Assignment Project Exam Help

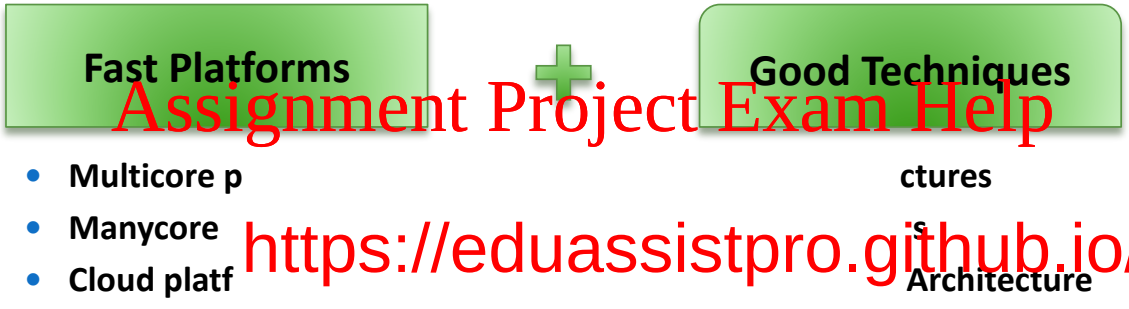
https://eduassistpro.github.io/

Add WeChat edu_assist_pro

Carnegie Mellon University

Ian Lane

What we discussed last time:



Assignment Project Exam Help
https://eduassistpro.github.io/
Add WeChat edu_assist_pro

- Highlighted the difference between mu core platforms
- Discussed the multicore and manycore platform intricacies

- **Exposing** concurrency in the k-means algorithm
- **Exploiting** parallelism by **exploring** mappings from application to platform

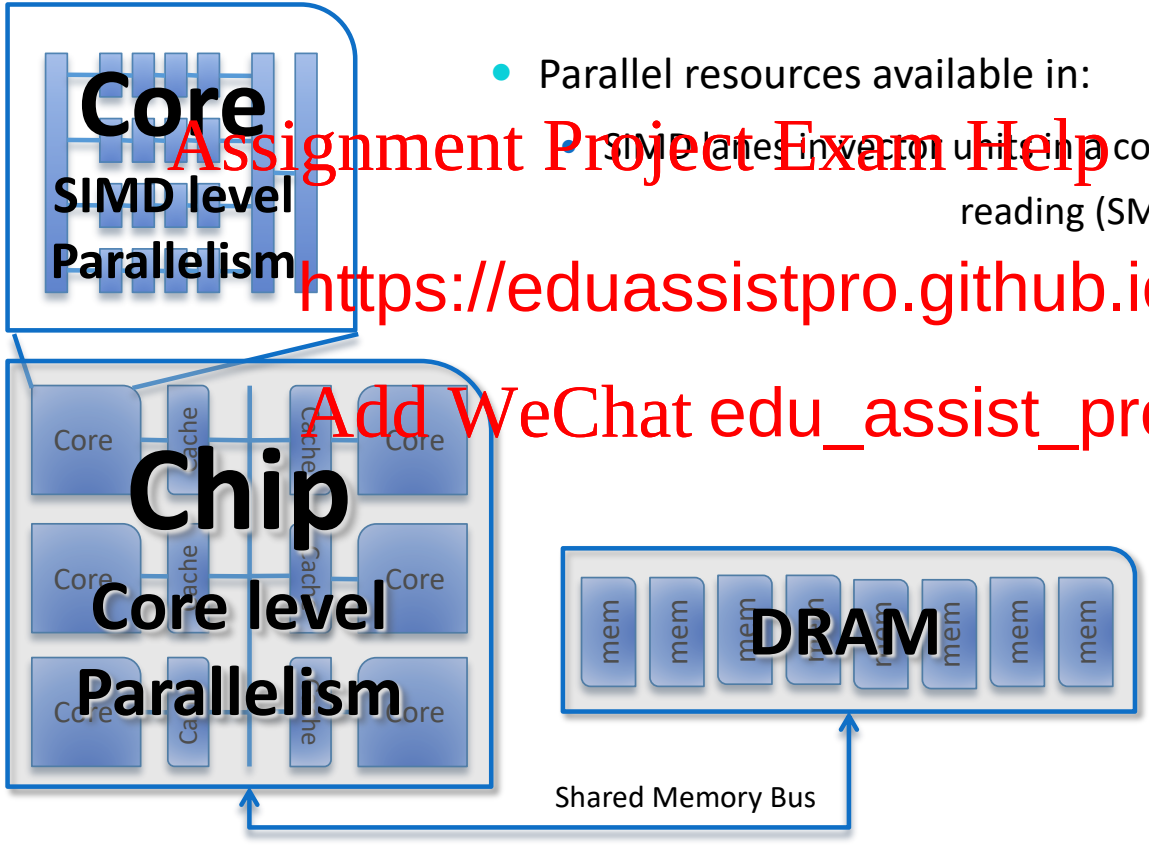
- This lecture: **An abstraction for multicore parallel programming**

Outline

- Multicore Shared-Memory Platforms
- OpenMP – API for Shared-Memory Multi-processing
  - Overview
  - Example: numerical integration
  - Shared variable
  - SPMD vs worksharing
  - Data environment options
  - Advanced worksharing options
  - Synchronization options
- How is This Relevant to Writing Fast Code?

Assignment Project Exam Help
https://eduassistpro.github.io/
Add WeChat edu_assist_pro

Multicore Shared-Memory Platforms



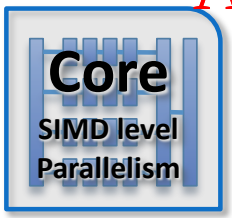
- Parallel resources available in:
  - SIMD lanes in vector units in a core
  - reading (SMT) in a core

Assignment Project Exam Help
https://eduassistpro.github.io/
Add WeChat edu_assist_pro

Exploiting Different Levels of Parallelism

- How do we write fast code to exploit the many levels of parallelism on a multicore platform?

Assignment Project Exam Help



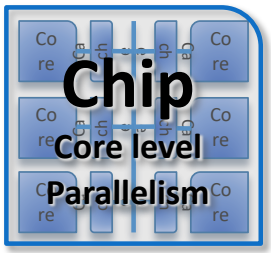
Exploited using
g compiler and
de intrinsics

https://eduassistpro.github.io/

Add WeChat edu_assist_pro

SMT-Level Parallelism

act it to
core-level parallelism



Core-Level Parallelism

Using **threads** to describe work done on different cores

What is a thread?

- A thread of execution is a unit of processing scheduled by the OS
  - Allows system resources to be shared efficiently

Assignment Project Exam Help

Processes	Threads
Exist independ	**s of processes**
Independent states: virtual memory space, handles to system objects, a security context, a unique process identifier, environment variables, a priority class, minimum and maximum working set sizes, and at least one thread of execution	**a process share:** **and system resources** **ndent:** **exception handlers, a scheduling priority, a unique thread identifier, and thread context**
Interact through system-provided inter-process communication mechanisms	**Interact through shared memory**
OS Preemptive	**OS Preemptive**

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

Landscape of Thread Programming

- POSIX threads
  - POSIX: Portable Operating System Interface for Unix – IEEE Standard
  - Defines a set of C programming language types, functions and constants
  - ***Manually expose and manage all concurrency with the API***
- OpenMP
  - OpenMP: Open emory multiprocessing
  - ***Programmer hints at the concurrency***
  - ***Compiler manages the parallelism***
- Intel Cilk Plus, Intel Thread Building Block (TBB):
  - Fork-join parallelism ("spawn" and "sync" in Clik, Parallel loops/containers in TBB)
  - ***Programmer exposes the concurrency***
  - ***Runtime determines what is run in parallel***

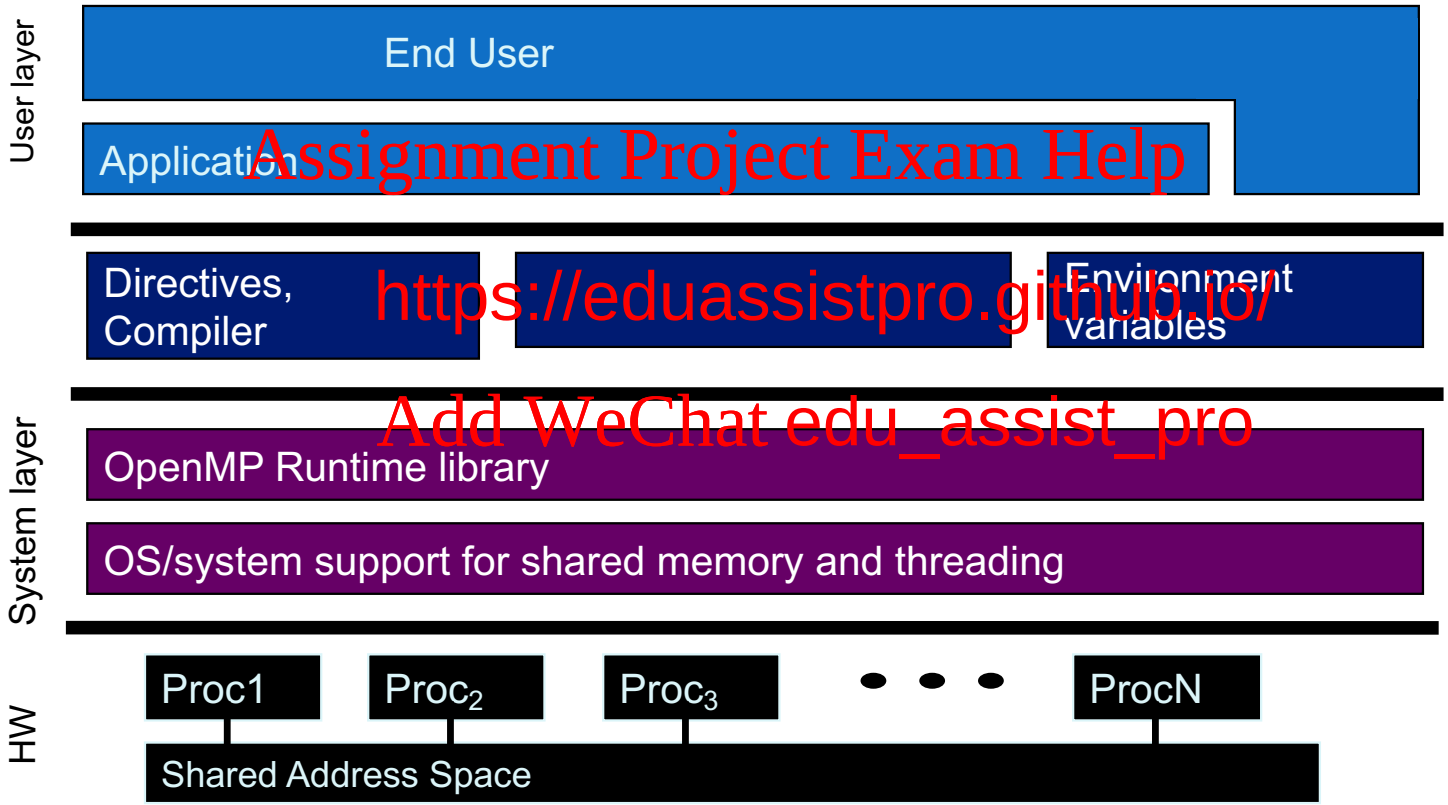
Assignment Project Exam Help
https://eduassistpro.github.io/
Add WeChat edu_assist_pro

Outline

- Mutlicore Shared-Memory Platforms
- OpenMP – API for Shared-Memory Multi-processing
  - **Overview**
  - Example: numerical integration
  - Shared variable
  - SPMD vs worksharing
  - Data environment options
  - Advanced worksharing options
  - Synchronization options
- How is This Relevant to Writing Fast Code?

Assignment Project Exam Help
https://eduassistpro.github.io/
Add WeChat edu_assist_pro

OpenMP Basic Defs: Solution Stack



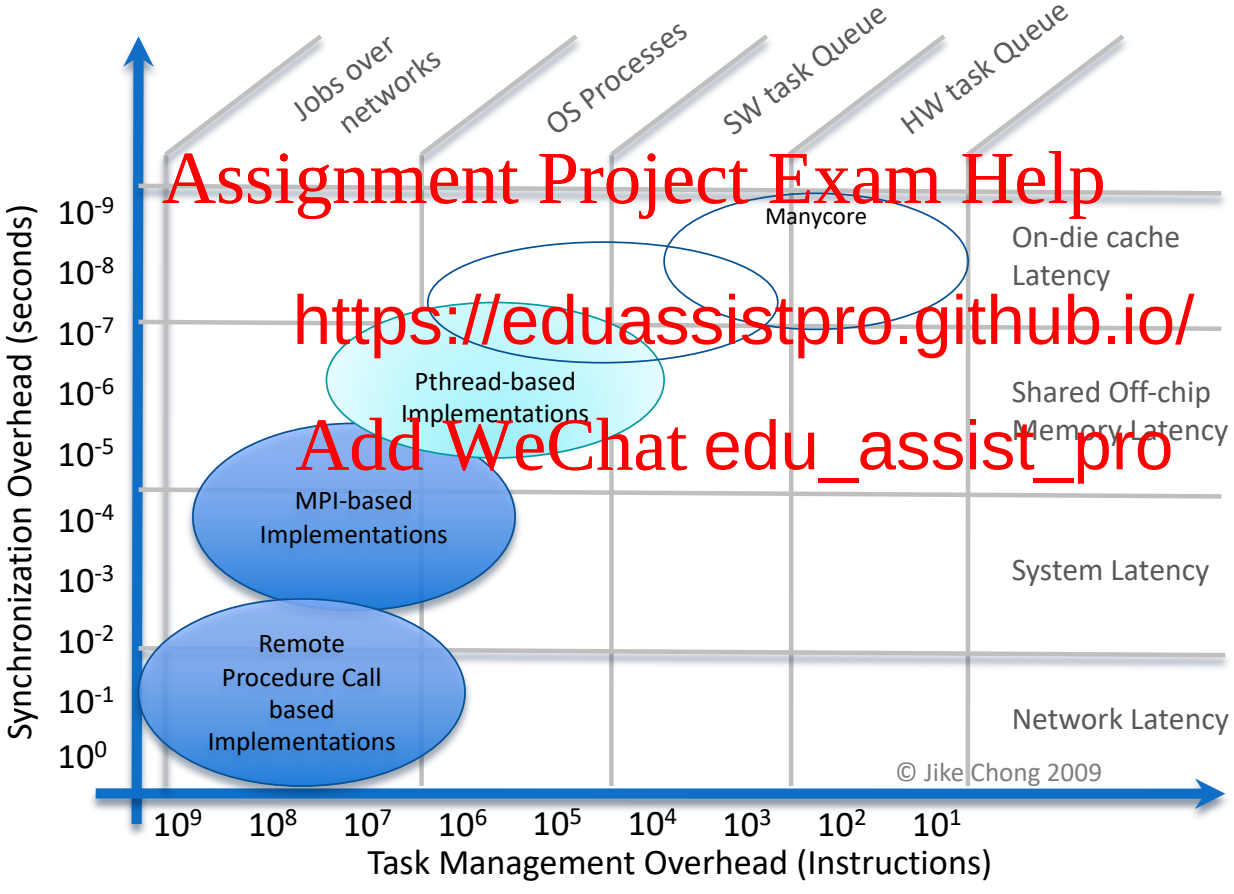
Assignment Project Exam Help
https://eduassistpro.github.io/
Add WeChat edu_assist_pro

Scope of OpenMP in this course

Topic	Concepts
I. OMP Intro	**Parallel regions**
II. Creating threads	**Parallel, default data environment, runtime library calls**
III. Synchronization	
IV. Parallel loops	
V. Odds and ends	Single, sections, master, vironment variables, synchronization, etc.
VI. Data Environment	Data environment detail tion
VII. OpenMP 3 and tasks	Tasks and other OpenMP 3 features
VIII. Memory model	The need for flush, but its actual use is left to an appendix
IX. Threadprivate	Modular software

Assignment Project Exam Help
https://eduassistpro.github.io/
Add WeChat edu_assist_pro

Spectrum of Granularity



Assignment Project Exam Help
https://eduassistpro.github.io/
Add WeChat edu_assist_pro

Example 1: Hello World

Verifying that your OpenMP environment works

- Write a multithreaded program where **each thread** prints “hello world”.

Assignment Project Exam Help

```
void main()
{

    int ID = 0;
    printf(" hello(%d) ", ID);
    printf(" world(%d) \n", ID);

}
```

https://eduassistpro.github.io/

Add WeChat edu_assist_pro

Example 1: SPMD with "omp parallel"

Verifying that your OpenMP environment works

- Write a multithreaded program where each thread prints "hello world".

Assignment Project Exam Help

```
#include <omp.h>
void main()
{
  #pragma omp parallel
  {
    int ID = omp_get_thread_num();
    printf(" hello(%d) ", ID);
    printf(" world(%d) \n", ID);
  }
}
```

https://eduassistpro.github.io/

```
compiling and linking

gcc
```

Add WeChat edu_assist_pro

Example 1: SPMD with "omp parallel"

A multi-threaded "Hello world" program

- Write a multithreaded program where each thread prints "hello world".

Assignment Project Exam Help

OpenMP include file

Parallel region

https://eduassistpro.github.io/

```
#include <omp.h>
void main()
{
  #pragma omp parallel
  {
    int ID = omp_get_thread_num();
    printf(" hello(%d) ", ID);
    printf(" world(%d) \n", ID);
  }
}
```

Add WeChat edu_assist_pro

Possible Output:

hello(0) world(1)

hello (3) hello(2) world(3)

world(2)

End of the Parallel region

Runtime library function to return a thread ID.

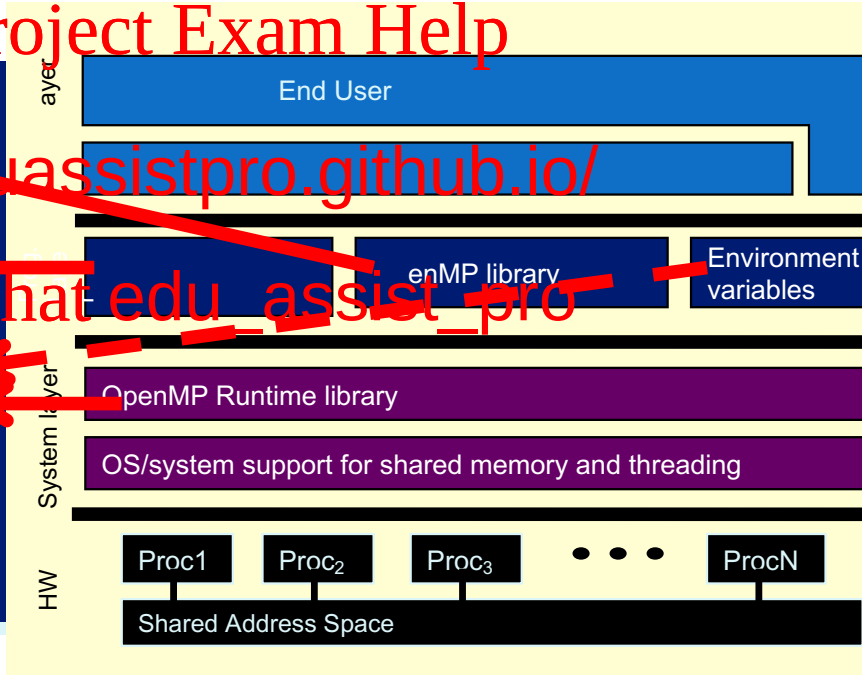
Example 1: SPMD with "omp parallel"

A multi-threaded "Hello world" program

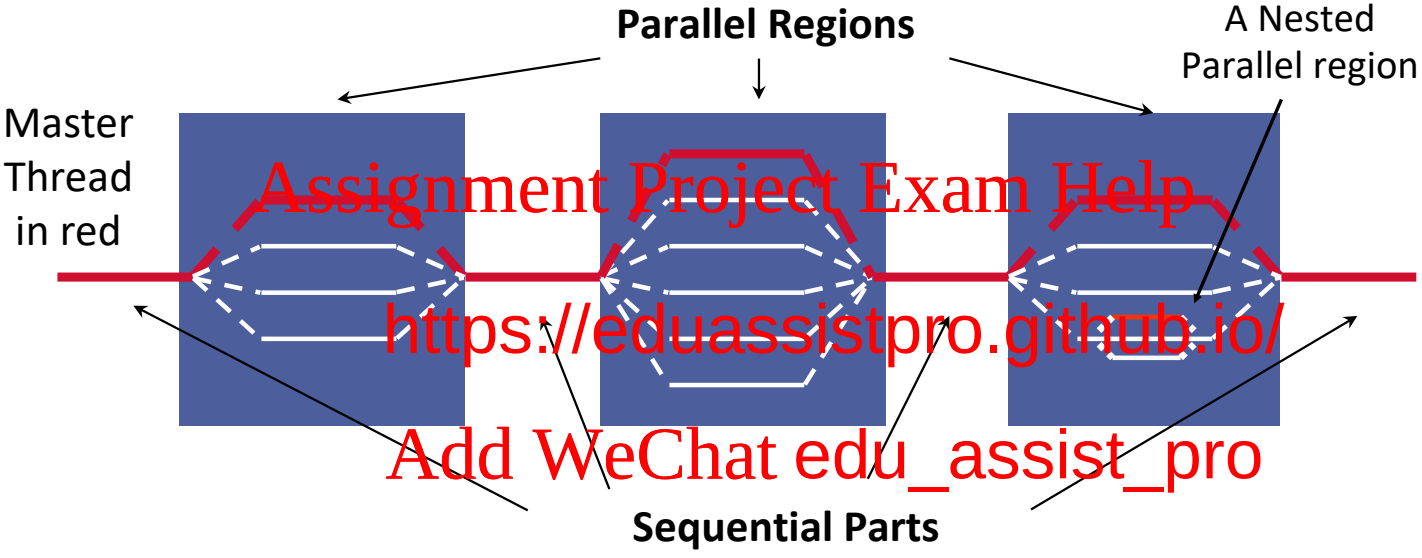
- Write a multithreaded program where each thread prints "hello world".

Assignment Project Exam Help
https://eduassistpro.github.io/
Add WeChat edu_assist_pro

```
#include <omp.h>
void main()
{
  #pragma omp parallel
  {
    int ID = omp_get_thread_num();
    printf(" hello(%d) ", ID);
    printf(" world(%d) \n", ID);
  }
}
```



OpenMP: Execution Pattern



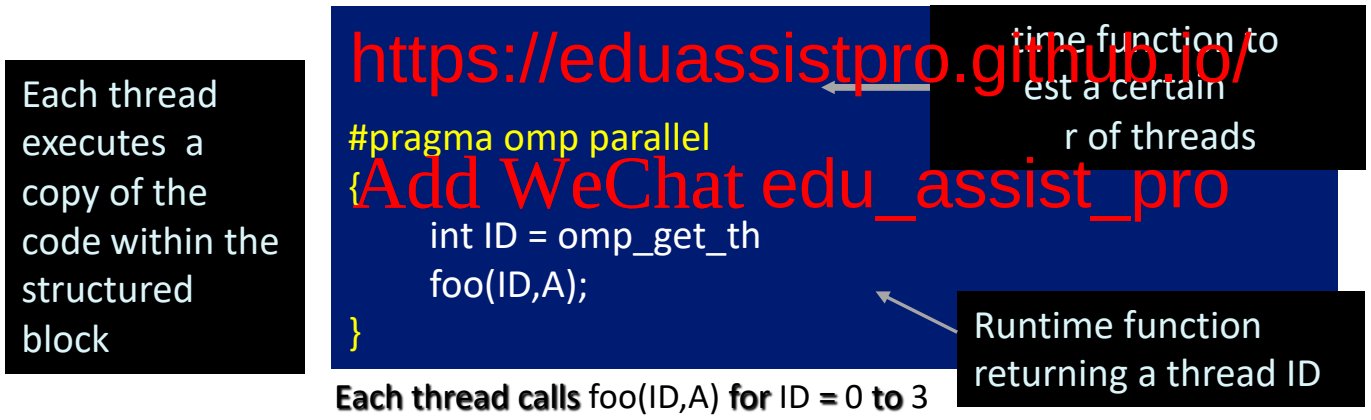
Assignment Project Exam Help
https://eduassistpro.github.io/
Add WeChat edu_assist_pro

- Managed **Fork-Join** Parallelism:
  - **Master thread** creates a **team of threads** as needed
  - Parallelism added incrementally until performance goals are met
  - i.e. the **sequential program evolves into a parallel program**

Thread Creation: Parallel Regions

- Threads in OpenMP are created with the parallel construct
  - To create a 4 thread Parallel region:

Assignment Project Exam Help



Thread Creation: Parallel Regions

- Threads in OpenMP are created with the parallel construct
  - To create a 4 thread Parallel region:

Assignment Project Exam Help

Runtime function to
est a certain
ber of threads

https://eduassistpro.github.io/

```
#pragma omp parallel nu
{
        int ID = omp_get_th
        foo(ID,A);
}
```

Add WeChat edu_assist_pro

Each thread executes a copy of the code within the structured block

Runtime function returning a thread ID

Each thread calls foo(ID,A) **for** ID **=** 0 **to** 3

Outline

- Mutlicore Shared-Memory Platforms
- OpenMP – API for Shared-Memory Multi-processing
  - Overview
  - **Example: num**
  - Shared variable
  - SPMD vs worksharing
  - Data environment options
  - Advanced worksharing options
  - Synchronization options
- How is This Relevant to Writing Fast Code?

Assignment Project Exam Help
https://eduassistpro.github.io/
Add WeChat edu_assist_pro

Example: Numerical Integration of Pi

Mathematically, we know that:

$$\int_0^1 \frac{4}{1+x^2}dx = 4 \cdot \tan^{-1} x \Big|_0^1 = \pi$$

Assignment Project Exam Help

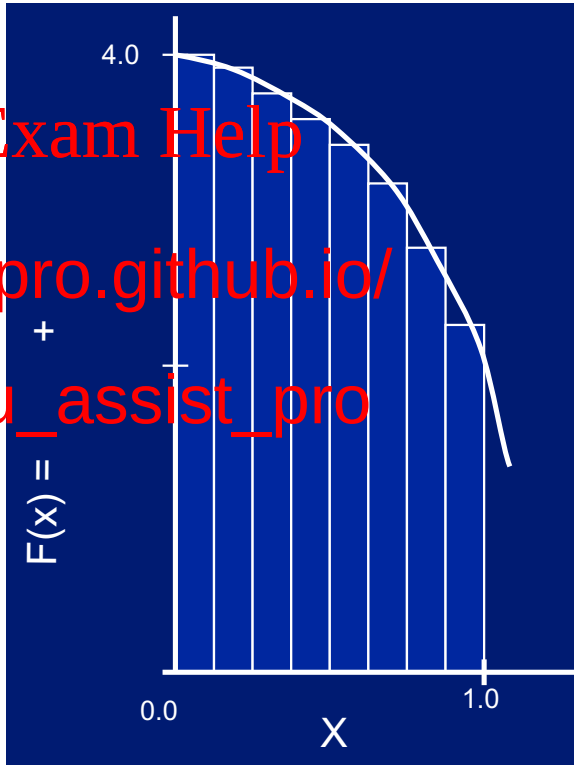
We can approximat rectangles:

https://eduassistpro.github.io/

Add WeChat edu_assist_pro

$$\sum_{i=0}^{N} F(x_i)\Delta x \approx \pi$$

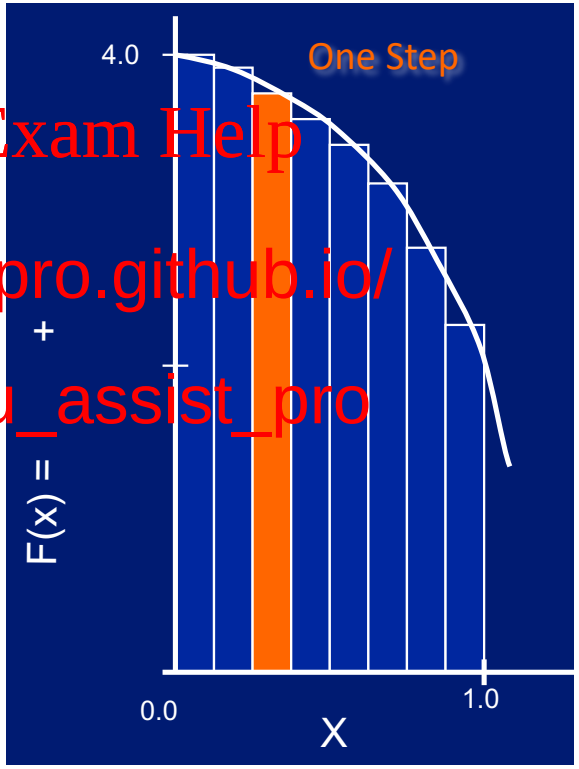
Where each rectangle has width Δx and height $F(x_i)$ at the middle of interval i.



Numerical Integration of Pi

```
#include <stdio.h>
static long num_steps = 100000; double step;

int main ()
{
    int i; double pi, sum;
    step = 1.0/(doub

    int i; double x;

    for (i=0, sum=0.0;i< num_steps; i++){
        x = (i+0.5)*step;
        sum += 4.0/(1.0+x*x);
    }

    pi = sum * step;
    printf("pi = %f \n", pi);
    return 0;
}
```

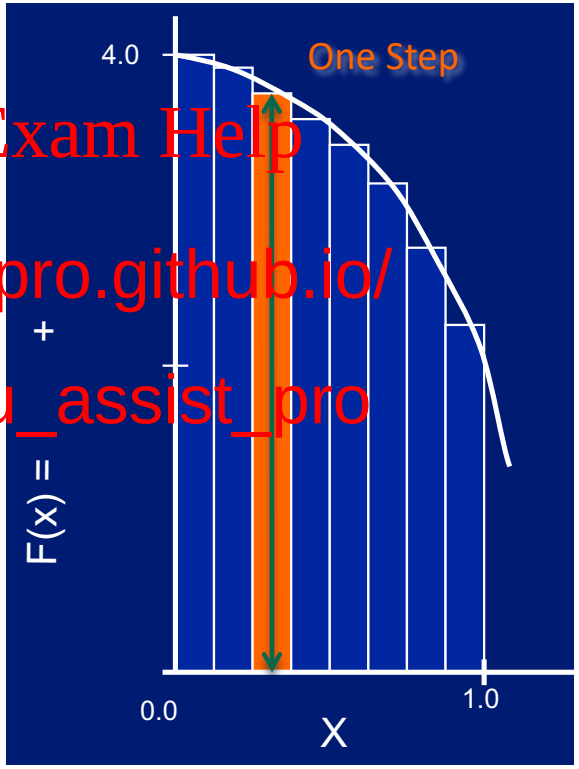
Assignment Project Exam Help
https://eduassistpro.github.io/
Add WeChat edu_assist_pro



Numerical Integration of Pi

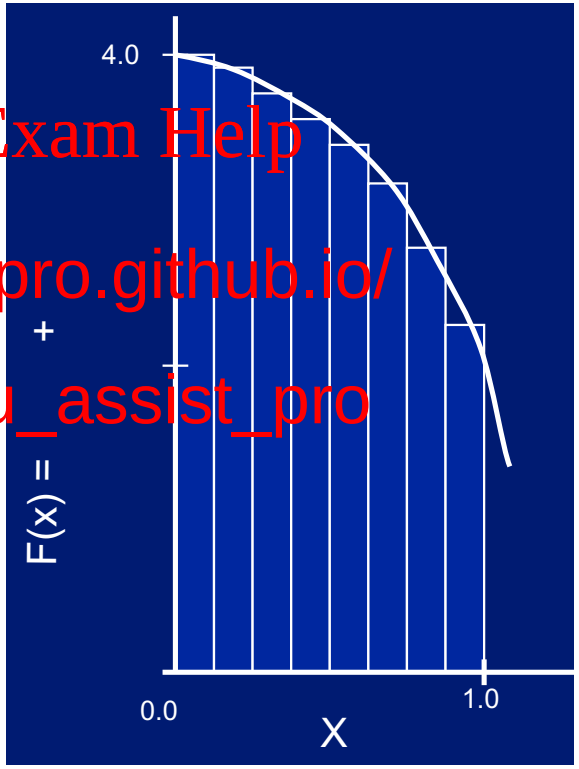
```
#include <stdio.h>
static long num_steps = 100000; double step;

int main ()
{
    int i; double pi, sum;
    step = 1.0/(doub

    int i; double x;

    for (i=0, sum=0.0;i< num_steps; i++){
        x = (i+0.5)*step;
        sum += 4.0/(1.0+x*x);
    }

    pi = sum * step;
    printf( "pi = %f \n", pi);
    return 0;
}
```

Assignment Project Exam Help
https://eduassistpro.github.io/
Add WeChat edu_assist_pro



Concurrency Opportunities

- What is parallelizable?
  - Calculation of each step is independent
- One can map each of the steps to an independent thre
  - **What's wrong**
- Significant thread management overhe
  - Compute each step as a separate thread involves significant thread management overhead (hundreds of cycles)

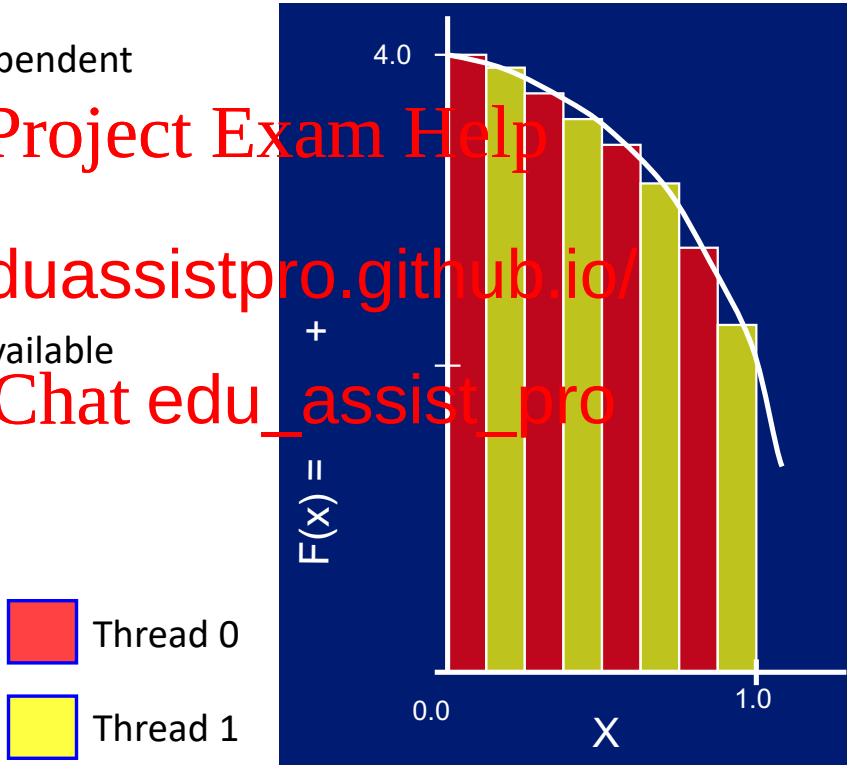


Assignment Project Exam Help
https://eduassistpro.github.io/
Add WeChat edu_assist_pro

Concurrency Opportunities

- What is parallelizable?
  - Calculation of each step is independent
- Alternative:
  - Specify a fixed
    e.g. two thread
  - Parallelize workload over the available
    number of threads

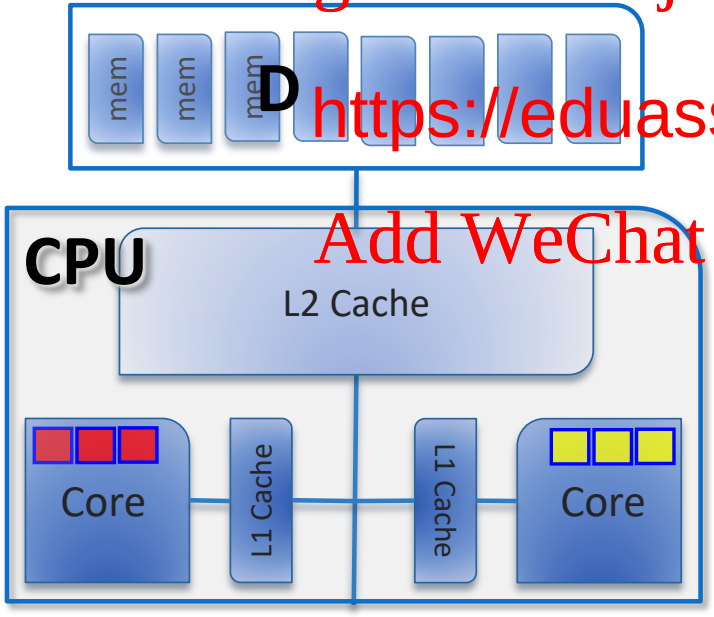
Assignment Project Exam Help
https://eduassistpro.github.io/
Add WeChat edu_assist_pro

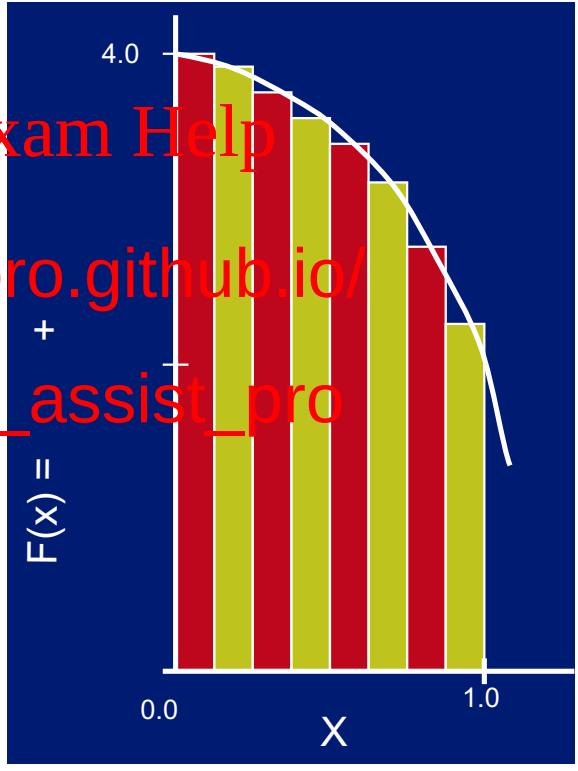


Implementation View

- Work is evenly distributed across two processors

Assignment Project Exam Help
https://eduassistpro.github.io/
Add WeChat edu_assist_pro

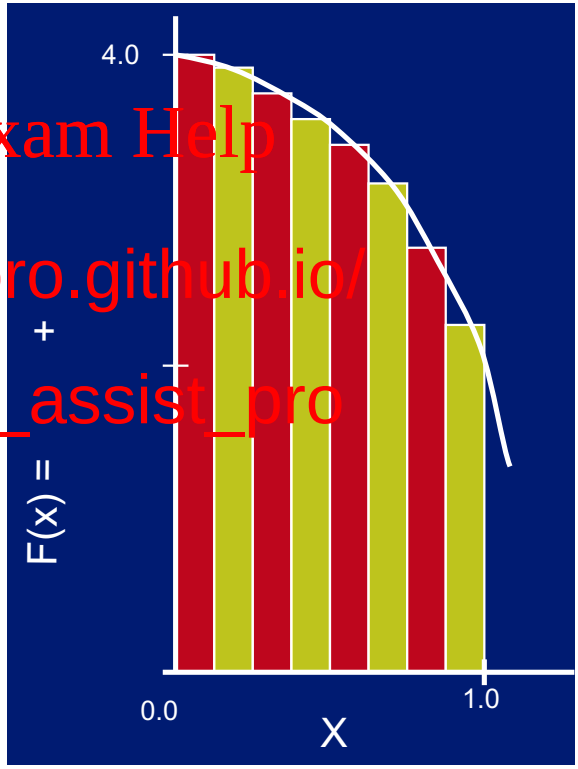




Example 2: Numerical Integration of Pi

Sequential: 2.12 seconds

```
#include <stdio.h>
static long num_steps = 100000000; double step;

int main ()
{
     int i; double pi, sum = 0.0;
     step = 1.0/(dou


     double x;

     for (i=0;i< num_steps; i++){
          x = (i+0.5)*step;
          sum += 4.0/(1.0+x*x);
     }


     pi = sum * step;
     printf("pi = %f \n", pi);
     return 0;
}
```



Assignment Project Exam Help
https://eduassistpro.github.io/
Add WeChat edu_assist_pro

Outline

- Mutlicore Shared-Memory Platforms
- OpenMP – API for Shared-Memory Multi-processing
  - Overview
  - Example: nume
  - **Shared variabl**
  - SPMD vs worksharing
  - Data environment options
  - Advanced worksharing options
  - Synchronization options
- How is This Relevant to Writing Fast Code?

Assignment Project Exam Help
https://eduassistpro.github.io/
Add WeChat edu_assist_pro

Example 3: Critical Region

```
#include <omp.h>
#include <stdio.h>
static long num_steps = 100000000; double step;
#define NUM_THREADS 2
int main ()
{
     int i; double pi; int nthreads; double sum=0.0;
     step = 1.0/(dou
     #pragma omp par
     {
          int i, id = omp_get_thread_num(); double x;
          int nthrds = omp_get_num_threads
          for (i=id;i< num_steps; i=i+nthr
               x = (i+0.5)*step;
               #pragma omp critical
               {sum += 4.0/(1.0+x*x);}
          }
          if (id == 0)   nthreads = nthrds;
     }
     pi = sum * step;
     printf("pi = %f \n", pi);
     return 0;
}
```

Sequential: 2.12 seconds

64.45 seconds

Why so slow?

Assignment Project Exam Help
https://eduassistpro.github.io/
Add WeChat edu_assist_pro

Example 4: Atomic Operation

```
#include <omp.h>
#include <stdio.h>
static long num_steps = 100000000; double step;
#define NUM_THREADS 2
int main ()
{
     int i; double pi; int nthreads; double sum=0.0;
     step = 1.0/(dou
     #pragma omp par
     {
          int i, id = omp_get_thread_num(); double x, tmp;
          int nthrds = omp_get_num_threads
          for (i=id;i< num_steps; i=i+nthr
               x = (i+0.5)*step;
               tmp = 4.0/(1.0+x*x);
               #pragma omp atomic
               {sum += tmp}
          }
          if (id == 0)   nthreads = nthrds;
     }
     pi = sum * step;
     printf("pi = %f \n", pi);
     return 0;
}
```

Sequential: 2.12 seconds

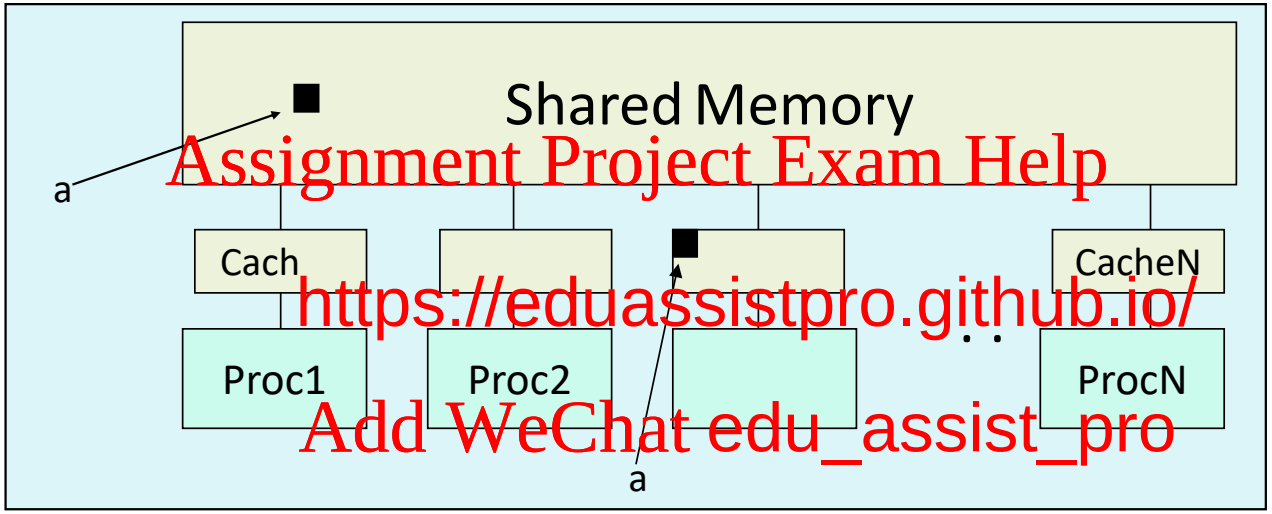
4.93 seconds

Better, why still so slow?

Assignment Project Exam Help
https://eduassistpro.github.io/
Add WeChat edu_assist_pro

Shared Memory Model

Based on slides from Mattson et al, A "Hands-on" Introduction to OpenMP

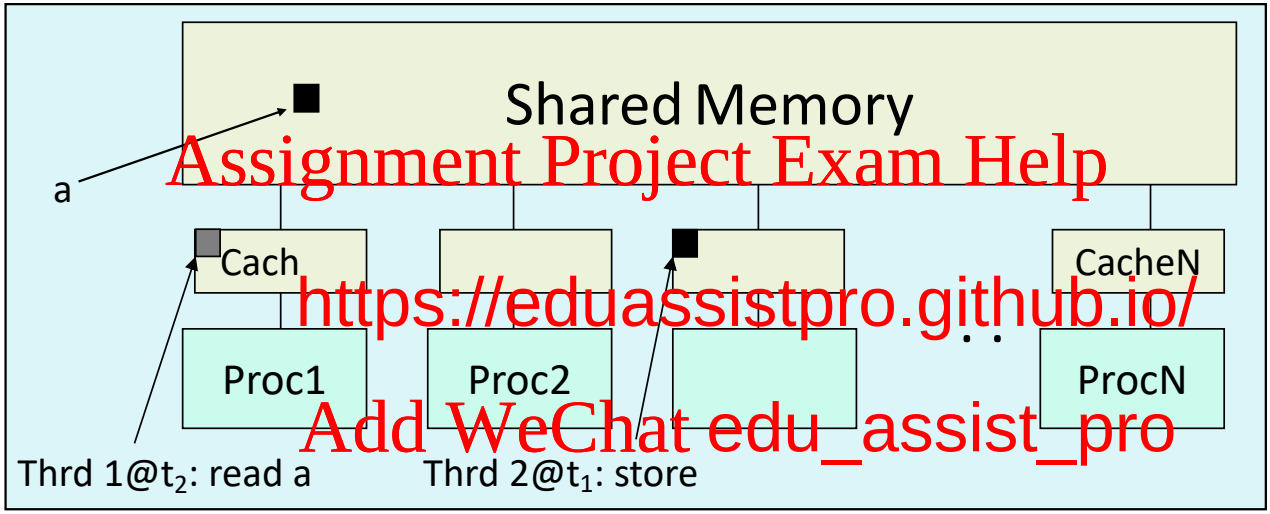


Assignment Project Exam Help
https://eduassistpro.github.io/
Add WeChat edu_assist_pro

- All threads share an address space
- Multiple copies of data may be present in various levels of cache

Shared Memory Model

Based on slides from Mattson et al, A "Hands-on" Introduction to OpenMP

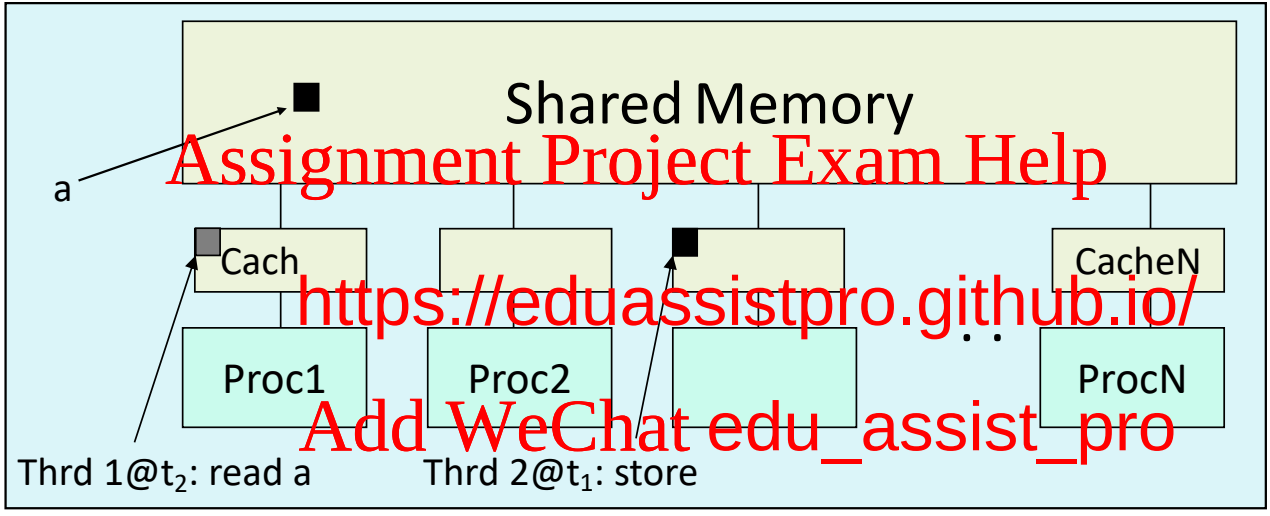


Assignment Project Exam Help
https://eduassistpro.github.io/
Add WeChat edu_assist_pro

- A piece of data (at address "a") can be:
  - Written into memory by Thrd 2 at t_1
  - Then, read by Thrd 1 at t_2
- We say the data at address "a" is in a ***shared*** state

Managing Coherence Costly

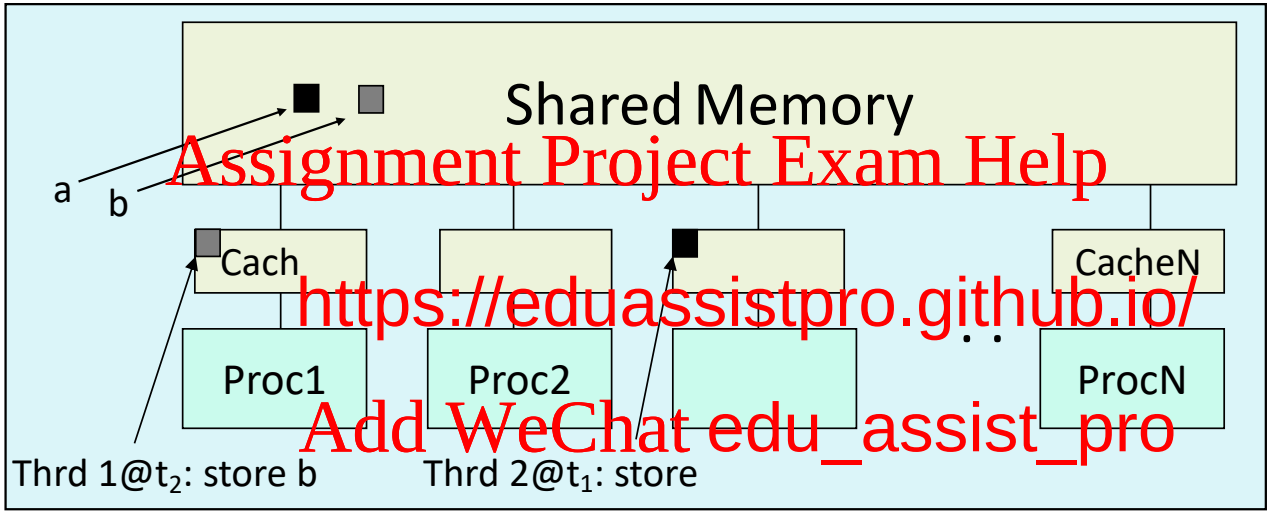
Based on slides from Mattson et al, A "Hands-on" Introduction to OpenMP



Assignment Project Exam Help
https://eduassistpro.github.io/
Add WeChat edu_assist_pro

- Synchronization protocols are implemented in hardware to manage coherence between caches
  - e.g. **MESI Protocol** (Papamarcos & Patel 1984)
  - Allows data to be in one of four states: *Modified, Exclusive, Shared, Invalid*
- Transparent to software, but could have **severe performance penalties**

Threads can Write to Different Addresses…



Assignment Project Exam Help
https://eduassistpro.github.io/
Add WeChat edu_assist_pro

- Avoid cache **synchronization protocols** induced **severe performance penalties** when no sharing is required
- Technique:
  - Write to separate addresses during parallel computation

Example 5: Separate Result Storage

```
#include <omp.h>
#include <stdio.h>
static long num_steps = 100000000; double step;
#define NUM_THREADS 2
int main ()
{
     int i; double p                    EADS                    ]
     step = 1.0/(dou
     #pragma omp par
     {
          int i, id = omp_get_thread_num()
          int nthrds = omp_get_num_threads
          for (i=id, sum[id]=0.0;i< num_st                    {
               x = (i+0.5)*step;
               sum[id] += 4.0/(1.0+x*x);
          }
          if (id == 0)   nthreads = nthrds;
     }
     for(i=0, pi=0.0;i<nthreads;i++)pi += sum[i] * step;
     printf("pi = %f \n", pi);
     return 0;
}
```

Sequential: 2.12 seconds

1.47 seconds

Better, but why slower than 2x speedup?

Assignment Project Exam Help
https://eduassistpro.github.io/
Add WeChat edu_assist_pro

Structure of a Cache

- **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

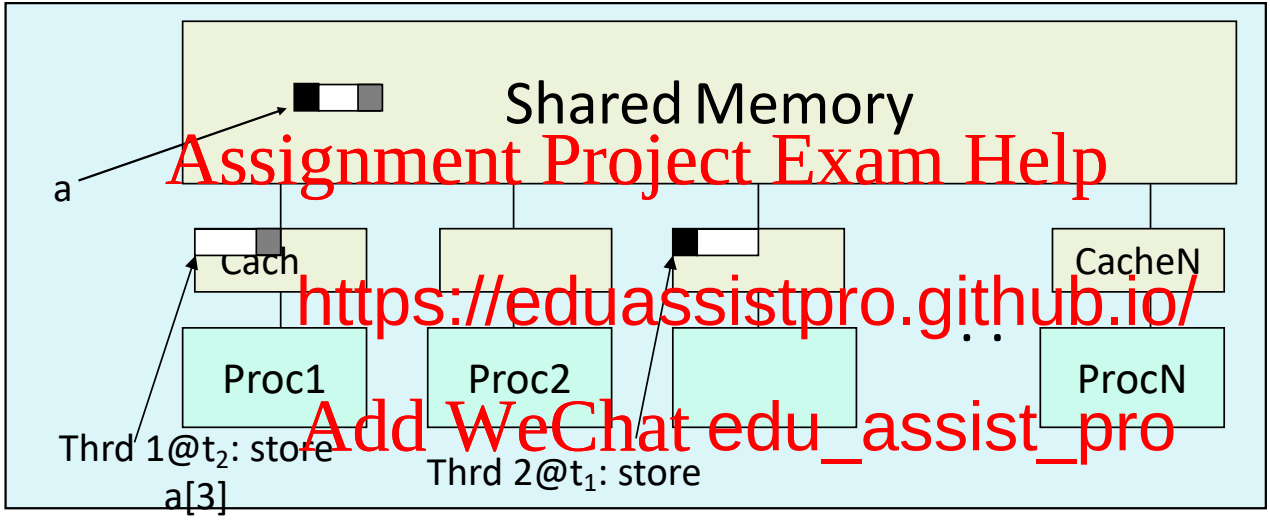
- **Temporal Locality**
  - Reuse of specific and/or resources within relatively small time durations

https://eduassistpro.github.io/

Add WeChat edu_assist_pro

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

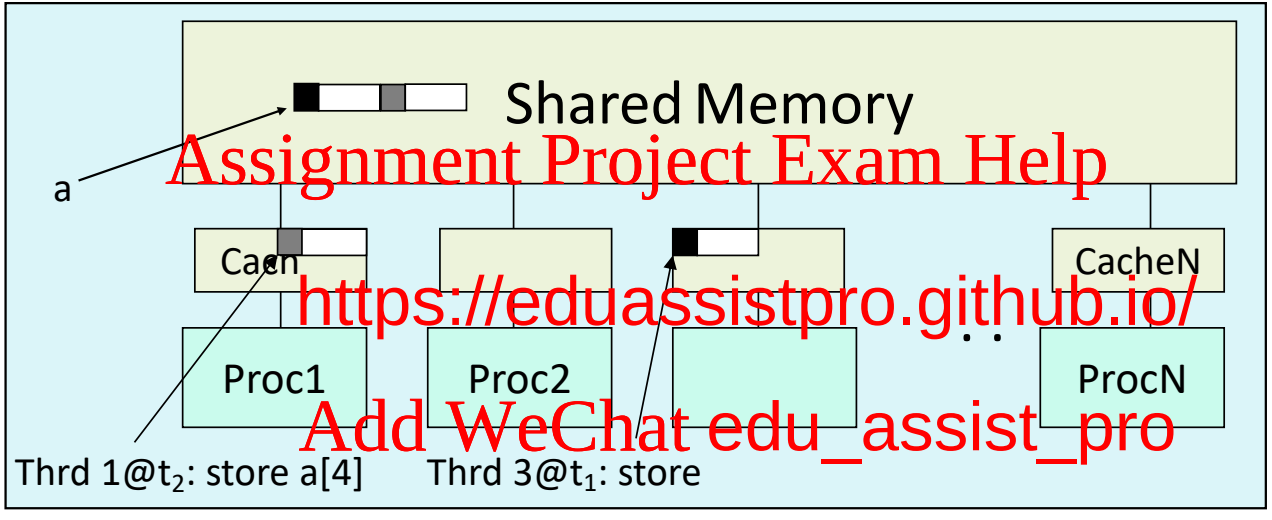
Is data being share?



Assignment Project Exam Help
https://eduassistpro.github.io/
Add WeChat edu_assist_pro

- Cache loads and stores work with 4-16 word long cache lines (64B for Intel)
  - If two threads are writing to the same cache line, conflicts occurs
- Even if the address differs, one will still suffer **performance penalty**
- This is called **false sharing**

Solution for False Sharing



Assignment Project Exam Help
https://eduassistpro.github.io/
Add WeChat edu_assist_pro

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

Example 6: Eliminate False Sharing

```
#include <omp.h>
#include <stdio.h>
static long num_steps = 100000000; double step;
#define NUM_THREADS 2
int main ()
{
     int i; double p
     step = 1.0/(dou
     #pragma omp par
     {
          int i, id = omp_get_thread_num()                    sum=0.0;
          int nthrds = omp_get_num_threads
          for (i=id;i< num_steps; i=i+nthr
            x = (i+0.5)*step;
            local_sum += 4.0/(1.0+x*x);
          }
          #pragma omp critical
          pi += local_sum * step;
     }
     printf("pi = %f \n", pi);
     return 0;
}
```

Sequential: 2.12 seconds

1.15 seconds

Assignment Project Exam Help
https://eduassistpro.github.io/
Add WeChat edu_assist_pro

Example 6: Eliminate False Sharing

```
#include <omp.h>
#include <stdio.h>
static long num_steps = 100000000; double step;
#define NUM_THREADS 2
int main ()
{
     int i; double p
     step = 1.0/(dou
     #pragma omp par                                    pte(x  , local_sum)
     {
          int i, id = omp_get_thread_num()
          int nthrds = omp_get_num_threads
          for (i=id;i< num_steps; i=i+nthr
            x = (i+0.5)*step;
            local_sum += 4.0/(1.0+x*x);
          }
          #pragma omp critical
          pi += local_sum * step;
     }
     printf("pi = %f \n", pi);
     return 0;
}
```

Sequential: 2.12 seconds

1.15 seconds

Great!
But is this the best we can do?

Assignment Project Exam Help
https://eduassistpro.github.io/
Add WeChat edu_assist_pro

Example 7: Sequential Optimization

```
#include <omp.h>
#include <stdio.h>
static long num_steps = 100000000; double step;
#define NUM_THREADS 2
int main ()
{
     int i; double p
     step = 1.0/(dou
     #pragma omp par                                  ate(x, local_sum)
     {
          int i, id = omp_get_thread_num();
          int nthrds = omp_get_num_threads
          double hoop = nthrds*step;
          for (i=id, x=(i+0.5)*step;i< num_steps; i=i+nthrds) {
            x += hoop;
            local_sum += 4.0/(1.0+x*x);
          }
          #pragma omp critical
          pi += local_sum * step;
     }
     printf("pi = %f \n", pi);
     return 0;
}
```

Sequential: 2.12 seconds

0.46 seconds

Assignment Project Exam Help
https://eduassistpro.github.io/
Add WeChat edu_assist_pro

Outline

- Mutlicore Shared-Memory Platforms
- OpenMP – API for Shared-Memory Multi-processing
  - Overview
  - Example: numerical integration
  - Shared variable
  - **SPMD vs worksharing**
  - Data environment options
  - Advanced worksharing options
  - Synchronization options
- How is This Relevant to Writing Fast Code?

Assignment Project Exam Help
https://eduassistpro.github.io/
Add WeChat edu_assist_pro

SPMD vs. Worksharing

- A parallel construct by itself creates an SPMD
  - "Single Program Multiple Data"
  - Programmer must explicitly specify what each thread must do differently
  - The division of work is hard-coded in the program
- Opportunity:
  - Many parallel regions are loops
- Question:
  - Can we make it easier to parallelize these loops?

Assignment Project Exam Help
https://eduassistpro.github.io/
Add WeChat edu_assist_pro

OpenMP: Parallel For

- One of the worksharing constructs OpenMP provide is **"omp for"**

Sequential code

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

Assignment Project Exam Help
https://eduassistpro.github.io/
Add WeChat edu_assist_pro

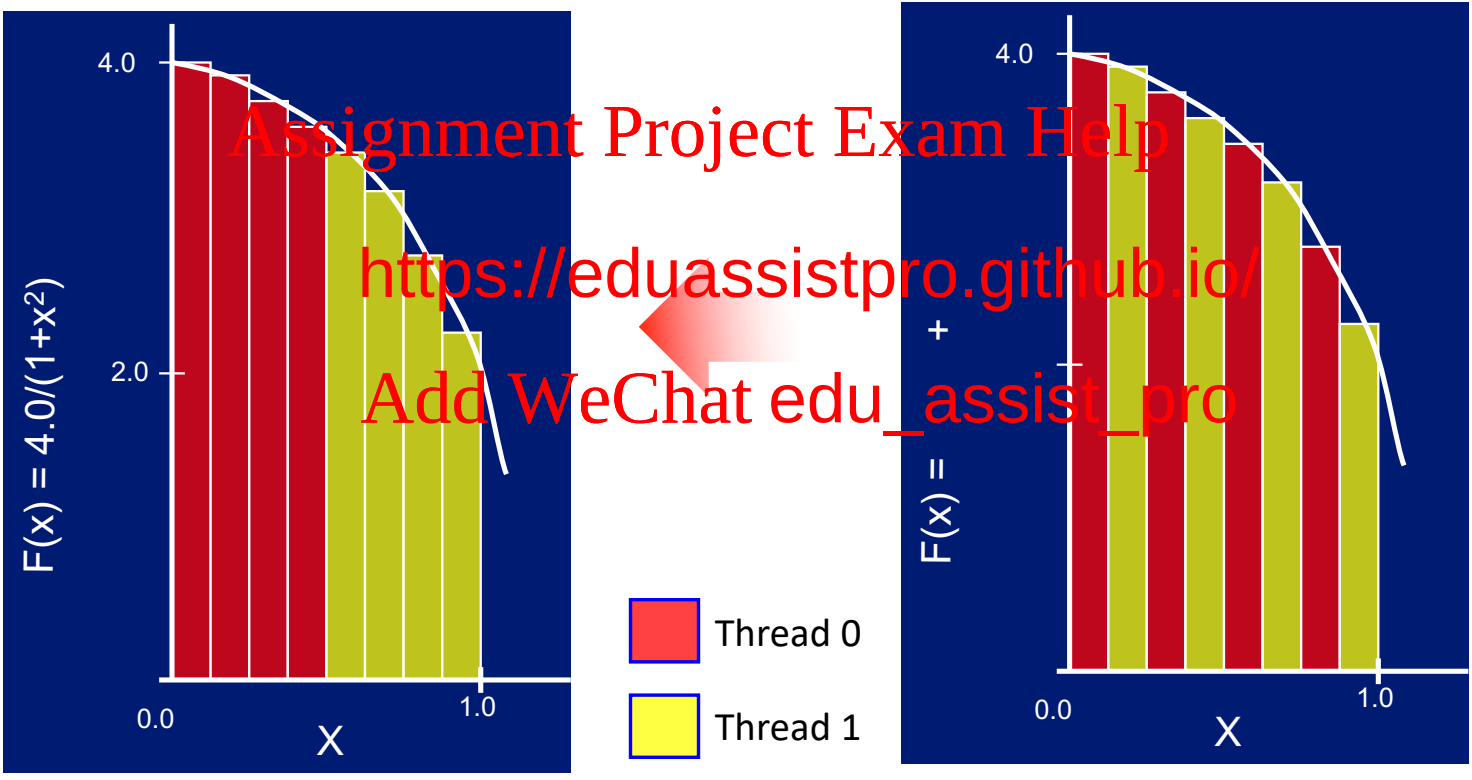
OpenMP **"parallel"** region

```
#pragma omp parallel
                                                    ;
                                                    ();
        Nthrds = o                                      ds();
        istart = id *
        iend = (id+
        if (id == Nthrds-1)iend = N;
        for(i=istart;i<iend;i++)   { a[i] = a[i] + b[i];}
}
```

OpenMP **"parallel"** region and a worksharing **"for"** construct

```
#pragma omp parallel
#pragma omp for
        for(i=0;i<N;i++)   { a[i] = a[i] + b[i];}
```

Other Concurrency Opportunities



Assignment Project Exam Help
https://eduassistpro.github.io/
Add WeChat edu_assist_pro

Example 8: Parallel For and Reduction

```
#include <omp.h>
#include <stdio.h>
static long num_steps = 100000000; double step;
#define NUM_THREADS 2
int main ()
{
     int i; double p
     step = 1.0/(dou
     #pragma omp par
     {
          #pragma omp for reduction(+:sum)
          for (i=0;i< num_steps; i++){
               x = (i+0.5)*step;
               sum = sum + 4.0/(1.0+x*x);
          }
     }
     Pi = sum * step;
     printf("pi = %f \n", pi);
     return 0;
}
```

Sequential: 2.12 seconds

1.22 seconds

Reduction is more scalable than critical sections.

"i" is private by default

Loops must not have loop carry dependency. **Limitation of compiler technology.**

Assignment Project Exam Help
https://eduassistpro.github.io/
Add WeChat edu_assist_pro

Combined Parallel/Worksharing Construct

- OpenMP shortcut: Put the "parallel" and the worksharing directive on the same line

Assignment Project Exam Help

```
double  res[MAX];
#pragma omp pa
{
   #pragma omp for
   for (i=0;i< MAX; i++) {
       res[i] = huge();
   }
}
```

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

```
res[MAX]; int i;
a omp parallel for
    i< MAX; i++) {
    = huge();
}
```

OpenMP: Working With Loops

- Basic approach
  - Find compute intensive loops
  - Make the loop iterations independent . So they can safely execute in any order without loop-carried dependencies
  - Place the appro

Assignment Project Exam Help
https://eduassistpro.github.io/
Add WeChat edu_assist_pro

```
double hoop
for (i=id, x=(i+0.5)*step;i< num                    s) {
  x += hoop;
  sum = sum + 4.0/(1.0+x*x);
}
```

```
for (i=0;i< num_steps; i++){
     x = (i+0.5)*step;
     sum = sum + 4.0/(1.0+x*x);
}
```

OpenMP: Reductions

```
for (i=0;i< num_steps; i++){
     x = (i+0.5)*step;
     sum = sum + 4.0/(1.0+x*x);
}
```

```
#pragma omp for r
for (i=0;i< num_s
     x = (i+0.5)*step;
     sum = sum + 4.0/(1.0+x*x);
}
```

reduction (op : list)

1. A local copy of each list variable is made and initialized depending on

g. 0 for "+")

upon the local copy

are reduced into a

combined with the

alue

Assignment Project Exam Help

https://eduassistpro.github.io/

Add WeChat edu_assist_pro

- Accumulating values into a single variable (sum) creates true dependence between loop iterations that can't be trivially removed
- This is a very common situation … it is called a "reduction".
- Support for reduction operations is included in most parallel programming environments.

OpenMP: Reduction Operators

- Many different associative operands can be used with reduction:
- Initial values are the ones that make sense mathematically.

Assignment Project Exam Help
https://eduassistpro.github.io/
Add WeChat edu_assist_pro

Operator	Initial value
+	0
*	1
-	0

	Initial value
	~0
	0
^	0
&&	1
||	0

Example 8: Parallel For and Reduction

```
#include <omp.h>
#include <stdio.h>
static long num_steps = 100000000; double step;

int main ()
{
     int i; double p
     step = 1.0/(dou
     #pragma omp par                    m )
     for (i=0;i< num_steps; i++){
          x = (i+0.5)*step;
          sum = sum + 4.0/(1.0+x*x);
     }
     Pi = sum * step;
     printf("pi = %f \n", pi);
     return 0;
}
```

Sequential: 2.12 seconds

1.22 seconds

Assignment Project Exam Help
https://eduassistpro.github.io/
Add WeChat edu_assist_pro

we created a parallel
out changing any code
and by adding 2 simple lines of text!

Disadvantage:
1) Lot's of pitfalls if you don't understand system architecture
2) The basic result may not the fastest one can do

Outline

- Multicore Shared-Memory Platforms
- OpenMP – API for Shared-Memory Multi-processing
  - Overview
  - Example: numerical integration
  - Shared variable
  - SPMD vs worksharing
  - **Data environment options**
  - Advanced worksharing options
  - Synchronization options
- How is This Relevant to Writing Fast Code?

Assignment Project Exam Help
https://eduassistpro.github.io/
Add WeChat edu_assist_pro

Data Environment in OpenMP

- Shared Memory programming model:
  - Most variables are shared by default
- Global variables are **SHARED** among threads
  - File scope v
  - Dynamically ew)
- But not everything is shared...
  - Functions called from parallel regions are **PRIVATE**
  - Automatic variables within a statement block are **PRIVATE**.

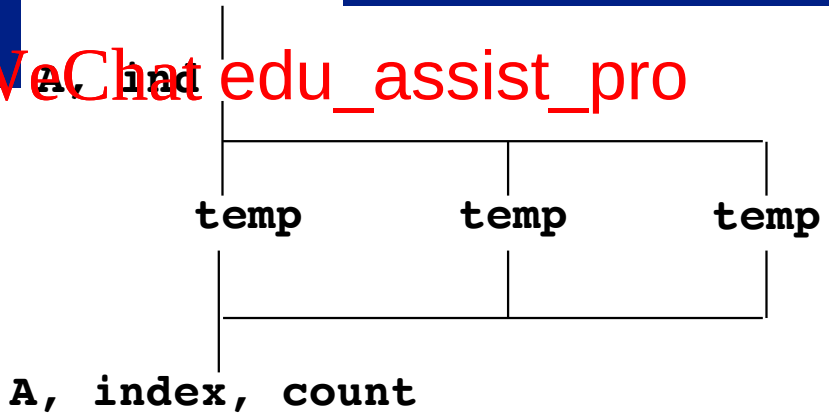
Assignment Project Exam Help
https://eduassistpro.github.io/
Add WeChat edu_assist_pro

Example: Data Sharing

```
double A[10];
int main() {
    int index[10];
    #pragma omp parallel
            work(index)
    printf("%d\n", ind
}
```

```
extern double A[10];
void work(int *index) {
  double temp[10];
  static int count;
```

Assignment Project Exam Help
https://eduassistpro.github.io/
Add WeChat edu_assist_pro



A, index and count are shared by all threads.

temp is local to each thread

Changing Storage Attributes

- One can selectively change storage attributes for constructs using the following clauses:
  - SHARED
  - PRIVATE
  - FIRSTPRIVA
- The final value of a private inside p can be transmitted to the shared variab loop with:
  - LASTPRIVATE

Assignment Project Exam Help
https://eduassistpro.github.io/
Add WeChat edu_assist_pro

Example 9: firstprivate example

```
#include <omp.h>
#include <stdio.h>
static long num_steps = 100000000; double step;
#define NUM_THREADS 2
int main ()
{
     int i; double pi, x, local_sum = 0.0; int nthreads;
     step = 1.0/(dou
     omp_set_num_thr
     #pragma omp par sum
     {
          int i, id = omp_get_thread_num()
          int nthrds = omp_get_num_threads
          double hoop = nthrds*step;
          for (i=id, x=(i+0.5)*step;i< num_steps; i=i+nthrds) {
            x += hoop;
            local_sum += 4.0/(1.0+x*x);
          }
          #pragma omp critical
          pi += local_sum * step;
     }
     printf("pi = %f \n", pi);
     return 0;
}
```

Assignment Project Exam Help
https://eduassistpro.github.io/
Add WeChat edu_assist_pro

Outline

- Mutlicore Shared-Memory Platforms
- OpenMP – API for Shared-Memory Multi-processing
  - Overview
  - Example: numerical integration
  - Shared variable
  - SPMD vs worksharing
  - Data environment options
  - **Advanced worksharing options**
  - Synchronization options
- How is This Relevant to Writing Fast Code?

Assignment Project Exam Help
https://eduassistpro.github.io/
Add WeChat edu_assist_pro

The Scheduling Cause

- The schedule clause affects how loop iterations are mapped onto threads
  - schedule(static [,chunk])
    - Deal-out blocks of iterations of size “chunk” to each thread.
  - schedule(dyna
    - Each thre til all iterations have been handled.
  - schedule(guided[,chunk])
    - Threads dynamically grab blocks of e of the block starts large and shrinks down to size “chunk” as the calculation proceeds.
  - schedule(runtime)
    - Schedule and chunk size taken from the OMP_SCHEDULE environment variable (or the runtime library)

Assignment Project Exam Help
https://eduassistpro.github.io/
Add WeChat edu_assist_pro

The Scheduling Cause

Schedule Clause	When To Use
STATIC	programmer
DYNAMIC	Unpredictable, variable work per iteration
GUIDED	Special case of dynamic to reduce scheduling overhead

Least work at runtime : scheduling done at compile-time

Most work at runtime : complex scheduling logic used at run-time

Assignment Project Exam Help
https://eduassistpro.github.io/
Add WeChat edu_assist_pro

Example 9: Scheduling

```
#include <omp.h>
#include <stdio.h>
static long num_steps = 100000000; double step;

int main ()
{
     int i; double p
     step = 1.0/(dou
     #pragma omp par                    ) schedule(static,
     10000000)
     for (i=0;i< num_steps; i++){
          x = (i+0.5)*step;
          sum = sum + 4.0/(1.0+x*x);
     }
     Pi = sum * step;
     printf("pi = %f \n", pi);
     return 0;
}
```

Sequential: 2.12 seconds

1.31 seconds

Assignment Project Exam Help
https://eduassistpro.github.io/
Add WeChat edu_assist_pro

Outline

- Mutlicore Shared-Memory Platforms
- OpenMP – API for Shared-Memory Multi-processing
  - Overview
  - Example: numerical integration
  - Shared variable
  - SPMD vs worksharing
  - Data environment options
  - Advanced worksharing options
  - **Synchronization options**
- How is This Relevant to Writing Fast Code?

Assignment Project Exam Help
https://eduassistpro.github.io/
Add WeChat edu_assist_pro

Synchronization: ordered

- The **ordered** region executes in the sequential order
- Important for some scientific code and optimization code
  - Order of reduction may cause rounding differences

Assignment Project Exam Help
https://eduassistpro.github.io/
Add WeChat edu_assist_pro

```
#pragma om
#pragma omp for ordered reducti
      for (I=0;I<N;I++){
            tmp = NEAT_STUFF(I);
#pragma ordered
            res += consum(tmp);
      }
```

Synchronization Barrier

- **Barrier**: Each thread waits until all threads arrive

Assignment Project Exam Help

```
#pragma omp parallel shared (A, B, C) private(id)
{
      id=omp_get_
      A[id] = big_c
#pragma omp barrier
#pragma omp for
      for(i=0;i<N;i++){C[i]=big_calc3(i,A);}
#pragma omp for nowait
      for(i=0;i<N;i++){ B[i]=big_calc2(C,  i); }
      A[id] = big_calc4(id);
}
```

https://eduassistpro.github.io/

Add WeChat edu_assist_pro

it barrier at the end of
orkSharing construct

it barrier due to
nowait

Implicit barrier at the end of
a parallel region

“Single” Worksharing Construct

- The **single** construct denotes a block of code that is executed by only one thread (not necessarily the master thread).
- A barrier is implied at the end of the single block
  - can remove the barrier with a nowait clause

Assignment Project Exam Help
https://eduassistpro.github.io/
Add WeChat edu_assist_pro

```
#pragma omp parallel
{
      do_many_things();
#pragma omp single
      {    exchange_boundaries();   }
      do_many_other_things();

}
```

```
                arallel
      do_many_things();
#pragma omp single nowait
      {    exchange_boundaries();   }
      do_many_other_things();

}
```

Nested Parallelism

- **Q:** Is nested parallel possible with OpenMP?
- **A:** Yes. But be sure to understand why you want to use it.

Assignment Project Exam Help
https://eduassistpro.github.io/
Add WeChat edu_assist_pro

```
#include <omp.h>
#include <stdio.h>
void report_num_thre
     #pragma omp sin
     { printf("L %d: num_threads = %d\n",                    ());}
}
int main() {
     omp_set_dynamic(0);
     #pragma omp parallel num_threads(2)
     {
          report_num_threads(1);
          #pragma omp parallel num_threads(2)
          {
               report_num_threads(2);
               #pragma omp parallel num_threads(2)
                    report_num_threads(3);
          }
     }
     return(0);
}
```

http://download.oracle.com/docs/cd/E19205-01/819-5270/aewbc/index.html

Nested Parallelism

- Compiling and running this program with nested parallelism enabled produces the following (sorted) output:

Assignment Project Exam Help

```
$ export OMP_NE                                  NESTED=FALSE
$ ./experimentN                          tN
L 1: # threads                  s in team 2
L 2: # threads in team 2                          n team 1
L 2: # threads in team 2                          n team 1
L 3: # threads in team 2                          n team 1
L 3: # threads in team 2                          n team 1
L 3: # threads in team 2
L 3: # threads in team 2
```

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

OpenMP Environment Variables

- OMP_SCHEDULE=algorithm
  - dynamic[, n]
  - guided[, n]
  - Runtime
  - **static[, n]**
- OMP_NUM_THREADS=num
- OMP_NESTED=TRUE|FALSE
- OMP_DYNAMIC=TRUE|FALSE

Assignment Project Exam Help
https://eduassistpro.github.io/
Add WeChat edu_assist_pro

Outline

- Mutlicore Shared-Memory Platforms
- OpenMP – API for Shared-Memory Multi-processing
  - Overview
  - Example: numerical integration
  - Shared variable
  - SPMD vs worksharing
  - Data environment options
  - Advanced worksharing options
  - Synchronization options
- **How is This Relevant to Writing Fast Code?**

Assignment Project Exam Help
https://eduassistpro.github.io/
Add WeChat edu_assist_pro

k-means Algorithm Concurrency

1. **Initialization:** Randomly select *k* cluster centers
2. **Expectation:** Assign closest center to each data point
   - N (independent)
   - k (min re
   - D (sum re
3. **Maximization:** Update centers nments
   - D (independent)
   - N (Histogram computation into k bins)
4. **Evaluate:** Re-iterate steps 2-3 until convergence

Assignment Project Exam Help
https://eduassistpro.github.io/
Add WeChat edu_assist_pro

Project 1: Search Among Alternatives

- What are the implementation **alternatives**?
  - Different mapping of **application concurrency** to **platform parallelism**

Assignment Project Exam Help

- The **search** process
  - More complex **platform parallelism**
  - May want to se
    - Some parallel operations are as **"wor** uential operations
  - One level of concurrency could map to arallelism

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

How is this relevant to writing fast code?

- **Multicore p** **ctures**
- **Manycore** **s**
- **Cloud platf** **Architecture**

Assignment Project Exam Help
https://eduassistpro.github.io/
Add WeChat edu_assist_pro

- This lecture: **An abstraction for multico** **amming**

- **Abstractions:**
  - Help establish a mental model for the programmers
  - Make it easier to write parallel code
  - **Performance depend on deep understanding of the implementation platform**