内容感知存储系统中的两阶段检索策略

刘 科 秦磊华 周敬利 聂雪军 曾 东
(华中科技大学计算机科学与技术学院 武汉 430074)

摘 要 随着存储系统规模的不断扩大,如何有效组织、管理和查询存储系统中的资源,成为了研究者必须应对的一个问题。目前存储系统中的查询需求主要来自系统管理员对元数据的查询以及普通用户对关键字内容的查询等两个方面。而内容感知存储系统自身所具备的重复数据删除和块相似性检测能力并没有被用于优化上述查询过程。为了充分利用存储系统感知到的上层语义和底层重复数据块信息,为使用者提供高效、便捷的查询服务,提出了内容感知网络存储系统中的两阶段检索策略。该策略将上层基于元数据和关键字的查询与底层存储系统的块相似性查询相结合,利用两次查询相关度的加权平均值作为相似度评价指标。最终的实验结果表明了该策略在降低失效性、提高查全率等方面的有效性。

关键词 元数据,数据迁移,内容寻址存储,两阶段检索,内容感知
中图法分类号 TP302 **文献标识码** A

Two-phrase Retrieval Strategy in Content Aware Network Storage System

LIU Ke QIN Lei-hua ZHOU Jing-li NIE Xue-jun ZENG Dong
(Department of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China)

Abstract As the storage capacity approach Exabytes, how to efficiently organize, find and manage data is becoming increasingly difficult for us. The query requests in storage system are coming from two aspects, the first one is metadata retrieval delivered by administrator and the second one is user's common keyword query. But the functions of de-duplication and block similarity detection in content aware storage system are not utilized to enhance the above query processing. In order to take advantage of the upper semantic information and the lower storage system's duplicate block information to deliver efficient query service for users, a two-phrase retrieval strategy was introduced. It combined metadata/keyword query with block similarity query and utilized ranking coefficient to evaluate similarity among query results. The experiments indicate that the retrieval strategy has efficiently enhanced the retrieval recall.

Keywords Metadata, Data migration, Content addressable storage, Two-phrase retrieval, Content aware

随着存储系统规模的不断扩大,要在海量数据集中找到所需的信息变得越来越困难。现有的查询需求来自两个方面,一方面是系统管理员为了制定存储策略,对元数据信息的查询;另一方面是普通用户通过输入关键字查询文本内容信息。前者通过查询数据的存放时间、所有者、访问权限等元数据信息,按照信息生命周期管理 ILM(Information Lifecycle Management)的要求制定分级存储策略,实现数据在存储设备间的迁移。后者通过对文件格式的解析,从文件中抽取出文本信息,并通过对关键字建立索引来实现基于文本内容的检索。上述两类查询操作由于所处理对象的差异,因此需要采用不同的索引结构和检索策略。数据库索引和倒排索引[1]是目前常用的两种索引结构。数据库索引通常被用于存储结构化的元数据信息,倒排索引被用于存储非结构化的文本信息。然而数据库没有针对存储系统的局部性特点做出优化,并不适用于大规模的元数据检索。而倒排索引也会消耗大量的存储空间。据统计,全文倒排索引的大小占被索引数据的25%~30%,其中95%的关键字很少或者从来不会被查询到[2]。因此,如何在存储系统中建立高效的索引成为了目前的研究热点。例如 SmartStore[3]采用分布式 R 树来索引元数据,并通过潜在语义索引聚类元数据;加州存储系统研究中心的 Ethan Miller[4]利用倒排索引实现海量文件系统中的全文检索。然而上述系统只考虑到基于元数据或关键字等上层信息的查询,却没有考虑如何利用底层存储系统的特点来优化索引结构,提高数据间的相关度计算效率,实现基于内容感知的相似性查询。

另一方面,在对数量庞大的数字内容进行高效存储与检索的研究过程中,归档存储和基于内容的智能存储引起了学术界和企业界的广泛关注[5]。在归档存储研究方面,IBM 海法实验室的 Preservation DataStores[6]通过将数据及复杂元数据对象整合到一起,来提升系统整体性能;在基于内容的存

到稿日期:2010-06-06 返修日期:2010-10-29 本文受国家自然科学基金(60673001),部委基金"基于服务定制的智能存储系统研究"资助。
刘 科(1979-),男,博士生,主要研究方向为内容感知存储系统及内容检索,E-mail:liuke@smail.hust.edu.cn;秦磊华(1968-),男,副教授,主要研究方向为高性能网络存储及网络仿真,E-mail:lhqin@mail.hust.edu.cn(通信作者);周敬利(1946-),女,教授,CCF 高级会员,主要研究方向为高性能网络及网络存储技术等;聂雪军(1979-),男,博士生,主要研究方向为内容感知存储系统;曾 东(1967-),男,副教授,主要研究方向为高性能网络存储。

储原型方面，加利福尼亚大学研究开发的 Deep Store 存储系统[7]采用了内容寻址存储 CAS(Content Addressable Storage)技术，使得内容成为识别数据的一个重要指标。因此，如何在复杂的存储环境下，通过内容信息智能地管理和检索各类变化的数据，成为当前存储领域的另一个研究热点。本文对数据的内容信息进行了扩充，以元数据、关键字和数据块的标识符信息作为计算相关度和实现查询的依据。

据此，本文提出了一种将上层语义查询与底层数据块查询相结合的两阶段检索策略。该策略因为有效地利用了存储系统所感知到的各类信息，所以能与存储系统很好地融合，实现了内容信息驱动的数据分级存储和查询，提升了系统的整体性能。本文第 1 节介绍内容感知网络存储系统的基本架构；第 2 节介绍内容信息的分类、生成及其编码方式；第 3 节引入两阶段检索实现机制并介绍该机制下的相关度计算和排序算法；第 4 节的实验结果表明，该策略在内容感知存储系统中取得了较好的检索效果。

1 内容感知网络存储架构

本文所采用的内容感知网络存储系统架构如图 1 所示，该架构主要针对 IP 存储网络而设计。系统总体架构划分为 3 个层次，分别为存储应用服务层 SASL(Storage Application Service Layer)、内容存储管理层 CSML(Content Storage & Management Layer)和存储子系统层 SSSL(Sub Storage System Layer)。存储应用服务层包括存储服务代理、内容感知服务代理和检索服务代理 3 个模块，它们对上层的存储应用提供一组标准的存储应用 API，对下层的内容管理控制器提供经过扩展的 SCSI 命令；内容管理控制器是总体架构中的核心层次，它主要实现的功能包括：解析标准/扩展 SCSI 命令、调用 XAM(Extensible Access Method)库及 VIM(Vendor Interface Module)存储子系统接口、内容索引的构建更新以及查询命令的解析和执行；存储子系统层提供了一个 SCSI 设备的抽象接口，该层由各种在物理性能、访问方式上存在差异的 iSCSI 存储设备构成，包括块存储设备、OSD 存储设备和支持 iSCSI 协议的 WORM 存储子系统。XAM 的引入屏蔽了各种 SCSI 设备的实现细节，将上层存储应用与底层存储设备隔离开来，使得存储应用在写入和读取信息时不依赖特定的存储系统和存储设备，极大地提高了各功能模块间的独立性。

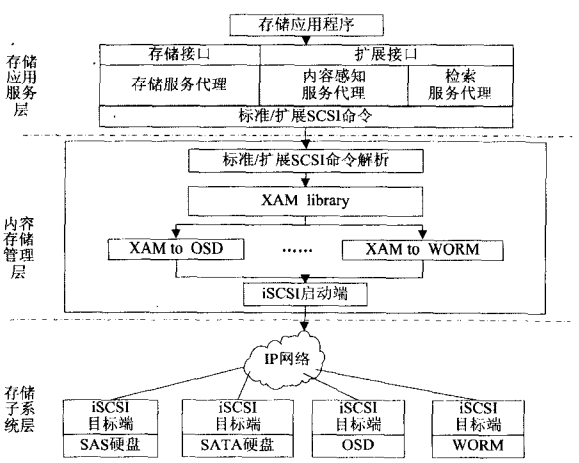

图 1 内容感知存储系统架构图

本系统依据存储网络工业协会 SNIA(Storage Networking Industry Association)提出的 XAM 接口规范来设计内容存储管理层，并通过 XAM 和 VIM 接口实现与底层存储系统之间的通讯。XAM 作为存储中间层，提供了一个逻辑上的数据容器的抽象。XAM 被选作中间抽象层的原因在于它支持将扩展元数据与数据绑定在一起的方式，可以确保在分级存储数据时元数据的一致性。同时它集成了存储管理功能，可以方便地依据信息生命周期和用户自定义的规则来管理数据。底层通过 VIM 与多种可切换的分级存储子系统相连，包括标准的文件系统、OSD 系统或者 WORM 存储子系统，以实现多种存储设备间的智能迁移和无缝融合。更重要的是，利用 XAM 所定义的查询接口，可以在存储系统中实现通用的检索功能，满足了两阶段查询的需求。XAM 所采用的查询语言是标准 SQL 的一个子集，它支持基于简单元数据的一级查询和基于流内容的二级查询。具体的检索流程在第 3 小节中详细描述。

内容感知存储系统架构是在传统存储系统基础上引入内容感知技术实现的，它使得整个存储系统能够根据数据的内容采取相应的存储和检索策略。由于这些功能的实现都是与内容相关的，因此本系统的关键在于如何感知和使用内容信息。

2 内容信息感知

2.1 内容信息分类

内容感知网络存储系统能够感知到的内容信息分为 3 类：第一类为属性元数据，由存储服务代理模块在用户提交存储请求时提取，文中使用的属性元数据主要包括文件的创建时间(CreateTime)、文件的所有者(Owner)、文件扩展名(FileExt)、文件的全局标识符(DATA_ID)和文件分块之后的块标识符(BLOCK_ID)等，BLOCK_ID 的具体结构见文献[10]；第二类为扩展元数据，由内容管理控制器根据信息生命周期管理的要求自动生成，主要用于存储、检索策略的制定。文中使用的扩展元数据包括：由当前时间与文件创建时间的差值得到的存储时间(StoreTime)、表明文件发生迁移后存储级别(磁盘级、光盘级或磁带级)的标志(StoreLevel)、反映文件物理存储位置的信息(Location)等；第三类为文件格式解析后，从各种文本文件中抽取的关键字信息，用于实现基于关键字的索引和查询操作。

2.2 内容信息生成与 XML 编码

上述内容信息中的某些部分(例如文件的创建时间)，在文件初始化时就可以获得，而有些信息例如文件的大小以及最近修改时间只有在文件处于稳定状态时才能获得。因此，内容信息的提取和更新分为以下 3 个阶段：第一阶段为新建文件阶段，此时文件过滤驱动程序截获系统调用，提取初始元数据，包括文件名以及产生文件的应用程序名等，生成初始的内容信息描述文件；第二阶段为数据块写入/更新阶段，在此过程中计算每个数据块的 Hash 值，并连同数据块的大小、偏移量等信息一同保存到描述文件中；第三阶段为关闭文件阶段，此时文件过滤驱动程序截获系统调用，根据文件格式解析并提取文件的内容信息，保存到内容描述文件中。

为了更好地理解文件内容，生成的内容描述文件必须具有结构化以及与应用无关的特点，XML 由于具有以上优点而成为本文所采用的编码格式。它将文件的非结构化字节流映

射为能够表示结构化信息的逻辑模型，为后续索引构建工作打下了基础。如图 2 所示，内容描述文件包含了在写入代理中生成的属性元数据、在内容管理控制器中生成的扩展元数据以及对文件格式解析后生成的文本内容信息。文本文件的 XML 编码过程如下：内容感知服务模块首先解析文件格式，然后从文件中提取出关键字信息，并在元数据 XML 文件中插入新标签 TEXT，最后将这些文字内容记录在该标签项中。

```
<?xmlversion="1.0"?>
<Metadata>
<Attribute_Metadata>
 <FileType> PDF </FileType>
 <Owner> liuke </Owner>
 <Data_ID>67534f2598f2045063a787796c426051 </Data_ID>
</Attribute_Metadata>
<Extended_Metadata>
 <StoreTime> 624 </StoreTime>
 <StoreLevel> 1 </StoreLevel>
  ..............
</Extended_Metadata>
  <TEXT>..... </TEXT>
</Metadata>
```

图 2　内容信息的 XML 编码形式

3　两阶段检索机制

要实现存储系统中的两阶段查询，必须满足两个基本条件，其一，通过有效的索引结构实现快速的元数据查询和关键字查询；其二，利用上层语义信息与底层数据块信息计算复合相关度。为实现上述要求，需要在存储系统各层中开展的工作包括：(1) 通过扩展 SCSI CDB 命令，支持元数据传输和基于内容的查询操作；(2)根据内容信息的不同类型，建立相应的索引结构，并制定对应的索引优化策略；(3)采用两阶段查询策略连接上层语义查询与底层块相似性查询；(4)通过新的相关度计算及排序算法，提高检索效率。

3.1　CDB 扩展

在 SASL 中定义了一组扩展 SCSI 命令，以承载在服务代理中生成的服务请求及参数。根据 SCSI 命令规范，选取可变长度的 CDB(Command Descriptor Block)格式来定义扩展命令。在可变长度 CDB 中，第 0 个字节为 OPERATION CODE，固定为 7FH，表示该命令是变长 CDB。第 7 字节为附加 CDB 长度，默认值为 172。第 8,9 字节定义的是 SERVICE ACTION，取值范围为 9000H-FFFFH，用于区分不同的服务。本文通过设定 SERVICE ACTION 的值，定义了如下几类与查询相关的命令：包括 SET METADATA(9001H)，GET METADATA(9002H)，METADATA SEARCH(9003H)，KEYWORD SEARCH(9004H)等，分别用于读写元数据信息，并实现基于元数据和基于关键字内容的检索。

3.2　索引生成及更新

为了提高查询效率，需要对内容信息进行分析并建立索引。我们对元数据信息和文本文件中的关键字信息分别建立了索引：对于属性元数据，由于其含有丰富的语义描述信息并具有结构化的特点，因此本系统将属性元数据从 XML 文件中抽取出来，采用 K-D 树的结构化形式建立索引[8]。选用 K-D 树是因为它是多维的二叉树，而且可以方便地实现点查询、最近邻查询等复杂查询操作。以文件名为例，索引的生成方式是采用 XML 查询语言[11]，找到 FileName 所对应的 tag，提取其中与文件名相关的信息，写入 K-D 树结构中，并构造形为〈FileName,DATA_ID〉的二元组，以更好地实现元数据与数据之间的关联。对于属性元数据中的 BLOCK_ID，由于其结构的特殊性，采用改进的倒排索引对其建立索引，具体结构见文献[10]；由于扩展元数据对用户透明，主要对存储策略的制定产生影响，因此不对其建立索引；由于文本内容不具有结构化的特点，并且关键字的数量要远远大于元数据类型个数，因此对这类信息不能采用结构化形式来构建索引，而是采用倒排索引的方式对分词后的关键字建立索引。

用户修改文件名称、调整文件内容或根据信息生命周期管理的要求迁移数据导致内容信息的更新，进而触发索引的同步更新机制。索引更新与优化分为两种情况：第一类为元数据索引的更新，在用户或操作系统修改元数据信息后，内容描述文件中的元数据属性发生变化，同时触发索引更新机制对发生变化的属性项重新建立索引。第二类为倒排索引的更新，通常采用 In-place，Re-Build 和 Merge-based 3 种倒排索引升级策略。Lester[9] 等人通过实验证明，在实际系统中绝大多数情况下基于重新合并的方法要优于其他两种索引更新方法。本系统采用 Merge-based Update 作为索引的升级策略，并按照访问频率对索引进行分割和分级存储，将不常用的索引存放到低速的存储设备上，以节约存储资源。

3.3　两阶段检索基本步骤

本文将内容信息的查询分为两个阶段来执行。在第一阶段查询过程中，用户通过查询界面提交文件名或关键字查询请求，得到与查询相匹配的 DATA_ID 集合；第二阶段查询对集合中的每个 DATA_ID 执行内容相似性查询，得到含有重复数据块的与之近似的结果；最后根据复合相关度计算公式对所有查询结果排序，并将查询结果返回给用户。两阶段检索具体实现步骤如图 3 所示，图中省略了与查询无关的部分模块。

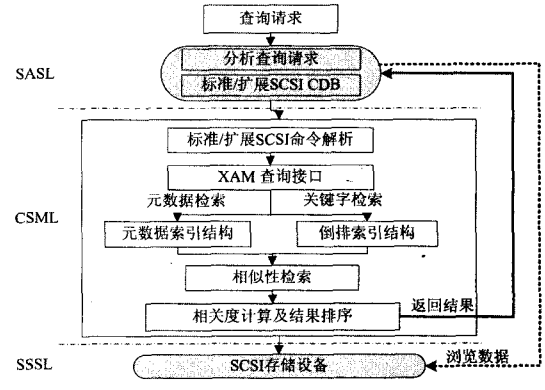

图 3　两阶段检索步骤

Step1　用户通过检索界面提交查询请求。

Step2　检索代理模块从用户的查询请求中解析出查询命令，经扩展 CDB 封装后，传送至内容管理控制器。

Step3　内容管理控制器解析查询命令，并判断查询类型。如果是元数据查询请求，转 Step4；否则通过 XAM 查询接口执行基于关键字的查询，在倒排索引结构中进行检索。如果命中，则记录与之相匹配的 DATA_ID；如果没有相匹配的内容，则记录为空，转 Step5。

Step4　通过 XAM 查询接口执行元数据检索操作，记录命中数据的 DATA_ID。如果没有相匹配的内容，则记录为空。

Step5　CSML 根据得到的 DATA_ID 执行相似性查询，

依据相关度计算公式对查询结果排序，然后 CSML 将命中的 DATA_ID 和数据的实际存储地址一同返回给 SASL。用户通过查询界面得到排序后的结果列表，并浏览数据。

Step6 SASL 将用户的浏览请求转化为标准的读命令，按照数据的物理存储地址，从存储设备上取得数据并返回给用户，本次查询操作结束。

上述两阶段检索策略的优势体现在，以 DATA_ID 作为文件的唯一全局标识符，通过〈DATA_ID，Location〉的二元组实现逻辑地址到物理地址的快速转换，可以将文件的检索过程与文件的物理存储位置独立开来，简化了检索流程。另外，该二元组的同步更新，使得返回给用户的存储位置是最近的有效值。但是当修改和迁移数据时，Location 和 DATA_ID 的值会发生变化，此时如果还按照原始的数值进行操作，就会访问到无效的位置和 ID，查询准确率也会显著下降。本文将上述两种失效分别称为位置失效和 ID 失效。

位置失效是数据迁移后的存放位置没有及时更新造成的，解决方法是实时更新〈DATA_ID，Location〉二元组。当失效发生时，SASL 不再通过 Location 直接访问存储设备，而是将文件的 DATA_ID 发送给内容管理控制器，找到最新的 Location 后，在更新二元组的同时将该位置对应的数据返回给用户进行浏览；ID 失效发生在用户修改文件内容时，由于内容感知存储系统支持 WORM 存储子系统，因此变化后的数据以增量方式存储并被赋予新的 DATA_ID 和 Location。此时，原始的 DATA_ID 所对应的数据不再是当前的最新内容，失效发生。ID 失效产生了一系列不同版本的文件，这些文件所对应的 DATA_ID 不同，但却具有内容上的相关性。因此采用两阶段检索策略中的相似性查询，可以得到最近修改过的所有相似文件，从而减少 ID 失效问题的发生，提高查询命中率。

3.4 相关度计算及排序

由于存储系统中存放的通常是非结构化的数据，因此计算存储系统中查询结果间的相关度是非常困难的。目前常用的相关度计算是通过命名空间的层次分割来实现的，即认为在同一子目录下的文件具有语义联系，并以此作为依据来计算查询的相关度。然而，文件系统的目录结构并不适用于所有的存储系统，在内容感知存储系统中，底层的存储子系统由 OSD 和块设备组成，不具备类似于文件系统的层次结构。为了计算这类子系统中的相关度，本文使用 Block-Ranking 算法[10]来衡量底层数据块之间的内容相似性，并针对两阶段检索策略的特点提出了一种新的相关度计算方法。此方法将相关度分为基于查询的相关度 $RANK_{query}$ 以及与查询无关的相关度 $RANK_{sim}$ 两个方面的内容，将两者加权平均后得到的值作为最终的相关度衡量指标，计算公式如下：

$$RANK_{final}=\alpha * RANK_{query}+(1-\alpha) * RANK_{sim} \quad (1)$$

式中，α 参数用于控制两种相关度所占比例；$RANK_{query}$ 和 $RANK_{sim}$ 分别对应于第一阶段和第二阶段查询中得到的相关度数值。当第一阶段查询为元数据查询时，采用 top-8 最近邻查询策略计算第一级的相关度。当第一阶段查询为关键字查询时，依据信息检索中常用的词频法来计算第一级相关度。与查询无关的二级相关度则采用 Block-Ranking 算法实现。完成上述两类相关度的计算后，将式(1)中得到的加权平均值作为最终的复合相关度，并按照其大小顺序对查询结果排序。

4 实验结果及性能分析

实验环境为 Intel Dual-Core 2.6G 双核处理器，2G 内存，操作系统为 Windows XP Professional，采用的数据集是信息检索中常用的 TREC(Text Retrieval Conference)[12] 数据集，我们选用的数据集中共包含 1803 个纯文本文件。检索模块的源代码在 FirteX[13] 和 Lucene[14] 的基础上结合本系统的特点做了部分改进。采用查询时间和查全率两个评价指标来衡量系统的整体查询性能。

4.1 查询响应时间

查询响应时间用于衡量两阶段检索策略对存储系统造成的影响。我们从 Trec 数据集中选择了 100，500，1000 和 1800 个文件来模拟不同的实验环境，并以 10 次实验得到的平均查询响应时间作为最终的实验结果。如图 4 所示，两阶段查询比传统查询增加了一次相似性查询操作，因此也带来了大约 4%的额外开销。但本系统所采用的索引分割策略减少了需要检索的索引空间，因此对整体性能影响并不明显。另外，在索引结构和查询特性方面，由于元数据的树状索引结构限制了查询的可扩展性，并且元数据的 Top-8 查询需要同时检索多个元数据属性以得到最终查询结果，而关键字检索中输入的关键字数目通常少于元数据检索。上述两方面的差异，导致了基于元数据的查询时间比基于关键字的查询时间长。

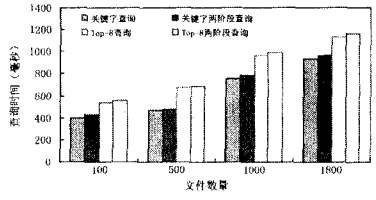

图 4 查询响应时间对比

4.2 查全率

查全率分为基于元数据检索(Top-k)的查全率以及基于关键字检索的查全率两大类。查全率的计算有如下约定：对于给定的查询 q，如果为元数据查询，假定 Top(q)表示理想的最近邻集合，Actual(q)表示本系统中得到的实际最近邻集合；如果为关键字查询，则用 Ref(q)表示满足查询要求的参考集合，将理想的关键字结果集合记为 Keyword(q)。依据上述约定，基于元数据的查全率和基于关键字的查全率分别定义为

$$Recall_{Top-k}=\frac{Top(q)\cap Actual(q)}{Top(q)} \quad (2)$$

$$Recall_{Keyword}=\frac{Keyword(q)\cap Ref(q)}{Keyword(q)} \quad (3)$$

图 5 反映了在采用两阶段查询策略前后基于元数据的 Top-8 查询和基于关键字的查询的效率对比情况。对于元数据查询，实验针对其中的文件名、创建时间、所有者等多个属性进行 Top-8 最近邻查询。而关键字查询则按照信息检索领域中的词频分布特点，对各类区分度较高的关键字进行检索。实验表明，由于两阶段查询考虑了数据块之间的相关性，增加了查询结果间的内容相似性，减小了失效问题发生的概率，使得最终的查全率有了明显的提升；而随着查询次数的增多，发生查询失效的概率也同时增加，因此查全率呈下降趋势。

(下转第 48 页)

networks [C]// Proc. of ACM Mobile Computing and Communications Review(MCCR). Canada, 2004: 50-65

[5] Bononi L, Di Felice M, Molinaro A, et al. Joint Channel Assignment and Multi-Path Routing for Multi-Radio Wireless Mesh Networks [C]// Proc. of 29th IEEE International Conference on Distributed Computing Systems Workshops. Montreal, June 2009: 476-481

[6] Kopparty S, Krishnamurthy S V, Faloutsos M, et al. Split TCP for mobile Ad hoc networks [C]// Proc. IEEE GLOBECOM. Taipei, Taiwan, Nov. 2002, 1: 138-142

[7] 程赓, 李昀照, 刘威, 等. 认知无线电网络路由及频谱分配联合策略研究[J]. 电子与信息学报, 2008, 30(3): 695-698

[8] Javadi F, Jamalipour A. Multi-Path Routing for a Cognitive Wireless Mesh Network[C]// Radio and Wireless Symposium. New Orleans, LA, USA, Jan. 2009: 232-235

[9] Brik V, Rozner E, Banarjee S, et al. DSAP: a protocol for coordinated spectrum access [C]// Proc. of IEEE DySPAN. USA, 2005: 611-614

[10] Cao L, Zheng H. Distributed spectrum allocation via local bargaining[C]// Proc. of the Second Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks. China, 2005: 475-486

[11] Zhao J, Zheng H, Yang G. Distributed coordination in dynamic spectrum allocation networks [C]// Proc. of IEEE DySPAN. USA, 2005: 259-268

[12] Gong Michelle X, Midkiff Scott F, Mao Shiwen. Design principles for distributed channel assignment in wireless Ad hoc networks[C]// Proc. of IEEE ICC. Seoul, 2005: 3401-3406

(上接第 23 页)

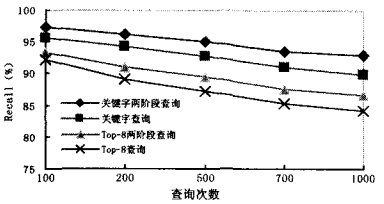

图 5 查全率对比

除了查询响应时间和查全率之外，我们还通过实验得到了 α 的最佳取值。依据式(1)，α 越小，表明二级查询所占比重越大，因而相似性查询得到的查询结果数目就越多，这些结果与给定查询请求相关的可能性就越大，所以带来查全率的提升；但是如果 α 太小，会导致一级查询对相关度的影响小于二级相似性查询，造成查询结果集合中相似性结果增多，影响了查询相关度的判断。表 1 反映了 α 的取值对查全率的影响。实验结果表明，当 α 小于 0.75 或者大于 0.9 时，两阶段查询的查全率显著下降。本文选择 0.9 作为最优的 α 值，用于计算加权相关度。前面的实验结果均在 $\alpha=0.9$ 的实验条件下得出。

表 1 α 参数对两阶段查全率的影响

α 参数大小	Top-8 两阶段查询	关键字两阶段查询
0.50	78.4%	82.3%
0.75	88.1%	90.5%
0.90	93.6%	96.4%
0.95	90.7%	92.2%

结束语 本文在描述内容感知存储系统整体架构的基础上，针对存储系统自身的特点，提出了两阶段检索策略。本策略将上层语义检索与底层块相似性检索结合起来，通过两者相关度的加权平均来对最终的查询结果排序。由于同时考虑了各类相关性，查询的结果集合可以最大程度地满足查全率的要求。最终的实验结果表明，在带来少量查询开销的前提下，本策略减少了各类失效问题的发生，使查全率有了较大提升。

参 考 文 献

[1] Blanco R, Barreiro A. Probabilistic static pruning of inverted files[J]. ACM Transactions on Information Systems, 2010, 28 (1): 1-33

[2] Mitra S, Winslett M, Hsu W W. Query-based partitioning of documents and indexes for information lifecycle management [C]// Proceedings of ACM SIGMOD on Management of Data. 2008: 623-636

[3] Hua Yu, Jiang Hong, Zhu Yifeng, et al. SmartStore: a new metadata organization paradigm with semantic-awareness for next-generation file systems [C]// Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis. 2009: 1-12

[4] Leung A, Miller E L. Scalable Full-text Search for Petascale File Systems[C]// Proceedings of the 2008 Petascale Data Storage Workshop. 2008

[5] Raghuveer A, Jindal M, et al. Towards efficient search on unstructured data: an intelligent-storage approach [C]// Proceedings of the Sixteenth ACM Conference on CIKM. 2007: 951-954

[6] Factor M, Dalit Naor, Simona R, et al. Preservation DataStores: new storage paradigm for preservation environments[C]// Proceedings of IEEE Conference on MSST. 2008: 3-15

[7] You L L, Pollack K T, Long D D E. Deep Store: An archival storage system architecture[C]// 21st International Conference on Data Engineering. 2005: 804-815

[8] Leung A W, Shao Minglong, Bisson T, et al. Spyglass: fast, scalable metadata search for large-scale storage systems[C]// Proceedings of the 7th conference on FAST. 2009: 153-166

[9] Lester N, Zobel J, Williams H E. In-place versus re-build versus re-merge: Index maintenance strategies for text retrieval systems [C]// Proc. of the Australasian Computer Science Conference. 2004: 15-22

[10] Zhou Jingli, Liu Ke, Qin Leihua, et al. Block-Ranking: Content similarity retrieval based on data partition in network storage environment [J]. International Journal of Digital Content Technology and its Applications, 2010, 4(3): 85-94

[11] Saito T L, Morishita S. Relational-style XML query [C]// Proceedings of ACM SIGMOD on Management of data. 2008: 303-314

[12] Trec[EB/OL]. http://trec.nist.gov. 2009

[13] 郭瑞杰, 程学旗, 许洪波, 等. FirteX—高性能全文索引和检索平台[C]// 内容计算的研究与应用前沿, 第九届全国计算语言学学术会议论文集. 2007

[14] Lucene[EB/OL]. http://lucene.apache.org, 2009