**Report Week 3**

Contents

[**1.** **Caching là gì** 1](#_Toc202520474)

[**1.1** **Khái niệm cơ bản** 1](#_Toc202520475)

[**1.2** **Các chiến lược caching cơ bản** 1](#_Toc202520476)

[**1.3** **Kết luận** 1](#_Toc202520477)

[**2.** **So sánh giữa redis và aerospike** 2](#_Toc202520478)

[**2.1** **Bảng so sánh:** 2](#_Toc202520479)

[**2.2** **Kết luận:** 2](#_Toc202520480)

[**3.** **Apache Hbase** 3](#_Toc202520481)

[**3.1** **Apache Hbase** 3](#_Toc202520482)

[**3.2** **Kiến trúc của Hbase** 3](#_Toc202520483)

[**3.3** **Mô hình dữ liệu** 3](#_Toc202520484)

[**3.4** **Đặc điểm nổi bật** 3](#_Toc202520485)

[**3.5** **Kết luận** 4](#_Toc202520486)

[**4.** **Apache Kafla** 4](#_Toc202520487)

[**4.1** **Apache Kafla là gì** 4](#_Toc202520488)

[**4.2** **Kiến trúc hệ thống** 4](#_Toc202520489)

[**4.3** **Các đặc điểm nổi bật** 5](#_Toc202520490)

[**4.4** **Mô hình dữ liệu** 5](#_Toc202520491)

[**4.5** **Kết luận** 5](#_Toc202520492)

1. **Caching là gì**
   1. **Khái niệm cơ bản**

* Trong điện toán, bộ nhớ đệm (cache) là một phần cứng hoặc phần mềm lưu trữ dữ liệu tại các điểm truy cập nhanh hơn. Dữ liệu lưu trữ trong bộ nhớ đệm có thể là kết quả của tính toán trước đó hoặc bản sao dữ liệu được lưu trữ ở nơi khác
* Caching là hành động thực hiện lưu trữ và quản lý dữ liệu trong cache để tối ưu hóa quá trình truy cập dữ liệu trong tương lai. Caching bao gồm việc xác định dữ liệu nào nên được lưu trữ, cũng như quản lý việc lưu trữ và cập nhật dữ liệu đó trong cache để đảm bảo dữ liệu luôn được cập nhật và phản ánh chính xác trạng thái hiện tại
* Caching có thể được áp dụng ở nhiều cấp độ khác nhau, bao gồm phần cứng, hệ điều hành, ứng dụng web và cơ sở dữ liệu.
  1. **Các chiến lược caching cơ bản**

Chiến lược caching được chia thành 2 loại chính:

* Chiến lược đọc dữ liệu:
* Cache aside(lazy loading): Ứng dụng tự quyết địng lúc nào nên dọc/ghi từ cache
* Read through: Cache sẽ là trung gian giữa ứng dụng và database
* Chiến lược ghi dữ liệu:
* Write through: Ghi cả vào cache và database cùng lúc
* Write back: Ghi vào cache trước, database sau
* Write around: Bỏ qua cache khi ghi, khi đọc mới đưa vào cache
  1. **Kết luận**
* Ưu điểm:
* Nâng cao performance: cải thiện hiểu suất và tốc đổ tải dữ liệu ứng dụng
* Giảm connection và truy vấn cho cơ sở dữ liệu: có những dữ liệu tĩnh rất ít thay đổi trong cở sở dữ liệu nên khi giữ dữ liệu ở đâu đó sẽ không cần phải mở kết nối và truy vấn dữ liệu này một cách thường xuyên
* Cải thiện khi thời gian xử lý tăng đột biến:
* Giảm tải cho Backend
* Nhược điểm:
* Vấn đề về tính thống nhất dữ liệu: khi có bất cứ dữ liệu nào vào bộ nhớ cache, sẽ có một bản sao được tạo ra => hai bản sao của cùng một dữ liệu có thể gây bất động bộ dữ liệu
* Lựa chọn dữ liệu được lưu vào cache:
* Xử lý dữ liệu cache khi bị mất:

1. **So sánh giữa redis và aerospike**
   1. **Bảng so sánh:**

|  |  |  |
| --- | --- | --- |
| **Tiêu chí** | **Redis** | **Aerospike** |
| **Nguồn gốc** | Redis labs | Aerospike Inc |
| **Hệ điều hành** | Windows, Os X, Linux, BSD | Chỉ dành cho Linux |
| **Kiểu CSDL** | In-memory key-value store | Hybric memory-disk key-value store |
| **Hiệu suất** | Rất nhanh(in-memory) | Rất nhanh, tối ưu cho low-latency |
| **Độ bền dữ liệu(Durability)** | Tùy chọn(RDB, AOF) | Có thể cấu hình lưu disk mạnh mẽ hơn |
| **Khả năng mở rộng** | Cluster mode(Redis Cluster) | Horizonal scalling mạnh hơn |
| **Kiểu dữ liệu** | Đa dang: string, set, hash, stream... | Chủ yếu là key-value + TTL |
| **Hỗ trợ TTL(expiration)** | Có | Có, rất hiệu quả |
| **Replication** | Có (Master/Replice) | Có,cấu hình mạnh hơn (syns/async) |
| **Use case phổ biến** | Cache, Pub/Sub, Session Store,  Counter | Cache, real-time analytics,  fraud detetion |
| **Ngôn ngữ hỗ trợ** | Python, Node.js, Java, Go…. | C, Java, Python, Go…. |
| **Open source** | Có (AGPL + Redis Stack có bản thương mại) | Có (Apache 2.0) |
| **Quản lý dữ liệu lớn** | Không tối ưu cho data size lớn | Tốt hơn nhờ thiết kế hybrid memory/disk |
| **Yêu cầu ram** | Tất cả dữ liệu phải nằm trong ram | Dữ liệu có thể vượt ram (dùng disk) |
| **Khả năng tự phục hồi** | Hạn chế, cần setup thêm | Có sẵn trong enterprise edition |
| **Persistence (Lưu trữ lâu dài)** | Tùy chọn (RDB snapshot hoặc AOF log) | Có write-to- disk, strong consistency |

* 1. **Kết luận:**
* Redis phù hợp với:
* Cache đơn giản, cần tốc độ
* Muốn thao tác linh hoạt (nhiều kiểu dữ liệu)
* Ram dư dả
* Hợp với các hệ thống web từ nhỏ đến vừa
* Aerospike phù hợp với:
* Dữ liệu lớn, cần mở rộng ngang
* Cần bền vững, ít mất dữ liệu
* Ram giới hạn, dùng disk hybrid
* Phù hợp các hệ thống enterprise hoặc realtime analytics

1. **Apache Hbase**
   1. **Apache Hbase**

Apache Hbase là một hệ thống cơ sở dữ liệu phân tán mã nguồn mở, được thiết kế để lưu trữ và xử lý dữ liệu theo mô hình NoSQL. Nó chạy trên nên tảng Apache Hadoop và sử dụng HDFS (Hadoop Distributed File System) để lưu trữ. Hbase lấy cảm hứng từ Bigtable – một hệ thống lưu trữ dữ liệu phi quan hệ của google.

* 1. **Kiến trúc của Hbase**

Hbase có kiến trúc theo mô hình Master-Slave, gồm 3 thành phần chính:

* HMaster: Quản lý metadata, điều phối các thao tác như chia tách vùng (region splitting) hay phân phôi lại các dữ liệu
* RegionServer: Chịu trách nhiệm lưu trữ và xử lý dữ liệu. Mỗi RegionServer quản lý nhiều vùng(regions)
* Zookeeper: Hệ thống điều phối , giúp phát hiện lỗi và theo dõi các trạng thái node
  1. **Mô hình dữ liệu**

Dữ liệu trong Hbase được lưu theo cấu trúc bảng (table), nhưng không như RDBMS truyền thống gồm:

* Row key: Định danh cho mỗi hàng (duy nhất)
* Column family: Nhóm các cột liên quan. Mỗi bảng có nhiều column family
* Column qualifier: Tên cột cụ thể trong column family
* Timestamp: Mỗi ô dữ liệu (cell) có thể chứa nhiều phiên bản được phân biệt theo timestamp
  1. **Đặc điểm nổi bật**
* Phân tán và mở rộng ngang: Có thể mở rộng bằng cách thêm các RegionServer
* Lưu trữ dữ liệu lớn: Hỗ trợ hàng tỷ dòng và hàng triệu cột
* Hỗ trợ truy cập theo thời gian thực (real-time), khác với Mapreduce batch trong Hadoop
* Tích hợp tốt với Hadoop: có thể kết hợp với MapReduce, Hive, Pig…
  1. **Kết luận**

Apache HBase là một giải pháp mạnh mẽ cho việc lưu trữ và truy xuất dữ liệu lớn theo thời gian thực. Nó phù hợp với những hệ thống cần khả năng mở rộng linh hoạt, tốc độ cao và không yêu cầu nghiêm ngặt về cấu trúc dữ liệu. Tuy nhiên, nó cũng có một số hạn chế như thiếu hỗ trợ đầy đủ transaction hoặc truy vấn phức tạp.

1. **Apache Kafla**
   1. **Apache Kafla là gì**

Apache Kafla là một nền tảng streaming phân tán mã nguồn mở, được phát triển ban đâu bởi Linkedln và hiện thuộc về Apache Software Foundation. Kafla được thiết kế để xử lý dòng dữ liệu thời gian thực với hiệu suất cao, độ bền vững và khả năng mở rộng tốt

Kafla thường được sử dụng trong các hệ thống yêu cầu thu nhập, xử lý và phân phối dữ liệu liên tục như hệ thống phân tích log, giám sát, phân tích sự kiện (event streaming), hoặc làm trung gian truyền thông giữa các dịch vụ

* 1. **Kiến trúc hệ thống**

Kafla hoạt động theo mô hình pub-sub (publish – subscribe) gồm các thành phần chính:

**A diagram of a company

AI-generated content may be incorrect.**

* Producer: Gửi (publish) dũ liệu đến Kafla
* Broker: Là máy chủ Kafla lưu trữ dữ liệu. Một cluster kafla có nhiều broker
* Topic: Là danh mục phân loại dữ liệu. Mỗi topic chia thành nhiều partion để tăng hiệu năng và khả năng song song
* Consumer: Nhận (subscribe) dữ liệu từ kafla .
* Zookeeper: Quản lý Cluster Kafla (trong phiên bản Kafla mới, Zookeeper dần được loại bỏ)
  1. **Các đặc điểm nổi bật**
* Thông lượng cao (High throughput): Kafla có thể xử lý hàng triệu messenger mỗi giây
* Độ trễ thấp (Low latency): Phù hợp cho xử lý dữ liệu thời gian thực
* Bền vững (Durability): Dữ liệu được ghi vào các ổ đĩa và có thể được sao lưu giữa các broker
* Mở rộng ngang (Scalable): Thêm broker và partition dễ dàng để mở rộng
* Fault-tolenrant: Kafla hỗ trợ tự động phục hồi khi có lỗi node
  1. **Mô hình dữ liệu**
* Mỗi topic chứa các bản tin (messenges), được ghi tuần tự vào partition
* Mỗi bản tin có định dạng đơn giản gồm:
* Key (tùy chọn)
* Value (nội dung)
* Timestamp
* Messenge trong Kafla là bất biến, được lưu giữ trong khoảng thời gian cấu hình(ví dụ 1 tuần)
  1. **Kết luận**

Apache Kafka là một công cụ mạnh mẽ để xử lý dữ liệu thời gian thực với độ tin cậy, tốc độ cao và khả năng mở rộng linh hoạt. Nó đặc biệt phù hợp với những hệ thống lớn cần xử lý và truyền dữ liệu liên tục như giám sát hệ thống, xử lý log, hoặc tích hợp hệ thống.