

세이(Sei)

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.

세이: 트레이딩 특화 레이어 1

거래소와 트레이딩 앱의 최종 목적지

세이 랩스

1 초록

새로운 기술의 발전은 애플리케이션-인프라 주기(application-infrastructure cycle)라 불리는 반복적 패턴을 보여주었다. 이는 애플리케이션이 개선된 인프라에 대한 수요를 견인하고, 그 인프라가 다시 새로운 유형의 애플리케이션이 번성할 수 있도록 한다. 퍼블릭 블록체인의 1세대는 초기 인프라를 제공하여 최근 몇 년간 탈중앙화 애플리케이션의 폭발적 성장을 위한 길을 열었다. 그러나 이러한 애플리케이션들이 초기 도입 단계를 넘어설 때, 확장성, 속도, 신뢰성과 관련된 문제에 직면하는 경우가 많다. 웹2 인프라가 산업이 성숙하면서 주요 사용 사례를 중심으로 특화된 것과 유사하게, 웹3 인프라도 주요 애플리케이션 유형의 필요에 적응하여 웹3의 대형 애플리케이션들이 대중화될 수 있도록 해야 한다. 의심할 여지 없이, 트레이딩은 암호화폐에서 가장 광범위하게 채택된 사용 사례이며, 산업이 확장됨에 따라 기하급수적으로 성장할 것이다. 우리는 트레이딩을 위해 설계된 범용 레이어 1 블록체인인 세이(Sei)를 소개한다. 프로토콜 차원에서 세이는 블록 전파, 트랜잭션 순서 결정, 블록 처리, 병렬화에 대한 새로운 접근 방식을 도입하여 다양한 트레이딩 애플리케이션에 이점을 제공한다.

2 서론

트레이딩이 디파이 애플리케이션에만 국한된다는 것은 흔한 오해이다. 실제로 디지털 자산을 교환해야 하는 필요성은 소셜 애플리케이션부터 대체 불가능 토큰(NFT)과 게임에 이르기까지 암호화폐의 모든 측면에서 찾아볼 수 있다. 암호화폐 산업의 대부분의 애플리케이션은 트래픽의 원천으로 트레이딩 기능에 크게 의존하거나, 사실상 트레이딩 애플리케이션의 형태를 하고 있다. 예를 들어, 많은 웹3 게임은 게임 내 자산 거래를 사용자 경험의 핵심 요소로 통합하고 있다. 암호화폐의 가장 설득력 있는 가치 명제 중 하나는 전 세계 사용자가 언제든지 디지털 자산을 교환할 수 있는 무허가적이면서(permissionless) 신뢰를 수반하지 않는(trustless) 장을 제공하는 데에 있다.

대부분의 사람들은 상당한 수준의 제품-시장 적합성(product-market fit)을 달성한 제품의 기하급수적 성장 잠재력을 과소평가하는 경향이 있으며, 웹3에서 트레이딩은 이미 그 수준에 도달했다. 그러나 강력한 제품-시장 적합성을 가진 모든 제품과 마찬가지로, 다음 단계는 성장을 촉진하는 것이다. 오늘날의 트레이딩 앱은 기존 레이어 1 블록체인의 제약으로 인해 한계를 마주하고 있다. 신뢰성, 확장성, 속도와 같은 문제들이 트레이딩 애플리케이션의 대중화에 필요한 매끄러운 사용자 경험을 제공하는 것을 방해한다.

이러한 도전 과제를 해결하기 위해, 우리는 트레이딩을 위해 설계된 레이어 1 블록체인인 세이를 제안한다. 프로토콜 차원에서 세이는 트윈-터보(Twin-Turbo) 합의와 여러 단계의 병렬화를 사용하여 지연(latency)을 최소화하고 처리량(throughput)을 극대화한다. 또한 세이는 네이티브 주문 매칭 엔진을 통해 애플리케이션이 사용자 경험을 맞춤화할 수 있도록 한다. 애플리케이션은 트레이딩 사용자 경험의 독특한 요구를 충족하기 위해 맞춤 설계된 인프라 위에서 번성할 수 있다. 이전 레이어 1 블록체인의 한계를 해결함으로써, 세이의 견고한 아키텍처는 애플리케이션이 확장되고 대중화에 도달할 수 있는 기반을 제공한다.

3 프로토콜 개선

3.1 트윈-터보 합의

3.1.1 지능적 블록 전파(Intelligent Block Propagation)

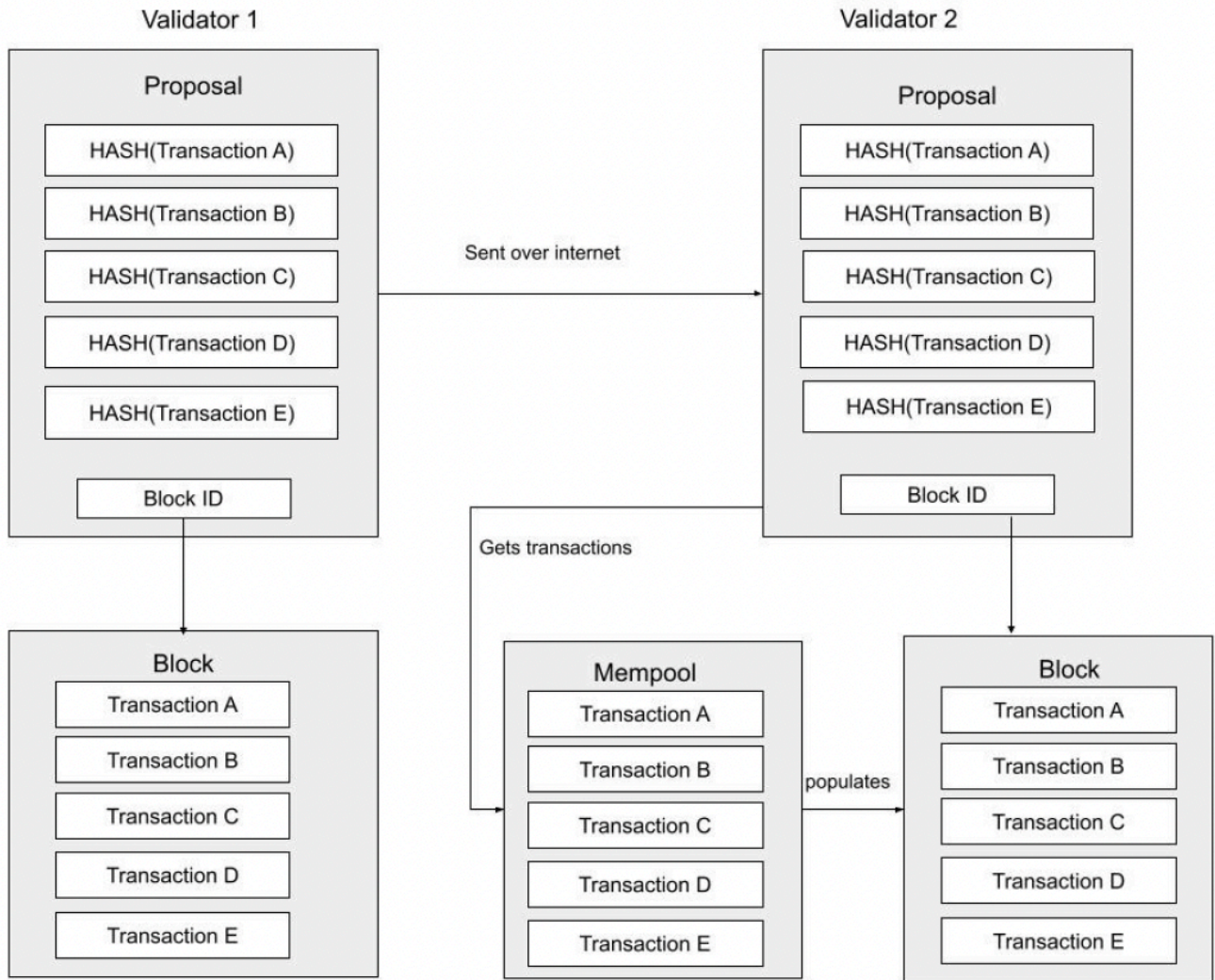


Fig.1. Block proposals with transaction identifiers

풀노드가 사용자로부터 트랜잭션을 수신하면, 이를 네트워크의 다른 노드들에게 전파해야 한다. 풀노드는 트랜잭션을 네트워크 내 다른 노드들에게 무작위로 가십(gossip)한다. 트랜잭션을 검증자가 수신하면, 해당 트랜잭션의 유효성을 확인하고 그 트랜잭션을 해당 검증자의 로컬 메모풀(mempool)에 추가한다.

블록 제안자는 자신의 메모풀의 현재 상태를 살펴보고 커밋할 블록을 제안한다. 대부분(혹은 전부)의 트랜잭션은 앞서 설명한 트랜잭션 전파 방식을 통해 이미 검증자에게 도달했기 때문에, 제안자는 블록 제안에 고유한 트랜잭션 식별자와 전체 블록에 대한 참조를 포함한다. 제안자는 먼저 네트워크의 다른 검증자들에게 블록 제안을 전파한 후, 각 트랜잭션의 전체 내용을 담은 블록 전체를 전송한다. 블록 제안은 하나의 메시지로 전송되고, 블록 전체는 여러 부분으로 나뉘어 네트워크에 가십된다.

검증자가 제안에 포함된 모든 트랜잭션을 로컬 메모풀에 가지고 있다면, 모든 블록 조각이 도착하기를 기다리지 않고 자신의 메모풀에서 전체 블록을 재구성한다. 만약 모든 트랜잭션을 가지고 있지 않다면, 네트워크로부터 모든 블록 조각을 수신할 때까지 기다린 후, 모든 트랜잭션을 포함하여 블록을 구성한다.

이 과정은 검증자가 블록을 수신하기 위해 기다려야 하는 전체 시간을 크게 단축한다. 검증자들이 블록 제안의 일부로 모든 트랜잭션을 확보하면, 텐더민트(Tendermint) BFT 합의 방식을 바탕으로 트랜잭션 순서에 대한 합의를 진행한다. 구체적으로, 블록과 그에 따른 상태 변경이 블록체인에 커밋되기 전에 프리보트(prevote) 단계, 프리커밋(precommit) 단계, 그리고 커밋(commit) 단계가 선행된다.

3.1.2 낙관적 블록 처리(Optimistic Block Processing)

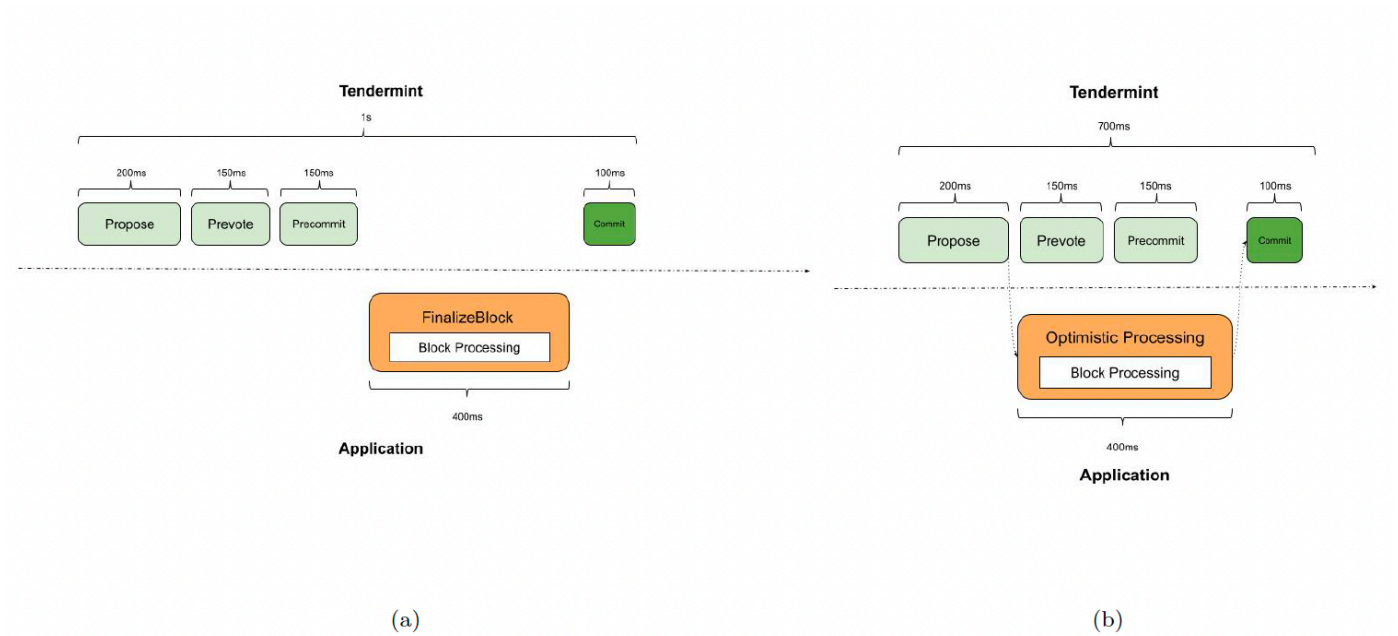


Fig.2. Block processing with example times (a) Block processing after precommit (b) Optimistic block processing

텐더민트 합의의 일부로, 검증자들은 블록 제안을 수신해 블록의 유효성을 검증한 후 프리보트(prevote) 단계로 넘어간다.

트랜잭션 처리를 프리커밋(precommit) 단계 이후까지 기다리는 대신(그림 2a), 검증자들은 임의의 높이(height)에 대해 수신한 첫 번째 블록 제안을 낙관적으로 처리하기 위한 프로세스를 동시에 시작한다(그림 2b). 낙관적 블록 처리는 후보 상태(candidate state)를 캐시에 기록한다.

만약 해당 블록이 네트워크에 의해 승인되면, 캐시의 데이터가 커밋된다. 네트워크가 블록을 거부하면, 캐시의 데이터는 폐기되며 해당 높이의 이후 라운드에서는 낙관적 블록 처리가 사용되지 않는다.

낙관적 블록 처리로 인한 지연(latency)의 이론적 개선 정도는 다음과 같다:

$$\min(T_{prevote} + T_{precommit}, N \cdot T)$$

여기서 $T_{prevote}$ 는 프리보트 지연 시간, $T_{precommit}$ 은 프리커밋 지연 시간, N 은 트랜잭션의 개수, T 는 단일 트랜잭션의 평균 지연 시간이다.

3.2 병렬화(Parallelization)

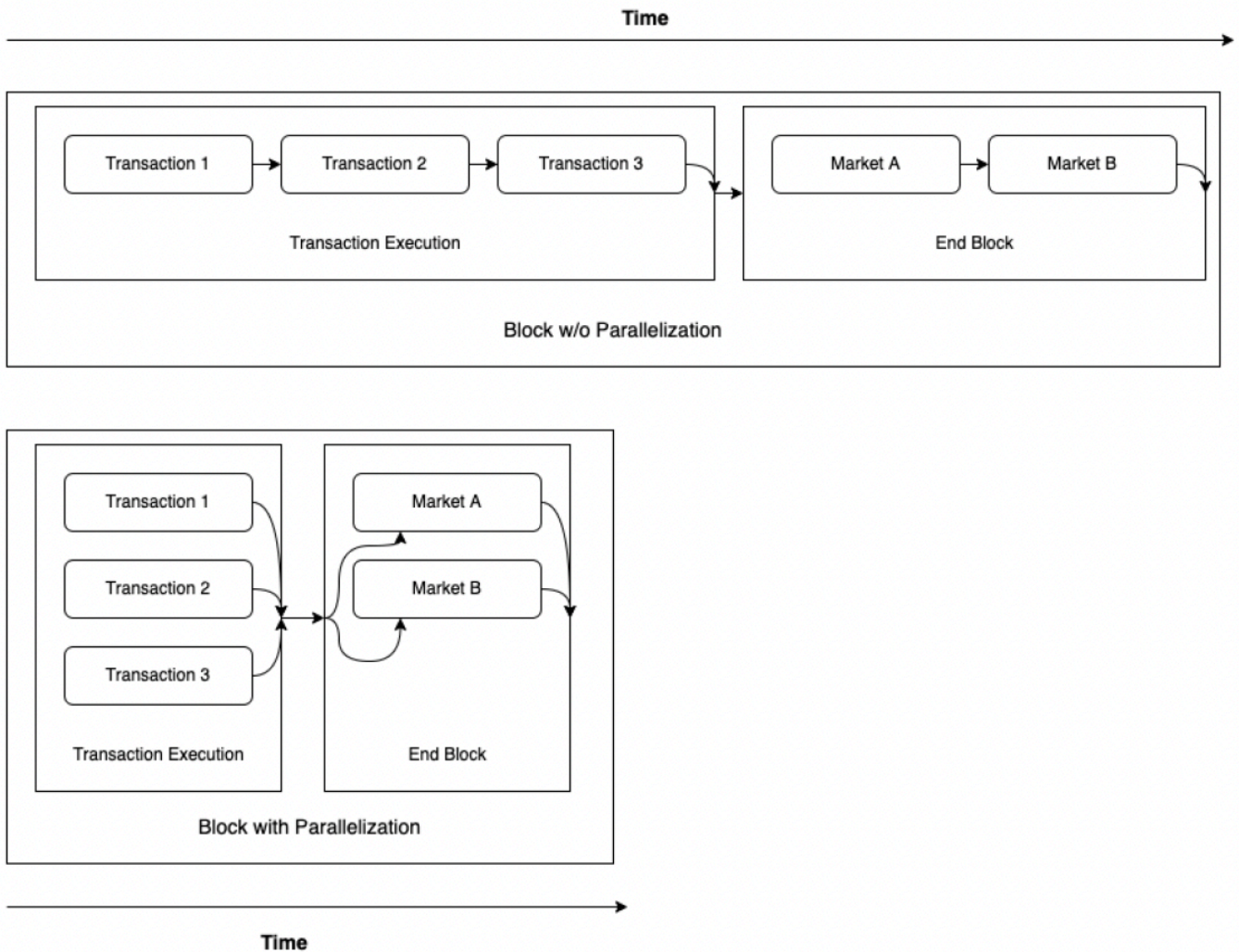


Fig.3. Block processing with and without parallelization

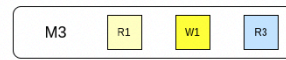
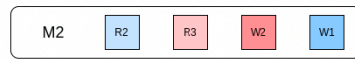
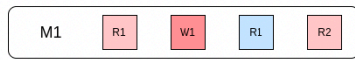
세이는 코스모스(Cosmos) SDK를 애플리케이션 로직의 기반으로 사용한다. 이 로직의 일환으로 검증자가 블록을 수신하고 네트워크 상태를 갱신하기 위해 처리할 때, 먼저 BeginBlock 로직을 실행하고, 이어서 DeliverTx 로직을 실행하며, 마지막으로 EndBlock 로직을 실행한다. 이들 각각은 맞춤 구성이 가능한데, 세이는 트랜잭션을 병렬 처리할 수 있도록 DeliverTx와 EndBlock을 구성하였다(그림 3 참조).

세이는 DeliverTx 단계에서 블록 내 모든 트랜잭션을 먼저 처리한다. 이는 대부분의 트랜잭션 유형(토큰 전송, 거버넌스 제안, 스마트 계약 호출 등)에 대해 상태 변화를 일으킨다. 그러나 주문 매칭 엔진과 관련된 트랜잭션은 DeliverTx 단계에서 기본적인 처리만 거치며, 주요 상태 변화는 EndBlock 로직 동안 적용된다. 이는 빈번한 배치 경매를 지원하기 위함으로, 주문들이 집계되고 균일한 균형 가격(uniform clearing price)이 계산되어 주문이 실행되도록 한다(자세한 내용은 4.1.1절 참조).

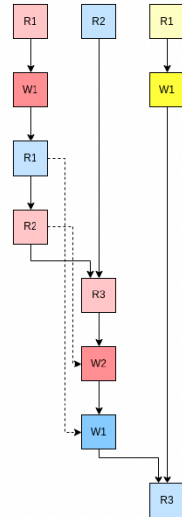
세이는 최적의 성능을 얻기 위해 DeliverTx와 EndBlock 모두에 병렬화 기능을 추가하였다.

3.2.1 DeliverTx 트랜잭션 병렬화

Sequential Processing



Access DAG



Parallel Processing

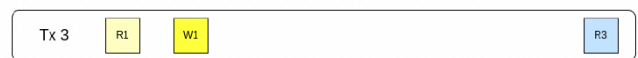


Fig.4. Access DAG for parallel processing

DeliverTx 동안 트랜잭션을 순차적으로 처리하는 대신, 세이는 트랜잭션을 병렬로 처리한다(그림 4 참조). 이를 통해 여러 트랜잭션을 동시에 처리할 수 있으며, 성능 향상으로 이어진다. 세이의 데이터는 키-값(key-value) 형태로 영구 저장된다. 경쟁 조건(race condition)과 비결정성(nondeterminism)을 방지하기 위해, 세이는 여러 병렬 프로세스가 동일한 키를 업데이트하지 않도록 해야 한다.

이는 트랜잭션 메시지 유형을 해당 메시지가 접근해야 하는 키와 매핑(의존성 매핑, dependency mappings)함으로써 달성된다. 서로 다른 키를 업데이트하는 메시지는 병렬로 실행될 수 있지만, 동일한 키를 업데이트하는 메시지는 순차적이고 결정적인 순서(블록 내 트랜잭션 순서에 의해 결정됨)로 실행되어야 한다.

블록의 트랜잭션을 실행하기 전에, 각 트랜잭션의 메시지가 사용하는 다양한 자원 간의 의존성을 식별하기 위해 방향성 비순환 그래프(Directed Acyclic Graph, DAG)를 구성한다.

예를 들어, 특정 X 모듈과 관련된 메시지의 기본 의존성 매핑은 모든 메시지가 동일한 키 ABC를 업데이트하는 경우이다. 따라서 이들 메시지는 순차적으로 실행되어야 한다.

많은 경우, 메시지의 구체적 내용이 더 높은 수준의 병렬성을 제공한다. 예를 들어, 계정 A에서 B로, 계정 C에서 D로의 토큰 전송은 서로 다른 키를 업데이트하므로 병렬로 실행될 수 있다. 그러나 메시지 유형만으로 매핑을 정의하고 메시지 내용을 무시하면, 이러한 두 전송은 순차적으로 실행되게 된다.

이러한 유형의 병렬성에 유연성을 제공하기 위해, 의존성 매핑은 템플릿으로 정의될 수 있으며, 런타임 시 더 세분화된 자원으로 채워진다. 토큰 전송 예시에서 송신자와 수신자 계정이 템플릿에 전달되어 더 세밀한 병렬성을 제공하게 된다.

체인에 의해 정의된 메시지 유형(스테이킹, 오라클 업데이트, 은행 내 전송 등)의 경우, 매핑은 블록체인이 최초로 시작할 때 설정되며, 거버넌스 제안을 통해 업데이트될 수 있다. 여기서 예외 케이스가 존재하는데, 바로 모든 트랜잭션에 영향을 미치는 가스 수수료 징수와 관련된 메시지 유형이다. 이는 DeliverTx 로직 종료 시 플러시(flush)되는 인메모리 데이터 저장소에 데이터를 기록함으로써 처리된다.

세이 위에서 애플리케이션을 구축하는 개발자가 정의한 메시지 유형의 경우, 스마트 컨트랙트는 자체 자원의 존성을 정의해야 한다. 이는 컨트랙트 초기화 시 설정되며, 스마트 컨트랙트 관리자가 업데이트 트랜잭션을 통해 수정할 수 있다. 의존성이 올바르게 작성되면, 스마트 컨트랙트는 병렬화의 혜택을 누리고 더 낮은 가스 수수료를 지불한다. 의존성이 정의되지 않은 경우, 스마트 컨트랙트는 순차적으로 실행되며 다른 트랜잭션의 실행을 막는다. 네트워크의 나머지를 차단하므로, 해당 스마트 컨트랙트에 대한 트랜잭션은 더 높은 가스 수수료를 지불해야 한다. 특정 스마트 컨트랙트의 의존성이 잘못 정의된 경우, 해당 스마트 컨트랙트에 대한 메시지는 실패하고 더 높은 가스 수수료가 청구되지만, 네트워크 전체에는 영향을 주지 않으며 다른 메시지들은 정상적으로 처리된다.

3.2.2 시장 기반 병렬화(Market Based Parallelization)

블록의 끝에서, 모든 매칭 엔진 관련 주문은 네이티브 매칭 엔진에 의해 처리된다. 주문을 순차적으로 처리하는 대신, 세이는 블록 종료 시 독립적인 매칭 엔진 관련 주문을 병렬로 처리한다. 두 주문이 동일한 블록에서 동일한 마켓에 영향을 주지 않는다면, 이들은 독립적이다. 기본적으로 체인은 서로 다른 마켓에 걸친 모든 주문을 독립적인 것으로 간주하며, 개발자가 명시적으로 서로 다른 마켓 간의 의존성을 정의하지 않는 한 그렇다. 이러한 의존성은 스마트 컨트랙트가 배포될 때 정의된다. 만약 이러한 의존성이 잘못 정의되면, 해당 스마트 컨트랙트에 대한 트랜잭션은 제대로 처리되지 않는다.

3.3 네이티브 가격 오라클(Native Price Oracles)

세이는 네이티브 가격 오라클을 통해 자산 환율 가격 산정을 지원한다. 검증자들은 자산에 대해 가장 신뢰할 수 있고 정확한 가격을 보장하기 위해 오라클로서 반드시 참여해야 한다. 오라클 가격의 최신성을 유지하기 위해, 투표 기간(voting window)은 최소 1 블록 길이로 설정될 수 있으며, 이를 통해 신속한 가격 업데이트와 최신 자산 가격 산정이 가능하다.

투표 기간 동안의 투표 단계에서, 검증자는 해당 기간에 대한 자신이 제안하는 환율을 제공한다. 투표 기간이 끝나면, 모든 환율 투표가 집계되고 가중 중앙값(weighted median, 검증자 투표 권한에 의해 가중됨)이 계산되어 각 자산의 실제 환율을 결정한다.

불참이나 잘못된 데이터 제출에는 패널티가 부과된다. 누락 횟수(miss count)는 검증자가 데이터를 제공하지 않았거나, 제공한 데이터가 가중 중앙값에서 지나치게 벗어난 투표 기간의 횟수를 추적한다. 특정 수의 투표 기간 내에서 검증자의 누락 횟수가 너무 높으면, 장기간의 비정상 행위에 대한 처벌로 삭감(slash)이 적용된다.

4 네이티브 주문 매칭 엔진(Native Order Matching Engine)

세이는 범용 블록체인 기능(즉, 사용자가 자산을 전송하고 스마트 계약을 배포할 수 있는 기능)을 제공한다. 여기에 더해, 세이는 주문 배치 및 매칭 엔진(이하 본 페이퍼에서는 “매칭 엔진(matching engine)“이라 함)을 구축하였으며, 이는 세이 위에 구축되는 어떤 거래소든 활용할 수 있다.

4.1 세이를 통한 탈중앙화 거래소 배포(Deploy decentralized exchanges with Sei)

매칭 엔진은 세이 위에서 구축되는 탈중앙화 거래소가 자체 오더북을 배포할 수 있도록 한다. 매칭 엔진은 체인 수준에서 해당 오더북을 유지하며, 시장을 생성하고 사용자가 거래할 수 있도록 기능을 제공한다.

4.1.1 주문의 생애주기(Lifecycle Of An Order)

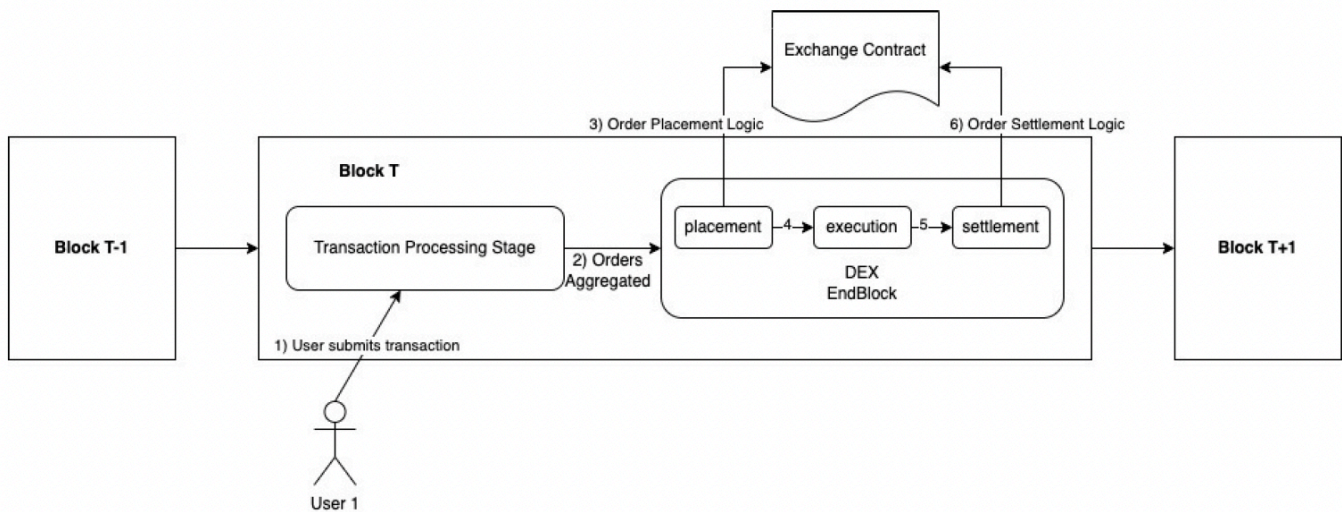


Fig.5. Lifecycle of a transaction

모든 매칭 엔진 관련 트랜잭션은 블록의 범위 내에서 원자적으로(atomically) 실행된다. 매칭 엔진과 관련된 트랜잭션은 그림 5에서 보듯이 dex 모듈로 전송된다. 하나의 트랜잭션은 하나 이상의 주문으로 구성될 수 있다(자세한 내용은 4.4.1절 참조). 제출 시, 트랜잭션 핸들러(transaction handler)는 각 트랜잭션에 포함된 주문을 dex 모듈의 내부 MemState에 추가하여 트랜잭션을 처리한다(그림 5 action 1).

각 블록을 처리하는 동안, dex 모듈은 EndBlocker 혹은 사용하여 MemState에 기록된 주문을 일괄 처리한다(그림 5 action 2). 구체적으로, dex 모듈의 EndBlocker 혹은 호출되면, 트랜잭션 전반의 주문이 시장별로 집계된다(예: BTC 무기한 선물에 대한 모든 주문). 그리고 해당 특정 시장에 대해 하나의 스마트 계약 호출로 결합된다(체인 수준 번들링에 대한 자세한 내용은 4.4.2절 참조).

그 후, 체인은 해당 시장과 연관된 스마트 계약(예: 무기한 선물 거래소 스마트 계약)를 호출한다. 이 컨트랙트는 매칭 엔진과 상호작용하는 방법에 대한 모든 로직을 정의하고 있다(그림 5 action 3). 스마트 컨트랙트는 자체 커스텀 로직을 구현하고, 이후 매칭 엔진을 호출한다(그림 5 action 4).

매칭 엔진은 먼저 모든 주문 취소를 처리한다. 이는 관련 지정가(limit) 주문을 오더북 저장소에서 제거한다. 그 다음 모든 지정가 주문이 오더북에 추가된다. 이를 통해 주문이 최대한의 유동성으로 체결되도록 보장한다.

그 후, 매칭 엔진은 시장가(market) 주문을 처리한다. 균일 균형 가격이 계산되며(자세한 내용은 4.3절 참조), 모든 시장가 주문은 해당 가격으로 체결된다. 모든 시장가 주문을 체결할 유동성이 부족한 경우, 더 큰 슬리피지(slippage)를 감수하는 주문이 우선적으로 체결된다.

그 다음, 매칭 지정가 주문이 처리된다. 만약 체결 가능한 지정가 주문이 있다면, 최적의 가격에서 체결된다. 예를 들어, 오더북 저장소에 P_1 가격에 수량 Q_1 , P_2 가격에 수량 Q_2 의 매도 주문이 있다고 가정하자. 이때 $P_3 > P_2$, 수량 $1.5 \cdot Q_2$ 인 매수 지정가 주문이 존재한다면, 매수 지정가 주문은 P_1 가격에서 Q_2 를 매수하고, P_2 가격에서 $0.5 \cdot Q_2$ 를 매수함으로써 체결된다.

마지막으로, 체결되지 않은 모든 시장가 주문은 만료된다. 매칭 엔진 로직이 완료되면, 매칭 엔진은 자산 정산을 처리하기 위해 관련 스마트 컨트랙트를 호출한다(그림 5 action 6).

4.1.2 훅 지원(Hook Support)

세이는 컨트랙트가 네트워크에 “훅(hook)”을 등록할 수 있도록 허용한다. 등록된 훅은 매 블록마다 호출되며, 플래시 론 상황과 같은 연산이 연관된 거래 정산과 같은 블록 내에서 수행될 수 있게 한다. 구체적으로, 컨트랙트는 두 개의 훅을 정의할 수 있다. 첫 번째 훅은 블록 시작 시 호출되어, 해당 블록에서 발생할 수 있는 잠재적 거래에 대비할 수 있도록 한다. 두 번째 훅은 주문 매칭과 정산이 끝난 블록의 마지막 시점에 호출되어, 필요하다면 컨트랙트가 사후 거래(post-trade) 로직을 수행할 수 있도록 한다.

4.2 자산 불가지론적 매칭 엔진(Asset Agnostic Matching Engine)

매칭 엔진은 반드시 토큰을 거래 대상으로 요구하지 않는다. 대신 탈중앙화 거래소가 자산을 어떻게 표현할지 결정할 수 있도록 유연한 인터페이스를 제공한다. 예를 들어, 탈중앙화 거래소가 포지션을 토큰화하지 않고 스마트 컨트랙트 상태 내 리스트 형태로 추적할 수 있다.

4.3 빈번한 배치 경매(Frequent Batch Auctioning)

주문을 순차적으로 실행하면 검증자들이 자신에게 이익이 되는 방식으로 트랜잭션을 배열할 수 있는 유인을 제공한다. 예를 들어, 검증자가 시장가 주문의 유입을 봤을 때, 해당 트랜잭션이 처리되기 전에 스스로 그 자산을 매수하는 시장가 주문과 더 높은 가격에 매도하는 지정가 주문을 포함시킬 수 있다. 이러한 형태의 MEV(Maximal Extractable Value)를 방지하기 위해, 세이의 매칭 엔진은 모든 시장가 주문을 집계한 뒤 동일한 균일 균형 가격으로 실행한다.

예를 들어, 오더북에 P_1 , P_2 가격의 두 개의 매도 주문이 있고, B_1 , B_2 라는 두 개의 매수 주문이 들어오는 경우, B_1 은 P_1 에서, B_2 는 P_2 에서 체결되는 대신, 두 매수 주문 모두 균일 균형 가격 $\frac{P_1 + P_2}{2}$ 에서 체결된다. 이 방식은 기존 지정가 주문이 의도된 가격(P_1 과 P_2)에서 체결되도록 하면서, 신규 시장가 주문이 보다 공정한 가격으로 체결되도록 한다.

4.4 트랜잭션 주문 번들링(Transaction Order Bundling)

세이는 사용자 경험과 성능을 개선하기 위해 여러 계층의 주문 번들링을 제공하며, 이는 아래 절들에 설명되어 있다.

4.4.1 클라이언트 주문 번들링(Client Order Bundling)

세이 트랜잭션은 여러 시장을 대상으로 하는 주문들(심지어 스마트 컨트랙트를 넘나드는 주문들, 예: BTC/USDC 현물 페어와 BTC 무기한 선물 거래소에 동시에 주문)을 포함할 수 있다. 블록이 처리되는 동안 세이는 모든 주문을 해당 스마트 컨트랙트로 올바르게 라우팅한다. 이는 마켓 메이커들이 포지션 업데이트와 관련된 가스 비용을 절감하는 데 도움이 된다.

4.4.2 체인 수준 주문 번들링(Chain Level Order Bundling)

매칭 엔진과 관련된 각 트랜잭션은 가상머신(Virtual Machine, VM)을 인스턴스화해야 한다. 여러 번의 가상 머신 인스턴스화를 수행하는 대신, 세이는 모든 트랜잭션에 걸친 주문을 시장 단위로 번들링하여 하나의 가상머신 인스턴스화만 수행한다. 이는 주문당 약 1ms의 지연(latency)을 줄여주며, 특히 높은 처리량 구간에서 상당한 성능 개선을 제공한다.

4.5 트레이딩 수수료(Trading Fees)

매칭 엔진은 출시 시 체인 수준에서 어떠한 트레이딩 수수료도 부과하지 않는다. 거버넌스는 향후 트레이딩 수수료 적용 여부를 선택할 수 있다. 세이 위에 구축되는 탈중앙화 거래소는 자신들의 사용자 경험에 따라 별도의 트레이딩 수수료를 추가할 수 있다. 이는 스마트 컨트랙트 수준에서 정의되며, 개발자가 손쉽게 구성할 수 있다.