Multicore Programming

Outline

- What is parallel computing
- How create a parallel computing

Computation

- **Serial computation**
  - A problem is broken into a discrete series of instructions.
  - Instructions are executed one after another on CPU.
- **Parallel computation**
  - Dividing a problem into discrete parts
  - Each part can be solved concurrently
  - Instructions from each part execute simultaneously on different processing elements

How create a Parallel Program

- Decomposition
- Assignment
- Orchestration
- Mapping

Issues with Parallel Computing

- Problem Decomposition
  - Domain decomposition
  - Functional decomposition
- Data dependency and communication
- Synchronization
- Parallel execution

Data/ Control Parallelism

```
#pragma omp parallel
#pragma omp for
  for(i = 0; i < 12; i++)
      C[i] = A[i] + B[i];
```

```
pthread_create(
    /* thread id */,
    /* attributes */,
    /* any function */,
    /* args to function */);
```

Data Dependency and Communication

- When two parts have data dependencies
  - cannot be executed in parallel
  - the order of the operations is critical
    - RAW, WAR, WAW
- When two parts need communication
  - to exchange data
  - to send a message
  - introduce overhead

Synchronization

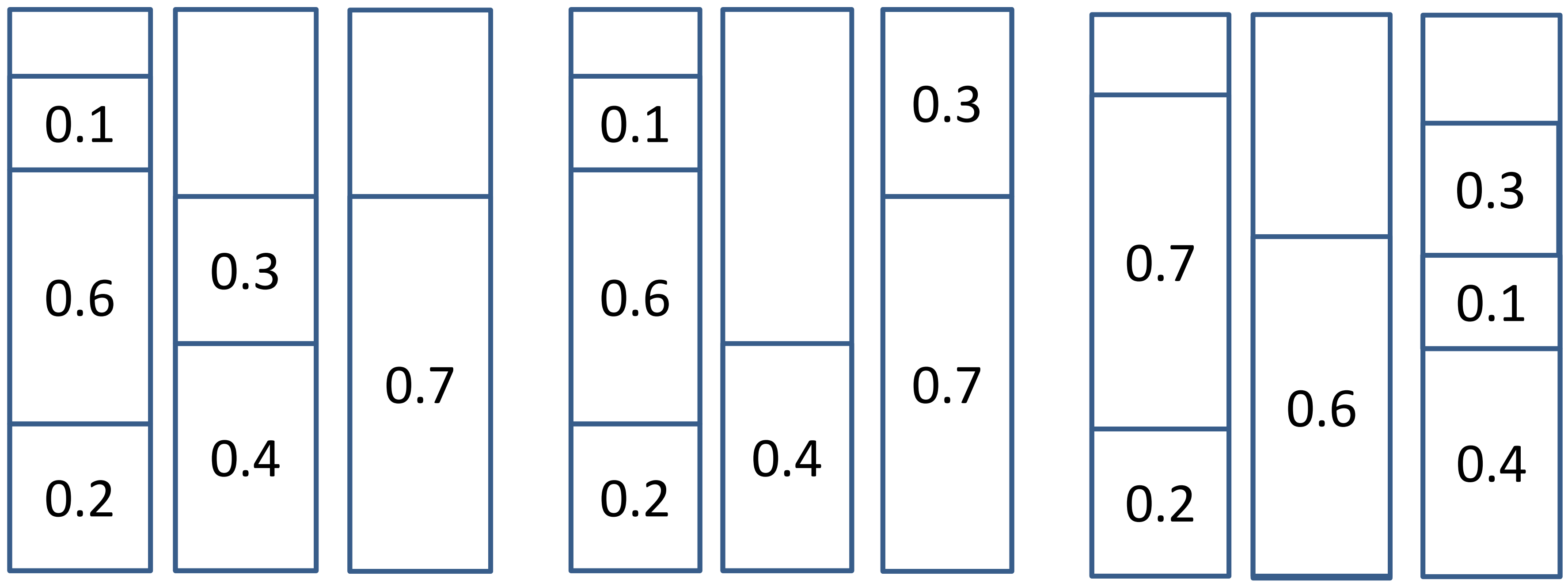
- Semaphore
- Mutex
- Interrupt disabling
- Spin lock

Parallel execution
Flynn’s Taxonomy

- Single Instruction stream, Single Data stream (SISD)
- Single Instruction stream, Multiple Data streams (SIMD)
- Multiple Instruction streams, Single Data stream (MISD)
- Multiple Instruction streams, Multiple Data streams (MIMD)
  - SPMD (Single Program, Multiple Data)
  - MPMD (Multiple Programs, Multiple Data)

Load Balance

- Which is load balance?



First Fit Best Fit Worst Fit

Static Load Balancing

- Assigns a fixed amount of work to each core in a prior
- Better for homogeneous multicores
  - All core are identical
  - All the processors have the same characteristics
- About heterogeneous multicores
  - Each task has its own execution time on a specified processor
  - A job might be executed faster on a processor, but other jobs might be slower on that processor.
- Examples: Loop, array

Dynamic Load Balancing

- Assigns work among processors at runtime
- Better for heterogeneous multicore
- Dynamic load balancing is needed when static load balancing is difficult, e.g., Sparse arrays

Data Locality

- Principle of data locality:
  - Task accesses a relatively small portion of the address space at continuous time
- Temporal locality (locality in time)
  - e.g. instruction and data in a loop
  - Parallel computation is serialized due to memory contention and lack of bandwidth
- Spatial locality (locality in space)
  - e. g. instruction are normally accessed sequentially, good spatial locality
  - how to allocate tasks and assign data to cores

Performance of Parallel Computing

- Coverage
- Granularity
- Synchronization
- Communications
- Load balance
- Locality