C-DAC Four Days Technology Workshop

ON

Hybrid Computing – Coprocessors/Accelerators
Power-Aware Computing – Performance of Applications Kernels

hyPACK-2013
(Mode-1:Multi-Core)

Lecture Topic:

Multi-Core Processors : Multi-Core Architecture Part-I

Venue : CMSD, UoHYD ; Date : October 15-18, 2013

Multi-Core Processors

Lecture Outline

Following Topics will be discussed

- An overview Multi Cores
- Understanding of Intel /AMD Multi-Core Architectures
- Performance Issues

Source : http://www.intel.com ; http://www.amd.com

Source : Reference : [4], [6], [14],[17], [22], [28]

Part-I: Threading

Evolving towards model-based Computing

Large dataset mining
Semantic Web/Grid Mining
Streaming Data Mining
Distributed Data Mining
Content-based Retrieval

Collaborative Filters
Multidimensional Indexing
Dimensionality Reduction
Efficient access to large, unstructured, sparse datasets
Stream Processing

Multimodal event/object Recognition
Statistical Computing
Machine Learning
Clustering / Classification
Model-based:
Bayesian network/Markov Model
Neural network / Probability networks
LP/IP/QP/Stochastic Optimization

Photo-real Synthesis
Real-world animation
Ray tracing
Global Illumination
Behavioral Synthesis
Physical simulation
Kinematics
Emotion synthesis
Audio synthesis
Video/Image synthesis
Document synthesis

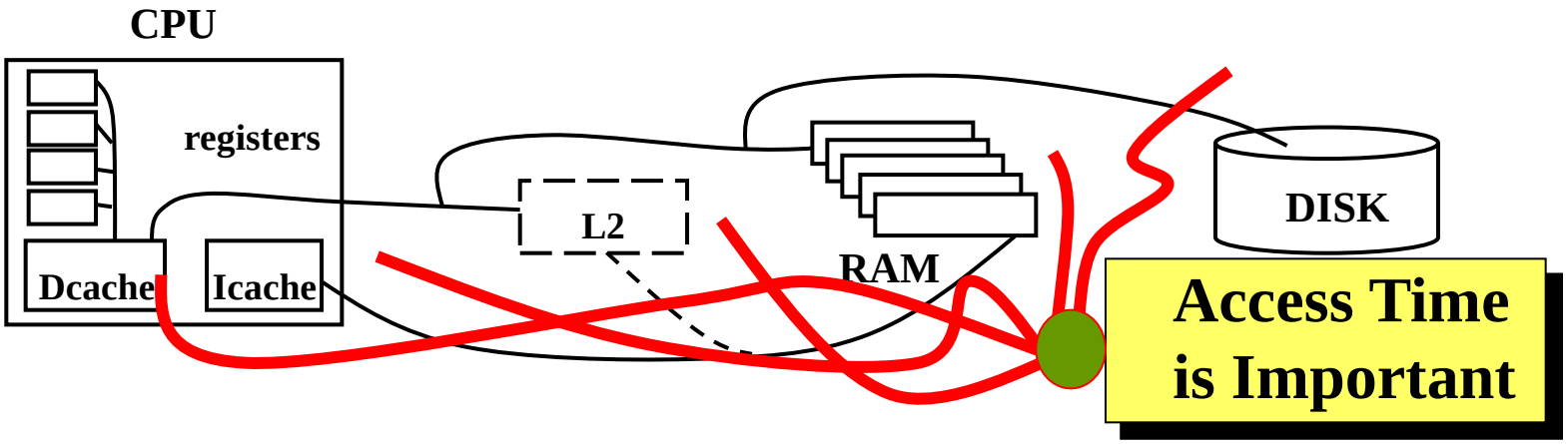
Source : http://www.intel.com ; Reference : [6]

Killer Apps of Tomorrow

- Workload convergence
  - The basic algorithms shared by these high-end workloads
- Platform implications
  - How workload analysis guides future architectures
- Programmer productivity
  - Optimized architectures will ease the development of software
- Call to Action
  - Benchmark suites in critical need for redress

The Memory sub-system : Access time

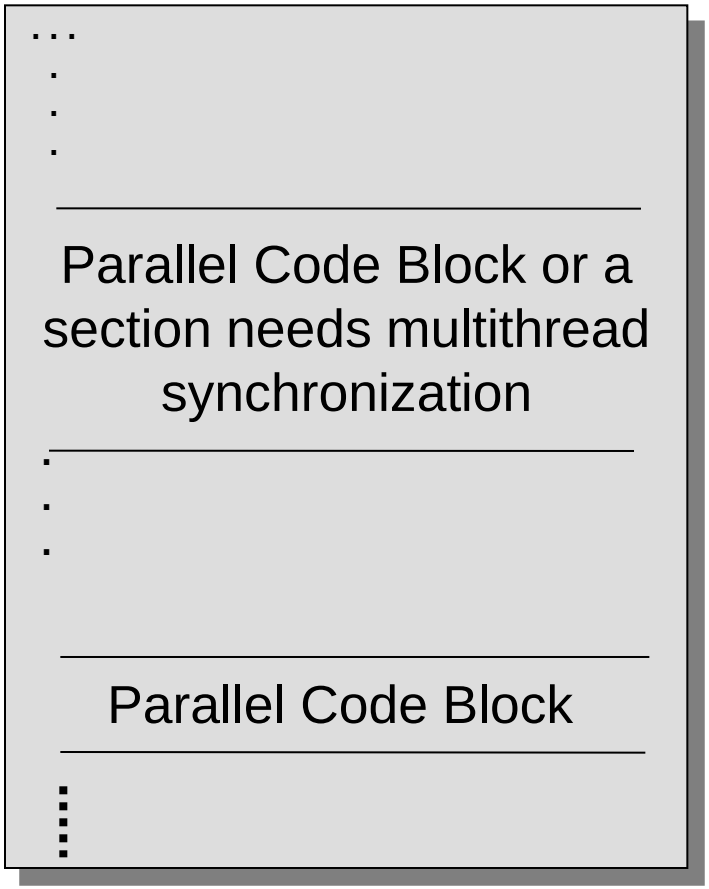
- A lot of time is spent accessing/storing data from/to memory. It is important to keep in mind the relative times for each memory types:



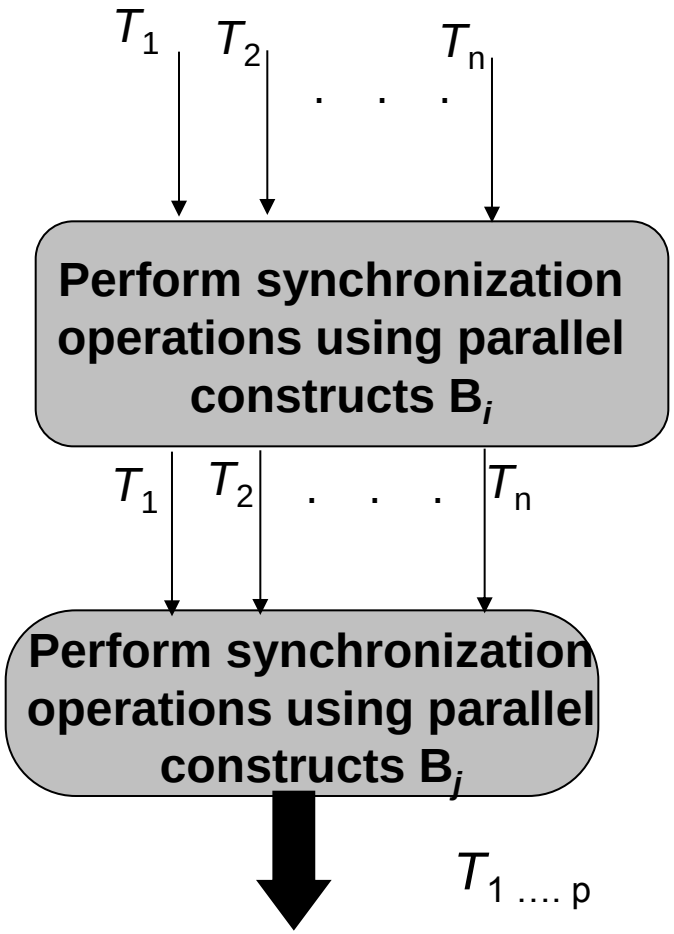
- **Approximate access times**
  - CPU-registers: 0 cycles (that's where the work is done!)
  - L_1 Cache: 1 cycle (Data and Instruction cache). Repeated access to a cache takes only 1 cycle
  - L_2 Cache (static RAM): 3-5 cycles?
  - Memory (DRAM): 10 cycles (Cache miss);
  - 30-60 cycles for Translation Lookaside Buffer (TLB) update
  - Disk: about 100,000 cycles!
  - connecting to other nodes - depending on network latency

Operational Flow of Threads for an Application

Implementation Source Code



Operational Flow of Threads



Source : Reference : [6]

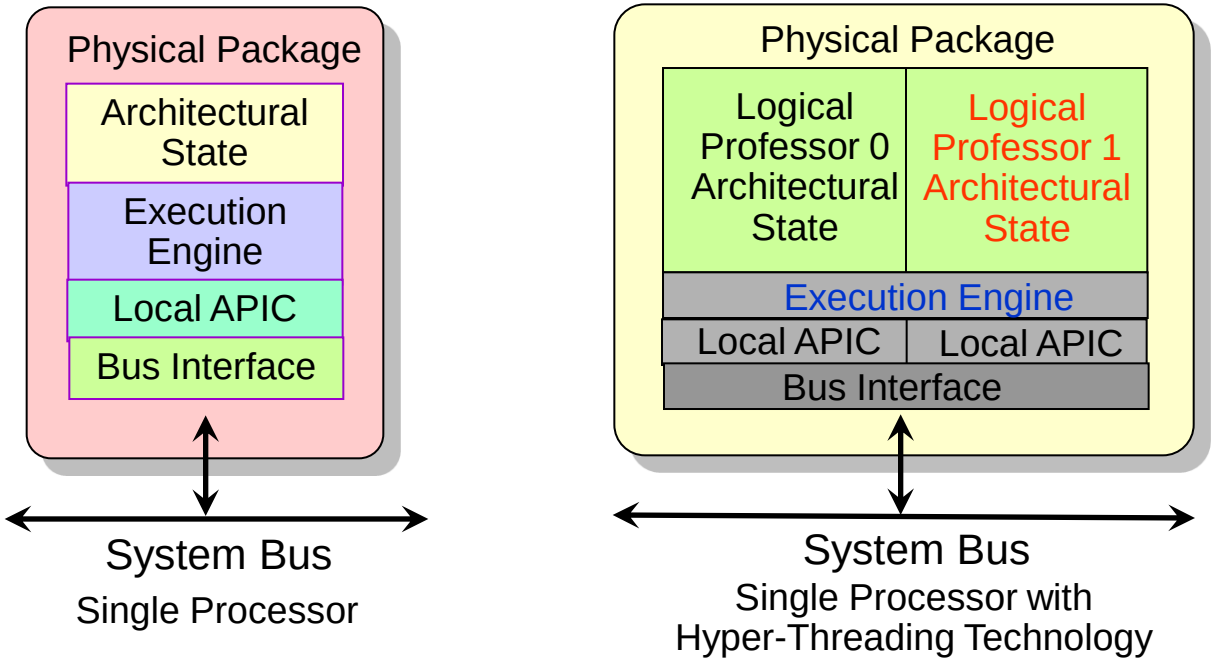
Multi Core : Programming Issues

- Out of Order Execution
- Preemptive and Co-operative Multitasking
- SMP to the rescue
- Super threading with Multi threaded Processor
- Hyper threading the next step (Implementation)
- Multitasking
- Caching and SMT

Source : http://www.intel.com ; Reference : [6], [29], [31]

Hyper-threading : Partitioned Resources

- Hyper-threading (HT) technology is a hardware mechanism where multiple independent hardware threads get to execute in a single cycle on a single super-scalar processor core.

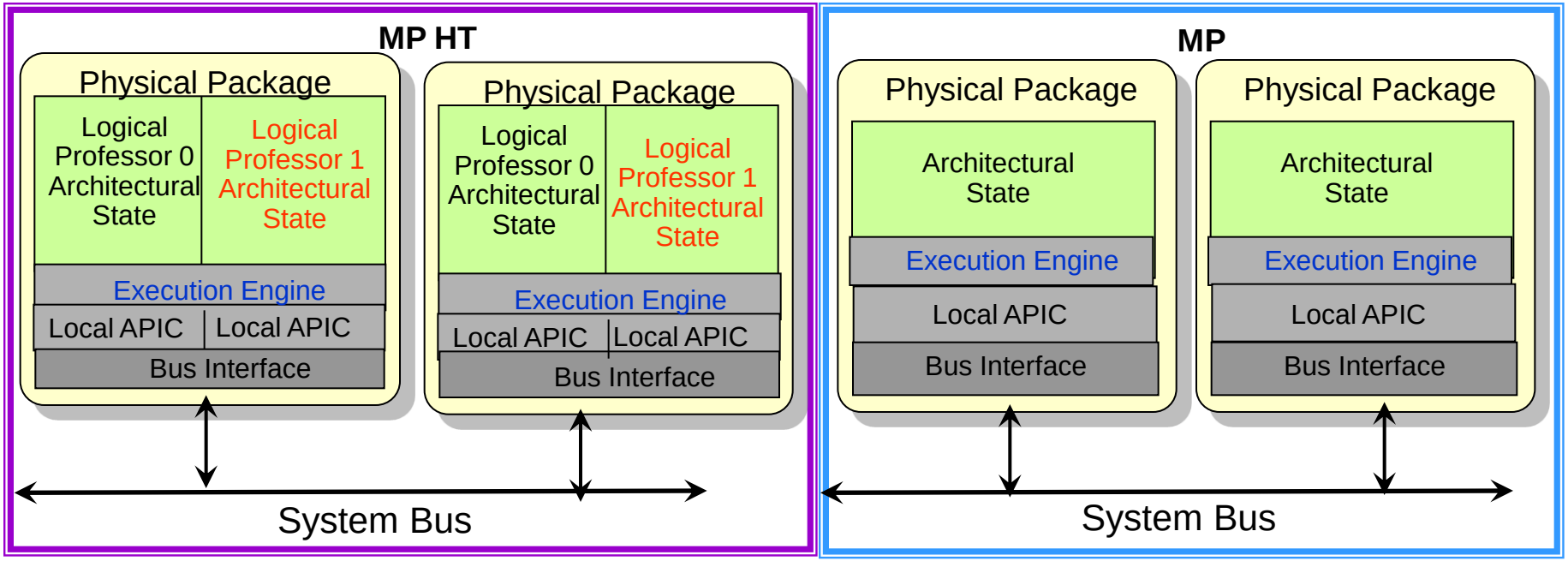


Single Processor System without Hyper-Threading Technology and Single Processor System with Hyper-Threading Technology

Source : http://www.intel.com ; Reference : [6], [10], [19], [23], [24],[29], [31]

Hyper-threading Technology

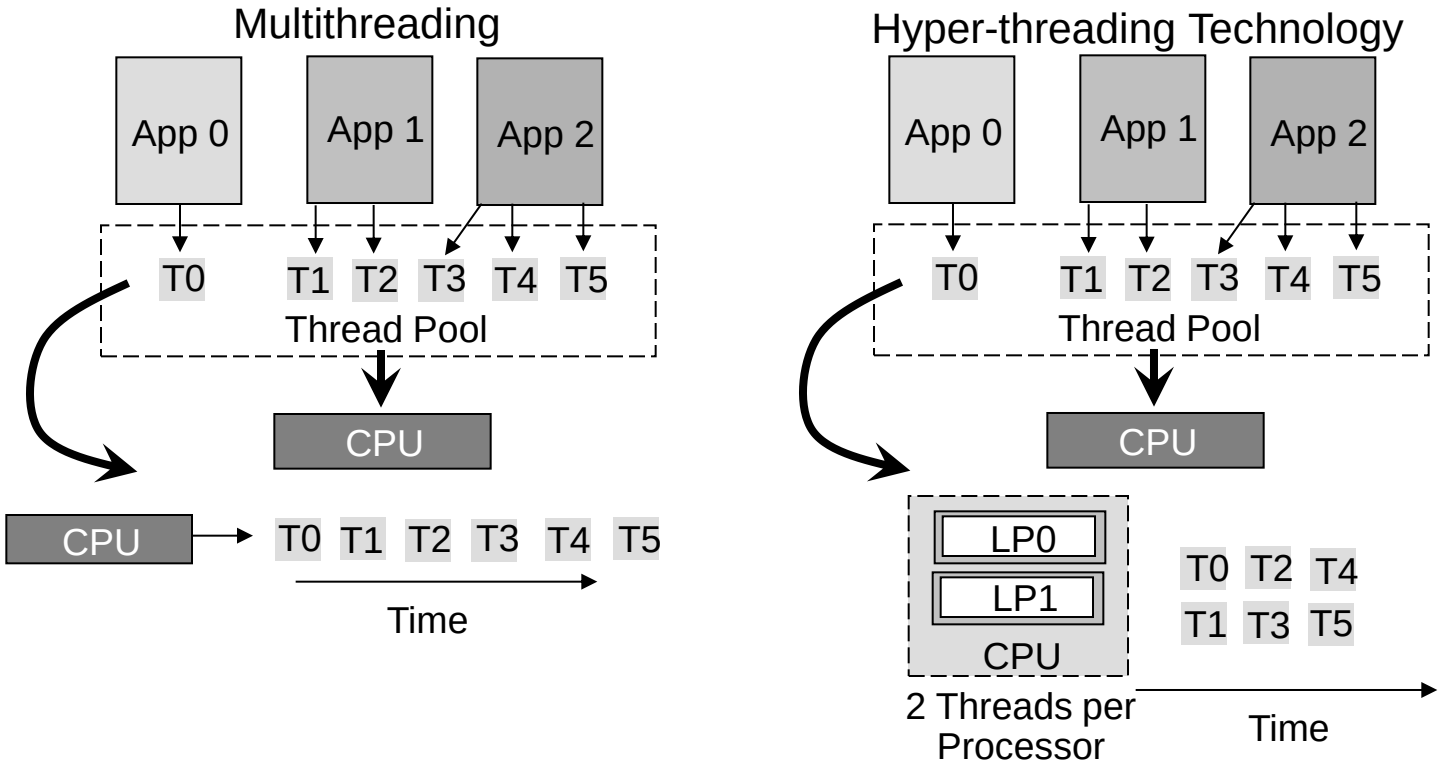
- Multi-processor with and without Hyper-threading (HT) technology



Source : http://www.intel.com ; Reference : [6], [10], [19], [23], [24],[29], [31]

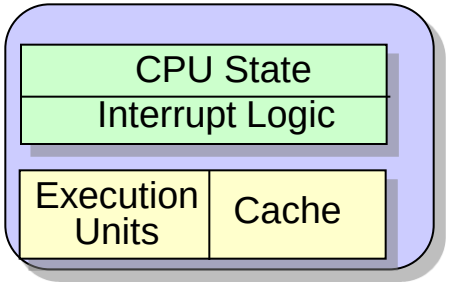
Multi-threaded Processing using Hyper-Threading Technology

- Time taken to process *n* threads on a single processor is significantly more than a single processor system with HT technology enabled.

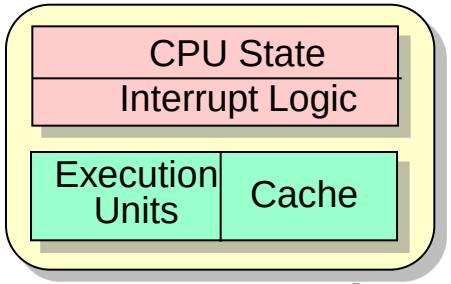


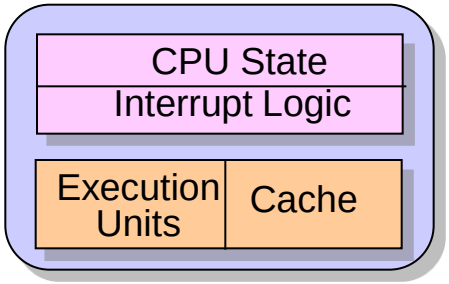
Source : http://www.intel.com ; Reference : [6], [29], [31]

Simple Comparison of Single-core, Multi-processor, and multi-Core Architectures

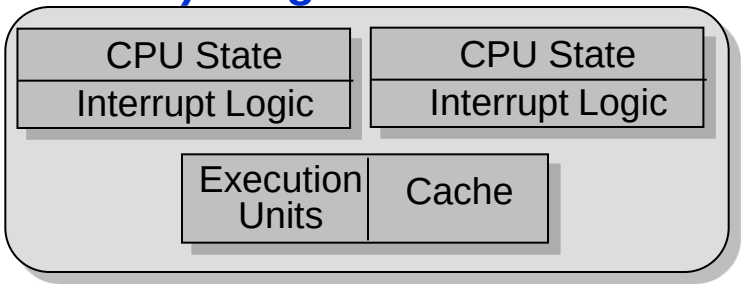


A) Single Core

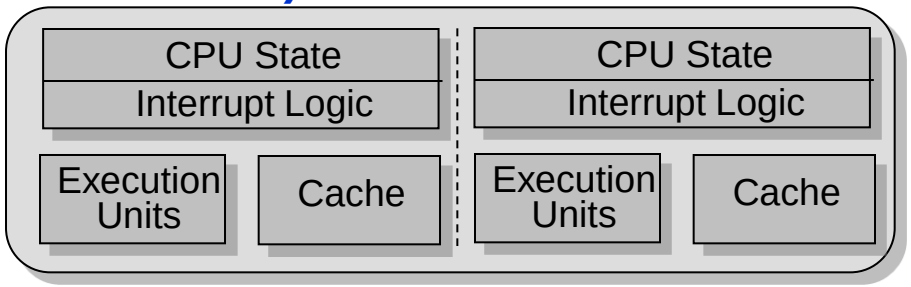




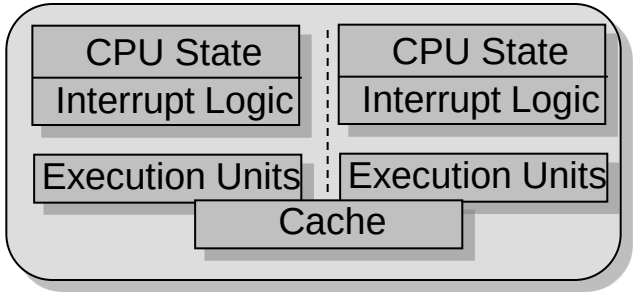
B) Multi Processor



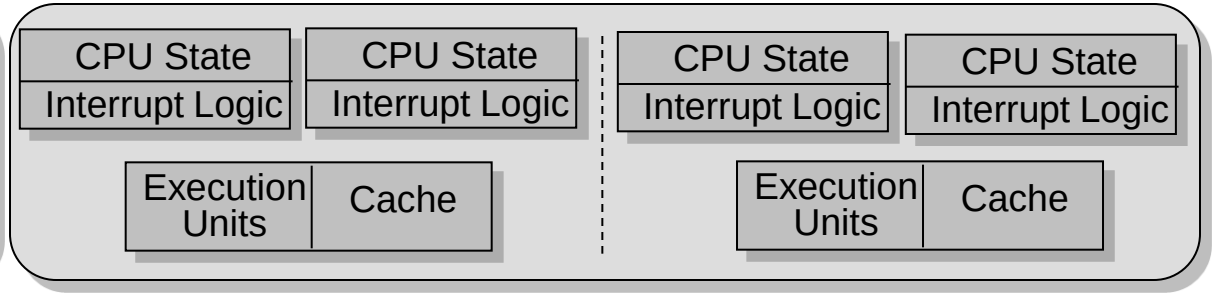
C) Hyper-Threading Technology



D) Multi-core



E) Multi-core with Shared Cache



F) Multi-core with Hyper-threading Technology

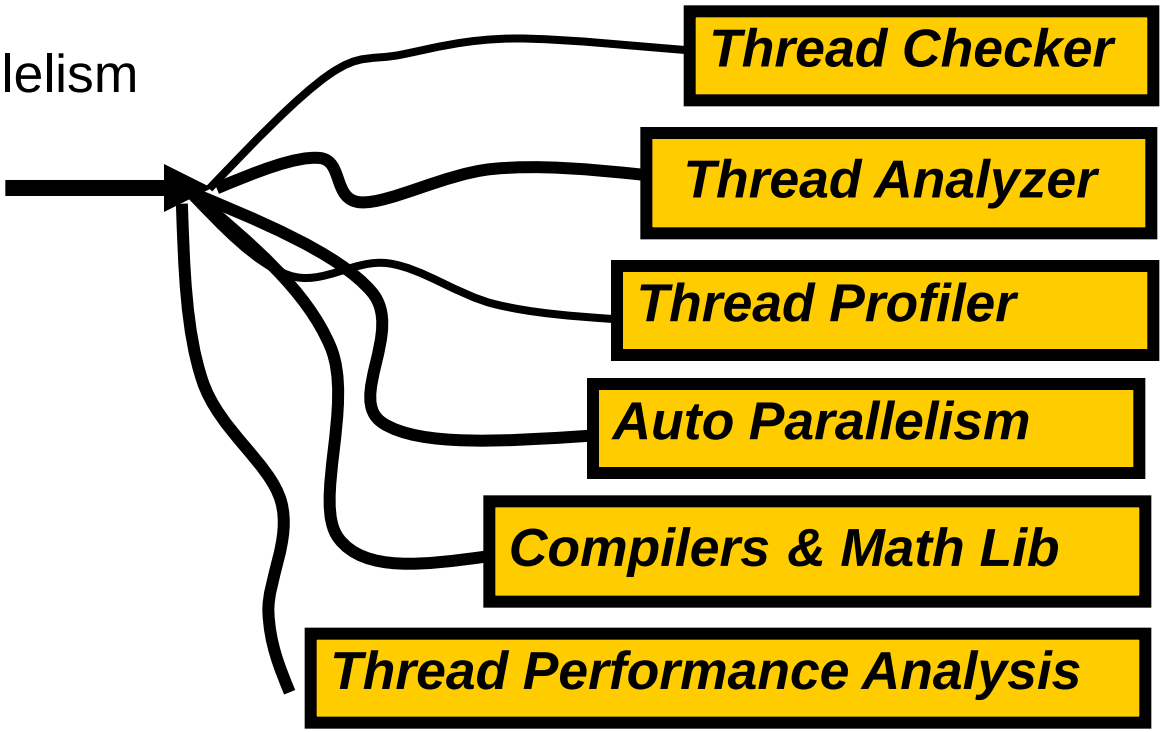
Source : http://www.intel.com ; Reference : [4],[6], [29], [31]

Multi Core Programming Tools

Intel Programming Tools : Intel Thread Building Blocks

Source : http://www.intel.com ; Reference : [6]

- Performance
- Tools to Discover Parallelism
- Use of Math Libraries
- Measure Overheads of Threads
- Hardware Counters



Programming Multicore Processors

- Explicit Parallel Programming
  - Thread-based Programming Models.
  - Data Parallel Programming Models
  - Stream Programming Models
- Automatic Parallelization
  - Features of Most compliers for SMP systems, but currently see very little practical use
  - Polyhedral framework for dependencies and loop transformations – enabling composition of complex transformations over multiple statements.

Spawning Threads

- Initialize Attributes (pthread_attr_init)
  - Default attributes OK
- Put thread in system-wide scheduling contention
  - `pthread_attr_setscope(&attrs, PTHREAD_SCOPE_SYSTEM);`
- Spawn thread (pthread_create)
  - Creates a thread identifier
  - Need attribute structure for thread
  - Needs a function where thread starts
  - One 32-bit parameter can be passed (void *)

Source : Reference : [4],[6], [29]

Thread Spawning Issues

- How does a thread know which thread it is? Does it matter?
  - Yes, it matters if threads are to work together
  - Could pass some identifier in through parameter
  - Could contend for a shared counter in a critical section
  - `pthread_self()` returns the thread ID, but doesn't help.
- How big is a thread's stack?
  - By default, not very big. (What are the ramifications?)
  - `pthread_attr_setstacksize()` changes stack size

Join Issues

- Main thread must join with child threads (`pthread_join`)
  - Why?
  - Ans: So it knows when they are done.
- `pthread_join` can pass back a 32-bit value
  - Can be used as a pointer to pass back a result
  - What kind of variable can be passed back that way? Local? Static? Global? Heap?

Thread Pitfalls

- Shared data
  - 2 threads perform A = A + 1

Thread 1:	Thread 1:
1) Load A into R1	1) Load A into R1
2) Add 1 to R1	2) Add 1 to R1
3) Store R1 to A	3) Store R1 to A

  - Mutual exclusion preserves correctness
    - Locks/mutexes
    - Semaphores
    - Monitors
    - Java “synchronized”

- False sharing
  - Non-shared data packed into same cache line

```
int thread1data;
int thread1data;
```

  - Cache line ping-pongsbetween CPUs when threads access their data

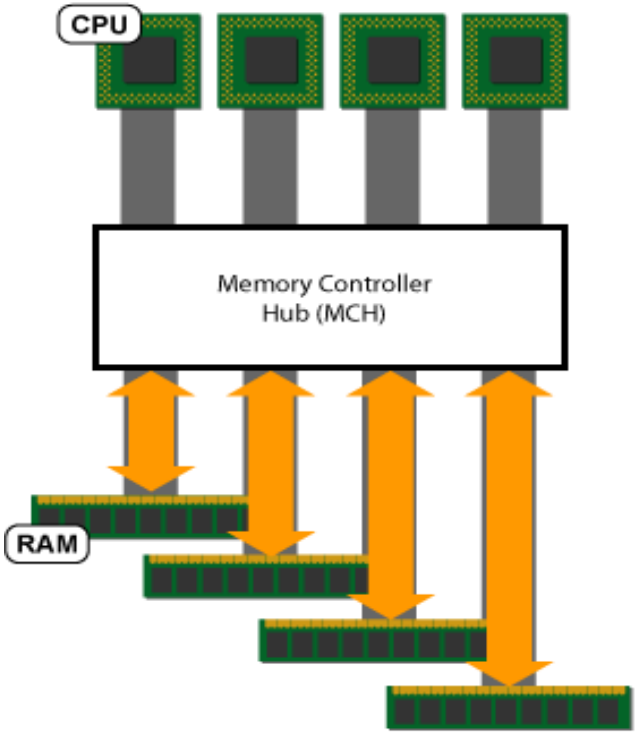
- Locks for heap access
  - malloc() is expensive because of mutual exclusion
  - Use private

Java Threads

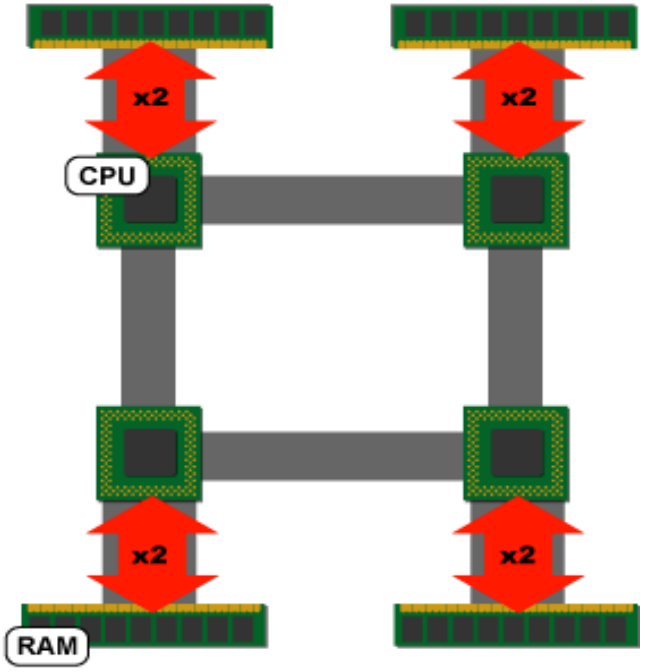
- Threading and synchronization built in
- An object can have associated thread
  - Subclass Thread or Implement Runnable
  - “run”method is thread body
  - “synchronized”methods provide mutual exclusion
- Main program
  - Calls “start”method of Thread objects to spawn
  - Calls “join”to wait for completion

Multi Cores Today

The Role of Intelligent Design : Multi-Core Processors Intel Quad Core (Clovertown) Server Processor with *Blackford* Chipset [2007]



INTEL'S FOUR-SOCKET PLATFORM



AMD'S FOUR-SOCKET PLATFORM

Source : http://www.intel.com ; http://www.amd.com

Part-I I: Intel

Memory Performance of Dual Core Systems

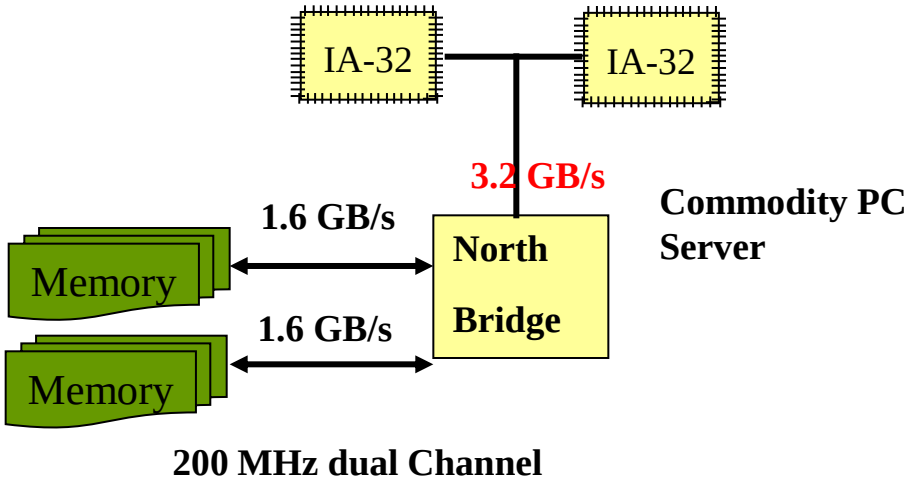
- Latency from different levels of Memory
- Bandwidth from different levels of Memory
- Stream Benchmark
- Prefetch Streams – Effective Bandwidth
- Serial and parallel dot-product (dot-product)
- Matrix-vector Multiplication (DAXPY) loop
- In-direct dot product
- Remote versus local memory

Source : http://www.intel.com ; Reference : [6]

Commodity PC -Server

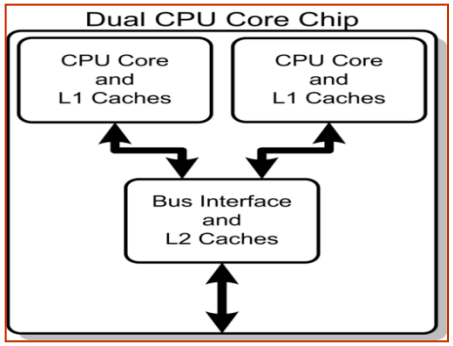
Memory Architecture for dual processor system

- Shared Bus Micro Architecture
  Intel IA-32 processors – Incorporates two processors on a single motherboard that shares a common Northbridge and Memory DIMMS
- Share the 400 MHz front-side bus (FSB)
  3.2 GB /s Bandwidth

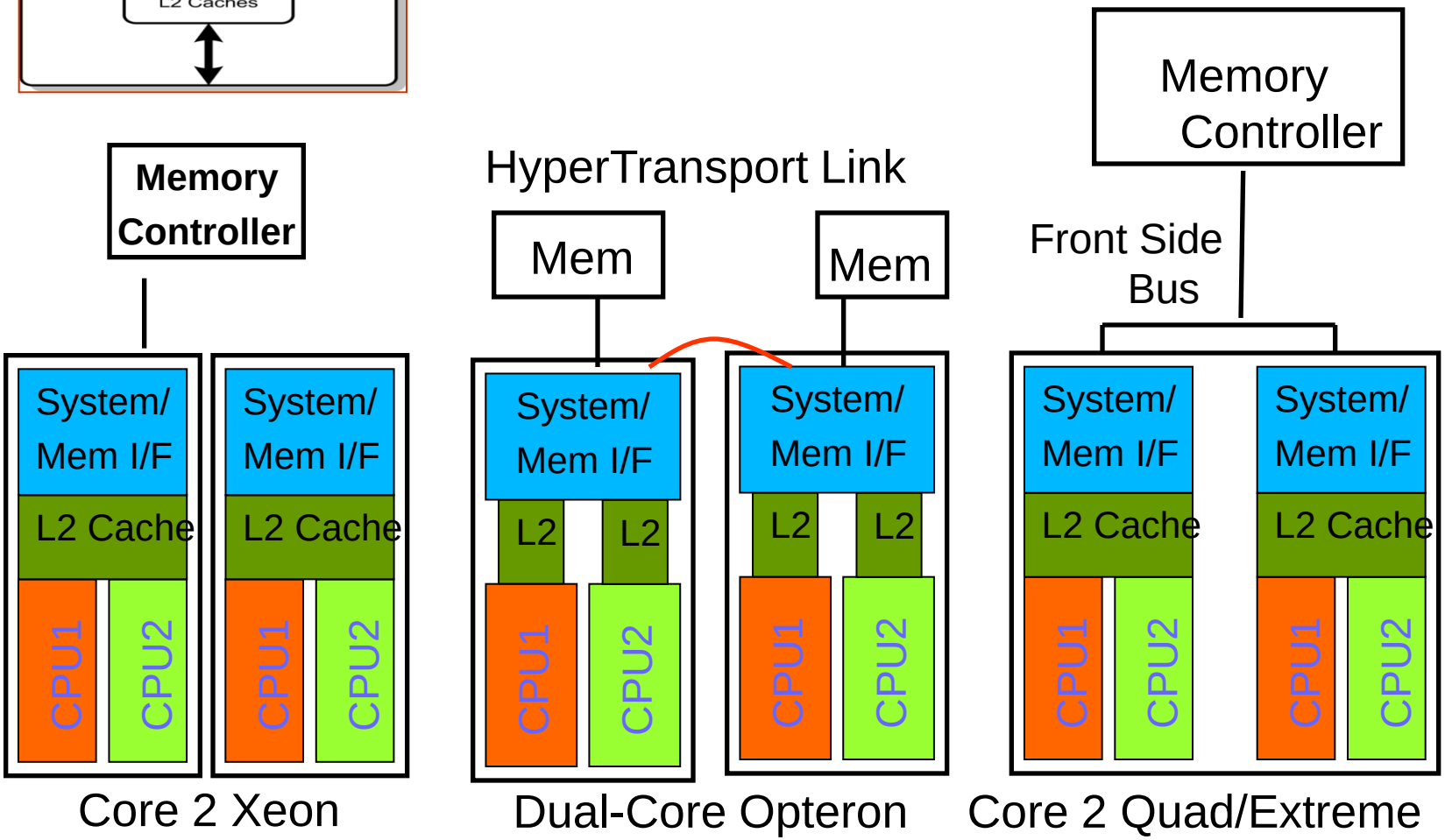


Source : http://www.intel.com ; Reference : [6]

Multi Cores Today



Source : http://www.intel.com; http://www.amd.com



The Role of Intelligent Design in the Evolution of Multi-Core Processors

- Intel's First Dual-Core Designs - Performance improvement
  - Dual Independent Front Side Buses (FSBs) for each CPU Socket (Enhancement of Single Core)
    - FSB Shared by the both Cores
    - Interprocessor Cache Snooping (Ensure Coherency of Cached data must be performed over the FSDB places on incremental load on FSB

  source : http:/www.intel.com

- AMD Opteron – Dual Core; source : http:/www.amd.com
- Sun Micro Systems – Niagara processors - UltraSPARC1

  source : http:/www.sun.com

The Role of Intelligent Design in the Evolution of Multi-Core Processors

- **Intel Quad Core** : Quad Processors demand more Memory Bandwidth – Starving for Memory Bandwidth
  - Cache Snooping over the Front Side Bus (FSB)
  - Contention for FSB Access
  - Overwork Memory Controller
  - Performance is marginally better in most Applications Significant improvement in some other Applications

source : http:/www.intel.com

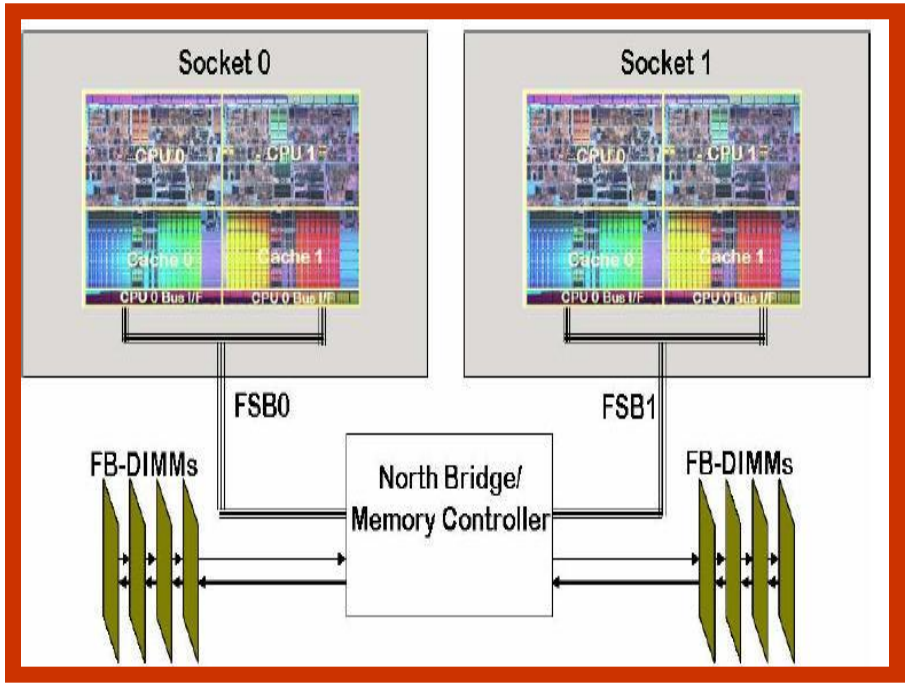
Memory Performance of Dual Core Systems

- The **Intel Xeon** Memory architecture – Single Core – Good Memory Bandwidth from all level of the memory hierarchy.
- Performance of shared bus memory architecture - when executing two memory intensive processes in parallel on a dual-processors node.
- The effect of memory contention, the resulting degradation in performance, is especially bad for random or strided memory access.
- Effect of entire FSB Bandwidth can be utilized
- Good Scalability of the memory bandwidth leads to efficient use of second CPU in a dual processor node.

source : http:/www.intel.com

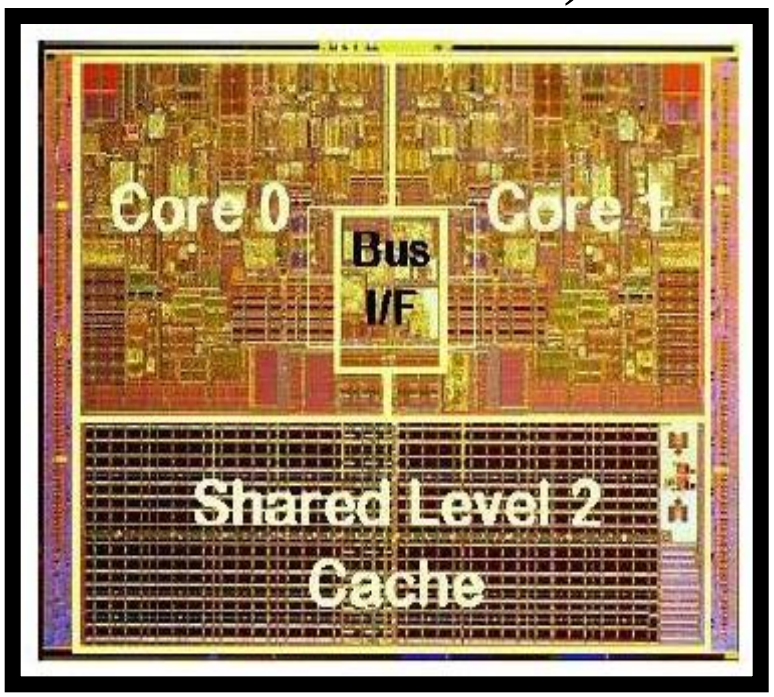
Multi-Core Processors

Intel Dual Core (Dempsey with Blackboard Chipset)



- Common L2 Cache
- Shared bus interface

Intel Dual Core (Yonah Mobile Processor)



- Inter Core Sharing in Dual-Core Architecture

Source : http://www.intel.com ;

Memory Performance of Dual /Quad Core Systems

- The Intel's First Dual Core arena : Smithfield, Paxville, Demsey & Presier
- Communication between the cores must be accomplished over the external front side bus that connects both the core of the north bridge.

Dempsey with Blackford chipset – that provides a separate front-side bus for each CPU socket in the system.

- Good Scalability of the memory bandwidth leads to efficient use of second CPU in a dual processor node.

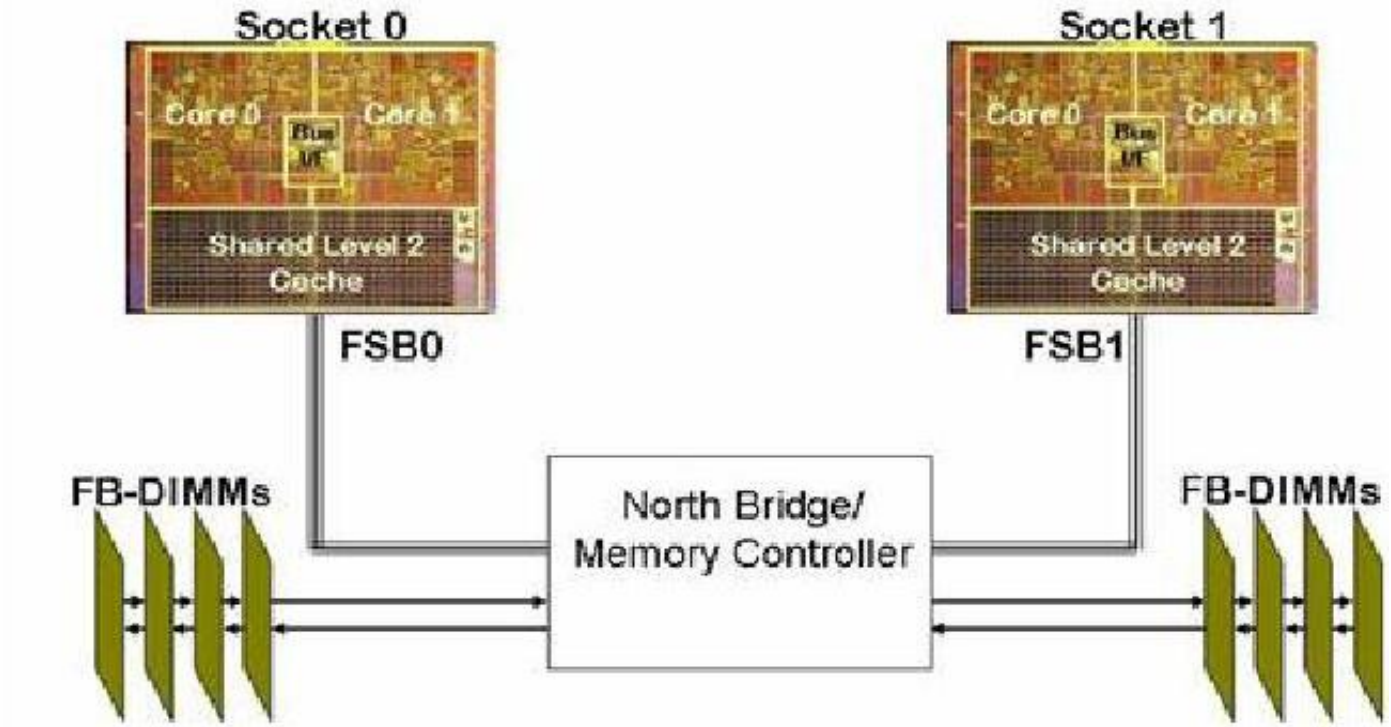
Yonah Intel's Dual-Core (yonah) Mobile Processor –

- Two cores share a common L2 Cache, they never need to go off-chip to ensure cache coherency.
- Eliminates front-side bus (FSB) contention issues

source : http:/www.intel.com

Multi-Core Processors

Intel Dual Core (Woodcrest) Server Processor with *Blackford* Chipset : Improves FSB Speed.



source : http:/www.intel.com

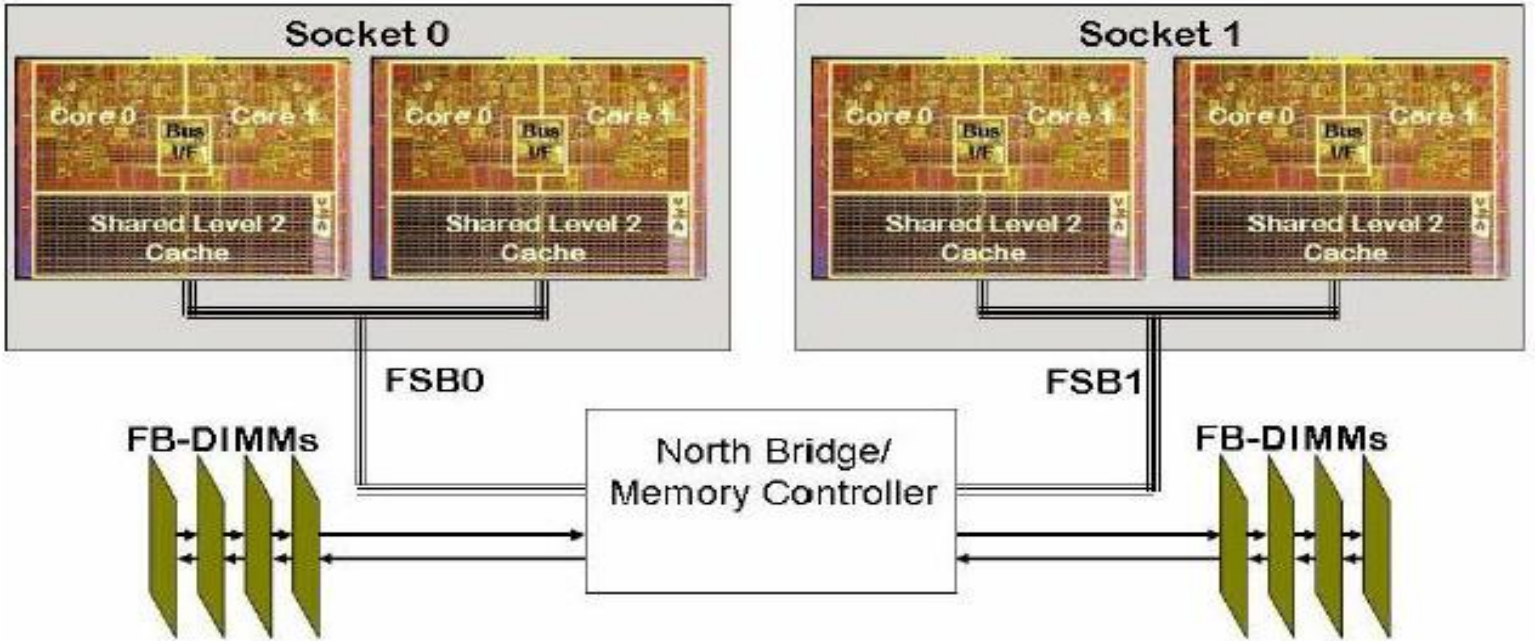
Memory Performance of Dual /Quad Core Systems

- Intel Dual Core (Woodcrest) Server Processor with Blackford Chipset
- Communication between the cores must be accomplished over the external front side bus that connects both the core of the north bridge.
- Dempsey with Blackford chipset – that provides a separate front-side bus for each CPU socket in the system.
- Good Scalability of the memory bandwidth leads to efficient use of second CPU in a dual processor node.

source : http:/www.intel.com

Multi-Core Processors

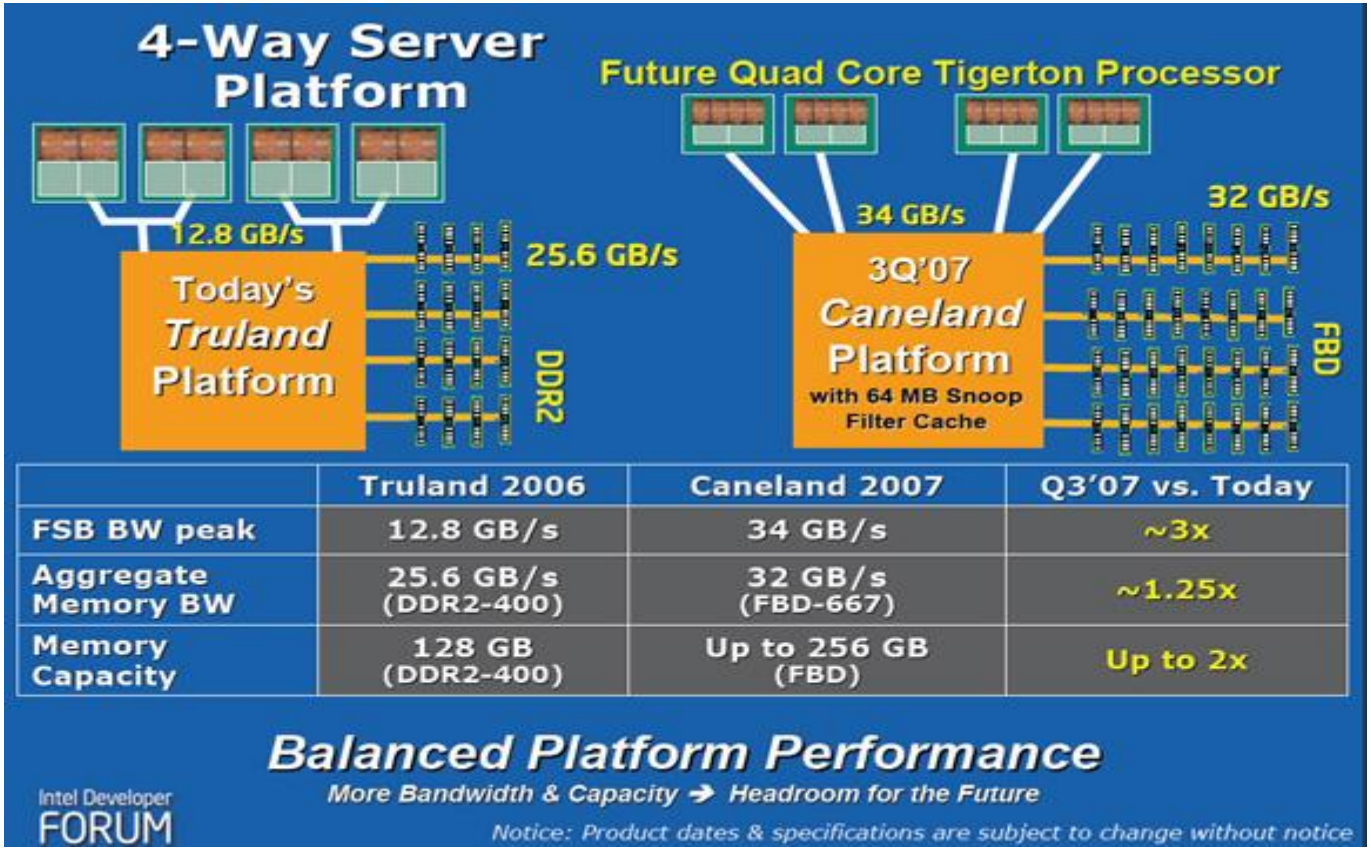
Intel Quad Core (Clovertown) Server Processor with *Blackford* Chipset : Clovertown will consist of two Woodcrest dice crammed into a single package.



source : http:/www.intel.com

Multi-Core Processors

The Role of Intelligent Design : Multi-Core Processors Intel Quad Core (Caneland) Server Processor with *Blackford* Chipset [2007]



	Truland 2006	Caneland 2007	Q3'07 vs. Today
FSB BW peak	12.8 GB/s	34 GB/s	~3x
Aggregate Memory BW	25.6 GB/s (DDR2-400)	32 GB/s (FBD-667)	~1.25x
Memory Capacity	128 GB (DDR2-400)	Up to 256 GB (FBD)	Up to 2x

source : http:/www.intel.com

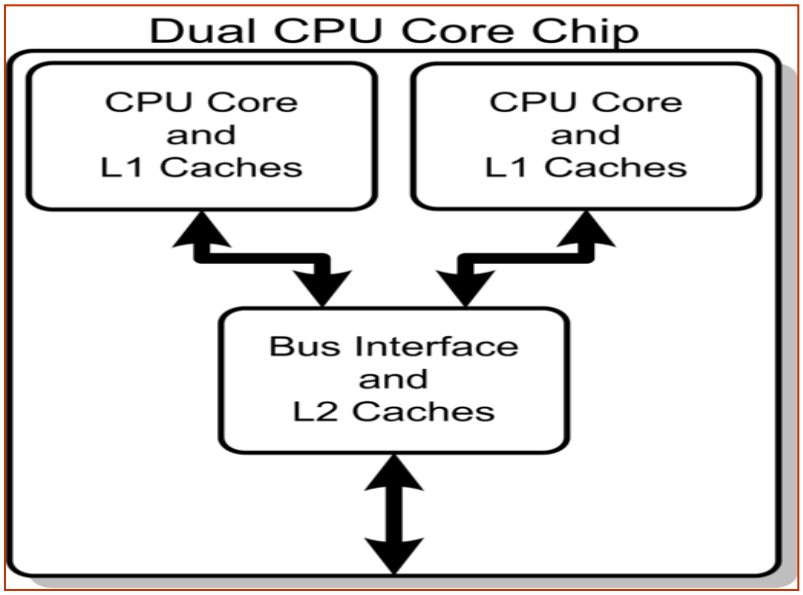
Part-III: AMD

AMD : Introducing Multi-core technology

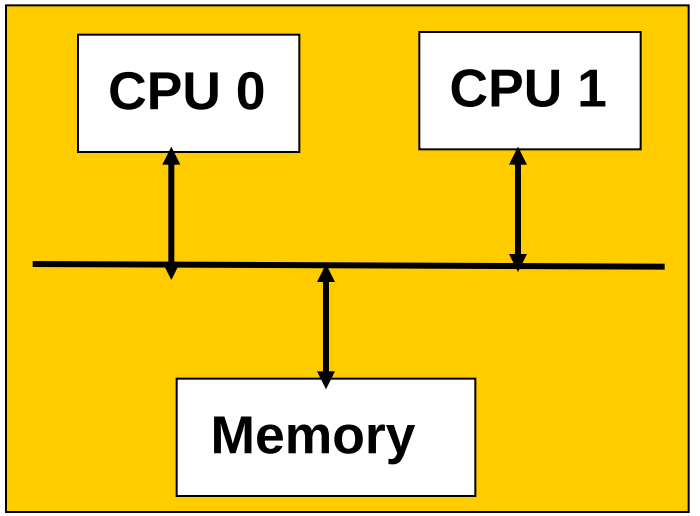
- AMD introduced the first multi-core technology for x86 based servers
- Multi-core processors represent a major evolution in computing technology. Placing tow or more powerful computing cores on a single processor opens up a world of important new possibilities.
- The AMD Platform is leading industry to pervasive 64 bit computing
- AMD Opteron processor –server and workstation

source : http:/www.amd.com

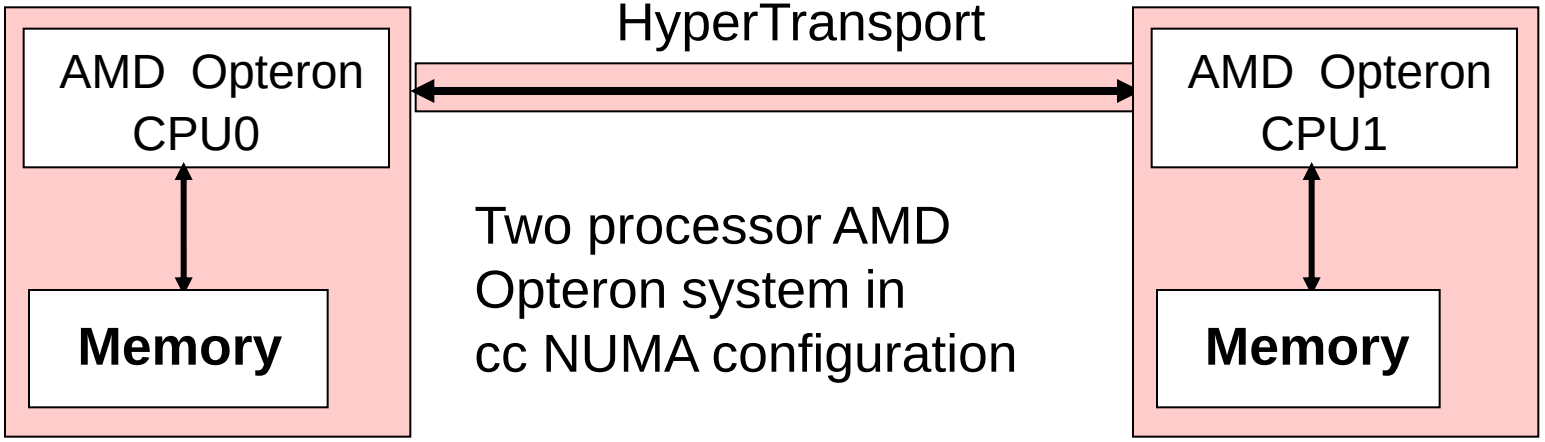
Multi Cores Today



Two processor Dual Core

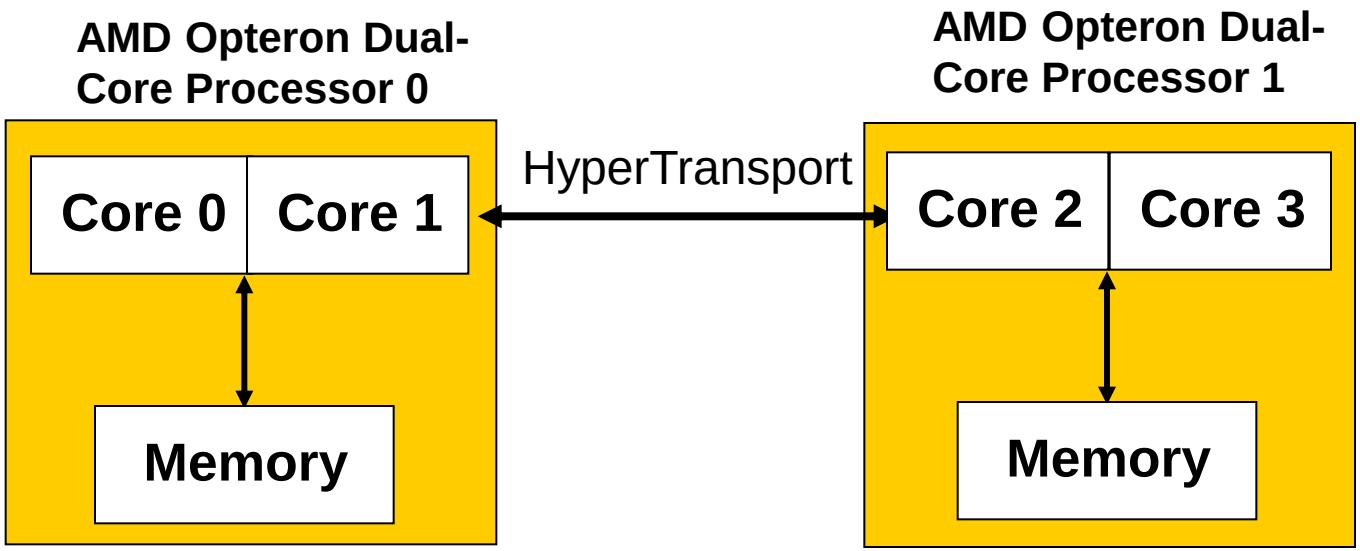


Simple SMP Block Diagram for a two processors



Two processor AMD Opteron system in cc NUMA configuration

AMD Multi Cores



Dual-Core AMD Opteron Processor configuration

- AMD : Cache-Coherent nonuniform memory access (ccNUMA)
  - Two or more processors are connected together on the same motherboard
  - In ccNUMA design, each processor has its own memory system.
  - The phrase 'Non Uniform Memory access' refers to the potential difference in latency

source : http:/www.amd.com

AMD Opteron : Hyper Transport technology

- HyperTransport technology is a high-speed bi-directional ,low latency point to point communication link
- Provides bandwidth interconnect between computing cores.
- AMD Opteron support up to three coherent HyperTransport links yielding up to 24.0 GB/s peak bandwidth per processor
- The AMD Opteron processor provides scalable architecture that next-generation performance.
- Integrated DDR Memory Controller : Increase application performance by dramatically reducing memory latency

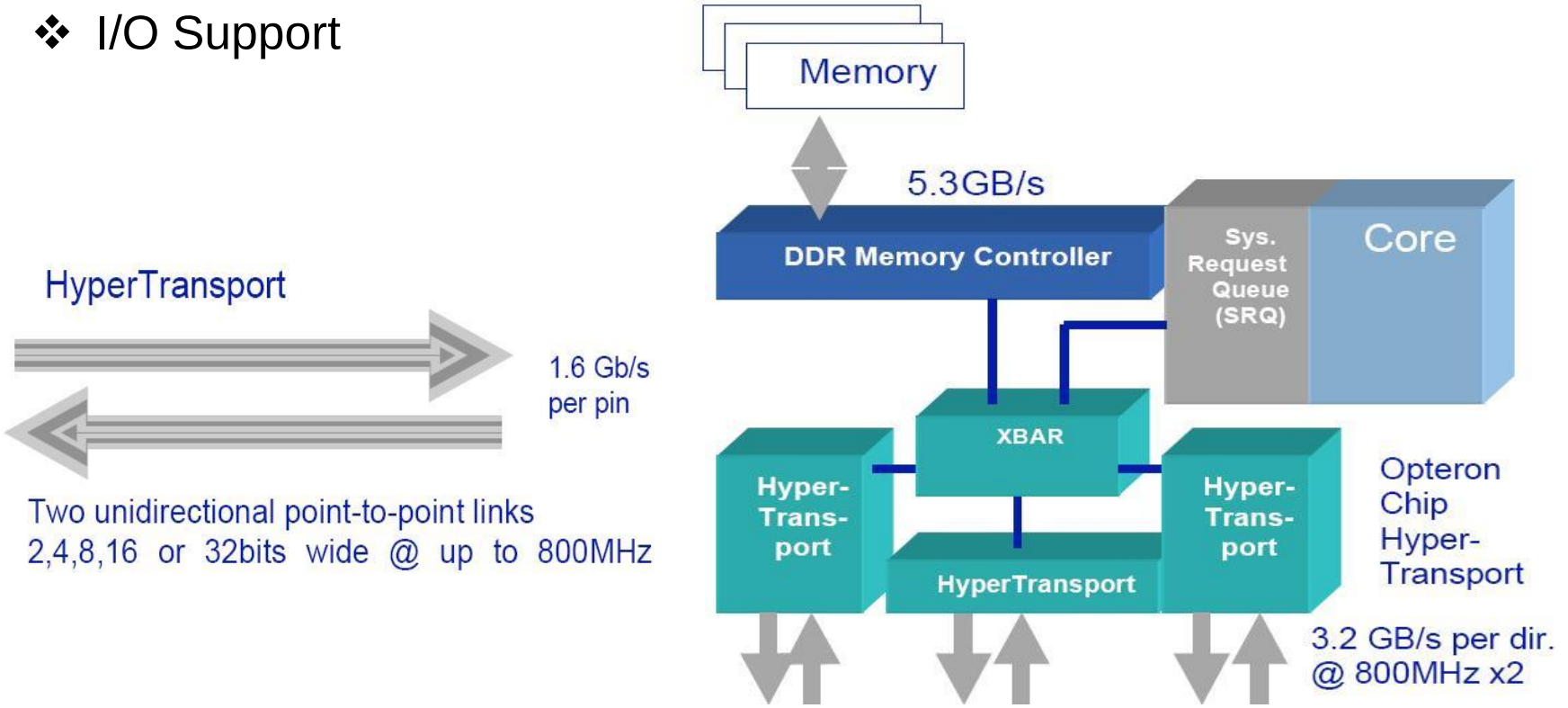
AMD : Scalable Memory Interconnect

- Memory access is optimized
  - First, Controllers are build into the chips, to create a direct path to each processor's locally attached memory
  - Scaling memory in coherency is achieved through a virtual chipset, interconnected via high speed transport layer, called **HyperTransport.**
  - Each processor can share its local memory and devices with other processors through Hyper Transport Links; but each processor retains a local, direct path to its attached memory.
  - It is NUMA-like architecture because of latencies when accessing local versus remote memory.

source : http:/www.amd.com

AMD : Scalable Memory Interconnect

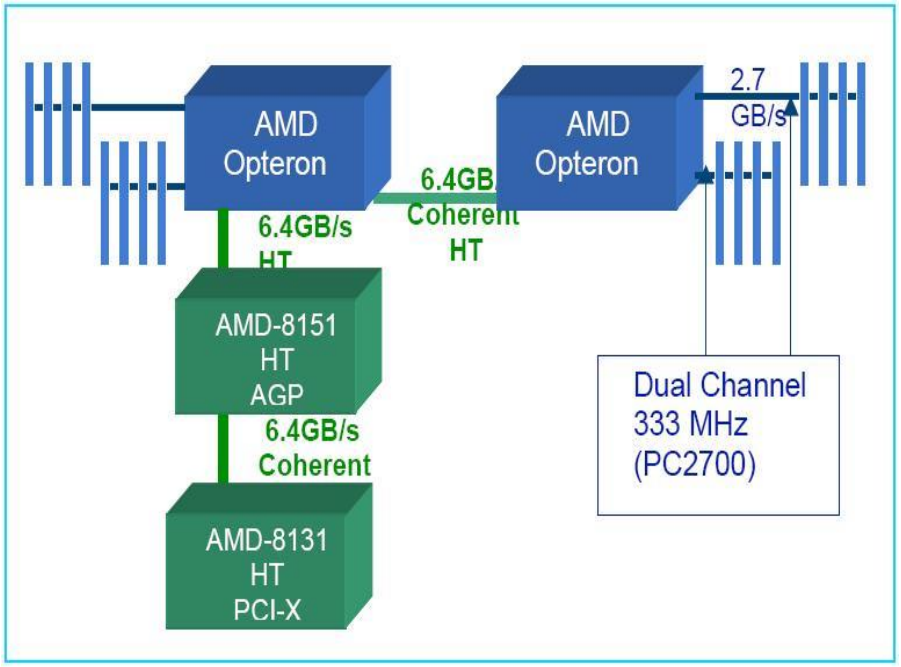
- Scale in Memory Bandwidth : Memory access is optimized
- I/O Support

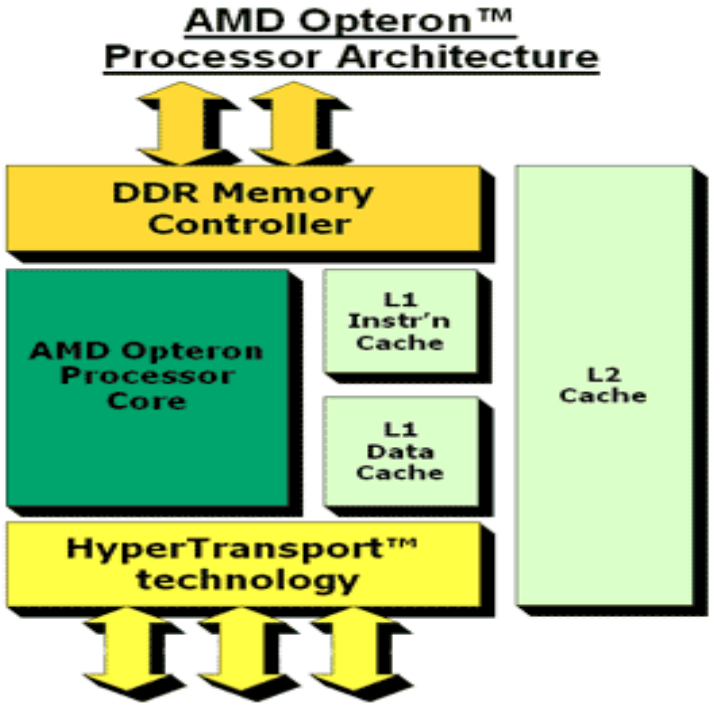


- Hyper Transport technology within Opteron micro-architecture

source : http:/www.amd.com

Block Diagram Of AMD Opteron Processor





source : http:/www.amd.com

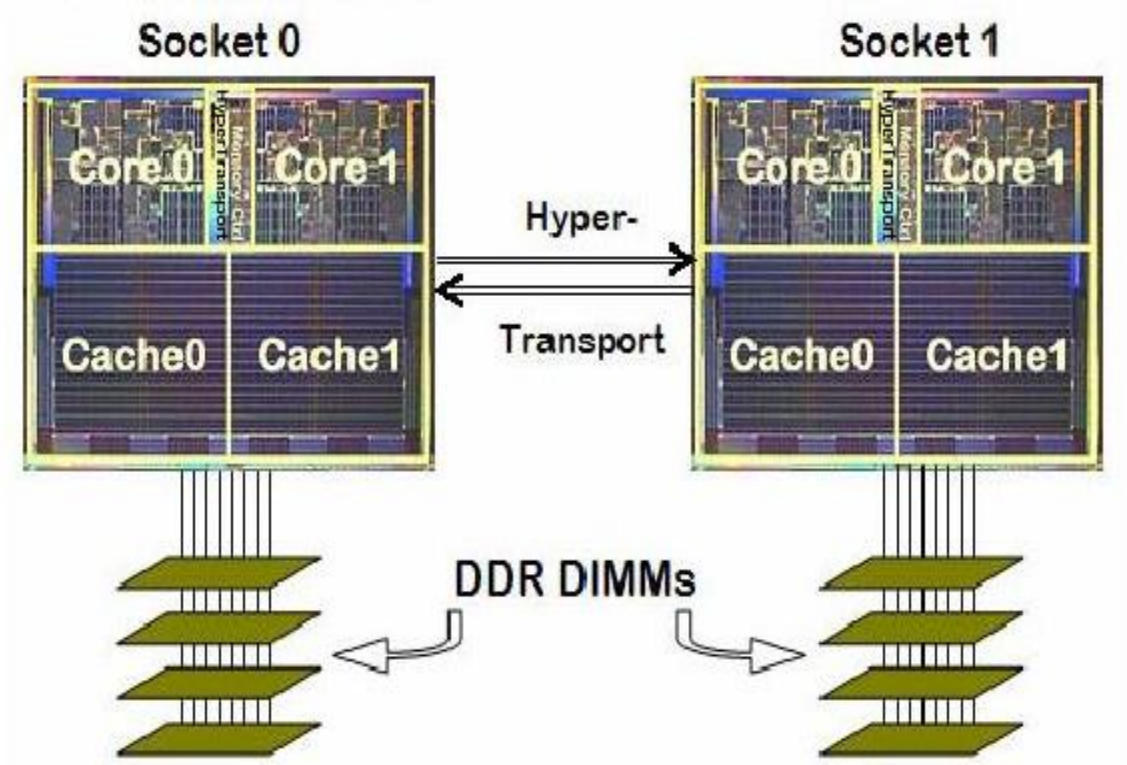
AMD Opteron Processor Key Architecture features

- I/O is directly connected to CPU for more balanced throughput and I/O
- CPUs are directly connected to each other allow linear symmetrical multiprocessing
- 128 bit wide integrated DDR DRAM memory controller capable of supporting up to eight registered DDR DIMMs per processor
- Allows end users to run their existing 32 bit application
- Designed to enable 64 bit computing
- Enable single architecture across 32 and 64 bit environment
- Memory is directly connected to the CPU optimizing performance

AMD Opteron Dual Core Processor

- AMD Dual-Core Opteron, Circa 2005

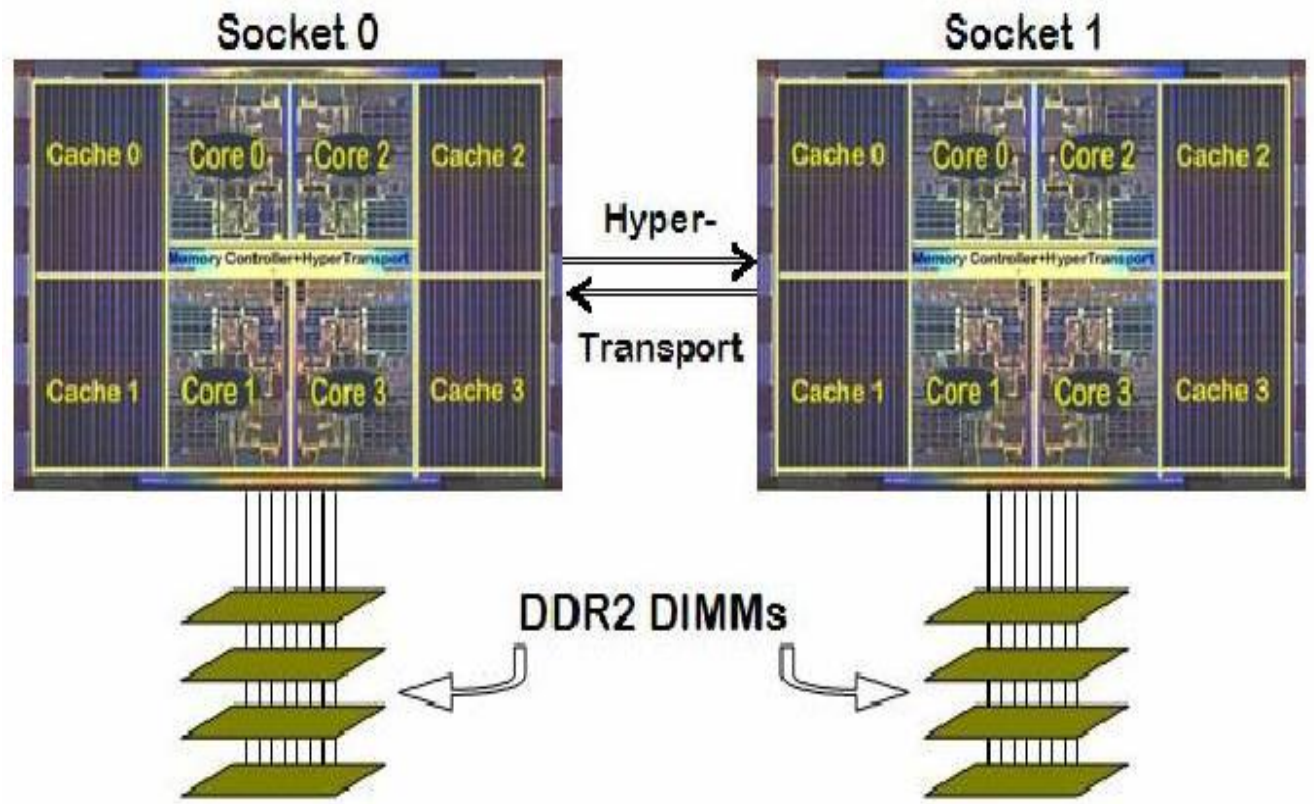
  Socket F used in the motherboard OEMs



source : http:/www.amd.com

AMD Opteron Quad Core Processor

- AMD Quad-Core Opteron, Circa 2007

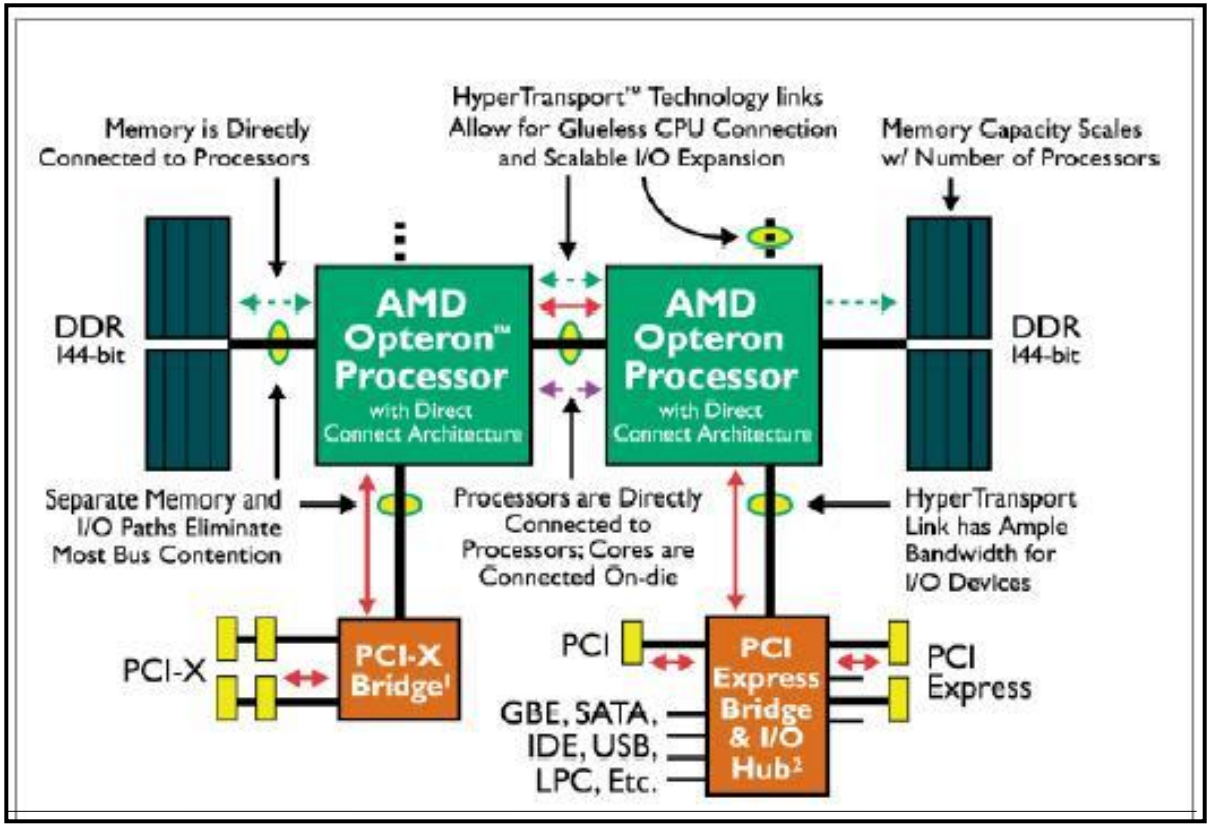


source : http:/www.amd.com

AMD Opteron : Direct Connect Architecture

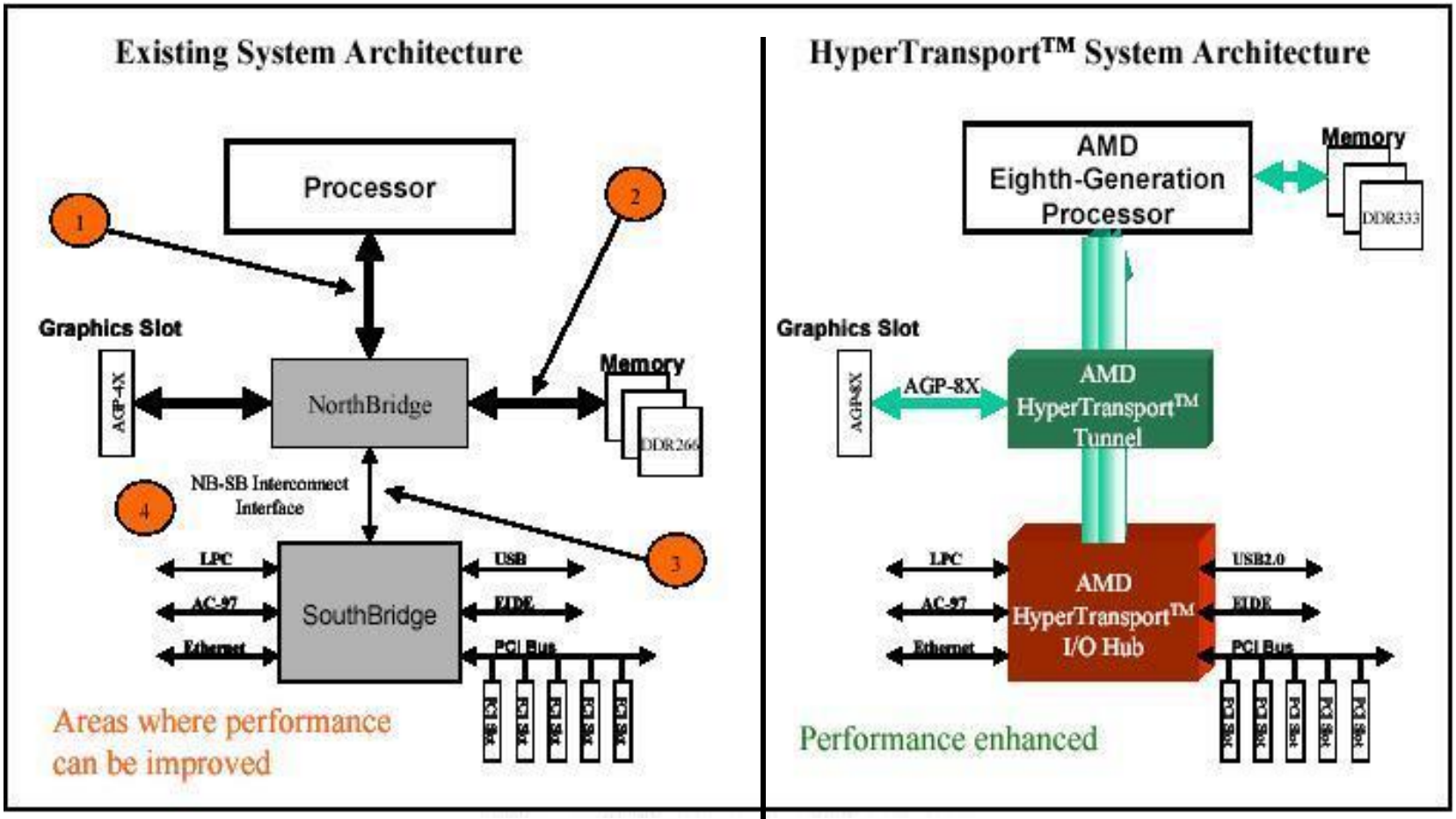
- MD64 with direct connect architecture can improve overall performance and efficiency
- No front side data buses, instead ,the processors, memory controller and i/o are directly connected to CPU and communicate at CPU speed
- Available with all types of AMD (Opteron, Athlon)

AMD Opteron: Direct Connect Architecture



source : http:/www.amd.com

AMD Opteron: Direct Connect Architecture



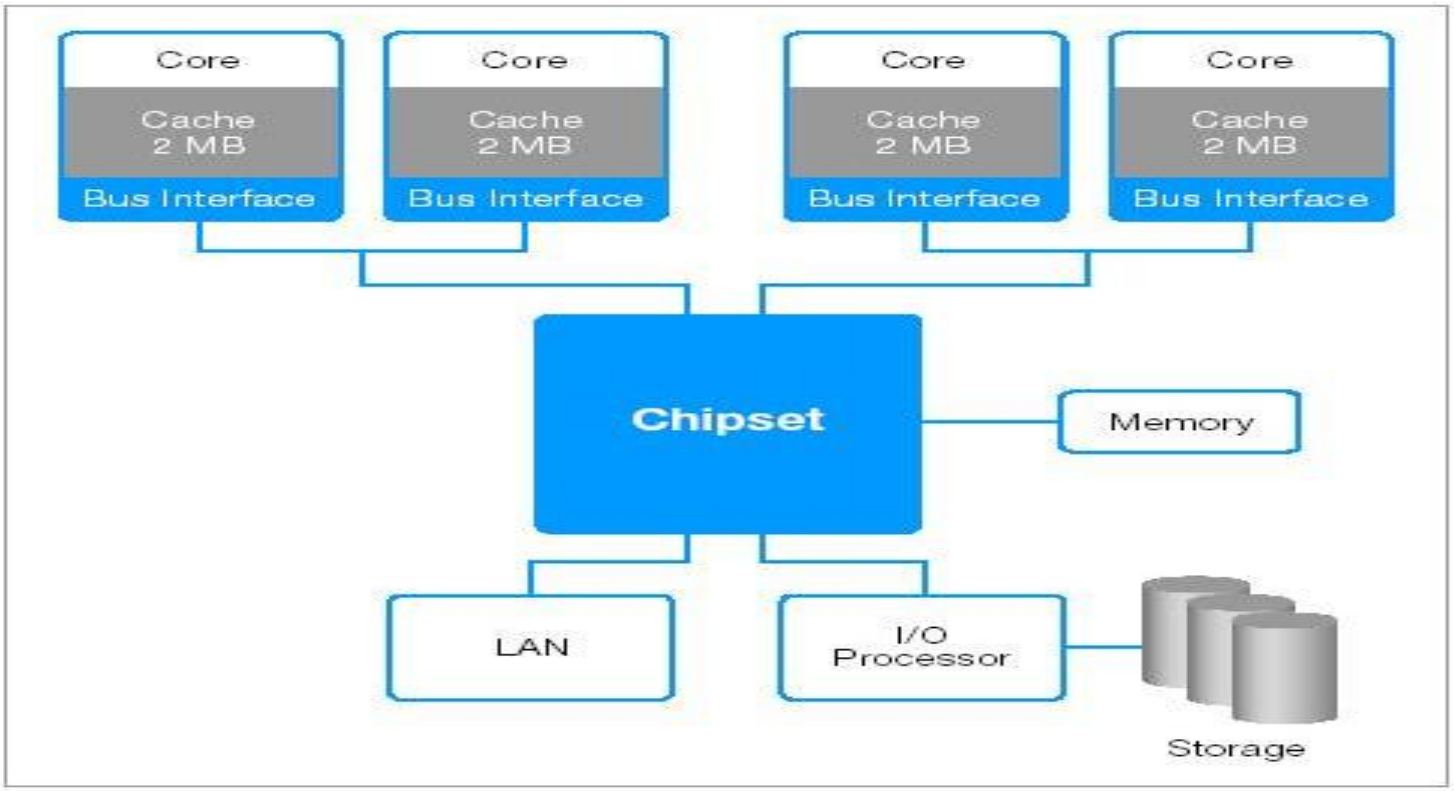
source : http:/www.amd.com

AMD Opteron : Processor Benefits

- AMD64 Core :
  - Enables simultaneous 32 and 64 bit computing
  - Eliminates the 4GB memory barrier imposed by 32-bit only systems
- HyperTransport Technology :
  - provides maximum peak bandwidth and reduce bottle neck
  - Directly connect CPU enabling scalability

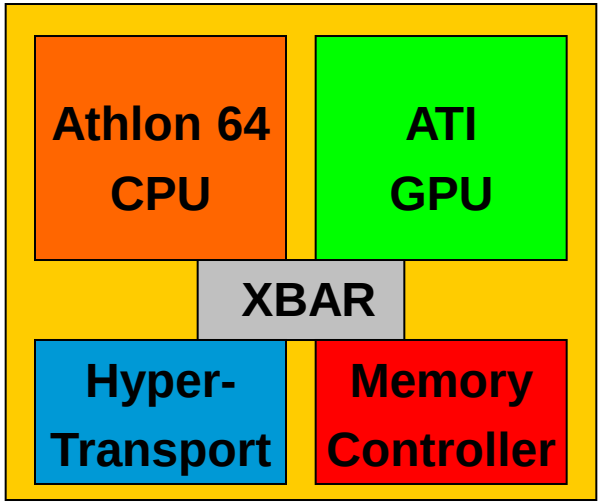
Multi Core Architecture : I/O Bottelnecks

- Balanced Architecture with Dual independent buses reduce I/O Bottlenecks



Future CMP Technology

- 8 cores soon
- Room for improvement
  - Multi-way caches expensive
  - Coherence protocols perform poorly
- Stream programming
  - GPU or multi-core
  - GPGPU.org for details



Possible Hybrid AMD Multi-Core Design

Summary

- An overview of Multi-Core Systems
- Examples of Multi-Core Systems
- Memory Performance Issues

References

1. Andrews, Grogory R. **(2000),** Foundations of Multithreaded, Parallel, and Distributed Programming, Boston, MA : Addison-Wesley

2. Butenhof, David R **(1997),** Programming with POSIX Threads , Boston, MA : Addison Wesley Professional

3. Culler, David E., Jaswinder Pal Singh **(1999),** Parallel Computer Architecture - A Hardware/Software Approach , San Francsico, CA : Morgan Kaufmann

4. Grama Ananth, Anshul Gupts, George Karypis and Vipin Kumar **(2003),** Introduction to Parallel computing, Boston, MA : Addison-Wesley

5. Intel Corporation, **(2003),** Intel Hyper-Threading Technology, Technical User's Guide, Santa Clara CA : Intel Corporation Available at : http://www.intel.com

6. Shameem Akhter, Jason Roberts **(April 2006),** Multi-Core Programming - Increasing Performance through Software Multi-threading , Intel PRESS, Intel Corporation,

7. Bradford Nichols, Dick Buttlar and Jacqueline Proulx Farrell **(1996),** Pthread Programming O'Reilly and Associates, Newton, MA 02164,

8. James Reinders, Intel Threading Building Blocks – (**2007**) , O'REILLY series

9. Laurence T Yang & Minyi Guo (Editors), (**2006**) *High Performance Computing - Paradigm and Infrastructure* Wiley Series on Parallel and Distributed computing, Albert Y. Zomaya, Series Editor

10. Intel Threading Methodology ; Principles and Practices Version 2.0 copy right **(March 2003),** Intel Corporation

References

11. William Gropp, Ewing Lusk, Rajeev Thakur **(1999),** Using MPI-2, Advanced Features of the Message-Passing Interface, The MIT Press..

12. Pacheco S. Peter, **(1992),** Parallel Programming with MPI, , University of Sanfrancisco, Morgan Kaufman Publishers, Inc., Sanfrancisco, California

13. Kai Hwang, Zhiwei Xu, (**1998**), Scalable Parallel Computing (Technology Architecture Programming), McGraw Hill New York.

14. Michael J. Quinn (**2004**), Parallel Programming in C with MPI and OpenMP McGraw-Hill International Editions, Computer Science Series, McGraw-Hill, Inc. Newyork

15. Andrews, Grogory R. **(2000),** Foundations of Multithreaded, Parallel, and Distributed Progrmaming, Boston, MA : Addison-Wesley

16. SunSoft. Solaris multithreaded programming guide. SunSoft Press, Mountainview, CA, **(1996),** Zomaya, editor. Parallel and Distributed Computing Handbook. McGraw-Hill,

17. Chandra, Rohit, Leonardo Dagum, Dave Kohr, Dror Maydan, Jeff McDonald, and Ramesh Menon, **(2001)**,Parallel Programming in OpenMP San Fracncisco Moraan Kaufmann

18. S.Kieriman, D.Shah, and B.Smaalders **(1995),** Programming with Threads, SunSoft Press, Mountainview, CA. 1995

19. Mattson Tim, **(2002),** Nuts and Bolts of multi-threaded Programming Santa Clara, CA : Intel Corporation, Available at : http://www.intel.com

20. I. Foster **(1995,** Designing and Building Parallel Programs ; Concepts and tools for Parallel Software Engineering, Addison-Wesley (1995)

21. J.Dongarra, I.S. Duff, D. Sorensen, and H.V.Vorst **(1999),** Numerical Linear Algebra for High Performance Computers (Software, Environments, Tools) SIAM, 1999

References

22. OpenMP C and C++ Application Program Interface, Version 1.0". **(1998),** OpenMP Architecture Review Board. October 1998

23. D. A. Lewine. *Posix Programmer's Guide:* **(1991),** Writing Portable Unix Programs with the Posix. 1 Standard. O'Reilly & Associates, 1991

24. Emery D. Berger, Kathryn S McKinley, Robert D Blumofe, Paul R.Wilson, *Hoard : A Scalable Memory Allocator for Multi-threaded Applications* ; The Ninth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS-IX). Cambridge, MA, November (**2000)**. Web site URL : http://www.hoard.org/

25. Marc Snir, Steve Otto, Steyen Huss-Lederman, David Walker and Jack Dongarra, (**1998**) *MPI-The Complete Reference: Volume 1, The MPI Core, second edition* [MCMPI-07].

26. William Gropp, Steven Huss-Lederman, Andrew Lumsdaine, Ewing Lusk, Bill Nitzberg, William Saphir, and Marc Snir (**1998**) *MPI-The Complete Reference: Volume 2, The MPI-2 Extensions*

27. A. Zomaya, editor. Parallel and Distributed Computing Handbook. McGraw-Hill, **(1996)**

28. OpenMP C and C++ Application Program Interface, Version 2.5 (**May 2005**)", From the OpenMP web site, URL **: http://www.openmp.org/**

29. Stokes, Jon 2002 Introduction to Multithreading, Super-threading and Hyper threading *Ars Technica*, October **(2002)**

30. Andrews Gregory R. 2000, Foundations of Multi-threaded, Parallel and Distributed Programming, Boston MA : Addison – Wesley (**2000)**

31. Deborah T. Marr , Frank Binns, David L. Hill, Glenn Hinton, David A Koufaty, J . Alan Miller, Michael Upton, "Hyperthreading, Technology Architecture and Microarchitecture", Intel (**2000-01)**

- **Thank You**
- *Any questions ?*