着重于PC文件系统与人的交互设计智能GDBFS

详细设计报告

张益博，毕昊阳，余浩

Operating System (H) Spring 2017

目录

前言

初衷与目标

在使用个人电脑时，寻找文件是一个最常见的举动。相比于日益攀增的磁盘IO速率，人在不同文件夹中翻寻找文件的速率可谓是效率极低。我们相信未来人们在使用个人电脑办公时，可以做到我告诉电脑我要什么样子的文件，电脑就返还我什么文件。换句话说，我们的理想目标是，用户只需要往这个盒子里扔文件，和通过描述来找文件，而不用管分类、存放位置等细节。

如何提升人搜寻文件的效率，现在各大厂商已经有了解决方案，如macOS的Spotlight与Windows的索引服务，皆是用一个索引程序搜寻磁盘上的文件并建立索引。然而这种做法恐怕会增加CPU的负担，并且难以做到更好的效果，因为它与文件系统是分离的。正如一个不住在房子里的管家，不知道什么时候有人放了什么东西到房子里，所以他要经常跑到房子里搜寻有什么东西改变了并记录。那么为何不让这个管家住在房子里?

于是我们想到了做一个文件系统，因为一般的读写请求都将由文件系统操作，于是我们可以知道在这个文件系统上的所有操作，从而方便统一管理。

接下来，要准确的知道用户描述的文件是什么，需要文件系统对每个文件了如指掌。比如用户要“我去年放进来的欧美的曲风激烈的男歌手唱的歌”。其中属性有“去年的”、“我创建的”、“欧美的”、“曲风激烈的”、“男歌手唱的”以及“歌曲”这么多属性，并且属性之间有依赖关系，比如“男歌手唱的”属性属于“歌曲”属性。复杂的属性以及属性依赖关系，将为人工智能提供莫大的施展空间，来提供人搜寻时最好的体验。为了处理复杂的属性及属性依赖关系，我们选择了图数据库来处理。

因此，本次大作业，我们选择在Linux平台上基于图数据库写一个着重于人与文件系统交互的GDBFS（Graph Database File System）。

背景调研

DBFS的基本概念和设计理念

DBFS是一种新类型文件系统. 事实上, 与其称它为文件系统, 不如称它为文档系统. 它的设计焦点集中于用户, 目标是 使得用户的文件管理更加轻松.

计算机发展到今天, 人类逐渐开始关注如何让计算机像人类一样“思考”. 对于　文件系统来说, 我们也希望它能像人类一 样管理文件. 本小组认为, DBFS是向该目标的一次迈进.

人脑对于文件的搜索并不是层次化的; 或者说整体人脑思维都并不是这样. 当我们看见一只兔子, 我们并不会像这样思 考: 这是一只哺乳类动物 -> 它属于兔形目 -> 它是兔科. 取而代之地, 我们脑中大体是这样的思考过程: 它的耳朵很 长(一种特性), 它的体型不大(另一种特性)... 诸如此类, 我们脑中零散地闪现出一系列该动物的特性 (characteristics), 经过与经验和知识的匹配, 发现只有一种动物符合所有这些动物: 兔. 于是我们说, 这是一只兔子. 而上述过程正如一个数据库的搜索过程. 我们相信, 当用户在使用计算机, 尤其是PC时, 类数据库的文件系统系统的索 引方式, 在交互体验足够好的情况下, 能让用户更好地管理自己的文件. 这就是我们研究和设计DBFS的理念.

DBFS最早由Onne Gorter在他的硕士论文开发。它现在是一个sf.net项目。

在千兆字节的文件中查找文件很困难。基于层次结构的文件系统创建位置，除非你知道在哪里看，你将找不到你的文件。 DBFS尝试将查找文件的责任置于计算机上，因此您不必记住存储在哪里。

看看苹果正在试图用他们的Spotlight→或微软与他们的WinFS→。或者看谷歌→，iTunes→等等（像这样的公告→）。搜索是新的浏览。DBFS通过脱离目录而采取不同的方法。仍然所有这些不同类型的搜索技术在DBFS（将来）中绝对有一个地方。

但是除了搜索之外，还有更多的东西可以使文件管理更容易。以文件为导向。在DBFS这是两倍。首先，DBFS不存储系统文件：没有共享库，没有字体文件或类似的其他文件。这些不是文件，而不是每天查找的文件，并且在文件系统中没有位置。其次，因为需要彼此而属于一起的多个文件实际上是一个文件。如果你有电影文件是第一部分和第二部分，实际上是一部电影;两个文件，一个文件。

DBFS的现状

1. sf.net项目

DBFS最先由University of Twente的Onne Gorter进行开发, 现已成为sf.net项目. 3.2.2 KDE的DBFS实现
KDE的core中已经加入了DBFS. 打开文件和保存文件界面已经被DBFS替代.

但那并不是纯净的DBFS, 底层仍然是传统层次化的文件系统, 一个文件管理器应用程序(桌面应用)将底层文件系统进 行抽象, 向用户呈现出了DBFS的表象.

这些工作都是sf.net项目的一部分.

由于上面提到的原因, KDE的DBFS的实现并不是完美的.

2. WinFS的DBFS实现

WinFS(全名Windows Future Storage)是以关联式资料库为基础之资料储存与管理系统的代号名称，它由微软开 发，在年首次用于2003的Microsoft Windows作业系统中的进阶储存子系统，它针对结构化，半结构

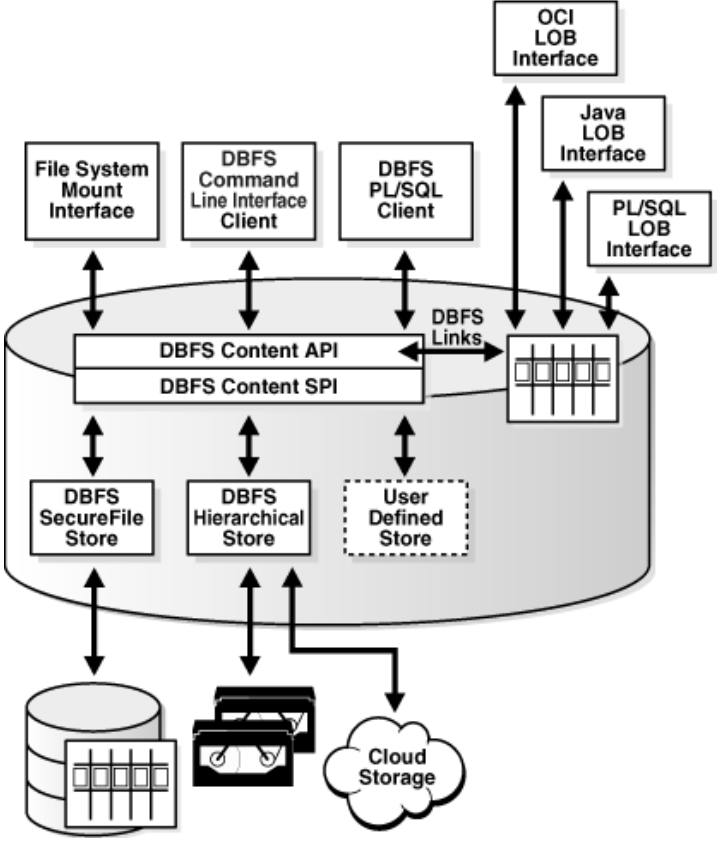
化与未结构化 资料的保存与管理用途而设计. WinFS文件系统是众多为“Longhorn”技术的基础，而且将包含在下一版本的视窗中。原本WinFS文件系统应在 Windows Vista中发行后提供，但相关计画后搁置，然而其某些元件技术已整合到即将发行的ADO.NET与Microsoft SQL Server的. 史蒂夫·鲍尔默曾公开称WinFS文件系统仍在开发，至于此技术将以何种方式提供仍未知.

3. Oracle DBFS

Oracle数据库通常用于存储与数据库应用密切相关的文件，包括CAD，医学图像，发票图像，文档等。应用程序使用SQL标准数据类型BLOB（和CLOB）将文件存储在数据库中。与传统文件系统相比，Oracle数据库提供了更好的安全性，可用性，鲁棒性，事务和可扩展性。当文件存储在数据库中时，它们将被备份，使用Data Guard同步到灾难恢复站点，并与数据库中的关系数据一起恢复。这使得数据库中的存储文件成为许多应用程序的吸引人的选择。

在Oracle Database 中，Oracle引入了Oracle SecureFiles LOB。SecureFiles LOB为文件提供高性能存储，与传统文件系统的性能相当。SecureFiles LOB支持对文件进行压缩，重复数据删除和加密的高级功能。由于SecureFiles LOB保持与BLOB（和CLOB）的向后兼容性，所以针对BLOB编写的应用程序继续透明地使用SecureFiles LOB，即使使用前面提到的功能。

数据库文件系统（DBFS）利用数据库的功能来存储文件，数据库的优势在于有效管理关系数据，为存储在数据库中的文件实现标准的文件系统接口。使用此界面，将文件存储在数据库中不再局限于专门用于使用BLOB和CLOB编程接口的程序。现在可以使用作用于文件的任何操作系统（OS）程序来透明地访问数据库中的文件。例如，ETL（Extract，Transform and Load）工具可以将数据库中的暂存文件透明地存储在数据库中。



Oracle DBFS 的结构是我们参考的对象。

之前对于DBFS的尝试都是从搜索方面来提高交互效率。搜索固然是一个提升方向，但是我们的角度是以预测用户意图的角度来提升交互效率。或者说，推荐。当然，这两者是可以结合的。

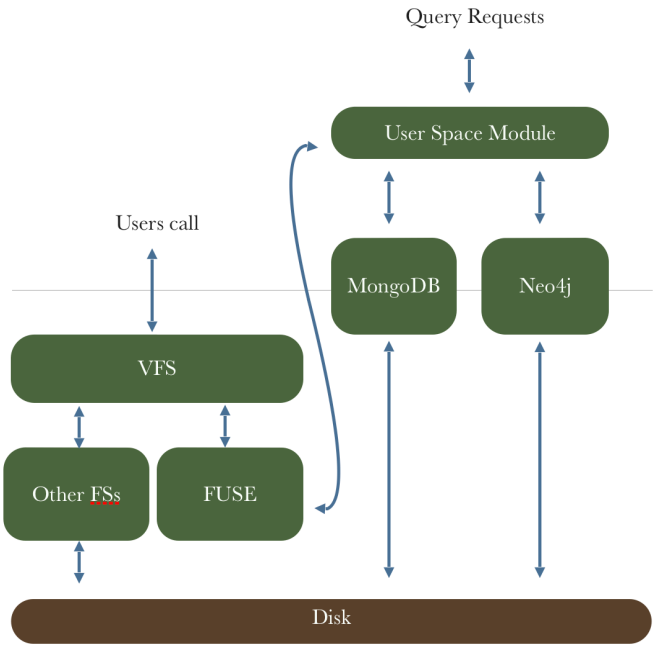
可行性调研

一、 技术设计与原理

1. 数据库选择目前最好的图数据库—Neo4j。

2. 为了在Linux上实现通用文件系统，需要与VFS对接。因此我们选择FUSE (Filesystem in User Space) 来帮我们实现这件事。这样可以兼容原有文件系统，使应用程序可以自然的工作在我们的文件系统上。

3. 为了提高用户搜寻的效率，我们需要做一个CLI的Client来完成用户与文件系统的交互，而不通过VFS。



左图是我们对如何实现我们所要文件系统的图示说明。这里这里有几点说明：

1. 由于Neo4j适合存复杂的属性以及属性关系，而不适合存储如视频文件的BLOB文件（Big Binary Object），因此我们需要另一个存取文件的方式。就是图示中IO Methods所表示的模块，这里我们经过调查，可以选择MongoDB或者还是原有的文件系统。由于效率依具体情况而决定，并且文件读写速率并不是我们注重的重点。所以这点的选择不在可行性报告中具体决定。

2. FUSE由C语言写成，而Neo4j由Java写成，但这两个都提高各种语言的接口，所以我们可以选择合适的语言来完成我们的项目。

3. 由于层次比传统文件系统更复杂，因此我们不期望文件读写效率比原有文件系统高，我们的文件系统更偏重于人在使用个人电脑时对文件搜寻效率的提升，从而提升人操作的使用体验。

二、理论设计与原理

设计上面临的最大问题，是如何设计属性，属性的依赖关系，属性的性质，以及如何跟原有文件路径映射。

Neo4j图数据库中，有一下结构：

Entity：

- Node（可有多个Label）
- Relation（可有一个Relation Type）

Path：节点路径

Token（字符串）：

- Label（分配给Node）
- Relation Type（分配个Relation）
- Property Key（to identify a Entity）

因此，我们打算将文件与属性作为节点，属性之间有从属关系等关系构成图。

我们打算将属性分成三类：

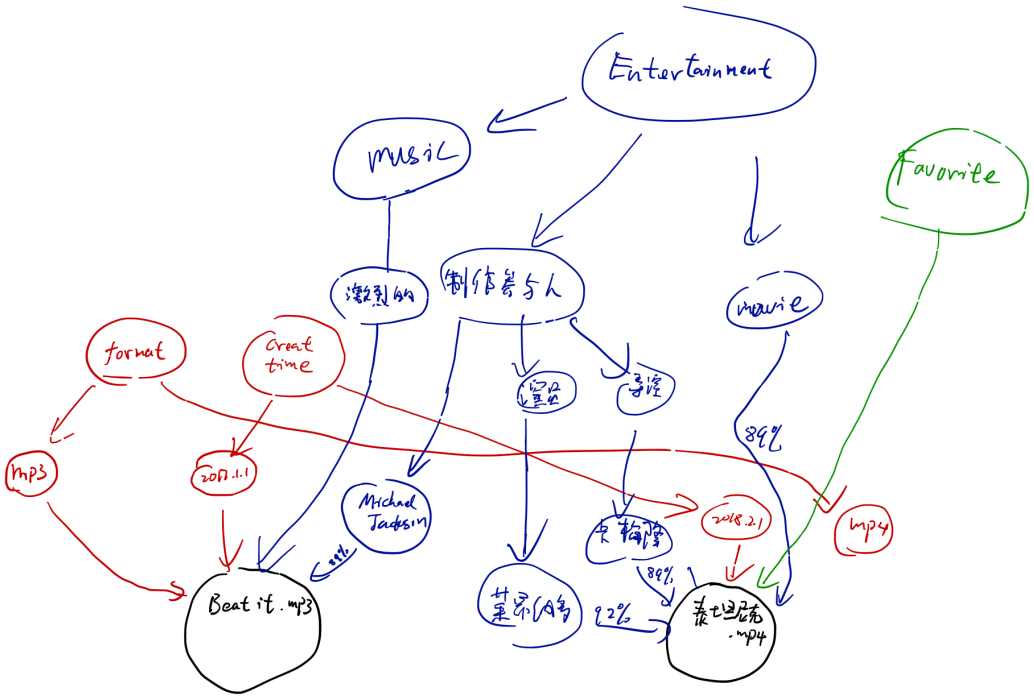
1. 固有属性（如格式，创建时间等）。

2. 推测属性（由文件系统分析文件后推测出的属性，如一首mp3的风格，一个pdf文件是小说论文亦或教材等）。推测属性这个功能可开放给程序员写识别模块后建立自己的推测方法与推测属性。文件的推测属性可有程度值，因很多情况推测是有置信程度的。

3. 用户自定义属性（如喜爱属性等）。

需要说明的是，推测模块的具体实现，已与操作系统课程无关，我们会视具体情况实现。

为了更清晰的体现设计理念，我们有如下草图所画的例子：



红色表示固有属性

蓝色表示推测属性（由Entertainment推测模块产生，百分比表示置信程度）

绿色表示用户自定义属性

将属性如此规定后，将形成树形结构，与路径的映射也不会太难。而是否再在属性之间加入平行的关系，先不置定论。

同时，关系也需分为两种：

1. 动态关系：随着用户的使用，动态关系会随之更改长度值，删除或者增加，以学习预测用户意图。

2. 静态关系：随着用户的使用，动态关系会随之更改长度值，但不会删除或者增加。这种关系的存在是为了不丢掉属性之间的通用性和为了方便与传统文件系统逻辑结构的映射。

在这种设置下，文件系统属性与文件有一个骨架结构（由静态关系构成），这个结构可以直接与普通文件系统逻辑映射，也提供了文件系统

逻辑的通用性。动态关系由学习算法产生，负责学习用户的习惯，并对未来用户可能做出的操作进行预测。具体算法在后续详细介绍。

三、应用场景设计

我们的文件系统可挂载在现有Linux文件系统的一个节点下，兼容原有文件系统的各种操作。又由于其高效率的人与文件系统交互，可作为用户储存日常文件的地方。

四、创新与优势

1. 我们着重于人与文件系统等交互，并着重利用人工智能的潜力。

2. 目前成熟的DBFS是Oracle DBFS，其建立在自家的关系数据库上，难以处理复杂而灵活的属性。我们则选用了的成熟的图数据库，可以快速处理复杂多元属性及属性关系。

3. 高可扩展性，第三方可开发模块来添加推测属性以及推测方法。（如Adobe可为其下的Photoshop开发一套推测属性以及推测方法，来自动为ps文件添加属性）。

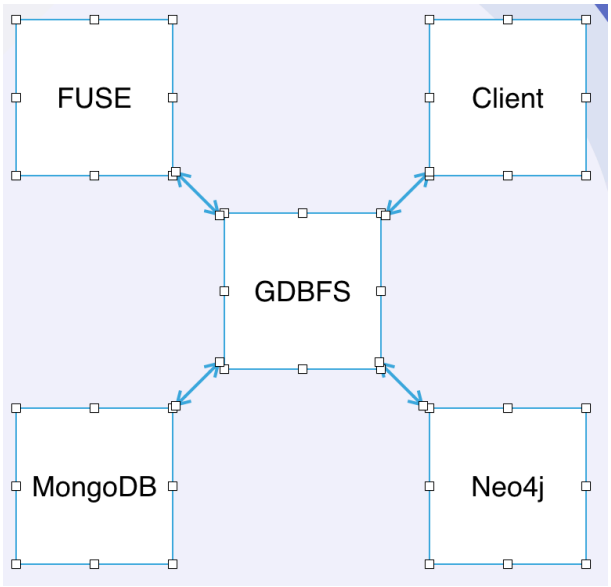
4. 文件系统在用户态实现，可以轻松跨平台。

详细设计

结构设计

为了在用户态实现跨平台文件系统，我们采用 python 语言。需要的工具为 FUSE，Neo4j，MongoDB。相应的python工具为 fusepy，py2neo，pymongo。

在此结构下，可以通过两种方式来访问我们的文件系统。一是通过Client直接操作，可以看到文件系统的所有内容（动态关系与静态关系），能够看到动态关系就代表了可以被推荐内容。而另一个方式是通过FUSE，这是为了兼容原有文件系统与兼容现有的所有应用程序而设定的，这并不对我们的学习与推荐机制有帮助，但它确实必不可少的。



算法设计

算法上分为两部分，一部分是对用户行为的学习，然后作出预测推荐。另一部分是对新加入文件的分类，即智能连接动态关系。

第一部分是我们研究的重点，而第二部分现在已经有充分的算法来做各种分类器了。

针对第一部分，首先我们对问题建模。

在这里，我们不区分属性与文件，而将所有节点一视同仁。我们也不去分辨动态关系的意义，而将其视为由统计得到的推荐机制。

将用户对节点的访问视为一串节点串，我们希望能够在这一长串的节点序列中学习到用户的习惯，并对用户接下来的要访问的节点进行预测。

也就是，根据节点的历史序列求出：

$$P(node_i|node_k,node_{k-1}\dots)$$

可想而知，对于文件系统中节点访问的序列，当用户访问完一个序列后，他接下来会大概率访问的节点并不多。也就是，问题的转移矩阵是一个非常稀疏的矩阵。因此我们用图来储存这个转移矩阵并不会带来太多的代价。为了在保证效率的基础上减少复杂度，我们采用一阶马尔可夫模型（First Order Markov Model）与基于LZ78算法进行分词的变阶马尔可夫模型（Variable Order Markov Model）。然后采用 Voting 机制来综合两个模型的结构共同决策。一阶马尔可夫模型的优势在于对于变化响应的迅速，而变阶马尔可夫模型的优势在于记忆性。

需要提的是，我们个人PC使用的个人文件数大概在几百几千左右（不考虑系统文件的话），并且会相对频繁的删除与添加。因此神经网络恐怕不是很好的选择，而节点数相对比较少的这个问题，又让我们可以不用常规针对大数据库搜索请求的推荐算法。

1. Markov Model （MM）：

目标为学习出：

$$P(node_I|node_j)$$

我们对概率比较高的项建立关系，而不理概率比较少的节点对。关系有一个数值length表示距离，当我们保证

a）一个节点的出度为1时，这条出去的弧length为1.

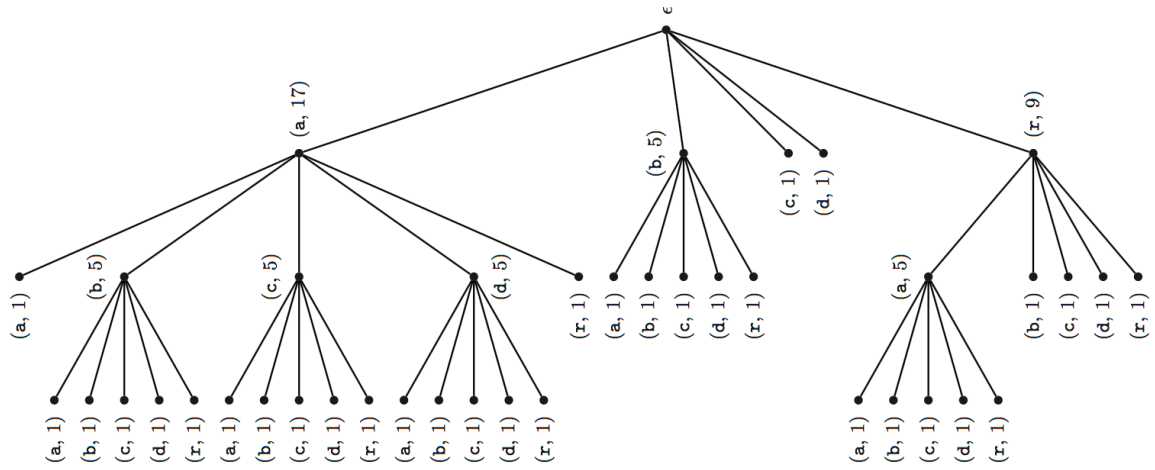
b）当用户从一个节点访问了另一个节点时，我们用如下公式更新各出弧度length值，甚至创建与删除弧：

$$\frac{1}{l_i} := \frac{1}{l_i} + \alpha(1 - \frac{1}{l_i})y_i - \alpha\frac{1}{l_i}(1 - y_i)$$
$$for \quad i \in (1,d(node))$$

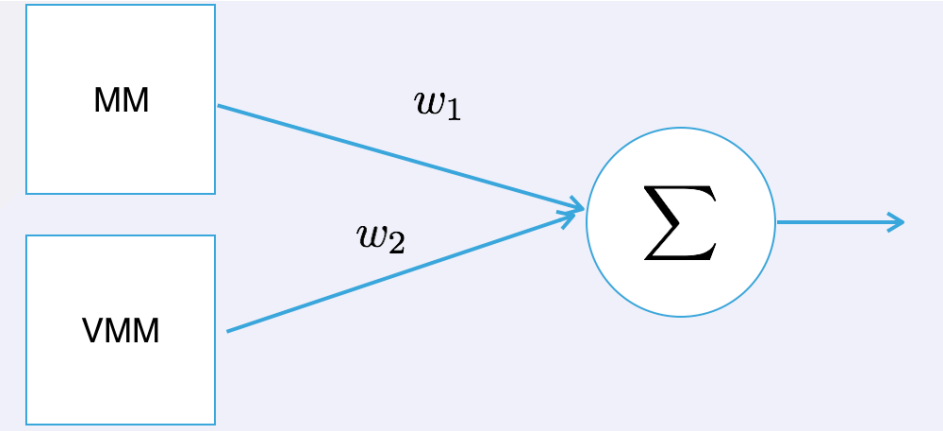
这种设置当好处是，我们保证了$\frac{1}{l}$是对转移概率的估计。同时，若距离过大则删去此弧，这保证了转移矩阵一直是稀疏的。这种模型的缺陷在于马尔可夫性质，即没有记忆性。因此我们需要第二种模型来帮忙。

2. Variable Order Markov Model（VMM）

我们使用LZ78算法的分词机制，帮我们确定所预测序列的长度（而不只是1），然后用概率统计的方式来学习。



然后，采用Voting机制来使这两个模型共同决策：



我们在MATLAB中对这个算法进行了编写与模拟，测试了它的效果。

对于如何测试，我们根据文件系统推荐的性质（生成一个列表，用户要的越靠前表示我们的推荐结果越好），我们采用如下指标作为效果参数：

$$\sigma = \frac{1}{m}\sum_{i=1}^{m}\frac{1}{k_i}$$

其中k为用户点击的我们列表排列中的序数，取m次做平均以表示一个平均的预测效果，并且平滑化我们的曲线。

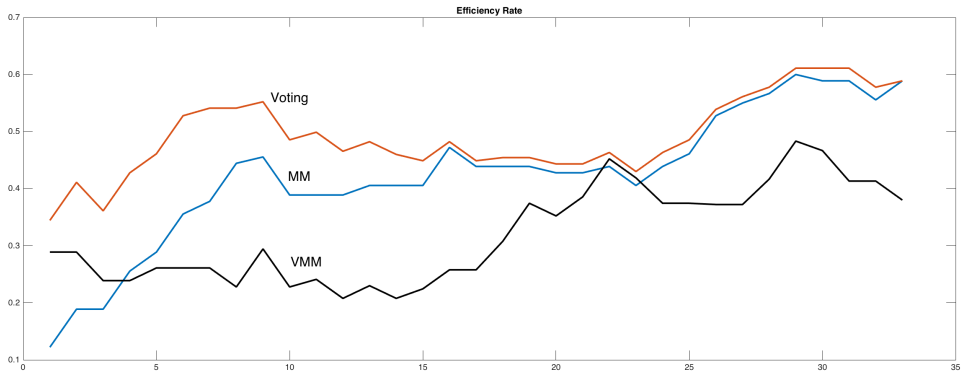
结果我们放在接下来的成果展示中说明。

成果展示

1. 预测准确性结果

采用上节所说的算法与测试算法，针对一个简单的访问模拟序列（我们提交了MATLAB代码，运行其中的脚本输入任意序列即可得到结果），我们得到了如下结果：

可以发现，随着操作步数的增加，效果参数的值总体呈上升趋势。令人意外的是，Voting之后的结果确实比两个算法单独的出的结果要好。



2. GDBFS访存效率测试

采用数据库进行储存，必然会与采用文件系统来储存BLOB文件在访存效率上有所不同。

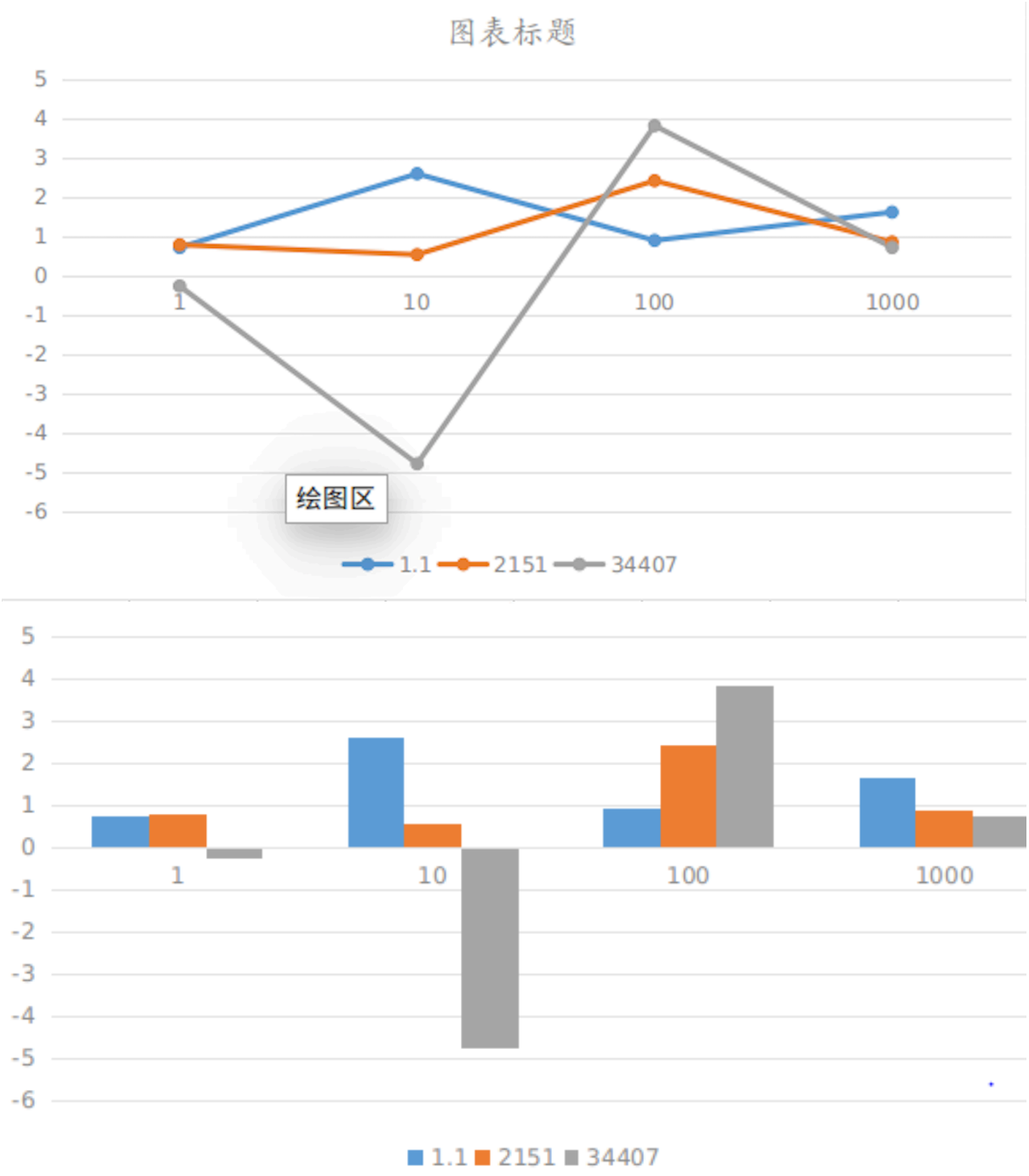
我们测试了不同BLOB文件大小以及不同读磁盘线程数对读文件效率的影响：

结果显示了如下几点：

1）文件越小，MongoDB的读操作越快。

2）在一定范围内线程数越多，MongoDB相对VFS系统调用的读取效率越快。

3）当线程数过多时，由于系统颠簸的发生，MongoDB读取效率相对VFS系统调用又降了下来。



3. 工程建设

我们完成了与FUSE在基本操作：
1. readdir
2. getattr
3. read
4. create
5. mkdir
6. write

完成了Neo4J的许多操作：
1. 创建删除与修改文件节点
2. 创建删除与修改属性节点
3. 创建删除与修改节点间关系
4. 对节点搜寻关系的匹配
5. 获取节点信息
6. 获取关系信息
7. 写了基本的CLI操作，为Client的实现提供了基础
等等

完成了许多MongoDB对BLOB的操作：
读取BLOB文件
写BLOB文件
修改文件信息
等等

完成了在MATLAB测试算法可行性与算法效果

完成了测试MongoDB读取效率的测试

对文件系统的功能进行了测试，并使文件系统达到了初步可用的阶段。但并未实现我们之前设计的所有功能。如客户端与学习算法未全部完成。

结语

这次大作用，我们从一开始什么都不知道的小白，甚至说计算机小白（三位组员均为转院生，计算机学习可以说从大二才正式开始），到最后能够走出一条独特的设计路线，并对设计进行了深入的研究与测试。从最开始各种学习，到看论文并逐步提炼自己的idea，我们都投入了十分多的时间。在设计上投入了过多的时间，虽然导致最后的工程并不完善，但我们也尽了全力并收获了很多。值得一提的是，这些都离不开老师与助教的帮助。期间我们访问与咨询了老师与助教多次，也获得了很多翔实的建议。因此，我们十分感谢老师与助教们的栽培。

我们自己十分希望我们的研究能做出对未来文件系统是什么样子的预测，甚至希望我们的idea能被采纳进其中，然而时间有限能力不足，对于这门课而言这个大作业只能止于此。

但是我们希望，我们的目标：

“一个知道你要什么，并且能预测你要什么的智能文件系统”

将会在不久的将来出现，造福全社会人类。

参考资料

[1] http://dbfs.sourceforge.net/

[2] https://en.wikipedia.org/wiki/List_of_file_systems

[3] https://sourceforge.net/projects/dbfs/

[4] https://en.wikipedia.org/wiki/Database

[5] http://www.skytopia.com/project/articles/filesystem.html#intro

[6] 《an introduction to machine learning》

[7] Human Intent Prediction Using Markov Decision Processes

Catharine L. R. McGhan1, Ali Nasir2, and Ella M. Atkins3

Autonomous Aerospace Systems Lab, University of Michigan, Ann Arbor, MI, 48109

[8] On Prediction Using Variable Order Markov Models

Ron Begleiter

Ran El-Yaniv

Department of Computer Science Technion - Israel Institute of Technology Haifa 32000, Israel

Golan Yona

Department of Computer Science Cornell University

Ithaca, NY 14853, USA