Lecture 7

Concurrency and Synchronization

Institute of Computer Science & Information Technology,
Faculty of Management & Computer Sciences,
The University of Agriculture, Peshawar, Pakistan.

Concurrency and Synchronization

- Concurrency is the tendency for things to happen at the same time in any system.

- Concurrency is a natural phenomena, of course.

- In the real world, at any given time, many things are happening simultaneously.

Concurrency (1)

- When dealing with currency issues in software systems, there are generally two aspects that are important:

1. Being able to detect and respond to external events occurring in a random order, and

2. Ensuring that these events are responded to in some minimum required interval.

Challenges in Concurrency

- The challenges of designing concurrent systems arise mostly because of the interaction which happen between concurrent activities.

- In concurrent activities, some sort of coordination is required.

- This coordination also known as synchronization.

Basic synchronization primitives

- Mutual exclusion: Locks, Mutex, Semaphores, ..
- Conditions: Flags, Condition Variable, Signal,

Mutual Exclusion

- Mutual exclusion is a property of concurrency control, which is introduced for the purpose of preventing race conditions.

- It is the requirement that one thread of execution never enters a critical section while a concurrent thread of execution is already accessing critical section.

Mutual Exclusion(1)

- Race condition is an undesirable event that can happen when multiple entities access or modify share resources in a system.
- The system behaves correctly when these entities use the shared resources as expected.
- When race condition happens, the system may enter a stated not designed for and hence fail.

Mutual Exclusion (2)

- A lock or Mutual Exclusion(Mutex) is a synchronization primitive: is a mechanism that enforces limits on accessing a resource when there are many threads of execution.

- The first person to propose such primitive was Edsger Dijkstra, who suggested a new data type called a semaphore.

Semaphore

- A semaphore is an integer variable that supports a set operation.
- A semaphore is an integer variable used to control access to a common resource by multiple threads and avoid critical section problems.
- A trivial semaphore is a variable that is changed (e.g., incremented or decremented, or toggled) depending on conditions.

Semaphore (I)

- More specifically, if a S is a variable of type semaphore, then two atomic operations are supported as S: P(S) and V(S)

- The letters P and V come from the Dutch words passeren, to pass (allow a resource to thread), and Vrygeven, to release (released a resource by).

Semaphore (2)

- The operation P(S) achieves the following in an atomic manner:

If (S>0)

Decrement S;

Else

Wait for S to become positive

Semaphore (3)

- The operation V(S) is defined as follows:

```
If (threads waiting for S)
   Assign one of them;
Else
   Increment S;
```

Semaphore (4)

- Using semaphores, we can now easily program mutual exclusion to critical sections as follows:

Thread 1

```
----
P(S);
// Enter Critical Section
…..
// Leave Critical Section
V(S);
```

Thread 2

```
----
P(S);
// Enter Critical Section
…..
// Leave Critical Section
V(S);
```

- …

Data and Work Partitioning

Decomposition (Partitioning)

- One of the fundamental steps that we need to undertake to solve a problem is to split the computations to be performed into a set of tasks for concurrent execution defined by the task-dependency graph.

- Decomposition techniques are broadly classified as:

1. Recursive Decomposition
2. Data Decomposition
3. Exploratory Decomposition
4. Speculative Decomposition

Recursive Decomposition

- The recursive and data decomposition techniques are relatively general purpose as they can be used to decompose a wide variety of problems.

- On the other hand, speculative and exploratory decomposition techniques are more of a special purpose nature because they apply to specific classes of problems.

Recursive Decomposition

- Recursive decomposition is a method for inducing concurrency in problems that can be solved using the divide-and-conquer strategy.
- In this technique, a problem is solved by first dividing it into a set of independent sub-problems.
- Each one of these sub-problems is solved by recursively applying a similar division into sub-problems followed by a combination of their results.
- The divide-and-conquer strategy results in natural concurrency, as different sub-problems can be solved concurrently.

Quick Sort

- Consider the sorting a sequence A of n elements using quick sort algorithm.

- Quicksort is a divide and conquer algorithm that starts by selecting a pivot element x and then partitions the sequence A into two subsequences A_0 and A_1 such that all the elements in A_0 are smaller than x and all the elements in A_1 are greater than or equal to x.

Quick Sort(1)

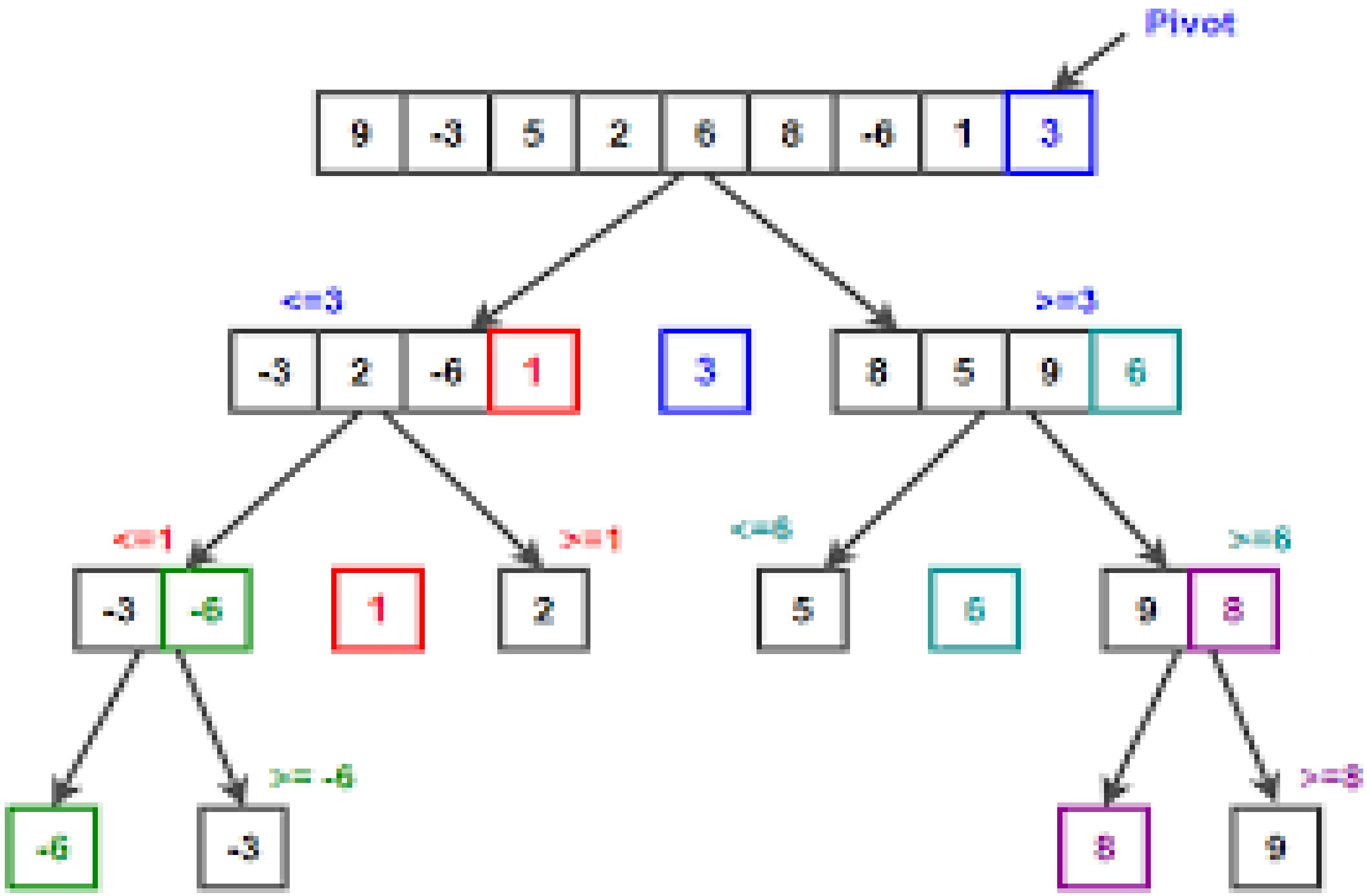
- This partitioning step forms the divide step of the algorithm.

- Each one of the subsequences A_0 and A_1 is sorted by recursively calling quicksort.

- Each one of these recursive calls further partitions the sequences.

- This is illustrated in following Figure for a sequence of 12 numbers.

QuickSort(2)



QuickSort(3)

- The recursion terminates when each subsequence contains only a single element.
- Then the results will be combined to form a sorted list.

Data Decomposition

- Data decomposition is a powerful and commonly used method for deriving concurrency in algorithms that operate on large data structures.

- In this method, the decomposition of computations is done in two steps:

1. In the first step, the data on which the computations are performed is partitioned, and

2. In the second step, this data partitioning is used to induce a partitioning of the computations into tasks.

Data Decomposition (1)

- The operations that these tasks perform on different data partitions are usually similar.
- The partitioning of data can be performed in many possible ways.
- But we are going to discuss matrix-multiplication.

Matrix Multiplication

- To multiply a matrix by another matrix we need to do "dot product" of rows and columns.. What does that mean?

- Let see with example: 1st Row x 1st Column:
- (1,2,3)*(7,9,11) = 1x7+2x9+3x11=58
- (1,2,3)*(8,10,12) = 1x8+2x10+3x12=64

Exploratory Decomposition

- Exploratory decomposition is used to decompose problems whose underlying computations correspond to a searching of a solution from search space.

- In exploratory decomposition, we partition the search space into smaller parts, and search each one of these parts concurrently, until the desired solution are found.

Speculative Decomposition

- Speculative Decomposition is used when a program may take one of many possible computationally significant branches depending on the output of other computations.

- In this situation, while one task is performing the computation whose output is used in deciding the next computation, other task can concurrently start the computations.

Speculative Decomposition(1)

- This scenario is similar to evaluating one or more of the branches of a switch statement in C in parallel before the input for the switch is available.