日志易
rizhiyi.com

日志易产品技术白皮书

业务运维 • 基础架构运维 • 安全审计 • 业务数据分析

北京优特捷信息技术有限公司

Version 3.6, 2020-12-03

1. 引言

1.1. 日志分析技术背景

日志是计算机系统、设备、软件等在某种刺激下反应生成，用来标明发生某些事情的消息。典型的日志消息的基本内容至少包括：时间戳、源、数据三部分。日志分析则是处理和分析日志，从中得到它的含义。健全的日志记录和灵活的日志分析系统，是 IT 系统正常运营、优化和事故响应的基础。

在 21 世纪的今天，随着互联网和大数据时代的到来，中国的企业正面临着前所未有的挑战。机房中的各种系统一直在源源不断的产生日志，但在系统出现问题或者隐患时，能不能从日志中分析出端倪？在系统遭受严重的攻击和破坏时，日志分析系统能不能帮助我们走出泥沼？这是对整个 IT 部门实际工作能力的考验。

此外，日志本身的频繁变动也给分析带来了很大的不确定性难度。根据 2016 年《软件学报》一篇综述论文的统计：

1. 在软件开发中进行日志记录是普遍的,平均 30 行代码中就有一行是日志
2. 日志信息对实际部署系统的运行故障调试帮助较大,缩短故障调试时间的加速比为 2.2
3. 日志代码的更新频率比其他代码要快约 1 倍
4. 约四分之一的日志修改是把新的程序变量写入日志
5. 约一半的日志修改是对日志消息静态文本的修改

完整的日志分析系统建设，涉及日志的结构规划、采集存储、过滤关联、统计分析、数据挖掘、图表报告以及管理章程等各个方面。几乎每个方面，都有着足够的技术深度和难点等待人们一一克服。

常见的来说，一些公司的日志分散在各台服务器上，每次查找日志都要登录到各台服务器，效率低下。这些公司首先需要统一管理日志，在一个界面上查看所有日志，大大提高运维效率。

一些公司的日志由各业务部门分别处理，导致了日志数据及分析结果的碎片化。日志是一家公司运营情况的真实数据，不同业务部门的日志往往互相关联。在公司层面统一处理、分析日志，可以把不同来源的日志对照关联分析，去除噪音，反应真实情况。

此外，黑客在入侵服务器或网络设备时，往往会删掉日志，抹除作案证据。统一上传、管理日志，可及时发现入侵行为，监控告警，也可以长期保存日志，方便事后安全审计。

1.2. 通用实现的局限性

日志分析技术在多年的发展中，已经经过了几代的发展，涌现了各种不同的技术手段和工具。这些工具在解决一些问题的同时，也都还带有一定的局限性。

1.2.1. grep/sed/awk等命令行工具

一些公司的运维工程师在运维故障发生后，登录各台服务器，使用grep/sed/awk等Linux脚本工具去日志里查找故障原因，排障时间长，未必能及时找到故障根源。

采用awk配合sort、uniq等可以进行初步的统计分析，但是终端表格类型的输出非常不直观，不具备图表可视化的视觉，无法第一时间发现异常点。

1.2.2. Hadoop/Spark/Storm等分布式系统

Hadoop、Storm和Spark都是一种开发框架，用户需要开发单独的Hadoop、Storm或Spark程序处理日志，使用门槛较高，优秀的分布式开发工程师不容易招到。

此外，Hadoop是批处理，实时性较差：不少使用Hadoop处理日志的公司通常是每天晚上处理当天的日志，第二天出统计报表。有些公司做得好些但也只能看到几小时前的日志分析。一些用户的日志结构化设计较为规范，可以使用Hadoop框架下的Hive或Pig查询日志，但延时依然可能达到几十分钟。

Storm 是流处理框架，可以很好的做到实时处理，但缺乏对历史数据，哪怕是并不太久远的历史数据的回顾能力，导致 Storm 只适合在类似 PV 统计等累计指标需求上，进行快速统计和展现。一般而言，只有大屏上的交易金额等个别关键数值有必要进行毫秒级的实时精准计算。

1.2.3. Druid/ClickHouse/Kylin等列式存储

采用列式存储系统进行日志的存储和分析，本质上依然是早期使用 MySQL 做日志存储思想的延伸。这种方式要求对日志格式有清晰的了解，对分析目的有足够的掌握，才能通过预聚合的方式，提前将日志结构化和处理成多维统计数据表，以便后续的高速查询。而配置预聚合，本身也需要调研、设计和调试时间，并不能做到全自动高效实现。

对于有明确的全文搜索需求的运维、安全场景，列式存储是力不从心的。一如早年 MySQL 方案需要额外搭配 Sphinx 索引。

这种方式一般只适合用在网站访问日志的业务分析场景上。

1.2.4. Elastic Stack/Graylog等项目

Elastic Stack是目前开源领域最流行的日志分析选择。虽然 Elasticsearch 本身并不是专门针对日志分析需求开发的，Elastic Stack确实也提供了日志的采集、传输、处理、存储、查询、统计、展现等一整个环节的能力。但过于分散的产品线导致其部署和管理具有一定的门槛，而且其开源部分缺乏常见的设计、权限、告警、管理等高级功能。事实上，从 6.0 版本开始，Elastic Stack 的绝大多数非 Lucene 层面新功能，都是在 Elastic License 商业许可证下发布。

此外，Elastic Stack 的内部 API 频繁变动，想要对其进行深度的二次开发极为艰难。

Graylog 是在 Elasticsearch 系统上整合和构造开源日志分析系统的典范。在一个统一界面上可以轻松完成对不同组件和流程的管理使用。由于架构单一，在日志量不大，延时要求不高，分析需求不复杂的情况下，使用 Graylog 搭建入门级日志分析系统不失为明智之选。但限于 Elasticsearch 的能力，Graylog 也只能提供对已经提取入库的字段的查询和简单统计功能，无法提供基于结果的二次统计和高端数据挖掘能力。

1.2.5. Grafana/Fluentd/Loki等云原生项目

在云原生项目中，经常会使用 Grafana 展示容器和微服务环境的监控指标和统计数据。其中部分数据的来源就是日志。Grafana 也在 2019 年顺势推出了 Loki 开源项目，配合 Fluentd 采集，专门进行容器环境日志的存储和查询。但 Loki 的设计前提是：容器平台上，微服务拆分已经非常细粒度，每个日志仅凭 kubernetes labels 或 docker tags 的查询已经可以充分定位，用户只需要上下页看日志原文就行了。这就意味着，loki 不提供日志原文的索引查询能力，不提供日志字段提取及统计分析能力。

由于没有全文索引，关键字过滤通过并发 grep 方式进行，而 Loki 项目还处于早期阶段，容错处理不佳，时间跨度稍微大一些，就会进入僵死状态。

2. 日志易技术原理

日志易是由北京优特捷信息技术有限公司开发的智能日志中心。围绕公司特制的 beaver 日志搜索引擎平台，提供了配置简单、功能强大、容易使用的日志采集、处理、分析和管理功能，帮助企业进行线上业务实时监控、业务异常原因定位、业务日志数据统计分析、及安全与合规审计。

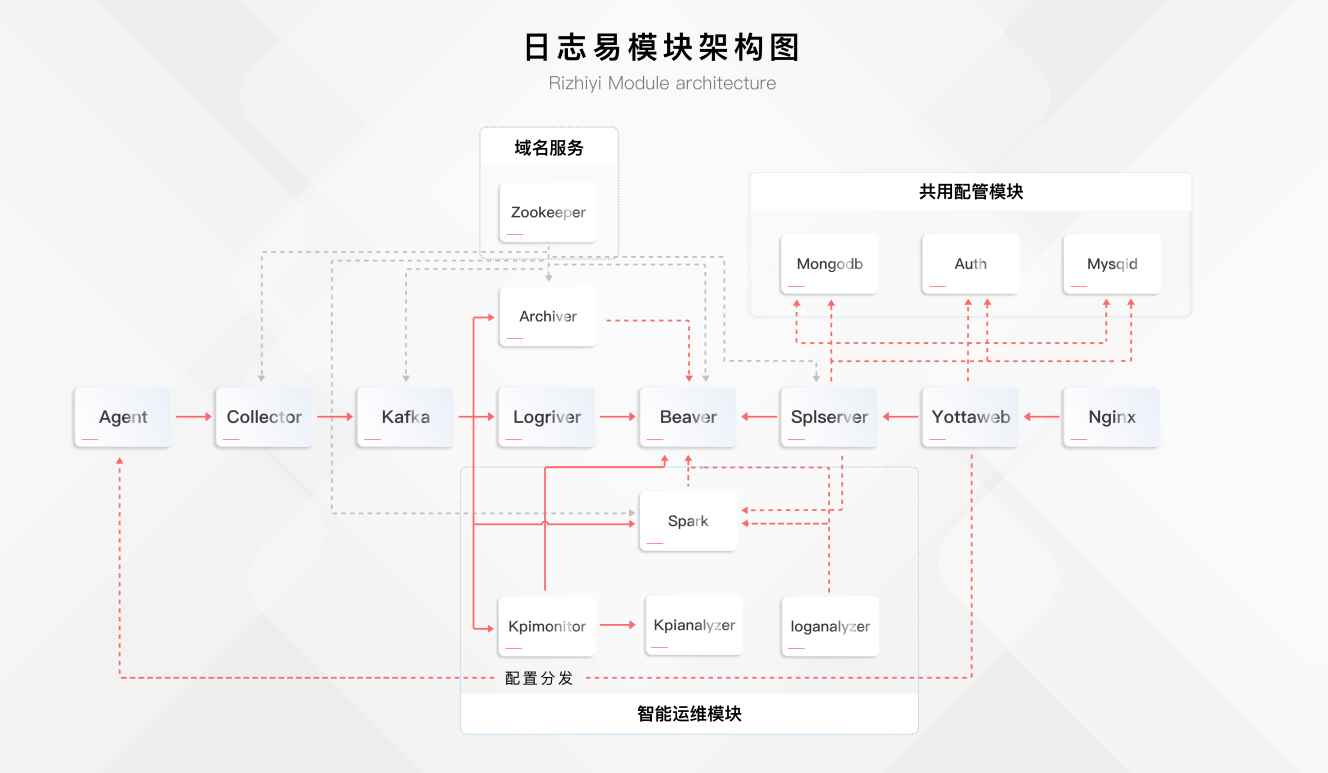
本章节主要讲述日志易产品模块架构，主要模块的技术原理及特色。

2.1. 架构图

日志易产品架构图如下：

2.2. 数据流图

日志易采用分布式服务架构，由不同模块负责数据处理的不同阶段。各模块的数据流图如下：



图中最中心部分，从左往右的直线，就是日志数据在日志易中的读写主流程。

2.3. 主要组件特色

2.3.1. 数据采集 Agent

日志易支持多种数据接入方式，包括直接由客户端发送数据到 collector 模块，比如标准的 syslog 协议 (RFC5424)、HTTP(S) 和 protobuf。更主要的方式则是通过日志易提供的专属 Agent。

日志易提供两种 Agent 实现方式，一种为 Golang 语言编写的轻量级实现，功能全，性能好。另一种为不支持 Golang 语言的 UNIX 平台单独开发的 Java 语言编写，仅支持基础功能。一般情况下，建议用户采用 Golang 版本 Agent。

为了解决跨机房传输的网络安全问题，日志易还提供了专用的 Agent-Proxy 实现，同一机房内的数据可以通过 Agent-Proxy 进行中转代理，方便网络策略设置。

在数据采集层面，开源社区选择很多，功能和性能上各有差异。本节对此稍作对比。

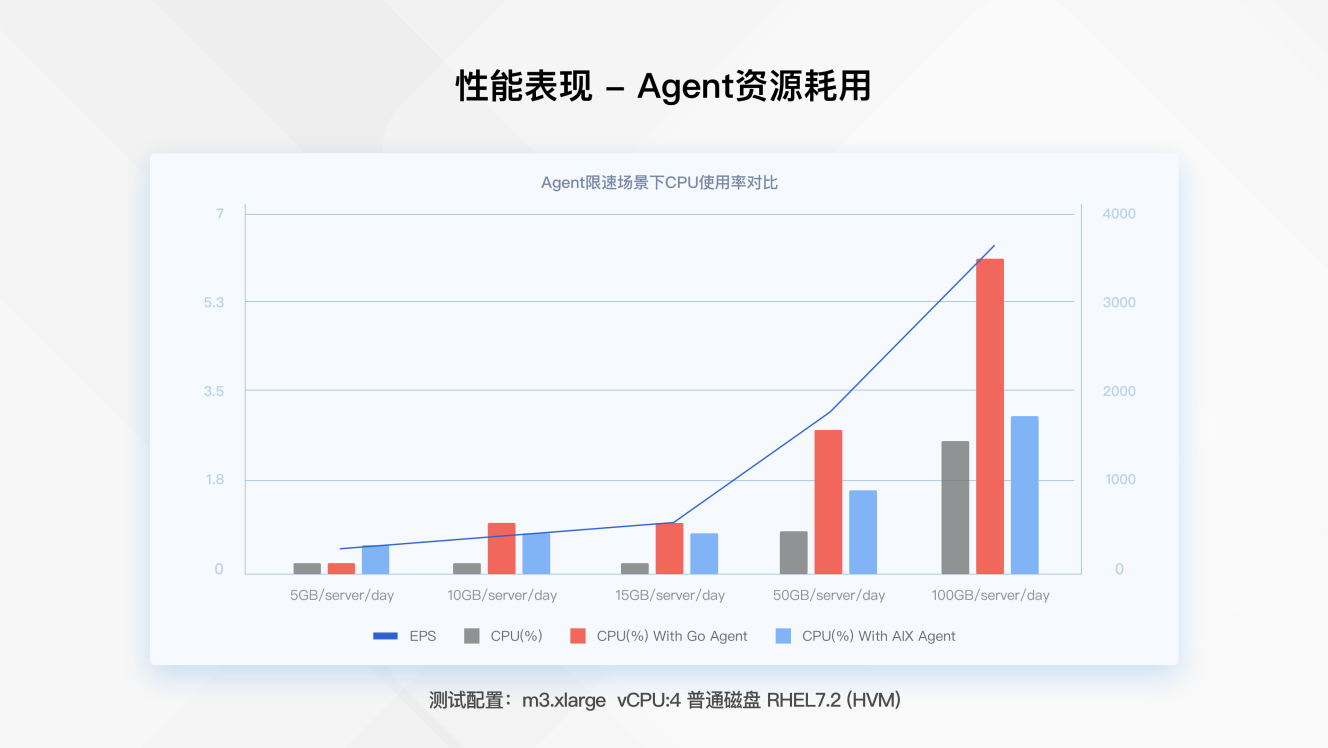
日志易Agent与主流开源Agent的功能对比见下表：

Table 1. 开源Agent功能对比

	Rsyslog	Syslog-ng	Logstash	Fluentd	日志易Agent	nxlog
语言	C	C	Java/ruby	C/ruby	Go	C
源日志支持目录	Inotify模式下支持通配符	√	√	√	√	√
源日志支持exclude	×	×	√	×	√	×
支持其他脚本输出作为input	×	×	√	√	√	√

	Rsyslog	Syslog-ng	Logstash	Fluentd	日志易Agent	nxlog
SNMP TRAP支持	╳	PE版支持	√	╳	√	企业版支持
多行处理	√	√	√	√	√	√
缓存在内存，可配置	√	√	√	√	√	√
缓存在文件，可配置	√	√	√	╳（替代方案是用RabbitMQ output）	√	√
支持FlowControl，（输出有问题时，降低读文件速度）	╳（可配置rate limiting，控制input速度，且可以配置Queue Slow Down在部分时段减速）	√	√	可以配置rate limiting	目前用go的channel能达到部分效果，如果output出问题，input的向后转的channel会满，导致input会阻塞	√
支持RPC输出	╳	╳	√	╳	╳	╳
支持批量传输	√	√	√	√	√	√
支持压缩	√	√	√	√	√	√
支持加密	√	√	√	√	√	√
文件输入时，允许重传，防止日志丢失	╳	╳	√	╳	╳(文件输入，tcp输出时，会将输出的数据先写文件，然后从文件读出）	企业版支持
兼容windows	付费版支持	PE版支持	√	╳	√	√
中央配置下发	╳	╳	商业版支持	╳	√	企业版支持
Plugin的支持	√	√	√	√	√（静态编译go plugin，动态加载lua plugin）	√

日志易Agent与主流开源Agent在相同发送速率下的CPU使用率对比见下图：



日志易Agent与主流开源Agent在相同资源消耗下的最大发送速率对比见下图：

2.3.2. 消息队列 Kafka

在数据接入以后，将首先进入消息队列组件中进行暂存。部分开源/商业日志分析产品中并不引入消息队列组件，在数据量因为业务原因突发增量时，很有可能会导致后续索引流程达到瓶颈，最后导致整个集群性能波动直至数据丢失。

因此，从数据可靠性角度出发，消息队列是日志分析系统中不可或缺的组件。

此外，消息队列组件还有助于不同目的的数据分发和消费。在日志易平台中，包括数据清洗组件 LogRiver、备份组件 Archiver、智能运维组件 KPIMonitor、数据工厂组件 DataFlow 等，都可以从消息队列中的特定

Topic 主题进行数据消费。日志易同样允许非自身的其他第三方应用进行数据消费。

2.3.3. 数据清洗和路由 LogRiver

对数据进行 ETL 清洗是日志分析的前提步骤。为了尽量减少对客户应用的影响，数据采集层一般不进行具体的数据清洗操作。LogRiver 组件是日志易研发的数据清洗组件，它运行在消息队列组件的下游，可以进行无状态横向集群扩展。LogRiver 专注于流式单行日志解析。

除了数据清洗以外，LogRiver 还负责对数据的转发路由管理。目前，日志易支持对清洗以后的日志数据，根据指定规则，转发到消息队列组件的其他 Topic 主题，或者写入到索引存储组件的其他 Index 索引。经测试验证，LogRiver 组件比基于开源 Spark Streaming 实现的数据清洗入库速度，在相同资源消耗情况下提高了一个数量级。

2.3.4. 索引存储 Beaver

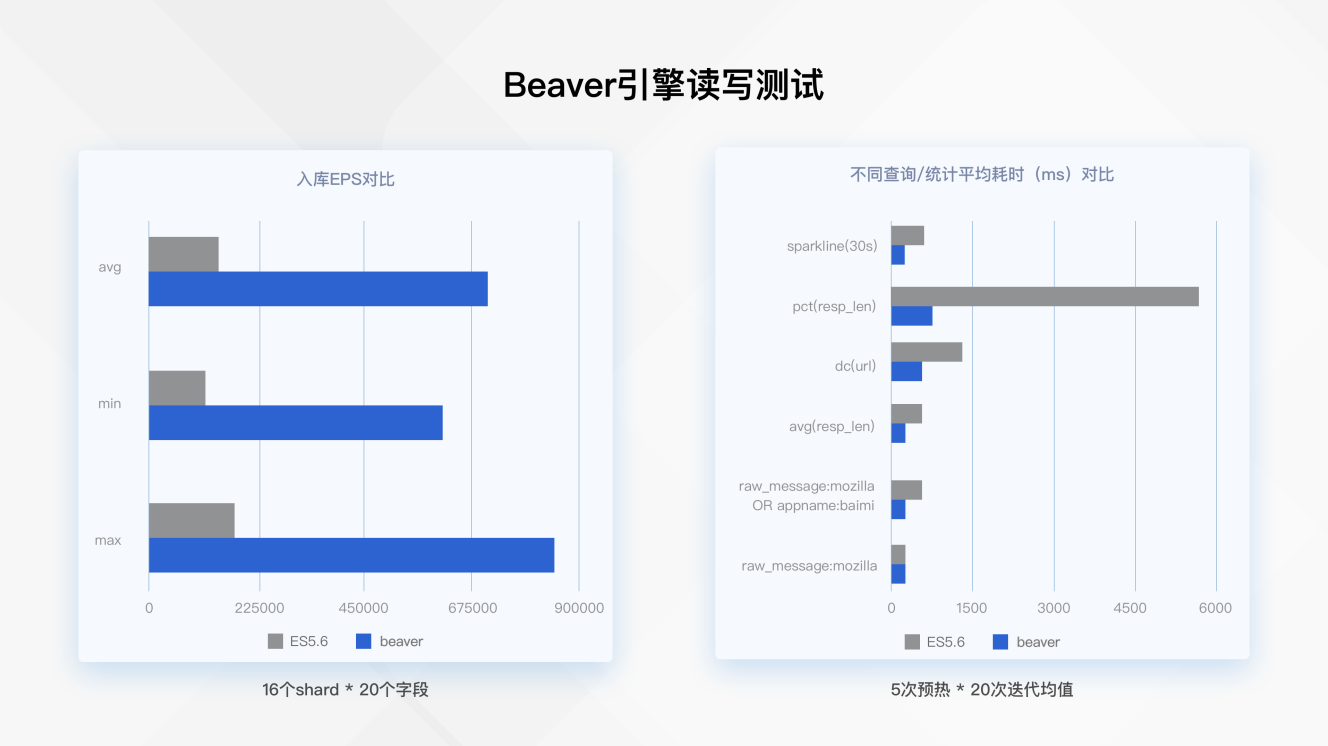
Beaver 存储引擎是日志易的核心技术所在。日志易在吸取了 Lucene 开源社区多年积累经验的基础上，针对日志数据的特点，采用 C++ 语言自主开发了 Beaver 索引存储组件。和主流开源方案相比，在日志分析场景上，提供了诸多功能和性能上的优势。

Table 2. 方案优缺点对比

对比	开源方案	Beaver
真实时	1. 准实时索引检索引擎 2. 模拟实时需要定期Flush将数据转化成磁盘结构（Refresh）,会消耗大量IO	1. 纯实时索引检索引擎 2. 内存原地检索不必耗费I/O
并发写入效率提升400%－500%	1. Segment只能由一个线程写 2. 产生的Segment的数目极多，每个Segment的DOC数目少，严重依赖Merge 3. Merge会占用大量的IO和CPU 4. Merge不及时会导致低性能	1. Segment支持内部并发写 2. 单Segment更大，减少Segment数量 3. Merge的规模小，正常情况下因Merge产生的IO和CPU消耗很低
查询性能提升50%－200%	1. 实时系统中会导致频繁的Global Ordinal失效和重建 2. 由于Segment频繁变动导致Global Ordinal大量失效 3. 每次检索都需要检索全部的Segment 4. 按时间排序取Top N的算法不够高效	1. Global Ordinal能长期使用 2. 可以根据时间戳和Query的范围排除大量不需要检索的Block 3. 某些情况下可以优化Query，不进行时间戳过滤 4. 按时间戳排序取Top N时排除掉大量的Block

对比	开源方案	Beaver
有效的内存控制 能同时打开更多索引	1. 能同时打开的索引数目有限 2. 所有打开的Segment都需要加载全部数据的Meta 3. Meta往往耗费大量的内存	1. 选择性加载Meta 2. 可以管理的Segment没有内存限制 3. Meta耗费的内存可以控制
更有效冷温热索引分级控制	1. 长期保留索引时，需要把所有索引源数据打开，带来很大资源消耗 2. 需要上层干预索引打开和关闭 3. 无法自动进行索引分级控制	1. 对索引打开数量不敏感 2. 引擎会自动根据ssd，sata，nas不同级别存储完成热，温，冷索引迁移和控制
内存控制更好	1. 采用Java开发 2. 高性能依赖JIT编译器 3. 容易引起GC	1. 采用C++开发 2. 性能优化可以做到极致 3. 内存使用完全可控

Beaver 与主流开源方案的读写性能测试对比见下图：



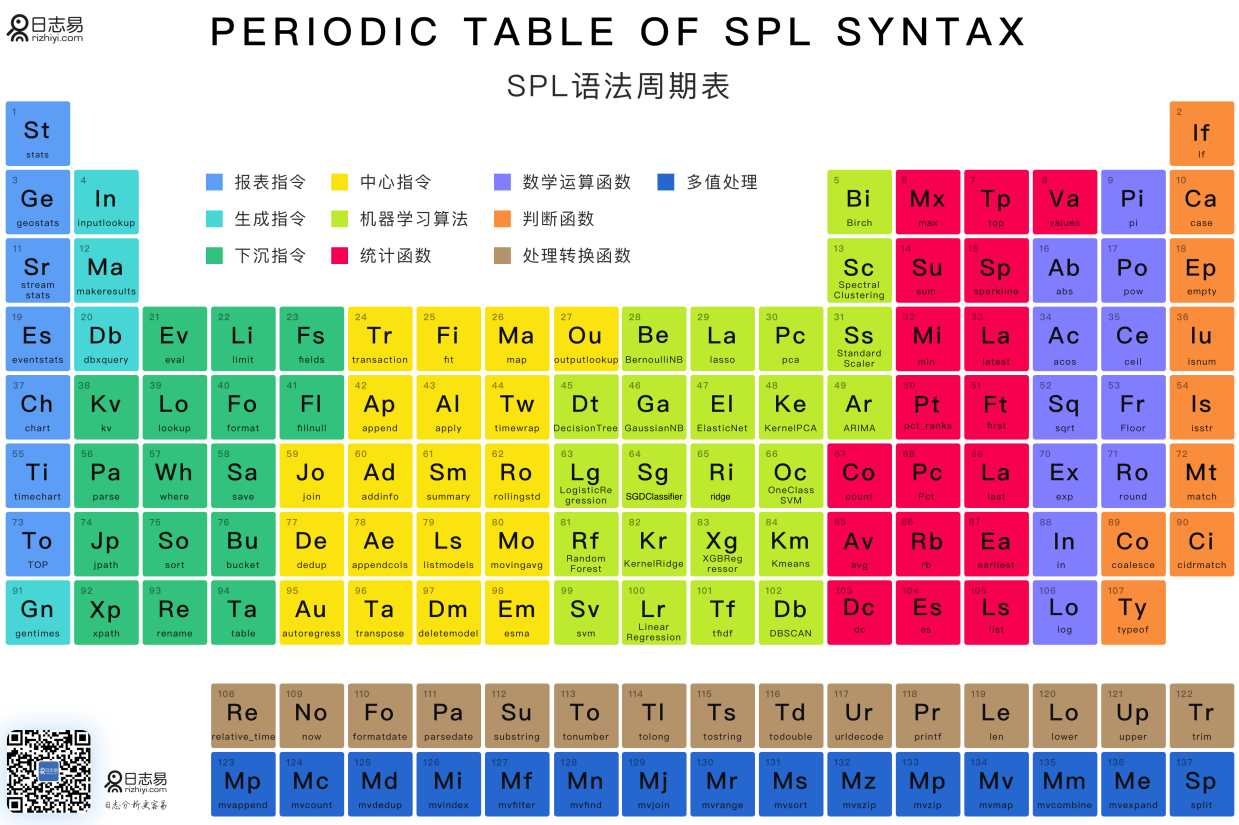
根据全方面对比，同样入库和查询条件下，beaver比主流开源方案要节省一半硬件成本。

2.3.5. 搜索统计 SPLServer

为了提供更加丰富和灵活的查询和统计功能，日志易设计了独特的 SPL(Search Processing Language) 语法。搜索统计模块 SPLServer 承担了对 SPL 的语法解析和任务调度工作。

其中，SPL 语法指令又分为流式指令和集中式指令。流式指令部分，SPLServer 在解析之后，可以自动分发到 Beaver 存储组件上分布式执行。集中式指令则在 SPLServer 本地执行。

此外，根据 SPL 语法作用的不同，还能分为生成指令等 10 类，我们总结 SPL 语法周期表如下：



SPLServer 上负责进行两类任务的调度工作。一类是查询层级，SPLServer 会自动对所有的 SPL 查询进行基于时间的分片，并对分片后的子任务进行公平轮询调度，以尽量保证不同用户的查询都能得到应有的响应。避免主流开源方案中一个大范围查询挂起全集群的现象。此外，SPLServer 可以根据用户需要，将特定场景(如海量分组数据精准统计)的查询，自动转换为 Spark 任务，分发给 Spark 集群执行。避免主流开源方案中对海量分组统计的结果误差和 OutOfMemory 故障现象。

另一类是结果处理层级，SPLServer 负责日志易所有需要后台离线执行的查询任务的执行计划调度。包括：离线任务、下载任务、定时任务、告警任务、报表任务等。

SPLServer 同样可以横向集群扩展，其中任务状态和任务结果数据，将依赖于名字服务组件 Zookeeper 和共享存储组件 MongoDB GridFS。

2.3.6. 展现和接口 YottaWeb

在搜索统计基础上，所有的日志易展现层功能，包括但不限于：搜索、统计、可视化、仪表盘、报表、告警等。均有 YottaWeb 组件提供。此外，YottaWeb 组件还负责权限配置、资源管理和数据查询等各种功能服务的对外 API 接口。

YottaWeb 组件同样可以横向集群扩展，并通过最外层的负载均衡组件 Nginx 进行代理转发。

2.3.7. 算法服务 Analyzer

算法服务组件包括 KPIAnalyzer 和 LogAnalyzer。组件提供了日志易针对智能运维需求定制开发的各种 AI 算法。算法服务组件本身并不提供数据处理和产品功能层面的封装。指标数据的消费处理和算法调用，由 KPIMonitor 完成。日志数据的消费处理和算法调用，则会生成 SparkStreaming 任务在 Spark 组件上完成。

2.3.8. 管理维护 Manager

日志易集群本身的运行维护，包括各组件的部署、配置变更、性能监控告警、版本升级，通过独立的 Manager 组件完成。日志易本身可以脱离 Manager 组件独立运行，但是为了长期运维考虑，建议用户采用 Manager 模块来管理和维护日志易集群。其中，各组件的性能监控指标数据，也同样存储在独立的 InfluxDB 组件中。

3. 日志易部署需求

3.1. 服务器要求与资源评估

应用部署策略用以指导用户依据自身的业务规模，以及对性能、可靠性等方面的具体要求，来确定合适的系统配置和部署方案。用户的环境和要求千差万别，本节只是给出一个指导性的配置策略，依实际情况的不同，用户可能需要在本部署策略的基础之上做适当调整以满足特定需求。

3.1.1. 单服务器最低要求

在数据流量较小，eps 在 1000 以下的体验试用环境中，可以尝试最低配置的单台服务器部署。日志易要求的最低配置如下：

- OS：CentOS 6.5 x86_64
- CPU：4核 2.0GHz
- MEM：16GB
- DISK：300GB

数据量较大的正式环境中，需要在数台到上百台服务器上进行集群化部署。集群化部署的资源评估参见稍后章节。

3.1.2. CPU资源需求评估

集群部署的资源需求因规模不同而异，但规模只是影响部署需求和架构的诸多因素之一。以下推荐值可为您的规划提供一些参考，每个特定部署的实际数量将有所出入，根据日志量大小，我们分别为用户提供了小型、中型、大型3种推荐部署方案，部署的特性将随着规模的增长而变化，您可以在一定程度上了解所需要的内容。

- 小型

  数据量20-100 GB/日，1000 ~ 5000 eps （event per second，每秒日志数量），需要10-20核CPU。

- 中型

  数据量100GB-300GB /日，5000 ~ 15000 eps，需要20-54核CPU。

- 大型

  数据量 >= 300GB /日，20000+ eps，至少需要54核CPU。

3.1.3. 磁盘容量需求评估

磁盘容量需求主要集中在消息队列集群和索引集群。我们建议使用带副本的集群方案作为数据容灾的考虑。对于数据的保留方式，通过如下的计算公式便可以根据自身情况进行磁盘容量评估。

- 消息队列集群

  磁盘容量 =（原始日志大小（GB）／24小时）* 峰值（3倍）消息队列存储放大系数（6）* 存储份数（2份）* 保留小时数 ／（机器数 * 磁盘使用率（80%））

示例：

机器数量	保留时间	存储份数	单台磁盘	计算方法
3	2 小时	2	1.25 TB	（原始日志大小（1000GB）／24小时）* 峰值（3倍）* 消息队列存储放大系数（6） * 存储份数（2份）* 保留小时数 ／（机器数 * 磁盘使用率（80%））
	8 小时	2	5 TB	

- 索引集群

磁盘容量 = 原始日志大小（GB）* 索引放大系数（2.5倍）* 索引份数 * 保留天数 ／（机器数 * 磁盘使用率（80%））

索引放大系数与抽取的关键字段数相关，抽取的关键字段数越多，索引放大系数就越大，2.5倍是最大值。

示例：

机器数量	保留时间	索引份数	单台磁盘	计算方法
36	3 天	2	520GB	原始日志大小（1000GB）* 索引放大系数（2.5倍）* 索引份数（2） * 保留天数 ／（机器数 * 磁盘使用率（80%））
	7 天	2	1.22 TB	

3.2. 浏览器要求

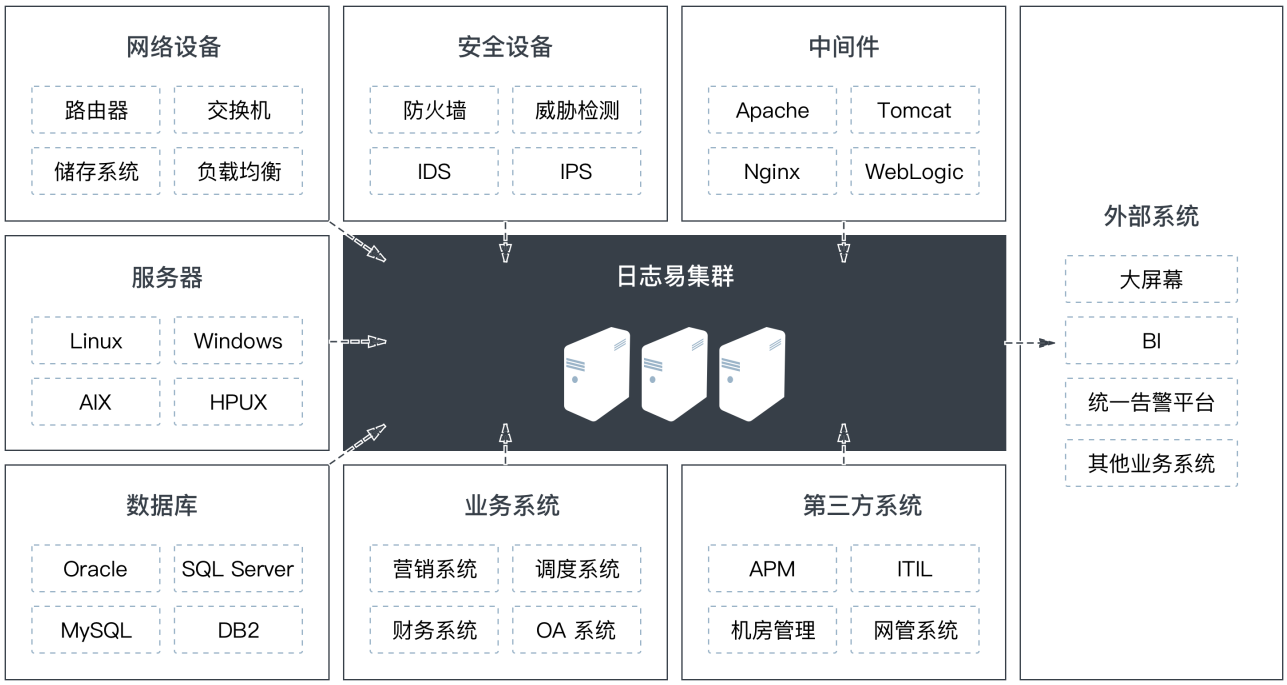
日志易支持采用当前主流浏览器访问，包括 Chrome，Firefox，Safari，Edge 等。

如果想获得比较好的体验，建议:

- Chrome75+

4. 日志易应用场景

日志易作为通用的日志分析技术平台，可以在各种不同的 IT 场景上发挥重要作用。

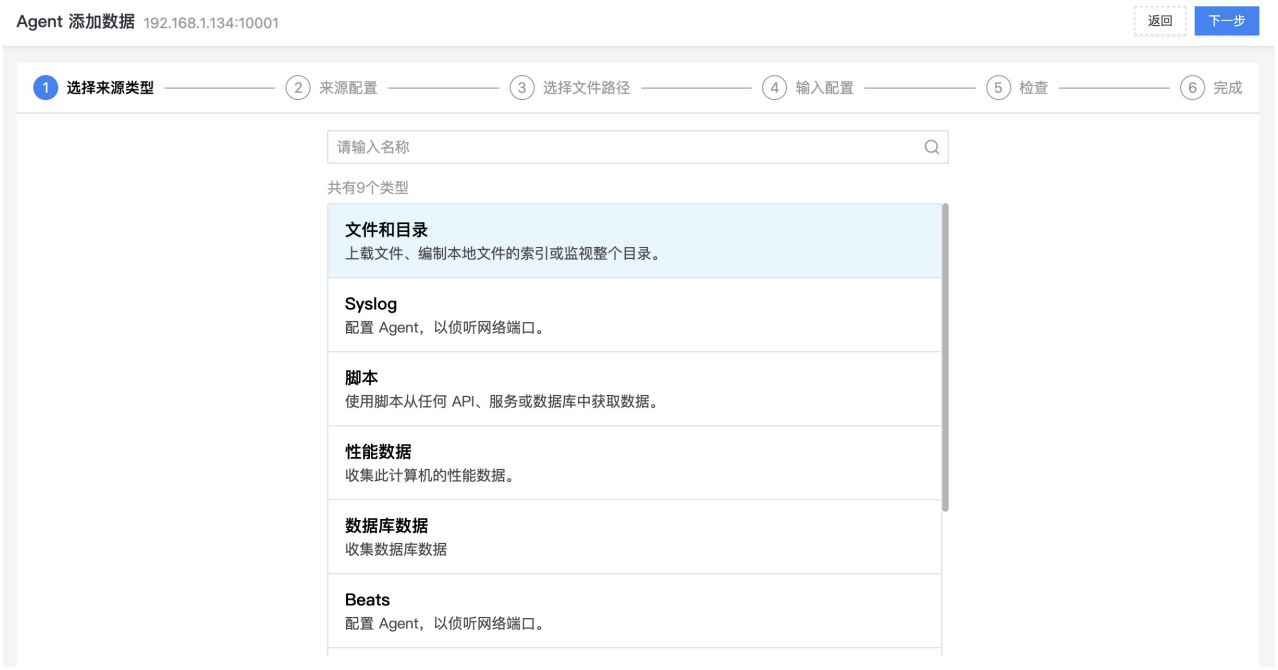


1. 端到端的全链路性能监控
    - 通过日志或客户端埋点数据对接方案，如开源的 count.ly 方案，进行最终用户监控(Real User Monitor)
    - 通过日志或服务端埋点、JVM 探针数据对接方案，如开源的 zipkin、skywalking 方案，进行应用性能监控(Application Performance Monitor)
    - 关联不同系统或模块的日志，进行端到端的服务监控和故障排查
2. 安全信息与事件管理（Security Information and Event Management）
    - 通过服务器日志发现端口扫描和非法入侵
    - 防火墙、网络设备、服务器日志安全跟踪分析
    - 用户及端点行为分析审计(User & Entity Behivour Analysis)
    - 安全编排和自动响应(Security Orchestration, Automation & Response)
3. 业务统计分析
    - 网站用户及手机用户访问统计及留存分析
    - 社交、视频、电商、游戏网站用户行为及交易路径分析
    - 客户端设备、操作系统、浏览器统计
4. 运维故障和程序Bug分析
    - 通过日志对网络设备、服务器及应用程序状态实时监控，迅速定位问题根源
    - 快速关联分析大规模分布式系统各个模块产生的大量Debug日志

5. 日志易产品功能概览

5.1. 日志采集

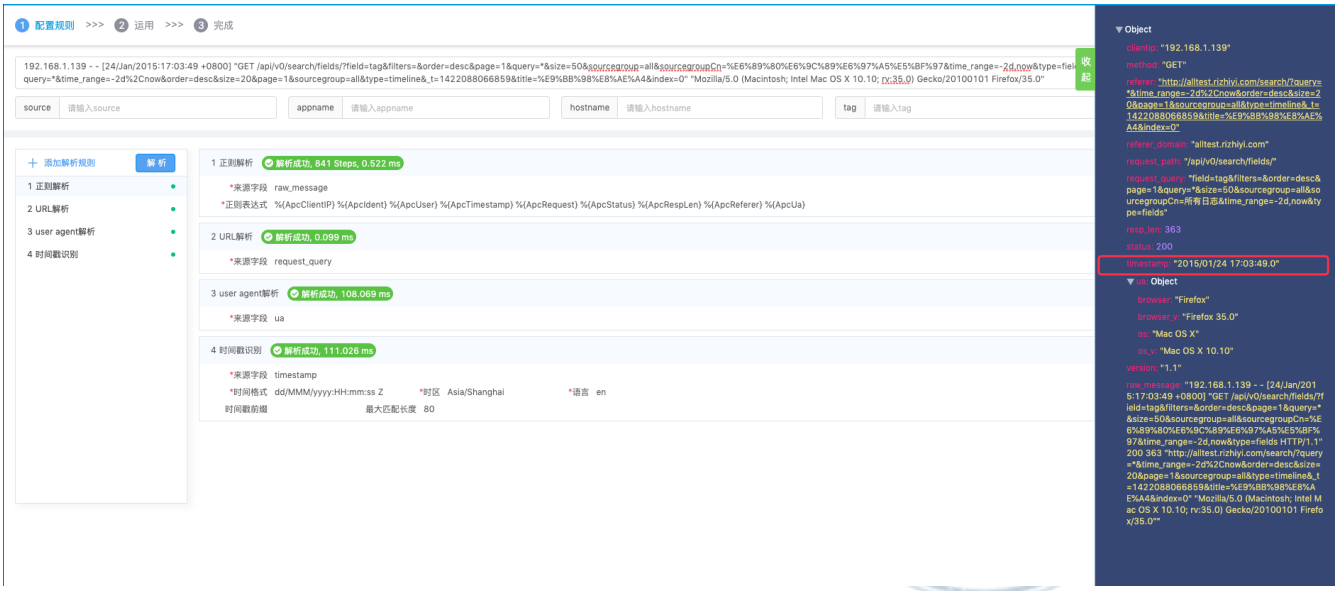
- 日志易使用Linux系统自带的rsyslog或syslog-ng agent，或日志易agent，可采集服务器、网络设备、操作系统、应用系统的文本日志及二进制日志数据。我们支持标准的syslog协议（RFC5424）、HTTP和HTTPS。
- 向导式的数据采集流程



- 日志集中管理：无需登录单台服务器或授权开发人员访问生产环境，所有日志都可通过日志易Web界面授权访问。
- 支持各种类型的日志：任何基于文本类型的日志，无论来自服务器或是客户端，例如Apache、Java、PHP、Tomcat、MySQL、syslog-ng、rsyslog、nxlog、路由器等网络设备的日志，都可以上传到日志易。

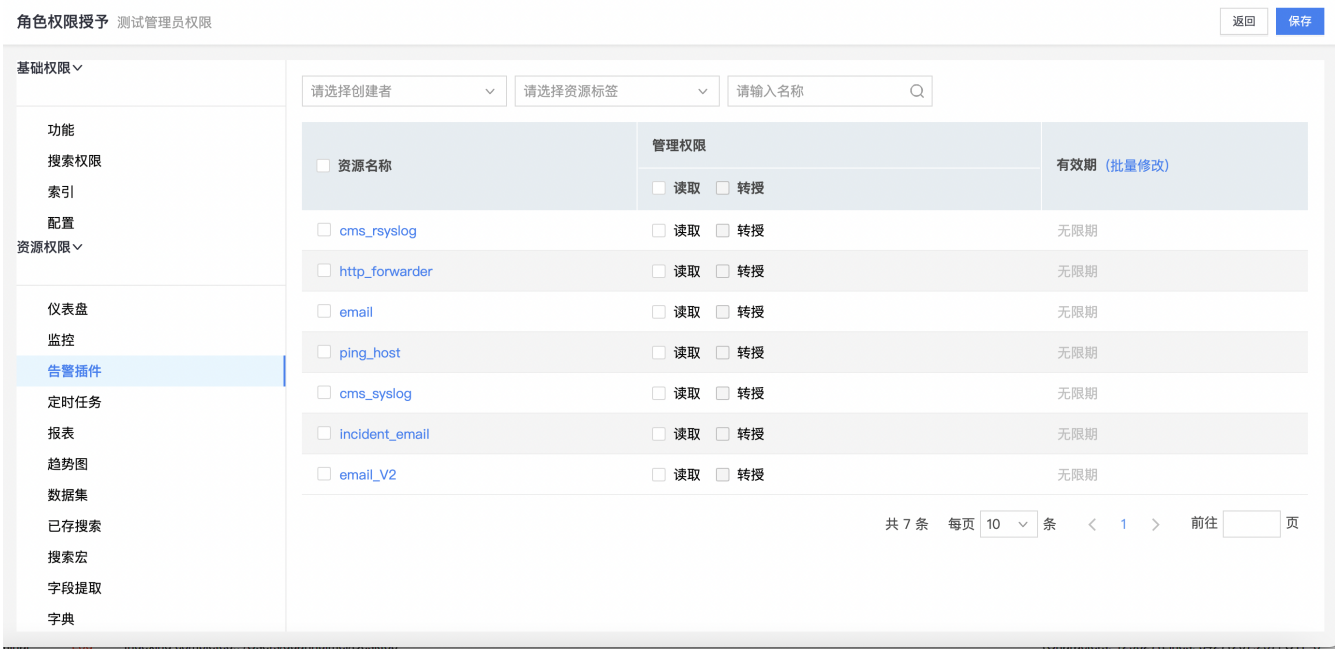
5.2. 字段提取

- 自动解析日志：自动提取日志的关键字段，将非结构化日志转化为结构化数据，标准的日志格式支持有Apache、Nginx、Syslog、Java、JSON等。
- 日志格式自定义：为特殊日志格式量身定做，提供向导式解析规则配置，实现精准解析，提供正则匹配、KeyValue分解、url解码、时间戳识别、字典翻译、IP 地址库等多种提取方式，简单操作，轻松上手。



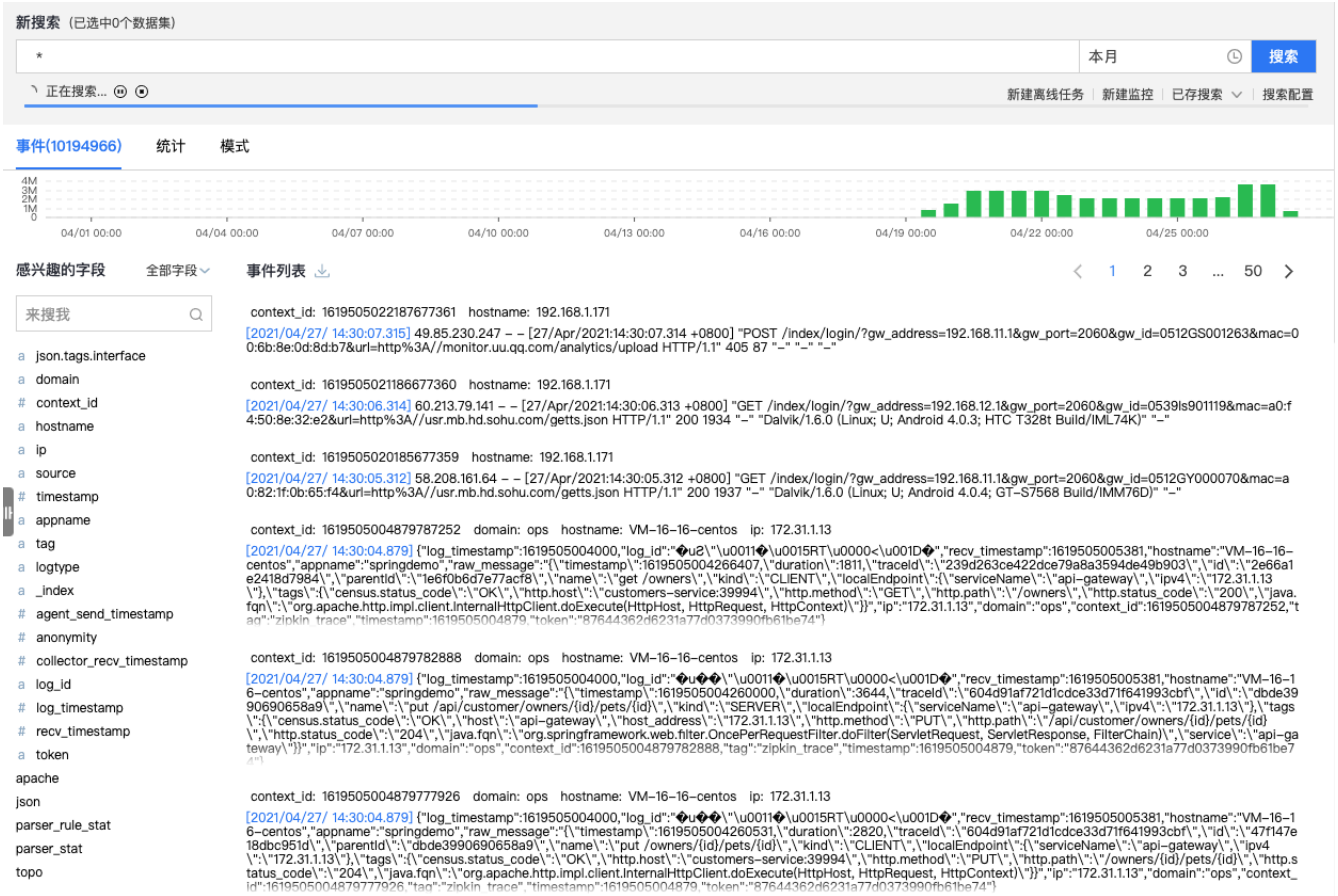
5.3. 权限管理

· 用户角色：建立完善的基于角色的用户权限体系。不同角色的用户使用日志易的功能及搜索、统计、告警等资源时，都可以限定到不同级别的权限；



5.4. 搜索

· 强大的搜索功能：支持全文检索，还可以使用字段、数值范围和布尔检索，查询指定时间范围内的日志。



5.5. 监控

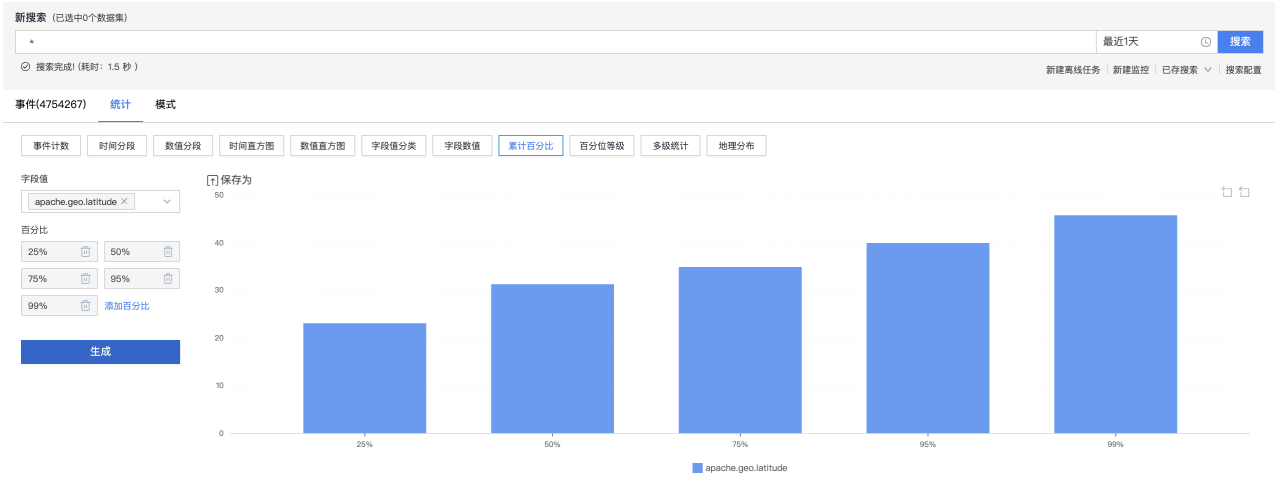
- 监控告警：基于已存搜索配置灵活的监控告警，支持秒级告警，当触发配置的告警条件时通过邮件、短信等接口通知。

· 事件处理：对告警事件提供完善的归并、抑制、处理标记、事件操作功能，并提供处理状态统计报表。

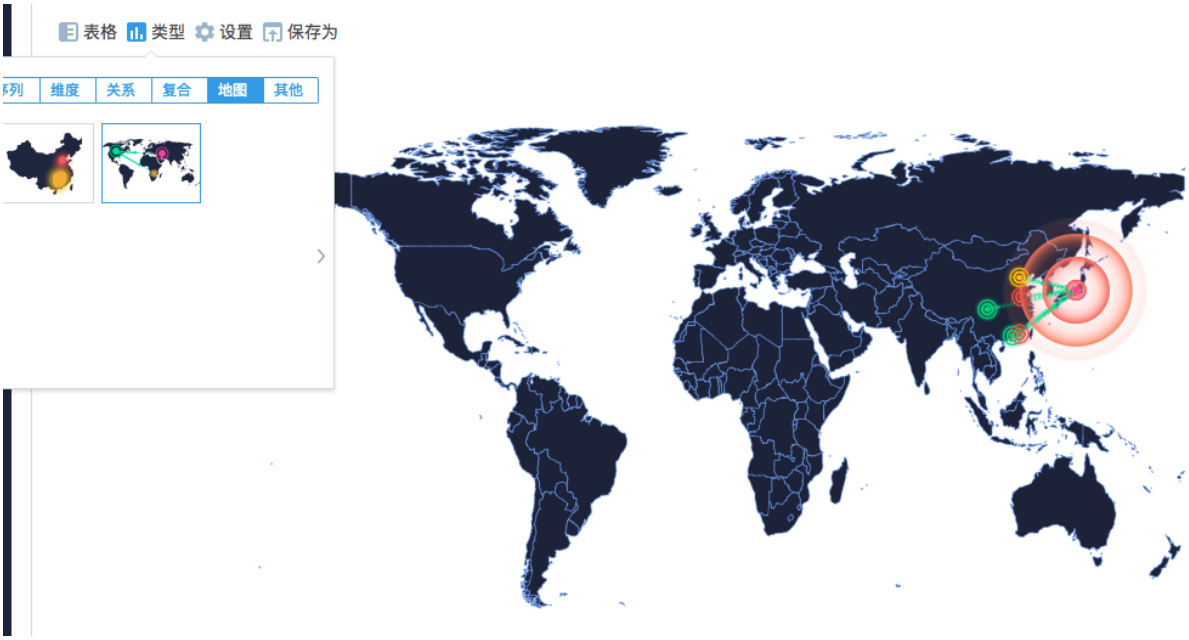
[incident batch operation] | *images/incident-batch-operation.png*

5.6. 可视化与统计

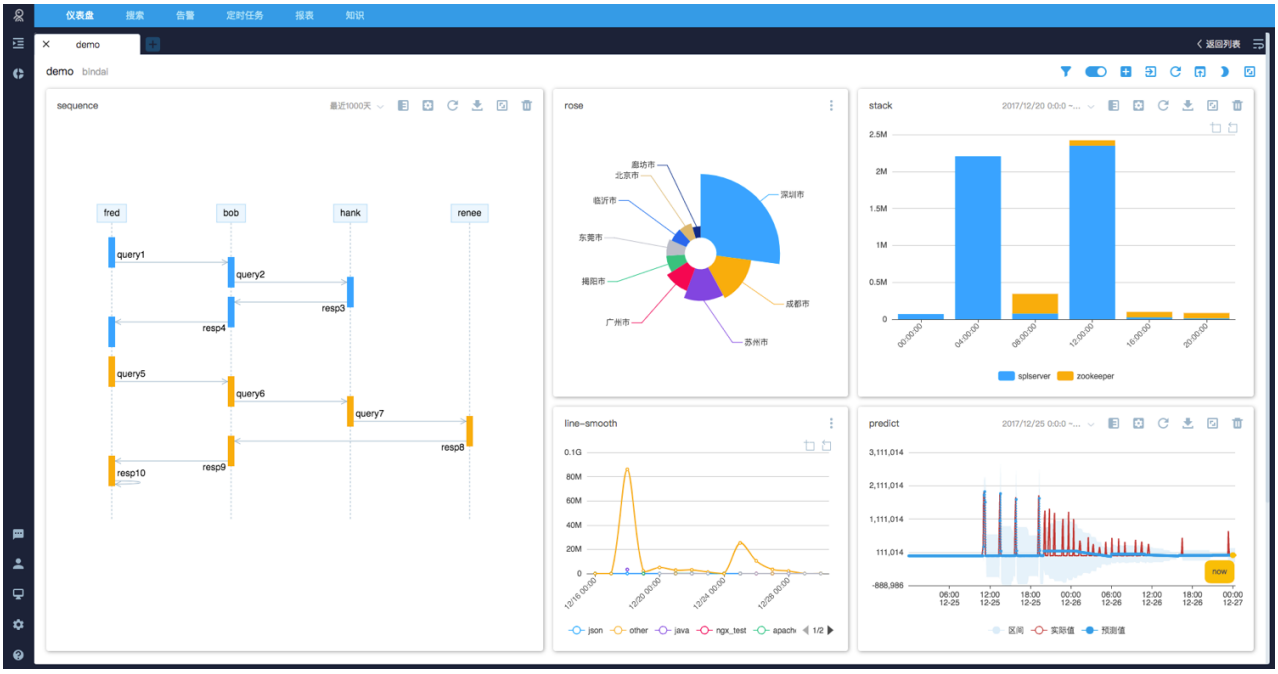
· 快捷统计：统计类型可支持计数统计、时间分段统计、数值分段统计、时间直方图、数值直方图、字段值分类、字段数值统计、累计百分比、多级统计、地图展示等多维度统计方式，数值型字段可从下拉菜单选择计数、求和、最大值、最小值、平均值、标准偏差等创建折线图、条形图、饼状图等不同形状的趋势分析图。



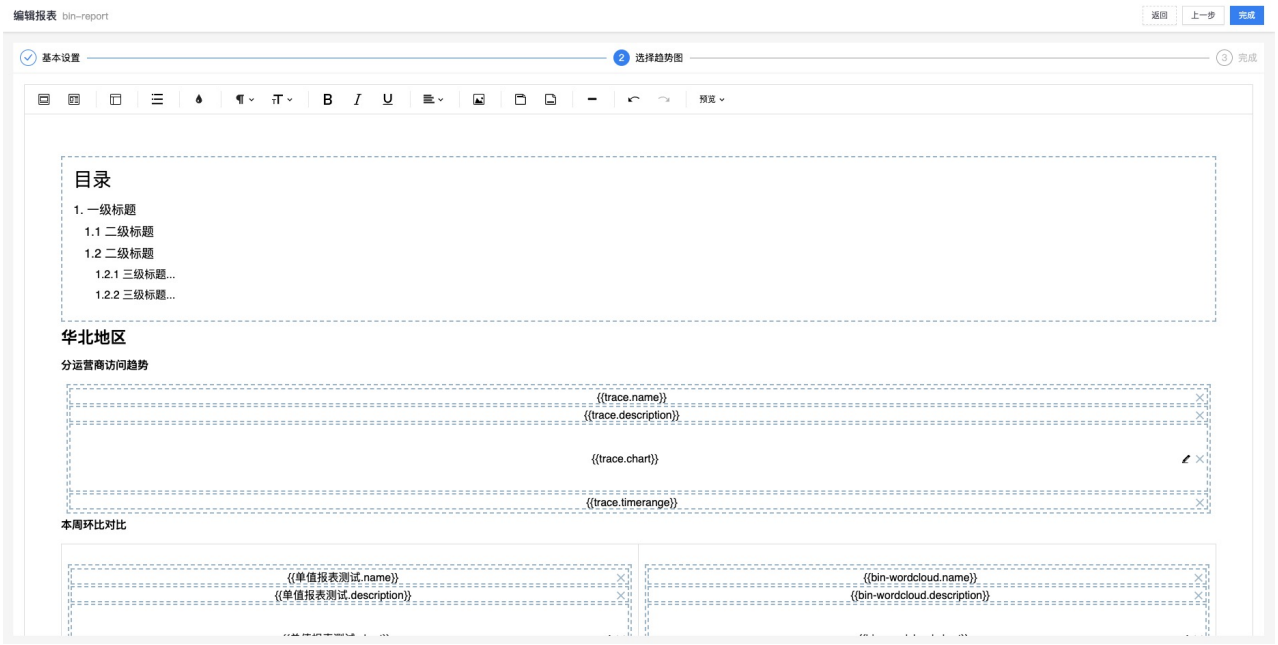
- 高级可视化：在电子表格的基础上，用户可以选择指定字段设置，生成自己想要的复杂可视化效果。日志易提供折线图、区域图、堆叠区域图、散点图、柱状图、分组柱状图、堆叠柱状图、饼图、玫瑰图、条带图、和弦图、桑基图、力引导图、区间图、多Y轴、区划图、热力图、轨迹图、单值、水球图、词云图、循序图等可视化效果。



- 仪表盘：通过趋势图组建仪表盘，拖放选择合适的大小来组织界面布局。仪表盘支持分享，支持自定义钻取和过滤功能，用户可以快速查看符合条件的数据。



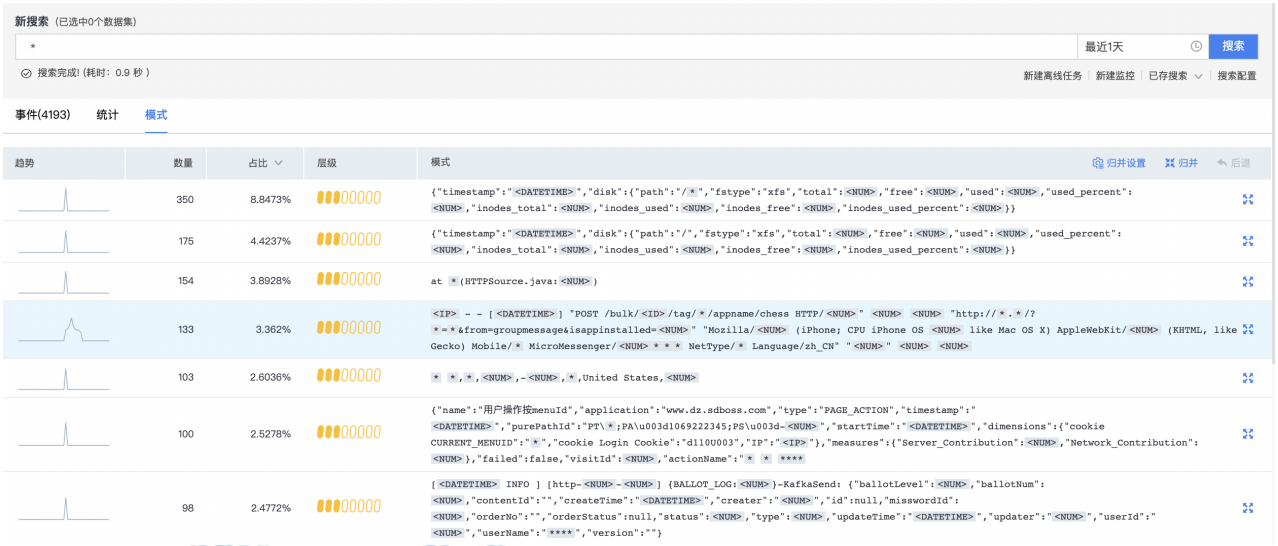
· 报表：支持多种文件类型的个性化日报、周报、月报，可设置报表布局和接收邮箱地址，方便查看图形化报表内容。并支持自定义布局的 PDF 类型报表和自定义富文本模板的 Word 类型报表。



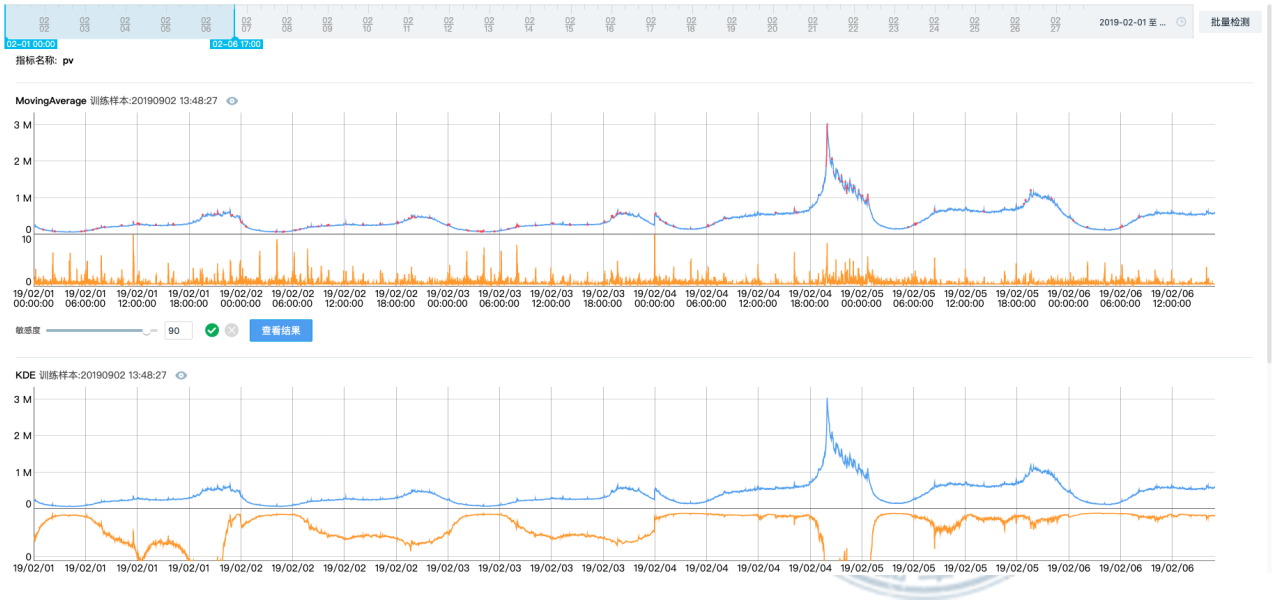
· 拓扑图：拖拽方式自定义业务拓扑和数据流转关系图，各拓扑节点支持按照业务关系分组，支持分区展示业务状态统计指标，支持点击钻取和定时刷新。

5.7. 人工智能

- 机器学习平台：日志易提供 20 多种标准机器学习算法，并支持在界面化交互平台上探索机器学习算法的数据运用。日志易按照数值预测、分类预测、离群检测、聚类分析、时序预测五个场景分类，提供不同的模型测试效果评价和可视化。
- 日志模式：日志易针对非结构化日志分析场景，提供专属的模式学习智能算法。方便用户快速了解百万行日志内含的行为模式和趋势，快速定位和发现未知故障。



- 指标异常检测：日志易针对 IT 监控指标的异常检测场景，提供多种专属的聚类和异常检测智能算法，并支持数据接入后自动选择和训练检测。用户也可以根据对比效果，手动进行参数调整并挑选最合适的模型来投入在线运用。


02-01 00:00
02-06 17:00
2019-02-01 至 ...
批量检测
指标名称: pv
MovingAverage 训练样本:20190902 13:48:27
3 M
2 M
1 M
0
10
0
19/02/01 00:00:00
19/02/01 06:00:00
19/02/01 12:00:00
19/02/01 18:00:00
19/02/02 00:00:00
19/02/02 06:00:00
19/02/02 12:00:00
19/02/02 18:00:00
19/02/03 00:00:00
19/02/03 06:00:00
19/02/03 12:00:00
19/02/03 18:00:00
19/02/04 00:00:00
19/02/04 06:00:00
19/02/04 12:00:00
19/02/04 18:00:00
19/02/05 00:00:00
19/02/05 06:00:00
19/02/05 12:00:00
19/02/05 18:00:00
19/02/06 00:00:00
19/02/06 06:00:00
19/02/06 12:00:00
敏感度
90
查看结果
KDE 训练样本:20190902 13:48:27
3 M
2 M
1 M
0
0
19/02/01
19/02/02
19/02/03
19/02/04
19/02/05
19/02/06

微信公众号：日志易

扫描预约专家交流

北京优特捷信息技术有限公司

电话：400-085-0159

E-mail：contact@yottabyte.cn

官网：https://www.rizhiyi.com