AlienChain　　中文名：外星链

目录

观点

AlienChain 是一个去中心化开源区块链平台，类似于以太坊，可供开发者通过编写智能合约开发应用程序服务于用户。AlienChain 是一款全民共建共享的区块链开发应用系统， 致力于解决现有比特币、以太坊、EOS 三大区块链基础网络因其开发语言所导致的拓展性差、二次开发难度大等问题。AlienChain 采用使用更加广泛、二次开发难度更低的 Java 语言进行操作系统开发，并且封装大量的 api 接口以便非专业区块链开发人员来使用，大的降低了普通开发人 员进行去中心化区块链应用的开发，大幅促进区块链技术的发展。

AlienChain 将许多技术解决方案整合，希望克服现有同类平台中存在的问题。AlienChain 最大的挑战是这些技术是否能够获得市场的认可。面对巨头企业的竞争，AlienChain 发挥技术优势，在未来具有弯道超车的潜力。

AlienChain 作为 AlienChain 主网的通证，作为链上智能合约运行的燃料以及系统交易费用的结算。AlienChain 的分布集中程度可观，持有 AlienChain 的地址数量日渐增多。 AlienChain 的社区热度相对于以太坊、EOS 等平台，呈现用户年轻化。

展望

AlienChain 项目采用的解决方案能够解决目前通用平台类项目的痛点，通过降低智能合约的使用门槛，让更多无基础的用户参与区块链建设，目前方向符合区块链技术发展趋势。

1.概述

AlienChain 是一个去中心化应用程序（DApp）开发平台。同以太坊一样，AlienChain 将区块链（Blockchain）和智能合约（Smart contract）相结合，为 DApp 的开发者提供了基础设施，使得开发者能够通过开发 DApp 为用户提供服务。为了满足作为 DApp 开发平台的要求，AlienChain 整合了新的技术解决方案，试图在交易处理能力、对开发者的友好程度以及用户交互等方面都能够有所突破。

首先，在共识机制层面，AlienChain 采用了不同于比特币的工作量证明（Proof of Work）机制，使得 AlienChain 在产币方面具有一定的规律性，由 21 个代理节点共同担任验证，这样在一定程度上能避免出现在以太坊或比特币一样的浪费资源的情况，同时能保证 AlienChain 的每秒吞吐量；在生产区块时，区块生产者每轮（5 秒）进行验证，帮助提高交易处理能力。其次，AlienChain 引入了状态通道（State Channel），利用链下处理来提高区块链整体网络的交易处理速度。并且，除了简单的转账交易外，AlienChain 还把交易双方通过智能合约进行的部分交易也放到了状态通道处理，区块链只用作于最终结算以及特殊情况的“仲裁”，在引入状态通道后，区块链的交易处理能力理论上没有上限。此外，AlienChain 还内置了预言机（Oracle）系统，帮助区块链获取链下信息，完成链下与链上的信息交互。

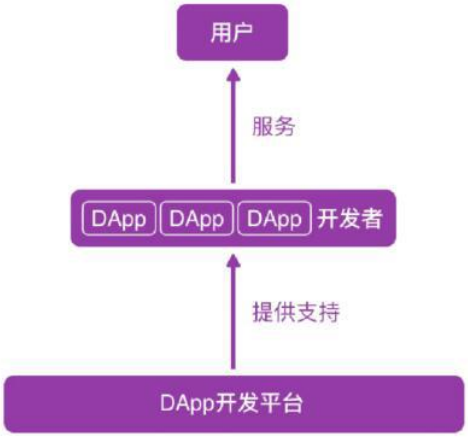
值得注意的是，AlienChain 的工作是对这些方案的“整合”。这些方案都是 AlienChain 原创编译的代码。并且，这些方案都是目前在区块链领域比较前沿的解决方法。

2.行业简介

AlienChain 是区块链领域内的一个 DApp 开发平台，为 DApp 的开发提供基础设施服务。开发平台属于区块链行业中的较底层。开发平台的项目方通过搭建一个平台，为 DApp 的开发者提供一个环境，随后 DApp 的开发者利用平台开发应用程序为用户提供服务。这种模式在区块链技术出现之前就十分普遍。其中，用户最熟悉的例子当属手机应用程序及其开发平台。如在 Google、Apple 等开发平台上，开发者可以通过开发手机应用为用户提供服务。并且，这些公司每年都会定期举办开发者大会，为开发者提供一个交流平台，帮助开发者更好的在平台上开发应用程序，开发者的角色在整个生态中的作用至关重要。

大部分的区块链 DApp 开发平台项目都做着类似的事情。在这个生态中，一共存在三类角色：平台、DApp 开发者、用户，

三者之间的关系如图 所示。



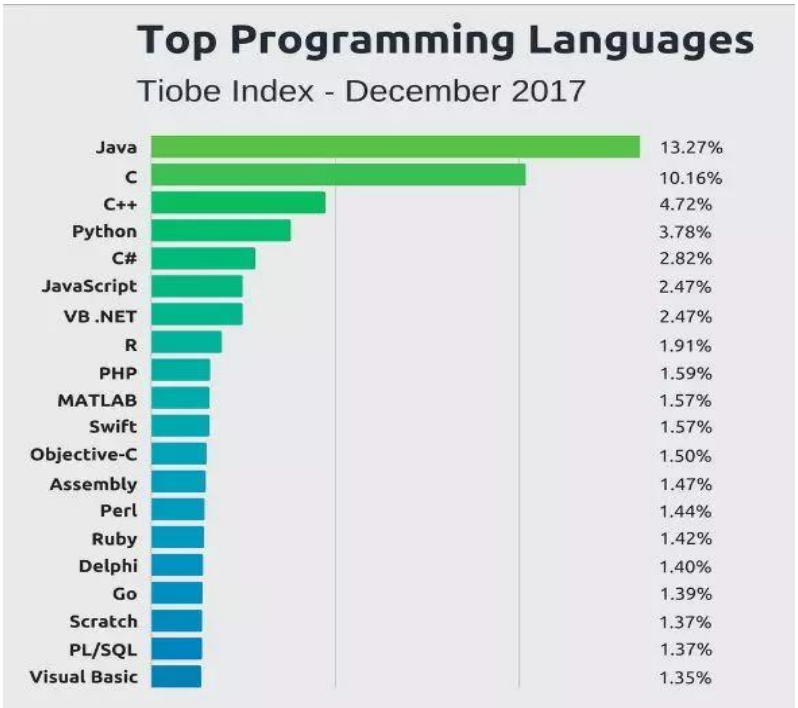
3.技术分析

3.1 编程语言

编程语言(programming language)，是用来定义计算机程序的形式 语言。它是一种被标准化的交流技巧，用来向计算机发出指令。一种计 算机语言让程序员能够准确地定义计算机所需要使用的数据，并精确地 定义在不同情况下所应当采取的行动。

编程语言俗称"计算机语言"，种类非常的多，总的来说可以分成机器 语言、汇编语言、高级语言三大类。电脑每做的一次动作，一个步骤， 都是按照已经用计算机语言编好的程序来执行的，程序是计算机要执行 的指令的集合，而程序全部都是用我们所掌握的语言来编写的。所以人 们要控制计算机一定要通过计算机语言向计算机发出命令。 目前通用的编程语言有两种形式:汇编语言和高级语言。

流行的区块链开发语言，开发语言流行度排行：



三大区块链基础网络开发语言

语言 网络	主开发语言	智能合约开发语言
比特币系列	C++	——
以太坊	Go	Solidity、LLL、Serpent
EOS	C++	C、C++、 WebAssembly

3.1.1 C++语言特点

C++语言是具有面向对象特性的 C 语言的继承者。面向对象编程，或称 OOP 是结构化编程的下一步。OO 程序由对象组成，其中的对象是数据和函数离散集合。有许多可用的对象库存在，这使得编程简单得只需要将一些程序"建筑材料"堆在一起(至少理论上是这样)。比如说，有很多的 GUI 和数据库的库实现为对象的集合。

优点:组织大型程序时比 C 语言好得多。很好的支持面向对象机制。通用数据结构，如链表和可增长的阵列组成的库减轻了由于处理低层细节的负担。

缺点:非常大而复杂。与 C 语言一样存在语法滥用问题。比 C 慢。大多数编译器没有把整个语言正确的实现。

移植性:比 C 语言好多了，但依然不是很乐观。因为它具有与 C 语言相同的缺点，大多数可移植性用户界面库都使用 C++对象实现。

3.1.2 Go 语言

Go 语言是谷歌 2009 发布的第二款开源编程语言。Go 语言专门针对多处理器系统应用程序的编程进行了优化，使用 Go 编译的程序可以媲美 C 或 C++代码的速度，而且更加安全、支持并行进程。

优点：简洁、快速、安全、并行、有趣、开源、内存管理、数组安全、编译迅速。

缺点：Go 语言没有一个主要的框架。Go 语言通过函数和预期的调用代码简单地返回错误(或返回调用堆栈)而帮助开发者处理编译报错。Go 语言的软件包管理绝对不是完美的。

3.1.3Java 语言

Java 是由 Sun 最初设计用于嵌入程序的可移植性"小 C++"。在网页上运行小程序的想法着实吸引了不少人的目光，于是，这门语言迅速崛起。事实证明，Java 不仅仅适于在网页上内嵌动画-它是一门极好的完全的软件编程的小语言。"虚拟机"机制、垃圾回收以及没有指针等使它很容易实现不易崩溃且不会泄漏资源的可靠程序。计算机语言虽然不是 C++的正式续篇，Java 从 C++中借用了大量的语法。它丢弃了很多 C++的复杂功能，从而形成一门紧凑而易学的语言。不像 C++，Java 强制面向对象编程，要在 Java 里写非面向对象的程序就像要在 Pascal 里写"空心粉式代码"一样困难。

优点:二进制码可移植到其他平台。程序可以在网页中运行。内含的类库非常标准且极其健壮。自动分配合垃圾回收避免程序中资源泄漏。网上数量巨大的代码例程。

缺点:使用一个"虚拟机"来运行可移植的字节码而非本地机器码，程序将比真正编译器慢。

移植性:最好的，但仍未达到它本应达到的水平。低级代码具有非常高的可移植性，但是，很多 UI 及新功能在某些平台上不稳定。

3.2 AlienChain 的设计思路

上述几个开发语言中，C++比较主流，但是公认的难度很大的语言，需要很高的技术水平才能把控。Go 语言还没有能够被广泛使用，仍存在一些短板。至于 Solidity、LLL、Serpent、WebAssembly 这几个智能合约开发语言，更是很少有人听说过。这些主网开发语言和智能合约的开发语言的复杂性和高学习成本，严重制约了区块链应用的发展。JAVA 作为一个使用最广泛的开发语言，融入到了当今各个行业的各个系统中去。同时，以太坊的 token 和智能合约方式是目前使用最为广泛和成熟的。使用 JAVA 语言进行开发，并完成以太坊标准的智能合约机制，构建简单易懂的区块链技术开发社区，能够促进区块链技术快速融入现有行业中去。

基于上述原因，AlienChain 选择用 JAVA 语言来实现主网的开发。AlienChain 技术路线是用纯 Java 语言，基于 PBFT 算法的改进，实现了 AlienChain 的高性能共识算法 ABFT。AlienChain 主网实现了以太坊的 ERC20 token 标准【后续会扩展为 ERC-827 标准】，使用 java 语言为智能合约的开发语言。并且封装大量的 api 接口以便非专业区块链开发人员来使用，大的降低了普通开发人员进行去中心化区块链应用的开发，更快促进区块链技术的发展。

3.3 AlienChain 技术方案

3.3.1 共识机制

区块链中最重要理念：共识机制。以太坊则是继承了比特币的 POW，(原定于 17 年底切换为 POS，但从目前来看，短期内无法彻底脱离 POW)。其交易确认时间长、吞吐量性能低下、严重依赖算力竞争的记账确认机制存在安全隐患。

AlienChain 共识模块算法基于 PBFT 算法并结合 DPOS 节点代表选举规则进行改进而成，实现了高性能鲁棒共识算法 AlienChain BFT。在保证 BFT 系统强一致性的 前提下，提升了系统的整体交易吞吐能力以及系统稳定性，可以稳定达到百万的 TPS，交易确认行时间控制在 5 s。

以太坊和 AlienChain 共识机制性能对比

	以太坊(POW)共识机制	AlienChain(DPOS)公示机制
性能（交易处理 TPS）	10-20/s	1000/s
区块产生时间	15s	5 s

3.3.2 什么是 PBFT 算法

BFT（Byzantine Fault Tolerance），即拜占庭容错，是分布式计算领域的容错技术，拜占庭容错来源于拜占庭将军问题。 PBFT（Practical Byzantine Fault Tolerance），即实用拜占庭容错算法，由 Miguel Castro 和 Barbara Liskov 在 1999 年发表的论文《Practical Byzantine Fault Tolerance and Proactive Recovery》中提出。PBFT 算法可以工作在异步环境中，并且通过优化解决了原始拜占庭容错算法效率不高的问题，将算法复杂度由指数级降低到多项式级，使得拜占庭容错算法在实际系统应用中变得可行，目前已得到广泛应用。PBFT 算法可以在失效节点不超过总数 1/3 的情况下同时保证 Safety 和 Liveness。

AlienChain

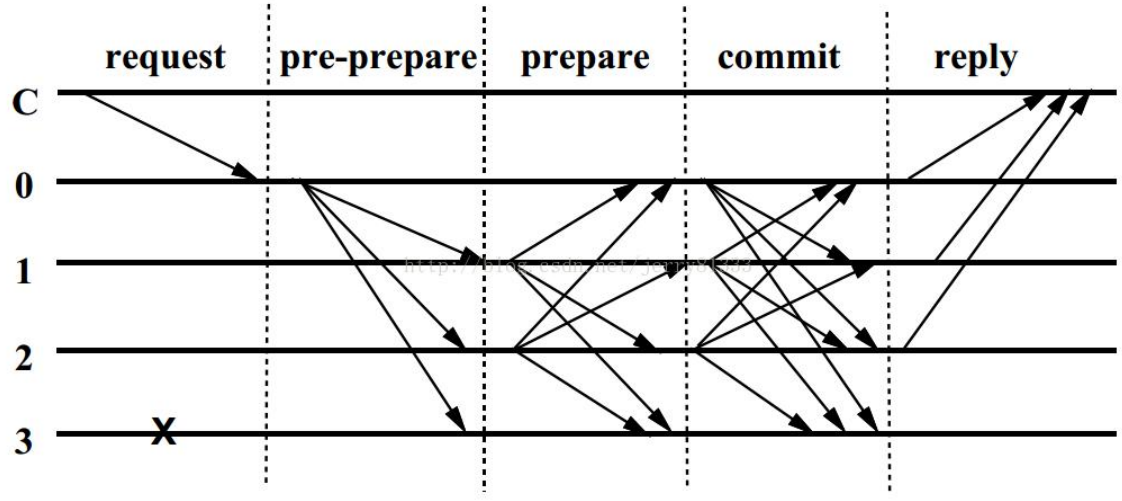
PBFT 算法采用密码学相关技术(RSA 签名算法、消息验证编码和摘要)确保消息传递过程无法被篡改和破坏。

3.3.3 PBFT 算法原理

PBFT 是一种状态机副本复制算法，即服务作为状态机进行建模，状态机在分布式系统的不同节点进行副本复制。每个状态机的副本都保存了服务的状态，同时也实现了服务的操作。将所有的副本组成的集合使用大写字母 R 表示，使用 0 到|R|-1 的整数表示每一个副本。为了描述方便，通常假设故障节点数为 f 个，整个服务节点数为|R|=3f+1 个，f 是有可能失效的副本的最大个数。尽管可以存在多于 3f+1 个副本，但额外的副本除了降低性能外不能提高可靠性。

所有的副本在一个被称为视图（View）的轮换过程中运作。在某个视图中，一个副本作为主节点（primary），其它的副本节点作为备份节点（backups）。视图是连续编号的整数。主节点由公式 p = v mod |R|计算得到，v 是视图编号，p 是副本编号，|R|是副本集合的个数。当主节点失效的时候就需要启动视图轮换过程。

PBFT 算法实现流程如下：



其中 C 为发送请求端，0123 为服务端，3 为宕机的服务端，具体步骤如下：

1. Request：请求端 C 发送请求到任意一节点，这里是 0

2. Pre-Prepare：服务端 0 收到 C 的请求后进行广播，扩散至 123

3. Prepare：123,收到后记录并再次广播，1->023，2->013，3 因为宕机无法广播

4. Commit：0123 节点在 Prepare 阶段，若收到超过一定数量的相同请求，则进入 Commit 阶段，广播 Commit 请求

5.Reply：0123 节点在 Commit 阶段，若收到超过一定数量的相同请求，则对 C 进行反馈

拜占庭容错能够容纳将近 1/3 的错误节点误差，IBM 创建的 Hyperledger 就是使用了该算法作为共识算法。

3.3.4 PBFT 算法优缺点和应用场景

PBFT 算法的优点:

PBFT 算法具有高交易通量和吞吐量，高可用性，易于理解。

PBFT 算法的缺点:

A、计算效率依赖于参与协议的节点数量，由于每个副本节点都需要和其它节点进行 P2P 的共识同步，因此随着节点的增多，性能会下降的很快，但在较少节点的情况下可以有不错的性能，并且分叉的几率很低，不适用于节点数量过大的区块链，扩展性差。

B、系统节点是固定的，无法应对公有链的开放环境，只适用于联盟链或私有链环境。

C、PBFT 算法要求总节点数 n>=3f+1(其中，f 代表作恶节点数)。系统的失效节点数量不得超过全网节点的 1/3，容错率相对较低。

PBFT 算法的应用场景：

PBFT 算法的节点数量是固定的，节点身份提前确定，无法动态添加或删除，只能适用于节点数目固定的联盟链或私有链场景中。

PBFT 在很多场景都有应用，在区块链场景中，一般适合于对强一致性有要求的私有链和联盟链场景，但如果能够结合 DPOS 节点代表选举规则，也可以应用于公有链，并且可以在一个不可信的网络里解决拜占庭容错问题。

3.4 AlienChain BFT 算法

根据 PBFT 算法的原理和优缺点，我们以他为基础，制定了自己的 DPOS 节点代表选举规则，形成 AlienChain BFT 算法应用于 AlienChain 公网中。我们的节点代表选举规则如下：

代理：代理是可用于投票的账号，可以成为验证节点。任何 AlienChain 用户都能申请成为代理(需要消耗 100 万 枚 ALC，且电脑的硬件要求能达到节点需要的最低要求)。

验证节点：验证节点是允许挖掘区块并验证 BFT 协议事务的代理。要成为代理，需要获取足够多的投票数进入前 21 名。每隔 200 区块，重新刷选一次验证节点。没有参与挖掘区块和验证 BFT 协议事务的验证节，已经投票数下降的验证节点，将变为普通代理，由其他符合要求的新代理转变为新的验证节点。验证节点挖掘区块所得的 ALC，作为验证节点的奖励。同时也可以对接矿池，与他的选民一起分享所得到的 ALC。

DPOS 在比特股和 Steem 上已运行多年，交易速度达到每秒上千笔，出块时间短，1.5s；在 EOS 的测试网络上，出块时间更短 0.5s。但是 EOS 是以 21 个超级节点为基础运行的，他要求的节点服务器的软硬件配置极高，一般用户无法参与进来。为了真正达到去中心化，让更多人参与进来，我们建议节点服务器达到 8GB 内存、双核 CPU、100 Mbps 带宽。同时，我们以最低标准节点服务器 4GB 内存、单核 CPU、10 Mbps 带宽的测试环境进行的部署测试，使用 3s 的出块时长，交易确认时间为产生一个区块的时间 3s，系统是能够稳定运行的。（比特币最快交易确认时间为 10 分钟。以太坊的出块时长是 15s，默认以太坊需要 15 次到 30 次的确认，也就是交易上链后 3 分 45 秒到 7 分 30 秒左右的确认时间。EOS 的出块时长是 0.5s，交易广播到区块链中之后，需要经过两轮的节点确认(大致最长时间为 3 分钟左右，336 个区块，然后一笔交易才会成为不可逆的状态。）

ABFT 共识有六个阶段。在每个块上，验证器将经历这些阶段以伪造下一个块。下面的时间，是以最低标准节点服务器设置的。

4.AlienChain 智能合约

4.1 AlienChain 智能合约

AlienChain 的的智能合约执行引擎 AlienChainVM 采用模块化可拔插的设计方式，首先开发的是支持 Java 语言的执行引擎 AlienChain JVM，后续会提供了支持 Solidity 语言的执行引擎 AlienChain EVM。AlienChain JVM 是为了最大程度利用开源社区在智能合约技术和经验方面的积累，提高智能合约的重用性而借鉴了以太坊的 EVM 的虚拟机。AlienChainVM 的智能合约实现完全兼容 Ethereum 的智能合约规范，使用 JAVA 作为智能合约的开发语言，通过微服务的架构设计以及多重安全检查机制为原生 Java 智能合约执行提供了一个高性能安全的执行沙盒。

4.2 什么是智能合约？

智能合约是一种只有通过区块链才能实现的新技术。普通、标准的合同涵盖了当事人之间协议的条款，且常通过法律来强制执行；智能合约是数字化的，存储在区块链中，并使用加密代码强制执行协议。

换句话说，智能合约只是软件程序，与所有程序一样，它们完全按照程序员的意图执行。智能合约就像编程应用程序一样：“一旦出现，就去执行。”

基本上通过数学计算，智能合约可以协商协议中的条款，自动验证履行，甚至执行约定的条款，所有这些都不需要通过中央组织来批准。智能合约使公证人、代理人和律师等中间人几乎毫无意义。

4.3 智能合约如何运作？

智能合约的概念最初是由计算机科学家、密码学家 Nick Szabo 于 1993 年构思出来的。在 1994 年的一篇文章中，Nick 写道：“智能合约的总体目标是满足共同的合同条件（例如付款项、留置权、保密性，甚至强制执行），最大限度地减少异常以及对可信中介的需求。 相关的经济目标包括减少欺诈损失、仲裁和执行成本以及其他交易成本。现今存在的一些技术可以被视为粗略的智能合约，例如 POS 终端和（信用卡）、电子数据交换（EDI）以及公共网络带宽的 agoric 分配。

尽管智能合约在 2009 年比特币诞生时才出现一线生机，但以太坊完全接受了它，使得在其分布式账本中执行和存储智能合约成为可能。以太坊的平台专为执行智能合约而设计，使交易和 ICO（初始代币发行）成为可能且无可挑剔。在许多方面，智能合约是所有区块链技术的基石。此外，许多新兴的区块链初创公司依赖于智能合约有望创造的革命。

就像有一个验证比特币交易的节点网络一样，智能合约也使用节点网络来验证协议的各个方面是否已经完成。他们不需要像律师这样的中间人来验证这些方面是否存在，这些节点和智能合约中的代码本身就可验证。这也使得智能合约透明且可被所有相关方追溯。因此，各方之间的信任不再具有争议。某些时候律师仍会被需要，但大部分工作都已完成。

最后，由于智能合约嵌在所有数据都以分散的分布式方式存储的区块链中，因此直到合同履行完成，没有人能够控制资金。这笔钱通常是区块链的本地加密货币，就像以太坊的以太币一样。

4.4 AlienChain 智能合约优势？

AlienChain 实现了更彻底的去中心化的同时大大提高了 TPS。就拿现有的一代、二代、三代公链 TPS 做比较，BTC 的 TPS 仅仅只有 7 次/秒，ETH 的 TPS 有 30 次/秒，EOS 也只有 3500 次/秒，而 AlienChain 的 TPS 达到了惊人的百万次/秒，完全达到了商业级别的应用，其共识机制也是最公平、最安全的，代码均为原创编写，而非一些公链的 ctrl+v。

AlienChain 未来真正引入 Dapp。就目前来说，时下的主流公链存在难以扩展、缺乏互操作性等不足，在区块链上开发 Dapp 还需要自建模块。底层公链是实现区块链技术应用的基础，Dapp 如果发现某一底层公链具备更好的技术系统，入驻几率会大大提升。据 AlienChain 团队技术负责人介绍，AlienChain 在 Dapp 上进行了大量技术投入，就是为了解决上述难题，为今后区块链系统的 Dapp 引入开辟一条崭新道路，在兼容时下主流公链合约（erc20 合约等）的同时，就国内主流开发语言 JAVA 有着大大的友好度。

过硬的底层技术加持。未来成熟的底层公链应该具备比较完善的技术特质，诸如：“创新的智能合约、“分层”、“分片”、“侧链”、“跨链”、“多链并行”、“去重加密”和“海量存储”等技术，AlienChain 将这些技术在自己身上一一变为了现实，开创了公链界的一个先河。

AlienChain

AlienChain 生态，极大提高了社区用户的积极性和持币能力，这为未来 AlienChain 推广商业应用提供社区能动性，同时极其简单的一键 Token 功能，相比以太坊价格昂贵和复杂的创建合约发币等所消耗的 gas 而言，AlienChain 有着极其低廉的燃料费。

AlienChain 落实商业应用。商业应用引进诸如：电商、支付、物流、游戏、工业机器人、信息溯源等上链，推动区块链技术向传统行业延伸，带动传统行业转型升级，从而享受到 AlienChain 带来的技术红利。

去中心化交易环境。AlienChain 建立的去中心化交易环境，让用户的资产去中心化托管，让所有交易都上链，无法篡改，公开、透明可查，此去中心化交易非交易所，而是商业应用所建立的通证，通过链上交换，完成交易，通过合同等方式完成商业通证的有效权，并展开一系列商业活动。

简单来说，AlienChain 相当于为区块链技术应用到各行各业提供了一个比现有公链网络更好的系统平台，试想，有了 AlienChain 这样更完善、先进的公链系统，其区块链技术应用到商业化前景将更加广阔，AlienChain 的持有价值不可估量。

可以说，整个区块链产业中，底层公链布局都是在刚刚起步阶段，像 AlienChain 这样宝藏公链来说，未来一切皆有可能。

5.使用 AlienChain 创建您的第一个智能联系人！

AlienChain 是一种加密货币，增加了对以太坊智能合约的支持。让我们逐步介绍如何使用稳固性，混音和 AlienChain 控制台与之交互。可能有点难以理解，但请告诉我它的崩溃之处，我将进行更新！我通常会假设您知道以太坊如何处理智能合约，并尝试着重于差异。有一种更聪明的人可以编写与以太坊交互的更好的文档的方法，因此，如果这是一个新的起点，请从此处开始，然后返回 AlienChain 中的差异。

5.1 介绍

AlienChain 是基于 Java 的加密货币。它使用权益证明共识，21 个“验证器”节点，现在支持以太坊虚拟机（EVM）智能联系人（无论如何在其测试网络中）。他们声称> 166 TPS（与以太坊的∽15 TPS 相比），并且由于它们是新的网络，因此每张合约的价格也应更低。

我将从基础知识开始，但是如果您已经熟悉扎实性并编译为字节码，请跳过。

5.2 运行测试网

只需运行./ALC-gui.sh-network = testnet（两个连字符）。让它同步（大约需要 2 个小时），然后从交易中心得到一些 ALC 。

5.3 坚固性

Solidity 是用于智能联系人的编程语言。关于此主题的信息太多了，远远超出了此主题的范围，因此我们仅从其他人的代码：D 开始。让我们以大家最喜欢的 ERC-20 令牌为例。

具体来说，示例 ERC-20 合同，以及有关将其修改为您自己的合同的教程。由于这是一个测试网，因此我很高兴出于演示的目的将随机的开放源代码从 Internet 上删除。

我们修改此代码以为其赋予唯一的名称，编辑地址以及我们要使其成为自己的其他任何变量。

但是，您要与之发布/交互的其他任何固体合同的一般流程应相似。您要获取坚固性代码，将其编译为字节码，然后将其部署到 AlienChain 区块链，然后在其上调用方法。

5.4 编译稳定性

我们采用可靠的源代码，然后重新混合粘贴我们的代码，替换默认的演示，选择“ 0.4.24 + commit.e67f0147 ”作为编译器版本，然后选择“ 开始编译 ”。

然后，我们从下拉列表中选择新令牌（不是 Erc20Interface 或其他任何类），然后单击“ 字节码 ”按钮，这会将我们的合同复制到剪贴板中，将此粘贴到文本编辑器中，发现乱码：

{

“ linkReferences” ：{},

“ object” ： “ 608060405…” ，

“ opcodes” ： “ PUSH1 0x80 PUSH1…”

}

我们只想选择以 608 开头的“ 对象 ”中的内容。这就是我们的“合同”。让我们在其前面添加“ 0x” ，现在为其添加 0x608…我们这样做是为了 eth / ALC，它是格式正确的十六进制字符串。

现在，我们必须将其放入 AlienChain。他们在添加 web3js 支持方面存在一个开放性问题，这将使它变得更加标准，但是，还没有实现，因此我们需要使用其 API 或控制台。我将选择使用他们的控制台。

打开 AlienChain 应用程序，然后转到“ 帮助” |“帮助”。控制台。通过键入“帮助” ，我可以看到以下呼叫。

创建<来自> <值> <数据> <gasPrice> <gas> <fee> <nonce> <validateNonce>

好吧，这有点稀疏，但让我们弄清楚。

来自—您拥有的创建合同的地址。必须保持平衡。

值—’ 0’ 不知道为什么会在这里？

数据-您的合同，以 0x

gasPrice 为前缀-在测试网上，“1”的天然气成本显然很便宜。

气—’1000000’这是一个很大的合同，所以请给它一些爱。

费用-‘5000000’0.005 AlienChain，长格式。AlienChain 称其为 wei 或 satoshi。

随机数-应该是可选的，但在控制台中是必需的，请参见下文了解如何找到

validateNonce-嗯，“真”不确定是否重要

随机数

随机数只是您已发送的交易数。在控制台中，您可以通过键入’ getAccount <您的地址> ‘找到它

> getAccount 0xebefc16b7c4d29224d67ced1e1400a1e8e024b9d

{

“成功”：true，

“消息”：“成功操作”，

“结果”：{

“地址”：“0xebefc16b7c4d29224d67ced1e1400a1e8e024b9d”，

“可用”：“10000000000”，

“锁定”：“0”，

“立即”：“0”，

“transactionCount”：0，

“pendingTransactionCount”：0

}

}

太酷了，因此在这种情况下，我的现时值为 0。看起来我很有钱！但这仅是 10 个 AlienChain。

我喜欢在文本编辑器（记事本）中构造它，因此我可以更轻松地对其进行编辑，因此，在带有合同的粘贴文件中，让我们添加其他内容。它看起来像

创建 0xebefc16b7c4d29224d67ced1e1400a1e8e024b9d 0 0x60806040523...giantstring 1 1000000 5000000 0 是

将其粘贴到 AlienChain 控制台中，然后按 Enter。运气好的话，你应该看到

{

“成功”：true，

“消息”：“成功操作”，

“结果”：“0x2aec2fd2ef5a1309fd36ad2e2ed2303e9132fd9ad015fa0c6a5c1bd7039ed5dd”

}

这意味着它现在应该在“交易”标签中显示为“待处理”。等待 5 秒。

好的，现在我们有了合同！查找“合同地址”与以太坊相同（源地址和随机数的某种哈希值），但是我们将采取捷径。如果我们双击交易中的 create call，则“收件人”字段是我们的合同地址。在这种情况下，“0xdf678702e40cc6daf1974d431b9bb573459e0306”。

恭喜，您已经在区块链上创建了合同！

调用我们的合同

让我们将一些新的 ERC-20 硬币发送到另一个地址

在我们的 AlienChain 控制台窗口中寻求帮助

调用<从> <至> <值> <gasPrice> <gas> <fee> <nonce> <validateNonce> <data> <local>

描述参数：

从–我们的合同所有者，从第一次调用

到–我们的新合同地址

值–再次为 0，我们没有发送气体

价格– 1

气体–呼叫量小得多，因此让我们尝试 “ 50000”

费用- “ 5000000”

随机数-将是新的随机数，因为我上次的随机数为’ 0’ ，现在将为’

1’ validateNonce- “ true”

数据-ABI 调用，请参见下文

local- “ false”这控制我们是进行交易，还是使用只读方法，例如 getBalance（）

ABI 通话数据

在线上有很多关于此的信息。它与以太坊相同，因此让我们以简单的方式进行。这是另一个相当复杂的主题，但让我们简化一下。如果我们回到混音窗口...

转到“运行”选项卡，从下拉列表中选择我们的令牌，然后单击部署。

在“已部署的合同 ”中，展开窗口，您可以看到可用的呼叫。让我们展开 “ transfer ” 并填写地址和令牌。我将把 10000 个 TestToken 发送到另一个地址。填写字段后，单击小工具箱图标，它将以下内容复制到我的剪贴板中

0xa9059cbb000000000000000000000000c1e6ceca4ccad20e59849cb7a9678394f7313000000000

00000000000000000000000000000000000000000000000000000000000710710

您可以手动构建它们，但是使用 web3js 或 remix 网站更容易。这是我们的电话数据字段。

我们最终以

呼叫 0xebefc16b7c4d29224d67ced1e1400a1e8e024b9d

0xdf678702e40cc6daf1974d431b9bb573459e0306 0 1 50000 5000000 1 true

0xa9059cbb000000000000000000000000000000000837c1e6ceca4ccad20e59849c000000000000000000000000

我们将填写好的版本粘贴到控制台中，然后运行它。再次，我们看到成功和一笔未决的交易，一旦交易成功，我们就将资金转移到了一个新帐户中。

哇，这里有很多要讲的内容，但希望能给您一些想法，让他们知道如何使用主要是以太坊工具和偶尔的 AlienChain 控制台功能与 AlienChain 进行交互。我希望以后会更容易。

6.AlienChain 基金会

AlienChain 是由 AlienChain 基金会发起的项目，基金会组成：

1，技术团队：目前已过渡到智能合约开发工作，由 BastKey 团队担任智能合约合作方

2，运营团队：运营 AlienChain

7.AlienChain 核心思想

AlienChain 是一个开源的区块链平台，JAVA 语言原创编写，交易确认速度极致 5 秒，可在 AlienChain 主网上开发智能合约、Dapp，AlienChain 公链将提供大量封装接口，提供给开发者和零基础用户开发和使用智能合约，这点是其他任何公链都无法做到的。

了解用户习惯才能发展至大，以用户为基础，以商用为筑基，以应用为落地，才是 AlienChain 最为核心的理想。

8.AlienChain 分配方案

8 亿 ALC - 第三方 BastKey 团队持有

2 亿 ALC - 用于内部运营团队基金会

14.79 亿 ALC - 流通数量

1.21 亿 - 已销毁（截至时间：2020/6/18）

总计：26 亿（不计入销毁数量，总计：24.79 亿）

21 个代理（验证节点）每年共计 6300 万产出，平均每个代理 300 万 ALC/年（无限通胀，区块奖励/10）

9.AlienChain 起源

AlienChain 起源于电影《ET 外星人》，寓为外星人的前沿技术。

10.如果您想加入技术团队

相关要求：

1、有以太坊区块链智能合约开发经验。

2、熟悉 solidity 智能合约语言

3、了解一点主流的共识算法如 PoS, DPoS, PBFT，了解一点安全协议和加密算法。

如果您想了解智能合约初学者，请打开以太坊入口：

https://ethereum.org/build/

学习相关知识

AlienChain

招聘要求：

1、至少 3 年以上互联网或应用软件技术开发经验，精通 C++/C、Go、Python、Java、Javascript 中的一种或多种开发语言； Python 和 Go 语言最好掌握。

2、精通 Linux，熟练掌握 Docker 容器技术的原理，部署和使用优化，有加密货币或区块链技术实际项目经验者；

3、熟悉 Ethereum、OpenChain、Bitcoin、Hyperledger Fabric 等相关开源项目，有研究和贡献经验者；

4、有相关分布式系统架构和 P2P 网络系统开发经验者；

5、具备密码学技术相关背景，有相关项目背景者；

6、精通 solidity 等常用智能合约语言及开发优化者；

7、理解各类主流的共识算法，包括不限于 PoW，PoS，DPoS，PBFT，Paxos，Raft 等；

8、了解主流 NoSQL 数据库的原理与使用，尤其是 KV 型数据库，包括不限于 LevelDB，RocksDB 等；

9、精通各种数据结构和算法，对密码学、安全协议和加密算法有研究，熟悉分布式、多线程及高性能的设计、编码及性能调优；

10、了解 HTTP/2 协议，理解 RPC 框架，具备使用 protobuf 的开发实践经验。

如果您是一位初学者，请认真学习以太坊官网上的智能合约编译教程。

学习步骤：

1，学会使用以太坊编译智能合约

2，学会使用智能合约做成 Dapp

3，学会使以太坊上的智能合约转移到 AlienChain 主网

最终扩散思维，能够发挥想象空间基于区块链主网编译创新性合约。