

Irys: 검증 가능한 스토리지와 온체인 연산을 위한 프로그래밍 가능한 데이터체인

Josh Benaron, Dan MacDonald, and Jesse Cruz Wright

공개: 본 백서에 기술된 정보는 예비 정보이며 언제든지 변경될 수 있습니다. 또한, 본 백서는 "미래 예측 진술"을 포함할 수 있습니다. 본 백서의 어떠한 내용도 토큰의 판매 제안이나 구매 제안의 권유가 아닙니다.

초록

본 백서는 스마트 컨트랙트가 핫 액세스 지연 시간으로 온체인 바이트를 읽고 처리할 수 있도록 고성능 스토리지, 데이터 가용성, 그리고 EVM 호환 실행 환경을 통합하여 데이터를 프로그래밍 가능하게 만드는 레이어-1 데이터체인 Irys 를 제시합니다. 하이브리드 작업증명 및 지분증명 합의는 블록 생성을 하드 디스크 드라이브 기반의, 검증 가능한 지연 함수 속도에 맞춘 효율적인 샘플링에 연결합니다; 채굴자들은 매초 결정론적인 200 MiB 순차 읽기를 수행하며 슬래싱될 수 있어, 낮은 메시징 오버헤드와 빠른 포함으로 지속적인 스토리지 증명을 생성합니다. 매트릭스 패킹은 채굴자의 주소를 각 256 KiB 청크에 인코딩하여, 고유한 복제본을 강제하고 원격 채굴 공격 지점을 차단합니다. 다중 원장 아키텍처는 임의의 보존 기간을 지원합니다; 데이터는 단기 원장에 진입하고 인그레스 증명이 임계값에 도달하면 영구로 승격되어, 수수료 시장을 분리하고 비용을 안정화합니다. IrysVM 은 청크 범위를 컨트랙트로 스트리밍하는 프리컴파일로 EVM 을 확장하여, 데이터셋이 일급 프로그램 입력인 프로토콜 내 라이선싱, 로열티, 그리고 검증 가능한 AI 워크플로우를 가능하게 합니다. 일반적인 1 Gbps 하드웨어에서, 처리량은 채굴자 수에 따라 확장되며 쓰기 지연 시간은 드라이브에 의해 제한됩니다; 가격 책정은 HDD 의 원가 경제성을 목표로 합니다. 이러한 기본 요소들은 함께 가용성, 순서, 그리고 내구성 있는 스토리지를 증명하는 단일 체인을 제공하며, 동시에 데이터를 정적 페이로드에서 실행 가능한 자산으로 전환합니다.

1 서론

데이터체인은 내결함성을 갖춘 복제된 데이터 및 실행 머신의 구현체입니다. 현재의 스토리지 블록체인들은 데이터를 정적 페이로드로 취급하며, 연산 내에서의 데이터 가용성, 검증 가능성, 그리고 사용성에 대해 약한 가정을 합니다. 각 노드는 일반적으로 데이터를 고립된 상태로 검증하며, 데이터가 존재하고, 활발히 저장되며, 또는 스마트 컨트랙트에 의해 결정론적으로 처리될 수 있음을 증명할 통합된 메커니즘이 없습니다.

검증 가능한 데이터 레이어의 부재는 스토리지가 탈중앙화될 수는 있지만, 실행은 여전히 분리되어 있어, 진정한 온체인 애플리케이션이 대규모 데이터셋에서 작동하는 것을 막는다는 것을 의미합니다.

Irys 는 스토리지, 접근, 그리고 가용성에 대한 증명을 합의 프로세스에 직접 인코딩하도록 설계된 검증 가능한 데이터 아키텍처를 도입합니다. 각 블록은 트랜잭션의 순서를 정할 뿐만 아니라, 데이터의 모든 파티션이 샘플링되었고 접근 가능하게 유지됨을 암호학적으로 증명합니다. 이는 원장을 프로그래밍 가능한 데이터 기반으로 변환하며, 여기서 상태 전이와 데이터 증명이 동일한 무신뢰 환경 내에서 검증됩니다. Irys 네트워크의 모든 노드는 외부의 가정이나 중개자 없이 온체인에 기록된 데이터의 검증 가능한 지속성과 접근성에 의존할 수 있을 것으로 예상됩니다.

2 개요

본 백서의 나머지 부분은 다음과 같이 구성됩니다. 섹션 3 은 상위 수준 아키텍처와 수직적 통합 모델을 제시합니다. 섹션 4 는 블록, 트랜잭션 레인, 그리고 다중 원장 파티션 생명주기를 정의합니다. 섹션 5 는 PoW/S 합의를 소개하며, 효율적인 샘플링과 VDF 읽기 제한기를 핵심 하위 섹션으로 강조합니다. 섹션 6 은 데이터 트랜잭션 생명주기를 설명합니다. 섹션 7 은 IrysVM 과 프로그래밍 가능한 데이터를 상술합니다. 섹션 8 은 토크노믹스를 상세히 다룹니다. 섹션 9 는 기간 및 영구 스토리지를 위한 수수료 모델을 공식화합니다. 섹션 10 은 에포크 경계 처리를 정의합니다. 섹션 11 은 프로그래밍 가능한 데이터 L2 와 더 빠른 블록/완결성에 대한 향후 작업을 개략적으로 설명합니다.

3 고수준 아키텍처

Irys 는 통합 체인으로, 스토리지와 실행을 단일 체인에 통합하여 사용자가 훨씬 저렴한 비용으로 온체인에 데이터를 저장하고 데이터 위에 온체인 애플리케이션 로직을 구축할 수 있게 합니다.

Irys 를 설계할 때 우리의 초점은 수직적 통합, 즉 개발자가 앱을 구축하기 위해 Irys 에만 의존할 수 있는 능력이었습니니다. 우리는 개발자들이 앱을 구축하기 위해 100 가지의 다른 인프라를 사용해야 하는 것을 관찰했으며, Irys 는 스택의 모든 부분을 수직적으로 통합하는 것을 목표로 합니다.

아래 다이어그램은 Irys 가 고유한 속성을 가질 수 있게 하는 Irys 내의 프로세스를 개략적으로 설명합니다:

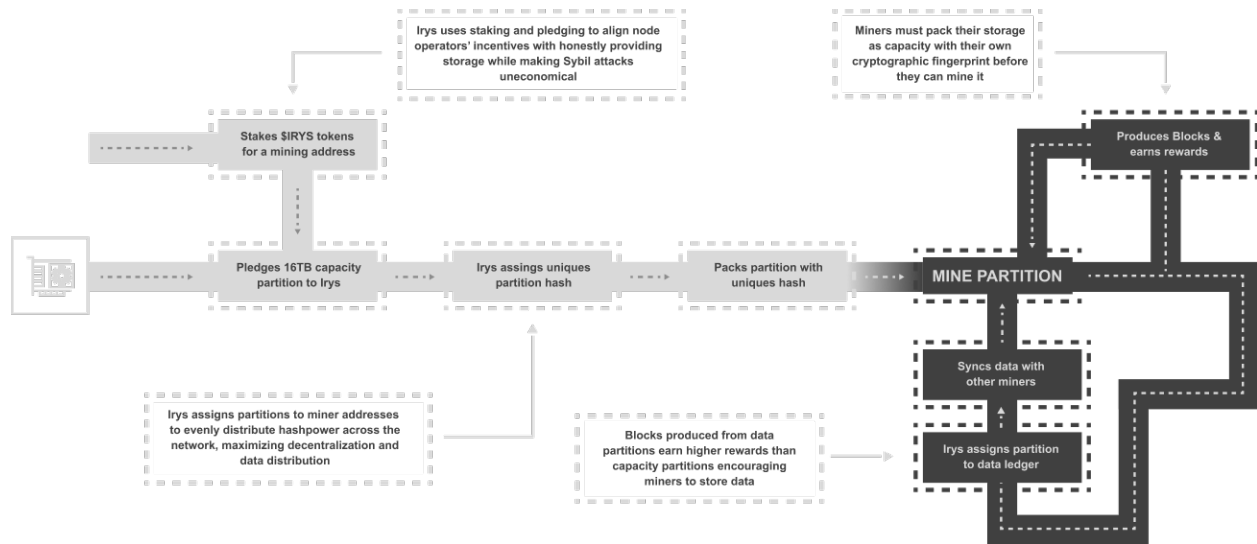


그림 1: 채굴 프로세스

Irys 는 고성능 데이터와 강력한 검증을 결합하여 온체인 데이터를 위한 네이티브 실행 환경을 가능하게 합니다. 전통적인 스토리지 프로토콜은 이 두 가지 측면의 균형을 맞추기 어려워 동일한 체인에서 실행과 스토리지를 관리하기 어렵게 만들며, 이는 데이터체인에서 온체인 앱을 구축하는 것이 지금까지 불가능했음을 의미합니다. Irys 는 작업증명을 스테이킹 및 슬래싱 메커니즘과 결합하여 이러한 상충 관계를 극복합니다. 이를 통해 Irys 는 모든 형태의 데이터를 효율적으로 처리하는 동시에 완전한 온체인 애플리케이션을 위한 강력한 보안을 갖출 수 있습니다. 우리는 단일 프로토콜에서 보안 스토리지와 네이티브 실행의 이러한 통합을 프로그래밍 가능한 데이터라고 부릅니다.

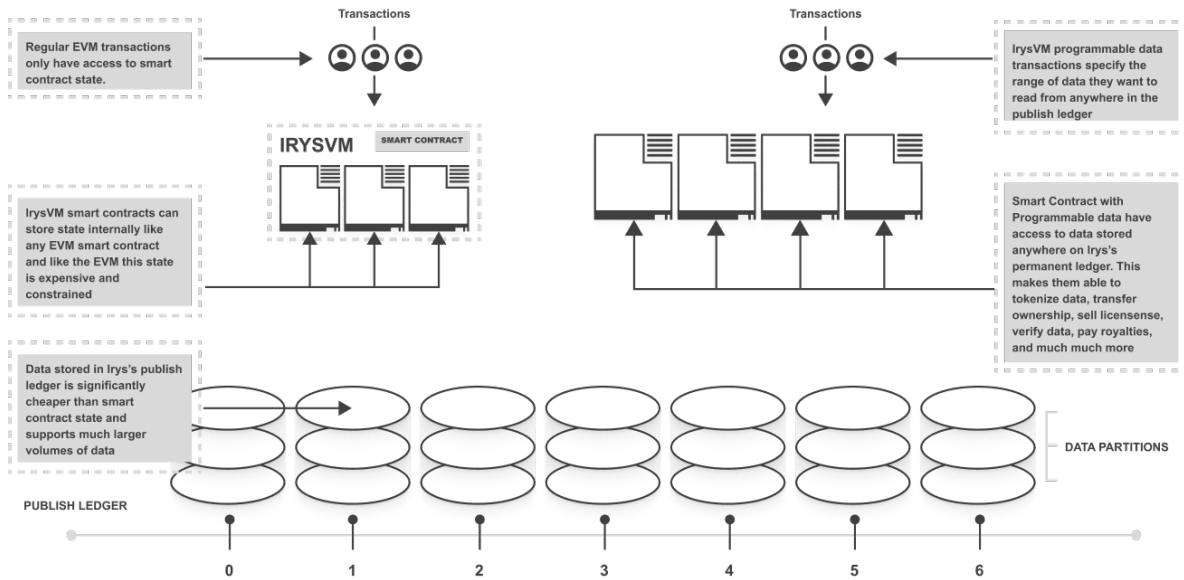


그림 2: 프로그래밍 가능한 데이터

Irys 가상머신은 프로그래밍 가능한 데이터, 즉 Irys에 저장된 대량의 데이터에 접근할 수 있는 능력을 가능하게 합니다. 이는 데이터를 정적인 상품에서 스마트 계약에 의해 활용되고 조작될 수 있는 유용한 자산으로 변환시킵니다.

4 블록과 트랜잭션

트랜잭션에는 두 가지 유형이 있습니다:

- 데이터 트랜잭션: 네트워크에 데이터를 저장하는 트랜잭션
 - 실행 트랜잭션: Irys 가상머신을 통해 스마트 계약과 상호작용하는 트랜잭션.
- 여기에는 프로그래밍 가능한 데이터 트랜잭션이 포함됩니다.

각 블록은 별도의 블록 레인에 두 세트의 트랜잭션을 포함합니다. 블록 구조는 아래에서 볼 수 있습니다:

필드명	설명
block_hash	블록 식별자
height	블록 높이
diff	현재 블록을 생성하는 데 사용된 난이도 임계값

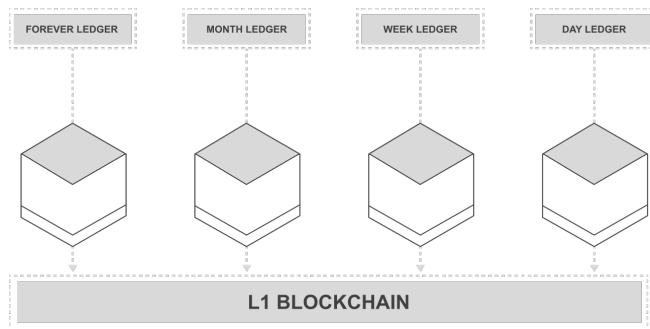
cumulative_diff	이 블록을 포함하여 과거 블록을 생성하기 위해 네트워크가 계산한 평균 해시 수의 합
-----------------	--

last_diff_timestamp	마지막 난이도 조정의 유닉스 에포크 이후 타임스탬프(밀리초)
solution_hash	블록의 솔루션 해시
previous_solution_hash	체인의 이전 블록 솔루션 해시
last_epoch_hash	마지막 에포크 블록의 솔루션 해시
chunk_hash	접근증명 청크 (인코딩되지 않음) 바이트의 SHA-256 해시
previous_block_hash	이전 블록 식별자
previous_cumulative_diff	이전 블록의 누적 난이도
poa	리콜 청크-증명
reward_address	블록 보상이 전송되어야 하는 주소
miner_address	블록 생성자의 주소 - 블록 해시/서명 및 접근증명 청크 검증에 사용됨
signature	블록 서명
timestamp	블록이 발견/생성되었을 때의 유닉스 에포크 이후 타임스탬프(밀리초)
ledgers	활성 데이터 원장 각각에 대한 트랜잭션 원장 목록
evm_block_hash	이더리움 가상 머신 블록 해시
vdf_limiter_info	검증 가능한 지연 함수 리미터에 대한 정보

Irys 는 블록 레인을 사용하여 실행 레인이 혼잡할 때에도 사용자가 항상 데이터 트랜잭션을 제출할 수 있도록 보장합니다. 이를 통해 데이터 트랜잭션은 안정적인 가격을 가질 수 있습니다.

4.1 파티션 생명주기 (용량 및 데이터, 다중 원장)

Irys 는 데이터가 서로 다른 기간 사이를 원활하게 전환할 수 있는 다중 리더 시스템을 도입합니다. 이는 사용자가 효과적으로 어떤 기간 동안이든(예: 1 일, 1 주, 1 년 또는 영구) 데이터를 저장할 수 있게 합니다. 또한 데이터를 장기 원장으로 "승격"시키는 내장 메커니즘도 있습니다. 예를 들어, 사용자가 영구 원장에 데이터 저장을 요청하면, 데이터는 5 일 원장에 들어가고 네트워크에 의해 검증된 후 영구 원장으로 전환됩니다.



원장은 공유된 속성(일반적으로 기간)을 가진 데이터의 모음입니다. 즉, 원장 0 의 모든 데이터는 5 일 동안 저장됩니다.

4.2 파티션 생명주기

Irys 의 원장은 16TB 파티션으로 나뉩니다. 이를 통해 채굴자들은 저렴하게 HDD 를 사용할 수 있으며 SSD 에 의해 경쟁에서 밀려나지 않습니다.

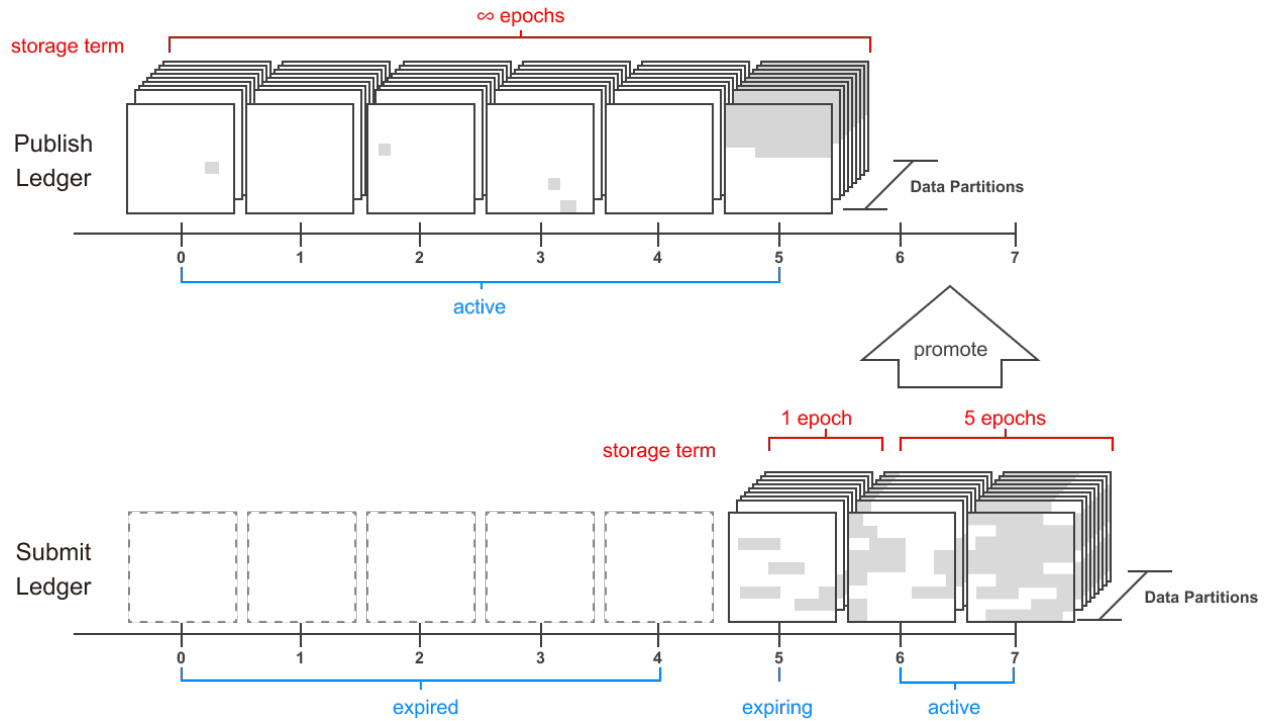


그림 4: 원장 승격

Irys 는 업로드된 데이터 양을 측정하고 이를 사용하여 대기 상태에 필요한 파티션 수를 예측합니다. 이러한 대기 파티션에는 (아직) 데이터가 포함되어 있지 않으며 용량 파티션이라고 합니다.

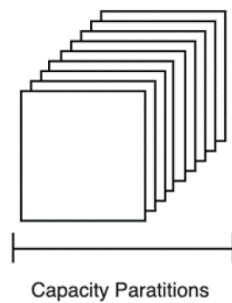


그림 5: 용량 파티션

1 단계: 파티션 서약

채굴자들은 새로운 파티션을 온라인 상태로 만들 의사를 나타내는 서약 트랜잭션을 네트워크에 게시합니다. 그런 다음 프로토콜은 청구되지 않은 용량 파티션을 서약한

채굴자들에게 무작위로 할당합니다. 용량 파티션 할당을 받지 못한 채굴자들은 서약금을 환불받습니다.

2 단계: 파티션 패킹

용량 파티션을 할당받으면, 채굴자는 Irys 의 매트릭스 패킹 방식을 사용하여 이를 패킹합니다. 이 과정은 채굴자의 지문을 파티션에 완전히 인코딩합니다.

3 단계: 파티션 채굴

용량 파티션이 패킹되면, 프로토콜은 매초 채굴자에게 무작위 200MB 순차 읽기를 할당합니다. 채굴자는 이 200MB 읽기를 가져와 800 개의 256KiB 청크로 나누고, 각 청크를 해싱하여 채굴 솔루션을 찾습니다. 채굴자가 운 좋게 채굴 솔루션을 찾으면, 새로운 블록을 생성하고 이를 네트워크에 알릴 권리를 얻어 보상을 받게 됩니다.

4 단계: 원장 할당

Irys 의 데이터 원장 중 하나에 더 많은 스토리지 용량을 추가할 때가 되면, 프로토콜은 활발하게 채굴되고 있는 용량 파티션을 무작위로 선택합니다. 이 무작위성은 블록 채굴에 사용된 파티션에 가중치를 두어, 더 오래 참여하고 채굴에 효과적이었던 채굴자들에게 보상합니다.

데이터 원장에 할당되고 이를 채굴하는 것은 빈 용량 파티션을 채굴하는 것보다 더 높은 보상을 가지므로, 채굴자들은 용량을 효율적으로 채굴함으로써 자신의 능력을 입증하도록 동기 부여됩니다.

5 단계: 파티션 이탈

파티션이 네트워크를 떠나는 방법에는 질서 있는 이탈과 무질서한 이탈 두 가지가 있습니다.

4.2.1 질서 있는 이탈

1. 채굴자는 파티션을 서약 해제하고 처음에 스테이킹했던 약정금을 회수하기 위해 트랜잭션을 게시합니다.
2. 프로토콜이 오프라인으로 전환되는 데이터를 동기화하기 위해 다른 용량 파티션을 할당하면 타임아웃 기간이 시작됩니다.
3. 타임아웃이 지나면, 이탈하는 채굴자는 스테이킹된 약정금을 회수하고 파티션을 제거할 수 있습니다.

4. 채굴자가 타임아웃이 지나기 전에 오프라인 상태가 되면, 서약한 본드를 잃을 위험이 있습니다.

4.2.2 무질서한 이탈

1. 채굴자가 네트워크에 대해 적대적인 행동(예: 블록 이중 서명)에 관여합니다.
2. 프로토콜은 그들의 스테이킹된 약정금을 가져갑니다.
3. 프로토콜은 적대적인 채굴자가 남긴 공백을 채우기 위해 새로운 용량 파티션을 할당합니다.

이는 네트워크가 새로운 용량 파티션을 신속하게 할당해야 하므로 바람직하지 않은 이탈입니다. 이 때문에 Irys 는 채굴자들이 프로토콜에 참여하고 보상을 얻기 위해 채굴 주소에 토큰을 스테이킹하여 상당한 선행 투자를 하도록 요구합니다.

4.3 매트릭스 패킹

패킹은 채굴자들이 자신의 스테이킹 주소로 데이터를 인코딩하는 과정을 설명합니다. 이는 그들이 단일 원격 복사본을 채굴하는 것(데이터체인에서 흔한 공격 벡터)이 아니라 데이터의 고유한 복제본을 저장하고 있음을 증명합니다.

Irys 는 검증 가능 지연 함수를 사용하여 채굴자의 주소를 그들이 저장하는 각 256KiB 청크에 인코딩하는 매트릭스 패킹을 도입합니다. 이는 데이터의 각 비트에 대해 충분한 계산 비용을 제공하여, 모든 적대적인 채굴 패턴을 수익성 없게 만듭니다. 그 과정은 아래 이미지에 자세히 설명되어 있습니다:

4.3.1 1 단계 - 순차 해싱

1. **mining_address** , **partition_hash** , 그리고 **chunk_offset** 은 SHA-256 으로 해시되어 **seed_hash** 가 됩니다.
2. **seed_hash** 는 SHA-256 해시되어 초기 **segment** 를 형성합니다.
3. **segment** 바이트는 빈 청크의 시작 부분에 추가됩니다.
4. **segment** 는 다음 SHA-256 을 위한 입력으로도 사용됩니다.
5. 결과로 나온 **segment** 는 이전 **segment** 의 바이트 뒤에 추가됩니다.
6. 4 단계와 5 단계는 빈 청크가 **segment** 들로 채워질 때까지 이전 **segment** 를 사용하여 반복됩니다.

7.

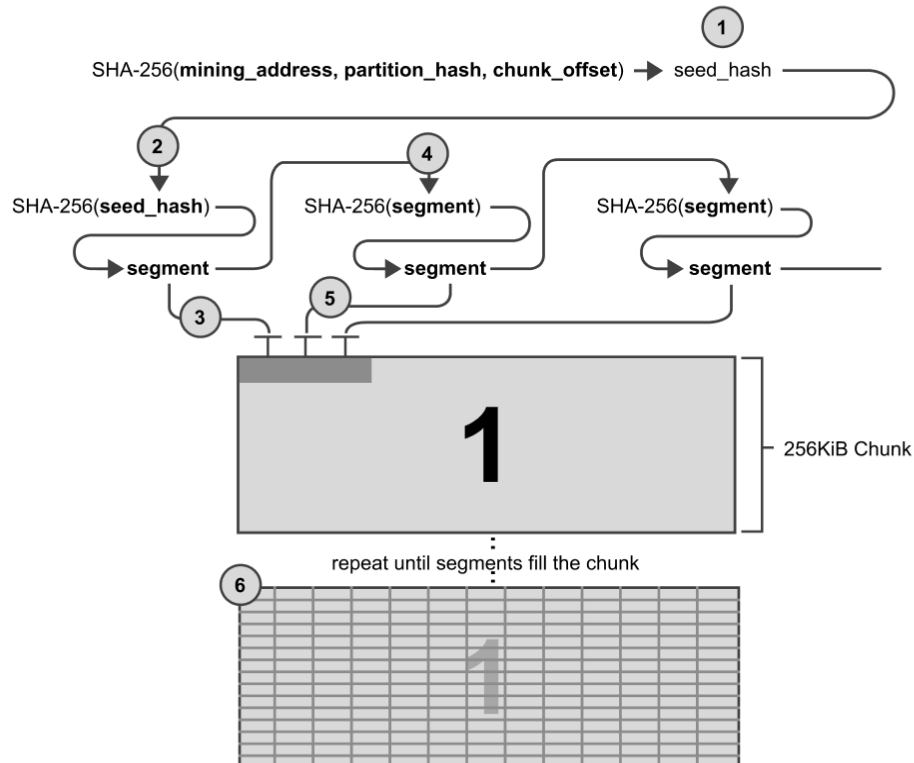


그림 6: 순차 해싱

순차 해싱은 각 segment 가 이전 segment 에 의존하기 때문에 싱글 코어 청크 패킹을 보장하며, 최소 패킹 시간을 설정합니다. 이 방법은 (로컬에서 패킹을 재생성하고 제공된 패킹된 청크와 비교함으로써) 다른 채굴자들에 의해 결정론적으로 검증 가능한 데이터가 없는 "용량" 청크의 채굴을 허용합니다.

32 바이트 segment 와 256KiB 청크로, 순차 해싱의 원 패스는 8,192 개의 해시를 필요로 합니다. 문제는 Ryzen 5900X 가 초당 약 1,500 만 SHA-256 해시를 연산할 수 있어, 이 정도의 패킹 지연이 1 초의 아주 작은 부분만을 차지하게 만든다는 것입니다. 적대적 채굴자들이 "온디맨드"로 패킹하는 것을 막기 위해, 우리는 패킹 길이를 약 1500ms 정도로 늘려야 할 것입니다.

4.3.2 1 단계 - 순차 해싱 + 파라미터화된 패킹 시간

청크 패킹 시간을 늘리기 위해, 순차 해싱은 청크의 마지막 segment 에서 첫 번째 segment 로 순환하며 덮어쓸 수 있습니다. 이 과정은 원하는 패킹 지속 시간에 도달할 때까지 여러 패스 동안 반복될 수 있습니다.

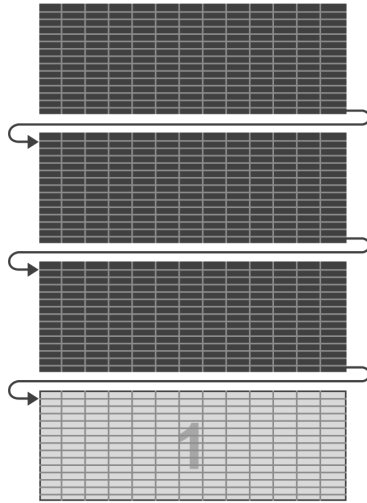


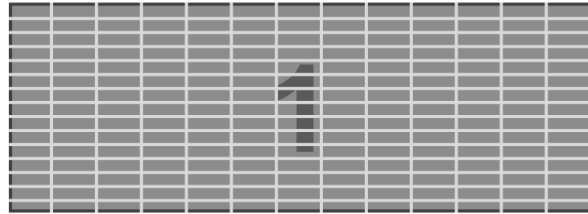
그림 7: 파라미터화된 패킹 시간

청크의 반복적인 순차 해싱은 segment 들의 레이어를 생성하며, 최종 레이어는 청크에 저장됩니다. 이 프로세스는 특정 패킹 시간을 실현하는 데는 성공하지만, 적대적 채굴자들에게 잠재적인 취약점을 노출하기도 합니다.

4.3.3 빠른 패킹 취약점



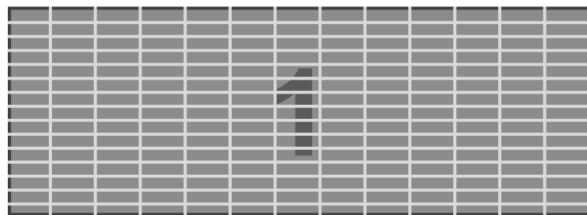
채굴자는 결과로 나온 패킹 레이어의 첫 번째 segment 만 저장하고 나머지는 버립니다.



청크를 패킹해야 할 때, 그들은 이 segment 들을 사용하여 (병렬로) 모든 이전 패킹 레이어를 연산하지 않고도 최종 패킹 레이어를 재생산할 수 있습니다.

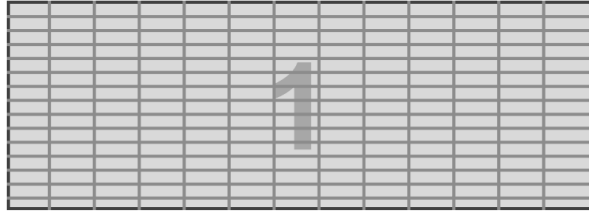


Miner then stores only the first segment of the resulting packing layer and discards the rest.



When they need to pack a chunk, they can use these segments to reproduce the final packing layer without computing all previous packing layers (in parallel)





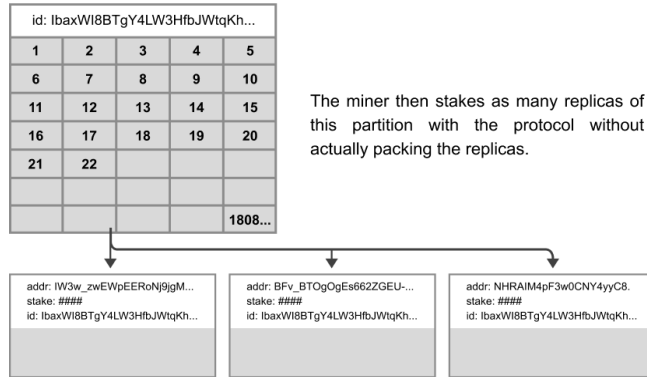
만약 한 채굴자가 동일한 파티션의 복제본에 대해 여러 채굴 주소를 스테이킹하기를 원한다면, 그들은 이 전략을 사용하여 각 스테이킹된 주소에 대해 고유한 파티션을 저장하는 것을 피할 수 있습니다. 대신 그들은 패킹되지 않은 파티션을 저장하고, 그 파티션에 스테이킹한 각 채굴 주소에 대해 "온디맨드"로 청크를 빠르게 패킹할 수 있습니다.

이는 프로토콜의 합의 규칙을 만족시키기 위해 스토리지 대신 컴퓨팅을 사용하는 예시이며, (파티션 데이터를 확실하게 복제한다는 프로토콜의 목표를 달성하지 못하는) 변질된 채굴 전략으로 간주됩니다.

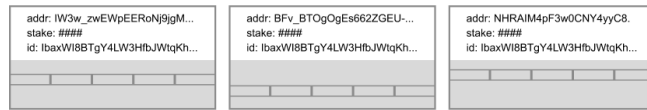
4.3.4 적대적 채굴

왜 채굴자가 "온디맨드" 패킹을 사용하기를 원할까요?

채굴자들은 가장 수익성 높은 채굴 전략을 사용하도록 경제적으로 인센티브를 받습니다. 저렴한 에너지와 컴퓨팅 풀(GPU 또는 CPU)에 접근할 수 있는 채굴자들에게는, 스토리지 기반 전략의 스토리지 비용을 지불하는 것보다 채굴에 컴퓨팅 전략을 사용하는 것이 더 수익성이 높을 수 있습니다. 그들은 (그것이 프로토콜의 유틸리티를 강화하거나 지원하지 않는다는 점에서) 이 변질된 전략을 사용할 것입니다 — 채굴자는 파티션의 패킹되지 않은 단일 복제본을 유지합니다.



Every tick of the VDF, the protocol will pseudorandomly assign recall ranges from the staked partitions.



For each staked partition, the miner then packs the unpacked chunks for that partition in parallel using their compute.

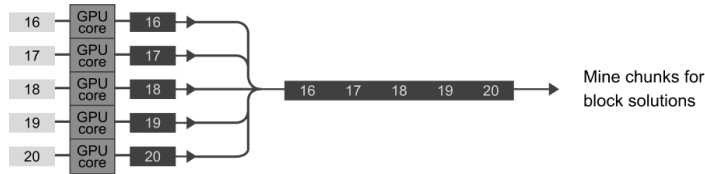


그림 8: "온디맨드 패킹" 예시 (적대적)

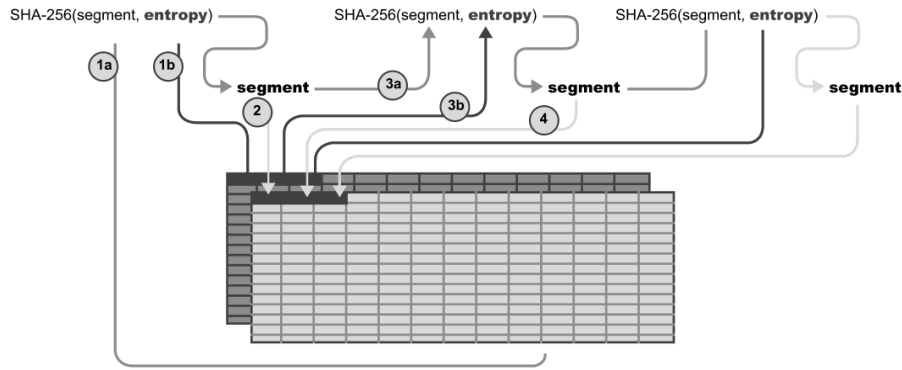
만약 이 과정이 블록 보상을 획득할 만큼 충분히 빠르다면, 채굴자는 프로토콜이 기대하는 것처럼 파티션 복제본의 모든 청크를 패킹하는 대신 그것을 선택할 수 있습니다.

4.3.5 2 단계 - 2D 매트릭스 패킹

"온디맨드" 패킹을 중지시키기 위해, 프로토콜은 채굴자들이 새로운 레이어를 연산할 때 이전 레이어의 모든 청크 segment 들을 저장할 것을 요구합니다. 이는 폭과 깊이라는 2 차원적 순차 패킹을 통해 모든 이전 레이어들을 재연산해야만 최종 레이어 연산을 가능하게 만듭니다. 각 레이어는 1 단계 - 순차 패킹에서 시작하여, 이전 레이어에 의존합니다.

1. a) 이전 레이어의 마지막 segment 는 새 레이어의 첫 번째 순차 해시의 입력이 됩니다.
1. b) 우리가 연산하고 있는 segment 와 동일한 위치에 있는 이전 레이어의 segment 는 현재 segment 에 대한 입력 엔트로피가 됩니다.
2. 이전 segment 와 엔트로피 segment 를 해싱하여 생성된 segment 는 새 레이어에 추가됩니다.
3. a) 단계에서와 같이, 2 단계에서 연산된 segment 는 다음 순차 SHA-256 해시의 입력으로 사용됩니다.

3. b) 1b 단계에서와 같이, 현재 연산 중인 segment 와 동일한 위치에 있는 이전 패킹 레이어의 segment 가 엔트로피로 사용됩니다.
4. 결과로 나온 segment 는 이전 segment 뒤에 청크로 추가됩니다.
5. 패킹 지속 시간 요구 사항을 만족시키기 위해 필요한 수의 패킹 레이어가 달성될 때까지 반복합니다.



그림

9: 순차 패킹 프로세스

이러한 방식으로 2 차원으로 패킹하는 결과는, 새로운 패킹 레이어의 모든 segment 가 바로 아래 패킹 레이어의 segment 들을 요구한다는 것입니다. 그 아래의 모든 패킹 레이어를 먼저 연산하지 않고서는 최종 패킹 레이어를 연산할 방법이 없습니다. 이는 "온디맨드"로 패킹하고자 하는 적대적 채굴자들이 청크를 패킹할 때마다 전체 패킹 지속 시간 동안 패킹해야 함을 보장합니다.

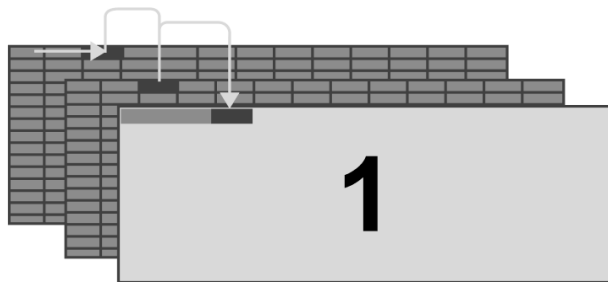


그림 10: 2D 매트릭스 패킹

5 작업증명/지분증명 합의

Irys 는 하이브리드 작업증명-지분증명 합의 알고리즘을 도입합니다. 이는 다음과 같은 요구 사항을 달성하기 위해 수행되었습니다:

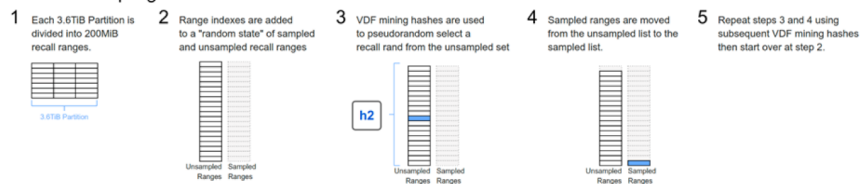
- 영구 데이터를 위한 지속 가능한 경제학
- 장기적으로 인플레이션 압력 최소화
- 채굴자들이 악의적인 행동에 대해 처벌받을 수 있도록 책임성 생성
- 저장된 모든 데이터를 하루에 한 번씩 안정적으로 샘플링해야 함

하이브리드 합의는 우리가 슬래싱과 같은 보안 메커니즘과 함께 작업증명 경제학을 채택하여, 고성능 실행과 함께 최대한 신뢰할 수 있는 온체인 데이터를 제공할 수 있게 합니다.

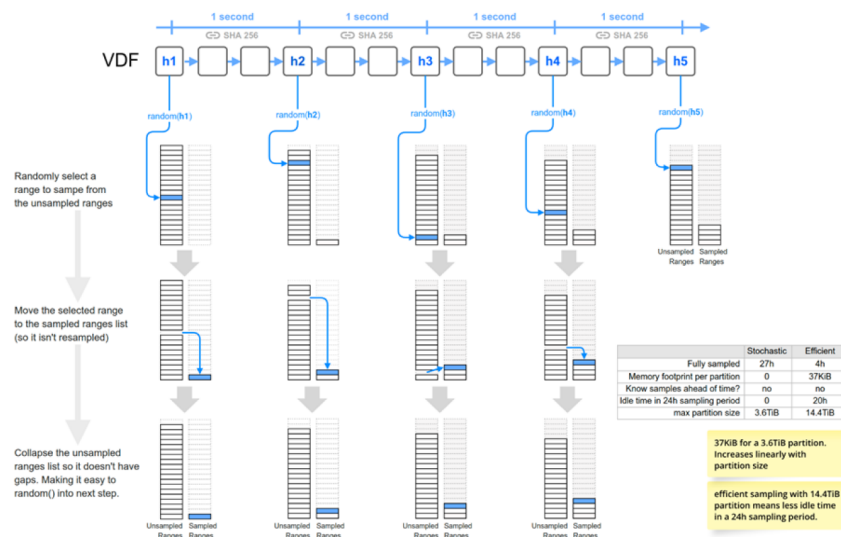
5.1 합의 알고리즘

Irys 는 체인이 매일 모든 파티션을 완전히 샘플링하도록 보장하는 효율적인 샘플링이라는 프로세스를 사용합니다.

Efficient Sampling



Sampling Example



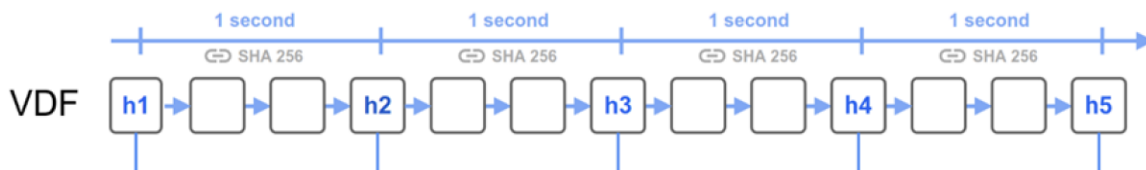
알고리즘은 다음과 같이 요약될 수 있습니다:

1. 1 초마다 "틱"을 실행하여 시드를 생성하는 검증 가능 지연 함수가 있습니다.
2. 채굴자가 저장하는 모든 파티션에 대해, 시드를 결정론적 랜덤 함수에 사용하여 범위의 시작 위치를 생성합니다.
3. 200MiB 상당의 청크(800 개)를 읽습니다.
4. 각 청크에 대해 솔루션을 계산합니다.
 - a. 숫자로 변환합니다.
5. 솔루션이 현재 네트워크 난이도보다 크면 블록을 게시합니다. 난이도보다 낮으면 1 단계로 돌아갑니다.

여기서의 아이디어는 채굴자들이 합의에 참여하기 위해 지속적으로 데이터를 샘플링하여 데이터가 샘플링된다는 강력한 보장을 생성한다는 것입니다. 채굴자가 데이터를 잃으면 이익을 제기받을 수 있으며, 악의적인 것으로 입증되면 결국 슬래싱될 수 있습니다.

5.2 읽기 속도 동기화를 위한 검증 가능 지연 함수

Irys의 핵심 요구 사항 중 하나는 사람들이 HDD를 사용하도록 강제하는 것, 즉 SSD를 사용하여 다른 사람을 경쟁에서 이기지 못하게 하는 것이었습니다. 검증 가능 지연 함수는 채굴자가 저장하는 파티션당 매초 200MB만 읽을 수 있도록 하여 이를 달성하며, 본질적으로 각 파티션의 읽기 속도를 200MB(HDD의 최대 읽기 속도에 근접)로 제한합니다.



난이도가 네트워크 난이도보다 크고 채굴자가 충분한 지분을 가지고 있다면, 블록은 수락됩니다.

6 데이터 트랜잭션 생명주기

각 데이터 트랜잭션은 데이터가 저장되어야 할 원장을 지정하며, 각 원장은 데이터가 저장되어야 하는 기간을 나타냅니다. 예를 들어, 트랜잭션은 "나는 이 1GB의 데이터를 2주 동안 데이터를 저장하는 원장 0에 저장하고 싶다"를 나타낼 수 있습니다. 사용자가 트랜잭션을 게시하고 모든 인그레스 증명이 검증되면, 데이터는 파티션에 배타적 논리합 연산되어 합의 증명에 사용될 수 있습니다.

유일한 특별한 경우는 트랜잭션이 영구 원장을 지정할 때입니다. 이 경우, 트랜잭션은 서브밋 원장에 들어가고 10 명의 채굴자가 데이터를 저장하고 있음을 증명하면 영구 원장으로 마이그레이션됩니다.

6.1 인그레스 증명

인그레스 증명은 트랜잭션 데이터의 청크에 접근해야만 생성할 수 있는 머클 루트입니다. 인그레스 증명은 채굴자가 특정 데이터를 가지고 있음을 증명하는 데 사용됩니다. 이는 채굴자들이 데이터를 저장하기 전에 필요한 단계로, 그들이 데이터의 고유한 복제본을 저장한다는 증거를 제공합니다.

데이터 청크는 채굴자의 주소와 함께 해싱되어 해당 채굴 주소에 고유한 증명을 생성합니다. 이 과정은 데이터를 가진 사람이 증명을 쉽게 생성하고 검증할 수 있게 합니다. 허위 주장을 방지하기 위해, 이러한 증명은 채굴자의 개인 키로 서명되어야 합니다. 이 서명이 없으면 데이터에 접근할 수 있는 모든 채굴자가 다른 채굴 주소에 대한 증명을 생성하여 데이터를 다운로드했다고 거짓으로 주장할 수 있습니다.

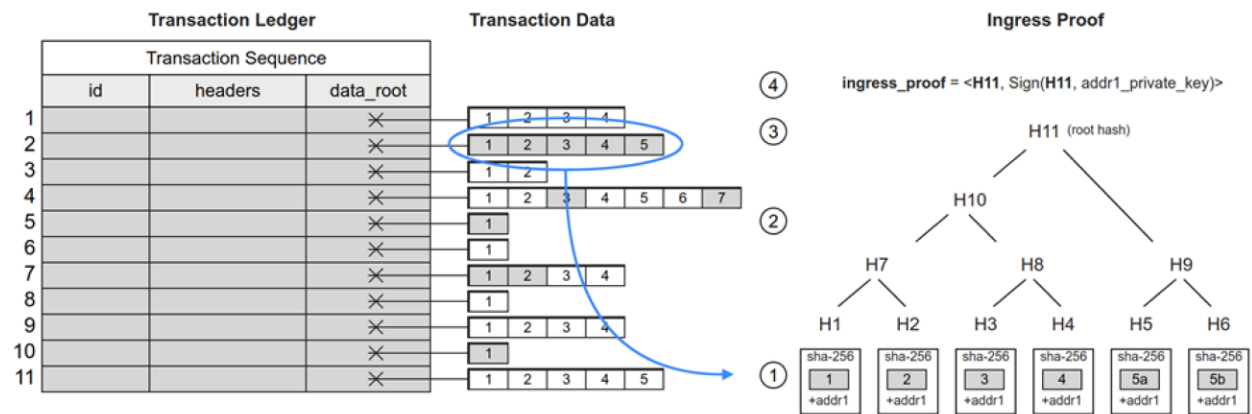


그림 13: 인그레스 증명

서브밋 원장에서 퍼블리시 원장으로 승격되기를 기다리는 최근에 추가된 트랜잭션을 식별합니다. 사용자로부터 직접 또는 다른 채굴자와의 가십을 통해 이러한 트랜잭션과 관련된 데이터를 얻습니다.

모든 데이터 청크가 수집되면 다음과 같이 인그레스 증명을 생성합니다:

1. 데이터를 청크로 나누고 각 청크를 채굴자의 주소와 함께 해싱합니다.
(참고: 청크 5는 더 균형 잡힌 머클 트리를 만들기 위해 5a와 5b로 나뉩니다.)

2. 이 해시들을 계속 해싱하여 머클 트리의 가지를 만듭니다.
3. 머클 트리의 루트 해시를 계산합니다.
4. 1 단계에서 사용된 채굴 주소와 연관된 개인 키로 루트 해시에 서명합니다.

7 Irys 가상머신과 프로그래밍 가능한 데이터

Irys 는 새로운 옴코드를 추가하는 이더리움 가상머신++ 구현체인 Irys 가상머신을 도입합니다. 주요 업그레이드는 스마트 계약 내에서 저장된 데이터를 사용할 수 있도록 하는 프로그래밍 가능한 데이터를 활성화하는 것입니다. 본질적으로 이는 1 엑사바이트의 데이터를 저장하고 그 데이터의 일부를 보안 환경 내에서 사용할 수 있음을 의미합니다.

프로그래밍 가능한 데이터를 완전히 구현하려면, 기술 스택의 여러 계층에 걸쳐 이 기능을 통합해야 합니다. 호출자가 프로그래머블 데이터 스마트계약 호출 시 접근하고자 하는 청크의 범위를 지정할 수 있는 트랜잭션부터 시작합니다. 다음으로, 트랜잭션을 가십하고 청크의 범위를 요청하여 가용성을 확인합니다. 마지막으로, 블록에 트랜잭션을 포함시키고 상태 전환을 실행합니다.

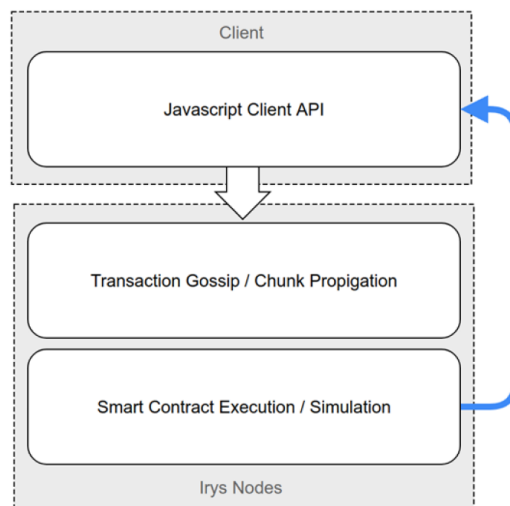


그림 14: 상태 트랜잭션의 실행

7.1 트랜잭션 게시

프로그래머블 데이터 트랜잭션을 게시하려면, 트랜잭션은 스마트계약 호출 중에 사용 가능하게 하려는 청크의 범위를 지정해야 합니다. 이더리움 가상머신 툴체인과의 호환성을 유지하기 위해, 이는 [EIP-2930](#) 접근 목록을 사용하여 참조할 청크의 범위를 요소로

포함함으로써 이루어집니다. 여기서 주소는 프로그래머블 데이터 프리컴파일의 주소이고, 데이터는 하나 이상의 범위 명세입니다.

7.2 범위 명세

청크 범위 명세는 간단한 형식을 따릅니다.

```
<partition_index>:<offset>:<chunk_count>
```

partition_index : 첫 번째 청크를 포함하는 파티션의 퍼블리시 원장 내 파티션 인덱스.

offset: 파티션에서 청크 읽기를 시작할 청크 오프셋.

chunk_count: 오프셋 이후에 읽을 순차적인 청크의 수.

partition_index : 26 바이트 - 0 부터 아주 큰 수까지

offset: 4 바이트 - 0 부터 약 80,000 까지 (파티션 내 청크 수)

chunk_count: 2 바이트 - 0 부터 65,535 까지 (프로그래머블 데이터는 블록당 약 7,000 청크로 제한되므로, 이는 10 배 가량 안전합니다)

범위를 나타내는 총 바이너리 바이트: $26+4+2=32$ 바이트

범위를 32 바이트로 제한하면 이더리움 가상머신 스마트 계약 상태에 쉽게 저장할 수 있습니다.

참고: 범위는 항상 32 바이트이며, 값은 각 값에 대한 올바른 오프셋에서 바이트를 슬라이싱하여 압축 해제됩니다.

7.3 Irys 클라이언트 SDK

Irys 에서 데이터를 참조하는 기본 방법은 트랜잭션 아이디를 사용하는 것이므로, 개발자가 프로그래머블 데이터에 관심 있는 데이터의 파티션 및 청크 오프셋을 찾는 부담은 크고 체인의 내부 데이터 모델에 대한 많은 지식이 필요합니다.

프로그래머블 데이터 트랜잭션 게시 과정을 단순화하기 위해, SDK 는 청크 범위 지정의 복잡성을 추상화하는 몇 가지 유틸리티 메서드를 구현합니다.

```
const chunkRange = getTxChunkRange(txid);
```

이 함수의 구현은 클라이언트가 사용하는 게이트웨이에 의해 제공됩니다.

1. 트랜잭션 아이디가 실제로 데이터아이템 아이디인 경우 번들 트랜잭션 아이디를 조회합니다.
2. 트랜잭션 아이디가 퍼블리시 원장에 있는지 확인합니다.
3. 퍼블리시 원장에서 청크 오프셋을 조회합니다.
4. 퍼블리시 원장에서 파티션 인덱스를 조회합니다.
5. 3 과 4 를 사용하여 파티션 오프셋을 계산합니다.
6. 퍼블리시 원장 트랜잭션 아이디를 사용하여 트랜잭션 데이터의 크기를 조회합니다.
7. 트랜잭션을 읽는 데 필요한 청크 수를 계산합니다.
8. 범위 명세를 구축하고 호출자에게 반환합니다.

7.4 트랜잭션 가격 추정

프로그래머블 데이터의 가격은 트랜잭션을 이더리움 가상 머신에 게시할 때 필요한 가스 가격을 추정하는 것과 동일한 트랜잭션 시뮬레이션에 의해 결정된다. 청크 범위가 트랜잭션의 호출 데이터에 포함되어 있기 때문에, 시뮬레이션을 실행하면 시뮬레이션 메커니즘은 트랜잭션이 필요로 하는 계산 예산과 함께 조회해야 하는 청크 개수에 대한 가격도 포함한다.

Irys 에서 프로그램형 데이터의 비용을 정확하게 산정하기 위해서는 네트워크 성능에 영향을 미치는 주요 매개변수를 정의해야 한다:

전파 지연 (D): 데이터가 네트워크 전반을 통해 전파되는 데 허용되는 최대 시간. 상한값을 선택하면 네트워크의 블록 생성 시간과 포크 발생 확률을 추정할 수 있다.

최소 연결 속도: 전파 지연이 정의되면 네트워크에서 기대되는 데이터 처리량을 설명하기 위한 최소 연결 속도를 산정할 수 있다.

가십 네트워크에서의 피어 연결: 각 피어는 제한된 수의 다른 피어와 연결된다 (예: 20 개). 이러한 연결 수는 개별 피어의 네트워크 연결 활용도를 극대화하는 동시에, 네트워크 상의 임의의 두 피어 간에 필요한 '홉'의 수를 최소화할 수 있게 한다.

7.4.1 가십 네트워크

네트워크 전체에 데이터를 전파하는 데 걸리는 시간을 계산하려면 기본 가십 프로토콜에 대한 약간의 이해가 필요합니다. 가십 프로토콜은 모든 피어를 다른 모든 피어와 연결하는 것보다 더 잘 확장되고 더 높은 처리량을 갖기 때문에 존재합니다.

1000 개의 피어로 이루어진 네트워크에서, 각 피어는 999 개의 연결을 유지하며, 메모리와 CPU 를 소모합니다. 피어가 블록을 생성할 때, 각 피어에게 사본을 보냅니다. 1 Gbps 연결(125 MBps)과 500 KB 블록으로, 피어는 초당 250 피어에게 브로드캐스트할 수 있습니다. 모든 피어에게 도달하는 데 4 초가 걸리며, 연결을 완전히 사용합니다.

동일한 1000-피어 네트워크에서, 각 피어는 핑 시간을 기준으로 20 개의 "가장 가까운" 피어에만 연결합니다. 블록은 20 개의 피어에게 브로드캐스트되고, 그들은 다시 그들의 20 개 가장 가까운 피어에게 전달합니다. 1 Gbps 연결로, 20 개의 피어에게 브로드캐스팅하는 것은 매우 짧은 시간 안에 일어납니다. 블록은 몇 번의 홉 만에 전체 네트워크에 도달하며, 이는 다음과 같이 모델링됩니다:

$$\log_{\text{number_of_peer_connections}}(\text{total_network_peers})$$

$$\log_{20}(1,000) = 2.31\text{hops}$$

$$\log_{20}(10,000) = 3.07\text{hops}$$

블록이 몇 번의 홉 만에 피어들 간에 브로드캐스트됨에 따라, 전파 지연 D 는 합의에 도달하는 시간에 직접적인 영향을 미치므로, 시기적절한 블록 전파를 보장하는 데 중요한 요소가 됩니다.

7.4.2 전파 지연

전파 지연(D)은 Irys 네트워크의 피어 간에 데이터가 이동하는 시간입니다. 가십 네트워크에서, 데이터는 여러 홉을 통해 이동합니다. 홉당 평균 60ms 의 링크 속도를 가정하면, 데이터는 세 번의 홉으로 10,000 개의 피어에 도달할 수 있으며, D 는 약 180ms 가 됩니다.

포크는 하나의 블록이 전파되는 동안 다른 블록이 생성될 때 발생할 수 있습니다. 180ms 의 D 로 1%의 포크 확률을 유지하기 위해, 블록 시간은 약 18 초가 되어야 합니다. 테스트넷을 위해, Irys 는 300ms 의 D 로 포크 확률을 1%로 유지하기 위해 30 초 블록을 구현할 것입니다. 네트워크 합의는 전파 지연 내에 최소 51%의 노드가 블록을 확인해야 하며, 더 느린 링크를 가진 소수의 노드를 허용합니다.

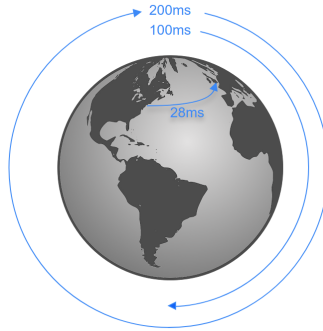


그림 15: 전파 지연

이는 네트워크 전파 지연이 핵심 성능 및 보안 파라미터에 직접적인 영향을 미치기 때문에 중요합니다. 더 짧은 전파는 동시 블록 생성의 창을 최소화하여 포크의 가능성을 줄여, 단일의 정식 체인이 더 안정적으로 형성되도록 보장합니다. 또한 블록 시간의 하한을 정의합니다; 더 빠른 전파는 블록 간의 간격을 더 짧게 허용하여, 응답성과 트랜잭션 처리량을 향상시킵니다. 추가적으로, 전파 지연을 이해하고 최적화하는 것은 네트워크의 확장성 한계를 결정하며, 시스템 전반에 걸쳐 효율적인 데이터 흐름과 동기화된 상태를 유지하면서 얼마나 많은 노드가 합의에 참여할 수 있는지를 좌우합니다.

7.4.3 프로그래밍 가능한 데이터 모델링

1Gbps 연결에서, 피어는 초당 500 개의 데이터 청크(256KB)를 전송할 수 있으며, 연결을 완전히 활용합니다. 프로토콜은 청크 인그레스 및 블록 가십과 같은 다른 작업들을 처리하므로, 대역폭의 50%가 프로그래밍 가능한 데이터 청크 전파에 사용 가능하다고 가정할 것입니다.

전송된 프로그래밍 가능한 데이터의 비용을 결정하기 위해, 다음을 고려하십시오:

네트워크 용량:

- 가정된 연결: 1Gbps
- 최대 전송 속도: 초당 500 청크 (각 256KB)
- 프로그래밍 가능한 데이터에 사용 가능한 대역폭: 50%
- 유효 전송 속도: 초당 250 청크

블록 용량:

- 블록 시간: 30 초
- 블록당 최대 청크: 7,500 (250 청크/초 * 30 초)

비용 분석:

- 월 \$300 의 10Gbps 연결(소매 가격) 기준
- 전송 비용: 1000MB 당 \$0.10
- 블록당 비용 (7,500 청크 = 1,875MB): \$0.20

- 청크당 비용: \$0.00002667 (개별 가격 책정에는 무시할 수 있는 수준)

7.4.4 프로그래밍 가능한 데이터 가격 책정

원시 대역폭을 넘어, IrysVM 내에서 사용하기 위해 데이터 청크를 압축 해제하고 역직렬화하는 것은 스팸을 방지하기 위해 가격이 책정되어야 하는 연산 오버헤드를 발생시킵니다. 이를 고려하기 위해, Irys는 프로그래밍 가능한 데이터 트랜잭션에 대한 최소 기본 수수료를 구현합니다. 테스트넷 동안, 1MB의 프로그래밍 가능한 데이터에 대한 기본 수수료는 \$0.01로 설정되며, 이는 또한 그러한 트랜잭션에 대한 네트워크 전체의 최소 비용으로도 작용합니다.

처리량과 수요의 균형을 맞추기 위해, Irys는 이더리움의 가스 수수료 메커니즘과 유사하게 기본 수수료가 블록당 $\pm 12.5\%$ 씩 조정되는 동적 혼잡 가격 책정 모델을 사용합니다. 만약 블록에서 사용 가능한 7,500개의 프로그래밍 가능한 데이터 청크 중 50% 이상이 사용되면, 기본 수수료는 최대 +12.5%까지 선형적으로 증가합니다. 반대로, 블록에 프로그래밍 가능한 데이터 청크가 포함되어 있지 않으면, 기본 수수료는 최대 -12.5%까지 감소합니다. 혼잡 중 수수료 성장에 대한 상한선은 없지만, 시스템은 네트워크 무결성을 유지하기 위해 \$0.01의 절대적인 하한선을 강제합니다.

기본 수수료가 상승하고 있지만 프로그래밍 가능한 데이터 수요가 여전히 가용 용량을 초과할 때, 사용자는 채굴자나 검증인이 다음 블록에 자신의 트랜잭션을 포함시키도록 인센티브를 주기 위해 우선순위 수수료(기본 요율 이상의 추가 지불금)를 첨부할 수 있습니다. 이는 일반적인 사용에 대한 예측 가능한 기준 비용을 보존하면서, 고가치 또는 시간에 민감한 워크로드에 대한 시장 주도적 포함을 보장합니다.

프로그래밍 가능한 데이터 트랜잭션의 모든 기본 수수료는 네트워크 트레저리로 직접 입금되어, 유통에서 효과적으로 제거되고 네이티브 \$IRYS 토큰에 디플레이션 압력을 생성합니다. 대조적으로, 우선순위 수수료—사용자가 기본 및 연산 수수료 이상으로 지불하는 부분—는 블록 생성자에게 지급됩니다. 이 이중 인센티브 모델은 채굴자 수익성을 네트워크 효율성과 일치시키며, 동시에 프로토콜 수준의 소각 및 트레저리 축적을 통해 토큰 가치를 점진적으로 강화합니다.

7.5 가십 및 멤폴

프로그래머블 데이터 트랜잭션에 대한 중요한 제약은 블록 사이에서 네트워크상의 대다수 노드 간에 프로그래머블 데이터 청크를 전파하기 위한 최대 용량입니다.

7.6 전송

노드가 프로그래머블 데이터 트랜잭션을 수신하면, 청크를 가지고 있는지 여부를 나타내며 피어들에게 브로드캐스트합니다. 수신하는 피어들은 응답으로 청크를 요청할 수 있습니다.

브로드캐스팅하는 노드는 어떤 피어가 청크를 필요로 하는지 추적하고 수신한 후 전송합니다.

수신하는 피어도 프로그래머블 데이터 트랜잭션을 브로드캐스트하며, 이미 수신한 피어를 표시하고 어떤 피어가 청크를 가지고 있는지 기록합니다.

피어가 청크를 수신하면, 청크가 없는 피어에게 전송합니다.

피어가 일정 시간 후에도 청크를 받지 못하면, 원장을 검사하고 파티션 소유자를 찾아 무작위로 선택된 파티션에서 청크를 요청하여 할당된 스토리지 파티션에서 검색할 수 있습니다.

7.7 검증

트랜잭션 검증은 다른 트랜잭션과 동일한 정적 검증을 따릅니다.

청크 검증은 조금 더 복잡합니다. 프로그래머블 데이터 트랜잭션 청크는 여러 가지 방법으로 검색될 수 있습니다.

1. 노드가 캐시된 압축 해제된 청크를 로컬에 가지고 있습니다.
2. 노드가 채굴하는 파티션에 패킹된 청크를 가지고 있습니다.
3. 노드가 피어로부터 압축 해제된 청크를 받습니다.
4. 노드가 피어에게 패킹/압축 해제된 청크를 요청합니다.

캐시된 압축 해제된 청크: 이 경우, 노드는 이미 청크를 머클 루트로 검증했으며 그 안의 데이터가 정확하고 프로그래머블 데이터 실행을 위해 가상머신에 노출될 준비가 되었다고 확신할 수 있습니다.

로컬 패킹된 청크: 노드가 우연히 트랜잭션에 의해 요청된 프로그래머블 데이터 청크를 포함하는 파티션을 채굴하지만, 이들은 패킹되어 있습니다.

노드는:

1. 청크 범위에 대한 엔트로피를 생성합니다.
2. 계산된 엔트로피를 사용하여 청크를 압축 해제합니다.
3. 압축 해제된 청크로 머클-루트를 구축합니다.
4. 블록 인덱스에서 청크를 게시한 트랜잭션을 조회합니다.
5. 계산된 머클-루트를 트랜잭션에 있는 것과 비교합니다.
6. 유효하면, 노드는 압축 해제된 청크를 가지고 있지 않다고 표시된 모든 피어에게

게시합니다.

압축 해제된 청크 수신: 이 경우 노드는 피어 중 한 명으로부터 압축 해제된 청크를 받고 있습니다. 청크를 압축 해제할 필요는 없지만, 청크에 올바른 데이터가 포함되어 있는지 확인해야 하며 그렇지 않으면 유효하지 않은 블록을 제안할 위험이 있습니다.

노드는:

로컬 패킹된 청크 경로의 4-6 단계를 따릅니다.

청크 요청: 안전장치로서, 노드가 디 시간 내에 청크를 받지 못하는 경우, 여기서 디는 네트워크의 전파 지연 시간입니다(테스트넷의 경우 디 = 200 밀리초로 가정).

노드는:

1. 청크 저장을 담당하는 파티션을 조회합니다.
2. 청크를 요청할 파티션을 무작위로 선택합니다.
 - a. 파티션이 패킹된 청크를 제공하면, 압축 해제합니다.
3. 로컬 패킹된 청크 경로의 4-6 단계를 따릅니다.

7.8 블록 생성

채굴 노드가 프로그래머블 데이터 트랜잭션과 그 청크를 수신하고 검증한 후, 해당 트랜잭션을 블록에 포함시킬 수 있습니다. 대다수가 청크를 받지 못한 프로그래머블 데이터 트랜잭션을 포함하는 블록을 생성할 가능성을 최소화하기 위해, 채굴자는 이를 포함하기 전에 전파 지연 시간 디를 기다릴 수 있습니다. 이는 네트워크의 대부분이 블록에 포함될 때 프로그래머블 데이터 트랜잭션을 검색하고 검증할 수 있게 합니다.

7.9 스마트 계약 실행

프로그래머블 데이터 스마트 계약 상호작용을 실행하려면 추가적인 탐구가 필요합니다. 몇 가지 가능한 접근 방식이 있지만, 그 타당성을 평가하기 위해 코드 탐구가 필요할 것입니다.

7.9.1 청크 데이터 노출

프로그래머블 데이터 트랜잭션에는 범위 명세를 입력으로 받는 사전 컴파일된 "시스템" 계약에 대한 지침이 포함되어 있습니다. 이 프리컴파일은 청크 데이터를 범위 내로 가져옵니다. 최소 두 가지 가능한 접근 방식이 있습니다.

반환 값: 사전 컴파일된 "시스템" 계약에 대한 외부 호출은 범위에 의해 지정된 청크를 버퍼로 반환할 수 있습니다.

전역 상태: 사전 컴파일된 "시스템" 계약이 호출되면, 청크는 프로그래머블 데이터 트랜잭션의 모든 후속 지침에 노출되는 전역 변수를 통해 접근 가능하게 됩니다.

7.9.2 컴퓨트 유닛 계산

특정 스마트 계약 함수의 실행이 청크에서 읽은 데이터에 따라 한 코드 경로 또는 다른 코드 경로를 취할 수 있기 때문에, 컴퓨트 유닛 계산은 문제가 될 수 있습니다. 몇 가지 가능한 접근 방식이 있습니다.

청크로 시뮬레이션: 지시의 실행을 완료하는 데 필요한 컴퓨트 유닛을 결정론적으로 시뮬레이션하는 유일한 방법은 시뮬레이션 중에 압축 해제된 청크를 사용할 수 있도록 하는 것입니다. 이를 위해서는 시뮬레이션 노드가 시뮬레이션 요청 중에 압축 해제된 청크를 검색해야 합니다.

컴퓨트 상한 시뮬레이션: 이 경우, 시뮬레이션은 모든 코드 경로를 평가하고 가장 계산 집약적인 코드 경로의 비용을 반환합니다. 이렇게 하면 사용자는 가능한 모든 계산 리소스에 대해 항상 충분한 가스를 지불하게 됩니다.

7.10 프로그래밍 가능한 데이터 로드맵

블록 데이터 – 최소 기능 제품

프로그래머블 데이터 트랜잭션의 첫 번째 버전은 청크를 계약에 버퍼 또는 블록 데이터로 노출하고 이러한 바이트의 해석을 호출자에게 맡길 것입니다. 이를 통해 프로그래머블 데이터 청크는 호출자가 상상할 수 있는 모든 구조나 형식을 가질 수 있습니다.

번들 형식 버전 1 – 번들

블록 청크가 작동하면, 다음 기능 계층은 프로그래머블 데이터 계약이 청크에서 번들 및 데이터 항목을 로드할 수 있게 하는 개발자 경험 업그레이드가 될 것입니다. Irys 가상머신은 범위 명세의 청크를 버전 1 번들 형식으로 파싱합니다.

번들 형식 버전 2 – 데이터아이템

버전 2 (머클화된) 번들 형식이 존재하면, 개발자 경험 업그레이드는 더 큰 번들에서 특정 데이터 항목을 파싱하거나, 호출자가 관심 있는 특정 데이터 항목만 저장하는 청크를 로드하여 더 작은 중첩 번들(또는 그 데이터 항목)을 검색할 수 있게 합니다.

프로그래밍 가능한 데이터 레이어-2

미래에는 사용자들이 단일 상태 머신을 넘어 계산 용량을 확장하는 프로그래밍 가능한 데이터 레이어-2를 개발할 것으로 기대합니다. 여기서 최종 목표는 누구나 레이어-2를 가동하여 데이터, 컴퓨트 및 유동성 리소스에 접근할 수 있는 공유 데이터셋입니다.

8 토크노믹스

Irys 네이티브 토큰인 IRYS는 Irys 프로토콜의 경제 및 보안 아키텍처를 뒷받침합니다. 그것의 설계는 사용자, 채굴자, 그리고 더 넓은 생태계 전반의 인센티브를 공동으로 정렬하는 다섯 가지 기본 구성 요소를 통합합니다. 그것의 역할은 초기 채택자와 커뮤니티 구성원부터 기관 파트너 및 장기 투자자에 이르기까지 모든 참여자 전반의 인센티브를 정렬하는 것입니다.

8.1 토큰 분배

IRYS의 총 공급량은 10,000,000,000 토큰입니다. 토큰 생성 이벤트(TGE) 시, 유통 공급량은 약 2,000,000,000 토큰이 될 것이며, 이는 최대 공급량의 20.0%를 나타냅니다.



TOKEN ALLOCATION

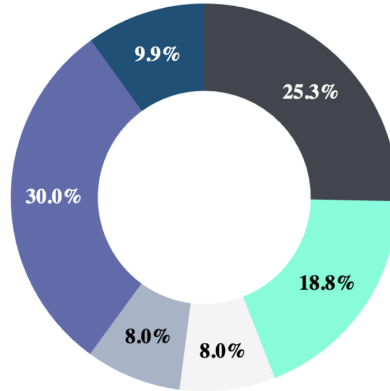
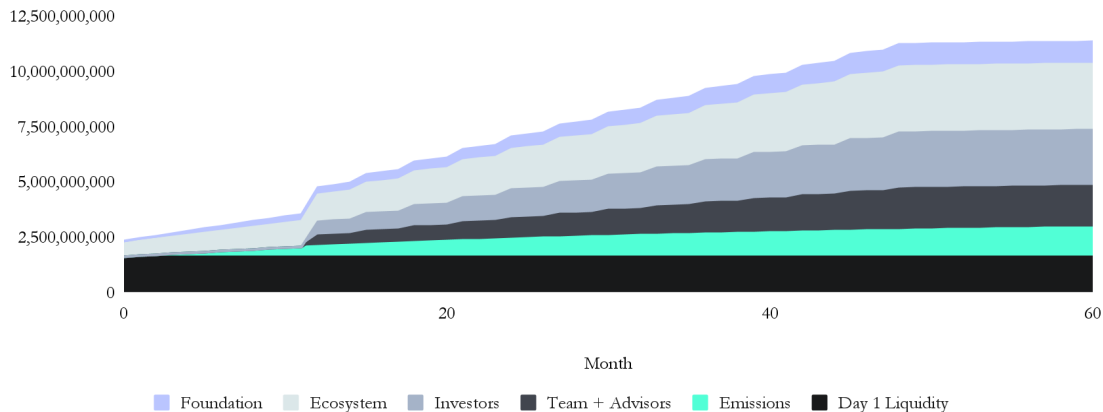


그림 16: Irys 토큰 할당.

Token Distribution Schedule



그림

17: 토큰 분배 일정

생태계 (30%): 이 할당량은 새롭고 탈중앙화된 애플리케이션을 유지하고 인센티브를 제공하는 것을 포함하여, 다양한 Irys 이니셔티브, 그리고 다양한 크로스체인 이니셔티브, 파트너십 등에 사용될 것입니다. 이 할당량은 해당 분야의 최고 수탁 기관들과 협력하여 보안 멀티시그 월렛에 의해 보유될 것입니다.

재단 (9.9%): 이는 Irys의 도달 범위를 넓히고, 중앙화된 데이터 스토리지와 파편화된 개발자 경험에 대한 암호화폐의 의존도를 줄이는 데 기여하는 이니셔티브를 증진하기 위해 사용됩니다. 이 \$IRYS 할당량은 추가 개발, 위험 평가, 감사 등에 자금을 지원하는 데 사용될 것입니다.

핵심 팀 및 어드바이저 (18.8%): \$IRYS 할당량의 이 부분(18.8%)은 Irys 레이어 1 프로토콜의 핵심 기여자들, 그리고 L1 프로토콜을 시장에 출시하기 위해 노력한 어드바이저들에게 분배되는 것을 나타냅니다. 모든 핵심 기여자들은 1년의 클리프로

락업되며, 이후 3 년간 월별 선형 베스팅이 적용됩니다. 1 년의 클리프 이전에 락업 해제되는 핵심 기여자 토큰은 없으며, 베스팅 일정에 앞서 락업 해제되는 스테이킹 보상은 없습니다.

에어드랍 및 향후 인센티브 (8%): 이 할당량은 Irys 를 시장에 출시하는 데 도움이 된 의미 있는 기여들을 인정하기 위해 설계되었습니다. 이 할당량의 일부는 시간이 지남에 따라 뜻이 맞는 기여자, 개발자, 사용자를 유치하기 위해 고안된 향후 인센티브 프로그램을 위해 예약되어 있습니다. 이는 Irys 가 단지 가장 목소리가 큰 사람이 아니라, 올바른 사람들과 함께 계속 성장하도록 보장합니다.

유동성, 시장 안정성 및 런치 파트너십 (8%): 이 할당량은 첫날부터 강력한 유동성 프로필을 확립하기 위해 마켓 메이커 및 런치 파트너들에게 배포될 것입니다. 우리의 목표는 유기적인 거래량 성장을 지원하고 조작을 방지하는 효율적인 거래 환경을 만드는 것입니다. 이 할당량을 전략적 파트너 및 CEX 와 DEX 인프라의 혼합에 고정함으로써, 우리는 장기적인 지속 가능성을 저해하지 않으면서 Irys 가 접근 가능하고, 유동적이며, 거래 준비가 되도록 보장합니다.

투자자 (25.3%): 이는 프로토콜을 부트스트랩하기 위해, Irys 프로토콜의 개발을 지원하는 투자자들이 획득한 토큰 권리를 나타냅니다. 모든 투자자들은 1 년의 클리프로 락업되며, 이후 3 년간 월별 선형 베스팅이 적용됩니다. 1 년의 클리프 이전에 락업 해제되는 투자자 토큰은 없으며, 베스팅 일정에 앞서 락업 해제되는 스테이킹 보상은 없습니다.

8.2 토큰 유틸리티

토큰은 출시 시 다음과 같은 모든 유틸리티를 갖습니다:

네이티브 자산: \$IRYS 는 Irys 네트워크의 네이티브 유틸리티 토큰으로서 기능하며, 모든 프로토콜 수준 운영에서 결제, 담보, 정산을 위한 기본 매개체 역할을 합니다.

수수료: 데이터 업로드, 프로그램형 데이터 실행, 계약 상호작용을 포함한 모든 네트워크 활동은 \$IRYS 로 지불됩니다. Irys 는 임시 및 영구 데이터 저장 모두에 대해 실제 저장 비용을 반영하여 매년 재조정되는 미 달러 연동 가격 모델을 사용합니다. 이를 통해 토큰 변동성과 무관하게 사용자와 개발자는 안정적이고 예측 가능한 경제 구조를 유지할 수 있습니다.

보안: 합의 무결성과 데이터 가용성을 유지하는 검증자와 채굴자에게 토큰 발행 및 보상이 배분됩니다. 이러한 인센티브 구조는 스팸, 서비스 거부 공격 및 기타 적대적 행위로부터 경제적 보호를 제공하며, 참여가 증가함에 따라 네트워크 신뢰성이 확장되도록 보장합니다.

기금: \$IRYS의 일부는 채굴자의 장기 저장 의무를 보증하는 프로토콜 수준의 기금으로 배정됩니다. 이 기금은 미래 저장 부담을 충당하는 자가 유지형 준비금으로 기능하며, 네트워크 데이터 레이어의 영구성 보증을 강화합니다.

스테이킹: 채굴자와 검증자는 합의 및 데이터 검증에 참여하기 위해 담보 형태로 \$IRYS 토큰을 스테이킹해야 합니다. 이 메커니즘은 책임성을 강제하며, 악의적 또는 부주의한 행동 시 슬래싱이 적용됩니다. 위임 메커니즘을 통해 토큰 보유자는 간접적으로 네트워크 보안에 기여할 수 있으며, 위임된 지분에 비례한 보상을 받을 수 있습니다.

8.3 소각 메커니즘

\$IRYS 토큰 경제는 네트워크 생애주기 초기에 강력한 디플레이션 역학을 나타내도록 설계되었으며, 사용 수요가 발행의 감소를 상쇄함에 따라 점진적으로 균형 상태로 전환됩니다. 블록 보조금을 통해 분배되는 인플레이션 보상은 시간이 지남에 따라 감소하도록 프로그래밍되어 있으며, 여러 소각 및 싱크 메커니즘이 지속적으로 유통에서 토큰을 제거하여, 궁극적으로 네트워크 활동이 확장됨에 따라 순 디플레이션 효과를 생성합니다.

이러한 디플레이션의 주요 원천은 장기 스토리지 수수료에서 발생합니다. 연장된 기간 또는 영구 원장(예: 2 주 이상)에 저장된 데이터에 대해 지불된 금액은 프로토콜의 기금 컨트랙트, 즉 채굴자들의 미래 스토리지 의무를 충당하기 위해 설계된 예비금으로 할당됩니다. 이 기금 토큰들은 네트워크의 데이터 영속성 보장을 보존하기 위해 사실상 무기한 락업되기 때문에, 해당 \$IRYS 공급량을 활성 유통에서 제거하는 단방향 경제 싱크로 기능합니다. 이 메커니즘은 저장된 데이터의 총량이 증가함에 따라, 기금에 의해 가해지는 디플레이션 효과 또한 커지도록 보장합니다.

두 번째 디플레이션 벡터는 실행 수수료 소각 메커니즘을 통해 도입됩니다. IrysVM에서 실행되는 모든 프로그래밍 가능한 데이터 또는 스마트 컨트랙트 트랜잭션에 대해, 관련된 실행 수수료의 50%는 영구적으로 소각되며, 나머지 부분은 블록 생성자에게 보상으로 지급됩니다. 이 설계는 토큰 공급 축소를 네트워크 활용도와 직접 연결하며, 이는 온체인 활동, 데이터 구성 가능성, 프로그래밍 가능한 워크로드가 증가함에 따라 비례하는 몫의 토큰이 소멸될 것임을 의미합니다.

기금 락업과 실행 수수료 소각이라는 이 두 메커니즘은 함께 프로토콜 사용과 토큰 희소성 사이의 피드백 루프를 확립합니다. 초기 단계에서는 감소하는 블록 보상이 디플레이션 압력을 증폭시키며, 후기 단계에서는 스토리지와 연산에 대한 지속적인 수요가 지속적으로 축소되는 유통 공급량을 보장합니다. 그 결과는 장기 참여, 데이터 성장, 애플리케이션 활동 모두가 \$IRYS의 구매력과 지속 가능성을 강화하는 경제 시스템입니다.

9 수수료

9.1 최소 수수료 매개변수

Irys 는 네트워크 스팅을 완화하고 트랜잭션 수수료가 원자 단위로 쉽게 표시될 수 있도록 최소 수수료를 구현합니다.

최소 수수료는 Irys 의 가격 근사 메커니즘에 의해 결정된 \$0.001(1 센트의 1/10)입니다.

영구 데이터를 게시하기 위한 인그레스-증명 제공자에게도 동일한 최소 수수료가 지불됩니다.

9.2 기간 수수료

기간 스토리지의 가격 모델은 엔 에포크 동안 10 개의 복제본에 대한 데이터 스토리지 제공 비용을 결정합니다.

Pricing Parameter		Value	
Annualized Cost of operating 16TB HDD		\$44.00	
Number of Replicas		10	
Calculation		Value	
Daily Cost per TB		\$0.0075	
Daily Cost of 16TB HDD		\$0.12	
Total Fee Per Epoch Storage Price (TB)		\$0.0753	
Total Fee Per Epoch Storage Price (GB)		\$0.00007358	
Epoch Fee Calculator		Data Size (TB)	Total Fee Per Epoch Storage Price
1 Epoch		1	\$0.0753
5 Epochs		1	\$0.3767

Pricing Parameter		Value
Annualized Cost of operating 16TB HDD		\$44
Number of Replicas		10
Calculation		Value
Daily Cost per TB		\$0.0075
Daily Cost of 16TB HDD		\$0.12
Total Fee Per Epoch Storage Price (TB)		\$0.0753
Total Fee Per Epoch Storage Price (GB)		\$0.00007358
Epoch Fee Calculator		
Number of Epochs	Data Size (TB)	Total Fee Per Epoch Storage Price
1	1	\$0.0753
5	1	\$0.3767

블록 포함에 대해 추가 5% 수수료가 추가됩니다(트랜잭션 데이터 크기에 따라 확장됨).

서브밋 원장의 1TB 기간 데이터 (5 에포크)

기간_수수료 = 기간_비용 + 5%

기간_수수료 = \$0.3767 + 5% = 0.3955 -> \$0.40

서브밋 원장의 1GB 기간 데이터 (5 에포크)

기간_수수료 = \$0.00039

\$0.00039 는 네트워크 최소 수수료인 \$0.01 보다 낮으므로 기간_수수료는 다음과 같이 됩니다: 기간_수수료 = \$0.01

참고: 만료 후 기간 파티션을 재패킹하는 것이 채굴자에게 지속적인 비용을 발생시키는 경우, 이 비용은 정량화되어 기간 데이터 가격에 포함될 것입니다.

9.3 영구 수수료

영구 데이터의 가격 모델에는 고려해야 할 몇 가지 추가 요소가 있습니다. 사용자들이 수세기에 걸친 스토리지 비용을 선불로 지불하기 때문에, 모델은 그 기간 동안 (기술적 발전으로 인한) 물리적 스토리지 비용의 하락을 고려해야 합니다. Irys 는 영구 데이터 가격

책정 요소로 스토리지 비용의 연간 1%라는 매우 안전한 하락률을 선택합니다. (지난 50 년간 관찰된 스토리지 비용 하락률은 연간 25% 이상이었습니다.)

Pricing Parameter	Values
Annualized Cost of operating 16TB HDD	\$44
Safe annual decline in cost of storage (decay rate)	1.00%
Number of Replicas	10
Years of storage paid for	200
Cost Per TB	\$2,381.54
Cost Per GB	\$2.33

영구 데이터는 퍼블리시 원장으로 가는 길에 먼저 서브밋 원장(기간 데이터)을 통과해야 하므로, 수수료에는 서브밋 원장 스토리지 비용도 포함됩니다.

영구 데이터는 10 개의 인그레스-증명이 필요하며, 인그레스-증명은 트랜잭션을 블록에 포함시키는 것에 대한 5% 즉시 보상과 동일합니다. (데이터 크기에 따라 확장되며, 최소 \$0.01 의 수수료 하한을 공유합니다).

1TB 의 영구 데이터

$$\begin{aligned} \text{영구_수수료} &= \text{기간_수수료} + (\text{인그레스_수수료} * 10) + \text{영구_비용} \\ \text{영구_수수료} &= \$0.40 + (\$0.018835 * 10) + \$2,381.56 \rightarrow \$2,382.14 \end{aligned}$$

1GB 의 영구 데이터

$$\begin{aligned} \text{영구_수수료} &= \text{기간_수수료} + (\text{인그레스_수수료} * 10) + \text{영구_비용} \\ \text{영구_수수료} &= \$0.01 + (\$0.01 * 10) + \$2.33 \rightarrow \$2.44 \end{aligned}$$

사용자가 서브밋 원장 기간 동안 데이터를 업로드하지 못하거나 네트워크가 필요한 수의 인그레스-증명을 달성하지 못하는 경우, 5 에포크(서브밋 원장 기간)가 끝날 때 서브밋 원장 트랜잭션이 완료되면 사용자의 인그레스_수수료와 영구_비용이 환불됩니다.

9.4 합의 가격 책정 메커니즘

서브밋 원장에서 영구 원장으로 데이터를 승격시키는 과정은 여러 단계를 포함하며, 이로 인해 영구 데이터에 대한 단계적 지불 모델이 발생합니다. 모든 트랜잭션은, 영구 또는 임시 데이터를 위한 것이든, 처음에 서브밋 원장으로 들어갑니다. 기간 데이터에 대한 지불 과정은 모든 트랜잭션에서 일관되지만, 영구 데이터는 완전한 게시 과정을 장려하기 위해 추가적인 지불이 발생합니다.

9.5 기간 데이터 지불 분배

1. 사용자는 기간_수수료를 포함한 트랜잭션을 게시합니다.
2. 블록 생성자 트랜잭션 포함:
 - a. 블록 생성자는 트랜잭션을 블록에 포함시킵니다.
 - b. 블록 생성자의 잔액은 기간_수수료의 5%만큼 증가합니다.
 - c. 기간_수수료의 나머지 95%는 트레저리에 추가됩니다(블록 헤더에서 추적됨).
3. 사용자는 트랜잭션과 관련된 데이터 청크를 업로드합니다.
4. 청크를 저장하도록 할당된 채굴자들은 서로 가십합니다.
5. 기간 원장 만료 지급:
 - a. 서브밋 원장에서 트랜잭션이 만료될 때(청크를 포함하는 파티션이 에포크 경계에서 재설정될 때), 각 채굴자는 자신의 파티션에서 만료되는 모든 할당된 청크에 대해 자신의 몫(기간_수수료 / 10)을 지급받습니다.
 - b. 전체 16TB 파티션의 경우, 이 지급액은 채굴자당 약 \$0.60 입니다.
 - c. 채굴자들은 이 파티션을 채굴하는 동안 생성하는 모든 블록에서 전체 인플레이션/블록 보상을 계속해서 얻습니다.

9.6 추가 인센티브

이 지불 구조는 채굴자들이 기간 원장에 참여하도록 추가적인 인센티브를 만듭니다:

- 채굴자들은 데이터가 자신의 파티션에서 만료될 때 지급금을 받습니다.
- 채굴자들은 만료 후 파티션을 다시 패킹해야 하므로, 이 추가 수수료는 지속적인 참여와 네트워크 유지를 장려합니다.

9.7 영구 데이터 지불 분배

사용자는 영구 데이터 스토리지에 대해 다음 수수료를 지불합니다:

- 기간_수수료: 기간 스토리지에 대한 표준 수수료

- 영구_수수료: 영구 스토리지에 대한 수수료
- 블록 포함에 대한 기간_수수료의 5%
- 각 인그레스-증명에 대한 기간_수수료의 5%

9.8 수수료 분배

1. 기간_수수료: 일반 기간 데이터 트랜잭션과 동일하게 처리됩니다.
2. 블록 포함 수수료:
 - a. 기간_수수료의 5%가 트랜잭션을 포함하는 블록 생성자에게 즉시 지불됩니다.
3. 인그레스-증명 수수료:
 - a. 제공된 각 인그레스-증명에 대해 기간_수수료의 5%가 지불됩니다.
4. 영구_수수료:
 - a. 200 년 x 10 개 복제본과 연 1%의 스토리지 비용 하락을 포함하는 선불 금액.
 - b. 트레저리에 추가됩니다.

10 서브밋 원장 만료 (에포크 경계) 처리

환불 시나리오

트랜잭션 데이터가 업로드되지 않은 경우:

- 인그레스-증명 수수료와 영구_수수료는 업로더에게 환불됩니다.

승격 시나리오

데이터가 영구 스토리지로 승격된 경우:

- 프로토콜은 인그레스-증명이 있는 모든 영구 트랜잭션을 검사합니다.
- 인그레스-증명 제공자에게 지급합니다.

10.1 에포크 경계 지불 분배 작업

만료되는 각 서브밋 원장 트랜잭션에 대해:

1. 트랜잭션이 퍼블리시 원장을 위한 것인지 확인하기 위해 트랜잭션을 검사합니다.
2. 퍼블리시 원장을 위한 것이라면, 도착했는지 확인합니다:
 - a. 게시됨: 인그레스-증명 제출자에게 5% 보상을 지급합니다.
 - b. 게시되지 않음: 영구_수수료와 인그레스-증명 수수료를 트랜잭션을 게시한 주소로 환불합니다.

3. 완료되는 파티션에 게시된 데이터의 양을 집계합니다.
4. 각 파티션 소유자에게 해당 양의 데이터를 저장한 데 대한 기간_수수료를 지불합니다.

11 로드맵

11.1 프로그래밍 가능한 데이터 확장

프로그래밍 가능한 데이터는, 그 핵심에 있어, 데이터셋이 더 이상 수동적인 기록이 아니라 온체인 애플리케이션을 위한 능동적이고 구성 가능한 빌딩 블록이 되는 새로운 연산 패러다임을 나타냅니다. 그것은 보편적인 데이터 레이어를 확립합니다: 개발자, 사용자, 프로토콜이 검증 가능한 온체인 데이터에 대해 직접 로직을 읽고, 쓰고, 실행할 수 있는 공개적으로 접근 가능한 기반입니다. 스토리지를 실행과 분리하는 전통적인 블록체인과 달리, 프로그래밍 가능한 데이터는 둘 다를 통합하여, 정보, 컴퓨팅, 가치가 애플리케이션 간에 원활하게 흐르는 공유 환경을 생성합니다.

이 모델의 중대한 발전은 프로그래밍 가능한 데이터 레이어-2(PD L2s)를 통해 이루어질 것입니다: Irys의 글로벌 데이터셋과의 무신뢰 상호운용성을 유지하면서 IrysVM의 연산 용량을 확장하는 스케일링 환경입니다. PD L2는 전문화된 실행 도메인으로 기능하여, 상태나 데이터 가용성을 파편화하지 않으면서 병렬 연산, 더 낮은 지연 시간, 도메인 특화 최적화를 가능하게 합니다. 그것들은 검증 가능한 증명을 통해 Irys의 데이터 레이어에 네이티브하게 고정되어, 오프체인에서 수행되는 모든 읽기, 쓰기, 연산이 베이스 레이어의 정식 데이터와 암호학적으로 연결되도록 보장합니다.

이 아키텍처 내에서, Irys 데이터셋은 퍼블릭 및 프라이빗 상태로 나뉩니다. 퍼블릭 상태는 무허가 데이터를 나타냅니다: 누구나 자신의 스마트 컨트랙트나 애플리케이션에 접근하고, 쿼리하며, 통합할 수 있도록 개방되어 있습니다. 이는 프로젝트 간의 구성 가능성을 가능하게 하여, AI 에이전트, 분석 프로토콜, 또는 DePIN 네트워크와 같은 생태계가 공유 데이터셋 위에 구축할 수 있게 합니다. 프로토콜 수준의 라이선싱 기본 요소는 경제적 로직을 데이터 접근에 직접 내장할 수 있어, 창작자들이 프로그래밍 가능한 로열티나 사용료를 통해 사용을 수익화할 수 있게 합니다.

대조적으로, 프라이빗 상태는 영지식 증명 또는 보안 인클레이브와 같은 프라이버시 보존 컴퓨팅 기본 요소들을 활용하여, 민감하거나 독점적인 데이터가 검증 가능하지만 기밀 환경 내에서 처리될 수 있도록 합니다. 이는 조직이나 사용자가 정확성과 감사 가능성에 대한 암호학적 보장을 유지하면서 프라이빗 데이터셋에 대한 연산을 실행할 수 있도록 보장합니다.

궁극적으로, 프로그래밍 가능한 데이터 L2 는 Irys 를 확장성, 프라이버시, 상호운용성이 수렴하는 다계층 실행 생태계로 확장합니다. 그것들은 Irys 를 데이터체인에서, 퍼블릭이든 프라이빗이든 어떤 데이터셋이라도 온체인 경제의 살아있는 구성 요소가 될 수 있는, 보편적인 컴퓨팅 및 조정 레이어로 변환합니다.

11.2 빠른 블록과 빠른 완결성

Irys 위에 애플리케이션을 구축하는 것은 근본적으로 블록 시간, 완결성, 구성 가능성 간의 관계에 달려있습니다. 더 빠른 블록 시간은 상태 업데이트 간의 지연 시간을 줄여, 애플리케이션, 특히 실시간 또는 거의 실시간 데이터에 의존하는 애플리케이션이 온체인 이벤트에 최소한의 지연으로 응답할 수 있게 합니다. 이는 추론 캐싱, AI 모델 조정, 또는 자동화된 데이터 파이프라인과 같은 유스케이스에 중요하며, 여기서 1 초 미만의 응답성은 사용자 경험과 시스템 효율성 모두에 의미 있는 영향을 미칠 수 있습니다.

프로그래밍 가능한 데이터 L2 의 경우, 빠른 완결성은 훨씬 더 필수적이 됩니다. 이러한 레이어-2 환경은 Irys 의 글로벌 데이터셋과의 결정론적 동기화를 유지하기 위해 베이스 레이어로부터의 신속한 확인에 의존합니다. 완결성 시간이 짧을 때, L2 는 자신의 상태를 베이스 체인으로 자주 체크포인트할 수 있어, 일관성을 보장하고 데이터 불일치 또는 롤백의 위험을 줄입니다. 이는 교차 도메인 구성 가능성을 가능하게 하여, 별개의 L2 에 배포된 애플리케이션들이 과도한 지연 시간이나 신뢰 가정을 도입하지 않고도 공유된 Irys 데이터셋을 통해 안전하게 상호 운용될 수 있게 합니다.

실제로, 더 빠른 완결성은 개발자들이 더 상호작용적이고 모듈식인 시스템을 설계할 수 있게 하며, 여기서 한 파티션에 저장된 데이터가 즉시 다른 파티션의 연산이나 로직을 트리거할 수 있습니다. 예를 들어, L2 에서 작동하는 AI 추론 엔진은 베이스 체인에 쓰여진 새로운 센서 데이터를 즉시 소비하고, 예측을 연산하며, 결과를 거의 실시간으로 Irys 에 다시 커밋할 수 있습니다.

이러한 성능 목표에 도달하기 위해, Irys 는 하이브리드 PoW/S 모델을 보완하거나 확장할 수 있는 비잔틴 장애 허용(BFT) 합의 변형들을 계속 탐색할 것이며, 파티션 간의 더 엄격한 블록 확인과 결정론적 완결성을 가능하게 하는 것을 목표로 합니다. 그러한 메커니즘들은 노드들이 작업증명의 확률적 보안을 보존하면서 제한된 시간 내에 블록 유효성에 동의할 수 있게 할 것입니다. BFT 원칙을 합의 파이프라인에 통합함으로써, Irys 는 전파 지연을 최소화하고, 네트워크 응답성을 향상시키며, L2 와 베이스 레이어 간의 구성 가능한 완결성을 보장하는 것을 목표로 합니다.

종합적으로, 이러한 최적화들은 Irys 를 고성능 실행 기반으로 자리매김하게 하여, 현대 클라우드 인프라에서 기대되는 것과 동일한 응답성을 가지면서도 완전한 탈중앙화 네트워크의 신뢰 보장을 갖춘 복잡한, 데이터 주도 애플리케이션을 지원할 수 있게 합니다.