CSC 360
Operating Systems
Introduction

Wenjun Yang

https://wenjun-y.github.io/csc360

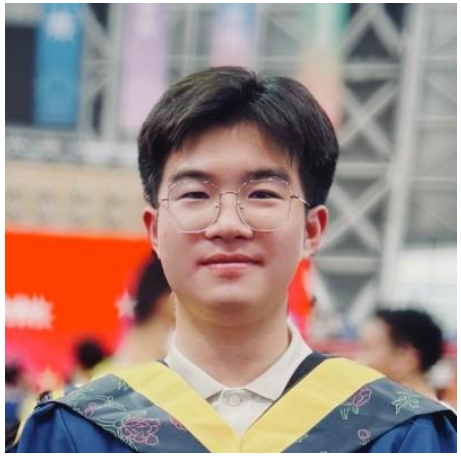
Fall 2025

Instructor

- Dr. Wenjun Yang <wenjunyang@uvic.ca>
  - teacher-scholar & postdoc with CS
  - use UVic Teams first, as email not always reliable
    - to help, always include **[csc360]** in your email subject line
  - office hours: **MR 10:15--11:15am**
    - or by appointment
    - **on UVic Teams**
  - research area
    - Networked system for distributed AI
    - https://wenjun-y.github.io/

CSC360's Mighty TAs

Jinwei Zhao



Quanwei Zhang

Johnathan Warawa

Lab/tutorial instructors

- Jinwei Zhao, Quanwei Zhang, and Johnathan
  - their email on Brightspace
- Tutorials: start next week!
  - Please attend your registered section only!
    - tutorial lectures
      - **C**, libc, sockets, pthreads, ...
    - assignment help
      - spec go-through, common problems, ...
    - practice problems

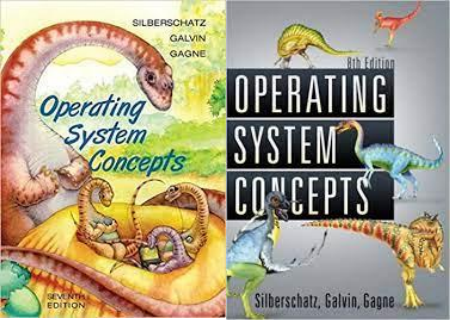
T01: MAC D115 T 8:30am
T02: COR B129 W 1:30pm
T03: DSB C118 W 11:30am

About the course

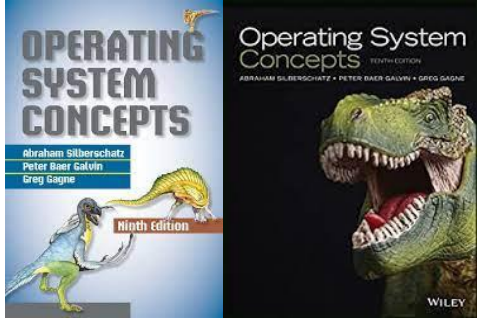
- *Introduction* to **Operating Systems**
  - (A01/2) **MR 8:30--9:50am**, **DSB C118**
  - (A03/4) **11:30am--12:50pm, CLE A224**
  - Bright: “Fall 2025 CSC 360 A01 - A04 **X**”
    - assignments, gradebook, etc
    - **discussion channel hosted on UVic Teams**
  - prerequisites
    - Data structures (CSc **225** or 226)
    - Computer architecture (CSc **230** or CENG 255)
    - System programming (CSc/SENG **265**, CENG)

Message from Undergrad Advisor

- Email: cscadvisor@uvic.ca
- Do not have the prerequisite course(s)?
  - need to obtain a waiver
  - otherwise, prerequisite drop after the first week
- Taking the course more than twice?
  - need to have a letter from the Chair and the Dean
  - otherwise, being dropped from the class
- Make sure you can receive email from Bright!
  - use UVic Teams first



Course materials

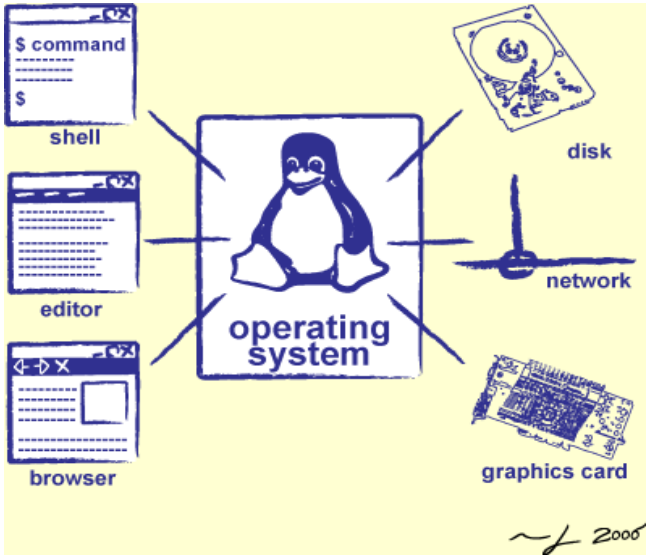


- Required textbook
  - Operating system concepts, 7th or **newer** editions
    - 6th and other editions: different chapter schedules
  - online resources

    http://codex.cs.yale.edu/avi/os-book/
    or **http://os-book.com/**
    - errata, slides, practice exercises and solutions
- Explore further
  - Google!

FUN: why dinosaurs on the book cover?

Goals for Today



- Why should you care?
- Why is it hard?
- What is an Operating System?
- Policy/advice/challenges/opportunities

Goal 1: Why should you care?

Goal 1: Why should we care?

- The OS is everywhere
  - Every device, from your smartwatch, your smart light bulb, to your mobile phone and laptop runs an operating system
  - Every program you will ever write will run on an operating system
  - Its performance and execution behavior will depend on the operating system

Goal 2: Why is designing an OS hard?

Diverse OSs needed

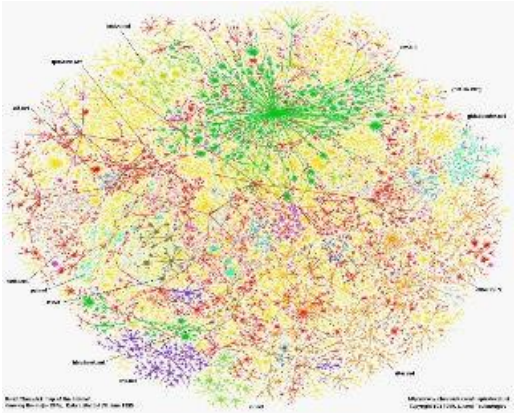




- Examples
  - **Linux**/Unix, MacOS, Windows, and many others
  - -- **Android**, iPhone iOS, Symbian, etc

Across many devices

Have an operating system



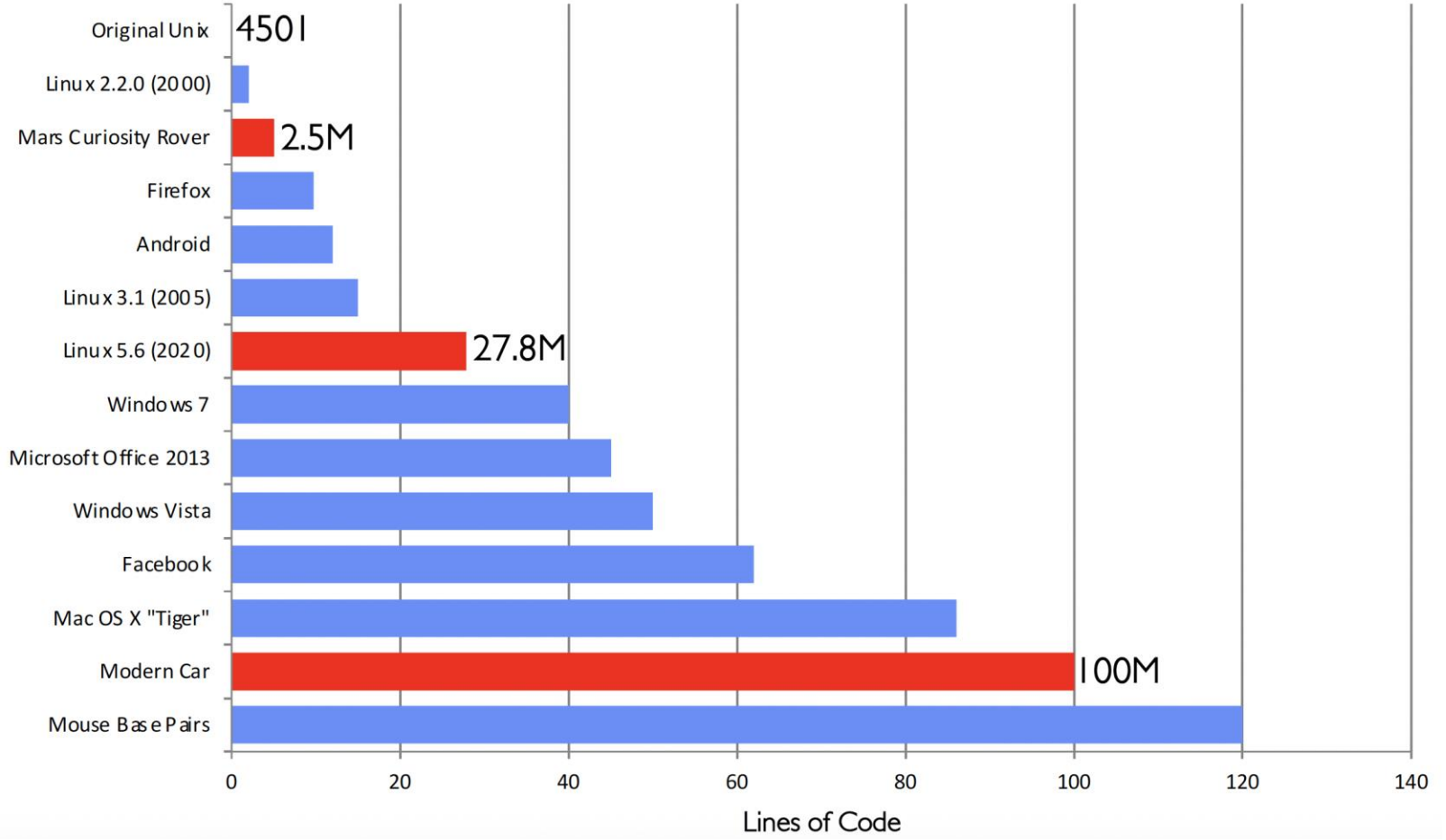
Communicate over the Internet

Interface across huge diversity of devices

Across many timescales

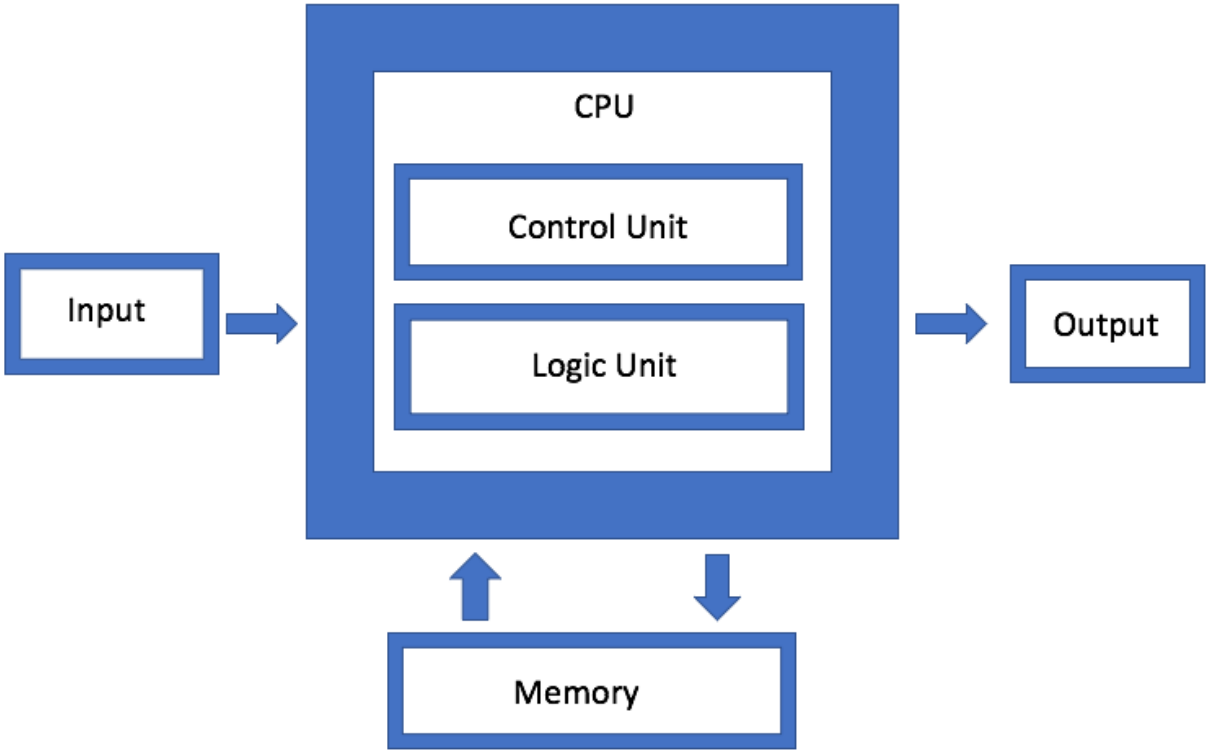
L1 cache reference	0.5 ns
Branch mispredict	5 ns
L2 cache reference	7 ns
Mutex lock/unlock	25 ns
Main memory reference	100 ns
Compress 1K bytes with Zippy	3,000 ns
Send 2K bytes over 1 Gbps network	20,000 ns
Read 1 MB sequentially from memory	250,000 ns
Round trip within same datacenter	500,000 ns
Disk seek	10,000,000 ns
Read 1 MB sequentially from disk	20,000,000 ns
Send packet CA->Netherlands->CA	150,000,000 ns

With increased complexity



Why so much complexity?

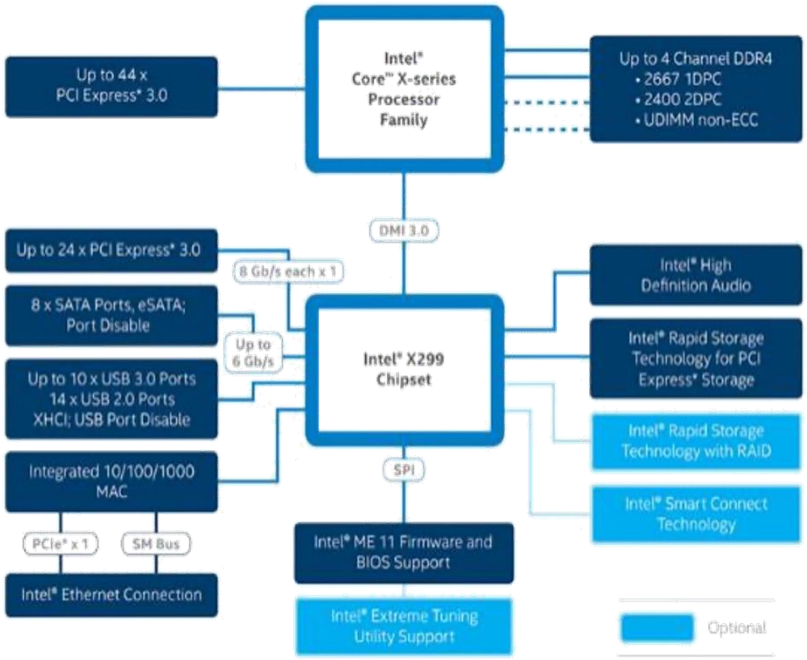
Started with a simple architecture (Von Neumann Architecture)…



John von Neumann
(1903 – 1957)

Why so much complexity?

- Hardware is becoming smarter!
- Better reliability and security
- Better performance (more efficient code, more parallel code)
- Better energy efficiency



Goal 3: What is an Operating System?

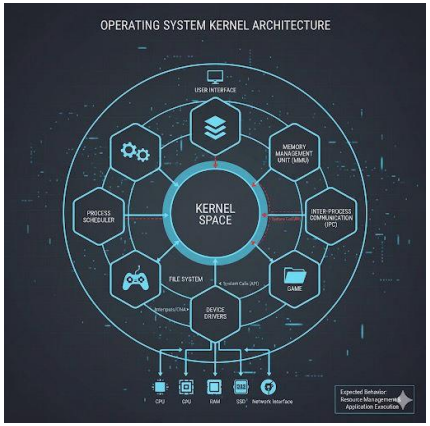
What's operating system?

Operating

System

Manages multiple tasks and users

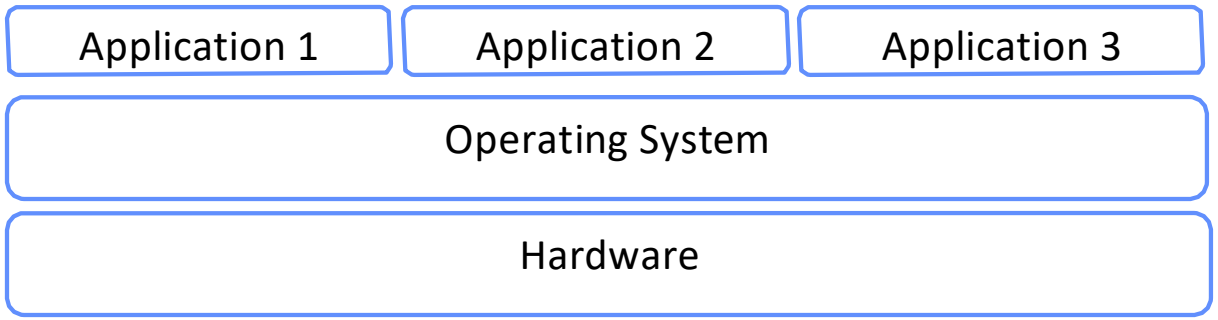
A set of interconnected components with an expected behavior observed at the interface with its environment



Generated by Gemini 2.5 Proc

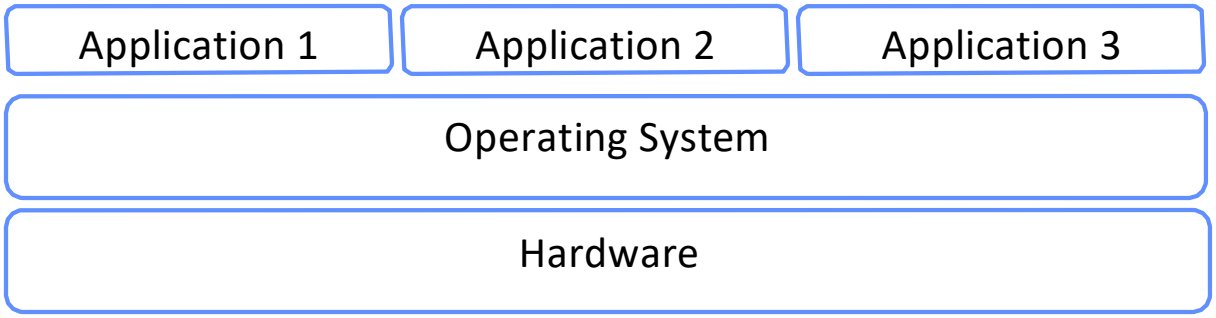
Operating System (v1)

An operating system is the layer of software that interfaces (many) applications running on a machine with (diverse) hardware resources of that machine



Operating System (v2)

An operating system implements a virtual machine for the application whose interface is more convenient than the raw hardware interface (convenient = portability, reliability, security)

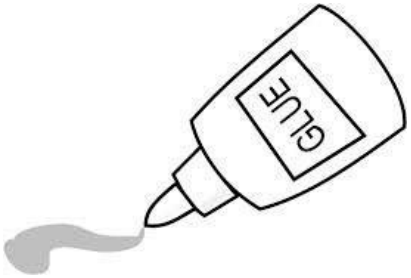


Still hard to understand what's OS?

Roles of OS

Referee
Manage protection, isolation, and sharing of resources

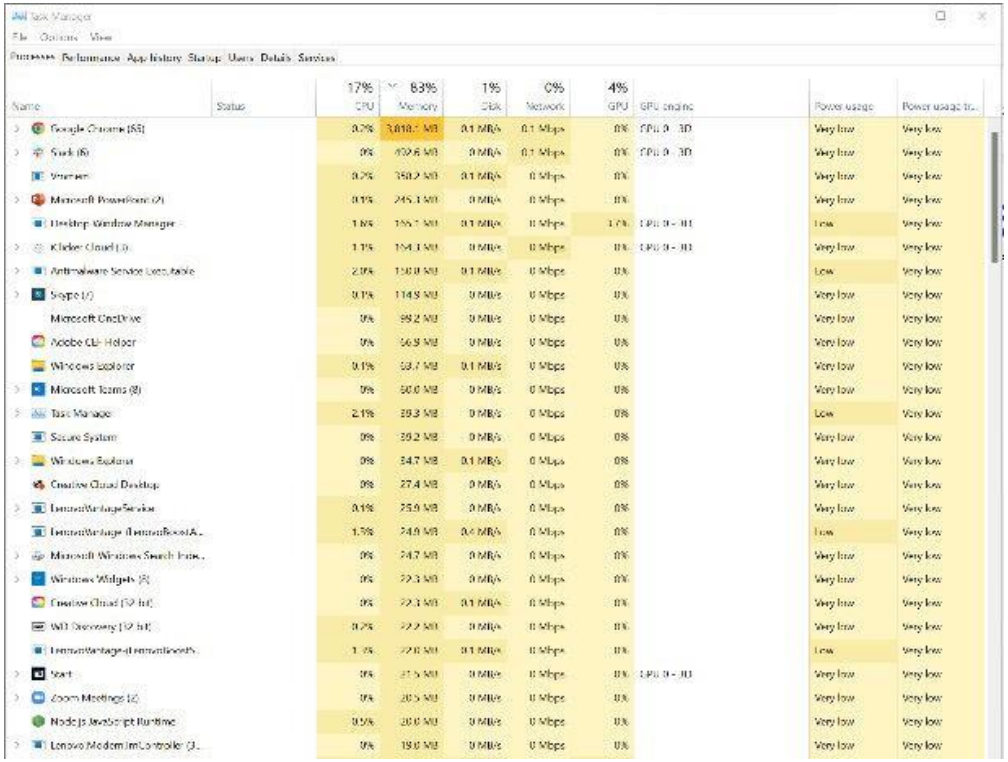
Illusionist
Provide clean, easy-to-use abstractions of physical resources



Glue
Provides a set of common services

OS as a referee

Allow multiple (untrusted) applications to run concurrently



OS as a referee

Fault Isolation

Isolate programs from each other

Isolate OS from other programs

Process

Dual Mode Execution

Resource Sharing

How to choose which task to run next?

How to split physical resources?

Scheduling

Communication

How can OS support communication to share results?

Pipes/Sockets

What does this program do?

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/time.h>
#include <assert.h>

int main(int argc, char *argv[]){
  char *str = argv[1];
  while (1) {
    printf("%s\n", str);
  }
  return 0;
}
```

```
ion@laptop> gcc –o cpu cpu.c -Wall
ion@laptop> ./cpu A
A
A
A
A
…
```

```
ion@laptop> ./cpu A & ./cpu B & ./cpu C
```

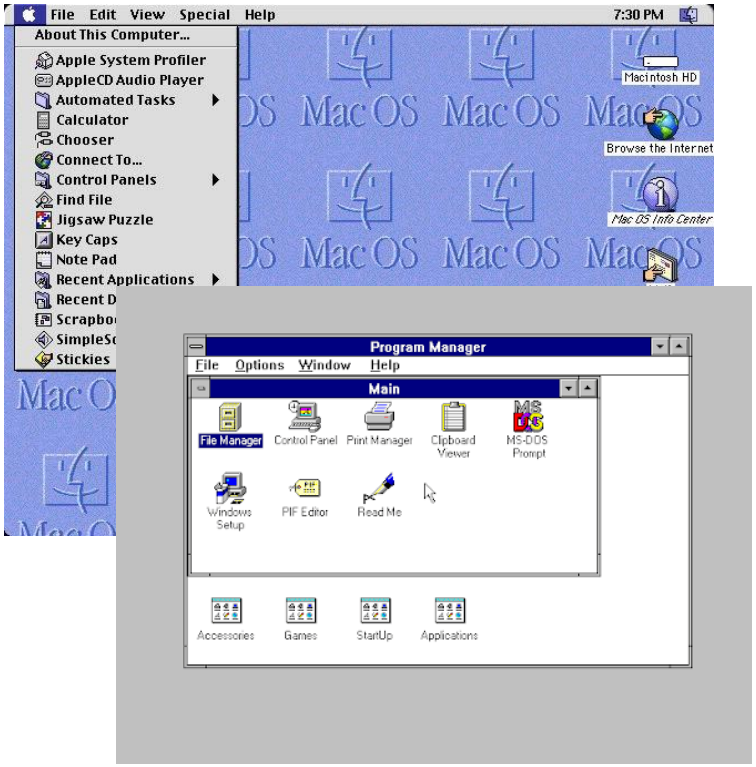
a)
```
A
A
A
A
…
```

b)
```
A
B
C
A
B
C
…
```

c)
```
A
B
B
A
C
B
C
…
```

```
ion@laptop> ./cpu & ; ./cpu B
Segmentation Fault
B
…
```

Refereeing is hard!



Up to MacOS 9x and Windows 3.1

OS cannot force program to give up control!

```
ion@very-old-laptop>
./cpu A & ./cpu B & ./cpu C

A
A
A
A
A
A
A
...
```

Three main roles

Referee

Manage protection, isolation, and sharing of resources

Illusionist

Provide clean, easy-to-use abstractions of physical resources

OS as Illusionist

Mask the restrictions inherent in computer hardware through virtualization

All alone

Provide illusion that application has exclusive use of resources

All powerful

Provide illusion that hardware resources are infinite

All expressive

Provide illusion of hardware capabilities that are not physically present

What does this program do?

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

int main(int argc, char *argv[]){
  int *p = malloc(sizeof(int));
  printf("(%d) p: %p\n", getpid(), p);
  *p = 0;
  while (1) {
    *p = *p + 1;
    printf("(%d) p: %d\n", getpid(), *p);
  }
  return 0;
}
```

```
ion@laptop> gcc -o memory memory.c -Wall
ion@laptop> ./memory
(120) p: 0x200000
(120) p: 1
(120) p: 2
(120) p: 3
(120) p: 4
ion@laptop> ./memory & ./memory
(120) p: 0x200000
(254) p: 0x200000
```

a)

```
(120) p: 1
(254) p: 2
(120) p: 3
(254) p: 4
(120) p: 5
(254) p: 6
…
```

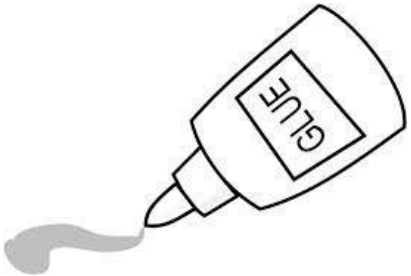
b)

```
(120) p: 1
(254) p: 1
(120) p: 2
(254) p: 2
(120) p: 3
(254) p: 3
…
```

Three main roles

Referee
Manage protection, isolation, and sharing of resources

Illusionist
Provide clean, easy-to-use abstractions of physical resources



Glue
Provides a set of common services

OS as Glue

Provide set of common, standard services to applications to simplify and regularize their design

Make sharing easier

Simpler if all assume same basic primitives

Maximise reuse

Avoid re-implementing functionality from scratch.

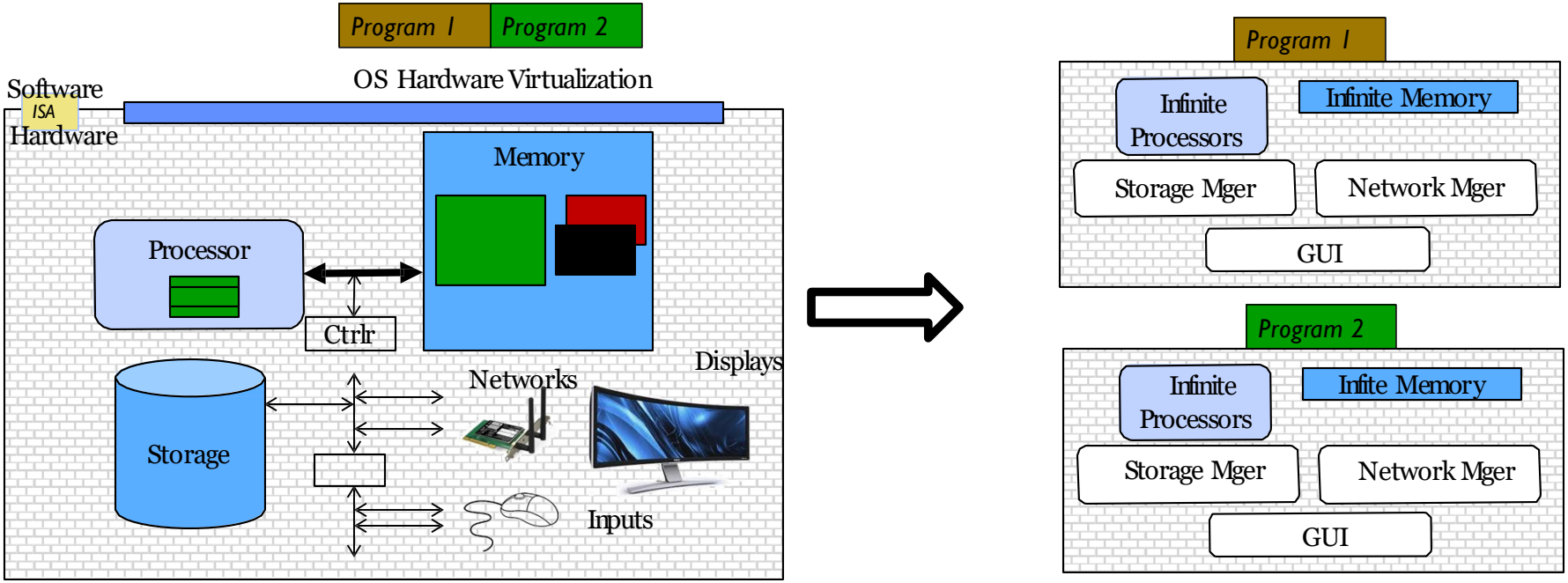
Evolve components independently

File System, User Interface, Network, etc.

Putting it all together

Referee + illusionist + Glue

=> Easy to use virtual machine



Three aspects for the Class

- Understanding OS principles
- System Programming
- Map Concepts to Real Code

Topic Breakdown (Course Objective)

- virtualizing the CPU **(process)**: process, thread, scheduling, synch
- virtualizing memory (**memory)**: memory management, virtual memory
- **storage**: file systems, I/O systems

- Distributed system: communication, data processing, etc (not covered but falls into my research focus, happy to chat!)

Goal 4: Policy/advice/challenges/opportunities?

Your participation

- Lectures
  - essential for doing well in assignments/exams
- Assignments (55% total)
  - 3 programming assignments, 1 written assignment
- Tutorials
  - extra details and hints on assignments
- Exams
  - 3 midterms (15% each on Oct 6, Nov 3, Dec 1, 2025)
- See the course website (https://wenjun-y.github.io/csc360) for detailed schedules

* each exam focuses on one major module

Suggested approach

- Before lectures
  - read textbook; preview video; find questions
- Attend lectures
  - take notes; ***ask*** questions; answer questions!
- After lectures
  - read textbook; explore further
  - write assignments (start early!)
  - get help and help others (discussion forum, etc)
- Do attend tutorials

Common *mistakes*/*complaints*

- "Slides are already online"
  - Lectures are much more than just browsing slides
  - Pay attention to in-class questions/discussion too!
- "Slides are too brief"
  - Slides are just guidelines to navigate/understand
  - Take notes in class and read the textbook!
- "Start to do assignments on the due date"
  - Simple fact: you cannot finish, or even start, them
  - Start early and let us know if you have questions!

read slides the night before the exam/assignment is NOT useful

More systems courses

- *Computer networks (CSc 361 ~ networkOS)*
  - suggestion: take it after CSC360
- *Advanced computer networks (CSc 466)*
- Advanced communication networks (CSc 467)
- Wireless and mobile networks (CSc 463)
- Network management and security (topics/DS)
- Embedded systems (CSc 460)
- Multimedia systems (CSc 461)
- Distributed systems (CSc 462)

Your feedback

- Teaching/learning is interactive
  - two-way communications
- Let me know
  - what you think about lectures, assignments, tutorials, exams, topics, …
  - what you want to know more or probe further
- You can *always* reach me
  - in class, during office hours, by Teams/email

Course Reps

- Please volunteer yourself!
  - **A**nonymize
  - **A**ggregate
  - **A**mplify
- Feedback to the teaching team
  - lecture and lab/tutorial instructors, markers
  - meet once a month face-to-face or by Teams
  - communicate electronically when necessary

Let me know your willingness to volunteer. Thanks!

Course policies

- See official course outline
  - late assignments, mark appeals, etc
  - academic integrity
    - zero tolerance on cheating!
  - accommodation, etc
- No group assignment/project
  - collaboration/participation is encouraged
  - responsibility: your submitted work is yours
  - obligation: give credits to references

whatever you can find elsewhere, we've probably already had it

Need more challenge?

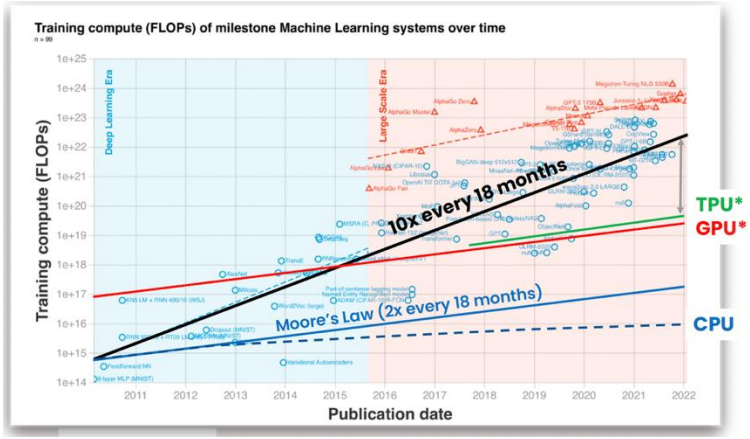
- ACM International Collegiate Programming Contest (ICPC)
  - UVic has been participating in the last few years
  - https://oac.uvic.ca/programmingclub/
- BCNET Broadband Innovation Challenge, or DMC
  - any applications running over broadband networks
  - previous winning projects (some from UVic)
    - http://www.cs.uvic.ca/~pan/bcnet
- Other student competitions and clubs
  - https://www.uvic.ca/ecs/computerscience/undergraduate/student-clubs/index.php

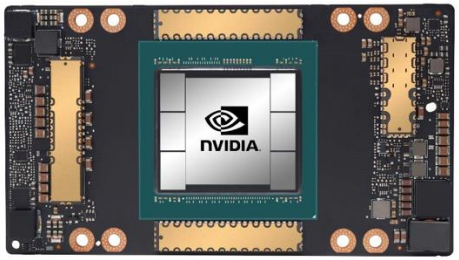
Good News!!!

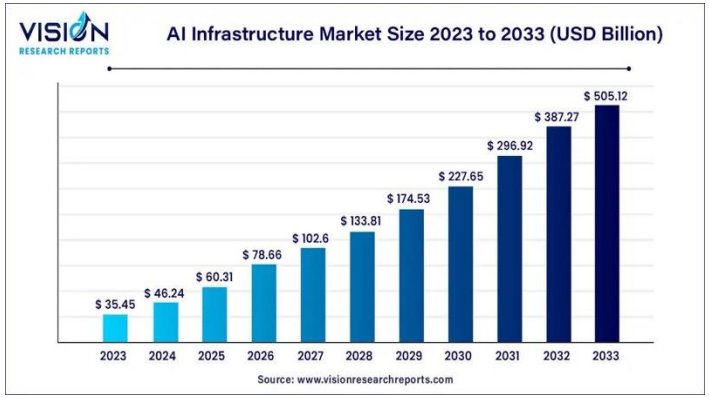
- *"LTI has issued a call for extra JCURA nominations with a short deadline. If you have any new JCURA applications (that were not already submitted for the initial call earlier this year), please submit them to Prof. Jianping Pan (https://webhome.cs.uvic.ca/~pan/) by the internal deadline of* **Tuesday, September 9th**.*"*

One last note…

- Operating systems in the age of AI
- Everything gets reinvented
- This class more relevant than ever







Lots of activity over past 15 years!

Apache MESOS
ALLUXIO

How do you share a cluster across different distributed workloads ?

APACHE Spark™

How do you share a cluster across different users, jobs, and queries?

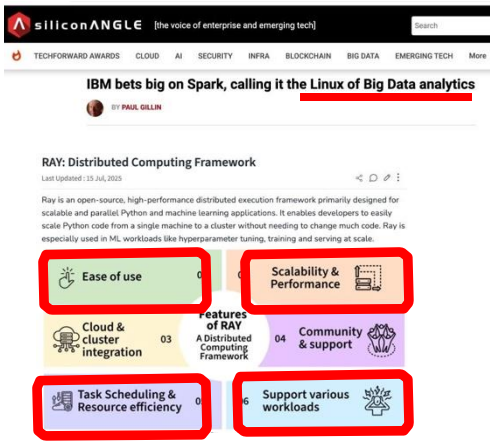
RAY

How do you simplify the implementation and execution of distributed AI apps?

vLLM
SGL

How do you do high performance LLM inference for many models across many GPUs supporting different workloads?

Mesosphere unveils Mesos-based datacenter OS plus $36m injection



- **Memory management**
- **Scheduling, load balancing**
- **Performance isolation**
- **Hierarchical storage (GRAM, RAM, SSD)**
- **…**

This lecture so far

- An introduction to the course
  - who, when, where, what, and why
  - course materials
  - course objectives
  - course topics
  - you and the course

This lecture so far

- Summary of goals
  - Why should you care?
    - The OS is everywhere
  - Why is it hard?
    - Deal with many different devices, many different time scales. Safety-critical
  - What is an Operating System?
    - Provides abstraction of a simple, infinite virtual machine
    - Three roles: illusionist, referee and glue
    - A good OS cares about performance, reliability, security and portability

Next lecture

- Interfaces to OS
  - CLI, GUI, system calls, API
  - read OSC7 Chapter 2 (or OSC6 Chapter 3)

Don’t miss the JCURA chance