arXiv:2301.04508v1 [cs.DC] 11 Jan 2023

分布式系统中机制设计与优化的部署优先方法

Martijn de Vos、Georgy Ishmaev、Johan Pouwelse、
Stefanie Roos 代尔夫特理工大学，荷兰

摘要——**受区块链技术日益普及的推动，近年来，人们再次对去中心化系统的设计、实现和评估产生了浓厚兴趣。这些系统大多旨在大规模部署于异构环境中，面向真实用户并应对不可预测的工作负载。然而，该领域的大多数研究都在受控环境中评估此类系统，而这些环境难以反映现实环境的复杂条件。本文认为，部署对于理解现实环境中的去中心化机制至关重要，也是构建更稳健、更可持续系统的推动因素。我们将这种方法与其他实验设置进行比较，阐述部署的优势，并展示我们的实验室如何采用**部署优先方法。**随后，我们概述如何利用点对点文件共享应用 Tribler，大规模部署和监控去中心化机制。我们通过介绍代币经济学领域的一项部署试验，展示该方法的应用。最后，我们总结了在多次采用该方法的部署试验中汲取的四点经验。**

索引词—**去中心化系统、研究方法、实验设置、系统故障。**

I. 引言

自该领域诞生以来，分布式系统的规模和复杂性大幅提升。我们生活在一个复杂的超大规模网络时代，这些网络的特征体现在参与节点的数量，以及高度异构性、灵活性、非平凡的社会依赖关系和涌现属性等方面 [3], [11], [25]。确保此类系统正常运行、部署、监控和维护，需要系统设计者深入了解其性能和正确性。鉴于此类系统的规模和不可预测性，获取工程方面的洞见为研究人员和开发人员带来了独特挑战。

去中心化区块链应用体现了复杂超大规模网络的特征 [2], [24]。与此同时，由于内置的金融机制，此类应用中的故障成本非常高 [29]。尽管测试网等现实实验设置旨在一定程度上应对这些挑战，但对这类系统进行适当的测试和评估仍然是一项艰巨任务 [5]。

分布式系统研究中的主流范式是自顶向下设计，侧重于通过在受控环境中进行实验，预测性能、故障和局限性。这些实验通常以仿真或模拟的形式开展，为研究人员的设计选择提供依据 [4]。故障实证研究能够为了解不同类型的分布式系统提供宝贵见解 [12], [13], [26]。由于存在复杂的相互依赖关系，检测导致系统无法按规范运行的事件——即故障检测——往往是分布式系统中的一项关键任务 [17]。

然而，在学术研究中，依据从已部署系统提取的工作负载开展实证实验并不常见 [4]。即使是在分布式系统研究等由实证驱动的领域（例如区块链应用 [18]），仍缺乏成熟且注重部署的研究方法。

我们通过介绍用于设计和评估去中心化系统的部署优先方法，弥补了这一空白。[1] 我们特别关注从部署后故障研究中获得的发现。我们的方法基于近二十年来开发 Tribler 软件的经验 [8], [22], [28], ；该软件作为基础设施，支持大规模部署和评估去中心化机制。我们证明，尽管部署优先研究方法需要投入大量时间，但由此获得的额外见解值得付出这些时间。对不同实验设置之间权衡的细致评估，也有助于设计研究方法。

总而言之，本文的贡献如下：
1) 我们比较了不同的实验设置，并强调部署作为构建和评估分布式系统时研究方法中的关键步骤所具有的价值（第二节）。2) 我们提出了自己的部署优先研究方法，用于大规模评估去中心化机制（第三节）。我们还介绍了如何使用 Tribler——我们的去中心化文件共享应用程序和研究平台——开展大规模部署试验。3) 我们阐述了如何将研究方法应用于去中心化网络中的代币经济学这一复杂用例，并从中获得独到见解（第三节 B）。4) 基于运用该方法的经验，我们总结了四条经验教训（第三节 C）。

II. 去中心化系统实验的现状

本文认为，部署实验对于构建稳健的去中心化机制至关重要。我们首先

[1]本文主要聚焦于去中心化系统。不过，其中的一些见解也可能适用于更广泛的分布式系统领域。

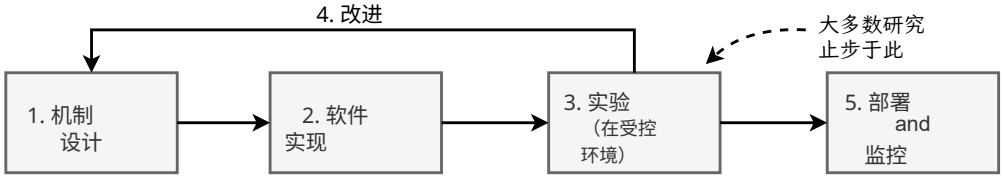


图 1. 设计、构建、评估和部署去中心化系统的标准研究方法。

介绍研究分布式系统的标准方法，然后概述将部署纳入研究方法的优点。

A. 标准研究方法

图 1 展示了设计、构建、评估和部署去中心化系统的标准研究方法。这种研究方法基于该领域文献中的报告 [4], [14], [15]、与其他研究人员的讨论，以及我们自身的经验。以下五个步骤描述了这种方法：

1) **机制设计**。 研究人员首先设计一种特定的机制。专注于理解理论模型的研究通常仅限于这一步骤 [4]。

2) **软件实现**。 随后，研究人员着手将设计好的机制实现为软件。假设最初的设计得到了妥善执行，实现阶段不应导致原始模型发生重大变化。因此，我们在图 1 中省略了这一反馈回路。

3) **实验**。 借助这一实现，研究人员在由分析人员控制的环境中开展实验，例如在本地计算机或计算集群上。这些实验通常旨在验证实现的正确性，并量化系统指标，例如可扩展性和容错性。

4) **根据实验结果完善设计**。根据上一步获得的实验结果，研究人员会更新机制设计和配套实现，并重新运行实验。例如，如果实验表明某项设计的可扩展性较低（例如系统可支持的参与者数量较少），研究人员可能需要找出机制中的瓶颈并加以解决。

5) **部署与监控**。实验完成后，研究人员可以在现实环境中部署算法。部署的软件很可能会受到持续监控，以便检测故障或异常。

大多数学术研究不会进一步通过部署来测试和评估其机制，而是在步骤 (4) 结束[4]。这并不令人意外，因为本地实验通常足以向科学界证明机制在权衡、正确性或性能方面的表现。因此，投入的时间和资源成本并不足以证明部署的必要性。[2] 然而，我们认为，使用实验设置来评估去中心化系统所涉及的权衡与局限性要更加复杂。

B. 部署的优势

我们首先比较用于评估去中心化系统的不同实验设置之间的权衡。根据文献调研，我们选择比较以下四种实验设置：

1) 仿真是在环境模型上测试应用程序模型；2) 模拟是在某个环境中运行的应用程序，其中部分内容经过建模；3) 测试网是部署在多台机器上的应用程序，这些机器由研究人员或志愿测试者运行；4) 现实世界部署是由最终用户在自己的机器上运行的应用程序。

我们在表 I 中比较了这些实验设置的八项不同属性，并在下文中进行简要讨论。

实验成本。 实验成本并不总是能够以可比较的尺度来表示。通常，实验成本可以用购买必要计算资源所需的货币成本来描述。在仿真和模拟中，这些成本都相对可控。然而，对于已部署系统，实验成本可能高得多，也可能低得多，具体取决于应用场景。对已部署系统进行需要更改系统参数或功能的实验，可能会导致故障，并造成用户流失或市场份额损失 [5]。不过，在某些情况下，对已部署系统进行实验的成本可能很低，例如志愿用户提供资源时。[3] 在测试网中，如果大部分资源由志愿用户提供，研究人员的实验成本可能较低。然而，为测试网吸引足够数量的用户也可能需要一定的前期投入和启动成本，以获得初步发展势头；区块链测试网的营销成本便是一个例子 [10]。

可扩展性。 实验的可扩展性与实验成本密切相关。在仿真中，扩大仿真规模的成本相对较低。在模拟中，可扩展性的上限通常受测试平台可用计算资源的限制。在

[2]与此相反，在区块链行业中，部署新颖的系统设计是常见做法 [18]。然而，与学术研究相比，实验评估的质量有所欠缺，因为行业白皮书往往会呈现有偏差且夸大的结果 [20]。[3]参见 https://github.com/ethereum/ropsten/blob/master/revival.md。

属性	仿真	模拟	测试网	现实世界部署
实验成本	中	中	低/高	低/高
可扩展性	中/高	资源受限	资源受限	资源受限
环境真实性	Low	低/中	High	很高
故障可发现性	不可能	Low	中	High
可复现性	High	中	Low	Low
控制	High	High	中	Low
变化速度	Fast	Fast	中	Slow
可调试性	High	中	中	Low

表 I 四种不同实验设置的比较：仿真、模拟、测试网和现实世界部署。

测试网中，可扩展性受研究人员可用计算资源的限制。在真实世界的部署中，实验规模通常受终端用户数量及其贡献的资源限制。

环境真实性。 这是使测试网和部署区别于其他实验设置的两个关键特征之一。环境真实性由三个不同参数构成：(1) 客户端异质性，例如硬件能力方面的差异；(2) 网络条件等外部参数的变化；(3) 用户行为的影响。现实世界部署涵盖了这三个参数。与测试网的主要区别在于用户行为的影响；例如，测试网可能仅向专家用户开放，从而降低其真实性。测试网中的用户激励有时也会简化，例如在区块链测试网中，通常没有经济激励。与其他实验设置相比，仿真和模拟中的环境真实性三个参数都较低。模拟可以更真实地呈现网络条件等外部参数。若在异构硬件上进行模拟，要真实地体现客户端异质性颇具挑战。一项观察是，如果依据已部署系统测量所得的已知数值来设计仿真和模拟设置，就能提升其环境真实性。两个关键因素限制了此类测量：首先，被测系统应具有非常相似的应用场景；其次，无法确定随着系统演进，测量结果是否仍然适用。

已有一些工作致力于提高仿真或模拟环境的环境真实性。Sarzyniec 等人提出了 Distem，这是一种虚拟化平台，可在同构计算集群中实现资源异质性 [23]。虽然这有助于提高实验的准确性，但现实环境中的故障模型和用户生成的工作负载并未被纳入其中。近期关于拜占庭容错共识协议可扩展性的实验工作提出了一种模拟器 [1]。了解模拟器与现实环境之间的真实性差距是这项工作的关键部分。

新故障的可发现性。 这是区分受控与非受控实验设置的另一项关键属性。某些类型的故障只能在现实环境中发现，尤其是涌现故障和用户导致的故障 [13]。测试网可以揭示某些涌现故障和部分故障 [5]。通过采用模拟设置进行实验，能够揭示的新型故障相对较少。原则上，仿真无法发现新类型的故障。

可复现性。这个参数与不同研究人员能否获得相同的设置有关。至少从理论上讲，仿真可实现最高水平的可复现性，因为所有制品都可以轻松发布。对于模拟，可以提供评估的软件，但并非总能获得完全相同的实验测试平台。有人可能会认为，尽管测试网和已部署系统的可复现性都较低，但测试网相对更容易复现实验条件，而已部署系统几乎不可能做到这一点。

控制。 仿真和模拟是实验设置中让研究人员能够最充分控制实验流程的两种方式。对于测试网和现实世界中的已部署系统，研究人员通常几乎无法控制系统的运行，因为他们依赖于运行软件的志愿者或最终用户。

变化速度。实验可被修改的速度，是区分不同实验设置的一个关键因素。在已部署的现实世界系统中，由于软件更新传播需要时间，变化速度通常最低。测试网可以支持相对更快的部署周期。在这两种情况下，部署和收集结果都可能耗时较长。仿真和模拟设置受到的外部限制相对较少，也能快速进行修改。

可调试性。在已部署系统中发现、分析和复现软件缺陷，需要专门的基础设施，而且可能耗费大量时间。相比之下，测试网和模拟具有更高的可调试性，因为研究人员拥有更高程度的控制权。只要仿真的规模不是太大，仿真就能提供最高的软件缺陷可发现性。

这项分析表明，没有部署就无法进行全面评估。我们需要考虑环境真实性和故障，而这些只有在真实场景中才能被发现。

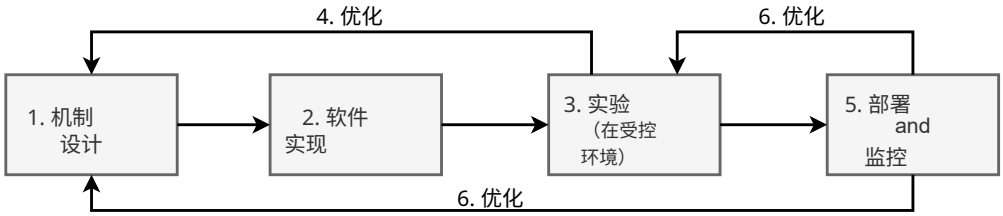


图 2. 我们的部署优先研究方法。与图 1 的主要区别在于，我们会将新的见解或真实世界的工作负载轨迹直接用于机制设计和实验（分别对应步骤 6）。

III. TRIBLER：大规模部署和监控去中心化机制

Tribler 是我们实验室的点对点文件共享软件，也是用于大规模部署和评估去中心化机制的研究平台 [8]。近二十年来，我们一直使用 Tribler 软件，从实际运行的点对点网络中获得了关于复杂交互与动态的独特见解 [22]。30 多位博士研究人员和学士／硕士生使用我们的软件评估他们的机制。Tribler 还拥有稳定的用户群，使纵向部署实验成为可能。截至本文撰写时，已有超过 180 万用户下载 Tribler 软件，每月有 40,000 名独立用户使用 Tribler。[4]

Tribler 最初被设计为一款文件共享应用程序，允许用户使用自定义的洋葱路由协议 [16] 匿名下载种子文件。Tribler 使用 IPv8 网络库，该库支持认证消息传递，并支持构建和维护去中心化覆盖网络。然而，多年来，Tribler 已从 BitTorrent 下载客户端发展为一款功能多样的应用程序，提供关键词搜索、将种子打包到频道中，以及用于应对搭便车行为的声誉机制等功能 [19]。

A. 部署优先研究方法

Tribler 是我们实验室研究方法中的重要组成部分，因为它使我们能够大规模部署和评估去中心化机制。Tribler 也让我们得以尝试一种更加注重部署的不同研究方法。基于我们使用 Tribler 的经验和部署工作，我们现在提出一种部署优先的去中心化系统设计与改进方法，该方法以持续实验周期为基础。图2展示了对图1所示传统研究方法的更新方案。我们认为，在持续实验中，系统的部署阶段并非发生在实验之后（图2中的步骤3）。相反，我们将部署视为研究方法中至关重要的下一步，安排在实验之后进行。部署研究可能带来的发现包括：发现受控环境中不会出现的新型故障，以及获得新的见解，或发现由现实环境的不可预测性引发的性能问题。这些见解会通过以下两种方式直接用于改进设计和实验（图2中的步骤6）。首先，我们利用新的见解更新并改进去中心化机制，这与标准研究方法利用实验结果进行改进的做法类似。其次，我们利用从部署中获取的信息来改进受控环境中的实验。例如，可以在内部实验中重放从实时网络获取的工作负载轨迹，以便在更真实的工作负载下评估机制。部署期间的一项重点工作是监控机制，以检测故障或异常。第III-C节将对此作进一步讨论。

我们的方法并不能取代在受控环境中进行实验的必要性。相反，部署试验所获得的数据，例如网络特征、客户端性能和用户行为，应当用于弥补其他实验设置（如仿真和模拟）的局限性。因此，这些数据能够提升本地实验的真实性，并有助于在部署前进一步验证机制。

B. 动机用例：实验代币经济学

现在，我们介绍如何在最近的一次部署试验中应用以部署为先的方法。该试验利用代币经济学来应对用户通过 Tribler 下载内容时的搭便车行为。

机制设计与目标。 点对点网络中的一个根本问题是搭便车行为，即某个对等节点从社区获取的资源多于其贡献的资源 [9]。在 Tribler 中，这表现为用户从他人处下载的数据多于其回馈的数据（做种）。早期研究表明，搭便车行为在 Tribler 中十分普遍，导致上传者减少、下载速度下降 [19]。由于我们的匿名下载机制会进一步增加资源使用，随着 Tribler 网络的发展，应对搭便车行为成为一个重要问题。

我们通过结合三种互补的机制来解决搭便车问题。每种机制都采用我们的部署优先方法，在 Tribler 中进行设计、评估和部署（另见图 3）。第一种机制是一个名为 TrustChain 的轻量级去中心化账本，以记录的形式存储网络中用户之间所有成对的带宽传输 [21]。TrustChain 专为

[4]参见 https://release.tribler.org

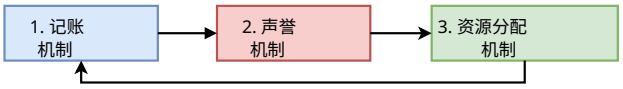


图3. 我们用于解决 Tribler 中搭便车行为的三种互补机制 [7]。我们使用部署优先方法对每种机制进行了评估。

去中心化网络中的轻量级记账而设计。由于它避免了全局共识机制，因此参与者数量增加时仍具有很强的可扩展性。用户通过一种简单的 gossip 机制与其他用户共享这些记录。第二种机制是声誉机制，它根据收到的记录计算其他用户的可信度评分。第三种机制是资源分配机制，用于决定每个用户向哪些其他用户上传数据。这些机制协同工作，使用户能够自行识别搭便车者，并因此拒绝为其提供服务，同时优先照顾诚实用户。更多详情和实验结果见我们的其他研究 [7]。

应用我们的部署优先方法。 我们部署了这三种机制中的每一种，并经历了多个部署周期，以改进和优化它们。首先，我们设计并实现了 TrustChain，并在计算集群上开展了正确性和验证实验（图2中的步骤1—3）。随后，我们将 TrustChain 集成到 Tribler 软件中，实现了一个爬虫来收集生成的记录，并发布了新的软件发布版本。由于缺乏真实世界轨迹和相关洞见，我们无法恰当地设置某些参数，例如 TrustChain 记录与其他用户共享的时间间隔。只有经历了几轮部署周期后，我们才获得了设置此类参数的相关洞见。

我们持续监控 Tribler 网络中生成的 TrustChain 记录，并得以发现许多在本地实验期间未被发现的故障和设计缺陷。例如，我们的部署显示，TrustChain 的初始设计存在不足，因为用户一次只能参与记录一笔交易。这一缺陷显著限制了记录的创建速度，而由于 Tribler 软件经常同时与其他用户通信，这也是一项关键限制。该问题促使我们重新设计 TrustChain 记录的格式，使其支持并发交易（见 [7]）。与此同时，我们利用收集到的 TrustChain 记录开始设计我们的声誉机制（见 [21]）。我们还在部署阶段发现了各种软件缺陷，例如，其中一个软件缺陷与特定版本 Windows 上偶尔发生的数据库损坏有关。

C. 经验教训

我们已使用 Tri-bler 进行了多次部署试验。由于篇幅有限，我们无法讨论应用部署优先研究方法时获得的所有见解。不过，我们将总结四条经验，这些经验是我们在研究前文所述用例以及其他部署试验时总结出来的。

教训 I：规划机制升级并维持向后兼容性。 Tribler 包含各种机制，我们会持续监控并改进这些机制。升级这些机制有时需要进行会破坏与旧版本兼容性的更改。这种兼容性中断会导致网络碎片化，因为使用某个机制不同版本的用户将无法再相互通信。此外，此类破坏性更改通常需要通过软件逻辑更新本地存储的数据（例如数据库中的数据），使其与新机制兼容。

我们力求尽量减少会破坏兼容性的机制更改，以避免网络过度碎片化，并确保新部署机制得到充分使用。我们也有此考虑，是因为观察到用户发布新版本后往往不会立即更新 Tribler 软件，尤其是在软件更新的益处不明确时。[5] 在部署试验期间，我们认识到，在设计特定机制时就应规划其升级。我们注意到，这个问题并非 Tribler 独有，因为许多区块链系统偶尔也需要通过发布新软件版本或分叉网络来升级其网络协议，例如修复安全问题或提升性能 [27]。

经验二：监控的重要性。持续监控新机制的运行情况，对于在部署过程中检测故障和异常至关重要 [28]。每次部署试验期间，我们都会开发一个爬虫，并在发布新的 Tribler 版本时部署它。该爬虫会作为对等节点加入特定的覆盖网络，向其他 Tribler 实例发送数据查询，并将检索到的信息保存到本地数据库中。这种做法类似于收集、分析和可视化区块链网络中的交易。

我们已部署多个爬虫，从实时网络收集数据。例如，除了 TrustChain账本，我们还部署了一个爬虫，用于收集用户创建的记录。然而，由于节点 churn，爬虫有时无法收集特定的数据点。因为用户可能在爬虫发送请求之前就已离线，我们的数据集并不总是包含所需的所有数据点。尽管如此，部署期间收集的数据揭示了一次由软件错误导致的大规模服务中断，因为创建的 TrustChain 记录数量显著下降。这些经验让我们认识到，监控基础设施对于规划部署至关重要。

第三课：记录所有设计决策和变更。要成功应用部署优先方法，需要充分规划，同时也会给开发者和研究人员带来独特的挑战。在早期的部署试验中，我们本可以更妥善地记录设计和部署决策，从而避免重蹈覆辙。多年来，我们

[5]请参阅 https://release.tribler.org。

采用了开放科学方法 [6]，公开记录所有源代码、设计决策和会议纪要。我们还会仔细报告在部署环境中的观察结果，并记录故障，以免在机制的后续迭代中重犯特定错误。这种开放科学方法如今已成为 Tribler 开发周期和研究方法的重要组成部分。开放科学也有助于其他研究人员理解并复现我们以往的研究成果。对于希望了解 Tribler 软件的运行方式、收集哪些数据，以及在其设备上执行哪些机制的用户而言，这些信息也很有帮助。所有这些信息都可在我们的 GitHub 代码库中公开获取。[6]

第四课：不要一次部署太多内容。我们在早期部署试验中常犯的一个错误是，倾向于在单个版本中加入多个新功能或机制。这不仅延长了软件发布版本之间的间隔，还增加了新部署的某个机制存在缺陷时导致 Tribler 软件出故障的风险，也使我们无法单独分析具体变更所产生的影响。为避免这些风险，我们目前力求每次发布最多加入一项新功能，并采用较短的发布周期。例如，第三节 B 小节用例中描述的各项机制，我们都分别通过单独的版本发布，各次发布间隔数月。

我们还了解到，机制设计是一个渐进的过程，需要经过多次迭代才能逐步完善并发挥作用。例如，在设计社会经济机制时，由于研究人员事先并不了解部署环境的动态特征，往往无法为该机制设置恰当的参数。只有根据从现实环境中收集的数据，才能使该机制更加稳健，并针对特定应用领域进行优化。

IV. 前进之路

我们认为，区块链应用等去中心化系统日益增加的复杂性及其依赖关系，需要更加稳健和成熟的实验方法。这类方法对于在现实环境中识别新型故障是必不可少的。我们认为，应将部署明确整合为研究方法中的关键步骤，以改进对去中心化机制的评估。

我们介绍了超越标准研究方法的部署优先方法。我们还介绍了 Tribler，这是我们用于部署去中心化机制的研究平台。我们展示了如何利用部署试验中的洞见，改进去中心化机制及其相关实验的设计。我们还表明，基于部署的实验设置能够：（1）揭示新类型的故障；（2）为在受控环境中设计逼真的实验奠定基础。通过介绍一个代币经济学用例，我们证明了部署优先方法在实践中的可行性。

我们的部署优先方法是一种持续演进的方法。一个可能的扩展方向，是增加用于对去中心化机制进行 A/B 测试的基础设施和方法。这种方法将向不同的用户子集提供不同的算法和参数。

参考文献

[1] Christian Berger 等人。我的拜占庭容错协议实现具备可扩展性吗？载于 DICG，2022 年。[2] Alyssa Blackburn 等人。去中心化失败期间，一个匿名群体的合作保护了比特币。arXiv，2022 年。[3] Gordon Blair。《复杂分布式系统：需要新的视角》。载于 ICDCS，2018 年。[4] Tomasz Buchert 等人。分布式系统通用实验管理工具综述。FGCS，2015 年。[5] Franck Cassez 等人。以太坊 2.0 信标链的形式化验证。载于 TACAS'22，2022 年。[6] Sophia Cr¨uwell 等人。开放科学的七个简单步骤。《心理学杂志》，2019 年。[7] Martijn de Vos 等人。Contrib：通过核算工作量，在去中心化大型科技替代方案中维护公平性。《计算机网络》，2021 年。[8] Martijn de Vos 等人。Tribler/tribler：v7.12.1。https://doi.org/10.5281/ zenodo.7097138，2022 年 9 月。[9] Michal Feldman 等人。点对点系统中的搭便车与洗白。载于 PINS 研讨会，2004 年。[10] 以太坊基金会。宣布以太坊基金会及共同资助的项目。https://blog.ethereum.org/2019/08/26/announcing-ethereum-foundation-and-co-funded-grants，2019 年。访问日期：2023-01-11。[11] Richard P. Gabriel 等人。超大规模系统。载于 OOPSLA，2006 年。[12] Peter Garraghan 等人。涌现故障：重新思考大规模《云计算》的可靠性。《云计算》，2018 年。[13] Saurabh Gupta 等人。大规模系统中的故障：长期测量、分析及其影响。载于《超级计算》，2017 年。[14] Jens Gustedt 等人。大规模系统的实验方法：综述。《并行处理快报》，2009 年。[15] Andreas Haeberlen 等人。评估去中心化系统时的谬误。载于国际点对点系统研讨会，2006 年。[16] Rolf Jagerman 等人。去中心化隐私增强技术的十五年奋斗。arXiv，2014 年。[17] Bahman Javadi 等人。故障轨迹档案：实现分布式系统故障测量结果与模型的比较。《JPDC》，2013 年。[18] Dusica Marijan 和 Chhagan Lal。区块链验证与确认：技术、挑战与研究方向。《计算机科学评论》，2022 年。[19] Michel Meulpolder 等人。Bartercast：一种防止点对点网络中懒惰搭便车的实用方法。载于 ISPDP，2009 年。[20] Bulat Nasrulin 等人。Gromit：区块链系统性能与可扩展性基准测试。载于 DAPPS，2022 年。[21] Pim Otte 等人。TrustChain：抗女巫攻击的可扩展区块链。《FGCS》，2020 年。[22] Johan Pouwelse 等人。Tribler：一种社交型点对点系统。《并发与计算：实践与经验》，2008 年。[23] Luc Sarzyniec 等人。分布式系统虚拟实验环境的设计与评估。载于 PDP，2013 年。[24] Claudio J Tessone 等人。区块链共识的随机建模。arXiv，2021 年。[25] Alessandro Vespignani。预测技术社会系统的行为。《科学》，2009 年。[26] Guosai Wang 等人。我们能从四年的数据中心硬件故障中学到什么？载于 DSN，2017 年。[27] Alexei Zamyatin 等人。一个狂野的软分叉出现了！实践中的包容性区块链协议变更。载于《金融密码学》，2018 年。[28] Niels Zeilemaker 等人。Tribler：点对点媒体搜索与共享。载于 ACM 多媒体，2011 年。[29] Liyi Zhou 等人。SoK：去中心化金融（DeFi）攻击。密码学电子预印本档案，2022 年。

[6]参见 https://github.com/tribler/tribler/issues。