Summer Bootcamp 2021

Introduction to Computer Science

Lecture 1

Selected Topics in Computer Science

Artem Burmyakov

August 02, 2021

Innopolis University

Our Objectives

• To overview some Computer Science (CS) topics;
• To review some relevant material for the following study;
• To become familiar with English terminology in CS;
• To overview a typical semester structure, grading policies;
• To get used to each other and course instructors

About the Primary Instructor (PI)

Education	
2004, 2007	BSc and MSc in Applied Math, Moscow Engineering Physics Institute
2016	PhD in Computer Science, University of Porto

Tutorial / Lab Instructor for IU courses
Computer Architecture (1st semester, core course), with Prof. Tormasov
Computer Networks (4th semester, core course), with Prof. Hussain
Consensus Theory and Concurrent Programming (6th semester, elective), with Prof. Tormasov
Research Methodology and Scientific Writing (for PhD students), with Prof. Meyer

Research Interests
Real-time multiprocessor scheduling
Optimization of computationally intensive algorithms
Concurrent programming

Course Outline

Day	Lecture (Time: 10:30-12:00)	Lab (Time: 13:00-14:30; 14:45-16:15)
02/Aug	**Selected Topics in Computer Science**	**Recap/Introduction of C++ syntax**: - installation of a C++ development tool; - for-loops, if-statements, math operations, inclusion of external libraries; - program compilation;
03/Aug	**Evolution and classification of programming languages**: - Evolution history; - Classification based on abstraction levels from hardware (low and high programming languages), and purpose (general-purpose or domain-specific); - Key differences (in performance, complexity, etc.), incl. C/C++, Java, and Python	**Programming exercises in C++;** **Performance evaluation of C++, Java, and Python:** - A brief syntax comparison; - Execution time measurement; - Performance evaluation of a sample algorithm implementation
04/Aug	**Evolution and classification of programming languages (cont.)** **Iterative and recursive algorithms**: - Advantages and disadvantages of iterative and recursive implementations; - Introduction to a program runtime analysis; - Examples (incl. a factorial computation)	**Evaluation of iterative and recursive algorithms**
05/Aug	**The Basics of Combinatorics**: - Canonical problems of combinatorics: combinations and permutations; - Overview of a Travelling Salesman Problem (TSP); - Introduction to computation and space complexity of algorithms;	**Exercises on combinatorics;** **The implementation of a brute-force solution for TSP:** - The overview of data structures and instructions needed; - The evaluation of iterative and recursive solution implementations;
06/Aug	**The Basics of Probability Theory**: - Discrete random variable; - Probabilities of independent, mutual exclusive, and not mutual exclusive events; - Probability distribution	**Exercises on computing probabilities** **Preparation to the final exam**
07/Aug	**Exam**	

Topics of Computer Science:

Hardware architecture of computers

Topics of Computer Science:

Hardware architecture of computers
Programming languages (machine language, high-level programming languages)

Topics of Computer Science:

Hardware architecture of computers
Programming languages (machine language, high-level programming languages)
Data structures and algorithms, runtime and memory complexity of algorithms

Topics of Computer Science:

Hardware architecture of computers
Programming languages (machine language, high-level programming languages)
Data structures and algorithms, runtime and memory complexity of algorithms
Operating systems

Topics of Computer Science:

Hardware architecture of computers
Programming languages (machine language, high-level programming languages)
Data structures and algorithms, runtime and memory complexity of algorithms
Operating systems
Parallel and distributed computing

Topics of Computer Science:

Hardware architecture of computers
Programming languages (machine language, high-level programming languages)
Data structures and algorithms, runtime and memory complexity of algorithms
Operating systems
Parallel and distributed computing
Computer networks

Topics of Computer Science:

Hardware architecture of computers
Programming languages (machine language, high-level programming languages)
Data structures and algorithms, runtime and memory complexity of algorithms
Operating systems
Parallel and distributed computing
Computer networks
Artificial intelligence (including Machine Learning)

Topics of Computer Science:

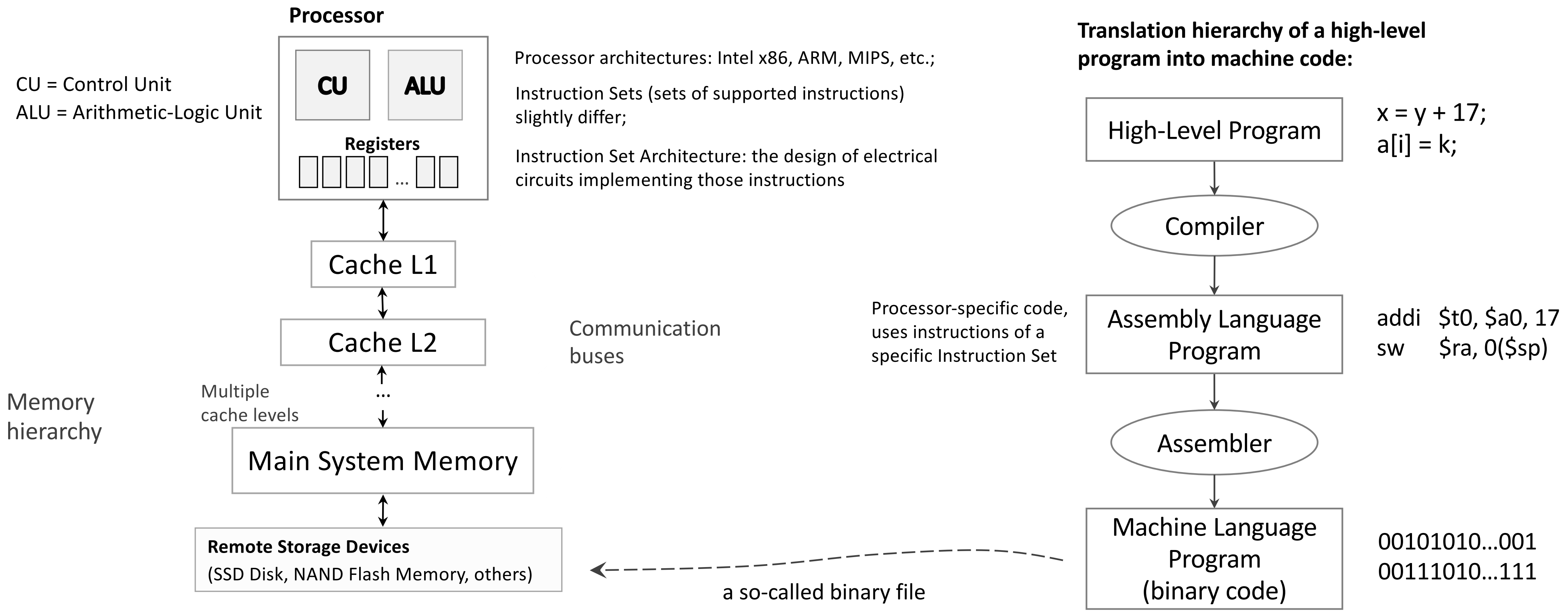
Hardware architecture of computers
Programming languages (machine language, high-level programming languages)
Data structures and algorithms, runtime and memory complexity of algorithms
Operating systems
Parallel and distributed computing
Computer networks
Artificial intelligence (including Machine Learning)
Many other

Selected Courses Taught at IU

Course Name	*Some aspects studied*
Computer Architecture	Hardware layout of a computer (processor, communication bus, memory units, etc.); Software/hardware interaction; Key ideas of modern computers: multiprocessing, caching, execution by prediction, etc.

Selected Courses Taught at IU

Course Name	*Some aspects studied*
Computer Architecture	Hardware layout of a computer (processor, communication bus, memory units, etc.); Software/hardware interaction; Key ideas of modern computers: multiprocessing, caching, execution by prediction, etc.



Selected Courses Taught at IU

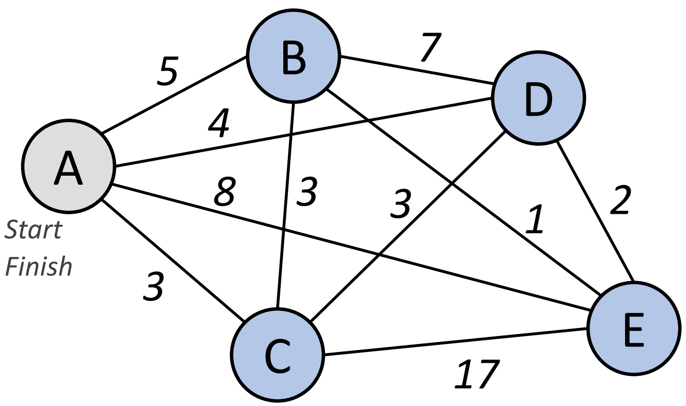
Course Name	*Some aspects studied*
Computer Architecture	Hardware layout of a computer (processor, communication bus, memory units, etc.); Software/hardware interaction; Key ideas of modern computers: multiprocessing, caching, execution by prediction, etc.
Data Structures and Algorithms	Classical algorithms in Computer Science (e.g. sorting, graph traversal); Algorithms analysis (e.g. runtime and space complexity, asymptotic analysis); Data structures (e.g. priority queues, hash tables)

Selected Courses Taught at IU

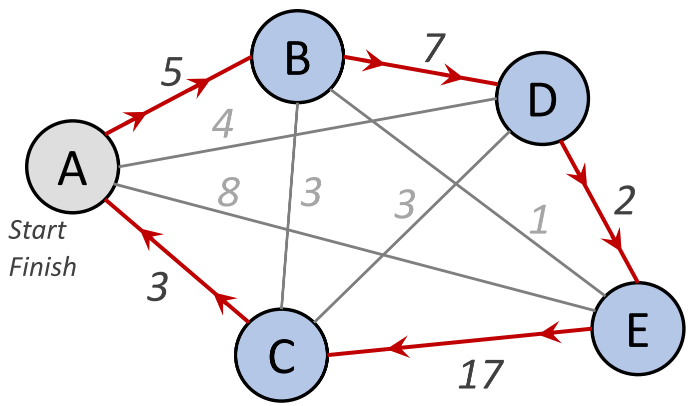
Course Name	*Some aspects studied*
Computer Architecture	Hardware layout of a computer (processor, communication bus, memory units, etc.); Software/hardware interaction; Key ideas of modern computers: multiprocessing, caching, execution by prediction, etc.
Data Structures and Algorithms	Classical algorithms in Computer Science (e.g. sorting, graph traversal); Algorithms analysis (e.g. runtime and space complexity, asymptotic analysis); Data structures (e.g. priority queues, hash tables)

Example problem: To find the shortest path through all cities on a map

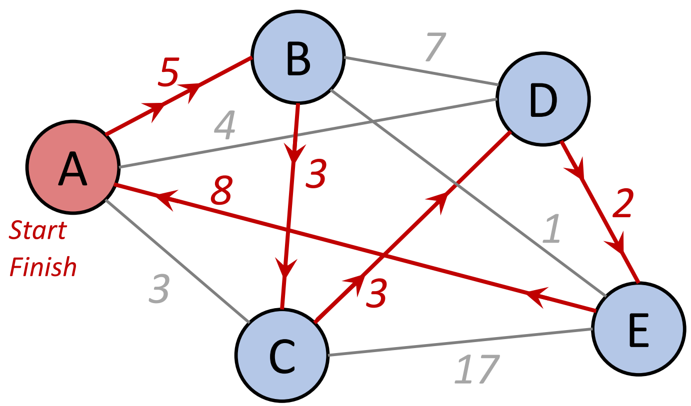
Graph representing the distances between cities:
A, B, C, D, E – cities;
Numbers near links – distances



One possible path, of length 34:



A shorter path, of length 21:
(But is it the optimal one?!)

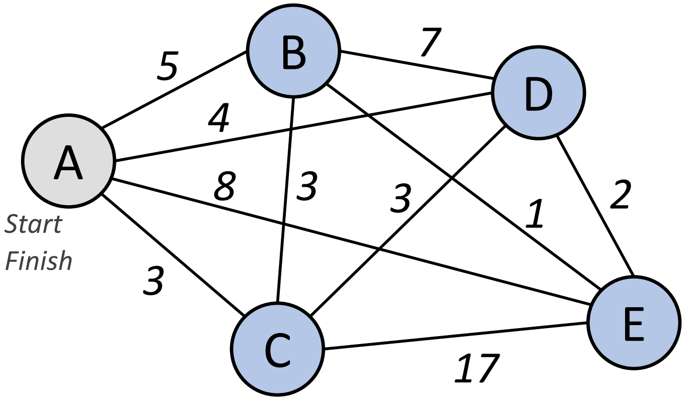


Selected Courses Taught at IU

Course Name	*Some aspects studied*
Computer Architecture	Hardware layout of a computer (processor, communication bus, memory units, etc.); Software/hardware interaction; Key ideas of modern computers: multiprocessing, caching, execution by prediction, etc.
Data Structures and Algorithms	Classical algorithms in Computer Science (e.g. sorting, graph traversal); Algorithms analysis (e.g. runtime and space complexity, asymptotic analysis); Data structures (e.g. priority queues, hash tables)

Example problem: To find the shortest path through all cities on a map

Graph representing the distances between cities:
A, B, C, D, E – cities;
Numbers near links – distances



Multiple solving algorithms exist;
Brute-force enumeration (or exhaustive search) – just one of them (and not the best)

Algorithms differ in computational and space complexity:

- Computational complexity: How quickly increases an algorithm runtime with the number of cities?
- Space complexity: How much increases the memory demand of an algorithm with the number of cities?

Some algorithms are incomparable
from the perspective of runtime and memory demand

Selected Courses Taught at IU

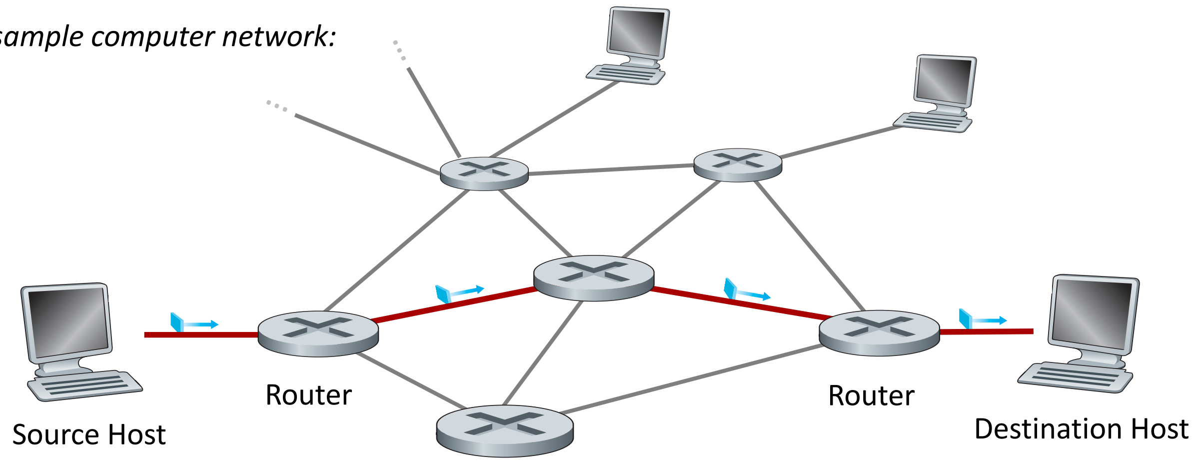
Course Name	*Some aspects studied*
Computer Architecture	Hardware layout of a computer (processor, communication bus, memory units, etc.); Software/hardware interaction; Key ideas of modern computers: multiprocessing, caching, execution by prediction, etc.
Data Structures and Algorithms	Classical algorithms in Computer Science (e.g. sorting, graph traversal); Algorithms analysis (e.g. runtime and space complexity, asymptotic analysis); Data structures (e.g. priority queues, hash tables)
Computer Networks	Communication and routing protocols for computer networks; Client-server and peer-to-peer applications; Communication security

Problem:
Develop an efficient routing algorithm to transfer data packets between network hosts:

- Multiple routing paths exist;
- They result in different transfer times;
- Network congestion might occur

Different routing protocols are derived, based on routing algorithm used (e.g. BGP and OSPF)

A sample computer network:

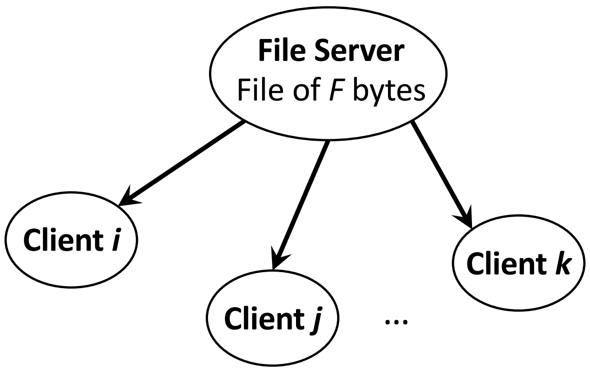


Selected Courses Taught at IU

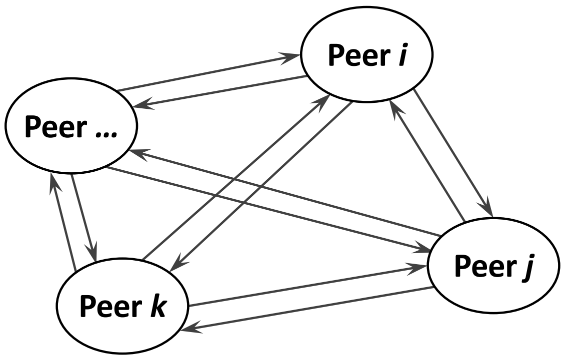
Course Name	*Some aspects studied*
Computer Architecture	Hardware layout of a computer (processor, communication bus, memory units, etc.); Software/hardware interaction; Key ideas of modern computers: multiprocessing, caching, execution by prediction, etc.
Data Structures and Algorithms	Classical algorithms in Computer Science (e.g. sorting, graph traversal); Algorithms analysis (e.g. runtime and space complexity, asymptotic analysis); Data structures (e.g. priority queues, hash tables)
Computer Networks	Communication and routing protocols for computer networks; Client-server and peer-to-peer applications; Communication security

Another problem: Client-Server versus Peer-to-Peer file distribution

Client-Server: All clients download a file from the same server



Peer-to-peer: Each peer downloads different portions of a file from other peers, as well as uploads a file to other

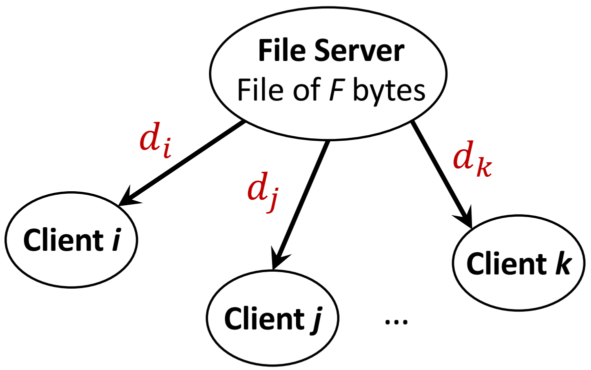


Selected Courses Taught at IU

Course Name	*Some aspects studied*
Computer Architecture	Hardware layout of a computer (processor, communication bus, memory units, etc.); Software/hardware interaction; Key ideas of modern computers: multiprocessing, caching, execution by prediction, etc.
Data Structures and Algorithms	Classical algorithms in Computer Science (e.g. sorting, graph traversal); Algorithms analysis (e.g. runtime and space complexity, asymptotic analysis); Data structures (e.g. priority queues, hash tables)
Computer Networks	Communication and routing protocols for computer networks; Client-server and peer-to-peer applications; Communication security

Another problem: Client-Server versus Peer-to-Peer file distribution

Client-Server: All clients download a file from the same server

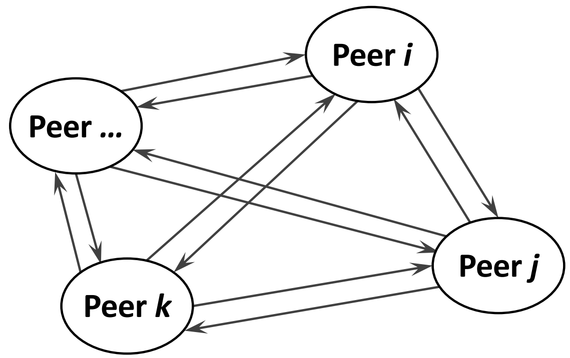


d_ℓ- download speed by Client ℓ

u_s- upload speed by Server

$$\sum_{\text{Clients}} d_\ell \leq u_s$$

Peer-to-peer: Each peer downloads different portions of a file from other peers, as well as uploads a file to other

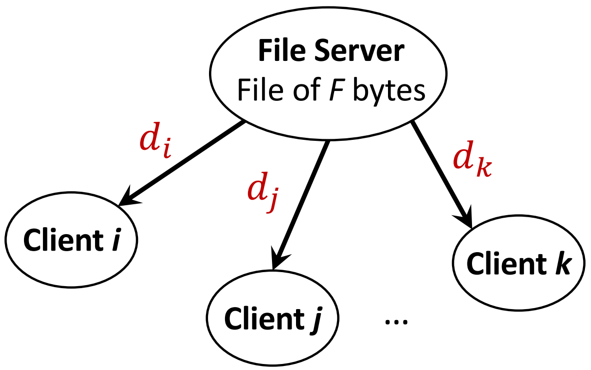


Selected Courses Taught at IU

Course Name	*Some aspects studied*
Computer Architecture	Hardware layout of a computer (processor, communication bus, memory units, etc.); Software/hardware interaction; Key ideas of modern computers: multiprocessing, caching, execution by prediction, etc.
Data Structures and Algorithms	Classical algorithms in Computer Science (e.g. sorting, graph traversal); Algorithms analysis (e.g. runtime and space complexity, asymptotic analysis); Data structures (e.g. priority queues, hash tables)
Computer Networks	Communication and routing protocols for computer networks; Client-server and peer-to-peer applications; Communication security

Another problem: Client-Server versus Peer-to-Peer file distribution

Client-Server: All clients download a file from the same server

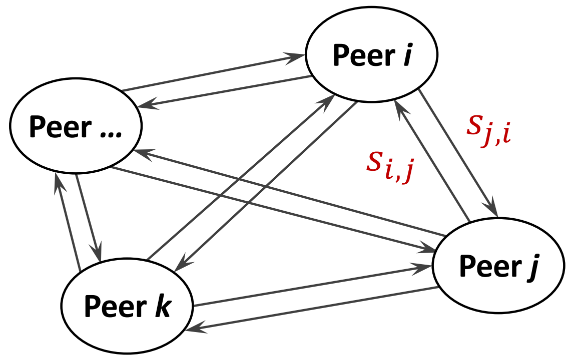


d_ℓ- download speed by Client ℓ

u_s- upload speed by Server

$$\sum_{\text{Clients}} d_\ell \leq u_s$$

Peer-to-peer: Each peer downloads different portions of a file from other peers, as well as uploads a file to other



$s_{i,j}$ - transfer speed from Peer j to i

$$s_{i,j} = min\{d_i, u_j\}$$

Download capacity by Peer *i*

Upload capacity by Peer *j*

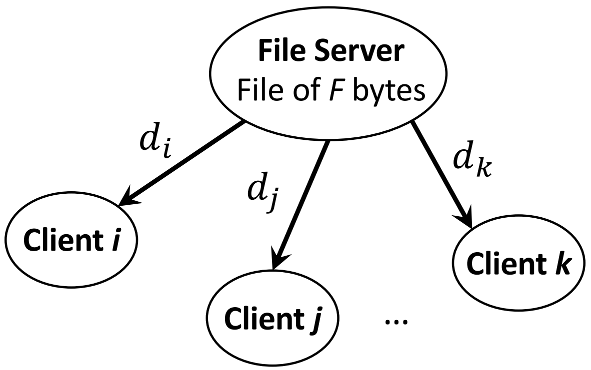
Math tools are then applied for models comparison

Selected Courses Taught at IU

Course Name	*Some aspects studied*
Computer Architecture	Hardware layout of a computer (processor, communication bus, memory units, etc.); Software/hardware interaction; Key ideas of modern computers: multiprocessing, caching, execution by prediction, etc.
Data Structures and Algorithms	Classical algorithms in Computer Science (e.g. sorting, graph traversal); Algorithms analysis (e.g. runtime and space complexity, asymptotic analysis); Data structures (e.g. priority queues, hash tables)
Computer Networks	Communication and routing protocols for computer networks; Client-server and peer-to-peer applications; Communication security

Another problem: Client-Server versus Peer-to-Peer file distribution

Client-Server: All clients download a file from the same server

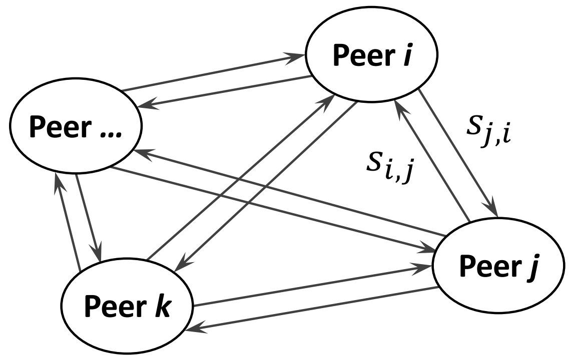


d_ℓ- download speed by Client ℓ

u_s- upload speed by Server

$$\sum_{\text{Clients}} d_\ell \leq u_s$$

Peer-to-peer: Each peer downloads different portions of a file from other peers, as well as uploads a file to other



$s_{i,j}$ - transfer speed from Peer j to i

$$s_{i,j} = min\{d_i, u_j\}$$

Download capacity by Peer *i*

Upload capacity by Peer *j*

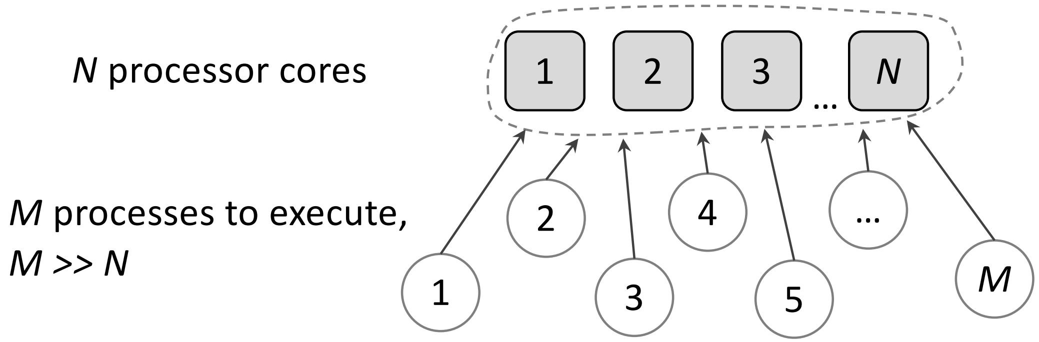
Which approach is better?

Selected Courses Taught at IU

Course Name	*Some aspects studied*
Computer Architecture	Hardware layout of a computer (processor, communication bus, memory units, etc.); Software/hardware interaction; Key ideas of modern computers: multiprocessing, caching, execution by prediction, etc.
Data Structures and Algorithms	Classical algorithms in Computer Science (e.g. sorting, graph traversal); Algorithms analysis (e.g. runtime and space complexity, asymptotic analysis); Data structures (e.g. priority queues, hash tables)
Computer Networks	Communication and routing protocols for computer networks; Client-server and peer-to-peer applications; Communication security
Operating Systems	Operating system design (incl. kernel, file system, I/O interaction); Processor scheduling algorithms, multithreading; Virtual memory

Problem: Operating System Scheduler

There are more processes than cores;
How to prioritise (schedule) processes?
For how long to allocate CPU time slot?
Many other questions…



Selected Courses Taught at IU

Course Name	*Some aspects studied*
Computer Architecture	Hardware layout of a computer (processor, communication bus, memory units, etc.); Software/hardware interaction; Key ideas of modern computers: multiprocessing, caching, execution by prediction, etc.
Data Structures and Algorithms	Classical algorithms in Computer Science (e.g. sorting, graph traversal); Algorithms analysis (e.g. runtime and space complexity, asymptotic analysis); Data structures (e.g. priority queues, hash tables)
Computer Networks	Communication and routing protocols for computer networks; Client-server and peer-to-peer applications; Communication security
Operating Systems	Operating system design (incl. kernel, file system, I/O interaction); Processor scheduling algorithms, multithreading; Virtual memory
Introduction to Programming (in C language)	Basic programming concepts (loops, branching, etc.); Functional and object-oriented programming; Exception handling

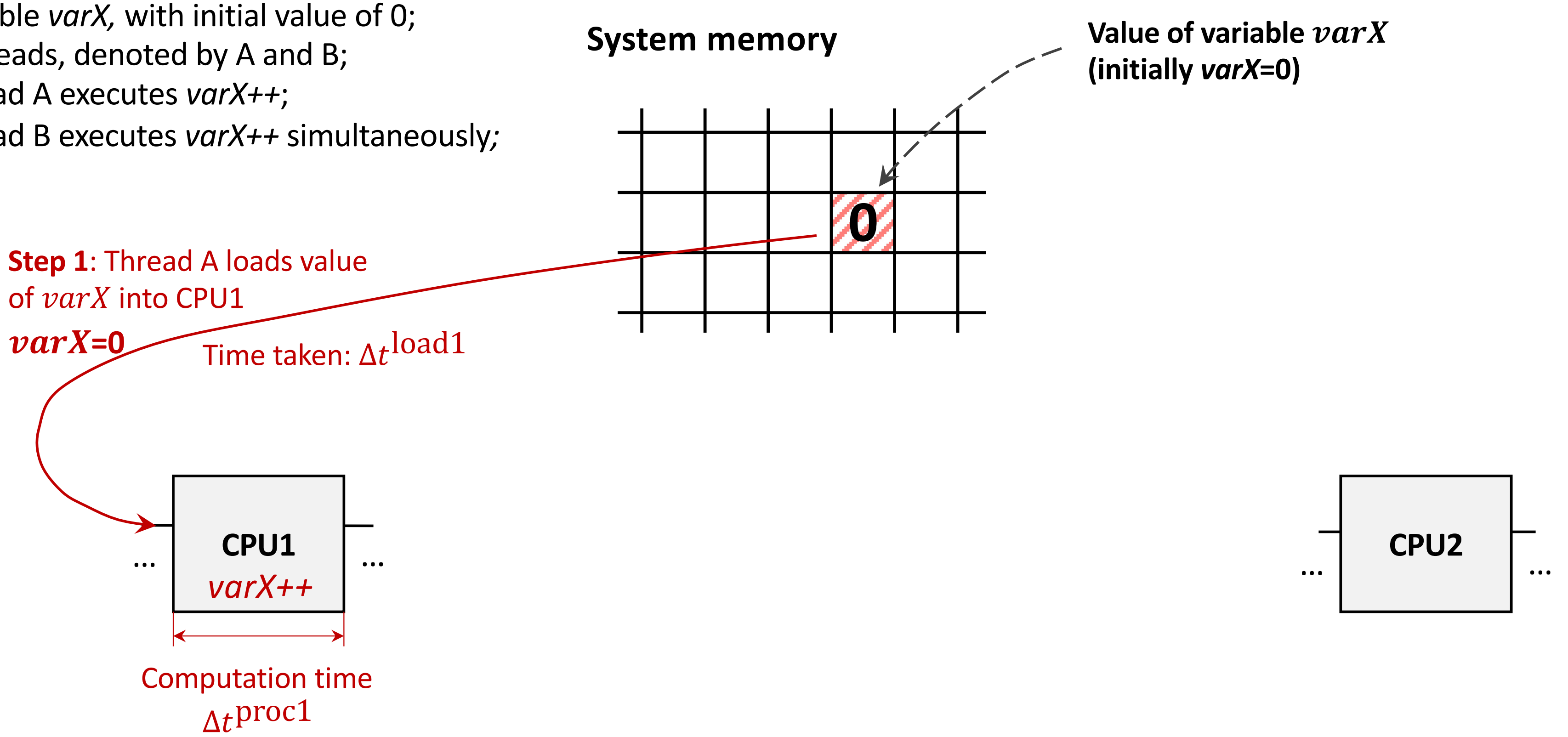
Selected Courses Taught at IU

Course Name	*Some aspects studied*
Computer Architecture	Hardware layout of a computer (processor, communication bus, memory units, etc.); Software/hardware interaction; Key ideas of modern computers: multiprocessing, caching, execution by prediction, etc.
Data Structures and Algorithms	Classical algorithms in Computer Science (e.g. sorting, graph traversal); Algorithms analysis (e.g. runtime and space complexity, asymptotic analysis); Data structures (e.g. priority queues, hash tables)
Computer Networks	Communication and routing protocols for computer networks; Client-server and peer-to-peer applications; Communication security
Operating Systems	Operating system design (incl. kernel, file system, I/O interaction); Processor scheduling algorithms, multithreading; Virtual memory
Introduction to Programming (in C language)	Basic programming concepts (loops, branching, etc.); Functional and object-oriented programming; Exception handling
Consensus Theory and Concurrent Programming on a Shared Memory	Synchronization of concurrent threads; Concurrent thread-safe implementation for data structures (e.g. FIFO queue, hash tables); Limitations of existing hardware architectures for concurrent programming

Given:
Variable *varX,* with initial value of 0;
2 threads, denoted by A and B;
Thread A executes *varX++*;
Thread B executes *varX++* simultaneously*;*

What is the result of this execution?

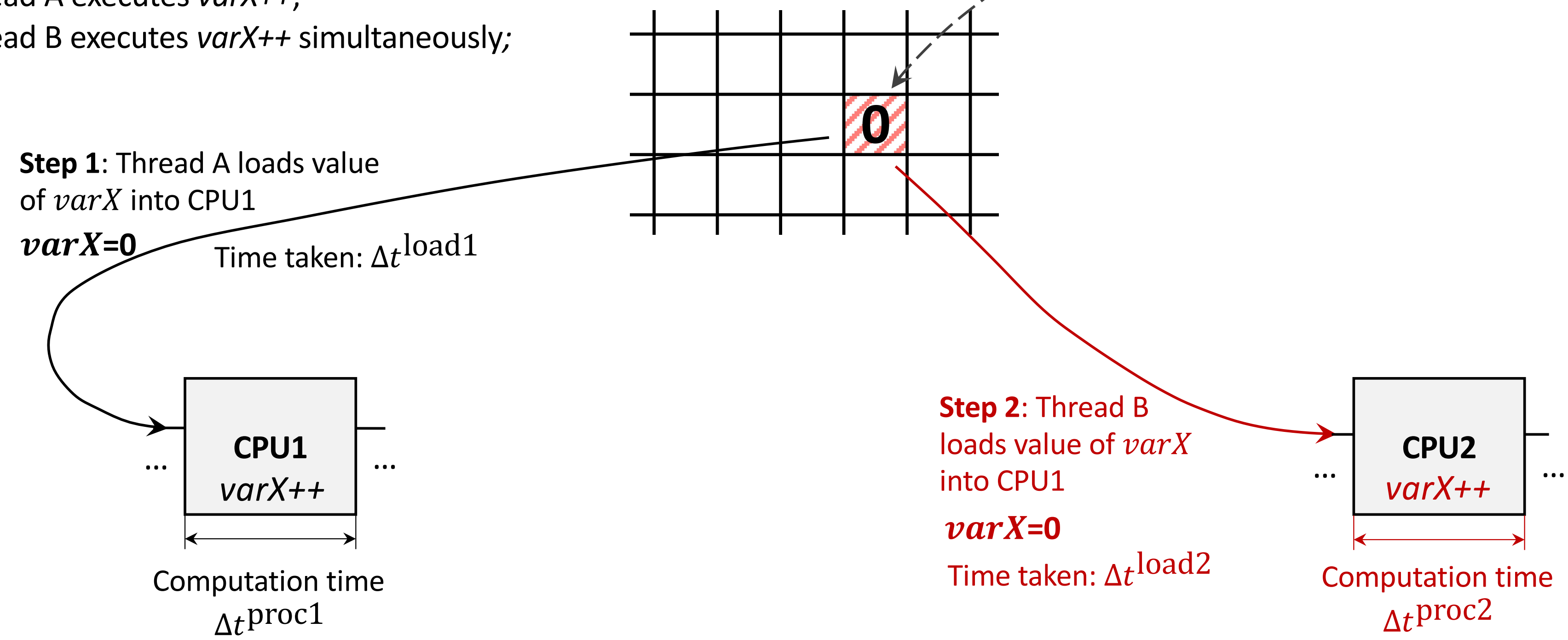
Given:
Variable *varX,* with initial value of 0;
2 threads, denoted by A and B;
Thread A executes *varX++*;
Thread B executes *varX++* simultaneously*;*



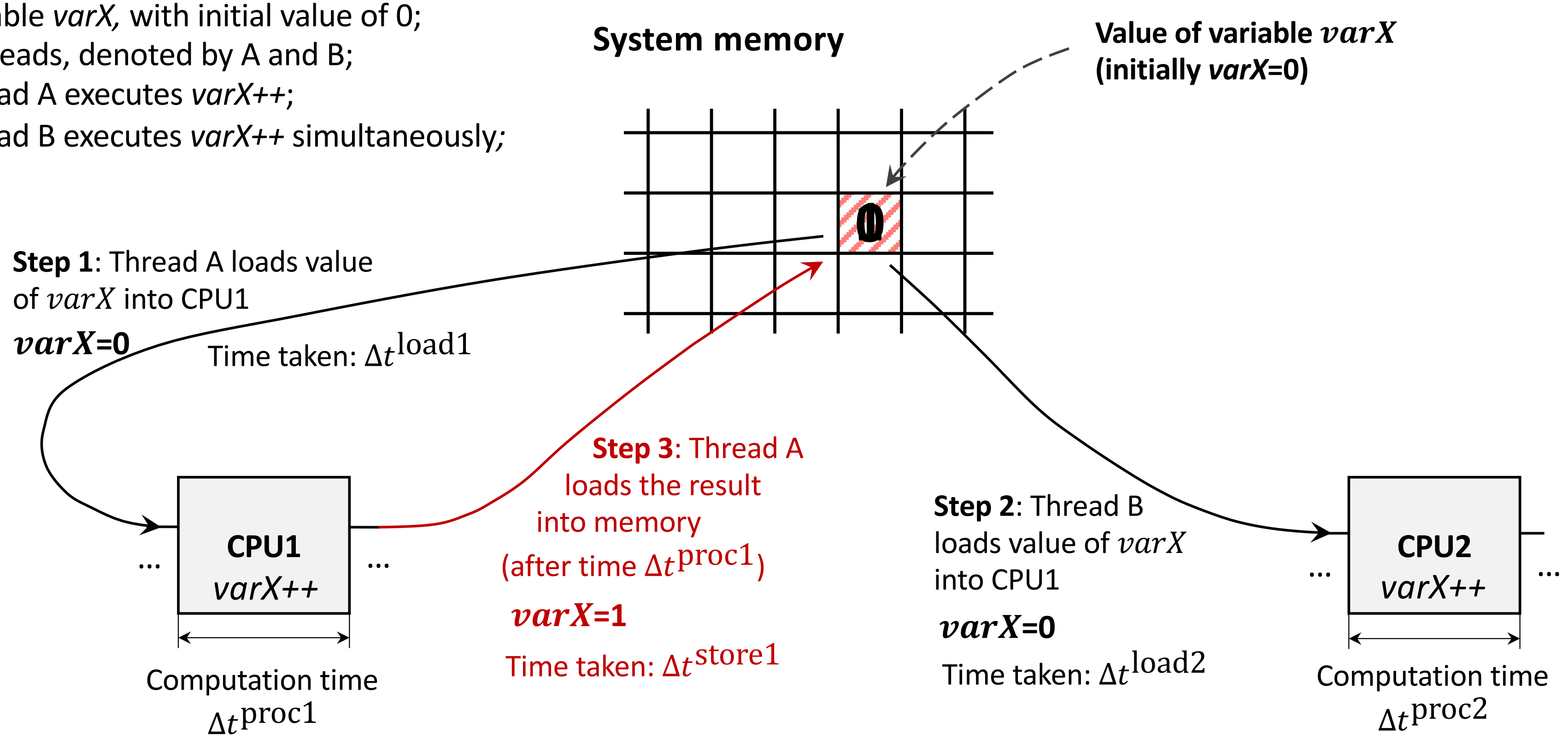
Given:
Variable *varX,* with initial value of 0;
2 threads, denoted by A and B;
Thread A executes *varX++;*
Thread B executes *varX++* simultaneously*;*

System memory

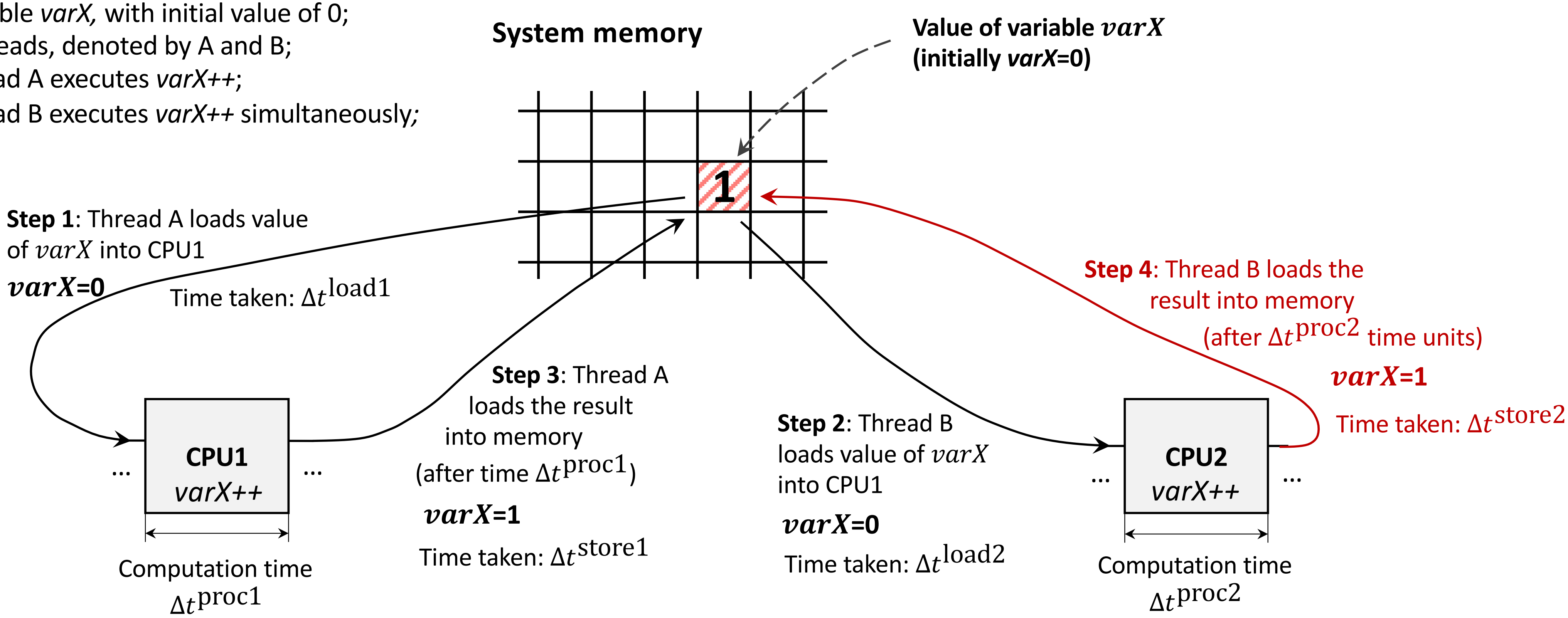
Value of variable $varX$
(initially *varX*=0)



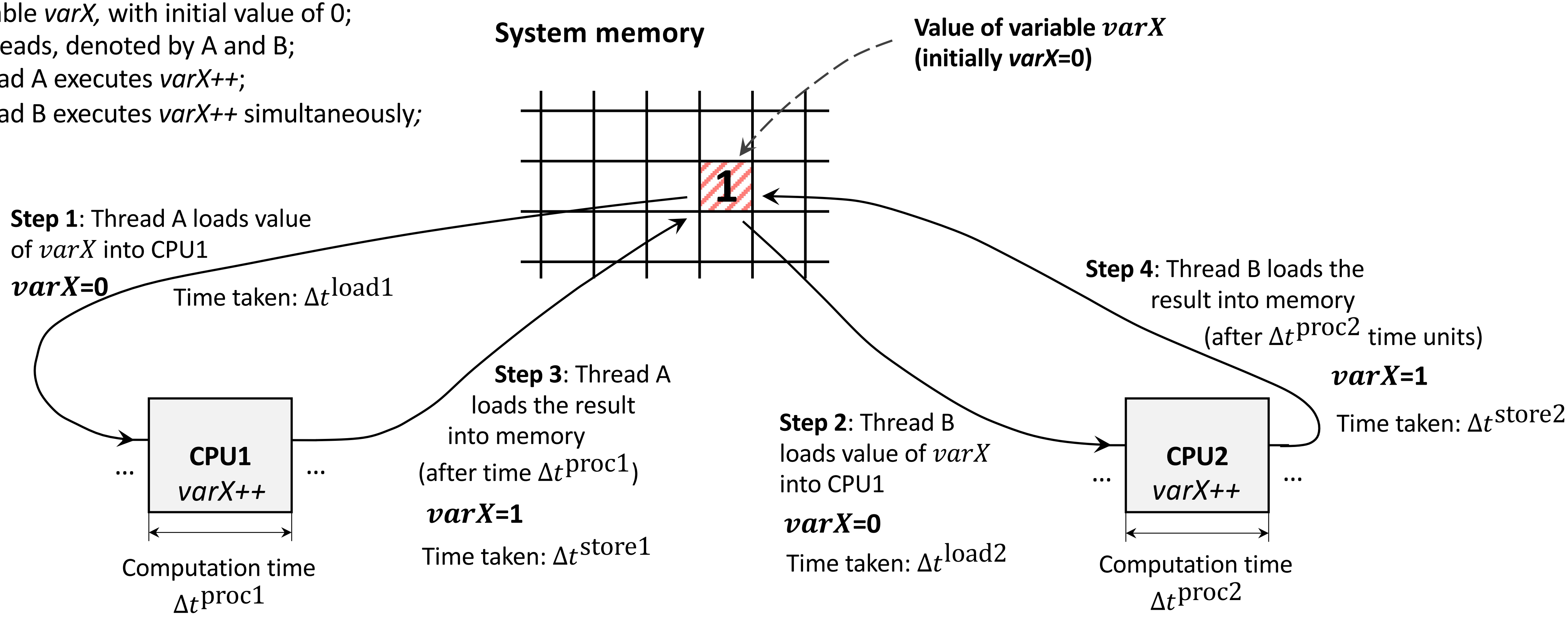
Given:
Variable *varX*, with initial value of 0;
2 threads, denoted by A and B;
Thread A executes *varX++*;
Thread B executes *varX++* simultaneously;



Given:
Variable *varX*, with initial value of 0;
2 threads, denoted by A and B;
Thread A executes *varX++*;
Thread B executes *varX++* simultaneously;



Given:
Variable *varX,* with initial value of 0;
2 threads, denoted by A and B;
Thread A executes *varX++*;
Thread B executes *varX++* simultaneously;



Race Condition problem

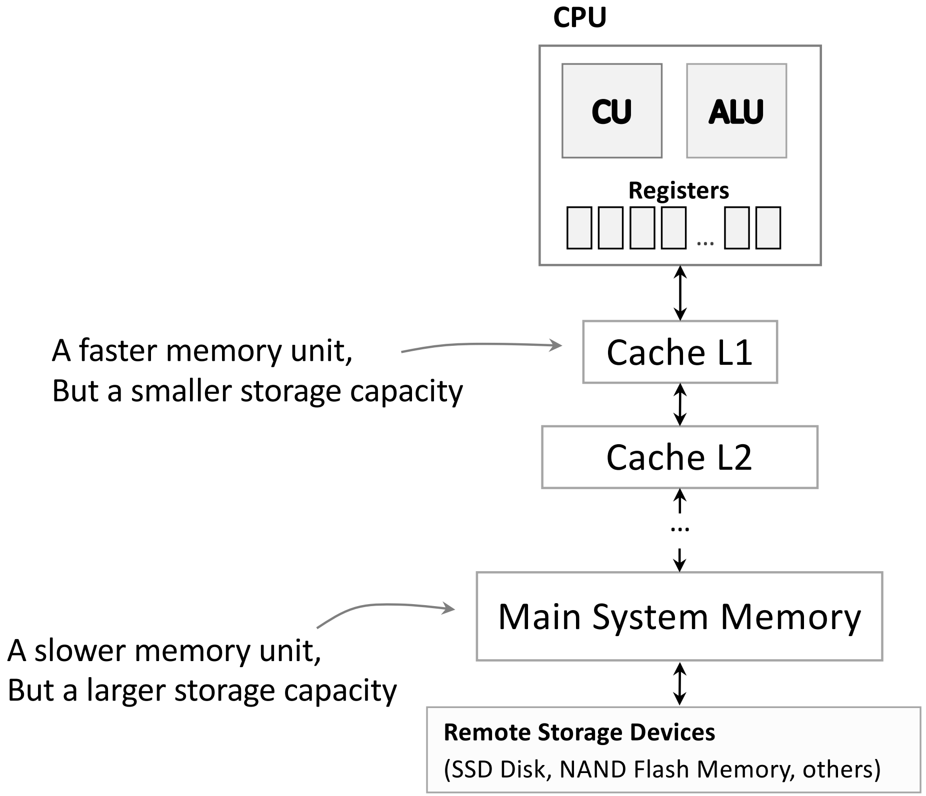
Selected Courses Taught at IU

Course Name	*Some aspects studied*
Computer Architecture	Hardware layout of a computer (processor, communication bus, memory units, etc.); Software/hardware interaction; Key ideas of modern computers: multiprocessing, caching, execution by prediction, etc.
Data Structures and Algorithms	Classical algorithms in Computer Science (e.g. sorting, graph traversal); Algorithms analysis (e.g. runtime and space complexity, asymptotic analysis); Data structures (e.g. priority queues, hash tables)
Computer Networks	Communication and routing protocols for computer networks; Client-server and peer-to-peer applications; Communication security
Operating Systems	Operating system design (incl. kernel, file system, I/O interaction); Processor scheduling algorithms, multithreading; Virtual memory
Introduction to Programming (in C language)	Basic programming concepts (loops, branching, etc.); Functional and object-oriented programming; Exception handling
Consensus Theory and Concurrent Programming on a Shared Memory	Synchronization of concurrent threads; Concurrent thread-safe implementation for data structures (e.g. FIFO queue, hash tables); Limitations of existing hardware architectures for concurrent programming

Other courses (some are electives): Compiler Construction, Databases, Information Retrieval, etc.

Some Trends in Computer Science / Program Engineering
(among many others; a subjective selection)

- Multiprocessor systems and scheduling
- Parallel and distributed systems
- Concurrent programming
- Real-time systems and scheduling
- Energy-efficient systems
- Cache-prefetching algorithms

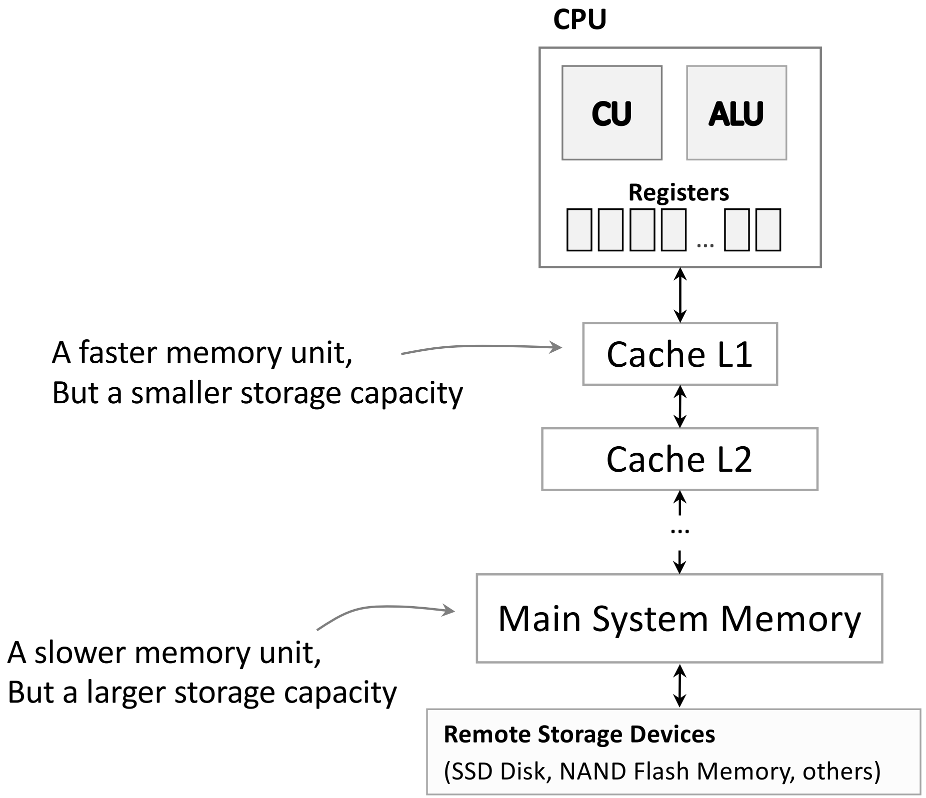


Cache-prefetching problem:
To determine which data is more likely to be needed by CPU next, to be transferred into L1 cache in advance

Some Trends in Computer Science / Program Engineering

(among many others; a subjective selection)

- Multiprocessor systems and scheduling
- Parallel and distributed systems
- Concurrent programming
- Real-time systems and scheduling
- Energy-efficient systems
- Cache-prefetching algorithms

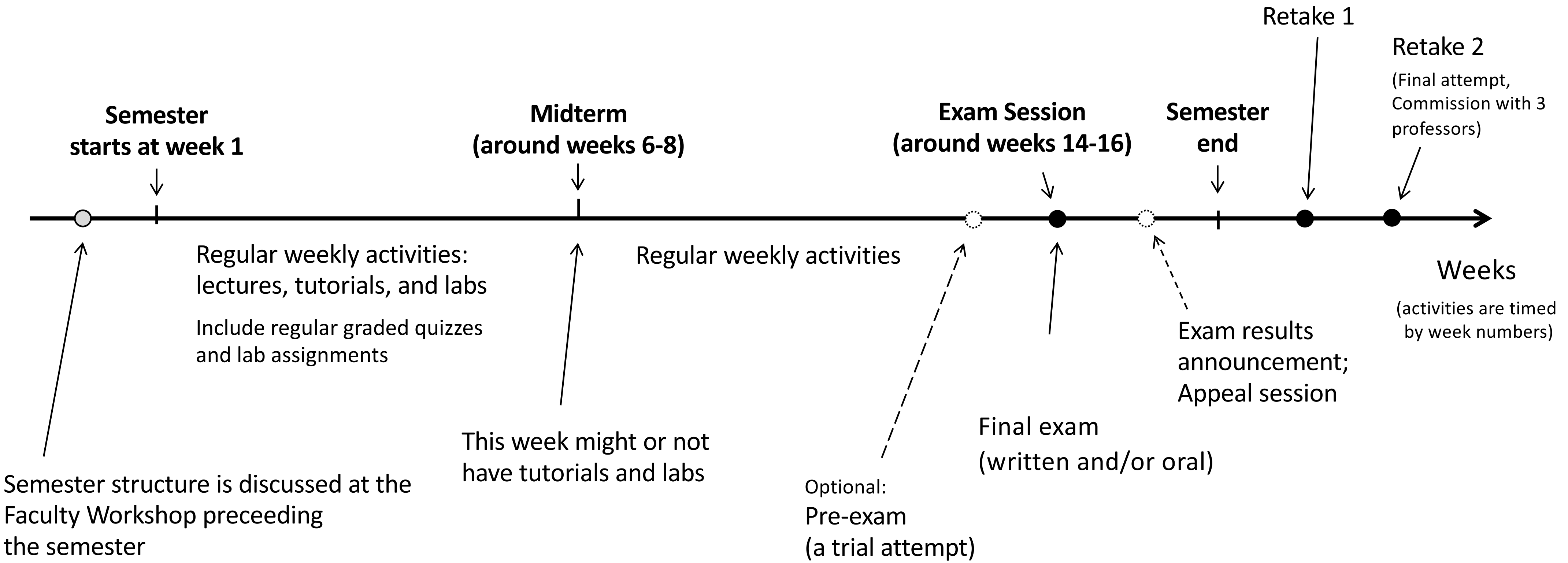


Cache-prefetching problem:
To determine which data is more likely to be needed by CPU next, to be transferred into L1 cache in advance

Other “trendy” topics: Artificial Intelligence, Blockchain, etc.

A Typical Course Structure for Bachelors

(The example of Computer Architecture; differences apply to other courses)



Grading Policy for Our Course

Course Component	*Component Weight in Final Grade*
Regular quizzes **(excluding one worst quiz)**	**40%**
Lab assignments, Attendance (late delivery is not accepted)	*N/A*
Midterm *(exam in the middle of the semester)*	*N/A*
Final exam	**60%**
Optional project or activity (e.g. with FPGA board for Computer Architecture course)	*N/A*
Instructor discretion (extra points) **(e.g. for optional homework assignments and active participation)**	***5% (attendance) +*** ***5% (participation)***

The summation equals to 100%

Notes:

- Course grade is first computed as a number (by the scale of 100), and then converted into a letter grade (“A”, “B”, “C”, or “D”)
- Each course component must score at least *k*%, otherwise the course is failed (granted grade is “D”);