尤特: 一种基于指令签名的可验证计算网络

utnet.org

Utnet research@utnet.org

【摘　要】 本文将详细讨论尤特（Utility Network）这一全新的区块链结构，它基于可验证计算指令构建共识。我们将从当前并行计算架构出发，探讨并行计算的规模化，以及其在点对点网络中的实现困难。然后本文将提出全新的基于计算指令的共识 (Proof Of Computing Integrity, 下文简称 POCI)，并分析 POCI 与计算力证明 (Computing Avaliability Over Time, 下文简称 CAT) 的各个方面，包括指令集加密、可验证指令执行、可验证隔离执行环境等。本文也提出了与 POCI 共识匹配的区块链与数据结构、代币循环与计算生态等概念。紧接下节将深入讨论设计执行虚拟机，包括虚拟化和隔离执行环境、环境证明等技术，并加入与 Solidity VM、Kubernetes、WASM 和 PyTorch 的集成。第六章展示基于尤特区块链的人工智能算力网络，包括计算和推理开发、跨域数据调度，并展开性能与可用性等问题。紧接着文章将从共识速度，链上爆块状态，k8s 技术等方面展开描述尤特网络的功能和在各方面的优势。最后第八章中将讨论尤特的未来可能性，包括技术创新、应用场景扩展和市场潜力等。我们将分析尤特如何适应不断变化的市场需求，以及其在未来 Layer-1 公链生态中的潜在角色，最后汇总分析我们将尤特的关键特性和优势，以及其在解决当前大规模计算、区块链和 P2P 网络中的问题方面所取得的成就。总结将重申尤特作为一种创新区块链网络和计算共识技术的重要性，以及其对整个计算和金融生态的影响。

【关键词】 PoCI **可信算力证明**，CAT，**指令集加密**，**跨域数据调度**，**部署大模型**

1　简介

尤特（Utility Network）是一项全新的基于 TPU(Tensor Processing Units) 芯片算力组成的分布式区块链网络技术，将点对点支付和针对并行计算的可验证去中心化计算结合，为区块链网络生态和计算市场带来革命性的变革。尤特采用了基于指令签名的可验证计算能力的共识机制，实现安全、透明和高效的交易处理。同时尤特通过引入执行虚拟机，Kubernetes、WASM 和 PyTorch 集成以及跨域数据调度等技术，为分布式计算提供更强大的软硬件支持。

尤特网络基于芯片可验证计算构建共识，将计算力证明与区块链加密技术相结合，进一步提升了系统的安全性和透明度。借助全新的计算共识和计算力证明，尤特实现了高效的性能和丰富灵魂的可用性，同时保证了分布式挖矿、去中心化金融 (Defi)、算力调度、大数据处理和人工智能等领域的需求得以满足。

尤特网络为开发者提供了更为灵活的编程环境，通过集成 Solidity VM、Kubernetes、WASM 和 PyTorch 等技术，推动了跨领域的数据调度与合作和人工智能的开发。我们团队将致力于将尤特网络打造成为一个具有广泛应用前景和市场潜力的区块链网络，为整个计算和金融生态带来深远的影响 [1-4]。

1.1　当前的大规模计算优劣

作为技术创始团队，我们始终关注大规模计算的发展趋势以及其在现实世界中的应用。当前，大规模计算主要由两种架构支撑：Intel（英特尔）的通用计算和 NVIDIA（英伟达）的异构并行计算架构 [9, 10]。这两种架构各自具有优劣，以下我们将对它们进行详细分析。

1.1.1　Intel（英特尔）的通用计算

Intel 的通用计算主要采用多核 CPU 架构 [9]，强调单核处理器的性能，同时实现多任务并行处理。通用计算具有较强的灵活性，可以适用于各种计算需求，尤其是那些需要高度串行化处理的任务。然而，这种架构的缺点在于其能效比较低，难以应对大规模并行计算场景，如图形渲染、深度学习和科学计算等。

1.1.2 NVIDIA（英伟达）的异构并行计算架构

NVIDIA 的异构并行计算架构则采用 GPU（图形处理器）作为计算单元 [11]，具有大量的并行处理核心。这种架构在处理大规模并行任务时具有明显优势，例如图像处理、机器学习和大数据分析等。相较于通用计算，异构并行计算在能效和性能方面更具优势。但它也存在局限性，对于高度串行化的任务，其性能表现不如通用计算。

1.1.3 异构并行架构的优势

通过对比分析，异构并行计算架构更具潜力。在越来越多的场景中，大规模并行计算需求不断增加，特别是在人工智能、大数据处理和科学计算等领域。异构并行计算架构能够有效提高计算性能和能效，满足这些领域的需求。

然而，大规模计算设备的铺设和组网服务能力仍然存在挑战。首先，大规模计算设备的投资成本较高，尤其是在部署分布式计算系统时。此外，组网服务能力方面，大规模计算网络需要处理庞大的数据传输、协调和安全等问题，这些问题都需要在网络构建中加以解决。

综上，异构并行计算架构是大规模计算的发展方向，但需克服投资成本和组网服务能力的挑战。尤特（Utility Net）正是为解决这些问题而设计的一种区块链网络，它利用区块链技术和计算共识机制，实现了分布式计算资源的高效利用和协调。通过引入可验证计算、执行虚拟机和跨域数据调度等技术，尤特网络为大规模计算提供了强大的支持。

尤特的去中心化特性有助于降低大规模计算设备的投资成本，通过激励机制和租赁市场，让计算资源的提供者和需求者实现更有效的资源共享。此外，尤特网络采用先进的 P2P 技术和计算力证明机制，实现了高性能、安全和可靠的组网服务能力。尤特作为一种创新区块链网络，有着广泛的应用前景和市场潜力。通过解决大规模计算设备铺设的资金问题和组网服务能力问题，尤特有很大的潜力成为支撑未来大规模计算发展的关键基石。

1.2 并行计算的规模化

并行计算的规模化是未来计算发展的趋势，它将带来更高效、经济和灵活的计算资源利用。Filecoin 作为一个去中心化的存储公链，通过代币机制将全球范围内的存储资源有效整合，实现了超过 20EB 的存储规模，这是任何一个单一数据中心难以实现的。Filecoin 的成功为分布式计算启示着我们通过区块链网络和代币经济可以实现大规模并行计算的整合与优化。[4, 5]

通过区块链网络，尤特（Utility Network）有望实现超大规模的芯片部署，解决算力问题。尤特采用计算力证明和计算共识机制，将全球范围内的计算资源进行整合和协调，实现了更高效的资源利用。同时，尤特的代币经济循环模型和应用开发为计算资源的提供者和需求者创造了新的价值和激励，进而推动了并行计算规模化的发展。

在人工智能领域，特别是 SoA(State of Art) 通用的 Transformer 模型的大规模应用层面，尤特网络将为智能上限带来了无限机遇。Transformer 模型在自然语言处理、计算机视觉和强化学习 [23] 等多个领域取得了显著的突破。然而，这些模型对计算资源的需求极高，尤其是在 AI 的训练和推理过程中要求尤为苛刻。通过尤特网络实现的大规模并行计算，可以为这些模型提供更强大的计算支持，从而推动人工智能技术的发展和创新。

尤特能将区块链网络和代币经济应用于并行计算的规模化，为未来人工智能计算带来了巨大的潜力。通过有效整合全球计算资源，该网络体系有望推动计算领域的创新，为人工智能技术发展开辟新的道路和创造更多机遇。

1.3 区块链的落地困难

区块链作为一种具有革命性的技术，其去中心化、透明度高和安全性等特点为各行各业的应用带来了巨大潜力。然而，在实际应用中，区块链面临着诸多落地困难。以下将从资源效率和服务场景的角度，阐述几个区块链落地的挑战。

首先，密码学并不能完美解决很多问题。尽管密码学为区块链提供了安全性保障 [6-8]，但在实际应用中，它会引入额外的计算和存储成本。例如，为了验证计算，需要进行大量无效计算以完成工作量证明（Proof of Work，PoW）。同样，为了验证存储，需要采用类似 Filecoin 的空间证明技术，填满存储空间 [1]。这些验证过程中消耗的资源往往远远超过实际需求，导致资源的利用效率严重低下。

其次，区块链技术在满足共识的同时，往往会浪费大量资源。例如，比特币和以太坊等采用 PoW 机制的区块链网络中，矿工需要进行大量的哈希计算，才能找到满足共识条件的区块。这些计算过程中消耗的电力和计算资源无法为其他有价值的计算任务所用，导致资源的浪费。

最后，实际应用场景对资源的高效服务需求与区块链技术之间存在矛盾。在许多场景中，如金融、物联网和供应链管理等，对数据处理速度、实时性和资源利用率有着严格的要求。然而，区块链技术目前在性能、扩展性和资源效率方面仍存在局限，难以满足这些应用场景的需求。

综上所述，区块链落地困难主要表现在资源效率和实际应用场景的需求之间的矛盾。为了解决这些问题，我们需要不断探索和创新，寻找更高效、可扩展且资源利用率更高的区块链技术方案。尤特网络（Utility Network）正是为解决这些棘手的问题而诞生的一种新型区块链网络，通过引入高效计算共识、区块链数据结构、代币循环与计算生态模型分析、隔离执行环境与执行虚拟机、不同语言环境的集成以及跨域数据调度等技术，实现了分布式计算资源的高效利用和协调，为区块链技术在实际应用场景提供了无限的可能。

1.4 P2P 网络桥接计算网络

P2P(point-To-point) 网络与计算网络的紧密结合，能够为区块链技术带来更高的性能、扩展性和资源利用率。本节将从专业的角度阐述基于计算共识和 P2P 计算结合的过程，以及指令和计算验证的实现。

计算共识机制是 P2P 网络与计算网络桥接的核心。通过引入计算共识，可以将全球范围内的计算资源进行整合和协调，实现更高效的资源利用。计算共识机制与传统的工作量证明（Proof of Work, PoW）和权益证明（Proof of State, PoS）机制相比，能够更好地满足实际应用场景的需求，同时降低资源浪费。

P2P 计算可以为区块链网络提供更强大的计算能力 [3]。基于 P2P 计算的区块链网络，可以将节点间的计算任务进行分布式处理，从而提高计算效率和网络性能。此外，P2P 计算还能够实现计算资源的动态分配，以适应不同应用场景的需求。

在指令和计算验证方面，基于计算共识和 P2P 计算结合的区块链网络可以采用可验证指令执行和隔离执行环境等技术。可验证指令执行技术能够在不泄露计算内容的情况下，验证计算结果的正确性。隔离执行环境（例如执行虚拟机）则能够确保计算过程的安全性和可靠性。通过这些技术，区块链网络可以实现对指令和计算的有效验证，进一步提高计算网络的性能和资源利用率 [13, 16]。这些技术将在下面章节详细展开。

基于计算共识和 P2P 计算结合的区块链网络能够实现 P2P 网络与计算网络的有效桥接。通过引入计算共识机制、P2P 计算技术以及指令和计算验证技术，该网络可以为实际应用场景提供更高的性能、扩展性和资源利用率，从而为区块链技术的发展创造新的机遇。

2 计算共识与计算力证明

计算共识是区块链技术中的一种重要机制，用于解决分布式环境下的数据一致性和信任问题。工作量证明（Proof of Work）是其中一种常用的共识算法，其基本原理是通过消耗大量的计算资源来证明自己对网络的贡献，从而获得记账权和奖励。

工作量证明在保证区块链网络安全性方面发挥了重要作用，但也存在一些问题，如消耗大量的能源和计算资源、容易引发算力竞赛和中心化等。为了解决这些问题，近年来出现了一些新的共识算法，如权益证明（Proof of Stake）、权威证明（Proof of Authority）等 [4,15,18]。而在本节将介绍我们全新的计算力证明机制：可信算力证明 (Proof of Computation Integrity, PoCI)。

不同的共识算法适用于不同的场景和应用，需要根据实际情况进行选择和使用。计算共识和计算力证明是区块链技术中的核心机制之一，对于保证区块链的可靠性和安全性具有重要意义。

2.1 指令集加密

指令集加密（Instruction Set Encryption，ISE）是一种将计算机指令集进行加密以保护程序代码和数据的技术。其主要目的是提高系统的安全性和抵抗代码篡改、代码注入等攻击，作为芯片指令的高保密特征，保障芯片拥有者或矿工绝对安全

的密码学安全性，算能芯片同样配备了基础的指令集加密技术。在使用指令集加密的系统中，处理器会使用一个密钥对 CPU 指令进行加密和解密。加密后的指令集存储与内存或磁盘中，而解密器负责将加密后的指令解密，并将其传递给 CPU 执行。这里指的注意的是，只能被授权的解密器解密执行，非授权解密器无法执行加密指令，从而保护程序和数据的安全性和完整性。指令集加密通常应用于高安全性要求的系统，如金融交易、军事设备、智能卡等领域。与传统软件加密相比，指令集加密的优势在于其能够保护代码和数据在运行时的安全性，从而更加难以被攻击者破解。在我们未来提供的尤特挖矿或 AI 训练服务中，指令集加密同样发挥重要的数据和核心数字资产等的保护作用。

2.2 可验证指令执行

可验证指令是一种特殊类型的计算机指令，其目的和上节提到的指令集加密类似，是一种保障计算机程序安全性的重要技术，在高安全性要求的系统中有着广泛的应用。而区别于指令集加密，可验证指令执行更注重程序运行的安全性和正确性，使计算机程序的执行过程可以被验证，并且可以确保程序的结果是正确的。可验证指令通常用于高安全性要求的系统中，如智能卡、加密通讯、金融交易等领域。在芯片指令中，有诸如以下基本的指令集：

1. SGX 指令：Intel SGX 是一种安全增强技术，可用于保护机密信息和代码。SGX 指令集包括 enclave 创建、enclave 进入、enclave 退出等指令，用于确保计算过程的机密性和安全性。

2. ARM TrustZone 指令：ARM TrustZone 是一种安全技术，可用于保护处理器和系统的安全性。TrustZone 指令集包括 Secure Monitor 调用、TrustZone 寄存器等指令，用于确保计算过程的安全性和完整性。

3. MIPS TrustZone 指令：MIPS TrustZone 是一种安全技术，可用于保护系统的安全性和完整性。MIPS TrustZone 指令集包括 Secure Monitor 调用、Secure Monitor 返回等指令，用于确保计算过程的机密性和安全性。

4. RISC-V PMP 指令：RISC-V PMP（Physical Memory Protection）是一种内存保护技术，可用于保护系统的安全性和完整性。PMP 指令集包括对物理内存访问的权限管理等指令，用于确保程序的安全性和完整性。

可验证指令的实现原理是基于加密学中的零知识证明技术和可计算安全技术。具体而言，可验证指令可以实现以下重要的程序保护功能：1. 防止恶意篡改：可验证指令可以确保指令的执行过程是不可篡改的，防止恶意篡改程序的执行过程。2. 确认程序正确性：可验证指令可以证明程序的执行过程是正确的，从而避免因为程序错误导致的系统崩溃和数据损坏。3. 防止信息泄漏：可验证指令可以确保程序和数据的安全性，防止信息泄漏和恶意攻击。4. 提高性能效率：可验证指令可以减少程序的执行时间和计算量，从而提高程序的性能效率。总体来说，可验证指令在程序运行的安全性，完整性和正确性发挥着至关重要的作用。

2.3 隔离执行环境（转执行虚拟机）环境证明

隔离执行环境证明 (Proof of Isolated Execution Environment, PIEE) 是一种基于数学和加密学技术的证明方法，用于证明处理器中隔离执行环境的正确性和安全性。我们知道每一台云服务器都存在被攻击的风险，在其上用户代码运行和信息数据保存的安全性的保证极为重要。引入可信执行环境和机密计算（CC，Confidentail Computing）的概念可以比较好应对执行环境证明挑战。在 CC 方法中，一般认为由云系统软件引导的执行环境不可靠，建议在隔离的可信执行环境中（TEE，Trusted Execution Environment）运行安全敏感工作负载。TEE 的安全保证根植于平台的深层硬件层中，远程验证将被用于验证和声明安全。

这里 PIEE 的关键词是隔离和远程证明。主要隔离手段包括内存隔离，CPU 隔离，网络隔离和文件隔离等，这些手段都很好保证了工作的负载运行在可靠的 TEE 中，接下来远程证明将以加密学的形式充分证明 TEE 运行的安全性。在尤特网络架构中将采用 Kubernetes（简称 K8s）部署运行环境，通过隔离手段获得一段用于证明的 TEE 片段，则安全性具体证明如下：

假设某个需要进行 PIEE 的节点 θ 中，在部署

环境里所被指定的语言框架（如 PyTorch）随机获得一个任务函数 (计算算子)$\xi(x)$，并在 Kubernetes 执行容器中抽取一个任务 U 进行一次验证运算，获得结果：

$$Q_\theta = \xi(U_\theta) \tag{1}$$

其中 U_θ 为任务在该节点具体的输入值，Q_θ 为计算结果。

1. 对计算结果加密签名，并与函数算子和任务一并打包发布于链上。2. 各节点获得这个函数算子 $\xi(x)$ 和被加密的结果 Q_θ，通过公钥重新解密获得原结果，并各自进行各自环境里的计算：

$$Q_1 = \xi(U_1) \quad Q_2 = \xi(U_2) \quad ... \quad Q_i = \xi(U_i) \tag{2}$$

其中 U_i 和 Q_i 分别为 i 节点对应的任务输入和最终计算结果，如果计算结果与 Q_θ 相同，证明原节点的 PIEE 通过，每个节点的 PIEE 验证如此类推。

2.4 POCI (Proof of Computation Integrity)

区块链的共识机制常见于计算力的证明，矿机的算力将与区块记账权和奖励息息相关。传统的算力证明最早依赖于工作量证明 (Proof of Work, PoW)，具体实现方式是通过计算一个难以解决的数学难题来消耗计算资源，该难题的解答必须满足一定的条件，则认为拥有满足条件的算力资源。例如比特币中的难题是要对一个随机数进行哈希运算得到一个特定的哈希值 H，该哈希值必须满足比特币网络设定的难度要求 Diff（即 $H>Diff$）[15,25]。因此，解决这个难题需要进行大量的计算，需要耗费大量的时间和计算资源，这就是算力证明的核心思想。但是这个思想存在一个严重的缺点，既巨大的能源和计算资源消耗，而且未能产生实际的社会价值意义 (如 Arweave 网络的永久存储特征)，产生对环境和经济不可忽视的负面影响。此外，算力证明也容易引发算力竞赛和中心化，因为只有大型矿场和矿池才有足够的资源来参与网络，小型节点难以参与竞争，这可能导致网络中的权力集中在少数人手中，与当初的去中心思想有背道而驰之嫌疑。

本小节所展示的算力证明将采用一种全新的证明机制，通过可信算力拥有量间接证明算力，即为可信算力证明 (Proof of Computation Integrity, PoCI)。该证明机制完全摒弃高成本，高消耗的计算模式，取而代之的是芯片算力拥有权的模式。在我们的尤特网络链中，Utility 将采用高性能 Sophon TPU 计算芯片组成强大的算力网，并用于未来提供给用户进行去中心化挖矿和人工智能训练服务。TPU(Tensor Processing Unit) 是为人工智能而设计的加速器芯片，肩负着深度学习计算的任务，特点包括：每秒高达万亿级别浮点运算的高性能；美妙高达万亿级别浮点运算；低功耗；优化的硬件设计和与 TensorFlow、PyTorch 等多种开发框架软件库兼容的灵活性。

如下图 1 所示为一块具备安全引擎 (Security Engine, SE) 的 Sophon TPU，刻蚀于芯片上的序列号唯一的身份识别。在这里最重要部分的是刻蚀在芯片一小段名为 efuse 存储区域的 128 位 AES(Advanced Encryption Standard) 钥匙，即 “secure key(安全钥匙)”。这个安全钥匙将被看成为每片 TPU 芯片的 “秘密身份”，而秘密钥匙只能被送进芯片的 SPACC(security protocol accelerator) 模块（灰色阴影区域）进行加密学运算（AES 加解密运算），同时能保证自身不被暴露。因此这个机制保证了私密性和唯一性。另一方面，在 PKA(public key accelerator) 模块里，可以进行私钥签名和验证。通常在椭圆曲线加密算法 (Ellipse Curve Ctyptography, ECC) 中，取一个基点 P 和一个大随机数种子作为私钥 $PriK$，经过 ECC 加密方程运算，可以得到公钥 $PubK$ [7, 8]：

$$PubK \quad = P \times PriK \tag{3}$$

应用这些工具，接下来展示每位芯片拥有者如何用被刻蚀的安全钥匙，SPACC 模块和 PKA 模块证明自身算力的拥有权：用户先将安全钥匙 S 一次性刻蚀在 TPU 芯片上，并激活 SPACC 模块。芯片商将为每一片芯片提供一组私钥 $PriK$ 和公钥 $PubK$，以及加密解密算法。在 SPACC 模块里，用户可以通过 AES 加密得到加密后的私钥 $PriKEnc$：

$$PriKEnc = \mathcal{A}_\mathcal{E}(S, PriK) \tag{4}$$

用户上传往链上传输 $\{PriKEnc, PubK\}$ 来声称获得芯片的拥有权，并广播于各节点以验证。注意，这些信息是被加密保护后将记录于区块链上，保证了不可篡改的安全特性。用户可以永久丢弃原来的 $PriK$ 或者本地存储于一个私密的地方。由

于 SPACC 模块的存在，恶意节点无法在不拥有正确芯片前提下，从 $PriKEnc$ 偷窃 $PriK$，因为：

$$PriK = \mathcal{A}_{\mathcal{D}}(S, PriKEnc) \tag{5}$$

这里 $\mathcal{A}_{\mathcal{D}}$ 是 AES 解密函数，所以错误或者无效的芯片无法给出正确的 S，也就无法给出正确的 $PriK$，这保证了 $PriK$ 的绝对保密。因此，获得这个芯片身份的用户或节点即可声称该芯片算力的拥有权，而无需进行大功率数学计算来证明算力。接下来部分将展现如何通过一个简单的算法进行 PoCI。

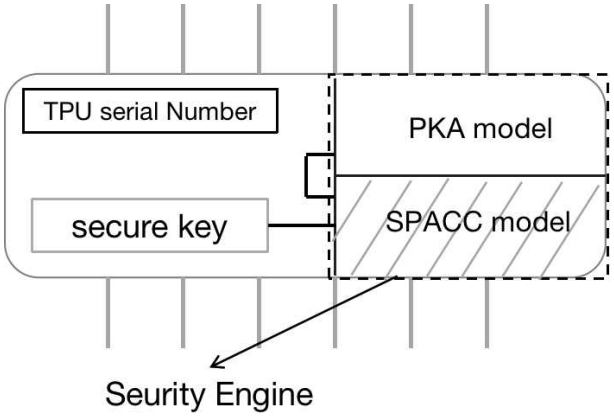

图 1 基于安全引擎 (Security Engine) 的 TPU 简单示意图

假设节点 θ 获得了序列号为 α 的 TPU，那他将获得一个加密的指令集 σ（解码器和解码算法由芯片供应商提供），它可以从中获得签名函数 $Sign_{\alpha}(x)$，并对一段消息 m 进行签名 [7,8]:

$$Sign_{\alpha} = \mathcal{ECC}_{\mathcal{E}}(\mathcal{A}_{\mathcal{D}}(S, PriKEnc)) \tag{6}$$

$$E_{\alpha} = Sign_{\alpha}(m) \tag{7}$$

$\mathcal{ECC}_{\mathcal{E}}$ 是使用 SPACC 模块里获得的 $PriK$ 在 ECC 里的加密签名函数，而 $E(m)$ 为通过 $Sign_{\alpha}$ 签名后的加密信息。其他验证节点则通过从 $PubK$ 的 ECC 解密函数 $\mathcal{ECC}_{\mathcal{V}}$ 推算得到的验证函数 $V_{\alpha}(x)$ 对其进行验证，如果满足：

$$V_{\alpha} = \mathcal{ECC}_{\mathcal{V}}(PubK) \tag{8}$$

$$m = V_{\alpha}(E_{\alpha}) \tag{9}$$

则 PoCI 成功，证明 θ 合法拥有 α，则获得了 α 的算力 ϕ，计为，可以声称获得了大小为 ϕ 的算力。

2.5 计算力定义与 CAT (Computing Availability over Time)

按上一小节 POCI 里面的核心理念，我们就可以定义集群计算力和在线可支配算力 (Computing Availability over Time，CAT)，真正意义上分配全网算力和挖矿权益。再次回到节点 θ，如果它声称获得了 N 个算力 ϕ，那按照以上原理，它会接受 N 次签名来验证 PoCI，显然有点让人生厌，尤其当这个 N 是千以上量级，则会大大增加了证明的工作量。因此提出以下整合优化，将 N 个签名函数整合成一个新的签名函数 $Sign\chi$，由正则整合算子 τ 生成，即：

$$E\chi = Sign\chi(m) = \tau(Sign_1(m)+$$

$$(Sign_2(m) + ... + (Sign_N(m)) \tag{10}$$

其中 $E\chi$ 为 N 个签名整合后的加密信息，其他验证节点则通过 N 个公钥的整合验证函数 $V\chi$ 对其进行验证，其中：

$$D\chi(x) = \tau^{-1}(V_1(x) + (V_2(x) + ... + (V_N(x)) \tag{11}$$

考虑到签名函数，验证函数和整合算子 τ 之间的正则和对易特性：

$$\tau^{-1}V_i(\tau Sign_i) = V_i(Sign_i) \tag{12}$$

$$\tau^{-1}V_i(\tau Sign_j) = 0(i \neq j) \tag{13}$$

不难发现如果满足：

$$Nm = V\chi(E\chi(m)) = V_1(E_1(m))+$$

$$V_2(E_2(m)) + ... + V_N(E_N(m)) \tag{14}$$

则证明获得了的算力。如果当前网络的总算力为 C，则占总网算力的比率 η 为

$$\eta = \frac{N\phi}{C} \tag{15}$$

即为 PoCI 下的计算力。每一个节点的算力将序列化成一段段片段，一片芯片算力计为一个单位片段，$N\phi$ 则抽象为 N 个片段，并标上序号，如下图 2 由 1, 2, ..., n 个节点组成一个总片段 λ，每个节点获得的片段长度则取决于算力大小。并利用 VRF (Verifiable Random Function) 函数每隔一段时间间隔 T 对总片段进行随机取样，得到序号 κ，则查询到片段该芯片序号对应的节点即为爆块节点：

$$\kappa = VRF(T, \lambda) \tag{16}$$

$$\kappa \to \phi(\kappa) \quad \phi \to \theta(\phi) \tag{17}$$

由此机制可看出，拥有大算力的节点片段中占据更大的空间，被 VRF 抽中的概率 η 越大，由此实现了 PoCI。

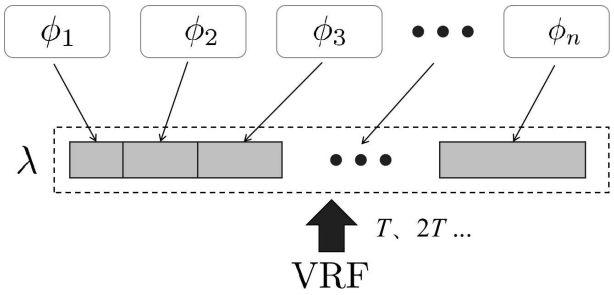

图 2 基于 VRF 对节点组成的算力片段的随机取样过程

然而为了激励节点算力的维持，获得爆块权利的节点需要进行 CAT 验证，即再次通过前面的 PoCI 证明自己在任意时刻 t 均处于高算力状态。同理再次进行算力芯片的数字签名和各节点的验签环节，完成后即获得区块奖励。如此 PoCI 算力证明和 CAT 通过低能耗，低成本，高效率的方式完成了整个尤特网络的 PoCI 机制的实现。

2.6 RPOI (Rent Proof of Instructions)

完成了 POCI，算力拥有者可以考虑如何进一步有效利用算力，他们或考虑链上挖矿，赚取代币（奖励分配将在下一小节描述），进一步购买算力，或者进行租赁 (rent) 服务集群给客户进行分布式挖矿或 AI 训练任务等。在本节，我们将定义租赁拥有权证明，即 Rent Proof of Instructions (RPOI)，完成每一次安全和完整的算力租赁协议。

定义 β 为算力租赁者，R 为用户。假设 R 向 β 发起了租赁 M 个 TPU 芯片 ϕ 的请求 $\gamma(R \to \beta, M\phi)$, 该请求连同时间戳 T, 进行私钥签名 [7,8], 并与本次的押金 ρ 和租金 ω 一起打包交易上链记录于区块，获得交易哈希 $H(R \to \beta, M\phi, T, \rho, \omega)$, 并在使用 π 的公钥加密后将资金质押于公帐地址 π。而 β 收到消息公钥验签后，在消息串里加入芯片加密指令集和相关算力租赁加密文件集合 σ，用 R 的公钥加密再发送给 R，R 用自己私钥解密即获得对算力的租用和使用：

1. R 向 β 发起请求：$\gamma(R \to \beta, M\phi)$

R 完成签名：$E_R = sign_R(m(R \to \beta, M\phi, T, \rho, \omega))$

R 提交交易：$H(R \to \beta, M\phi, T, \rho, \omega) = SHA(E_R)$

2. R 质押资金：$K_\pi(E_R, \rho, \omega) \to \pi$

3. β 验签：$m(\gamma, T, \rho, \omega) = V_R(E_R)$

β 签名和重新加密：$Enc = K_R(sign_\beta(m'(m + \sigma(M\phi \to \beta))))$

4. R 解密获得算力使用：$m'(m + \sigma(M\phi \to \beta)) = V_\beta(D_R(Enc)) \to \sigma(M\phi \to \beta)$

$m(R \to \beta, M\phi, T, \rho, \omega)$ 租赁消息串和 $\sigma(M\phi \to \beta)$ 加密算力文件即为 R 所完成的第一阶段 RPOI(RPOI phase I，图 3)。

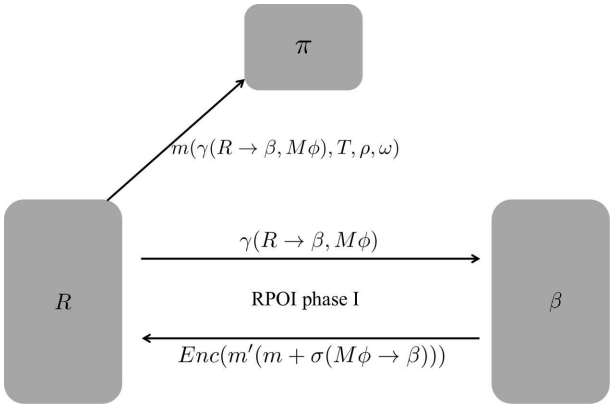

图 3 RPOI phase I 过程示意图

当 R 完成使用并归还算力，需要进行双方的评分 (Rating) 并签名完成对 R 押金退换和 β 的租金收入。步骤如下：

1. 生成对 β 的评分消息 $Ra_R(R \to \beta)$，并将 $m(R \to \beta, M\phi, T, \rho, \omega)$ 和 $\sigma(M\phi \to \beta)$ 整合成新消息串，并签名发送给 β：

$$E'_R = sign_R(m'(m + Ra_R)) \tag{18}$$

2. β 公钥解密签名 E'_R，并生成对 R 的消息评分 $Ra_R(\beta \to R)$，并组合，签名和利用 π 的公钥加密成为“最后的消息”：

$$mfinal = K_\pi(sign_\beta(m''(m' + Ra_\beta))) \tag{19}$$

最后将 $sign_\beta(m''(m' + Ra_\beta))$ 提交上链，把加密的“最后的消息” $mfinal$ 传给 π，由 π 使用自己的私钥解密确认 RPOI 的完成，然后生成调度算子 ζ 和 ξ 分别作用于被 R 公钥解密的加密请求信息 $E(m(\gamma(R \to \beta, M\phi), T, \rho, \omega))$，从资金池里调度资金，分别将押金和租金发往 R 和 β，完成一次算力租赁过程：

$$mfinal \to \zeta(\gamma(R \to \beta, M\phi), \rho, \omega) \Rightarrow \rho \to R \tag{20}$$

$$mfinal \to \xi(\gamma(R \to \beta, M\phi), \rho, \omega) \Rightarrow \omega \to \beta \quad (21)$$

这里 $mfinal$ 包含了之前租赁请求，资金金额，两者互相评分信息，由 π 验证通过，完成第二阶段 RPoI(RPOI phase II，图 4)。π 可以认为是公正的裁决者，$mfinal$ 里任何一部分信息的缺失都会导致资金分配失败。上述第一步骤中，如 R 没能生成评分消息，$mfinal \to \zeta(\gamma(R \to \beta, M\phi), \rho, \omega) \Rightarrow \rho \to R$ 将无法执行；而在第二步骤中，如果 β 没能生成评分消息，则 $mfinal \to \xi(\gamma(R \to \beta, M\phi), \rho, \omega) \Rightarrow \omega \to \beta$ 将无法执行，押金 ρ 和租金 ω 将全部返回 R 地址。以上每一次消息的传递由签名和公私钥加密解密保证过程中的不可篡改性和防止恶意监听 [19,20]。

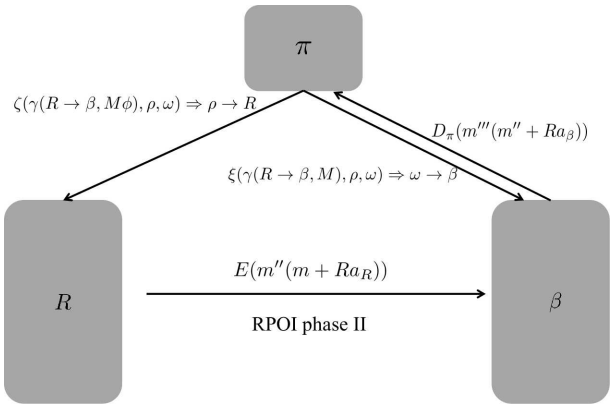

图 4 RPOI phase II 过程示意图

2.7 区块奖励与链上信息

我们规定每一个节点挖矿爆块将获得尤特代币 (UNC) 作为区块奖励，为了激励集群算力的迅速集结，我们设定前 n 个高度爆块奖励为 100UNC，n 至 n_1 为 75UNC，n_1 至 n_2 高度为 50UNC，从 n_2 高度开始变成 30UNC，并遵循规律指数递减机制：每一个高度奖励减少 $0.01exp(-\beta(H - n_2))$UNC，如下图所示 (其中 H 为区块高度，β 为衰减因子)。这个机制激励前期奖励高昂且衰减迅速，激励矿工迅速入场并建立起算力生态；后期代币奖励处于一个稳定慢速下降的过程，维持稳定的生态系统 [15,22]。

节点矿工挖出区块，获得了 UNC 代币奖励，则需要提供链上交易或数据信息记账服务。比如芯片算力的购买交易和 2.4 节 POCI 过程，上一节我们定义的 PROI 两阶段也将作为重要的数字信息记录于区块上。

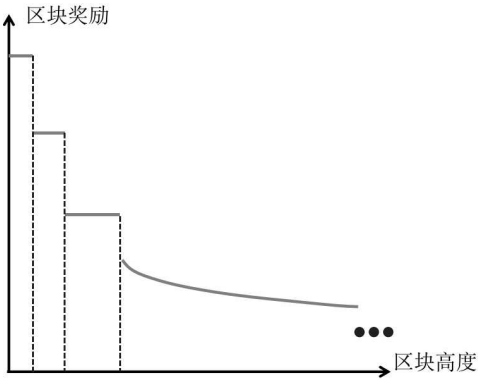

图 5 区块奖励与区块高度关系

$$Reward(H) = \begin{cases} 100 & H \le n \\ 75 & n < H \le n_1 \\ 50 & n_1 < H < n_2 \\ 30 & H = n_2 \\ Reward(H-1) - 0.01exp(-\beta(H - n_2)) & H > n_2 \end{cases}$$

图 6 区块奖励与区块高度关系

3 区块链与数据结构

区块链技术作为一种去中心化的分布式数据存储和管理系统，其核心在于其独特的数据结构设计。区块链数据结构通过加密技术和分布式共识算法，实现了数据的安全、完整和不可篡改。默克尔树（Merkle Tree, MT）[7] 是存储部分区块链数据的一种常用数据结构。MT 是一种二叉树结构，其叶子节点存储数据的哈希值，非叶子节点则存储其子节点哈希值的哈希。MT 的主要作用是提供数据完整性验证和快速数据检索。当数据块数量很大时，通过验证根节点的哈希值就可以判断所有数据块是否被篡改。另外，由于哈希值的唯一性和不可逆性，可以通过默克尔树快速地找到任何一个数据块的哈希值或者验证某个数据块。这种结构可以确保数据的完整性和一致性，同时提高了数据验证和查询的效率。

在实际应用中，区块链往往需要处理大量的链外数据。为了确保链外数据的安全性和完整性，区块链网络可以采用链外数据的链上校验技术。这种技术通过将链外数据的哈希值或默克尔树根节点 [7] 存储在区块链上，实现对链外数据的有效验证。这种方式既降低了区块链网络的存储压力，又

保证了链外数据的安全和完整性。

区块链与数据结构的关系密切。通过默克尔状态机、块构成以及链外数据链上校验等技术，区块链技术实现了数据的安全、完整和不可篡改，为各种应用场景提供了可靠的数据基础。以下小节将详细介绍区块链与数据结构的关系，包括默克尔状态机、块构成以及链外数据链上校验等方面的内容。

3.1 默克尔状态机

作为区块链数据结构的关键组成部分，默克尔状态机（Merkle State Machine, MSM）通过默克尔树实现数据的有序存储和快速验证。MSM 可以维护多套存储体系，以满足不同应用场景的需求。本节将介绍如何使用 MSM 维护三套存储体系：地址账户体系、合约账户体系和链外数据，并结合布隆过滤器实现高效的数据检索。

地址账户体系：通过默克尔状态机维护地址账户体系，可以实现对用户资产和交易记录的高效管理。默克尔状态机为每个地址账户生成一个唯一的哈希值，并将其存储在默克尔树中。这种方式保证了数据的完整性和一致性，同时提高了数据验证和查询的效率。

合约账户体系：默克尔状态机也可以用于维护智能合约的账户体系。通过为每个合约账户生成一个唯一的哈希值，并将其存储在默克尔树中，可以实现对智能合约的状态、变量和函数结果的高效管理。同时，这种方式也有助于降低智能合约执行过程中的数据存储和计算压力 [4]。

链外数据：默克尔状态机还可以用于维护链外数据的存储体系。通过将链外数据的哈希值或默克尔树根节点存储在区块链上，可以实现对链外数据的有效验证。这种方式既降低了区块链网络的存储压力，又保证了链外数据的安全和完整性。

为了实现对这三套存储体系的快速检索，可以采用布隆过滤器（Bloom Filter）技术。布隆过滤器是一种概率型数据结构，可以高效地检测一个元素是否在集合中。通过将布隆过滤器应用于默克尔状态机，可以实现对链上合约地址变量、函数结果、地址资金和链外运算调用记录等数据的高效检索。使用默克尔状态机结合布隆过滤器技术，可以实现对多套存储体系的高效管理和检索，为区块链技术的发展和创新提供强大的数据支持。

3.2 区块构成

区块链由一系列相互链接的数据块组成。每个数据块包含一定数量的交易数据、前一个区块的哈希值、时间戳以及其他元数据。这些数据经过加密和验证后，形成一个完整的区块。区块链网络中的节点将这些区块按照时间顺序链接起来，形成一个不断增长的链式数据结构，确保了整个网络的数据安全和一致性。

区块链中的区块是整个数据结构的基本单元，承载着网络中的交易记录和其他关键信息。一个区块由两部分组成：区块头（head）和区块体（body）。区块的组成以及其各个部分的功能将如下展开。

3.2.1 区块头（block-head）

区块头包含了关于区块的必须公示的关键信息，以下列举：

1、随机数 Nonce：Nonce 用于确保防止双花问题/支链分叉问题的发送，通过工作量证明（PoW）共识算法调整随机数，使得区块头哈希值满足特定的难度条件。

2、混哈希 Mixhash：Mixhash 结构体用于确保可验证计算的多签，为区块链提供额外的安全性。例如，在以太坊中，Mixhash 用于防止长程攻击，确保每个区块的唯一性。

3、交易哈希 TxRoot：TxRoot 用于确保交易的顺序和结构在默克尔状态机 (MSM) 中唯一。通过对所有交易的哈希值进行哈希计算，生成一个默克尔树根节点，存储在区块头中。

4、其他安全字段：例如，时间戳（timestamp, Ts）记录区块生成的时间，确保区块链的时序性。前一个区块的哈希值（previous block hash, prevH）用于将区块链接在一起，形成区块链结构。

因此新的区块哈希值可以表示为 [3]：

$$H(n) = \mathcal{H}(H(n-1), TxRoot, Ts, Nonce) \quad (22)$$

其中，$\mathcal{H}$ 表示哈希函数，n 表示当前区块的高度，$H(n)$ 表示当前区块的哈希值，$H(n-1)$ 表示上一区块的哈希值，$TxRoot$ 表示当前区块包含的所有交易信息的哈希值，Ts 表示时间戳，$Nonce$ 表示随机数。值得注意的是，哈希函数 $\mathcal{H}$ 在这个公式

中是一个单向函数，即无法通过哈希值反推出原始数据，因此保证了区块中的信息不被篡改。同时，$Nonce$ 的引入增加了计算复杂度，使得区块链网络更加安全和可靠。

3.2.2 区块体（block-body）

区块体包含了所有的交易记录，每个交易包括以下几个部分：

1、交易发起方签名 Sign：交易发起方对交易进行数字签名，确保交易的安全性和不可抵赖性。

2、目标地址 TargetAddress：交易的接收方地址。

3、交易数据 TxData：交易数据包含了所有的智能合约发布、算子调用栈、数据堆叠路由信息以及自定义二进制数据。这些数据用于指导智能合约的执行和处理链上业务逻辑。

4、值 Value：交易中涉及的代币数量（金额）。

综上，基本的区块结构包含了区块头和区块体两部分，承载着区块链网络中的交易记录和关键信息。这种结构确保了区块链数据的完整性、安全性和唯一性，为去中心化的分布式应用提供了可靠的基础设施。

3.3 链外数据链上校验

为解决区块链的扩展性问题，链外数据链上校验通过采用类似 Layer-2 方案的扩展扩容和共识机制，将部分数据和计算任务移到链下处理，同时保证链上数据的安全性和完整性。本节将简要介绍如何通过 rollup 方案实现链外数据链上校验。

Rollup 是一种 Layer-2 扩展方案，它将链下的交易数据进行聚合处理，然后将聚合后的数据提交到链上。具体而言 rollup 方案包括以下几个步骤：

1. 链下数据聚合：在链下环境中，各个节点处理数据集的传输和计算服务任务。这些节点将交易数据进行聚合，形成一个称为“rollup”的数据结构。

2. 链上数据提交：将聚合后的 rollup 数据提交到 Layer-1 网络，如以太坊主链。这些提交的数据包括聚合交易的元数据（meta-data），如默克尔树根节点哈希值等。

3. 链上数据校验：Layer-1 网络节点对提交的 rollup 数据进行验证，确保链下数据的正确性。这一过程主要通过验证默克尔树根节点哈希值实现。

整个 Rollup 链外数据校验流程也可如下图所示：

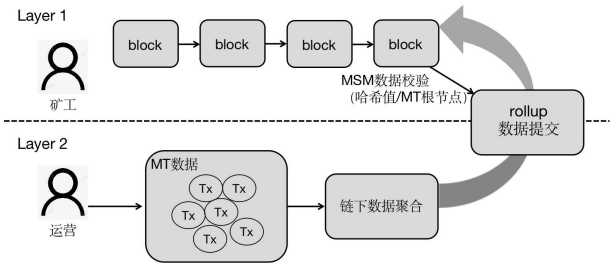

图 7 Rollup 链外数据校验流程

通过 rollup 方案，也可以将大量数据和计算任务从链上迁移到链下处理，降低链上网络的负担，提高区块链的吞吐量和可扩展性。同时，rollup 方案通过将聚合数据提交到 Layer-1 网络，并进行链上校验，保证了链下数据的安全性和完整性。

4 代币循环与计算生态

代币循环与计算生态是区块链网络系统中关键的组成部分，涉及到挖矿机制、代币减产、Gas 费用、Burn Fee、机时租赁订单以及推理部署订单等方面。其中挖矿机制是区块链网络系统的核心组成部分，负责创建新的代币并将其分配给为网络提供计算资源的参与者。尤特网络中的挖矿机制采用可信计算证明（PoCI）共识算法，将挖矿奖励与参与者提供的计算能力挂钩。而为了维持 UNC 的稀缺性和价值 [15, 20]，代币减产机制也尤为重要。减少代币的方式则用到了 Burn Fee[16]。为了发挥计算生态的优势，一种去中心化的计算资源租赁机制和生成推理部署订单，将分别允许用户租赁闲散计算资源进行执行复杂的计算任务和将 AI 模型部署在计算节点上以完成模型的训练和推理任务。

由此综上，代币循环与计算生态在尤特网络中发挥着关键作用，通过多种机制共同维护网络生态和代币经济的安全性、稳定性和可扩展性。以下将展开介绍这些概念及其在尤特网络中的应用。

4.1 挖矿机制

尤特网络的挖矿机制采用可信计算证明（Proof of Computation Integrity, PoCI）共识算法，将挖矿奖励与参与者提供的计算能力紧密联系。在这个过程中，全网算力是由所有参与者提交的算

力共同确认交易并达成共识。为了确保挖矿过程的公平性，尤特网络采用了链上随机性决策机制，通过可验证随机函数（Verifiable Random Function, VRF）[17] 来生成随机数，从而在所有参与者中无偏置地选择挖矿者。

挖矿者需要根据 VRF 生成的随机数和自身的算力，完成区块的挖掘任务。算力越大，挖矿者在网络中爆块的概率将线性提升。一旦成功挖出一个区块，挖矿者需要提交一个证明并广播，以表明自己已经完成了区块的挖掘任务。网络中的其他节点将对这个证明进行验证，确保其有效性。

尤特代币（UNC）作为尤特网络的原生代币，可以通过挖矿来获得。在尤特网络的代币发行计划中，97% 的 UNC 将通过挖矿方式分配给参与者。这种挖矿机制激励更多的参与者加入尤特网络，提供计算资源，从而提高网络的安全性和稳定性。

尤特网络的挖矿机制依赖计算力证明和链上随机性决策两者可以共同确保挖矿过程的公平性和透明性。通过此机制，UNC 可以在网络中广泛分布，激励更多参与者为尤特网络提供计算资源，从而促进其安全、稳定和可扩展性。

4.2 代币减产

尤特网络采用了类似比特币的代币减产机制 [3]，以控制代币总量并维护其稀缺性和价值。这里规定每过 2 年（或某个预定的区块高度），尤特网络的挖矿奖励将减半。数学上可以表示为：

$$q = q_0 \times 0.5^{\frac{t}{2}} \tag{23}$$

其中，t 为经历时间（以年为单位），q 为 t 时刻的奖励，q_0 为初始奖励，即在网络刚启动时每个区块的挖矿奖励。

代币减产机制使尤特代币的总量受到限制，从而确保其长期稀缺性和价值。随着时间的推移，挖矿奖励不断减少，新产生的代币数量逐渐减少。这种设计有助于抵消通货膨胀的影响，保持代币价值的稳定增长。

尤特网络的代币减产机制与市场供需关系相辅相成。随着 UNC 的稀缺性增加，市场上对其需求可能会上升，从而推动其价值上涨。此外，代币减产还能激励参与者在网络初期加入，以获得更高的挖矿奖励，从而有助于网络的成长和发展。

综上，尤特网络的代币减产机制通过限制代币总量，维护了 UNC 的稀缺性和价值。这种设计有助于抵消通货膨胀的影响，保持代币价值的稳定增长，并激励更多参与者加入网络。

4.3 Gas 和 Burn Fee

在尤特网络中，智能合约的执行需要消耗一定量的燃料 (Gas)，以衡量计算资源的使用情况。Gas 的设计借鉴了以太坊 [4,5] 的方案，通过对合约运算符的计算开销进行度量，从而为智能合约的执行分配相应的计算资源。

每个运算符都有一个预定义的 Gas 消耗值，这取决于其执行所需的计算资源。当用户提交智能合约交易时，需要为交易附带足够的 Gas 费用，以确保合约执行的顺利进行。尤特网络中的执行虚拟机（VM）会在运行合约时跟踪 Gas 消耗，确保其不超过用户提供的限额。一旦 Gas 消耗超出限额，合约执行将中止，避免无限循环等恶意行为。

此外，尤特网络引入了燃烧费 (Burn Fee) 机制，即每一笔交易和消息传输都会有一部分费用被燃烧并永久移出链上流通。这种设计也将有助于降低代币的通胀率，提高代币的稀缺性，保持代币价值的稳定增长。Burn Fee 的具体数值可以根据网络的实际运行情况进行调整，以确保网络的经济健康。

总之，Gas 和 Burn Fee 机制在尤特网络中起到了重要的作用。Gas 用于度量合约执行的计算资源消耗，保证网络中的计算资源分配公平有效；而 Burn Fee 则有助于降低通胀率，提高代币的稀缺性，从而维护代币的价值。这两种机制共同为尤特网络提供了一个可持续发展的经济基础。

4.4 机时租赁订单

在租赁拥有权证明 (Rent Proof of Instructions, RPOI) 的支持下，尤特网络中也推出了允许用户在链上租用计算资源的机时租赁订单。这一过程可以用数学符号描述如下：设 $\mathcal{O}$ 表示一笔机时租赁订单，其中包括以下几个主要元素：

u: 用户

t: 租赁时间（以秒为单位）

r: 计算资源需求（例如 CPU 核心数、内存大小等）

p: 用户出价（以 UNC 为单位）

e: 计算环境（例如虚拟机类型、操作系统等）
s: 服务提供者
f: 评分交易
v: 验证结果集
订单发送到链上的过程可以表示为：

$$\mathcal{O}(u,t,r,p,e,s,f,v) \rightarrow UtilityNet \tag{24}$$

即 2.4 节中提到的请求 γ 的具体表现形式。

一旦订单被发送到链上，网络中的计算资源提供者将根据用户需求和出价决定是否接受订单。接受订单后，服务提供者需按照用户的计算环境要求分配资源，完成加密文件签署，并在租赁期间提供服务。

在服务完成后，用户和服务提供者进行评分交易。评分交易 f 包括服务质量和用户体验等方面的信息，有助于构建一个公平、透明的租赁市场。双方可以通过验证结果集 v 确保租赁过程中的计算结果的正确性和完整性。验证结果集可以包括计算输出、执行日志等信息，以便用户对租赁服务的质量进行评估和对用户和服务提供者的资金调配。

由此可见机时租赁订单在尤特网络中实现了计算资源的有效分配和利用，为用户提供了灵活、可靠的计算服务。通过链上交易、评分机制和验证结果集，尤特网络确保了去中心和租赁市场的公平、透明和安全。

4.5 推理部署订单

基于 TPU 部署的尤特网络，具备了成熟的 AI 模型训练和推理环境，因此也推出了推理部署订单服务。推理部署订单允许用户在链上使用 TPU 资源对被训练的 AI 模型进行日常的推理演算。
设 $\mathcal{R}$ 表示一笔机时租赁订单，其中包括以下几个主要元素：
u: 用户
t: 租赁时间（以秒为单位）
r: 计算资源需求（例如 CPU 核心数、内存大小等）
m: 训练的算法模型
d: 训练的数据语料库
p: 用户出价（以 UNC 为单位）
s: 服务提供者
f: 评分交易
v: 验证结果集
订单发送到链上的过程可以表示为：

$$\mathcal{R}(u,t,r,m,d,p,s,f,v) \rightarrow UilityNet \tag{25}$$

同样地，一旦订单被发送到链上，网络中的计算资源提供者将根据用户需求和出价决定是否接受订单。接受订单后，服务提供者需按照用户的模型，数据和环境要求分配资源，完成加密文件签署，并在租赁期间提特定的 AI 训练推理服务。审核通过的推理部署订单会被匹配到合适的推理节点上进行部署。匹配过程会根据节点的空闲资源、硬件配置、节点信誉度等因素进行综合考虑，确保推理部署的高效性和可扩展性 [26]。经过匹配后，系统会将推理部署订单发送给被选中的推理节点进行部署。推理节点会下载模型文件和数据文件，并根据参数设置进行推理计算，将结果返回给用户。

用户收到推理结果后，系统会对结果进行验证，确保结果的准确性和安全性。验证过程包括结果的数字签名验证、数据完整性验证、算法可解释性验证等，只有通过验证的结果才会被认为是有效的推理结果。

推理部署订单完成后，与机时租赁订单中相同，用户和服务提供者进行评分交易，验证后系统会自动进行结算，并将费用分配给参与推理部署的节点和押金退还用户。从请求到结算过程是由区块链智能合约订单自动执行的，费用的安全性和公正性能得到最大保证。

4.6 UNC 经济模型

Utitlity 网络代币具有一套完整成熟的经济学体系模型。根据计划设定分为了四个纪元 (Age) 时期：塔罗斯纪 (Talos Age), 金刚纪 (Vajra Age), 格勒姆纪 (Golem Age), 玛丽亚纪 (Maria Age)。以下将进行具体阐述。

4.6.1 Talos Age

截止该白皮书发布为止，Utility 网络环境正处于第一纪元塔罗斯纪 (Talos Age)，在此阶段 UtilityNet(UNC) 代币发行，公链基金会成立，社区和公链处于开发阶段。在此阶段内将开启第一次内测，将基于 BEP20（币安智能链 BSC）来进行激励模型测试。参与者通过去中心化的网络进行交易，

然后通过在其区块中写入和处理消息来更新网络状态。运行传输数据和消息会消耗网络上的计算和存储资源。相应的，所消耗的 GAS 费为 BNB。

内测阶段也需要填充一定数量的 UNC 作为燃料 (Fuel) 燃烧进行挖矿，计划燃料在 200 天燃烧完毕，每天燃烧总量的 5% 会进入项目基金会钱包作为运营和技术开发经费。挖矿产出的 UNC 币收益直接释放，没有锁仓。产量将以每 200 天为一个出矿周期，每个周期减产 5%，预计 90+ 周期能够产完 97 亿枚。设定初始周期产量 5 亿枚，每天产 250 万枚币。具体如下表格示：

	第1个周期	第2个周期	第3个周期	第4个周期	
释放总量	5亿	4.75亿	4.5125亿	4.286865亿	……
每日释放	250万	237.5万	225.625万	214.3438万	……
每个周期减产5%					

表 1 UNC 代币释放量与周期关系

根据这个产出规律，Utility 预计铸造限额为 100 亿 UNC，即为 UtilityNet 通证。而最终 UNC 的燃烧总量为 95%，总量通缩至 5 亿枚。UNC 将会被分配给 UtilityNet 矿工作为挖矿奖励，用于提供 AI 算力、区块链维护、分发数据、运行合同等服务。未来这些奖励将支持多种类型的开采。

4.6.2 Vajra Age

在第二纪元金刚纪 (Vajra Age) 里，测试网 Utility Testnet 将部署上线，并公布早期激励计划。算力矿工是唯一一个在网络上线时获得奖励的矿工群体，它也是最早的矿工群体，同时负责维持协议的核心职能。激励计划的宗旨旨在奖励这些早期的矿工来达到算力快速集结的效果。

4.6.3 Golem Age

第三纪元格勒姆纪 (Golem Age) 中的重大事件包括主网 Utility Mainnet 将部署上线，测试网 UNC 代币映射到主网 UNC 代币和展开人工智能去中心化算力时代。一个完整的 UNC 生态网络正式成立。

4.6.4 Maria Age

在第四纪元玛丽亚纪 (Maria Age) 里，人类制造的最大参数模型训练和运行于尤特网络中。一个强大的 UNC 生态网络不断发展，并扩大 AI 训练规模，构建出万亿级别参数的模型和超万兆亿的算力网。

5 执行虚拟机

本章主要关注尤特执行虚拟机 (Virtual Machine, VM) 的实现和集成。执行虚拟机是整个尤特网络中的关键组件，负责处理各种计算任务和执行智能合约。本章将详围绕以下几个方面进行阐述：

1. 虚拟化和隔离执行环境：即讨论如何利用虚拟化技术创建安全、隔离的执行环境，确保计算任务在一个受控制和安全的沙箱中进行。

2. 环境证明：证明如何生成和验证执行环境，确保计算任务的正确性和可靠性。

3.Solidity VM：介绍 Solidity 虚拟机在尤特网络中的作用，以及如何与尤特的其他组件进行集成。

4.Kubernetes 集成：探讨如何将尤特虚拟机与 Kubernetes 进行集成，以便最大化高效管理和调度分布式计算任务。

5.WASM 集成：WebAssembly 技术提高尤特虚拟机的性能和兼容性的可能。

6.PyTorch 外联运算支持：将尤特虚拟机与 PyTorch 进行集成，支持更广泛的人工智能和机器学习应用。

接下来将深入探讨以上关键技术，解释如何将它们和尤特网络有机结合，形成高效、安全、可扩展的计算生态。

5.1 虚拟化和隔离执行环境

在尤特网络中，虚拟化和隔离执行环境对于保障计算任务的安全性和可靠性至关重要。尤特采用基于 Arm 框架驱动和显存虚拟化支持的技术，结合 TPU 芯片的显存虚拟化，实现高效的资源管理和调度。借助 Arm 框架驱动，尤特可以充分利用底层硬件资源，实现对计算任务的高性能处理。同时，显存虚拟化技术使得 TPU 芯片能够在不同的计算任务之间灵活划分和共享显存资源，提高整体的计算效率 [27]。框架如下图所示：

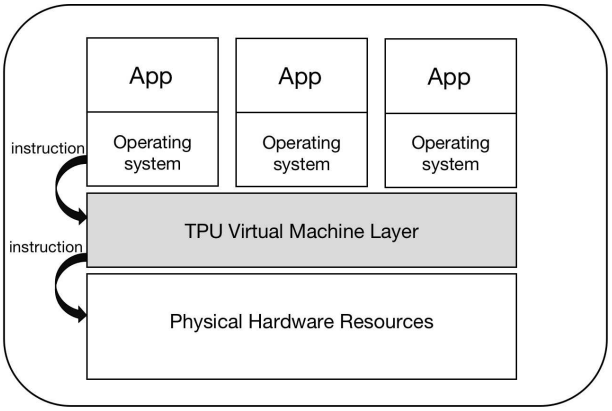

图 8 TPU 虚拟化架构

在尤特网络中，每个隔离执行环境都位于一个容器内，这些容器包含了标准镜像和数据算法挂载。标准镜像为执行环境提供了基本的运行时环境和依赖库，而数据算法挂载则包含了用于处理计算任务的具体算法和数据。通过将执行环境与容器技术相结合，尤特能够确保计算任务在一个受控制、安全的沙箱 (sandbox) 环境中进行，对潜在的安全风险和数据泄露进行有效干预。

5.2 环境证明

尤特网络采用一种去中心化的环境证明机制，以确保执行环境的正确性。该机制通过共识算法和爆块过程来实现，强制要求矿工在每轮共识中都具备标准环境执行力以及相应的计算能力。

在尤特网络中，新区块的生成需要矿工在短时间内（如几秒钟）完成一系列任务。这些任务包括在标准执行环境下运行随机代码以产生 PyTorch 算子。在这个过程中，矿工需要确保其执行环境满足网络的要求，同时具备足够的计算能力来训练和推理任务。

当矿工完成任务并准备提交新区块时，其他矿工需要对其结果进行验证。这一验证过程涉及到检查预爆块方生成的 PyTorch 算子是否正确。如果大部分矿工同意该算子的正确性，则新区块被认可并添加到区块链上。反之，如果算子无法通过验证，则该爆块被视为无效，共识进程进入下一轮。

这种环境证明机制有效地阻止了潜在的作恶者。因为在尤特网络中，作恶者在极短的时间内无法启动一个符合要求的执行环境来完成任务，从而错失爆块的机会。通过这种方式，尤特网络确保了执行环境的正确性，同时为训练和推理任务提供了可靠的计算基础。

5.3 Solidity VM

尤特网络支持使用 Solidity 语言编写智能合约，并在自己的 VM 中进行执行。为了实现与硬件、运算和数据的高效连接，Solidity VM[28] 中可被允许引入一系列新的运算和结构，包括基础数据类型等。这些扩展功能可以帮助构建与底层硬件更紧密集成的应用。

通过采用第 3 节描述的 rollup 技术，尤特网络实现了链上与链下数据的高效传输和校验。在这个架构下，一层网络负责撮合交易并确保执行双方的代币支持，而二层网络则负责实际的运算和数据处理。

尤特网络的去中心化特性保证了计算状态的分布式管理，使得算法的运算过程紧密绑定于矿工本体。通过提供标准服务套件，尤特网络确保了从大型数据中心到单机矿工的所有参与者都能获得一流的服务体验。这种设计使得尤特网络具备高度的可扩展性和强大的计算能力，为未来的分布式计算应用奠定了坚实的基础。

5.4 Kubernetes 集成

尤特网络的火山引擎与普通 Kubernetes 集成，共同构建了矿工侧的可验证环境和服务基础设施。火山引擎作为一个高度优化的计算引擎，与 Kubernetes 无缝对接，实现了强大的计算能力和高度可扩展性。

通过集成 API Gateway、调度器（Scheduler）、张量计算单元（Tensor Processing Unit, TPU）以及快速缓存虚拟化技术，尤特网络在 Kubernetes 平台上构建了隔离的沙箱环境。这些隔离沙箱通过容器技术和标准镜像实现，使得尤特网络能够支持超大规模的并行计算任务，例如 Transformer 模型的训练和推理部署。

在尤特网络中，Kubernetes 负责处理基于容器的应用部署、扩展和管理。这使得尤特网络能够充分利用 Kubernetes 生态系统的优势 [29]，包括灵活的资源管理、高度可扩展性以及对各种硬件和软件平台的支持。Kubernetes 的集成还提供了一系列先进的工具和服务，包括自动弹性伸缩

（Autoscaling）、服务发现（Service Discovery）以及故障恢复（Fault Recovery）等，以确保在分布式计算场景中实现高可用性和高性能。同时尤特也将虚拟化技术与 Kubernetes 进行集成，实现对计算任务的自动调度和管理。Kubernetes 则根据任务需求和资源状况，动态分配和调整 TPU 芯片的显存资源，确保尤特网络中的计算任务能够高效、稳定地运行。

由此可见，尤特网络与 Kubernetes 的集成为构建分布式计算环境提供了一个强大的基础。通过这种集成，尤特网络能够为各类计算任务提供高度可扩展、高性能且安全可靠的计算资源，满足未来人工智能和大数据应用的需求。

5.5 WASM 集成

尤特网络中，WebAssembly（WASM）的集成是实现跨平台算法代码运行的关键。WASM 是一种通用的二进制指令集，旨在为各种计算设备提供一种高效、紧凑的 VM 格式 [30]。通过 WASM 集成，尤特网络能够支持各种不同的硬件和软件环境，以确保跨平台的兼容和相关性能。

WASM 与尤特网络中的 rollup 后的算法代码集成紧密。在尤特网络中，算法代码首先会被编译成 WASM 字节码，然后通过 rollup 技术进行打包和优化。这样，WASM 字节码可以快速地在尤特网络中的各个节点上执行，提高整体计算效率。

此外，尤特网络与 Kubernetes 的 WASM 执行环境无缝对接。Kubernetes 可以利用 WASM 运行时（Runtime）支持多种语言和平台，使得尤特网络中的算法代码能够在不同的计算设备上运行。这种无缝集成有利于降低资源消耗，提高计算效率，同时确保代码的安全性和可靠性。

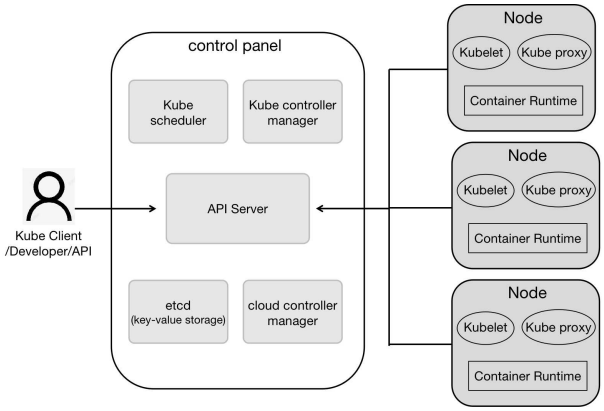

图 9 Kubernetes 框架与 WASM 环境的结合

通过 WASM 集成，尤特网络能够实现高度灵活、可扩展的计算生态。无论是处理复杂的 AI 模型训练任务，还是执行轻量级的推理部署，尤特网络都能够在各种硬件和软件环境中提供强大的计算能力。

5.6 PyTorch 外联运算支持

5.6 PyTorch 外联运算支持尤特网络的 PyTorch 外联运算支持解决了 Solidity 合约无法直接嵌入基于 Python 的 PyTorch 代码的问题，尤特网络通过 rollup 技术，允许链上合约与链下 PyTorch 代码进行高效的协同计算 [31]，从而实现了对 AI 的服务能力展现。

在尤特网络中，训练数据和函数输入通过 rollup 技术传输至矿工端，与 Layer1 的合约约定相结合。这种结构使得矿工可以在链下环境中执行扩展计算或 AI 模型的训练和推理。在计算完成后，计算结果会通过 rollup 回传至链上，与链上合约进行交互和验证。

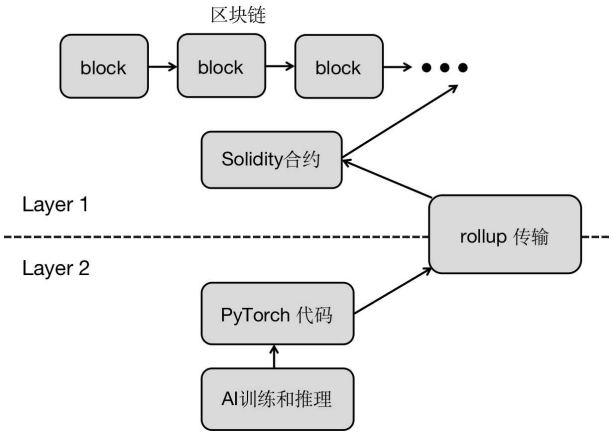

图 10 PyTorch 外联运算支持

通过引入 PyTorch 外联运算支持，尤特网络实现了链上与链下之间的无缝连接。这种混合计算模式使得尤特网络能够充分利用现有的 AI 框架，如 PyTorch，进一步提升计算效率和性能。此外，这种方法还为 AI 领域带来了更大的创新空间，支持研究人员和开发者在尤特网络中开发更复杂、更强大的 AI 应用。

6 基于尤特区块链的人工智能算力网络

尤特网络的运营模式允许出现 AI 算力服务提供商，AI 算力服务提供商通过租赁 IDC 机房自建人工智能算力中心，通过尤特区块链可以在链上公平地认证 AI 训练和推理芯片的类型、算力大小、是否在线等信息，从而确认其在尤特网络上的总算力。

但是，AI 算力 ≠AI 算力服务。算力订单买方进行 AI 模型训练和部署需要的是具有相当服务质量的 AI 算力服务。链上在线的 AI 算力只是 AI 算力服务的其中一个生产要素，算力服务提供商提供 AI 算力服务，还需要采购 CPU 服务器，硬盘，租用网络带宽，维持 AI 服务器稳定电力供应等等生产要素。如何衡量与保证 AI 算力服务的服务质量？这一问题的高效解决方案将决定尤特区块链是否具有生产力、尤特币是否能形成流通闭环。我们设计了一个链下的人工智能算力网络方案来高效联接算力服务提供商各地的人工智能算力中心，

实现数据、算力、算法、模型、服务等多要素的高效共享与流通。且搭配基于尤特区块链的 AI 算力服务质量评分机制，链上可以追溯和公示算力服务买卖双方的交易。链上提供公开验证能力，链下提供高效的智能时代生产力，链上链下的结合可最大程度激励 AI 算力服务商加入尤特网智算网络，降低算力使用费用，推动异地异构算力统一结算，使 AI 算力的生产、分配、流通、消费各环节更加畅通，提高市场运行效率。目标构建一个世界范围内的算力资源方便接入、任务高效统一调度、用户可方便廉价使用且具有可持续发展运营模式和机制的 AI 算力基础设施。

尤特智算网络平台层设计了四个子平台，分别为智算中心调试训练部署一站式平台、智算网络任务统一调度与管理平台、智算网络数据统一存储与管理平台、智算网络数据市场。

6.1 智算网络任务统一调度与管理平台

人工智能任务统一调度与管理平台将不同智算中心不同类型的线上资源进行有效地调度和管理，以最大程度地提高整个智算网络的利用率和效率。

6.1.1 功能

采集资源信息：算力资源不仅包含 TPU，还包括 CPU、GPU、TPU、NPU、GPGPU、FPGA 等，网络资源包括具有专线互联网、公网、受控管理的超算内网、受控管理的智算内网等，数据资源包括各种人工智能的公开数据集、用户私有数据集等，网络上每种资源都有其独特的优点和适用场景。

整合资源：根据每种异构资源的最小粒度不一样，人工智能任务要方便使用，往往需要组合单种算力资源且搭配不同种类的异构资源，形成一个资源包，将资源包分配给最合适的任务需求，以优化任务的完成效率和资源的利用率。

管理作业：每个人工智能作业往往包含多个嵌套的任务，可称为子任务，各个子任务可被分布式地调度到不同的智算中心，作业调度成功将构成作业所需的子任务网络。适时需要对作业、子任务运行状态和性能进行监控和管理，及时发现问题并进行处理。

调度作业：平台对用户提交的作业进行调度，根据各智算中心负载、数据位置、算力价格因素、通信效率、作业资源需求等因素（简称” 调度因子”，下面会对 “调度因子” 进行概述）将作业所包括的各个子任务调度到合适的智算中心运行，以最大化任务的完成效率和资源的利用率。

调度策略：将若干调度因子作为输入，使用一定的处理逻辑对符合作业资源需求的候选智算中心进行处理，输出每个智算中心的规范化评分。各个调度多略的输出将作为调度评估模型的输模型进行进一步的综合决策。调度策略根据实际的场景需求确定，可选的调度策略有负载最小优先、资源空闲优先、数据亲和性、算力价格最低优先、算力性能最高优先、网络性能优先等。

调度因子：调度因子根据实际的场景需求选择，也可自定义调度因子，可选的调度因子包括：作业要求、资资源需求规格、数据位置、算力价格、

算力性能、智算中心负载等。

调度评估模型：多个调度策略的输出将作为评估模型的输入，经过评估模型的计算得出最优的智算中心，获得最终的调度结果。采用以下的评估模型：对于一个等待调度的作业，使用了调度策略 S_1,S_2,.....,S_n，分别为这些调度策略赋予权重 W_1,W_2,.....,W_n. 在第 k 个智算中心 C_k，每个调度策略输出的评分为：G_{k1},G_{k2},.....,G_{kn}, 则智算中心 C_k 最终分数 G_k:

$$G_k = \sum_{i=1}^{n} G_{ki} \times W_i \tag{26}$$

从所有智算中心 C_1,C_2,....C_n 中选出最终评分最高的作为最终的调度结果，将任务派发到目标智算中心。

数据调度因子：平台可通过人工智能数据统一存储于管理平台（详看 6.2 章节）查询接口获取目标数据集的 Id、名称、版本、数据源分布位置、数据缓存分布位置、访问权限等相关信息。还可通过人工智能数据统一存储于管理平台查询接口智算中心否允许下载、公网下载带宽、平均下载耗时、最大下载并发数、最大下载比特数、近期下载失败率等信息。平台依据以上目标数据集与目标智算中心相关信息作为调度因子之一。

数据调度策略：调度引擎依据数据调度策略的输出决定是否进行跨智算中心迁移数据集。平台有以下两种数据调度策略。

算随数动策略：存在目标数据集的智算中心都可以输出为目标作业可调度的候选智算中心。作业选出将要调度的目标智算中心后，根据候选数据源的多维数据调度因子筛选最优数据源，然后向目标智算中心迁移并缓存该目标数据集，等待数据迁移任务完成后再调度目标作业。

6.1.2 架构

人工智能任务统一调度与管理平台的核心是云际管理与调度系统，它负责将用户任务分配到不同的异构智算中心。

在设计平台架构时，我们采用了存算分离的设计思想，将云际资源管理拆分成算力资源管理与数据管理两大组件，以实现高度可扩展性和灵活性。在该架构中，云际管理与调度系统分为四个主要模块：作业管理模块、调度引擎模块、异构算力资源管理模块、数据管理模块。各模块的主要功能如下：作业管理模块：负责完成作业对象的增删改查、日志管理、运行负载监控等功能。

调度引擎模块：根据内部调度策略与调度评估程序的综合打分，将计算作业分发到目标智算中心，最大化任务的完成效率和资源的利用率。

异构算力资源管理模块：负责对智算中心异构计算、网络、数据资源进行统一接入和管理，周期性地采集资源信息、分类、存储和管理，以供后续作业调度使用。

数据管理模块：实时感知智算网络目标数据资源，当目标智算中心缺少人工智能作业所需目标数据集时，数据管理模块可感知网络位置最优的数据源，为调度引擎提供数据迁移策略，实现数随算动、算随数动调度效果。

6.2 智算网络数据统一存储与管理平台

智算网络需要将大量任务调度到满足算力资源条件的智算中心，不同的计算任务依赖于各种各样的数据集，这些数据集可能分布在某一个或多个智算中心里，同时各智算中心可能使用不同的存储系统，存储架构和对外接口存在较大的差异性，用户使用起来复杂，云际间多次数据集迁移将付出带宽占用、延迟时间、副本存储空间等开销成本。因此，智算网络需要针对云际间的数据构建统一存储与管理平台，协同管理智算中心异构存储介质和数据集，提高智算网络生态参与者之间资源共享的效率，赋予智算网络更高的扩展性，为未来更多的智算中心和数据加入到智算网络生态提供更多便利性。

6.2.1 功能

快捷接入异构存储系统：各个智算中心的存储系统可能是异构的，采用不一致的数据存储方案，应用接口协议可能是 S3、OBS、OSS、MINIO、FTP 以及各种自定义存储系统 API。要实现网络数据的高效管理，则需要对各个智算中心异构存储系统的接口进行适配和统一，形成一套内部统一的存储管理接口。

系统高可用：接入的智算中心归宿不同的算力服务商，且具有地理隔离性和管理隔离性，接入的异构存储系统不可认为完全可靠与一直在线。智算中心突然下线其存储系统应不影响正常其他智算中心继续服务。

感知数据集：为提高计算任务的效率，实现算随数动的调度策略。智算网络需要对数据集生成全网唯一 id 并感知数据集在各个智算中心的存储情况，形成全局数据存储视图。依据该全局视图，人工智能任务统一调度与管理平台可查询目标数据是否存在于目标智算中心。在多个智算中心同时满足计算任务算力需求的情况下，可优先将用户计算任务调度到数据集已存在的智算中心，提高任务启动效率。

验证数据集完整性：人工智能领域的数据集可能包含大量个文件夹和多个文件，人工智能训练任务强依赖于数据集，数据集的不一致将影响人工智能模型的生成效果，快速验证数据集完整性将是该平台的一个挑战。

支持多版本数据集：人工智能领域的数据集保存着各种标注数据，随着社会的发展和商业公司的发展投入，同类数据集往往会不断新增内容，这种情况下需要新建不同的数据集版本，以区分和归档不同时期的数据集内容。

数据集云际迁移：算力服务商可能以私有云方式部署智算中心，也可以将智算中心部署到公有云上。当计算任务将被调度到智算中心 N，当该中心没有任务指定的目标数据集，则需在云际间迁移目标数据集。平台感知到智算中心 M 存在该目标数据集，就可以以智算中心 M 为数据源迁移到目标智算中心 N 的存储系统中。

数据集压缩：人工智能领域的数据集大多都是非结构化数据，经过压缩后再存储到智算中心，将会使智算网络节省大量传输带宽、传输时间和存储空间。

数据集缓存：数据集在云际间迁移需要经过 4 个步骤：感知数据源、从数据源下载到中间代理服务器、代理服务器将数据集上传到目标智算中心。每次迁移需要通过公网进行上传与下载，带宽和时间成本巨大。在智算网络中构建一个数据集的内存缓存网络将会极大提升数据集传输效率，大大提升人工智能任务调度完成效率，是的整个智算网络使用体验大大提高。

支持多租户：智算网络将服务多用户、多机构。数据集具有隐私属性，要求平台具有一套完备的用户角色权限管理系统，支持用户对数据集进行私有、有限域公开、全网公开等访问权限的管理。

对外统一 API：人工智能数据统一存储与管理平台与人工智能任务任务统一调度与管理平台采用存算分离架构，人工智能数据统一存储与管理平台支持多租户，需要开发一套向外统一的 API 接口，提供多方调用进行数据集的感知与迁移。

6.2.2 数据结构

本系统存储数据集前需对原数据进行压缩和格式化操作，生成元数据（DataSet Metadata Part）和原数据块 (Dataset Blobs Part) 两部分。其中，元数据部分负责存储原数据集文件的描述信息和文件夹层级结构，原数据块部分是数据集所有文件的分块。原数据块部分每个数据分块都会生成一个哈希摘要（Digest），摘要算法设计为可变可配置（可选 Sha256,Blake3 等），以适应日益强大的算力破解攻击和性能更高的哈希算法创新。根据原数据集的文件夹层级结构，在元数据部分形成一个哈希摘要的默克尔树数据结构，并产生数据集的唯一 id。由于原数据块部分的存在，单个数据集内相同的灵活地随机选取部分或者全部数据块，并使用对应的摘要算法验证数据集完整性。

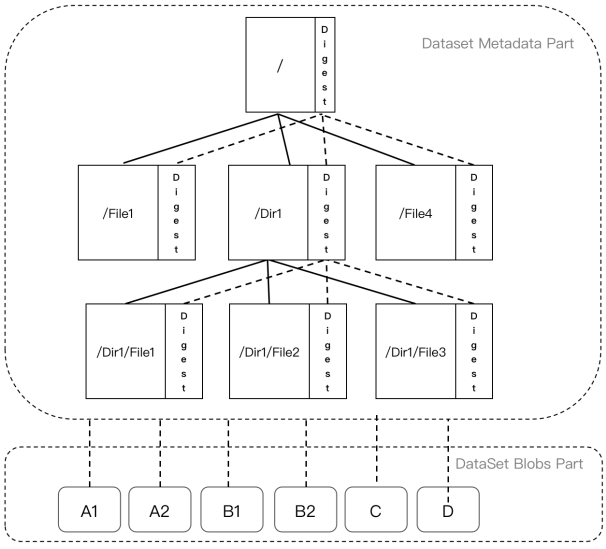

图 11 智算网络数据集结构

多版本数据集结构：共享数据块部分则按块分隔数据集原文件并拼接在一起，负责存储当前版本数据集的实际字节内容。这样的分离式数据结构可以将高频访问的元数据存储在内存和高速存储设备上，将可能占用庞大存储空间的共享数据块文件存储在成本较低的持久化存储中，提高

系统存储效率。

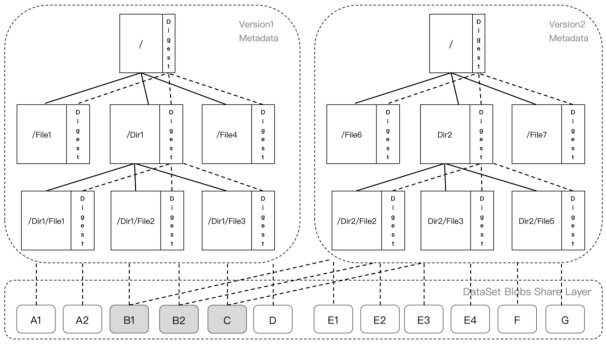

图 12 智算网络多版本数据集结构

6.2.3 架构

平台是一个支持异构多中心数据统一存储与调度的数据存储系统，旨在为异构多中心的计算任务提供全局统一的存储管理和高效的数据调度解决方案，它提供了统一存储管理、数据集全局感知、数据集上传、数据集下载、缓存管理和数据集调度等功能，整体架构如下图所示。系统采用微服务架构，由系统管理器、云际数据集管理器、统一代理服务器和统一存储应用接口四部分组织。系统管理器、数据集调度器和统一代理服务器之间采用远程过程调用（RPC）的方式进行交互与监控。

6.3 智算中心调试训练部署一站式平台

智算中心调试训练部署一站式平台面向单中心的算力用户，承接用户算力订单。提供了数据、算法、镜像、模型与算力等资源管理与使用功能，方便智算中心用户一站式构建人工智能任务计算环境。同时，向智算中心管理人员提供单中心内资源管理与监控、单中心内计算任务管理与监控等功能，方便管理人员对单中心进行操作与分析。

6.3.1 特性

一站式开发：为用户提供面向 AI 计算场景的调试、训练、部署的一站式功能，通过数据管理、模型开发和模型训练等模块，打通 AI 计算全链路。而且管理方便，为平台管理者提供一站式的资源管理平台，通过资源配置、监控、权限管控等可视化工具，大大降低平台管理者的管理成本。

性能优越：提供高性能的分布式计算体验，通过多方面优化来保证各环境的流畅运行，同时通过资源调度优化与分布式计算优化，进一步提高模型训练效率。

兼容性好：平台除了支持 Sophon TPU 还支持异构硬件与异构网络，如 GPU、NPU、FPGA、IB 网、以太网等，满足各种不同的硬件集群部署需求。同时支持多种深度学习框架，如 TensorFlow、Pytorch、PaddlePaddle 等，还可以通过自定义镜像的方式支持新增框架。

6.3.2 用户功能

镜像面板：容器镜像用于打包应用程序及其依赖的操作系统环境，提供算法运行的环境。镜像管理模块包括用户镜像管理与平台管理员预置镜像管理。该模块主要功能包括：获取镜像列表，创建/查询/更新/删除镜像、获取镜像列表版本列表、创建/查询/更新/删除镜像版本、分享镜像版本到智算网络数据市场，取消分享。

算法面板：用户可向平台上传其代码文件，然后创建人工智能调试环境修改代码。算法代码调试成功并保存后，可一键启动对应人工智能训练环境，输出模型结果和训练过程日志。该模块主要功能包括：获取算法列表、创建/查询/更新/删除算法、获取算法版本列表、创建/查询/更新/删除/下载算法版本、分享算法版本到智算网络数据市场、取消分享。

数据面板：单智算中心内的数据集可分类为我的数据集、公共数据集和预置数据集。我的数据集是用户自己的数据集，公共数据集是用户分享的数据集，预置数据集是管理员上传的数据集。单中心内所有数据集都可作为智算网络数据源接入智算网络数据统一存储与管理平台。该模块主要功能包括：获取数据集列表、创建/查询/更新/删除/数据集、获取数据集版本列表、创建/查询/更新/删除/下载数据集版本、分享数据集版本到智算网络数据市场、删除分享。

调试作业：平台提供在线编程环境用来调试、运行和保存算法以支撑后续的创建训练作业。调试作业模块的支持多种在线编程环境（如 JupterLab，VsCode)，通过该模块提供创建、打开、启动、保存、停止、删除调试作业等管理功能。

训练作业：该模块支持用户创建人工智能分布式训练作业和非分布式训练作业; 且兼容多个深度学习计算框架（如 TensorFlow、PaddlePaddle、MindSpore、PyTorch 等）, 选择对应算法和操作系

统镜像即可启动人工智能训练作业；支持「训练-模型」一站式开发，训练完成后自动保存模型，方便部署调用。

模型面板：模型是算法训练产生的结果，通过管理可以帮助用户归档训练结果，并为后续的模型验证、部署和评测提供模型数据。该模块主要功能包括：获取模型列表、创建/更新/删除模型、获取模型版本列表、创建/查询/更新/删除/下载模型版本、分享模型到智算网络数据市场、取消分享。

推理服务：该模块支持将模型一键部署为在线推理服务，并可配置生成服务对外 API 文档，为用户开展大模型业务提供推理服务平台。该模块可以查询服务名称、模型名称、模型版本、服务描述、创建时间、运行时长、状态。推理服务采用弹性伸缩技术架构，当夜间无人访问时，推理服务副本个数缩小，最小缩小至 0，减少资源占用，节省用户费用，使得平台推理算力卡资源得以最高效共享和利用。

6.3.3 管理功能

多租户管理：支持多租户资源隔离，为满足不同 Team 的不同资源需求，划分不同工作空间并匹配相应的训练资源，使得 Team Leader 可以更便利地对训练资源进行管理。

平台监控：平台监控对象为智算中心集群监控和训练任务监控。Prometheus 将每个节点的 Node-Exporter 提供的监控数据收集汇聚到其时序数据库。Grafana 网页服务定时请求 Prometheus 的数据指标，可向管理员和普通用户动态地展示集群和用户正在运行的任务的指标数据的图表。

资源管理：平台资源管理对象可以分为服务器节点列表、系统资源对象、自定义资源对象、资源包对象、资源池对象。

机时管理：平台使用机时为单位进行计费，当用户创建人工智能调试环境或者训练环境时，需要扣取相应的机时。平台按照资源包价格设置单价 n 个 UNC/机时。人工智能作业计费规则如下：

作业机时 = 子任务 1 机时 + 子任务 2 机时 + ... + 子任务 n 机时

子任务机时 = 副本 1 机时 + 副本 2 机时 + ... + 副本 n 机时

任务副本机时 = 资源包使用权重 * (任务副本运行终止时间 - 任务副本运行起始时间)

预置数据管理：预置数据集是智算中心管理员上传的数据集，一般为公有数据集，为单智算中心内用户和智算网络提供初始数据源。

预置算法管理：预置算法是指智算中心管理员创建和上传的算法代码，智算中心内用户可以查询、使用、复制、下载、迁移预置算法。

预置镜像管理：预置镜像是指智算中心管理员创建的容器镜像，智算中心内用户可以查询、使用、复制、下载、迁移预置镜像。

调试作业管理：智算中心管理员可通过该模块运维智算中心内用户的调试作业，协助用户排查调试作业问题。可进行查询单个作业详情、查看作业各维度监控信息、查看作业日志和强制停止等操作。由于平台存算分离架构，平台管理员无法查看用户数据集、算法、模型等输入输出原数据。

训练作业管理：智算中心管理员可通过该模块运维智算中心内用户的训练作业，协助用户排查训练作业问题。可进行查询单个作业详情、查看作业各维度监控信息、查看作业日志和强制停止等操作。由于平台存算分离架构，平台管理员无法查看用户数据集、算法、模型等输入输出原数据。

推理服务管理：智算中心管理员可通过该模块运维智算中心内所有推理服务，运维功能包括：查询服务列表、查看服务详情和强制停止等。

6.4 人工智能数据市场

人工智能领域数据包括算法代码、容器镜像、标注数据集、各种视频、NLP 自然语言大模型等，智算网络数据集市主要功能是为 AI 算力服务买卖双方提供数据交易的平台。

6.4.1 功能

数据发布：数据提供者可以在智算网络数据集市上发布自己的数据，并对数据进行描述、标注和定价。

数据购买：数据需求者可以购买智算网络数据集市上展示的数据。购买前可以进行询价议价，交易确认和结算后可获得数据访问权限，可以在智算网络上使用和迁移该数据。

数据安全性保障：智算网络数据集市采用数据指纹、数据加密、数据备份等措施，保障数据的安全性和完整性。

数据交易监管：智算网络数据集市采用智能

合约技术，对数据交易进行监管和验证，确保交易的公平、透明和可追溯。

数据共享和流通：智算网络数据集市还支持数据的免费共享和流通，数据提供者可以将自己的数据免费共享给其他 AI 算力服务买家，从而提高数据利用率和价值。

6.4.2 架构

数据管理模块：该模块用于数据的存储、管理、搜索和浏览，包括数据分类、标签、描述等功能，该模块在架构上作为智算网络数据统一存储与管理平台的应用层。数据唯一 id 生成、数据存储介质、数据权限获取等底层功能都由智算网络数据统一存储与管理平台实现数据交易模块：该模块使用 UNC 对数据进行定价、购买、支付、确认和交易监管，模块功能包括订单管理、支付接口、数据质量评价、智能合约实现等功能。数据安全模块：该模块用于数据的加密、备份、恢复和监管，包括数据加密算法、备份策略、安全性检测等功能。该模块在架构上也作为智算网络数据统一存储与管理平台的应用层，智算网络数据统一存储与管理平台提供底层安全技术功能。数据共享模块：该模块用于数据的免费共享、权限获取、数据流通等功能。

7 性能与可用性

尤特网络关注性能与可用性，力求为用户提供高效、可扩展且安全的区块链基础设施。以下小节将从共识速度，链上爆块状态，k8s 技术，抗女巫攻击和代理人攻击和 Ranking 机制展开详细讨论，并阐述尤特网络在各方面的优势。

7.1 共识速度

尤特网络采用先进的异步共识算法，提高了共识速度并降低了确认延迟。这使得尤特网络具有更高的吞吐量，能够在短时间内处理大量交易和计算任务。而尤特网络的共识速度得益于其先进的共识算法和多层验证机制。以下详细介绍共识速度的各个阶段。

首先，通过 VRF（可验证随机函数）生成和验证随机数，并且按轮次高度确保公平且不可预测地选择预爆块矿工，为共识提供了一个坚实的基础，这个过程通常在 10 秒内完成。

接下来，在 2-3 秒内，预爆块矿工会生成一个新区块并将其提交到网络。这一过程确保了尤特网络的高效性能。

紧接着，在 5-10 秒内，新区块会被广播到整个网络，让其他矿工节点得知新区块的存在。这有助于提高网络的透明度和一致性。

在新区块广播之后，其他矿工节点将在 1-2 秒内对新区块进行验证，以确保其合法性和正确性。这是尤特网络安全性的关键环节。

最后，在整个共识过程的 30 秒内，一个新的区块诞生并被添加到尤特网络的区块链上。这表明尤特网络具有较快的共识速度，能够在短时间内处理大量交易和计算任务。

7.2 链上爆块状态

尤特网络通过链上爆块状态实现了对矿工的双重校验，确保了矿工的计算环境和算力可用性，区块链的安全性和数据一致性。通过采用难度调整算法和 VRF 技术，尤特网络能够实现公平的矿工竞争和链上状态的稳定更新。这一机制包括两个部分：计算任务可用性证明（CAT）和可信算力证明（POCI）。

计算任务可用性证明（CAT）要求矿工在生成新区块之前证明其执行环境的正确性和安全性。这样一来，只有具备完整计算环境的矿工才能参与爆块，从而防止恶意矿工提交伪造或篡改的区块。

可信算力证明（POCI）则是通过链上随机数和算力竞争来选举合适的矿工生成新区块。这一机制确保了矿工的算力可用性，以及在网络中的公平竞争。

通过这两个校验机制，尤特网络避免了大量无效的计算用于对抗拜占庭将军问题，从而将 99% 的运算能力释放出来，用于满足基于尤特的服务体系。这使得尤特网络能够在分布式训练和推理计算场景中实现大规模应用，为用户提供高效、安全、可靠的计算服务。

7.3 kubernetes 运行扩展与端侧标准组件库 USL（Utiility Standard Library）

尤特网络充分利用了 Kubernetes[29] 的优势，支持大规模运行扩展，通过提供端侧标准组件库，

帮助开发者轻松构建和部署去中心化应用，降低开发门槛。这为矿工侧提供了一套高度可用、易于部署和管理的解决方案。通过 Utility Standard Library（USL），尤特网络实现了端侧标准组件的快速集成 [32]，使得具备可验证算力芯片的矿工可以轻松启动高可用的服务集群。

USL 包含了一系列预先构建的端侧标准组件，它们可以与 Kubernetes 无缝对接，简化了部署和管理过程。这意味着矿工无需进行复杂的配置和调试，只需通过一键克隆操作，即可将尤特网络的服务集群部署到自己的硬件设备上。

尤特网络的这一特性突出了对矿工侧可用性的关注，让任何持有可验证算力芯片的矿工都能快速启动并加入到尤特生态中。这种“克隆即可用，可用即好用”的理念为矿工提供了极大的便利，降低了参与门槛，进一步促进了尤特网络的普及和发展。

7.4 抗女巫攻击和代理人攻击

女巫攻击，也称为“假冒攻击”或“伪造攻击”，是指攻击者伪造或冒充另一个用户或系统实体，以获得未经授权的访问权限或进行其他恶意行为。例如，在电子商务中，女巫攻击者可以伪造订单或交易，从而获得未经授权的商品或服务 [33]。

代理人攻击是指攻击者利用代理人或中间人来欺骗系统或用户。攻击者通过代理人来截取、篡改或重放通信数据，以获得未经授权的访问权限或其他恶意目的。如在网络环境中，攻击者通过欺骗路由器、网关或其他中间节点来截取、篡改或重放通信数据，从而获得敏感信息或进行其他恶意行为 [34]。

尤特网络采用了多重安全机制，如强化身份验证，保证多轮次的加密通信，控制访问量等有效抵御女巫攻击和代理人攻击。同时通过限制恶意矿工的权益和行为和给予严厉的惩罚和制裁，尤特网络确保了整个网络的安全和稳定。

7.5 Ranking 机制和基金会索引 miner 服务

尤特网络引入了 Ranking 机制，对矿工提供的服务进行评级和排序。Ranking 机制是指在搜索引擎、电子商务等应用中，根据一定的算法和规则对相关结果进行排序和展示的机制。其目的是为了提高用户体验，使用户可以更快速地找到所需的信息或商品 [35, 36]。

Ranking 机制通常会考虑多种因素，如关键词匹配度、页面质量、用户行为、社交信任度等，以确定每个结果的权重和排名。具体的算法和规则因应用而异，可以是基于机器学习、协同过滤、规则引擎等技术的实现。

基金会索引 miner 服务 [37] 是基于区块链技术的一种索引服务，由基金会索引团队提供。这种服务主要是为去中心化应用提供索引服务，帮助开发者更方便地实现数据查询和交互。该服务提供的索引包括基于合约的事件、基于链上数据的查询等，可以用于支持各种应用场景，如去中心化交易所、DeFi 应用等。

综上，Ranking 机制保证了矿工所提供的良好服务和用户对他们搜索的多向选择权，而基金会维护提供的 miner 服务索引，令用户根据需求和信任度选择合适的矿工。这两个机制有助于提升矿工服务质量和用户的人性化体验，双方促进了尤特网络内矿工的良性竞争，而且进一步优化尤特网络的整体性能和可用性。

8 面向未来的可能性

尤特网络作为一种创新的区块链技术，创新性地融合了加密学算法，互联网技术，数据存储技术等多种技术，具有非常广泛的发展前景和潜力。在未来，尤特网络有望实现更多突破性的技术进展，为用户提供更多优质的服务。

8.1 更多异构芯片存储芯片/指令支持

随着计算机技术的不断发展，越来越多的异构芯片应运而生，这些芯片具有各自独特的优势和特性。在未来，尤特网络将不断升级或积极拓展对这些异构芯片的支持，诸如存储芯片和更多指令集。这能进一步提高尤特网络的通用性和适应性，使其能够满足更多不同场景和用户需求。

尤特网络致力于构建一个全面、高效且安全的计算资源网络，以满足超大规模计算力需求。为实现这一目标，尤特网络将通过制定 Utility Proposal (UP) 来支持更多具有安全引擎的可验证芯片、算力存储以及网络设备的加入。有了该协议的支持，

尤特网络将从以下三个方面入手推动自设的发展。

首先，尤特网络将积极推动各类异构芯片的集成和应用，以发挥各自计算性能的优势。这些异构芯片包括大众熟悉的 CPU、GPU、TPU 等，它们具有各自独特的性能优势，可以充分满足不同场景下的计算需求。此外，尤特网络将探索与各种存储设备的深入合作，实现数据存储高效管理，以配合计算资源。这包括对传统硬盘、固态硬盘以及新型存储技术的支持，旨在提高数据存储的速度和可靠性。值得一提，尤特网络也将关注网络设备的发展，以实现计算资源的高速传输和分布式调度。这一切将有助于构建一个高度优化的计算资源网络，为用户提供更加灵活且高效的计算服务。

通过以上措施，尤特网络将发挥存算并行优势和实现激励一体化，为超大规模计算诞生的全人类网络提供坚实基础。尤特网络将继续探索和创新，推动计算资源网络的发展，为人类的科技进步贡献巨大的力量。

8.2 私有化算力服务

尤特网络还将探索私有化算力服务的可能性。这种服务将允许企业和个人在保障数据安全和隐私的前提下，利用尤特网络提供的算力资源。通过私有化算力服务，用户可以在自己的专属环境中进行计算任务，同时避免数据泄露和未经授权的访问。同时，尤特网络将提供完善的技术支持和管理工具，帮助用户轻松实现私有化算力服务的部署和运维。

尤特网络的私有化算力服务为用户提供了一种更加安全且高效的计算资源解决方案。通过私有化算力服务，在数据安全的前提下，用户可以实现大模型部署、私有大模型部署、高性能定制化伸缩、模型微调以及解决敏感数据问题。

8.2.1 高性能定制化伸缩

私有化算力服务允许用户根据自身需求灵活调整计算资源，从而实现高性能计算的优化。用户可以根据任务规模和复杂度，快速扩展或缩小计算资源，以保证计算效率和成本的平衡。

8.2.2 大模型部署与私有大模型部署

尤特网络支持用户在私有化环境中部署大型 AI 模型，确保数据安全且不泄露至中心化云服务。这意味着用户可以在自己的计算环境中独立运行和管理 AI 模型，从而满足特定场景下的计算需求。

8.2.3 模型微调

私有化算力服务支持对已部署的 AI 模型进行微调，以提高模型在特定任务和场景下的表现。用户可以基于自己的数据集和需求，对模型进行调优，以实现更高的精度和性能。

8.2.4 解决敏感数据问题

尤特网络的私有化算力服务在处理敏感数据时具有显著优势。由于数据仅在私有模型和推理任务中流通，用户无需担心数据泄露给第三方，从而确保数据的隐私和安全。

总体来看，尤特网络的私有化算力服务为用户提供了一种既安全又高效的计算资源解决方案。而通过上述的高性能定制化伸缩、大模型部署、模型微调以及解决敏感数据问题这四个环节，尤特网络也能为用户提供了一种强大且灵活的计算资源管理方式。

展望未来，尤特网络的未来发展将聚焦于以下几个方面：支持更多类型的异构芯片，扩大尤特网络的适用范围；探索私有化算力服务，提供更加安全、可靠的计算资源；持续优化网络性能和可用性，为用户提供更高效的服务；深入研究和应用区块链技术，推动尤特网络与其他领域的结合和创新。尤特网络将继续秉承创新精神，为用户提供优质的服务，推动区块链技术的发展和应用。

9 总结

尤特网络是一个基于区块链技术构建的去中心化计算服务平台，旨在解决现有集中式计算资源的局限性，为用户提供更高效、安全且可扩展的计算服务。尤特网络的主要技术特点包括：

高效的共识机制：可信算力证明 (Proof of Computation Integrity, PoCI) 打破以往算力耗费的模式，通过芯片拥有权和 VRF 实现随机数生成区间和验证，结合多签验证和基于 round 的预爆块选举，实现了快速的共识和区块生成。同时，RPOI (Rent Proof of Instructions) 也保证了算力的有效租赁和高效使用，促进分布式算力网络的生成。

链上爆块状态：通过计算任务可用性证明（CAT）和可信算力证明（POCI）在链上计算完成对爆块者的双重校验，避免了恶意访问和选举

爆块者的公平性和竞争性，确保有效的计算资源分配，同时降低拜占庭将军问题的影响。

默克尔状态机链外数据校验：默克尔树(Merkle Tree) 是尤特网络中为了缓解链上数据压力而采用的数据存储结构，通过 Rollup 将链外数据传输至尤特网络，启动默克尔状态机（Merkle State Machine, MSM）将这些数据的哈希值或默克尔树根节点有序存储在区块链上，同时实现快速验证。该技术降低了区块链网络的存储压力又同时保证了链外数据的安全和完整性。

虚拟化与隔离执行环境：基于 Arm 框架驱动和显存虚拟化支持，令 TPU 芯片具备计算任务的灵活共享，提高了提高整体的计算效率。执行环境与容器技术相结合使其实现高度隔离的执行环境，保证计算服务的安全性。

多种执行虚拟机支持：包括 Solidity VM、Kubernetes 集成、WASM 集成以及 PyTorch 外联运算支持，构建了一个功能丰富且兼容性强的计算平台。

尤特人工智能算力网络：尤特智算平台层主要由四个智算中心调试训练部署一站式平台、智算网络任务统一调度与管理平台、智算网络数据统一存储与管理平台、智算网络数据市场四个子平台构成。目标旨在构建一个创新性的算力资源易接入、任务高效统一调度、用户方便廉价调用且具有备可持续发展运营模式和机制的 AI 算力基础设施。

尤特网络的经济学模型：主要围绕代币循环与计算生态展开，包括挖矿机制、代币减产、Gas 与 Burn Fee、机时租赁订单和推理部署订单。通过这一经济模型，尤特网络实现了公平的激励分配和合理的资源利用。

在面向未来的可能性方面，尤特网络通过制定 UP（Utility Proposal）支持更多安全引擎可验证异构芯片、存储芯片以及指令支持，算力存储，指令集和网络设备的发展，构建一个充分发挥存储和计算双优势的激励网络；并实现私有化算力服务，从而为用户提供更加多样化的计算服务。

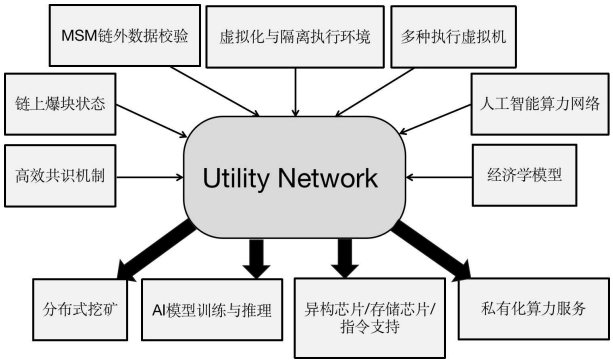

图 13 总结 Utility Newtork 的技术构成，功能和未来展望

综上，尤特网络通过高效的共识机制、安全的计算环境、丰富的执行虚拟机支持以及灵活的经济模型，为用户提供了一个去中心化、高效且安全的计算服务平台。尤特网络的发展将极大地推动分布式计算的发展，为人类的科技进步做出重要贡献。

10 参考文献

[1] Benet, J. (2014). IPFS - Content Addressed, Versioned, P2P File System. [arXiv:1407.3561v1] [2] Protocol Labs. (2021). Filecoin: A Decentralized Storage Network. Filecoin Whitepaper. Retrieved from https://filecoin.io/filecoin.pdf

[3] Nakamoto, S. (2008). Bitcoin: A Peer-to-Peer Electronic Cash System. Retrieved from https://bitcoin.org/bitcoin.pdf

[4] Buterin, V., et al. (2014). A Next-Generation Smart Contract and Decentralized Application Platform. Ethereum Whitepaper. Retrieved from https://ethereum.org/whitepaper

[5] Wood, G. (2014). Ethereum: A Secure Decentralised Generalised Transaction Ledger. Ethereum Yellow Paper. Retrieved from https://ethereum.github.io/yellowpaper/paper.pdf

[6] Boneh, D., et al. (2013). Public Key Cryptography for the Financial Services Industry: The Elliptic Curve Digital Signature Algorithm (ECDSA). ANSI X9.62-2005. Retrieved from https://www.secg.org/sec2-v2.pdf

[7] Merkle, R. C. (1987). A Digital Signature Based on a Conventional Encryption Function. Advances in

Cryptology –CRYPTO ' 87. Lecture Notes in Computer Science, vol 293. Springer, Berlin, Heidelberg. https://doi.org/10.1007/3-540-48184-2-32

[8] Lamport, L. (1981). Password Authentication with Insecure Communication. Communications of the ACM, 24(11), 770-772. https://doi.org/10.1145/358790.358797

[9] Intel Corporation. (2016). Intel® Software Guard Extensions (Intel® SGX). Retrieved from https://software.intel.com/en-us/sgx

[10] AMD. (2013). AMD64 Architecture Programmer's Manual Volume 2: System Programming. Retrieved from https://www.amd.com/system/files/TechDocs/24593.pdf

[11] ARM Limited. (2018). ARM Architecture Reference Manual ARMv8, for ARMv8-A architecture profile. Retrieved from https://developer.arm.com/documentation/ddi0487/latest/

[12] Boneh, D., Gentry, C., Waters, B. (2005). Collusion Resistant Broadcast Encryption With Short Ciphertexts and Private Keys. Advances in Cryptology - CRYPTO 2005, 258-275. https://doi.org/10.1007/11535218$_1$6

[13] Parno, B., et al. (2013). Pinocchio: Nearly Practical Verifiable Computation. 2013 IEEE Symposium on Security and Privacy. https://doi.org/10.1109/SP.2013.44

[14] Dwork, C., Naor, M. (1993). Pricing via Processing or Combatting Junk Mail. Advances in Cryptology –CRYPTO ' 92. Lecture Notes in Computer Science, vol 740. Springer, Berlin, Heidelberg. https://doi.org/10.1007/3-540-48071-4$_2$9

[15] Ben-Sasson, E., et al. (2014). Zerocash: Decentralized Anonymous Payments from Bitcoin. 2014 IEEE Symposium on Security and Privacy. https://doi.org/10.1109/SP.2014.36

[16] Szabo, N. (1997). Formalizingand Securing Relationships on Public Networks. First Monday, 2(9). https://doi.org/10.5210/fm.v2i9.548

[17] Micali, S., Rabin, M., Vadhan, S. (1999). Verifiable Random Functions. 40th Annual Symposium on Foundations of Computer Science, 120-130. https://doi.org/10.1109/SFFCS.1999.814594

[18] Benet, J., Greco, N. (2017). Proof of Replication. Filecoin Research. Retrieved from https://filecoin.io/proof-of-replication.pdf

[19] Ben-Or, M., et al. (1988). Simple Efficient Asynchronous Byzantine Agreement with Optimal Resilience. Proceedings of the first annual ACM symposium on Principles of Distributed Computing, 42-52. https://doi.org/10.1145/62546.62550

[20] Luu, L., et al. (2016). A Secure Sharding Protocol for Open Blockchains. Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, 17-30. https://doi.org/10.1145/2976749.2978389

[21] Kalodner, H., et al. (2018). An Empirical Study of Namecoin and Lessons for Decentralized Namespace Design. WEIS 2015: Workshop on the Economics of Information Security. Retrieved from https://weis2015.econinfosec.org/wp-content/uploads/sites/5/2015/05/WEIS 2015 submission 72.pdf

[22] Back, A. (2002). Hashcash - A Denial of Service Counter-Measure. Retrieved from http://www.hashcash.org/papers/hashcash.pdf

[23] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... Polosukhin, I. (2017). Attention is all you need. In Advances in neural information processing systems (pp. 5998-6008).

[24] Daian, P., et al. (2020). Flash Boys 2.0: Frontrunning, Transaction Reordering, and Consensus Instability in Decentralized Exchanges. 2020 IEEE Symposium on Security and Privacy. https://doi.org/10.1109/SP40000.2020.00047

[25] Eyal, I., Sirer, E. G. (2018). Majority is not Enough: Bitcoin Mining is Vulnerable. International Conference on Financial Cryptography and Data Security, 436-454. https://doi.org/10.1007/978-3-662-45472-5$_2$8

[26] Ren, X., Yu, H. (2020). Blockchain-Based Intelligent Transportation Systems: A Comprehensive Review. IEEE Transactions on Intelligent Transportation Systems, 21(5), 18

[27] Chen, Y., Zhang, Y., Xu, Z., Huang, J. (2019). Secure execution of blockchain smart contracts based on virtual machine. IEEE Access, 7, 61933-61943. https://doi.org/10.1109/ACCESS.2019.2918648

[28] Chia, M. P., Gan, C. Y., Low, K. L. (2018). Securing blockchain execution environment through isolation and hardware-enhanced attestation. IEEE Transactions on Dependable and Secure Computing, 15(4), 570-583. https://doi.org/10.1109/TDSC.2016.2615552

[29] Sun, Z., Cui, Y., Zhang, Y., Yu, Y. (2020). Containerized deployment of Ethereum smart contract based on Kubernetes. In Proceedings of the 2020 IEEE International Conference on Smart Internet of Things (SmartIoT) (pp. 182-187). https://doi.org/10.1109/SmartIoT49753.2020.00039

[30] Tian, Y., Han, G., Chen, S., Chen, Y., Wu, D., Wei, Y. (2021). A WASM-based execution model for blockchain smart contracts. In Proceedings of the 2021 IEEE International Conference on Blockchain (Blockchain) (pp. 166-173). https://doi.org/10.1109/Blockchain50986.2021.00028

[31] Zhou, Z., Yan, X., Guan, S., Zhang, J., Zhu, X. (2021). PyTorch on the blockchain: Integrating deep learning and smart contracts. IEEE Transactions on Services Computing, 14(1), 144-157. https://doi.org/10.1109/TSC.2019.2935417

[32] Chen, X., Chen, G., Wu, H., Wang, Y. (2018). Design and implementation of IoT platform based on USL. In Proceedings of the 2018 5th International Conference on Systems and Informatics (pp. 558-562). https://doi.org/10.1109/ICSAI.2018.8574167

[33] Diffie, W., Hellman, M. (1976). On the security of public key protocols. IEEE Transactions on Information Theory, 22(6), 644-654. https://doi.org/10.1109/TIT.1976.1055638

[34] Maurer, U. (1993). Secure communication over insecure channels. Advances in Cryptology —EUROCRYPT'93, 1-14. https://doi.org/10.1007/3-540-48285-7$_1$

[35] Yang, J., Ma, Y., Xu, X., Wang, X. (2020). Research on a blockchain-based ranking mechanism in academic community. In Proceedings of the 2020 International Conference on Intelligent Sustainable Systems (ISS) (pp. 55-60). https://doi.org/10.1109/ISS49245.2020.9240599

[36] Wu, J., Xie, X. (2021). Design and implementation of search engine ranking algorithm based on blockchain. In Proceedings of the 2021 IEEE 4th International Conference on Information Systems and Computer Aided Education (ICISCAE) (pp. 556-560). https://doi.org/10.1109/ICISCAE53270.2021.00098

[37] Li, X., Li, C., Li, W., Li, Y., Li, G. (2020). Research on blockchain technology based on the Foundation index miner service. In Proceedings of the 2020 3rd International Conference on Computer Communication and Informatics (ICCCI) (pp. 1-6). https://doi.org/10.1109/CCCI48366.2020.9117489