基于IPFS的存储解决方案

**北京网录科技有限公司**

|  |  |  |  |
| --- | --- | --- | --- |
| 版本 | 作者 | 修改描述 | 修改日期 |
| 1.0 | 李冠军 | 初稿 | 2018-6-13 |
|  |  |  |  |

目录

[一、引言 3](#_Toc517179841)

[1.1 编写目的 3](#_Toc517179842)

[1.2 项目背景 3](#_Toc517179843)

[1.3 缩略语 3](#_Toc517179844)

[1.4 参考资料 3](#_Toc517179845)

[二、IPFS简介 3](#_Toc517179846)

[三、IPFS安装部署 4](#_Toc517179847)

[3.1 IPFS的安装 4](#_Toc517179848)

[3.2 使用Docker安装IPFS 4](#_Toc517179849)

[3.3 IPFS的config 4](#_Toc517179850)

[3.4 IPFS私网搭建 5](#_Toc517179851)

[四、现有数据使用IPFS存储解决方案 5](#_Toc517179852)

[4.1 数据存储现状和需求 5](#_Toc517179853)

[4.2 文件存储解决方案 5](#_Toc517179854)

[4.3数据记录存储方案 6](#_Toc517179855)

[五、现有数据兼容IPFS存储解决方案 6](#_Toc517179856)

[六、IPFS功能扩展方案 7](#_Toc517179857)

[6.1 扩展原因 7](#_Toc517179858)

[6.2 ipfs多级目录的修改问题 7](#_Toc517179859)

[6.3 多版本支持 7](#_Toc517179860)

[6.4 数据、文件联合读写 7](#_Toc517179861)

# 一、引言

## 1.1 编写目的

本文档编写的目的是为了在IPFS框架内解决在wanchain基础上改造的联盟链以及其他区块链产品的永久存储问题（包括文件、各种类型的数据），并给出可行的解决方案。

## 1.2 项目背景

公司基于Ethereum开发了wanchain，并正在开发联盟链。由于区块链的固有特点，导致区块链无法应对大数据的存储需求。

## 1.3 缩略语

IPFS：the InterPlanetary File System

## 1.4 参考资料

IPFS在github的网站：https://github.com/ipfs/ipfs

IPFS官网文档：https://ipfs.io/docs/

# 二、IPFS简介

IPFS全名为the InterPlanetary File System，是一个永久的、去中心化、共享的分布式文件系统，通过内容生成唯一hash作为地址，支持版本化和点对点的超媒体。

IPFS是基于以往成功的P2P项目成功经验，并综合这些项目的优点实现的一个系统。它使用DHT（Distributed Hash Tables）定位文件的地址，使用BitTorrent进行数据交换，参考了Git进行版本控制，使用SFS（Self-Certified Filesystems）解决可变路径问题。

IPFS在远程文件共享、版本控制、分布式key-value数据库各种CDN、永久静态Web页面等方面有很好的支持，但是对于频繁修改的场景不太适合。

# 三、IPFS安装部署

## 3.1 IPFS的安装

下载地址：<https://dist.ipfs.io/#go-ipfs>

安装： tar xvfz go-ipfs.tar.gz

cd go-ipfs

./install.sh

## 3.2 使用Docker安装IPFS

1、docker安装

下面给出的是ubuntu的安装方法，其他版本的安装可以自己网上查询。依次执行下面的命令，有些版本可能有错误，按照提示执行即可，例如有些情况下需要执行如下命令：

apt --fix-broken install

**安装命令行：**

sudo apt-get update

sudo apt-get install docker

sudo apt-get install docker.io

sudo apt-get install docker-registry

**Ubuntu下docker使用非root权限运行docker：**

sudo groupadd docker

sudo gpasswd -a ${USER} docker

sudo systemctl restart docker

newgrp – docker

**测试docker：**

docker run hello-world

2、从docker启动IPFS

docker run -d

--name {ipfs容器的名字，例如ipfs\_host}

-v {映射到容器用于文件输入输出的路径，需要读写权限，例如/data/export}:/export

-v {映射到容器用于IPFS配置文件的路径，需要读写权限，例如/data/config}:/data/ipfs

-p 4001:4001 -p 127.0.0.1:8080:8080 -p 127.0.0.1:5001:5001 //IP端口映射

ipfs/go-ipfs:latest //使用的影响和版本号

## 3.3 IPFS的config

默认仓库路径：~/.ipfs

可以通过设置下面的变量修改：export IPFS\_PATH=/path/to/ipfsrepo

## 3.4 IPFS私网搭建

1、共享密钥

私有网络所有的节点可以共享同一个密钥，这样就可以拒绝非授权的连接。

使用go下载安装：

go get -u github.com/Kubuxu/go-ipfs-swarm-key-gen/ipfs-swarm-key-gen

go install github.com/Kubuxu/go-ipfs-swarm-key-gen/ipfs-swarm-key-gen

然后创建密钥：ipfs-swarm-key-gen > ~/.ipfs/swarm.key

复制到所有节点对应路径下

2、启动节点

在所有节点下执行下面操作，删除掉所有的默认启动节点

ipfs bootstrap rm —all

选择一个默认节点，在其它节点执行下面的操作

ipfs bootstrap add /ip4/<默认节点IP>/tcp/4001/ipfs/<默认节点ID>

3、验证

在不同节点上执行add和get验证有效性

# 四、现有数据使用IPFS存储解决方案

## 4.1 数据存储现状和需求

用户数据已经存在，有多种格式，可能是文件，例如合同文本、图片、音频、视频等，也可能是数据库的记录。

用户对数据存储的要求也有很大差别，首先是文件的存储需求，具体如下：

1. 对于文件，直接存入IPFS，返回访问地址，以便以后访问。
2. 文件存入IPFS，但需要加密，只有赋予权限的人才能访问。
3. 用户自己存储文件，需要在IPFS里面存储访问的地址。
4. 用户可能需要按照目录和版本的方式存储文件。

用户对数据记录的存储需求：

1. 把用户的数据按照key-value的方式存入到IPFS中。
2. 某些敏感的数据只存放验证的hash和访问地址到IPFS中，或者对数据加密，只有经过授权的用户才可以访问。

## 4.2 文件存储解决方案

1. 对于普通的文件或者目录存储需求，直接通过ipfs add命令上传到IPFS，把返回的地址保持到区块链或者其他地方，使用时通过地址访问即可。
2. 如果需要对他人提供可变的命名服务，可以使用ipfs name publish命令把一个目录使用节点id或者另外生成的秘钥的公钥生成的hash发布出去。
3. 如果只是本节点访问的可变命名服务，可以ipfs files的相关命令处理文件目录。
4. 目前对于需要版本控制或者对现有改变多级目录的情况，虽然可以支持，但较为麻烦，需要做的事情太多，在下面的章节将会给出解决方案。

## 4.3数据记录存储方案

对于数据存储，IPFS提供的dag和object命令都可以实现，但是dag更加灵活一些。如果使用object，需要把数据直接按照二进制处理，修改起来相对麻烦一些。而dag可以吧数据按照json格式处理，处理起来相对灵活。下面的方案是按照dag的格式处理的。

用户要保存到数据，保存成json格式。json格式定义如下

{

“name”: “张三”,

“number”: 20, //对于普通数据，按照平常的格式即可

“mobile”: {

“type”: “contenthash”, //数据类型，下面会给出支持的类型

“hash”: “0x1234567890”,

“addr”: “读取真实数据的访问地址”

},

……

}

数据类型定义：

1. contenthash

用户出于保密需求不给具体内容，只给内容hash用于验证，具体内容通过给出的地址在经过授权后获取。

需要的字段包括：

hash（内容hash，目前只支持sha256，如果需要，可以按照ipfs的方式保存hash）

address（获取具体内容的访问地址）

1. ipfspath

该字段对应的保存到ipfs中的文件或目录

需要的字段包括：

address（该文件或目录在ipfs中的地址）

name（文件或目录的名字）

size（文件或目录的大小）

# 五、现有数据兼容IPFS存储解决方案

对于目前有各种原因不能使用IPFS，以后可能会转移到IPFS的用户，在保存数据时要考虑到数据迁移的兼容和成本问题。

在使用IPFS存储的场景中，一般是把文件和其他数据存储到IPFS，把返回的指向该数据的key保存到区块链或者其他可用于查询的系统中。

根据需求的不同，可以有两种解决方案解决迁移的问题：

1. 如果key值存储在可变的系统中，例如MongoDB，或者虽然存储在区块链等不易改变的系统中，但是可以花费一定的代价在迁移时重新发布新的key值。这种情况在开始的存储中较为简单，只要按照平常的存储方式存储就可以了，在迁移的时候需要做的工作量较大，需要更新为新的key值。
2. 第二种方案是在一开始存储时就按照IPFS的算法生成key值，这样在迁移到IPFS后只要改变程序中读取数据时的指向就可以实现迁移。但是这种方法使用起来较为复杂，需要开始做好规划设计，否则会出现迁移失败的情况。对于只要数据存储的情况处理起来可能还相对简单一些，如果有文件、目录的存储则处理起来比较麻烦。

# 六、IPFS功能扩展方案

## 6.1 扩展原因

IPFS提供了很强大的分布式文件存储功能，可以实现版本控制。目录管理功能，但是这些功能在使用的过程中并不是通过一次调用一个方法就能实现，需要反复多次调用才能实现这些功能，使用起来相对复杂，需要对IPFS有一定的了解才能使用，因此需要对这些功能进行集成，按照需求提供一些方法可以一次调用实现版本控制，数据、文件一次性存储等需求

## 6.2 ipfs多级目录的修改问题

对于多级目录的存储中，如果深层次的文件或目录结构发生改变，需要多次逐层更新链接才能最终反映到根目录，例如/a/b/c.txt，现在需要改变c.txt的内容，那么需要做的是更新c.txt的内容，使用返回的id更新b中的连接，再使用更新b返回的id更新a的连接，最后返回的a的id才是需要的最后结果。扩展后，提供一个新的方法，可以直接更改c.txt的内容，并以此更新连接后，直接返回更新后的a的id。

## 6.3 多版本支持

IPFS可以使用其提供IPNS功能来实现多版本支持，但是并没有提供直接使用多版本支持的方法，需要使用者自己通过综合IPFS提供的多个方法来实现该功能，这样也为使用者设置了障碍。如果把这个需求整理一下，对外提供一个接口，可以一次调用实现多版本读写的问题，可以提高使用效率，降低复杂度。

## 6.4 数据、文件联合读写

目前数据和文件是没有办法使用一个方法一次完成读写的，在写入的时候，需要先把文件保存，然后把返回的id写入到数据里面，在进行保存，读的时候相反，需要先读出数据，取出文件的id后在读取文件，在某些场合下不是很方便，可以通过添加一个方法，把这些功能集成到一块，通过一个方法完成读写。