**海量存储课程论文报告**

|  |
| --- |
| **分布式多种底层存储硬件的**  **文件系统研究** |

|  |  |
| --- | --- |
| 专 业 名 称 ： | 计算机软件与理论 |
| 学 生 姓 名 ： | 兰海 |
| 学生学号 ： | 2017202110029 |

二〇一七年十一月

**目 录**

[1 引言 1](#_Toc499976652)

[1.1 论文研究的目的与意义 1](#_Toc499976653)

[1.2 本课程论文调研的方向 2](#_Toc499976654)

[1.2.1 开源社区 2](#_Toc499976655)

[1.2.2 学术界探讨 2](#_Toc499976656)

[1.3 课程论文框架和主要内容 2](#_Toc499976657)

[2 开源社区 3](#_Toc499976658)

[2.1 HDFS-2832 3](#_Toc499976659)

[2.1.1 概述 3](#_Toc499976660)

[2.1.2 增加的功能 3](#_Toc499976661)

[2.1.3 整体方案概述 3](#_Toc499976662)

[2.1.4 思考 4](#_Toc499976663)

[2.2 HDFS-4672 4](#_Toc499976664)

[2.2.1 概述 4](#_Toc499976665)

[2.2.2 整体架构 5](#_Toc499976666)

[2.2.3 实现 5](#_Toc499976667)

[2.2.4 思考 5](#_Toc499976668)

[3 学术界探讨 7](#_Toc499976669)

[3.1 OctopusFS 7](#_Toc499976670)

[3.1.1 概述以及论文贡献 7](#_Toc499976671)

[3.1.2 文件系统API 7](#_Toc499976672)

[3.1.3 自动管理策略-文件放置 8](#_Toc499976673)

[3.1.4 自动管理-文件获取 9](#_Toc499976674)

[3.1.5 自动管理-副本管理 10](#_Toc499976675)

[3.1.6 总结思考 10](#_Toc499976676)

[3.2 hatS 10](#_Toc499976677)

[3.2.1 概述以及论文贡献 10](#_Toc499976678)

[3.2.2 文件系统API 11](#_Toc499976679)

[3.2.3 文件放置以及获取策略 12](#_Toc499976680)

[3.2.4 总结思考 12](#_Toc499976681)

[4 总结思考 14](#_Toc499976682)

[参考文献 15](#_Toc499976683)

# 引言

本章主要介绍本论文的研究目的与研究意义，然后介绍相关的研究工作，之后简述本论文的研究问题，最后介绍本论文的章节安排以及每章的主要内容。

## 论文研究的目的与意义

企业为了提升生产效率以及决策制定，在大规模数据分析相关的系统上有带来巨大技术革新。随着Hadoop生态圈的建立，现在部分的数据分析都是建立在该生态系统上。系统运行在商业电脑中，底层采用分布式文件系统。而且现代大规模数据分析场景下对分布式文件系统的数据存储和I/O有了更高的要求。有前景的趋势是探索在内存、存储介质以及网络上面提升来提供高性能和底层本。在存储介质方面，现在的存储使用了内存、SSD、HDD、网络附接存储（NAS）。不同存储有不同的特点适用于不同的场景以及不同的成本。例如，SSD能够提供提供低的访问延迟以及低的消耗等特性，但是在价格上相比较与传统的HDD，仍旧过高，导致企业成本过高。并且，也不是所有的存储数据都是频繁访问的，因此，对于那些不频繁访问的数据仍旧可以放在HDD上，现在的大型数据中心也是这么做的，当需要时，在传输到需要数据的地方。

将来仍旧会有许多具有不同的存储特点的介质出现，我们有必要在设计分布式文件系统的时候，考虑到这些介质的特点。一方面，我们通过综合考虑这些硬件特征，将数据进行合理的放置，提升文件系统整体的性能（例如，读、写、恢复等）。另一方面，考虑各个存储介质的成本，在保证性能要求的前提下，减少用户成本。例如，我们可以将所有数据放在内存中，不管是访问还是修改，都是纳秒级，完全符合用户的性能要求。内存单位成本是所有存储中价格最高的，同时数据都放在内存中，容量需要也是非常大的，这样对企业成本是非常大的。

因此，非常有必要提出一个考虑底层存储硬件特征的分布式文件系统的需要。而且，也是对当前分布式文件系统一个更加高层的概括以及扩展，提出一个更加高层的统一架构模型。更加满足一般性。

有了一个多层次存储硬件的分布式文件系统，如何针对具体的场景合理利用这些底层设施也是一个非常重要。例如，不同的场景下的数据放置，更希望是自动完成，而不是每次都是由用户来指定。因此，也有必要构建一个工具来利用好多层次底层存储硬件的分布式系统，提供更好的性能。

## 本课程论文调研的方向

本课程论文了调研了当前在开源社区以及学术界对多底层存储硬件在分布式文件系统中实现，使用以及研究现状。

### 开源社区

在ApachedeHDFS社区早在很多年就开始考虑HDFS的底层存储硬件不是单一的磁盘，而是多种不同特性硬件的集合体。和该问题相关的issues有：

* HDFS-2832：enable support for heterogeneous storage in HDFS-DN as a collection of storages[1]
* HDFS-4672：hdfs smart storage management[2]

### 学术界探讨

最新的论文是发在Sigmod2017的论文《OctopusFS: A Distributed File System with Tiered Storage Management》[3]，较为早期的论文2014 International Symposium on Cluster, Cloud and Grid Computing的《hatS: A Heterogeneity-Aware Tiered Storage for Hadoop》[4]。

## 课程论文框架和主要内容

1. 第1章 引言

首先介绍本论文研究目的和研究意义，介绍调研方向，以及介绍本论文的论文框架和各章节主要内容。

1. 第2章 开源社区

本章主要介绍开源社区在分布式文件系统在底层利用多种存储特性的存储硬件的思考以及实现调研。

1. 第3章 学术界对该问题的探讨

介绍一篇关于多底层硬件方面的论文。

1. 第4章 总结和思考

# 开源社区

本章主要介绍开源社区以及公司在分布式文件系统在底层利用多种存储特性的存储硬件以及相关方面的思考以及实现调研。

## HDFS-2832

### 概述

当前（2013年左右）的HDFS只支持一种存储介质，不能识别不同的存储介质，也就导致在应用层面无法选择性使用不同存储特征的底层存储硬件进行数据存储。增加对HDFS对底层存储硬件的识别有利于对应用层面来的数据进行放置来提供更好的性能。该问题就主要描述了在HDFS中添加对多种存储硬件的支持。为了支持不同的存储设备类型，每个DataNode被当做一系列存储硬件的集合。同时，额外增加新的文件API支持对存储介质的使用以及对扩展现有的限额机制。

### 增加的功能

1. 应用在创建一个文件或者发出一个块复制请求时能够指定在一个特定的存储设备上：所有的副本在在同一存储类型上；副本在不同的存储类型上，例如两个副本在HDD，一个在SSD上。
2. 应用能够改变一个文件的存储介质，包括改变所有的副本;部分副本的改变。
3. 管理者能够能够创建在一个目录中，特定硬件介质的限额。
4. 系统能够自动基于访问模式将热数据移动到更快的设备上。

在上面的功能的满足情况下，如果请求的存储类型不可用（不存在），那么会在该存储类型可用后，继续重复尝试。为了保证可靠，可以先维护一定量的副本在其它存储介质上面。其次，管理员一定能够配置存储空间限额，这样能够让资源在所有用户直接进行一种公平的分配。

### 整体方案概述

存储介质的类型有两种分类标准：逻辑上，依赖于存储介质的延迟、吞吐量、成本以及读写模式；以及物理上依赖于我们已知的类型，如SSD等。在本实现方案中采用了物理上的实现方案。作者提到的原因是底层易于实现，同时也易于API的实现。

在DataNode上，每个底层存储有一个标识符以及类型。在心跳期，DataNode会识别其存储，存储类型以及空间利用率。从DataNode发送到NameNode的块报告包含了存储介质类型以及其标识符。当一个块副本被创建时，存储类型作为block-token的一部分，这样保证客户端不会超过NameNode对该客户端在该类型上的存储限额。

在NameNode上，BlocksMap能够识别存储；能够尝试移动块副本到指定的存储类型上；会将存储类型作为getBlockLocations请求的一部分，这样客户端可以根据需要选择从什么介质获取目标块；同时还要确保容量限额的功能执行。

在Client上，提供的API，支持用户用户创建文件时，指定存储类型；以及支持用户修改以存在文件的存储类型；在发送读取请求后，可以根据返回的请求来获取存储文件的类型，来从指定的类型上读取文件。

### 思考

1. 根据具体的存储介质类型来区分有所不足，同样是SSD，但是不同类型的SSD类型差别也很大。
2. 向NameNode报告的信息还是过于简单，效率不高。不利于NameNode选择放置策略。
3. 只考虑了DataNode上多种存储介质的存在，是否在NameNode也可以进行类似的考虑。
4. 硬件没有考虑在DataNode上利用内存来存储数据，有些特别热的数据可以直接放在内存中。

该问题后续还有**HDFS-5682**。

## HDFS-4672

### 概述

该issue主要提出了一个高层引擎Smart Storage Management（SSM）。该引擎能够识别文件属性，数据温度，存储介质，以及硬件容量，用户/应用的偏好来优化HDFS，提供针对场景的端对端的存储策略，为具体的场景提供高性能的服务。最后要达到的效果是：在文件读取之前缓存文件，提高读取效率；当文件变为热数据时，将其移动到高速的存储设备；当文件变为冷数据时，将其移动到慢的存储设备；在文件进行存储前，向客户端推荐存储方案；能够评估存储利用以及性能获益利用块的存储硬件；最后将该引擎整合到云环境中。该issue不是一个实现底层多种硬件管理，而是在如何利用多种硬件来提供更好的服务。

### 整体架构

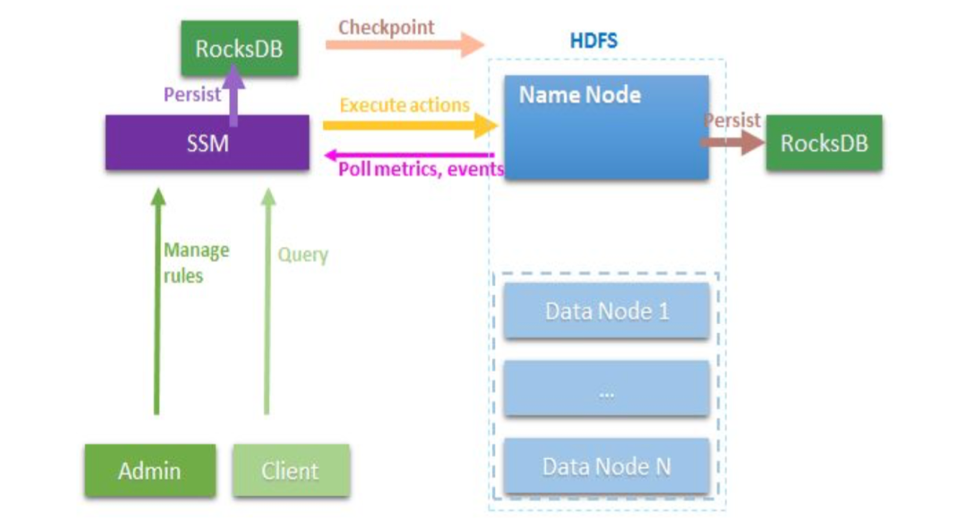
整体架构如图2-1所示。

图2-1

首先实现了一个基于规则的自动引擎，能够执行用户定义的放置规则。等到学到足够多的知识后，可以进行自动管理。SSM从NameNode中获取信息，这些信息由SSM中规则进行分析，如果一些规则的条件得到满足，那么通过NameNode执行相应的操作。使用RocksDB主要是用来存储SSM产生的数据，为了SSM状态恢复时使用。为了防止SSM奔溃，因此也需要在合适的时候进行检查点操作。

### 实现

需要再NameNode做出一下的修改来支持SSM:（1）信息收集逻辑（2）提供给SSM获取NN收集到信息的接口（3）接入一个RocksDB，用于存储在SSM崩溃或者重启期间的数据，因为此时NN一直在产生数据。这里的数据不需要做检查点操作。

SSM利用已经存在的HDFS功能如，HSM，Cache等来执行实际的数据维护工作。

### 思考

本质上将，该issue就是解决一个冷热数据动态存储的问题。但是本issue提出一个新的思考方面：就是在拥有丰富的基础功能上，我们如何能够统一利用这些基础功能。对于冷热数据的移动，对于数据进行均衡分布，对于数据缓存都是常见手段，但是如何针对具体场景提出一个通用的解决手段，抽象出一个更高的层次。

越来越多的企业服务向云上走，分布式文件系统是云服务的一个基础设施，云端自动化管理也是一个大趋势，加上人工智能的兴起，在云上构建一个智能的分布式文件系统是有前景的点。

# 学术界探讨

学术界对多层次硬件分布式文件系统构建和使用也有所讨论，从中抽取了下面两篇论文进行概述。

## OctopusFS

### 概述以及论文贡献

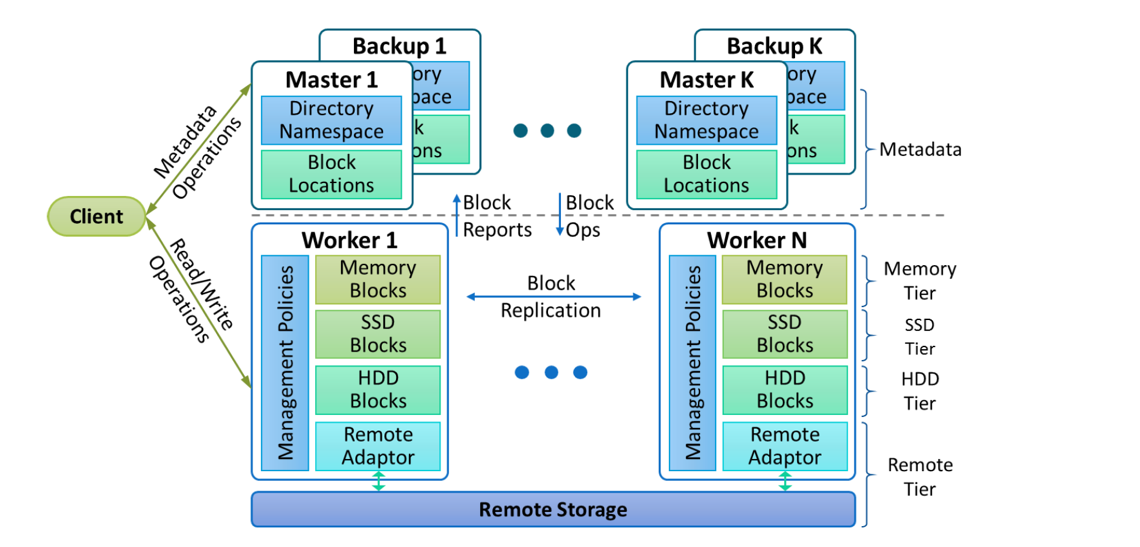
在本论文中，针对分布式文件系统的存储底层上是多种不同存储特性的硬件的集合，提出了一新的系统架构，见图3-1。同时，在HDFS 的基础实现了该架构。

图3-1 系统架构

本论文的主要贡献有三个方面：（1）将底层存储时多中不同特性的存储介质的情况，纳入到了分布式文件系统的具体实现中，丰富了分布式文件系统的能力；（2）对外使用API，只是简单对现有的标准文件API进行简单的扩展，见表3-1；（3）提出自主-自动数据管理策略。一方面客户端能够自主决定副本的放置位置，另一方面，如果客户端没有指定放置位置，文件系统会根据当前系统的整体情况来选择最优的放置策略。

### 文件系统API

文件系统API的设计对现在标准的分布式文件系统API进行了简单的扩展，主要体现在replication vector上面。在本论文中，该向量是一个五元组<M,S,H,R,U>。其中M代表内存，S代表SSD，H代表HDD，R代表remote，U代表随意放置。每个值代表了需要将创建文件时，副本放置位置。例如：<2,1,0,0,0>表示两个副本放在内存中，一个副本放在SSD中。<0,0,0,0,3>用户不指定放置位置，由系统来确定放置位置，就类似于原生HDFS。但是这里，OctopusFS提出了自己的考虑硬件环境的放置策略。

|  |  |
| --- | --- |
| API | Comments |
| FSDataOutputStream create ( Path f , ReplicationVector repVector , long blockSize ) | Creates a file and returns an output stream for writing. The original API uses “short replication” instead of “ReplicationVector repVector”. |
| boolean setReplication(Path f , ReplicationVector repVector) | Sets the replication vector for an existing file. The original API uses “short replication” instead of “Replication Vector repVector” |
| BlockLocation [ ] getFileBlockLocations ( Path f , long start , long len ) | Returns the list of block locations containing the data in the re- quested byte range. Each block location indicates the storage tier. |
| StorageTierReport [ ] getStorageTierReports ( ) | Returns the list of active storage tiers along with useful information per tier (e.g., total/remaining capacity, read/write throughput). |

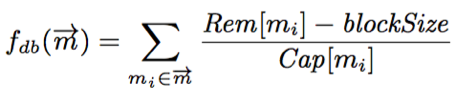
表3-1

在本论文中，对不同存储介质的分类是根据存储介质特性（延迟，吞吐量等）。因此，上面的五元组是可以扩展。可以根据getStorageTierReports API来获取当前环境中的不同介质类型信息。通过使用setReplication API就能实现文件块在不同介质之间的移动。该功能的实现极大的丰富了该系统的使用场景（类似于计算机中的多级存储结构的功能一样）。

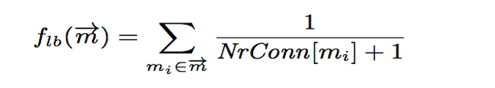
### 自动管理策略-文件放置

1. 问题定义

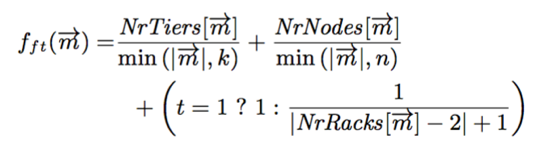
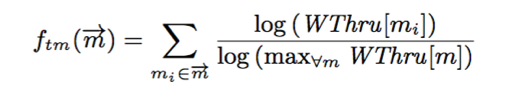
可以将文件放置形式化定义为：给定一系列可用的存储媒介（属于图3-1中所有worker中的所有tier的存储媒介），以及一个文件的副本数目r，通过文件放置策略我们选择一个列表 L=<m1,m2,…,mr>，其中mi ≠ mj。该r个存储媒介用来存储这些副本。

1. 考虑因素
2. 数据平衡：确保在集群中所有可用的存储上数据是均匀分布的。

其中Rem[mi]为媒介mi上的剩余可用空间，Cap[mi]为媒介上总空间。该公式计算了放置后，在存储媒介上还剩的空间百分比的和。显然每次选择剩余空间百分比高的。该值越大越好。

1. 装载平衡：有效地将I/O请求分布到集群中所有的可用的存储上。

其中NrConn[mi]代表媒介mi上的当前I/O连接数越少越好。因此该值越大越好。

1. 容灾能力：发生任何数据丢失，数据仍然能够恢复，系统仍旧可用。（存储数据到不同机架上的不同worker上的不同存储介质上）。
2. 吞吐量最大化：保证数据能够很快的写入到文件系统中。（所以选择越快的写入设备越好）。

以上的相关数据都是在worker每次收集好的数据，发到master中。将上述信息包括了HDFS的心跳包中。

3、问题转化

 将问题转化为求解：

的最大值.

解空间为集群中所有可用存储媒介中r的组合。这是一个多目标优化问题，本论文中采用了贪心算法+启发式规则来进行求解。贪心算法主要体现在每次只放置一个副本，启发式主要体现在不同副本在不同机架等常见的规则。

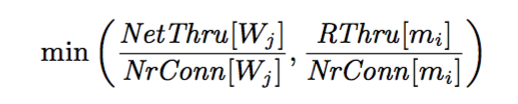
### 自动管理-文件获取

1. 问题定义

给定一个块的若干个副本，通过文件获取策略得到一个排序列表，该列表的一个项是能为客户端提供最好服务的位置。

1. 考虑因素

主要考虑：客户端位置，副本位置，网络拓扑结构，网速(NetThru[Wj])，每个存储媒介上的读输出（RThru[mi]） ，媒介所在worker上的网络连接数（NetThru[Wj]），媒介上的I/O连接数NrConn[mi]。

总和以上的因素，最后也是将该问题转化为一个数学问题:

### 自动管理-副本管理

副本管理主要考虑两个问题：副本丢失以及副本数过多。针对第一种情况的解决，就是放置策略和文件获取策略相结合，通过放置策略选择一放置位置，然后通过文件获取策略获取一个块。针对第二种情况，分布考虑移除每个副本对系统的影响（影响的评价指标，类似于文件放置策略，考虑数据分布均衡，冗余能力等），选择影响最下的副本进行移除。

### 总结思考

该论文设计了一种新的分布式文件系统架构，考虑底层存储硬件多种不同类型；提出一种考虑具体环境下的副本管理策略。而且由于提供了副本的存储介质变化的操作，极大丰富了使用场景，例如根据用户的工作负载已知，我们便能够进行数据预取策略，在用户执行上一个操作时，我们将下一个操作的数据从速度的存储介质上，移动到速度高的存储介质上，提升了整体工作负载的执行效率。

但是仍然有下面的不足：

1. 在自动副本获取策略中，并没有体现出其考虑的全部因素
2. 如果用户指定的存储介质不存在，论文中没有体现出对这种情况的处理讨论
3. 现在考虑了不同的存储介质，在应用层面只考虑不同预取相关的操作，没有考虑架在该分布式系统上其他应用如何利用多种存储介质来提升性能，例如MapReduce时，中间结果如何放置等等。

## hatS

### 概述以及论文贡献

hatS即a Heterogeneity-Aware Tiered Storage，在HDFS中考虑多种层次的存储介质到Hadoop生态系统中。同时，针对新的底层存储硬件提出了新的数据放置和获取策略。论文的主要贡献有：（1）设计并实现了hatS，为DN提供了识别不同存储介质的能力。（2）实现了新的基于存储设备特性的数据放置以及获取策略，这样能够高效的利用高速存储设备。（3）提供了一套API能够满足应用的存储需要以及在不同的层之间移动副本，提供更高性能的服务。（4）在实际的场景中验证了hatS的性能。

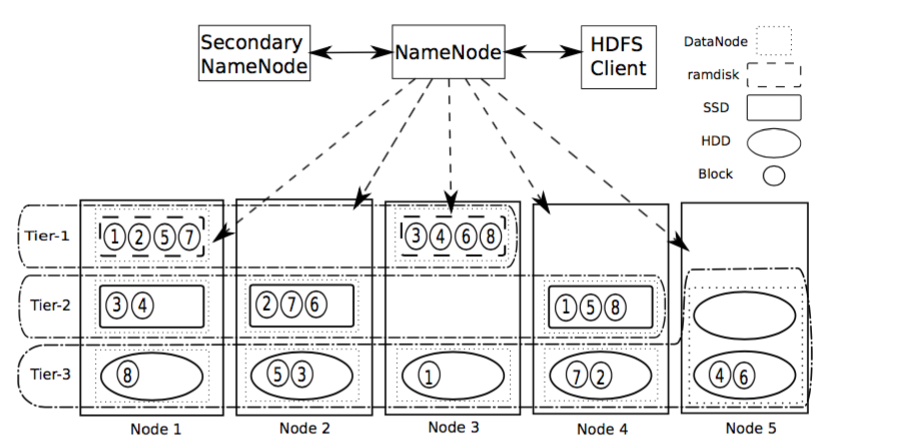
 图3-2是hatS的主要框架。 从图中可以看到，依据不同的存储介质，在DN

图3-2 hatS架构图

中对设计分层的存储层设计。在传统的HDFS中，每个node只有一个DN示例，但是在hatS中，每个node上有多个DN示例，与该node上的存储设备类型相关，例如，如果一个node上有两个HDD，一个SSD，则该node上有两个DN示例，一个对应HDD，一个对应SSD。不同node上的统一存储特性的存储设备被归为同一层，如图3-2左侧的tire-x所示。

### 文件系统API

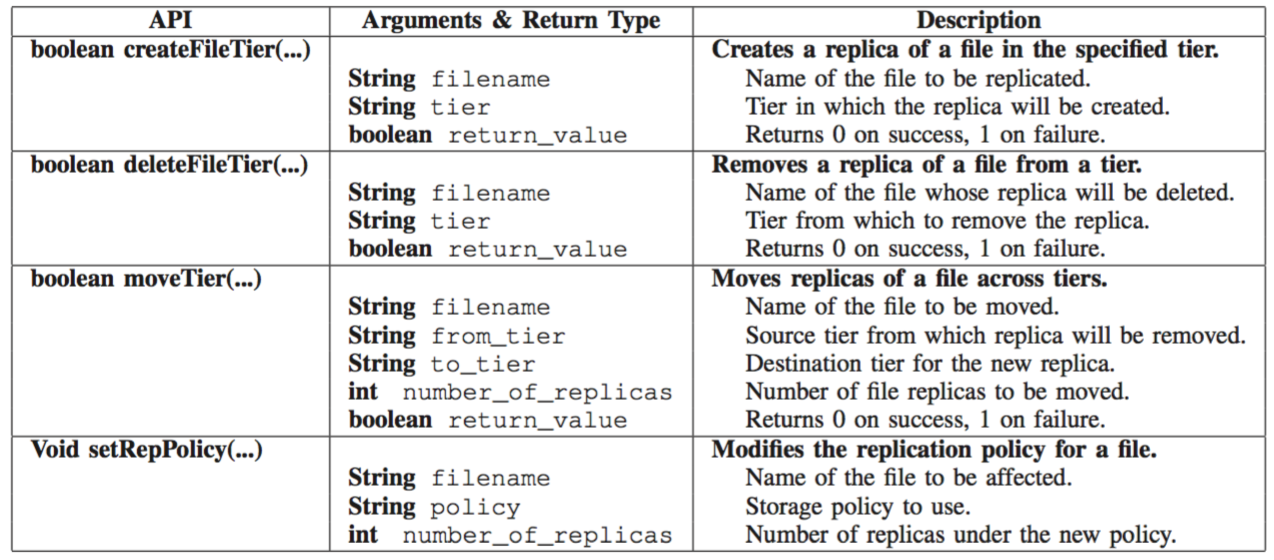
hatS文件系统API的设计并没有扩展现有通用的API，根据自己的功能，定

表3-2

义了新的API，如表3-2所示。注意，在hatS中一个文件整体只能存储在一个层次上，如果想要在不同层次存储文件，需要在该层次也必须存有完整的文件。

### 文件放置以及获取策略

文件放置和文件获取是相关联的，只有放置位置得当，才能提供好的文件获取性能，所以在本论文中将文件放置和获取策略合并为一个策略。论文中考虑了三个不同的策略：（1）Network-Aware策略 （2）Tier-Aware策略（3）Hybrid策略。

Network-Aware策略，数据完全按照原生HDFS的放置策略，在获取的时候按照，就近原则，获取靠近客户端的块，有利于网络负载的均衡。该策略没有考虑底层存储硬件的特性。可能存在的情况，近的块，实际存在低速的存储介质中，而高速存储介质中的块在远端，这对于多种存储介质存在的情况不利。

Tier-Aware策略，将底层存储硬件的特性考虑进来。在文件放置的时候，将文件放置在多个不同的层上，每个层次保存完整一个文件副本。如果有多个层，在最快的硬件层放一个副本，在慢的层次慢一个副本，剩下的随机放置（如果考虑三副本）。没有考虑网络的因素，在获取的时候，总是访问快存储层次上的数据。这种情况下，会导致某些DN过载，影响系统整体性能，而且快存储层次的容量是很受限制的。论文中设计了一个根据IOPS以及容量的获取文件的权值，最后从权值大的存储介质上来读取块。

Hybrid策略，考虑到Tier-Aware策略由于会将副本放在特定的层上，有利于提升I/O性能，Network-Aware会将副本放在不同的机架，有利于整体的性能的均衡。混合策略就是综合两者的特点。第一个副本放在一个local节点上的DN，第二个副本放在与第一个副本位于不同机架，且不同存储层次的DN上，第三个副本放在不同于前两个副本的存储层次，但是和其中的一个副本位于同一个机架上。关于存储层次的选择同Tier-Aware一样。在文件获取的时候，我们将Tier-Aware中的权值和Network-Aware中的网络位置，统一考虑，来获取读取块的位置。

### 总结思考

本篇论文也是设计实现了基于HDFS的考虑底层硬件特性的多层次存储介质分布式文件系统，但是有些地方值得讨论的：

1. 每个node上的每种存储介质对应一个DN，这样不利于扩展，需要手动配置，很麻烦。
2. 文件放置和获取策略考虑的因素过少，而且都是基于规则来考虑，没有看到具体的量化信息。相比较于OctopusFS，还是过于简单。
3. 在文件API的设计上，完全的修改和添加了新的接口，不利于别人的直接替换掉现有的HDFS系统进行使用（就商业目的而言）。

当然，在OctopusFS中涉及的问题，此论文中仍然存在。

# 总结思考

首先，我的方向是数据库，本次调研方向是分布式文件系统，两者虽然不同，但是都是与存储相关的系统。但是他们面临的某些问题是相同。如何提高系统性能（数据获取效率），如何区分冷热数据，如何提供系统的容灾能力，如何利用当前新的存储设备提供更优质的服务。针对新硬件只是其中一个问题，那数据库是否也可以考虑层次底层硬件的问题，都是值得思考和探索的。

其次，在当前的云计算以及人工智能的技术潮流下，都在进行上云工作，都是云存储一部分，而且也都在进行数据管理的自动化，今年SIGMOD一篇CMU的自动化数据库配置的论文点燃了数据库+AI的火，今年Oracle全球用户大会上，发布的oracle18号称是自动驾驶的数据库。分布式文件系统这边亦是如此，提升分布式文件系统自身的智能能力在云上非常有价值。

第三点，在各自针对同一问题的解决方案也可以进行相互参考。例如现在现在出现了很多SQL ON HADOOP也就是在探索两者的结合区域。

参考文献

[1] HSM https://issues.apache.org/jira/browse/HDFS-2832

[2] SSM https://issues.apache.org/jira/browse/HDFS-4672

[3] Kakoulli E, Herodotou H. OctopusFS: A Distributed File System with Tiered Storage Management[C]// ACM International Conference. ACM, 2017:65-78.

[4] Krish K R, Anwar A, Butt A R. hatS: A Heterogeneity-Aware Tiered Storage for Hadoop[C]// Ieee/acm International Symposium on Cluster, Cloud and Grid Computing. IEEE, 2014:502-511.