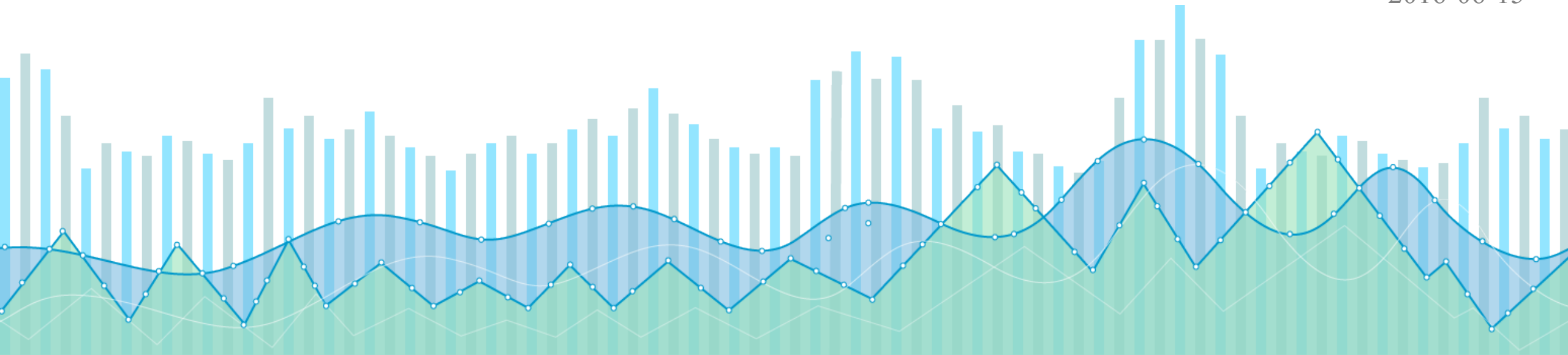
SUNING
苏宁云商

ES在苏宁海量日志平台的实践

彭燕卿
苏宁云商云事业部技术副总监
2016-06-15



个人简介

彭燕卿

- 2009年加入苏宁
- 目前任苏宁云商云事业部监控研发中心技术副总监
- 先后从事了SOA系统开发、苏宁易购、易付宝等大型网站的性能分析调优
- 目前主要从事苏宁监控系统架构及技术管理工作,主要专注于APM以及elasticsearch等实时计算领域。

Agenda

- ES在苏宁实时日志平台使用现状
- 实时日志平台架构演变
- ES优化经验总结
- ES运维经验分享
- Kibana4二次开发

概要

海量日志分析一直是数据分析领域非常火热的话题。苏宁海量日志分析平台经历了从1.0到2.0的演化，参与418、618等大促销活动等大促销活动，确保了99.99%的可用性和稳定性。

日志分析平台的核心基于EFK,通过为不同的日志类型定义格式和合并的规则，使用成熟可靠的数据搜集框架，基于弹性搜索技术从海量日志数据中挖掘出业务的关键指标，以及业务的异常数据，然后进行多维度的聚合统计，为用户最终定位和解决问题提供关键的价值。

苏宁实时日志平台使用现状-集群规模

- **43**个数据节点，**2TB**内存 for JVM
- 接入苏宁近2000个系统的应用日志、web访问、缓存、应用防火墙等日志
- 每天新索引8**T+**数据，每天doc数**超过80+亿条**
- **3个**不同的子集群
- 500+索引、53T+、800亿+数据、8000+ shard、7天存储
- 峰值40W/s数据写入

苏宁实时日志平台使用现状-业务场景

业务简介

- 承担了苏宁所有系统的应用、中间件、web、缓存、应用防火墙等日志的集中化准实时检索、统计分析，用于业务发生异常时及时定位问题，深度挖掘日志的大数据价值。

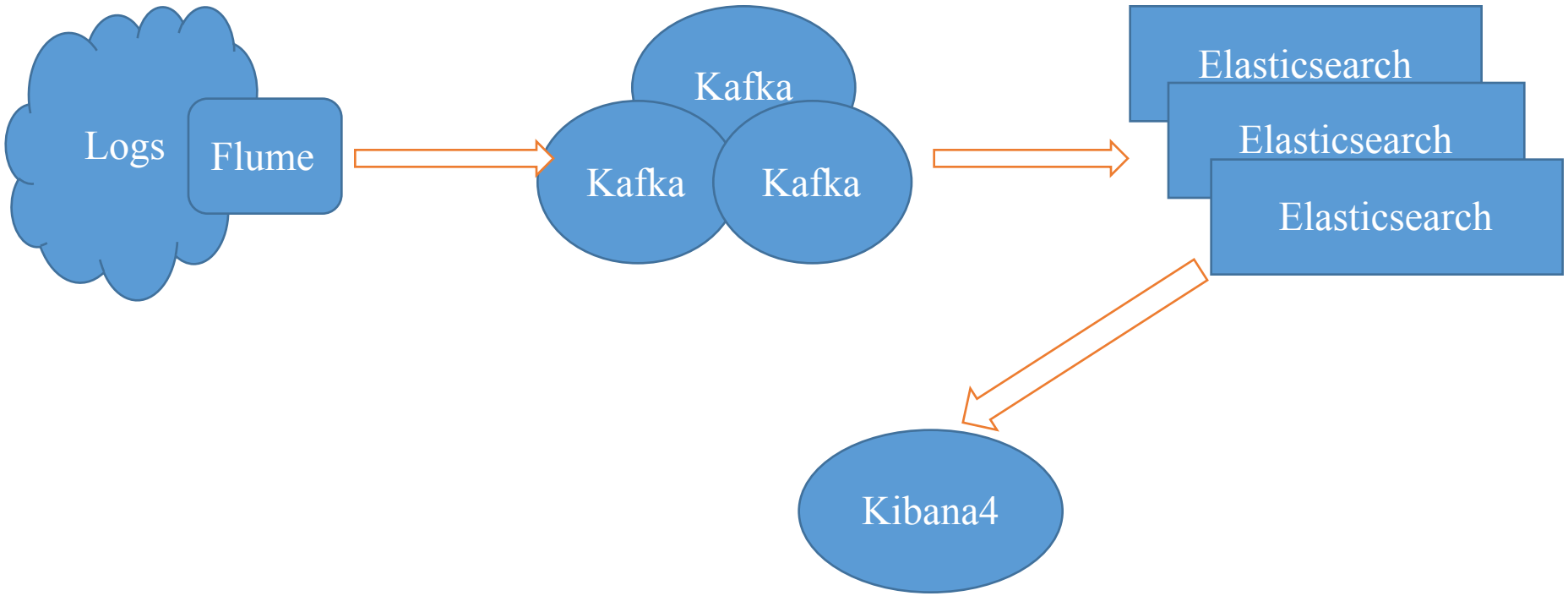
主要挑战

- 数据量大，每天8TB+数据，24小时不间断索引高并发，峰值时QPS 100以上
- 用户查询的时间跨度大，并且要求平均响应时间为秒级

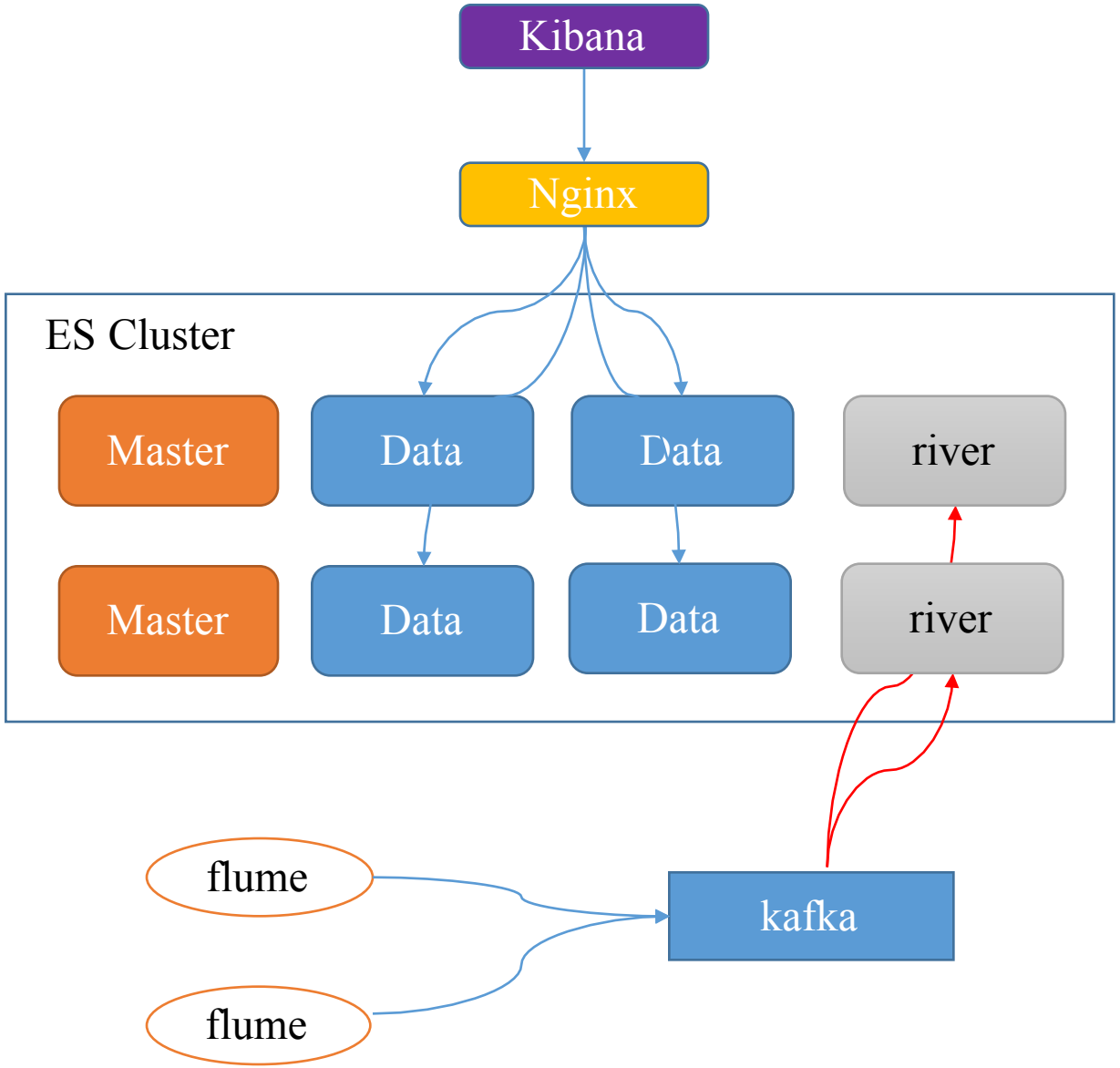
实时日志平台部署架构

实时日志平台采用的是flume+Kafka+Elasticsearch+Kibana的部署架构。

与传统的ELK架构不同，苏宁使用使用Flume实时日志采集系统收集日志，并且引入了Kafka消息队列，将flume采集到的日志存入kafka，然后通过Elasticsearch river插件消费Kafka里面的数据，将Kafka中的数据清洗过滤后，index到Elasticsearch集群中。



实时日志平台架构演变①



主要配置：

- ES集群中所有节点使用虚拟机
- 索引按照天生成
- 数据节点负责数据的索引
- 通过Nginx将用户的检索请求负载在数据节点

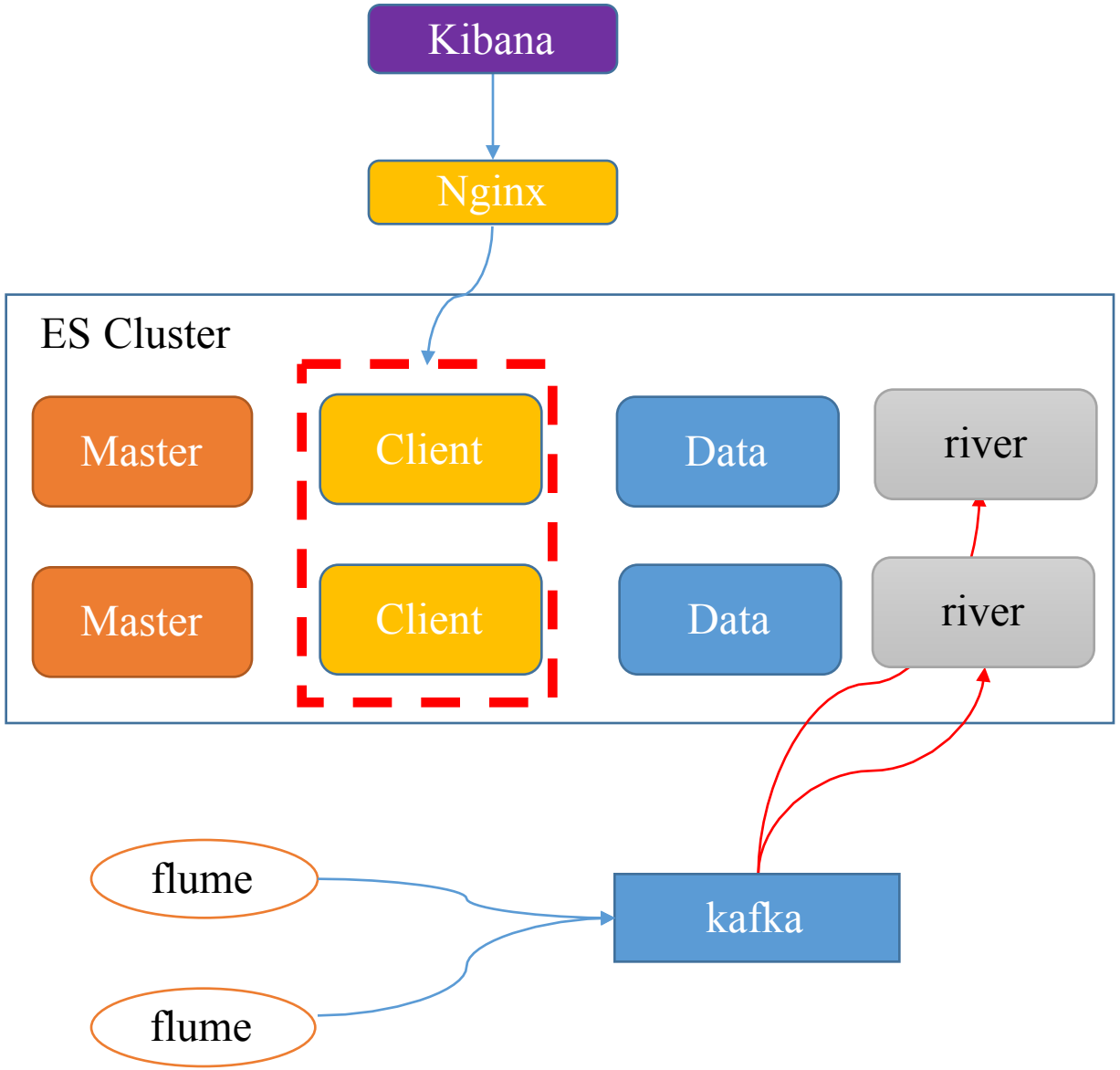
运行状况：

- 部分数据节点负载非常高
- 索引速度非常慢，经常出现Kafka堵塞情况
- 查询响应非常慢
- 并发访问数为10左右时，ES已经不能支撑

原因分析：

- 部分虚拟机在同一个物理机上，数据节点间的资源竞争非常激烈
- 数据节点同时承担索引和检索，负荷重
- 索引时有非必须的字段被索引（比如_all）
- 按天生成的索引太大
- 单索引

实时日志平台架构演变②



主要优化：

- 引入client节点，将检索的请求从数据节点转移到client节点。同时client节点通过river从kafka中拉数据将数据导入ES
- 移除在同一物理机上的虚拟机，并增加了一批物理机
- 关闭_all字段，
- 并且索引按照小时生成
- 根据日志类型划分不同的索引文件

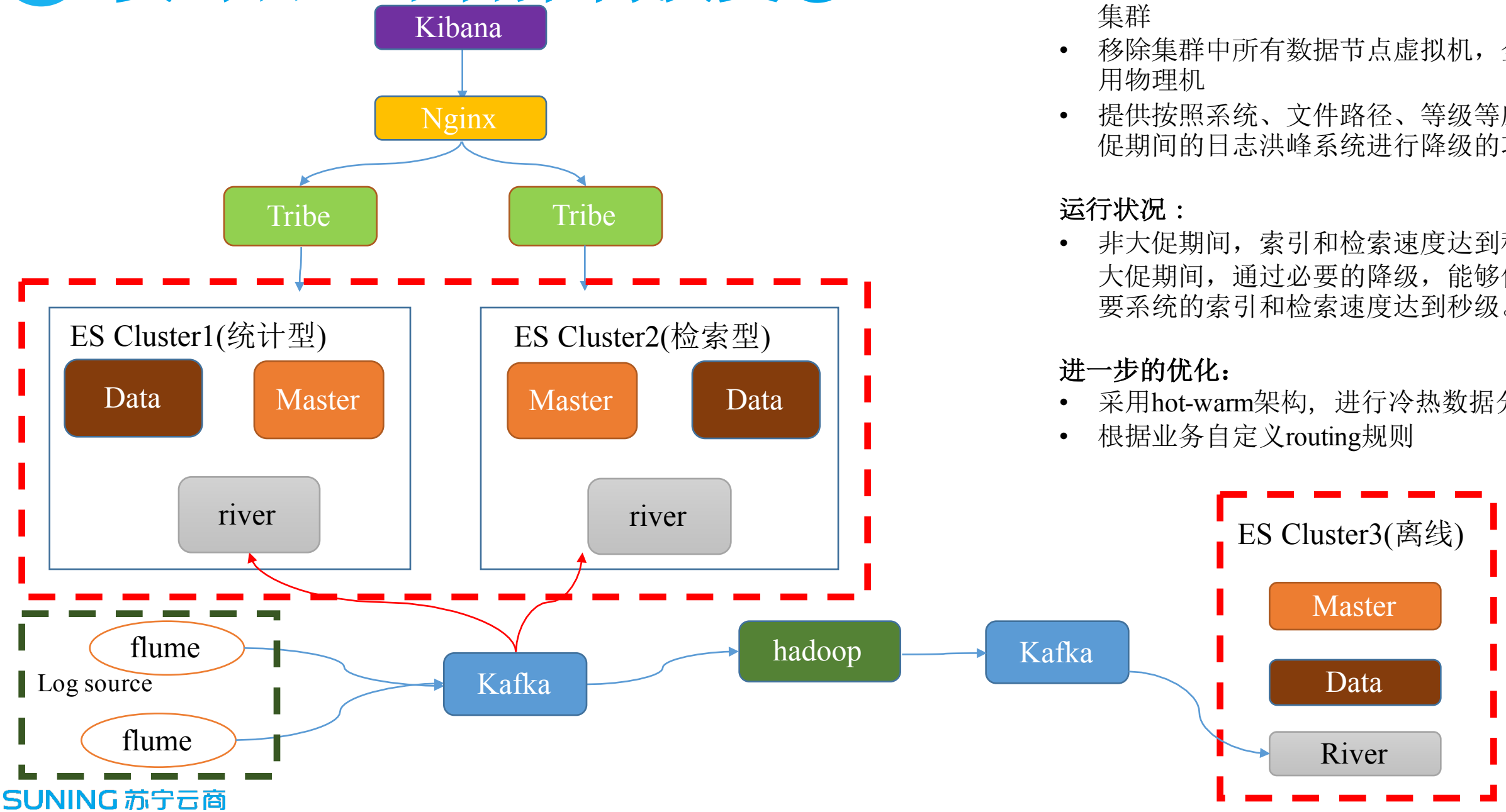
运行状况：

- 非大促期间，索引和检索速度基本能达到秒级，但是大促期间，出现日志量膨胀以及访问人数过多时，索引和检索速度很慢
- 随着集群规模的增加，运维的工作量也急剧增加

原因分析：

- 多种类型的日志（例如偏统计分析的web日志，偏检索的应用日志）混在一个大集群中,不同的日志间相互影响
- 缺少必要的降级功能

实时日志平台架构演变③



主要优化：

- 根据日志类型，将大集群拆分成不同的小集群
- 移除集群中所有数据节点虚拟机，全部使用物理机
- 提供按照系统、文件路径、等级等应对大促期间的日志洪峰系统进行降级的功能

运行状况：

- 非大促期间，索引和检索速度达到秒级。大促期间，通过必要的降级，能够保证重要系统的索引和检索速度达到秒级。

进一步的优化：

- 采用hot-warm架构，进行冷热数据分离
- 根据业务自定义routing规则

ES优化经验-注意问题

- 任何一个调优参数，都要经过全面的测试，不能直接在某个资料中看到某个参数就贸然运用到自己的生产环境里面。
- 每次只调优一个参数，不要多个参数一起调整。
- ES的调优可以分为硬件层面，OS层面，ES自身，进行调优时，这三个层面需要综合进行考虑。

ES优化经验-OS优化

- 适当调整OS的File Descriptors， 推荐使用64K。
- 禁用swap交换
- ulimit -l unlimited 锁内存

  bootstrap.mlockall: true

ES优化经验-索引速度优化

- 去掉_all字段可节省一半空间和提升索引速度
- 不分词的字符串字段设成not_analyzed
- 如果对于可靠性不要求100%的数据,可不设置副本
- 如果财大气粗的话,最好上SSD

```
},
"path": { -
  "index": "not_analyzed",
  "ignore_above": 5000,
  "store": true,
  "fielddata": { -
    "format": "fst"
  },
  "type": "string"
},
```

ES优化经验-索引速度优化

- 设置合理的refresh时间

  index.refresh_interval: 300S

- 设置合理的flush间隔

  index.translog.flush_threshold_size: 4g

  index.translog.flush_threshold_ops: 50000

- 适当增加索引限制

  indices.store.throttle.max_bytes_per_sec: 60mb

- 适当提高bulk队列

  threadpool.bulk.queue_size: 1000

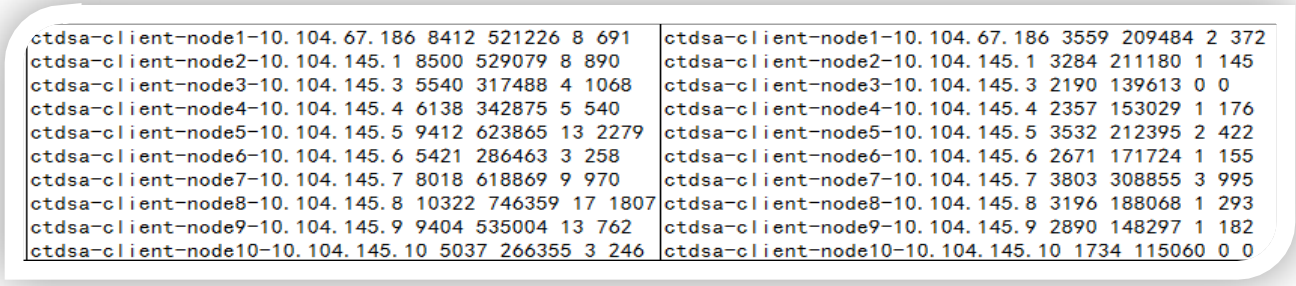
ES优化经验-稳定性优化

- 有时可能因为gc时间过长，导致该数据节点被主节点踢出集群的情况，导致集群出现不健康的状态，为了解决这样的问题，我们适当的调整ping参数。

discovery.zen.fd.ping_timeout: 40s
discovery.zen.fd.ping_interval: 5s
discovery.zen.fd.ping_retries: 5

- 调整所有client和数据节点的JVM新生代大小

数据节点young gc频繁,适当调转新生代大小（-Xmn3g），降低young gc的频率。



ctdsa-client-node1-10.104.67.186 8412 521226 8 691	ctdsa-client-node1-10.104.67.186 3559 209484 2 372
ctdsa-client-node2-10.104.145.1 8500 529079 8 890	ctdsa-client-node2-10.104.145.1 3284 211180 1 145
ctdsa-client-node3-10.104.145.3 5540 317488 4 1068	ctdsa-client-node3-10.104.145.3 2190 139613 0 0
ctdsa-client-node4-10.104.145.4 6138 342875 5 540	ctdsa-client-node4-10.104.145.4 2357 153029 1 176
ctdsa-client-node5-10.104.145.5 9412 623865 13 2279	ctdsa-client-node5-10.104.145.5 3532 212395 2 422
ctdsa-client-node6-10.104.145.6 5421 286463 3 258	ctdsa-client-node6-10.104.145.6 2671 171724 1 155
ctdsa-client-node7-10.104.145.7 8018 618869 9 970	ctdsa-client-node7-10.104.145.7 3803 308855 3 995
ctdsa-client-node8-10.104.145.8 10322 746359 17 1807	ctdsa-client-node8-10.104.145.8 3196 188068 1 293
ctdsa-client-node9-10.104.145.9 9404 535004 13 762	ctdsa-client-node9-10.104.145.9 2890 148297 1 182
ctdsa-client-node10-10.104.145.10 5037 266355 3 246	ctdsa-client-node10-10.104.145.10 1734 115060 0 0

ES优化经验-Filed Data

在进行检索和聚合操作时，ES会读取反向索引，并进行反向解析，然后进行排序，将结果保存在内存中。这个处理会消耗很多Heap，有必要进行限制， 不然会很容易出现OOM。

- Disabled analyzed field fielddata

- 限制Field Data的Heap Size的使用
  indices.fielddata.cache.size: 40%
  indices.breaker.fielddata.limit: 50%

ES运维经验-增加数据节点

- 技巧：

1.调整shard数

2.index.routing.allocation.total_shards_per_node: 2
index在每个node的shard数据
(如果后期需要移除节点,保证每个node有可分配的shard)

- 注意：

强行增加node，新索引数据会集中在新加的node上，导致新节点负荷高

ES运维经验-移除数据节点

移除node前可以先exclude要移除的node

- cluster：

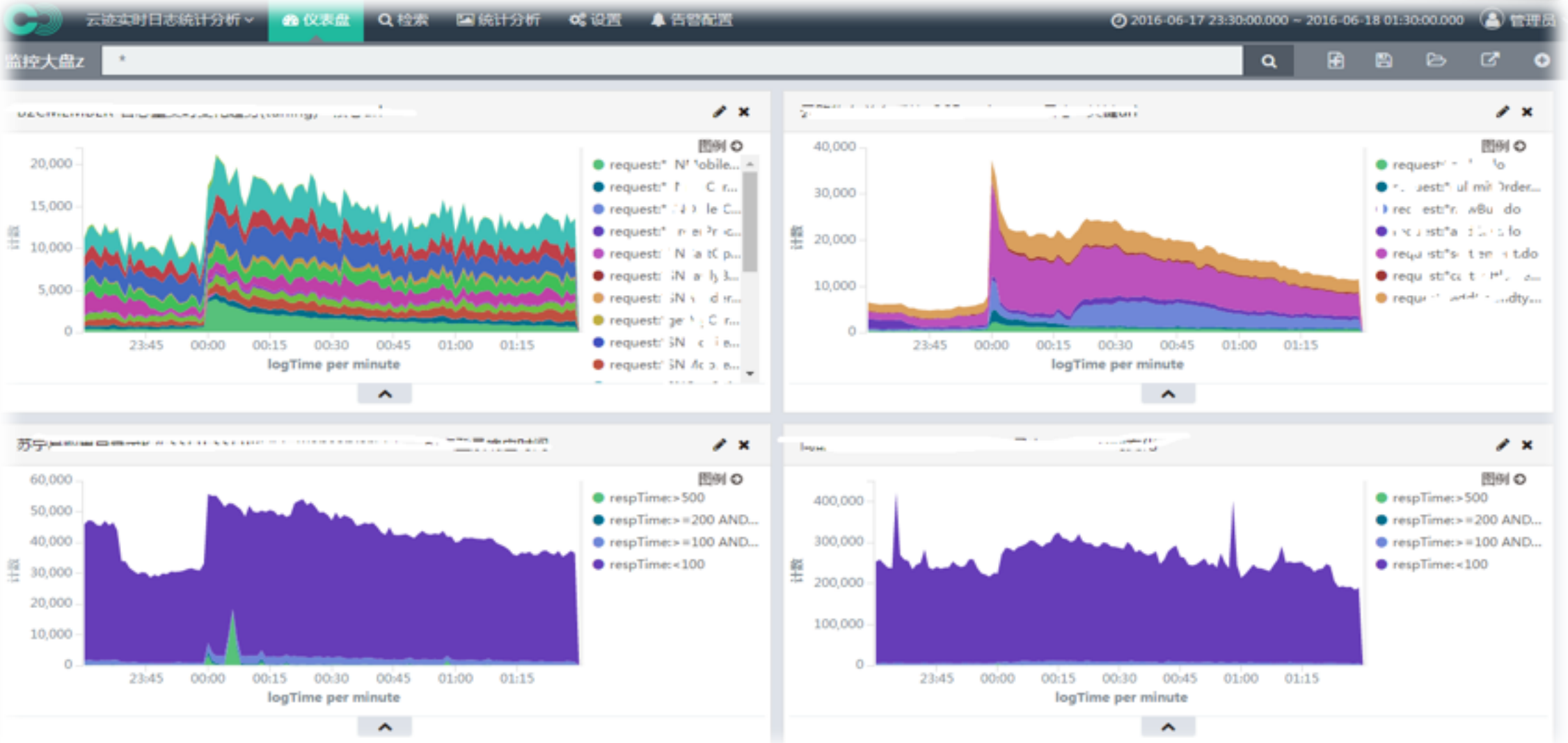
cluster.routing.allocation.exclude._name：node1

- **Index**：

index.routing.allocation.exclude._name: node1
index.routing.allocation.require.node_type: hot

技巧：以上参数可根据_ip、_host等来进行配置
以上参数可实现hot-warn

Kibana4二期开发-从汉化开始

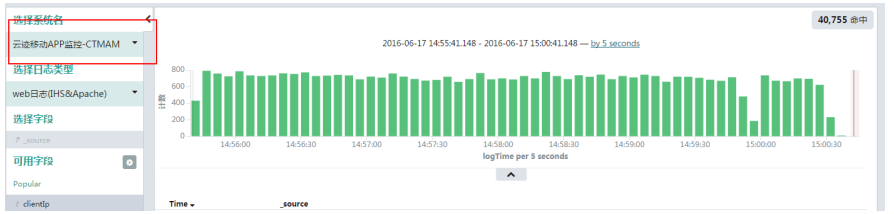


Kibana4二期开发-权限

实现cas单点登录

```
app.use('/', cas.bouncer,ctdsa_syslist.checkSysList);
```

不同用户只能查看授权的系统数据



Kibana和nodejs控制不同日志类型timepick

```
var time_limit_keys={
            "[dsa-appserver-]YYYY-MM-DD-HH":[7,2,"应用服务器日志"],
            "[dsa-appserver-]YYYY-MM-DD":[7,2,"应用服务器日志"],
            "[dsa-varnish-]YYYY-MM-DD-HH":[7,2,"varnish日志"],
            "[dsa-webserver-]YYYY-MM-DD-HH":[7,2,"web日志"],
            "[dsa-webserver-]YYYY-MM-DD":[7,2,"web日志"],
            "[dsa-net-]YYYY-MM-DD":[7,24,"网络设备日志"],
            "[dsa-storm-]YYYY-MM-DD":[7,24,"storm日志"],
            "[dsa-custom-]YYYY-MM-DD":[7,24,"移动终端日志"],
            "[dsa-firewall-]YYYY-MM-DD":[7,24,"应用防火墙日志"],
            "[dsa-nginx-]YYYY-MM-DD":[7,24,"nginx日志"],
            "[ctemm-]YYYY-MM":[90,720,"异常监管日志"]
};
```

禁用通过kibana访问_plugin、_shutdown

仪表盘、检索、统计分析和用户关联并每次加载上次打开的对象

Kibana4二期开发-细节优化

indexPattern映射为中文

禁用分词字段的统计分析和排除(误操作容易导致长GC甚至OOM,亲身经历),如果template中指定fileddata为disabled，进行统计分析或排序会报异常。

根据不同的数据类型默认展示不同的默认field

修改默认排序字段

```
//appserver日志和strom日志默认用seqNum升序 add by pengyq 20160603
if($scope.indexPattern.id.indexOf("dsa-appserver")>0 ||  $scope.indexPattern.id.indexOf("storm")>0){
  $state.sort[0]="seqNum";
  $state.sort[1]="asc";
}else{
  $state.sort[0]="logTime";
  $state.sort[1]="desc";
}
```

Kibana4二期开发-discover可查询更多

Kibana4二期开发-discover可查询更多

```
body= body.sort(sort[0],s1);
//设置最多查询100行
body = body.size(200);
//得到build后的query json body
var body_json = body.build();
body_json.query.filtered.query={query_string:state.query.query_string};

//增加辅助信息,参考原生discover的_msearch, Bodybuilder不支持
body_json.fields = ["*","_source"];
body_json.fielddata_fields =  ["createTime","logTime"];
body_json.highlight= {"pre_tags": ["@kibana-highlighted-field@"],"post_tags": ["@/kibana-highlighted-fiel
es.msearch({
  body:[{index:indexList},body_json]
}).then(function(resp){
  var objectArray1 = resp.responses[0].hits.hits;
  var objectArray1_Tmp = new Array();
  if(objectArray1.length>0){
     _.forEach(objectArray1,function(obj){
              obj[sort[0]] = obj._source[sort[0]]; //后面需要根据此字段进行一次升序排序
              objectArray1_Tmp.push(obj);
    });
    var previousLogs = _.sortBy(objectArray1_Tmp,[sort[0]]);//对数据统一进行一次升序排序,根据排序字段
    if(sort[1]== "desc"){//如果字段是降序,需要对array做一次降序(上一行代码升序了,倒排一次就行)
        previousLogs = previousLogs.reverse();
    }
    //在该数据组数据上合并$scope.rows
    var previousLogs_tmp = previousLogs.concat($scope.rows);

    var rows_length = $scope.rows.length;//用于对比是否有新数据增加

    //var rows_Tmp = $scope.rows.concat(objectArray1);
    //去掉_id重复的数据,但是可能会每次查询更多的数据不足200条
    $scope.rows = _.uniq(previousLogs_tmp,'_id');

    //如果合并完并且去重之后,长度一样, 说明没有数据新增,提示无更多日志
    if(rows_length == $scope.rows.length){
      toastr.warning('无更多日志!', '警告',{timeOut:1000});
```

Kibana4二期开发-日志上下文查询

Kibana4二期开发-日志上下文查询

- 这
顺
坑

```
app.controller('ApplogCtrl', ['$scope', 'DialogService', 'es', 'AlertService',
    function($scope, DialogService, es, AlertService) {
    var timeBounds = $scope.$root.timefilter.getBounds();
    var indexList =  $scope.indexPattern.toIndexList(timeBounds.min, timeBounds.max);
    //查询当前行前后的日志
    function queryLog(row, indexList) {
        var body1 = new Bodybuilder()
            //.query('query_string',"seqNum:<"+row.fields.seqNum[0])
            .filter('term','appId',row.fields.appId[0])
            .filter('term', 'ip', row.fields.ip[0])
            .filter('term', 'path', row.fields.path[0])
            .filter('range','sortNum',{le:row.fields.sortNum[0]})
            .sort('sortNum','desc')
            .size(101).build();//因为包含点击的日志行,所以需要多查询一行
        var body2 = new Bodybuilder()
            // .query('query_string',"seqNum:<"+row.fields.seqNum[0])
            .filter('term','appId',row.fields.appId[0])
            .filter('term', 'ip', row.fields.ip[0])
            .filter('term', 'path', row.fields.path[0])
            .filter('range','sortNum',{gt:row.fields.sortNum[0]})
            .sort('sortNum','asc')
            .size(100).build();

        es.msearch({body:
          [{index:indexList},body1,{index:indexList},body2]
        }).then(function(resp){
            var response = resp.responses;
            var objectArray1_source = new Array();
            //得到第一个对象的source
            if(!response[0].error && response[0].hits.hits && response[0].hits.hits.length){
                var objectArray1 = response[0].hits.hits;
                $scope.beforeSize =  objectArray1.length-1;
                _.forEach(objectArray1,function(obj){
                    var source = obj._source;
                    if(source.sortNum==undefined){
                        source.sortNum=new Date(source.logTime).getTime+""+source.lid;
```

题。

Kibana4二期开发-去掉_node请求

discover.js | index.html | Find Results | index.js | check_for_es.js | check_es_version.js | elasticsearch.js | elasticsearch.angular.js | visualize.js | visualize.html

```
var notify = new Notifier({
  location: 'Setup: Elasticsearch version check'
});

return notify.timed(function checkEsVersion() {
/*
  var SetupError = Private(require('components/setup/_setup_error'));

  return es.nodes.info()
  .then(function (info) {
    var badNodes = _.filter(info.nodes, function (node) {
      // remove client nodes (Logstash)
      var isClient = _.get(node, 'attributes.client');
      if (isClient != null && esBool(isClient) === true) {
        return false;
      }

      // remove nodes that are gte the min version
      var v = node.version.split('-')[0];
      return !versionmath.gte(minimumElasticsearchVersion, v);
    });

    if (!badNodes.length) return true;

    var badNodeNames = badNodes.map(function (node) {
      return 'Elasticsearch v' + node.version + ' @ ' + node.http_address + ' (' + node.ip + ')';
    });

    throw SetupError(
      'This version of Kibana requires Elasticsearch ' +
      minimumElasticsearchVersion + ' or higher on all nodes. ' +
      'I found the following incompatible nodes in your cluster: \n\n' +
      badNodeNames.join('\n')
    );
  });
*/
  return true;

});
```

509 ms

Next

解决跨天数据查询性能难题

集群hot-warn

自动识别异常Root Cause分析

智能告警

谢谢聆听

Q&A

pengyq@cnsuning.com

http://www.suning.com