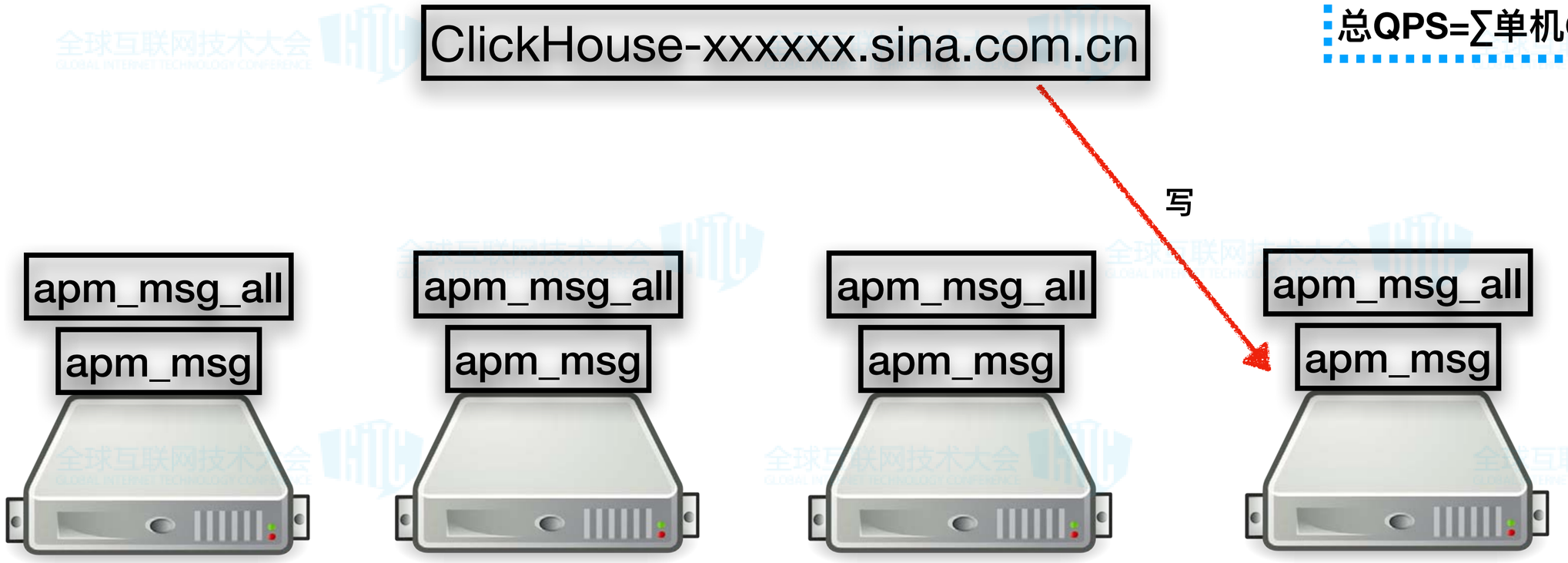
部署：‘分布式’

写操作

通过域名，写本地表

insert into apm_msg values()

总QPS=∑单机QPS

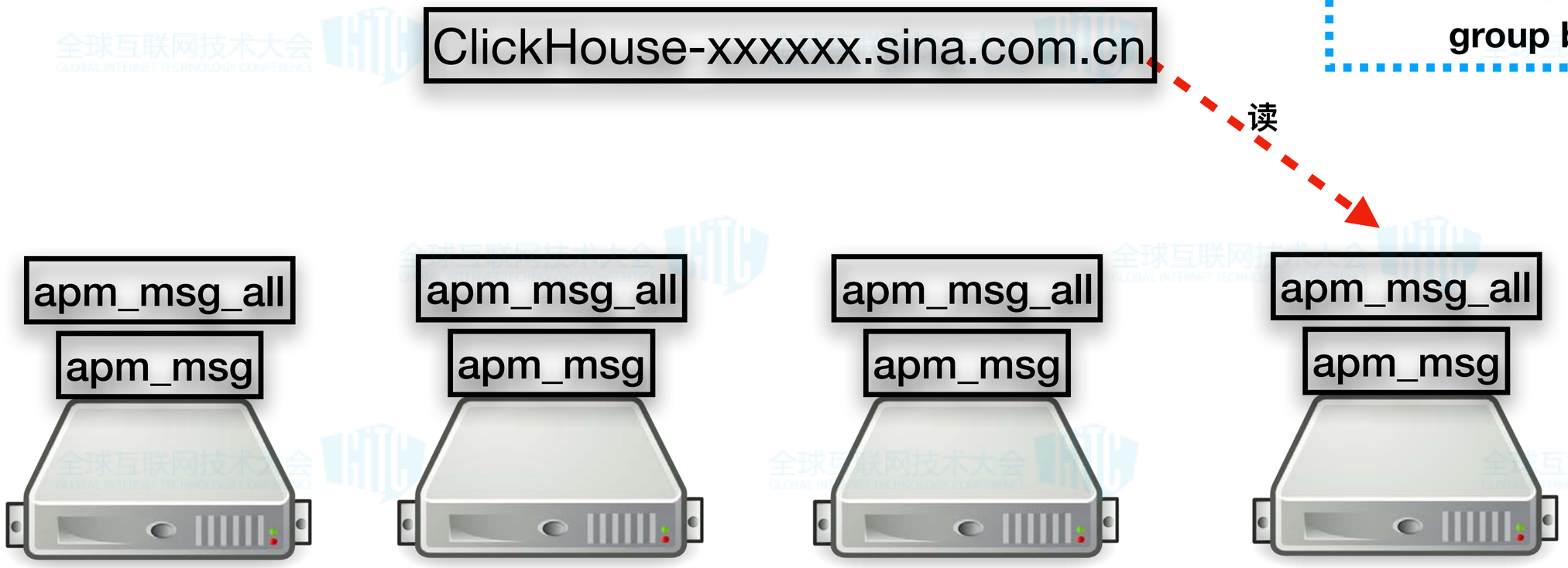


部署：‘分布式’

读操作

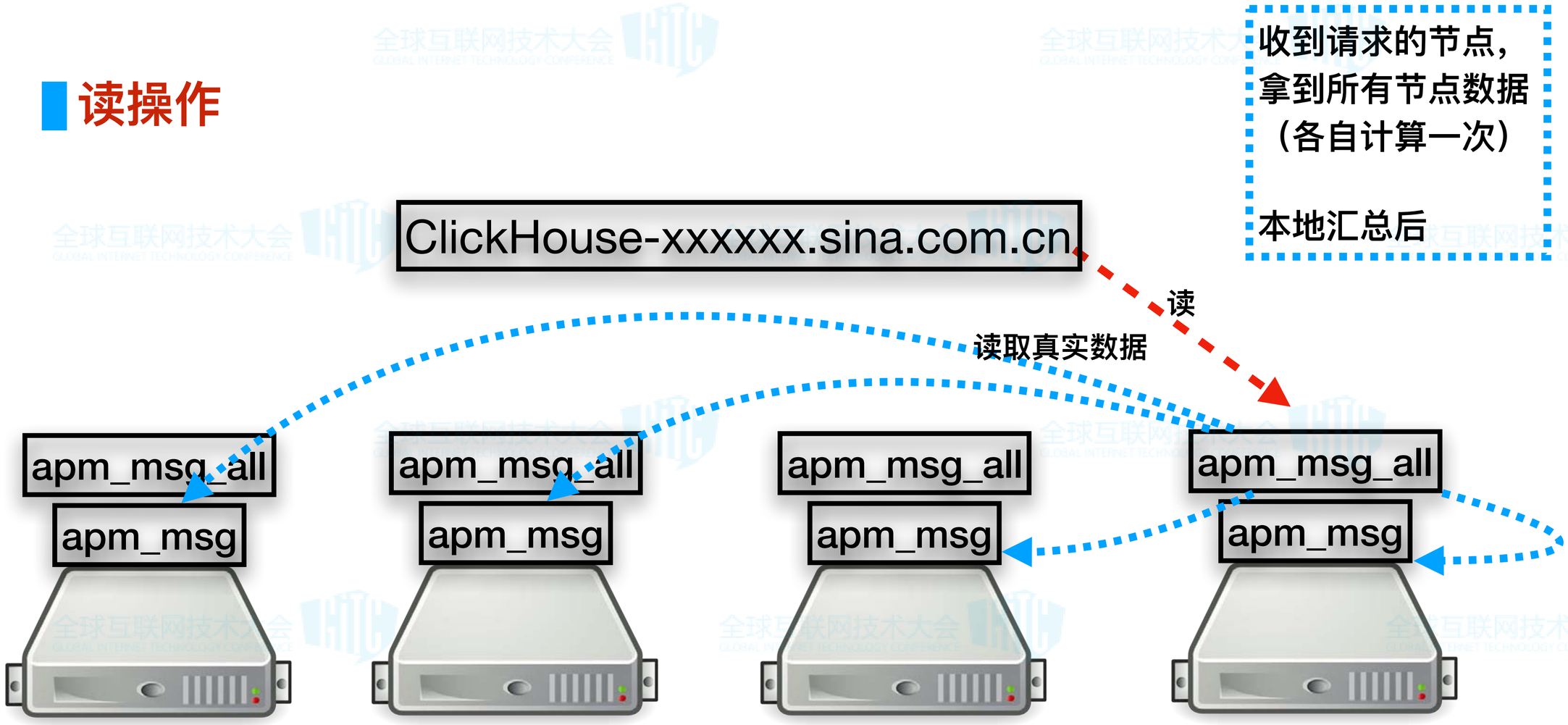
通过域名，读分布式表
select * from apm_msg_all
where xxx=yyy

原则：确保其他节点返回的数据，自己还可以聚合，如top/group by逻辑就不同



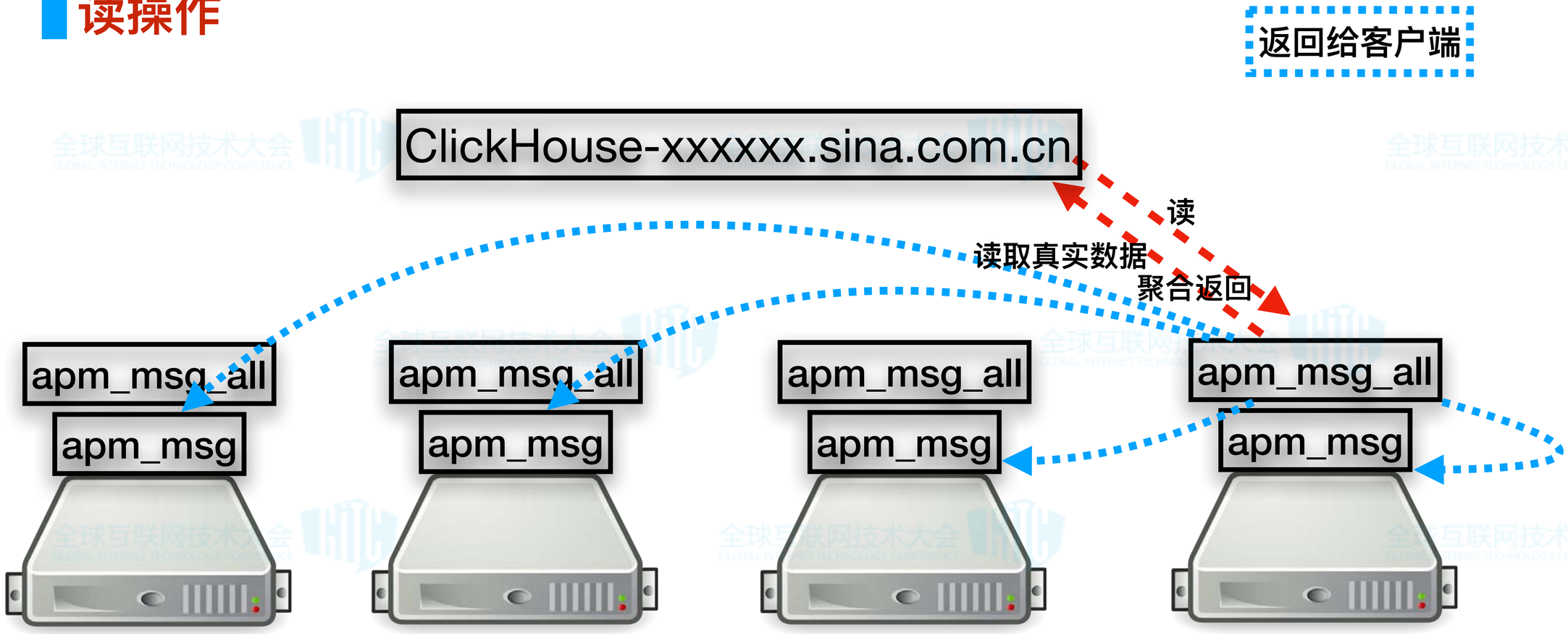
部署：‘分布式’

读操作


收到请求的节点，
拿到所有节点数据
（各自计算一次）
本地汇总后
ClickHouse-xxxxxx.sina.com.cn
读
读取真实数据
apm_msg_all
apm_msg
apm_msg_all
apm_msg
apm_msg_all
apm_msg
apm_msg_all
apm_msg

部署：‘分布式’

读操作


返回给客户端
ClickHouse-xxxxxx.sina.com.cn
读
读取真实数据
聚合返回
apm_msg_all
apm_msg
apm_msg_all
apm_msg
apm_msg_all
apm_msg
apm_msg_all
apm_msg

部署：‘分布式’

总结

通过全局配置文件，达到集群相互知晓

各自维护各自的数据，让用户自己写入

水平扩展性很好

查询/写入能力随机器数线性增加

cluster config updated on the fly

部署：‘分布式’

问题

1. 直接写分布式表，造成数据不均匀

2. 域名映射的IP只有在初始解析

3. 新增节点，历史数据不会搬迁，造成不均衡

4. 过度的group by，导致大量数据交换

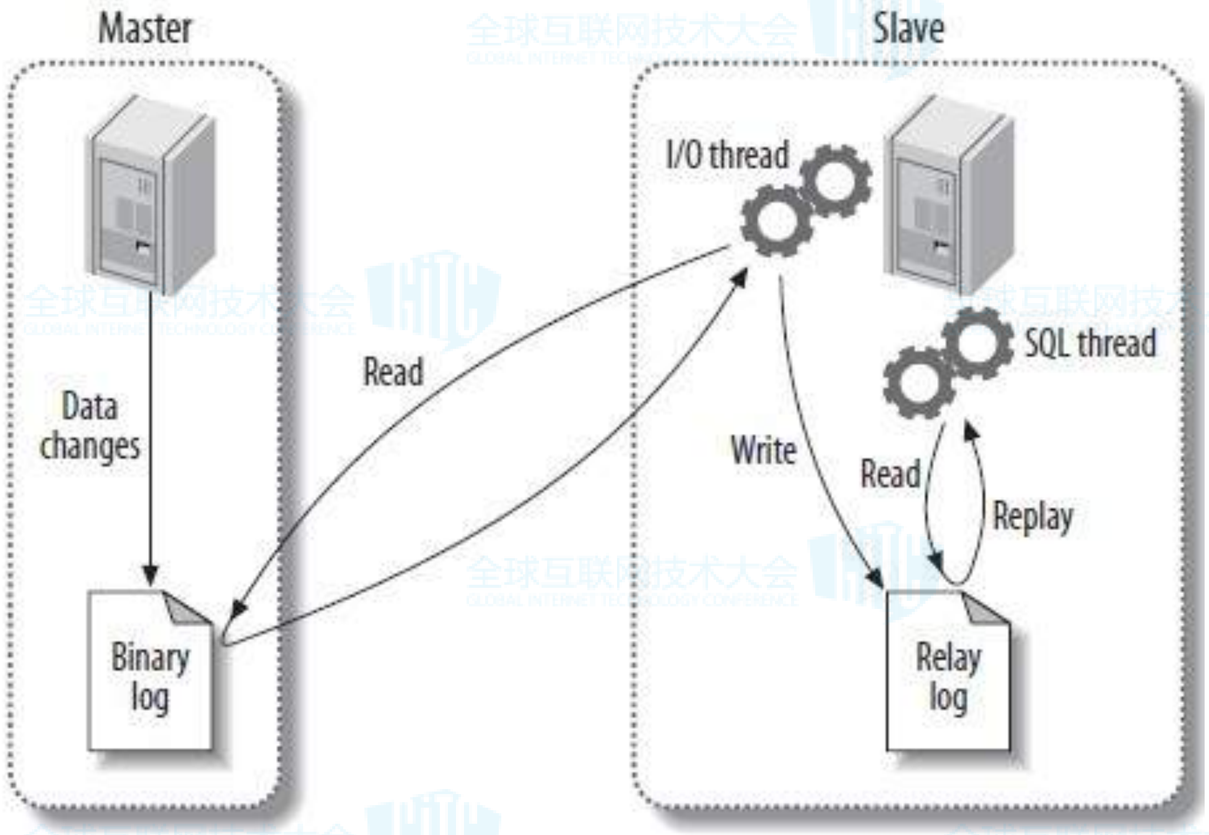
5. 数据分片依赖单机稳定性，缺乏可靠性

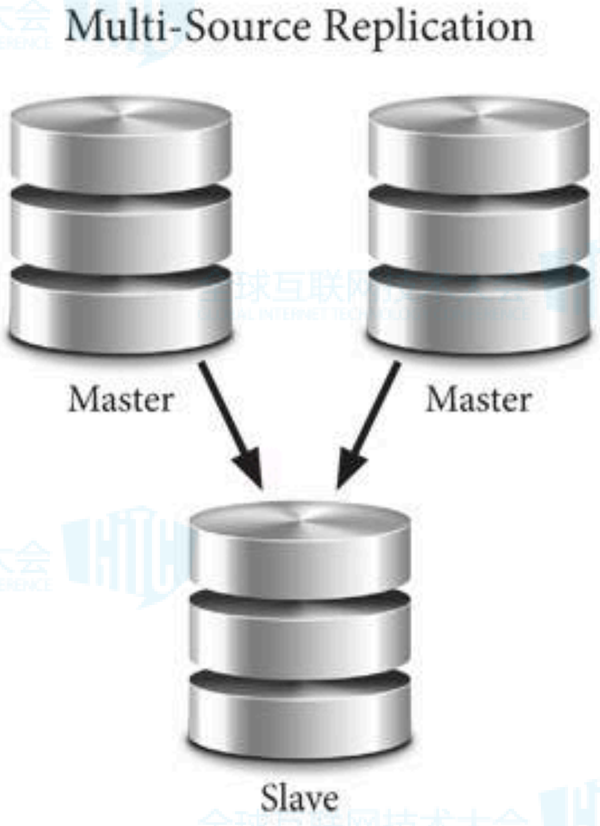
如何获得：

扩展性

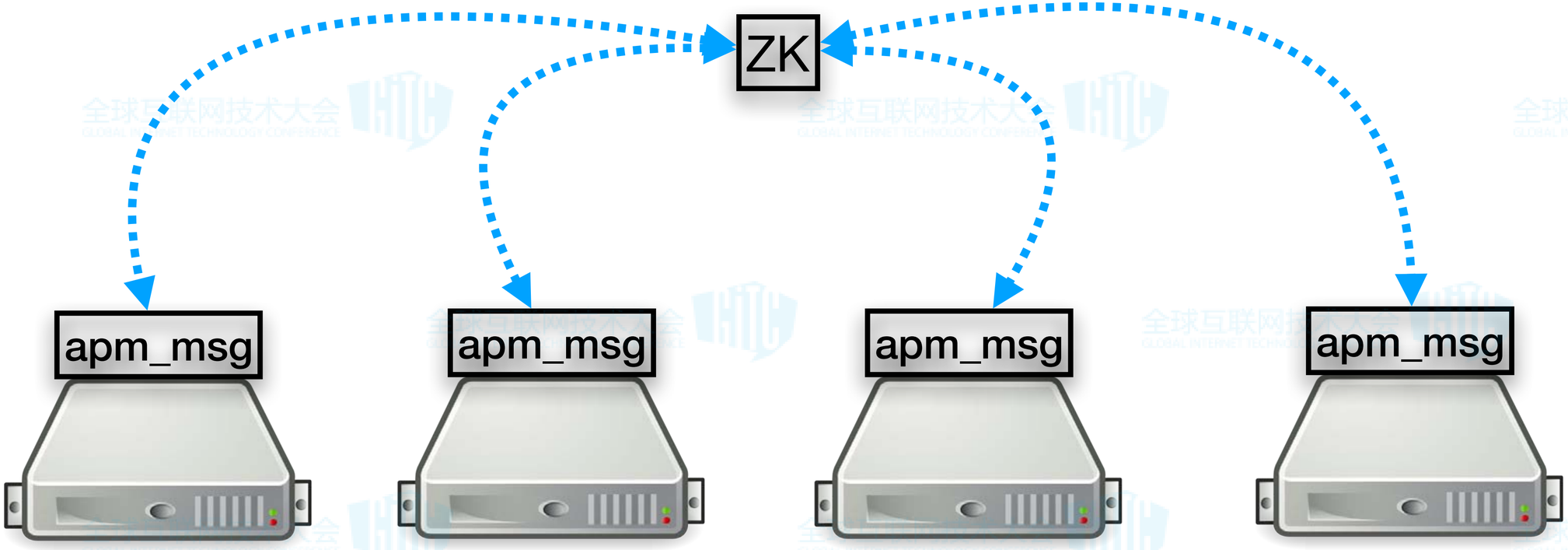
可靠性

我们知道的复制：


Master
Slave
I/O thread
SQL thread
Read
Data changes
Write
Read
Replay
Binary log
Relay log


Multi-Source Replication
Master
Master
Slave

ClickHouse的复制



多源、多主、多向复制

数据‘互通有无’

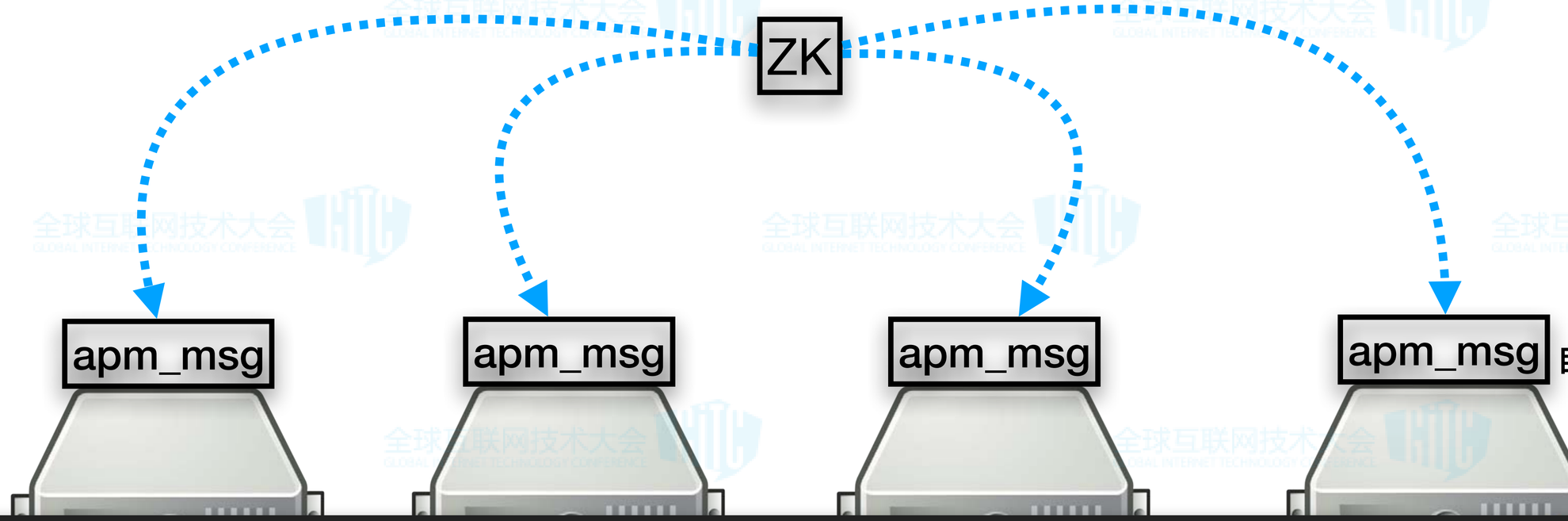
自带检测机制

自带同步机制（物理复制）

依赖ZK

非多数派写

部署： 复制



多源、多主、多向复制

数据‘互通有无’

自带检测机制

自带同步机制（物理复制）

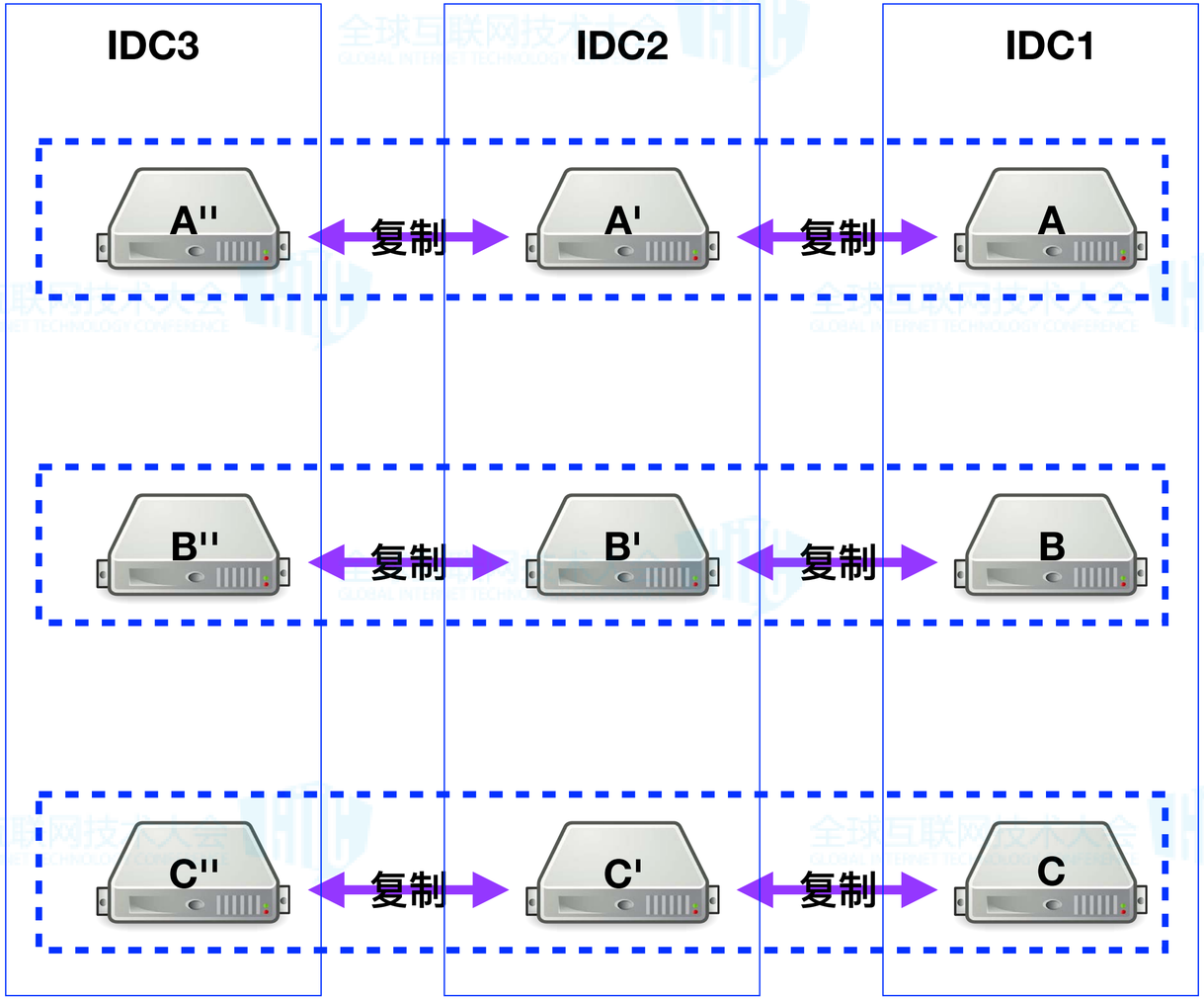
依赖ZK

非多数派写

```
CREATE TABLE apm.apm_msg (_clientip String,  _data_size Float32,
    date Date,  ts DateTime, hour Int8, minute Int8)
ENGINE = ReplicatedMergeTree('/clickhouse/tables/apm_msg', '195', date, (minute, hour, date), 8192);
```

ReplicatedMergeTree('zk路径', '副本名称', 日期列, (其他列, 日期列), 索引粒度)

部署：最佳架构



3个IDC使用复制机制做互备

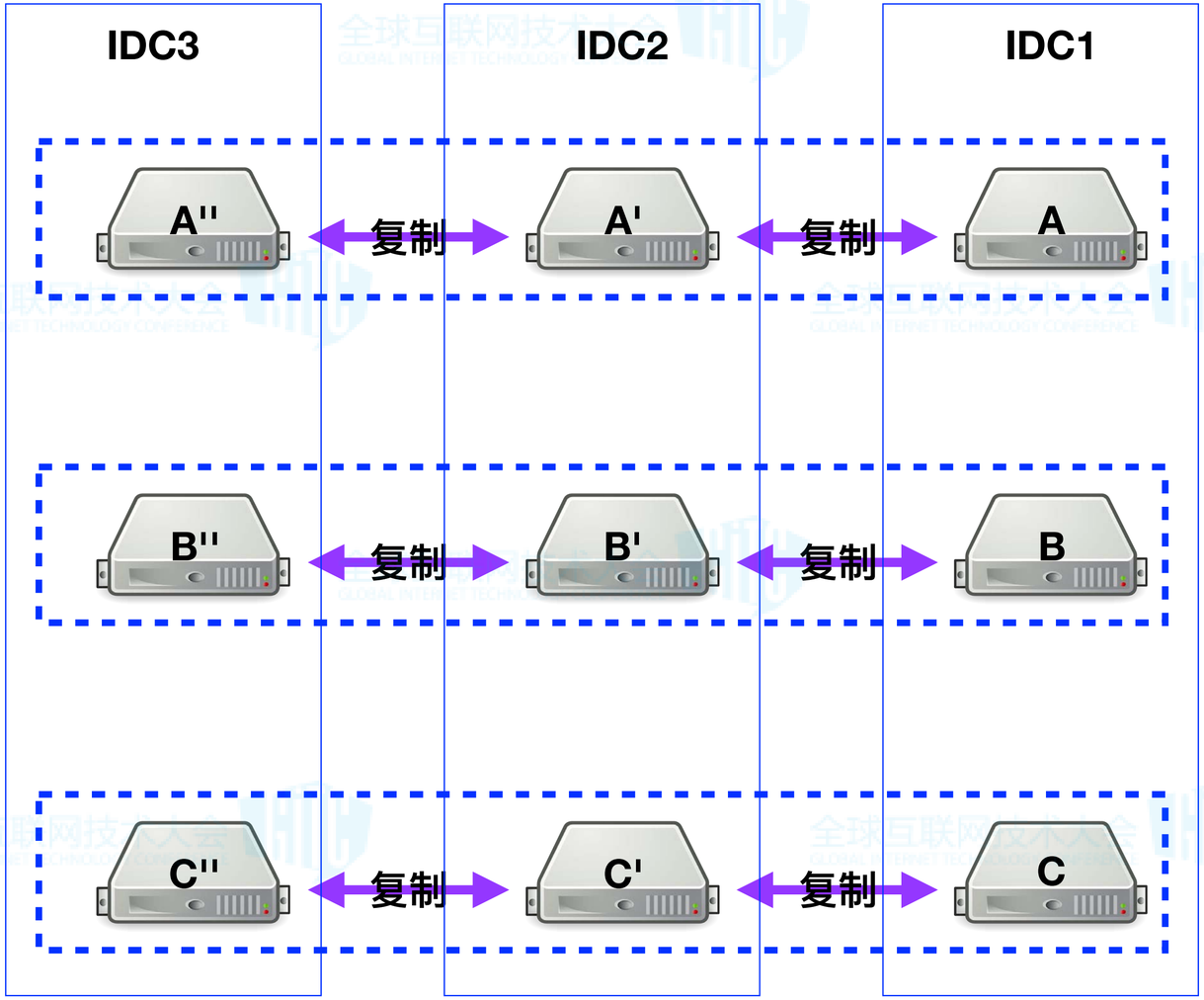
每个IDC 3个节点，做分布式表，分担查询压力

如：

A/B/C 分别是数据的3个分片，各自承担1/3的数据

查询性能：A+B+C

部署：最佳架构



宕机分析

1. 3个IDC，挂掉2个
不影响读写
恢复只需重启实例
ClickHouse自动完成数据同步

2. 挂掉某个实例，如A
切换读写到其他IDC，恢复实例A即可

3. 挂掉多个实例，A、B同时挂掉
处理方式同上

集群的ClickHouse有多快?

我们的架构

CPU E5-2620 @ 2.00GHz
开启超线程后24core

48G内存

3T*12 Raid5

X 4

表1：
14字段 900亿 4T

表2：
94字段 500亿 15T

请开始你的表演：

```
:) select count(*) from apm_msg_all;

SELECT count(*)
FROM apm_msg_all

┌──────count()─┐
│ 30249469866 │
└──────────────┘

1 rows in set. Elapsed: 0.927 sec. Processed 30.25 billion rows, 30.25 GB (32.63 billion rows/s., 32.63 GB/s.)

:)
```

select count(*)

300亿

0.9秒

请开始你的表演：

```
:) select date, count(*)/100000000 from apm_msg_all group by date order by date desc limit 10 ;

SELECT
    date,
    count(*) / 100000000
FROM apm_msg_all
GROUP BY date
ORDER BY date DESC
LIMIT 10

┌──────date─┬─divide(count(), 100000000)─┐
│ 2017-10-09 │                 2.95357962 │
│ 2017-10-08 │                 5.25704122 │
│ 2017-10-07 │                 5.02549947 │
│ 2017-10-06 │                 4.92416848 │
│ 2017-10-05 │                 4.98019803 │
│ 2017-10-04 │                 4.68522239 │
│ 2017-10-03 │                 4.81681714 │
│ 2017-10-02 │                 4.95085341 │
│ 2017-10-01 │                 4.78589495 │
│ 2017-09-30 │                 4.90037846 │
└────────────┴────────────────────────────┘

10 rows in set. Elapsed: 9.872 sec. Processed 30.27 billion rows, 60.53 GB (3.07 billion rows/s., 6.13 GB/s.)
:)
```

select date, count(*) from xx group by date

300亿group by日期

9.8秒

请开始你的表演：

```
        _request_url,
        count(1) AS hit_2hour
    FROM apm.apm_msg_all
    WHERE (date = toString(today())) AND (hour = (toInt8(substring(toString(now()), 12, 2)) - 2)) AND (_ek = '_error') AND (url_ty
e = 'PIC') AND (_http_code IN ('403', '404'))
    GROUP BY _request_url
  )
  ALL INNER JOIN
  (
    SELECT
        _request_url,
        round(MAX(hit) / 60, 2) AS maxhit_per_s,
        sum(hit) AS hit_hour
    FROM
    (
        SELECT
            _request_url,
            minute,
            COUNT(1) AS hit
        FROM apm.apm_msg_all
        WHERE (date = toString(today())) AND (hour = (toInt8(substring(toString(now()), 12, 2)) - 1)) AND (_ek = '_error') AND (ur
l_type = 'PIC') AND (_http_code IN ('403', '404'))
        GROUP BY
            _request_url,
            minute
    ) AS a
    GROUP BY _request_url
  ) USING (_request_url)
  GROUP BY
    _request_url,
    maxhit_per_s,
    hit_2hour,
    hit_hour
USING (_request_url)

Progress: 3.42 million rows, 213.70 MB (28.50 million rows/s., 1.78 GB/s.)  21%
Progress: 79.78 million rows, 6.18 GB (97.48 million rows/s., 7.55 GB/s.)  92%
```

```
10 rows in set. Elapsed: 1.140 sec. Processed 114.07 million rows, 9.46 GB (100.06 million rows/s., 8.29 GB/s.)
```

太复杂了

请开始你的表演：

```
SELECT
    hour,
    one_hit,
    two_hit,
    one_error_hit,
    two_error_hit,
    round((one_error_hit / ((one_hit / 0.3) + one_error_hit)) * 100, 2) AS one_error_rate,
    round((two_error_hit / ((two_hit / 0.3) + two_error_hit)) * 100, 2) AS two_error_rate
FROM
(
    SELECT
        hour,
        sum(caseWithoutExpr((area IN ('北京', '上海')) OR (city_name IN ('深圳', '广州')), 1, 0)) AS one_hit,
        sum(caseWithoutExpr((area NOT IN ('北京', '上海')) AND (city_name NOT IN ('深圳', '广州')), 1, 0)) AS two_hit
    FROM apm.apm_msg_all
    WHERE (date = toString(yesterday())) AND (_ek = '_network') AND (_snet != 'wifi') AND (country = '中国')
    GROUP BY hour
)
ANY LEFT JOIN
(
    SELECT
        hour,
        sum(caseWithoutExpr((area IN ('北京', '上海')) OR (city_name IN ('深圳', '广州')), 1, 0)) AS one_error_hit,
        sum(caseWithoutExpr((area NOT IN ('北京', '上海')) AND (city_name NOT IN ('深圳', '广州')), 1, 0)) AS two_error_hit
    FROM apm.apm_msg_all
    WHERE (date = toString(yesterday())) AND (_ek = '_error') AND (_snet != 'wifi') AND (country = '中国')
    GROUP BY hour
) USING (hour)

Progress: 558.25 million rows, 42.98 GB (102.60 million rows/s., 7.90 GB/s.)
```

```
24 rows in set. Elapsed: 14.543 sec. Processed 1.23 billion rows, 94.76 GB (84.60 million rows/s., 6.52 GB/s.)
```

更复杂了

支持Join

请开始你的表演：

官方demo

```
:) SELECT
:-]     count() as visits,
:-]     sum(PageViews) as hits,
:-]     uniq(UserID) as users
:-] FROM visits_all
:-] WHERE StartDate > today() - 7;

SELECT
    count() AS visits,
    sum(PageViews) AS hits,
    uniq(UserID) AS users
FROM visits_all
WHERE StartDate > (today() - 7)

┌──────visits─┬────────hits─┬──────users─┐
│ 12106577368 │ 46414744272 │ 1080418023 │
└─────────────┴─────────────┴────────────┘

1 rows in set. Elapsed: 6.245 sec. Processed 13.23 billion rows, 185.18 GB (2.12 billion rows/s., 29.65 GB/s.)
```

```
SELECT
    uniq(UserID) AS users,
    if(IsMobile, if(IsTablet, 'tablet', 'phone'), 'desktop') AS device
FROM visits_all
WHERE StartDate > (today() - 7)
GROUP BY device

┌──────users─┬─device─┐
│ 465363568 │ phone  │
└───────────┴────────┘
┌──────users─┬─device─┐
│ 548496870 │ desktop │
└───────────┴─────────┘
┌─────users─┬─device─┐
│ 70608335 │ tablet │
└──────────┴────────┘

3 rows in set. Elapsed: 7.650 sec. Processed 13.23 billion rows, 158.71 GB (1.73 billion rows/s., 20.75 GB/s.)

:) 
```

快

快

全球互联网技术大会
GLOBAL INTERNET TECHNOLOGY CONFERENCE

快

全球互联网技术大会
GLOBAL INTERNET TECHNOLOGY CONFERENCE

为啥这么快?

1. 优秀的代码编写，强大的底层优化，严格的单元测试

2. A vector engine & Code generation

3. CPU底层指令集的使用

4. C++新特性

为啥这么快?

Vectorized processing

Data is represented as small single-dimensional arrays (vectors), easily accessible for CPUs.
The percentage of instructions spent in interpretation logic is reduced by a factor equal to the vector-size
The functions that perform work now typically process an array of values in a tight loop
Tight loops can be optimized well by compilers, enable compilers to generate SIMD instructions automatically.
Modern CPUs also do well on such loops, out-of-order execution in CPUs often takes multiple loop iterations into execution concurrently, exploiting the deeply pipelined resources of modern CPUs.
It was shown that vectorized execution can improve data-intensive (OLAP) queries by a factor 50.

ClickHouse函数：

300+

ClickHouse高级函数：

统计类

quantile(level)(x)

quantiles(level1, level2, ...)(x)

median(X)

URL截取

cutQueryString(X)

domain(X)

其他

today()-1

yesterday()

substring(s, offset, length)

IPv4NumToStringClassC

extract(haystack, pattern)

高级函数举例：

- 域名类

```
SELECT protocol('https://weibo.com/jackpgao/home?wvr=5') as protocol;
┌─protocol─┐
│ https    │
└──────────┘

SELECT domain('https://weibo.com/jackpgao/home?wvr=5') as domain;
┌─domain───┐
│ weibo.com │
└──────────┘

SELECT domainWithoutWWW('https://weibo.com/jackpgao/home?wvr=5') as domainWithoutWWW;
┌─domainWithoutWWW─┐
│ weibo.com        │
└──────────────────┘

SELECT cutQueryString('https://weibo.com/jackpgao/home?wvr=5') as cutQueryString;
┌─cutQueryString──────────────────┐
│ https://weibo.com/jackpgao/home │
└─────────────────────────────────┘

SELECT path('https://weibo.com/jackpgao/home?wvr=5') as path;
┌─path─────────────┐
│ /jackpgao/home   │
└──────────────────┘

SELECT pathFull('https://weibo.com/jackpgao/home?wvr=5') as pathFull;
┌─pathFull─────────────────┐
│ /jackpgao/home?wvr=5     │
└──────────────────────────┘
```

高级函数举例：

- IP归类

```
SELECT
    IPv4NumToStringClassC(IPv4StringToNum(clientip)) AS k,
    count()
FROM xxx
WHERE (date = toString(today())) AND (hour = 12)
GROUP BY k
ORDER BY count() DESC
LIMIT 10
```

k	count()
118.190.218.xxx	99353
59.63.248.xxx	64207
122.224.52.xxx	49156
117.136.79.xxx	47350
120.241.4.xxx	47041
59.63.249.xxx	45717
115.124.31.xxx	40806
111.13.31.xxx	38786
223.104.3.xxx	37271
0.0.0.xxx	36596

来点干货

目录结构

```
├── config-preprocessed.xml
├── config.xml
├── users-preprocessed.xml
└── users.xml
├── metrika.xml

├── cores
├── flags
├── status

├── tmp

├── log
│   ├── error.log
│   ├── server.log
│   ├── stderr
│   └── stdout

├── metadata
│   ├── apm.sql
│   ├── default
│   ├── gaopeng4
│   │   └── apm_msg.sql
│   └── system

├── data
│   ├── default
│   ├── gaopeng4
│   │   └── apm_msg
│   │       ├── 20171001_20171001_0_9_2
│   │       │   ├── checksums.txt
│   │       │   ├── columns.txt
│   │       │   ├── date.bin
│   │       │   ├── date.mrk
│   │       │   ├── minute.bin
│   │       │   ├── minute.mrk
│   │       │   └── primary.idx
│   │       └── detached
│   └── system
```

配置文件

状态文件

日志

元数据

数据目录

配置文件：

config.xml

```xml
<?xml version="1.0"?>
<yandex>
    <logger>
        <level>trace</level>
        <log>/data1/clickhouse/log/server.log</log>
        <errorlog>/data1/clickhouse/log/error.log</errorlog>
        <size>1000M</size>
        <count>10</count>
    </logger>
    <http_port>8123</http_port>
    <tcp_port>9000</tcp_port>
    <interserver_http_port>9009</interserver_http_port>
    <interserver_http_host>ck21.[redacted].com.cn</interserver_http_host>
    <listen_host>0.0.0.0</listen_host>
    <max_connections>4096</max_connections>
    <keep_alive_timeout>3</keep_alive_timeout>
    <max_concurrent_queries>100</max_concurrent_queries>
    <uncompressed_cache_size>8589934592</uncompressed_cache_size>
    <mark_cache_size>10737418240</mark_cache_size>
    <path>/data1/clickhouse/</path>
    <tmp_path>/data1/clickhouse/tmp/</tmp_path>
    <users_config>users.xml</users_config>
    <default_profile>default</default_profile>

    <log_queries>1</log_queries>
    <default_database>default</default_database>
    <remote_servers incl="clickhouse_remote_servers" />
    <zookeeper incl="zookeeper-servers" optional="true" />
    <macros incl="macros" optional="true" />
    <builtin_dictionaries_reload_interval>3600</builtin_dictionaries_reload_interval>
    <max_table_size_to_drop>0</max_table_size_to_drop>
    <include_from>/data1/clickhouse/metrika.xml</include_from>
</yandex>
```

日志

本节点信息

本机域名

本地配置

集群相关配置

配置文件：

metrika.xml

```
<yandex>
<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>
```

配置文件：

metrika.xml

```
<zookeeper-servers>
  <node index="1">
    <host>1.[redacted].sina.com.cn</host>
    <port>2181</port>
  </node>
  <node index="2">
    <host>2.[redacted].sina.com.cn</host>
    <port>2181</port>
  </node>
  <node index="3">
    <host>3.[redacted].sina.com.cn</host>
    <port>2181</port>
  </node>
</zookeeper-servers>

<macros>
    <replica>ck11</replica>
</macros>

<networks>
   <ip>::/0</ip>
</networks>

<clickhouse_compression>
<case>
  <min_part_size>10000000000</min_part_size>
  <min_part_size_ratio>0.01</min_part_size_ratio>
  <method>lz4</method>
</case>
</clickhouse_compression>
</yandex>
```

配置文件：

user.xml

```
<profiles>
    <default>
        <max_memory_usage>10000000000</max_memory_usage>
        <use_uncompressed_cache>0</use_uncompressed_cache>
        <load_balancing>random</load_balancing>
    </default>

    <readonly>
        <max_memory_usage>10000000000</max_memory_usage>
        <use_uncompressed_cache>0</use_uncompressed_cache>
        <load_balancing>random</load_balancing>
        <readonly>1</readonly>
    </readonly>
</profiles>

<quotas>
    <!-- Name of quota. -->
    <default>
        <interval>
            <duration>3600</duration>
            <queries>0</queries>
            <errors>0</errors>
            <result_rows>0</result_rows>
            <read_rows>0</read_rows>
            <execution_time>0</execution_time>
        </interval>
    </default>
</quotas>
```

http://jackpgao.github.io/2017/11/20/ClickHouse-User-Password/

配置文件：

user.xml

```
<users>
    <default>
        <password_sha256_hex>967f3bf355dddfabfca1c9f5cab39352b2ec1cd0b05f9e1e6b8f629705fe7d6e</password_sha256_hex>
        <networks incl="networks" replace="replace">
            <ip>::/0</ip>
        </networks>
        <profile>default</profile>
        <quota>default</quota>
    </default>

    <ck>
        <password_sha256_hex>967f3bf355dddfabfca1c9f5cab39352b2ec1cd0b05f9e1e6b8f629705fe7d6e</password_sha256_hex>
        <networks incl="networks" replace="replace">
            <ip>::/0</ip>
        </networks>
        <profile>readonly</profile>
        <quota>default</quota>
    </ck>
</users>
```

```
PASSWORD=$(base64 < /dev/urandom | head -c8);
echo "$PASSWORD"; echo -n "$PASSWORD" | sha256sum | tr -d '-'
```

http://jackpgao.github.io/2017/11/20/ClickHouse-User-Password/

ClickHouse的问题：

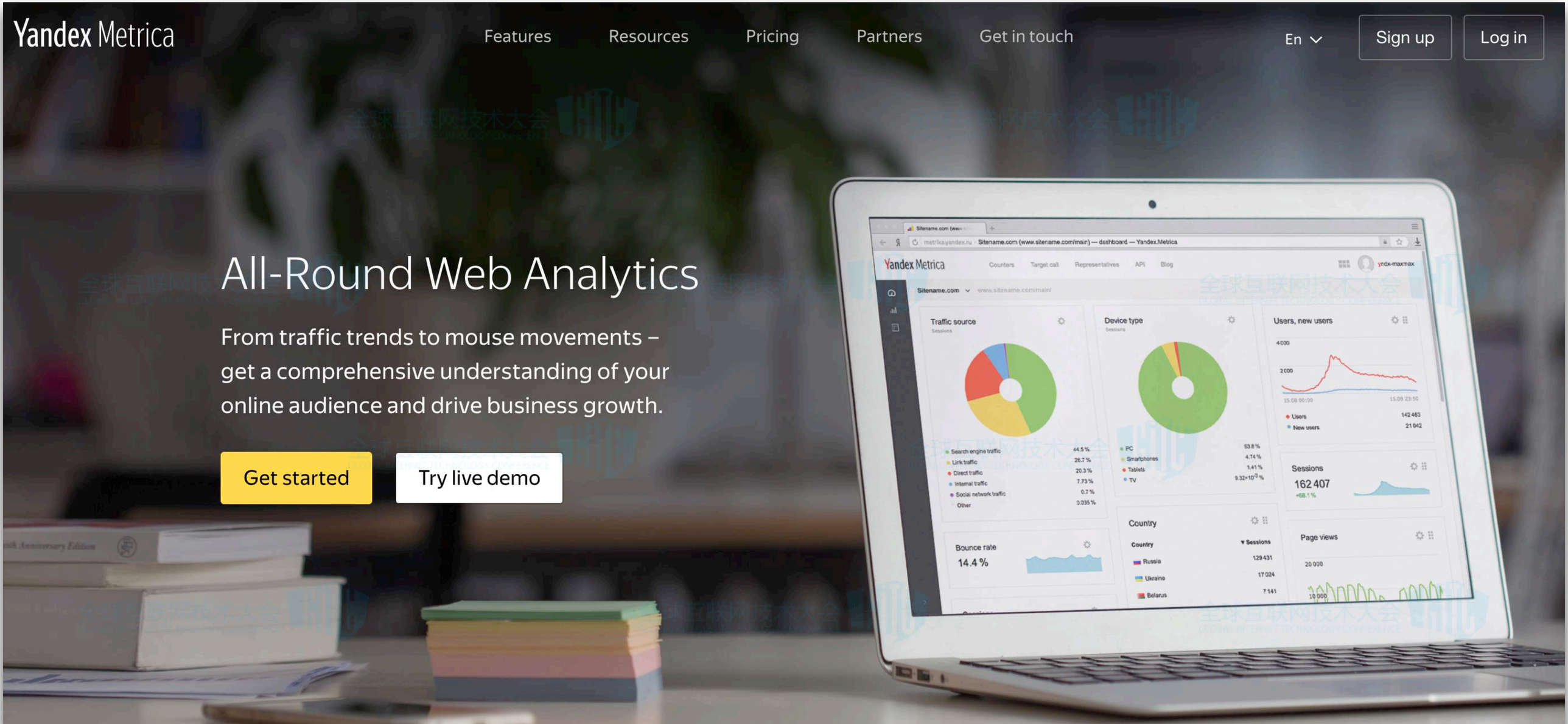
1. insert into xx (a, b, c) values ('a', 'b', 'c') 只能是单引号

2. 如果是int插入的是string，报错（不确定是否有类似SQL_MODE的参数）

3. 删除只支持到月纬度的分区

4. 改造官方的启动脚本，不要用root直接启动

谁在用


Yandex Metrica
Features
Resources
Pricing
Partners
Get in touch
En
Sign up
Log in
All-Round Web Analytics
From traffic trends to mouse movements – get a comprehensive understanding of your online audience and drive business growth.
Get started
Try live demo
Traffic source
Device type
Users, new users
Sessions
162 407
Bounce rate
14.4 %
Country
Page views