# 2018 CS313 - Assignment 3: IPL Computation with Map-Reduce

|  |  |  |
| --- | --- | --- |
| Member Name | USN | Section |
| Rhythm Girdhar | 01FB16ECS302 | Dr SG |
| Saloni Govil | 01FB16ECS333 | Dr KVS |
| Sanjay K A | 01FB16ECS341 | Dr KVS |
| Shashank M Bongale | 01FB16ECS355 | Dr KVS |

**Date of Evaluation**: 22/09/2018  **Evaluator**: Avinash

**Algorithm to generate batsman vulnerability with Map Reduce (What are intermediate keys)**

A csv file is fed into the Map Reduce function as input. The mapper function forms (key, value) pairs for every occurrence of the Combination (Batsman\_Bowler) with value initialized to 1 indicating that a wicket was taken. Since the mapper only takes into consideration the occurrences of a particular key, the csv file fed into the Map Reduce function only contains those Batsman\_Bowler combinations for which wicket was taken. The Reducer on the other hand, adds all the corresponding values for a given key and gives the total number of wickets taken by a particular bowler against a particular batsman (for all possible combinations).

Example for intermediate key value pair:

(Batsman\_Bowler , [1,1,1,1,1])

**Algorithm to generate sorting based on batsman vulnerability**

Once we obtain the Batsman\_Bowler combination as the key and the total number of wickets for that combination (for all possible combinations), this result is passed into another map reduce function by interchanging the key and value (i.e. Total number of wickets becomes Key and Batsman\_Bowler combination becomes value). Now after performing the Mapper function we obtain the total number of wickets as the key and a list of all the Batsman\_Bowler combinations with that key as the value. Now, the keys are sorted in descending order in the Map function and after the reduce function, all the combinations are printed individually with the help of the command IterableWrite.

**Timing Measurements** (#mappers = 1 )

|  |  |  |
| --- | --- | --- |
| Experiment | Without Map-reduce | With Map – reduce |
| Calculation of vulnerability and sorting based on values | Real : 3.591 ms  User: 3.284 ms  Sys : 0.032 ms | Real : 9.28 ms  User: 18.286 ms  Sys : 0.523 ms |

**Explanation for difference in performance**:

Based on the observations made above, we observe that the above problem statement takes more time in executing without the Map Reduce function. This is observed because the csv files that we’re dealing with are very small and using Map Reduce adds a lot of computation overhead. Thus the cost of computation is much higher than the cost of computation.

**Any other observations/challenges/comments:**

By default sorting in map-reduce program is done is ascending order. In order to sort the values in descending order we are using an inbuilt comparator function and interchanging key-value pairs.