今天和大家分享一篇 Eurosys2018的文章《Hyperledger Fabric: a Distributed Operating System for Permissioned Blockchains》。作者来自IBM苏黎世研究院。

先简单介绍一下 Hyperledger Fabric
的项目背景。Hyperledger超级账本项目是一个Linux基金会下的顶级项目。主要是孵化一些区块链工具、平台。
fabric是一个联盟链区块链底层平台。很大一部分开发者来自IBM。IBM对这个项目的贡献和支持力度都比较大。

文章对fabric的定义是一个联盟链的分布式操作系统。这篇文章会具体介绍fabric的架构和设计原理。

很多人对区块链已经有概念了。这里还是讲一下。

从数据结构的角度看，每一个区块中包含着一定数量排序过的交易。

对于整个区块链上的交易来说，交易的内容和顺序都是不可篡改的。

因为区块链是一个公共的分布式账本。很多的分布式节点通过共识协议，来达到账本内容的一致性、唯一性。

对于比特币来说，交易就是的虚拟货币的简单转移。
实际上交易不一定局限于虚拟货币。以太坊的智能合约，fabric的链码，都是为了实现分布式应用的通用交易而设计的。

2015年有人提出：智能合约是事件驱动的带有状态的程序，运行在共享的分布式账本上。

把智能合约引到状态机的概念上。
Replicated State Machines RSM 复制状态机 提出paxos帕克索斯 共识算法的Lamport 78年在研究分布式系统环境中达到一致性 就提出这个概念了。

复制状态机是针对单个可信应用。
如果采用容错协议，参与者被信任的。如果采用拜占庭协议，参与者可能是会作恶的。但是应用本身是可信的。

对于区块链来说。一个区块链上可能存在很多由第三方开发者开发的应用，应用不一定是可信的。

##
https://zh.wikipedia.org/zh-cn/%E7%8A%B6%E6%80%81%E6%9C%BA%E5%A4%8D%E5%88%B6
状态机复制

在计算机科学领域，状态机复制是实现容错服务的一种常规方法，主要通过复制服务器，并协调客户端和这些服务器镜像间的交互来达到目标。这个方法也同时提供了理解和设计复制管理协议的一套基本框架。
##

下面简单介绍一下区块链的发展。最初是比特币，硬编码了虚拟货币的应用。很难扩展。共识协议是POW 工作量证明。原生货币是比特币。

以太坊扩展了比特币应用。提出了分布式应用的概念。也就是智能合约。
能够支持任意的智能合约应用。但是同样依赖原生的以太币，采用pow协议。

fabric采用了以太坊智能合约的思路。但是在以太坊上，开发者需要用
特定领域语言solidity进行应用开发。
fabric支持开发者使用通用编程语言写智能合约，目前支持 GO java nodejs 。
fabric 是没有原生货币的。因为有一些场景不希望依赖货币。
模块化、可插拔的共识协议。

fabric的创新之处在于哪里呢？为了说明这一点，首先介绍一下目前区块链系统是如何运作的。

大部分区块链都是 先排序再执行 架构。

首先使用 共识协议 对 输入的交易进行排序。

然后每一个节点执行这些交易。

目前社区可用的很多复制状态机都遵循这种模式。包括 paxos 和许多拜占庭协议。

对于公链来说。每个节点是没有身份证明的。

排序执行的架构如下所示。

pow 协议下。矿工需要解决一个非常复杂且耗费资源的密码学难题。为了找到一个有效的区块。
区块的hash要小于某一个值。

当矿工发现了这个块，就通过gossip协议把区块信息广播到网络中。
其他人通过简单的hash来验证区块的有效性，也执行区块中包含所有交易。

这就是一个 排序执行的例子。首先用共识算法来排序区块，每个节点再执行区块中的所有交易。

在联盟链中，节点的身份是已知的。可以使用更加节省资源的共识算法，节点间也需要更多的通信交互。
比如用pbft算法。无需密集型的计算。但是需要更多的网络通信。
pbft算法 对消息的顺序达成共识，就可以执行。也是所有节点执行的。这也是一个 排序执行架构的例子。

下面重复一下 hyperledger fabric 希望实现的特性。
没有原生的虚拟货币。
用通用编程语言写分布式应用。
模块化的共识协议

也许看起来不是很直观。为了实现这些特性。fabric对架构做出了很大的改变。代码开源。

排序执行会产生什么问题呢？
每个节点都需要顺序执行智能合约或者交易。

没有可信的应用，每个人都可以编写。
比如执行很慢的应用、拒绝服务攻击，也被称为无限循环应用

以太坊通过使开发者为每一步计算操作支付gas。但是这和虚拟货币结合紧密。而fabric希望实现的是没有原生货币的区块链。

另一个是不确定性。几乎是所有的区块链或者复制状态机的前提假设都是：应用是确定的。
现在你赋予了别人 用通用编程语言 编写代码
的能力。这个前提就不成立了。代码就会包含不确定性。
比如系统时间是不确定的。

以太坊使用特定领域语言，用EVM虚拟机编译，来实现确定性

如果要支持通用编程语言

其他问题比如 硬编码的共识协议 已经提过了

机密性问题。在排序执行架构下，每个节点执行所有
合约；而有时候，你会希望只有一部分节点执行合约。
因此信任模型不灵活。

fabric 改变了这种 首先对输入的交易进行排序 ，然后用智能合约执行交易 的排序执行架构。
使用 执行 排序 验证的架构

我们首先执行。在这个步骤，为了最大限度地 消除不确定性，需要执行 来自应用开发者
的不信任的代码。
将执行转化为 应用对 状态的更新；状态是 一个 键值对 key value 的键值对；执行结果是
生成了一系列读写集 就是 read set 和write set。最后我们同意 这些
由潜在不确定性代码执行结果产生的更新。

第二个步骤是排序 ，但是我们不知道排序输入，我们只是对 读写集进行排序。是没有状态的。
因此排序服务 实际上看不到 交易 ，实际上是一个分布式的 排序者。

最后我们验证
下一页PPT会包含更多的细节。因此实际上，fabric中的应用由两部分组成。一部分是执行代码，我们可以称为链码。另一部分是 验证代码，我们称之为 背书策略。

那么这样的执行 排序 验证 架构 是什么样的呢？

首先由一个客户端开始，它把交易发送给一部分的节点，希望得到执行。这些节点被称为背书节点。
背书节点的数量取决于 分布式应用。这些节点然后执行代码。实际上，他们执行 不需要
持久化 任何计算结果。他们只是
模拟执行代码。链码存在于一个docker容器中。在键值对存储中的
本地状态上执行。每个节点上都有。

一旦节点模拟执行 ，就会将 签名后的response message，返回给客户端。response 是包含读写集的（读写集是链码合约执行的结果）。
客户端等待收集 足够数量的 消息。这个数量取决于 应用。因此这为 应用的信任模型 带来了 灵活性。实际上 解耦合了 排序服务的 信任模型。后面会提到一些例子。

客户端提交 读写集 给排序服务。读写集可以看做一个黑盒。排序服务看不到 交易的内容。实际上只是对 交易的顺序进行了排序。然后形成 区块链的数据结构。

排序后的结果被放在区块里，广播给所有节点。节点对交易的有效性进行验证。实际上会存在节点，不执行交易。但是所有节点都验证交易。

那么他们在验证什么？他们实际上在验证 是否满足了背书策略，即是客户端是否收集了足够的response。

这部分的代码可以被改变。 A应用可以选择 一个 背书策略。另一个B应用可以选择另一个背书策略。同时赋予 对读写集进行序列化操作的能力。

最后提交交易。在这个步骤上，交易最终有效，被持久化到每个节点上的本地状态中。

执行阶段 排序阶段 和验证阶段

回到fabric的分布式应用
由两部分组成，一个是执行代码。由第三方开发者编写。在共识前执行，有可能包含不确定性。

这不是说，开发者应该编写非确定性代码；在fabric架构下，非确定性的交易是无效的。比如要求两个执行（背书）节点以同样的方式执行交易，因为希望得到拜占庭容错。但是交易中有一个随机函数。无法使得交易以相同方式执行。
在之前的架构下，这样的代码是无法保证安全，可能导致区块链分叉。

第二部分是验证代码。

验证代码是无法由应用开者编写部署的。
只能由管理员修改，管理员的定义是 松散的。可以被定义为 网络中多数节点 ；且只能由链码 实例化。

举例说明：
N个链码背书者中的K个需要为交易背书。
我们也可以有 更复杂的背书策略。比如 需要获取某个特定密钥的签名。
文章中实现了 fabcoin，一种类似Bitcoin的数字货币，但是他是由中央银行发行的，没有对矿工进行奖励。
可以编写个性化的验证代码，但是代码部署方式受到保护。

fabric的架构混合了主动复制和被动复制。
我们可以看一下数据库中的主动复制和被动复制。
被动复制，比如执行之后同意更新的状态；主动复制，在共识之后验证。

fabric的共识协议的模块化的
目前排序服务有两种实现：一种是中心化，方便开发测试的；一种是 分布式容错协议 Lisbon大学实现的拜占庭协议原型系统。

生产级的拜占庭共识今年会实现。目前也在研究将一些顶会上的协议应用到fabric里。

网络情况、工作负载都会对fabric的性能造成影响。

这里以fabcoin为例，给大家一个直观的感受。

比起比特币每秒7笔交易，或者以太坊几十tps，fabcoin的性能取决于节点的CPU核数。最高可以达到 3500tps。如果把系统扩大到100个节点。对于很多应用来说，已经足够了。

很多的企业对fabric还是很关注的。