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-II)

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

## Lecture Outline

Following topics will be discussed

- ❖ Peak and Sustained performance
- ❖ Important Characteristics of Benchmarks
- ❖ Benchmarks on Message Passing Clusters /Multi Core Systems
- ❖ Classification of Benchmarks

# Performance Characteristics

## Approaches to measure performance

- Several Approaches exist to measure performance of a Multicore System
  - Summarize several key architecture of a given computer system and relate them in order to get measure of its performance
  - Most of the measures are based on some engineering or design considerations rather theoretical calculations
  - Define set of programs and observe the system’s run times on those programs

# 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

# Performance: Macro Benchmarks

(Contd…)

**Macro Benchmarks**

- Measures the performance of computer system as a whole.
- Compares different systems with respect to an application class, and is useful for the system BUYER. However, macro benchmarks do not reveal why a system performs well or badly.

| Name | Area |
|---|---|
| NAS | Parallel Computing (CFD) |
| PARKBENCH | Parallel Computing |
| SPEC | A mixed benchmark family |
| Splash | Parallel Computing |
| STAP | Signal Processing |
| TPC | Commercial Applications |

# Performance: Micro Benchmarks (Contd…)

**Micro Benchmarks :** Synthetic kernels & measure a specific aspect of computer system ( CPU speed, Memory speed, I/O speed, Operating system performance, Networking)

| Name | Area |
|---|---|
| LAPACK; ScaLAPACK; LINPACK; BLASBench; HPCC suite Benchmarks | Numerical Computing (Linear Algebra) |
| LMBENCH | System Calls and data movement operations in UNIX |
| STREAM | Memory Bandwidth |

Source : http://www.netlib.org

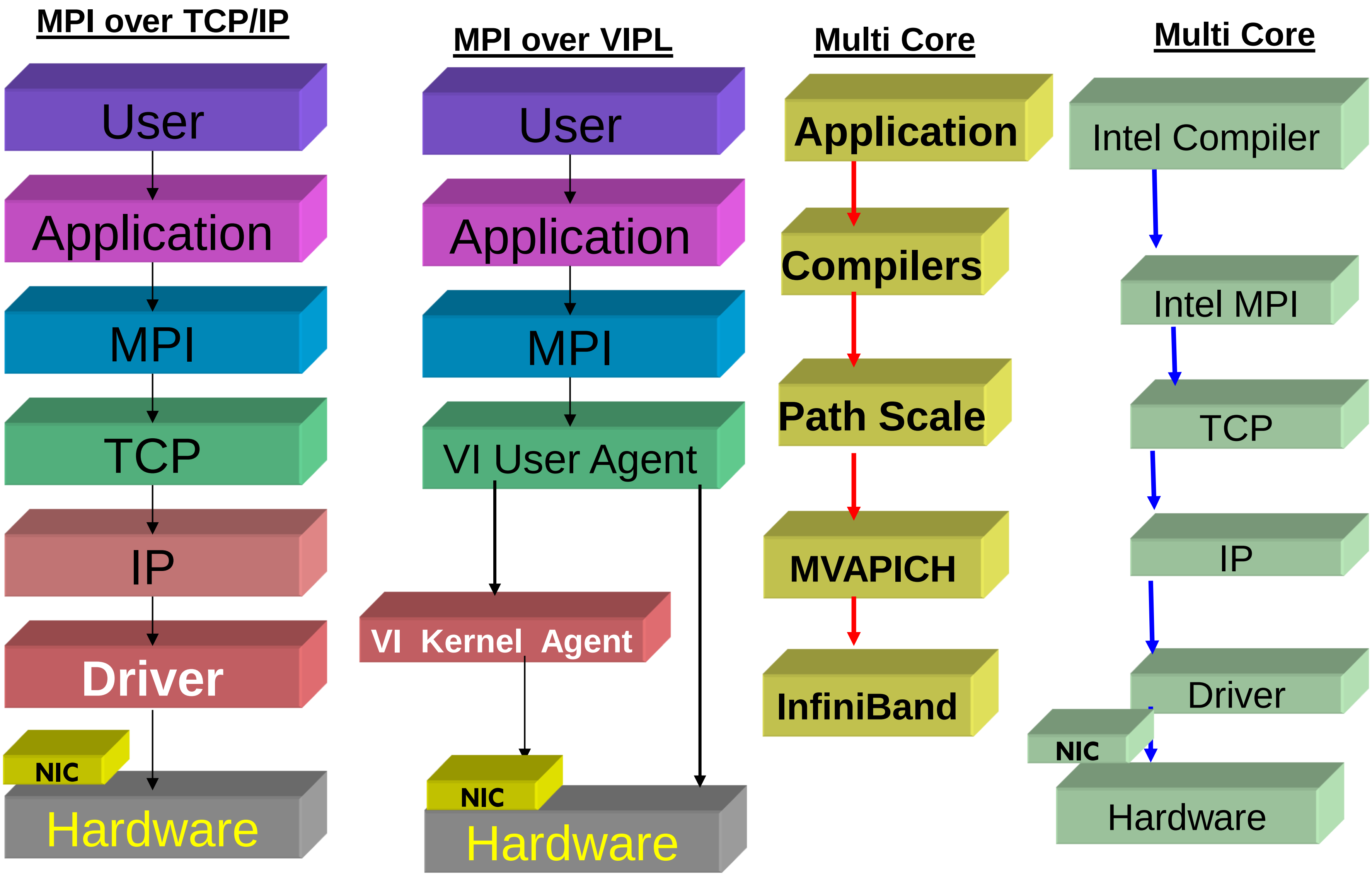
# Benchmarks on Multi Core Systems

(Contd…)

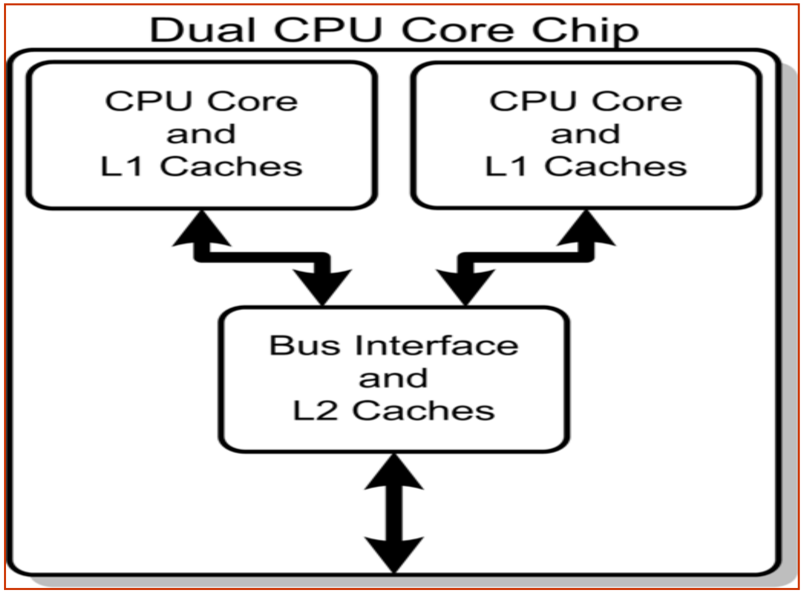
## Micro/Macro Benchmarks for Multi Core Processors

| Name | Area |
|---|---|
| SuperLU<br>HPCC Suite (Top-500 | Numerical Computing (Linear Algebra) |
| LMBENCH | System Calls & Data movement in Unix/Linux Environment |
| LLCBench | Low Level Cache Benchmarks |
| STREAM | Memory Bandwidth |
| I/O -Bench | I/O Benchmarks |
| TIO-Bench | Thread I/O Benchmarks |
| NAMD | Nanoscale Molecular Dynamics |
| NAS | Computational Fluid Dynamics |

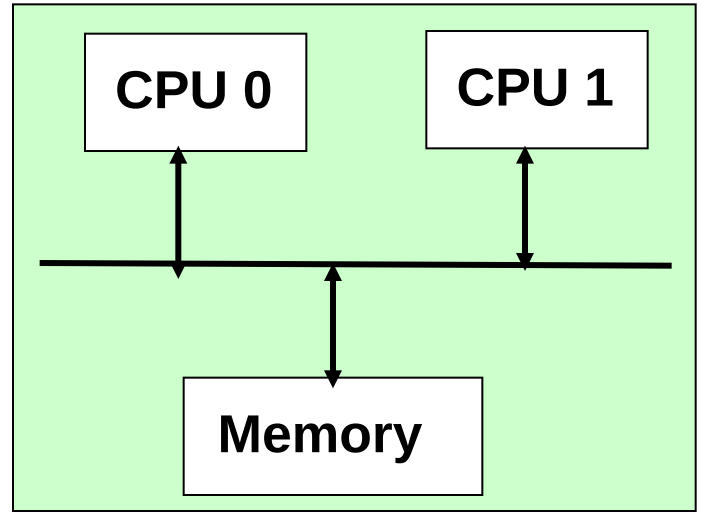
# Message Passing Cluster



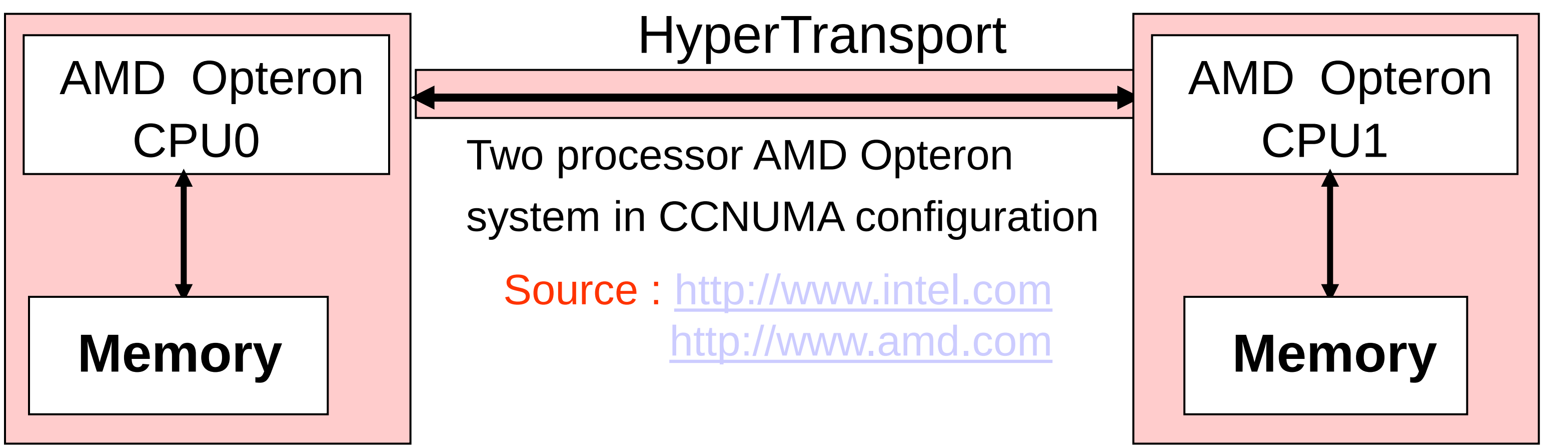
# Multi Cores Processors




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

# Micro Benchmarks: LAPACK

**Micro Benchmarks**

**LAPACK : (Used for SMP node performance)**

- To obtain sustained performance on one SMP node for Numerical Linear Algebra Computations
- Highly timed libraries of Matrix Computations may yield good performance for LAPACK

Source : http://www.netlib.org

# Performance: Micro Benchmarks (Contd…)

**Micro Benchmarks**

Representative Micro Benchmark Suite.

| Name | Area |
| --- | --- |
| LAPACK<br>ScaLAPACK<br>LINPACK | Numerical Computing (Linear Algebra)<br>Top-500 Benchmark; **http://www.top500.org/** |
| LMBENCH | System Calls and data movement operations in UNIX |
| STREAM | Memory Bandwidth |

Source : http://www.netlib.org

# Micro Benchmarks: LAPACK

## Micro Benchmarks

**LAPACK : (Used for SMP node /Multi Cores performance)**

- To obtain sustained performance on one SMP node for Numerical Linear Algebra Computations

- Highly timed libraries of Matrix Computations may yield good performance for LAPACK

Source : http://www.netlib.org

# Micro Benchmarks: LAPACK

(Contd…)

- LAPACK improves in four main aspects of a computer system
  - Speed
  - Accuracy
  - Robustness
  - Functionality
- Exploit Level 3 BLAS to get better performance : Level I, Level II, BLAS used
- Underlying concept
  - Use block partitioned algorithms to minimize data movement between different levels in hierarchical memory

Source : http://www.netlib.org

# Micro Benchmarks: LAPACK

(Contd…)

**Micro Benchmarks**

**LAPACK : (Used for SMP node Configuration )**

- Goal is to modify EISPACK and LAPACK libraries run efficiently on shared-memory vector and parallel processors
- LAPACK can be regarded as a successor to LINPACK and EISPACK
- LAPACK gives good performance on current Hierarchical memory computing systems
  - Reorganizing the algorithms to use block operations for matrix computations
  - Optimized for each architecture to account for the memory hierarchy

Source : http://www.netlib.org

# Micro Benchmarks: ScaLAPACK

## ScaLAPACK (Parallel LAPACK)

- ❖ Library of functions for solving problems in Numerical Linear Algebra Computations on distributed memory systems
- ❖ It is based on library ‘LAPACK’, ScaLAPACK stands for “Scalable LAPACK”.
- ❖ LAPACK obtains both portability and high performance by relying on another library the BLAS (Basic Linear Algebra Sub-program Library)
- ❖ The BLAS performs common operations such as dot product matrix vector product, matrix-matrix product.

Source : http://www.netlib.org

## ScaLAPACK (Parallel LAPACK)

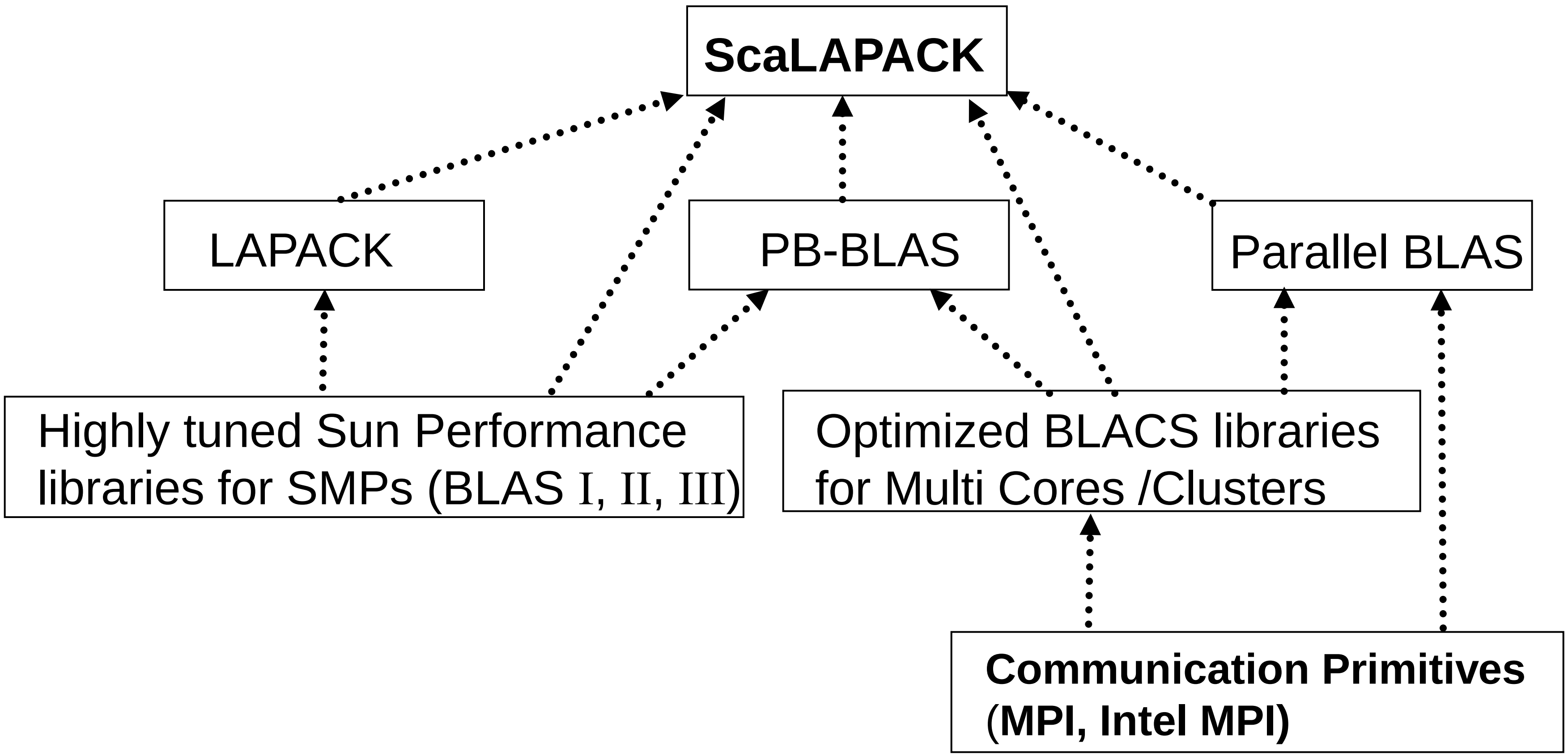
**The separate libraries are :**

- PBLAS (Parallel BLAS)
- BLACS (Basic Communication Sub-programs)
- In order to map matrices and rectors to processes, the libraries (BLACS, PBLAS, and ScaLAPACK) rely on the complementary concepts of process grid and block cyclic mapping.
- The libraries create a virtual rectangular grid of processor much like a topology in MPI to map the matrices and vectors to physical processors

Source : http://www.netlib.org

# Micro Benchmarks: ScaLAPACK

(Contd…)




Source : http://www.netlib.org

# Micro Benchmarks: LINPACK

## Micro Benchmarks

**The LINPACK (Top-500) Benchmark**

- ❖ The LINPACK benchmark was created and is maintained by Jack Dongarra at the University of Tennessee.
- ❖ It is a collection of Fortran subroutines that and solve linear equations and linear least square problems.
- ❖ Matrices can be general, banded, symmetric indefinite, symmetric positive definite, triangular, and tri-diagonal square.
- ❖ LINPACK has been modified to ScaLPACK for distributed-memory parallel computers.

Source : http://www.top500.org

# Micro Benchmarks: LINPACK

(Contd…)

## LINPACK

**Objective** : Performance of system of Linear equations (LU factorization) using MPI on Multi Core Systems.

- One can use LINPACK (ScaLAPACK version) maintained by Jack Dongarra, University of Tennessee (Top-500)

- Estimated performance equations exists but one should know the following characteristics of any parallel computer

  - Algorithm time

  - Communication time

  - Computation time

# Micro Benchmarks: LINPACK

(Contd…)

**LINPACK : Optimization**

- Arrangement of data in local memory of each process
- Hierarchical memory features usage for uniprocessor code performance
- Arrangement of matrix elements within each block
- Matrix blocks in the local memory
- Data in each block to be contiguous in physical memory
- Startup sufficiently large - it is preferable to send data as one large message rather than as several smaller messages

# Micro Benchmarks: LINPACK

(Contd…)

**LINPACK : Optimization**

- ❖ Overlapping communication and computation
- ❖ Tradeoff between load imbalance and communication
- ❖ Portability and ease of code maintenance
- ❖ Mapping logical memory to physical memory
- ❖ Assignment of processes to processors (topology)

# Micro Benchmarks: LMbench

- ❖ The LMbench benchmark suite maintained by Larry McVoy of SGI.

  - ➢ Focus attention on basic building blocks of many common computing systems performance issues

  - ➢ It is a portable benchmark aims at attempting to measure the most commonly found performance bottlenecks in a wide range of system applications - Latency and Bandwidth of data movement among processor, memory, network, file system and disk.

  - ➢ It is a simple, and useful tool for identifying performance bottlenecks and for the design of a system. It is meant to be a uniprocessor/Uni/Multi Core Benchmark.

# Micro Benchmarks: LMbench

(Contd…)

- ❖ Bandwidth : Memory Bandwidth; IPC Bandwidth ; Cached I/O Bandwidth
- ❖ Latency : Memory read Latency; Operating System Entry Latency;
- ❖ Signal Handling Cost
- ❖ Process creation costs; Null System Call; Context Switching (To measure System overheads)
- ❖ IPC Latency: Pipe, TCP & RPC/TCP Latency, UDP & RPC?UDP Latency, Network, TCP Connection; File System Latency; Disk Latency
- ❖ Memory latencies in nanoseconds - smaller is better

  Clock Speed (Ghz) , L1 Cache ; L2 Cache , Main memory

# Micro Benchmarks -LMBENCH

a) **LMBENCH:** System Calls and data movement operations in Unix; to measure the Operating system overheads and the capability of to measure the Operating system overheads and the capability of data transfer between processor, cache, memory and network, disk on various Unix platforms.

- **Bandwidth (MB/s):** Memory Copy, File Read, Pipe, TCP (Results are available.)

- **Latency(μs) :**Memory Read , File Create, Pipe, TCP (Results are available.)

- **System Overhead(μs) :** Null system call , Process creation , Context Switching

b) **LMBENCH:** Executed on Multiple cores and takes nearly 1-2 hours

# Micro Benchmarks: STREAM

- The Stream is a simple synthetic benchmark maintained by John McCalpin of SGI.
  - It measures sustainable memory bandwidth (in MBs) and the corresponding computation rate.
  - The motivation for developing the STREAM benchmark is that processors are getting faster more quickly than memory, and more programs will be limited in performance by the memory bandwidth, rather than by the processor speed.
  - The benchmark is designed to work with data sets much larger than the available cache
  - The STREAM Benchmark performs four operations for number of iterations with unit stride access.

# Micro Benchmarks: STREAM

(Contd…)

| Name | Code | Byte/ Iteration | Flop/ Iteration |
|---|---|---|---|
| COPY | a(i)=b(i) | 16 | 0 |
| SCALE | a(i)=q x b(i) | 16 | 1 |
| SUM | a(i) = b(i) + c(i) | 24 | 1 |
| TRIAD | a(i) =b(i) + q x c(i) | 24 | 2 |

❖ Machine Balance Metric =

$$\frac{\text{Peak floating-point (flop/s)}}{\text{Sustained TRIAD memory bandwidth (word/s)}}$$

❖ The machine balance metric can be interpreted as the number of flop that can be executed in the time period to read/write a word.

Array size = 10000000 (Your clock granularity/precision appears to be 1 microseconds.)

# Low Level Cache Benchmarks : LLCBench

- LLCbench is used to determine the efficiency of the various sub-systems that affect the performance of an application.
- Three Benchmarks :
  - BlasBench,
  - CacheBench,
  - MPBench
- Some sub-systems:
  - Memory
  - Parallel processing environment
  - System libraries
  - Communication sub-system.
  - File sub-system.

# LLC : BLASBench :Goals

- BLAS – basic linear algebra sub-routines.
- To provide a standardized API for common matrix & vector operations.
- Version available from http://www.netlib.org/
- Completely un-optimized FORTRAN codes.
- Evaluate the performance of vendors BLAS routines in MFLOPS.
- Provide info for performance modeling of applications that make heavy use of BLAS.

# LLC : BLASBench - Goals

- ❖ Evaluate Single/Multi Core efficiency by comparing performance of reference BLAS and hand tuned BLAS of the vendor.
- ❖ Evaluate the performance of vendors BLAS routines in MFLOPS.
- ❖ Provide info for performance modeling of applications that make heavy use of BLAS.
- ❖ Evaluate compiler efficiency by comparing performance of reference BLAS and hand tuned BLAS of the vendor.
- ❖ Validate vendors claims about the numerical performance of their processor.
- ❖ Compare against peak cache performance to establish bottleneck – memory or CPU

# LLC : CacheBench - Goals

- ❖ Evaluate the performance of the memory hierarchy of a computer system.
- ❖ Focus of the multiple levels of cache.
- ❖ Measures – Raw bandwidth in mbps.
- ❖ Establish peak computation rate given optimal cache reuse.
- ❖ Verify effectiveness of high levels of compiler optimizations on tuned and un-tuned codes.
- ❖ Provide a good basis for application perf. modeling and prediction that have already been tuned for cache reuse.
- ❖ Combination of 8 different benchmarks on cache and memory
  - ➢ Provide info about how good the compiler is.

# LLC : MPIBench - Goals

- Evaluate the performance of MPI
- Can be used over any Message passing layer
- Interpretation of results is left to the user
- Uses flexible and portable framework to be able to be used over any message passing layer.
- Tests Eight Different MPI Calls.
  - Bandwidth; Roundtrip; Application Latency;Broadcast
  - Reduce; All Reduce;Bidirectional Bandwidth;All to All

# Micro Benchmarks
# LLCBench (BLASBench, MpBench, CacheBench)

## Micro Benchmarks executed on Multi Cores

a) **BLASBench :** equivalent to HPCC-DGEMM. DGEMM (Double precision Matrix into Matrix Multiplication) achieved performance of 4.7 Gflops, which is equivalent to 90 % of the peak performance (5.2 Gflops)

b) **MpBench** – MPI benchmarks executed (Equivalent version of HPCC-HLRS Suites available)

c) **CacheBench** – Low level Cache Benchmarks executed

# Micro/Macro Benchmarks : HPCC Benchmark Suite

## HPCC Benchmark on Multi Cores

a) **Top-500 :** Peak /Sustained Performance : Matrix Solution of Linear System of Equations [**A**] **{x} = {b} .**

b) **Ptrans :** Transpose of a Matrix Algorithms

c) **STREAM :** COPY; SCALE; SUM; TRIAD **(OpenMP)**

d) **DGEMM :** Single DGEMM (Double Precision Matrix into Matrix Multiplication, part of **BlasBench** )

e) **Random Access** –Random Access - GUP/s

f) **Fast Fourier Transformations - FFT**

g) **(b_eff ) :** MPI Benchmarks

# Macro Benchmarks : NAS

## The NPB suite (MPI/OpenMP) on Multi Cores

- The NAS Parallel Benchmarks (NPB) is developed and maintained by the Numerical Aerodynamics Simulation (NAS) program at NASA Ames Research Centre.
- The computation and data movement characteristics of large scale Computational Fluid Dynamics (CFD) applications.
- NPB provides a pencil and paper specification as well as MPI-based Fortran source code implementations.
- NAS benchmarks and based on the key components of several large scale Aero science applications used on supercomputers

# Macro Benchmarks: NAS

(Contd…)

- EP "Embarrassingly parallel" Kernel.
  - It provides an estimate of the upper achievable limits for floating point performance; i.e. performance without significance interprocessor communication.
- MG "A Simplified Multigrid" Kernel.
  - It requires highly structured long distance communication and tests both host and long distance data communication.
- CG "A conjugate gradient method".
  - This kernel is typical of unstructured grid computations in that it tests irregular long distance communications, employing unstructured matrix vector multiplication.
- FT: "A 3D partial differential equation solution using FFT's".
  - This kernel performs the essence of many "spectral" codes. It is a test of long distance communication performance.

# Macro Benchmarks: NAS

(Contd…)

**NAS benchmarks**

- IS: “A large integer sort”:
  - This kernel performs a sorting operation that is important in “particle method” codes. (Integer communication of various size)
- LU: LU factorization used in CFD applications.
- BT: Block Tridiagonal solution system in CFD applications.
- SP: Scalar Penta Diagonal solution in CFD applications. (LU, SP, BT : Communication and computation test.
  - Finite Difference Discretization of the 3D compressible Navier Stokes equations is considered for numerical simulations.

# Micro Benchmarks executed on Multi Cores - Suite-I

**a) C-DAC In-house Developed Test suites on Multi-Core Processors**

**Low-Level Benchmarks :**

Enhanced Memory Bandwidth, focusing on Prefetching Streams – to observe the Consecutive cache line misses; Performance of dot-product- loop bisection, tri-section, different data streams

- ❖ Different Matrix Matrix Computations
- ❖ Remote versus local memory access on the System

**Programming Environment** : OpenMP, Pthreads

# Micro Benchmarks executed on Multi Cores - Suite-II

## a) C-DAC In-house Developed Test suites on Multi-Core Processors

- **Integer computations**
  - Non-Numerical Computations (List of Integers, Sorting)
  - Numerical Computations (Matrix Computations)
  - **Programming Environment** : MPI, OpenMP, Pthreads
  - **Performance Criteria** : Time Taken to execute Class A, B, C
- **Floating Point computations**
  - Numerical Computations (Different PI Computations; Matrix Computations, Solution of Matrix System of Equations )
  - **Programming Environment** : MPI, OpenMP, Pthreads
  - **Performance Criteria** : Time Taken to execute Class A, B, C

# Micro Benchmarks executed on Multi Cores - Suite-III

a) **C-DAC In-house Developed Test suites on Multi-Core Processors**

- **Partial Differential Equations**
  - **Use Different Advanced Point-to-Point Library Calls**
  - **Programming Environment** : MPI, OpenMP
  - **Performance Criteria** : Time Taken to execute Class A, B, C
- **Sparse Matrix Computations**
  - **Use Different Point-to-Point & Collective Library Calls**
  - **Programming Environment** : MPI, OpenMP
  - **Performance Criteria** : Time Taken to execute Class A, B, C
- **Multi Threaded I/O Suites**
  - **Read /Write/Random Read & Write**
  - **Programming Environment** : PThreads

# Micro Benchmarks executed on Multi Cores - Suite-IV

a) **C-DAC In-house Developed Test suites on Multi-Core Processors**

- **Partial Differential Equations**
  - **Use Different Advanced Point-to-Point Library Calls**
  - **Programming Environment** : MPI, OpenMP
  - **Performance Criteria** : Time Taken to execute Class A, B, C
- **Sparse Matrix Computations**
  - **Use Different Point-to-Point & Collective Library Calls**
  - **Programming Environment** : MPI, OpenMP
  - **Performance Criteria** : Time Taken to execute Class A, B, C
- **Multi Threaded I/O Suites**
  - **Read /Write/Random Read & Write**
  - **Programming Environment** : Pthreads

# Summary

❖ Discussed various issues of Performance of Parallel Computers.

❖ Characteristics of Benchmarks are discussed and identified set of benchmarks for Multi Core Processors

❖ Well known Benchmarks such as LLCBench, ScaLAPACK, LINPACK, HPCC – Top 500, NAS suite are discussed

❖ A set of representative benchmarks are used for performance of Multi Core / Cluster of Multi Core Systems.

# 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 ?*