大数据·反欺诈·价值

从零打造千万级的实时风控云服务

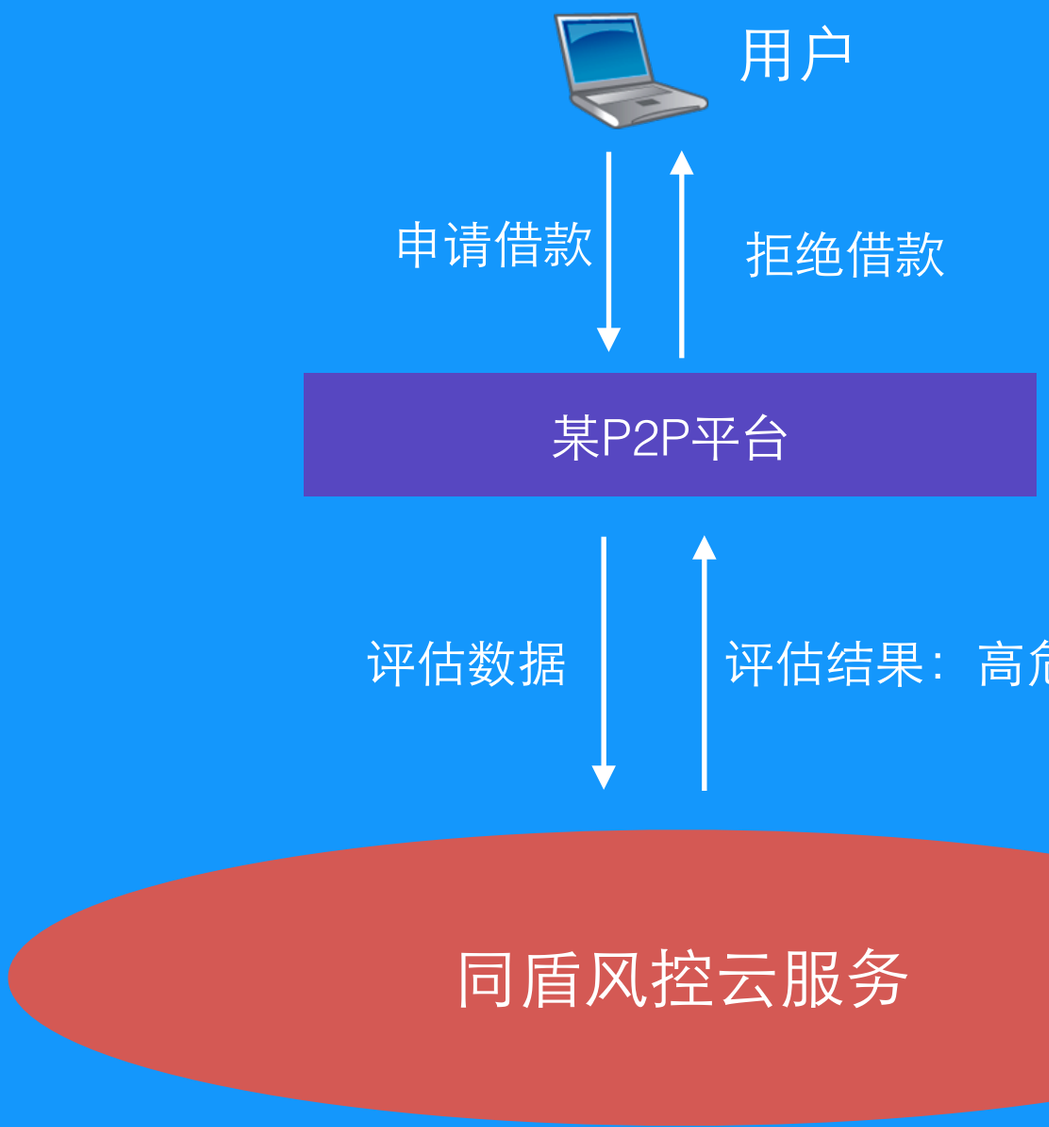
张新波

杭州同盾科技有限公司

关于同盾

- 2014年11份创建，目前有80人的团队
- 由原阿里、PayPal等反欺诈领域专家创建
- 国内第一家风险控制与反欺诈云服务提供商
- 服务领域涉及电商、支付、P2P金融、O2O等

我们的服务



其它场景：
注册
登陆
交易
支付
发消息
。。。

面临的挑战

性能

- 实时性要求非常高，响应时间通常小于500ms
- 无法直接拿到结果，全部需要实时动态计算
- 参与计算的数据量庞大，计算维度多且复杂
- 无法像静态资源那样做多机房的缓存和加速

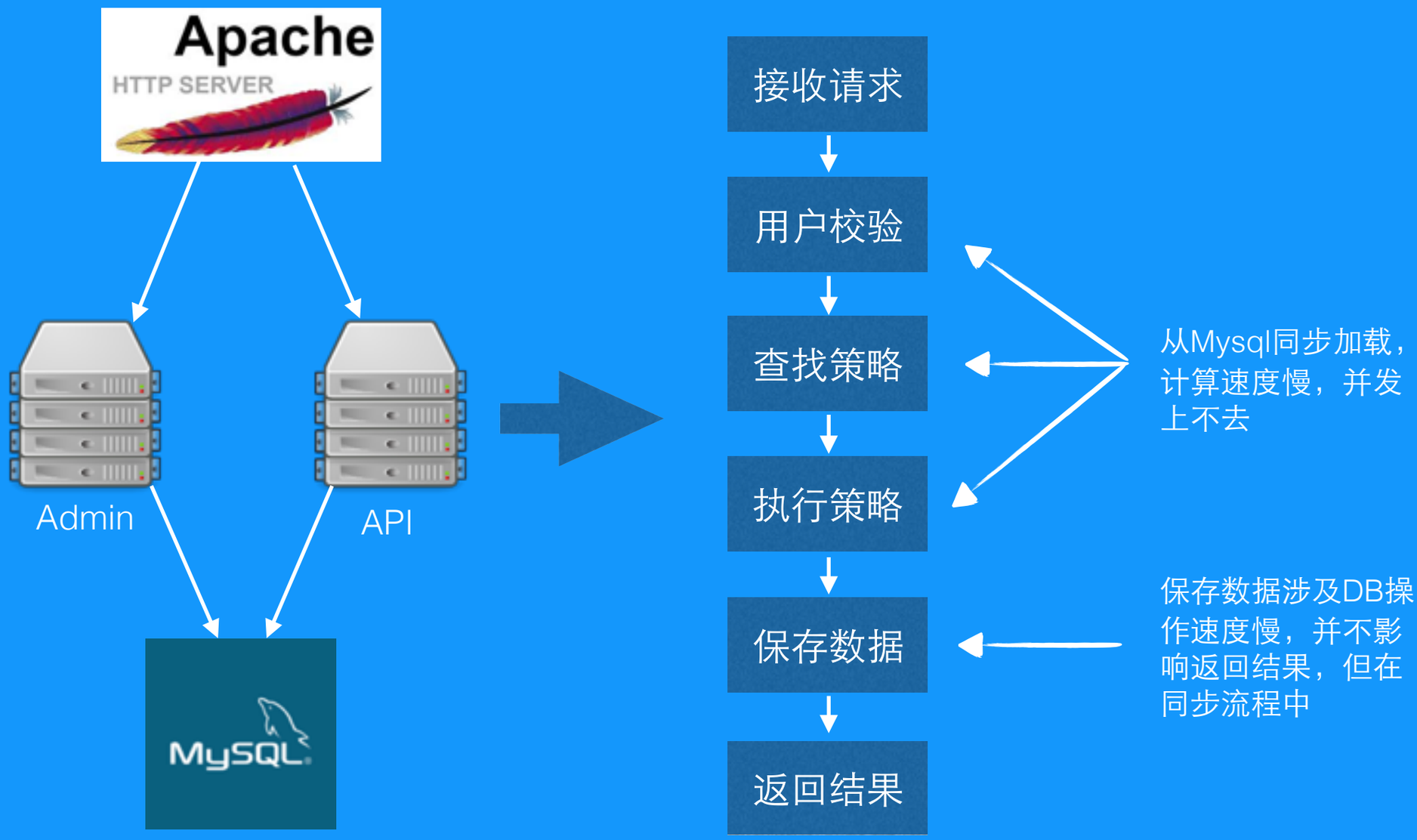
可用性

- 保证服务的稳定性，消除单点故障
- 完备的灾备和紧急处理方案，以备不时之需

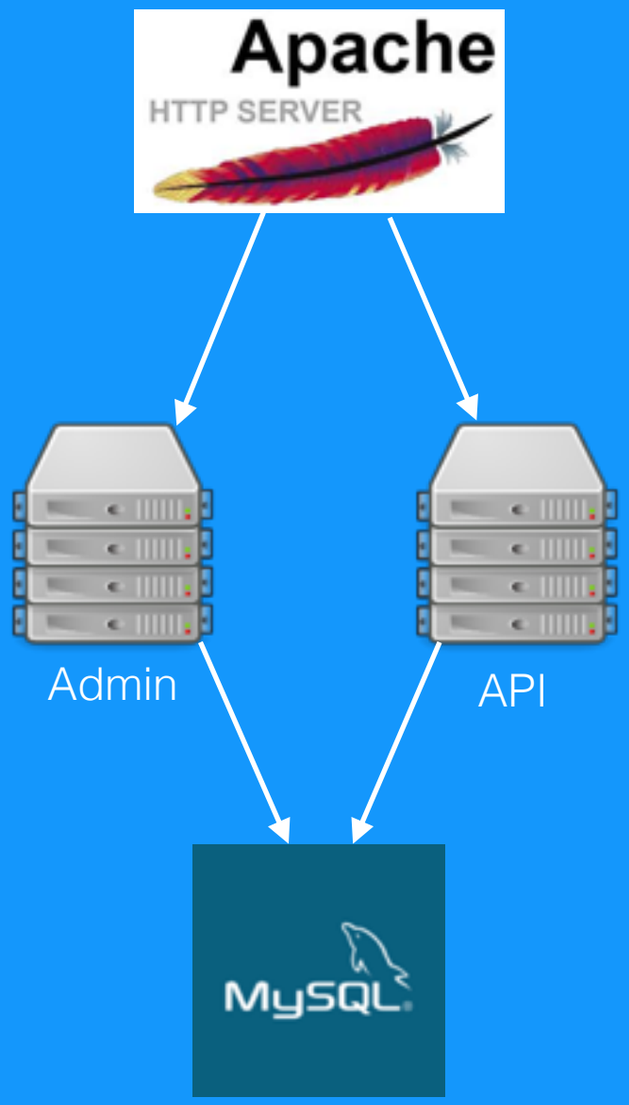
可扩展性

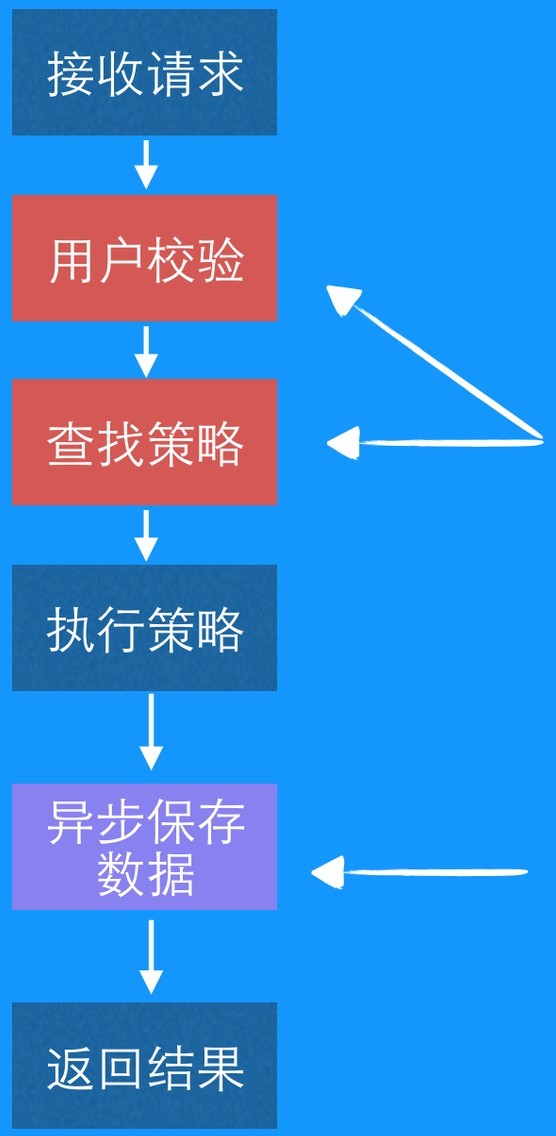
- 应用服务器可线性扩展，支撑业务的快速发展
- 海量数据的存储和计算，支持线性扩展

V0.1的架构


Apache
HTTP SERVER
Admin
API
MySQL
接收请求
用户校验
查找策略
执行策略
保存数据
返回结果
从Mysql同步加载，计算速度慢，并发上不去
保存数据涉及DB操作速度慢，并不影响返回结果，但在同步流程中

V0.2的架构

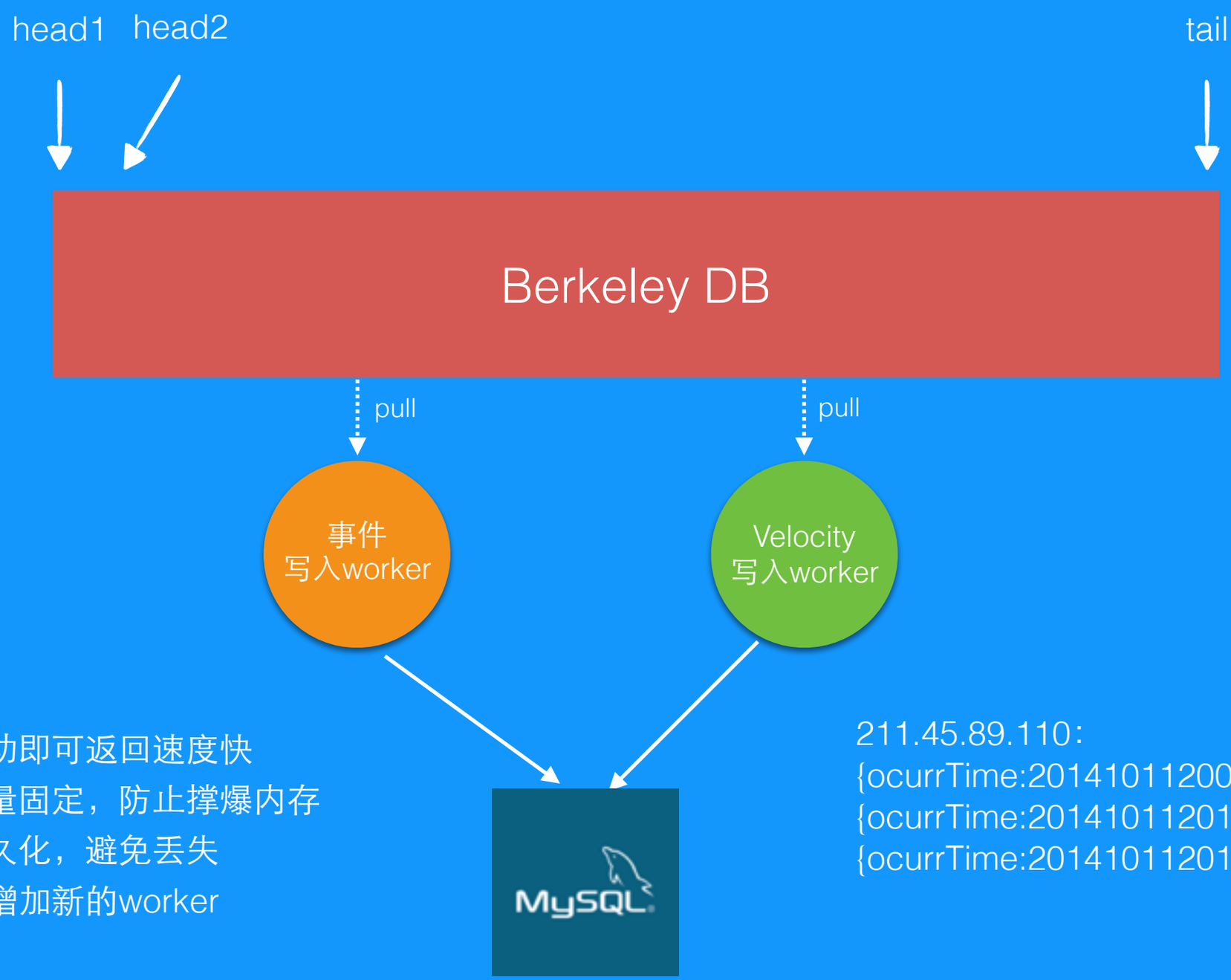




使用Guava Cache定时预先加载，缓存全部用户及对应的策略数据

使用线程池—>本地队列

本地队列



优点:

1.写入队列成功即可返回速度快

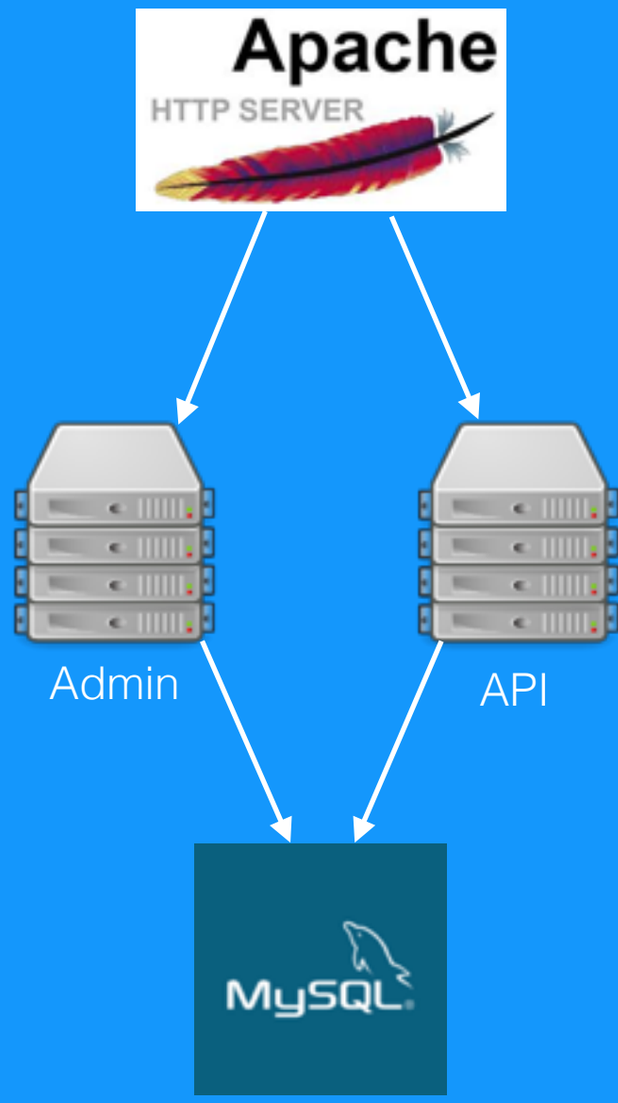
2.占用内存容量固定，防止撑爆内存

3.数据可以持久化，避免丢失

4.可以方便地增加新的worker

211.45.89.110:
{ocurrTime:20141011200932,payAmount:35}
{ocurrTime:20141011201021,payAmount:47}
{ocurrTime:20141011201458,payAmount:100}

V0.3的架构



优点：

1.大大减少MySQL的读取，性能有极大提升

2.使用压缩数据，大大减少了内存占用和网络开销

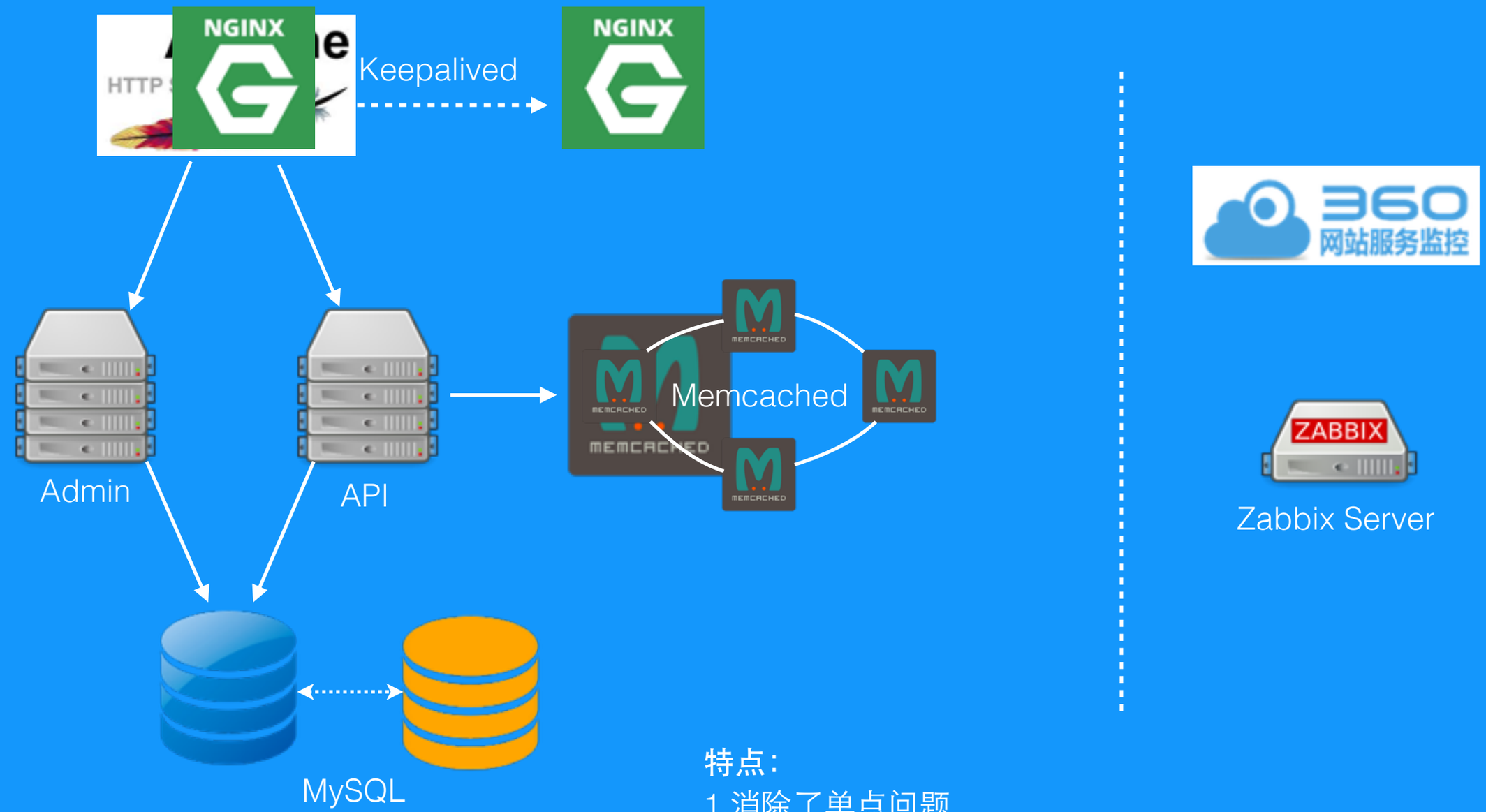
ipAddress.211.45.89.110：{};{};{}

userAccount.hello1234：{};{};{}

deviceId.1233215dc70b3274d43e459baa84ced8：{};{};{}

value使用base64+Gzip进行压缩

V1.0的架构



特点：

1.消除了单点问题

2.具备了完善的监控

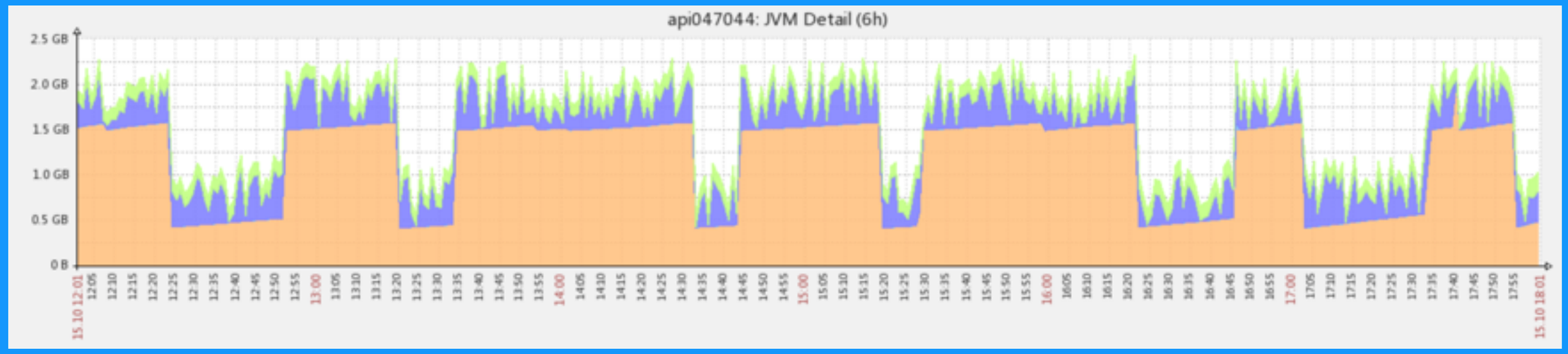
3.能支撑百万级的访问

MySQL的优化

- SAS RAID5 —> SSD RAID10
- MySQL 5.1 —> MySQL 5.6
- 默认配置参数 —> 缓存优化等
- 单机 —> 主备
- 单表 —> 按客户分表

JVM 的优化

- 并发场景下HashMap序列化导致的内存突然飙高



- +XX:DisabledExplicitGC 导致Direct Buffer无法回收
- 使用 -XX:+CMSIncrementalMode导致性能更加恶化
- 增加-XX:+PrintGCDateStamps -XX:+PrintGCDetails -Xloggc:logs/gc.log方便GC分析

CPU Load高的优化

```
top - 14:59:11 up 148 days, 18:28,  3 users,  load average: 13.64, 13.61, 13.78
Tasks: 137 total,   1 running, 136 sleeping,   0 stopped,   0 zombie
Cpu(s): 99.2%us,  0.5%sy,  0.0%ni,  0.2%id,  0.0%wa,  0.0%hi,  0.2%si,  0.0%st
Mem:   3924572k total,  3798320k used,   126252k free,    17776k buffers
Swap:  8191992k total,   180044k used,  8011948k free,   796328k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
 1193 admin     20   0 5844m 2.7g 6380 S 397.5 71.6   9477:47 java
18177 zabbix    20   0 76564  780  624 S  1.3  0.0  15:07.56 zabbix_agentd
   19 root      20   0     0    0    0 S  0.3  0.0   5:56.15 events/0
```

```
[xinbo.zhang@api047036 ~]$ top -H -p 1193
top - 17:17:28 up 148 days, 20:46,  3 users,  load average: 15.37, 15.36, 15.18
Tasks: 574 total,  14 running, 560 sleeping,   0 stopped,   0 zombie
Cpu(s): 98.4%us,  1.1%sy,  0.0%ni,  0.2%id,  0.0%wa,  0.0%hi,  0.3%si,  0.0%st
Mem:   3924572k total,  3791972k used,   132600k free,    19344k buffers
Swap:  8191992k total,   196012k used,  7995980k free,   769060k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
 1312 admin     20   0 5906m 2.7g 6376 R 28.1 71.9 803:03.57 java
 1395 admin     20   0 5906m 2.7g 6376 R 28.1 71.9 746:05.03 java
 8043 admin     20   0 5906m 2.7g 6376 R 27.8 71.9 130:50.07 java
11147 admin     20   0 5906m 2.7g 6376 R 27.8 71.9 519:29.27 java
22717 admin     20   0 5906m 2.7g 6376 R 27.8 71.9 104:55.01 java
32026 admin     20   0 5906m 2.7g 6376 R 27.4 71.9 406:48.21 java
 1299 admin     20   0 5906m 2.7g 6376 R 25.1 71.9 448:24.99 java
32664 admin     20   0 5906m 2.7g 6376 R 24.8 71.9 656:04.31 java
32028 admin     20   0 5906m 2.7g 6376 R 24.8 71.9 406:45.57 java
11877 admin     20   0 5906m 2.7g 6376 R 24.1 71.9   1847:28 java
22719 admin     20   0 5906m 2.7g 6376 R 23.8 71.9 104:44.32 java
 1298 admin     20   0 5906m 2.7g 6376 R 23.1 71.9   1783:48 java
 4825 admin     20   0 5906m 2.7g 6376 R 23.1 71.9 758:42.39 java
 1296 admin     20   0 5906m 2.7g 6376 S 22.8 71.9 436:37.00 java
11145 admin     20   0 5906m 2.7g 6376 R 22.8 71.9 519:47.74 java
 1297 admin     20   0 5906m 2.7g 6376 S  5.0 71.9  82:38.77 java
 1250 admin     20   0 5906m 2.7g 6376 S  1.3 71.9  18:12.51 java
 1300 admin     20   0 5906m 2.7g 6376 S  1.3 71.9  40:36.95 java
 1301 admin     20   0 5906m 2.7g 6376 S  1.0 71.9  40:33.46 java
```

```
http-nio-8088-exec-3" daemon prio=10 tid=0x00007f2a7c003000 nid=0x520 runnable [0x00007f2a8d01d000]
   java.lang.Thread.State: RUNNABLE
        at java.util.TreeMap.put(TreeMap.java:560)
        at java.util.AbstractMap.putAll(AbstractMap.java:273)
        at java.util.TreeMap.putAll(TreeMap.java:322)
        at java.util.TreeMap.<init>(TreeMap.java:180)
        at com.alibaba.fastjson.serializer.MapSerializer.write(MapSerializer.java:48)
        at com.alibaba.fastjson.serializer.JSONSerializer.writeWithFieldName(JSONSerializer.java:398)
        at Serializer_1.write1(Unknown Source)
        at Serializer_1.write(Unknown Source)
        at com.alibaba.fastjson.serializer.JSONSerializer.write(JSONSerializer.java:369)
        at com.alibaba.fastjson.JSON.toJSONString(JSON.java:414)
        at com.alibaba.fastjson.JSON.toJSONString(JSON.java:402)
        at cn.fraudmetrix.forseti.api.service.RiskServiceImpl.execute(RiskServiceImpl.java:167)
```

```
166
167         log.info("CTX:" + response.getSeq_id() + "," + JSON.toJSONString(context));
```

https://github.com/oldratlee/useful-shells/blob/master/show-busy-java-threads.sh

用好JVM工具

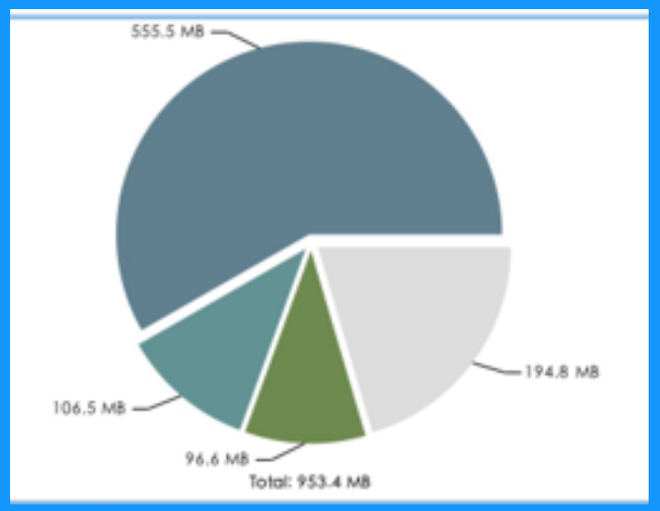
jstat

```
^C[xinbo.zhang@api01_host ~]$ sudo -u admin jstat -gccause -t -h 10 15498 10000
Timestamp         S0     S1     E      O      P     YGC     YGCT    FGC    FGCT     GCT    LGCC                 GCC
         7281.4   1.52   0.00  88.42  49.92  51.10   1298   24.503    16   21.715   46.218 Allocation Failure   No GC
         7291.5   2.06   0.00  71.62  49.92  51.10   1300   24.522    16   21.715   46.237 Allocation Failure   No GC
         7301.5   1.73   0.00  48.90  49.93  51.10   1302   24.542    16   21.715   46.256 Allocation Failure   No GC
         7311.5   1.49   0.00  24.76  49.95  51.10   1304   24.561    16   21.715   46.275 Allocation Failure   No GC
         7321.5   1.59   0.00   1.80  49.97  51.10   1306   24.580    16   21.715   46.294 Allocation Failure   No GC
         7331.5   0.00   3.01  74.93  49.98  51.10   1307   24.590    16   21.715   46.304 GCLocker Initiated GC No GC
         7341.5   0.00  10.48  53.57  49.99  51.10   1309   24.610    16   21.715   46.324 Allocation Failure   No GC
         7351.5   1.77   0.00   0.00  46.84  51.10   1312   26.213    19   22.106   48.319 Allocation Failure   Allocation Failure
         7361.5   0.00   0.00  78.68  46.84  51.01   1314   26.231    24   32.129   58.360 Allocation Failure   No GC
         7371.5   4.46   0.00  62.98  46.84  51.04   1316   26.255    24   32.129   58.384 Allocation Failure   No GC
```

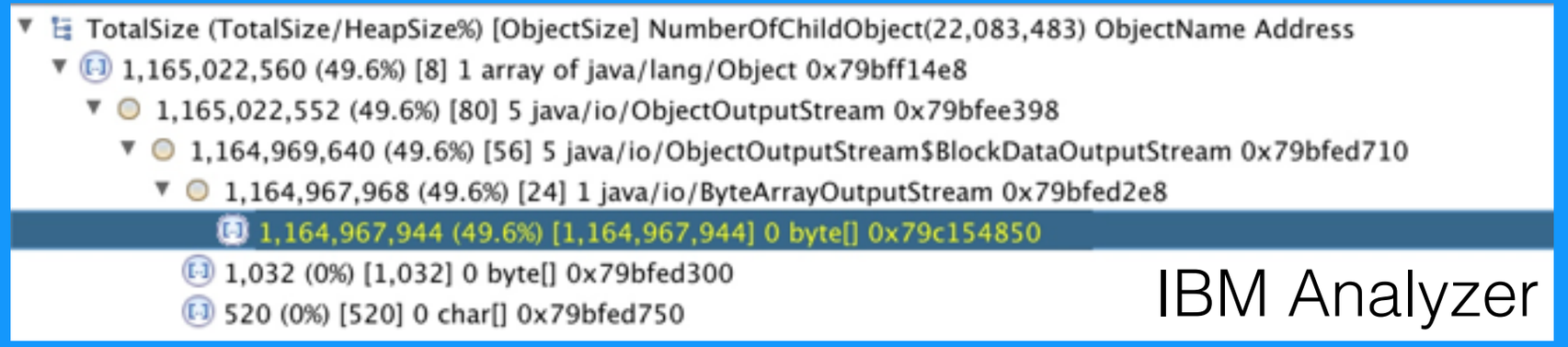
jstack

```
"IpNoticeWorker" daemon prio=10 tid=0x00007f2a314a2800 nid=0x512 runnable [0x00007f2a1b993000]
   java.lang.Thread.State: RUNNABLE
        at cn.fraudmetrix.forseti.api.worker.IpNoticeWorker.run(IpNoticeWorker.java:83)
        at java.lang.Thread.run(Thread.java:744)
```

MAT



IBM Analyzer

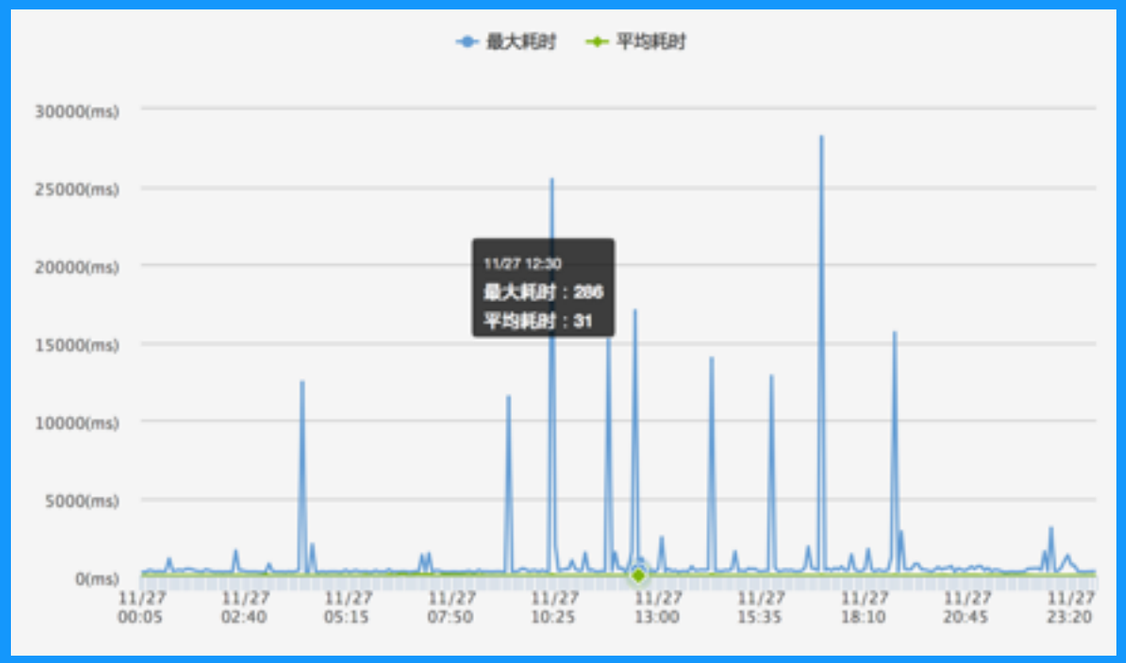


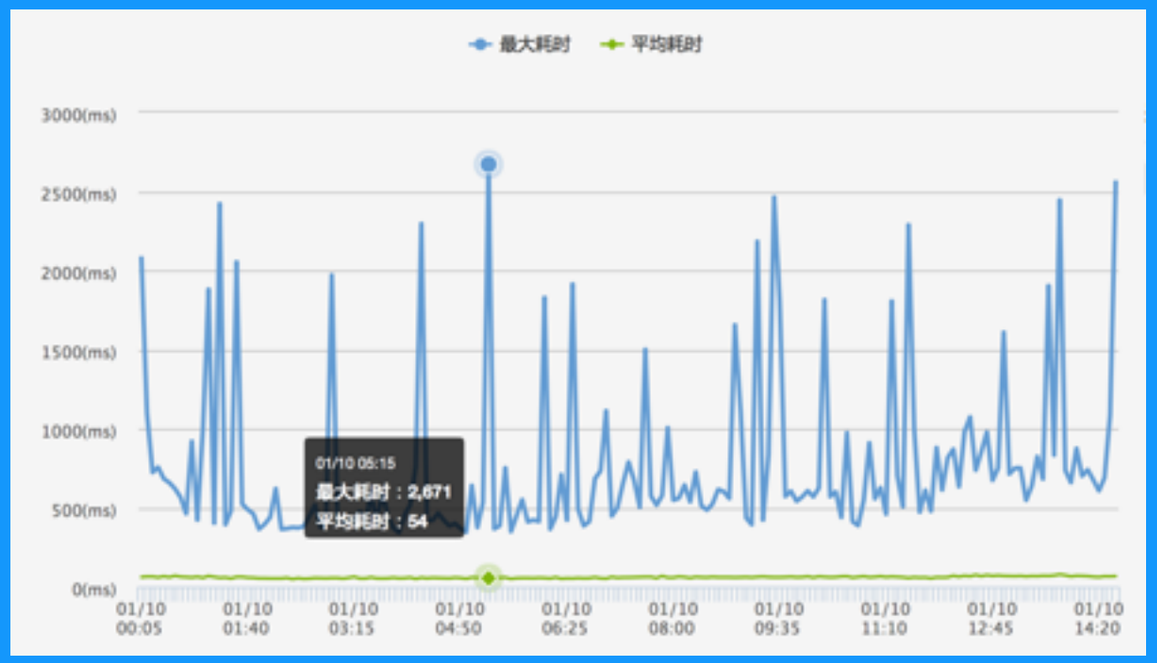
BTrace

```
@BTrace
public class TraceRequestTime {

    @OnMethod(clazz = "org.apache.catalina.valves.AccessLogValve", method = "log", location = @Location(value = Kind.ENTRY))
    public static void executeMonitor(@Self Object self,AnyType reqeust,AnyType response,long time) {
        println(strcat("spend time: ",str(time)));
        //jstack();
    }
}
```

API性能抖动的优化






```
<!-- 生成日志文件 -->
<appender name="FILE" class="ch.qos.logback.core.rolling.RollingFileAppender">
    <!--日志文件输出的文件名 -->
    <file>${app.output}/logs/app.log</file>

    <!-- 固定数量的日志文件，防止将磁盘占满 -->
    <rollingPolicy class="ch.qos.logback.core.rolling.FixedWindowRollingPolicy">
        <fileNamePattern>${app.output}/logs/app.%i.log.zip
        </fileNamePattern>
        <minIndex>1</minIndex>
        <maxIndex>10</maxIndex>
    </rollingPolicy>
```

Nginx的优化

- 使用Nginx组件ngx_http_dyups_module，动态修改upstream server，发布时自动将服务器摘除
- Nginx增加统一生成SeqID的Lua脚本，便于进行问题排查
- 禁用读取upstream server超时重试机制
- 设置upstream server读取超时为1s
- 打印upstream server调用时间

监控报警的完善

ZABBIX

- 硬件层面：CPU Load、Memory、Disk、IO等
- 应用层面：应用是否存活、JMX自定义扩展
- JVM层面：Heap使用情况、GC情况
- 其它：Nginx、Memcached、MySQL等专用插件

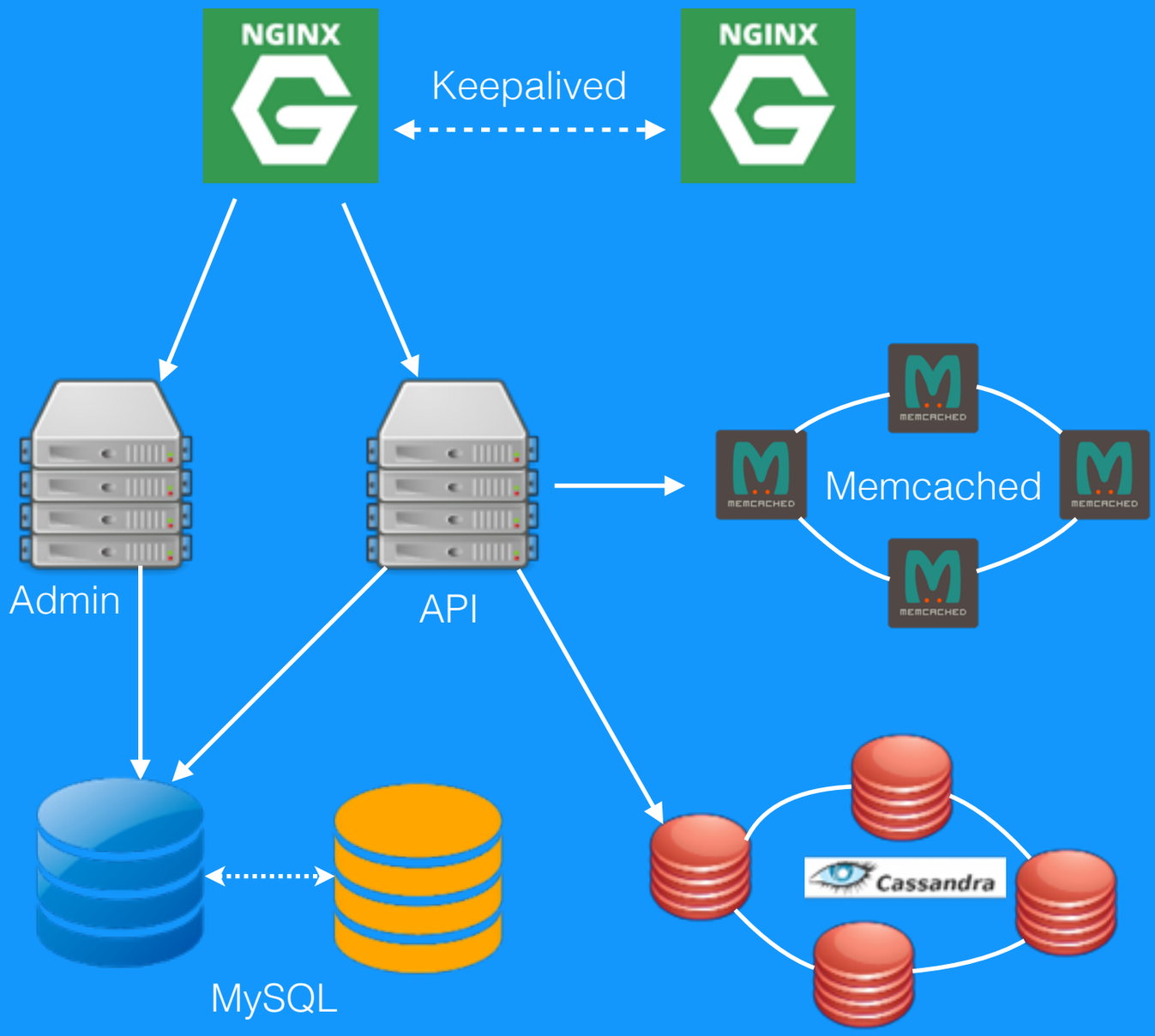
360 网站服务监控

- 监控所有对外服务的API接口、DNS、IP地址等

灾备和应急预案

- 阿里云部署简化版的服务
- Nginx使用Lua脚本做全局开关
- 应用做针对某个用户的开关
- 针对外部系统调用的开关
- 开关必须一键执行
- 全部进行实际演练

V2.1的架构



Zabbix Server

特点：

1.引入了NoSQL Cassandra

2.解决了事件数据单表过大无法存储的问题

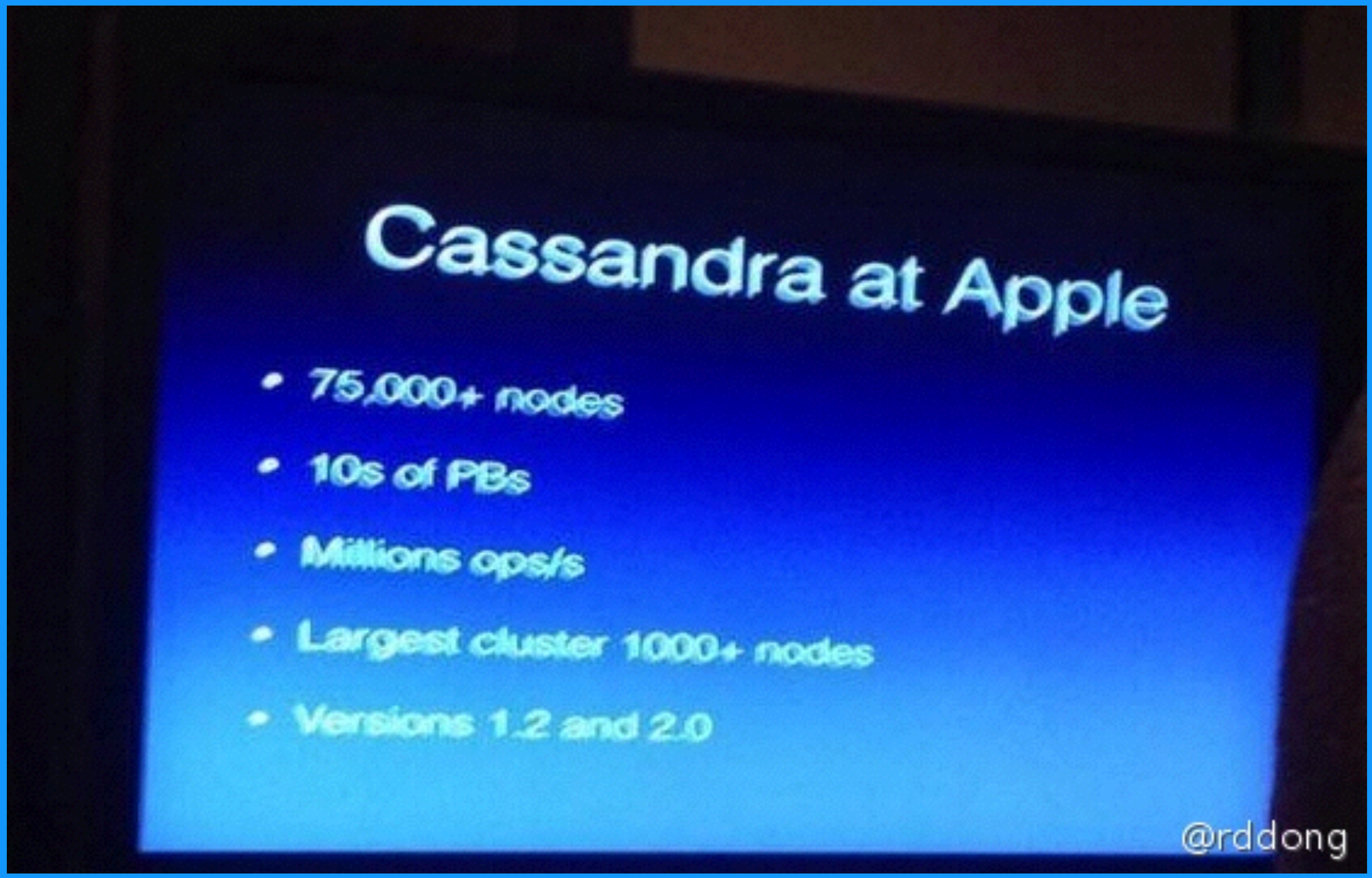
3.解决了Velocity数据单表过大和自动过期的问题

引入Cassandra

- 数据增长过快，分表已无法解决问题
- MySQL分库方案比较复杂，没有中间件支持
- 写入速度和读取速度（数据量不是特别大）非常快
- Cassandra P2P结构，增加节点非常简单
- 支持多数据中心，方便离线数据处理
- 宽行和数据自动过期，合适我们的应用场景
- 国外广泛使用社区活跃，有专门的公司支持

DB-ENGINES

Rank		Last Month	DBMS	Database Model	Score	Changes
1.		1.	Oracle	Relational DBMS	1439.16	-20.63
2.		2.	MySQL	Relational DBMS	1277.51	+8.93
3.		3.	Microsoft SQL Server	Relational DBMS	1198.61	-1.44
4.		4.	PostgreSQL	Relational DBMS	254.49	+0.48
5.		5.	MongoDB	Document store	250.90	+4.38
6.		6.	DB2	Relational DBMS	200.13	-10.12
7.		7.	Microsoft Access	Relational DBMS	139.14	-0.76
8.	↑	9.	Cassandra	Wide column store	98.75	+4.69
9.	↓	8.	SQLite	Relational DBMS	96.20	+1.49
10.		10.	Redis	Key-value store	94.24	+6.36
11.		11.	Sybase ASE	Relational DBMS	83.78	-2.21
12.		12.	Solr	Search engine	76.74	-1.68
13.		13.	Teradata	Relational DBMS	67.05	-0.35
14.	↑	15.	HBase	Wide column store	53.59	+2.51
15.	↓	14.	FileMaker	Relational DBMS	51.69	-0.53


Cassandra at Apple
• 75,000+ nodes
• 10s of PBs
• Millions ops/s
• Largest cluster 1000+ nodes
• Versions 1.2 and 2.0
@rddong

使用场景示例

```
create table velocity{
  attribute text,
  type text,
  partner text,
  data text,
  primary key ((attribute,type),partner)
};
```

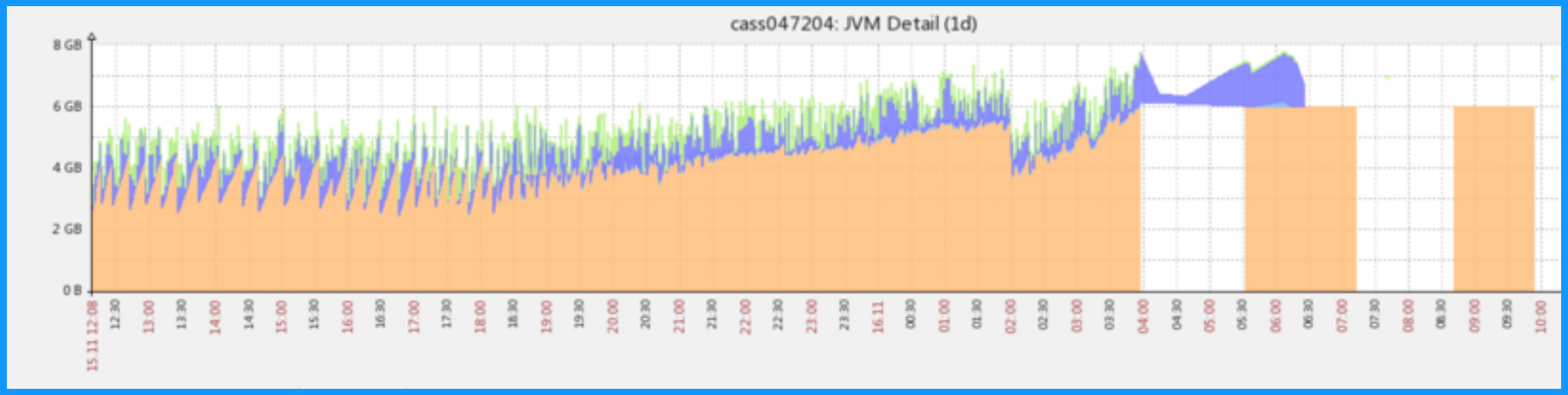
attribute	type	partner	data
117.165.48.235	ip	A	{a…}
117.165.48.235	ip	B	{b…}
182.85.85.80	ip	C	{c…}
117.165.48.235	ip	D	{d…}

	A:data	B:data	D:data
117.165.48.235:ip	{a…}	{b…}	{c…}

	C:data
182.85.85.80:ip	{c…}

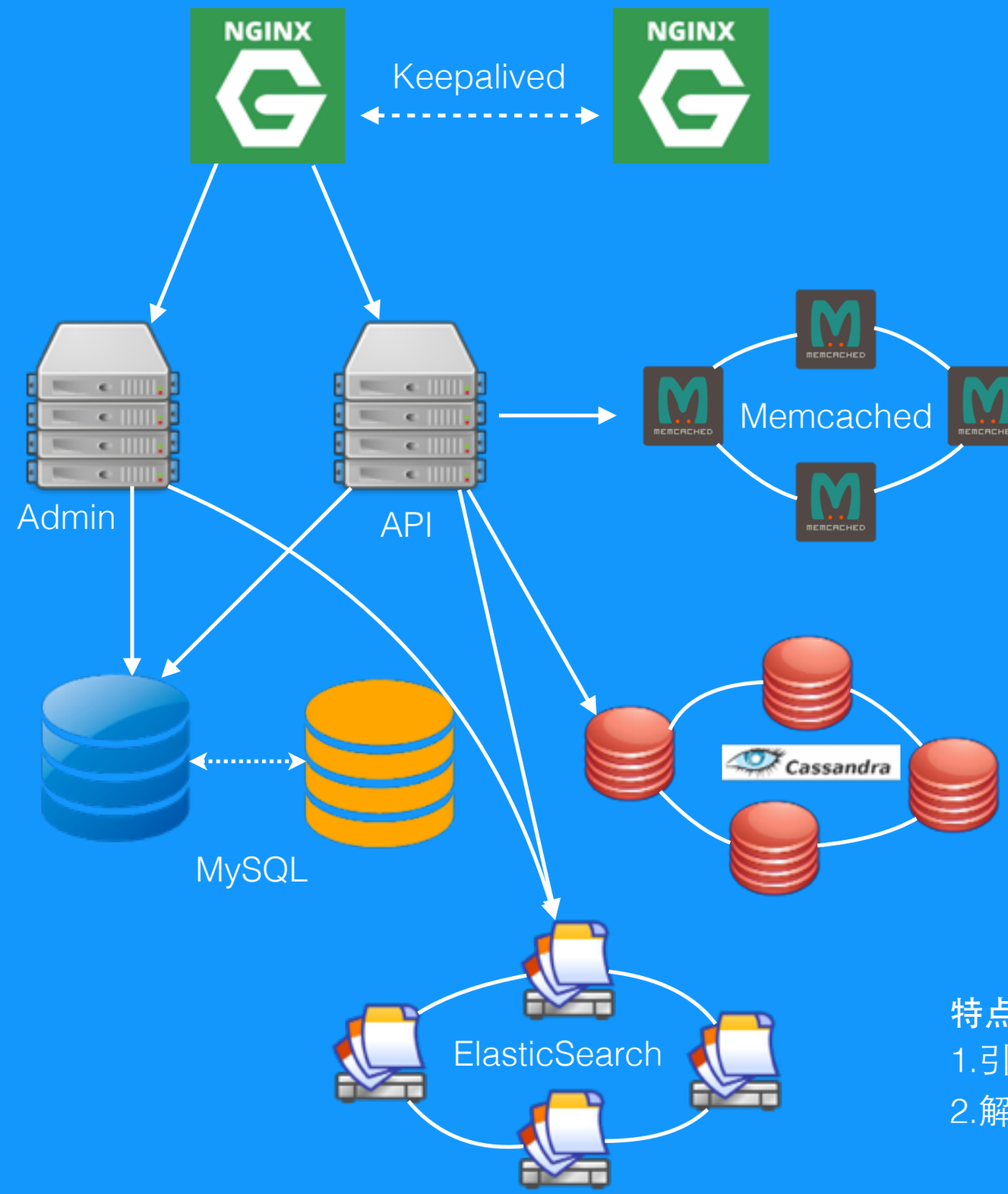
Cassandra遇到的问题

- LeveledCompactionStrategy导致的Compaction过于频繁，FGC时间过长，容易当机



- 没有热点数据时，Row Cache效果不明显，慎用
- 尽量避免Update操作，会导致大量tombstone
- DataStax Java Driver，避免重复创建Session

V2.1的架构



Zabbix Server

特点:

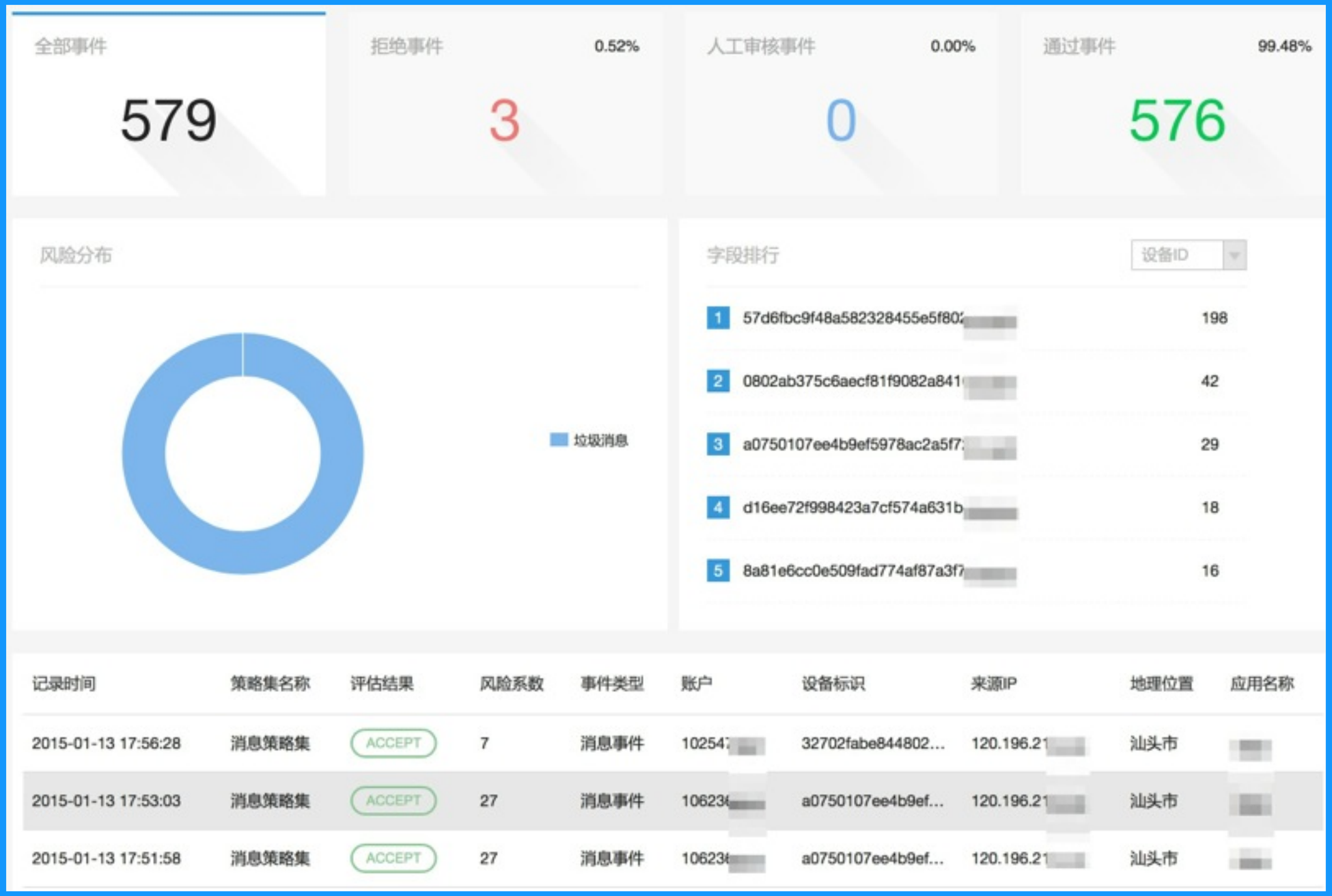
1.引入了ElasticSearch

2.解决了根据任意维度进行数据查询和分析的问题

引入ElasticSearch

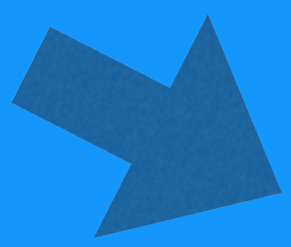
- 数据量过大，在MySQL中查询很慢
- 用户要求可以用任何字段进行统计分析，MySQL无法很好的支持
- Cassandra只适合Key/Value的查询
- ElasticSearch非常适合全字段索引查询
- 也是P2P结构，方便扩展

使用场景


全部事件
579
拒绝事件
0.52%
3
人工审核事件
0.00%
0
通过事件
99.48%
576
风险分布
垃圾消息
字段排行
设备ID
1 57d6fbc9f48a582328455e5f80
198
2 0802ab375c6aecf81f9082a841
42
3 a0750107ee4b9ef5978ac2a5f7
29
4 d16ee72f998423a7cf574a631b
18
5 8a81e6cc0e509fad774af87a3f7
16
记录时间 策略集名称 评估结果 风险系数 事件类型 账户 设备标识 来源IP 地理位置 应用名称
2015-01-13 17:56:28 消息策略集 ACCEPT 7 消息事件 32702fabe844802... 120.196.21 汕头市
2015-01-13 17:53:03 消息策略集 ACCEPT 27 消息事件 a0750107ee4b9ef... 120.196.21 汕头市
2015-01-13 17:51:58 消息策略集 ACCEPT 27 消息事件 a0750107ee4b9ef... 120.196.21 汕头市

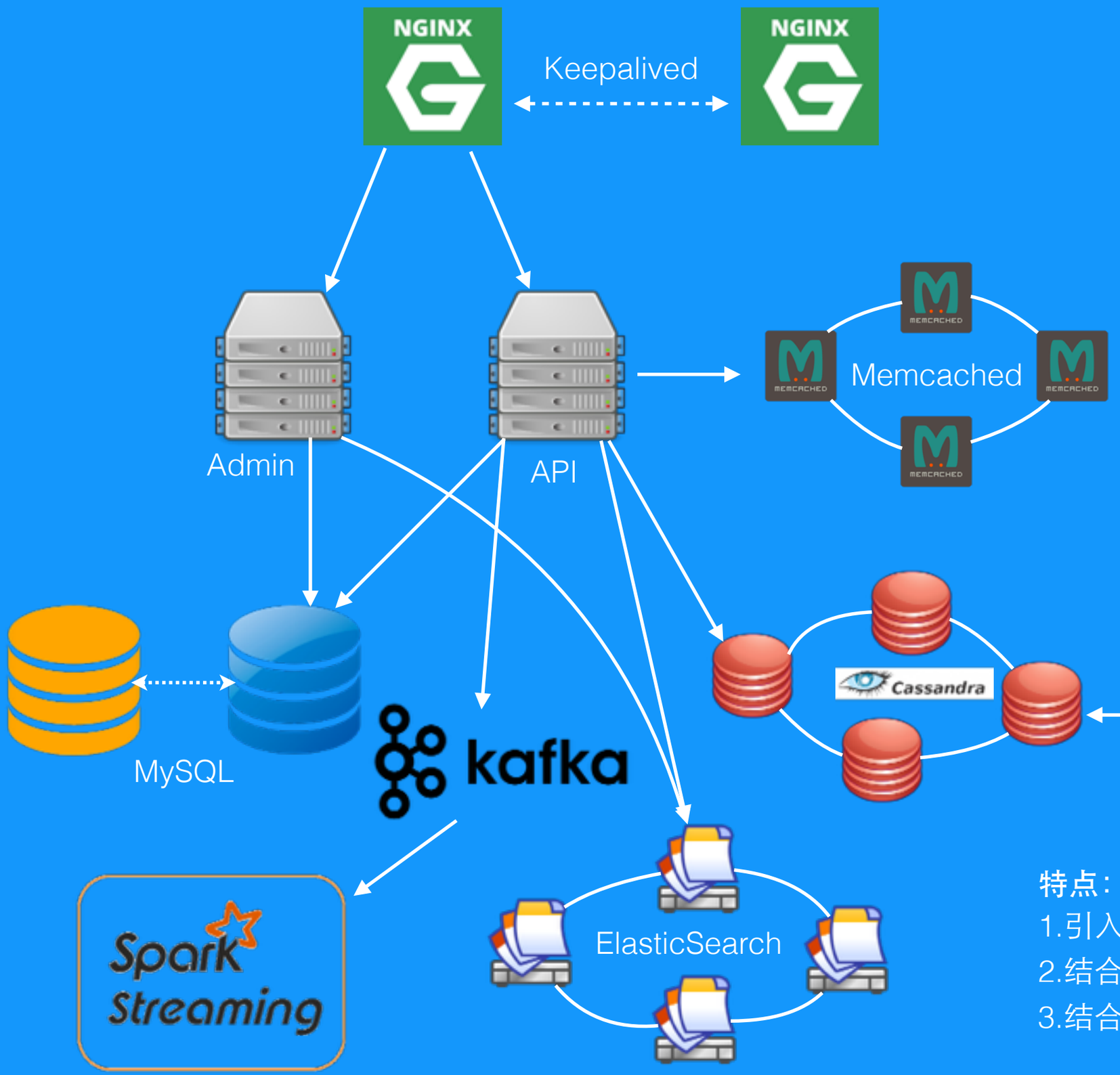
ElasticSearch遇到的问题

- 默认单库，加载所有数据
- 聚合查询容易导致内存溢出
- 单点故障



- 减少搜索线程量（threadpool.search.size）
- 按时间片分库（每周一个库）
- 部署多个节点（4个节点）

V2.3的架构



Zabbix Server

特点：

1.引入了Spark

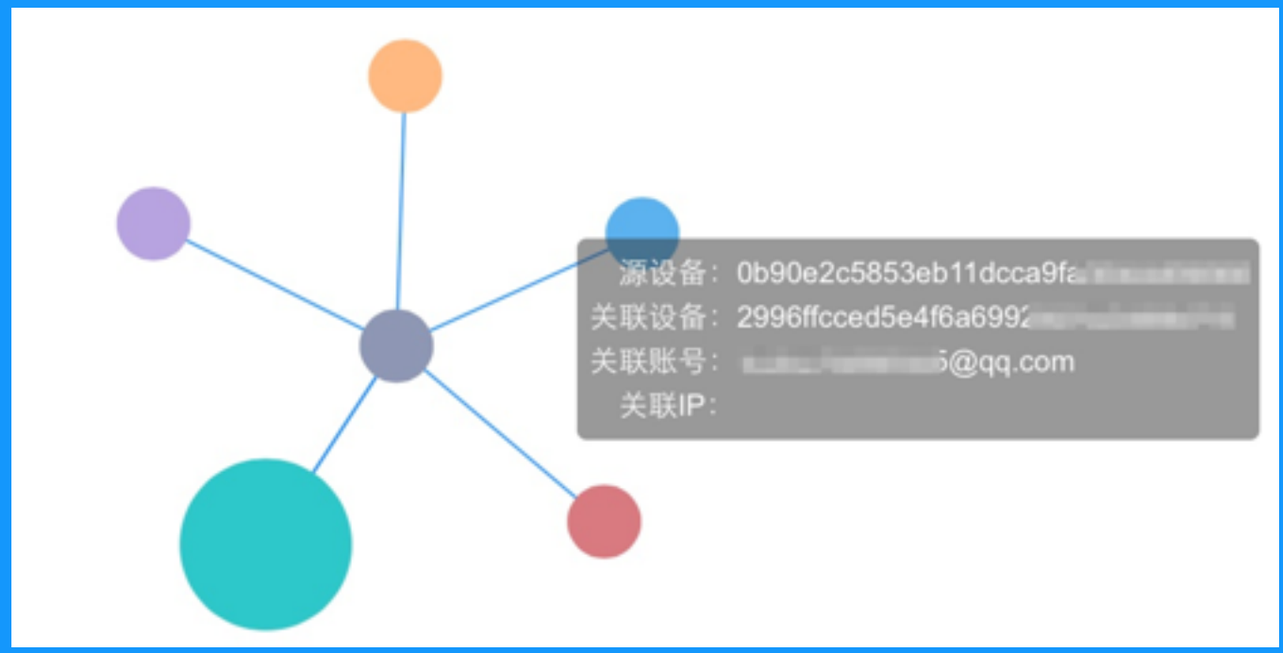
2.结合Cassandra进行离线数据分析

3.结合Kafka使用Streaming进行流计算

使用场景

同一帐号使用多个IP排行

关联设备拓扑图



使用Spark遇到的问题

- 大批量读取Cassandra性能较差
- 共用一个C*集群，影响线上环境
- Streaming只能做到准实时的计算

- 数据放置到HDFS
- 建立第二数据中心
- 应用到实时性要求不是太高的地方

总结

- 使用熟悉、成熟和社区活跃的开源技术
- 先满足业务需求，再优化，逐步演进
- 监控报警、应急预案应该成为系统功能的一部分
- 纸上得来终觉浅，绝知此事要躬行

欢迎交流

- Email：xinbo.zhang@fraudmetrix.cn
- 主页：http://yikebocai.com

Thanks!