Leveling the Field for Multicore Open Systems Architectures

Markus Levy

President, EEMBC

President, Multicore Association

Analyzing the Multicore Ecosystem

- Embedded Microprocessor Benchmark Consortium® (EEMBC)
- Industry benchmarks since 1997
- Tool for evaluating embedded processors, compilers, systems
- MultiBench for shredding multicore processors

Enabling the Multicore Ecosystem

- Initial engagement began in May 2005
- Industry-wide participation
- Current efforts
  - Communications APIs
  - Hypervisors
  - Multicore Programming Practices
  - Resource Management

Multicore Issues to Solve

- Concurrent programming
- Communications, synchronization, resource management between/among cores
- Performance analysis
- Debugging
- Distributed power management
- OS virtualization
- Modeling and simulation
- Load balancing
- Algorithm partitioning

Multicore Benchmarking Rules

- Do not rely on a single answer
- Match your application requirements
  - Small or large data sets
  - Few or many threads
  - Dependencies
  - OS overhead

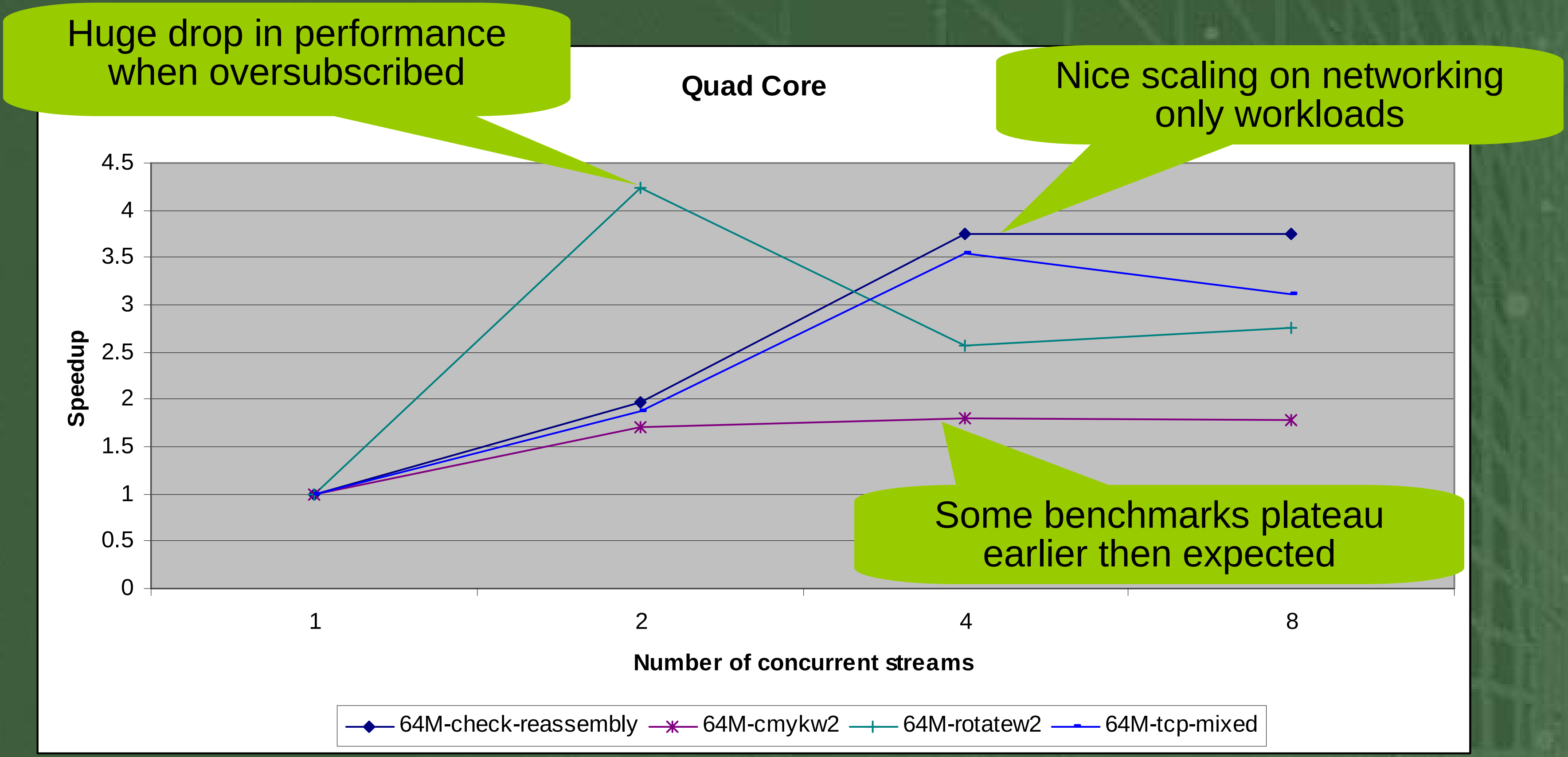
Benchmarking Multicore – What’s Important?

- Measuring scalability
- Memory and I/O bandwidth
- Inter-core communications
- OS scheduling support
- Efficiency of synchronization
- System-level functionality

EEMBC Multicore Strategy

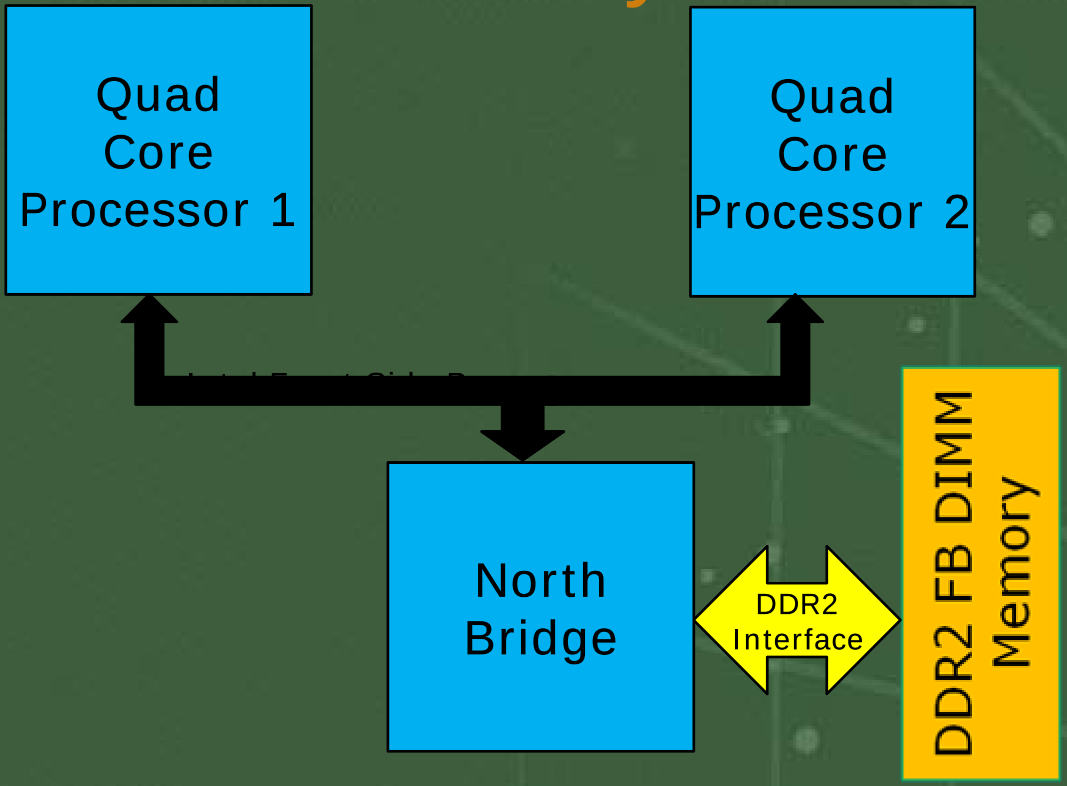
- Evaluation and future development of scalable SMP architectures
  - – Includes multicore and manycore
- Measure impact of parallelization and scalability across both data processing and computationally-intensive tasks

Some Results



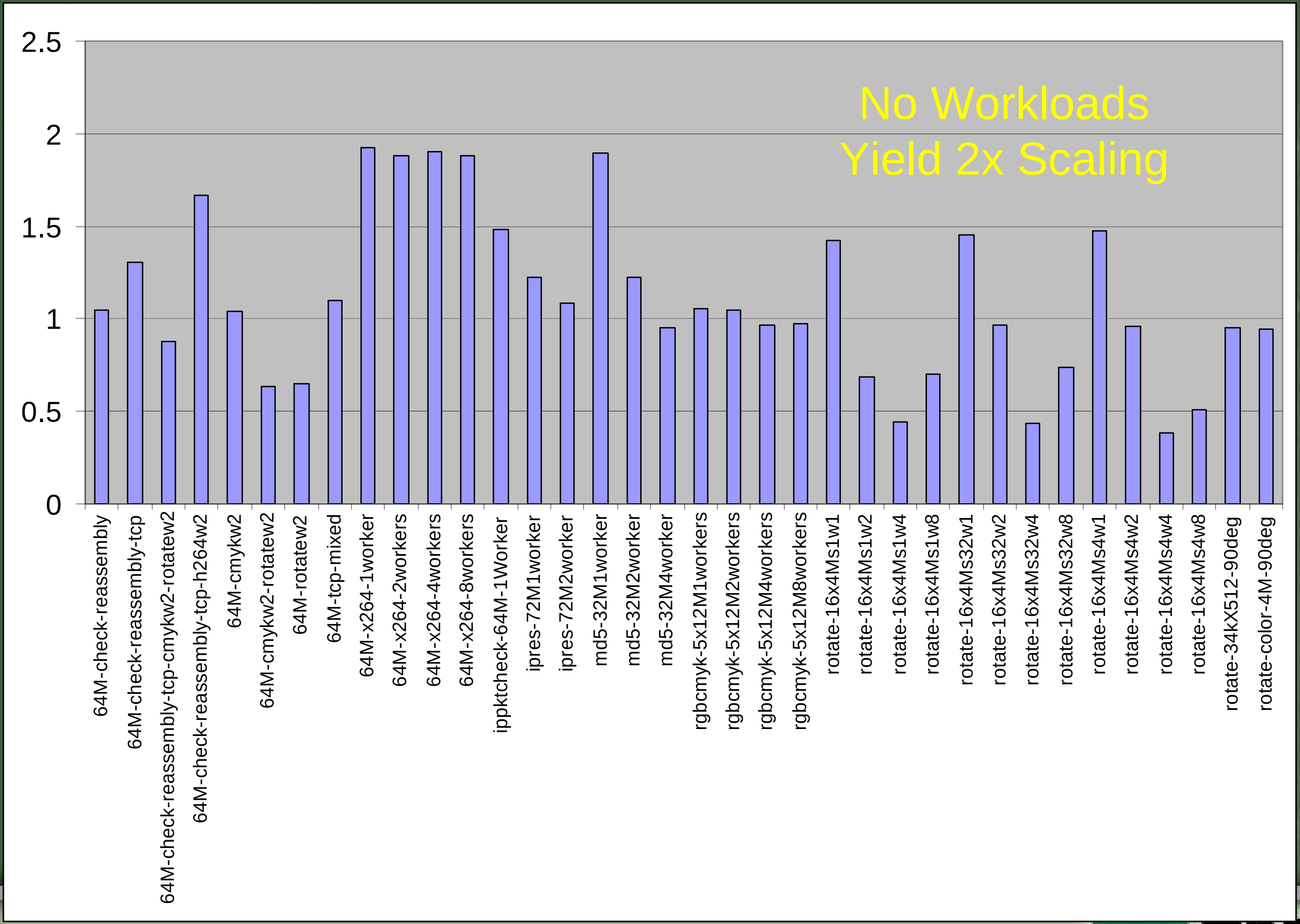
MultiBench Results

Two Processor System Utilizing Single Memory Controller



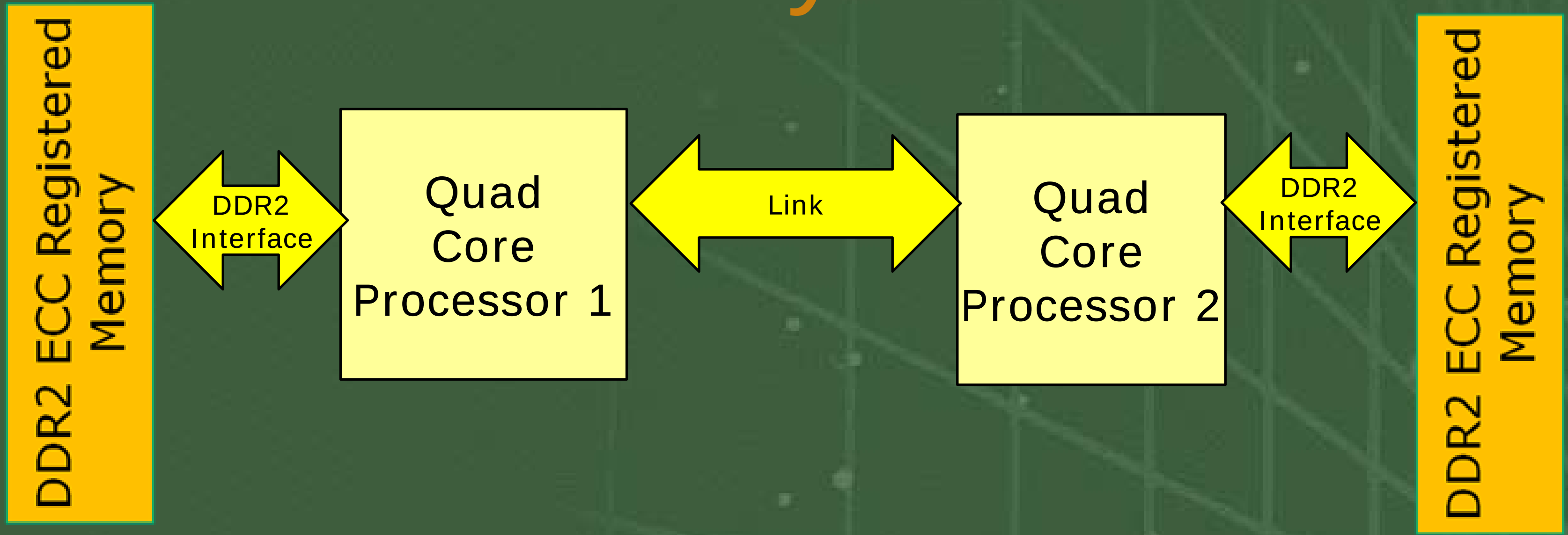
Processors 1 and 2 must always arbitrate for memory via their front side bus connection through the North Bridge.

Dual Quads vs. Single Quad



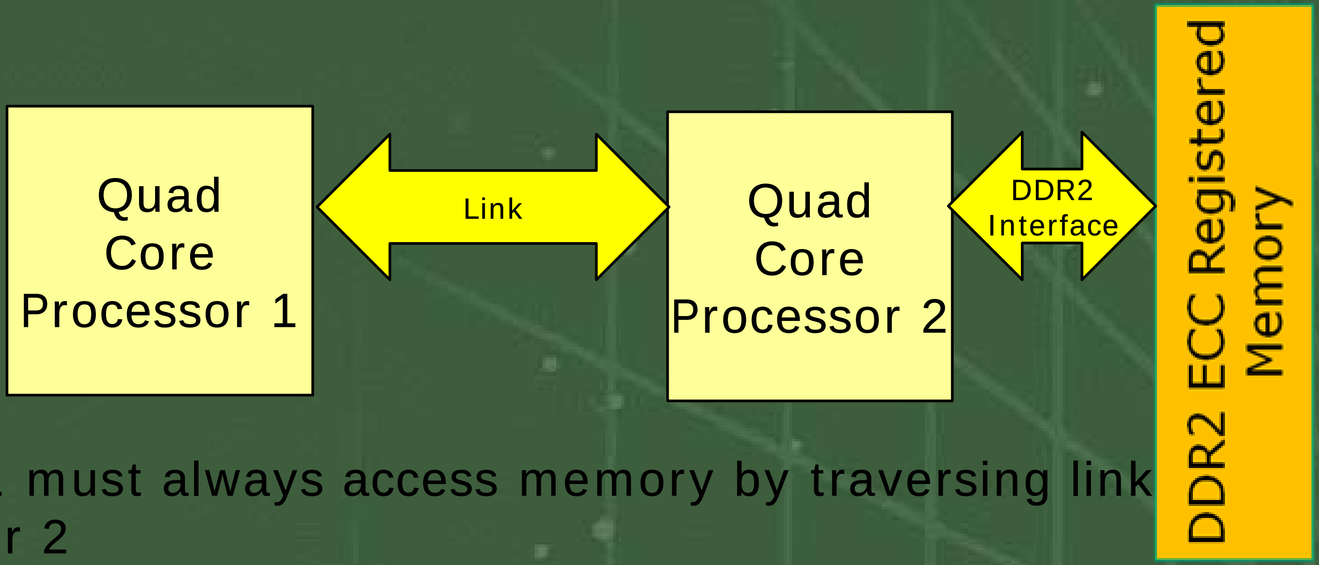
1. Find max values for each workload
2. Ratio of all scores

Two Processor System Utilizing Dual Memory Controllers

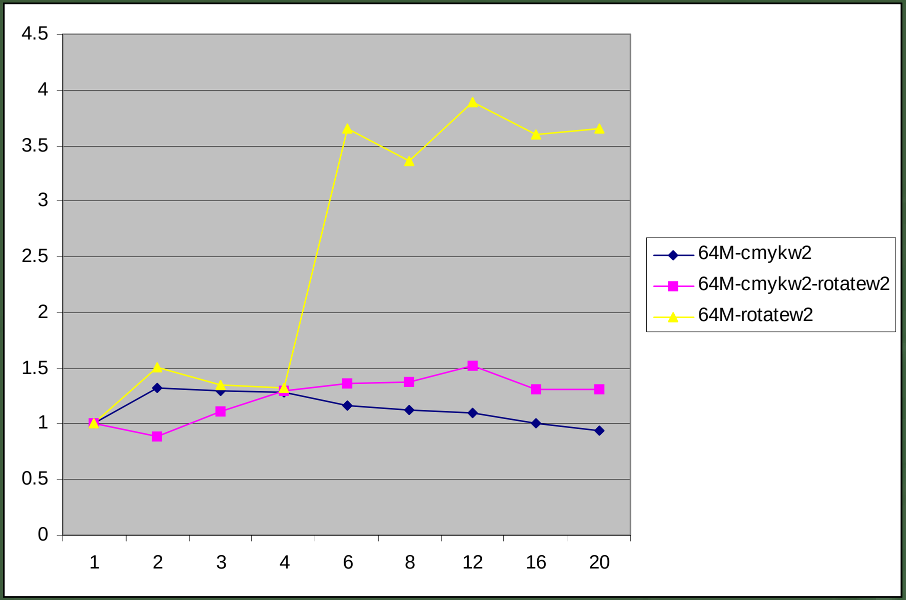


Direct Access
Shared Access
Doubly Shared Access

Two Processor System Utilizing Single Memory Controller

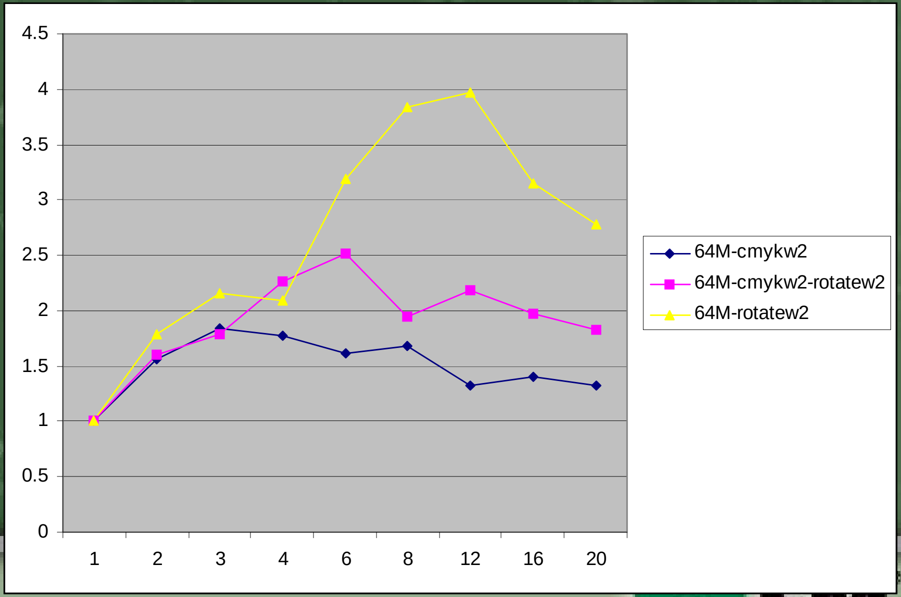


- Processor 1 must always access memory by traversing link to Processor 2
  - Requires arbitration to access Processor 2's memory
- Processor 2 always has prioritized access to this memory since it is directly attached.
- Affinity can help performance



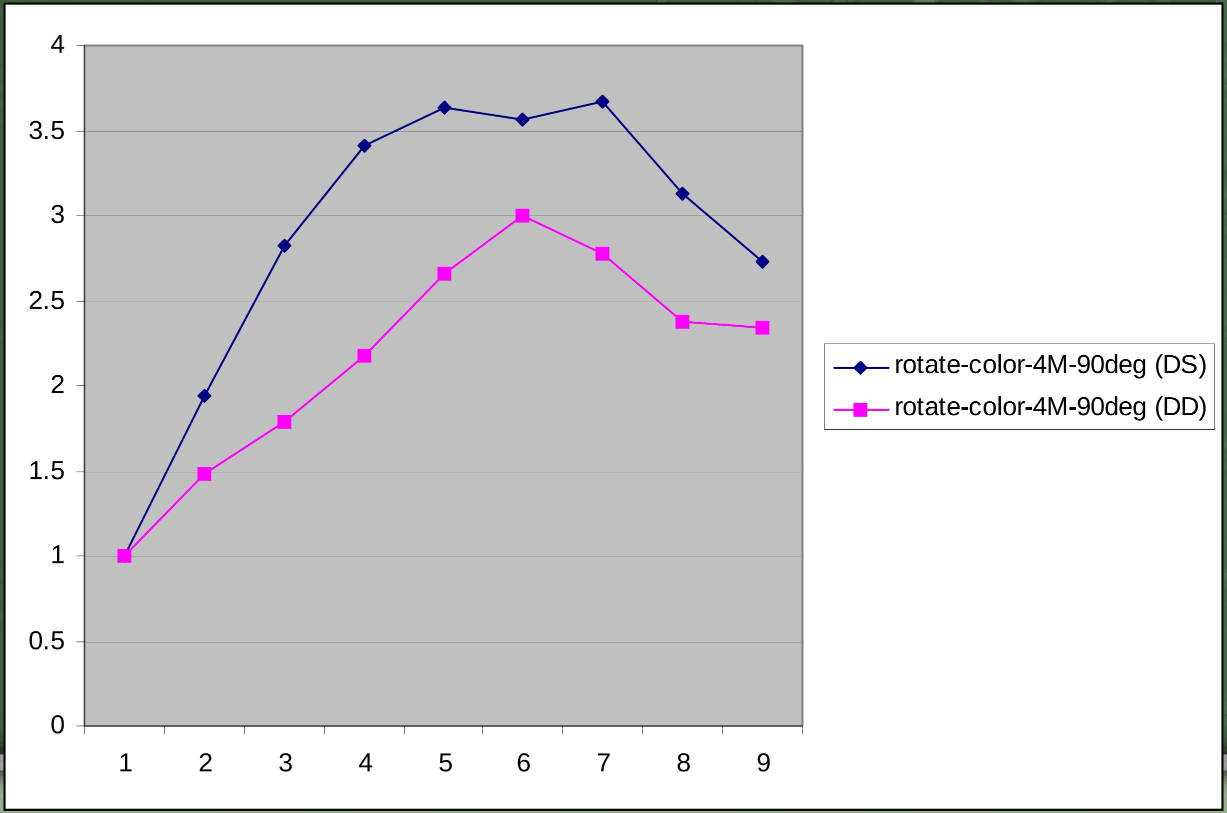
Dual Quad Cores –
Single Memory Controller

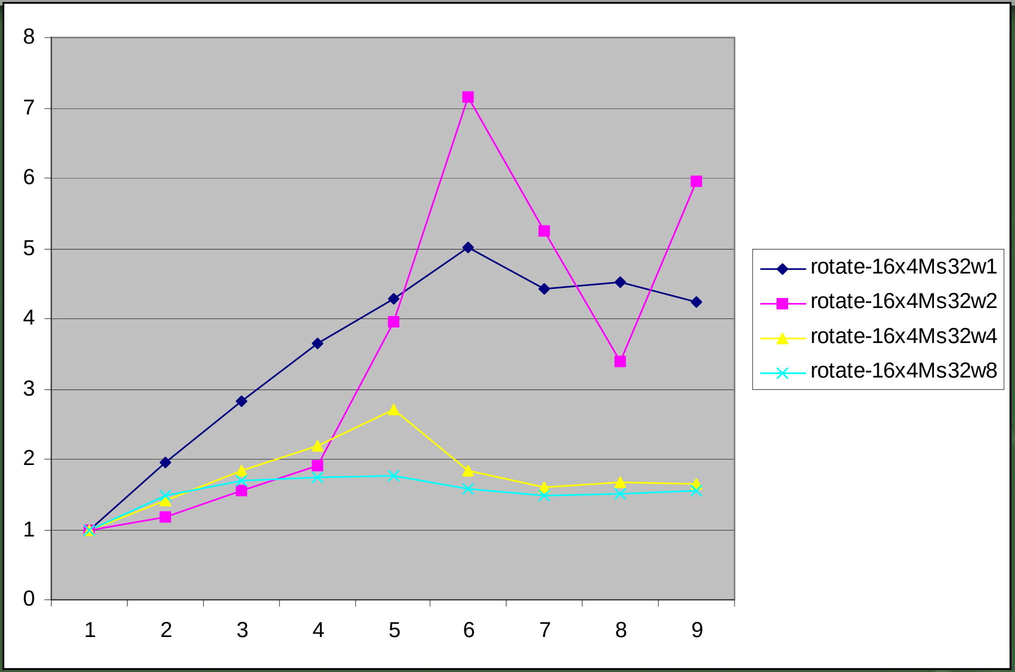
Dual Quad Cores –
Dual Memory Controllers



- Rotation by multiple workers cooperating to process a single image.
    - Each worker thread acquires slices and writes them to the output buffer.

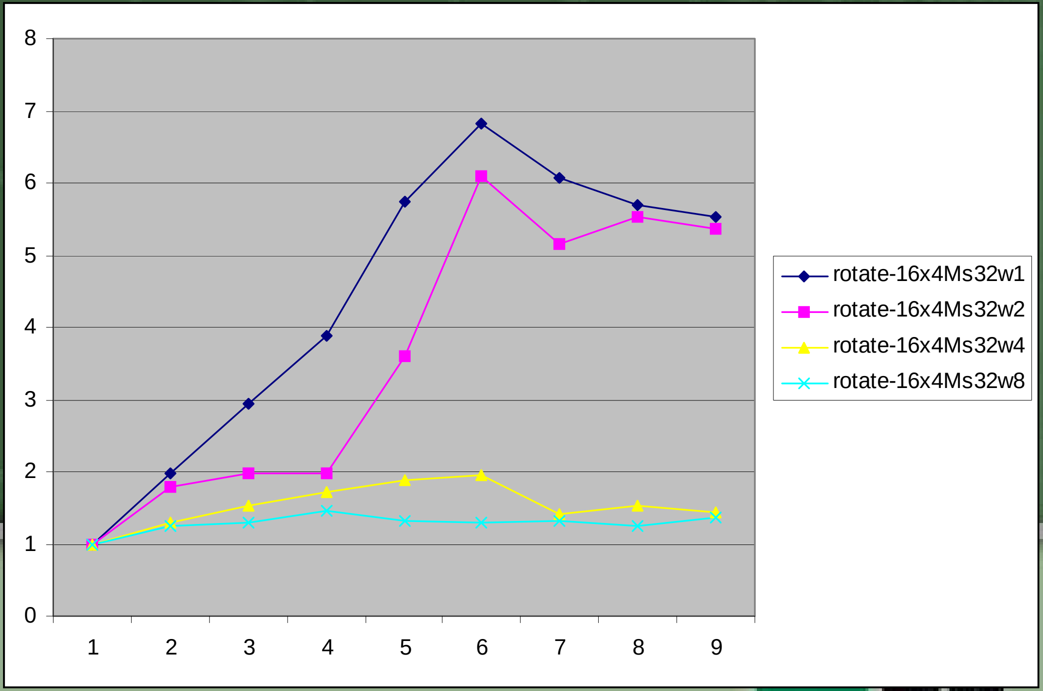
- Potential bottlenecks related to memory interfaces and synchronization between worker threads.



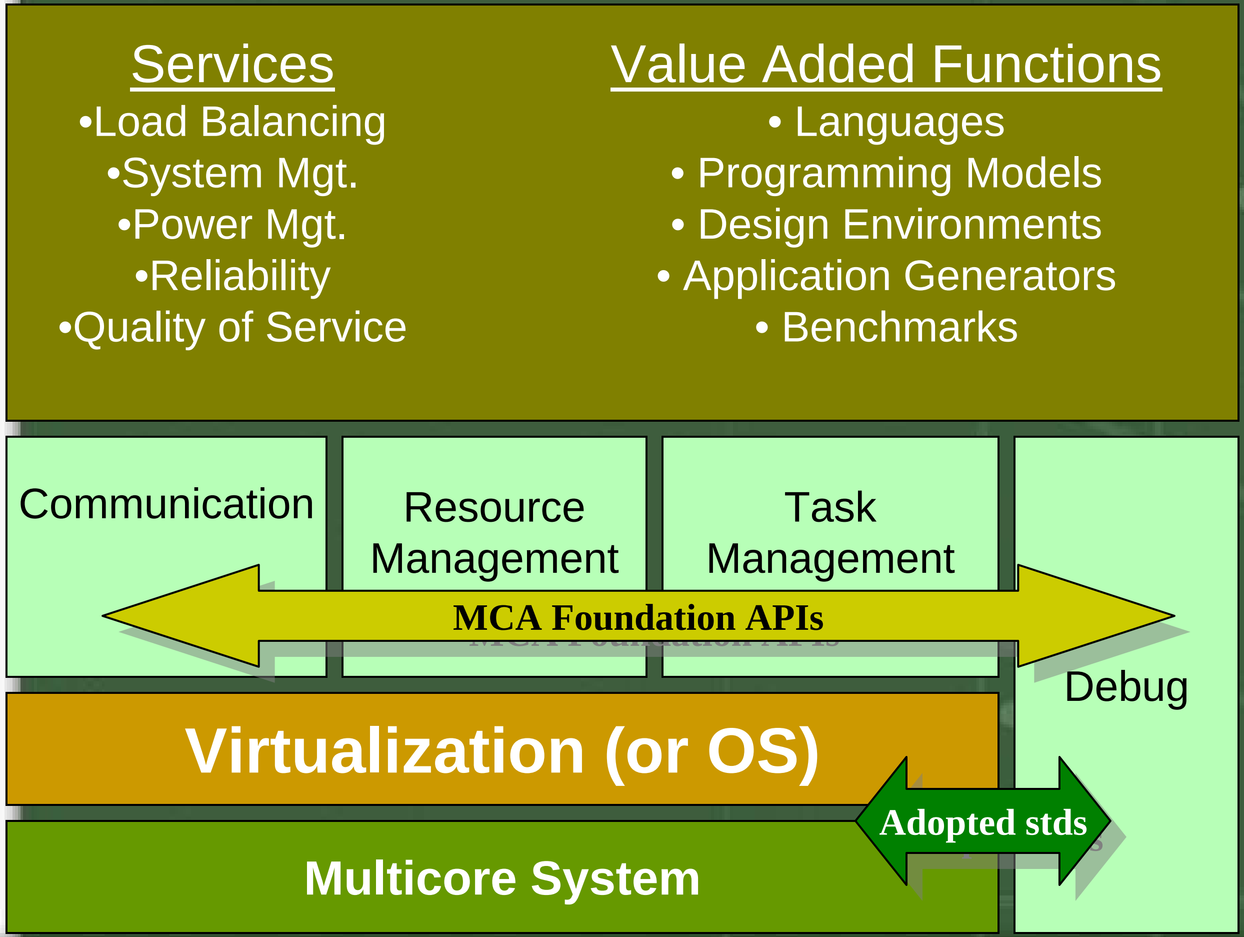


Dual Quad Cores –
Dual Memory Controllers

Dual Quad Cores –
Single Memory Controller



The Multicore Association Roadmap



The Four MCA Pillars

Communications
- MCAPI: ultra-light weight

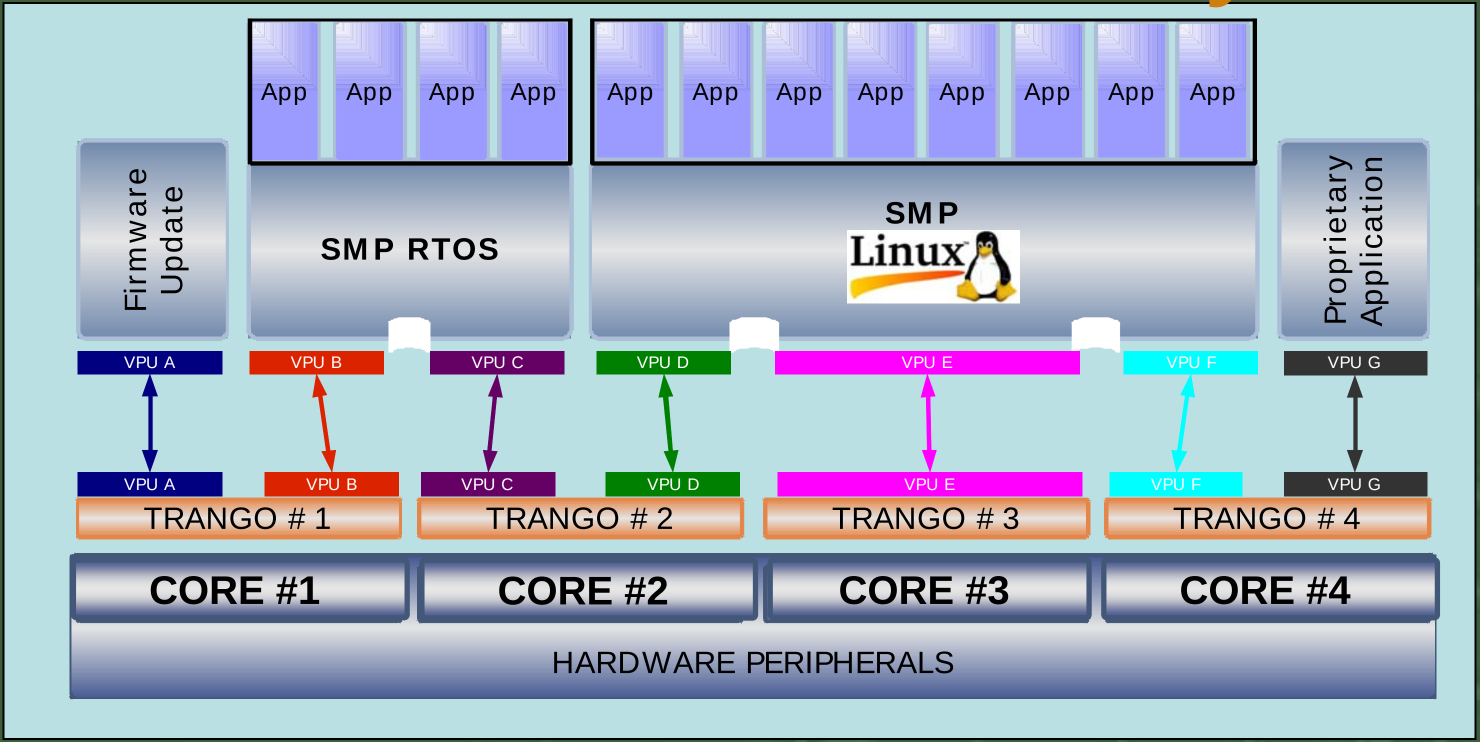
Resource Management
- Memory management
- Basic synchronization
- Resource registration
- Resource partitioning

Task Management
-Task scheduling

Why Virtualize?

- Hardware consolidation
- Migration and hosting of legacy applications
- Resource management and balancing
- Faster provisioning
- Fault tolerance

Virtualized Multicore System



- Fractional CPU assignment for background tasks such as firmware update/system health monitor/power management
- Benchmarking involves many system-level elements

EEMBC Hypermark Important Metrics

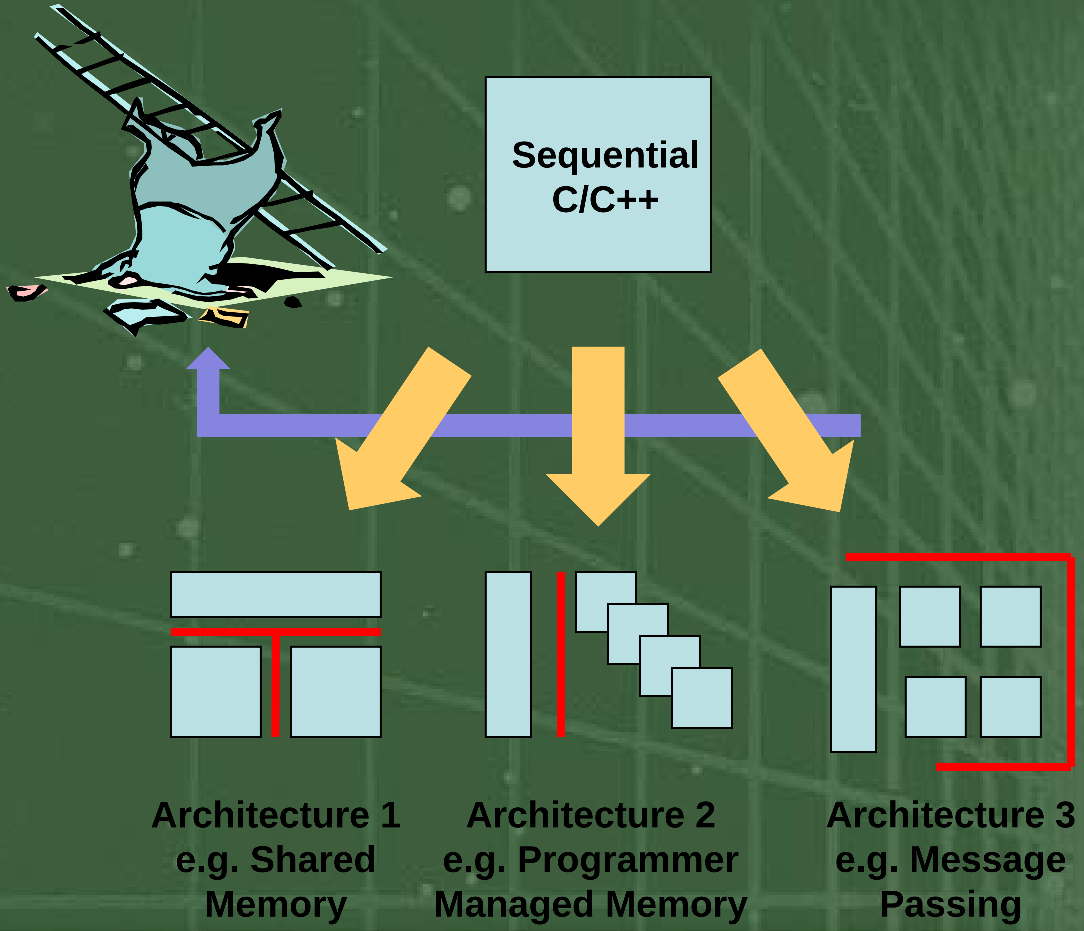
- Overall performance overhead of a hypervisor, i.e. CPU loading
- Static footprint (code and data size)
- Jitter
- Interrupt latency
- Comparison to native performance

Multicore Programming Practices (MPP)

- Long term
  - Continue research into languages, methodologies, etc
- Short term
  - How today's embedded C/C++ code may be written to be "multicore ready"
- Influence of a group of like-minded methodology experts to ensure completeness, usefulness and industry-wide compatibility
- Creation of a standard "best practices" guide through a recognized, neutral industry body
  - Based on capturing current best practices

MPP Scope & Approach

- Focus on existing C/C++ without extensions, targeting current architectures
- Draw up framework of common pitfalls when transitioning from serial to parallel
- Consider solutions or avoidance tactics
- Analyze performance of solution


Sequential C/C++
Architecture 1 e.g. Shared Memory
Architecture 2 e.g. Programmer Managed Memory
Architecture 3 e.g. Message Passing

Summary

- Performance analysis highlights benefits and pitfalls
- EEMBC licensing and membership

- Portability prevents entrapment
- Multicore Association standards and working groups