Aerospike

Kiến trúc tổng quan:

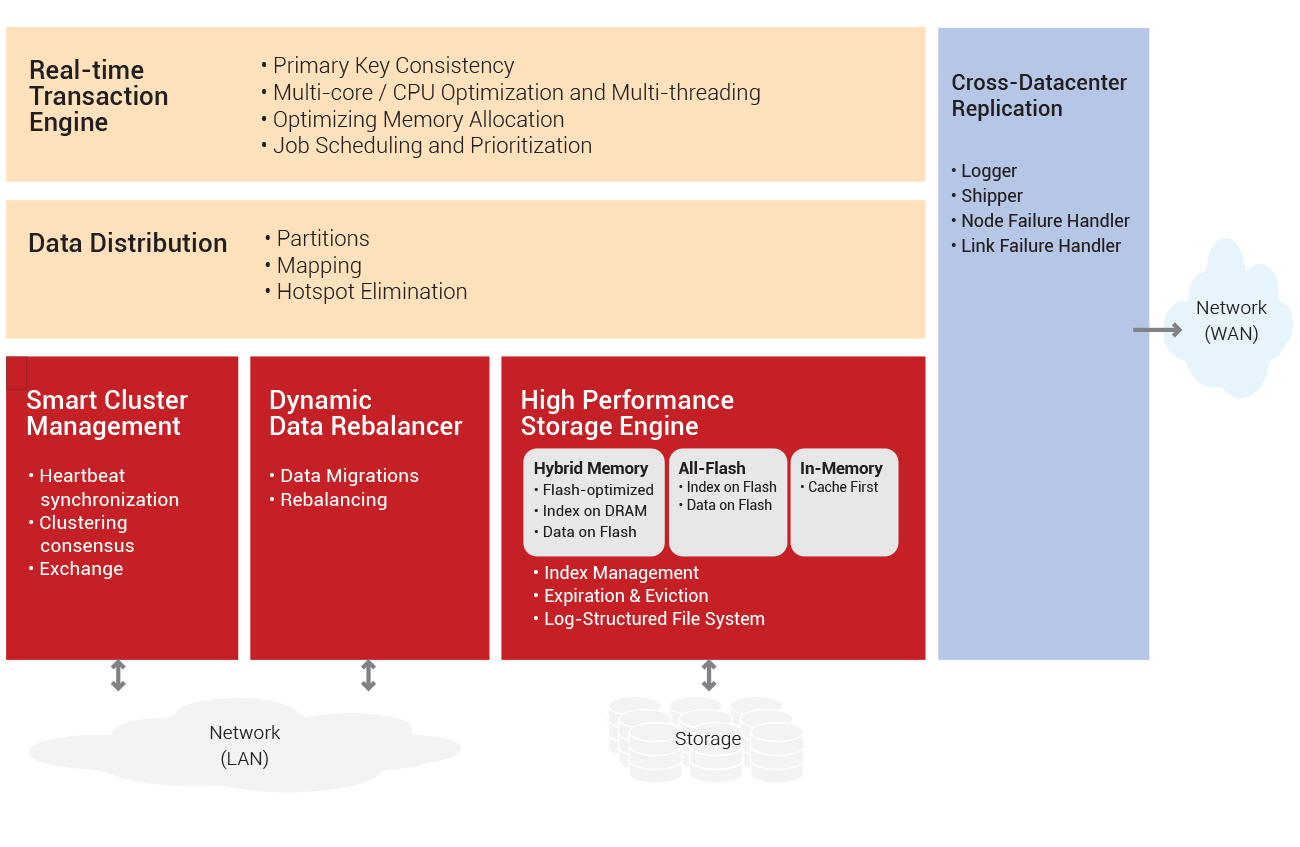
* Kiến trúc: client-server:

Ảnh có chứa văn bản, ảnh chụp màn hình, bãi đậu xe

Mô tả được tạo tự động

([Architecture Overview (aerospike.com)](https://docs.aerospike.com/docs/architecture/index.html))

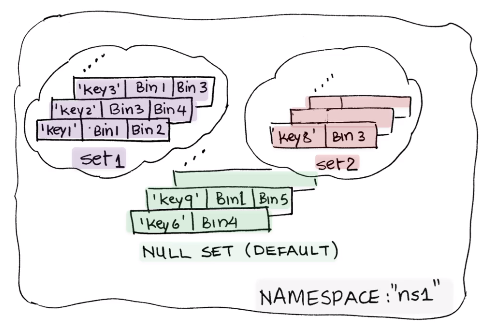
* Các layer bên client: bao gồm các thư viện mã nguồn mở, APIs, track nodes (các nút để theo dõi dữ liệu nằm ở đâu trong cụm).
* Các layer trong server:



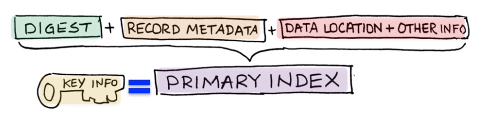
* + Data distribution: gồm có các module:
    - Cluster Management: Dùng thuật toán Paxos-based gossip-voting process để theo dõi dữ liệu, xác định những node nào là 1 phần của cluster; đồng bộ hóa “heartbeat” để quản lý việc giao tiếp giữa các cluster.
    - Data Rebalancer: Quản lý việc “di cư” dữ 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.

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.

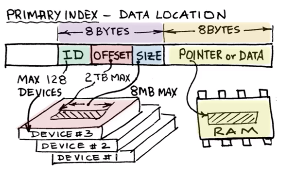


* Record metadata:
  + Last update time (LUT)
  + Generation (GEN): đếm số lần record được update
  + Time-To-Live (TTL): thời gian tồn tại, có thể là phút, ngày; tối đa là 10 năm hoặc có thể là mặc định live-for-ever.

Ảnh có chứa văn bản

Mô tả được tạo tự động

* 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:

Ảnh có chứa văn bản

Mô tả được tạo tự động

* 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.

Quản lý lưu trữ (Storage Management):

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.

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.

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).

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.

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.

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).

Quản lý cluster (Cluster management) (không hỉu ☹)

* Heartbeat: Các node trong cụm theo dõi nhau bằng “heartbeat”.

Thiết lập Aerospike (configuring Aerospike):

1. 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:

Ảnh có chứa văn bản

Mô tả được tạo tự động

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.

1. Thiết lập Namespace:

* Ảnh có chứa văn bản

  Mô tả được tạo tự động

Cài đặt Aerospike Server (window)

* 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:



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

* Thiết lập Aerospike namespace:

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

Ảnh có chứa văn bản

Mô tả được tạo tự động

1. Mở cmd.
2. 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

1. 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.

* 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

1. Chạy câu lệnh docker run -tid --name aerospike -p 3000:3000 -p 3001:3001 -p 3002:3002 -p 3003:3003 aerospike/aerospike-server
2. 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:



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

1. Chạy câu lẹnh 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:

Ảnh có chứa văn bản

Mô tả được tạo tự động

1. 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:

Ảnh có chứa văn bản

Mô tả được tạo tự động

1. Kiểm tra namespace bằng câu lệnh show namespaces:

Ảnh có chứa văn bản

Mô tả được tạo tự động

* Thử thêm và truy vấn dữ liệu:
* Ảnh có chứa văn bản

  Mô tả được tạo tự động

Cài đặt Aerospike Client node js:

Đối với window, cần phải có: [aerospike-client-nodejs/README\_WINDOWS.md at master · aerospike/aerospike-client-nodejs (github.com)](https://github.com/aerospike/aerospike-client-nodejs/blob/master/README_WINDOWS.md)

Đặc điểm

At scale: Phù hợp với tất cả quy mô, từ terabytes tới cả petabytes dữ liệu.

All the time: Tính khả dụng và tính nhất quán của dữ liệu cao.

Multi-site clustering (chia thành nhiều cụm địa điểm): Một node cũng như là một cụm, có thể được phân tán ở nhiều địa điểm. Một địa điểm có thể là 1 rack trong 1 datacenter, hoặc là cả datacenter, một availability zone trong cloud region, hay cả cloud region.

* Đảm bảo 100% tính khả dụng (availability), cho dù có 1 địa điểm xảy ra sự cố.
* Khi kết hợp với tính năng Strong Consistency, cơ sơ dữ liệu đảm bảo không mất mát dữ liệu với tất cả câu lệnh write.
* Có một khoảng thời gian bị trễ nhỏ khi truy vấn khoảng từ 2 đến 100 milliseconds, tùy thuộc vào khoảng cách giữa các địa điểm.

Strong Consistency: Là một cơ sở dữ liệu phân tán, Aerospike đảm bảo tất cả truy vấn đọc và ghi đều được thực thi theo lịch biểu khả tuần tự. Ngoài ra, Aerospike có hỗ trợ cài đặt giảm tính nhất quán của dữ liệu để đổi lại hiệu suất và tính khả dụng cao hơn nếu người dùng muốn.

Một vài thuật ngữ:

Single point of failure: A single point of failure (SPOF) is a part of a system that, if it [fails](https://en.wikipedia.org/wiki/Failure), will [stop the entire system from working](https://en.wikipedia.org/wiki/Cascading_failure).([Single point of failure - Wikipedia](https://en.wikipedia.org/wiki/Single_point_of_failure))

Cluster: The computer clustering approach usually (but not always) connects a number of readily available computing nodes (e.g. personal computers used as servers) via a fast local area network ([Computer cluster - Wikipedia](https://en.wikipedia.org/wiki/Computer_cluster#Basic_concepts))

Shared-nothing architecture: A shared-nothing architecture (SN) is a distributed computing architecture in which each update request is satisfied by a single node (processor/memory/storage unit) in a computer cluster. ([Shared-nothing architecture - Wikipedia](https://en.wikipedia.org/wiki/Shared-nothing_architecture))