Exploring the Effects of Hyperthreading on Scientific Applications

by

Kent Milfeld

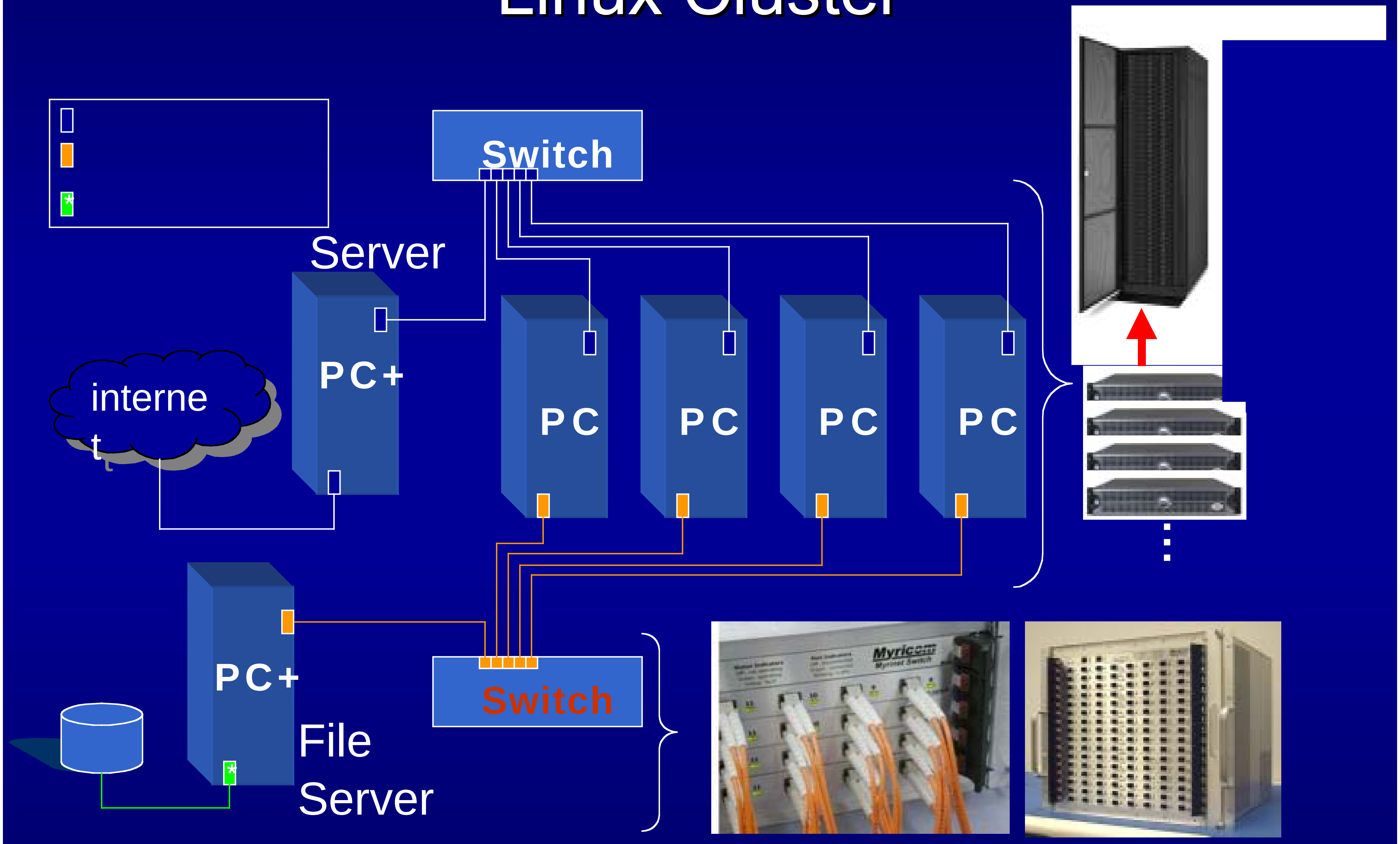
milfeld@tacc.utexas.edu

Kent Milfeld, Chona Guiang, Avijit Purkayastha, Jay Boisseau

TACC Linux Cluster

- 3.7 TFLOPS Cray-Dell Machine (to be installed in July)
- 300 Node Linux Cluster
  - CrayRx (SDSC Rocks + Cray System Admin)
  - Dell Service
- Dell 1750 Xeon Dual-Processor Nodes
- 3.0GHz processors
  dual channel 266MHz DDRAM (1.0GB/cpu)
- Myrinet CLOS configuration (2Gb/sec switch, “D” adapters)

Linux Cluster



Node

Dell PowerEdge 2650 / 1750

Processors: Two 3.0 GHz Intel® Xeon Processors

Chipset: ServerWorks Grand Champ LE chipset

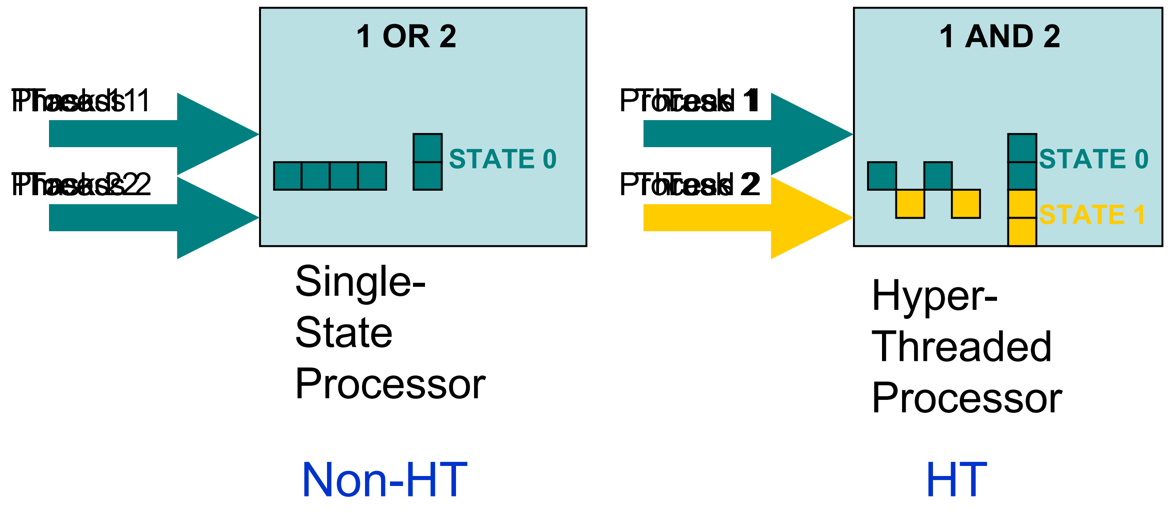
Memory: 2GB dual channel (266 MHz DDR SDRAM)

FSB: 533 MHz (Front Side Bus)

Cache: 512KB L2 Advanced Transfer Cache

Disk: Dual-channel integrated Ultra3 (Ultra160)
SCSI Adaptec® AIC-7899 (160Mb/s)
controller

Hyper-Threading Technology



Hyper-Threading Technology

- Hyper-Threading is an implementation of an architectural technique called Simultaneous MultiThreading (SMT)
- Exploits Instruction Level Parallelism on a Single CPU
- Why? Comm. Server Workload efficiency is about 67%
- Performance Benefits from “independent” processes/threads
  - Any time there is a stall for resources on a thread
  - Any time disparate operations are used
  - Commercial:

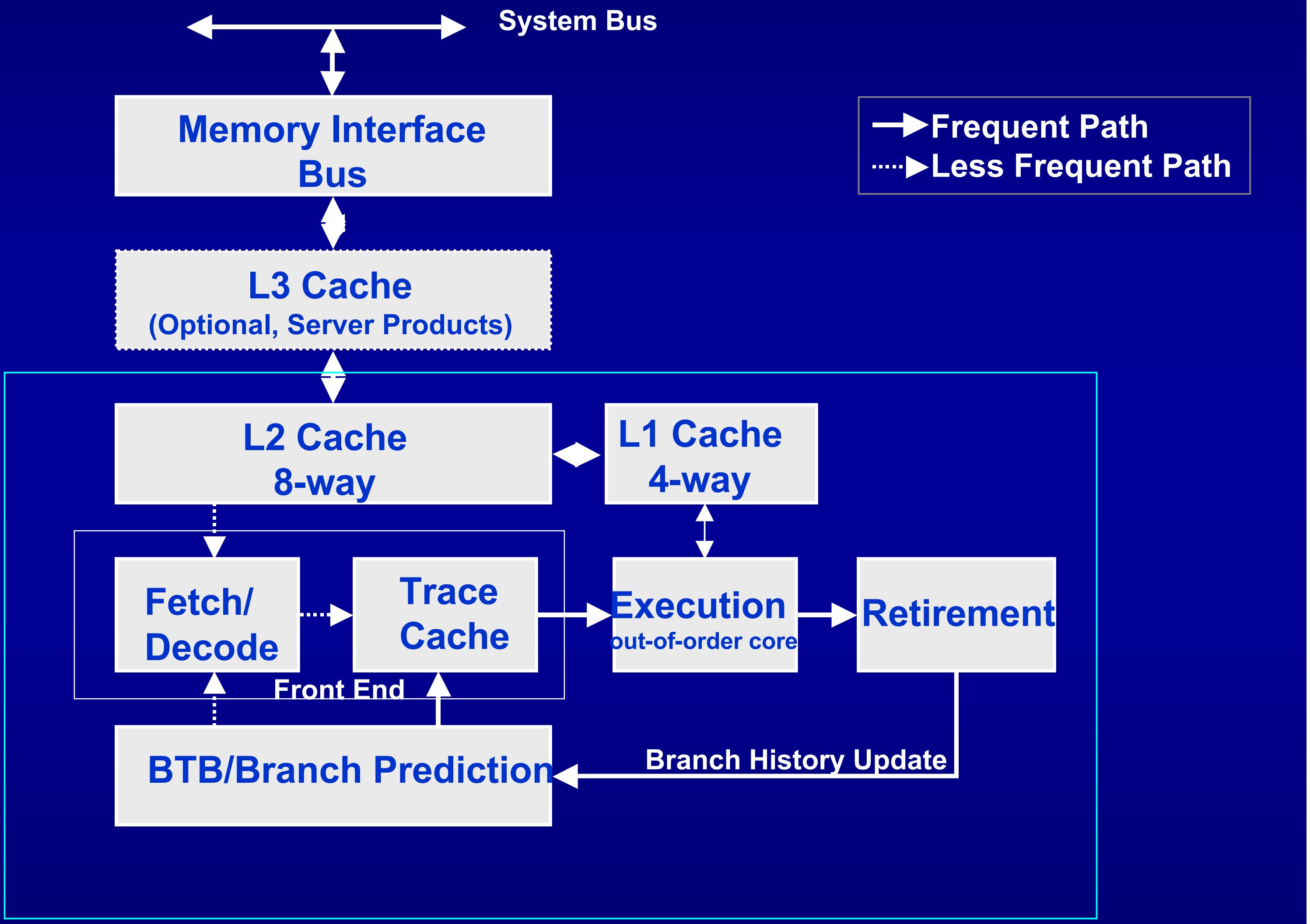
    Software (Algorithm & Code Modification)→ Multithreading
  - HPC:

    Already has large number of parallel applications
- What about the SHARING?

Architecture

- Intel Xeon
  - Seven-way superscalar
  - Able to pipeline 128 instructions
- Intel Solution:
  - Efficiency instead of Redundancy
  - With only 5% more real estate (die area)
- HT→ 2 Logical processors / CPU
  - Share most of the processor resources
  - Two processes and/or tasks execute in logical units concurrently

Microprocessor Architecture



Pipeline Depth

Pentium III processor misprediction pipeline

1	2	3	4	5	6	7	8	9	10
Fetch	Fetch	Decode	Decode	Decode	Rename	ROB Rd	Rdy/Sch	Dispatch	Exec

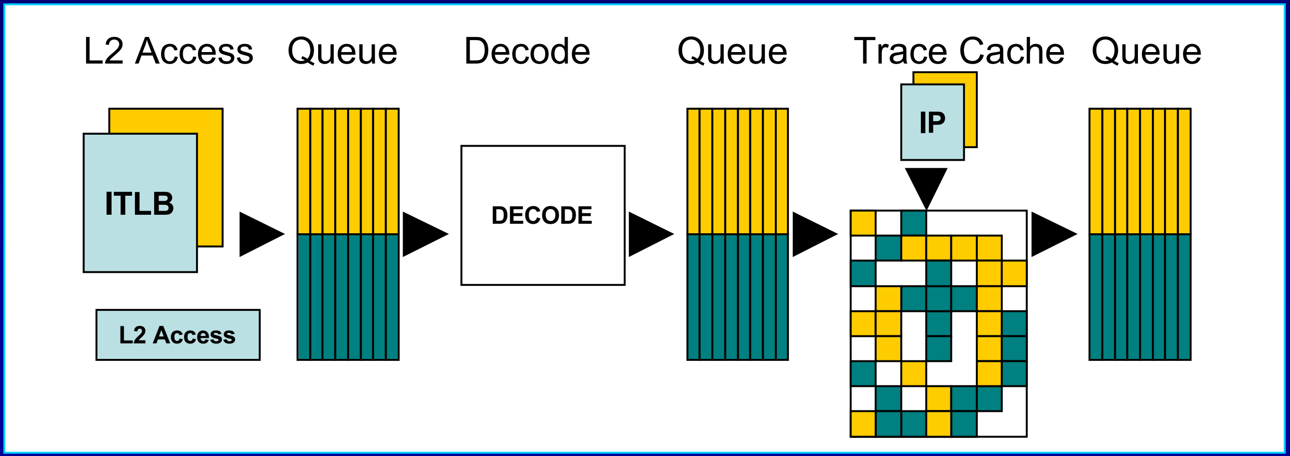
1 2	3 4	5 6	7 8	9 10	11 12	13 14	15 16	17 18	19 20
TC Fetch	TC Fetch	Drive Alloc	Rename	Que Sch	Sch Sch	Disp Disp	FR FR	Ex Flgs	BrCk Drive

Pentium 4 processor misprediction pipeline

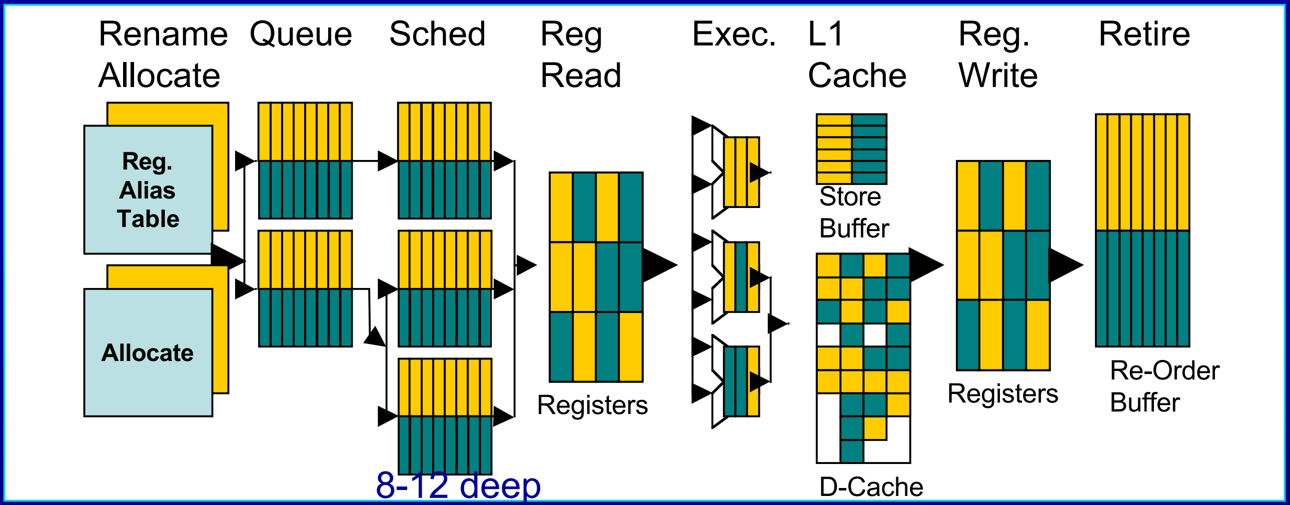
+Physical (renaming) Registers: 128 Integer and 128 Floating Point

Redesigned Pipeline for HT

Front End



Out-of-Order Execution Engine



Memory Measurements & Application Scaling with HT

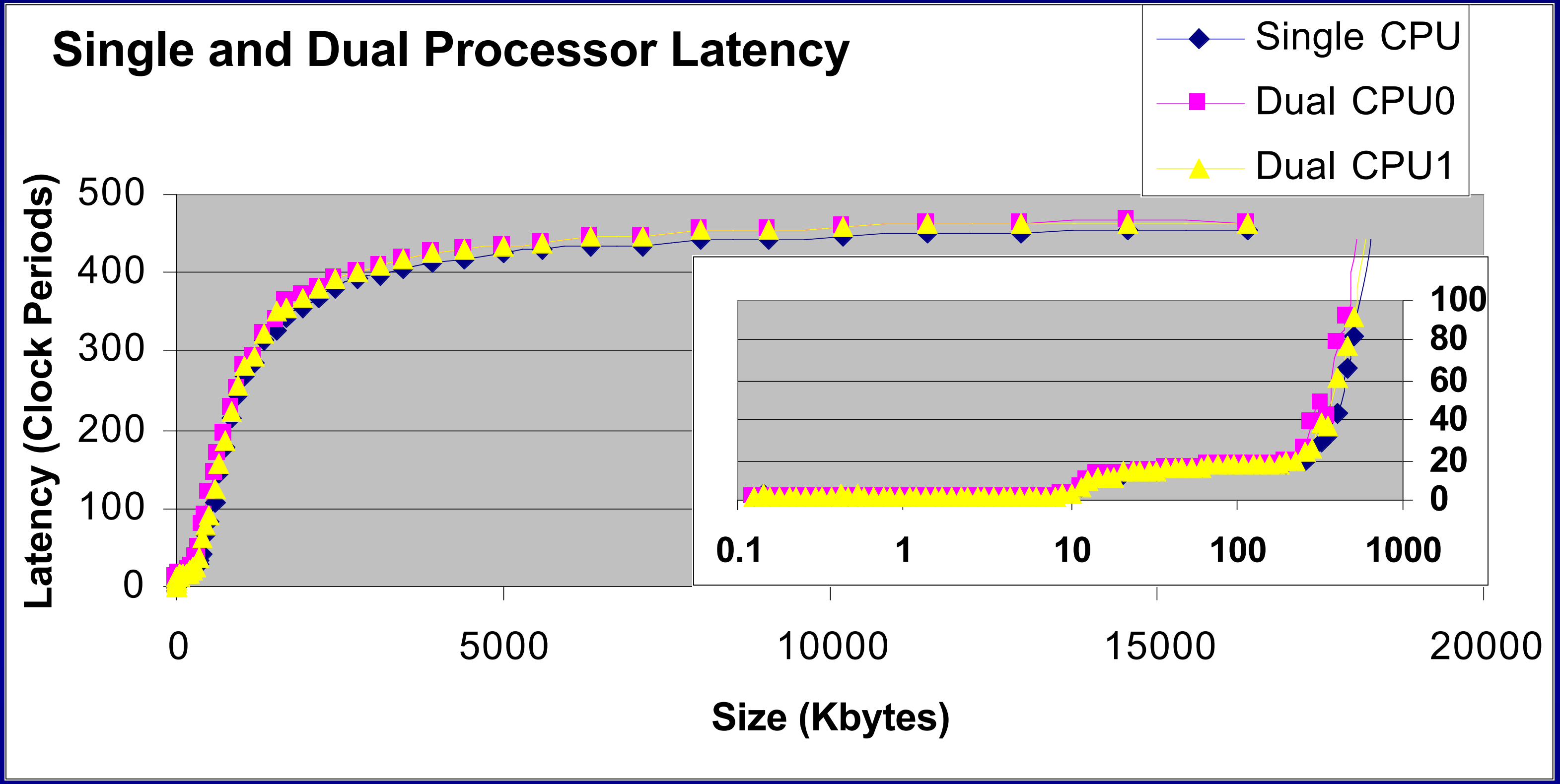
- Memory Characterization for a single processor
  - Latency
  - Bandwidth
- Memory Characterization for HT
  - Worst Use of Memory (non-strided Gather/Scatter)
  - Best Use of Memory (sequential access)
- Applications (MxM, MD, LP)

Memory Latency

- Measuring Latency

```
I1  =  IA(1)
DO  I  =  2,N
    I2  =  IA(I1)
    I1  =  I2
END  DO
```

Memory Latency Measurement

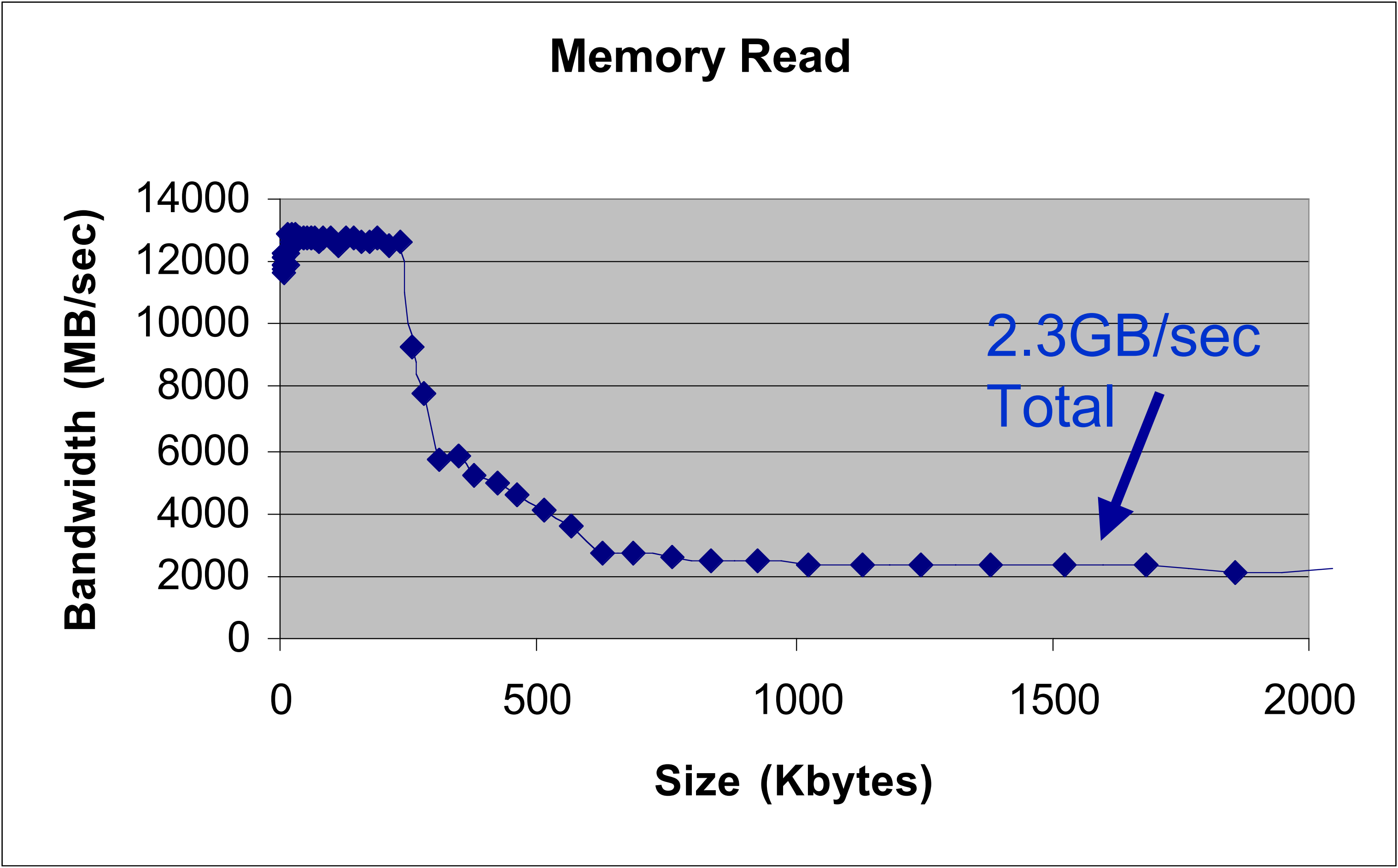


Bandwidth (with 2 Streams)

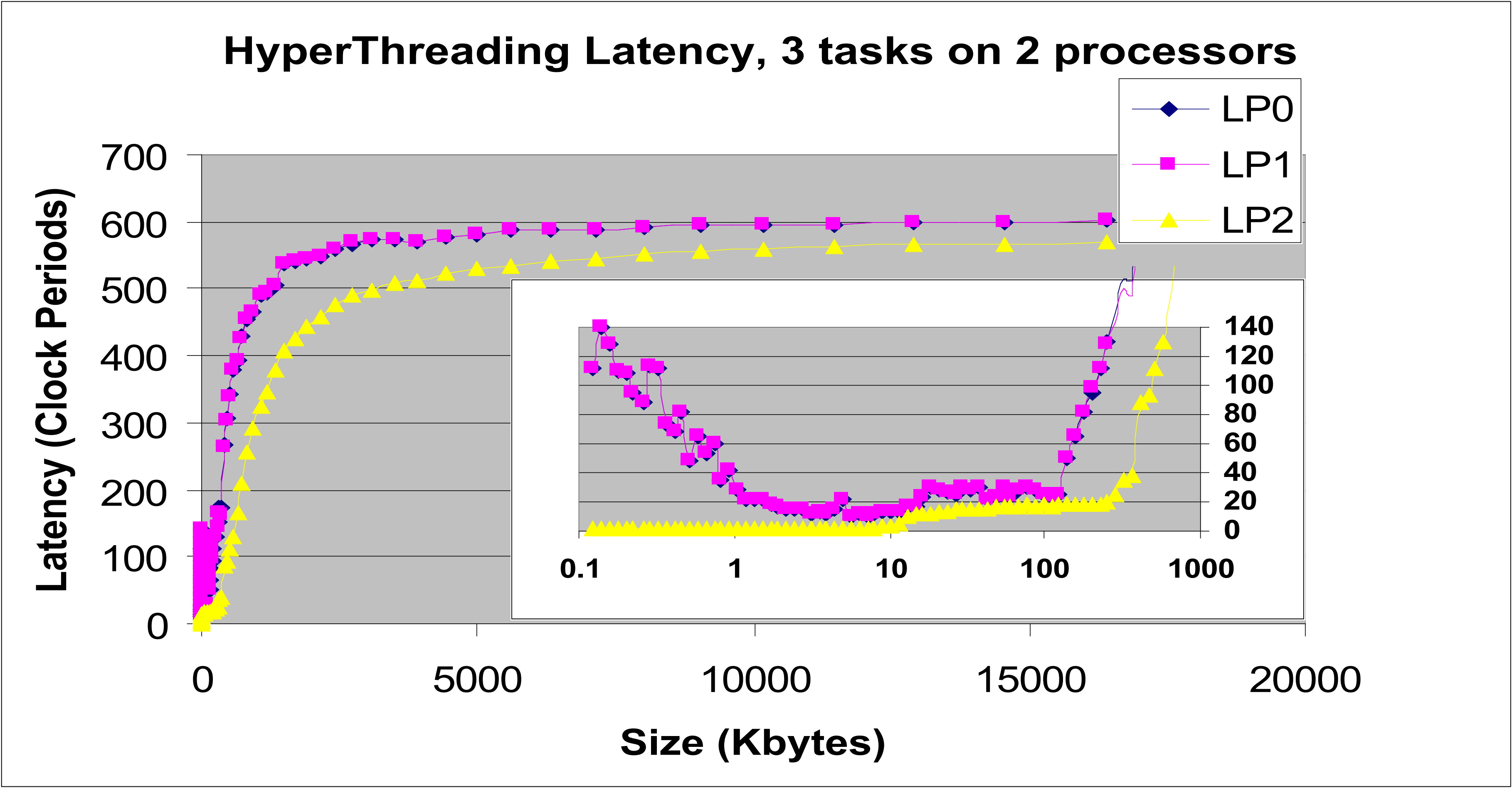
- Measuring Bandwidth

```
DO  I  =  1,N
    S  =  S  +  A(I)
    T  =  T  +  B(I)
END  DO
```

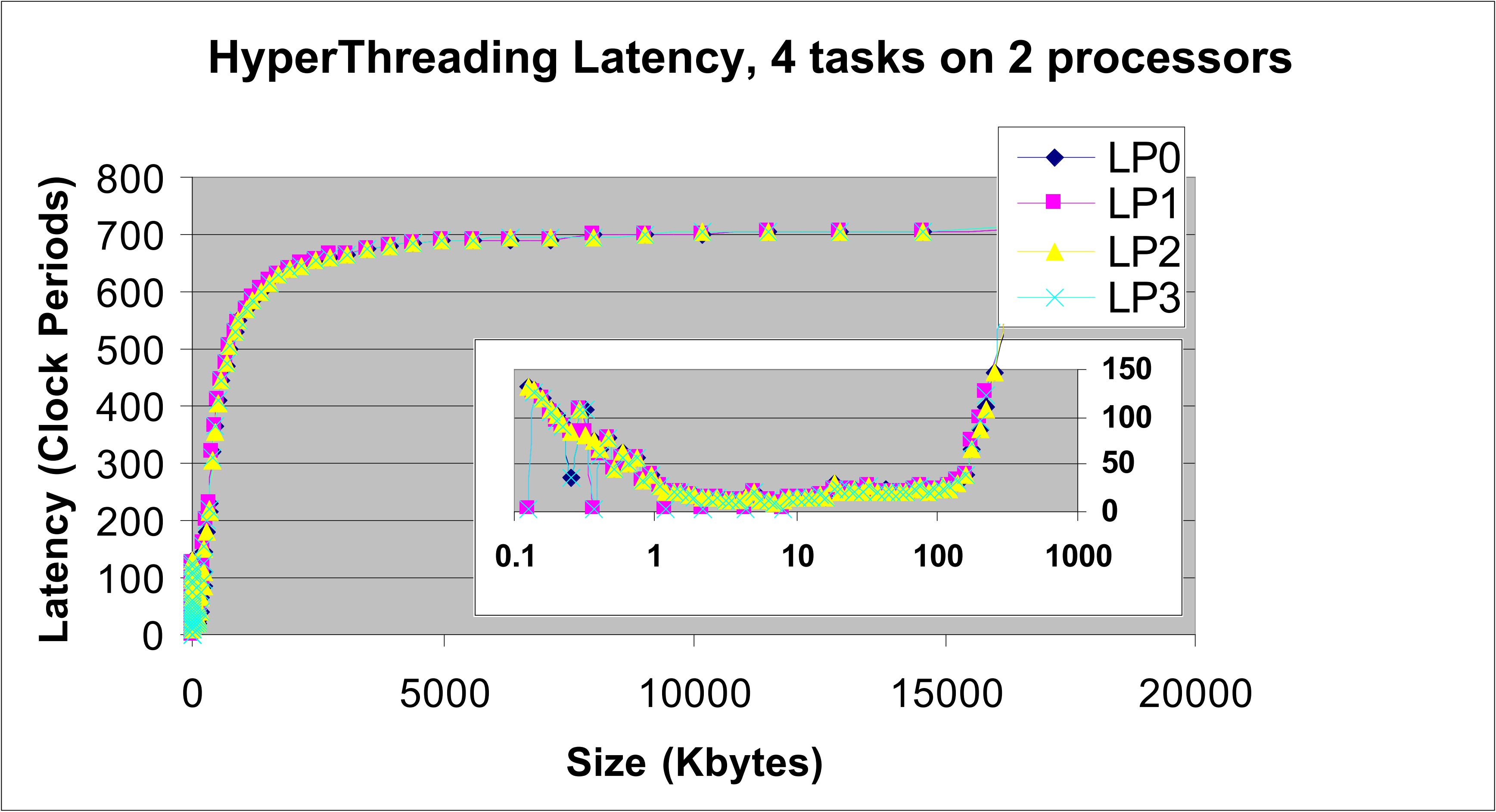
Memory Bandwidth



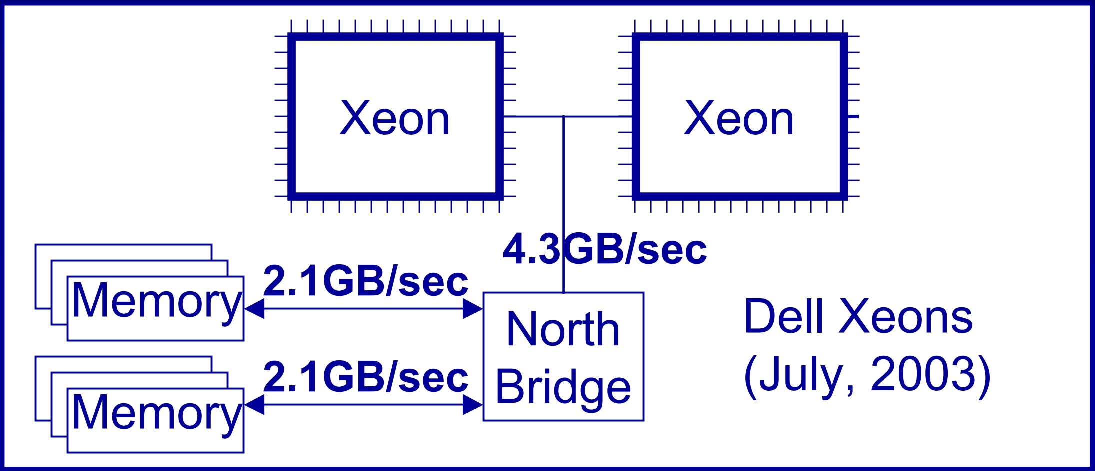
HT Memory Latency Measurement



HT Memory Latency Measurement

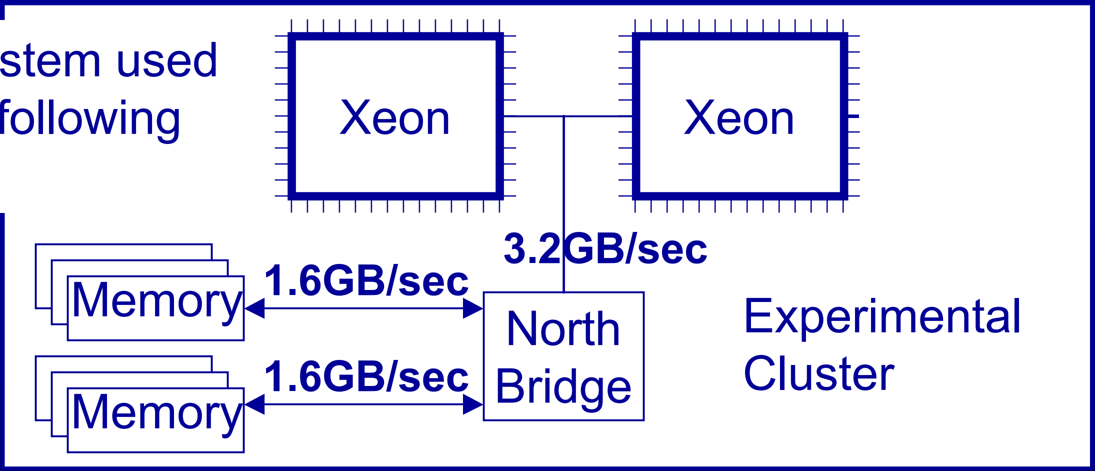


Intro



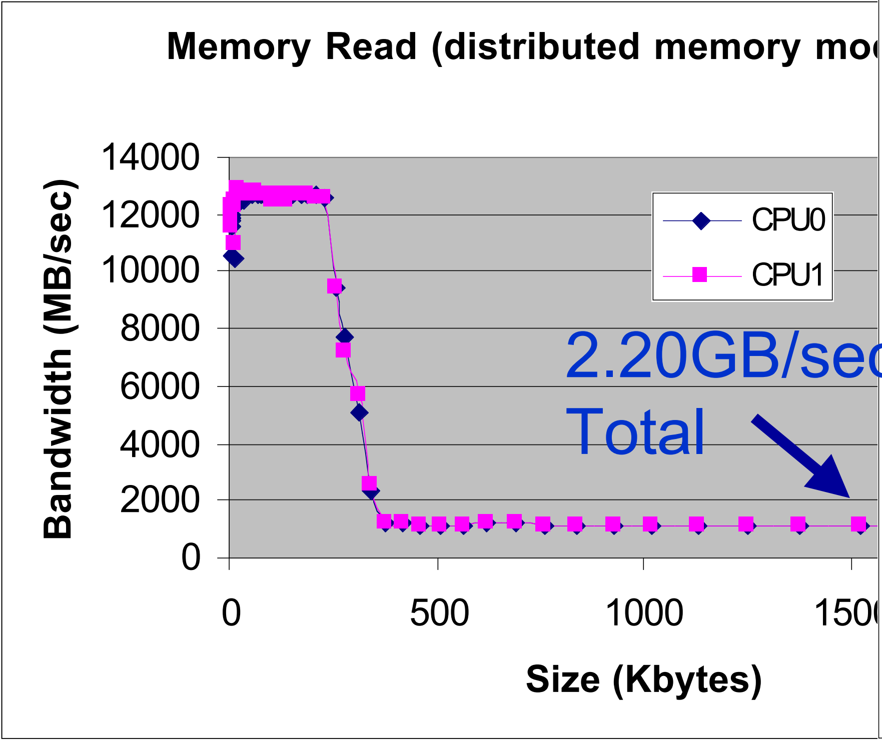
Dell Xeons
(July, 2003)

System used
In following



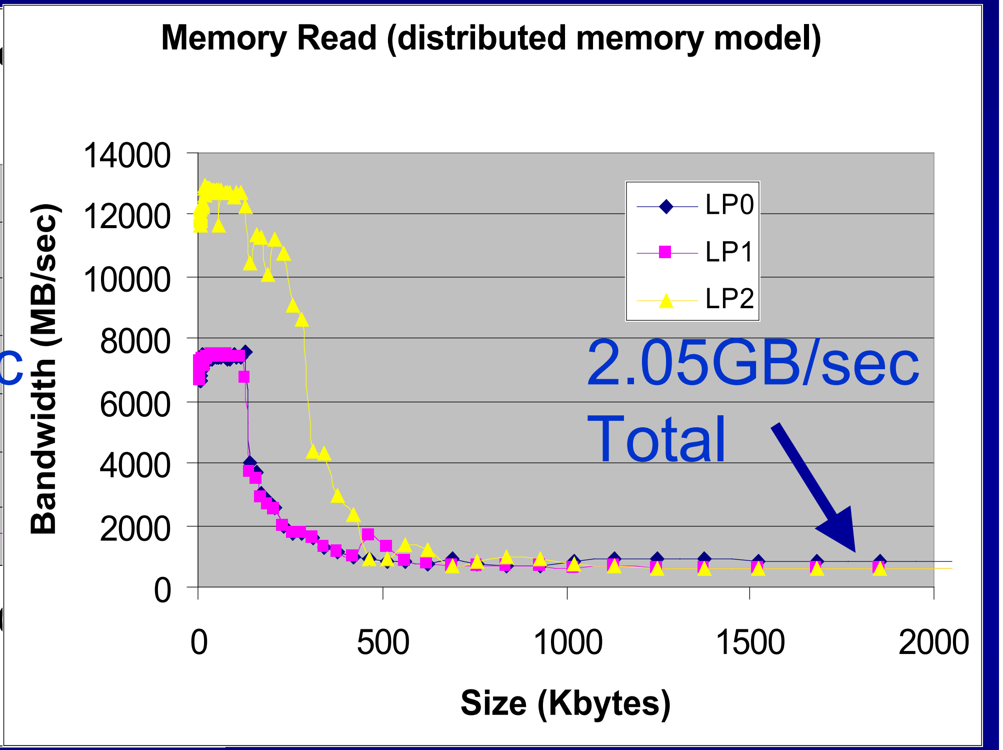
Experimental
Cluster

HT Memory Bandwidth Measurement



Memory Read (distributed memory mo

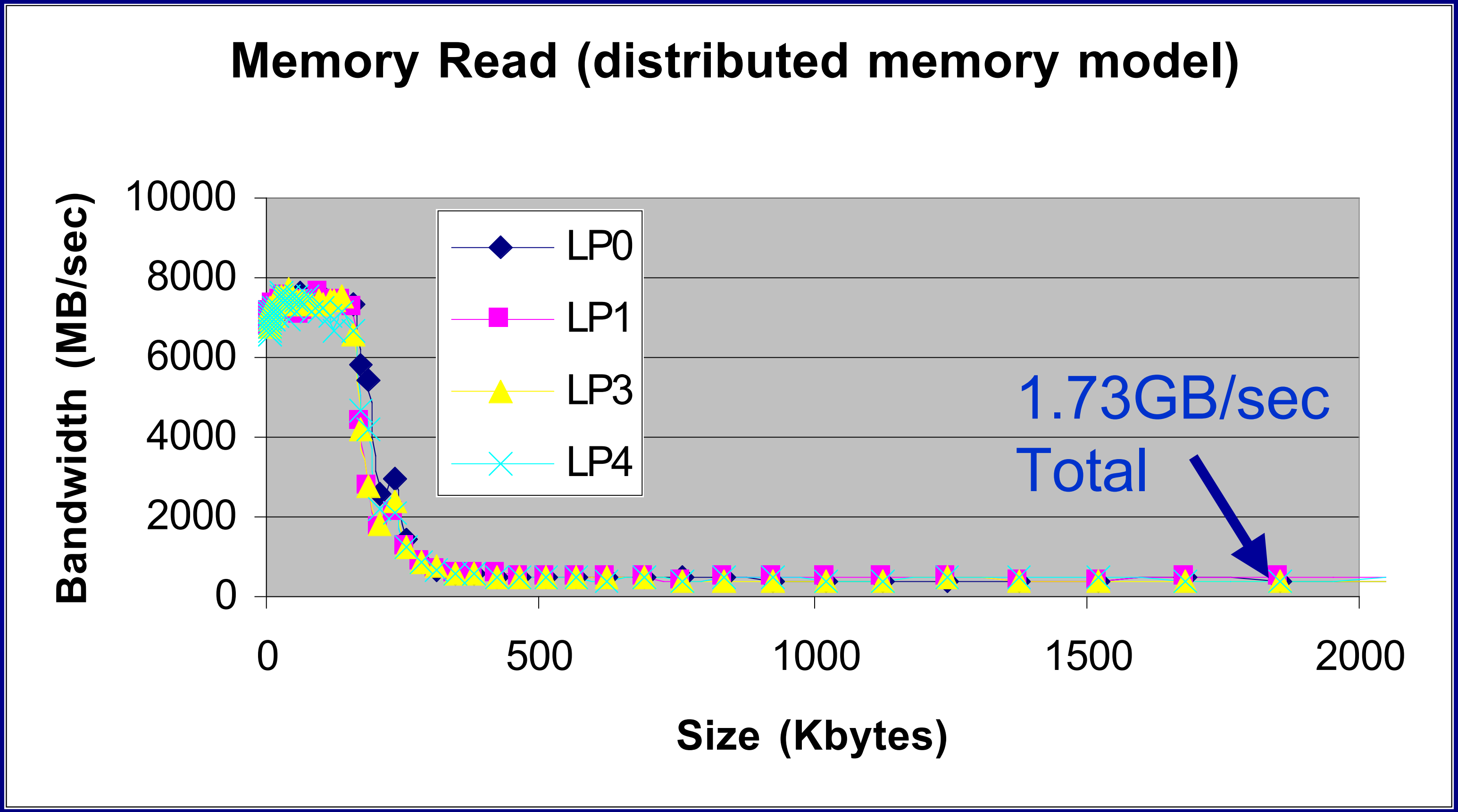
2.20GB/sec Total



Memory Read (distributed memory model)

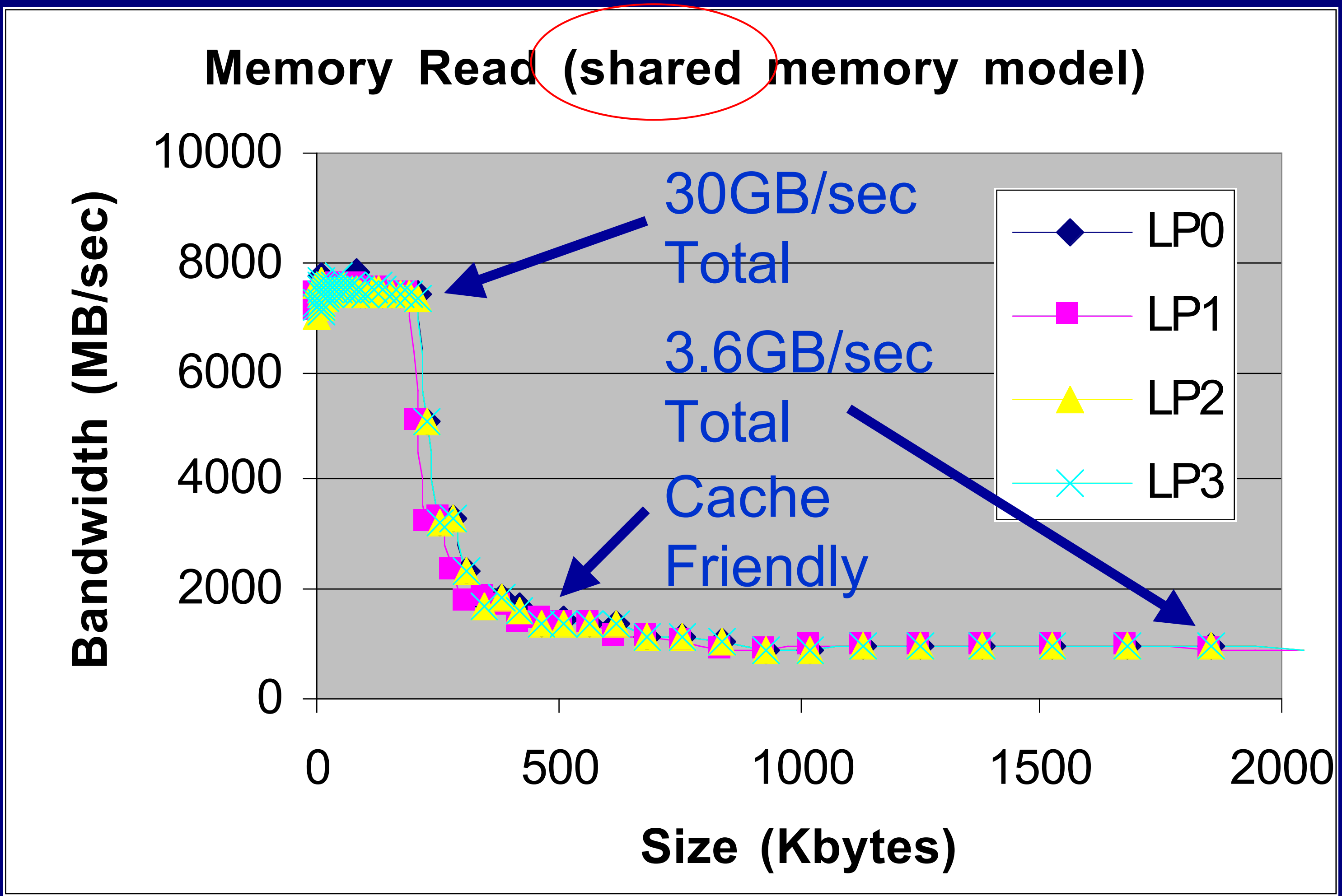
2.05GB/sec Total

HT Memory Bandwidth Measurement



1.73GB/sec Total

HT Memory Bandwidth Measurement

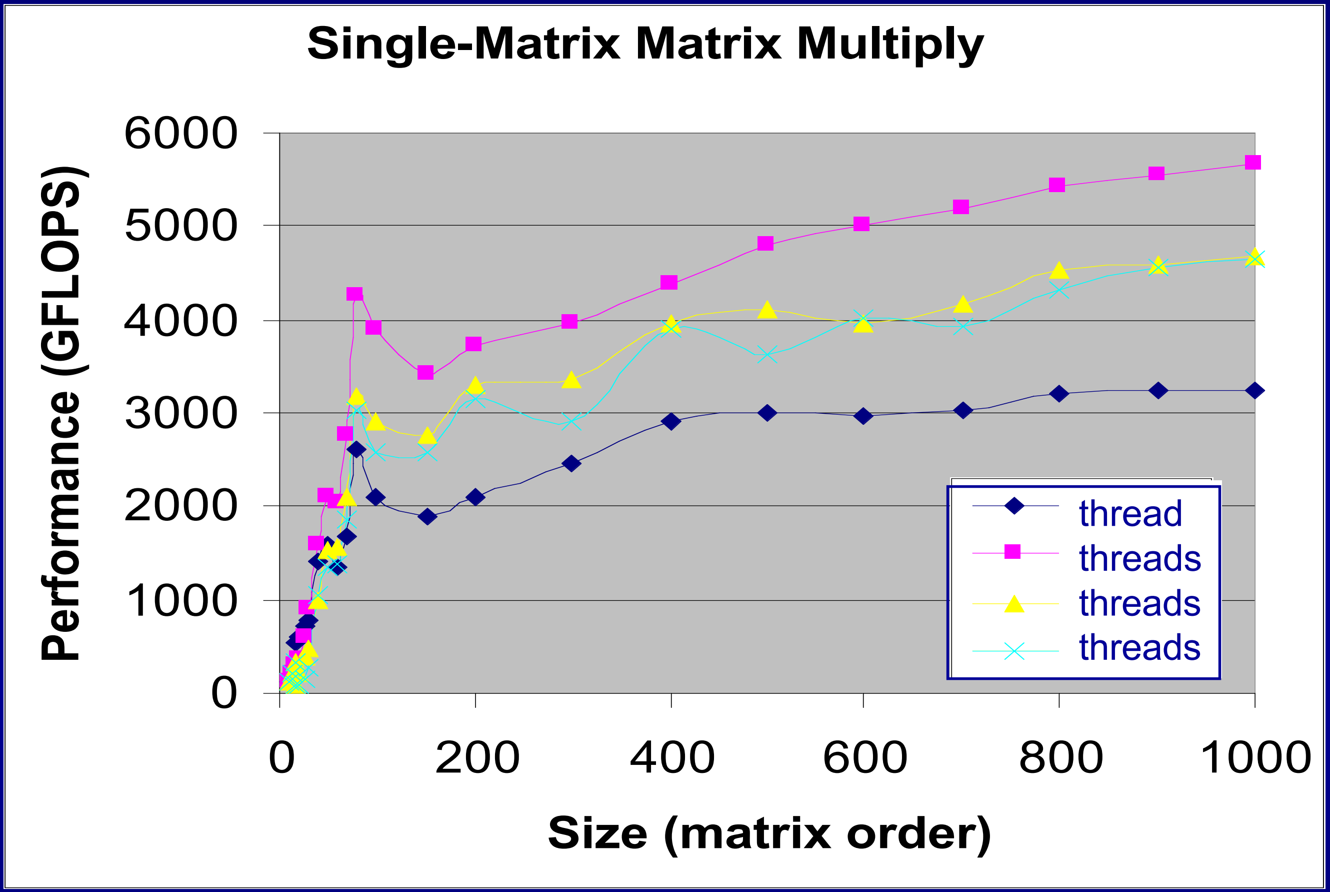


30GB/sec Total

3.6GB/sec Total

Cache Friendly

Parallel Matrix-Matrix Multiply



TACC

The University of Texas
At Austin

Molecular Dynamics

Molecular dynamics simulation of a 256 particle argon lattice
One picosecond
Verlet algorithm

Threading	1 Thread	2 Threads	3 Threads **HT**	4 Threads **HT**
Time(sec):	7.95	4.06	3.89	3.63
Scaling:	1	1.96	2.04	2.19

Monte Carlo Lithography Simulation

10**7 Monte Carlo iterations
Each thread outputs the lattice configuration and various Monte Carlo statistics to disk.

Threading	1 Thread	2 Threads	3 Threads HT	4 Threads HT
Time(sec):	19.9	15.7	13.1	11.5
Scaling:	1	1.27	1.52	1.73

Conclusions
HT Performance

- Multiple “independent” Processes/Tasks/Threads are necessary.
- Bandwidth-limited and/or Compute Intensive codes may see degradation of performance.
- Non-extreme Code may see performance enhancements
  - Cache sharing (synchronization may be required)
  - Non-Streaming memory access
  - I/O
  - Disparate operations (e.g., integer mult. & float mult.)