

Hyper-Threading Technology in the Netburst™ Microarchitecture

**Debbie Marr
Hyper-Threading Technology Architect**

Intel Corporation

August 19, 2002

Agenda

- **Hyper-Threading Technology in the Netburst™ Microarchitecture**
- **Microarchitecture Choices & Tradeoffs**
- **Performance Results**

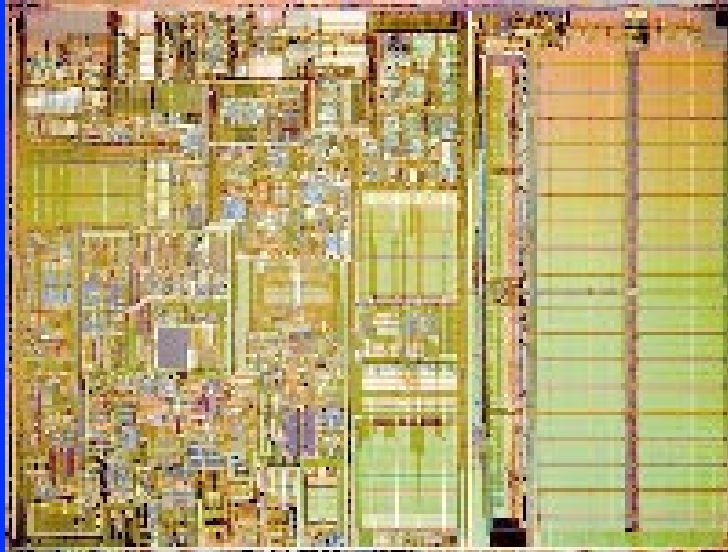
Agenda

- **Hyper-Threading Technology in the Netburst™ Microarchitecture**
- **Microarchitecture Choices & Tradeoffs**
- **Performance Results**

Hyper-Threading Technology

- **Simultaneous Multi-threading**
 - 2 logical processors (LP) simultaneously share one physical processor's execution resources
- **Appears to software as 2 processors (2-way shared memory multiprocessor)**
 - Shrink-wrapped operating system schedules software threads/processes to both logical processors
 - Fully compatible to existing multi-processor system software and hardware.
- **Integral part of Intel Netburst™ Microarchitecture**

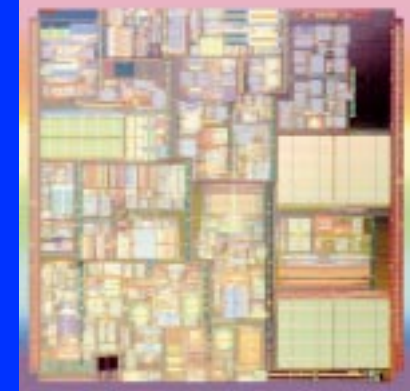
Intel® Processors with Netburst™ Microarchitecture



Intel® Xeon™ MP Processor
256KB 2nd-Level Cache
1MB 3rd-Level Cache
.18u process



Intel Xeon Processor
256KB 2nd-Level Cache
.18u process



Intel Xeon Processor
512KB 2nd-Level Cache
.13u process

Die Size Increase is Small

- Total die area added is small
 - A few small structures duplicated
 - Some additional control logic and pointers

What was added

Instruction Streaming Buffers

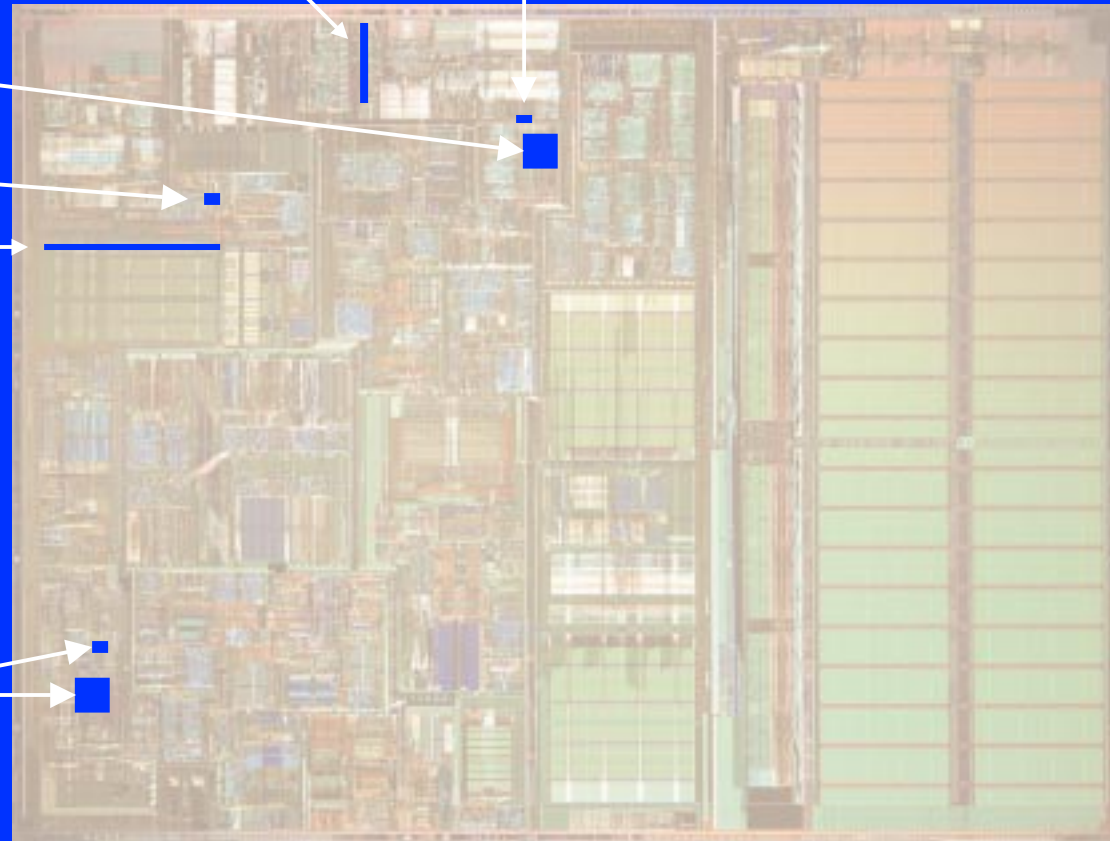
Next IP

Instruction TLB

Trace Cache Next IP

Trace Cache
Fill Buffers

Register Alias
Tables



Complexity is Large

- **Challenged many basic assumptions**
- **New microarchitecture algorithms**
 - To address new uop (micro-operation) prioritization issues
 - To solve potential new livelock scenarios
- **High logic design complexity**
- **Validation Effort**
 - Explosion of validation space

Agenda

- **Hyper-Threading Technology in the Netburst™ Microarchitecture**
- **Microarchitecture Choices & Tradeoffs**
- **Performance Results**

Managing Resources

- **Choices**

- **Partition**

- Half of resource dedicated to each logical processor

- **Threshold**

- Flexible resource sharing with limit on maximum resource usage

- **Full Sharing**

- Flexible resource sharing with *no* limit on maximum resource usage

- **Others (not discussed in this talk)**

- **Considerations**

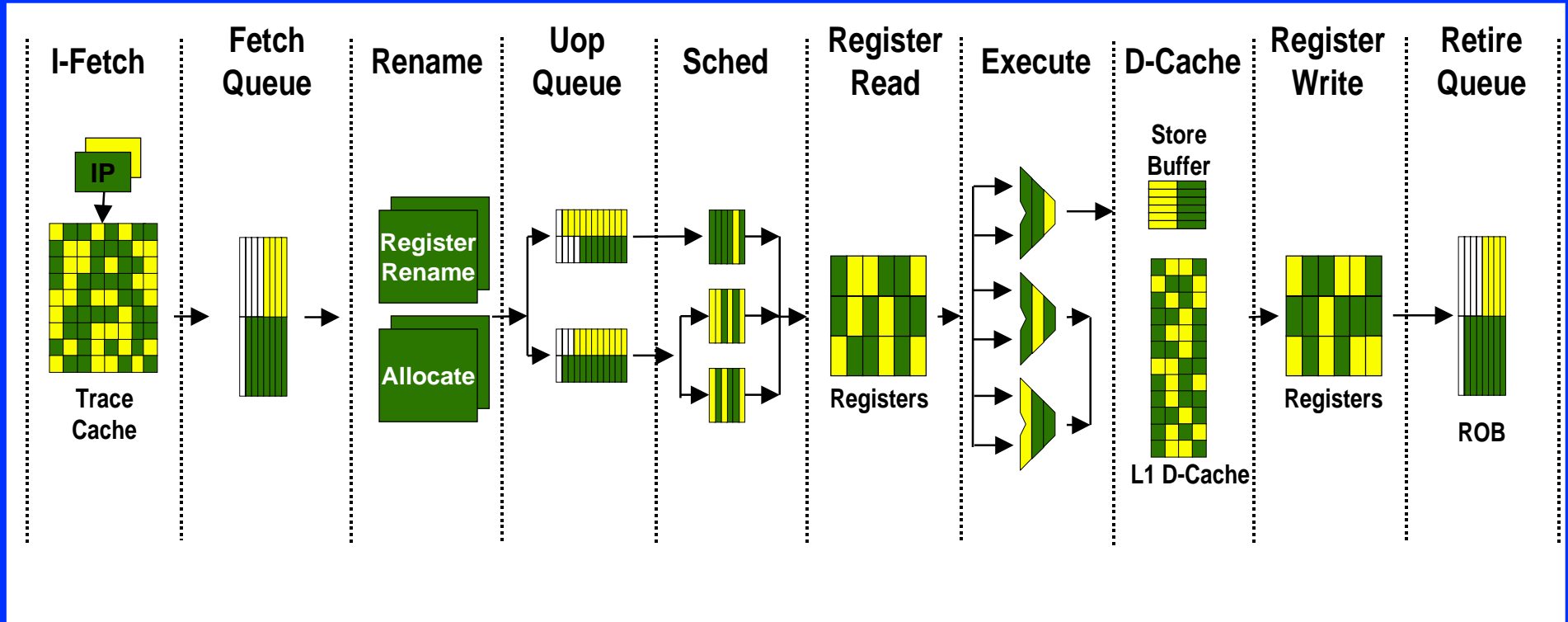
- **Throughput and fairness**

- **Die size and Complexity**

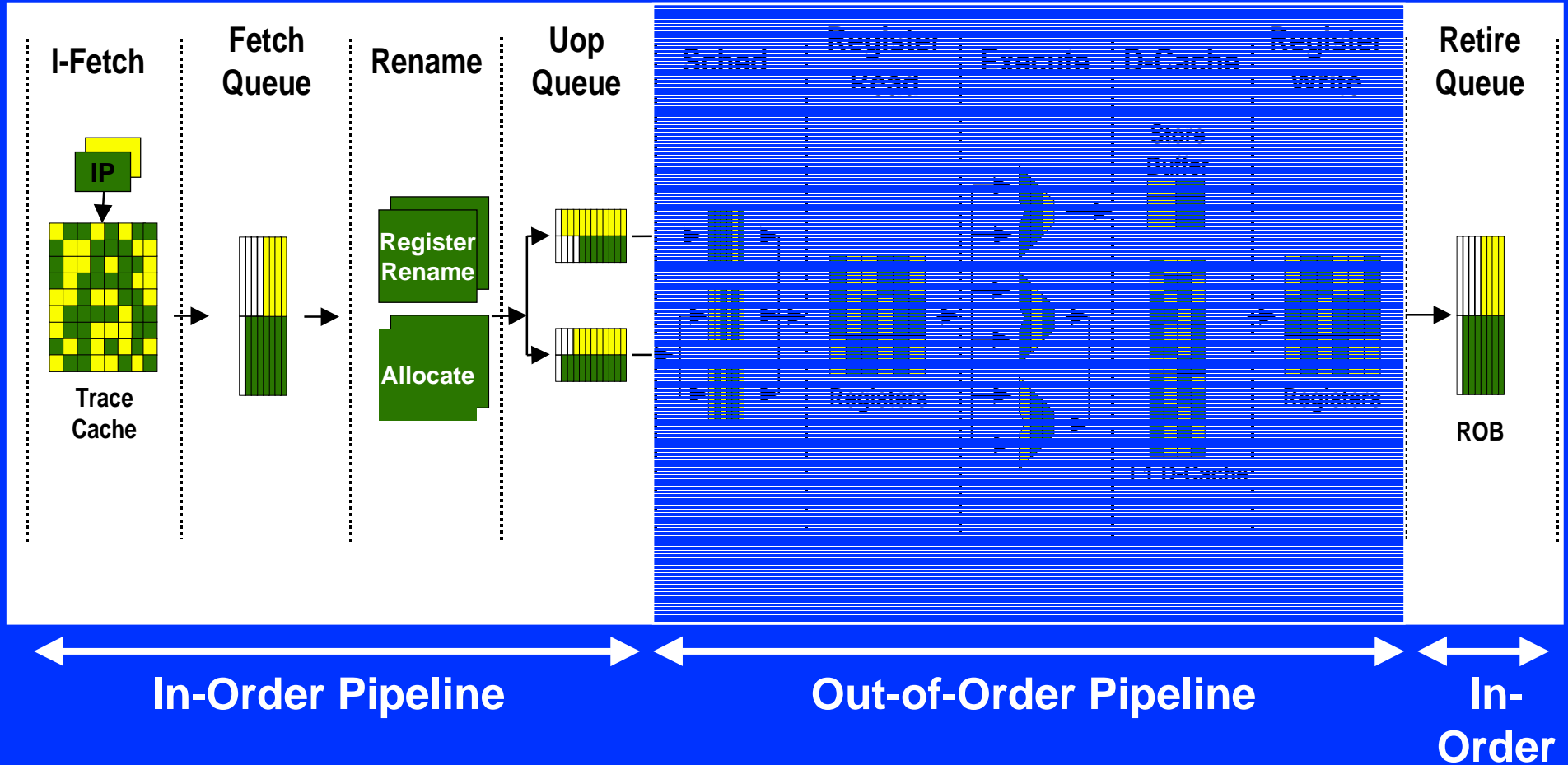
Partitioning

- **Half of resource dedicated to each logical processor**
 - Simple, low complexity
- **Good for structures where**
 - Occupancy time can be high and unpredictable
 - High average utilization
- **Major pipeline queues are a good example**
 - Provide buffering to avoid pipeline stalls
 - Allow slip between logical processors

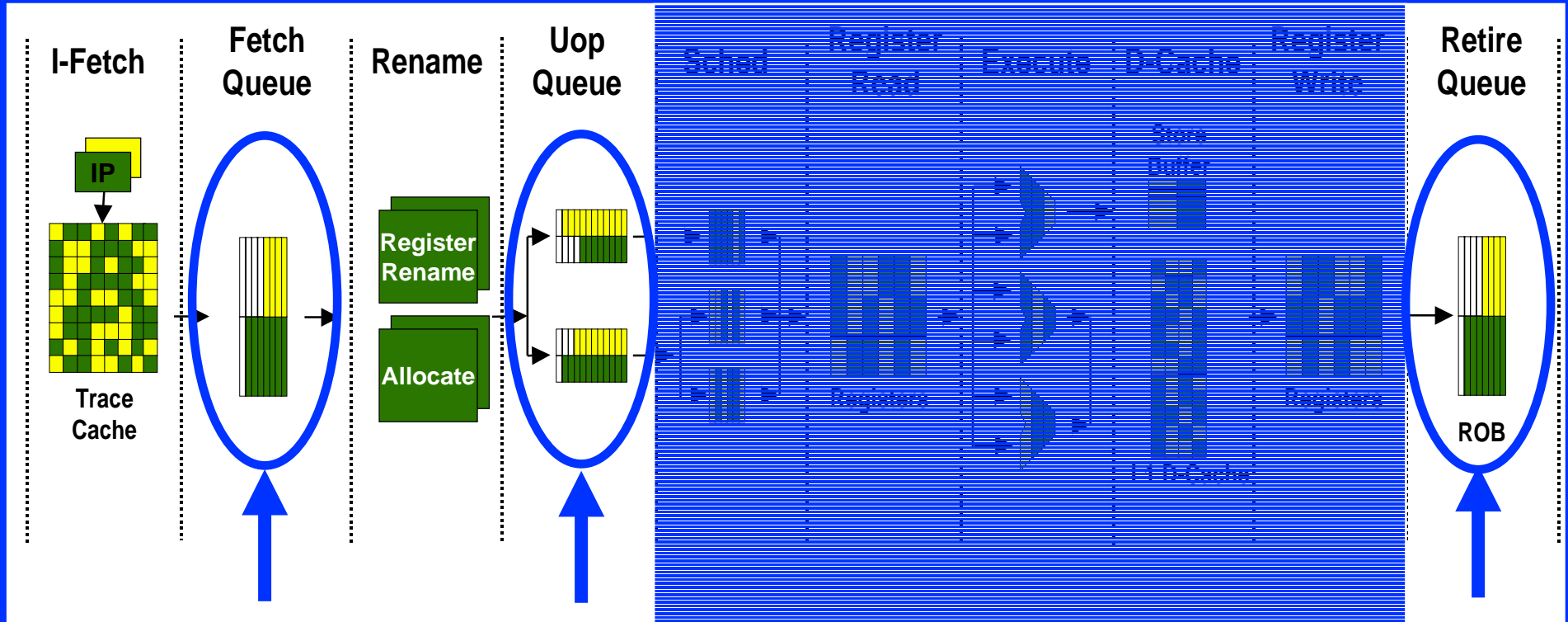
Execution Pipeline



Execution Pipeline



Execution Pipeline



Partition queues between major pipestages of pipeline

Partitioned Queue Example

- **With full sharing, a slow thread can get unfair share of resources**
 - **Can prevent a faster thread from making rapid progress.**

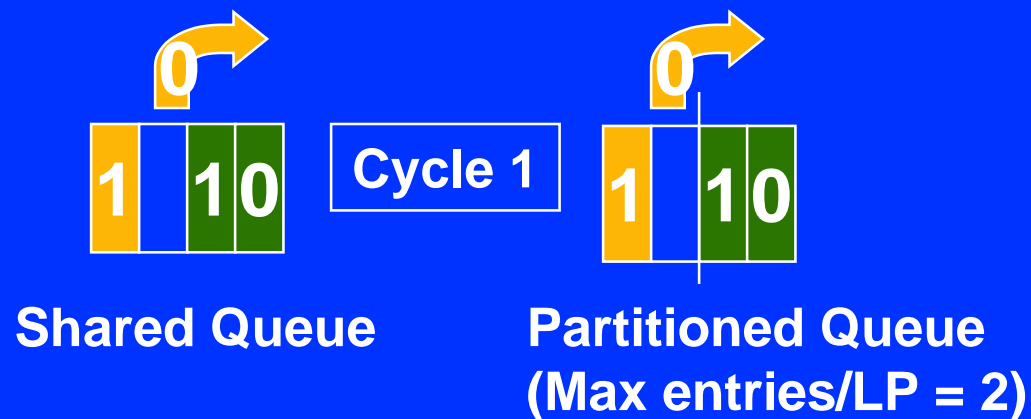
Partitioned Queue Example

- Green thread stalled
- Yellow thread not stalled



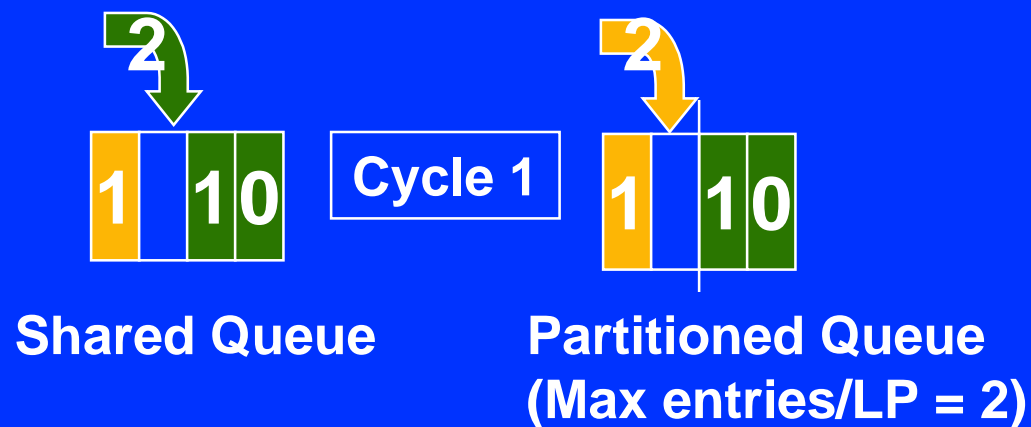
Partitioned Queue Example

- Green thread stalled
- Yellow thread not stalled



Partitioned Queue Example

- Green thread stalled
- Yellow thread not stalled



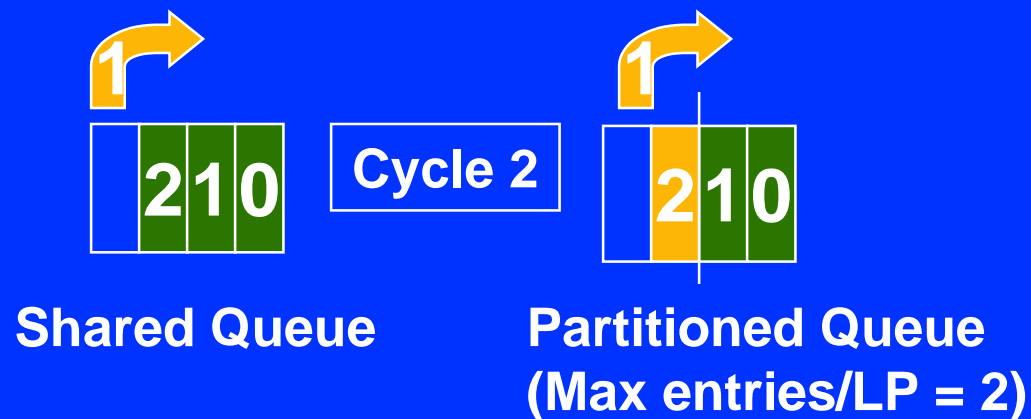
Partitioned Queue Example

- Green thread stalled
- Yellow thread not stalled



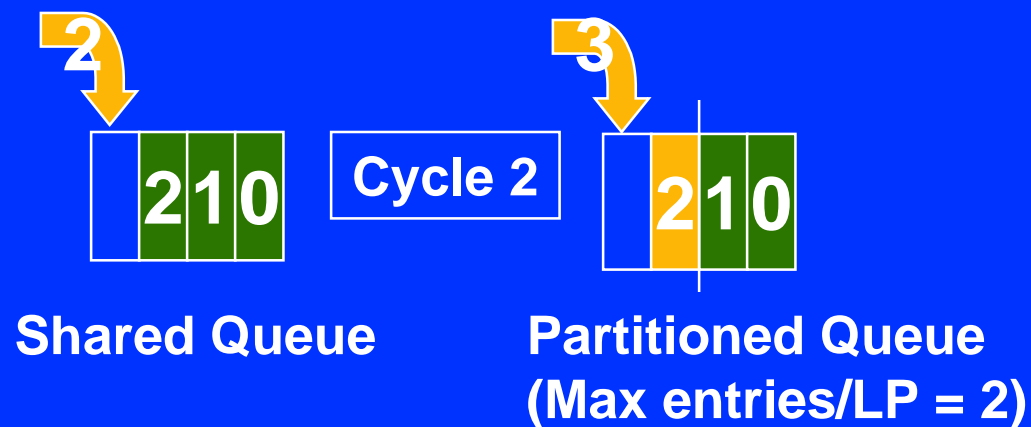
Partitioned Queue Example

- Green thread stalled
- Yellow thread not stalled



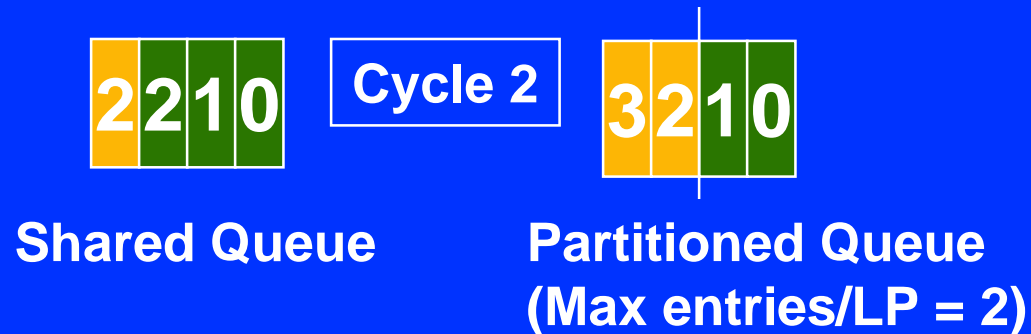
Partitioned Queue Example

- Green thread stalled
- Yellow thread not stalled



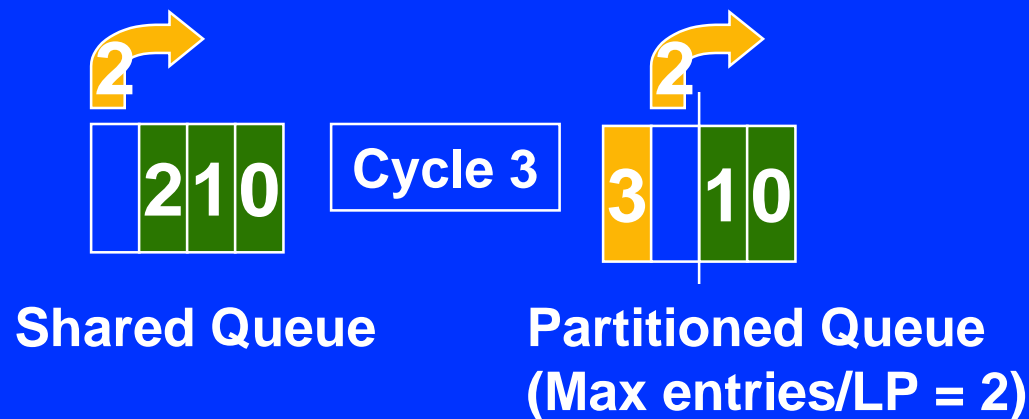
Partitioned Queue Example

- Green thread stalled
- Yellow thread not stalled



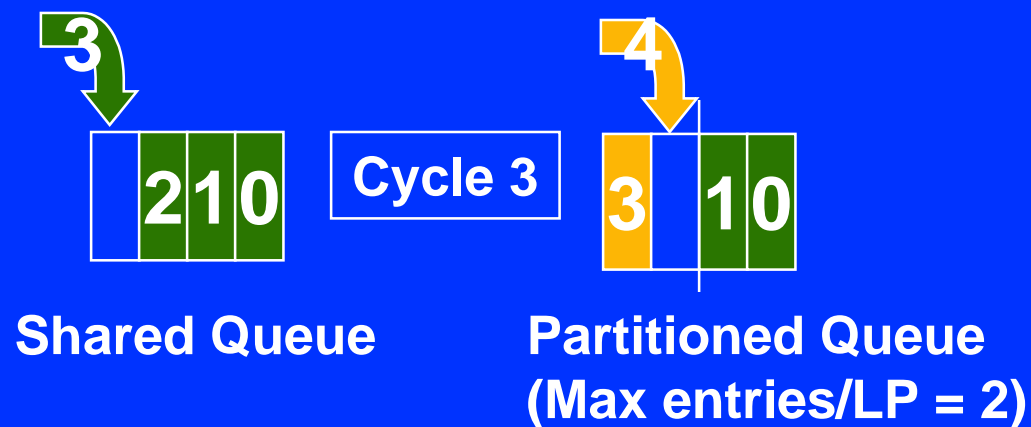
Partitioned Queue Example

- Green thread stalled
- Yellow thread not stalled



Partitioned Queue Example

- Green thread stalled
- Yellow thread not stalled



Partitioned Queue Example

- Green thread stalled
- Yellow thread not stalled



Partitioned Queue Example

- Green thread stalled
- Yellow thread not stalled

Partitioning resource ensures fairness and ensures progress for both logical processors

Yellow is Blocked!



Shared Queue

Cycle 4



Partitioned Queue
(Max entries/LP = 2)

Thresholds

- **Flexible resource sharing with limit on maximum resource usage**
- **Good for small structures where**
 - Occupancy time is low and predictable
 - Low average utilization with occasional high peaks
- **Schedulers are a good example**
 - Throughput is high because of data speculation (get data regardless of cache hit)
 - uOps pass through scheduler very quickly
 - Schedulers are small for speed

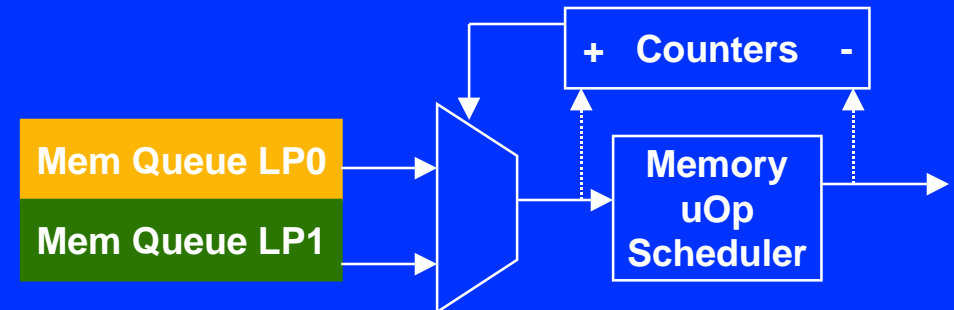
Schedulers, Queues

- 5 Schedulers

- MEM
- ALU0
- ALU1
- FP Move
- FP/MMX/SSE

- Threshold prevents one logical processor from consuming all entries

- Round robin until reach threshold



Similarly..

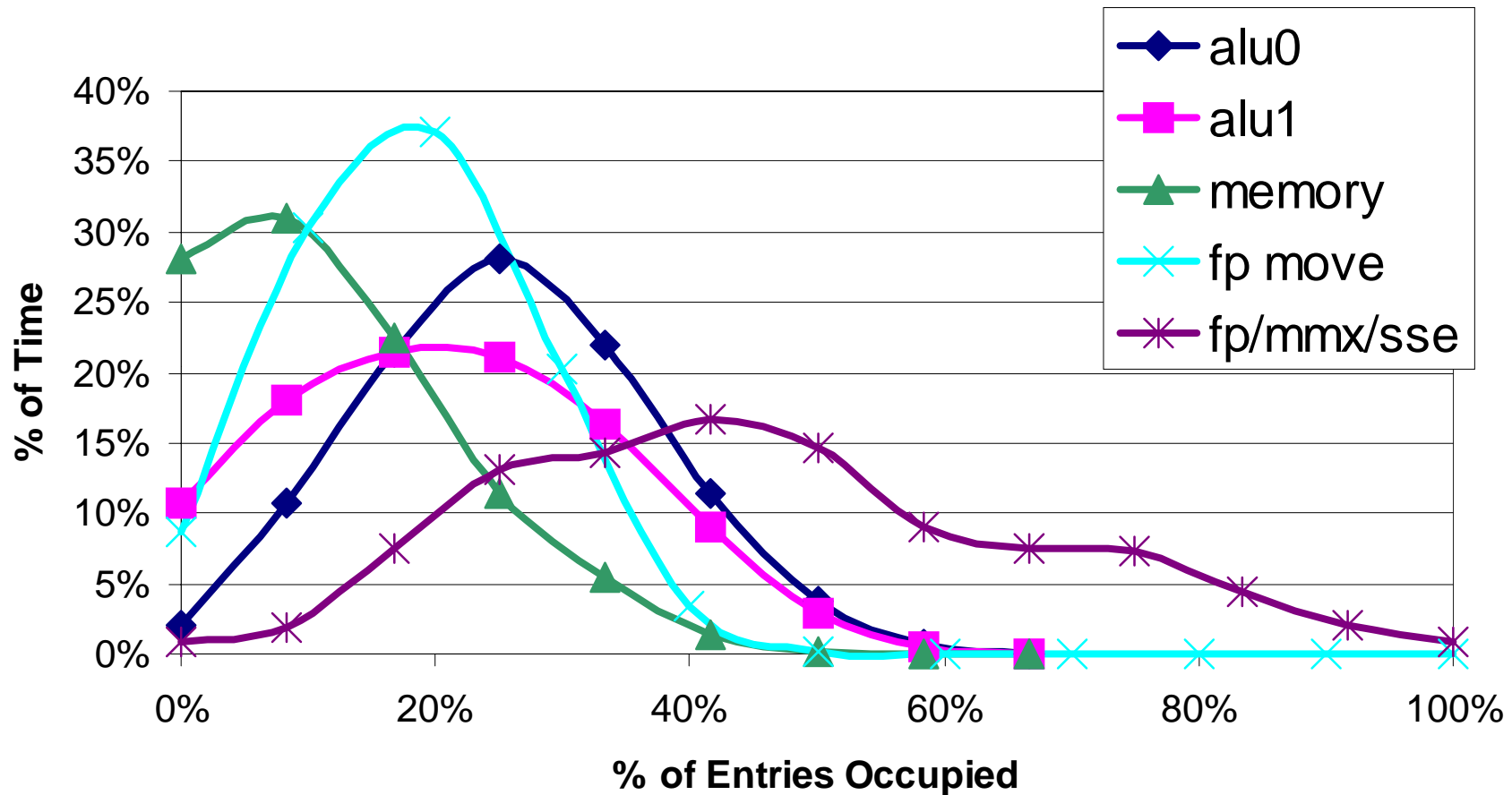
ALU0 Scheduler

ALU1 Scheduler

FP Move Scheduler

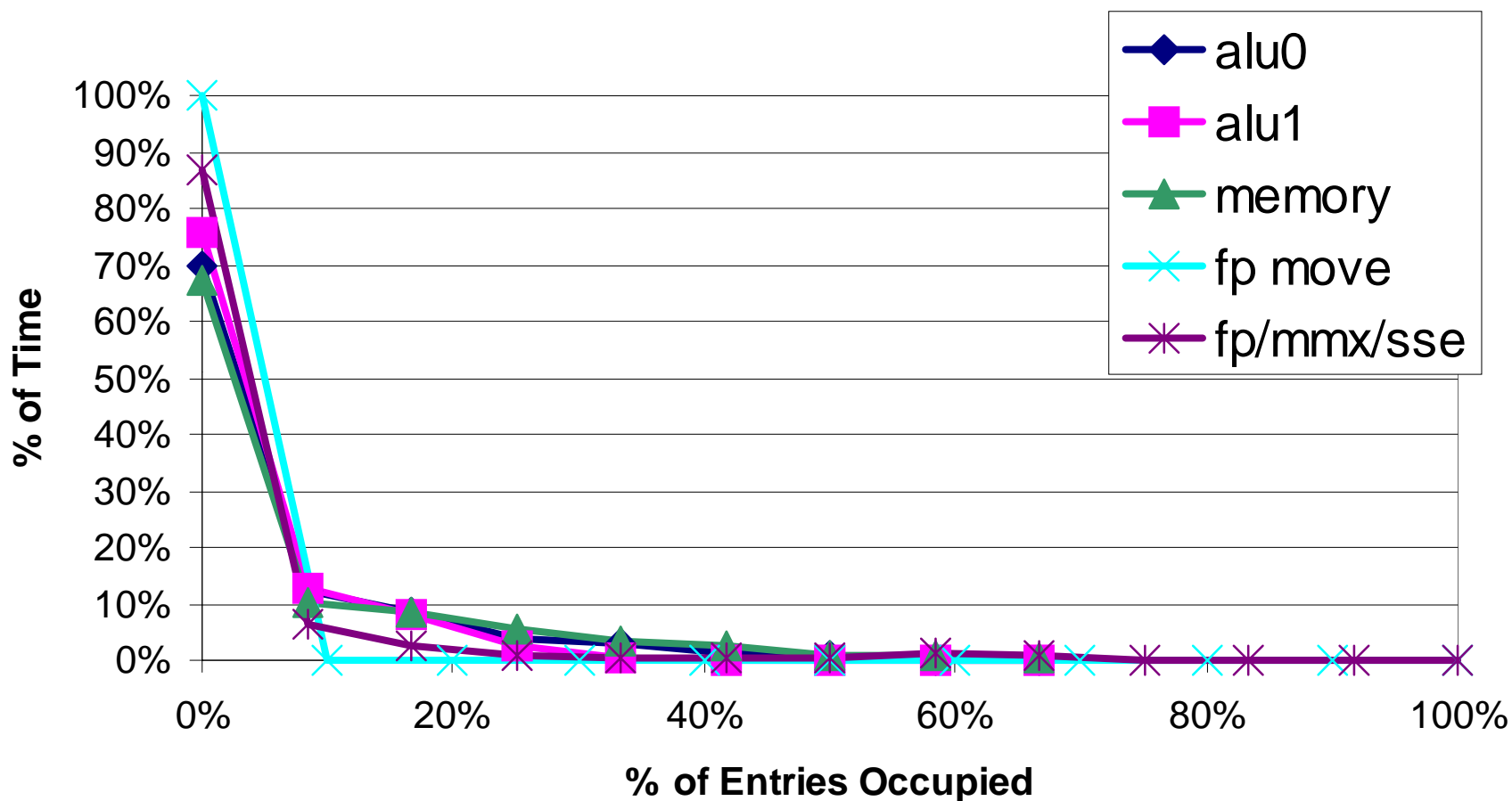
FP/MMX/SSE Scheduler

Scheduler Occupancy Histogram Image Composition Workload



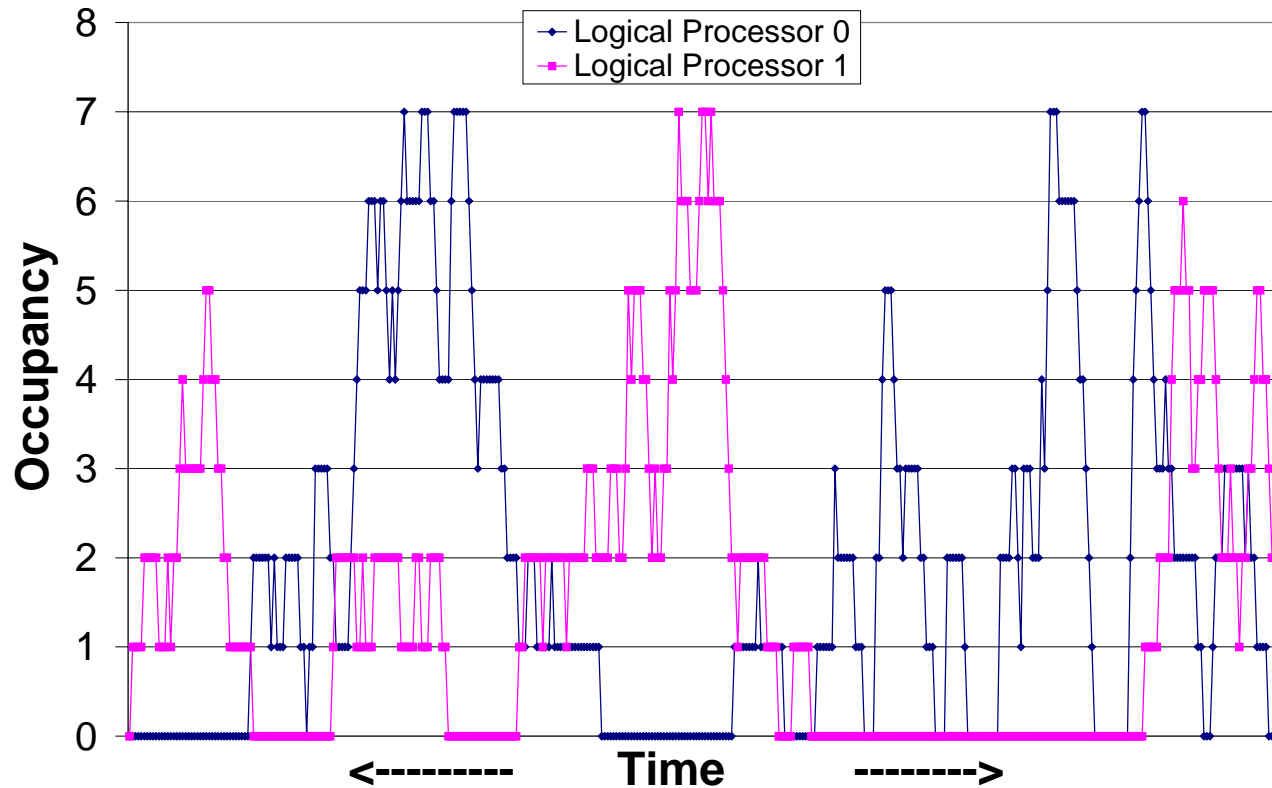
Measurement of image composition workload on an Intel® Xeon™ Processor

Scheduler Occupancy Histogram Transaction Processing Workload



Measurement of transaction processing workload on a 4P Intel® Xeon™ MP Processor System

Memory Scheduler Occupancy Over Time



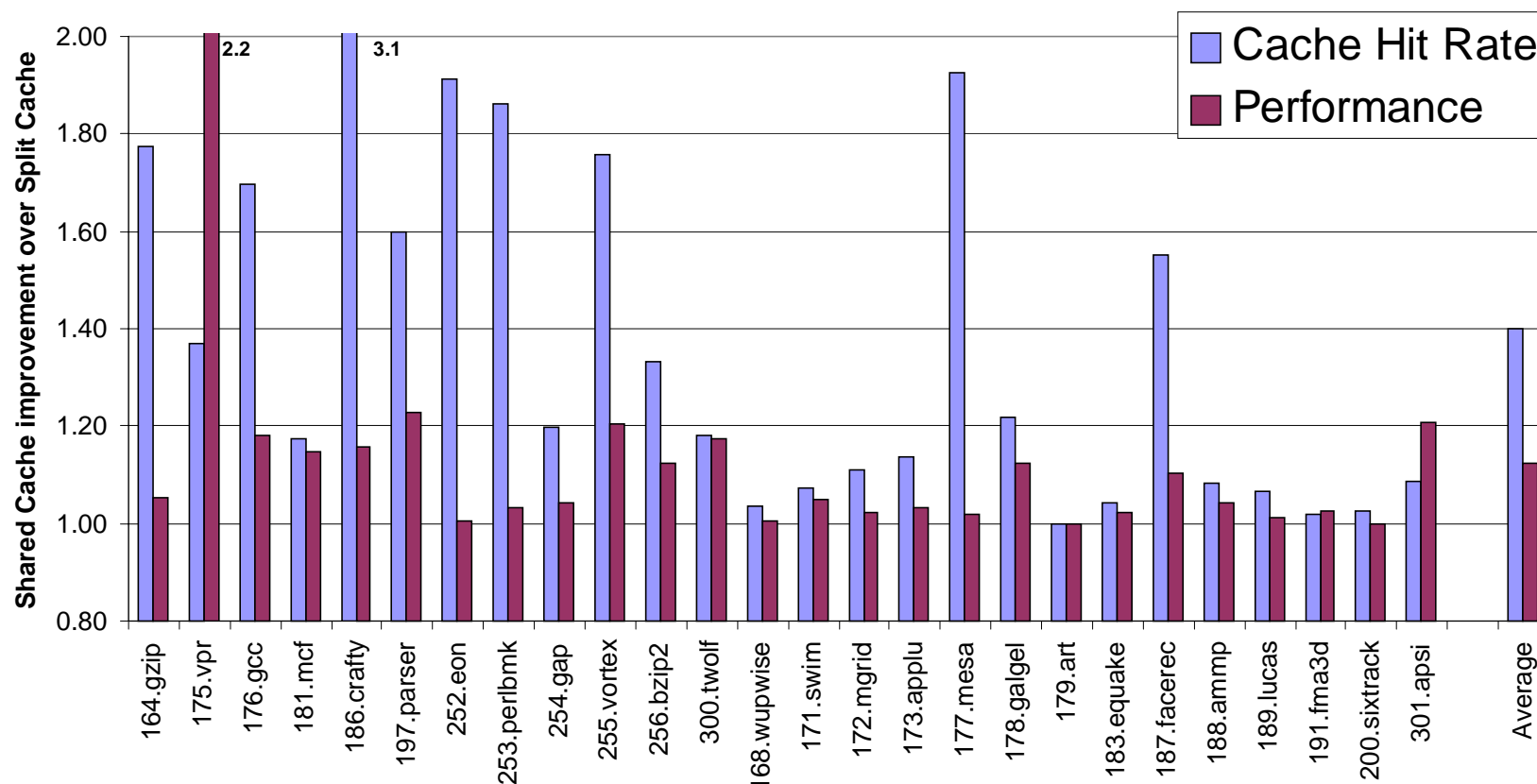
Measurement of transaction processing workload on a 4P Intel® Xeon™ MP Processor System

Variable partitioning allows a logical processor to use most resources when the other doesn't need them

Full Sharing

- **Flexible resource sharing with no limit on maximum resource usage**
- **Good for large structures where**
 - Working set sizes are variable
 - Sharing between logical processors possible
 - Not possible for one logical processor to starve
- **Caches are a good example**
 - All caches are shared
 - Better overall performance vs. partitioned caches
 - Some applications share code and/or data
 - High set associativity minimizes conflict misses.
 - Level 2 and Level 3 caches are 8-way set associative

Shared Cache vs. Partitioned Cache



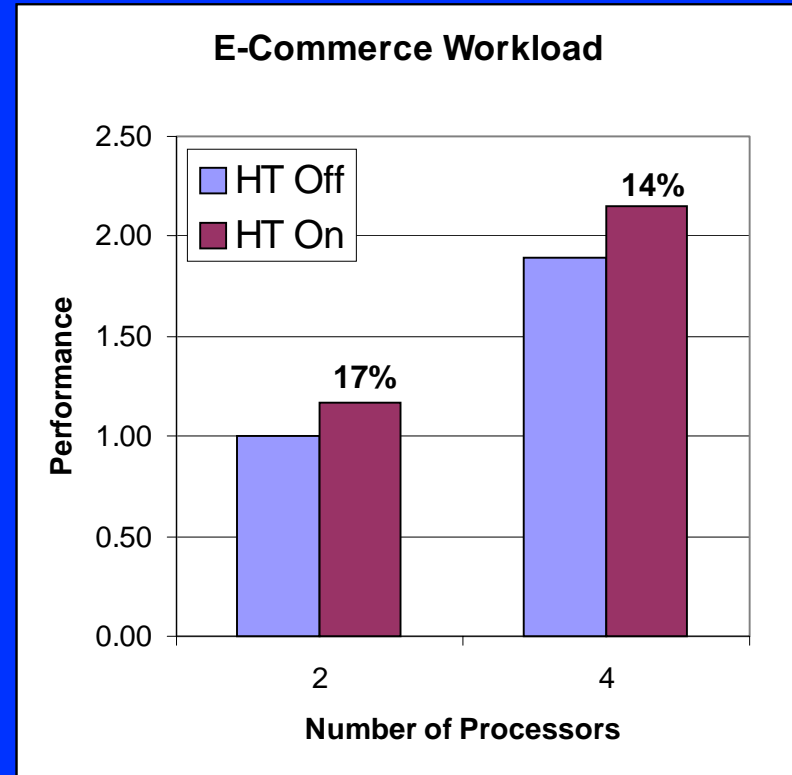
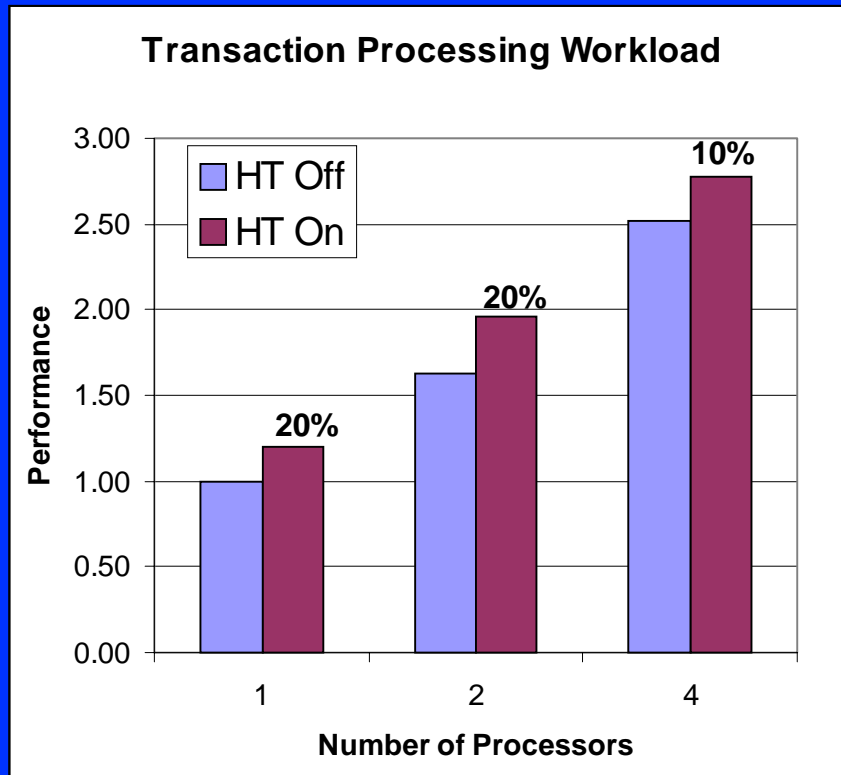
Results for 2 copies of application run simultaneously. Measured on Intel® Xeon™ Processor C0 step.
Cache miss statistics using EMON event: MN MEM:2nd Level Cache Load Misses Retired

On average, a shared cache has 40% better hit rate and 12% better performance for these applications

Agenda

- **Hyper-Threading Technology in the Netburst™ Microarchitecture**
- **Microarchitecture Choices & Tradeoffs**
- **Performance Results**

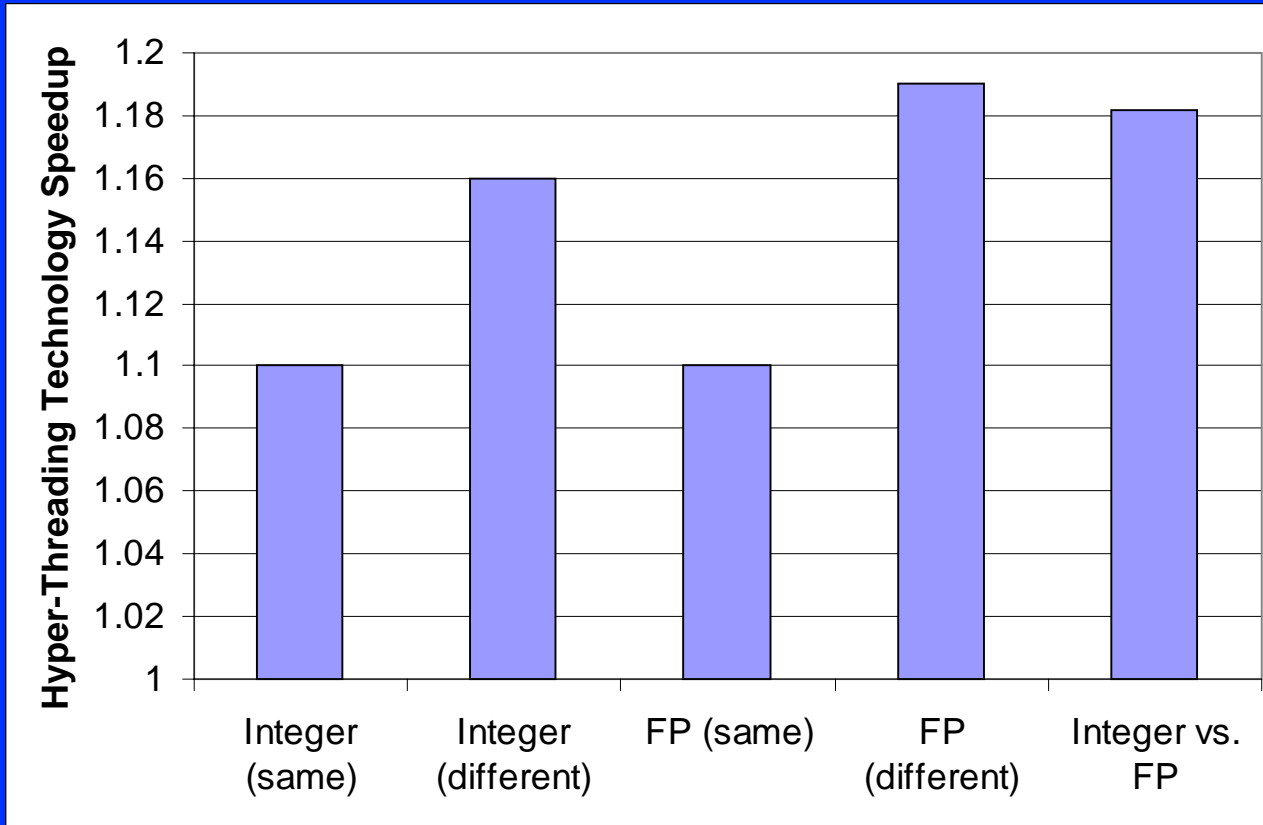
Server Performance



Good performance benefit from small die area investment

Performance tests and ratings are measured using specific computer systems and/or components and reflect the approximate performance of Intel products as measured by those tests. Any difference in system hardware or software design or configuration may affect actual performance. Buyers should consult other sources of information to evaluate the performance of systems or components they are considering purchasing. For more information on performance tests and on the performance of Intel products, reference www.intel.com/procs/perf/limits.htm or call (U.S.) 1-800-628-8686 or 1-916-356-3104.

Multi-tasking



Intel® Xeon™ Processor platform is a protobly system

Larger gains can be realized by running dissimilar applications due to different resource requirements

Performance tests and ratings are measured using specific computer systems and/or components and reflect the approximate performance of Intel products as measured by those tests. Any difference in system hardware or software design or configuration may affect actual performance. Buyers should consult other sources of information to evaluate the performance of systems or components they are considering purchasing. For more information on performance tests and on the performance of Intel products, reference www.intel.com/procs/perf/limits.htm or call (U.S.) 1-800-628-8686 or 1-916-356-3104.

Conclusions

- **Hyper-Threading Technology is an integral part of the Netburst™ Microarchitecture**
 - Very little additional die area needed
 - Compelling performance
 - Currently enabled for server processors
- **Microarchitecture design choices**
 - Resource sharing policy matched to traffic and performance requirements
- **New challenging microarchitecture direction**
 - Continuous improvements in future processors for years to come