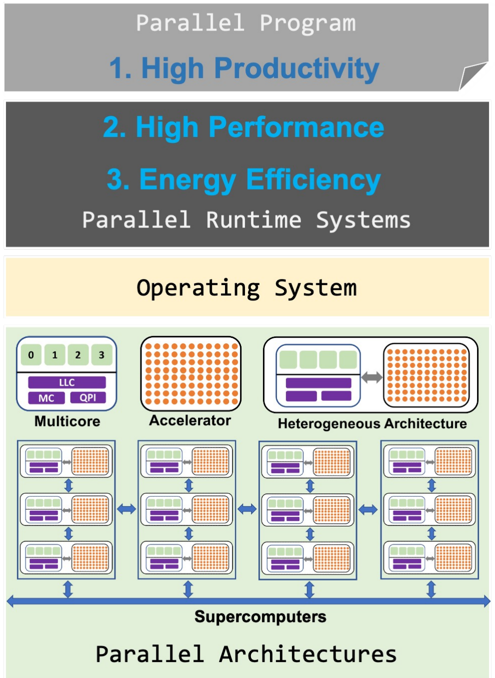
Lecture 01: Course Introduction

Vivek Kumar
Computer Science and Engineering
IIIT Delhi
vivekk@iiitd.ac.in

About me

- https://hipec.github.io/

Research Focus



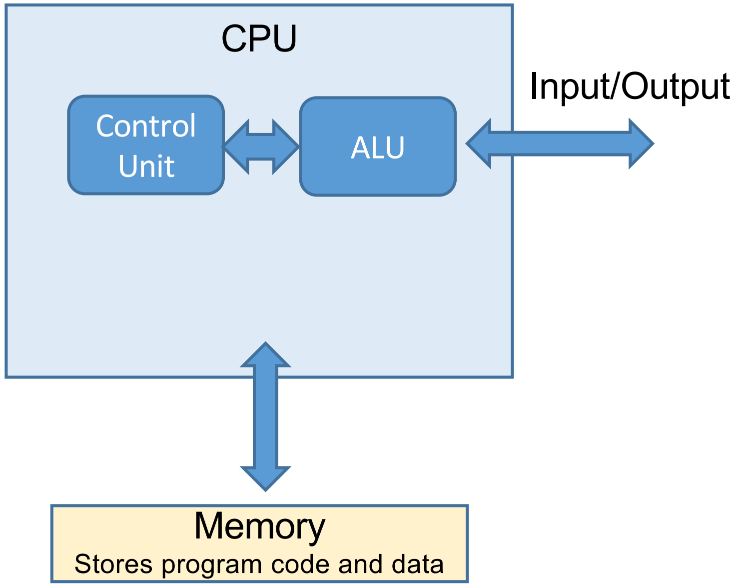
I spend my time at the HiPeC Lab or roaming in the majestic Himalayas..

Today's Lecture

- Modern computing architecture
- High-level overview of operating systems
- Roles of an OS
- Challenges in modern OS
- Course evaluation and logistics

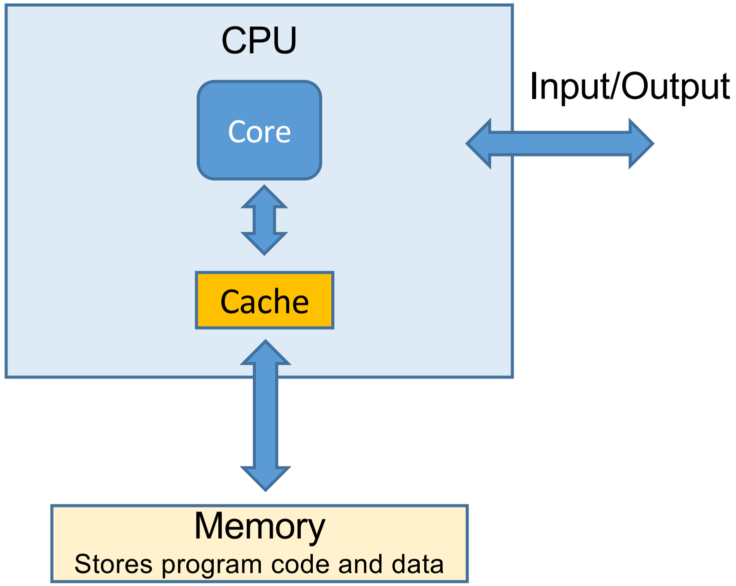
Let us first quickly try to understand the modern computer architecture

Von Neumann Architecture & Associated Issues (1/3)



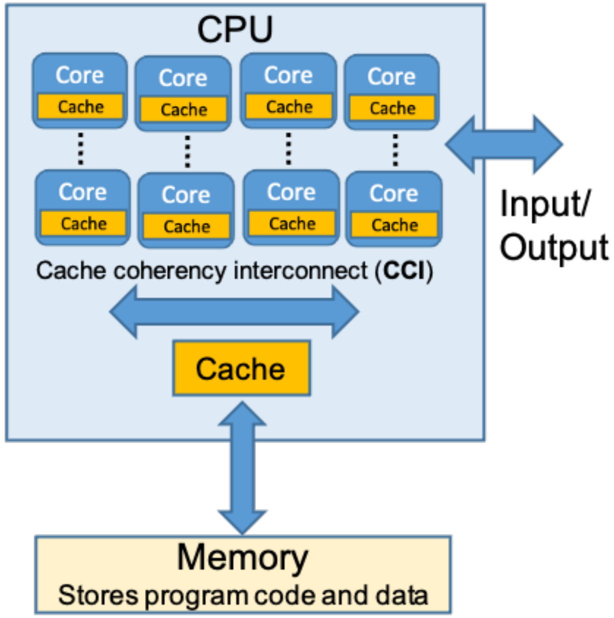
- John Von Neuman in 1945 came up with the architecture for computers that we even use today (albeit with several changes)

Von Neumann Architecture & Associated Issues (2/3)



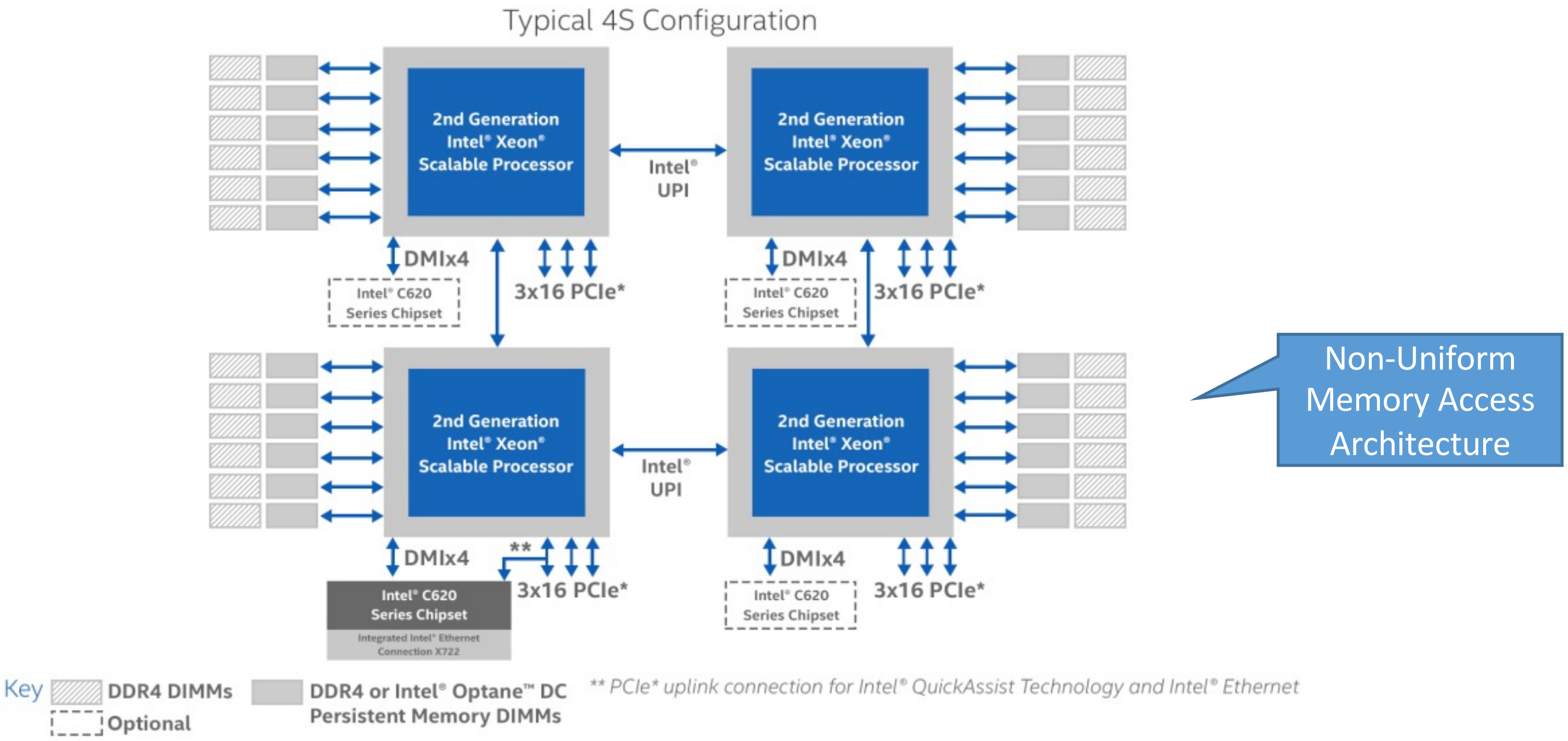
- Memory bottleneck
  - Problem
    - Access latency to memory quite high
      - High CPU stalls while fetching code and data
  - Solution
    - Add cache on the CPU chip to store frequently accessed memory

Von Neumann Architecture & Associated Issues (3/3)



- Performance bottleneck
  - Problem
    - Design issues with increasing the performance of single core processor
      - High heat dissipation
        - Even capable of melting the processor!
      - High power consumption
  - Solution (around 2004)
    - Add more cores to achieve better performance instead of increasing the performance of a single core
      - Still maintains the Moore's law
    - Add cache coherency interconnect (CCI) to fetch data on one core from the other core's cache instead of going all the way up to main memory

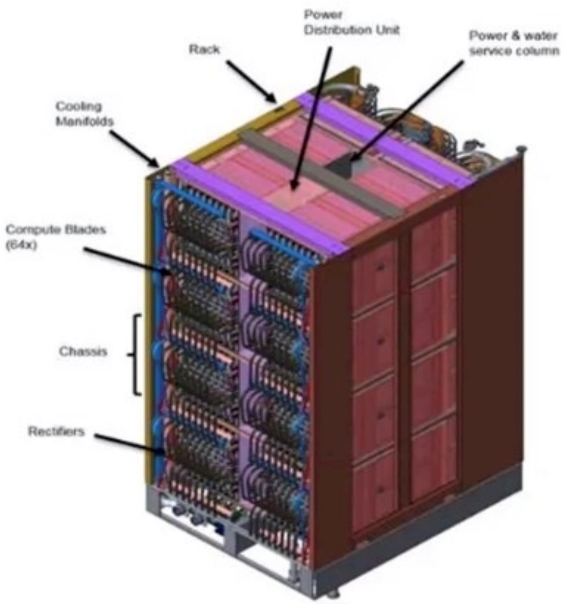
Latest Server Processors



Latest Supercomputer (June 2024)

System
- 2 EF Peak DP FLOPS
- 74 compute racks
- 29 MW Power Consumption
- 9,408 nodes
- 9.2 PB memory
  (4.6 PB HBM, 4.6 PB DDR4)
- Cray Slingshot network with dragonfly topology
- 37 PB Node Local Storage
- 716 PB Center-wide storage
- 4000 ft^2 foot print

Olympus rack
- 128 AMD nodes
- 8,000 lbs
- Supports 400 KW



AMD node
- 1 AMD "Trento" CPU
- 4 AMD MI250X GPUs
- 512 GiB DDR4 memory on CPU
- 512 GiB HBM2e total per node
  (128 GiB HBM per GPU)
- Coherent memory across the node
- 4 TB NVM
- GPUs & CPU fully connected with AMD Infinity Fabric
- 4 Cassini NICs, 100 GB/s network BW

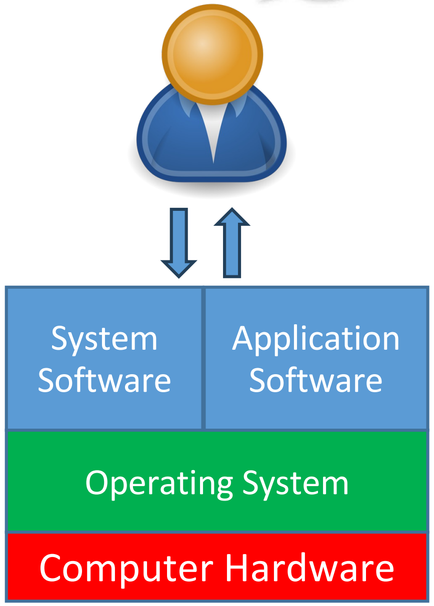
Compute blade
- 2 AMD nodes

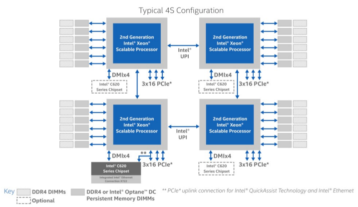
Source: https://www.nextplatform.com/wp-content/uploads/2022/03/oak-ridge-al-geist-frontier-specs.jpg

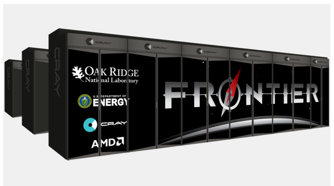
Today's Lecture

- Modern computing architecture
- High-level overview of operating systems
- Roles of an OS
- Challenges in modern OS
- Course evaluation and logistics

What is an Operating System?







- OS is a piece of software whose job is the manage the computer's resources for its users and applications

Analogy: The Hardware

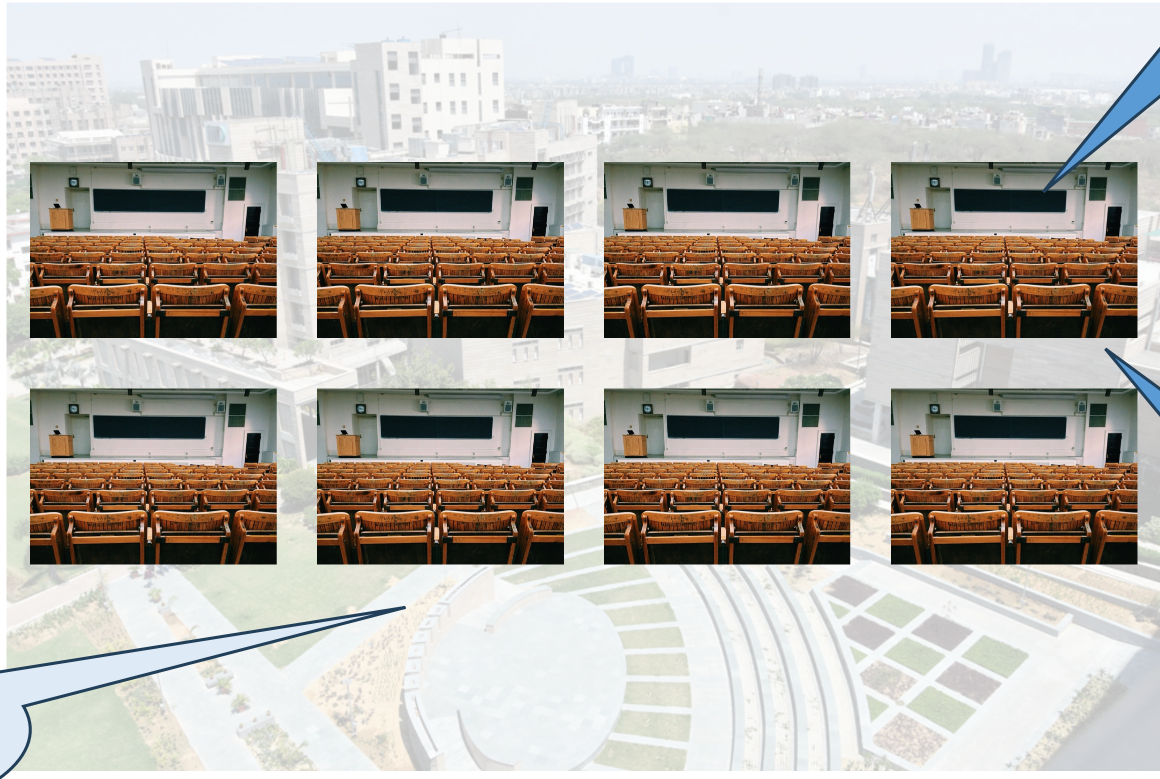
Analogy: Disk & DRAM

Library – persistent storage of data (Disk)

Data is read from the Disk (**Library**) and stored in central server (**DRAM**) in the form of **lecture slides**

Any familiarity with memory hierarchy?

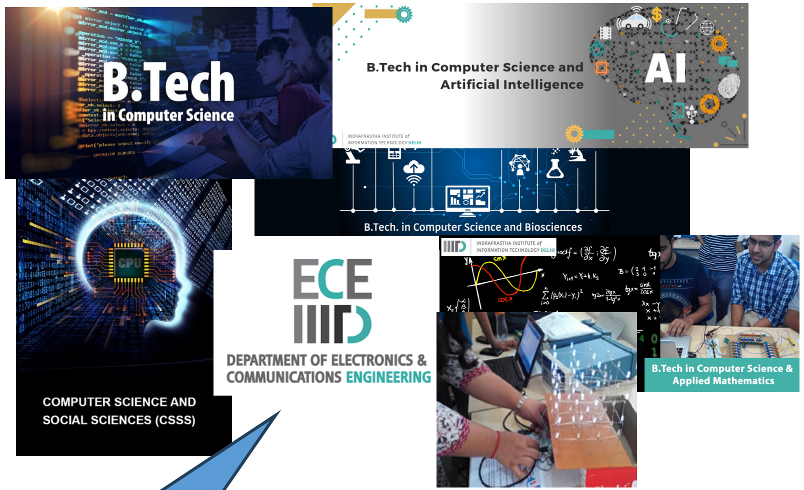
Analogy: The Multicore CPU



Analogy: The Complete Hardware

Analogy: Operating System & Applications

OS: Faculties and Staff



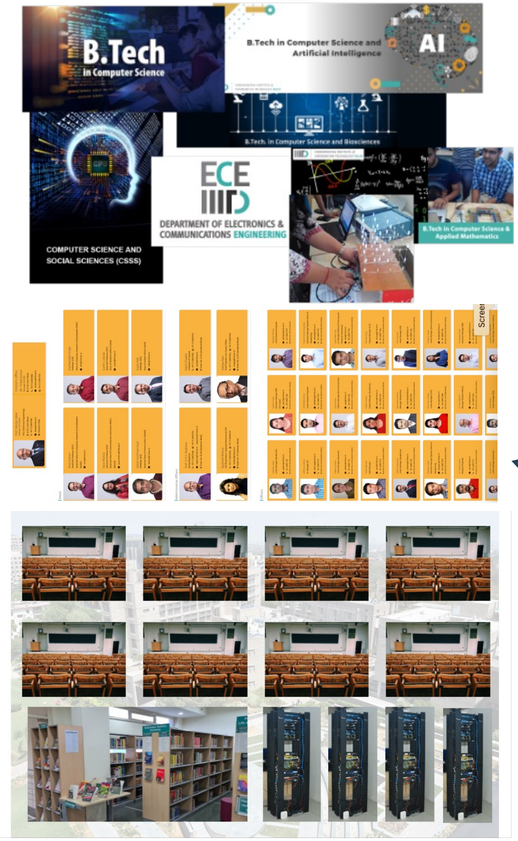
Applications: Several BTech/MTech programs

Each **application** has several **Tasks** to complete (courses)

Major Goals of Operating Systems

1. Virtualization
   - CPU
   - Memory
2. Concurrency
3. Persistence
4. Protection and security

Analogy: Virtualization (CPU cores)



Scheduling: Time table is prepared for each semester for smooth running of each program

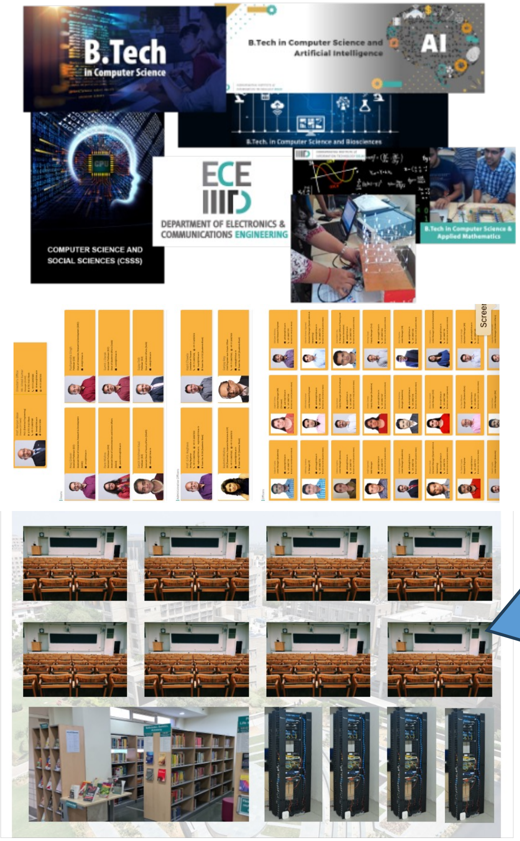
- The operating system gives an "**illusion**" to each application that the entire hardware is only for that application
  - So many applications can execute on the same hardware with limited resources (fixed number of cores, DRAM, and disk)

Analogy: Virtualization (CPU cores)

Every program has many courses, each with several lectures

- Application stored in file v/s running applications
- Each running application (**Process**) has some code and data associated with it (e.g., course and lecture materials)
  - Faculty prepares the lecture by fetching materials from library (**Disk**) and then preparing lecture material/slides saved on servers (**DRAM**)
  - Lectures are **Scheduled** in a classroom where the faculty uses the whiteboard (**Cache**) to teach the topic

Analogy: Virtualization (CPU cores)



- Only one **Task** can run on a CPU core at any given time
  1. A classroom is used to run lectures from different courses
  2. During the lecture, students write down notes from the board (**cache**) onto their notebook (persistent storage)
  3. Next faculty enters the classroom for delivering a different lecture
  4. First, a recap of the topics discussed in the last lecture is carried out before starting the lecture
     - Reloading the context
- Above steps are a very high-level overview of **context switch between processes for time sharing of a CPU core**!

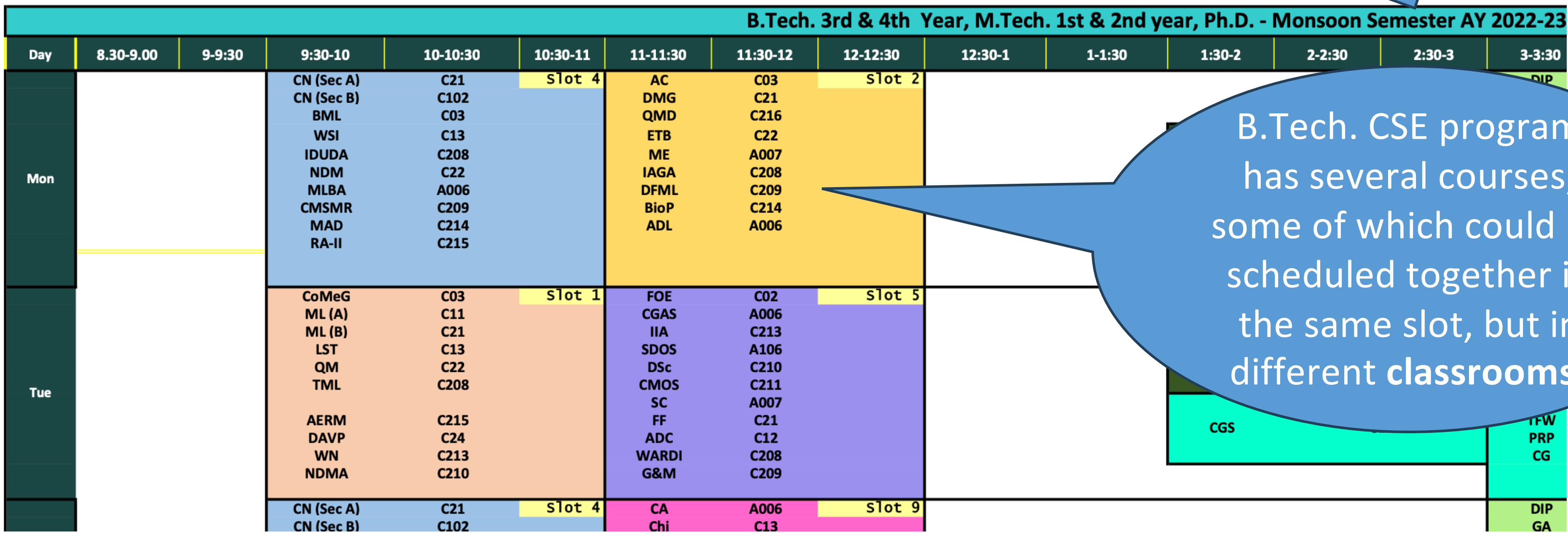
Analogy: Virtualization (Memory)

There are several servers (DRAMs) to store the data associated with each program

- Faculties and admin (OS) stores the data from each program (courses, lectures, documents, etc.) in the servers (DRAMs)
- It gives an illusion that there are dedicated memory spaces for each program
  - In the reality each program’s data could be stored in pieces over several DRAMs (physical address)
  - However, this complexity is abstracted away by the O.S.

Analogy: Concurrency

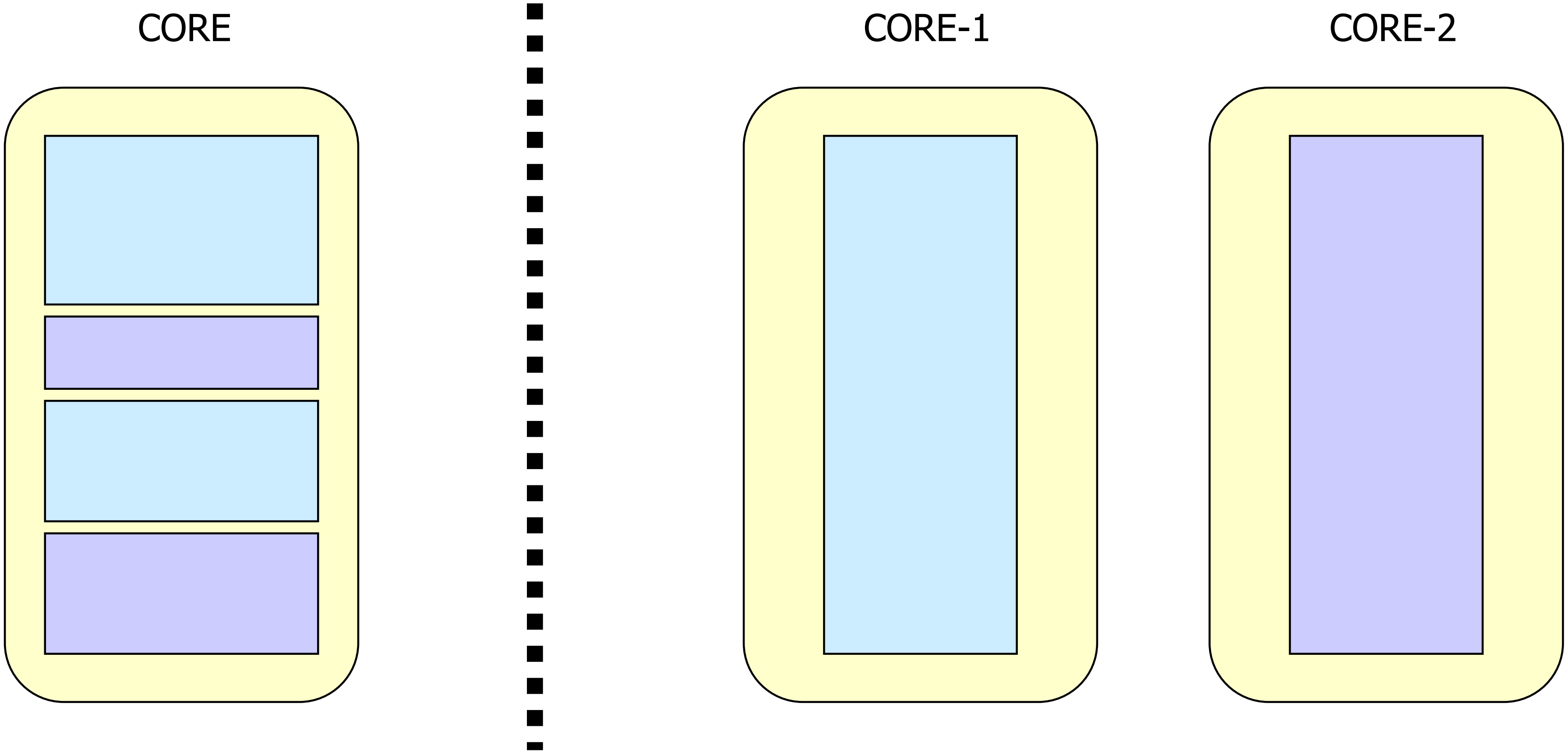
Scheduling algorithm of IIIT-Delhi



B.Tech. CSE program has several courses, some of which could be scheduled together in the same slot, but in different **classrooms**!

- A program could have several independent tasks (set of instructions) that could executed simultaneously over multicore CPUs
  - Multiple threads of execution
- However, some course has prerequisite courses that the students should complete first
  - Dependencies in multi-threaded execution

Concurrency vs. Parallelism



Analogy: Persistence

- At the end of each semester, academic department collect and permanently save the lecture slides, quiz sheets, assignments and project, and student’s answer sheets for mid/end semester exams (for each courses)

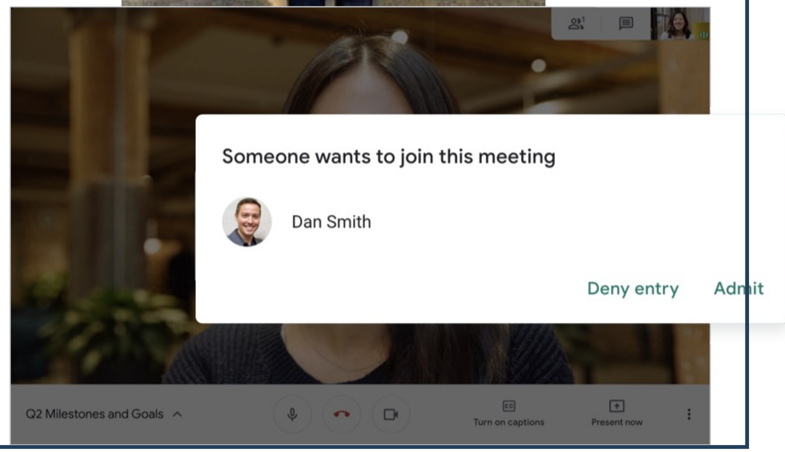
Analogy: Protection (Internal) & Security (External)

≠

ECE
'ARTMENT OF ELECTRONICS & COMMUNICATIONS ENGINEERING

Course Name	Course Acronym	Course Code
Computer Architecture	CA	CSE511/ECE511

Each program can only access its own data (courses) unless there is some authorization obtained by the O.S



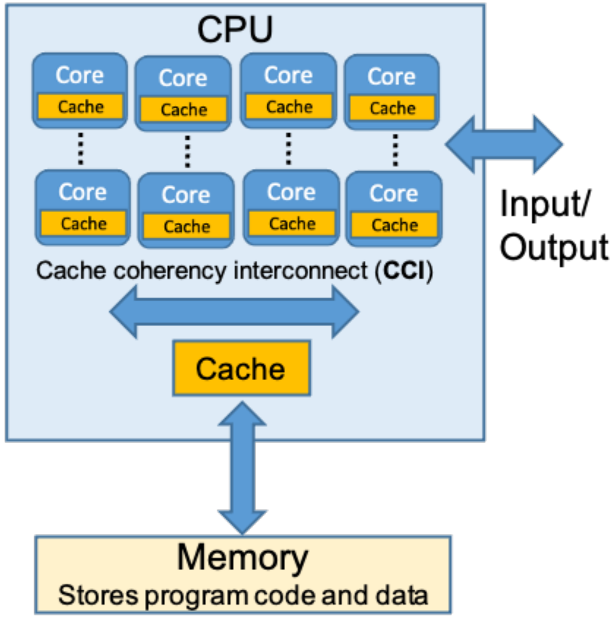
Today's Lecture

- Modern computing architecture
- High-level overview of operating systems
- Roles of an OS
- Challenges in modern OS
- Course evaluation and logistics

Roles of an Operating System

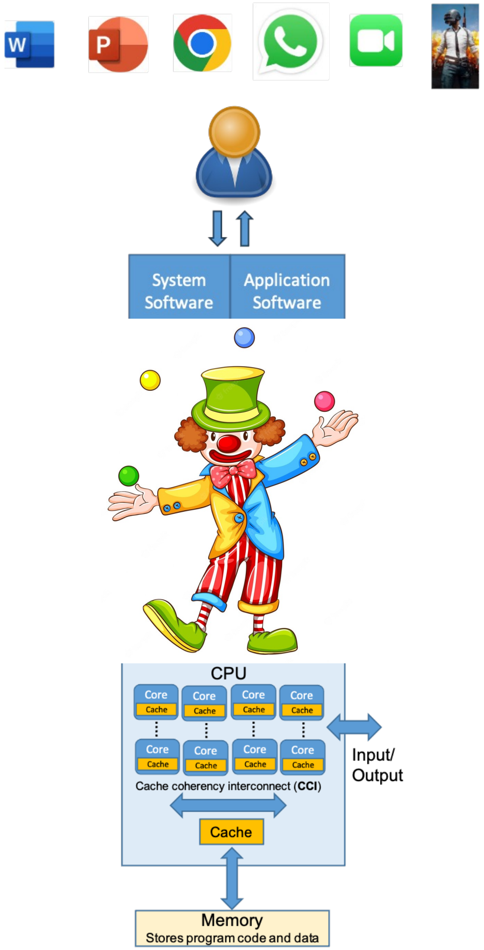
- Referee
  - Manages all the shared resources in the computer
- Illusionist
  - Each application thinks that the entire computer belong to itself
- Glue
  - Offers standard services to simplify application development and facilitate sharing

Roles of an Operating System (Example-1)



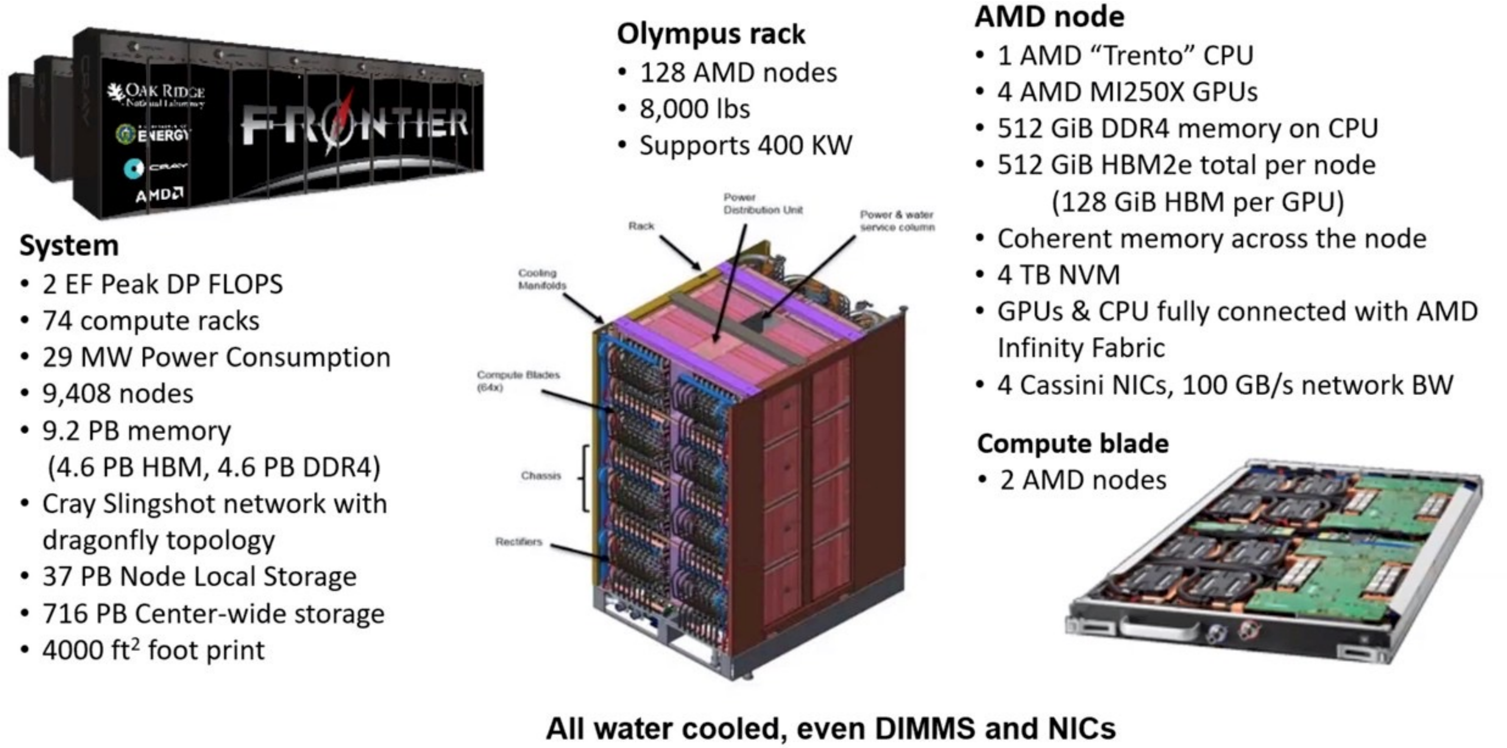
- Let us try to understand it from the perspective of a desktop/laptop

Roles of an Operating System (Example-1)



- **Referee**
  - Each application will be launching several tasks
    - Gaming app will require graphic rendering, network play support, taking player input via keyboard/mouse, etc.
    - Zoom will require tasks such as network connection, access to mike/camera, graphic rendering, etc.
  - As there are several cores, each task can run on a different core
    - But, there are only finite number of resources (cores, caches, input/output devices, memory etc.)
- **Illusionist**
  - Allows the application to have hundreds of tasks (several times more than available cores), infinite amount of memory (DRAMs are only in few GBs), etc.
- **Glue**
  - Standard APIs for network connection, accessing input/output devices, etc.

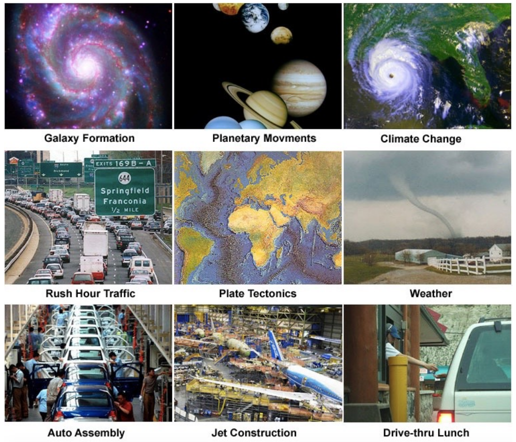
Roles of an Operating System (Example-2)

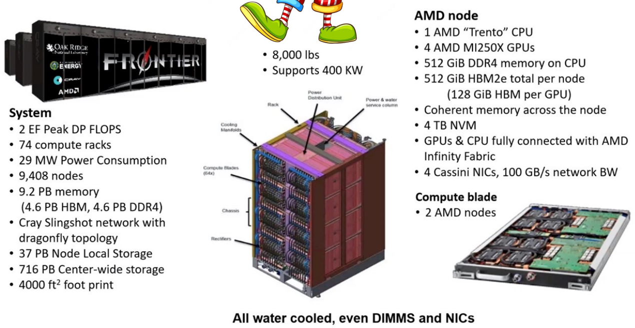


- Let us now try to understand it from the perspective of a **supercomputer**

Picture source: https://www.nextplatform.com/wp-content/uploads/2022/03/oak-ridge-al-geist-frontier-specs.jpg

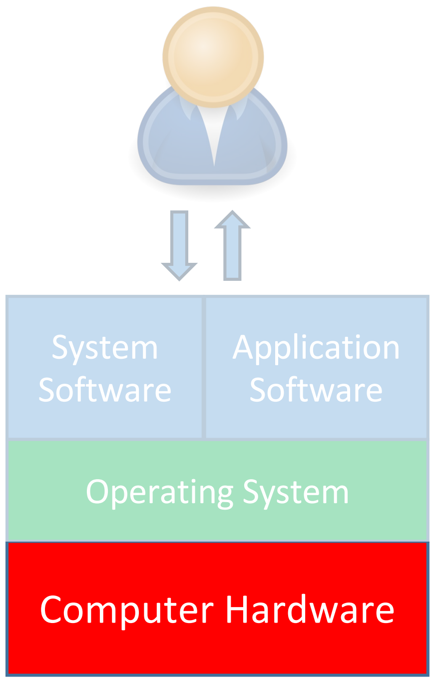
Roles of an Operating System (Example-2)





- **Referee**
  - Each application will be launching several tasks
    - Either computation, communication, or file handling (no camera, mic, etc.)
    - Application can wait in queue if it is requesting more cores than currently available
- **Illusionist**
  - Allows the application to interact with the supercomputer via a single computer node
    - Several computing nodes, each with its own OS
    - Application tasks distributed across all the nodes
    - Adapts to node failures (migrate tasks to another node)
- **Glue**
  - APIs for communication and accessing disjoint data (at different nodes) with low latency (e.g., Message Passing Interface APIs)

Operating System Specific to Hardware

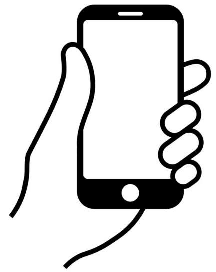


- Throughput oriented (manages several tasks)
- E.g., Windows, Linux, MacOS

Laptop/Desktop Operating System

- Lightweight
- Highly user friendly (GUI)
- Smaller memory footprint (more than RTOS)
- E.g., Andriod, iOS

Mobile Operating System



- Time bound and limited tasks
- May work with small memory
- E.g., Nucleus RTOS

Real time Operating System

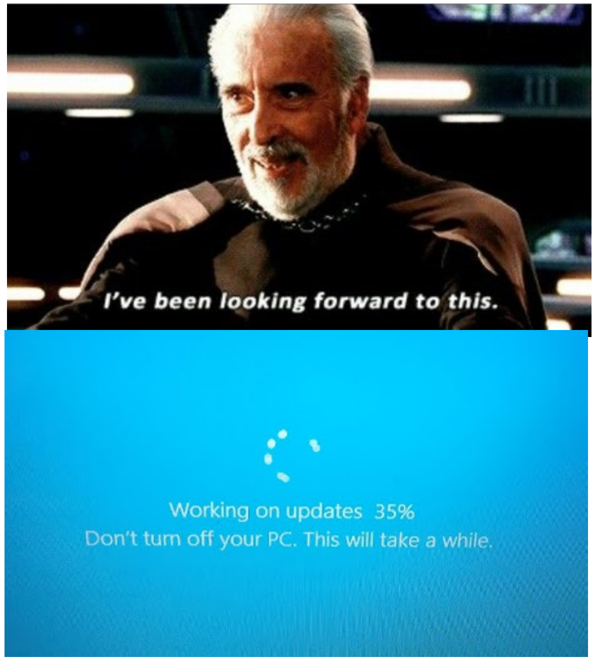
Today's Lecture

- Modern computing architecture
- High-level overview of operating systems
- Roles of an OS
- Challenges in modern OS
- Course evaluation and logistics

Major Challenges in Modern OS

- Dependability

Me: *restarts computer because it's lagging*
Windows Update:



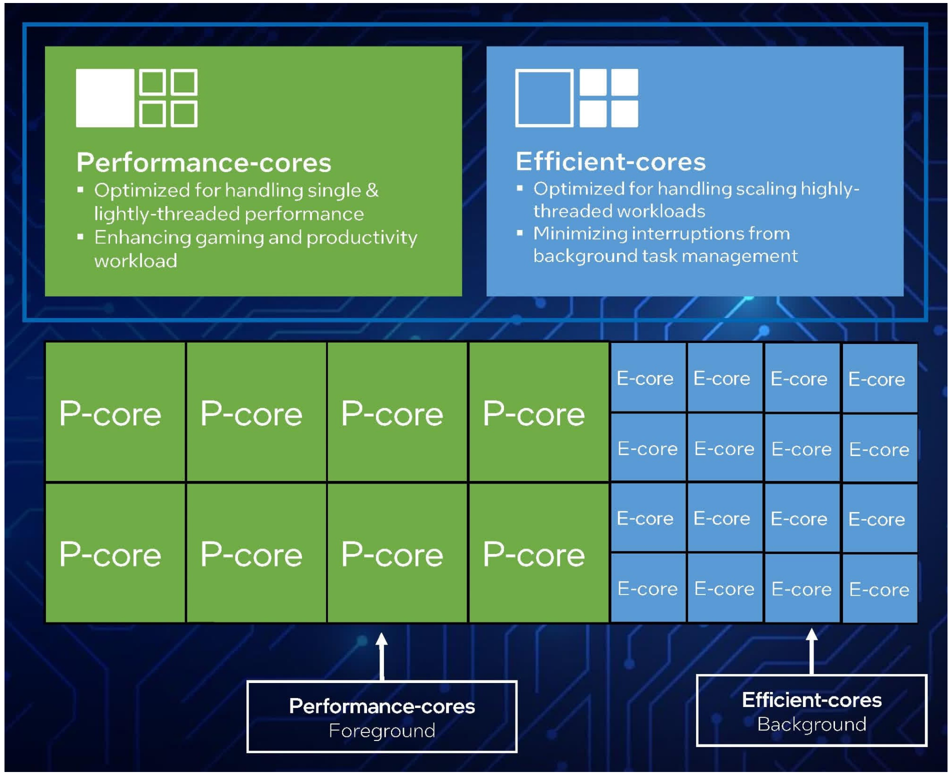
Predictable? Reliable? (frequent bug fixes, updates…)

V/S

Predictable? Reliable?

Major Challenges in Modern OS

- Performance and Energy

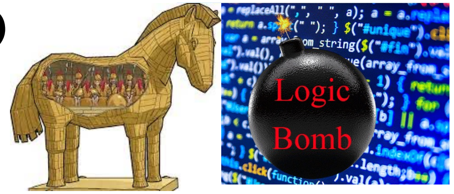


Which tasks should run on P-core and which ones on E-cores?

What should be the processor core frequency while running a particular task? High frequency can give better performance, but requires more energy, and vice-versa

Images source: https://www.techspot.com/news/94919-preview-core-i9-13900-engineering-sample-performance-looks.html

Major Challenges in Modern OS



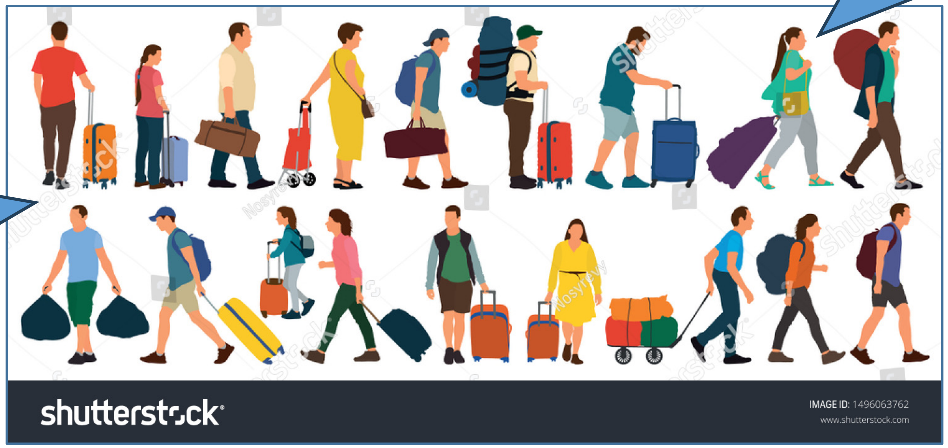
- Protection and Security

They are lacking **Security**

Let's unleash our arsenal

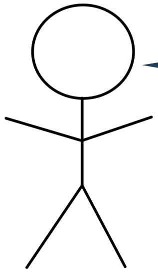
We are **Protected**

Each application can only access its own **baggage** (files, memory, CPU). They should not access another application's **baggage** unless they have proper authorization



Major Challenges in Modern OS

- Portability



I am quite **Portable**. I just need to change my shoes






I am quite **Portable too!** I just need to change my machine interfacing code in kernel

Today's Lecture

- Modern computing architecture
- High-level overview of operating systems
- Roles of an OS
- Challenges in modern OS
- Course evaluation and logistics

Course Evaluation (Section-A)

- Both sections will be graded separately
  - Evaluation components are same, but quizzes/assignments/exams will be total different
  - Below mentioned N-1 policy specific to Section-A only
- Quiz: 10% (N-1 policy)
  - No surprise quizzes
  - Will be held during lecture hours (around 20mins duration)
- Semester exams
  - Mid-sem: 20%
  - End-sem: 35%
- Take home assignments 35%
  - **NO (N-1 policy) !!!**

I might have a different lecture plan/content than you see currently on Techtree. However, the COs would remain the same

Important Information

1. Please **Don't** open-source assignment implementations after the course gets over
2. We will **Not** upload mid/end semester solution/rubric on Google Classroom
   - Although, we will discuss it in class
3. We will be taking in-class attendance
   - Lecture recordings will not be provided

Course Prerequisites

- **C programming and debugging skills are must**!
  - You must have used C in your DSA course and in OS refresher module
  - If you don’t know C then better start practicing so as you don’t face issues with the course assignments
    - First one will be released early next week!

We will strictly follow the IIITD **plagiarism policy**. No excuses if you get caught in plagiarism

Next Lecture

- Source code to machine code