Operations

Waterdrop

— 非常易用，高性能的实时、离线数据处理产品

InterestingLab

2018年9月

https://github.com/InterestingLab/waterdrop

个人介绍

Operations

sina新浪

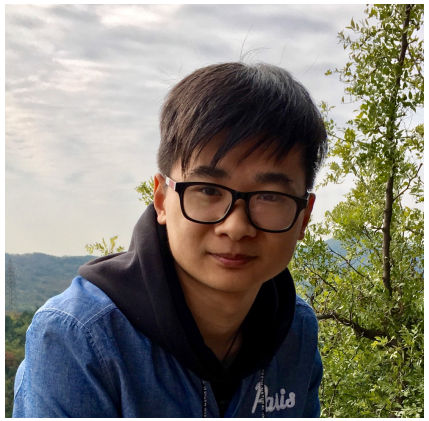
霍晨（Ricky Huo）

- 大数据研发工程师
- 新浪运维中心 - 数据分析平台
- 主要负责数据实时处理以及数据平台开发

关于我们

Operations

InterestingLab

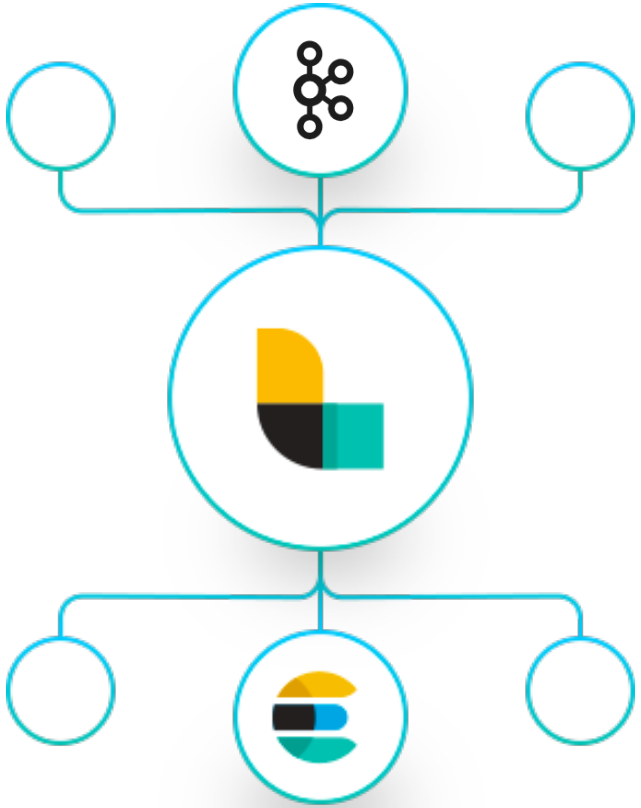
@ Gary Gao	@ Ricky Huo	@ Kevin Xu	@ Kid Xiong
			
yixia.com 一下科技 乐视云 新浪	新浪	滴滴 360 www.360.cn 新浪	yixia.com 一下科技

目录

数据平台架构演进

01

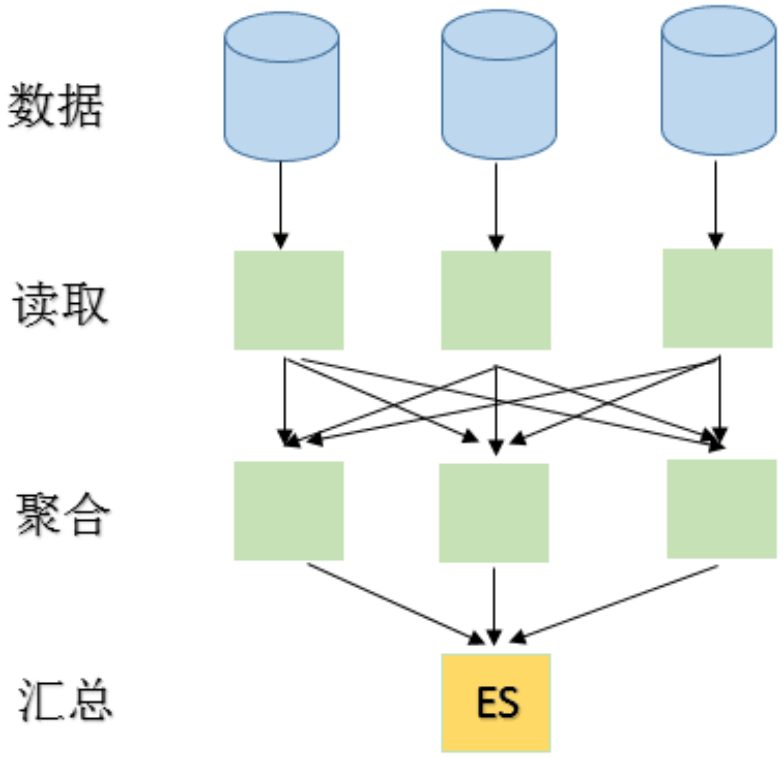
V1（Logstash）



V1遇到的线上挑战

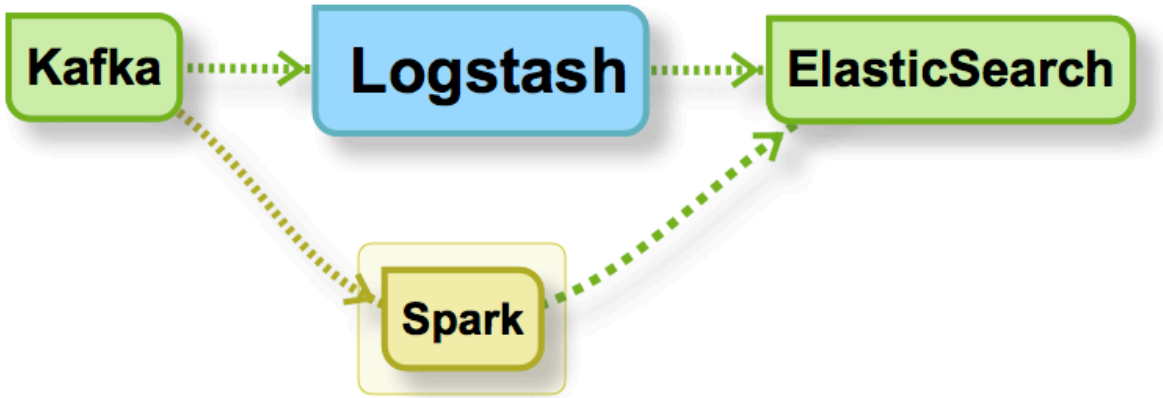
红包飞服务质量监控

- 日志量大
- 多维分析
- 资源有限
- 宏观指标



V2（Logstash+Spark）

Operations



Spark的引入带来的问题

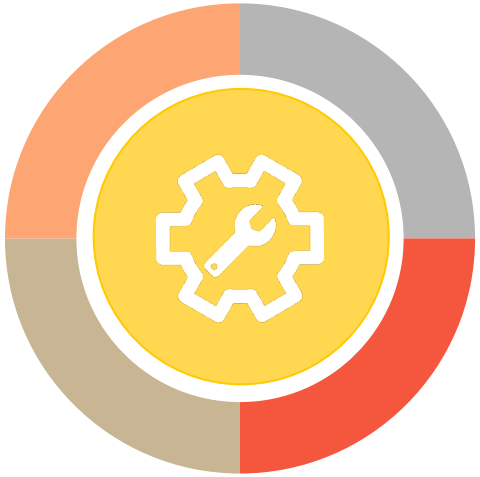
Operations

学习成本

启动一个Spark应用需要熟悉Spark提供的方法，了解常用的优化手段，才能完成一个高效的应用。

任务繁重，如何快速上线

随着业务和需求的增长，如何在有限的人力成本下快速地提供稳定且高效的数据处理服务



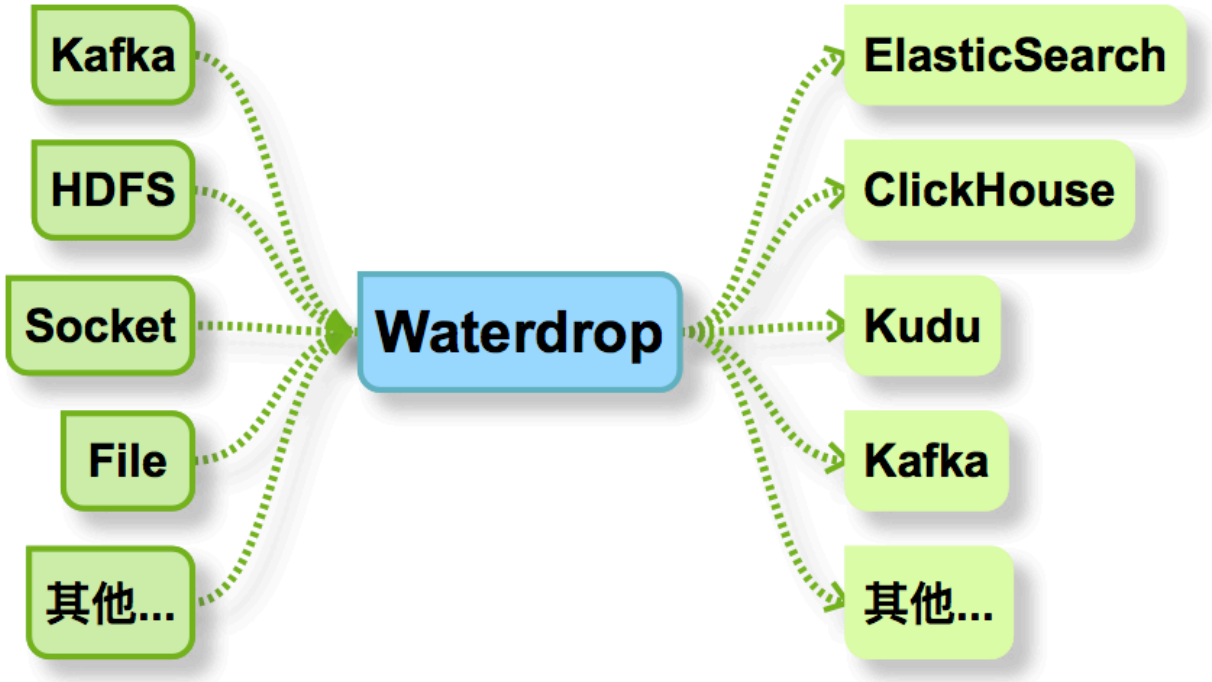
太多重复代码

每个人都有自己的编码习惯，每个人对接的业务需求不同。导致项目代码难以管理且难以复用。

线上应用的管理

应用的管理和监控给运维人员造成了不小的困扰。

V3（Waterdrop）



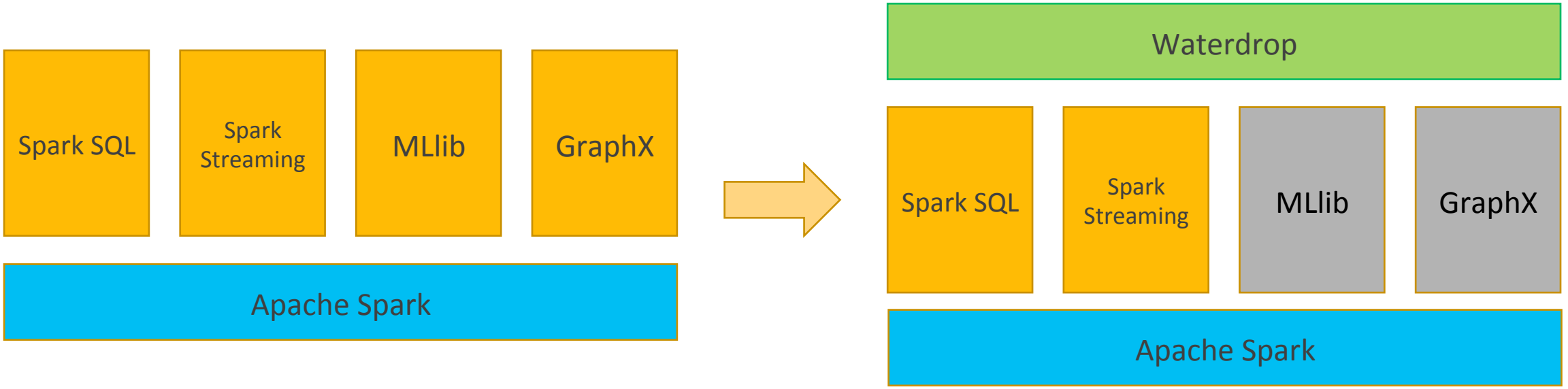
什么是Waterdrop

02

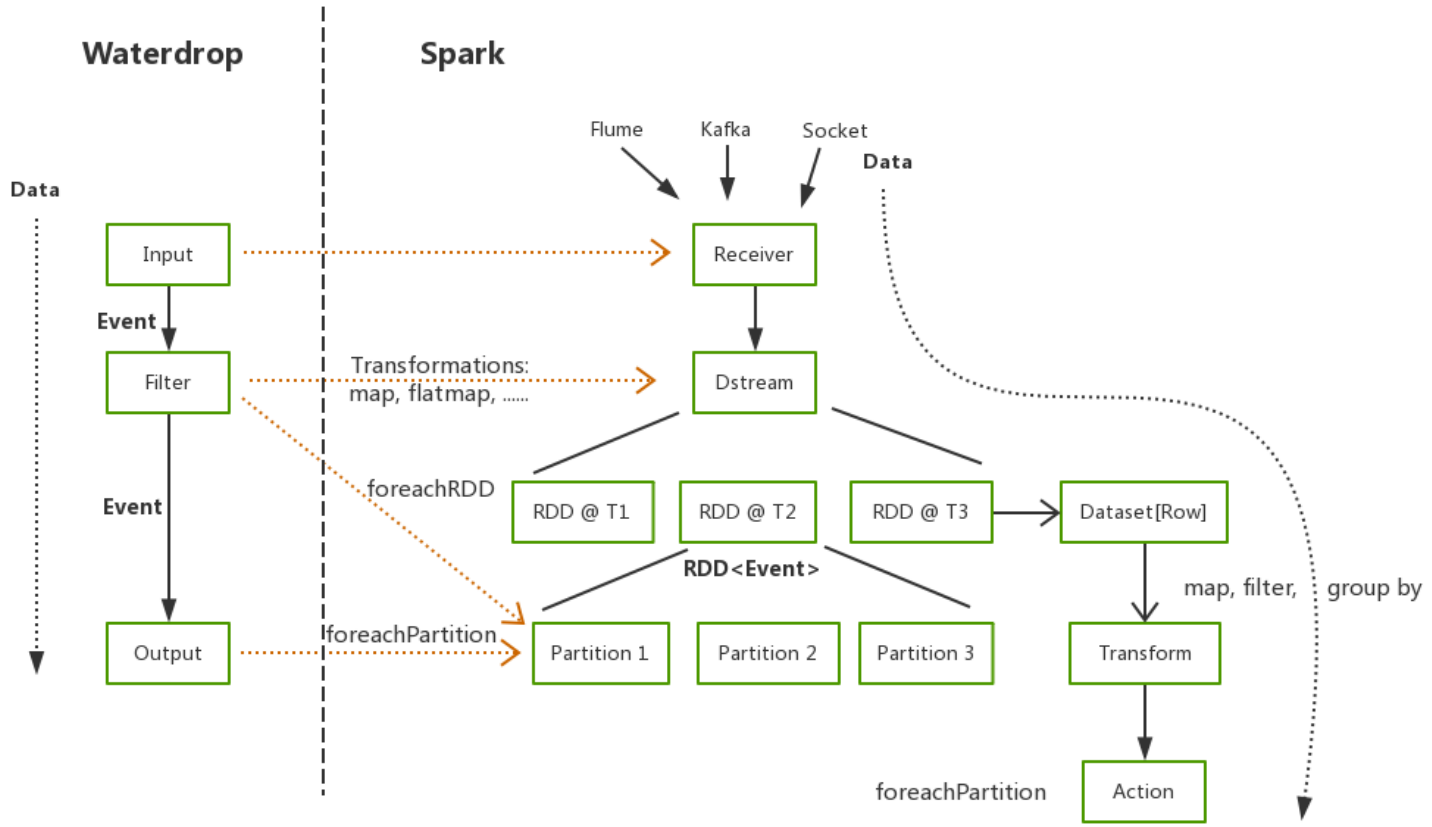
什么是Waterdrop

Operations sina新浪

Waterdrop是一个非常易用、高性能，能够应对海量数据的数据处理产品，构建于Apache Spark之上



Waterdrop 架构



整个架构主要由三部分组成：

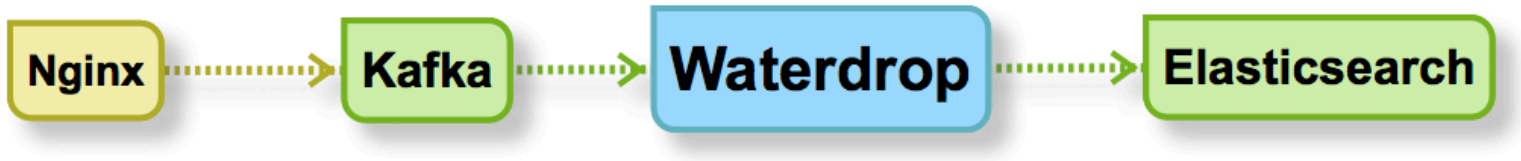
1. Input
2. Filter
3. Output

多个Filter构建了数据处理的Pipeline，满足各种各样的数据处理需求。

Waterdrop 使用场景

03

场景1：Nginx日志入Elasticsearch



Logstash实现

Operations 新浪

```
input {
    kafka {
        topics => ["waterdrop_nginx_demo"]
        bootstrap_servers => ["localhost:9092"]
        group_id => "logstash_nginx_demo"
        auto_offset_reset => "latest"
    }
}
filter {
    grok {
        match => {
            message => "%{IP:ha_ip}\\s%{NOTSPACE:domain}\\s%{IP:remote_addr}\\s%{NUMBER:request_time}s\\s\"%
{DATA:upstream_ip}\"\\s\\[%{HTTPDATE:datetime}\\]\\s\"%{NOTSPACE:method}\\s%{DATA:url}\\s%
{NOTSPACE:http_ver}\"\\s%{NUMBER:status}\\s%{NUMBER:body_bytes_send}\\s%{DATA:referer}\\s%
{NOTSPACE:cookie_info}\\s\"%{DATA:user_agent}\"\\s%{DATA:uid}\\s%{DATA:session_id}\\s\"%{DATA:pool}\"\\s%
{DATA:city_code}\\s%{DATA:tag3}\\s\"%{DATA:tag4}\""
        }
    }
    date {
        match => ["datetime", "dd/MMM/yyyy:HH:mm:ss Z"]
    }
}
output {
    elasticsearch {
        hosts => ["localhost:9200"]
    }
}
```

Waterdrop实现

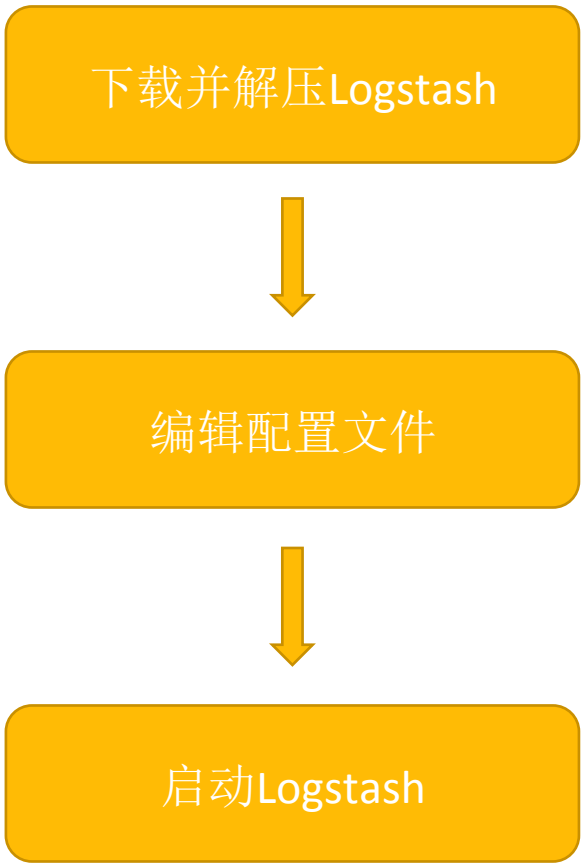
Operations 新浪

```
input {
    kafkaStream {
        topics = "waterdrop_nginx_demo"
        consumer.bootstrap_servers = "localhost:9092"
        consumer.zookeeper.connect = "localhost:2181"
        consumer.group.id = "logstash_nginx_demo"
    }
}
filter {
    grok {
        source_field = "raw_message"
        pattern = "%{IP:ha_ip}\\s%{NOTSPACE:domain}\\s%{IP:remote_addr}\\s%{NUMBER:request_time}s\\s\"%
{DATA:upstream_ip}\"\\s\\[%{HTTPDATE:datetime}\\]\\s\"%{NOTSPACE:method}\\s%{DATA:url}\\s%
{NOTSPACE:http_ver}\"\\s%{NUMBER:status}\\s%{NUMBER:body_bytes_send}\\s%{DATA:referer}\\s%
{NOTSPACE:cookie_info}\\s\"%{DATA:user_agent}\"\\s%{DATA:uid}\\s%{DATA:session_id}\\s\"%{DATA:pool}\"\\s%
{DATA:city_code}\\s%{DATA:tag3}\\s\"%{DATA:tag4}\""
        }
    }
    date {
        "source_field" = "datetime"
        "target_field" = "timestamp"
        "source_field_format" = "dd/MMM/yyyy:HH:mm:ss Z"
        "target_field_format" = "yyyy/MM/dd HH:mm:ss"
    }
}
output {
    elasticsearch {
        hosts = ["localhost:9200"]
        index = ["waterdrop"]
    }
}
```

Waterdrop 部署方式

Operations

Logstash



Waterdrop

https://interestinglab.github.io/waterdrop/#/zh-cn/quick-start

场景2：聚合指标入ElasticSearch

数据聚合需求

1. 统计每个域名在每个机房的各状态码访问情况（使用Waterdrop的SQL插件）

```
sql {
    table = “nginx_info”
    sql = “select count(1)  as count, status, idc, domain, datetime from nginx_info group by domain, status, datetime, idc”
}
```

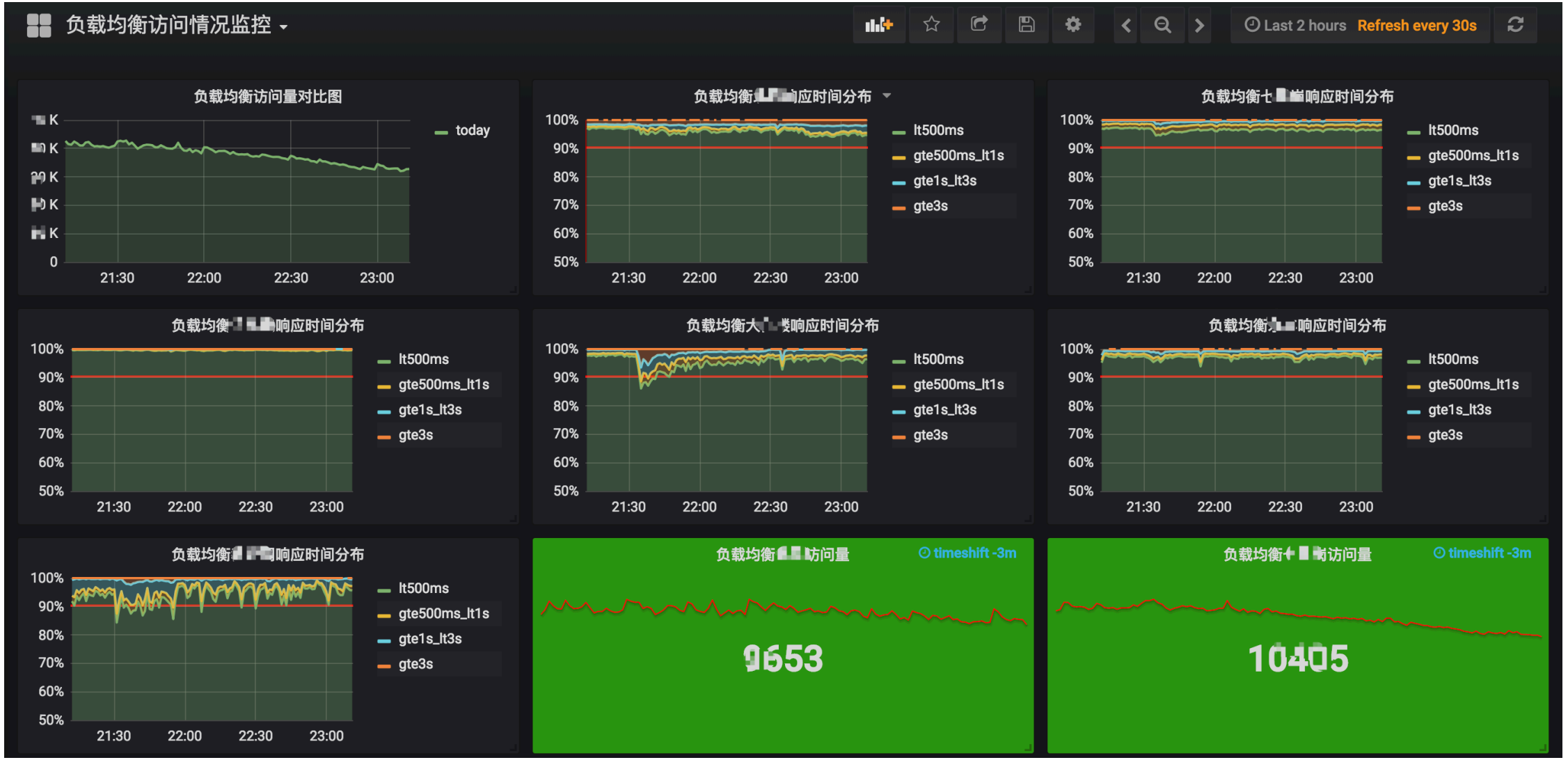
2. 统计每个域名在每个机房的各响应时间段的次数（使用Waterdrop的SQL插件）

```
sql {
    table = “nginx_info”
    sql = “select sum(int(request_time < 0.03)) as lt30, sum(int((request_time >= 0.03) AND (request_time < 0.1)))  as lt100, sum(int(request_time >= 0.1)) as gt100, domain, idc, datetime from nginx_info group by domain, idc, datetime”
}
```

聚合结果展示

Operations 新浪

聚合结果展示


负载均衡访问情况监控
Last 2 hours
Refresh every 30s
负载均衡访问量对比图
today
21:30
22:00
22:30
23:00
lt500ms
gte500ms_lt1s
gte1s_lt3s
gte3s
timeshift -3m
9653
10405

Why SQL

Operations

1. SQL简单易用

2. 提供基于SQL的ETL

- select int(status) from nginx_info …
- select to_date(datetime, 'yyyy-MM-dd’) from nginx_info …
- select * from nginx_info where domain = ‘sina.com.cn’ …

3. 提供基于SQL数据聚合

- select count(1) as c, status, idc, domain, datetime from nginx_info group by domain, status, datetime, idc

场景3：高度定制化场景

Waterdrop部分插件列表

Operations

新浪

Input

- File
- HDFS
- Kafka
- Socket
- S3
- ...

01

Filter

- SQL
- JSON
- Grok
- Sample
- Replace
- ...

02

Output

- Elasticsearch
- Kafka
- HDFS
- ClickHouse
- MySQL
- ...

03

https://interestinglab.github.io/waterdrop/#/zh-cn/

反抓站

针对实时数据，定期聚合数据，提取特征判断抓站行为

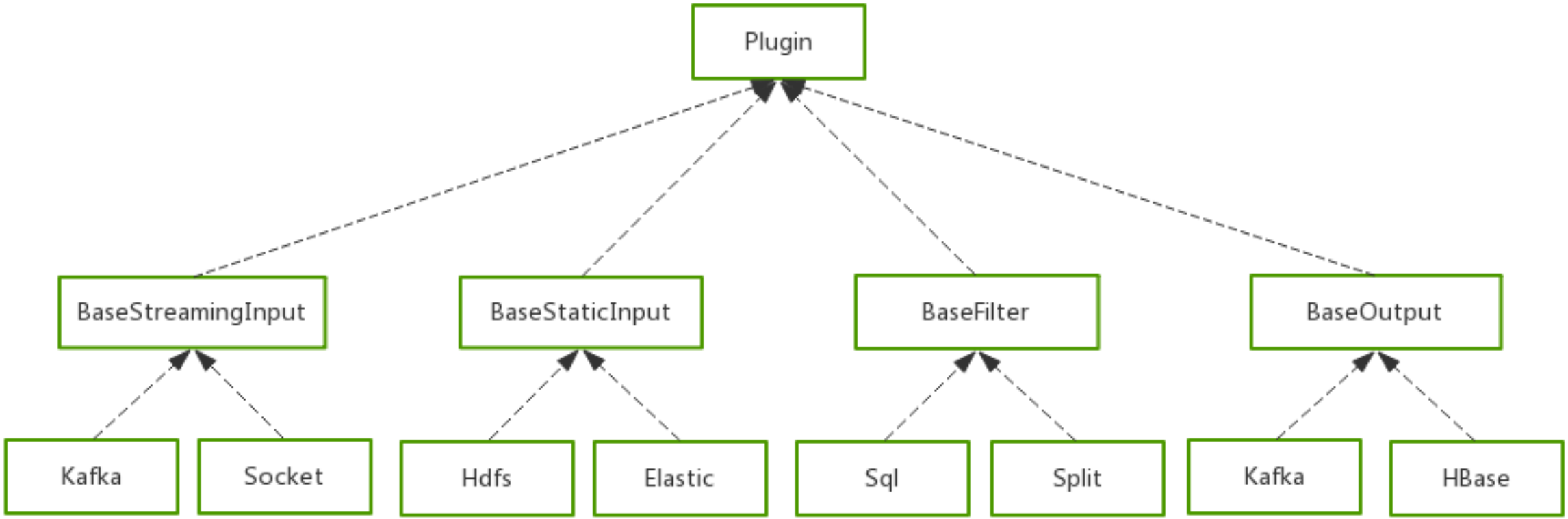
安全部门

- 不了解大规模数据实时处理方案
- 学习Spark并投入使用具有一定成本

数据平台

- 不了解抓站特征提取机制
- 应用从测试到上线需要不断调试

Waterdrop 插件体系



BaseFilter

Operations sina新浪

```scala
abstract class BaseFilter extends Plugin {

  def process(spark: SparkSession, df: Dataset[Row]): Dataset[Row]

  /**
   * Allow to register user defined UDFs
   * @return empty list if there is no UDFs to be registered
   * */
  def getUdfList(): List[(String, UserDefinedFunction)] = List.empty

  /**
   * Allow to register user defined UDAFs
   * @return empty list if there is no UDAFs to be registered
   * */
  def getUdafList(): List[(String, UserDefinedAggregateFunction)] = List.empty
}
```

Substring实现

```
org.interestinglab.waterdrop.filter.ScalaSubstring {
    source_field = "message"
    target_field = "sub"
    pos = 1
    len = 3
}
```

获取配置中的变量

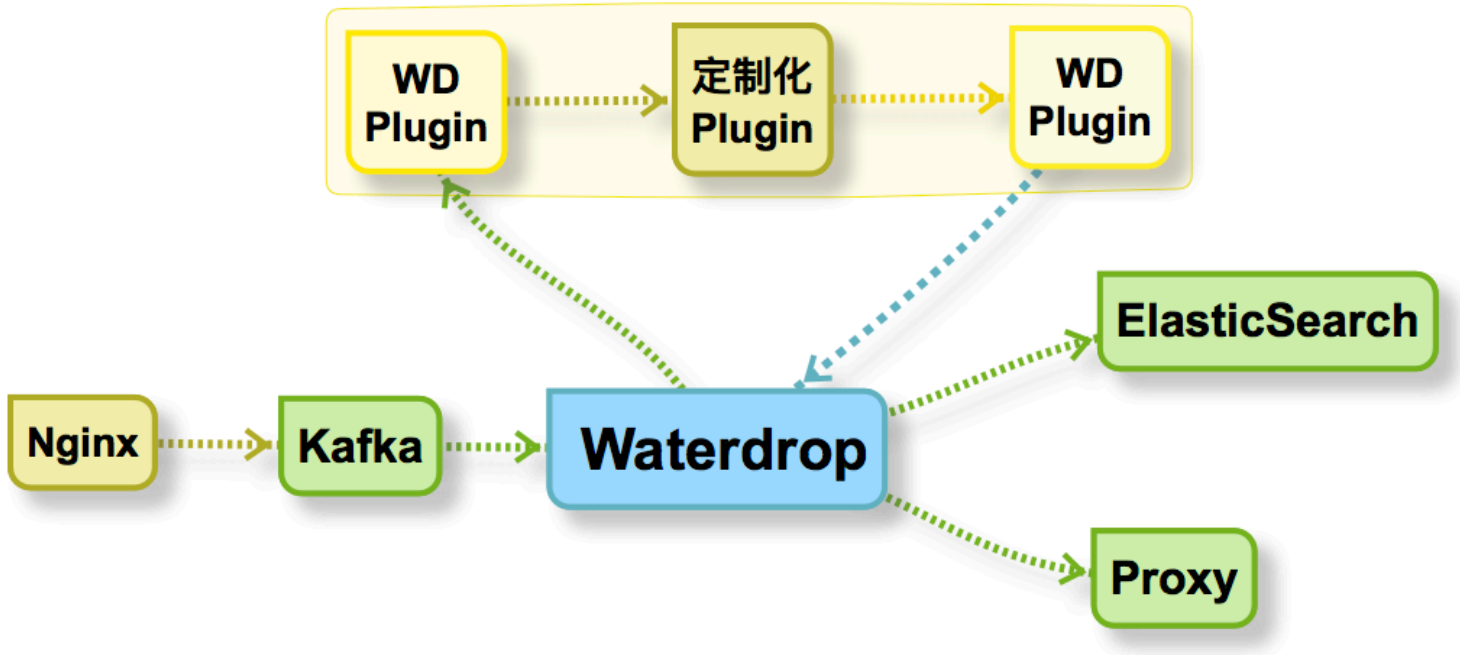
自定义方法（UDF）

应用到每一条数据

```
override def process(spark: SparkSession, df: Dataset[Row]): Dataset[Row] = {

  val srcField = config.getString("source_field")
  val targetField = config.getString("target_field")
  val pos = config.getInt("pos")
  val len = config.getInt("len")

  val func = udf((s: String) => s.substring(pos, pos+len))

  df.withColumn(targetField, func(col(srcField)))
}
```

更多插件使用方法参照：https://github.com/InterestingLab/waterdrop-example

反抓站与插件体系


WD Plugin
定制化 Plugin
WD Plugin
ElasticSearch
Nginx
Kafka
Waterdrop
Proxy

场景4：离线处理场景

实时离线统一融合

Operations 新浪

Streaming Input

```
input {
  kafkaStream {
    topics = "waterdrop"
    consumer.bootstrap.servers = "localhost:9092"
    consumer.zookeeper.connect = "localhost:2181"
    consumer.group.id = "waterdrop_group"
  }
}
```

Static Input

```
input {
  hdfs {
    path = "hdfs://m2.8022/waterdrop-logs/access.log"
    format = "json"
  }
}
```

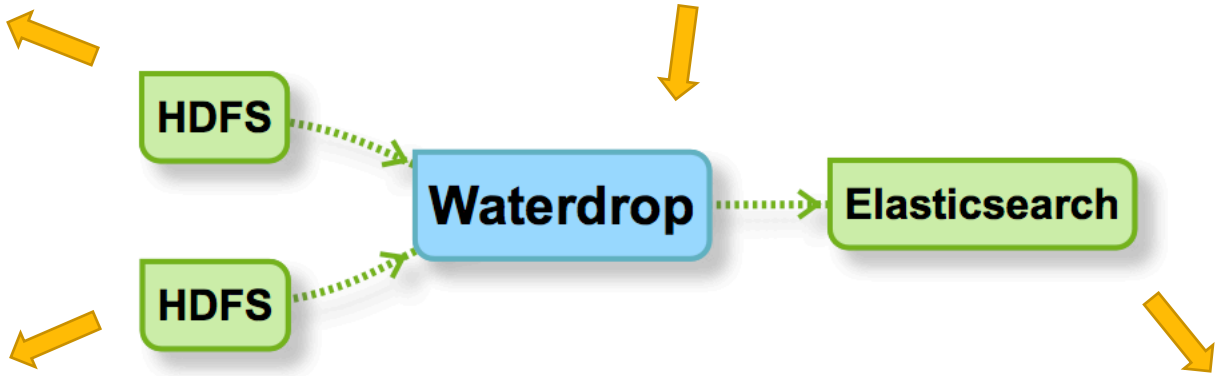
流式处理与离线处理入口统一，改变**input**即可自动切换到离线处理

离线多数据源Join

Operations 新浪

consumer_id	first_name	last_name	address
1	George	Jefferson	3200 Mt Vemon Hwy
2	John	Adams	1250 Hancock St
3	Thomas	Madison	931 Tomas Pkwy
4	James	Monroe	11350 Constitution Hwy

```
select first_name, last_name, order_date, order_amount
from customers c
inner join orders o
on c.customer_id = o.customer_id
```



order_id	order_date	amount	costomer_id
1	07/04/1776	$234.56	1
2	03/14/1760	$78.50	3
3	05/23/1784	$124.00	2
4	09/03/1790	$65.50	3
5	11/27/1787	$14.40	10

first_name	last_name	order_date	order_amount
George	Washington	07/04/1776	$234.56
John	Adams	05/23/1784	$124.00
Thomas	Madison	03/14/1760	$78.50
Thomas	Madison	09/03/1790	$65.50

More Complex ?

https://github.com/InterestingLab/waterdrop/issues

数据平台遇到的问题

Operations

sina新浪

不支持海量计算（数据转换或聚合）

数据处理学习成本较高

业务、工程代码复用率低

如何快速ETL、快速上线

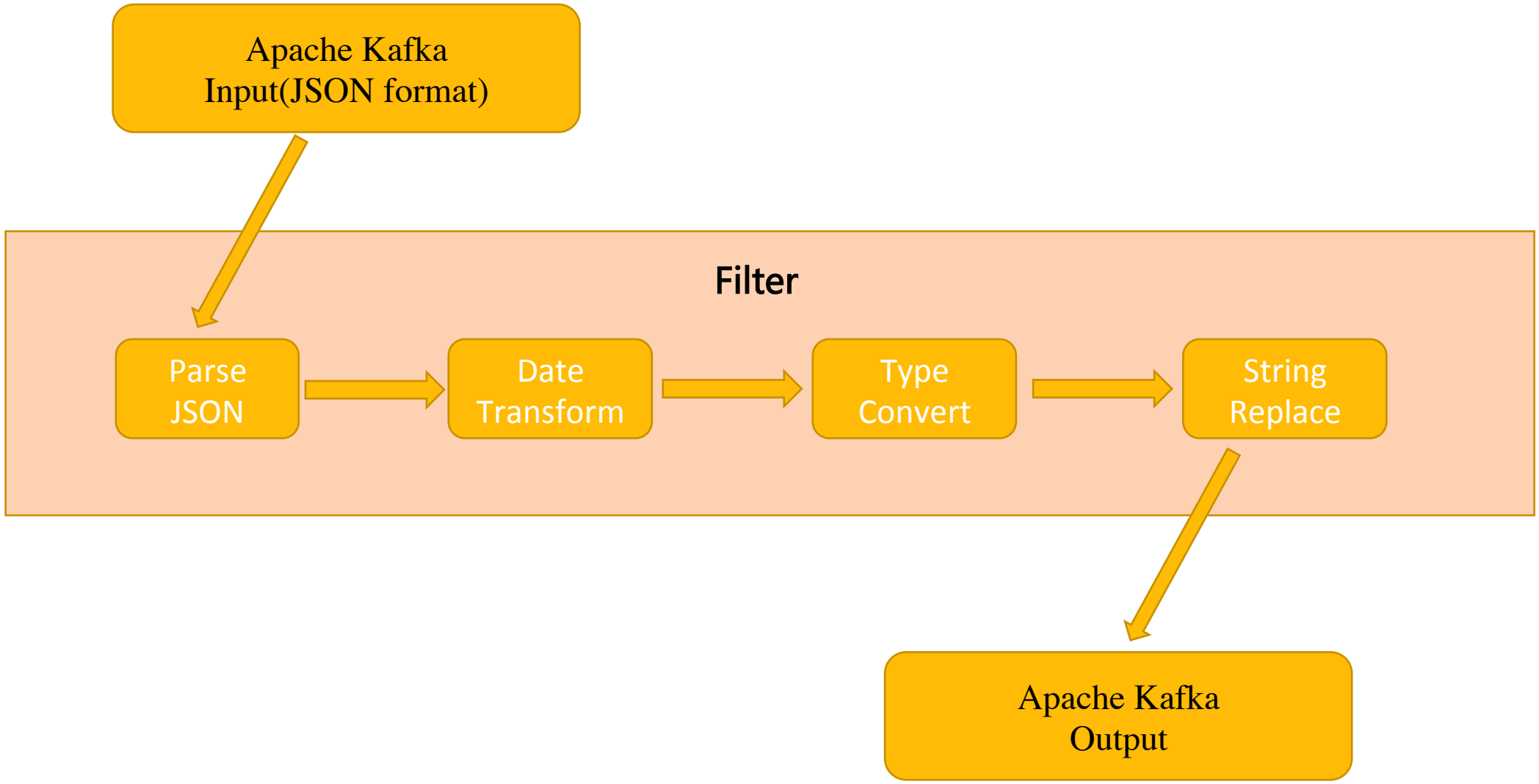
服务难以监控和保障

- Waterdrop底层使用Spark这个分布式计算框架作为计算引擎，并使用SQL支持聚合功能。
- Waterdrop高度配置化，并且本身提供了丰富的插件，无需了解Spark就可轻松上手。
- Waterdrop中离线实时统一融合。
- Waterdrop插件化体系，保证用户多为业务逻辑操心，少为流程和容错困扰。
- 子产品 Guardian，自动拉起异常退出spark程序，自动监控报警streaming处理延迟

Waterdrop 性能对比

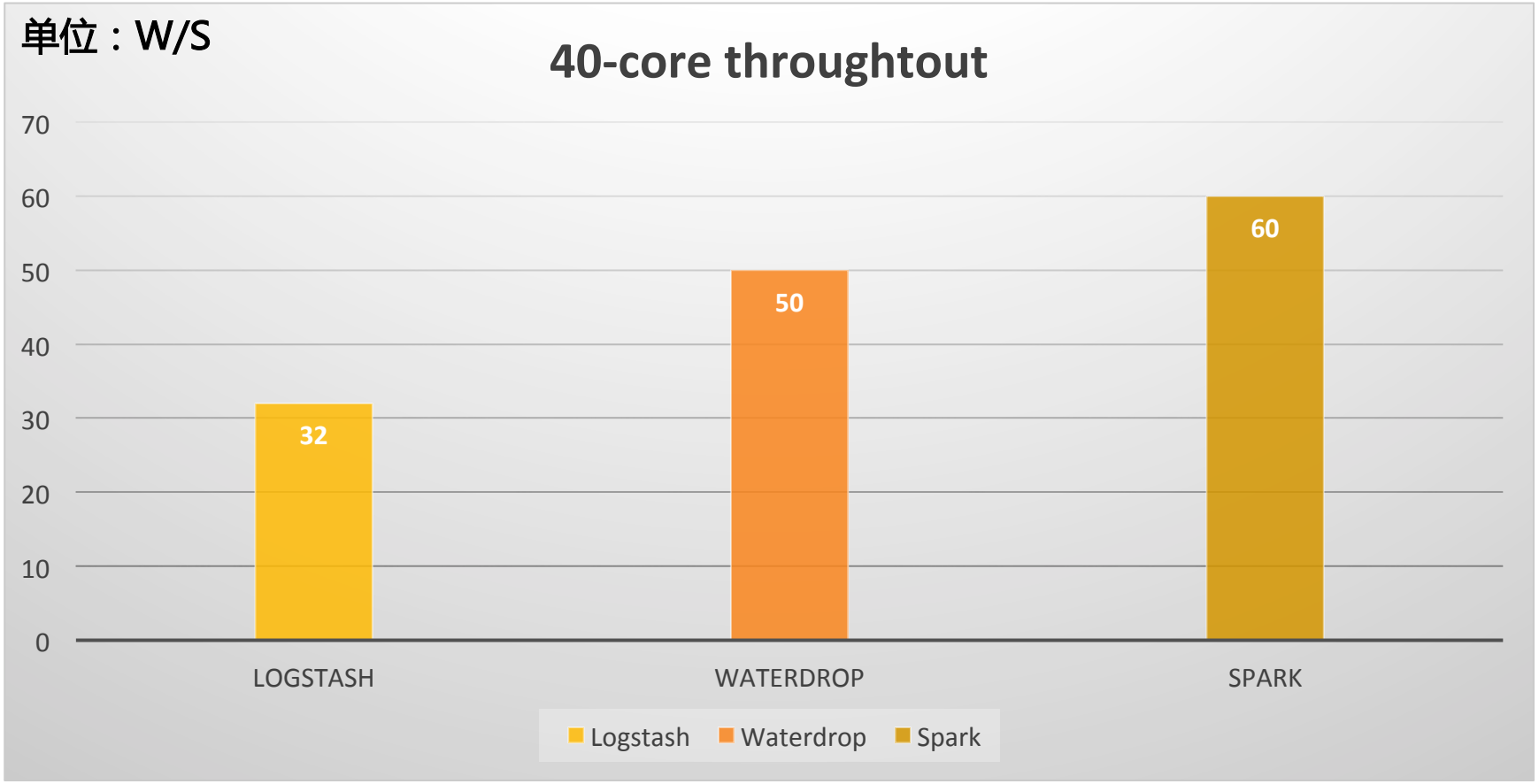
04

Waterdrop Benchmark



Waterdrop Benchmark

- Logstash 6.3.1
- Waterdrop 1.1.0
- Spark 2.2.0



Waterdrop Roadmap

05

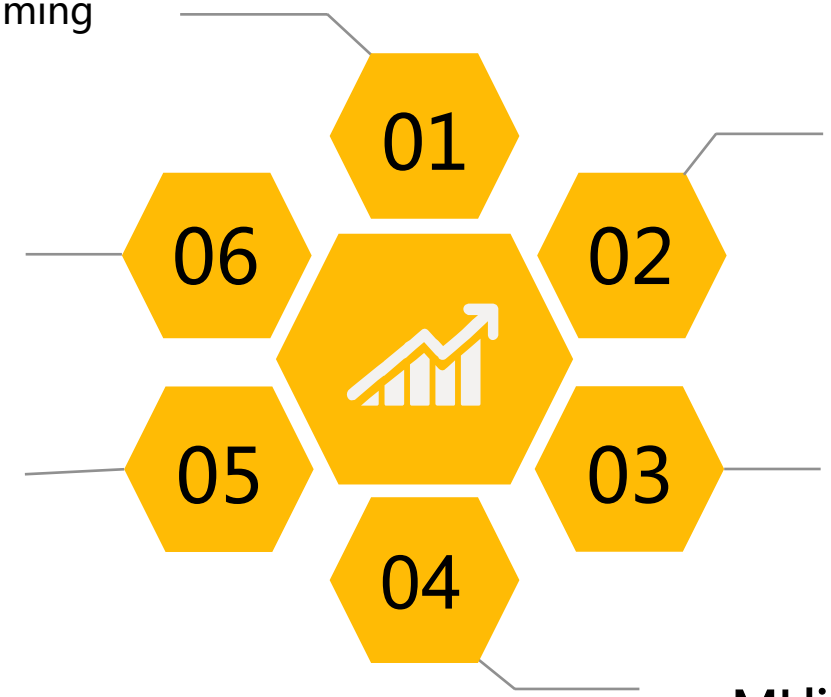
Roadmap

Operations

新浪

Structured Streaming

Spark Structured Streaming



更多的插件和处理逻辑

多pipleline 动态处理逻辑

数据处理平台化

可视化UI，血缘管理

插件性能统计和优化

Waterdrop性能和各插件耗时统计

Flink

支持Flink框架作为计算引擎

MLlib

流式机器学习

感谢Elastic，祝Elastic社区越来越好

本次分享由Interesting Lab出品

https://github.com/InterestingLab/waterdrop

elastic
中文社区

专业、垂直、纯粹的 Elastic 开源技术交流社区
https://elasticsearch.cn/