cs.uh.edu

Stephen Poole
Oak Ridge National Laboratory, and
Open Source Software Solutions
spoole@ornl.gov

Jeff Kuehn
Oak Ridge National Laboratory
kuehn@ornl.gov

Chuck Koelbel
Oak Ridge National Laboratory
koelbelch@ornl.gov

Lauren Smith
Open Source Software Solutions
lauren.l.smith@ugov.gov

Abstract

The OpenSHMEM community would like to announce a new effort to standardize SHMEM, a communications library that uses one-sided communication and utilizes a partitioned global address space.

OpenSHMEM is an effort to bring together a variety of SHMEM and SHMEM-like implementations into an open standard using a community-driven model. By creating an open-source specification and reference implementation of OpenSHMEM, there will be a wider availability of a PGAS library model on current and future architectures. In addition, the availability of an OpenSHMEM model will enable the development of performance and validation tools.

We propose an OpenSHMEM specification to help tie together a number of divergent implementations of SHMEM that are currently available.

To support an existing and growing user community, we will develop the OpenSHMEM web presence, including a community wiki and training material, and face-to-face interaction, including workshops and conference participation.

Categories and Subject Descriptors D.3.2 [*Language Classifications*]: Concurrent, distributed, and parallel languages; D.3.4 [*Processors*]: Run-time environments; D.2.4 [*Software/Program Verification*]: Validation

General Terms Performance, Languages, Standardization, Verification

PGAS '10 October 12-15, 2010, New York, New York USA.

Keywords ACM proceedings, SHMEM, OpenSHMEM, PGAS

1. Introduction

SHMEM is a communications library that is used for Partitioned Global Address Space (PGAS) [11] style programming. The SHMEM communications library was originally developed as a proprietary application interface by Cray for their T3D systems [6]. Since then different vendors have come up with variations of the SHMEM library implementation to match their individual requirements. These implementations have over the years diverged because of the lack of a standard specification. We aim to form a community that will unify all SHMEM library development effort and lead to the formulation of a standardized library to be called OpenSHMEM. OpenSHMEM will be a community-driven, open process.

1.1 SHMEM Features

In contrast to other language extensions such as Co-Array Fortran [10] and Unified Parallel C (UPC) [14] which fall under the PGAS programming model, SHMEM is a library-based approach that allows an application programmer to start using an efficient one-sided PGAS model incrementally in a known programming language.

The key features of SHMEM include one-sided point-to-point and collective communication, a shared memory view, and atomic operations that operate on globally visible, or "symmetric" variables in the program. Such symmetric variables can be either per-process globals (on the heap) or allocated by SHMEM (from a special symmetric heap) at runtime. These features allow the use of remote direct memory access (rDMA) when supported by the network.

Modern commodity networking technology such as Infiniband enables the efficient use of this PGAS library on a wide range of platforms. Runtime systems that have been developed for PGAS languages can easily be used by SHMEM library implementations.

1.2 Comparison with two-sided models

Traditional message-passing in environments like MPI [1] has been two-sided (although MPI 2 [2] introduced one-sided put and get calls that operate on data "windows"). Communication occurs with matching send and receive calls on both of the processors involved. By contrast, one-sided communication does not have a matching call pair: *e.g.* in a put, only the sender process participates; the receiver is unaware of the data arriving in its memory. This can lead to a simplified expression of parallelism and allows easier overlap of computation and communication.

1.3 Comparison with other PGAS libraries

The best known PGAS programming interfaces are Co-Array Fortran, UPC and Global Arrays. All of these provide global use of memory, target distributed memory systems and rely on one-sided communication. There also exist two commonly used communications libraries for the PGAS model: GASNet [3] and ARMCI [9]. These are targeted more at the developer level than the application programmer, providing lower-level and finer-grained control of data distribution and communication. The SHMEM library can sit above GASNet or ARMCI, and is a more explicit programming method intended for application programmers because of its ease of use. The Chapel [7] language, for example, uses GASNet for communication and data management in its runtime. SHMEM, on the other hand, is intended more for programmer use in applications that exploit the PGAS model, but this does not preclude use of SHMEM as a runtime implementation environment.

1.4 SHMEM Program Structure

SHMEM programs start with an initialization call, and then employ one-sided communication and synchronization routines to distribute data amongst processors (called processing elements, or PEs in SHMEM) participating in the program. PEs are numbered 0, 1, 2, ... n - 1. As with MPI [1], SHMEM programs follow the Single Program, Multiple Data (SPMD) model of operation. The code can query its runtime environment to discover which PE is running, and make decisions about what to do accordingly (*e.g.* PE 0 broadcasts image processing data to all other PEs).

Synchronization employs fences and barriers to ensure PEs have all finished dependent work. Point-to-point synchronizations are also possible through "wait" routines that test remote updates of global, symmetric variables.

1.5 History of SHMEM

SHMEM was developed originally by Cray for the Cray T3D and subsequently the T3E models. These systems typically consisted of a memory subsystem with a logically shared address space over physically distributed memories, a memory interconnect network, a set of processing elements, a set of input-output gateways, and a host subsystem. The systems were designed to support latency hiding mechanisms [8] such as prefetch queues, remote stores and the Block Transfer Engine (BLT). The prefetch queues and remote stores enabled fast transfer of small sized data, and the BLT could hide latency while transferring thousands of words.

Since then, SHMEM has been ported to a variety of platforms and has helped encourage the development of parallel programming, including PGAS languages. Some existing SHMEM implementations include Cray [5], [4], Quadrics' QsNet [12], SGI's versions for their IRIX and Altix platforms, HP [13], IBM and the SiCortex platforms [16], and the more portable but older GPSHMEM.

2. Towards an OpenSHMEM specification

2.1 Open Source Software Solutions

An umbrella organization called Open Source Software Solutions (OSSS) has been formed as a "home" for OpenSHMEM and other open source projects of interest to the U.S. government.

2.2 The current situation

There are several SHMEM implementations available at present. These offer slightly different API syntax and subtly divergent behaviors, making portability an issue for codes written using different implementations. Moreover, performance portability suffers as different implementors have focused on different aspects of the software. It is therefore hard to maintain a single, high-performing version of a code.

One simple example of this divergence is the initialization routine. In some versions it is called `start_pes` and takes a parameter (the number of PEs), in others it is called `shmem_init` and takes no parameters (the number of PEs is taken from the invoking environment).

Some, but not all, implementations provide an extra non-blocking version of the "put" call, `shmem_put_nb`, in which the reusability of memory is not guaranteed upon return from the put. With the base `shmem_put` call, local memory may be reused after return (although remote delivery can not be assumed in either case). Applications that use these extra routines are therefore Anot directly portable.

2.3 Extending the current specification

We propose an OpenSHMEM specification to help tie together a number of divergent implementations of SHMEM that are currently available. This specification will be termed version 1.0 and will be based on the SGI version of SHMEM [15], in terms of both the API and behavior. Also, we will produce a reference API implementation with the required functionality.

Later, a new specification, version 2.0, will be produced with input and guidance from the OpenSHMEM community, and there will be a reference implementation of the new API. The purpose of that implementation will be to enhance needed functionality (beyond what is supported in version 1.0), and to resolve any consistency or performance issues that are discovered in practice.

2.4 Validation and Verification

To make the validation and verification of any OpenSHMEM implementation easy, we are developing a V&V suite that will merge donated suites and include appropriate tests to exercise all components of the library. In addition, a performance suite of tests will also be made available to ensure that the performance of OpenSHMEM constructs within implementations is acceptable.

2.5 OpenSHMEM community

To support an existing and growing user community, we will develop an OpenSHMEM web presence. This presence will include documentation such as specifications, discussion documents, cheatsheets, a community-driven FAQ and tutorial material.

There will also be a wiki for development of 3rd. party material for and by the community. Face-to-face interaction, including discussion meetings, workshops and conference participation, will be organized as needed. Interested developers and users are invited to participate in this community-driven process. A mailing list already exists and is used for discussions about OpenSHMEM. To subscribe, visit

`https://email.ornl.gov/mailman/listinfo/openshmem`

3. Conclusions

Currently there is a number of slightly different SHMEM implementations available. To allow the community of SHMEM programmers to move forward and develop programs that are portable requires consolidation of the different APIs and behaviors of these implementations. OpenSHMEM is a proposal to achieve exactly that.

Making the OpenSHMEM process **open** to the SHMEM community will ensure that the project grows in the right direction, and that materials produced by the project accurately reflect actual usage of OpenSHMEM in the real world.

As High Performance Computing systems continue to grow in size and capacity, the overheads imposed by synchronization grow too. Libraries like SHMEM that provide one-sided communication and defer synchronization become ever more attractive for such systems. SHMEM can be used either directly for application development, or as a language implementation layer.

4. Acknowledgements

The OpenSHMEM project is supported by the U.S. Department of Energy, and Oak Ridge National Laboratory.

SHMEM is a trademark of SGI, Inc.

References

[1] A Message Passing Interface standard. *International Journal of Supercomputer*, 8(3/4):159–416, June 1994.

[2] Message Passing Interface Forum: MPI2: A message passing interface standard. *High Performance Computing Applications*, 12(1-2):1–299, 1998.

[3] D. Bonachea. GASNet specification, v1.1. Technical report, Computer Science Department, University of California, Berkelely, 2002.

[4] Cray, Inc. *Man Page Collection (Unicos LC): Shared Memory Access SHMEM*, .

[5] Cray, Inc. *Man Page Collection (Unicos MP): Shared Memory Access SHMEM*, .

[6] Cray, Inc. *CRAY T3D System Architecture Overview Manual*, .

[7] Cray, Inc. Chapel language specification, 0.796. Technical report, Cray, Inc, 2010.

[8] Cray Research, Inc. Cray T3D technical summary. Technical report, Cray Research, Inc, 1993.

[9] J. Nieplocha, V. Tipparaju, M. Krishnan, and D. Panda. High performance remote memory access comunications: The ARMCI approach. *International Journal of High Performance Computing and Applications*, 20(2):233–253, 2006.

[10] R. Numrich and J. Reid. Co-Array Fortran for parallel programming. *Fortran Forum*, 17(2), 1998.

[11] PGAS Forum. PGAS forum, `http://www.pgas.org/`.

[12] Quadrics Ltd. *User Manual - Running Parallel Programs with RMS and QsNet*.

[13] Quadrics Supercomputers World Ltd. *hp AlphaServer SC User Guide*.

[14] Sébastien Chauvin and Proshanta Saha and François Cantonnet and Smita Annareddy and Tarek El-Ghazawi. *UPC Manual*.

[15] SGI, Inc. *SHMEM API Man Pages*. `http://docs.sgi.com/`.

[16] SiCortex. *SiCortex System Programming Guide*.