Team members:

Kalyana Venkata Ramana Gelli (16140775)

Samyuktha Anumolu (12381492)

1. Design:

1.1 Objective: To design 3 scheduling algorithms.

1.2 Coding Language: Java

1.3 Tool (IDE): Eclipse

1.4 Deployment & Testing platform: Hadoop in IBM Cloud

1.5 Steps followed:

1. Create a project and import hadoop-0.20.2.tar.gz which is downloaded from <ftp://apache.mirrors.pair.com/hadoop/common/hadoop-0.20.2/>. as an Archive file.

2. Created 4 new class files:

Class file for Common Methods: This contains 4 methods each containing logic for Fifo, Scheduling Algorithm 1,2,3 respectively which internally calls methods from bottom to top (If the order mentioned earlier is considered top to bottom) in case of ties.

Logic included main parameters that are extracted from each JobInProgress: numMapTasks, numReduceTasks, finishedMaps(),pendingMaps(),pendingReduces()

All the classes that contained scheduling algorithm logic implemented java.util.Comparator interface with Generic type set to JobInProgress. The main logic is in overridden compare method that compared two jobs (say j1, j2).

Class file for Scheduling Algorithm 1: If total tasks of j1 is less than total tasks of j2 then job j1 is executed first. Else if total tasks of j1 is greater than total tasks of j2 then job j2 is executed first. Else tie is broken by calling Fifo code in Common methods.

Class file for Scheduling Algorithm 2: If percentage map tasks of j1 is less than percentage map tasks of j2 then job j1 is executed first. Else if percentage map tasks of j1 is less than percentage map tasks of j2 then job j2 is executed first. Else tie is broken by calling Scheduling Algorithm 1 code in Common methods.

Class file for Scheduling Algorithm 3: If total map and reduce tasks of j1 that are pending is higher than total map and reduce tasks of j2 that are pending then job j1 is executed first. Else if total map and reduce tasks of j1 that are pending is less than total map and reduce tasks of j2 that are pending then job j2 is executed first. Else tie is broken by calling Scheduling Algorithm 2 code in Common methods.

1. Test Cases:

Test cases are designed and run for all the 3 algorithms and all test results are captured. Job tracker logs are verified.

a) Priorities of the jobs are set by passing a priority array in the script for the modified word count java.

b) Number of reduces are set by passing them in a reduce array in the script for modified word count jar.

c) Ensured that number of Maps value is varied for different jobs (sinces number of Maps can’t be passed at run time) by using input files of different sizes (i.e; where 2 input files has a size difference greater than hadoop block size of 64 MB).

d) Jobs are submitted at different time stamp, using the script by looping job submission where each job goes in to then background (using &) once submitted and then next job is submitted and so on.

e) Logic of tie breaking for job priorities is verified by submitting 2 jobs that have same priority, number of maps, number of reduces but with different start time with in a job set.

f) Logic of tie breaking of Number of tasks is verified by submitted 2 jobs with same number of maps and reduces with different priorities or different start time with in a job set.

g) Scheduling algorithms are verified by submitting jobs later in time when few jobs are already in progress.

1. Challenges Faced:
   1. Building files of fairscheduler.jar – **Solution:** Got it done using simple ANT build.
   2. Overriding default map, reduce tasks and job priority while submitting jobs.

**Solution:** Set mapred.map.tasks, mapred.reduce.tasks, mapred.job.priority parameters of javax.security.auth.login. Configuration class.

* 1. Creating automation scripts for moving files into remote IBM cloud instance and vice versa.
  2. Creation of bash script to automate the invocation of jobs with different priorities maps and reduces.

1. Contribution

|  |  |  |
| --- | --- | --- |
| Name | Task Responsibility (Percentage Contribution) | Contribution Ratio |
| Samyuktha Anumolu | Coding(100%), Building(100%), Deployment(100%), Testing(30%), Report(70%) | 1:2 |
| Kalyana Venkata Ramana Gelli | Deployment(100%), Testing(70%), Report(30%) | 1:2 |