به نام خدا

پروژه کارشناسی: بررسی موازی سازی و کلاسترینگ در پایگاه داده کسندرا

استاد راهنما: دکتر مدرسی

داور: استاد صفری

نویسنده : حامد گرجی آرا

مقدمه:

طی این پروژه، ابتدا به بررسی ساختار ذخیره داده ها در پایگاه داده کسندرا ، یکی از پرکاربرد ترین پایگاه داده های NoSQL، پرداخته شد. پس از آن روی یک نود (کامپیوتر) ورژن ای از این برنامه نصب شد و توسط برنامه تست میزان بهره وری پایگاه داده مورد بررسی قرار گرفت. برنامه ی تست از طریق سایت زیر قابل دسترسی است:

https://github.com/hamed911/bsproj

این برنامه به نحوی نوشته شده که داده های تست به میزان دلخواه و قابل کانفیگ را برای تست performance پایگاه داده تولید می کند.

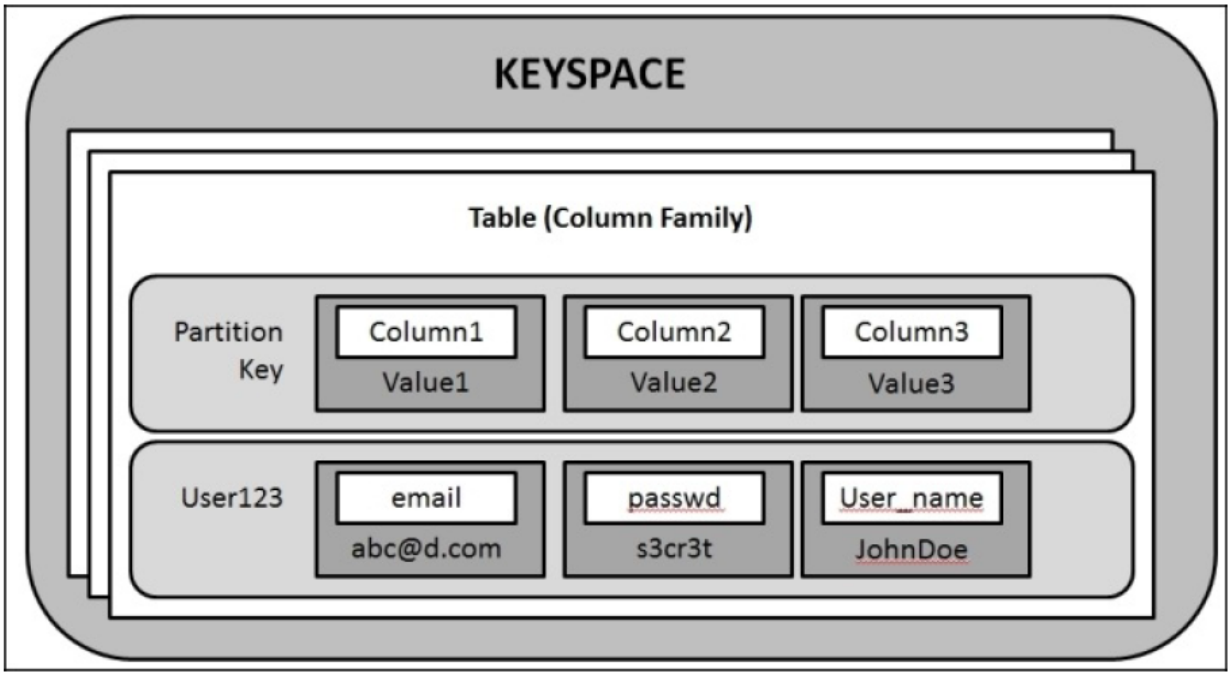
پس از بررسی performance پایگاه داده برای یک node، یک کلاستر از این برنامه ایجاد شد و مجددا تست performance روی کلاستر صورت گرفت و نتایج آن نسبت به یک نود مورد بررسی قرار گرفت.

معرفی کسندرا:

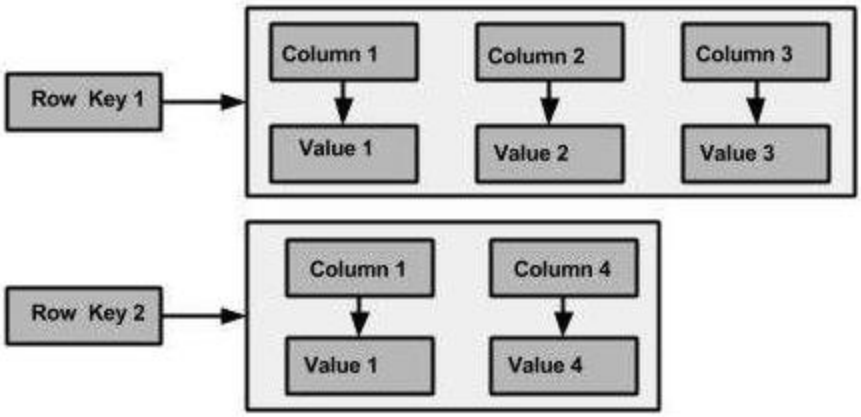
کسندرا یکی از معروف ترین و پرکاربرد ترین پایگاه داده های NoSQL است که اولین بار توسط شرکت facebook نوشته شد و هم اکنون در شرکت های بزرگی نظیر cisco، github ، adobe و ... استفاده میشود. ویژگی های کسندرا عبارت است از:
1- scalability بالا (تقریبا به صورت خطی) 2- performance بالا 3- بدون خرابی (No Single Point of Failure)

نحوه ذخیره داده:

کسندرا بر خلاف دیتابیس های relational، داده ها را به صورت زیر ذخیره میکند:



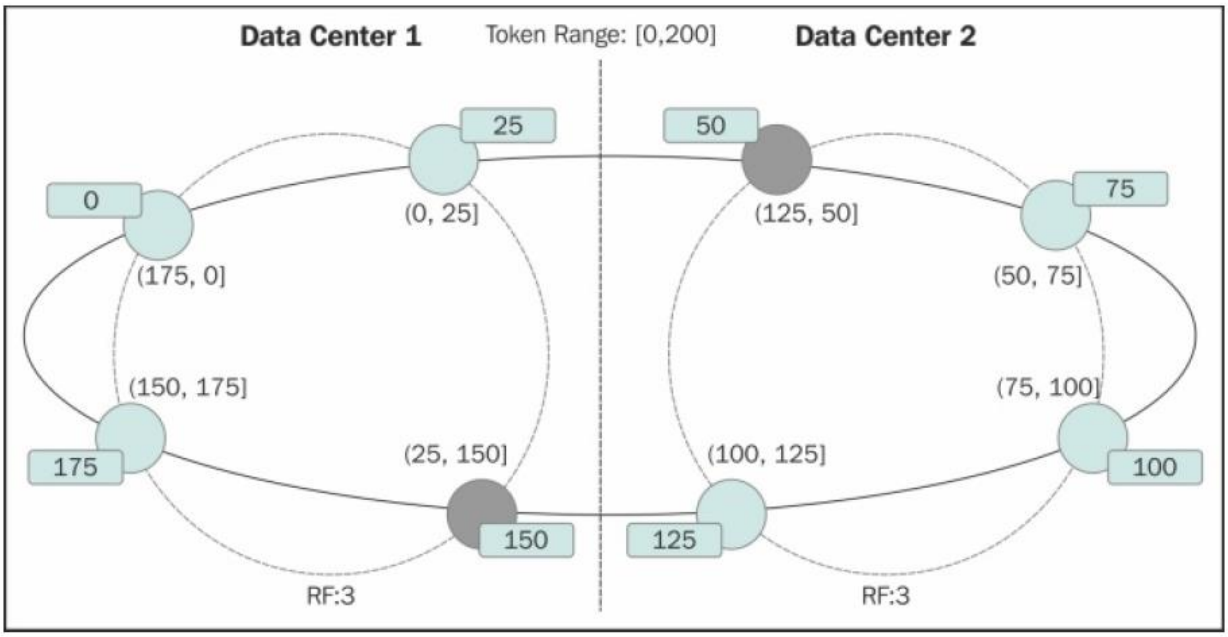
1. Keyspace : یک container که Column family ها را در خود نگه میدارد. ویژگی هایی نظیر Replication Factor را تعیین می کند (این متغییر معین میکند از هر row، چند کپی روی نود های مختلف داشته باشیم)
2. Column Family : مانند جدول ها در پایگاه داده های جدولی است که همانطور که در شکل مشخص است تعدادی row را در خود ذخیره میکند
3. Row (Partition): هر row شامل تعدادی column با ساختار key-value مانند است



در ابتدا، کسندرا داده ها را با یک فرمت خاص روی رم ذخیره میکند، و پس از پر شدن، آن ها را از رم به هارد منتقل میکند. این انتقال باعث سربار روی سیستم میشود ولی به گونه ای در خلال برنامه انجام میشود که تاثیر performance ای چندانی نداشته باشد.

ساختار کلاستر:

در هر کلاستر، یک یا تعدادی نود master وجود دارد که کسندرا به آن ها seed میگوید. seed ها وظیفه تقسیم query ها به بقیه نود ها و مدیریت query ها و پاسخگویی به کلاینت را بر عهده دارند.



برا مثال در شکل بالا node ها 25 ، 50، 125، 150 نود های seed هستند که مسئول ارتباط با خارج کلاستر و بقیه نود های داخل هستند. در هر کلاستر کسندرا، ابتدا نود های seed بالا می آیند و پس از آن بقیه نود ها به کلاستر اضافه میشوند. Node های جدید پس از بالا آمدن، خود را به seed ها معرفی میکنند و اطلاعات بقیه نود ها رو از آن ها میگیرند (در این مرحله یک سری تبادل اطلاعات بین نود ها انجام میشود تا داده ها به صورت یکنواخت بین هر نود پخش شود و بر اساس کانفیگ صورت گرفته، کپی از داده ها نگه داشته شود)
کسندرا با استفاده از یک hash function قوی، سعی در پخش کردن یکنواخت داده ها بین نود ها میکند. با این حال در ورژن های جدید کسندرا از یک مفهوم جدید به اسم virtual node برای هر نود استفاده شده است که باعث پخش یکنواخت load در هر نود می شود. نحوه کانفیگ کردن virtual node نیز در ادامه اشاره خواهد شد.

راه اندازی پایگاه داده روی کلاستر

با توجه به این که کسندرا به زبان جاوا نوشته شده است، برای راه اندازی آن نیاز به نصب جاوا بر روی node ها است. در ادامه به تفصیل به بررسی مراحل نصب و کانفیگ کلاستر خواهیم پرداخت.

بخش اول. نصب جاوا

برای راه اندازی کسندرا، به جاوا نیاز است. راه های مختلفی برای این کار وجود دارد، ولی با توجه به تحریم شدن ایران توسط جاوا و ارائه ندادن سرویس توسط این کمپانی، بهترین راه دانلود فایل tar. از سایت oracle میباشد. پس از دانلود فایل، دستور های زیر را به ترتیب در ترمینال وارد کنید:

1. sudo apt-get update
2. sudo mkdir /usr/java
3. sudo tar xvzf jdk-8u5-linux-i586.tar.gz -C /usr/java (check file name)
4. JAVA_HOME=/usr/java/jdk1.8.0_05/ (check folder name)
5. sudo update-alternatives --install /usr/bin/java java ${JAVA_HOME%*/}/bin/java 20000
6. sudo update-alternatives --install /usr/bin/javac javac ${JAVA_HOME%*/}/bin/javac 20000
7. Java -version

برای مطالعه بیشتر به لینک زیر مراجعه کنید:

1. http://askubuntu.com/questions/764849/how-can-i-install-jdk-8u91-linux-x64-tar-gz-on-ubuntu

بخش دوم. نصب کسندرا

پس از نصب جاوا، حالا میتوانیم اقدام به نصب کسندرا بر روی هر نود بکنیم. راه های متفاوتی برای این کار وجود دارد، ولی با توجه به این که به آی پی های ایران سرویس ارائه داده نمیشود، بهتر است فایل tar. را از سایت مربوطه دانلود کنید و با وارد کردن دستور های زیر در ترمینال آن را نصب کنید:

1. tar -xvzf apache-cassandra-1.2.4-bin.tar.gz
2. mv apache-cassandra-1.2.4 ~/cassandra
3. sudo mkdir /var/lib/cassandra
4. sudo mkdir /var/log/cassandra
5. sudo chown -R $USER:$GROUP /var/lib/cassandra
6. sudo chown -R $USER:$GROUP /var/log/cassandra
7. export CASSANDRA_HOME=~/cassandra
8. export PATH=$PATH:$CASSANDRA_HOME/bin

همانطور که اشاره شد، راه های دیگری نیز برای این کار وجود دارد، که با مراجعه به لینک های زیر میتوانید از آن ها اطلاع پیدا کنید:

1. https://www.digitalocean.com/community/tutorials/how-to-install-cassandra-and-run-a-single-node-cluster-on-a-ubuntu-vps
2. http://cassandra.apache.org/download/
3. http://idroot.net/tutorials/how-to-install-apache-cassandra-on-ubuntu-14-04/

بخش سوم. راه اندازی و تست پایگاه داده

چنانچه با این پایگاه داده آشنایی ندارید توصیه میشود که این مرحله رو انجام بدید در غیر این صورت به مرحله بعد بروید. برای تست کردن پایگاه داده و مطمئن شدن از کارکرد آن، ابتدا پایگاه داده را اجرا میکنیم و در محیط cqlsh جست و جو ها را انجام میدهیم:

۱. ابتدا دیتا بیس را با دستور زیر (اگر اجرا نشده بود) اجرا میکنیم:

```
sudo sh ~/cassandra/bin/cassandra
```

2. پس از آن وارد محیط اجرای کویری میشویم:

```
sudo sh ~/cassandra/bin/cqlsh
```

3. اگر برای اولین بار پایگاه داده را راه اندازی میکنیدُ برای ساختن entity های مختلف (برای فهم بیشتر ابتدا ساختار ذخیره ونگهداری داده در کسندرا را مطالعه کنید) ابتدا باید یک KEYSPACE جدید ساخته شود پس (فرض میکنیم نام KEYSPACE جدید weblog باشد):

```
CREATE KEYSPACE weblog WITH REPLICATION = {'class' : 'SimpleStrategy',
'replication_factor':1};
```

4. بعد از درست کردن KEYSPACE به صورت زیر به آن وارد میشویم تا بتوانیم دستورات مربوط به query رو اجرا کنیم:

```
USE weblog;
```

5.حال قادر خواهید بود تا entity های جدید ایجاد کنید و یا query ها مختلف را اجرا کنید. تعدادی query جهت تست، در فایلی به نام Instructions.cql ایجاد شده است، که از طریق لینک زیر قابل دسرسی خواهد بود:

https://github.com/hamed911/bsproj

بخش چهارم. راه اندازی کلاستر

کسندرا برای consistent کردن داده ها، برطرف کردن conflict بین داده ها از clock استفاده میکند. پس سینک بودن کلاک تمام نود ها اهمیت بالایی در consistent نگه داشتن داده ها دارد. برای این کار مراحل زیر را انجام دهید:

1.ابتدا باید برنامه ntp را دانلود کنید (دستورات در ترمینال وارد شود):

```
sudo apt-get install ntp
sudo  vi /etc/ntp.conf
```

خط زیر را به فایل بالا اضافه کنید:

```
server pool.ntp.org
peer 172.16.142.185 iburst
```

حالا دستور های زیر را وارد کنید

```
sudo /etc/init.d/ntp restart
ntpdate pool.ntp.org
```

2. نصب کردن Java Native Access (JNA) برای ورژن های بالاتر کسندرا:

```
sudo apt-get install libjna-java
```

3. حالا کسندرا را در تمام سرور ها، نصب و اجرا میکنیم (بخش های 1 و 2 و قسمت اول بخش 3)

4. آی پی سرور ها رو استخراج میکنیم: برای مثال در کلاستر ساخته شده آی پی ها به صورت دستی در بازه زیر قرار گرفت :

```
172.16.142.185 - 188
```

سرور 185 سرور اصلی (seed) می باشد.

بخش پنج. کانفیگ کردن کلاستر

1. کسندرا را میتوان از طریق فایل $cassandra_home/conf/cassandra.yaml کانفیگ کرد. فرض میکنیم سرور 185 سرور اصلی (seed) باشد و کانفیگ آن را به صورت زیر قرار میدهیم:

Cassandra.yaml (Server 185 – 'Seed'):

```
cluster_name: 'ModarressiCluster'
seed_provider:
  - class_name: org.apache.cassandra.locator.SimpleSeedProvider
    parameters:
        - seeds: "172.16.142.185"
Listen_address: 172.16.142.185
endpoint_snitch: GossipingPropertyFileSnitch
```

لازم به ذکر است اگر بیشتر از چند seed در یک کلاستر داشته باشیم، IP ها با علامت ","از هم جدا میشوند:

```
parameters:
        - seeds: "172.16.142.185, 172.16.142.186"
```

2. ابتدا کانفیگ فایل تمام نود های جدید را مانند بالا (سرور 186) اصلاح میکنیم

Cassandra.yaml (Server 186):

```
cluster_name: 'ModarressiCluster'
seed_provider:
  - class_name: org.apache.cassandra.locator.SimpleSeedProvider
    parameters:
        - seeds: "172.16.142.185"
Listen_address: 172.16.142.186
endpoint_snitch: GossipingPropertyFileSnitch
```

3. سپس کسندرا سرور اصلی 185 (seed) رو اجرا میکنیم و پس از آن، کسندرا سرور های دیگر را اجرا میکنیم (از طریق دستور زیر):

```
sudo ~/cassandra/bin/cassandra restart
```

4. متناسب با نیاز و تعداد نود های کلاستر، میزان replication factor را تغییر میدهیم:

```
ALTER KEYSPACE "system_auth" WITH REPLICATION = { 'class' : 'SimpleStrategy',
'replication_factor' : 3 };
```

مقدار replication_factor باید بر اساس نیاز به این که هر سطر از داده ها باید در چه تعداد نود کپی نگه داشته شود، باید تعیین شود و باید به نحوی باشد که با از دست رفتن یک نود، داده های آن از بین نرود.

5.یکی از مواردی که قابل کانفیگ است تعداد virtual node ها در هر نود می باشد. این تعداد باید میزان معقولی باشد. برای استفاده معمولی از کلاستر ، تعداد vnode ها میتواند 256 عدد باشد. از طریق تغییر متغیر زیر در فایل Cassandra.yaml میتوان تعداد آن را عوض کرد:

num_tokens: 256

بخش شش. مانیتور کردن کلاستر

برای مانیتور کردن تک تک نود ها میتوانید از برنامه nodetool خود کسندرا استفاده کنین. برای مثال از بالا بودن نود ها میتوانید از دستور زیر استفاده کنید

Nodetool status

که خروجی آن به صورت زیر خواهد بود:

```
Note: Ownership information does not include topology; for complete information, specify a keyspace
Datacenter: DC1
==============
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
--  Address     Load       Tokens  Owns   Host ID                               Rack
UN  10.0.0.41   28.04 MB   256     24.8%  c0c5efbd-43e5-465c-bd7d-e7233d5ff905  RAC1
UN  10.0.0.40   26.03 MB   256     25.1%  1623ac94-b071-4dee-a633-de1f1dffd503  RAC1
UN  10.0.0.42   24.94 MB   256     25.0%  80389087-59cc-4f9f-828b-de01ccfff98a  RAC1
UN  10.0.0.39   26.98 MB   256     25.1%  3fc82cb4-b612-41a4-8609-8351247b6538  RAC1
root@d499788c-6211-4211-9b06-9a39820257a5:~#
```

درباره این نرم افزار و قابلیت های آن یا به help آن و یا به سایت زیر مراجعه کنید:

https://docs.datastax.com/en/cassandra/2.1/cassandra/tools/toolsNodetool_r.html

نتایج کار

برای تست اوتومات کلاستر به زبان جاوا کدی زده شده که از طریق آدرس زیر قابل دسترسی است:
https://github.com/hamed911/bsproj

این کد، به صورت رندم تعدادی وبلاگ ایجاد میکند، که هر وبلاگ چندین پست دارد و زیر هر پست چندیدن تا کامنت موجود است. برای تست صحیح کسندرا، باید تعداد زیادی داده در پایگاه داده insert شود تا:

1) از سربار شبکه بتوان چشم پوشی کرد

2) فرآیند انتقال داده ها از رم به هارد، که قبلتر توضیح داده شد، در بازیابی داده ها، لحاظ شود تا خطا در بازیابی به وجود نیاید و به یک مقیاس واقعی از زمان بازیابی داده ها در کلاستر برسیم.

پس از اجرای برنامه بالا، نتایج زیر به دست آمد:

- اجرای برنامه روی یک نود برای بازیابی 900 داده:

Spent time is: 766 ms

- اجرای برنامه روی چهار کلاستر برای بازیابی 798 داده:

Spent time is: 484ms

لازم به ذکر است که زمان ذخیره داده در کلاستر به صورت زیر بوده است:

Inserting Time is: 3434 s

همانطور که انتظار داشتیم، سرعت بازیابی داده ها، در کلاستر چند نوده، نسبت به یک نود، به میزان قابل قبولی افزایش یافته است که نشان دهنده تاثیر مثبت موازی سازی در بهبود performance در این پایگاه داده است. البته لازم به ذکر است که نتیجه ی بالا با توجه به موارد زیر ممکن است تغییر کند:

1) اضافه شدن سر بار شبکه: دایما میان نود ها message هایی رد و بدل میشود. در صورت کند بودن سویچ داخلی شبکه بین نود ها، سرعت انتقال این پیام ها و در نتیجه پاسخگیری از کلاستر ممکن است با تاخیر همراه شود
2) وضعیت کنونی هر نود: همانطور که اشاره شد ممکن است نود ها مشغول انتقال داده از رم به هارد باشند که این امر سبب لاک شدن ساختار داده مورد استفاده می شود که نتیجه ی آن، افزایش تاخیر نود در پاسخگویی است

در آخر، باید به این نکته توجه داشت که Cassandra ، consistent بودن داده ها را در حال insert در تمام node ها تضمین نمیکند بلکه داده ها پس از گذشت زمانی، بالاخره consistent می شوند (eventual consistency). کسندرا سرعت write بسیار بالایی دارد و با استفاده از replication_factor به شدت دست طراح کلاستر را برای رسیدن به performance و reliability مورد نظر باز گذاشته است.