**YottaChain黄皮书**

1. 概念和符号

**YTFS**

YottaChain File System的简称，是YottaChain使用的去中心化存储系统。

**YTChain**

YottaChain主链，是基于EOS改造的。

**超级节点**

YottaChain采用EOS架构，但主超级节点和超级节点除了完成记账、运行智能合约等功能外，也承担存储管理、密钥管理和P2P管理的职能。超级节点承担运营YottaChain的重任，由利益相关方担任，并有相应的激励，属于可信节点。

**数据节点**

即通常说的存储矿机，存储各数据分片，要求7x24小时在线，并抵押一定的YTA作为保证金。YTFS设计支持至少1亿个数据节点。

**用户端**

即存储用户，付费在YottaChain上存储数据。用户端不要求7x24小时在线。

**HDD**

YottaChain中代表硬盘存储空间的资源通证。资源通证的定义和解释见白皮书。矿工挖矿得到的是HDD（需要兑换成YTA才能流通），用户存储数据需要耗费的也是HDD。

**去重**：

去掉重复数据的简称，也称为重删（重复数据删除）。YottaChain规模足够大的时候，去重率有望达到10倍以上。

**S3**

AWS的对象存储服务，已经成为行业事实标准。支持S3协议即可认为支持对象存储服务。

**数据块**

一个文件（或对象）在压缩后分为多个定长的数据块（最后一块的长度不固定），YTFS以每个数据块为单元进行数据去重处理。

**数据分片**

加密后的数据块分为多个定长的数据分片，通常是用一种冗余编码算法进行编码而产生，数据块过小时也可能不进行冗余编码计算。

**记账周期**

计算存储费用和矿工收益的时间间隔，例如1小时。

**更新周期**

计算每周期费用（定义见后）、每周期收益（定义见后）的时间间隔，例如1天。

**每周期费用**：

每个记账周期耗费的存储费用，以HDD为单位。与他人重复的数据分摊计算，每周期费用包含小额的一个元数据基数PCM，以鼓励用户尽量存储大文件，避免过多小文件消耗系统资源。

**每周期收益**

矿工在每个记账周期获得的HDD奖励，依据预采购空间的总和计算。

**MVP**

第一版上线测试的最小实现。MVP上线后逐渐迭代。

**符号命名原则**

A开头代表算法，其中AH开头代表Hash算法

SN开头代表超级节点

D开头代表数据库

K开头代表密钥

P开头代表系统配置参数

V开头代表流程执行过程中计算出来的值，其中VH代表Hash值

**数据库**

DUD：用户文件去重表，每个用户对应的超级节点SNU存储该用户所有的文件的数据信息，包含用户ID、VHW、VNU、引用次数、每个文件的所有数据块的密钥列表。

DFD：文件去重表，记录所有文件的数据块信息，包含VHW、引用次数和所有数据块的列表。该表可以全网同步，所有超级节点都有。

DB数据块表，记录每个数据块的信息，存储在该数据块对应的超级节点SND上。

DFN：数据分片表，记录每个数据分片的存储位置，存储在该数据分片所在数据块对应的超级节点SND上。

DMI：屏蔽索引表，每个监管区都有一个DMI，记录被该监管区屏蔽的文件的VBI索引。

DNF：新文件列表，记录每个用户刚上传的还没有达到最短存储时间的文件，存储在该用户对应的超级节点SNU上。

DNI：数据节点索引表，记录每个数据节点所保存的所有数据分片的Hash值VHF

**不可调整系统参数或算法**

AEY：YTFS采用的对称加密算法，例如AES-256。

AHE：密文Hash值算法，计算VHF时使用，须与AHP不同，对Hash值的长度敏感，例如MD5。

AHP：明文Hash值算法，计算VHW和VHP时使用，例如SHA2-256。

AHK：计算去重密钥使用的Hash算法，与AHP不同而且输出长度与AEY的密钥长度相同，例如SHA3-256。

PBL：数据块上限，取值可能在256K-2M之间，必须是数据分片大小PFL减1的整数倍。

PFL：数据分片大小，取值可能在4K-32K之间，必须是4K的整数倍。

**可调整系统参数或算法**

AP：压缩算法，要求具备确定性（相同的数据压缩后的结果一定相同，去重需要），可分块解压（可以不依赖之前的块而单独解压任何一块）。原则上新的压缩算法只会在新文件使用。

AR：冗余编码算法，例如RS。

PFR：矿池最高容忍故障率，例如5%。只要一个矿池所属的存储数据节点的同时故障率低于PFR，则不受惩罚（可能要扣除很小额的数据重建费用）。

PL1：数据块进行冗余编码的长度阀值，例如16K，高于该阀值用冗余编码。

PL2：数据块直接保存在数据库中的长度阀值，例如256字节，低于该阀值数据块直接保存在数据库,介于PL1和PL2之间用副本方式保存。

PMF：每个数据节点的索引项是分组的，按照数据分片的Hash值VHF来分组，PMF是每一组可以容纳的索引项数量，该值略大于该数据节点最大存储容量/PFL/分组数。当一组的索引项满了后，将无法再保存该组的新数据，即溢出。

PNB：系统每次预采购的数据分片数量。在每个数据节点新注册或预采购的空间存满后，系统都会增发HDD预采购PNB个数据分片。

PND：一个数据块内最多容许掉线数据节点的数量。例如取值16，代表同一个数据块允许 16个数据分片丢失也不会影响数据可靠性。

PNF：用户端发给数据节点的数据没通过验签和Hash验证时的重试次数，超过该次数都失败证明该用户端在作恶捣乱，将被惩罚。

PNR：重建一个故障节点的数据的节点数。例如当PNR=1000时，就是1000个节点同时重建一个故障节点的数据，每个节点重建0.1%的数据。

PPC：记账周期，每周期费用和每周期收益就是以记账周期为单位计算存储费用和挖矿收益的。

PMS：最短存储时间。每个文件在上传后必须存储的最短周期，例如60天，以避免频繁存储删除数据导致的系统负载过重。

**值**

KD：去重用的对称密钥，由数据块的明文计算而来。

KED：去重用加密存储密钥，用去重密钥KD加密存储密钥KS得到的值。

KL：导出授权时加密用的随机对称密钥。

KEL：用被授权方公钥加密KL得到的值。

KEU：用户加密存储密钥，以用户存储公钥KUSp加密存储密钥KS得到的值。

KS： 存储密钥，用于加密各数据分片的对称密钥，随机产生以保证在产生之前没有任何人知晓。前192位均为0的存储密钥保留为特殊用途。

KUEp：用户加密公钥。

VBI：每个数据块的“VHPVHF列表”组合的唯一序号，全网唯一。VBI中包含了该数据块所在的超级节点编号，可以依据VBI找到其对应的SND。

VHB：一个数据块的所有数据分片的Hash值VHF的列表的Hash值。

VHF：一个数据分片的Hash值。一个数据块的所有数据分片的VHF列表代表该数据块的密文Hash值。

VHP：一个数据块的明文Hash值。

VHW：文件的整体明文Hash值，暂定32字节。

VND：全局文件流水号，全局文件去重表DFD中使用。这是比Hash值更短的唯一ID，而且不泄露与文件内容相关的任何信息，暂定8字节，由生成VND的超级节点ID和该节点的自增序号构成。

VNF：一个压缩加密后的数据块分成多少个冗余的数据分片。YTFS采取数据分片定长的模式，所以每个数据块的VNF可能都是不同的。

VNU：用户文件流水号，用户文件去重表DUD中使用。这是比Hash值更短的唯一ID，而且不泄露与文件内容相关的任何信息，暂定8字节，由生成VNU的超级节点ID和该节点的自增序号构成。

VPS：整个YTFS系统的物理总存储容量。

**代号**

SNF：与某文件对应的超级节点，根据文件Hash值VHW计算

SNU：与某用户对应的超级节点，根据用户ID取模计算

SND：与某数据块对应的超级节点，根据VHP取模计算

SNM：与某数据节点对应的超级节点，根据数据节点ID取模计算

## 二、总体架构：



YottaChain按照满足商业目的的最小实现的原则设计，尽量降低工程实现的难度。为此，采用EOS半中心化的架构，利用可信的超级节点管理存储元数据、密钥数据、用户数据和矿工数据，从而在充分满足商业目的（存储是去中心化的、矿工能挖矿挣钱、中心化应用能无缝迁移、数据不会丢、数据不会被偷、成本比中心化存储大幅度降低、数据可靠性比中心化存储大幅度提升）的前提下大幅降低了工程实现的难度。

整体架构设计图如上图所示，共分三种节点

1. 超级节点：负责记录区块数据、内部交易市场、密钥管理、矿工激励、存储管理用户端。

超级节点上运行YottaChain主链超级节点程序（基于EOS代码修改）、智能合约、YTFS核心服务、YTFS扩展服务和系统管控超级节点端。其中每个超级节点上运行的YottaChain超级节点程序的数据是不同的， 105个备用节点分为21组，每组与一个超级节点共同构成一个1主5备的小集群，而YottaChain智能合约的数据保存在EOS RAM上，是所有超级节点都会同步的公共数据。

其中YTFS核心服务负责维护存储元数据和密钥管理数据，管理协调存储相关任务；YTFS扩展服务提供S3命名空间等扩展存储服务；系统管控超级节点端负责发布随机抽查任务，对抽查结果的处理（确认节点是否失效，确认后发布数据重建任务并处罚失效节点）等。

2. 用户端（即普通节点）：用户端在超级节点协调下在YTFS存储网络上传或下载文件，文件的压缩加密去重分片等操作均在此完成。另外用户端暴露出通用的API可用于实现各种通用存储接口（比如S3、FTP、NFS或CIFS）。MVP只实现S3。

3. 数据节点（即商业节点）：包括YTFS数据节点端、系统管控数据节点端、中继模块等，在超级节点的协调下负责响应来自用户端的的文件上传/下载操作，抽查其它节点，重建失效节点的数据，以及为不能直接连接的设备之间建立中继通道。

**二、各组件架构：**

1. 用户端：



用户端架构如上图所示，主要实现数据的压缩加密、纠删码分片、去重等功能，并对外暴露一个标准的API，可在该API之上实现其他文件传输协议，比如S3/FTP/CIFS/NFS/S3等协议。

(1) 用户端上传流程如下：

1. 用户端利用特定的Hash算法AHP计算文件的整体明文Hash值VHW，将VHW和文件长度发给该用户对应的超级节点SNU查询是否有重复
2. SNU检查用户文件去重表DUD，如果发现同一用户已经有Hash值为VHW的文件，而且文件长度相同，则将VHW的引用次数加1，然后终止流程（不需要验证数据是否正确，因为自己不会骗自己，同时同一用户重复不需要重复花HDD）；如果不重复则SNU验证该用户是否有足够的HDD用于存储该数据最短存储时间PMS（例如60天），不够的话反馈给用户端，用户端报错返回；
3. SNU分配一个此次上传任务的流水号VNU；SNU在新文件列表DNF中增加一项，包含VHW和VNU；SNU，冻结该用户相应的HDD，在用户文件去重表DUD中增加一个新记录，填入新记录的UserID和VHW字段，引用次数NLINK为0，VNU为流水号；
4. SNU向VHW对应的超级节点SNF查询全局文件去重表DFD，如果没有全文Hash值为VHW的记录转步骤i；
5. 读出将所有文件Hash值为VHW的记录，将VNU和每一个数据块的压缩算法AP、加密算法AHY、冗余编码算法AR、原文长度、去重密钥KED、数据分片的数量VNF、各数据分片的Hash值VHF的列表的Hash值VHB（但不需要VBI）都返回给用户端；
6. 用户端验证是否有一组数据是匹配的，方法是对每一个数据块（以该数据块的原文长度切分），用指定的压缩算法压缩（见步骤i），检查压缩后的长度刚好等于每个数据块的长度PBL（最后一块除外，但最后一块的长度也不能超过PBL），如果不相等则验证下一组数据，否则用该数据块计算出去重用的对称密钥KD（见步骤j），用KD对KED进行解密得到存储密钥KS，以用户加密公钥加密KS得到KEU，暂存起来，用KS和指定的加密算法AEY对压缩后的数据进行加密得到该数据块密文（见步骤o），用指定的冗余编码算法AR对密文数据进行冗余编码，编码成VNF个数据分片(见步骤p)，每个数据分片的长度是PFL，用Hash算法AHE计算计算每个数据分片的Hash值VHF，将所有数据分片的Has值VHF按顺序连接起来，计算其Hash值VHB，比较与传入的VHB是否相同，如果不同继续验证下一组；如果所有组都验证不通过，则转步骤i；
7. 此时找到了验证通过的重复文件，用户端将VNU、VND和各数据块的加密后的存储密钥KEU发给超级节点SNU；
8. SNU将收到的VND记录在用户文件去重表DUD中VNU记录中的VND字段，各数据块的KEU记录在KEYS字段，转步骤cc；
9. 对将要上传的文件用压缩算法AP进行压缩,按照系统设定的数据块上限参数PBL将压缩后的文件分成固定大小的若干块；分块的目的不仅为了提高数据去重率，更主要是为了保证不管多长的文件都能管理；  
   压缩时采用可以分块还原的算法（例如每一块的开始将B+树清零），这样每一块都可以单独还原，不依赖前面的块。记录每一块对应的原文长度。如果压缩后某一块的长度更长了（即对应原文的长度小于等于PBL-2），则该块采用非压缩方式，或者更换压缩算法选效果最好的；  
   每一块的前2个字节代表当前块的压缩算法和末尾未压缩的字节数（为了凑PBL-2的长度，可能最后有几个字节是不压缩的），如果为0x0000则表示全部未压缩。剩下的PBL-2字节才是该块的数据；
10. 将一个数据块用hash算法AHP计算其明文hash值VHP，计算HMAC-AHK（VHP， 数据块明文），得到去重用密钥KD；
11. 记N=数据块大小/（PFL-1），结果向上取整（PFL为每个数据分片大小，之所以减1是留出一个字节记录索引号）；对大小超过系统参数PL1(例如16K)的数据块，将根据最多容许掉线数量PND用冗余编码算法AR将该块数据分成VNF个分片，其中VNF=N +PND；  
    对大小介于PL1和PL2的数据块，将采用PND+1个副本的模式，每个副本有N个数据分片，VNF=N\*(PND+1)；  
    对于小于PL2的数据块，将直接记录在元数据库中，VNF=0；
12. 用户端向VHP对应的超级节点SND查询该块数据是否已经存在，SND检查数据块表DB中是否已有VHP的记录，如果存在则SND从该记录中读取去重用加密存储密钥KED、该数据块的序号VBI、加密后数据分片hash值VHF的列表的Hash值VHB返回给用户端（在有人恶意捣乱的情况下，VHP的记录可能会有多组，每组以VBI序号作为唯一ID，其中至多一组是正确的，也可能全都是错的，所有组都一次返回），转步骤x。
13. 当系统中没有该数据块时，超级节点SND分配VNF个数据节点用于保存该数据块的各数据分片，每个数据节点保存1个数据分片。被分配的数据节点必须是有空余存储空间的（即实际存储空间小于生产空间）。SND向用户端返回每个数据节点的节点ID、网络地址和数据加密公钥，每个数据节点ID都做一次签名（连同VNU一起签名，签名时不含网络地址和密钥）；
14. 用户端生成随机对称密钥KS作为存储密钥，如果KS的前192位均为0则丢弃重新生成；用去重用密钥KD和对称加密算法AEY（例如AES256）对存储密钥KS进行加密，得到去重用加密存储密钥KED;
15. 用户端利用对称加密算法AEY和存储密钥KS对该数据块进行加密。加密采用CBC模式，IV为密钥KS加密“YottaChain2018王东临侯月文韩大光”的密文。
16. 如果采用冗余编码方式，用AR算法将密文分成VNF个数据分片；如果采用副本模式，将密文直接切分成N个数据分片；如果采用记录在元数据库中模式，该数据块一定不会超过一个数据分片大小；  
    其中每个分片的第1个字节代表该分片的序号；
17. 用户端利用哈希算法AHE计算加密后的数据分片的Hash值VHF，将加密后的分片连同签名后的VHF发送到超级节点指定的数据节点(对于小于PL2长度的直接存储的特小文件，直接发给超级节点)；发送时用对方数据加密公钥建立安全会话通道（谁存了什么Hash值的数据也属于用户隐私），并附上超级节点SND对数据节点ID和VNU的签名和SND的ID，表明本次存储操作是超级节点冻结了HDD的（以防止不付费的情况下进行DDOS攻击）；
18. 数据节点收到存储数据的请求后，先验证用户签名和超级节点签名是否正确，验证Hash值VHF是否正确，如果任何一项不正确拒绝存储并同时反馈给对方和超级节点SND。如果连续PNF次（例如3-5次）都出现同样情形，SND没收冻结的HDD（或其它惩罚措施）并终止本次流程；如果数据节点验证成功，将保存该数据分片；
19. 数据节点将用户端签名的数据和自己的签名一同反馈给超级节点SND，SND再向用户端发确认消息；同时SND向该数据节点对应的超级节点SNM发送消息（为提高效率，该消息可以定期批量同步，不用一次一发），SNM在数据节点索引表DNI中记录该数据节点已经存储了VHF数据分片，相应增加该数据节点的已使用空间计数，但无需增加该数据节点的每周期收益；
20. 如果某个数据分片保存不成功（数据节点溢出或多次上传都不成功），则用户端请求超级节点SND重新指定数据节点（可以先处理下一个分片，最后再统一请求超级节点SND补充所有缺失的数据节点）；此时如果有SND收到反馈但用户端因网络故障没收到SND转达消息的数据分片，SND将告知用户端该数据分片已成功存储无需补充，只补充真正没有存储的数据分片所需的数据节点，并惩罚未能存储数据的数据节点（溢出的数据节点以及反馈Hash值不匹配的数据节点除外）；
21. 重复以上步骤直到所有分片都加密并上传成功；
22. 用户端将KED上传到超级节点SND；
23. SND将所有数据分片的VHF、数据节点ID连同数据块序号VBI记录在数据分片表DFN中，将VBI、VHP、KED、VHB、VNF记录在数据块表DB中，标记引用次数为1，将系统的物理总存储容量VPS增加VNF\*VFL。转步骤bb;
24. 用户端用去重用密钥KD解密去重用加密存储密钥KED，得到存储密钥KS；
25. 用户端以存储密钥KS和AEY加密算法加密该数据块；
26. 用户端以步骤(p)相同方式将该数据块分为一个或多个数据分片；
27. 用户端对每一个数据分片，用AHE算法计算Hash值得到密文hash值VHF，比较所有VHF是否与超级节点SND返回的VHF列表一致，如果有一个不一致则检验下一组。如果从超级节点返回的所有组都不匹配，代表所有数据都是虚假的或者出现了罕见的Hash碰撞，则向超级节点SND请求分配新的数据节点，转步骤m；否则将匹配组的引用次数加1；
28. 用户端以用户存储公钥KUEp对存储密钥KS加密生成用户加密存储密钥KEU，将该块的编号（在本文件中第几块）、块序号VBI、长度、冗余算法编号（其中两个特殊编号代表副本和直接存储在元数据数据库中）、压缩算法编号、原文长度、KEU上传到该用户对应的超级节点SNU，SNU在用户文件去重表DUD中VNU记录的KEYS字段的相应位置记录KEU，将其余数据连同数据分片数VNF和VHF列表的Hash值VHB发给SNW，SNW记录在全局文件去重表DFD中VND记录的BLKS字段的相应位置；
29. 在所有块都上传完毕后，超级节点SNU在用户文件去重表DUD中的VNU记录的引用次数置为1，在新文件列表DNF中记录该文件最短存储时间的到期时间，在用户表中相应增加该用户的数据存储总容量，同时增加系统总存储容量；
30. 超级节点SNU根据用户ID和VHW查询所有数据块的序号VBI和该数据块对应的超级节点SND，根据VBI向SND查询引用次数，从而计算该文件的每周期费用（在一个记账周期PPC消耗的HDD总数）；释放该文件冻结的HDD，扣减初次存储费用（在最短存储时间PMS所耗费的HDD数量，为每周期费用乘PMS）；
31. 该文件的最短存储时间到期时，超级节点SNU在更新用户每周期总费用时，从新文件列表DNF中删除该文件项（ID为VHW），将该文件的每周期费用增加到该用户每周期总费用中。

(2) 用户端下载流程如下：

1. 用户端将文件的Hash值VHW或序号VNU发给该用户对应的超级节点SNU；
2. SNU从用户文件去重表DFD中根据VHW或VNU查询，如果没有相应记录则报错返回；如果有多于一条记录则表明出现了数据一致性错误，取第一条记录同时在系统日志中记录错误信息；读记录中的VND、数据块的数量和所有数据块的加密存储密钥KEU，将VND发给相应超级节点SNW；
3. SNW从全局文件去重表DFD中查询VND记录，如果没有相应记录则表明出现了数据一致性错误，报错返回；读取每个数据块的序号VBI/长度/存储模式/原文长度，并将原文长度累加计算出文件长度，将以上信息都返回给超级节点SNU，SNU连同数据块的数量和每个数据块的加密存储密钥KEU一起反馈给用户端；
4. 如果一个数据块的VBI的最高位为1，表明该文件被某个监管机构屏蔽，VBI只是一个索引，如果不在该监管机构所在的监管区的话，可以查询获取真正的VBI，具体方法是从VBI中找出相关的监管区编号以及流水号，用该流水号向该监管区的索引服务请求原始VBI；
5. 如果无法访问该监管区的索引服务，则意味着用户端所在区域属于该监管机构管辖范围，该文件被本地区有权限的监管机构所屏蔽，该文件无法读取， 报错返回；
6. 如果取回的VBI仍然是被屏蔽的（即最高位为1），则重复步骤d和步骤e，直到找到真正的原始VBI；
7. 用户端将每个数据块的序号VBI发给该数据块对应的超级节点SND（从VBI中获取超级节点编号），SND返回该数据块的明文Hash值VHP、每个数据分片的Hash值VHF及其对应数据节点的ID；
8. 用户端向各数据分片的数据节点请求数据，验证各数据分片的Hash值是否正确；
9. 用户端根据读取的数据分片生成数据块密文:采用副本方案则每个不同的数据分片去掉首字节后按顺序拼接即可；采用冗余编码方案则需要获取到足够数量的分片，每个分片去掉首字节，根据首字节标注的序号排列，调用冗余编码的解码算法计算出密文数据；
10. 可选步骤：对冗余编码的数据块，搜集尽可能多的数据分片，核查冗余编码解码时是否有错误位； 如果有，证明出现了非常严重的数据差错，汇报给超级节点SND，同时也继续本流程；
11. 根据数据块长度计算密文长度，计算方法是按照密钥长度对齐。
12. 使用存储密钥KS和算法AEY对数据解密（CBC模式，初始IV按照上传步骤(o)的相同方法设置），得到该数据块的明文；
13. 用Hash算法AP计算该数据块的明文Hash值，对比与VHP是否相同；如果不同则表明发生了致命的数据错误，汇报给超级节点SND；是否继续下载流程取决于用户本次下载操作的选项（是遇错误就终止还是能读多少算多少）；
14. 读该块数据的前2个字节，得到该块数据的压缩算法和末尾的未压缩字节数，去掉头2个字节和末尾的未压缩的字节，用用相应解压算法解压该数据块，再拼接未压缩的字节；对比生成的原文的长度与超级节点返回的该数据块的原文长度是否相同，如果不同则证明发生了严重的数据错误，是否继续下载流程取决于用户本次下载操作的选项（是遇错误就终止还是能读多少算多少）；
15. 将所有数据块拼接在一起，截掉超出文件长度的部分，得到原文件。

附注：如果只读文件的一部分，则根据每个数据块的原文长度计算来计算该读哪一数据块或哪几个数据块，然后按照相同的方法下载、冗余编码还原、解密、解压缩、拼接，即可得到所需数据。

（3）用户端删除流程如下（MVP不实现）

* 1. 用户端向用户对应的超级节点SNU发出Hash值为VHW的文件的删除指令，SNU检查该文件是否在新文件列表中，如是代表该文件尚未存满最短存储时间，报错返回；
  2. 超级节点SNU在用户文件去重表DUD中查询该用户的VHW记录，将该记录的引用次数减1，如果引用次数减完后仍然大于零则流程结束；
  3. 超级节点SNU从该记录中读取VND，删除该记录，向VHW对应的超级节点SNW发出删除VND指令；
  4. 超级节点SNW在全局文件去重表DFD中查下VND记录，将该记录的引用次数减1，如果引用次数减完后仍然大于零则流程结束；
  5. 超级节点SNW从该记录中读取每个数据块的VBI，删除该记录，从每个数据块VBI中解析其对应的超级节点SND的编号，向超级节点SND发出删除VBI指令；
  6. 超级节点SND将数据块表DB中VBI项的引用次数减1；
  7. 超级节点SNU将该用户的数据存储总容量和每周期总费用都相应减少，同时也减少系统总存储容量；
  8. 系统每次做GC时，对于引用次数为0的数据块需要遍历所有用户进行核实，核实后向该数据库各分片对应的数据节点发出删除指令，删除各数据分片，减少各数据节点的占用存储空间，然后数据块表DB中删除VBI数据块记录，相应减少系统的物理存储容量。

（4）数据重建

当一个数据节点出现故障后，就将其数据转存到其他节点，具体如下：

1. 故障节点所属超级节点SNM另外分配PNR（例如100）个数据节点，每个节点重建1/PNR的数据。
2. SNM调取故障节点的所有数据分片信息，包括该分片hash值、所属数据块的序号VBI，从VBI可以获知该数据块所对应的超级节点SND。
3. 从各超级节点批量读出故障节点的数据分片所位于的数据块的其它分片信息（序号、hash值、数据节点ID和访问地址）；将这些信息组成重建列表，平均分配给所有重建节点。
4. 重建节点收到重建任务后，读取其它分片信息，还原出丢失的数据分片，存入本地。将重建结果反馈给超级节点SNM，其中可能的错误包括溢出（Hash值位于已经满的分组中）或还原数据出错等。
5. 对完成数据重建的数据分片，超级节点SNM向该数据分片所对应的超级节点SND更新该数据分片的数据节点信息（异步批量更新方式），相应增加该数据节点的已使用空间计数，但无需增加该数据节点的每周期收益，除非新建这个数据分片后该数据节点所有预采购空间全部存满，需要新预采购一批空间；对于故障节点应该在调用本流程前就已经将其每周期收益清为零，并可能还会有进一步的处罚措施；对于因为溢出等可恢复错误而不能重建的数据分片，超级节点SNM将其重建任务分配给另外的数据节点，直到所有数据分片全部重建完毕；万一有实在无法重建的数据，向超级节点管理员报警。

（5）YTNS（MVP不实现）

有一个数据库DN记录文件ID到Hash值的映射，可以考虑ID用从1开始的顺序增长的整数，这样用一个文件形式存在的数组就实现了，连KV数据库都不需要，而且性能极好。

（6）动态写（MVP不实现）

写文件时覆盖原来的内容，通过替换分片的方式实现。数据节点需要增加替换分片功能，即将HASH值为h1的数据分片替换为HASH值为h2的分片，数据节点直接在数据区原分片的位置上写新数据，同时调整索引区，需要等数据节点实现溢出区功能后再实现，以保证替换分片功能总是可以成功。

当某个数据块的引用次数为1时也可以直接替换数据分片表DFN中相应的分片hash值的方式实现存储元数据的修改，否则（非常罕见地有重复的块）就将原数据块的引用次数减1，新数据块创建新的数据库记录。

也可以考虑动态文件单独的机制，不去重。

在执行动态写之前，需要先验证写权限，即用户端用写权限私玥对新数据签名。新写动态数据时提供写权限的公钥，替换分片时需要用对应的私玥签名才能执行。

（7）导出文件授权（MVP最好能实现）

* 1. 用户U1的用户端发送被授权文件的Hash值VHW给U1对应的超级节点SNU1，SNU1返回该文件的VND、所有数据块的用户加密存密钥KEU；
  2. 该用户端用自己的私玥解密所有数据块的KEU得到存储密钥KS，用另一用户U2的加密公钥KUEp加密存储密钥KS得到新的KEU；
  3. 该用户端生成随机密钥对称密钥KL，用U2的加密公钥KUEp加密KL得到KEL；
  4. 该用户端将VHW、VND、所有数据块的KEU一起用KL加密，连同KEL一起生成文件授权数据。
  5. 对于动态文件，分为读权限授权和写权限授权，一次可以授权其中的一项或两项。

（8）导入文件（MVP最好能实现）

* 1. 用户U2的用户端用自己的加密私玥解密授权数据中的KEL，得到KL；
  2. 该用户端用K解密授权数据其余部分，得到VHW、VND和所有数据块的KEU，发给自己对应的超级节点SNU2；
  3. SNU2检查用户文件去重表DUD中U2名下是否已经有VHW项，如果有的话直接返回，无需任何操作，也无任何花费；
  4. 采用与下载时相同的流程下载该文件，同时对每个数据块验证其明文Hash是否与超级节点记录的VHP相同，对压缩模式保存的数据块验证原文长度是否正确，最后验证全文的明文Hash值与VHW是否相同，其中任何一项不正确时报错返回；
  5. SNU2在用户文件去重表DUD中增加VHW项，并将文件VHW的各个数据块的引用次数都加1，今后U1和U2(可能还有其他用户)共同分摊该文件的存储费用。

（9）屏蔽文件

授权若干监管机构屏蔽任意文件的权限。该授权由YottaChain治理机构决定。每个监管机构可以屏蔽约10亿文件（230）,最多允许256个监管机构。屏蔽指定文件的原理是将被屏蔽的数据块的VBI移到特定区域，该区域通过特定域名来访问，每个监管机构可以采用屏蔽相关域名的方法从而阻止其管辖的监管区的用户访问该文件。相关流程如下：

1. 授权监管机构向VHW对应的超级节点SNW发出屏蔽请求，包含被屏蔽文件的Hash值VHW、操作员证书、操作员签名，同时支付一定的YTA给系统账号（屏蔽文件是需要付费的，这样可以约束监管机构不能无限制屏蔽文件）；
2. 超级节点SNW验证操作员的证书是否为授权监管机构签发，签名是否正确。如果其中一项不正确则报错返回；
3. SNW记录此次屏蔽操作日志，包括被屏蔽文件的Hash值VHW、监管机构ID、操作员ID、操作员签名等信息；
4. SNW在全局文件去重表DFD中查找所有包含VHW的记录；
5. 将该记录的每个数据块的VBI存入该监管机构所对应的被屏蔽索引数据库DMI，取DMI的流水号加上监管机构编号组成新的VBI（最高位置1），替换BLKS字段中相应的VBI；

（10）解除屏蔽

1. 授权监管机构向VHW对应的超级节点SNW发出解除屏蔽请求，包含被解除屏蔽的文件Hash值VHW、操作员证书、操作员签名（解除屏蔽无需付费）；
2. 超级节点SNW验证操作员的证书是否为授权监管机构签发，签名是否正确。如果其中一项不正确则报错返回；
3. 超级节点SNW记录此次解除屏蔽操作日志，包括被屏蔽文件Hash值、监管机构ID、操作员ID、操作员签名等信息；
4. 超级节点SNW在全局文件去重表DFD中查找所有VHW的记录；
5. 如果该记录的BLKS字段的VBI最高位不为1，代表该数据块没有被屏蔽，记录错误信息，跳过该数据块；
6. 如果VBI是被本监管机构屏蔽的（即VBI中的监管机构编号与本监管机构相同），从VBI中取出流水号，根据流水号从本监管机构对应的DMI数据库中取出原始VBI，更新记录中相应的数据；
7. 如果VBI不是被本监管机构屏蔽的（即VBI中的监管机构编号与本监管机构不同，也就是说在本监管机构之后还有其它监管机构也屏蔽了该文件），则上该VBI所对应的监管机构的DMI去查找被对应的原VBI；如果原VBI仍然不是被本监管机构屏蔽的，则继续递归查找，直到找到被本监管机构屏蔽的VBI。从本监管机构对应的DMI数据库中找到原始VBI，将上一个监管机构对应的DMI数据库中的相关VBI记录更新为原始VBI；
8. 如果BLKS字段中的密钥是不同的监管机构屏蔽的，则系统存在数据一致性错误，报错返回。

2. 存储网络：

存储网络是由海量的数据节点组成的超大规模文件存储集群，其中的每一个数据节点都负责接受超级节点的调配并存储来自用户端的文件分片。数据节点的架构图如下所示：



数据节点主要有以下几个功能：

·为用户端存储数据分片

·提供数据分片的下载

·向超级节点上报数据节点状态

·接受挑战，验证所存储的数据

·重建故障节点的数据

·具备中继能力的节点，承担中继功能

(1) 存储流程如下：

1. 数据节点监听来自用户端的I/O消息（参见用户端上传部份）
2. 收到消息后使用消息中的签名校验该消息的真实性，包括验证用户签名、验证超级节点对数据节点ID和Hash值VHP的签名，验证失败则报错返回；
3. 如果是存储新数据的消息，消息中包括用户ID、数据分片、数据分片的Hash值VHF、该数据分片所属的文件的流水号VNU、该数据分片所属的数据块对应的超级节点SND的ID，以及SND对数据节点ID和VNU的签名；验证该数据分片的Hash值是否与VHF相符以及超级节点签名，如果验证失败则返回错误信息给用户端和超级节点SND，终止本流程；
4. 将该数据分片写入数据区域的空白处，并记录该数据分片的Hash值VHF在索引区的对应位置；
5. 保存成功后将VNU、VHF、自己节点的ID和签名信息发给超级节点SND，确认存储完成；
6. 极端情况下SND会通知数据节点废弃该数据分片，则删除该数据分片；
7. 与超级节点SNM核对数据存储总量是否相应增加。此步骤为可选步骤，可以没有。

(2) 下载流程比较简单，就是由数据节点响应来自用户端的请求并发送VHF对应的数据分片至对端。被抽查也是用同一接口。

(3) 上报状态流程：

数据节点定时将当前状态上报给超级节点，状态信息如下所示

|  |  |  |
| --- | --- | --- |
| 字段 | 类型 | 说明 |
| CPU | Int | CPU占用率 |
| Memory | Int | 内存占用率 |
| Bandwidth | Int | 带宽使用 |
| MaxDataSpace | int | 最大存储空间（以数据分片为单位，包括与其它系统共享的空间） |
| AssignedSpace | Int | 已分配空间（以数据分片为单位，指分配给YTFS使用而且抵押了相应YTA的空间） |
| ProductiveSpace | int | 生产空间（以数据分片为单位，指计费空间，略大于已使用空间） |
| UsedSpace | int | 已使用空间（以数据分片为单位） |

超级节点会使用这些数据判断数据节点状态，以更好的协调分片存储。其中MaxDataSpace可能是动态变化的，当其它系统（用户自用的软件，或其它存储公链的挖矿程序）占用空间时，该空间会变小。ProductiveSpace和UsedSpace是用于跟超级节点对账的。对账如果不一致，MVP暂时不处理，只记录到报警日志中，仍然以超级节点的记录为准。

(4) 存储共识：

采用改良心跳法，将每个节点主动写心跳信号，变为每个节点被抽查方式形成心跳信号，从而解决数据节点不可信的问题。该方法简单易行，可靠性足够好（即使每个小时才抽查一次，如果一个数据节点保存了一半数据，那也只有不到400万分之一的概率能活过第一天而不被发现），而且工程实现简单，对存储设备资源损耗低，可以有效降低系统成本。具体方法为：

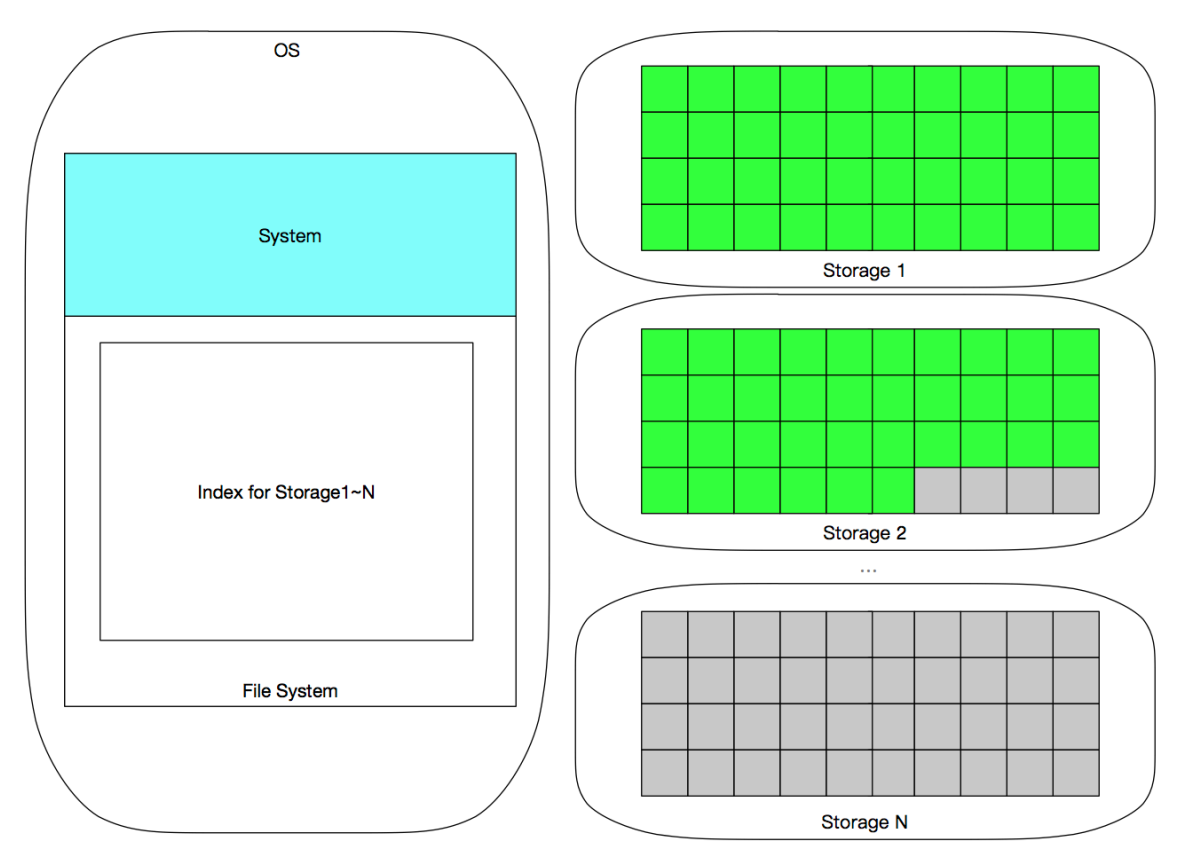
1. 超级节点定期向其管辖的具备抽查能力的节点（不需要中继节点即可直接访问所有数据节点的节点）发送被抽查名单（一次可以发送多个被抽查节点ID及其访问地址，以及被抽查的数据分片的hash值），被抽查名单是随机生成的。每个抽查周期PLL(例如1小时)每个数据节点都要被抽查一次（每次生成抽查名单都采用遍历所有数据节点，随机产生对应抽查节点的方式），然后再随机指定存储在该数据节点上的被抽查数据分片。
2. 抽查节点接收到超级节点下发的抽查指令后，将抽查名单加入到抽查队列中；
3. 抽查节点在每个抽查周期从抽查队列中取一项，向被抽查节点发送数据读取指令，读取指定的数据分片，然后验证该数据分片的hash值是否与指定的hash值相同。
4. 在下一次超级节点发送抽查名单时，抽查节点向超级节点反馈之前的抽查结果。对于未通过抽查的数据节点，超级节点将进行核实。如果该数据节点通过了超级节点的核实（例如连续抽查100次都正确），则不处罚，但是留下记录供今后类似判断时使用。如果该数据节点经核实属于恶意下线（例如48小时都不在线），则没收押金，踢出数据节点名单；如果经核实属于临时故障，但数据重建已经开始了，则仅扣除数据重建费用，该数据节点从零开始重新接单，无其它惩罚（这样允许不是特别稳定的数据节点加入进来，从而降低整个系统的平均存储成本）；如果在数据重建前恢复正常，则不做任何惩罚；对于矿池，只要同时故障率低于PFR(例如5%)，则不做任何惩罚。

（5）数据重建

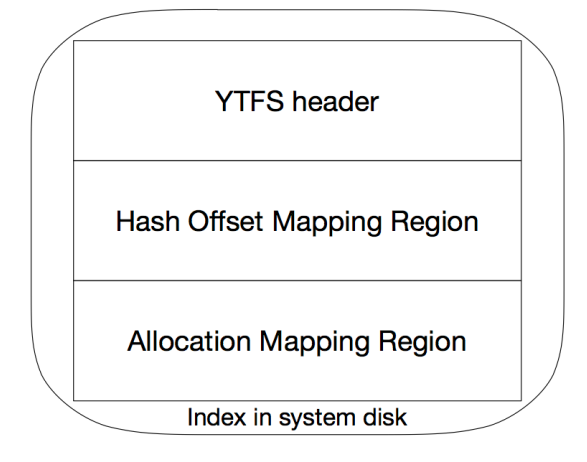
数据节点收到超级节点的数据重建任务后，对每个需要重建的数据分片，从其它数据节点读取同一数据块的其它分片信息，收集到足够多的数据分片后，还原出丢失的数据分片，存入本地。最后将重建结果反馈给超级节点SNM

(6) 数据节点的数据结构：

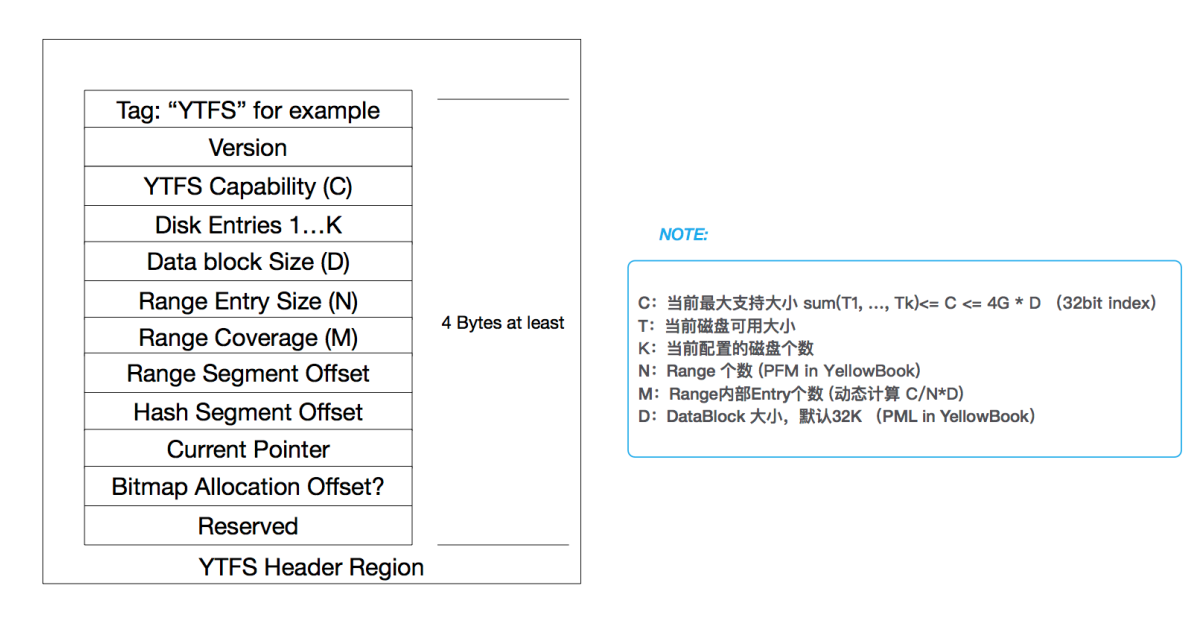
数据节点有至少两块盘，一块系统盘加载文件系统装系统文件(简化版Linux加数据节点程序)，以及索引文件，一块或多块数据盘是裸的块设备，没有格式化成任何文件系统。作数据盘的块设备可以是一块物理硬盘，可以是硬盘的一个分区，也可以是一个大文件映射得到的，有多种灵活方式处理，而且减少了一层文件系统，不仅提高了性能，也减少了文件系统的空间开销。



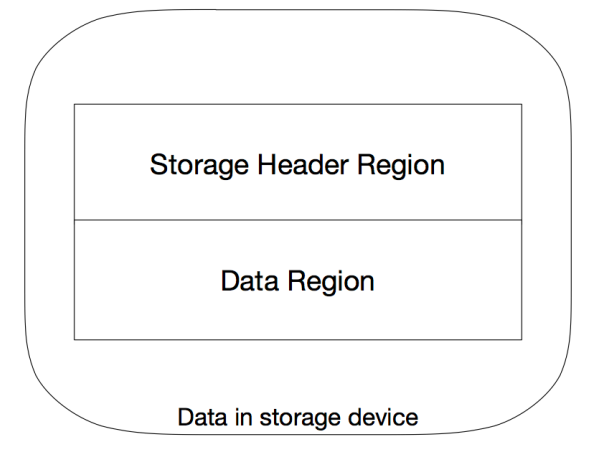
索引文件结构如下：



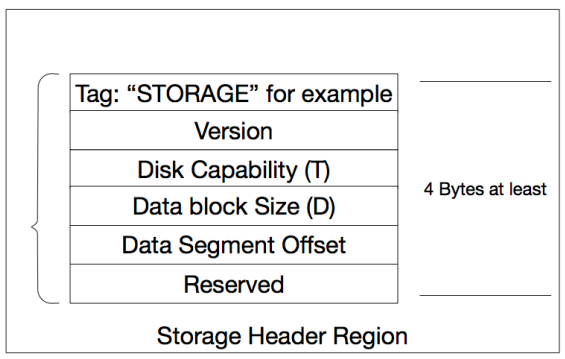
其中文件头的数据结构如下：



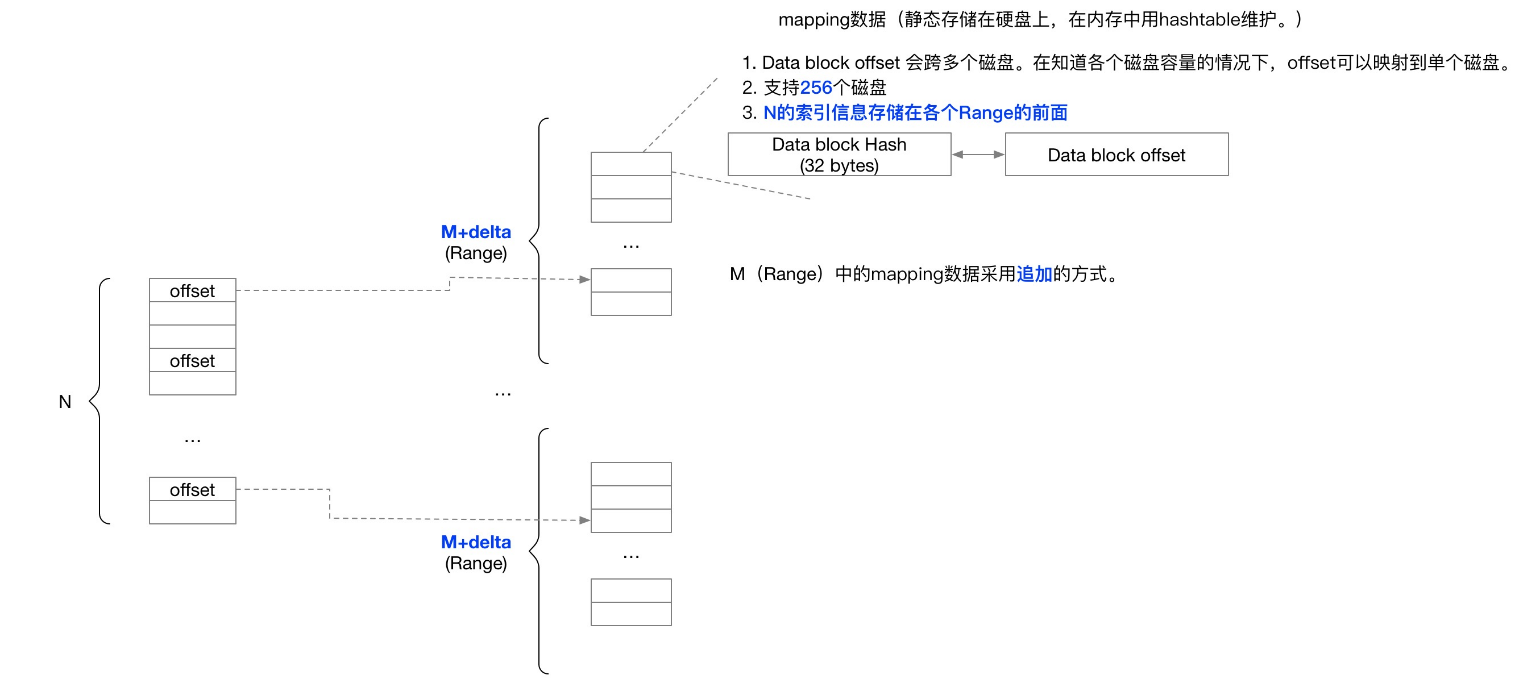
每个数据盘的结构如下：



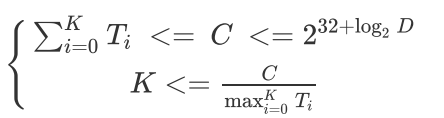
其中Storage Header区域的数据结构如下：



数据盘按照PFL（例如8K）长度分成很多分片，索引文件由很多索引项组成，每一个索引项包含一个数据分片的Hash值和4字节数据分片所在位置的索引值（从0开始自然增长）。可以根据每个数据盘的最大数据分片数确定该索引值位于哪个数据盘的哪个位置：



例如，如果PFL是8K，一个Hash是32字节（256位），每个数据分片的索引共占用32+4=36字节，则8TB的数据节点有1G个数据分片，对应大约40GB的索引区，有效存储容量大约99.5%。可以将8TB一次全部分配给YTFS，这时只有一个数据盘，也可以分成多个数据盘，每个数据盘的大小任意设置，各数据盘总容量的公式是：



当第一个数据盘存满数据后，就创建第二个数据盘加入到YTFS中。如果该矿机的存储空间超过上限，可以分成多个数据节点。

索引文件分组保存，按照hash值的某几位（前几位或后几位）来分组，每一组有PMF项，该数字略大于该数据节点最大容量/PFL/分组数，而且不同设备可以是不同值。每一组在索引文件固定占用36xPMF大小的空间，这一组超过PMF项时就溢出不存了，让超级节点另外分配其它数据节点保存。也可以在索引文件中增加一个溢出区，所有溢出的项都保存在此处，在查找Hash值时如果在该Hash值所属的分组没找到而且该组溢出了，则到溢出区继续查找。

在同一组中不排序，每次读写时都将整组的索引项读入内存，写入新数据后将其hash值在对应的组中末尾插入，读数据时在内存内逐项查找即可。

（7）矿工激励

每台矿机初始都要分配一定的空间给YTFS，该空间可以等于矿机的最大空间，也可以小于最大空间，留出一定空间给其它系统使用。注册后，系统都会预采购PNB个数据分片，此时该矿机的生产空间为PNB，已使用空间为0，最大空间则为所有可用的硬盘空间，每个记账周期都得到PNB为基数的HDD。当PNB个数据分片都被使用时，系统将继续预采购PNB个数据分片，直到所有存储空间都存满数据。

1. P2P网络：

YottaChain的P2P通信采用类似Napster的集中式索引服务+节点间P2P数据交换的模式。

（1）超级节点：

每个超级节点负责维护一个数据节点表DM，包含InterfaceList、是否直连和ExpireTime等字段，类似以下形式：

|  |  |  |  |
| --- | --- | --- | --- |
| NodeId | InterfaceList | Direct | ExpireTime |
| 0xDf85E2d1dD9867b9E30480ca4B5E2c32Ac0Fb578 | /ip4/117.189.34.182/tcp/8080  /ip4/192.168.1.105/tcp/8080 | true | 1548661941025 |
| 0x3d3503F258E6bBe47ED612759B7FfEe0d55fF1fa | /ip4/80.62.117.78/tcp/9080  /ip4/172.17.1.13/tcp/9080 | true | 1548661942124 |
| 0x3D57Ce10877CB03D92cb152E590EB44A95b645cc | /ip4/172.16.45.2/tcp/8080 | false | 1548661941025 |
| ...... | ...... |  | ...... |

其中NodeId为数据节点的ID，InterfaceList为该数据节点的所有网络接口列表，direct表示该节点是否可以直连，ExpireTime为该项目的超时时间。

（2）数据节点：



每个数据节点启动后都会向其归属的超级节点（根据节点ID计算）注册其网络接口信息，如上图所示，不同的数据节点各自在所属超级节点的注册表上进行注册。数据节点需在注册信息超时前重新注册网络接口信息，当超级节点在某数据节点超时后未收到注册信息则该节点视为失效，和抽查失败一样进入故障节点处理流程（停止分配新的存储任务，验证是否真的掉线，每周期收益清零，启动数据重建，以及可能的进一步处罚等）。

超级节点在注册消息超时或发生变化时需要进行连接测试以确定节点是否可以直达，同时更新direct字段。

注册消息的数据结构如下表：

|  |  |  |
| --- | --- | --- |
| 字段 | 类型 | 说明 |
| Type | Enum | 消息类型，这里为Register |
| NodeID | String | 数据节点ID |
| InterfaceList | Array<String> | 数据节点的网络接口列表 |
| ExpireTime | Timestamp | 当前注册信息的超时时间 |
| Signature | String | 消息签名 |

当超级节点接收到该信息时会更新注册表信息并返回回应信息，数据结构如下：

|  |  |  |
| --- | --- | --- |
| 字段 | 类型 | 说明 |
| Ack | Bool | true表示注册成功，false为失败 |
| Renew | Bool | 是否为续约，false为首次注册，true为续约 |

（3）用户端：

用户端与数据节点建立连接分两种情况，需要中继节点和不需要中继节点：

a. 不需要中继节点时，用户端在存储或下载数据时需要完成以下几项工作，如下图所示：



（i）获取数据节点：

当用户端上传或下载文件时，超级节点需要向用户端返回各分片对应的数据节点列表，列表每一项都包括相应数据节点网络地址信息（图中箭头1）。

（ii）建立连接：

当用户端获取到数据节点的网络接口列表后，分别向各接口发送ping消息，并选择延迟较低的接口作为数据传输接口（图中箭头2、3）。

（iii）传输数据

如果上一步中已经得到至少一个ping响应，则确定了目标数据节点及其网络接口，可以与数据节点直接建立传输通道来传输数据，如图中的箭头4所示。

b. 需要中继节点时，用户端在存储或下载数据时需要完成以下几项工作，如下图所示：



（i）获取数据节点：

当用户端上传或下载文件时，超级节点需要向用户端返回各分片对应的数据节点列表，列表每一项都包括相应数据节点网络地址信息（图中箭头1）。如果分配的数据节点中有无法直连的节点，超级节点还需要为本次数据传输分配一个中继节点，同时为每个需要中继的数据节点分配一个唯一配对标识（例如VHF），另外超级节点还需要告知无法直连的数据节点需要连接的中继节点及相应的配对标识（图中箭头2）。

（ii）用户端和数据节点分别与中继节点建立连接并发送配对标识（图中箭头3），并等待成功信息。

（iii）当中继节点发现有两个连接配对成功后会发送配对成功消息给双方，告知配对成功（图中箭头4），同时开始为这两个连接中转流量。

（iv）用户端和数据节点收到配对成功消息后了解到中继连接建立成功，可通过此虚拟链路进行数据交互。

c. 可能出现的问题:

当上传过程中可能由于数据节点宕机或网络变动导致分片上传失败但用户端没有收到响应，此时用户端不马上进行重试，而是收集本次上传中所有上传失败或无响应的分片信息重新向超级节点请求分配数据节点，此时可能有三种情况：

（i）数据节点确实上传失败，此时超级节点将为该分片重新分配数据节点

（ii）数据节点上传成功且超级节点已记录，此时超级节点将再次告知用户端

（iii）数据节点上传成功但向超级节点发送上传记录失败，此时超级节点将视为上传不成功，将为该数据分片重新分配数据节点；原始数据节点因为一直没收到超级节点的确认消息，在监听到VNU相关的存储操作已经完成的区块信息后丢弃相关的数据分片

1. 超级节点：

超级节点作为整个YottaChain的关键组件，主要提供以下几个功能：

·负责管理存储数据库、用户数据库、矿工数据库。

·负责协调用户端和数据节点之间的文件副本/分片的存储行为

·负责区块链的生成和保存

·提供智能合约功能

·负责对激励数据进行计算和更新

负责提供被屏蔽密钥服务；

·超级节点具有图形化管理端，可对超级节点进行监控和配置

1. 存储数据库，包括全局文件去重表DFD、数据块表DB和数据分片表DFN，其中DFD是每个文件的Hash值VHW对应的超级节点SNF负责维护，DB和DFN由每个数据块的明文Hash值VHP对应的超级节点SND负责维护。

全局文件去重表DFD，记录所有的文件的数据块信息：

VHW：byte[32]，主键，文件全文Hash值

VND：byte[8]，在DFD中的流水号，全局唯一

NLINK：byte[2]，引用计数（即保存了该文件的用户数。如果引用次数达到0xffff次，则该文件将永久保留，不再删除，即被某用户删除时不再执行引用次数减1的操作）

BLKS：变长，记录该文件所有数据块列表信息，每一项格式如下：

[VBI，块唯一ID，8字节] [块长度（3字节）] 【存储模式（1字节，其中第0-2位为冗余编码算法编号，其中副本和直接存储在元数据数据库中为两个特殊的编号；第4-6位为压缩算法编号，第3位和第7位保留，填0）】【该块原文长度，3字节 】【数据分片数VNF，1字节】【各数据分片的Hash值VHF的列表的Hash值VHB】【去重密钥KED】

数据块表DB记录每个数据块的信息：

VBI：byte[8]，主键，给该数据块生成一个唯一编号

VHP：byte[32]，索引，该数据块的明文摘要

KED：byte[32]，去重密钥

VHB：byte[16]，该数据块所有数据分片的Hash值VHF连在一起的Hash值

VNF：byte[1]，冗余后数据分片数

NLINK：byte[3]，引用计数，如果引用次数达到0xffffff，则该文件将永远不再删除，即再删除时不再执行引用次数减1的操作。

数据分片表DFN，记录每个数据分片的存储位置:

VBI：byte[8]，主键

VHF：byte[16]，每个数据分片的Hash值

NODEID：byte[4]，该数据分片所在数据节点的ID

1. 用户数据库，包含用户表、用户目录表、用户文件表、用户文件去重表和新文件列表，由每个用户所对应的超级节点SNU负责维护。

用户表记录用户数据，该表是同步到所有超级节点的：

UID：byte[4]，主键，用户ID

S3SecretKey：S3 API访问鉴权密钥

EOSID：Int64，对应的EOS账户

KUEp：byte[32]，用户加密公钥

KUSp：byte[32]，用户签名公钥

UsedSpace: 数据总存储容量

CostPerCycle: 每周期总费用

FirstNode: 第一个数据节点ID，没有的话填0。用链表方式记录该用户名下所有数据节点

用户目录表，记录目录信息，YTFS扩展层管理命名空间使用:

DirId：ObjectID，主键，mongodb中的uuid，12字节长度

ParentId： ObjectId ，父目录ID，用户第一级目录的父目录id就是用户表的UID

dirName：变长，目录名

用户文件表，记录各目录下的文件信息，YTFS扩展层管理命名空间使用:

ParentId：与fileName组合唯一键

fileName：与parentId组合唯一键

VNU：对应用户文件去重表中VNU

attributes：用户文件属性及acl等

用户文件去重表DUD，记录用户所有的文件的数据信息：

VNU：byte[8]，主键，文件流水ID，唯一

User4索引

VHW：byte[32]

NLINK：byte[2]，0-14位为引用计数（如果引用次数达到0x7fff次，则该用户将永久保存该文件，不再删除，即被删除时不再执行引用次数减1的操作）

VND：byte[8]，全局文件去重表中的索引

KEYS：变长，记录该用户对该文件所有数据块的密钥信息，每一项为用户加密存储密钥KEU（32字节）

新文件列表DNF，记录刚上传的还没有达到最短存储时间的文件：

FILEID：byte[36]，用户ID（4字节）+全文Hash值VHW（32字节）

VNU：byte[8]，索引，文件流水ID，唯一。

ExpireTime:最短存储时间到期时间

1. 矿工数据库：

矿工数据库包含数据节点表、矿池表和存储分片索引表，其中数据节点表包含数据节点ID、数据节点地址（IP或其它能通讯的信息）、是否可直接访问、是否可作为中继节点、是否可作为抽查节点、所归属矿池、Owner（即矿工的用户ID，有权对该矿机进行设置管理）、NextNode(属于同一owner的下一个数据节点，没有填0，用链表方式记录一个用户名下的所有数据节点)、数据节点登录公钥、数据节点签名公钥、数据节点加密公钥、最大空间（该矿机可用于数据存储的最大空间，可能与其它系统共享该空间）、已分配空间（已经分配给YTFS的空间）、生产空间（系统已采购的空间，也即矿工产生效益的计费空间）、实际存储空间（实际存储数据的空间大小）、当前在线状态、验证状态、矿机厂商、矿机系列号等。该表是同步到所有超级节点的。

矿池表包含每个矿池的ID、名称、登录公钥、签名公钥，以及所属数据节点的ID列表。该表是同步到所有超级节点的。

数据节点索引表DNI记录每个数据节点存储的数据分片的Hash值（或者指向数据分片表的索引）。

1. 智能合约数据库

每个持币用户都有HDD账户，包含上次余额、上次余额时间、每周期费用、每周期收益四个参数，HDD余额由这四个参数计算得到，计算公式为：

当前余额=上次余额+(当前时间-上次余额时间)\*（每周期收益-每周期费用）

其中时间以记账周期为单位。

矿机有自己单独的账户体系，每个矿机账户有生产空间和钱包地址两个参数。生产空间以数据分片数为单位，4字节整型。

1. 监管数据库

被屏蔽密钥数据库DMI，包含被屏蔽的密钥信息。该数据库是每个监管机构都有一个。

bStillMasked: BOOL，TRUE代表该记录存储的密钥仍然是被屏蔽的密钥；

originKey: byte[32]，原始密钥，当bStillMasked为TRUE时代表在该监管机构之前已经有其它监管机构屏蔽了用该密钥加密的文件，该密钥指向的是在先的监管机构的被屏蔽密钥数据库的记录。

1. 协调用户端和数据节点之间的存储行为

超级节点在接收到来自用户端的存储消息时（参见用户端上传的h小节），会根据注册的数据节点的状态（参见存储网络的第3节）计算出选择合适的数据节点（该算法将是长期优化的重点，MVP可以先按随机算法实现）。（该部分可参见用户端上传部分）

1. 负责记账数据的生成和保存

超级节点在接收到数据节点发来的存储完成的消息后验证签名，增加该数据节点的已使用空间，如果生产空间都已使用则增加生产空间并通过智能合约增加矿工的每周期收益；等该文件的所有块的所有数据分片都确认后，增加用户的每周期费用。对于新文件预扣最短存储周期的存储费用。

1. 负责对激励数据进行计算和更新

超级节点会根据区块中记录的信息计算各节点的开销和奖励，并将激励数据保存至激励网络中（此部分参见下一节激励网络）。每次新数据节点注册时，系统预采购PNB个数据分片的空间，当预采购空间都存满数据时再次预采购PNB个数据分片，直到该数据节点的所有存储空间都存满了数据。

为了节约区块链占用的空间，只在每次预采购时在区块链中记录一次。一个矿工的HDD余额按如下公式计算：

HDDm=HDDprev+Npro\*t\*u

其中HDDm为该矿工的HDD余额，HDDprev为该矿工上一次生产空间变动时的HDD余额，Npro为生产空间大小，t为上一次生产空间变动到现在的记账周期数,u为每个数据分片每个记账周期的收益，u=1/(1GB/PFL)/(1年/PPC)。

超级节点会在每个更新周期都遍历所有用户，更新其每周期费用。更新方法可以是这样：遍历每个用户的所有文件，比对其是否涉及最近系统上传的数据块，涉及的话就重新计算该文件的存储费用，相应减少该用户的每周期费用。

超级节点每隔一段时间（例如一周或一个月）做一次一致性检查，包括各元数据的一致性，也包括全部重算引用次数和每周期费用。

1. 在网络连通性出问题的情况下负责协调中继节点保证对上传/下载操作的正常进行

此部分可参考P2P网络部份的内容。

1. 超级节点具有图形化管理端，可对超级节点进行监控和配置

每个超级节点都包含一个图形化控制界面，可以在这里对各节点状态，存储空间使用状况，CPU使用率，带宽占用，日志信息（尤其是报警信息）等信息进行监控，以确保整个YottaChain存储网络的正常运行。

1. 激励网络：

激励网络是由基于EOS代码分叉的，智能合约为超级节点暴露出一组标准API，以供其对激励数据进行修改与更新。

YottaChain采用双层通证的激励方案，每个YottaChain账号拥有流通币YTA和若干资源通证（MVP只实现HDD），其中YTA对应原始代码中的EOS， HDD是合约币，通过智能合约与YTA进行交易。

其中YTA可在外部交易所进行流通，其市值由外部因素确定，HDD为存储资源的计费单位，矿工每存储1GB数据每年可获得1HDD，按每个记账周期为单位支付；用户存储时则跟其它有重复数据的用户分摊费用，所以平均每HDD可以获得超过1GB/年的存储服务。当某用户需要存储文件时，需要在内部交易市场以账号内的YTA兑换HDD，系统将自动扣减HDD作为存储费用。

当一个数据节点注册时，会为其创建一个矿机账户，系统会预采购部分存储量，该数据节点注册时指定的钱包地址会每个记账周期都得到HDD收益（比如该节点有1T存储，预采购10%，一个记账周期是一天，则该数据节点指定的HDD账户每天都能产生102.4/365HDD的收益），当预采购空间存满时追加采购量。此时该账户所有者可以选择将获得的HDD兑换为YTA，而HDD会被交换到系统HDD池。

普通用户账号（存储使用者）在使用YottaChain存储资源前需要使用YTA兑换HDD。当存储一个文件时，账户内的HDD至少要够该文件存储最短存储时间（按不去重计算），存储完成后HDD将直接扣除该文件最短存储时间的存储费用（以当时去重后的成本计算）；除了处于最短存储时间的新文件外，其它文件在每个记账周期都消耗用户每周期费用的HDD。每周期费用是按去重成本计算的，即拥有相同文件的各用户共同分摊其存储成本，每个更新周期重新计算所有用户的每周期费用。当用户账号内的HDD被消耗完之后，该用户将无法访问其文件；如果该用户不能及时补充HDD，其存储的数据都将被删除。

EOS源码级开发包括如下内容：

1）通证名称修改：

将EOS原生通证名称由EOS改为YTA

2）通证总发行量修改：

将YTA通证初始发行量由EOS的10亿改为40亿

3）增发规则修改：

将EOS每年增发5%修改为YottaChain白皮书规定的数量，所有增发YTA全部分给主超级节点和超级节点，分配方法和EOS相同。

4）YTA锁仓与解锁：（MVP不需要实现）

需支持YTA通证的锁仓与解锁，规则可参考目前ERC-20代币形式的YTA锁仓解锁规则。锁仓是某个账户在某种场景下获得YTA时自动实施的，一个账号可以由多种不同锁仓期的YTA，锁仓到期后自动解锁。可以用EOS的抵押机制实现。

智能合约开发包含以下功能：

资源通证是YottaChain内部使用的记账单位，只能用于与YTA兑换和购买对应资源，不能相互转账。目前只有HDD通证，对应硬盘资源，将来还会增加新的资源通证类型。

HDD账户用一个稀疏数组来实现，下标是YTA用户ID（即钱包地址），也就是说HDD用户和YTA用户的ID是相同的。每个用户账户有上次HDD余额、上次HDD时间、HDD每周期费用、HDD每周期收益、占用存储空间等参数，初始值都为零。HDD单位见HDD资源通证说明，占用存储空间以数据分片为单位，是最大248的无符号整型，即6字节。

为了提高效率，每个超级节点分管不同的YTA用户，根据YTA用户ID对21取模得到分管该用户的超级节点编号。

另有矿机账户，这是独立于YTA账户体系的另一套账户体系，用另一个稀疏数组来实现，下标是矿机ID。每个矿机账户有生产空间和钱包地址两个参数。生产空间也是以数据分片数为单位，4字节整型。

同样，每个超级节点分管不同的矿机，根据矿机ID对21取模得到分管该矿机的超级节点编号。

每个数据分片大小是上线前确定的一个系统参数，介于4KB和32KB之间。

具体说明如下：

1）. HDD资源通证合约：

a. HDD的最小值跟一些系统参数相关，这些系统参数在正式上线前才决定。大约在亿分之一左右，大致范围是百亿分之4.56至亿分之8.77之间。最大值1T应该够用了，1PB更好。

b. HDD余额计算：

每个YTA用户的HDD余额由HDD账户中的上次余额、上次余额时间、每周期费用、每周期收益四个参数计算得到，计算公式为：

当前余额=上次余额+(当前时间-上次余额时间)\*（每周期收益-每周期费用）

其中时间以记账周期为单位。

HDD账户的每周期费用、每周期收益和占用存储空间由超级节点所在服务器上运行的YTFS超级节点端程序调用合约外部调用接口设置，上次余额、上次余额时间由智能合约维护。

c. HDD/YTA兑换：

HDD与YTA之间的兑换都是以系统通证池为交易对手，HDD 兑换YTA是以用户账户中的HDD与系统交易通证池中的YTA进行兑换，YTA购买HDD是以用户账户中的YTA余额兑换系统交易通证池中的HDD。兑换后将上次余额记录为当前最新余额，将上次余额时间记录为当前时间。

其中系统通证池是一个特殊的系统账号，兑换过程中发生的HDD在用户账户和系统通证池之间转移是系统唯一允许HDD转账的例外情况，该转账功能只能嵌入在兑换程序中，不允许被其它外部程序调用。

HDD与YTA的兑换公式待定，暂定按照Bancor公式计算。

d. HDD的冻结和解冻：（MVP不需要实现）

能够以账户为单位对HDD余额进行部分冻结和解冻，用于文件上传过程中的HDD锁定与扣除操作。

2）. 合约外部调用接口（这些接口只有超级节点端部署的存储系统程序才会调用）：

a. 获取指定账户的HDD余额，传入参数为YTA用户ID。

b. 获取系统所有的HDD余额总和。可以考虑在HDD账户数组中用一个特殊的数组元素（其下标是一个特殊的不会与YTA用户ID冲突的值）记录总和，每次设置某个HDD账户的值的时候同步修改总和的值，

c. 设置指定账户的HDD每周期费用

输入参数包括新的每周期费用和调用者签名（即调用者对其它输入参数的签名，下同）。需要验证签名是否为该账户对应的超级节点签名（即只有分管该账户的超级节点才能调用该接口），验证通过后判断每周期费用是否不超过占用存储空间在不去重情况下的费用（这是为了防范超级节点作恶的一个保险措施），验证公式如下：

每周期费用 <= （占用存储空间\*数据分片大小/1GB）\*（记账周期/ 1年）

验证通过后再判断输入的每周期费用与当前值是否相同，如果不同的话需要计算当前余额保存为上次余额，记录当前时间为上次余额时间，同时写入新的每周期费用。

d. 创建矿机账户输入矿机ID和钱包地址，在该矿机名下记录钱包地址，并将该账户的生产空间初始化为零。

e. 增加矿机收益

输入参数是矿机ID、生产空间值和调用者签名。先验证签名是否为该矿机对应的超级节点签名，验证通过后再核查生产空间值是否正确，即当前生产空间加上预采购空间（这是一个事先设定的系统参数）是否等于传入的生产空间值（此步验证是为了防止误操作），核查通过后将生产空间增加预采购空间大小，并增加该矿机对应的钱包的每周期收益，计算公式为：

每周期收益 += (预采购空间\*数据分片大小/1GB）\*（记账周期/ 1年）

f. 扣除HDD余额

输入参数为扣除的值以及签名。先验证签名是否为该账户的签名（即超级节点只有获得该账户的授权后才能扣该账户的HDD），验证通过后计算该账户的HDD余额，将该账户的上次余额记录为当前余额减去扣除的余额，将上次余额时间记录为当前时间。

g. 冻结和解冻HDD余额（测试网不需要实现，上线前实现）。

h. 增加和减少占用存储空间

输入参数有用户ID、增加/减少的空间（数据分片为单位）、增加减少后的占用存储空间大小（数据分片为单位）、签名。先验证签名是否为该用户签名，验证通过后核查增加或减少后的占用存储空间大小是否正确（此步验证是为防止误操作），核查通过后相应增减占用存储空间

6.项目划分

1）、去中心化存储YTFS

1. YTSDK ：YTFS用户端核心模块，包括用户登录登出，上传主模块，下载主模块，授权管理，查询等功能，提供登录、登出、上传、下载、查询文件、导出文件授权（Later）、导入文件(Later)、查询状态（有多少文件，占多大空间，每周期总费用等）等核心接口。
2. YTFS-CoreService ：YTFS核心服务，运行在超级节点端，包括存储数据服务，数据节点服务，用户数据服务，数据重建调度，数据一致性检查（Later），GC（Later）等功能。
3. YTFS-DN ：YTFS数据节点端, 包括存储调度（从通讯层接受命令，解析后调用相应模块执行），写分片， 读分片， 重建分片， 删除分片（Later)， 数据自检（Later）等功能。
4. YTFS-ExtService：YTFS扩展服务(命名空间，ACL，S3用户key等)，运行在超级节点端。
5. YTObjectStorage ：YTFS S3 Server，提供标准S3的RestFul接口，后端调用YTFS核心接口实现。

2）区块链激励层YTChain

1. YTBP ：YottaChain主链，Fork EOS代码，将EOS改名为YTA，更改通证发行量和增发规则，以及相关的智能合约（主要提供HDD合约币的功能）。
2. YTSN：运行在超级节点上的系统管控服务，包括存储共识调度，失效节点惩罚，矿机矿池管理，智能合约接口等功能，同时引用YTFS-CoreService、YTFS-ExtService、P2PHost、P2PJavaBinding、MsgDef。
3. YTChain-DN：运行在数据节点上的管控服务，包括矿机注册注销，执行抽查任务，状态上报等。

3) P2P通讯

a. P2PHost: P2P通讯主模块

b. P2PJavaBinding: P2P通讯Java接口

c. MsgDef: 消息格式定义

(4) Portal

UserPortal：用户管理Portal, 包括用户存储文件信息（文件总数、本用户去重后总数、存储总容量、S3文件列表（Later）等），每周期总费用，上传下载（Later），系统状态（总存储容量，已使用空间总量，去重前存储空间总量，平均去重系数，数据节点总数，数据节点分布图（Later))，HDD管理（HDD余额，YTA/HDD双向兑换）等功能。

5）公共库

RS-Java: RS冗余编码算法Java实现

RS-Go: RS冗余编码算法Go实现

6）其它

YTDataNode: 引用YTFS-DN、YTChain-DN、P2PHost、RS-Go等所有数据节点端的库