OpenSHMEM Heterogenous Memory Kinds Specification v0.7

September 1, 2016

Joseph Robichaux, James Dinan (Intel Corp.)

Bob Cernohous, Dan Pou, David Knaak (Cray)

Notes to reader:

This document is a joint effort of Intel and Cray to define an OpenSHMEM API that will provide a portable means of allocating and using heterogeneous memory kinds, and in particular, for using high bandwidth memory (HBM). The format somewhat follows that used by OpenSHMEM for proposing new APIs. Readers of this document are asked to give us feedback on functionality, ease of use, clarity, etc. After review and any necessary changes, an API will be proposed to OpenSHMEM.

Note the following OpenSHMEM API Specification statement:

> All OpenSHMEM extension APIs that are not part of this specification must be defined in the shmemx.h include file. These extensions shall use the shmemx_ prefix for all routine, variable, and constant names.

The following proposal does not include "x" but it is understood that any implementation of this proposal prior to its inclusion in an approved OpenSHMEM API must include "x" for the routine names and "X" for the SMA environment variables and for SHMEM constants.

Summary of the Changes

Introduction of runtime changes and an API to support allocation of heterogeneous memory kinds for the symmetric heap.

Benefits as Result of the Changes

The capability described enables the user to utilize different page sizes and coherent heterogeneous memory kinds that future systems will likely support. The features described in this proposal allow users to allocate different symmetric objects on different kinds of memory, each possibly having different performance and capacity characteristics. For example, an on-die memory may have much high bandwidth but have a much smaller size than conventional memory.

SW Impact Assessment

Change in internal runtime allocation of symmetric heap, possibly a data structure to track multiple, symmetric partitions, internal memory management, and possibly memory registration for RMA operations. It is expected that for many runtimes very few underlying changes will need to be made to communication transport calls.

An implementation does not need to support multiple memory kinds but must either map the API specifications to memory kinds that are supported or abort with informative messages about what is and is not supported.

HW Impact Assessment

No hardware changes are required. The API is meant to be portable across all systems for which there is an OpenSHMEM implementation. It is for the software to map the new API to existing hardware or to new hardware with heterogeneous memory kinds as it becomes available.

Detailed Description of the Change

Operation Name

shmem_kind_malloc

shmem_kind_align

Summary

New environment variables.

Environment variables are specified in OpenSHMEM API section 7. Currently, all OpenSHMEM environment variables begin with SMA_. We propose adding:

SMA_SYMMETRIC_PARTITION1=size=<size>[:kind=<kind>][:policy=<policy>]
[:pgsize=<page_size>]

SMA_SYMMETRIC_PARTITION2=size=<size>[:kind=<kind>][:policy=<policy>]
[:pgsize=<page_size>]

...

SMA_SYMMETRIC_PARTITION<n>=size=<size>[:kind=<kind>][:policy=<policy>]
[:pgsize=<page_size>]

One, two, or more symmetric memory partitions can be specified. A maximum of SHMEM_MAX_PARTITIONS can be defined, however the partition numbers may be any number between 1 and SHMEM_MAX_PARTITION_ID.

Two new constants, SHMEM_MAX_PARTITIONS, and SHMEM_MAX_PARTITION_ID are defined in shmem.h and shmem.fh. SHMEM_MAX_PARTITIONS is an implementation's maximum for the number of memory partitions that can be defined by the user. The implementation's value must be at least 7 so

that a portable program can assume at least that many partitions are possible.
SHMEM_MAX_PARTITION_ID is the maximum number that can be used for a partition id. This value should be at least 127.

The existing OpenSHMEM environment variable SMA_SYMMETRIC_SIZE=<size> is still supported with its current syntax and meaning. However, it is an error to specify both SMA_SYMMETRIC_SIZE and SMA_SYMMETRIC_PARTITION<n>.

Memory allocation from a memory partition for use by the OpenSHMEM program is performed by calling shmem_malloc() if SMA_SYMMETRIC_SIZE was used or by calling shmem_kind_malloc() with memory partition ID <n> as an argument (see below) if SMA_SYMMETRIC_PARTITION<n> was used. If one or more partitions were specified using SMA_SYMMETRIC_PARTITION<n> and shmem_malloc() is called, this memory allocation maps to memory partition 1.

Note: This would allow, for example, taking an existing OpenSHMEM program that calls shmem_malloc() and running it first with partition 1 defined to use Normalmem and then running it with partition 1 defined to use Fastmem. This would show the performance difference.

The size specifier is the only required value. All value specifications are position independent.

<size> is the size in bytes for the memory partition. The suffixes "K", "M", and "G" are supported:

K multiplies by 2**10 (kilobytes)
M multiplies by 2**20 (megabytes)
G multiplies by 2**30 (gigabytes)

<kind> can be one of these strings:

F[astmem] is a faster memory (by some measure) that may be on a node in addition to NORMALMEM
N[ormalmem] is the primary memory kind for a node.
S[ysdefault] is the system selected default memory. Policy must also be set to Sysdefault.

[More memory kinds may be added in the future but for now we will support these 2 kinds]

An implementation must support each of these kind names but any kind name can map to another memory kind if the intended memory kind cannot be supported by the implementation. The supported kinds and the mappings must be documented by the implementation. An implementation may support and document additional memory kinds.

If kind is specified, policy must also be specified.

<page_size> must be a string consisting of a number and size prefix. For example, on x86 systems supporting 4 Kilobyte, 2 Megabyte, and 1 Gigabyte pages the strings could be one of the following strings:

4K

2M
1G

An implementation must document which page sizes are available and for OpenSHMEM-specified page sizes that are not available, what multiple number of pages of a smaller page size that page size specifier will map to.

<policy> can be one of these strings:

M[anditory]
P[referred]
I[nterleaved]
S[ysdefault]

[More policies may be added in the future.]

If Mandatory is specified and that memory kind, or page size is not available or the requested amount of that memory kind is not available, the program will abort with an informative runtime error message (see advice to implementers below).

If Preferred is specified and that memory kind is not available or the requested amount of that memory kind is not available, the library will try to allocate that amount of memory using another memory kind. If that is also not possible, the program will abort with an informative message.

If Interleaved is specified, page allocations are interleaved across numa domains associated with the memory kind.

If Sysdefault is specified, then the system default memory policy will be used. Kind must also be set to Sysdefault.

If policy is specified, kind must also be specified.

New memory management routines.

We propose adding these memory partition management routines:

Synopsis

C/C++:

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

Fortran:
N/A

Parameters

IN	size	The size in bytes, of a block to be allocated from the symmetric heap
IN	partition_id	The ID of the partition that the allocated memory will be in
IN	alignment	Byte alignment of the block to be allocated from the symmetric heap

Constraints
N/A

Effect
shmem_kind_malloc/shmem_kind_align allocate objects on the symmetric heap with the memory characteristics associated with the predefined partition_id.

Return Values

The shmem_kind_malloc and shmem_kind_align routines return a pointer to the allocated space; otherwise, it returns a NULL pointer.

Notes:

Requirements for all PEs to call existing shmem memory management routines are the same for the shmem_kind routines.

In a quality implementation the shmem_free and shmem_realloc should also handle symmetric memory partitions correctly. It is assumed that when shmem_realloc is called the new memory returned will be in the same symmetric memory partition associated with the input address.

Memory information report.

Cray had proposed an SMA_MEMINFO_REPORT to serve a similar purpose to Cray SHMEM's SHMEM_MEMINFO_REPORT. Such a requirement doesn't fit well into an API specification but we need to somehow make clear that an implementation needs to do something to let the user know what memory is available or not available. We will leave it to the implementation to decide what constitutes "an informative message" and maybe add:

> Advice to implementers: A quality implementation will provide whatever information is necessary for the user to understand what memory is available, what memory has been allocated, and why a memory request can't be honored.

EXAMPLE:

```
export SMA_SYMMETRIC_PARTITION1=size=2G:kind=S:pgsize=2M:policy=S
export SMA_SYMMETRIC_PARTITION2=kind=Fastmem:policy=Preferred:size=4G:pgsize=8M

main(...)
{
  ...
```

```
  shmem_init();
  ...
  /* array_A is not accessed often and will use 2M huge pages with the system default memory and
policy */
  int *array_A = (int*)shmem_kind_malloc(array_A_size*sizeof(int), 1)

  /* fast access to array_B is most important to this program,
   * we want it in the Fastmem memory partition using 8M huge pages*/
  long *array_B = (long*)shmem_kind_malloc(array_B_size*sizeof(long), 2)
  ...

  shmem_free(array_A);
  shmem_free(array_B);
  shmem_finalize();
  return 0;
}
```

Interface to memory topology

Both Cray and Intel will depend on memkind for parsing memory topology.