软件工程概论
课程综合报告（封面）

学号：2030416018

姓名：高歌

班级：20 软件卓越班

签名： 

2022 年 12 月 28 日

目　录

一种基于区块链技术的可靠云存储平台实现

高歌

（苏州大学计算机科学与技术学院，江苏苏州）

摘　要：随着互联网服务对数据的依赖性逐渐提高，对于长期可靠云存储乃至永久存储的需求也渐渐出现。然而目前各大云存储平台几乎都受限于其公司运营模形式和基于服务器集群的集中式存储方案而难以做到数据的长久存储。本文聚焦于数据在云平台上的长期可靠存储这一目标，尝试提出一个基于以激励机制驱动的去中心化网络的开源云存储系统，以尽可能实现数据在平台上的永久存储，并引入内容审查机制以适应各国法律法规。

关键词：去中心化存储；分布式存储；区块链；永久存储；隐私安全性；数据可靠性

第一章　背景介绍

1.1 系统背景

当前随着互联网数据访问频率的增加，对可靠云存储服务的需求日趋增加。然而考虑到平台运营成本等因素，很少有平台能够稳定且长期地存储数据，且多存在隐私数据泄露的问题，即数据可靠性与安全性不容易得到保证。如百度贴吧近年来发生的大规模数据丢失事件与 Instagram 的大规模账户数据泄露事件。

许多用户有长期存储数据留档或作为备份的需求，而这有时需要数据在十数年甚至数十年间都不被损坏。考虑到目前大多数云存储平台的公司运营形式，及多数采用缺乏冗余备份的基于服务器集群的集中式存储方案的事实，要达到这样的数据可靠性几乎不可能。因此对于类似的需求，本地存储数据、自建 NAS 甚至打印纸质材料留档仍是最优先考虑的选择。

随着区块链技术与分布式存储技术的发展，已经出现了一些相关的解决方案。如 IPFS 是一个基于区块链技术的去中心化存储网络，由于其技术特性具有相比于传统云存储平台更高的数据可靠性与隐私性，但其侧重点主要在于作为区块链基础设施提供应用部署服务，并不适合作为长期存储数据的云平台。

此外，有一些老牌的 P2P 协议也具有相似的特性，如 BitTorrent 协议，但由于其无法保证冷门资源一直能够存在服务端提供资源下载（即“做种”），因此经常发生冷门资源无法下载的问题，也无法保证数据的长期存储。

1.2 系统意义

本系统尝试提出一种基于区块链技术的隐私云存储方案，使用类似比特币账本服务器的去中心化方式部署文件服务器，使用积分激励系统鼓励用户提供文件服务器并以“挖矿”形式获取激励积分。（这里的“挖矿”仅为代称，并非目前国家规定属于违法行为的虚拟货币挖矿行为，仅用于指代通过提供文件服务器获取激励积分的形式，获取的积分不可转账交易，因此杜绝了虚拟货币炒作的可能。为了提供一套成本较低的可行云存储方案，激励积分炒作本身也是系统需要杜绝的可能之一）

文件将以冗余备份形式存储于文件服务器网络中，并以碎片化形式加密存储，且尽可能存储在多个物理位置分布较远的服务器中，因此尽可能提高了存储的可靠性与安全性。

为了适应各国法律，系统具有内容审查机制。对于有内容审查需求的国家地区，提供对本地区文件上传与下载请求的拦截审查权限。考虑到数据的长期存储需求，因内容审查被禁止下载的文件实际上不会在网络中被删除，只是陷入冻结状态（见下文详述），可供后续申诉找回。

同时系统提供子网络机制，供第三方开发者开发与部署应用，将子网络作为应用服务器构建服务。

1.3 研发现状

目前国内外对于区块链存储或去中心化存储这一领域的研究都较少，国内则尤其缺少，国内外相关综述都十分少见。与之相近的分布式存储领域也仍处于起步阶段，相关技术如 Dynamo、Cassandra、BigTable、HBase 等应用仍处于较初步的阶段。过去存在的近似技术如 P2P 技术和基于 P2P 的 BitTorrent 协议尽管已得到广泛应用，但其核心需求与本系统尝试解决的问题并不相符。

目前区块链存储的应用服务也比较有限。如上文已经提到的 IPFS 作为较老牌的去中心化存储服务网络存在一定的应用场景，根据 CoinMarketCap 数据，其当前市值约十亿美元，考虑到虚拟货币市场长期存在的泡沫现象，这实际上并非一个较大的数字。同时 Arweave、Storj、Siacoin 也为该领域较成功的项目，其

当前市值均大致在 1 至 2 亿美元左右。其中 Siacoin 的主要目的是基于 P2P 网络铺设应用服务，与本系统尝试解决的问题不太相关。而 Arweave 与 Storj 更注重长期存储服务。其中 Storj 以月租形式存储数据，聚焦于低门槛、低成本、以碎片化方式替代冗余存储。而 Arweave 一开始便聚焦于一次性付费的永久存储服务，然而由于其代币经济模式与抗审查机制，存在潜在的炒作风险，且不容易大规模推广，同时也欠缺承担应用服务器功能的能力。

第二章　系统分析

2.1 可行性分析

系统考虑采用开源方式开发，以基金会形式运作，且以里程碑形式推进。由于项目开发本身对时间周期没有硬性要求，因此采取开源方式开发应当是可行的。

考虑到上文提到的数据长期隐私存储需求客观存在，该系统应当具有一定的应用前景。目前基于区块链或其他去中心化技术的存储网络的应用场景大多集中于 Web 3 领域，如 Arweave 当前常用于 Solana 链上的 NFT 存储，而 IPFS 常用于区块链 DApp 部署如 Uniswap 去中心化交易平台的部署。将应用场景局限在 Web 3 领域实际上大大限制了相关技术的发挥，但相关项目普遍采用的代币经济模式又很难使其走出这一僵局。

本系统尝试构建一套不基于代币经济模式的激励系统以构建网络，并主动引入内容审查机制以适应各国法律法规，应当具有一定的应用前景，有助于打破目前由各大云服务存储巨头如 Amazon AWS、Microsoft Azure 等垄断的市场局面。

2.2 需求分析

2.2.1 功能性需求

1. 登录及身份认证：

   用户分为两种，认证用户及匿名用户。

   a) 匿名用户：仅能提交最基本的上传/下载文件请求，并且不能够使用积分。匿名用户发送请求的处理优先级将低于认证用户。

   b) 认证用户：匿名用户可以通过受基金会认可的第三方 DID（去中心化身份）服务进行身份认证，以后也将通过认证使用的 DID 服务进行登录。认证用户将可以使用积分处理请求，并且会定期获得一定的积分奖励。

2. 文件机制：

   a) 上传：用户可以提交文件上传请求，在被服务器处理后就将存在于网络之中。

   b) 下载：用户可以提交文件下载请求，并直接从最近且存有该文件的服务器中下载文件。

c) 加密：用户在上传文件时可以选择加密/不加密，也可标记为私有/公有。私有文件将不会直接加入公共文件索引之中，并仅会出现在用户的私有文件索引之中。

d) 共享：用户可以将私有文件通过指定链接进行共享。公有文件则可从网络中直接访问，不需要共享。

e) 冻结：长期未接收到下载请求的文件将会暂时冻结。冻结后的文件将以高度压缩的形式分块冗余存在于网络之中。冻结后的文件仍可下载，但冻结后第一次下载该文件需要额外提交一次合成文件的请求，合成之后文件将以原有形式存在于网络之中。同时因内容审查而被禁止下载的文件也会被冻结在网络中

3. 文件服务器提供：

a) 任何用户均可成为文件服务器提供者。服务器主要分为专业服务器与业余服务器两大类。两种服务器都通过成功处理附加积分的请求（下述）而获取积分。其中业余服务器仅需保持在线率高于 60%，响应率大于 80%即可正常获取积分。专业服务器将获得更多积分奖励，然而这类服务器运行之前必须先将一定量的积分锁入服务器才能被网络识别，并将在正常运行十二个月后才能取出，专业服务器所获得积分也必须要六个月后才能提取。此外专业服务器每次处理请求时若出现延迟及错误，将会扣除一定量的初始积分，初始积分归零后服务器将不再被网络识别。此举是为了保证网络的稳定性。

b) 服务器提供者将以更高的基础比率获得定时发放的积分（下述）。此举用于鼓励任何用户在提供带宽以促进网络运行，即使他们计算机的性能不足以同专业服务器竞争，但仍能够增加网络的安全稳定性。

c) 为了防止服务器只处理附加积分的请求而忽视未附加积分的请求，服务器将永远保留一定带宽用来处理未附加积分的请求。该比例将由网络拥堵情况自动分配，并且具有一定最小值。

4. 激励系统（积分）：

a) 用户提交请求时可以为请求附加积分（或委托其他文件服务器提供者，附带积分的请求将被服务器更快处理，而这些积分大部分将被文件服务器提供者赚取，仅有 0.5%将被基金会收取以维持其正常运行。

b) 用户可以为文件本身附加积分，并设定每次下载时将会扣除多少积分。这种形式的文件下载时会扣除一定自身积分并发送下载请求，相当于提前支付了下载请求所需积分。在文件自身积分扣除完毕后，文件将重新成为普通文件。

c) 积分仅有三种获取途径：认证用户定期领取一定积分，积分发放将以年通胀率 2.5%的方式进行；用户也可作为服务器提供者在处理请求时获取请求附加的积分；用户参与社区投票治理时能够获取一定积分作为激励。

d) 除消费自身账户上的积分外，认证用户也可委托其他文件服务器提供者附加积分代理上传或下载文件，这主要是由于系统本身不提供积分转账机制（以避免不必要的炒作），因此提供委托代理机制供自身积分不足的用户处理文件请求。系统本身不提供交易机制，需委托者与被委托者通过其他方式达成一致。

5. 内容审查

a) 系统运行将遵循各地法律法规进行。在国家或地区政府向基金会提交请求后，获得在该地区文件上传与下载时进行审查的权限，同时可选择对来自其他地区的文件下载请求进行屏蔽。

b) 受认可的政府组织同时拥有对该地区下的子网络认证的权限。可设置任何开发者子网络（下述）均需认证才可上线，并可屏蔽某个开发者子网络的访问。

6. 第三方开发：

a) 系统提供大量底层接口，以供任何感兴趣的开发者使用，如部署应用。

b) 开发者既可将应用运行于公用网络之上，也可专门将自己的服务器接入网络作为私有服务提供者。私有服务器仅处理网络中与自身利益相关的请求，或是将其作为开发者子网络（下述）的载体，此时便无需定期支付积分。但私人服务器不会获得任何积分奖励，即使用户在请求中附加

了积分。

c) 第三方开发者可以提交开发者子网络申请，开发者子网络将定期消耗一定的积分运行。开发者子网络可视为一个局域网，开发者在子网络中拥有最高权限（若所在地区已进驻政府组织进行内容审查，则拥有政府组织以下的最高权限），并且可以将自己的应用服务运行于子网络中。

7. 基金会治理：

a) 基金会由普通用户、文件服务器提供者、第三方开发者以及受认可的政府组织共同组成。活跃度较高的普通用户、信用较佳的文件服务器提供者（提供时间较长且服务器稳定性很高）、活跃的第三方开发者及任何政府组织均可申请加入基金会参与治理。

b) 基金会负责网络的基本治理与维护，包括但不限于漏洞修复、升级迭代等。

c) 基金会成员具有较高的用户权限，但基金会本身无权查看、修改、删除公共网络中任何涉及用户个人隐私的内容（受认证的政府组织具有更多受限权限）。此外，部分最高权限不被任何用户所有，例如关闭网络。

d) 基金会负责收集并发起提案，在用户投票通过后对提案内容进行实现。

e) 基金会的部分重大决议将下放至全体用户进行投票，每个用户根据其持有积分的数量以及活跃程度、贡献值将具有不同的投票比重。在十四天之内若决议获得了三分之二以上投票者的同意将会通过。

f) 基金会对子网络服务进行监督。在开发者无力支付积分后收回开发者子网络的所有权。

g) 基金会根据政府组织的要求，对特定地区屏蔽公共网络。

2.2.2 非功能性需求

1. 隐私安全性：

a) 一切上传文件在指定加密时都将通过格密码算法进行加密，以对抗量子解密算法。

b) 一切文件都将以分块形式在网络中以碎片化存储，以防止文件服务器提供者直接查看文件内容。

c) 一切账户都必须通过严格的 DID 方式登录。尽管账户本身的认证全部由第三方 DID 服务处理，网络本身并不存储账户信息。但一旦某个被认可的第三方 DID 服务出现安全事件，基金会将会立刻冻结所有通过该方式认证的账户，并强制要求通过其他方式重新认证。

d) 积分获取及使用记录将全部存储在一条独立的区块链公链之上，以保证其不可篡改。

2. 存储可靠性：

a) 一切文件将以冗余形式同时存在于多个专业服务器之中，并且会有大量索引同时存放在专业服务器与业余服务器之中。任何服务器都将保有一定带宽用于进行文件冗余。

b) 在极端网络条件下，网络优先确保文件的永久存储而非可用性。

c) 除冗余备份外，系统对分块文件使用纠删码机制以在不降低可靠性的前提下减少存储开销。

3. 一致性：

a) 系统中以冗余形式存放的任何同一文件都将尽可能保持一致。在进行删除操作后网络将快速将该文件从网络之中清除（先清除索引），以保证网络中的文件系统一致。

b) 在文件上传后该文件会以高优先级迅速复制到整个网络。

c) 在极端网络条件下，系统需做到对文件索引的快速更新，以保持文件系统一致。

4. 可用性：

a) 系统将在保证隐私安全性与存储可靠性的前提下尽可能提高响应速度。

b) 系统将尽可能减少未附加积分请求的等待时间，并尽可能减少附加积分请求的平均积分需求。

5. 可拓展性及开发人员友好：

a) 系统将提供大量底层接口供应用开发者使用，并且这些接口的调用将是易用的。

b) 系统将尽可能适应一切网络应用的底层云存储需求，从文件分享网站到文字论坛，从视频网站到综合网站，甚至交易应用。

c) 在保证高度拓展性的同时系统同时会保证其安全可靠性，尽可能减少底层云存储基于该网络的应用在该方面产生的安全问题。

d) 以该系统作为应用的底层云存储服务将不会消耗开发者太多成本，并且将能够以较低成本长久维持运行。

6. 可维护性：

a) 网络本身将是易于维护的，采用模块化方式构建。

b) 网络的底层处理方式，例如加密和传输，将是易于更新与替换的。基金会将会定期对这些逻辑结构进行修复及更新。此外基金会还会定期对加密模块进行重大更新，以适应密码学的发展。

2.3 过程模型

考虑到系统开发采用开源方式进行，系统采用敏捷模式作为主要过程模型。具体的敏捷过程模型采用 Scrum 模型。

开发初期将系统按照功能需求拆分为若干个模块，系统整体采用微服务架构，由核心团队分组开发不同模块（微服务）。开发以三十天为一个周期进行，每个周期开始前整个核心团队开会确定下一周期的目标，并在三十天内力求达成目标。系统每天发布 nightly 版本，每两周发布一个 Alpha 版本，每两月发布一个 GA 版本（系统成熟后），每半年推进一个里程碑版本。考虑到开源项目开发初期参与人数通常不多，初期时每周期确定的目标应较小。

系统每天在开发网上进行多次部署，每月在公共网上进行一次部署。功能升级采取类似 TC39 的方式，由开发者发起提案并在四个阶段通过后纳入开发计划。

第三章　系统设计

3.1 用例建模

3.1.1 用例图

根据上述功能性需求，对系统用例建模如下图所示。

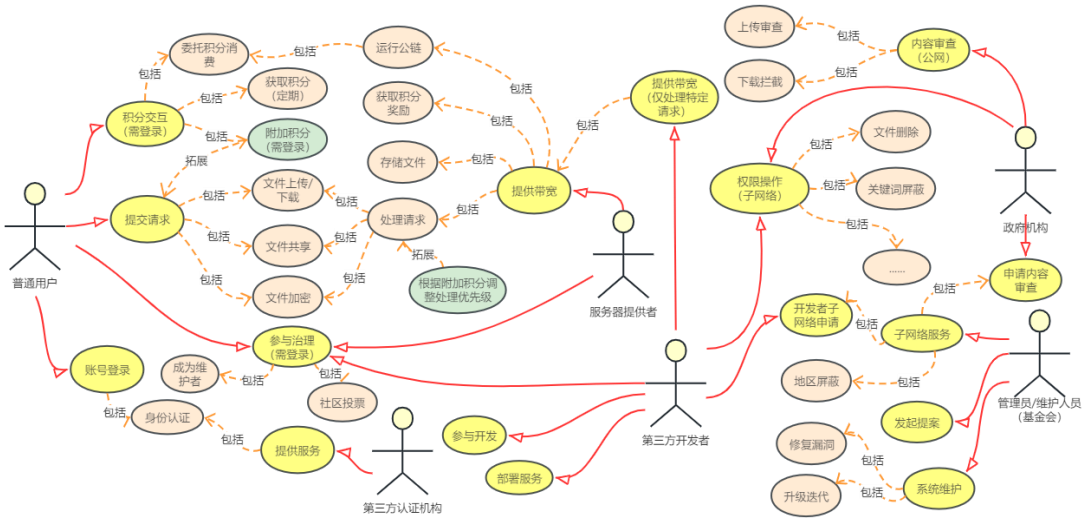

图 1 用例图

3.1.2 核心用例及用例描述

3.1.2.1文件交互

用例简述：用户进入网络后，提交文件请求，并选择是否附加积分或委托消费积分（需登录），等待处理完成。

行为者：普通用户。

前置条件：用户以登录或匿名状态进入网络，并提交请求。

后置条件：网络成功响应用户请求。

基本事件流：

① （用户需要附加积分时）用户选择使用的 DID 认证服务，点击确认，跳转到登录页面。

② （用户需要附加积分时）用户输入账户信息，点击登录按钮。

③ （用户需要附加积分时）认证服务验证账户信息，正确后将提示网络跳转到主界面。

④ 用户进入主界面，选择请求类型（上传/下载/共享/加密），选择是否附加积分或委托消费积分（需填写委托者），点击确认。

⑤ （附加积分时）系统扣除用户积分，并将请求广播到网络中。

⑥ （选择委托消费积分时）将请求发送到指定被委托者（文件服务器），由服务器确认后处理请求。

⑦ 用户页面跳转到处理页面，实时显示处理进度。

⑧ 服务器接受并处理请求，经过多次确认后完成处理。

⑨ 用户页面跳转到成功页面。用户可查看详细信息以及获取返回值（例如下载请求将返回下载链接，共享请求则会返回共享链接）

备选事件流：

① 用户输入登录信息错误，提示重新登录

② 用户中断请求处理，按当前处理进度收回一定积分（若附加）。

③ 指定被委托者拒绝接受请求或被委托者超时未响应请求，取消本次请求。

异常事件流：

① 附加积分不足/文件网络交互请求过多，请求处理超时。系统提示用户请求未成功处理，并退还一部分积分（若附加）。

3.1.2.2社区投票治理

用例简述：登录用户进入网络后，查看当前进行中的提案并进行投票，投票完成后领取激励积分。

行为者：任意已登录用户。

前置条件：用户以登录状态进入网络，进入社区投票治理页面。

后置条件：用户完成投票，并获得激励积分。

基本事件流：

① （已省略登陆操作）用户进入主界面，选择进入社区投票治理页面。

② 用户查看当前正在进行投票的提案。

③ 用户选择提案并投票赞成或反对。

④ 用户领取激励积分，同时用户页面跳转到成功页面。

备选事件流：

① 用户重复对同一提案投票，提示用户不可重复投票或选择修改之前的投票结果。

异常事件流：

① 网络繁忙，系统提示用户投票失败。

② 公链当前处理请求过多，无法响应领取激励积分的请求，提示用户之后领取。

3.1.2.3文件服务器处理文件请求

用例简述：文件服务器获取公共请求并选择是否响应请求，处理完成后获取积分奖励，处理失败则被惩罚。

行为者：文件服务器。

前置条件：文件服务器提供者按照规定方式已在公共网中部署专业或业余文件服务器。

后置条件：文件请求处理成功，服务器获得激励积分。

基本事件流：

① 服务器收到公共网文件请求。

② 服务器以一定比例选择响应未附加积分的请求或已附加积分的请求。对于已附加积分的请求按竞价模式响应。

③ 服务器开始处理请求并向请求发起者回送确认帧。

④ 请求处理过程中服务器不断向请求发起者发送确认帧以报告进度。

⑤ 请求处理完成后回送确认帧，通知请求发起者请求处理完成。

⑥ 服务器将文件分块，并与相邻服务器沟通将对不同分块的冗余备份请求发送到对应服务器的消息队列中。在此过程中服务器会接力将文件分块保存到物理位置相隔尽可能远的服务器上。

⑦ 服务器向索引服务器发送文件分块纠删码保存请求。

⑧ 文件冗余量达到一定值时，服务器将文件状态在索引服务器中标记为已持久存储。

备选事件流：

① 服务器接受请求后并未成功处理请求或未在规定时间内处理，扣除相应激励积分（仅对于专业服务器）。

异常事件流：

① 服务器掉线后未在规定时间内重新连接至系统，扣除相应的激励积分（仅对于专业服务器）。

3.2 静态建模

3.2.1 类图

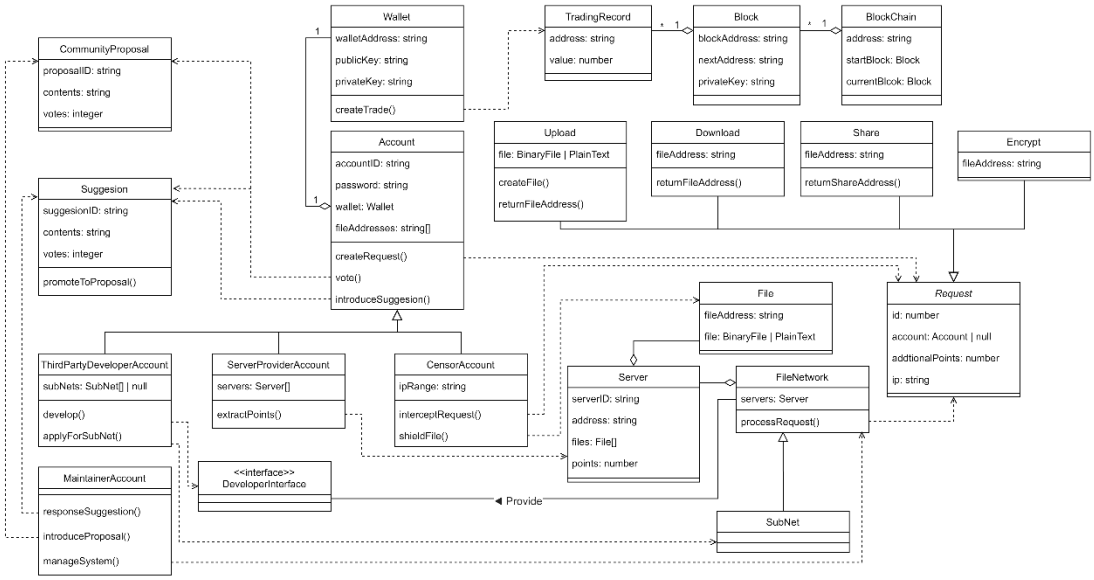

图 2 类图（静态建模）

3.2.2 方法说明

上述类图仅描述了系统的核心功能，服务器之间自身进行的冗余备份、文件冻结、更新索引服务器等功能并未包含在类图中。下面对其中一部分无法直观看出其功能的方法进行简单描述。

1. 每个账户（Account）绑定的钱包（Wallet）都具有消费/获取积分的方法 createTrade()，该方法用于领取激励积分、为请求附加积分等场景。钱包均保存在一条独立的区块链公链上以保证安全性。对于文件服务器提供者从服务器中提取积分的请求，则不使用 createTrade()方法，而使用文件服务器提供者账户（ServerProviderAccount）自身的 extractPoints()方法从对应服务器（Server）中提取积分。

2. 审查者账户（CensorAccount）具有对应能够审查的 IP 范围，并可拦截该 IP 范围内的账户发送的请求（上传/下载）（interceptRequest()方法），同时

也具有屏蔽该 IP 范围内指定文件下载的方法 shieldFile()。

3. 任何用户都具有参与社区提案投票的权限（Account 的 vote()方法），也可提出建议（Suggestion）待维护者确认后转换为提案（introduceSuggestion()方法）。而维护者负责确认或拒绝建议，并将确认的建议转换为提案（responseSuggestion()方法和 introduceProposal()方法），维护者也可自己直接提出相应提案供社区投票。

3.3 概要与详细设计

3.3.1 包图

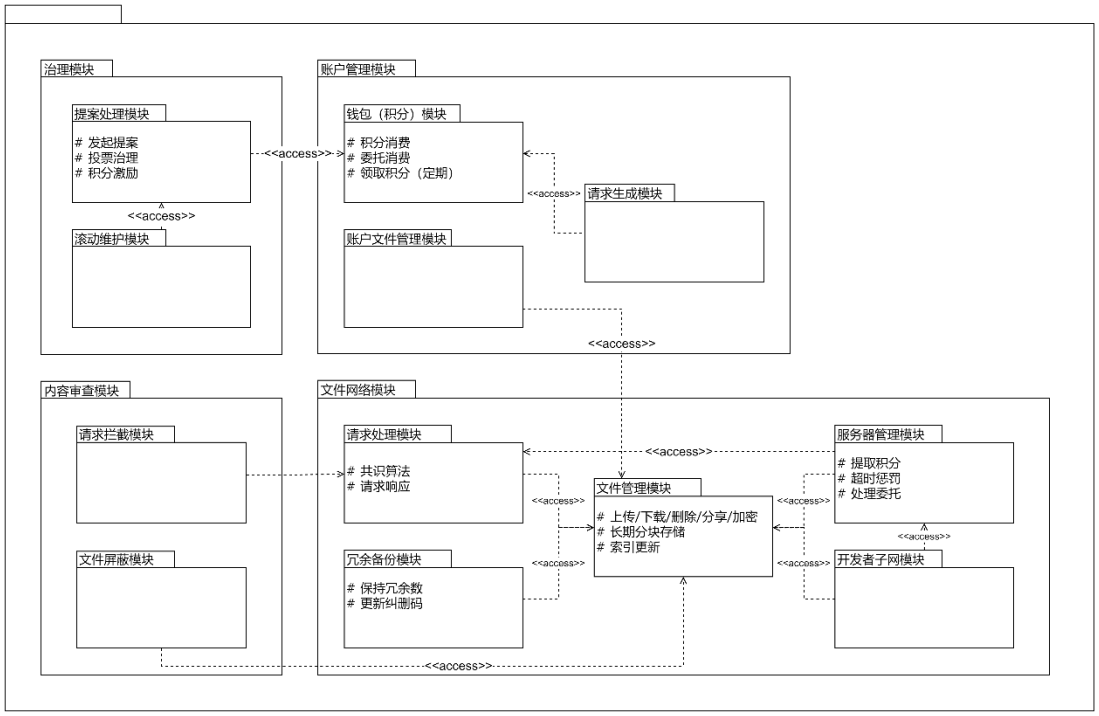

图 3 包图（概要设计）

3.3.2 核心模块详述（动态建模）

3.3.2.1文件管理模块

模块描述：文件管理模块为文件网络模块中的核心子模块，与其他各模块交互，负责文件的长期分块存储、加密、上传、下载等功能。

下面通过状态图简要描述文件网络中文件状态的变化情况。

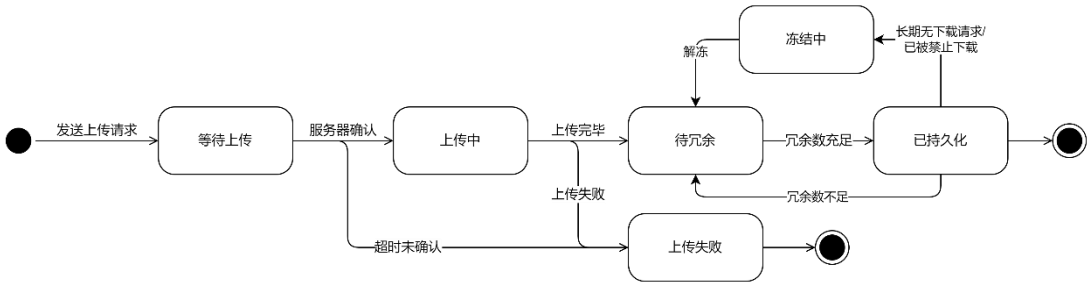

图 4 文件状态变化（状态图）

下面通过顺序图描述文件的分块冗余流程。

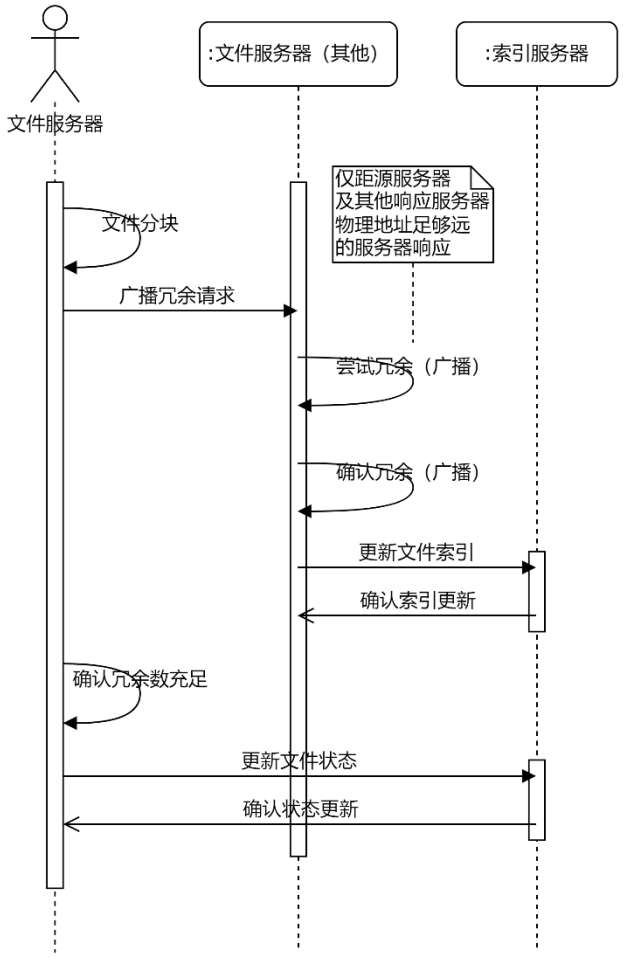

图 5 文件分块冗余流程（顺序图）

3.3.2.2提案处理模块

模块描述：提案处理模块负责整个系统的社区治理，包括采纳建议、引入提案、投票治理、分阶段推进等功能。

下面用活动图展示一个用户提出的建议转换为提案并投票通过的流程。

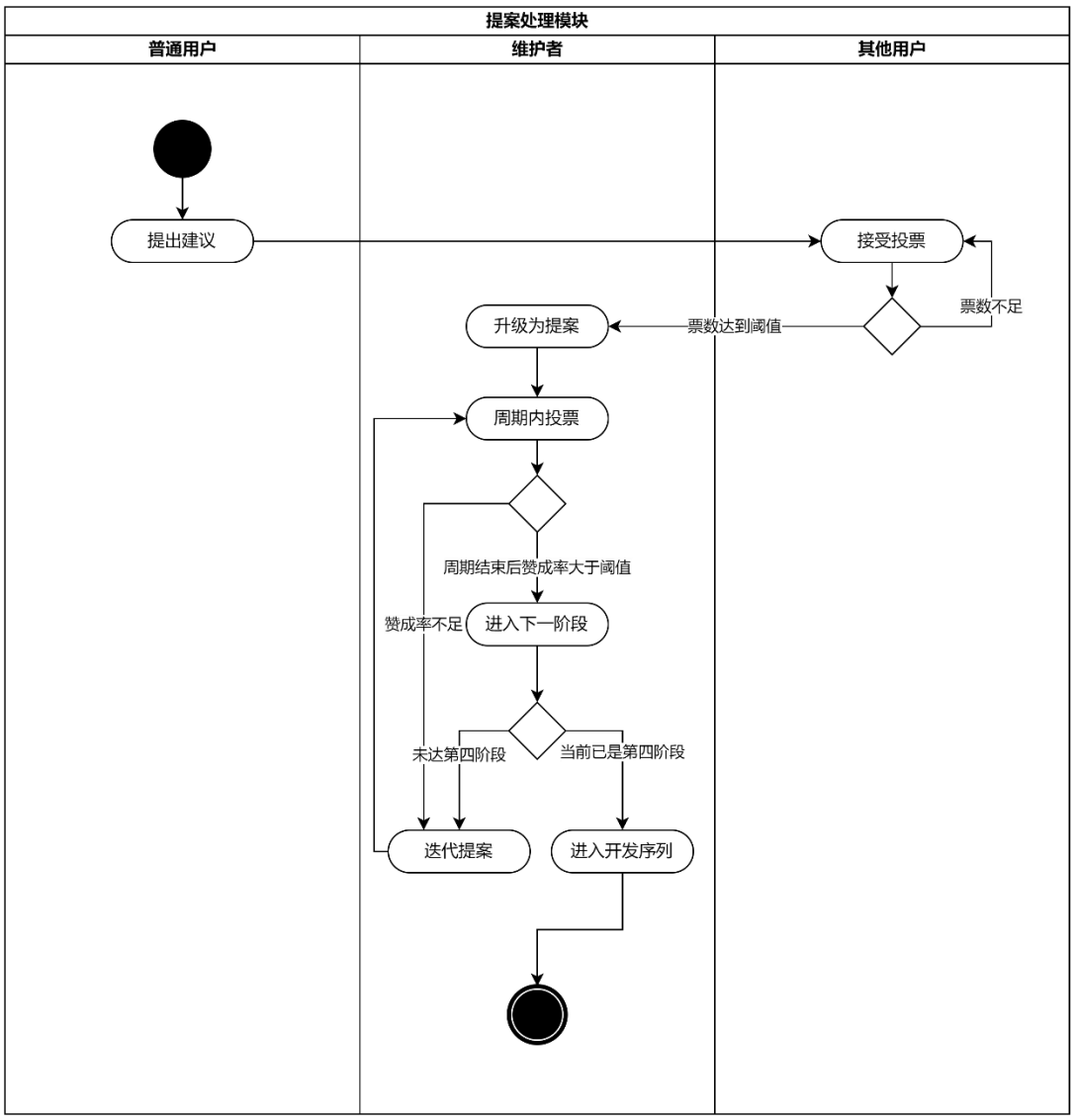

图 6 提案处理流程（活动图）

第四章　系统实现

4.1 程序设计语言选择

由于本系统提供的是较低层次的文件服务，因此对系统性能有较高的要求，因而选择如 Python、Ruby、JavaScript、Lua、LISP 方言（如 Racket、CLisp）、Basic 等解释性语言开发不能够满足需求。同理，一些抽象层级较高的语言如 Haskell 和 OCaml 等函数式语言也因性能原因而不再考虑。

其次，由于项目性质为开源项目，并且希望提供一套通用平台，因此需要尽可能摆脱平台与生态依赖，因此 C#同样不在考虑范围内。

再次，考虑到系统需要长期保持文件存储，且需要尽可能降低服务器提供者的门槛，因此需要减少内存占用，因而 Java 以及基于 JVM 的所有语言（包括但不限于 Scala、Kotlin、Clojure）不在考虑范围内。

再然后，考虑到系统对安全性有极高的要求，需要尽可能保证系统的稳定可靠，因此较易产生内存泄漏问题的 C++也不在考虑范围内。

最后，在主流语言中，还剩余 Go 与 Rust 作为可选项。在近主流语言中，还存在 Erlang（及 OTP 框架）与其继承者 Elixir（及 Phoenix 框架）以极低的响应时间可供考虑。

在这几门语言中，Rust 最为合适，因其性能最高，使用无垃圾回收的所有权机制，无“世界暂停”问题，且语法要求严格，代码可维护性很强。而其开发效率较低的缺点由于系统开发本身无明确时限，因而不算什么大问题，所以选择 Rust 作为底层模块的主要开发语言。

而对于更贴近应用层的模块，为提升开发效率，考虑使用 Go 或 Elixir 作为开发语言。

这几门语言都提供了官方的代码格式化工具，因此编码规范统一遵照官方标准。

此外，对于第三方开发者提供的接口计划涵盖 C++、Rust、Go、Python、Ruby、JavaScript（Node.js 环境）、C#、Java、Scala、Kotlin、Elixir 等语言。

对于较边缘的应用服务，考虑使用 Ruby 构建 DSL 作为推荐配置脚本，由于 Ruby 官方并未提供编码规范，使用 Rubocop 和 YARN 作为推荐编码规范。

对于文件传输协议的开发，可能需要开发专用 DSL 以提高效率。这里倾向于

参考 LP（逻辑式编程语言）如 Prolog 设计相关 DSL，这在一些图数据库查询语言已有先例，如 Cypher、SPARQL、Datalog 等。

4.2 实现方式

系统采用模块化方式实现，采用微服务架构构建系统。具体模块已在上文概要设计中给出，这里不再详述。

系统不使用任何第三方存储解决方案，如数据库。为最大程度匹配文件网络性质，且为了实现尽可能高的性能，文件存储及传输模块均由开发团队自己完成。此外，文件传输协议也需项目独立实现，而不能使用已有的文件传输协议。但对于数据交换格式，可使用已有的协议。考虑到系统对吞吐量有较高的要求，考虑使用 Protocol Buffer 作为主要的数据交换格式。对于功能与系统耦合度较高的请求，考虑使用独立的 DSL 作为请求体（以 Protocol Buffer 形式）。

此外，考虑到项目性质，应在项目初期便搭建完成持续集成及持续部署服务。每日在测试网中进行数次集成与部署，并且每月在主网中进行一次滚动升级。

项目将使用 Git 作为版本管理工具，并开源至 Github。

第五章　系统质量保障

项目应有独立开发的代码评审工具，以尽最大可能统一代码规范，在编程语言自身编码规范较模糊之处做出进一步规范。代码评审工具应能够检查出一定数量的潜在错误并禁止某些编码方式的使用，如每个函数的圈复杂度都应低于 11。代码每次提交到仓库都进行一次自动化代码评审，若不通过则退回提交并要求修改。

在单元测试上，推荐各开发组（以模块划分）以 TDD（测试驱动开发）模式进行编码，但不强制推行，以尊重开发者自身习惯。

规定每次推送到仓库的代码必须包含单元测试，并保证 95%以上的代码覆盖率，否则将回滚并要求修正后重新提交代码。考虑到系统对可靠性要求较高，单元测试采用的标准为修正判定-条件覆盖（Modified Condition/Decision Coverage, MC/DC），测试用例应基于该标准编写。

每日进行一次代码走查，三至四人一组，约半小时。

每周进行一次代码审查，将整个开发组按职责分割成若干小组来组织代码审查会议，约两小时。

除开发人员自己编写测试外，还应包含一个独立的测试团队，为每日提交的代码编写黑盒单元测试。黑盒测试采用边界值分析法与等价类分析法结合构建测试用例，对于关键代码逻辑应使用因果图结合判定表构建测试用例。其中边界值分析法采用健壮边界值测试，等价类分析法采用弱健壮等价类测试。除此之外，对于较重要但非关键的代码逻辑，使用 Pairwise 方法测试。

在项目的每个 Alpha 版本发布前都应由开发人员和测试团队共同进行一次集成测试。集成测试采用三明治式方法进行。测试用例的编写仍按照上述标准进行。

在每次公网的滚动升级之前，都应于测试网上进行系统测试，重点测试系统可靠性、性能、跨平台性以及安全性，并给出相应的测试报告。

由于系统对部分非功能性需求有较高的要求，因此这里给出具体指标。对于文件传输请求，系统在正常情况下应能够处理不低于 100k requests/sec 的并发请求，该数字应能够随着网络规模的扩大而线性或近似线性增加。由于可用性并非系统关注的核心要点，因此要求不高，但中位数响应时间也应不长于 1s，平均

响应时间应不长于 3s，而 99.9%的请求响应时间应小于 10s，且至少在 99.99%的时间都要达到上述服务指标。

在系统第一次正式上线前，应进行多次完整的验收测试，包含在α测试网上进行的α测试和在β测试网上进行的β测试（向第三方开发者开放的测试）。由于本系统设计为开源项目，因此应充分利用开源社区为项目进行测试。上线前计划进行半年至一年的开放验收测试（β测试）。在主网上线完成后只进行上述单元测试、集成测试、系统测试与下文将提到的回归测试，不再进行验收测试。

此外，每次修改代码后都应对相关部分进行一次回收测试。回收测试主要考虑通过自动化测试工具完成，不应有太多人工介入。

第六章　项目管理

6.1 人员配置

为减少沟通成本，项目核心开发团队控制在八十人之内，按照模块分成不同小组，同时设置一个二十人左右的测试团队。除核心开发团队外，仓库也接受其他贡献者的代码，将存在一个五至六人的团队负责整合代码、回应 issue、吸纳提案，并评审其他开发者贡献的代码，以考虑集成到仓库中。

对于每个模块，设置一位团队负责人领导模块开发，一至二位副管理者协调团队，一位技术专家进行指导，两至三人主要负责修复缺陷，剩余团队成员均为一般开发人员。每位开发人员同时担任开发者、开发文档编写者及单元测试编码者的职责。每两位开发人员结对编程，同时负责一个核心功能的编码。

在此之上，设置一位核心开发团队的总负责人以及三位副手进行协调。对于较重要的决策，由团队共同投票决定，且将定期吸纳社区提案加入开发计划中。

6.2 成本估算

本系统作为开源项目对人员工资的考虑较少，且不需要购置外部服务，但仍需一些开发者全职工作以协调团队，因此需要在对这些开发者有一定的经济补贴。此外，项目测试及初期上线需要较大的服务器投入，而这同样需要一定量的资金支持。

首先在对全职开发者的经济补贴上，考虑约十人作为全职开发者，以平均每月$1,500 的补贴计算，每年需要$180,000 的资金。

根据下一章系统部署中的方案，初期在世界各地部署五十台专业服务器，每台服务器按$6,000 计算，需要$300,000 的一次性投入。

综合来说，计划四年的开发周期后初步上线主网，需要共$1,020,000 的资金支持。这不是一笔小数字，但仍是有可能集齐的。如 Y Combinator 这样的创业孵化器每两年对大批初创公司投资$500,000，在四年内集齐所需资金虽然比较困难，但并非没有机会。

除通过创业孵化器募集资金外，区块链企业如一些知名的大型交易所也定期发放行业基金以激励业内项目，如 Binance 在最近设立了 20 亿美元的行业复苏基金用于资助遭遇资金困难且有技术实力的区块链项目。本系统应当可以考虑从

类似的资助基金中筹集开发资金。

6.3 进度管理

由于系统自身性质，几乎不可能出现产品需求变更，因此考虑在项目初期便安排大致进度。然而由于系统采用敏捷模型开发，进度计划不应过细，只应设置几个主要的里程碑节点，包括存储模块开发、网络协议开发、冗余备份模块开发、积分激励模块开发等几个主要的时间节点。在实际进行每个里程碑的开发时，再进行项目进度细分，且在这一步也不应过细，只做大致划分，细节功能的开发由Scrum过程模型不断设定每月的开发目标并冲刺开发。

当某个里程碑未按照预定时间发布时，不应尝试投入更多人手或以赶工的形式补救，而应立刻重新指定项目计划并重新考虑上线时间。

在每一Scrum周期的开发过程中，应确保每对开发者（上文已提到采用结对编程方式）只聚焦于一个核心功能点开发，不应包含任何额外任务。

第七章　系统部署

在四年的初步开发周期结束后开始初期网络部署。初期计划于世界各地投入五十台专业服务器构建主网，其中十台服务器作为专门的索引服务器，剩余服务器作为普通文件存储与冗余备份服务器。

除项目开发团队（基金会）自身投入的专业服务器外，也鼓励开源社区参与其中，贡献专业与业余服务器参与网络构建，并获得积分激励。由于系统本身存在激励机制，在基金会初期部署起主网后，若推广得当应不断有新的文件服务器加入构建主网。随着网络扩大与使用人数增加，各地政府组织也应逐渐介入其中，基金会将按照流程为各地政府提供符合当地法律法规的内容审查服务。

对于基金会官方设置的专业文件服务器，将在世界各地分成五个机房部署，即每个机房十台服务器。每个机房由开发团队中的二至三名当地志愿者负责维护。

参考当前较为成功的去中心化存储项目 Arweave，其于 2020 年十月上线，当前节点数量约一千余个（实际显示数量为 67 个，但考虑到一个矿池只计算一次，实际节点数量根据矿池数据统计约一千余个）。由此推广，考虑到本系统有更强的通用性，在理想情况下应能在初期部署后的两年内达成近万台专业服务器参与构成网络。考虑到业余服务器的低门槛，以五只十倍业余服务器数量计算，将有约五至十万台业余服务器参与构成网络。当前 Arweave 主网已经能达到不错的永久存储效果，本系统在这样的服务器规模下应当足够维持数据至少一百年不丢失。

系统总体上采用 C/S 架构部署，但服务器之间不分主从服务器，而是按照 P2P 方式连接。在对应的索引服务器、专业文件服务器以及业余服务器上都安装对应的服务端软件。计划全部使用 Linux 系统部署索引服务器及专业文件服务器，但对于业余服务器提供 Windows、Linux 以及 Mac OS 端的服务端软件，以降低门槛并鼓励用户参与网络构成。

服务端软件同时配有相应的部署脚本，可通过 Ruby 作为 DSL 进行一定的自定义配置。如业余服务器可配置自身的大致在线时间、带宽上限等以调整激励积分的发放。

对于客户端，基金会官方将发布相应的命令行工具及使用图形化界面的桌面软件，以供用户在网络上进行基本操作。由于系统集中于提供基础的云存储服务，

因此桌面应用的开发并非重点，仅负责提供基本功能。功能较全面精致的桌面应用、移动端应用等应由社区开发完成。

对于第三方开发者，可向基金会申请创建子网以部署自身的应用服务。子网可通过使用积分租用主网资源，也可开发者自己将服务器接入子网以较少积分维持子网运行。

除此之外，第三方开发者还可在主网中接入私有服务器以提高主网中与自身利益相关的请求处理速度，此类服务器不会获得积分激励。

对于所部署应用流行度较高的第三方开发者，将发放激励积分甚至免除维持子网所需的积分，以激励第三方开发者将应用部署至系统中。考虑到系统本身存在一些与 Serverless 系统相似的特性，应当具有足够的吸引力。

第八章　系统维护

系统主网上线后，维护主要由基金会负责。如上所述，基金会自身在每次请求中抽取 0.5%的积分作为主要盈利方式，次要盈利方式为接受企业以及其他开源组织资助。

基金会维护人员将定期修复社区提交的 BUG，并在下一次滚动更新中进行修复。对于严重 BUG，基金会将投票决定是否立即进行网络的滚动升级。

除系统自身的缺陷修复外，基金会还将定期吸纳社区提案并加入开发计划。这将采用类似 TC39 的方案，由社区提出建议（Stage 0），在建议获取足够票数后被基金会采纳升级为提案（进入 Stage 1）。每个阶段的提案具有投票周期，若提案在周期内收到足够的票数，且赞成率大于一定阈值，则进入下一阶段，并在提案进一步完善后开启下一阶段的投票周期。

除 Stage 0 外，提案共分为四个阶段。Stage 1 为被基金会注意到且刚从建议升级为提案的基本提案，为草稿性质的提案；Stage 2 为较完善的提案，已经包含了较为完整的计划及解决方案，但仍可能与最终提案相比有较大改动；Stage 3 的提案基本稳定，若非存在致命缺陷不会与最终提案相比出现大的变动；Stage 4 的提案为已经被基金会完全采纳，且已进入开发计划的提案，最终实现将基于该提案，且最终实现与提案本身之间仅可能有微小的改动。

除 Stage 4 外任何阶段的提案都有可能被关闭。这可能是由于已经有相似提案被采纳，因此无需该提案解决同样的问题，或发现提案自身存在重大缺陷，需要重新发布新提案，或提案发起者自己主动关闭提案（仅限 Stage 2 及之前的提案）等。

当提案被基金会正式采纳后，将在三月内被集成到测试网中，并在半年内通过滚动升级被集成到主网中。

除基本的缺陷修复及依据社区提案的系统升级外，系统本身还会根据密码学的发展定期升级加密方式。有一部分特殊的社区提案为对系统使用加密方式的改进，基金会将定期评估这些加密方案的有效性，并视情况升级系统使用的加密方式，以对抗密码破译方式如量子解密的进步，以最大程度地预防隐私泄露。

除上述改正性维护、完善性维护及预防性维护外，随着主流操作系统、硬件设备等升级，基金会将定期更新服务端及客户端软件版本，以确保它们始终能够

在最新的软硬件平台上运行。

对于每次改正性维护（缺陷修复），都将在缺陷跟踪系统中记录缺陷修复过程。对于每次系统的滚动升级，都应编写完整且详尽的升级日志。

在系统整体趋于稳定之后，基金会将在社区中开展若干后续项目以对系统进行持续重构和进行周边配套设施的开发。如对于系统早期开发过程中因考虑不周留下的缺陷，将开设新的代码分支，尝试从开头重构系统，并在成熟且经过完整测试后将代码纳入主代码分支中。又如对于系统周边设施的完善，启动多个社区项目以致力于简化服务端软件部署、提升客户端使用体验等。

第九章　系统总结

9.1 工作总结

总的来说，本文尝试构建了一个基于以激励机制驱动的去中心化网络的开源云存储系统。其主要目的是实现数据在平台上的永久存储，与实现数据存储的高度隐私性。在此基础上，本系统尽可能提高系统的易用性，以为第三方开发者提供部署应用服务的基础平台。

为了避免数据的集中化存储，本系统采用积分激励机制，通过大量分布于世界各地的专业与业余服务器构建去中心化存储网络。同时以分块冗余备份和纠删码方式保证数据在网络中的持久存储，并以加密方式保证数据在网络中较好的隐私性。

本系统为开源项目，并由开源基金会负责系统的开发与治理维护。同时为适应各国的法律法规，本系统允许各国政府介入网络进行内容审查，同时基金会本身也会协助政府机构提供文件屏蔽、请求拦截审查等服务。

9.2 知识产权保护

本项目作为开源项目采用 GPL v2 协议，以强制任何使用本项目代码的系统必须将代码开源，以此来保护项目的知识产权。系统不采用 GPL v3 协议的主要原因是考虑到项目体量较大，若后续可能需要修改协议很难征得所有开发者的同意。

当发现闭源商用系统违规使用本系统代码时，基金会有权根据协议起诉对应企业。

在中国大陆，基金会将在征得全部开发人员同意的前提下申请软件著作权以及版权保护。由于开源软件无法在公开源码后申请专利，因此对于部分关键技术实现，考虑在申请专利成功后再进行开源，该行为同样需要征得全部开发人员统一。

为开源软件申请专利及版权保护有助于避免不必要的纠纷。近年来发生多起开源技术被其他企业申请专利，反被索赔的案件发生。由于这些软件在开发初期通常未选择合适的开源协议，且忽视了对自身版权的保护，法院在知晓实际情况的前提下仍很难做出应有的判决，导致了不少麻烦的发生。

9.3 系统推广

本项目选择开源模式本身便充当了实质上的推广。Github 等代码托管平台上星标数的增长已经自然起到了推广的作用。

除通过开源效应自身推广外，基金会本身将抽出一部分资金用于在存储技术峰会、大数据峰会、物联网峰会、区块链发展峰会等技术会议上进行推广，并且将在相关技术媒体及信息整合平台如 CoinMarketCap 上进行推广。

推广应集中于区块链相关峰会，如中国大陆的全球产业区块链峰会、中国区块链技术产业发展峰会和中国产业区块链大会等。除大陆地区外，在亚洲主要集中在香港和新加波这两个区块链技术发展较为活跃地区的相关技术会议上进行推广，如香港的区块链周、Web3 创新者峰会，新加坡的亚洲 Web3 会议、Blockchain Fest Singapore 等。在欧美地区，将尝试在如英国的区块链经济伦敦峰会等会议上进行推广。

考虑到区块链本身的技术敏感性与自身炒作较为严重的现状，项目本身不应投入过多精力主动推广，而应主要依赖于开源项目自身的推广作用，否则有被视为炒作甚至被相关法律法规误伤的风险。