### Netty的概念

#### Netty的基本概念

Spark作为分布式计算框架，多个节点的设计与相互通信模式是其重要的组成部分。Spark一开始使用 Akka 作为内部通信部件。在Spark 1.3年代，为了解决大块数据（如Shuffle）的传输问题，Spark引入了Netty通信框架。到了 Spark 1.6, Spark可以配置使用 Akka 或者 Netty 了，这意味着 Netty 可以完全替代 Akka了。再到 Spark 2, Spark 已经完全抛弃 Akka了，全部使用Netty了。

为什么呢？官方的解释是：

1. 很多Spark用户也使用Akka，但是由于Akka不同版本之间无法互相通信，这就要求用户必须使用跟Spark完全一样的Akka版本，导致用户无法升级Akka。
2. Spark的Akka配置是针对Spark自身来调优的，可能跟用户自己代码中的Akka配置冲突。
3. Spark用的Akka特性很少，这部分特性很容易自己实现。同时，这部分代码量相比Akka来说少很多，debug比较容易。如果遇到什么bug，也可以自己马上fix，不需要等Akka上游发布新版本。而且，Spark升级Akka本身又因为第一点会强制要求用户升级他们使用的Akka，对于某些用户来说是不现实的。

#### Netty的应用场景

互联网行业：随着网站规模的不断扩大，系统并发访问量也越来越高，传统基于 Tomcat 等 Web 容器的垂直架构已经无法满足需求，需要拆分应用进行服务化，以提高开发和维护效率。从组网情况看，垂直的架构拆分之后，系统采用分布式部署，各个节点之间需要远程服务调用，高性能的 RPC 框架必不可少，Netty 作为异步高性能的通信框架，往往作为基础通信组件被这些 RPC 框架使用。典型的应用有：阿里分布式服务框架 Dubbo 的 RPC 框架使用 Dubbo 协议进行节点间通信，Dubbo 协议默认使用 Netty 作为基础通信组件，用于实现各进程节点之间的内部通信。

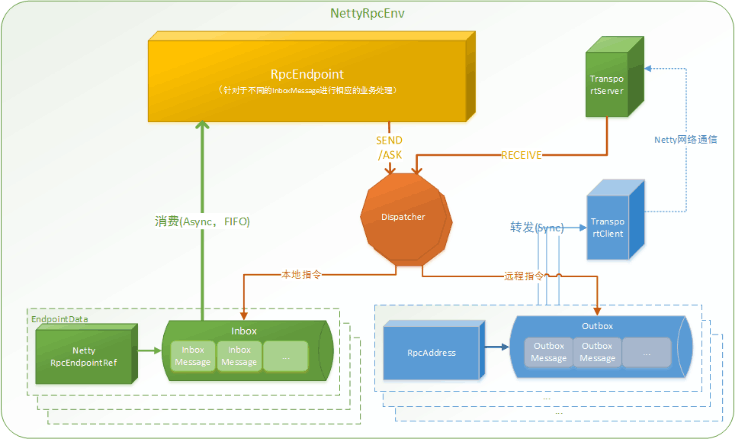
游戏行业：无论是手游服务端、还是大型的网络游戏，Java 语言得到了越来越广泛的应用。Netty 作为高性能的基础通信组件，它本身提供了 TCP/UDP 和 HTTP 协议栈，非常方便定制和开发私有协议栈。账号登陆服务器、地图服务器之间可以方便的通过 Netty 进行高性能的通信。

大数据领域：经典的 Hadoop 的高性能通信和序列化组件 Avro 的 RPC 框架，默认采用 Netty 进行跨节点通信，它的 Netty Service 基于 Netty 框架二次封装实现。大数据计算往往采用多个计算节点和一个/N个汇总节点进行分布式部署，各节点之间存在海量的数据交换。由于 Netty 的综合性能是目前各个成熟 NIO 框架中最高的，因此，往往会被选中用作大数据各节点间的通信。

企业软件：企业和 IT 集成需要 ESB，Netty 对多协议支持、私有协议定制的简洁性和高性能是 ESB RPC 框架的首选通信组件。事实上，很多企业总线厂商会选择 Netty 作为基础通信组件，用于企业的 IT 集成。

通信行业：Netty 的异步高性能、高可靠性和高成熟度的优点，使它在通信行业得到了大量的应用。

#### Netty的原理分析



1. RpcEndpoint：RPC端点 ，Spark针对于每个节点（Client/Master/Worker）都称之一个Rpc端点 ,且都实现RpcEndpoint接口，内部根据不同端点的需求，设计不同的消息和不同的业务处理，如果需要发送（询问）则调用Dispatcher。
2. RpcEnv：RPC上下文环境，每个Rpc端点运行时依赖的上下文环境称之为RpcEnv。
3. Dispatcher：消息分发器，针对于RPC端点需要发送消息或者从远程RPC接收到的消息，分发至对应的指令收件箱/发件箱。如果指令接收方是自己存入收件箱，如果指令接收方为非自身端点，则放入发件箱。
4. Inbox：指令消息收件箱，一个本地端点对应一个收件箱，Dispatcher在每次向Inbox存入消息时，都将对应EndpointData加入内部队列Receiver Queue中，另外Dispatcher创建时会启动一个单独线程进行轮询Receiver Queue，进行收件箱消息消费。
5. OutBox：指令消息发件箱，一个远程端点对应一个发件箱，当消息放入Outbox后，紧接着将消息通过TransportClient发送出去。消息放入发件箱以及发送过程是在同一个线程中进行，这样做的主要原因是远程消息分为RpcOutboxMessage, OneWayOutboxMessage两种消息，而针对于需要应答的消息直接发送且需要得到结果进行处理。
6. TransportClient：Netty通信客户端，根据OutBox消息的receiver信息，请求对应远程TransportServer。
7. TransportServer：Netty通信服务端，一个RPC端点一个TransportServer,接受远程消息后调用Dispatcher分发消息至对应收发件箱。

注意：

TransportClient与TransportServer通信虚线表示两个RpcEnv之间的通信。

一个Outbox一个TransportClient。

一个RpcEnv中存在两个RpcEndpoint,一个代表本身启动的RPC端点，另外一个为 RpcEndpointVerifier。

需求：

1. 创建Server端
2. 实现Server端的Handler
3. 创建Client端
4. 实现Client端的Handler
5. 进行通信