Multicore Programming (CS 535) Project Presentation

Students: Mustafa Ghanim and Cengiz Emre Dedeağaç

Instructor: Asst. Prof. İsmail Aktürk

Project Title:

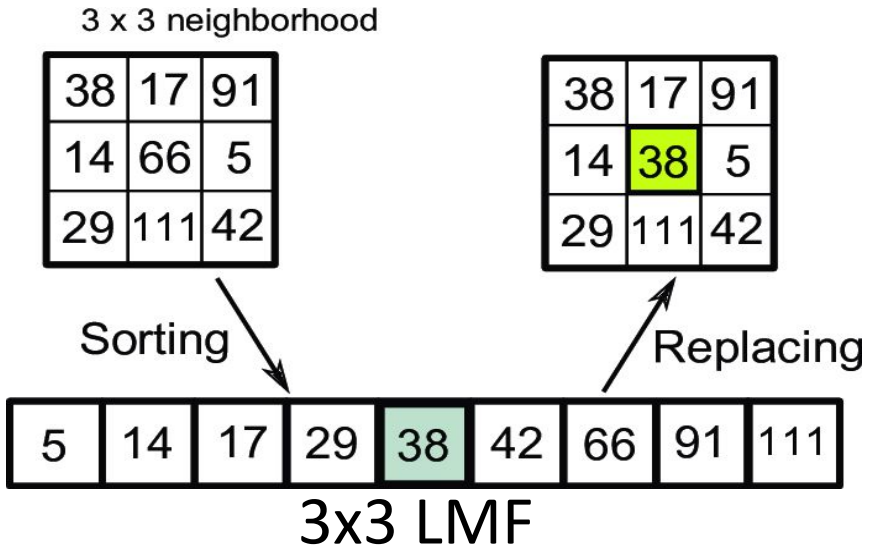
“A Fast Implementation of Adaptive Median Filter Using Multithreading”

Project Motivation

- 2D Median filter is widely used in image processing applications
- Adaptive median filter (AMF) enhances the noisy image quality better than the local median filter (LMF)
- AMF is more complex and does not have a direct implementation in OpenCV library
- Fast operations on video (image) frames are critical for real-time applications

LMF vs AMF

- Both of them are useful to remove impulsive noise (salt and pepper) from images



source:Marturi, N. (2013). *Vision and visual servoing for nanomanipulation and nanocharacterization in scanning electron microscope* (Doctoral dissertation, Université de Franche-Comté).

AMF Algorithm:

1. Initialize a filter kernel *W* of *N*x*N* size (*N* = 3)
2. Get *MIN*, *MAX*, CENTER, and *MED* of *W*
3. If (*MED* > *MIN* **AND** *MED* < *MAX*):
   - If (*CENTER* > MIN **AND** *CENTER* < *MAX*):
     - **No change**
   - Else:
     - ***CENTER*** ← ***MED***
   - Else:
     - ***N*** ← ***N + 2***
     - If (*N* <= N_{MAX}): (N_{MAX} = 15)
       - **Go to Step 2**
     - Else:
       - ***CENTER*** ← ***MED***

Images Dataset

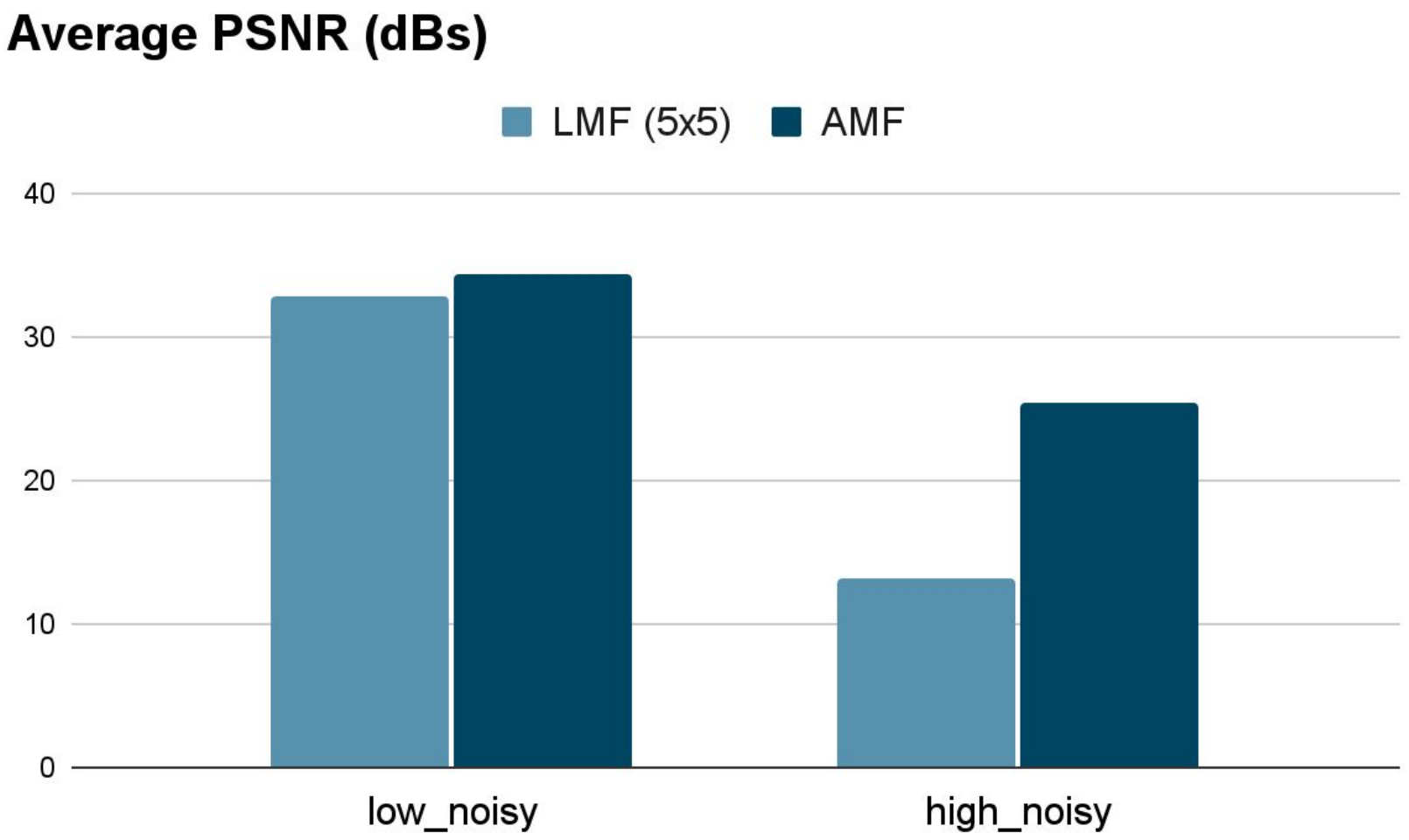
Image Name	Resolution	PSNR (dBs)
low_noisy_0801	1356x2040	12.57
low_noisy_0802	1356x2040	12.15
low_noisy_0803	1536x2040	12.93
low_noisy_0804	1200x2040	12.52
low_noisy_0805	1536x2040	12.41
high_noisy_0801	1356x2040	7.09
high_noisy_0802	1356x2040	6.28
high_noisy_0803	1536x2040	7.41
high_noisy_0804	1200x2040	7.19
high_noisy_0805	1536x2040	7.12

$$PSNR = 10\ log_{10}(\frac{MAX(I)^2}{MSE})\ dBs$$

$$MSE = \frac{1}{M.N} \sum_{n=0}^{n=N-1} \sum_{m=0}^{m=M-1} [I_{src}(n,m) - I_{dst}(n,m)]^2$$

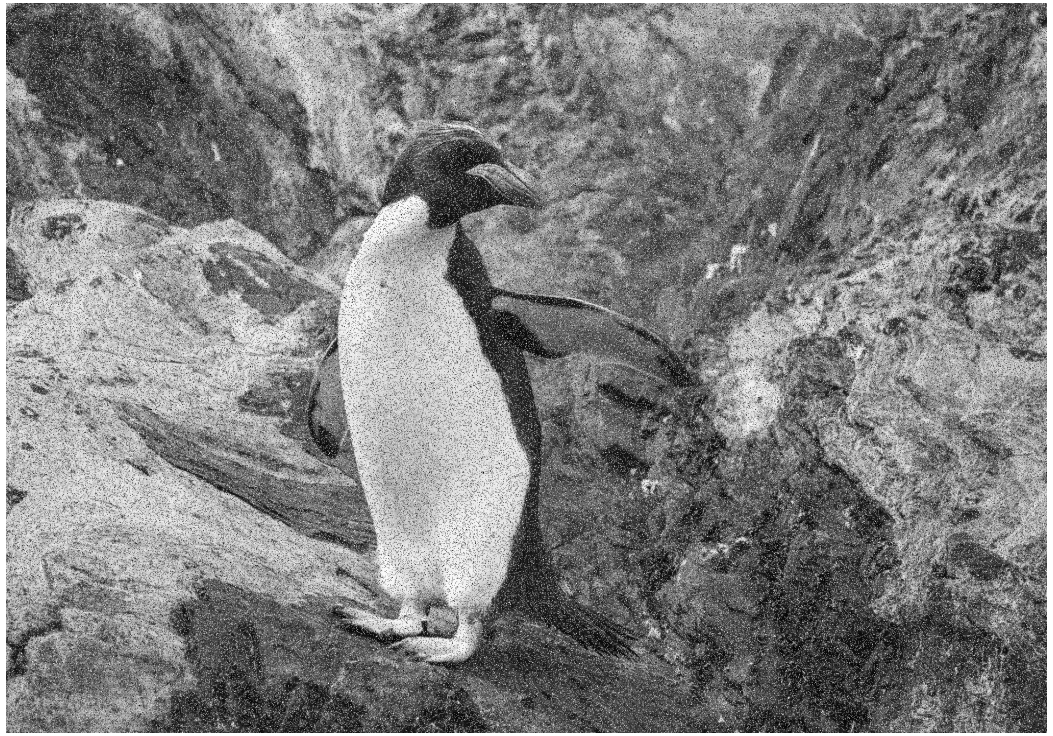
- *M*: total number of image rows
- *N*: total number of image columns
- *MAX (I)* = 255 for 8-bit images
- Quality categories according to PSNR values (dBs):
  <10 ⟶ Very Bad
  10-20 ⟶ Bad
  20 -25 ⟶ Poor
  25-31 ⟶ Fair
  31-37 ⟶ Good
  > 37 ⟶ Excellent

LMF vs AMF (Quality Performance)



LMF vs AMF (Quality Performance)

(**Original RGB Image**)



(**Low Noisy Grayscale Image)**

LMF vs AMF (Quality Performance)

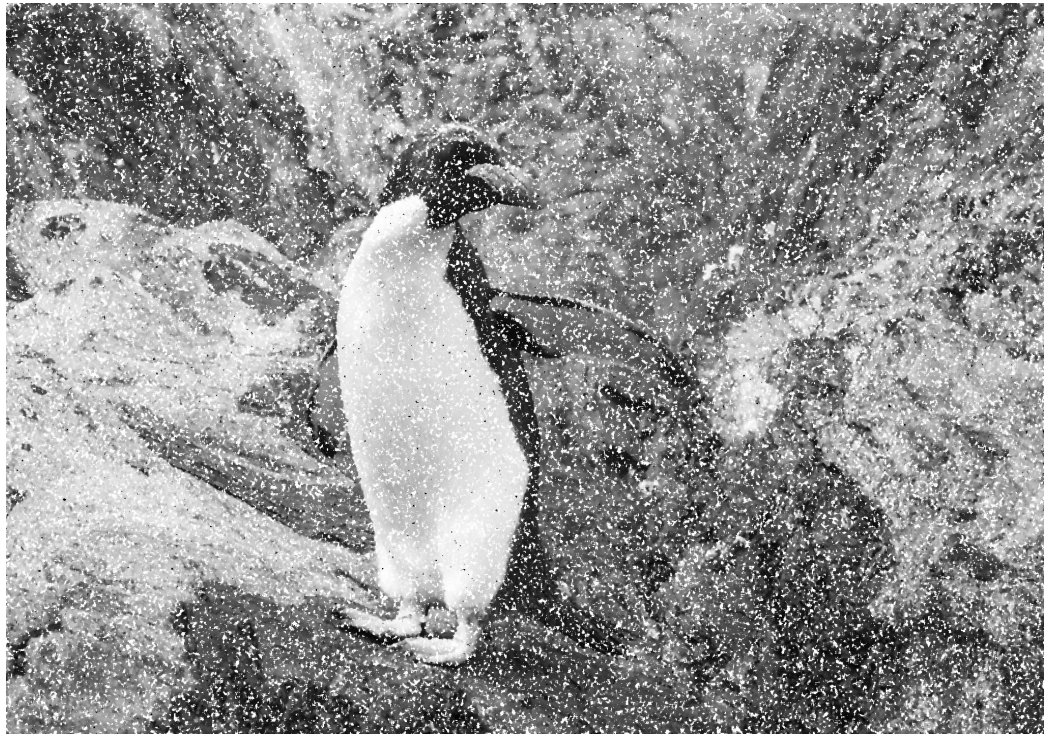
(Low Noisy Image After 5x5 LMF)

(Low Noisy Image After AMF)

LMF vs AMF (Quality Performance)

(High Noisy Image)

LMF vs AMF (Quality Performance)



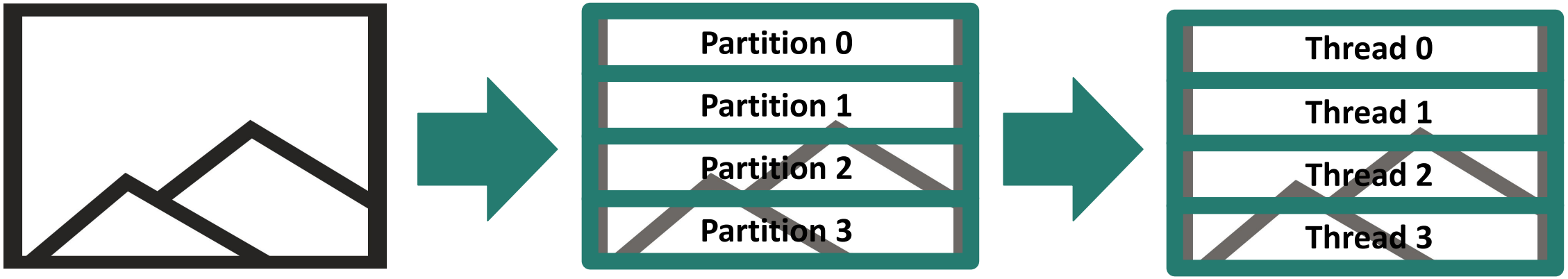
(High Noisy Image After 5x5 LMF)

(High Noisy Image After AMF)

Basic Parallelization

- Divide the image into partitions
- Create a thread for each partition
- Wait for all threads
- Exit

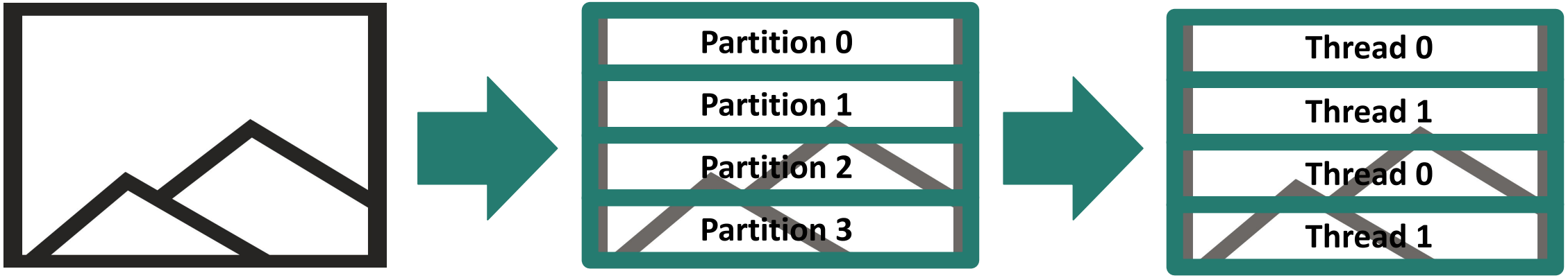
Basic Parallelization



Scheduling

- Divide the image into partitions
- Create a number of threads
- Threads will go through partitions
- Exit when the assigned partitions are complete

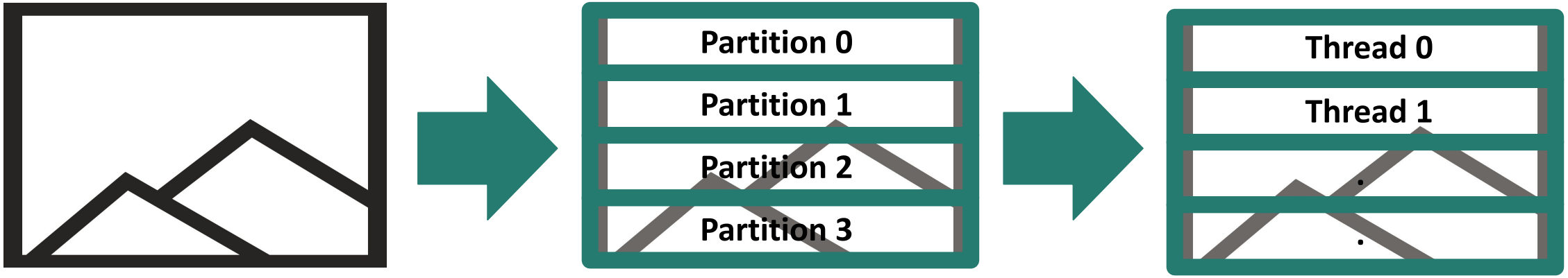
Scheduling


Partition 0
Partition 1
Partition 2
Partition 3
Thread 0
Thread 1
Thread 0
Thread 1

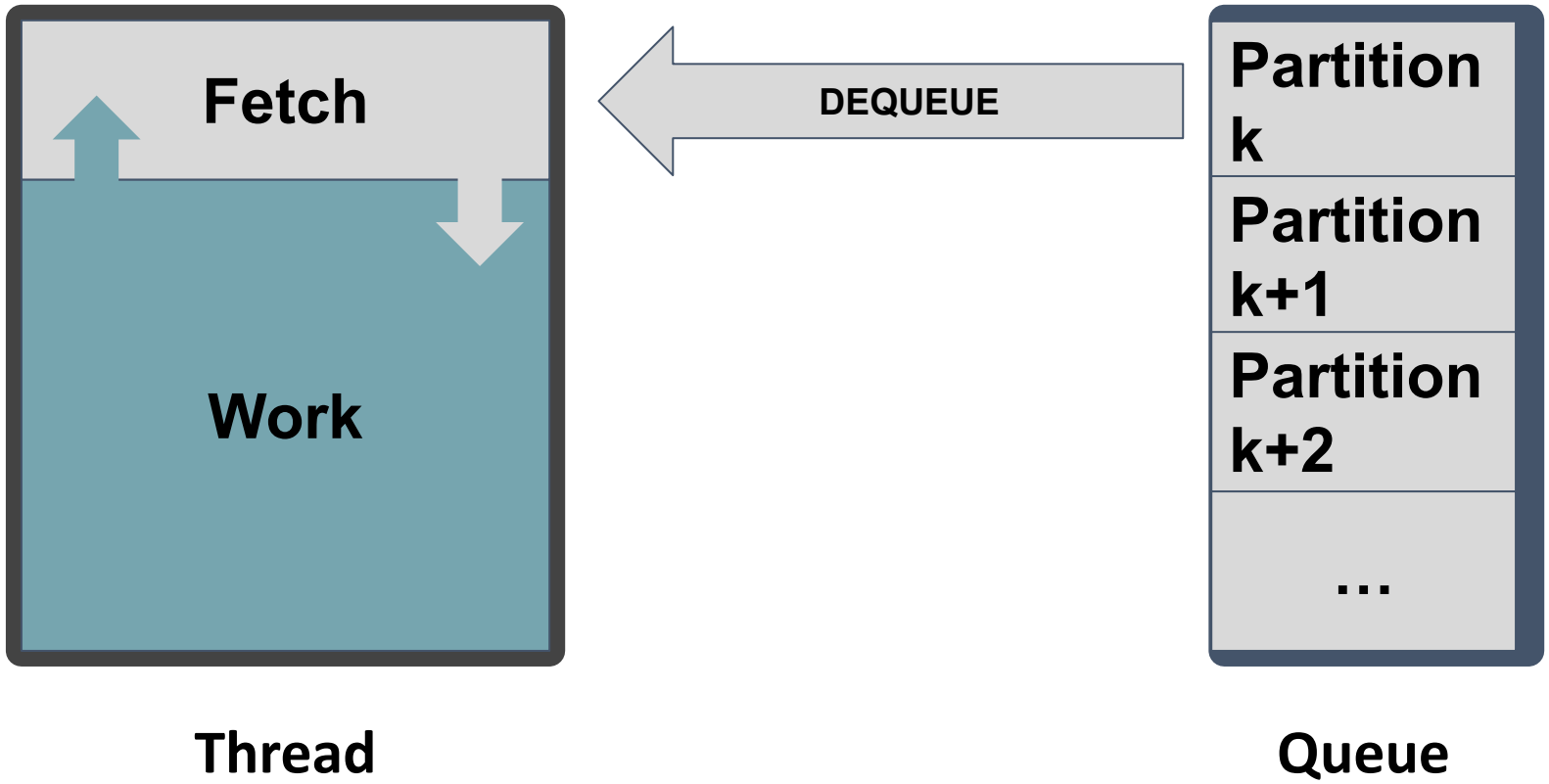
Dynamic Scheduling

- Threads will get initial partitions
- Thread will finish its job
- Thread will get the next ready partition
- Exit when there are no more partitions

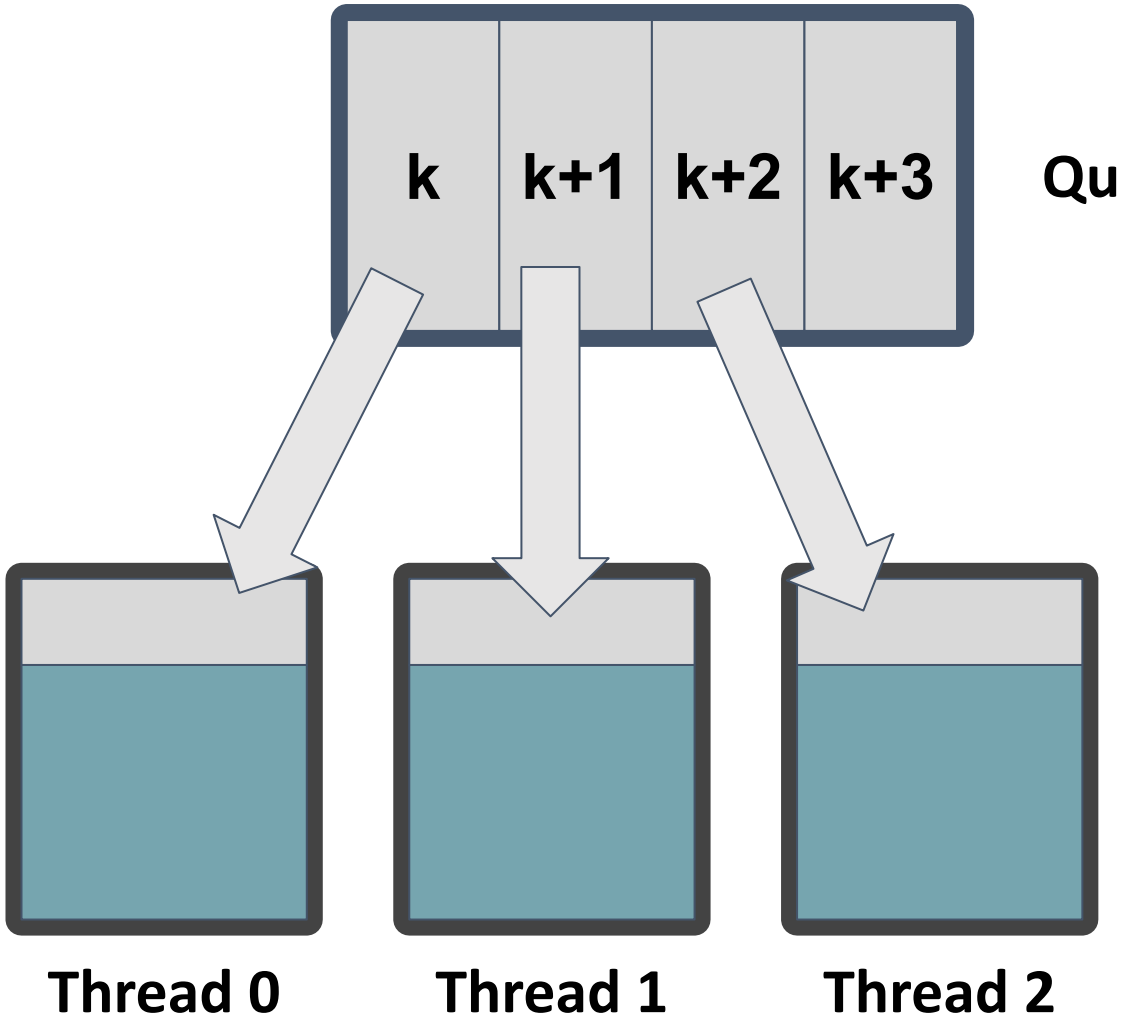
Dynamic Scheduling



Dynamic Scheduling

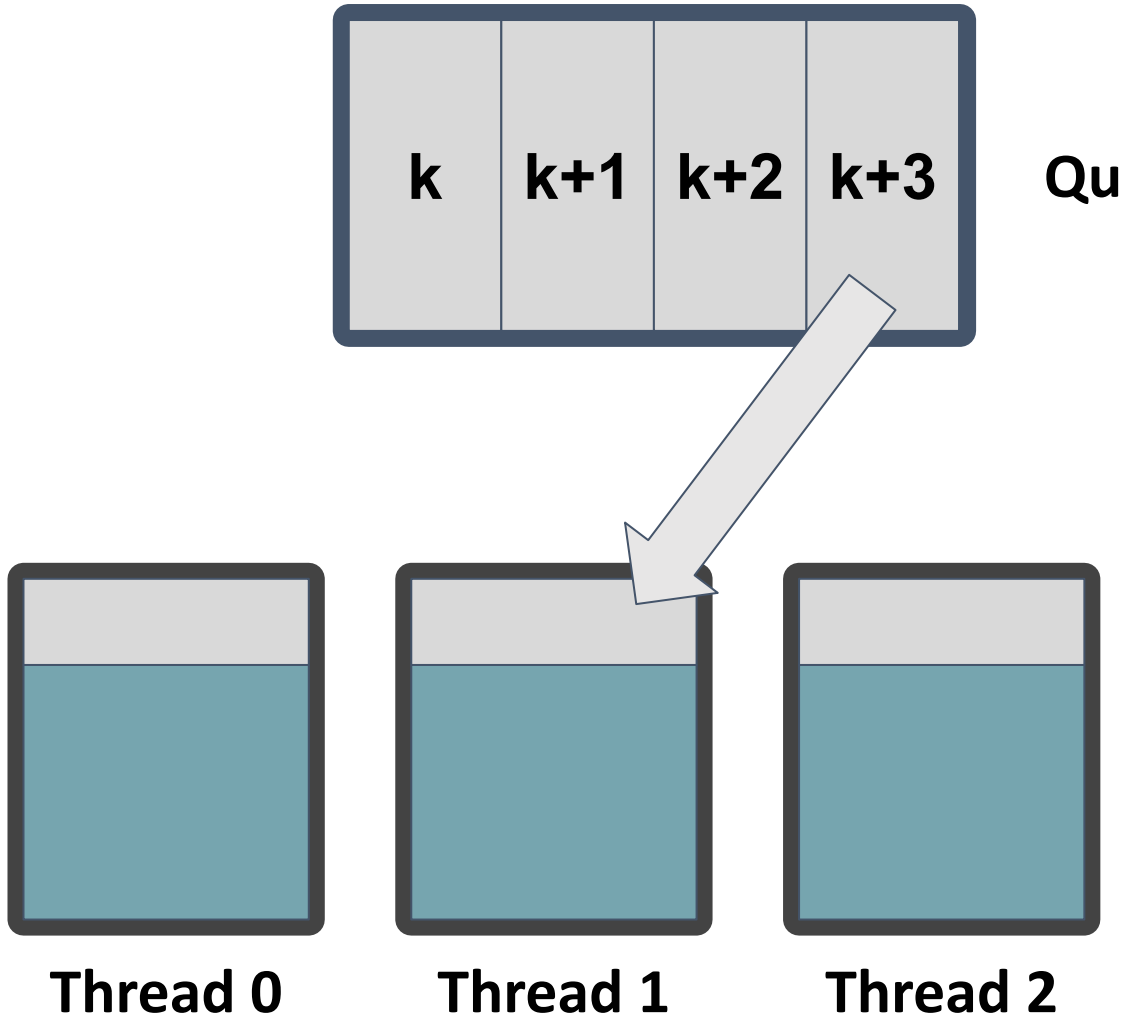


Dynamic Scheduling



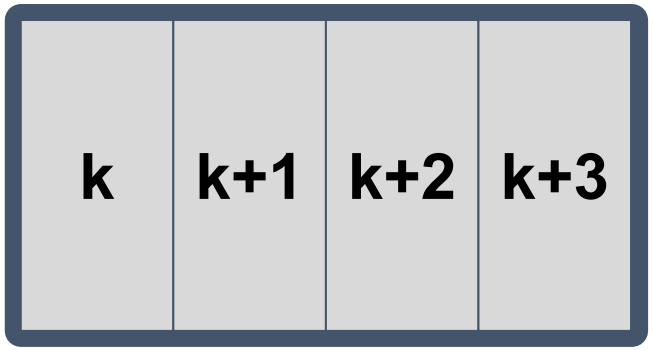
Thread 0 fetches partition k.
Thread 1 fetches partition k+1.
Thread 2 fetches partition k+2.

Dynamic Scheduling



Thread 0 fetches partition k.
Thread 1 fetches partition k+1.
Thread 2 fetches partition k+2.
Thread 1 finishes.
Thread 1 fetches partition k+3.

Dynamic Scheduling

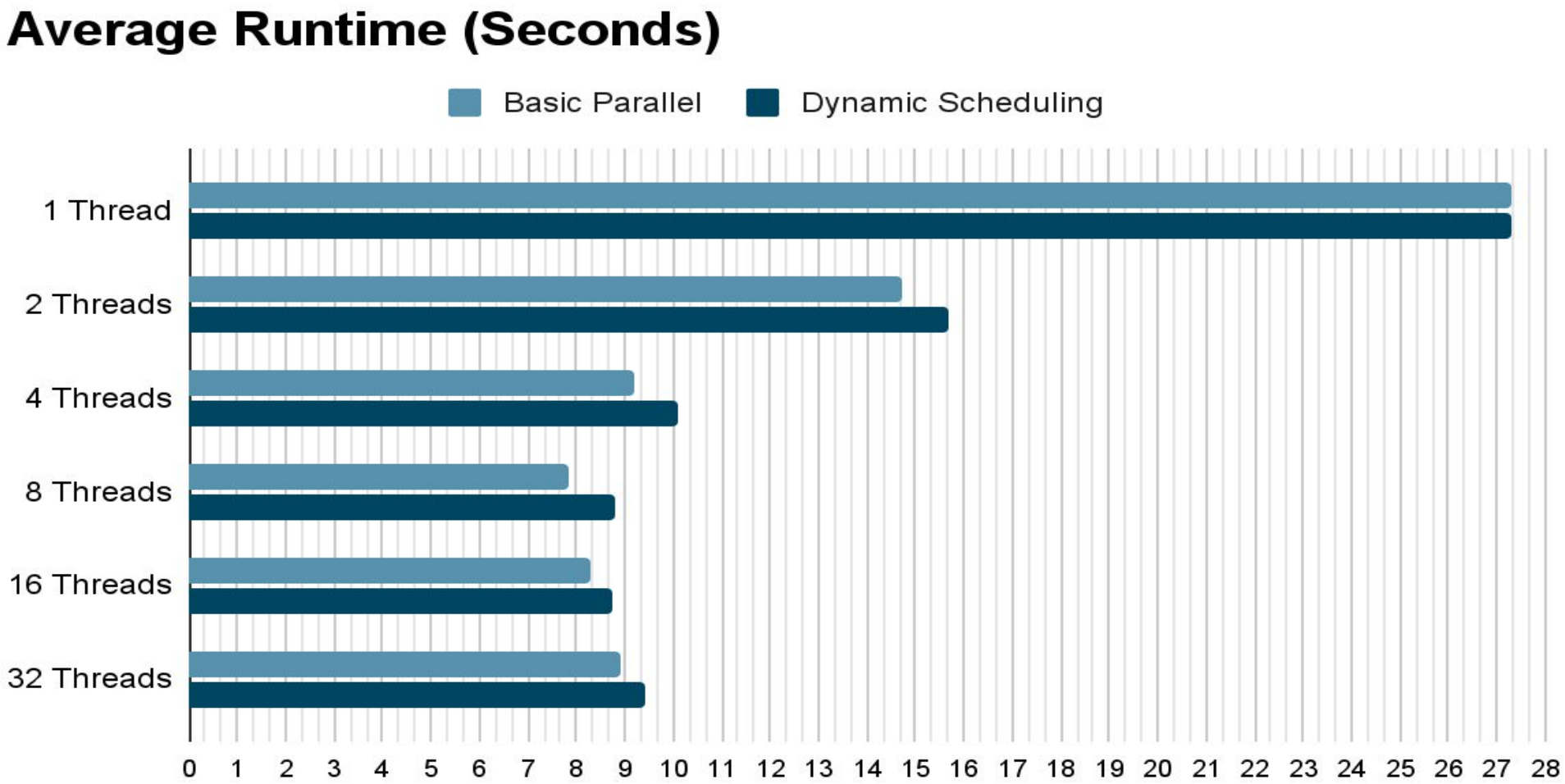


Queue

Thread 0 Thread 1 Thread 2

Thread 0: fetches partition k.
Thread 1: fetches partition k+1.
Thread 2: fetches partition k+2.
Thread 1: finishes.
Thread 1: fetches partition k+3.
Thread 0: finishes.
Thread 0: queue is empty.
Thread 0: returns.

Runtime Results



Memory Related Results

16 Threads	Basic Parallelization	Dynamic Scheduling	Difference
Memory Consumption	71.3 MB	71.0 MB	360.1 KB
Number of Allocations	1,506,613	1,500,705	5,908

32 Threads	Basic Parallelization	Dynamic Scheduling	Difference
Memory Consumption	78.5 MB	78.1 MB	440 KB
Number of Allocations	1,619,690	1,612,705	6,985

64 Threads	Basic Parallelization	Dynamic Scheduling	Difference
Memory Consumption	93.3 MB	92.7 MB	541.0 KB
Number of Allocations	1,849,557	1,841,650	7,907

Conclusions

- The average runtime of a 5x5 LMF with OpenCV (uses multithreading) is < 30 ms
- Applying LMF sequentially and with higher window sizes can also be tested
- The average runtime of a sequential AMF (3x3 $\rightarrow$ 15x15) $\approx$ 27 seconds
- On a 4-core hardware with 8 threads: an approximate 74% reduction of runtime is achieved using multithreading