数据分析领域的黑马
--ClickHouse

Power Your Data

新浪-高鹏-2017年11月

“世上无难事，只要有捷径”

“工具选的好，下班回家早”

目录

- 自我介绍
- 数据分析面临的问题
- ClickHouse原理、架构
- ClickHouse在新浪的实践与经验
- ClickHouse案例、生态

关于我

我是谁?

我是干啥的?

关于我

DBA

关于我

DBA

DA

关于我们

Data Analyst
Data Translator

致力于运维大数据
挖掘与分析

可视化、报警、数据分析

AI-OPS

“表哥”“表姐”们

我们需要什么样的工具?

Excel?

也用

Hadoop Spark Hive ?


But,
Hadoop这玩意，
不是一天就
能玩得转的啊~~

Google用Hadoop

多数人用Hadoop

太重了~

一切以需求作为第一位~

一切以需求作为第一位~

快速~ 好用~ 体量够用~

一切以需求作为第一位~

快速~ 好用~ 体量够用~

好维护！！

对结构化的数据

快速给出聚合/过滤结果

We Need

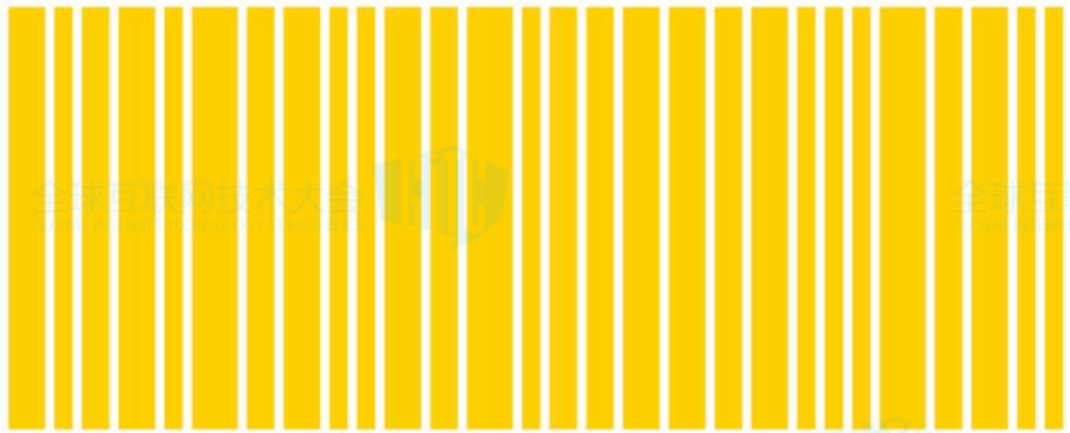
SQL

Fast SQL

Fast Complex SQL

没有什么数据统计是一个SQL解决不了的。

如果有，那就2个



俄罗斯搜索巨头Yandex开源

异步复制 OLAP

SQL 最终一致

统计函数 压缩

PB级别

列式存储

集群 驱动丰富

updated in real time

超高性能 线性扩展

跨数据中心

然鹅，

不支持事务

不支持update/delete

But,

查询‘巨’快

超大容量

Let's Begin

部署：单机

部署

1. 官方提供Ubuntu包
2. 第三方rpm包
2. Docker镜像

需要注意：

1. 修改网络，默认监控IPv4/v6
2. 自定义数据目录，修改官方启动脚本
3. Docker修改时区

```
:) select version();

SELECT version()

┌─version()─┐
│ 1.1.54289 │
└───────────┘

1 rows in set. Elapsed: 0.003 sec.

:) show tables ;

SHOW TABLES

┌─name─┐
│ test │
└──────┘

1 rows in set. Elapsed: 0.002 sec.
```

部署：单机

是不是很SQL

部署：单机

蚝，
我们来压测一下~

部署：单机

数据源

USA civil flights data
since 1987 till 2015

contains 166 millions rows
63 GB of uncompressed data

```
# 下载数据
for s in `seq 1987 2017`
do
for m in `seq 1 12`
do
wget http://transtats.bts.gov\
/PREZIP/On_Time_On_Time_Performance_${s}_${m}.zip
done
done

# 解压
for i in `ls *.zip`; do unzio -o $i;done

# 插入数据
for i in `ls *.csv`
do
echo '__________________'
echo $i
du -sh $i
wc -l $i
time cat $i | sed 's/\.00//g' | sed 1d | clickhouse-client \
-h  127.0.0.1 --port 9000  -d gaopeng4  \
--query="INSERT INTO ontime FORMAT CSVWithNames";
echo '__________________'
sleep 2
done
```

https://clickhouse.yandex/docs/en/getting_started/example_datasets/ontime.html

部署：单机

数据源

USA civil flights data
since 1987 till 2015

contains 166 millions rows
63 GB of uncompressed data

数据大小	173MB
文件行数	436951
插入耗时	4.731 Sec
平均速度	9.3 W/Sec
压缩率	5倍

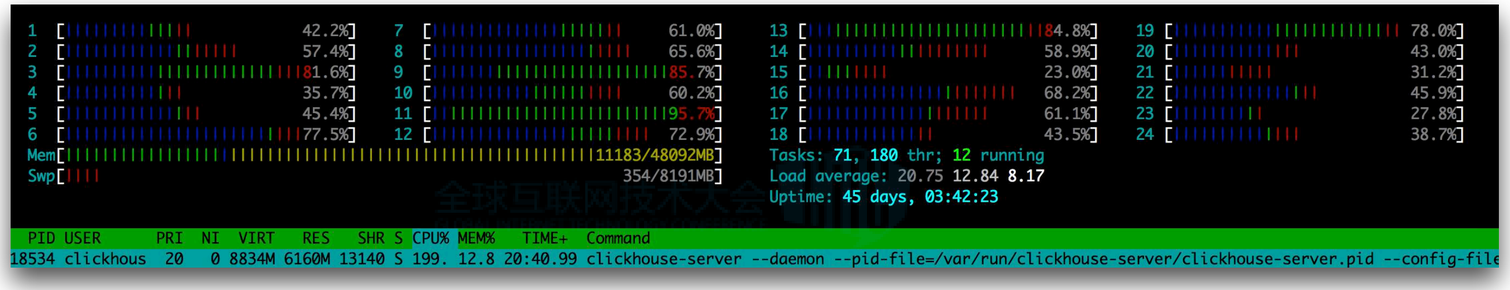
https://clickhouse.yandex/docs/en/getting_started/example_datasets/ontime.html

部署：单机

并发5个进程

```
while read a b c d e ;
do
echo $a $b $c $d $e;

time cat $a | sed 's/\.00//g' | sed 1d | clickhouse-client -h 127.0.0.1 \
--port 9000 -d gaopeng4 --query="INSERT INTO ontime FORMAT CSVWithNames" &

time cat $b | sed 's/\.00//g' | sed 1d | clickhouse-client -h 127.0.0.1 \
--port 9000 -d gaopeng4 --query="INSERT INTO ontime FORMAT CSVWithNames" &

time cat $c | sed 's/\.00//g' | sed 1d | clickhouse-client -h 127.0.0.1 \
--port 9000 -d gaopeng4 --query="INSERT INTO ontime FORMAT CSVWithNames" &

time cat $d | sed 's/\.00//g' | sed 1d | clickhouse-client -h 127.0.0.1 \
--port 9000 -d gaopeng4 --query="INSERT INTO ontime FORMAT CSVWithNames" &

time cat $e | sed 's/\.00//g' | sed 1d | clickhouse-client -h 127.0.0.1 \
--port 9000 -d gaopeng4 --query="INSERT INTO ontime FORMAT CSVWithNames" &

sleep 5
done <<EOF
`ls *.csv | xargs -n 5`
EOF
```

机器负载

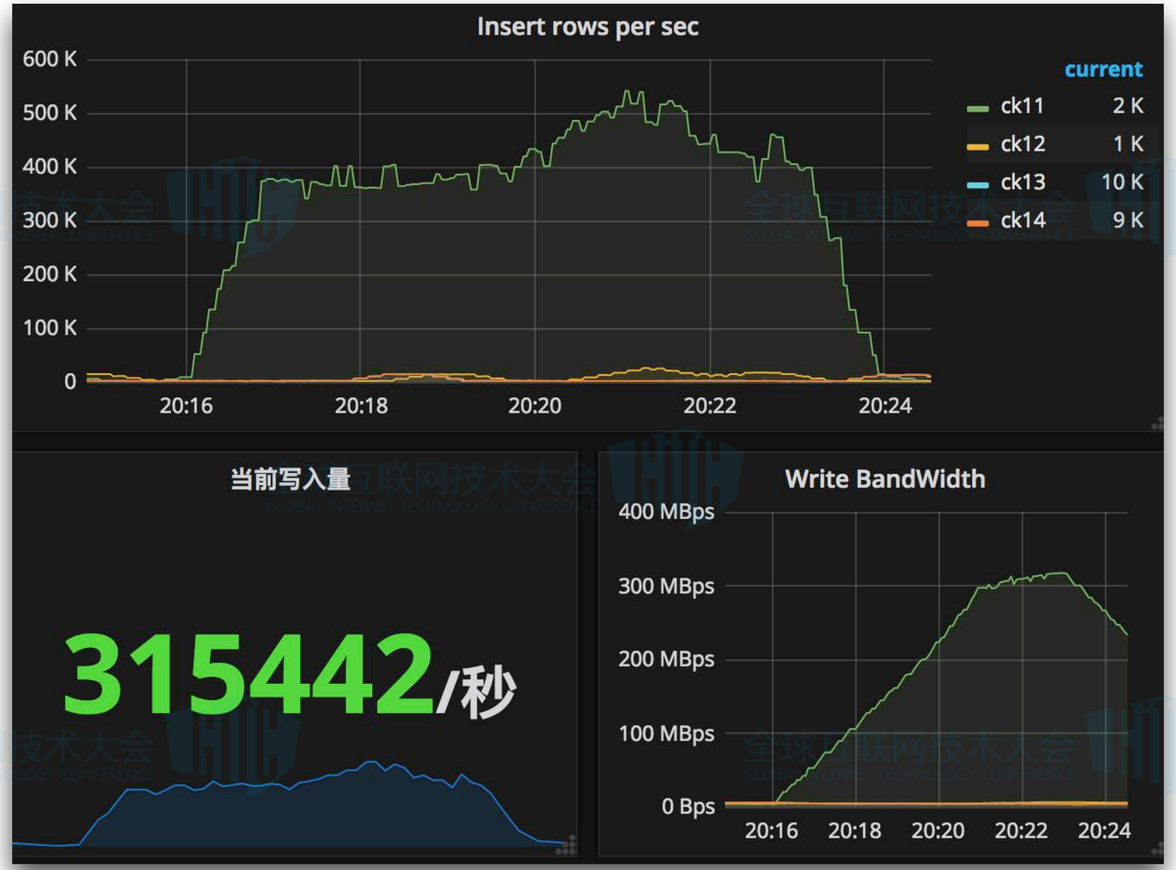


部署：单机

响应时间

```
SELECT elapsed
FROM processes
ORDER BY elapsed DESC
LIMIT 10

┌─────elapsed─┐
| 5.291229166 |
| 5.287489023 |
| 5.285718683 |
| 5.284233673 |
| 5.282636967 |
| 0.000210788 |
└─────────────┘
```

峰值50W QPS



部署：单机

查询类型

1. 查询总量

2. 简单group by

```
:) select count(*)/100000000 from ontime ;

SELECT count(*) / 100000000
FROM ontime

┌─divide(count(), 100000000)─┐
│                    1.71412868 │
└──────────────────────────────┘

1 rows in set. Elapsed: 0.051 sec. Processed 171.41 million rows, 171.41 MB (3.35 billion rows/s., 3.35 GB/s.)

:) select Year, count(*) as c1 from ontime group by Year limit 3
:-] ;

SELECT
    Year,
    count(*) AS c1
FROM ontime
GROUP BY Year
LIMIT 3

┌─Year─┬──────c1─┐
│ 1988 │ 5202084 │
│ 1989 │ 5041188 │
│ 1990 │ 5270881 │
└──────┴─────────┘

3 rows in set. Elapsed: 0.208 sec. Processed 171.41 million rows, 342.83 MB (825.43 million rows/s., 1.65 GB/s.)
```

https://clickhouse.yandex/docs/en/getting_started/example_datasets/ontime.html

部署：单机

查询类型

- 条件查询，聚合，排序

```
SELECT
    DestCityName,
    uniqExact(OriginCityName) AS u
FROM ontime
WHERE (Year >= 2000) AND (Year <= 2010)
GROUP BY DestCityName
ORDER BY u DESC
LIMIT 10

┌─DestCityName──────────┬───u─┐
| Atlanta, GA           | 193 |
| Chicago, IL           | 167 |
| Dallas/Fort Worth, TX | 161 |
| Minneapolis, MN       | 138 |
| Cincinnati, OH        | 138 |
| Detroit, MI           | 130 |
| Houston, TX           | 129 |
| Denver, CO            | 127 |
| Salt Lake City, UT    | 119 |
| New York, NY          | 115 |
└───────────────────────┴─────┘

10 rows in set. Elapsed: 1.185 sec. Processed 72.79 million rows, 3.37 GB (61.45 million rows/s., 2.85 GB/s.)
```

https://clickhouse.yandex/docs/en/getting_started/example_datasets/ontime.html

部署：单机

查询类型

- 复杂查询

```
SELECT
    min(Year),
    max(Year),
    Carrier,
    count(*) AS cnt,
    sum(ArrDelayMinutes > 30) AS flights_delayed,
    round(sum(ArrDelayMinutes > 30) / count(*), 2) AS rate
FROM ontime
WHERE (DayOfWeek NOT IN (6, 7)) AND (OriginState NOT IN ('AK', 'HI', 'PR', 'VI')) AND (DestState NOT IN ('AK', 'HI', 'PR', 'VI')) AND (FlightDate < '2010-01-01')
GROUP BY Carrier
HAVING (cnt > 100000) AND (max(Year) > 1990)
ORDER BY rate DESC
LIMIT 1000
```

min(Year)	max(Year)	Carrier	cnt	flights_delayed	rate
2003	2009	EV	1454777	237698	0.16
2003	2009	B6	683874	103677	0.15
2006	2009	YV	740606	110389	0.15
2003	2009	FL	1082489	158748	0.15
2006	2009	XE	1016010	152431	0.15
2003	2005	DH	501056	69833	0.14
2001	2009	MQ	3238137	448037	0.14
2004	2009	OH	1195868	160071	0.13
2003	2006	RU	1007247	126733	0.13
1988	2009	UA	9593281	1197052	0.12
2003	2006	TZ	136735	16496	0.12
1988	2009	AA	10600421	1185336	0.11
1988	2009	CO	6029147	673863	0.11
1988	2001	TW	2659963	280741	0.11
1988	2009	DL	11869418	1156256	0.1
2003	2009	OO	2654259	257069	0.1
2007	2009	9E	577223	59437	0.1
1988	2009	NW	7601726	725460	0.1
1988	2009	US	10276931	991016	0.1
1988	2009	AS	1506003	146920	0.1
1988	2009	WN	12722172	1107840	0.09
1988	1991	PA	206841	19465	0.09
1988	2005	HP	2607603	235675	0.09
2005	2008	F9	307569	28679	0.09

24 rows in set. Elapsed: 1.094 sec. Processed 128.68 million rows, 1.57 GB (117.57 million rows/s., 1.44 GB/s.)

https://clickhouse.yandex/docs/en/getting_started/example_datasets/ontime.html

部署：单机

使用

1. 启动Server
2. use db, create table
3. 尽情select
4. 推荐引擎：MergeTree

```
CREATE TABLE apm.apm_msg (_clientip String,  _data_size Float32,
    date Date,  ts DateTime, hour Int8, minute Int8)
ENGINE = MergeTree(date, (minute, hour, date), 8192);
```

分区　　主键　　稀疏索引粒度

总结

优点：
1. 部署简单
2. 全部CPU打满，查询效率极高

问题:
1. 性能依赖单机（scale up路线）
2. 存在单点故障风险（宕机数据全丢）

MergeTree

写

- 如何写的快?
- 是否可压缩?

类似LSM Tree，但是没有内存表，不记录log

直接落磁盘，按照主键排序，分块写入

异步merge，与写不冲突，最大merge到月纬度

不支持删除、修改

primary.idx+*.bin+*.mrk+checksums.txt+columns.txt

读

- 如何快速查找?
- 数据量大，如何适应内存?

- 主键查询：

eg: (x, y, z, date)

最左原则

- 其他列查询：

稀疏索引定位区间：不适合点对点查询，适合范围查询

颗粒度N：默认8192

查询问题：会带来过多的IO

缺乏：

扩展性

可靠性

如何获得：

扩展性

可靠性

部署：‘分布式’

概括

假的‘scale out’

借助于特殊引擎实现

借助配置文件

部署：‘分布式’

```
CREATE TABLE apm.apm_msg (_clientip String,  _data_size Float32,
    date Date,  ts DateTime, hour Int8, minute Int8)
ENGINE = MergeTree(date, (minute, hour, date), 8192);
```

分区　　主键　　颗粒度

```
CREATE TABLE apm.apm_msg_all
(_clientip String,  _data_size Float32,
    date Date,  ts DateTime, hour Int8, minute Int8)
ENGINE = Distributed(bip_ck_cluster, apm, apm_msg, rand());
```

集群名称　　库　　表　　分布算法

Distributed引擎：

1. 本身不存储数据

2. 被写入，做转发

3. 查询，作为中间件，聚合后返回给用户

部署：‘分布式’

- 分布式如何做到的

```
<clickhouse_remote_servers>
    <bip_ck_cluster>
        <shard>
            <internal_replication>true</internal_replication>
            <replica>
                <host>ck11.[redacted].com.cn</host>
                <port>9000</port>
            </replica>
        </shard>
        <shard>
            <replica>
                <internal_replication>true</internal_replication>
                <host>ck12.[redacted].com.cn</host>
                <port>9000</port>
            </replica>
        </shard>
        <shard>
            <internal_replication>true</internal_replication>
            <replica>
                <host>ck13.[redacted].com.cn</host>
                <port>9000</port>
            </replica>
        </shard>
        <shard>
            <internal_replication>true</internal_replication>
            <replica>
                <host>ck14.[redacted].com.cn</host>
                <port>9000</port>
            </replica>
        </shard>
    </bip_ck_cluster>
</clickhouse_remote_servers>
```

分片1

分片2

分片3

分片4