一. 引言

自 2015 年创世以来，以太坊区块链已历经五个寒暑。五年的时间不仅把作为一种理念的以太坊协议[1]变成现实、使这套协议变得更加成熟、更加具体，也使这样一套设计的特性和权衡关系暴露出来。这些权衡关系，作为设计上的挑战，自然也吸引了并持续吸引着无数聪明才智，尝试提升以太坊的可用性；其中汇聚了最多努力和想象力的，当属为提高 “可扩展性” 而提出的一系列方案。

本文想指出的是，就像以太坊范式面临着设计上的取舍，所有这些可扩展性方案也面临着取舍；**而评价这些取舍的值得与不值得，需要我们回到以太坊本身**，回到以太坊节点和以太坊用户的真实问题和真实需要。而 “状态” 视角，作为理解以太坊本身的视角，正好能帮助我们廓清这些方案的设计，并揭示我们的所得和所失。

本文将从解释 “状态” 的含义开始，揭示以太坊的终极之矛和阿喀琉斯之踵，然后探讨各种改进方向。“富状态性” 是以太坊智能合约 “可组合性” 的来源，但也是以太坊网络最大的弱点。由此，我们可以见出现在常被人提起的哪些方案会影响 “可组合性”，哪些 “可扩展性” 的意义更明显。

二. 以太坊作为一种范式

（一）富状态性与可组合性

什么是 “状态”？状态就是一个系统在某一个时刻的具体情形。以实现密码学货币的区块链为例，一个区块链在某一个时刻的状态就是该时刻，该区块链上所有地址的资产分布（A 地址有 10 个 币，B 地址有 100 个币，等等）（究其实，我们拥有的，只不过是一些状态，是我们的头脑和社会共识，将这些状态解读为 “资产”）。

在此视角下，每一套区块链协议都可以粗略地分成两个部分，**一部分是共识机制，另一部分是状态转换规则（有时也被称为 “共识规则”）**；前者定义了出块规则，它指明了所有参与该区块链的节点在什么时候需要在本地更新区块链数据库的状态（例如，作为一个以太坊区块链的节点，每当收到一个附带了符合难度要求的工作量证明的区块，该节点就更新本地的区块链数据库的最新状态）；而状态转换规则，则定义了什么交易是有效的（“一个账户不能花用超出自己余额的钱”），也定义了节点在处理交易时，应当如何更新状态（“这笔交易表示 A 账户转了 5 个币给 B 账户，那么 A 账户余额就减去 5，B 账户余额的加上 5”）。

对比特币来说，其共识机制是 “PoW + 中本聪共识”，而其状态转换规则基于 UTXO。对以太坊来说，其出块机制是 “PoW + Ghost 规则”，其状态转换规则基于账户。

那么，到底是什么东西让以太坊变成一个有突破性的创新呢？

我们时常听说，以太坊之所以特别，在于其 “引入了图灵完备的编程语言，支持可编程性” 等等。实际上这种说法并不准确[2]。因为允许使用更复杂的编程语言并不意味着什么，其实比特币也可以编程；允许比特币使用 solidity 编程语言并不能得到一个以太坊。以太坊真正特殊的地方在于 “富状态性（rich statefulness）”[3]：它允许一个合约调用另一个合约，并且，除了区块容量本身，不对这种调用的层级数量施加任何的限制。

B 合约可以根据 A 合约公开的代码来调用 A 合约、改变 A 合约的状态；调用 B 合约的 C 合约也可以间接地调用 A 合约、改变 A 合约的状态 …… 由此，一个状态，虽然保存在 A 合约里，但其控制逻辑，可以如此重重叠叠不断累加；**如果把状态理解为资产，这等于是让资产的使用权（使用条件）可以不断得到更严格、更复杂的控制；这意味着，从理论上来说，一个状态的更新逻辑能够无限接近于现实生活中的金融合约**（因为所有的金融合约都可以通过简单的逻辑叠加、组合出来）。这种允许累加任意多层的控制的属性，才是最关键的，而怎么编程这种控制，反而是次要的。

此外，以太坊还允许用户给区块链写入状态，使这些状态成为全局状态的一部分，并要求节点按照合约自定义的逻辑来更新状态。由此，一个合约能够把自身的状态向以太坊上的所有其它账户公开，前述的富状态性真正有了用武之地。

没错，现在我们已经像收集龙珠那样集齐了三种属性：

（1）链上计算范式：合约**可以**要求节点按自己定义的逻辑来执行计算；

（2）全局状态：合约的状态**可以**成为全局状态的一部分，向所有其他账户公开；

（3）富状态性：合约之间可以相互调用，且栈层数量没有限制，因此控制逻辑**可以**层层累加；

现在我们可以召唤出以太坊的最强之矛了：“可组合性”（货币乐高）！

链上计算方式使我们可以拥有各式各样的合约；全局状态让这些合约可以相互访问彼此的状态；富状态性让合约的组合方式可以无限多样。所以我们不仅可以拥有稳定币 DAI，还可以拥有借贷市场，还可以拥有彩票应用，还可以拥有把彩票的收益自动捐献的应用，还可以拥有在不同的借贷市场间自动再平衡储蓄比例的应用 ……

（二）状态数据爆炸问题

“可组合性”太美好了，以至于不像是真的，对吗？没错，上述三种属性的三位一体，实际上是一把双刃剑。

以太坊的状态转换过程可以抽象为：状态转换函数以旧的状态和事务（transaction）列表为输入，输出新的状态。这意味着，**以太坊的全验证节点必须在本地维护着以太坊区块链的最新状态，以便能执行状态函数并以结果来验证一个区块的有效性**（同时 也是与其他的以太坊节点达成共识）。

矛盾之处在于：对于合约及开发者来说，合约的状态作为以太坊全局状态的一部分，保存在以太坊的节点上，合约的状态更新由以太坊节点来计算（而且还是由发起计算请求的用户来付费的），这种 “无服务器” 的架构非常舒服；但是，这些状态只要一次付费就会永久保存在以太坊的全验证节点中，虽然每次更新都需要付费，但无法避免节点本地保存的状态数据会不断累积、膨胀。

状态数据的膨胀之所以是一个问题，在于它会为全验证节点带来越来越高的硬盘（随机）读写负担。状态数据不像区块数据，区块数据是静态的，持久化保存后并不需要频繁读写；但状态数据每多一个区块就要读写许多次；而随着状态数据量的增大，读写的负担也会越来越重。在过去几年中，我们时常听到有人说以太坊的全节点难以部署，一大原因即在于此。前一段时间 Infura 的免费以太坊节点服务崩溃[4]，导致许多依赖 Infura 的服务崩溃，算是给大家鸣响了一个警钟 —— 原来以太坊的节点维护如此不易，让大家宁愿选择信任他人。

这个问题也并不容易解决。在过去几年中，以太坊的多次硬分叉升级都提高了访问状态的操作码的 Gas 消耗量，正是为了以经济代价遏制合约创建新的状态。但这显然只是治标的办法，因为根本上逻辑并没有变，状态数据要持续存留在以太坊节点处，但创造状态数据的用户只需付一次费。也有人提出，为改变这一点，需要引入某种 “状态租金” 机制，要求保存了状态的合约不断支付租金，否则就终止该合约的可用性。但是，这种机制存在难以想象的复杂度，一方面，难以确定合理的收取租金的方法，另一方面，也难以确定合理的支付对象。所以状态租金机制的研究在 2019 年也停滞了[5]。还有项目（如 Nervos）尝试以持币量的多少来界定可用的状态空间大小，因此状态数据的大小将总是有上限的，这就避免了状态膨胀问题，但这也改变了原生资产的经济属性。

到目前为止，除了 “无状态性”，我没有看到令人满意的、从根本上解决这个问题的方法，而 “无状态性”，我们现实地说，也面临许多挑战。这个我们后文再说。

总而言之，全局状态、链上计算和富状态性，既使以太坊上的合约获得了可组合性，也使以太坊的网络有陷于中心化的危险；就像小说《指环王》中的魔戒，既能召唤强大的力量，也可能吞噬使用者自己。我担心以太坊还要承受这样的重负很久。

接下来，我们以 “状态” 视角，来理解诸种可扩展性方案的设计和权衡。

三. 以太坊的发展方向

在此章节中，我们会分析四种以太坊的发展方向：Layer-2 方案、分片、无状态性和 Rollup 方案。这个分类是完全不合理的，因为 rollup 是 Layer-2 方案的子集；而无状态性是分片的前置技术；甚至于，将它们并列也是不合理的，因为 Layer-2 方案几乎不需要改动以太坊的底层，而分片和无状态性则有这样的要求。这样做只是为了叙述和认识的方便。

（一）Layer-2 方案

Layer-2 方案背后的理念来源于一种简单但非常精准的直觉：以太坊之所以会面临吞吐量的瓶颈，是因为组成以太坊网络的节点的带宽、计算能力和维护状态数据的能力都是有限的，而且很难提高；单纯要求整个网络的节点在单位时间内处理更多的交易，节点的运行要求必然上升，这就损及了去中心化；但是，从使用的角度看，并不需要让所有的状态都放在以太坊上，也不需要让所有的状态计算都在以太坊上发生；我们可以把一个合约的中间状态（或者所有状态）都保存在别的地方，用户的交互（也即状态的更新）也不在以太坊区块链上发生；仅当用户认为有必要结算某个状态时，再将该状态发送到以太坊上 ，由以太坊来加以确认。

一句话：**如果我们没法让网络在单位时间内对更多的事务达成共识，那就提高单笔事务的内涵**。

经典的 Layer-2 方案 “状态通道” 最彻底地体现了这种思想：当参与一个通道的两个用户将资金锁入合约后，此后两人之间的交易都不会发到链上，他们彼此之间使用其他通讯工具来交流签名消息，并以此达成彼此对通道内状态的共识（因此无论他们之间收发多少消息、形成多少状态数据，都不会成为以太坊的负担）；直到两人认为不再需要交互了（或者有必要暂时结算一次了），就把共同认可的状态及两人的签名发送到以太坊上，以太坊这才更新该合约的状态，并根据此状态为两人结算资金。

如果你把 Layer-2 方案当成合约的一种设计模式，你会更清楚地看到 —— **Layer-2 方案选择了不去利用全局状态**。另一个合约并不能实时地知道某个 Layer-2 合约内部到底是什么状态（用户 A\B\C 都各有多少钱），因为这些状态并不在链上，也因此，一个 Layer-2 合约就无法与其它合约相组合了。

虽然如此，Layer-2 方案也换来了极有价值的东西：更快的交易速度（虽然被确认的中间状态没有主链状态这么安全）、更低的手续费、更小的主链节点负担。

但 Layer-2 方案为什么在过去几年中都没有结出果实呢？因为在 Rollup 方案出现以前，其他方案，包括状态通道和 Plasma，都没法证明锁入自己合约的资金与锁入带状态合约中的同样安全（既不会被冻结，也不会被盗走）。在状态通道方案中，如果你没有时刻监控区块链，你的交易对手可以通过向主链提交旧的状态，来 “盗” 走你的钱；在 Plasma 方案中，往往你需要依赖于运营者来为你提供自己状态的证明，因此它对运营者本身是很难设防的。

这跟以太坊主网带状态合约的使用体验完全不同。除去合约的代码风险，你存入合约的钱，如非有人发动 51% 攻击取消你的所得交易，否则不会被盗；除非有人一直通过 51% 攻击来审查你的交易，否则你必定可以将自己的资金取出。（在后文中，我们会看到 Rollup 方案是怎么解决这个问题的。）

（二）分片

另一种改进以太坊、提高以太坊吞吐量的直觉是：以太坊的吞吐量有限，源于每个全节点都必须处理所有链上交易；如果我们可以让每个节点只处理一部分交易，不同组的节点分别（并行化地）处理不同的交易，那么整个系统在单位时间内的吞吐量就等于这几组节点的单组处理量的总和；即，单个节点的负担并没有增加，但整个系统的吞吐量提高了（而且分组越细，吞吐量的乘数越大）。这就是所谓的 “分片”。

关于分片化架构，一个有趣的问题是其确切定义。在经典的、非分片化的区块链架构中，全节点必须（1）重复所有计算；（2）维护所有状态；（3）传播所有区块/交易。有人认为，只需打破一者，就算是分片化的了；但从更严格的视角看，必须打破所有三者，才能达到理论所推导的吞吐量提升效果[6]。取不同的定义，带来的效果也很不同。此处，我所采取的定义是，一个节点至少无需维护整个系统所有的状态。

但另一个更有趣的问题是，假使分片架构所要求的前置技术都能实现，这一架构能有多大的意义。

在以太坊基金会的研究团队所构想的分片架构中，整个系统的状态被分割成了几个部分，各部分是并行更新的，也就是说，一个分片并不能实时地了解另一个分片的内部状态。当位于分片 A 的合约 A' 尝试调用分片 B 上的合约 B' 时，并不能假设处理者拥有分片 B 的状态，也因此，处理的结果必须等待分片 B 完成状态更新后才能返回。**由此，跨分片的交易必须忍受事务处理中最难以忍受的代价：时延**。关键在于，此种时延是不能用货币代价来加以抽象的。因为分片 B 也并不知道分片 A 上发生了一笔需要调用自身的交易，因此只能等待一个可信的通信层为自己播报这条交易。

单位时间内能够创造的价值越高，时延越不可忍受；而这意味着，假设有某个 DeFi 应用集中的分片，这些应用根本不会通过跨分片的交易来调度其它分片上的处理能力，因为这根本没有意义，满足不了 DeFi 应用对时延的要求。同样，这也意味着，这个 DeFi 应用集聚的分片，其单分片处理能力就是其处理能力上限（与单条区块链没有区别）。在深山老林里造房子和修路，解决不了大都市里人们的生活空间问题。

（有人认为，这些 “闲置” 的分片吞吐量可以为更小众的应用所用。我认为，Layer-2 方案能达到同样的效果，而且对底层的改动更少，更安全。）

（三）无状态性（statelessness）

无状态性是唯一直面了状态数据膨胀问题的升级方向[7]。

在当前的以太坊协议中，交易自身并不携带自身所访问的状态的信息，正是因此，处理交易的节点才必须维护状态数据，作为执行状态转换函数的前置条件。也正因此，状态数据的膨胀才是一个问题（提高节点负担）。

而无状态性的关键，正是让 交易/区块 自身附带所访问状态的信息，因此，一个区块仅凭自身就是可验证的，无需处理交易的节点具有状态。

实际上，**无状态性是通过改变以太坊区块的结构，改变以太坊区块的验证方式**。无需维护状态数据，也就免去了读写硬盘的需要，区块验证的速度也可以更快。此外，各节点既可以完全不维护状态数据，也可以根据自己的需要，维护某些合约的状态数据。

但是，诚实地说，无状态性目前还面临许许多多的设计挑战。具体来说：（1）无状态性要求为 区块/交易 附加所访问状态的证明（witness），这部分数据的规模可能非常大，以太坊当前的区块数据大小大概是 20～40 KB，但 witness 数据的大小可能在 MB 级别（视所访问状态量的大小而定）；（2）只有维护了所有状态的节点才能组装出 witness，那么谁来为普通用户提供状态呢？（实际上，可以把这个视为以太坊运作假设的一个转变：原来所有的全节点都能参与挖矿；但无状态性实现之后就不是了）；（3）如何为交易的 gas 消耗量定价？尤其因为 witness 有时效性，并不能根据操作码来确定组装 witness 耗费了多少计算量。

正是因为这些困难，以太坊的全节点可能还必须在这种维护所有状态的模式中运行很久。但是，无状态性绝对是当前对以太坊协议的改进中，最激动人心的方向。因为它直面了以太坊的核心问题，并尝试釜底抽薪地解决这个问题。另外，对状态数据在以太坊协议中的使用的研究，也滋养着其他的研究方向，如同步方法[8]。

我有偏见地相信，以太坊的未来即使不是无状态性，也是得到无状态性启发的某种方案。

（四）Rollup 方案

Rollup 方案是一种 Layer-2 方案，其特殊之处在于，它会将每一次状态更新所用到的交易，都在以太坊区块链上发布出来。

与其他 Layer-2 方案一样，Rollup 方案也把状态存在链下，也不要求以太坊的节点来计算合约的新状态；但是，把会更改这个合约状态的交易，都作为数据发布出来，这意味着，**任意第三方，都可以凭借这些公开的数据和公开的规则，计算出该合约的状态**（尽管一个以太坊上的合约无法利用这些状态）。

前面说到，当 Lyaer-2 合约选择了把状态计算移到以太坊链外，并遮蔽了自己的合约状态，它就给用户引入了风险：用户并不知道，Layer-2 合约的运营者会不会把一个无效的状态发到链上交由区块链来敲定，如果运营者能够这样做，那就等于是能直接盗取用户的资金；此外，用户也不知道 Layer-2 合约的运营者会不是审查自己的交易，从而冻结自己的资金。

资金被盗问题有两种解决思路，一是保证每一次状态转换都是有效的，也即是每一次要更新合约的状态根时，都让以太坊来执行一次对计算完整性的验证程序，只有验证通过了，才允许合约更新状态根，这就是 zk-rollup 的思路；另一种思路是，要求人们在请求更新合约状态根时，都附带押金，如果所提交的状态根是无效的，检举此状态根的人可以获得原提交者的押金，这就是 optimistic-rollup。但后面这种思路有个前提条件：检举人必须有办法获得该次状态转换前的状态，否则无以生成该次转换的错误性证明。

资金被锁问题的解决思路只有一种：尽可能弱化 “运营者” 的概念，使得谁都能向以太坊提交交易来更新该 Layer-2 合约的状态。但这又回到了那个问题：如果提交者没有该合约的状态，怎么证明自己的状态访问的有效性，让合约放行呢？

终于，Rollup 方案用 “在每一次状态转换时都发表匹配的交易数据” 解决了这个问题。由此，Rollup 合约虽然没有把状态公开在链上，但是任何人 —— 也包括 rollup 合约的用户 —— 都能根据这些公开的交易包重建出一个 rollup 合约的内部状态。这意味着，假使设计得当，rollup 合约内的资金，与以太坊区块链上保存了状态的合约（也即是我们现在常用的合约），可以是同样安全的！

以 zk-rollup 为例，链上验证程序保证了一个 rollup 合约不可能更新一个错误的状态根，就像如果没有你的允许，Maker DAO 也不能没收你的 DAI；同时，假设它也公开了计算完整性的构造方法，则你随时可以在链上直接向合约发起状态转换，来取出自己的钱。这就跟普通的、带状态的合约完全一样了：如果没有 51% 攻击，就无法回滚 rollup 合约的状态；如果没有持续的 51% 攻击，就无法阻止你取回自己的钱。

Optimistic-rollup 依赖于一些密码经济学的假设，因此稍弱一些：除了 51% 攻击回滚状态以外，攻击者还可以通过持续一段时间的 51% 审查攻击来给合约注入错误状态根；或者可以赌一把，赌所有计算出了该合约最新状态的人都不会观察到错误（概率极小）。但 Optimistic-rollup 同样能提供很强的免托管性，你也能随时取回自己的钱。

换言之，**如果用户愿意把钱存进一个有状态的合约**（例如 compound、maker dao、uniswap），**那就没有理由不愿意存进 rollup 合约**（如果非要加限定词，那就是 zk-rollup 合约）。有了 zk-rollup，layer-2 方案已经能为用户提供以太坊区块链上合约所能提供的最大程度的资金安全了。显然，也只有做到这份上，才有可能获得大规模采用。

所有的 Layer-2 方案都可以视为一种趋向无状态性的折衷：Layer-2 方案本身是无状态的（对其它合约来说，Layer-2 合约的内部状态是不可访问的），其内部状态再复杂，都不会增加以太坊节点的负担；同时，更新 Layer-2 合约（的状态根）时，以太坊扮演的角色更多是验证，即验证状态根更新是有效的，而不是自己去计算出这个状态根。但是，到了 rollup 时代，Layer-2 方案才证明了自己可以和带状态的合约一样安全，其承诺这才有可能化为现实。

四. 结语

综上，在本文中，我解释了以太坊的 “可组合性” 的来源，以及各种受到热议的可扩展性方案是否牺牲了这种可组合性，又换来了什么。读者可能会注意到，我在推理和评价的时候，相当重视 “我们已经实现了什么/得到了什么” “人们的行为表明他们需要什么”。不错，我正是从这个角度出发，来解释 rollup 方案的魅力的（事实上，这可能是我写这篇文章的核心动力之一）。在我看来，这样一种思维倾向，能使我们的思考的起点更加可靠，免于臆测用户的需要，亦免于投入建设空中楼阁。

在历史的流转中，我看到人们在可选的东西中选了某些东西，这就使我不得不认为，这些东西是重要的；而如果某些技术，既没有增加人们可选的东西，又牺牲了人们实际上选择了的东西，那就没有理由对这些技术怀有信心。

（完）

注

注 1：以太坊白皮书，https://ethfans.org/posts/ethereum-whitepaper

注 2：Vitaik 的推特，https://twitter.com/vitalikbuterin/status/854271590804140033?lang=en

注 3：Vitalik：基础层和功能性的逃逸速度，https://ethfans.org/posts/vitalik-base-layers-and-funtionality-escape-velocity

注 4：如何看待 Infura 服务崩溃，https://ethfans.org/posts/38028

注 5：论状态租金和 Stateless Ethereum，https://ethfans.org/posts/on-the-state-rent-and-pivot-to-stateless-ethereum

注 6：基于委员会的分片区块链中的安全性和可扩展性，https://ethfans.org/posts/security-and-scalability-in-committee-based-blockchain-sharding

注 7：以太坊无状态客户端初探，https://ethfans.org/posts/data-from-the-ethereum-stateless-prototype

注 8：Beam Sync：同步以太坊节点的新方法，https://ethfans.org/posts/intro-to-beam-sync