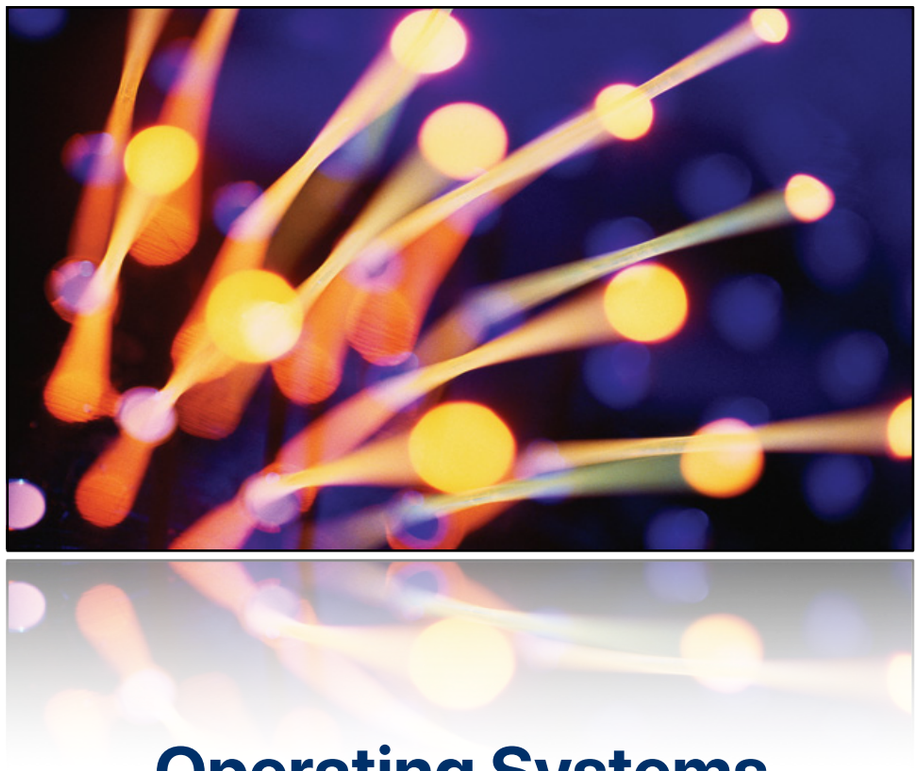
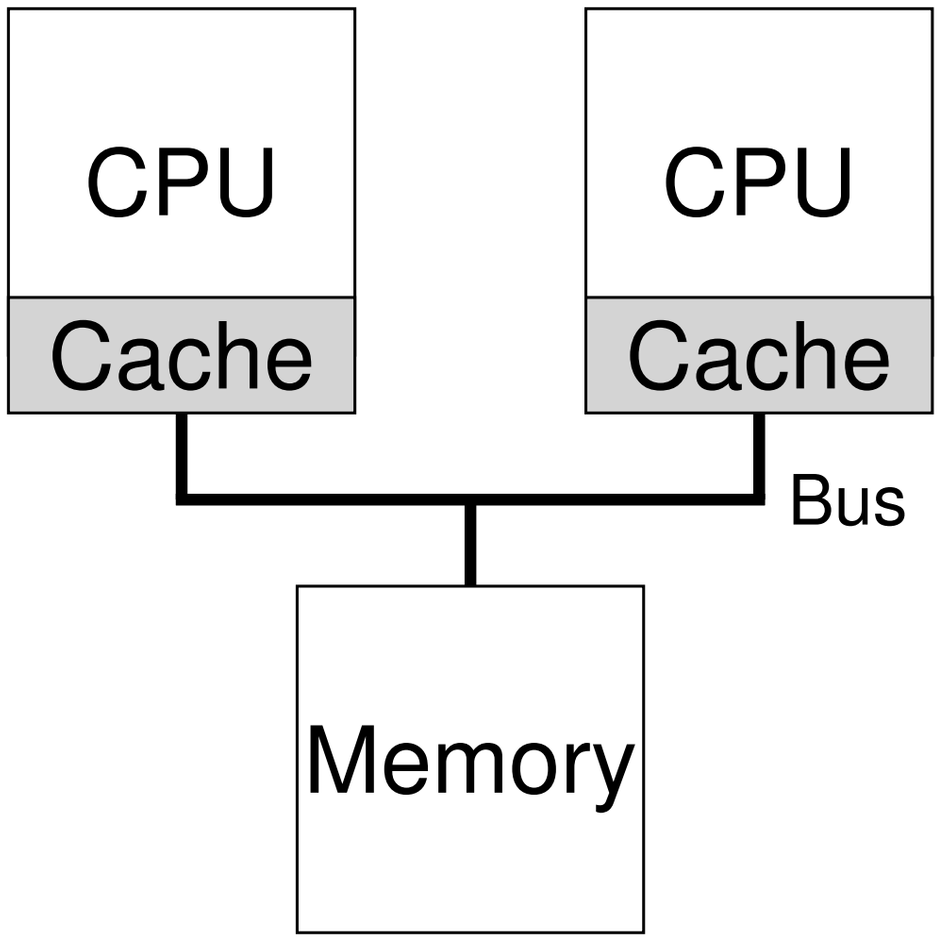
# Multiprocessor Scheduling



**Operating Systems**

Baochun Li
University of Toronto

# Multiprocessor architecture

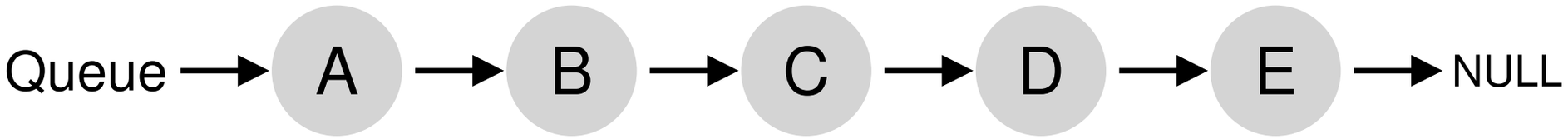


# Cache (processor) affinity

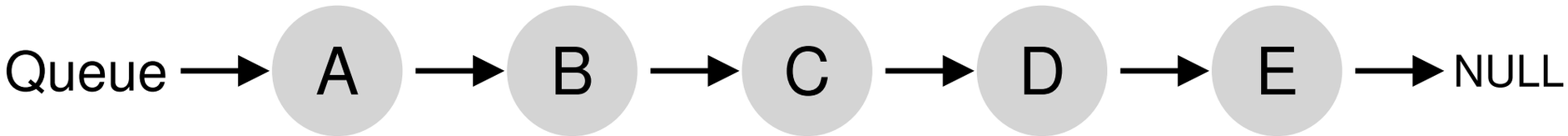
A process builds up some state in the caches (and TLBs, which we will discuss later in the course) of the CPU

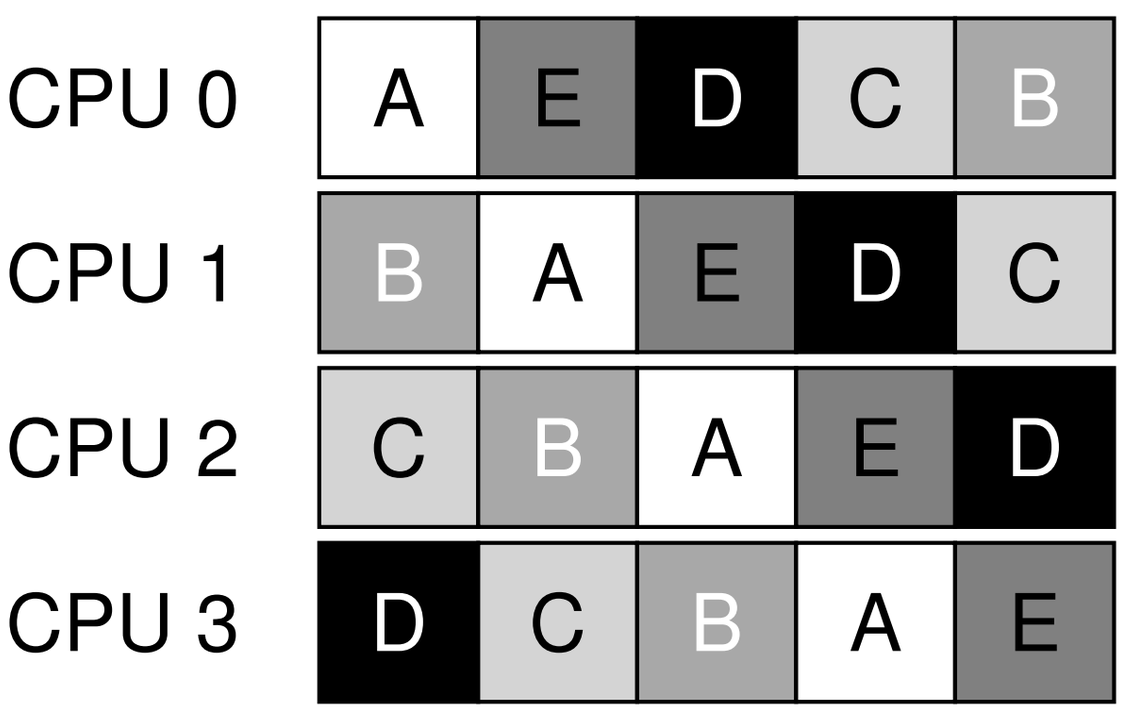
The next time the process runs, it is a good idea to run it on the same CPU

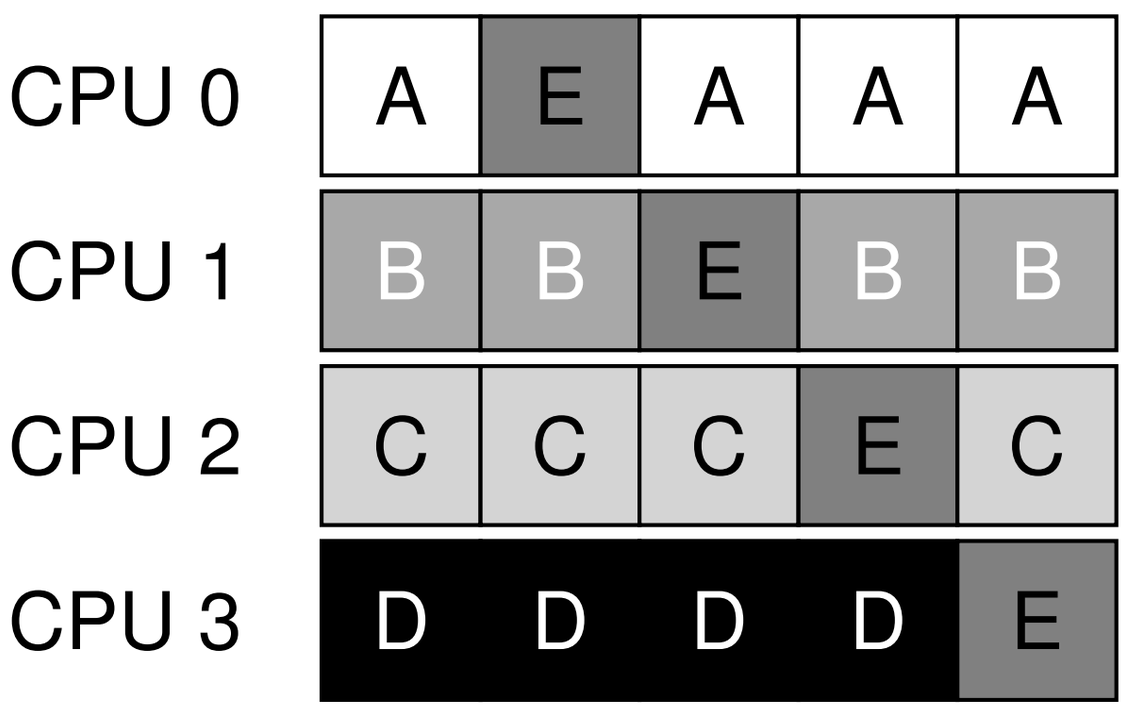
# Single-Queue Multiprocessor Scheduling



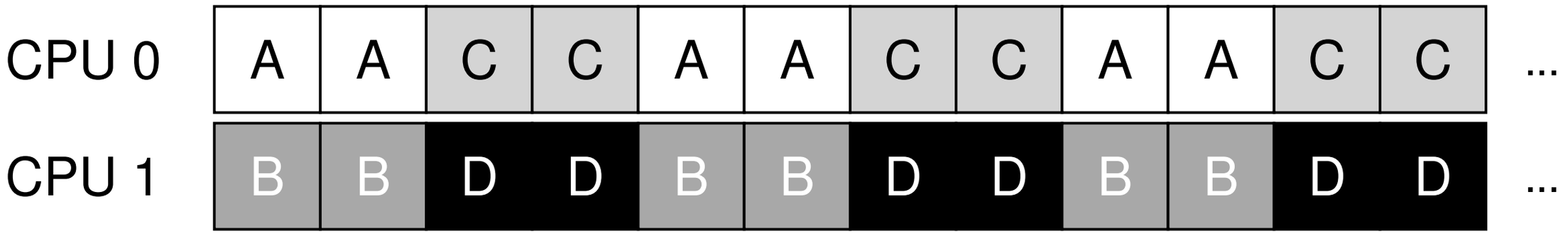
# Cache Affinity with Single-Queue Scheduling



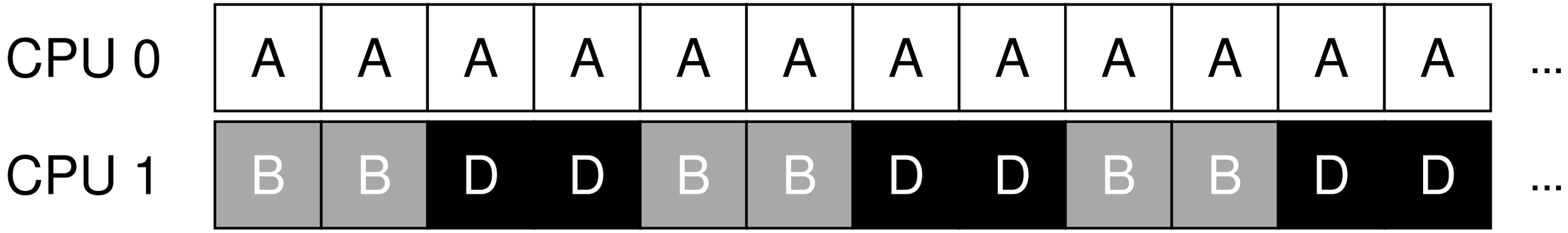



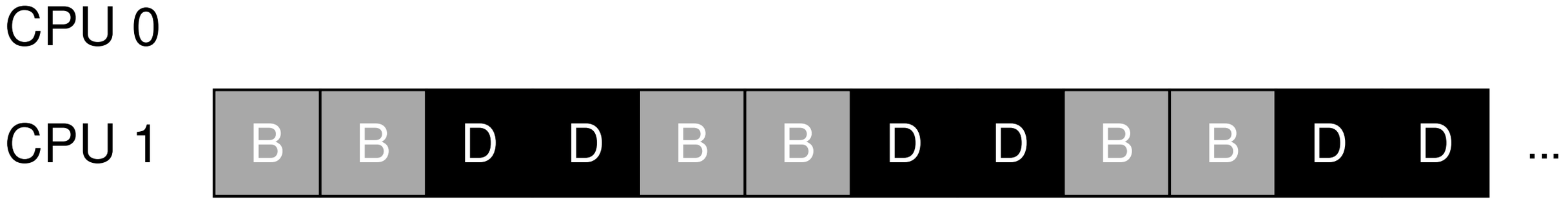
# Multi-Queue Multiprocessor Scheduling

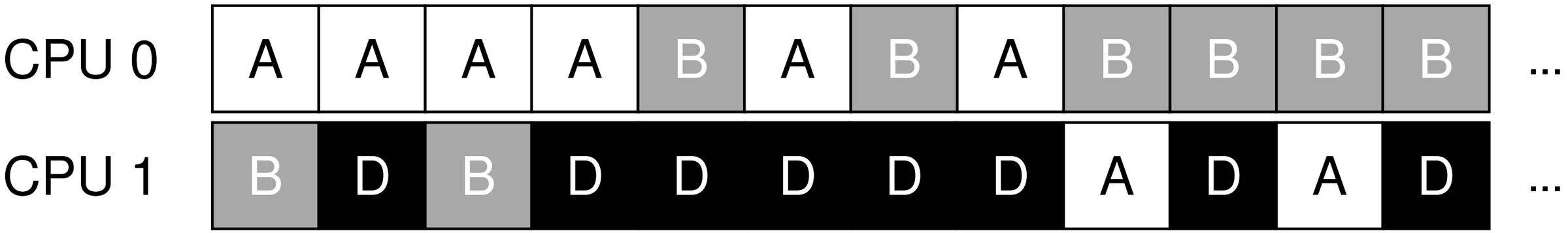
# Load Imbalance

# Load Imbalance can get even worse

# Possible fix: keep switching jobs across CPUs



# Summary

## The Linux O(1) and CFS schedulers use multi-queue scheduling: one "runqueue" per processor

## There is a "conflict of interest" —

Migrating jobs from one CPU to another requires a cost of invalidating and repopulating caches — so we don't wish to do this often

On the other hand, we don't wish to leave a CPU idle while another CPU is too busy with all its jobs

## Cache affinity: try to avoid migration of jobs from one CPU to another

## Load balancing: try to keep the workload evenly distributed

# What we've covered so far

## Three Easy Pieces, Chapter 10 (Multiprocessor Scheduling)