Symmetric Memory Partitions in OpenSHMEM: A Case Study with Intel KNL

Naveen Namashivayam ¥, Bob Cernohous ¥, Krishna Kandalla ¥, Dan Pou ¥, Joseph Robichaux £, James Dinan £, and Mark Pagel ¥

¥ **Cray Inc.** £ **Intel Corp.**

OpenSHMEM Workshop 2017

07-August-2017

Legal Disclaimer

Information in this document is provided in connection with Cray Inc. products. No license, express or implied, to any intellectual property rights is granted by this document.

Cray Inc. may make changes to specifications and product descriptions at any time, without notice.

All products, dates and figures specified are preliminary based on current expectations, and are subject to change without notice.

Cray hardware and software products may contain design defects or errors known as errata, which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Cray uses codenames internally to identify products that are in development and not yet publically announced for release. Customers and other third parties are not authorized by Cray Inc. to use codenames in advertising, promotion or marketing and any use of Cray Inc. internal codenames is at the sole risk of the user.

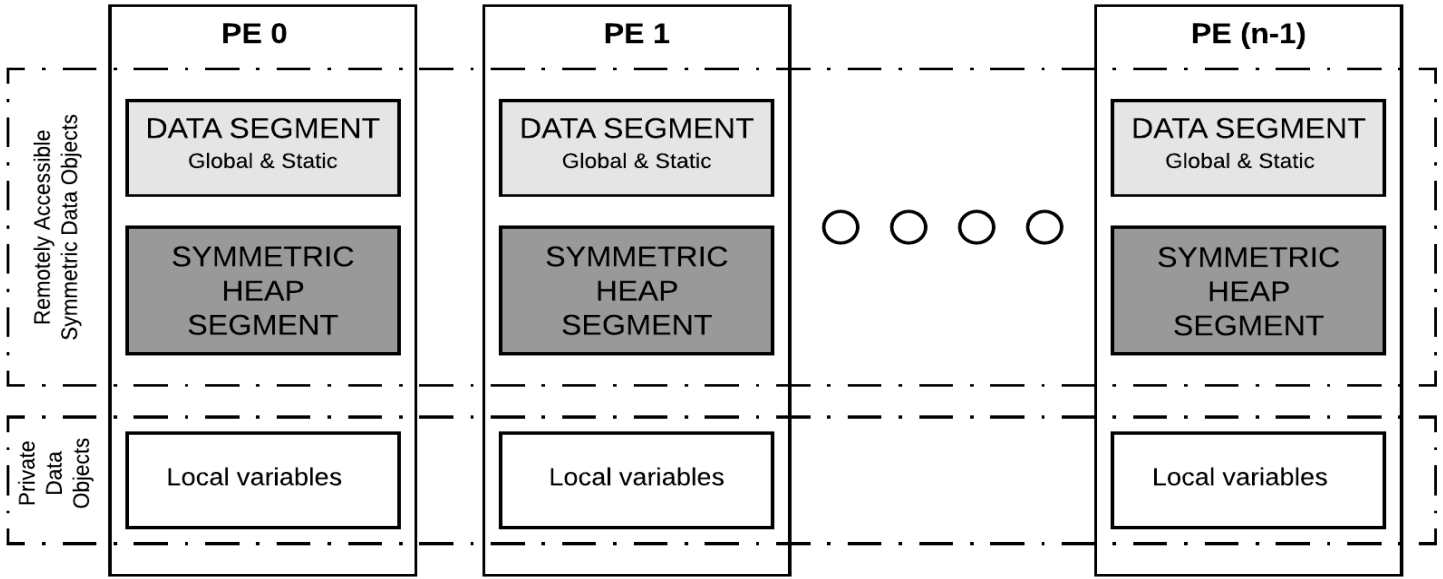
Performance tests and ratings are measured using specific systems and/or components and reflect the approximate performance of Cray Inc. products as measured by those tests. Any difference in system hardware or software design or configuration may affect actual performance.

The following are trademarks of Cray Inc. and are registered in the United States and other countries: CRAY and design, SONEXION, and URIKA. The following are trademarks of Cray Inc.: APPRENTICE2, CHAPEL, CLUSTER CONNECT, CRAYPAT, CRAYPORT, ECOPHLEX, LIBSCI, NODEKARE, THREADSTORM, REVEAL. The following system family marks, and associated model number marks, are trademarks of Cray Inc.: CS, CX, XC, XE, XK, XMT, and XT. The registered trademark LINUX is used pursuant to a sublicense from LMI, the exclusive licensee of Linus Torvalds, owner of the mark on a worldwide basis. Other trademarks used in this document are the property of their respective owners.

Introduction

- **Emerging systems features different kinds of memory**
  - With different performance and optimization characteristics
- **Identify and manage different kinds of memory**
  - Vendor specific programming approach
    - Example – Memkind, CUDA
  - Low-level programming approaches
- **Challenges**
  - Both programming approaches introduce issues on portability
  - Next generation systems will be complicated with multi-tiered mem hierarchies
- **Symmetric Memory Partition in OpenSHMEM**
  - Proposal to define a portable interface for symmetric heap placements on tiered memory systems

OpenSHMEM Memory Model



- **Symmetric Data Objects**
  - DATA SEGMENT – all Static and Global variables
  - SHEAP SEGMENT – variables with mem allocated using shmem_(malloc/align)

Intel KNL Architecture

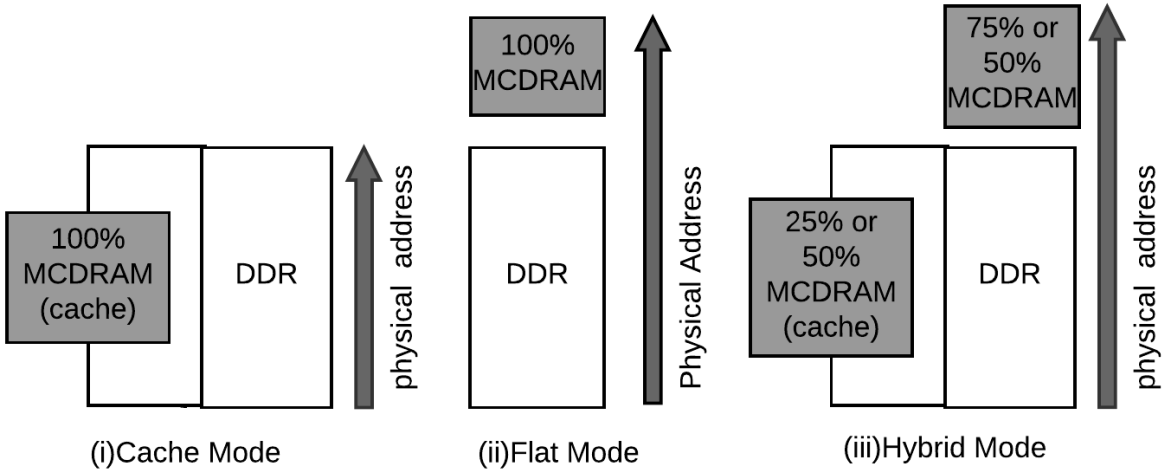
- **Xeon Phi - Intel's popular Many Integrated Core (MIC) architectures**
- **Second generation processors code named Intel KNL**
  - Supports at least 68 compute cores per chip with 4 threads per core
- **This paper – Intel KNL is used as an example use case for emerging tiered memory systems**
  - Traditional off-package DDR memory – ~384GB
  - High bandwidth on-package Multi-Channel DRAM (MCDRAM) – ~16GB
    - 4X increased bandwidth compared to DDR

Intel KNL – Clustering Modes

- **For improved message locality and cache coherence KNL supports different modes of cache operation**
- **Quadrant / Hemisphere mode**
  - Tiles(2 Cores) are divides into four quadrant
  - Single NUMA domain – “No breaks” in memory address space
- **Sub-NUMA clustering (SNC-2 / SNC-4) mode**
  - Again four quadrants
  - But, available as separate NUMA domains
  - Suitable for NUMA aware applications

Intel KNL – MCDRAM Configurations

- **Different modes based on MCDRAM configuration**
  - Cache Mode – MCDRAM as last-level cache
  - Flat Mode – MCDRAM as addressable memory
  - Hybrid Mode – MCDRAM as hybrid of last-level cache and addressable memory

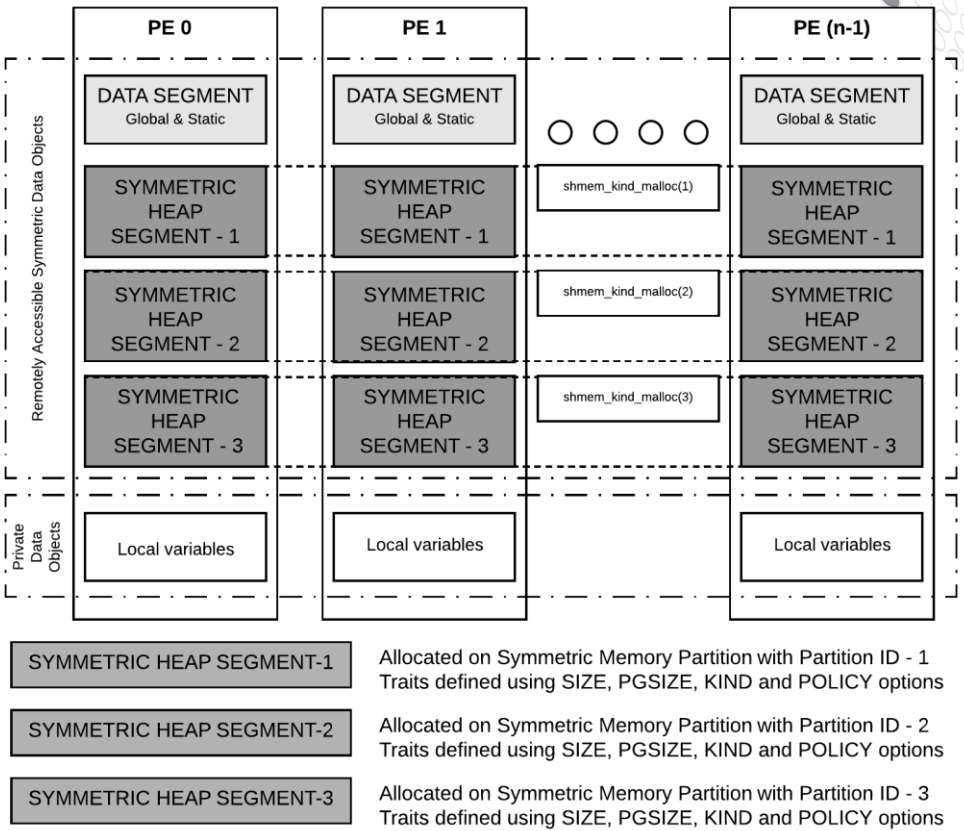


Using MCDRAM in OpenSHMEM Model

- **Cache Mode**
  - OpenSHMEM doesn't need to handle anything
- **Flat Mode**
  - Explicitly place data structures into HBM
  - Just fit the complete application memory in MCDRAM
    - Possible with current OpenSHMEM memory model
  - Identify specific bandwidth bound buffers and data access patterns – allocate bandwidth critical part on MCDRAM
    - Not possible with current OpenSHMEM memory model

Proposed Changes in OpenSHMEM Memory Model

- **Symmetric Heap (SHEAP)**
  - Created during program execution
  - Created on an implementation determined memory location or on multiple user-determined memory locations
- **Symmetric Memory Partitions**
  - User-determined memory locations
  - One SHEAP per partition
  - Multiple SHEAPS created with multiple separate partitions
  - Characteristics of each partition is identified using set a memory traits
  - Identified using Partition ID label
- **Symmetric Data Objects**
  - Remotely accessible data objects
  - Same name, data type, size and Partition ID across all accessible PEs



Symmetric Memory Partition Traits

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

- **Characteristics of each partition is set using list of memory traits**
- **Defined similar to existing environment variable based approach**
- **SMA_SYMMETRIC_SIZE**
  - Only the size property of the SHEAP can be 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

Memory Management Routines

```
void *shmem_kind_malloc(size_t size, int partition_id);
void *shmem_kind_align(size_t alignment, size_t size, int partition_id);
```

- Similar to existing memory management routines **shmem_malloc()** and **shmem_align()**
- Allocate from a specific symmetric heap created on a memory partition identified using ***Partition ID***
- **shmem_realloc()** – reallocation occurs on the same partition
- **shmem_free()** – release memory irrespective of the partition

SHMEM_SYMMETRIC_PARTITION1

- **SMA_SYMMETRIC_SIZE and SHMEM_SYMMETRIC_PARTITION1**
  - Partition with ID as 1 has special meaning
  - If SMA_SYMMETRIC_SIZE is not used, then any call to shmem_malloc() or shmem_align() will default to Partition-1
  - Using both env variables together causes fatal error
- **Provide backward compatibility to applications**

Memory Partitions in Cray SHMEM

- Available as SHMEMX prefixed features from version 7.4.0
- Library constants:

SHMEMX_MAX_PARTITION_ID	127	SHMEMX_MAX_PARTITIONS	7

- Memory identification performed using **numactrl**

KIND	NORMALMEM	DDR
	FASTMEM	MCDRAM
	SYSDEFAULT	Defined using numactl system calls

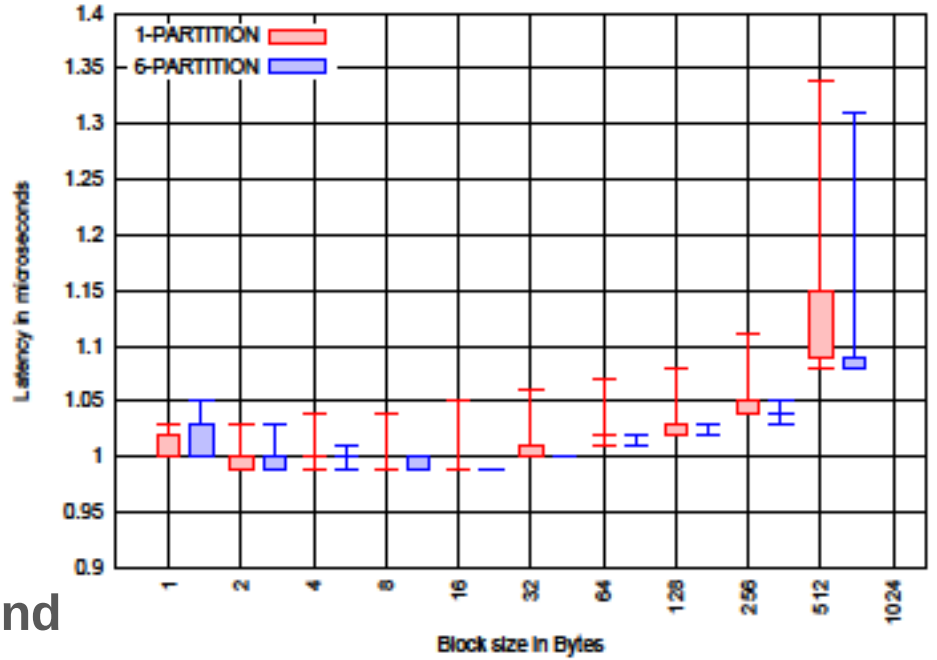
POLICY	MANDATORY	Abort if requested KIND is unavailable
	PREFERRED	Use other KINDS if request fails
	INTERLEAVED	Page allocation interleaved NUMA
	SYSDEFAULT	Defined using numactl system calls

Performance Regression Analysis: Segment Lookup

- **Modified OSU Put Micro-benchmarks**
- **Intel KNL based Cray XC systems – 2 PEs with 1 PE per Node**
- **Destination buffers**
  - Partitions are selected random on every iteration
  - No access patterns
  - 1 partition against 6 partitions:

  ~2-3% variations
  - 1 partition against 127 partitions:

  ~6-8% variations
- **SHMEMX_MAX_PARTITIONS in Cray SHMEM is 7**
- **Limit to 1 Partition per Memory Kind**

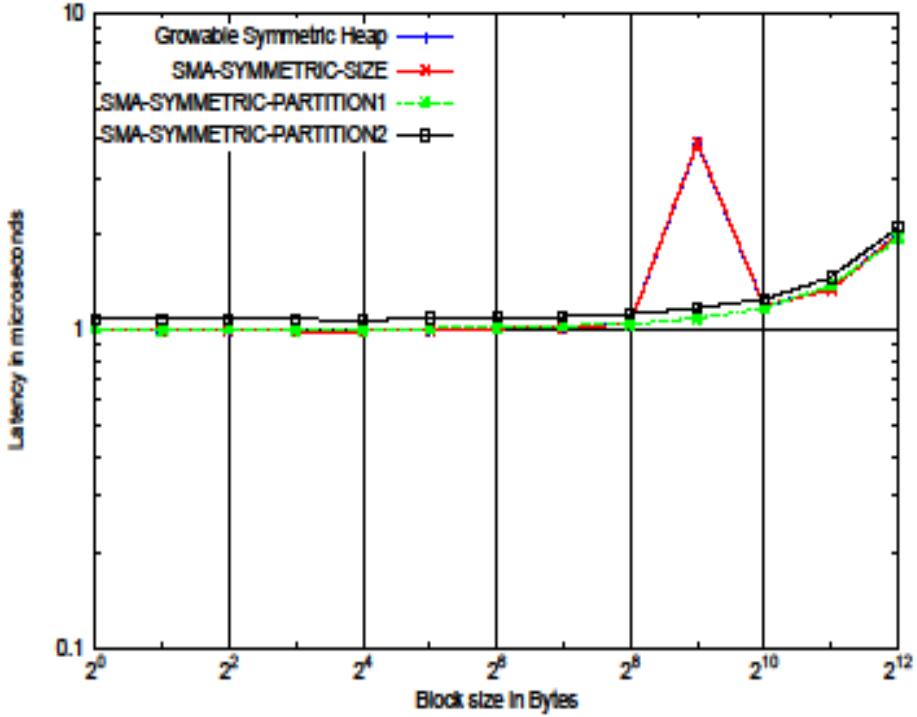


Performance Regression Analysis: SMA_SYMMETRIC_SIZE against PARTITIONS

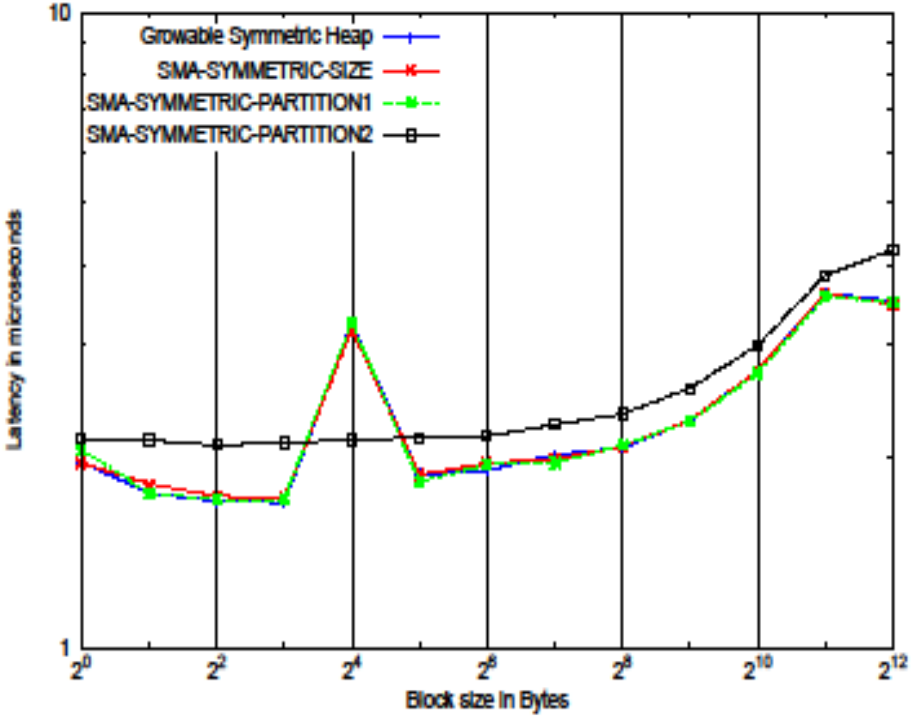
- **OSU Put Micro-benchmarks**
- **Using SHEAPs**
  - Created with SIZE determined from SMA_SYMMETRIC_SIZE
  - Created with SIZE from SMA_SYMMETRIC_PARTITION1
  - Created with SIZE from SMA_SYMMETRIC_PARTITION2
  - Cray SHMEM specific “Growable Symmetric Heap”
- **No restrictions on number of partitions on the same memory kind**
  - Test:1 – On BDW based Cray XC systems – Only use DDR memory
  - Test:2 – On KNL based Cray XC systems – Only use DDR memory
  - 2 PEs with 1 PE per Node

Performance Regression Analysis: SMA_SYMMETRIC_SIZE against PARTITIONS

BDW



KNL

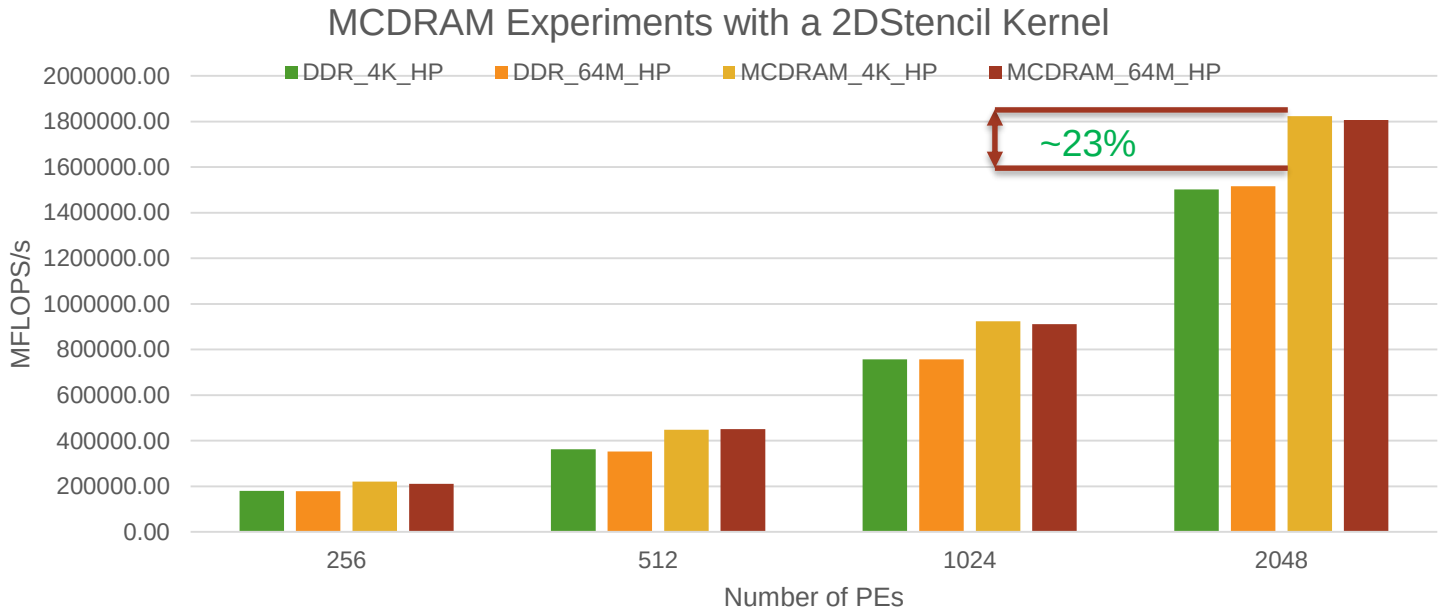


Performance Analysis

- **Experimented on KNL using a 2D-Stencil Kernel**
- **Clustering Mode – Quad Mode**
- **MCDRAM Configuration – Flat Mode**

- **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
  - 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

Performance Analysis



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

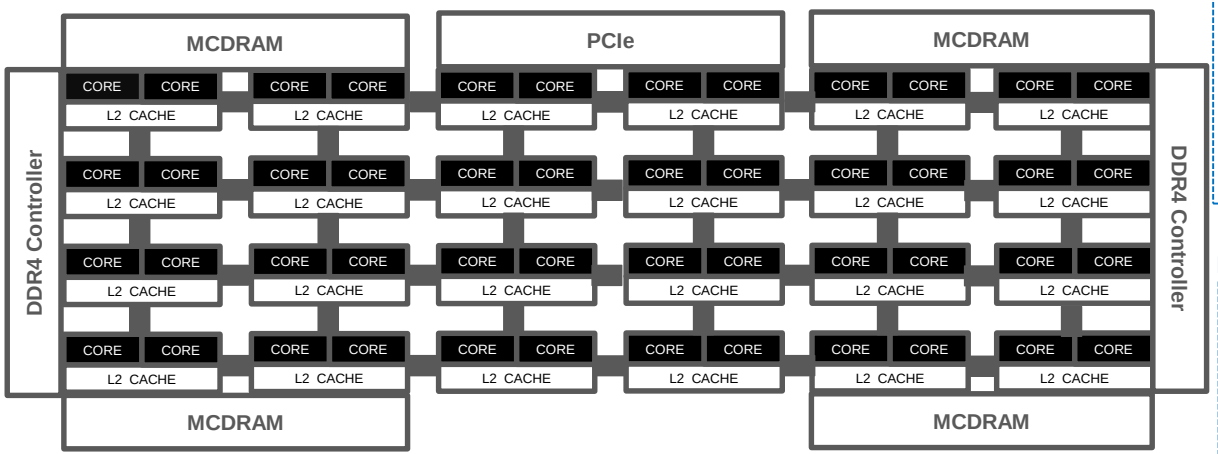
Existing Proposals and Future Work

- **Related Work**
  - OpenMP TR:5 – Memory Management Support
  - MPI_Alloc_mem() in Cray MPICH
- **Memory Spaces**
  - Aaron Welch's Team specific symmetric heap
  - http://nic.uoregon.edu/pgas14/papers/pgas14_submission_20.pdf
  - Another proposal that modifies the existing OpenSHMEM Memory Model
- **Future Work:**
  - Analysis on using Memory Partitions with Memory Spaces
  - Expand the Partition Traits (Example: Affinity – NEAR/FAR) to support truly heterogeneous memory
  - Support other additional types of memory like persistent memory
  - Test with bandwidth bound applications with requirement for more than one kind of memory

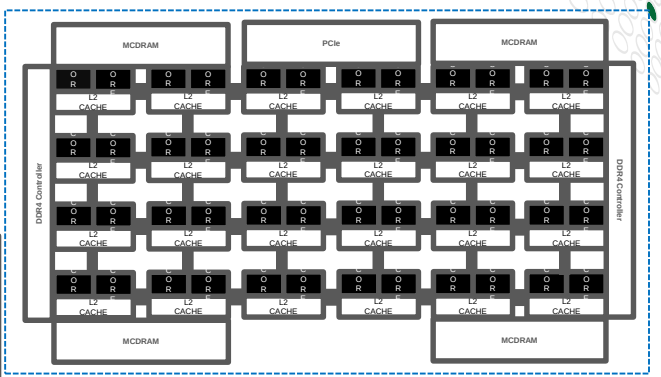
Conclusion

- **Proposal is not just defining multiple symmetric heaps**
  - At present, since we have one SHEAP per partition, it looks like we are defining the properties of the SHEAP directly
- **An attempt to create an OpenSHMEM feature to define a portable interface for SHEAP placements on tiered memory systems**
  - Partitions refer to an available memory resource on the system
  - Partitions have a defined characteristics with a set of user-defined traits
  - Current direction involves env. variable based approach

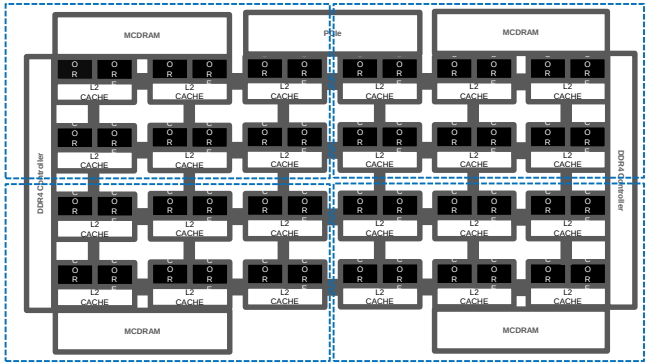
Thank you



NUMA domain 0



NUMA domain 0 NUMA domain 1



NUMA domain 2 NUMA domain 3