



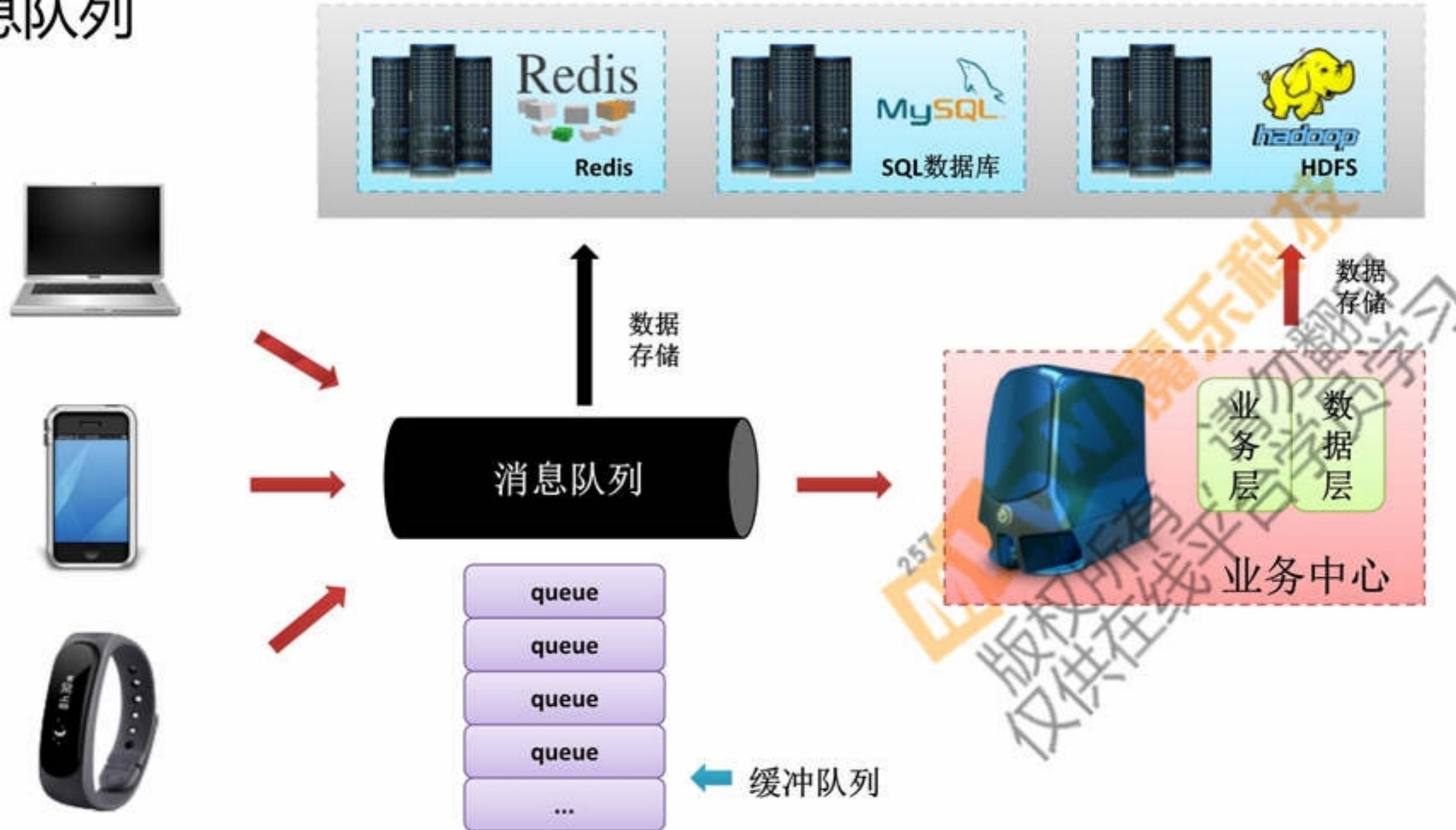
Kafka消息队列中间件

Kafka简介

Kafka消息框架

- Kafka是分布式发布-订阅消息系统。它最初由LinkedIn公司开发，之后成为Apache项目的一部分。
 - 官网地址：<http://kafka.apache.org>
- Kafka是一个分布式的，可划分的，冗余备份的持久性的日志服务。它主要用于处理活跃的流式数据。
- 据LinkedIn统计，最新的数据是每天利用Kafka处理的消息超过1万亿条，在峰值时每秒钟会发布超过百万条消息，就算是内存和CPU都不高的情况下，Kafka的速度最高可以达到每秒十万条数据，并且还能持久化存储。

消息队列



JMS与AMQP

- JMS (Java Message Service 、 Java 消息服务) 应用程序接口，是一个 Java 平台中关于面向消息中间件 (MOM) 的 API。
 - JMS 消息类型：
 - 点对点 (Point-to-Point) 队列消息；
 - 发布 / 订阅 (Publish/Subscribe) ；
 - JMS 支持的数据类型： TextMessage 、 StreamMessage 、 MapMessage 、 ObjectMessage 、 BytesMessage ；
- AMQP (Advanced Message Queuing Protocol) ，一个提供统一消息服务的应用层标准高级消息队列协议，是应用层协议的一个开放标准，为面向消息的中间件设计。
- JMS 与 AMQP 区别：
 - AMQP 是一种协议，更准确的说是一种 binary wire-level protocol (链接协议) 。
 - AMQP 不从 API 层进行限定，而是直接定义网络交换的数据格式。这使得实现了 AMQP 的 provider 天然性就是跨平台的。

常见消息服务组件

	ActiveMQ	RabbitMQ	Kafka
所属社区/公司	Apache	Mozilla Public License	Apache / LinkedIn
开发语言	Java	Erlang	Java / Scala
支持协议	OpenWire、STOMQ、REST、XMPP、AMQP	AMQP	仿AMQP
事务处理	支持	不支持	不支持
集群	支持	支持	支持
负载均衡	支持	支持	支持
动态扩容	不支持	不支持	支持 (ZK)



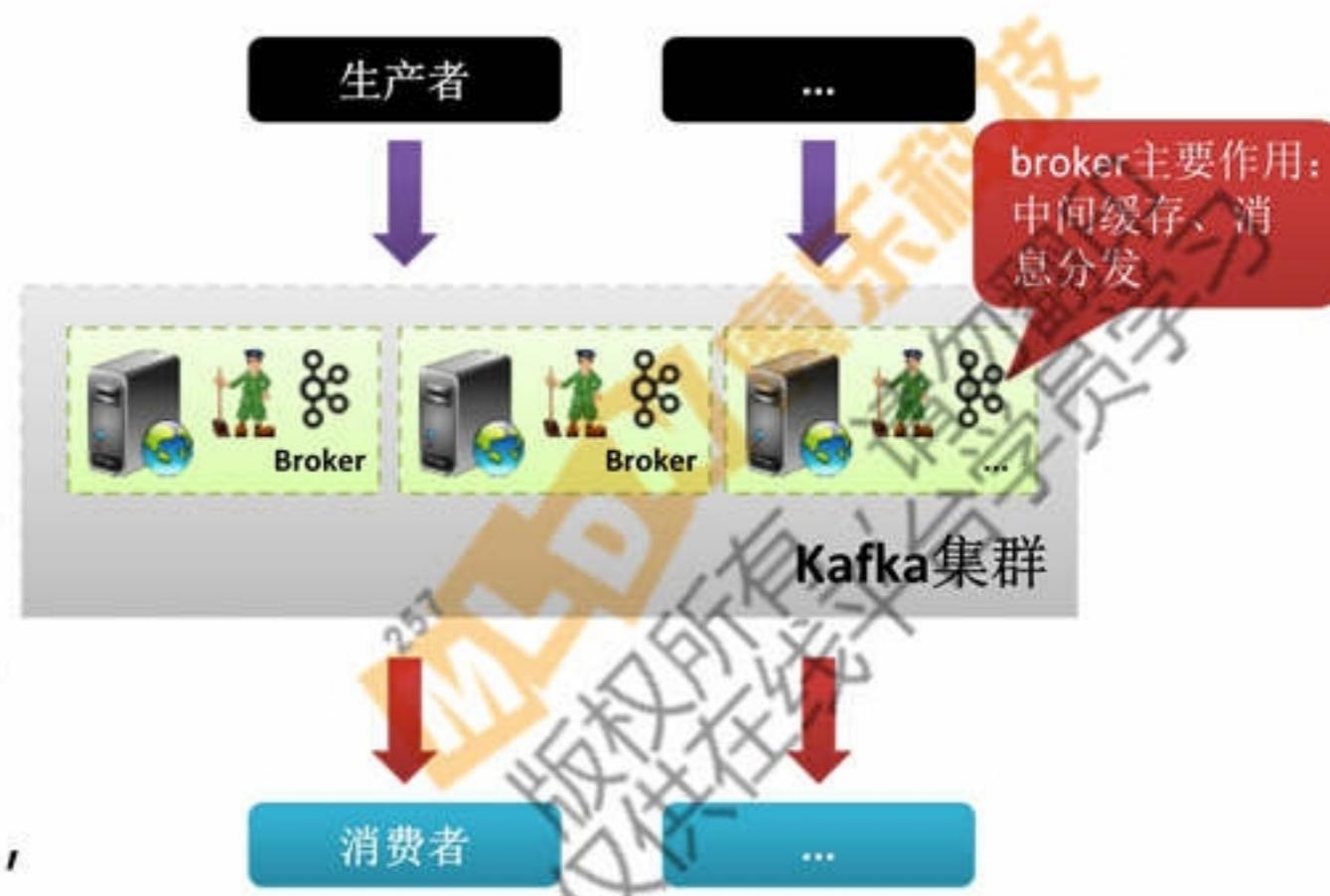
Kafka消息队列中间件

Kafka原理分析



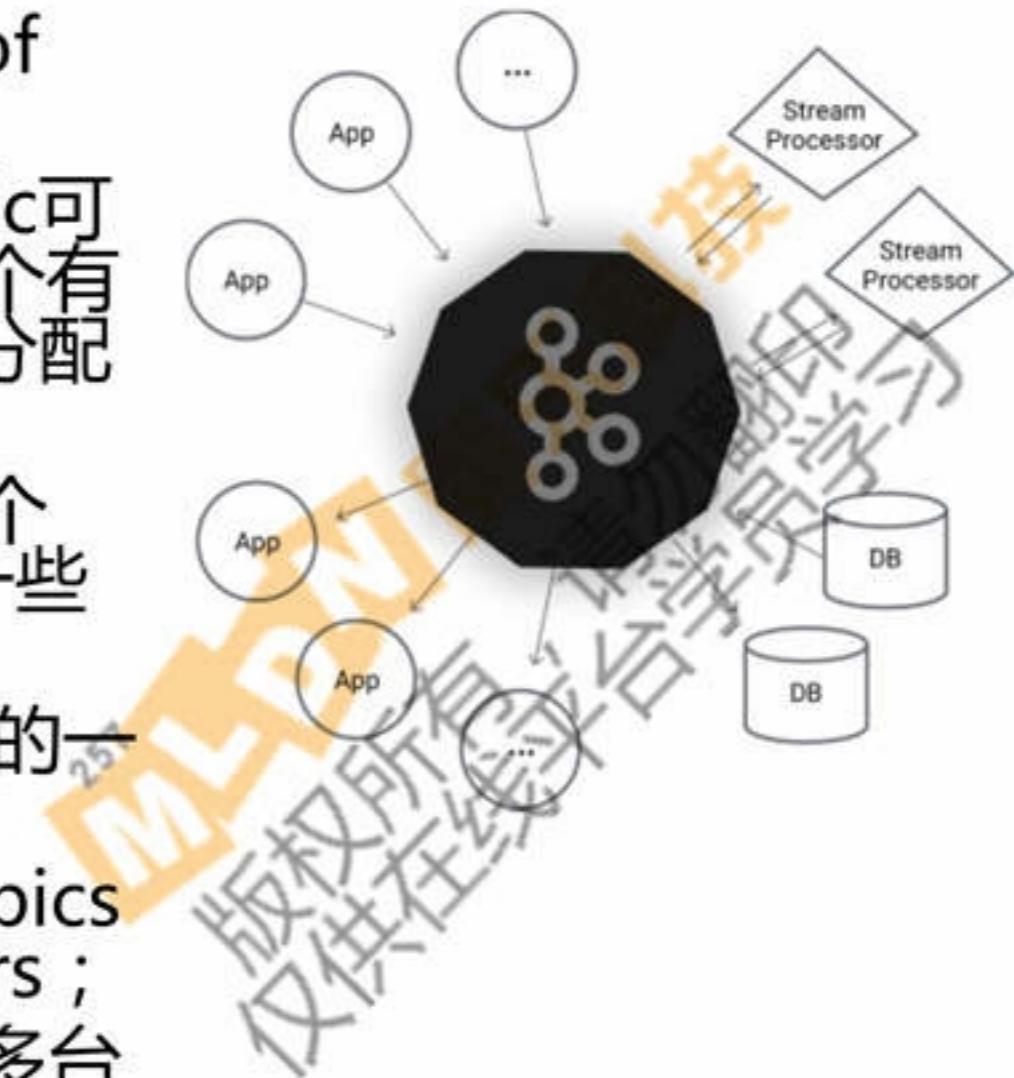
Kafka架构

- Kafka整体采用显式分布式架构，producer、broker (kafka) 和 consumer都可以有多个。
- Producer , consumer实现 Kafka注册的接口，数据从 producer发送到broker , broker承担一个中间缓存和分发的作用。 broker分发注册到系统中的consumer。
- broker的作用类似于缓存，即活跃的数据和离线处理系统之间的缓存。客户端和服务端的通信，是基于简单，高性能，且与编程语言无关的TCP协议。

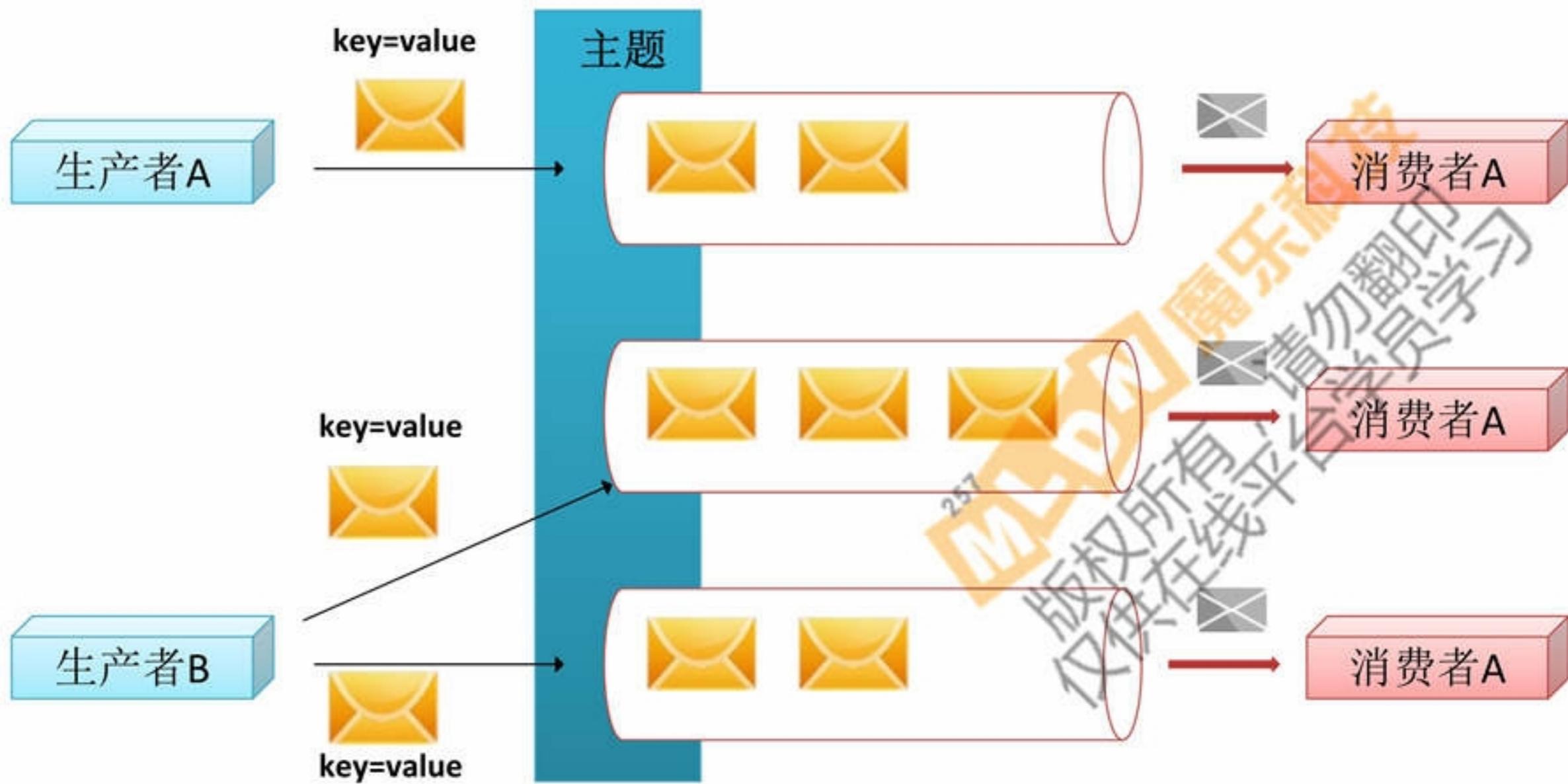


Kafka核心概念

- Topic : 特指Kafka处理的消息源 (feeds of messages) 的不同分类 ;
- Partition : Topic物理上的分组 , 一个topic可以分为多个partition , 每个partition是一个有序的队列。partition中的每条消息都会被分配一个有序的id (offset) ;
- Message : 消息 , 是通信的基本单位 , 每个producer可以向一个topic (主题) 发布一些消息 ;
- Producers : 消息和数据生产者 , 向Kafka的一个topic发布消息的过程叫做producers ;
- Consumers : 消息和数据消费者 , 订阅topics并处理其发布的消息的过程叫做consumers ;
- Broker : 缓存代理 , Kafka集群中的一台或多台服务器统称为broker。

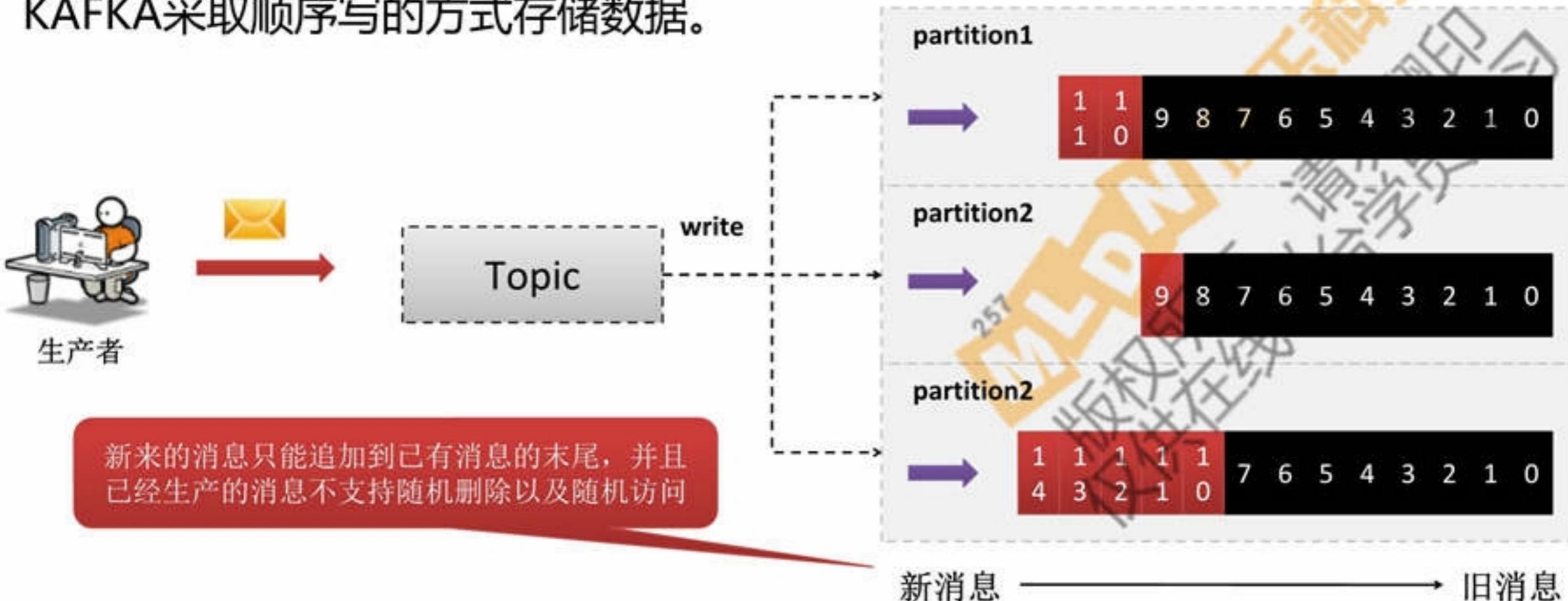


Kafka消息发送流程



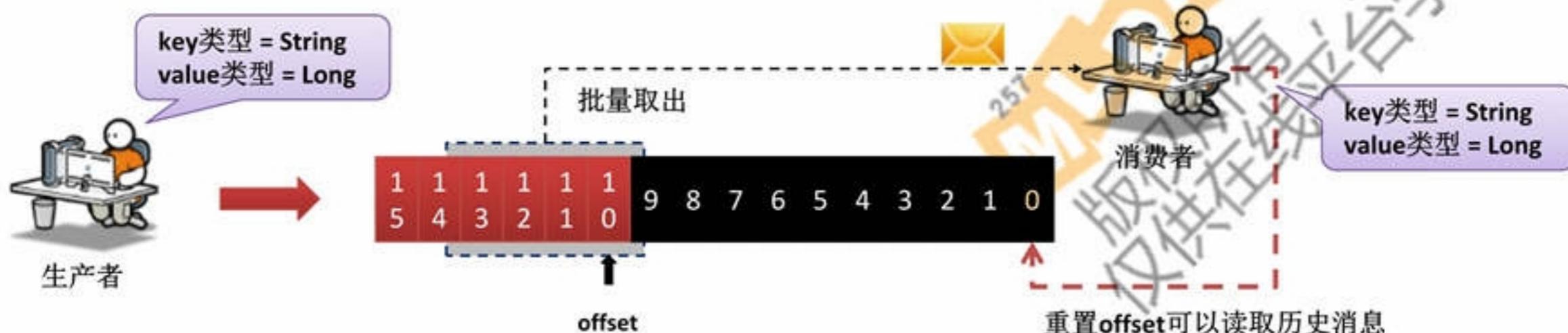
Kafka消息处理流程

- 如果将消息以随机写的方式存入磁盘，就会按柱面、磁头、扇区的方式进行（寻址过程），缓慢的机械运动（相对内存）会消耗大量时间，导致磁盘的写入速度只能达到内存写入速度的几百万分之一。为了规避随机写带来的时间消耗，KAFKA采取顺序写的方式存储数据。



offset与消息访问

- 消费者通过offset的方式来取得消息数据，利用offset偏移改变消息读取位置，可以实现历史消息读取；
- 为了避免频繁的大量小数据的磁盘IO操作，Kafka采用批量读取模式处理消息；
- 在高负载状态下，为放置无效率的字节复制，Kafka采用由Producer，broker和consumer共享的标准化二进制消息格式，这样数据块就可以在它们之间自由传输，无需转换，降低了字节复制的成本开销。



Kafka内存应用

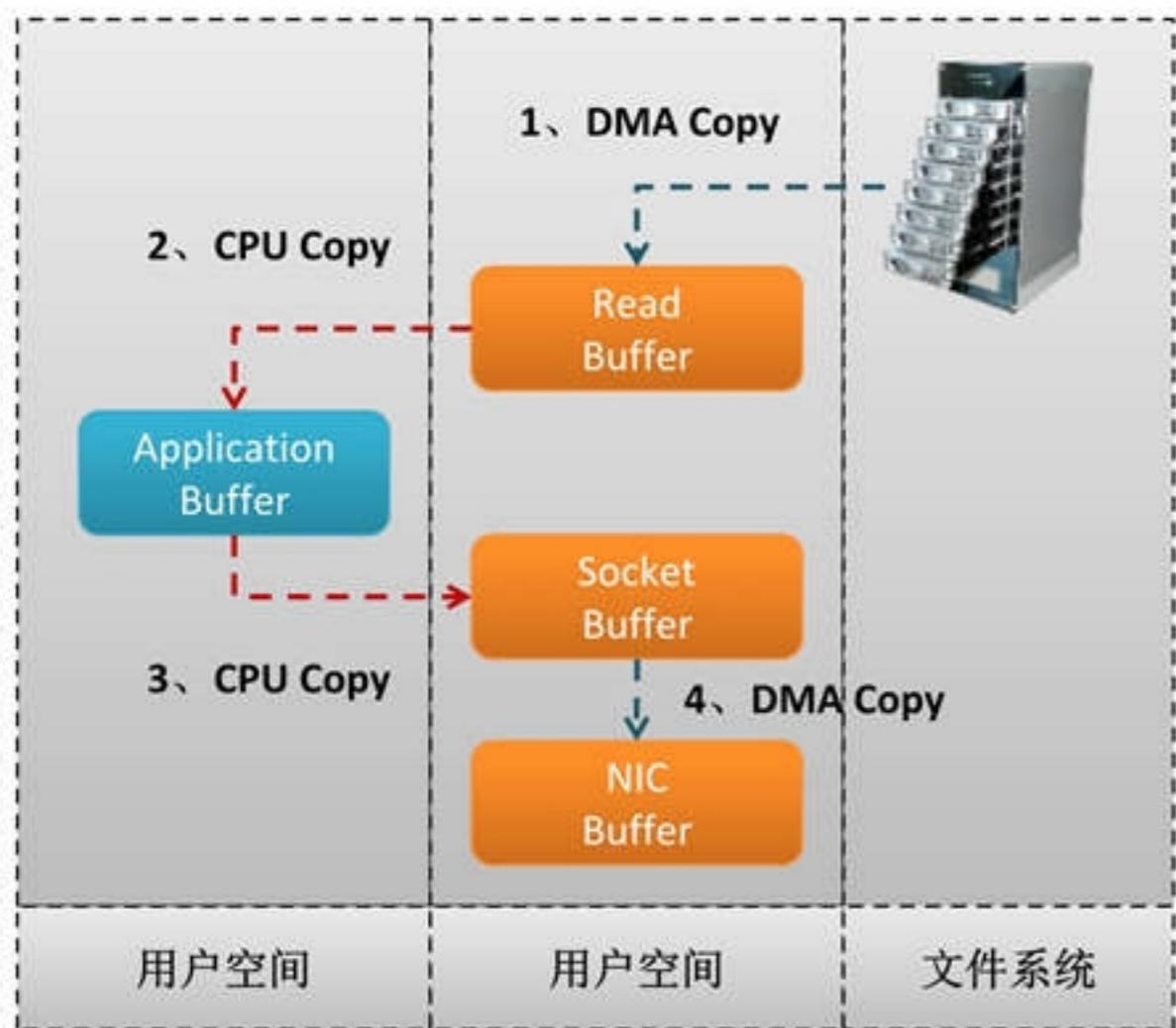
- KAFKA采用了MMAP (Memory Mapped Files , 内存映射文件) 技术。
 - 现代操作系统都大量使用主存做磁盘缓存，一个现代操作系统可以将内存中的所有剩余空间用作磁盘缓存，而当内存回收的时候几乎没有性能损失。
- Kafka是基于JVM技术的应用：
 - 传统JVM技术缺点：
 - 对象的内存开销非常高，通常是实际要存储数据大小的两倍；
 - 随着数据的增加，java的垃圾收集也会越来越频繁并且缓慢。
 - Kafka内存维护：文件系统 + 页缓存（例如：FIFO、CLOCK）：
 - 不使用进程内缓存，就腾出了内存空间，可以直接利用操作系统的页缓存来实现文件到物理内存的直接映射（容量翻倍），完成映射之后对物理内存的操作在适当时候会被同步到硬盘上。
 - 由于使用操作系统页缓存，即使KAFKA重新启动，数据依然可以使用。
 - KAFKA会将数据写入到持久化日志中而不是刷新到磁盘。实际上它只是转移到了内核的页缓存。

文件传输处理

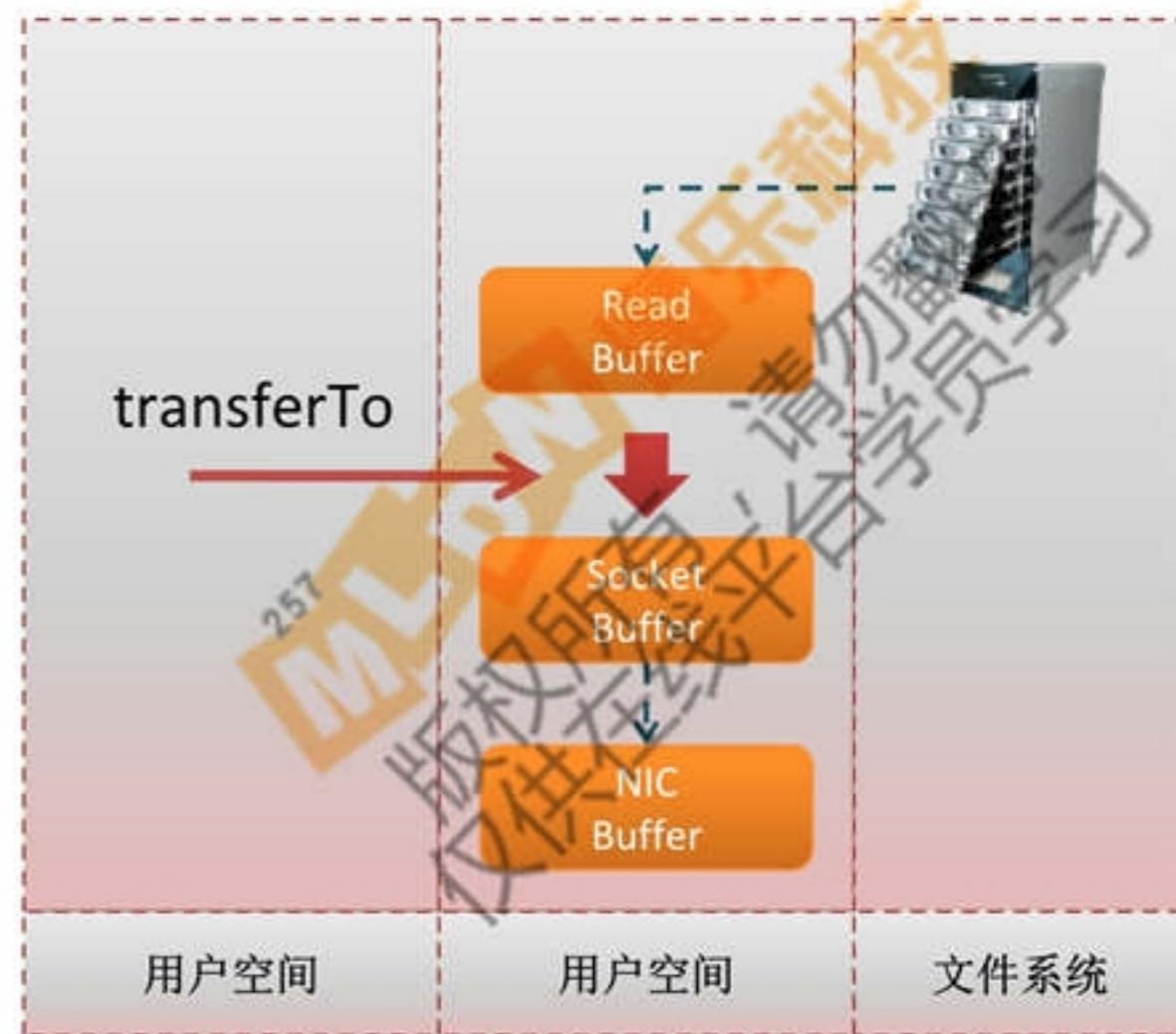
- 消息的数据接收需要通过broker完成，broker维护的消息日志就是文件的目录，每个文件都是二进制保存，生产者和消费者使用相同的格式来处理。
- 维护公共的格式并允许优化最重要操作：网络传输持久性日志块。
 - 传统文件传输处理操作步骤：
 - 操作系统将数据从磁盘读入到内核空间的页缓存；
 - 应用程序将数据从内核空间读入到用户空间缓存中；
 - 应用程序将数据写回到内核空间到socket缓存中；
 - 操作系统将数据从socket缓冲区复制到网卡缓冲区，以便将数据经网络发出；
 - sendfile系统传输：现代的unix操作系统提供一个优化的代码路径，用于将数据从页缓存传输到socket；在Linux中，是通过sendfile系统调用来完成的。

传统文件传输方式与零拷贝

➤ 传统文件传输方式



➤ sendfile系统

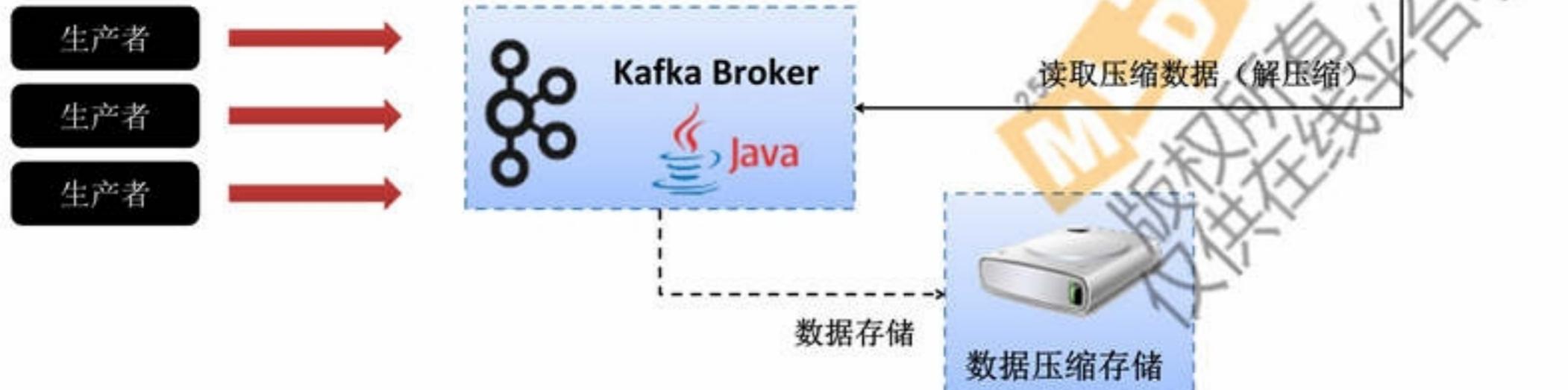


ZeroCopy性能分析

No.	文件大小	传统拷贝 (ms)	ZeroCopy (ms)
1	7MB	156	45
2	21MB	337	128
3	63MB	843	387
4	98MB	1320	617
5	200MB	2124	1150
6	350MB	3631	1762
7	700MB	13498	4422
8	1G	18399	8537

Kafka传输速度保障——批量压缩

- 为了提升消息传输速率，同时为了节约网络带宽，Kafka会在每次传输时将多个消息批量压缩；
- KAFKA批量消息可以通过压缩形式传输并且在日志中也可以保持压缩格式，直到被消费者解压缩。
 - KAFKA支持Gzip和Snappy压缩协议；



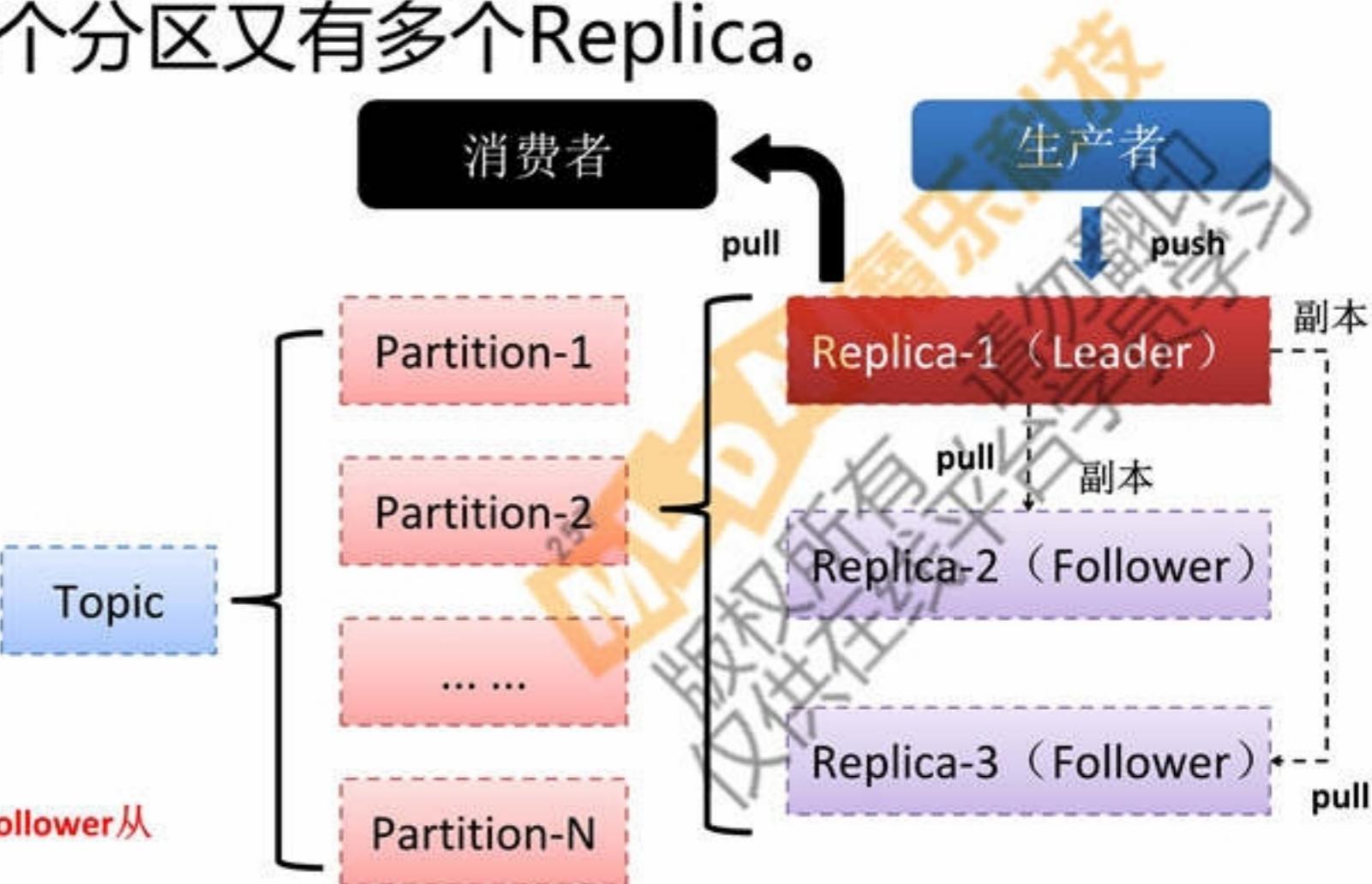
Kafka数据可靠性分析——Topic&Partition

➤ KAFKA的消息保存在Topic中，Topic可分为多个分区，为保证数据的安全性，每个分区又有多个Replica。

➤ 分区设计特点：

- 为了并发读写，加快读写速度；
- 利用多分区的存储，利于数据的均衡；
- 为了加快数据的恢复速率，一旦某台机器挂了，整个集群只需要恢复一部分数据，可加快故障恢复的时间。

Producer和Consumer都只与Leader交互，每个Follower从Leader拉取数据进行同步。



Kafka数据可靠性分析 —— Partition结构分析

- 一个分区（ Partition ）分为多个数据段（ Segment ），每个数据段（ Segment ）由如下几个文件组成：
 - *.log：保存所有的消息数据，同时每条消息设置有一个offset数据保存log文件的索引位置；
 - *.index：保存*.log文件的索引；
 - *.timeindex：保存每条消息的时间戳（ 0.10 版本后设置）；
- Consumer消费流程： Consumer查找 offset 时使用的是二分法根据文件名去定位到具体 Segment ，然后根据 offset 去解析匹配的消息。





Kafka消息队列中间件

Kafka安装与配置

Kafka启动命令

- 启动ZooKeeper进程：

- /usr/local/kafka/bin/zookeeper-server-start.sh
/usr/local/kafka/config/zookeeper.properties >> zookeeper.log
2>&1 &

- 启动KafkaServer进程：

- /usr/local/kafka/bin/kafka-topics.sh --create --zookeeper
localhost:2181 --replication-factor 1 --partitions 1 --topic mldn-topic
&

Kafka操作命令

- 创建消息主题：
 - /usr/local/kafka/bin/kafka-topics.sh --create --zookeeper localhost:2181 --replication-factor 1 --partitions 1 --topic mldn-topic
- 查看Kafka主题：
 - /usr/local/kafka/bin/kafka-topics.sh --list --zookeeper localhost:2181
- 启动消息发送进程：
 - /usr/local/kafka/bin/kafka-console-producer.sh --broker-list localhost:9092 --topic mldn-topic
- 启动消息接收者进程：
 - /usr/local/kafka/bin/kafka-console-consumer.sh --zookeeper localhost:2181 --topic mldn-topic [--from-beginning]



Kafka消息队列中间件

Kafka (SSL) 安全认证

Kafka支持的认证处理

➤ JavaSSL认证：

➤ SSL (Secure Sockets Layer 安全套接层) , 及其继任者传输层安全 (Transport Layer Security , TLS) 是为网络通信提供安全及数据完整性的一种安全协议。TLS与SSL在传输层对网络连接进行加密。

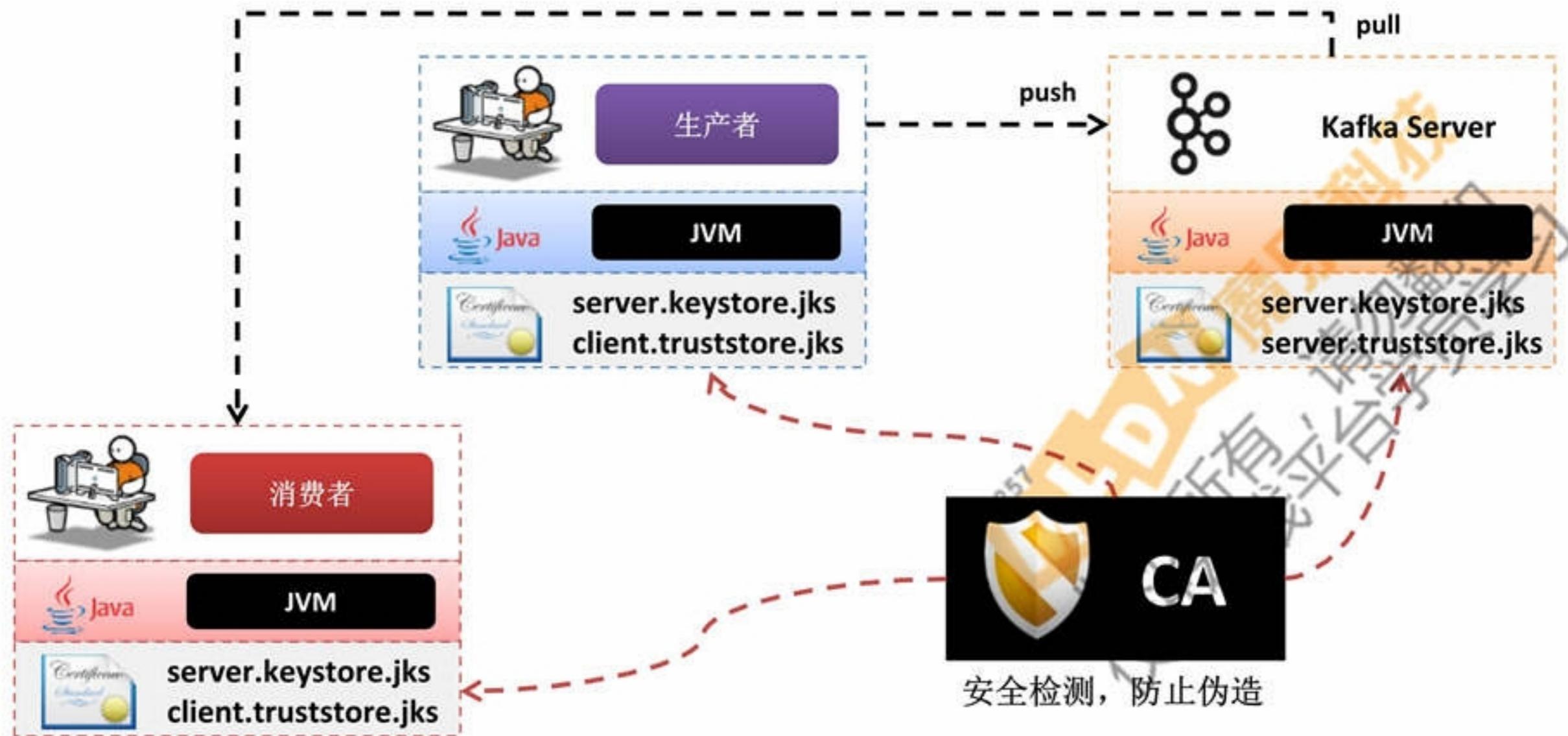
➤ Kerberos认证：

➤ Kerberos 是一种网络认证协议，其设计目标是通过密钥系统为客户端 / 服务器应用程序提供强大的认证服务。

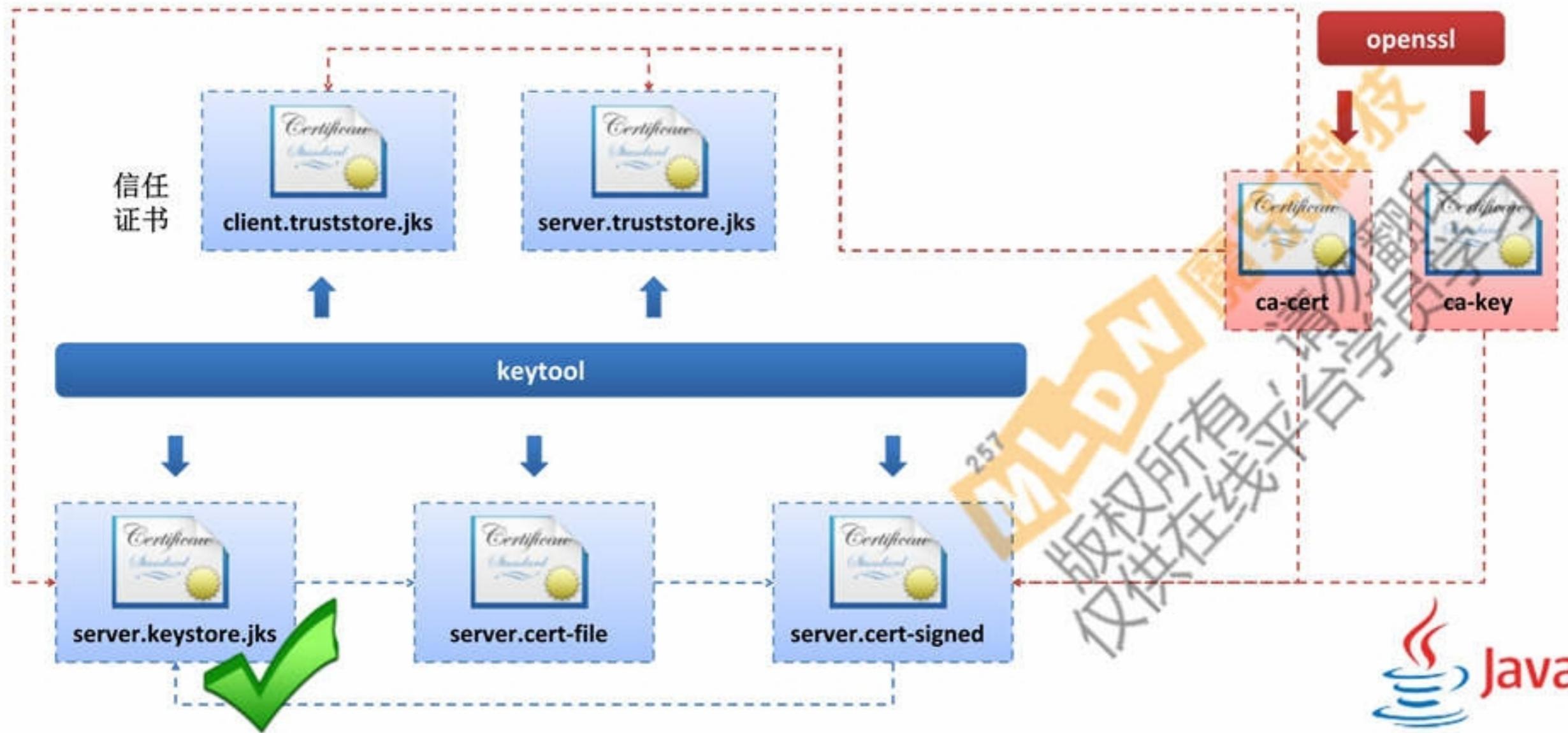
➤ 访问控制列表 (ACL) :

➤ 用户权限控制。

SSL认证流程

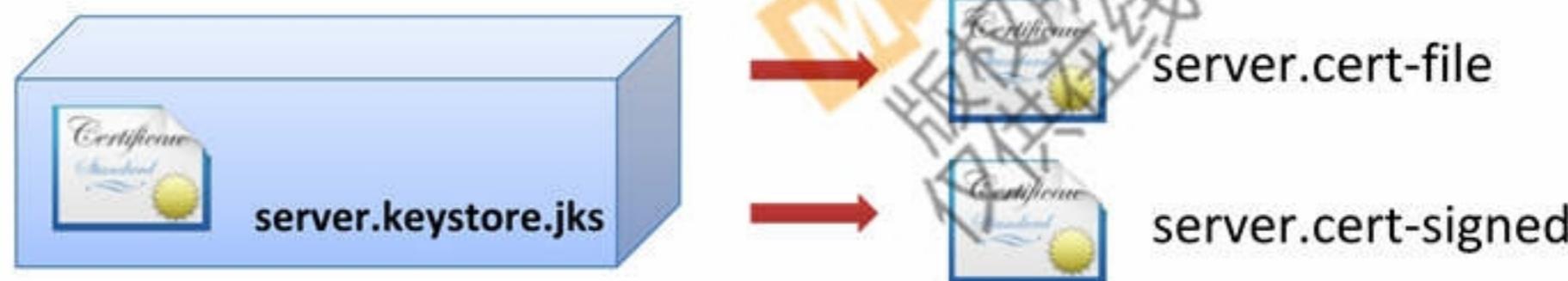


服务端SSL证书签发



创建服务器端keystore

- 创建所有需要使用到的证书保存目录：mkdir -p /usr/ca/{root,server,client,trust}；
- 生成server.keystore.jks文件：
 - keytool -keystore /usr/ca/server/server.keystore.jks -alias kafka-single -validity 365 -genkey -keystorepass mldnjava -dname "CN=kafka-single.com,OU=mldn,O=mldnjava,L=bj,S=bj,C=cn" -storepass mldnjava



服务器端证书签发准备

➤ 创建服务器keystore（密钥和证书）：

```
➤ keytool -keystore /usr/ca/server/server.keystore.jks -alias kafka-single -validity 365 -genkey -keypass mldnjava -dname "CN=kafka-single.com,OU=mldn,O=mldnjava,L=bj,S=bj,C=cn" -storepass mldnjava
```

➤ 创建CA认证证书：

```
➤ 利用OpenSSH创建CA认证处理证书“ca.crt”：openssl req -new -x509 -keyout /usr/ca/root/ca-key -out /usr/ca/root/ca-cert -days 365 -passout pass:mldnjava -subj "/C=cn/ST=bj/L=bj/O=mldn/OU=mldnjava/CN=kafka-single.com"
```

```
➤ 通过CA证书创建一个服务器端信任证书（以后的证书必须通过CA认证后才可以使用）：keytool -keystore /usr/ca/trust/server.truststore.jks -alias CARoot -import -file /usr/ca/root/ca-cert -storepass mldnjava
```

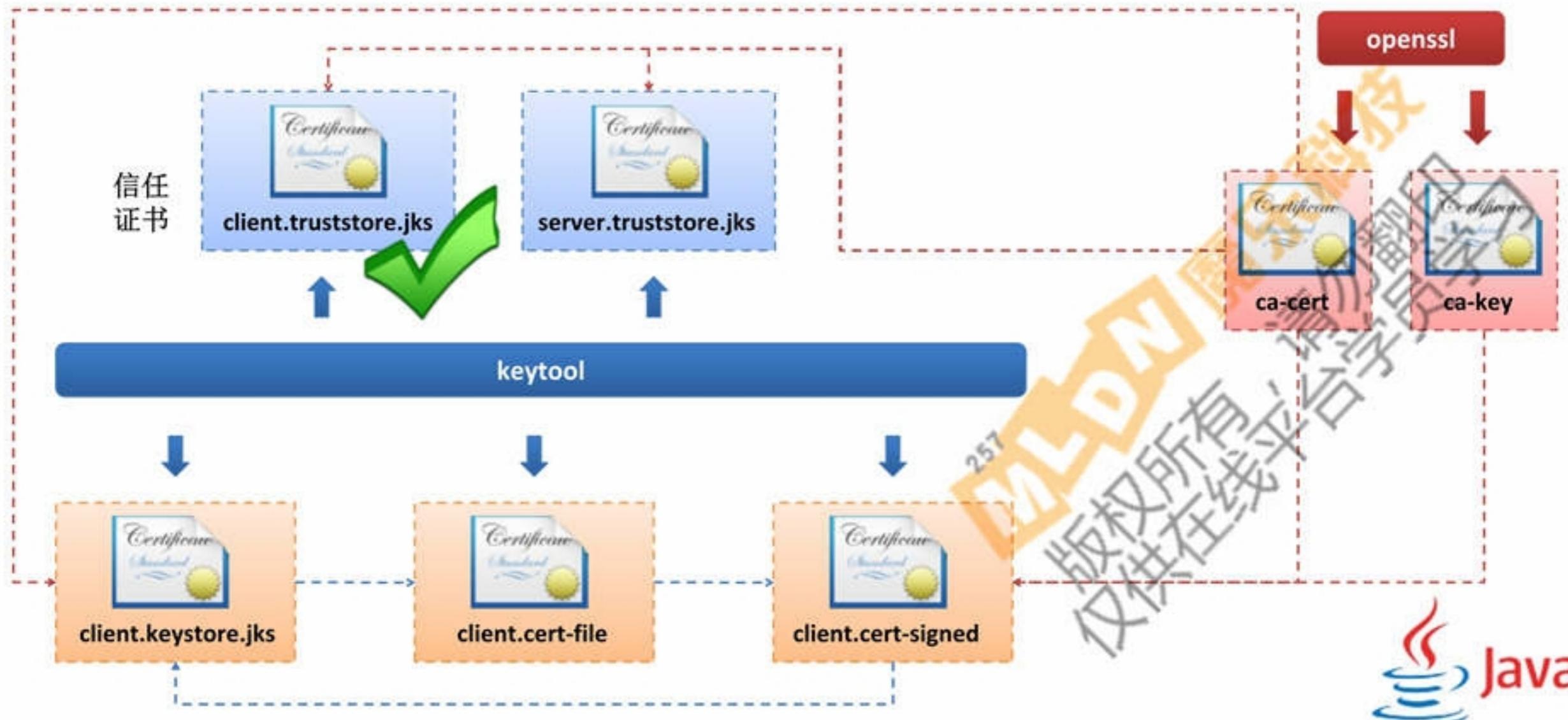
创建CA认证

- 未签名证书有可能被伪造，所以需要有一个证书颁发机构（CA）负责签名证书；
- 利用OpenSSH创建CA认证处理证书“ca.crt”：
 - `openssl req -new -x509 -keyout /usr/ca/root/ca-key -out /usr/ca/root/ca-cert -days 365 -passout pass:mldnjava -subj "/C=cn/ST=bj/L=b/O=mldn/OU=mldnjava/CN=kafka-single.com"`
- 通过CA证书创建一个服务器端信任证书（以后的证书必须通过CA认证后才可以使用）：
 - `keytool -keystore /usr/ca/trust/server.truststore.jks -alias CARoot -import -file /usr/ca/root/ca-cert -storepass mldnjava`
- 通过CA证书创建一个客户端信任证书（以后的证书必须通过CA认证后才可以使用）：
 - `keytool -keystore /usr/ca/trust/client.truststore.jks -alias CARoot -import -file /usr/ca/root/ca-cert -storepass mldnjava`

服务器证书签名处理

- 导出服务器端证书 “server.cert-file”：
 - keytool -keystore /usr/ca/server/server.keystore.jks -alias kafka-single -certreq -file /usr/ca/server/server.cert-file -storepass mldnjava
- 用CA给服务器端证书进行签名处理：
 - openssl x509 -req -CA /usr/ca/root/ca-cert -CAkey /usr/ca/root/ca-key -in /usr/ca/server/server.cert-file -out /usr/ca/server/server.cert-signed -days 365 -CAcreateserial -passin pass:mldnjava
- 将CA证书导入到服务器keystore：
 - keytool -keystore /usr/ca/server/server.keystore.jks -alias CARoot -import -file /usr/ca/root/ca-cert -storepass mldnjava
- 将已签名的服务器证书导入到服务器keystore：
 - keytool -keystore /usr/ca/server/server.keystore.jks -alias kafka-single -import -file /usr/ca/server/server.cert-signed -storepass mldnjava

客户端SSL证书签发



客户端证书签名处理（集群中每个主机都要处理）

- 导出客户端证书：
 - keytool -keystore /usr/ca/client/client.keystore.jks -alias kafka-single -validity 365 -genkey -keypass mldnjava -dname "CN=kafka-single.com,OU=mldn,O=mldnjava,L=bj,S=bj,C=cn" -storepass mldnjava
- 将服务器端证书文件导入到客户端keystore：
 - keytool -keystore /usr/ca/client/client.keystore.jks -alias kafka-single -certreq -file /usr/ca/client/client.cert-file -storepass mldnjava
- 用CA给客户端证书进行签名处理：
 - openssl x509 -req -CA /usr/ca/root/ca-cert -CAkey /usr/ca/root/ca-key -in /usr/ca/client/client.cert-file -out /usr/ca/client/client.cert-signed -days 365 -CAcreateserial -passin pass:mldnjava
- 将CA证书导入到客户端keystore：
 - keytool -keystore /usr/ca/client/client.keystore.jks -alias CARoot -import -file /usr/ca/root/ca-cert -storepass mldnjava
- 将已签名的服务器证书导入到客户端keystore：
 - keytool -keystore /usr/ca/client/client.keystore.jks -alias kafka-single -import -file /usr/ca/client/client.cert-signed -storepass mldnjava

在Kafka上配置SSL (server.properties)

- listeners=SSL://kafka-single:9093
- advertised.listeners=SSL://kafka-single:9093
- ssl.keystore.location=/usr/ca/server/server.keystore.jks
- ssl.keystore.password=mldnjava
- ssl.key.password=mldnjava
- ssl.truststore.location=/usr/ca/trust/server.truststore.jks
- ssl.truststore.password=mldnjava
- ssl.client.auth=required
- ssl.enabled.protocols=TLSv1.2,TLSv1.1,TLSv1
- ssl.keystore.type=JKS
- ssl.truststore.type=JKS
- security.inter.broker.protocol=SSL

Java客户端配置

- props.put(SslConfigs.***SSL_KEYSTORE_LOCATION_CONFIG***, "D:**server.keystore.jks**");
- props.put(SslConfigs.***SSL_KEYSTORE_PASSWORD_CONFIG***, "mldnjava");
- props.put(SslConfigs.***SSL_TRUSTSTORE_LOCATION_CONFIG***, "D:**client.truststore.jks**");
- props.put(SslConfigs.***SSL_TRUSTSTORE_PASSWORD_CONFIG***, "mldnjava");
- props.put(SslConfigs.***SSL_TRUSTSTORE_TYPE_CONFIG***, "JKS");
- props.put(CommonClientConfigs.***SECURITY_PROTOCOL_CONFIG***, "SSL");

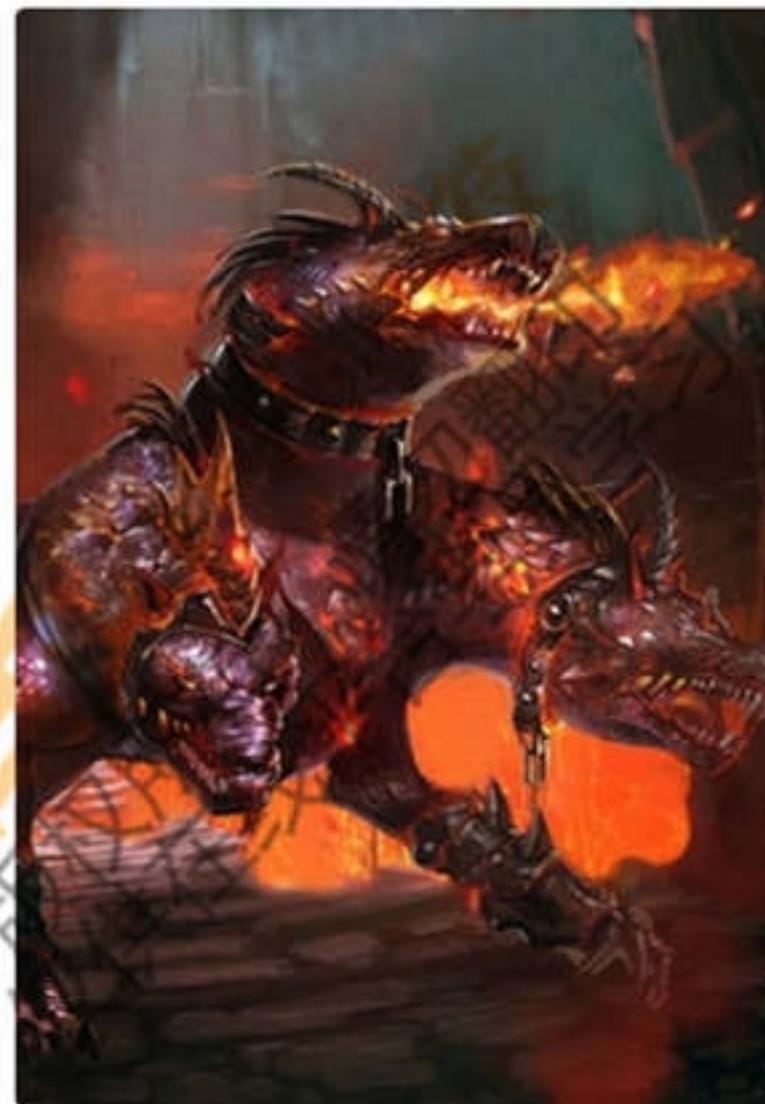


Kafka消息队列中间件

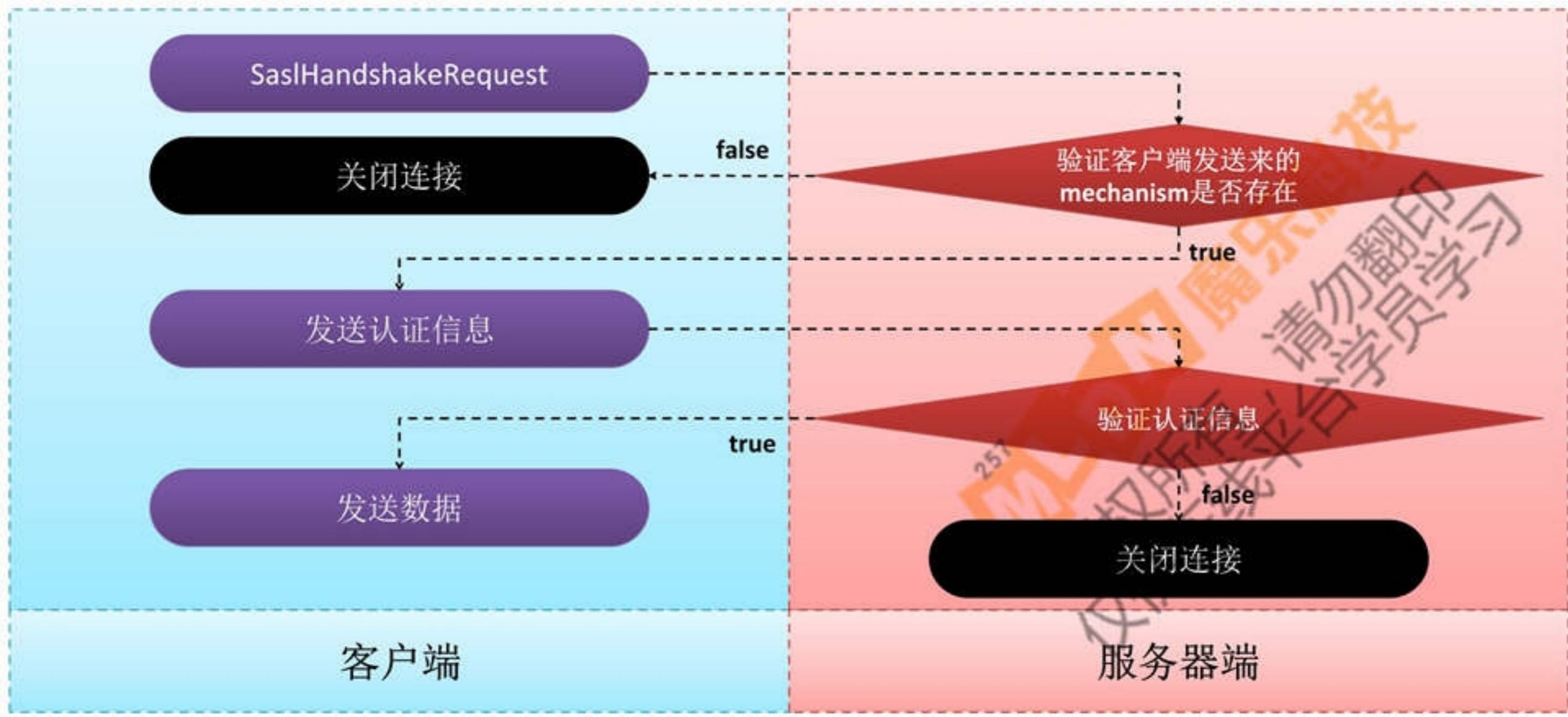
Kerberos (SASL) 认证

Kerberos: : Network Authentication Protocol (网络认证协议)

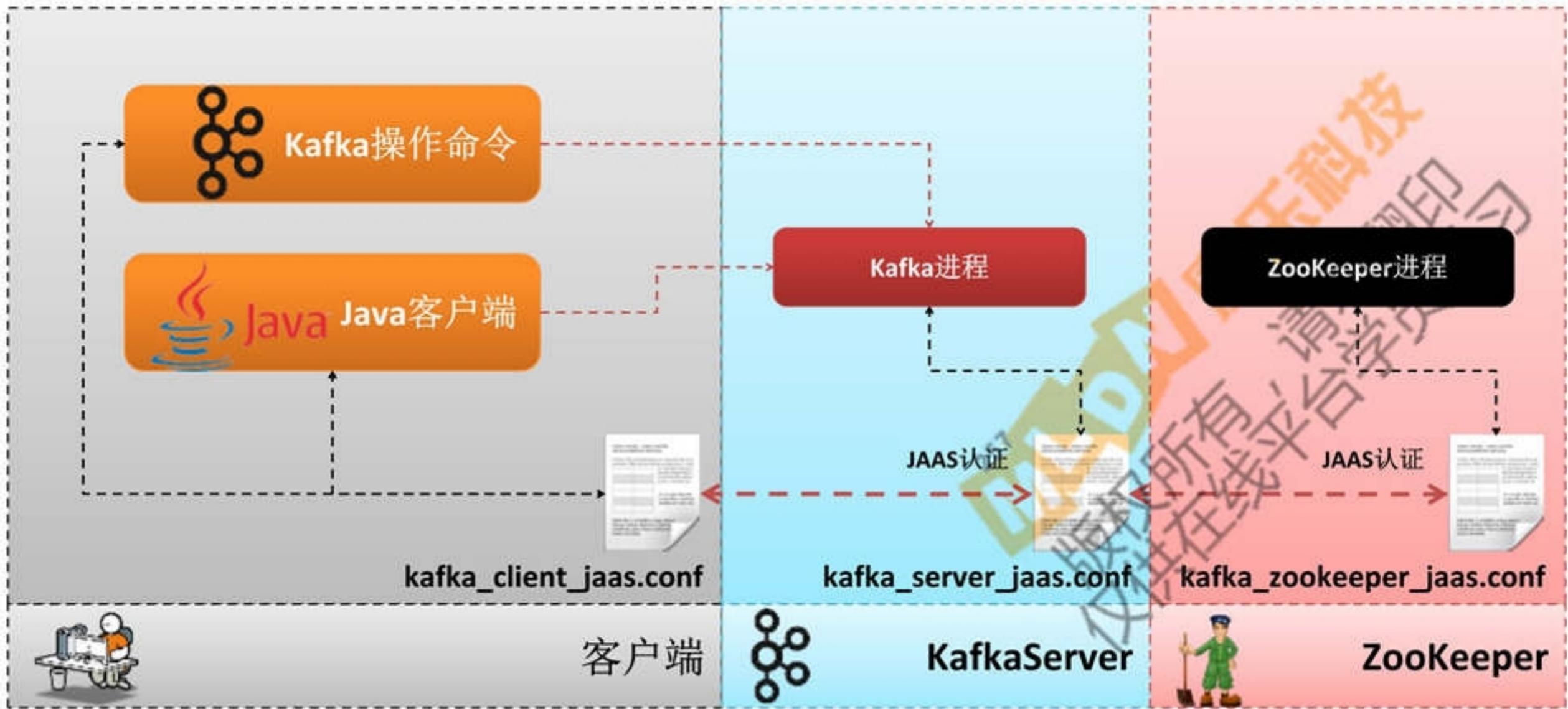
- Kerberos这一名词来源于希腊神话 “三个头的狗——地狱之门守护者” ；
- 系统设计上采用客户端/服务器结构与 DES 加密技术，并且能够进行相互认证，即客户端和服务器端均可对对方进行身份认证。可以用于防止窃听、防止 replay 攻击、保护数据完整性等场合，是一种应用对称密钥体制进行密钥管理的系统。
- 支持SSO ；



Kerberos处理流程



Kerberos配置流程



建立Kerberos配置文件

- /usr/local/kafka/jaas/kafka_zookeeper_jaas.conf

```
Server {  
    org.apache.kafka.common.security.plain.PlainLoginModule required  
    username="zkadmin"  
    password="zkadmin-pwd";  
};
```

- /usr/local/kafka/jaas/kafka_server_jaas.conf

```
KafkaServer {  
    org.apache.kafka.common.security.plain.PlainLoginModule required  
    username="zkadmin"  
    password="zkadmin-pwd"  
    user_zkadmin="zkadmin-pwd"  
    user_alice="alice-pwd"  
    user_bob="bob-pwd";  
};
```

Kafka配置JAAS选项

- Kafka监听端口： port=9095
- 设置监听地址：
 - listeners=**SASL_PLAINTEXT**//kafka-single:9095
- 设置监听地址，与listeners设置相同：
 - advertised.listeners=**SASL_PLAINTEXT**//kafka-single:9095
- 设置使用处理协议：
 - security.inter.broker.protocol=SASL_PLAINTEXT
- 启用SASL处理机制间的通讯：
 - sasl.enabled.mechanisms=PLAIN
- 设置broker之间的传输机制：
 - sasl.mechanism.inter.broker.protocol=PLAIN
- 允许客户端自动创建主题： auto.create.topics.enable=true

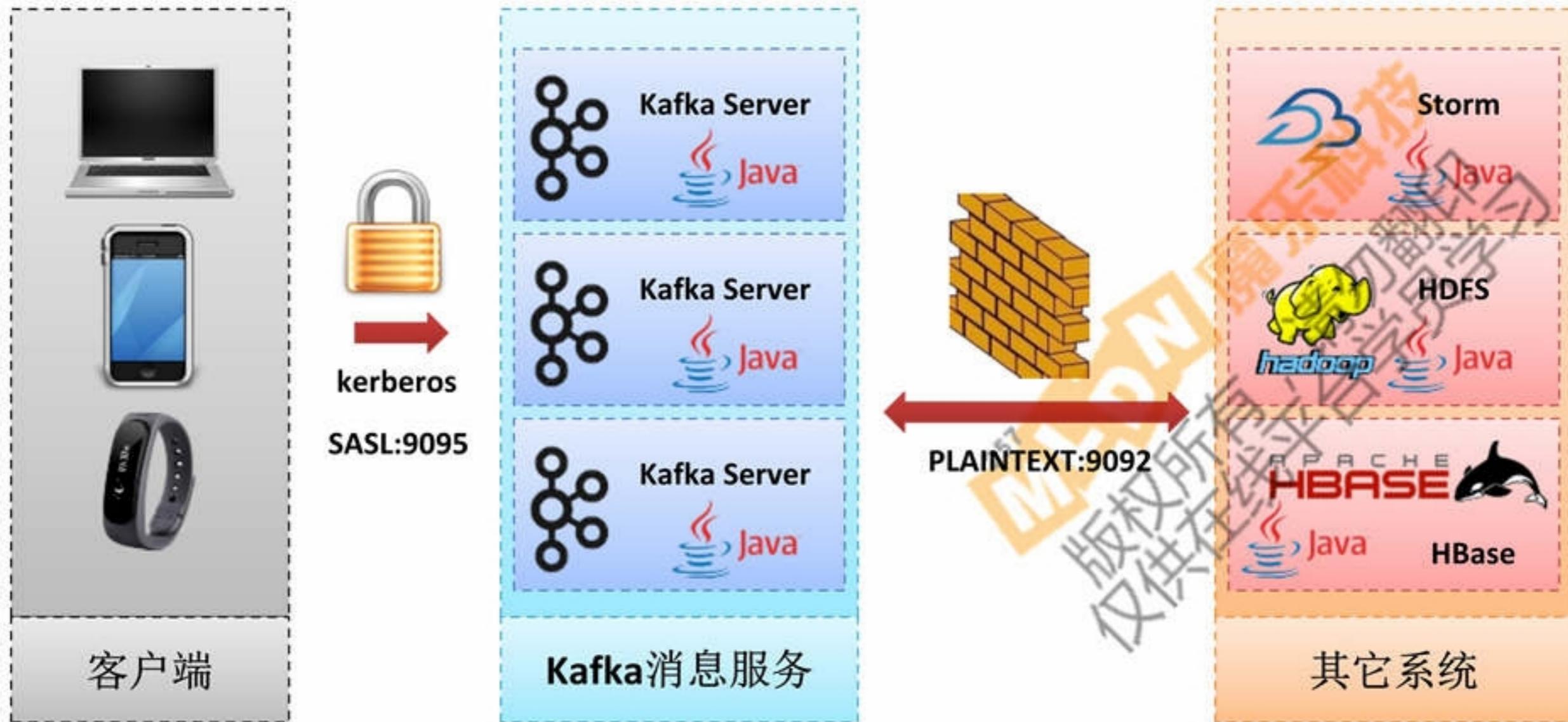
建立客户端JAAS认证文件

- 服务器端现在一共建立有三个账户 : zkadmin、alice、bob ;
- 建立客户端连接的jaas配置文件 : kafka_client_jaas.conf ;

```
KafkaClient {  
    org.apache.kafka.common.security.plain.PlainLoginModule required  
    username="bob"  
    password="bob-pwd";  
};
```

- Java设置Kafka属性 :
 - System.setProperty("java.security.auth.login.config", "d:/kafka_client_jaas.conf");
 - props.setProperty(SaslConfigs.SASL_MECHANISM, "PLAIN");
 - props.setProperty(CommonClientConfigs.SECURITY_PROTOCOL_CONFIG, "SASL_PLAINTEXT");

Kafka多端口配置





Kafka消息队列中间件

Kerberos访问控制列表（ACL）

kafka-acls.sh命令

No.	命令项	描述	默认值	操作类型
01	--add	增加ACL配置项		指令
02	--remove	删除ACL配置项		指令
03	--list	列出全部ACL配置项		指令
04	--authorizer	设置授权管理器	kafka.security.auth.SimpleAclAuthorizer	配置
05	--authorizer-properties	设置“key=value”的属性		配置
06	--cluster	设置集群资源		资源信息
07	--topic [topic-name]	设置主题资源		资源信息
08	--group [group-name]	设置操作组资源		资源信息
09	--allow-principal	允许用户列表		用户名
10	--deny-principal	禁用用户列表		用户名
11	--allow-host	允许主机列表	**表示所有主机	主机名称
12	--deny-host	禁用主机列表	**表示所有主机	主机名称
13	--operation	设置权限: Read、Write、Create、Delete、Alter、Describe、ClusterAction、All	所有权限	权限标记
14	--producer	设置权限: WRITE、DESCRIBE on topic、CREATE on cluster		权限标记
15	--consumer	设置权限: READ、DESCRIBE on topic、READ on consumer-group		权限标记

Kafka权限控制：

- 为“mldn-topic”设置ACL定义：

- /usr/local/kafka/bin/kafka-acls.sh --authorizer-properties zookeeper.connect=localhost:2181 --add --allow-principal User:alice --group group-1 --topic mldn-topic

- ACL列表显示：

- /usr/local/kafka/bin/kafka-acls.sh --authorizer-properties zookeeper.connect=localhost:2181 --list --topic mldn-topic

- 删除ACL列表信息：

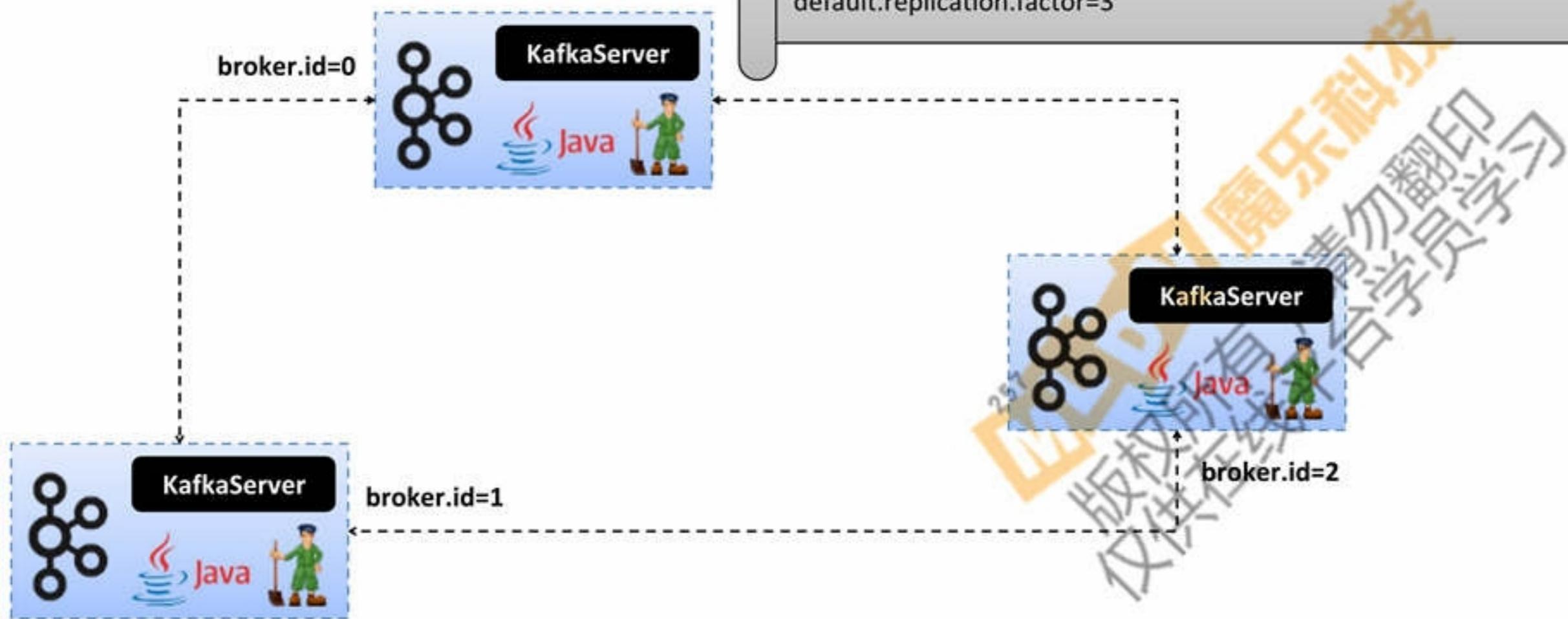
- /usr/local/kafka/bin/kafka-acls.sh --authorizer-properties zookeeper.connect=localhost:2181 --remove --allow-principal User:alice --group group-1 --topic mldn-topic



Kafka消息队列中间件

搭建Kafka集群

Kafka集群





Kafka消息队列中间件

Spring-Kafka

Spring-Kafka

- Spring Kafka是Spring官方提供的一个Spring集成框架的扩展,用来为使用Spring框架的应用程序提供Kafka框架的集成。

The screenshot shows the official Spring for Apache Kafka project page. At the top left, there's a 'PROJECTS' navigation bar. Below it, the title 'Spring for Apache Kafka' is displayed next to a small circular icon containing a gear. The main content area contains a detailed description of the project, mentioning its application of core Spring concepts to Kafka-based messaging solutions, support for message-driven POJOs with annotations like @KafkaListener, and a listener container. It also notes similarities to JMS support in Spring Framework and RabbitMQ support in Spring AMQP. At the bottom left, there's a green-bordered 'QUICK START' button.

PROJECTS

Spring for Apache Kafka

The Spring for Apache Kafka (spring-kafka) project applies core Spring concepts to the development of Kafka-based messaging solutions. It provides a "template" as a high-level abstraction for sending messages. It also provides support for Message-driven POJOs with `@KafkaListener` annotations and a "listener container". These libraries promote the use of dependency injection and declarative. In all of these cases, you will see similarities to the JMS support in the Spring Framework and RabbitMQ support in Spring AMQP.

QUICK START