Feedback-Driven Threading

M. Aater Suleman†

Moinuddin K. Qureshi‡

Yale N. Patt†

†HPS Research Group

The University of Texas at Austin

‡ IBM T.J. Watson Research Center

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

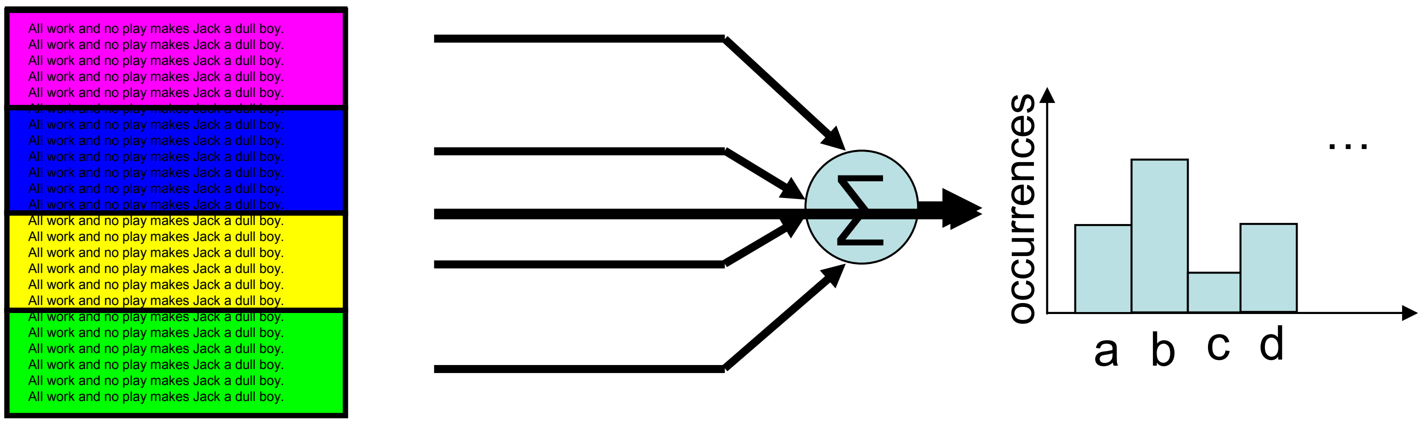
Two Important Limitations

- Contention for shared data
  - Data synchronization: Critical section

- Contention for shared resources
  - Off-chip bus

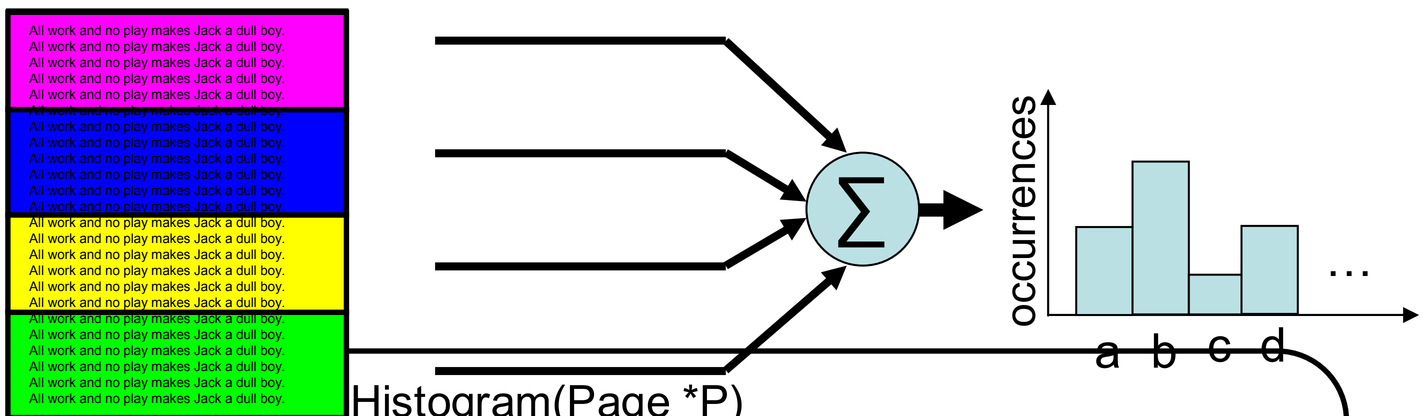
Contention for Critical Section

Kernel
from
PageMine



Contention for Critical Section

Kernel from PageMine



Histogram(Page *P)

Parallel Part

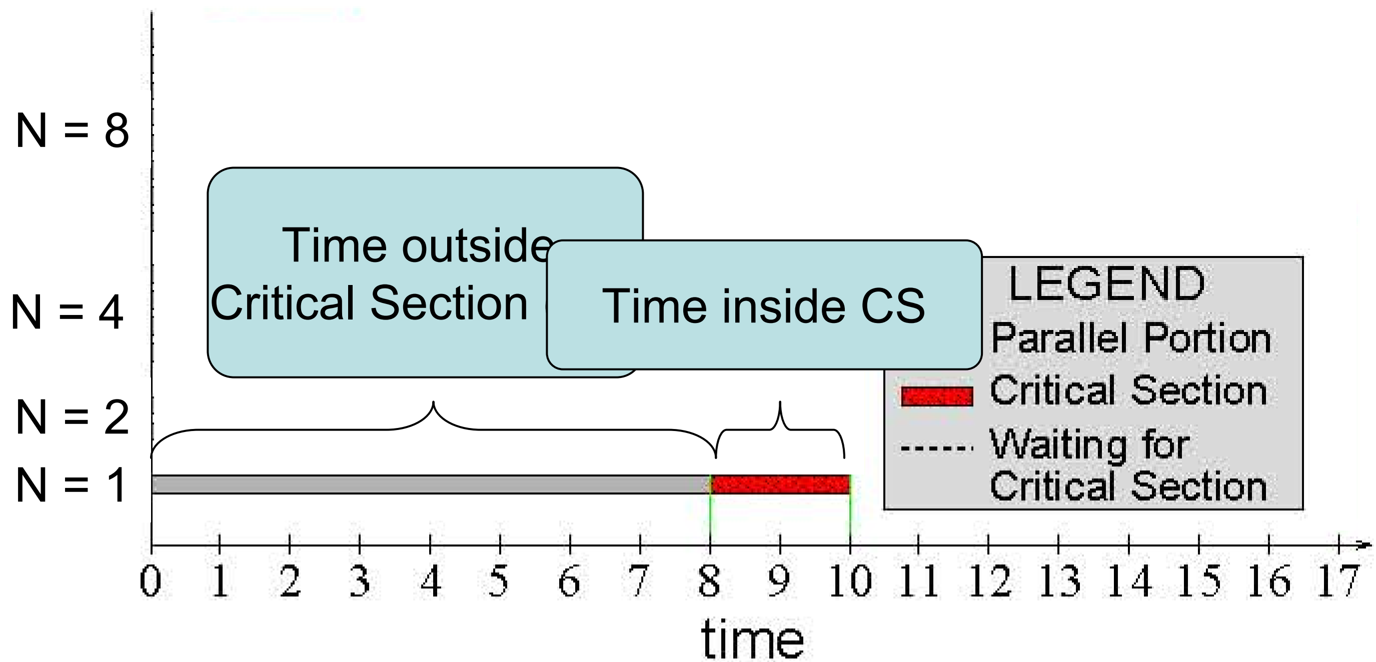
UpdateLocalHistogram(Fraction of Page)

Serial Part

Critical Section:
Add local histogram to global histogram

Barrier

Contention for Critical Section



Two Important Limitations

- Contention for shared data
  - Data-synchronization: Critical section
- Contention for shared resources
  - Off-chip bus

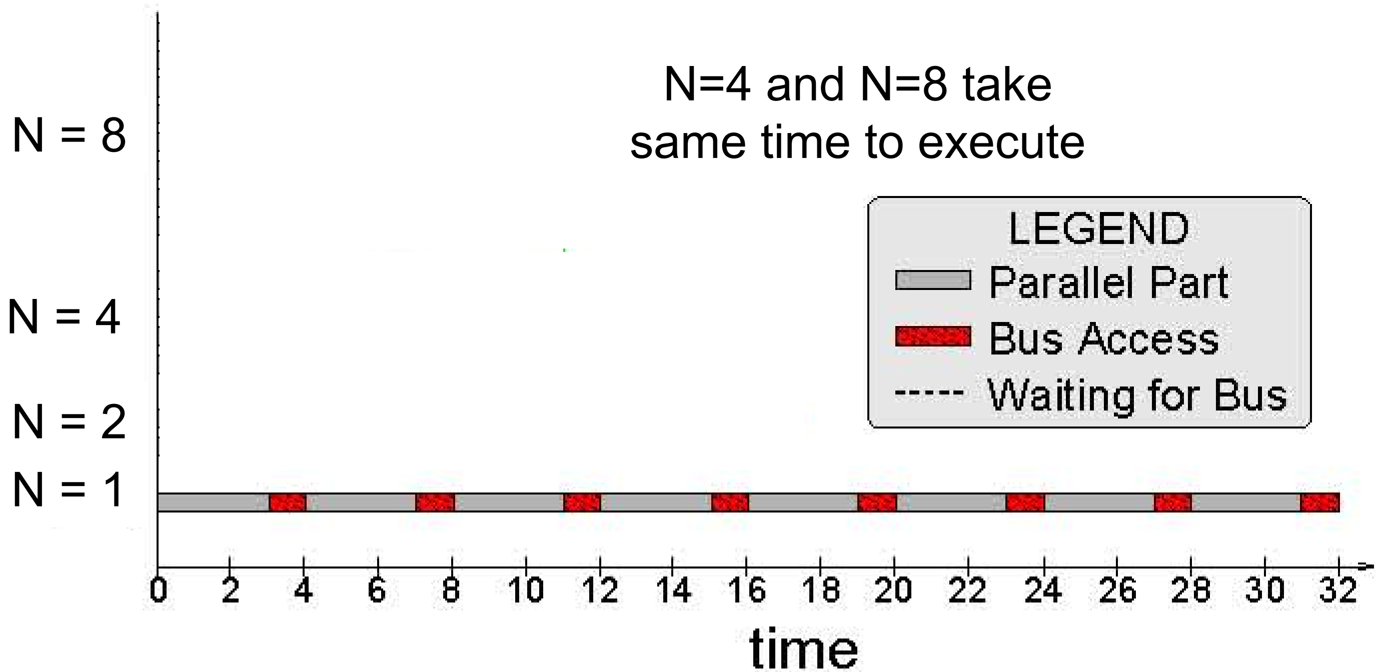
Contention for Off-chip Bus

Kernel
from ED

EuclideanDistance (Point A)

```
for i = 1 to num_dimensions
        sum = sum + A[i] * A[i]
```

Contention for Off-chip Bus



Who Chooses Number of Threads?

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

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

  - – me

- Set equal to the number of cores
  - Assumption:

    More threads $\nRightarrow$ more performance

Outline

- Motivation
- Feedback-Driven Threading
  - Synchronization-Aware Threading (SAT)
  - Bandwidth-Aware Threading (BAT)
  - Combining SAT and BAT
- Related Work and Summary

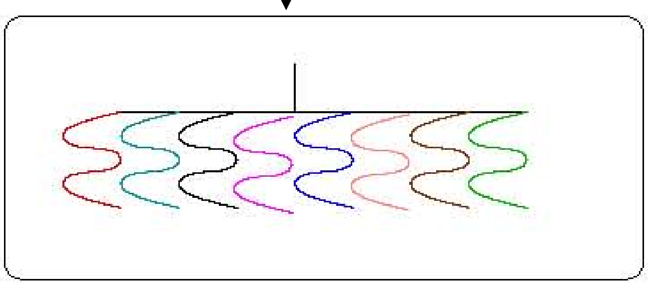
Feedback-Driven Threading (FDT)

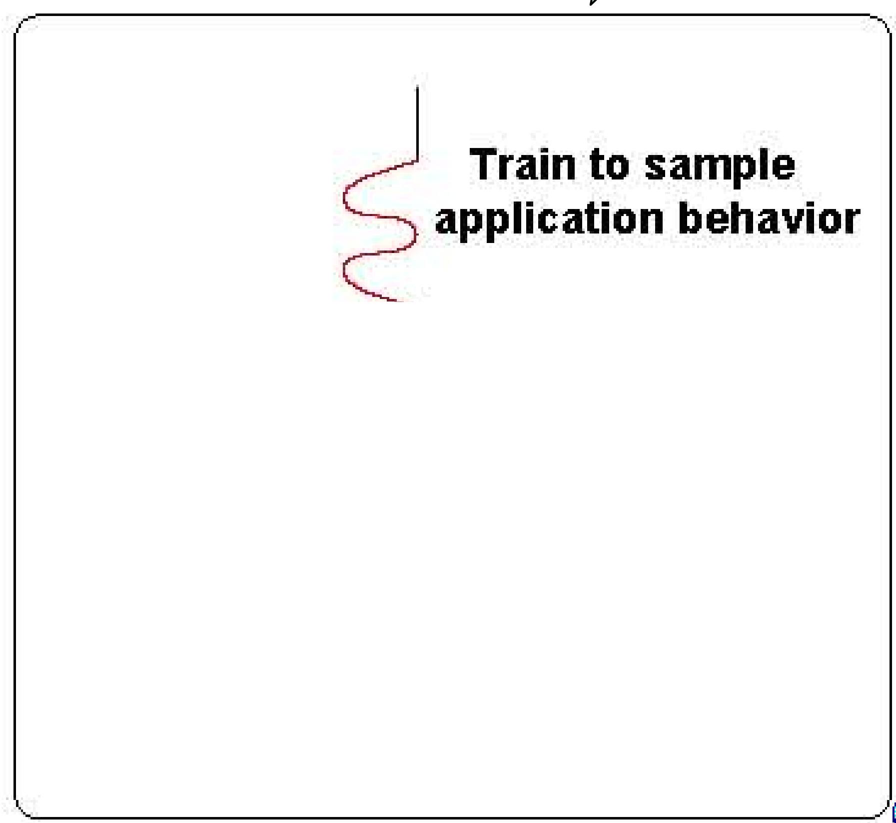
Conventional Multi-Threading

N = No. of threads
K = No. of cores

Feedback-Driven Threading

N = K

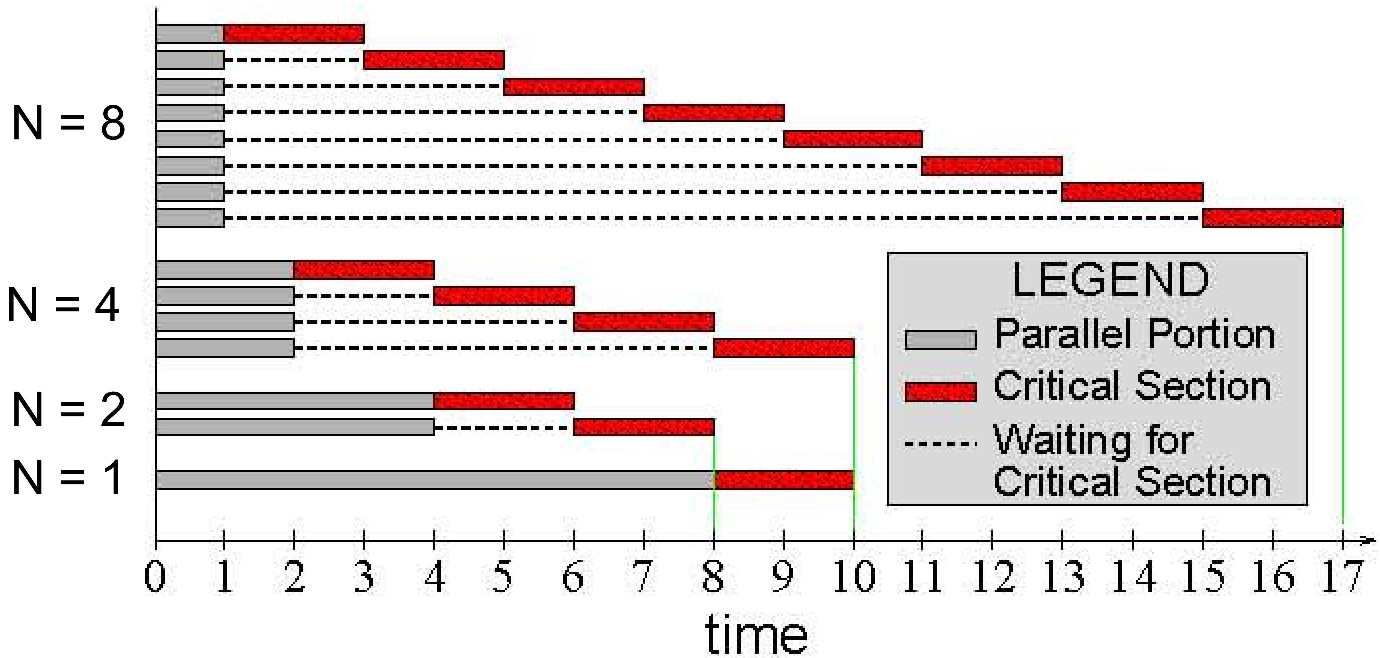




Outline

- Motivation
- Feedback-Driven Threading
  - Synchronization-Aware Threading (SAT)
  - Bandwidth-Aware Threading (BAT)
  - Combining SAT and BAT
- Related Work and Summary

Synchronization-Aware Threading (SAT)



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

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

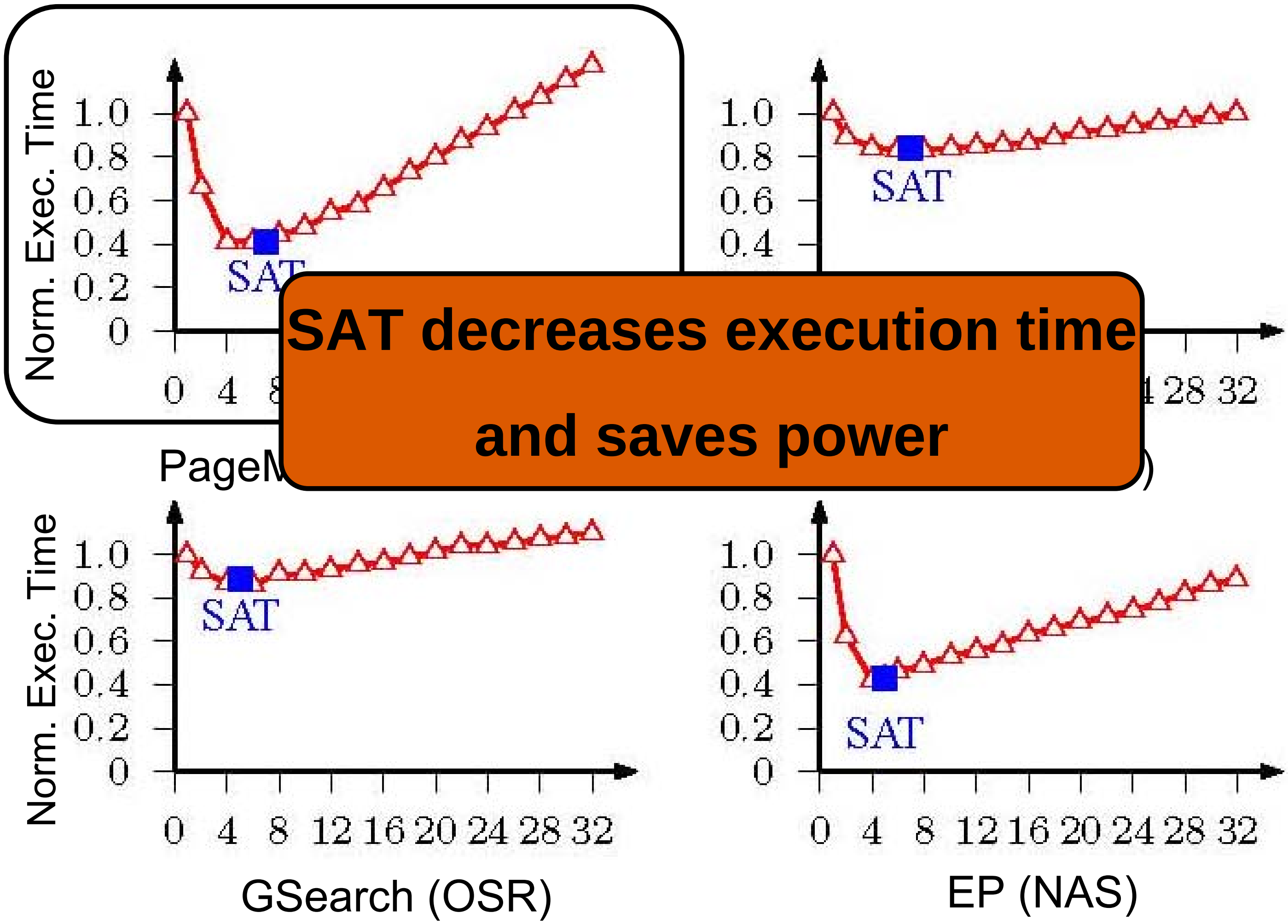
Implementing SAT using FDT

- Train
  - Measure the time inside and outside the critical section using cycle counter
- Compute $N_{CS} = \sqrt{\dfrac{\text{Time outside C.S.}}{\text{Time inside C.S}}}$
- 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

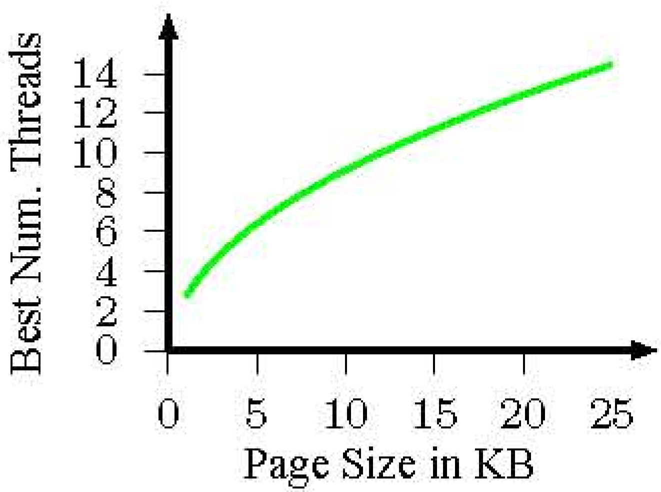
Results of SAT

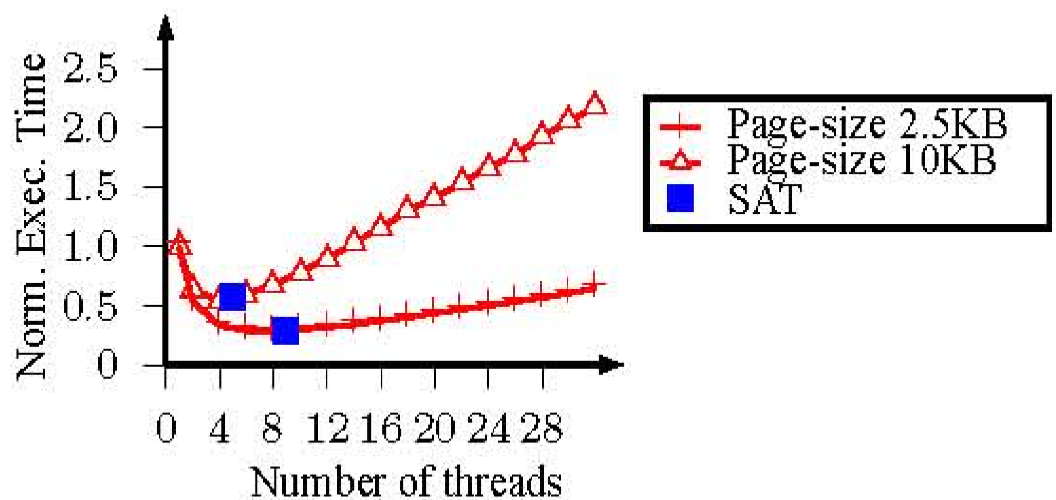


SAT decreases execution time and saves power

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

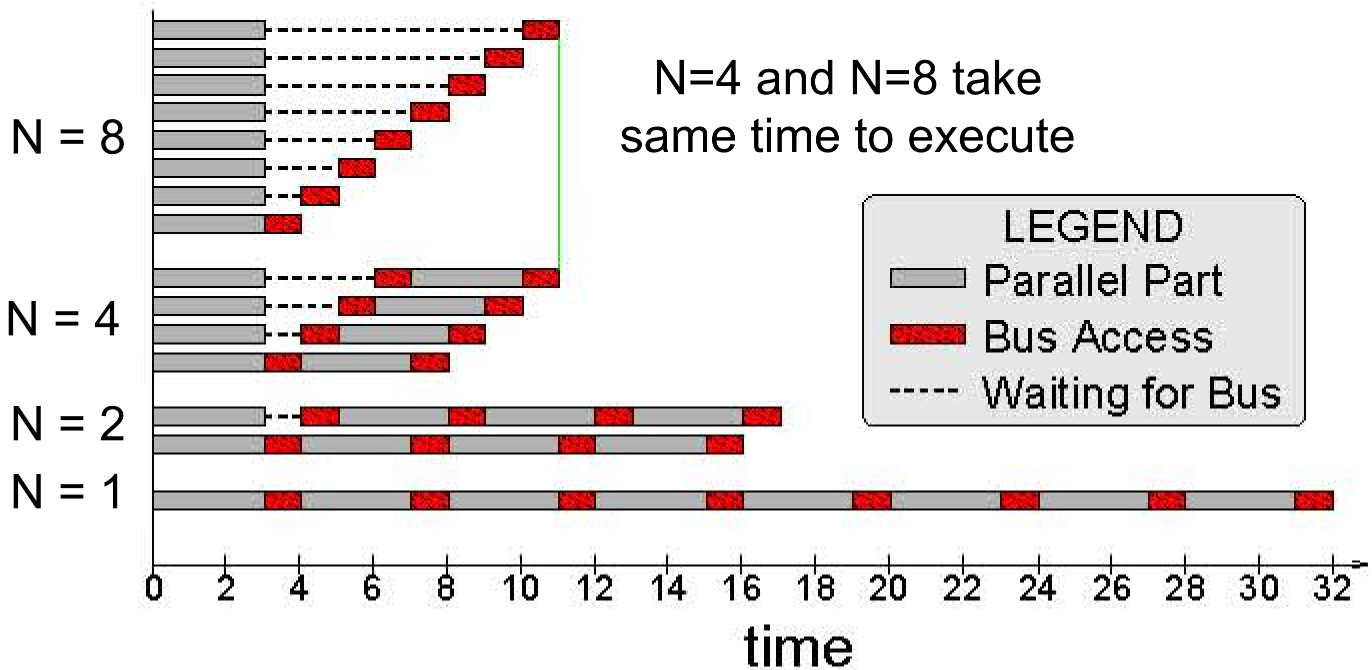




Outline

- Motivation
- Feedback-Driven Threading
  - Synchronization-Aware Threading (SAT)
  - Bandwidth-Aware Threading (BAT)
  - Combining SAT and BAT
- Related Work and Summary

Bandwidth-Aware Threading (BAT)

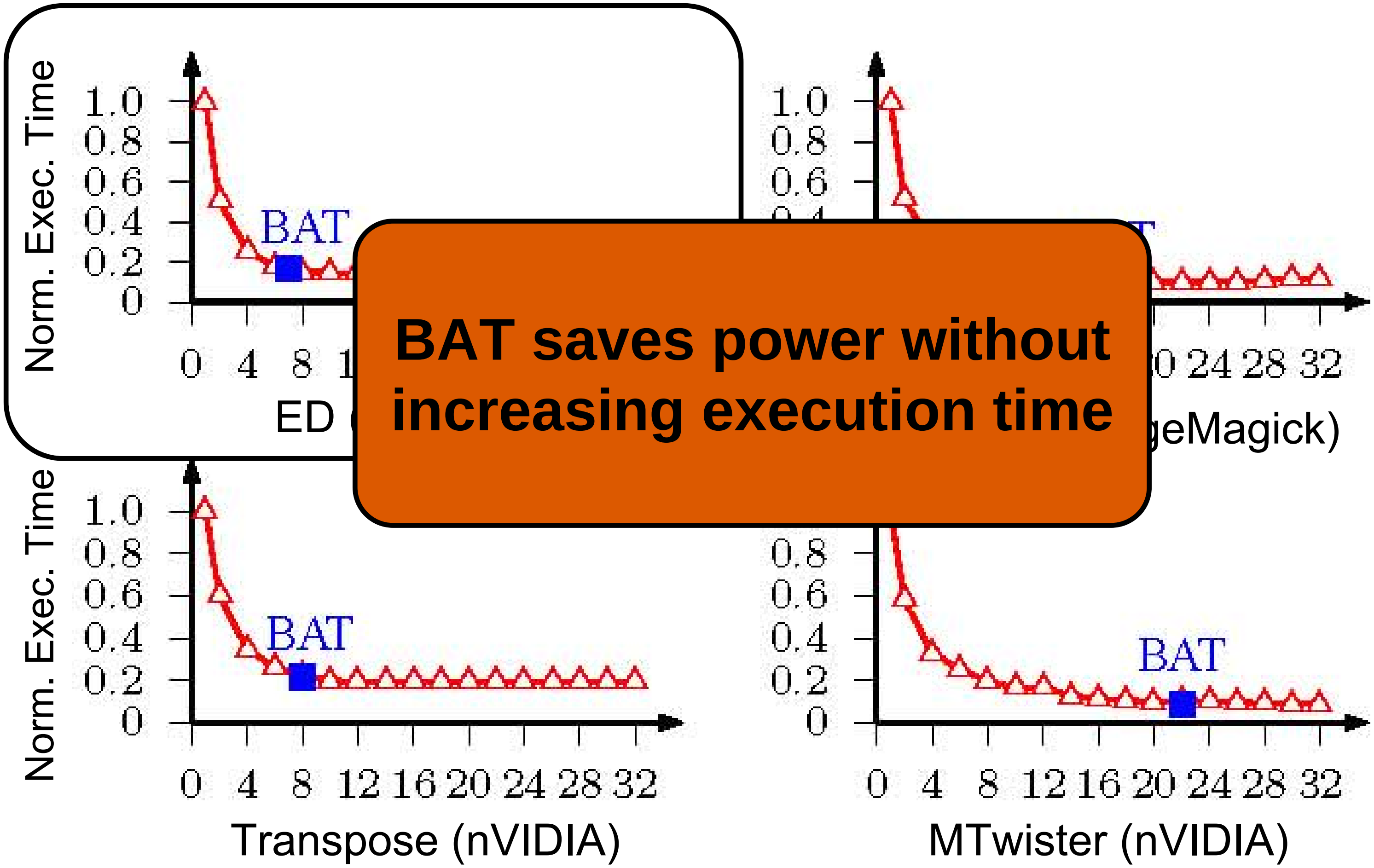


$$N_{BW} = \frac{\text{Total Bandwidth}}{\text{Bandwidth used by a single thread}}$$

Implementation BAT using FDT

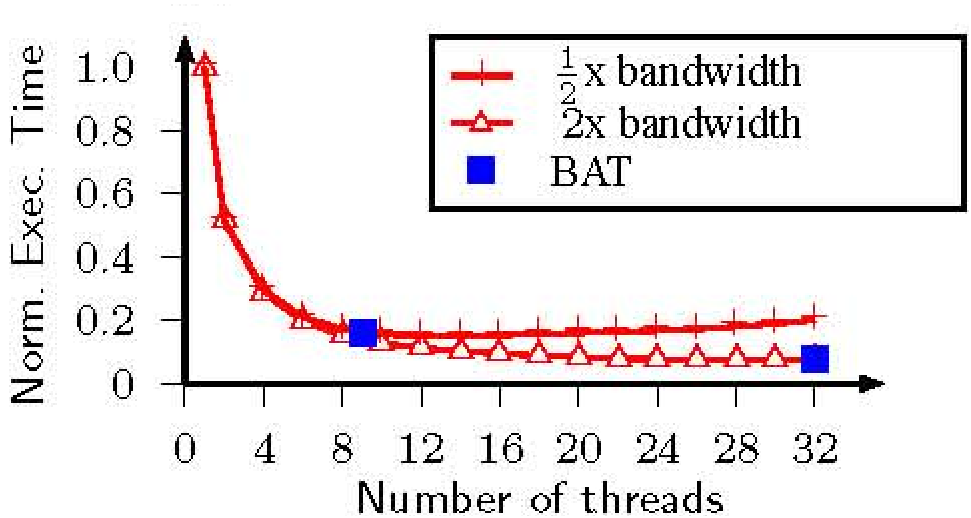
- Train
  - Measure bandwidth utilization using performance counters
- Compute $N_{BW} = \frac{\text{Total Bandwidth}}{\text{Bandwidth used by a single thread}}$
- Execute

Results of BAT



BAT saves power without increasing execution time

Adaptation of BAT to System Configuration



convert (ImageMagick)

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

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

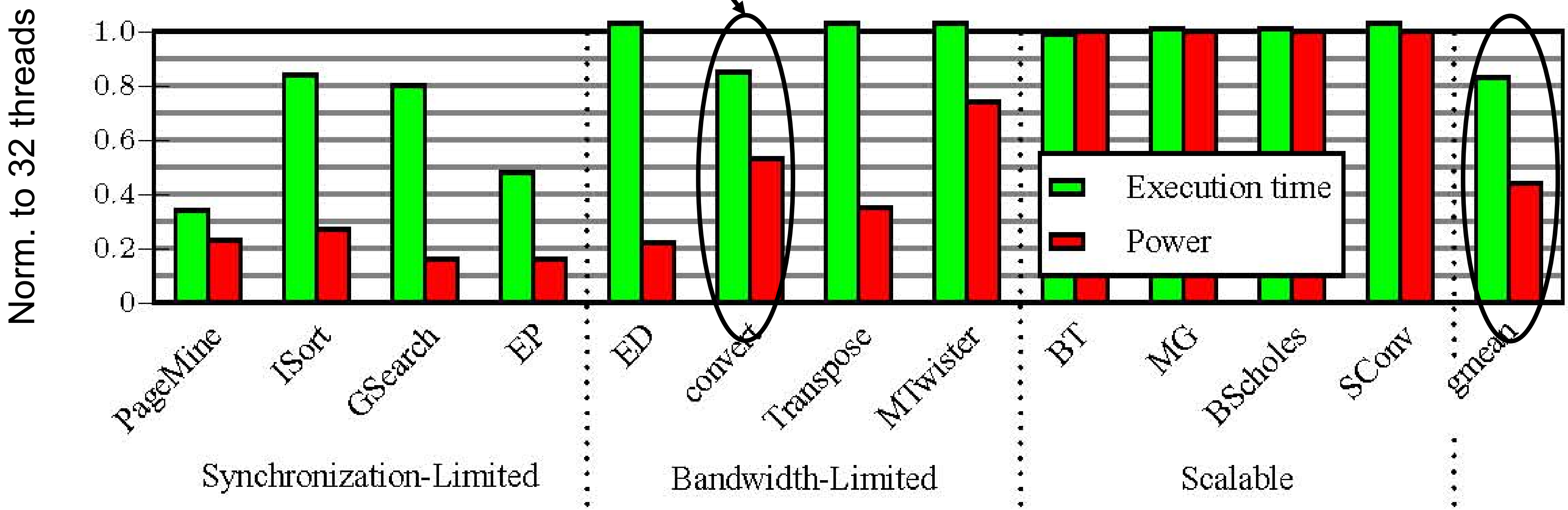
  $N_{SAT+BAT} = MIN\ (N_{CS}, N_{BW}, Num.\ cores)$

- Execute

Results of SAT+BAT

Fewer threads → fewer cache misses

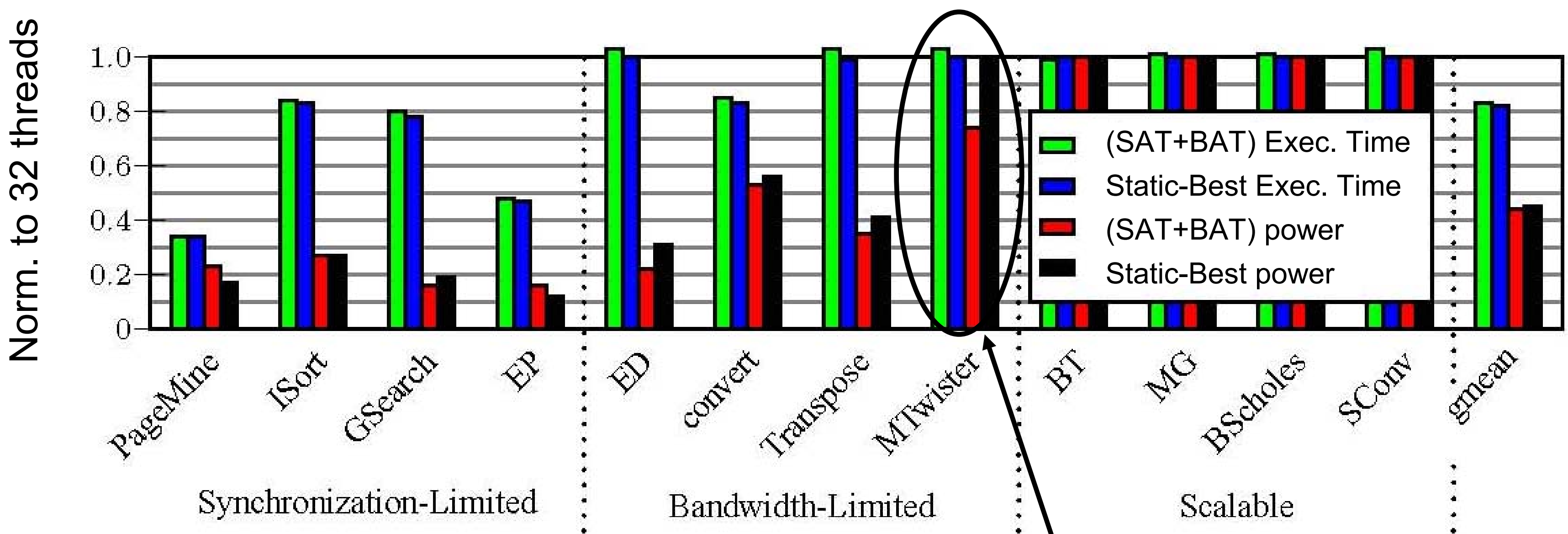
(SAT+BAT) reduces power and execution time



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