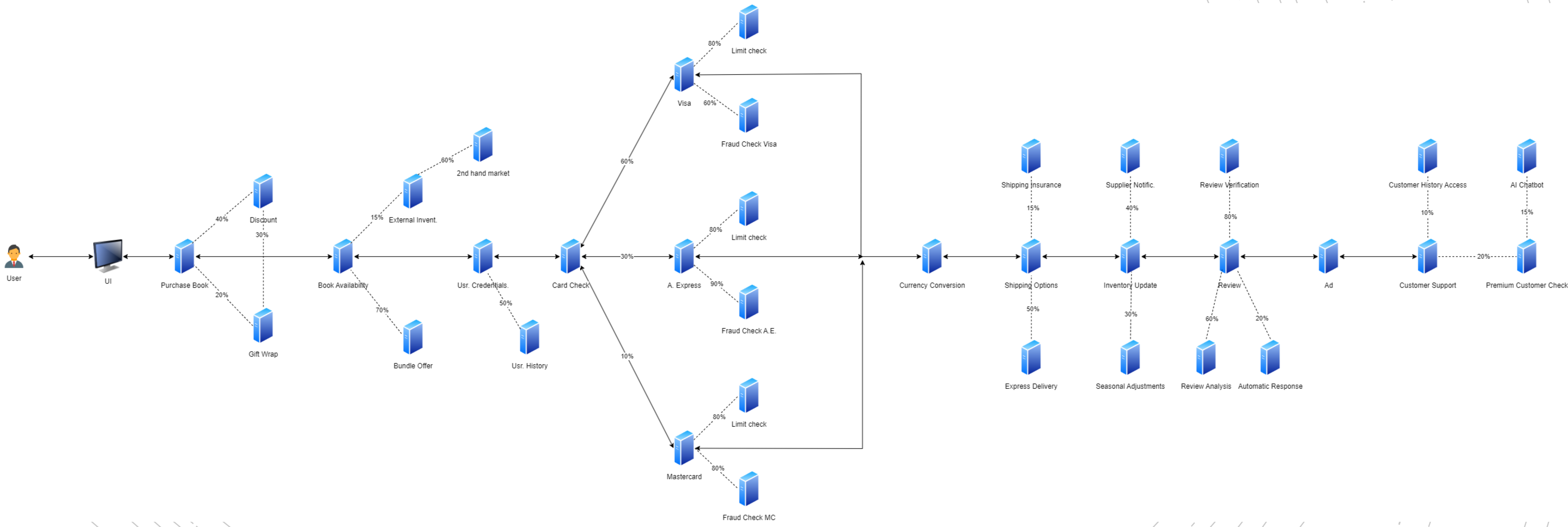
Data Intensive Systems – Logs Management

Finding similar processes from large datasets using Spark

Data Generation

- Variety in servers and process depth
  - Different credit card checks, multiple inventory checks, shipping calling insurance.
  - Multiple servers for each server type (e.g., visa_1, visa_2, . . . etc.)
- Large datasets with different combinations generated for experimentation and testing
  - Combinations based on predefined probabilities



Part 1 Approach

1 Preprocess Dataset

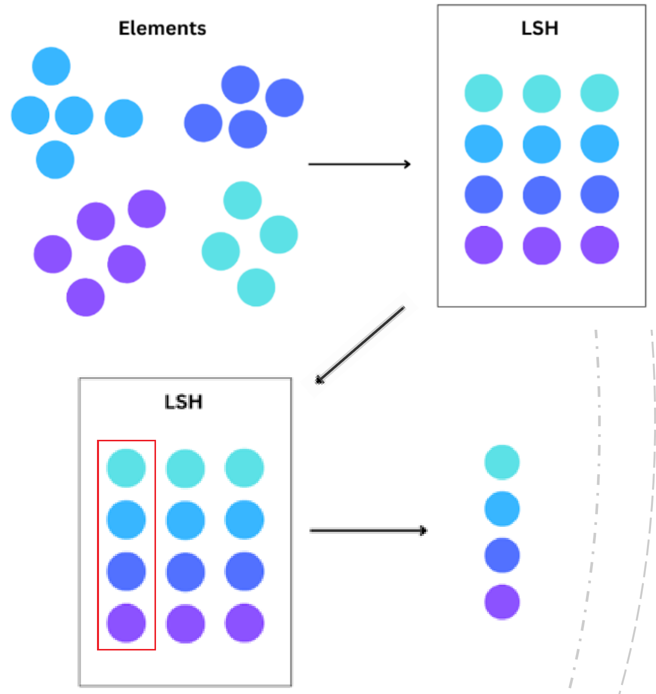
- Split the log data in columns
- Group splitted logs by ‘process_id’
- Sort paths according to ‘from_server’ names so that processes with the same server visits in different order are similar
- Remove common substrings, spaces, and special characters

2 Create Shingles

- Create shingles with length equal to 30% of average server name (or length 2 for short server names)
- Extract vocabulary of unique shingles from dataset
- Create vectors of shingle occurrences in process paths

3 Perform MinHashing and LSH

- Use Spark’s MinHashLSH function to hash sparse shingle occurrence vectors for each process
- Set number of hash tables (hashing functions) to 20



4 Find Similar Processes

- With the hashing from MinHashLSH we found similar process pairs using a Jaccard distance threshold of 0.5
- Group similar processes together and keep only 1 process to represent the group

Part 2 Approach

1

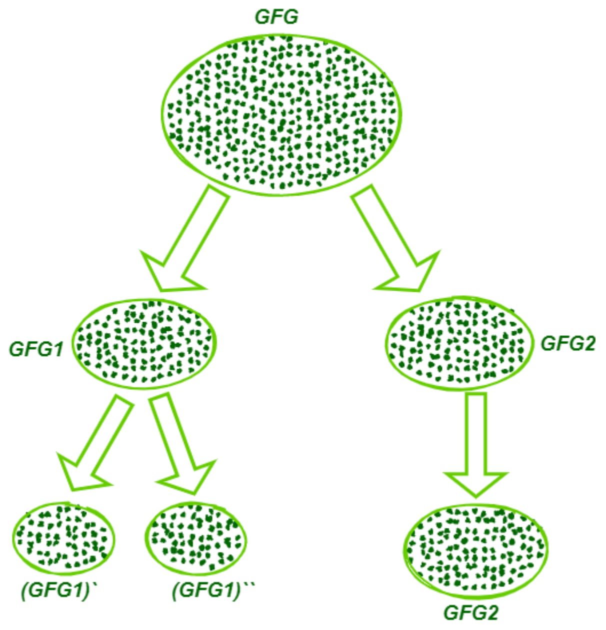
Load Processes from Part 1

- Use resulting processes from Part 1 that were kept from each group of similar processes

2

Prepare Features for Clustering

- Use one-hot-encoding of unique server names from all loaded processes
  - Features of 1 and 0 for every server name for every process
- Sum the time taken to complete a process as another feature



3

Cluster Features

- Use Spark's Bisecting K-Means implementation from MLlib to cluster process features
  - Hybrid approach between partitional and hierarchical clustering
  - Better results than K-Means as number of clusters increases
- Automatically determine number of cluster centers (k) using Elbow method and Kneedle algorithm
  - Set minimum k as 2 and maximum k as 1/2 of total number of processes to ensure clusters are returned
  - Compute Within Set Sum of Squared Error (WSSSE) plot and find elbow point automatically
- Each cluster center is considered a similar process group

Why Spark?

Parallelization & Distributed Computation

- Resilient Distributed Datasets (RDDs) for partitioning of data
- Parallelization across partitions with mapping
- Horizontal scaling to handle larger datasets

Speed

- In-memory computation, significantly faster than disk-based processing
- Optimized execution plants for minimized data shuffling and recomputation

Machine Learning & Data Handling

- MLlib offering different machine learning algorithms, from which bisecting k-means was used
- Dataframes for data manipulation, aggregation, and querying used across the project

Thank you for your time

Questions?

Group 21

Christos Papageorgiou

Artificial Intelligence

Marios Iacovou

Artificial Intelligence