Contents

Data Processing Patterns

This document covers data processing patterns and strategies for system design.

Components

Batch Processing

- **Characteristics:** Process large volumes of data in scheduled chunks or batches
- **Benefits:** High throughput, cost-effective resource usage, simpler error handling and recovery
- **Challenges:** Higher latency, delayed insights, less responsive to real-time needs
- **Technologies:** Apache Spark, Hadoop MapReduce, AWS Batch, Google Dataflow
- **Use Cases:** ETL pipelines, data warehousing, financial reporting, log analysis

Stream Processing

- **Characteristics:** Process data continuously as it arrives in real-time streams
- **Benefits:** Low latency, real-time insights, immediate responsiveness to events
- **Challenges:** Higher complexity, increased resource requirements, harder error handling
- **Technologies:** Apache Kafka Streams, Apache Storm, AWS Kinesis, Google Cloud Dataflow
- **Use Cases:** Real-time analytics, fraud detection, IoT data processing, live dashboards

Lambda Architecture (Hybrid)

- **Approach:** Combines both batch and stream processing layers
- **Speed Layer:** Real-time processing for immediate results
- **Batch Layer:** Comprehensive processing for accuracy and completeness
- **Serving Layer:** Merges results from both layers for queries
- **Benefits:** Balances latency and throughput, fault tolerance, comprehensive data coverage

Processing Pattern Comparison

- **When to Use Batch Processing**
  - Large volumes of data that don't require immediate processing
  - Cost optimization is important
  - Complex analytics and reporting
  - Historical data analysis
  - Scheduled operations (nightly reports, data backups)

- **When to Use Stream Processing**
  - Real-time decision making required
  - Low latency is critical
  - Continuous monitoring and alerting
  - Live user interactions
  - Time-sensitive business operations
- **When to Use Lambda Architecture**
  - Need both real-time and comprehensive analytics
  - High availability requirements
  - Complex data processing pipelines
  - Balance between speed and accuracy
  - Large-scale data processing with mixed requirements

Related Trade-offs

Batch Processing vs Stream Processing

- **Summary:** Batch processing handles large volumes of data in scheduled chunks, optimizing for throughput and cost efficiency. Stream processing handles data continuously as it arrives, optimizing for low latency and real-time insights.
- **Trade-off:** High throughput and resource efficiency vs. low latency and real-time responsiveness.
- **Processing Comparison:**
  - **Batch Processing:** High throughput, cost-effective, simpler error handling, but higher latency and delayed insights
  - **Stream Processing:** Low latency, real-time processing, immediate insights, but higher complexity and resource requirements
  - **Lambda Architecture (Hybrid):** Combines both batch and stream processing to balance throughput and latency
- **Questions to Ask:**
  - What's the acceptable delay between data arrival and processing results?
  - Is the data volume predictable or highly variable?
  - Are real-time insights critical for business decisions?
  - What's the cost tolerance for processing infrastructure?
  - How complex are the data transformations and analytics required?
  - Can the system tolerate occasional processing delays or must it be always responsive?