

딥북(DeepBook)

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.

DeepBookV3

DeepBookV3는 Sui 위에 구축된 차세대 분산형 중앙 한도 주문장부(CLOB)입니다. DeepBookV3는 Sui의 병렬 실행 및 낮은 거래 수수료를 활용하여 고성능, 저지연 온체인 거래소를 구현합니다. DeepBookV3는 Sui를 기반으로 구축된 차세대 분산형 중앙 한도 주문장부(CLOB)입니다. DeepBookV3는 Sui의 병렬 실행 및 낮은 거래 수수료를 활용하여 고성능, 저지연 온체인 거래소를 구현합니다.

최신 버전은 플래시 대출, 거버넌스, 개선된 계정 추상화, 기존 매칭 엔진 강화 등 새로운 기능을 제공합니다. 또한 [DEEP 토큰](#)을 통해 자체 토큰 경제를 도입합니다. 스테이킹을 통해 추가 혜택을 얻을 수 있습니다. DeepBookV3는 Sui를 기반으로 구축된 차세대 분산형 중앙 한도 주문장부(CLOB)입니다. DeepBookV3는 Sui의 병렬 실행 및 낮은 거래 수수료를 활용하여 고성능, 저지연 온체인 거래소를 구현합니다.

DeepBookV3는 토큰 거래를 위한 최종 사용자 인터페이스를 포함하지 않습니다. 대신, 분산형 거래소, 지갑 또는 기타 앱에서의 토큰 거래를 지원할 수 있는 내장형 거래 기능을 제공합니다. 사용 가능한 SDK는 체인과의 상호작용 및 프로그래밍 가능한 트랜잭션 블록 구축의 복잡성을 상당 부분 추상화하여, 적극적인 시장 조성에 대한 진입 장벽을 낮춥니다.

DeepBookV3 토큰노믹스

DEEP 토큰은 거래소에서 거래 수수료를 지불하는 데 사용됩니다. 사용자는 DEEP 토큰이나 입력 토큰으로 거래 수수료를 지불할 수 있지만, DEEP을 보유, 사용 및 스테이킹하는 것이 Sui 네트워크에서 DeepBookV3를 활발히 이용하는 트레이더에게 가장 큰 혜택을 지속적으로 제공합니다.

Sui 네트워크에서 DeepBookV3를 활발히 거래하는 사용자에게 가장 큰 혜택을 제공합니다.

예를 들어, 거버넌스는 DEEP 토큰으로 지불하는 수수료를 결정하며, 이는 입력 토큰 사용 수수료보다 20% 낮습니다.

DEEP을 스테이킹하는 사용자는 테이커 및 메이커 인센티브를 누릴 수 있습니다. 테이커 인센티브는 거래 수수료를 절반으로 줄여 안정적인 페어의 경우 최저 0.25 베이스 포인트(bps), 변동성이 큰 페어의 경우 2.5bps까지 낮출 수 있습니다. 메이커 인센티브는 생성된 메이커 거래량에 따라 획득하는 리베이트입니다.

유동성 지원

다른 시장의 주문장과 유사하게, DeepBookV3의 CLOB 아키텍처는 시장가 주문과 지정가 주문을 입력할 수 있게 합니다. SUI 토큰을 매도(매도 호가)할 때는 가격을 설정(지정가 주문)하거나 시장가에 따라 매도할 수 있습니다. SUI를 매수(매수 호가)하려는 경우 현재 시장 가격을 지불하거나 지정가를 설정할 수 있습니다. 한도 주문은 CLOB이 매수자와 매도자 간의 매칭을 찾을 때만 체결됩니다.

1,000 SUI에 대한 지정가 주문을 넣었을 때, 현재 해당 수량의 토큰을 판매하는 단일 판매자가 없는 경우 DeepBookV3는 자동으로 현재 매도 호가를 모아 귀하의 매수 주문 수량을 충족시킵니다.

투명성과 프라이버시

CLOB(대용량 객체 저장소)인 DeepBookV3는 디지털 원장처럼 작동하여 매수 호가와 매도 호가를 시간 순으로 기록하고 양측 간의 매칭을 자동으로 찾아냅니다. 거래 시 가격 등 사용자 설정을 고려합니다.

디지털 원장은 공개되어 있어 누구나 거래 내역과 가격을 확인할 수 있어 공정성을 명확히 입증합니다. 이러한 투명성을 활용해 거래 활동을 모니터링하는 지표와 대시보드를 생성할 수 있습니다.

문서

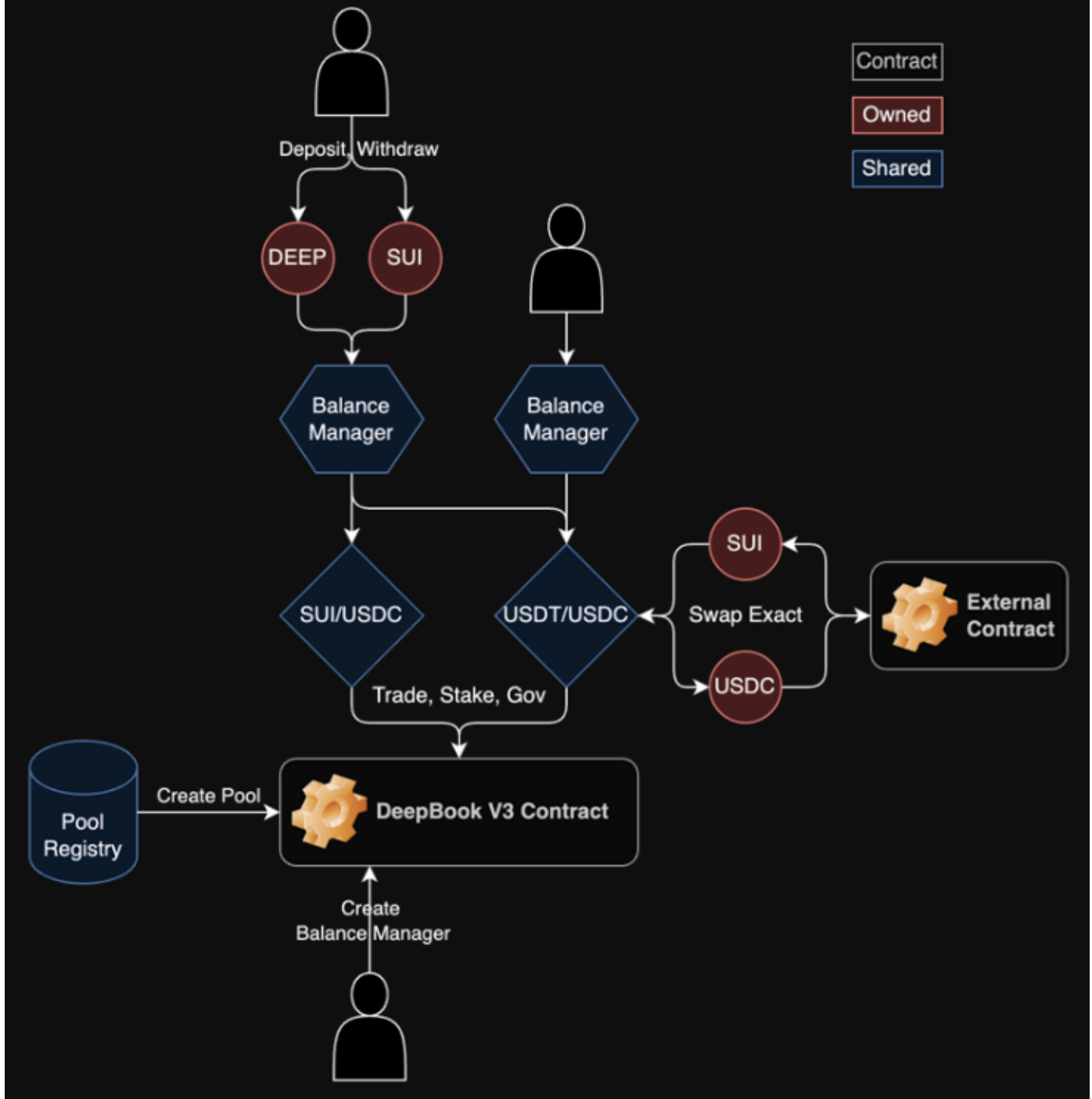
본 문서는 DeepBookV3의 설계, 공개 엔드포인트를 설명하고 통합 가이드라인을 제공합니다. SDK는 체인 상호작용 및 프로그래밍 가능한 트랜잭션 블록 구축의 복잡성을 상당 부분 추상화하여 능동적 시장 조성의 진입 장벽을 낮춥니다.

설계

대략적으로 DeepBookV3 설계는 다음 흐름을 따르며, 이는 세 가지 공유 객체를 중심으로 구성됩니다:

- 풀(Pool): 하나의 시장을 나타내는 공유 객체로, 해당 시장의 주문장, 사용자, 스테이크 등을 관리합니다. 자세한 내용은 [풀 공유 객체](#) 섹션을 참조하세요.
- 풀 레지스트리(PoolRegistry): 풀 생성 시에만 사용되며, 중복 풀 생성을 방지하고 패키지 버전 관리를 수행합니다.
- 밸런스 매니저(BalanceManager): 주문 시 사용자의 자금을 조달하는 데 사용됩니다. 모든 풀에서 단일 밸런스 매니저를 공유할 수 있습니다. 자세한 내용은 [밸런스 매니저](#)를 참조하세요.

DeepBook



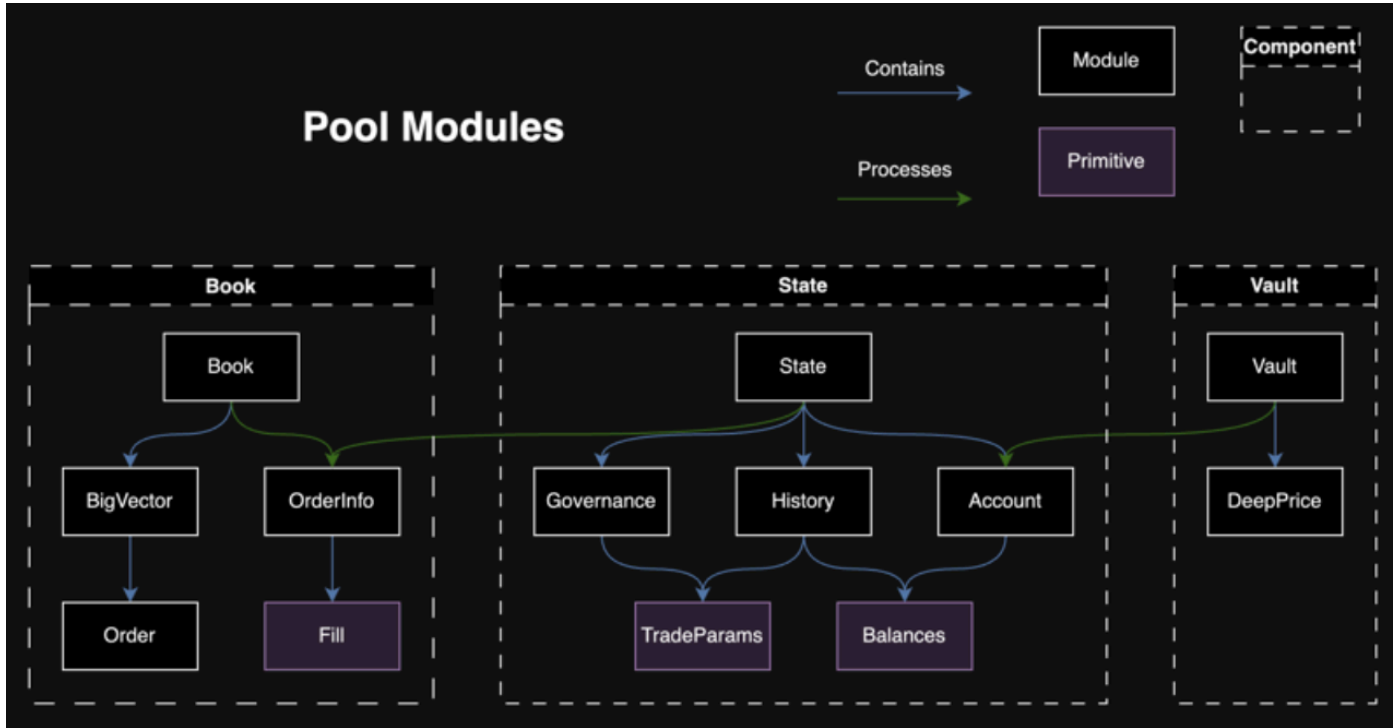
풀 공유 객체

모든 공개 함수는 풀 공유 객체를 변경 가능 또는 변경 불가능 참조로 인수로 받습니다. 풀은 세 가지 별개의 구성 요소로 이루어집니다:

- 주문장부

- 상태
- 볼트

로직은 구성 요소 간에 격리되며 각 구성 요소는 이전 구성 요소를 기반으로 구축됩니다. 주문장부(Book), 스테이트(State), 볼트(Vault) 관계를 유지함으로써 DeepBookV3는 데이터 가용성 보장을 제공하고, 코드 가독성을 향상시키며, 프로토콜 유지 관리 및 업그레이드를 용이하게 합니다.



주문장부

이 구성 요소는 메인 주문장부 모듈과 함께 Fill, OrderInfo, Order 모듈로 구성됩니다. 주문장부 구조체는 매수 호가와 매도 호가를 위한 두 개의 BigVector 객체와 일부 메타데이터를 유지합니다. 주문 저장, 매칭, 수정 및 제거를 담당합니다.

주문을 제출할 때 먼저 OrderInfo가 생성됩니다. 해당되는 경우 기존 메이커 주문과 먼저 매칭되며, 이 과정에서 Fills가 누적됩니다. 남은 수량은 Order 객체를 생성하는 데 사용되어 북에 주입됩니다. 북 처리 종료 시점에는 OrderInfo 객체가 모든 관련 사용자와 전체 상태를 업데이트할 수 있는 충분한 정보를 보유하게 됩니다.

상태

상태는 거버넌스, 히스토리, 계정을 저장합니다. 모든 요청을 처리하며, 저장된 구조체 중 적어도 하나를 업데이트합니다.

거버넌스

거버넌스 모듈은 풀의 거래 매개변수와 관련된 데이터를 저장합니다. 이 매개변수는 테이커 수수료, 메이커 수수료, 그리고 필요한 스테이크입니다. 필요한 스테이크는 사용자가 테이커 및 메이커 인센티브를 받을 자격을 얻기 위해 이 특정 풀에 스테이킹해야 하는 DEEP 토큰의 양을 나타냅니다.

매 에포크마다 스테이크가 0이 아닌 사용자는 이러한 매개변수를 변경하기 위한 제안을 제출할 수 있습니다. 제안된 수수료는 상한선이 설정됩니다.

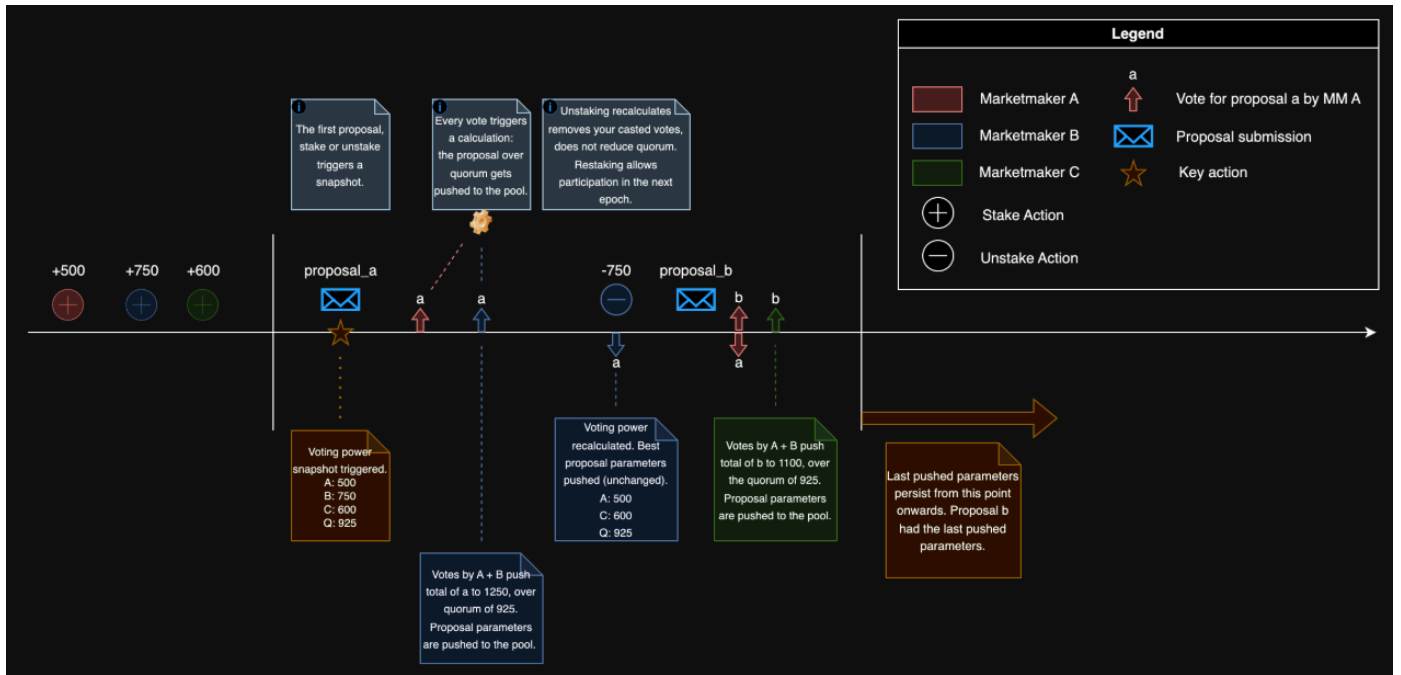
min_value (bps)	max_value (bps)	Pool type	Taker or maker
1	10	Volatile	Taker
0	5	Volatile	Maker
0.1	1	Stable	Taker
0	0.5	Stable	Maker
0	0	Whitelisted	Taker and maker

거버넌스 투표 권한

사용자는 진행 중인 제안에 대해서도 투표할 수 있습니다. 제안이 정족수를 초과하면 새로운 거래 매개변수가 다음 에포크부터 적용되도록 대기열에 등록됩니다. 제안과 투표는 매 에포크마다 초기화됩니다. 사용자는 스테이킹 후 다음 에포크부터 제안 제출 및 투표를 시작할 수 있습니다. 정족수는 총 투표권의 절반에 해당합니다. 사용자의 투표권은 다음 공식으로 계산되며, 여기서 V 는 투표권, S 는 스테이킹 금액, V_c 는 투표권 커트오프입니다. V_c 는 현재 100,000 DEEP로 설정되어 있습니다.

$$V = \min(S, V_c) + \max(\sqrt{S} - \sqrt{V_c}, 0)$$

다음 다이어그램은 거버넌스 라이프사이클을 시각화하는 데 도움이 됩니다.



히스토리

히스토리 모듈은 현재 에포크 및 이전 에포크의 집계된 거래량, 거래 매개변수, 징수된 수수료, 소각 예정 수수료를 저장합니다. 주문 처리 중 체결 내역은 총 거래량 계산 및 업데이트에 활용됩니다. 또한 거래 메이커가 충분한 스테이킹을 보유한 경우 총 스테이킹 거래량도 업데이트됩니다.

각 에포크의 첫 번째 작업은 업데이트를 트리거하여 현재 에포크 데이터를 역사적 데이터로 이동시키고, 현재 에포크 데이터를 재설정합니다.

사용자 리베이트 계산은 이 모듈에서 수행됩니다. 각 에포크 동안 메이커는 스테이킹한 DEEP이 요구 스테이크를 초과하고 메이커 거래량에 기여한 경우 리베이트를 받을 자격이 있습니다. 메이커 수수료 계산에는 [백서: DeepBook 토큰 문서에서 인용한 다음 공식이 사용됩니다.](#) 메이커 인센티브에 대한 자세한 내용은 백서 2.2절에서 확인할 수 있습니다.

인센티브 계산

인센티브 계산은 에포크 종료 후 수행되며, 사전에 필요한 수량의 DEEP 토큰을 스테이킹한 메이커에게만 지급됩니다. 특정 메이커 i 에 대한 인센티브는 방정식 (3)에 따라 계산됩니다. 방정식 (3)에는 몇 가지 새로운 변수가 도입됩니다. 첫째, M 은 충분한 수의 DEEP 토큰을 스테이킹한 메이커 집합을, $\overline{M}$ 은 이 조건을 충족하지 못한 메이커 집합을 나타냅니다. 둘째, F 는 특정 에포크 동안 메이커의 거래량이 발생한 총 수수료(테이커와 메이커 모두로부터 징수된 금액)를 의미합니다. 셋째, L 은 메이커가 제공한 총 유동성을 나타내며, 특히 호가 유동성이 아닌 실제 거래된 유동성을 의미합니다. 마지막으로 핵심 점 p 는 '페이지아웃' 지점으로, 다른 메이커들이 제공한 총 유동성이 이 지점을 넘어서면 해당 에포크에서 해당 메이커의 인센티브가 제로가 되는 지점이다. 이 점 p 는 풀과 에포크 내 모든 메이커에게 동일하게 적용된다.

$$\text{Incentives for Maker } i = \max \left[F_i \left(1 + \frac{\sum_{j \in \bar{M}} F_j}{\sum_{j \in M} F_j} \right) \left(1 - \frac{\sum_{j \in M \cup \bar{M}} L_j - L_i}{p} \right), 0 \right] \quad (3)$$

본질적으로, 에포크 기간 동안의 총 거래량이 지난 28일간의 중간 거래량을 초과할 경우 리베이트가 지급되지 않습니다. 중간 거래량 대비 거래량이 낮을수록 더 많은 리베이트를 받을 수 있습니다. 에포크당 최대 리베이트 금액은 해당 에포크 동안 수집된 DEEP 총량과 동일합니다. 잔여 DEEP은 소각됩니다.

계정

계정은 단일 사용자와 해당 관련 데이터를 나타냅니다. 거래량, 스테이킹, 투표 제안, 미청구 리베이트, 이체될 잔액 등 모든 관련 정보를 포함합니다. BalanceManager와 계정 간에는 일대일 관계가 성립합니다.

매 에포크마다 사용자가 수행하는 첫 번째 작업은 계정을 업데이트하여 이전 에포크의 잠재적 리베이트를 계산하고 현재 에포크의 거래량을 재설정합니다. 이전 에포크의 신규 스테이킹은 활성화됩니다.

각 계정에는 정산된 잔액과 미지급 잔액이 존재합니다. 정산된 잔액은 풀이 사용자에게 지급해야 할 금액이며, 미지급 잔액은 사용자가 풀에 지급해야 할 금액입니다. 예를 들어 주문을 할 때 사용자의 미지급 잔액이 증가하며, 이는 해당 주문 실행을 위해 사용자가 지불해야 할 자금을 나타냅니다. 이후 메이커 주문이 다른 사용자에게 의해 체결되면 메이커의 정산된 잔액이 증가하며, 이는 메이커가 지급받을 금액을 나타냅니다.

볼트

사용자가 DeepBookV3에서 수행하는 모든 거래는 정산된 잔액과 미결제 잔액을 재설정합니다. 이후 볼트는 사용자를 위해 이 잔액을 처리하여 자금의 차감 또는 추가를 BalanceManager에 반영합니다.

볼트는 또한 DeepPrice 구조체를 저장합니다. 이 객체는 풀의 기준 자산 또는 호가 자산과 DEEP 간의 환율을 나타내는 최대 100개의 데이터 포인트를 보유합니다. 이 데이터 포인트는 화이트리스트에 등록된 DEEP/USDC 또는 DEEP/SUI 풀에서 가져옵니다. 이 환율은 거래 수수료 지불에 필요한 DEEP 토큰 수량을 결정하는 데 사용됩니다.

BigVector

BigVector는 임의 크기의 벡터형 데이터 구조로, 온체인 B+ 트리를 사용하여 구현됩니다. 이를 통해 거의 상수 시간(log base max_fan_out)의 임의 접근, 삽입 및 제거를 지원합니다.

리프 노드(슬라이스)에 대한 접근을 노출함으로써 반복 처리가 지원됩니다. 초기 슬라이스 찾기는 거의 상수 시간에 수행될 수 있으며, 이후 이전 또는 다음 슬라이스 찾기도 상수 시간에 수행될 수 있습니다.

B+ 트리의 노드는 BigVector에 연결된 개별 동적 필드로 저장됩니다.

지정가 주문 흐름 배치

주문 배치 작업의 라이프사이클을 보여주는 다음 다이어그램은 주문장, 상태, 보관소 흐름을 시각화하는 데 도움이 됩니다.

풀

풀 모듈에서 `place_order_int` 함수는 사용자 입력 매개변수와 함께 호출됩니다. 이 함수 내에서 다음 네 가지 작업이 순차적으로 수행됩니다:

1. `OrderInfo` 객체가 생성됩니다.
2. 북 모듈의 `create_order` 함수가 호출됩니다.
3. 스테이트 모듈의 `process_create` 함수가 호출됩니다.
4. 볼트 모듈의 `settle_balance_manager` 함수가 호출됩니다.

주문장부

주문장부 내 주문 생성은 세 가지 주요 작업을 포함합니다:

- 입력값 검증.
- 기존 주문과의 매칭.
- 잔여 수량을 지정가 주문으로 주문장에 투입.

입력값 검증은 수량, 가격, 타임스탬프, 주문 유형이 예상 범위에 있는지 확인합니다.

주문정보(`OrderInfo`)를 주문장과 매칭하기 위해 주문장 반대편의 주문 목록을 반복 처리합니다. 가격이 겹치고 기존 메이커 주문이 만료되지 않은 경우, `DeepBookV3`는 양을 매칭하여 체결(`Fill`)을 생성합니다. `DeepBookV3`는 해당 체결을 `OrderInfo` 체결 내역에 추가하여 상태 관리 시 활용합니다. `DeepBookV3`는 매칭 시마다 기존 메이커 주문 수량과 상태를 업데이트하며, 완전히 체결되거나 만료된 주문은 주문장부에서 제거합니다.

마지막으로, `OrderInfo` 객체에 잔여 수량이 남아 있을 경우 `DeepBookV3`는 이를 간결한 `Order` 객체로 변환하여 주문장부에 주입합니다. `Order`는 매칭에 필요한 최소한의 데이터를 포함하는 반면, `OrderInfo`는 일반 처리를 위한 최대량의 데이터를 포함합니다.

방향이나 주문 유형에 관계없이, 모든 `DeepBookV3` 매칭은 단일 함수 내에서 처리됩니다.

상태

상태 내 `process_create` 함수는 풀 상태 내에서 주문 생성 이벤트 처리를 담당합니다: 주문에 대한 거래 금액 및 수수료를 계산하고, 이에 따라 계정 거래량을 업데이트합니다.

먼저, 함수는 `OrderInfo` 객체의 체결 목록을 처리하여 추적 중인 거래량을 업데이트하고 관련 메이커의 자금을 정산합니다. 다음으로, 함수는 계정의 총 거래량과 활성 스테이킹을 조회합니다. 테이커 수수료는 사용자의 계정 스테이킹과 DEEP 토큰 거래량을 기반으로 계산되며, 메이커 수수료는 거버넌스 거래 매개변수에서 가져옵니다. 할인된 테이커 수수료를 받으려면 계정이 풀의 최소 스테이크를 초과해야 하며, DEEP 토큰 거래량도 동일한 기준을 넘어야 합니다. `OrderInfo` 객체에 잔여 수량이 남아 있다면, 이는 계정의 주문 목록에 주문(`Order`)으로 추가되며 이미 북(`Book`)에 생성된 상태입니다.

마지막으로, 해당 함수는 테이커 수수료와 메이커 수수료를 고려하여 부분 테이커 체결량과 메이커 주문량을 계산합니다(해당하는 경우). 이를 계정의 이전 정산 및 미지급 잔액에 추가합니다. 주문에서 징수된 총 수수료로 거래 내역이 업데이트되며, (기본, 호가, DEEP) 형식의 정산 잔액과 미지급 잔액으로 구성된 두 개의 튜플이 풀에 반환됩니다. 이를 통해 볼트에서 올바른 자산이 이전되도록 보장합니다.

볼트

볼트의 `settle_balance_manager` 함수는 `BalanceManager`에 대한 정산 완료 및 미지급 금액의 이체를 관리하는 역할을 담당합니다.

먼저, 이 함수는 트레이더가 `BalanceManager` 사용 권한이 있는지 검증합니다.

그런 다음 각 자산 유형별로 출금 잔액(`balances_out`)과 입금 잔액(`balances_in`)을 비교합니다. 출금 잔액 총합이 입금 잔액 총합을 초과할 경우, 함수는 차액을 금고 잔액에서 분할하여 `BalanceManager`에 입금합니다. 반대로 입금 잔액 총합이 출금 잔액 총합을 초과할 경우, 함수는 차액을 `BalanceManager`에서 인출하여 금고 잔액에 합류시킵니다.

이 프로세스는 기본 자산(`base`), 상대 자산(`quote`), DEEP 자산의 잔액에 대해 반복되어 모든 자산 잔액이 금고와 `BalanceManager` 간에 정확히 반영되고 정산되도록 보장합니다.