이번 주 block.one의 블록체인 c++ 개발팀이 4명의 신입 멤버들을 모집했고 몇 명을 더 인터뷰 중에 있습니다. 신입들은 이미 코드에 대한 기여를 하고 있습니다.

등록된 메시지 스키마 삭제

WebAssembly(WASM)으로 변경하게 됨으로써 우리는 Wren을 사용한다는 가정을 토대로 결정한 초기 디자인의 일부를 재평가해야 했습니다. WASM는 독립적 언어이므로, 모든 애플리케이션은 애플리케이션에 전달된 메시지를 분석하기 위해 서로 다른 애플리케이션을 제공해야 할 것입니다. 이것은 ABI(애플리케이션 바이너리 인터페이스)가 기술적인 측면에서 보면 우리가 이전에 내장시킨 등록된 메시지 유형 및 형식 대신에 WASM 파싱에 의해 정의된다는 것을 의미합니다.
블록체인이 메시지 스키마를 강제로 적용할 수 없고 모든 컨트랙트는 이진 데이터가 들어오면 독립적으로 구문 분석하고 검증해야 하기 때문에, 저희는 메시지 스키마로 정의된 블록체인을 제거하기로 결정했습니다. 이를 통해 애플리케이션 개발자에게 더 많은 권한을 부여할 수 있습니다.

계정 이름 구조 변경

이번 주에 코드의 큰 변화 중 하나는 256비트 문자열에서 60비트 문자열(base32 인코딩을 통해 사람이 읽을 수 있는 이름으로 변환하는)로 계정의 이름을 변경하는 것이었습니다. 이렇게 하면 계정 이름이 12개의 소문자와 숫자로 제한됩니다. 평균 트위터 계정의 길이는 12자밖에 되지 않습니다.

이러한 변경 사항은 주로 대역 폭, 메모리 및 성능 고려 사항으로 인해 발생했습니다. 애플리케이션 논리적 관점에서 볼 때, 계정 이름은 고유 식별자이며 어디에나 사용됩니다. 이전의 ABI는 이러한 계정 이름들을 접두사 길이 문자열(length-prefixed strings)에 따라 직렬화하고 계산상 집약적인 언패킹 프로세스(intensive unpacking process)를 필요로 할 것입니다.
계정에 대한 고정 너비 정수 표시(fixed-width integer representation)로 이동함에 따라 성능을 얻었지만 개발자의 경험은 저하되었습니다. 이 문제를 해결하기 위해서 우리는 정수를 base32로 출력할 것이며, 결과적으로 사람이 읽을 수 있는 계정명으로 바뀔 것입니다.
더 긴 계정명은 네이밍 서비스 컨트랙을 통해 계속 지원될 것입니다.
일반적인 전송 메시지는 이러한 변경 이후 75% 더 줄어듭니다. 이는 초당 수천개의 패킷을 처리할 때 큰 차이를 만듭니다.

WASM에서 부동 소수점 필터링

비결정적 동작을 생성할 수 있는 부동 소수점 연산을 사용하는 코드를 찾아내고(detect) 거부하기 위해 저희는 WASM 프로세싱을 업데이트하였습니다.

WASM 실행시간 샌드박드**(sandbox)**

우리는 시간이 너무 경과하게 되면 실행시간을 체크하고 중단할 수 있도록 검사점에서 WASM를 더하는 방향으로 상당한 발전을 이루었습니다. 이는 신뢰할 수 없는 코드를 실행하는 데 있어서 중요한 요소입니다.

새로운 **C++ Simplecoin (80,000 TPS)**

이전 업데이트 중 하나에서, 저는 C로 쓰여진 개념 증명 통화 스마트 컨트랙트를 보여 주었고, 초당 50K의 트랜잭션을 달성했습니다. 저희가 그 구문을 정리할 수 있을거라고 언급했었는데요. 오늘 저는 WASM으로 컴파일 되어 실행되는 C++에서의 simplecoin의 새로운 구현을 보여드릴 수 있어서 기쁩니다. 저희는 블록체인으로 트랜잭션을 생성하고 제거하는 타이트 루프(tight loop)를 사용하여 단일 쓰레드에서 한개의 계정에서 다른 계정으로 자금을 이체하는 80K TPS를 달성했습니다.

실제 퍼포먼스는 진화하는 아키텍처에 기초하여 더 높거나 낮을 수 있으며, 아직 구체적으로 말하기엔 아직 이르다고 생각합니다. 모든 수치는 *2014*년 *iMac*의 *4Ghz* 인텔코어 *i7*에서 측정되었습니다.

```
struct Transfer { uint64_t from; uint64_t to; uint64_t amount; char
memo[]; };
static_assert( sizeof(Transfer) == 3*sizeof(uint64_t), "unexpected
padding" );
struct Balance { uint64_t balance; };
void on_init() {
static Balance initial = { 1000*1000 };
static AccountName simplecoin;
simplecoin = name_to_int64( "simplecoin" );
store( simplecoin, initial );
}
void apply_simplecoin_transfer() {
static Transfer message;
static Balance from_balance;
static Balance to_balance;
```

```
to_balance.balance = 0;
readMessage( message );
load( message.from, from_balance );
load( message.to, to_balance );
assert( from_balance.balance >= message.amount, "insufficient funds" );
from_balance.balance -= message.amount;
to_balance.balance += message.amount;
if( from_balance.balance )
store( message.from, from_balance );
else
remove( message.from );
store( message.to, to_balance );
}
```

코드가 읽기 더 명확하고 쉬울 뿐 아니라, 생성된 WASM 코드 또한 훨씬 작습니다. 이 컨트랙트의 계정 이름과 ABI를 바꿔줌으로써 대부분의 시간과 공간 절약이 가능해집니다. 이 경우에 ABI는 데이터베이스에서 c++ 구조체까지 zero-parse 복사본을 사용하고 모든 계정 이름은 고정 길이입니다.

이 공유 링크를 사용하여 WasmFiddle에서 이 코드를 실험할 수 있습니다.

결론

EOS개발 팀은 성장하고 있으며, 저희 기술은 매우 높은 성능의 플랫폼을 향해 진보하고 있습니다. 심지어 단일 스레드 측면에서도 그러합니다. 계속될 업데이트에 관심을 가져주시기 바랍니다.