搜索平台

# 引言

## 编写目的

* 创建搜索的统一处理平台，进行业务解耦
* 提高搜索服务的可靠性
* 提供底层搜索的优化，包括搜索精度，广度和性能
* 提供底层搜索组件的管理系统

## 背景

当前搜索开放的接口都是通过solr直接对外提供的。业务侵入性很强，耦合度高，所有业务处理的算法都是使用solr插件的方式完成的。同时，由于业务的需求特点，导致solr内部增大了死锁的可能性。更重要的是当前的搜索服务架构很难提高性能，同时对硬件资源浪费太多。

# 搜索平台

搜索平台需要满足的需求：

1、高可用，高并发，高吞吐量

1、业务解耦

负责业务的转换与搜素条件的调优

3、业务降级

4、查询接口向前兼容

5、计流限流

6、权限认证

7、负载均衡

8、熔断

## 总体架构



## 搜索平台架构

HDFS

SolrCloud/ES

搜索中间件

实时计算平台

搜索平台

API网关

业务转换

业务降级

检索调优

检索

更新

缓存(1)

Kong

Zuul

redis

缓存(3)

缓存(2)

管理系统

## 新旧系统对比

1、从设计上保证了搜索的效率

按业务，按客户分索引。索引边界明确，各个客户之间不会相互影响。

2、搜索平台与业务耦合度低

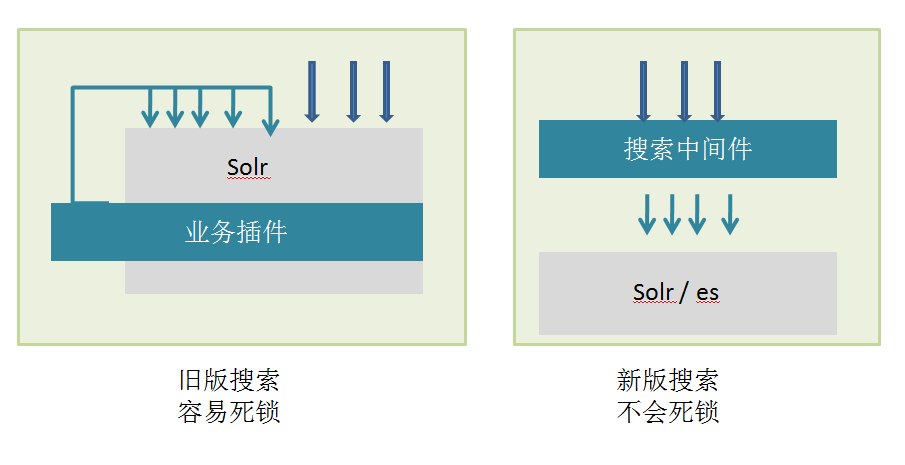
提供业务流计算插件，业务端很容易自定义搜索的实体doc。

3、底层搜索组件可以很灵活的更换和升级

通过搜索平台适配底层搜索组件，底层搜索组件可以灵活升级和更换。

4、新系统中的各模块更容易水平扩容

5、解决了旧系统中的死锁问题



6、提供索引管理平台

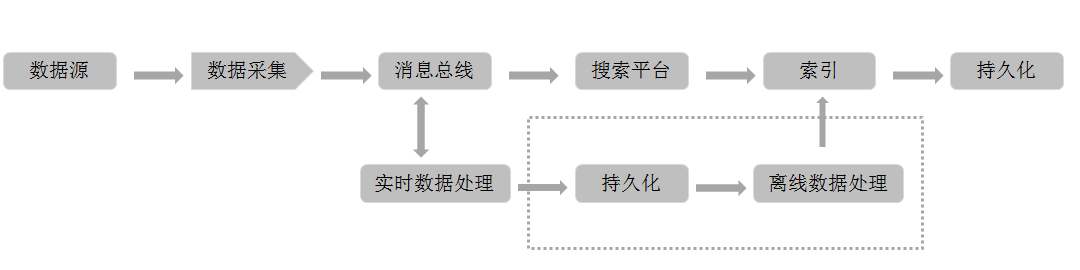
索引管理平台能实现自动管理客户的索引。根据监控组件反馈的数据，自动优化索引文件，自动分片等。

7、提供监控系统

将搜索平台的各层节点运行数据导入监控系统，能清晰发现和解决问题。

8、springcloud提供的微服务，与docker无缝对接，便于后续CI和CD。

## 数据流模型



为了提高搜索服务的高可用性，搜索平台同时设计了实时数据处理流程和离线数据处理流程两种。

### 实时数据处理流程

1、业务数据源提供slave数据库用于读取binlog；

2、数据采集模块负责将数据采集并发送给消息总线；

3、实时数据处理模块从消息总线中消费步骤2中消息，根据topic对应的业务特点对数据进行业务处理，生成最终符合搜索要求的doc，并将该doc发送给消息总线；

4、搜索平台消费消息总线中的doc，并生成索引存储到分布式文件系统。

### 离线数据处理流程

1、在实时数据处理流程中的第3步，将生成的doc同时持久化存储

2、当索引文件出现故障需要重建索引时，使用离线方式将上述doc生成索引并存储到分布式文件系统。

## 各模块技术选型

### 数据采集

选型：阿里Canal组件

目前搜索的数据源均为mysql数据库。通过读取数据库binlog的方式进行数据采集，可靠性高，不影响业务的数据库服务。

Canal是阿里内部用于多数据中心数据同步的组件。基于数据库增量日志解析，提供增量数据订阅&消费，目前主要支持了mysql。

### 消息总线

选型(1)：kafka

Kafka是一种高吞吐量的[分布式](https://baike.baidu.com/item/åå¸å¼)发布订阅消息系统。

* 通过O(1)的磁盘数据结构提供消息的持久化，这种结构对于即使数以TB的消息存储也能够保持长时间的稳定性能。
* 高吞吐量  ：即使是非常普通的硬件Kafka也可以支持每秒数百万的消息。
* 支持通过Kafka服务器和消费机集群来分区消息。
* 支持在线水平扩展。
* 不保证消息被成功消费，与storm结合使用可以保证。

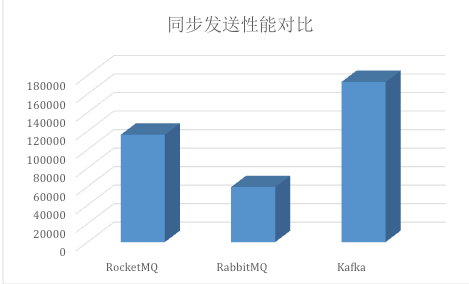
选型(2)：RabbitMQ

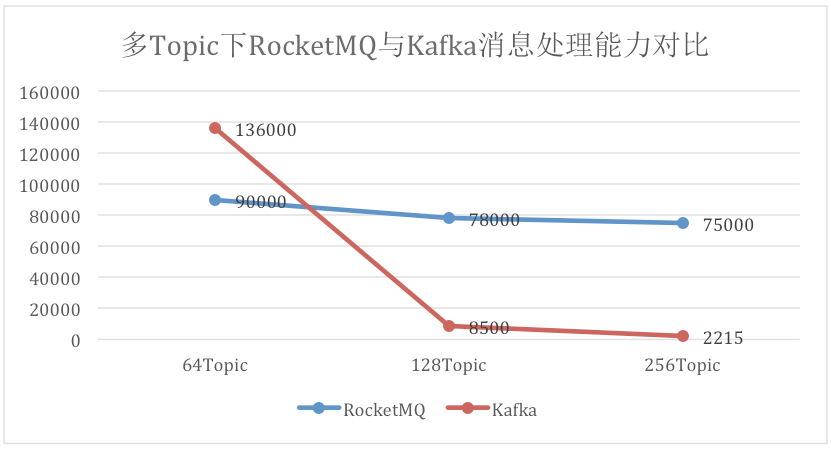
RabbitMQ是一个高可靠，安全的消息队列

选型(3)：RocketMQ

RocketMQ是淘宝自主研发的Notify消息中间件，使用Mysql作为消息存储媒介，可完全水平扩容

* 支持严格的消息顺序
* 支持Topic与Queue两种模式
* 亿级消息堆积能力
* 比较友好的分布式特性
* 同时支持Push与Pull方式消费消息





### 实时数据处理

选型：storm

Storm是一个免费开源、分布式、高容错的实时计算系统

### 搜索平台

选型：springcloud

## 环境配置

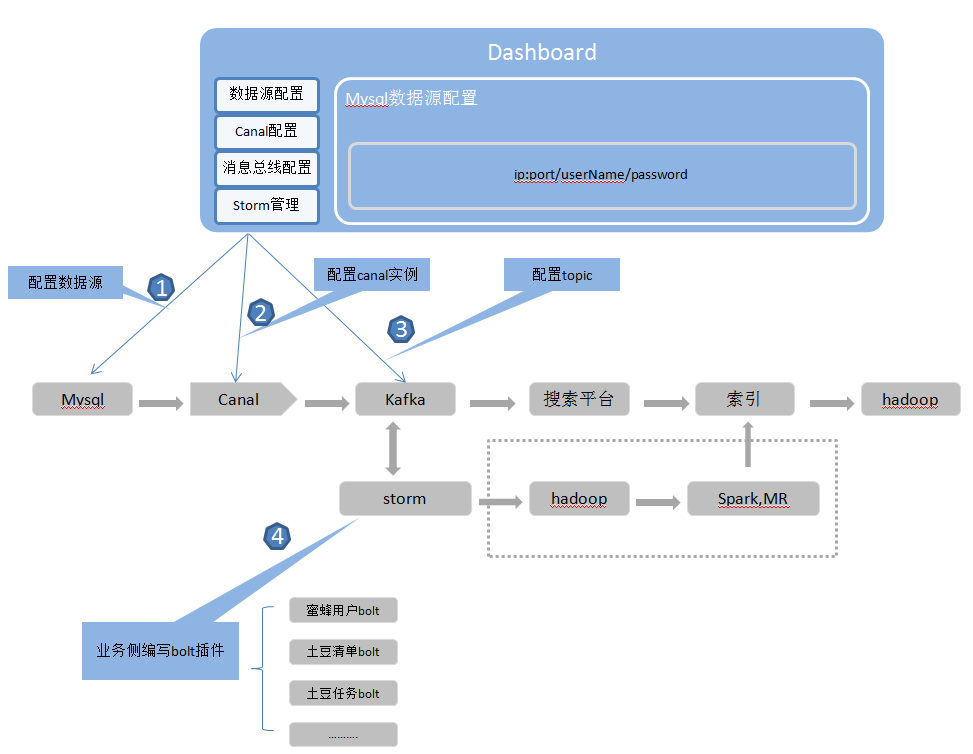
3台：8核cpu、8G、1TB

2台：8核cpu、16G、1TB

可以利用现有搜索服务器资源：192.168.88.56、192.168.88.26、192.168.88.27

## 业务接入

业务接入，采用插件的方式。如下图：



### 新业务接入(含升级)

每个业务根据自身的检索需求，编写业务插件。自测通过后，将插件上传到storm系统，并启用业务计算任务。

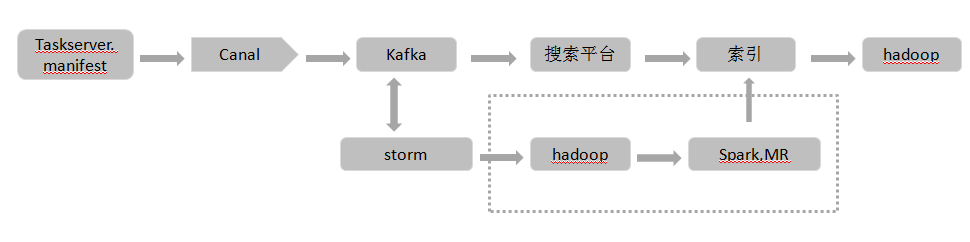
### 业务下架

从storm系统中将业务的计算任务删除。

## 样例说明

### 土豆清单搜索

土豆清单是土豆业务中的一部分，该业务数据存储在mysql数据库taskserver的表manifest中。将清单生成索引的过程如下：



1、表manifest中添加一个清单，信息结构为：



2、canal注册为taskserver的从库，从binlog中读取改条记录，并组装为消息发送给kafka，消息结构为：

|  |
| --- |
| taskserver.manifest:{"INSERT":[{"id":10808,"mf\_id":50002351,"mf\_name":"待处理任务","product\_type":50,"creator\_id":63683599}]} |

3、storm从kafka中消费消息，并根据业务计算出最终doc，结构如下：

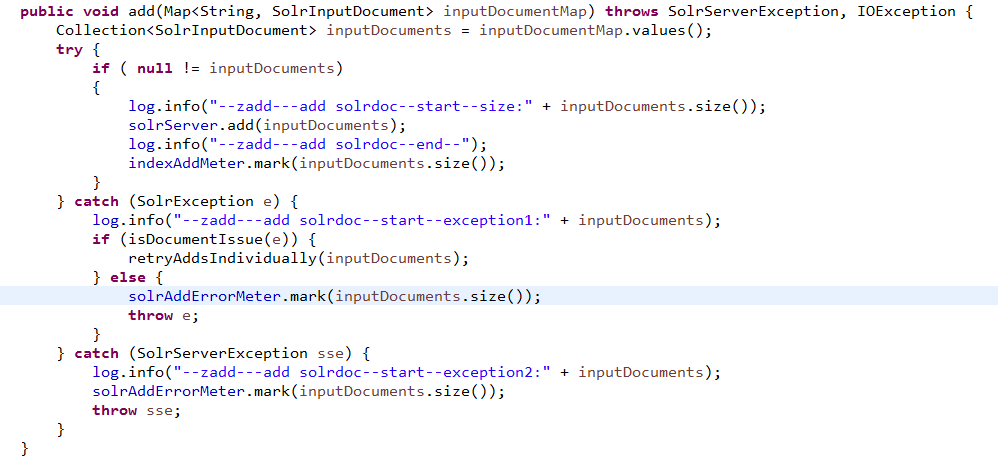
|  |
| --- |
| {  "solr\_docid" : "solr\_tudou\_manifest-10808",  "id" : 10808,  "mf\_id" : 50002351,  "mf\_name" : "待处理任务",  "product\_type" : 50,  "customercode" : "000092",  "member\_relation" : ["63683599"],  "status" : 1,  "creator\_id" : 63683599  } |

storm在将最终doc存入kafka，同时将该消息持久化到hadoop中。

storm处理业务逻辑代码样例如下：

|  |
| --- |
| @Override  public void execute( Tuple input, BasicOutputCollector collector ) {  // TODO Auto-generated method stub  String line = input.getString( 0 );  if ( line.startsWith( LogGroup.TASKSERVER ) ) {  Manifest manifest = JSON.fromJson(line, Manifest.class);  try {  if ( null != manifest.getCreatorId() ) {  Map<String, String> userBean = HBaseDBUtils.getData( HbaseTableMgr.SOLR\_ENTERPRISE\_USER, manifest.getCreatorId() );  data.put( "customercode", userBean.get( "customercode" ) );  }  String productType = data.get( "product\_type" );  if ( null != productType && "20".equals( productType.trim() ) ) {  data.put( "product\_type", "20" );  }  else {  data.put( "product\_type", "50" );  }  data.put( "indexDoctype\_ID", "92" );  HBaseDBUtils.putData( HbaseTableMgr.SOLR\_MANIFEST, mfId, data );  }  catch ( Exception e ) {  e.printStackTrace();  }  }  } |

4、搜索平台从kafka中消费消息，并生成索引。代码样例如下：



## API网关

当前作为搜索api的网关，运行稳定后，可以扩展为其他业务的网关。

需要实现的功能：

* 访问控制：黑白名单，访问次数，访问频率
* 协议适配
* 身份认证
* 权限管理
* 计流限流
* 负载均衡
* 调用链
* 熔断，降级
* 监控
* 统计
* 管理面板

## 搜索中间件

需要实现的功能：

* 业务解析与转换
* 搜索调优：效率、精度、广度
* 业务降级
* 检索结果聚合
* 适配下游搜索产品

搜索中间件

Scope=[1,2,3,…]

Solr

Cloud

OR

es

通讯录

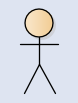
公告

消息

群聊

土豆

……



### 查询

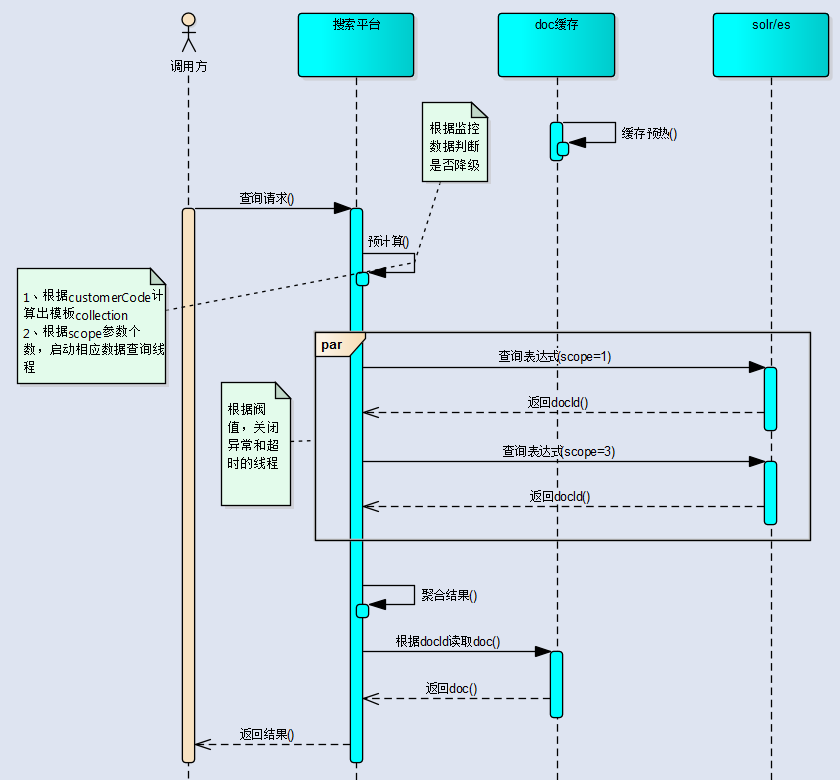
“调用方”发起查询请求时，“搜索平台”根据请求参数搜集如下两个方面的信息：

1、根据customercode参数，计算出目标collection；

2、根据scope参数的个数，启动相应数量的查询线程；

“搜索平台”根据上述两个信息，直接向指定的collection发起查询请求。查询请求并发执行。可根据相应阀值关闭查询异常或者响应慢的请求线程。待所有查询线程执行完成后，“搜索平台”将查询结果进行聚合。

查询流程如下图：

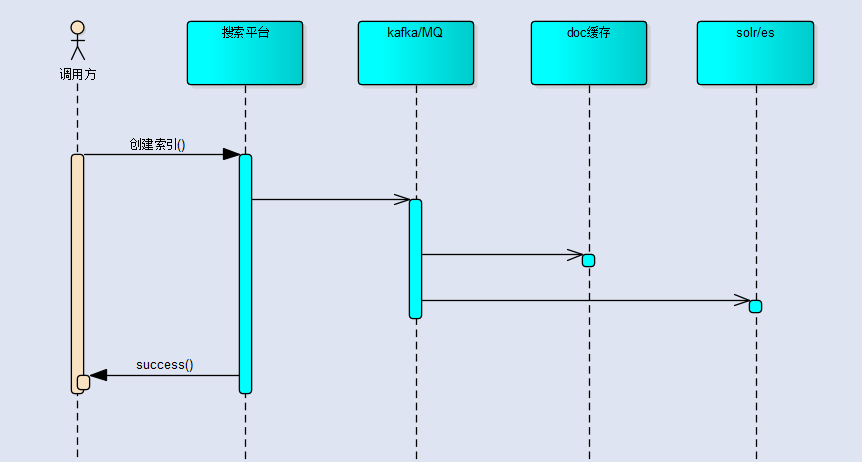


<说明> 查询表达式：

|  |
| --- |
| Query query = new Query();  query.setQuery("customercode:990003 AND text:test ");  query.addField("solr\_docid");  query.addSortField("time", Query.ORDER.asc);  query.set("qf","display\_name^2 last\_name^0.2");  query.setStart(0);  query.setRows(4); |

### 创建（修改）索引

搜索平台根据doc中的customercode信息，将索引创建到指定的collection中。



### 缓存

* doc缓存
* 客户数据缓存

#### 缓存预热

缓存系统在启动后，先将HDFS或者磁盘中的冷数据（doc）加载到缓存系统中。加载完成后，才能对外提供服务。

#### 热词缓存

根据“实时计算平台”反馈的客户搜索行为，将热点行为和数据加入“客户数据缓存”中，并根据客户行为频率进行更新客户数据缓存。

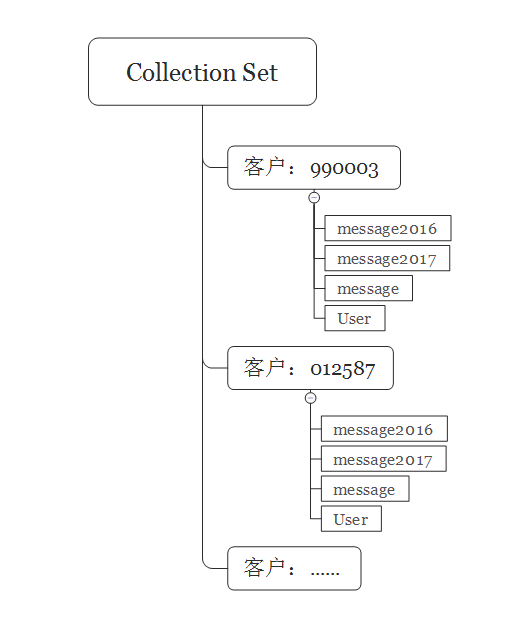
## 管理系统

需要实现的功能：

* 根据业务定期备份索引
* 索引Collection的管理（根据监控平台的反馈）

### Collection的管理原则

以客户编码为边界来管理索引空间



1、每个客户创建一个CollectionSet，用客户编码命名

2、每个客户的CollectionSet分为“消息索引”和“用户信息索引”

3、消息索引按时间分片（分collection）

4、根据对客户索引容量的监控，来确定是否对索引进行分片。当每个索引达到10G后，对索引进行自动分片。

# 大数据平台

愿景：成长为企业级大数据解决方案



## 数据采集

### 数据源

* 业务数据
* 运营数据
* 环境数据

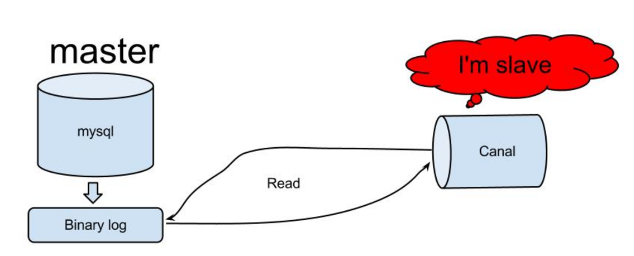
### 采集数据

快速收集数据，无阻塞。

将各种数据，搜集并发送到消息总线。包括关系型数据，非关系型数据；结构化数据，非结构化数据......

#### mysql数据库

特点：无业务侵入，业务无感知



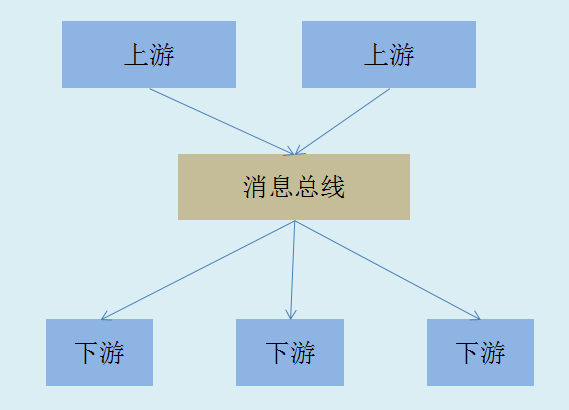
#### agent

* Flume
* filebeat
* logstash

## 消息总线

消息订阅与消费机制。可成长为企业级消息中间件。

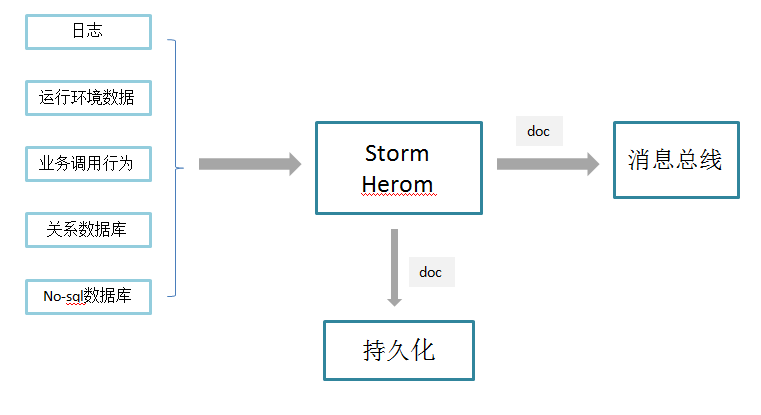
* Kafka : 高吞吐
* RabbitMQ：高可靠
* RocketMQ：高可靠



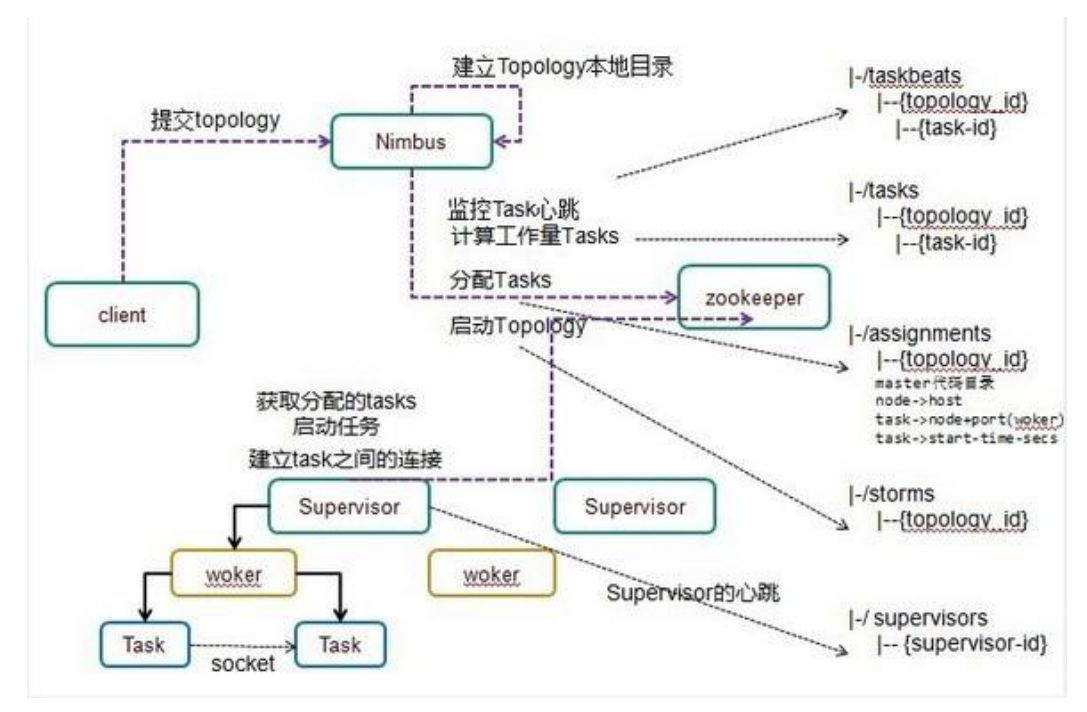
## 搜索平台

详见“[搜索平台](#_搜索平台)”章节

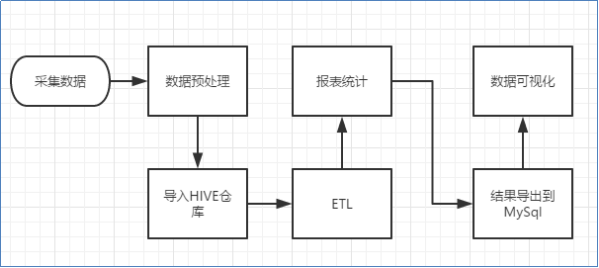
## 实时计算平台



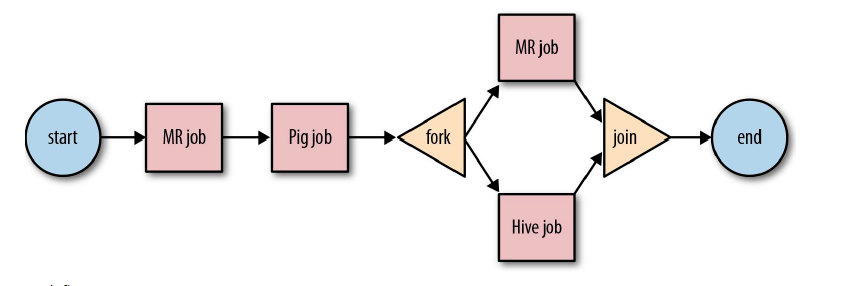
* 实时处理业务数据，将业务数据（关系型和非关系型）转换为读快模型的数据；
* 实时处理业务相关的运维数据，并反馈给监控平台，用于系统优化、自我保护和告警；
* 实时处理环境数据，并反馈给监控平台，用于环境调优、扩容；
* 实时统计业务接口调用量



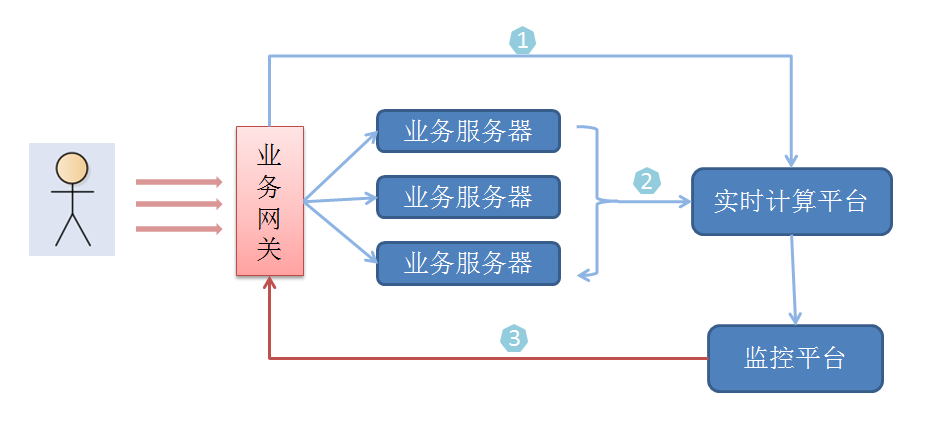
## 离线计算平台



## 任务调度



## 监控平台



<说明>

1、上报业务请求数据。包括接口调用，和系统健康检查数据

2、上报运行日志，和系统健康检查数据

3、发布业务性能监控数据，供业务网关做负载均衡参考。