University of Science and Technology of China

GBFS*图文件系统

丁峰　牛田　谢灵江　张立夫 †

2018 年 4 月 5 日

*Graph Based File System

†丁峰：PB16110386　牛田：PB15010419　谢灵江：PB16111096　张立夫：PB15020718

目录

1 小组成员

小组成员介绍:

丁峰 计算机学院大二学生，熟练使用 google scholar 查找文献，学习过 C 语言、数据结构、计算机系统概论，初步了解 linux 内核编译

牛田 学习过 c 语言，以及所有计算机学院前期的基础课程。初学 java、python、html。

谢灵江 计算机学院大二学生，学习过 c 语言，以及所有计算机学院前期的基础课程。

张立夫 掌握编程语言：C C++ Python HTML CSS JavaScript（不熟练），参与过前端开发

总体来说，组员掌握的技能都比较基础，后期项目实现过程需要进行大量相关知识的学习。

2 项目背景

2.1 传统的文件系统

什么是文件系统? 计算机的文件系统是一种存储和组织计算机数据的方法，它使得对其访问和查找变得容易，文件系统使用文件和树形目录的抽象逻辑概念代替了硬盘和光盘等物理设备使用数据块的概念，用户使用文件系统来保存数据不必关心数据实际保存在硬盘（或者光盘）的地址为多少的数据块上，只需要记住这个文件的所属目录和文件名。在写入新数据之前，用户不必关心硬盘上的那个块地址没有被使用，硬盘上的存储空间管理（分配和释放）功能由文件系统自动完成，用户只需要记住数据被写入到了哪个文件中。

严格地说，文件系统是一套实现了数据的存储、分级组织、访问和获取等操抽象数据类型（Abstract data type）。

2.2 标签文件系统

传统的文件系统的存储方式为层次化的，文件的内容由存储在文件系统中的一系列位构成。索引结构提供文件的路径，路径充当了访问文件内容的密钥，文件路径是从上到下一直到底的，构成了树形的结构，每个路径可以分解为“目录 + 文件名”

“标签”，有时也叫“关键字”，可以是任意的字符串，用来组织一组对象。“加标签”(tagging) 通常是一种与“层次结构”(hierarchical system) 不同的组织方式。在层次结构文件系统中，文件只能位于一个特定的目录下。如果在其它目录下需要引用此文件，通常需要特别的操作，如 Unix 系统上的硬或软链接。而在标签式文件系统中，对象可以附加任意数量的标签；然后可用特定的标签来寻找所有带此标签的对象。在更复杂的系统中，可以将标签的集合运算结果作为查询对象的标准。

而有别于传统的层次文件系统的标签文件系统对于用户而言，采取了不透明的存储结构，对于特定的文件，用户可以添加标签（tag）来对文件进行标识，并可以根据相关标签来对文件进行检索。

标签文件系统可以虚拟文件系统实现在传统的层次文件系统的基础上。TagFS 主要有三类实现方式：

- 第一类 TagFS: 与 Fat32, NTFS, Ext3, ZFS 等实用文件系统相等价的文件系统。不过，通常在 TagFS 上，目录即标签。
- TagFS 模拟器: 实际文件保存在现有的文件系统上，用户通过一层标签式接口来访问这些文件。这个思路与 iTunes, Calibre 的实现类似。这种 TagFS 就是一个用户界面，用户通过相应的文件属性数据库 (如 Calibre 的 metadata.db) 访问这些文件。
- TagFS 仿真系统: 以“用户空间文件系统”为后端实现仿真，用户体验上同第一类 TagFS.
- “用户空间文件系统”(Filesystem in Userspace, 简称 FUSE) 是操作系统中的概念，指完全在用户态实现文件系统。FUSE 在 Linux 内核中提供支持，在其它系统上也有相应的实现，如 FreeBSD, Mac OS X 以及 Windows.

2.3 语意文件系统

什么是语意文件系统? 语义文件系统是一种为系统提供灵活的关联访问的信息存储系统。通过从文件中自动提取其内容、属性、类型来确定并构建数据，而不是像当前文件系统那样依靠位置构建。这样的好处是可以提供对系统内容的灵活关联访问，而自动索引是在创建或更新文件或目录时执行的。

举个例子，假设用户的“/相簿”目录下有“/2016 照片”和“/2017 照片”两个文件夹，分别存储了 2016 以及 2017 年的照片，如果用户在传统文件系统中，检索“/2016 到 2017 照片”这个文件夹，是找不到的，因为在文件系统中不存在这个名字的文件夹，只有“/2016 照片”和“/2017 照片”两个文件夹，但是语义文件系统可以根据用户提供的“2016 到 2017 照片”的语义，来同时输出“/2016 照片”和“/2017 照片”两个文件夹。

文件和目录的自动索引之所以被称为“语义”，是因为可编程”传感器”使用关于文件系统对象的语义信息来提取索引。通过使用专门的”传感器”，语义文件系统“理解”系统包含的文档，程序，目标代码，邮件，图像，名称服务数据库，参考书目和其他文件。例如，C 程序的传感器可以提取程序导出或导入的过程的名称，过程类型以及程序包含的文件。语义文件系统可以通过增加专门的传感器来轻松扩展。

优点? 联合访问可以通过帮助用户发现和查找程序，文档和其他相关对象，让用户更轻松地共享信息。例如，可以根据传感器生成的属性（如作者，导出或导入过程，包含的单词，类型和标题）来定位文件。由于语义文件系统与现有的树形结构文件系统兼容，因此语义文件系统的实现可以与现有的网络文件系统协议（如 NFS 和 AFS）完全兼容。NFS 兼容性允许现有客户端机器无需修改即可使用语义文件系统的索引和关联访问功能。通过 NFS 存储在语义文件系统中的文件将被自动编入索引，并且查询结果集将在 NFS 名称空间中显示为虚拟目录。这种方法通过允许将现有的 UNIX 文件服务器透明地转换为语义文件系统来直接解决现有 UNIX 文件系统的“dusty data”问题。

2.4 知识图谱

2.4.1 产生

搜索引擎是人们在线获取信息和知识的重要工具，最初的搜索引擎工作方式简单，即用户输入查询词然后搜索引擎返回它认为与这个关键词最相关的网页。直到 2012 年 5 月，搜索引擎巨头谷歌在它的搜索页面首次引入“知识图谱”：用户除了得到搜索网页链接外，还将看到与查询词有关的更加智能化的答案：例如，当用户输入“Marie Curie”（玛丽居里）这个查询词，谷歌会在右侧提供居里夫人的详细信息，如个人简介、出生地点、生卒年月等，甚至包括一些与居里夫人有关的历史人物。在谷歌推出知识图谱之后，美国的微软必应，中国的百度、搜狗等搜索引擎公司纷纷宣布了各自的类谷歌知识图谱的产品，在此方向开始迅速的发展。因此，最早知识图谱是由谷歌推出的产品名称，现在的知识图谱已被用来泛指各种大规模知识库。

2.4.2 构建

大规模知识库 以词条作为基本的组织单位，由世界上的编辑者协同构建的知识库，在数量、质量、更新速度上都已非常惊人，称为人类获取知识的主要来源之一。除了维基百科等大规模在线百科以外，各大搜索引擎公司和机构还维护和发布了其他各类大规模知识库，如 Freebase，DBpedia，YAGO 等。

互联网链接数据 国际万维网组织 W3C 在 2007 年发起了开放互联数据项目（Linked Open Data，LOD），该项目旨在将由互联文档组成的万维网（Web of documents）扩展成由互联数据组成的知识空间（web of data）。LOD 以 RDF（Resource Description Framework）形式在 Web 上发布各种开放数据集。LOD 还允许在不同来源的数据项之间设置 RDF 链接，实现语义 Web 知识库。

互联网网页文本链接 很多研究者致力于直接从无结构的互联网网页中抽取结构化信息。华盛顿大学的 Oren Etzioni 教授主导的“开放信息抽取”（open information extraction，OpenIE）项目，以及卡耐基梅隆大学的 Tom Mitchell 教授主导的“永不停止的语言学习”（never-ending language learning,NELL）项目

开放信息抽取从无结构网页中抽取的信息准确率还很低，主要原因在于网页形式多样，噪声信息较多，信息可信度较低。研究者需要在规模与质量之间寻找到一个最佳的平衡点。

多数据源的知识融合 进行知识图谱构建并非孤立地进行，需要实现多数据源的只是融合。谷歌发布的 Knowledge Vault 技术中，知识图谱的数据来源包括了文本、DOM Trees、HTML 表格、RDF 语义等多个来源。知识融合主要包括实体融合、关系融合和实例融合三类。

3 立项依据

随着电子计算机的发展与普及，个人计算机中的用户文件种类以及数量都已经大大增加。用户在使用个人电脑进行工作、娱乐等活动时产生了大量的用户文件，其数量已经超过了我们能管理的范围。我们在自己的电脑上通过“ls -lR | grep “ ^ - ” | wc -l”命令进行统计之后，发现 Mac 电脑“/Downloads”文件夹内就有 103480 个文件。这说明我们平时在电脑内使用、存储的文件之多，仅下载的文件的数量就超乎了我们的想象。面对如此数量庞大的文件，如何高效地管理它们，已经成为许多用户面临的一道难题。遗憾的是，到目前为止，手工处理文件分类仍然是许多用户的唯一选择。但是手工对文件分类的弊端也很明显，文件数量、种类越多，用户手工分类时，就越复杂繁琐，如果用户因为嫌麻烦而不去管理自己电脑内数量巨大的个人文件，就会造成，在需要使用某些文件时，“寻而不得”的困扰。需要注意到的是，造成这种困扰的根本原因是传统的文件系统采用的是垂直管理——即一个目录（文件）下有数个子目录或文件，这些子目录或文件中又包含了许多子目录和文件。这种文件系统的管理方式适用于早期个人电脑中文件数量较少的时代，而且用计算机很容易实现，因此被保留了下来，而如今，我们迎来了大数据时代，面对越来越多的数据需求和越来越大的硬盘空间，靠人的记忆力来从一个一个的文件、目录中精确地找到我们想要的文件显然已经不太可能，而且传统的文件系统的管理方式也无法适应越来越多的数据，这是因为：

- 文件层次过多，导致绝对路径特别长。例如这是我电脑里的一个常用文件夹的绝对路径：“/Users/Flint/Desktop/EasyAiMegumi/2018spring/OSH”

- 用户为了维护文件内容的一致性需要花大量精力处理许多复杂的信息。现代许多个人企业需要专门的秘书来处理、分类工作相关的文档。
- 不考虑文档里面词汇之间复杂的语义关系，无法处理短文本的稀疏问题。
- 单个文件只能存储于单一路径。例如，陈奕迅与周杰伦合唱的某歌曲存储于“/周杰伦”目录下，但是有另一个目录“/周杰伦”，如果将文件拷贝一份分别存储到两个目录下，又会占用很多的空间。最好的解决方式是提供一种方式使用户能通过“周杰伦”、“陈奕迅”两个关键词都可以在文件系统中检索到该歌曲。

为了方便用户使用文件，许多的操作系统提供了很多文件搜索的应用程序（例如：Apple's spotlight, Linux's KDE Baloo, Windows Desktop Search），这些基于内容或文件名的搜索应用程序可以对普通用户有所帮助，但它们是建立在特定的操作系统基础之上，缺乏兼容性。而且用户对其搜索结果并不满意，因为它们的搜索结果是基于字符串匹配而非被搜索对象本身的属性，即不符合语义搜索的要求。并且，这些搜索软件不提供对用户的文件数据进行分类的服务。

基于上述层次文件分类的不足之处，并受到搜索引擎（例如：google）利用知识图谱为用户提供更好的搜索服务的启发，小组打算开发一种可视化的语义文件系统。该文件系统基于知识图谱的相关知识对计算机的个人文件数据进行表示。这项工作主要的目的在于弥补传统文件系统无法表示的复杂语义关系的缺陷。即使用基于知识的文档表示方案，使得一篇文章不再只是由一组代表词汇的字符串表示，而是由文章的实体及其复杂语义关系表示。

4 重要性与前瞻性分析

4.1 重要性

随着时代的发展，文件系统中所存储的文件数量与内容也在大量的增加，尤其是在企业、学校等非个人电脑中，文件数量都是十分庞大的。在这样数量巨大的文件系统中，对于文件的分类显得至关重要，一个成熟的、人性化的文件分类系统可以有效地提高文件检索效率，这就是图文件系统的着力之处。

对于个人来说，记忆力是有限的，当其所使用的文件系统中的文件数量多到一定程度时，往往很难记住每一个文件的储存位置。对于目前所使用的层次文件系统，如果要找到一个忘记准确位置的文件，往往都要根据印象中该文件的大致位置以及所存放目录的名称内容来进行手动遍历操作，以此来找到该文件，该方法是目前绝大部分 PC 用户遇到上述问题时的首选解决方案，由操作过程可以看出该方法的效率极其低下；除了手动遍历之外，就是通过使用搜索功能，在所记忆的最精确的目录内进行搜索，由系统进行该目录内所有内容的遍历，以此找到该文件，如果对于文件名称难以记忆完整，虽然可以通过正则表达式进行搜索，但是这样又降低了搜索精度。对于以上两种方法，如果对于记忆目录有较大偏差，则几乎无法找到所需文件。

对于图文件系统来说，它会对放入其中的每一个文件进行关键词提取，并利用所有文件的关键词构建知识图谱，得到知识图谱后即可根据文件内容关键词进行检索，并且可以设置对同一关键词的文件进行预加载，可以提高文件读取速度。除此之外，可以通过知识图谱来构建可视化的关键词网络，通过关键词网络，用户可以高效地选取相关内容文件。举个例子，如果一个同学想要在自己的电脑中查找一个操作系统课上讲进程管理的 PPT ，但是他忘了该文件的存放路径和文件名称（因为课件很有可能是像 ch01.ppt 这样的命名），在图文件系统中，他只需要搜索“操作系统”,“进程管理”和“PPT”即可找到具有相关内容的文件，除此之外，还可以通过可视化文件网络看到相关文件，比如“操作系统”分类下的讲“内存管理”, “文件系统”等等的课件，也可以看到“计算机科学”分类下的“计算机组成原理”, “网络安全”等分类，由此可以方便地找到相关内容的文件。而这样的操作在标准的层次文件系统中是完全做不到的。

对于企业和学校，服务器中的文件数量要远比个人电脑中的数量庞大，单纯利用标准的层次文件系统进行存储与检索，需要记忆的文件目录也会十分繁杂，而因为企业、学校的服务器中的文件内容更为专业详细，通过建立知识图谱来构建图文件系统可以更加高效地提供文件的检索功能，对于文件检索与文件预加载效率都会有非常大的提升。

4.2 前瞻性

图文件系统具有广阔的发展空间，不仅仅是个人电脑或是独立服务器可以使用图文件系统，对于应用范围越来越广的分布式文件系统，其也可以与图文件系统进行结合。利用图文件系统中的知识图谱，实现从分布式文件

系统中根据内容关键词检索文件，也可以实现已打开文件的相关文件预加载，进一步加快分布式文件系统的读取速度。

5 相关工作

5.1 GFS: a Graph-based File System Enhanced with Semantic Features

在这篇文章中，重点阐述了 GFS(Graph-based file system)，这是一个含有语义特点的传统层次文件系统。在不改变系统的文件夹的前提下，GFS 可以将语义空间嵌套在传统文件系统的层次目录中。语义空间中，用户可以自定义文件标识符和浏览的层次来指导文件搜索。

如何管理并高效地对自己的文件进行分类已经成为很多用户和系统管理者的难题。到目前为止，手工处理文件分类仍然是许多用户的唯一选择。随着文件越来越多，文件的误分类也会产生严重的问题。而目前许多操作系统采用的是文件系统的层次管理 (hierarchical organization of file systems)，即由一个基本路径（层次结构根）和一个描述内部文档的一组标签的子目录组成。很明显，这是存在缺陷的，因为用户必须处理复杂的信息，才能保持内部的一致性分类，并能找到文件。这些问题由于文件系统的限制而变得复杂。文件系统的层次结构强制用户嵌套这些类别。况且对于类别中的优先顺序，没有严格的规则去规定，这可能导致不一致层次结构出现。

当用户找不到文件时，可以求助于桌面搜索应用程序，一般来说，操作系统开发人员会给它提供专有接口去高效地搜索文件。然而它只能基于文件名搜索文件，因为启用基于内容的搜索需要的开销太大，根本不现实。因此，一旦用户连文件名都记不住（事实上这很有可能），那么这样的方式就遇到了困难。

尽管有很多缺点，传统的文件系统仍然有很大的优势。1、它们是操作系统自带的不需要额外的代价去安装。2、使操作系统更容易将 API 与应用程序相对接，便于控制文件系统。3、用户可以通过相同的标准接口与文件系统进行交互。

语义文件系统（SFS）的目标是用基于关联的方式替换基于位置的方式。SFS 开发了一种在 API 保持不变的情况下，仅改变系统调用的行为来提供对文件的关联访问。后一种策略的优点是标准应用程序能直接继承新的关联功能。当然，语义文件系统失去了原来的基于位置的文件访问方式。

综上所述，作者提出了一个想法：将标准的文件目录结构 (standard directory) 与语义的文件目录结构 (semantic directory) 共存在一个树形结构中，并且用户可以选择使用哪一个文件目录结构。这样做的目的是在充分利用语义目录结构的优点的同时，不会牺牲传统层次目录的优点。每个语义目录都是独立的语义文件系统，并且它们有自己的命名空间和一组标签。最后，这一方法可以扩展成新的目录类型，只需扩展目录类型的分类标准，并添加新的语义目录类型到 API 即可。

GFS 是一个可以在任何地方都能建立两个目录类型的文件系统。标准目录结构有自己独立的命名空间 (namespace)，并且几乎与传统文件系统上的表现相同。而语义目录结构也有自己的命名空间。语义目录结构的根目录称为入口点 (entry point)。

语义目录空间可以被看成直接连接的自我图，其入口点是中心，标签是其他节点 (Node)。一旦创建了一个新的标签，就会创建出入点的两条边。此外，新标签与入口点路径中的所有标签都是双重链接。一旦用户在语义目录内创建一个标准文件目录结构就会导致语义空间退出，这便是标准层次结构与语义空间交换的方式。入口点和标签通过目录管理的标准系统调用进行操作。mkdir () 用于初始化新的语义空间，添加新节点或添加新边。unlink () 用于删除节点，边或整个语义空间。GFS 使用两种方法来标记文件和目录：链接和复制。链接是通过 link () 系统调用完成的。为了加速标记，GFS 分配目标路径的所有标记。作为进一步的简化，同一个标签的多个赋值不会引发错误，但会被默默忽略。复制需要注意，这个操作不是原子操作，而是一系列系统调用的结果。

语义目录的大小会随着它的使用而迅速增加。特别是在入口点列出语义空间的所有文件，目录和标签，这会造成两个问题。

- 需要注意的是，语义空间很可能包含周期，即目录可以生成无限长路径来达到相同的文件。如果该问题处理得不好，那么语义文件目录的可视化可能会导致非常差的用户体验。GFS 使用一种非常简单而有效的方法来防止上述问题的产生。可视化语义目录 (visualizing semantic directory)，即使相应节点之间的存在链接，也不会显示路径中已有的标记。具体来说，将多个相同标记实例的路径标记为无效的，试图访问它们时，系统会返回错误。

- 由于文件数量过多，将所有文件的标签让用户进行全面管理是不现实的。如果不采用专门的机制来简化此任务，手动标记是不切实际，并

且会使语义功能无法使用。GFS 设计了几个快捷方式（递归拷贝、多重标签）帮助用户快速构建或复制他们的分类标准并实现文件的快速多标记。

5.2 基于知识的图文档建模 (Knowledge-based Graph Document Modeling)

在这篇文中，作者提出了一个基于图的语义文件内容的模型。定义每个文件为实体 (entity)，实体里面包含了许多基于文件内容的基本属性。它们使用语义网络，即从 DBpedia 知识库获取关于实体及其实体的细粒度信息语义关系，从而产生一个知识丰富的文档模型。通过计算文档之间语义模型的相似性，结合 DBpedia 的结构与信息论的措施概念关联的基础上，明确的它们的语义关系，并使用 Graph Edit Distance (GED) 来计算语义相似度，再使用匈牙利算法找到最佳匹配。

尽管最近的研究趋势表明语义信息和知识丰富的方法可以有效地用于高端 Information Retrieval(IR) 和 Natural Language Processing(NLP) 任务，但为了有效利用这些丰富的模型并进一步推进这些领域。实际上，大多数利用文档表示的方法仅仅依靠形态句法信息，尽管已经提出了更复杂的模型，包括概念和基础矢量空间，但这些模型仍然没有利用广泛覆盖知识库（如 YAGO 或 DBpedia）中编码的关系知识和网络结构。

为此，作者提出了一个基于图形的文档模型，并提出了一种方法来生成文本的结构化表示，将消除歧义的实体与细粒度的语义关系相结合; 提出了各种信息理论措施来加权本体内不同的语义关系，并自动量化它们与尊重他们所连接的概念。因此，语义图中的边缘被加权以捕获概念之间的关联程度以及它们不同的特异性水平;

作者基于 GED 开发了一种新的测量方法，以便使用语义图计算文档相似度。他们的方法将计算本体内的语义距离视为概念匹配问题，并且使用匈牙利方法来解决这个组合优化问题。

5.3 Graph-based Text Representation and Knowledge Discovery

在这篇文章中，讲述了 Graph-based Text Representation 的情况发展以及作者提出的一种图形文本表示，建立在传统的向量空间模型（VSM）基

础上进行的变化与改进，以在文档的自动化搜寻检索以及关键词索引、关联中提供大量帮助。

如今，越来越多的文本文档存储在我们的电脑内，这可能使得非常庞大数量的有价值的信息被淹没在这些文档里面。这促使人们寻求一种自动化的方法，即在不用阅读全部内容的情况下来发现相关信息。最广泛使用的方法是通过文档的加权关键字表示。在 VSM 中，每一个文档或是一次查询都被表示为一个 t 维空间中的向量，并且有一组被称为"索引术语"的有代表性的关键字来描述。基于索引术语的检索很简单，但这同时也提出了关于信息检索任务的关键性问题。例如，用户使用索引术语的检索的一个根本基础的想法是，文件的语义和用户信息的需求可以自然地用一系列索引术语表达。显然，这是一个想得过于简单的问题，需要更多的思考，因为一个文档的大量的原本语义和用户需求都在我们用一系列关键字的同时丢失了。

此外，对于特定的文本挖掘任务，VSM 也展示了它的局限性。文本挖掘着重于发现新的知识，例如隐藏在巨大的采集样本之中的趋势和模式。与此相关的一个有意义的应用，是用户对于发现两个或多个概念之间的关联很有兴趣。概念和关联虽然可以被处理成术语并用索引代表，但概念之间的相关性本身却丢失了。

此文章提出了一种图形文本表示法，尝试更有效地捕获结构和文章语义。图形模型着重于检测文章中概念之间的联系，进而完成文本挖掘任务。

作者由 VSM 所受的限制而得到启发，我们提出了一种简单但更合理的图解表示法，它能够捕获文档中的（i）术语顺序（ii）术语频率（iii）术语共现（iv）术语上下文。

5.4 TagFS —Tag Semantics for Hierarchical File Systems

本文首先总结了传统的层次文件系统（hierarchical file system）无法帮助人们组织文件系统的原因：

- 单一位置。一个文件只能存储于严格目录层次中的一个位置。
- 没有查询优化。文件系统在浏览文件时不提供动态优化策略：每个目录只包含已经明确放在那里的目录。没有列出了其他可能相关的目录来帮助搜索。
- 没有导航援助。目录中的所有目录看起来都是一样的，没有任何指示子文件夹的内容，不管它们是否几乎是空的还是包含较大的子层次。

为了解决这种问题，本文提出了给文件加“标签（tag）”的办法，用户可以自由地用标签来给文件增加注释，这些标签可以与传统的文件系统共存，但是当我们检索文件时，标签将会起到很大的作用，通过标签基于语义，检索、复制、修改文件。

参考文献

[1] David Gifford, Pierre Jouvelot, Mark Sheldon, James O'Toole: *Semantic File Systems. ACM Operating Systems Review* Oct. 1991, pp 16-25.

[2] Daniele Di Sarli, Filippo Geraci: *GFS: a Graph-based File System Enhanced with Semantic Features*

[3] Michael Schuhmacher, Simone Paolo Ponzetto: *Knowledge-based Graph Document Modeling*

[4] Wei Jin, Rohini K. Srihari: *Graph-based Text Representation and Knowledge Discovery*

[5] Stephan Bloehdorn, Max Völkel: *TagFS — Tag Semantics for Hierarchical File Systems*

[6] Adrian Sampson *Design for a Tag-Structured Filesystem*