recognized that implementing this can be hard and currently discussions are ongoing with the MCA Hypervisor working group to understand a potential relationship for supporting coherent shared memory. Remote memory is provided for systems that require the use of explicit CPU, DMA, or other non-CPU mechanisms to move data between memory subsystems, or which do not support hardware managed cache coherency. The MRAPI specification allows for implementations to support only those types of MRAPI memory that are feasible for a given system, but the implementation must provide all API entry points and indicate via error reporting that a given request cannot be satisfied.

3.5.1 MRAPI Shared Memory

The functionality provided by the MRAPI shared memory API is similar to that of POSIX® shared memory, but MRAPI extends the functionality beyond the scope of a single operating system. It provides the ability to manage the access to physically coherent shared memory between heterogeneous threads of execution that may be on different operating systems and different types of cores.

3.5.2 MRAPI Remote Memory

Modern heterogeneous multicore systems often contain multiple memory spaces, where data is moved between memory spaces via non-CPU mechanisms such as direct memory access (DMA). One example is the Cell Broadband Engine processor: the Power Processor Element (PPE) is a standard PowerPC® core connected to main memory, but the processor also contains 8 synergistic processor elements (SPEs) which are each equipped with a small local store. Data must be copied to and from SPE local store via explicit DMA operations.

Remote memory might be implemented in many different ways, depending on the underlying hardware. Sometimes actual copying (i.e. read/write operations) are needed, sometimes just some software initiated cache operations are needed (invalidate/flush). However, the very purpose of an API should be to hide these differences in order to enable portable and hardware independent software. So in order to access data, some API call should be made that *might* boil down to either a "read", and "sync" or some combination (depending on the underlying hardware). The thing is that the user shouldn't really have to care. The software layer that constitutes the API should make sure that the necessary operations are performed, depending on the hardware.

From the point of view of a given processing element, remote memory is memory which cannot be accessed via standard load and store operations. For example, host memory is remote to a GPU core; the local store of a Cell SPE is remote to the other SPEs or the PPE.

MRAPI offers a set of API functions for manipulating remote memory. As with MRAPI shared memory, functions are provided for creating, initializing and attaching to remote memory. Unlike with MRAPI shared memory, the API provides functions for reading from and writing to remote memory.

The API does not place restrictions on the mechanism used for data transfer. However, catering to the common case where it is desirable to overlap data movement with computation, the API provides non-blocking read and write functions. In addition, flush and sync primitives are provided to allow support for software-managed caches. The API read and write functions also support scatter/gather style accesses.

For MRAPI users and implementers concerned about performance of the flush and synch functions, the MRAPI working group recommends use of multiple memory regions when the application writer is overly concerned; the implementation of the flush routine should have the semantics of "anything that is dirty should be pushed back to memory", versus “everything should be pushed back to memory”.