Sui 智能合约平台

MystenLabs 团队 hello@mystenlabs.com

1 引言

Sui 是一个分散的无许可智能合约平台，倾向于低延迟的资产管理。它使用移动编程语言将资产定义为地址可能拥有的对象。移动程序定义对这些类型化对象的操作，包括创建这些对象的自定义规则、将这些资产转移给新所有者以及对资产进行变异的操作。

Sui 由一组无权限的机构维护，这些机构的作用类似于其他区块链系统中的验证器或矿工。与拜占庭协议相比，它在当局之间使用拜占庭一致的广播协议，以确保资产上常见操作的安全，确保更低的延迟和更好的可扩展性。为了共享对象的安全，它只依赖拜占庭协议。以及治理操作和检查点，在关键延迟路径下执行。在可能的情况下，智能合约的执行也自然并行。Sui 支持可以对读取进行身份验证的轻客户端以及可以审核所有转换完整性的完整客户端。这些设施允许建立到其他区块链的信任最小化桥梁。

本地资产 SUI 用于支付所有作业的天然气费用。其所有者还使用它将股份委托给当局，以便在各个时代内运营 Sui，并且定期根据委托给他们的股份重新配置当局。使用过的天然气根据其股份和对 Sui 运营的贡献分配给当局及其代表。

本白皮书分为两部分，第一节。2.使用 Move 语言描述 Sui 编程模型，以及 Sect。4 描述无许可分散系统的操作，以确保 Sui 的安全性、活性和性能。

2 智能合约编程

Sui 智能合约是用 Move[4]语言编写的。Move 安全且具有表达能力，其类型系统和数据模型自然支持并行协议/执行策略，使 Sui 具有可扩展性。Move 是一种用于构建智能合约的开源编程语言，最初由 Facebook 为 Diem 区块链开发。该语言与平台无关，除了被 Sui 采用外，它还在其他平台（如 0L、StarCoin）上越来越流行。

在本节中，我们将讨论 Move 语言的主要功能，并解释如何使用它在 Sui 上创建和管理资产。Move 编程语言书籍中有对 Move 功能的更透彻解释，Sui 开发者门户网站中有更多针对 Sui 的 Move 内容，第 3 节中有对 Sui 背景下 Move 的更正式描述。[1]

2.1 概述

Sui 的全局状态包括一个由 Move 包创建和管理的可编程对象池，Move 包是包含 Move 函数和类型的 Move 模块的集合（详见第 2.1.1 节）。移动包本身也是对象。因此，Sui 对象可以分为两类：

- 结构数据值：由移动模块控制的类型化数据。每个对象都是一个结构值，其字段可以包含基元类型（例如整数、地址）、其他对象和非对象结构。
- 包代码值：作为原子单元发布的一组相关移动字节码模块。包中的每个模块都可以依赖于该包中的其他模块和以前发布的包中的模块。

对象可以对资产（例如，可替换或不可替换的令牌）进行编码，授予调用某些函数或创建其他对象的权限的功能，管理其他资产的“智能合约”，等等——这由程序员决定。用于声明自定义 Sui 对象类型的移动代码如下所示：

```
结构具有密钥 Obj 公司{
  idu64：VersionedID，//全局唯一 ID 和版本 f://对象可以有基元字段 g:OtherObj//字段还可以存储其他对象
}
```

代表 Sui 对象的所有结构（但不是所有 Move 结构值）必须具有字段和能力，表明该值可以存储在 Sui 的全局对象池中。**id 密钥**[3]

2.1.1 模块。Move 程序被组织为一组模块，每个模块由一系列结构声明和函数声明组成。模块可以从其他模块导入结构类型，并调用其他模块声明的函数。

在一个移动模块中声明的值可以流入另一个移动模块–e、 例如，上述示例中的模块 OtherObj 可以在不同于定义 Obj 的模块中定义。这与大多数智能合约语言不同，后者只允许非结构化字节跨合约边界流动。然而，Move 能够支持这一点，因为它提供了封装特性来帮助程序员编写安全可靠的代码。具体来说，Move 的类型系统确保像上面的 Obj 这样的类型只能由声明该类型的模块内的函数创建、销毁、复制、读取和写入。这允许模块在其声明的类型上强制执行强不变量，即使这些类型跨越智能合约信任边界时，这些类型仍然有效。

2.1.2 交易和入口点。全局对象池通过可以创建、销毁、读取和写入对象的事务进行更新。事务必须将其希望操作的每个现有对象作为输入。此外，事务必须包括包对象的版本 ID、包内模块和函数的名称以及函数的参数（包括输入对象）。例如，调用函数

```
公共娱乐入口点(
o1:Obj，o2:&Obj，o3:&Obj，x:，ctx:&TxContextmut u64mut
```

){…}

事务必须为三个不同的对象提供 ID，其类型为 Obj 和一个要绑定到 x 的整数。TxContext 是一个由运行时填充的特殊参数，其中包含发送方地址和创建新对象所需的信息。

可以使用类型中编码的不同可变权限传递到入口点（更一般地说，传递到任何移动函数）的输入。Obj 输入可以读取、写入、传输或销毁。对象输入只能读取或写入，对象输入只能读取（&O）。必须授权事务发送方使用具有指定可变权限的每个输入对象–有关详细信息，请参阅第 4.4 节。**mut 公司**

*2.1.3 创建和传输对象。*程序员可以通过使用传递到入口点的 TxContext 来创建对象，以生成对象的新 ID：

```
公共娱乐创建然后传输(
  f：，g:OtherObj，o1:Obj，ctx:&TxContextu64mut
){
  let ido2=对象{：TxContext：：fresh_id（ctx），f，g}；传输：：传输（o1，TxContext:sender（））；
  传输：：传输（o2，TxContext:sender（））；
}
```

该代码将 OtherObj 和 Obj 类型的两个对象作为输入，使用第一个对象和生成的 ID 创建新的 Obj，然后将两个 Obj 对象传输给事务发送方。一旦对象被传输，它就会流入全局对象池，并且不能被事务其余部分中的代码访问。传输模块是 Sui 标准库的一部分，其中包括将对象传输到用户地址和其他对象的功能。

我们注意到，如果程序员代码忽略包含一个转移调用，则移动类型系统将拒绝该代码。Move 强制执行资源安全[5]保护，以确保未经许可不能创建、复制或意外破坏对象。资源安全的另一个示例是尝试两次传输同一对象，这也会被移动类型系统拒绝。

3 SUI 规划模型

在本节中，我们通过提供详细的语义定义来扩展第 2 节中对 Sui 编程模型的非正式描述。上一节展示了 Move 源代码的示例；这里我们定义了移动字节码的结构。开发人员编写、测试并正式验证[10，16]在本地移动源代码，然后编译以移动字节码，然后将其发布到区块链。在链上发布的任何移动字节码都必须通过字节码验证器[4，5]，以确保其满足类型、内存和资源安全等关键属性。

如第 2 节所述，Move 是一种与平台无关的语言，可以进行调整以适应不同系统的特定需求，而无需分叉核心语言。在下面的描述中，我们定义了核心移动语言（用黑色文本表示）和扩展核心移动语言（用橙色文本表示）的特定于 Sui 的特征的概念。

3.1 模块

模块=	模块名称× （结构名称~结构 DECL）× （FunName）~FunDecl）×FunDecl
通用参数=	[能力]
结构 DECL=	（字段名~可存储类型）× [能力]×[通用参数]
FunDecl 公司=	[Type][Type]×[Instr]×[GenericParam]
仪表=	TransferToAddr | TransferToObj | ShareMut | ShareImmut |。

表 1：模块

移动代码被组织成模块，其结构在表 1 中定义。模块由一组命名结构声明和一组命名函数声明组成（第 2.1 节提供了这些声明的示例）。模块还包含一个用作模块初始值设定项的特殊函数声明。在链上发布模块时，该函数只调用一次。

结构声明是命名字段的集合，其中字段名映射到可存储类型。它的声明还包括一个可选的能力列表（有关可存储类型和能力的描述，请参阅第 2 节）。结构声明还可以包括具有能力约束的泛型参数列表，在这种情况下，我们将其称为泛型结构声明，例如 Wrapper<T:>{T:T}。泛型参数表示在声明结构字段时使用的类型–在声明结构时未知，在实例化结构时（即创建结构值时）提供了具体类型。**结构副本**

函数声明包括参数类型列表、返回类型列表和构成函数体的指令列表。函数声明还可以包括具有能力约束的泛型参数列表，在这种情况下，我们将其称为泛型函数声明，例如 unwrap<T:>（p:Wrapper<T>）{}。与结构声明类似，泛型参数表示在函数声明时未知的类型，但在声明函数参数、返回值和函数体时使用它（调用函数时提供具体类型）。**有趣的副本**

可以出现在函数体中的指令包括除全局存储指令（例如，，）之外的所有普通移动指令。有关 core Move 指令及其语义的完整列表，请参见[14]。在 Sui 中，持久存储通过 Sui 的全局对象池而不是核心移动的基于帐户的全局存储来支持。move\u tomove\u frombrow\u global

有四种特定于 Sui 的对象操作。每个操作都会更改对象的所有权元数据（请参见第 3.3 节），并将其返回到全局对象池。最简单的方法是，可以将 Sui 对象传输到 Sui 最终用户的地址。一个对象也可以传输到另一个父对象–此操作要求调用者除了提供子对象外，还提供对父对象的可变引用。一个对象可以可变地共享，因此它可以被 Sui 系统中的任何人读/写。最后，一个对象可以不可变地共享，因此它可以被 Sui 系统中的任何人读取，但不能被任何人写入。

区分不同所有权类型的能力是 Sui 的一个独特特征。在我们知道的其他区块链平台中，每个合约和对象都是相互共享的。正如我们将在第 4 节中解释的那样，Sui 利用这些信息执行并行事务（对于所有事务）和并行协议（对于涉及没有共享可变性的对象的事务）。

3.2 类型和能力

PrimType 公司=	{地址，idbool，u8，u64，…},
结构类型=	ModuleName×StructName×[存储类型]
可存储类型=	PrimType 公司⊎结构类型⊎通用类型⊎矢量类型
矢量类型=	可存储类型
通用类型=	N
易变性相等=	{mut，immut}
参考类型=	可存储类型×可变质量
类型=	参考类型⊎可存储类型
能力=	{key，store，copy，drop}

表 2：类型和能力

移动程序处理存储在 Sui 全局对象池中的数据和移动程序执行时创建的瞬态数据。对象和瞬态数据都是语言级别的移动值。然而，并非所有的值都是平等的——它们可能具有不同的属性和不同的结构，正如它们的类型所规定的那样。

表 2 中定义了 Move 中使用的类型。Move 支持其他编程语言中支持的许多相同的基元类型，例如布尔类型或各种大小的无符号整数类型。此外，核心移动有一个表示系统中最终用户的类型，该类型还用于识别交易的发送者和（在 Sui 中）对象的所有者。最后，Sui 定义了一个表示 Sui 对象的标识的类型——有关详细信息，请参见第 3.3 节。地址 id

结构类型描述在给定模块中声明的结构的实例（即值）（有关结构声明的信息，请参阅第 3.1 节）。表示泛型结构声明的结构类型（即泛型结构类型）包括一个可存储类型列表——该列表与结构声明中的泛型参数列表相对应。可存储类型可以是具体类型（基元或结构）或泛型类型。我们称这些类型为可存储的，因为它们可以作为结构的字段和持久存储在链上的对象中，而引用类型则不能。

例如，包装器<>结构类型是用具体（原始）可存储类型参数化的通用结构类型–这种类型可以用于创建结构实例（即值）。另一方面，相同的泛型结构类型可以用来自封闭结构或函数声明的泛型参数的泛型类型（例如，父类）参数化–这种类型可以用于从结构上声明结构字段、函数参数等，泛型类型是在封闭结构或函数声明中的泛型参数列表中的整数索引（在表 5 中定义为 N）。u64u64 结构

Move 中的向量类型描述了同质值的可变长度集合。移动向量只能包含可存储类型，它本身也是可存储类型。

移动程序可以直接对值进行操作，也可以通过引用间接访问值。referencetype 包括引用的可存储类型和用于确定（并强制）给定类型的值是可以读写（mut）还是只能读（immut）的可变性限定符。因此，移动值类型（表 2 中的类型）的最常见形式可以是可存储类型或引用类型。

最后，移动中的能力控制对给定类型的值允许哪些操作，例如是否可以复制（复制）给定类型的值。能力约束结构声明和泛型类型参数。移动字节码验证器负责确保像副本这样的敏感操作只能在具有相应能力的类型上执行。

3.3 对象和所有权

TxDigest 公司=	*Com*（德克萨斯州）

对象=	*Com*（TxDigest×N）
单一所有者=	地址⊎对象
共享=	{shared\u mut，shared\u immut}
所有权=	单一所有者⊎共享
StructObj 结构=	结构类型×结构
对象内容=	StructObj 结构⊎包裹
Obj 公司=	对象内容×对象×所有权×版本

表 3：对象和所有权

每个 Sui 对象都有一个全局唯一标识符（表 3 中的 ObjID），当对象在所有者之间流动以及进出其他对象时，该标识符作为对象的持久标识。该 ID 由创建该对象的事务分配给该对象。对象 ID 是通过对当前事务的内容和记录事务创建了多少对象的计数器应用抗冲突哈希函数来创建的。由于事务输入对象的约束，事务（及其摘要）保证是唯一的，我们将在后面解释。

除了 ID 之外，每个对象还携带关于其所有权的元数据。对象由一个地址或另一个对象唯一拥有，与写/读权限共享，或仅与读权限共享。对象的所有权决定了事务是否以及如何将其用作输入。一般来说，唯一拥有的对象只能在其所有者发起的事务中使用，或将其父对象作为输入包含在内，而共享对象可以由任何事务使用，但只能具有指定的可变权限。完整解释见第 4.4 节。

有两种类型的对象：包代码对象和结构数据对象。包对象包含一个模块列表。结构对象包含移动结构值和该值的移动类型。对象的内容可能会更改，但其 ID、对象类型（包与结构）和移动结构类型是不变的。这确保了对象是强类型的，并且具有持久标识。

最后，一个对象包含一个版本。新创建的对象的版本为 0，每次事务将对象作为输入时，对象的版本都会增加。

3.4 地址和身份验证器

验证器=	Ed25519PubKey⊎ECDSAPubKey⊎. . .
地址=	*Com*（验证器）

表 4：地址和验证器

地址是终端用户的持久身份（尽管请注意，单个用户可以有任意数量的地址）。要将对象传输给另一个用户，发件人必须知道收件人的地址。

正如我们稍后将讨论的那样，Sui 事务必须包含发送（即发起）事务的用户的地址和其摘要与该地址匹配的验证器。地址和身份验证器之间的分离实现了加密灵活性。验证器可以是任何签名方案的公钥，即使这些方案使用不同的密钥长度（例如，支持后量子签名）。此外，验证器不需要是单个公钥–它也可以是（例如）K-of-N 多签名密钥。

3.5 交易

ObjRef=	ObjID×版本×*Com*（Obj）
呼叫目标=	ObjRef×ModuleName×FunName
卡拉格=	ObjRef⊎对象⊎PrimType 公司
包装=	[模块]
发布=	包×[ObjRef]
呼叫=	调用目标×[StorableType]×[CallArg]
气体信息=	ObjRef×MaxGas×BaseFee×Tip
德克萨斯州=	（呼叫⊎发布）×GasInfo×Addr×Authenticator

表 5：交易

Sui 有两种不同的事务类型：发布新的 Move 包和调用以前发布的 Move 包。发布事务包含一个包–将作为单个对象一起发布的一组模块，以及该包中所有模块的依赖关系（编码为必须引用已发布包对象的对象引用列表）。要执行发布事务，Sui 运行时将在每个包上运行移动字节码验证器，根据其依赖项链接包，并运行每个模块的模块初始化器。模块初始值设定项用于引导由包实现的应用程序的初始状态。

调用事务最重要的参数是对象输入。对象参数可以通过对象引用（对于单个所有者和共享的不可变对象）或对象 ID（对于共享可变对象）。对象引用由对象 ID、对象版本和对象值的哈希组成。Sui 运行时将对象 ID 和对象引用解析为存储在全局对象池中的对象值。对于对象引用，运行时将对照池中对象的版本检查引用的版本，并检查引用的哈希是否与池对象匹配。这确保了运行时的对象视图与事务发送方的对象视图相匹配。

此外，调用事务接受类型参数和纯值参数。类型参数实例化要调用的入口点函数的通用类型参数（例如，如果入口点函数是 send\u coin<T>（c:coin<T>，…），泛型类型参数 T 可以用类型参数 SUI 实例化，以发送 SUI 本机令牌）。纯值可以包括基元类型和基元类型的向量，但不能包括结构类型。

调用调用的函数通过对象引用（必须引用包对象）、该包中模块的名称和该包中函数的名称指定。要执行调用事务，Sui 运行时将解析函数，将类型、对象和值参数绑定到函数参数，并使用 Move-VM 执行函数。

呼叫和发布交易均受天然气计量和天然气费用的约束。计量极限由最大气体预算表示。运行时将执行事务，直到达到预算为止，如果预算用完，运行时将终止事务，而不产生任何影响（扣除费用和报告终止代码除外）。

费用从指定为对象参考的气体对象中扣除。该对象必须是隋本地令牌（即，其类型必须是硬币）。Sui 使用 EIP1559 风格的费用：协议定义了一个基本费用（以每个 Sui 令牌的气体单位表示），该费用在历元边界进行算法调整，交易发送方还可以包括一个可选的 tip（以 Sui 令牌表示）。在正常的系统负载下，即使没有提示，也会迅速处理事务。然而，如果系统拥塞，则具有较大 tip 的事务将优先处理。从气体对象推断的总费用为（气体使用* 基本费用）+小费。[4]

3.6 交易影响

事件=	结构类型×结构
创建=	Obj 公司
更新=	Obj 公司
包裹=	ObjID×版本
删除=	ObjID×版本
对象=	创造⊎ 使现代化⊎ 包⊎ 删去
中止代码=	N×ModuleName
成功效应=	[对象]×[事件]
流产影响=	中止代码
TX 效应=	成功效应⊎ 流产影响

表 6：交易影响

交易执行产生的交易效果在交易成功执行的情况下有所不同
（表 6 中的成功效果）和不成功时（表 6 中的中止效果）。

成功执行事务后，事务影响包括有关对 Sui 的全局对象池所做的更改（包括对现有对象和新创建的对象的更新）以及事务执行期间生成的事件的信息。成功执行事务的另一个效果可能是从全局池中删除对象（即删除），并将一个对象包装（即嵌入）到另一个对象中，这与删除具有类似的效果-包装的对象从全局池中消失，并且仅作为包装它的对象的一部分存在。由于删除和包装的对象在全局池中不再可访问，因此这些影响由对象的 ID 和版本表示。

除了更新全局对象池外，事件还编码了成功执行事务的副作用。从结构上讲，事件由移动结构及其类型组成。事件旨在被区块链之外的参与者消费，但不能被移动程序读取。

移动中的事务具有全有或全无语义——如果事务的执行在某个点中止（例如，由于意外故障），即使在此点之前对对象发生了一些更改（或生成了一些事件），这些影响都不会在中止的事务中持续。相反，中止的事务效果包括数字中止代码和发生事务中止的模块的名称。中止交易仍需收取汽油费。

4 SUI 系统

在本节中，我们从系统的角度描述了 Sui，包括在拜占庭式失败的情况下确保跨当局安全和活力的机制。我们还解释了客户端的操作，包括需要在不验证其完整状态的情况下确保系统状态的轻客户端。

简要背景。在系统级，Sui 是 FastPay[3]低延迟结算系统的演变，通过用户定义的智能合同扩展到对任意对象进行操作，并具有无许可委托的利害关系委员会组成证明[2]。对象所有者的基本资产管理基于拜占庭一致广播[6]的变体，与拜占庭共识的传统实现[8、11、12]相比，拜占庭一致广播具有更低的延迟，并且更容易在许多机器上扩展。当需要完全一致时，我们使用基于高吞吐量 DAG 的共识，例如[9]来管理锁，同时并行执行不同的共享对象。

协议大纲。图 1 说明了客户和 Sui 机构之间的高层交互，以提交交易。我们在此简要介绍：

- 具有私人签名密钥的用户创建并签署用户事务，以在 Sui 中变异他们拥有的对象或共享对象。随后，不需要用户签名密钥，剩余的过程可以由用户客户端或代表用户的网关执行（在图中表示为无密钥操作）。
- 用户交易被发送给 Sui 机构，每个机构都会检查其有效性，成功后签署并将签署的交易返回给客户。客户从法定人数的机构收集回复，以形成交易证书。
- 然后将交易证书发送回所有当局，如果交易涉及共享对象，则还将其发送到由 Sui 当局操作的拜占庭协议协议。当局检查证书，如果涉及共享对象，还需要等待协议协议将其与其他共享对象事务排序，然后执行事务并将其效果汇总为签名效果响应。
- 一旦法定人数的机构签署了证书，其效力即为最终效力（在图表中表示为最终效力）。客户可以收集法定数量的权威机构回复，创建影响证书，并将其用作交易影响最终性的证明。

本节详细描述了这些操作中的每一个，以及跨权限重新配置和管理状态的操作。

4.1 系统模型

隋在一系列的时代中运作$e \in$ {0, . . .}. 每个时代都由一个委员会管理C= (V（·））， 其中V是一组具有已知公共验证密钥和网络端点的权限。函数$S(v)$ 映射每个权限$v \in V$多个委托股份单位。我们假设C对于每个 epoch，由 epoch 的法定代表人（见下文）签署e –1.（第 4.7 节讨论了委员会的组建和管理）。在一个时代内，一些权威是正确的（他们忠实地遵守协议并且是活的），而另一些是拜占庭式的（他们任意偏离协议）。安全性假设是一组诚实的权限$H \subseteq V$在新纪元内分配了法定人数的股份，即：。$\in He$ () 2/3 $\in Ve$ （）（并指法定人数超过三分之二股份的任何机构）。$_{ee}S_{eeeeeee}$ $\sum_{h}^{} S_{e}\ h > \sum_{v}^{} S_{e}\ v$

至少存在一个活跃且正确的一方，作为诚信机构之间每个证书的中继（见第 4.3 节）。这确保了活跃性，并为拜占庭广播提供了最终交付属性（见[6]中的可靠广播总体）。每个机构单独或通过集体传播协议操作此类中继。包括 Sui light 客户端、副本和服务在内的外部实体也可能扮演这一角色。被动授权核心与不太可靠或受信任的内部或外部主动中继组件之间的区别确保了对可信计算基础[15]的明确划分和最小化，而 Sui 的安全性和活性依赖于此。

4.2 权限和副本数据结构

隋当局依靠许多数据结构来代表国家。我们根据它们支持的操作定义这些结构。它们都具有确定性字节表示。

对象（Obj）在 Sui 中存储用户智能合约和数据。它们是第节中介绍的移动对象的 Sui 系统级编码。2、它们支持以下一组操作：

- （Obj）返回对象的引用（ObjRef），即三元组（ObjID，Version，ObjDigest）。ObjID 实际上是*裁判*

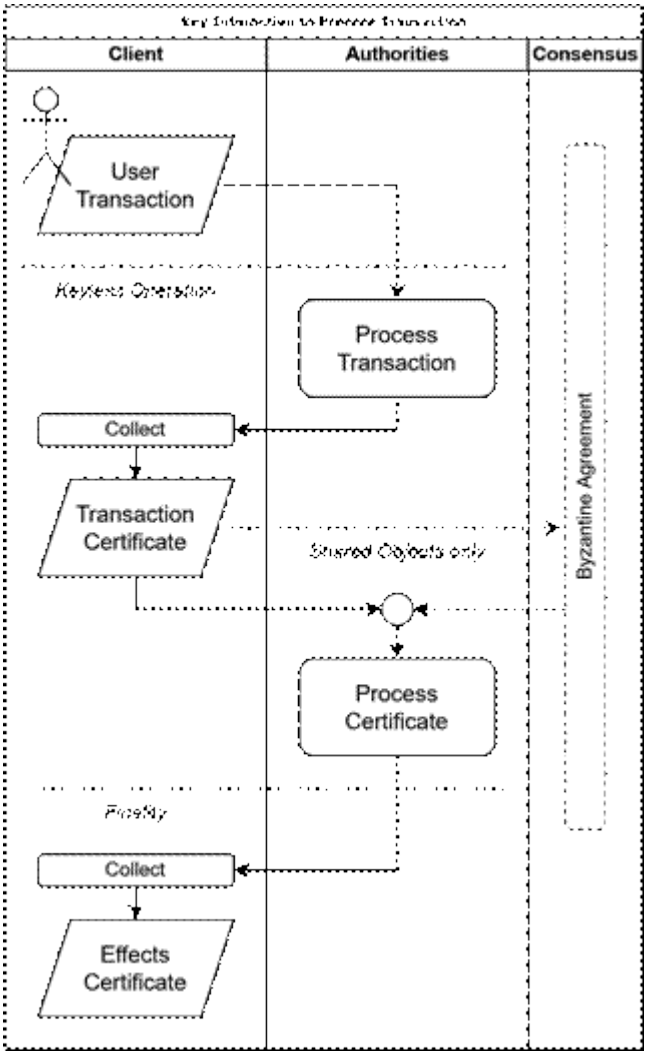

图 1：提交事务的交互概述。对于所有创建的新对象来说都是唯一的，版本是一个递增的正整数，表示正在变异的对象版本。

- （Obj）返回对象所有者的验证器 Auth。在最简单的情况下，Auth 是一个地址，表示可以使用此对象的公钥。也可以使用更复杂的验证器（见第 4.4 节）。*物主*
- （Obj）如果对象为只读，则返回 true。只读对象可能永远不会被变异、包装或删除。他们也可能被任何人使用，而不仅仅是他们的主人。*只读*
- （Obj）返回上次变异或创建对象的事务摘要（TxDigest）。*父母亲*
- （Obj）返回可用于检查交易有效性的对象类型和数据，并携带对象的应用程序特定信息。*目录*

对象引用（ObjRef）用于索引对象。它还用于验证对象，因为 ObjDigest 是对其全部内容的承诺。

事务（Tx）是表示一个或多个对象的状态转换的结构。它们支持以下一组操作：

- （Tx）返回 TxDigest，这是对交易具有约束力的加密承诺。*消化*
- （Tx）返回执行该交易的时间点。*纪元*
- （Tx）返回事务需要执行的对象[ObjRef]序列。*输入*
- （Tx）返回用于支付天然气的 ObjRef 的参考，以及最大天然气限制，以及天然气支付对象中天然气单位和价值单位之间的转换率。*付款*

- （Tx，[Obj]）如果事务有效，则在提供请求的输入对象的情况下返回 true。有效性在第节中讨论。4.4，并涉及被授权对输入对象采取行动的交易，以及足以支付其执行成本的可用天然气。*有效的*
- （Tx，[Obj]）执行交易并返回表示其影响的结构效果。有效的事务执行是绝对正确的，其输出是确定性的。*执行董事*

事务由其 TxDigest 编制索引，TxDigest 也可用于验证其完整内容。所有有效交易（特殊硬编码 genesis 交易除外）至少有一个自有输入，即用于支付天然气的对象。

事务效果（effects）结构总结了事务执行的结果。它支持以下操作：

- （效果）是对效果结构的承诺，可用于索引或验证。*消化*
- （Effects）返回产生影响的已执行交易的 TxDigest。*交易*
- （Effects）返回一系列依赖项[TxDigest]，这些依赖项应该在具有这些影响的事务可能执行之前执行。*依赖项*
- （效果）返回执行摘要。状态报告智能合约执行的结果。创建、修改、包装、取消包装和删除的列表列出了经过相应操作的对象引用。事件列出了执行发出的事件。*日录*

交易上的交易证书 TxCert 包含交易本身以及来自法定机构的标识符和签名。请注意，证书可能不是唯一的，因为同一逻辑证书可能由构成法定人数的不同权限集表示。此外，证书可能不会严格由 2/3 的法定人数签署，但如果有更多的机构响应，则可能会有更多的法定人数签署。但是，同一事务上的两个不同有效证书应视为在语义上表示同一证书。部分证书（TxSign）包含相同的信息，但来自一组代表低于所需法定人数的权限的签名，通常是单个权限。证书中包含签名者的标识符（即责任签名[？]）确定准备处理证书的机构，或用于下载处理证书所需的过去信息的机构（见第 4.8 节）。

类似地，effects 结构上的 effects 证书 EffCert 包含 effects 结构本身和来自代表事务有效期法定人数的权威机构的签名。关于非唯一性和身份的警告同样适用于交易证书。部分效力证书，通常包含单个授权签名，效力结构表示为 EffSign。[5]

持久存储。每个授权和副本都维护一组持久存储。存储实现持久映射语义，可以表示为一组键值对（表示为 $map[key] \rightarrow value$), 这样，只有一对具有给定密钥。在插入配对之前，包含(key) call 返回 false，并获取(key) 返回错误。插入配对后包含(key) 调用返回 true，并获取(key) 返回值。当局维护以下持久存储：

- 订单锁映射锁[ObjRef]→ TxSignOption 记录管理局为拥有的对象版本 ObjRef 看到和签署的第一个有效事务 Tx，如果对象版本存在，但没有有效事务将其用作已看到的输入，则记录无。它还可以记录与此对象一起看到的第一个证书作为输入。该表及其更新规则表示跨 Sui 权限的对象上分布式锁的状态，并确保事务并发处理下的安全。v
- 证书映射 Ct[TxDigest]→ （TxCert，EffSign）记录管理局在其有效期内处理的所有完整证书 TxCert，其中还包括 Tx，以及其签名效果 EffSign。它们由事务摘要 TxDigest 索引v
- 对象映射 Obj[ObjRef]→ Obj 记录由 ObjRef 索引的 CTINDEX 中的证书中包含的事务创建的所有对象 Obj。通过重新执行 Ct 中的所有证书，可以完全派生此存储。维护一个二级索引，将 ObjID 映射到具有此 ID 的最新对象。这是处理新事务所需的唯一信息，而维护旧版本只是为了方便读取和审计。$_{v}vv$
- 同步映射同步[ObjRef]→ TxDigest 根据它们创建、变异或删除的对象，将 CTF 中的所有证书作为元组 ObjRef 进行索引。通过处理 Ct 中的所有证书，可以完全重新创建此结构，并用于帮助客户端同步影响其关心的对象的事务。$_{v}vv$

当局维护所有四个结构，并提供对其证书映射的本地检查点的访问，以允许其他当局和副本下载其完整的已处理证书集。副本不处理事务，只处理证书，并像权威机构那样重新执行它们以更新其他表。它还维护一个顺序锁映射，以审核非模糊性。

权限可以设计为维护所有四个存储（和检查点）的完整副本，以便于读取和同步，并结合最小权限核心，该核心仅维护用于处理新事务和证书的最新版本对象的对象锁和对象。这最大限度地减少了安全所依赖的可信计算基础。

只有顺序锁映射需要强键自一致性，即对键的读取应该始终返回存在的键是否存在值，并且这种检查应该是原子的，更新时将锁设置为非 None 值。这是一个弱于键间强一致性的属性，并允许对存储进行有效切分以进行扩展。其他门店最终可能会保持一致，而不会影响安全。

4.3 权限基础操作

处理事务。在收到交易 Tx 后，当局会进行大量检查：

（1） 它确保 epoch（Tx）是当前 epoch。

（2） 它确保所有对象参考输入（Tx）和付款中的气体对象参考（Tx）存在于 OBJAN 中，并将其加载到[Obj]。对于拥有的对象，应提供准确的参考；对于只读或共享对象，对象 ID 应该存在。$_{v}$

（3） 确保气体对象中有足够的气体可用，以支付执行交易的成本。

（4） 它检查 valid（Tx，[Obj]）是否为 true。此步骤确保事务中的身份验证信息允许访问所拥有的对象。

（5） 它检查所有拥有的输入（Tx）对象的锁[ObjRef]是否存在，它要么为 None，要么设置为相同的 Tx，并自动将其设置为 TxSign。（我们称之为“锁检查”）。v

如果任何检查失败，则处理结束，并返回错误。然而，锁的部分更新是安全的（尽管我们当前的实现不进行部分更新，但对所有锁进行原子更新）。v

如果所有检查均成功，则管理局将返回交易的签名，即部分证书 TxSign。成功处理订单时是幂等的，并返回部分证书（TxSign）或完整证书（TxCert）（如果有）。

任何一方均可为构成 epoch 法定人数的一组机构核对交易和签名（TxSign）*e*, 形成交易证书 TxCert。

工艺证书。在收到证书后，机构会检查交易的所有有效性条件，但与锁相关的条件除外（所谓的“锁检查”）。相反，它执行以下检查：对于输入（Tx）中的每个拥有的输入对象，它检查锁是否存在，以及锁是否为无、是否设置为任何 TxSign，或者是否设置为与当前证书相同事务的证书。如果此修改的锁检查失败，则管理局已检测到无法恢复的拜占庭故障，停止正常操作，并启动灾难恢复过程。对于共享对象（见第 4.4 节），当局检查是否已通过一致排序的证书设置了锁，以确定要使用的共享对象的版本。如果是这样，则可以执行交易；否则，它需要先等待这种排序。

如果检查成功，管理局会将证书添加到其证书映射中，以及其执行所产生的影响，即 Ct[TxDigest]→（TxCert，EffSign）；它更新锁映射以记录证书锁[ObjRef]→ TxCert 用于所有拥有的输入对象，这些对象的锁未设置为证书。一旦输入（Tx）中的所有对象插入到 Obj 中，EffSign 中的所有效果也会通过向 Obj 添加其 ObjRef 和内容来具体化。最后，对于 EffSign 中创建或变异的所有对象，更新同步映射以将其映射到 Tx。vv_{vv}

评论。处理事务和证书的逻辑产生了许多重要属性：

- 因果关系和平行性。交易和证书的处理条件都确保了因果执行：如果管理机构已经处理了创建交易所依赖的对象的所有证书（包括所有的、共享的和只读的），则只通过签署交易进行“投票”。类似地，只有当它所依赖的所有输入对象都存在于其本地对象映射中时，权限才会处理证书。这规定了因果执行顺序，但也允许在不同的核心或机器上并行执行彼此不因果依赖的事务。
- 签字一次，确保安全。锁[·]中所有拥有的输入对象锁被设置为第一个使用它们通过检查的事务 Tx，然后是第一个使用该对象作为输入的证书。我们称之为将对象锁定到此事务，并且在一个时期内没有解锁。因此，一个机构只对每个锁签署一个事务，这是一致广播[6]的一个重要组成部分，从而保证了 Sui 的安全性。v
- 灾难恢复。检测到同一锁的两个相互矛盾的证书的机构，具有不可恢复拜占庭行为的证明，即证明法定人数诚实权限假设不成立。这两个相互矛盾的证书是防欺诈的[1]，可以与所有机构和副本共享，以触发灾难恢复过程。当局还可能获得其他形式的不可恢复拜占庭行为的证明，例如超过 1/3 的影响签名（EffSign），表示证书执行不正确。或包含输入对象的证书，这些输入对象不代表先前处理的证书的正确输出。这些文件也可以打包为防欺诈文件，并与所有机构和副本共享。注意，这些证据不同于少数权威机构的证明(≤ 1/3（按桩计算）或对象所有者（任何数字）是拜占庭式或模棱两可的，这可以在不中断任何服务的情况下容忍。
- 最终性。对于 Lock、ctan 和 Obj、Sync 中的索引的任何读取请求，权限将返回证书（TxCert）和签名效果（EffSign）。如果超过法定人数的机构报告其 Ctstore 中包含的 Tx，则交易被视为最终交易。这意味着效力证书（EffCert）是可转让的最终性证明。然而，使用对象的证书也证明其因果路径中的所有从属证书也是最终的。向任何一方提供证书，然后将其提交给绝大多数当局处理，也可以确保证书效力的最终性。请注意，最终性晚于 fastpay[3]，以确保重新配置时的安全性。然而，授权机构可以在看到证书而不是等待提交时应用事务的效果。$vv_{vv}v$

4.4 所有者、授权和共享对象

交易有效性（见第 4.3 节）确保交易被授权在交易中包括所有指定的输入对象。此检查取决于对象的性质以及所有者字段。

*只读对象*不能被变异或删除，可以在事务中同时使用，也可以由所有用户使用。例如，移动模块是只读的。这些对象确实有一个所有者，可以作为智能合约的一部分使用，但这并不影响使用它们的授权。它们可以包含在任何交易中。

*拥有的对象*拥有所有者字段。所有者可以设置为表示公钥的地址。在这种情况下，如果事务由该地址签名，则授权事务使用该对象并对其进行变异。事务由单个地址签名，因此可以使用该地址拥有的一个或多个对象。但是，单个事务不能使用由多个地址拥有的对象。对象（称为子对象）的所有者可以设置为另一个对象（称为父对象）的对象 ID。在这种情况下，只有当父对象包含在事务中，并且事务有权使用该对象时，才可以使用子对象。合同可以使用此功能来构建高效的集合和其他复杂的数据结构。

*共享对象*是可变的，但没有特定的所有者。相反，它们可以包含在不同方的交易中，并且不需要任何授权。相反，它们执行自己的授权逻辑。由于这些对象必须支持多个编写器，同时确保安全性和活性，因此需要安全使用完整的协议协议。因此，它们在执行之前需要额外的逻辑。当局按照第节的规定处理交易。4.3 为拥有的对象和只读对象管理其锁。然而，当局并不依赖一致广播来管理共享对象的锁。相反，涉及共享对象的事务的创建者将事务上的证书插入到高通量共识系统中，例如[9]。所有机构都遵守此类证书的一致序列，并根据该序列分配每个事务使用的共享对象的版本。然后可以继续执行，并保证在所有当局之间保持一致。权限包括在生效证书内的事务执行中使用的共享对象的版本。

上述规则确保涉及只读和所有对象的事务的执行只需要一致的广播和单个证书即可继续；只有涉及共享对象的事务才需要拜占庭协议。因此，智能合约作者可以设计其类型和操作，以优化单个用户对象上的传输和其他操作，从而降低延迟，同时享受使用共享对象实现需要多个用户访问的逻辑的灵活性。

4.5 客户

完整客户端和副本。副本有时也称为 fullclients，它不验证新事务，但维护系统有效状态的一致副本，以用于审计，以及构建事务或操作服务，包括用于轻客户端查询的读取基础结构。

轻客户端。对象引用和事务都包含允许验证导致其创建或执行的完整因果链的信息。具体来说，对象引用（ObjRef）包含一个 ObjDigest，它是对象完整状态的验证器，包括获取父对象的工具（Obj），即创建对象的 TxDigest。类似地，TxDigest 对事务进行身份验证，包括通过输入（Tx）提取输入对象的对象引用的功能。因此，对象和证书集形成了自认证的二部图。此外，还对 effects 结构进行签名，并将其整理成 effects 证书，直接证明事务执行的结果。

这些设施可用于支持轻客户端，这些轻客户端可以对 Sui 状态执行高完整性读取，而无需维护完整副本节点。具体而言，授权或完整节点可以提供简洁的证据束，包括事务 Tx 上的证书 TxCert 和对应于输入（Tx）的输入对象[Obj]，以使轻客户端确信可以在 Sui 内发生转换。然后，轻客户可以提交此证书，或检查是否已被法定人数或权威机构样本看到，以确保最终性。或者，它可以使用执行产生的对象创建一个事务，并观察它是否成功。

更直接地说，服务可以向客户提供效果证书，以使他们相信 Sui 中的转换的存在和最终性，而不需要在系统中进行进一步的操作或交互。如果最终证书的检查点可用，在历元边界或其他位置，包括输入对象和证书的一束证据，以及证书包含在检查点的证明也是最终性证明。

当局可以使用定期检查机制来创建最终交易的集体检查点，以及随时间变化的 Sui 状态。light 客户端可以使用在检查点上具有法定人数的证书来有效验证对象和发射事件的最近状态。检查点机制对于各时代之间的委员会重组是必要的。更频繁的检查点对于轻型客户端很有用，当局也可以使用它来压缩其内部数据结构，以及更有效地将其状态与其他当局同步。

4.6 桥梁

通过拜占庭协议管理的对轻客户端和共享对象的本地支持，Sui 可以支持到其他区块链的双向桥梁[13]。此类网桥的信任假设反映了 Sui 和其他区块链的信任假设，如果其他区块链也支持轻客户端，则不必依赖受信任的预言机或硬件[7]。

桥接用于导入在另一个区块链上发行的资产，以表示该资产，并将其用作 Sui 系统内的包装资产。最终，打包的资产可以解锁并转移回本地区块链上的用户。桥接还可以允许锁定在 Sui 上发行的资产，并将其用作其他区块链上的包装资产。最终，可以销毁另一个系统上的包装对象，并更新 Sui 上的对象以反映国家或所有权的任何变化，然后解锁。

桥接资产的语义对于确保包装资产的有用性具有一定的重要性。跨区块链桥接的可替代资产可以提供更丰富的包装表示，使其在包装时可以分割和转移。不可分割资产不可分割，但只能转让。它们还可能支持在包装时以受控方式改变其状态的其他操作，这可能需要在桥接和展开时执行自定义智能合约逻辑。Sui 非常灵活，允许智能合约作者定义此类体验，因为桥梁只是在 Move 中实现的智能合约，而不是原生的 Sui 概念，因此可以使用 Move 提供的可组合性和安全保证进行扩展。

4.7 委员会重组

重新配置发生在委员会成立的两个时代之间C由一个委员会取代C', 哪里$e = e + 1$.重新配置安全性确保如果事务发送在e 或之前，之后不能提交冲突事务e. 活跃度确保发送在或之前提交e, 那么它也必须在e.$_{e}e'$

我们利用 Sui 智能合约系统来执行重构所需的大量工作。在 Sui 中，系统智能合约允许用户锁定股份并将其委托给候选机构。在一个时代，硬币的所有者可以通过锁定代币来自由授权，通过解锁代币来取消授权，或者将其授权更改为一个或多个机构。

一旦达到时代法定人数e 投票结束新纪元，当局交换信息，承诺设立一个检查站，决定下一个委员会，并改变新纪元。首先，当局在协议协议[9]的帮助下运行一个检查点协议，以在新纪元结束时商定一个经认证的检查点e. 检查点包含所有事务的联合，以

及可能产生的对象，这些事务和对象已由仲裁机构处理。因此，如果交易已由法定人数的机构处理，则至少一个处理该交易的诚实机构将其处理的交易包括在时代末检查点中，以确保交易及其影响在各个时代都是持久的。此外，这样一个经过认证的检查点保证了所有交易都可供 epoch 的诚实权威机构使用e.

然后，使用 epoch 检查点结束时的桩委托来确定 epoch 的新权限集$e+1$.由法定人数的旧机构股份和法定人数的新机构股份签署新委员会C', 以及新纪元开始的关卡。一旦两组签名都可用，新的一组权限开始处理新纪元的事务，旧权限可以删除其纪元签名密钥。$_e$

恢复由于客户端错误或客户端含糊其词，拥有的对象可能在一个 epoch 内被“锁定”，从而阻止与其相关的任何交易被认证（或最终确定）。例如，一个客户端使用相同的拥有对象版本签署两个不同的交易，每个交易都有一半的权限进行签名，这将无法形成要求两个证书中任何一个上的签名达到法定人数的证书。恢复可确保一旦年代发生变化，此类对象再次处于允许在事务中使用的状态。由于无法形成证书，因此在要操作的下一个纪元开始时，原始对象可用。由于事务包含一个历元数，旧的模棱两可的事务将不会再次锁定对象，使其所有者有机会使用它。

奖励与加密经济学。隋有一个土生土长的令牌隋，有固定的供应。SUI 用于支付天然气费用，也被用作一个时代内当局的委托股份。在这个时代，当局的投票权是这种委托股份的函数。在新纪元结束时，通过处理的所有交易收取的费用将根据其对 Sui 运营的贡献分配给当局，反过来，他们将部分费用作为奖励分享给向其委托股份的地址。我们将对隋象征经济学的完整描述推迟到专门的论文中。

4.8 权限和副本更新

客户驱动。由于客户端故障或非拜占庭机构故障，一些机构可能没有处理所有证书。因此，依赖于这些证书生成的缺失对象的因果关联交易将被拒绝。然而，客户始终可以更新诚实权威，使其能够处理正确的交易。它可以使用自己的过去证书存储来实现这一点，或者使用一个或多个其他诚实的权威机构作为过去证书的来源。

获得证书c 和 aCt商店包括c 以及其因果历史，客户可以更新诚实的权威v达到以下程度：c 也将适用。这涉及到查找不在中的最小证书集v这样，当应用对象时v包括所有输入c. 更新滞后权限B 使用商店Ct包括证书 TxCert 涉及：$_{v'''}{}_{v}$

- 客户端维护一个要同步的证书列表，最初设置为仅包含 TxCert。
- 客户认为最后一个 TxCert 需要同步。它在 TxCert 内提取 Tx 并导出其所有输入对象（使用输入（Tx））。
- 对于每个输入对象，它检查最后生成或突变的 Tx（使用上的 Syncindex）Ct) 证书在B, 否则，从中读取其证书Ct并添加到要同步的证书列表中。$_{vvv}$
- 如果没有更多证书可以添加到列表中（因为没有更多输入丢失）B) 证书列表按因果顺序排序，并提交给B.

上述算法还适用于将对象更新到特定版本以启用新事务。在这种情况下，生成对象版本的 Tx 证书（在同步[ObjRef]中找到）将提交给滞后机构。一旦在上执行B 正确版本的对象将可用。$_{v}$

执行此操作的客户端称为中继层。可以有多个继电器同时独立运行。它们在完整性方面不受信任，并且其操作是无钥匙的。除了客户端之外，当局还可以运行中继层逻辑来相互更新，副本操作服务也可以充当中继层来更新滞后的当局。

大部分当局为追随者在处理证书时接收更新提供了便利。这允许复制副本维护授权状态的最新视图。此外，当局可以使用推拉式八卦网络在短期内相互更新最新处理的交易，并减少中继人执行此功能的需要。从长远来看，落后当局可以在时代边界或更频繁地使用定期国家承诺，以确保他们已经处理了一整套证书，直至某些检查点。

5 扩展和延迟

Sui 系统允许通过当局投入更多资源（即一台机器内或多台机器上的 CPU、内存、网络和存储）来扩展事务处理。更多的资源会提高处理交易的能力，从而增加为这些资源提供资金的费用收入。更多的资源也会降低延迟，因为操作是在不等待必要资源可用的情况下执行的。

吞吐量为了确保更多的资源以准线性方式增加容量，Sui 设计大大减少了需要权限内全局锁的瓶颈和同步点。处理事务被清晰地分为两个阶段，即（1）确保事务对特定版本的拥有或共享对象具有独占访问权限，以及（2）随后执行事务并提交其效果。

阶段（1）要求事务以对象的粒度获取分布式锁。对于拥有的对象，这是通过可靠的广播原语执行的，该原语不需要权限内的全局同步，因此可以通过 ObjID 在多台机器上共享锁管理来扩展。对于涉及共享对象的事务，需要使用共识协议进行排序，该协议确实对这些事务施加了全局顺序，并且有可能成为瓶颈。然而，工程化高吞吐量共识协议[9]的最新进展表明，顺序执行是状态

机复制的瓶颈，而不是排序。在 Sui 中，排序仅用于确定输入共享对象的版本，即增加对象版本号并将其与事务摘要关联，而不是执行顺序执行。

第（2）阶段发生在所有输入对象的版本为当局所知（并在当局之间安全同意）时，涉及移动事务的执行及其影响的承诺。一旦知道输入对象的版本，就可以完全并行执行。移动多核上的虚拟机或物理机读取版本化的输入对象，执行并将结果对象从存储区写入存储区。对象和事务的存储一致性要求（除订单锁映射外）非常宽松，允许每个机构在内部使用可扩展的分布式键值存储。执行是幂等的，使得处理执行的组件上的崩溃或硬件故障都很容易恢复。

因此，对于彼此没有因果关系的事务，可以并行执行。因此，智能合约设计者可以设计合约内对象和操作的数据模型，以利用这种并行性。

检查点和状态承诺是在关键事务处理路径之外计算的，以不阻止处理新事务。这些操作涉及对提交的数据进行读取操作，而不是在事务达到最终状态之前要求进行计算和协商。因此，它们不会影响处理新事务的延迟或吞吐量，并且本身可以分布在可用资源中。

读取可以受益于非常积极且可扩展的缓存。当局签署并提供 light 客户端读取所需的所有数据，这些数据可以作为静态数据由分布式存储提供。证书在其交易和对象的完整因果历史中充当信任的根源。国家承诺进一步允许整个系统对所有国家和处理的交易具有定期的全球信任根，至少在每个时代或更频繁。

延迟智能合约设计人员可以灵活地控制他们定义的操作的延迟，这取决于他们涉及的是拥有的还是共享的对象。拥有的对象在执行和提交之前依赖可靠的广播，这需要两次往返到法定人数的机构才能达到最终结果。另一方面，涉及共享对象的操作需要一致的广播来创建证书，然后在共识协议内进行处理，从而导致延迟增加（截至[9]为止，4 到 8 次往返到 quorum）。

参考文献

[1] 穆斯塔法·巴萨姆、阿尔贝托·索尼诺、维塔利克·巴特林和伊斯梅尔·霍菲。2021。欺诈和数据可用性证明：检测轻客户端中的无效块。金融加密和数据安全-第 25 届国际会议，2021 金融委员会，虚拟事件，2021 3 月 1 日至 5 日，修订论文选集，第二部分（计算机科学讲稿，第 12675 卷），Nikita Borisov 和 Claudia Diaz（编辑）。斯普林格，279-298。

[2] Shehar Bano、Alberto Sonnino、Mustafa Al-Bassam、Sarah Azouvi、Patrick McCorry、Sarah Meiklejohn 和 George Danezis。2019 年，《SoK：区块链时代的共识》。在 2019 年 10 月 21 日至 23 日于瑞士苏黎世举行的金融技术进步第一届 ACM 会议记录中。ACM，183–198。

[3] 马修波德特、乔治·达内齐斯和阿尔贝托·索尼诺。2020 年。FastPay：高性能拜占庭容错解决方案。在 AFT'20:2020 年 10 月 21 日至 23 日于美国纽约举行的第二届 ACM 金融技术进步会议上。ACM，163–177。

[4] Sam Blackshear、Evan Cheng、David L.Dill、Victor Gao、Ben Maurer、Todd Nowacki、Alistair Pott、Shaz Qadeer、Ra in、Dario Russi、Stephane Sezer、Tim Zakian 和 Runtian Zhou。Move：一种具有可编程资源的语言。https://developers.libra.org/docs/move-paper.

[5] Sam Blackshear、David L.Dill、Shaz Qadeer、Clark W.Barrett、John C.Mitchell、，Oded Padon 和 Yoni Zohar。资源：一种安全的语言抽象钱 CoRR abs/2004.05106（2020）。arXiv:2004.05106https://arxiv.org/abs/2004.05106

[6] 克里斯蒂安·卡钦、拉希德·格雷奥伊和路易斯·罗德里格斯。2011 年，《可靠和安全分布式编程简介》。施普林格科学与商业媒体。

[7] Panagiotis chatzigianis、Foteini Baldimtsi 和 Konstantinos Chalkias。2021. SoK：区块链轻客户端。IACR 密码。ePrint 拱门。(2021), 1657.

[8] Daniel Collins、Rachid Guerraoui、Jovan Komatovic、Petr Kuznetsov、Matteo Monti、Matej Pavlovic、Yvonne Anne Pignolet、Dragos Adrian Seredinschi、Andrei Tonkikh 和 Athanasios Xygkis。2020 年。仅通过广播信息进行在线支付。第 50 届 IEEE/IFIP 可靠系统和网络国际会议，DSN 2020，西班牙巴伦西亚，2020 年 6 月 29 日至 7 月 2 日。IEEE，26–38。

[9] 乔治·达内齐斯、Eleftherios Kokoris Kogias、阿尔伯托·索尼诺和亚历山大·斯皮格曼。2021，《独角鲸和图斯克：基于 DAG 的内存池和高效 BFT 共识》。CoRR abs/2105.11827（2021）。

[10] 大卫·L·迪尔、沃尔夫冈·格里斯坎普、容克公园、沙兹·卡迪尔、孟旭和静宜·艾玛·钟。2021。使用 Move Prover 快速可靠地验证智能合约。更正 abs/2110.08362（2021）。arXiv:2110.08362https://arxiv.org/abs/2110.08362

[11] Rachid Guerraoui、Petr Kuznetsov、Matteo Monti、Matej Pavlovic 和 DragosAdrian Seredinschi。2018.AT2：异步可信传输。CoRR abs/1812.10844（2018）。

[12] Rachid Guerraoui、Petr Kuznetsov、Matteo Monti、Matej Pavlovic 和 Dragos-Adrian Seredinschi。2019 年。加密货币的共识编号。Peter Robinson 和 Faith Ellen（编辑）于 2019 年 7 月 29 日至 8 月 2 日在加拿大安大略省多伦多市举行的 2019 年 ACM 分布式计算原理研讨会上发表了论文集。ACM，307-316。

[13] Patrick McCorry、Chris Buckland、Bennet Yee 和 Dawn Song。2021。SoK：验证桥梁作为区块链的扩展解决方案。IACR 密码。ePrint 拱门。(2021), 1589.

[14] 马可·帕特里格纳尼和萨姆·布莱克希尔。2021。移动安全可靠。更正 abs/2110.05043（2021）。arXiv:2110.05043https://arxiv.org/abs/2110.05043

[15] Jerome H Saltzer 和 Michael D Schroeder。1975.计算机系统中的信息保护。过程。IEEE 63，9（1975），1278-1308。

[16] 静宜：钟爱玛、张凯文、沙兹·卡迪尔、沃尔夫冈·格里斯坎普、山姆·布莱克希尔、容克公园、尤尼·佐哈尔、克拉克·W·巴雷特和大卫·L·迪尔。2020 移动验证程序。在第 32 届计算机辅助验证国际会议上，*CAV 2020，美国加利福尼亚州洛杉矶，2020 年 7 月 21 日至 24 日，会议记录，第一部分（计算机科学讲稿，第 12224 卷）*，Shuvendu K.Lahiri 和 Chao Wang（编辑）。斯普林格，137–150。https://doi.org/10.1007/978-3-030-53288-8_7

[1]https://diem.github.io/move/

[2]https://github.com/MystenLabs/fastnft/blob/main/doc/SUMMARY.md
[3]https://diem.github.io/move/abilities.html
[4]https://github.com/ethereum/EIPs/blob/master/EIPS/eip-1559.md
[5] 请注意，如果签名算法允许，可以压缩权限签名，但始终使用可靠的签名聚合，因为跟踪签名者对于天然气利润分配和其他网络健康测量非常重要。