在学习过程中分析的为hadoop-2.6.0源码

#### Hadoop Yarn:

###### 5个源码包

1. API : Yarn内部涉及的4个RPC协议的java声明
2. Common : 底层库实现 : 事件库,服务库,状态机库,Web界面…
3. Application : 包含Application编程实例
4. Client : 封装了与Yarn RPC协议交互相关的库
5. Server : Yarn的核心实现 : RM,NM…

###### Yarn

1. ResourceManager(RM) :

全局资源管理器,负责整个系统的资源管理和分配.主要两个组件 :

调度器(Scheduler) : 根据容量,队列等限制条件,将系统中的资源分配给各个正 在运行的应用程序.资源分配单位 : Resource Container(简称Container).

常见调度器 : Capacity Scheduler , Fair Scheduler

应用程序管理器(Applications Master ASM) : 负责管理整个系统中所有应用 程序,包括程序提交,与调度器协商资源以启动AM,监控AM运行状态并在失败时重新启动它.

1. Application Manager(AM) :

\*与RM调度器协商申请资源

\*将得到的任务进一步分配给内部的任务

\*与NM通信以启动/停止任务

\*监控任务运行状态

1. NodeManager(NM) :

每个节点上的资源和任务管理器

\*beat heart to RM

\*接收并处理AM的Container启动/停止各种请求

1. Container

###### Protocol Buffers

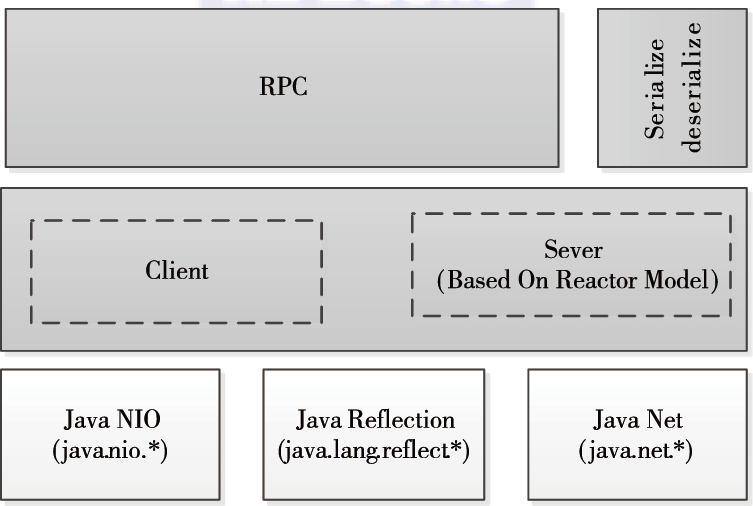
轻便高效的结构化数据存储格式,用于结构化数据序列/反序列化

###### RPC通信模型

客户机/服务器模型 : 请求程序是客户机,服务提供程序是一个服务器. 同步/异步

4个部分 : 序列化层,函数调用层(java反射,动态代理),网络传输层(Socket),服务器端处理框架. Java NIO

Hadoop RPC总体架构 :



RPC类(**package** org.apache.hadoop.ipc) :

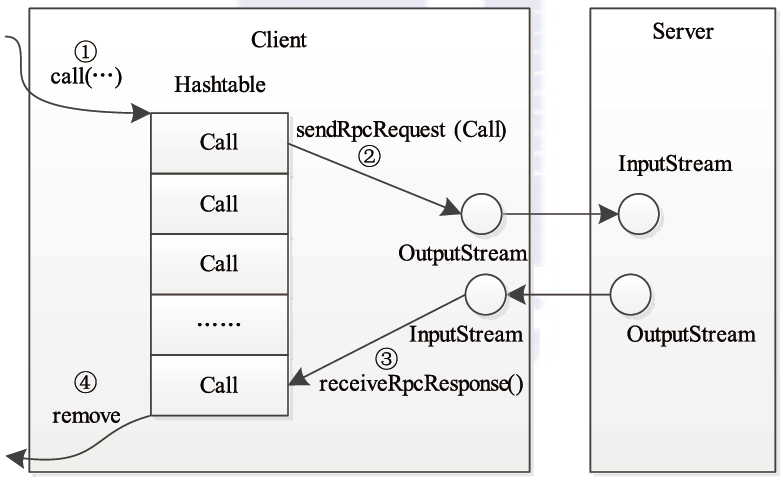
* interface RpcInvoker
* getProxy()
* RPC.Server
* RPC.Builder

Client(**package** org.apache.hadoop.ipc) :

* Call()
* 两个内部类 :

1. Call类 : 封装一个RPC请求
2. Connection类 : Client与每个Server之间维护一个通信连接,相关信息封装在此类中

类中主要方法call()方法源码逻辑流程 :



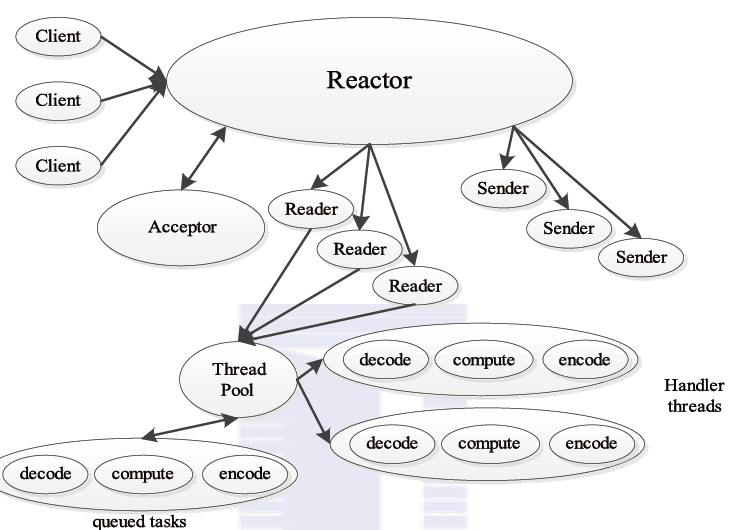
Server(**package** org.apache.hadoop.ipc) : 高并发,可扩展性 --> 线程池,事件驱 动,Reactor设计模式(\*反应器/堆设计模式)

Reactor设计模式主要包括几个角色 :

* Reactor : I/O事件的派发者
* Acceptor : 接收来自Client的连接,建立与Client对应的Handler,并向Reactor注册此Handler
* Handler : 与一个Client通信的实体,并按一定的过程实现业务的处理

内部进一步划分 : read,decode,compute,encode,send等过程

* Reader/Sender : 为加快处理速度,Reactor模式构建一个存放数据处理线程的线程池,数据读出后,立即扔到线程池中等待后续处理.为此,Reactor模式一般分离Handler中的读/写两个过程,分别注册成单独的读事件和写事件.由对应Reader/Sender线程处理.



**Server实际上是一个典型的Reactor模式.(学习Reactor模式例子记录在node-study项目中)**

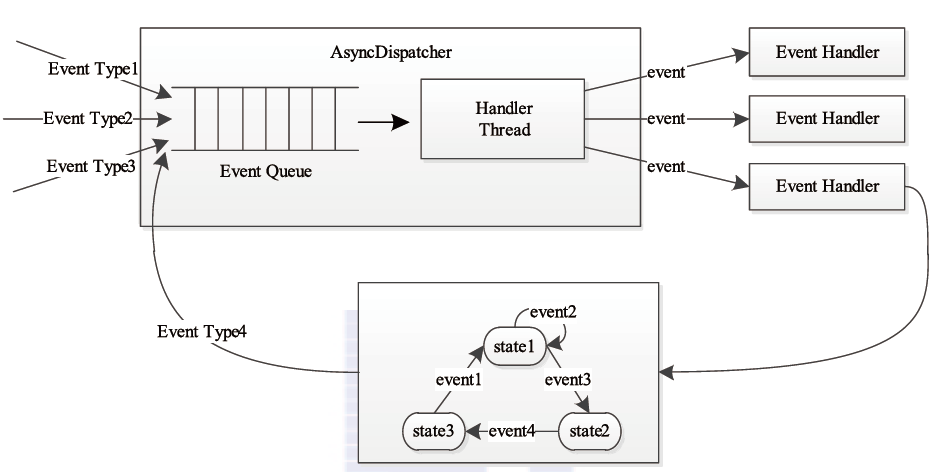
###### 服务库

非组合服务：AbstractService

组合服务：CompositeService

###### 事件库

处理逻辑抽象成事件和对应时间调度器,事件处理过程分成多个步骤,用有限状态机表示：



###### Yarn应用程序设计

设计三个协议：

ApplicationClientProtocol(Client与RM)

ApplicationMasterProtocol(AM与RM)

ContainerManagementProtocol(AM与NM)

客户端client:

Client与RM通信,ApplicationClientProtocol协议,具体实现：ClientRMService

通过RPC.getProxy()构建一个客户端对象

Records类：newRecord()构造一个可序列化对象

