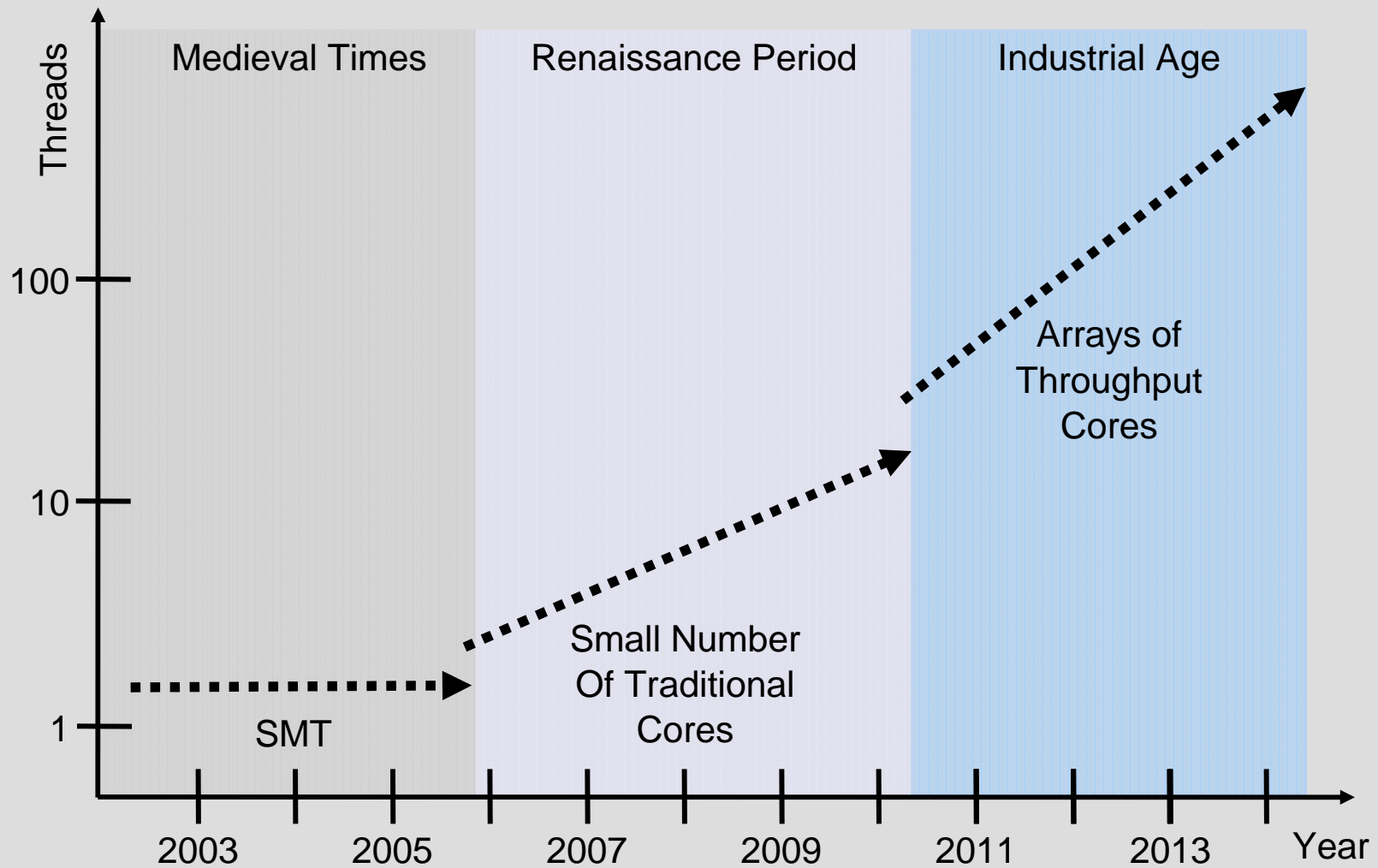


CAC 07 Panel: The role of accelerators in cluster communication

Fabrizio Petrini
Applied Computer Science Group
Pacific Northwest National Laboratory
fabrizio.petrini@pnl.gov

CAC Panel, Long Beach, CA, March 26th 2007

The advent of teraflop-scale, many-core processors.



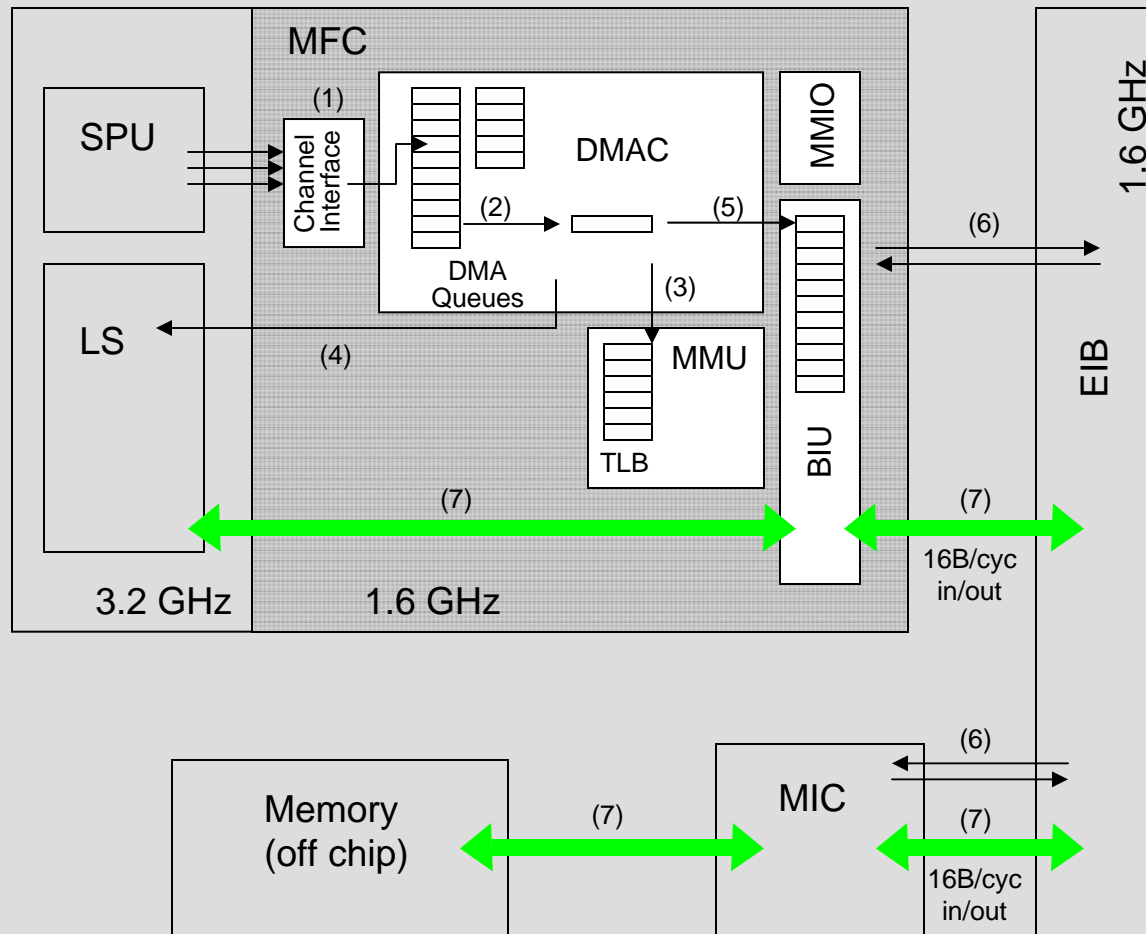
HW questions, SW answers

- ▶ With the many-core technology, we are finally going to have the ability to put low cost high performance (teraflop-scale) computing on the desktop, not only on a cluster
- ▶ We can integrate a large number of cores/specialized processing units (HW is not the real problem)
 - Accelerators & cores will happily live together in the same chip
- ▶ SW challenge: if programming/parallelism is an insurmountable (or too expensive) obstacle, then we may need a whole rethinking of the innovation process.

Many-core processors rapidly appearing on the horizon: ballpark analysis

- ▶ Tens of cores
 - each one with specialized I/O DMA units, synchronization units, etc.
- ▶ Hundreds of gigaflops/teraflop in a socket
- ▶ Approaching a hundred Gigabyte/sec I/O (memory + network) bandwidth
- ▶ Network bandwidth in the few Gigabytes/second
 - So it is not the major communication bottleneck
- ▶ Should we use
 - Standard processing cores
 - Or specialized units?
 - Is there a clear distinction between the two?

Cell SPE Internal Architecture



What Communication Functions Should ALWAYS be Off-loaded to Accelerators

- ▶ Low/Mid-level HW protocol functions
 - DMAs
 - Virtual to Physical
 - Packetization
 - CRC checking
 - Packet re-transmission
 - Anything idempotent (that can be fixed by re-trying)
 - Atomic operations
- ▶ Tag Matching, if possible
- ▶ And it would be nice to see support for collective communication

Which Should NEVER be offloaded to accelerators?

- ▶ Complex protocol processing that requires sophisticated algorithmic design
- ▶ Things that cannot be implemented/debugged easily
 - Again, complex protocols which lead to complex software, expensive to port and maintain on “special purpose” accelerators
- ▶ Business more than technical problem
 - The cost of the SW development

Are There Cases for Which it Makes Sense to Offload to Another Core Rather than to an Accelerator

- ▶ Interesting question: this is a clear trend
- ▶ Multi-core Uni-bus
 - Many-cores, but not clear whether there is enough
 - Bandwidth
 - Local Memory
 - Work to do
 - For all of them
- ▶ Possibility of implementing rather complex communication protocols
 - Not possible by offloading to standard, slow network processors
- ▶ Quadrics QsNET3 (Elan5) vs BlueGene/C (Cyclops)

Collective Communication

- ▶ Commercial computing does not seem to understand the need of collective communication
- ▶ Mostly because it is “sold” in the wrong way
 - A performance improvement
- ▶ Collective communication is essential for
 - (Performance) Debugging
 - Resource management
 - Attacking the real problem of large-scale/parallel systems
 - Non-determinism

Multicore Processors: Architectural Trends

- ▶ Explicit vs Speculative Parallelism
 - VLIW, SIMD, thread parallelism vs HW speculation
- ▶ Simpler Processing Cores
 - Lack branch prediction, re-ordering buffers and other specialized units
- ▶ Limited amount of on-chip memory per core
 - Memory size scales at a slower rate
- ▶ Limited amount of off-chip bandwidth
 - Important technological constraint
- ▶ Teraflop on a socket rapidly appearing on the horizon

Software Trends

- ▶ The burden is shifting onto the programmer/compiler/run-time system
 - Explicit parallelism
 - Management of the memory hierarchy
 - Data orchestration between different levels
 - Communication,
 - Synchronization
 - Non-determinism
- ▶ Harsh reality: we need new parallel algorithms

Proposed Approach

- ▶ Simplify First: we have to learn to crawl before we try to run
 - High-level algorithmic design vs machine-dependent optimizations
 - Separate
 - Computation
 - On-chip communication
 - Off-chip communication
- ▶ Accurate Performance Modeling of Each Building Block
 - Provide insight into the performance of the essential components of the application
- ▶ Kill the real problem, non-determinism
 - Bulk-synchronous parallel software design
 - Use of “collective memory transactions” for global coordination
- ▶ Put the pieces back together
 - Allow concurrent execution of several activities (but only at the **very end**)

Recent Work

- Challenges in Mapping Graph Exploration Algorithms on Advanced Multi-core Processors, *International Parallel and Distributed Processing Symposium, March 2007*
 - <http://hpc.pnl.gov/people/fabrizio/papers/ipdps07-graphs.pdf>
- Cell Multiprocessor Interconnection Network: Built for Speed, *IEEE Micro, May/June 2006*
 - <http://hpc.pnl.gov/people/fabrizio/papers/ieeemicro-cell.pdf>
- Multicore Surprises: Lesson Learned from Optimizing Sweep3D on the Cell Broadband Engine, *International Parallel and Distributed Processing Symposium, March 2007*
 - <http://hpc.pnl.gov/people/fabrizio/papers/ipdps07-sweep3d.pdf>
- Peak-Performance DFA-based String Matching on the Cell Processor, *Third International Workshop on System Management Techniques, Processes, and Services (SMTPS), March 2007*
 - <http://hpc.pnl.gov/people/fabrizio/papers/smtps07.pdf>
- Programming the Cell Processor, *Dr. Dobbs, April 2007.*