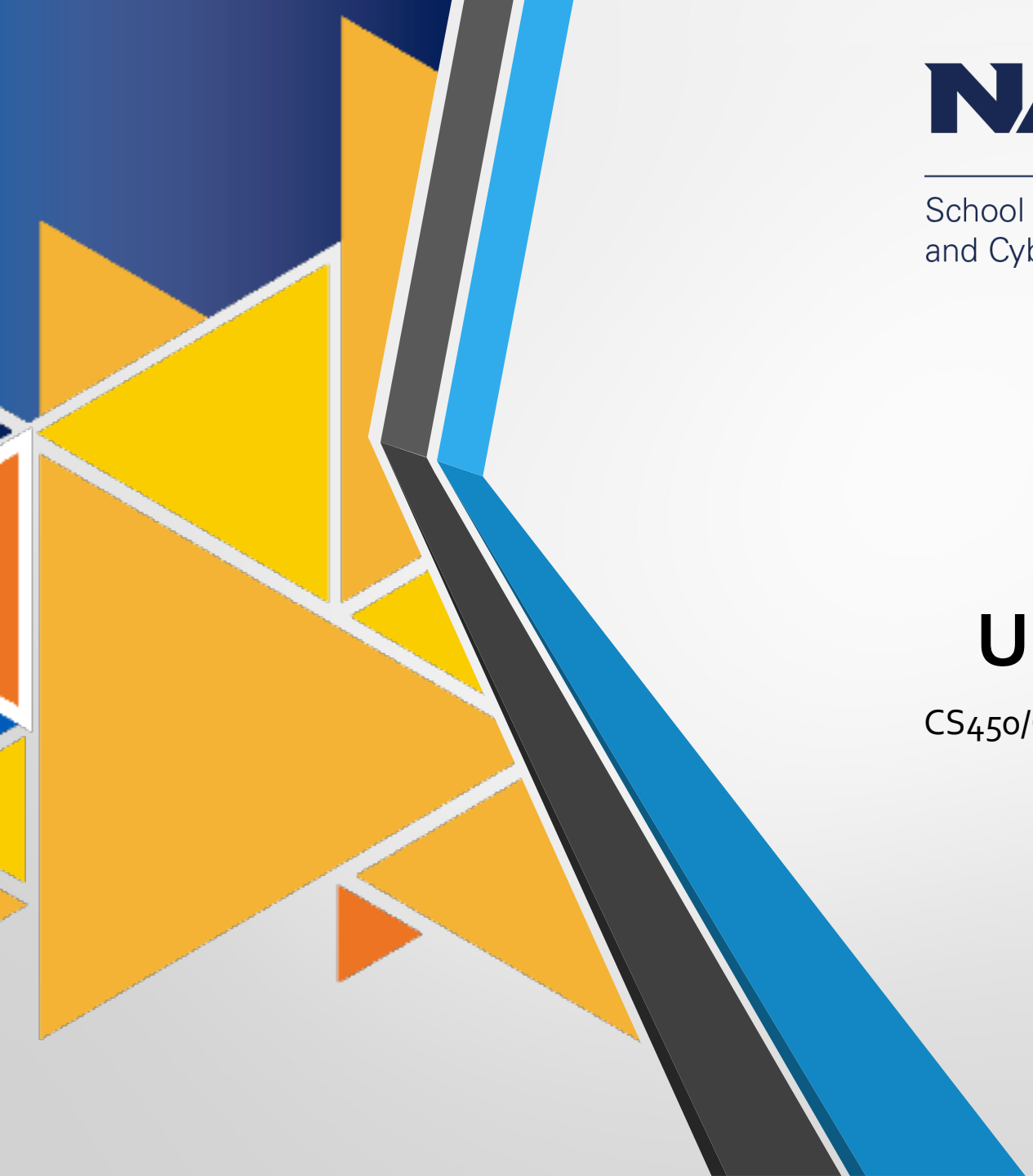


NAU NORTHERN ARIZONA UNIVERSITY

School of Informatics, Computing, and Cyber Systems

Using OpenMP

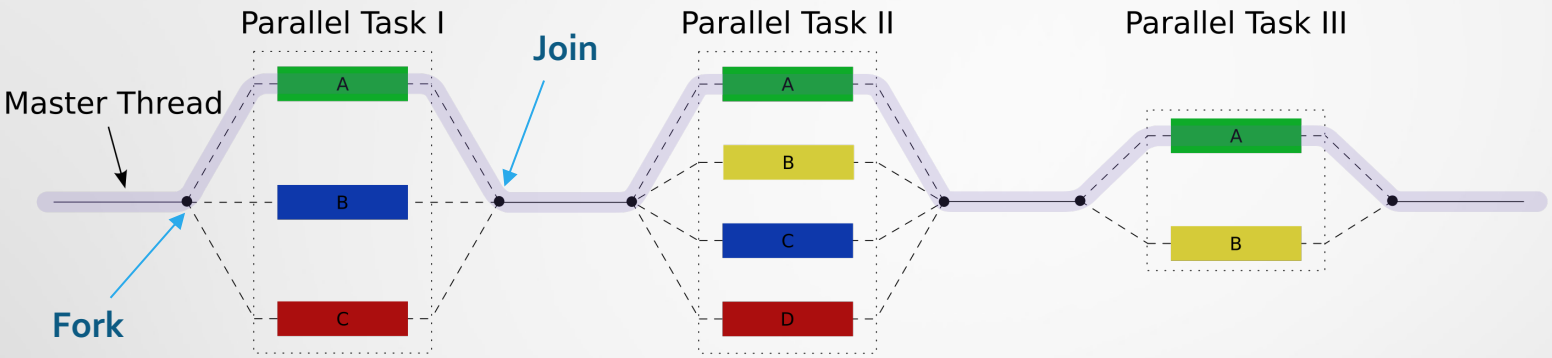
CS450/CS550: Parallel Programming

Week 9

OpenMP

- OpenMP (Open Multi-Processing) is an API that supports shared-memory multiprocessing programming in C/C++ and Fortran
- OpenMP defines of a set of compiler **directives**, run time **routines**, and **environment variables** that influence run-time behavior
- OpenMP is managed by the nonprofit technology consortium OpenMP Architecture Review Board (OpenMP ARB)
- The last stable OpenMP release was published in November 2021
- The OpenMP specification: https://www.openmp.org/spec-html/5.2/openmp.html
- The official web site: https://www.openmp.org/
- To use OpenMP in C, the `omp.h` header should be included, and the `-fopenmp` option should be added when compiling a program

OpenMP execution model



Source: https://en.wikipedia.org/wiki/OpenMP

- A program begins with a Master Thread
- Fork: groups of threads created during the program execution
- Join: threads in the team synchronize (barrier) and only the master thread continues execution

OpenMP directives

- The syntax for the OpenMP directives

```
#pragma omp directive [clause list]
{
    // the code to be executed in parallel
}
```

- `pragma` stands for "pragmatic information"
- a program executes serially until it encounters the `parallel` directive
- a thread that encounters the `parallel` directive becomes the **master** of the group of threads and is assigned the thread ID `0` within the group

Some clauses in the OpenMP directives

- the clause `if (expression)` determines whether the parallel threads will be created
- the clause `num_threads(n)` specifies the number `n` of threads to be created
- the `private(list)` indicates the list of variables to be local to each of the threads
- the `shared(list)` indicates the list of variables to be shared by all threads
- the `default({private | shared | none})` indicates the default state of variables in a thread; `none` implies that the state of each variable must be explicitly specified

Some OpenMP functions

- `int omp_get_num_procs()`: returns the number of processor the program can use
- `int omp_get_num_threads()`: returns the number of threads that are currently active
- `int omp_get_thread_num()`: returns the thread ID
- `int omp_in_parallel()`: returns `1` if it was called inside the parallel block, or `0` otherwise
- `void omp_set_num_threads(int n)`: sets the `n` number of parallel threads for their execution in parallel regions

OpenMP environment variables

- To set an OpenMP environment variable `OMP_VAR` in csh-like shells:

```
setenv OMP_VAR "value"
```

- To set an OpenMP environment variable in bash-like shells:

```
export OMP_VAR="value"
```

- Last version of the OpenMP specification contains 17 environment variables
- The variable `OMP_NUM_THREADS` sets the number of threads to use for parallel regions

Code example: sum of array

The `critical` directive

- The syntax of the `critical` directive:

```
#pragma omp critical [name]
{
    // the code to be executed
}
```

- the optional *name* is used to identify a critical region
- the `critical` directive ensures that at any point in the execution of the program only one thread may run the critical section; other threads are waiting until the current thread is done
- the `critical` directive is an application of the mutex function in Pthreads

The `atomic` directive

- If a critical section consists of an update to a single memory location, the `atomic` directive may be used:

```
#pragma omp atomic [name]
    // the atomic instruction
```

- The `atomic` directive specifies that the memory update in the following line should be performed as an atomic operation
- Possible update instructions for the `atomic` directive:

```
x binary_operation = expr
x++, ++x, x--, --x
```

The *`binary_operation`* is one of `{+,*,-,/,&,||,<<,>>}`

The `for` directive

- The `for` directive precedes a `for` loop with independent iterations that may be divided among threads executing in parallel:

```
#pragma omp for [clause list]
    // for loop
```

- The iterations inside of the `for` loop are divided between threads automatically
- The `for` directive may be combined with the `parallel` directive:

```
#pragma omp parallel for [clause list]
    // for loop
```

The `reduction` clause

- The `reduction` clause is used in the `parallel` directive:

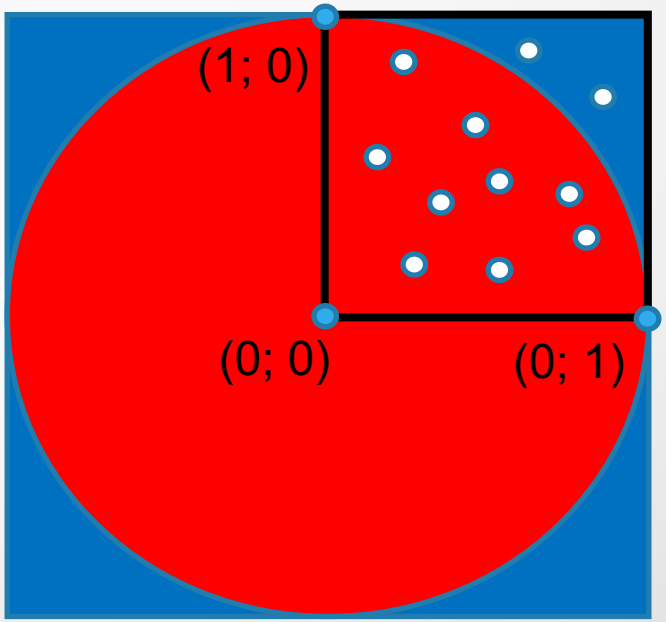
`#pragma omp parallel` `[clauses]` **`reduction`**`(op: var)`

- The **reduction variable** *`var`* (should be shared) and the **reduction operator** *`op`* must be specified for the `reduction` clause

Operator	Operation	Allowable types	Initial value
+	sum	float, int	0
*	product	float, int	1
&	bitwise and	int	all bits 1
|, ^	bitwise or, xor	int	0
&&	logical and	int	1
||	logical or	int	0

Calculating π with Monte Carlo method

- For some big number of N generated points calculate the number M of points located inside the circle:
  - generate random coordinates (x, y) of the points in the range $[0; 1]$
  - check if the point is located inside the circle: $x^2 + y^2 \leq 1$
- Then $\frac{M}{N} \approx \frac{S_c}{S_s}$, where $S_c = \pi \cdot r^2$ is the area of the circle, $S_s = 4 \cdot r^2$ is the area of the square, $r = 1$
- Hence, $\frac{M}{N} \approx \frac{\pi \cdot r^2}{4 \cdot r^2} \Longrightarrow \pi \approx 4 \cdot \frac{M}{N}$



Code example: calculating π

Assignment #3

- By using the Monte-Carlo method and OpenMP, develop the method that calculates the cumulative distribution function of the variable with standard normal distribution for the given bounds a and b:

$$F(x) = \frac{1}{\sqrt{2\pi}} \cdot \int_{-\infty}^{x} e^{-\frac{1}{2} \cdot x^2} dx$$

$$F(b) - F(a) = \frac{1}{\sqrt{2\pi}} \cdot \int_{a}^{b} e^{-\frac{1}{2} \cdot x^2} dx$$

Assignment #3: standard normal distribution

Probability density function:

$$f(x) = \frac{1}{\sigma \cdot \sqrt{2\pi}} \cdot exp\left[-\frac{1}{2} \cdot \left(\frac{x-\mu}{\sigma}\right)^2\right]$$

Standard distribution ($\mu = 0, \sigma = 1$):

$$f(x) = \frac{1}{\sqrt{2\pi}} \cdot exp\left[-\frac{1}{2} \cdot x^2\right]$$

$$f(0) = \frac{1}{\sqrt{2\pi}} \cdot exp\left(-\frac{1}{2}\right) = 0.3989$$