* دوره‌ی nonrelational DB :

جلسه‌ی اول ۹۷/۱۰/۶ :

فرق structured و unstructured و streaming ؟ ← ساختار زبانی وجود دارد که می‌توان با آن پرس و جو انجام داد

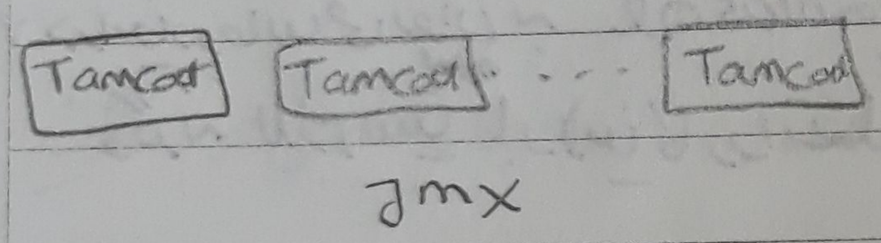
* بقیه جلسه‌ی اول * Big Data

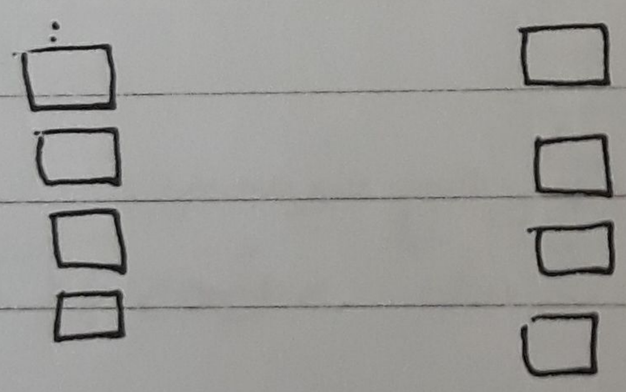
میزان داده‌های تولید شده توسط شرکت‌ها ← ؟ سال Go-Globe 60 seconds

ایده‌ها از کجا آمده؟

google ← hadoop facebook ← cassandra linkedin ← kafka

تبدیل کامپیوتر (سیستم با سیستم قدرتمندتر) ← افزایش شرکت‌ها / محدودیت فنی به لحاظ پردازش / محدودیت تکنولوژی





نیازمندی = رسیدن تمام داده‌های تولید کننده به مصرف کننده

راه‌حل ۱) point to point : شلوغی :D

راه‌حل ۲) بهینه) وجود صف در وسط، همه داده‌ها را دریافت کنند و مشتریان فقط باید ارتباط خود با صف را حفظ کنند و از آن بخوانند.

تعریف object storage = یک فایل به هر فرمت (متن، تصویر، ویدیو، ...) دریافت می‌کند و در معماری ذخیره می‌کند. یک آدرس یکتا تولید کرده و در ری‌سپانس برمی‌گرداند

می‌تواند metadata مانند tag هم کنار object ذخیره کند. اما نمی‌تواند query بزند.

سینتی. یک شناسه‌ی فایل را می‌گوید و آدرس آن را می‌دهد و می‌توان با آن آدرس به فایل راحت دسترسی یافت. با تابع Hash شیئی که آدرس به آن متعلق است را می‌یابد.

Subject ..... Date .....

از طرفی replica دارد. به دلیل پیاده‌سازی REST وابستگی به زبان خاصی ندارد.

amazon S3 یک انبار خوب تجاری است و swift از نمونه‌های open source آن است.

* NoSql DBS :

↳ Not only SQL : we don't also RDBMS SQLDB ها را نمی‌گیرند نه جای ها باشند

بلکه برای کمک در جاهای خاصی آمده‌اند. لزوماً یک Datastore در هیچ شرکتی نداریم که از چندین Datastore برای اهداف مختلف استفاده می‌شود.

✓ در این دو حالت می‌توان به NOSQL ها مراجعه کرد وگرنه لازم نیست بار این کار را تحمل کرد.

① متناسب پذیری
② Datamodel

* nosql-databases.org

- Datamodel
  - key-value
  - wide column
  - Document
  - Graph Neuj, infinite Graph

- API/CP/CA

آسیب‌پذیری NoSql = بعد از MR , GFS google یک DB به نام Bigtable

↓ سورس‌باز open source شده‌ی آن = Hbase

سافت. DynamoDB ← آمازون

معماری کاساندرا شبیه این است

Subject Date

*key_value :

شبیه داده ساختار MAP است key → value

در RDBMS مثل دیتابیس دو ستونه است

insert / fetch / update / Delete

داده‌ها در RAM نگه داشته می‌شود ← سریع

مقیاس‌پذیری افقی مدل ندارند.

redis / cauch /

redis = C++ پیاده‌سازی با

String
hash
list
set
Sortedset

} key باید String باشد و value آن می‌تواند باشد

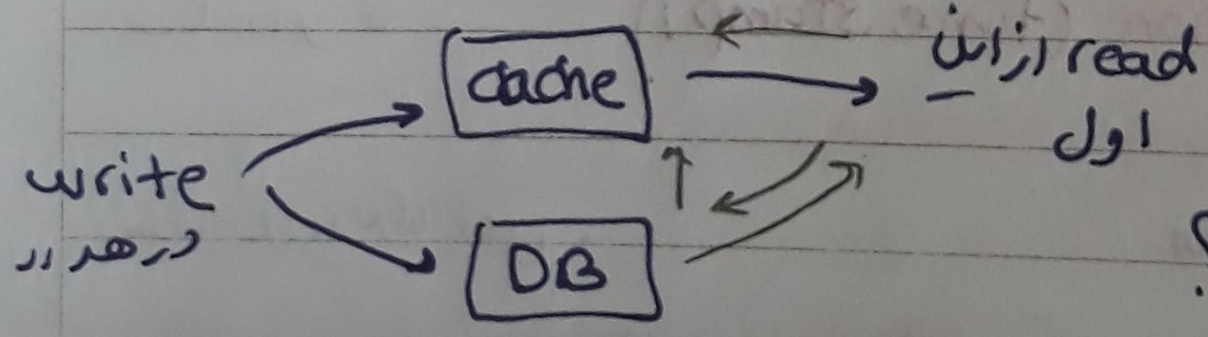
in memory → بخواهیم در دیسک هم می‌نویسید

- برای cache جلوی DB اصلی خوب است ولی به‌عنوان DB اصلی خوب نیست

کاربردهای مناسب:
- messaging
- score game

با قطع جریان برق ؟ - علاوه بر memory در دیسک هم بنویسیم

- اگر داده‌ی حساس نباشد مثلاً cache باشد به محض دوباره بالا آمدن سیستم دوباره ساخته می‌شود.



مهندس: خیلی از RDBMS ها خودشان cache دارند خوب کاربرد Redis ؟

① می‌توان redis را در ماشین دیگری داشت (در واقع نقش cache لایه‌ی بالاتر را داشته باشد).

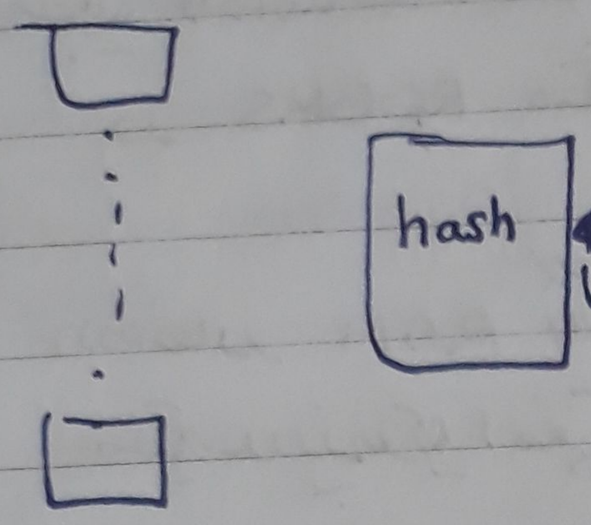
② می‌توان Session managment را روی آن مدیریت کرد.

③ مثلاً برای بازی کاربران را بر اساس امتیاز سورت کند.

clips

Subject ..... Date .....

وقتی داده داریم در تمام ابزارها کجا ذخیره می‌شود؟



primary key را در hash می‌اندازند

و باقی مانده‌ی آن را بعد در ماشین‌ها به دست آورده و جای نوشتن داده مشخص می‌شود.

FS / DB / OBJ

* Column - Oriented DB

گوگل مساله‌ای داشت برای search. به ازای هر URL جدول به ذخیره‌ی property بود اما تعداد این ویژگی‌ها برای هر URL متفاوت بود. اگر RDBMS کار کنیم باید پیش‌بینی کرد که حداکثر چند تا ویژگی داریم و به آن اندازه جدول بسازیم ⇐ ماتریس sparse

ولی فضا گرفته است. گوگل bigtable را طراحی کرد. یک primarykey داریم و مجموعه key-value ذخیره می‌کنیم که چه از نظر نوع و چه تعداد لزومی ندارد شبیه هم باشند.

ساختار = Row key , set (column, version (time stamp))

مدل‌ها: BigTable / HBase

cassandra → منبع فیسبوک

معماری پذیری افقی.

مفهوم column family ⊥ table (ویژگی‌های جدید آن) ممکن است تمام rowها شکل name را داشته باشند برای آن column family تعریف می‌کنیم.

به هر row یک timestamp نسبت داده می‌شود.

مقیاس پذیری کاساندرا خطی است ← با افزایش تعداد نود عملکرد بهبود پیدا می‌کند در بعضی DB ها اینجوری نیست.

* Document Store

- Schema free
- DOCID کافی است ذخیره شود — به ازای هر داده‌ای که ذخیره می‌کنیم یک DOCID
- JSON (BSON) — JSON داده با هر یک درک یک پاس داده می‌شود.
- Query Model: Javascript or custom
- indexs are done in BTree
- Aggregation: MAPreduce
- مثال‌ها ⟶ MongoDB / CouchDB / CouchBase / RethinkDB

هر ۳ نوع DB معرفی کرده = هر کدام کجا؟

① key-value = توی ram و سرعت

② فرق column و Document ؟

زمانی که مبتنی بر کلید query می‌زنیم خوب جواب می‌دهد.

مثلاً اینا رو ذخیره می‌کنیم بعد می‌خواهیم مربوط به سایت خبری تابان ← column based خوب است

ولی مثلاً دنبال خبرهای حادثه‌ی کلاه‌ی ایران باشیم؟ ← Document based خوب هستند.

کلاساندرا خوب است

دسترسی‌ها موردی که کلید مشخص باشد.

Subject ..... Date .....

* Graph Databases

معروف ها

- Neo4j | Titan | OrientDB

مثلاً کوتاه‌ترین مسیر روی RDBMS ها لازم است که چندبار self join روی دیتابیس اجرا کنیم که صدوبیست هم دارند مثلاً RDBMS ها 6 تا 7 تا بیشتر نمی‌توان self join زد.

این توزیع شده است

open است

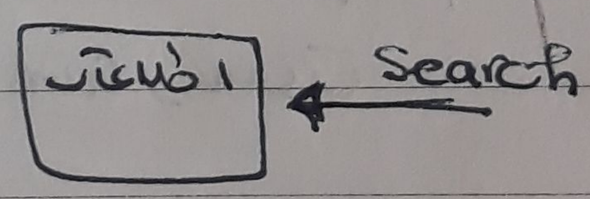
ArangoDB →

رفتار دیتا یک document based است و می‌گه گراف هم با ارتباط بین این‌ها تعریف کرد.

می‌گه من یک DB داریم و همه را doc based ذخیره می‌کنم.

کپ متاس نذیری =

Elastic Search =



مرتبط‌ترین براساس score را برمی‌گرداند.

تئوری CAP در سیستم توزیع شده ۳ مفهوم وجود دارد

همیشه فقط دوتا را می‌توان داشته باشیم.

- Consistency = مجموعه‌ای از نودها داریم و دوتا کوئری write و read به DB داریم می‌زنیم

- Availability یا اگه من الان consistence نیستم یا اگر جواب برمی‌گردانند most recent نمی‌دهد داده را می‌دهد.

- Partition torelance = مجموعه به دو یا چند جزیره دستگاه به خاطر یه خرابی جدا می‌شود ولی سیستم همچنان به کار خود ادامه دهد.

در DB داریم → ASID

- Consistency = اگر جواب دهد یعنی تضمین کرده به روزترین داده را جواب داده

- availability = در حالت قبل حتماً جوابی می‌دهد ولی تضمینی بر داده most recent ندارد.

~~PT~~ ~~AV~~

* ۳ نوع توزیع شده خواهیم داشت:

CP →

AP → جوابی می‌دهد اما شاید قدیمی‌ترین نسخه.

CA → مثل RDBMS چون اصلاً توزیع شده نیستند

• انواع تایپ های Consistency:

✓ strong consistency = هر زمان آپدیت رخ دهد DB باید در تمام نودها آن را انجام دهد بعد ack برگرداند.

After the update completes, any subsequent access will return the same value.

✓ weak consistency = به محض دریافت آپدیت و نوشتن در node اول ack را برمی‌گرداند و سپس در background روی سایر نودها می‌نویسد. ممکن است هم لهو و سطش بیافتد و به بقیه نرسد.

✓ Eventual consistency = مانند nosql نوعی از weak هستند اما اگر شلوغ نباشد و آپدیت‌ها زیاد نباشد بهت تضمین می‌دهم (قول می‌دهم فقط) که سریع همه را آپدیت کنم.

facebook این راه را انتخاب کرده ← چون تعداد کاربران active (ابیلون) زیاد است اگر صبر می‌کرد strong بود سایت‌ها طول می‌کشد نشون داده شود.

Subject ..... Date .....

به خاطر نیاز تجاری از دست ندادن کاربران این کار رو کرد.

مثال 2) حتی از ATM ها حتی وقتی دسترسی به شبکه نداشته باشند یک سری خدمات مثل مبالغ کم را به کاربر می دهد. Availibity برایش مهم تر است.

مثال 3) خروجی شبکه اینترنتی: Availibity = وقتی صفحه را درخواست می کنیم سریع نشان دهد

این جا با اولین نود نشان می دهد.

consistency = موجودی کالاهای صفحه ی اول درست باشد

اما در زمان صفحه ی پرداخت consistency مهم است پس اینجا همه ی نودها را چک می کند.

رویه ی NewSQL = با داشتن partition torelance، پس Availibility و Consistency داشته باشد.

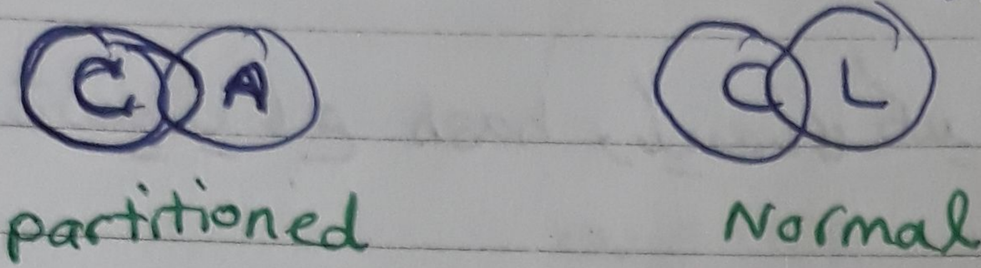
در هیچ کدام از NoSQL ها transaction نداریم.

جلسه دوم : ۹۷/۱۰/۱۳

در 2012 مقاله‌ای نواقص CAP را با ارائه‌ی تئوری PACELC

وقتی که fault ای رخ دهد (سیستم جزیره‌ای شود) باید بین A (Availability) و C (Consistancy) انتخاب کنیم else باید بین latency و consistancy باید انتخاب کنیم.

(سازگاری / ثبات)



partitioned Normal

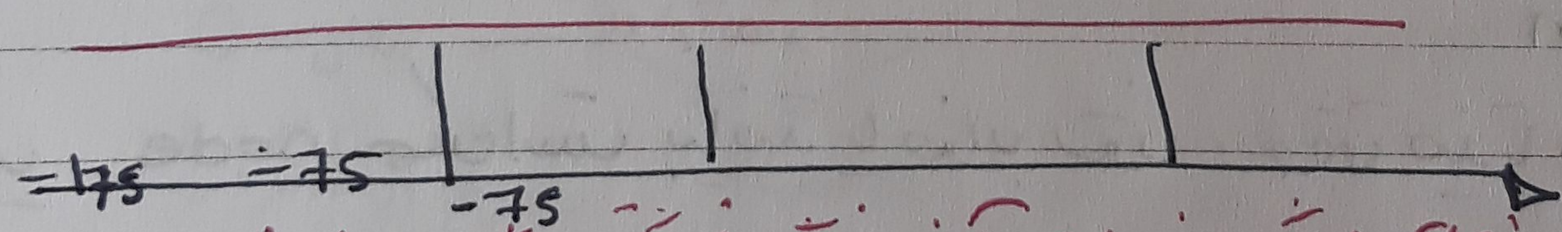
* Nosql common techniques:

① Sharding = partitioning , fragmentation

بخش‌بندی کردن
تیکه تیکه کردن

وقتی که حجم داده‌ها زیاد باشد ← یک سری ماشین برای ذخیره‌سازی می‌بینیم . چگونه کلید قرار دهیم؟

1) range based



کلید اصلی در هر بازه‌ای باشد همین داده در آن بخش نوشته شود یا خوانده شود.

مزایا =

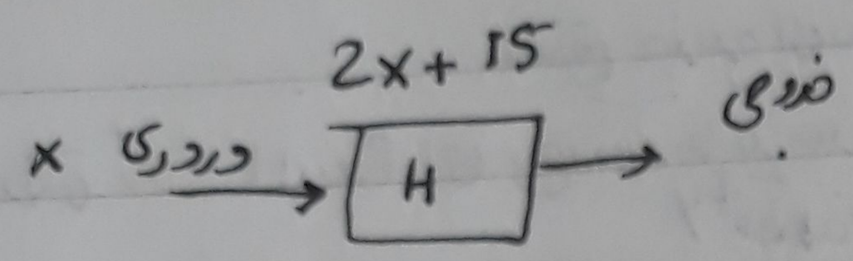
* تمام داده‌های یک رنج در یک ماشین قرار می‌گیرند و query را در یک ماشین اجرا می‌شود.

معایب =

* ممکن است برخی از ماشین‌ها bottleneck شوند
یک کلید انتخاب کنیم که (تاریخ ماه + روز) : مثال
پس همه‌ی داده‌های امروز در یک ماشین قرار می‌گیرند
* ممکن است هم ماشین (داده‌ها) حجم زیادی شود و balancing یک صفی خوبی انجام نشود.

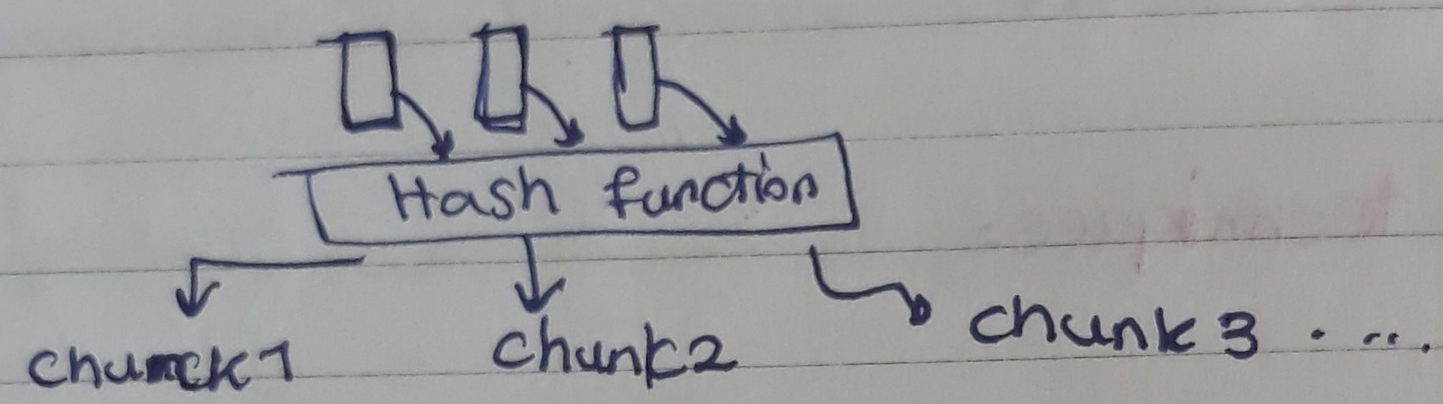
② Hash function

hash همیشه تابعی است که با گرفتن داده جوابی می دهد همیشه با یک داده ی ثابت جواب ثابت می دهد اما برعکس پذیر نیست.



hash های معروف مثل MD5.

می توانیم یک تابع hash و یک فضای کلید داشته باشیم.



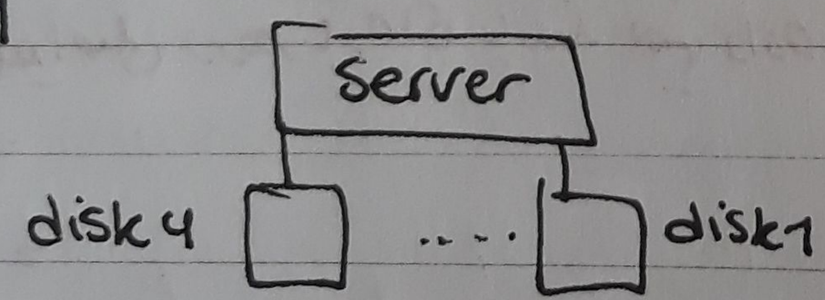
برخلاف rangebased که کلیدهای شبیه هم در یک ماشین قرار می گیرند در اینجا این گونه نیست.

اکثر دیتابیس ها hash based هستند. MurMurhash function یکی از روش های خوب در آن هاست.

2) Replication

چون یک node ممکن است بیافتد یا خراب شود. وقتی چند تا دیسک داریم تکرار کنیم

raid level = 1



RAID یعنی چه؟

0 : داده را تیکه تیکه می کند در ۴ تا دیسک بالا. خواندن داده سریع می شه عملکرد بالا می رود. اگر یکی از دیسک ها از بین برود همه را از دست می دهیم.

1 : داده را کپی می کند علاوه بر دیسک در دیسک دیگر هم می گذارد.

2 = تیکه ی دوبخش داده را هر تیکه را دوبار کپی می کند.

= 5 parity محاسبه می‌کند و ذخیره می‌کند ؟؟

در سیستم‌های توزیع‌شده RAID در قالب یک سیستم اجرا نمی‌شود چون خیلی فضا می‌گیرد.

معایب = گرفتن فضای زیاد

مزایا = scalability + failure Tolerance

Consistency Model = { synchronous (consistance است) vs asynchronous }

زمانی که همه جای تقاضای replica نوشت Ack می‌دهد.

روی تعدادی از تعداد replica نوشت Ack می‌دهد و بقیه را background اجرا می‌دهد.

* replication = 3 * مثال

{ Availibity , performance بین tradeoff }

* Apache Cassandra *

تعریف = distributed , decentralized , fault tolerant ,

evaunlily consistent , ~~beam~~ linearly scable , column-oriented

تاریخچه : bigTable Google , Dynamo Amazon

از این‌ها ارائه گرفت Facebook

dataslacks یک سیستم Cassandra است.

پیاده‌سازی با جاوا ، مکانیزم { masterless همه نودها خاصیت را دارند } slave , master

master-slave مثل mongoDB . هیچ وقت داده روی master نمی‌شود حتی job را نباید اجرا کند فقط وظیفه‌ی مدیریت را دارند

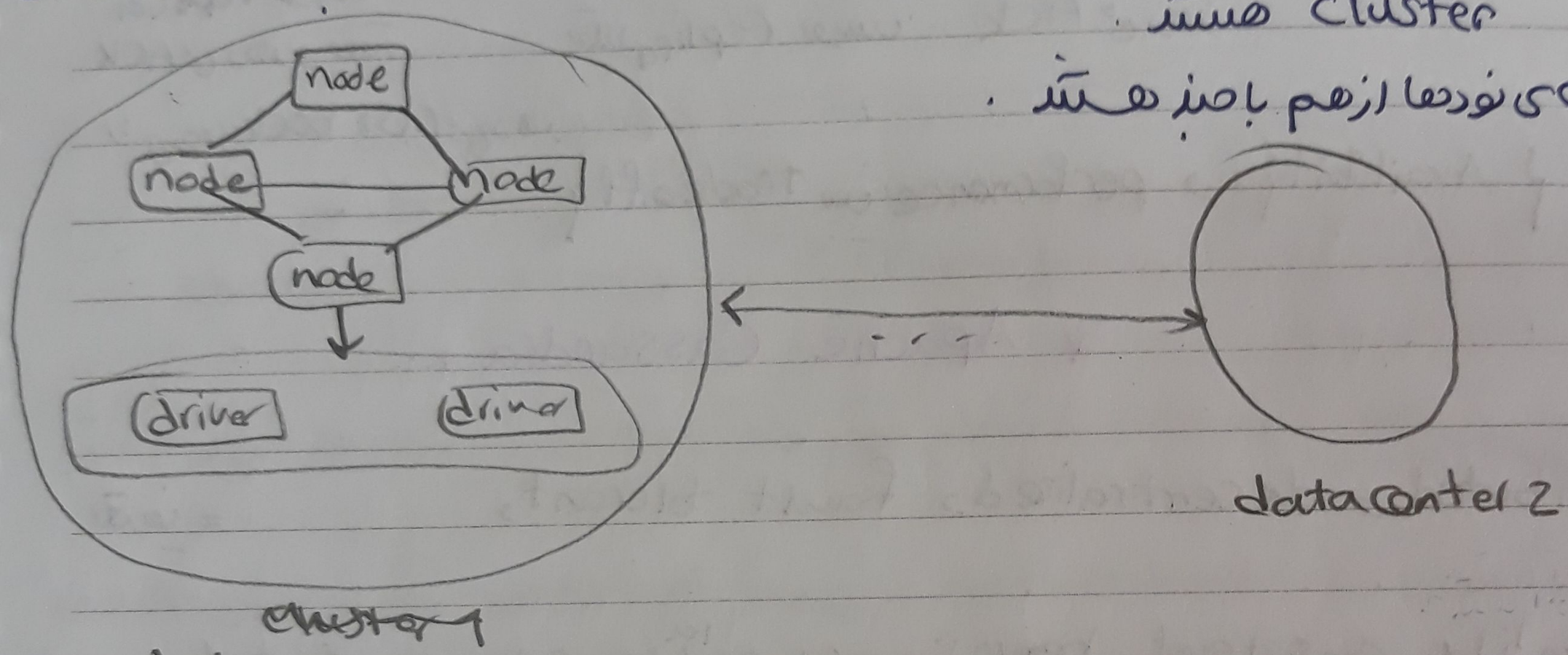
چون = Single point of failure ندارد.

Subject ..... Date .....

* Tunable consistency - CQL languge - Timeseries
Multi Data center - High performance - predictable scaling
100% uptime - Distributed - Masterless - java - NoSQL
opensource

Cassandra key structures: Node, Datacenter, cluster

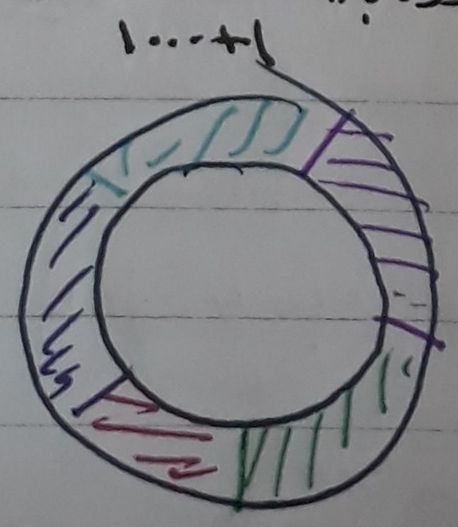
یک سری نود داخل Rack . Rack داخل datacenter و datacenter داخل cluster .
همه‌ی نودها از هم باخبر هستند .



* مفهوم = Ring Representation

هر کلید مخصوص یک ماشینی است .

* در کاساندرا برای درست اولیه فضای کلید n را به تعداد مساوی تقسیم کرده و $\frac{n}{1000}$ به هر نود می‌دهد. بعد عیب آن = هر اضافه کردن هر نود جدید باید دوباره $\frac{n}{1000+1}$ تا تقسیم شود یا اگر نود افتاد.

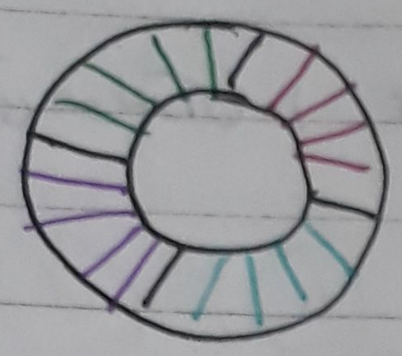


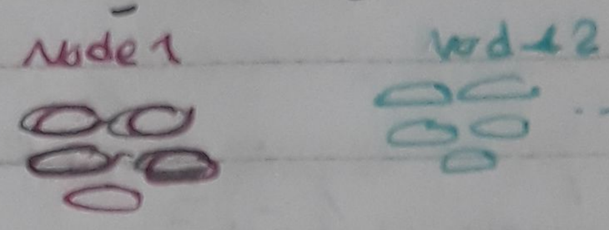
Node1 و Node2 و ...

Subject Date

virtual nodes

به تعداد $\frac{n}{100}$ تکه می‌شکنیم و به هر نود 5 تا از این را می‌دهیم. نود جدید می‌آید از این تقسیم می‌کند.





token تولید می‌کند.

= Data Modeling

در RMDBS داریم اما اینجا نداریم!

join
foreignkey
aggregation

فلسفه‌ی NoSQL دنبال performance است. در RMDBS، Normalization داریم ← ذخیره‌سازی بهینه شده ولی بار محاسباتی زیاد

فقط

Cassandra می‌گه قسمت ذخیره‌سازی نسبت به محاسبات ارزان‌تر است. از روش denormalization استفاده می‌کنیم یعنی join را ذخیره می‌کنند!!

RMDBS

مدل user		مدل Tweet
name ID	و	ID tweet

وقتی کاربر وارد صفحه می‌شود باید join این دو مدل را بسازد name, tweet

اما در کاساندرا همین را ذخیره می‌کنم:

name	tweet

combine columns from different tables

عیب = مشکل هنگام آپدیت

رابطه‌ای	کاساندرا
Data base	keyspace
Table	column family (CF)
Primary key	row key

rowkey	Column key 1	Column key 2	.....
	Column value 1	Column value 2	

clips™

Subject ..... Date .....

تنظیمات ← Config این فایل لازم است : cassandra.yaml

وقتی فولدر آن را extract می‌کنیم :

دستوراتی که اجرا می‌کنیم اینجاهاست

bin →
sbin →

مثل cassandra برای اجرا

برای اجرای ویندوز روی آن bin همه بزنیم. cassandra.bat →

conf → cassanda.yaml

- cluster name : 'Test cluster'
- num_token : 256 → hash هون

چه تعداد virtual node به این کاساندرا بدهه یعنی هر ماشین 256 تا از آن شیء ها باشد. وقتی 4 تا نود داریم و هر کدام 256 را بخواهند ⇐ $4 \times 256 = 1000$ تا token باید کلید داشته باشیم.

وقتی کلاسترها ناهمگن است. برای کلاستر قوی‌تر این عدد را بالاتر ببریم.

- listen_address : localhost

ارتباط بین نودها از این IP می‌گذرد. نودها از این طریق صحبت می‌کنند. مثلاً سرور دو تا interface داشته باشد : برای نودهای خود کاساندرا

- Rpc_address : localhost برای اتصال client ها (ما) مثلاً یک interface سرور را اینجا به کلاینت بدهیم.

در معماری master slave آدرس master را می‌دهیم و همه هم را پیدا می‌کنند.

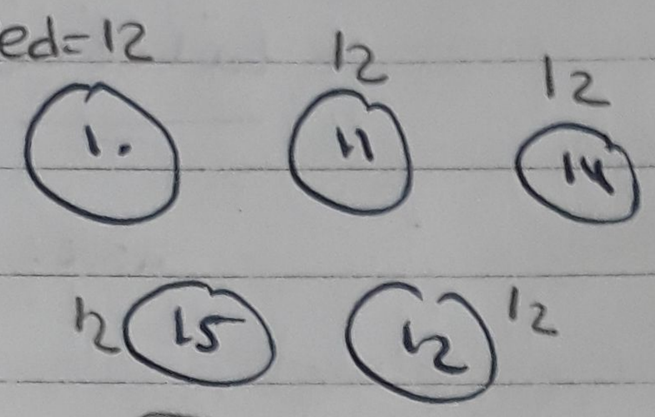
اما در masterless مفهوم gassib داریم (پیچ پیچ کردن) دارند.

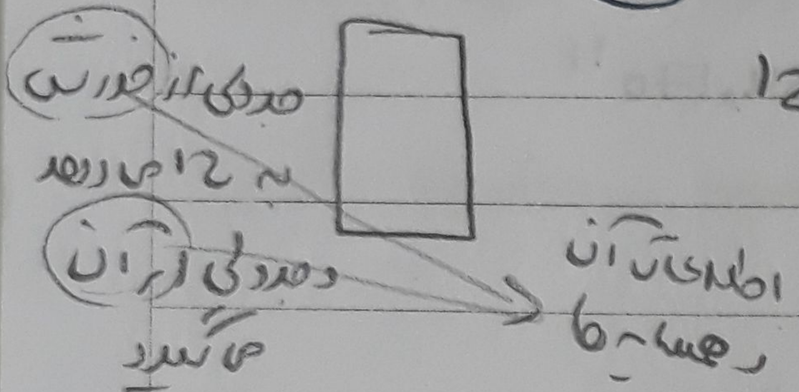
_seeds: "127.0.0.1", ( ) , ( )

در تمام ماشین‌ها آدرس چه تا ماشین‌ها را باید در این seed مشخص کرد. تا زمانی که ماشین بالاست هر ۳ ثانیه یک بار این gossib رخ می‌دهد.

قابل تغییر در yaml

seed=12





① دفعه‌ی اول 12 از همه می‌گیرد

② " دوم به بعد 12 اطلاعات همه را به هدکی gossib می‌کند می‌دهد

③ از دفعه‌ی بعد هدکی با همسایه‌ی خود gossib می‌کند و با 12 کاری ندارند.

وقتی بیشتر از مثلاً 10min جواب از gossib نگیرد آن را dead فرض کرده و در gossib بعدی به بقیه که دارد مردن آن نود هم اطلاع می‌دهد.

به نودی از کاسندرا که کلاینت به آن وصل می‌شود: coordinator می‌گویند.

✓ commitlog_directory

✓ datafile_directory = داده‌ی اصلی در این جا ذخیره می‌شه

داده‌های کاربر اینجا می‌ریزه /var/lib/cassandra/data

دیفالت آن cassandra home همان فولدر کاساندرا می‌ریزد. ویژگی‌های

✓ vm : /mnt

RAID 0 & SSD استفاده می‌کنند.

✓ saved_caches_directory = table key & row caches are stored.

✓ hinted hand off_directory =

* نقش‌های سیستم کلان داده:

Data Enginner ↓	Data Developer ↓	Data sciensitic ↓
تعدادی از ابزارها را می‌شناسد و مهارت برای این مدل داره	Mapreduce	statistic
و دیگر این ابزارها و نحوه	spark	Machine learning
performane tunning	elastic	مفاهیم صورت مسائل آن حوزه را بشناسد
Linux	algorithms for machine learning	
Virtulization		
docker		
Container		
AWS/Google Cloud		
Azure Microsoft		

query Model بلد باشد

Cloud

Subject ..... Date .....

ulimit -a → openfile ⬭ مثلا وقتی با kafka کار می‌کنیم کلی فایل باز می‌کند باید در این بخش linux تغییر دهیم.

performance tunning

yum install nano

nano conf/cassandra.yaml — تنظیم کاساندرا

listen_address = 192.168.234.190 → master

rpc_address = 192.168.234.190 → slave1

-seeds = "192.168.234.190, 192.168.234.170" → slave2 ← 120

در vi با escape وارد command می‌شویم /seeds تا find seeds را آنالیز

vi /etc/profile

export cassandra_HOME= "•"

export PATH = $cassandra_HOME/bin : $PATH

rm -fr ⬭ پاک کردن فولدر D:

docs/html → مستند آخرین ورژن کاساندرا را دارد.

cql3 مستند دستورات

ssh 192.168.234.170 slave1 → وارد یم داخل Yamel

(I) listen_address = --- 170

(II) rpc_address = --- 170

(II): ---120 , (I)2 ---120 : همین کار در slave2

* putty duplication : 190 / 120 / 170 یک session دیگر روی

cassandra -f

forgrond وگرنه خودش background ایجاد می‌کند.

user باید start خوب نیست بلکه user root با جدید ساخت در کاساندرا برای کار تعریف کرد.

cassandra -fR

در تمام ۳ تا جا این جوری استارت می‌کنیم.

Subject Date

برای gossip لازم است ماشین‌ها passwordless باشند؟ نباید :D

تعداد زیادی ماشین داریم. برای monitoring و تعداد نودهای بالا ... چه کنیم؟

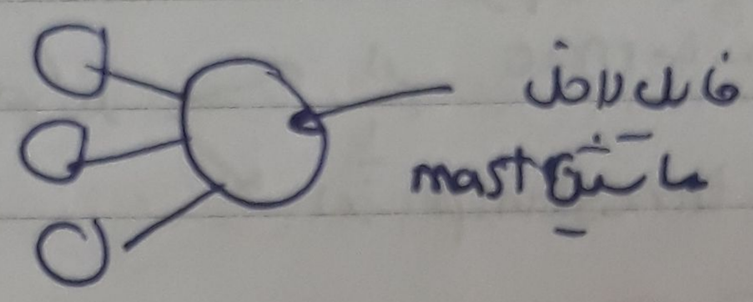
① BigData tools = Ambiere ، opscenter

Datastack UK — ولی بیشتر برای hadoop است

② Config managment = pupple , chef , amsible , ?

این فرق

* دقیقه‌ی ۴۴



هدوپ مثلاً فلان فایل تغییر کرد سرویس را ریستارت کن.

کاساندرا

③ Virtualization : یک سری image از پیش آماده مثلاً کاساندرا توش نصب هست داشته باشیم. بعد با ابزارهای

Vmware / docker / Kubernetes — اینا خوبن!

{docker compose cassandra} = search کنید :D

مفهوم devops : ؟

cqlsh 192.168.234.190

netstat -pltun — کل پورت های سرویس

netsta -pltun | grep java — پورت های سرویس جاوا

top ای از بین بردن یک پورت: آزاد کردن ✓ kill -9

ما از این طریق وصل میشیم 9042

کاساندرا بالا میاد 7000

[باید این دو تا port را به firewall اضافه کرد!]

با tab زدن میشه هدرس

create keyspace ks1 with replication =
{'class': 'simplestr...'
'replication factor':

Subject ..... Date .....

replication factor در متاسفانه keyspace است (نه کل کلاستر!)
همه table داخل این keyspace بسازیم replica همان قدر دارند.

DESCribe keyspaces; لیست keyspace های تعریف شده می‌دهد.

فلانی keyspaces
use

انواع Datatype کاساندرا داریم

CREATE TAble tbl1 { id int, name text, lastname text,
primary key (id) };

* UID = یک عدد unique تولید می‌کند
از mac ماشین
یا
از timestamp

P

INSERT

{ select * from tbl1;
INSERT tbl1 (id, name, lname) values {1, man1, fman1}
یا { select * from ~~keyset~~ ks1.tbl1;

Select * from ks1.tbl1 where name='man1';

می‌شه name جزو کلید نیست و اگر اجرا کنم performance ام کم می‌شه.

آخر دستور Allow Filtering می‌ذارم

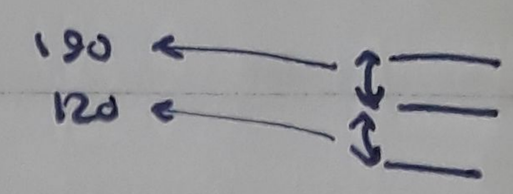
query ها حول کلیدها باشه ⟸ مناسب است.

ابزار nodetool بهترین ابزار برای مدیریت کاساندرا.

* nodetool status
کلیه نودها را
تمام نودها را
نشان می‌دهد.

* nodetool ring
Virtual Node (VN) ها را نشان می‌دهد.
Token یک عدد 22 رقمی است

در nodetool status هر سرور 256 تا token دارند



256x3 =768

در این جا یل 768 تا کلیدها انگار تقسیم شده در هر کدام 256 را گرفته اند.

داخل فولدر داده
- data ← keyspace های که داخلشان table ساخته شده فولدر دارند.
- commitlog
- backup = داخل این خالی :D

ctrl c = kill cassandra اگر یکی از نودها را kill کنیم برگردیم داده هست تازه !!

زمانی که یک table بسازیم در تمام ماشین های کلاستر table ساخته شده و داده توی تمام آن ها پخش می شود (کنار هم قرار نمی گیرد لزوماً)!

داخل پوشه ی tools یک bin قرار دارد که sstabledumps قرار دارد که داخل

/tools/sst.... /root/cassandra/data/data/ks7/table....

/ .db

یک viewer است داده ها را نشان می دهد. به صورت JSON نمایش می دهد وگرنه داده ی کوه نوشت اصلی خودش فرمت binary دارد.

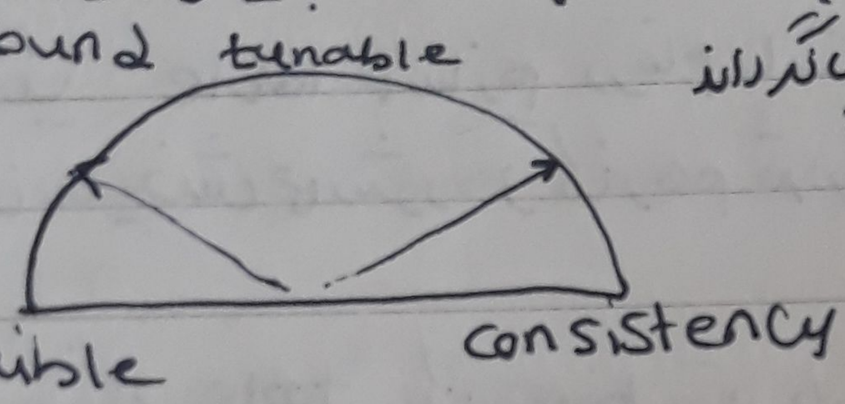
جلسه‌ی سوم ۲۰، ۱۵، ۹۷

Tunable consistency=

در کاساندرا این ویژگی per request است.

فرض کنیم ۴ تا نود داریم و replication factor = 3 است. در CL تعیین می‌کنیم که از ۱ Ack برگردانند. به تعداد مشخص شده در CL رمز می‌نوشتیم rep-CL تا بقیه را در back ground نویسید

Ack را می‌گردانند



cassandra (f)(R) → چون با کانت root داریم

forground

اول نودهایی که در seed معرفی کردیم را بالا بیاریم بعد بقیه رو. چون اگر اول بقیه (non seed) بالا بیاید چون می‌توانند گسیپ می‌افتند.

gossip

حال با * cql sh IP. نودی که می‌خواهیم وصل شیم

consistency → default one هست می‌نویسیم میزه

" TWO به ۲ تعیین می‌دهیم.

consistency All → می‌ره در کل تعداد replica factor می‌ریزد بعد Ack می‌دهد.

consistency Three ; → حال بعد این یک نود را بندازیم insert کنیم بعد

Nothost Available ← خطا می‌دهد

می‌تواند select کنیم هم همین خطا را می‌دهد.

فرض کنید ~~F~~ تا نود داریم ، Rep=5 است. consistency را چند بگذاریم؟

C=5 → شبیه RDBMS می‌شویم که خوب نیست

* می‌توان consistency هر نوع دستور (مثلاً read ، write) را جدا تعریف کرد.

read
اگر R = 1 و write, update = All . اینجوری هر زمان read بیاید مطمئنیم که درست است

اگر read = All , " = 1 : وقتی در محیط write به شدت خیلی بیشتر از read است. مثلاً داده‌های سنسور می‌آید.

اگر read و write تقریباً یکی باشد ، $\frac{Rep}{2}$ تا بیشتر (اصلاً نه کوچکتر) نگذاریم

حل است : redd = 3 write = 3

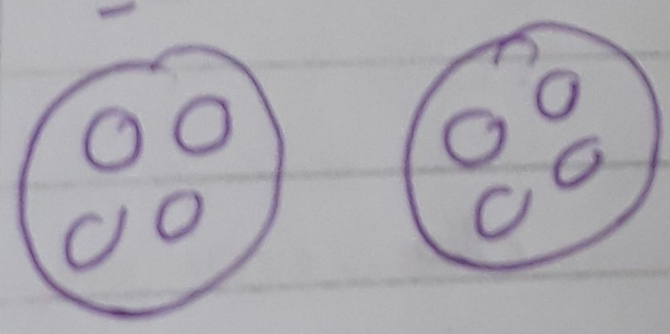
* quorum = (#replica / 2) + 1 *

در محیط‌های توزیع شده از این کلمه زیاد استفاده می‌شود.

Tunable consistency level:

- Any
- one / two / three : از نزدیک‌ترین replica‌ها می‌گیرد
- All
- QUORUM با محاسبه‌ی آن جا را می‌یابد
- LOCAL_QUORUM
- EACH_QUORUM

زمانی که چند تا Datacenter داریم.



Create keyspace class = 'NetworkTopology'
{dc1: 3, dc2: 4}

به ازای هر Datacenter یک replication factor متفاوت انتخاب می‌کنیم.

EACH_QUORUM = ? 30 دقیقه‌ای در سطح هر DS انجام دهد

localhost:ips...

quorum ← { تعداد نودهایی که می‌نویسیم و می‌خوانیم بیشتر از replica باشد }
از این آسانه

read always reflects the most recent write
if not : weak consistency ?

INSERT INTO tbl (id, name, lname, tel) Values(. . . .
)

!! خطا می‌دهد

ALTER Table tbl

" WITH می‌توان خواص خود جدول را تغییر داد.

" ADD tel;

حال دستور (*) اجرا می‌شود.

بریم ببینیم داده چگونه ذخیره شده؟

tools/bin/sstabledump /---/data/data/ds3/ . . . .

خروجی json طور ترمینال داده‌ها را نشان می‌دهد.

Secondary Index :

Create Index on tbl (name);

حال

Select * from tbl where name = 'me'

را می‌دهد.

چگونه می‌سازد؟ فرض کنیم یک جدول ۱۵ میلیون داریم اگر بخواهیم کلیدی (a) را در آن‌ها بیابیم؟

حال کلمه‌ی 'b'. مقدار زیاد فایل حجم زیاد داریم invertable index در search engin ها ها ساخته

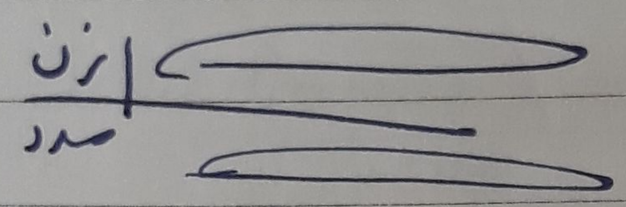
a	doc1 , doc2
b	doc2
c	doc1

در واقع تمام doc ها را یک بار می‌پیماید و این جدول را با آن ها پر می‌کنیم. می‌توان تعداد تکرار کلمات را کنار نام فایل‌ها نوشت بعد از این به بعدی search کند از این جدول می‌خواند.

کاساندرا هم دقیقاً همین کار را می‌کند. به ازای ستونی که secondary index تعریف کردیم می‌آید ID را جلوی آن می‌نویسید.

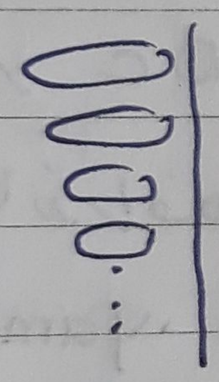
در مصطلح‌هایی که کاردینالیتی زیاد یا کم باشد خوب نیست. مثلاً جنسیت کلاً دو تا

جلو دارد پس جدول ID و کلی ستون



$c=2$

query هایی که بیشتر از 50٪ داده‌ها را می‌خواهد بدون کلید جزء کاساندرا می‌گه allow filter بذاری بهتره تا index



بنی سطری

وقتی سمتاره موبایل باشد کاردینالیتی = ۱ می‌شود و خیلی زیاد زیاد می‌شود. ← خوب نیست

جایی که مصطلح خیلی update و delete دارد ← secondary Index موجب مشکل performance می‌شود.

شبیه secondary index است

materialized view → چیست؟

Partitioner = اولین نود را مشخص می‌کند. اگر یک کلاستر با چندین نود داشته باشیم داخل هر نود یک partitioner داریم. به اولین نودی که وصل می‌شویم

می‌گوییم conti...

Subject ..... Date .....

و متی به آن وصل می‌شویم partitioner ، primary key را به token تبدیل می‌کند.

select id, token(id), name from

الگوریتم های partitioner ؟

[murmur , random , byorder]

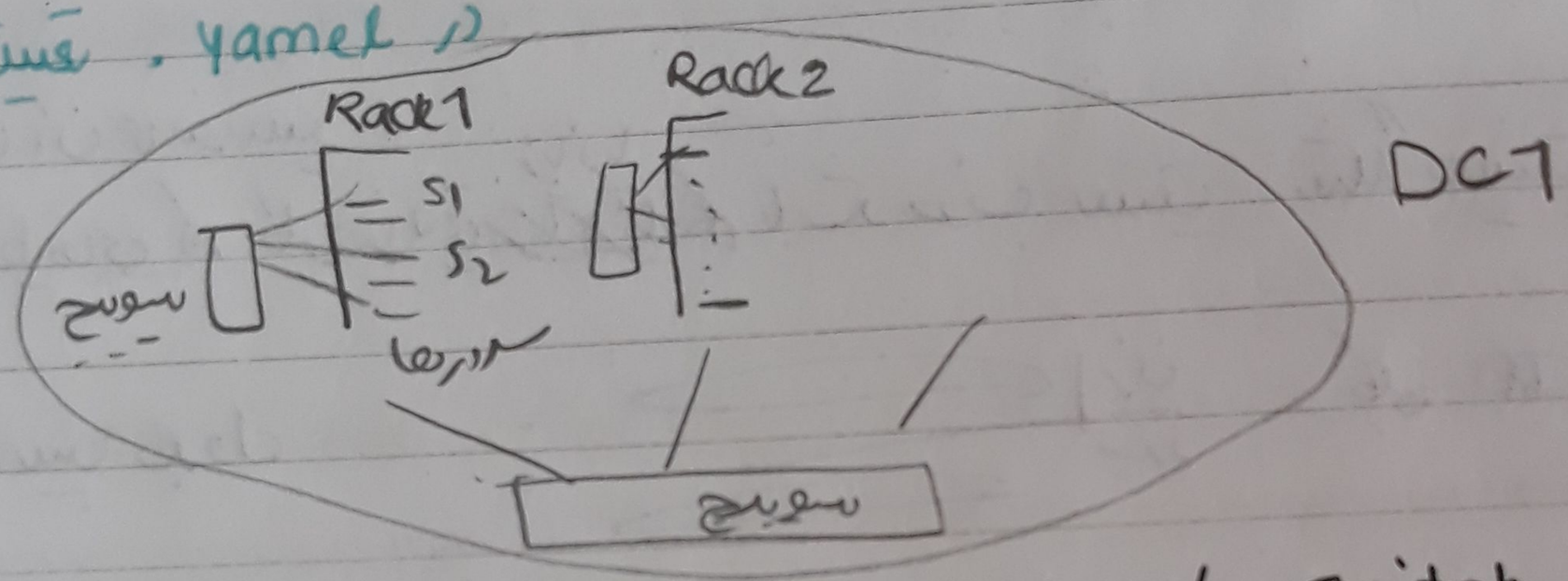
murmur: کمترین

random: MD5 algorithm

byorder: کلید را براساس حروف مرتب می‌کند

مثل از ۱ ساعت م

در yamel . تعیین می‌کنیم



مفهوم ساخت snitch در کاساندرا = می‌توان بر هر کلاستر کلاستر rack و

nodetool status

مثلا همه در Rack1 هستند.

ساختار DC را می‌توان به کاساندرا فهماند.

در yamel : "snitch"

① simple = flat همه چیز به صورت همه را در یک Rack می‌بیند.

② property

cassandra topology property

Ip: DC1: RAC1

در همان فولدر کاساندرا

این فایل باید در تمام ماشین‌ها اضافه شود. اگر دیتا نتر استاتیک باشد دیگر نه در دینامیک داستان داریم

③ Gossip

cassandra RackDC.pro ...

dc=dc1

rack = rack1

هر ماشین برای خودش را دارد و هنگام Gossip کردن ماشین‌ها توپولوژی را می‌فهمند

AECSnitch → Amazon Elastic

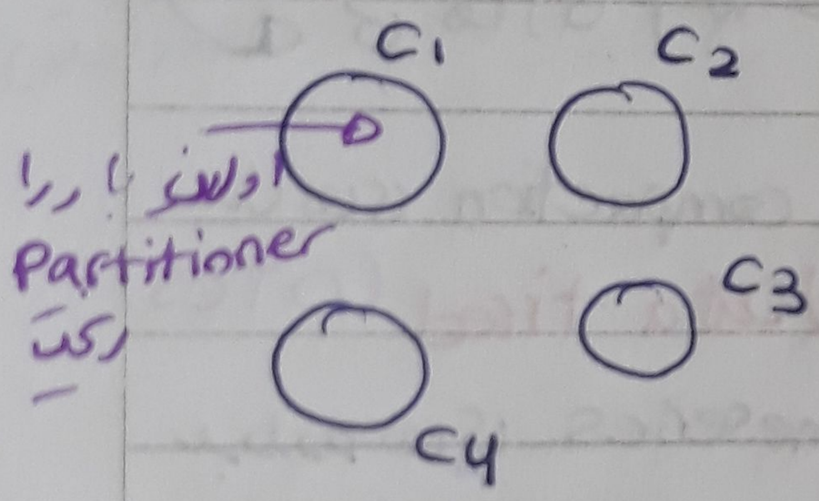
RAckInference

با تغییر این مقادیر توان DC و Rack مختلف هنگام IP دادن تعیین کرد.

192.168.10.21 ←
DC RACK

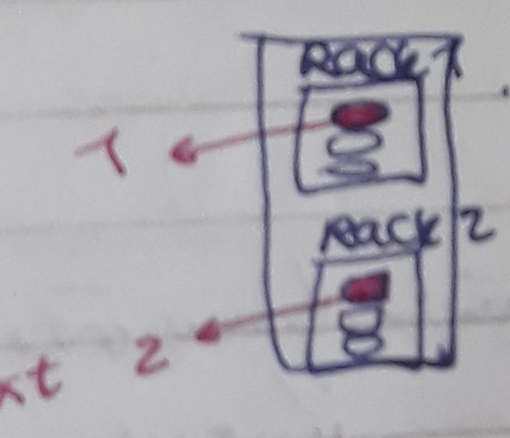
اولین جایی که داده نوشته شود را partitioner مشخص می‌کند بقیه را replication strategy تعیین می‌کند دو روش داریم که هنگام تعریف keyspace مشخص می‌کنیم:

① Simple: فرض می‌کند flat هستیم در جهت عقربه‌های ساعت در نود بعدی می‌نویسید. (در ring)



next node = کسی است که vnode بعدی را دارد.

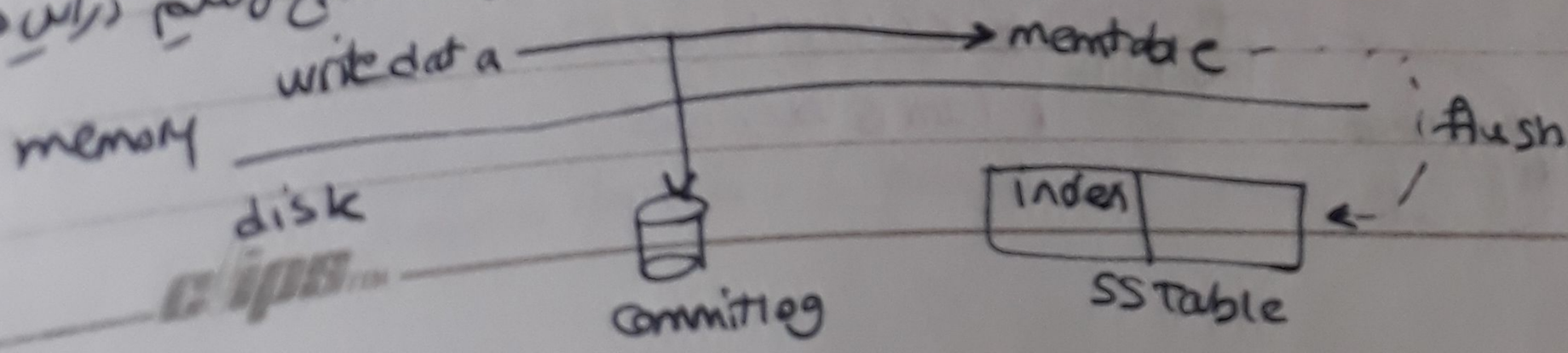
② Network Topology = در نود بعدی در Rack بعدی به صورت ساعتگرد قرار می‌دهد. این باعث availibity می‌شه یعنی در DC چند دفعه در یک نوری از Rack بعدی می‌نویسید.



زیرساخت ذخیره‌سازی کاساندرا =

چگونه داده را ذخیره می‌کند؟ write خیلی ساده و سریع

memtable داخل رم است. یک فایل commitlog هم دارد. وقتی داده رو می‌کنیم در این‌ها می‌نویسید.



رم محدودیت دارد. یک زمان‌های خاصی sort شده بر اساس کلید memory را در SSTable می‌ریزد.

به ازای هر table یک memtable و SS table داریم. داخل هر table ... data table,

یک سری dump SSTable، نوشته شده. چرا هنوز ~~[illegible]~~ خالی بود اینجا بودی چون هنوز از ram توی disk نریخته بود. در زمان‌های خاص این کار می‌دهد.

وقتی یک query می‌آید همه‌ی SSTable‌ها را باید ببینید. چرا؟ چون هر update در یک dump قرار دارد. سه بد:

راه

compaction: SSTable را بزرگ دارد و merge می‌کند در یک SSTable جدید با timestamp

کی‌ها انجام می‌شود؟ در Yamel. ست می‌کنیم و اتوماتیک انجام می‌شود.

مدل‌های compaction =

default:

① size-tired

هر مثلاً "4" تا SSTable ساخته شد merge می‌کند. همسایه‌ها را تجمیع می‌کند.

بدجاهایی که write زیاد است خوب است (سخت نفوذ!)

② leveled (LCS)

چگونه؟

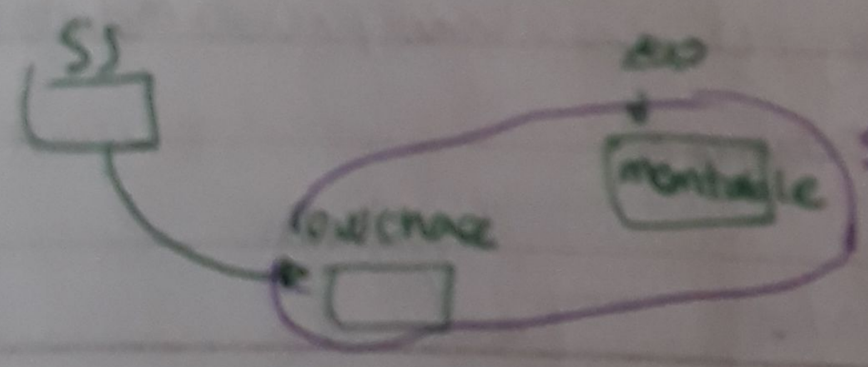
read زیاد (۱۴)

③ Data tired (DTCS)

برای داده‌های timeseries خوب است!

وقتی سطح‌ها ساختن بیانند می‌رود از commitlog می‌خوانه و آنچه در SSTable‌ها هست را وارد می‌کند.

فضای read پیچیده‌تر است. row chace = بخشی از SSTable‌ها در آن است.



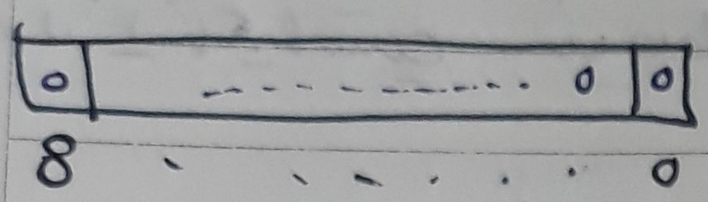
از این‌ها جواب بگیرید بخشی از فضای ram را به

Subject ..... Date .....

Create table tbl2 {'id int, primary (id)) with caching={'key', 'All', 'row per partition'}

row cache حتماً سیک شده با sstable هاست.

hash2 | hash1



data structure (bloom filter): یک آرایه‌ی یک بعدی.

وقتی write می‌کنیم id=20 وارد توی hash function و باقی‌مانده‌ی جواب حاصل از آن

hash1 و hash2 را می‌یابد و درایه‌های x_1 و x_2 را یک می‌کند.
x_1 x_2

Id=3 را بخواهیم بدانیم به hash ها می‌رود و جواب hash ها را می‌بیند اگر هر دو درایه ۰ باشد می‌گه حتماً نیست وقتی ۱ شد می‌گه احتمالاً وجود دارد چون ممکن است با id دیگری ۱ شده باشد پس کار این فیلتر شبیه قاطع می‌گوید نیست وجود داشتن را با شک می‌گوید.

اگر سایز آرایه را خیلی کوچک بگیریم که همه را می‌گوید هست پس بد هیچ بزرگ تر پس بهتر جواب می‌دهد. ضرر = فضای زیاد از دست می‌دهیم.

key-cache:

key	offset
10	15

offset | جای آن یافت می‌شود.

partition Summary در memory است و

heap size را تعیین کنیم ← Max heap = 64GB پیشنهاد خود کاساندرا که خوب است.

کاساندرا با جاوا است

garbeg collection GC

اگر JMX را در جاوا فعال کنیم * Jconsol همراه با JDK می‌توان مشاهده کرد

CMD
Jconsole

اطلاعات heap و garbes collector و ...

دو مدل دارد

G1 - قدیمی: با کارش تموم نشه نمی‌شه کار جدید کرد.

CM ۲- جدید: همزمان کار می‌کند و عملکرد را نشان می‌دهد. ولی تاثیر دارد.

به خاطر عملکرد GC می‌توان حتی heapsize را تا کل ram گذاشت

Subject ..... Date .....

پارامترها

* Cassandra stress tool: tools/bin/cassandra-stress write

* می‌توان کاساندرا را تست کرد!! *

چه آیدی توی کاساندرا می‌نویسه. read ، mix (چند درصد write چند درصد read)

✓ از 1 تا 5 باشیم. یک جدول بسازید rep=3 ،

یک سرور دیتا نت دارد. 5 تا فایل هر کدام 10 تا مگابایت باشد

10 میلیون رایت انجام دهید.

با دستور schema

heap size و cache ، bloom filter size ، secondary index تعریف کنیم

بعد از هر کدام از این‌ها چه میلیون read بفرستید و (tps) یا (response time) چه تغییری می‌کند؟

با دستور graph شکل کل کارها را نشان می‌دهد.

مکانیزم‌های repair کاساندرا = چه‌طور یک نود همیشه موقع write بعد از آن بالا بیاید. کی باید داده را بگیره از بقیه؟

در هنگام شکستن یک فضای در نظر گرفته می‌شود. زمانی که یک نود افتاده باشد در والیدکتوری hint خودش می‌نویسد.

در نود coordinator

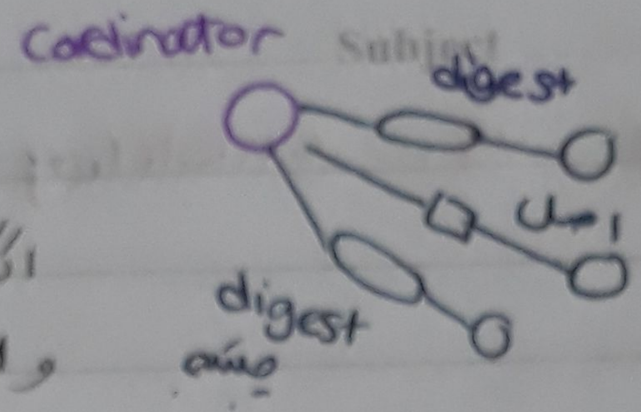
hint = hand off

وقتی نود مرده درباره بالا بیاید coordinator از hint خودش آن ریپلیکه و پاک می‌کند hint را.

ls -h در لینوکس فولدری را رصد می‌کند.

→ data/hint

بیایم قبل insert یک نود را بیاندازیم آن وقت در این فولد می‌بینیم که file hint با محتوای باتری ساخته. حالا هر وقت نود بالا بیاد می‌آره توش می‌ریزه و این hint را پاک می‌کند.



- Read repair =
اگر تفاوت ببیند که چیزی دیگر نه max می‌گیرد و اون متفاوت را درست می‌کند بعد با بقیه sync می‌کند.

- Anti-entropy repair : جوابیز D :

جلسه چهارم :

Subject ..... Date .....

partition key vs. clustering key:

model/table/entity driven — نمودار ER : شمای کلی پروژه را مشخص می‌کند.

داشتن این نمودار ها برای طراحی کاملا مناسب نیست چون باید به query توجه کنیم!

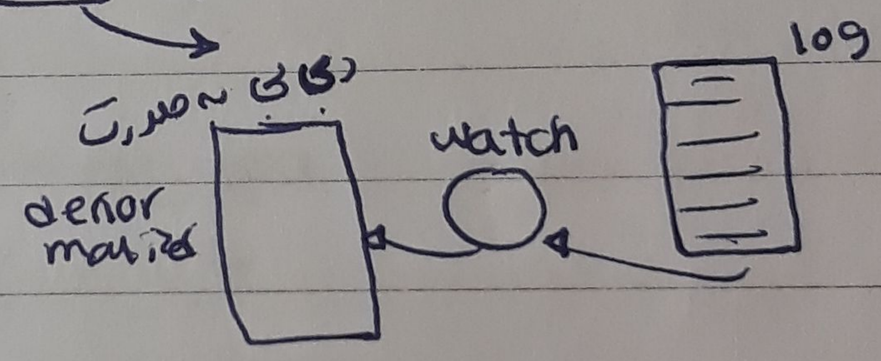
No join / No primary & foriengkey

Denormalize → در اینجا نیاز داریم که

فلسفه‌ی CQRS + Command Query Responsibity Segregation

event sourcing / جداسازی approach برای خواندن و نوشتن

log به append

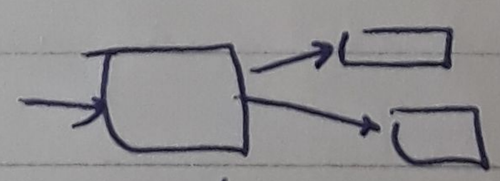


فرض کنیم 3 تا query داریم : راه ۱= با یکی primary table به Secondary

۲ ساخت view

۳ → performance poor — ۳ تا جدول جدا ذخیره‌سازی کل داده با کلیدهای مختلف در جدول مختلف

۴



با جواب کوئری اصلی به جواب بقیه دست بیابیم.

- مقدار داده‌ی ذخیره شده
- سرعت query

* Cassandra Client Driver
* Cassandra datastacks jar mvn

lfu = least frequency used — کم تر استفاده شده

8 حالت داریم :

volatile → lru 1 4 / lfu 2 5 / random 3 6 → volatile-ttl 7 → noeviction 8

allkeys

کلیدی که TTL آن به زمان انقضا نزدیک تر است را دور بریزد.

*Replication = slaveof <masterip> <masterport>

در یک ساختار اطلاعات مستر را می دهیم ولی اگر master بیاید کاری نمی کند. روی slave ها می توان ~~write~~ read کرد اما ~~read~~ write نه. read نتها از طریق master. هیچ مدیریتی ندارد.

*Redis cluster: با فعال کردن آن از Standlone mode به cluster mode می رسیم.

Pipelining

فرض کنیم که در هنگام کار با JAPI هر بار set بفرستیم مثلاً 7 تا set. می توان به جای آن یک pipe باز کرد و کل 7 تا دستور آنجا فرستاد ← هدف: افزایش performance

atomic نیست

jedis.flushAll → کل کلیدهای redis را خالی می کند

شبیه drop در RDBMS

Pipeline ~~pipeline~~ p = jedis.pipelined();

p.set("u", "1")

p.set("p", "2")

p.sync(); → حالا همه اجرا می شود.

p.syncAndReturnAll — p.get هم داریم

در Redis یک دستور monitor هست اگر بزنیم دستورات ارسالی را می نویسد.

Multi — redis بعضی کارها را باید همزمان single thread بودن atomic است و باید پیاپی تا این تمام شود.

Transction t = jedis.multi();

Subject ..... Date .....

t.set( ) List<Object> lst = t.exec();

جلسه‌ی ششم : ۱۰ ، ۱۱ ، ۹۷

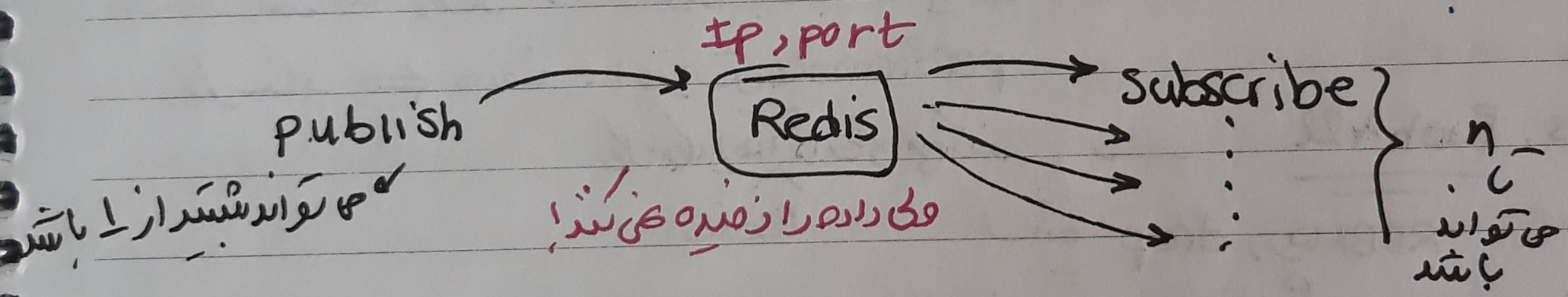
- Subscribe C1
- publish CH1 Salam

* بسم messaging در Redis :

redis یک نقش broker دارد که یک terminal سمت به redis نقش subscriber و روی دیگری publisher است.

فرق آن با kafka ، کل داده‌های قرار داده شده روی کانال را تا یک مدتی نگه می‌دارد
مثلاً اینجا در redis اگر یکی دیگر subscribe شود داده‌های از اولی که آن subscriber دیگر گرفته را نمی‌بیند اما در kafka اینطور نیست.

سرعت این ویژگی redis خیلی خوب است. جایی که یک session برقرار باشد و بعد از کمی بسته شود خوب است.



redis-5.0 /redis.conf | grep bind SubMsg extends

کلاسی به اسم JedisPubSub می‌سازیم و با overwrite کردن متد

onMessage(String channel, String message)

- onpung = کل ping-pong رفتار
- OnpMessage (pattern → مثلاً *!.... ، مثلاً کلمه‌ی news داخلش باشد) می‌گیرد

psubscribible ? ___ → فقط پیامی با الگوی ___ را بگیر

SubMsg sub = new SubMsg();
jedis.subscribe(sub, "ch2");

در redis.conf باید در bind ، Ip دستگاهی که روی آن هقب تو را نوشت .

Subject Date

class PubMain ;> → main: Jedis jedis= new Jedis("Ip");}
for(i=1;i<100;i++)
jedis.publish("ch2", "from pub"+i);

redis برای authentication در ۶ مکانیزم گذاشته ولی خیلی خوب نیست :D

configset protected mode no

در redis.config ، پیش‌فرض yes است

یعنی وقتی روی سرور هقب است می‌توان از بیرون ماشین آن به‌شت دسترسی یافت .

requirpass 123 انواع redis user ندارد فقط یک پسورد در این جا می‌نویسیم و هرکسی آن را داشته باشد می‌تواند استفاده کند .

redis-cli -h 192.168.214.120

AUTH 123 بهترین شرایط : redis باید یک Application باشد .

redis-cli config با این می‌توانند یکی protected mode را از بیرون ببندد .

redis کاری کرده که بتوان دستوری را rename کرد، حلوی این کارها را گرفت

در .conf می‌توان از rename-command استفاده کرد .

rename-command Config "1"
abcztrs2@zzt.ab

یا خالی اش کنیم انگار حذف کنیم یا با یک دستور اختصاصی جایگزین کنیم .

می‌توان دایماً هی AUTH های جدید با پسوردهای جدید تست کرد ← امکان brutforce وجود داشت

می‌تواند انتخاب کند 150k per sec

مانع طول پسورد و سختی آن را باید زیاد کرد تا نشه به دست بیارن .

Redis benchmark: معیاری memory ، نفع استفاده در ... را می‌توان تست کرد

Subject ..... Date .....

lmset , hpop

redis-benchmark -h IP -a 123 -t set,get -P 20 -d 200

هر ۲۰ دستور یک pipe داشته باشیم.

اطلاعات ارسالی 200 بایتی هستند

-n 200 — تعداد کلاینت ها را 200 تا می کنیم

select 3 → 3 db ← name space

می رود به این شکل set و get ، Key های مختلف در db های مختلف داریم.

* redis هم مکانیزم های برای scale دارد. ۳ تا مود برای این منظور دارد:

① Masteronly

② یک Redis Master داریم که می توان چندین تا slave به آن اضافه کرد. Master slave.

می توان از هر ۳ خوانده شود اما نوشتن فقط در master است

همه ی دستی در اگر Master بیاید یکی متوجه نمی شود.

③ Master slave Multi-level

④ Master slave with sentinel

Redis topology - Sentinel

یک master و هر slave یک sentinel بالا می آید که به محض افتادن master یکی از این sentinel ها نقش آن را بر عهده می گیرد.

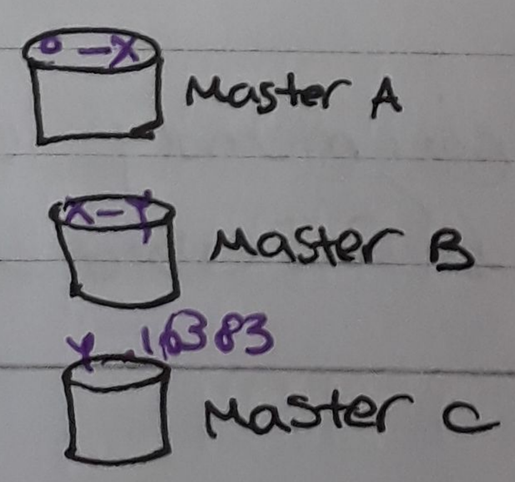
اما باز هم هنوز data partitioning ندارد.

تا الان فقط HA و fault tolerance داشتیم.

ⓐ وقتی که سایز کلیدها بزرگ است و یک ماشین جواب نمی دهد ⇐ redis cluster

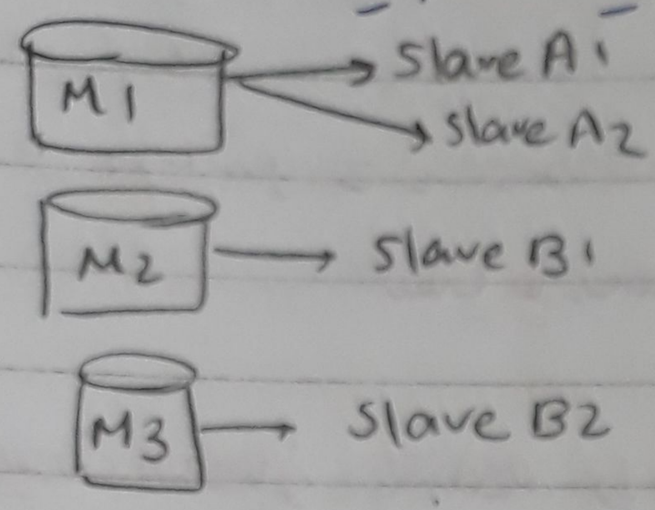
چندین master داریم و هر master چندین slave می توانند داشته باشند و هم data Partition داریم.

redis در بحث clustering یک مفهوم slot دارد که از ۰ تا 16383 است همیشه CRC (شبیه hash function MD5) کلید را به آن می دهد و باقی مانده آن بر 16384 گرفته و عددی بین ۰ تا 16383 می دهد که بر اساس آن ها در Master های مختلف می ریزد



* در حالت دفالت سینک شدن redis به صورت asynchoze است ←

هنگام کار با redis احتمال dataloss با تنظیمات مختلف زیاد است باید حواسمان باشد.



فرض کنیم یهو موقع کار M3 از کار بیافتد آن گاه SB2 آن master می‌شود وقتی دوباره M3 بالا بیاید نقش آن الان یک slave می‌شود.

هنگام کار با redis در حالت cluster باید در config این تغییرات انجام شود:

port اولی که داشته
port+10000 برای صحبت Master, slave است
} cluster در حالت redis port
در firewall این دو تا باید set شود.

① cluster enabled yes
② cluster-config file (آدرس) → node-6379.conf
③ cluster-node-timeout 7989 (به میلی ثانیه) → وقتی یک master می‌افتد به این اندازه صبر می‌کند اگر master برنگشت slave را به آن عنوان متول می‌کند. ← در این زمان صبر write های روی این نود را متوقف می‌کند اما read را جدا.

④ appendonly yes → برای داشتن مکانیزم persist
⑤ protected-mode no
⑥ port 7000 یا 8000 یا ...

mkdir redis-cluster (داخل این) mkdir node1 mkdir node2

vi node1/redis.conf → اطلاعات بالا را ⑥ تا بیست می‌کنیم
vi node2/redis.conf → همان 6 تا می‌گذاریم فقط مقدار port را تغییر می‌دهیم

این اطلاعات را به ماشین های slave1 و slave2 هم scp می‌کنیم.

* حال در master وارد فولدر node1 & 2 redis-cluster شده و با دستور زیر اجرا می‌کنیم:
redis-server redis.conf
running in cluster mode می‌شه

Subject ..... Date .....

* حال در master / slave1 , slave2 برای node1 , node2 به این شکل redis را بالا می‌آوریم.

حال هر ماشین 2 تا instance داریم ← مثلاً 6 instance

netstat -pltun → لیست پورت‌های باز را نشان می‌دهد
در لینوکس

پورت‌های باز redis را نشان می‌دهد ← | grep redis

node-config appendonly.aof } مثلاً بیایم این‌ها در فولدر node1 دیده می‌شود.
های که خودمان برای persist
گفتیم

در src یک script با زبان rubby نوشته شده redis-trib.rb
برای شناسایی slave و master ● اجراش سخت بود :D
کار با redis version 4 باید rubby و gem redis را نصب کرد.
در redis version 5 کل این فرآیند به redis-cli اضافه شده

redis-cli --cluster create --replica-factor=1 192.168.234.120
:7000 192...:8000 ——.120:7000 —— :8000 xx:7000 xx:8000

اگر همه چی درست باشه ← slot بندی را انجام می‌دهد و پرسیست می‌کند

Adding replica ——:8000 to xx:7000
این ماشین را slave این ماشین ⊛ می‌کنم. ⊛ Master

آخرش تایید می‌گیره که این کارها را کردم.

هم همه باید یکی باشد تا sync شود ؟؟ دقیقه‌ی 37 درست متوجه نشدم.

redis-cli -c -h IP -p 7000
یعنی در مد cluster داریم کار می‌کنیم.

حال set k1 v2 ← هر جا که slot ای که بذارز
redirect می‌شویم به IP ماشینی که برای این slot است

66086019

Subject ..... Date .....

* cluster INFO
* cluster Slots
* claster keyslot k2 — مربوط slot این کلید عددش چنده!
* cluster nodes
* cluster replica ID یک ماشین → slave های آن ماشین را نشان می‌دهد

```
Set<HostandPort> hosts = new HashSet<>();
RedisCluster jedisCluster cluster = new JedisCluster(hosts);
hosts.add(new HostAndPort("192....", 7000));
```

سه تا master ها را می‌گوییم

= Connection pooling — وقتی DB بالا می‌آید می‌گوییم چندین تا Connection اول بسازد
وقتی هر کی جدید می‌آد از همین‌ها بدی دارد کارش تموم شد به pooling
آن را اضافه می‌کند.

spark-submit --class ( ) --master spark://master:7077
disk/ [ ]

با nodetool می‌توان sstable را گرفت در SDB که مثلاً keyspace و table
وجود دارد دیفلت کلی وقتی در سرور دیگری می‌رویم که هیچی ساخته نشده باید با دست
از نو همه را ساخت و بعد داده را import کرد تا partitioning را بشناسد

Subject ..... Date .....

جلسه‌ی هشتم : ۹۷,۱۲,۹

نصب mongoDB :

اول باید آدرس mongo را داد.

① yum install mongodb-org

② cd RPM

mp -ivh *

پکیج ها را بگیریم و اینجوری نصب کنیم :

تمام پکیج های داخل فولدر فوق را نصب می کند.

/etc/yum.repos.d/centos_base.repo

systemctl status mongodb

فعال کردن

netstat -pltun → * می‌بینیم که 27017 پورت باز mongo هست.

server

database

→ contain Collectors

→ contain documents → از جنس JSON

* بعد از نصب دستور mongo می‌زنیم

use db1 → db1 یک دیتابیس به اسم db1 می‌سازد

db.user.insert (("name": "A")) ← : Collection برای ساختن و وارد کردن یک doc :

یک doc از user Collection که در db1 است

فیلد name و value A ساختیم.

db.user.find () → مثل select * است.

فیلد '_id' در هر row db در mongo باید وجود داشته باشد می‌توانیم ما بدهیم یا خودش می‌زند.

insert ("id": "123", "name": "B")

db.user.find (("name": "A"))

base64 → یک binary object را به یک tex تبدیل می‌کند.

Subject Date

mongo به این شکل داده‌ها را ذخیره می‌کند ولی به صورت JSON به ما نشان می‌دهد

bson → binary json

```
.insert("tags": ["t1", "t2", "t3"])
.findone(  )
```

→ اولین سند به order ذخیره را برمی‌گرداند.

find → یک cursor برمی‌گرداند و مجموعه‌ای از data است.

findone → اولین objective ای که با این جمع می‌شود را برمی‌گرداند.

```
* var var1 = db.user.find(("name": "ali"))
  while (var1.hasnext()){
      print(tojson(var1.next))
  }
* find( ).limit(20)
* find( ).skip(2).limit(20)
```

→ دو تای اول را نادیده گرفت

pagination : فایده

skip(1xn) limit(n)

2xn

استفاده از & در query:

```
.find(($and, ("name": "A"), ("location": "Tehran"))
```

$and
$or

: update روش

آن doc یک field

```
.user.update("  ": "  ", ($set), )
.update(                    )
```

به جای set جدید اضافه می‌کند

find می‌رود و replace می‌کند

remove → تمام داده‌های db را پاک می‌کند

```
db.user.remove(("username": "A"))
db.user.drop()
```

db جدول user را پاک می‌کند.

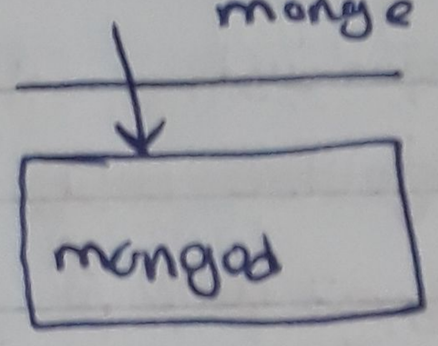
mongo به دنبال شبیه BTree نگاه می‌کند می‌توان به هر فیلدی index زد.

```
for(int i=0; i<2000; i++){
    db.numbers.save(i)}
                 (("num": i))

db.numbers.count()
.find(("num": ("$gt": "
```

توی field خاص، create index میزنیم :

وصل شدن با shell



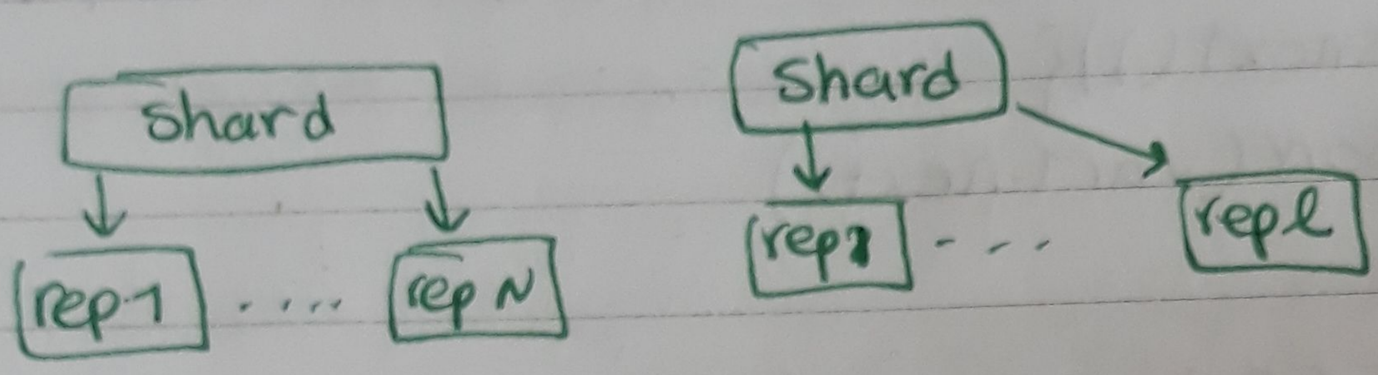
در مکانیزم داریم در nosql ها :

sharding و replication

" "

همه‌ی داده در یک جا ذخیره نمی‌شود و بخشی در یک جای دیگر

کپی کل داده



معماری سیستم‌های توزیع شده ← masterless ← همه‌ی نودها shard بودند و replica کاسندرا

← masterslave

shard key ← چه‌طور انتخاب بکنیم ؟ از همان id باشد.

چگونه shard ها از هم باخبر شوند ؟

syn

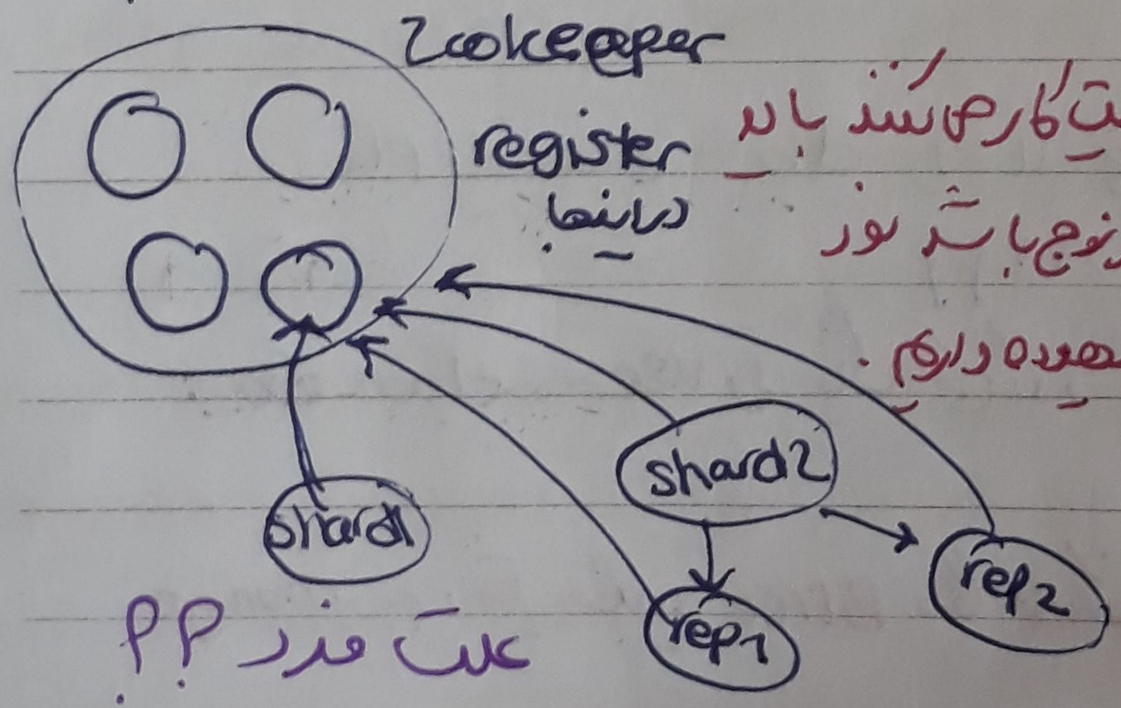
leader en کی مستر باشه

Config management

distributed log system

live nodes ?

مجموعه‌ای فلت Conf node بالا می‌آوریم که مدیریت metadata سرور را برعهده دارد.

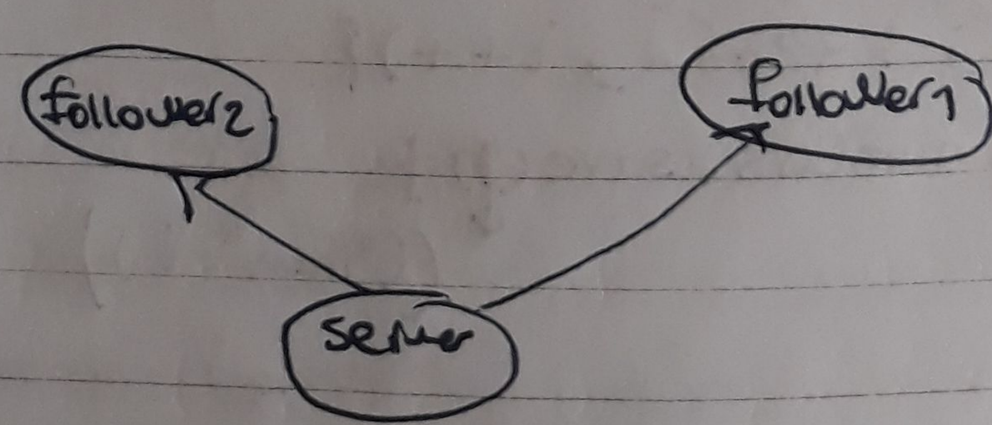


ابزار Zookeeper ← چون روی رای اکثریت کار می‌کنند باید فرد باشند. اگر زوج باشد نود افزونگی بیهوده داریم.

علت فرد ؟؟

طاقت یک Cnfgser لازم است.

دقیقه 15 بعد از break



/etc/resolv.conf تغییر dns هدر شن

/etc/mongo.conf → # replication :

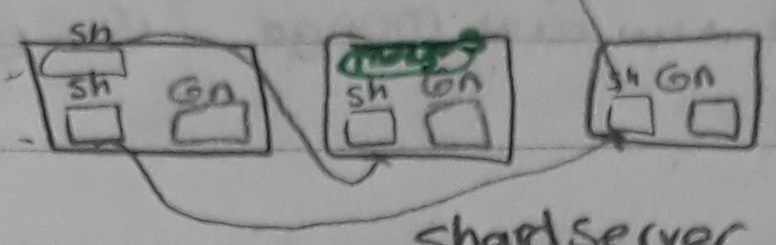
Sharding : مشخص بدکن نقش اینه

shardserver shard هستم confserv

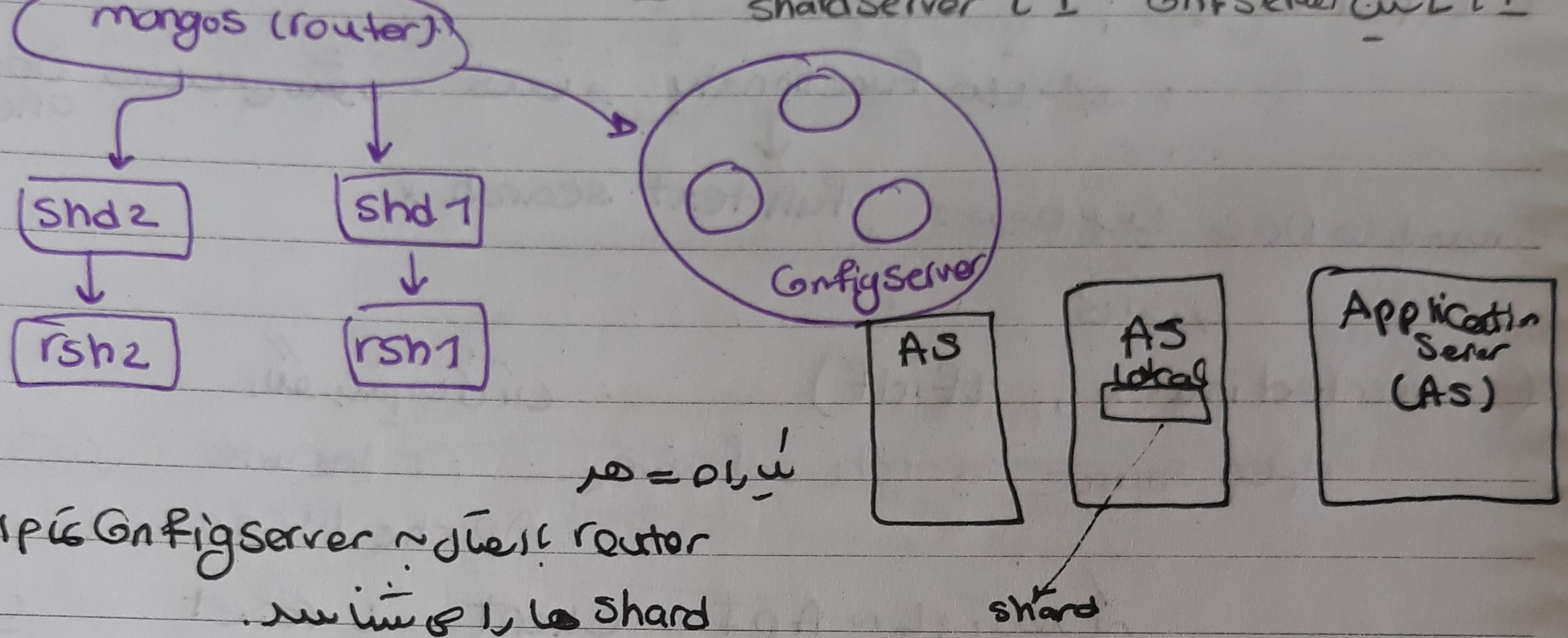
هستم کی

cluster role =

port مشارکت



shardserver تا ۴ ۳ تا ماشین configserver



یک راه = هر

اطلاعات configserver با اتصال به router

shard ها را می شناسد.

* mongod --config ..../mongo.conf

182 190 153 جلسه نهم ۹۷/۱۲/

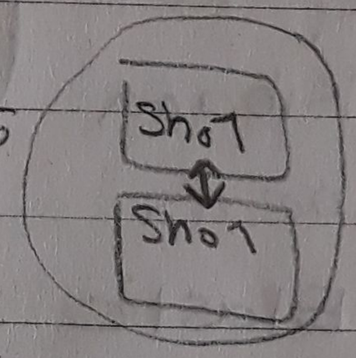
182	190	153
Cnfgserver shard1 ~~mongos~~	shard1 → 28000 shard2 → 29000	Cnfg-Server shard2 → 29000 monogos 123

→ 28000

Cnfg 2701

Path
dbpath
longpath

تداوم shard بشین



* دستور mongos : mongos --help

--configdb → replconfig / آدرس ماشین ها

بهترین performance زمانی است که query روی shardkey ها زده شود.

بهترین = روی یک ماشین یک Config.server و دو تا shard بدون replica server و یک mongos بالا بیاورید. یک usercollection ساخته و قسمتی از آن را به

Subject ..... Date .....

عنوان shard key نگذارید. مثلا data insert کرده و find کنید.
یک جاهایی خطا می‌دهد چون باید یک چیزهایی را رعایت نمایید.
معمولی نیست!

* engin ها و برخی ویژگی های mongo گفته نشد :D

Elastic :

Doc oriented حساب می‌شود. هدف و functionaly فرق دارد.
↓
full text search *

مرتبط‌ترین Doc ها را به دست می‌آورد :

الگوریتم‌های آن : (inverted index , tfidf)

پروژه‌ی : lucene ← open source search software
Apache lucene

① یک elastic
reset با client API به ما می‌دهد که
set Get Delete Update

② solr
می‌توان روی چند instance بالا می‌آورد. مکانیزم sharding وجود دارد.
(solr ← masterless است)

logstash
Kibana
beats
Elastic
} ELK

elastic search روی پروژه‌ی lucene بالا آمده است
به زبان جاوا است
برای مقیاس‌پذیری افقی و reliability طراحی شده است.

distributed / High Availible / opensoure / full text search

فایده‌ی Elk = یک سری data source داریم که دائما در حال تولید داده هستند. این داده‌ها می‌توانند event باشند. ممکن است بخواهیم یک سری metric از منابع را جمع کنیم (مثلا رم آن ها) می‌خواهیم زنده بودن server و استیت

clips

Subject

راه‌اندازی کنیم . filebeat / packebeat / log beat / Mybeat

خودمان با extend کلاس
beat برای کار خودمان
beats/file/TCP/redis/

input
اطلاعات beat

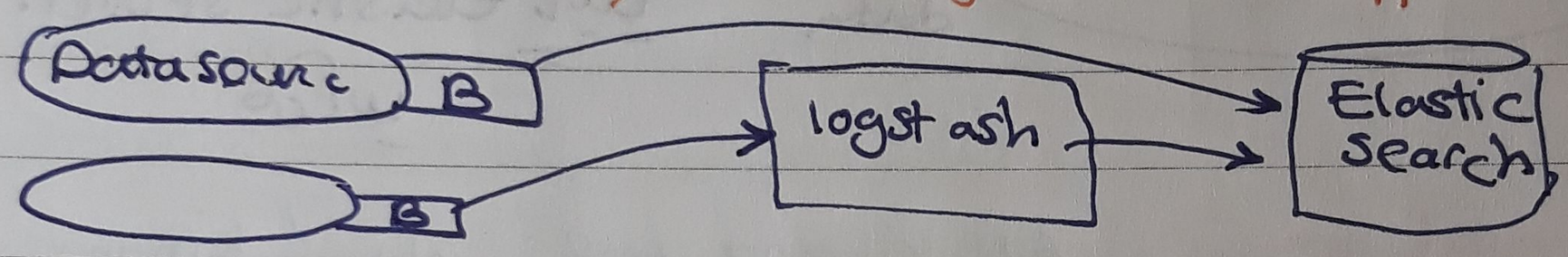
logstash* :

output = می‌تواند elastic باشه

filter = drop کنیم ، فرمت تغییر دهیم ، بخشی از text را بردارم
در واقع لایه‌ی transform هست .

(nagios ، zabbix) ← ابزارهای مانیتورینگ

elastic cloud
x-pack } یکی
→ usermanagment / security / monotoring / Graph



الاستیک را نمی‌توان با کاربر root بالا آورد !!

adduser elastic user -m passwd elasticuser

chown -R elasticuser:elasticuser آدرس فایل elastic Search

su elasicuser → حالا می‌توان از طریق فولدر bin داخل elastic ، سرویس elastic Search را بالا آورد .

netstat -pltun | grep java
سال الاستیک → 9200 / 9300

elasticsearch.yml
jvm.options → heap → Xms 2g initiale / Xmx 2g max
→ GC →
networkhost

در GC بهینه‌سازی ای و cms وجود دارد .

دو خطا می‌خوریم : ① تعداد openfile های linux :

stdin
stderror stdout
مثل شکول

clips™

این‌ها واسط‌های استاندارد کار با linux است. / server / file / socket
disk ... در نیکوس باشید همه این استاندارد را پیاده کرده‌اند.

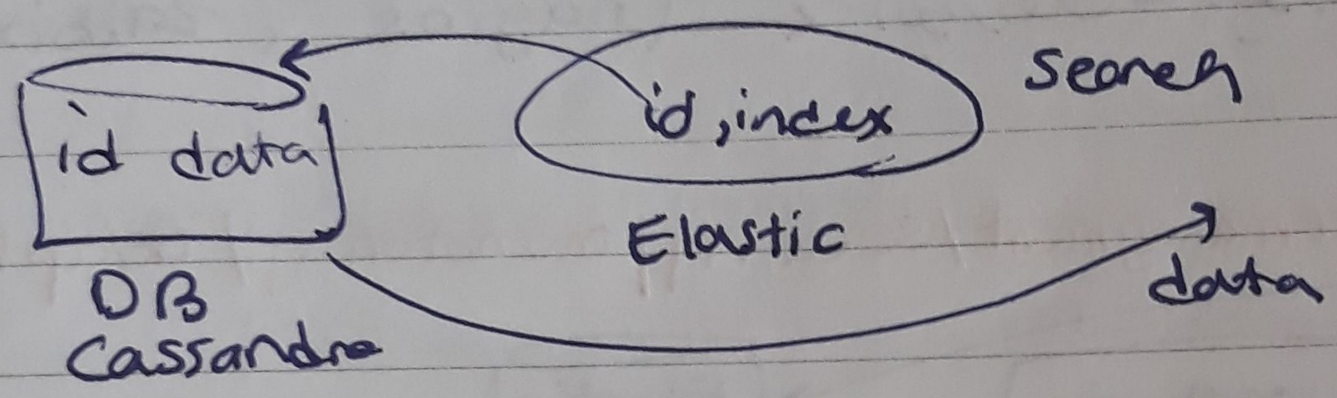
Elastic / kafka / Cassandra

* ulimit _ a
تعداد فایل‌های بازی که نیکوس می‌تواند باز کند

nano etc/security/limits.conf

max virtual memory ②

sysctl -w vm.max_map_count=270000



: Elassandra ✓
اینجوری performance
elastic search بین
می‌آید

در kibana تنظیمات kibana.yml داریم ←

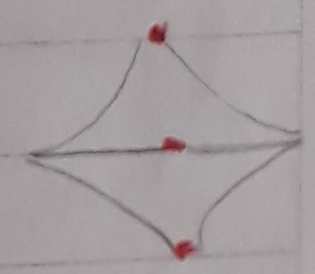
elasticsearch . url = " "

Server host

خودش رنگی می‌آید

port پیش‌فرض
5601

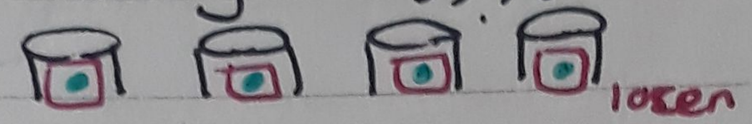
Dev tools in kibana → rest کار با



جلسه‌ی دهم :

* Devtools در Kibana :

کی سراغ Sharding ؟ حجم دیتا زیاد باشد . کاربرد : partitioning Data



هر چیزی که تایپ JSON داشته باشد در الاستیک Document گفته می‌شود .

put → باید خودمان ID را بدهیم

post → خودش ID می‌دهد type

فرق text و keyword :

text ↓ کلمه را index می‌کند جوری می‌کند

keyword : عین کل کلمه را می‌گیرد و با tokenized چیپ نمی‌کرد

* روی فیلدهایی که از نوع text هستند index انجام می‌شود *

type text → بیشتر از 256 کلمه را به صورت دیفالت ignore می‌کند اگر بیشتر لازم داریم باید در تنظیمات تغییر داد .

* score به ازای search انجام شده برمی‌گرداند .

GET ___

{

"query"

},

✓ "from" :

✓ "size" :

} از چیزی نشان بده چه تعداد نشان دهد } برای Paging خوب است .

✓ sort : {

براساس چه چیزی

✓ highlight → آنچه دنبالش گشتیم را در text مشخص می‌کند و می‌توان در سفحه‌ی وب هایلایت کرد مثلاً !

در Solr وقتی shard تعریف می‌کنیم هر shard روی یک ماشین می‌افتد . در elastic data node و master node داریم

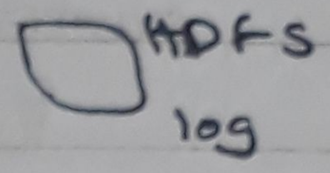
← نقش مدیریتی

"aggs"

" stats : " min, max, average

/opt/kibana/config/kibana.yaml ← برای بهترین سطوح دسترسی authentication

Subject ..... Date .....

event sourcing (CQRS)

دیتا سورس یک سری event دارند تولید می‌کنند.