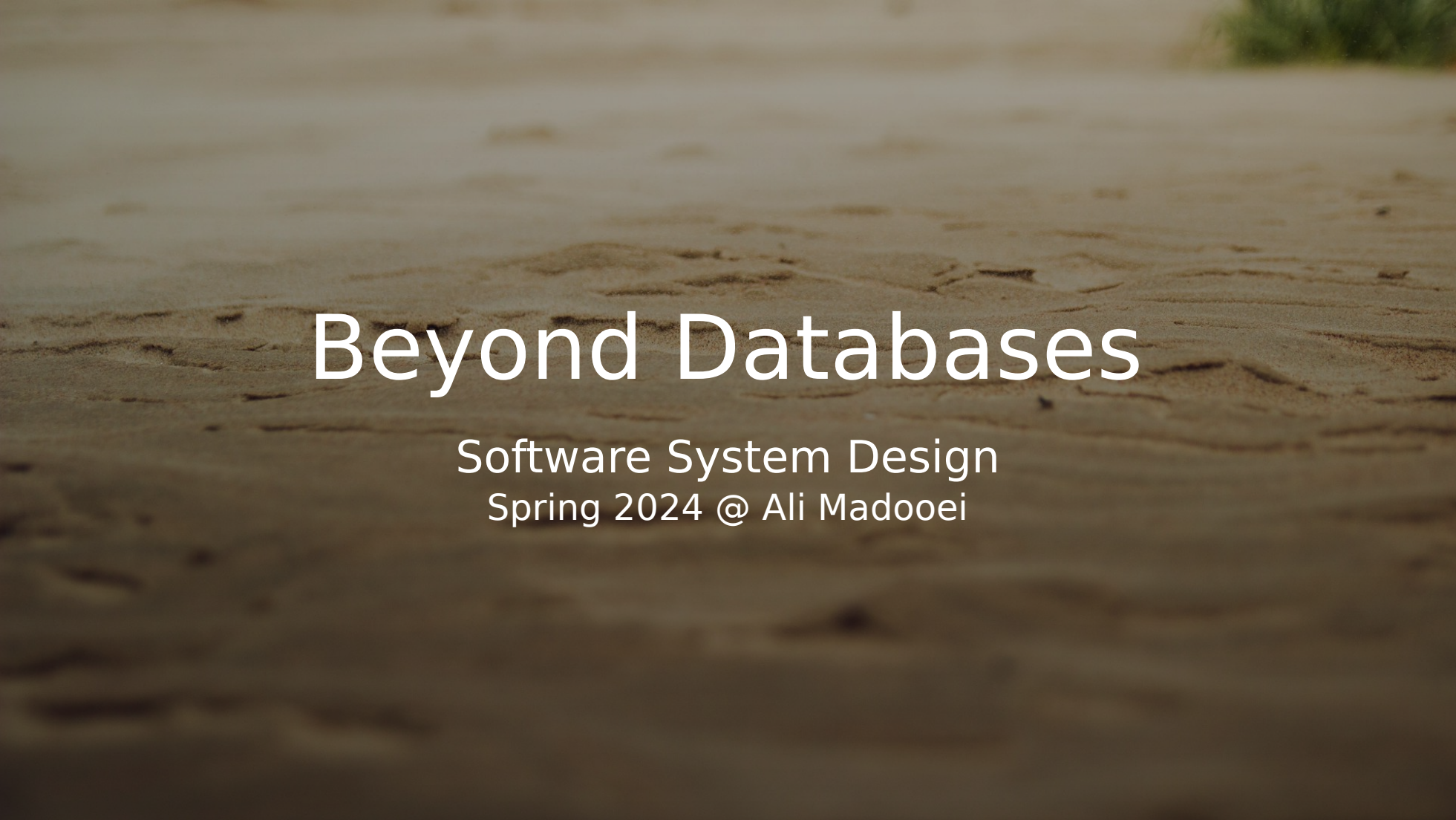


Beyond Databases

Software System Design

Spring 2024 @ Ali Madooei

Learning Outcomes

By the end of this lecture, you should be able to:

- Explain the role of caching, object storage, and big data technologies in software system design.
- Identify use cases where caching, object storage, and big data technologies can be applied to improve system performance and scalability.
- Evaluate different providers and services for caching, object storage, and big data technologies in cloud environments.

Use Case: E-Commerce Platform

An e-commerce platform enables customers to browse and purchase products online. The platform needs to manage a large catalog of products, handle high traffic volumes, and provide personalized shopping experiences. Key features include product search, shopping cart management, payment processing, and order tracking. The platform must ensure fast page loads, secure transactions, and scalable storage for product images and customer data.

Use Case: Streaming Service

A streaming service offers a wide range of multimedia content, including movies, TV shows, and music, to subscribers. The platform must support high-quality streaming, content discovery, and personalized recommendations. Key challenges include managing a vast content library, ensuring low-latency streaming, and analyzing user preferences to curate content. The service aims to provide an engaging and seamless viewing experience across various devices.

Use Case: Healthcare Analytics Platform

A healthcare analytics platform provides tools for analyzing and visualizing medical data to improve patient care and operational efficiency. It integrates data from electronic health records, medical imaging, wearable devices, and genomic research. The platform supports predictive analytics, clinical decision support, and population health management. Key objectives include enhancing diagnostic accuracy, optimizing treatment plans, and identifying trends in patient outcomes.

Caching: What & Why…

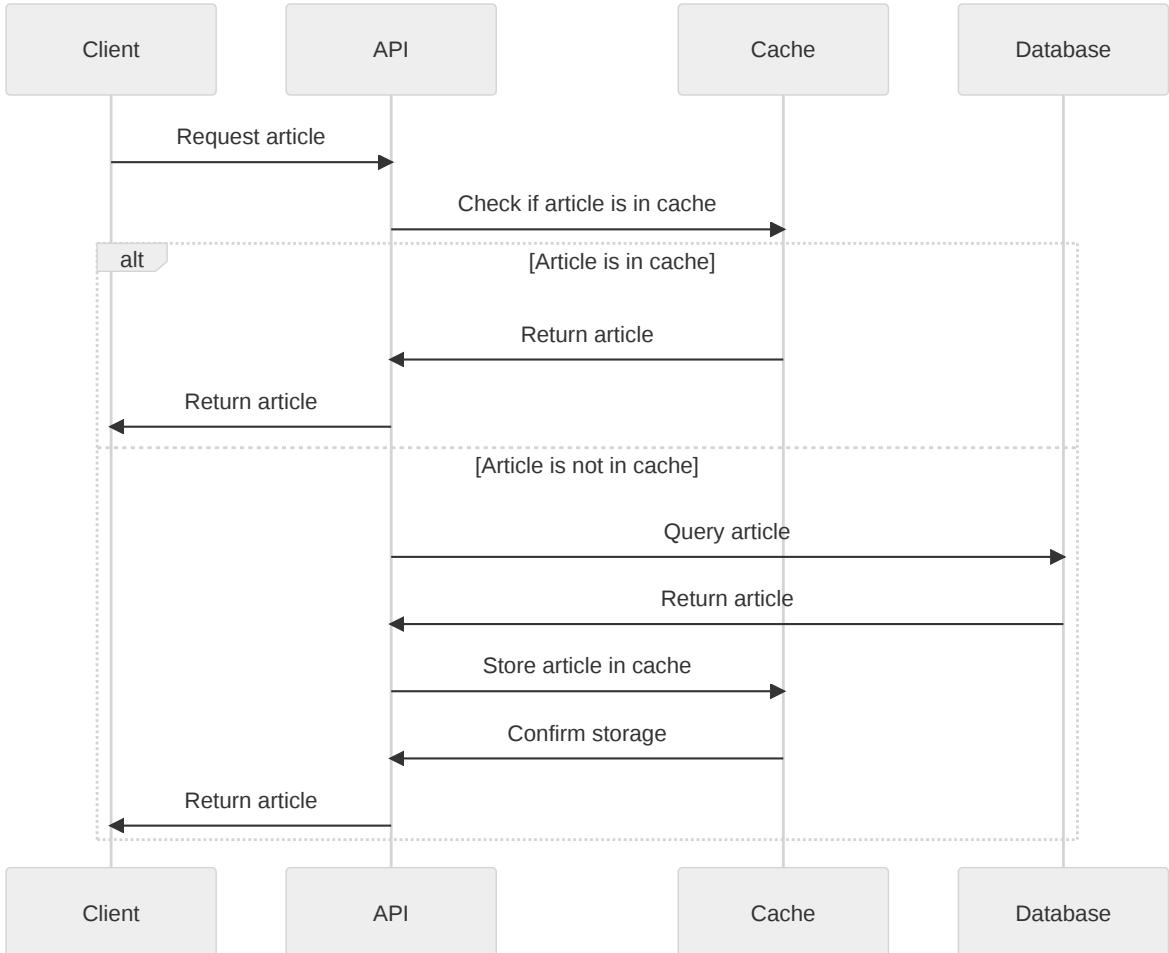
- Caching is a technique used to store copies of frequently accessed data in a temporary storage area, known as a cache.
- It aims to reduce the time and resources required to retrieve data, improving the overall performance and efficiency of a system.

Database and Caching

- In the context of databases, caching is particularly important for reducing latency and minimizing the load on the database by serving repeated requests for the same data without the need to query the database each time.
- Caching can be implemented at various levels, including the application layer, the database layer, and the network infrastructure.

Caching: Example

- Consider a news website that where some articles become popular and are accessed by a large number of users.
- Caching can be used to store the content of these articles in memory, reducing the need to query the database each time a user requests the same article.


Client
API
Cache
Database
Request article
Check if article is in cache
alt
[Article is in cache]
Return article
Return article
[Article is not in cache]
Query article
Return article
Store article in cache
Confirm storage
Return article
Client
API
Cache
Database

Caching: Activity

- Consider the case study of an e-commerce platform, where might caching be used to improve performance and reduce the load on the database?

Caching: Activity Solution

- In an e-commerce platform, caching can be used to store frequently accessed product information.
- This reduces the number of database queries required when users browse the site, leading to faster page load times and a better user experience.

Common Caching Strategies

- **LRU (Least Recently Used):** Discards the least recently accessed items first.
- **FIFO (First In First Out):** Discards the oldest cache items first.
- **LFU (Least Frequently Used):** Discards the least often accessed items first.

Caching & Key-Value Databases

- Caching is closely related to key-value databases, which are designed for storing and retrieving key-value pairs.
- Key-value databases are often used as caching solutions due to their high performance and simplicity.

Caching vs. Database Replication

- **Caching**: Stores copies of frequently accessed data in a temporary storage area to reduce latency and minimize database load.
- **Database Replication**: Involves creating and maintaining multiple copies of the database to improve fault tolerance, availability, and read scalability.

When Consider Caching

- **Rapid Data Retrieval is Essential:** To significantly improve data access times compared to querying the main database.
- **Minimizing Database Load is Crucial:** To maintain optimal performance and longevity of the main database by reducing the number of queries made to it.
- **Operational Cost Efficiency is a Goal:** When optimizing database operations to reduce resource usage and operational costs is essential.

Caching vs. Replication: Activity

Decide between caching and database replication for the following scenarios:

- University database storing student records, course enrollments, and grades.
- The system experiences high read loads at the beginning of each semester and whenever results are announced, but writes are relatively infrequent.

Caching vs. Replication: Activity Solution

- **Caching:** To handle high read loads efficiently during peak times by storing frequently accessed data in memory.
- **Replication:** Not necessary as writes are infrequent, and the primary concern is read scalability and performance during peak times.

Caching: Providers

- **Redis**: An in-memory data structure store, used as a database, cache, and message broker. It supports various data structures such as strings, hashes, lists, sets, and more.
- **Memcached**: A high-performance, distributed memory caching system designed to speed up dynamic web applications by reducing database load.

Google Cloud Memorystore, **Amazon ElastiCache**, **Microsoft Azure Cache for Redis** provide fully managed in-memory caching service compatible with Redis and Memcached, providing scalable and secure caching solutions.

Object Storage: What & Why…

- Object Storage is a data storage architecture that manages data as objects, unlike traditional file or block storage systems.
- It is designed for scalability, durability, and accessibility, making it ideal for storing large amounts of unstructured data like images, videos, and backups.
- Object Storage is often used in cloud environments due to its ability to handle massive amounts of data and provide easy access over the internet.

Object Storage vs. Database

- Object Storage and databases serve different purposes: Object Storage is optimized for storing and retrieving large files, while databases are designed for structured data and complex queries.
- Object Storage provides a simple interface for storing and retrieving data, whereas databases offer more advanced features like indexing, transactions, and relationships.
- Databases are typically used for transactional data and real-time queries, while Object Storage is better suited for archival, backup, and static content.

Object Storage: Example

- Consider a media streaming service that stores and delivers movies and TV shows to users.
- Object Storage can be used to store the video files, allowing for efficient retrieval and distribution of content to a large number of users.

Object Storage: Activity

- Consider the case study of an e-commerce platform, where might object storage be used to improve performance and reduce the load on the database?

Object Storage: Activity Solution

- In an e-commerce platform, object storage can be used to store product images, videos, and other media assets.
- This offloads the storage and delivery of these assets from the database, reducing the load on the database and improving the overall performance of the platform.
- The database will still store the metadata and other structured information about the products, while the media assets are stored in object storage.

Object vs Block vs File Storage

- **Object Storage**: Stores data as objects, each with a unique identifier, metadata, and data. Ideal for unstructured data and large files.
- **Block Storage**: Stores data in fixed-sized blocks, typically used for operating systems, databases, and virtual machines.
- **File Storage**: Stores data in a hierarchical structure with directories and files, suitable for file sharing and network-attached storage.

When Consider Object Storage

- **Scalability is a Priority:** When the system needs to store and manage large volumes of unstructured data, such as media files, backups, and logs.
- **Cost-Effective Storage is Essential:** When the system needs a cost-effective solution for storing and accessing large amounts of data over the internet.
- **High Durability and Availability are Required:** When the system needs a highly durable and available storage solution for critical data and assets.
- **Cold Data Storage is Needed:** When the system needs to store infrequently accessed data for archival or backup purposes.

Object Storage: Providers

- **Amazon S3 (Simple Storage Service)**: A scalable, high-speed, and low-cost web-based cloud storage service from AWS.
- **Google Cloud Storage**: A unified object storage solution from Google Cloud for developers and enterprises, offering high durability and availability.
- **Microsoft Azure Blob Storage**: A scalable and secure object storage service from Azure for storing large amounts of unstructured data.

Big Data: What & Why…

- Big Data refers to the vast volumes of structured, semi-structured, and unstructured data generated from various sources like social media, sensors, and transactions.
- The importance of Big Data lies in its potential for insights, decision-making, and innovation when properly analyzed and utilized.

Data Lake: What & Why…

- A Data Lake is a centralized repository that allows for the storage of large amounts of raw data in its native format.
- It is designed for flexibility, scalability, and the ability to store diverse data types, making it ideal for Big Data analytics and machine learning.

Data Warehouse: What & Why…

- A Data Warehouse is a structured data repository designed for query and analysis, optimized for fast retrieval of structured data.
- It is used for consolidating data from various sources, providing a unified view for reporting and business intelligence.

Database vs. Data Lake vs. Data Warehouse

- **Database**: Optimized for transactional processing and efficient data management for structured data.
- **Data Lake**: Focuses on storing raw, unstructured data at scale, with flexibility for various data types and analytics.
- **Data Warehouse**: Specialized for structured data analysis, supporting complex queries and business intelligence.

Big Data: Example

- A social media platform generates vast amounts of data from user interactions, posts, and activities.
- This Big Data can be analyzed to understand user behavior, trends, and preferences, enabling targeted advertising and content recommendations.

Big Data: Activity

- Consider a healthcare provider with large volumes of patient data, including medical records, lab results, and imaging.
- How can Big Data technologies be applied to improve patient care and operational efficiency?

Big Data: Activity Solution

- Implement a Data Lake to store and manage diverse healthcare data, enabling advanced analytics and research.
- Use a Data Warehouse to consolidate and analyze structured data for reporting and decision support.
- Apply Big Data analytics to identify patterns, predict outcomes, and personalize treatment plans.

Big Data: Providers

- **Amazon Web Services (AWS)**: Offers a comprehensive suite of Big Data services, including Amazon S3 for data lakes and Amazon Redshift for data warehousing.
- **Google Cloud Platform (GCP)**: Provides Big Data solutions like Google Cloud Storage for data lakes and BigQuery for data warehousing.
- **Microsoft Azure**: Offers Azure Data Lake for storing large datasets and Azure Synapse Analytics for data warehousing and analytics.

Conclusion

- Caching, Object Storage, and Big Data technologies play crucial roles in modern software system design.
- They enable high-performance, scalable, and cost-effective solutions for managing data, content, and analytics.
- Understanding these technologies and their applications is essential for building robust and efficient software systems.