PARALLEL ALGORITHMS ASSIGNMENT-II

GROUP 2

SAI TEJASWEE REDDY PASHAM

SOWJANYA POLEPALLY

MANOJ KUMAR RAVILLA

Q1. Submit Cheat sheet generated through the In-Class Exercises. Identify the Best Indexing Techniques in Parallel Manner and Best Searching Techniques in Sequential and Parallel Manner.

**BEST INDEXING TECHNIQUES IN PARALLEL MANNER**:

**1.SEPARATE CHAINING OR OPEN HASHING**

Worst case: O(N)

Average case: O(NH)

Where N is number of Items, H is the size of hash table

**2.OPEN ADDRESSING OR CLOSED HASHING**

**LINEAR PROBING**

**Serial Approach Parallel Approach**

**Look up** Best Case: O(n) Best Case: O(1)

Worst Case: O(n2) Worst Case: O(n)

**Insert** Best Case: O(n) Best Case: O(1)

Worst Case: O(n2) Worst Case: O(n)

**Removal** Best Case: O(n) Best Case: O(1)

Worst Case: O(n2) Worst Case: O(n)

**QUADRATIC PROBING**

Worst case: O(nlogn) or O(logn)

**DOUBLE HASHING**

Uses two hash functions and access becomes inefficient at higher load factor.

**3.CUCKOO HASHING**

Hashing with worst-case O(1) lookups

Deletions take time O(1)

Worst case look-up is O(log n)

ADVANCED INDEXING TECHNIQUES

1. **BLOOM FILTER**

Time Complexity for Insertion and Search is O(k)

Space Complexity is O(m) {M= bits, k hash function}

1. **INVERTED INDEX**

Inverted index is a data structure storing a mapping form content. Inverted file index, list of references to documents for all words. Full inverted index, positions of each word is also referred in documents.

1. **BIG TABLE**

Maps row key, column key and timestamp. Here tables are split into tablets through GFS.

The best indexing technique in parallel manner is **Quadratic probing** because, Quadratic probing can be a more efficient algorithm in a closed hash table, since it better avoids the clustering problem that can occur with linear probing.  It also provides good memory caching because it preserves some locality of reference. Faster Insertion when compared to other techniques. Insertion never fails in this technique.

**BEST SEARCHING TECHNIQUES IN SEQUENTIAL PARALLEL MANNER**

**RANGE SEARCHING**

The Space and Time complexity is O(n(logn)d-1) ford-dimensional range trees.

Single-Shot:

Space S(n) and Time T(n)

Repetitive-Mode:

Preprocessing time: P(n)

Space occupied by Data Structures: S(n)

Query Time: Q(n)

Dynamic Case: Update Time U(n)

**SEARCH IN MAP REDUCE**

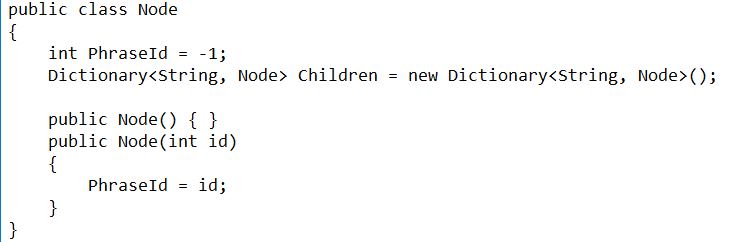
Uses “map”, “shuffle” and “reduce” functions. Scalability, flexible and fastness is achieved. It is simple and implements parallel programming.

(2). Design a parallel indexing and searching technique and implement it in spark. Use the technique within a scenario such Organization (UMKC) search. For example, building search, people search or department search. Consider different aspects or attributes for indexing such as person name, phone number, department name etc.

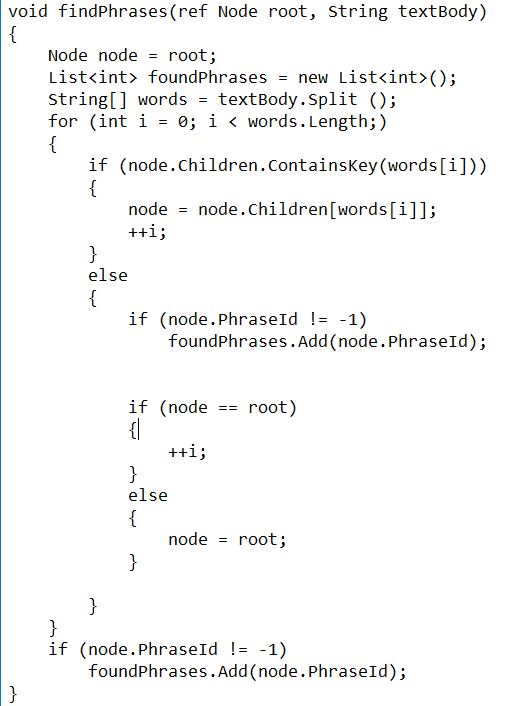
Hint: Multidimensional indexing can be considered

(A).

Indexing:

****

Searching:



Time complexity: O(nlogn)