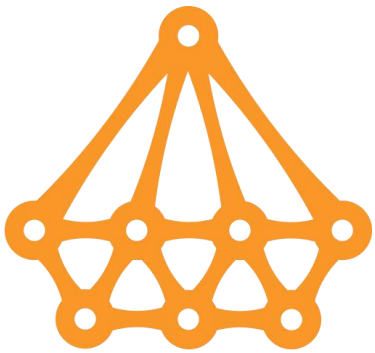


NERTHUS 白皮书

一个基于 DAG 技术通用的区块链编程平台

摘要

Nerthus 与以太坊一样，致力于打造一个通用的智能合约编程平台与区块链操作系统。Nerthus 有自己的图灵完备的编程语言 witstone，以及运行环境 OVM。与以太坊不同的是，我们在底层使用 DAG 技术，有效地解决了传统区块链系统面临的低吞吐量，交易确认延时，区块膨胀等区块链式结构先天性的悖论问题。

龚剑锋

目录

什么是 Nerthus?

Nerthus 是一个基于 DAG 技术的通用的区块链编程平台，一个去中心化分布式区块链操作系统。Nerthus 内置图灵完备的编程语言，用户可以用之来建构和定义他自己的各种特性，可以开发自己的应用与区块链系统，可以发行自己的货币。

有了以太坊为什么还要有 Nerthus?

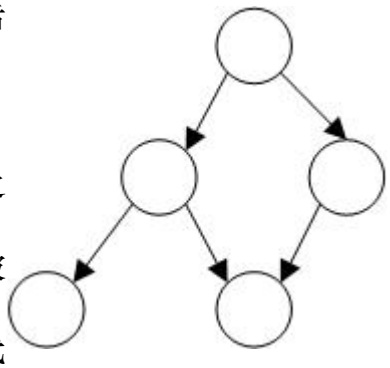
包括比特币、以太坊在内的传统的区块链，底层采用的是区块+链式结构。这种结构设计会带来区块数据膨胀，交易延时，吞吐量低等先天性缺陷。比如著名的比特币扩容之争，双方的争议点在于是否进行区块扩容。支持扩容的一方认为，在交易越来越多的情况下，很多交易无法及时打包进区块，会造成大量交易长时间延时，为解决该问题需要进行扩容。而不支持扩容的一方则认为，扩容之后区块变大，将会使区块数据迅速膨胀，以至于普通的个人电脑难以储存。这就是传统区块链先天性缺陷的体现，而且还是悖论式的。Nerthus 采用的是 DAG（有向无环图）结构。交易无需矿工打包，自己创建自己发布，不存在交易延时的问题，也不存在打包矿工机器的性能限制，因此 Nerthus 不存在吞吐量的问题。在 Nerthus 系统里，每个用户只维护自己单元的所有父辈单元和储存、交易对手方所在父辈单元的数据，无需存储全网所有数据。即使全网所有数据量非常庞大，对单个用户也没有影响。

	吞吐量	交易延时	数据膨胀
Nerthus	理论无限制	除网络延时	即使全网数据非常庞大，对单个用户也影响不大
以太坊	理论 2000TPS	15 秒出块，	2017 年底预计 1T

技术篇

背景

DAG，英文全称是 Directed Acyclic Graph（有向无环图）。在图论中，如果一个有向图无法从某个顶点出发经过若干条边回到该点，这个图则叫有向无环图。下图便是一个典型的有向无环图。圆代表顶点，线叫边，代表顶点与顶点之间的关系。

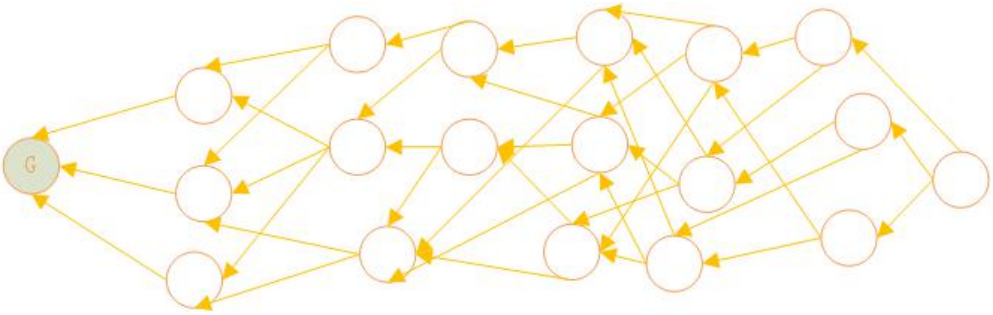


DAG 结构由 IOTA 团队率先使用，之后 Byteball 借鉴 IOTA 的 DAG 结构，并加以改进。在 IOTA 中，要验证新的交易前，必须直接验证之前的两个交易，这也使得在这两个交易之前所有被验证过的交易得到间接验证。在 DAG 中，顶点代表交易，带箭头的线代表交易的验证关系。在 IOTA 中，有一个权重积分的概念，所谓权重积分是指它自身的权重与它验证过的所有交易的自身权重之和。在 DAG 结构中，交易总是自己创建并发布。从理论上看，攻击者总是可以建构比

它要推翻掉的那个交易权重更高的交易用以双花。Byteball 在 IOTA 的基础上，做了改进，引入主链与见证人概念，并鼓励验证多个父辈交易单元。但是它在每个交易单元都有一个见证人列表，除了会造成单元数据变大之外，还会在恶意攻击者尝试双花时，故意发布不同见证人列表的双花单元引起混乱。同时他的确认机制是沿着一条 MC 前行的，一路遇到多少见证人，这都将增加交易证验的复杂度与不确定性。虽然交易最终会达到一个稳定的确定性，但是交易确认的时间是不确定的。NERTHUS 在 Byteball 的基础上，做了进一步的改进——维护用户级别的见证人列表。并受 DPOS 机制的启发，交易单元一旦发布且经所有见证人共同签署的见证单元验证后，该交易单元就是最终确认的。具体情况，将在后面详细介绍。

单元

Nerthus 底层数据结构采用的不是像比特币、以太坊那种传统的链式结构模式。而是采用一种 DAG（有向无环图）结构模式。



图一

上图是 DAG 的结构模型，圆表示顶点，线条表示顶点与顶点的关系，箭头表示从子单元到父辈单元的方向，G 是创世单元，每个单元，通过其箭头所指的

父辈单元，一直追溯，可达创世单元。对应于 Nerthus，圆表示一个单元。Nerthus 中的单元包含引用之前一个和多个单元作为其的父辈单元，以此建立单元次序。单元可以包含多条不同类型的数据，如支付，文本消息，智能合约等等。

DAG 中的每个新单元，验证并确认其父辈单元，父辈单元的父辈单元，可达创世单元，并将其父辈单元的哈希包含到自己的单元里面。如果有人篡改数据，其单元的哈希必将改变，那就会使得它与直接或间接验证确认它的子单元中引用它的哈希不一致。如果要成功篡改单元数据，需要与它的所有的子单元合作，子单元修改它引用的 Hash,这又会导致子单元的 Hash 发生改变，那么子单元又要与子单元的所有子单元合作，直到最后的子单元。所以，如果一个单元被广播到网络中，被其它单元验证确认并建构在其上后，篡改数据需要协调的人数便会呈指数级增长。传统的单线链式结构，理论上它们要修改数据，只需和后面几个区块达成一致，就可以达到篡改数据的目的（51%攻击，通过算力快速产生几个区块，这些区块都是自己控制的，相互合作，就可以篡改数据了）。DAG 这个模式相比来说，篡改数据的复杂度更高，更难以篡改。

双花与地址顺序单元系列链

在去中心化系统里面，有效防止双花，是必要条件，更是基础。如果不能有效防止双花，整个系统就不成立了。DAG 通过下面协议规则解决双花问题。

1、 一个单元不能引用它的其它父单元直接或间接引用过的单元做父单元。

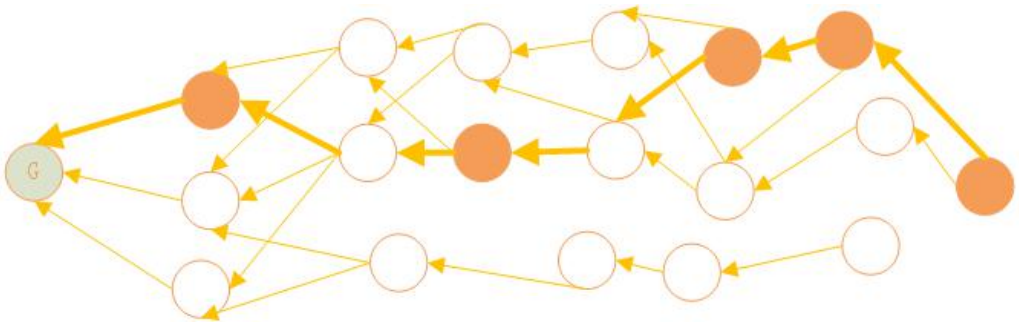
2、 一个地址如果创建发布超过一个单元，后发布的单元必须直接或间接

地包含引用其之前发布的所有单元，形成这个地址的顺序单元系列。

3、 如果一个地址发布的单元，违反规则二，发布一个或多个，没有顺序引用关系的单元或单元系列，都会视为双花，不论是否存在实质性双花行为。

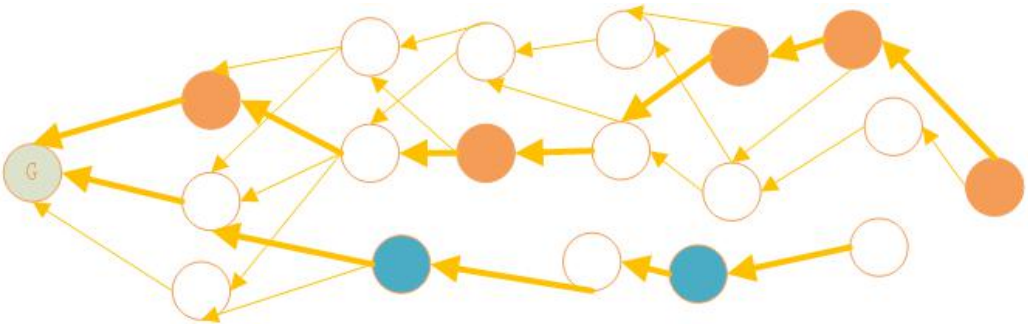
4、 在遵守规则二的情况下，出现双花问题，顺序单元系列里，发布较早的有效，发布晚的无效。如果不遵守规则二，发布多个非顺序引用关系的单元或单元系列，根据最优顺序单元系列算法，只有一个单元或顺序单元系列有效，其余单元或顺序单元系列无效。

5、 如果一个地址的单元间接或直接包含引用两个或以上的自己发布的没有顺序的单元，该单元无效，不论是否存在实质性双花行为。



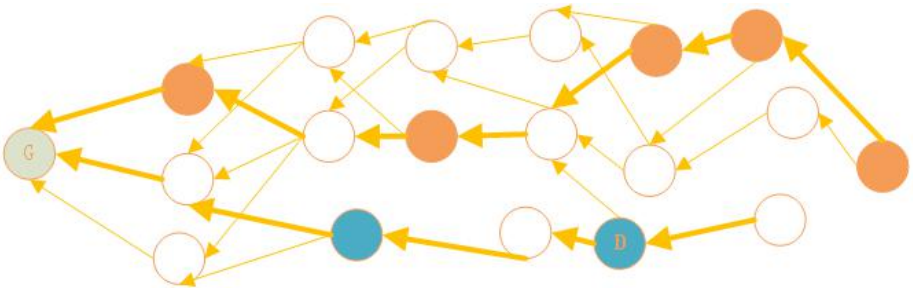
图二

图二中，橙色实心圆点代表了是同一个地址所发布的所有单元。后面的直接或间接地包含前面的单元，形成一个有序的单元序列。



图三

图三中，橙色实心圆点与蓝色实心圆点都是同一个地址所发布的单元。可以看出蓝色实心圆点与橙色实心圆点之间没有顺序包含引用关系。这种情况下，只有一个顺序单元系列被承认。其它顺序单元的所有交易都会被视为无效的。



图四

图四中，橙色实心圆点与蓝色实心圆点都是同一个地址所发布的单元。蓝色单元 D，间接包含引用了没有顺序关系的橙色单元与蓝色单元。根据规则四，因此，单元 D 是不被承认的无效单元。建构在 D 之上的由该地址发布的后续单元，由于 D 引用了没有顺序关系的多个同地址单元，所以 D 的后续单元也间接地引用了它们，因而也都是无效的。

在图二、图三与图四中，可以看到，从右到左，从一个单元出发，用粗箭头线指向它的一个父单元，它的这个父单元又用粗箭头线指向这个父单元的一个父单元，直到创世块，形成了一个链条，我们称其为地址顺序单元系列链。

地址顺序单元系列链是指，从一个子单元开始，选出其最佳父单元，其最佳父单元又选出这个父单元的最佳父单元，直至创世块。最佳父单元由单元高度、包含的见证单元数，时间戳，包含该子单元的顺序单元数量及路径，共同计算一个权重而获得。

根据规则五，两条开始相互独立，而在后面的某一点上又相互交叉的地址顺序单元系列，我们简化了其引起的不必要的复杂性。因而，如果出现两条或多条地址顺序单元系列链（为了简化表述，我们把同一地址发布的没有包含引用关系的相互独立单元，即使只有一个独立的单元构不成顺序单元系列，称为地址顺序单元系列链），根据规则四，相互独立的地址顺序单元序列，只有一条被认为有效。用下面原则，选定一条地址顺序单元系列链为有效，其余的皆为无效而不被承认。

1、 如果出发点是无效的，那么其后续的所有都是无效的。在同一地址的多个不同地址顺序单元系列中，如果在这些地址顺序单元系列链中，第一个地址单元无效，那么这个地址顺序单元系列链都无效。

2、 根据规则一，我们把问题简化为，当同一地址出现多个不同的地址顺序单元系列时，只需比较这些顺序单元系列中的第一个地址单元，并选择出一个最优的地址单元，那么这个最优地址单元所在的地址顺序单元系列链就是有效的，其余的都是无效的。

见证人

单个地址顺序单元中,每个单元都会有先后次序关系。当出现双花问题时，根据先后次序关系，判定早的单元有效，后的单元无效。这就很容易就解决了双花问题。但是，如果攻击者发布故意多个没有顺序关系的单元或单元系列，那么情况就变得复杂了。因为在 DAG 中，单元是可以自己创建并发布的，他可以选

择自己的父单元，可选择自己的高度，伪造时间戳，他可以根据规则伪造出比他想要推翻的单元权重更高的单元，可以创建更多的单元来确认这个双花单元，用来进行双花。而且他可能会创建一个隐藏的单元系列链，然后在某一时刻公布这个单元系列链，把之前所有的单元全都推翻，更致命的是，你不知道他会在什么时候公布这条隐藏的链。为此，我们引入了见证人机制来解决这个问题。

见证人机制是受传统区块链的委托权益证明（DPOS）启发而设计的。见证人通过选举产生，用户可以提交竞选，并缴纳一笔保证金后，就成为候选见证人。系统根据候选见证人的得票数，保证金，声誉以及是否有实名信息，以及平时表现等，计算出一个指数，并根据指数，为其分配它的见证用户。作为报酬，他会与其被见证用户的所有其它见证人分享被见证用户的所有单元的交易费，但见证者的报酬，会冻结三个月，三个月后方可领取。如果见证人长时间不履行其职责，不发布见证单元，将会被取消见证人资格，并在一定时间内不得参选见证人。如果见证人不遵守见证规则，发布无效恶意见证区块，将会被没收保证金，并永远不得参选见证人。

每个用户都有其见证人列表，这个见证人列表的数量为奇数。在用户创建第一个单元之前，系统会为它分配一个见证人列表，并公布到网络。网络会维护每一个用户的见证人列表。为了防止用户与见证人串通，用户无法选择见证人，而见证人也无法选择用户，并且会每间隔一定的时间，更新一次见证人列表，全网也会记录所有的更新，并记录见证人成为该用户见证人的起止时间。

用户发布单元后，其见证人列表中的见证人将给单元进行见证确认，其过程如下：

1、 用户发布单元到网络（同时也直接发送给他的见证人）。

2、 见证人接收到单元，并与该被见证人的其他所有见证人通信，以确定所有见证人收到的单元是一致的。这是必要的步骤。如果一个恶意的用户，想攻击网络，他可能会给不同的见证人发送不同的无序单元。如果见证人之间不进行通信确认的话，每个见证人将会对不同的无序单元进行确认，这样系统将会进入混乱状态。举个极端例子，某用户拥有一个 21 人的见证人列表，在还没形成单元系列时，他同时发布出 21 个不同的无顺序关系的单元，并分别给每个见证人发送一个不相同的单元。在见证人之间不进行相互通信的情况下，都对收到的单元进行确认验证。那么就有 21 个独立的单元同时有效，这肯定是不可接受的。

3、 如果所有在场见证人收到的单元是一致的，经由所有在场见证人校验，如果没有错误，则所有在场见证人共同签署见证单元，并发布到网络。在场见证人一定要超过该用户见证人列表里所有见证人的 50%。用户的见证人列表有 21 个见证人，那么需要的在场见证人至少要有 11 个。之所以有在场见证人这个概念是因为，可能存在网络问题、有时有些见证人离线，或者有些见证人可能有恶意，留有 49%的冗余，以确保系统的健壮。

4、 如果所有在场见证人获得的用户单元是不一致的，他们将相互交换单元，并检查这些单元是否存在有效包含关系，如果存在有效包含关系，以最后的子单元为准。如果单元中在恶意含包关系（恶意包含关系指单元直接或间接包含引用了自己两个非序顺单元系列），则此单元被抛弃。如果多个单元间是相互独立，没有顺序关系的，所有在场见证

人根据最小哈希原则选择其中一个单元作为有效单元，共同签署见证单元，发布到网络。

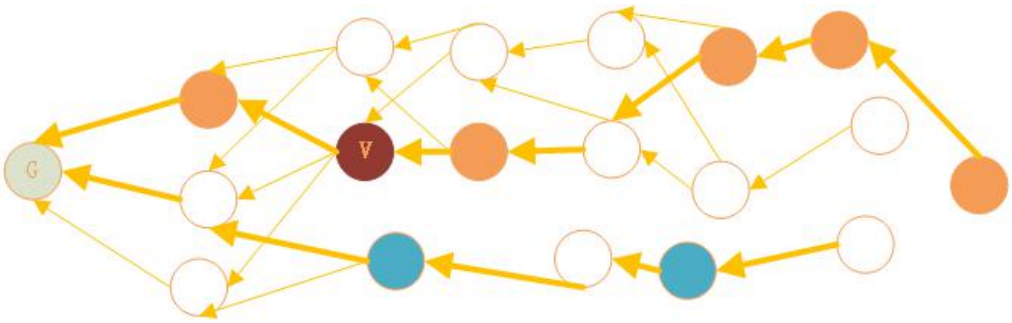
5、 见证单元一经发布，该用户被验证的单元及其引用自身的前辈单元皆被确认，并具有最终性。

见证规则：

1、 在同一分叉点，同一验证者只能对其中的一个单元或单元系列投票，来作有效验证。

2、 如果用户单元或单元系列已经存在了验证单元，其后的验证单元只能在这条已有的验证单元的地址顺序单元系列链上进行，并且验证单元间也要建立次序关系。

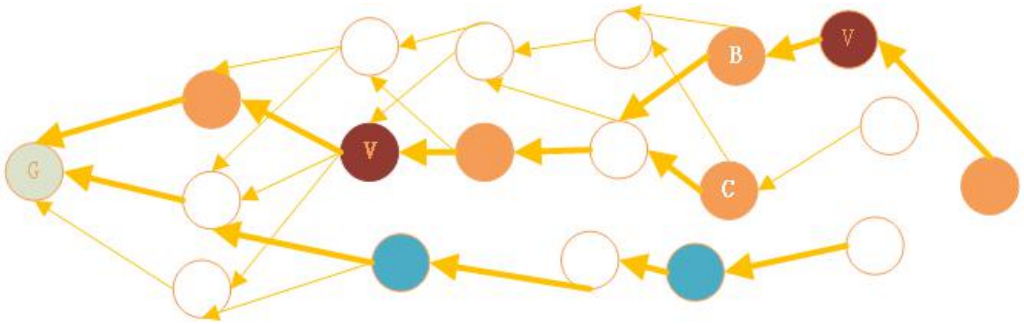
3、 如果见证者违反上规则一和二，则其保证金及冻结的三个月交易手续费都将被没收，并且将永远不能成为见证人。



图五

图五中，橙色与蓝色是同一地址发布的两个独立的顺序单元序列，在它们的第一个单元之前，都没有自己的单元，那么我们把没有包含自己任何单元的单元称为 0 点单元。在这里分叉我们称为 0 点分叉。紫色的 V 单元是见证人签署的

验证单元，它确认的是橙色地址顺序单元系列。



图六

图六中，橙色地址序顺单元系列在第一个验证单元后出现了 B 与 C 两个分叉单元，验证单元选择了 B，那么 B 就是有效的，C 是无效的。我们以橙色单元为出发点，排除与地址无关的节点单元计数，B 与 C 处于第四层的位置。我们称为 4 分叉点。

在 0 分叉点上，见证单元支持了橙色，在 4 分叉点上它支持了 B，根据见证规则一，在 0 分叉点上与 4 分叉点上任何现有或后来可能伪造的其它地址顺序单元系列或单元都是无效的。根据见证规则二，如果用户单元或单元系列已经存在了验证单元，其后的验证单元只能在这条已有验证单元的地址顺序单元系列链上进行，并且验证单元间也要建立次序关系。那就可以确保任何用户地址，都只有一条明确清晰的被认可的顺序单元系列。

最终性

比特币与以太坊，有一个最大的问题就是没有一个确定的不可更改的最终状态。理论是，如果有足够的算力，足够的出块速度，产生一条更长的隐藏链，就可以把之前的区块推翻。Nerthus 单元经过见证人发布见证区块后，就已是最

终确定的状态，无法推翻。

确认时间

确认时间取决于见证者发布见证区块的时间。我们把见证者发布见证区块的时间分为加急，急，快，普通，慢五个等级，每个等级需要支付的验证费用不同，以平衡见证者的负载及被见证者的时间需求。最快的加急，在不考虑网络状况的情况下，一秒内完成。

NERTHUS 的五大突破

1、 更彻底去中心化

传统的区块+链式结构，需要有一个类中心化的操作，即需要一个记账人，将当前所有交易进行验证处理,然后打包到一个区块,再发布到网络。而 Nerthus 系统，如上所述，采用的是单元+DAG 结构，没有区块这一概念。所有单元由用户自己创建与发布。其验证与确认由引用其作为先辈单元的后辈单元来承担。无需传统区块+链式结构那样，需要一个记账人，将当前所有交易打包到区块这一中心化的操作，因而是一种更彻底的去中心化系统。

2、 无吞吐量瓶颈

因为传统区块+链式结构存在着中心化的操作过程，即需要记账人将交易打包到区块。那么区块链系统处理交易能力的大小，必定受制于以下三点：（1）记账人节点机器的性能；（2）记账人节点的网络带宽；（3）区块的大小。因为

存在这一中心化色彩的操作，无论怎样优化，始终都会存在着一个处理能力的瓶颈点。如上所述，Nerthus 系统，采用的是单元+DAG 结构，没有记账人打包区块这一中心化的操作，因此也不存在区块这一概念。单元由用户创建发布，并由其它单元验证确认，因而不存在吞吐量瓶颈。

3、 无区块扩容与数据膨胀的悖论性两难

传统区块+链式结构需要所有交易要打包到区块才有效。若区块的容量设置小，当交易量大时，很多交易无法及时打包进区块。若区块容量设置大，则会使区块链数据迅速膨胀，普通个人电脑无法运行全节点，只有少数人才可能运行全节点，会造成中心化的结果。这也是比特币扩容之争的根本矛盾点。如上所述，Nerthus 没有区块这一概念，所以对于传统区块+链式结构先天性的悖论两难问题，在 Nerthus 中就根本不存在。

4、 明确可预期的最终性

传统区块+链式结构，不排除可能同时产生两个甚至多个区块，由此导致分叉。对于出现分叉的情况，传统区块链将以最长链作为有效链。该机制在理论上会将无法确定最终性，因为无法保证是否存在一条隐藏长链。而 nerthus 通过见证人机制，只要通过见证人发布的见证单元验证确认，即具最终性，无法推翻。

5、 可选交易确认速度

见证人发布见证区块分为加急、急、快、普通、慢五个等级。用户可根据自身需求，选择交易确认速度。

NERTHUS 的生态系统与三层架构

Nerthus 是一个通用的智能合约平台与区块链操作系统。同时也致力于打造一个基于 Nerthus 的生态系统。

Nerthus 在底层采用单元+DAG 结构，无需记账者打包，打造了一个更为彻底的去中心化区块链系统。一个没有吞吐量瓶颈限制的区块链系统，是 Nerthus 的核心部分，也是 Nerthus 提供的基础设施。在整个 Nerthus 系统中，它处于底层，是基础层。

在 Nerthus 基础层之上，我们还引入了服务层，以供基于 Nerthus 开发的开发者快速开发各种应用。在服务层中，我们除了封装好核心层的各种 API 之外，还提供了区块链翻译系统与侧链系统。区块链翻译系统主要是指，使两个相互独立的区块链，能够相互读懂对方，能够无障碍通迅。侧链系统是指企业用户，可以快速生成基于 Nerthus 的私有链和联盟链，并链结在 Nerthus 主链上，可以利用 Nerthus 翻译系统与其它链相互通信与交易。建立一个良好的生态系统，除了拥有先进性技术之外，还应该能够整合各种资源。区块链翻译系统，主要目的在于整合现有的各种区块链资源，让现有的各个区块链项目，通过 Nerthus 能够相互通讯，并且能够相互交互，成为 Nerthus 生态圈的一环。而基于 Nerthus 的应用开发者，能开发出跨越各种链的应用。而侧链系统则为建立联盟链与私有链的企业与机构，提供一个快速、低成本的方式。虽然这些企业与机构因为各种原因，只能建立联盟链或私有链，但他们依然有与其它公有链用户的通讯交易需求，Nerthus 的侧链系统正好为它们提供这样的功能接口。

在服务层之上，是 Nerthus 的应用层。Nerthus 应用层是指，基于 Nerthus

上开发的各种区块链应用，这些应用主要由第三方开发者开发。目前，区块链的各种应用，最典型的比如钱包，用户体验还不是很好。区块链开发者更多的还是注重于功能的实现，对于用户体验关注度不是很高。一项技术，一个系统，一项应用，如果想要大家都使用，用户体验是非常重要的一环。Nerthus 非常关注用户体验，借鉴 IOS 的经验，将建立一套 Nerthus 应用层的规范与标准。

Nerthus 核心基础层是 Nerthus 生态系统的根基。Nerthus 上的开发者以及基于各区块链系统，是 Nerthus 生态中的上游参与者，我们在服务层为他们提供各种便利的服务接口。普通用户则是 Nerthus 生态中的消费者，我们制定一套用户友好的应用层规范标准，方便普通用户体验。

Wintrone 与 OVM

智能合约编程语言与运行环境是实现可编程的区块链操作系统的基础设施。OVM 是专门为 Nerthus 区块链系统研发的虚拟机，为 Nerthus 智能合约提供安全可靠高性能的运行环境。

同时，我们还提供了一门编写 Nerthus 智能合约的语言——witstone。Witstone 是一种类似于 javascript 的脚本语言，从语言层面上加入了对 Nerthus 一些特性的支持。Witsone 是一门简单易用、易上手的智能合约编程语言，开发者无需多大学习成本，在短时间内就能编写出 Nerthus 智能合约。

NERTHUS 代币

NERTHUS 将提供 NERTHUS 代币。NERTHUS 代币是维护 NERTHUS 系统的血液与燃料。用户创建发布单元需要支付给见证人见证费，运行智能合约需要支付计步费。

代币分配

NERTHUS 代币总共发行 100 亿。

ICO（60%)

团队(30%)

NERTHUS 基金（10%)

总结

Nerthus 系统与以太坊一样，致力于打造一个通用的智能合约编程平台与区块链操作系统。Nerthus 有自己的图灵完备的编程语言 witstone，以及运行环境 OVM。与以太坊不同的是，我们在底层使用 DAG 技术，有效地解决了传统区块链系统面临的低吞吐量，交易确认延时，区块膨胀等区块链式结构先天性的悖论问题。在智能合约方面，为了使其更具实用性，Nerthus 系统设计了一套链外数据验证确认共识的机制，使之前智能合约难以实现的场景，变成了可由智能合约实现的领域。Nerthus 将推动智能合约在现实世界的普及应用。

参考文献

[1] Satoshi Nakamoto. Bitcoin: A Peer-to-Peer Electronic Cash System.

[2] Vitalik Buterin. Ethereum White Paper : A Next-Generation Smart Contract and Decentralized Application Platform.

[3]https://github.com/ethereum/wiki/wiki/White-Paper.

[4] Serguei Popov for Jinn Labs. The tangle.https://iota.org/IOTA_Whitepaper.pdf, April 2016.

[5] Anton Churyumov. Byteball: A Decentralized System for Storage and Transfer of Value.

[6] SergioDemianLerner DagCoin, https://bitslog.files.wordpress.com/2015/09/dagcoin-v41.pdf, 2015.

[7] Delegated Proof of Stake.

http://docs.bitshares.org/bitshares/dpos.html

[8] Nxt, 2013, http://wiki.nxtcrypto.org/wiki/Whitepaper:Nxt

[9] Smart contracts: https://en.bitcoin.it/wiki/Contracts