携程新一代监控告警平台 Hickwall

陈汉

Agenda

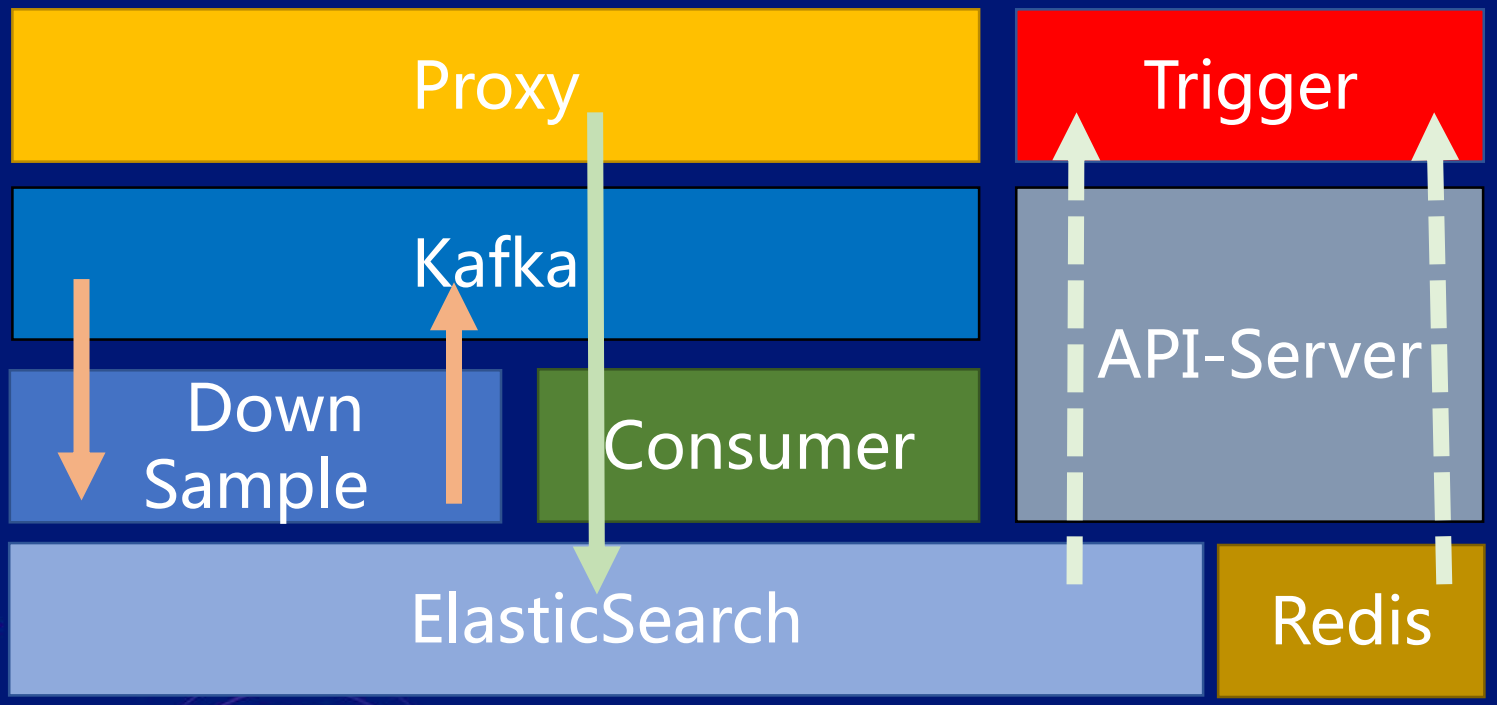
Part 1 hickwall架构演进

Part 2 influxdb集群设计

Part 3 数据聚合的探索

Part 4 流式告警的实现

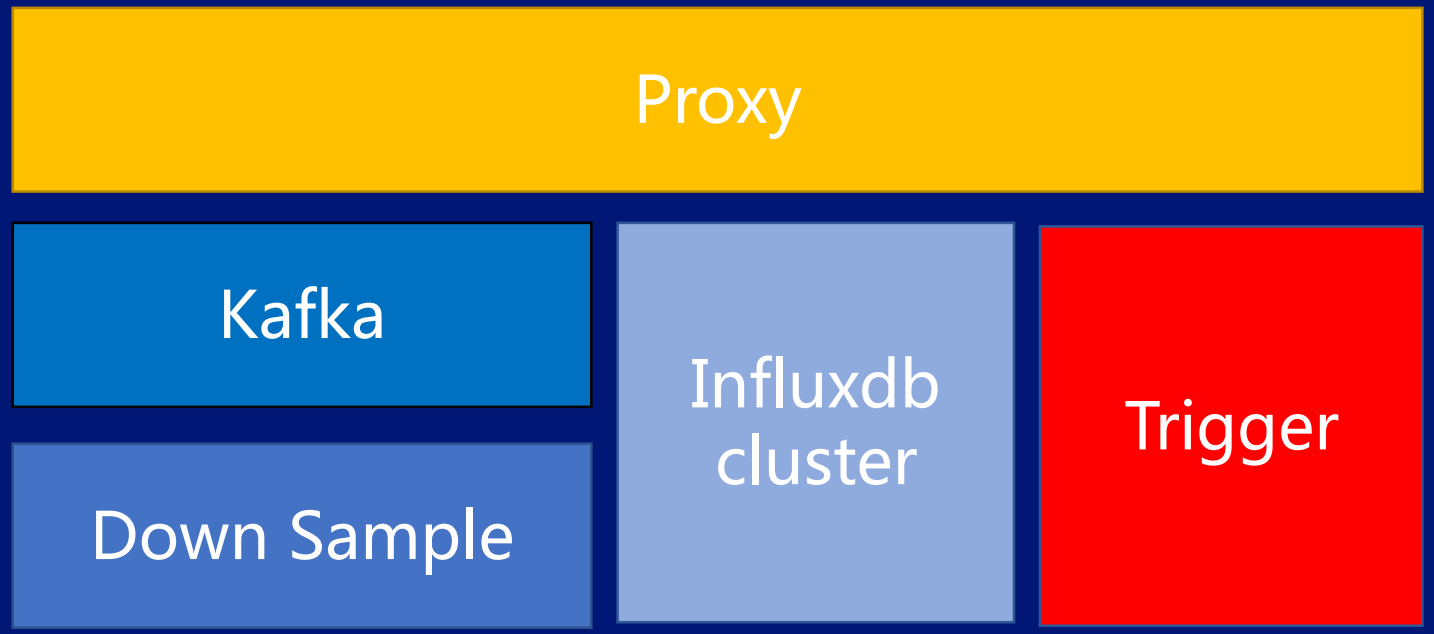
第一代架构



- 组件太多
- 数据堆积
- 链条过长

Proxy → Kafka → Consumer → Redis → API-Server → Trigger

当前架构



1. 存储
2. 聚合
3. 告警

Agenda

Part 1 hickwall架构演进

Part 2 influxdb集群设计

Part 3 数据聚合的探索

Part 4 流式告警的实现

Elasticsearch

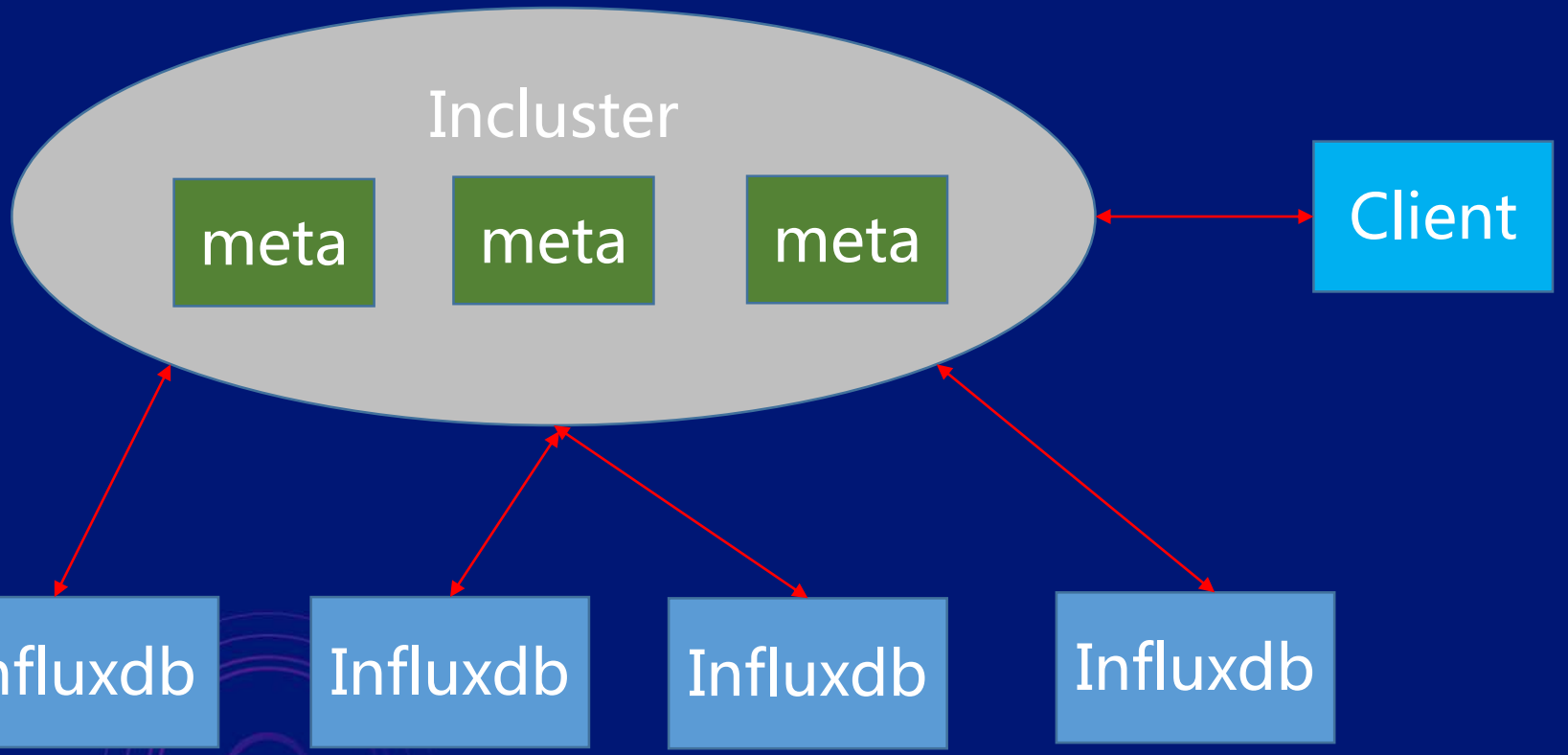
1、磁盘空间占用大（200+T）
2、磁盘IO使用多
3、索引维护复杂
4、写入查询速度慢(5w/s)

VS

Influxdb

1、针对时间范围高效查询
2、Down Sample*
3、自动删除过时数据
4、较低的使用成本

Incluster架构



- 低耦合
- CAP
- 负载均衡
- 灾备

数据分布策略

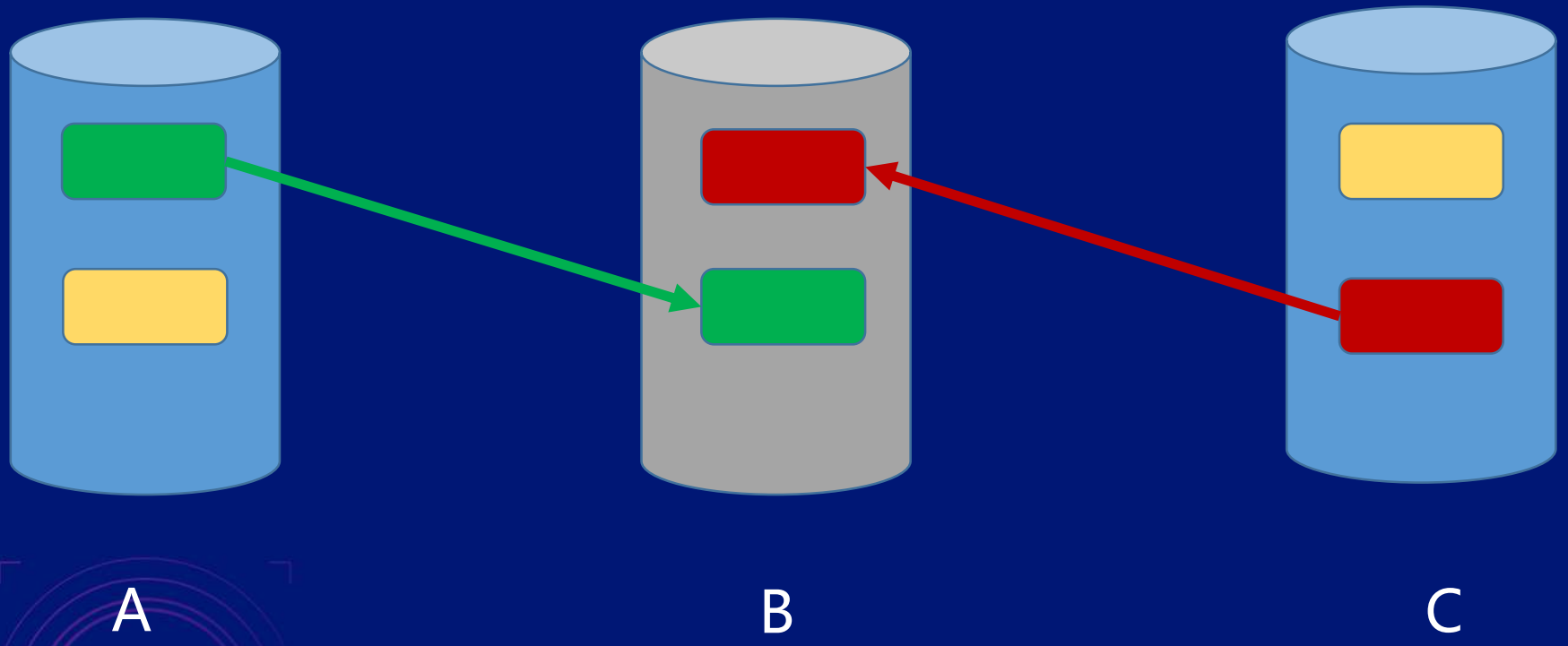
1、数据特征->减少数据热点
2、查询特征->减少查询节点

- Series
- Measurement
- Measurement + Tags

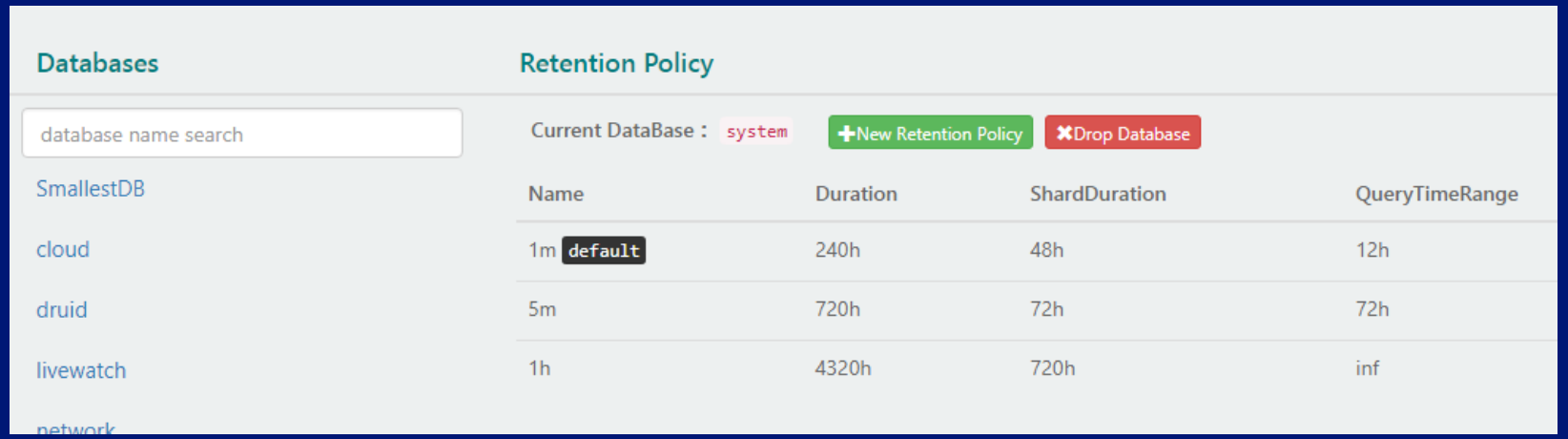
cpu.load，appid，pool，server ——> Measurement

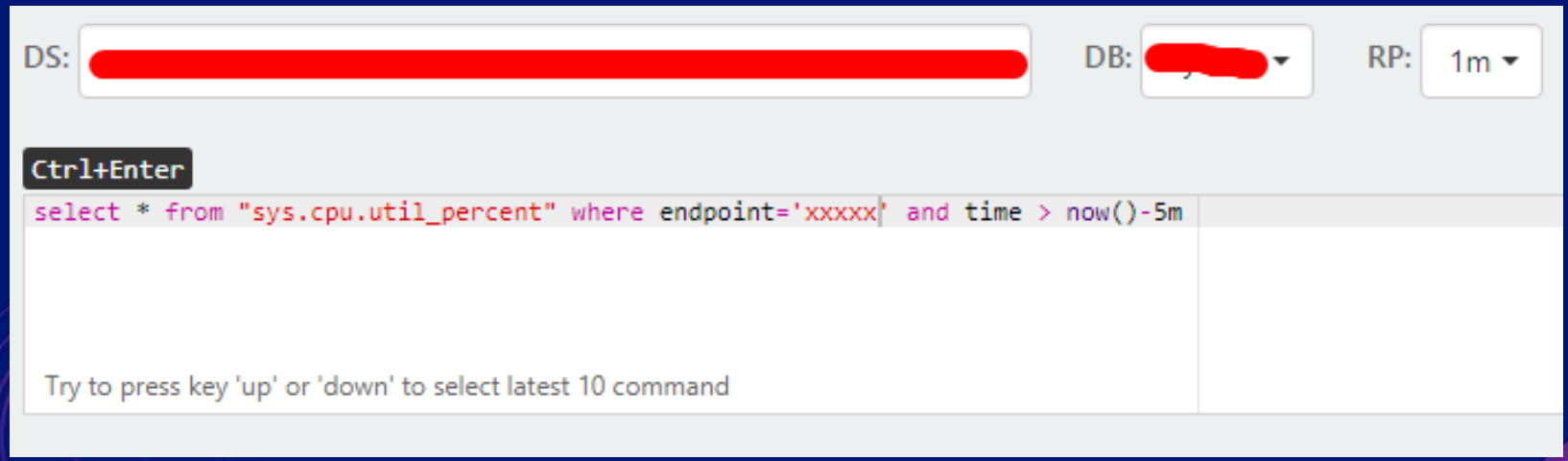
request.count，appid，... ——> Measurement+appid

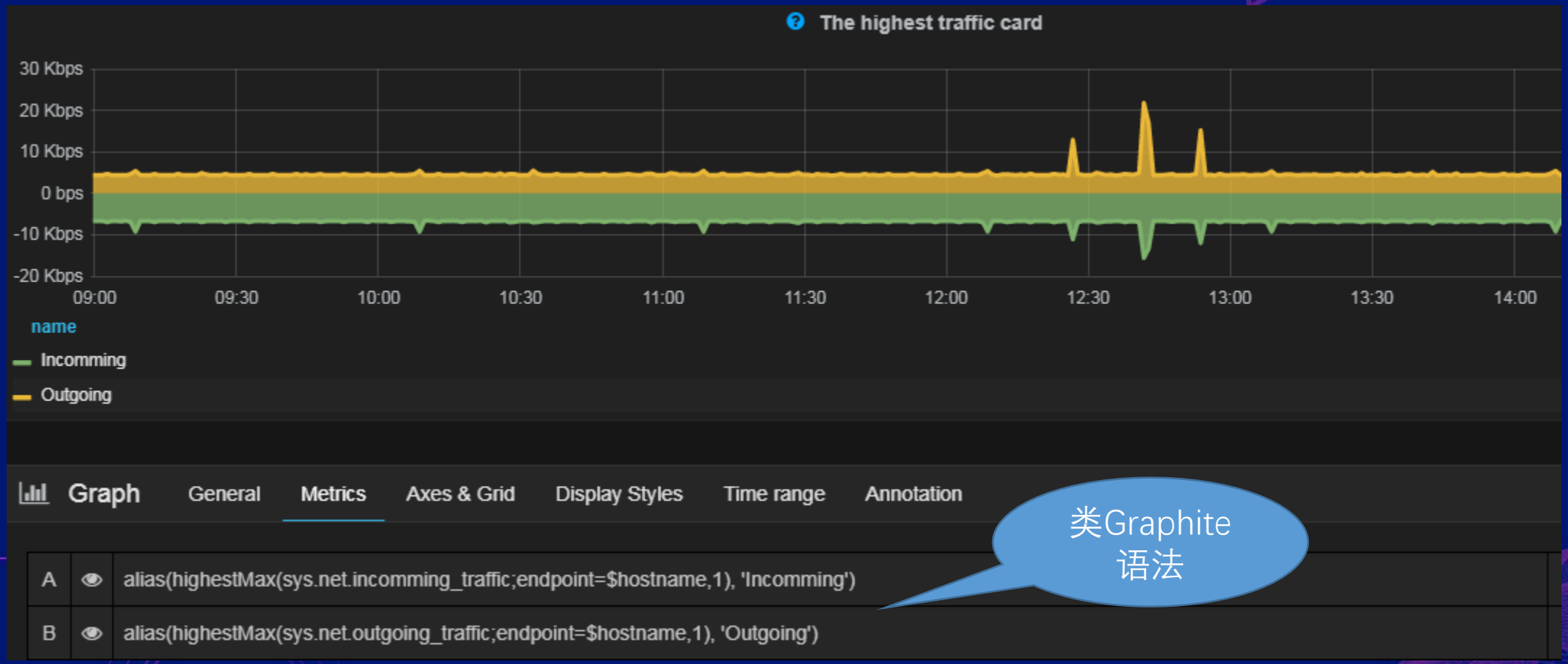
数据恢复



Incluster管理界面

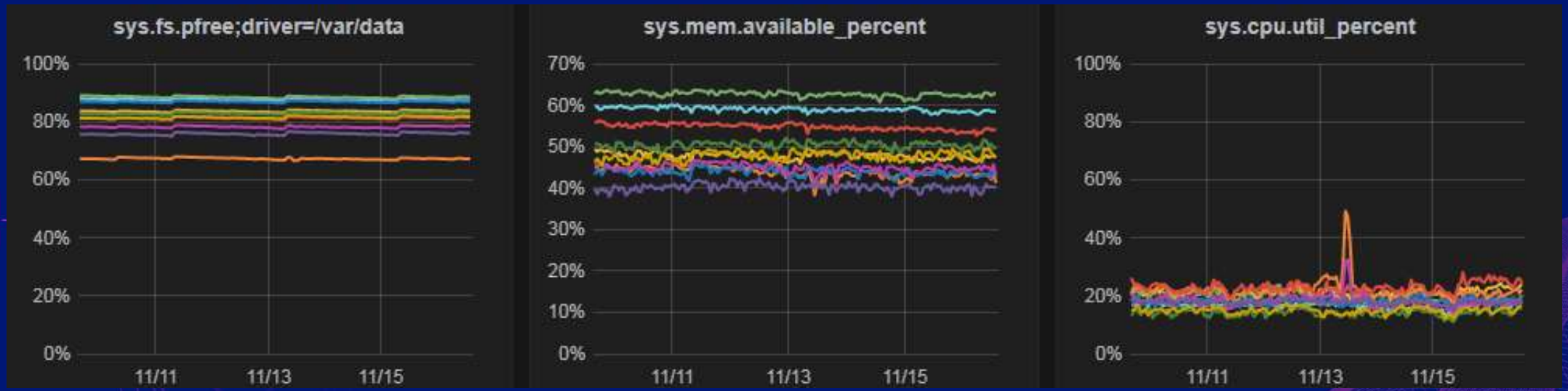





The highest traffic card
30 Kbps
20 Kbps
10 Kbps
0 bps
-10 Kbps
-20 Kbps
09:00
09:30
10:00
10:30
11:00
11:30
12:00
12:30
13:00
13:30
14:00
name
Incomming
Outgoing
Graph
General
Metrics
Axes & Grid
Display Styles
Time range
Annotation
A
alias(highestMax(sys.net.incomming_traffic;endpoint=$hostname,1), 'Incomming')
B
alias(highestMax(sys.net.outgoing_traffic;endpoint=$hostname,1), 'Outgoing')
类Graphite
语法

Incluster的使用

1、10台40C，128G，4T
2、1m-10天，5m-30天，1h-180天
3、约7500w series，45w points/s



Agenda

Part 1 hickwall架构演进

Part 2 influxdb集群设计

Part 3 数据聚合的探索

Part 4 流式告警的实现

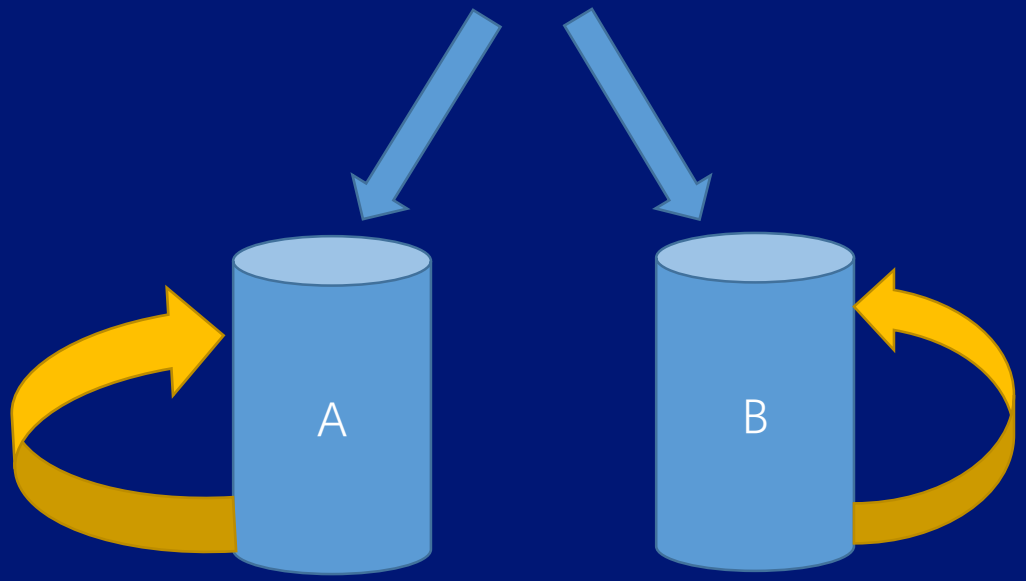
Continuous query

1、内存占用高

2、只作用于本节点

3、集群维护麻烦

4、资源浪费



时间维度聚合Down Sample

有状态计算的挑战

1、内存（指定partition、元数据去重）

2、准确（指定时间范围）

4个节点(8C18G)，每个节点峰值处理能力2400w

业务场景的聚合挑战

1、 查询基数高

2、 聚合逻辑比较复杂

```
req.count appid=001 hostname=server001 type=A source=S …
req.count appid=001 hostname=server002 type=A source=S …
…
```

ClickHouse

面向OLAP的分布式列式数据库

1、高性能读写

2、提供sql语言和上百个函数

```
Select quantileExact(0.95)(value) as p95,appid,
    toUInt32(toStartOfMinute(timestamp)) as time
From ……
Group By time,appid
```

Agenda

Part 1 hickwall架构演进

Part 2 influxdb集群设计

Part 3 数据聚合的探索

Part 4 流式告警的实现

Pull

1、读取频率高
2、响应时间要求高
3、数据可靠性要求高

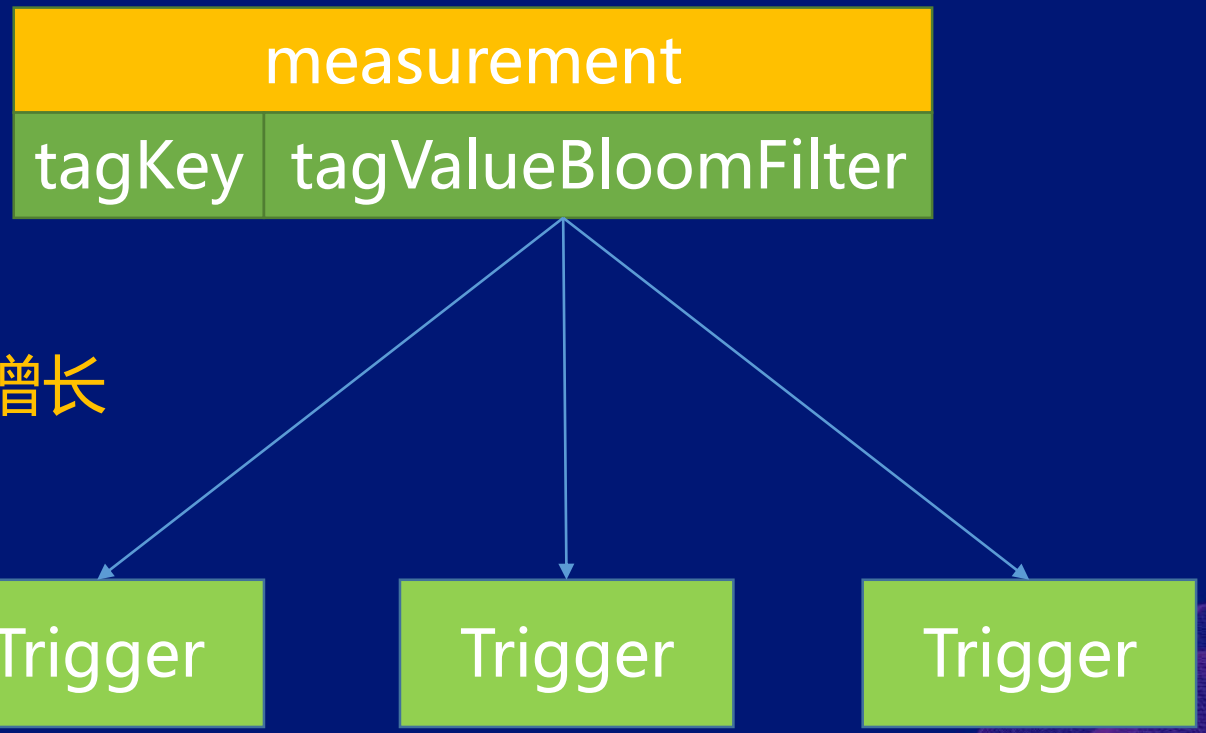
告警数据占比少
每个告警所需数据少
告警逻辑灵活多变
时效性极强

Stream

1、数据过滤
2、无读取压力
3、可靠实时

数据订阅

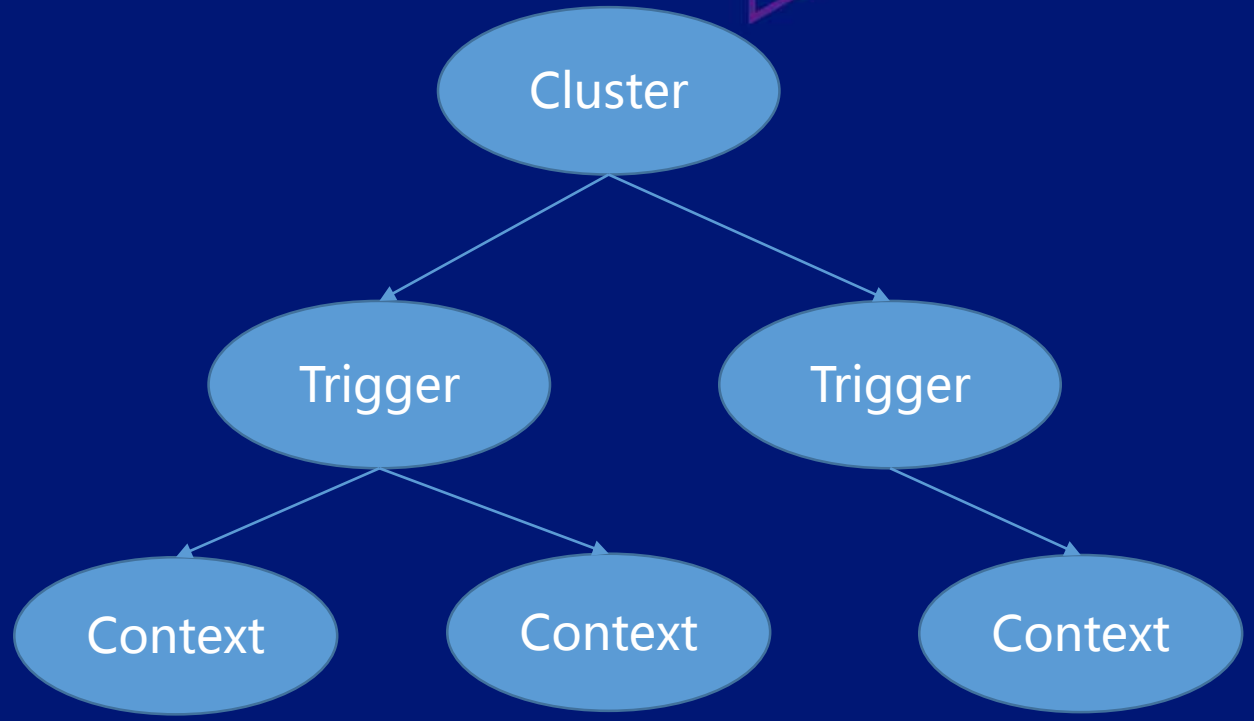
1、measurement精确匹配
2、tagValue布隆过滤器



时间复杂度不随规则数量线性增长

数据处理

1、akka异步高并发
2、actor编程模型

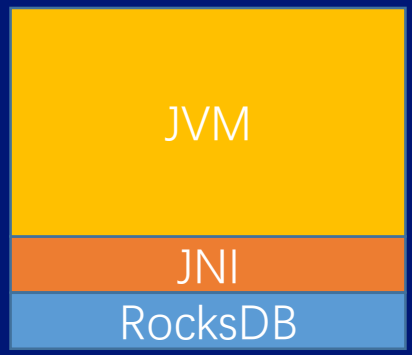


数据存储

1、In Memory

2、RocksDB

* 写入速率高(LSM Tree)
* 减少内存使用
* 减少fullGC



Init DSL:

```
T.require('m1', "sys.cpu.util_percent","endpoint",M.hostName, "5m", 4);
```

Run DSL:*

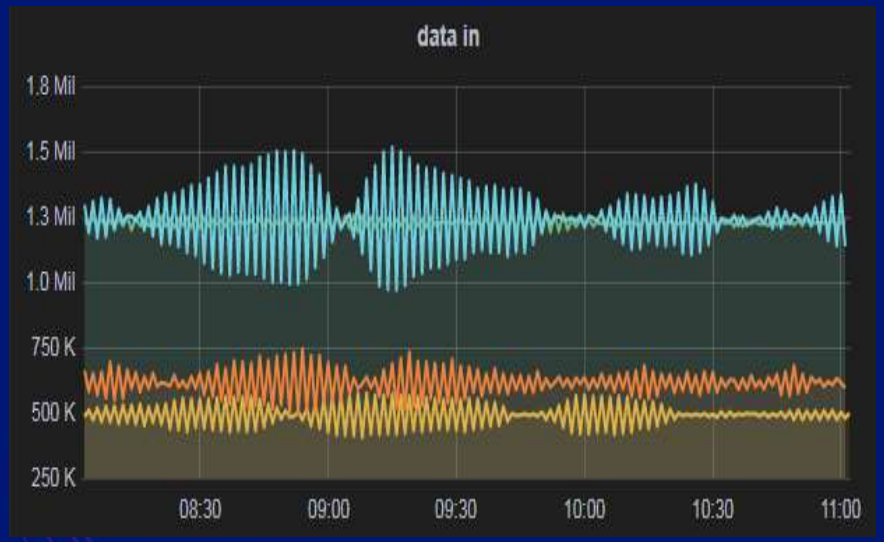
```
var ValueThreshold=M.ValueThreshold||90;
var CountThreshold=M.CountThreshold||3;
var m1=V.m1;

for(var endpoint in m1){
    if(m1[endpoint].len()<CountThreshold) continue;
    if(m1[endpoint].streak(">=",ValueThreshold) >= CountThreshold ){
        return WARNING("Processor(_Total)\\% Processor Time:Processor Time is overloaded");
    }else{
        return OK("");
    }
}
```

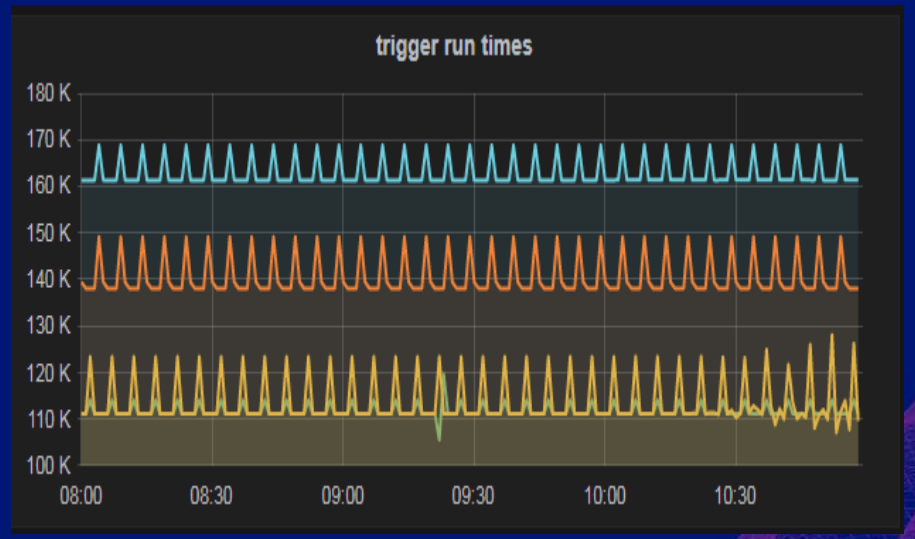
语法检查

回测

4个节点，每个节点6G

每分钟50w+次告警逻辑，400w+points





本PPT来自2018携程技术峰会

更多技术干货，请关注“携程技术中心”微信公众号