大数据基础与应用
BIG DATA FOUNDATION AND APPLICATION

大数据开源平台与工具

北京理工大学计算机学院　王一拙

2019年1月

内容提要

- 数据采集与清洗
- 数据存储与管理
- 数据处理与分析
- 资源管理与调度

1. 数据采集与清洗

- 数据来源和获取技术
- 什么是数据清洗

- 数据采集的开源工具
  - Scrapy爬虫框架
  - Flume日志采集系统
  - Kafka消息中间件
  - Sqoop工具（RDBMS<->Hadoop）

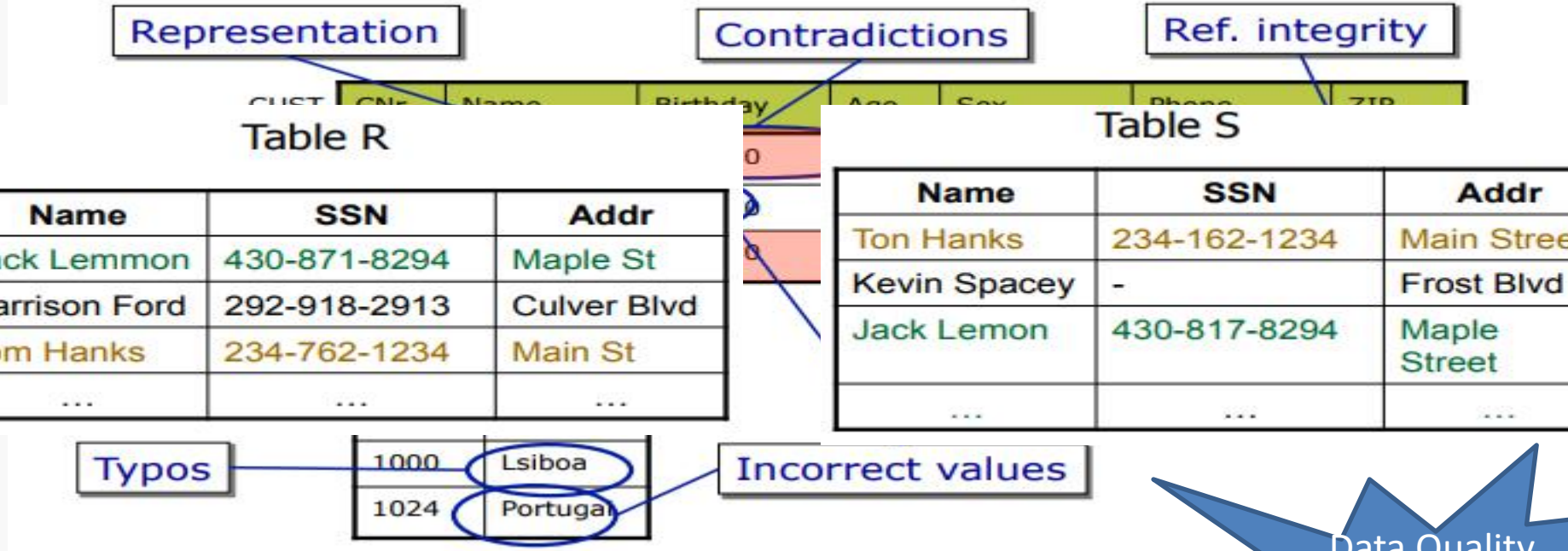
大数据的来源

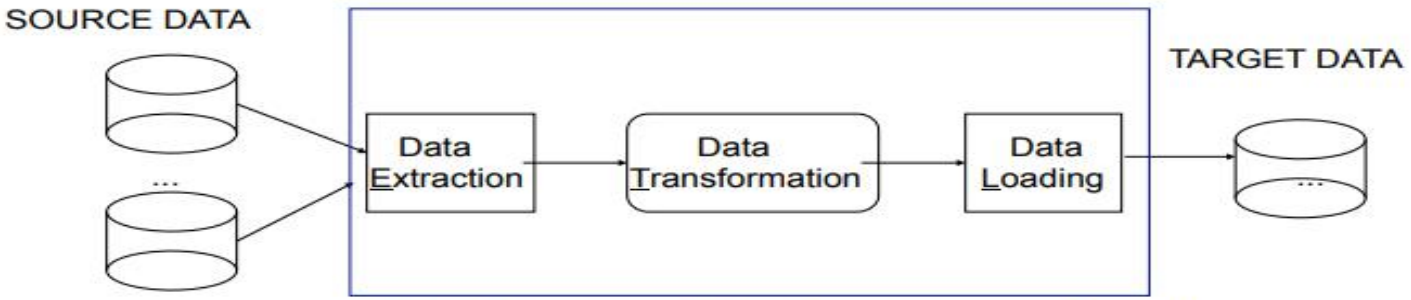
https://www.allerin.com/blog/top-5-sources-of-big-data

数据获取技术

- 传感器
- 网络爬虫
- 网站公开API
- 日志收集工具
- 底层数据采集引擎
- 中间件系统
- 关系数据库
- ……

什么是数据清洗？





ETL: Extraction, Transformation and Loading

https://fenix.tecnico.ulisboa.pt/downloadFile/563568428715919/7_DataCleaningIntro.pdf

大数据基础与应用 BIG DATA FOUNDATION AND APPLICATION

爬虫

- Python简单爬虫程序实现
  - urllib,urllib2(Python2.7)

```
import urllib
import re

def getHtmlContent(url):
    page = urllib.urlopen(url)
    return page.read()

def getJPGs(html):
    jpgReg = re.compile(r'<img.+?src="(.+?\.jpg)" width')
    jpgs = re.findall(jpgReg, html)
    return jpgs
```

https://blog.csdn.net/mr_blued/article/details/79180017

```
def downloadJPG(imgUrl, fileName):
    urllib.urlretrieve(imgUrl, fileName)

def batchDownloadJPGs(imgUrls,
path='C:\\Users\\frank\\PycharmProjects\\untitled\\TieBaTu\\'):
    count = 1
    for url in imgUrls:
        downloadJPG(url, ''.join([path,
'{0}.jpg'.format(count)]))
        print 'number'+str(count)+'page'
        count = count + 1
```

```
def download(url):
    html = getHtmlContent(url)
    jpgs = getJPGs(html)
    batchDownloadJPGs(jpgs)

def main():
    url = 'http://tieba.baidu.com/p/2256306796'
    download(url)

if __name__ == '__main__':
    main()
```

https://blog.csdn.net/mr_blued/article/details/79180017

Fast and powerful

write the rules to extract the data and let Scrapy do the rest

Easily extensible

extensible by design, plug new functionality easily without having to touch the core

Portable, Python

written in Python and runs on Linux, Windows, Mac and BSD

用Scrapy创建简单爬虫

安装好Scrapy后，用下列命令创建一个项目：

```
$ scrapy startproject <your_project_name>
```

到新项目目录下创建一个爬虫：

```
$ cd <your_project_name>
$ scrapy genspider <name> <domain>
```

用Scrapy创建简单爬虫

上述两步完成后，项目目录下的spiders目录下会有所创建的爬虫程序文件：

```
import scrapy

class MySpider(scrapy.Spider):
    name = 'example.com'
    allowed_domains = ['example.com']
    start_urls = ['http://www.example.com/']

def parse(self, response):
    pass
```

```
tutorial/
    scrapy.cfg
    tutorial/
        __init__.py
        items.py
        pipelines.py
        settings.py
        spiders/
            __init__.py
            ...
```

用Scrapy创建简单爬虫

在大多数情况下，需要做的只是把数据和链接的提取代码添加到parse方法中：

```
def parse(self, response):
    for h3 in response.xpath('//h3').extract():
        yield {"title": h3}

    for url in response.xpath('//a/@href').extract():
        yield scrapy.Request(url, callback=self.parse)
```

用Scrapy创建简单爬虫

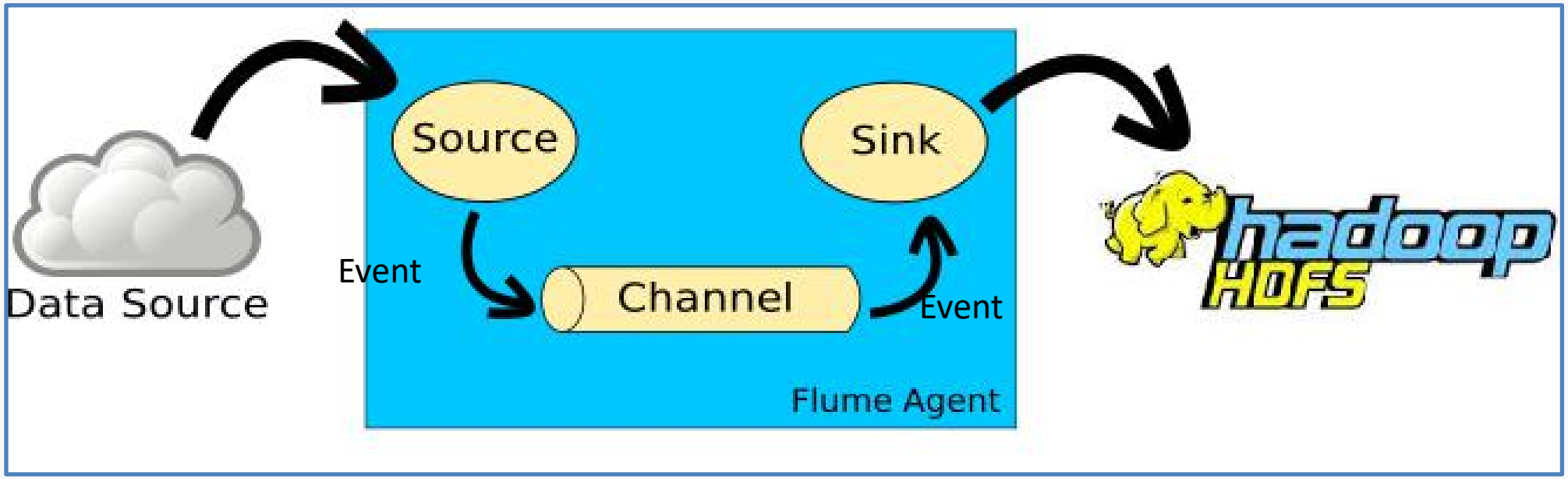
如果需要指定爬取的起始URLs，可以重写start_requests 方法：

```
def start_requests(self):
    yield scrapy.Request('http://www.example.com/1.html', self.parse)
    yield scrapy.Request('http://www.example.com/2.html', self.parse)
    yield scrapy.Request('http://www.example.com/3.html', self.parse)
```

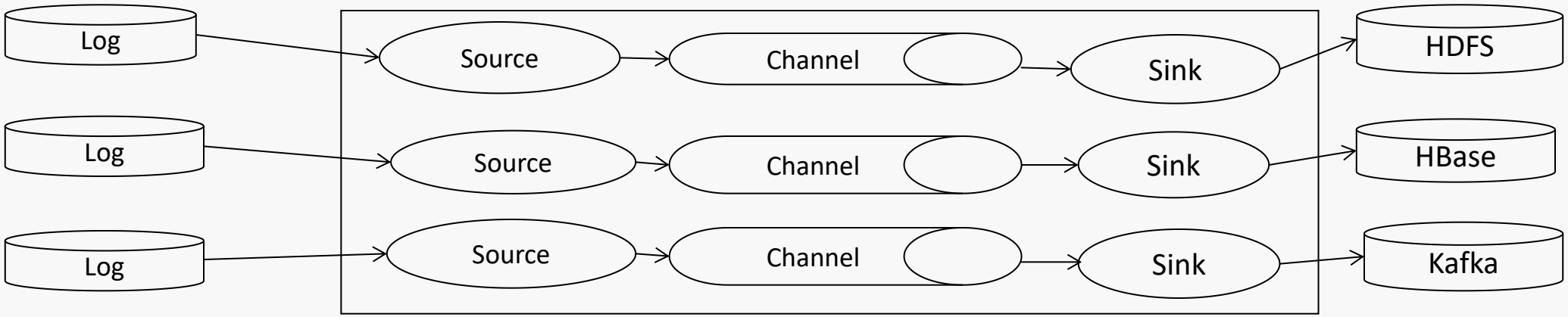
日志数据采集 Flume

- 由Cloudera公司开源
- 分布式、可靠和高可用的海量日志采集、聚合和传输的日志收集系统
- 数据源可定制、可扩展
- 数据存储系统可定制、可扩展
- 中间件：屏蔽了数据源和数据存储系统的异构性

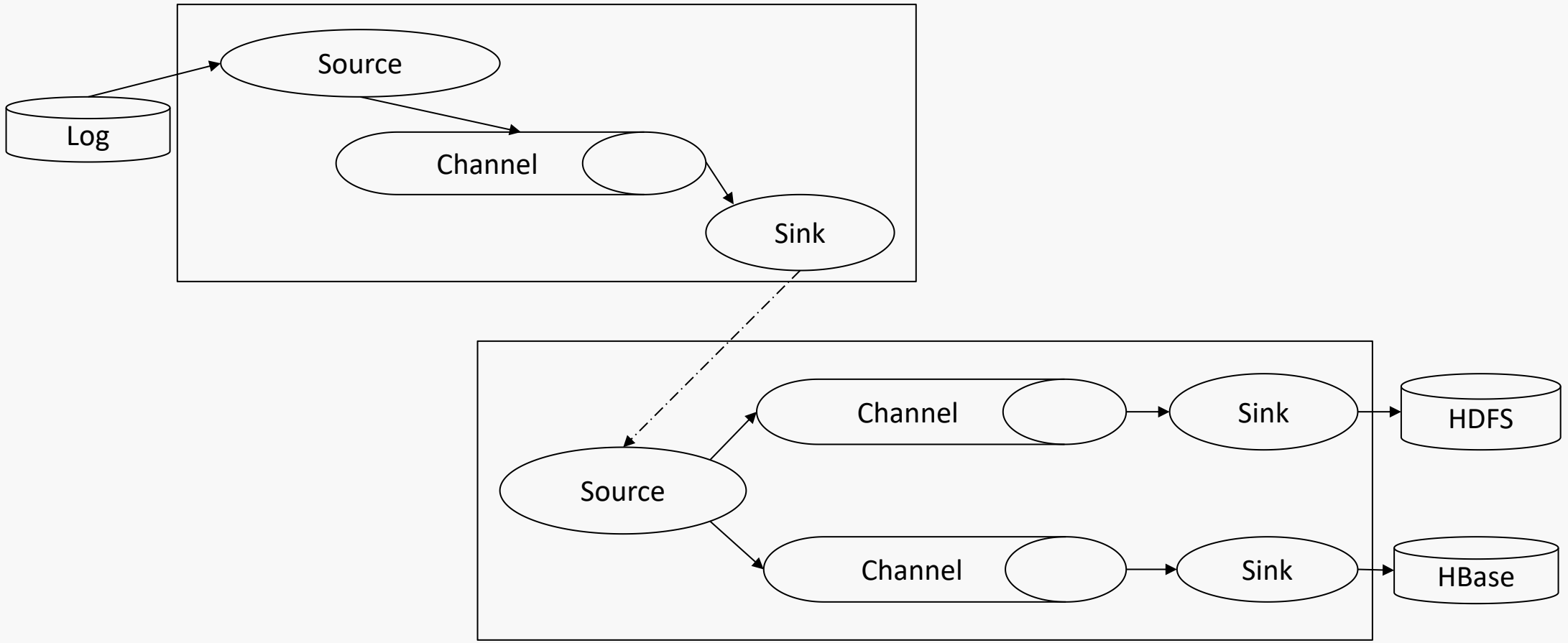
Flume NG基本架构



Flume支持采集日志文件



Flume支持多级级联和多路复制



Source类型

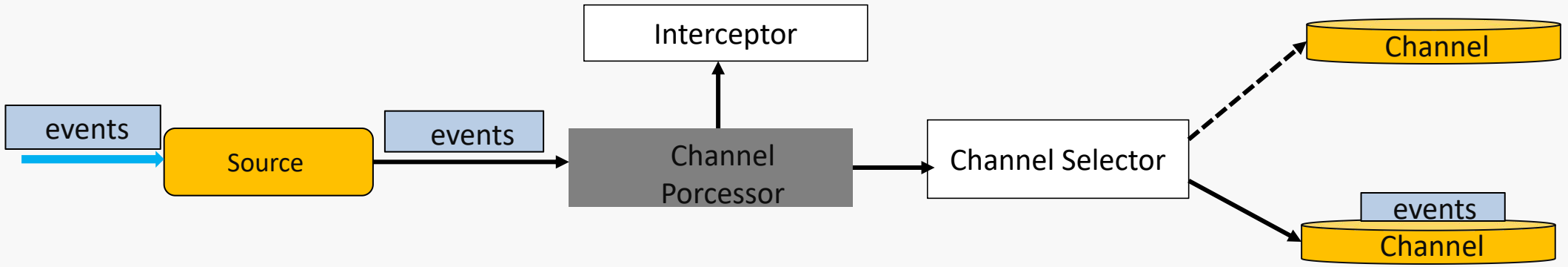
Source 类型	说明
exec source	执行某个命令或者脚本，并将其执行结果的输出作为数据源
avro source	提供一个基于avro协议的server，bind到某个端口上，等待avro协议客户端发过来的数据
thrift source	同avro，不过传输协议为thrift
http source	支持http的post发送数据
syslog source	采集系统syslog
spooling directory source	采集本地静态文件
jms source	从消息队列获取数据
Kafka source	从Kafka中获取数据

Sink类型

Sink类型	说明
hdfs sink	将数据写到hdfs上
avro sink	使用avro协议将数据发送给另下一跳的flume
thift sink	同avro，不过传输协议为thrift
file roll sink	将数据保存在本地文件系统中
hbase sink	将数据写到hbase中
Kafka sink	将数据写入到Kafka中
MorphlineSolr sink	将数据写入到Solr中

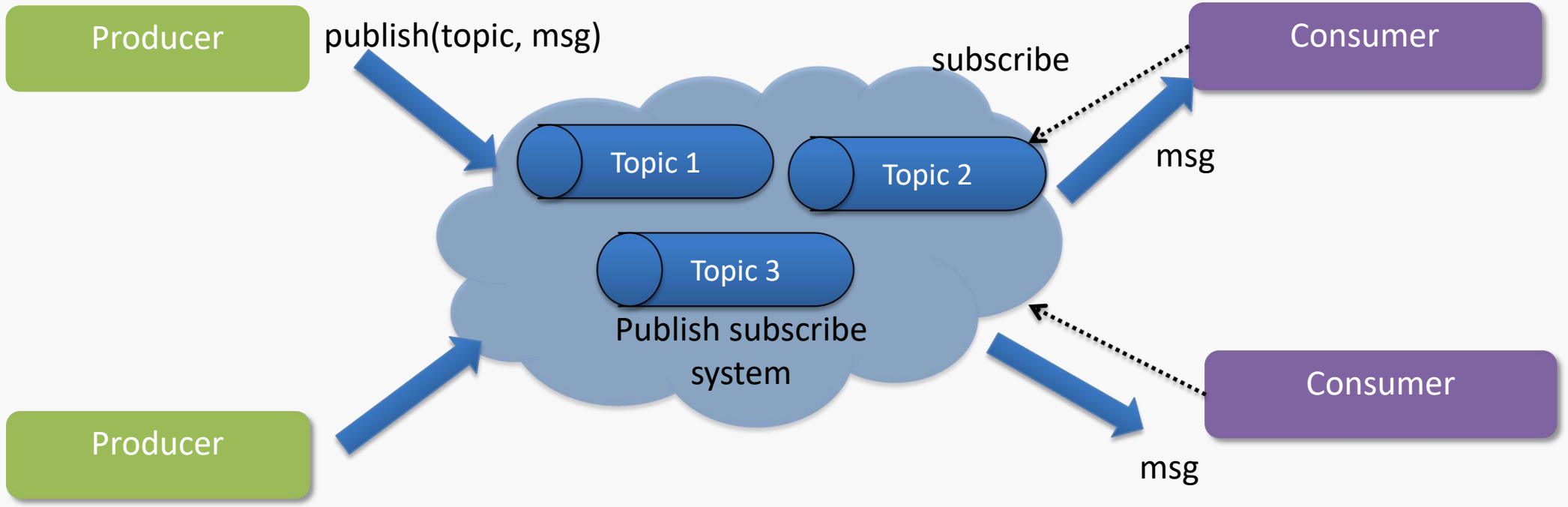
Flume传输过程中数据过滤

- Flume在传输数据过程中，可以简单的对数据过滤、清洗，去掉不关心的数据，如果需要对复杂的数据过滤，需要用户自己开发过滤插件，Flume支持第三方过滤插件调用。



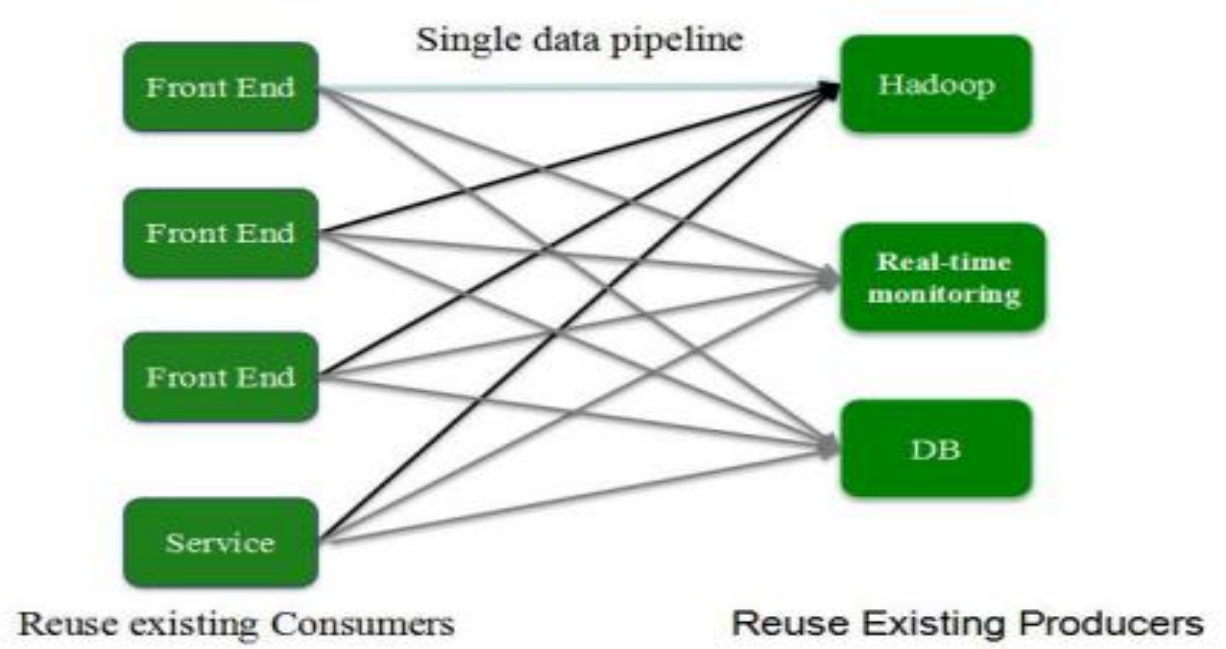
数据分发中间件Kafka

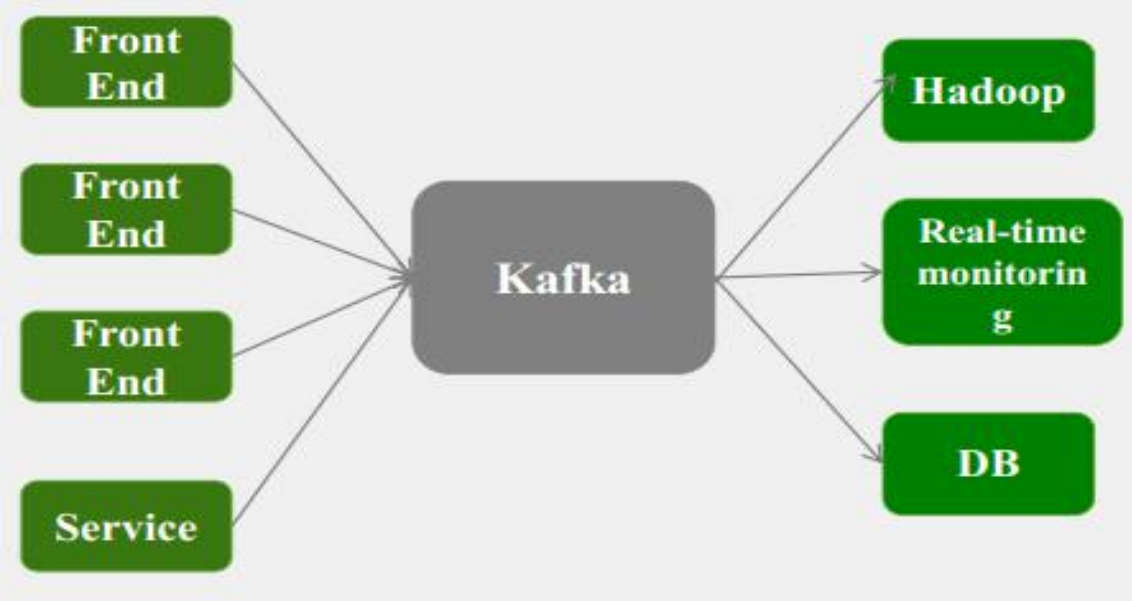
发布与订阅系统（PUB-SUB System）



Kafka简介

Kafka 是一个高吞吐、分布式、基于发布/订阅的消息系统，利用Kafka技术可在廉价PC Server上搭建起大规模消息系统。

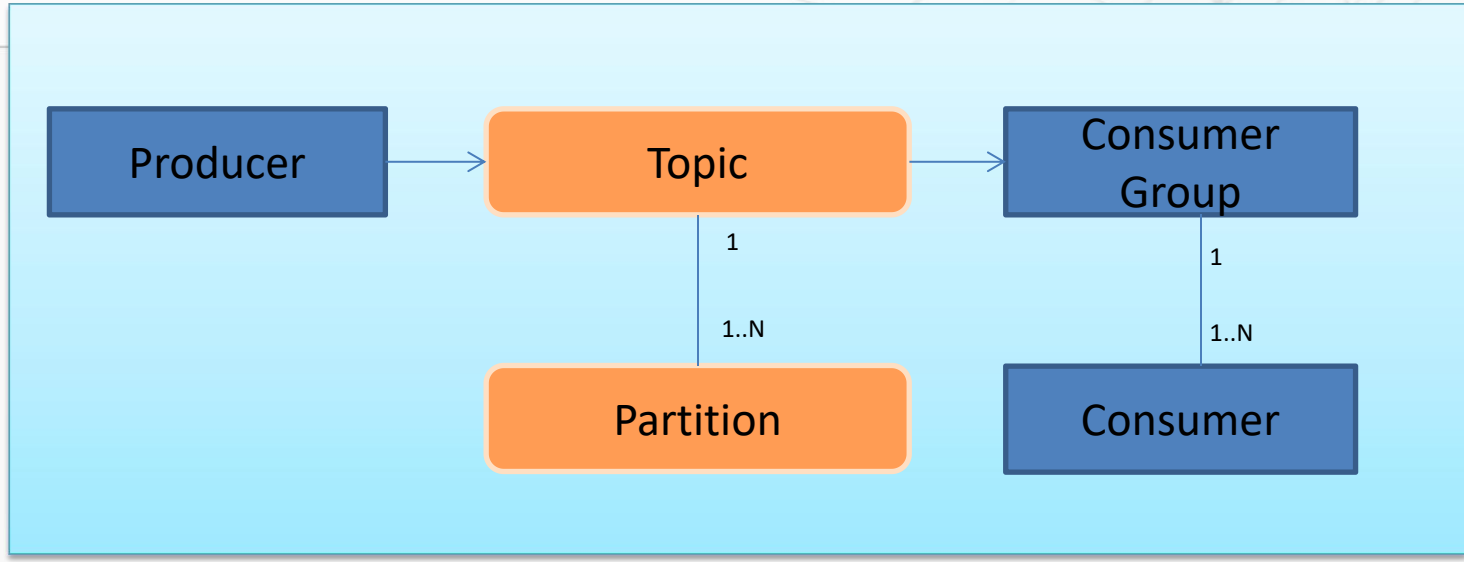




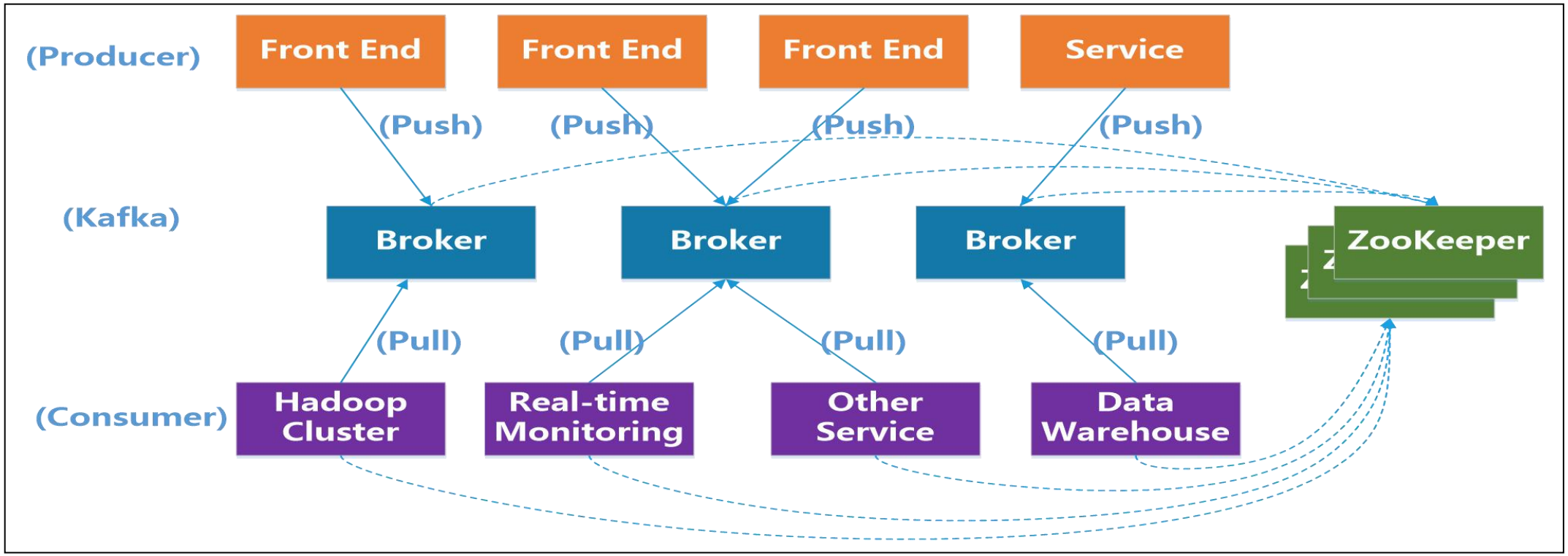
Kafka基本概念

- 基本概念
  - 主题：topic
  - 分区：partition
  - 消息：message

  - 生产者：producer
  - 消费者：consumer group
  - 缓存代理：broker

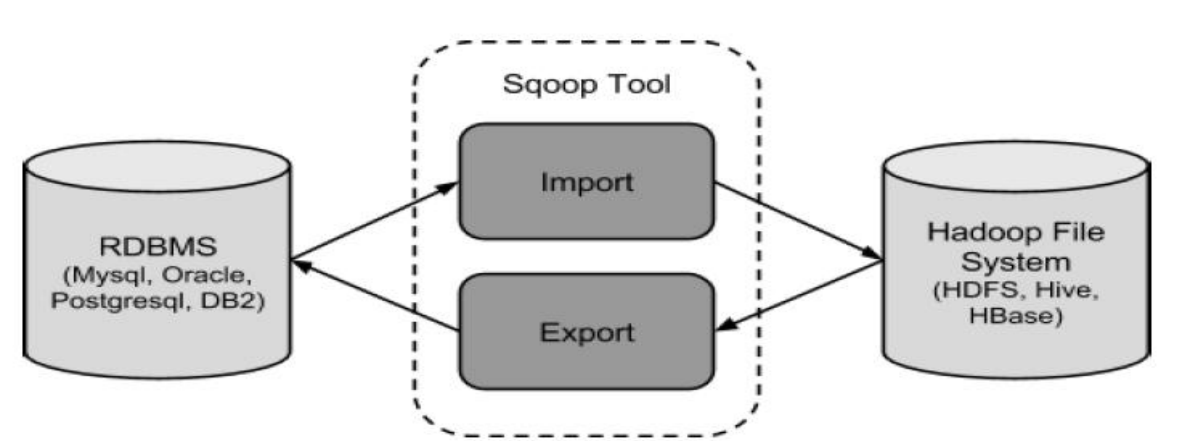


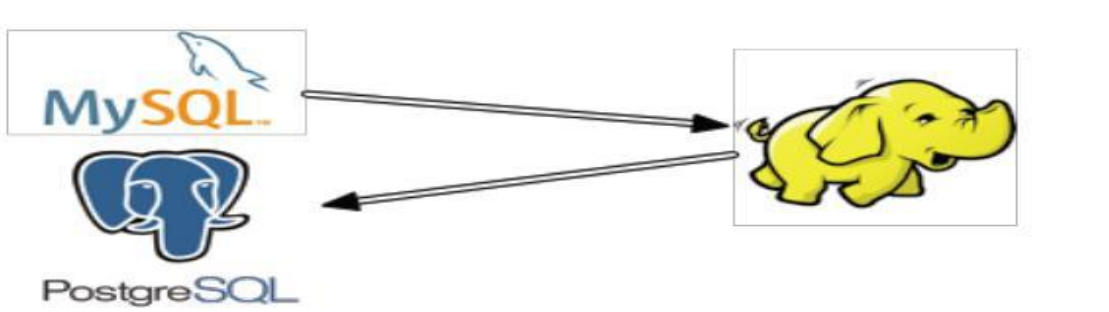
Kafka拓扑结构图


(Producer)
Front End
Front End
Front End
Service
(Push)
(Push)
(Push)
(Push)
(Kafka)
Broker
Broker
Broker
ZooKeeper
(Pull)
(Pull)
(Pull)
(Pull)
(Consumer)
Hadoop Cluster
Real-time Monitoring
Other Service
Data Warehouse

Sqoop

- Sqoop：SQL-to-Hadoop
- 连接传统关系型数据库和Hadoop的桥梁
- 底层用MapReduce程序实现ETL





MySQL to HDFS

- Example:
- sqoop import \
- --connect jdbc:mysql://<ip address> \ <database name>
- --table <mysql_table name>
- --username <username_for_mysql_user> --password <Password>
- -m <number pf mappers to run>
- --target-dir <target directory where we need copied data>

Import to Hive

- sqoop import
- --connect jdbc:mysql://<ip address> \ <database name>
- --table <mysql_table name>
- --username <username_for_mysql_user> --password <Password>
- -m 1
- --hive import

Import to HBase

- sqoop import \
- --connect jdbc:mysql://<ip address> \ <database name>
- --table <mysql_table name>
- --hbase-table <hbase_target_table_name>
- --column-family <column_family_name>
- --hbase-row-key <row_key_column>
- --hbase-create-table

Export from HDFS to Mysql

- sqoop export
- --connect jdbc:mysql://<ip address> \ <database name>
- --table <mysql_table name>
- --username <username_for_mysql_user> --password <Password>
- -m <number pf mappers to run>
- --export-dir <the directory name from where data has to be exported>