2 Introduction

This document is intended to assist software developers who are either implementing MRAPI or writing applications that use MRAPI. The MRAPI goals are presented, followed by a brief review of existing standards to motivate the need for a new standard and a short discussion of how MRAPI is intended to provide the desired functionality while meeting the goals of the MRAPI working group.

2.1 MRAPI Goals

MRAPI is intended to provide essential capabilities for allowing applications to cooperatively manage shared resources in Multicore systems. Therefore, MRAPI runtimes are not required to provide secure enforcement of sharing policies, and furthermore MRAPI intentionally stops short of directly providing a full-featured dynamic resource manager capable of orchestrating a set of resources to satisfy constraints on performance, power, and quality of service. It is envisioned that MRAPI (in conjunction with other Multicore Association APIs) can serve as a valuable tool for implementing applications, as well as for implementing such full-featured resource managers and other types of layered services. For these reasons, the following set of goals were established and used to carefully weigh each feature included in the MRAPI API:

- Small application layer API, suitable for cores on a chip and chips on a board
- Easy to learn and use, incorporates essential feature set
- Supports lightweight and performant implementations
- Does not prevent/preclude use of complementary approaches
- Allows silicon providers to optimize their hardware and take advantage of hardware features
- Allows implementers to differentiate their offerings
- Can run on top of an OS, hypervisor, or bare metal
- Can co-exist with hardware acceleration
- Supports hardware implementations of the API
- Does not require homogeneous cores, operating system, or memory architecture on chip
- Provides source code level portability

2.2 Existing Standards and APIs

This section provides brief reviews of related standards and discusses how the MRAPI working group chose to address specific areas of functionality.

2.2.1 POSIX® Shared Memory

Shared memory is used to allow access to the same data by multiple threads of execution, which may be on the same processor or on multiple processors, thereby avoiding copying of the data. Posix provides a standard API for using shared memory, including allocation, deletion, mapping and managing the shared memory. Posix shared memory generally provides this functionality within the scope of one operating system, across one or multiple processor cores. This functionality is considered essential for Multicore programming, but is only one feature that MRAPI is intended to provide. The MRAPI working group wanted to add two additional features to a shared memory API, namely: (1) the ability for programmers to specify attributes of the memory to be shared (for example on-chip SRAM versus off-chip DDR), and (2) the ability for programmers to specify which elements of a Multicore system would be seeking access to the shared memory segment such that MRAPI could support shared memory for parts of Multicore systems where physical shared memory is non-uniformly accessible.