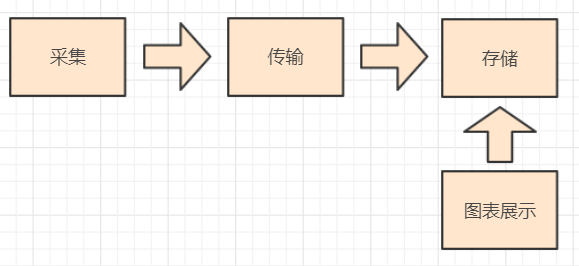
**概要：**

1. 追踪日志传输方案
2. 日志存储方案
3. 可视化展示

一、追踪数据传输方案

一个调用链系统的实现最基本的模块包括:

* 1. 性能日志采集
  2. 数据传输
  3. 数据存储
  4. 图表展示



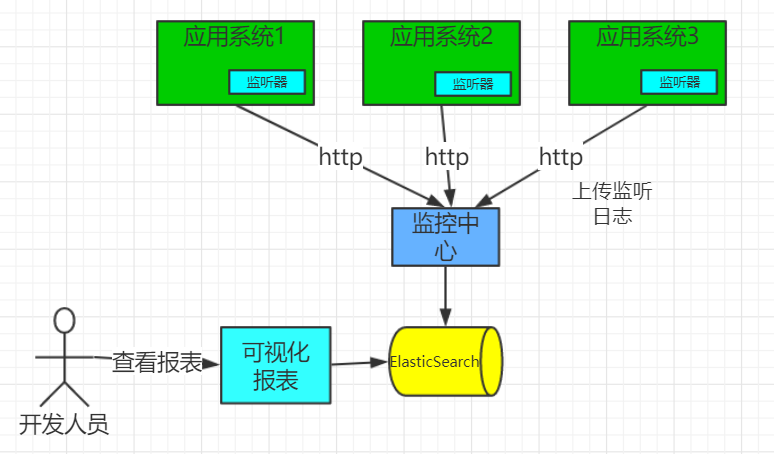
前面三节课基本上都是在讲采集，这是系统是重点和难点。剩下的传输、存储、图表展示虽然没有那么复杂但不代表它们不重要，接下来就一起搞清楚剩下的三个模块是如何实现的。

**数据传输所面临的问题和挑战**

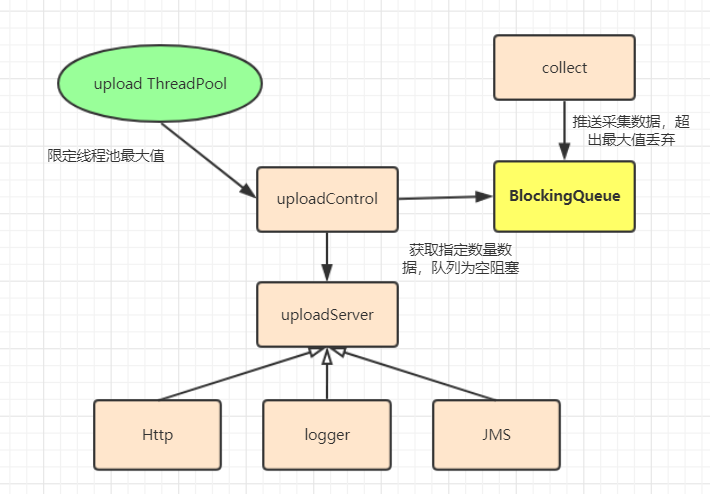
1. 业务系统高并发高承载的情况下采集器对资源的占用降至最低
2. 保证数据采集上报的及时性
3. 数据丢失率在可控范围之类

**现有架构：**

基于这些问题在来看我们架构是如何满足上述要求的：



上述架构中监听器采集到节点数据之基于Http发送至监控中心在发送至Elasticsearch进行存储。为保证不影响业务系统发送逻辑采用后台线程异步发送，并控制发送线程的数量。



上传流程说明：

* 1. 初化一个限定容量的阻塞队列
  2. 采集器抓取数据并上传至队列，超出容量直接丢弃
  3. 线程池分配上传线程
  4. 控制器取出指定数量数据，如果数量小于0线程阻塞。
  5. 调用上传服务，根据策略选择具体(http、logger、jms)服务进行发送

演示上传流程:

* 阻塞队列的存取
* 上传线程初始化
* Http具体发送

相关代码：*com.cbt.agent.transfer.UploadServiceImpl*

其它问题：

a. 上传流程为什么不直接采用线程池直接控制数据发送，而非得在自己去维护一个阻塞队列呢？首先线程池本身可以限定发送线程最大值、其次减没有了从阻塞队列当中存取的过程、另外线程池本身也有队列和相关的饱和策略设置。

答：

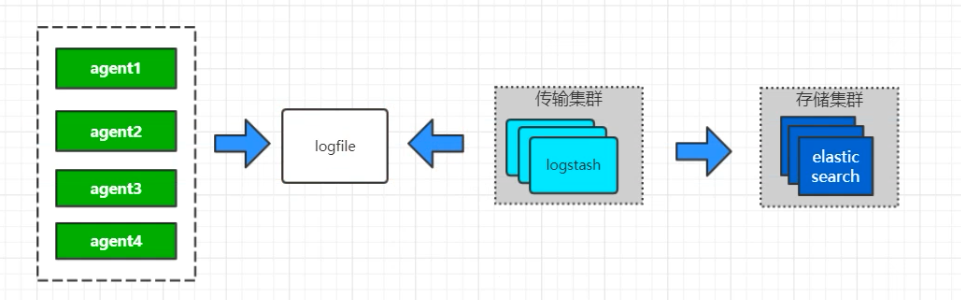
b. 采集器其中有一个是对 Http的监控，而日志传输也是采用Http 不会出现死循环吗？

答：不会，因为上传线程 没有开启监控会话

**传输解决方案升级：**

上述传输的解决方案对于，对于并发量不高的系统是最优解，因为简单。但随着目标系统的并发量增加，就会显得乏力，当然不管怎么样都不会影响到业务系统，只是采集数据会出现大量的丢失。为解决该问题就得进一步升级传输方案。

现有方案中最大的瓶颈是直接通过Http发送，所以最好的办法是先将其打印到本地日志，在基于logstash、flume日志收集工具进行发送。



二、日志存储方案

知识点：

1. 存储需求
2. 方案选择
3. 具体实现

存储需求

跟踪节点模型表：

|  |  |  |
| --- | --- | --- |
| **字段** | **类型** | **描述** |
| traceId | string | 跟踪ID |
| rpcId（EventID \spanID\nodeID） | string | 节点ID |
| appId | string | 目标应用ID |
| appDetail | string | 应用名称 |
| nodeType | string | 节点类型 |
| resultState | string | 结果状态 |
| resultSize | number | 结果大小 |
| servicePath | string | 服务路径 |
| serviceName | string | 服务名称 |
| beginTime | long | 开始时间 |
| endTime | long | 结束时间 |
| addressIp | string | 目标 IP地址 |
| fromIp | string | 发起方IP地址 |
| inParam | text(json) | 输入参数 |
| outParam | text(json) | 输出结果 |
| errorMessage | string() | 异常类型 |
| errorStack | text | 异常堆栈 |
|  |  |  |

存储方案要求：

1. 大文本的存储
2. 足够快的写入速度

方案选择:

* 1. ~~mysql~~
  2. MongoDB
  3. ElasticSearch
  4. ~~Redis~~

基于上述要求 mysql作为关系型数据肯定不能和NoSQL相比。剩下的MongoDB与ElasticSearch都满足需求，只不过ElasticSearch 关于日志传输有完整的解决方案，即ELK。另外加上其搜索功能加持 ，所以选择了ElasticSearch。

具体实现

具体实现并不复杂只需要注意两点即可：

* 1. 接收到数据后异步发送至ElasticSearch
  2. Agent 采用fastjson 作为数据存储，而ElasticSearch 采用的是jackson,特殊字符转义的时候会存在格式化失败的问。

三、可视化展示

**知识点：**

1. 列表视图
2. 调用链TreeTable
3. 调用链关系图
4. 节点详情视图
   1. 输入输出参数 Json 视图
   2. SQL语句格式化展示
   3. SQL返回结果展示

**列表视图**

主要功能：

1. 时间过滤
2. IP过滤
3. 关键字搜索
4. 基于条件查询并展示相关节点

源码位置：*com.cbt.server.control.TraceRequestControl*

页面：*page/trace/requestTableView.ftl*

**节点展示表格视图（TreeTable）**

目的只有一个以TreeTable的形式展示链条节点，并重点标记状态，简单起见这里直接选择了 EasyuI。

源码位置：*com.cbt.server.control.TraceDetailViewControl#openTraceListView*

页面：*page/trace/traceListView.ftl*

**调用链关系图**

其目的是以图的形式直观展示调用关系。

JsPlumb：是一套开源的流程图创建工具,早期一款画图工具,

D3.js：html5领域，d3可谓是最好的可视化基础库，提供方面的DOM操作，非常强大

Go.js：go.js 提供一整套的JS工具 ，支持各种交互式图表的创建。有免费版和收费版

[chartmage.co](http://chartmage.com)m: 基于文字直接生成时序图，使用非常简单。

非简单起见最终选择了Go.js 的免费版，直接基于Json进行渲染，几乎没有额外的学习成本。

相关源码：*com.cbt.server.control.TraceConsoleControl#getflowChartNodeData*

页面: page/trace/traceListView.ftl 216L

**节点详情视图**

**弹窗组件：**layer.js

**布局组件：**bootstrap.js

**SQL语法高亮**：highlight.pack.js

**SQL格式化**：com.alibaba.druid.sql.SQLUtils#formatMySql(java.lang.String)

相关源码:com.cbt.server.control.TraceDetailViewControl#openDetailView

普通节点页面：page/trace/NormalDetailsView.ftl

SQL节点页面：page/trace/SqlDetailsView.ftl