PARALLEL ALGORITHMS

TEAM 3

SNEHA DIDIGAM

SIRISHA MANTRAMURTHY

DEEPTHI PENMETSA

**Single-level Ordered Indexes**

**Primary Indexes**:

* notation used for an index entry i in the index file:

<K(i), P(i)>

K(i) -primary key value, P(i) –block pointer address

* Time complexity: O(n)

**Clustering Indexes**

* Time complexity: O(n)

**Secondary Indexes**

* Time complexity: O(n)

**Multi-Level Indexes**

**Search Tree**

* Time complexity: O(log n)

**Point Access Methods**

**B Tree**

**Average case Worst case**

* **Space complexity** O(n) O(n)
* **Searching** O(logn) O(logn)
* **Insertion** O(logn) O(logn)
* **Deletion** O(logn) O(logn)
* Keys are kept in sorted order for sequential traversing
* As it uses hierarchical indexing, reduces the number of disk reads
* In order to speed insertions and deletions, uses partially full blocks
* Balances index with an elegant recursive algorithm

**B+ Tree**

**Average case Worst case**

* **Space complexity** O(n) O(n)
* **Searching** O(logn) O(logn)
* **Insertion** O(logn) O(logn)
* **Deletion** O(logn) O(logn)
* Leaf nodes have pointers to the subsequent leaves, results in faster access.

**Multidimensional Hashing:**

**Grid File**

* Time complexity: O(n)

**Hierarchical methods:**

**KD tree**

* is a data structure for [space-partitioning](https://en.wikipedia.org/wiki/Space_partitioning) & organizing [points](https://en.wikipedia.org/wiki/Point_(geometry)) in a *k*-dimensional [space](https://en.wikipedia.org/wiki/Euclidean_space)

**Average case Worst case**

* **Space complexity** O(n) O(n)
* **Searching** O(logn) O(logn)
* **Insertion** O(logn) O(logn)
* **Deletion** O(logn) O(logn)

**R tree**

* R-Trees can organize any n-dimensional data by representing it in a minimum bounding region.
* try to extend/merge B-trees and k-d trees
* Search O(log n)
* Insertion O(n)
* Deletion O(n)

**Space Filling Curves:**

**Z-ordering**

* Given a set of point, use a B+-tree to index the z-values
* Time Complexity: O(n)

**Hilbert Curve**

* use the Hilbert values to insert objects into the tree
* Time Complexity: O(n)

Single level ordered indexes is best suited for sequential manner. All point Access methods, multidimensional hashing and hierarchical methods suits best for parallel manner.

In all the parallel indexing techniques ‘HILBERT CURVE’ is the best technique for the following reasons:

* The Hilbert curve is easy to generate.
* Without making any changes to the underlying databases, Hilbert curves can be used to manage multidimensional data efficiently.
* The jumps in Z-ordering technique are avoided in Hilbert curve method.
* Hilbert curves are used in Image Processing
* They are used to implement task scheduling in parallel processing computing applications.

In all the sequential indexing techniques ‘CLUSTERED INDEXING’ is the best technique for the following reasons:

* Has better performance for query results.
* Possible to retrieve or fetch data quickly.
* Can be used for sorting. We can eliminate post-fetch-sort operation.
* Unique indexes guarantee uniquely identifiable records in the database

**2. Inverted Indices Mapreduce:**

**Algorithm:**

**class Mapper**

**procedure Map(documentid n, document d)**

**A ← new Array**

**for all terms w ∈ document d do**

**A{w} ← A{w} + 1**

**for all terms t ∈ A do**

**Emit(term w, posting An, A{w}i)**

**class Reducer**

**procedure Reduce(term w, postings [An1, p1i,An2, p2i. . .])**

**N ← new List**

**for all postings Aa, pi ∈ postings [An1, p1i,An2, p2i. . .] do**

**Append(N,Aa, pi)**

**Sort(N)**

**Emit(term w, postings N)**

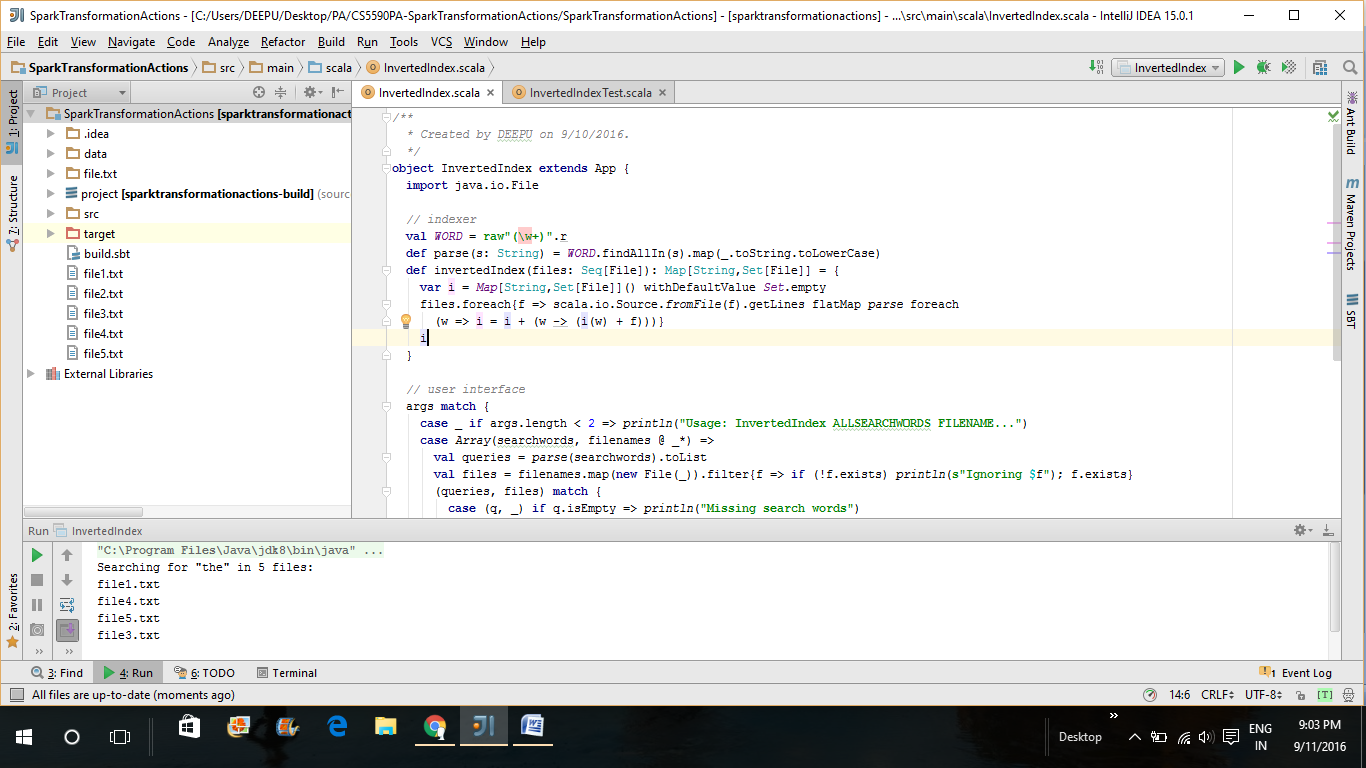
**CODE:**

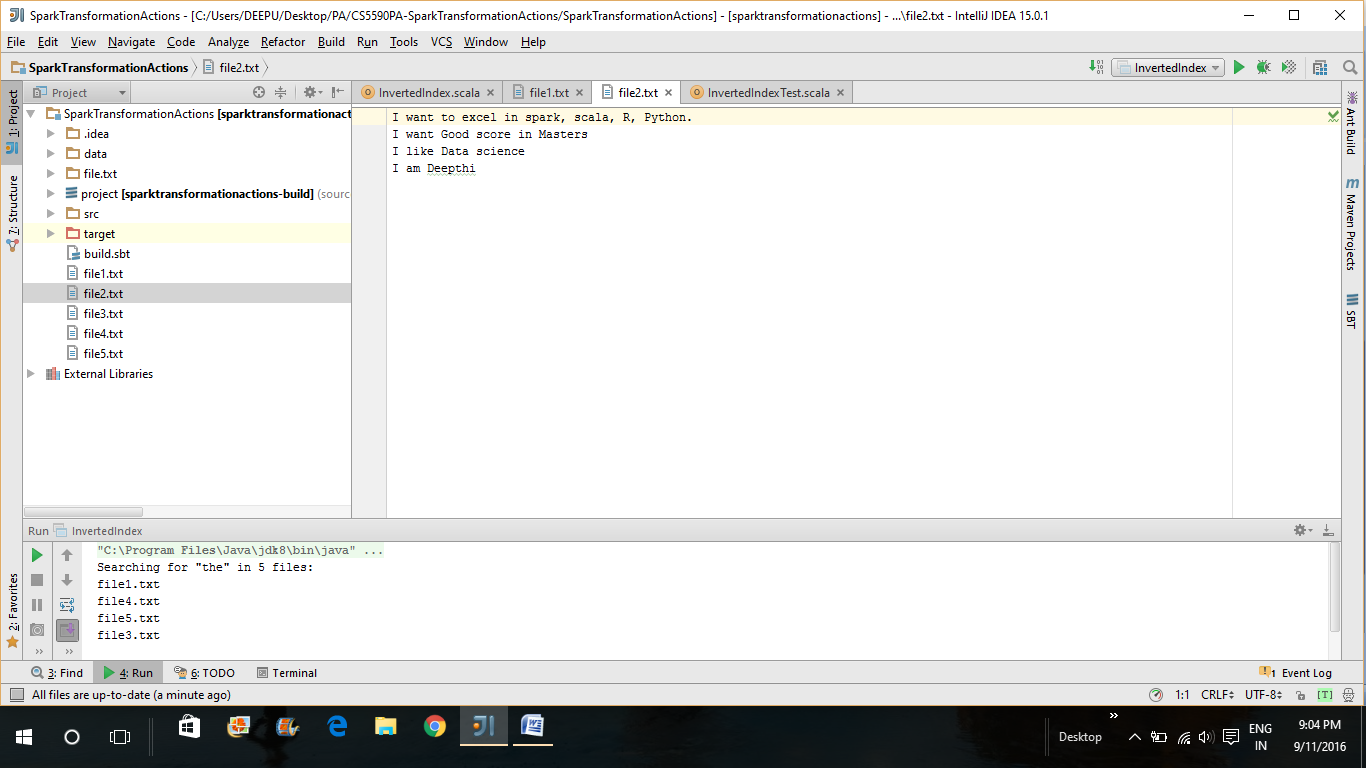
github link: [www.github.com/Deepu123start/ParallelAlgorithms](http://www.github.com/Deepu123start/ParallelAlgorithms)

Input: The input to the mapreduce code is the text files containing data.

Output: The output obtained is the list of references of the documents for the searched word. In other words, it gives the list of documents that contains the word we search for.

In the screenshots below, we searched for the word “the”, so it gives the list of files in which “the” is present. ( all files except in file2.txt- “the” is not present in file2.txt)





**Time complexity and performance:**

Inverted index has O(logN) Time complexity, Where n is the number of Documents in the index.

Space complexity is O(N).