Assignment 6

DAA

Q1) write note on dynamic multithreading

→ Dynamic multithreading (also known as thread-level speculations) is an advanced threading technique used in parallel computing to improve the execution of multithreaded programs.

It focuses on increasing the efficiency of parallel execution by dynamically adjusting the number of threads and their work distributed based on the workload and system conditions.

key concepts and characteristics -

1) Adaptive Thread creation - Dynamic multithreading allows the runtime system to create and manage threads on-the-fly

2) workload balancing:- It involves intelligent load balancing Threads can be dynamically reassigned to different tasks or data regions ensuring.

3) Fine-Grained parallelism:- Dynamic can exploit fine grained parallelism in a program by creating many light weight threads.

4) Task based parallelism:- Instead of using a fixed number of threads, dynamic multithreading is open based on a ~~thread~~ task-based model.

5) Scalability:- Dynamic multithreading allows programs to scale better with the available hardware resources.

6) Complexity:- Implementing dynamic multithreading can be complex due to the need for dynamic thread creation.

Q2) Write and explain multithreaded merge sort algorithm

Sol:-

Multithreaded merge sort is an efficient parallel version of the classic merge sort algorithm, designed to take advantage of multiple threads or processors to improve the sorting performance.

Algorithm:-

1) Divide the problem - the input array to be sorted is divided into smaller subarrays.

2) Parallel sorting:- Each subarray is sorted independently by a seperate thread

3) Wait for threads - After launching threads to sort subarrays, the main thread waits for all subtasks to complete

4) Merge subarrays:- in a single sorted array.

5) Final result

Pseudo code :-

```
function multi_threaded_merge_sort(arr, num_threads):
    if num_threads == 1 or len(arr) <= 1:
        return merge_sort(arr)
    else:
        mid = len(arr) // 2
        left_thread = thread(multi_threaded_merge_sort,
            arr[mid], num_threads // 2)
        right_thread = thread(multi_threaded_merge_sort,
            arr[mid], num_threads // 2)
        left_thread.join()
        right_thread.join()
        return merge(left_thread.result, right_thread.result)
```

Q3) what is distributed system? Explain distributed breadth first algorithm with example.

Sol:-

Distribute system is a collection of interconnected independent computers that work together to provide a unified computing capacity.

In a distributed system, these computers communicate and coordinates their actions by passing messages to achieve a common goal to perform a specific task.

Characteristics:-

1) Concurrency:- Distributed systems often involve multiple processes running concurrently on different nodes.

2) Scalability:- Distributed systems can be designed to scale horizontally

3) Reliability and fault tolerance:- incorporate redundancy and error handling to ensure that the system can continue

4) Transparency:- Distributed system aim to hide the complexities of their distributed nature from end users and application developers

Distributed breadth first search Algorithm is an essential graph traversal algorithm adapted for distributed systems

BFS is used to explore and discover nodes and their relationships in a graph or network

Q1) What are distributed algorithms?

Sol:-

Distributed algorithms are a category of algorithms designed to solve problems and perform computations in a distributed systems, where multiple interconnected, autonomous computers or nodes work together to achieve a common goal.

Some of the distributed algorithms include:-

1) Distributed consensus algorithms:- These ensure that all nodes in a distributed system agree on a single value or decision, even in the presence of failures. Examples include the Paxos and Raft algorithms.

2) Distributed data storage and retrieval algorithms:-
These enable data to be stored and accessed across multiple nodes, such as in distributed databases and content delivery networks.

3) Distributed resource allocation algorithms:-
These manage the allocation of resources like CPU, memory or network bandwidth in a distributed system.

4) Distributed load balancing algorithms:-
These distribute incoming requests or tasks evenly across multiple servers to optimize resource usage and improve system performance.

Distributed algorithms play a crucial role in modern computing, as they underpin the operation of various large scale ~~operat~~ applications

Q5) write naive string matching algorithm. what is the time complexity of algorithm? Explain complexity of algorithm with example.

Sol:-

The naive string matching algorithm, also known as the brute force or simple pattern. It works by systematically comparing the pattern to all possible substrings of the text and checking for matches.

Algorithm:-

1) start at beginning of the text
2) For each character position in the text:-
   a) Compare the characters ~~of~~ of the text and the pattern.
   b) Mismatch is formed, move one position to the right in the text. and reset the pattern.
   c) the correct substring of the text.
3) Repeat this process until the ~~end~~ of the text.

Time complexity

$O(n-m+1)*m)$

n:- length of text.

m:- length of the pattern

eg:- Text - 'ABABABA ABABABA'
Pattern - 'ABABA'

1) We start comparing the pattern ABABA with the text from the first position
2) We compare pattern with next substring
3) This process continues till the end of text is reached

There are 5 occurances of the pattern within the text. The total number of comparisons is determined by the number of positions where the pattern can start, which is $(n-m+1) = 15-5+1 = 11$

Therefore the time complexity is $(n-m+1)m) = O(11*5) = O(55)$ which is a linear function of the length of the text and length of the pattern.