CSE502: Foundations of Parallel Programming

Lecture 01: Course Overview, Evaluation Style, Rules and Regulations

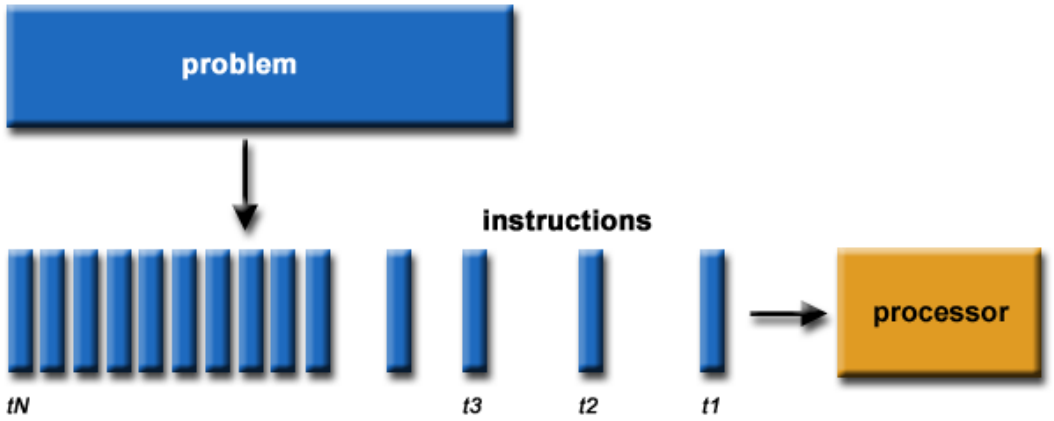
Vivek Kumar

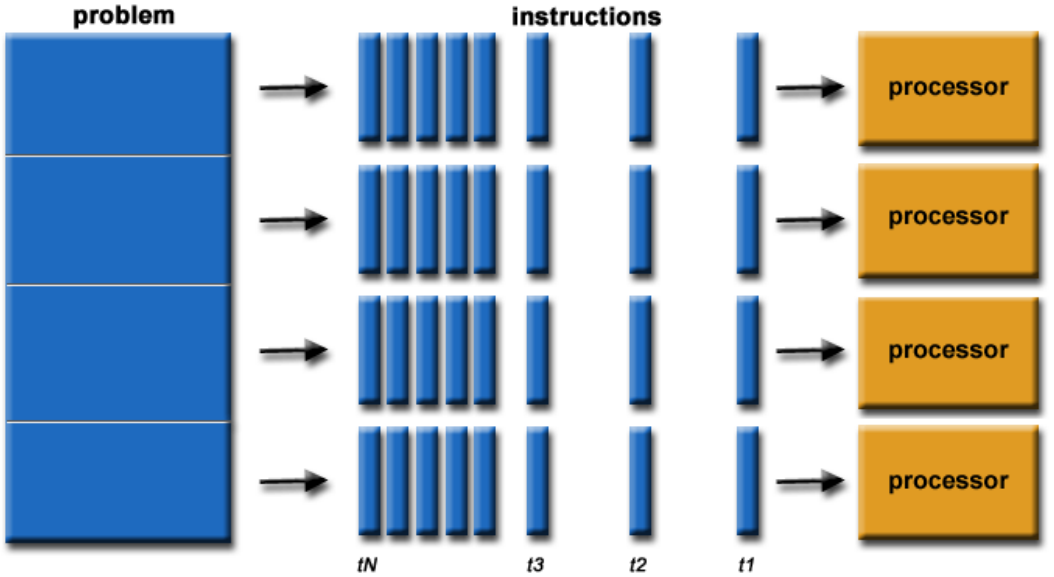
Computer Science and Engineering

IIIT Delhi

vivekk@iiitd.ac.in

What is Parallel Programming?





Source: https://computing.llnl.gov/tutorials/parallel_comp/

- Serial
  - One instruction at a time
- Parallel
  - Multiple instructions in parallel

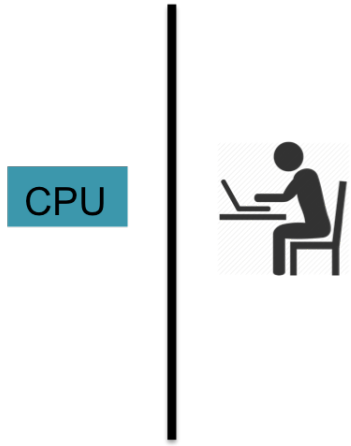
Course Overview

1. Why the hell do we need multicore processors

Why Parallel Programming?

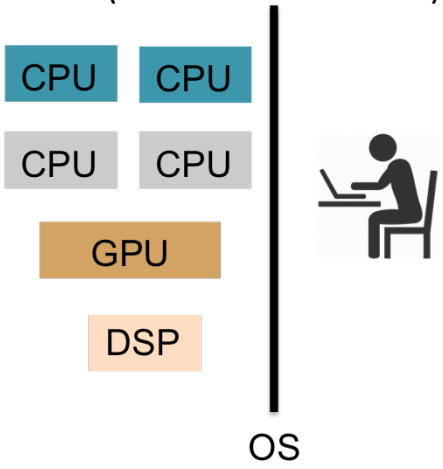
Technology Push

1990's and early 2000s

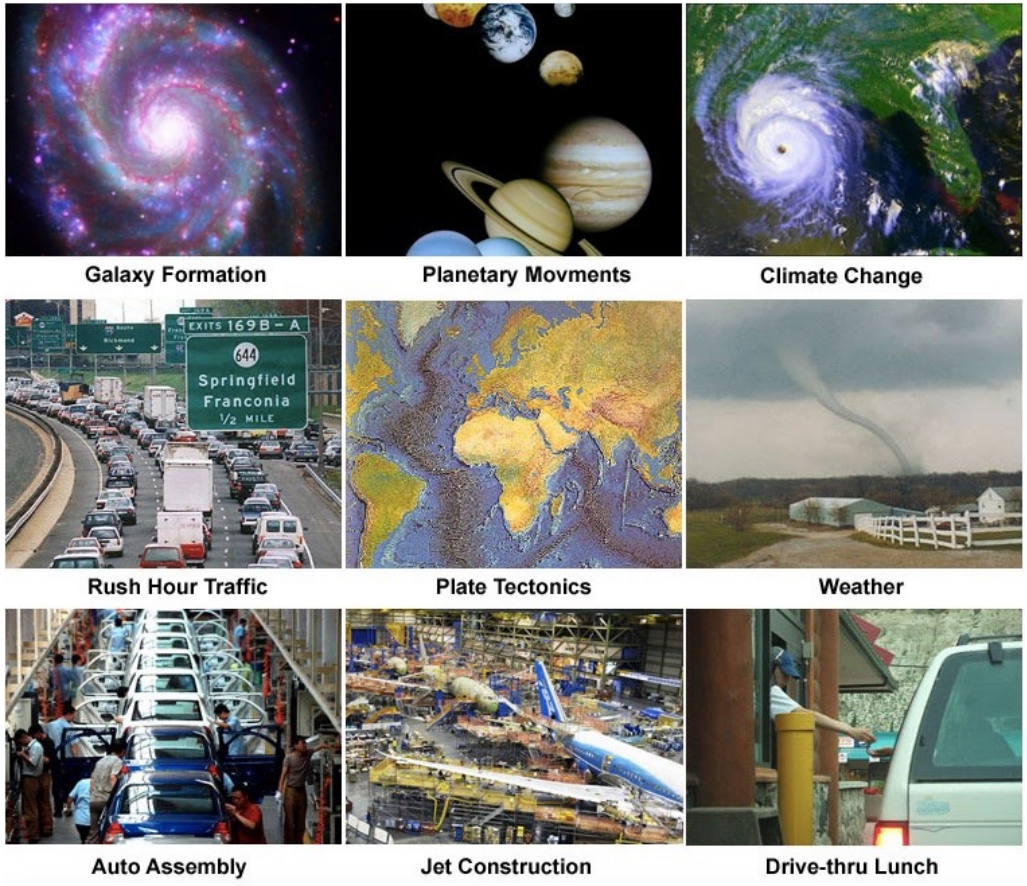


Today

(no more free lunch)



Application Push



- Complex problems require computation on large-scale data
- Sufficient performance available only through massive parallelism

Course Overview

1. Why the hell do we need multicore processors
2. How can we decompose a sequential program into a parallel program

Can we Parallelize these Programs?

```
uint64_t array_sum(uint64_t* array, uint64_t size) {
  uint64_t sum = 0;
  for(uint64_t i=0; i<size; i++) {
    sum = sum + array[i];
  }
  return sum;
}
```

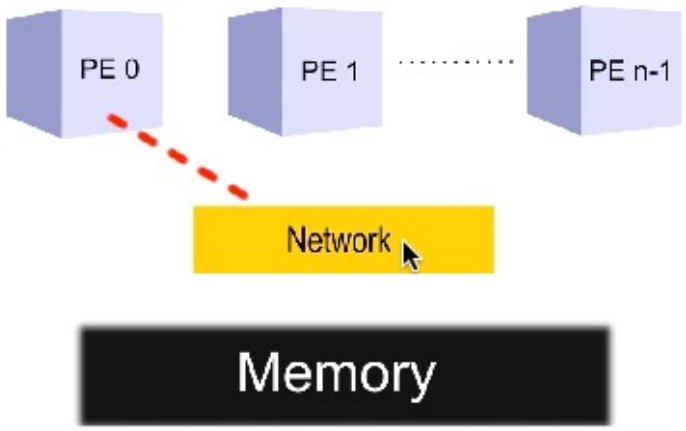
```
uint64_t fibonacci(uint64_t n) {
  if (n < 2) {
    return n;
  } else {
    uint64_t x = fibonacci(n-1);
    uint64_t y = fibonacci(n-2);
    return (x + y);
  }
}
```

Course Overview

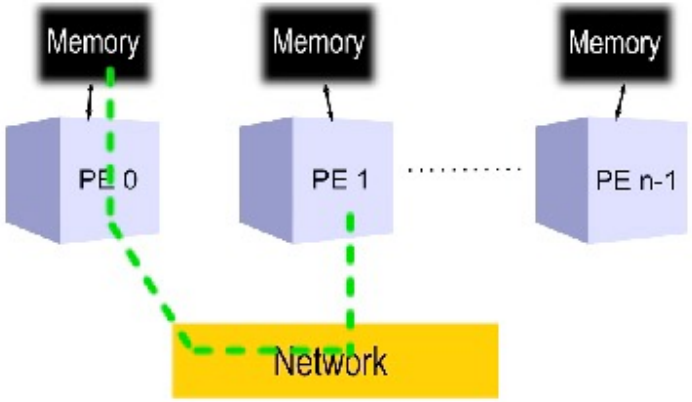
1. Why the hell do we need multicore processors
2. How can we decompose a sequential program into a parallel program
3. What are the different parallel programming models

Which all Parallel Programming Models?

- Shared memory parallel programming models (4)
  - Pthread, OpenMP, and Habanero-C/C++ library (HClib)
  - ForkJoinPool (Java)



- Distributed memory parallel programming (4)
  - MPI, MPI+OpenMP, UPC++, and HabaneroUPC++



Source: http://cnx.org/contents/82d83503-3748-4a69-8d6c-50d34a40c2e7@7

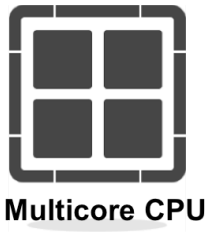
Course Overview

1. Why the hell do we need multicore processors
2. How can we decompose a sequential program into a parallel program
3. What are the different parallel programming models
   - Shared memory
   - Distributed memory
   - Hybrid shared and distributed memory
4. Productivity and performance of different parallel programming models

Productivity and Performance

- **Productivity** = how easily you can convert a sequential program into a parallel program
- **Performance** = how you can write a parallel program that takes minimum time to execute

```
uint64_t array_sum(uint64_t* array, uint64_t size) {
  uint64_t sum = 0;
  for(uint64_t i=0; i<size; i++) {
    sum = sum + array[i];
  }
  return sum;
}
```



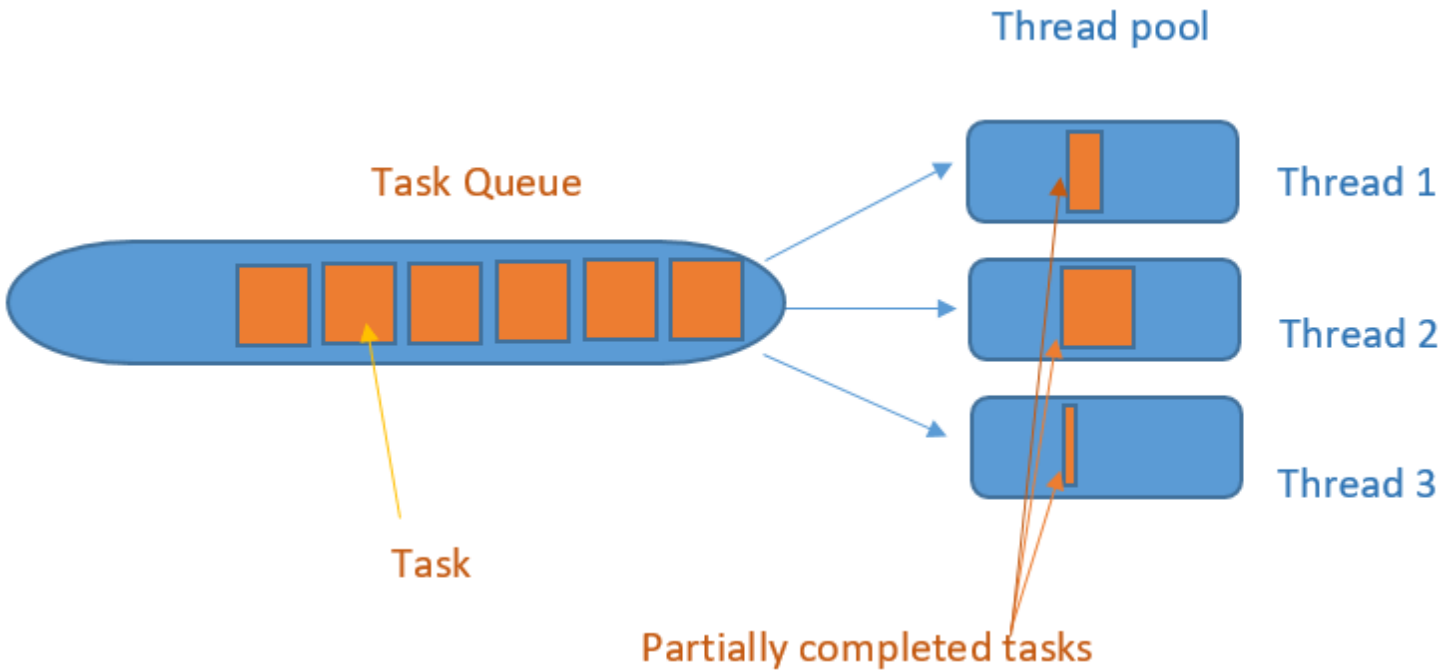
Pthread ?
ForkJoinPool ?
OpenMP ?
Habanero-C ?

MPI ?
MPI+OpenMP ?
UPC++ ?
HabaneroUPC++ ?

Course Overview

1. Why the hell do we need multicore processors
2. How can we decompose a sequential program into a parallel program
3. What are the different parallel programming models
   - Shared memory
   - Distributed memory
   - Hybrid shared and distributed memory
4. Productivity and performance of different parallel programming models
5. How to make effective use of available multicore processors
   - Designing thread pool based runtimes that uses load balancing algorithms such as work-stealing and work-sharing

Thread Pool Based Load Balancing Runtimes

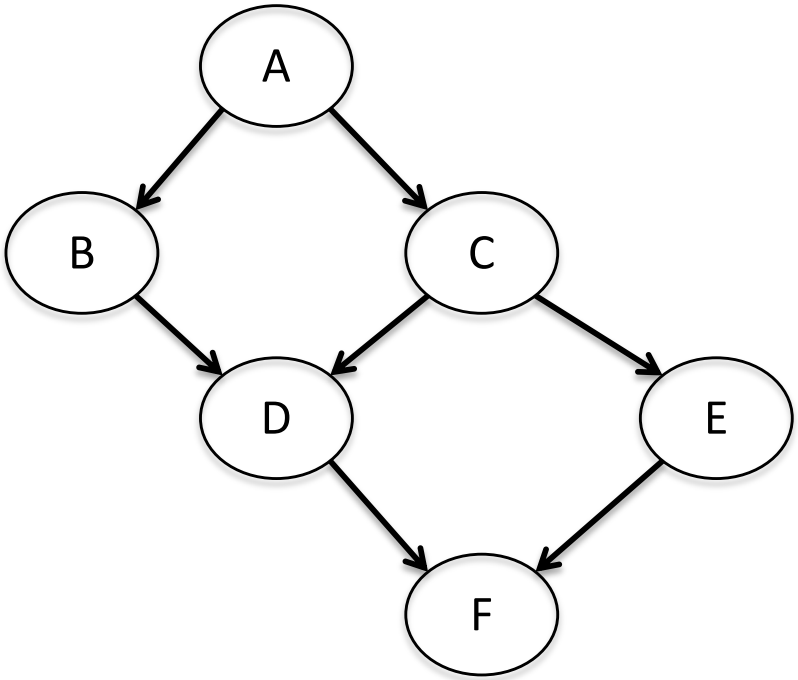


- How to design a thread pool?

Course Overview

1. Why the hell do we need multicore processors
2. How can we decompose a sequential program into a parallel program
3. What are the different parallel programming models
   - Shared memory
   - Distributed memory
   - Hybrid shared and distributed memory
4. Productivity and performance of different parallel programming models
5. How to make effective use of available multicore processors
   - Designing thread pool based runtimes that uses load balancing algorithms such as work-stealing and work-sharing
6. Data flow programming model using C++11 futures, promises, and task dependencies

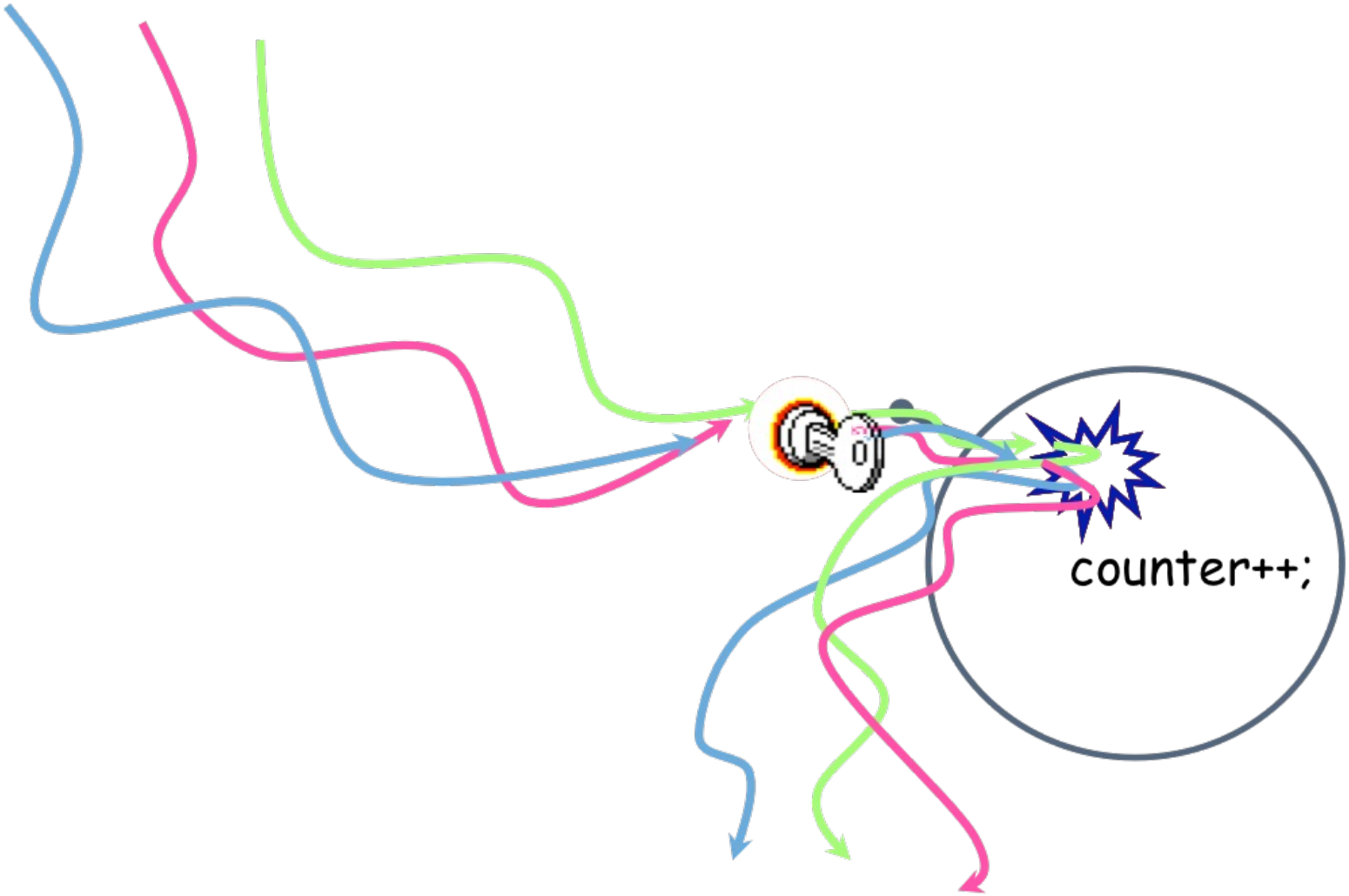
Dataflow Programming



Course Overview

1. Why the hell do we need multicore processors
2. How can we decompose a sequential program into a parallel program
3. What are the different parallel programming models
   - Shared memory
   - Distributed memory
   - Hybrid shared and distributed memory
4. Productivity and performance of different parallel programming models
5. How to make effective use of available multicore processors
   - Designing thread pool based runtimes that uses load balancing algorithms such as work-stealing and work-sharing
6. Data flow programming model using C++11 futures, promises, and task dependencies
7. Mutual exclusion in tasks based programming model

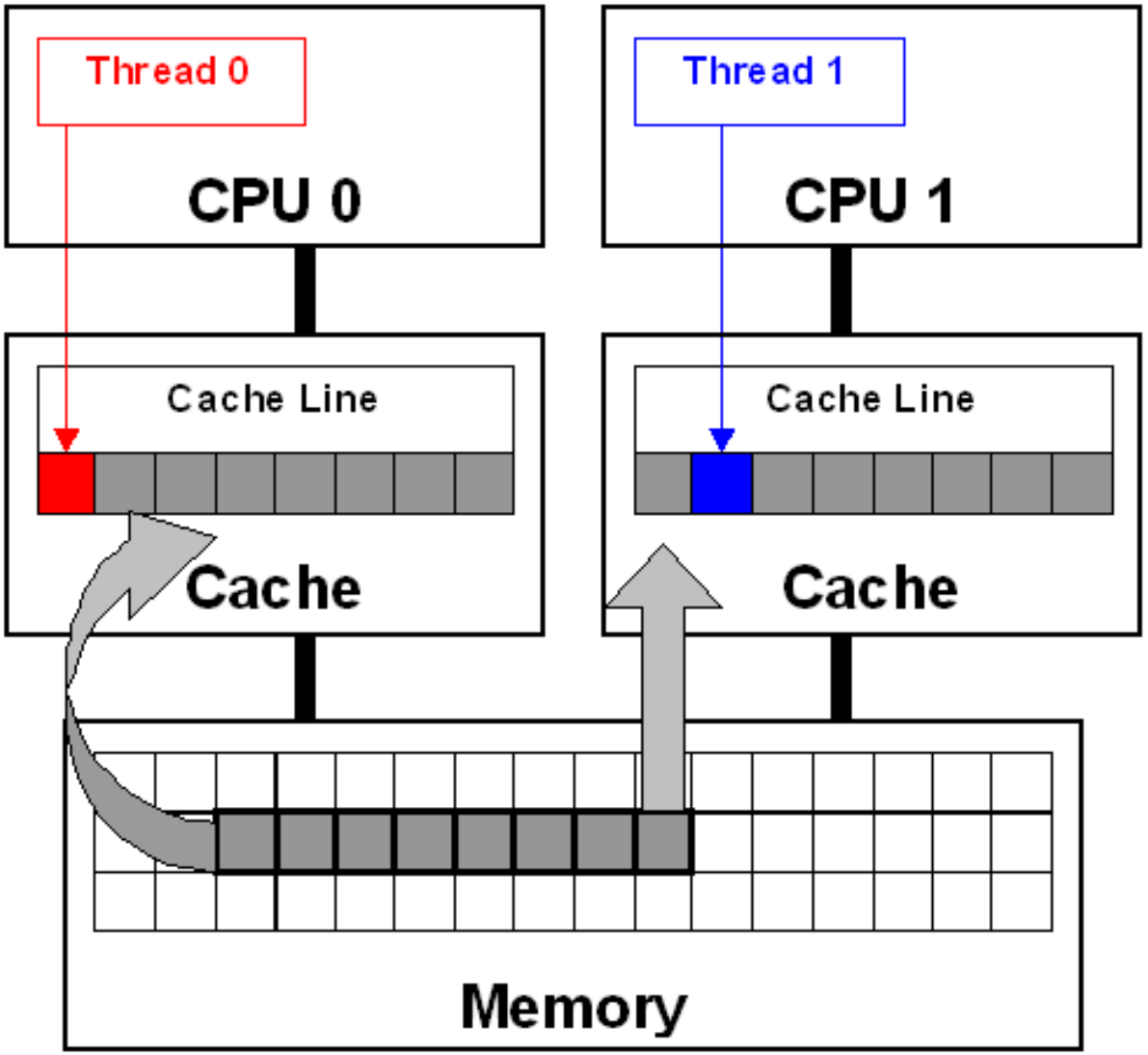
Mutual Exclusion



Course Overview

1. Why the hell do we need multicore processors
2. How can we decompose a sequential program into a parallel program
3. What are the different parallel programming models
   - Shared memory
   - Distributed memory
   - Hybrid shared and distributed memory
4. Productivity and performance of different parallel programming models
5. How to make effective use of available multicore processors
   - Designing thread pool based runtimes that uses load balancing algorithms such as work-stealing and work-sharing
6. Data flow programming model using C++11 futures, promises, and task dependencies
7. Mutual exclusion in tasks based programming model
8. How does caches affect the performance of parallel program on a shared memory architectures

Effects of Caches on the Performance of Parallel Program



Course Overview

1. Why the hell do we need multicore processors
2. How can we decompose a sequential program into a parallel program
3. What are the different parallel programming models
   - Shared memory
   - Distributed memory
   - Hybrid shared and distributed memory
4. Productivity and performance of different parallel programming models
5. How to make effective use of available multicore processors
   - Designing thread pool based runtimes that uses load balancing algorithms such as work-stealing and work-sharing
6. Data flow programming model using C++11 futures, promises, and task dependencies
7. Mutual exclusion in tasks based programming model
8. How does caches affect the performance of parallel program on a shared memory architectures
9. Student run lectures (1-2) for research paper presentations

Course Prerequisites

- **Programming in C/C++ is a must**!
  - If you don't know C/C++ then you should be confident that you can pick it up on your own
- Basics of Operating Systems and Data-structures

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

Textbook

- **None**
  - Be sure to attend all the lectures!
- Course material derived from multiple sources
- Course notes / references will be provided depending on the lecture
- References will also be mentioned on the last slide in each lecture

Course Logistics

- Machines for your labs and assignments
  - You can use your laptop, but would require Linux OS (or any VM running Linux OS)
- In case you have problems please feel free to shoot me an email

Next Class

- Refresher on processes and threads