A group of cubes connected together

Description automatically generated

Task:03

Domain: Algorithm and Data-structure

Mehjabin Mostafa | Real-Time Processing with Divide and Conquer Algorithms in Big Data Applications

**Divide and Conquer:**

The idea of divide and conquer is to break the problem in to many subproblem until the sub problems are smaller enough to solve directly. It uses recursion to divide the problems again and again. After finding the answers to all the subproblems, they are combined or merge to produce the overall solution.

Dividing into sub problems and solving them at the same time allows parallel processing which is more efficient.

Examples:

* Merge Sort
* Quick Sort
* Karatsuba Algorithm for fast multiplication
* Strassen’s matrix multiplication
* Convex hull
* Quick hull

**Divide and conquer in Big Data:**

* Scalability
* Benefit of parallel processing
* Improved efficiency
* Fault tolerance
* Optimization of resources
* Application in big data algorithm(MapReduce)

Example:

1.merger sort with big data:

*#include <iostream>*

*#include <vector>*

*using namespace std;*

*void merge(vector<int>& data, int left, int mid, int right) {*

*int n1 = mid - left + 1; // Size of left subarray*

*int n2 = right - mid; // Size of right subarray*

*vector<int> leftArray(n1);*

*vector<int> rightArray(n2);*

*for (int i = 0; i < n1; i++)*

*leftArray[i] = data[left + i];*

*for (int j = 0; j < n2; j++)*

*rightArray[j] = data[mid + 1 + j];*

*int i = 0; // Initial index of left subarray*

*int j = 0; // Initial index of right subarray*

*int k = left; // Initial index of the merged array*

*while (i < n1 && j < n2) {*

*if (leftArray[i] <= rightArray[j]) {*

*data[k] = leftArray[i];*

*i++;*

*} else {*

*data[k] = rightArray[j];*

*j++;*

*}*

*k++;*

*}*

*while (i < n1) {*

*data[k] = leftArray[i];*

*i++;*

*k++;*

*}*

*while (j < n2) {*

*data[k] = rightArray[j];*

*j++;*

*k++;*

*}*

*}*

*void mergeSort(vector<int>& data, int left, int right) {*

*if (left >= right) {*

*return; // Base case: a single element or empty subarray*

*}*

*int mid = left + (right - left) / 2; // Find the midpoint*

*// Recursively sort each half*

*mergeSort(data, left, mid);*

*mergeSort(data, mid + 1, right);*

*merge(data, left, mid, right);*

*}*

*int main() {*

*// Example dataset*

*vector<int> data = {38, 27, 43, 3, 9, 82, 10};*

*cout << "Original Data: ";*

*for (int num : data)*

*cout << num << " ";*

*cout << endl;*

*mergeSort(data, 0, data.size() - 1);*

*cout << "Sorted Data: ";*

*for (int num : data)*

*cout << num << " ";*

*cout << endl;*

*return 0;}*

example: MapReduce (pseudo code):

*INPUT: List of text lines (text\_data)*

*OUTPUT: Word counts (global\_word\_count)*

*START*

*1. Load input text data (text\_data).*

*2. Divide text\_data into CHUNKS based on the number of threads (num\_threads):*

*- chunk\_size = ceil(size of text\_data / num\_threads)*

*- For each thread i:*

*Assign chunk[i] = text\_data[i \* chunk\_size to (i+1) \* chunk\_size]*

*3. Initialize local\_word\_counts[num\_threads] as an empty list of maps.*

*- Each map will store key-value pairs for words and their counts.*

*4. Mapper Phase:*

*For each thread i in [0, num\_threads):*

*Run MAPPER function on chunk[i] in parallel:*

*a. For each line in chunk[i]:*

*Split the line into words.*

*For each word:*

*Add word to local\_word\_counts[i] with count = count + 1.*

*5. Initialize global\_word\_count as an empty map.*

*6. Reducer Phase:*

*For each thread i in [0, num\_threads):*

*Run REDUCER function in parallel:*

*a. Lock access to global\_word\_count.*

*b. For each word in local\_word\_counts[i]:*

*Add word count to global\_word\_count.*

*c. Release lock.*

*7. Output Results:*

*For each word in global\_word\_count:*

*Print (word, count).*

*END*

***Mapper function:***

*MAPPER(chunk, local\_word\_count):*

*For each line in chunk:*

*Split the line into words.*

*For each word:*

*Clean word (remove punctuation, convert to lowercase).*

*Increment local\_word\_count[word].*

*Return local\_word\_count.*

***Reducer Function:***

*REDUCER(local\_word\_count, global\_word\_count):*

*Lock global\_word\_count for thread-safe access.*

*For each word in local\_word\_count:*

*Increment global\_word\_count[word] by local\_word\_count[word].*

*Unlock global\_word\_count.*

***Sample Input:***

*text\_data = [*

*"Hello world! This is a simple example of MapReduce.",*

*"MapReduce is designed for processing massive data.",*

*"Divide and conquer is the core of MapReduce.",*

*"Hello again, MapReduce!"*

*]*

***Output:***

*Word Counts:*

*again: 1*

*and: 1*

*conquer: 1*

*core: 1*

*data: 1*

*designed: 1*

*divide: 1*

*example: 1*

*hello: 2*

*is: 2*

*mapreduce: 4*

*massive: 1*

*of: 3*

*processing: 1*

*simple: 1*

*the: 1*

*this: 1*

*world: 1*

**Decision Tree:**

It’s a Machine learning algorithm that is used for classification and regression.it shows the decisions and their possible consequences in a tree-like or flowchart-like structure. Decision Tree can also be used in Data Mining, Statistics etc.

**Root node:**

Top node of the tree which represents the basic if whole data set and it splits based on information gain and impurity.

There are many more things in the decision tree. Internal node, branches and leaf node (final outcome of the DT) which holds a important part in a Decision Tree.

A diagram of a tree

Description automatically generated Fig: Structure of DT

**Advantages of DT:**

* Simplicity and Interpretability
* Versatility
* No Need for Feature Scaling
* Handles Non-linear Relationships

**Disadvantages of Decision Trees**

* Overfitting
* Instability
* Bias towards Features with More Levels

**Pruning**

To overcome overfitting, pruning techniques are used. Pruning reduces the size of the tree by removing nodes that provide little power in classifying instances. There are two main types of pruning:

* Pre-pruning (Early Stopping): Stops the tree from growing once it meets certain criteria
* Post-pruning: Removes branches from a fully grown tree that do not provide significant power.

**Applications of Decision Trees in Big Data**

**Fraud Detection**:

Detect anomalies or suspicious activities in massive transactional datasets. Example: Credit card companies identifying fraudulent transactions.

**Healthcare Analytics**:

Diagnose diseases or predict patient outcomes using medical data. Example: Decision support systems for personalized treatment recommendations.

**Financial Forecasting**:

Predict market trends or risks in large financial datasets. Example: Stock price prediction or credit risk assessment.

**Sentiment Analysis**:

Analyze customer reviews or social media data to classify sentiment. Example: Sentiment classification for brand monitoring.

**Manufacturing Optimization**:

Analyze production data to optimize processes or predict machine failures. Example: Predictive maintenance in industrial IoT.

**Environmental Monitoring**:

Classify environmental data for climate change research or disaster prediction. Example: Predicting forest fires or floods using sensor data.

**Educational Analytics**:

Predict student performance and suggest interventions using academic data. Example: Identifying at-risk students in massive online courses.

**Large Scale Numerical Calculation:**