http://www.openshmem.org/

OpenSHMEM: A Communications Library for Performance and Resilience

Delafrouz Mirfendereski[1], Md Abdullah Shahneous Bari[1], Wenbin Lu[1], Tony Curtis[1] and Dr. Barbara Chapman[1,2]
[1]Exascale Programming Model Laboratory, Stony Brook University (https://you.stonybrook.edu/exascallab/)
[2]Brookhaven National Lab

Motivation

- Maintaining performance at scale for applications with irregular and dynamic communication patterns (e.g. irregular graph, multi dimensional fast Fourier transform) is difficult using traditional distributed programming models such as MPI
- The PGAS programming models present an alternative approach to improve programmability
- The lower overhead in one-sided communication and the global view of data in PGAS models have the potential to increase the performance at scale
- OpenSHMEM is a standardized PGAS library

About OpenSHMEM

- Partitioned Global Address Space (PGAS) Parallel Programming Library
- SPMD (Single Program Multiple Data) programming model
- Provides global view of memory in a distributed programming environment
- One sided communication compared to traditional two sided communication (e.g. MPI)
- Work is divided into multiple processes known as processing elements (PEs)
- OpenSHMEM routines supply remote data transfer, work-shared broadcast and reduction, barrier synchronization, and atomic memory operations across PEs
- For communication OpenSHMEM uses RDMA (Remote Direct Memory Access) **1-sided** put/get instead of matched pairs (e.g. MPI send/recv)
- Less communication overhead
- OpenSHMEM provides specification for a standardized API and an execution model
- Exascale Programming Models Laboratory at Stony Brook University is responsible for the reference implementation
- Many implementations available for different hardware platforms

Interested? Contact

Tony Curtis (Project Lead)
Exascale Programming Models Laboratory
Email: *anthony.curtis@stonybrook.edu*

Eureka!

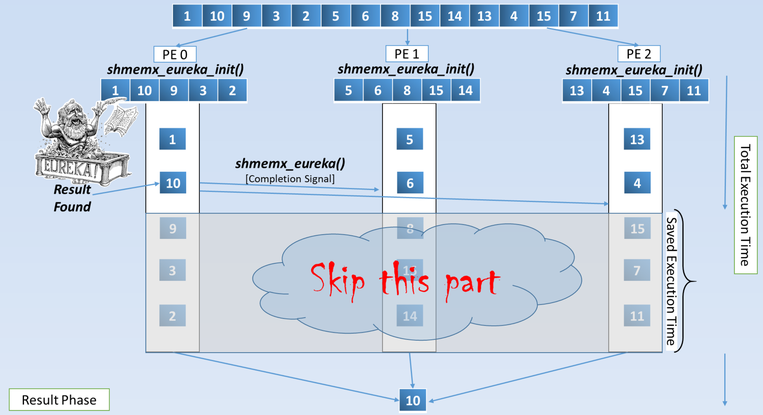

Figure 1. Eureka in action, workflow of a simple parallel search algorithm, searching for 10 in an array using 3 PEs [using Eureka Framework]

Motivation

- A wide range of problems, such as combinatorial optimization, constraint satisfaction, image matching, genetic sequence similarity, iterative optimization methods, can be reduced to tree or graph search problems
- Algorithms for these problems employ a common pattern, a eureka event
  - A point in the program which announces that a result has been found
- Reduces computation time by avoiding further exploration of a solution
- How can we make the eureka method available efficiently in a parallel programming model?

Eureka Framework

- Provides API to handle eureka events efficiently
- Allows a worker that encountered eureka event to send early completion signal to other workers
- Improves performance by avoiding unnecessary computation
- Implemented as an extension to the OpenSHMEM library
- API description:
  - ***shmemx_eureka_init()***: Initializes a eureka region
  - ***shmemx_eureka()***: Signals a eureka event, which in turn sends completion signals to other processing elements (PEs)

Sponsors

Los Alamos NATIONAL LABORATORY EST. 1943 · U.S. DEPARTMENT OF ENERGY · NSF

Collaborators

RICE Unconventional Wisdom · THE UNIVERSITY OF TENNESSEE KNOXVILLE · RUTGERS THE STATE UNIVERSITY OF NEW JERSEY

Implementation over UCX & PMIx

- Unified Communication X (UCX) is a new communications library for high performance computing and big data with backing from major vendors
  - http://www.openucx.org/
- Process Management Interface for Exascale (PMIx) is a launch- and run-time management layer for scalable and resilient high performance computing programs
  - https://pmix.github.io/pmix/

Fault Tolerance

- Mean Time Between Failures decreasing to a few hours on petascale machines
- Looking for ways and techniques to detect failures, propagate the information to parts involved, recover from failures or continue the normal computations and communications
- Cross-Layer Application-Aware Resilience at Extreme Scale (CAARES) research plan:
  - Collaborating with University of Tennessee Knoxville and Rutgers University
  - Providing the basic support for OpenSHMEM and MPI
  - Leveraging User Level Failure Mitigation (ULFM) and other resilience libraries
  - Application-aware and collaborative resilience frameworks
  - Enhance the resilience techniques using compilers
- Using a compiler to analyze the computation and communication patterns
  - Inserting and suggesting automated resilience techniques

For more information

http://www.openshmem.org/