Symmetric Memory Partitions in OpenSHMEM

Naveen Ravichandrasekaran
nravi@cray.com

Cray Inc.

Intel PGAS WG presentation

08-June-2017

Agenda

- **Introduction and Motivation**
- Overview of OpenSHMEM Memory Model
- Proposing Symmetric Memory Partition
- Cray SHMEM – Introduction and Implementation
- Performance Analysis
- Conclusion and Future Work

Introduction

- **Emerging systems feature different kinds of memory**
  - Different performance and optimization characteristics
  - Example: Intel KNL
    - Off-package DDR and On-package MCDRAM
    - MCDRAM bandwidth is about 4X higher compared to DDR
    - Size of MCDRAM limited to 16GiB compared to 384GiB in DDR
  - Access different kinds of memory
    - Vendor specific programming approach
      - Example – Memkind, CUDA
    - Low-level programming approach
- **Next generation systems are expected to support complicated multi-tiered memory hierarchies**
  - Major challenge – Application portability
- **Symmetric Memory Partition in OpenSHMEM**
  - Define a portable interface for symmetric heap placements on tiered memory systems

Agenda

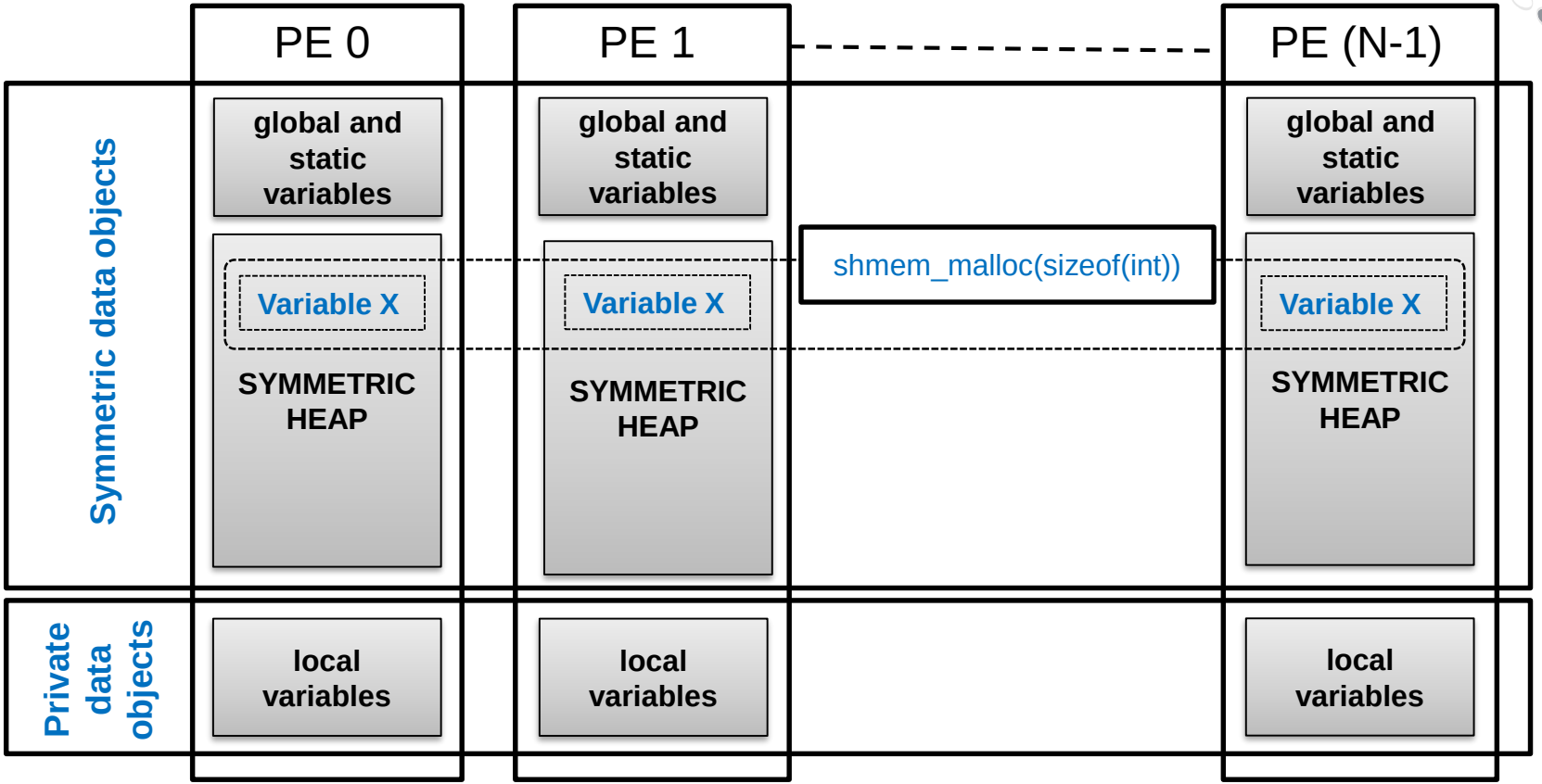
- Introduction and Motivation
- Overview of OpenSHMEM Memory Model
- Proposing Symmetric Memory Partition
- Cray SHMEM – Introduction and Implementation
- Performance Analysis
- Conclusion and Future Work

OpenSHMEM - Introduction

- **What is OpenSHMEM?**
  - Partitioned Global Address Space (PGAS) library interface **specification**
  - Aims to provide standard API for SHMEM libraries
    - http://openshmem.org/site/

- **What is Cray SHMEM?**
  - Cray SHMEM is a SHMEM library implementation from Cray Inc.
  - Cray SHMEM is OpenSHMEM standards-1.3 compliant

OpenSHMEM Memory Model



Symmetric Data Object & Symmetric Heap

- **Symmetric data objects**
  - Remotely accessible data objects
  - Same variable name, data type and size across all accessible PEs
  - Variables that are symmetric in OpenSHMEM:
    - Global and Static
    - Allocated using *shmem_malloc()* & *shmem_align()*
- **Symmetric Heap (SHEAP)**
  - Special memory region for dynamically allocating symmetric data objects
  - Standard allows creation of one SHEAP per PE
  - Created during program execution on an implementation determined region
  - Users control only the SHEAP size using SHMEM_SYMMETRIC_SIZE env variable

Agenda

- Introduction and Motivation
- Overview of OpenSHMEM Memory Model
- **Proposing Symmetric Memory Partition**
- Cray SHMEM – introduction and implementation
- Performance analysis
- Conclusion and Future Work

Proposed OpenSHMEM Memory Model

- **Symmetric Heap (SHEAP)**
  - Special memory regions for allocating symmetric data objects
  - Create during program execution
    - Implementation determined memory region, or
    - Multiple user-determined memory regions
- **Symmetric Memory Partitions**
  - User-determined memory regions
  - Identified using a label called *Partition ID*
  - Characteristics of each partition is defined using set a memory traits
- **Symmetric data objects**
  - Remotely accessible data objects
  - Same name, data type, size and *Partition ID* across all accessible PEs

SHEAP vs Symmetric Memory Partition

- **Symmetric Memory Partition just determines the memory region on which SHEAP can be created**
  - Characteristics defined using set of traits
- **In current proposal – one SHEAP per partition**
  - Looks like directly defining the characteristics of the SHEAP
- **WIP – determine need for integrating Memory Spaces with Memory Partitions –** explained later
  - http://nic.uoregon.edu/pgas14/papers/pgas14_submission_20.pdf

Defining Partition Characteristics

- **Similar to existing environment variable based approach**
- **SHMEM_SYMMETRIC_SIZE**
  - Only the size property of the SHEAP is defined by the users
- **SHMEM_SYMMETRIC_PARTITION**
  - One or more partitions defined using this env variable
  - Allow users to define partition characteristics with different traits
  - At present, each partition can take a maximum of four traits
  - Available traits:
    - SHEAP Size - Required value
    - Page Size - Optional with default documented by implementation
    - Memory kind - Optional with default documented by implementation
    - Memory Policy - Required if Memory Kind is defined by the user

SHMEM_SYMMETRIC_PARTITION

SHMEM_SYMMETRIC_PARTITION<**ID**>=**SIZE**=<size>[:**PGSIZE**=<pgsize>][:**KIND**=<kind>:**POLICY**=<policy>]

```
SHMEM_SYMMETRIC_PARTITION1=size=2G:kind=NORMALMEM
SHMEM_SYMMETRIC_PARTITION23=size=2G:kind=FASTMEM:policy=MANDATORY
SHMEM_SYMMETRIC_PARTITION2=size=1G:kind=F:policy=PREFERRED
SHMEM_SYMMETRIC_PARTITION15=size=2G:kind=N:policy=P
```

- **ID – user specified part of env variable to represent *Partition ID***
  - A maximum of SHMEM_MAX_PARTITIONS defined
  - Any non-zero positive integer between 1 and SHMEM_MAX_PARTITION_ID
- **SIZE –** Number of bytes to allocate for symmetric heap
- **PGSIZE –** Number of bytes to specify the size of the page used

SHMEM_SYMMETRIC_PARTITION

SHMEM_SYMMETRIC_PARTITION<**ID**>=**SIZE**=<size>[:**PGSIZE**=<pgsize>][:**KIND**=<kind>:**POLICY**=<policy>]

```
SHMEM_SYMMETRIC_PARTITION1=size=2G:kind=NORMALMEM
SHMEM_SYMMETRIC_PARTITION23=size=2G:kind=FASTMEM:policy=MANDATORY
SHMEM_SYMMETRIC_PARTITION2=size=1G:kind=F:policy=PREFERRED
SHMEM_SYMMETRIC_PARTITION15=size=2G:kind=N:policy=P
```

- **KIND –** String constant to specify implementation supported memory kinds
  - NORMALMEM – primary memory kind of a node(Example: DDR)
  - FASTMEM – example: MCDRAM on KNL
  - SYSDEFAULT – system defined memory kind(based on node configuration)
- **POLICY –** String constant to specify memory allocation policy
  - MANDATORY, PREFERRED, INTERLEAVED and SYSDEFAULT
- **SYSDEFAULT is not the implementation specific default value**

Memory Management Routines

- **Existing memory management routines**

```
void *shmem_malloc(size_t size);                          // memory allocation
void shmem_free(void *ptr);                               // memory deallocation
void shmem_align(size_t alignment, size_t size);          // allocation based on specific alignment
void shmem_realloc(void *ptr, size_t size);               // reallocate memory
```

- **Additional new memory management routines**

```
void *shmem_partition_malloc(size_t size, int partition_id);
void *shmem_partition_align(size_t alignment, size_t size, int partition_id);
```

  - similar to existing routines
  - allocated from specific partition identified using ***partition_id***
- **shmem_realloc() – reallocate on the same partition**
- **shmem_free() – release memory irrespective of the partition**

Partition ID - 1

- Partition ID – 1 has special meaning for maintaining backward compatibility
- Using SHMEM_SYMMETRIC_SIZE & SHMEM_SYMMETRIC_PARTITION1 is fatal
- Any call to shmem_malloc() and shmem_align() defaults to partition – 1

Agenda

- Introduction and Motivation
- Overview of OpenSHMEM Memory Model
- Proposing Symmetric Memory Partition
- **Cray SHMEM – introduction and implementation**
- Performance analysis
- Conclusion and Future Work

Cray SHMEM - Introduction

- **Closed source vendor-specific OpenSHMEM implementation**
- **Part of Message Passing Toolkit (MPT) software stack from Cray Inc.**
- **Use DMAPP (Distributed Shared Memory Application) library as a low-level communication layer**
- **OpenSHMEM specification version-1.3 compliant**
- **Apart from standard OpenSHMEM features, supports:**
  - Thread-hot extensions
  - **Support for multiple-symmetric heap for heterogeneous memory kinds**
  - Flexible PE subsets creation and management – OpenSHMEM Teams
  - Point-to-point put operation with signal,
  - Local shared-memory pointers, and
  - Non-blocking Atomics

Memory Partitions on KNL

- **Initial implementation for accessing DDR & MCDRAM on Intel KNL**
- **Different modes of configuration available to use MCDRAM**
  - Cache mode
    - Configured entirely as last-level cache
  - **Flat mode**
    - Available entirely as addressable memory
    - Required mode for using memory partitions features in Cray SHMEM
    - Mostly used with quad/flat configuration – not tried SNC2/SNC4
    - Used numactrl to bind MCDRAM during memory allocation
      - numactrl membind=1 or preferred=1
      - This controls the SYSDEFAULT trait in the partition
  - Hybrid Mode

SW Impact Analysis

- **Memory footprint involved in creating multiple symmetric heaps**
  - Increase memory footprint – somewhere in the software stack
  - Data structure to track multiple partitions
  - Memory registration for RMA and AMO operations
    - Registration done at DMAPP level

- **Impact on memory segment lookup during RMA and AMO**
  - Existing memory model queries
    - DATA segment and SHEAP segment
  - New memory model queries
    - DATA segment, SHEAP segment and multiple USER-HEAP segments

Agenda

- Introduction and Motivation
- Overview of OpenSHMEM Memory Model
- Proposing Symmetric Memory Partition
- Cray SHMEM – introduction and implementation
- **Performance analysis**
- **Conclusion and Future Work**

Performance Analysis

- **Initial experimentation on a Cray XC system with Intel KNL processors**
  - Cray developed Aries interconnect
  - Cray SHMEM/7.5.0 series – bit old version
  - Configured as Quad/Flat mode
  - Used numactrl membind=1

- **Tests performed**
  - Modified OSU micro-benchmarks
    - Study the impact of memory segment lookup during RMA operations
  - 2D SHMEM Stencil Kernel
  - Suggestions for other suitable benchmarks?

Modified OSU Put micro-benchmark

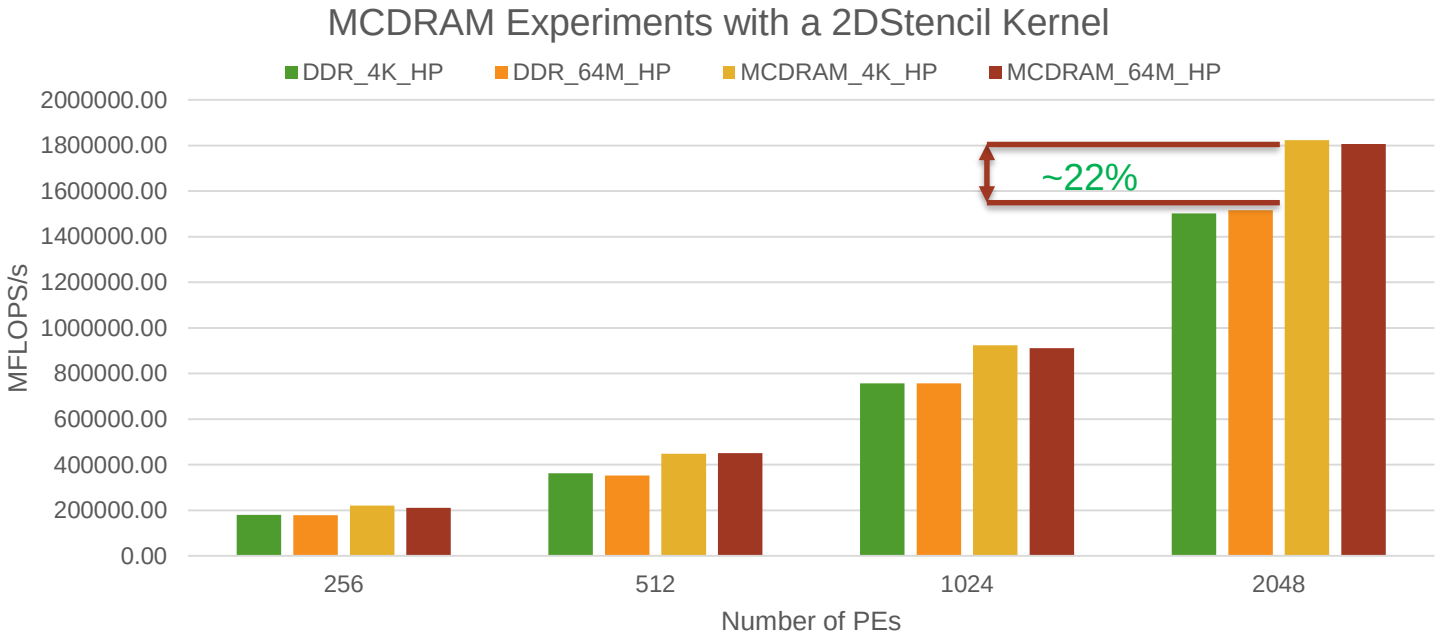
- Time bunch of puts followed by a shmem_quiet() operation
- Create multiple symmetric memory partitions
- Allocate multiple source and dest buffers on separate partitions
- Randomly select the partition for every iteration

- 8-partitions and 10,000 iterations
- No huge performance impact – ~2% performance variations

2D Stencil Kernel Performance Analysis

- **Can fit the entire test grid in the kernel inside a single memory kind**
  - No code change required
    - used default memory management routines with partition-1
    - Memory Kind=N(DDR)
    - Memory Kind=F(MCDRAM)
  - SMA_SYMMETRIC_PARTITION1=size=500M:kind=**N**:policy=M:pgsize=4K
  - SMA_SYMMETRIC_PARTITION1=size=500M:kind=N:policy=M:pgsize=64M
  - SMA_SYMMETRIC_PARTITION1=size=500M:kind=**F**:policy=M:pgsize=4K
  - SMA_SYMMETRIC_PARTITION1=size=500M:kind=F:policy=M:pgsize=64M
- **Also used different page sizes (4K, 64M) apart from memory kind**

Performance Analysis



- **Don't attribute the performance benefit to any SHMEM+MDCRAM BW**
- **The benefit is whatever the app is doing in MCDRAM algorithmically**
  - And SHMEM not restricting the users in enabling the use of MCDRAM

Agenda

- Introduction and Motivation
- Overview of OpenSHMEM Memory Model
- Proposing Symmetric Memory Partition
- Cray SHMEM – introduction and implementation
- Performance analysis
- **Conclusion and Future Work**

Conclusion and Future Work

- **Attempt to create a conversation for defining a portable interface for SHEAP placements on a tiered memory systems in OpenSHMEM**
  - Current direction involves env variable based approach
  - Other possible directions would include introducing new extensions to query and set new memory partitions

Future Work:

- **Analysis on using memory partitions with the spaces**
- **Possibilities for combining with Teams(PE – subset groups) based approach to remove symmetric work arrays (pWrk and pSync)**
- **Expand support to cover additions memory kinds**
  - persistent memory?
- **Expand memory traits for defining Memory partitions**

Acknowledgements

- **Proposal based on joint effort from Intel and Cray**
- **Cray Inc.**
  - Bob Cernohous, Dan Pou, Krishna Kandalla, ***Naveen Ravi,*** David Knaak and others
- **Intel Corp.**
  - Joseph Robichaux, James Dinan, Ulf Hanebutte and Others

- **Started as an effort to define a portable OpenSHMEM API for allocating and using different kinds of memory that is available on Intel KNL**

- **Proposed extensions are available as SHMEMX features in Cray SHMEM**

Thank you