Simultaneous Multi-threading Implementation in POWER5 -- IBM's Next Generation POWER Microprocessor

Ron Kalla, Balaram Sinharoy, Joel Tendler
IBM Systems Group

Outline

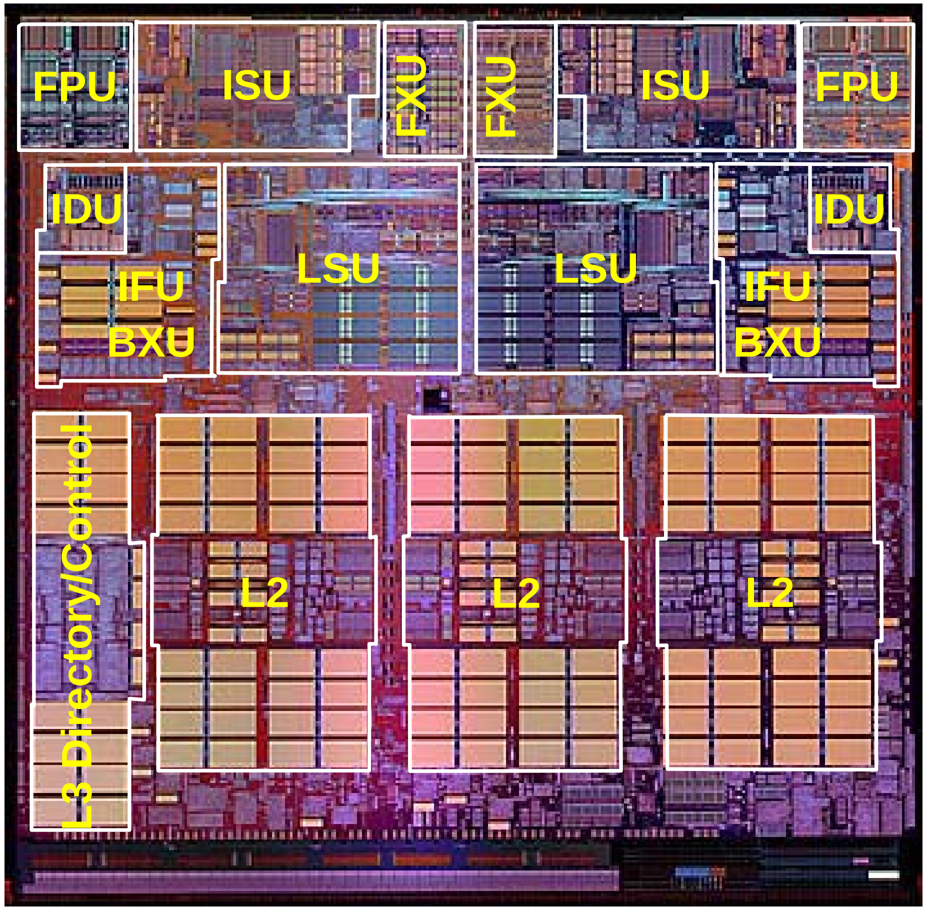
- Motivation
- Background
- Threading Fundamentals
- Enhanced SMT Implementation in POWER5
- Additional SMT Considerations
- Summary

Microprocessor Design Optimization Focus Areas

- Memory latency
  - Increased processor speeds make memory appear further away
  - Longer stalls possible
- Branch processing
  - Mispredict more costly as pipeline depth increases resulting in stalls and wasted power
  - Predication drives increased power and larger chip area
- Execution Unit Utilization
  - Currently 20-25% execution unit utilization common
- Simultaneous multi-threading (SMT) and POWER architecture address these areas

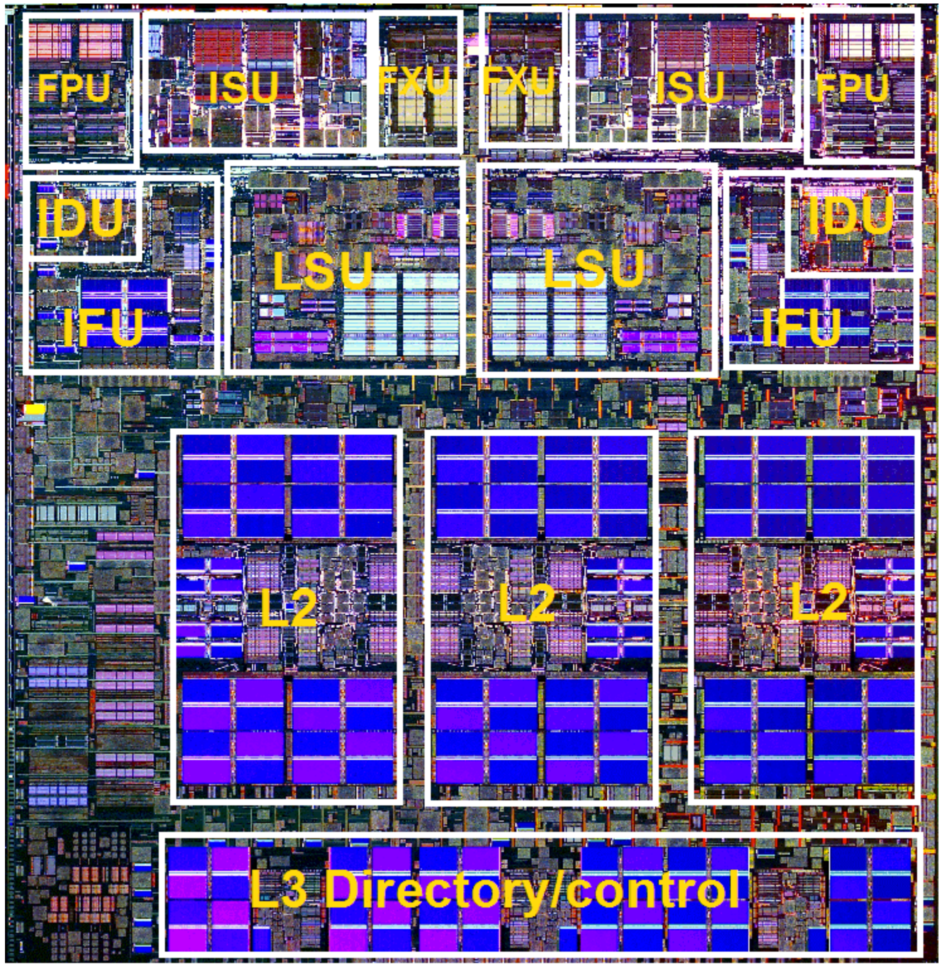
POWER4 --- Shipped in Systems December 2001

- Technology: 180nm lithography, Cu, SOI
  - POWER4+ shipping in 130nm today
- Dual processor core
- 8-way superscalar
  - Out of Order execution
  - 2 Load / Store units
  - 2 Fixed Point units
  - 2 Floating Point units
  - Logical operations on Condition Register
  - Branch Execution unit
- > 200 instructions in flight
- Hardware instruction and data prefetch



POWER5 --- The Next Step

- Technology: 130nm lithography, Cu, SOI
- Dual processor core
- 8-way superscalar
- Simultaneous multithreaded (SMT) core
  - Up to 2 virtual processors per real processor
  - 24% area growth per core for SMT
  - Natural extension to POWER4 design



Multi-threading Evolution

Single Thread

FX0	🟦	⬜	⬜	🟦	⬜	⬜	⬜	⬜
FX1	⬜	🟦	⬜	⬜	⬜	⬜	🟦	⬜
FP0	⬜	⬜	⬜	⬜	⬜	⬜	🟦	⬜
FP1	⬜	⬜	🟦	⬜	🟦	⬜	⬜	⬜
LS0	🟦	⬜	⬜	⬜	⬜	⬜	🟦	🟦
LS1	⬜	⬜	⬜	🟦	⬜	⬜	⬜	⬜
BRX	⬜	🟦	⬜	⬜	⬜	🟦	⬜	🟦
CRL	⬜	⬜	⬜	🟦	⬜	⬜	⬜	⬜

Coarse Grain Threading

FX0	🟦	⬜	⬜	⬜	⬜	⬜	⬜	⬜
FX1	⬜	🟦	⬜	⬜	⬜	⬜	🟥	⬜
FP0	⬜	⬜	⬜	⬜	⬜	⬜	🟥	⬜
FP1	⬜	⬜	⬜	⬜	🟥	⬜	⬜	⬜
LS0	🟦	🟦	⬜	⬜	⬜	⬜	🟥	⬜
LS1	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜
BRX	⬜	🟦	⬜	⬜	⬜	🟥	⬜	⬜
CRL	⬜	⬜	⬜	⬜	🟥	⬜	⬜	⬜

Fine Grain Threading

FX0	🟦	⬜	⬜	🟥	⬜	⬜	⬜	⬜
FX1	⬜	🟥	⬜	⬜	⬜	⬜	🟦	🟥
FP0	⬜	⬜	⬜	⬜	⬜	🟥	🟦	⬜
FP1	⬜	⬜	🟦	⬜	🟦	⬜	⬜	⬜
LS0	🟦	⬜	⬜	⬜	⬜	⬜	🟦	🟥
LS1	⬜	⬜	⬜	🟥	🟦	⬜	⬜	⬜
BRX	⬜	🟥	🟦	⬜	⬜	🟥	⬜	🟥
CRL	⬜	⬜	⬜	🟥	⬜	⬜	⬜	⬜

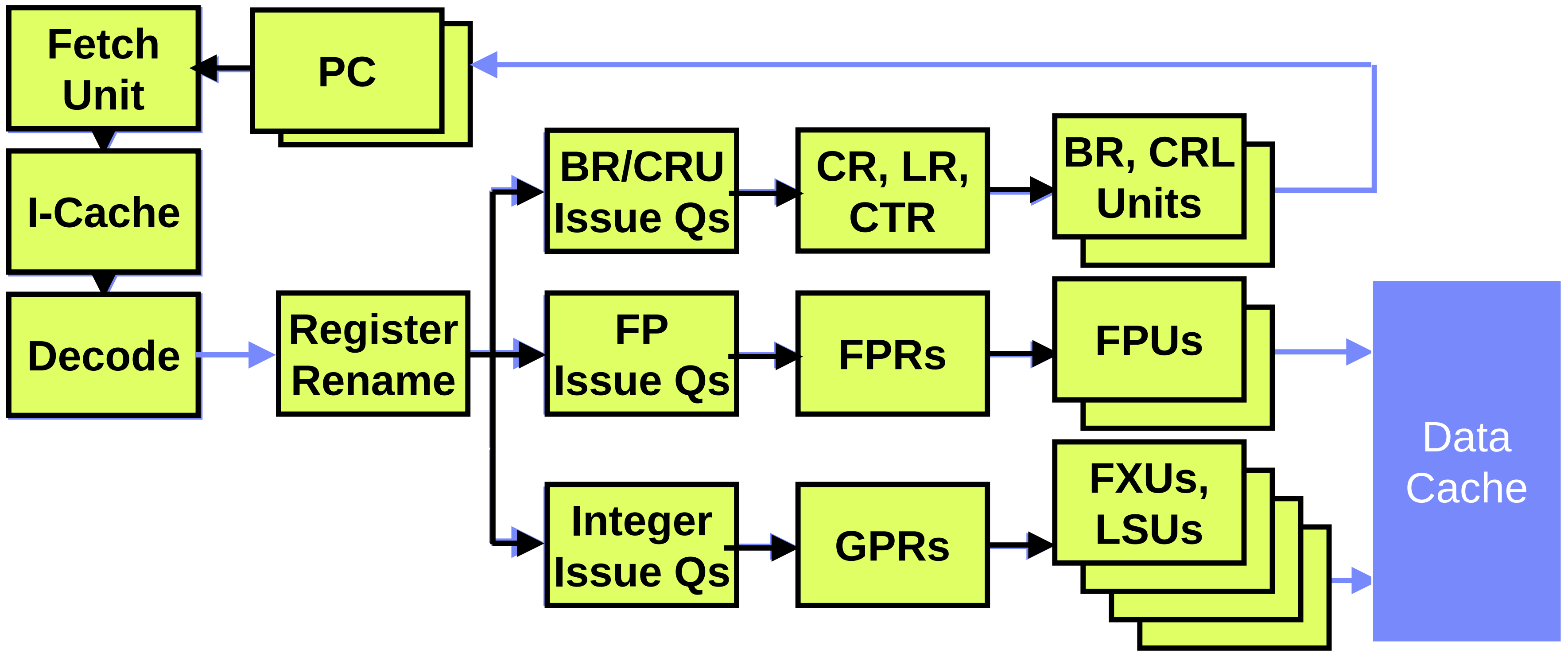
Simultaneous Multi-Threading

FX0	🟦	⬜	⬜	🟥	⬜	⬜	⬜	🟦
FX1	⬜	🟥	⬜	⬜	⬜	⬜	🟥	⬜
FP0	⬜	🟦	⬜	⬜	⬜	🟥	🟦	⬜
FP1	⬜	⬜	🟦	⬜	🟥	⬜	⬜	⬜
LS0	🟦	⬜	⬜	⬜	⬜	🟦	🟦	🟥
LS1	🟥	⬜	⬜	🟥	🟦	⬜	⬜	⬜
BRX	⬜	🟦	🟦	⬜	⬜	🟥	🟥	🟥
CRL	⬜	⬜	⬜	🟥	⬜	⬜	⬜	⬜

🟦 **Thread 0 Executing** 🟥 **Thread 1 Executing** ⬜ **No Thread Executing**

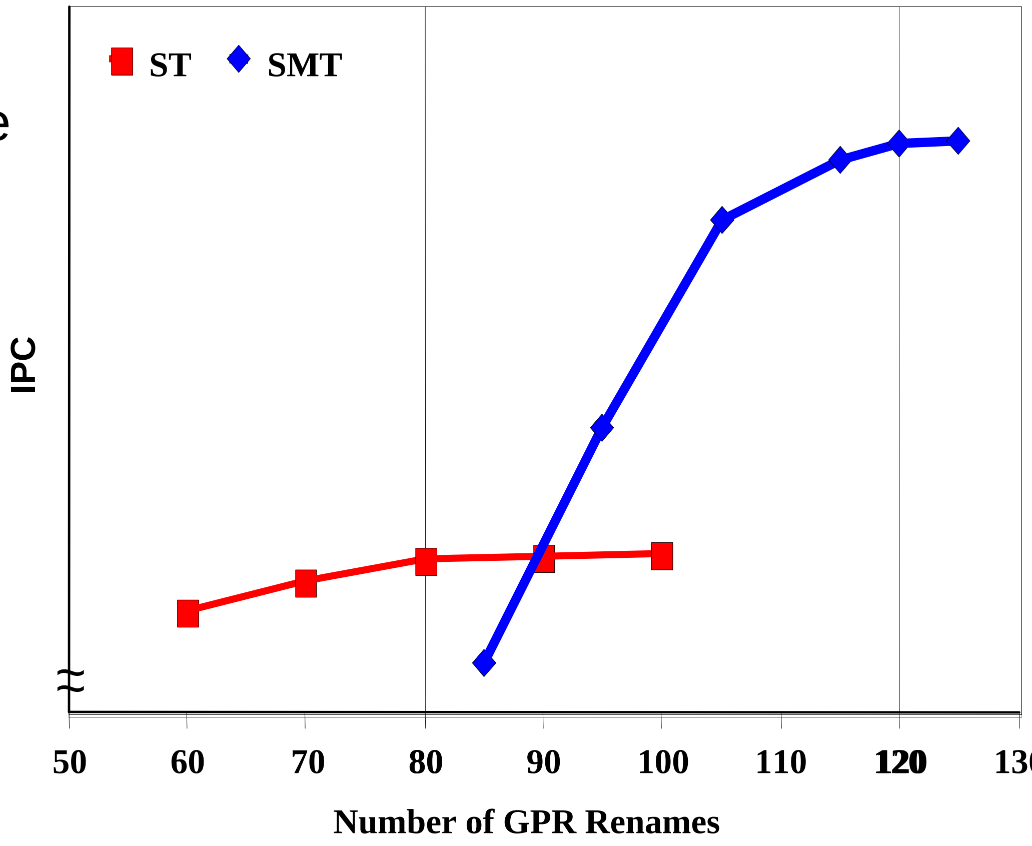
Changes Going From ST to SMT Core

- SMT easily added to Superscalar Micro-architecture
  - Second Program Counter (PC) added to share I-fetch bandwidth
  - GPR/FPR rename mapper expanded to map second set of registers (High order address bit indicates thread)
  - Completion logic replicated to track two threads
  - Thread bit added to most address/tag buses



Resource Sizes

- Analysis done to optimize every micro-architectural resource size
  - GPR/FPR rename pool size
  - I-fetch buffers
  - Reservation Station
  - SLB/TLB/ERAT
  - I-cache/D-cache
- Many Workloads examined
- Associativity also examined

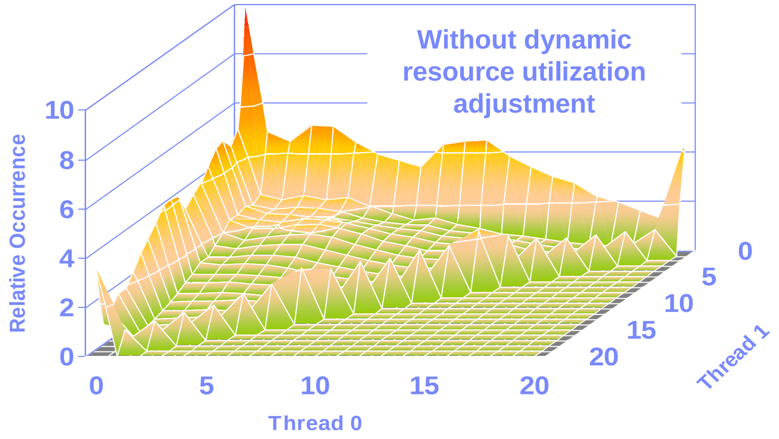


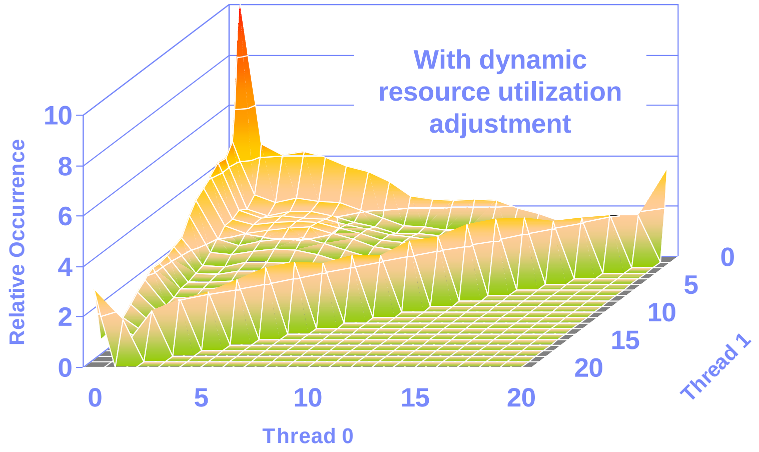
Results based on simulation of an online transaction processing application

Vertical axis does not originate at 0

Resource Sharing

Global Completion Table Occupancy

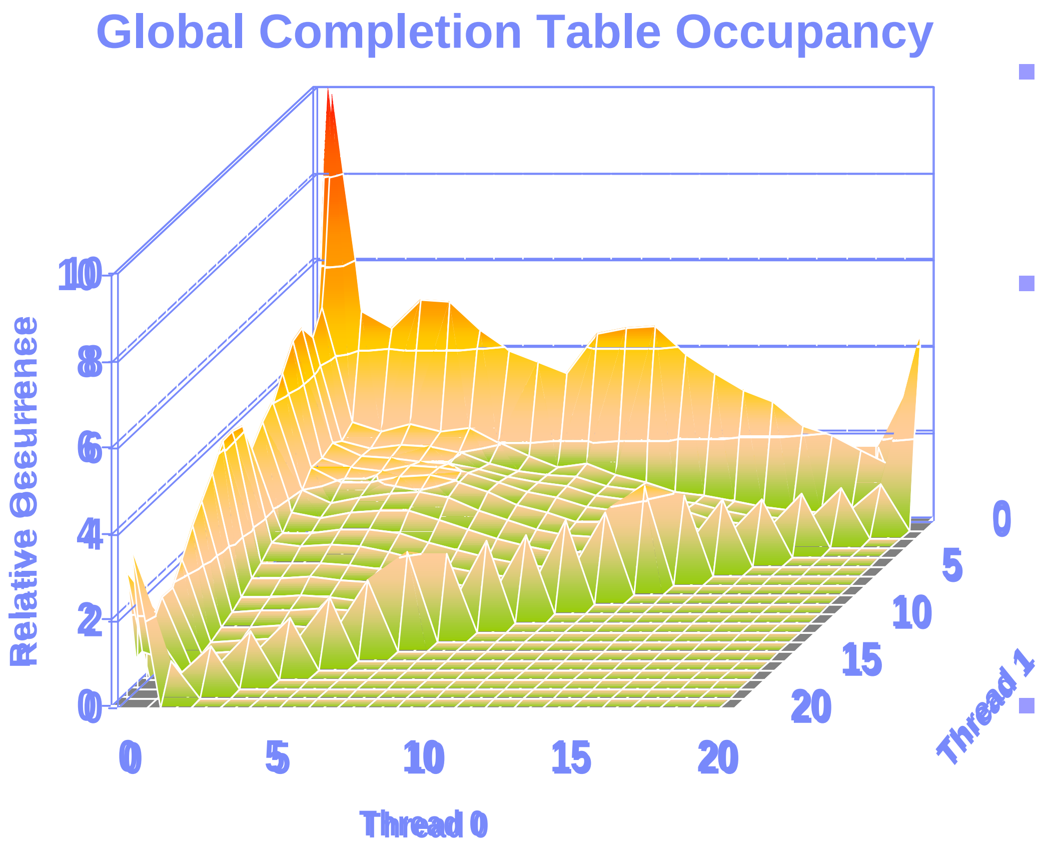




Results based on simulation of an online transaction processing application

- Threads share many resources
  - Global Completion Table, BHT, TLB, . . .
- Higher performance realized when resources balanced across threads
  - Tendency to drift toward extremes accompanied by reduced performance
- Solution: Dynamically adjust resource utilization

Resource Sharing

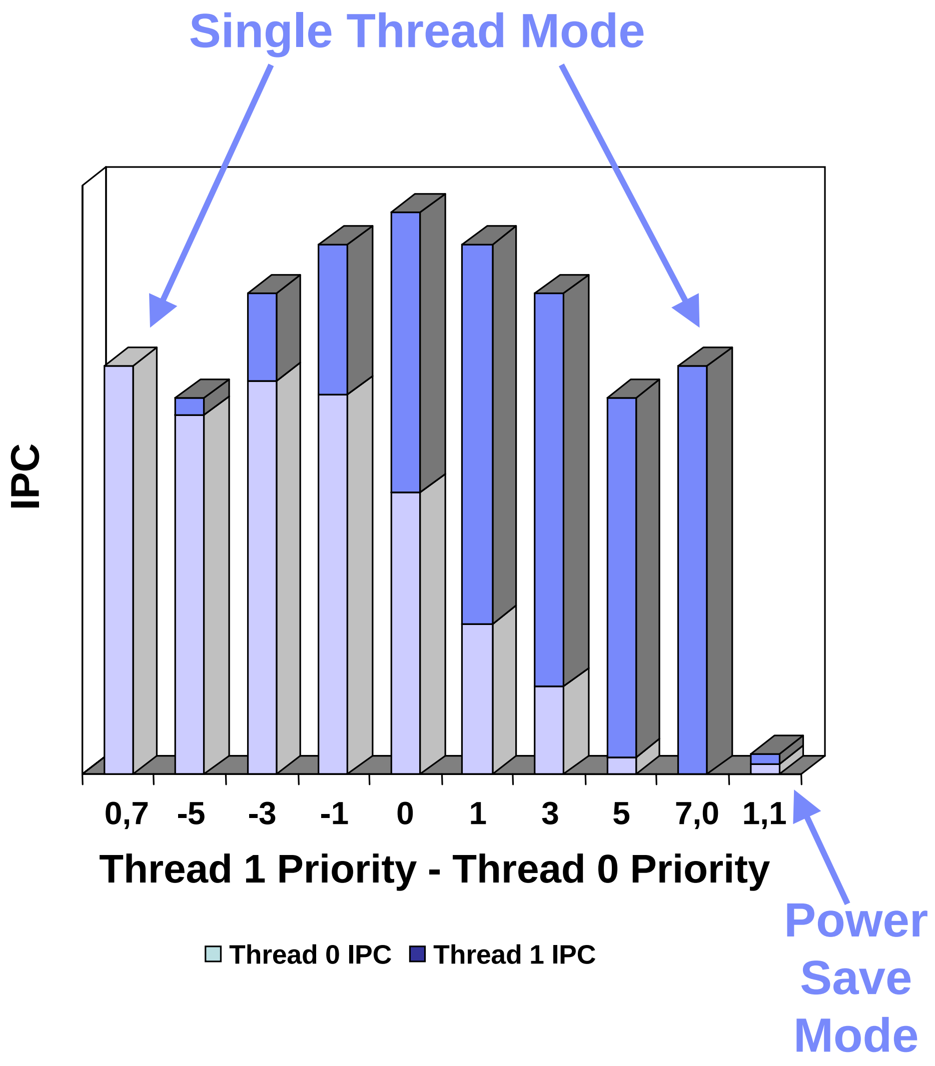


Results based on simulation of an online transaction processing application

- Threads share many resources
  - Global Completion Table, BHT, TLB, . . .
- Higher performance realized when resources balanced across threads
  - Tendency to drift toward extremes accompanied by reduced performance
- Solution: Dynamically adjust resource utilization

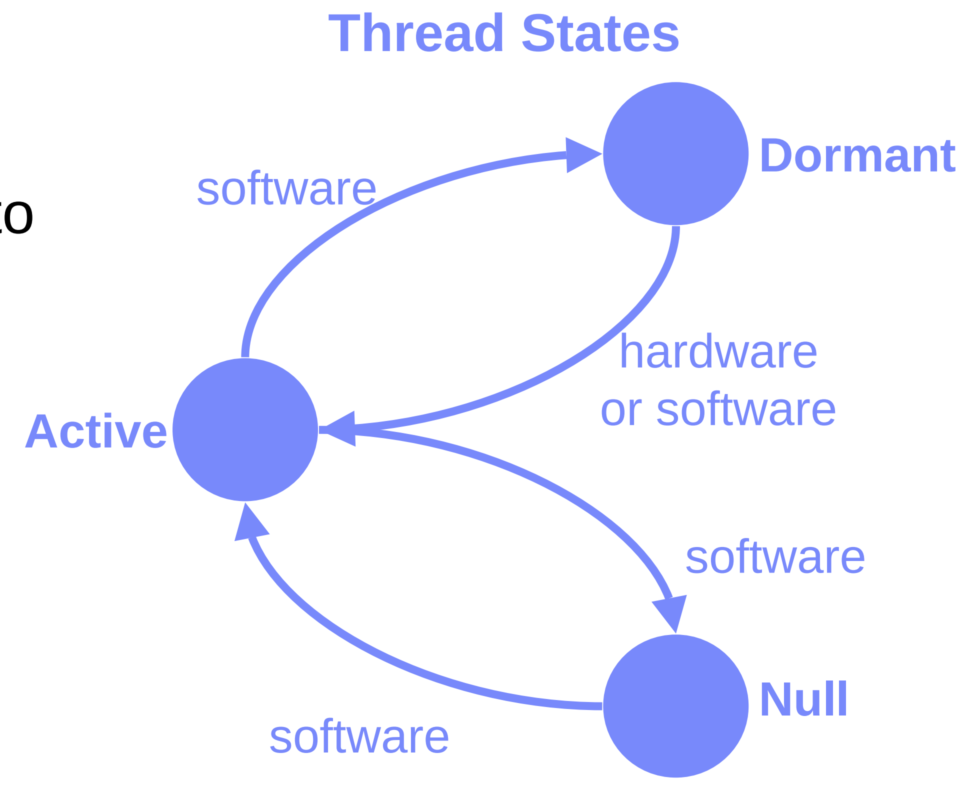
Thread Priority

- Instances when unbalanced execution desirable
  - No work for opposite thread
  - Thread waiting on lock
  - Software determined non uniform balance
  - Power management
  - …
- Solution: Control instruction decode rate
  - Software/hardware controls 8 priority levels for each thread



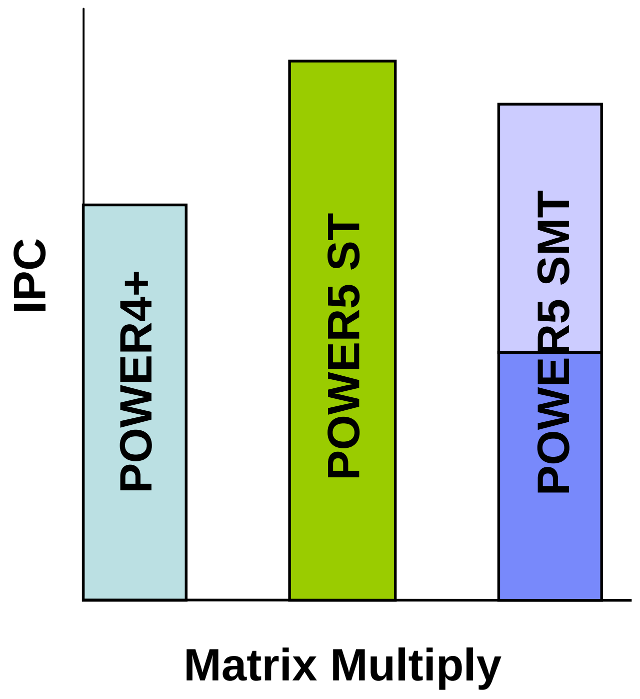
Dynamic Thread Switching

- Used if no task ready for second thread to run
- Allocates all machine resources to one thread
- Initiated by software
- Dormant thread wakes up on:
  - External interrupt
  - Decrementer interrupt
  - Special instruction from active thread



Single Thread Operation

- Advantageous for execution unit limited applications
  - Floating or fixed point intensive workloads
- Execution unit limited applications provide minimal performance leverage for SMT
  - Extra resources necessary for SMT provide higher performance benefit when dedicated to single thread
- Determined dynamically on a per processor basis



Other SMT Considerations

- Power Management
  - SMT Increases execution unit utilization
  - Dynamic power management does not impact performance
- Debug tools / Lab bring-up
  - Instruction tracing
  - Hang detection
  - Forward progress monitor
- Performance Monitoring
- Serviceability

Summary

- POWER5 SMT implementation is more than SMT
  - Good ROI for silicon area: Performance gain > Area increase
  - Resource sizes optimized
  - Dynamic feedback enhances instruction throughput
  - Software controlled priority exploits machine architecture
  - Dynamic ST to/from SMT mode capability optimizes system resources
- SMT impacts pervasive throughout chip
- Operating in laboratory
  - AIX, Linux and OS/400 booted and running