ID2221 - Data Intensive Computing - Review Questions 5

Deepak Shankar Reethika Ambatipudi
deepak | reethika @kth.se

October 3, 2023

Q1. 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.

Answer:
Total resources: 28 CPU and 56GB RAM (e.g., 1 CPU = 2 GB)

- User 1 has x tasks and wants $< 2CPU, 2GB >$ per task
- User 2 has y tasks and wants $< 1CPU, 4GB >$ per task

Asset fairness yields:

```
max(x, y)
2x + y <= 28
2x + 4y <= 56

//Equalize total value given to each user
user1 - <2,1> // 2CPU, 2GB, where 2 GB = 1 CPU.
user2 - <1,2> // 1CPU, 4GB, Where 4GB = 2 CPU.

3x = 3y -> x = y

Solving the above 2 equations by replacing x=y,
x = y = 9

User 1: x = 9: 18CPUs(64%), 18GB RAM(32%)
User 2: y = 9: 9CPUs(32%), 36GB RAM(64%)
```

DRF:

```
Equalize the dominant share of the users.
    Total resources: < 28CPU, 56GB >
    User 1 wants < 2CPU, 2GB > ; Dominant resource: CPU(2/28 > 2/56)
    User 2 wants < 1CPU, 4GB > ; Dominant resource: RAM (1/28 < 4/56)

    max(x, y)
    2x + y <= 28
    2x + 4y <= 56

    2x/28 = 4y/56
=>  x/14 = y/14
=>   x = y

    Solving the above 2 equations by replacing x=y,
    x = y = 9

    User 1: x = 9: 18CPUs(64%), 18GB RAM(32%)
    User 2: y = 9: 9CPUs(32%), 36GB RAM(64%)
```

Q2. What are the similarities and differences among Mesos, YARN, and Borg?

Mesos, YARN, and Borg are all resource management and cluster orchestration systems used to efficiently manage and allocate resources in large-scale distributed computing environments.

Similarities:

1. **Resource Management:** All three systems efficiently manage resources in distributed computing environments.
2. **Multi-Tenancy:**They support multiple applications or users sharing cluster resources.
3. **Container Support:** Mesos, YARN, and Borg facilitate containerization for deploying applications.
4. **Job Scheduling:**They include scheduling mechanisms to allocate resources to applications or tasks.
5. **Fault Tolerance:**All three systems have mechanisms for ensuring system reliability.

Differences:

	Mesos	Yarn	Borg
Origin	Developed at UC Berkeley	Apache Hadoop Project	Developed at Google
Open Source	Yes	Yes	No (Internal to Google)
Container Support	Mesos supports Docker, etc.	YARN supports Docker, etc.	Borg uses Google containers
Fine-Grained Sharing	Yes	No (Coarse-Grained)	Yes
Resource Types	CPU, Memory, GPU, etc.	CPU, Memory, GPU, etc.	CPU, Memory, etc.
Job Scheduling	Pluggable schedulers	Capacity and Fair schedulers	Google's Omega scheduler

Q3. What are the differences between Warehouse and Datalake? What is Lakehouse?

Warehouses, data lakes, and lakehouses are all terms related to data management, and each serves a specific purpose in handling and analyzing data.

- **Data Warehouse:**
  - **Structure:** Data warehouses are structured repositories that organize data in a tabular, relational format. They typically use structured query language (SQL) for data retrieval and analysis.
  - **Data Type:** Data warehouses primarily store structured data, which is highly organized and follows a predefined schema. This makes them suitable for transactional and analytical data.
  - **Data Processing:** Data warehouses are optimized for complex querying and reporting. They often involve Extract, Transform, Load (ETL) processes to clean, transform, and load data into a structured format.
  - **Storage Cost:** Data warehousing solutions can be expensive due to the need for specialized hardware, software, and ongoing maintenance.
  - **Use Case:** Data warehouses are commonly used for business intelligence, reporting, and decision support applications. They are suitable for structured data analysis.
- **Data Lake:**
  - **Structure:** Data lakes are less structured and can store a wide variety of data types, including structured, semi-structured, and unstructured data. They don't require a predefined schema.

  - **Data Type:** Data lakes can store diverse data formats, including raw files, logs, images, videos, JSON, XML, and more. This flexibility makes them suitable for big data and analytics.

  - **Data Processing:** Data lakes support various data processing engines and tools, including batch processing, stream processing, and machine learning. They allow for data exploration and schema-on-read.

  - **Storage Cost:** Data lakes are often more cost-effective than data warehouses, as they can use scalable and affordable cloud storage solutions.

  - **Use Case:** Data lakes are used for data storage, exploration, and analysis. They are particularly valuable for big data and data science applications.

- **Lakehouse:** Lakehouse is a relatively newer concept that combines the best aspects of data lakes and data warehouses. It aims to provide the flexibility and scalability of data lakes while incorporating the structured query capabilities of data warehouses. Here are some key characteristics:

  - **Unified Platform:** Lakehouse integrates data storage and data processing on a single platform, typically built on top of cloud-based data lakes. It allows users to query and analyze data using SQL.

  - **Schema Evolution:** Lakehouse supports schema evolution, meaning data can be stored without a predefined schema, but a schema can be applied when needed for analysis.

  - **Performance:** Lakehouse solutions are designed for high-performance analytics and reporting, providing near real-time query capabilities.

  - **Data Governance:** They offer data governance features, including access control, auditing, and data lineage, to ensure data quality and compliance.

  - **Use Case:** Lakehouse solutions are suitable for a wide range of use cases, including data warehousing, data lakes, and big data analytics. They aim to bridge the gap between structured and unstructured data.

In summary, data warehouses are structured, suitable for structured data, and optimized for SQL-based querying. Data lakes are less structured, flexible, and suitable for diverse data types and analytics. Lakehouse combines the advantages of both by offering a unified platform that supports structured and unstructured data with SQL-based analytics capabilities.

Q4. Briefly explain how Delta lake handles concurrent writing on the same file.

Delta Lake is an open-source storage layer that brings ACID transactions to Apache Spark and big data workloads. It is commonly used in lakehouse architectures. When it comes to concurrent writing on the same file, Delta Lake employs a mechanism known as optimistic concurrency control. Here's a brief explanation:

- **Metadata Management:**
  - Delta Lake maintains metadata about the transactions and the data stored in the lake. This metadata includes information about the transaction log, schema, and other important details.
- **Transaction Log:**
  - All write operations in Delta Lake are logged in a transaction log. This log maintains a record of all the changes made to the data, including inserts, updates, and deletes.
- **Delta Protocol:**
  - Delta Lake uses a protocol known as the Delta Protocol to coordinate transactions. This protocol ensures that multiple transactions can be executed concurrently without conflicting with each other.
- **Optimistic Concurrency Control:**
  - When multiple transactions attempt to write to the same file concurrently, Delta Lake uses optimistic concurrency control. Each transaction is assigned a unique identifier, and the system checks for conflicts during the commit phase.
- **Conflict Resolution:**
  - If there are conflicting changes (e.g., two transactions trying to update the same data), Delta Lake identifies the conflict during the commit phase. It then rolls back one of the transactions and allows the other to proceed.
- **Isolation:**
  - Delta Lake provides isolation between transactions, ensuring that the changes made by one transaction are not visible to others until the transaction is committed.
- **ACID Transactions:**
  - Delta Lake follows the ACID (Atomicity, Consistency, Isolation, Durability) properties, ensuring that transactions are atomic, consistent, isolated, and durable even in the presence of concurrent writes.

In essence, Delta Lake's optimistic concurrency control, along with the Delta Protocol and transaction log, enables multiple transactions to write to the same file concurrently while maintaining data consistency and integrity. It provides a robust mechanism for handling concurrent writes in a distributed and parallel processing environment.