

솜니아(Somina)

해당 번역본은 이용자가 디지털자산 관련 정보를 편리하게 확인할 수 있도록 단순 참고용으로 번역된 것이며, 투자 권유를 목적으로 하지 않습니다.

그 내용이 정확하지 않을 수 있으므로 정확한 정보 습득을 위해서는 원문을 참고하시거나 원문 작성 주체 측에 문의하시기 바랍니다.

두나무는 해당 번역본 내용의 진실성 및 정확성을 보장하지 않고, 제공된 정보에 의한 투자 결과에 대하여 어떠한 책임도 지지 않으며, 투자 손실은 투자자에게 귀속됩니다.

Somnia-미션

웹3 특성을 갖춘 실시간 대중 소비자 애플리케이션을 웹2 수준으로 구축할 수 있도록 하여, 보다 개방적이고 공정한 인터넷을 창조하는 것.

핵심 목표

- 웹2 사용 수준(수백만 동시 접속자, 초당 수십만 건의 트랜잭션)까지 확장 가능한 애플리케이션
- 인터넷에 접근할 수 있는 누구나 사용할 수 있는, 검열 저항적이고 개방된 시스템
- 단일 주체가 가치를 통제하거나 독점하지 않는 구조
- 참여자 간에 가치가 공유되는 조합 가능한 시스템
- 사용자와 자산이 플랫폼 간에 자유롭게 이동할 수 있어 창작자와 사용자가 선택의 자유를 가지는 환경

보다 개방적이고 공정한 세상.

핵심 원칙

온체인 경험

우리는 완전 온체인 방식의 실시간 대규모 애플리케이션을 가능하고 실용적으로 만드는 기술 개발에 전념하고 있습니다. 이는 조합 가능한 개방형 인터넷을 구축하는 데 필수적이라고 믿습니다.

단일체보다 다중세계

우리는 각기 다른 상호 연결된 경험을 가진 애플리케이션들의 집합이 하나의 단일체보다 더 바람직하다고 믿습니다. 이는 마치 전 세계 공동체 속의 국가들처럼, 상호 성장을 촉진하는 유틸리티를 공유합니다.

조합성의 강화

우리 비전의 중심에는 조합성의 원칙이 있습니다. 이는 빌더들이 서로의 작업 위에 구축할 수 있는 능력을 의미하며, 전체 산출물이 부분의 합을 능가하는 협업 문화를 창출합니다.

빌더 권한 부여

우리는 빌더들이 지속 가능한 참여 및 소유 모델을 구축할 수 있도록 자유를 보장할 것입니다. 이는 디지털 자산에 대한 권리를 보호하고, 플랫폼의 제약에 의해 혁신이 억제되지 않는 환경을 조성하는 것을 포함합니다.

모두를 위한 접근성

가상 사회가 지닌 잠재력은 보편적 접근성에 있습니다. 이는 콘텐츠 창작과 참여의 장벽을 제거하며, 인터넷 접속만으로 누구나 참여하고 성공할 수 있는 평등한 환경을 의미합니다.

문제

웹3는 탈중앙화 금융의 새로운 시대를 열었고, 금융 접근성을 민주화했습니다. 그러나 대중 소비자용 애플리케이션 생성에는 실패하였으며, 여전히 금융 중심에 머물러 있습니다. 우리는 블록체인 기술이 새로운 개방형 인터넷 애플리케이션의 미래를 위한 초석이라고 믿습니다.

오늘날 온체인으로 구축할 수 있는 것에는 한계가 있습니다. 이는 애플리케이션 실행 비용부터 기존 블록체인의 근본적인 성능 제한까지 다양한 요소에 의해 제약됩니다. 우리는 새로운 기술이 일반적으로 웹2 기반에서만 가능했던 실시간 애플리케이션의 새로운 범주를 가능하게 할 수 있다고 믿습니다. 이러한 시스템을 온체인으로 구축할 수 있다면, 비즈니스와 사용자가 온라인 플랫폼 간에 자유롭게 이동할 수 있게 되어, 우리가 '진정한 가상 사회'라 부르는 환경이 실현될 것입니다.

우리는 이를 달성하기 위해 Somnia를 구축하고 있습니다. Somnia는 빠르고 비용 효율적인 EVM 기반 블록체인으로, 초당 400,000건 이상의 트랜잭션 처리, 1초 미만의 확정 시간, 낮은 수수료를 갖춘 완전한 EVM 호환 레이어 1 블록체인입니다. 이를 통해 새로운 형태의 온체인 애플리케이션들이 가능해질 것입니다. 초기에는 게임, 메타버스, 소셜 경험에 집중하고 있지만, 그 사용 사례는 이들 분야를 훨씬 넘어설 것입니다. 우리는 우리가 만든 기술로 무엇을 구축할 수 있을지 100% 확신하지 못할 정도입니다.

현재 상태

많은 블록체인들이 EVM 바이트코드 표준을 지원하지만, 처리량(throughput) 측면에서는 여전히 부족한 점이 많습니다. 아래는 전형적인 UniswapV3 트랜잭션(DEX 스왑 기준)의 초당 트랜잭션 수(TPS)를 나타낸 수치입니다. 해당 트랜잭션은 130k gas를 사용합니다. 보시다시피, 대부분의 체인은 약 200 DPS, 혹은 하루 약 2천만 DPS 수준에 머물고 있습니다. 이 수치는 [이 글](#)에서 인용되었습니다:

체인	평균 DPS	최대 DPS	확정성(Finality)
ETH	9.19	18.38	66초
Polygon	47.67	95.33	변동형
AVAX	31.65	175.68	3.7초
BNB	194.6	194.6	75초

(계속) 현재 상태

이는 실제 세계에서 관측된 수치와 유사합니다 (출처: [Chaininspect](#)). Uniswap 트랜잭션은 보다 복잡한 사용자 인터랙션이며 다수의 트랜잭션으로 구성되어 있으므로, 위 표보다 수치가 더 높게 나타납니다.

체인 이름	최대 기록 TPS	블록 시간	확정까지 소요 시간
Base	293	2초	16분
BNB Chain	1731	3초	7.5초
Polygon	429	2.22초	4분 16초
Arbitrum	944	0.25초	16분
Ethereum	62.34	12.08초	16분
Optimism	67.41	2초	16분
Avalanche	92.74	2.03초	0초

최근 벤치마크는 초당 가스 처리량(gas per second)을 기준으로 측정되고 있습니다.

선택된 EVM 체인	초당 가스 처리량	블록당 타겟 가스량 (공급)	블록 시간
opBNB	100.0 mg/s	100M	1.0초
BSC	46.6 mg/s	140M	3.0초
Polygon	7.5 mg/s	15M	2.0초
Avalanche C-Chain	7.5 mg/s	15M	2.0초
Arbitrum One	7.0 mg/s	1.75M	0.25초
Base	5.0 mg/s	15M	2.0초
Optimism Mainnet	2.5 mg/s	5M	2.0초
Conduit*	2.5 mg/s	5M	2.0초
Ethereum L1	1.25 mg/s	15M	12.0초

* 대부분의 Conduit 체인을 대표하는 수치입니다.

이 수치는 [이 글](#)에서 인용되었습니다. 이를 DPS로 환산하면, 1 mg/s당 약 6.5 DPS 수준입니다. 따라서 위 목록에서 가장 빠른 체인조차 이론상 최대 650 DPS(약 3700 TPS)에 해당합니다.

EVM 생태계를 벗어나면 훨씬 더 높은 성능을 달성할 수 있습니다. 다만, 이 TPS 수치는 앞선 수치들과는 테스트 방식이 다르므로 동일하게 비교할 수는 없습니다.

(계속) 현재 상태

체인	TPS	확정성(Finality)
Aptos	30,000 (출처)	0.9초 (출처)
SUI	11,000–297,000 (출처)	480밀리초 (출처)
Solana	1608 (출처)	12.8초 (출처)

Aptos와 SUI의 실제 환경에서의 라이브 테스트는 아직 이러한 수준에 도달하지 못했습니다. 이 수치들은 해당 체인들의 벤치마크 결과를 기반으로 한 것입니다. Solana의 수치는 Solana 공식 웹페이지에 있는 값과는 다릅니다. 본 문서에서는 실측 수치를 사용하였습니다. 위에서 사용된 동일한 방법론에 따르면, Solana의 수치는 초당 273.34건의 스왑으로 상당히 낮은 성능을 보입니다.

이러한 성능 향상에도 불구하고, 우리는 여전히 EVM의 중요성을 믿습니다. EVM을 중심으로 이미 방대한 개발자 및 콘텐츠 생태계가 형성되어 있으며, 그 생태계에 접근하지 못하는 것은 큰 기회를 놓치는 것입니다. 이들 체인은 성능을 확보하기 위해 그런 트레이드오프를 선택해야 했습니다. 하지만 EVM 상에서 비슷한 수준의 성능을 달성할 수 있다면 어떨까요?

현재 EVM 기반 블록체인 구현 방식에서 우리가 직면하고 있는 병목 현상은 다음과 같습니다:

- **실행 속도** – 스마트 컨트랙트 코드를 실행하고 블록을 생성하는 속도.
- **스토리지** – 체인의 과거 데이터를 저장하고 조회하는 기능. 이더리움의 [EIP-4844](#)는 ETH 및 기타 L2 체인의 저장 비용을 크게 개선한 최근 업그레이드입니다. 그러나 우리는 여전히 블록체인에서의 데이터 읽기/쓰기 성능 개선이 필요합니다.
- **대역폭** – 높은 트랜잭션 처리량을 달성할 때, 네트워크 상의 노드 간 데이터를 전송하는 데 필요한 대역폭.

개요

Somnia 블록체인은 기존 EVM 체인들과 비교하여 성능을 수십 배 향상시킬 수 있도록 하는 여러 혁신을 내포하고 있습니다:

- **MultiStream 합의 알고리즘** – [Autobahn BFT](#)에 영감을 받은 지분증명 기반 부분 동기 BFT 프로토콜.

- **독립 데이터 체인** – 각 검증자는 자신만의 블록체인, 즉 “데이터 체인”을 운영합니다. 이를 통해 개별 블록체인의 독립적인 블록 생성이 가능하며, 해당 체인 내부에는 별도의 합의 메커니즘이 필요 없습니다. 이 구조는 데이터 처리 워크플로우를 간소화합니다.
- **합의 체인** – 모든 데이터 체인의 블록 헤드를 수집하는 별도의 블록체인이 존재하며, 수정된 PBFT 알고리즘을 통해 지분증명 합의를 수행합니다. 이 구조는 데이터 생성과 합의 과정을 분리하여 전체 시스템의 효율을 크게 향상시킵니다.
- **컴파일된 바이트코드** – EVM 바이트코드를 고도로 최적화된 네이티브 코드로 변환함으로써, Somnia는 수작업으로 작성된 C++ 스마트 컨트랙트에 근접한 실행 속도를 달성합니다. 이를 통해 단일 코어에서 수백만 건의 트랜잭션을 초당 실행할 수 있습니다.
- **빠르고 예측 가능한 데이터베이스 성능** – Somnia는 IceDB라는 커스텀 데이터베이스를 사용합니다. 이 데이터베이스는 예측 가능한 읽기/쓰기 성능을 위한 성능 보고서를 제공하며, 평균 15–100 나노초 수준의 읽기/쓰기 연산을 실현하는 독자적 아키텍처와 스냅샷 기능을 내장하고 있습니다.
- **고급 압축 기술** – Somnia의 데이터 체인 아키텍처는 스트리밍 압축을 가능하게 설계되어 데이터 처리량을 극대화합니다. Somnia는 이를 BLS 서명 집계와 결합하여 매우 높은 압축률을 달성하며, 방대한 트랜잭션 데이터 처리량을 지원합니다. 이를 통해 소위 “대역폭의 한계”를 넘어서는 이론적 성능을 가능하게 합니다.

MultiStream 합의 알고리즘

Somnia에서는 각 검증자가 자신만의 블록체인인 데이터 체인을 발행합니다. 이 혁신은 2024년 백서 “[Autobahn: Seamless high speed BFT](#)”에서 영감을 받았습니다. 모든 데이터 체인은 완전히 독립적이며, 각 데이터 체인의 블록은 바이트(blob of bytes)로 이루어진 덩어리를 포함합니다. 데이터 체인에 블록을 추가하는 것은 해당 체인의 소유자인 검증자만 가능하며, 포크를 방지하거나 유효하지 않은 블록 생성을 막기 위한 안전 메커니즘은 존재하지 않습니다. 다시 말해, 데이터 체인에는 **전혀 합의 메커니즘이 존재하지 않습니다**.

Somnia는 여기에 **합의 체인(consensus chain)**을 도입합니다. 각 합의 블록에는 모든 데이터 체인의 현재 헤드가 포함됩니다. 이 합의 체인은 수정된 PBFT 합의 알고리즘을 사용하며, 전형적인 지분증명(PoS) 합의 구조를 따릅니다. 합의 체인은 각 데이터 체인의 최상위 블록을 포함함으로써, 검증자가 자신의 데이터 체인을 포크하는 것을 방지하는 **완전한 보안성**을 제공합니다. 각 합의 블록은 의미적으로 해당 시점에서 진전된 모든 데이터 체인의 모든 트랜잭션을 포함하게 됩니다. 이들 데이터 체인은 결정론적 의사난수 방식으로 정렬되어, 전역적으로 정렬된 바이트 스트림을 생성하고, 이는 모든 데이터 체인에서 실행됩니다.

이 구조는 새로운 데이터의 생성 및 전파(데이터 체인의 진전)와 합의 알고리즘을 완전히 분리합니다. 이는 아래 “고급 압축 기술” 섹션에서 설명된 여러 핵심 이점을 제공합니다 (예: 스트리밍 압축). 이를 통해 이 데이터 프론트엔드는 초당 거의 1기가비트에 달하는 트랜잭션 데이터를 게시할 수 있습니다.

고속 순차 실행

병렬 실행의 문제점

많은 최신 블록체인들은 병렬 처리(parallelism)를 통해 실행 성능을 확장하려 시도해왔습니다. 이는 서로 관련 없는 트랜잭션들을 서로 다른 코어에서 병렬로 실행하는 방식입니다. 블록 내 트랜잭션들이 서로 무관할 경우 이 방식은 잘 작동하지만, 동일한 상태(state)를 수정하는 경우에는 실패하게 됩니다.

하지만 현실적으로, 부하 스파이크(load spike)는 대부분 어떤 특정한 이벤트로 인해 발생하며, 이로 인해 **트랜잭션 간의 높은 상관관계가 생깁니다**. 이는 일상적인 실행 시기에는 존재하지 않던 특수한 상태입니다.

예를 들어:

이더리움의 가장 큰 부하 스파이크 중 하나는 Otherside Otherdeed 민팅 시기였습니다. 이 이벤트 동안, 각 블록의 대부분 트랜잭션이 동일한 상태를 수정하고 있었습니다 (Otherdeed를 민팅했기 때문). 이 경우 병렬 실행은 작동하지 않았을 것입니다.

DEX에서는 하루 동안 다양한 무관한 자산 쌍에 대한 거래가 이루어지지만, **실제 부하 스파이크는 특정 자산 쌍의 변동성(volatility)에서 발생합니다**. 이들 거래는 모두 동일한 상태를 수정하게 되어 병렬 실행이 효과가 없습니다.

다시 말해, **병렬 실행은 가장 필요한 순간에 작동하지 않습니다**.

이러한 관찰에 따라 Somnia는 병렬 실행에 의존하기보다, 단일 코어를 극한으로 빠르게 만들기로 선택했습니다.

EVM 컴파일

Somnia는 EVM 트랜잭션을 실행합니다. 이는 다중 코어에서 병렬화를 시도하는 대신, **단일 코어의 속도를 극도로 높여 스케일링을 달성합니다**.

EVM은 상대적으로 단순한 스택 기반 아키텍처입니다. 스택 기반 바이트코드는 암시적 스택 구조 덕분에 레지스터 기반 바이트코드보다 크기가 작을 수 있지만, **많은 중복 연산을 포함합니다**.

VM의 바이트코드를 실행하는 방법에는 크게 두 가지가 있으며, **인터프리터 방식**과 **네이티브 실행 방식(JIT 또는 AOT 컴파일로 세분화 가능)**이 있습니다. 전자는 소프트웨어 상에서 VM을 직접 시뮬레이션하는 방식이며, 후자는 바이트코드를 네이티브 명령어로 변환하여 CPU가 직접 실행하게 합니다.

이더리움과 대부분의 EVM 블록체인은 EVM 바이트코드를 실행하기 위해 인터프리터 VM을 사용합니다. 이 방식은 대개 비교적 단순한 구현으로, 자체 스택을 유지하며 각 연산을 반복적으로 순회하고, 연산의 기능을 룩업 테이블에서 조회하는 방식입니다. **이는 네이티브 실행 방식에 비해 매우 느립니다**.

또한 Solidity 및 기타 이더리움 컴파일러들은 종종 가스 비용보다 바이트코드 크기 최적화를 우선시합니다. 예를 들어, 큰 상수를 인라인으로 포함시키는 대신, 이를 생성하는 코드를 삽입합니다. 이로 인해 컴파일 시 정적으로 계산할 수 있었던 연산이 런타임에 수행되며 **불필요한 실행 비용이 추가**됩니다.

이러한 중복성을 제거하기 위해 Somnia는 자체 EVM 컴파일러를 내장하고 있으며, EVM 바이트코드를 x86 네이티브 코드로 변환합니다. 이는 네이티브 속도(수작업으로 작성된 C++ 코드에 준하는 성능)에 근접합니다. 벤치마크에 따르면, 이 방식은 **ERC-20 전송을 수백 나노초 내에 실행**할 수 있으며, 단일 코어에서 수백만 TPS 달성이 가능합니다.

그러나 공짜는 없습니다. 컴파일 과정은 상대적으로 비용이 높기 때문에, 이 방식은 **빈번하게 호출되는 컨트랙트에만 적용**하며, 나머지는 일반적인 인터프리터 방식의 EVM으로 되돌아갑니다.

하드웨어 수준 병렬성

앞서 언급했듯, "**병렬 실행**", 즉 무관한 트랜잭션을 서로 다른 코어에서 병렬로 처리하려는 시도는, 정작 **부하 스파이크가 발생했을 때 상호 연관된 트랜잭션을 처리하지 못하기 때문에 무용지물**이 됩니다. 이러한 병렬성은 **소프트웨어 병렬성**이라고도 합니다.

이러한 이유로 Somnia는 무관한 트랜잭션을 병렬로 실행하려 하지 않습니다. 대신 Somnia는 **각 개별 트랜잭션을 CPU가 하드웨어 수준에서 병렬 처리할 수 있도록** 지원합니다.

이는 현대 CPU가 제공하는 단일 코어 내 병렬성을 활용하여 **모든 개별 트랜잭션의 실행 속도를 향상**시키는 방식입니다. 이 기술은 동일한 상태를 수정하는 트랜잭션에도 적용 가능하기 때문에, 부하 스파이크 상황에서도 효과를 발휘합니다.

그렇다면 하드웨어 수준 병렬성이란 무엇일까요?

현대의 CPU 코어는 마치 어셈블리 명령어를 순차적으로 실행하는 것처럼 보이지만, 실제로는 이들 명령어를 **완전히 비순차적(out-of-order)**으로, 그리고 종종 **병렬로 실행**합니다.

예를 들어, 메모리에서 값을 읽는 동안 CPU는 **앞으로 필요한 연산을 미리 진행**합니다. 이는 개발자에게는 완전히 투명하며, 성능 향상에 큰 기여를 합니다. 예를 들어, ERC-20 토큰 스왑의 초기 절차를 살펴보면 다음과 같습니다:

1. 보낸 사람 계정을 해싱한다.
2. 이 해시로 보낸 사람의 잔액을 조회한다.
3. 받는 사람 계정을 해싱한다.
4. 이 해시로 받는 사람의 잔액을 조회한다.
5. ...

계정 해싱 속도가 150ns, 잔액 조회(단일 RAM 읽기)가 100ns라고 가정해 봅시다.

CPU가 이 과정을 순차적으로 실행하면 총 소요 시간은 $150 + 100 + 150 + 100 = 500\text{ns}$ 가 됩니다. 그러나 이 연산이 네이티브 코드로 컴파일되어 실행된다면, **CPU는 1번과 2번을 3번과 4번과 병렬로 처리**하여, 전체 실행 시간이 **250ns**로 절반으로 줄어듭니다.

문제는, **EVM 인터프리팅 방식은 CPU가 이러한 어셈블리 병렬 실행을 하지 못하도록 막는다는 점**입니다. CPU의 실리콘 성능을 진정으로 활용하기 위해서는 **네이티브 어셈블리를 실행해야 합니다**. Somnia의 컴파일러와 데이터베이스는 스마트 컨트랙트의 **하드웨어 수준 병렬성(hardware-level parallelism)**을 실현하기 위해 설계되었습니다.

Somnia의 IceDB

Somnia는 자체 데이터베이스인 **IceDB**를 포함하고 있습니다. IceDB는 다음의 세 가지 핵심 특성을 갖고 있습니다:

1. 결정론적 성능
2. 읽기 승격 기능이 포함된 인메모리 캐시
3. 내장된 스냅샷 기능

결정론적 성능

대부분의 블록체인은 LevelsDB나 RocksDB와 같은 임베디드 데이터베이스를 사용합니다. 이러한 데이터베이스는 보통 **쓰기 처리량(write throughput)**을 우선적으로 최적화하며, 일정 시점이 되면 모든 데이터를 디스크로 플러시하여 RAM 사용량을 일정하게 유지합니다.

이로 인해 이러한 데이터베이스에서 읽기 작업은 **RAM 또는 디스크의 여러 위치 중 어디서든 발생할 수 있습니다**. 값이 어디에 저장되어 있는지에 따라 **레이턴시는 최대 1000배까지 차이**가 날 수 있습니다.

그렇다면, 읽기 1회에 대해 사용자가 지불해야 할 가스는 얼마여야 할까요?

모든 읽기가 최악의 경우 디스크를 여러 번 조회한다고 가정해야 할까요? 아니면 일부는 메모리에서 조회된다고 가정해야 할까요?

전자의 경우 블록체인의 처리 속도가 급격히 저하되고, 후자의 경우 공격자가 체인을 의도적으로 느리게 만들 수 있습니다.

각 노드에서 실제 읽기 시간에 따라 사용자의 가스 요금을 계산하는 것은 어떨까요?

안타깝게도 이 방법은 **노드 간 결정론적이지 않기 때문에 불가능**합니다. (LevelsDB나 RocksDB 등은 데이터를 결정론적인 위치에 저장하려는 시도를 하지 않습니다.) 따라서, 해당 정보를 바탕으로 가스 비용을 조정할 수 없습니다.

Somnia의 IceDB는 **처음부터 완전히 결정론적인 성능**을 갖도록 설계되었습니다. IceDB에 대한 모든 읽기 및 쓰기 작업은 결과뿐만 아니라 **“성능 보고서(performance report)”**를 반환합니다. 이 보고서는 RAM에서 몇 개의 콜드 캐시 라인이 읽혔는지, SSD에서 몇 개의 디스크 페이지가 읽혔는지를 **정확하게 기술**합니다.

이 정보가 **결정론적**이기 때문에, 사용자가 시스템에 가한 실제 부하에 기반하여 가스를 청구할 수 있습니다. 자주 조회되는 데이터를 읽은 트랜잭션은 RAM에서 처리되므로 **더 적은 가스를 사용하며**, 블록에 더 많은 트랜잭션을 포함시킬 수 있습니다.

향상된 읽기/쓰기 캐시

앞서 언급했듯, 디스크에 데이터를 저장하는 데이터베이스는 **자주 접근되는 데이터를 RAM에 보관하는 인 메모리 캐시**를 운영하여 낮은 레이턴시로 응답하려고 합니다.

대부분의 데이터베이스는 읽기 혹은 쓰기 중 **하나에만 최적화**되어 있는 경우가 많습니다. 그러나 Somnia는 **읽기와 쓰기를 모두 효율적으로 처리할 수 있는 캐시**를 구현하였습니다. 이로 인해 IceDB의 평균 읽기/쓰기 속도는 **15-100 나노초** 사이를 유지합니다.

내장된 스냅샷 기능

블록체인은 모든 노드가 주기적으로 **특정 블록 이후의 상태를 완전히 일치시키는 것**을 요구합니다.

대부분의 블록체인은 Merkle 트리와 같은 데이터 구조를 사용하여 **블록체인의 모든 데이터를 포괄하는 단일 상태 해시**를 계산합니다. Merkle 트리는 각 쓰기 작업마다 해당 해시를 효율적으로 갱신할 수 있으며, 데이터베이스에 직접 접근하지 않는 제3자에게 상태 증명을 제공할 수 있습니다.

일반적으로, 대부분의 블록체인은 이 Merkle 트리의 각 노드를 **데이터베이스의 일반 키(key)**로 저장합니다. 이로 인해 트리에서의 읽기 또는 쓰기 작업은 매우 높은 레이턴시를 유발하게 됩니다. 왜냐하면 각 노드 접근이 임베디드 데이터베이스의 비용 높은 쿼리로 이어지기 때문입니다.

하지만 실제로, 이러한 데이터베이스가 내부적으로 사용하는 구조인 **로그 구조 병합 트리(LSM Tree)**는 이미 나무 구조에 적합하며, 방대한 데이터 블록이 디스크에 **불변 상태(immutable)**로 저장됩니다. IceDB는 이 **불변 구조를 활용**하여 Merkle 트리를 키-값 추상화로 별도로 저장하지 않고도 **1급(state-first-class) 스냅샷 기능**을 지원합니다.

이로 인해 IceDB의 성능은 크게 향상되며, 각 값에 대한 **오버헤드도 최소화**됩니다.

고급 압축 기술

트랜잭션이 많아지면, 노드 간 주고받아야 할 데이터도 많아집니다.

초당 수십만에서 수백만 건의 트랜잭션을 처리하게 되면, **막대한 양의 데이터**가 생성되기 시작합니다.

표준적인 ERC-20 전송은 모든 구성 요소를 고려했을 때 약 **200바이트**의 크기를 가집니다.

이제 초당 100만 건의 ERC-20 스왑이 발생한다고 가정해 봅시다.

이는 **초당 약 190MB**, 또는 **1.5Gbps**의 전송량입니다.

이러한 양은 퍼블릭 인터넷을 통해 전달되기 어렵기 때문에, **매우 중앙화된 체인**을 만들거나 **트랜잭션 속도를 제한**할 수밖에 없습니다.

멱함수 분포(Power law distribution)

그러나 이 바이트들에는 **많은 중복 정보**가 존재합니다.

정보를 전송하는 데 이론적으로 필요한 비트 수는 해당 정보가 발생할 확률의 **로그값**에 비례합니다.

실제로, **어떤 계정이 트랜잭션을 생성했는지, 어떤 컨트랙트가 실행되었는지, 호출된 메서드의 인자** 등은 **매우 불균형적인 확률 분포**를 갖습니다 (종종 멱함수 분포를 따릅니다).

즉, 일부 소수의 계정이나 컨트랙트가 전체 데이터의 상당 부분에서 반복적으로 나타납니다.
예를 들어, 특정 컨트랙트가 전체 트랜잭션의 10%에서 호출된다면, 해당 주소는 **3.3비트로 인코딩**할 수 있습니다. 이는 **압축되지 않은 20바이트 주소 대비 48배의 압축률**입니다.

스트리밍 압축

그렇다면 노드 간 이동하는 트랜잭션 데이터를 어떻게 효과적으로 압축할 수 있을까요?
압축에는 크게 두 가지 형태가 있습니다:

- 블록 압축(Block compression)
- 스트리밍 압축(Streaming compression)

블록 압축은 단일 데이터 블록을 받아, 해당 블록만으로 압축을 해제할 수 있도록 처리합니다.
이는 일반적인 압축 방식이며, ZIP이나 TAR 파일과 유사합니다.
이 방식은 클라이언트가 어떤 정보를 보유하고 있는지 또는 데이터가 어떤 순서로 전송되는지에 대해 **압축기가 가정할 필요가 없기 때문에** 매우 편리합니다.

스트리밍 압축은 송신자와 수신자가 이전에 압축 및 해제한 **데이터 이력을 공유하고 있다고 가정**합니다.
이를 통해 내부적으로 **암묵적인 정보(implicit state)**를 유지하며, 전송할 필요 없이 참조할 수 있습니다.
예를 들어 “3.456MB 이전에 등장한 주소를 재사용하라”고 표현할 수 있습니다.
따라서 스트리밍 압축은 블록 압축보다 **훨씬 높은 압축률**을 달성할 수 있습니다.

단점은, **송신자와 수신자가 정확히 동일한 데이터 흐름을 공유해야 하며**,
이 ‘암묵적 상태’를 다른 머신이나 프로세스로 이전하려면 **많은 대역폭이 필요**하다는 점입니다.
즉, 전체 데이터 스트림은 **동일한 프로세스 내에서 압축되어야 합니다**
(압축 알고리즘에 따라 해결책이 있긴 하지만, 대부분 **높은 CPU 자원**을 필요로 합니다).

이는 일반적인 블록체인 구조에서는 어려운 점입니다.
왜냐하면 대부분의 블록체인은 **각 블록이 서로 다른 머신에 의해 제안되기 때문**입니다.
이로 인해 블록체인은 보통 **블록 기반 압축**만 사용할 수 있고, 이는 압축률을 크게 제한합니다.

Somnia는 스트리밍 기반 압축을 지원하는 **합의 및 데이터 가용성 알고리즘**을 가지고 있습니다.
각 검증자는 **자신만의 데이터 스트림을 자신의 블록체인(데이터 체인)에 게시**할 책임이 있으며,
앞서 소개된 데이터 체인 개념이 여기에 해당합니다.
이 데이터 체인은 **동일한 프로세스가 일관된 방식으로 데이터 스트림을 게시하기 때문에**,
스트리밍 압축이 가능해지는 구조입니다.

해시와 서명

이더리움 트랜잭션에는 먹함수 분포를 따르는 데이터가 많기 때문에 압축이 잘 됩니다.
그러나 두 가지 예외가 존재합니다: **해시(hash)**와 **서명(signature)**입니다.

이들은 정의상 트랜잭션의 단 하나의 비트만 바뀌어도 결과가 완전히 달라지기 때문에, **균등 분포를 갖고 있으며 압축이 불가능합니다.**
(Ethereum 트랜잭션은 재실행 공격을 막기 위해 nonce를 포함하므로 절대 동일하지 않습니다.)

트랜잭션 해시는 간단합니다.
정의상 해당 트랜잭션 데이터를 기반으로 재생산 가능하기 때문에,
전송하지 않아도 되고 수신 측에서 재계산이 필요합니다.

서명은 더 어렵습니다.
트랜잭션마다 서명이 반드시 포함되어야 하고,
압축이 불가능하기 때문에 이는 트랜잭션 데이터의 **압축률을 심각하게 제한합니다.**

하지만 **BLS 서명 스킴**을 사용하면 서명을 **집계(aggregate)** 할 수 있습니다.
BLS 서명은 다수의 개별 서명을 **단 하나의 서명으로 집계**할 수 있어,
얼마나 많은 트랜잭션이든 상관없이 고정 크기의 서명을 만들 수 있습니다.

Somnia는 BLS 서명 집계를 활용하여 **서명 검증 속도를 높이고,**
무엇보다 **트랜잭션 데이터의 압축률을 극적으로 향상**시킵니다.

BLS 디지털 서명

이 **집계 암호화**는 선택적으로 트랜잭션들을 **배치(batch)** 형태로 제출할 수 있게 하며,
해당 배치의 모든 트랜잭션 서명을 검증하는 비용이 **단일 트랜잭션 서명을 검증하는 비용과 거의 동일**합니다.
큰 배치를 사용할 경우, **서명 검증 비용은 수십 배에서 수백 배 이상 절감**될 수 있습니다.

대역폭의 대칭성

일반적인 블록체인에서는 한 명의 검증자가 각 블록을 제안하고 해당 '슬롯'의 트랜잭션 데이터를 모든 피어에게 게시할 책임이 있습니다. 블록체인 처리량이 초당 X바이트이고 초당 B개의 블록과 N개의 피어가 있는 경우, 리더는 $1 / B$ 초 안에 $N * X / B$ 바이트를 전송해야 하므로 초당 $N * X$ 바이트의 업로드 속도가 필요합니다(일부 블록체인은 트리 구조로 분산되며, 이 경우에도 분기율에 X를 곱한 값이 필요합니다).

이에 비해 소니아는 모든 피어가 자신의 데이터 샤드를 게시하기 때문에 각 피어는 $1 / B$ 초에 $X / (N * B)$ 바이트를 게시해야 하므로 초당 X 바이트의 업로드 속도가 필요합니다. 또한 각 블록에 대해 다른 모든 사람의 데이터를 다운로드해야 하므로 초당 $N * (X / N)$ 바이트, 즉 X의 다운로드가 필요합니다.

전체 데이터 전송량은 줄어들지 않지만 대역폭 프로파일은 모든 피어에서 항상 대칭을 이루도록 상각되었습니다. 어떤 피어도 블록체인 자체의 대역폭보다 빠른 속도로 업로드할 필요는 없지만, 블록을 제안할 때만 업로드하는 것이 아니라 지속적으로 업로드해야 합니다. 이를 통해 전체 블록체인 처리량이 피어 자체의 대역폭에 훨씬 더 가까워질 수 있습니다.