ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

ĐỒ ÁN MÔN HỌC:

XỬ LÝ DỮ LIỆU LỚN

ĐỀ TÀI:

XÂY DỰNG HỆ THỐNG RÚT TRÍCH TIN TỨC TỪ INTERNET VÀ THỐNG KÊ CÁC TỪ KHÓA CHÍNH

GVHD: PGS.TS Lê Đình Duy

Nhóm 16:

Nguyễn Châu Long – CH1802052

Nguyễn Tân Kim – CH1801030

Phạm Đức Duy – CH1801024

Nguyễn Chí Thương - CH1801015

TP. Hồ Chí Minh – 11/2019

MỤC LỤC

TÓM TẮT

Lý do chọn đề tài:

- Đề tài đáp ứng yêu cầu của môn học về xử lý dữ liệu lớn.
- Rút trích tin tức từ Internet và thống kê từ khoá là một bài toán hay của hệ thống bài toán trích rút từ khoá cho một văn bản. Ở mức cao hơn, nó là một bài toán con trong hệ thống trích xuất thông tin (Information Retrieval). Trong nhiều năm qua, bài toán này đã được đề cập, quan tâm nhiều ở các hội nghị quốc tế và các công ty lớn. Bài toán trích rút từ khoá từ trang web là việc trích rút từ khóa trong văn bản nội dung trang web. Đây cũng là vấn đề khá mới mẻ và được áp dụng trong rất nhiều lĩnh vực khác nhau như: Hỗ trợ tìm kiếm, hỗ trợ gợi ý người dùng....

Nội dung đề tài:

Chúng tôi đề xuất và xây dựng hệ thống rút tin tức từ Internet, sau đó thực hiện thống kê các từ khóa chính xuất hiện theo thời gian thực. Cơ sở dữ liệu căn bản của chúng tôi dùng là Postgresql, kết hợp với các tiện ích khác để xây dựng hệ thống lưu trữ phân tán và cơ sở dữ liệu chuỗi thời gian giúp đáp ứng được các yêu cầu cơ bản. Hệ thống được đề xuất tuy đơn giản nhưng có khả năng đáp ứng được các yêu cầu của dữ liệu lớn và dễ dàng mở rộng khi cần.

Kết quả thực hiện:

Nhóm đã thiết kế và xây dựng thành công hệ thống rút trích thông tin từ internet và thống kê các từ khóa quan trọng. Kết quả thực nghiệm ban đầu đã đạt yêu cầu đặt ra.

Chương 1: Giới Thiệu Tổng Quan

I. TỔNG QUAN ĐỀ TÀI

Rút tin (Web scraping) là quá trình xử lý, rút trích thông tin tự động từ các trang trên Internet để tổng hợp và làm giàu nguồn dữ liệu cho chính ứng dụng hiện tại. Một hệ thống rút tin đơn giản bao gồm frontier chứa các url cần được xử lý. Các url sẽ được xử lý tuần tự hoặc theo một nguyên tắc nào đó và rút các nội dung từ thẻ html. Các nội dung sẽ được thêm vào một cơ sở dữ liệu lưu trữ cùng với các url mới được tiếp tục thêm vào frontier.

CSDL quan hệ (SQL) được ra mắt từ rất sớm và đem lại nhiều lợi ích cho con người. Nhưng khi nhu cầu khai thác dữ liệu tăng cao, các CSDL phi quan hệ (NoSQL) khi đó ra đời như MapReduce (1), Google Bigtable (2), Cassandra (3), Mongodb (4),... đem lại nhiều tiện ích khiến chúng ta dần quên đi SQL trong các ứng dụng. Phải mất một thời gian, các CSDL quan hệ được mới tiếp tục được sử dụng nhiều trở lại bởi các dự án mã nguồn mở đã làm cho chúng ngày càng mềm dẻo.

Trong nhiều trường hợp, việc sử dụng các NoSQL là không cần thiết và thậm chí còn gây lãng phí bởi các CSDL quan hệ ngày nay đã đáp ứng được các dữ liệu phi cấu trúc. Postgresql sử dụng những kiểu dữ liệu cơ bản của SQL thông thường cộng thêm một số kiểu dữ liệu phi cấu trúc làm cho nó trở nên tiện dụng. Ngoài ra, các tiện ích mở rộng cũng như bản quyền Postgres đã giúp cho nó phát triển nhanh và trở thành CSDL được yêu thích nhất năm 2019 (5). Chính vì các điểm trên, chúng tôi đề xuất sử dụng Postgresql làm cơ sở dữ liệu lưu trữ thông tin cho ứng dụng.

Các điểm nhấn của đồ án cuối kỳ như sau:

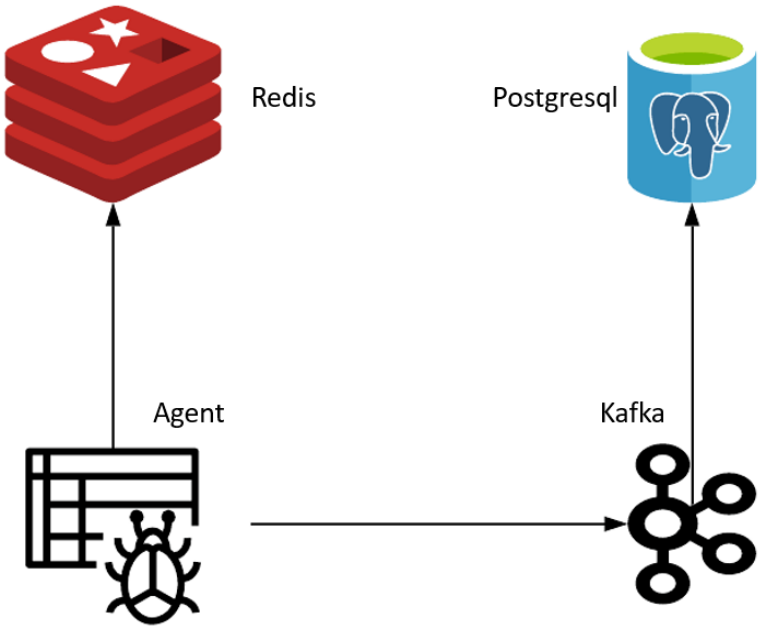
Xây dựng hệ thống rút tin tức và thống kê từ khóa đơn giản nhưng đáp ứng được luồng dữ liệu đến cực lớn và có khả năng mở rộng linh hoạt.

Đề xuất sử dụng cơ sở dữ liệu (CSDL) quan hệ và các tiện ích mở rộng để lưu trữ và thống kê dữ liệu thay cho các CSDL phi quan hệ (NoSQL).

Triển khai các dịch vụ trên docker.

II. CẤU TRÚC HỆ THỐNG

Chúng tôi đề xuất sử dụng Postgresql (Postgres), Redis, Kafka cùng một số tiện ích mở rộng của Postgres làm cơ sở xây dựng hệ thống. Hình bên dưới minh họa tóm tắt hệ thống chúng tôi xây dựng.



Hình 1. Sơ đồ cấu trúc hệ thống

Hệ thống trên gồm hai thành phần chính. Agent có nhiệm vụ thu thập các url từ những trang web cho trước. Các url được chuẩn hóa và lưu trữ dưới dạng mã hóa sha1 trong Redis một bản duy nhất (với key là mã băm của url). Với những url mới, chính đưa chúng vào Kafka và chờ chương trình rút nội dung, rút thực thể xử lý.

Các bài báo, từ khóa được chương trình rút nội dung xử lý và đưa vào Kafka trước khi vào Postgres và PipelineDB. Các công cụ để thực hiện việc này như Streamsets (6), KSQL (7), Pipeline kafka (8) nhưng chúng tôi không sử dụng ở đây vì chúng cồng kềnh. Thay vào đó, chúng tôi thực hiện vài dòng lệnh python cho việc stream dữ liệu từ Kafka về database. Các công cụ này sẽ được dùng khi cần mở rộng ứng dụng, sẵn sàng xử lý một luồng dữ liệu lớn. Sau khi được đưa vào cơ sở dữ liệu Postgres, các từ khóa sẽ được thống kê thông qua khung nhìn liên tục của PipelineDB được viết giống khung nhìn của SQL cơ bản.

Để phân tán dữ liệu, chúng tôi sử dụng Citusdata với master và các worker node. Bảng lưu trữ dữ liệu sẽ được tạo thành bảng phân tán thông qua lệnh

create_distributed_table của Citus. Vì phần cứng hạn chế nên chúng tôi chỉ thực hiện khởi tạo 1 master node và 3 worker tương ứng với master và slave 1 - 3.

Chúng tôi thực hiện cài đặt các phần mềm cơ sở dữ liệu trực tiếp trên các slave master và slave 1 - 3. Các dịch vụ, mã nguồn khác được chạy độc lập trên container ở slave 4. Các cài đặt tại slave 4 được tìm thấy trong mã nguồn của đồ án này. Để cài đặt chỉ cần chạy lệnh docker-compose up -d.

Chương 2: Xây Dựng Hệ Thống Và Thực Nghiệm

I. CÀI ĐẶT CƠ BẢN

Cài đặt docker và docker-compose trên slave 4

Cài đặt các gói cần thiết

sudo apt install apt-transport-https ca-certificates curl software-properties-common

Tiếp theo, cập nhật key của docker vào repository của ubuntu

curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -

Thêm vào apt source list

sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu bionic stable"

Cập nhật repository

sudo apt-get update

Cài đặt docker community edition

sudo apt install -y docker-ce

Cài đặt Postgresql tại master và các slave còn lại

Thêm key postgresql

wget -q https://www.postgresql.org/media/keys/ACCC4CF8.asc -O- | sudo apt-key add -

Thêm vào source list

echo "deb http://apt.postgresql.org/pub/repos/apt/ bionic-pgdg main" | sudo tee /etc/apt/sources.list.d/postgresql.list

Cập nhật repository

sudo apt-get update

Cài đặt postges 11

sudo apt-get install -y postgresql-11

Citusdata tại master và các slave còn lại

Thêm vào repository
curl https://install.citusdata.com/community/deb.sh | sudo bash

Cài Citus
sudo apt-get -y install postgresql-11-citus

Sau khi cài đặt xong Postgres và Citus ở các máy, tiếp tục thực hiện mở địa chỉ để các node Postgres giao tiếp với nhau. Thực hiện các bước sau trên tất cả các máy này:

Chỉnh listen
sudo vim /etc/postgresql/11/main/postgresql.conf

Tìm dòng listen_addresses='127.0.0.1 và thay bằng listen_addresses='*'

Tiếp theo thêm extension bằng cách thêm dòng
shared_preload_libraries = 'citus'

Lưu và tiếp tục sửa
sudo vim /etc/postgresql/11/main/pg_hba.conf

Thay
host all all 127.0.0.1/32 md5
thành
host all all 0.0.0.0/0 trust

Lưu và chạy lệnh khởi động lại pg server
service postgresql restart

Kiểm tra xem pg còn hoạt động hay không
service postgresql status
và cluster
pg_lsclusters

Citusdata tại master node
Chọn csdl và tạo extension bằng lệnh
CREATE EXTENSION citus;

Kết nối với các slave bằng lệnh

SELECT * from master_add_node(SLAVE_IP_ADDRESS_OR_NAME, BINDING_PORT);

ví dụ:

SELECT * from master_add_node('slave-1', 5432);

Cài PipelineDB tại master

Thêm apt repository

curl -s http://download.pipelinedb.com/apt.sh | sudo bash

*Cài đặ*t

sudo apt-get install pipelinedb-postgresql-11

Chỉnh preload của pg trong tập tin sudo vim /etc/postgresql/11/main/postgresql.conf

shared_preload_libraries = 'citus,pipelinedb'

và

max_worker_processes = 128

khởi động lại Postgres server bằng lệnh service postgresql restart

Cài kafka-pipeline

Thực hiện lần lượt các lệnh sau:

sudo apt-get install git gcc g++ zlib1g-dev

git clone https://github.com/edenhill/librdkafka.git

git clone https://github.com/pipelinedb/pipeline_kafka.git

cd librdkafka

./configure --prefix=/usr

make && sudo make install

cd pipeline_kafka

./configure

make && sudo make install

Cuối cùng, chỉnh preload trong postgresql.conf và khởi động lại server

shared_preload_libraries = 'citus,pipelinedb,kafka-pipeline'

Khởi động các container app tại slave 4

Di chuyển đến thư mục project và gõ lệnh:

sudo docker-compose up –d

II.THỰC NGHIỆM

Thực hiện tạo bảng Postgres và distributed bằng lệnh

```
CREATE TABLE articles (
   id PRIMARY KEY,
   data jsonb,
   created_time timestamp without timezone
)
SELECT create_distributed_table('articles', 'id')
```

hoặc dùng ORM mapping (như mã nguồn)

Chúng tôi hiện thực chương trình bằng python3.6 được chạy trên ubuntu 18.04. Chúng tôi thực hiện kết nối với Postgresql bằng sqlalchemy, thực hiện các thao tác với Kafka bằng kafka-python.

Các bài báo được rút từ 150 nguồn tiếng Anh (xem pages.txt). Chúng tôi thực hiện rút tiêu đề, tóm tắt, nội dung, ngày đăng, tác giả từ bài báo thông qua thư viện newspaper3k. Từ tiêu đề và nội dung, chúng tôi tiếp tục thực hiện rút các thực thể, hay còn gọi là từ khóa bằng thư viện nltk. Các từ khóa sẽ được thống kê số lượng xuất hiện mỗi ngày bằng khung nhìn liên tục của PipelineDB từ bảng stream. Khung nhìn liên tục được viết bằng lệnh sql:

```
CREATE VIEW kw_stream WITH (action=materialize) AS
SELECT kw_stream.keyword, DATE(kw_stream.time) AS date, COUNT(DATE(kw_stream.time)) AS total
FROM public.kw_stream
GROUP BY kw_stream.keyword, DATE(kw_stream.time)
```

Mã nguồn đồ án nằm trong thư mục src gồm các tập tin:

arguments.py chứa các tham số khi chạy chương trình như sau:

--kafka_host địa chỉ host và port để kết nối đến kafka server. Ví dụ: 127.0.0.1:9092

--kafka_link_topic topic chứa các liên kết, dùng như hàng đợi chuẩn bị cho rút nội dung

--kafka_keyword_topic topic chứa các keyword sau khi đã được rút nội dung. PipelineDB sẽ liên kết đến topic này và tự động stream dữ liệu và bảng chuẩn bị cho thống kê

--kafka_default_group nhóm làm việc của kafka consumer, dùng để theo dõi những offset đã duyệt qua để tiếp tục chạy phòng trường hợp consumer bị ngắt

--redis_host địa chỉ host của redis database

--redis_port port redis

--redis_db số thứ tự của redis database. Mặc định ở đây là 1

--redis_password mật khẩu của redis server

--pg_host địa chỉ host của postgresql

--pg_port port postgres

--pg_user user được quyền truy cập vào database

--pg_password password của user

--pg_db database cần được truy cập

--pg_relation tên bảng chứa dữ liệu tin được rút về

helper.py hiện thực các phương thức như kết nối database, database model, rút thực thể (keyword) từ text, mã hóa url, chuẩn hóa url

logger.py ghi log của chương trình

visitor.py lấy các link nội dung từ url cho trước và gởi vào Kafka

scraper.py rút nội dung từ các link trong Kafka topic. Sau khi thực hiện xong, nội dung bài báo sẽ được gởi thẳng vào Postgresql. Các keywords được gởi vào PipelineDB thông qua Kafka

*** Ví dụ chạy xem trong tập tin .sh

Chương 3: Kết Quả Thực Nghiệm

Sau khi cài đặt thành công citus cluster, kiểm tra các worker nodes và thêm dữ liệu vào node có sẵn. Lệnh thêm dữ liệu sử dụng như sql cơ bản:

```
data=# select * from master_get_active_worker_nodes();
  node_name   | node_port
--------------+-----------
 10.255.255.7 |      5433
 10.255.255.9 |      5432
 10.255.255.8 |      5432
(3 rows)
```

```
data=# insert into articles(id, data) values(1, '{"title": "This is title"}');
INSERT 0 1
data=# insert into articles(id, data) values(2, '{"title": "This is another title"}');
INSERT 0 1
data=#
```

Master node có địa chỉ là 10.255.255.6.

Hàm master_get_active_worker_nodes() trả về danh sách các worker. Thử thêm một record có id trùng ta thu được kết quả "duplicate" trên node thứ 1:

```
SET
data=# insert into articles(id, data) values(2, '{"title": "Try with 3 replicates"}');
ERROR:  duplicate key value violates unique constraint "articles_pkey_102192"
DETAIL:  Key (id)=(2) already exists.
CONTEXT:  while executing command on 10.255.255.7:5433
```

Vào docker redis bằng lệnh docker exec -it redis bash và gõ redis-cli. Kiểm tra các key được lưu trong database 0:

```
root@017683356de4:/opt/redis#
root@017683356de4:/opt/redis# redis-cli
127.0.0.1:6379> SCAN 0 COUNT 100
1) "909312"
2)   1) "478fd439a9be4839c9a607aaf7e914f6"
     2) "c7c5c9cdd8a14a39232835ba09afa066"
     3) "f51424fa149f1c654be38016ec4c17db"
     4) "ed473c18752ae1b9c53ee3c61eb2c342"
     5) "a5fd93a3c751fc9780c642e1a1b94293"
     6) "23dc5b531822759d5e1a44cda05c8c1b"
     7) "8c01c8b23dcde54ae660bf7bd430f035"
     8) "2d90a347ef307c88a7b2ac4b5fb5479f"
     9) "a4b9bb53152cd446f96f71c6a841b882"
    10) "810638a9f08bceb1147c0e1bec68c3c7"
    11) "40b3fce7e92cdf6f205c191f5f73f2a7"
    12) "a99bbde3c2763a4da795e80a21faa0b3"
    13) "ca7e847a864abdab7fdb63a9bf88bf54"
    14) "42896c780ac294fe09b145c9c184bc59"
    15) "598d4e72a331272fdcde14b6c64605ec"
    16) "91c11264fe3b9592ab53f85e8296dab0"
    17) "8dd642fba9796146912e3d187a925c8d"
    18) "5106da0ed3ff0fd470789e0078296b75"
    19) "7c26bac479192dd5e2cb955f6b929065"
    20) "7bdd0870046d320602103b89757705cd"
    21) "88a7a1b0343c147dedeb4db42fb97e7c"
    22) "83760e179b5c8986bfeb6c6a6038125b"
    23) "abe8cd0163e2fc56b8df3f65e09ce2f8"
    24) "5a60c92f96a34e4769932a9bd62f964b"
    25) "67bb2785312dc99df67e8013898227ff"
    26) "d1cbb4c6cd1a97d002877ff5d41fbda9"
    27) "8c02ad031703743f6b1bd93aff4cf756"
    28) "882adbc9ecf6932d68b40f1e0184af56"
    29) "8a94482f3607d94dec9b8bfe7ae55e44"
```

Khung nhìn liên tục của PipelineDB được sử dụng như một view:

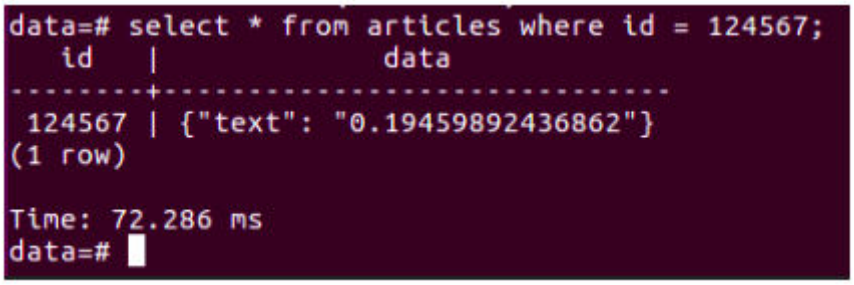
```
data=# select * from keyword_stats order by total desc limit 10;
  keyword   |    date    | total
------------+------------+-------
 VOV        | 2019-11-14 |    25
 ĐÀI TIẾNG  | 2019-11-14 |    20
 Việt Nam   | 2019-11-14 |    17
 Những      | 2019-11-14 |    15
 Các        | 2019-11-14 |    14
 UBND       | 2019-11-14 |    11
 Quốc       | 2019-11-14 |    11
 Chính      | 2019-11-14 |    10
 Với        | 2019-11-14 |    10
 Trung Quốc | 2019-11-14 |    10
(10 rows)

data=#
```

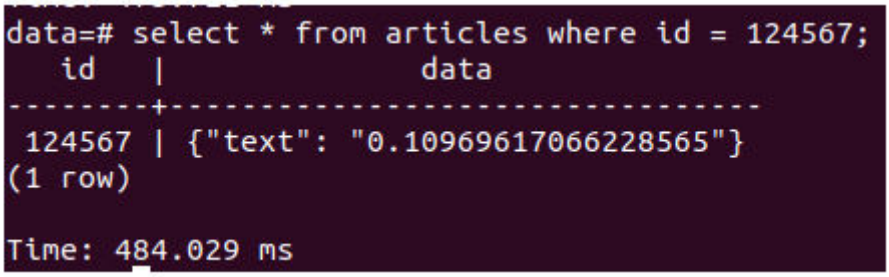
Thử tốc độ với 3 workers, tạo data giả gồm 9 triệu dòng và đếm:

INSERT INTO Articles(id, data)

SELECT generate_series(3, 9000000) as id, json_build_object('text', random()::text) as data;

Bật timing postgresql và kiểm tra tốc độ khi query theo id. Ảnh bên dưới thực hiện cùng query trên vlab cluster (trên) và trên máy cá nhân (dưới) với cùng số dòng dữ liệu, cùng index. Máy cá nhân sử dụng CPU core i7, ram 16gb.





TÀI LIỆU THAM KHẢO

Các trích dẫn:
[1]. MapReduce, https://ai.google/research/pubs/pub62
[2]. Bigtable, https://ai.google/research/pubs/pub27898
[3]. Cassandra, https://en.wikipedia.org/wiki/Apache_Cassandra
[4]. MongoDB, https://en.wikipedia.org/wiki/MongoDB
[5]. Stackoverflow survey 2019, https://insights.stackoverflow.com/survey/2019
[6]. Streamsets, https://streamsets.com/
[7]. KSQL, https://www.confluent.io/product/ksql/
[8]. Pipeline - Kafka, https://github.com/pipelinedb/pipeline_kafka

Các tài liệu hướng dẫn cài đặt:
[9].https://www.digitalocean.com/community/tutorials/how-to-install-and-use-docker-on-ubuntu-18-04
[10]. Docker Kafka và Zookeeper: https://github.com/wurstmeister/kafka-docker
[11]. Docker Redis: https://hub.docker.com/_/redis
[12].https://www.itzgeek.com/how-tos/linux/ubuntu-how-tos/how-to-install-postgresql-10-on-ubuntu-18-04-lts.html
[13]. Citus data: http://docs.citusdata.com/
[14]. PipelineDB: http://docs.pipelinedb.com/