- Keep only the concept of named semaphores and also match semantics of MCAPI for endpoints
- Ignore the XSI type interface (avoid requiring the API user to create and manage a set of semaphores and semaphore operations per call)
- Provide non-blocking operations in a way that is consistent with MCAPI
- default is visible across processes/tasks/etc.
- does not require shared memory or SMP OS
- priority inversion cannot be dealt with by MRAPI until MTAPI comes along
- provide primitive deadlock reporting

2.2.3 Performance API (PAPI)

PAPI is a high performance oriented API that defines a common set of useful performance counters. PAPI provides a high level interface to start, stop, read, and register callbacks for counter overflow. PAPI provides metadata about resources of a system, including resources such as number of cores, number of counters, and shared libraries in use by an application.

PAPI also provides derived counters, such as IPC (Instructions Per Cycle), and timing and measurement functions, such as wall clock time consumed. It also provides mutexes, supports external monitoring of counters associated with a process or thread, and management functions concerning registering threads. PAPI has no memory management, has no concept of system partitioning, and the metadata is limited with respect to the total resources in an SOC.

The MRAPI working group views the PAPI features for metadata and performance counters as being a useful concept for the types of systems targeted by MRAPI.

2.2.4 IBM DaCS

IBM's DaCS library provides a portable API for managing distributed memory systems. It allows programmers to take advantage of the Cell processor's "Synergistic Processing Unit" (SPU) DMA engines, while still being able to execute the program on machines that do not have DMA engines. It provides functions for creating memory regions, registering memory regions on multiple distributed processors, and copying data in and out of those memory regions via DMA.

The MRAPI API shares several concepts with DaCS. In particular, both APIs provide functions for creating memory regions, registering them with multiple processors, and performing DMA operations between distributed shared memory and local memory. In order to minimize API complexity, MRAPI does not provide some features included in DaCS. In particular, the MRAPI APIs avoid the need to specify permissions on memory regions and limit DMA operations to linear or strided data arrays.

2.2.5 GASNet specification

The GASNet (Global-Address Space Networking) specification is an API aimed at implementers of Global-Address Space languages such as Unified Parallel C, and Titanium. Unlike MRAPI, GASNet is geared towards single program multiple data (SPMD) high-performance computing applications, rather than embedded systems.

GASNet is divided into a small core API, and a richer extended API. The core API consists of functions for job control, message passing (based on *Active Messages*), and atomicity control. The extended API enriches this functionality with memory-to-memory data transfer functions, lower-level register-to-memory operations, barrier synchronization, and threading support. The extended API has been designed to be implementable using only the core API, and the GASNet designers provide a portable reference implementation of the extended API in terms of the core API. However, high-performance GASNet implementations are expected to efficiently implement as much of the extended API as possible, exploiting platform-specific characteristics.