Overview of Mini-Project 1

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

Outline

- **CPU Achievable Peak Performance**
- Matrix-Multiplication Discussion
- Overview of Mini-Project 1
- A Quick Review:
  - SIMD Parallelism
  - OpenMP Pragmas
  - CPU Memory Hierarchy
  - False Sharing
- Mini-Project 1.1 - Matrix Multiply
- Mini-Project 1.2 - K-Means

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

Peak Single-precision Performance

limit in theory

- Intel Sandy Bridge 3.5 Ghz, 4 cores * 2 sockets

[GFLOPS]	1 core	2 cores	4 cores	8 cores
None, scalar	**7**	14	28	**56**
SSE	28	56	112	224
AVX	56	112	224	**448**

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

for intel sandy Bridge : 2 ALU

Most CPU applications achieve a fraction of this

In addition …

- Deep memory hierarchy
  - Registers
  - L1 cache
  - L2 cache
  - TLB (Translation Lookaside Buffer) ⇒ a table used to translate virtual mem & physical mem

Assignment Project Exam Help
https://powcoder.com

- Other forms of parallelism
  - Pipelining
  - Instruction-level parallelism (multiple functional units) 2 mult's same time

Add WeChat powcoder

- "Free operations" are not free

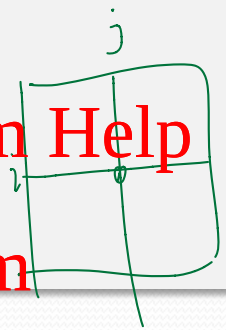
Outline

- CPU Achievable Peak Performance
- **Matrix-Multiplication Discussion**
- Overview of Mini-Project 1
- A Quick Review:
  - SIMD Parallelism
  - OpenMP Pragmas
  - CPU Memory Hierarchy
  - False Sharing
- Mini-Project 1.1 - Matrix Multiply
- Mini-Project 1.2 - K-Means

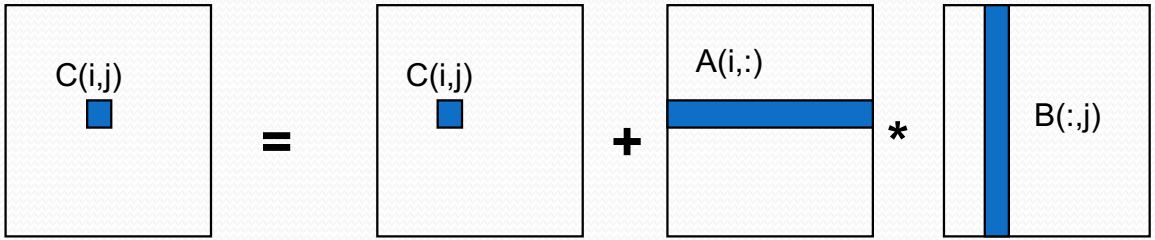
Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

Naïve Matrix Multiply

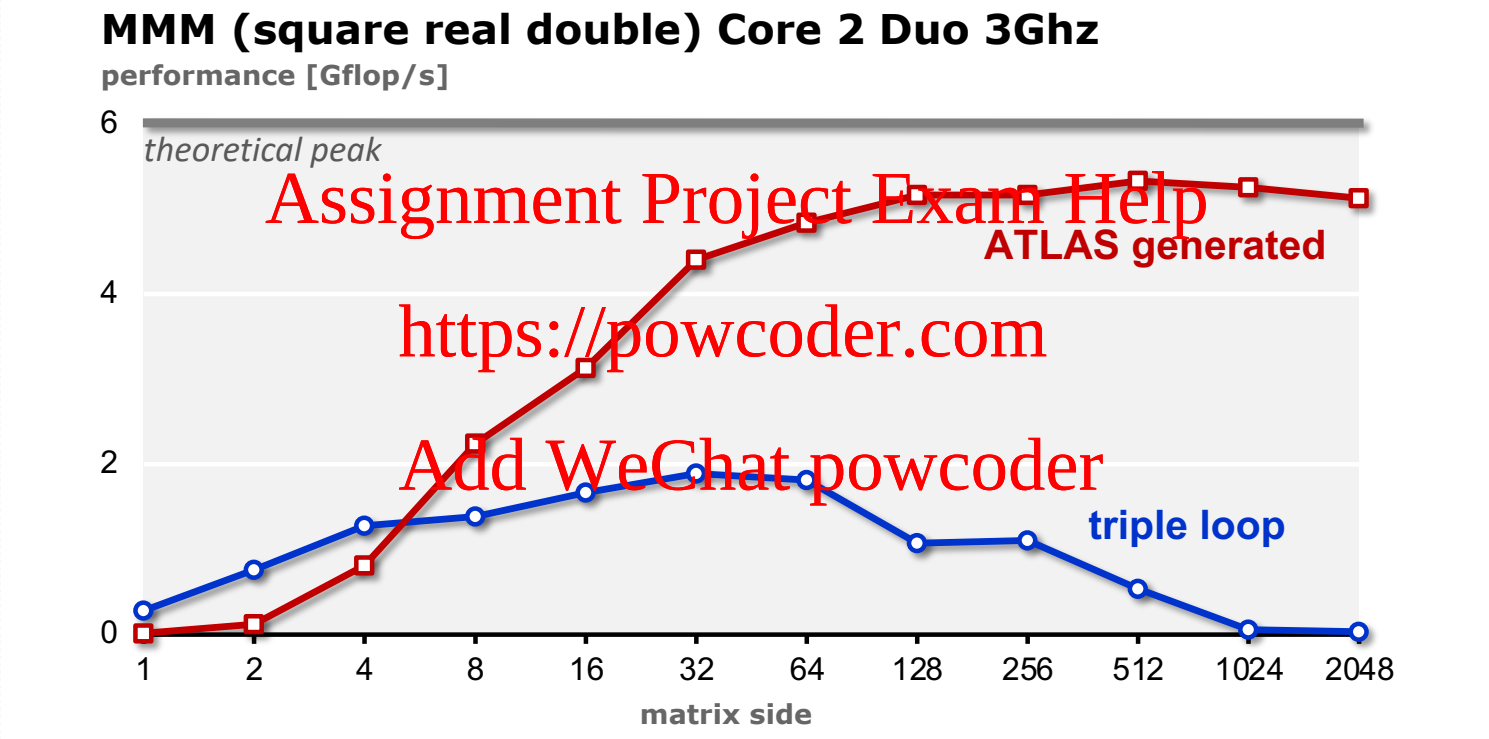
```
{implements C = C + A*B}
for i = 1 to n
    for j = 1 to n
        for k = 1 to n
            C(i,j) = C(i,j) + A(i,k) * B(k,j)
```



Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder



Matrix-Matrix Multiplication Speed



Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

- Huge performance difference for large sizes
- Great case study to learn memory hierarchy optimization

Note on Matrix Storage

- A matrix is a 2-D array of elements, but memory addresses are "1-D"
- Conventions for matrix layout
  - by column, or "column major" (Fortran default);
    `A[i,j]` at `i + j*n`
  - by row, or "row major" (C default)
    `A[i,j]` at `i*n + j`

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

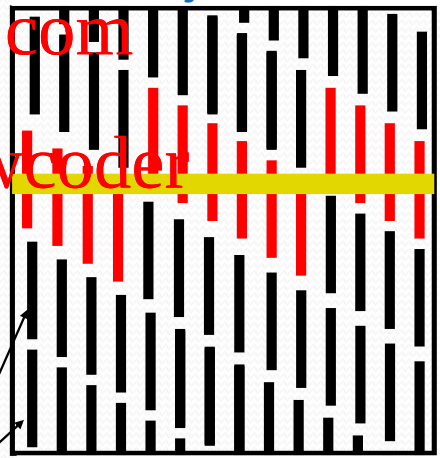
Column major

0	5	10	15
1	6	11	16
2	7	12	17
3	8	13	18
4	9	14	19

Row major

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15
16	17	18	19

Column major matrix in memory



cachelines

Yellow row of matrix is stored in red cachelines

- Column major (for now)

Naïve Matrix Multiply

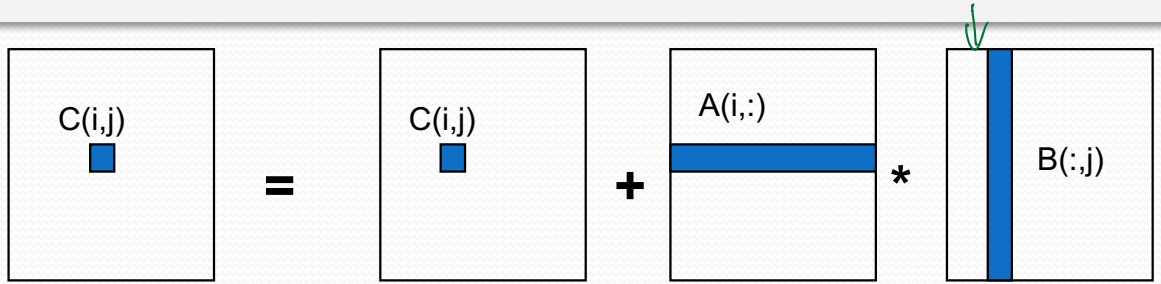
row major.

```
{implements C = C + A*B}
for i = 1 to n
  {read row i of A into fast memory}
  for j = 1 to n
     {read C(i,j) into fast memory}
     {read column j of B into fast memory}
     for k = 1 to n
        C(i,j) = C(i,j) + A(i,k) * B(k,j)
     {write C(i,j) back to slow memory}
```

A(i,:)

// O(n^3) slow loads

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder



Naïve Matrix Multiply

Number of slow memory references on unblocked matrix multiply

$m = n^3$ to read each column of B n times

$+ n^2$ to read each row of A once

$+ 2n^2$ to read and write each element of C once

$= n^3 + 3n^2$

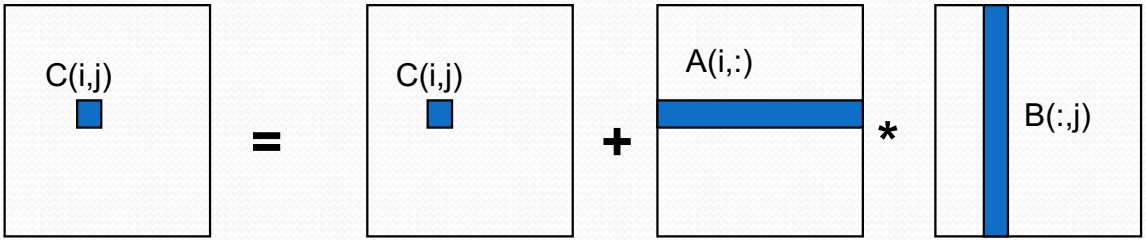
So $q = f / m = 2n^3 / (n^3 + 3n^2)$

≈ 2 for large n, no improvement over matrix-vector multiply

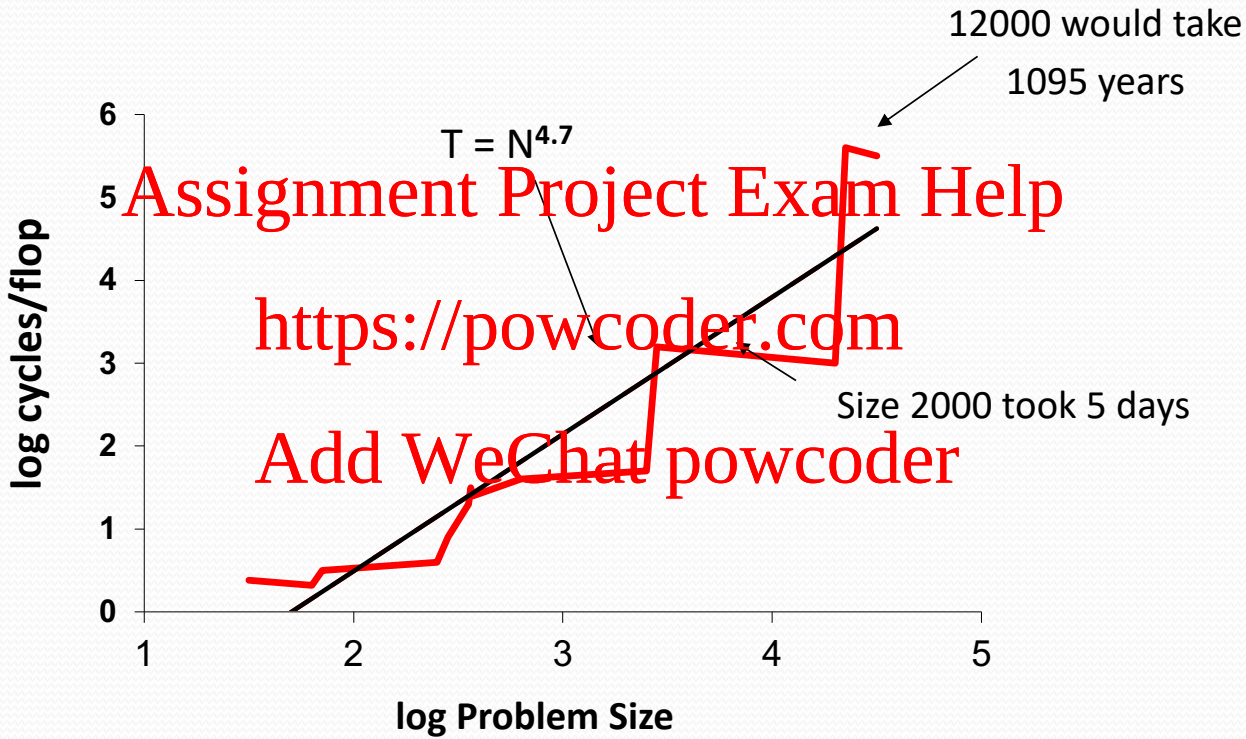
Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

Inner two loops are just matrix-vector multiply of row i of A times B

Similar for any other order of 3 loops



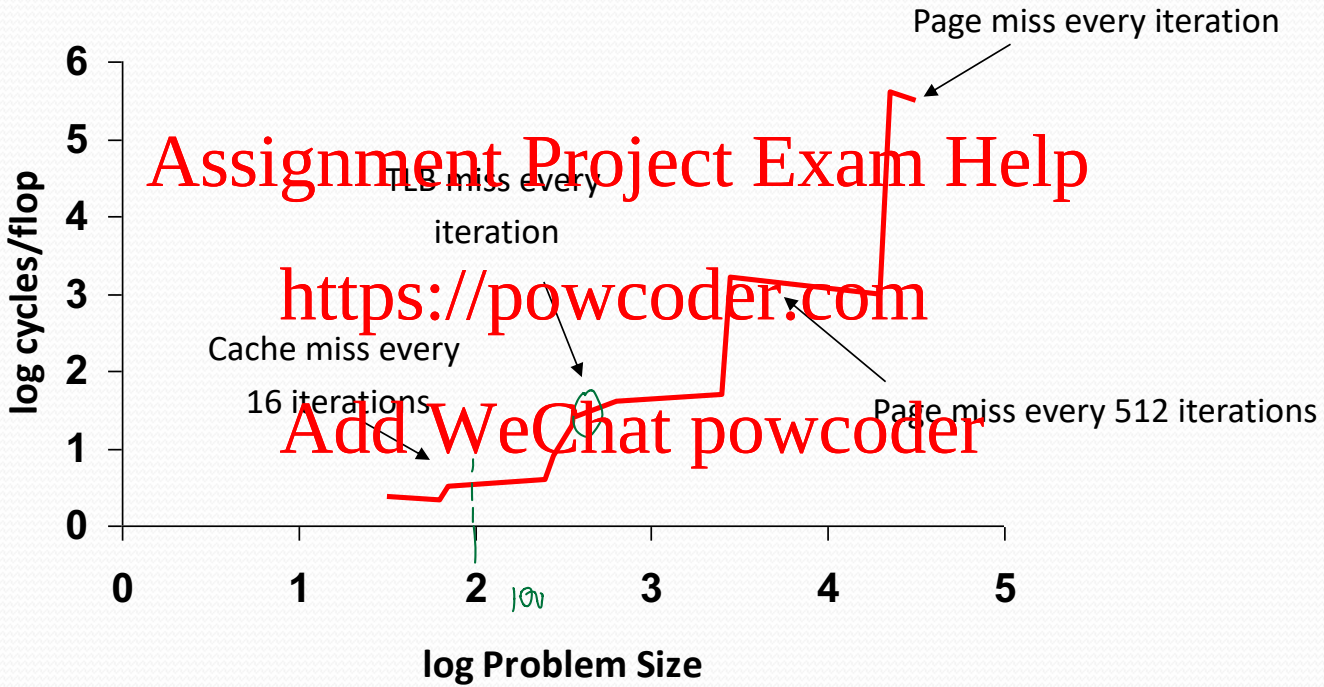
Naïve Matrix Multiply on RS/6000



Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

$O(N^3)$ performance would have constant cycles/flop

Naïve Matrix Multiply on RS/6000



Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

Slide source: Larry Carter, UCSD

Blocked (Tiled) Matrix Multiply

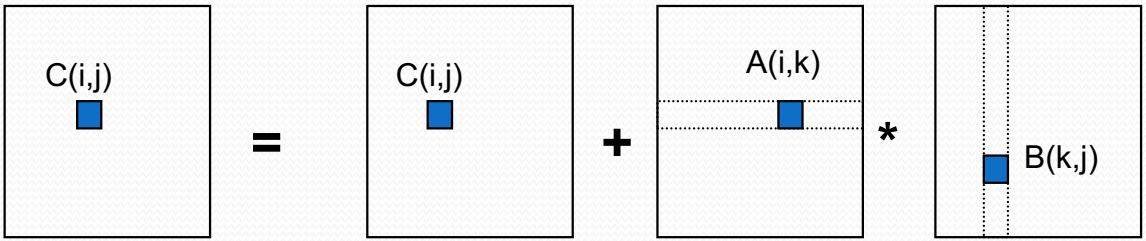
A,B,C = **NxN** matrices of **b x b** sub-blocks, **N = n / b**

```
for i = 1 to N
  for j = 1 to N
    {read block C(i,j) into fast memory}
    for k = 1 to N
      {read block A(i,k) into fast memory}      // O(N^3) slow loads
      {read block B(k,j) into fast memory}
      C(i,j) = C(i,j) + A(i,k) * B(k,j) {do a matrix multiply on blocks}
    {write block C(i,j) back to slow memory}
```

block should be smaller than cache

smallest

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder



Further Matrix-Multiply Optimizations in Real-world Libraries

- Block size adaptation for appropriate caches

Assignment Project Exam Help

- **Register-level blocking**

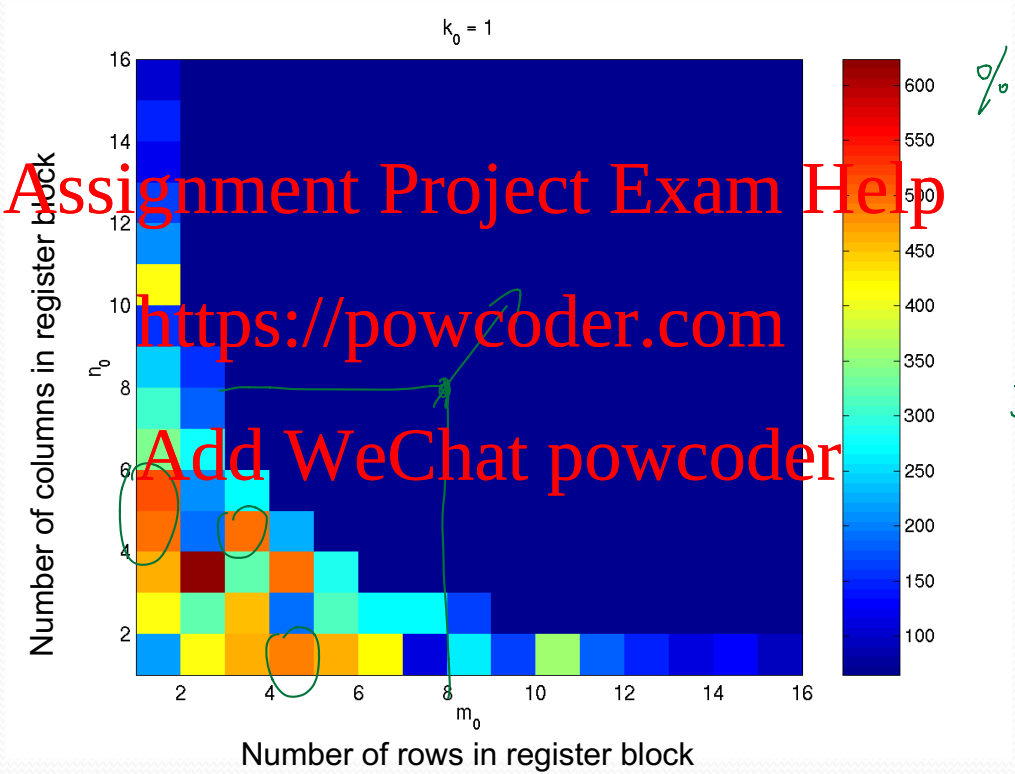
https://powcoder.com

- **Copy optimization (data layout)**

Add WeChat powcoder

- **Optimizing the mini-matrix-multiply (base case)**
- Multi-level blocking
- Multi-level copying

Register-level blocking



Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

%

3x4 / 4x3

A 2-D slice of a 3-D register-tile search space. The dark blue region was pruned.

Slide source: Jim Demmel, UCB

Copy optimization

- Copy input operands or blocks
  - Reduce cache conflicts
  - Constant array offsets for fixed size blocks
  - Expose page-level locality
  - Alternative: use different data structures from start (if users willing)

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

Original matrix
(numbers are addresses)

0	4	8	12
1	5	9	13
2	6	10	14
3	7	11	15

Reorganized into
2x2 blocks

0	2	8	10
1	3	9	11
4	6	12	13
5	7	14	15

Slide source: Jim Demmel, UCB

Outline

- CPU Achievable Peak Performance
- Matrix-Multiplication Discussion
- **Overview of Mini-Project 1**
- A Quick Review:
  - SIMD Parallelism
  - OpenMP Pragmas
  - CPU Memory Hierarchy
  - False Sharing
- Mini-Project 1.1 - Matrix Multiply
- Mini-Project 1.2 - K-Means

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

Mini-Project 1

The goal of this project is to use your understanding of parallel computing resources in a multicore microprocessor to optimize two fully functional applications.

Assignment Project Exam Help

The applications are Matrix Multiplication and K-Means Clustering. For a functional description of the applications, please refer to:

https://powcoder.com

- http://en.wikipedia.org/wiki/Matrix_multiplication
- http://en.wikipedia.org/wiki/K-means

Add WeChat powcoder

The code optimization techniques you may want to explore include:

- OpenMP Pro??? (omp parallel, omp for. omp atomic, omp reduction, …)
- Cache blocking
- SIMD and Sequential Optimizations

Mini-Project 1

- Mini-Project Due Monday, 3/8 @ 11:59PM EST
- Mini-Project One Review Thursday, 3/11 @ 6:00PM EST

Assignment Project Exam Help

https://powcoder.com

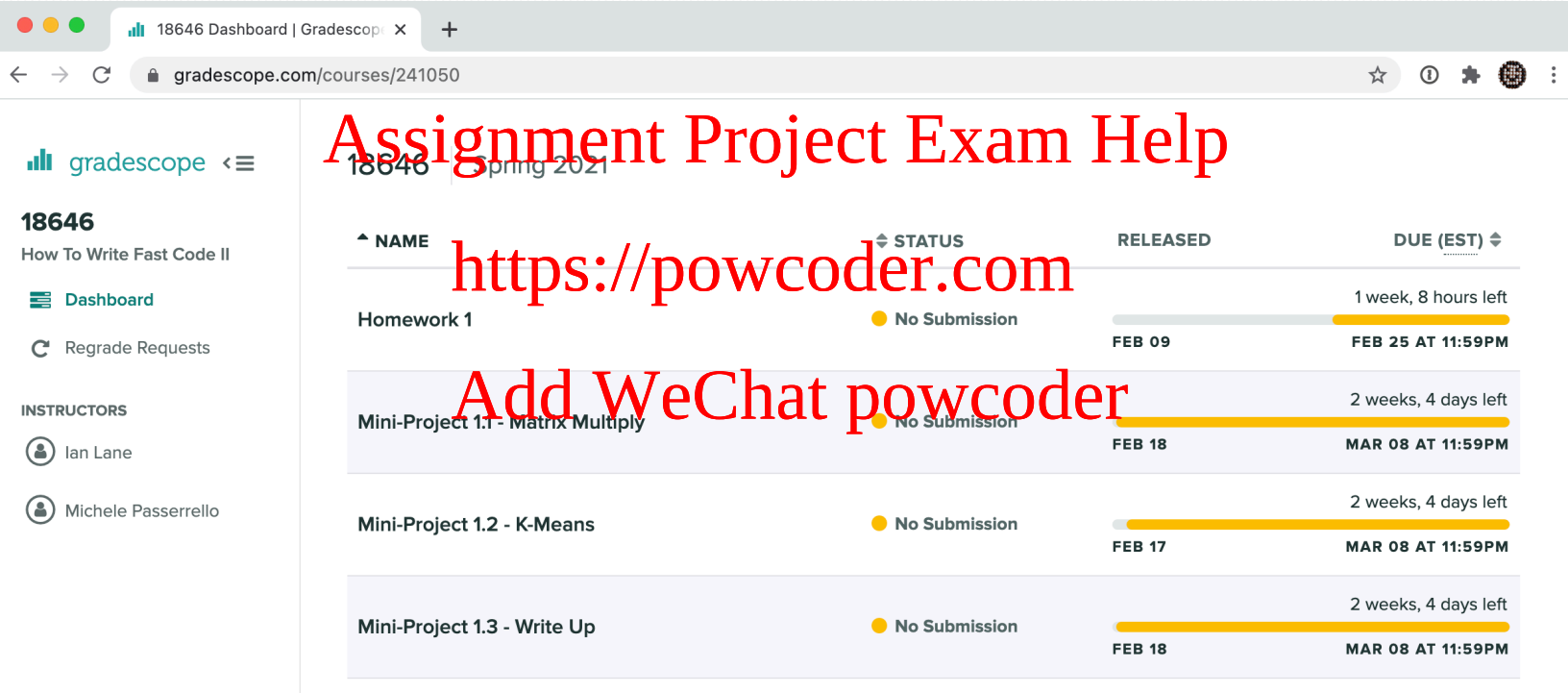
Add WeChat powcoder

Handout: https://cmu.box.com/s/uzfkghtkzfs0qc8acwwwzgdjep0lu9o4

Gradescope Links:

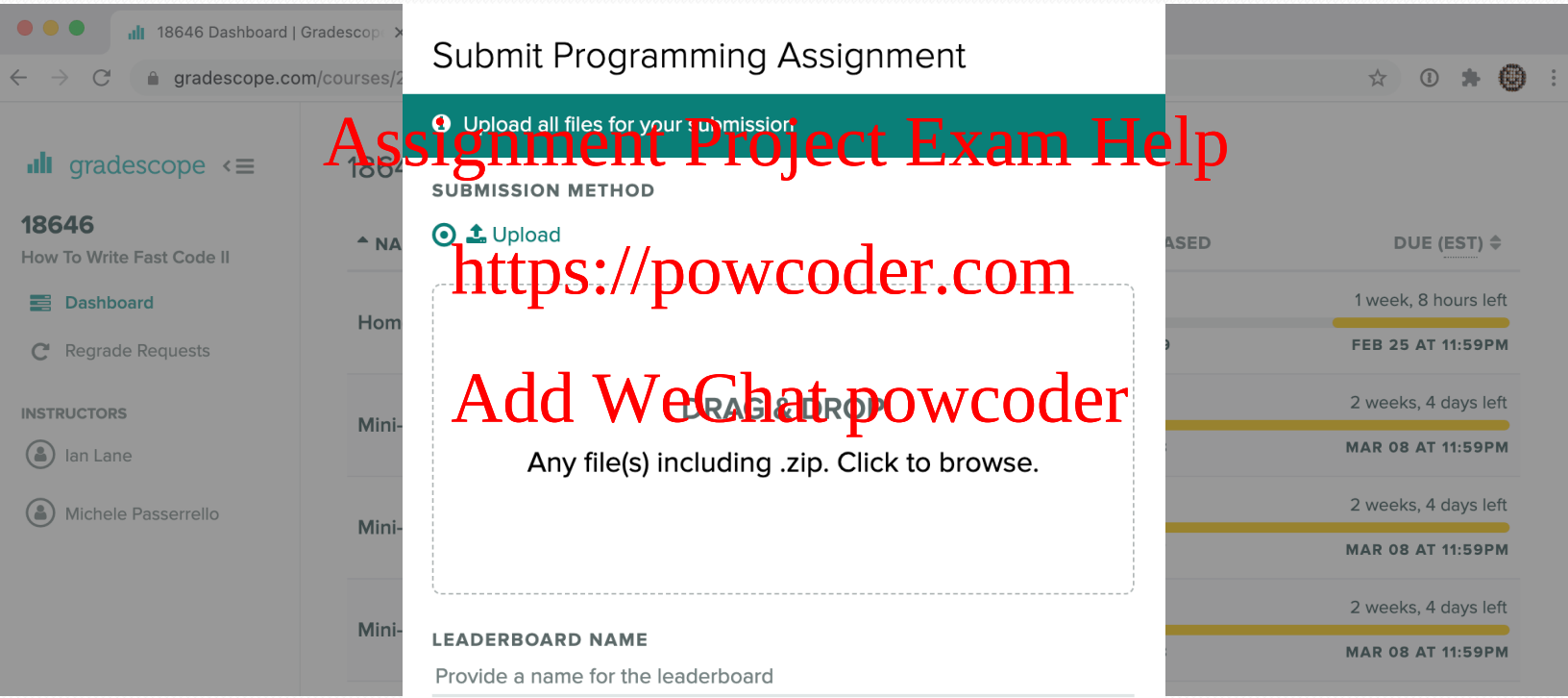
- **Matrix Multiply:** https://www.gradescope.com/courses/241050/assignments/998372
- **K-Means:** https://www.gradescope.com/courses/241050/assignments/1025865
- **Project Writeup:** https://www.gradescope.com/courses/241050/assignments/997551

Mini-Project 1: gradescope



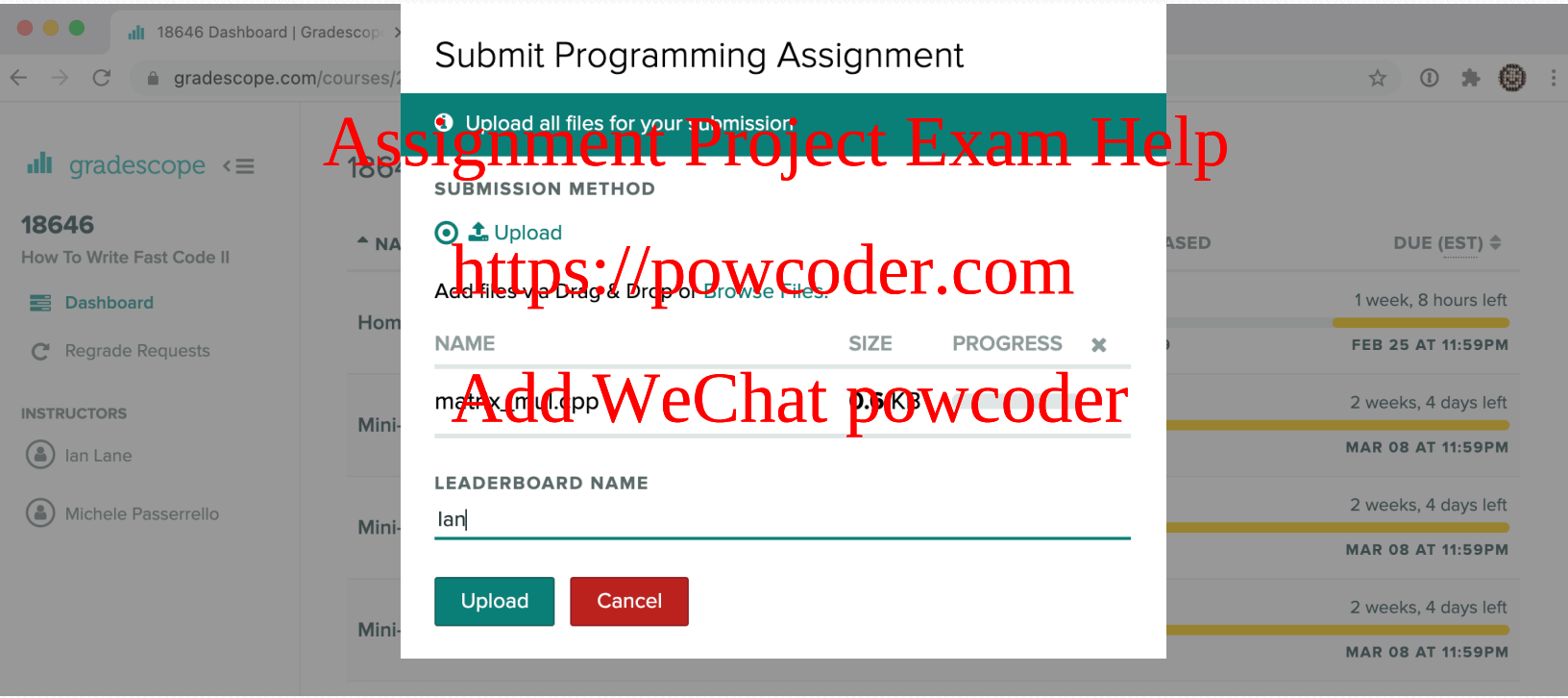
Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

Mini-Project 1: gradescope



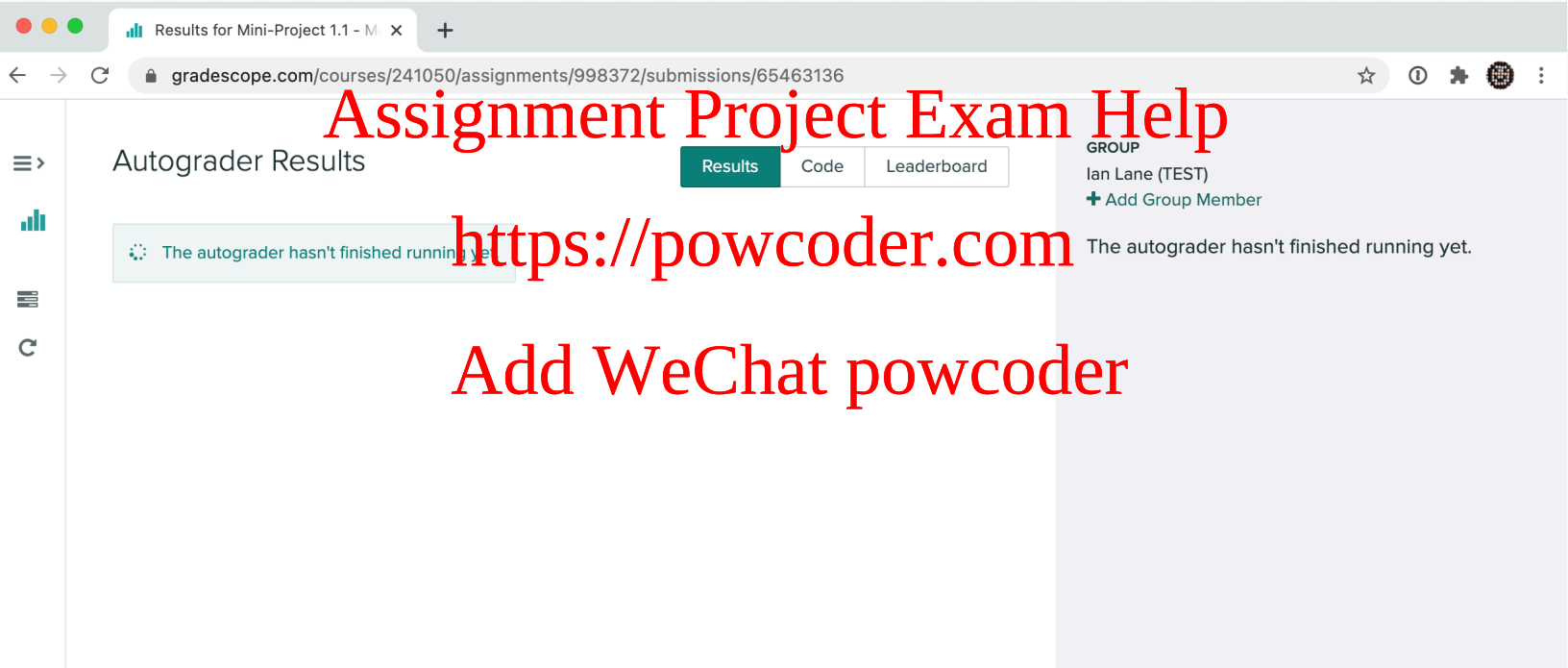
Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

Mini-Project 1: gradescope



Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

Mini-Project 1: gradescope



Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

Mini-Project 1: gradescope

Results for Mini-Project 1.1 - M

gradescope.com/courses/241050/assignments/998372/submissions/65463136?view=results

Assignment Project Exam Help

Autograder Results

Results | Code | Leaderboard

GROUP
Ian Lane (TEST)
+ Add Group Member

AUTOGRADER SCORE
-1.0 / 100.0

https://powcoder.com

Autograder Output

```
Getting submission...
Compiling matrix_mul.cpp ...
rm -f matrix_mul tests.o matrix_mul.o *.xml
g++-4.8 -fopenmp -c -g -Wall -Wno-unused-result -I/opt/local/include -I/autograder/sourc
g++-4.8 -fopenmp -c -g -Wall -Wno-unused-result -I/opt/local/include -I/autograder/sourc
g++-4.8 -fopenmp -L/opt/local/lib -L/autograder/source/cpunit/lib tests.o matrix_mul.o
Benchmarking matrix_mul.cpp ...
Done
```

Add WeChat powcoder

Total

```
MIN=14226.06 MAX=16942.15, AVG=14885.12, MED=14453.95
```

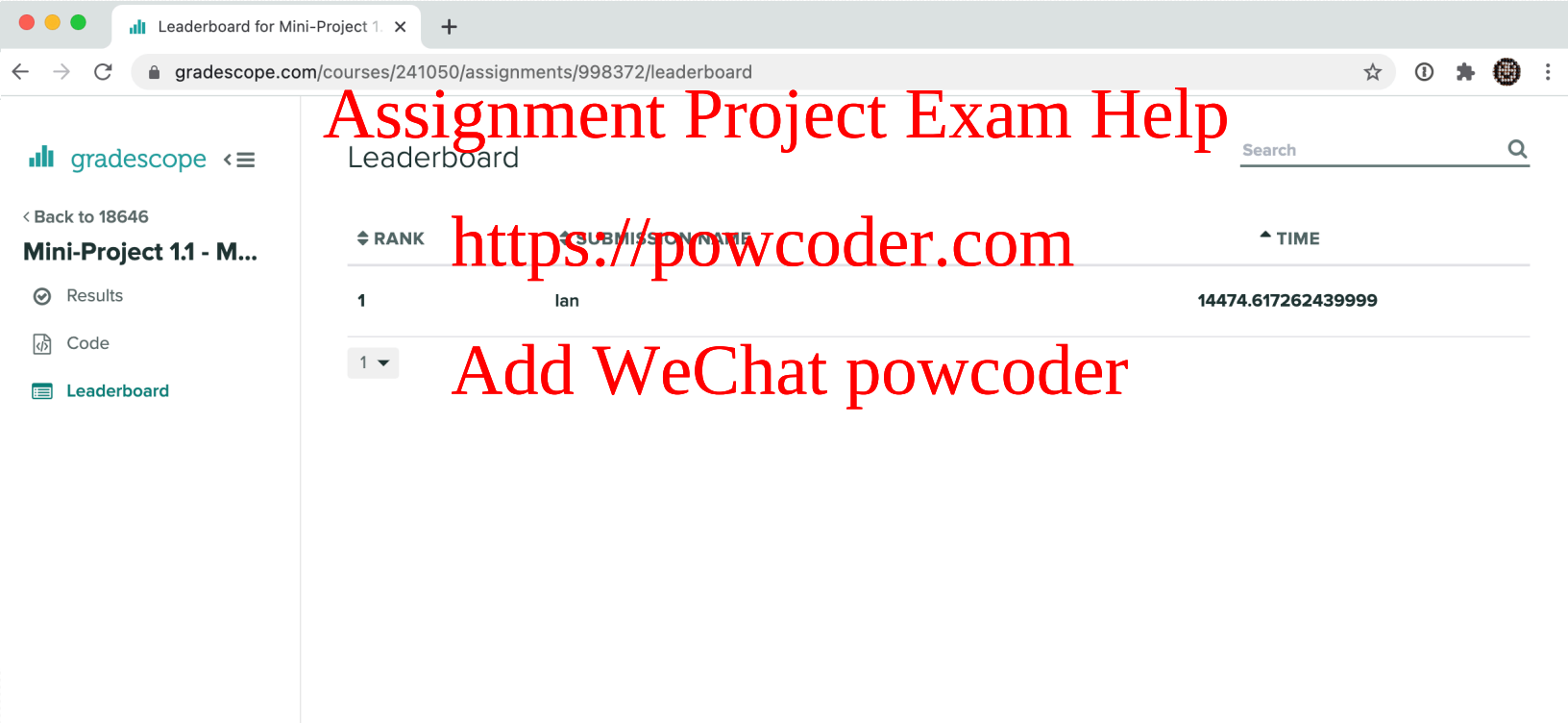
Testcase 1

```
MIN=0000.0391 MAX=0013.3801, AVG=0002.7969, MED=0000.0481
```

Testcase 2

```
MIN=0000.0039 MAX=0025.2930, AVG=0003.5956, MED=0000.0110
```

Mini-Project 1: gradescope



Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

Mini-Project 1: Grading Criteria

- 30% - Correctness - Correctness of the results (program output)
- 30% - Performance
  - `matrix_mul/omp/matrix_mul.cpp` (Matrix Multiply):
    Achieve **at least a 2x** speed up compared to the current **OpenMP version** (SUM of all the testcases on `matrix_mul_m3.wat`)
  - `kmeans/omp_kmeans.c` (K-means):
    Achieve **at least a 5x** speed up compared to the current **OpenMP version** (SUM of all tests)
- 30% - Write up - For each performance optimization explored, describe clearly:
  - How the speed up works
  - What is the expected speed up
  - What is the observed speed up
  - An explanation of any difference between the expected and observed speed ups
- 10% - Code quality - Good coding practices and well commented code

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

Mini-Project 1: Write-Up

Minimum of one 8.5x11 page write-up for each optimization. The write up should include:

- Optimization goal:
  - Hardware resources being optimized toward? (cache? SIMD? multicore?)
  - What is the specification of the hardware you are optimizing for?
- Optimization process:
  - Data considerations
  - Parallelization considerations
- Optimization results:
  - Performance before optimization
  - Performance after optimization

The three teams with the fastest implementations will present the techniques they attempted in a 10-minute presentation during the project review session.

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

Mini-Project 1 - Setup

- **Matrix Multiple**

```
$ cd ~/18646_MP1/matrix_mul/omp
$ make
$ ./matrix_mul -i ../matrix_mul_03.dat -o
                ...
Test Case 10    1111.42 milliseconds
```

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

- **K-Means**

```
$ cd ~/18646_MP1/kmeans
$ make
$ time ./omp_main -i ~/18646_MP1/data/kmeans03.dat -n 184
Writing coordinates of K=184 cluster centers…
Writing membership of N=191681 data objects…
115.659u 0.100s 0:15.57 743.4%
```

Mini-Project 1 – Makefile

- **The only two files you will submit are "`matrix_mul.cpp`" and "`omp_kmeans.c`"**

Assignment Project Exam Help

- **However!!!**
  - **For your <u>REPORT</u> you can explore the use of different flags during compilation and report on their effect!**
  - `-mavx` (code changes may be required)
  - `-O3`

https://powcoder.com
Add WeChat powcoder

If you make changes to the **`Makefile`** for your fastest performing version, please include the code in your project report!!!

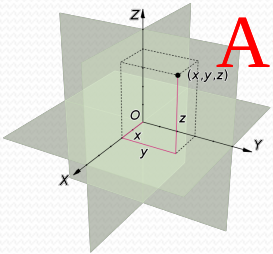
Outline

- CPU Achievable Peak Performance
- Matrix-Multiplication Discussion
- Overview of Mini-Project 1
- **A Quick Review:**
  - **SIMD Parallelism**
  - **OpenMP Pragmas**
  - **CPU Memory Hierarchy**
  - **False Sharing**
- Mini-Project 1.1 - Matrix Multiply
- Mini-Project 1.2 - K-Means

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

Exploiting SIMD Parallelism

- Applied to finding distance between two points:



Assignment Project Exam Help

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}$$

https://powcoder.com

Add WeChat powcoder

```
V1 = _MM_LOAD(&X1[0])
V2 = _MM_LOAD(&Y1[0])
V1 = _MM_SUB(V2, V1)
V1 = _MM_MUL(V1, V1)
_MM_STORE(&res[0], V1)
add r1, &res[0], &res[1]
add r1, r1, &res[2]
sqrt r1, r1
st d, r1
```

```
ld r1, x1
ld r2, x2
ld r3, x3
ld r4, y1
ld r5, y2
ld r6, y3
sub r1, r4, r1
sub r2, r5, r2
sub r3, r6, r3
mul r1, r1, r1
mul r2, r2, r2
mul r3, r3, r3
add r1, r1, r2
add r1, r1, r3
sqrt r1, r1
st d, r1
```

SIMD Parallelism

```
#include <stdio.h>
#include <pmmintrin.h>  //  header for SSE3
#define k 32

int main(){
  float x[k]; float y[k];      // vectors of length k
  __m128 X, Y;                 // 128-bit values
  __m128 acc = _mm_setzero_ps(); // set to (0, 0, 0, 0)
  float inner_prod, temp[4];
  int i, j;

  for (j=0; j<k; j++){ x[j] = 1.0; y[j] = 1.0;}

  for(i = 0; i < k - 4; i += 4) {
    X = _mm_load_ps(&x[i]);    // load chunk of 4 floats
    Y = _mm_load_ps(y + i);    // alternate way, pointer arithmetic
    acc = _mm_add_ps(acc, _mm_mul_ps(X, Y));
  }
  _mm_store_ps(&temp[0], acc);   // store acc into an array of floats
  inner_prod = temp[0] + temp[1] + temp[2] + temp[3];

  // add the remaining values
  for(; i < k; i++)
    inner_prod += x[i] * y[i];

  printf("Inner product is: %f\n", inner_prod);

  return 0;
}
```

SSE Version	gcc header
SSE	xmmintrin.h
SSE2	emmintrin.h
SSE3	Pmmintrin.h

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

Common SSE Intrinsics

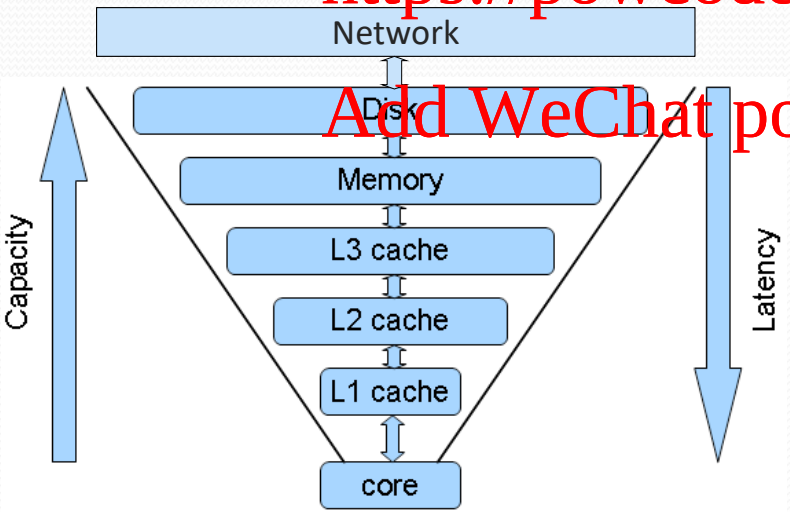
```
__m128 _mm_setzero_ps(void)// set to (0, 0, 0, 0)

__m128 _mm_set_ps1(float f)// set to (f, f, f, f)

__m128 _mm_set_ps(float w, float x, float y , float z) // set to (z,y,x,w)

__m128 _mm_setr_ps(float w, float x, float y , float z) // set to (w,x,y,z)

__m128 _mm_add_ps(__m128, __m128)

__m128 _mm_sub_ps(__m128, __m128)

__m128 _mm_mul_ps(__m128, __m128)

__m128 _mm_load_ps(float *base_addr) // load 4 floats from base_addr

__m128 _mm_loadr_ps(float *base_addr) // load in reverse order void

_mm_store_ps(float *addr, __m128 val) // write val into location addr
```

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

CPU Memory Hierarchy

- **Cache**:
  - A piece of fast memory close to the compute modules in a microprocessor
  - Allows faster access to a limited amount of data
  - Physical properties of wires and transistors determines trade-off between **cache capacity** and **access throughput/latency**

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder



(Capacity), (throughput), (latency)

Exa-Bytes, 0.1 GB/s, ~1-10B cycles

3TB, 6 GB/s, ~1M cycles

16GB, 34.08 GB/s, ~200 cycles

8 MB, 108.8 GB/s, 26-31 cycles

256 KB, 108.8 GB/s, 12 cycles

32 KB Data Cache, 163.2 GB/s, 4-6 cycles

Intel Core2 – 2600k @ 3.4GHz
(viewed from one core)

Working with a Memory Hierarchy

- **Principle of Locality**
  - The phenomenon of the same value or related storage locations being frequently accessed, usually amenable to performance optimization

Assignment Project Exam Help

- **Temporal Locality**
  - Reuse of specific data and/or resources within relatively small time durations

https://powcoder.com

Add WeChat powcoder

$$A \quad B$$

$$AB = \begin{pmatrix} -16 & -4 & -29 & 37 & -46 \\ 10 & 18 & 21 & -17 & 22 \\ 6 & -22 & -2 & -10 & 12 \\ -13 & -1 & -7 & 6 & -9 \end{pmatrix}$$

$$(-1)(3) + (-4)(4) + (-2)(-1)$$

$$-3 + -16 + 2$$

$$-17$$

- **Spatial Locality**
  - Use of data elements within relatively close storage locations

OpenMP Pragmas

- **OpenMP Pragmas**
  - for
  - atomic
  - reduction
  - private
  - shared
  - num_threads

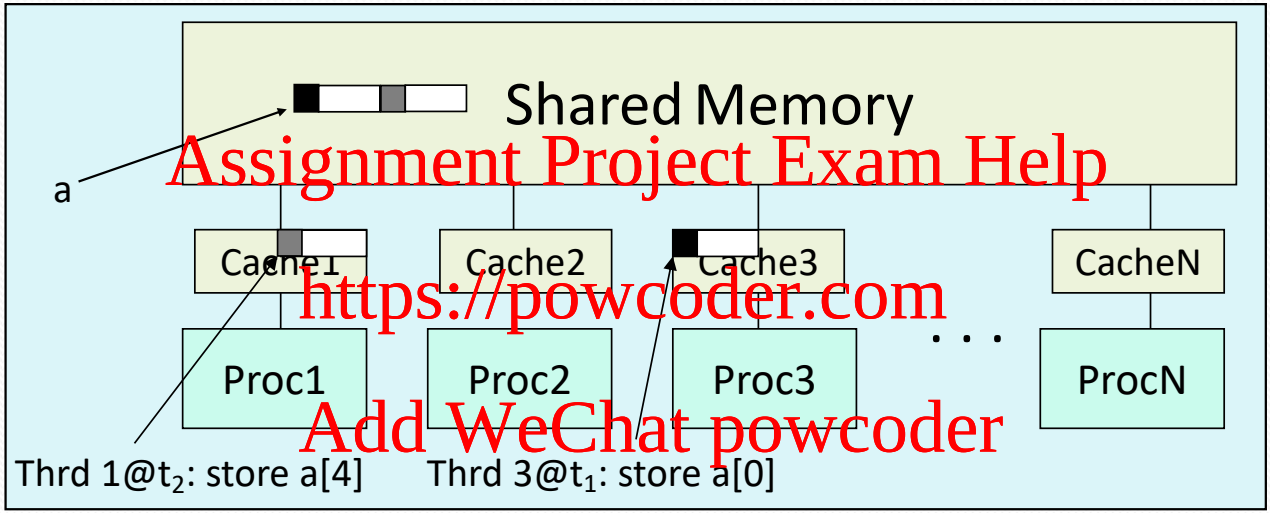
Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

- **OpenMP Scheduling**
  - schedule(static, dynamic,)

False Sharing



Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

- Be aware of the cache line sizes for a platform
- Avoid accessing the same cache line from different threads

Outline

- CPU Achievable Peak Performance
- Matrix-Multiplication Discussion
- Overview of Mini-Project 1
- A Quick Review:
  - SIMD Parallelism
  - OpenMP Pragmas
  - CPU Memory Hierarchy
  - False Sharing
- **Mini-Project 1.1 - Matrix Multiply**
- Mini-Project 1.2 - K-Means

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

Matrix Multiply – Problem Criteria

- 30% - Correctness - Correctness of the results (program output)
- 30% - Performance
  - `matrix_mul/omp/matrix_mul.cpp` (Matrix Multiply): Achieve **at least a 2x** speed up compared to the current **OpenMP version** (SUM of all the testcases on `matrix_mul_03.dat`)
- 30% - Write up - For each performance optimization explored, describe clearly:
  - How the speed up works
  - What is the expected speed up
  - What is the observed speed up
  - An explanation of any difference between the expected and observed speed ups
- 10% - Code quality - Good coding practices and well commented code

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

Matrix Multiply: OpenMP

```
void
matrix_transpose_copy(float *matrix_dst, float *matrix_src, unsigned int dim)
{
#pragma omp parallel for
    for (unsigned int i = 0; i < dim; i++) {
      for (unsigned int j = 0; j < dim; j++) {
        matrix_dst[ELEM(i,j,dim)] = matrix_src[ELEM(j,i,dim)];
      }
    }
}
```

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

Matrix Multiply: SIMD operations

```
__m128 vector_1, vector_2/*, vector_result*/;
for( unsigned int i = 0; i < sq_dimension; i++ ) {
  for( unsigned int j = 0; j < sq_dimension; j++ ) {
    //vector_result = _mm_setzero_ps();
    sq_matrix_result[i*sq_dimension + j] = 0;
    for( unsigned int k = 0; k < sq_dimension; k+=4 ) {
      vector_1 = _mm_load_ps(sq_matrix_1_8 + (i*sq_dimension_8 + k) );
      vector_2 = _mm_load_ps(sq_matrix_2t_8 + (j*sq_dimension_8 + k) );
      vector_1 = _mm_dp_ps(vector_1, vector_2, 0xf1);
      sq_matrix_result[i*sq_dimension + j] += _mm_cvtss_f32(vector_1);
    }
  }
}
```

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

- Could also:
  - Loop unrolling of inner loop to reduce overhead
  - Resize and pad original matrix to simplify code

Matrix Multiply: Cache blocking

```
#pragma omp parallel for private(i,j,k,i1,j1,k1)
for (i = 0; i < sq_dimension; i += B) {
  for (j = 0; j < sq_dimension; j += B) {
    for (k = 0; k < sq_dimension; k += B ) {
      for (i1 = i; i1 < i + B && i1 < sq_dimension; i1++) {
        mult_i = i1 * sq_dimension;
        for (j1 = j; j1 < j + B && j1 < sq_dimension; j1++) {
          for (k1 = k; k1 < k + B && k1 < sq_dimension; k1++) {
            sq_matrix_result[mult_i + j1] += sq_matrix_1[mult_i + k1] *
sq_matrix_2[k1*sq_dimension + j1];
}}}}}}
```

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

- Could also:
  - Unroll inner loops more to decrease overhead

Matrix Multiply: Transpose B

(probably don't need this for project 1)

```
void
matrix_transpose_copy(float *matrix_dst, float *matrix_src, unsigned int dim)
{
#pragma omp parallel for
    for (unsigned int i = 0; i < dim; i++) {
      for (unsigned int j = 0; j < dim; j++) {
        matrix_dst[ELEM(i,j,dim)] = matrix_src[ELEM(j,i,dim)];
      }
    }
}
```

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

- Could also:
  - Use Cache Blocking for Transpose
  - Use SSE for Transpose

Outline

- CPU Achievable Peak Performance
- Matrix-Multiplication Discussion
- Overview of Mini-Project 1
- A Quick Review:
  - SIMD Parallelism
  - OpenMP Pragmas
  - CPU Memory Hierarchy
  - False Sharing
- Mini-Project 1.1 - Matrix Multiply
- **Mini-Project 1.2 - K-Means**

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

K-Means – Problem Criteria

- 30% - Correctness - Correctness of the results (program output)
- 30% - Performance
  - `kmeans/omp_kmeans.c` (K-means):
    Achieve **at least a 5x** speed up compared to the current **OpenMP version** (SUM of all tests)
- 30% - Write up - For each performance optimization explored, describe clearly:
  - How the speed up works
  - What is the expected speed up
  - What is the observed speed up
  - An explanation of any difference between the expected and observed speed ups
- 10% - Code quality - Good coding practices and well commented code

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

K-means: The Phases

1. **Initialization:** Randomly select *k* cluster centers
   - Select *k* samples from data as initial centers [Forgy Partition]
2. **Expectation:** Assign each data point go closest center
   - Compare each data point (*N*) to each cluster center (*k*)
   - Distance Metric: Euclidean distance (*D* dimensions)
3. **Maximization:** Update centers based on assignments
   - For each cluster (*k*) compute mean (D dimensions) from data points assigned to that cluster
4. **Evaluate:** Re-iterate steps 2-3 until convergence or stopping criteria met
   - Percentage of data points re-assigned
   - Number of iterations (2-3)

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

Example Code (Expectation)

```
....
    delta = 0.0;
    for (i=0; i<numObjs; i++) {
      /* find the array index of nestest cluster center */
      index = find_nearest_cluster(numClusters, numCoords, objects[i], clusters);

          ...

      /* if membership changes, increase delta by 1 */
      if (membership[i] != index) delta += 1.0;

      /* assign the membership to object i */
      membership[i] = index;
...
```

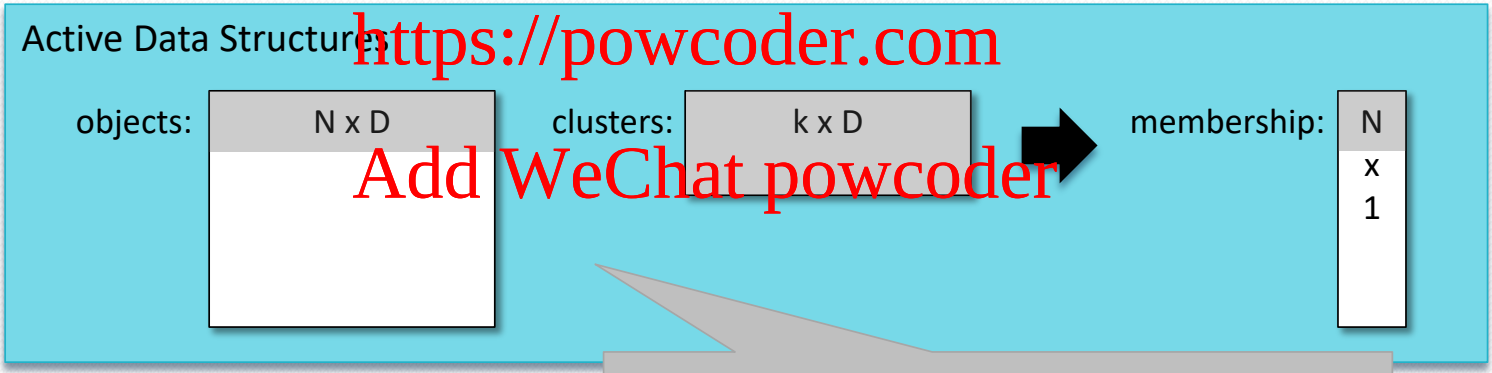
Evaluate distance to each cluster centroid and select closest

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

Example Code (Expectation)

```
....
    delta = 0.0;
    for (i=0; i<numObjs; i++) {
      /* find the array index of nestest cluster center */
      index = find_nearest_cluster(numClusters, numCoords, objects[i], clusters);
```

Assignment Project Exam Help



https://powcoder.com
Add WeChat powcoder

```
      membership[i] = index;
...
```

Possible Concurrencies:

N (independent)
D (sum reduction) ← euclid_dist_2()
k (min reduction)

Example Code (Maximization)

```
....
    /* update new cluster centers : sum of objects located within */
        newClusterSize[index]++;
        for (j=0; j<numCoords; j++)
            newClusters[index][j] += objects[i][j];
    }

    /* average the sum and replace old cluster centers with newClusters */
    for (i=0; i<numClusters; i++) {
        for (j=0; j<numCoords; j++) {
            if (newClusterSize[i] > 0)
                clusters[i][j] = newClusters[i][j] / newClusterSize[i];
            newClusters[i][j] = 0.0;   /* set back to 0 */
        }
        newClusterSize[i] = 0;       /* set back to 0 */
    }
...
```

prepare for next iteration

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

Example Code (Maximization)

```
....
    /* update new cluster centers : sum of objects located within */
        newClusterSize[index]++;
        for (j=0; j<numCoords; j++)
            newClusters[index][j] += objects[i][j];
```

Assignment Project Exam Help

Active Data Structures

https://powcoder.com

objects: N x D

membership: N x 1

NewClusters: k x D

Add WeChat powcoder

```
        }
        newClusterSize[i] = 0;      /* set bac
    }
...
```

Possible Concurrencies:

D (independent)

N (Histogram computation into k bins

K-Means: Parallelization

```
#pragma omp parallel for
  private(i, ???)
  firstprivate(numObjs, ???)
  shared(objects, ???)
  reduction(???)
```

Assignment Project Exam Help

- How can we avoid false sharing in OMP?

https://powcoder.com

- Could we introduce additional local variables in each thread?

Add WeChat powcoder

- Can we parallelize this further?

K-Means: Cost Reduction

```
for (i=0; i<numCluster; i++){
        float tmp = 1.0f / newClusterSize[i];
        for (j = 0; j < numCoords; j++)
            if (newClusterSize[i] > 1)
                clusters[i][j] = newClusters[i][j] * tmp;
}
```

Assignment Project Exam Help

https://powcoder.com

- Remove the dividing operation and shift it to outer loop

Add WeChat powcoder

K-Means: Other Ideas?

- Predict "stable" points, which means that if a point doesn't change its cluster for "m" rounds, we regard it as "stable" and skip the computation next round

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder