Parallel Programming
0024

Matrix Multiplication

Spring Semester 2010

Outline

- Discussion of last assignment
- Presentation of new assignment
  - Introduction to matrix multiplication
  - Issues in parallelizing matrix multiplication
  - Performance measurements
- Questions/Comments?

Discussion of Homework 4

Questions to be answered

- Is the parallel version faster?
- How many threads give the best performance?
- What is the influence of the CPU model/CPU frequency?

- RUNNABLE
- TIMED_WAITING
- TERMINATED

- NEW
- BLOCKED
- WAINTING

new Thread()	NEW
thread.start()	RUNNABLE
synchronized	BLOCKED
sleep()	TIMED_WAITING
join()	WAITING*
interrupt()	TERMINATED

Presentation of Homework 5

Matrix multiplication

- Problem: Given two matrices A, B of size N * N. Compute the matrix product C = A * B with

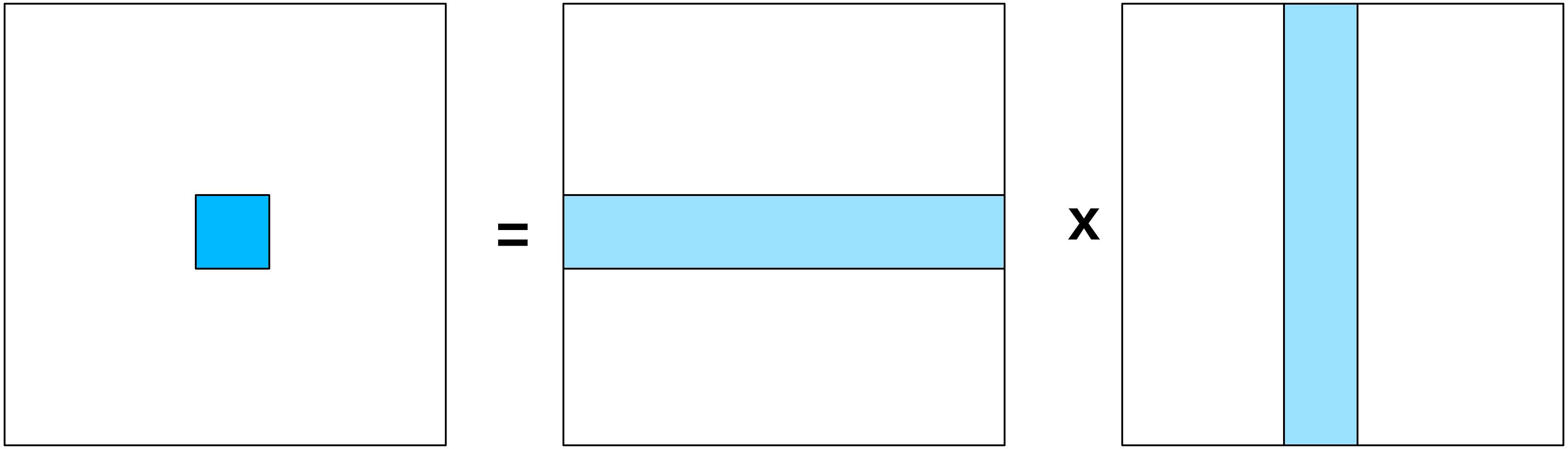
  $C_{ij} = \text{Sum}(A_{ik} * B_{kj})$ $(0 <= k < N)$

  A, B elements are double-precision floating point numbers (“`double`”)

- Assume that A and B are dense matrices
  - Sparse matrices have many zero elements
    - Only the non-zero elements are stored
  - Dense matrices have mostly non-zero elements
    - Each matrix requires N^2 storage cells

Parallel matrix multiplication

Which operations can be done in parallel?



Programming matrix multiplication

- Java code for the loop nest is easy.

```
double[][] a = new double[N][N];
double[][] b = new double[N][N];
double[][] c = new double[N][N];

for (i=0; i<N; i++) {
    for (j=0; j<N; j++) {
        a[i][j] = rand.nextDouble();
        b[i][j] = rand.nextDouble();
        c[i][j] = 0.0;
    }
}
```

```
for (i=0; i<N; i++) {
    for (j=0; j<N; j++) {
        for (k=0; k<N; k++) {
            c[i][j] +=  a[i][k]*b[k][j];
        }
    }
}
```

Parallel matrix multiplication

- Data partitioning based on
  - Input matrix A
  - Input matrix B
  - Output matrix C

Parallel matrix multiplication

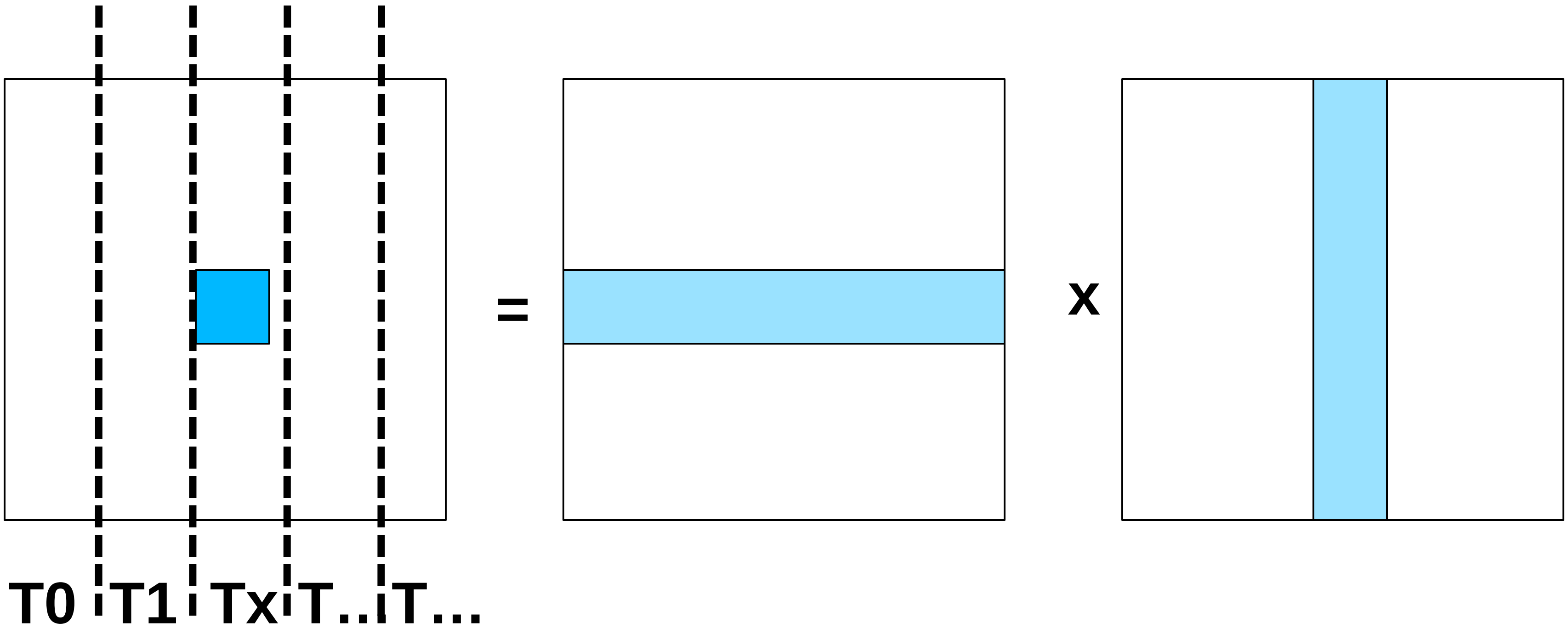
- Data partitioning based on
    - Input matrix A
    - Input matrix B
    - Output matrix C

- We assume that all threads can read inputs A and B
    - Start with partitioning of output matrix C
        - No need to use `synchronized` !

Parallel matrix multiplication

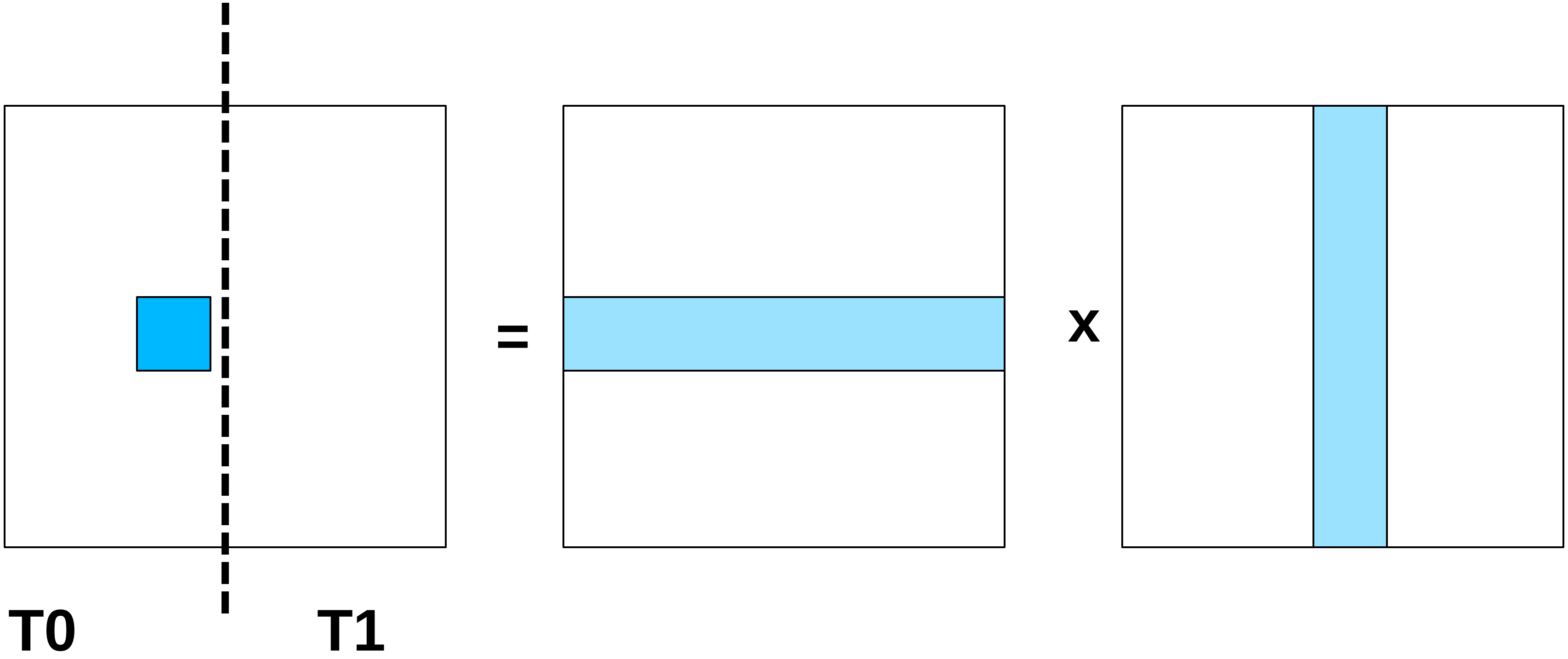
Each thread computes its share of the output C

Partition C by columns



Two threads

One thread computes columns 0 .. N/2, the other columns N/2+1 .. N-1



Two threads

Thread 0

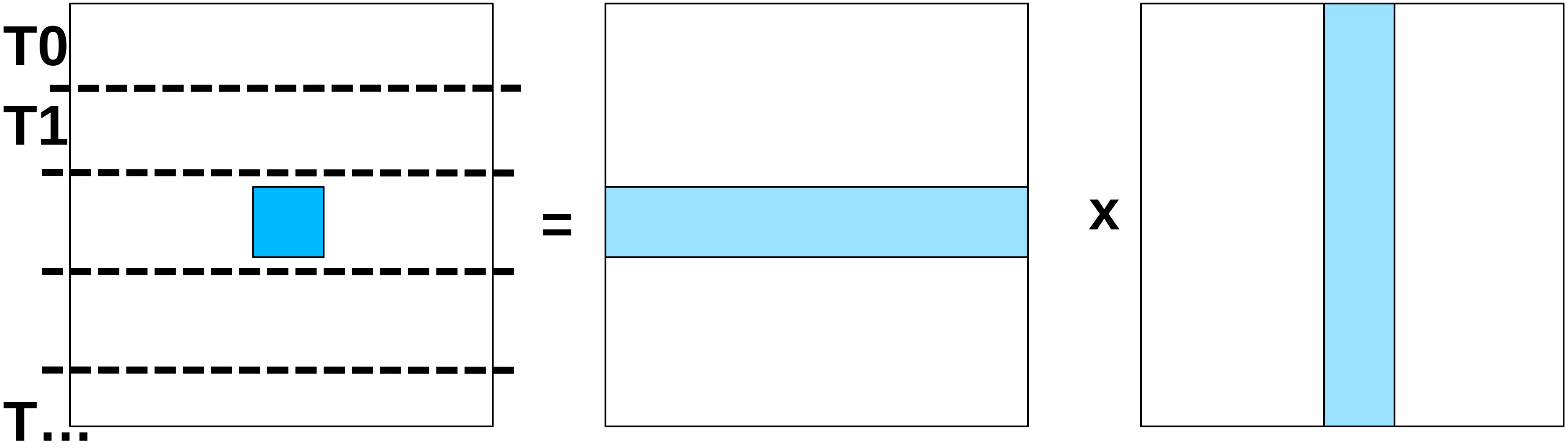
```
for (i=0; i<N; i++) {
    for (j=0; j<N; j++) {
        for (k=0; k<N/2; k++) {
            c[i][j] +=  a[i][k]*b[k][j];
        }
    }
}
```

Thread 1

```
for (i=0; i<N; i++) {
    for (j=0; j<N; j++) {
        for (k=N/2; k<N; k++) {
            c[i][j] +=  a[i][k]*b[k][j];
        }
    }
}
```

Other aspects

- Partition C by columns or by rows?



Other aspects

- What should be the order of the loops?

```
for (i=0; i<N; i++) {
    for (j=0; j<N; j++) {
        for (k=0; k<N; k++) {
            c[i][j] +=  a[i][k]*b[k][j];
        }
    }
}
```

- Or?

```
for (k=0; i<N; i++) {
    for (i=0; j<N; j++) {
        for (j=0; k<N; k++) {
            c[i][j] +=  a[i][k]*b[k][j];
        }
    }
}
```

- Or?

Performance Measurement

# of threads /matrix size	1	2	4	8	16	32	64	…	1024?
100	x								
200	x								
…									
10,000?									

Any Questions?

- synchronized
- Thread, Runnable
- wait(), notify(), notifyAll()
- Thread States