

硕 士 研 究 生 读 书 报 告



题目IPFS -内容寻址，版本化，P2P文件系统

作者姓名 郭宁翔

作者学号 22051410

学科专业 软件工程

所在学院 软件学院

IPFS - Content Addressed, Versioned, P2P File System

A Dissertation Submitted to

Zhejiang University

in partial fulfillment of the requirements for

the degree of

Master of Engineering

Major Subject: Software Engineering

By

Guo Ningxiang

Zhejiang University, P.R. China

2020

摘要

星际间文件系统(IPFS)是一个点对点分布式文件系统， 旨在连接所有计算设备于同一文件系统。在某些方面，IPFS类似于Web，但IPFS可以看作是一个单一的BT种子群，在一个Git存储库中交换对象。换句话说，IPFS提供了具有内容寻址超链接的高吞吐量内容寻址块存储模型。这形成了一个通用的Merkle DAG，一种可以在此基础上构建版本化的文件系统、区块链，甚至是永久Web的数据结构。IPFS组合了分布式哈希表、激励块交换和自认证命名空间。IPFS不存在单点故障，节点之间不需要相互信任。

**关键词**：分布式文件系统，区块链，分布式哈希表，自认证命名空间, Merkel有向无环图

Abstract

The Inter-Planetary File System (IPFS) is a peer-to-peer distributed file system that seeks to connect all computing devices with the same system of files. In some ways, IPFS is similar to the Web, but IPFS could be seen as a single BitTorrent swarm, exchanging objects within one Git repository. In other words, IPFS provides a high throughput content-addressed block storage model, with content-addressed hyper-links. This forms a generalized Merkle-DAG, a data structure upon which one can build versioned file systems, blockchains, and even a Permanent Web. IPFS combines a distributed Hashtable, an incentivized block exchange, and a self-certifying namespace. IPFS has no single point of failure, and nodes do not need to trust each other.

**Keywords：**distributed file system, Blockchain, self-certifying namespace, distributed Hashtable, Merkle-DAG

1引言

业界在构建一个全局分布式文件系统方面已经进行了许多尝试。一些系统取得了显著的成功，而另一些则完全失败了。在学术界的尝试中，AFS获得了广泛的成功，至今仍在使用。其他人没有取得同样的成功。在学术界之外，最成功的系统是主要面向大型媒体(音频和视频)的点对点文件共享应用程序。最值得注意的是，Napster、KaZaA和BitTorrent部署了支持超过1亿同时用户的大型分布式文件系统。即使在今天，BitTorrent仍然保持着大规模的部署，每天都有数千万的节点在这个系统中运作。与学术型文件系统相比，这些应用程序实现了更多的用户和文件的分发。但是，它们并不是作为基础设施来设计的。虽然已经成功地重新定义了用途，但还没有出现提供全局、低延迟和去中心化分发的通用文件系统。

也许这是因为对于大多数场景来说，已经存在一个“足够好的”系统:HTTP。到目前为止，HTTP是有史以来部署的最成功的“分布式文件系统”。加上浏览器，HTTP已经产生了巨大的技术和社会影响。它实际上已经成为在互联网上传输文件的一种方式。然而，它没有利用过去15年发明的几十种出色的文件分发技术。从一个角度来看，扩展Web基础设施几乎是不可能的，因为有大量的向后兼容性约束和在当前模式中的强大既得利益者。但从另一个角度来看，自从HTTP出现以来，新的协议出现了，并得到了广泛的使用。缺少的是升级设计:增强当前的HTTP WEB，在不降低用户体验的情况下引入新功能。

业界之所以能够使用HTTP这么长时间，是因为移动小文件的成本相对较低，即使对于拥有大量流量的小型组织也是如此。但我们正在进入一个充满数据分发新挑战的新时代:(a)托管和分发的千兆级的数据集,(b)跨组织的大数据计算(c)大容量高清点播或实时流媒体,(d)版本管理和连接的大规模数据集,(e)防止意外丢失的重要文件,等等。其中许多可以归结为”大量的数据，随处可得”。由于受到关键特性和带宽担忧的压力，所以放弃了HTTP，转而采用不同的数据分发协议。下一步是让它们成为web本身的一部分。与有效的数据分发正交，版本控制系统已经成功地发展出了重要的数据协作工作流。分布式源代码版本控制系统Git发展出了许多有用的方法来建模和实现分布式数据操作。Git工具链提供了大型文件分发系统严重缺乏的通用版本控制功能。受到Git启发的新解决方案正在出现，比如Camlistore(个人文件存储系统)和Dat(数据协作工具链和数据集包管理器)。Git已经影响了分布式文件系统的设计，它解决了Merkle DAG数据模型的一些问题，支持强大的文件分发策略。有待探索的是这种数据结构如何影响面向高吞吐量的文件系统的设计，以及它如何可能升级Web本身。

IPFS，一种新颖的点对点版本控制的文件系统，试图解决这些问题。IPFS借鉴了过去许多成功的系统的经验。以接口为重点的集成产生的系统大于其各部分的总和。IPFS的核心原则是将所有数据建模为同一个Merkle DAG的一部分。

**2背景介绍**

**2.1分布式哈希表**

分布式哈希表(DHTs)被广泛用于协调和维护点对点系统的元数据。例如，BitTorrent MainlineDHT跟踪一组对等体，这是torrent群的一部分。

Kademlia是一种流行的DHT，提供:

1. 通过大规模网络高效查找:查询平均遍历⌈log2 (n)⌉节点。(例如，对于一个有1000万个节点的网络，有20个跳点)。

2. 低协调开销:它优化了发送到其他节点的控制消息的数量。

3.通过选择寿命较长的节点来抵抗各种攻击。

4. 广泛应用于点对点应用，包括Gnutella和BitTorrent，形成了超过2000万个节点的网络。

虽然一些点对点文件系统直接在DHTs中存储数据块，但这“浪费了存储和带宽，因为数据必须存储在不需要数据的节点上”。Coral DSHT在三个特别重要的方面扩展了Kademlia:

1. Kademlia将值存储在id“最近”(使用XOR-distance)到键的节点中。这没有考虑应用程序数据局部性，忽略可能已经拥有数据的“远”节点，并强制“最近的”节点存储数据，不管它们是否需要。这浪费了大量的存储空间和带宽。相反，Coral将地址存储给能够提供数据块的对等体。

2. Coral将DHT API从get\_value(key)放宽到get\_any\_values(key) (DSHT中的“草率”)。这仍然有效，因为Coral用户只需要一个(正在工作的)对等体，而不需要完整的列表。作为回报，Coral只能将值的子集分发到“最近的”节点上，以避免热点(当一个键变得流行时，重载所有最近的节点)。

3.此外，Coral根据区域和大小组织了称为集群的独立DSHT的层次结构。这使得节点能够首先查询其所在区域的对等节点，“查找附近的数据而不查询遥远的节点”，大大减少了查找的延迟。

S/Kademlia扩展了Kademlia，以防止恶意攻击在两个特别重要的方式:

1. S/Kademlia提供了保证NodeId生成安全的方案，防止Sybil攻击。它要求节点创建一个PKI密钥对，从中获得它们的身份，并相互签署它们的消息。其中一个方案包括一个工作量证明的加密谜题，以使生成西比尔的成本很高。

2. S/Kademlia节点在不相交的路径上查找值，以确保诚实的节点可以在网络中大量对手存在的情况下相互连接。S/Kademlia获得了0.85的成功率，即使敌人分数占节点总数的一半。

**2.2区块交换- BitTorrent**

BitTorrent是一个非常成功的点对点文件共享系统，它成功地协调了互不信任的对等网络(群)来合作向彼此分发文件。BitTorrent及其生态系统在IPFS设计方面的主要特点包括:

1. BitTorrent的数据交换协议使用了一种准“以牙还牙”的策略，奖励那些为彼此做出贡献的节点，惩罚那些只索取他人资源的节点。

2. BitTorrent对等体跟踪文件碎片的可用性，优先发送最稀有的碎片。这就减轻了种子的负担，使得非种子节点能够相互交易。

3.BitTorrent标准的“以牙还牙”很容易受到一些利用带宽共享策略的攻击。PropShare是一种不同的对等端带宽分配策略，可以更好地抵抗攻击策略，提高集群的性能。

**2.3版本控制系统- Git**

版本控制系统提供了工具来建模随时间变化的文件，并有效地分发不同的版本。流行的版本控制系统Git提供了一个强大的Merkle DAG对象模型，它以分布式友好的方式捕获文件系统树的更改。

1. 不可变对象表示文件(blob)、目录(树)和更改(提交)。

2. 对象是通过其内容的加密散列进行内容寻址的。

3.嵌入到其他对象的链接，形成Merkle DAG。这提供了许多有用的完整性和工作流属性。

4. 大多数版本控制元数据(分支、标记等)只是简单的指针引用，因此创建和更新的成本很低。

5. 版本更改只更新引用或添加对象。

6. 向其他用户分发版本更改只是简单地传输对象和更新远程引用。

**2.4区块交换- BitTorrent**

SFS提出了(a)分布式信任链和(b)平等的共享全局命名空间的令人信服的实现。SFS引入了一种构建自认证文件系统的技术:使用以下方案解决远程文件系统

/sfs/<Location>:<HostID>

其中Location为服务器网络地址，并且:

HostID = hash(public\_key || Location)

因此，SFS文件系统的名称验证其服务器。用户可以验证服务器提供的公钥，协商共享秘密，保护所有通信流量。所有SFS实例共享一个全局命名空间，其中的名称分配是加密的，不受任何中央主体的限制。

**3 IPFS架构**

IPFS是一个分布式文件系统，它综合了以前的点对点系统的成功思想，包括DHTs、BitTorrent、Git和SFS。IPFS的贡献是简化、发展和将已证实的技术连接成一个单一的有凝聚力的系统，比其各部分的总和要大。IPFS为编写和部署应用程序提供了一个新的平台，为分发和版本化大型数据提供了一个新的系统。IPFS甚至可以发展网络本身。

IPFS是对等的,没有节点享有特权。IPFS节点将IPFS对象存储在本地存储中。节点之间相互连接并传输对象。这些对象表示文件和其他数据结构。IPFS协议被分为一组负责不同功能的子协议。

**3.1身份**

节点由NodeId标识，这是公钥的加密散列，使用S/Kademlia的静态加密puzzle创建。节点存储它们的公钥和私钥(用密码短语加密)。用户可以自由地在每次启动时实例化一个“新的”节点标识，尽管这样会损失已累积的网络收益。节点被鼓励保持不变。

在第一次连接时，对等体交换公钥，并检查:

hash(other.PublicKey) equals other.NodeId

如果不等于，则终止连接。

IPFS倾向于自描述值，而不是将系统锁定在一组特定的函数选择上。哈希摘要值以多哈希格式存储，其中包括指定使用的哈希函数的短头和以字节为单位的摘要长度。例子:

<函数码><摘要长度><摘要字节>

这允许系统(a)为用例选择最好的功能(例如，更强的安全性vs更快的性能)，并且(b)随着功能选择的改变而升级。自描述值允许兼容地使用不同的参数选择。

**3.2网络**

IPFS节点定期与网络中的数百个其他节点通信，可能是通过广域网internet。IPFS网络堆栈特性:

•传输:IPFS可以使用任何传输协议，最适合WebRTC数据通道(用于浏览器连接)或uTP(LEDBAT)。

•可靠性:如果底层网络不提供可靠性，IPFS可以提供可靠性，使用uTP (LEDBAT)或SCTP。

•连通性:IPFS也使用ICE NAT穿越技术。

•Integrity:可选使用散列校验和检查消息的完整性。

•真实性:可选地通过使用发送者的私钥对消息进行数字签名来检查消息的真实性。

IPFS可以使用任何网络;它不依赖或假定访问IP。这允许IPFS在overlay网络中使用。IPFS将地址存储为multiaddr格式的字节字符串，供底层网络使用。multiaddr提供了一种表示地址及其协议的方法，包括对封装的支持。

**3.3路由**

IPFS节点需要一个路由系统，它可以找到(a)其他对等点的网络地址和(b)能够服务特定对象的对等点。IPFS使用基于S/Kademlia和Coral的DSHT实现这一点，并使用上面讨论的属性。IPFS的对象大小和使用模式与Coral和Mainline相似，因此IPFS DHT根据对象大小对存储的值进行区分。小于等于1KB的小值直接存储在DHT上。对于较大的值，DHT存储引用，这些引用是可以服务该块的对等体的节点id。

不同的用例将需要本质上不同的路由系统(例如广域网中的DHT，局域网中的静态HT)。因此IPFS路由系统可以被交换为一个符合用户需求的路由系统。只要满足这些接口，系统的其余部分就会继续工作。

**3.4区块交换- BitSwap协议**

在IPFS中，数据分布是通过使用一种受BitTorrent启发的协议与对等体交换块来实现的:BitSwap。像BitTorrent一样，BitSwap对等体希望获得一组块(want\_list)，并有另一组块提供交换(have\_list)。与BitTorrent不同，BitSwap并不局限于一个torrent中的区块。BitSwap作为一个持久的市场运行，节点可以获得他们需要的块，不管这些块是什么文件的一部分。这些块可能来自文件系统中完全不相关的文件。节点聚集在市场上以物易物。虽然物物交换系统的概念意味着可以创建虚拟货币，但这需要一个全球分类帐来跟踪货币的所有权和转移。这可以作为BitSwap策略来实现。

在基本情况下，位交换节点必须以块的形式相互提供直接的值。当块在节点之间的分布是互补的，即它们拥有对方想要的东西时，这种方法就很有效。通常情况下，情况并非如此。在某些情况下，节点必须为它们的块工作。在一个节点没有其他节点想要的东西(或者根本没有)的情况下，它寻找其他节点想要的东西，其优先级低于节点自己想要的东西。这鼓励节点缓存和传播稀有的片段，即使它们对这些片段并不直接感兴趣。

**3.5对象Merkle 有向无环图**

DHT和BitSwap允许IPFS形成一个大规模的点对点系统，用于快速和健壮地存储和分发块。在这些基础上，IPFS构建了一个Merkle DAG，这是一个有向无环图，其中对象之间的链接是源中嵌入的目标的加密散列。这是对Git数据结构的概括。Merkle DAG为IPFS提供了许多有用的特性，包括:

1. 内容寻址:所有内容通过其多哈希校验和唯一标识，包括链接。

2. 抗篡改:所有内容都用校验和验证。如果数据被篡改或损坏，IPFS会检测到它。

3.重复数据删除:所有包含完全相同内容的对象都是相同的，并且只存储一次。这对于索引对象特别有用，比如git树和提交，或者数据的公共部分。

IPFS Merkle DAG是一种非常灵活的数据存储方式。唯一的要求是对象引用必须按照上面的格式(a)处理内容，(b)编码。IPFS授予应用程序对数据字段的完全控制;应用程序可以使用它们选择的任何自定义数据格式，IPFS可能不理解这些格式。单独的对象链接表允许IPFS:

* 列出一个对象中所有的对象引用。
* 解析字符串路径查找，如foo/bar/baz。给定一个对象，IPFS将第一个路径组件解析为对象的链接表中的散列，获取第二个对象，并与下一个组件重复。因此，无论数据格式是什么，字符串路径都可以遍历Merkle DAG。
* 解析所有递归引用的对象。

原始数据字段和公共链接结构是在IPFS上构造任意数据结构的必要组件。虽然很容易看到Git对象模型如何适合这个DAG，但请考虑以下其他潜在的数据结构:(a)键值存储(b)传统关系数据库(c)链接数据三重存储(d)链接文档发布系统(e)链接通信平台(f)加密货币区块链。这些都可以在IPFS Merkle DAG之上建模，它允许这些系统中的任何一个使用IPFS作为更复杂的应用程序的传输协议。

**3.6文件**

IPFS还定义了一组对象，用于在Merkle DAG之上对版本化的文件系统进行建模。这个对象模型类似于Git的:

1. 块:一个可变大小的数据块。

2. 列表:块或其他列表的集合。

3.树:块、列表或其他树的集合。

4. commit:树的版本历史中的快照。

希望准确地使用Git对象格式，但不得不引入一些在分布式文件系统中有用的特性，即(a)快速大小查找(对象中添加了聚合字节大小)，(b)大文件重复数据删除(添加了一个列表对象)，以及(c)将提交嵌入到树中。然而，IPFS文件对象与Git足够接近，因此两者之间的转换是可能的。此外，还可以引入一组Git对象来进行转换，而不会丢失任何信息(unix文件权限等)。

**3.7 IPNS:命名和可变状态**

到目前为止，IPFS堆栈形成了一个点对点块交换，构造了一个内容寻址的对象DAG。它用于发布和检索不可变对象。它甚至可以跟踪这些对象的版本历史。但是，缺少一个关键组件:可变命名。如果没有它，所有新内容的通信都必须在带外进行，发送IPFS链接。所需要的是在同一路径上检索可变状态的某种方法。

值得说明的是——如果最终需要可变数据——努力构建一个不可变的Merkle DAG。考虑来自Merkle DAG的IPFS属性:对象可以(a)通过散列检索，(b)完整性检查，(c)链接到其他对象，(d)无限缓存。在某种意义上:对象是永久的。

这些都是高性能分布式系统的关键特性，在这种系统中，数据在网络链路之间移动是昂贵的。对象内容寻址构建了一个web，具有以下特点:(a)显著的带宽优化，(b)提供不可信内容，(c)永久链接，以及(d)对任何对象及其引用进行完全永久备份的能力。

Merkle DAG(不可变内容寻址对象)和命名(指向Merkle DAG的可变指针)实例化了许多成功的分布式系统中存在的二分法。这包括Git版本控制系统，其不可变对象和可变引用;以及UNIX的分布式继承者Plan9，它具有可变的固定文件系统和不可变的Venti文件系统。LBFS还使用可变索引和不可变块。

**4 展望未来**

IPFS背后的思想是学术界和开源数十年成功的分布式系统研究的产物。IPFS综合了迄今为止最成功的系统中的许多最好的想法。除了BitSwap这一新颖的协议之外，IPFS的主要贡献是系统的耦合和设计的综合。

IPFS是一个雄心勃勃的新去中心化互联网基础设施的愿景，许多不同类型的应用程序可以在此基础上构建。最不济，它可以用作全局的、挂载的、版本化的文件系统和命名空间，或者用作下一代文件共享系统。最理想情况下,它可以推动网络新视野, 在发布有价值的信息的同时对出版商保持不透明，但对利益相关者保持透明, 用户可以信任他们所接收到的内容，而不需要信任接收内容的对等体,也没有旧的重要文件会丢失。IPFS期待带我们走向永久的网络。

参考文献

[1] J. H. Howard, M. L. Kazar, S. G. Menees, D. A. Nichols, M. Satyanarayanan, R. N. Sidebotham, and M. J. West. Scale and performance in a distributed file system. ACM Transactions on Computer Systems (TOCS), 6(1):51–81, 1988.

[2] I. BitTorrent. Bittorrent and ̧ttorrent software surpass 150 million user milestone, Jan. 2012.

[3] P. Maymounkov and D. Mazieres. Kademlia: A peer-to-peer information system based on the xor metric. In Peer-to-Peer Systems, pages 53–65. Springer, 2002.

[4] L. Wang and J. Kangasharju. Measuring large-scale distributed systems: case of bittorrent mainline dht. In Peer-to-Peer Computing (P2P), 2013 IEEE Thirteenth International Conference on, pages 1–10. IEEE, 2013.

[5] M. J. Freedman, E. Freudenthal, and D. Mazieres. Democratizing content publication with coral. In NSDI, volume 4, pages 18–18, 2004.

[6] I. Baumgart and S. Mies. S/kademlia: A practicable approach towards secure key-based routing. In Parallel and Distributed Systems, 2007 International Conference on, volume 2, pages 1–8. IEEE, 2007.

[7] B. Cohen. Incentives build robustness in bittorrent. In Workshop on Economics of Peer-to-Peer systems, volume 6, pages 68–72, 2003.

[8] A. J. Mashtizadeh, A. Bittau, Y. F. Huang, and D. Mazieres. Replication, history, and grafting in the ori file system. In Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles, pages 151–166. ACM, 2013.

[9] D. Levin, K. LaCurts, N. Spring, and B. Bhattacharjee. Bittorrent is an auction: analyzing and improving bittorrent’s incentives. In ACM SIGCOMM Computer Communication Review, volume 38, pages 243–254. ACM, 2008.

[10] D. Mazieres and F. Kaashoek. Self-certifying file system. 2000.