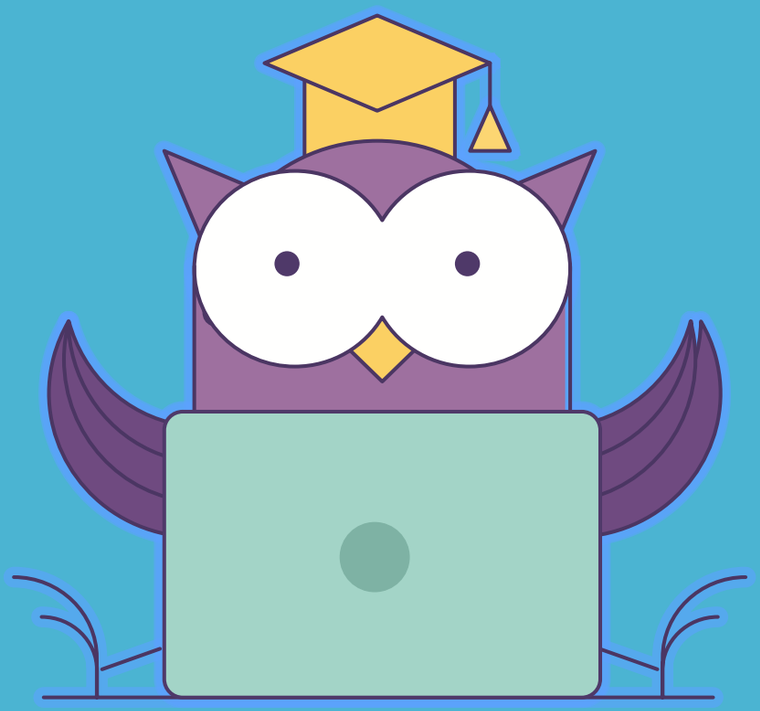
ОНЛАЙН-ОБРАЗОВАНИЕ

Шардинг (часть 1)

Юрочко Юрий

OTUS

Как меня слышно и видно?



> Напишите в чат

+ если все хорошо

– если есть проблемы со звуком или с видео

Преподаватель

Юрочко Юрий

- окончил МГТУ им. Н.Э. Баумана в 2016 (ИУ-7)
- 5+ лет разработки на C++/Go
- руководитель команды Go

Цель занятия

- Узнать, что такое шардирование
- Понять, что решает и зачем нужно
- Познакомиться с видами и стратегиями шардирования
- Обсудить проблемы и узкие места различных подходов
- Поговорить о решардинге, консистентном хешировании

Проблема

- бизнес растет -> данные растут
- время обработки запросов растет
- сервера ограничены (CPU, RAM, Disk, ...)

Вертикальное масштабирование ?

Вертикальное масштабирование

- докупить то, чего не хватает (CPU, RAM, Disk, ...)
- поможет (скорее всего временно)
- есть предел

А может как-то нарезать данные?

- берем данные и разделяем по какому-то признаку
- разделенные данные физически лежат отдельно
- все в пределах одного сервера!
- бывает разных типов
- в теории, должно работать быстрее (?)

- key
- range
- list
- hash

Пример MySQL

```
CREATE TABLE members (
    firstname VARCHAR(25) NOT NULL,
    lastname VARCHAR(25) NOT NULL,
    username VARCHAR(16) NOT NULL,
    email VARCHAR(35),
    joined DATE NOT NULL
)
PARTITION BY RANGE( YEAR(joined) ) (
    PARTITION p0 VALUES LESS THAN (1960),
    PARTITION p1 VALUES LESS THAN (1970),
    PARTITION p2 VALUES LESS THAN (1980),
    PARTITION p3 VALUES LESS THAN (1990),
    PARTITION p4 VALUES LESS THAN MAXVALUE
);
```

https://dev.mysql.com/doc/refman/8.0/en/partitioning.html

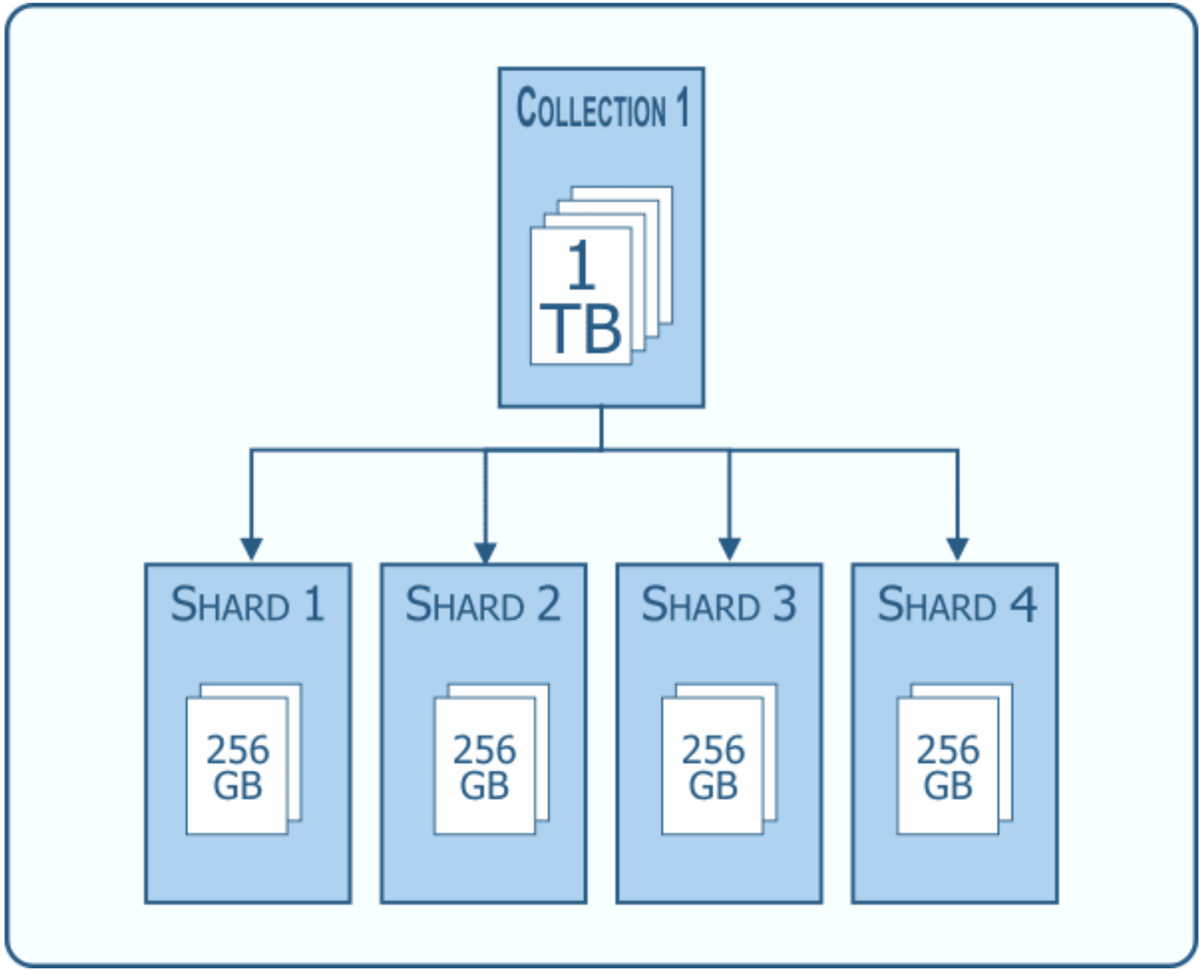
Пример Postgres

```
CREATE TABLE measurement (
    city_id         int not null,
    logdate         date not null,
    peaktemp        int,
    unitsales       int
) PARTITION BY RANGE (logdate);

CREATE TABLE measurement_y2006m02 PARTITION OF measurement
    FOR VALUES FROM ('2006-02-01') TO ('2006-03-01');

CREATE TABLE measurement_y2006m03 PARTITION OF measurement
    FOR VALUES FROM ('2006-03-01') TO ('2006-04-01');
```

https://www.postgresql.org/docs/10/ddl-partitioning.html

- может ускорить запросы
- может замедлить запросы

Не решает проблему ограниченности сервера!


COLLECTION 1
1 TB
SHARD 1
256 GB
SHARD 2
256 GB
SHARD 3
256 GB
SHARD 4
256 GB

Шардирование

- один из вариантов масштабирования
- разбиение данных на куски
- данные живут физически на разных серверах (ну обычно так)
- не репликация - !
- не партиционирование - !

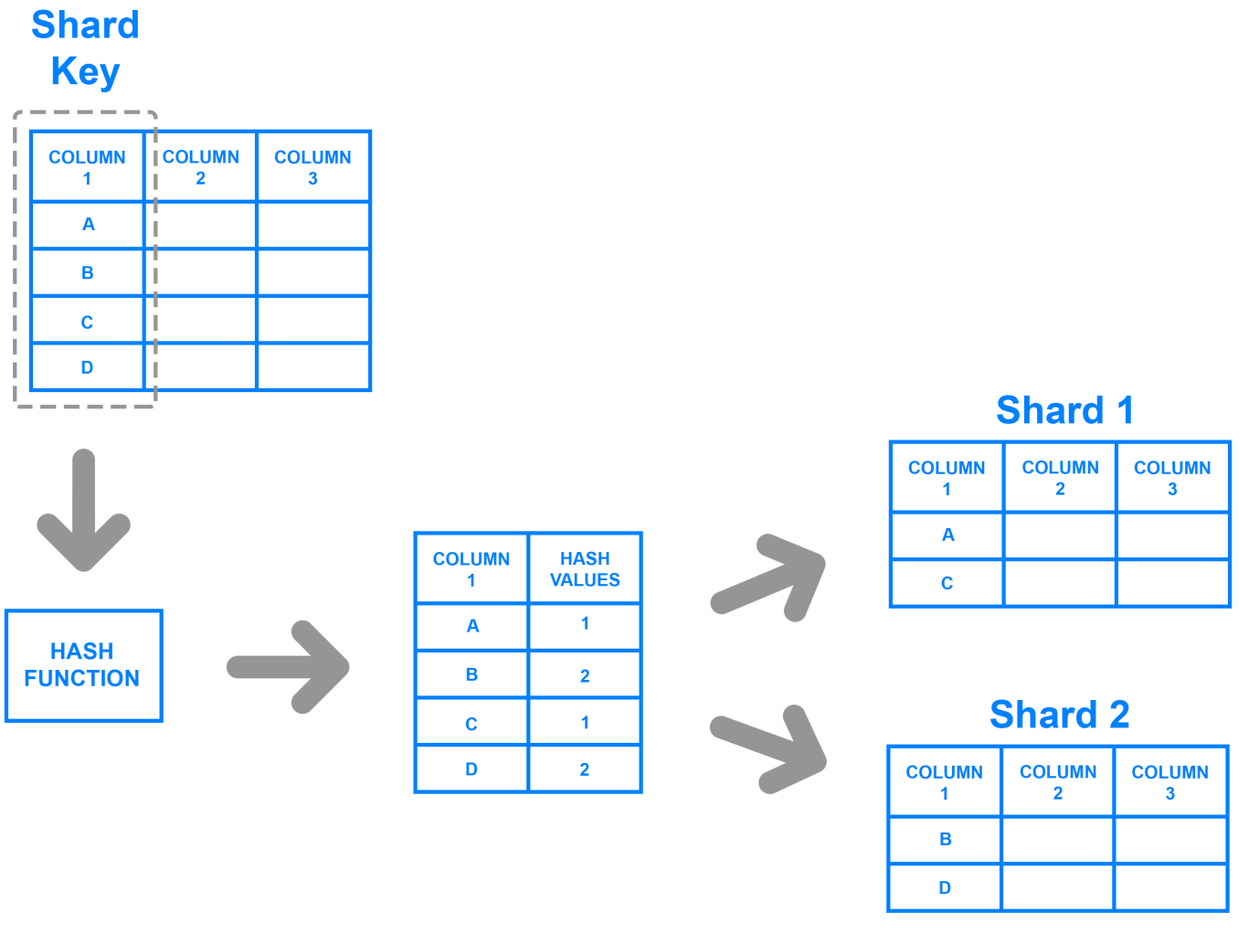
В итоге для чего оно вообще?

- горизонтальное масштабирование
- ускорить обработку запросов (особенно запись!)
- повысить отказоустойчивость(*)

Вопросы?

Как шардить?

- посмотрите на свои данные
- посмотрите на свои запросы
- выберите правильно критерий шардирования - вам с ним жить!


Shard Key
COLUMN 1
COLUMN 2
COLUMN 3
A
B
C
D
HASH FUNCTION
COLUMN 1
HASH VALUES
A
1
B
2
C
1
D
2
Shard 1
COLUMN 1
COLUMN 2
COLUMN 3
A
C
Shard 2
COLUMN 1
COLUMN 2
COLUMN 3
B
D

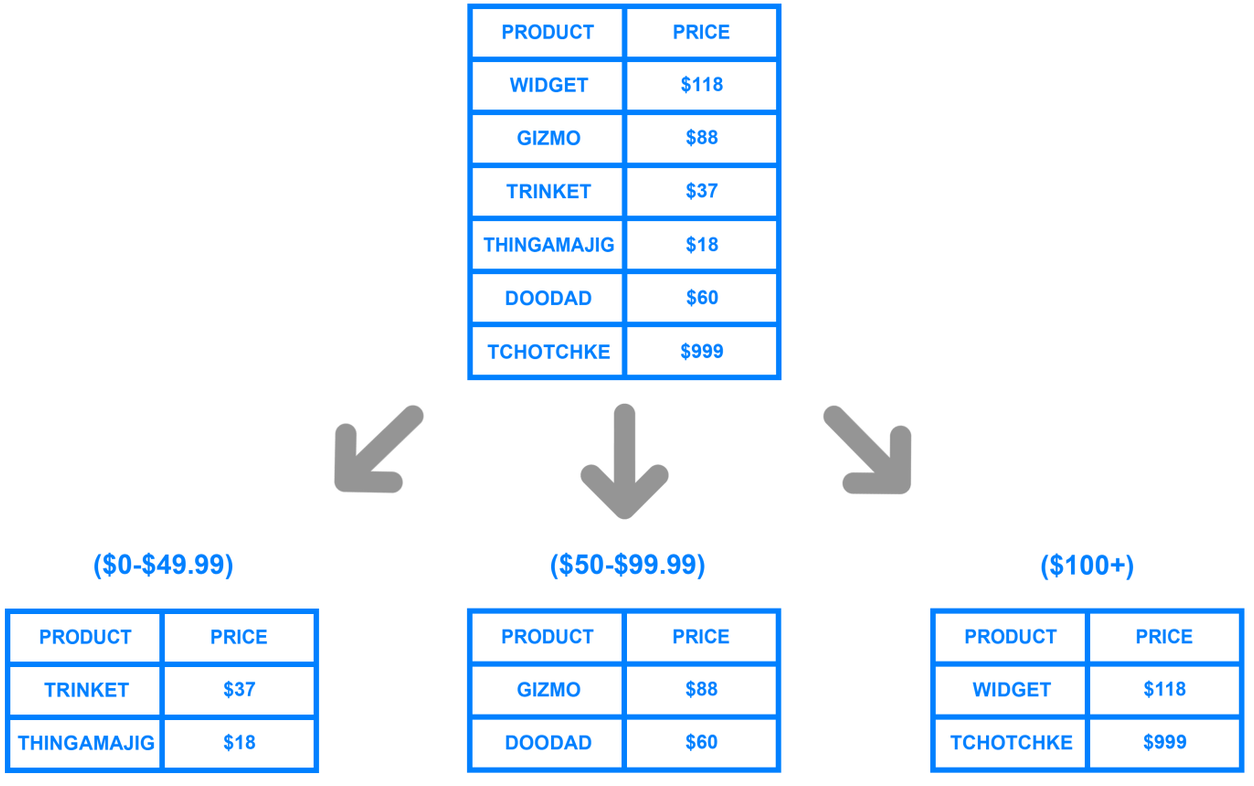
Key Based Sharding - характеристики

- еще называют hash based
- формула примерно такая: F(key) -> shard_id
- F и key очень важны
- наиболее распространенный способ

Key Based Sharding - плюсы/минусы

- просто и понятно (+)
- равномерное распределение (+)
- алгоритмическое распределение (+)
- добавление/удаление шарда всегда боль (-)

Range Based Sharding

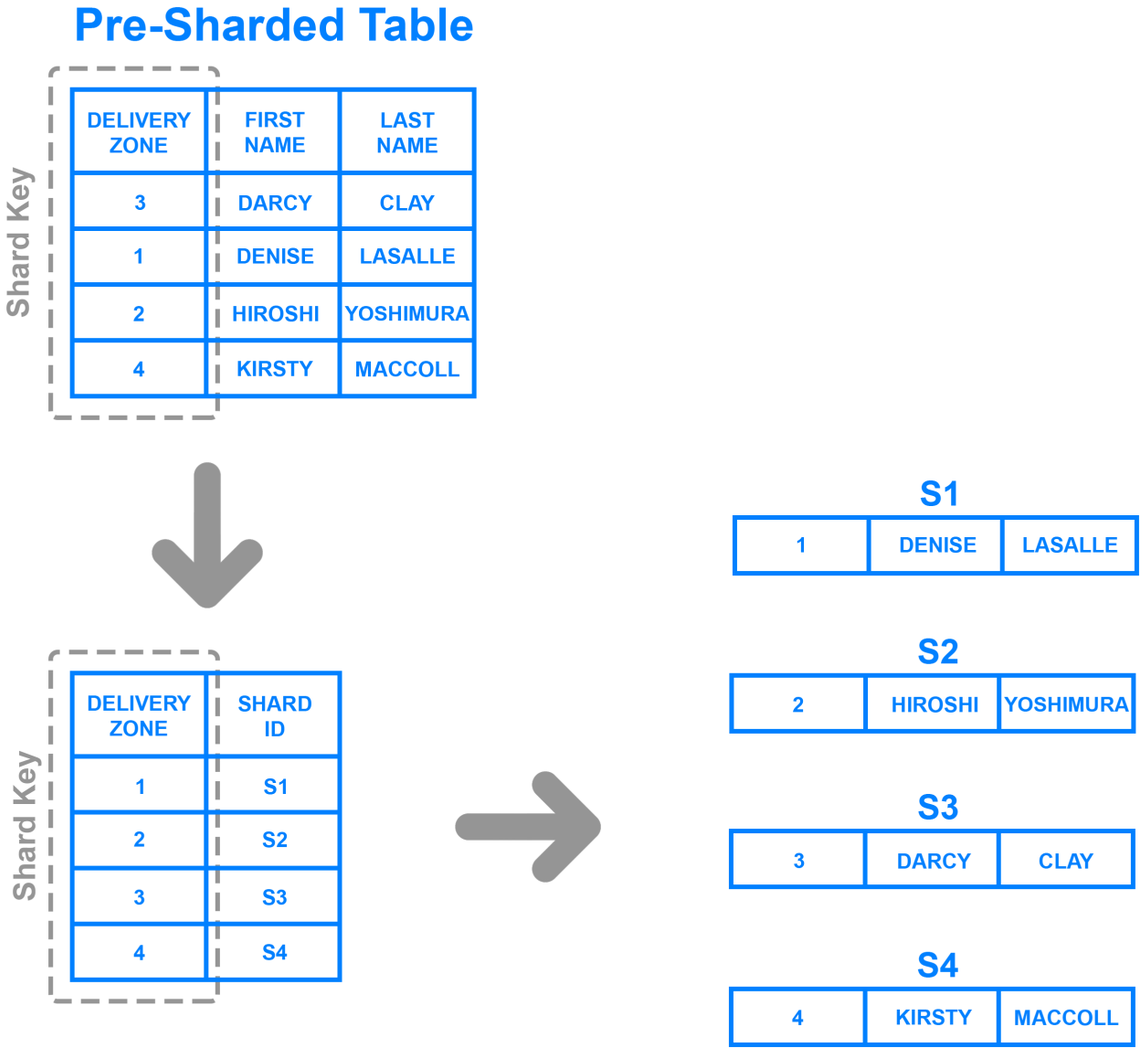


Range Based Sharding - характеристики

- может называться table function/virtual bucket
- статический конфиг range -> shard
- формула примерно такая: hash(key) -> virtual_bucket -> shard_id

Range Based Sharding - плюсы/минусы

- прост в реализации (+)
- потенциально неравномерная нагрузка (-)
- обновление (-)


Pre-Sharded Table
Shard Key
DELIVERY ZONE
FIRST NAME
LAST NAME
3
DARCY
CLAY
1
DENISE
LASALLE
2
HIROSHI
YOSHIMURA
4
KIRSTY
MACCOLL
Shard Key
DELIVERY ZONE
SHARD ID
1
S1
2
S2
3
S3
4
S4
S1
1
DENISE
LASALLE
S2
2
HIROSHI
YOSHIMURA
S3
3
DARCY
CLAY
S4
4
KIRSTY
MACCOLL

Directory Based Sharding - характеристики

- похож на range based
- отображение key -> shard_id
- low cardinality для key - наш случай

Directory Based Sharding - плюсы/минусы

- гибкость (+)
- обновление (-)
- SPOF (-)

Варианты:

- из приложения (умный клиент)
- прокси
- координатор
- есть еще...

Умный клиент

- простой метод
- нет лишних хопов
- как обновлять/решардить?

Прокси

- приложение вообще не знает о шардинге
- код не меняется
- лишний хоп (*)
- могут падать
- им нужно общаться

https://www.proxysql.com https://shardingsphere.apache.org/document/current/en/manual/sharding-proxy

Координатор

- приложение вообще не знает о шардинге
- код не меняется
- лишний хоп
- нагрузка
- SPOF

Вопросы?

Плохо распределили данные

Выход:

- подобрать лучший ключ/функцию шардирования
- решардинг (если имеет смысл)

JOIN

Выход:

- держать нужные данные на одной машине
- делать вычисления на одной машине

Решардинг

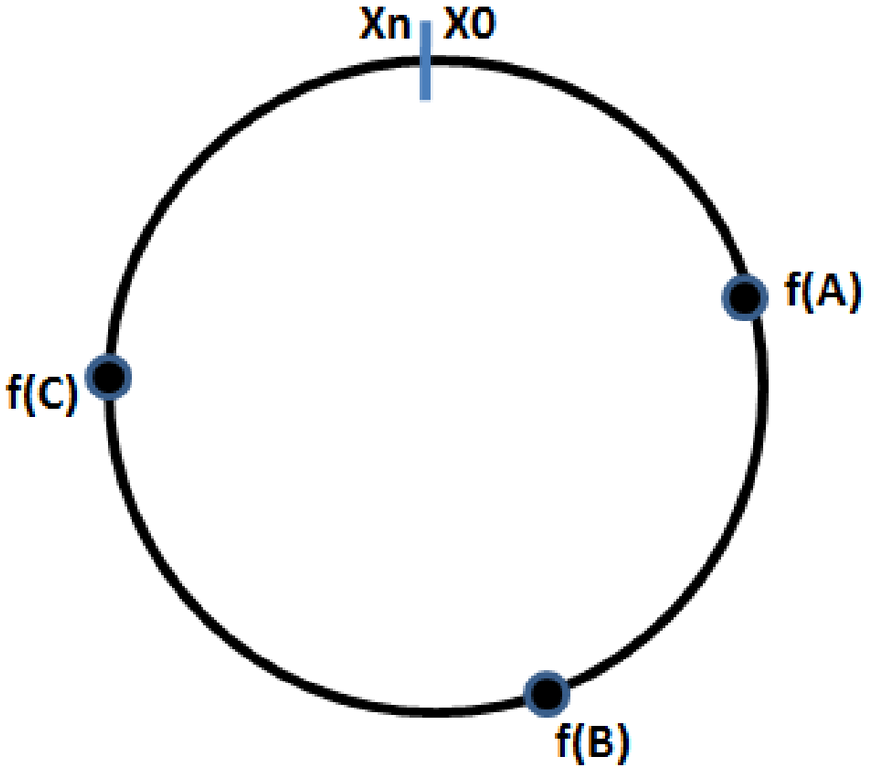
- добавление/удаление нод
- потребовалось "передвинуть данные"
- скорее всего придется решардить когда-то

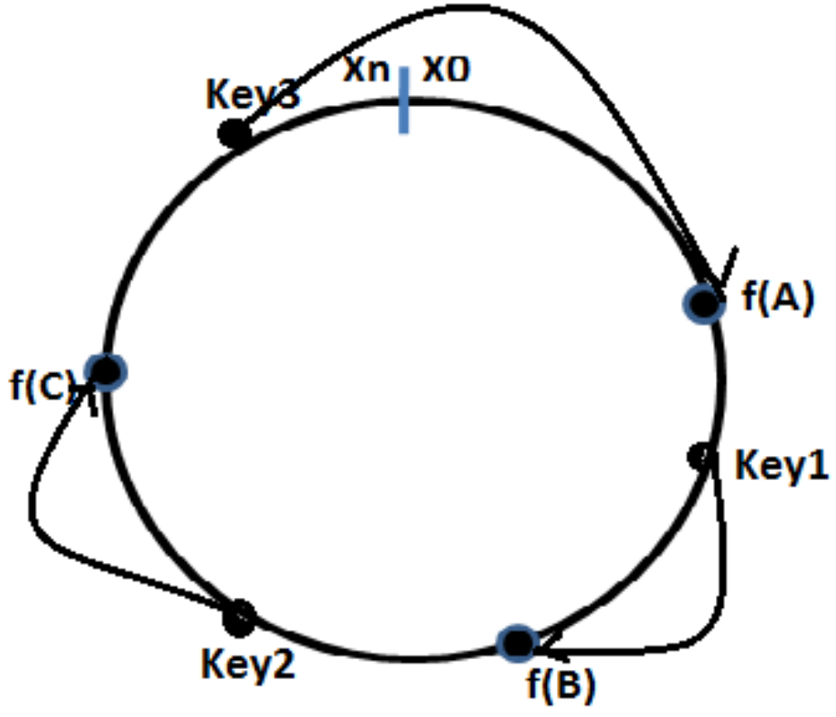
В лоб:

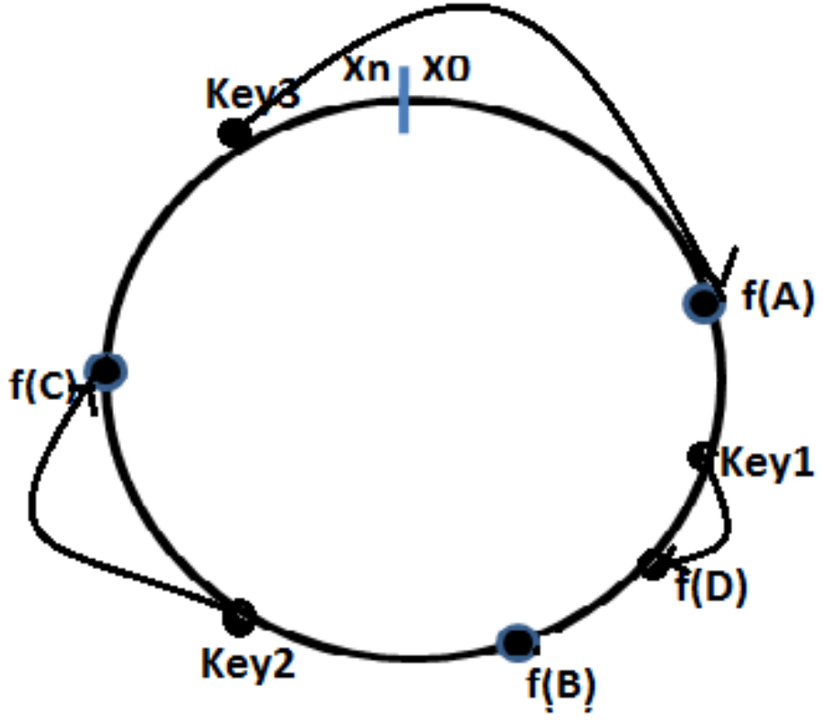
- в худшем случае нужно перемешать все данные
- если степени 2ки - уже неплохо (переместить ~50%)

Пример:

- формула **shard_id % count**
- 16 записей на 8 шардов => 2 записи на шард
- 16 записей на 16 шардов => 1 запись на шард

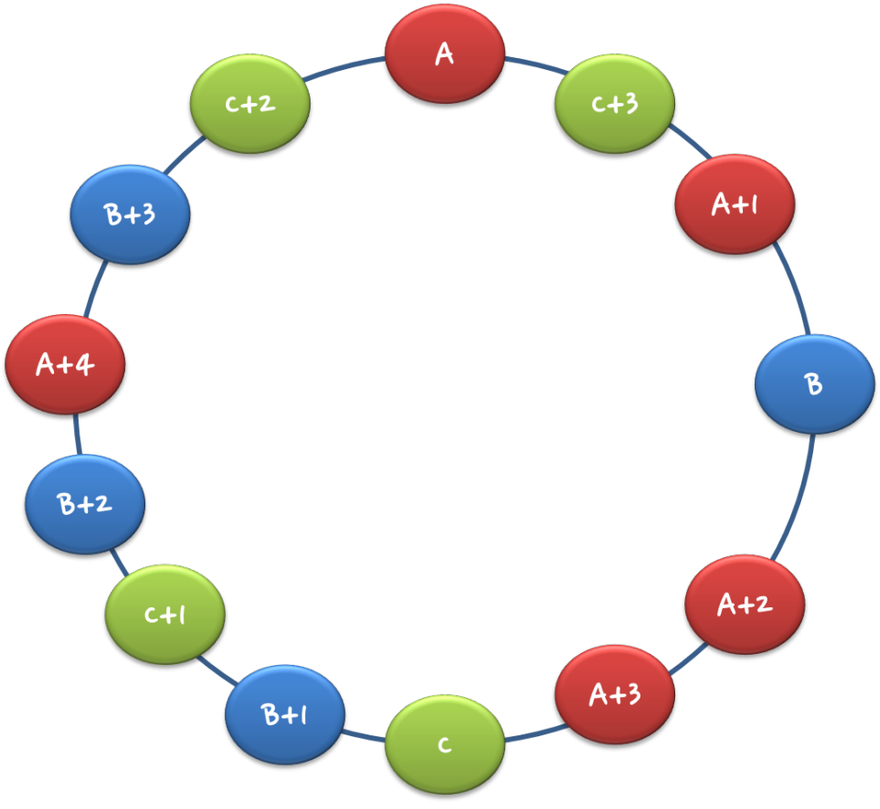

Xn
X0
f(A)
f(C)
f(B)


Key3
Xn
X0
f(A)
f(C)
Key1
Key2
f(B)


Key3
Xn
X0
f(A)
f(C)
Key1
f(D)
Key2
f(B)

- используем одну hash-функцию для нод и данных
- выбирается ближайший по часовой стрелке узел
- при добавлении/удалении затрагивается только часть
- формально не больше K/N (K-ключи, N-сервера)

Данные могут быть распределены неравномерно.


A
C+3
A+1
B
A+2
A+3
C
B+1
C+1
B+2
A+4
B+3
C+2

- помогает разделить данные более равномерно
- количество физических серверов не меняется

Вопросы?

- запросы по ключу шардирования вероятно ускорятся
- запросы не по ключу обойдут все шарды
- запросы по диапазону ключей могут обойти все шарды
- aggregating, joinы - отдельная тема

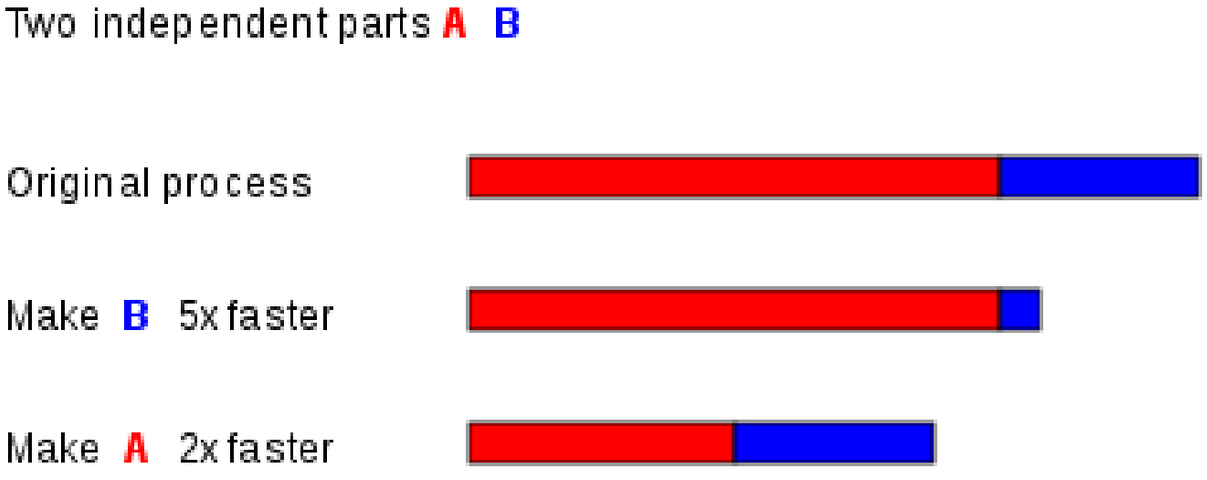
Было:

- Total = Serial + Parallel

Стало:

- Total = Serial + Parallel/N + Xserial

https://en.wikipedia.org/wiki/Amdahl%27s_law


Two independent parts A B
Original process
Make B 5x faster
Make A 2x faster

- MySQL не умеет автошардинг (NDB ?)
- Postgres умеет автошардинг (FDW)
- Cassandra умеет автошардинг
- MongoDB умеет автошардинг

```
CREATE TABLE temperature (
  id BIGSERIAL PRIMARY KEY NOT NULL,
  city_id INT NOT NULL,
  timestamp TIMESTAMP NOT NULL,
  temp DECIMAL(5,2) NOT NULL
);
```

```
CREATE TABLE temperature_201904 (
  id BIGSERIAL NOT NULL,
  city_id INT NOT NULL,
  timestamp TIMESTAMP NOT NULL,
  temp DECIMAL(5,2) NOT NULL
);
```

```
CREATE EXTENSION postgres_fdw;
GRANT USAGE ON FOREIGN DATA WRAPPER postgres_fdw to app_user;
CREATE SERVER shard02 FOREIGN DATA WRAPPER postgres_fdw
    OPTIONS (dbname 'postgres', host 'shard02', port
    '5432');
CREATE USER MAPPING for app_user SERVER shard02
    OPTIONS (user 'fdw_user', password 'secret');
```

```
CREATE FOREIGN TABLE temperature_201904 PARTITION OF temperature
    FOR VALUES FROM ('2019-04-01') TO ('2019-05-01')
    SERVER remoteserver01;
```

- https://www.percona.com/blog/2018/08/21/foreign-data-wrappers-postgresql-postgres_fdw
- https://www.percona.com/blog/2019/05/24/an-overview-of-sharding-in-postgresql-and-how-it-relates-to-mongodbs

- https://www.youtube.com/watch?v=MhGO7BBqSBU
- https://www.youtube.com/watch?v=xx_Lv1P_X_I
- https://www.youtube.com/watch?v=ihrPoHlDFkk
- https://www.mysql.com/products/cluster/scalability.html
- https://www.postgresql.org/docs/9.5/postgres-fdw.html
- https://clickhouse.yandex/docs/en/operations/table_engines/distributed/

Итоги занятия

- Узнали, что такое шардирование
- Поняли, что решает и зачем нужно
- Познакомились с видами и стратегиями шардирования
- Обсудили проблемы и узкие места различных подходов
- Поговорили о решардинге, консистентном хешировании

Вопросы?

Пройдите, пожалуйста, опрос

https://otus.ru/polls/5532/

Спасибо за внимание!