基于 Tree-lib 的大数据实时分析研究

沈来信[1,2] 王 伟[2]
(同济大学嵌入式系统与服务计算教育部重点实验室 上海 200092)[1]
(黄山学院信息工程学院 黄山 245041)[2]

摘 要 为提高大数据的存储和并行处理能力,建立了以列存储 Infobright 与分布式 MySQL Cluster 为核心的大数据实时并发分析、管理模式,以完成对开源 brighthouse 引擎的二次开发。利用管理程序 Tree-lib 对分布式大数据进行可视化监控、维护和管理。实验结果表明,Infobright 和 Cluster 组合具备对大数据的高压缩存储、多并发查询和高效实时分析的能力,Tree-lib 完成对树和库的生成、检测、更新、备份和灾难恢复等,实现可视化双向管理和维护的目的。

关键词 大数据,Infobright,MySQL Cluster,brighthouse 引擎,Tree-lib,灾难恢复
中图法分类号 TP392 **文献标识码** A

Real Time Analytics Study of Big Data Based on Tree-lib

SHEN Lai-xin[1,2] WANG Wei[2]
(Key Laboratory of Embedded System and Service Computing, Ministry of Education, Tongji University, Shanghai 200092, China)[1]
(School of Information & Engineering, Huangshan University, Huangshan 245041, China)[2]

Abstract In order to improve the storage and parallel processing capabilities of big data, the big data of real time concurrent analytics and management mode were built with the center of column stored Infobright and distributed MySQL Cluster, to complete to secondary develop of the open source brighthouse engine. The managing procedure Tree-lib was used to visual monitor, maintain and manage the distributed big data. The results of experiments show the combination of Infobright and Cluster have the capability of high compression storage and multiple concurrent inquiries and the efficient real time analysis with big data. The Tree-lib accomplishes generation, detection, update, backup and disaster recover of tree and library. Finally, the purpose of visual bidirectional management and maintenance is achieved.

Keywords Big data, Infobright, MySQL Cluster, Brighthouse engine, Tree-lib, Disaster recover

1 引言

为建立古民居、古村落无损特征量化模型,我们建立无线传感器 WSN(Wireless Sense Networks)对其特征参数(温度、湿度、颗粒、气体、噪声、建筑等)进行测量、存储、分析和应用,此时涉及到的古民居数据应用分类众多,累计字段达 1 万之多,再加上实时采集的数据中,有的数据是关系数据,有的数据是无关系数据,同时存储将达到 TB 级,普通的关系型数据库已经不能满足这样的大数据存储,必须借助数据集市(仓库)。Infobright 是新兴的分析型数据仓库实现技术[1],它采用的列存储技术能够得到高压缩比(达到 10∶1 甚至 40∶1)的存储,同时,它采用的知识网格技术又使得大数据查询高效响应。但是它的缺点是不支持高并发访问(目前开源版本仅支持 10 个用户),不能满足多用户对数据实时查询访问、分析的实际需求。开源分布式集群 MySQL Cluster(以下简称 Cluster)具备并行处理能力,相应的服务器配置不高,是一个基于内存的存储引擎,存储能力受制于内存的大小(依赖于 Cluster 中最小内存的机器),Infobright 结合 Cluster 的架构具备并行处理、大数据存储和分析能力,是大数据存储、高效处理和实时分析的有效方案。

同时,为实现大数据在 Infobright+Cluster 架构下处理和分析的可视化,我们对开源 brighthouse 引擎进行二次开发,使用树技术,设计自动维护管理程序 Tree-lib,完成对树的结构建立、剪枝、统计、更新、备份、恢复和对库(Infobright 和 Cluster)的创建、字段生成、更新、备份和恢复等,提供可视化监控、管理和维护方案等。

2 研究现状和存在问题

Philip Russom 给出了大数据分析的基本介绍、发展阶段、当前的最佳实践、使用的工具、技术和发展趋势等,其主要还是采用传统的数据库存储或分析技术,如 SAS,SPSS 等[2],不能有效解决大数据存储时查询的快速响应和实时并发性。Randal E. Bryant 等介绍了大数据实现技术的挑战:高速网络环境、集群计算编程、机器学习和数据分析技术等[3]。

到稿日期:2012-09-12 返修日期:2012-12-20 本文受国家自然基金项目(61103068,61174158),安徽省优秀青年人才基金项目(2012SQRL183)资助。

沈来信(1979—),男,硕士,讲师,主要研究方向为算法、数据挖掘、数据仓库,E-mail:slx965@yahoo.com.cn;王 伟(1979—),男,博士,讲师,主要研究方向为信任管理、机器学习、信息检索。

Herodotos Herodotou 等提出一个对大数据分析的自调节系统 Starfish，其建立在分布式架构 Hadoop 上，能够适应用户需求和系统负载的高性能自动管理模式[4]。Jorge Bernardina 等使用分布式集群和分布式计算来提高在数据仓库和 OLAP 上的查询性能[5]。Bill Allcock 等提出一个数据网格工具，用以提高大分布式数据的效率、高性能、可靠性和安全性等[6]。Paolo Costa 等提出大数据实际应用中的实时性需求，设计了类似 MapReduce 的基于 Cluster 的企业集群 Camdoop[7]，来提高网络的聚合能力。王珊等分析了大数据分析的重要特性，并对当前主要实现平台，即并行数据库和 MapReduce 及两者混合架构进行了分析与对比[8]。张延松等提出了面向星形模型特点的以维表为中心的分布式存储模型，提高了 SPJ-GA-OLAP 操作集上的优化[9]。琳琳等研究了在 Map-Reduce 框架上的海量数据 Skyline 查询的优化问题，提出了一系列的 Skyline 查询算法[10]。覃雄派等对当前的大数据分析技术进行了比较，分析了 MapReduce 和并行数据库的优缺点以及相互渗透的发展和扩展等[11]。吴广君等设计了分布式海量结构化数据存储和检索系统 MDSS，采用列存储结构，使用 B+Tree 索引等来提高检索效率[12]。曾志勇等提出一个基于内存的并行化方案，亦即将原始数据集平均分配到各子计算节点中，实现 Apriori 算法的并行化[13]。关晓蔷等利用极大相容块作为决策支持度，并将其属性选择作为启发式信息，在不完备信息系统中完成决策树的自动生成，以能快速准确地得到规则集[14]。

以上研究大都是对大数据独立的海量存储或分析算法的研究，没有考虑到实时多并发性需求。在实际应用中，既要能达到对大数据高压缩比存储下的快速查询响应，又要能满足多用户的实时并发查询、分析与应用。

Infobright 的核心引擎 brighthouse 采用列存储模式可以实现高压缩比存储，采用的知识网格技术又使得查询高效响应，是高效率的分析级数据仓库实现方案。为解决 Infobright 引擎不提供多并发实时查询需求的问题，需要建立分布式集群 Cluster 环境。但是前者是基于硬盘的数据仓库技术，后者为基于内存的分布式数据集群，两者的数据不能直接完成交换，需要借助第三方软件或架构才能实现。本文重点研究 Infobright 与 Cluster 架构的数据交换并使用树和 Tree-lib 对框架进行可视化管理。

下面首先介绍 Infobright 和 Cluster 的关键技术以及两者数据交换方案的实现。

3 Infobright 和 Cluster 关键技术及数据交换方案

3.1 Infobright 关键技术

Infobright 是基于 MySQL 的硬盘存储级开源数据仓库实现的，具有核心模块 3 个，优化器保证了最小化的解压缩数据和有效提高执行计划，知识网格存储的是元数据、列信息、表关系、数据块分布状态统计信息和同等查询状态缓存信息等，数据块是数据经过压缩后的实际的存储位置，数据是按照数据存储块。Infobright 结构简图如图 1 所示。

这种层次结构既保证了数据的高压缩比，又能实现高速查询，知识网格架构是高性能的关键，缺点是并发数小，不能满足多用户的并发实时查询，Infobright 还不支持索引、不支持 DML(Insert、Update、Delete)语句，数据加载只能用 Load 方法等。

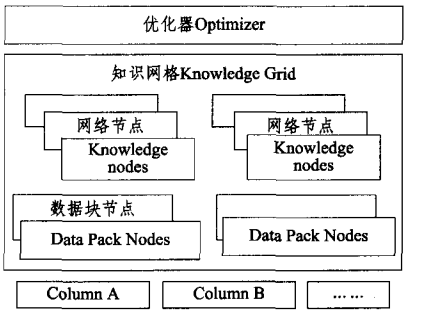

图 1 Infobright 结构简图

3.2 Cluster 关键技术

MySQL Cluster 是基于 MySQL 的内存存储级分布式数据库实现的，主要由 3 类服务器节点组成，SQL 节点主要提供客户端访问接口以及与数据节点交互，数据节点不仅与 SQL 节点进行数据读写，而且相互之间还实现数据同步，保证了部分节点崩溃后的数据恢复，管理节点负责 SQL 节点和数据节点的配置、启动、负载均衡、监控和维护等，SQL 节点可以与数据节点或管理节点部署在同一服务器中，如图 2 所示。

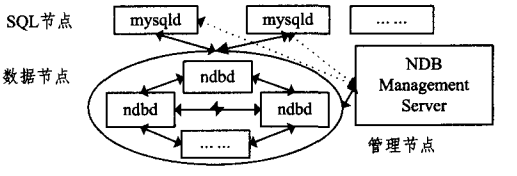

图 2 MySQL Cluster 集群结构简图

Cluster 具有很强的写操作和可扩展性、实时性和高吞吐量，缺点是数据库大小受制于节点机内存，不能适应大数据的查询与应用。

在实际 WSN 测量网络中，需要上百台甚至上千台的数据接受服务器，单一的 Infobright 是不能满足需要的，需要采用 Cluster 接受各传感器网络传来的数据，然后再不断地写入到 Infobright 中。同时，面对大量的并发用户实时查询访问，也需要用 Cluster 来响应用户的查询，此时则需要从 Infobright 中读出数据到 Cluster 中。

Infobright+Cluster 的有效组合是解决分布式网络数据库和大数据存储的有效途径。但是这里存在很多问题，如数据库的引擎和存储引擎不一致，如何进行协调，数据库数据与存储数据交换如何实现，如何进行管理等，特别是可视化、高效率的管理。

3.3 Infobright 和 Cluster 数据交换实现

Cluster 提供数据的查询与初步处理服务，而 Infobright 提供数据的存储和分析服务。当采集数据到来时，通过处理(清洗，排序)进入到 Cluster 中，供用户进行低级查询。在 Infobright 与 Cluster 之间还存在两个数据转移过程：第一个转移过程是采集的数据先写入到集群 Cluster 中，再把数据写入到 Infobright 中存储，用算法 1 来实现，第二个转移过程为将 Infobright 中的数据读出到 Cluster 集群中，供用户多并发请求，用算法 2 完成，以响应用户查询请求，Infobright 与 Cluster 的数据转移过程如图 3 所示。

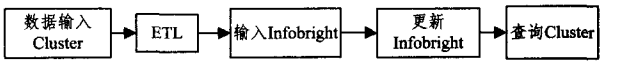

图 3 Infobright 与 Cluster 数据转移过程

由于 Infobright 不支持 Select 等更新语句，只支持 Load 方法，因此使用第三方文件(文本文件或 XML 文件)方式，如算法 1 所示。

算法 1 Cluster 数据存储到 Infobright

(1)把 Cluster 中未标记的记录不断用 Select into 写入到文本文件或 XML 文件中，并对此记录进行标记；

(2)把形成的文本或 XML 文件用 Load 方法载入到 Infobright 里，并标记此文件中相应行；

(3)删除 Cluster 中已经标记过的数据。

Cluster 数据存储到 Infobright 的方案如图 4 所示。

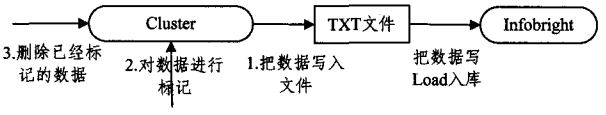

图 4 数据采集 Cluster 到 Infobright 入库方案

当响应多用户实时查询时，需要在 Cluster 和 Infobright 中同时进行搜索，当用户搜索的数据在 Infobright 中时，数据需要从 Infobright 读出到 Cluster 中，如算法 2 所示。

算法 2 Infobright 中数据读出到 Cluster 以响应用户

(1)当用户提出查询请求时，一个线程进入 Cluster 数据库进行查询，尽可能把 Cluster 里的满足要求的数据全部查询出来，反馈给用户；

(2)同时，另外一个线程进入到 Infobright 内，在 Infobright 里进行搜索，随着用户的查询条件越来越清晰、准确，最终找到用户所需的数据，并把这些数据转移到 Cluster 内，提供给用户查询。

基于 Infobright+Cluster 的数据实时查询分析方案如图 5 所示。

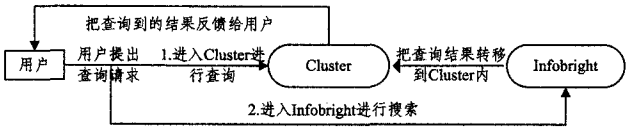

图 5 基于 Infobright+Cluster 的数据查询分析方案

基于 Infobright+Cluster 的架构完成了数据写入、查询和分析的结构建设，这里还存在一个很关键的问题，需要一个可视化的界面来完成对存储过程和查询过程的实时监控、维护和管理。

4 基于树和 Tree-lib 的分布式大数据管理模式

4.1 Infobright 结构和数据更新

当用户输入不相关数据时，普通的关系型数据库就不能满足需求了，此时 Infobright 的列存储模式就能够很好地完成数据的存储和压缩，Infobright 不支持索引，列是存储的最小单位。

Infobright 的列结构是不断变换的，它可以根据用户输入的内容，自动生成列，并完成内容存储与关键字 Keyword 标识，Infobright 结构与数据更新算法如算法 3 所示。

算法 3 Infobright 结构与数据更新算法

(1)使用字、词的统计信息作为权值，采用分类 C45 算法对输入内容进行分类，完成基于内容的列自动生成；

(2)形成相应的列存储，如果当前列存在，则把内容并入到已有的列中并且标识对应的关键字 Keyword，如果当前列不存在，则形成一个新列存储相应内容并且同时标识 Keyword。

(3)为实现高性能，brighthouse 引擎还不断地利用过滤算法与剪枝算法，利用一个加权阈值，对低于阈值的 Keyword 进行剪枝或过滤，实现引擎的更新；

(4)最后把内容存入相应列中，完成列存储，实现 brighthouse 引擎更新；

(5)完成库、表级结构和内容更新，更新 Infobright。

算法过程如图 6 所示。

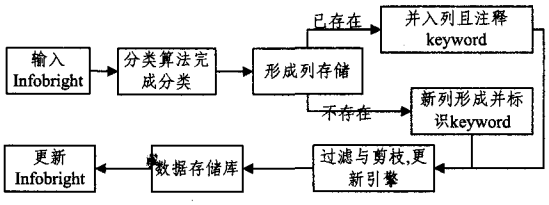

图 6 Infobright 结构与数据更新示意图

对于 Infobright 的数据更新，采用的是对其开源引擎 brighthouse 进行二次开发，这里设计了一个树和 Tree-lib 的架构来对这个过程进行可视化监控、管理和维护。

4.2 基于树和 Tree-lib 的 Infobright+Cluster 架构设计

我们采用树技术，设计一个 Tree-lib 结构，对 Infobright+Cluster架构进行可视化管理和维护。

这里需要设计两棵树，分别用于对应的 Infobright 和 Cluster 的可视化监控，实现树和库表的直接映射，树与 Infobright 和 Cluster 的映射关系如图 7 所示。

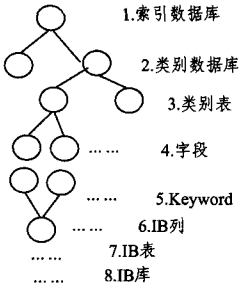

图 7 树-库映射关系图

基于树的 Infobright+Cluster 架构管理算法如下：

(1)第 1 棵树：前 4 层对应 Cluster 库，数据库名称组成为 4 级结构，对应前 4 层树，其中的顶层为索引数据库，所有数据库在这个节点属性中都有一个标识，第二层为类别数据库层，每个库对应一个节点，第三层是类别表层，每个表对应一个节点，第四层为字段层，Cluster 所有字段都对应一个节点，此时，形成一个 4 层胖树结构，以完成对 Cluster 中数据库、表、字段和数据的监控、管理和维护；

(2)第 2 棵树：倒置的树结构：第 5 层以下对应 Infobright(图中简称 IB)知识网格中的关键字(Keyword)，这里可以将其分成多层，Infobright 列字段自动生成算法如下：

①初始化：Infobright 中 Keyword(对应树中的第 5 层)初始化为 Cluster 第 4 层所有字段，即每个字段都是一个 Keyword，即为 Infobright 的一列，该层为树的叶子节点层，当数据库内容中出现当前列不能存储的词或词组时，Infobright 中就会产生新列，并标识为 Keyword，相应的在树中生成对应的叶节点，这是一个胖树结构，该层节点树在百万级甚至更多，一般一个 Keyword 能够管理约 1MB 数据；

②利用分类算法 C4.5 对树进行剪枝：利用用户查询的统计信息，对小于阈值的 Keyword 节点进行剪枝。

③根据用户访问 Keyword 的统计信息，进行树的节点融合，实现 Keyword 树的智能化自动管理。

同时，设计管理程序 Tree-lib 完成对树生成、更新，以及

数据和 Infobright+Cluster 库数据、结构的维护和管理，维护和管理过程如图 8 所示。

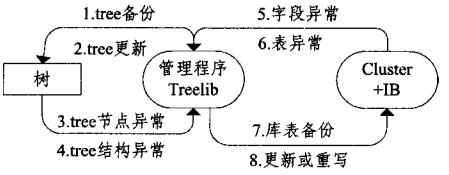

图 8　自动管理程序 Tree-lib 示意图

Tree-lib 算法设计如下：

(1)Tree-lib 是核心，首先根据库表结构生产树，然后定期完成对树的检测、监控、备份和更新；

(2)树中发生节点异常或结构异常时，通过 Tree-lib 和 Infobright+Cluster 库完成恢复或重建；

(3)Tree-lib 定期对 Infobright+Cluster 库或表进行检测、监控、备份或更新等；

(4)Infobright+Cluster 库中表字段发生异常、表或表名异常时，通过 Tree-lib 和树完成库、表或字段的灾难恢复或重建。

5　实验设计与结果分析

我们开始建立了有 10 个节点的 Cluster 体系，其中 2 个作为管理节点，4 个作为数据节点，4 个作为 SQL 节点，其中硬件环境为：WINDOWS2003+浪潮 PC 服务器 8 台+其它服务器 2 台，形成约 4GB 内存，完成节点机器的安装(SQL 节点、NDB 数据节点、管理节点)、配置(引擎、登录权限、IP；主机名、标识 ID；数据空间、索引空间、SQL 节点信息、NDB 节点信息等)和启动(首先启动管理节点服务器，然后启动存储节点服务器，最后才启动 SQL 节点服务器)。我们分别设计 4 个实验对 Cluster、Infobright 查询、Brighthouse 引擎与传统引擎和 Infobright+Cluster 进行测试。

实验 1　使用 500 台机器、每台机器 100 个进程对 Cluster 做并发访问测试，当前内存空间为 2GB，硬盘空间为 1T。分别设计小表数据为 2GB，大表数据为 1TB，测试结果如表 1 所列。

表 1　对 Cluster 进行插入查询测试表

操作	Insert	小表 Select *	大表 Select *	两表 Select *
时间	4.3s	16.18s	3h 25min 12.5s	5h 16min 25.4s

从表 1 可以看出，对大数据(TB 级)的查询与统计等的处理时间在万秒以上，这对基于 Cluster 的大表查询和嵌套查询在应用上是不实用的。

实验 2　对 Infobright 做查询分析测试，表数据平均为 1TB，对 Infobright 进行查询分析的测试结果如表 2 所列。

表 2　对 Infobright 查询分析测试表

操作	装载 load	单表 Select *	两表 Select *	嵌套 Select *
时间	3h 23min 45.8s	3min 27.5s	6min 46.2s	13min 25.6s

从表 2 数据对照看出，Infobright 中多表查询使用的时间对应 Cluster 中多表查询的使用时间有明显减少，存储空间越大，Infobright 查询分析优势就越强。

实验 3　对 brighthouse 引擎和传统引擎进行比较实验，对古民居建筑库中一个有 1 亿 3 千万数据行数的表进行分组聚合、where 限定查询、日期筛选与分组、报表归并与汇总查询分析，如表 3 所列。

表 3　brighthouse 引擎与传统引擎查询分析表

引擎	分组聚合	Where 限定	筛选分组	归并汇总
brighthouse	46.9s	5.42s	48.5s	25min 46.4s
MyISAM	12min 10.8s	49.27s	4min 45.8s	4h 39min 32.6s

从实验 3 可以看出，brighthouse 引擎在大数据的组合查询时效率明显比传统引擎 MyISAM 高，虽然 Infobright 支持多种引擎，但是各个引擎的效率还是明显不同的，其默认引擎 brighthouse 有着明显的优势。

实验 4　对 Infobright+Cluster 数据管理体系进行数据交换测试。分别对 Cluster 到 Infobright 的数据载入 Load 和 Infobright+Cluster 的数据查询进行测试，表大小为 1TB，当查询内容在 Cluster 中时，记为第一次查询，此时使用的查询时间为 1Select，当查询内容不在 Cluster 中，而需要到 Infobright 中查询时，记为第二次查询，此时使用的查询时间为 2Select，如表 4 所列。

表 4　Infobright+Cluster 数据交互测试表

操作	装载 Load	(1Select)	(2Select)
时间	1h 15min 26.5s	6min 12.5s	12min 20.3s

从以上实验可以看出，Infobright+Cluster 组合显著提高了大数据在 Cluster 中的响应时间，克服了 Infobright 实时并发能力弱的缺点。

结束语　当 Cluster 数据库的引擎与 Infobright 存储引擎不一致时，存在一个多引擎的相互协调问题，典型的有相同库间引擎协调、不同库间引擎协调、异构网络引擎的无缝高性能安全过渡等，后期研究的重点就是解决这些多种引擎的自动适应问题。

Infobright 的开源引擎 brighthouse 允许二次开发，在实际应用中，可以设计更合理的分类算法和字段自动生成机制，采用胖树结构去管理关键字，生成关键字树，采用适当的剪枝算法和过滤算法优化关键字树，提高 Infobright 的性能，实现 Infobright 引擎的可视化管理。

大数据的处理可以在数据仓库存储和分布式数据库访问基础上完成。基于 Tree-lib 构建的树，能够实现对 Infobright+Cluster 的可视化监控、管理和维护，同时可以实现树和库的双向备份和灾难恢复等。

参 考 文 献

[1] Slezak D. Brighthouse: An Analytic Data warehouse for Ad-Queries[C]//PVLDB '08. August 2008:1337-1344

[2] Russom P. Big Data Analytics[R]. Tdwi Best Practices Report. Fourth Quarter, 2011:15-21

[3] Bryant R E, Katz R H. Big-Data Computing: Creating revolutionary breakthroughs in commerce, science, and society (Version 8)[M]. Computing Community Consortium. 2008:1-7

[4] Herodotou H, Lim H, Luo Gang. Starfish: A Self-tuning System for Big Data Analytics[C]//5th Biennial Conference on Innovative Data Systems Research (CIDR'11). Asilomar, California, USA, 2011:261-272

[5] Bernardino J, Madeira H. Data Warehousing and OLAP: Improving Query Performance Using Distributed Computing[C]// 12th Conference on Advanced Information Systems Engineering. June 2000:1-12

(下转第 237 页)

高[14]。因此，总的来说，此种检索方式的检索能力是比较优秀的。

从图2中看到，传统检索方式的查全率的平均值只有72.3%，与模糊查询相比，其查全率要低于16%。而且anxiety这一类的情感和sad这一类的情感具有一定的类似性，所以在非量化判断某首乐曲到底归属于哪一类情感类型时，出现判断失误的概率较其他类型的情感也要高一些。同时，对这两类情感的查准率和查全率也会有影响。通过对比，发现模糊检索的检索能力比普通的非量化查询的查询能力要高很多。

从图3中可得，这两种检索方式的 F 值都在65%以上。对于模糊检索来说，每种情感的 F 值都在93%以上，显然比传统检索方式的 F 值高。尤其是“anxiety”这一类情感与“sad”一类情感具有较大的相似性，若按照传统的检索方式，就很容易将那些带有伤感又带有焦虑情绪的音乐直接归到“sad”这一类情感里，从而导致其查全率大幅度降低。不妨设想一下，如果是对于大量的音乐资源，按照传统的这种检索方式，其检索能力必然会大大下降，尤其是查全率。若是用模糊检索这种方式，其查全率也会保持良好的势头。

3.2.4 相关度

音乐资源与这些情感类型不只有相关和不相关这两种描述，同时还有其相关度的级别。对于返回结果来说，其相关度级别越高，排名越前。由于对相关度的描述还没有统一的函数模型，在此，将隶属度函数的值直接作为其相关度的级别值。所以对模糊检索来说，能够清楚地知道每个音乐资源与每种情感类型的相关度。如果相关度值为0，则说明两者完全不相关，反之亦然。但是，如果按照隶属度来计量相关度，则不能肯定地说，“happy”隶属度值为0.05的音乐资源与“happy”这一类情感相关。由于相不相关是由用户来判断的，因此，对于相关这一概念的值的界定，需要若干用户或专家来确定到底这个隶属度值在什么范围才能说明相关。然而，对于传统的检索方法，则不能看出音乐资源与情感类型的相关度。

3.2.5 覆盖率和出新率

从图4可得，模糊检索能更多地覆盖用户已经知晓的相关资源，尤其是对于那些情感界限很明显的音乐资源，如“happy”和“sad”这两类情感类型的音乐资源，它们的覆盖率都达到了100%。显然，对于中间类型的情感“anxiety”，模糊检索的覆盖率要比传统检索的覆盖率高出很多。这也说明模糊检索更能体现出音乐情感的多样性。

图5中模糊检索的 N 值曲线位于传统检索的 N 值曲线的上方，这表示模糊检索的出新率要高于传统检索的出新率。从用户的角度上来说，模糊检索一方面更能包含用户的检索需求，也能为用户提供认知以外的一些新的相关性比较大的音乐资源。

结束语 音乐模糊检索方法从技术上满足了音乐情感的多样性，使音乐的数字化工作更贴近音乐的本质特征。实验结果证明，基于模糊数学来实现对音乐情感的检索是可行的、有效的。当然，由于实验设计中音乐样本数目不是很大，也没有考虑不同环境和不同心理状态下用户对音乐情感的鉴定会有所变化，因此导致 A 值和 V 值有一定程度的偏差。下一步的研究中，将针对用户情感波动对 A 值和 V 值的影响进行深入分析，使其根据情感变化自动调整，以实现更加可靠的音乐情感检索。

参考文献

[1] 康庆阳，聂自非. 基于内容的音乐检索及其在媒资管理系统中的应用[J]. 现代电视技术，2007，5：90-95

[2] 鞠源. 基于哼唱的音乐检索系统的研究与探索[J]. 情报杂志，2010，29：160-163

[3] 张宝华，张品. 基于旋律的音乐检索系统[J]. 电声基础，2005，12：4-8

[4] 徐欣，周运，邵曦. 基于音乐情感特征提取的音乐检索分析[J]. 信息通信，2011，115(5)：9-12

[5] 孙晓煜. 自动化音乐情感分类问题的研究[D]. 杭州：浙江大学计算机科学与技术学院，2010

[6] 马希荣，梁景莲. 给予情感音乐模板的音乐检索系统研究[J]. 计算机科学，2009，36(1)：239-242

[7] 谢季坚，刘承平. 模糊数学方法及其应用[M]. 武汉：华中科技大学出版社，2007

[8] 甘露. 论音乐的情感特征[J]. 吉林艺术学院报，2006，67(4)：10-13

[9] Russell J S. A Circumflex Model of Affect [J]. J. Personality Social Psychology，1980，39(6)：1161-1178

[10] Thayer R E. The biopsychology of mood and arousal [M]. Oxford University Press，1989

[11] 王浩，庄钊文. 模糊性可靠分析中的隶属函数确定[J]. 电子产品可靠性与环境试验，2000，8(4)：2-7

[12] Stanley D J，Meyer J P. Two-Dimensional Affective Space：A New Approach To Orienting the Axes [J]. American Psychological Association，2009，9(1)：214-237

[13] 王胜，张石清. 基于二维情感空间的语音情感识别[J]. 台州学院学报，2007，29(6)：49-51

[14] 沈建人. 查准率与查全率之间的关系[J]. 情报探索，2006，102(4)：32-34

[15] Yang Y-H，Lin Y-C，Su Ya-fan，et al. A Regression Approach to Music Emotion Recognition[J]. IEEE Trans. Audio，Speech and Language Processing，2008，16(2)：448-457

[16] 李名标. 模糊分类与模糊匹配相结合的模糊检索[J]. 计算机科学，2000，28(8)：37-39

(上接第195页)

[6] Allcock B，Chervenak A. Data Grid tools：enabling science on big distributed data[C]// Journal of Physics：Conference Series. 2005：1-5

[7] Costa P，Donnelly A. Camdoop：Exploiting In-network Aggregation for Big Data Application[C]// 9th USENIX Symposium on Networked Systems Design and Implementation. April 2012：1-14

[8] 王珊，王会举，覃雄派，等. 架构大数据：挑战、现状与展望[J]. 计算机学报，2011，34(10)：1741-1752

[9] 张延松，焦敏，王占伟，等. 海量数据分析的One-size-fits-all OLAP技术[J]. 计算机学报，2011，34(10)：1936-1946

[10] 琳琳，信俊昌，王国仁，等. 基于Map-Reduce的海量数据高效Skyline查询处理[J]. 计算机学报，2011，34(10)：1875-1796

[11] 覃雄派，王会举，杜小勇，等. 大数据分析—RDBMS与MapReduce的竞争与共生[J]. 软件学报，2012，23(1)：32-45

[12] 吴广君，王树鹏，陈明，等. 海量结构化数据存储检索系统[J]. 计算机研究与发展，2012，49(Suppl.)：1-5

[13] 曾志勇，杨辉，余建坤. 基于HMT和哈希树的Apriori并行算法研究[J]. 计算机工程与设计，2012，33(1)：214-248

[14] 关晓蕾，钱宇华. 基于不完备信息系统的决策树生成算法[J]. 计算机科学，2012，30(1)：156-158

[15] 王柯柯，崔贯勋，倪伟，等. 基于单元的快速的大数据集离群数据挖掘算法[J]. 重庆邮电大学学报：自然科学版，2010，22(5)：673-677