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 : Benchmarks (Part-I)

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

Measuring Multi-Core Performance Benchmarks

Lecture Outline

Following topics will be discussed

- Are Multi-Core technologies highly differentiated ?
- Is there a need for Multi-core Benchmarks to be highly differentiated ?
- An Overview of Past Two Decades (First Generation / Second Generation) Benchmarks
- Classification and Important Characteristics of Benchmarks
- Peak and Sustained Performance of Benchmarks
- Benchmarks on Message Passing Cluster of Multi Core Systems

Ref : 32, 33, 34 Source : http://www.eembc.org/

Multitude of Multi-Core - Performance Characteristics

Background

1978 →

- Started with X86
  - Multi-core represents a fraction of Multi-Core enabled devices
- A plethora of general-purpose shared memory, Symmetric Multi-Processing (SMP) Systems
  - X86 Architecture & Non-X86 Architecture
  - Shared Memory Programming

Before 2000 →

- Focusing on the Multi-Core Processors
  - Characteristics more or less similar to X86

Performance Characteristics: Peak Performance

Peak Performance

- Defined as the MFLOPS rate which the manufacturer guarantees the computer will never exceed
  - It is obtained by taking the clock rate of the given system an dividing it by the number of clock cycles a floating point instruction requires
- Peak Performance calculations assume the maximum number of operations that the hardware can execute in parallel or concurrently
- Peak Performance is a rough hardware measure; it essentially reflects the cost of the system
- There are some (rare) instances where peak performance can give a creditable idea of performance

Performance Characteristics: Sustained Performance

Sustained Performance

- It may be defined as the highest MFLOPS rate that can actual program achieved doing something recognizably useful for certain length of time

- It essentially provide an upper bound on what a programmer may be able to achieve

Efficiency rate = The achieved (sustained) performance divided by the peak performance

Note : The advantage of this number is its independent of any absolute speed.

Performance: Benchmarks Classification

Benchmark Classification

- Benchmarks can be classified according to applications
  - Scientific Computing
  - Commercial applications
  - Network services
  - Multi media applications
  - Signal processing
- Benchmark can also be classified as
  - Macro benchmarks and Micro benchmarks

Multitude of Multi-Core - Performance Characteristics

Background

- Started with X86
  - **2004** → Multi-core represents a fraction of Multi-Core enabled devices
- Focusing on the Multi-Core Processors
  - Characteristics more or less similar to X86
- **1996** → A plethora of general-purpose shared memory, Symmetric Multi-Processing (SMP) Systems
  - Shared Memory Programming

Multitude of Multi-Core - Performance Characteristics

Multi-Core : Vendors/Companies

- IBM, SUN, SGI, Intel, AMD, Free Scale Semiconductor MIPs Technologies
  - Embedded Systems
- ARM
- Free Scale Semiconductor
- Multi-core products in the form of Systems on Chip (SoCs) ……

Multitude of Multi-Core - Performance Characteristics

Application Specific Systems on a chip (SoC)

- A Processor with a general computing core plus a digital signal processing core
  - Shared
  - Distributed Memory Architecture
    - heterogeneous & homogenous
  - Employ various interconnection technologies

- Multi-Core technologies are highly differentiated
- Multi-core Benchmarks need to be highly differentiated

Brief Historical Perspective

Past two decades **Generation -1**

- Focused on a processor codes internal work – is sufficient
  - Valid depending on the processor characteristics to be ascertained
- Predominantly exercise on processor core and have little interaction with the external memory

  (En: Test functions & features such as pipelines, branch predictions, units, instruction sets, cache performance)

Brief Historical Perspective

Past two decades **Generation -1**

- Benchmarks are measured on top of most Operating systems
  - Measurement – Iterations for Second (Iteration is sequential execution of the Benchmark kernel)

- Compiler plays a bigger role in this benchmarking

  We have seen 70% difference depending upon the compiler used to generate the benchmark results

- EEMBC Benchmarks – First Generation

Brief Historical Perspective

EEMBC Benchmarks (Generation –1)

- Exercise on Processor Core
  - Little Interaction with external Memory
  - Test functions and features such as
    - Pipelines,
    - Branch prediction units
    - Instruction Sets
    - Caches
  - Measurement – Iterations for Second (Iteration is sequential execution of the Benchmark kernel)

Brief Historical Perspective

EEMBC Benchmarks (Generation –2)

- Go one step further [beyond first generation]
  - Provide significantly larger code and data sets to test robust memory and Cache hierarchy
  - Test CPU Speed, Throughput, Utilization
  - Test Application Kernels
  - Theory : Multiples Instant ions of each benchmark can be launched

Source : http://www.eembc.org/

Brief Historical Perspective

Current Trends

- Measure Multi-Core processor performance
  - Processor vendors
  - System developers
  - Academic research groups

Research Theory

- Multiple instantiations of each benchmark can be launched simultaneously
  - SPECrate (Same metric cane be used for Multi-Core processors to single processor)

www.speec.org/spec/glossary

Brief Historical Perspective

Issues to be addressed

- It is not possible to guarantee that the system is utilizing more than one core!

  (Possible – Employ some form of processor affinity)

- Programmer intervention may/may not be required
- The Multi-Core Platform's Scheduler will assume control over execution of the individual benchmark instantiations

Multi-Core Benchmark Criteria

Issues to be addressed

- Is Benchmarks are computationally or memory intensive, or some combination of both ?
- Sequential Code – Does it really work on all cores?
- Complier Hags – ?
- Processor Affinity - ?
- Any tools?

Source : http://www.eembc.org/

Multi-Core Benchmark Criteria

At Highest Level

Benchmarks for Multi-Core Architecture should be

Computationally

or

Memory intensive

or

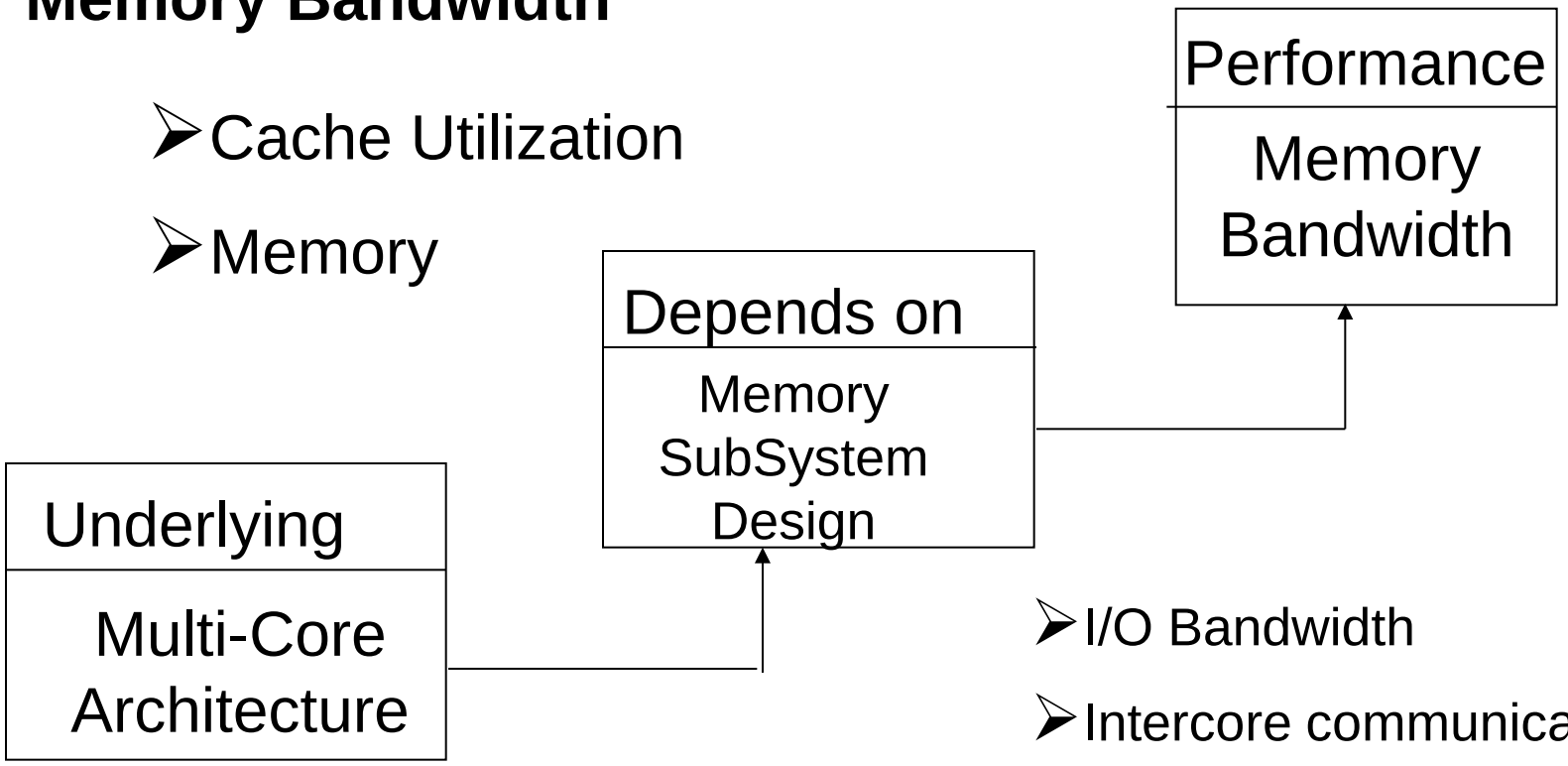
Some combination of both

Source : http://www.eembc.org/

Multi-Core Benchmark Criteria

Memory Bandwidth

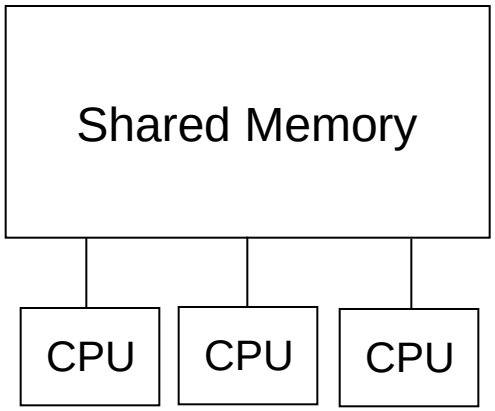
- Cache Utilization
- Memory



- I/O Bandwidth
- Intercore communication
- OS Scheduling Support
- Synchronization efficiency

Multi-Core Benchmark Criteria

(Underlying Multi-Core Architecture)

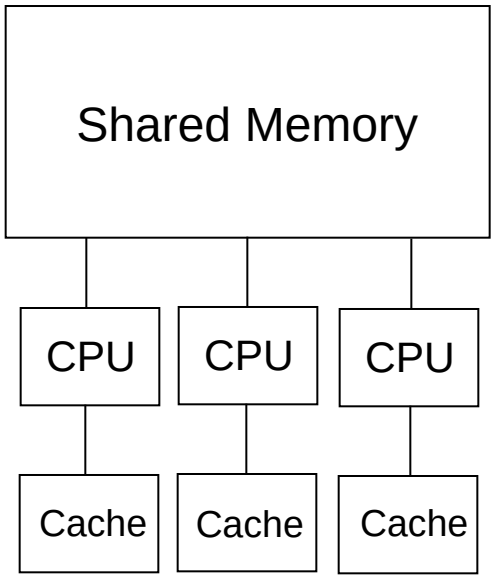


- Accessed through bus
- Controlled by some type of locking mechanism

  (To avoid simultaneous access by multiple locks)
- Straight framework programming model
- Each processor can directly access the memory

Multi-Core Benchmark Criteria

Memory Bandwidth



- Facilitate programming with traditional languages
  - POSIX thread
  - OpenMP thread Modelling allows persons data by reference without actually running data for cores with individual caches coherence
- There must be a coherency mechanism
- Ease of use of this architectures but performance bottlenecks due to competition between multiple cores accesses the some memory locations

Multi-Core Benchmark Criteria

Memory Bandwidth

- **Remark :** In Distributed Memory System, within an SoC each processor can access its own local memory but doesn't have to show with other cores even though there is a global memory address space)
- SoCs might have any number of cores, ranging in complexity from single-function hardware accelerators to full-fledged processors
- SoCs consists of either homogeneous or heterogeneous cores
- Employ variety of interconnect technologies

Multi-Core Benchmark Criteria

Memory Bandwidth

- Homogeneous or Heterogeneous cores
- Date Movement of Cores & Synchronization : When cores requires that data from another core, or cores must synchronize the system must synchronize, the system must physically move data or the control code must switch to run on a different core.
- Even though own core has its local memory, there might still be memory bottlenecks depending on how data moves on or off the chip itself.

Multi-Core Benchmark Criteria

Scalability

- Input performance penalty
- Over subscription computational resources
- Limitations on Memory bandwidth
- Varying number of threads
  - Number of reasonable to have hundreds of threads in a relatively complex program
  - Number of threads exactly matches with the number of processor cores (performance could linearly scale assuming no limitations memory bandwidth)

Multi-Core Benchmark Criteria

Scalability

- Number of threads will exceed the number of cores-
- Performance will depend on other factors
  - Cache Utilization
  - Memory
  - I/O Bandwidth
  - Intercore communication
  - OS Scheduling Support
  - Synchronization efficiency

Multi-Core Benchmark Criteria

Scalability

Question : Does sequential code work for benchmarking multicore processors?

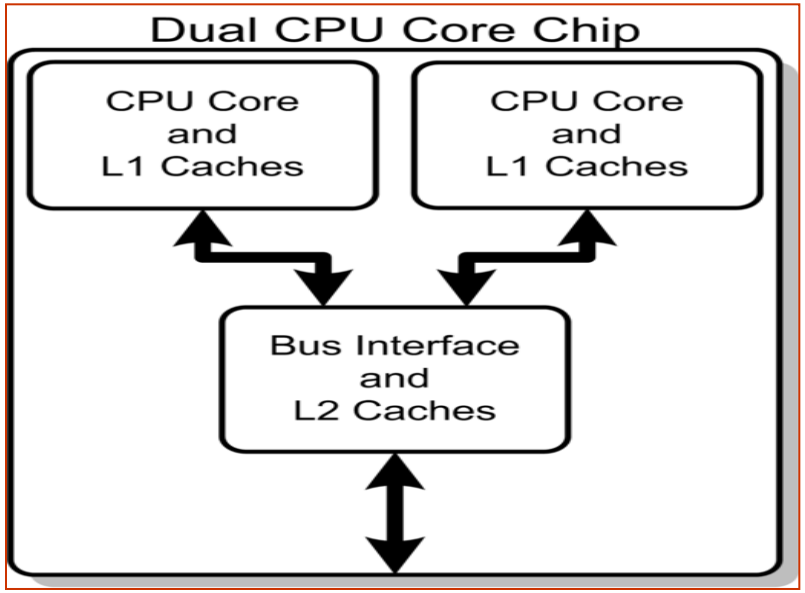
Answer : YES -

- But cumulative throughput of each individual core
- Memory bandwidth and computation can be evaluated

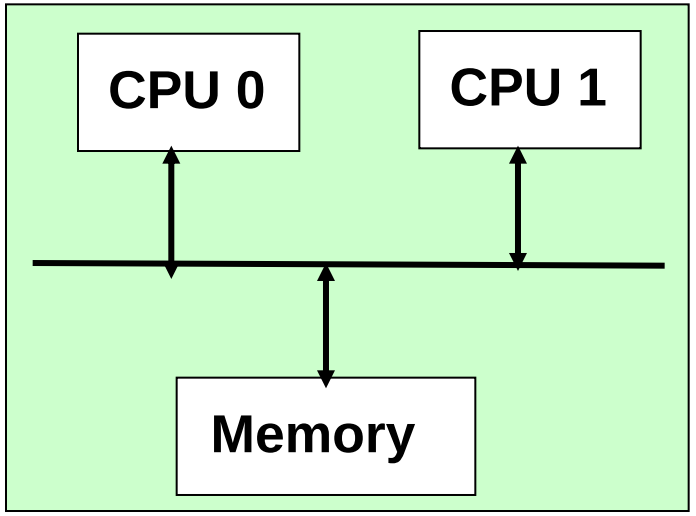
Example : Tele Communications

Telecommunication equipment manufacturer transitioned its application from a multi processor to a Multi-Core system.

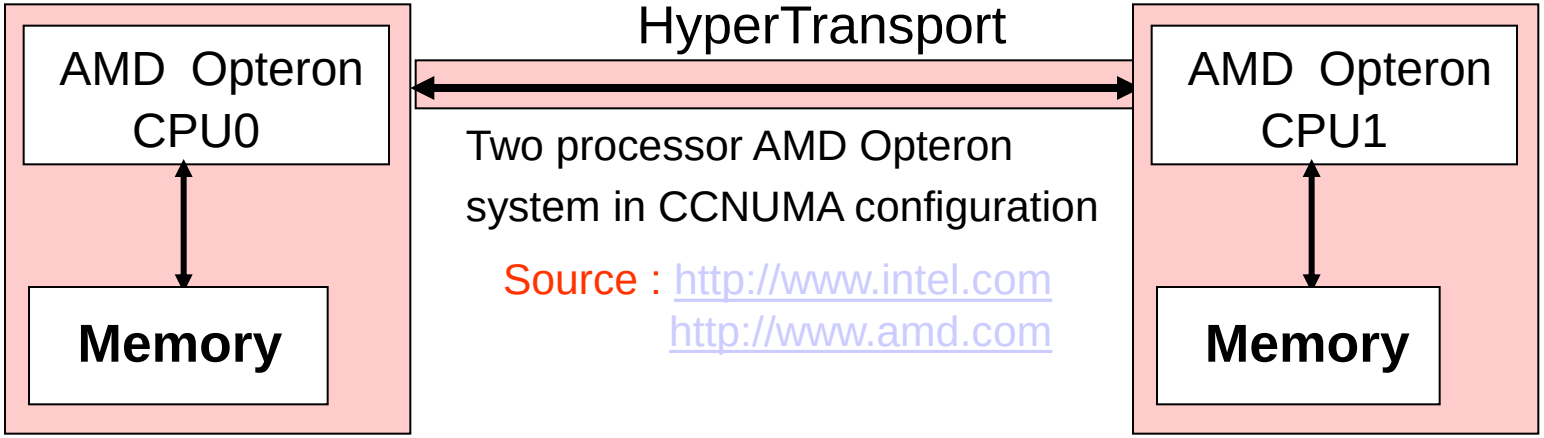
Multi-Cores Processors



Two processor Dual Core



Simple SMP Block Diagram for a two processors



Two processor AMD Opteron system in CCNUMA configuration

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

SMP Based Multi-Core Benchmarks

Scalability : Using parallelism to improve the performance of individual tasks rather than improving overall throughput

Example : Throughput - Scalable Web Servers

What is Given : Dual Core Processor

Method 1: Load one web page twice as fast as possible

Method 2: Parallelize the work by using both cores to load two web pages (one per core) in the same amount of time, it would be take to load one.

Requirements: Benchmarks should utilize task decomposition data decomposing, functional decomposition

Remark : Well known Benchmarks such as LLCBench, ScaLAPACK, LINPACK, HPCC ,Top 500, NAS Suite, I/O Bench can be used.

SMP Based Multi-Core Benchmarks

Remarks

- Same of Work - Number of contexts used
- Performance improvement or degradation
  - Number of contexts used
- Utilize any number of computation contexts because it will be used across many different platforms.
- Define Multi-core Benchmark Criteria – depends upon Application Characteristics

Multi-Core Benchmark Criteria

MULTI BENCH

- Number of API calls used to exploit parallelism
- Use of portable Operating systems interfere (POSIX) threads
- System should support Pthreads
- Benchmarks are based on set of workloads
  - Select list of workloads that closely resemble their application
- Can be ported on SMPs
- Example ; EEMBC Benchmarks

Multi-Core Benchmark Criteria

MULTI BENCH FrameWork (EEMBC Embedded Market**)**

SMP Benchmarks

- Consists of homogeneous threads
- Homogeneous cores
- Workload partitioning
- Support of easy API Abstraction on SMPS

NonSMP Benchmarks

- Support of heterogeneous core find in SOC's
- Orthogonal threads
- Heterogeneous cores
- workload partitioning
- Careful analysis is required to use different tool Chain
- Communication between parts of the system

Multi-Core Benchmark Criteria

Application-Specific Standard Benchmarks (ASSB)

- SMP based benchmarks
- Heterogamous core benchmarks
- Workloads are Heterogeneous
  - Compute or Memory or combination of both
- ASSBs are designed to take into account more System Level Benchmarks
  - TPC
  - NAS (Scientific Computations)

Multi-Core Benchmark Criteria

Black-box benchmarks

- Specify Input – Expect Output
- Developer can improve the performance than a commercial product
- Example: Performance of an SoC running the session Initiation protocol (SIP)
  - Multimedia communication sessions such as voice & Video calls over the Internet
  - SIP is used to set up and tear down multimedia communication sessions such as voice and video calls over the internet.

Multi-Core Benchmark Criteria

Black-box benchmarks

- Example: Performance of an SoC running the session Initiation protocol (SIP)
  - A commercial product should be able to receive and handle packet streams consists of mostly valid, but some invalid packets.
  - Leaving out the code that processes invalid bad packets
  - ASSB must also inject and test for bad packets

Multi-Core Benchmark Criteria

A New Era of Benchmark Evaluation

1. Creating industry standard benchmarks
2. ASSB should run properly on an evaluations platform (requires assembly of many system level components including hardware and software)
3. ASSB requires the hardware SoC, memory system I/O and the software (OS, application, running stacks)

- Well known Benchmarks such as LLCBench, ScaLAPACK, LINPACK, HPCC – Top 500, NAS suite can be used.

Source : http://www.eembc.org/

EEMC Benchmarks

- **EEMBC**, the Embedded Microprocessor Benchmark Consortium,
  - Itis a non-profit corporation formed to standardize on real-world, embedded benchmark software to help designers select the right embedded processors for their systems.
- **EEMBC** is a collection of "algorithms" and "applications" organized into benchmark suites targeting telecommunications, networking, digital media, Java, automotive/industrial, consumer, and office equipment products.

Ref : 32, 33, 34 Source : http://www.eembc.org/

EEMC Benchmarks

- **EEMBC**, the Embedded Microprocessor Benchmark Consortium,
- An additional suite of benchmarks, called MultiBench, specifically targets the capabilities of multicore processors based on an SMP architecture.
- These benchmarks may be obtained by joining EEMBC's open membership or through a corporate or university licensing program.
- The EEMBC Technology Center manages development of new benchmark software and certifies benchmark test results.

Ref : 32, 33, 34 Source : http://www.eembc.org/

EEMC Benchmarks

- Benchmark Scores
  - Automotive Consumer Digital Entertainment Java/CLDC
- Software Licensing and Membership
  - AutoBench, ConsumerBench, DENBench, GrinderBench (Java)
- Hypervisors
  - MultiBench, Networking, OABench, TeleBench, HyperBench
- Power/Energy
  - EnergyBench

Ref : 32, 33, 34 Source : http://www.eembc.org/

EEMC Benchmarks

❖ MultiBench™ 1.0 Multicore Benchmark Software

- Extends EEMBC benchmark scope to analyze multicore architectures, memory bottlenecks, OS scheduling support, efficiency of synchronization, and other related system functions.
- Measures the impact of parallelization and scalability across both data processing and computationally intensive tasks
- Provides an analytical tool for optimizing programs for a specific processor
- Leverages EEMBC's industry-standard, application-focused benchmarks in hundreds of workload combinations
- First generation targets the evaluation and future development of scalable SMP architectures
- MultiBench™ is a suite of embedded benchmarks that allows processor and system designers to analyze, test, and improve multicore architectures and platforms. MultiBench uses standardized workloads and a test harness that provides compatibility with a wide variety of multicore embedded processors and operating systems.

Conclusions

- Multi-Core performance requires new way of benchmarking
- New benchmarks is only half of the challenge, the other half of the challenge is interpreting the results
- Multi-Core technologies are highly differentiated
- Multi-Core Benchmarks need to be highly differentiated
- Characteristics of Benchmarks are discussed and identified set of benchmarks for Multi-Core Processors

Ref : 32, 33, 34 Source : http://www.eembc.org/

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

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

References

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

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

References

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)**

32. Shay Gai-On, & Markus Levy EEMBC,Measuring Multi-core Performance**, Computer** pp.99-102, Nov **2008**

33. EEMBC, the Embedded Microprocessor Benchmark Consortium URL : http://www.eembc.org/

34. MultiBench™ 1.0 Multicore Benchmark Software, URL : http://www.eembc.org/benchmark/multi_sl.php

Thank You

Any questions ?