基于区块链项目开发语言选型的初探①

陈 胜[1], 曾 靖[1], 左 春[2]

[1](中国科学院软件研究所 互联网金融技术研究中心, 北京 100190)
[2](中科软科技股份有限公司, 北京 100190)

摘 要: 区块链技术及持续发展的各种开源区块链项目, 正吸引着越来越多开发者的目光. 而繁多的主流编程语言及特色各异的区块链项目, 使得开发者在开发基于区块链的系统与应用时, 在技术选型上难免会有所困惑. 为此, 对 12 个较知名的开源区块链项目, 使用了源码统计工具 Cloc 统计其源代码, 并进一步通过阅读相关代码及文档, 分析和比较了其开发语言的构成. 同时, 以图表的形式呈现其体量和关注度, 从而为区块链开发者在选型上提供参考信息和建议.

关键词: 区块链; 开源; 编程语言; 统计

Discussion of Selection for Languages Based on Blockchain Projects Development

CHEN Sheng[1], ZENG Jing[1], ZUO Chun[2]

[1](Institute of Software, Chinese Academy of Sciences, Research Center for Internet Finance Technology, Beijing 100190, China)
[2](Sinosoft Company Limited, Beijing 100190, China)

Abstract: More and more developers are being attracted by blockchain technologies and sustainable development of various open source blockchain projects. Nevertheless, the various popular programming languages and distinctive blockchain projects could perplex the developers inevitably when they commence to develop systems or applications based on blockchain. So, this paper utilizes a tool named Cloc to count the lines of source code in 12 well-known open source blockchain projects. By further reading codes and documents associated, we try to analyze and compare the programming language constitutions of the projects. Moreover, the paper demonstrates their project sizes and the attention degree to them in the form of charts. Finally, some reference information and suggestions about selections are given to the blockchain developers.

Key words: blockchain; open source; programming language; statistics

区块链是起源于比特币实验的分布式数据库技术. 该概念在中本聪的白皮书[1]中被首次提出.

比特币网络中的所有全节点以区块链的方式共享交易数据库的完整副本. 交易由交易发起者或生成区块的挖矿者签名, 并被打包在区块中. 每个区块包含了指向前一个区块的哈希值, 以此构造出一条不可篡改的, 从最新区块指向创世区块的链. 每个全节点也可由此推算出在区块链上的任意历史点以及任意地址/账户的未花费余额.

尽管针对不同行业不同场景的区块链应用存在衍生功能个性差异, 但是它们都被贴上了“区块链”的标签, 这些应用在基本功能和系统结构方面也就存在着诸多共性. 这些共性的基本功能包括:

① 用户密钥对以及账户/地址的生成;
② 交易的构造和签名以及验证签名;
③ P2P 消息应答以及数据传输;
④ 共识机制以及交易的打包入块与成链.

这些基本功能, 是形成区块链应用网络不可或缺的部分. 除此之外, 伴随着区块链生态的发展, 也衍生出了其他辅助功能. 例如: 命令行人机交互(cli)、更友好的钱包(wallet)应用以及区块链的 web 浏览检索(explorer)等.

① 收稿时间:2016-07-18;收到修改稿时间:2016-10-12 [doi: 10.15888/j.cnki.csa.005775]

针对比特币实验所暴露的问题以及应用场景的各自特点, 区块链应用通常对基本功能进行改进以符合其需求. 例如, 为了改进出块性能, BitShares[2]在共识机制方面采用授权股份证明(Delegated Proof of Stake, DPoS)[3]代替比特币的工作量证明(Proof of Work, PoW). 为了解决数据膨胀危机, 在链形态上, Lisk 采用主侧链加 DApp(Decentralized Application)机制代替比特币的单一主链[4]. 而以太坊(Ethereum)则将比特币的脚本机制发展为图灵完备的智能合约, 允许开发者编写和发布可以运行在以太坊虚拟机之上的分布式智能合约程序[5].

无论是基本功能, 还是衍生功能, 具体到它们的编程实现, 目前主流的编程语言均可以胜任. 在动手搭建自己的区块链应用之前, 有必要对目前的相关开源项目进行一番审视, 从而针对具体的应用场景, 选择合适的编程语言及区块链入门项目. 本文对较知名的 12 个开源区块链项目进行源码统计, 并根据相关源码与文档, 分析其开发语言的构成. 同时, 结合各项目的体量与热度, 给出选型建议.

1 入选标准

通常去中心化的区块链应用会相应地开放源码. 本文采用以下标准:

① 已公开发布的开源竞争币, 包括: 比特币(Bitcoin[6])、以太坊(Ethereum[7])、比特股(BitShares[8])、NXT[9]及 LISK[10];

② 针对特定行业或领域的开源区块链代表, 包括: 超级账本(HyperLedger[11], 智能合约)、公证通(Factom[12], 公证防伪)、IPFS[13](分布式文件)、比特信(BitMessage[14],社交通讯)、OpenBazaar[15](电子商务)、Storj[16](文件存储)以及 RSCoin[17](中心化数字货币).

2 项目分析

2.1 源代码获取

上述入选项目都采用了 github 作为代码仓库, 在本地安装一个 git 客户端, 通过命令行即可将代码 clone 到本地文件目录以进行分析:

例: git clone git@github.com:bitcoin/bitcoin.git

有两种情况必须加以考虑:

① 同一项目存在多种编程语言实现的, 例如: 比特币、以太坊等, 选取 star 数最高的语言实现纳入统计.

② 某些项目将相对独立的功能集合分拆为独立的 git 工程, 例如: Lisk 项目下的 LiskHQ/lisk 和 LiskHQ/lisk-ui、LiskHQ/lisk-cli. 相应地对每个项目建立一个父目录 blockchain 以容纳多个 git 工程的代码(见图 1).

图 1 获取源码后的目录结构

2.2 代码分类统计

CLOC(Count Lines of Code)是一款开源的跨平台代码统计工具, 可以在终端下以命令行方式在线下载安装.

2.2.1 初步统计

在终端下进入 blockchain\lisk 目录, 使用 cloc 默认命令: "*cloc .*", 结果如图 2 所示.

Language	files	blank	comment	code
JSON	13	0	0	94085
JavaScript	168	4108	1643	24719
LESS	32	431	58	4956
HTML	41	12	763	3026
SQL	2	22	10	130
SUM:	256	4573	2474	126916

图 2 初步统计 lisk 源码的结果

2.2.2 可疑的 JSON

显然, 占据榜首的 JSON 很可能不是源代码, 进一步使用命令: "*cloc . --by-file --include-lang=JSON*"单独对 JSON 文件进行统计, 结果如图 3 所示.

File	blank	comment	code
./lisk/genesisBlock.json	0	0	91088
./lisk/test/genesisBlock.json	0	0	1760
./lisk/test/genesisDelegates.json	0	0	726
./lisk/test/config.json	0	0	164
./lisk/package.json	0	0	100
./lisk/config.json	0	0	83
./lisk-cli/config.json	0	0	51
./lisk-ui/package.json	0	0	50
./lisk-cli/package.json	0	0	40
./lisk-ui/bower.json	0	0	12
./lisk/helpers/validator/package.json	0	0	5
./lisk/helpers/json-schema/package.json	0	0	5
./lisk-cli/genesisBlock.json	0	0	1
SUM:	0	0	94085

图 3 单独统计 lisk 中 JSON 文件所得结果

通过阅读官方文档, 可以获知 genesisBlock.json 是用于创建预挖矿创世区块的定义文件, 应该排除出源码统计.

类似地, 应该排除其它项目中用于本地化显示的以及以太坊中用于测试用例的 JSON 文件. 另外比特信使用了 39227 行 TypeScript 作为本地化文件, 也在排除之列.

Cloc 提供了过滤参数"*--exclude-lang=*"用于排除指定类别语言. 我们使用"*--exclude-lang=JSON, TypeScript*"可以达到上述目的.

2.2.3 项目体量

应用上述命令, 首先针对每个项目统计代码, 然后用"*cloc --sum-reports*"合并统计, 结果[1]如图 4.

File	files	blank	comment	code
ethereum.txt	1280	43066	61757	478939
hyperledger.txt	805	34689	83899	207216
nxt.txt	691	21134	21495	142156
bitcoin.txt	757	22945	20411	138073
bitshare.txt	556	15568	17985	131631
ipfs.txt	757	16672	9374	85640
openbazaar.txt	248	6399	3565	44488
lisk.txt	243	4573	2474	32831
bitmessage.txt	115	2639	2152	25952
storj.txt	117	2580	1902	12452
factom.txt	86	2517	1652	10498
rscoin.txt	14	721	199	1663
SUM:	5669	173503	226865	1311539

图 4　合并统计所得结果

代码行数反映项目体量. 从图 4 中可以发现, 同样定位为智能合约平台的竞争币, Lisk 的体量尚不及以太坊(Ethereum)的十分之一. 如果希望完整学习及理解一款区块链平台, Lisk 将会是更好的选择.

2.2.4 项目热度

托管平台上项目的 star 数及 fork 数可体现该项目受关注的热度. 根据从 github 及 bitbucket 上所获取的数据, 可得到如图 5 所示的各项目的 star 数及 fork 数情况.

图 5 中, 除 NXT 的数据来源于 bitbucket 外, 其他项目的数据均来自 github. 并且由于 bitcoin 的 star 数与 fork 数分别为 9839 与 6564, 远大于其余 11 个项目的, 故图 5 未显示 bitcoin 的情况. 同样以 Lisk 和以太坊做对比说明: 以太坊的受关注热度是 Lisk 的约 20 倍. 因此如果不是想自己另起炉灶, 而是融入主流生态, 现阶段融入以太坊的生态才是更合理的选择.

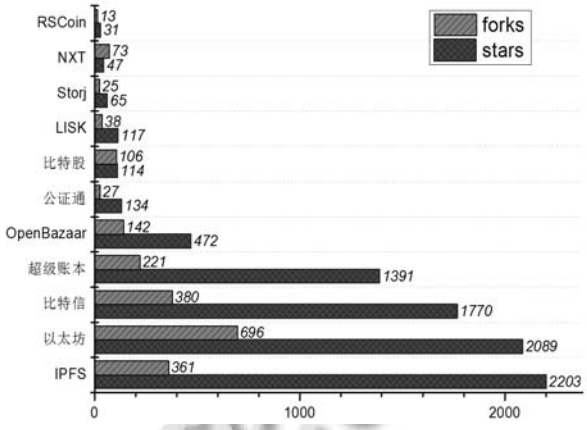

图 5　各项目的 star 数及 fork 数对比

2.2.5 语言热度

对于上述 12 个项目, 分析对比它们对编程语言的选择, 对区块链开发者在技术选型上也会有所帮助. 我们单独审视各个项目, 并对编程语言进行投票, 投票规则如下: 如果该项目使用某种语言的代码量超过其总代码量的 1/10, 那么此项目投了该语言一票.

根据上述投票规则, 得到的统计结果如图 6 所示. 图中各语言得票情况的对比正好符合一个趋势: 近年来脚本语言的兴起, 已经在越来越多的领域代替传统的编程语言.

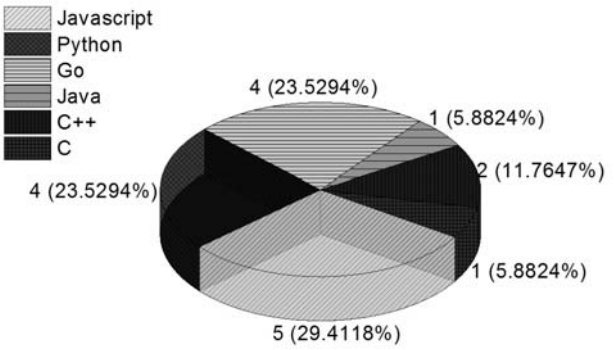

图 6　各编程语言得票情况

具体到本文考察的开源区块链项目, 就主要开发语言的选择上, bitcoin 和 bitshares 采用了 C++, NXT 采用 Java, 而其他项目均采用脚本语言或兼具了脚本语言优点的 Go 语言作为主开发语言.

在前端界面, javascript 占据着主导地位. 且 Lisk 和 Storj 采用了 Nodejs 实现, 故在前后端均采用 javascript, 而其他采用 javascript 的项目均将其用于前

[1] 本文统计结果访问地址：https://github.com/chen4w/blockchain_lang

端或客户端.

而在后端, 脚本语言又常常通过扩展机制借助 C 语言实现性能卓越的加密/解密以及签名这样的基础功能. 比如, Lisk 引用了用 C 语言实现的 ed25519 密钥对生成模块, 同样地, 其所引用的用于加密/解密, 签名/验证的 crypto 模块是对 C 语言实现的 OpenSSL 的封装.

3 结语

尽管对比特币试验本身褒贬不一, 但比特币通过结合数学原理与网络共识, 成功建立和生长出的试验网络给人们带来了新的认知, 即构建去中心化的信任是可行的.

针对比特币的不足并结合具体的应用场景, 形形色色的区块链应用正不断涌现. 从 Lisk 那样的小型团队到超级账本那样的大型团队, 无论是普通程序员还是行业巨头, 都对区块链表现出了浓厚的兴趣, 并切身地参与其中. 而面对如此众多, 各具特点的开源项目, 选择适合自己的项目就显得不太容易了. 本文从分析各个项目所采用的编程语言的组成入手, 直观地展示项目的体量、关注度并关注开发人员对各种编程语言的结合使用.

相比较传统形态的应用, 区块链仍未形成系统性的软件生态. 未来也许会出现主导行业的规约或者架构. 但现阶段各类应用仍处于演化过程中. 故在选型开源项目之前, 区块链开发者需要审视如下问题.

自行维护主链生态, 还是开发寄生在其他主链生态上的应用, 前者开发难度较大, 而后者只需要理解 DApp(Decentralized Application)或智能合约的编程思想; 开发一个同样的区块链平台, 如 Lisk, 还是针对特定行业的区块链应用, 如 Storj, 虽然都主要采用脚本语言来实现, 但后者需要的代码体量可能会少很多; 选择编译型语言, 还是解释型语言来开发区块链项目, 也需要对开发效率、运行效率、平台适应性及应用场景等多种因素进行综合考量.

参考文献

1 Nakamoto S. Bitcoin: A peer-to-peer electronic cash system. https://bitcoin.org/bitcoin.pdf. 2009.

2 Consensus technology. https://docs.bitshares.org/bitshares/whatis.html. 2016.

3 Delegated proof of stake. https://docs.bitshares.org/bitshares/dpos.html. 2016.

4 Lisk. What is Lisk? And what it isn't. https://blog.lisk.io/what-is-lisk-and-what-it-isnt-e7b6b6188211#.joflk4qbs. 2016.

5 White Paper. https://github.com/ethereum/wiki/wiki/White-Paper#ethereum. 2016.

6 Bitcoin. https://github.com/bitcoin. 2016.

7 Ethereum. https://github.com/ethereum. 2016.

8 Bitshares. https://github.com/bitshares. 2016.

9 NXT. https://bitbucket.org/JeanLucPicard/nxt/overview. 2016.

10 LISK. https://github.com/LiskHQ. 2016.

11 超级账本. https://github.com/hyperledger. 2016.

12 公证通. https://github.com/FactomProject. 2016.

13 IPFS. https://github.com/ipfs. 2016.

14 比特信. https://github.com/Bitmessage/PyBitmessage. 2016.

15 OpenBazaar. https://github.com/openbazaar. 2016.

16 Storj. https://github.com/Storj. 2016.

17 RSCoin. https://github.com/gdanezis/rscoin. 2016.