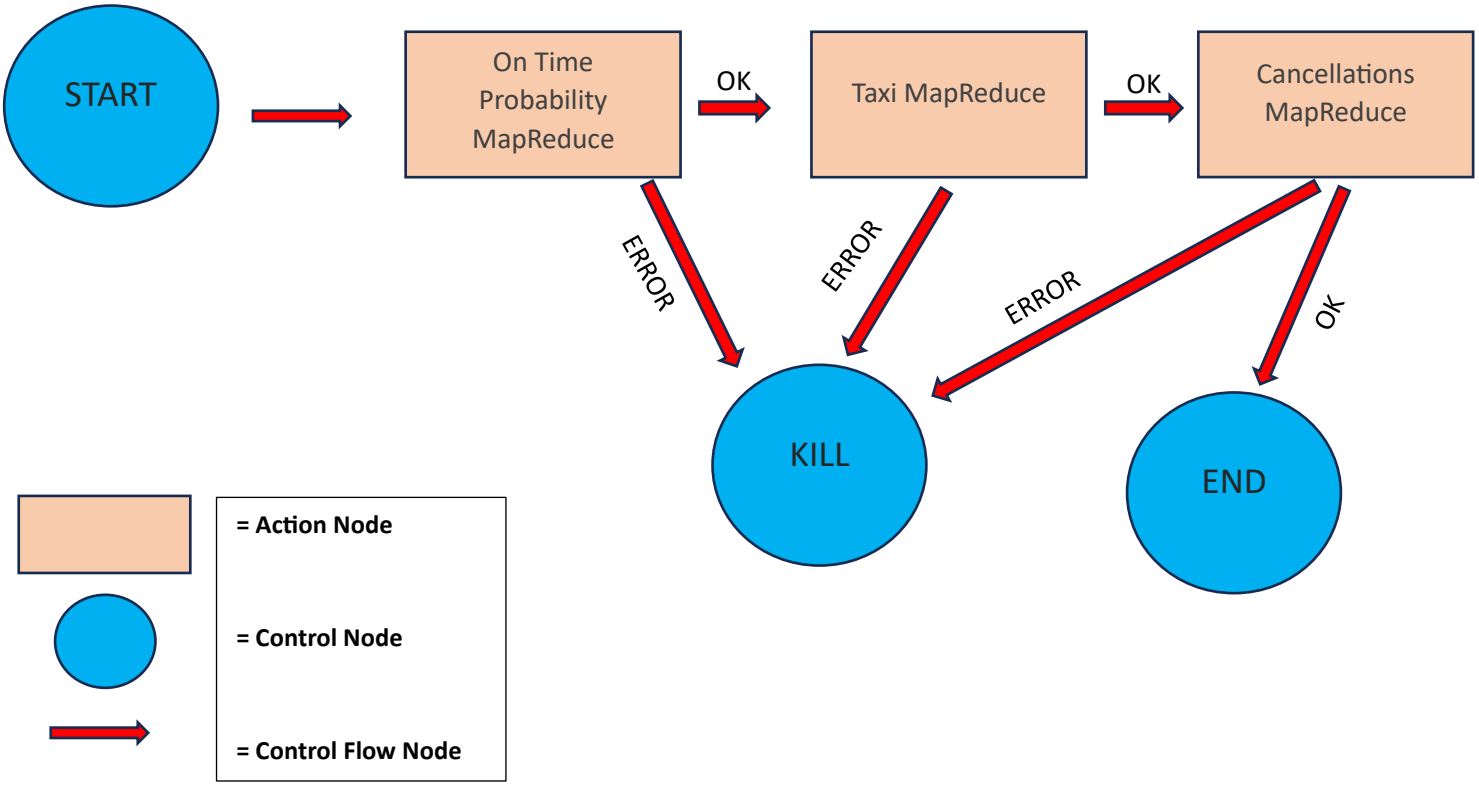
Flight Data Analysis

Hani Shakrah

Professor Wu

DS 644

Oozie Workflow Structure



On-Schedule Probability Algorithm

Mapper:

1. Input text converted to string
2. Line is split with comma as the delimiter. An array of field values is created
3. Skip the header row (when x[0] is “Year”)
4. Extract arrival delay field value, make sure this is not NA
5. Convert this value to an integer
6. If it is less than 15 (minutes), consider this a not a delay. Assign a variable (“ontime”) to 1
7. Extract the carrier’s name
8. Write the carrier and ontime value as a key-value pair

Reducer:

1. Create new Hash Map

2. Inside reduce method, create new array list
3. Sum of list values is computed, followed by its average
4. Averages are stored in the hash map, with the carrier name as key
5. This is sorted in descending order by average values, then converted into a stream
6. The stream is collected into a list
7. The highest 3 and lowest 3 averages are written alongside the corresponding carrier name

Average Taxi Time Algorithm

Mapper:

1. Convert input text to string
2. Split with comma as delimiter, store in array
3. Extract the origin, destination, taxi-in, and taxi-out fields
4. If row is not header and fields aren't "NA", convert values to integers
5. Emit key-value pairs of origin/taxi out and destination/taxi in

Reducer:

1. Create new empty list
2. Collect all taxi times as integers, put in list
3. Calculate average taxi time
4. Write average taxi time or the exception message
5. Create a hash map
6. Calculate and store average times for each airport
7. Write top 3 highest and top 3 lowest times, following the corresponding airport

Cancellation Reasons Algorithm

Mapper:

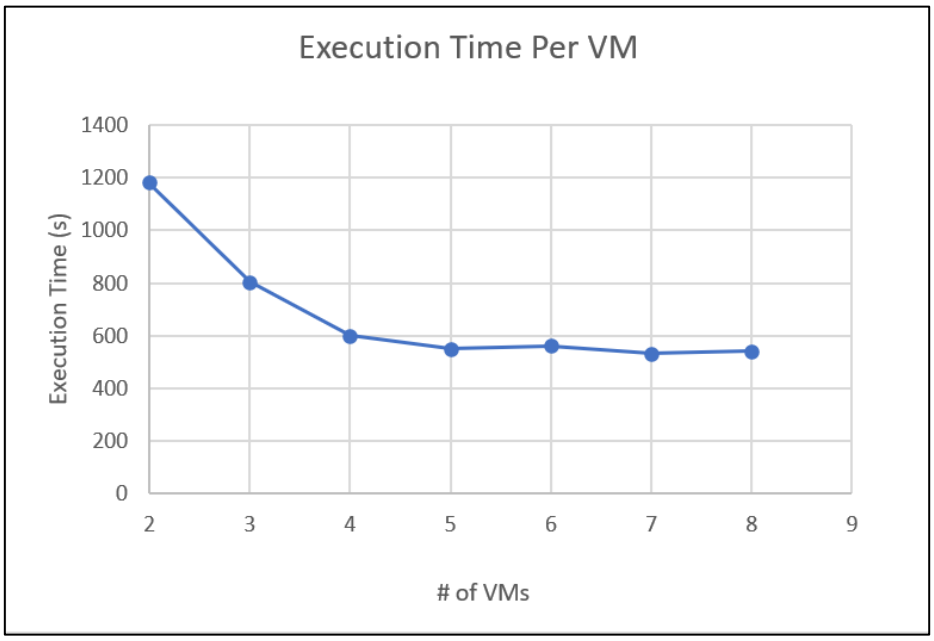
1. Initialize constant ONE as value 1
2. Initialize variable to store cancellation reason
3. Convert each line of input into a string
4. If row is not header row, split using comma as delimiter
5. Extractor cancellation fields, trim and store
6. If indicator is 1 and code is not NA or empty, write the cancellation reason (A,B,C, or D) and ONE

Reducer:

1. Initial count variable to store count of occurrences for each reason
2. Convert intwritable to an integer
3. Sum all integers

4. Write output for current cancellation reason along with total count

Execution Time Plots



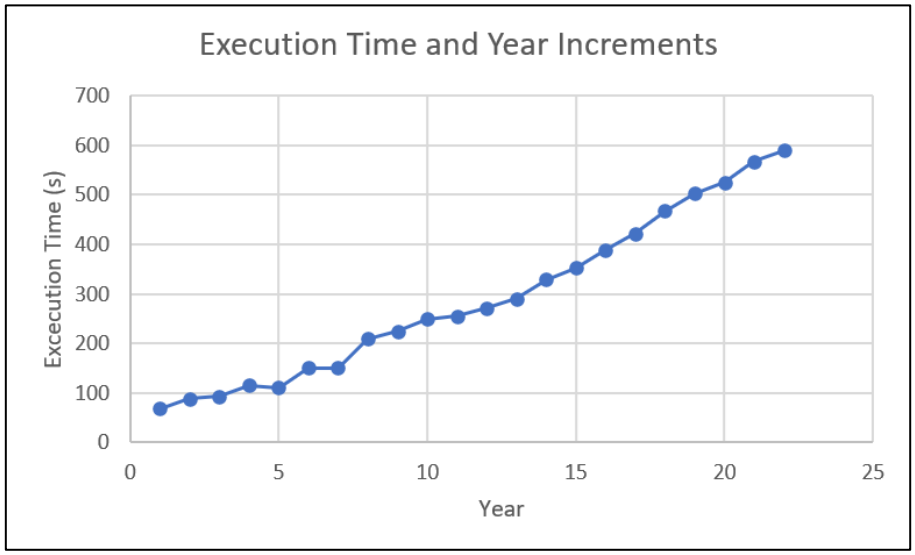
The plot above depicts the relationship between the execution time of my Hadoop/Oozie job and the number of virtual machines (VMs) utilized for the computation. The x-axis represents the number of VMs, while the y-axis signifies the execution time in seconds. Analyzing the plot reveals a pattern that highlights the impact of VM count on the efficiency of the job.

Initially, when only 2 virtual machines are employed, the execution time is at a high. This is likely due to the overhead associated with setting up and coordinating the parallel processing environment, as well as contention for resources among the limited VMs.

As the number of VMs gradually increases beyond 2, the execution time begins to decrease. This decrease in execution time can be attributed to the inherent parallelism offered by Hadoop's distributed processing framework. With more VMs working simultaneously, the workload can be split into smaller tasks that are processed concurrently. This parallel processing leads to a reduction in the overall time required to complete the job.

The plot shows a point where the reduction in execution time starts to stabilize after a certain number of VMs, specifically around 4 in this case. This suggests that adding more VMs beyond this point has little to no significant impact on further reducing the execution time. The law of diminishing returns comes into play, where the benefits of additional parallelism begin to level off.

From an optimization perspective, using 4 VMs seems to strike a balance between execution time and resource utilization. Compared to a smaller number of VMs, 4 VMs result in a faster run time due to increased parallelism and efficient resource utilization. On the other hand, employing more VMs might slightly reduce execution time, if at all, but can also lead to increased resource consumption without significant gains in job efficiency.



The above plot provides insight into execution time and number of years/amount of data analyzed. Each point along the x-axis corresponds to a distinct count of years, with the number increasing from the starting point of 1987. For instance, a year labeled as "1" signifies the year 1987, while "2" signifies data from both 1987 and 1988, etc.

Evidently, the plot reveals a pattern: as the scope of years expands, the execution time of the cluster grows. This observation is in line with the expectations of the distributed computing learning taught in class.

The quantity of virtual machines (VMs) remains consistent across all scenarios, regardless of the extent of years included. The computational resources available for each task remain fixed. Thus, an augmentation in data leads to a more substantial computational workload for a given cluster to manage.