

EOS.IO 技术白皮书

作者: block.one

2017 年 6 月 5 号

翻译: Harvey 老狼、谭智勇、宋承根@OracleChain, 梓岑@YOYOW

本中文白皮书翻译自 EOS 白皮书英文版, 如果有表述不一致的地方, 以英文版本为准。

摘要

EOS.IO 软件引入了一种新的块链架构, 旨在实现分布式应用的性能扩展。这是通过创建一个可以构建应用程序的类似操作系统的架构来实现的。该软件架构提供帐户, 身份验证, 数据库, 异步通信以及在数以百计的 CPU 或群集上的程序调度。该技术的最终形式是一个块链体系架构, 该区块链每秒可以支持数百万个交易, 同时普通用户无需支付使用费用。

1. 背景

Blockchain 技术源于 2008 年推出的比特币，自那时以来，企业家和开发人员一直在努力推广该技术，以便在单个块链平台上支持更广泛的应用。

虽然一些通用区块链平台还在努力实现第一个能正常运行的区块链应用，针对特定场景的区块链应用诸如 BitShares 去中心化交易所（2014）和 Steem 社交媒体平台（2016）已经成为日活跃用户上万的成功应用。这两个应用成功的把性能提高到每秒数千个交易，延迟降低到 1.5 秒，降低交易费用，并实现了与中央服务器方案相似的用户体验。

由于现有的块链平台使用费用高昂，性能有限，阻碍了区块链应用的广泛传播。

2. 区块链应用的要求

成为一个成功的区块链应用平台块，应该需要满足以下要求：

支持百万级别用户

如 Ebay，Uber，AirBnB 和 Facebook 这样的应用，需要能够处理数千万日活跃用户的区块链技术。在某些情况下，应用程序可能无法正常工作，除非达到了大量用户，因此可以处理大量用户数量的平台至关重要。

免费使用

有时候应用开发人员需要灵活的为用户提供免费服务；用户不必为了使用平台而付出费用。可以免费使用的区块链平台自然可能会得到更多的关注。有了足够的用户规模，开发者和企业可以创建对应的盈利模式。

轻松升级和 Bug 恢复

基于区块链的应用程序在进行功能迭代的时候自然需要能支持软件升级。所有软件都有可能受到 bug 的影响。一个区块链底层平台在遭遇 bug 的时候，需要能够从 bug 中修复错误。

低延迟

及时的反馈是良好用户体验的基础。延迟时间如果超过了几秒钟，会大大影响用户体验，严重降低程序的竞争力。

串行性能

有些应用程序由于命令执行必须是顺序的，从而无法用并行算法进行实现。诸如交易所之类的应用经常需要处理大量的串行操作，因此一个成功的区块链架构需要具有强大的串行性能。

并行性能

大规模应用程序需要在多个 CPU 和计算机之间划分工作负载。

3. 共识算法（DPOS）

EOS.IO 软件架构中采用目前为止唯一能够复合上述性能要求的区块链共识算（DPOS）。根据这种算法，全网持有代币的人可以通过投票系统来选择区块生产者，一旦当选任何人都可以参与区块的生产。

EOS.IO 里预计每 3 秒生产一个区块。任何时刻，只有一个生产者被授权产生区块。如果在某个时间内没有成功出块，则跳过该块。

EOS.IO 架构中区块产生是以 21 个区块为一个周期。在每个出块周期开始时，21 个区块生产者会被投票选出。前 20 名出块者首选自动选出，第 21 个出块者按所得投票数目对应概率选出。所选择的生产者会根据从块时间导出的伪随机数进行混合。以便保证出块者之间的连接尽量平衡。

如果出块者错过了一个块，并且在最近 24 小时内没有产生任何块，则这个出块者将被删除。这确保了网络的顺利运行。

在正常情况下，DPOS 块链不会经历任何叉，因为块生产者合作生产区块而不是竞争。如果有区块分叉，共识将自动切换到最长的链条。具有更多生产者的区块链长度将比具有较少生产者的区块链增长速度更快。此外，没有块生产者应该同时在两个区块链分叉上生产块。如果一个块生产者发现这么做了，就可能被投票出局。

交易确认

由 DPOS 共识算法维护的区块链一般出块者都是 100% 在线的。这就是说一个交易平均 1.5 秒后，会被写入区块链中，同时被所有出块节点知晓这笔交易。这就意味着只需要 1.5 秒，一笔交易可以认定为 99.9% 被区块链接收了。

有一些非常情况下例如，软件 bug，Internet 拥塞或恶意出块者出现，区块链可能出现分叉。为了确保一个交易是不可逆转的，可以等待 15 个区块确认。根据 EOS.IO 软件的配置，在正常情况下 15 个区块确认平均需要 45 秒。

在分叉产生的 9 秒钟内，出块节点就可能发现这个分叉可能并警告用户。一个节点观察网络的时候如果发现连续 2 次的丢块事件，这意味着改节点由 95% 可能性在区块链的分叉分支上。有出现 3 个连续的丢块以后，该节点有 99% 的可能性在一条分叉出来的区块链上。可以生成一个预测模型，它将利用节点丢失的信息，最近的参与率以及其他因素来快速地警告用户出现什么问题。

对这种警告的反应完全取决于业务交易的性质，但最简单的反应是等待 15/21 确认，直到警告停止。

交易证明（TaPoS）

EOS.IO 软件要求每个交易都包括最近的区块头的哈希。这个哈希有两个目的：

1. 防止分叉区块链上出现大量交易记录；和
2. 使得系统能感知到用户是否在分叉出来的区块链上

随着时间的推移，所有用户最终直接确认块链，这使得难以伪造假冒链，因为假冒将无法从合法链路迁移交易。

4. 帐户

EOS.IO 软件允许使用唯一的长度为 2 到 32 个字符的可读的名称来实现对帐户的引用。该名称由帐户的创建者自行选择。所有帐户必须在创建时必须充入最小的帐户余额以支付存储帐户数据的费用。帐户名称还支持命名空间，因此帐户@domain 的所有者是唯一可以创建帐户@user.domain 的用户。

在去中心化的情况下，应用程序开发人员将支付创建帐户名义上的成本以注册新用户。通常企业已经以广告和免费服务等形式为获取每个客户花费了大量资金。相比之下，创建新的区块链帐户所需的资金成本是微不足道的。并且幸运的是，没有必要为已经由另一个应用程序注册的用户创建帐户。

消息和消息处理程序

每个帐户可以将结构化消息发送到其他帐户，并且可以定义消息被接受后的处理脚本。EOS.IO 软件为每个帐户提供其自己独有的数据库，只能由自己的消息处理程序访问。消息处理脚本还可以向其他帐户发送消息。消息和自动的消息处理程序的组合正是 EOS.IO 定义智能合约的方式。

基于角色的权限管理

权限管理主要涉及明确特定的消息是否被正确授权。权限管理的最简单形式是检查事务是否具有所需的签名，但这隐含着所需的签名是已知的。通常权力是与可以分类的个人或个人群组绑定在一起的。EOS.IO 软件提供了一个声明式权限管理系统，可以让帐户细粒度和高级别地控制谁在何时能够做什么。








至关重要的是，身份认证和权限管理被标准化实现，并与应用程序的业务逻辑分离。这使得开发某种工具以通用方式管理权限成为可能，并为性能优化提供了巨大的空间。

每个帐户都可以通过其他帐户和私钥的任何加权组合来控制。这种机制创建了一个能够真实反映权限在现实中的组织情况的层次化权限结构，并使得多用户对资金的控制比以往任何时候都更容易。多用户控制是提升安全性的最重要因素，如果能正确地使用，可以极大地消除黑客盗窃的风险。

EOS.IO 软件允许帐户可以定义与其他帐户密钥的“and”和“or”的组合，并且把这个组合以将特定类型的消息发送到另一个帐户。例如，可以为用户的社交媒体帐号提供一个密钥，另一个用于交易。甚至用户可以给予其他帐户许可让其代表自己的帐户行事，而无需向其他帐户分配密钥。

命名权限级别

使用 EOS.IO 软件，帐户可以定义命名权限级别，每个权限级别可以从更高级别的命名权限派生。每个命名权限级别定义一个权力，这个权力可以是其他帐户的密钥和(/或)命名权限级别组成的多签名检查的阈值。例如，帐户的“朋友”权限级别可以设置为帐户能被其任何朋友帐户平等地控制。

Authorities		
Signing		
	STM7xh66F5ZHyfN9u4rNZmTioBteZhvWdqDwaR2kR55tBXeCjTr2z	
Owner		
	STM7iTj8quuiqX7aUHZWrYXfAqQTDbpL8FwxoCUzEeKE745mvBY41	
Active		
	STM5DiwrKfp4ngW7fWcDej1Kd3efogYSGxsMKfkz3xi4AsNGYWU2D	
Posting		
	streemian	1 33.3%
	esteemapp	1 33.3%
	STM89kBiwp15R8CBrgAFWd5p5uayTvvS7B6Zb4oBenWx8ChjeLMTf	1 33.3%
Threshold		1 33.3%
Memo		
	STM7S4xvQgdvuQ8SFAD8vzFcLskoUdLqzEPbudcXuyoku2i.rwzt6v	

另一个例子是 Steem 区块链，其具有三个硬编码命名的权限级别：*owner*，*active* 和 *posting*。*Posting* 权限只能执行诸如投票和发布等社交行为，而 *active* 权限可以做除了更改所有者的所有事情。*owner* 权限实质是被保留了起来，它能够做所有一切。EOS.IO 通过允许每个帐户持有者定义自己的权限层次结构以及动作的分组来实现类似的管理理念。

命名消息处理程序组

EOS.IO 软件允许每个帐户将自己的消息处理程序以命名嵌套的方式予以组织。这些命名的消息处理程序组可以通过配置其权限级别被其他帐户引用。

最高级别的消息处理程序组是处理帐户名称的程序组，最低级别的消息处理程序组是处理该帐户正在接收的单独消息类型的程序组。这些程序组的引用格式为：`@accountname.groupa.subgroupb.MessageType`。

在这种模式下，可以将创建和取消订单的交易合约与存取款的交易合约分离。这种交易合约的分组对用户使用交易合约提供了较大便利。

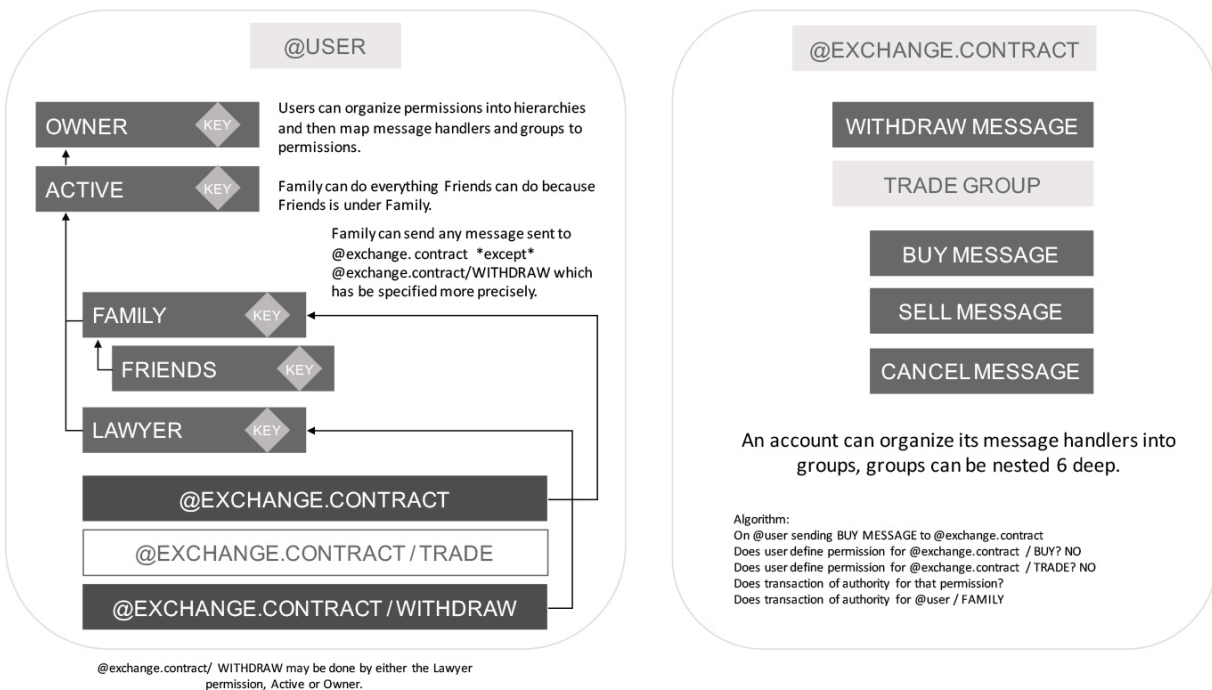
权限映射

EOS.IO 软件允许每个帐户定义任何帐户的命名消息处理程序组与其自己的命名权限级别之间的映射。例如，账户持有人可以将账户持有人的社交媒体应用程序映射到帐户持有者的“朋友”权限组。通过此映射，帐户的任何朋友都可以和帐户持有者一样，在帐户的社交媒体上发布内容。即使他们将作为帐户持有者发布，他们仍然会使用自己的密钥来签名。这意味着总是可以辨别出来哪些朋友以何种方式使用了其帐户。

权限评估

当从@alice 到@bob 传送 “Action” 类型的消息时，EOS.IO 软件将首先检查@alice 是否为@bob.groupa.subgroup.Action 定义了权限映射。如果没有找到任何结果，那么将会检查@bob.groupa.subgroup，然后是@bob.groupa，最后是@bob 的映射。如果此时仍然没有找到匹配的结果，那么假定的映射将是命名的权限组@alice.active。

一旦识别出权限映射，则启动多签名阈值校验过程。如果校验成功，所命名的权限将与关联的权限建立关联。如果失败，那么它会遍历父权限，最终遍历到其所有者的权限@alice.owner。



默认权限组

该技术还允许所有帐户都有一个可以做所有事情的 “owner” 权限组，和一个除了更改所有者组之外可以执行所有操作的 “active” 权限组。所有其他权限组均派生自 “active” 权限组。

权限的并行评估

权限评估是个 “只读” 的过程，即使在事务过程中对权限进行了修改，在运行到区块结尾时这种修改也会失效。首先，这意味着所有事务的所有密钥和权限评估可以并行执行。其次，这种机制意味着可以快速验证权限，而不需要考虑启动可能回滚的昂贵的应用程序逻辑。最后，这意味着当挂起的事务继续执行时，事务权限的评估可以继续执行，而无需重新执行。

这么设计考虑的主要原因，是因为权限验证占据交易验证的很大计算量比例。使之成为一个只读和可并行化的过程，可以显着提高性能。

当区块链消息被重放，来从消息日志中重新生成确定状态时，并不需要再次评估权限。事务被包含在一个已知的状态良好的区块中的事实使其可以跳过这个步骤。这极大地减少了重放之前的区块链数据相关的计算量。

有强制延迟的消息

时间是安全的关键组成部分。在大多数情况下，在私钥被使用前不可能知道其是否已经被盗用。基于时间的安全机制在人们使用一些特殊类型应用程序时更为关键，这些应用程序需要将密钥保存在连接到互联网的人们日常使用的计算机上。EOS.IO 软件支持应用程序开发者指定某些消息在包含在区块后，实际应用之前必须等待一段比较小的时间段。在此期间，这些消息可以被取消。

当这类消息被广播时，用户可以通过电子邮件或短信收到相应通知。如果他们不授权该消息，那么他们可以登录其帐户来还原帐户数据并撤回消息。

所需的延迟取决于操作的重要程度。支付一杯咖啡可以毫不拖延地在几秒钟内确认，而买房子可能需要 72 小时清算周期。将整个帐户转移到新的控制者手上可能最多需要 30 天。具体延迟的选择取决于应用程序开发者和用户。

密钥被盗后的恢复

EOS.IO 软件为用户提供了一种在密钥被盗时恢复其帐户控制的方法。

帐户所有者可以使用在过去 30 天内活动的任何其批准的帐户恢复合作伙伴的密钥，在其帐户恢复合作伙伴的允许后，重置其帐户上的所有者密钥。在没有帐户所有者的配合情况下，帐户恢复合作伙伴无法重置其帐户的控制权。

对于攻击帐户的黑客而言，由于其已经“控制”该帐户，因此尝试执行恢复过程没有任何收获。此外，如果他们确进行恢复的过程，那么恢复合作伙伴可能需要身份认证和多因素认证（如电话和电子邮件）。这或者会暴露黑客的身份，或者黑客在恢复过程中毫无所得。

这个过程与简单的多重签名机制有极大的不同。通过多重签名的交易，会有一个对象会执行并参与每一个交易；而通过恢复过程，恢复过程的操作者仅参与了恢复的过程，并没有权力参与日常的交易。这极大降低了相关参与者的成本和法律责任。

5. 应用程序的确定性并行执行

区块链共识取决于确定性（可重现）行为。这意味着所有并行执行都能使用互斥体或其他锁的原语的情况下正常运行。没有锁，必须有一些方法来保证所有帐户只能读写自己的私有数据库。这也意味着每个帐户会顺序处理其消息，并行性确定在帐户级别。

使用 EOS.IO 软件，区块生成器的工作是将消息传递到独立的线程中，以便它们可以并行地评估。每个帐户的状态只取决于传递给它的消息。调度表是区块生成器的输出，并且将被确定性地执行，但是生成调度的过程不必是确定性的。这意味着区块生成器可以利用并行算法来调度事务。

并行执行还意味着当脚本生成新消息时，它不会立即发送，而是在下一个周期中发送它。无法立即发送的原因是因为接收方可能会在另一个线程中主动修改自己的状态。

通信延迟优化

延迟时间是一个帐户将消息发送到另一个帐户并收到响应所需的时间。EOS.IO 软件的目标是使两个帐户能够在单个区块内来回交换消息，而不必在每个消息之间等待 3 秒。为了实现这一点，EOS.IO 软件将每个区块分为周期（cycle）。每个周期分为线程（thread），每个线程包含事务列表。每个事务包含一组要传递的消息。该结构可以被可视化为树，其中各层依据其特性被顺序或者并行地进行处理。

区块 Block

周期 Cycles（顺序）

线程 Threads（并行）

交易 Transactions（顺序）

消息 Messages（顺序）

接收方和通知的帐户 Receiver and Notified Accounts（并行）

在一个周期中生成的交易可以在任何后续周期或区块中传送。区块生成器将不断把周期添加到区块中，直到最长的区块时间间隔达到，或者没有新的可传送交易生成。

可以使用区块的静态分析来验证在给定周期内是否存在两个线程包含修改同一个帐户的事务。只要这种静态分析机制一直起作用，就可以通过并行运行所有线程来处理区块。

只读消息处理

部分账户可能会处理一些只需要决定通过与否的消息，而不会改变自己内在状态。这种情形下，只需要有一个或多个进程包含这个特殊账户下的只读消息处理器，这些处理就能并行进行。

多账户原子交易

有时，我们希望确保消息被多个帐户以原子方式交付和接受。在这种情况下，两个消息被放置在一个交易中，两个帐户将被分配相同的线程和消息按顺序执行。这种情况在性能上并不理想，并且当涉及到“付费”用户的使用时，他们将会被根据交易所涉及的特殊帐户的数量来收费。

出于性能和费用的考虑，最好将涉及两个或更多帐户的原子操作最小化

部分区块链状态评估

大规模区块链技术组件应该是模块化的。每个人都不应该运行所有东西，特别是如果他们只需要使用一小部分应用程序的时候。

出于将交易状态显示给用户的目的，交易应用的开发者将维护一个完整的节点。这款交易应用不需要与其他社交媒体应用关联状态。EOS.IO 系统允许任何完整的节点选择性的运行任意应用子集。传递给其他应用的消息将被安全地忽略，因为应用的状态完全来自于传递给它的消息。

这对于多帐户之间的通信有一些重要的影响。最重要的是，不能假定另一个帐户的状态在同一台机器上是可访问的。它还意味着，虽然允许一个帐户同步调用另一个帐户的“锁”是一种诱人的设计模式，但如果其他帐户不在内存中，这种设计模式将会崩溃。

因此，所有帐户间的状态通信必须通过区块链上的消息进行传递。

自主最优任务安排

EOS.IO 系统不能强制阻止区块生成者向其他帐户发送的任何消息。每个区块的生成者对处理交易的计算复杂度和时间复杂度都有自己的主观度量，无论这个交易是由用户生成的还是由脚本自动生成的。

在网络层面，EOS.IO 系统处理的每一笔交易都有固定的计算带宽成本，不管它是耗费 01ms 还是 10 ms 来处理它。但是，使用该系统的每个单独的区块生成者会使用它们自己的算法和度量来衡量资源使用。当一个区块生成者发现一个交易或帐户已经消耗了大量的计算能力时，他们会在生成自己的块时拒绝该交易；但是，如果其他区块生成者认为它是有效的，他们仍然会处理该交易。

一般来说，只要一个区块生成者认为一个交易是有效的，并且所消耗的资源是可控的，那么其他所有的区块生成者也会接受它，但可能要花费 1 分钟才能使该交易传播到这个区块生成者处。

在某些情况下，区块生成者可以创建一个区块，其中包括在可接受范围之外的交易。在这种情况下，下一个区块生成者可能会选择拒绝这个区块，而这条线路将会被第三个区块生成者终结。这与一个大区块导致网络传播延迟所引发的情况没有什么不同。社区会注意到这种异常模式，并最终清理该流氓区块生成者的选票。

这种对计算、资源成本的主观评估将使区块链不必精确地去度量运行一个任务需要多长时间。有了这个设计，就不需要精确地数指令，这将极大地增加优化的机会，而不会打破共识。

6. 令牌模型和资源使用

所有的区块链都是资源受限的，并且需要一个系统来防止滥用。在 EOS.IO 系统中，有三大类资源被应用程序消耗：

1. 带宽和日志存储(磁盘)；
2. 计算和计算积压(CPU)；
3. 状态存储器(RAM)。

瞬时使用和长期使用的这两类组件都会消耗带宽和计算。区块链系统将维护所有消息的日志，这些日志将会被所有的完整节点下载和存储。通过日志信息，可以重构所有应用程序的状态。

可计债务是指通过对消息日志重新生成状态的计算消耗。如果可计债务增长太大，就有必要对区块链的当前状态进行拍照，并抛弃区块链的历史状态。如果计算债务增长过快，区块链将使用 6 个月的时间来重放 1 年的交易。因此，对可计债务进行精心管理是至关重要的。

区块链存储的状态指那些可从应用程序逻辑访问的信息。它包括诸如订单和帐户余额等信息。如果应用程序不读取该状态，则不应该存储它。例如，博客文章的内容和注释不被应用程序逻辑读取，因此它们不应该存储在区块链的状态中。与此同时，邮件或评论的索引、投票数和其他属性将作为区块链状态的一部分被存储起来。

区块生成者可以发布它们可用的带宽、计算资源和状态的容量。EOS.IO 系统允许每个帐户在一个 3 天对赌合约中消耗一定比例的可用容量。例如，假设一个基于 EOS.IO 系统的区块链应用启动，如果一个帐户持有该区块链提供的总令牌的 1%，那么这个帐户就有可利用该区块链 1% 的状态存储容量。

使用 EOS.IO 系统，带宽和计算能力将被分配到部分储备基础中，因为它们是短暂的(未使用的容量不能存储下来为将来使用)。EOS. IO 系统将使用类似于 Steem 的算法来限制带宽使用速率。

目标和主观度量

正如前面所讨论的，可度量计算的使用对性能和优化有很大的影响；因此，所有资源的使用限制最终都是主观的，并且根据各自的算法和估计来执行。

也就是说，从客观角度来说，有些事情是微不足道的。站在客观的角度，传递的消息数量和存储在内部数据库中的数据的大小都是很便宜的。EOS.IO 系统允许区块生成者能够在这些客观度量上应用相同的算法，但是也可以在主观度量上选择更严格的主观算法。

接收方支付

传统上，它是用来支付办公空间、计算电力以及运营业务所需的其他费用的业务。客户从该业务中购买特定产品，而这些产品的销售收入将用于支付业务成本。同样，没有任何网站要求访问者为维护服务器而支付小额费用。因此，分散的应用程序不应该强迫它的客户直接为使用区块链支付费用。

EOS.IO 系统不要求用户直接向区块链支付，因此不限制或阻止企业确定其产品的货币化策略。

授权能力

如果一个区块链是使用 EOS.IO 系统开发的，而其令牌是由一个持票人持有，他可能不需要立即消耗全部或部分可用带宽，这样的持有者可以选择将未消耗的带宽给予或租给他人；使用 EOS.IO 系统的区块生成者将识别这样的授权并直接分配相应的带宽。

将交易成本与令牌价值分开

EOS.IO 系统的主要优点之一是，应用程序可用的带宽完全独立于任何令牌价格。如果应用程序所有者持有相应数量的令牌，那么应用程序可以在固定的状态和带宽使用中持续运行。开发人员和用户不会受到令牌市场价格波动的影响，因此不会依赖于价格。EOS.IO 系统运行区块生成者能够自然地增加带宽、计算资源和每个令牌的可用性，这与令牌的价值无关。

EOS.IO 系统将奖励那些生成了区块的区块生成者一定的令牌。令牌的值将影响一个区块生成者能够购买的带宽、存储和计算量；这个模型自然会利用上升的令牌价值来提高网络性能。

状态存储成本

因为带宽和计算资源可以被委托，因此应用程序状态的存储要求应用的开发者持有令牌直到该状态被删除。如果状态不被删除，那么令牌将被有效地从循环中删除。

每个用户帐户需要一定数量的存储空间；因此，每个帐户必须保持最低余额。随着网络存储容量的增加，最低余额的要求将会下降。

块奖励

每次生成一个块时，EOS.IO 系统都会奖励该区块生成者一个新的令牌。所创建的令牌数量由所有区块生成者所公布的期望报酬的中位数决定。EOS.IO 系统可能被配置为限制区块生成者所得奖励上限，这样，令牌供应的年总增长不超过 5%。

社区福利应用

除了加入以 EOS.IO 系统为基础的区块生成者团队，用户还可以选择 3 个社区福利应用，也称为智能合约。这 3 个应用程序最多能按配置的比例接收到每年的令牌配额减去已支付给区块生成者的部分。这些智能合约将根据每个应用程序从令牌持有者收到的选票比例来收取令牌。经选举的应用程序或智能合约可以由新当选的应用程序或令牌持有人的智能合约所替代。

7. 治理

治理是人们在主观问题上达成共识的过程，而这些问题不可能完全被软件算法所捕获。EOS.IO 系统实现了一个治理过程，有效地影响到现有的区块生产商。在被定义治理流程之外，之前的区块链依赖于临时的、非正式的、经常有争议的治理过程，从而导致不可预知的结果。

EOS.IO 系统认识到，治理权力源来自于将权力代理给区块生成者的令牌持有者。区块的生成者被给予有限的和被监督的权限来冻结帐户，更新有缺陷的应用程序，并提出对底层协议的变更。

EOS.IO 系统的一部分是区块生成者的选举。在对区块链进行任何更改之前，这些区块生成者必须批准它。如果区块生成者拒绝做出让令牌持有人所期望的改变，那么他们可以被投票否决。如果区块生成者未经令牌持有者允许进行更改，那么所有其他非生产的全节点验证器(交换器等)将拒绝更改。

冻结账户

有时，智能合约的行为会发生异常或不可预知，无法按照预期执行；有时应用程序或帐户可能发现一个漏洞，使其消耗不合理的资源。当此类问题不可避免地发生时，区块生成者应当有能力纠正这种情况。

所有区块链的区块生成者有权选择哪些交易被包含在区块中，从而使他们有冻结帐户的能力。EOS.IO 系统通过冻结一个帐户到 17 / 21 活跃区块生成者的投票结论中，使这一授权成为正式结论。如果生成者滥用权力，他们可以被淘汰，帐户将被解冻。

改变帐户代码

当其他一切都失败了，而“不可阻挡的应用程序”以一种不可预知的方式运行时，EOS.IO 系统允许区块生成者在不需要硬分叉整个区块链的情况下替换帐户的代码。与冻结帐户的过程类似，此代码的替换需要 17 / 21 被选中的区块生成者的投票。

宪法

EOS 操作系统可以用区块链技术签名用户之间建立 P2P 服务协议或约束性合约，也就是所谓的“宪法”。宪法内容定义了仅依靠代码无法完全执行的用户间义务，同时结合相互间的公认规则，确立司法权和适用法律。每一个在网络中签名广播的交易，其签名信息中必须包含宪法的哈希值，以明确约束合约签名者。

宪法还定义了源代码协议的人类可读性 **intent**（意图）。当出现系统错误时，**intent**（意图）可用来区分这个错误是 **bug** 还是系统特性，并且判断社区对此的修复措施是否正确。

升级协议和宪法

EOS 操作系统使用源代码定义宪法和协议，同时也定义了宪法及协议的更新方法。对宪法或协议进行变更，需要完成以下步骤：

1. 区块生产者（译注：**miner/delegate/witness**，因此没有译作矿工）提交一个宪法变更动议，并获得 17/21 以上的赞成票；
2. 区块生产者将 17/21 以上的赞成票维持连续 30 天；
3. 要求所有用户都使用新宪法的哈希值确认交易；
4. 区块生产者采用修改源代码的方式反映宪法变更，使用 **git** 提交的哈希值将变更提交到区块链上；
5. 区块生产者继续将 17/21 以上的赞成票维持连续 30 天；
6. 变更的代码 7 天后生效，源代码修改通过后，将有 1 周的时间来对所有节点的进行升级；
7. 所有没有升级为新代码的节点将自动关闭。

根据 EOS 操作系统的默认配置，更新区块链来添加新功能这一进程需要 2 到 3 个月时间，而修复那些不需要更改宪法的非关键性漏洞需要 1 到 2 个月时间。

紧急变更

面临一个损害用户利益的有害漏洞或安全漏洞时，区块生产者可以加速宪法变更过程。一般来说，加速新特性更新过程或修复无害漏洞，都是违反宪法的行为。

8. 脚本&虚拟机

EOS 操作系统将首先作为一个传递账户间已认证信息的平台。脚本语言和虚拟机的实现将独立于 EOS 操作系统技术，任何开发语言或虚拟机，只要有适当的、性能足够的沙箱，都可以通过 API 与 EOS 集成在一起。

模式定义的消息

在账户之间发送的所有消息都是由区块链共识状态的一个模式定义的，该架构允许消息在二进制和 JSON 格式之间的无缝转换。

模式定义的数据库

数据库状态也使用类似的模式定义，这确保所有应用程序存储的数据都以一种格式呈现，同时具备 JSON 的人类可读性，以及二进制格式的高效率存储和易操作性。

将身份验证与应用程序分离

为了最大化并行运算，同时将从程序日志中重新生成应用程序状态的计算任务降至最低，EOS 操作系统将验证逻辑分为三个部分：

1. 确认消息在内部是一致的；
2. 确认所有的前置条件都是有效的；
3. 修改应用程序状态。

验证消息的内部一致性是只读的，不需要访问区块链状态，这意味着它可以最大化并行运算来执行。验证前置条件（例如需求平衡）也是只读的，因此也可以从并行运算中获益。只有对应用程序状态进行修改才需要写访问，并且需要按顺序对每个应用程序进行处理。

身份验证是验证消息是否可以应用的只读过程，应用程序实际上就是在做这项工作。实时的计算都需要执行，但交易一旦被包含在区块链中，就不再需要执行身份验证操作了。

虚拟机独立架构

这是 EOS 操作系统软件的目的是可以支持多种虚拟机，同时可以随着时间推移持续按需求增加新的虚拟机。出于这个原因，本文不会讨论任何特定语言或虚拟机的细节，但即便如此，目前也已经有三种虚拟机正在评估接入 EOS 系统。

Wren

Wren(<http://wren.io>)是一种小型的、快速的、基于类别的编程语言。Wren 的开发人员将其描述为“就像是把小型 tak 文件装进 lua 大小的软件包，再加上一点 Erlang 特性，再包进一个熟悉的、现代的语法里面”。之所以选择 Wren 语言和虚拟机，是因为它的短小精悍、易于文档记录和理解的代码库。它还具有非常好的性能，并且可以很容易地嵌入 C++ 应用程序中。

Web 组件(WASM)

WASM 是构建高性能 Web 应用程序的新兴 Web 标准，通过少量适配就可以被明确定义和沙箱化。WASM 的好处在于业界广泛支持，因此可以用熟悉的语言开发智能合约，例如 C 或 C++。

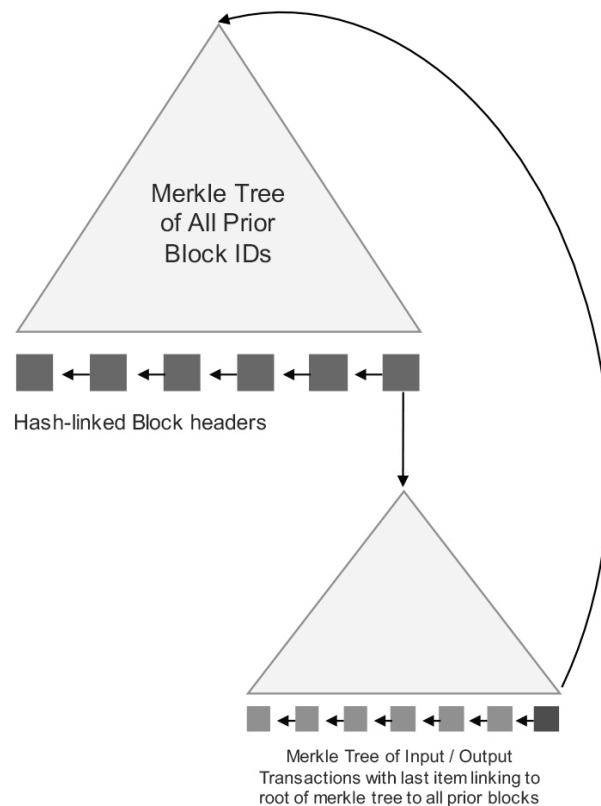
以太坊人员已经开始适配 WASM，以提供适当的沙箱并使用以太坊 WASM 定义 (<https://github.com/ewasm/design>)。这种方法很容易改编后用于 EOS 系统软件集成。

以太坊虚拟机(EVM)

这个虚拟机已经被用于大多数现有的智能合约，并且可以在 EOS 系统区块链上使用。可以想象，在 EOS 操作系统区块链上，EVM 合约可以在内部沙箱中运行，只需要少量适配就可以与其他 EOS 应用程序交互。

9. 跨链交互

EOS 操作系统旨在促进区块链间的跨链交互，这是通过简化消息存在证明和消息序列证明来实现的。这些证明与围绕信息传送的应用架构设计相结合，同时可以隐藏跨链交互和验证的细节，避免向应用程序开发人员公开。



用于轻客户端验证的默克尔证明(LCV)

要更容易地与其他区块链集成，对于客户端而言最好是不需要处理全部的交易。毕竟，一个交易所关心的不过只是入账和出账操作。更进一步而言，一个更加理想的状态是，对于交易所自身所维持的链来说，如果可以将轻量级的默克尔存款证明应用其中，那么就不必完全依赖全节点矿工，全节点矿工同步时也能维持尽可能小的开销。

LCV 的目标是能产生相对轻量级的交易存在证明，并且该证明能被其他人通过跟踪一个轻量级数据集进行验证。既然如此，目的就是证明一个特定的交易是被一个特定的区块包含其中，并且这个区块是被包含在已经验证的特定区块链历史中。

比特币的轻量级验证方式是，假设所有节点都有读取区块头数据完整记录的能力。而区块头数据每年增长 4MB，假设每秒产生 10 笔交易，一个有效的证明需要 512 bytes，这对于一个出块时间为 10 分钟的区块链来说是可行的。但对于一个出块时间为 3 秒的区块链来说则远远不够。

EOS 操作系统的轻量级证明只需要验证包含某个特定的不可逆交易之后的区块头数据，使用哈希链表架构（如下图），数据集可以保持在 1024 bytes 内，即可证明任何一个交易是否存在。这是基于验证节点保留着前一天的所有区块头数据（2 MB 大小），然后证明这些交易只需要 200 bytes 大小的证明数据。

当生成区块时候使用合适的哈希链表时，使用这种方法只会带来很小的增量开销，这意味着没有理由不以这种方式去生成区块。

当与其他链验证证明的时候，时间、空间和带宽都有很大的优化空间。跟踪所有区块头数据(420 MB/年)可以使证明体积尽可能小。只跟踪最近的区块头可以使得在持久区块头数据保存体积以及证明体积之间获得平衡。同样的一个区块链可以“懒惰地”只记录过去数据的哈希值作为之前数据的证据，新证明只需要保留已知的 **sparse tree**（稀疏树）结构，具体的方法会视乎于外部区块所占的默克尔证明所包含的交易比例。

在链与链之间经过一定密度的相互关联之后。他们将会变得越来越高效。一条链可能会包含另外一条链的全部历史记录，那么就不再需要互相证明。从性能的角度来说，这将极大地减少链间互相证明操作的频率。

跨链通信延迟

与其他区块链通信的时候，矿工必须等待其他区块链不可逆地确认之后才会接受其为有效的输入。使用 EOS 系统软件，凭借出块时间为秒的委任权益证明以及 21 个矿工，这大概只需要 45 秒的确认时间。如果某个链上的矿工不等到交易确认，就像一个交易所接受了一笔存款而后又撤销这笔操作，这会影响这条链共识的有效性。

完成证明

使用外部区块链的默克尔证明时，知道所有已处理的交易是有效的和知道有没有交易被忽略，这两者之间有巨大的不同。因为不可能证明所有最近交易是已知的，但有可能证明历史交易数据之间没有缺失。EOS 操作系统通过分配一个顺序的标识编号给每一笔到达账户的信息来完成这个功能，用户可以使用这些标号来证明所有给这个账号的消息已经被处理并且是被按顺序处理的。

10. 结论

EOS 操作系统是基于经过普遍证实、并通过长期实践考验的概念来设计的，代表着区块链技术的根本性进步。它是可扩展的全球性区块链社会的宏伟蓝图的一部分，分布式应用程序可以轻松地以此为环境开发和管理。