The GASNet memory-to-memory data transfer functionality shares similarities with remote memory operations in MRAPI. Unlike MRAPI, GASNet does not support scatter/gather operations. On the other hand, GASNet provides more sophisticated synchronization primitives for non-blocking operations, and supports register-to-memory copies. The extended GASNet API includes barrier synchronization, which is out of scope for MRAPI (as discussed in 2.2.2, co-ordination between processes is part of the scope of MTAPI). Another significant distinction is that GASNet provides for both message passing and remote memory operations. Message passing is *not* part of MRAPI, which is intended to co-exist with a message passing API such as MCAPI.

2.2.6 ARMCI library

ARMCI (Aggregate Remote Memory Copy Interface) is a library for remote memory access operations. ARMCI has been designed to be general purpose and portable, but is aimed at library implementers rather than application developers.

ARMCI shares similarities with remote memory operations in MRAPI. Unlike MRAPI, ARMCI provides guarantees on the order of remote memory operations issued by a given process. ARMCI uses *generalized I/O vectors* to support movement of multiple data segments between arbitrary remote and local memory locations. This is more general than the form of remote memory operations supported by MRAPI; the structure of MRAPI operations matches the ARMCI *strided* format, a special class of generalized I/O vectors where local and remote memory regions are regularly spaced. As well as put and get operations, ARMCI supports remote accumulate. This functionality is mainly useful in the high-performance/scientific computing domain (accumulation is also featured in the MPI-2 one-sided communication API). Accumulate operations are not present in MRAPI, which is not specifically geared towards this application domain.

2.3 The MRAPI Feature Set

MRAPI provides features in three categories: *Synchronization Primitives*, *Memory Primitives*, and *Metadata Primitives.*

The Synchronization Primitives (Section 4.3) are:

- **Mutexes** - binary primitives which could be provided by shared memory, a distributed runtime, or other means
- **Semaphores** - counting primitives that provide more capability than mutexes, although at perhaps a slight performance penalty
- **Reader/Writer Locks** – more advanced primitives that give the ability to support multiple readers concurrently while allowing only a single writer

The Memory Primitives (Section 4.4) are:

- **Shared Memory** - allows an application to allocate and manage shared memory regions where there is physical shared memory to support it, including special features which provide support for requesting memory with specific attributes, and support for allocation based on a set of sharing entities
- **Remote Memory** - allows an application to manage buffers that are shared but not implemented on top of physical shared memory; transport may be via chip-specific methods such as DMA transfers, SRIO, software cache, etc. Remote Memory Primitives also provide random access, scatter/gather, and hooks for software managed coherency.

The Metadata Primitives (Section 4.6) provide access to hardware information. They are not intended to be a facility for an application to create and manage its own metadata. This additional functionality could be a layered service or a future extension.