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 : Tuning & Performance –

Cluster of Multi-Core Processors

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

An Overview of Multi-Core Processors

Lecture Outline

Following Topics will be discussed

- Tuning & Performance – Cluster of Multi-Core Processors
- Compiler Issues; Inter-connection Issues
- Cost of Message Passing
- Use of Performance Tools

Gains from tuning categories

Tuning Category	Typical Range of Gain
Source range	25-100%
Compiler Flags	5-20%
Use of libraries	25-200%
Assembly coding / tweaking	5-20%
Manual prefetching	5-30%
TLB thrashing/cache	20-100%
Using vis.inlines/micro-vectorization	100-200%

Source : Reference [4],[6], [7], [8], [32], 34]

Loop Optimization Techniques

- Dependence Analysis
- Transformation Techniques
- Loop distribution
- Loop Alignment
- Node Splitting
- Strip Mining
- Loop Collapsing
- Loop Fission Loop Fusion
- Wave front method
- Loop Optimizations

More about Loop Optimizations

- Basic Loop Unrolling & Qualifying Candidates for Loop Unrolling
- Negatives of Loop Unrolling
- Outer and Inner Loop Unrolling
- Associative Transformations
- Loop Interchange

Source : Reference [4],[6], [7], [8], [32], 34]

Compiler Options for Performance

- Improve usage of data cache, TLB
- Use VIS instructions (templates) directly, via –xvis option
- Optimize data alignment Prevent Register Window overflow
- Creating inline assembly templates for performance critical routines
- Loop Optimizations that compilers may miss:
  - Restructuring for pipelining and prefetching
  - Loop splitting/fission
  - Loop Peeling
  - Loop interchange
  - Loop unrolling and tiling
  - Pragma directed

Source : Reference [4],[6], [7], [8], [32], 34]

Compiler Options for Performance

Compiler optimization options:

- -xO1 thru -xO5 (default is “none”, -O implies -xO3)
- -fast:easy to use, best performance on most code,but it assumes compile platform = run platform and makes Floating point arithmetic simplifications.
- Understand program behavior and assert to optimizer:
  - -xrestrict, if only restricted pointers are passed to functions
  - -xalias_level, if pointers behave in certain ways
  - -fsimple if FP arithmetic can be simplified
- Target machine-related:
  - -xprefetch, -xprefetch_level
  - -xtarget=, -xarch=, -xcache=, -xchip=
  - -xvector to convert DO loops into vector

Compiler Options for Performance

Compiler Optimization Switches

- Fortran and C compilers have different levels of optimization that can do a fairly good job at improving a program's performance. The level is specified at compilation time with –O switch.
  - A same level of optimization on different machines will not always produce the same improvements (don't be surprised!)
  - –O is either default level of optimization. Safe level of optimization.
  - – O2 (same as –O on some machines) simple inline optimizations
  - – O3 (and –O4 on some machines) more complex optimizations designed to pipeline code, but may alter semantics of program
  - – fast Selects the optimum combination of compilation options for speed.
  - – parallel Parallelizes loops.
- Quite often, just a few simple changes to one's code improves performance by a factor of 2,3, or better!

Algorithm Overheads

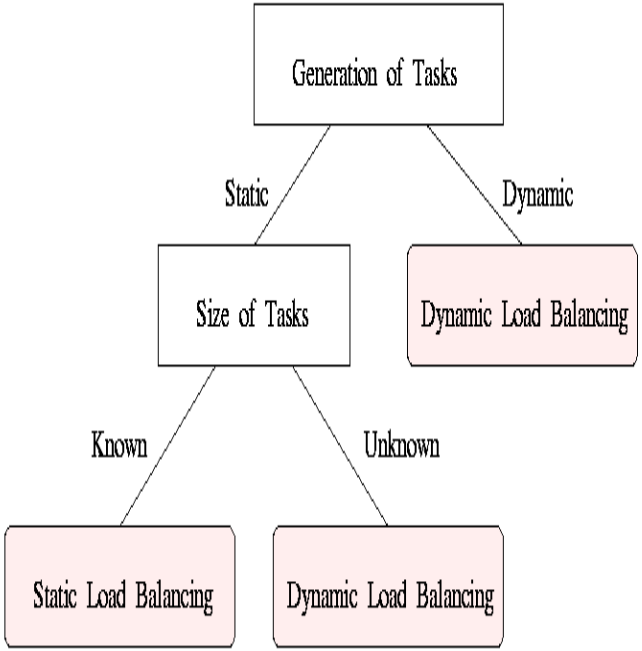
- Data parallelism;
- Task parallelism;
- Combination of Data and Task parallelism
- Static and Load Balancing
  - Mapping for load balancing
  - Minimizing Interaction
  - Overheads in parallel algorithms design
- Data Sharing Overheads

Decomposition techniques

- Recursive decomposition
- Data decomposition
- Exploratory decomposition
- Hybrid decomposition

Source : Reference :[1], [4]

Load Balancing Techniques



- **Static load-balancing**
  - Distribute the work among processors prior to the execution of the algorithm
  - Matrix-Matrix Computation
  - Easy to design and implement
- **Dynamic load-balancing**
  - Distribute the work among processors during the execution of the algorithm
  - Algorithms that require dynamic load-balancing are somewhat more complicated (Parallel Graph Partitioning)

System Configuration

- System configuration (OS, Compilers, Profilers)
- Interconnection Net-work – Point-point, One-all & All-to-All communications (Buffer Mechanism)
- Disk I/O Computations (Tuning II/O Controller)
  - Hardware; Processor Choice; Interconnect Choice
  - Network Card Choice (in some cases); IO Subsystem
  - Software; Compiler Choice; Compiler Options
  - MPI Choice; MPI Tuning; NIC Driver Tuning
  - Switch Tuning; OS Choice
- Sequential code Optimization (Compiler – Code Restructuring, Loop Optimization, Memory Allocators)

Tuning & Performance on Multi-Core Processors

Simple Approach

- Use Compiler Switches & explore performance – Localize the Data – Cache Utilization
- Use Profiler to understand behavior of programme
- Use Linux tool “top” to know about the CPU 7& Memory Utilization as well as Scalability with respect to varying size
- Threading APIs used; Locks & Heap contention
- Thread affinity – Explore the performance
- Sequential code Optimization – Use tuned libraries
- Check for Swapping (is the code Swapping ?) – Use “top” tools

Comparing compiler at various levels of Optimization

- Compiler vendors, sometimes include aggressive optimization at a lower level (For example -O2 may include some optimizations that other compiler vendors put in at -O3)
- Difficult to compare the same optimization levels among Compilers

CPU /Compiler	-O2	-O3	Aggressive	Tuning
AMD/PGI				
AMD PathScale				
Intel /PGI				
Intel Software				

- AbSoft
- Gcc

- Increasing the level of optimization doesn't improve the performance of the code.

Multi-Core Sub-system – Details

- CPU Utilization;
- Memory Usage;
- Network traffic,
- Disk Usage,
- Process Information

Multi-Core Sub-system – Details

Intel Compiler Optimization Switches

- Intel High-Level Optimizations (HLO)
- Intel Multiphase Optimizations
  - (IPO – Interprocedural Optmizations)
  - PGO –(Profile Guided Optimization) Switches
- Intel Math Kernel Library (MKL)
- Intel Integrated Performance Primitives

General Optimizations

Linux	Windows	-
-O0	/Od	Disables optimization
-g	/Z1	Creates symbols
-O1	/O1	Optimize binary Code for Server Size
-O2	/O2	Optimize for Speed (default)
-O3	/O3	Optimize for Data Cache : Loop floating point code
-axP	QaxP	Optimize for Intel Processors with SSE 3 Capabilities.
-parallel	-Qparallel	Auto-paalleization

Multi-Core Computing Systems

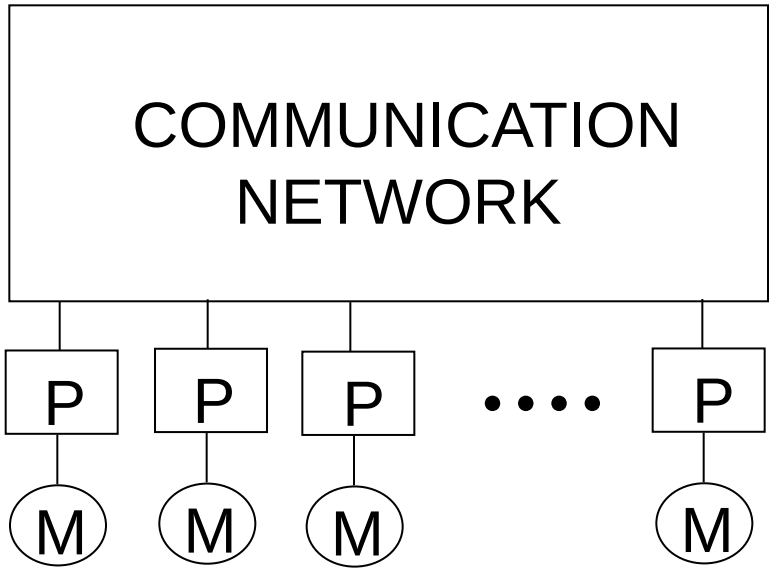
- Intel
- Cray
- IBM - Cell
- AMD
- SGI
- SUN
- HP

Multi-Core Sub-system – Details

- CPU Utilization; Memory Usage; Network traffic, Disk Usage, Process Information
- Intel Compiler Optimization Switches
- Intel High-Level Optimizations (HLO)
- Intel Multiphase Optimizations (IPC, PGO)
- Intel Multiphase Optimizations (PGO – Profile Guided Optimization Switches)
- Intel Math Kernel Library (MKL)
- Intel Integrated Performance Primitives

Message Passing Architecture Model

Message-Passing Programming Paradigm : Processors are connected using a message passing interconnection network.



- On most Parallel Systems, the processes involved in the execution of a parallel program are identified by a sequence of non-negative integers. If there are *p* processes executing a program, they will have ranks 0, 1,2, ……., *p-1.*

Interconnection Networks – Latency Bandwidth

- **TCP**
  - GiGE,
  - GigE with Jumbo Frames,
  - GAMMA, Level 5 GigE
  - 10 Gigabit Ethernet
  - 100 gigabit Ethernet
- **Myrinet**
- **Infiniband**
- **PathScale (Infinipath)**
- **Quadrics**
- **Dolphin**

- I/O time for Codes (CFD /Seismic Codes)

MPI Libraries (Open Source)

- Open-source
- **MPICH1 (Standard)**
- **MPICH2 (Standard)**
- LAM
- Open-MPI **(*)**
- GAMMA-MPI
- FT-MPI
- LA-MPI
- LA-MPI
- PACX-MPI
- MVAPICH
- OOMPI
- MPICH-GM
- MVICH
- MP_Lite

(*) OpenMPI combines the best features of LAM, FT-MPI, LA-MPI, and PACX-MPI. It supports TCP, Myrinet, Infiniband networks. http://icl.cs.utk.edu/open-mpi/

- OpenMPI has addition of the fault tolerance capability of FT-MPI.
- This will allow an MPI code to lose a node and then add a new node to finish the computations without lose of data.

Tuning & Performance on Multi-Core Processor Clusters

Test the following MPI Libraries (Open Source)

- Open-source
- **MPICH1 (Standard)**
- **MPICH2 (Standard)**
- LAM
- MPICH-GM (forMyrinet)
- MVAPICH (for Infiniband)
- Atleast one commercial MPI

- Choice of Compilers -Computing systems – Price / Performance
- Choice of MPI libraries Computing Systems – Price / Performance

Interconnection Networks – Latency Bandwidth

- **TCP**
  - GiGE,
  - GigE with Jumbo Frames,
  - GAMMA, Level 5 GigE
  - 10 Gigabit Ethernet
  - 100 gigabit Ethernet
- **Myrinet**
- **Infiniband**
- **PathScale (Infinipath)**
- **Quadrics**
- **Dolphin**

- I/O time for Codes (CFD /Seismic Codes)

Tuning & Performance on Multi-Core Processor Clusters

- **MPI Library calls & MPI algorithms implementation**
  - MPI Point-to-Point communication Calls;
  - MPI collective Communications,
  - MPI Communication & Computation Library Calls
  - MPI-2 library Calls
  - MPI I/O library Calls
- **Communication overheads for all MPI library calls**
- MPI 3.0 Thread enabled MPI

MPI Point-to-Point Communication: Communication Modes

MPI Primitive	Blocking	Nonblocking
Standard Send	MPI_Send	MPI_Isend
Synchronous Send	MPI_Ssend	MPI_Issend
Buffered Send	MPI_Bsend	MPI_Ibsend
Ready Send	MPI_Rsend	MPI_Irsend
Receive	MPI_Recv	MPI_Irecv
Completion Check	MPI_Wait	MPI_Test

Different Send/Receive operations in MPI

MPI Collective Communications

Type	Routine	Functionality
Data Movement	MPI_Bcast	One-to-all, Identical Message
	MPI_Gather	All-to-One, Personalized messages
	MPI_Gatherv	A generalization of MPI_Gather
	MPI_Allgather	A generalization of MPI_Gather
	MPI_Allgatherv	A generalization of MPI_Allgather
	MPI_Scatter	One-to-all Personalized messages
	MPI_Scatterv	A generalization of MPI_Scatter
	MPI_Alltoall	All-to-All, personalized message
	MPI_Scatterv	A generalization of MPI_Alltoall

Tuning & Performance on Multi-Core Processor Clusters

- Commercial Code tuning
  - File System
  - Network (TCP Buffers NIC, Switches)
  - L1 Cache /L2 Cache on Multi-Core Processors
  - Compiler Switches
  - MPI Libraries
  - OS & Memory Usage (80 %)

- Scalability of computing Systems & Performance Issues are Challenging

Thread Safety & MPI – Issues to be addressed

- Performance tradefoffs between multi-threaded and single-threaded code.
  - I/O operations
  - Against Inconsistent updates to the same memory location from different threads
  - Software locks and System locks are quite expensive
- Vendors sometimes provide single threaded /Multi-threaded libraries
  - Have I been linked with the right library ?
  - DO I suffer with occasional and mysterious errors

Source : Reference : [4], [6], [11],[12],[24],[25], [26]

Using PAPI : Events

- Events are occurrences of specific signals related to a processor's function.

  Ex: cache misses, number of floating point operations

- Preset events are mappings from symbolic names to machine specific definitions for a particular hardware resource.
  - Ex: **PAPI_TOT_CYC** (I.e Total Cycles), **PAPI_FLOPS**

- Native events comprise the set of all events that are countable by the CPU.

Using PAPI : PAPI library Interface

- PAPI provides two APIs to access the underlying counter hardware:
  - The low level interface manages hardware events in user defined groups called EventSets. (PAPI low level)
  - The high level interface simply provides the ability to start, stop and read the counters for a specified list of events. (PAPI high level)

An Overview of Multi-Core Processors

Conclusions

- An Overview of Tuning & Performance of Software threading on Multi-Core Software is discussed.

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)**

References

32. http://www.erc.msstate.edu/mpi/

33. http://www.arc.unm.edu/workshop/mpi/mpi.html

34. http://www.mcs.anl.gov/mpi/mpich

35. The MPI home page, with links to specifications for MPI-1 and MPI-2 standards : http://www.mpi–forum.org

36. Hybrid Programming Working Group Proposals, Argonne National Laboratory, Uchiacago (2007-2008)

37. TRAC Link : https://svn.mpi-forum.org/trac/mpi-form-web/wiki/MPI3Hybrid

38. Threads and MPI Software, Intel Software Products and Services 2008 - 2009

39. Sun MPI 3.0 Guide November 2007

40. Treating threads as MPI processes thru Registration/deregistration –Intel Software Products and Services 2008 - 2009

41. Intel MPI library 3.2 - http://www.hearne.com.au/products/Intelcluster/edition/mpi/663/

42. http://www.cdac.in/pemg2010

Thank You

Any questions ?