




Thread Tailor

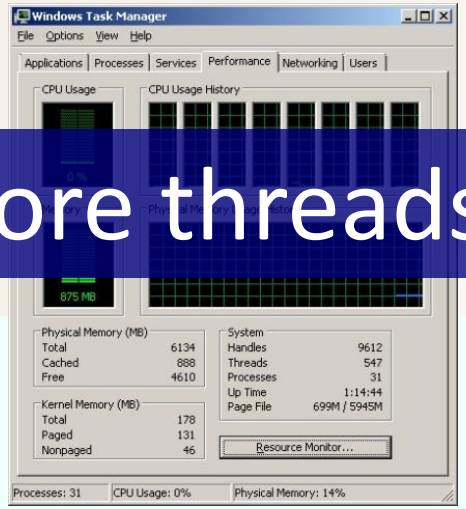
Dynamically Weaving Threads Together for Efficient, Adaptive Parallel Applications

Janghaeng Lee, Haicheng Wu,
Madhumitha Ravichandran, Nathan Clark

Georgia Tech College of Computing

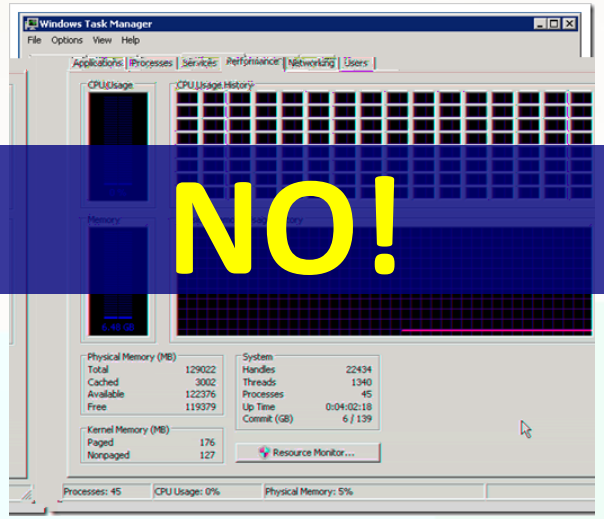
Motivation

- Hardware Trends
  - Put more cores in a single chip



2009

More threads always win? **NO!**



201X

- CPU intensive programs
  - Exploits Thread Level Parallelism

Georgia Tech College of Computing



Optimal Number of Threads

- Too many threads
  - More synchronization
  - More contention for system resources
- Too few threads
  - Resource underutilization
- Who can decide the number?
  - **Not a programmer**



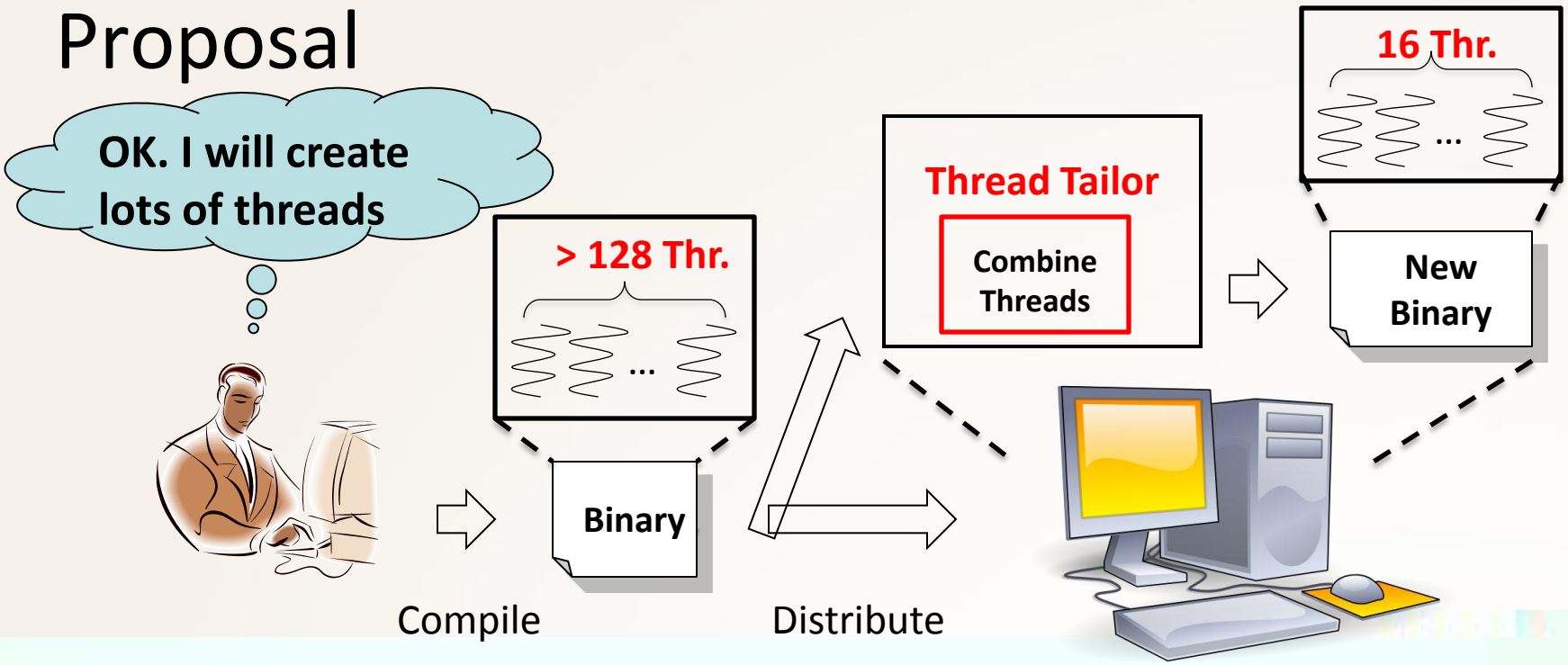
Why NOT?

- Input changes
  - Various working-set size
- The system changes
  - Various available resources
- Hardware changes
  - Various L2/L3 cache structure / size, etc.

Decision must be made at runtime

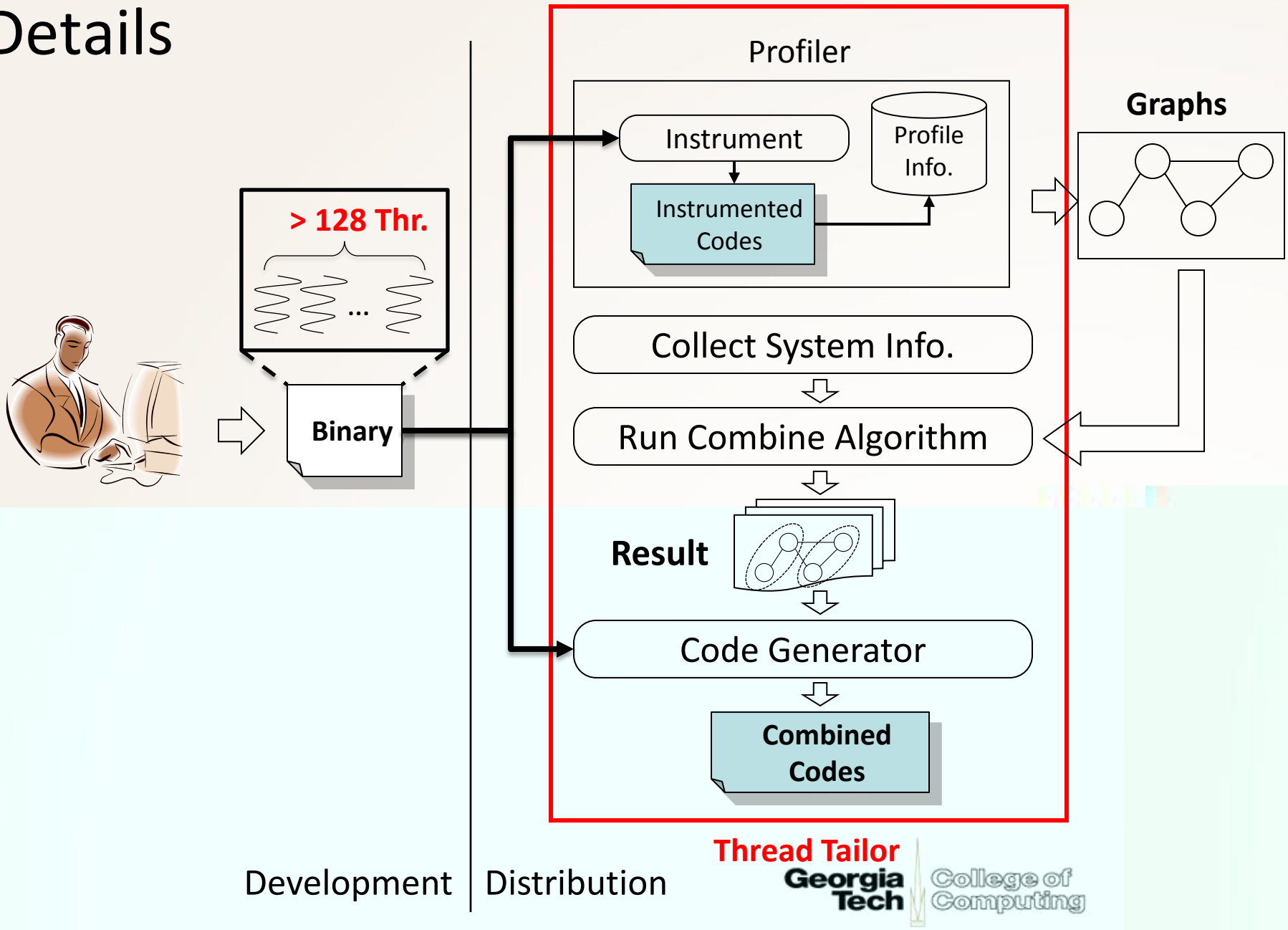
Georgia Tech College of Computing

Proposal

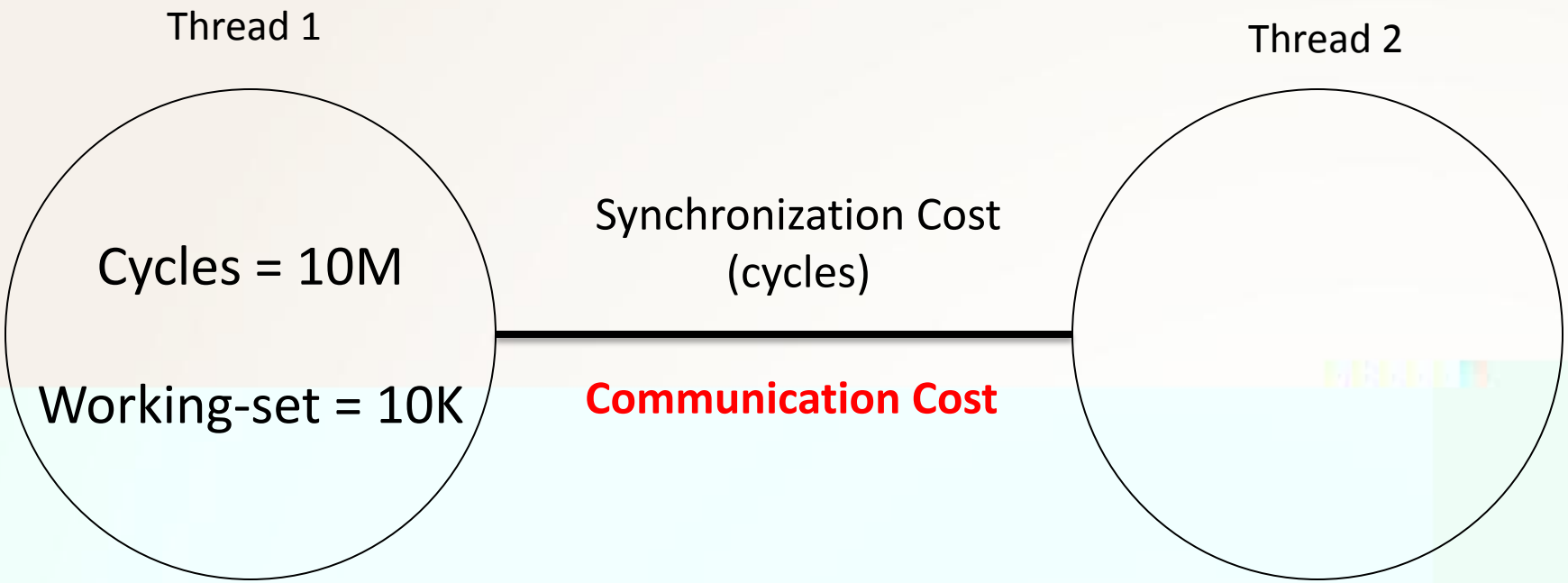


- Combining Threads
  - Group Several Threads into a Single Thread
    - Threads in the same group are executed in serial
    - Executed on the SAME core

Details


Profiler
Instrument
Profile Info.
Instrumented Codes
Graphs
> 128 Thr.
...
Collect System Info.
Binary
Run Combine Algorithm
Result
Code Generator
Combined Codes
Development
Distribution
Thread Tailor
Georgia Tech
College of Computing

Graph Construction

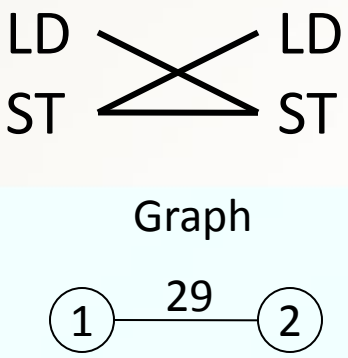


Communication Cost

- Intuition : STORE Instruction causes coherence miss in cache
- Log Memory Access per Thread

Thread 1

Address	LD Count	ST Count
…	…	…
0x00001234	5	10
0x00001338	4	9
…	…	…
0x00004000	7	7
…	…	…



Thread 2

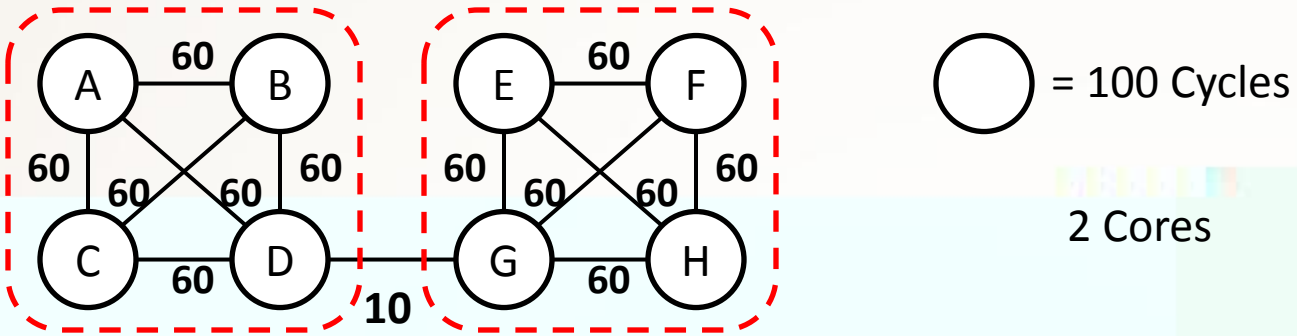
Address	LD Count	ST Count
…	…	…
0x00001234	0	7
0x00002000	4	4
…	…	…
0x00004000	3	8
…	…	…

```
0x00001234: MIN(5, 7)  + MIN(10, 0) + MIN(10, 7) = 12

0x00004000: MIN(7, 8) + MIN( 7, 3) + MIN( 7, 8) = 17

Total Communication Cost: 12 + 17 = 29
```

Georgia Tech College of Computing

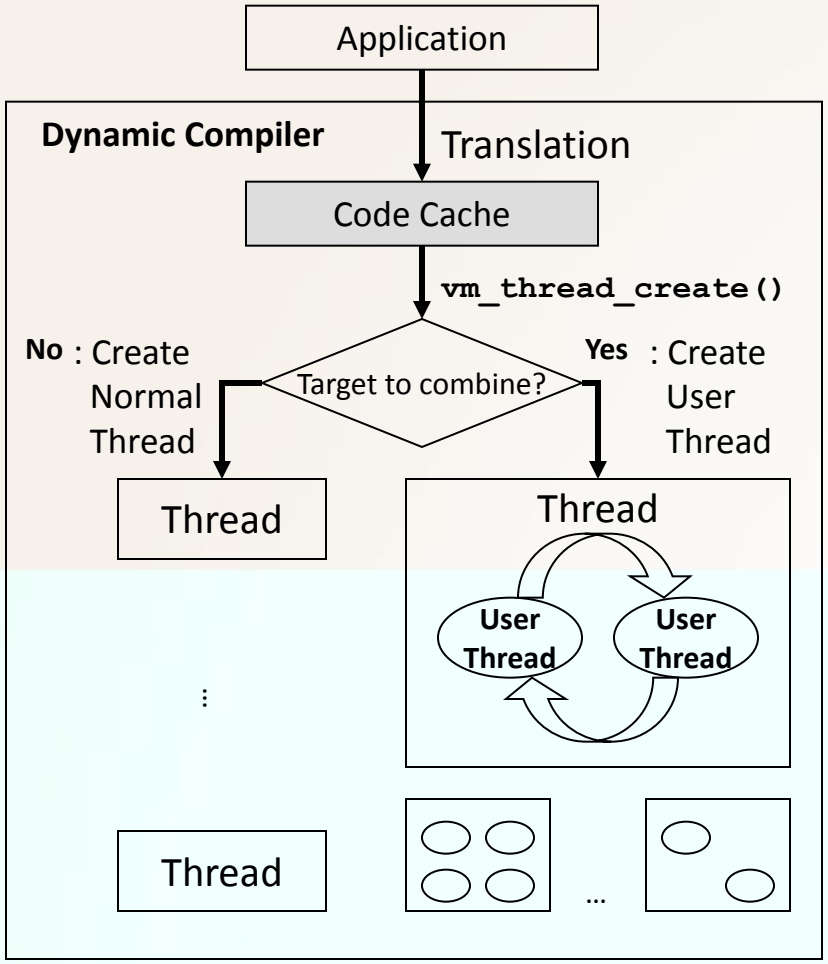
Combining Algorithm

- Kernighan-Lin(KL) Graph Partitioning Heuristic
  - Goal : Minimize Execution Cycles
  - Precondition : Combined Threads ≤ Cores



Partition 1	Partition 2	Partition 1 Cycle Estimation	Partition2 Cycle Estimation	Move From	Move Node
B C D G	A E F H	210	220	2	A
A B C D G	E F H	130	120	1	G
A B C D	E F G H	40	40	1	D

Thread Combining



Replace Thread APIs with Wrapper Functions

Wrapper Function for Thread Creation

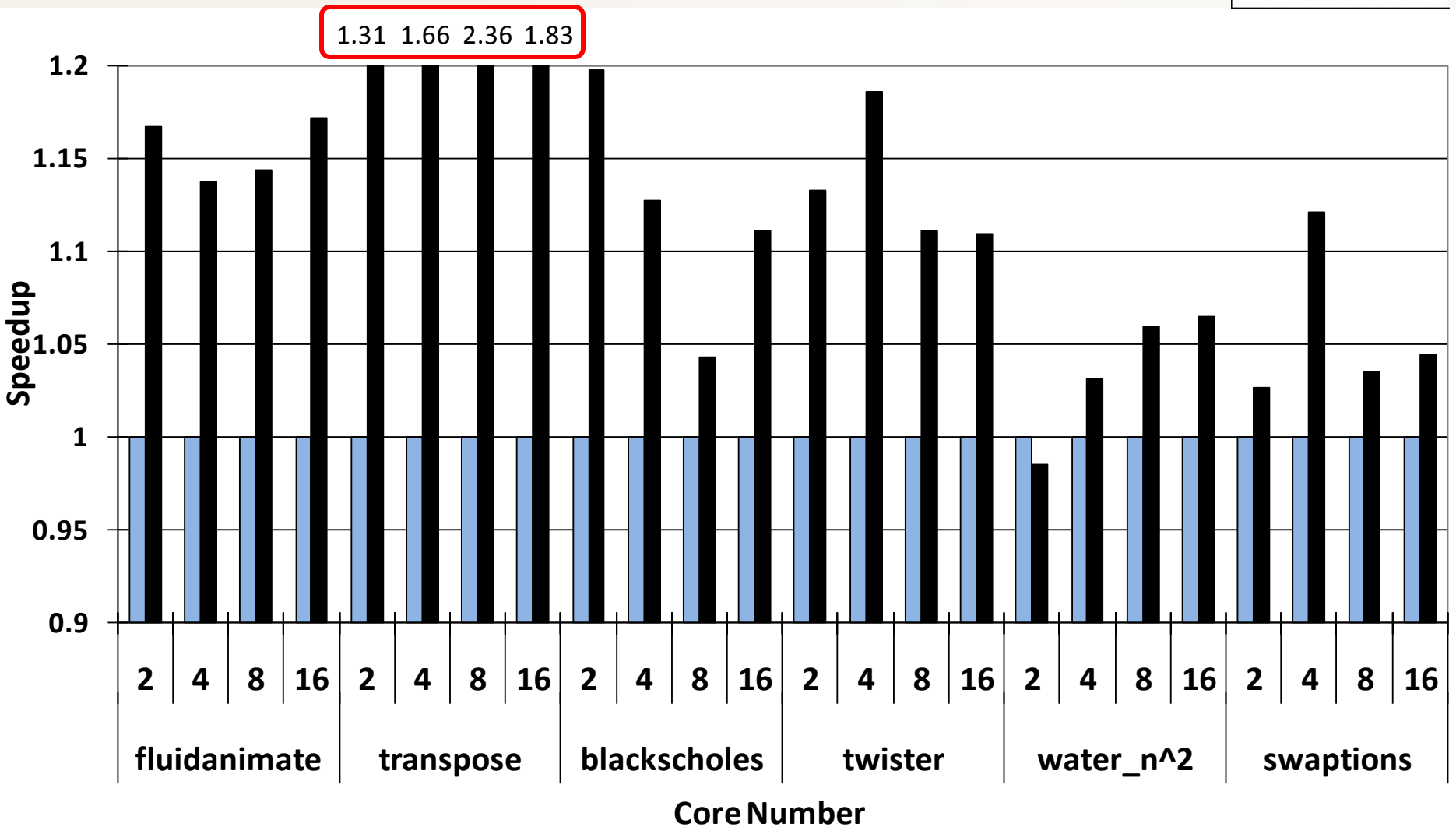
Context Switched by Dynamic Compiler

Serially Execute User Threads in Real Thread

Experimental Setup

- 2 cores
  - Intel Core 2 Duo 6600 (2.4 Ghz)
- 4 cores
  - Intel Core 2 Quad Q6600 (2.4.Ghz)
- 8 cores
  - 2 Quad-core CPUs with SMT
  - Intel Xeon E5520 (2.26 Ghz)
- 16 cores (Logical)
  - 2 Quad-core CPUs with SMT and HyperThreading
  - Intel Xeon E5520 (2.26 Ghz)

Results

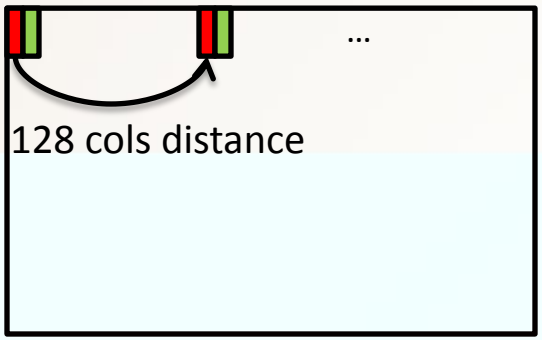

#thr=#core
Thread Tailor
1.31 1.66 2.36 1.83
Speedup
1.2
1.15
1.1
1.05
1
0.95
0.9
2 4 8 16 2 4 8 16 2 4 8 16 2 4 8 16 2 4 8 16 2 4 8 16
fluidanimate
transpose
blackscholes
twister
water_n^2
swaptions
Core Number

Result Analysis - Transpose

- Transpose ***m * n*** matrix to ***n * m***

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix} \Rightarrow \begin{pmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{pmatrix}$$

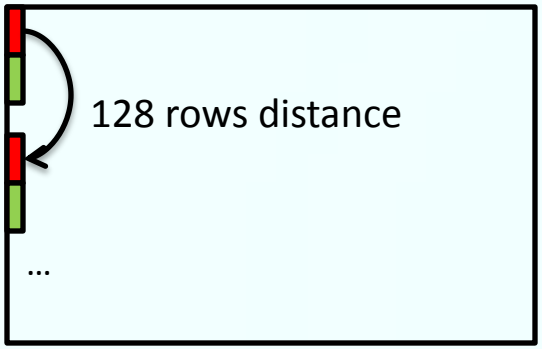
- Parallel Transpose



Thread 1

Thread 2

Input Matrix



Output Matrix

Georgia Tech College of Computing

Result Analysis - Transpose

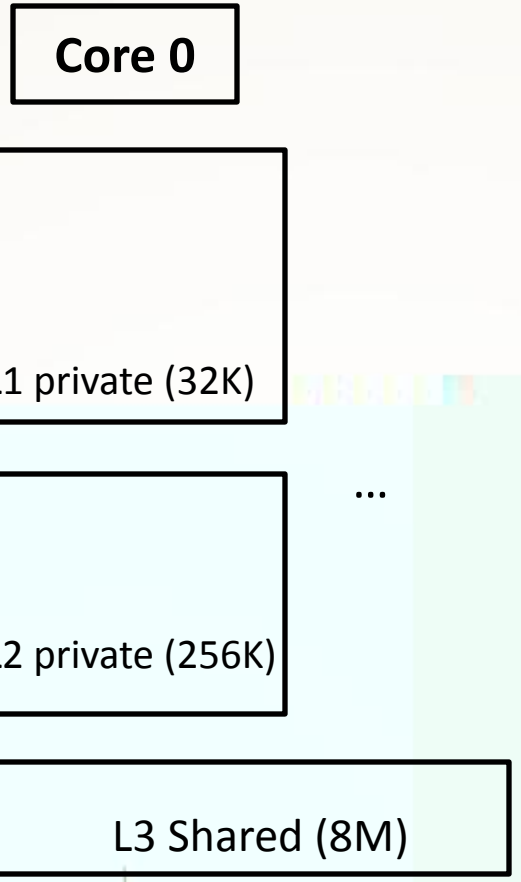
- Transpose ***m * n*** matrix to ***n * m***

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix} \Rightarrow \begin{pmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{pmatrix}$$

Input Matrix
16K x 16K

Output Matrix
16K x 16K

Intel Nehalem

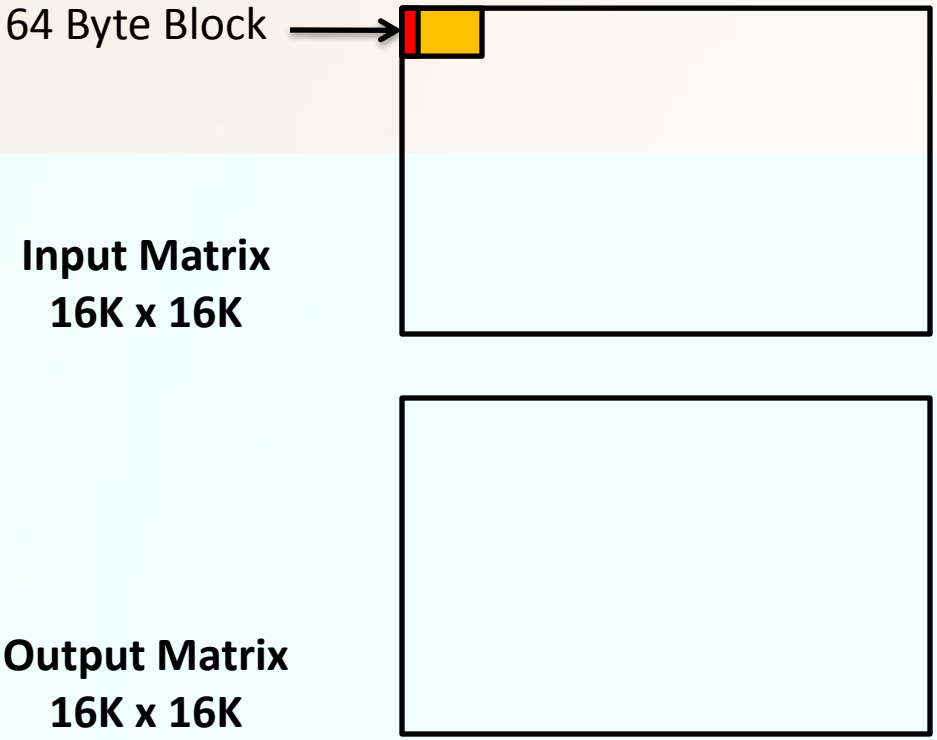


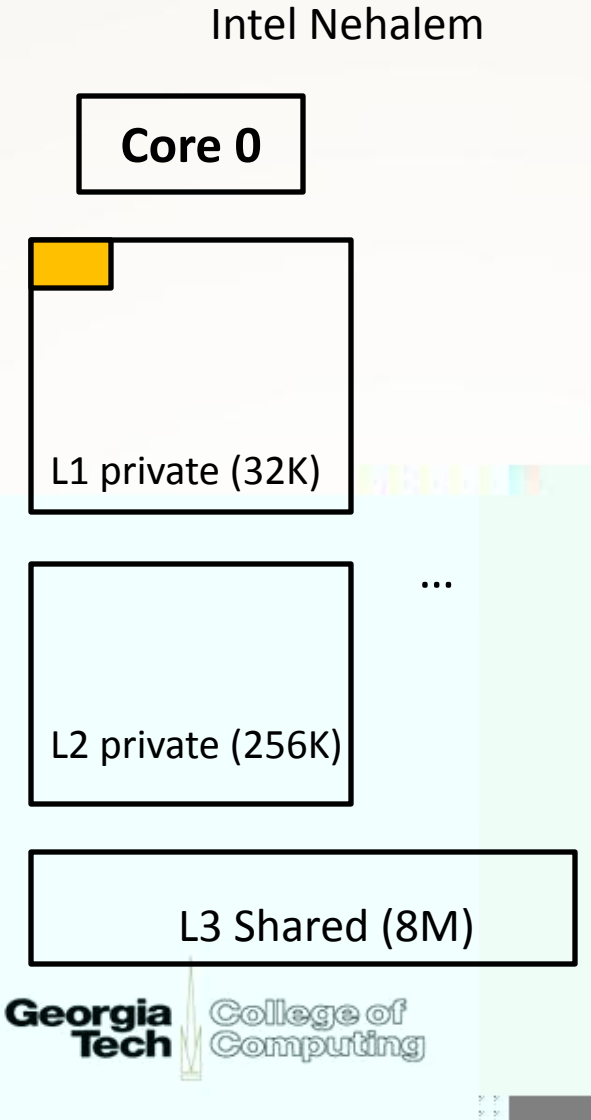
Georgia Tech College of Computing

Result Analysis - Transpose

- Transpose ***m * n*** matrix to ***n * m***

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix} \Rightarrow \begin{pmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{pmatrix}$$





Georgia Tech College of Computing

Result Analysis - Transpose

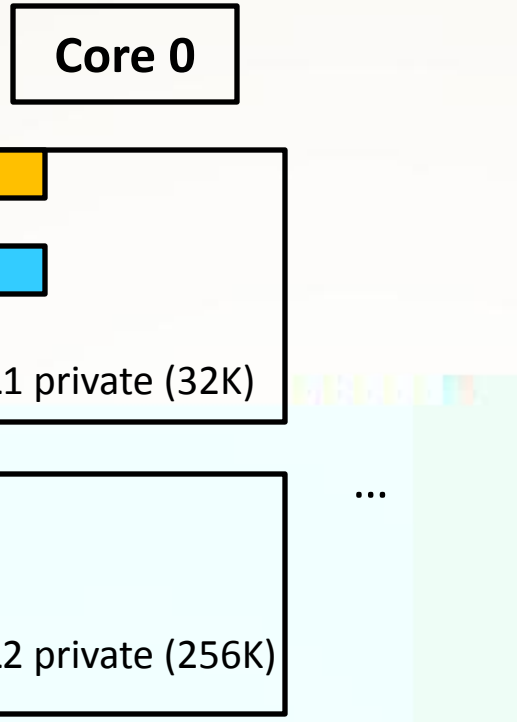
- Transpose ***m * n*** matrix to ***n * m***

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix} \Rightarrow \begin{pmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{pmatrix}$$

Input Matrix
16K x 16K

Output Matrix
16K x 16K

Intel Nehalem

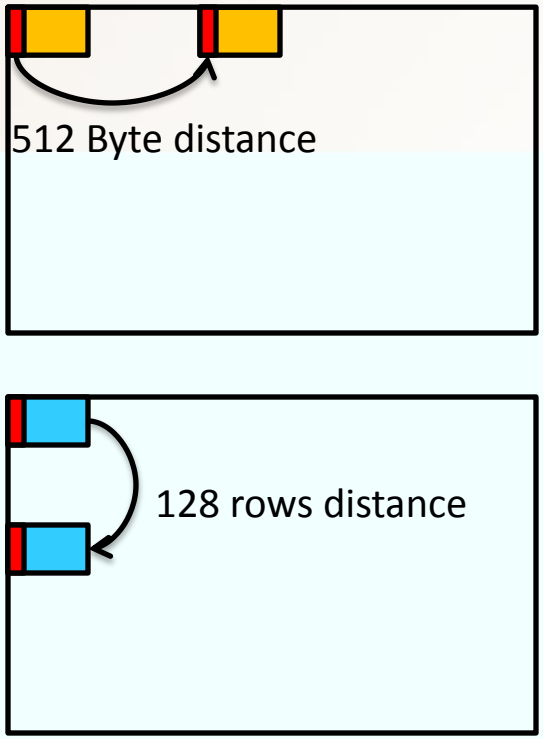


Georgia Tech College of Computing

Result Analysis - Transpose

- Transpose ***m * n*** matrix to ***n * m***

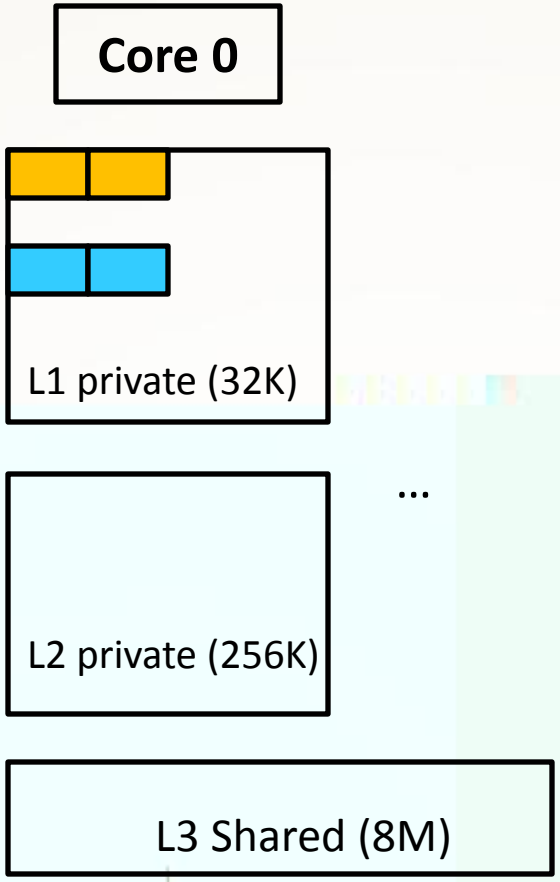
$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix} \Rightarrow \begin{pmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{pmatrix}$$



Input Matrix
16K x 16K

Output Matrix
16K x 16K

Intel Nehalem

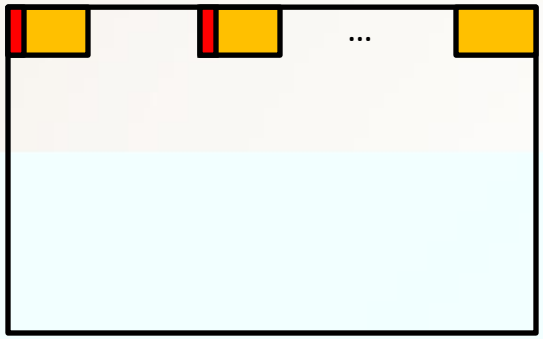


Georgia Tech College of Computing

Result Analysis - Transpose

- Transpose ***m * n*** matrix to ***n * m***

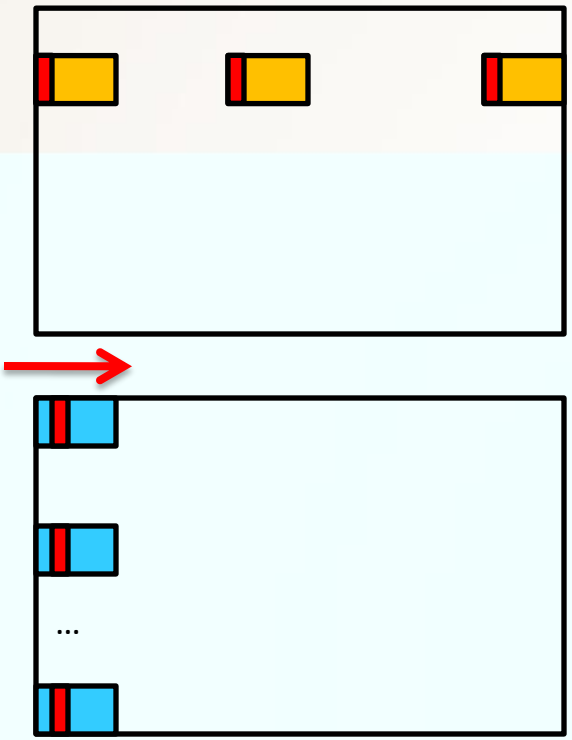
$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix} \Rightarrow \begin{pmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{pmatrix}$$



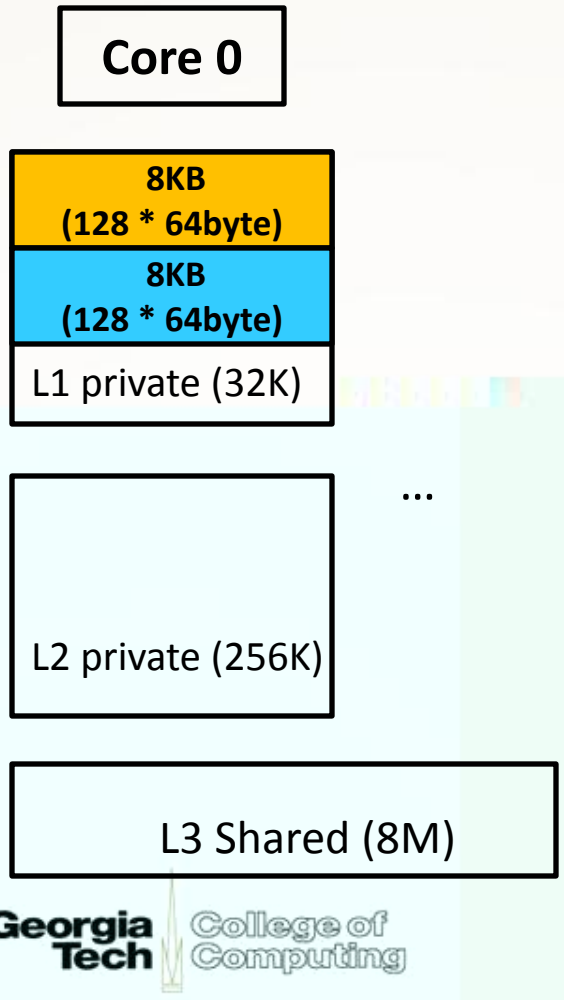
Result Analysis - Transpose

- Transpose ***m * n*** matrix to ***n * m***

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix} \Rightarrow \begin{pmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{pmatrix}$$



Intel Nehalem

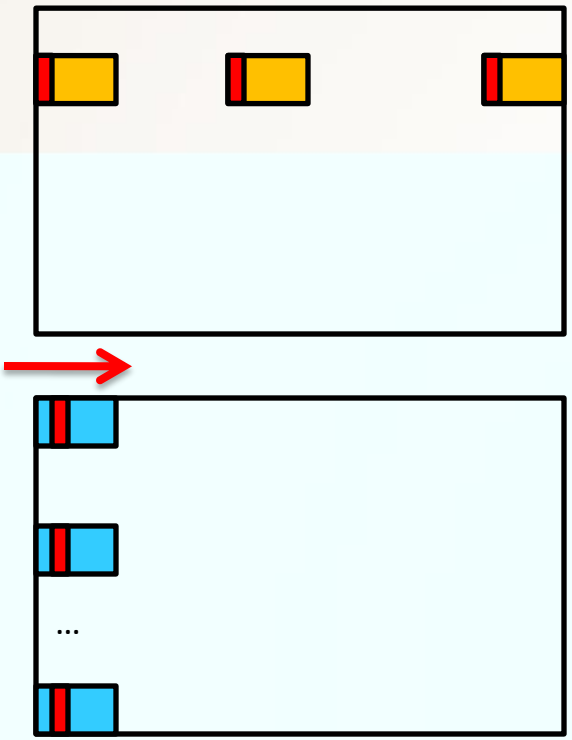


Georgia Tech College of Computing

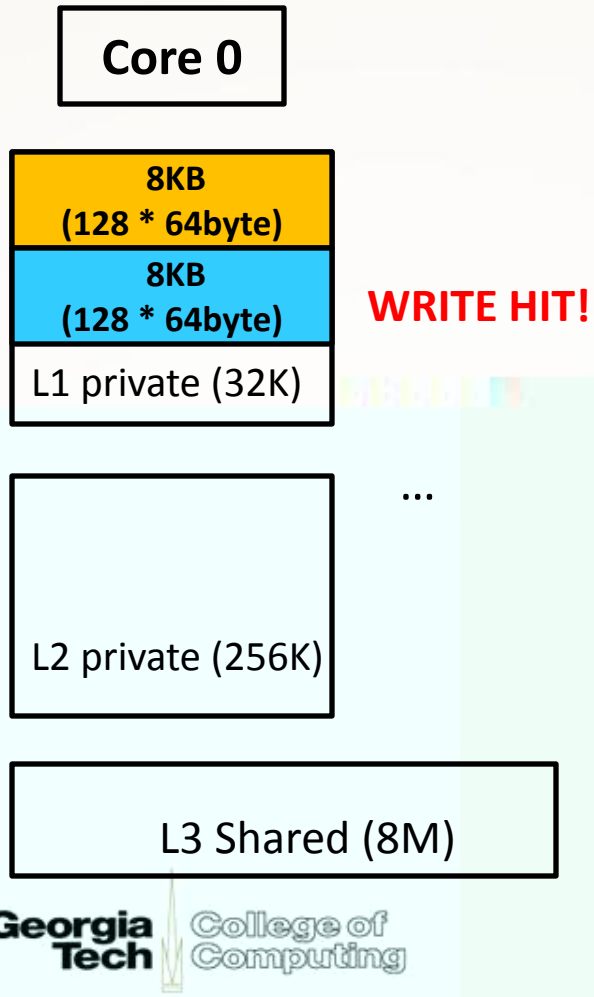
Result Analysis - Transpose

- Transpose ***m * n*** matrix to ***n * m***

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix} \Rightarrow \begin{pmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{pmatrix}$$



Intel Nehalem



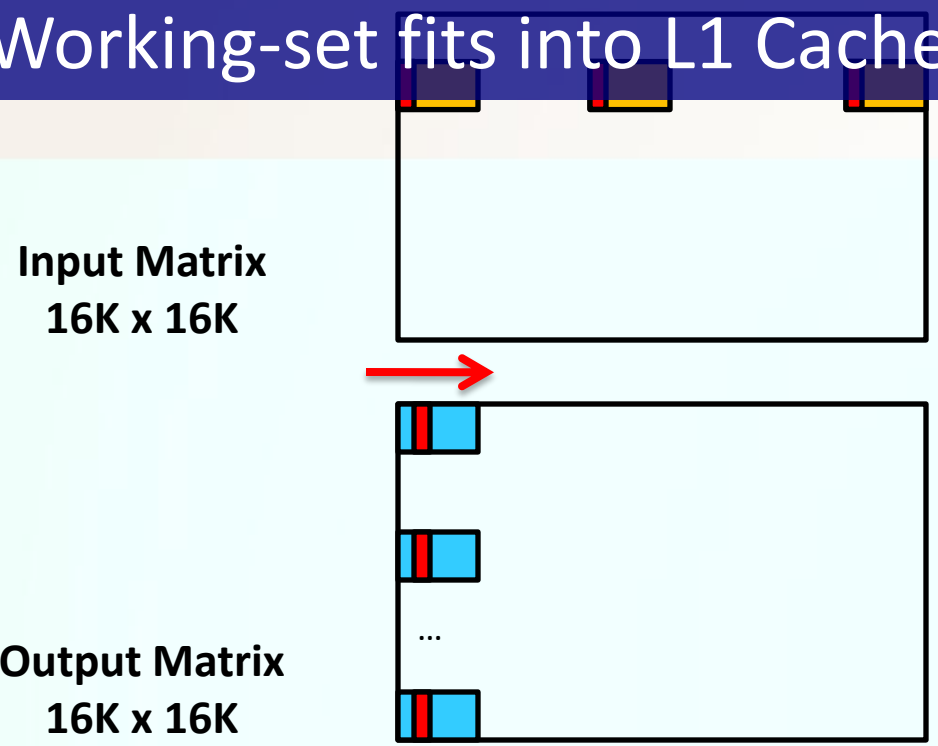
Georgia Tech College of Computing

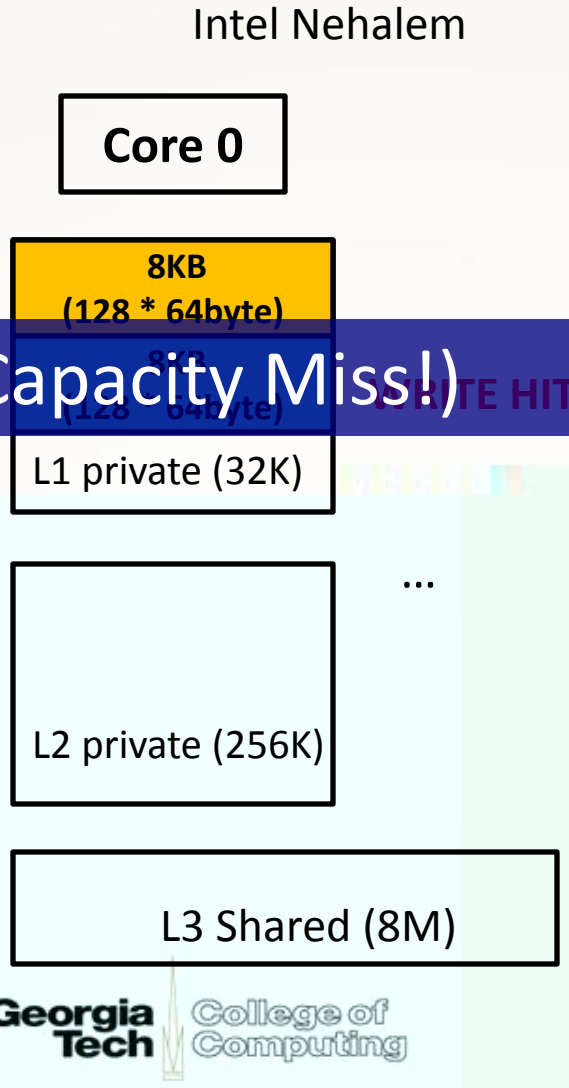
Result Analysis - Transpose

- Transpose ***m * n*** matrix to ***n * m***

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix} \Rightarrow \begin{pmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{pmatrix}$$

Working-set fits into L1 Cache (No Capacity Miss!)







Summary

- Choosing Optimal Number of Threads is Hard
- Thread Tailor Ease the Pain
  - Graph Representation
  - Combine Threads at Runtime



Thank you

Georgia Tech College of Computing