

基于 HDFS 的安全云存储模型

林穗, 黄健, 姜文超, 覃国民
(广东工业大学 计算机学院, 广东 广州 510006)

摘要: 针对基于 HDFS 的云存储平台对元数据绝对全权管理控制问题, 提出元数据自主组织模型 ASOM, 通过分析 DataNode 与 NameNode 之间的交互过程, ASOM 设计并实现了元数据子服务, 通过提取 HDFS 中 DataNode 元数据信息直接跟元数据子服务器交互, 并且由元数据子服务器处理文件与块、块与 DataNode 之间的映射关系. 通过对 HDFS 源代码进行分析编译和模块替换, 搭建原型系统进行测试, 测试结果表明: 对 NameNode 中的元数据进行必要的安全隔离保证了用户对元数据的控制与管理, 可有效避免平台管理员对元数据可能的恶意操作和安全隐患.

关键词: Hadoop; 云存储; HDFS ASOM 模型

中图分类号: TP309

文献标志码: A

文章编号: 1007-7162(2014)03-0049-06

A Secure Cloud Storage Model Based on HDFS

Lin Sui, Huang Jian, Jiang Wen-chao, Qin Guo-min
(School of Computers, Guangdong University of Technology, Guangzhou 510006, China)

Abstract: It proposed a novel secure cloud storage model (ASOM), based on HDFS. Through isolating the meta-data from the physical storage and the communication between Data nodes and metadata sub-servers, the ASOM model can guarantee secure meta-data management. The main advantages of our approach include avoiding the superposition of complex security policies and the mistrust between the users and the platform. Furthermore, our security storage service can be easily integrated into the cloud computing environment.

Key words: Hadoop; cloud storage; HDFS; ASOM model

伴随信息技术快速发展, 信息量呈指数增长, 数据持久化逐步成为企业发展的关键问题之一. 考虑到云存储成本优势和管理便捷性优点, 很多企业也开始将数据转移到云平台完成数据持久化.

HDFS 是 GFS 的开源实现, 是 Hadoop 的一个组成部分, 由于具有高容错、高吞吐量、高可靠性等优点, 已成为云存储主流平台之一^[1-3]. Hadoop 最新的计算框架 Yarn 不仅解决了 JobTracker 单点故障问题, 而且也开始支持除了 MapReduce 编程模型外的其他计算模型. 另外, 最新版的 Hadoop2.3.0, 实现异构层次化存储和 DatanodeCache 两大特性. 未来 Hadoop 的发展方向将支持高吞吐量的离线计算和低延时的实时计算.

利用 HDFS 部署私有云存储平台^[4]与租用大企业共有云存储相比, 前者的开销较高. 但将企业内部

机密数据及其之间的映射关系完全存到云服务提供商的服务器上, 无法确保隐私和数据的安全性, 甚至会因数据安全问题给企业带来灾难.

文献 [5] 提出一种数据周期保护和自我销毁的框架和协议, 但不适合企业用户. 文献 [6] 针对安全方面提出了安全机制和安全策略. 目前对安全和可信方面的缺陷大部分是采用加密和访问控制方式来实现控制. 文献 [7] 基于 HDFS 输入输出方式在客户端采用经典的加密算法和属性方式对数据进行加密, 同时还对密钥进行了加密传输, 即采用双重加密, 以实现加密后数据的可操作性. 密钥保护管理方面除了加密, 文献 [8] 提出一种在云环境下的新的基于全局逻辑层次的密钥推导机制的密钥管理方法. 文献 [9] 中也提出对数据存储安全问题, 要注意对加密算法的更新换代. 然而, 对数据的加密方法在

收稿日期: 2014-04-02

基金项目: 广东省自然科学基金资助项目 (S2012040006729); 广州市科技计划项目 (2013Y2-00074)

作者简介: 林穗 (1972-), 女, 副教授, 主要研究方向为云计算、操作系统和 Web 服务等.

保证数据安全的同时都会和读写效率成反比. 文献[10]则是利用图论语义方式的访问控制策略实现了云计算安全. 文献[11-12]从云计算安全架构进行研究, 提出新的数据安全模型.

上述安全机制在一定程度上能保证数据的安全性, 但由于数据及其组织、映射关系全部存储于云平台端, 不能有效避免云平台管理人员的恶意操作, 无法从根本上确保数据安全可靠, 因此本文提出

ASOM(all self-organization model) 安全云存储模型, 通过将元数据及其物理映射与物理数据隔离的方法进行元数据自主管理, 将元数据的组织管理转移到客户端进行, 可有效避免因平台内部管理漏洞等因素导致的数据安全问题[13].

1 ASOM 模型分析

图1是ASOM模型图.

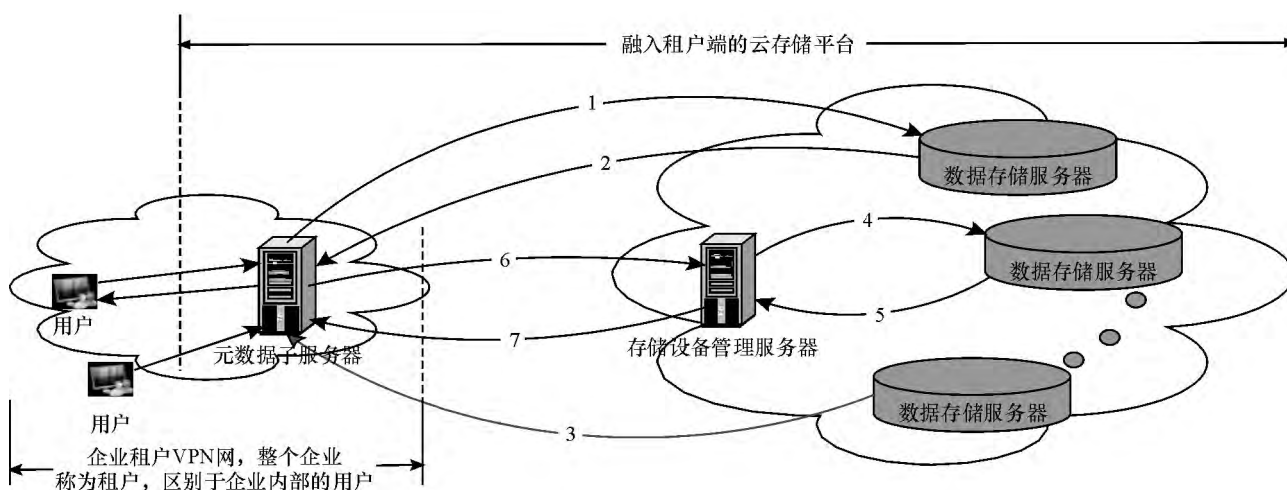


图1 ASOM模型图

Fig.1 ASOM model

从图1中可知, ASOM 主要分为元数据子服务器, 存储设备管理服务器, 数据存储服务器3个部分. 其中, 元数据子服务器管理云平台的元数据, 存储管理服务器完成数据的持久化, 元数据子服务器位于租户本地局域网或VPN内, 负责租户隐私信息和元数据的本地化管理, 实现了元数据和数据存储管理的隔离. 整个云存储平台进行数据通信的主要有存储设备管理服务器与元数据子服务器, 数据存储服务器与元数据子服务器, 数据存储服务器与存储设备管理服务器.

图1中, 数据流1、2是数据读写流. 数据流4是存储管理服务器向数据存储服务器发送的命令, 以此来管理数据存储服务器上的数据信息. 数据流5是数据存储服务器向存储设备管理服务器汇报其状态信息. 数据流6是存储设备管理服务器为管理数据存储服务器获取来自元数据子服务器必要的元数据信息. 数据流7是存储设备管理服务器向元数据子服务器反馈的信息. 数据流3及其交互实现是数据存储服务器直接向元数据子服务器传送的数据块元信息, 也将是本文详细讨论的安全方法.

2 数据节点与元数据服务器

元数据是指数据的数据[14], HDFS的元数据有两种形式: 一种是存储在内存的处于活动状态的元数据, 供服务器使用, 另外一种则是静态元数据文件. 同时动态元数据又分为临时元数据和可持久化的元数据, 后者主要保存了文件或目录等构建文件系统的信息[15]. 内存中元数据持久化后, 形成元数据文件. 持久化元数据是在HDFS启动和写数据时创建并将其持久化成元数据文件, 非持久化元数据是由DataNode节点和NameNode节点进行交互, 并由NameNode节点组织.

如图2所示, 将DataNode节点上报给NameNode节点信息分离成两类, 分别为Datanode节点信息的元数据和数据块对应Datanode节点映射关系的元数据信息. 前者信息不改变其交互机制, 继续向NameNode节点传递, 后者信息向元数据子服务器传递. 因此, HDFSClient相对DataNode节点要实现RPC服务器, DataNode节点相对HDFSClient是客户端. DataNode节点周期性调用HDFSClient远程方法向元数据子服务器上信息.

DataNode节点是通过RPC机制, 利用blockRe-

port() , sendHeartbeat() 和 blockReceived() 这 3 个远程方法传递元数据信息^[4]. 在 DataNode 类的线程执行方法 run() 中调用 offerService() , offerService() 调用以上 3 个远程方法. 部分类图如图 3 所示.

如图 4 所示, 根据原 HDFS 元数据传递思想和 RPC 机制, 实现元数据信息的分离需如下操作:

(1) 创建接口. 在 org. apache. hadoop. hdfs. protocol 包中创建 DatanodeClientProtocol 接口. 在接口中声明两个方法 blockReport() 和 blockReceived().

(2) 实现接口. DatanodeClientProtocol 接口是由 org. apache. hadoop. hdfs. DataClient 类实现.

(3) 替换原有的远程过程调用. 在调用 DatanodeProtocol. blockReport() 方法中替换成调用 Datanode-

Protocol. blockReport() 方法, blockReceived() 方法做同样的处理.

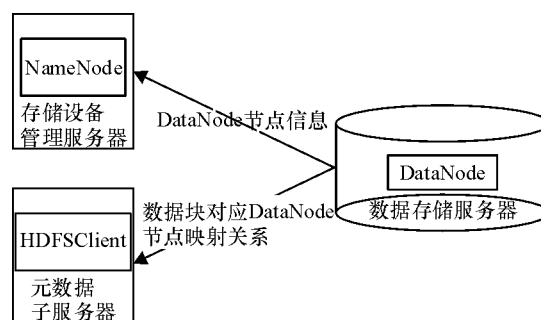


图2 ASOM 模型两类元数据传递示意图

Fig. 2 Two types of metadata transmission on ASOM model

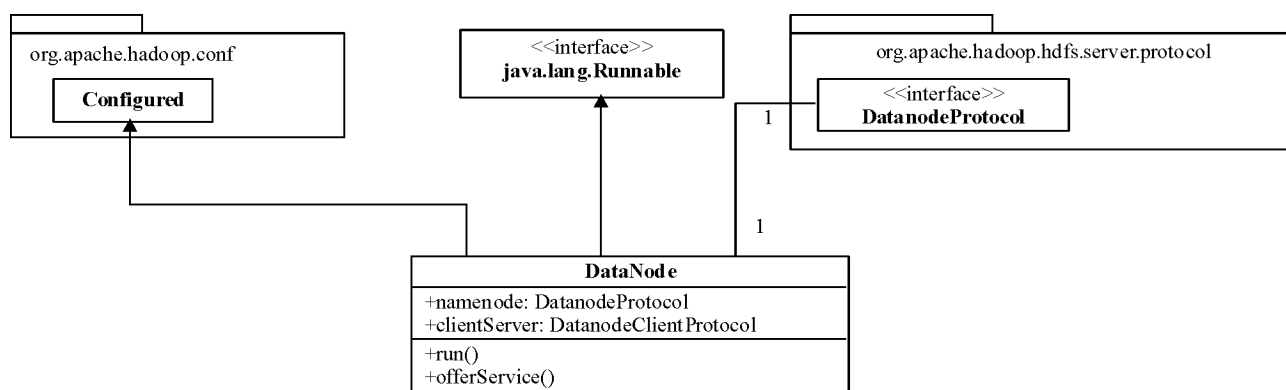


图3 HDFS 元数据传递类图

Fig. 3 HDFS metadata transmission class diagram

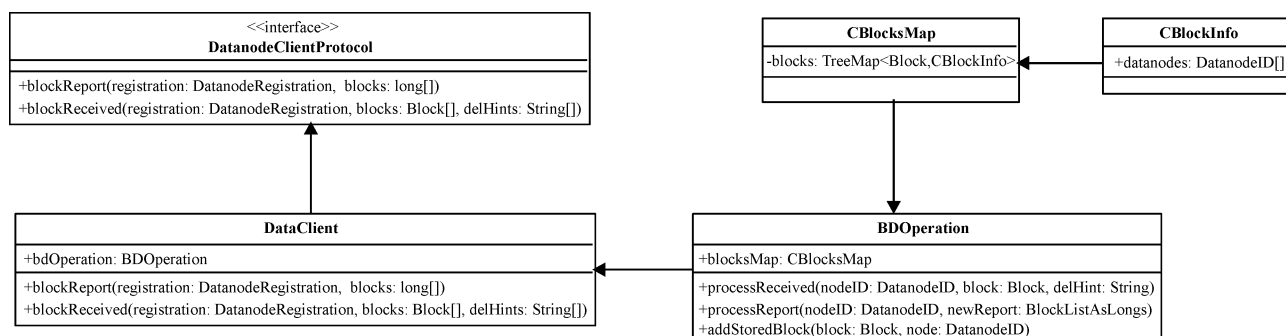


图4 ASOM 模型处理元数据映射关系类图

Fig. 4 Metadata mapping class diagram on ASOM model

完成数据的提取则需要重新实现 blockReport() 和 blockReceived(). DatanodeClientProtocol 接口中 blockReport() 方法实现过程: (1) 前期一些必要检查; (2) 调用 BDOperation. processReport() 方法; (3) 后期处理. 实现流程如图 5 所示.

BDOperation. processReport() 则是处理数据块信息关键方法, 主要步骤包括: (1) 开始前检查操作; (2) 对数据块进行分类; (3) 处理各类数据块; (4) 结束处理. 实现流程如图 6 所示.

HDFSClient 对 DataNode 节点上报的元数据信息进行处理主要是为在 HDFSClient 节点内存中保存数据块和对应数据节点映射关系. 首先在 org. apache. hadoop. hdfs 包中创建 CBlockMap 类, 然后在 BDOperation 类中定义 CBlockMap 类型的静态变量用来保存数据块和对应 Datanode 节点映射关系. 在图 4 的类图中, CBlockMap 是通过 TreeMap 来保存映射关系. CBlockInfo 类是 CBlockMap 内部类, 该类 DatanodeID 数组类型的变量是保存相应数据块的数据

节点集. 这就形成数据块和数据节点映射关系的数据结构. `blockReceived()` 方法会将数据节点最近的接收到数据块信息传递给 `HDFSClient` 端, 实现流程图类似 `blockReport()`.

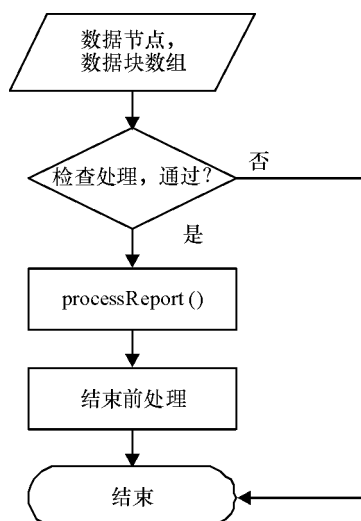


图5 `blockReport()` 实现流程图

Fig. 5 Flow chart of `blockReport()` implementation

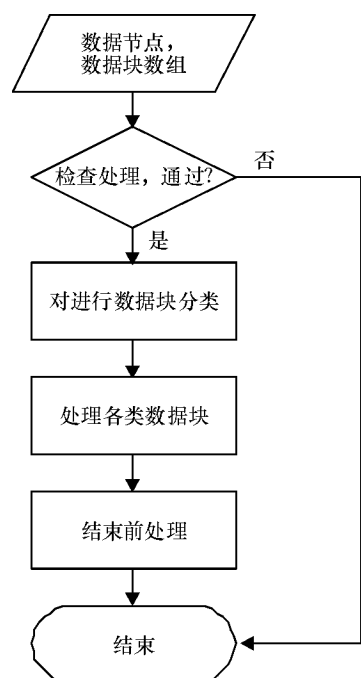


图6 `processReport()` 实现流程图

Fig. 6 Flow chart of `processReport()` implementation

元数据子服务器实现了数据块与数据节点映射关系处理后. 元数据的持久化和加载元数据文件功能需要在 `HDFSClient` 端利用重构的类 `FSImage` 和类 `FSEditLog` 实现, 通过调用这两个类可以在 `HDFSClient` 端的内存中完成文件和数据块的映射关系.

通过将构建文件系统层次结构和映射关系的重要元数据移植到 `HDFSClient` 端, 用户对文件的常用

管理操作都会访问元数据, 这些对元数据操作也被转移到 `HDFSClient` 上处理. 若涉及数据管理的元数据操作, 则需要把相关的信息向 `NameNode` 节点上报, 以便 `NameNode` 节点可以正常管理数据存储服务上的数据. 在读写过程的实现中, 需要在 `org. apache. hadoop. hdfs` 包中创建一个类, 该类主要处理读写过程所涉及的数据. 读的过程需要得到指定文件对应数据块位置, 需要向 `NameNode` 节点报告发现的坏块. 写过程则需要创建文件, 实现数据块分配.

`NameNode` 节点在管理数据存储服务用到元数据子服务器上的元数据时, 利用 `RPC`, 获得部分元数据信息. `RPC` 可以避免将元数据信息完整地保存在 `NameNode` 节点上, 进而避免平台管理员利用技术手段获得 `NameNode` 节点内存中的完整元数据信息. 其实现步骤如下: (1) 创建接口. 接口中声明获得 `NameNode` 节点所需元数据方法 `method`. (2) 实现接口. 在自定义类中实现接口. (3) 在 `FSNamesystem` 类中调用 `method`, 以得到所需的结果.

通过以上分析可知, `HDFSClient` 端除了扮演传统云存储平台的客户端角色外, 相对 `NameNode` 节点和 `DataNode` 节点, 同时具有服务器部分管理功能.

对于 `NameNode` 节点 `HDFSClient` 作为服务端的实现步骤如下:

- (1) 在初始化 `HDFSClient` 时建立 `RPC` 服务器;
- (2) `NameNode` 节点由原先的 `C/S` 架构获得 `HDFSClient` 的 `IP` 地址;
- (3) `NameNode` 节点再根据 `IP` 地址建立连接.

对于 `DataNode` 节点 `HDFSClient` 作为服务端的实现步骤如下:

- (1) `NameNode` 节点获知 `HDFSClient` 端 `IP` 后通过心跳机制将地址发送给所有的 `Datanode` 节点;
- (2) 在心跳机制中我们需要再添加一个变量 `linkState` 来标识 `Datanode` 节点是否成功连接到 `HDFS` 客户端机器上;
- (3) 在 `Datanode` 中定义一个方法, 用来检查 `Datanode` 节点是否连接到 `HDFSClient`, 若没连接成功就从获取 `HDFSClient` 的 `IP` 地址处理连接; 若连接成功后改变 `linkState` 的值来告知 `Namenode` 节点.

3 实验测试

测试实验在一个 `Hadoop` 集群上完成, 集群由一台存储设备管理服务器、一台元数据子服务器和两台数据存储服务器组成. 各服务器操作系统是 `ubuntu13.04`. 实验部署如图7所示.

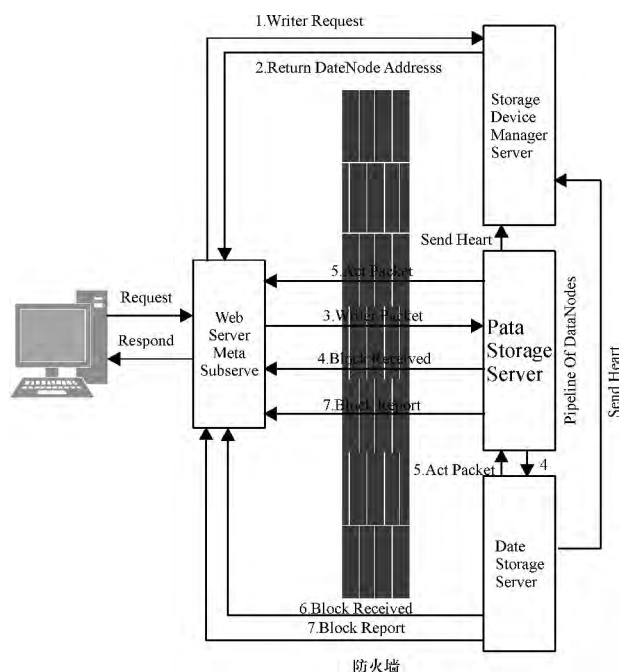


图7 实验部署图

Fig. 7 The deployment process

3.1 安全性分析

首次启动集群,通过向平台写数据开始创建元数据.其过程转变成在元数据子服务器内存中生成对应的目录树和文件所对应的数据块,并定时将需要持久化的元数据持久化到元数据子服务器上.数据在数据节点完成存储后会向元数据子服务器发送 Block Recived 信息,完成数据块和 DataNode 节点的映射关系.此后通过 Block Report 实现元数据子服务器上元数据映射关系和 Datanode 节点存储的数据块的同步.重启集群时,持久化元数据的加载也是在元数据子服务器上完成.在整个过程中重要的元数据一直存放在元数据子服务器中,和云存储平台分离,有效地避免了平台管理员对元数据可能的恶意操作的安全隐患.

传统的 HDFS 文件系统是将元数据文件存放在 NameNode 节点的机器上,其元数据文件存放位置如图 8 所示.利用 ASOM 模型后其持久性元数据文件存放在客户端规定的磁盘位置如图 9 所示.

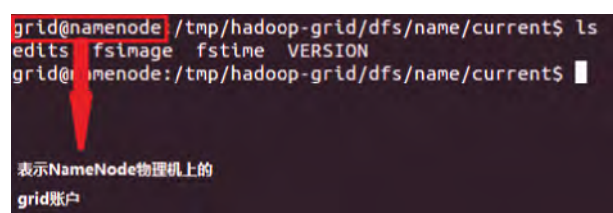


图8 传统 HDFS 元数据文件保存位置

Fig. 8 File location of traditional HDFS metadata



图9 ASOM 元数据文件保存位置

Fig. 9 File location of ASOM metadata

3.2 性能测试

云存储系统的安全提高会在一定程度上降低系统的效率^[7],为此本文设计如下的性能测试对比.对比传统的 HDFS、经加密的 HDFS 和 ASOM 3 种云存储方式的数据传输效率,结果如表 1 所示.

表 1. 数据传输效率表

Tab. 1 Data transmission efficiency

文件大小 /M	文件 类型	加密后上传 /ms	原版上传 /ms	ASOM 上传 /ms
8.56	mp3	1 963	1 089	1 485
33.3	doc	7 357	4 212	5 369
61.7	pdf	15 682	7 508	9 534
145.4	txt	29 893	16 383	18 065
1 007.8	rmvb	117 283	76 320	88 592

表 1 中,加密方式是利用 128 位 AES 在客户端上加密.可以看出当文件越大时该模型的效率比加密更明显.同时,ASOM 模型比传统模型中的信息交互更频繁,但由于 ASOM 模型采用企业单位元数据的自组织管理方式,属于一种元数据分布式组织与管理方法,因此不会给平台服务器带来随客户量增多而产生的负载过重问题.

4 总结

ASOM 模型通过分析 DataNode 与 NameNode 之间的交互过程,针对 HDFS 文件系统中平台对元数据绝对全权管理控制问题,设计并实现了元数据子服务,通过提取 HDFS 中 DataNode 元数据信息直接跟元数据子服务器交互,并且由元数据子服务器处理文件与块、块与 DataNode 之间的映射关系.通过对 HDFS 源代码进行分析编译和模块替换,搭建原型系统进行测试.测试结果表明,对 NameNode 中的元数据进行必要的安全隔离保证了用户对元数据的控制与管理,可有效避免平台管理员对元数据可能的恶意操作的安全隐患.

接下来将着手完善基于 ASOM 模型的原型系统中的用户体验,在存储量增加的同时保证元数据服务器能动态扩充.

参考文献:

- [1] Ghemawat S, Gobioff H, Leung S T. The Google file system [C] // ACM SIGOPS Operating Systems Review. [S. l.]: ACM, 2003, 37(5): 29-43.
- [2] Shvachko K, Kuang H, Radia S, et al. The hadoop distributed file system [C] // Mass Storage Systems and Technologies (MSST), 2010 IEEE 26th Symposium on, Inclineville, Nevada: IEEE, 2010: 1-10.
- [3] Liu K, Dong L. Research on cloud data storage technology and its architecture implementation [J]. Procedia Engineering, 2012, 29: 133-137.
- [4] Zhang D W, Sun F Q, Cheng X, et al. Research on hadoop-based enterprise file cloud storage system [C] // Awareness Science and Technology (iCAST), 2011 3rd International Conference on. [S. l.]: IEEE, 2011: 434-437.
- [5] 张逢喆, 陈进, 陈海波, 等. 云计算中的数据隐私性保护与自我销毁 [J]. 计算机研究与发展, 2011, 48(7): 1155-1167.
Zhang F Z, Chen J, Chen H B, et al. Lifetime privacy and self-destruction of data in the cloud [J]. Journal of Computer Research and Development, 2011, 48(7): 1155-1167.
- [6] 冯登国, 张敏, 张妍, 等. 云计算安全研究 [J]. 软件学报, 2011, 22(1): 71-83.
Feng D G, Zhang M, Zhang Y, et al. Study on cloud computing security [J]. Journal of Software, 2011, 22(1): 71-83.
- [7] 余琦, 凌捷. 基于 HDFS 的云存储安全技术研究 [J]. 计算机工程与设计, 2013, 34(8): 2700-2705.
Yu Q, Ling J. Research of cloud storage security technology based on HDFS [J]. Computer Engineering and Design, 2013, 34(8): 2700-2705.
- [8] 程芳权, 彭智勇, 宋伟, 等. 可信云存储环境下支持访问控制的密钥管理 [J]. 计算机研究与发展, 2013, 50(8): 1613-1627.
Cheng F Q, Peng Z Y, Song W, et al. Key management for access control in trusted cloud storages [J]. Journal of Computer Research and Development, 2013, 50(8): 1613-1627.
- [9] 吴吉义, 沈千里, 章剑林, 等. 云计算——从云安全到可信云 [J]. 计算机研究与发展, 2011, 48(1): 229-233.
Wu J Y, Shen Q L, Zhang J L, et al. Cloud computing: Cloud security to trusted cloud [J]. Journal of Computer Research and Development, 2011, 48(1): 229-233.
- [10] Ray I, Ray I. Trust-based access control for secure cloud computing [M] // High Performance Cloud Auditing and Applications. New York: Springer, 2014: 189-213.
- [11] 林闯, 苏文博, 孟坤, 等. 云计算安全: 架构、机制与模型评价 [J]. 计算机学报, 2013, 36(9): 1765-1784.
Lin C, Su W B, Meng K, et al. Cloud computing security: architecture, mechanism and modeling [J]. Chinese Journal of Computers, 2013, 36(9): 1765-1784.
- [12] Shen Q, Yang Y, Wu Z, et al. SAPSC: Security architecture of private storage cloud based on HDFS [C] // Advanced Information Networking and Applications Workshops (WAINA), 2012 26th International Conference on. Fukuoka: IEEE, 2012: 1292-1297.
- [13] 徐小龙, 周静岚, 杨庚. 一种基于数据分割与分级的云存储数据隐私保护机制 [J]. 计算机科学, 2013, 40(2): 98-102.
Xu X L, Zhou J L, Yang G. Data privacy protection mechanism for cloud storage based on data partition and classification [J]. Computer Science, 2013, 40(2): 1613-1627.
- [14] 文艾, 王磊. 高可用性的 HDFS: Hadoop 分布式文件系统深度实践 [M]. 1 版. 北京: 清华大学出版社, 2012.
- [15] 蔡斌, 陈湘萍. Hadoop 技术内幕深入解析 Hadoop Common 和 HDFS 架构设计与实现原理 [M]. 1 版. 北京: 机械工业出版社, 2013.
- [16] 张鑫. 深入云计算: Hadoop 源代码分析 [M]. 北京: 中国铁道出版社, 2013.
- [17] 傅颖勋, 罗圣美, 舒继武. 安全云存储系统与关键技术综述 [J]. 计算机研究与发展, 2013, 50(1): 136-145.
Fu Y X, Luo S M, Shu J W. Survey of secure cloud storage system and key technologies [J]. Journal of Computer Research and Development, 2013, 50(1): 136-145.