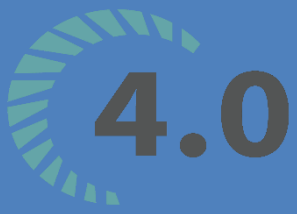
BỘ MÔN HỆ THỐNG THÔNG TIN – KHOA CÔNG NGHỆ THÔNG TIN

ĐẠI HỌC KHOA HỌC TỰ NHIÊN THÀNH PHỐ HỒ CHÍ MINH, ĐẠI HỌC QUỐC GIA TP HCM

MÔN HỆ QUẢN TRỊ CƠ SỞ DỮ LIỆU



Sinh viên thực hiện: 19127366 – Long Mỹ Du (Nhóm trưởng)

19127304 – Trần Khải Trúc

19127649 – Tô Thanh Tuấn

GV phụ trách: Hồ Thị Hoàng Vy

Tiết Gia Hồng

ĐỒ ÁN MÔN HỌC: DALT-01 - HỆ QUẢN TRỊ CƠ SỞ DỮ LIỆU

HỌC KỲ I – NĂM HỌC 2021-2022

KHOA CÔNG NGHỆ THÔNG TIN
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
227 Nguyễn Văn Cừ, Phường 4, Quận 5, TP.HCM
Điện Thoại: (08) 38.354.266 - Fax:(08) 38.350.096

BẢNG THÔNG TIN CHI TIẾT NHÓM

Mã nhóm:	Nhóm 10	
Số lượng:	**3**	
Github đồ án:	https://github.com/LongMyDu/HQTCSDL-LT-Group10-Aerospike.git	
MSSV	**Họ tên**	**Email**
19127366	Long Mỹ Du	lmdu19@clc.fitus.edu.vn
19127304	Trần Khải Trúc	tktruc19@clc.fitus.edu.vn
19127649	Tô Thanh Tuấn	tttuan19@clc.fitus.edu.vn

Đánh giá mức độ hoàn thành:

DALT	Báo cáo (5%)	Kiến trúc, tính năng (15%)	Mô hình (15%)	Cài đặt HQT (20%)	DD hệ thống vs HQT (5%)	Cài đặt mô hình dl (10%)	Khai thác truy vấn (25%)	Đánh giá (5%)	Tổng (100%)	Tỷ lệ sv tự đánh giá
Long Mỹ Du	1	5	5	7	1	4	9	2	34	34
Trần Khải Trúc	2	5	5	7	2	3	8	1	33	33
Tô Thanh Tuấn	2	5	5	6	2	3	8	2	33	33

Bảng phân công & đánh giá hoàn thành công việc			
Công việc thực hiện	**Người thực hiện**	**Mức độ hoàn thành**	**Đánh giá của nhóm**
- Tìm hiểu cấu trúc tổng quan, các khái niệm và đặc điểm của mô hình dữ liệu - Cài đặt Aerospike server & client - Thiết kế và cài đặt mô hình dữ liệu	Long Mỹ Du	100%	10/10
- Tìm hiểu đặc điểm của HQT, tính năng giao tác, cơ chế phục hồi dữ liệu - Phân tích đặc điểm hệ thống nên sử dụng Aerospike	Trần Khải Trúc	100%	10/10
- Tìm hiểu cơ chế quản lý giao tác đồng thời, cơ chế bảo mật. - Đưa ra kịch bản truy vấn - Lập bảng đánh giá tính năng	Tô Thanh Tuấn	100%	10/10

YÊU CẦU ĐỒ ÁN- BÀI TẬP

Loại bài tập	**☑ Lý thuyết ☐ Thực hành ☑ Đồ án ☐ Bài tập**
Ngày bắt đầu	**30/10/2021**
Ngày kết thúc	**25/12/2021**

A. Yêu cầu của đồ án

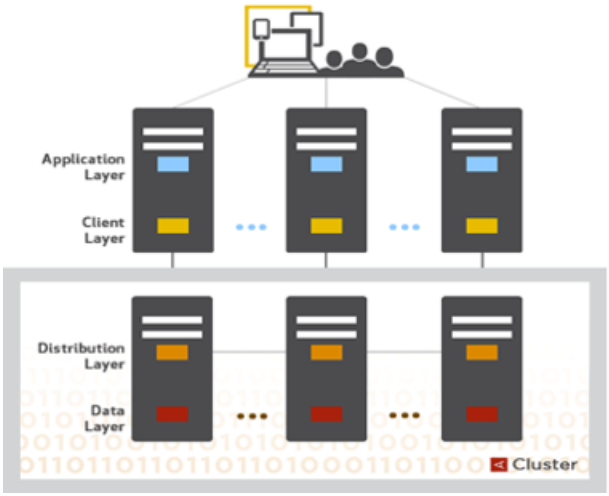
1. Tìm hiểu kiến trúc, đặc điểm, tính năng HQT CSDL Aerospike
2. Khái niệm và đặc điểm của mô hình lưu trữ dữ liệu được sử dụng trong HQT
3. Cách triển khai/ cài đặt HQT CSDL Aerospike
4. Phân tích đặc điểm hệ thống nên áp dụng Aerospike
5. Mô tả và nêu đặc điểm phù hợp của hệ thống với Aerospike
6. Thiết kế và cài đặt mô hình dữ liệu của hệ thống.
7. Khai thác truy vấn và các tính năng đã tìm hiểu.
8. Đánh giá các tính năng của Aerospike

B.Kết quả

Mục lục

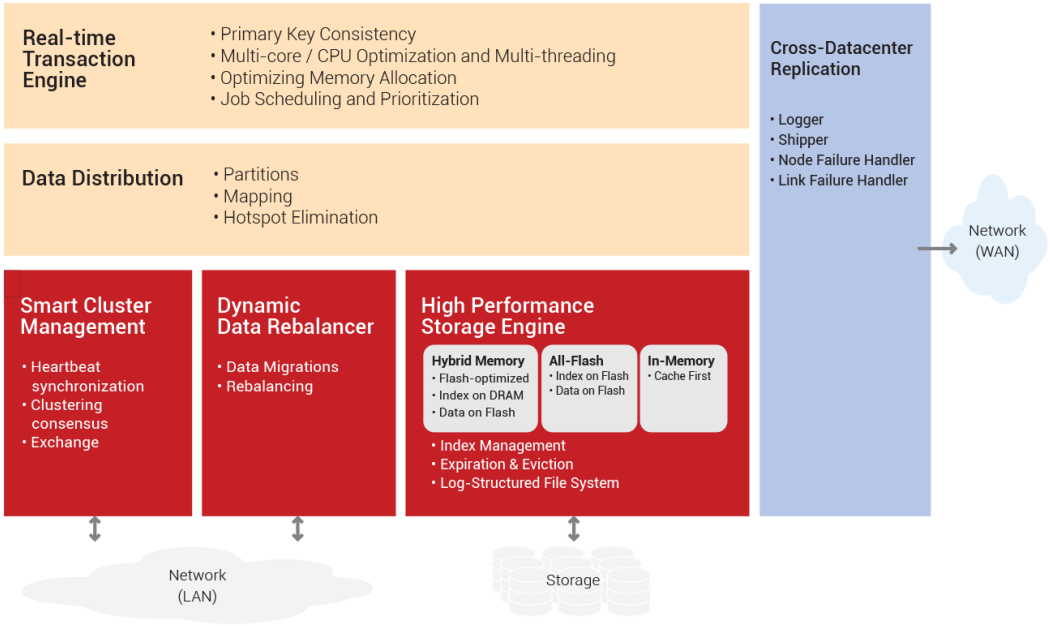
I. Kiến trúc tổng quan

1. Kiến trúc: client-server:



Architecture Overview (aerospike.com)

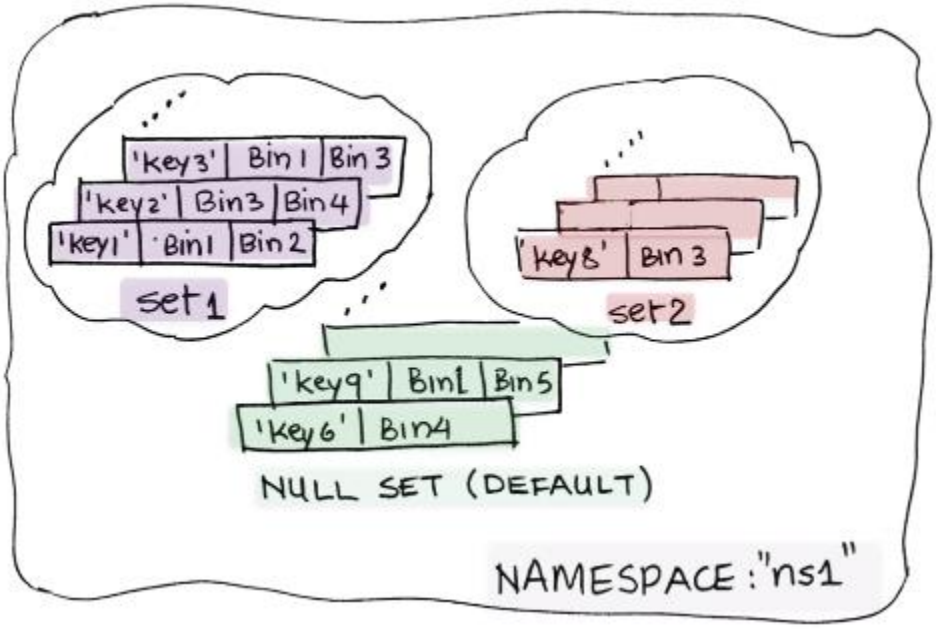
- Các layer bên client: bao gồm các thư viện mã nguồn mở, APIs, track nodes (các nút để theo dõi dữ liệu nằm ở đâu trong cụm).
- Các layer trong server:



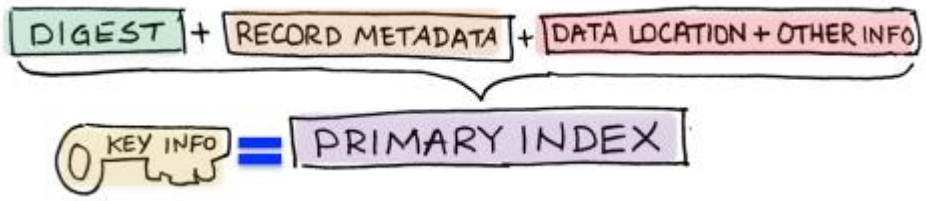
- Data distribution: gồm có các module:
  - Cluster Management: Dùng thuật toán Paxos-based gossip-voting process để theo dõi dữ liệu, xác định những node nào là 1 phần của cluster; đồng bộ hóa "heartbeat" để quản lý việc giao tiếp giữa các cluster.
  - Data Rebalancer: Quản lý việc "di cư" dữ liệu và tái cân bằng dữ liệu.
  - Transaciton Processing Module: Quản lý việc đồng bộ hóa các bản sao dữ liệu.
  - Resolution of Duplicate Data: Xử lý mâu thuẫn giữa các bản sao.
  - Ngoài ra, có thể cài đặt thêm Cross-Datacenter Replication để quản lý các bản sao dữ liệu không đồng bộ từ nhiều hệ thống khác nhau.
- Data Storage: lưu trữ dữ liệu.

2. Mô hình dữ liệu (Data model):

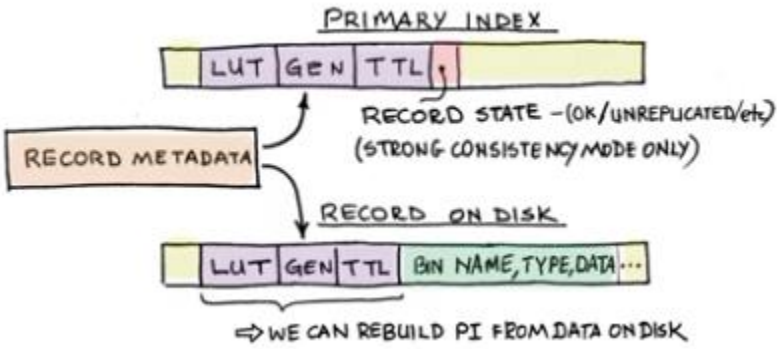
- Không cần định nghĩa trước lược đồ cơ sở dữ liệu (schemaless).
- Dữ liệu được nhóm với nhau qua namespaces. (Có thể hiểu namespace tương ứng với database trong mô hình dữ liệu quan hệ)
- Trong namespace, dữ liệu được chia thành các sets, trong một set sẽ gồm nhiều records. (Có thể hiểu đơn giản set tương ứng với bảng dữ liệu, và record tương ứng với các dòng trong bảng dữ liệu trong mô hình dữ liệu quan hệ). Mặc định, các records sẽ được lưu trong null set.



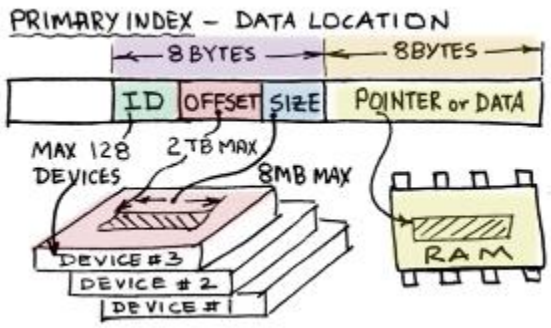
- Mỗi record gồm: Key và bins. Key là khóa duy nhất, đi kèm với khóa này là bins là các giá trị dữ liệu cần lưu trữ. Trong bins có thể bao gồm nhiều trường dữ liệu với các kiểu khác nhau: integers, doubles, strings, byte arrays, BLOBs, maps, lists, GeoJSON, HyperLogLog.
- Các record trong cùng set cũng có thể có cấu trúc bins khác nhau.
- **Khóa chính trong Aerospike:**
    - Digest: kết quả RIPEMD160 hash (20 bytes) của key + key type id + set name. Ví dụ: Mình có key của record đó là "key1", kiểu dữ liệu của key này là String, và record này nằm trong set là "set1". Khi đó, digest sẽ là kết quả hash của "key1" + id của kiểu dữ liệu String + "set1".
    - Khóa chính 64 bytes = Digest + record metadata + datalocation + otherinfo. Vì thế, tất cả record đều có 1 khóa chính duy nhất, và khoá này luôn cố định là 64 bytes.



- **Record metadata:**
    - Last update time (LUT)
    - Generation (GEN): đếm số lần record được update
    - Time-To-Live (TTL): thời gian tồn tại, có thể là phút, ngày; tối đa là 10 năm hoặc có thể là mặc định live-for-ever.



- **Record Data Location:**



- Data-on-Device (Disk): (8 bytes) bao gồm device ID (tối đa 128 devices), offset (tối đa 2TB), size của record (tối đa 8MB)
- Data-in-memory: (8 bytes) bao gồm pointer trỏ đến data trong RAM. (nếu data là các kiểu dữ liệu đơn giản như int, Boolean, thì có thể được lưu trong 8 bytes này luôn)

- **Chi tiết cấu trúc của khóa chính:**

Primary Index – 64 Bytes – Ver 4.2+

2 bytes – Internal housekeeping counter. (Additional **1 byte** reserved for future use)
1 byte – Partition change history Id, (6 bits), R-B Tree Color (1 bit), 1 bit unused
20 bytes - Digest (RIPEMD-160 Hash of "set name + key value type (1byte) + key value", 12 specific bits out of these 20 bytes used to calculate the partition ID)
10 bytes - R-B Tree navigation info. [5 bytes right*, 5 bytes left*]
2 bytes: Set id (10 bits), *1 bit for XDR-5.0 write*, (5 bits unused)
4 bytes - (1 unused bit +) 30 bits - delta from epoch time for implementing **TTL** (When current time delta from epoch time equals to record delta, record is no longer available for reads, deleted eventually) + **1 bit** for XDR-5.0 tombstone.
5 bytes - Last update timestamp
2 bytes - Generation counter. (**AP mode** – 16 bits: Max 64K, **SC Mode** – 10 bits: Max 1K + 6 bits partition regime LSBs)
8 bytes - SSD Record location info (64 bits)
37 bits for offset, spaced **16 byte blocks** => 2^{37} x 16 = 2 TB of storage per SSD is addressable.
19 bits for record size in number of 16 byte blocks. 2^{19} x 16 = **8 MB max record size**
7 bits for file_id => max 128 SSD drives
1 bit for key stored
1 Byte - Flex bits (Used for single-bin, replication state, tombstones etc.)
8 Bytes - Data in memory location pointer or data-in-index storage with additional 4 bits borrowed from flex bits.

rc | ID Bits | Digest | RBTree | SetId | TTL | LastUpdate | Gen | SSD ptr-size-ids | Mem-ptr
(2+1 | 1 | 20 | 10 | 2 | 4 | 5 | 2 | 8 | 1 | 8) = 64 bytes total.

- Bên cạnh việc truy vấn bằng cách tra khóa chính, Aerospike còn hỗ trợ parallel scans, secondary indexing, geospatial indexing and filtering, user-defined functions.

3. Quản lý lưu trữ (Storage Management):

a. Cấu hình đồng nhất:

- All DRAM: Tất cả chỉ mục (indexes) và dữ liệu được lưu trong DRAM (dynamic RAM). Dữ liệu được lưu trong disk chỉ để dự phòng.
- All PMEM: Tất cả chỉ mục và dữ liệu được lưu trong PMEM (persistent memory).
- All FLASH: Tất cả chỉ mục và dữ liệu được lưu trong Flash.

b. Cấu hình hỗn hợp:

- Lưu dữ liệu trong Flash, chỉ mục trong DRAM hoặc PMEM.
- Mặc định: Aerospike lưu trữ chỉ mục trong DRAM.
  ➔ Giảm thiểu tối đa độ trễ khi đọc do disk I/O.
- Ngoài ra, người dùng có thể tự cấu hình cách lưu trữ trên từng namespace theo cách mình mong muốn.

4. Phân bổ dữ liệu (Data distribution):

- Aerospike sử dụng kiến trúc Shared-nothing: các truy vấn chỉ được xử lý bởi một node trong cluster giúp hạn chế SPOF (single points of failure).

a. Phân vùng:

- Mỗi namespaces được chia thành 4096 phân vùng (partitions), được phân tán đều khắp các cluster node.
- Phân bổ record vào các phân vùng: Sử dụng thuật toán hash RIPEMD, Aerospike hash khóa chính của một record thành một giá trị (hash value) có 160 bit. Sau đó, Aerospike sử dụng 12 bit của giá trị này để xác định phân vùng mà record này thuộc về.
- Phân bổ phân vùng vào các nodes:
  - Sử dụng một thuật toán hash ngẫu nhiên để đảm bảo các phân vùng được phân đều vào các nodes. Ngoài ra, tất cả node là cặp, khi một node gặp sự cố sẽ không ảnh hưởng tới toàn bộ database.
  - Khi một node mới được thêm hoặc xóa bỏ khỏi cluster, một cluster mới sẽ được tạo, và các phân vùng sẽ được phân bổ đều lại vào các node.

  ➔ Tất cả dữ liệu được phân bổ đều và ngẫu nhiên vào các node, nên sẽ không tạo thành các hotspots, bottlenecks (một node phải chịu truy vấn nhiều hơn các node khác), giúp cải thiện hiệu suất và giảm thiểu rủi ro.

b. Sao chép và đồng bộ hóa dữ liệu (Data Replication and Synchronization):

- Để đảm bảo tính khả dụng và độ tin cậy cao, Aerospike sẽ tạo các bản sao của các phân vùng trên một hoặc nhiều node. Mỗi node có thể chứa data chính (data master) và bản sao của các data master của các node khác.
- Người dùng có thể cấu hình chỉ số mở rộng (replication factor). Chỉ số này càng cao, càng nhiều bản sao sẽ được tạo ra, và độ tin cậy sẽ càng cao, tuy nhiên hiệu suất sẽ thấp hơn do mỗi truy vấn ghi phải được thực hiện trên tất cả bản sao. Ngoài ra, chỉ số này không được vượt quá số node có trong cluster.
- Ví dụ:
    - Khi chỉ số mở rộng = 1 (không tạo bản sao): Giả sử trong cluster có 4 node, mỗi node sẽ chứa 1024 phân vùng ngẫu nhiên trong tổng cộng 4096 phân vùng. Tất cả node đều sẽ là data master.
    - Khi chỉ số mở rộng = 2 (một bản chính, một bản sao): Lúc này, mỗi node sẽ chứa 1024 phân vùng là data master, và 1024 phân vùng là một phần bản sao (replica) của các data master của các node còn lại.

c. Cơ chế tái cân bằng dữ liệu (Automatic Rebalancing):

- Đảm bảo lượng truy vấn được chia đều trên các node trong cluster.
- Đảm bảo tính khả dụng liên tục, dù cho có node bị lỗi.
- Người dùng có thể cấu hình tốc độ của quá trình tái cân bằng. Có thể tăng tốc độ của việc tái cân bằng cách giảm tốc độ xử lý giao tác.
- Trong quá trình cân bằng, một vài node có thể không giữ nguyên chỉ số mở rộng như ban đầu.
- Quá trình tái cân bằng là tự động, không cần sự can thiệp của các người điều hành. Khi một node gặp sự cố, cơ chế sẽ tự cân bằng và khắc phục, những người điều hành không cần phải làm gì để duy trì cluster đó.

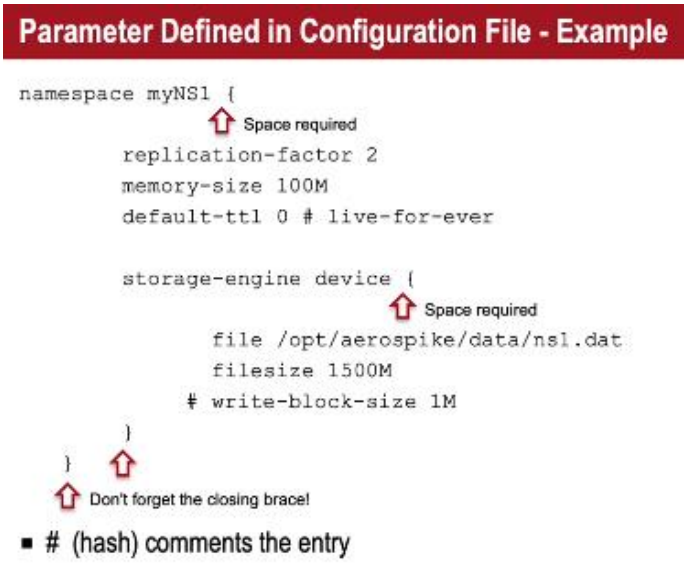
Ngoài ra, Aerospike còn có các cơ chế quản lý bão hòa lưu lượng (Traffic Saturation Management) và cơ chế xử lý tràn dung lượng (Capacity Overflows).

5. Thiết lập Aerospike (configuring Aerospike):

a. Configuration file:

- File thiết lập database Aerospike.
- Có bảy loại tham số (contexts), trong đó tham số network và ít nhất 1 tham số namespace phải được thiết lập mới có thể bắt đầu chạy Aerospike server.
- Đường dẫn mặc định: /etc/aerospike. Tên mặc định: aerospike.conf.

- 7 loại tham số (có thể có các loại con (sub-context) trong các loại này):
  - Network
  - Namespace
  - Service
  - Logging
  - Security
  - Xdr
  - mod-lua
- Ví dụ:



Giải thích:

Replication-factor: chỉ số mở rộng. Chỉ số này là 2, tức là sẽ có 1 bản replica, 1 bản master

Memory-size: hông bt :v

Default-ttl: thời gian tồn tại: 0 nghĩa là live-for-ever

File /opt/aerospike/data/ns1.dat: đường dẫn chứa file data của namespace 1.

Filesize: qui định kích thước tối đa của 1 file. Ở đây là 1500M.

b. Thiết lập Namespace:

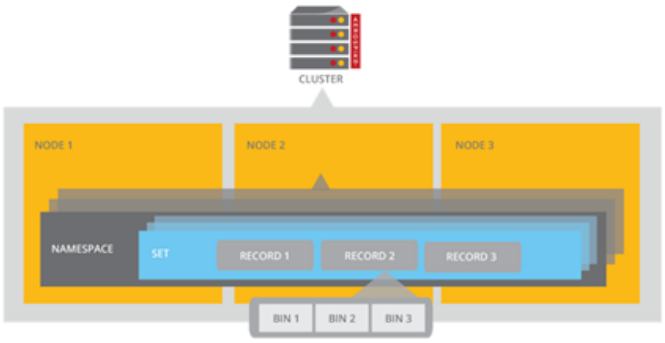
The general configuration parameters for every namespace are:

```
namespace myNS1 {
    replication-factor 2        # Master and One Replica
    high-water-memory-pct 0     # percent, threshold disabled by default
    high-water-disk-pct 0       # percent, threshold disabled by default
    stop-writes-pct 90          # percent
    memory-size 4G              # min 1M, required parameter, no default
    index-stage-size 1G         # min 128M, max 16G
    default-ttl 0           # Default live-for-ever, other e.g. 30d: 30 days expiration
    nsup-period 0           # seconds, 0: disabled by default.

    storage-engine ...{         # Discussed in next section
       # min-avail-pct 5          # percent, device or pmem
    }
}
```

II. Đặc điểm:

Aerospike là hệ quản trị lưu dữ liệu theo hàng, có nghĩa là mọi bản ghi (tương đương với một hàng trong cơ sở dữ liệu quan hệ) được xác định bằng duy nhất 1 khóa (key-value store). Khóa của bản ghi và các dữ liệu khác của nó sẽ nằm trong chỉ mục chính (primary index). Dữ liệu của bản ghi (record) sẽ nằm trong thiết bị lưu trữ đã được định nghĩa trước của không gian (namespace) mà nó chiếm giữ.



Aerospike	RDBMS
Namespace	Tablespace or Database
Set	Table
Record	Row
Bin	Column

Bin type
Integer
String
BLOB
List
Map

- Mỗi thùng chứa (bin) có thể giữ một kiểu dữ liệu dạng tập hợp (list, map, …) hoặc giữ một kiểu dữ liệu dạng đơn (int, double, string, …). Thùng chứa (bin) tương đương với một cột (thuộc tính) trong cơ sở dữ liệu quan hệ.
- Aerospike cho phép các truy vấn dựa trên giá trị bằng cách sử dụng các chỉ mục phụ (secondary index), trong đó các giá trị chuỗi và số nguyên được lập chỉ mục và tìm kiếm bằng cách lọc các giá trị bằng nhau (dành cho cả chuỗi và số nguyên) hoặc lọc trên phạm vi giá trị (chỉ dành cho số nguyên).
- Các chức năng do người dùng xác định (UDF) mở rộng chức năng và khả năng hoạt động của công cụ Cơ sở dữ liệu Aerospike.
- Trong Aerospike, khung kiến trúc Kết tập (aggregation) cho phép các hoạt động truy vấn linh hoạt và nhanh chóng. Giống như hệ thống MapReduce, các phép kết tập cho ra kết quả nhanh tương tự.
- Aerospike hỗ trợ lưu trữ, lập chỉ mục và truy vấn dữ liệu trong không gian dưới dạng GeoJSON.

1. Thao tác Key-value

Aerospike cung cấp hỗ trợ đầy đủ cho mô hình lưu trữ tài liệu và mô hình lưu trữ khóa-giá trị. Trên thực tế, các hoạt động của nó cung cấp hỗ trợ rộng hơn nhiều so với các định nghĩa trong sách giáo khoa về các mô hình đó. Aerospike là một cơ sở dữ liệu hướng hàng (row-oriented), trong đó mọi bản ghi (record) được xác định duy nhất bằng một khóa. Khóa của bản ghi và siêu dữ liệu khác của nó nằm trong chỉ mục chính. Dữ liệu của bản ghi nằm trong thiết bị lưu trữ được xác định trước của Không gian tên mà nó chiếm.

Các mô hình được hỗ trợ:

Aerospike hỗ trợ cả mô hình lưu trữ khóa-giá trị và lưu trữ tài liệu:

a. Mô hình lưu trữ khóa-giá trị (key-value store model)

Các bản ghi được xác định duy nhất bởi một Khóa bao gồm tên của Bộ và khóa người dùng. Khóa người dùng là mã định danh chính cho bản ghi từ ứng dụng bạn đang xây dựng trên Aerospike. Mặc dù một giá trị trong mô hình lưu trữ khóa-giá trị là một khối dữ liệu không rõ ràng và không có hỗn hợp, tuy vậy thì các bản ghi bao gồm một hoặc nhiều thùng dữ liệu khối đã xác định hoặc chưa có kiểu dữ liệu.

b. Mô hình lưu trữ tài liệu (document store model)

Một biến thể của mô hình lưu trữ khóa-giá trị, trong đó một bản ghi là một tài liệu dữ liệu duy nhất, được mã hóa bằng XML, YAML, JSON, BSON. Người dùng của Aerospike thường lưu trữ một tài liệu của dữ liệu trong loại dữ liệu Bản đồ (Map). Bản đồ Aerospike là một tập hợp JSON có thể chứa các kiểu dữ liệu khác nhau, bao gồm cả cấu trúc Bản đồ và Danh sách lồng vào nhau. Bản đồ sử dụng tính năng nén MessagePack một cách rõ ràng để lưu trữ hiệu quả. Trong khi mô hình lưu trữ tài liệu truyền thống lưu trữ một tài liệu dữ liệu trên mỗi bản ghi, bản ghi Aerospike có thể chứa nhiều ngăn và do đó có thể có nhiều kiểu dữ liệu cho mỗi bản ghi.

2. Các loại thao tác (Operation types)

Aerospike cung cấp hai loại thao tác nguyên tố, hỗ trợ các trường hợp sử dụng rộng hơn so với hai mô hình được mô tả ở trên. Vì Aerospike lưu trữ dữ liệu theo cặp khóa và giá trị, nên các thực thi sẽ được mô tả như là các thao tác Khóa-Giá trị.

Bộ lọc thao tác: Thao tác có thể được áp dụng có điều kiện, dựa trên kết quả của một biểu thức bộ lọc. Nếu bộ lọc đánh giá là true, thao tác sẽ thực thi.

a. Các thao tác trên bản ghi (Record Operations):

Là các hàm tạo, đọc, cập nhật, xóa bản ghi (CRUD – create, read, update, delete).

- Aerospike cung cấp các giao diện đơn giản để đọc dữ liệu và ghi dữ liệu vào một cụm (cluster). Chúng bao gồm các trường hợp sử dụng tiêu chuẩn CRUD.
- Mọi máy khách (client) của Aerospike đều có thể thực hiện các thao tác ghi đầy đủ sau đây, bất kể nó được viết bằng ngôn ngữ nào:

Thao tác (operation)	**Chức năng**
put	Viết một bản ghi. Thực hiện tương tự như create (tạo), update (cập nhật) hoặc replace (thay thế).
delete	Xóa bản ghi khỏi cơ sở dữ liệu.
get	Đọc bản ghi. Có thể chọn các thùng dữ liệu cụ thể.
add	Thêm (hoặc trừ) một giá trị trong thùng dữ liệu số nguyên. Được sử dụng để thực hiện bộ đếm, còn được gọi là increment.
append prepend	Sửa đổi thùng dữ liệu kiểu chuỗi hoặc mảng byte.
touch	Tăng bộ đếm.
operate	Áp dụng nhiều thao tác thùng dữ liệu như một giao tác nguyên tố.

b. Các thao tác trên giao tác (Transaction Operations):

Là các thao tác dựa trên thùng được thực thi theo trình tự bằng cách sử dụng phương thức máy khách operation().

- Bản ghi chứa một hoặc nhiều Thùng có nhiều kiểu dữ liệu khác nhau. Bản đồ và danh sách lồng nhau, còn được gọi là kiểu dữ liệu tập hợp (CDT – Collection data types) có thể được kết hợp để triển khai cấu trúc dữ liệu phức tạp dưới dạng tài liệu. Để cung cấp các tài liệu dữ liệu phức tạp này, mọi máy khách (client) của Aerospike cung cấp phương pháp sau:

- Operate: kết hợp nhiều thao tác (operation) trên một bản ghi thành một giao tác (transaction). Các hoạt động bao gồm các hoạt động đọc và ghi trên Thùng, và bất kỳ hoạt động nào trên một kiểu dữ liệu, được lưu trữ trong một Thùng.

- Một trong những đối số của phương thức "OPERATE" là danh sách các thao tác trên Thùng có thứ tự để thực thi. Cú pháp được sử dụng bởi các máy khách Aerospike khác nhau tùy theo ngôn ngữ lập trình. Không giống như thao tác "GET" trên bản ghi sẽ trả về bản ghi, thao tác này trả về kết quả đầu ra từ mỗi thao tác con trên mỗi thùng. Điều này rất hữu ích để xác nhận quá trình xử lý giao tác và khắc phục sự cố logic dẫn đến giao tác thành công hoặc thất bại không mong muốn.

- Mỗi lệnh gọi "OPERATE" dẫn đến một giao tác tuân thủ ACID bao gồm một hoặc nhiều thao tác trên Thùng có thứ tự trên bản ghi. Thao tác này là một giao tác nguyên tố đơn lẻ được xử lý bởi Aerospike dựa theo một khóa trên bản ghi với mức cô lập và tính bền vững. Để so sánh, mọi thao tác ghi dẫn đến một giao tác nguyên tố bao gồm một thao tác duy nhất.

- Sau đây là danh sách đầy đủ các thao tác trên Thùng có thể là một phần của Giao tác có sử dụng thao tác. Bảng bên dưới chứa các thao tác hoặc liên kết đến danh sách đầy đủ các thao tác cho từng loại dữ liệu.

Kiểu dữ liệu (data type)	**Thao tác (Operation)**	**Chức năng**
Tất cả	op_put	Nâng cấp (tạo hoặc cập nhật) trên Thùng. Còn được gọi là write.
Tất cả	op_get	Đọc trên Thùng. Còn được gọi là read.
Tất cả	op_touch	Tăng bộ đếm trên bản ghi.
Tất cả	op_delete	Xóa bản ghi khỏi cơ sở dữ liệu.
Số nguyên (Integer) Số thực (Float)	op_add	Cộng (hoặc trừ) một giá trị. Được sử dụng để thực hiện bộ đếm. Còn được gọi là increment.
Chuỗi (String)	op_append op_prepend	Sửa đổi trên chuỗi.
Danh sách (List)	Danh sách các thao tác trên Lists	
Bản đồ (Map)	Danh sách các thao tác trên Maps	
Khối (Blob)/Bytes	Danh sách các thao tác trên Blob/Bytes	
HLL - HyperLogLog	Danh sách các thao tác trên HyperLogLog	
Geo	Danh sách các thao tác trên Geospatial	

c. Thao tác đồng thời (Batch Operations):

Yêu cầu hàng loạt gửi cùng một thao tác đọc cho nhiều bản ghi song song. Đây là một dạng truyền có thể sử dụng băng thông mạng hiệu quả hơn và giảm độ trễ khi một số lượng lớn các khóa chính tham gia vào.

Sử dụng các yêu cầu hàng loạt để:

- Triển khai hiệu quả phép kết (join) phía máy khách.
- Lưu trữ và sau đó truy xuất nhiều điểm dữ liệu trong một phép tính.
- Xác định xem nhiều khóa có còn tồn tại hay không.

Aerospike hỗ trợ các yêu cầu hàng loạt này:

- get: Đọc nhiều bản ghi hoặc tùy chọn một thùng từ các bản ghi đó
- exists: Thực hiện kiểm tra siêu dữ liệu xem các khóa được chỉ định có tồn tại hay không.
- getHeader: Chỉ đọc siêu dữ liệu bản ghi. Không trả về Thùng dữ liệu.
- operate: Thực hiện một giao tác của các thao tác đọc đối với nhiều bản ghi.

3. Một số lưu ý trong việc triển khai:

Các đặc điểm của Aerospike cho thấy các phương pháp hay nhất để triển khai mô hình dữ liệu. Một số điểm cần ghi nhớ và lưu ý như:

- Aerospike là vô hình. Trước khi đưa dữ liệu vào cơ sở dữ liệu, bạn xác định không gian để liên kết dữ liệu với các loại phương tiện lưu trữ phần cứng. Bằng cách so sánh, các tập hợp (tương đương với mỗi bảng trong cơ sở dữ liệu quan hệ) và thùng (tương đương với mỗi cột trong cơ sở dữ liệu quan hệ) được tạo trong thời gian thực khi nó thực hiện các thao tác ghi vào Aerospike.
- Aerospike hỗ trợ các giao tác cho các bản ghi nguyên tố. Nhiều thao tác trên một hoặc nhiều thùng chứa bản ghi có thể được kết hợp thành một giao tác. Giao tác (transaction) được thực thi dưới sự kiểm soát của các khóa bản ghi với mức cô lập và tính bền vững.
- Mỗi loại dữ liệu của Aerospike đều có API riêng của các thao tác ghi và thao tác đọc phía máy chủ (server), chẳng hạn như hàm get_by_rank_range () của loại dữ liệu Bản đồ.
- Đối với các thao tác phi nguyên tố, siêu dữ liệu tạo của bản ghi (số lượng sửa đổi dữ liệu) có thể được sử dụng để đảm bảo rằng dữ liệu đang được ghi không bị sửa đổi kể từ lần đọc cuối cùng. Sự tối ưu đồng thời này cho phép sử dụng mẫu kiểm tra và thiết lập (đọc-sửa đổi-ghi).

4. Các kiểu dữ liệu

Các bản ghi gồm một hoặc nhiều Thùng. Mỗi thùng có thể chứa kiểu dữ liệu dạng đơn (số nguyên, chuỗi,…) hoặc dạng tập hợp (danh sách, bản đồ) hoặc dữ liệu dạng xác suất (HyperLogLog) hoặc dữ liệu không gian (GeoJSON)

a. Kiểu dữ liệu dạng đơn

- Integer (số nguyên): giá trị số 64-bit. Mỗi số nguyên yêu cầu 8 bytes (64 bits) dung lượng lưu trữ. Phạm vi giá trị từ [-(2^{63}) đến 2^{63}-1]. Có thể áp dụng phép toán tăng increment để triển khai bộ đếm.
- Double (số thực): được lưu trữ ở định dạng IEEE-754 64-bit. Có thể áp dụng phép toán tăng increment.
- String (chuỗi): mặc định mã hóa dạng UTF-8. Kích thước bản ghi giới hạn 128kb
- Boolean: giữ giá trị true hoặc false, được lưu trữ dưới dạng một byte duy nhất trên máy chủ
- Blob/Bytes: Aerospike hỗ trợ hai loại dữ liệu Blob/Bytes chính. Có các thùng Blob dành cho mục đích chung và thùng Blob dành riêng cho ngôn ngữ.
    - Blob/Bytes chung: là mảng các byte có kích thước cụ thể. Mọi dữ liệu nhị phân thuộc bất kỳ loại nào đều có thể được lưu trữ. Các byte không được kết thúc bằng NULL. Có hỗ trợ các thao tác Bitwise.
    - Blob dành riêng cho ngôn ngữ (Language-Specific Serialized Blobs): chỉ có thể truy cập được từ thư viện ngôn ngữ của máy khách sử dụng. Chúng rất hữu ích để tuần tự hóa các đối tượng bằng ngôn ngữ đó vào cơ sở dữ liệu Aerospike. Hiện tại Aerospike hỗ trợ các loại Blob theo ngôn ngữ cụ thể sau: C#, Java, PHP, Python và Ruby.

b. Kiểu dữ liệu dạng tập hợp

Các kiểu dữ liệu tập hợp (CDT) là các vùng chứa linh hoạt, không có giản đồ, có thể chứa dữ liệu dạng đơn hoặc lồng các tập hợp khác vào trong chúng. Các phần tử trong CDT có thể thuộc nhiều loại khác nhau. CDT là một tập hợp lớn của JSON, hỗ trợ nhiều kiểu dữ liệu hơn dưới dạng phần tử của tập hợp, chẳng hạn như số nguyên cho các khóa Bản đồ và dữ liệu nhị phân dưới dạng Danh sách.

- List (danh sách): là kiểu dữ liệu trừu tượng gồm 2 loại nhỏ là Danh sách có thứ tự và không có thứ tự. Cả hai loại đều chia sẻ cùng một API và có thể được truyền giữa nhau bằng thao tác set_order(). Một danh sách có thể lưu trữ các kiểu dữ liệu dạng đơn, hoặc chứa các Bản đồ/Danh

sách lồng nhau. Các thao tác trên Danh sách là cách tốt nhất để thao tác với danh sách trực tiếp trên cơ sở dữ liệu của Aerospike.

- Map (bản đồ): là kiểu dữ liệu trừu tượng gồm 3 loại nhỏ: Bản đồ có thứ tự Khóa (K-ordered), Bản đồ có thứ tự Khóa-Giá trị (KV-ordered) và Bản đồ không theo thứ tự. Tất cả các loại Bản đồ đều dùng chung một API và có thể truyền giữa nhau bằng thao tác set_type(). Bản đồ có thể lưu trữ các kiểu dữ liệu dạng đơn hoặc chứa các Bản đồ/Danh sách lồng nhau. Thao tác trên bản đồ là tối ưu để thao tác với bản đồ trực tiếp trên cơ sở dữ liệu của Aerospike. Các phần tử trong Bản đồ được lưu trữ dựa trên kiểu sắp xếp. Các khóa bản đồ có thể thuộc loại Chuỗi, Byte, Số nguyên hoặc Số thực.

c. Chỉ mục không gian và truy vấn

Sử dụng lưu trữ không gian, lập chỉ mục và truy vấn để cho phép truy vấn nhanh về các điểm trong một vùng, trên một vùng có chứa các điểm và các điểm trong bán kính. Định dạng GeoJSON để chỉ định và lưu trữ các đối tượng hình học GeoJSON, đồng thời đạt được tiêu chuẩn hóa dữ liệu và khả năng tương tác.

d. Dữ liệu xác suất

Kiểu dữ liệu HyperLogLog cung cấp số lượng ước tính của các thành viên trong một tập dữ liệu lớn để ứng dụng của bạn tạo thành các ước tính nhanh, hợp lý của các thành viên trong liên hiệp hoặc giao nhau giữa nhiều thùng HyperLogLog. Các ước tính của HyperLogLog là sự cân bằng giữa độ chính xác hoàn toàn và khả năng tiết kiệm hiệu quả về dung lượng cũng như tốc độ xử lý các tập dữ liệu cực lớn. Dữ liệu HyperLogLog được trả về ứng dụng của bạn dựa trên một số ý tưởng cơ bản từ lý thuyết tập hợp.

5. Tính nhất quán (Consistency)

Tính nhất quán mạnh mẽ đảm bảo rằng tất cả các lần ghi vào một bản ghi sẽ được áp dụng theo một thứ tự cụ thể (tuần tự) và các lần ghi sẽ không được sắp xếp lại hoặc bỏ qua, tức là chúng sẽ không bị mất. Trong tất cả các hình thức Nhất quán mạnh mẽ, việc ghi sẽ không bị mất (trong giới hạn của các lỗi phần cứng đồng thời).

Aerospike đảm bảo rằng dữ liệu sẽ không bị mất, với ba trường hợp ngoại lệ:

- Quá trình tạm dừng nút hơn 27 giây hoặc độ lệch đồng hồ giữa các cụm nút lớn hơn 27 giây.
- Máy chủ không đáng tin cậy đồng thời bị tắt khi "commit-to-device" không được bật.
- Nhiều lỗi lưu trữ phần cứng trước khi dữ liệu có thể tái tạo.

Trong mỗi trường hợp, Aerospike cố gắng cung cấp tính khả dụng tốt nhất và giảm thiểu các vấn đề có thể xảy ra. Trong trường hợp lệch đồng hồ, Aerospike liên tục theo dõi lượng sai lệch và sẽ cảnh báo nếu độ lệch trở nên lớn - và vô hiệu hóa rước khi xảy ra mất dữ liệu. Trong trường hợp máy chủ gặp sự cố hoặc tắt máy, Aerospike sẽ tự động nhanh chóng tái tạo và cân bằng lại dữ liệu trong trường hợp lỗi đầu tiên. Nếu sự cố xảy ra nhanh chóng, thì các phần của dữ liệu có thể bị đánh dấu là "chết" và cần sự can thiệp của người vận hành. Điều này là để cho phép sử dụng dữ liệu ở chế độ chỉ đọc hoặc cho phép tải lại các thay đổi gần đây.

6. Truy vấn (Query)

Sử dụng các truy vấn nhanh đồng thời của Aerospike để thực hiện các tìm kiếm dựa trên giá trị bằng cách sử dụng các chỉ mục phụ. Các truy vấn dựa trên Chỉ mục phụ có thể trả về một tập hợp các bản ghi rất nhanh chóng, so với việc thay thế quét bảng.

Với chỉ mục phụ, có thể thực hiện truy vấn sau:

- Truy vấn ngang bằng với chỉ mục chuỗi hoặc số.
- Truy vấn phạm vi đối với các chỉ mục số. Tập hợp kết quả phạm vi là bao gồm (nghĩa là, cả hai giá trị đã chỉ định đều được bao gồm trong kết quả).
- Truy vấn điểm trong vùng (Point-In-Region) hoặc vùng chứa điểm (Region-Contain-Point) dựa trên chỉ mục địa lý.

Các kết quả của truy vấn chỉ mục phụ có thể tùy chọn được xử lý sau bằng Bộ lọc dự đoán của Aerospike hoặc Chức năng do người dùng xác định (UDF) của Aerospike trước khi trả về máy khách.

Sử dụng tính năng Bộ lọc dự đoán của Aerospike, bạn có thể:

- Lọc ra các bản ghi dựa trên dữ liệu meta bản ghi, chẳng hạn như thời gian cập nhật cuối cùng hoặc kích thước lưu trữ.
- Lọc ra các bản ghi dựa trên Thùng dữ liệu, chẳng hạn như số nguyên lớn hơn/nhỏ hơn trên các Thùng chuỗi.

Sử dụng các UDF của Aerospike, bạn có thể:

- Sử dụng bản đồ / giảm chức năng để thực hiện các phép kết hợp.
- Sử dụng các UDF trên bản ghi để tác động các bản ghi.
- Sử dụng toán tử không gian địa lý để trả về thông tin bản ghi vị trí địa lý.

7. Phép tổng hợp (Aggregation)

Sử dụng các UDF của Aerospike để tổng hợp các kết quả truy vấn theo kiểu phân tán. Khung tổng hợp Aerospike cho phép các hoạt động truy vấn linh hoạt và nhanh chóng. Khung chương trình này tương tự như một hệ thống MapReduce, trong đó một chức năng bản đồ ban đầu chạy trên một tập hợp và đưa ra kết quả. Kết quả đi qua một đường dẫn của các bước bản đồ tiếp theo hoặc các bước rút gọn và các bước tổng hợp.

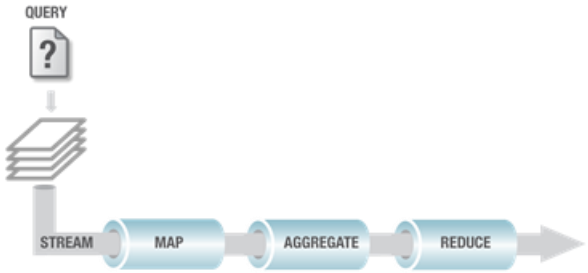
Không giống như Hadoop hoặc các khung công tác khác sử dụng Java, tập hợp Aerospike được thực hiện bằng cách sử dụng Lua. Mỗi máy khách gửi một yêu cầu tổng hợp đến tất cả các máy chủ trong cụm, máy chủ này sẽ đếm kết quả một cách độc lập và trả về các kết quả riêng lẻ cho máy khách yêu cầu.

Khung tổng hợp Aerospike khác với các hệ thống khác vì Aerospike khuyến nghị chạy tập hợp dựa trên một chỉ mục; đây thực chất là một mệnh đề "WHERE". Lọc theo chỉ mục duy trì hiệu suất cao. Aerospike hỗ trợ tổng hợp các bảng và toàn bộ không gian tên. Sau đó, khách hàng chạy một giai đoạn giảm cuối cùng, cũng trong Lua, để tổng hợp kết quả.

Trường hợp sử dụng:

- Triển khai các hàm tổng hợp như SUM, COUNT, MIN, MAX dưới dạng các UDF do người dùng xác định.
- Bảng điều khiển thời gian thực.
- Sử dụng các chỉ mục phụ trên Thùng với thời gian cập nhật, tổng hợp nhanh chóng thu thập số liệu thống kê về các bản ghi được thay đổi gần đây.

Để triển khai phép tổng hợp:



- Tạo một truy vấn.
  - Tạo chỉ mục trên thùng.
  - Chèn bản ghi vào thùng được lập chỉ mục.
- Tạo một mô-đun tổng hợp trong Lua.
- Đặt đường dẫn mô-đun tổng hợp.
- Đăng ký mô-đun với cụm Aerospike.
- Tạo một truy vấn tổng hợp với một vị từ (mệnh đề where).

III. Giao tác:

Aerospike là một cơ sở dữ liệu Khóa-Giá trị hiệu suất cao. Nó nhắm đến các tổ chức và trường hợp sử dụng cần thông lượng cao, với độ trễ thấp (hoàn thành 95% trong <1ms), đồng thời quản lý lượng lớn dữ liệu với 100% thời gian hoạt động, khả năng mở rộng và chi phí thấp.

Đối với một lớp dữ liệu, tính đúng đắn là rất quan trọng. Các giao tác cơ sở dữ liệu cung cấp đảm bảo ACID cho các dữ liệu đó:

- Atomic (Tính nguyên tố): Thành công hoặc thất bại về tổng thể.
- Consistency (Tính nhất quán): Để cơ sở dữ liệu ở trạng thái nhất quán.
- Isolation (Tính độc lập): Có thể được sắp xếp theo thứ tự đối với các giao tác khác.
- Durability (Tính bền vững): Lưu trữ dữ liệu khi có sự cố về nguồn hoặc nút.

Trong cơ sở dữ liệu phân tán và sao chép, bản cập nhật giao tác cũng phải đảm bảo tính nhất quán của các giá trị trên tất cả các bản sao. Đối với một lớp dữ liệu khác, bạn có thể quyết định ưu tiên tính khả dụng hơn tính nhất quán.

Các doanh nghiệp dành nhiều nỗ lực trong việc thiết kế, phát triển và vận hành nhiều cơ sở dữ liệu được nhắm mục tiêu cho các nhu cầu dữ liệu đa dạng như vậy. May mắn thay, Aerospike xử lý truy cập dữ liệu đa dạng qua phổ tính khả dụng và nhất quán ở hiệu suất mili giây có thể dự đoán được, quy mô petabyte và khả năng phục hồi cao. Aerospike cũng loại bỏ nhu cầu về bộ nhớ đệm front-end thường cần thiết để đáp ứng các yêu cầu về độ trễ và thông lượng của các ứng dụng hiện đại vốn làm phức tạp hơn nữa việc thiết kế và triển khai.

1. Giao tác trong Aerospike

Trong Aerospike, tất cả các thao tác trên bản ghi đơn lẻ đều có giao tác để đảm bảo. Mọi yêu cầu trên mỗi bản ghi, bao gồm cả những yêu cầu chứa nhiều thao tác trên một hoặc nhiều Thùng, thực hiện nguyên tố dưới một khóa bản ghi với mức cô lập và tính bền vững, và đảm bảo rằng tất cả các bản sao đều nhất quán.

Các giao tác không vượt qua nhiều ranh giới bản ghi. Một ứng dụng có thể loại bỏ hoặc giảm thiểu nhu cầu giao tác trên nhiều bản ghi (MRT – Multi-Record Transactions) bằng cách tận dụng mô hình dữ liệu phong phú của Aerospike bao gồm các loại Bản đồ và Danh sách lồng nhau cũng như các giao tác trên bản ghi đơn đa hoạt động của nó. Với mô hình dữ liệu thích hợp, có thể

tránh được các giao tác trên nhiều bản ghi yêu cầu các phép kết (join) và các ràng buộc. Ví dụ: dữ liệu trong nhiều bảng trong cơ sở dữ liệu quan hệ có thể được mô hình hóa dưới dạng một bản ghi không chuẩn hóa duy nhất. Do đó, các hoạt động kiểm soát giao tác đa hoạt động như Begin, commit, Abort không phải là một phần của Aerospike API.

2. Cài đặt các giao tác đọc-ghi

Aerospike cho phép nhiều thao tác đọc / ghi trên cùng một khóa trong một giao tác duy nhất. Tuy nhiên, ghi trong Aerospike là một thao tác đơn giản (set, add, append) và không thể là một chức năng tùy ý của một hoặc nhiều thùng trong bản ghi. Nói cách khác, logic cập nhật phức tạp không thể được gửi đến máy chủ để thực hiện trong một giao tác. Thiết kế Aerospike này giữ cho các thao tác đọc thông thường trở nên đơn giản và nhanh chóng có thể dự đoán được, đồng thời trì hoãn ứng dụng bất kỳ sự phức tạp nào của các giao tác đọc-ghi ít thường xuyên hơn.

Aerospike có hai tùy chọn cho các giao tác đọc-ghi chung:

- Sử dụng mẫu Record Read-Modify-Write (hoặc Check-and-Set) ở phía máy khách trong ứng dụng.
- Cài đặt update logic trong chức năng do người dùng xác định (UDF) nằm trên máy chủ. UDF, tương tự như một thủ tục được lưu trữ trong cơ sở dữ liệu, phải được cài đặt bởi quản trị viên và sau đó có thể được gọi thông qua API.

Các UDF, được triển khai trong Lua và nằm trên máy chủ, thực thi, phát triển, kiểm tra và thay đổi chậm hơn. Vì vậy, chúng không phù hợp nhất cho hầu hết các giao tác đọc-ghi trong các ứng dụng. Thay vì thực hiện ghi có điều kiện đối với siêu dữ liệu tạo của bản ghi, giao tác ghi có thể được thực hiện có điều kiện đối với giá trị đọc bằng Bộ lọc vị từ. Trong trường hợp đó, việc ghi sẽ chỉ thành công nếu giá trị đã đọc trước đó không thay đổi.

3. Giải quyết các giao tác nghi ngờ:

Ứng dụng phải biết rằng giao tác thành công hay thất bại để có thể thực hiện bất kỳ điều chỉnh cần thiết nào. Nếu giao tác không thành công, ứng dụng có thể thực hiện các bước thích hợp để thử lại giao tác (hoặc có logic cao hơn bao gồm can thiệp thủ công để thực hiện hành động thích hợp).

Trong những trường hợp hiếm hoi trong quá trình chuyển đổi cụm (khi cụm đang tách), không thể biết ngay giao dịch thành công hay thất bại. Trong trường hợp như vậy, giao dịch ghi hết thời gian với cờ "InDoubt" trong đối tượng ngoại lệ được đặt thành true, phản ánh kết quả vẫn chưa biết. Giao tác nghi ngờ có thể được giải quyết khi phân vùng dữ liệu bị ảnh hưởng hoạt động trở lại như một phần của quá trình khôi phục cụm tự động.

Kết quả của một giao dịch có thể không xác định được khi mạng đang trong giai đoạn phân tách, khi ghi nhận được trong cụm con thiểu số và cụm con khác không thể truy cập được trước khi tất cả các bản sao được biết là đã được cập nhật thành công. Kết quả chính xác của giao tác ghi không có sẵn ngay lập tức là thành công hay thất bại. Phân vùng bị ảnh hưởng cần phải hoạt động và có thể truy cập được sau khi khôi phục để giải quyết kết quả ghi.

Yêu cầu bị nghi ngờ sẽ hết thời gian. Không có một cách tích hợp nào để giải quyết một giao tác bị nghi ngờ và do đó, máy khách phải đưa ra kế hoạch của riêng mình để tìm ra kết quả của việc ghi. Một lần nữa, điều này phù hợp với nguyên lý thiết kế của Aerospike rằng để giữ cho các thao tác thông thường trở nên đơn giản và nhanh chóng có thể đoán trước được, nó sẽ trì hoãn bất kỳ sự phức tạp nào thêm vào các ứng dụng để xử lý các tình huống hiếm gặp.

Một cách điển hình để giải quyết thao tác ghi bị nghi ngờ là ghi lại ID giao tác (một máy khách được tạo id duy nhất trên toàn cục) trong bản ghi cùng với bản cập nhật thông qua một yêu cầu đa hoạt động. Thùng trong bản ghi giữ danh sách các ID của giao tác đã thành công, vì mỗi lần ghi sẽ thêm trước ID giao tác vào danh sách thông qua một yêu cầu đa thao tác nguyên tố. Ứng dụng có thể xác định giao tác ghi nghi ngờ có thành công hay không bằng cách kiểm tra xem ID giao tác có trong danh sách hay không. Danh sách được giữ cho không phát triển vô thời hạn bằng cách cắt nó theo một kích thước cụ thể. Kích thước cần được chọn thích hợp cho tần suất ghi của ứng dụng: nó phải đủ lớn để đảm bảo ID giao tác không bị cắt bớt khi thực hiện kiểm tra, nhưng không quá lớn vì nó sẽ làm tăng kích thước bản ghi và dung lượng đĩa. Danh sách có thể được xem như một giao tác trên bản ghi cụ thể đơn giản do ứng dụng quản lý.

IV. Cơ chế quản lý giao tác đồng thời.

Đối với Aerospike, cơ chế quản lý giao tác đồng thời được gọi là Strong Consistency. Strong Consistency đảm bảo rằng tất cả dữ liệu được ghi vào trên mỗi một dòng dữ liệu được thực thi theo một trình tự nhất định (theo thứ tự tuần tự), và các dòng dữ liệu ghi vào sẽ không bị sắp xếp lại thứ tự hoặc là bị bỏ qua..v.v các dữ liệu này sẽ không mất

Ở mọi dạng của Strong Consistency, dữ liệu ghi vào sẽ không bị mất – trong giới hạn của các lỗi phần cứng đồng thời.

Aerospike đảm bảo rằng dữ liệu sẽ không bị mất, với ba trường hợp ngoại lệ:

1. Tiến trình xử lý các node dừng sau khi quá 27 giây, hoặc thời gian chênh lệch giữa các cụm node lớn hơn 27s

2. Server đồng thời không sạch hoặc không đáng tin cậy sập nếu **commit-to-device** (cơ chế chờ cho dữ liệu ghi vào đĩa hoặc RAM) không cho phép

3. Nhiều lỗi lưu trữ phần cứng trước khi dữ liệu có thể được sao chép

Trong mỗi trường hợp, Aerospike cố gắng cung cấp tính khả dụng tốt nhất và giảm thiểu các vấn đề có thể xảy ra.

Trong trường hợp lệch đồng hồ (ngoại lệ 1), một giao thức của Aerospike được gọi là ***Aerospike's gossip cluster protocol*** liên tục theo dõi lượng sai lệch thời gian và sẽ cảnh báo nếu độ lệch trở nên lớn - và vô hiệu hóa cụm này trước khi xảy ra mất dữ liệu.

Trong trường hợp server lỗi hoặc bị sập, Aerospike nhanh chóng tự động tái tạo và cân bằng lại dữ liệu trong trường hợp lần lỗi đầu tiên. Nếu lỗi xảy ra nhanh chóng, các phần của dữ liệu có thể được đánh dấu là “dead”, và yêu cầu sự can thiệp của người điều hành. Điều này là để cho phép sử dụng dữ liệu ở chế độ chỉ đọc hoặc cho phép tải lại các thay đổi trước đó .

1. Cấu hình cho Strong Consistency

Strong Consistency khả dụng cho toàn bộ một namespace.

Các bước để cấu hình Strong Consistency cho một namespace:

1. Lấy khóa để mở khóa cho cấu hình Strong Consistency.

2. Cài đặt NTP(hoặc đồng hồ đồng bộ hóa khác)

3. Thêm tệp cài đặt để xác định Strong consistency namespace.

4. Cấu hình cho roster ban đầu.

5. Cấu hình node IDs với Strong consistency

6. Cấu hình Rack Awareness

Chi tiết xem thêm: Tài liệu về quản lý Strong Consistency giải thích cách cấu hình một namespace và một roster.

2. Quản lý Strong Consistency:

Quản lý Strong Consistency namespace phức tạp hơn nhiều so với quản lý những namespace có sẵn.

Tài liệu này mô tả cách thêm và xóa bỏ node, bắt đầu và dừng server một cách an toàn, và nhiều khái niệm quản lý khác

3. Nâng cấp từ AP(Available and Partition-tolerant) lên SC(Strong Consistency) namespaces:

Về cơ bản, chuyển từ AP lên SC namespace không được hỗ trợ. Một số trường hợp vi phạm tính nhất quán có thể xảy ra trong quá trình chuyển đổi. Tạo ra một namespace, sau đó sao lưu và khôi phục được đề xuất trong trường hợp này (quá trình này được gọi là: thủ tục nâng cấp “forklift”).

4. Sử dụng Strong Consistency:

Có rất ít thay đổi lập trình viên có thể làm. Chỉ cần biết đơn giản là dữ liệu an toàn.

Tuy nhiên, vẫn có một số thay đổi mà lập trình viên có thể thực hiện chủ yếu được quản lý thông qua client Policy object, cũng như là sự khác biệt trong ý nghĩa của mã lỗi.

a. Cơ chế đọc khả tuyến tính: (Linearizable Reads)

Đọc khả tuyến tính là một trường mới tồn tại trong 'Policy' object mà lập trình viên có thể thay đổi. Để sử dụng đọc tuyến tính (Linearizable reads), cần phải đặt lại giá trị của trường **linearizableRead** thành **true.** Nếu giá trị trường này được thiết lập để đọc trên một namespace không cấu hình Strong consistency, việc đọc sẽ thất bại. Nếu đặt giá trị này lên một SC namespace, bạn sẽ đạt được Session Consistency.

Policy object có thể được cài đặt mặc định trong hàm khởi tạo của AerospikeClient, như vậy tất cả những lệnh gọi sau đó cũng sẽ mặc định dùng giá trị của Policy object. Ngược lại, ta nên sử dụng hàm khởi tạo Policy object trên từng khởi tạo để xác định rõ thực thi một cơ chế đọc tuần tự toàn phần, hay đọc nhất quán từng phần.

Một số Policy objects như là BatchPolicy, WritePolicy, QueryPolicy và ScanPolicy cũng kế thừa từ Policy object. Trong những Policy này, trường linearizableRead chỉ có nghĩa cho đối tượng mặc định và đối tượng kế thừa là BatchPolicy, được áp dụng cho tất cả các thao tác batch

b. InDoubt Errors:

Trường **InDoubt** được thêm vào để xác định sự khác biệt giữa một lệnh ghi *chắc chắn chưa được* thêm vào, hay *có thể* đã được thêm vào.

Đối với hầu hết cơ sở dữ liệu, những lỗi như TIMEOUT là không rõ ràng, ví dụ như khi client gửi dữ liệu tới server nhưng không nhận được phản hồi từ server thì không thể biết được liệu server đã ghi dữ liệu vào cơ sở dữ liệu hay chưa. Tuy nhiên, đối với Aerospike API, khi có lỗi **InDoubt** thì chắc chắn là dữ liệu chưa được ghi vào.

c. Data Unreachable Errors:

Đôi khi client không thể truy xuất được dữ liệu, lý do có thể là mạng hoặc các hạ tầng khác. Có 4 lỗi có thể xảy ra khi một cluster bị phân mảnh, hoặc xảy ra lỗi phần cứng dẫn đến dữ liệu không có sẵn (4 lỗi này có mã không giống nhau ở các phiên bản Client khác nhau, dưới đây là mã lỗi được sử dụng trong Java Client): **PARTITION_UNAVAILABLE**, **INVALID_NODE_ERROR**, **TIMEOUT** và **CONNECTION_ERROR.**

PARTITION_UNAVAILABLE: trả về khi server xác định được cluster không có dữ liệu chính xác.

INVALID_NODE_ERROR: được tạo ra trên client khi client không có một node cụ thể để gửi yêu cầu đi.

TIMEOUT và **CONNECTION_ERROR:** trả về trong trường hợp xảy ra phân vùng mạng.

5. Tính khả tuyến tính:

Tất cả những truy cập gần như là song song với nhau đều là tuần tự. Cơ chế quản lý hiệu quả của Aerospike đảm bảo cho từng bản ghi (record) và khiến một giao tác ngầm kiểm tra tình trạng của các server khác.

Nếu một lệnh ghi được thực hiện và được đọc ra bởi client, sẽ không có bất kì một phiên bản nào trước đó được đọc ra nữa. Khi chế độ 'global consistency ' này được bật lên, mỗi client liên kết với một cluster sẽ chỉ thấy cùng một giá trị cho một bản ghi tại một thời điểm.

Chế độ này yêu cầu thêm đồng bộ hóa trên từng lệnh đọc, điều này dẫn đến ảnh hưởng hiệu quả truy xuất.

6. Nhất quán theo phiên (Session Consistency)

Chế độ Nhất quán theo phiên là chế độ thực tế nhất trong chế độ Strong Consistency.

Không giống với Linearizable, Nhất quán theo phiên nhắm tới từng phân vùng client, ở đây là một cluster trên một hệ thống client riêng biệt, trừ khi phần bộ nhớ chia sẻ được dùng để chia sẻ trạng thái giữa cluster và các tiến trình.

Nhất quán theo phiên sẽ luôn đảm bảo các giao tác đọc hoặc ghi diễn ra riêng lẻ. Nhất quán theo phiên cung cấp nhất quán có dự đoán cho một phiên, và lưu lượng đọc tối đa mà vẫn đảm bảo độ trễ thấp nhất cho các giao tác đọc và ghi.

7. Các trường hợp ngoại lệ với đảm bảo nhất quán (Consistency Guarantees)

Các trường hợp dưới đây có thể dẫn tới một số vấn đề với tính nhất quán hoặc tính bền vững của dữ liệu.

a. Đồng hồ không liên tục(Clock discontinuity)

Một Clock discontinuity với một khoảng thời gian xấp xỉ hơn 27s có thể dẫn đến mất dữ liệu ghi.

Độ lệch của organic clock không phải là vấn đề, cơ chế quản lí thời gian heartbeat của từng cụm sẽ phát hiện ra độ lệch thời gian là 15s và thực hiện cảnh báo. Nếu độ lệch lớn hơn 20s, node sẽ từ chối lệnh ghi dữ liệu, trả về mã lỗi 'forbidden' từ đó ngăn cản vi phạm nhất quán.

Loại lỗi lệch thời gian này thường được chi làm 4 nguyên nhân chính:

1. Administrator Error, khi một người quản trị thực hiện cài đặt khóa tới tương lai hoặc trong quá khứ .

2. Các thành phần đồng bộ hóa thời gian hoạt động sai.

3. Chế độ ngủ đông của máy ảo.

4. Tiến trình Linux tạm dừng, hoặc Docker container tạm dừng.

b. Clock discontinuity do sự hợp nhất máy ảo trực tiếp.

Trong môi trường máy ảo hoặc môi trường ảo hóa vẫn có thể triển khai nhất quán an toàn.

Tuy nhiên, có 2 vấn đề cấu hình cụ thể là nguy hiểm.

Live migration: là quá trình di chuyển một máy ảo từ một máy vật lý sang một máy khác có thể gây ra nhiều nguy hiểm. Những đồng hồ thời gian di chuyển với sự gián đoạn và bộ đệm mạng có thể duy trì ở mức thấp và được áp dụng vào sau này.

Nếu việc di chuyển trực tiếp là dưới 27s, strong consistency có thể được duy trì. Tuy nhiên, cách an toàn hơn là dừng Aerospike an toàn, sau đó di chuyển máy ảo và khởi động lại Aerospike.

Trường hợp thứ hai liên quan tới tiến trình tạm dừng, có thể xảy ra trong môi trường container. Những tiến trình này gây ra mối nguy tương tự như di chuyển máy ảo trực tiếp, và không nên được thực hiện. Thật ra có rất ít trường hợp cần sử dụng những tiến trình gây lỗi này.

c. Xóa dữ liệu hết hạn và dữ liệu cần loại bỏ :

Việc xóa không nhất quán, bao gồm cả xóa dữ liệu cũ hoặc dữ liệu cần loại bỏ cũng gây mất tính nhất quán. Việc xóa đi những dữ liệu này khỏi cơ sở dữ liệu liên tục có thể gây vi phạm đảm bảo tính nhất quán. Trong những trường hợp này có thể thấy dữ liệu trả về. Vì lý do này, thường yêu cầu người dùng tắt tính năng 'eviction and expiration' trong cấu hình Strong Consistency.

d. Client retransmit

Nếu người dùng bật lại tính năng cho phép Aerospike client truyền lại lệnh ghi (retransmission), có thể sẽ có trường hợp gây vi phạm tính nhất quán. Đó là bởi vì một lệnh ghi bị quá thời gian sau đó truyền lại lần nữa, và có nguy cơ được ghi vào cơ sở dữ liệu nhiều lần.

Điều này gây ra nhiều lỗi:

1. dữ liệu được ghi vào nhiều lần gây mất tính nhất quán. Có thể tránh bằng việc sử dụng mẫu "read-modify-write" và chỉ định cho một lệnh cụ thể, cũng như là vô hiệu hóa retransmission

2. Tạo ra các mã lỗi sai. Ví dụ, một lệnh ghi có thể đã được ghi nhưng do lỗi đường truyền khiến cho client gửi lại lỗi ghi. Trong lần ghi thứ 2, bộ nhớ đầy, và tạo ra một lỗi 'InDoubt' thậm chí khi giao tác đã hoàn tất thành công. Loại lỗi này có thể được giải quyết bằng cách dùng mẫu 'read-modify-write'.

e. Secondary Index requests (scan and query) :

Việc thực thi một truy vấn có thể xảy ra hiện tượng dirty read. `Để đảm bảo về hiệu suất, Aerospike sẽ trả về dữ liệu ở trạng thái chưa được commit.

8. Ngoại lệ về tính bền vững dữ liệu (Durability Exceptions)

a. Lỗi lưu trữ phần cứng:

Khi lỗi xảy ra, phân vùng bộ nhớ bị lỗi sẽ được đánh dấu là **dead_partition** ở giao diện phía admin.

Lỗi này xảy ra khi tất cả các node trong toàn bộ roster sẵn sàng và đã kết nối, tuy nhiên một số phân vùng dữ liệu chưa sẵn sàng. Để cho phép một phân vùng chấp nhận đọc và ghi trở lại, admin cần ghi đè lỗi bằng cách thực thi một lệnh "revive", hoặc đưa máy chủ lên trực tuyến với dữ liệu bị thiếu.

b. Quản lý Roster không đúng đắn.

Giảm số roster bằng cách giảm hệ số nhân bản hoặc nhiều node tại cùng một thời điểm có thể gây ra mất dữ liệu. Quy trình thêm hoặc xóa bỏ an toàn dữ liệu khỏi cluster phải được tuân thủ.

c. Xóa một phần bộ nhớ (Partial storage erasure)

Trong trường hợp một số lượng sectors hoặc portion của ổ đĩa bị xóa bởi người vận hành, Aerospike sẽ không thể ghi nhận lỗi. Xóa một phần dữ liệu trên vùng nhớ nhân bản hoặc nhiều node có thể gây xóa bản ghi hoặc làm mất phát hiện lỗi.

d. Máy chủ khởi động lại đồng thời:

Mặc định, Aerospike ghi dữ liệu vào bộ nhớ đệm và xem xét các dữ liệu được ghi khi tất cả các node máy chủ yêu cầu đã nhận được dữ liệu và bỏ dữ liệu vào đúng hàng chờ.

Aerospike sẽ ngay lập tức sao lưu lại dữ liệu khi server xảy ra lỗi. Nếu quá trình sao lưu nhanh chóng, sẽ không có dữ liệu ghi bị mất.

Để ngăn chặn việc mất dữ liệu trong hàng đợi trong suốt quá trình lỗi liên tục, việc bật tính năng **commit-to-device** là cần thiết. Thuật toán mặc định của tính năng này khiến cho dữ liệu mất được hạn chế và vẫn đảm bảo hiệu suất cao. Tính năng này cũng có thể được bật trên từng namespace riêng biệt để đảm bảo mức hiệu suất cao hơn. Trong trường hợp một tệp hệ thống hay thiết bị có bộ đệm ghi được sử dụng như bộ nhớ lưu trữ, **commit-to-device** có thể không ngăn chặn bất kì loại mất mát dữ liệu nào

V. Cơ chế bảo mật:

Ở phiên bản doanh nghiệp của Aerospike có cơ chế kiểm soát truy cập hệ thống gồm nhiều chế độ xác thực. Những chế độ xác thực này có thể sử dụng riêng biệt hoặc đồng thời.

Admin có thể tạo người dùng cho cho server Aerospike, phân quyền, hoặc có thể dựa vào hệ thống LDPA bên ngoài.

Phiên bản doanh nghiệp của Aerospike hỗ trợ các loại xác thực như:

1. Xác thực dựa trên mật khẩu người dùng nội bộ được xác định trong máy chủ.

Tài khoản người dùng được tạo ra trong cơ sở dữ liệu Aerospike (người dùng nội bộ) có thể sử dụng xác thực dựa trên mật khẩu (Password-based authentication), hoặc xác thực dựa trên cơ sở hạ tầng khóa công khai (PKI: Public Key Infrastructure) dựa trên TLS

TLS: Transport layer security: là một hỗ trợ về luân chuyển dữ liệu giữa các kênh khác nhau.

Password-based authentication:

Quản trị viên tạo ra người dùng của một cụm Aerospike (Aerospike cluster) với tài khoản , mật khẩu và có thể có phân quyền cho người dùng. Đây cũng là chế độ xác thực mặc định cho clients.

2. Xác thực PKI cho người dùng nội bộ, sử dụng chứng chỉ TLS

Xác thực PKI là một chế độ xác thực khác cho người dùng nội bộ. Người quản trị sẽ tạo ra username và một password ngẫu nhiên cho người dùng. Nếu người dùng bị hạn chế truy cập dựa trên PKI mTLS, người quản trị sẽ không thông báo mật khẩu cho họ.

3. Xác thực bên ngoài bằng cách sử dụng một máy chủ LDPA

LDPA là một tính năng của Aerospike phiên bản dành cho doanh nghiệp.

Xác thực LDPA là một loại xác thực bên ngoài, trái ngược với xác thực cục bộ máy chủ cho tài khoản người dùng nội bộ được xác định trong máy chủ.

Một cụm dữ liệu Aerospike cho phép, hỗ trợ xác thực dựa trên một máy chủ LDPA bên ngoài, không sử dụng người dùng nội bộ. Đồng thời tài khoản người dùng nội bộ cũng không thể sử dụng xác thực LDPA.

VI. Cơ chế sao lưu và phục hồi dữ liệu:

Theo định nghĩa, Aerospike sao chép dữ liệu giữa các node của một cluster và giữa các trung tâm dữ liệu. Tuy nhiên, chính sách phổ biến trong các hoạt động của máy tính là bảo mật dữ liệu bằng cách sao lưu nó trong trường hợp xảy ra sự cố. Hầu hết các tổ chức đều có lịch trình và thủ tục để sao lưu. Các công cụ sao lưu và khôi phục Aerospike là asbackup và asrestore.

1. Sao lưu

Tiện ích 'asbackup' được sử dụng để sao lưu namespace, set hoặc partition từ một cluster vào bộ nhớ cục bộ. Theo mặc định, các bản sao lưu không được song song hóa. Nếu một bản ghi được tạo hoặc cập nhật sau khi phân vùng của nó được sao lưu, thì bản sao lưu sẽ không phản ánh điều đó.

a. Cách sử dụng

Hình thức cơ bản nhất của việc chạy sao lưu là chỉ cần chỉ định cluster để sao lưu '—host', namespace để sao lưu '—namespace', cũng như thư mục cục bộ cho các tệp sao lưu '—directory'. Giả sử rằng chúng ta có một cluster chứa một node có địa chỉ IP 1.2.3.4. Để sao lưu namespace 'test' trên cluster này vào thư mục 'backup_2015_08_24', sử dụng lệnh sau:

```
$ asbackup --host 1.2.3.4 --namespace test --directory backup_2015_08_24
```

b. Ước tính kích thước sao lưu

Khi truyền tùy chọn dòng lệnh '—estimate' sang asbackup (và bỏ qua --directory và --output-file), asbackup sẽ tạo một bản sao lưu thử nghiệm tạm thời gồm 10.000 bản ghi từ namespace. Sau đó, nó xuất ra, dựa trên các kích thước bản ghi được quan sát, ước tính kích thước trung bình của một bản ghi trong bản sao lưu. Để ước tính tổng kích thước của tệp sao lưu hoặc các tệp, hãy nhân kích thước này với số lượng bản ghi trong namespace và thêm 10% cho chỉ mục và chi phí.

Lưu ý: các bộ lọc cho mỗi bản ghi (`filter-exp`, `modified-after`, `modified-before`, `no-ttl-only`, `after-digest`, `partition-list`) và `node-list` không được tính đến trong ước tính và sử dụng các tùy chọn này sẽ không ảnh hưởng đến ước tính.

2. Khôi phục

Tiện ích 'asrestore' giúp khôi phục các bản sao lưu được tạo bằng 'asbackup'. Nếu namespace trên cluster đã chứa các bản ghi hiện có, chính sách ghi có thể định cấu hình sẽ xác định, bản ghi nào được ưu tiên hơn - các bản ghi trong namespace hoặc bản ghi từ bản sao lưu.

Nếu việc chèn bản ghi ban đầu không thành công, bộ lưu trữ sẽ thử chèn lại bản ghi 10 lần. Giữa các lần thử có 1 giây tạm dừng và lỗi được ghi vào nhật ký ở mức độ nghiêm trọng gỡ lỗi. Nếu bản ghi không được ghi sau 10 lần thử, quá trình khôi phục sẽ bị hủy bỏ với thông báo lỗi "Quá nhiều lỗi, đang bỏ cuộc". Các lỗi cụ thể khiến việc thử lại không được thực hiện là "tồn tại bản ghi" (khi tùy chọn --unique được sử dụng), "tạo không khớp" (trừ khi nếu tùy chọn - không tạo được sử dụng) và "tên người dùng hoặc mật khẩu không hợp lệ" .

Khi chạy với tùy chọn --directory, 'asrestore' mong đợi nhiều tệp sao lưu .asb trong thư mục directory. Ngoài ra, --input-file giúp asrestore đọc bản sao lưu hoàn chỉnh từ một tệp duy nhất đã cho.

a. Cách sử dụng

Hình thức cơ bản nhất của việc chạy khôi phục là chỉ cần chỉ định cụm để khôi phục '—host' và thư mục cục bộ chứa các tệp sao lưu '—directory'. Giả sử rằng chúng ta có một cluster chứa một node có địa chỉ IP 1.2.3.4. Để khôi phục bản sao lưu từ thư mục backup_2015_08_24, sử dụng lệnh sau:

```
$ asrestore --host 1.2.3.4 --directory backup_2015_08_24
```

Theo mặc định, bản sao lưu được khôi phục về namespace mà nó đã được lấy từ đó. Tùy chọn --namespace có thể được sử dụng để khôi phục lại một namespace khác. Giả sử rằng bản sao lưu ở trên được lấy từ namespace 'test' và muốn khôi phục nó về namespace 'prod', dùng lệnh sau:

```
$ asrestore --host 1.2.3.4 --directory backup_2015_08_24 --namespace test,prod
```

3. Sao lưu và phục hồi trên AWS (Amazon Web Services)

Cách truyền thống để sao lưu và phục hồi là sử dụng Aerospike Tools:

- Để sao lưu:
  - Sao lưu bình thường bằng 'asbackup'
  - Di chuyển tệp đến một vị trí an toàn.
- Để phục hồi: sử dụng 'asrestore' tải bản sao lưu lên hệ thống, khôi phục dữ liệu lại thành một cluster.

Tuy nhiên, cách này có một vài ưu/khuyết điểm cần lưu ý như:

Ưu điểm	Khuyết điểm
Dễ sử dụng	Tốn thời gian
Ít chi phí	Load server bổ sung khi chạy
Chứa dữ liệu SMD	Lãng phí cluster

Bên cạnh đó, Aerospike cũng có một cách khác cho việc sao lưu và phục hồi dữ liệu. Trong khi chạy, có thể sử dụng trực tiếp EBS hoặc thông qua các thiết bị ẩn, có thể tận dụng ảnh chụp nhanh của EBS. Điều này được thực hiện với các dữ liệu cấp khối, trong suốt đối với Aerospike.

- Để sao lưu: đơn giản chỉ cần chụp nhanh các khối dữ liệu trên EBS.
- Để phục hồi: Nếu các bản sao lưu được thực hiện thông qua ảnh chụp nhanh EBS, cần đưa các ảnh đó vào tập EBS. Khi cung cấp các ổ đĩa EBS thay thế, cần định cấu hình lại phiên bản thay thế mới chính xác như cách cấu hình phiên bản không thành công. Điều này có nghĩa là các ổ đĩa EBS được gắn ở cùng vị trí và cùng một số lượng ổ đĩa tạm thời được cung cấp.

Cách làm này cũng có các ưu/khuyết điểm như sau:

Ưu điểm	Khuyết điểm
Nhanh chóng	Tốn kém hơn
Ít bị ảnh hưởng	Cần có các thành phần bổ sung khác
Dựa trên từng node riêng lẻ	Không tính đến SMD

Giả sử phiên bản của bạn không thành công, nhưng bạn vẫn sử dụng EBS lâu dài. Trong trường hợp này, bạn có thể tách tập EBS khỏi bản sao bị lỗi và gắn lại vào bản sao thay thế.

1. Loại bỏ / Chấm dứt phiên bản không thành công, nhưng vẫn giữ volume EBS.
2. Triển khai Phiên bản mới, có cùng IP riêng với phiên bản cũ trong cùng một khu vực.
3. Đính kèm volume EBS từ Bước 1 vào phiên bản thay thế.
4. Khôi phục cấu hình cho node thay thế và khởi động lại node.

Khi node tham gia lại vào cluster, quá trình di chuyển sẽ xảy ra và dữ liệu sẽ được cân bằng lại.

VII. Hướng dẫn cài đặt và thao tác với AQL (Aerospike query language):

1. Cài đặt Aerospike Server (window)

a. Thiết lập Aerospike-server container

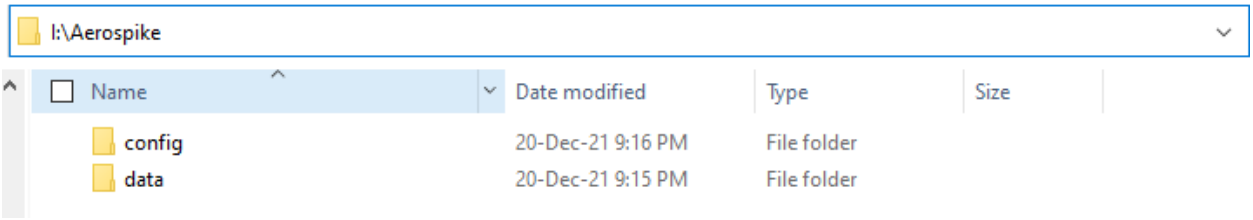
1. Tải Docker Desktop for Windows 10 Professionals Uers
2. Mở cmd
3. Chạy câu lệnh `docker run -tid --name aerospike -p 3000:3000 -p 3001:3001 -p 3002:3002 -p 3003:3003 aerospike/aerospike-server`
4. Sau khi đã setup thành công, khi chạy câu lệnh docker ps -a sẽ hiển thị:

```
I:\MyDu\Năm 3\HQT CSDL\Aerospike02>docker ps -a
CONTAINER ID   IMAGE                          COMMAND                  CREATED          STATUS          PORTS                                NAMES
2db22a6572e1   aerospike/aerospike-server   "/usr/bin/dumb-init …"   27 seconds ago   Up 19 seconds   0.0.0.0:3000-3003->3000-3003/tcp   aerospike
```

Chú ý: lưu lại container ID. Ở trường hợp này, container id = 2db22a6572e1.

b. Thiết lập Aerospike namespace:

1. Tạo một thư mục Aerospike mới với 2 thư mục con là data và config:



2. Mở cmd.
3. Chạy câu lệnh

`docker cp <containerId>:/etc/aerospike/aerospike.conf <path>`

Với `<containerId>` là container ID đã được tạo ở các bước trước, `<path>` là đường dẫn thư mục Aerospike đã được tạo ở bước 1

Ví dụ, trong trường hợp này là: `docker cp 2db22a6572e1:/etc/aerospike/aerospike.conf I:\Aerospike\config\`

Nếu việc cài đặt diễn ra thành công, trong folder config sẽ xuất hiện file `aerospike.conf`

4. Mở file aerospike.conf, chỉnh sửa đoạn thiết lập namespace:

Tùy chỉnh thay đổi tên namespace hoặc thêm 1 namespace khác. Ở đây, em sẽ sửa tên namespace thành MyApp. Ngoài ra, em cũng thay đổi kích thước tối đa của file là 1G.

```
namespace MyApp {
  replication-factor 2
  memory-size 1G
```

```
  default-ttl 30d # 5 days, use 0 to never expire/evict.
  nsup-period 120
  #     storage-engine memory

  # To use file storage backing, comment out the line above and use
the
  # following lines instead.
  storage-engine device {
        file /opt/aerospike/data/MyApp.dat
        filesize 1G
        data-in-memory true # Store data in memory in addition to
file.
  }
}
```

Lưu lại file.

c. Thiết lập lại server container để cập nhật những thay đổi trong configuration file:

1. Mở cmd.
2. Chạy câu lệnh `docker rm -f <container id>`
   Ví dụ, em sẽ chạy câu lệnh: `docker rm -f 2db22a6572e1`
3. Chạy câu lệnh `docker run -tid --name aerospike -p 3000:3000 -p 3001:3001 -p 3002:3002 -p 3003:3003 aerospike/aerospike-server`
4. Kiểm tra trạng thái server bằng câu lệnh: docker ps -a. Nếu container có trạng thái là Up, tức là đã thiết lập thành công:

```
CONTAINER ID   IMAGE                        COMMAND                  CREATED          STATUS          PORTS                                       NAMES
67a4d50e22b4   aerospike/aerospike-server   "/usr/bin/dumb-init …"   13 seconds ago   Up 9 seconds    0.0.0.0:3000-3003->3000-3003/tcp            aerospike
```

d. Kết nối tới Aerospike server bằng aql (Aerospike Query Language):

1. Chạy câu lệnh `docker inspect <container id>`
   Ví dụ: ở trường hợp của em, em sẽ chạy câu lệnh `docker inspect 67a4d50e22b4`
   Màn hình sẽ hiển thị thông tin configuration của server. Kéo đến cuối, ta sẽ thấy IPAddress của server:

```
"SandboxKey": "/var/run/docker/netns/594b4e58a12a",
"SecondaryIPAddresses": null,
"SecondaryIPv6Addresses": null,
"EndpointID": "09b1f8436bb9a2a514a3488cf86e33b1e0879b354ceefc25c5ccce5b6cd0cab0",
"Gateway": "172.17.0.1",
"GlobalIPv6Address": "",
"GlobalIPv6PrefixLen": 0,
"IPAddress": "172.17.0.2",
"IPPrefixLen": 16,
"IPv6Gateway": "",
"MacAddress": "02:42:ac:11:00:02",
```

2. Chạy câu lệnh `docker run -ti --name aerospike-aql --rm aerospike/aerospike-tools aql -h <IPAddress> aerospike/aerospike-server`

   Với <IPAddress> là địa chỉ IP của server. Trường hợp của em là 172.17.0.2, nên em sẽ chạy câu lệnh: `docker run -ti --name aerospike-aql --rm aerospike/aerospike-tools aql -h 172.17.0.2 aerospike/aerospike-server`

   Khi kết nối thành công, màn hình sẽ hiển thị:

```
Aerospike Query Client
Version 6.2.0
C Client Version 5.2.5
Copyright 2012-2021 Aerospike. All rights reserved.
```

3. Kiểm tra namespace bằng câu lệnh `show namespaces`:

```
aql> show namespaces
+------------+
| namespaces |
+------------+
| "test"     |
+------------+
[172.17.0.2:3000] 1 row in set (0.002 secs)

OK
```

e. Thử thêm và truy vấn dữ liệu:

```
aql> INSERT INTO test.set1 (PK, bins1) VALUES ('Name', 'Long My Du')
OK, 1 record affected.

aql> SELECT * FROM test.set1
+--------------+
| bins1        |
+--------------+
| "Long My Du" |
+--------------+
1 row in set (0.294 secs)

OK
```

2. Thao tác với Aerospike bằng AQL (Aerospike Query Language):

a. Xem thông tin các namespaces:

Câu lệnh: show namespaces

```
aql> show namespaces
+------------+
| namespaces |
+------------+
| "test"     |
+------------+
[172.17.0.2:3000] 1 row in set (0.002 secs)

OK
```

b. Xem thông tin các set có trong namespace:

Câu lệnh: show sets

```
aql> show sets
+------------------+--------+------------------+---------+------------------+--------------------+-------------+----------+------------------+-------------
----+-------------+------------+
| disable-eviction | ns     | index_populating | objects | stop-writes-count | set                | enable-index | sindexes | memory_data_bytes | device_data_b
ytes | truncate_lut | tombstones |
+------------------+--------+------------------+---------+------------------+--------------------+-------------+----------+------------------+-------------
----+-------------+------------+
| "false"          | "test" | "false"          | "21"    | "0"              | "users_images"     | "false"     | "0"      | "4368837"        | "4370064"
    | "0"          | "0"        |
| "false"          | "test" | "false"          | "4"     | "0"              | "users_information" | "false"     | "0"      | "497"            | "720"
    | "0"          | "0"        |
+------------------+--------+------------------+---------+------------------+--------------------+-------------+----------+------------------+-------------
----+-------------+------------+
[172.17.0.2:3000] 2 rows in set (0.001 secs)

OK
```

c. Thêm 1 record:

Cú pháp: **INSERT INTO** <ns>[.<**set**>] (PK, <bins>) **VALUES** (<**key**>, <**values**>)

Trong đó:

- ns: tên namespace
- set: tên set
- bins: tên các bins
- key: khóa chính của record
- values: giá trị các bins tương ứng

Ví dụ: Thêm một tài khoản mới (thêm 1 record vào namespace test, set users_information):

Insert into test.users_information (PK, fullname, password, email, country, image_count) VALUES (‘longmydu’, ‘Long My Du’, ‘passcuatoi’, ‘longmydu@gmail.com’, ‘Vietnam’, 0)

```
aql> insert into test.users_information (PK, fullname, password, email, country
, image_count) VALUES ('longmydu', 'Long My Du', 'passcuaMyDu', 'longmydu@gmail
.com', 'Vietnam', 0)
OK, 1 record affected.
```

d. Truy vấn:

Cú pháp: **SELECT** <bins> **FROM** <ns>[.<**set**>] WHERE PK =<key>

Trong đó:

- ns: tên namespace
- set: tên set
- bins: tên các bins (các trường)
- key: khóa chính của record

Ví dụ: Lấy ra thông tin tất cả tài khoản (xem tất cả record trong namespace test, set users_information):

Select * from test.users_information

```
aql> select * from test.users_information
+--------------+--------------+----------------------+-----------+------------
-+
| fullname     | password     | email                | country   | image_count
 |
+--------------+--------------+----------------------+-----------+------------
-+
| "Long My Du" | "passcuaMyDu" | "longmydu@gmail.com" | "Vietnam" | 0
 |
+--------------+--------------+----------------------+-----------+------------
-+
1 row in set (0.214 secs)

OK
```

e. Xóa 1 record:

Cú pháp: **DELETE FROM** <ns>[.<**set**>] **WHERE** PK=<**key**>

Trong đó:

- ns: tên namespace
- set: tên set
- key: khóa chính của record cần xóa

Ví dụ: Xóa tài khoản 'longmydu' (xóa 1 record trong namespace test, set users_information, có tên tài khoản cũng như key là 'longmydu'):

Delete from test.users_information where PK = 'longmydu'

```
aql> Delete from test.users_information where PK = 'longmydu'
OK, 1 record affected.
```

f. Sửa 1 bins trong record

Cú pháp: Giống với câu lệnh insert

Ví dụ: Thay đổi mật khẩu của tài khoản

Insert into test.users_information (PK, password) VALUES ('longmydu', 'passmoicuaMyDu')

```
aql> insert into test.users_information (PK, password) VALUES ('longmydu', 'pas
smoicuaMyDu')
OK, 1 record affected.
```

Pass đã được thay đổi:

```
aql> select * from test.users_information
+--------------+------------------+----------------------+-----------+---------
----+
| fullname     | password         | email                | country   | image_co
unt |
+--------------+------------------+----------------------+-----------+---------
----+
| "Long My Du" | "passmoicuaMyDu" | "longmydu@gmail.com" | "Vietnam" | 0
    |
+--------------+------------------+----------------------+-----------+---------
----+
1 row in set (0.178 secs)

OK
```

VIII. Đặc điểm hệ thống nên áp dụng Aerospike

Key-value Database:

Key-value Database cho phép lưu trữ dữ liệu với dung lượng lớn và khả năng truy vấn nhanh và bền bỉ hơn hẳn SQL database nhờ hệ thống khóa phụ và chính hỗ trợ truy vấn cùng với đó là dữ liệu được lưu trên Dram có tốc độ đọc cao và lưu trữ dữ liệu kiểu hướng đối tượng.

Ví dụ: các ứng dụng hệ thống lớn như: FaceBook, Twitter, Uber, Pinterest… cũng sử dụng cơ sở dữ liệu key-value.

Lưu trữ hồ sơ người dùng:

Các ứng dụng cần lưu trữ dữ liệu phục vụ cho việc marketing, tiếp thị, phân tích thị hiếu người tiêu dùng thì cơ sở dữ liệu key-value nói chung và Aerospike nói riêng là một sự lựa chọn tốt nhờ khả năng lưu trữ lớn và truy vấn nhanh, phục vụ dữ liệu cho học máy, phân tích dữ liệu.

Tính nhất quán cao:

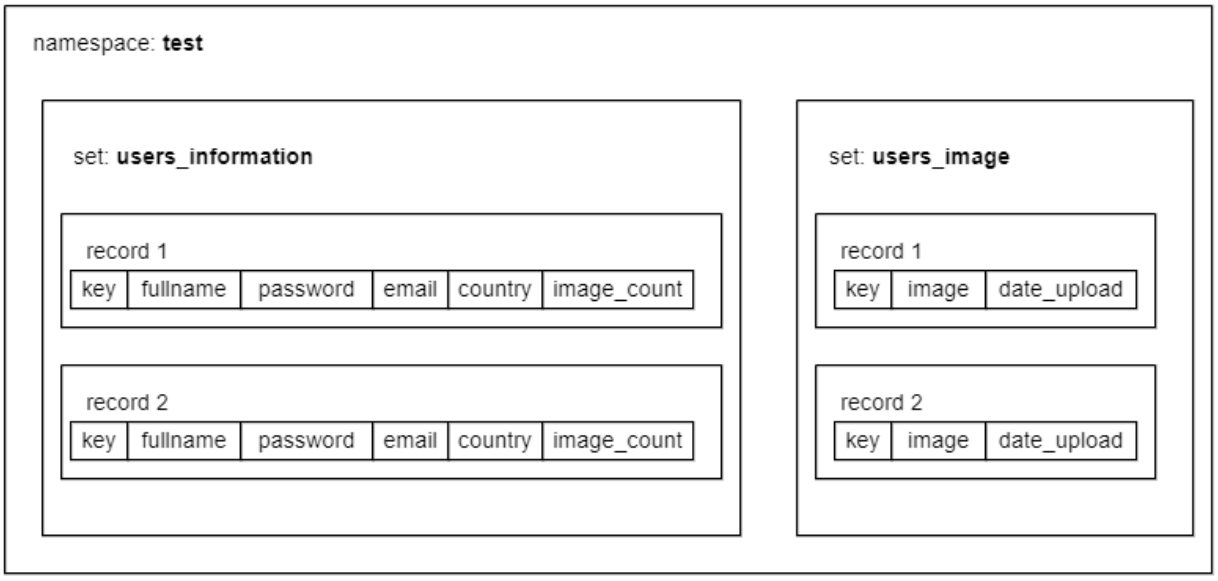
Các ứng dụng trong lĩnh vực tài chính là đối tượng chính của Aerospike hướng đến nhờ vào khả năng truy vấn nhanh với tốc độ truy vấn hàng triệu dòng một lần với độ trễ dưới mili giây và tính nhất quán cao trong lưu trữ dữ liệu để đảm bảo dữ liệu không bị sai sót trong truy vấn, đây là điều tối quan trọng trong ngân hàng và tài chính.

IX. Ứng dụng mô phỏng:

Ứng dụng lưu trữ hình ảnh.

Chức năng: đăng ký, đăng nhập, đăng xuất, lưu trữ hình, xem lại hình đã lưu trữ.

Data model:



X. Đánh giá các tính năng của Aerospike:

Tên tính năng	Mô tả	Ưu điểm	Nhược điểm
Kiến trúc đa bộ nhớ	Sử dụng song song nhiều bộ nhớ Flash trên một máy, server (SSD, PCIe, NVMe)	Tốc độ truy vấn, xử lý dữ liệu nhanh, hạn chế tắc nghẽn dữ liệu.	Tốn kém chi phí cho phần cứng.
Công cụ thời gian thực	Sử dụng cấu trúc dữ liệu phân vùng, đơn luồng	Hiệu suất truy vấn cao nhất có thể lên tới vài triệu transaction mỗi với độ trễ dưới milisecond	Phụ thuộc phần cứng thiết bị, phần cứng không tốt cũng làm giảm hiệu năng
Mô hình dữ liệu linh hoạt	Dữ liệu không cần có lược đồ, đơn vị lữu trữ cao nhất là namespace, chứa set, record,bin… vì không cần lược đồ nên set hay bin không cần cấu hình trước	Linh hoạt về cấu trúc dữ liệu lưu trữ làm tăng hiệu quả lưu trữ dữ liệu và bộ nhớ	Không có lược đồ, khiến tăng thời gian chạy để duy trì index do lược đồ lưu trữ tùy ý.
Các tính năng cho phiên bản doanh nghiệp.	Các tính năng bổ sung cho phiên bản doanh nghiệp nhằm lưu trữ dữ liệu lớn hơn với tốc độ truy vấn cao cùng mức bảo mật cao.	Triển khai hệ thống trên lượng thiết bị lớn. Hỗ trợ TLS để nâng cao hiệu quả giao tiếp giữa máy chủ và máy khách. Tăng tốc độ tái cân bằng để các hệ thống lớn giảm tài nguyên cho việc này.	Bản Enterprise cần phải trả phí cao để duy trì

XI. Tài liệu tham khảo:

Architecture Overview (aerospike.com)

Transaction:

https://aerospike.com/blog/developers-understanding-aerospike-transactions/

Sao lưu và phục hồi dữ liệu:

https://docs.aerospike.com/docs/tools/backup/index.html

https://docs.aerospike.com/docs/tools/backup/asbackup.html

https://docs.aerospike.com/docs/tools/backup/asrestore.html

https://docs.aerospike.com/docs/deploy_guides/aws/backup/index.html

Đánh giá Aerospike :

https://helpex.vn/article/danh-gia-nosql-aerospike-314-60a2b2799665e524df274d1a

Các tính năng của Aerospike:

https://aerospike.com/products/features/