MCA Standards
For
Closely Distributed Multicore
Sven Brehmer

Multicore Association, cofounder, board member, and MCAPL WG Chair
CEO of PolyCore Software
Embedded Systems

• Spans the computing industry
  – Telecom & datacom
  – Consumer electronics
  – A&D
  – Automotive
  – Industrial

• Comes in many shapes and forms
  – Resource limitations
  – Real-time requirements

• Increasingly contains multicore
Multicore drivers

• Performance
  – Concurrency
  – Acceleration

• Power consumption
  – Multiple processors at lower clock rates

• Consolidation
  – System => board => chip
  – Combination of products
Multicore Challenges

• Unlimited variety
  – Cores/chip, accelerators, operating systems, transports/interconnects, memory architectures

• Programming Models
  – Messaging passing, new paradigms

• Portability and scalability

• Concurrency and parallelization granularity

• Virtualization?
The Multicore Association

- Established in 2005

- Mission: Improve time to market through the use of industry standards

- International membership encompasses Asia, Europe, North America

- Committee-based standards development
MCA Strategic Roadmap

MCA Foundation APIs

**Communications** (MCAPI)
- Lightweight messaging

**Resource Management** (MRAPI)
- Basic synchronization
- Shared/Distributed Memory
- System Metadata

**Task Management** (MTAPI)
- Task lifecycle
- Task placement
- Task priority

**SW/HW Interface for Multicore/Manycore** (SHIM)
- XML HW description from SW perspective
MxAPIs Target Domain

- Embedded multi-processing that requires task to task communication
- Closely distributed systems and/or tightly coupled
- Heterogeneous & homogeneous systems
  - Cores, interconnects, memory architectures, OSes
- Scalable: 1 – 1000’s of cores
- Allow implementations with significantly lower latency, overhead & memory footprint
- Provides incremental migration path for application codes
- Supports multiple levels of parallelism
Working Groups

• Communications API – MCAPI, 2.0
• Programming Practices – MPP, 1.0
• Resource Management – MRAPI, 1.0
• Software-Hardware Interface - SHIM
• Task Management – MTAPI, 1.0
• Tools Infrastructure - TIWG
• Multicore Virtualization - MVWG
System Developer is Primary Beneficiary

• Better ease of use and increased availability of products
• Simplified development and management of multicore software
• A more consistent interface and documentation between hardware and software vendors
• Quicker time to market
• Software re-use
• More vendor choice
Secondary Beneficiaries

• Tool vendors
  – Reduced cost for tool support
  – Increased tool availability

• Processor and/or chip company
  – Higher application performance achievable
  – Reduced support burden through standardization
  – Easier upgrade path for customers

• OS and middleware vendor
  – A uniform foundation for present/future products
  – Abstracts platform capabilities; helps auto-configuration
  – Larger potential market
MCAPI – Addressing the Need

- Multicore evolution requires message passing
- Messages, Packet & Scalar channels
- Basic topology discovery
- Inherent synchronization – SMP/AMP
- Standardized programming model works with any C compiler
MRAPI Conceptual Architecture

- Basic synchronization – mutexes, semaphores
- Shared/remote memory
- System metadata provides hardware information
• Tasks execute a job, implemented by an action function, on a local or remote node
• Task can be started individually or via queues (to influence the scheduling behavior)
• Arguments are copied to the remote node
• Results are copied back to the node that started the task
Shim (spacer)

From Wikipedia, the free encyclopedia

A shim is a thin and often tapered or wedged piece of material, used to fill small gaps or spaces between objects. Shims are typically used in order to support, adjust for better fit, or provide a level surface. Shims may also be used as spacers to fill gaps between parts subject to wear.

- Multicore tools, as well as OS/middleware, help applications run optimally on multicore chips while hiding the hardware specifics
- However, the tools must understand the hardware specifics
- SHIM provides tools with the specific multicore hardware description in a standardized, open XML model
Tools – Abstraction, automation & portability

- Wizards provide guidance
  - MCAPI inserted into application under programmer direction

```c
// Worker node gets data
mcapi_msg_recv( recv_endpoint, buffer, buffer size, receive count, status);
if (status != MCAPI_SUCCESS)
{
    handle error;
}

// Worker operates on data

// Worker node sends results
mcapi_msg_send( send_endpoint, recv_endpoint, buffer, buffer size, priority, status);
if (status != MCAPI_SUCCESS)
{
    handle error;
}
```
Questions

Contact info:
• sven.brehmer@polycoresoftware.com