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-II : Memory Allocators

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

An Overview of Memory Allocator for Multithreaded Application

Lecture Outline

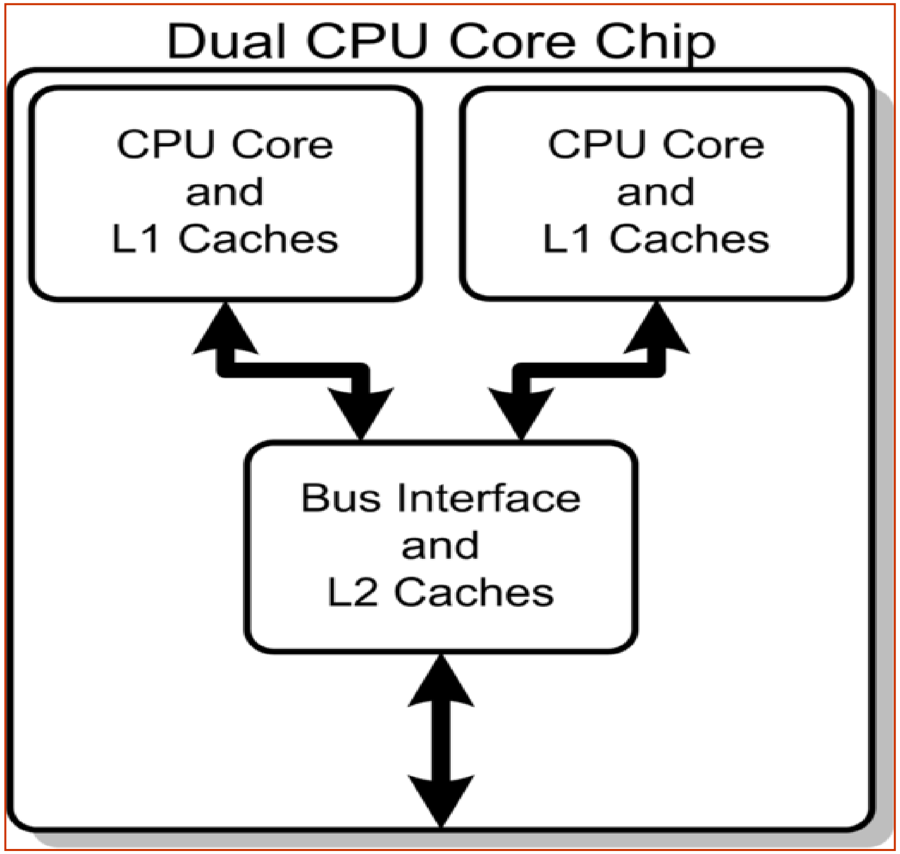
Following Topics will be discussed

- Introduction
- Understanding of Memory Allocation on Threads
- Case Studies & Examples

Dual Core Processor

Conceptual diagram of

- A dual-core CPU, with
- CPU-local Level 1 caches, and
- Shared, on-chip Level 2 caches



An Overview of Memory Allocator for Multithreaded Application

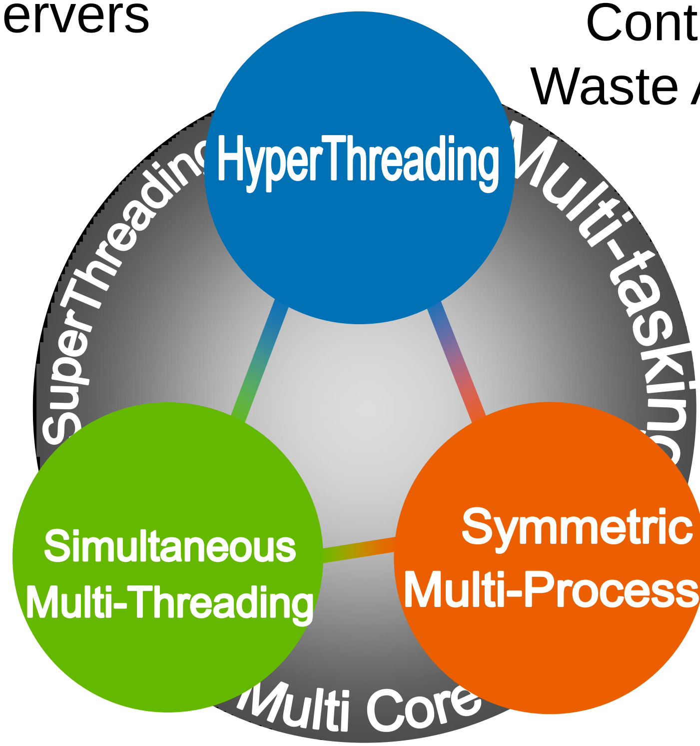
- Web Servers
- Data Base Managers
- news servers
- Parallel Scientific Applications
- Applications are written in C & C++
  - Shared Memory Multi-processors
  - **Make intensive use of Dynamic Memory Applications**

An Overview of threading

SMP and Cluster Platforms
based on Intel/AMD

Industry Standard Servers

Single Threaded CPU
Preemptive vs. co-operative
Multitasking
Context, process and Thread
Waste Associated with Threads
Time Slicing
I/O Threads



Implementing
Hyper-threading
- Replicated
- Partitioned
- Shared

Implementing
Hyper-threading
- Caching & SMT

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

An Overview of Memory Allocator for Multithreaded Application

❖ **Memory Allocation is often a bottleneck that severely limits program scalability on multiprocessor systems**

- Existing Serial memory allocations do not scale well for multithreaded applications.
- Concurrent memory allocators do not provide one or more following features….
  - Speed (fast malloc & free)
  - Scalability
  - False Sharing avoidance (Cache line)
  - Low fragmentation (Poor Data Locality, Paging)
  - Still some execution block is utilized
- Blowup

Hoard : A Memory allocator

- **Achieve Scalable Memory Performance on Shared Memory Architectures**
  - Questions should be addresses on Multi Cores
    - Per Core "heap" & "global heap"
    - Transfer of "heap" from processors to global
  - False Sharing : It occurs when multiple processors share words in the same cache line without actually sharing data.
  - False sharing of heap objects
  - The Scheduling of multithreaded programs can cause them to require much more memory when run on multiple processors rather than single processor.

Source : http://www.hoard.org/

Hoard : A Memory allocator

- **Example :**
  - Threads in Producer–consumer relationship
    - Blow-up mechanism exists ….
    - Memory Consumption grows linearly
  - Producer thread repeatedly allocates a block of memory and it gives it to a consumer thread which frees it.
  - If the memory freed by the consumer is unavailable it the producer, the program consumes more and more memory as it runs…
  - Memory Consumption grows without bound while the memory required….

Hoard : A Memory allocator

- Each thread calls **x= malloc(S); …free(s)**.
  - If these threads are serialised the total memory required is **s**.
  - For serial – Memory requirement is **s**
  - For **p** threads – memory requirement is **ps**.

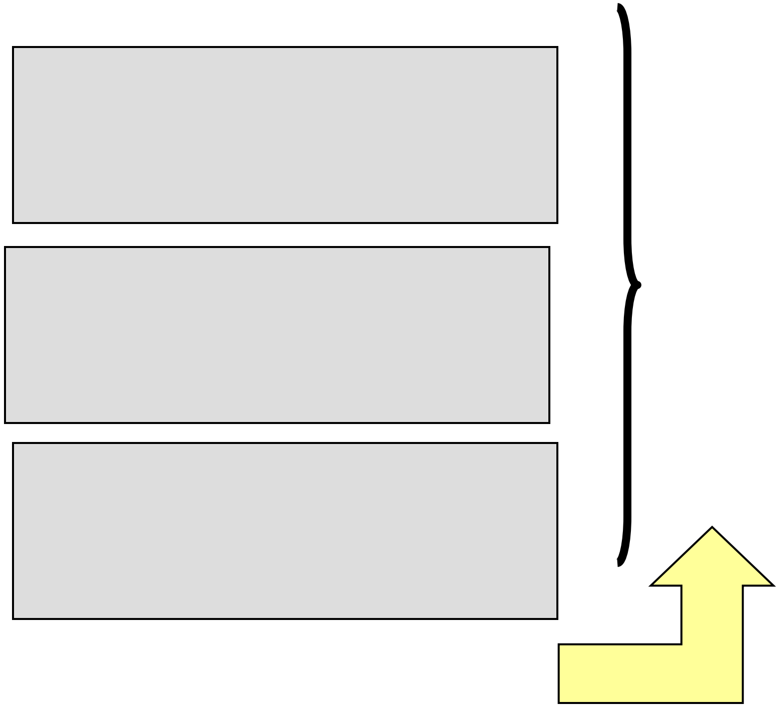
- If they execute on p processors, each call to malloc may run in parallel, increasing the memory requirement to **P*s.**

- Hoard can be viewed as an allocator that generally avoids false sharing & reduce synchronization costs..

- Each thread can access only its heap and global heap. Designation of heaps : 0 as global heap & heap 1 through p as the per-processor heaps.

Source : http://www.hoard.org/

Hoard : A Memory allocator



Heap 0 : Global heap

Heap 1 Processor 1

Heap 2 Processor 2

Heap 3 Processor 3

Heap 4 Processor 4

Superblocks

Each superblock has some blocks (Empty/Partially filled /Fully Filled)

thread 'k' maps to heap 'k'

Source : http://www.hoard.org/

Hoard : A Memory allocator

- Allocation and freeing in Hoard Memory Allocator
- Hoard maintains usage statistics for each heap
  - The amount of memory allocated by Hoard from the operating system held in heap i.
  - The amount of memory is use ("Live") in heap "I"
- Hoard allocates memory from the system in chunks as well as **superblocks**
- Each superblock is an array of some number of blocks (objects) and contains a free list of its available blocks maintained in LIFO order to improve locality.
  - All the superblocks are of same size (S), a multiple of system page size.

Hoard : A Memory allocator

❖ Allocation and freeing in Hoard Memory Allocator

- Collision of heap segments to threads by hashing on the LWP id.

❖ The number of LWP's – No of Processors

❖ Initially global heap is empty

❖ Thread 'k' is mapped to heap 'k'

❖ Global heap is empty

❖ Heap 1 has two superblocks (one is partially full & one is empty)

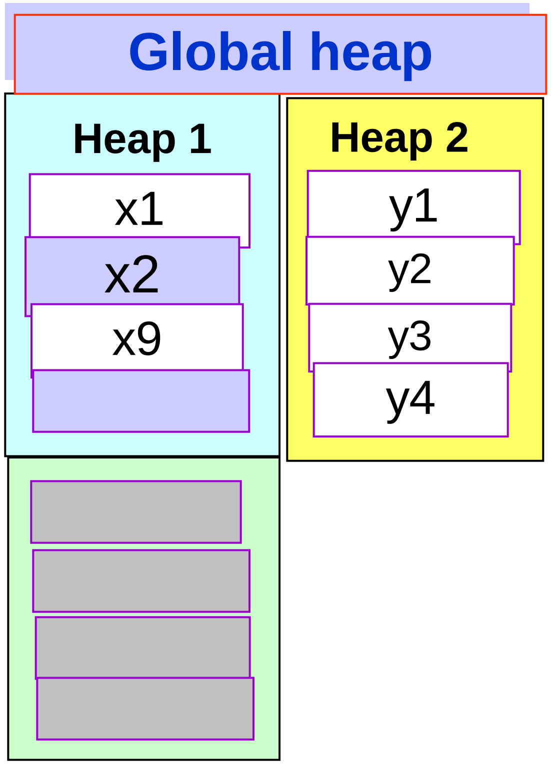
❖ Heap 2 has completely full superblock

❖ Superblock size = S

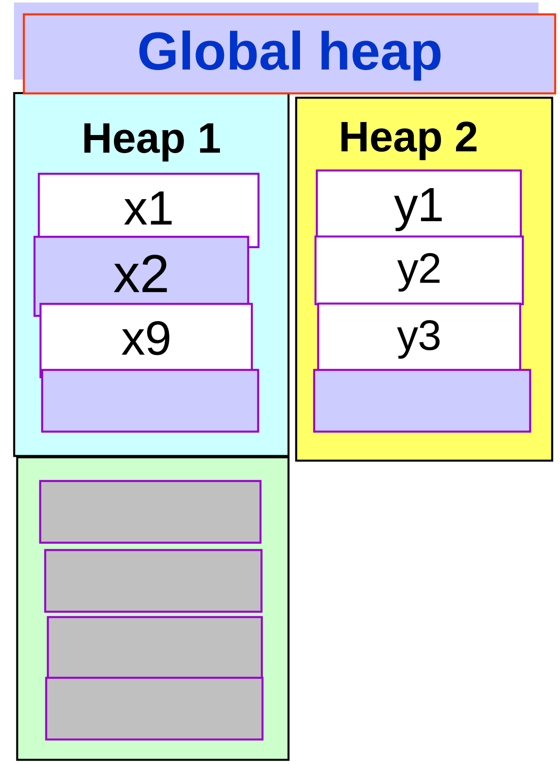
Hoard : A Memory allocator

- ❖ Allocation and freeing in Hoard Memory Allocator
    - ➢ Collision of heap segments to threads by hashing on the LWP id.
- ❖ The number of LWP's I set to No of Processors

t1: x9=malloc(s);

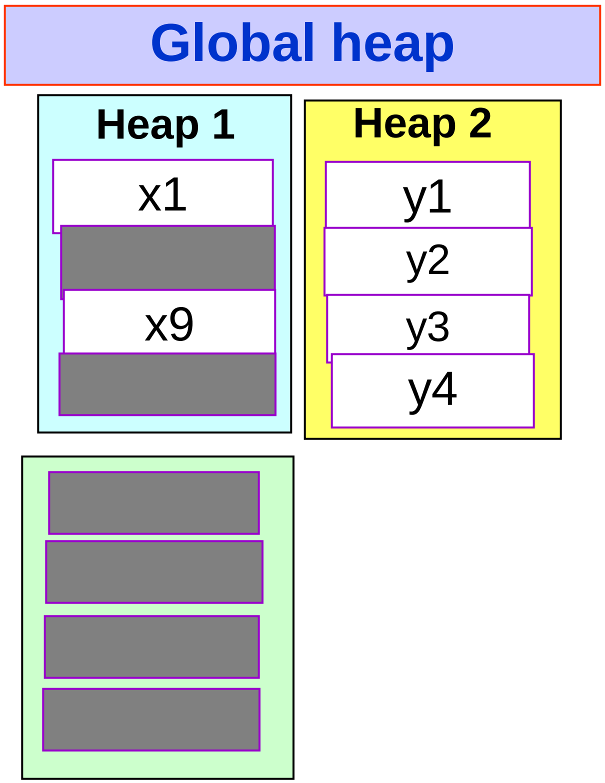


t1: free(y4);

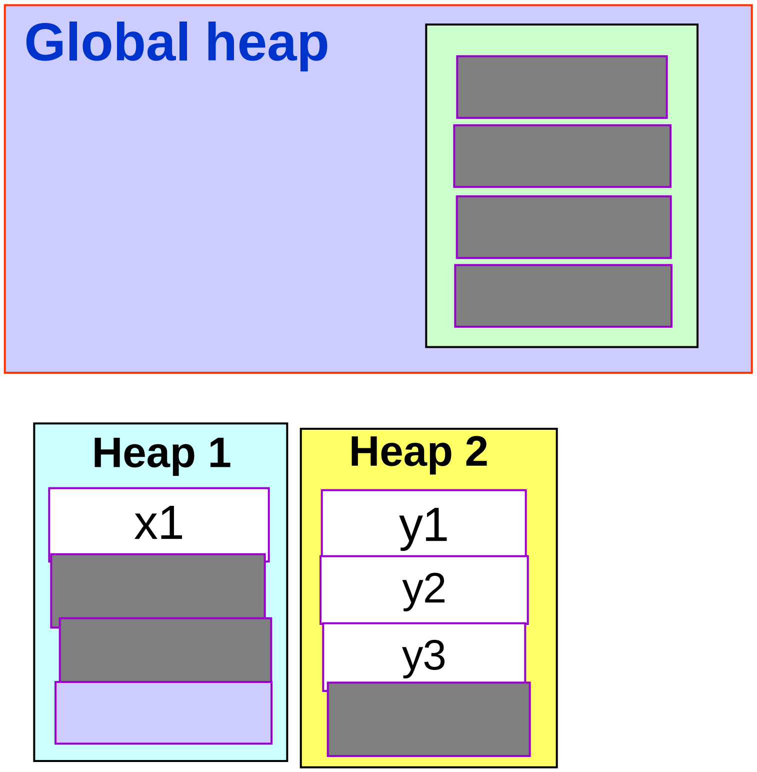


Hoard : A Memory allocator

t2: free(x2);



t2: free(x9);



Source : http://www.hoard.org/

Hoard : A Memory allocator

- ❖ Allocation and freeing in Hoard Memory Allocator
- ❖ Fragmentation Problems
  - ➢ Re-cycle completely empty superblocks for re-use.
- ❖ Avoid false Sharing
- ❖ Memory Allocation and De-allocation Algorithms

Hoard : A Memory allocator

- Single Threaded Applications
  - Each thread allocates one small object, writes on it a number of times and then **frees** it.
  - Overheads ignore
  - Superblock size = 1024*1024 Bytes.
  - Different classes & the number of classes
  - Avoid false Sharing

Hoard : A Memory allocator

- Multi-threaded Threaded Memory Benchmarks
  - **Shbench :** The large object size – randomly scattered in the super block
    - Represents real program
    - One Size Class per Superblock
    - Dynamic Storage Allocation
  - Larson Benchmark: Estimation of workload for server
  - Speedup, Scalability, and False Sharing avoidance

Hoard : A Memory allocator

❖ Taxonomy of Memory Allocated Algorithms

- **Serial Single heap**
- **Concurrent Single Heap**
- **Pure Private heaps**
- **Private heaps with ownership**
- **Private heaps with thresholds**

Hoard : A Memory allocator

Taxonomy of Memory Allocated Algorithms :Issues

- **Contention for the lock primitives**
- **Number of size Classes**
- **Freeing Blocks O(log C)**
- **Multiple heap Allocation**
- **Speed, Scalability, false Sharing avoidance and low fragmentation**

Conclusions

- An overview of Memory Allocation for Multi-threading Applications

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 ?