Báo cáo tuần 2

# Tìm hiểu Caching (Redis và Aerospike)

# &

# Hệ sinh thái Big Data (HDFS, Hbase, Apacha Kafka và Apache Spark)


Người phụ trách: Anh Ngô Văn Vĩ

Người thực hiện: Nguyễn Đình Hiếu

# Mục lục

**Phần I: Caching (Redis và Aerospike)** ..... 3

**1.1 Redis** ..... 3

**1.2 Aerospike** ..... 4

**1.3 So sánh Redis và Aerospike** ..... 6

**Phần 2: Hệ sinh thái Big Data** ..... 7

**2.1 HDFS** ..... 7

**2.2 Hbase** ..... 8

**2.3 Apache Kafka** ..... 8

**2.4 Apache Spark** ..... 9

**Nguồn tham khảo:** ..... 11

# Phần I: Caching (Redis và Aerospike)

## 1.1 Redis

Redis là cụm từ viết tắt của Remote Dictionary Server. Được biết đến là một hệ quản trị cơ sở dữ liệu NoSQL mã nguồn mở thiết kế để lưu trữ và quản lý dữ liệu dưới dạng key-value trên bộ nhớ đệm (in-memory).

Redis là mã nguồn mở được phát triển bởi Salvatore Sanfilippo vào năm 2009 và nhanh chóng trở thành một trong những công cụ phổ biến nhất trong lĩnh vực quản lý dữ liệu và tối ưu hóa hiệu suất ứng dụng.

**Ưu điểm:**

***- Tốc độ truy cập dữ liệu cực nhanh:*** Redis hoạt động dựa trên bộ nhớ trong (in-memory), cho phép truy cập dữ liệu với tốc độ cao hơn nhiều so với các hệ thống lưu trữ dữ liệu truyền thống.

***- Khả năng mở rộng linh hoạt:*** Redis có thể dễ dàng mở rộng bằng cách thêm các máy chủ mới vào hệ thống, đáp ứng nhu cầu lưu trữ và truy cập dữ liệu ngày càng tăng.

***- Tính đơn giản:*** Cấu trúc dữ liệu đơn giản và dễ sử dụng cũng là yếu tố quan trọng khi nhắc về ưu điểm của Redis là gì. Theo đó, đặc trưng này góp phần giúp các lập trình viên dễ dàng tích hợp Redis vào các ứng dụng của mình.

***- Mã nguồn mở:*** Redis là một hệ thống mã nguồn mở, miễn phí và dễ dàng tích hợp với các ứng dụng khác.

***- Hỗ trợ nhiều cấu trúc dữ liệu:*** Redis hỗ trợ nhiều cấu trúc dữ liệu khác nhau, bao gồm string, hash, list, set, sorted set,... Trên cơ sở đó, chúng có thể đáp ứng nhu cầu lưu trữ đa dạng của các ứng dụng.

**Nhược điểm:**

***- Lưu trữ dữ liệu tạm thời:*** Redis lưu trữ dữ liệu trong bộ nhớ trong, do đó dữ liệu có thể bị mất nếu máy chủ bị lỗi hoặc khởi động lại. Để khắc phục vấn đề này, cần sử dụng các cơ chế sao lưu dữ liệu như RDB (Redis Database) hoặc AOF (Append-only file).

***- Khả năng truy vấn dữ liệu hạn chế:*** Redis không hỗ trợ các truy vấn phức tạp như các hệ thống cơ sở dữ liệu truyền thống.

***- Không phù hợp cho dữ liệu lớn:*** Redis cũng được cho là không phù hợp cho việc lưu trữ dữ liệu với dung lượng lớn vì có thể ảnh hưởng đến hiệu suất hệ thống.

***- Yêu cầu kiến thức chuyên môn:*** Việc triển khai và quản lý Redis hiệu quả đòi hỏi kiến thức chuyên môn về hệ thống và vận hành.

Nhờ đặc điểm giúp giảm thời gian truy vấn, nên Redis có tác dụng rất mạnh mẽ trong việc sử dụng làm cache cho các ứng dụng web với các kĩ thuật caching phổ biến như: Cache-Aside, Read-Through. Write-Through, Write-Back và Cache Invalidation để đảm bảo dữ liệu luôn đúng và không bị cũ so với dữ liệu thực.

## 1.2 Aerospike

Aerospike là cơ sở dữ liệu phân tán NoSQL hiệu năng cao mã nguồn mở. Nó hỗ trợ mô hình lưu trữ key-value và document. Được thiết kế để truy xuất dữ liệu cực nhanh (sub-millisecond), mở rộng theo chiều ngang, và ổn định khi xử lý dữ liệu thời gian thực (real-time) kể cả với hàng triệu request mỗi giây.

**Kiến trúc Aerospike gồm 3 lớp:**

**- Client layer:** Aerospike client là lớp tự động theo dõi cluster liên tục xem nên biết được truy cập dữ liệu từ vị trí nào trong một cụm cluster, tự động detect các transaction lỗi, và query lại ở một máy khác chứa cùng dữ liệu.

**- Clustering and Data Distribution Layer:** Lớp này quản lý các giao tiếp cluster, tự động fail-over, replication, cross-datacenter replication (XDR), và cân bằng tải lại và di chuyển dữ liệu thông minh.

**- Data Storage Layer:** Lớp này chịu trách nhiệm lưu trữ dữ liệu trong DRAM và Flash cho phản hồi nhanh.

Aerospike lưu trữ key-value với mô hình dữ liệu schemaless. Dữ liệu được đẩy vào trong các hộp chứa, namespace policy. Namespace chia dữ liệu thành các tập (set) tương tự như table trong RDBMS và các bản ghi (record) tương ứng với row trong RDBMS. Mỗi record được đánh một index key duy nhất trong tập set và một hoặc nhiều hơn các tên bins tương ứng với column trong RDBMS.

**Ưu điểm:**

***- Hiệu năng cực cao và độ trễ rất thấp****:* Aerospike được thiết kế để xử lý hàng triệu truy vấn mỗi giây với độ trễ dưới 1 mili giây. Nhờ kiến trúc kết hợp giữa RAM và SSD (Hybrid Memory), nó có thể truy xuất dữ liệu nhanh như khi chỉ dùng RAM nhưng vẫn tiết kiệm tài nguyên, phù hợp cho các ứng dụng real-time như gợi ý sản phẩm, phân phối quảng cáo, hoặc chống gian lận.

***- Khả năng mở rộng ngang linh hoạt****:* Aerospike hỗ trợ mở rộng dễ dàng bằng cách thêm node vào cụm mà không làm gián đoạn hệ thống.

Điều này giúp hệ thống có thể phục vụ lượng lớn người dùng hoặc dữ liệu tăng trưởng liên tục mà vẫn giữ hiệu năng ổn định.

- ***Đảm bảo tính bền vững và độ tin cậy cao**:* Dữ liệu trong Aerospike không chỉ được lưu tạm thời trong RAM mà còn có thể được ghi xuống SSD, giúp đảm bảo tính bền vững. Ngoài ra, cơ chế sao chép và phục hồi tự động giúp hệ thống hoạt động ổn định ngay cả khi một số node bị lỗi.

- ***Chi phí vận hành tối ưu hơn so với cache thuần RAM**:* Thay vì yêu cầu toàn bộ dữ liệu phải nằm trong RAM như Redis, Aerospike có thể lưu phần lớn dữ liệu trên SSD tốc độ cao mà vẫn giữ hiệu năng tốt. Điều này làm giảm đáng kể chi phí phần cứng, đặc biệt trong các hệ thống cần lưu trữ hàng terabyte dữ liệu.

## 1.3 So sánh Redis và Aerospike

Ban đầu việc triển khai với Redis khá đơn giản và dễ sử dụng. Nhưng khi mọi thứ thay đổi quá nhanh, khối lượng dữ liệu và khối lượng công việc liên tục tăng.

**Total cost of ownership (TCO):**

- Redis: Việc mở rộng quy mô sẽ tốn thêm DRAM và DRAM thì đắt, hơn nữa việc tăng số lượng cluster cũng chẳng dễ dàng.

- Aerospike: Do cơ chế lưu trữ Hybrid nên ta có thể kết hợp lưu trữ nhiều nơi việc này giúp tối ưu hiệu suất và tiết kiệm chi phí.

**Tính nhất quán dữ liệu:**

- Redis: Redis không vượt qua bài kiểm tra Jepsen (tính nhất quán). Redis chỉ có tính nhất quán "cuối cùng", dẫn đến việc đọc dữ liệu cũ hoặc mất dữ liệu (trong trường hợp tràn data).

- Aerospike: vượt qua bài kiểm tra Jepsen. Aerospike cũng đảm bảo rằng dữ liệu sẽ được đồng nhất giữa các node trước khi giao dịch xảy ra.

**Khả năng mở rộng (Scalability):**

- Redis: Việc mở rộng cụm Redis cần cấu hình thủ công (sharding), việc tự động phân vùng dữ liệu và cân bằng tải giữa các node không linh hoạt.

- Aerospike: Hỗ trợ scale-out (mở rộng ngang) rất tốt, tự động phân phối dữ liệu và đồng bộ giữa các node mà không cần can thiệp sâu từ phía devops.

# Phần 2: Hệ sinh thái Big Data

## 2.1 HDFS

HDFS (Hadoop Distributed File System) là hệ thống tệp phân tán trong Hadoop, được thiết kế để lưu trữ dữ liệu lớn (Big Data) trên nhiều máy khác nhau, với độ tin cậy và khả năng mở rộng cao.

**- Phân tán dữ liệu**
Dữ liệu được chia nhỏ thành các block (mặc định 128MB) và lưu trên nhiều máy trong cụm (cluster).

**- Chịu lỗi tốt**
Mỗi block được sao chép (thường 3 bản) để đảm bảo vẫn hoạt động khi một node bị lỗi.

**- Thiết kế cho dữ liệu lớn, truy cập theo lô**
HDFS tối ưu cho việc đọc/ghi dữ liệu lớn (GB–TB), không phù hợp với truy cập nhỏ và ngẫu nhiên.

**- Là nền tảng lưu trữ cho Hadoop ecosystem**
Các công cụ như MapReduce, Hive, Spark, HBase… đều có thể sử dụng HDFS để lưu và đọc dữ liệu.

## 2.2 Hbase

HBase là một cơ sở dữ liệu NoSQL dạng cột (column-oriented), được xây dựng trên HDFS, dùng để lưu trữ và truy xuất dữ liệu lớn theo thời gian thực.

**- Dữ liệu dạng bảng, nhưng phi quan hệ**

Dữ liệu được tổ chức thành bảng, dòng, cột — nhưng không yêu cầu schema chặt chẽ như SQL. Mỗi ô có thể lưu nhiều phiên bản theo thời gian.

**- Xây dựng trên HDFS**
HBase tận dụng khả năng lưu trữ lớn và chịu lỗi của HDFS, nhưng bổ sung khả năng *truy cập ngẫu nhiên (random read/write)* mà HDFS không có.

**- Truy xuất dữ liệu theo thời gian thực**
HBase cho phép ghi và đọc nhanh, phù hợp với ứng dụng cần xử lý dữ liệu lớn nhưng vẫn yêu cầu thời gian thực (real-time).

**- Phù hợp cho Big Data có cấu trúc**
HBase được dùng trong các hệ thống như log phân tán, bảng điểm người dùng, phân tích hành vi, gợi ý sản phẩm…

## 2.3 Apache Kafka

Apache Kafka là một nền tảng xử lý luồng dữ liệu (streaming platform), chuyên dùng để truyền tải, lưu trữ và xử lý dữ liệu theo thời gian thực giữa các hệ thống.

**- Mô hình publish-subscribe (pub/sub)**
Kafka cho phép một hệ thống (producer) gửi dữ liệu vào topic, và nhiều hệ thống khác (consumer) có thể đăng ký nhận dữ liệu từ topic đó.

**- Xử lý dữ liệu theo thời gian thực**
Kafka được tối ưu để xử lý các luồng dữ liệu liên tục, như log hệ thống, hành vi người dùng, clickstream, cảm biến IoT, v.v.

**- Tốc độ cao, độ bền dữ liệu tốt**
Kafka có khả năng ghi và đọc hàng triệu sự kiện mỗi giây, với dữ liệu được ghi xuống đĩa và có cơ chế replication để chống mất dữ liệu.

**- Dễ mở rộng và tích hợp**
Kafka có thể mở rộng theo chiều ngang và dễ tích hợp với các công cụ như Spark, Flink, Hadoop, Elasticsearch, và hệ thống microservices.

## 2.4 Apache Spark

Apache Spark là một nền tảng xử lý dữ liệu lớn (Big Data) tốc độ cao, hỗ trợ cả xử lý theo lô (batch) và theo luồng (stream).

**- Xử lý dữ liệu rất nhanh**

Spark sử dụng cơ chế in-memory computation (tính toán trên RAM), giúp nhanh hơn nhiều so với Hadoop MapReduce truyền thống.

**- Hỗ trợ nhiều kiểu xử lý dữ liệu**

Batch (theo lô), Streaming (thời gian thực), Machine learning (MLlib), SQL (Spark SQL), Đồ thị (GraphX)

**- Tương thích với nhiều nguồn dữ liệu**

Spark đọc/ghi dữ liệu từ HDFS, Cassandra, HBase, Hive, Kafka, JDBC, S3, Parquet,... rất linh hoạt.

**- Mở rộng dễ dàng và chạy phân tán**

Spark có thể chạy trên một cụm máy (cluster), dùng YARN, Mesos, Kubernetes, hoặc standalone — dễ mở rộng theo chiều ngang.

## Nguồn tham khảo:

[1] *Redis là gì?*

https://viettelidc.com.vn/tin-tuc/redis-la-gi-tat-tan-tat-uu-nhuoc-diem-va-ung-dung

[2] *Sử dụng Redis làm cache để tăng tốc độ truy vấn*

https://viblo.asia/p/su-dung-redis-lam-cache-de-tang-toc-do-truy-van-GrLZD0dwZk0

[3] *Giới thiệu về Aerospike DB*

https://ailab.siu.edu.vn/article/9/gioi-thieu-ve-aerospike-db