Parallel processing of UB Classroom Scheduling data using Hadoop MapReduce

Compiled by
Amit Kamat

Question 1: UB Course Demand

Every semester UB offers great courses to its students and provide the best world class faculty to teach those courses. However, we have observed that some of the courses are always in high demand either due to increasing technology demand of that field or due to extra ordinary teaching faculty.
UB student affairs want to globalize some of its courses which are highly demanding among students but we are not able to shortlist the courses. Can you utilize the "UB Course Scheduling data" to gather an insight on which courses have always been the first choice for the students?

Solution:

We are using number of students enrolled and class capacity to get an insight on the course demand**.** Since the problem is to find the most demanding courses we tackle it using 2 step MR Job. In the 1st step we calculating the total enrollment and capacity of the class for each course over the years and than in the 2nd step averaging over the number of semesters that course have been offered.

In some of the courses the enrolled student's count is greater than the class capacity which shows that those courses are very high in demand. However, only this factor was not sufficient to derive the results so we are also calculating the %age of vacant seats to figure out that on an average how many seats remained vacant and this information will provide us course demand. We have categorized the course demand into 5 different categories High Demand (< 10%), Above Average Demand (10-25 %), Average Demand (25-40 %), Below Average Demand (40-70 %) and No Demand (> 70%)

Below are the input/output key-value pairs from the Mappers and Reducers used in this MR Job: -

Input to Mapper 1
bina_classschedule.csv

Output of Mapper 1 <Key, Value>
<CourseName_DepartmentName, StudentsEnrolled_ClassCapacity>

Input to Reducer 1 <Key, <Values>>
<CourseName_DepartmentName, <StudentsEnrolled_ClassCapacity>>

Output of Reducer 1 <Key, Values>
<CourseName_DepartmentName,
TotalEnrollment_TotalCapacity_SemestersCourseOffered>

Input to Mapper 2
Temp file created with output of Reducer 1

Output of Mapper 2 <Key, Value>
<CourseName_DepartmentName,
TotalEnrollment_TotalCapacity_SemestersCourseOffered >

Input to Reducer 1 <Key, <Values>>
<CourseName_DepartmentName,
<TotalEnrollment_TotalCapacity_SemestersCourseOffered>>

Output of Reducer 1 <Key, Values>
<CourseName, AverageEnrollment_AverageCapacity_CourseDemand>

Question 2: Most Popular Department Every Year

We see that UB consists of many departments. However, which among these is the most popular? How do we decide which one is the most popular? Answering this question can give us an idea as to which department needs more classrooms and more time slots. Since the popularity might change every year we also need to observe trends over the past few years. Therefore, our question would be, find the most popular department by number of enrollments every year.

Solution:

To solve this problem, we first read and split every value at the first mapper and then set the key to be ‘year-dept’. We set the value to be the total number of enrolled students. We pass this value to the first reducer. The reducer gets the aggregated vale for the same key. We add all enrolled students at the reducer and write down this result to ‘Temp’.

In our second mapper we read the ‘Temp’ and set the key as ‘year’ and value as ‘dept-enrolled’.
This will then be passed to the reducer. The reducer will then compare every value for a given year and determine the department with the highest enrollment for that year.

Below are the input/output key-value pairs from the Mappers and Reducers used in this MR Job: -

Input to Mapper 1
bina_classschedule.csv

Output of Mapper 1 <Key, Value>
<Year_DepartmentName, StudentsEnrolled >

Input to Reducer 1 <Key, <Values>>
<Year_DepartmentName, <StudentsEnrolled >>

Output of Reducer 1 <Key, Values>
<Year_DepartmentName, TotalEnrollment >

Input to Mapper 2
Temp file created with output of Reducer 1

Output of Mapper 2 <Key, Value>
<Year, DepartmentName_TotalEnrollment >

Input to Reducer 1 <Key, <Values>>
<Year, < DepartmentName_TotalEnrollment >>

Output of Reducer 1 <Key, Values>
<Year, PopularDept_TotalEnrollment>

Question 3: Find wasted space in every building over the years

UB is broken into buildings and each building has a lab or a lecture hall. A lot of this space is usually wasted. We decided that we wanted to find out the wasted space in every building. We can find out which buildings have been historically wasting space and which buildings have been efficient. Using this we can allot more classes in the inefficient buildings.

Solution:

To solve this problem, we first read and split every value at the first mapper and then set the key to be ‘buildingname_lectureroom’. We set the value to be the total ‘enrolledstudents_classcapacity’. We pass this value to the first reducer. The reducer adds the enrolled students for every classroom and also adds the total capacity for every room and then writes down this result to ‘Temp’.

In our second mapper we read the ‘Temp’ and set the key as ‘building_name’ and value as ‘enrolled_totalcapacity’. This will then be passed to the reducer. The reducer will then add the enrolled students for every building and also adds the total capacity for every building. Using this data, the reducer will calculate the wasted space by (totalcapacity – totalenrolled)/totalcapacity * 100.
This will generate percentage wastage for very hall.

Below are the input/output key-value pairs from the Mappers and Reducers used in this MR Job: -

Input to Mapper 1
bina_classschedule.csv

Output of Mapper 1 <Key, Value>
<BuildingName_LectureRoom, StudentsEnrolled_ClassCapacity >

Input to Reducer 1 <Key, <Values>>
<BuildingName_LectureRoom, <StudentsEnrolled_ClassCapacity>>

Output of Reducer 1 <Key, Values>
< BuildingName_LectureRoom, TotalStudentsEnrolled_TotalClassCapacity >

Input to Mapper 2
Temp file created with output of Reducer 1

Output of Mapper 2 <Key, Value>
< BuildingName, TotalEnrollment_TotalCapacity >

Input to Reducer 1 <Key, <Values>>
< BuildingName, <TotalEnrollment_TotalCapacity>>

Output of Reducer 1 <Key, Values>
<BuildingName, PercentVacant>

Question 4: Lecture Time Analysis

Class scheduling is very complex problem and its all the more difficult in a department where the enrollments are increasing. We have observed that some of the classes are small as compared to the number of students enrolled and vice versa. This not only make it difficult for the professor to deliver his/her lecture efficiently but also creates a bad image of us among international students.

Being a World Class University we do not want our students to fight for a seat during the class. Can you utilize the “UB Course Scheduling data” to provide an insight on what time of the day have remained most occupied with lectures during the years and what is the most suitable time to re-schedule these courses, so that we can allocate them class of proper size?

Solution:

We are using lecture timings and corresponding days of week to get an insight on which time the classes remain idle and can be utilized more efficiently**.** We tackle it using 2 step MR Job. In the 1st step we calculating the total number of lectures being held at a given time and the day than in the 2nd step finding the busiest time of the day for each year.

This information will not only help us analyze that at a particular time how many classes are being scheduled but also provide insight on busiest time of the day and the time when classes remain idle. Thus we can convince student affairs committee to re-schedule class timings of some of the courses and allocate a bigger classroom to them.

Below are the input/output key-value pairs from the Mappers and Reducers used in this MR Job: -

Input to Mapper 1
bina_classschedule.csv

Output of Mapper 1 <Key, Value>
<Semester_LectureTime_DayOfWeek, LecturesCount>

Input to Reducer 1 <Key, <List of Value>>
<Semester_LectureTime_DayOfWeek, <LecturesCount >>

Output of Reducer 1 <Key, Value>
<Semester_LectureTime_DayOfWeek, TotalLecturesCount >

Input to Mapper 2
Temp file created with output of Reducer 1

Output of Mapper 2 <Key, Value>
<Semester_DayOfWeek, LectureTime_TotalLecturesCount >

Input to Reducer 1 <Key, <List of Values>>
<Semester_DayOfWeek, <LectureTime_TotalLecturesCount> >

Output of Reducer 1 <Key, Value>
<Semester_DayOfWeek, MostBusyTime>

Question 5: Find the most popular exam slots in every building

Exam season is stressful for everyone. For the students and especially for the staff. They have to ensure classes are allotted and everyone finds their place. From the 6 years' exam data that we have we aim to find the most popular exam slots in every building. By this data we can find out what time slot most exams are held. We can further analyze this data and make decisions to move some of the exams at this time slot to another. We can also make decisions to move them to other halls to.

Solution:

To solve this problem, we first read and split every value at the first mapper and then set the key to be 'ExamHall_StartTime'. We set the value to be the 'one'. We pass this value to the first reducer. The reducer adds the 'ones' for every exam hall and then writes down this result to 'Temp'.

In our second mapper we read the ‘Temp’ and set the key as ‘Building Name’ and value as ‘Sum of all slots’. This will then be passed to the reducer. The reducer will then add the sums for every building. We now have the output for number of slots per hall.

Using this we can move/remove certain slots to over under-assigned halls.

Below are the input/output key-value pairs from the Mappers and Reducers used in this MR Job: -

Input to Mapper 1
bina_classschedule.csv

Output of Mapper 1 <Key, Value>
<ExamHall_StartTime, one >

Input to Reducer 1 <Key, <Values>>
< ExamHall_StartTime, <one>>

Output of Reducer 1 <Key, Values>
< BuildingName, StartTime_Sum >

Input to Mapper 2
Temp file created with output of Reducer 1

Output of Mapper 2 <Key, Value>
< BuildingName, Slot_Sum >

Input to Reducer 1 <Key, <Values>>
< BuildingName, <Slot_Sum>>

Output of Reducer 1 <Key, Values>
<BuildingName_Slot, NoOfSlots>