LAU
الجامعة اللبنانية الأميركية
Lebanese American University

LAU
School of Engineering

Introduction to Parallel Computing

Parallel Programming Techniques and Applications Using Networked Workstations and Parallel Computers

Chapter 1

Dr. Hamdan Abdellatef

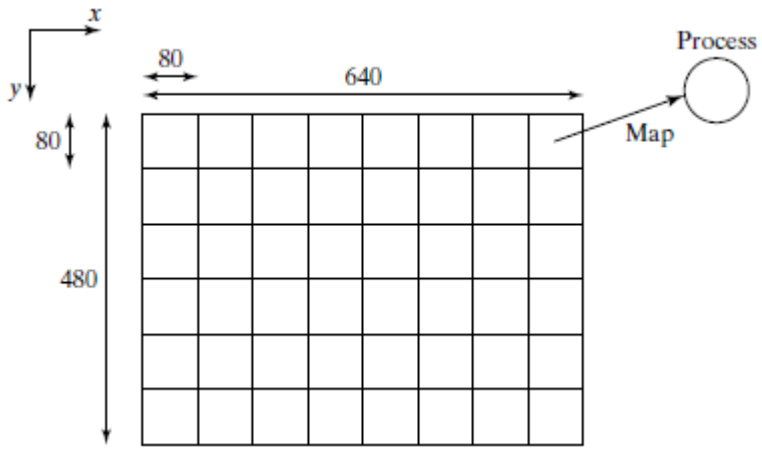
LAU
الجامعة اللبنانية الأميركية
Lebanese American University

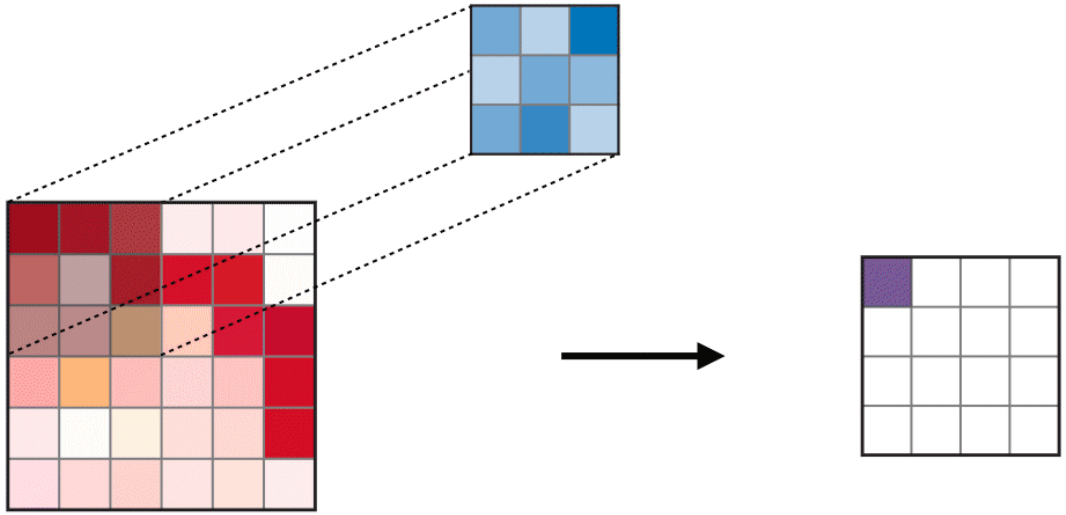
LAU
School of Engineering

Parallel Programming Workflow

Parallel Programming Workflow

- Identify compute intensive parts of an application
- Adopt scalable algorithms
- Optimize data arrangements to maximize locality
- Performance Tuning
- Pay attention to code portability and maintainability





```
Given: input_image, kernel( Keranl is Another name for filter )
ouput_height = input_height - kernel_height + 1
output_width = input_width - kernel_width + 1
output_image = np.zeros((output_height, output_width))
for i in range(0, output_height):
  for j in range(0, output_width):
    for ii in range(0, kernel_height):
      for jj in range(0, kernel_width):
        output_image[i,j] += input_image[i+ii,j+jj] * kernel[ii,jj]
```

This is a psudoCode for the filter slides from previous images above !

Principles of Parallel Computing

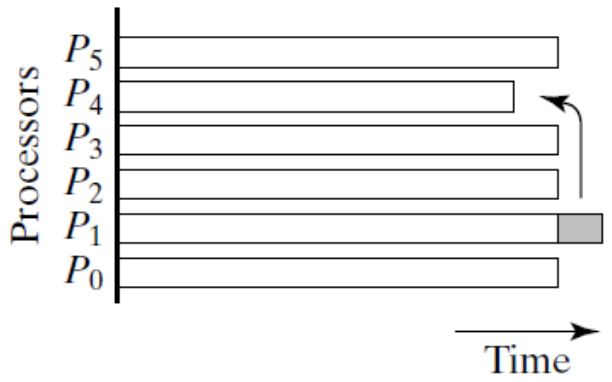
- Finding enough parallelism (Amdahl's Law)
- Granularity
- Locality
- Load balance
- Coordination and synchronization
- Performance modelling
- **All of these make parallel programming even harder than sequential programming.**

Granularity

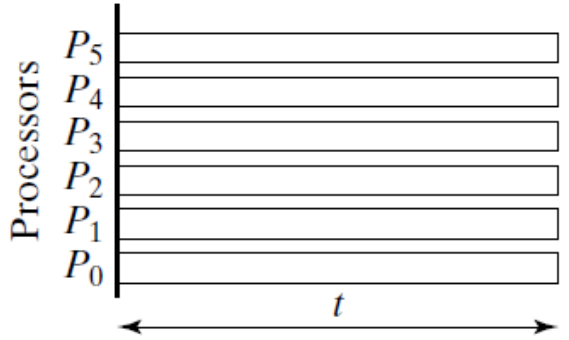
- Parallelism is not free. Overhead includes:
  - Cost of starting a thread or process
  - Cost of communicating shared data
  - Cost of synchronizing
  - Extra (redundant) computation
  - Each of these can be in the range of milliseconds (=millions of flops) on some systems
- **Tradeoff: Algorithm needs sufficiently large units of work to run fast in parallel (I.e. large granularity), but not so large that there is not enough parallel work**

Load Balance/ imbalance

- Load imbalance is the time that some processors in the system are idle due to:
  - insufficient parallelism (during that phase)
  - unequal size tasks
- Algorithm needs to balance load



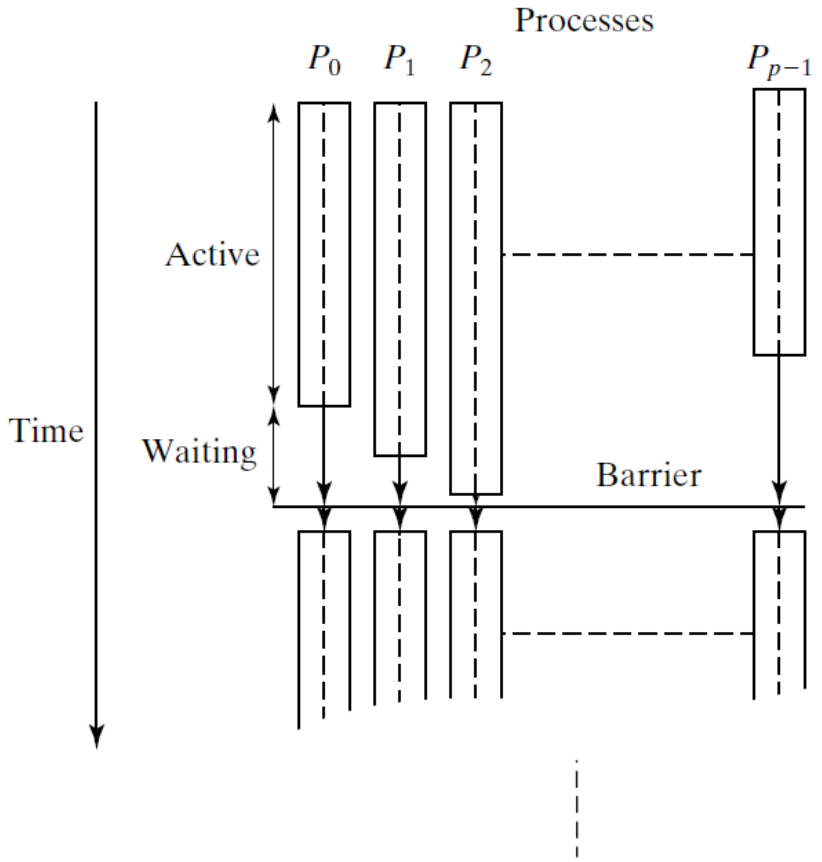
(a) Imperfect load balancing leading to increased execution time



(b) Perfect load balancing

Synchronization

- Need to manage the sequence of work and the tasks performing
- Often requires "serialization" of segments of the program
- Various types of synchronization maybe involved
  - Locks/Semaphores
  - Barrier
  - Synchronous Communication Operations



Performance Modeling

- Analyzing and tuning parallel program performance is more challenging than for serial programs.
- There is a need for parallel program performance analysis and tuning.

Think, understand then code

- Think about the problem you are trying to solve
- Understand the structure of the problem
- Apply mathematical techniques to find solution
- Map the problem to an algorithmic approach
- Plan the structure of computation
  - Be aware of in/dependence, interactions, bottlenecks
- Plan the organization of data
  - Be explicitly aware of locality, and minimize global data
- Finally, write some code! (this is the easy part)

LAU
الجامعة اللبنانية الأميركية
Lebanese American University

LAU
School of Engineering

Thank You