ID2221 Data Intensive Computing - Review Questions 5 - Group 19

by Artem Sliusarenko, Mihailo Cvetkovic and Eugen Lucchiari Hartz

1.) Assume we have two types of resources in the system, i.e., CPU and Memory. In total we have 28 CPU and 56GB RAM (e.g., 1 CPU = 2 GB). There are two users in the systems. User 1 needs ⟨2CPU, 2GB⟩ per task, and user 2 needs ⟨1CPU, 4GB⟩ per task. How do you share the resources fairly among these two users, considering (i) the asset fairness, and (ii) DRF.

(I) Share resources according to asset fairness:

We need to calculate asset fairness.
max(x, y)
2x + y <= 28 (CPU constraint)
2x + 4y <= 56 (RAM constraint)

Considering we can equate **1 CPU = 2 GB** of RAM we can say that User 1 needs **2 CPU + 1 CPU(2 GB RAM) = 3** and User 2 needs **1 CPU + 2 CPU (4 GB RAM) = 3.** We can use this values to define the fairness formula below.
3x = 3y

The calculation yielded the following resource distribution:
Decimals are rounded to the floor integer value.
x = 9 (64% CPU, 32% GB) (sum = 96%)
y = 9 (32.% CPU, 64% GB) (sum = 96%)
Despite each user receiving a sum of 96% of shared resources the above distribution still violates the **share guarantee** defined by the asset fairness approach, which states that each user is guaranteed to get at least 1/n of the resources. **User 1** received less than **50% GB of RAM**. **User 2** received less than **50% CPU.**

(II) Share resources according to Dominant Resource Fairness (DRF)

First we need to identify the dominant resource for each user.
User 1 wants **(2CPU, 2 GB)**: Dominant resource is: **CPU (2/56 < 2/28)**
User 2 wants **(1 CPU, 4 GB)**: Dominant resource is: **RAM (1/28 < 4/56)**

Defining constraints:
max(x, y)
2x + y <= 28 (CPU constraint)
2x + 4y <= 56 (RAM constraint)

Fairness is defined by the dominant resources:
2x/28 = 4y/56
2x/28 because dominant resource of User 1 is CPU
4x/56 because dominant resource of User 2 is RAM

The calculation yielded the following resource distribution:
Decimals are rounded to the floor integer value.
x = 9 (64% CPU, 32% GB) (sum = 96%)
y = 9 (32.% CPU, 64% GB) (sum = 96%)

2.) What are the similarities and differences among Mesos, YARN, and Borg?

Mesos, YARN, and Borg are all cluster management systems designed to manage and schedule resources in large-scale distributed computing environments. Their primary goal is to maximize resource utilization by enabling multiple frameworks or applications to share the same cluster and thereby providing flexibility and efficient resource sharing. All three systems support the execution of multiple frameworks on a single cluster and offer mechanisms for allocating resources like CPU and memory.

Despite their similarities, Mesos, YARN, and Borg differ in several key aspects, including their origin, resource abstraction, architecture, and openness. Mesos uses a fine-grained resource allocation strategy and employs a two-level scheduling model, making it highly adaptable. YARN, originally developed for Hadoop, uses containers for resource management and also follows a two-level scheduling architecture. Borg, on the other hand, was developed by Google for internal use and features a more abstract resource model and a unique architectural design suited for Google's infrastructure needs.

3.) What are the differences between Warehouse and Datalake? What is Lakehouse?

A Data Warehouse is a centralized storage system developed in the 1980s, specifically designed to handle structured data with well-defined schemas. It is optimized for SQL-based analytics and Business Intelligence (BI) tasks and focuses on data quality, consistency, and high-performance querying.

In contrast, a Data Lake emerged in the 2010s and offers a more flexible and cost-effective approach to data storage. Unlike Data Warehouses, Data Lakes can handle various data types, including structured, semi-structured, and unstructured data. They allow schema definition after the data is stored which makes them highly adaptable and suitable for storing raw data.

A Lakehouse combines the strengths of both Data Warehouses and Data Lakes. It integrates the data management and performance features of a Data Warehouse with the open and flexible storage capabilities of a Data Lake. This hybrid approach provides the high-performance analytics of a Data Warehouse while retaining the cost-effectiveness and flexibility of a Data Lake, so it offers "the best of both worlds" for modern data storage and analytics.

4.) Briefly explain how Deltlake handles concurrent writing on the same file.

Delta Lake handles concurrent writing on the same file using an optimistic concurrency control mechanism. Here's how it works:

1. **Transactional Logs:** Every write operation in Delta Lake is treated as a transaction and logged in a transaction log. This log keeps track of all changes made to the dataset.
2. **Versioning:** Each write operation creates a new version of the dataset, ensuring that older versions are preserved and can be accessed or restored if needed.
3. **Conflict Detection:** Before committing a write, Delta Lake checks for conflicts by comparing the changes being written with the latest version of the data. If multiple writers attempt to modify the same part of the dataset, a conflict is detected.
4. **Resolution:** If a conflict occurs, Delta Lake rejects the transaction for the conflicting writer, who must then reattempt the write operation based on the updated state of the data.

This approach ensures data consistency and prevents issues like file corruption, even with multiple concurrent write operations.