CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM

Độc lập – Tự do – Hạnh phúc

BÁO CÁO KẾT QUẢ CÔNG VIỆC

Từ 04/06/2024 đến 10/06/2024

Họ và tên	:	Trần Danh Linh
Chức vụ	:	Thực tập sinh
Bộ phận công tác	:	Trung tâm Dịch vụ Truyền hình

HÀ NỘI, 6/2024

Mục lục

I. Sự khác nhau giữa HTTP và HTTP2

1. Nội dung

HTTP2 là tên gọi chính thức cho phiên bản tiếp theo của HTTP dựa trên công nghệ lõi từ SPDY được phát triển bởi Google. Trong đó, điểm khác biệt giữa HTTP và HTTP 2 là HTTP2 nhằm đơn giản hóa, cải thiện về tốc độ tải trang, tối ưu hóa tài nguyên và hỗ trợ các ứng dụng trên internet. Cụ thể là:

- Ghép kênh (Multiplexing):
    - HTTP truyền thông điệp bằng văn bản thuần túy và tải lần lượt tài nguyên, tức là nếu một tài nguyên không được tải thì nó sẽ chặn các tài nguyên đằng sau.
    - HTTP2 mã hóa thông điệp bằng mã nhị phân. Khi đó, dữ liệu sẽ gọn nhẹ và dễ dàng xử lý hơn -> HTTP2 có thể sử dụng một kết nối TCP để gửi nhiều luồng dữ liệu cùng một lúc, không gây ra tình trạng tắc nghẽn như HTTP và các luồng dữ liệu sẽ được đánh thứ tự để client có thể phân biệt được thứ tự các luồng dữ liệu.
- Server Push:
    - Đối với HTTP thì khi client gửi yêu cầu tài nguyên thì server mới gửi lại tài nguyên tương ứng -> Dẫn tới lãng phí thời gian khi client cần nhiều tài nguyên.
    - Trong khi đó HTTP2 có thể tải nhiều tài nguyên cùng lúc chỉ với một lần yêu cầu của client -> Giúp tiết kiệm thời gian và tránh cho client phải gửi nhiều yêu cầu.
- Nén header (Header compression):

- HTTP thông thường sẽ sử dụng gzip để nén header lại, tuy nhiên vẫn giữ header dưới dạng văn bản thuần túy -> Khi số lượng yêu cầu lớn vẫn sẽ dẫn tới vấn đề về hiệu năng.
- HTTP2 sử dụng HPACK để nén header. Tức là ở cả client và server đều sẽ lưu lại các header đã được sử dụng trước đó. Khi đó, nếu có một header được gửi tới thì sẽ thực hiện việc tìm kiếm và khôi phục lại header -> Gia tăng hiệu năng và tối ưu băng thông đáng kể.

2. Test hiệu năng trên cùng 1 API (Spring boot)

a. Source code

- API được viết bằng Spring boot, thực hiện việc tải một danh sách 10000 người dùng gồm ID, tên, tuổi và email được sinh ngẫu nhiên dưới dạng json.
- Sau đó, API được cấu hình theo HTTP và HTTP2 và được test lần lượt trên JMeter với khoảng 2000 người dùng (Tức là 500 threads chạy 4 lần).
- Link GitHub: Link

b. Kết quả

- File kết quả trong link GitHub ở trên dưới dạng file .csv

Label	# Samples	Average	Median	90% Line	95% Line	99% Line	Min	Max	Error %	Throughput	Received KB/sec	Sent KB/sec
HTTP Request	2000	516	142	1762	2211	2632	9	2851	0.00%	120.05523	79337.48	14.07
HTTP2 Request	2000	101	86	192	240	344	9	538	0.00%	128.92413	85198.48	15.11

Hình 1. Kết quả khi test HTTP và HTTP2

Từ kết quả trên ta thấy:

- Thời gian phản hồi trung bình của HTTP2 là 101 ms nhanh gấp 5.1 lần HTTP là 516 ms.
- Thời gian xử lý request ở giữa của HTTP2 là 86 ms nhanh gấp 1.65 lần HTTP là 142 ms.

- Thời gian phản hồi lâu nhất của HTTP2 là 538 ms nhanh gấp 5.3 lần HTTP là 2851 ms.

➔ Qua đó, ta có thể thấy HTTP2 có hiệu năng tốt hơn và thời gian tải trang ngắn hơn so với HTTP.

II. Elastic-cluster

1. Nhiệm vụ của các node trong cluster

Một cluster bao gồm các nút được kết nối với nhau, trong đó, mỗi nút có một vai trò đặc biệt thực hiện các chức năng khác nhau, bao gồm:

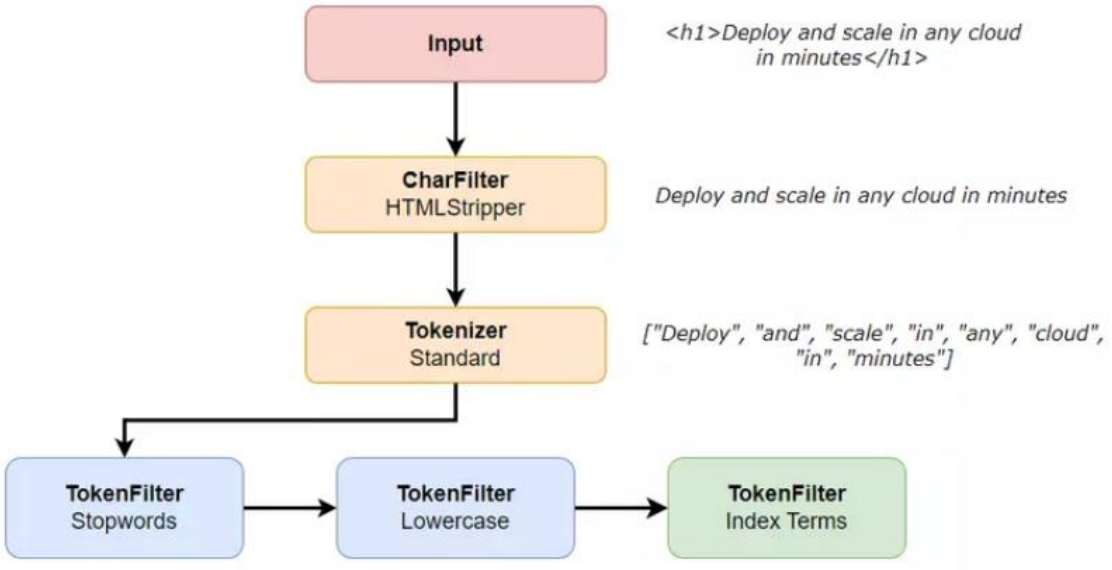
- Master Node: Chịu trách nhiệm thực hiện các tác vụ quản lý trên cluster như: thêm, xóa chỉ mục, theo dõi hoạt động của các node khác và quyết định phân mảnh nào sẽ được lưu trữ ở node nào.
- Data Node: Chịu trách nhiệm lưu trữ và quản lý dữ liệu trong cluster. Chúng sẽ thực hiện xử lý các yêu cầu chỉ mục, lưu trữ các dữ liệu trong phân mảnh và thực hiện truy vấn dữ liệu. Mỗi data node lưu trữ các dữ liệu trong các phân mảnh chính và bản sao, phân phối dữ liệu trên toàn cluster để có khả năng mở rộng và chịu lỗi. Ngoài ra, có các vai trò riêng biệt cho data node như: Data Content, Data Hot, Data Warm, Data Cold, Data Frozen.
- Ingest Node: Thực hiện việc tiền xử lý dữ liệu trước khi được đánh chỉ mục. Chúng giúp giảm tải các tác vụ xử lý cho data node, từ đó gia tang hiệu suất cho cluster.
- Coordinating-Only Node: Không lưu trữ dữ liệu hay tham gia vào quá trình bầu chọn master node. Nhiệm vụ của chúng là hoạt động như một proxy cho các yêu cầu của client, phân phối các yêu cầu tìm kiếm và lập chỉ mục đến cho các node thích hợp.

- Machine Learning Node: Thực hiện việc xử lý các yêu cầu Machine Learning API.
- Remote Eligible Node: Phục vụ cho các remote cluster, trong đó, các chỉ mục được sao chép liên cụm và được truy vấn liên cụm.
- Transform Node: Thực hiện xử lý các yêu cầu Transform API để chuyển đổi chỉ mục, cung cấp thông tin chi tiết và phân tích dữ liệu tóm tắt.

2. Cơ chế truy vấn/lưu dữ liệu

Elasticsearch sử dụng cơ chế **Inverted Index** (Là một chỉ mục cơ sở dữ liệu lưu trữ ánh xạ từ nội dung, chẳng hạn như từ hoặc số, đến các vị trí của nó trong bảng hoặc trong tài liệu hoặc tập hợp tài liệu) để lưu trữ và truy vấn dữ liệu.

a. Cơ chế lưu dữ liệu:



Hình 2. Mô tả quá trình Analyzer

- Dữ liệu được gửi đến Elasticsearch dưới dạng JSON document.
- Elasticsearch sử dụng quy trình Analyzer để phân tích văn bản thành các term (từ khóa) riêng lẻ.

- Tokenizer trong Analyzer sẽ tách văn bản thành các term dựa trên bộ quy tắc.
- Các term được lưu trữ trong Inverted Index cùng với thông tin về document chứa term đó (document ID, vị trí của term).

b. Cơ chế truy vấn dữ liệu:

1. Yêu cầu tìm kiếm được gửi đến Elasticsearch, bao gồm các term cần tìm kiếm.
2. Elasticsearch sử dụng Inverted Index để tìm kiếm các document chứa các term đó.
3. Elasticsearch tính toán relevance score (điểm liên quan) cho mỗi document dựa trên số lần xuất hiện của term, vị trí của term trong document.
4. Elasticsearch trả về danh sách các document phù hợp nhất với yêu cầu tìm kiếm, được sắp xếp theo relevance score.

c. Các loại truy vấn:

Elasticsearch hỗ trợ nhiều loại truy vấn khác nhau, bao gồm:

- Match query:
    - Là truy vấn chuẩn để thực hiện full text query. Bao gồm truy vấn kết hợp và truy vấn cụm từ hoặc gần đúng.
    - Match query trả về các document chứa ít nhất 1 trong các từ trong truy vấn.
- Match Phrase Query: Tìm kiếm các document chứa cụm từ trong truy vấn.
- Match Phrase Prefix Query: Tìm kiếm các document khớp với tiền tố trong truy vấn.
- Multi Match Query: Tương tự như Match Query nhưng cho phép tìm kiếm trên nhiều trường.
- Fuzzy Search:
    - Cho phép tìm kiếm gần đúng, dựa trên khoảng cách Levenshtein.

- Giúp người dùng dễ dàng tiếp cận được với nội dung hơn, khi mà họ có thể tìm thấy được những thứ cần thiết, ngay cả khi họ không nhớ được chính xác nội dung mình muốn tìm kiếm là gì.

3. Các mô hình triển khai trong thực tế

- Triển khai với Kubernetes: Triển khai trên đám mây với ưu điểm là dễ dàng thiết lập, nâng cấp, bảo mật trên Kubernetes.
- Triển khai với AWS: Amazon cung cấp dịch vụ Elasticsearch (Amazon ES) được đơn giản hóa trong việc triển khai, mở rộng quy mô và vận hành các cụm trong môi trường đám mây của Amazon Web Service (AWS).
- ELK Stack: Gồm 3 dự án phổ biến là Elasticsearch, Logstash và Kibana. ELK Stack đem lại khả năng tổng hợp nhật ký từ tất cả các hệ thống và ứng dụng của người dùng, phân tích những nhật ký này, hiển thị dữ liệu để giám sát ứng dụng và cơ sở hạ tầng, khắc phục sự cố nhanh hơn, phân tích bảo mật.

III. Redis-cluster

1. Nhiệm vụ các node trong mô hình cluster

- Mỗi Redis Cluster yêu cầu 2 cổng TCP được mở cho mỗi nút: một cổng cho clients và một cổng cho kết nối giữa các nút trong cụm.
- Redis Cluster sử dụng 16384 hash slot để phân bố dữ liệu trên các nút. Mỗi nút trong cụm chịu trách nhiệm cho một tập hợp con các hash slot.
- Redis Cluster sử dụng cấu hình master-slave để tăng khả năng sẵn sàng. Mỗi master node có thể có một hoặc nhiều slave node để sao chép dữ liệu.
- Các node trong cluster liên tục tương tác với nhau để đảm bảo các master node luôn hoạt động. Nếu một master node gặp sự cố thì một slave node sẽ được chọn để thay thế để đảm bảo tính liên tục của hệ thống.

- Quá trình lựa chọn slave node thay thế được thực hiện bởi một số lượng node nhất định trong cluster và có thể được cấu hình. Quá trình này sử dụng cơ chế "Gossiping", cho phép các nút trao đổi thông tin về trạng thái của nhau.

2. Cơ chế truy vấn/lưu dữ liệu

- Khi client muốn ghi hoặc đọc dữ liệu, nó sẽ gửi yêu cầu đến một nút bất kỳ trong cluster.
- Nút nhận yêu cầu sẽ sử dụng hàm băm để tính toán hash slot tương ứng với key của dữ liệu.
- Dữ liệu được lưu trữ trong một hash slot duy nhất, dựa trên kết quả của phép modulo giữa giá trị băm của key và tổng số hash slot.
- Nếu hash slot nằm trong dải quản lý của nút nhận yêu cầu, nút đó sẽ trực tiếp xử lý yêu cầu.
- Nếu hash slot thuộc về nút khác, nút nhận yêu cầu sẽ chuyển hướng client đến nút chính xác để truy cập dữ liệu.

3. Các mô hình triển khai trong thực tế

- Triển khai Redis trên Kubernetes: Mang lại khả năng mở rộng dễ dàng, tính sẵn sàng cao, hỗ trợ cân bằng tải, triển khai và quản lý đơn giản, giám sát và ghi lại nhật ký.
- Triển khai Redis thông qua AWS.