

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

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://eduassistpro.github.io/>
Add WeChat edu_assist_pro

Peak Single-precision Performance

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

limit in theory

[GFLOP]	Assignment	Project	Exam	Help
None, scalar	0 Gflop	2 cores	4 cores	8 cores
SSE	112	224	448	896
AVX	224	448	896	1792

Add WeChat `edu_assist_pro`

for intel sandyBridge : 2 ALU

Most CPU applications achieve a fraction of this

In addition ...

- Deep memory hierarchy

- Registers

- L1 cache

- L2 cache

- TLB (Translati

Assignment Project Exam Help

<https://eduassistpro.github.io/>

- Other forms of parallelism

- Pipelining

Add WeChat  edu_assist_pro

- Instruction-level parallelism (multiple functional units)

2 multfs sume the

- “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://eduassistpro.github.io/>
Add WeChat edu_assist_pro

Naïve Matrix Multiply

{implements $C = C + A * B$ }

for $i = 1$ to n

 for $j = 1$ to n

 for $k = 1$ to n

Assignment Project Exam Help



<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

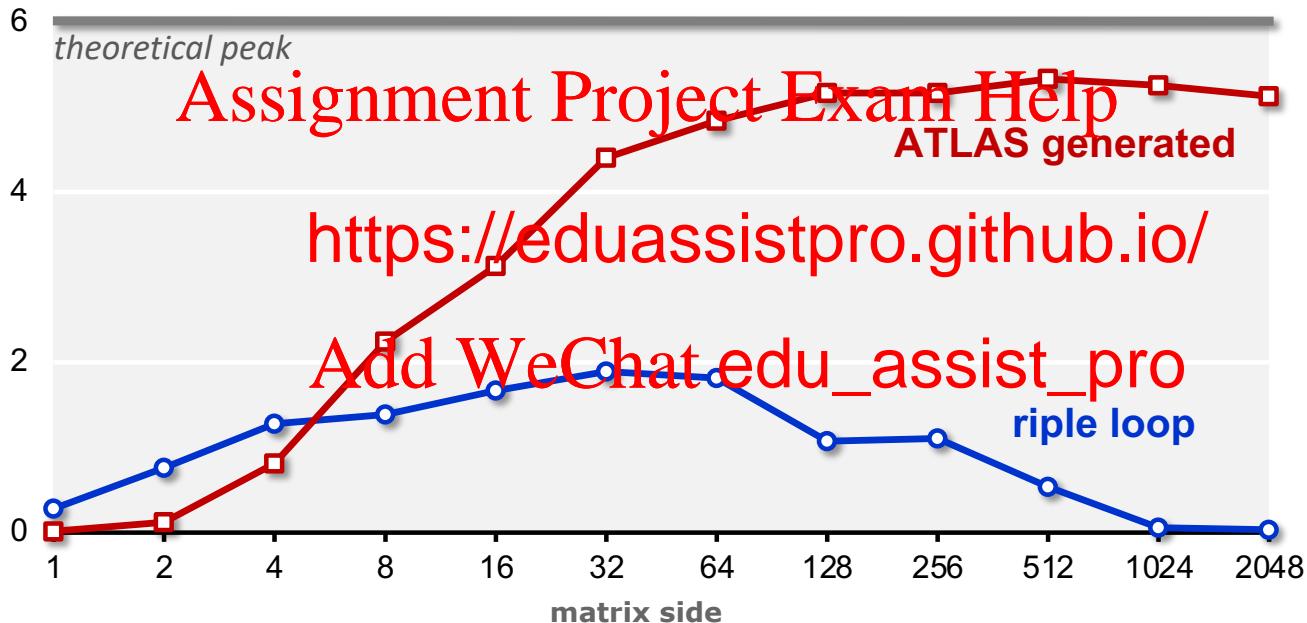
$$C(i,j) = C(i,j) + A(i,:) * B(:,j)$$

The diagram illustrates the computation of a single element $C(i,j)$. On the left, a large white square represents the matrix C , with a blue square in the (i,j) position. An equals sign follows. To the right of the equals sign is another white square representing C , also with a blue square in the (i,j) position. This is followed by a plus sign. To the right of the plus sign is a white square representing the row $A(i,:)$, with a thick blue horizontal bar highlighting the j -th column. To the right of this is a multiplication symbol (*). Finally, to the right of the multiplication symbol is a white square representing the column $B(:,j)$, with a thick blue vertical bar highlighting the j -th row.

Matrix-Matrix Multiplication Speed

MMM (square real double) Core 2 Duo 3Ghz

performance [Gflop/s]



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

Note on Matrix Storage

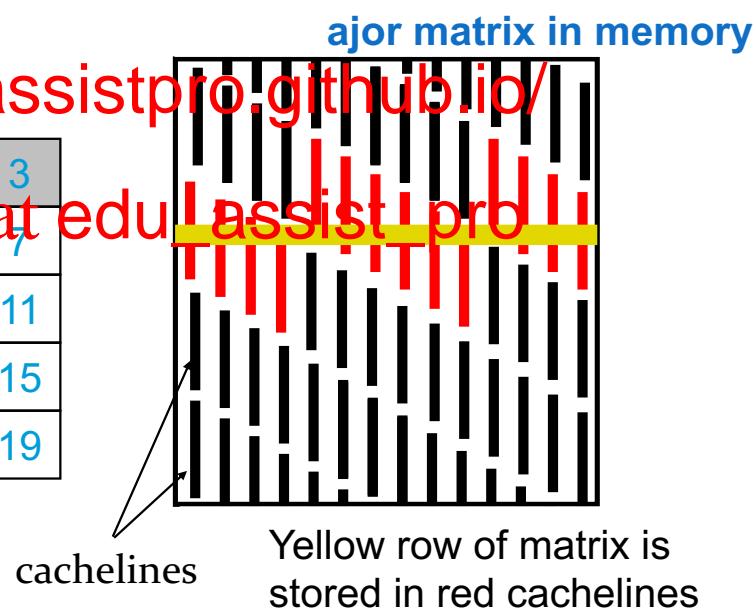
- A matrix is a 2-D array of elements, but memory addresses are “1-D”
- Conventions for matrix layout
 - by column, or “column major” (Fortran default);
 $A[i, j]$ at $i + j \cdot n$
 - by row, or “row major” (C default)
 $A[i, j]$ at $i * n + j$

Assignment Project Exam Help

Column major

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

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



- 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}

$A(i, :)$

for $j = 1$ to m

{read $C(i,j)$ into f

{read column j of B into g

loads

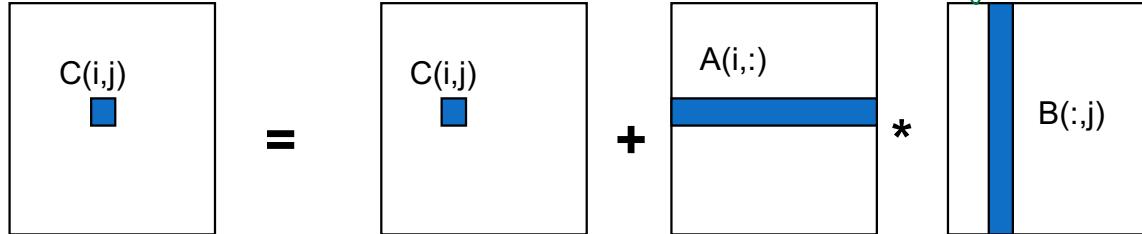
<https://eduassistpro.github.io/>

for $k = 1$ to n

$C(i,j) = C(i,j) + A(i,k) * B(k,j)$

Add WeChat edu_assist_pro

{write $C(i,j)$ back to slow memory}



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

$$= n^3 + 3n^2$$

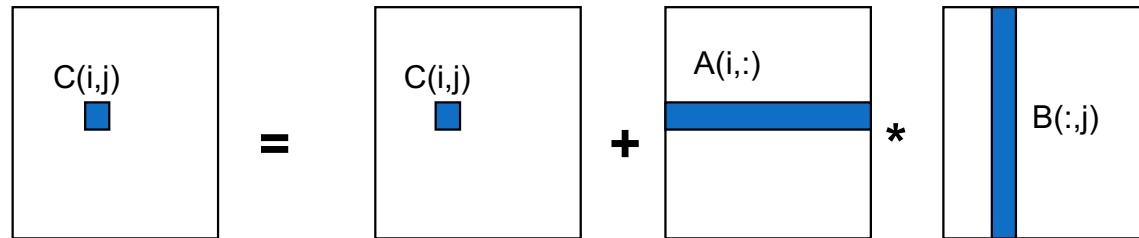
So $q = f / m = 2n^3 / (n^3)$

≈ 2 for large n , n

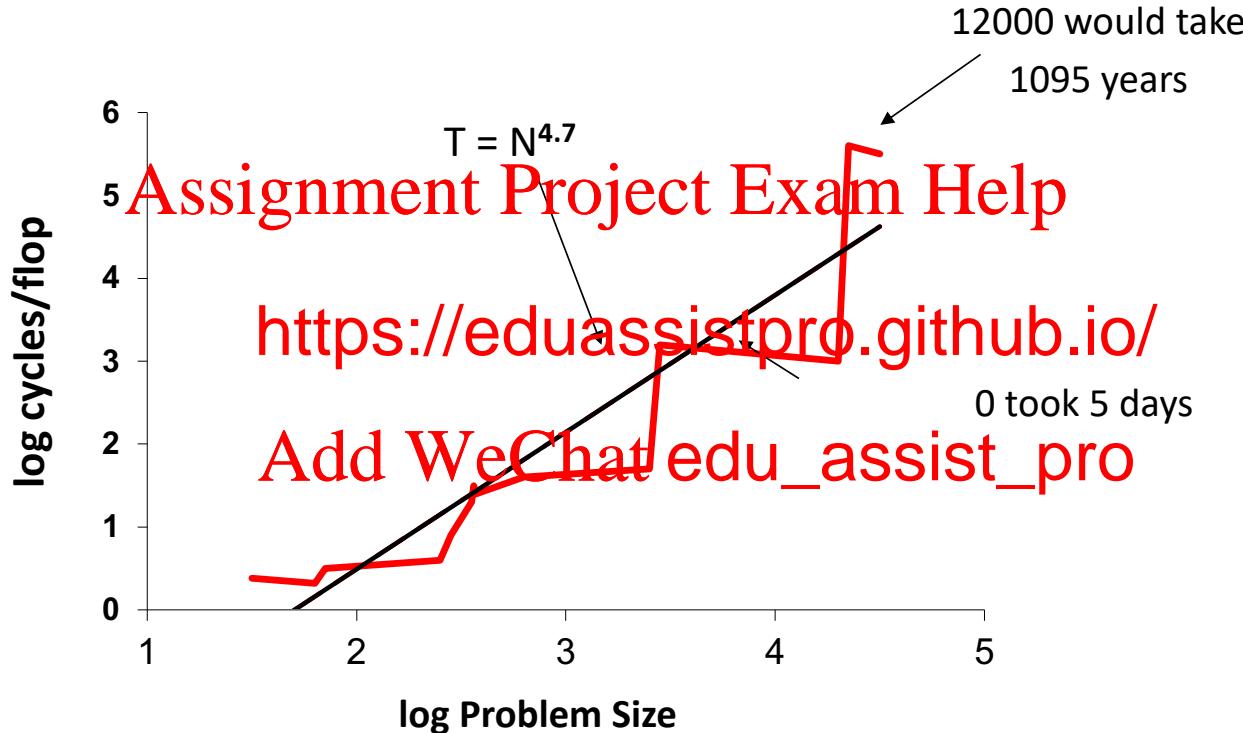
Assignment Project Exam Help

<https://eduassistpro.github.io/>

Inner two loops are just matrix-vector multiplication times B
Similar for any other order of 3 loops

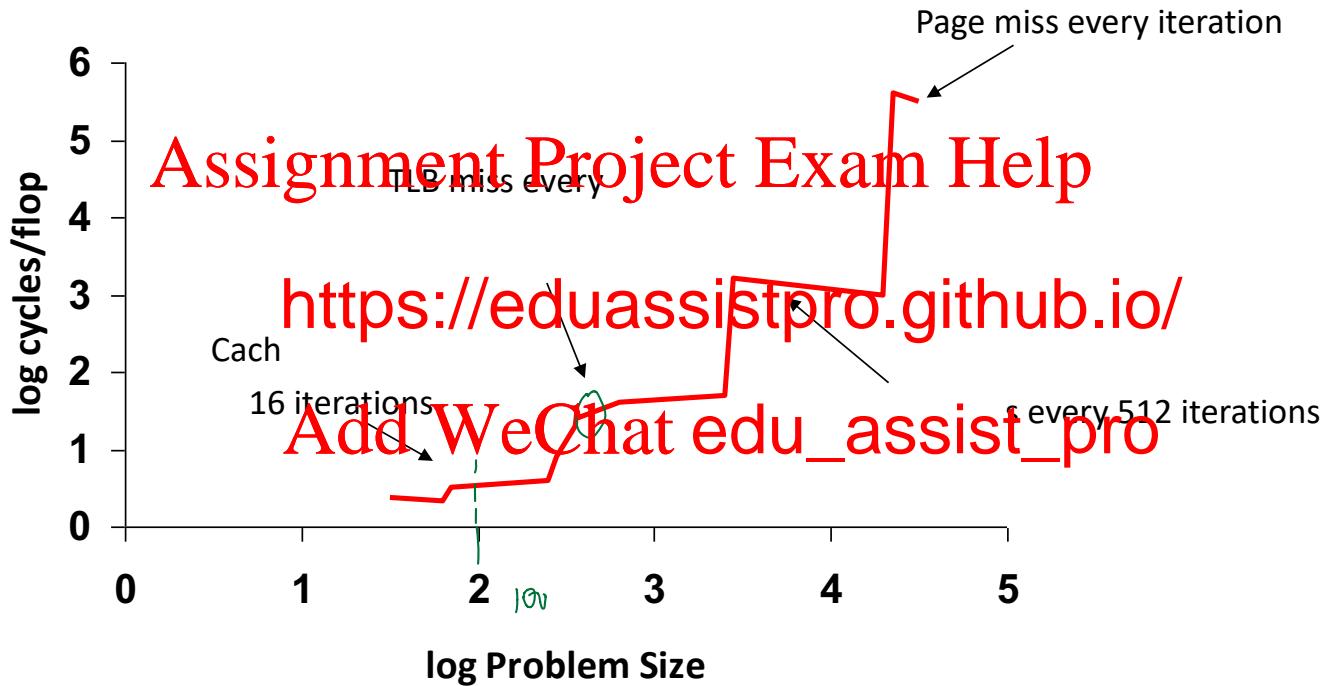


Naïve Matrix Multiply on RS/6000



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

Naïve Matrix Multiply on RS/6000



Blocked (Tiled) Matrix Multiply

$A, B, C = N \times N$ matrices of $b \times b$ 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

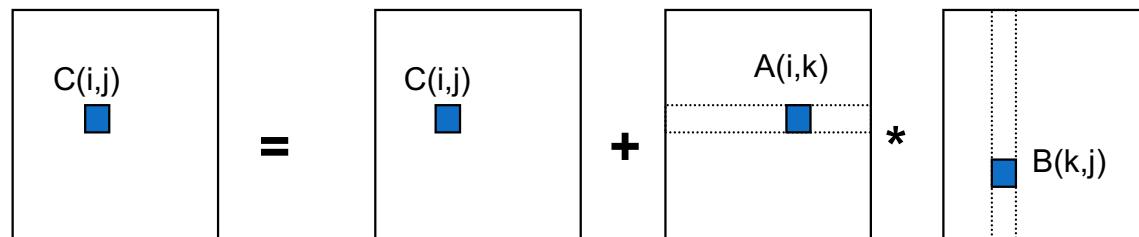
{read block $A(i,k)$ into fast mem
<https://eduassistpro.github.io/>

{read block $B(k,j)$ into fast mem}

$C(i,j) = C(i,j) + A(i,k) * B(k,j)$ [in blocks]

{write block $C(i,j)$ back to slow mem}

blocks should
be smaller
than cache
X
smallest



Further Matrix-Multiply Optimizations in Real-world Libraries

- Block size adaptation for appropriate caches

Assignment Project Exam Help

- Register-level b

<https://eduassistpro.github.io/>

- Copy optimization (data layout)

Add WeChat edu_assist_pro

- Optimizing the mini-matrix-multiply (

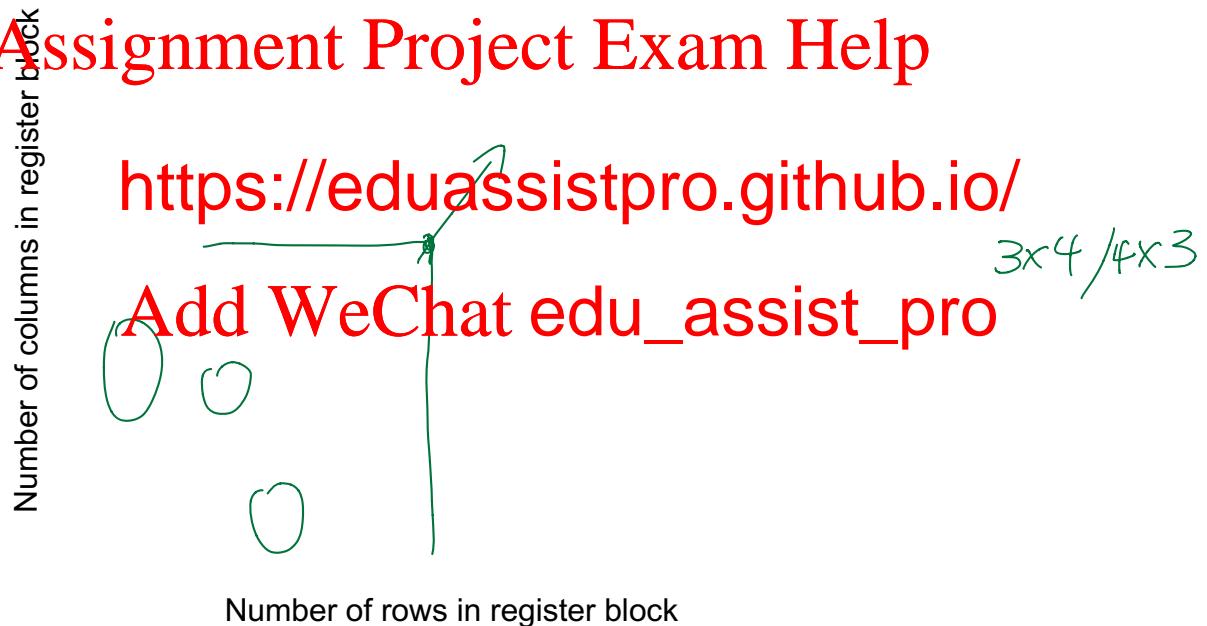
- Multi-level blocking

- Multi-level copying

Register-level blocking

%

Assignment Project Exam Help



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: u

f users willing)

<https://eduassistpro.github.io/>

Original matrix

(numbers are addresses)



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

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

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://eduassistpro.github.io/>
Add WeChat edu_assist_pro

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 app

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

Add WeChat edu_assist_pro

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://eduassistpro.github.io/>

Add WeChat `edu_assist_pro`

Handout: <https://cmu.box.com/s/uzfkghtkzfs0> lu9o4

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://eduassistpro.github.io/>

Add WeChat edu_assist_pro

Mini-Project 1: gradescope

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

Mini-Project 1: gradescope

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

Mini-Project 1: gradescope

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

Mini-Project 1: gradescope

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

Mini-Project 1: gradescope

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

Mini-Project 1: Grading Criteria

Assignment Project Exam Help

 <https://eduassistpro.github.io/>

 Add WeChat edu_assist_pro



Mini-Project 1: Write-Up

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

Mini-Project 1 - Setup

- **Matrix Multiple**

```
$ cd ~/18646_MP1/matrix_mul/omp
```

```
$ make
```

```
$ ./matrix_mul -i ../matrix_mul_03.dat -o ..
```

Test Case 10 11 **Assignment Project Exam Help**
https://eduassistpro.github.io/

- **K-Means**

Add WeChat edu_assist_pro

```
$ 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 REP <https://eduassistpro.github.io/> compilation add the following flags:
 - mavx (code changes may be required)
 - O3

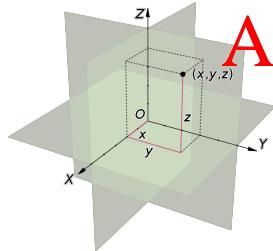
If you make changes to the **Makefile** for your fastest performing version, please include the code in your project report!!!

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://eduassistpro.github.io/>
Add WeChat edu_assist_pro

Exploiting SIMD Parallelism

- Applied to finding distance between two points:



Assignment Project Exam Help

<https://eduassistpro.github.io/>

```
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
```



Add WeChat edu_assist_pro

```
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 <pmmmintrin.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
    int i, j;

    for (j=0; j<k; j++) { x

        for(i = 0; i < k - 4; i += 4) {
            X = _mm_load_ps(&x[i]); // load chunk of 4
            Y = _mm_load_ps(y + i); // alternate way, p
            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;
}
```

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

SSE Version	gcc header
SSE	xmmmintrin.h
SSE2	emmmintrin.h
SSE3	Pmmmintrin.h

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(                                     ) // set to (w,x,y,z)
__m128 __mm_add_ps(_                                     https://eduassistpro.github.io/
__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://eduassistpro.github.io/>

Add WeChat edu_assist_pro

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
- Assignment Project Exam Help**
- <https://eduassistpro.github.io/>
- 16 GB, 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
- **Temporal Locality**
 - Reuse of specific and/or resources within relatively small time durations
- **Spatial Locality**
 - Use of data elements within relatively close storage locations

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

OpenMP Pragmas

- **OpenMP Pragmas**

- for
- atomic
- reduction
- private
- shared
- num_threads

Assignment Project Exam Help

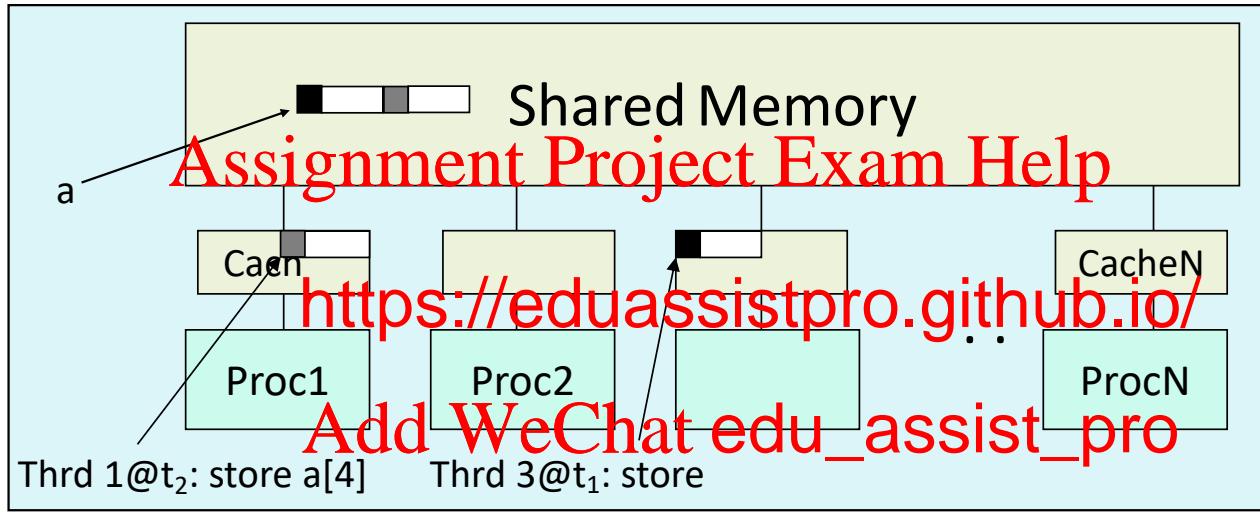
<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

- **OpenMP Scheduling**

- schedule(static, dynamic,)

False Sharing



- 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://eduassistpro.github.io/>
Add WeChat edu_assist_pro

Matrix Multiply – Problem Criteria

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

Matrix Multiply: OpenMP

```
32 void
33 matrix_transpose_copy(float *matrix_dst, float *matrix_src, unsigned int dim)
34 {
35 #pragma omp parallel for
36     for (unsigned int i = 0; i < dim; i++) {
37         for (unsig
38             matrix_d https://eduassistpro.github.io/
39         }
40     }
41 }
```

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

Matrix Multiply: SIMD operations

```
65     __m128 vector_1, vector_2/*, vector_result*/;
66     for( unsigned int i = 0; i < sq_dimension; i++ ) {
67         for( unsigned int j = 0; j < sq_dimension; j++ ) {
68             //vector_result = mm_setzero_ps();
69             sq_matrix_result[i*sq_dimension + j] = 0;
70             for( unsigned int k = 0; k < sq_dimension; k++ ) {
71                 vector_1 = mm_dp_ps(vector_1,
72                                     vector_2,
73                                     vector_2);
74                 vector_1 = mm_dp_ps(vector_1,
75                                     vector_2);
76                 sq_matrix_result[i*sq_dimension + j] += _mm_ss_f32(vector_1);
77             }
78         }
79     }
```

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

Matrix Multiply: Cache blocking

```
48 #pragma omp parallel for private(i,j,k,i1,j1,k1)
49 for (i = 0; i < sq_dimension; i += B) {
50     for (j = 0; j < sq_dimension; j += B) {
51         for (k = 0; k < sq_dimension; k += B ) {
52             for (i1 =
53                 mult_i = https://eduassistpro.github.io/
54                 for (j1 = j; j1 < j + B && j1 < sq_dimension; j1++) {
55                     for (k1 = k; k1 < k + B && k1 < sq_dimension;
56                         sq_matrix_result[mult_i +
57                             _1[mult_i + k1] *
58 sq_matrix_2[k1*sq_dimension + j1];
59 }}}}}}
```

Assignment Project Exam Help

Add WeChat edu_assist_pro

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

Matrix Multiply: Transpose B

(probably don't
need this
for project 1)

```
32 void
33 matrix_transpose_copy(float *matrix_dst, float *matrix_src, unsigned int dim)
34 {
35 #pragma omp parallel for
36     for (unsigned int i = 0; i < dim; i++) {
37         for (unsi
38             matrix_ https://eduassistpro.github.io/
39         }
40     }
41 }
```

Assignment Project Exam Help

Add WeChat edu_assist_pro

- 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://eduassistpro.github.io/>
Add WeChat edu_assist_pro

K-Means – Problem Criteria

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

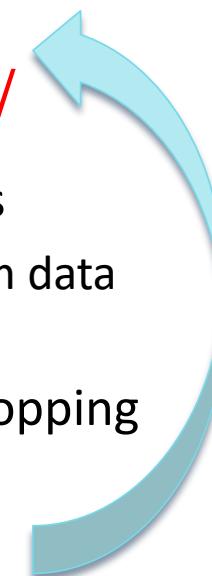
K-means: The Phases

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 e
 - Distance M
3. **Maximization:** Update centers
 - For each cluster (k) compute mean (m) of all 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://eduassistpro.github.io/>

Add WeChat edu_assist_pro



Example Code (Expectation)

```
....  
delta = 0.0;  
for (i=0; i<numObjs; i++) {  
    /* find the array index of nearest cluster center */  
    index = find_near  
                    [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;  
    ...
```

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Evaluate distance to
each cluster centroid
and select closest

Add WeChat edu_assist_pro

Example Code (Expectation)

```
....  
delta = 0.0;  
for (i=0; i<numObjs; i++) {  
    /* find the array index of the test cluster center */  
    index = find_near
```

[i], clusters);

Active Data Structure

<https://eduassistpro.github.io/>

objects:

N x D

clusters:



membership:

N
x
1

Add WeChat edu_assist_pro

membership[i] = index;

Possible Concurrency:

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[ind  
    }
```

<https://eduassistpro.github.io/>

```
/* average the sum and replace old cluster center */  
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

Example Code (Maximization)

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

Active Data Structur

<https://eduassistpro.github.io/>

objects:

N x D

membership

ters:

k x D

Add WeChat edu_assist_pro

}

newClusterSize[i] = 0; /* set bac

}

Possible Concurrency:

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 <https://eduassistpro.github.io/>
- Could we introduce additional ~~variables~~ in each thread?
- 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] > 0)  
            clusters[i][j] = newClusters[i][j] * tmp;  
}
```

Assignment Project Exam Help

<https://eduassistpro.github.io/>

- Remove the dividing operation a
~~outer loop~~
~~Add WeChat edu_assist_pro~~

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 **Assignment Project Exam Help**

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro