

Lecture 10

Midterm Review

Announcements

- Abe accounts
- Midterm on Thursday
- Sign up sheet
- No Quiz
- Office hours cancelled on Thursday

Blocked Matrix Multiplication

for i = 0 to N-1

for j = 0 to N-1

// load each block C[i,j] into cache, once : n^2

// B = n/N = cache line size

for k = 0 to N-1

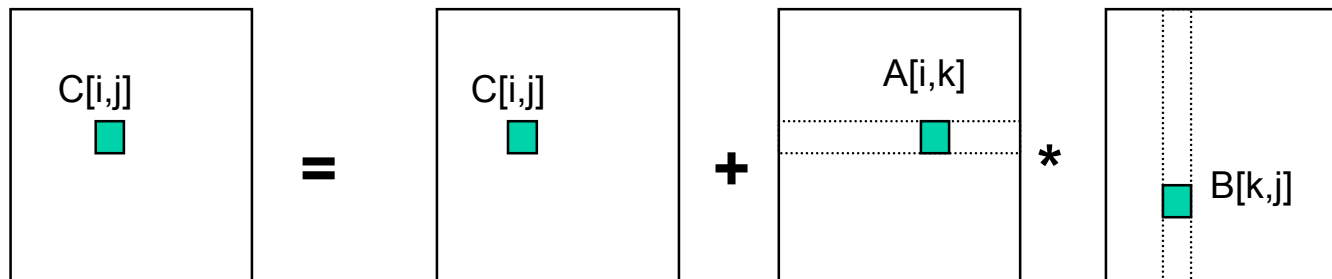
// load each block A[i,k] and B[k,j] N^3 times

// = $2N^3 \times (n/N)^2 = 2Nn^2$

C[i,j] += A[i,k] * B[k,j] // do the matrix multiply

// write each block C[i,j] once : n^2

Total: $(2*N+2)*n^2$

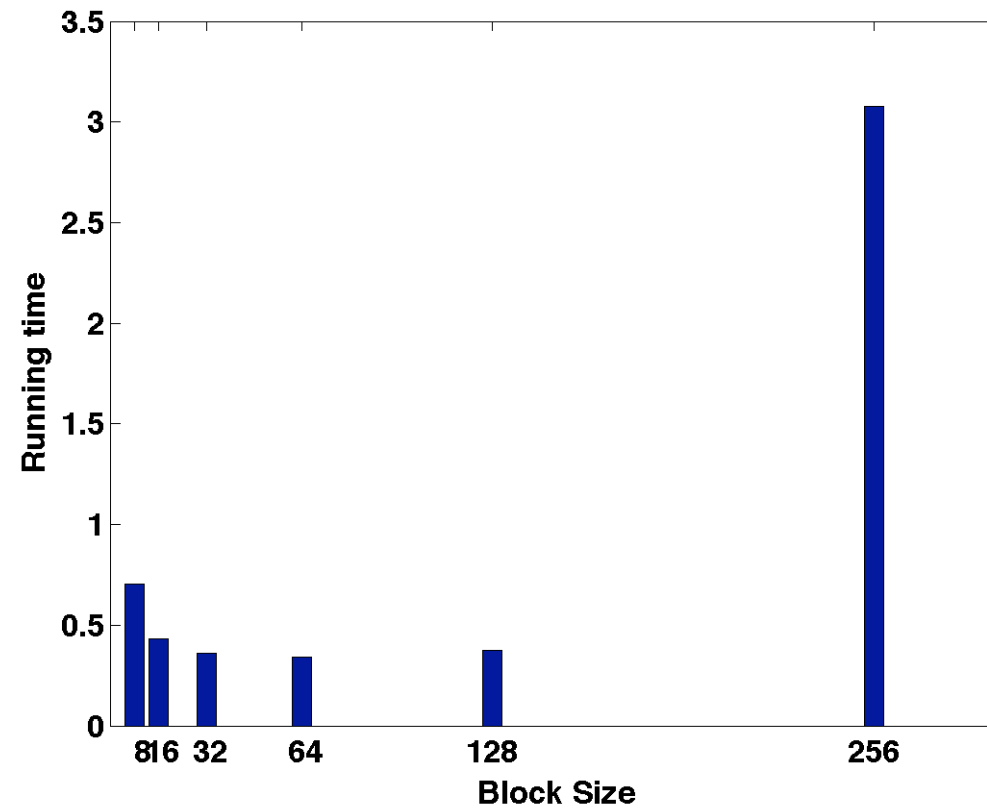


The results

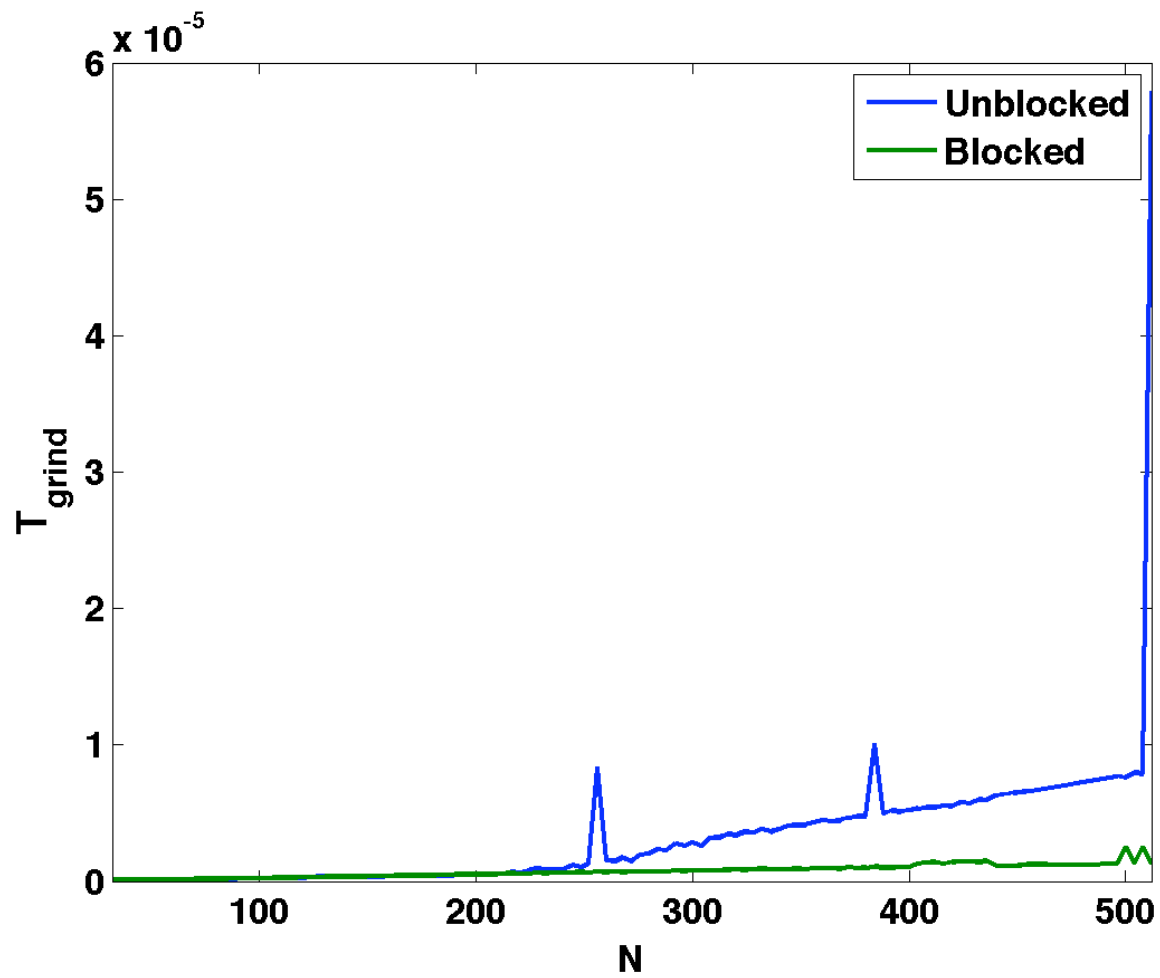
N,B	Unblocked Time (sec) (grind time in microsec)	Blocked Time
256, 64	0.55 (0.033)	0.048 (.0029)
512,128	15 (0.11)	0.37 (.0028)

Optimizing the block size

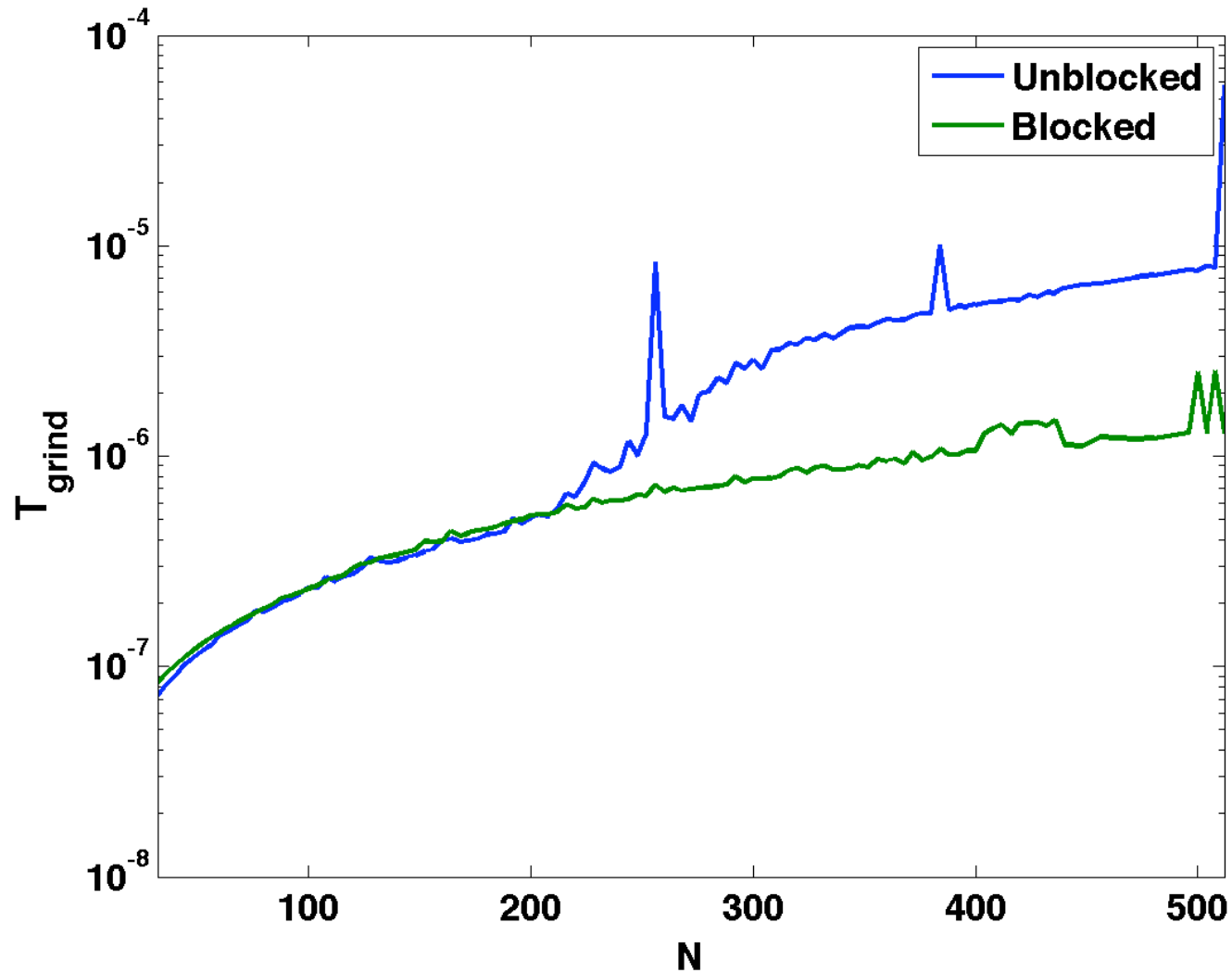
N=512



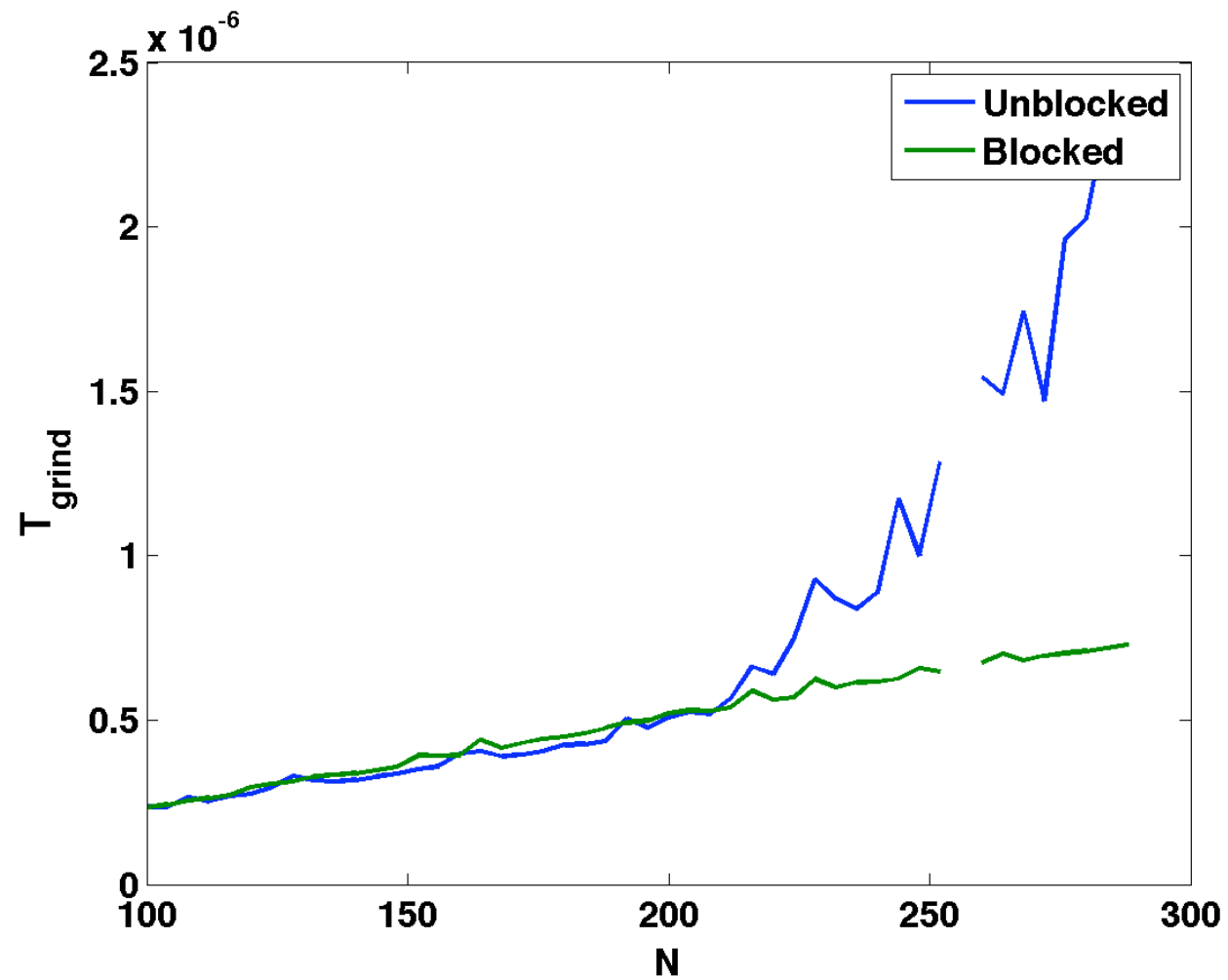
Comparison



Log Y scale improves readability



Crossover



Midterm review

Terms and concepts

- Know the definition and significance of
- Parallel speedup and efficiency, super-linear speedup, strong scaling, weak scaling
- Amdahl's law, serial bottlenecks
- Granularity
- Data dependencies, loop carried dependence

Terms and concepts

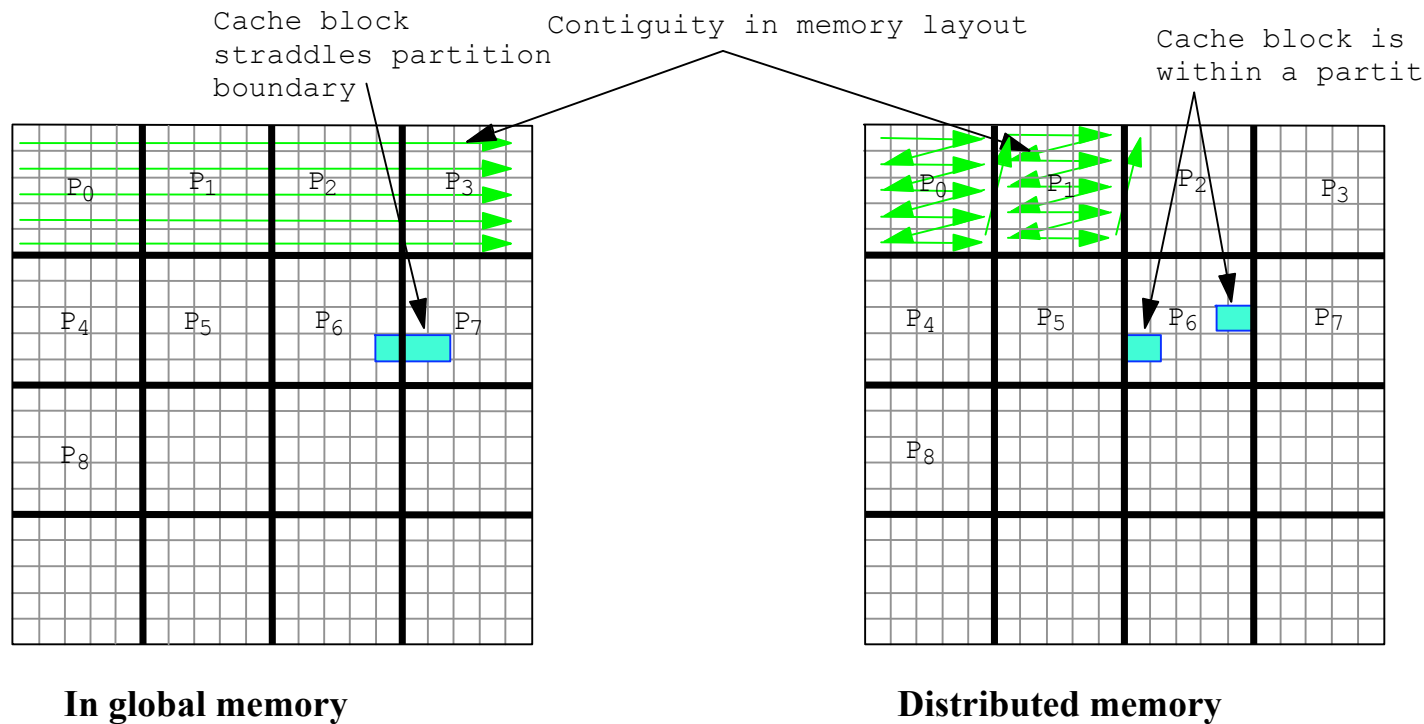
- PRAMs, CRCW
- SPMD, MIMD, SIMD
- Multiprocessors and multicomputers
- Cache coherence and consistency
- False sharing
- Snooping, directories
- NUMAs and SMPs
- Processor geometry

Implementation techniques

- Threads
- pthreads
- OpenMP
- Mutexes, semaphores, and barriers
- Critical sections, race conditions
- Red-Black/ Odd-even

False sharing in higher dimension arrays

- Compare with distributed memory solution



*Parallel
Computer
Architecture,
Culler,
Singh, &
Gupta*

Practice problems

The following code performs barrier synchronization, where
 n_{proc} = number of processors.

Explain how the code works, demonstrating correct operation
on 3 threads

Practice problems

```
Void barrier( . . . ){  
    P(arrival);  
    Count++;  
    If (Count < n$proc) V(arrival);  
    Else V(departure);  
    P(departure);  
    Count--;  
    If (count > 0) V(departure);  
    Else V(arrival);  
    Return;  
}
```

**Why are two binary
semaphores needed:
arrival and departure**

**How must they be
initialized to ensure
correctness?**

**How must Count be
initialized?**

OpenMP

List all possible outputs that result when the following OpenMP annotated C code is executed:

```
#pragma omp parallel for shared(j,k)
for ( int i=0, j=0, k=0; i< 2; i++ )
    j = j + 10;
    k = j + 10;
}
cout << "k = " << k << endl;
```

OpenMP

Now with critical sections

```
#pragma omp parallel for shared(j,k)
for ( int i=0, j=0, k=0; i< 2; i++ )
    #pragma omp critical
    j = j + 10;
    #pragma omp critical
    k = j + 10;
}
cout << "k = " << k << endl;
```

Consistency

Assume that memory is sequentially consistent
How many times can procedure foo() be run?

Shared int X=0, Y=0;

Thread 0	Thread 1
X = 1;	Y = 1;
If (y==0)	If (x == 0)
foo();	foo();