A: The working group has considered the possibility of allowing the “promotion” of local memory to remote memory, and then allowing all memory accesses to occur through the API. This would effectively hide the difference, but at a performance cost. For now this is a deferred feature.

Q: In many HW systems transitions between low power (or no power) and fully working conditions are extremely frequent. In such systems some state change callbacks will become a nightmare. How are you planning to handle the situation?

A: In the situation where the application does not want to be disturbed by frequent callbacks then it would be better for the application to periodically poll MRAPI at a time of its own choosing. This is certainly possible with MRAPI.

Q: What is the idea of API asking for HW accelerators if these accelerators are actually powered off because of inactivity?

A: In such a scenario the application would determine that there was no acceleration available and would have to find an alternative means to perform its work, perhaps by executing code on the CPU.

Q: Are there any plans to include trigger APIs? For example, invoke callback when a particular resource hits some pre-defined conditions/threshold?

A: Currently there are no threshold-related callbacks other than counter wrap-arounds. MRAPI may consider this for a future version.

Q: Primitives - didn't you consider including RCUs (Read Copy Update locks)?

A: The MRAPI working group did consider read copy update locks. After discussion with some of the original creators of the RCU code for Linux we determined that for now there is not sufficient evidence that a high performance user-level implementation of RCU was feasible. We intend to monitor developments in this area as we are aware that it is an active area of research.

Q: These primitives are necessary, but seem to be insufficient. I would think that the goal of MRAPI would include the ability to write a resource manager that any app that uses MRAPI could plug into. That implies that at a minimum: resource enumerations should be standardized, or a mechanism for self-describing enumerations be created.

A: MRAPI is intended to provide some of the primitives that could be used for creating a higher level resource manager. However, it is also intended to be useful for application level programmers to write multicore code, and for this reason it was kept minimal and orthogonal to other Multicore Association APIs. The working group believes that a full-featured resource manager would require all of the Multicore Association APIs, e.g., MCAPI, MRAPI, and MTAPI.

Q: Are any companies currently incorporating or have plans to incorporate MRAPI in their products. If so, can you name the products?