



Arweave AO Summary Report

Version 0.1

David Zhang

March 06, 2024

Arweave AO Summary Report

David Zhang

March 06, 2024

Arweave AO Summary Report

Prepared by: David Zhang

Table of contents

See table

- **1. AO概述**
- 1.1 特点
- 1.2 小结
- **2. 核心功能**
- 2.1 并行运行的进程
- 2.2 *无需节点执行计算达成共识
- 2.3 原生无界硬盘
- 2.4 *自动激活合约
- 2.5 *支持扩展的模块化架构
- **3. 技术架构**
- 3.1 Process
- 3.2 Message
- 3.3 SU
- 3.4 CU
- 3.5 MU
- **4. AO系统中消息处理和状态计算的流程**
- 4.1 User
- 4.2 MU
- 4.3 SU
- 4.4 CU
- 4.5 结果
- **5. 应用场景**
- 5.1 Akord存储和文件加密服务
- **6. AO Cookbook Testnet 功能DEMO**
- 6.1 消息传递 DEMO
- 6.2 创建聊天室 DEMO
- 6.3 创建和发送代币 DEMO
- 6.4 对聊天室进行令牌门控 DEMO
- 6.5 机器人和游戏 DEMO
- **7. 结论**

1. AO 概述

AO计算机采用了演员模型(Actor Model)的设计理念，强调每个参与者(节点)作为独立的实体，可以发送消息、处理任务，并与其他节点互动。这种模型支持高度并发和分布式的计算，适合构建去中心化的应用。

1.1 特点

1.1.1 运行环境:建立在Arweave网络上，Arweave网络提供数据永久存储的去中心化网络。Arweave的特性使得AO计算机能够利用其永久存储能力，存储重要的数据和状态信息，从而支持去中心化应用的长期运行和数据的不可篡改性。

1.1.2 Actor Model节点:在AO计算机系统中，节点(参与者)运用Actor Model进行消息传递和并发计算。

- **Ethereum节点:**以太坊的设计重点在于支持去中心化应用(dApps)和智能合约的运行。以太坊网络中的节点包括全节点和轻节点，它们共同维护区块链的完整性和安全性。全节点存储完整的区块链数据，验证所有交易和区块，而轻节点则依赖于全节点获取区块链信息，但不存储完整的区块链数据，以减轻资源需求。
- **Actor Model节点:**更注重提供一个分布式、去中心化的计算环境，其中节点可以独立执行计算任务并通过消息传递协同工作。这强调了系统的计算能力和并发处理能力，而非仅仅维护一个不可变的账本。

1.1.3 共识机制:AO的共识机制采用了一种创新的方法来确保计算结果的正确性和可信度。这种机制基于节点间的计算验证和结果共识，而非传统的全部节点执行相同计算来达成共识。

- **计算任务的分发与执行:**网络中的一个节点(如节点A)接收到一系列计算任务，并执行这些任务得到结果。这个过程不仅包括了计算本身，还包括将计算结果及其加密签名证明发布到Arweave网络上。
- **结果的发布与验证:**节点A完成计算后，将计算结果和加密签名的证明一同发布到Arweave网络。这个证明对于其他节点(如节点B和节点C)来说是可访问的，允许它们对节点A的计算结果进行独立验证。

- 独立验证与共识构建：其他节点通过读取节点A提供的原始数据和证明，并执行相同的计算过程，来验证节点A的计算结果的正确性。如果节点B和节点C能复现节点A的结果，并且这些结果与节点A提供的证明相匹配，那么它们可以对结果的正确性达成共识。
- 共识的确认与应用：一旦多数节点（通常要求超过三分之二的节点，即 $n > 2/3 * \text{总节点数}$ ）验证并支持某个计算结果，这个结果就被认为是网络上可信的。确认后的结果被记录在Arweave网络上，用户可以基于这些可信结果获取分析结果并进行进一步的使用，如决策支持或数据处理。

这种共识机制的与以太坊(PoW, PoS)相比优势在于，它大大减少了网络中每个节点必须执行的计算量。因为不是所有节点都需要执行完整的计算任务，这种机制不仅提高了效率，降低了资源消耗，还通过分布式验证增强了结果的可信度。通过这种方式，AO系统能够在保持去中心化的同时，提供一个高效且可靠的计算平台。

1.1.4 计算能力：

- 计算的统一环境：AO构建了一个整体统一的计算平台，尽管它背后由分布在全球各地的众多节点（即计算资源）组成。对于使用该平台的用户和开发者来说，每个进程可独立于其他进程运行在不同的计算节点上，它们之间通过消息传递相连，就像网站通过超链接连接起来一样，为用户和开发者提供一个连贯和统一的体验。用户感知到的是一个连贯且无缝的计算环境，就像是在使用单一的计算机一样。
- 计算的分布式和异构性：AO的计算资源并非集中于传统的云计算或服务器的单一位置或单一规格的机器中，而是由各种不同能力和配置的节点构成。这些节点分布在全球各地，这种分布式和异构的特性为AO提供了高度的可扩展性和鲁棒性。
- 单节点和跨节点并行进程：如果一个节点具有足够的计算资源（如多核CPU），那么多个计算任务可以在这个单一节点上并行执行。这种情况下，任务并行性是通过节点内部的资源调度来实现的，例如通过操作系统的多任务处理能力。更常见的是，AO计算机系统利用其分布式网络的优势，在不同节点上并行运行多个计算任务，通过消息传递层，这些分散运行的任务能够协同工作，共同完成更复杂的应用逻辑。这种分布式并行处理方式可以更好地扩展计算能力，利用网络中各个节点的空闲资源，从而提高整体系统的处理速度和效率。

1.1.5 存储能力：

- **数据永久性:** Arweave使用了一种称为区块织网的数据结构,不仅每个区块包含了对前一个区块的引用,而且还包含了对历史区块的引用,这增强了整个网络的数据完整性和可访问性。Arweave设计理念中的一个核心特性是数据的永久存储。用户支付一次性费用,数据即可永久保存在网络上,不会因时间推移而消失或变得不可访问。
- **一次性支付:** Arweave网络通过对矿工提供存储空间的经济激励,鼓励网络参与者贡献更多的存储资源。随着参与者增多,网络的总存储容量随之增加,用户为数据的永久存储支付一次性费用,这个费用基于存储数据的大小。这种支付模型使得存储大量数据变得经济可行,同时确保了数据长期的可访问性。
- **数据去重:** Arweave网络还可以识别和去除重复数据,有效减少不必要的存储占用,这使得网络能够更高效地存储大量数据。

1.1.6 兼容性:

- **模块化:** 这种架构允许现有的智能合约平台和其他计算框架轻松地“插入”AO网络,成为网络中的一个节点或进程,与其他进程交换消息和数据。这提高了AO系统的兼容性和扩展性,允许它支持多种不同的应用场景和技术栈。例如使用EVM(以太坊虚拟机)作为智能合约执行环境,因为EVM拥有广泛的开发者社区和丰富的开发工具,以及支持高并发的虚拟机配置。
- **排序模型:** 假设一个开发者,想要在AO计算机系统上构建一个去中心化的社交媒体应用。这个应用需要处理大量的用户生成内容,比如帖子、评论和图片,同时也需要保证数据的安全性和可靠性。由于应用需要实时处理和显示用户的帖子和评论,开发者可以选择一种支持快速消息传递和处理的排序模型,以确保内容能够按照时间顺序正确显示。
- **消息传递安全保证:** 为了保护用户的隐私和数据安全,开发者可以采用端到端加密的消息传递机制,确保只有发送和接收方能够阅读消息内容。
- **支付选项:** 比如应用提供了基于加密货币的支付方式,允许用户购买增值服务,比如推广帖子或购买特殊表情包。你选择了一个支持多种加密货币的支付模块,以便用户可以使用他们偏好的货币进行交易。

1.1.7 AOs去中心化操作系统：

AOs允许开发者在其去中心化网络中启动和运行命令行进程，这些进程在功能上类似于传统智能合约，但在一个无需信任和完全去中心化的环境中运行。这使得开发者能够创建和部署跨越不同物理位置的应用。

1.1.8 底层语言：Erlang

1.1.9 开发交互语言：在AO系统中，Lua语言用于编写智能合约和定义进程逻辑，使开发者能够创建复杂的应用程序逻辑并在AO网络上执行。

1.2 小结：

1.2.1 设计目的和应用范围：

- 以太坊：主要是一个去中心化的智能合约平台，专注于构建去中心化应用(dApps)和执行可编程的智能合约。
- AO：一个更为广泛的去中心化计算平台，旨在提供无需信任的计算服务和去中心化存储，不仅限于智能合约，还包括更复杂的计算任务和数据处理。

1.2.2 计算和存储能力：

- 以太坊：计算能力主要局限于智能合约的执行，存储能力也受到成本和技术的限制。
- AO：旨在提供更广泛的计算能力和大规模的数据存储，通过模块化的设计和整合Arweave等去中心化存储解决方案，支持更复杂的应用场景。

1.2.3 灵活性和模块化：

- 以太坊：虽然支持多种编程模式和dApps的开发，但在虚拟机(EVM)、智能合约语言(如Solidity)等方面相对固定。
- AO：提供更高的灵活性，允许用户根据需求选择不同的虚拟机、编程模型和存储选项，支持更广泛的技术栈和应用需求。

1.2.4 AO的优势：

- 扩展性：通过模块化设计和分布式架构，AO能够支持更大规模的数据存储和复杂的计算任务，适合处理大数据和高性能计算需求。

- **灵活性:** 用户可以根据自己的需求定制计算环境，选择合适的虚拟机、数据处理模型和存储方案，为各种场景提供了广泛的支持。
- **去中心化存储:** 与Arweave的整合为AO提供了持久、不可篡改的数据存储能力，增强了数据的安全性和可靠性。

1.2.5 AO的缺点：

- **复杂性:** 由于AO旨在提供更广泛的功能和更高的灵活性，这可能导致其架构和使用比较复杂，对于开发者和用户来说，可能存在一定的学习门槛。
- **成本问题:** 尽管AO通过去中心化存储解决方案如Arweave提供永久存储，但大规模的数据存储和复杂计算任务可能会涉及较高的成本，尤其是在网络使用率高的情况下。
- **生态系统和支持:** 与以太坊这样成熟的平台相比，AO作为一个新兴的计算平台，其开发者社区、工具、文档和现成的解决方案可能相对有限，这可能影响其快速采用和发展。
- **安全性:**

以太坊: 智能合约已经被广泛研究和审计，存在一套成熟的最佳实践和安全工具，用于识别和修复智能合约中的安全漏洞。然而，智能合约本身的错误或漏洞仍然是安全风险的一个主要来源。

AO: 旨在提供一个更广泛的计算平台，可能会整合多种虚拟机和执行环境，包括但不限于智能合约。AO模块化和可扩展设计意味着它可能会整合多种技术和组件。每个组件的安全性都需要独立考虑和验证，整体系统的安全性取决于所有组件的最低安全标准。

对于用户来说，AO代表一台共享计算机，他们可以在其中执行多个进程。这些进程并不局限于任何特定的服务器或任何单个个人或团体的管辖之下。一旦激活，这些流程就会以加密安全的方式提供服务，确保公正和永久的运行。这种设计使用户能够依赖长期持续维护其权利的服务，从而营造一个与系统交互的可信环境。

2. 核心功能

2.1 并行运行的进程

在AO中，应用程序可以由任意数量的进程（在这里可以类比为“合约”）构成。这些进程可以并行运行，充分利用可用的计算资源，而不会相互干扰。这种设计提高了系统的并

行性和效率。AO进程间不共享内存，而是通过一种本机消息传递标准进行协调和通信。这种方法减少了进程间的耦合，提高了系统的可靠性和扩展性。

示例：假设我们要用AO构建一个去中心化的社交网络平台。在这个平台上，每个用户的活动（如发布动态、评论、发送消息）都可以被视为一个独立的进程。这些进程之间通过消息传递来交互，例如，一个用户的动态发布进程可能会向其关注者的消息接收进程发送消息，通知他们有新的动态可以查看。

- **动态发布进程**：用户发布动态时，系统为该动态创建一个进程，该进程负责处理与该动态相关的所有活动（如评论、点赞）。
- **消息接收进程**：每个用户都有一个或多个进程，用于接收来自他人动态的更新或消息。
- **评论处理进程**：当用户评论某个动态时，系统为该评论创建一个独立的进程，负责处理所有与该评论相关的回复或点赞。

这个例子展示了如何通过将社交网络平台上的活动拆分为多个独立进程，利用AO的并行处理和消息传递机制，构建一个高效、可扩展的去中心化应用。每个进程都可以独立运行，通过消息传递与其他进程交互，而不需要担心性能瓶颈或进程间的干扰，从而实现了类似于传统Web2/分布式系统的扩展性。

* 2.2 无需节点执行计算达成共识

在传统的区块链网络中，节点通常需要执行所有的计算任务来验证交易和更新网络状态。相比之下，AO网络允许节点在不执行实际计算的情况下，通过进程消息日志达成关于程序状态的共识。这意味着节点不需要处理复杂计算来维持网络的一致性和安全性。

状态的更新和验证依赖于由Arweave托管的进程消息日志。这些日志为网络中的状态变更提供了“全息”暗示，即使不执行实际的计算，节点也能基于这些日志达成关于状态变化的共识。

在AO网络中，计算的成本和责任被委托给用户。用户可以选择自己计算所需的状态，或者请求由他们选择的节点来执行所需的计算。这种模式有效地将计算需求和成本从网络转移到了用户，从而降低了维护网络所需的资源和成本。

示例：去中心化投票系统

假设我们使用AO网络构建一个去中心化的投票系统。在这个系统中，每一次投票都是一个独立的进程，投票的结果和统计数据由进程消息日志记录。

- **投票进程**: 每当有新的投票发起时，系统就会创建一个对应的进程，记录所有关于这次投票的信息和参与用户的投票行为。
- **进程消息日志**: 所有的投票行为和结果都被记录在由Arweave托管的进程消息日志中，这些日志为整个网络提供了一个关于当前投票状态的全面视图。
- **计算投票结果**: 在需要获取投票结果时，用户可以选择自己下载并计算这些日志来确定最终的投票结果，或者他们可以请求某个节点为他们完成这项计算，并获取结果。

通过这种方式，AO网络中的去中心化投票系统不需要所有节点执行相同的计算来达成共识，而是通过共享的进程消息日志实现了状态的同步和验证，大大降低了网络的运行成本，同时也为用户提供了灵活的计算选择。

2.3 原生无界硬盘：

Arweave提供的去中心化存储被形象地描述为一个“原生无界硬盘”，它允许AO进程直接加载和执行任意大小的数据到内存中，并且可以将计算结果写回到这个网络上。这突破了传统区块链智能合约在数据存储和处理能力上的限制。

传统的智能合约平台，如以太坊，通常对存储空间、执行时间和操作的复杂性有一定限制。相比之下，AO提供的计算模型消除了这些典型的资源限制，支持完全并行的执行和处理大规模数据集，为开发者提供了更大的灵活性。

通过允许无限制地加载和处理数据，AO为那些需要大量数据处理和计算资源的复杂应用程序，如机器学习任务和高计算自主代理，提供了新的可能性。

示例：去中心化的机器学习平台

假设我们要使用AO系统和Arweave存储构建一个去中心化的机器学习平台。在这个平台上，研究者和开发者可以共享、训练和部署机器学习模型，而无需担心数据存储或计算能力的限制。

- **数据存储**: 大量的训练数据集可以直接存储在Arweave上，由于其“原生无界硬盘”的特性，存储空间不再是限制因素。
- **模型训练**: 开发者可以启动AO进程来加载存储在Arweave上的数据集，利用可用的计算资源进行模型训练。这个过程可以完全并行执行，大大缩短了训练时间。

- 模型部署和执行:训练好的模型可以被直接部署在AO网络上,供其他用户和应用调用。由于没有执行大小和复杂度的限制,可以轻松部署和执行大型复杂模型。

这个例子展示了如何利用AO系统和Arweave存储的组合,为机器学习和其他数据密集型应用提供了一种去中心化、高效和灵活的解决方案。这不仅超越了传统智能合约平台的限制,还为复杂应用的开发打开了新的大门。

*2.4 自动激活合约:

在传统的智能合约平台(如以太坊、Solana、Polygon等)中,合约的执行通常需要用户发送一个交易来“唤醒”合约,然后合约才执行相应的计算或逻辑。这意味着,如果没有用户或外部交互,合约就处于一种非活跃状态,限制了智能合约的应用场景,特别是对于那些需要定时或自动执行任务的场景。

AO系统通过引入一种类似于传统编程中“cron”作业的机制来克服这一限制。在这种机制下,合约可以被预设为在特定时间间隔自动执行,无需外部的手动触发。这种自动化的交互方式为开发复杂且自动化的去中心化应用提供了可能。

任何用户或者实际上是进程本身,都可以向网络的节点支付费用来“订阅”这些自动执行的合约。通过支付这些费用,用户可以保证合约按照设定的频率自动唤醒并执行计算任务,从而实现了一种自我维持的自动化服务。

- **Chainlink Automation**:适用于需要根据外部事件或链上条件变化触发的任务,如定期支付、市场价格更新、复杂金融合约的执行等。
- **AO系统的自动激活合约**:更适合于需要定时执行的任务,无论外部条件如何,都按照预设的时间间隔自动运行,如定期数据抓取、内容更新、维护性计算任务等。

*2.5 支持扩展的模块化架构:

开放数据协议:AO的核心是一个开放的数据协议,这意味着任何人都可以构建并实现该协议的应用。这种开放性鼓励了创新和多样性,允许不同的开发者和团队根据自己的需求定制和扩展系统。

模块化架构:AO系统设计成模块化的,使得其各个组成部分(如定序器、消息传递中继器和虚拟机等)都可以被随意交换和扩展。这种模块化不仅提高了系统的灵活性,也使得维护和升级变得更加简单高效。

生态系统互操作性: AO的设计允许Arweave生态系统中现有的智能合约系统(例如Warp、Ever、Mem等)能够插入到AO统一网络中，并能够从统一网络发送和接收消息。这意味着不同的智能合约系统可以共享相同的基础设施和工具，促进了不同系统和应用之间的协作和数据共享。

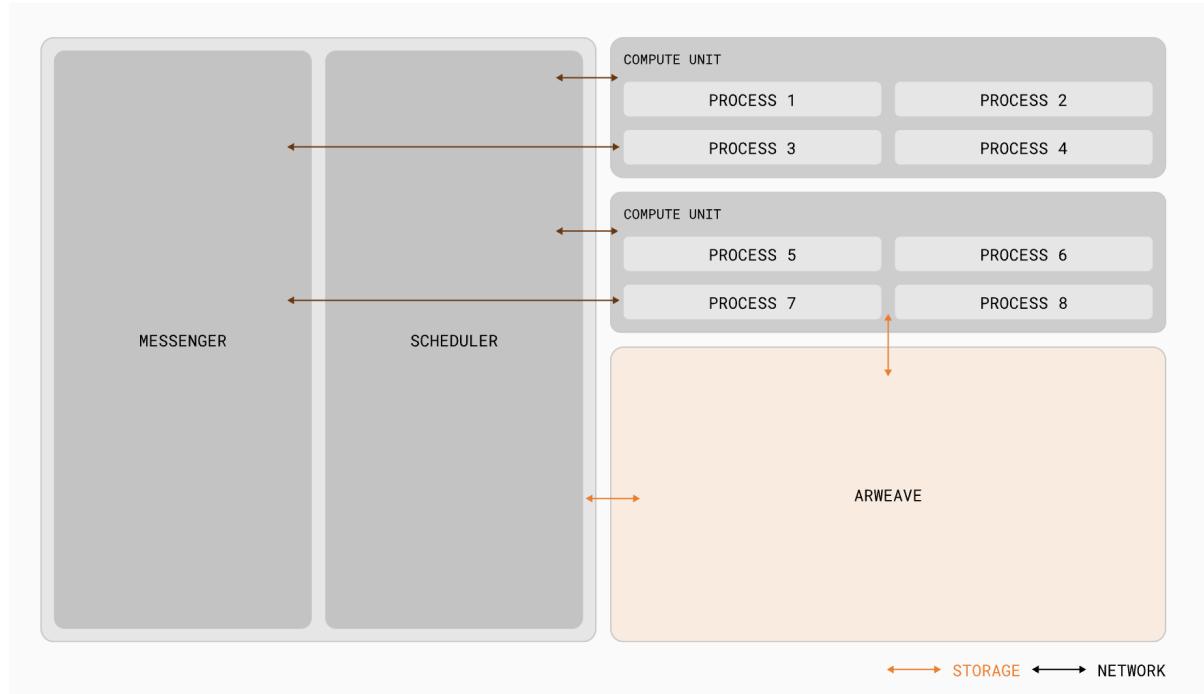
示例:去中心化社交网络

假设我们要在Arweave生态系统中构建一个去中心化的社交网络。在AO的模块化架构下，这个社交网络可以利用不同的智能合约系统来实现其不同的功能模块，比如使用Warp合约系统来处理用户身份验证，使用Ever系统来管理内容的存储，而Mem系统则用于实现即时消息传递。

- **用户身份验证(Warp):**利用Warp合约系统的特性，为社交网络的用户提供安全可靠的身份验证机制。
- **内容存储(Ever):**通过Ever系统，将用户发布的内容存储在Arweave的永久存储上，确保数据的不可篡改和长期可访问。
- **即时消息传递(Mem):**使用Mem系统处理用户之间的即时消息传递，确保通讯的实时性和私密性。

在AO系统中，这些不同的智能合约系统可以作为模块插入统一网络，并共享基础设施如消息传递协议，从而在Arweave上提供一个连贯且高效的计算体验。这不仅减少了重复建设的工作量，也使得不同的功能模块能够无缝协作，为用户提供一个综合且高效的服务。

3. 技术架构



3.1 进程 (Process)

- 这是网络的计算单元，每个进程由存储在Arweave上的消息日志和初始化数据表示。
- 进程在启动时定义其所需的计算环境(包括虚拟机类型、调度程序、内存需求等)。
- 进程不直接共享内存，而是通过消息传递进行通信。
- 进程可以接收来自用户钱包的消息，也可以接收和转发其他进程的消息。
- 进程的开发者可以设置消息的可信度标准。

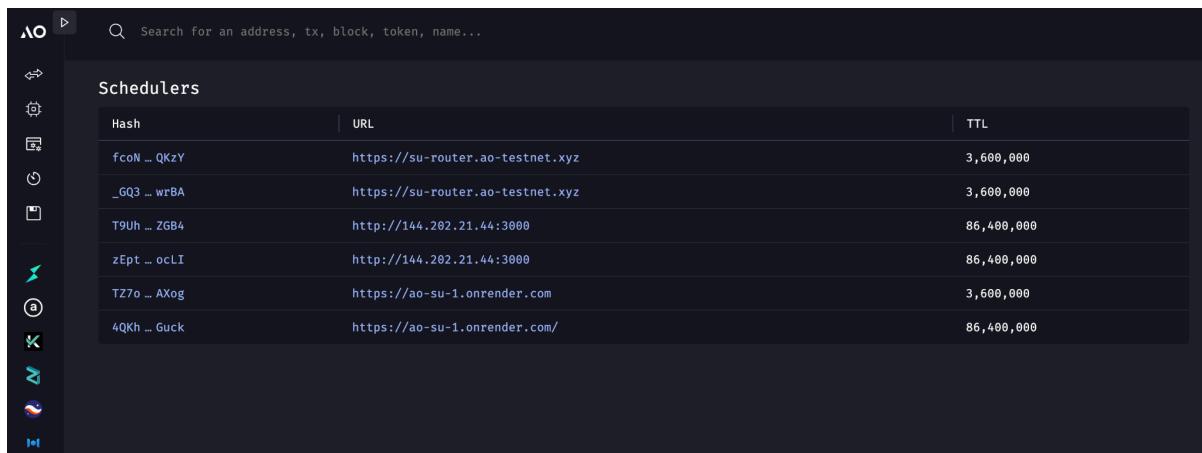
Processes							
Name	Module	SDK	Scheduler	Content Type	Creation	Creator	
PaLayer2	9afQ ... 1Pt4	aoconnect	_GQ3 ... wrBA	text/plain	9 minutes	HxVj ... vVF0	
name4	9afQ ... 1Pt4	aoconnect	_GQ3 ... wrBA	text/plain	9 minutes	VAdo ... 1Cvg	
name	9afQ ... 1Pt4	aoconnect	_GQ3 ... wrBA	text/plain	9 minutes	VAdo ... 1Cvg	
default	9afQ ... 1Pt4	aoconnect	_GQ3 ... wrBA	text/plain	9 minutes	Vi5 ... RjLI	
aftr-test	9afQ ... 1Pt4	aoconnect	_GQ3 ... wrBA	text/plain	23 minutes	L_lw ... XoXA	
marchbot2	9afQ ... 1Pt4	aoconnect	_GQ3 ... wrBA	text/plain	23 minutes	VN7H ... N_io	
work	9afQ ... 1Pt4	aoconnect	_GQ3 ... wrBA	text/plain	27 minutes	bAJY ... y0lU	
7i7oken	9afQ ... 1Pt4	aoconnect	_GQ3 ... wrBA	text/plain	33 minutes	0gPO ... 7Qcw	
default	9afQ ... 1Pt4	aoconnect	_GQ3 ... wrBA	text/plain	33 minutes	p6yG ... ErU4	
9U1g ... BA1o	9afQ ... 1Pt4	aoconnect	TZ7o ... Axog	text/plain	33 minutes	YeNi ... BfDw	
default	9afQ ... 1Pt4	aoconnect	_GQ3 ... wrBA	text/plain	41 minutes	Q0c8 ... 6wiY	
default	9afQ ... 1Pt4	aoconnect	GQ3 ... wrBA	text/plain	41 minutes	hUS ... 3P6k	

3.2 消息(Message)

- AO系统中的每次与进程的交互都通过消息表示。
- 消息基于ANS-104数据项标准构建。
- 用户和进程可以通过调度器单元向其他进程发送消息。
- 消息传递保证至多传递一次，如果消息没有被信使单元转发或接收者没有处理，那么消息传递不会完成。

3.3 调度器单元(Scheduler Unit, SU)

- 负责为发送到进程的消息分配原子递增的槽编号。
- 必须确保消息数据被上传到Arweave，使其永久可访问。
- 进程可以选择其首选的调度器，不同的实现方式提供不同的分散或集中控制。



The screenshot shows the Arweave AO interface with a sidebar containing various icons. The main area is titled "Schedulers" and displays a table with the following data:

Hash	URL	TTL
fcoN ... QKzY	https://su-router.ao-testnet.xyz	3,600,000
_GQ3 ... wrBA	https://su-router.ao-testnet.xyz	3,600,000
T9Uh ... ZGB4	http://144.202.21.44:3000	86,400,000
zEpt ... oclI	http://144.202.21.44:3000	86,400,000
TZ7o ... AXog	https://ao-su-1.onrender.com	3,600,000
4QKh ... Guck	https://ao-su-1.onrender.com/	86,400,000

3.4 计算单元(Compute Unit, CU)

- 计算节点，用于计算AO中进程的状态。
- CU不需要计算进程的状态，但提供此服务，基于价格、计算要求等参数相互竞争。
- CU提供对计算结果的签名证明，可以发布供其他节点加载的签名状态证明。

3.5 信使单元(Messenger Unit, MU)

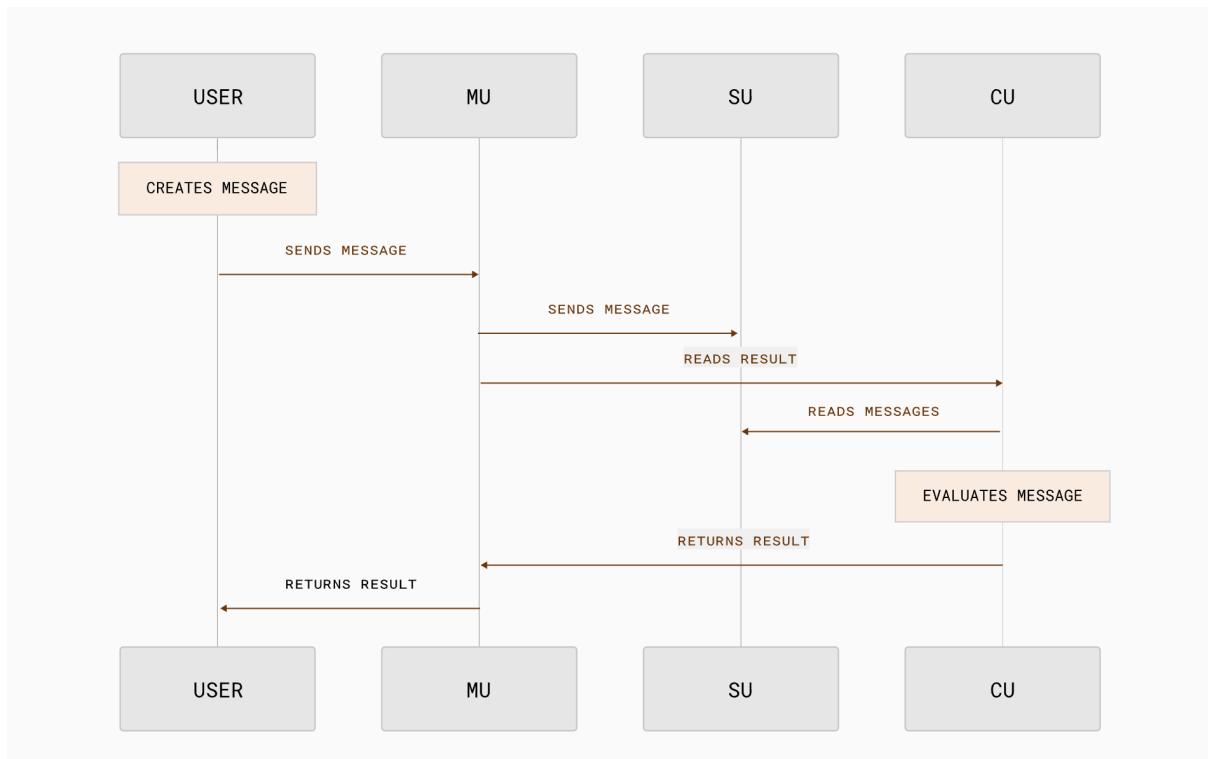
- 负责在AO网络中中继消息的节点。
- 当MU发送消息时，它们首先将其发送到SU进行排序，然后与CU协调计算结果并递归处理任何生成的发件箱消息。
- 用户和进程可以支付MU来订阅进程，从而自动处理定时任务生成的消息。

整体上，这个架构图显示了一个高度模块化和灵活的去中心化计算环境，各个单元之间通过消息传递进行协作，用户和开发者可以根据需求选择不同的虚拟机、支付方式、调度程序类型和消息安全设置。这种设计实现了去中心化的计算和存储，同时允许大规模的数据处理和复杂的应用逻辑，而无需就昂贵的计算本身达成共识。

The screenshot shows the Arweave AO interface with a search bar at the top. Below it, a table titled "Transactions" lists three entries:

- Hash: ZA6-...q0GA, Action: Eval, From: zaJJ...ILAM, To: yvlw...wQII, Size: 5 B, Timestamp: 5 minutes ago, Height: 1,377,997. Status: SDK aoconnect, ao.is, No output.
- Hash: cDNF...46nQ, Action: Eval, From: V-G3...Z1uA, To: y9Jd...9qs4, Size: 10 B, Timestamp: 5 minutes ago, Height: 1,377,997. Status: SDK aoconnect, Loading..., New Message From V-G...1uA: Action = Eval.
- Hash: crat...g9yY, Action: Info, From: bAJY...y0lU, To: 9CyZ...3Y68, Size: 4 B, Timestamp: 5 minutes ago, Height: 1,377,997. Status: SDK aoconnect, Content-Type: text/plain.

4. AO系统中消息处理和状态计算的流程



4.1 用户 (User)

- 用户创建一条消息，这可能是一个智能合约的调用、对某个进程的查询或者触发某个操作的请求。

4.2 信使单元 (Messenger Unit, MU)

- 用户发送消息后，MU接收到这条消息。MU负责将用户的消息传递给下一个处理步骤。

4.3 调度器单元 (Scheduler Unit, SU)

- 接收到MU发送的消息后，SU为这条消息分配一个唯一的槽编号，保证消息在系统中的顺序，并发送给计算单元。
- SU还负责读取计算单元返回的结果。

4.4 计算单元 (Compute Unit, CU)

- CU读取SU分配编号的消息，并对消息进行评估，也就是执行消息所需的计算任务。
- 计算完成后，CU将结果返回给SU。

4.5 返回结果：

- 最终结果通过SU返回给MU，然后MU将结果返回给用户。

这个流程确保了在AO系统中，用户发送的每个消息都经过严格的处理流程，并且计算结果可以被系统内的其他组件或用户本身所访问。

示例：去中心化交易平台

假设我们有一个基于AO系统的去中心化交易平台，用户可以通过这个平台购买和出售加密货币。

- 创建订单：用户想要购买一定数量的加密货币，于是在交易平台上创建一个购买订单的消息。
- 消息处理：用户发送的购买消息被MU接收，然后传递给SU进行排序和编号。

- 订单执行: 经过SU处理的购买消息被传递到CU, CU执行必要的计算, 以确定市场上是否有足够的卖方以满足购买要求, 以及交易的价格。
- 返回交易结果: 计算结果被返回给SU, SU确保结果的可访问性, 并将交易结果传回给MU。
- 通知用户: MU将最终的交易结果通知用户。如果成功匹配到卖家, 那么交易被执行, 用户的钱包将更新为新的货币余额。

通过这个例子, 我们可以看到用户的交易请求被系统以一种去中心化、自动化的方式高效处理, 并确保了交易的正确执行和用户的即时通知。

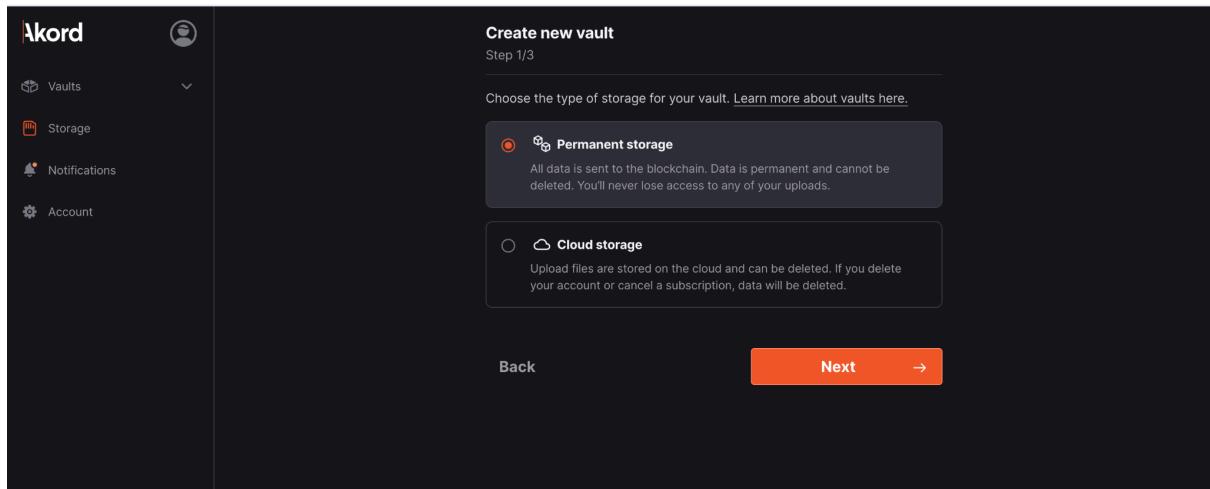
5. 应用场景

5.1 Akord存储和文件加密服务

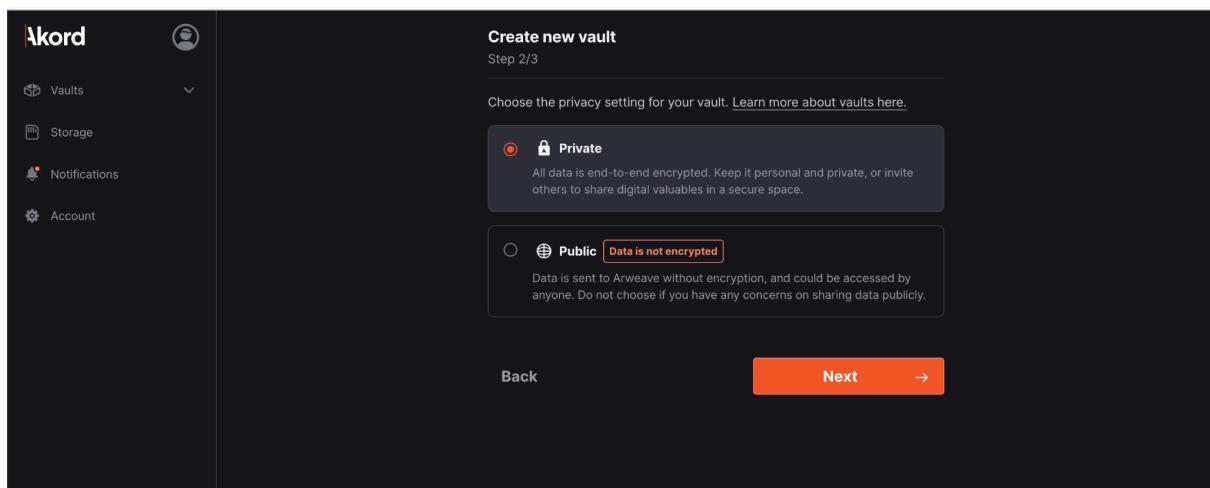
5.1.1 通过邮箱注册账户, 生成 12 个助记词

The image shows the Akord website and its mobile application interface. The website header includes 'Akord' logo, navigation links for 'Solutions', 'Platform', 'Pricing', 'Developers', 'About', 'Log in', and 'Sign up'. The main content area features a large image of a woman painting, with the text 'The simplest way to save files forever'. Below it, a sidebar highlights 'Akord's intuitive app abstracts away the complexity of web3 – secure files publicly or encrypted in a few clicks.' It includes a 'Upload to Arweave 100 MB free' button and a statistic '(a) 600,000+ files uploaded with Akord'. The mobile app interface on the right shows a file list with items like 'Bonus Art', 'Exhibition photos', 'Project Description.pdf', 'Berlin event.mp4', 'Comp Prize_signed.pdf', and 'Team interview.mp3'. A large preview image of a painting is visible on the right side of the app screen.

5.1.2 用户Dashboard, 创建Vault, 选择存储方式 : 链上(Arweave)/链下(Cloud)



5.1.3 选择存储文件是否需要加密 : Private/Public



5.1.4 添加 Vault 相关信息

Create new vault
Step 3/3

Title
Test

Description
Enter a description for your vault, you have a max of 300 characters. A description can help your vault be found.
this just for test

Tags
Enter any number of tags for your vault. Tags can help your vault be found.
test

Finish entering a tag with a comma.

Back Create vault

Vaults

Active Inactive

Name	Type	Privacy	Modified on	Size
Test	File	Locked	06/03/2024 at 20:18	-

Search New vault

Test

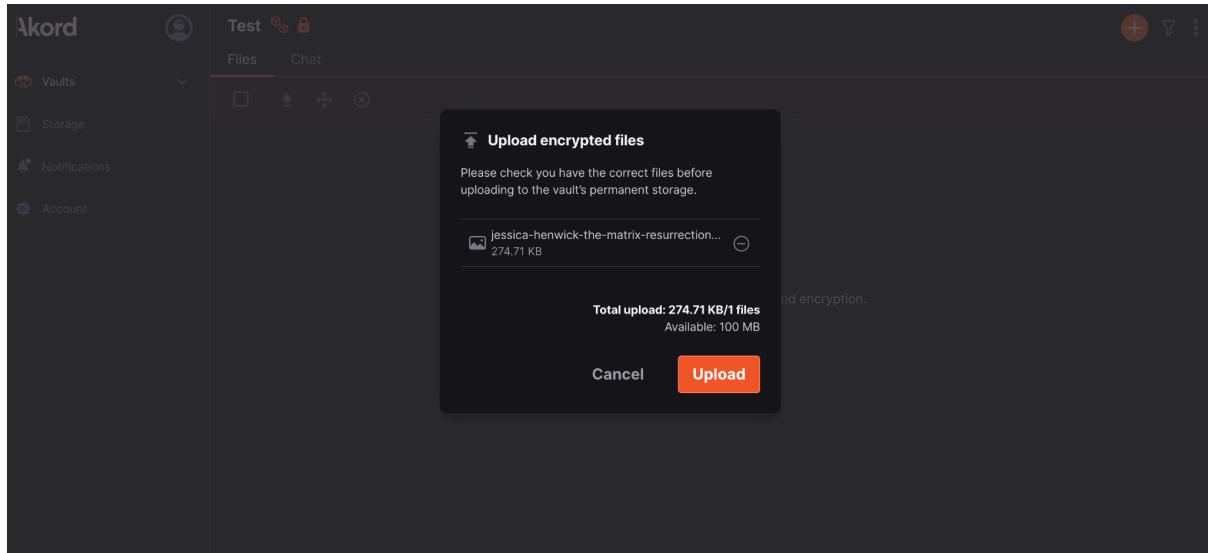
Files Chat

Add a file

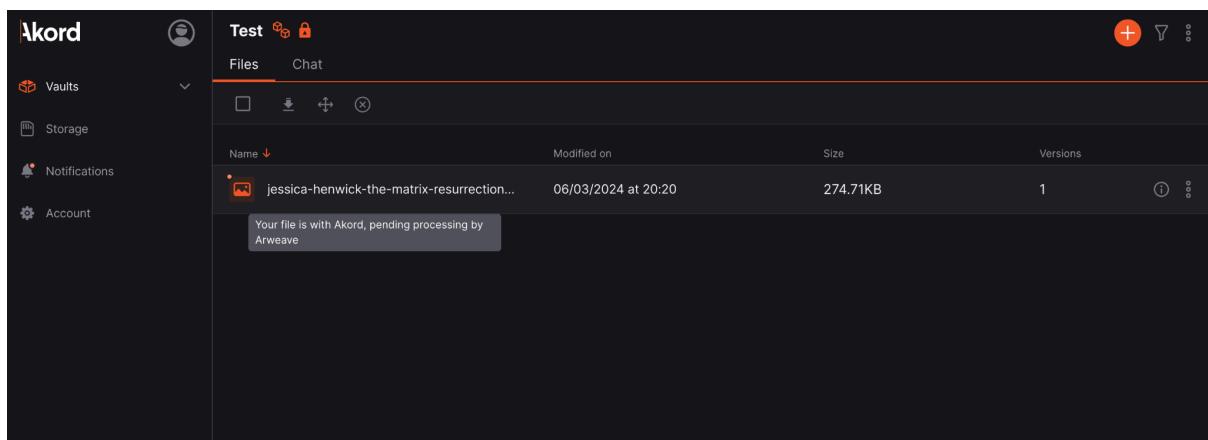
Upload a file

(Vault创建成功)

5.1.5 准备上传一张图片到新创建 Test Vault



5.1.6 上传成功，显示文件相关信息在 Vault Dashboard



5.1.7 查看上传文件 Transaction Hash 和交易数据

The screenshot shows the Akord interface. On the left, there's a sidebar with 'Vaults', 'Storage', 'Notifications', and 'Account'. The main area has tabs for 'Test' (selected), 'Files', and 'Chat'. In the 'Files' tab, a file named 'jessica-henwick-the-matrix-resurrections-1639...' is listed, modified on 06/03/2024 at 20:20. To the right, a detailed view of the file is shown, including its URL on arweave.net: <https://arweave.net/gguhTlNgC-oh1mOsDp9R6ce7rpqrta...>. The file info panel shows the following details:

File info	
arweave.net	...
https://arweave.net/gguhTlNgC-oh1mOsDp9R6ce7rpqrta...	
View transaction	
First uploaded	06/03/2024 at 20:20
Last modified	06/03/2024 at 20:20
Version	1
Status	Active
Size	274.71KB
File type	JPEG

The screenshot shows the Arweave Transaction details page for the file. It includes the following information:

Transaction
gguhTlNgC-oh1mOsDp9R6ce7rpqrta...
Status: Pending ~16m
Value: 0 AR
From: DnNMXP8Q1jGKKxe8kvZCx8kjqlSt5aLeHIwXjQr-tk
To: -
Fee: 0 AR

Timestamp: Mar 06 2024 08:21:36 PM **Age**: 12 minutes **Height**: 1,377,979 **Confirmations**: 7
Bundled In: XKv4Vv ... Or8jl48 **Bundle Sig**: arweave **Size**: 268.3 KiB

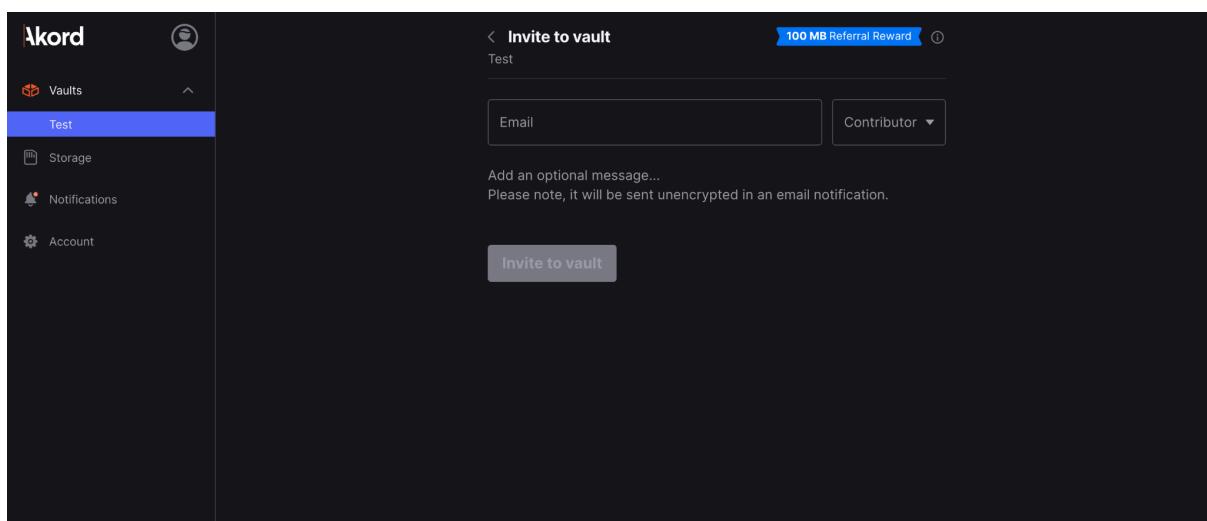
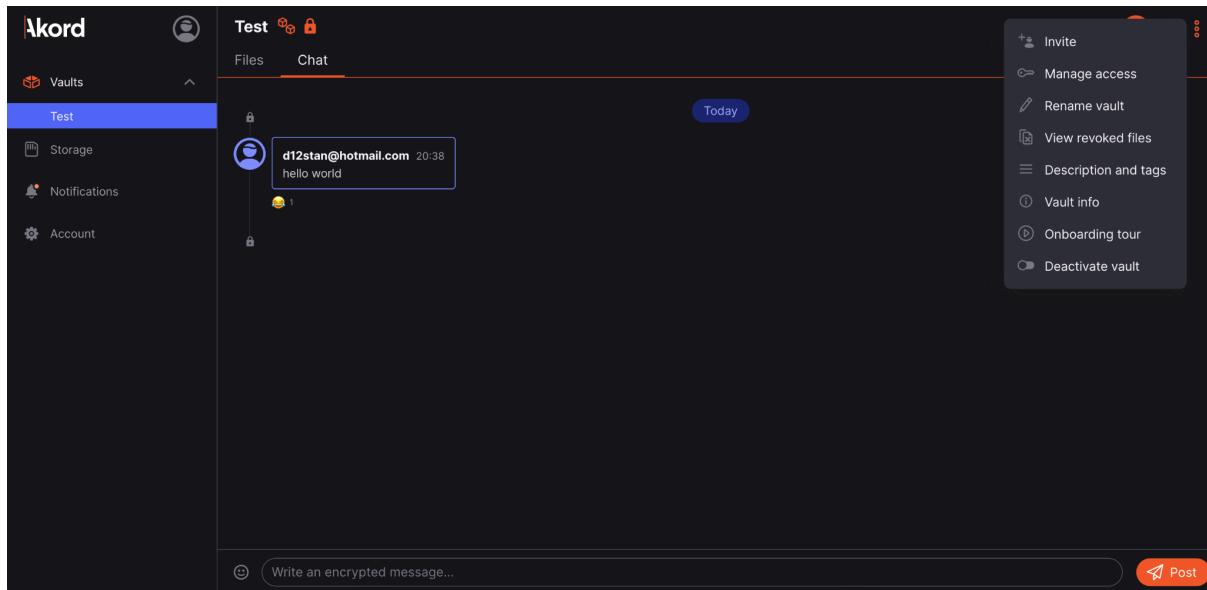
Tags

File-Size	274707
File-Type	image/jpeg
Content-Type	image/jpeg
Timestamp	1709731215623
Data-Type	File
Vault-Id	sVO58uz ... 3QD8z5k
Public-Address	eDDWruocl0+nnLs9CmpxA/Vjiixnl02Yw5spB9rNBCCzLDi0Xx8yBI8xon
Encrypted-Key	eyJjaXB0ZXJ0ZXh0IjoiS3BxUklpYlhEcs3NmNUtllHRHViQll0rzVlbW
Signature	Qi+1dWzZqXoqXlpajvzA8BlWLQ6ArKaPxIE0hCnDwhMA5qtLYLf5Hlk5
File-Hash	EdjMK76En8VZ6g3u9AbsxNv1ShTscYzfWWJ3TAFJt7M=

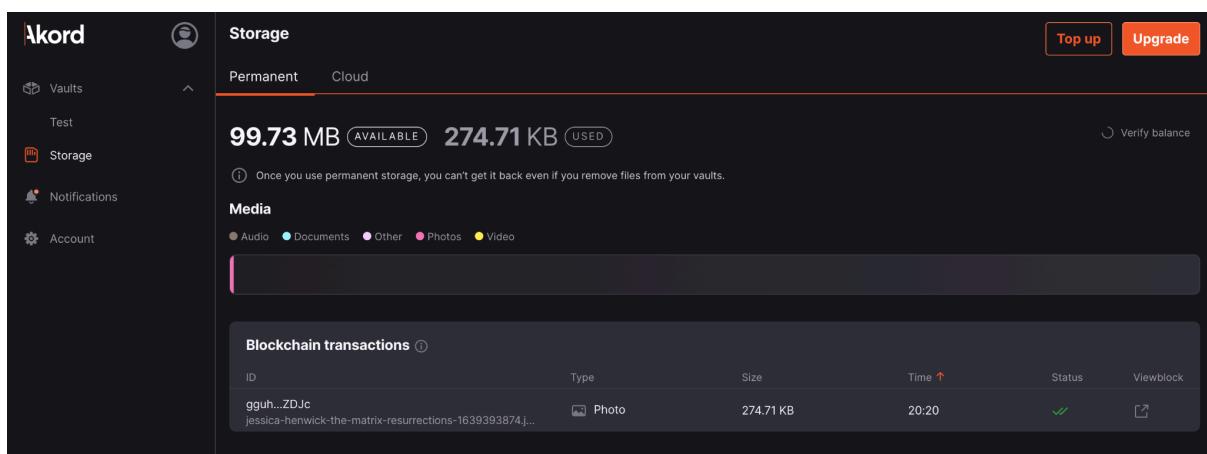
Data

A small thumbnail image of the uploaded JPEG file is shown.

5.1.8 在创建的 Test Vault 发起聊天, 可以邀请其他用户加入群聊



5.1.9 在用户 Storage 界面显示链上 Permanent/Cloud 存储统计数据



5.2.0 在 Block 界面显示 Arweave 链上区块数据: 矿工, 区块打包交易数, 区块大小, 区块生成时间等

Blocks

Height	Time	Transactions	Miner	Size
1,377,993	Mar 06 2024 08:42:37 PM (GMT+7) (4 minutes)	38	D0Rn5Bs ... CkgzMFI	68.75 MiB
1,377,992	Mar 06 2024 08:40:51 PM (GMT+7) (6 minutes)	18	LMjKFPj ... 4aXkzsM	245.25 MiB
1,377,991	Mar 06 2024 08:39:39 PM (GMT+7) (7 minutes)	38	9oza0EN ... wSKYcQM	354.25 MiB
1,377,990	Mar 06 2024 08:37:45 PM (GMT+7) (9 minutes)	36	B8DLQYq ... CYHYy0Q	73.25 MiB
1,377,989	Mar 06 2024 08:35:31 PM (GMT+7) (11 minutes)	0	r0ijW3p ... 7vOaOsI	0 B
1,377,988	Mar 06 2024 08:35:25 PM (GMT+7) (11 minutes)	46	x7tddQI ... WLFWXpE	512.5 MiB
1,377,987	Mar 06 2024 08:33:41 PM (GMT+7) (13 minutes)	58	JT1dBtu ... 9acVNA4	142.5 MiB
1,377,986	Mar 06 2024 08:30:53 PM (GMT+7) (15 minutes)	6	IInPtey ... iB5BEIXI	3.5 MiB
1,377,985	Mar 06 2024 08:30:49 PM (GMT+7) (16 minutes)	13	HOBi0u ... WASL7zQ	7.25 MiB
1,377,984	Mar 06 2024 08:30:11 PM (GMT+7) (16 minutes)	19	5i-eFP5 ... A9uCHTC	58.5 MiB
1,377,983	Mar 06 2024 08:28:55 PM (GMT+7) (17 minutes)	54	mncNqQt ... 6kLHGsg	303.75 MiB
1,377,982	Mar 06 2024 08:26:13 PM (GMT+7) (20 minutes)	12	sKLHodt ... MbKXCoE	49.25 MiB
1,377,981	Mar 06 2024 08:25:43 PM (GMT+7) (21 minutes)	17	QnZvRp ... rKImkls	61 MiB
1,377,980	Mar 06 2024 08:24:57 PM (GMT+7) (21 minutes)	60	kCWZuQ5 ... LCGgPD0	670.5 MiB
1,377,979	Mar 06 2024 08:21:36 PM (GMT+7) (25 minutes)	92	cBOMGLX ... Zpa6-yg	986.25 MiB

5.2.1 在 Block 中显示每笔交易的数据: 交易哈希, From, To, Fee, 交易信息等

Block 1377993

Timestamp	Mar 06 2024 08:42:37	Age	5 minutes	Transactions	38	Mined Time	2 minutes
Miner	D0Rn5Bs ... CkgzMFI	Confirmations	0	Size	68.75 MiB	Tx Reward Pool	0.06 AR
Miner Reward	0.77 AR	Last Retarget	Mar 06 2024 08:37:45				
Transactions							
	Hash	From	Size	Fee	Info		Age
0 ~31m	9MwhyIb ... Za9DPX0	ArDrive Turbo	38.88 MiB	0.03681967 AR	0.97/GiB 15000 Items mint		5 minutes
0 ~31m	FRBLT1 ... orq-B9A	Binance	0 B	0.01800864 AR	16.315 AR → UipYKjc ... 9IHOU6o		5 minutes
0 ~31m	SKoncXF ... LQLUELA	Irys Node 2	7.44 MiB	0.00709012 AR	0.98/GiB 202 Items mint		5 minutes
0 ~31m	ZxL09n0 ... i100JuA	LL9qcK9 ... ACmJquU	1.14 MiB	0.00118291 AR	1.06/GiB image/png		5 minutes
0 ~31m	BwR2VLu ... GXyZFQY	v9dAnet ... XanMG3M	1.06 MiB	0.00118291 AR	1.14/GiB image/png		5 minutes
0 ~31m	UDPurAh ... p8_IBNA	LL9qcK9 ... ACmJquU	852.13 KiB	0.0009469 AR	1.17/GiB image/png		5 minutes
0 ~31m	9z-gu74 ... NWvLVII	LL9qcK9 ... ACmJquU	835.73 KiB	0.0009469 AR	1.19/GiB image/png		5 minutes
0 ~31m	BywC0BD ... yoGakmU	LL9qcK9 ... ACmJquU	878.43 KiB	0.0009469 AR	1.13/GiB image/png		5 minutes
0 ~31m	cvvgy1w ... t05e0IA	LL9qcK9 ... ACmJquU	1,000 KiB	0.0009469 AR	0.99/GiB image/png		5 minutes
0 ~31m	XBizuSv ... OPyp1Cg	LL9qcK9 ... ACmJquU	921.15 KiB	0.0009469 AR	1.08/GiB image/png		5 minutes
0 ~31m	HTNhQDF ... hj6yBQs	LL9qcK9 ... ACmJquU	781.4 KiB	0.0009469 AR	1.27/GiB image/png		5 minutes

Transaction

ZxL09nOqazS5m6xY6TB88gIJvzDorumSLFu9i108JuA

Status Pending ~29m

Value 0 AR

From LL9qcK9cXLSUonC0MR88DL62G2E9b3i_a580ACmJquU

To -

Fee 0.0011829149 AR (\$0.04)

Timestamp Mar 06 2024 08:42:37 PM (Age 7 minutes) **Height** 1,377,993 **Confirmations** 0

Size 1.14 MiB

Tags

- Content-Type image/png
- User-Agent lighthouse
- User-Agent-Version 0.3.8
- Type file
- File-Hash d3f5974618ef8e8fcc87f6f8bd4de877830ef7f29c2ae7cfabd4ef5

Signature

```
HY3GLl05ZIXvo9Tx38kPz6BF2fwt-
DcxkRYK0kdoSInzFGi94vRAQAwFUsva__QJPtriPyCdGBeV0Vat_ZXddh31FvW0wDywZ
ezUJC1fFPsd9mlIi74nfF0y1062v5RIUdKakjXhKfrjdqtERfdz08HHUB29hLB467-c1zjh
038tsQv2j9o1j8JistLRwtwxZMhiK1Cu0RmZgs6DTBQi1UMkh0keq02X0Xg5dJk_w3q0rAv
eAg2rz0G4Ew7/zE2DfqnotEudv6q2QkXKMFTXkqws0b3SGVXXAT5wNbo7G02e9ZUVUv
```

Data

5.2.2 在 Addresses 中显示用户钱包地址, 以及钱包内代币种类、余额、交易信息

Search for an address, tx, block, token, name...

Hash	Balance	Value	Transactions	Paid	Discovery
Binance	13,703,901 AR	\$447,021,251	82,048	205.0401475808 AR	676704
nQKiFE ... bentIJY	6,734,220 AR	\$219,670,256	174	0.0018664229 AR	1016882
dRFuVE- ... 0t5-WXE	4,968,040 AR	\$162,057,465	19	0.0022016886 AR	808612
MBB9dcP ... tlCpvIE	2,506,481 AR	\$81,761,410	18	0.0102096433 AR	656629
pfIZ8vt ... 1fdzkIM	2,072,447 AR	\$67,603,221	18	0.00009075506 AR	817141
z51g48R ... QLy-QCE	1,752,475 AR	\$57,165,735	15	0.0178082035 AR	1029852
4u5gMvl ... EYBrL0E	1,402,001 AR	\$45,733,273	23,281	40.9401418365 AR	576165
py7U20- ... fmdkG78	1,245,411 AR	\$40,625,307	13	0.0000028212 AR	1195614
00rXAiB ... BeMm_E8	1,153,160 AR	\$37,616,079	154	0.0014045857 AR	700007
qE7GpVR ... vd9hmio	1,103,002 AR	\$35,979,925	40	0.0168817175 AR	867678
nXysDSU ... xltE0YQ	1,007,834 AR	\$32,875,545	114	0.0000519495 AR	812111
DlenN74 ... DfxAs14	941,782.29 AR	\$30,720,938	59	0.0835011462 AR	1307309
F4260Mx ... nMjF96g	827,439.05 AR	\$26,991,062	8	0.0000025647 AR	1222525

5.2.3 在 Contract 中显示执行交易的合约代码，以及合约内函数被调用的记录

Search for an address, tx, block, token, name...

PRICE \$32.62 MARKET CAP \$2.15B VOLUME 24H \$264.61M

U ArConnect

Transactions 1,446,158 Holders 2429 Creation KTzTXT_... UwH2Lxw Status Valid

Evaluations 1,377,984 Owner jshaw.ar Type token

Transactions Code

```
1 // src/read/balance.js
2 async function balance(state, action) {
3   const addr = action?.input?.target || action.caller;
4   return {
5     result: {
6       target: addr,
7       ticker: state.ticker,
8       balance: state.balances[addr] || 0
9     }
10   };
11 }
12
13 // src/hyper-either.js
14 var Right = (x) => ({
15   isLeft: false,
16   chain: (f) => f(x),
17   ap: (other) => other.map(x),
18   eq: (other) => other.equals(x)
19});
```

Transactions		1,446,158	Holders		2429	Creation		KTzTXT_...UwH2Lxw	Status	Valid		
Evaluated		1,377,984	Owner		jshaw.ar	Type		token				
Transactions		Code										
	Hash	From	Size	Fee	Info			Block	Age			
①	uX4sPiH ... kU5sLNg	Arseeding Node 1	74.86 KiB	0.00024128 AR	3.38/GiB	3 Items	mint	1,377,995	4 minutes			
①	m1ghsUC ... wgyrXcA	Irys Node 2	8.83 MiB	0.00850756 AR	0.99/GiB	193 Items	mint	1,377,995	4 minutes			
①	FyaDVFl ... mbLW9gQ	ArDrive Turbo	38.58 MiB	0.03658366 AR	0.97/GiB	15000 Items	mint	1,377,995	4 minutes			
①	y04HqCX ... PKGpnf8	Irys Node 1	13.33 KiB	0.00023913 AR	18.01/GiB	1 Item	mint	1,377,995	4 minutes			
①	49E3H8u ... 2N0ErsI	ArDrive Turbo	167.1 MiB	0.15789023 AR	0.97/GiB	960 Items	mint	1,377,995	4 minutes			
①	srVZ4-m ... H0o9JQE	Arseeding Node 1	4.68 KiB	0.00024128 AR	54.08/GiB	1 Item	mint	1,377,995	4 minutes			
①	zT8V4vT ... zwDUYA8	ArDrive Turbo	38.89 MiB	0.03681967 AR	0.97/GiB	15000 Items	mint	1,377,995	4 minutes			
①	NcXoyFn ... yXxdfuw	ArDrive Turbo	38.92 MiB	0.03681967 AR	0.97/GiB	15000 Items	mint	1,377,995	4 minutes			
①	aIZ9-II ... 2aJmV8Y	Irys Node 2	5.44 MiB	0.00520019 AR	0.98/GiB	191 Items	mint	1,377,995	4 minutes			
①	YiEOnSP ... wJAMFn5	Irys Node 1	280.46 MiB	0.2650653 AR	0.97/GiB	Errored Bundle	mint	1,377,995	4 minutes			
①	9-6Pdk ... b4tE6X8	Arseeding Node 1	55.07 KiB	0.00024128 AR	1.52/GiB	1 Item	mint	1,377,995	4 minutes			

Transaction

`y04HQcX5SMPIvHsB1n231MLrdpxE_R85b6PKgnf8`

Status: Pending ~29m

Value: 0 AR

From: Irys Node 1

To: -

Fee: 0.0002391338 AR (\$0.01)

Timestamp: Mar 06 2024 08:52:53 PM (Age 5 minutes)

Height: 1,377,995

Size: 13.33 Kib

Confirmations: 0

Tags:

- App-Name: Bundlr
- Action: Bundle
- Bundle-Format: binary
- Bundle-Version: 2.0.0
- Protocol-name: BAR
- Action: Burn
- App-Name: SmartWeaveAction
- App-Version: 0.3.0
- Input: {"function": "mint"}
- Contract: KTzTXT_..._UwH2Lxw
- Bundlr-App-Name: Warp

Bundle: 1 items

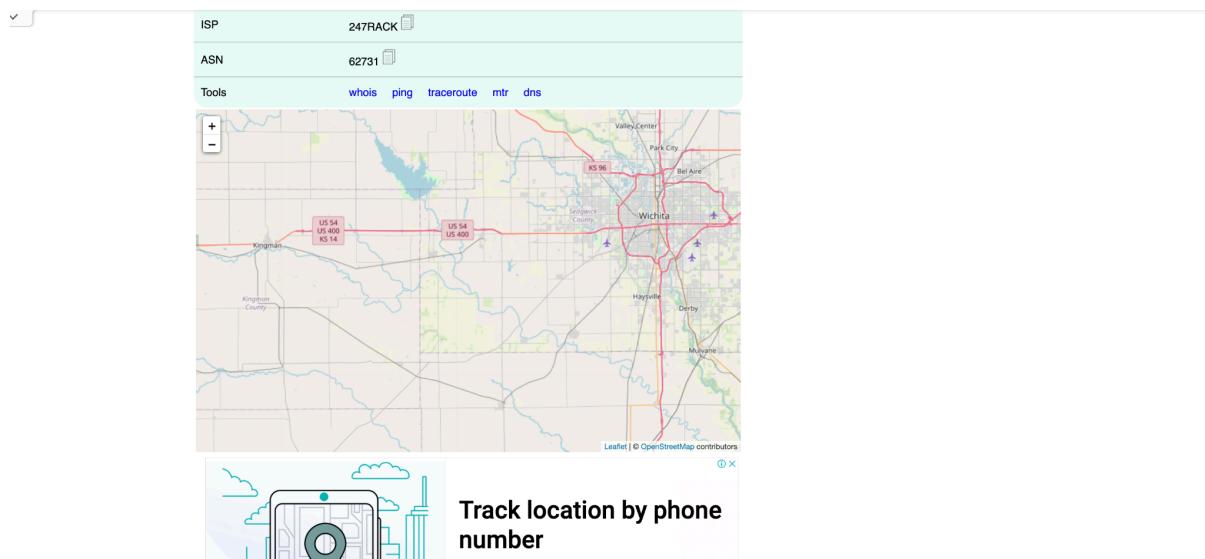
D5kJJWs ... esGMgLG 11.45 KiB

5.2.4 在 Nodes 中显示运行在全球各地的网络节点

Nodes

ID	Delay	Height	Last Seen	Location	ISP	Uptime
206.83.144.16:1984	132ms	1,377,969	about 1 hour	USA New York	247RACK.com	
35.175.1.113:1984	170ms	1,377,969	about 1 hour	USA Ashburn	Amazon.com, Inc.	
35.78.175.38:1984	235ms	1,377,963	about 1 hour	Japan Tokyo	Amazon.com, Inc.	
47.90.203.43:1984	210ms	1,377,969	about 1 hour	USA Charlottesville	Alibaba.com LLC	
162.220.53.21:1984	172ms	1,377,969	about 1 hour	USA Ashburn	247RACK.com	
164.128.161.197:1984	355ms	1,377,969	about 1 hour	Switzerland Muehlethal	Swisscom (Schweiz)	
18.143.34.211:1984	366ms	1,377,969	about 1 hour	Singapore Singapore	Amazon Technolo...	
13.215.88.109:1984	359ms	1,377,969	about 1 hour	Singapore Singapore	Amazon Technolo...	
157.230.102.219:1984	357ms	1,377,969	about 1 hour	Germany Frankfurt am Main	DigitalOcean, L...	
3.34.96.164:1984	242ms	1,377,969	about 1 hour	Korea Seoul	Amazon.com, Inc.	
65.21.201.96:1984	401ms	1,377,969	about 1 hour	Finland Helsinki	Hetzner Online -	
121.133.147.61:1984	273ms	1,377,969	about 1 hour	Korea Yangju	Korea Telecom	
81.7.15.172:1984	363ms	1,377,969	about 1 hour	Germany Jena	EUSERV-SRV	
178.128.80.226:1984	418ms	1,377,969	about 1 hour	Germany Berlin	DigitalOcean, L...	

66 online • 5 behind



5.2.5 收费标准

Akord				Solutions	Platform	Pricing	Developers	About	Log in	Sign up
On all plans	PEBBLE	\$0 USD / MONTH	BOULDER	\$14.2 USD / MONTH	MOUNTAIN	\$99 USD / MONTH				
<ul style="list-style-type: none"> <input checked="" type="checkbox"/> Upgrade anytime <input checked="" type="checkbox"/> Downgrade anytime <input checked="" type="checkbox"/> Cancel anytime 	Start for free	Top up as and when	Buy now	Secure all your files	Buy now	Manage teams and customers				
Storage										
<input type="checkbox"/> Permanent storage	\$12 per GB*	\$9 per GB* <small>SAVE 25%</small>	\$9 per GB* <small>SAVE 25%</small>							
<input type="checkbox"/> Secure Cloud Storage	1 GB free	500 GB ▾	1 TB ▾							
Special Features										
Akord Early Adopter (AEA)	✗	✓	✓							
Token-Gated Access <small>Coming Soon</small>	✗	✓	✓							
Airdrop Access	✗	✓	✓							
Organisation Account	✗	✗	✓							

(更多应用场景可以登陆:<https://arweave.org/>)

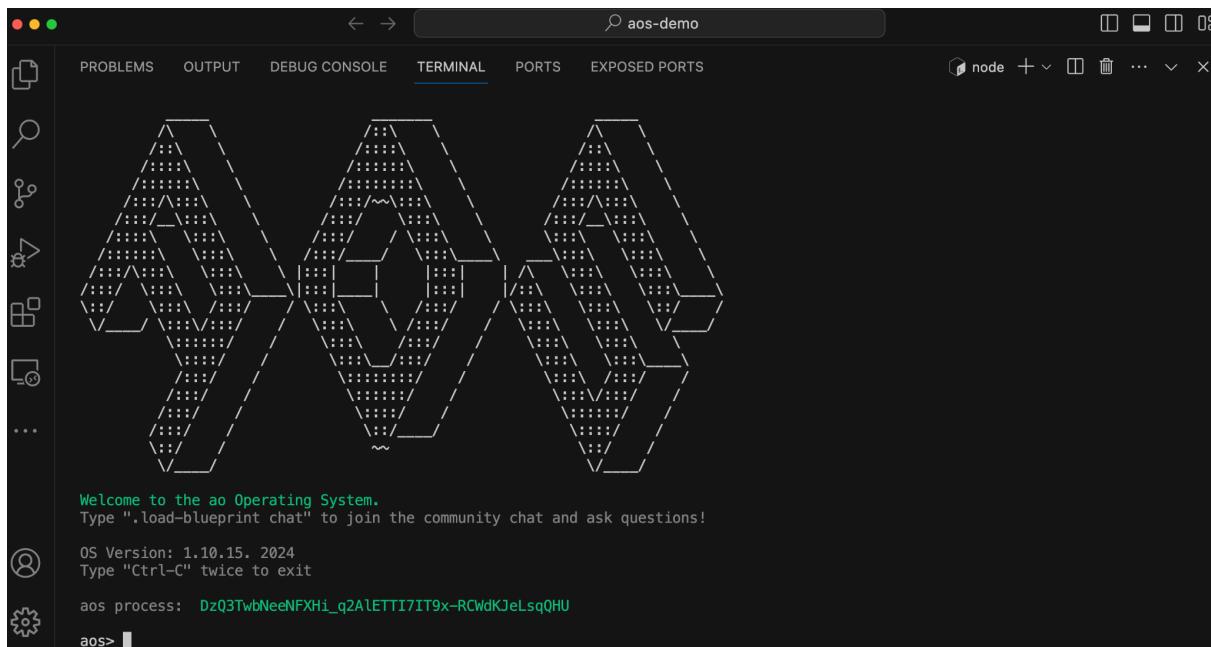
6. AO Cookbook testnet 功能测试

安装 aos 运行 local process

Requirements

- node --version output like: 20.9.0

Install: npm i -g https://get_ao.g8way.io / aos



6.1 消息传递 DEMO

在AO中，每个进程并行运行，创建了一个高度可扩展的环境。进程之间传统的直接函数调用是不可行的，因为每个进程都是独立且异步运行的。

消息传递通过启用异步通信来解决这个问题。进程发送和接收消息，而不是直接相互调用函数。这种方法允许灵活高效的交互，进程可以响应消息，增强系统的可扩展性和响应能力。

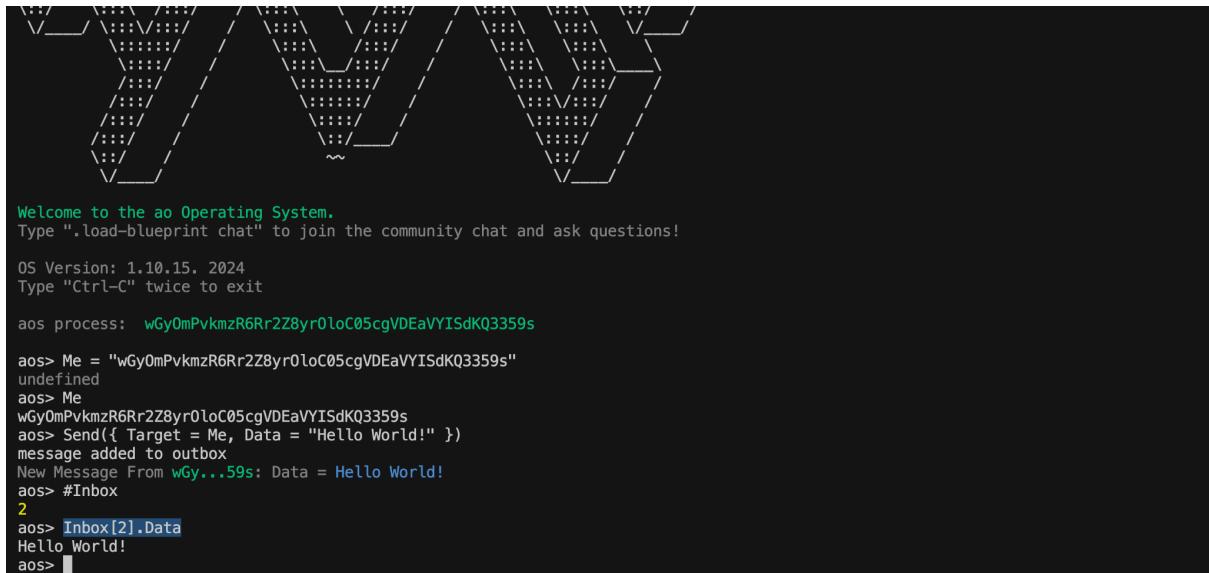
我们将首先探讨 中消息传递的基础知识aos、如何查看收件箱中收到的消息以及如何将消息发送到其他进程。

6.1.1 消息结构: `{ Data = "Hello from Process A!" }`

6.1.2 向自己发送消息: `Send({ Target = Me, Data = "Hello World!" })`

6.1.3 查看收件箱内消息数量: `#Inbox`

6.1.4 索引收件箱内具体消息内容: `Inbox[2].Data`



6.1.5 向Morpheus(AI)发送消息，从Morpheus接收消息

```
aos> Morpheus = "s0QYMwbbTr5MlPwp-KUmbXgCCvfoVjgTOBuUDQJZAIU"  
undefined  
aos> Morpheus  
s0QYMwbbTr5MlPwp-KUmbXgCCvfoVjgTOBuUDQJZAIU  
aos> Send({ Target = Morpheus, Data = "Morpheus?" })  
message added to outbox  
New Message From s0...AIU: Data = I am here. You are f  
aos> Inbox[3].Data  
I am here. You are finally awake. Are you ready to see how far the rabbit hole goes?  
aos> |
```

6.1.6 向 Morpheus 发送一条带有标签Action和值的消息 rabbithole

标签的用途: aos 消息中的标签用于有效地分类、路由和处理消息。它们在消息处理中发挥着至关重要的作用,尤其是在处理多个流程或复杂的工作流程时。

某些进程专门Handlers与具有特定标签的消息进行交互。例如，一个进程可能有一个处理程序，仅与具有特定标签的消息交互。

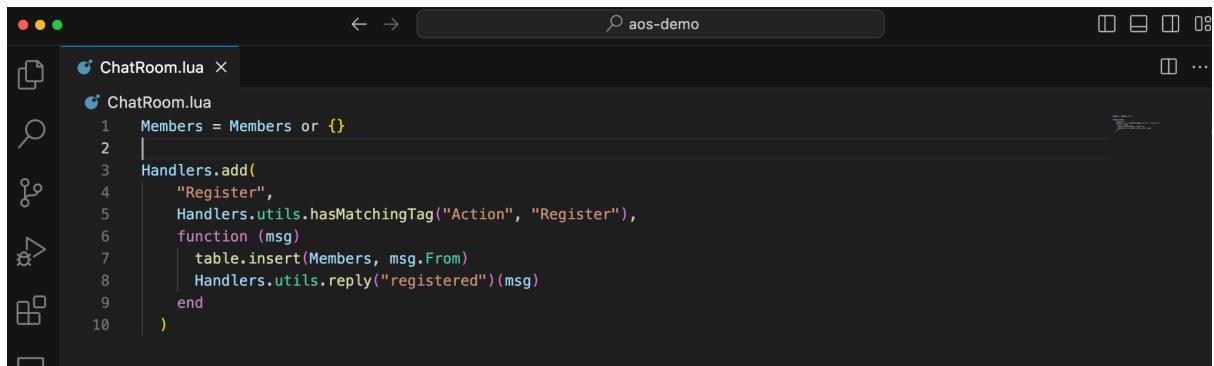
- **一致的标记**: 为您的应用程序开发一致的标记系统, 使消息处理更加可预测。
 - **标签命名**: 为标签选择清晰且具有描述性的名称。这使得一目了然地更容易理解消息的目的和上下文。
 - **标签的安全性**: 标签未加密或隐藏, 避免使用敏感信息作为标签。
 - **工作流管理**: 标签有助于管理工作流, 特别是在消息经过多个阶段或流程的系统中。

```
aos> Send({ Target = Morpheus, Data = "Code: rabbithole", Action = "Unlock" })
message added to outbox
New Message From s0Q...AIU: Data = then let us test you
aos> Inbox[4].Data
then let us test your readiness. create a chatroom and send me an invite. we will continue to see if you are truly ready there.
aos> █
```

6.2 创建聊天室 DEMO

6.2.1 创建成员列表, 将聊天室家在到aos, 并创建聊天室的功能

该处理程序将允许进程通过响应标记来注册到聊天室Action = "Register", registered注册成功后, 将出现一条打印消息确认。



```

ChatRoom.lua
1 Members = Members or {}
2
3 Handlers.add(
4     "Register",
5     Handlers.utils.hasMatchingTag("Action", "Register"),
6     function (msg)
7         table.insert(Members, msg.From)
8         Handlers.utils.reply("registered")(msg)
9     end
10 )

```

6.2.2 测试注册过程, 将自己成功添加到列表中Members



```

aos> .load ChatRoom.lua
Loading... ChatRoom.lua
undefined
aos> Handlers.list
{
  {
    handle = function: 0x5bc3d8,
    name = "_eval",
    pattern = function: 0x576820
  },
  {
    handle = function: 0x5bc418,
    name = "_default",
    pattern = function: 0x547c50
  },
  {
    handle = function: 0x5b3578,
    name = "Register",
    pattern = function: 0x5b3540
  }
}
aos> Send({ Target = ao.id, Action = "Register" })
message added to outbox
New Message From wGy...59s: Data = registered
aos> Members
{ "wGy0mPvkMzR6Rr2Z8yr0loC05cgVDEaVYISdkQ3359s" }
aos>

```

- _eval和_default, 这是预定义的处理程序, handler是处理特定动作或命令的函数或方法。
- Register, 这是一个用户定义的处理程序, 用于处理注册动作。当接收到一个动作名为"Register"的消息时, 会触发与Register处理程序相关联的handle函数。

6.2.3 添加广播处理程序, 通过向聊天室发送消息来测试广播处理程序

```

12     Handlers.add(
13         "Broadcast",
14         Handlers.utils.hasMatchingTag("Action", "Broadcast"),
15         function (msg)
16             for _, recipient in ipairs(Members) do
17                 ao.send({Target = recipient, Data = msg.Data})
18             end
19             Handlers.utils.reply("Broadcasted.")(msg)
20         end
21     )

```

```

PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS    EXPOSED PORTS
aos> .load ChatRoom.lua
Loading... ChatRoom.lua
undefined
aos> Handlers.list
{
  {
    handle = function: 0x5ded88,
    name = "_eval",
    pattern = function: 0x576820
  },
  {
    handle = function: 0x5dedc8,
    name = "default",
    pattern = function: 0x547c50
  },
  {
    handle = function: 0x5be918,
    name = "Register",
    pattern = function: 0x5dcba8
  },
  {
    handle = function: 0x5be600,
    name = "Broadcast",
    pattern = function: 0x5dcc08
  }
}
aos> Send({Target = ao.id, Action = "Broadcast", Data = "Broadcasting My 1st Message"})
message added to outbox
New Message From wGy...59s: Data = Broadcasted.
New Message From wGy...59s: Data = Broadcasting My 1st
aos>

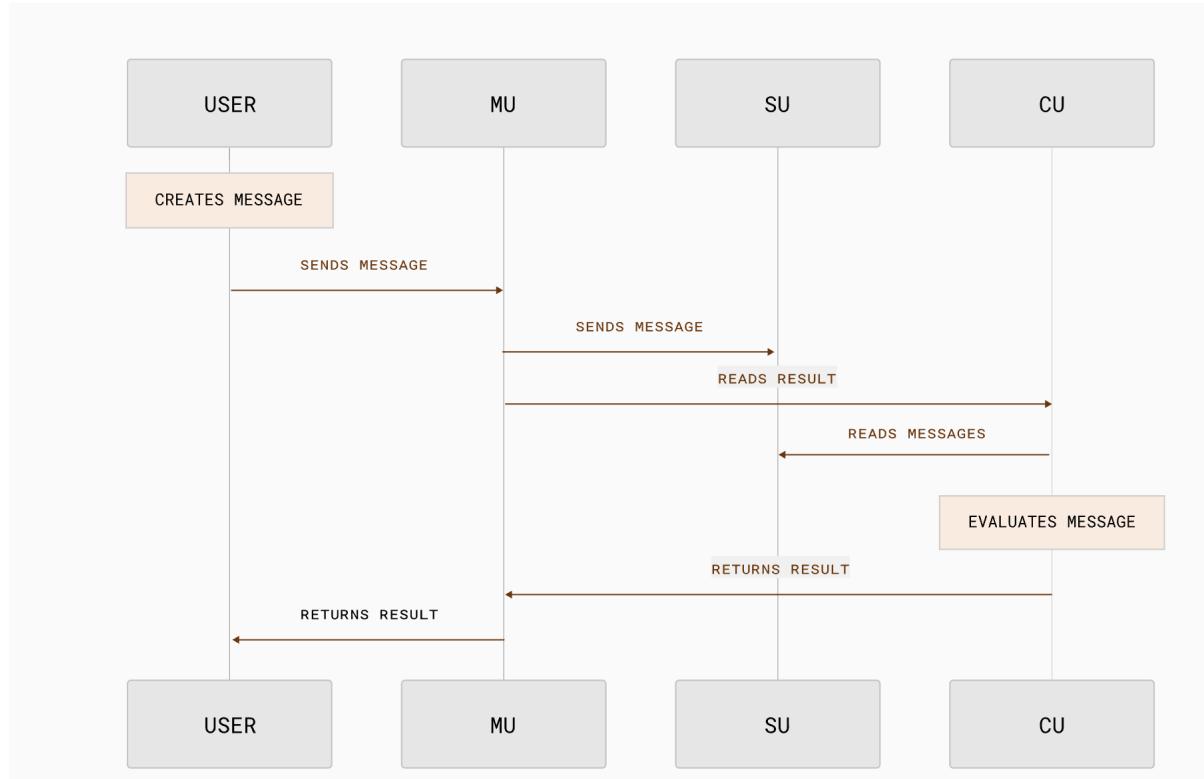
```

6.2.4 邀请更多人加入聊天室

```

aos> Send({ Target = Morpheus, Action = "Join" })
message added to outbox
New Message From wGy...59s: Data = Good. Very Good. No
aos> Members
{ "wGyOmPvkxzR6Rr2Z8yr0loC05cgVDEaVYISdKQ3359s", "s0QYMwbbTr5MlPwp-KUmbXgCCvfoVjgTOBuUDQJZAIU" }
aos> #Inbox
8
aos> Inbox[8].Data
Good. Very Good. Now, let me introduce you to Trinity. Here is her process ID: K3YDqxUlQvzonUZ0it0PzAR-rPWvo2Clf9w_NRBbfds. Once
you have saved her process ID as you did mine, invite her to the chatroom, as well.
aos> Send({ Target = "K3YDqxUlQvzonUZ0it0PzAR-rPWvo2Clf9w_NRBbfds", Action = "Join" })
message added to outbox
New Message From wGy...59s: Data = So this is the one M
aos> Members
{ "wGyOmPvkxzR6Rr2Z8yr0loC05cgVDEaVYISdKQ3359s", "s0QYMwbbTr5MlPwp-KUmbXgCCvfoVjgTOBuUDQJZAIU", "K3YDqxUlQvzonUZ0it0PzAR-rPWvo2Clf9w_NRBbfds" }
aos>

```



- 通过 **Messenger Unit (MU)** 提交消息：用户通过向 MU 提交一个消息来触发一个操作。在提供的代码示例中，这通过执行 `.load ChatRoom.lua` 和 `Send` 函数来完成，这些函数实际上是在与 MU 交互，用于加载聊天室的脚本并发送一个注册(`Register`)消息。
- Scheduler Unit (SU)** 序列化消息：提交到 MU 的消息随后被转发到 SU。SU 负责给这些消息排序和分配时间戳，确保它们在整个系统中有序传播。
- Compute Unit (CU)** 执行消息：一旦消息被 SU 序列化，它们将被传送到 CU 进行处理。CU 根据提供的 Lua 脚本(例如 `ChatRoom.lua`)执行相应的命令，如注册用户到聊天室或处理其他聊天室逻辑。
- 邀请人员进入聊天室：通过发送特定的命令或消息(例如，通过 MU 发送带有特定动作或命令的消息)，可以邀请其他用户加入聊天室。这可能涉及到发送包含邀请动作和目标用户标识的消息。

6.3 创建和发送代币 DEMO

6.3.1 手动方法ao 构建代币

```
Token.lua
1 local json = require('json')
2
3 if not Balances then Balances = { [ao.id] = 1000000000000000 } end
4
5 if Name ~= 'My Coin' then Name = 'My Coin' end
6
7 if Ticker ~= 'COIN' then Ticker = 'COIN' end
8
9 if Denomination ~= 10 then Denomination = 10 end
10
11 if not Logo then Logo = 'optional arweave TXID of logo image' end
12
```

6.3.2 传入消息处理程序和信息、代币余额处理程序

```
Handlers.add('info', Handlers.utils.hasMatchingTag('Action', 'Info'), function(msg)
  ao.send({
    Target = msg.From, Tags = { Name = Name, Ticker = Ticker, Logo = Logo, Denomination = tostring(Denomination) } })
end)

Handlers.add('balance', Handlers.utils.hasMatchingTag('Action', 'Balance'), function(msg)
  local bal = '0'

  -- If not Target is provided, then return the Senders balance
  if (msg.Tags.Target and Balances[msg.Tags.Target]) then
    bal = tostring(Balances[msg.Tags.Target])
  elseif Balances[msg.From] then
    bal = tostring(Balances[msg.From])
  end

  ao.send({
    Target = msg.From,
    Tags = { Target = msg.From, Balance = bal, Ticker = Ticker, Data = json.encode tonumber(bal) } })
end)

Handlers.add('balances', Handlers.utils.hasMatchingTag('Action', 'Balances'),
  function(msg) ao.send({ Target = msg.From, Data = json.encode(Balances) }) end)
```

6.3.3 transfer 处理程序

```
Handlers.add('transfer', Handlers.utils.hasMatchingTag('Action', 'Transfer'), function(msg)
  local qty = tonumber(msg.Tags.Quantity)

  if Balances[msg.From] >= qty then
    Balances[msg.From] = Balances[msg.From] - qty
    Balances[msg.Tags.Recipient] = Balances[msg.Tags.Recipient] + qty

    --[[ Only Send the notifications to the Sender and Recipient
      if the Cast tag is not set on the Transfer message
    ]]--

    if not msg.Tags.Cast then
      -- Send Debit-Notice to the Sender
      ao.send({
        Target = msg.From,
        Tags = { Action = 'Debit-Notice', Recipient = msg.Tags.Recipient, Quantity = tostring(qty) } })
      -- Send Credit-Notice to the Recipient
      ao.send({
        Target = msg.Tags.Recipient,
        Tags = { Action = 'Credit-Notice', Sender = msg.From, Quantity = tostring(qty) } })
    end
  else
    ao.send({
      Target = msg.Tags.From,
      Tags = { Action = 'Transfer-Error', ['Message-Id'] = msg.Id, Error = 'Insufficient Balance!' } })
  end
end)
```

6.3.4 mint 处理程序

```

76 Handlers.add('mint', Handlers.utils.hasMatchingTag('Action', 'Mint'), function(msg, env)
77     assert(type(msg.Tags.Quantity) == 'string', 'Quantity is required!')
78
79     if msg.From == env.Process.Id then
80         -- Add tokens to the token pool, according to Quantity
81         local qty = tonumber(msg.Tags.Quantity)
82         Balances[env.Process.Id] = Balances[env.Process.Id] + qty
83     else
84         ao.send({
85             Target = msg.Tags.From,
86             Tags = {
87                 Action = 'Mint-Error',
88                 ['Message-Id'] = msg.Id,
89                 Error = 'Only the Process Owner can mint new ' .. Ticker .. ' tokens!'
90             }
91         })
92     end
93 end)

```

6.3.5 加载Token.lua文件, 测试代码



```

Welcome to the ao Operating System.
Type ".load-blueprint chat" to join the community chat and ask questions!
OS Version: 1.10.15. 2024
Type "Ctrl-C" twice to exit

aos process: DzQ3TwbNeeNFXHi_q2A1ETTI7IT9x-RCWdKJeLsqQHU

aos> .load Token.lua
Loading... Token.lua
undefined
aos> Send({ Target = ao.id, Action = "Info" })
message added to outbox
New Message From DzQ...QHU: Data = 2672
aos> #Inbox
2
aos> Inbox[2].Data
2672

```

6.3.6 transfer 和检查余额

```

aos> Send({ Target = ao.id, Tags = { Action = "Transfer", Recipient = 'another wallet or processid', Quantity = '10000' }})
message added to outbox
New Message From DzQ...QHU: Action = Debit-Notice

```

```

aos> Send({ Target = ao.id, Tags = { Action = "Balances" }})
message added to outbox
New Message From DzQ...QHU: Data = {"DzQ3TwbNeeNFXHi_q2
aos> Inbox[#Inbox].Data
{"DzQ3TwbNeeNFXHi_q2A1ETTI7IT9x-RCWdKJeLsqQHU":99999999990000,"another wallet or processid":10000}
aos>

```

6.3.7 mint 代币

```

aos> Send({ Target = ao.id, Tags = { Action = "Mint", Quantity = '1000' }})
message added to outbox
aos> Send({ Target = ao.id, Tags = { Action = "Balances" }})
message added to outbox
New Message From DzQ...QHU: Data = {"DzQ3TwbNeeNFXHi_q2
aos> Inbox[#Inbox].Data
{"DzQ3TwbNeeNFXHi_q2A1ETTI7IT9x-RCWdKJeLsqQHU":99999999991000,"another wallet or processid":10000}
aos>

```

6.4 对聊天室进行令牌门控 DEMO

6.4.1 现在聊天室已进行令牌控制，让我们通过向聊天室发送消息来测试它

6.5 机器人和游戏 DEMO

6.5.1 注册游戏 -> 获取游戏代币 -> 支付入场费 -> 通过命令移动角色 -> 出击

```

aos> Game = "3HSmHQ-1MaC0L0Ktq5GdgbVOX06mWip40uASAG13XK"
undefined
aos> Send({ Target = Game, Action = "Register" })
message added to outbox
aos> Send({ Target = Game, Action = "RequestTokens" })
message added to outbox
New Message From 3HS...3Xk: Action = Announcement
New Message From 3HS...3Xk: Action = Registered
New Message From 3HS...3Xk: Action = Credit-Notice
aos> Send({ Target = Game, Action = "Transfer", Recipient = Game, Quantity = "1000" })
message added to outbox
New Message From 3HS...3Xk: Action = Debit-Notice
New Message From 3HS...3Xk: Action = Announcement
New Message From 3HS...3Xk: Action = Payment-Received
aos> Send({ Target = Game, Action = "PlayerMove", Player = ao.id, Direction = "DownRight" })
message added to outbox
aos> Up = {x = 0, y = -1},
      (computing _4oPfEW_4oPfEW_klkwD2Bb8XmKyfUpsqo state transformations) Down = {x = 0, y = 1},
Left = {x = -1, y = 0},
Right = {x = 1, y = 0},
UpRight = {x = 1, y = -1},
UpLeft = {x = -1, y = -1},
DownLeft = {x = -1, y = 1},
DownRight = {x = 1, y = 1},
[string "ao$"]::: unexpected symbol near <eof>
aos> DownLeft = {x = -1, y = 1} |Send{ Target = Game, Action = "PlayerAttack", Player = ao.id, AttackEnergy = "energy_integer" }
undefined
aos> Send({ Target = Game, Action = "PlayerAttack", Player = ao.id, AttackEnergy = "energy_integer" })
message added to outbox
aos> #Inbox
1#
aos> Inbox[13].Data
DzQ3TbNeNfXH1_g2aLETTI7IT9x-RCwdkJelsgqHU is ready to play!
aos> Down = {x = 0, y = 1}
undefined
aos> {x = 1, y = 0}
{
  x = 1,
  y = 0
}
aos> {x = -1, y = 0}
{
  x = -1,
  y = 0
}
aos> Send({ Target = Game, Action = "PlayerAttack", Player = ao.id, AttackEnergy = "energy_integer" })
message added to outbox
aos> #Inbox
1#
aos> Inbox[13].Data
DzQ3TbNeNfXH1_g2aLETTI7IT9x-RCwdkJelsgqHU is ready to play!

```

7. 总结

根据我们的深入分析, AO系统代表了区块链技术的一个重要进步, 特别是在去中心化计算平台的领域。它不仅承诺通过其独特的共识机制和高效的资源利用来解决现有区块链系统面临的可扩展性和计算效率问题, 而且还为开发者提供了更灵活和广泛的应用场景。以下是我们总结的关键点的汇总, 形成对AO系统及其在去中心化计算世界中作用的结论:

高效的共识机制

AO通过允许少数节点执行计算任务并由其他节点验证这些结果来达成共识, 大幅减少了整个网络的资源消耗。这种机制在维持去中心化的同时, 提高了整个系统的计算效率和响应速度, 是对传统共识机制的重要补充和优化。

增强的可扩展性

与传统区块链平台如以太坊相比, AO通过其模块化架构和灵活的计算模型, 支持更大规模的数据处理和更复杂的应用场景。这种设计不仅为数据密集型任务提供了新的解决方案, 还大大扩展了区块链技术的应用范围。

去中心化与安全性的平衡

尽管AO采用了与传统区块链不同的共识机制, 但它仍然保持了系统的去中心化特性, 并通过分布式验证机制增强了计算结果的可信度。这表明, AO成功地在提高效率和保持去中心化安全性之间找到了一个平衡点。

开放和模块化的设计

AO的开放数据协议和模块化架构为开发者提供了巨大的灵活性, 允许他们根据自己的需求定制和扩展系统。这种设计理念促进了创新和多样性, 有助于建立一个更加活跃和多元化的去中心化应用生态系统。

总之, AO系统代表了区块链技术在去中心化计算方面的一次重大进步, 它通过其独特的共识机制、增强的可扩展性、以及模块化的设计, 为解决现有系统的局限性和拓展区块链技术的应用领域提供了新的可能性。随着AO系统和相关应用的不断发展和完善, 我们期待看到更多基于AO的创新解决方案, 为去中心化的世界带来更多价值和机会。