Matrix Multiplication

CSCI 4576/5576

Outline

Memory and Matrix Multiplication

- Storage
- Blocking
- Cachegrind

Library examples

- HDF5
- MKL

Discuss homework

Warm-up: Memory

Program makes a memory reference

- If it’s in cache, it gets returned immediately.
- If not: cache miss
  - New cache line is fetched

Cache Lines

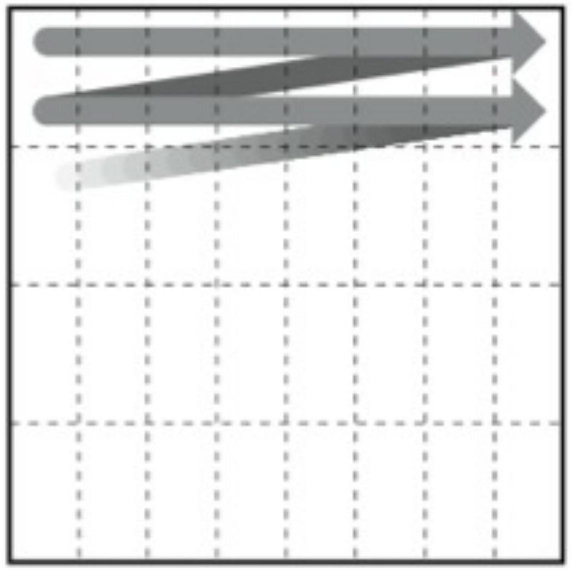
- Good performance: use all the values in the line
- Bad performance: use a single element
  - Lots of memory bandwidth

Matrix storage

Matrix is 1D in memory

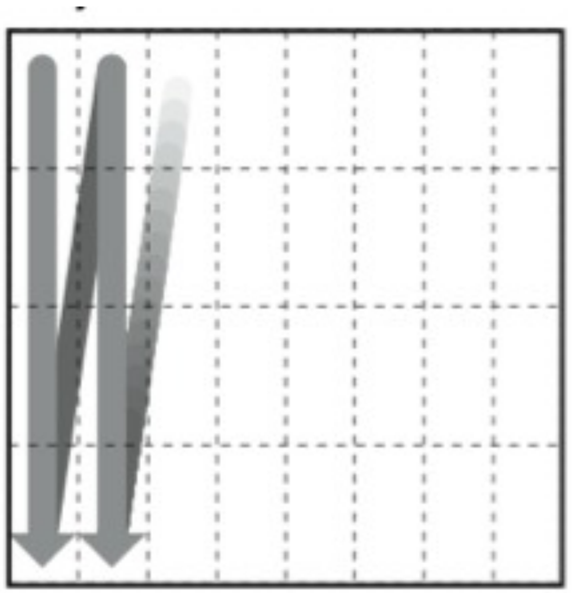
Row major

- A(i,j) = A + i*n + j



Column major

- A(i,j) = A + i + j*n

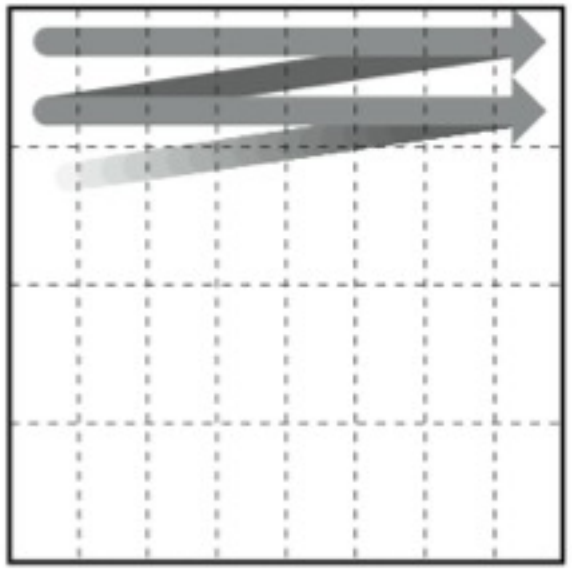


Example

```
int *M = new int [N*N];

// Access the matrix
for(int r=0; r<N; ++r)
{
   for(int c=0; c<N; ++c)
   {
      M[r*N + c] = r*c;
   }
}

// Cleanup
delete[] M;
```

Example

```
int *M = new int [N*N];

// Access the matrix
for(int r=0; r<N; ++r)
{
   for(int c=0; c<N; ++c)
   {
      M[r*N + c] = r*c;
   }
}

// Cleanup
delete[] M;
```

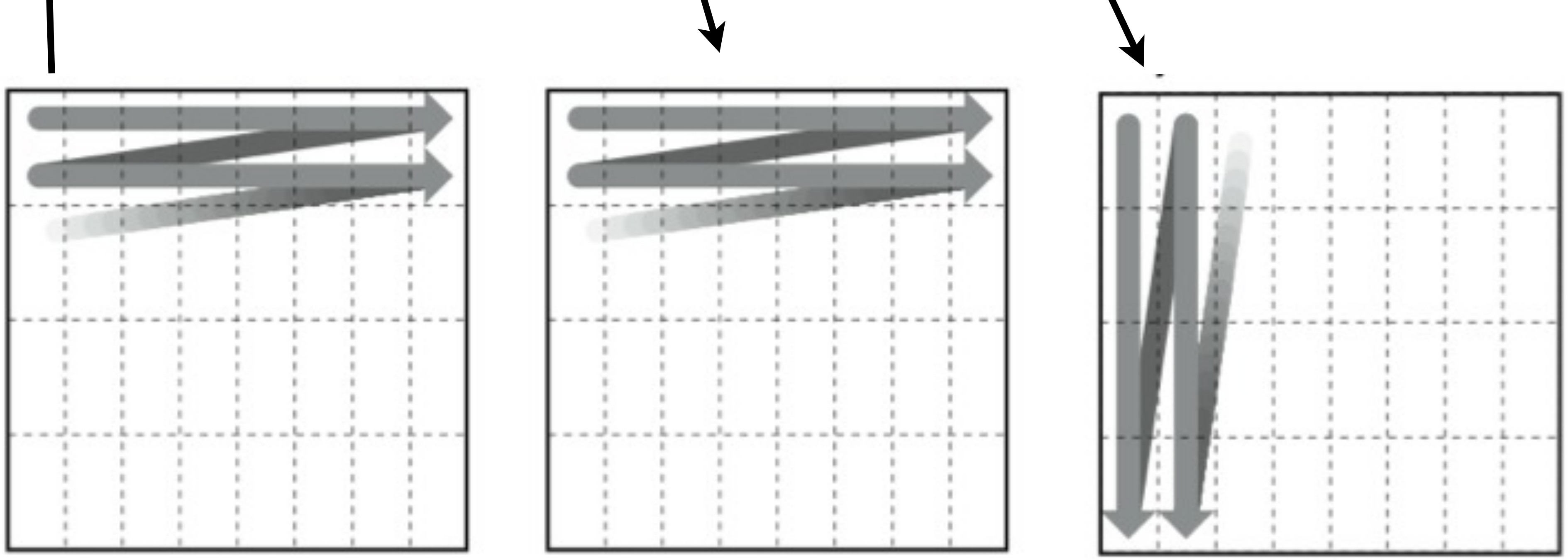


Matrix Multiply

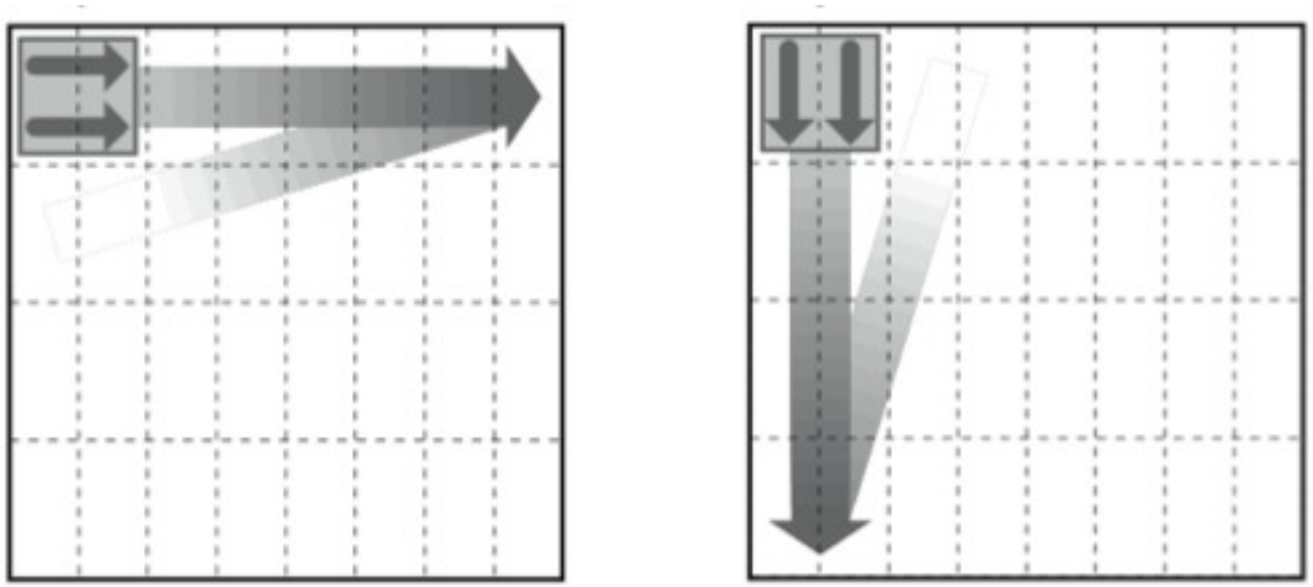
```
for(int i=0; i<N; ++i)
{
   for(int j=0; j<N; ++j)
   {
      double sum = 0;
      for(int k=0; k<N; ++k)
      {
          sum = sum + A[i*N + k]*B[k*N + j];
      }
      M[i*N + j] = sum;
   }
}
```

Matrix Multiply

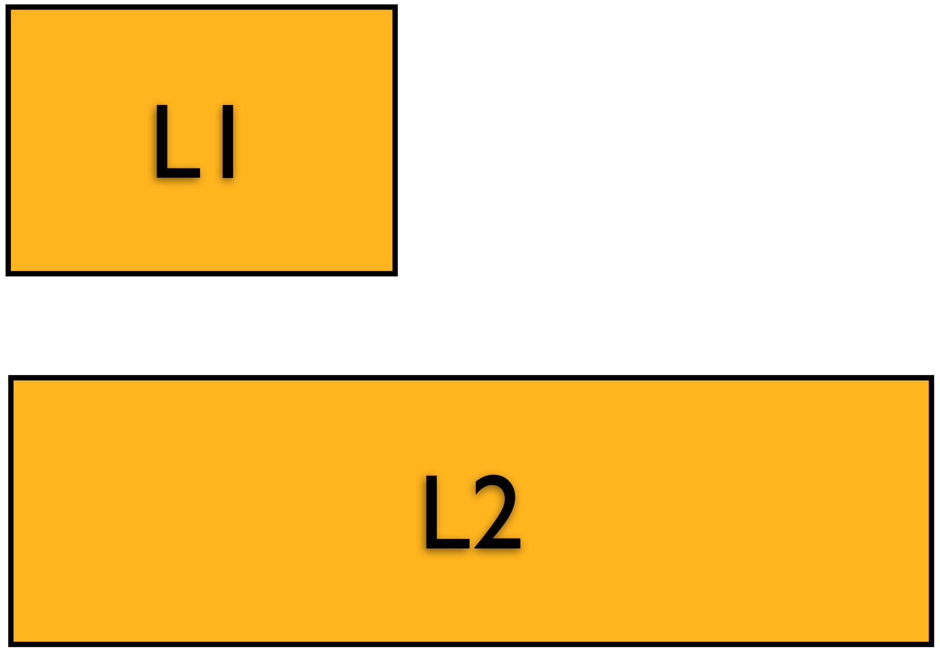
```
for(int i=0; i<N; ++i)
{
    for(int j=0; j<N; ++j)
    {
        double sum = 0;
        for(int k=0; k<N; ++k)
        {
            sum = sum + A[i*N + k]*B[k*N + j];
        }
        M[i*N + j] = sum;
    }
}
```



Block



Memory and Cachegrind


L1

L2

I1 D1

lli lld

example: row, col

```
int N = atoi(argv[1]);

// Allocate
int **M = new int * [N];
for (int i = 0; i < N; i++)
   M[i] = new int [N];

// Access the matrix
for(int r=0; r<N; ++r)
   for(int c=0; c<N; ++c)
      M[r][c] = r*c;

// Cleanup
for (int i = 0; i < N; i++)
   delete[] M[i];
delete[] M;
```

Cachegrind

read write

```
==91220== I   refs:      433,595,222
==91220== I1  misses:          2,590
==91220== LLi misses:          2,141
==91220== I1  miss rate:        0.00%
==91220== LLi miss rate:        0.00%
==91220==
==91220== D   refs:      277,732,903  (226,908,950 rd   + 50,823,953 wr)
==91220== D1  misses:      1,589,774  (     13,180 rd   +  1,576,594 wr)
==91220== LLd misses:      1,585,912  (      9,563 rd   +  1,576,349 wr)
==91220== D1  miss rate:         0.5% (        0.0%     +        3.1%  )
==91220== LLd miss rate:         0.5% (        0.0%     +        3.1%  )
==91220==
==91220== LL refs:         1,592,364  (     15,770 rd   +  1,576,594 wr)
==91220== LL misses:       1,588,053  (     11,704 rd   +  1,576,349 wr)
==91220== LL miss rate:          0.2% (        0.0%     +        3.1%  )
```

example: col, row

```
int N = atoi(argv[1]);

// Allocate
int **M = new int * [N];
for (int i = 0; i < N; i++)
   M[i] = new int [N];

// Access the matrix
for(int c=0; c<N; ++c)
   for(int r=0; r<N; ++r)
      M[r][c] = r*c;

// Cleanup
for (int i = 0; i < N; i++)
   delete[] M[i];
delete[] M;
```

Cachegrind

```
==91221== I   refs:      433,595,222
==91221== I1  misses:          2,590
==91221== LLi misses:          2,141
==91221== I1  miss rate:        0.00%
==91221== LLi miss rate:        0.00%
==91221==
==91221== D   refs:      277,732,903  (226,908,950 rd   + 50,823,953 wr)
==91221== D1  misses:     28,149,149  (  3,137,555 rd   + 25,011,594 wr)
==91221== LLd misses:     23,167,621  (     59,194 rd   + 23,108,427 wr)
==91221== D1  miss rate:        10.1% (        1.3%     +       49.2%  )
==91221== LLd miss rate:         8.3% (        0.0%     +       45.4%  )
==91221==
==91221== LL refs:        28,151,739  (  3,140,145 rd   + 25,011,594 wr)
==91221== LL misses:      23,169,762  (     61,335 rd   + 23,108,427 wr)
==91221== LL miss rate:          3.2% (        0.0%     +       45.4%  )
```