Sui Move: Modern Blockchain Programming with Objects

Adam Welc
adam@mystenlabs.com
Mysten Labs
Palo Alto, California, USA

Sam Blackshear
sam@mystenlabs.com
Mysten Labs
Palo Alto, California, USA

Abstract

This paper presents Sui Move, a new smart contract language for programming blockchains using objects as an abstraction.

CCS Concepts: • **Software and its engineering → Specialized application languages**.

Keywords: programming languages, blockchain, Move, Sui

ACM Reference Format:
Adam Welc and Sam Blackshear. 2023. Sui Move: Modern Blockchain Programming with Objects. In *Companion Proceedings of the 2023 ACM SIGPLAN International Conference on Systems, Programming, Languages, and Applications: Software for Humanity (SPLASH Companion '23), October 22–27, 2023, Cascais, Portugal.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3618305.3623605

1 Introduction

Smart contract languages are a new category of programming languages aimed at programming *blockchains* [10]. A traditional blockchain (e.g., Ethereum [5]) is a persistent distributed list of *blocks* chained together using cryptographic hashes. Blockchain users submit transactions to store their data on chain, which are then sent to a set of distributed *validators*, which have to agree on both validity of each transaction and on the ordering of transactions. Smart contracts are programs written in a smart contract language, which can be called by the user as part of a transaction to read the current blockchain state and store new data on chain.

A data model supported by traditional blockchains is typically *account-based* – user data is read from and written into accounts in a course of a transaction, which makes it difficult to determine potential data access conflicts between transactions. This is why traditional blockchains maintain a total transaction order by executing all transactions sequentially.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
SPLASH Companion '23, October 22–27, 2023, Cascais, Portugal
© 2023 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 979-8-4007-0384-3/23/10…$15.00
https://doi.org/10.1145/3618305.3623605

At the same time, the emerging blockchain applications, such as massively multi-player games, require blockchains to scale orders of magnitude beyond what the traditional blockchains are capable of. The Sui blockchain [11] is an emerging platform with a novel *object-based* data model designed to facilitate fine-grained parallel execution. Sui Move [6] is a smart contract language created to safely and effectively program Sui. Both systems are open source and available at https://github.com/MystenLabs/sui

2 Sui and Sui Move Overview

The main data abstraction to represent user assets (e.g., on-chain currency or NFTs [4]) in the Sui blockchain is an object. In Sui, an object can be directly *owned* by an address representing a blockchain user – more traditional blockchains only allow a notion of indirect ownership: a smart contract would maintain a mapping from user addresses to on-chain data and all data manipulation would have to be mediated by the smart contract. Without direct asset management, traditional smart contract languages such as Solidity [2], the most popular language to program Ethereum, could only provide a limited set of safety features for managing assets.

Sui Move is part of a new wave of smart contract languages that started to emerge to improve the blockchain programming experience. It is a descendant of the Move language [3] originally developed to support smart contract programming for the Diem framework [1]. While Move was certainly a step in the right direction, providing strong language-level safety guarantees for asset management, it was not suited to support Sui's object-based data model. This led to emergence of Sui Move, which adapts the Move language to work with Sui objects, and attempts to make parallel programming as ergonomic as possible while preserving the strong safety guarantees of Move. In the following section we will use an example of a smart contract to describe some of the unique features of the Sui Move language, including *dynamic object fields* [8] and *programmable transaction blocks* [7].

3 Sui Move Programming

We illustrate important features of the Sui Move language using the example in Figure 1, which showcases implementation of a simple `marketplace` module in the `nft` package. The code has been simplified due to space constraints (e.g., marketplace creation has been omitted). Sui Move strongly

```
1  module nft::marketplace {
2    use sui::dynamic_object_field;
3    use sui::{tx_context, object};
4    use sui::{coin, transfer};
5
6    struct Marketplace<T: key+store> has key {
7      id: object::UID,
8    }
9
10   struct Listing<T: key+store> has key, store {
11     id: object::UID, ask: u64,
12     owner: address, item: T,
13   }
14
15   public fun list<T: key+store>(
16     marketplace: &mut Marketplace<T>,
17     item: T, ask: u64, ctx: &mut TxContext
18   ) {
19     let item_id = object::id(&item);
20     let listing = Listing {
21         ask, id: object::new(ctx),
22         owner: tx_context::sender(ctx), item
23     };
24     dynamic_object_field::add(
25       &mut marketplace.id, item_id, listing
26   )}
27
28   public fun buy<T: key + store>(
29     marketplace: &mut Marketplace<T>,
30     item_id: object::ID, paid: coin::Coin<SUI>,
31   ): T {
32      let Listing {
33        id, ask, owner, item
34      } = dynamic_object_field::remove(
35        &mut marketplace.id, item_id
36      );
37      assert!(ask == coin::value(&paid));
38      transfer::public_transfer(paid, owner);
39      object::delete(id);
40      item
41 }}
```

Figure 1. Marketplace contract example

advocates code modularity and certain common functionality related to object management is provided in the Sui framework code (imported in lines 2–4 but omitted for brevity).

A Sui object is represented in Sui Move as a `struct` whose first field is a unique object identifier (`UID`) and which has the `key` modifier (a.k.a. *ability*), for example the `Marketplace` and `Listing` objects (lines 6-8 and 10-13), but also the `Coin` object defined in the Sui framework code. Sui Move supports generic types to allow parameterization of object definitions. For example the `Marketplace` object is parameterized with a specific type of item `T` it can sell, and the `Coin` object is similarly parameterized to allow creating different coin types (in our example we use `Coin<SUI>` which is the main coin in Sui). Objects, such as coins, can be *owned* by a given address representing a blockchain user. Transactions operating on different owned objects can execute in parallel, for example a peer-to-peer trade between different users using different coins. However, objects in Sui can also be shared, and marketplaces are intended to be shared objects. Transactions accessing the same shared object must be totally ordered, but transactions accessing different shared objects can still execute in parallel. In particular, even though trading within the same marketplace will be serialized, trading across marketplaces selling different items can proceed in parallel.

The marketplace supports two types of activity: listing an item and buying an item. When a seller lists an item for sale, the smart contract creates a listing object (lines 20-23) and stores it in the marketplace (lines 24-26). The store operation leverages the *dynamic object fields* [8] mechanism unique to Sui Move, which allows creation of item collections of unbounded size and arbitrary structure. This mechanism allows treating any Sui object as a map from a set of unique IDs (line 19) to a set of arbitrary objects (different listings).

When a buyer wants to purchase an item from a marketplace, it provides the item's unique ID and a payment in the form of a coin. The smart contract then retrieves the item from the marketplace (line 34-36), "unpacks" its content (lines 32-34), transfers payment to the seller (line 38), and returns the purchased item (additional operation in line 39 is needed to inform the system that the listing object has been "unpacked" which effectively deleted it from the chain). If the marketplace does not contain an item with a given ID, or if the payment does not match the asked price (line 37), the call to `buy` will fail, aborting the user transaction. Returning the item from the `buy` function instead of, say, transferring it to the transaction sender, is very intentional – it increases code composability due to another mechanism unique to Sui and Sui Move, that is *programmable transaction blocks* [7] (or *PTBs*). Traditional blockchains can typically process only a single action within a given transaction. On the other hand, PTBs give Sui Move programmers the ability to chain multiple actions (i.e., transaction blocks) together – in particular, a call returning a purchased item may be followed by a transfer operation (to indeed transfer the item) or by another call (to further process the item).

In Sui Move it is also easy to understand data access patterns by just looking at function signatures. It is a useful feature both for developers (to help them avoid asset mismanagement) and for the blockchain users (to give them clarity on what happens to their assets inside a smart contract). In particular, objects passed by a mutable reference (e.g., `&mut marketplace` in line 16) can be modified by a smart contract, whereas objects passed by an immutable one (with the `mut` keyword omitted) will be read-only. Additionally, only owned objects can be passed by value (e.g., item to be listed in line 17), and since the `transfer` function requires its first argument to be passed by value (line 38), only these types of objects can be transferred to other addresses. Sui Move also preserves strong safety guarantees of the Move language [3], such as double-spending [9] prevention.

Last but not least, we extend acknowledgments and huge thanks to all Sui Move developers, particularly Todd Nowacki, Ashok Menon, Dario Russi, and Tim Zakian.

References

[1] Diem Association. 2020. The Diem Blockchain. https://developers.diem.com/docs/technical-papers/the-diem-blockchain-paper.

[2] The Solidity Authors. 2023. Solidity. https://docs.soliditylang.org.

[3] Sam Blackshear, Evan Cheng, David L. Dill, Victor Gao, Ben Maurer, Todd Nowacki, Alistair Pott, Shaz Qadeer, Rain, Dario Russi, Stephane Sezer, Tim Zakian, and Runtian Zhou. 2020. Move: A Language With Programmable Resources. https://developers.diem.com/papers/diem-move-a-language-with-programmable-resources/2020-05-26.pdf.

[4] Robyn Conti. 2023. What Is An NFT? Non-Fungible Tokens Explained. https://www.forbes.com/advisor/investing/cryptocurrency/nft-non-fungible-token.

[5] Ethereum. 2023. Ethereum. https://ethereum.org.

[6] Sui Foundation. 2022. Why We Created Sui Move. https://blog.sui.io/why-we-created-sui-move.

[7] Sui Foundation. 2023. All About Programmable Transaction Blocks. https://blog.sui.io/programmable-transaction-blocks-explained.

[8] Sui Foundation. 2023. Dynamic Fields. https://github.com/sui-foundation/sui-move-intro-course/blob/main/unit-four/lessons/2_dynamic_fields.md.

[9] Jake Frankenfield. 2022. Understanding Double-Spending and How to Prevent Attacks. https://www.investopedia.com/terms/d/doublespending.asp.

[10] Adam Hayes. 2023. Blockchain Facts: What Is It, How It Works, and How It Can Be Used. https://www.investopedia.com/terms/b/blockchain.asp.

[11] The MystenLabs Team. 2023. The Sui Smart Contracts Platform. https://docs.sui.io/paper/sui.pdf.

Received 2023-08-15; accepted 2023-08-30