CS 3410 Lab 12

Fall 2025

Agenda

1. Threads Overview
2. Amdahl’s Law
3. Circular (Ring) Buffer
4. Making Threads in C
5. Assignment 12 Tips

Threads Overview

Threads overview

A *thread* is an execution state within a process

Key Features

- **Shared Memory** - threads communicate easily via heap data
- **Multiple Stacks** - each thread has its own stack, but operates *within the same memory space*

Why threads?

- **Concurrency** - Handle overlapping tasks (e.g., web browsers)
- **Parallelism** - Leverage multiple CPU cores for faster computation

Amdahl's Law

Amdahl’s Law

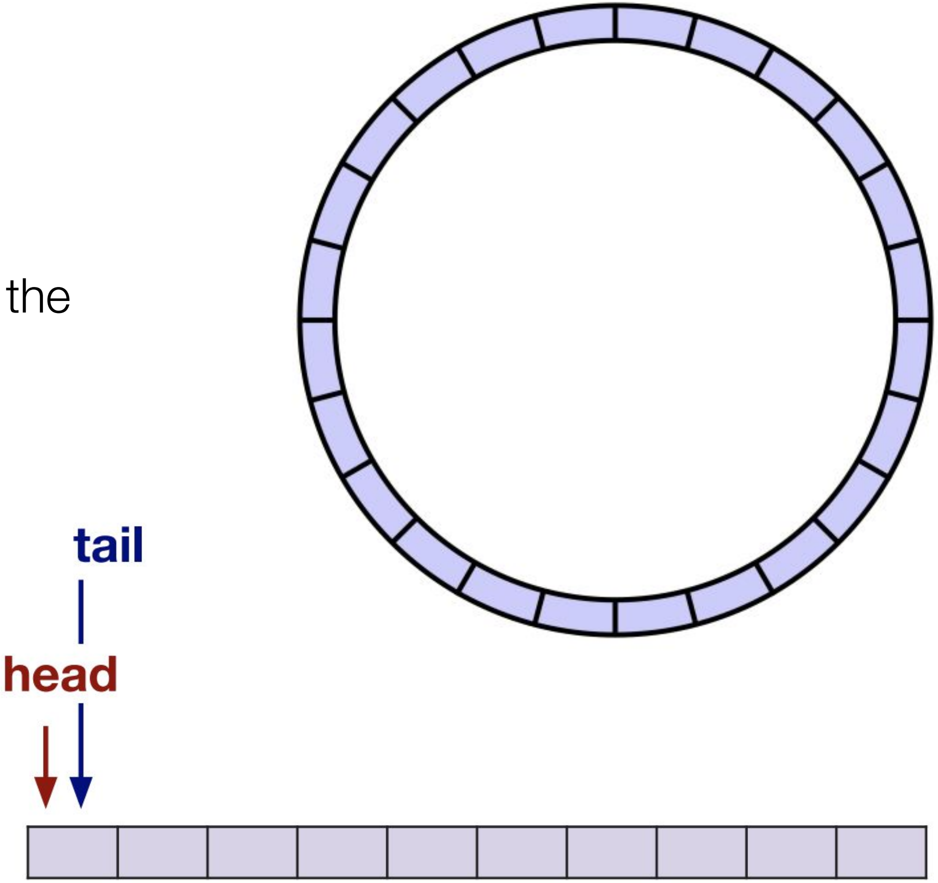
- Gives you the **theoretical maximum speedup** over sequential execution.
- **p**: Fraction of time originally spent in the part of code you are now parallelizing
- **n**: Number of threads/workers

$$\text{speedup} = \frac{1}{(1 - p) + \frac{p}{n}}$$

Circular (Ring) Buffer

Circular (Ring) Buffer

- Implemented with an array
- **Producer threads** push tasks to the tail pointer
- **Consumer threads** pull tasks from the head pointer
- What about when we reach the end of the array?
  - Wrap pointer around with modulus



Making Threads in C

Steps to making threads in c

Include <pthread.h> and follow these steps!

1. Create a struct for thread arguments	2. Define the thread function	3. Prepare arguments and spawn thread	4. Join threads and wait for completion

`typedef struct {`
`    int value1;`
`    ...`
`} ThreadArgs;`
	`void* thread_func(void* arg)`
	`{`
	`  ThreadArgs* args =`
	`           (ThreadArgs*)arg;`
	`  ... // do stuff`
	`}`
		`ThreadArgs args =`
		`      {.value = 1, ...};`
		`pthread_t thread;`
		
		`pthread_create(&thread,`
		`   NULL, thread_func,`
		`   &args);`
			`pthread_join(`
			`  thread, NULL);`

Assignment 12 Tips

Assignment Tips

- Assignment 12 is **optional**
  - It **cannot** replace previous assignments.
- A12 content will still be tested on the exam!
- Read through *all the provided files*!
  - Make sure you understand the data structures!
- Start with the *sequential* implementation!
  - Use this to compare to other implementations!
- Review the parallel programming lecture notes!
- Look through the pthread documentation!
- Submissions/autograder open until *December 8*.

Good Luck!