Big Data Processing Architectures and their Role in the Future of AI Systems

1. Analytical Processing Foundations

The Role of data warehouses and analytical engines in modern data architectures:

Many modern data architectures rely heavily on data warehouses and analytical processing engines to properly process the large volumes of data that are accumulated every day.

A **data warehouse** is a centralized storage system which collects data from all kinds of different sources, for example applications, transaction databases or business systems. Data warehouses are specifically designed for analysis, reporting and long-term decision making, rather than just focusing on daily operations.

The **Analytical engines** run on top of the data warehouses and execute complex queries, for example calculating the trend, averages or correlations across millions of records. They often use distributed computing and column-oriented storage to handle these huge workloads efficiently by focusing on scanning only the relevant parts of large datasets.

1.1. What distinguishes analytical processing from transactional processing?

Transactional processing (OLTP) systems support real-time operations such as payments, bookings or order processing. They can handle many small write transactions and typically store the data in rows, which makes them fast to insert and update individual records.

Analytical processing (OLAP) systems are optimized for reading and aggregating large amounts of historical data in smaller batch sizes. They use column-based storage because it improves their performance for analytical queries since only the needed columns must be accessed.

A main advantage of separating the OLAP and OLTP workloads is that it prevents analytical queries from slowing down the operational systems.

1.2. How do indexes, materialized views and query optimization support analytical workloads?

Indexes such as B+ trees provide structured access paths improving filtering and range queries by making them faster. Another example would be Log structured Merge trees (LSM), they are often used for large scale data ingestions because they can handle continuous data writes efficiently.

Materialized views are useful because they store precomputed results of frequent queries, e.g. daily revenue totals. This avoids recalculating the same aggregations repeatedly. Many systems also support incremental view maintenance which means that whenever new data arrives, only the affected parts of the view will be updated instead of recomputing everything again.

Query optimizers improve the workloads even further by automatically selecting efficient execution strategies.

Putting all these technologies together, we get a strong foundation that enables scalable data analysis and supports downstream workloads (e.g. AI model - or business intelligence training).

2. Streaming, Event Processing and CDC

How do stream processing systems, event streaming platforms and CDC pipelines complement or replace batch-oriented analytics:

Traditional analytical systems process data in large batches, whereas modern data architectures increasingly rely on event-driven technologies and streaming to handle continuous data flows. The benefit of **stream processing systems** (such as e.g. Apache Flink or Spark Structured Streaming) is that they are designed to process the data while it is being generated, rather than waiting for it to be stored first. This is immensely helpful for organizations and allows them to compute metrics, detect anomalies or even update dashboards in near real time. Thus, stream processors operate incrementally by updating results as new events arrive instead of processing the entire stored datasets.

Event streaming platforms such as Apache Kafka serve as foundations for these real-time architectures. As we have learnt in class, Kafka functions as a distributed event log where systems consume and publish streams of records.
As an example: when a user interacts with an application, the user's activity can be recorded as an event and can then be sent to Kafka. Multiple downstream systems (e.g.

analytical engines, monitoring tools, AI services, etc.) can then consume the same event stream independently. Decoupling the data producers from the consumers creates fault-tolerant and scalable data pipelines.

The **Change Data Capture** (CDC) pipelines are based on the same logic and extend this model by tracking all the changes in the transactional databases and converting them into event streams. The benefit of CDC tools is that they capture inserts, updates and delete as they happen instead of periodically where entire tables would have to be copied into the analytical systems. As a result, analytical environments are continuously synchronized with the operational data sources and batch-oriented analytics can be partially or sometimes fully replaced by incremental processing approaches.

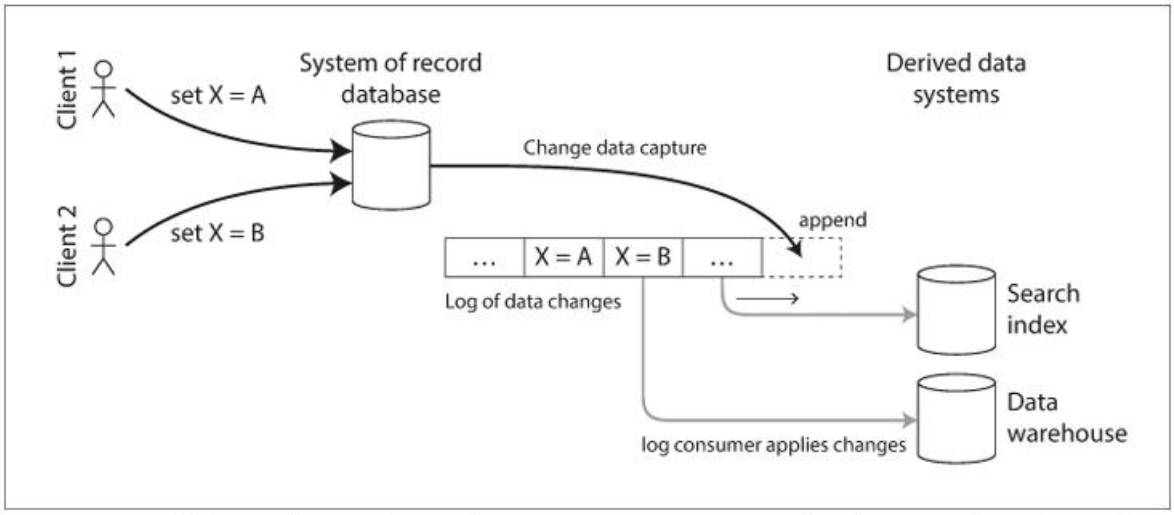


Figure 11-5. Taking data in the order it was written to one database, and applying the changes to other systems in the same order.

2.1. What are the differences and overlaps between stream processing and incremental analytics?

Even though stream processing and incremental analytics share similarities, they operate at different layers, meaning that stream processing focuses on transforming and analyzing live event flows, whereas incremental analytics update the stored query results (e.g. such as materialized views). Both approaches aim to reduce the re-computations and make sure that the latest data is available. Their differences are that analytics emphasize query efficiency while streaming prioritizes immediacy of the data.

2.2. Discuss the trade-offs between latency and consistency

One of the central trade-offs in these systems is latency versus consistency. Low-latency systems are designed to deliver rapid insights but might rely on eventual consistency, meaning results are temporarily approximate and consistency is reached eventually.
On the other hand systems that prioritize strong consistency ensure highly accurate results but can introduce delays in e.g. processing. Modern architectures balance these factors depending on the requirements of the application – we cannot generalize them as one size fits all.

The combination of streaming platforms, CDC pipelines and incremental processing frameworks help the data systems to move from static batch computations to real-time, continuous analytics.

3. Implications for AI Systems

Analyze how the above systems support or constrain modern AI workloads

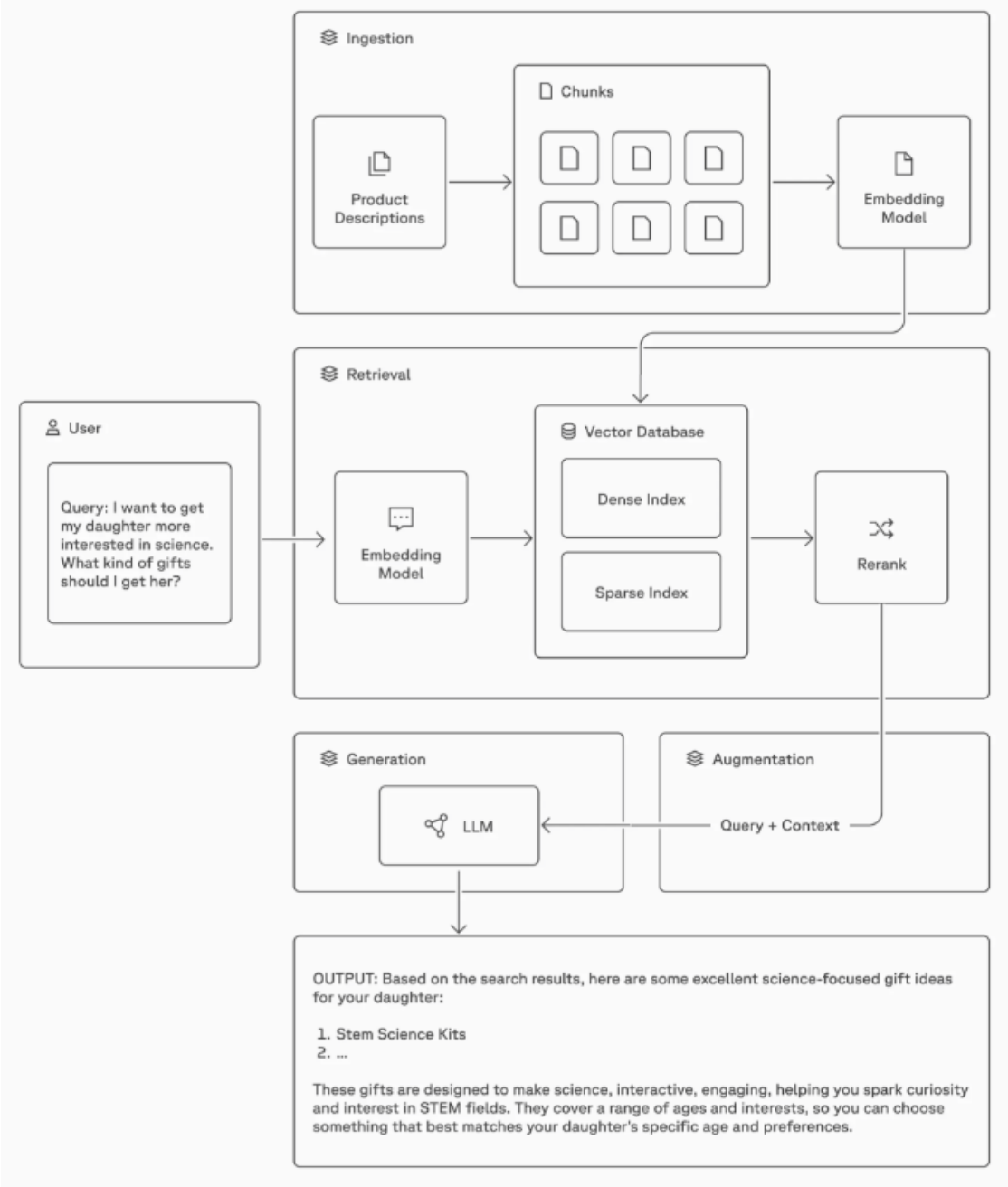
Modern AI workloads place new and often extreme demands on data processing architectures. The systems that were originally designed for reporting or business intelligence must now also support autonomous agents, real-time data retrieval and machine learning pipelines. To achieve all of this, scalable architectures that provide data “freshness” and high query performance are needed.

3.1. Training and fine-tuning LLMs

One major workload of modern AI is the training and fine tuning of large language models (LLMs). Training LLMs requires massive datasets collected from logs, documents, user interactions or external knowledge sources, etc. . Typically, these datasets are stored and prepared for use in analytical environments such as data lakes or warehouses. Before models can be trained, the data must be cleaned, transformed and structured through large scale batch processing pipelines. Analytical engines are especially handy in this case because they can aggregate, filter and join large amounts of training data efficiently. Fine-tuning workflows also depend on continuous updates, especially if we adapt our models to new domains or when dealing with user feedback.

3.2. Retrieval Augmented Generation

Retrieval Augmented Generation (RAG) is another important architectural pattern that retrieves external information at query time (instead of only relying on the static model knowledge). This means that for example when a user asks a question, the system searches document indexes or vector databases to find context that is relevant and then passing it to the language model. This approach improves the factual accuracy and helps with working with up-to-date information. For this to work, retrieval systems must support low-latency queries, efficient indexing and continuous data ingestion so newly added knowledge becomes searchable fast.



3.3. AI agents that continuously generate data, feedback and queries

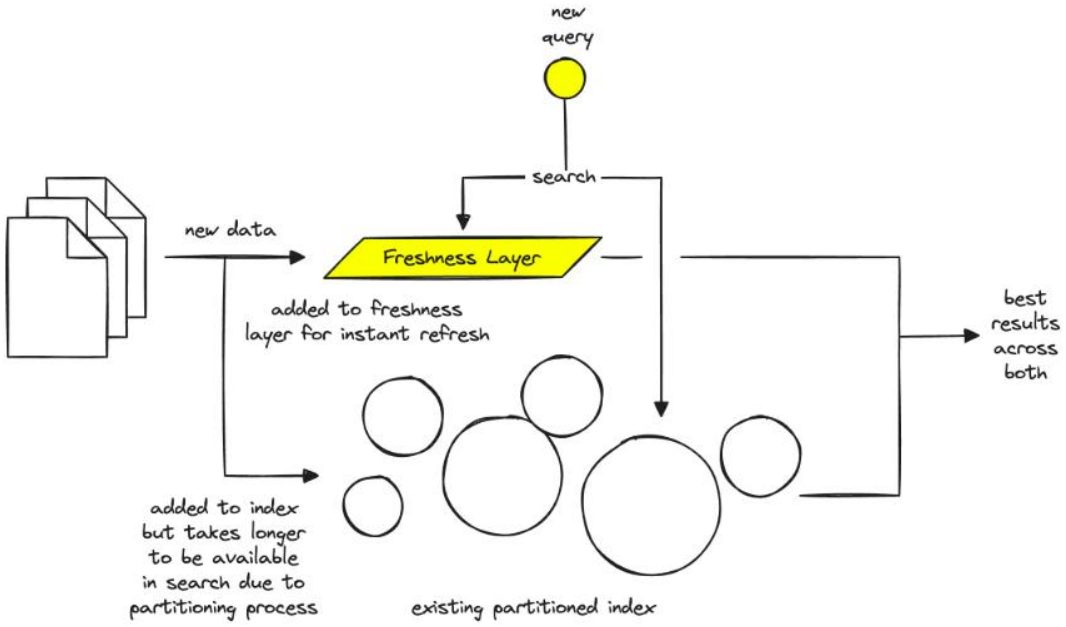
AI agents introduce an additional layer of complexity. Unlike traditional applications, agents generate data through actions, conversations and feedback loops. For example, an AI assistant might log conversations, tool usage or decision outcomes. This creates a constant stream of new events that again must be stored, processed and made available for learning or retrieval in a proper way. The systems must be able to handle frequent updates as well as heavy query traffic, as they produce both – high write volumes and high read amplifications.

3.4. Why do AI systems amplify the importance of incremental computation?

At this level, recomputing entire datasets whenever new data arrives would be computationally infeasible - thus architectures rely on incremental indexing, stream updates and continuously maintained views.

3.5. How do high query rates and freshness requirements affect system design?

Freshness becomes a critical design factor, especially for RAG and agent systems where the quality and recency of the information directly affect the response quality. As a result, modern AI data infrastructures must balance historical large-scale processing with real-time updates, generating pipelines that combine analytical, streaming and serving technologies.



4. Technical Positioning and Future Outlook

Taking a clear technical position on the future:

4.1. Will AI workloads push data systems toward unified architectures or deeper specialization?

In my opinion, AI workloads will push data systems toward a mix of both unified architectures and deeper specialization rather than fully committing to only one direction.

On the one hand, unified architectures are becoming more attractive because AI pipelines require access to historical, real-time and operational data at the same time. Lakehouses or HTAP (hybrid transactional/analytical platforms) already try to combine these workloads in a single environment. This reduces data movement and simplifies the system design, making it especially useful for training and analytics in machine learning where historical datasets and fresh updates must be processed together.

On the other hand, deeper specialization is still necessary because AI introduces completely new requirements that the traditional systems were not designed for. For example, vector databases are optimized for embedding similarity search, or feature stores are built to specifically manage machine learning features across training and inference. Such highly specialized workloads cannot be handled efficiently by general purpose warehouses alone. Because of this, I think the future will not replace one or the other but rather layer specialization on top of unified core systems.

4.2. Which components become more central (streaming, incremental views, serving layers)?

I believe streaming platforms, incremental views and serving layers will all become central components of future AI data stacks because they each solve different but complimentary problems.

- **Streaming** engines like Flink or Kafka will be important for ingesting and processing data in real time, to ensure that AI systems always work with the latest information.
- **Incremental views** and continuous materialized caches will reduce the re-computation overhead, this helps analytics and feature updates to keep pace with high-velocity data.

- **Serving layers** (including vector search and feature stores) will be essential for delivering low-latency responses for e.g. RAG, operational AI workloads and real-time inference.

4.3. What architectural principles are expected to matter most in the next 5-10 years?

In terms of architectural principles that are likely to matter in the next 5-10 years, I would argue that:

First, **incremental-first design** will definitely play a key role as systems will be expected to update views, indexes and feature stores continuously rather than just relying on periodic batch refreshes.

Second, **event-driven architectures** will be a central part with platforms like Kafka sort of acting as a nervous system that connects producers, processors and consumers easily.

Third, the **separation of storage and compute** will also remain important, allowing for elastic scaling of the processing power while avoiding moving or duplicating massive datasets.

Finally, I also think that **multi-modal indexing and serving layers** (supporting SQL, vector and graph queries) will become essential for AI applications in order to get rapid and flexible access to structured as well as unstructured data.

Overall, I think that the future of data systems will neither purely be unified or specialized. Instead, I'm expecting architectures that push toward hybrid models where the core storage and computation converge and dedicated layers that serve the specific needs of AI applications. Hybrid approaches are particularly useful in my opinion because they maximize the flexibility, scalability and performance which helps in handling the growing volume, velocity and variety of data in AI driven systems.

Resources:

- The Log-Structured Merge-Trees (LSM-Trees)
- https://docs.oracle.com/en/database/oracle/oracle-database/19/dwhsg/basic-materialized-views.html
- https://kafka.apache.org/41/getting-started/introduction/
- https://docs.confluent.io/kafka/introduction.html
- https://nightlies.apache.org/flink/flink-docs-stable/docs/concepts/overview/4
- Paper: Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks
- https://developers.openai.com/api/docs/guides/model-optimization
- https://www.pinecone.io/learn/vector-database/
- https://docs.databricks.com/gcp/en/lakehouse-architecture/scope
- https://docs.feast.dev/
- https://cratedb.com/blog/what-is-htap-hybrid-transactional-analytical-processing-explained
- https://celerdata.com/glossary/hybrid-transactional-analytical-processing
- PDF: Designing Data-Intensive Applications by Martin Kleppmann