CMT 428 Module 5: Advanced Hadoop Topics

Topic	Description	Practical Examples
YARN (Yet Another Resource Negotiator)	YARN is the cluster resource management layer in Hadoop that decouples resource management from the processing framework. It allows for running diverse applications and frameworks on a Hadoop cluster, not just MapReduce.	- Running a Spark application on a Hadoop cluster managed by YARN. - Deploying a real-time stream processing application with Storm on YARN. - Using YARN to manage resources for machine learning workloads.
Spark for In-Memory Processing	Spark is a fast and general-purpose cluster computing system that provides high-level APIs in Java, Scala, Python and R. It excels at iterative computation and interactive data analysis due to its in-memory processing capabilities, making it significantly faster than traditional MapReduce.	- Analyzing large datasets in real-time for fraud detection. - Performing interactive data exploration and visualization on massive datasets. - Building machine learning models using Spark's MLlib library. - Processing streaming data from social media platforms for sentiment analysis.
Kafka for Real-Time Data Streaming	Kafka is a distributed streaming platform used for building real-time data pipelines and streaming applications. It provides high-throughput, fault-tolerance, and scalability for handling real-time data feeds.	- Building a real-time fraud detection system that analyzes transaction streams. - Creating a real-time dashboard for monitoring website traffic and user behavior. - Ingesting and processing sensor data from IoT devices. - Streaming log data for real-time analysis and alerting.
Hadoop Performance Tuning and Optimization	This involves understanding the various factors that affect Hadoop performance and applying optimization techniques to improve efficiency and throughput.	- Tuning HDFS parameters like block size and replication factor for optimal data storage. - Optimizing MapReduce jobs by adjusting the number of mappers and reducers. - Configuring YARN parameters to efficiently manage cluster resources. - Using data compression techniques to reduce storage and network I/O. - Monitoring and analyzing Hadoop cluster performance using tools like Ganglia and Ambari.

More examples

Advanced Hadoop Topic	Description	Practical Examples
YARN (Yet Another Resource Negotiator)	YARN is Hadoop's resource management layer that allows different data processing engines, such as MapReduce, to run and share resources effectively. It decouples the resource management and scheduling functions from the data processing component, making Hadoop more scalable and flexible.	- **Multi-tenancy**: Manage multiple data processing frameworks (like Spark and MapReduce) running simultaneously in the same Hadoop cluster. - **Capacity Scheduler**: Allocate resources dynamically for different jobs, ensuring efficient resource use in a large enterprise environment.
Spark for In-Memory Processing	Apache Spark is an open-source distributed computing system that performs in-memory data processing. Unlike MapReduce, which writes intermediate results to disk, Spark retains data in memory, providing a faster alternative for processing big data workloads.	- **Real-Time Analytics**: Use Spark for processing and analyzing streaming data from IoT devices. - **Data Science**: Perform complex machine learning algorithms on large datasets in a Spark cluster using its in-memory processing capabilities.
Kafka for Real-Time Data Streaming	Apache Kafka is a distributed event streaming platform capable of handling real-time data feeds. It allows real-time publishing, storing, and processing of event data, making it essential for applications requiring real-time analytics.	- **Real-Time Data Pipelines**: Build a data pipeline using Kafka to ingest and process live data from social media platforms. - **Log Aggregation**: Use Kafka to aggregate and stream logs from distributed systems for real-time monitoring.
Hadoop Performance Tuning and Optimization	This involves fine-tuning various Hadoop components (such as HDFS and MapReduce) for optimal performance. It includes adjusting parameters, resource allocation, and configuring the environment to improve job execution efficiency and resource utilization.	- **MapReduce Job Optimization**: Fine-tune parameters like the number of mappers and reducers to decrease job runtime. - **HDFS Block Size Tuning**: Adjust the HDFS block size to optimize storage and improve

Advanced Hadoop Topic	Description	Practical Examples
		data read/write speeds for specific workloads.