Big Data Systems: Assignment3

Deadline 12 26, 2024 at 23:59 PM

LEE CHIH-PIN NDHU CSIE

Problem 1

In Spark cluster, we combine Hbase and Spark to process the YouBike.csv data, try to find the top 10 stations with the most available bikes. The data is stored in Hbase:

Pseudo code

Data: Youbike dataset: $D = \{\text{Time}, \text{Station}, \text{Action}\}$
Result: Top 10 stations per hour based on total activity: $\{\text{Hour}, \text{Station}, \text{Rank}\}$
Initialize Spark Session with app name `"Youbike Analysis"`
Load Data: Read dataset D into DataFrame **Extract Hour:** Add column Hour = hour(Time)
Aggregate Activity:

- Group by $\text{Hour}, \text{Station}, \text{Action}$ to compute counts: $C(\text{Action})$.
- Pivot Action into columns Borrow and Return.
- Replace NULL $\rightarrow 0$ for missing values.

Calculate Total Activity:

$$\text{Total Activity} = \text{Borrow} + \text{Return}$$

Rank Stations:

- Define window $\mathcal{W}(\text{Hour})$ ordered by Total Activity in descending order.
- Compute rank: $\text{Rank} = \text{rank}_{\mathcal{W}}(\text{Total Activity})$.

Filter Top Stations: Retain rows where Rank ≤ 10
Order Results: Sort by Hour, Rank
Select Columns: $\{\text{Hour}, \text{Station}, \text{Borrow}, \text{Return}, \text{Total Activity}, \text{Rank}\}$
Display Top 100 Rows
Stop Spark Session

Algorithm 1: the top 10 busiest stations every hour

Explanation

This algorithm analyzes Youbike station data to determine the top 10 busiest stations for each hour based on total activity, which is the sum of borrow and return counts. The key steps are detailed below:

1. **Initialize Spark Session:** A SparkSession is created to enable distributed data processing.

2. **Load and Preprocess Data:** - The dataset D is loaded into a DataFrame from a CSV file, with schema inference enabled.
- A new column, `Hour`, is derived by extracting the hour component from the timestamp in the `Time` column.

3. **Aggregate Activity:** - The data is grouped by `Hour`, `Station`, and `Action` to compute counts for each action (`Borrow` and `Return`).
- The `Action` column is pivoted to create separate columns for borrow and return counts, with missing values filled as zero.

4. **Calculate Total Activity:** - The total activity for each station is calculated as:

$$\text{Total Activity} = \text{Borrow} + \text{Return}$$

5. **Rank Stations by Activity:** - A window specification, $\mathcal{W}$(Hour), is defined to rank stations within each hour by total activity in descending order.
- The rank for each station is assigned using the `rank` function.

6. **Filter Top Stations:** - Stations with a rank Rank ≤ 10 are retained for further analysis.

7. **Order and Display Results:** - The results are sorted by `Hour` and `Rank`.
- Relevant columns (`Hour`, `Station`, `Borrow`, `Return`, `Total Activity`, `Rank`) are selected and displayed.

8. **Stop Spark Session:** - The Spark session is terminated to release resources.

This algorithm is efficient for identifying activity patterns across stations on an hourly basis, facilitating further analysis of Youbike usage trends.

Result

Hour	Station	Borrow	Return	Total Activity	Rank
0	Station_10	8	7	15	1
0	Station_6	9	4	13	2
0	Station_5	9	4	13	2
0	Station_1	9	2	11	4
0	Station_2	9	2	11	4
0	Station_4	8	3	11	4
0	Station_8	8	2	10	7
0	Station_7	6	4	10	7
0	Station_3	7	2	9	9
0	Station_12	4	3	7	10
0	Station_9	4	3	7	10
1	Station_9	9	8	17	1
1	Station_11	10	7	17	1
1	Station_6	11	4	15	3
1	Station_4	5	9	14	4
1	Station_12	5	9	14	4
1	Station_7	7	7	14	4
1	Station_5	6	6	12	7
1	Station_10	8	4	12	7
1	Station_8	6	6	12	7
1	Station_1	5	5	10	10
1	Station_2	3	7	10	10
2	Station_6	8	11	19	1
2	Station_3	10	8	18	2
2	Station_5	7	7	14	3
2	Station_2	8	5	13	4
2	Station_11	5	8	13	4
2	Station_4	5	7	12	6
2	Station_9	6	6	12	6
2	Station_1	4	7	11	8
2	Station_7	7	4	11	8
2	Station_10	3	7	10	10
2	Station_8	8	2	10	10
3	Station_8	6	11	17	1
3	Station_9	10	6	16	2
3	Station_3	7	9	16	2
3	Station_1	8	6	14	4
3	Station_11	7	7	14	4
3	Station_12	6	7	13	6
3	Station_4	8	4	12	7

Figure 1: The result of top 10 busiest station in terminal

Problem 2

Now Based on bike IDs, find the hottest rent return station pairs.

Pseudo code

Data: Youbike dataset: $D = \{\text{Bike ID}, \text{Time}, \text{Station}, \text{Action}\}$
Result: Top 10 station pairs ranked by rent/return frequency: $\{(\text{Rent Station}, \text{Return Station})\}$
Initialize Spark Session: Create a Spark session named `"Youbike Rent/Return Analysis"`
Load Data: Read dataset D from the file `Youbike.csv` into a DataFrame
Separate Borrow and Return Records:

- Extract borrowing records as:

$$D_{\text{borrow}} = \{(b, t_b, s_r) \mid b \in \text{Bike ID}, t_b \in \text{Borrow Time}, s_r \in \text{Rent Station}\}$$

- Extract returning records as:

$$D_{\text{return}} = \{(b, t_r, s_r) \mid b \in \text{Bike ID}, t_r \in \text{Return Time}, s_r \in \text{Return Station}\}$$

Join Borrow and Return Data:

- Perform an inner join on D_{borrow} and D_{return} using Bike ID as the key:

$$D_{\text{pairs}} = \{(b, t_b, t_r, s_r, s'_r) \mid b \in \text{Bike ID}, t_r > t_b\}$$

  Where t_b and t_r represent the borrow and return times, and s_r, s'_r denote rent and return stations respectively.

Aggregate Station Pairs:

- Group by (s_r, s'_r) (rent and return station pairs) and count occurrences:

$$C(s_r, s'_r) = |\{(b, t_b, t_r) \mid b \in \text{Bike ID}, t_r > t_b, (s_r, s'_r) \in D_{\text{pairs}}\}|$$

- Order pairs (s_r, s'_r) by $C(s_r, s'_r)$ in descending order.

Display Top Station Pairs:

- Select the top 10 station pairs:

$$\text{Top}_k = \text{argmax}^k_{(s_r, s'_r)} C(s_r, s'_r)$$

Stop Spark Session: Terminate the Spark session to release resources

Algorithm 2: Based on bike IDs, find the hottest rent/return station pairs

Explanation

This algorithm processes Youbike dataset records to analyze rent and return activity between stations. The steps are as follows:

1. **Initialize Spark Session:** - A Spark session named `"Youbike Rent/Return Analysis"` is created to perform distributed data processing.

2. **Load Data:** - The dataset D is loaded from the file `Youbike.csv` into a Spark DataFrame. The dataset contains fields: `Bike ID`, `Time`, `Station`, and `Action`.

3. **Separate Borrow and Return Records:** - Borrowing records are filtered where `Action` = `"Borrow"`. The relevant fields extracted are:

$$D_{\text{borrow}} = \{\text{Bike ID}, \text{Borrow Time}, \text{Rent Station}\}$$

- Returning records are filtered where `Action` = `"Return"`. The relevant fields extracted are:

$$D_{\text{return}} = \{\text{Bike ID}, \text{Return Time}, \text{Return Station}\}$$

4. **Join Borrow and Return Data:** - A join operation is performed between D_{borrow} and D_{return} using `Bike ID` as the key. The join condition ensures:

$$\text{Return Time} > \text{Borrow Time}$$

- This results in a combined dataset:

$$D_{\text{pairs}} = \{\text{Bike ID}, \text{Borrow Time}, \text{Return Time}, \text{Rent Station}, \text{Return Station}\}$$

5. **Aggregate Station Pairs:** - Rent and return station pairs (Rent Station, Return Station) are grouped, and their occurrences are counted:

$$C(s_r, s_r') = |\{(b, t_b, t_r) \mid b \in \text{Bike ID}, t_r > t_b, (s_r, s_r') \in D_{\text{pairs}}\}|$$

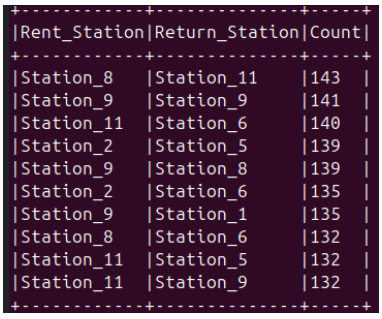
- These pairs are then sorted by count in descending order.
6. **Display Top Station Pairs:** - The top 10 station pairs with the highest rent/return activity are selected:

$$\text{Top}_{10} = \text{argmax}^{10}_{(s_r, s_r')} C(s_r, s_r')$$

7. **Stop Spark Session:** - The Spark session is terminated to release system resources.

This algorithm identifies patterns in Youbike rent and return activities, specifically focusing on the most frequent station pairs. It facilitates understanding of user movement trends between stations.

Result



Rent_Station	Return_Station	Count
Station_8	Station_11	143
Station_9	Station_9	141
Station_11	Station_6	140
Station_2	Station_5	139
Station_9	Station_8	139
Station_2	Station_6	135
Station_9	Station_1	135
Station_8	Station_6	132
Station_11	Station_5	132
Station_11	Station_9	132

Figure 2: The result of Based on bike IDs, find the hottest rent/return station pairs

Problem 3

The purpose of this analysis is to study YouBike (bike-sharing system) usage data to identify patterns of peak activity. Specifically, the objectives are as follows:

1. **Hourly Activity Analysis:**
   - Calculate the number of bikes borrowed and returned at each station during different hours of the day.
   - Aggregate the total activity, defined as the sum of borrow and return actions, for each station at every hour.
2. **Identify the Busiest Hour at Each Station:**
   - Determine the hour with the highest total activity for each station, referred to as the *busiest hour*.
3. **Rank Stations by Peak Activity:**
   - Compare all stations based on the total activity during their busiest hour.
   - Rank stations in descending order of their peak activity.
4. **Provide Insights:**
   - Generate a ranked list of stations with details about their busiest hour, including:
     - The hour of peak activity.
     - Total activity during that hour.
     - Counts of borrow and return actions.

Problem 3 continued on next page...

Pseudo code

Data: Youbike dataset: $D = \{\text{Station}, \text{Time}, \text{Action}\}$
Result: Stations ranked by their peak usage hours: $\{(\text{Station}, \text{Hour}, \text{Total Activity})\}$
Initialize Spark Session: Create a Spark session named `"Youbike Peak Usage Analysis"`
Load Data: Read dataset D from the file `Youbike.csv` into a DataFrame
Extract Hour: Add a column Hour by extracting the hour component from Time:

$$D_{\text{hourly}} = D \cup \{\text{Hour} = \text{hour}(\text{Time})\}$$

Calculate Total Activity:

- Group data by Station and Hour.
- Compute borrow and return counts:

$$\text{Borrow\_Count} = \sum_{a \in D_{\text{hourly}}} [a.\text{Action} = \text{Borrow}]$$

$$\text{Return\_Count} = \sum_{a \in D_{\text{hourly}}} [a.\text{Action} = \text{Return}]$$

- Calculate total activity:

$$\text{Total\_Activity} = \text{Borrow\_Count} + \text{Return\_Count}$$

Identify the Busiest Hour for Each Station:

- Define a window specification $\mathcal{W}(\text{Station})$ to rank rows by Total_Activity in descending order.
- Assign rank r:

$$r = \text{rank}_{\mathcal{W}(\text{Station})}(\text{Total\_Activity})$$

- Filter rows where $r = 1$:

$$D_{\text{busiest}} = \{(s, h, a) \mid r = 1, (s, h, a) \in D_{\text{hourly}}\}$$

Rank Stations by Peak Usage:

- Define a global window $\mathcal{W}_{\text{global}}$ to rank rows by Total_Activity in descending order.
- Assign an overall rank:

$$\text{Overall\_Rank} = \text{rank}_{\mathcal{W}_{\text{global}}}(\text{Total\_Activity})$$

Display Results:

- Select columns: $\{\text{Station}, \text{Hour}, \text{Total Activity}, \text{Borrow Count}, \text{Return Count}, \text{Overall Rank}\}$.
- Sort by Overall Rank.
- Display the top results.

Stop Spark Session: Terminate the Spark session to release resources

Algorithm 3: Youbike Peak Usage Analysis

Problem 3 continued on next page...

Explanation

This algorithm analyzes Youbike station usage data to identify the peak usage hour for each station and ranks stations based on their busiest periods. The steps are as follows:

1. **Initialize Spark Session:** - A Spark session named `"Youbike Peak Usage Analysis"` is initialized to process the dataset.

2. **Load Data:** - The dataset $D = \{\text{Station}, \text{Time}, \text{Action}\}$ is loaded from the file `Youbike.csv` into a Spark DataFrame.
- Each record represents an activity (borrow or return) performed at a station at a specific time.

3. **Extract Hour:** - A new column, `Hour`, is created by extracting the hour component from the `Time` field:

$$\text{Hour} = \text{hour}(\text{Time})$$

4. **Calculate Total Activity:** - The data is grouped by `Station` and `Hour`. - Borrow and return counts are computed as:

$$\text{Borrow\_Count} = \sum_{i \in D} [i.\text{Action} = \text{Borrow}]$$

$$\text{Return\_Count} = \sum_{i \in D} [i.\text{Action} = \text{Return}]$$

- The total activity for each station-hour combination is then calculated:

$$\text{Total\_Activity} = \text{Borrow\_Count} + \text{Return\_Count}$$

5. **Identify the Busiest Hour for Each Station:** - A window specification $\mathcal{W}(\text{Station})$ is defined to rank rows by `Total_Activity` in descending order for each station. - The rank is assigned as:

$$r = \text{rank}_{\mathcal{W}(\text{Station})}(\text{Total\_Activity})$$

- Rows with $r = 1$ are selected to identify the busiest hour for each station.
6. **Rank Stations by Peak Usage:** - A global window $\mathcal{W}_{\text{global}}$ is defined to rank stations based on their peak `Total_Activity` across all stations. - The overall rank is assigned as:

$$\text{Overall\_Rank} = \text{rank}_{\mathcal{W}_{\text{global}}}(\text{Total\_Activity})$$

7. **Display Results:**
- The results include the station, its busiest hour, borrow and return counts, total activity, and overall rank.
- These are ordered by `Overall Rank` and displayed.

8. **Stop Spark Session:**
- The Spark session is terminated to release system resources.
This algorithm provides insights into station usage patterns by identifying peak activity periods and ranking stations based on their busiest hours.

Result

Station	Hour	Total_Activity	Borrow_Count	Return_Count	Overall_Rank
Station_5	11	19	8	11	1
Station_2	7	19	9	10	1
Station_6	2	19	8	11	1
Station_3	2	18	10	8	4
Station_11	5	18	9	9	4
Station_10	9	18	10	8	4
Station_12	17	18	6	12	4
Station_9	7	18	8	10	4
Station_8	10	17	12	5	9
Station_8	3	17	6	11	9
Station_8	20	17	7	10	9
Station_1	21	16	8	8	12
Station_1	6	16	9	7	12
Station_7	18	16	8	8	12
Station_7	16	16	8	8	12
Station_4	10	15	9	6	16

Figure 3: The result of Peak hour in terminal