初链：高性能去中心化公开账本

（初稿）工作进行中

初链研究小组ARCHIT SHARMA1, JASPER L2, HAN ZHANG3, ERIC ZHANG3
1*ARCHIT@PM.ME* 2JASPER@TRUECHAIN.PRO 3HAN.ZHANG2@GMAIL.COM 4 ERIC@TRUECHAIN.PRO

摘要：本文介绍了初链共识协议的初步设计和一些其他技术细节。简单地说，我们的共识设计具有一致性、活跃度、交易终止性和安全性保障，一个真正意义上的混合共识。我们讨论了一些优化，如轮换委员会的更新频率和物理时间戳限制。我们接着提出了在以太坊上建立一个新的虚拟机的构思，它在一个没有许可的环境中增加了基于许可链的交易处理能力。我们还使用数据分片和投机性交易的概念，评估在混合云基础设施中智能合约的运行情况，使用现有的志愿计算协议来实现我们引入的奖励基础结构。

在下一个版本的黄皮书中，我们将正式地讨论其中的一些特性以及本文末尾列出的一些未来方向。

1 介绍

随着加密货币的日益普及，区块链技术引起了业界和学术界的关注。我们可以将区块链看作一个共享的计算环境，运行着多种共识机制，该环境下的所有节点可以自由地加入和退出。区块链的去中心化特性，以及交易透明性、自治性和不可篡改特性，对于加密货币来说是至关重要的，这些特性为此类系统定下了基调。然而，早期设计的加密货币，如比特币[21]和以太坊[11]，在交易效率方面已被广泛认为是不可扩展的，而且在经济上也不可行，因为它们需要大量的能源消耗和计算能力。

随着在现实世界中使用公链上的应用程序和平台的需求不断增长，支持更高交易效率的可行性协议是新一代公链系统的主要关注点。例如，考虑到一个通用的公链系统，它可以在一个非常庞大的用户基础上托管CPU密集型点对点游戏应用程序。在这样的链中，除了数字广告应用外，如果它也能够为ICOs（初始代币发行）提供智能合约，那么我们很容易预计交易确认时间出现巨大的延迟。

还有一些其他模型，如权益证明的委托机制(POS)和允许拜占庭容错协(BFT)。只要系统中每次都不能超过三分之一的参与者是有意或无意的恶意节点，那么BFT共识协议就能够确保该系统的安全性。这是一个非常好的机制，但是单独的基于BFT共识的公链存在可扩展性和伪去中心化的问题。使用少量验证节点的POS协议尽管可以提供高吞吐量，但是系统本身高度依赖于少数的利益相关方来决定是否包含和排除委托。此外，因为它没有使用默克尔树数据结构，数据根本就没有透明度，同时这类机制总是会遭受无端的悖论的影响。

在本文中，我们提出了TrueChain，一种混合共识协议[24]，它结合了一种改进版的PBFT(实用拜占庭)[13]和POW(工作量证明)共识。POW共识确保了激励和委员会的选举，而PBFT层则承担一种具有瞬时处理高吞吐量事物，交易验证，公平交易贸易委员会的成员轮值功能的

高效共识机制，以及作为一种补偿基础设施去处理不同的基础设施。混合共识机制的特性允许它最大限度地容忍三分之一对等节点的腐败。

2 背景

本方案的核心优势在于对由Pass和Shi提出的混合共识[24]论文中理论的认知和应用。从这篇论文中，我们发现有很多设计空间还可以进一步优化。使用DailyBFT作为委员会成员允许委员会轮值功能，为节点间的共识验证提供了更好的公平性。

POW节点可以从激励性基础设施中获益，同时它们也是慢链的一部分，有助于部署智能合约。

2.1相关工作:

混合共识遵循一种设计范式，即BFT和PoW结合在一起，使其在两个方面的优良特性都得到最好的体现。一般来说，混合共识将利用BFT协议作为快速处理大量传入交易的快捷途径。在默认情况下，BFT协议应用于一个许可的设置中，里面所有的身份都是先验已知的。PoW协议选择BFT委员会成员的依据是csize(挖出的区块数量)和节点权益的结合。这就提供了一种必要的准入系统，以处理动态的成员以及在许可的环境下切换委员会。

3 共识

我们的共识设计主要基于Pass和Shi[24]的混合共识，并进行了一些修改和改进，以便为我们关注的应用场景量身打造。在本节中，我们假设读者熟悉混合共识协议的每个细节。

3.1设计概述

在本小节中，我们将概述我们的共识协议。在这个协议中，我们在[24]中使用相同的抽象符号和定义。在下一节中，我们将在混合共识的基础上解释我们的修改和进一步的构建。

我们的对手模型遵循[24]中的假设，即允许对手轻微地自适应地破坏任何节点，而破坏不会立即生效。在下一个版本的黄皮书中，我们将正式地解释我们在通用可组合模型[12]中的修改。

请注意，为了便于解释，本文中的所有伪代码都被简化了。他们没有对工程进行优化。

3.2 混合共识协议的概述

在本小节中，我们回顾混合共识协议中的主要内容和定义。

3.2.1 水果链

我们采用水果链作为慢链(snailchain),用以代替Nakamoto(传统链),以抵御1/3−ε腐败(inhashpower)随机小常数ε,以获得最优的适应性。

3.2.2 Daily 链下共识协议

在DailyBFT中，委员会成员运行一个离线的BFT实例来决定每天的日志，而非成员则统计委员会成员的签名数量。它将安全性扩展到非委员会成员和后期生成节点。它附带一项终止协议，该协议要求所有诚实的节点在终止时同意相同的最终日志。在DailyBFT中，委员会成员输出签名的每天的日志哈希，然后被混合共识协议使用。这些签名的日志哈希满足完整性和不可伪造性。

在公钥生成器上，将公钥添加到公钥列表中。在收到通讯信号后，就会有条件地选择该节点作为委员会成员。环境使委员会有选择地开放。

子协议在节点是BFT委员会成员时的工作方式: 分叉一个BFT虚拟节点。这里的BFT虚拟节点由BFTpk表示，然后开始接收TXs(交易)。如果停止信号由至少三分之一的初始委员会特定公钥签名，则检查日志完成结束。在此期间，将进行连续的“直到完成”检查，一旦每个步骤都完成了gossip，所有停止日志条目将被删除。

当节点不是BFT成员时，子协议的工作原理如下:在接收交易时，消息被添加到历史记录中，并由三分之一的初始委员会特定公钥签名

签名算法用前缀0标记内部BFT实例的每个消息，用前缀1标记外部DailyBFT的每个消息，以避免命名空间冲突。

3.2.3内存池协议.

内存池子协议：使用0初始化交易列表TXs，并使用并集跟踪传入的交易。在接收到提议调用时，它将交易添加到日志中，并使用gossip协议进行通信。它还支持查询方法来返回具体的交易。通过在一个集合中跟踪交易，它清除了已经确认的交易。

3.2.4 混合共识协议.

带有隐式消息路由的新生成的节点，带有发送和接收的消息副本的历史记录。这与以下组件相互作用:内存池、慢链、预处理器、链下共识和链验证。

3.3. 不同的选举周期和混合选举委员会

在[24]中，BFT委员会成员在一段特定的时间后进行换届(以慢链作为逻辑时钟)。新的委员会是由慢链内部产生最新csize区块的矿工组成的。在我们的共识设计中，我们希望利用这样一种直觉:如果委员会表现良好，我们就不必强迫他们换届，因此在某些情况下可以避免委

员会换届的开销。另一方面，如果前一届委员会保持良好的记录，这将增加新节点当选为委员会成员的难度。因此，我们仍然保持在固定时间点强制切换委员会的设计，但频率要低得多(例如，每K天进行一次委员会换届)。另一方面，我们结合了来自Thunderella [26]的经过认证的投诉的观点，其中慢链可以作为BFT委员会成员不当行为的证据。也就是说，每当委员会的不当行为从慢链中被发现时，第二天的起始点(不一定是第k天)就会触发委员会强制换届。

此外，我们将替换委员会的选举标准。在混合共识中，委员会成员是从最近的慢链区块的矿工中挑选出来的。我们决定根据股份制和随机性的混合标准来选举委员会成员。更具体地说,我们允许完整的节点提议对交易进行特殊的持股和分配，以暂时性冻结它们的token资产。每当一次委员会换届产生,根据他们冻结了的股份,我们将选出θ·csize个账户,θ∈[0,1]是一个手动参数。和后面[16]的设计,基于VRF的结果[20],我们选择其余(1−θ)·csize个节点，其中种子参数是由先前的委员会选举中使用的种子以及从最近csize块提议的随机性共同决定的。不同于Algorand[16]的选举，这里我们不计算这部分选举的权重。

注意，由随机函数选择的节点不在线的可能性很高。出于这个原因,考虑到评估的在线概率ron,因为我们不想离线的节点得到拜占庭委员会名额,我们需要确保ron·θ< f/csize。通常我们赋值θ= f/(2*ron*csize)。另一个可能的设计方案是，我们可以通过设计来允许离线节点[25]。

注意，一个拥有压倒性计算资源的一方可能会操纵VRF的输入，但这已经超出了我们的安全性假设

3.4 具体应用设计

在一致性、活跃性和安全性不受影响的条件下，我们的共识机制设计应意识到应用程序特定的需求场景和它们的定制。

3.4.1 物理时间限制

传统的共识机制设计默认允许矿工、委员会成员或leader在一个小的时间窗口内重新排序交易。这对于一些去中心化应用提出了一个问题，例如商业交易应用，公平的交易是按照时间顺序有序进行的，这个排序需要很好的保护起来，否则恶意的（或甚至是正常的理性的）参与者将有动机重新排序交易，甚至插入自己的交易，以获得额外的利润。这种利益驱使将在高吞吐量下放大。

更糟糕的是，这种恶意的重新排序交易是不可能被区分出来的，因为正常的网络延迟也会导致重新排序交易，这种延迟只能被接收者自己观察到，因此它有关于网络延迟的最终数字证据。

为了支持去中心化的数字化交易，我们尝试通过增加一个叫做粘性时间戳的限制来减少这些问题。更具体地说,用一个启发式参数TΔ提出交易时,我们要求客户端把一个物理时间戳Tp放入交易的元数据中,这个物理时间戳连同其他部分的交易被签名。稍后，当BFT委员会中的成员验证交易时，它将执行如下的额外检查，如算法1所示。

在实现BFT中的日志阶段，leader将根据其物理时间戳对交易进行批处理排序，并使用顺序编号断开连接(虽然可能性不大)。实际上，这一步是不必要的，因为我们可以在以后的评估和验证中执行命令。为了简单起见，我们把它放在这里。

这组修改给了我们几个额外的属性:

(1)对于任意节点Ni的交易顺序根据其物理时间戳在内部进行保存。因此，这些交易的顺序是严格执行的。这将消除涉及对来自同一节点的两个交易的恶意重新排序的可能性。

(2) 由BFT委员会输出的一批交易中的顺序严格按照时间戳进行排序。

(3)由于时间窗口限制，节点无法操作伪物理时间戳。

这种改进的一个明显缺点是当参数TΔ在不同的网络延迟中不合适时，由于终止了一部分交易而降低吞吐量。另一个缺点是，BFT委员会成员仍被允许谎报他们的当地时间并拒绝某些交易。然而，委员会成员可以以任何方式拒绝某些交易。但是，诚实的节点可能会因为它们不同步的时间而拒绝未知的交易。这个问题可以通过增加对BFT委员会的职责的限制来减少。稍后我们将看到，要进入委员会，节点应该提供同步时间戳的证据。

3.5计算和数据分片以及投机性的交易执行

在本小节中，我们将介绍我们的分片方案。

对原始混合共识的一个重要修改是我们为它添加了计算和数据分片的支持。更重要的是，我们设计了一个基于分片的投机性交易处理系统。这个思路是清晰的。在混合共识中，DailyBFT委员会被索引为一个决定序列DailyBFT[1…R]。我们允许同时存在多个DailyBFT委员会序列。准确地说，我们用分片St.表示第t个DailyBFT委员会序列，为了简单起见，我们将碎片的数量固定为c。每个DailyBFT都是一个普通的分片。除了C普通分片，我们还有一个由csize节点组成的主分片Sp。主分片的任务是最终确定正常分片输出的顺序，并在分布式交易处理系统中实现协调。而正常的分片，不是直接连接到混合共识的部分，而是提交日志到原始分片，而原始分片反过来又与混合共识连接。

我们不允许任何两个分片(正常的或原始的)共享公共节点，这可以在委员会选举程序中执行。多个分片的选举类似于第3.3节所述的选举流程。

我们将状态数据(按帐户范围)均匀地划分为 C个分片。这将确保对相应分片的每个查询都将返回一致的状态。由于我们将为每个数据单元包含元数据，所以我们将数据分割为数据分区的单元，并为每个数据分区分配一个地址。我们有从数据位置到数据分区地址的映射。为了简单起见，从现在开始，我们只讨论数据分区的级别。每个数据分区DS[addr]都有rts、wts、reader、writer的元数据。

我们假设分区原则是公开的，给定地址addr，我们可以通过调用函数host(addr)获得它的主机分片。

请注意，如果我们将每个普通的分片(当对手的数量不是很大时)作为一个分布式处理单元，我们可以将逻辑时间戳[31]的设计集成到分布式交易处理系统[19]中，它将授权处理交易。这里我们使用了简化版的MaaT，在这里我们不做其他交易的时间戳的自动调整。

对于正常的分片，它的作用与DailyBFT委员会中描述的完全一样，除了以下更改以使它与并行的投机执行兼容之外。

对于主分片，它收集所有正常分片的输出。请注意，交易的数据依赖项可以很容易地通过它们的元数据推断出来。事实是，如果一个交易访问多个远程分片，它将在涉及的所有分片中留下痕迹。当一个普通的分片提交日志到主分片时，它也将写入慢链中。

当主分片接收(或从慢链获取)来自某个分片的一批交易时，它将检查这些交易是否从该批交易中的所有分片中收到。如果在一定的超时之后，它没有从一个特定的批处理接收交易，这意味着该批处理失败。在这种情况下，整个委员会的换届将在第二轮开始时触发。在收到所有分片的日志后,主分片根据他们提交的时间戳对交易做排序(如果一些交易比批处理编号更早,它将被视为排序的最关键因素,然而,如果其物理时间戳违背了来自许多分片的时间戳,我们决定这批无效以及该批处理内的所有交易流产)。排序之后，就物理时间戳而言，主分片筛选所有交易并保持最长的非递减序列。将日志输出到混合共识部分，作为此轮的日志。

对于数据分片仍然有许多优化空间。有一个问题是这个设计中的确认时间不是即时的。

算法2：分片和投机交易处理

```
On BecomeShard:
  Initialize all the state data sectors: lastReaderTS = −1, lastWriterTS = −1, readers = [], writers = []
With transaction TX on shard  :
On Initialization:
  TX.lowerBound = 0;
  TX.upperBound = +∞;
  TX.state = RUNNING;
  TX.before = [];
  TX.after = [];
  TX.ID = rand;
On Read Address(addr):
if host(addr) ==  then
    Send readRemote(addr) to itself;
else
    Broadcast readRemote(addr, TX.id) to host(addr);
    Async wait for 2f + 1 valid signed replies within timeout  ;
    Abort TX when the timeout ticks;
Let val, wts, IDs be the majority reply;
TX.before.append(IDs);
TX.lowerBound = max(TX.lowerBound, wts);
return val;
On Write Address(addr):
if host(addr) ==  then
    Send writeRemote(addr) to itself;
else
    Broadcast writeRemote(addr, TX.id) to host(addr);
    Async wait for 2f + 1 valid signed replies within timeout  ;
    Abort TX when the timeout ticks.
Let rts, IDs be the majority reply;
TX.after.append(IDs) TX.lowerBound = max(TX.lowerBound, rts);
```

```
return;
On Finish Execution: for every TX′in TX.before do
  TX.lowerBound = max(TX.lowerBound, TX’.upperBound);
for every TX′ in TX .after do
  TX.upperBound = min(TX.upperBound, TX’.lowerBound);
if TX.lowerBound ¿ TX.upperBound then
  Abort TX;
Broadcast Precommit(TX.ID,⌊ ⌋) to all the previous remote shards which TX has accessed;
// If  TX.upperBound = ∞, we can set an arbitrary number larger than TX.lowerBound.
On receive readRemote(addr, ID):
if host(addr) ==  then
  DS[addr].readers.append(ID);
  return DS[addr].value, DS[addr].wts, DS[addr].writers;
else
  Ignore
On receive writeRemote(addr, ID):
if host(addr) ==  then
  DS[addr].writers.append(ID);
  Write to a local copy;
  return DS[addr].rts, DS[addr].readers;
else
  Ignore
On receive Precommit(ID, cts)
Look up TX by ID;
if Found and cts not in [TX.lowerBound, TX.upperBound] then
  Broadcast Abort(ID) to the sender’s shard.;
TX.lowerBound = TX.upperBound = cts;
For every data sector DS [addr ] TX reads, set DS [addr ].rts = max (DS [addr ].rts , cts );
For every data sector DS [addr ] TX writes, set DS [addr ].wts = max (DS [addr ].wts , cts );
Broadcast Commit(ID, batchCounter)to the sender’s shard.;
    // batchCounter is a number which increases by 1 whenever the shard submit a batch of log to the primary shard.
On receive 2f + 1 Commit(ID, batchCounter) from each remote shards which TX has accessed:
TX.lowerBound = TX.upperBound = cts;
For every data sector DS [addr ] TX reads, set DS [addr ].rts = max (DS [addr ].rts , cts );
For every data sector DS [addr ] TX writes, set DS [addr ].wts = max (DS [addr ].wts , cts );
Mark TX committed;
Let TX .metadata = [ShardID , batchCounter ];
On output log
Sort TX’s based on their cts . Break ties by physical timestamp.

On receive Precommit(ID, cts)
Look up TX by ID;
if Found and cts not in [TX.lowerBound, TX.upperBound] then
  Broadcast Abort(ID) to the sender’s shard.;
TX.lowerBound = TX.upperBound = cts;
For every data sector DS [addr ] TX reads, set DS [addr ].rts = max (DS [addr ].rts , cts );
For every data sector DS [addr ] TX writes, set DS [addr ].wts = max (DS [addr ].wts , cts );
Broadcast Commit(ID, batchCounter)to the sender’s shard.;
    // batchCounter is a number which increases by 1 whenever the shard submit a batch of log to the primary shard.
On receive 2f + 1 Commit(ID, batchCounter) from each remote shards which TX has accessed:
TX.lowerBound = TX.upperBound = cts;
For every data sector DS [addr ] TX reads, set DS [addr ].rts = max (DS [addr ].rts , cts );
For every data sector DS [addr ] TX writes, set DS [addr ].wts = max (DS [addr ].wts , cts );
Mark TX committed;
Let TX .metadata = [ShardID , batchCounter ];
```

On output log
Sort TX's based on their cts . Break ties by physical timestamp.

4 运行在虚拟机中的智能合约

4.1 设计原理阐述

在拥有Ethereum虚拟机(EVM)[30]的所有原因中，目标之一就是在POW共识模型中使用交易费来计量使用量。由于我们是一个混合共识模型，我们将更深入地探索这个领域。我们会考虑一下混合云生态系统的可能性。

人们对以太坊黄皮书[30]遇到的一个基本问题是里面的数学符号。因此，我们希望能遵循像KEVM 黄皮书[17]的做法来列出EVM和TVM(在4.2中描述)规范。将来，我们希望通过初链的github帐户(https://github.com/truechain)来维护我们自己的规范。

4.1.1 如果将虚拟机替换成容器会怎样

区块链架构中最接近这个概念的是Hyperledger的Fabric框架[9]。如果尝试着将fabric的有准入许可性质转化为无准入许可，最主要的挑战之一将是解决链码问题。这意味着，虽然可以将链码或智能合约保存在一个容器中，但这对公链来说不是一个可扩展的模型。拥有这样的模型意味着必须在单个节点上运行数千个容器，对应运行几千个智能合约(因为每个节点都维护一个副本)。

社区已经有人尝试限制运行在一个节点上面的最多容器数。正如Kubernetes容器编排平台[5]和Red Hat的Openshift容器平台3.9的集群限制[7]所示，目前的极限是每个节点100个pod，每个节点大约能运行250个容器。

即使使用最新的存储技术，例如brick多路复用技术[1]，容器的最大可能值(比如最大接触值)也不可能达到(至少现在)1000。在关于kubernetes问题的讨论中可以进一步研究这个问题，这个问题可以在Kubernete的github的issues页面[4]上看到关于负载限额决定了一个pod上可运行的最大容器数（MAX_CONTR）的更深入的讨论。希望扩展容器的人通常偏向于水平扩展而不是垂直扩展[2,6]，因为后者显著增加了设计决策的复杂度。对于集群规模化配置来说，没有一种通用的规则，因为它完全依赖于工作负载，在我们的例子中，由于它的去中心化，它更依赖于工作负载，对于向扩展工作负载迈出一步来说，并不是很有说服力。此时，它更像是一个创新性问题，而不是简单的技术规范研究问题。目前以太坊已经部署了超过1000个智能合约。因此这已经变成优化容器生态的设计问题了。

现在让我们扩展一下容器场景。考虑到上述危机，一种可能的解决方案是在无服务器架构中使用容器。但是考虑一个场景，当2000个智能合约是同时在线的，并且并发请求，即，每次调用链码(一个移动窗口)超过MAX CONTR的值时，我们将再次面临相同的问题。因此，只能建议在最大并发请求上增加节流率限制。通过设计，这严重限制了每秒钟的并发交易量。工程不应凌驾于可实现的目标之上。因此，我们选择坚持EVM设计，尽管为了我们的目的稍微做了一些修改。

4.2初链虚拟机(TVM)

在这个领域的一个典型案例是以太虚拟机(EVM)[30]，它试图遵循完全确定，并且尽可能简单，以使激励成为计算的一个简单步骤。它还支持各种特性，如内存的堆栈外存储、合约委托和中间调用值存储。

我们将为慢链复用EVM规范，但是在本黄皮书的下一个版本中，在仔细考虑类似于EVM的设计原理之后，我们为TVM添加了一个新的规范，通过使用kecak -256哈希算法和椭圆曲线加密技术(ECC)派生出基于堆栈的架构。

初链基础架构将整合EVM和类似EVM字节码执行引擎来运行智能合约。我们会使用一个虚拟机来处理POW共识，另外一个虚拟机处理PBFT共识，都集成在全节点中，因此它们可以处理按需调用。

TVM虚拟机基于DailyBFT公链技术，与以下组件交互:

复用一些tendermint的概念，比如ABCI(区块链应用编程接口)，其提供了一个抽象层，允许在一个进程中运行的共识引擎管理另一个进程的应用状态;

- 适合dailyBFT的另外一个共识引擎。
- 权限化的以太坊虚拟机。
- 保证交易达成的RPC网关。
- 待办事项-正式定义TVM的转换状态、智能合约部署策略以及将权限化的虚拟机部署到无权限链上的方法。
- 待办事项-定义参数在POW和完整节点(POW和PBFT)之间切换。

5 区块、状态和交易

待办事项-讨论区块、世界状态流、交易和执行模型的更改

6 激励设计

POW工作量证明协议有一个已被证实的事实，以前所未有的速度吸引计算资源。虽然比特币和以太坊等现有的基于PoW共识网络本身就很成功，但它们所吸引的计算资源不过是非常强大的哈希计算器。它们运行起来需要耗费大量的电力，而且没有产生任何有用的东西。

在本节中，我们将介绍奖励基础设施的概念，以便平衡BFT委员会成员和非委员会成员全节点的工作量。我们基于PoW发明了一种新的激励设计，在这里，参与的资源可以被引导到做有用的事情，比如每秒处理的交易(这里称为“TPS”)，并提供链路数据存储。

以太坊gas价格是由一个不可能套利的现货市场决定的，类似于[28]研究的电力现货市场。我们认为这个市场是不完整的，因此资产定价的基本定理不适用[14]。因此，潜在的gas价格将遵循“喷射噪声”过程，即众所周知的高波动性。我们引入了一个“gas市场”，在这里gas将作为期货进行交易，这个市场是在极小的极限内完成的。与以太坊相比,这预计将显著降低gas价格波动。

下面的小节将详细讨论激励设计的每个部分。

6.1 水果连作为慢链

目前基于PoW共识面临的最大挑战是效率问题和可扩展性。执行效率低下以及确认时间比较缓慢使它不适合开发复杂的应用程序，并且过度的能源消耗使其并不太符合能源节约型社会的准则。我们现在提出的协议是使用PBFT和POW的混合机制作为核心共识。与以太坊不同的是，交易和智能合约由网络上的每个节点执行，轮值BFT委员会进行大量处理的能力将会有所提升，而PoW（慢链）只用于选择委员会成员。在极限情况下，我们将委员会轮转频率到一个块和cSize = 1，我们恢复传统的POW共识。

BFT委员会必须有2/3的诚实成员[13]。因此，对于一些 $\varepsilon > 0$的情况，则需要链的质量$Q > 2/3 + \varepsilon$。单纯的使用Nakoto链作为慢链，很容易就能明显的感受到自私挖矿攻击策略的影响。如果一个自私的矿工控制了超过25%的区块链中的算力，他可以控制33%以上区块的产生22. 。根据[24]中的流程描述的，被选为BFT委员会的概率等于一个区块的产生。因此，自私的矿工的控制权值很可能超过BFT委员会的1/3，因此BFT协议受到影响。

最坏的情况可能是在[27]中所示的策略。如果一个自私的矿工控制了以太坊类型的区块链40%的算力，她可以通过优化自私的挖矿来控制70%的区块产生。根据[24]委员会选举流程，她将拥有超过70%的BFT委员会的控制权。拜占庭委员会不仅做出了妥协，而且这个自私的矿工还将拥有独裁统治的权力，宣布任何诚实的委员会成员在她的意愿下都是“不诚实的”。

我们选择水果链[23]作为初链混合共识的基础慢链。正如[23]中的公平定理所指出的那样，水果链更能抵抗自私的矿工。然而，如果攻击者直接控制区块链中33%的算力，BFT委员会仍然很脆弱。因此，在[24]和[23]中，我们将做出进一步的细分，用来缓解这些问题因此。我们还注意到，考虑到当前的矿池市场份额，在区块链中，一个团体获得相当大比例的算力并不是不可想象的。

我们需要在两种极端中找到平衡，

- 通过VRF[20]随机选择BFT的成员，这对女巫攻击（Sybil attack）是脆弱的。
- 任何选择过程中，被选择的概率与算力成正比。拜占庭委员会很容易受到拥有大量算力的的矿池的攻击。

我们提出的解决方案如下。当一个诚实的BFT节点达到λ链的长度,将发布链中的每个果实链的唯一矿工ID作为候选人(或者,每个矿工ID开采超过ν水果)。通过VRF的应用，从候选人中随机选出新的BFT委员会。

显然，Sybil攻击在这个方案下是不可能的，因为你需要最低等级的PoW才能成为候选人。对于一个庞大的矿池来说，要在拜占庭委员会达成妥协也并非易事。一个水果比一个区块更容易挖掘。

6.2 挖矿流程

接着[23]中的变量定义（第14-16页），fruitchain由区块链组成，每个区块都包含自己的一组水果。BFT执行的交易最初是打包成一份记录，作为果实挖矿。 下一个块被挖掘时，最新的参数R更新的果实将会打包到一个块中。

矿工只能运行一个挖矿算法随机产生散列值h。当[h]-ϰ<Dpf时开采果实，并且当[h]ϰ<Dp时开采块，其中Dpf和Dp分别是果实和块的挖矿难度参数。元组（R，Dp，Dpf）用来决定挖矿过程。

为了阻止ASIC矿机部署，我们会使最新参数ϰ=ϰ（t）与时间有关。VRF每3个月生成一次，生成广播产生一个新的ϰ（t）（落入有效范围内）。 这个过程的详细信息将包含在黄皮书的新版本中。

更具体地说，挖矿算法如下。 一个fruit是一个元组f =（h-1;h';η; digest; m; h），而块是元组b =（（h-1;h';η; digest; m; h），F ）下面是每个参数的意思：

- h-1指向前一个块的参考，仅用于果实的验证。
- h' 指向包含水果的块，仅用于块验证。
- η 是随机数。
- digest 是一个碰撞抵抗散列函数，用于检查果实的有效性。
- m 是果实中的记录。
- h = H（h-1;h';η，d（F）; m）是块/果实的散列值。
- F 是[22]中定义的有效果实集。

区块链= {链[0]，链[1]，...，链[1]}是由索引i排序的单个区块的集合，其中chain [i] .h-1 = chain [i-1] .h，l是链的长度。 我们称之为近期的'w.r.t. 链如果f∈{chain [1 -R + 1].F∪...∪chain [1] .F}。 在我们的实现中，我们选择R = 17。
算法四：区块链过程算法

```
Initialize
 chain = chain[0]
 chain[0] = (0; 0; 0; 0;⊥; H(0; 0; 0; 0;⊥), ∅)
 F = ∅
 if heard fruit f' then
    if f' is the unique fruit corresponding to message m'then
       F = F ∪ f'
   if f'.h−1 < f.h−1 for all other fruits f such that f.m = m'. then
       F = F ∪ f'
    if heard blockchain chain? and |chain'F| > |chain.F| then
       chain = chain'
       where |chain.F| is the total number of fruit contained in chain.
```

```
foreach time step (1 sec) do
  Heard signed message m, broadcasted by PBFT. Let
  l = |chain| − 1, so chain = (chain[0], ..., chain[l]).
  F' = {f ∈ F : f recent w.r.t. chain, f ?∈ chain}
  h' = chain[pos].h−1 where pos = max(1, l − ϰ).
  h−1 = chain[l − 1].h.
  while mined = FALSE do
    Randomly pick η ∈ {0, 1}ϰ
    Compute h = H(h−1; h'; η; d(F');m)
    if [h]−ϰ: < Dpf then
      f = (h−1; h'; η; d(F');m, h)
      F = F ∪ f
      boardcast fruit f
      mined = FRUIT
      if [h]:ϰ < Dp then
        chain[l] = ((h−1; h'; η; d(F')m, h), F')
        broadcast blockchain chain
        mined = BLOCK
```

我们暂时选择Dp和Dpf，预计开采一个水果和区块的时间分别为1秒和10分钟。我们暂时选择Dp和Dpf，预计开采一个水果的时间分别为1秒和10分钟。

我们在挖矿过程中得出以下结论：

- 水果比区块更容易挖到，因此矿工们没有激励机制来参与形成矿池。这使得PoW成为一个更公平的过程。
- 由于水果挖取难度较低，很可能两个水果同时挖到。 确定哪个有效的一种方法是通过选择具有较低hash值的那个水果。

直到他们写在一个区块后，水果才会稳定。因此，挖矿奖励将支付给区块矿工，然后矿工会将奖励分配给区块内的水果矿工。

水果链方案的一个优点是水果可以以任何顺序开采，这可以使挖矿高度平行进行。与分片结合时，这将特别有用。

6.3 存储挖矿

初链的目标是在每个分片上达到10,000 TPS，并且分片的数量被设计成相对于交易量的需求线性增长(大致与节点数量相关)。在10年的时间里，比特币网络在10年时间里产生了大约170 gb的交易历史数据。从10到10000，相同的数据量预计每3天生成一次。在100个切分，或100万TPS，我们可以期望它每45分钟产生一次。因此，在一个高TPS的公共链中，让每个节点存储像比特币这样的整个交易历史就不再可行了。

已经提出了许多解决方案，例如每隔几小时/天检查一次，其中每个节点只需要存储来自前n个检查点的交易历史记录。但是我们应该把交易历史的其余部分存储在哪里呢，因为没有人有动机去存储它。

我们的解决方案是在统一的激励基础设施中无缝地将交易处理与IPFS的存储能力合并。这将提供存储交易历史的解决方案，并允许在初链体系结构上运行大量复杂的应用程序。初链上的数据存储可以分为三个层次，

层次1：存储在每个PoW节点上，比如比特币和以太坊。 这是最永久的存储方式，也是效率最低的方式。不建议存储短文本以外的任何内容，例如关键信息的hash值。用户向PoW矿工支付gas费。

层次2：将会有一个类似ipfs的文件系统，其中数据的有限副本分布到整个链中的存储节点。这是为存储主网络的大量历史交易和大量数据的去中心化应用程序而设计的。用户将向矿商支付存储和检索的费用。

层次3：本地存储。该文件将只存储在您自己的本地节点。这对于点对点通信很有用，例如聊天程序，用户更关注没有人窃听。这是免费的。

快链协议的一个优点是水果挖矿有一个相对灵活的结构。ＩＰＦＳ的两个中心支柱是时空（PoSt）和复制（PoRep）共识的证明。 可以对水果链协议进行修改，使其在存储过程中可以挖矿。然而，非PoW矿工将拥有自己的报酬结构，独立于PoW水果矿工的报酬结构。

6.4 gas价格和分片

gas价格在期货市场上交易，期货合约以智能合同的形式出现。具体来说，合同将按以下方式执行。

甲方同意支付给乙方x个 True ，乙方承诺在T0 - T1期间执行甲方的智能合同，运行费用为1个gas。
乙方将向执行甲方智能合约的委员会C相应的资金池提供x个True。这被称为gas 池。
委员会C中的所有成员将平均获得gas池中的份额,并返回在池中平均每gas的消耗μ。
如果B支付的小于μ,她必须通过支付超过μ的第三方来弥补差额。如果B支付的超过μ,她将从第三方收到差额。

在这种设计机制下，当流动性提供者正确地预测网络压力，从而在极小的限度内创建一个完整的市场时，他们就会得到奖励。gas池中的平均机制吸收了价格波动，使价格本身成为网络压力的良好指标。

我们确保gas价格在预定区间内交易的意图。因此，如果动态平均价格维持在一定的阈值以上，一个新的PBFT委员会将通过量子观测过程产生。另一方面，如果动态平均价格维持在某一阈值以下，则现有的PBFT委员会在完成其任期后将不会被授予委员会资格。

挖矿奖励的比例分配如下:设置n为PBFT委员会在一个特定的场景下运行的节点数量,同时设置α> 1。挖矿奖励的比例的分配，其中PBFT节点= n/(α+ n),POW节点=α/(α+ n)。这种分配机制的目的为了在初链发展的后期，新的节点被激励为区块链总的TPS做出贡献，从而确保链的可扩展性。当挖矿奖励被对半分时，参数α等于PBFT委员会成员的数量。

根据网络带宽、CPU为标准对所有分片一视同仁将导致倾斜性结果，如不一致的TPS，或者更严重的是有时超过了超时限制，因为交易的顺序是由主片决定的。为了处理这个问题，我

们提议的奖励基础，与网络计算领域的伯克利开放网络一起工作。前面在这领域做过类似尝试的是Gridcoin [3] 和Golem网络 [29]。

Gridcoin的分布式处理模型基于类似网络计算的伯克利开放网络（BOINC）[8]的预审框架，它是一个开源的分布式志愿计算网络，广泛应用于cernVM[10]，由LHC@Home[17]项目治理。这样的框架用来处理长期非均匀财富分配。另一方面，Golem是一个非常好的采用稳健激励模型的项目，可以拿来作为奖励机制的借鉴。但值得注意的是，区块链技术驱动的基于奖励模型的志愿计算网络，如果设计不当，很容易陷入利息膨胀的陷进。所以说，在早期投资者因为初学者运气的因素获得的收益和后来者的获益差距，随着时间的推移会越来越大。

取决于交易的类型和针对一些智能合约是否需要分布式存储，我们采用BOINC和IPFS/Swarm的混合架构，包括EVM和TVm。这样可以采用Linux容器进行资源隔离。我们希望在黄皮书的下一个版本里展开这一节的讨论。

7 未来方向

即使对最初的混合共识机制采取优化，我们认为在本文所提出的基础上还有更多的优化空间，如:

- 改进所有节点的时间戳同步，而不需要中心化的NTP服务器。
- 奖励基础设施里的详细激励技术，使重度投资基础设施的投资人不会遭遇“被忽视”，“亏本”的问题。
- 支持副本创建的分片技术，尽量减少被BFT会员会拒绝的交易集。
- 添加零知识证明以增强隐私。
- EVM、TVM和Linux容器技术的混合基础设施。
- 改进虚拟机规范中的二进制数据编码方法，交易签名，收费模型等章节。

8 结论

我们正式定义了混合共识协议和其实现方法，在本草案里，我们介绍了下一版将引入的多种新的理念。我们建议大家在部署POW全节点时采用抗ASIC的硬件方案，关于硬件方案的更多细节将尽快给出。

在许可的基于POW的网络中的少数节点上运行的许可BFT链。

BFT委员会是一个轮值的委员会，可以及时防止腐败。

BFT委员会负责交易验证，而POW节点只负责根据我们得出和重新定义的一些规则选举委员会成员。

我们推测，新许可的虚拟机可能受EVM的启发，但具有不同的块状态和交易执行流程。

在POW链中没有权限的EVM与这个新的许可VM（我们称之为Truechain Virtual Machine - TVM）共存。

TVM将是验证与共识有关的交易的传统方法，而传统的EVM需要重新进行工作才能真正达成共识致，但对于BFT选举来说使用变日长是一个难题。

激励模式需要重新开展工作，使其基于TVM，仍然奖励POW链中的矿工。

我们最终会支持BFT委员会节点的分片，以提高可扩展性。

考虑到节点配置不一致性（节点池中不同的CPU/内存/网络带宽）的奖励机制最终将成为共识的一部分，从而加速交易。

因此，智能合约执行将仅在TVM（BFT节点）中发生。

9 致谢

我们对付出的孜孜不倦的努力，推动整个分布式协议进展，涵盖设计理念，实现细节以及前文所述的各个方案等架构的以下人员，致以我们最真诚的谢意：

· Rafael Pass, Miguel Castro, Satoshi Nakamoto, Vitalik Buterin, Gavin Wood, Ethan Buchman, Andrew Miller et al。向他们在论坛提出改进建议，通过参与Reddit、邮件组、聊天群、白皮书和黄皮书撰写等多种不懈努力致谢。

· 对CNCF and Kubernetes 社区提出的混合云计算的灵感致谢

参考文献

[1] Container-native storage for the openshift masses. URL https://redhatstorage.redhat.com/2017/10/05/container-native-storage-for-the-openshift-masses/.

[2] Deploying 2048 openshift nodes on the cncf cluster. URL https://blog.openshift.com/deploying-2048-openshift-nodes-cncf-cluster/.

[3] Gridcoin whitepaper: The computation power of a blockchain driving science and data analysis. URL https://www.gridcoin.us/assets/img/whitepaper.pdf.

[4] Increase maximum pods per node: github/kubernetes/kubernetes#23349. URL https://github.com/kubernetes/kubernetes/issues/23349.

[5] Kubernetes: Building large clusters. URL https://kubernetes.io/docs/admin/cluster-large/.

[6] Kubernetes scaling and performance goals. URL https://github.com/kubernetes/community/blob/master/sig-scalability/goals.md.

[7] Redhatopenshiftcontainerplatform'sclusterlimits.URLhttps://access.redhat.com/documentation/en-us/openshift container platform/3.9/html/scaling and performance guide/.

[8] D. P. Anderson. Boinc: A system for public-resource computing and storage. URL https://boinc.berkeley.edu/grid paper 04.pdf.

[9] E. Androulaki, A. Barger, and V. e. a. Bortnikov. Hyperledger fabric: A distributed operating system for permissioned blockchains. URL https://arxiv.org/pdf/1801.10228v1.pdf, 2018.

[10] J. Blomer, L. Franco, A. Harutyunian, P. Mato, Y. Yao, C. Aguado Sanchez, and P. Buncic. Cernvm a virtual software appliance for lhc applications. URL http://iopscience.iop.org/article/10.1088/1742-6596/219/4/042003/pdf, 2017.

[11] V. Buterin. Ethereum white paper, 2014. URL https://github.com/ethereum/wiki/wiki/White-Paper.

[12] R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. In Foundations of Computer Science, 2001. Proceedings. 42nd IEEE Symposium on, pages 136–145. IEEE, 2001.

[13] M. Castro, B. Liskov, et al. Practical byzantine fault tolerance. In OSDI, volume 99, pages 173–186, 1999. [14] W. Delbaen, Freddy; Schachermayer. A general version of the fundamental theorem of asset pricing. Mathematische Annalen. 300 (1): 463520., 1994.

[15] E. G. Eyal, Ittay; Sirer. Majority is not enough: Bitcoin mining is vulnerable. In eprint arXiv: 1311.0243. [16] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich. Algorand: Scaling byzantine agreements for cryptocurrencies. In Proceedings of the 26th Symposium on Operating Systems Principles, pages 51–68. ACM, 2017.

[17] E. Hildenbrandt, M. Saxena, and X. e. a. Zhu. Kevm: A complete semantics of the ethereum virtual machine. URL https://www.ideals.illinois.edu/handle/2142/97207, 2017.

[18] D. e. a. Lombraa Gonzlez. Lhchome: a volunteer computing system for massive numerical simulations of beam dynamics and high energy physics events. URL http://inspirehep.net/record/1125350/.

[19] H.A.Mahmoud,V.Arora,F.Nawab,D.Agrawal,andA.ElAbbadi.Maat: Effectiveandscalablecoordinationofdistributedtransactionsinthecloud.Proceedings of the VLDB Endowment, 7(5):329–340, 2014.

[20] S. Micali, M. Rabin, and S. Vadhan. Verifiable random functions. In Foundations of Computer Science, 1999. 40th Annual Symposium on, pages 120–130. IEEE, 1999.

[21] S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system. URL http://bitcoin.org/bitcoin.pdf, 2008. [22] S. M. A. S. E. Nayak, Kartik; Kumar. Stubborn mining: Generalizing selfish mining and combining with an eclipse attack. In IACR Cryptology ePrint Archive.

[23] R. Pass and E. Shi. Fruitchains: A fair blockchain. In IACR Cryptology ePrint Archive, volume 2016:916, 2016.

[24] R. Pass and E. Shi. Hybrid consensus: Efficient consensus in the permissionless model. In LIPIcs-Leibniz International Proceedings in Informatics, volume 91. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2017.

[25] R. Pass and E. Shi. The sleepy model of consensus. In International Conference on the Theory and Application of Cryptology and Information Security, pages 380–409. Springer, 2017.

[26] R. Pass and E. Shi. Thunderella: blockchains with optimistic instant confirmation, 2017.

[27] F. Ritz and A. Zugenmaier. The impact of uncle rewards on selfish mining in ethereum. In http://ieee-sb2018.cs.ucl.ac.uk/presentations/5-Ritz.pdf, 2018.

[28] T. Schmidt. Modelling energy markets with extreme spikes. In: Sarychev A., Shiryaev A., Guerra M., Grossinho M..R. (eds) Mathematical Control Theory and Finance, pp 359-375. Springer, Berlin, Heidelberg, 2008.

[29] T. G. team. The golem project: The golem project. URL https://golem.network/doc/Golemwhitepaper.pdf, 2016.

[30] G. Wood. Ethereum: A secure decentralized generalized transaction ledger. URL https://ethereum.github.io/yellowpaper/paper.pdf, 2018.

[31] X. Yu, A. Pavlo, D. Sanchez, and S. Devadas. Tictoc: Time traveling optimistic concurrency control. In Proceedings of the 2016 International Conference on Management of Data, pages 1629–1642. ACM, 2016.

附录A.术语 TrueChainVirtualMachine(TVM): 与处理激励和轮值委员会选择的EVM不同，TVM基于类似的设计原则，但基于PBFT的混合共识进行实际共识和投票。