CS4414 Recitation 1
Course Introduction and C++ Setup

08/27/2021
Sagar Jha, Alicia Yang
About TA  ---  Sagar

• Senior PhD student in CS

• Advised by Prof. Birman on distributed systems (with a focus on RDMA networks)

• TA experience at Cornell
  • Practicum in Database Systems (Fall ‘16)
  • Cloud Computing (Spring ‘18, Spring ‘20)
  • Systems Programming (Fall ‘20, Fall ‘21)

• Office Hours
  • Thursday and Friday 5-7 pm (starting next week)
  • At https://cornell.zoom.us/j/99522656755?pwd=WTdzV1hFSzVIM1BLR0Q3TDZsRHzKd09
Goals

Develop systems in C++ that perform well
Goals

Develop **systems** in C++ that **perform well**

For the recitation:

- **Basic C++ proficiency**: Read, write, and debug C++ code
- **Working knowledge of Linux**: The Linux command line and the filesystem
Goals for the recitation

• **Basic C++ proficiency**: Read, write, and debug C++ code
  • Standard containers – `std::vector<T>`, `std::map<K, V>`
  • pointers, iterators, templates, classes...
  • `gdb` for debugging, `gprof` for profiling
  • multi-threading, synchronization

• **Working knowledge of Linux**: The Linux command line and the filesystem
Secondary goals

- Learn to characterize code performance
- Make efficient use of hardware – learn to exploit CPU cores with threads
- Understand solutions to assignments/exams
System Performance will be a mainstay of this course!

What do we mean by performance?

- Latency: Time taken to compute
- Throughput: Number of operations per second
Focus on system performance

It’s not just about algorithm complexity. Why?
Reasoning about system performance

• Theoretical improvements don’t always translate to better application runtimes

Insertion sort outperforms quick sort in some cases

Why?
1. Insertion sort is iterative – no overhead from recursive function calls (good for sorting a small set)
2. Insertion sort is fast when data is nearly sorted
Reasoning about system performance

- Theoretical improvements don’t always translate to better application runtimes
- Which algorithm? A system can be very complex with many features

![Diagram of a sequential program with 2 steps]

- Fairly optimized code
- Highly inefficient code

- A = processing files, B = printing 1 million lines of output
Reasoning about system performance

• Theoretical improvements don’t always translate to better application runtimes

• Which algorithm? A system can be very complex with many features

What if step A takes about 99% of the total time? We need to profile and understand performance characteristics of code we write!
Reasoning about system performance

• Theoretical improvements don’t always translate to better application runtimes
• Which algorithm? A system can be very complex with many features
• What if the code that implements the algorithm is inefficient?
• Sometimes heuristics work better
C++ Environment Setup
<table>
<thead>
<tr>
<th>Name</th>
<th>Office Hour 1</th>
<th>Office Hour 2</th>
<th>Zoom Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alicia Yang</td>
<td>Saturday 5-7PM</td>
<td>Sunday 6-7PM</td>
<td><a href="https://cornell.zoom.us/j/93560684279?pwd=S1c0NF5Y1ZLNnpVVU0xQII5K2tRUT09">https://cornell.zoom.us/j/93560684279?pwd=S1c0NF5Y1ZLNnpVVU0xQII5K2tRUT09</a></td>
</tr>
<tr>
<td>Andrew D.</td>
<td>Tuesday 9-11AM</td>
<td>Thursday 10-11AM</td>
<td><a href="https://cornell.zoom.us/j/91466505032?pwd=U01wTEqvTE0T3NWcDdMeWMrdW1Zdz09">https://cornell.zoom.us/j/91466505032?pwd=U01wTEqvTE0T3NWcDdMeWMrdW1Zdz09</a></td>
</tr>
<tr>
<td>Zheng Wang</td>
<td>Monday 1-3PM</td>
<td>Wednesday 10-11AM</td>
<td><a href="https://cornell.zoom.us/j/8812491232?pwd=VUtRWndqR2lvMjU1S1VZVkJ5VkrXdz09">https://cornell.zoom.us/j/8812491232?pwd=VUtRWndqR2lvMjU1S1VZVkJ5VkrXdz09</a></td>
</tr>
<tr>
<td>Sagar Jha</td>
<td>Thursday 5-7PM</td>
<td>Friday 5-7PM</td>
<td><a href="https://cornell.zoom.us/j/99522656755?pwd=WTdzV1hFSzVIM1BLR0Q3TDZsRHdKdz09">https://cornell.zoom.us/j/99522656755?pwd=WTdzV1hFSzVIM1BLR0Q3TDZsRHdKdz09</a></td>
</tr>
<tr>
<td>Aahli Awatramani</td>
<td>Wednesday 9-10AM</td>
<td>Wednesday 2-4PM</td>
<td><a href="https://cornell.zoom.us/j/92630999231?pwd=b0RkeVQ5TWcwcWd1WgdLbXnpT21MQT09">https://cornell.zoom.us/j/92630999231?pwd=b0RkeVQ5TWcwcWd1WgdLbXnpT21MQT09</a></td>
</tr>
<tr>
<td>Arthur Tanjaya</td>
<td>Tuesday 5-7PM</td>
<td>Thursday 5-6PM</td>
<td><a href="https://cornell.zoom.us/j/3877784348?pwd=dkvwcFBwS1RDSHh2SXhRXZVaVtdzd09">https://cornell.zoom.us/j/3877784348?pwd=dkvwcFBwS1RDSHh2SXhRXZVaVtdzd09</a></td>
</tr>
<tr>
<td>Muhammad Moughal</td>
<td>Monday 6-7PM</td>
<td>Tuesday 4-6PM</td>
<td><a href="https://cornell.zoom.us/j/4905170673?pwd=V3dXS00wEFieC9YSDBG53Z4UTR5Zz09">https://cornell.zoom.us/j/4905170673?pwd=V3dXS00wEFieC9YSDBG53Z4UTR5Zz09</a></td>
</tr>
</tbody>
</table>
About TA --- Alicia

• 1st year PhD student in CS, TAed this course Fall 2020
• Working with Prof. Birman in the area of distributed system
• Interested in scheduling and cluster management in machine learning system
• Office Hours: Thursdays 6PM – 7PM, Saturdays 6PM - 7PM
• Meeting by appointment for questions or assignment discussion
C++ Coding Environment

- C++ 20
- gcc-8 or recent
  - To check your gcc compiler version:  $ g++ -v
C++ Coding Environment

• Editing Tools:
  • Visual Studio Code ([link](link))
  • Emacs
  • Vi, …

• Compilation Tools: GNU Compiler Collection (GCC)
  • Cornell Engineering linux server remote access
  • Virtual Box
  • Most linux distributions have GCC
  • MacOS user has Clang compiler ([not recommended for this course](not recommended for this course), since Clang and GCC are two different compilers and sometimes have different compilation results. The submitted assignments are run via GCC)
C++ Coding Environment

• C++ 20

• gcc-8 or recent
  • To check your gcc compiler version: $ g++ -v

• We will introduce three main ways of setting up the coding environment

https://visualstudio.microsoft.com
C++ Coding Environment

- C++ 20
- gcc-8 or recent
  - To check your gcc compiler version: $ g++ -v
- We will introduce three main ways of setting up the coding environment
C++ Environment Setup

**method1:** compile & run on cornell engineering linux server from terminal

- Install Visual Studio Code
- Install C/C++ extension in VSCode
C++ Environment Setup

method1: compile&run on cornell engineering linux server from terminal

• Install VSCode and C++ extension on VSCode

• Login to Cornell VPN
C++ Environment Setup

method1: compile & run on cornell engineering linux server from terminal

- Install VSCode and C++ extension on VSCode
- Login to Cornell VPN
- ssh to your cornell student account on VSCode through
  - View -> Command Palette -> Remote SSH: Connect to host
  - In command type:
    
    % ssh [your netid]@ugclinux.cs.cornell.edu

C++ Environment Setup

**method1:** compile & run on cornell engineering linux server from terminal

- Install VSCode and C++ extension on VSCode
- Login to Cornell VPN
- `ssh` to your cornell student account on VSCode through
- All set! Start coding (Demo)
  - Helloworld simple program
    - Compile: `g++ -std=c++20 -Wall -o helloworld helloworld.cpp`
    - Run: `./helloworld`
  - If there are multiple files, compile with: `g++ -std=c++20 main.cpp other.cpp etc.cpp`
1. Install IDE (VSCode in this example), and extensions: C/C++, Code Runner
   - Specify C++ standard version on Code runner extension
C++ Environment Setup

method2 : VSCode Edit, compile/run locally

1. Install IDE (VSCode in this example), and extensions: C/C++, Code Runner

   - Add `-std=c++17` to the json file:

     "cpp": "cd $dir && g++ -std=c++17 $fileName -o $fileNameWithoutExt && $dir$fileNameWithoutExt"
C++ Environment Setup

**method 2:** VSCode Edit, compile/run locally

1. Install IDE (VSCode in this example), and extensions: C/C++, Code Runner

2. Install Compiler
   - Why install gcc? mac default C++ compiler is Clang
1. Install IDE (VSCode in this example), and extensions: C/C++, Code Runner

2. Install GCC Compiler with following command
   - % brew update
   - % brew upgrade
   - % brew info gcc
   - % brew install gcc
   - % brew cleanup
C++ Environment Setup

method 2: VSCode Edit, compile/run locally

• Install IDE (VSCode in this example), and extensions: C/C++, Code Runner

• Install GCC Compiler with following command

• Run and debug locally:
  • Control + shift + D
1. Install IDE (VSCode in this example), and extensions: C/C++, Code Runner

2. Install GCC Compiler with following command

3. Run and debug locally

4. Configure compiled file
   - .vscode/tasks.json
C++ Environment Setup

- **method3 (windows):** Visual Studio Edit, compile/run locally

  • Download Visual Studio

  • Configure GCC property on Visual Studio
C++ Environment Setup

- method3 (windows): Visual Studio Edit, compile/run locally
  
  • Download Visual Studio
  
  • Configure GCC property on Visual Studio
  
  • Create C++ Project, right click project -> build
C++ Environment Setup

- method3 (windows): Visual Studio Edit, compile/run locally

- Download Visual Studio

- Configure GCC property on Visual Studio

- Create a new project
C++ Environment Setup

- **method3 (windows):** Visual Studio Edit, compile/run locally

  • Download Visual Studio

  • Configure GCC property on Visual Studio

  • Create C++ Project, right click project -&gt; build

  • Run the executable
    
    • Click the .exe

    • Click on localWindowDebugger on Visual Studio
C++ Environment Setup

- method3 (windows): Visual Studio Edit, compile/run locally