什么是以太坊

以太坊 经常被称为世界计算机.” 但这意味着什么呢? 让我们从一段突出计算机科学的说明开始, 然后再尝试对以太坊的功能和特征做实用分析来解密它, 同时将它与比特币或其他去中心化的信息交互平台 (或简称为“区块链”)作对比.

从计算机科学的角度来看，以太坊是一种确定性的但实际上是不受限制的状态机，由全局可访问的单例状态和将更改应用于该状态的虚拟机组成。

从一个更实用的角度来看，以太坊是一个开源的、全面去中心化的，运行被称为智能合约（smart contracts）的计算基础设施。它使用区块来同步和存储系统状态改变，同时用被称为“以太”（ether）的加密货币来度量和限制运行所消耗的资源。

以太坊平台让开发者可以利用内置的经济函数来建立强大的去中心化程序。在提供高可用性、高可审计性、高透明度和高中立性的同时，它还减少或消除了审查，并降低了特定的交易对手风险。

与比特币的比较

许多人会带着加密货币（特别是比特币）的一些先入经验来认识以太坊。以太坊和其他开放区块链一样有诸多共同元素: 连接用户的点对点网络，拜占庭容错的共识算法来同步状态更新 (工作量证明的区块链)，使用诸如数字签名和哈希等加密货币的原始技术， 以及一种数字货币 (以太).

然而在许多方面，以太坊不管是目标还是构造都与之前的开放区块链截然不同，包括比特币。

成为数字货币支付网络不是以太坊的主要目标. 因为数字货币以太既是以太坊必要的内部成分，同时也是以太坊运行的必需， 以太被设计为效用货币（utility currency） 来支付以太坊作为世界计算机的操作。

不同于脚本语言十分有限的比特币，以太坊被设计为一个通用用途的可编程区块链，运行能够执行任意和不受限的复杂度的代码的虚拟机（virtual machine）。而比特币的脚本语言则被有意限制为简单地对消费条件作真/假评估。以太坊的语言是图灵完备的（Trring complete），意味着以太坊可以像一般电脑一样直接运作。

区块链的组件

一般来说，开放的公共区块链的组件包括:

- 点对点 (P2P) 网络，连接用户并且基于标准化的"gossip"通道传播交易和确认过的交易的区块:[协议]
- 消息, 以交易的形式存在，表示状态转换 *一组共识规则，用于控制构成交易以及促成有效状态转换的因素 状态机，根据共识规则处理交易 *一串经过加密保护的区块，包含所有已验证和接受状态转换的记录 共识算法，通过强制参与者在共识规则的执行上合作来达到区块链控制去中心化的目的 基于博弈论的激励机制（如工作量证明花销和区块回馈）保证状态机在公开通道上的经济性（环境） 一个或多个由以上部分构成的开源软件（“客户端”）

所有或大部分这些组件通常组合在一个软件客户端中。例如，在Bitcoin中，参考实现由_Bitcoin Core_开源项目开发，并作为_bitcoind_客户端实现。在以太坊中，有一个 *reference specification*，而不是参考实现，这是黄皮书中系统的数学描述（参见 <<references> >)。有许多客户端，它们是根据参考规范构建的。

过去，我们使用术语“区块链”来代表列出的所有组件，作为对包含上述特征的技术的集合的便捷引用。但是今天有多种多样的具备不同属性的区块链。我们需要限定词来帮助理解有疑问的区块链。比如开放（open），公开（public），全局（global），去中心化（decentralized）, 中立（neutral）和防审查（censorship-resistant），来确定一个“区块链”系统上这些组件能够构成的重要的新特征。

并非被命名为区块链的系统都是一样的。当一些人告诉你什么东西是区块链时，你并非得到最终答案；更进一步

地，你需要开始问大量的问题来弄清楚当他们使用“区块链”这个词的时候他们实际是指什么。从问上述列表中的组件的说明开始，然后问这个“区块链”是否展示了诸如开放性，公开性等特征。

以太坊的诞生

伟大的创新总是解决实际问题，以太坊也不例外。 当人们认识到比特币模型的力量，并尝试做加密货币以外的应用时，以太坊被构想出来了。但是开发者面临一个难题： 他们不是需要在比特币之上构建，就是需要开始一条新链。在比特币之上构建意味着不得不面对该网络中那些有意的限制， 并且寻找解决办法。交易类型、数据类型和数据存储大小等一系列限制看上去限制了比特币之上能够运行的应用程序；其他一切都需要额外的链下层，这马上削减了使用公共区块连的优势。对于那些保持在链上且需要更多自由和弹性的项目来说，一个新的区块链是唯一的选择。但这意味着大量的工作：自力完成所有基础设施，耗尽心神的测试等等

临近2013年底时， Vitalik Buterin，一个年轻的程序员和比特币拥护者，开始思考对比特币和万事达币（一种扩展自比特币的提供初级智慧合约的中继协议）的进一步扩展。那年10月，Vitalik给大师币团队提供了一种更通用的方法，允许弹性的和可脚本化的合约 (但并非图灵完备的)来取代大师币特殊的合约语言。尽管大师币团队印象深刻，但是这个提议是一个过大的改变，不符合他们的开发路线图。

2013年12月，Vitalik开始分享一份指明了以太坊的创意的白皮书：一个图灵完备的，一般用途的区块链。有一些人看到了这份早期草稿并且提供了反馈，帮助Vitalik推进了这个提议。

本书的两个作者都收到了这份白皮书的早期草稿，并且提了意见。Andreas M. Antonopoulos 对这个想法感兴趣并询问了 Vitalik关于使用分离区块链来增强智能合约执行的共识规则，和图灵完备语言的隐含意义。Andreas继续带着极大的兴趣跟踪以太坊的进程，不过因为他正在撰写《Mastering Bitcoid》的早期阶段，直到很久之后才直接参与以太坊。Gavin Wood博士却是接触Vitalik并帮助他C++编程技巧的第一批人士之一。Gavin成为以太坊的共同创始人、共同设计者和CTO。

正如Vitalik在他的 "Ethereum Prehistory" post所述:

> *这段时间以太坊是我个人的创造。从那之后，新的参与者开始加入。迄今为止，在协议侧最突出的是Gavin Wood。*
>
> *Gavin也有功于眼界上的微妙改变：从把以太坊看做一个构建可编程金钱的平台，附带基于区块链的合约来控制数字资产并且根据事前规则转移它们，到一个通用目的的计算平台。这从要点和术语的微妙改变开始，然后这一影响随着对Web 3 ensemble日益增大的关注而变得更强，它视以太坊为去中心化技术套件的一员，其他两个是Whisper和Swarm。*

从2013年开始，Vitalik和Gavin提炼和推进了这个想法，一起建立了后来成为以太坊的协议层。

以太坊的创始人们设想了一种没有明确目的的区块链，它可以通过变成来支持多种多样的应用程序。这个想法是通过使用一个通用目的的区块链，比如以太坊，开发者可以编写他们特定的程序，而不需要完成点对点网络的底层机制、区块链、共识算法等。以太坊平台被设计来提取这些细节并且为去中心化区块链应用程序提供一个确定性的安全编程环境。

跟中本聪很像的是，Vitalik和Gavin并非只发明了一项新技术；他们将新发明与现有技术以一种新颖的方式结合起来，并推送了原型代码向世界证明了他们的想法。

创始人们工作了数年，建立并提炼了这一前景。然后再2015年7月30号，第一个以太坊区块被挖掘出来。世界计算机开始服务世界。

NOTE

2017年9月，Vitalik Buterin发表《以太坊的早期历史》的文章，其中为以太坊的开发初期提供了有意思的第一人称视角。

你可以在下面的链接中获取更多信息 https://vitalik.ca/general/2017/09/14/prehistory.html.

===以太坊发展的四个阶段

以太坊的发展被分为四个明确的阶段，每个阶段都有重要的改变。 一个阶段可能包含多个子发布，也就是众所周知的“硬分叉”，它以一种不向后兼容的方式来改变功能。

以太坊四个主要的开发阶段分别名为_Frontier， Homestead， Metropolis_和_Serenity_。迄今为止已经发生（或计划中）的硬分叉的代号为_Ice Age， DAO， Tangerine Whistle， Spurious Dragon， Byzantium_和_Constantinople_。在以下时间轴上显示了开发阶段和中间的硬分叉，并按区块号标记为“日期”：

区块**#0**

Frontier&#x2014以太坊的初始阶段，持续时间从2015年7月30日到2016年3月。

Block#200,000

Ice Age—硬分叉版本引入指数难度增加，促使系统在适当时间向PoS过渡。

区块 **#1,150,000**

Homestead—以太坊的第二阶段，于2016年3月推出。

区块 **#1,192,000**

DAO—硬分叉，用于补偿被黑客攻击的DAO合同的受害人，并导致以太坊和以太坊经典分成两个竞争系统。

Block #2,463,000

Tangerine Whistle—硬分叉，用于更改某些I / O繁重操作的gas计算，并且当黑客利用这些操作的低gas成本来实施拒绝服务（DoS）时，可以清除系统中的累积状态，保护系统。

Block#2,675,000

Spurious Dragon— 硬分叉用于解决更多DoS攻击媒介，以及另一种状态清除方法。此外，还提供了对重放攻击的保护机制。

区块 **#4,370,000**

Metropolis Byzantium—Metropolis是以太坊的第三阶段，在撰写此书时，该阶段于2017年10月启动。Byzantium是计划用于Metropolis的两个硬叉中的第一个。

在Byzantium分叉之后，还有针对Metropolis计划的另一个硬分叉： Constantinople。在Metropolis之后，将进行以太坊部署的最后阶段，代号为Serenity。

===以太坊：通用区块链

最初的区块链（即比特币的区块链）跟踪比特币单位的状态及其所有权。你可以将比特币看作是分布式共识_状态机_，交易会导致全局状态转换，从而改变比特币的所有权。状态转换受到共识规则的约束，允许所有参与者在挖掘出足够多的区块后，可以（最终）收敛于系统的共同（共识）状态。

以太坊也是一个分布式状态机。但是，以太坊（Ethereum）不仅记录和保存货币所有权的状态，还记录通用数据的状态转换。元组结构中每个键值的数据存储区包含任意值，每个值都由某个键引用。例如，键“书名”所指向的值为“ 精通以太坊”。在某些方面，这与大多数通用计算机使用的_随机存取存储器_（RAM）数据存储模型具有相同的用途。以太坊具有存储代码和数据的内存，并使用以太坊区块链来跟踪该内存随时间的变化。像通用存储程序计算机一

样，以太坊可以将代码加载到其状态机中并运行该代码，从而将结果状态更改存储在其区块链中。与大多数通用计算机的两个关键区别是，以太坊状态变化受共识规则支配，并且状态在全球范围内分布。以太坊回答了一个问题："如果我们可以跟踪任何任意状态并对状态机进行编程以创建在共识下运行的全球计算机呢？"

===以太坊的组成部分

在以太坊中，对区块链系统组件的描述在< <blockchain_components> >中，更具体地说：

P2P网络：：以太坊运行在_Ethereum主网络_上，该主网络可在TCP端口30303上寻址，并运行称为_ÐΞvp2p_的协议。

共识规则：：以太坊的共识规则在"黄皮书"的参考规范中定义（请参见[references]）。

交易
: 以太坊交易是网络中的消息，其中包括（除其他事项外）发送者，接收者，价值和有效数据载荷。

状态机
: 以太坊状态转换由_Ethereum虚拟机_（EVM）处理，EVM是执行_二进制代码_（机器语言指令）的基于堆栈的虚拟机。 EVM程序（称为"智能合约"）以高级语言（例如Solidity）编写，并可以编译为二进制码以在EVM上执行。

数据结构
: 以太坊的状态可以作为_数据库_（通常是Google的LevelDB）存储在每个节点上，该状态在名为_Merkle Patricia Tree_默克尔树的序列化哈希数据结构中包含交易和系统状态。

共识算法：以太坊使用比特币的共识模型中本共识，该模型单个签名块按时间顺序排列后，通过工作量证明PoW对其重要性进行加权，以确定最长的链，从而确定当前状态。但是，有计划在不久的将来使用代号为_Casper_的PoS加权投票系统。

经济安全性：：以太坊当前使用一种称为_Ethash_的PoW算法，但是最终转到PoS共识。

客户端：：以太坊有几种可互操作的客户端软件实现，其中最突出的是_Go-Ethereum_（*Geth*）和_Parity_。

====深入阅读

以下参考资料提供了此处提到的技术的更加详细的信息：

*以太坊黄皮书： https://ethereum.github.io/yellowpaper/paper.pdf

- The Beige Paper（橙皮书），以比较通俗的语言重写了黄皮书，以面向更广泛的用户：https://github.com/chronaeon/beigepaper

*ÐΞvp2p网络协议： http://bit.ly/2quAlTE

*以太坊虚拟机资源列表： http://bit.ly/2PmtjiS

- LevelDB数据库（最常用于存储区块链数据的本地数据库备份）： https://github.com/google/leveldb
- 默克尔 Merkle Patricia树： https://github.com/ethereum/wiki/wiki/Patricia-Tree
- Ethash PoW算法： https://github.com/ethereum/wiki/wiki/Ethash
- Casper PoS v1实施指南： http://bit.ly/2DyPr3l

*以太坊GO语言客户端（Geth）： https://geth.ethereum.org/

*以太坊Rust语言客户端Parity： https://parity.io/

===以太坊和图灵完备性

一旦开始了解以太坊，您将立即遇到术语“ 图灵完备”。以太坊与比特币的一个主要不同，就是以太坊具备图灵完备。这到底是什么意思呢?

该术语是指被认为是计算机科学之父的英国数学家艾伦·图灵Alan Turing。 1936年，他创建了由状态机组成的计算机的数学模型，该状态机通过在顺序存储器（类似于无限长的纸带）上读写符号来操纵符号。通过这种构造，图灵继续提供了数学基础，以回答（否定的）有关“通用可计算性”的问题，即所有问题是否都可以解决。他证明了有些问题是无法解决的。 具体来说，他证明了_停机问题_（是否有可能在给定任意程序及其输入的情况下确定该程序最终是否停止运行）是无法解决的。

Alan Turing进一步定义了一个系统为_图灵完备_，如果该系统可用于模拟任何图灵机。这样的系统称为“通用图灵机”（UTM）。

以太坊能够在称为以太坊虚拟机的状态机中执行存储的程序，同时将数据读写到内存中，从而使其成为一个图灵完备的系统，从而成为一个UTM。在有限内存的限制下，以太坊可以计算任何可以在图灵机上执行的算法。

以太坊的突破性创新是将存储程序计算机的通用计算架构与分布式区块链相结合，从而创建一个分布式单状态（单例）世界计算机。以太坊程序可以“无处不在”运行，但是会产生一种通行规则所保证的共同状态：[共识]。

====作为“特性”的图灵完备

如果说以太坊是图灵完备的系统，你可能会得出这样的结论：这是_feature_，在某种程度上缺少图灵不完整的系统。相反，情况恰恰相反。图灵完备性非常容易实现；实际上，http://bit.ly/2ABft33 [已知的最简单的图灵完备状态机]具有4个状态，并使用6个符号，并且状态定义只有22条指令。确实，有时发现系统“偶然地完成了图灵完备”。可以在http://bit.ly/2Og1VgX []中找到此类系统的有趣参考。

但是，由于我们前面提到的暂停问题，图灵完备性可能会非常危险，特别是在诸如公共区块链之类的开放式访问系统中。例如，现代打印机是图灵完备的打印机，可以通过给它们提供打印文件，使打印机进入冻结状态。以太坊是图灵完备的事实意味着以太坊可以计算任何复杂程度的程序。但是这种灵活性带来了一些棘手的安全性和资源管理问题。无响应的打印机可以关闭然后重新启动。但是公共区块链是无法做到关闭和重新启动的。

====图灵完整性的含义

图灵证明你无法通过在计算机上模拟程序来预测程序是否将终止。简单来说，如果不运行程序，我们将无法预测程序的路径。完整的系统可以在“ 无限循环”中运行，该术语用于简化程序，用于描述不终止的程序。创建一个运行永远不会结束的循环的程序很简单。但是由于起始条件和代码之间的复杂交互，可能会在没有警告的情况下出现意想不到的永无止境的循环。在以太坊中，这构成了一个挑战：每个参与节点（客户端）必须验证每个交易，并运行它调用的任何智能合约。但是这里存在一个问题。依据图灵的证明，以太坊如果不运行一个智能合约，则无法预测这个智能合约是否会终止，或者它会运行多长时间（可能永远运行）。那么，智能合约可以在创建的时候，无论是偶然还是有意，使得当节点尝试对其进行验证时，就可以永远地运行下去。这实际上是一种拒绝服务攻击DoS。当然，在仅需要花费一毫秒时间进行验证的程序与永远运行的程序之间，还有无数浪费资源的程序。这些臭名昭著的程序会滥用资源，消耗内存，以及空转导致CPU过热。在世界计算机中，滥用资源的程序会滥用整个世界的资源。如果以太坊无法提前预测一个智能合约的资源使用情况，以太坊将如何限制其使用资源?

为了应对这一挑战，以太坊引入了一种称为_gas_的计量机制。当EVM执行智能合约时，它会仔细考虑每条指令（计算，数据访问等）。每条指令具有以gas为单位的预定成本。当交易触发智能合约的执行时，它必须包含一定数量的gas，该gas量设置了运行智能合约可以消耗的上限。如果计算所消耗的gas量超过交易中可用的gas量，则EVM将终止

执行。 Gas是以太坊用于允许图灵完备计算同时限制程序消耗资源的机制。

下一个问题是，“如何获得gas来支付以太坊世界计算机上的计算费用？”你不会在任何交易所找到gas。 只能作为交易的一部分购买，并且只能与ether一起购买。ether需要与交易一起发送，并且必须明确指定用于购买gas以及可接受的gas价格。就像在加油站的油泵上一样，gas的价格不是固定的。购买gas以进行交易，执行计算，并将所有未使用的gas退还给交易的发送者。

===从通用区块链到去中心化应用程序（DApps）

以太坊起初是一种制造通用区块链的方法，可以编程用于多种用途。但是很快，以太坊的愿景就扩展到了成为一个可以对DApp进行编程的平台。 DApp比智能合约具有更广阔的前景。 DApp最少包括智能合约和Web用户界面。更广泛地说，DApp是一个Web应用程序，它建立在开放的，分散的，点对点的基础架构服务之上。

一个DApp至少需要包括：

-区块链上的智能合约 -网页前端用户界面

此外，许多DApp还包括其他分布式的组件，例如：

-一套分布（P2P）存储协议和平台 -分布式（P2P）通讯协议和平台

> 您可能会看到DApp拼写为_ÐApps_。 Ð字符是拉丁字符，称为“ ETH”，暗指以太坊。要显示此字符，请使用Unicode代码点+0xD0+，或者在必要时使用HTML字符实体+eth+（或十进制实体 #208）。

===互联网的第三纪元

2004年，术语“ Web 2.0”开始流行，描述了网络向用户生成的内容，响应式界面和交互性的发展。 Web 2.0不是技术规范，而是描述Web pass的新焦点的术语：[应用程序]。

DApps的概念旨在将互联网带入下一个进化阶段，将点对点协议的去中心化引入网页应用程序的各个方面。用于描述这种演变的术语是_web3_，表示网络的第三个“版本”。 首先由Gavin Wood博士提出，web3代表了Web应用程序的新愿景和新焦点：从集中拥有和管理的应用程序到基于分布协议构建的应用程序。

在后面的章节中，我们将介绍以太坊web3.js JavaScript软件库，该软件库将在用户的网页浏览器中运行的JavaScript应用程序与以太坊区块链联系起来。 web3.js软件库还包括一个称为_Swarm_的P2P存储网络的接口和一个名为_Whisper_的P2P消息服务。通过网页浏览器中运行的JavaScript库中包含的这三个组件，开发人员可以使用完整的应用程序开发组件来构建基于web3 的DApp。

===以太坊的发展文化

到目前为止，我们已经讨论了以太坊的目标和技术与之前的其他区块链，比如比特币，有何不同之处。以太坊也有非常不同的社区发展文化。

在比特币中，开发遵循保守的原则：认真研究所有更改，以确保不会破坏现有系统。在大多数情况下，更改只有在向后兼容的情况下才能实现。现有客户端可以选择加入，但是如果他们决定不升级，也将继续运行。

在以太坊中，相比之下，社区的发展文化侧重于未来而不是过去。 社区的口头禅（并不完全是认真的）是“快速行动，打破常规”。如果需要做出更改，则尽快将其实施，即使这意味着使先前的假设无效，破坏兼容性或强迫客户端进行更新。以太坊的发展文化的特点是快速创新，快速发展，并且愿意部署前瞻性改进，即使这样做是以牺牲一些向后兼容性为代价的。

对于开发人员来说，这意味着你必须保持灵活性，并准备在系统的某些基本假设发生变化时重建基础架构。以太坊开发人员面临的最大挑战之一是将代码部署到不可变系统和仍在发展的开发平台之间的内在矛盾。你不能简单地“升级”你已经部署的智能合约。你必须准备部署新的，迁移用户，应用程序和资金，然后重新开始。

具有讽刺意味的是，这还意味着构建具有更多自主性和较少集中控制系统的目标仍未完全实现。未来几年，以太坊的稳定性可能无法达到当前自主性和分布性的要求。为了“适应”平台，你必须准备废弃并重新开发智能合约，这意味着你必须对这些智能合约保持一定程度的控制。

但是，从积极的方面来说，以太坊正在快速发展。 类似“建不建自行车棚”的问题很少，这意味着通过争论一些细微的问题（例如如何在核电站后方建造自行车棚）来阻止发展。如果你只是关注建不建自行车棚这些细节，你可能会突然发现，当你分心时，开发团队的其他人已经更改了计划，比如放弃了自行车，转而使用自动气垫船。

最终，以太坊平台的开发将放缓，其接口将变得稳定。但与此同时，创新是驱动力。你最好保持同步，因为没人会慢下来等你。

===为什么要学习以太坊?

区块链具有非常陡峭的学习曲线，因为它们将多个学科结合在一起：编程，信息安全，密码学，经济学，分布式系统，对等网络等。但是，在一个看似简单的环境的表面之下，还有更多的东西。当您学习并开始更深入地学习时，总会有另一层复杂性和奇迹。

以太坊是一个学习区块链的好平台，它正在建立一个庞大的开发者社区，比其他任何区块链平台都要快。以太坊比其他任何东西都更重要，它是开发人员为开发人员构建的_开发人员区块链_。熟悉JavaScript应用程序的开发人员可以加入以太坊，并开始快速开发可运行的代码。在以太坊创建的头几年，很常见的情况是看到一些T恤衫上印着“仅用五行代码就可以创建自己的货币”。当然，这是一把双刃剑。编写代码很容易，但是很难编写_运行良好_和_安全的_代码。

===这本书将教导你什么

本书深入探讨了以太坊，并研究了它的每个组成部分。您将从一个简单的交易开始，剖析它的工作原理，创建一个简单的智能合约，了解如何使其变得更好，并跟踪它在以太坊系统上的整个执行过程。

您不仅会学习以太坊&#x2014，它的使用方法，而且还会了解为什么以这种方式设计以太坊。您将能够理解每个部分的工作原理，以及它们如何组合在一起以及为什么。range="endofrange", startref="ix_01what-is-asciidoc0")

Last updated 2022-01-10 16:21:06 +0800