Parallel Array Sorting

Agenda

What is Parallel Sort in Java?

What is Parallel Sort in Java?

- Parallel Sort is a new feature added in Java 8
- Java 8 introduced a new method parallelSort() in the Arrays class of java.util package
- The Parallel Sort is using multiple threads to perform sorting
- The Fork/Join common thread pool is used to execute any parallel tasks
- The Fork/Join framework is introduced in Java 7
- The algorithm used in Parallel Sorting is **"Parallel Merge Sort"**

Parallel Merge Sort - Algorithm

Parallel Merge Sort - Algorithm

1. Divide the array into two parts

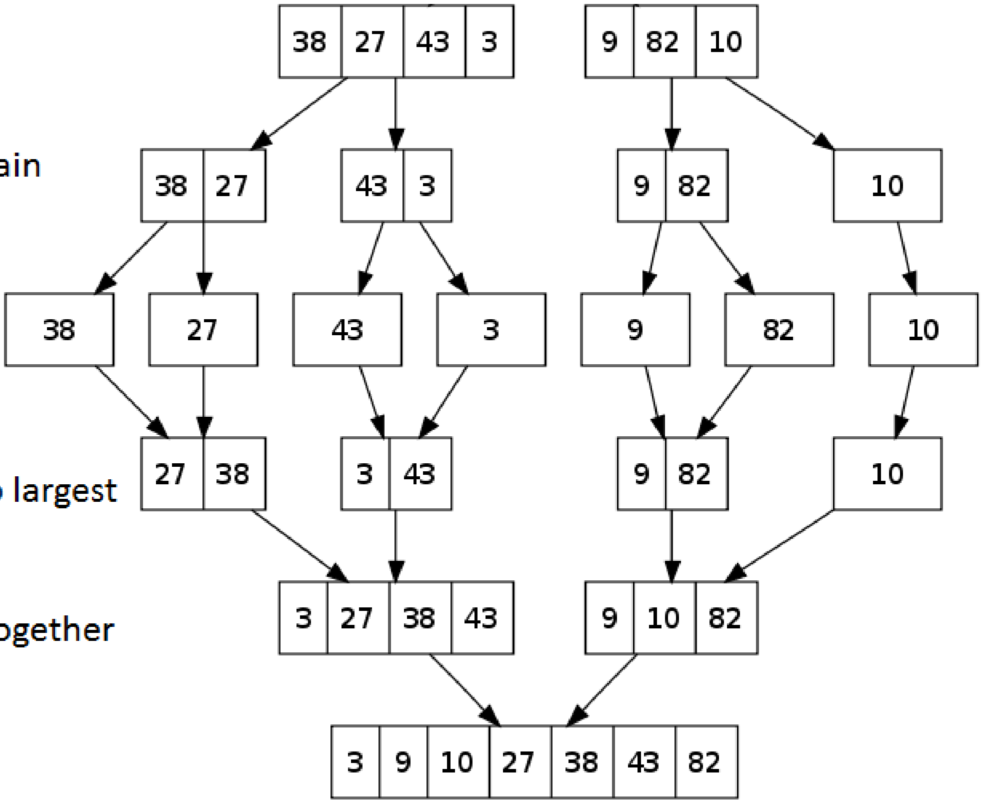
2. Divide the array into two parts again

3. Break each element into single parts

4. Sort the elements from smallest to largest

5. Merge the divided sorted arrays together

6. The array has been sorted



Parallel Array Sorting - Example

Parallel Array Sorting - Example

```
import java.util.Arrays;

public class ParallelArraySortingDemo {

public static void main(String[] args) {
    int arr[] = new int[] {5,2,8,1,9,3};
    System.out.println("Before Sorting");
    for(int i : arr)
        System.out.print(i+" ");

    Arrays.parallelSort(arr);

    System.out.println("After Sorting");
    for(int i : arr)
        System.out.print(i+" ");
    }
}
```

Output:

```
Before Sorting
5 2 8 1 9 3

 After Sorting
1 2 3 5 8 9
```

Serial vs Parallel Sort in Java

Serial vs Parallel Sort in Java

- **Arrays.sort( )** : is a Serial / Sequential Sorting.
- Serial / Sequential Sorting uses single thread to perform sorting.
- It takes bit longer time to perform sorting.

- **Arrays.parallelSort( )** : is a Parallel Sorting.
- Parallel Sorting uses multiple threads to perform sorting.
- It is faster when there is more elements in the array whereas slower for less elements.

Serial vs Parallel Sort - Example

Serial vs Parallel Sort - Example

```java
import java.util.Arrays;   import java.util.Random;
public class SerialVsParallelSortDemo {

public static void main(String[] args) {
int[] arraySizes = {10000, 100000, 1000000, 10000000};

        for(int arraySize : arraySizes ) {

            System.out.println("When Array size is : "+arraySize);

            int[] arr = new int[arraySize];
            Random random = new Random();

            for(int i=0; i < arraySize; i++)
                arr[i] = random.nextInt(arraySize);

            int[] sequentialArr = Arrays.copyOf(arr, arr.length);
            int[] parallelArr = Arrays.copyOf(arr, arr.length);


            long startTime = System.currentTimeMillis();
            Arrays.sort(sequentialArr);
            long endTime = System.currentTimeMillis();

            System.out.println("Time Taken for Serial Sort in Milli seconds : " + (endTime - startTime));

            startTime = System.currentTimeMillis();
            Arrays.parallelSort(parallelArr);
            endTime = System.currentTimeMillis();

            System.out.println("Time Taken for Parallel Sort in Milli seconds : " + (endTime - startTime));
            System.out.println("---------------------------------------------------------");
          }
      }
   }
```

Output:

```
When Array size is : 10000
Time Taken for Serial Sort in Milli seconds : 3
Time Taken for Parallel Sort in Milli seconds : 6
---------------------------------------------------------
When Array size is : 100000
Time Taken for Serial Sort in Milli seconds : 14
Time Taken for Parallel Sort in Milli seconds : 22
---------------------------------------------------------
When Array size is : 1000000
Time Taken for Serial Sort in Milli seconds : 134
Time Taken for Parallel Sort in Milli seconds : 45
---------------------------------------------------------
When Array size is : 10000000
Time Taken for Serial Sort in Milli seconds : 1030
Time Taken for Parallel Sort in Milli seconds : 468
---------------------------------------------------------
```

Parallel Array Range Sort

Parallel Array Range Sort

- Using Arrays.parallelSort( ) method we can sort the elements in the specified range of an Array
- Syntax : **Arrays.parallelSort (inputArray, fromIndex, toIndex);**
- If fromIndex and toIndex is same then we will get empty results
- Here fromIndex is inclusive and toIndex is exclusive
- The range should be in between zero and length of array

Parallel Array Range Sort - Example

```
import java.util.Arrays;

public class ParallelArrayRangeSortingDemo {

public static void main(String[] args) {

    int arr[] = new int[] {5,2,8,1,9,3};
    System.out.println("Before Sorting");
    for(int i : arr)
        System.out.print(i+" ");

    Arrays.parallelSort(arr,1,4);

    System.out.println("After Sorting");
    for(int i : arr)
        System.out.print(i+" ");
    }
}
```

Output:

```
Before Sorting
5 2 8 1 9 3

After Sorting
5 1 2 8 9 3
```

Thank you