How Many Threads?

• To leverage CMPs:
  – Applications must be divided into *threads*

• Some applications:
  – As many threads as the number of cores

• Other applications:
  – Performance saturates
  – Fewer threads than cores

The number of threads must be chosen carefully
Two Important Limitations

- Contention for shared data
  - Data synchronization: Critical section

- Contention for shared resources
  - Off-chip bus
Contestion for Critical Section

Critical Section:
Add local histogram to global histogram

GetPageHistogram(Page *P) → UpdateLocalHistogram(Fraction of Page) → Critical Section: Add local histogram to global histogram

Parallel Part

Serial Part

Barrier

Kernel from PageMine
Contention for Critical Section
Two Important Limitations

• Contention for shared data
  – Data-synchronization: Critical section

• Contention for shared resources
  – Off-chip bus
Contestion for Off-chip Bus

Kernel from ED

EuclideanDistance (Point A)

for i = 1 to num_dimensions
    sum = sum + A[i] * A[i]
Contestion for Off-chip Bus

N = 1
N = 2
N = 4
N = 8

N = 4 and N = 8 take the same time to execute.

LEGEND
- Parallel Part
- Bus Access
--- Waiting for Bus
Who Chooses Number of Threads?

- Programmer
  - No! Not for general-purpose workloads
    - Large variation in input sets and machines

- User
  - No! I do not want my media player to ask me
  - Set equal to the number of cores
    - Assumption:
      More threads → more performance

Goal: A run-time mechanism to estimate the best number of threads
Outline

• Motivation

• Feedback-Driven Threading
  – Synchronization-Aware Threading (SAT)
  – Bandwidth-Aware Threading (BAT)
  – Combining SAT and BAT

• Related Work and Summary
Feedback-Driven Threading (FDT)

Conventional Multi-Threading

\[ N = K \]

Feedback-Driven Threading

\[ N = \text{No. of threads} \]
\[ K = \text{No. of cores} \]

Train to sample application behavior
Outline

• Motivation

• Feedback-Driven Threading
  – Synchronization-Aware Threading (SAT)
  – Bandwidth-Aware Threading (BAT)
  – Combining SAT and BAT

• Related Work and Summary
Synchronization-Aware Threading (SAT)

\[ T_N = \frac{\text{Time outside C.S.}}{N} + N \times \text{Time inside C.S.} \]

\[ N_{CS} = \sqrt{\frac{\text{Time outside C.S.}}{\text{Time inside C.S.}}} \]
Implementing SAT using FDT

• Train
  – Measure the time inside and outside the critical section using cycle counter

• Compute $N_{CS} = \sqrt{\frac{\text{Time outside C.S.}}{\text{Time inside C.S.}}}$

• Execute
Machine Configuration

- CMP: 32 in-order cores (2-wide, 5-stage deep)
- Caches: L1: 8-KB, L2: 64KB. Shared L3: 8MB
- Off-chip bus: 64-bit wide, 4x slower than cores
- Memory: 200 cycle minimum latency
Results of SAT

SAT decreases execution time and saves power
Adaptation of SAT to Input Sets

- Time inside and outside the critical section depends on the input set
- For PageMine, the best number of threads changes with the page size
Outline

- Motivation

- Feedback-Driven Threading
  - Synchronization-Aware Threading (SAT)
  - Bandwidth-Aware Threading (BAT)
  - Combining SAT and BAT

- Related Work and Summary
Bandwidth-Aware Threading (BAT)

N = 1
N = 2
N = 4
N = 8

N=4 and N=8 take the same time to execute

LEGEND
- Parallel Part
- Bus Access
- ----- Waiting for Bus

\[ N_{BW} = \frac{\text{Total Bandwidth}}{\text{Bandwidth used by a single thread}} \]
Implementation BAT using FDT

- **Train**
  - Measure bandwidth utilization using performance counters

- **Compute** \( N_{BW} = \frac{\text{Total Bandwidth}}{\text{Bandwidth used by a single thread}} \)

- **Execute**
Results of BAT

BAT saves power without increasing execution time
Adaptation of BAT to System Configuration

- The best number of threads is a function of off-chip bandwidth
- BAT correctly predicts the best number of threads for systems with different bandwidths
Outline

• Motivation

• Feedback-Driven Threading
  – Synchronization-Aware Threading (SAT)
  – Bandwidth-Aware Threading (BAT)
  – Combining SAT and BAT

• Related Work and Summary
Combining SAT and BAT

• Train
  – Train for both SAT and BAT

• Compute
  \[ N_{\text{SAT+BAT}} = \min (N_{\text{CS}}, N_{\text{BW}}, \text{Num. cores}) \]

• Execute
Results of SAT+BAT

Fewer threads $\rightarrow$ fewer cache misses
(SAT+BAT) reduces power and execution time

On average, (SAT+BAT) reduces the execution time by 17% and power by 59%
Comparison with Static-Best

Simulate all possible number of threads and choose the best

Two kernels: First needs 12 threads, second needs 32. Static-Best uses 32 for both.
Outline

• Motivation

• Feedback-Driven Threading
  – Synchronization-Aware Threading (SAT)
  – Bandwidth-Aware Threading (BAT)
  – Combining SAT and BAT

• Related Work and Summary
Related Work

• Performance vs. number of threads on real machines
  – Neoplosha+ [CF’07], Saini+ [Comp. methods’06]

• Multiple Multi-threaded workloads on SMPs
  – McCann+ [Trans. CS’93], Corbalan+ [Trans. PDS’05]

• Techniques to control number of threads
  – Compile-time: Kumar+ [IPDPS’02]
  – Run-time: Li+ [HPCA’06]

• Resource Allocation in SMPs
  – Nguyen+[IPPS’96], Corbalan+ [Trans. PDS’05]
Summary

• Feedback-Driven Threading (FDT)
  – Estimate best number of threads at run-time
  – Enables power-efficient and high-performance execution
  – Adapts to input sets and machine configurations
  – Does not require programmer/user intervention

• Future Work
  – Other limitations: fine-grain locking, data sharing
• Thank You