火龙果•整理
uml.org.cn

青岛大学

硕士学位论文

一种基于云计算的数据挖掘平台架构设计与实现

姓名：纪俊

申请学位级别：硕士

专业：计算机软件与理论

指导教师：邵峰晶

20090603

火龙果•整理
uml.org.cn

摘 要

网络技术在带给人们大量信息的同时，也极大地增加了人们从海量数据中发现有用知识的难度，而解决这一问题的努力促进了数据挖掘技术的出现和快速发展。目前，数据挖掘技术已在金融、医疗、军事、管理等诸多领域的决策分析方面被广泛应用。

云计算是能提供动态资源池、虚拟化和高可用性的计算平台。云计算开发平台可被用来开发高性能应用程序，但数据本身具有噪声、异构等问题，而现有的云计算开发平台当前还没有数据规约功能，利用云计算进行数据挖掘的解决方案尚未被提出。针对上述问题，首先,扩展 Google App Engine 开发平台，设计与实现其数据规约功能，主要解决异构数据访问问题与未存在数据类型访问问题；然后，在扩展的 Google App Engine 开发平台基础上设计与实现一数据挖掘系统，通过实验验证数据规约功能的有效性及云计算平台执行数据挖掘操作的高效性。

在数据规约模块设计过程中，通过采用可被扩展的元数据定义、数据集模板定义及数据集定义逐层抽象数据集的数据类型、数据结构、访问地址等信息，实现对异构数据集的抽象以及对未有数据类型的访问。在基于云计算的数据挖掘系统设计过程中，基于分层设计的思想，将该平台的层次自底向上划分为：算法层、任务层和用户层，其中，底层透明的为其上层提供服务，上层通过层间开放接口调用下层服务，使得层与层之间的功能相对独立，易于系统的二次开发。同时各层接口均以 REST 接口方式对外开放，可被开发者嵌入到其应用程序中。在算法层设计过程中设计了多层插件框架结构，增加了算法实现与维护的灵活性。

最后，在对系统开发环境及系统开发关键技术分析的基础上，给出了基于云计算的数据挖掘平台原型的实现过程。并通过实验数据分析，验证了目标的有效性与高效性。

关键词：数据挖掘；云计算；数据规约；分布式系统

火龙果•整理
uml.org.cn

Abstract

The development of network techniques brings people in a great deal of information, it also greatly increase the difficulty to find useful knowledge from mass data. The efforts to solve this problem promote the emergence and rapid development of data mining techniques. At present, the data mining technologies and tools have been used in the financial, medical, military, and many other areas of commercial decision-making analysis.

Cloud computing is a style of computing in which dynamically scalable and often virtualized resources are provided as a service over the Internet. Cloud computing platform can be used to development high capability programmes. However, it does not provide data reduction service which is one of the bases of data-mining. So the method to implement data-mining system on cloud computing platform has not been worked out yet. To solve these problems, in this paper, a data reduction module for accessing isomeric and new type data is designed and implemented to expand Google App Engine cloud platform, further more, a new data-mining system built on top of cloud computing services is designed and implemented to verify the validity of data reduction module and the efficiency of cloud based data-mining process.

Data reduction module supports uniform definition of data sets by using meta-data definition, data set definition and data set instance definition to abstract data type, data structure, location information and so on. Layered thinking model is used and a new muti-layered plugin architecture is induced to enhance the scalability of the system. All system interfaces are RESTful which could be embeded in other applications.

Meanwhile, many important thoughts and methods to design and implement the platform are induced in this paper and a prototype of data-mining platform based on this architecture is introduced at last section of this paper. Experimental results show that the proposed method is very promising.

Key words: data mining; cloud computing; data reduction; distributed system

火龙果•整理
uml.org.cn

学位论文独创性声明

本人声明，所呈交的学位论文系本人在导师指导下独立完成的研究成果。文中依法引用他人的成果，均已做出明确标注或得到许可。论文内容未包含法律意义上已属于他人的任何形式的研究成果，也不包含本人已用于其他学位申请的论文或成果。

本人如违反上述声明，愿意承担由此引发的一切责任和后果。

论文作者签名：张俊　　日期：2009年5月10日

学位论文知识产权权属声明

本人在导师指导下所完成的学位论文及相关的职务作品，知识产权归属学校。学校享有以任何方式发表、复制、公开阅览、借阅以及申请专利等权利。本人离校后发表或使用学位论文或与该论文直接相关的学术论文或成果时，署名单位仍然为青岛大学。

本学位论文属于：

保密 □，在　　年解密后适用于本声明。

不保密☑。

（请在以上方框内打“√”）

论文作者签名：张俊　　日期：2009年5月10日

导 师 签 名：[illegible]　　日期：2009年5月10日

（本声明的版权归青岛大学所有，未经许可，任何单位及任何个人不得擅自使用）

火龙果•整理
uml.org.cn

第一章　引言

1.1 课题研究目的与意义

随着计算机科学与技术的迅速发展，越来越多的数据被存储在计算机存储介质中，这些数据往往是大量的、复杂的、异构的、有噪声的，很难通过人工直接理解，其中蕴含的知识与模式更是难以被发现。而在科研与商业等领域，经常需要分析这些数据，从中发现有价值的信息与模式或对未来做预测。为迎接这些挑战，来自不同学科的研究者汇集到一起，着手开发可以处理不同数据类型的更有效的、可伸缩的技术与工具，数据挖掘（Data Mining）[1,2] 作为一门交叉学科应运而生。这些工作建立在研究者先前使用的方法学和算法之上，在数据挖掘领域达到高潮。特别地是，数据挖掘利用了来自如下一些领域的思想：(1) 来自统计学的抽样、估计和假设检验，(2) 人工智能、模式识别和机器学习的搜索算法、建模技术和学习理论。数据挖掘也迅速地接纳了来自其他领域的思想，这些领域包括最优化、进化计算、信息论、信号处理、可视化和信息检索。数据挖掘又称为数据库中的知识发现(Knowledge Discovery in Database, KDD)是从大量的、不完全的、有噪声的、模糊的、随机的实际应用数据中，提取隐含在其中的、人们事先不知道的、但又是潜在有用的信息和知识的过程。它的出现使得之前很多无法被理解的历史数据得到了利用，为科学研究与商业决策等多个领域起到了指导作用，具有深远的社会和经济意义。

这些年来，随着计算机网络技术特别是网格计算与云计算的发展，越来越多的数据分布式的存储在网络中，这些海量的、分布式的、异构的、复杂的数据给数据挖掘系统的实现带来了两大难题：首先，处理这些数据的复杂度很高，系统的计算能力很难达到要求，此时传统的单机服务器所能提供的有限计算资源往往不能满足要求，需要借助分布式计算技术来实现大规模并行计算。

云计算是能提供动态资源池、虚拟化和高可用性的计算平台。云计算开发平台可被用来开发高性能应用程序，但数据本身具有噪声、异构等问题，而现有的云计算开发平台当前还没有数据规约功能，利用云计算进行数据挖掘的解决方案尚未被提出。针对上述问题，首先,扩展Google App Engine开发平台，设计与实现其数据规约功能，主要解决异构数据访问问题与未存在数据类型访问问题；然后，在扩

火龙果•整理
uml.org.cn

展的Google App Engine开发平台基础上设计与实现一数据挖掘系统，通过实验验证数据规约功能的有效性及云计算平台执行数据挖掘操作的高效性。

1.2 国内外研究现状

为解决上述问题，国内外学者们相继提出了基于集群的[3]、基于三层C/S结构的[4]、基于Agent的[5]等各种分布式并行数据挖掘平台框架，这些框架提高了数据挖掘系统的处理能力，但系统实现相对复杂且适应性较差，往往开发投入很大却只能针对特殊应用。之后，文献[6]总结了前人的经验与思想，提出了知识网格的概念并基于Globus Toolkit[7]设计了一种分布式并行知识发现平台框架，框架充分利用Globus Toolkit所提供的网格计算[8]能力，解决了传统数据挖掘框架计算能力不足的问题，成为了近几年此类问题研究的基石并有很多针对其框架且基于Globus Toolkit的系统实现与改进方法被提出[9,10,11]。但网格计算缺少商业化实现，且Globus Toolkit是基于中间件技术，需要用户通过编程或者安装设置来搭建底层架构，这为系统实现增加了难度，更重要的是，这种框架下实现的系统所部署到的网格的计算能力即服务器的数量与硬件指标是其系统的瓶颈，针对中小企业、科研单位、个人用户而言很难负担起太大的开销来满足处理能力要求较高的系统硬件需求。即使大型机构能提供足够硬件设备，在系统不工作或者非满负荷工作时也会造成很多资源被闲置而浪费。与此同时，配备专业人员来维护网格计算环境也是必需的，这也在一定程度上增加了系统的开销。云计算作为网格计算的升级，在近两年被提出，他具有超大规模、虚拟化、高可靠性、通用性强、高扩展性、按需服务、易用等优点，这些优点可被用来搭建数据挖掘系统解决部分上述问题，而之前未有基于云计算的数据挖掘系统解决方案被提出。

1. 3 论文各部分的内容安排

论文的第二章主要介绍云计算平台及其特点与服务，第三章介绍数据规约工具的设计与实现，第四章详细阐述了基于云计算的数据挖掘系统设计思想、设计要点、特性以及具体层与层之间的关系等。文章的第四章介绍基于本文提出的框架所实现的一个数据挖掘平台的原型系统以及实现系统的关键思想、技术与步骤，第五章通过实验数据验证了数据规约工具的有效性及基于云计算的数据挖掘系统的高效性，第六章是文章的总结，最后一部分是结束语与致谢。

火龙果•整理
uml.org.cn

第二章　云计算平台

2.1 云计算定义

云计算（Cloud Computing）[12, 13]是一种分布在大规模数据中心、能动态的提供各种服务器资源以满足科研、电子商务等领域需求的计算平台。云计算是分布式计算(Distributed Computing)[14]、并行计算(Parallel Computing)[15]和网格计算(Grid Computing)[16]的发展，是虚拟化(Virtualization)[17]、效用计算(Utility Computing)[18]、IaaS(基础设施即服务)、PaaS(平台即服务)、SaaS(软件即服务)等概念混合演进并跃升的结果。云计算平台通过相关调度策略利用虚拟化技术，针对用户的不同需求，动态、透明的提供其所需的虚拟计算与存储资源，并在当前用户不使用时将其资源动态回收供给其他用户，就像发电厂供电一样为用户输送廉价的计算与存储资源，让普通用户实现大规模并行计算与海量数据操作能为可能，也为搭建统一开放的知识网格系统提供了底层支持。

2.2 云计算特点

云计算具有以下特点：

(1) 超大规模。“云”具有相当的规模，Google 云计算已经拥有 100 多万台服务器， Amazon、IBM、微软、Yahoo 等的“云”均拥有几十万台服务器。企业私有云一般拥有数百上千台服务器。“云”能赋予用户前所未有的计算能力。

(2) 虚拟化。云计算支持用户在任意位置、使用各种终端获取应用服务。所请求的资源来自“云”，而不是固定的有形的实体。应用在“云”中某处运行，但用户无需了解、也不用担心应用运行的具体位置。

(3) 高可靠性。“云”使用了数据多副本容错、计算节点同构可互换等措施来保障服务的高可靠性，使用云计算比使用本地计算机可靠。

(4) 通用性。云计算不针对特定的应用，在“云”的支撑下可以构造出各种应用，同一个“云”可以同时支撑不同的应用运行。

火龙果•整理
uml.org.cn

(5) 高可扩展性。“云”的规模可以动态伸缩，满足应用和用户规模增长的需要。

(6) 按需服务。“云”是一个庞大的资源池，可按需购买；云可以象自来水，电，煤气一样计费。

(7) 极其廉价。由于“云”的特殊容错措施可以采用极其廉价的节点来构成云，“云”的自动化集中式管理使大量企业无需负担日益高昂的数据中心管理成本，“云”的通用性使资源的利用率较之传统系统大幅提升，因此用户可以充分享受“云”的低成本优势，经常只要花费几百美元、几天时间就能完成以前需要数万美元、数月时间才能完成的任务。

2.3 云计算与网格计算的对比

Table 2.1 Comparation cloud computing with grid computing

表2.1 云计算与网格计算的对比

		网格	云计算
共有特性		目标一致：虚拟计算机资源并通过各种网络共享它	
功能性方面	虚拟化(Virtualized)	使用中间件，例如 Globus 来集成各种 硬件与操作系统提供虚拟资源	使用管理程序(Hypervisor)例如 Xen 来虚拟操作系统
	数据	使用基于 OGSI-DAI 标准来集成各种不同数据，例如 DB、文件、FTP，来提供统一的数据访问	使用大尺度的文件系统与优化算法来提供海量虚拟数据，例如 GFS、Bigtable、SimpleDB
非功能性方面	自复(Self-healing)	没有统一方法针对不同应用实现	Recovery-oriented computing[19]被提倡 CAP (consistency、availability、partition-tolerance)[20]理论被应用
	面向服务(Service-oriented)	基于 SOAP 的应用程序领域	基于 Web 2.0 应用程序领域
	伸缩性(Scalability)	提供给单一服务的中等尺度的伸缩性(一台物理计算机为单位，受限于网格自身机器数目)	为一至多个服务提供海量的线性的动态伸缩性(一台虚拟计算机为单位)

火龙果•整理
uml.org.cn

可见：云计算与网格计算目标是一致的，都是虚拟计算机资源并通过各种网络共享它们为上层应用服务，但云计算在理论上增加了自复方面的支持，在伸缩性、自复性以及其他各方面均优于网格计算。特别是其以虚拟计算机为粒度的动态伸缩性为搭建高伸缩性高可靠性而廉价的系统开发带来了极大便利。

2.4 云计算架构

云计算抽象了计算与存储资源并动态的分配给需要使用的用户，它是一个高伸缩性、高可靠性、底层透明、安全的底层架构并具有友好的监控与维护接口。如图2.1，在其上开发应用时只需要按照其应用程序接口规范调用所需资源即可，不必像使用 Globus Toolkit 那样花费大量时间来降低系统所需吞吐量以减少硬件投资，其使用费用跟总的资源使用量成正比而不像以往跟系统吞吐量成正比。如此用户只需关心业务逻辑实现，针对数据挖掘实现而言，我们可以把各种算法部署到云计算平台运行然后通过云计算平台的控制面板或者接口设定目标响应时间就能得到满意的结果。

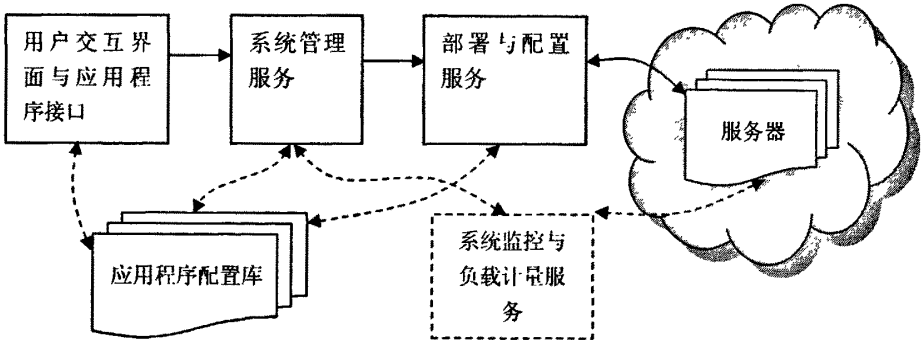

Fig.2.1 Cloud Computing architecture

图 2.1 云计算架构

云计算平台具有动态伸缩性，如图 2.2 所示，一个应用程序在资源请求很少的时候可能执行在一个粒度的虚拟机上，而当资源请求增长时，最先成为系统瓶颈的往往是当前运行环境的计算能力，这时云计算平台通过系统监控服务发现当前运行环境负载过高，自动动态从云计算资源池中请求新的虚拟机加入到当前运行环境，以集群的方式线性增长当前运行环境的计算能力以满足应用程序的资源请求。而当应用程序的资源请求进一步增长时，这时不只运行环境的计算能力，数据库端也将成为瓶颈，特别是当虚拟机数量的增加所带来的并发与协调执行代价过高时，这时数据库所在的运行环境也将被动态扩展以满足海量的资源请求。而当应用程序资源

火龙果•整理
uml.org.cn

请求降低时，则是相反的情况，虚拟机逐步被回收回资源池以待被其他当前高资源请求的应用所使用。

如此一来，世界各地的应用程序可通过共享同一个庞大的云计算资源池来获得超大的系统吞吐能力以满足用户在某些情况下所需要的超高计算或者存储资源请求，而付出的代价却只是其总的资源使用量的费用。以上系统的动态扩展与收缩过程并不需要用户干预，系统会自动进行，开发者在其平台上开发时除了按照其规范并遵循程序易于被横向扩展的原则外，跟开发本地应用程序没有太大区别，这给系统开发者与使用者都带来了很大的实惠。

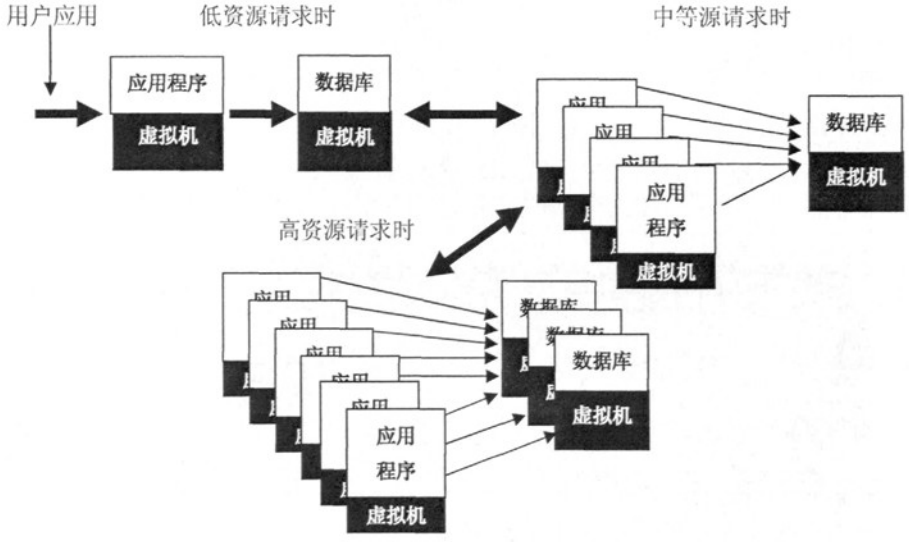

Fig.2.2 **Cloud Computing application run mode**

图 2.2 **云计算应用程序运行方式**

2. 5 现有云计算平台简介

现有的云计算平台已趋于成熟，像 Google App Engine[21, 22]与 Amazon Elastic Compute Cloud[23]等，基于云计算实现的应用可以方便的部署其上使用其计算资源。而数据存储方面， Google Bigtable[24]与 Amazon Simple Storage Service[25] 等为实现海量数据的分布式存储与访问提供了支持。云计算平台的商业化发展以与成熟为实现本文提出的数据挖掘系统提供了很好的底层架构支持。本文提出的系统的实现是基于 Google App Engine，下文将主要介绍 Google App Engine 各方面特点。

火龙果•整理
uml.org.cn

作为新一代的基于云计算的网络程序开发平台，Google App Engine 使用户可以在 Google 的基础架构上开发与运行网络应用程序。Google App Engine 应用程序易于构建和维护，并可根据用户访问量和数据存储需要的增长轻松扩展。使用 Google App Engine 将不再需要维护服务器：开发者只需上传其应用程序，它便可立即为用户提供服务。开发者可以与全世界的人共享应用程序，也可以限制为只有授权用户成员可以访问。

应用程序环境方面，通过 Google App Engine，即使在重载和数据量极大的情况下，也可以轻松构建能安全运行的应用程序。该环境包括以下特性：

1. 动态网络服务，提供对常用网络技术的完全支持

2. 持久存储有查询、分类和事务

3. 自动扩展和载荷平衡

4. 用于对用户进行身份验证和使用 Google 帐户发送电子邮件的 API

5. 一种功能完整的本地开发环境，可以在您的计算机上模拟 Google App Engine

6. Google App Engine 应用程序是使用 Python 编程语言实现的。该运行时环境包括完整 Python 语言和多数 Python 标准库。

目前，Google App Engine 支持 Python 语言和 Java 语言，将来它将支持更多语言。

在安全性方面，Google App Engine 引入了 Sandbox 技术。在安全环境中运行的应用程序，仅提供对基础操作系统的有限访问权限。这些限制让 App Engine 可以在多个服务器之间分发应用程序的网络请求，并可以启动和停止服务器以满足访问量需求。Sandbox 将您的应用程序隔离在它自己的安全可靠环境中，该环境与网络服务器的硬件、操作系统和物理位置无关。

安全 Sandbox 环境的限制实例包括：

火龙果•整理
uml.org.cn

1. 应用程序只能通过提供的网址获取和电子邮件服务和 API 访问互联网中的其他计算机。其他计算机只能通过在标准端口上进行 HTTP（或 HTTPS）请求来连接至该应用程序。

2. 应用程序无法向文件系统写入。应用程序只能读取通过应用程序代码上传的文件。该应用程序必须使用 App Engine 数据库存储所有在请求之间持续存在的数据。

3. 应用程序代码仅在响应网络请求时运行，且必须在几秒钟内返回响应数据。请求处理程序不能在响应发送后产生子进程或执行代码。

在应用程序开发与运行语言环境方面，App Engine 提供了一个使用 Python 编程语言的运行时环境。将来的版本将考虑使用其他编程语言和运行时环境配置。Python 运行时环境使用 Python 2.5.2 版。该环境包括 Python 标准库。Python 环境为数据库、Google 帐户、网址获取和电子邮件服务提供了丰富的 Python API。App Engine 还提供了一个称为 webapp 的简单 Python 网络应用程序框架，从而可以轻松开始构建应用程序。方便起见，App Engine 还包括 Django 网络应用程序框架 0.96.1 版。只要这些库是完全使用 Python 实现并且不需要任何不受支持的标准库模块，开发者就可以使用开发的应用程序上传其他第三方库。这也是本系统插件系统实现的技术保障，通过这项支持，数据挖掘平台的各种算法作为库可被动态更新与上传。

数据库方面，App Engine 提供了强大的分布式数据存储服务 Big Table，其中包含查询引擎和事务功能。就像分布式网络服务器随访问量增加一样，该分布式数据库也会随数据而增加。该 App Engine 数据库与传统关系数据库不同。数据对象有一类和一组属性。查询可以检索按属性值过滤和分类的给定种类的实体。属性值可以是受支持的属性值类型中的任何一种。数据库的 Python API 包括一个可以定义数据库实体结构的数据建模界面。数据模型可以指示属性值必须位于给定范围内，如果未给定任何范围，还可以提供默认值。应用程序可以根据需要向数据提供或多或少的结构。数据库使用乐观锁定进行并发控制。如果有其他进程尝试更新某实体，而同时该实体位于以固定次数进行重新尝试的事务中，此时该实体将更新。应用程序可以在一个事务中执行多项数据库操作（全部成功或者全部失败），从而确保数据的完整性。数据库通过其分布式网络使用“实体组”实现事务。一个事务操作一个组内的实体。同一组的实体存储在一起，以高效执行事务。应用程序可以在实体创建时将实体分配到组。

火龙果•整理
uml.org.cn

Google App Engine 云计算开发平台的上述特点使其可被用来开发高性能的数据挖掘应用程序，但数据本身具有噪声、异构等问题，而当前 Google App Engine 没有数据规约功能以访问异构数据。所以本文首先要做的是扩展其平台使其具有数据规约功能。

第三章　数据规约工具的设计与实现

3. 1 数据规约工具设计

数据集(Data set)是一个数据的集合，是数据挖掘的对象。数据集实例是包括访问地址在内的一个具体数据集。数据规约工具目标是扩展Google App Engine开发平台使其支持异构数据与未存在类型数据的访问。如图3.1所示，数据规约工具即可被数据挖掘系统调用，也可被其他云计算平台上运行的应用调用以访问异构数据。

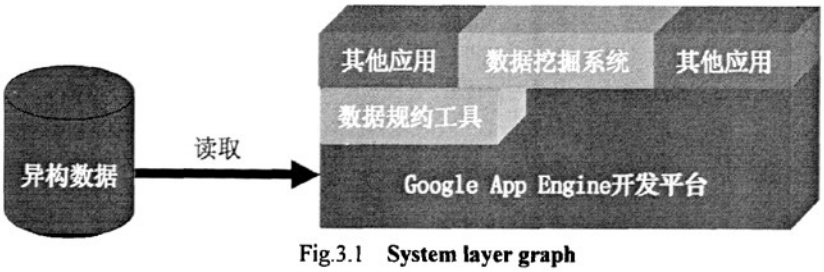

Fig.3.1　**System layer graph**

图 3.1　**系统层次图**

用户在访问其数据集实例之前，首先通过数据规约工具的定义管理模块添加一数据集实例定义文件描述其数据集实例，然后使用其定义文件ID调用数据规约工具的数据访问模块进行数据读写。

火龙果•整理
uml.org.cn

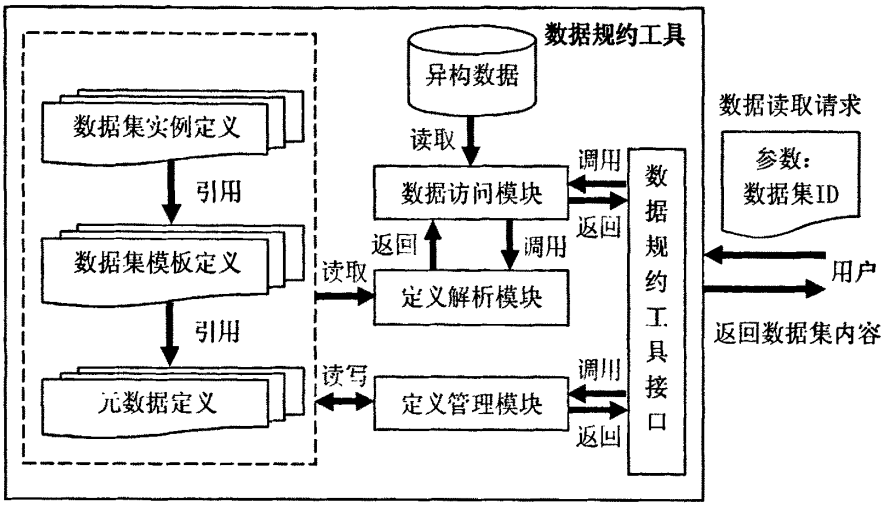

Fig.3.2 **Relationship between inner modules of data reduction tool**

图 3.2 数据规约工具内部模块关系图

其中，数据访问模块提供数据集读取服务，定义管理模块提供数据定义服务与、数据集定义服务、数据集注册与注销服务，定义解析模块包括数据解析服务、数据集解析服务与数据集实例解析服务。图3.3描述了数据规约工具各服务、各定义文件、异构数据集及其之间的关系，箭头代表调用。

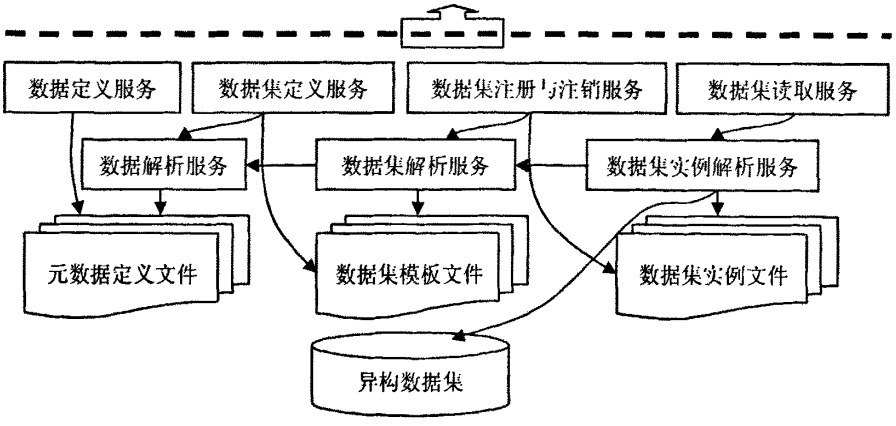

Fig.3.3 **Relationship between services of data reduction tool**

图 3.3 数据规约工具各服务关系

火龙果•整理
uml.org.cn

定义解析模块通过采用可被扩展的元数据定义、数据集模板定义及数据集定义逐层抽象数据集的数据类型、数据结构、访问地址等信息，实现对异构数据集的抽象以及对未有数据类型的访问。图 3.4 描述了数据规约工具各服务、各定义文件、异构数据集及其之间的关系，箭头代表调用。其中，元数据定义文件存放每个数据类型的定义，它由只向管理员开放的数据定义服务维护，它为数据集模板定义提供了原子定义。数据集模板文件存放用户自定义数据集模板，它由数据集定义服务维护并被数据集注册服务使用，与实际数据集对应（配置具体数据集位置、数据源类型等）而实例化为数据集实例文件。数据集实例文件则是数据访问的依据，数据访问组件调用服务通过它选择指定的数据引擎访问指定位置的数据，通过数据集解析服务返回此数据集的含义被各种算法作为参数使用，也就是说，在数据集被访问时，用户不但可以像以往一样得到数据的内容，而且可以获得数据集的定义，此定义最终被解析为能被各种算法所识别的各种原子类型数据的组合从而达到数据层抽象的目的。具体的调用流程如下：

用户定义数据集时，系统调用数据集定义服务查看要定义的数据集类型是否在已定义的数据集模板内，如果模板已存在，则只需调用数据集注册与注销服务将其数据集存储位置与数据引擎等信息注册到数据集实例文件即可，而当模板均不符合要求，则先调用数据集定义服务，并利用数据解析服务所提供的数据元定义来定义目标数据集模板，再来调用数据集注册与注销服务就完成了。

用户访问数据集时，系统调用数据集存取服务，此服务通过数据访问组件服务获得数据集的实例文件并按照实例文件记录的具体数据引擎类型启用对应服务连接数据源返回数据内容，同时，数据访问组件服务调用数据集解析服务，并最终通过数据解析服务获得数据集的定义并返回给用户。

3. 2 数据规约工具实现

3.2.1 数据定义规范

数据定义规范包括数据元定义文件规范、数据集模板文件规范、数据集实例文件规范。三种规范的意义在于抽象数据集结构，数据元定义文件抽象每种数据类型，数据集模板文件定义各种数据集模板，数据集实例文件则定义具体数据集实例。

XML（Extensible Markup Language）即可扩展标记语言，它与 HTML 一样，都是 SGML(Standard Generalized Markup Language,标准通用标记语言)。XML 是

火龙果•整理
uml.org.cn

Internet 环境中跨平台的，依赖于内容的技术，是当前处理结构化文档信息的有力工具。扩展标记语言 XML 是一种简单的数据存储语言，使用一系列简单的标记描述数据，而这些标记可以用方便的方式建立，虽然 XML 占用的空间比二进制数据要占用更多的空间，但 XML 简单易于掌握和使用。

XML 的简单使其易于在任何应用程序中读写数据，这使 XML 成为数据交换的唯一公共语言，虽然不同的应用软件也支持其它的数据交换格式，但他们都将支持 XML，这意味着各种程序可以更容易的与 Windows、Mac OS、Linux 以及其他平台下产生的信息结合，然后可以加载 XML 数据到程序中并分析他，并以 XML 格式输出结果。为提高系统的可扩展性与平台无关性， 在本文中，XML 语言被用来定义数据规范。

一个关于商品类型定义的实例如图 3.4 所示。在数据元定义文件中所有元数据的定义在<fields>…</fiels>标签内，每个字段的定义定义在<field>…</field>内，其 name 属性作为标识符被各种数据集模板文件引用，type 属性对应计算机开发语言的内部数据结构类型，其他属性则可选，针对具体内部类型有关。此文件由系统管理员维护，在定义数据集模板文件时被载入并被解析，用于定义数据集模板文件。

数据集模板文件中所有数据集模板定义在<templates>…</templates>标签内，每个数据集模板定义在<template>…</template>内，其 name 属性作为标识符被各种数据集实例文件引用，数据集每列则定义在<field>…</field>内，其具体数据结构通过 type 属性引用之前定义的数据元定义文件中的数据元定义，而 column 属性则描述此列所在的物理位置，其中 ALL 预定义为表示全部，OTHER 预定义为其他未被定义的数据列。

数据集实例文件中所有数据集实例定义在<instances>…</instances>标签内，每个数据集实例定义在<instance>…</instance>内，其 name 属性作为标识符用于被用户所查询，具体数据集实例所引用的模板通过 template 标签描述，通过其 name 属性指向数据集模板文件。而具体数据集类型与所在位置以及访问引擎等信息则由 datasource 标签定义，其 type 属性表示其数据访问引擎类型，location 属性则描述具体的数据集实例所在的物理位置，其他属性则与具体的数据集类型有关，例如 database 属性则是在数据集类型为 SQL Server 与 Oracle 等关系数据库时才有用的属性，用于指向具体的数据库。

火龙果•整理
uml.org.cn

数据元定义文件

```
<?xml version="1.0" encoding="UTF-8" ?>
- <fields>
        <field name="int" type="INTEGAR" lower="-65535" upper="65535"
      />
        <field name="float" type="FLOAT" />
        <field name="string" type="STRING" maxlength="256" />
        <field name="remark" type="REMARK" />
  ...
  </fields>
```

数据集模板文件

```
<?xml version="1.0" encoding="UTF-8"?>
<templates>
    <template name="全浮点数">
        <field type="float"
column="ALL"/>
    </template>
    <template name="商品模板1">
        <field name="编号" type="int"
column="1"/>
        <field name="名称" type="string"
column="2"/>
        <field name="价格" type="int"
column="3"/>
        <field name="销售量" type="int"
column="4"/>
        <field name="销售时间"
type="time" column="5"/>
        <field type="remark"
column="OTHER"/>
    </template>
    ...
</templates>
```

数据集实例文件

```
<?xml version="1.0"
encoding="UTF-8"?>
<instances>
    <instance name="某图像SVM分类用
特征库">
        <template name="全浮点数"/>
        <datasource type="TEXT"
location="../dataset"
database="SAR_img1.dat"
seperator=" ">
    </instance>
    <instance name="某商品数据库">
        <template name="商品模板
1"/>
        <datasource type="MSSQL"
location="http://mssql03.1and1.fr"
database="db183881703"
username="jijun"
password="aWUmVJVx"/>
    </instance>
    ...
</instances>
```

Fig. 3.4 **Relationship between three XML files**

图 3.4 三个 XML 文件之间的关系

火龙果•整理
uml.org.cn

3.2.2 基于 DOM 的 XML 解析方法

DOM 是 Document Object Model 文档对象模型的缩写。根据 W3C DOM 规范（http://www.w3.org/DOM/），DOM 是一种与浏览器，平台，语言无关的接口，使得你可以访问页面其他的标准组件。DOM 解决了 Netscape 的 Javascript 和 Microsoft 的 Jscript 之间的冲突，给予 WEB 设计师和开发者一个标准的方法，让他们来访问他们站点中的数据、脚本和表现层对像。

DOM 是以层次结构组织的节点或信息片断的集合。这个层次结构允许开发人员在树中导航寻找特定信息。分析该结构通常需要加载整个文档和构造层次结构，然后才能做任何工作。由于它是基于信息层次的，因而 DOM 被认为是基于树或基于对象的。

DOM 的优势主要表现在：易用性强，使用 DOM 时，将把所有的 XML 文档信息都存于内存中，并且遍历简单，支持 XPath，增强了易用性。DOM 的缺点主要表现在：效率低，解析速度慢，内存占用量过高。

另一方面，DOM 还提供了一个 API，允许开发人员添加、编辑、移动或删除树中任意位置的节点，从而创建一个引用程序。如下图所示，这个模型就是基于树型数据结构的，他在使用之前必须加载整个文档或者至少是结构良好的文档片段。

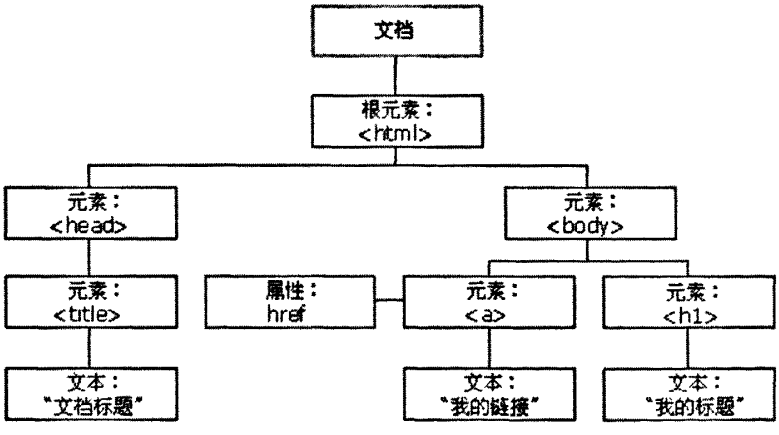

Fig.3.5 DOM Tree Structure

图 3.5 DOM 树形结构

解析器是一个软件应用程序，设计用于分析文档（这里是指 XML 文件），以及做一些特定于该信息的事情。在诸如 SAX 这样基于事件的 API 中，解析器将向某

火龙果•整理
uml.org.cn

种监听器发送事件。在诸如 DOM 这样基于树的 API 中，解析器将在内存中构造一颗数据树。

作为 API 的 DOM 从 DOM Level 1 开始，DOM API 包含了一些接口，用于表示可从 XML 文档中找到的所有不同类型的信息。它还包含使用这些对象所必需的方法和属性。Level 1 包括对 XML 1.0 和 HTML 的支持，每个 HTML 元素被表示为一个接口。它包括用于添加、编辑、移动和读取节点中包含的信息的方法，等等。然而，它没有包括对 XML 名称空间（XML Namespace）的支持，XML 名称空间提供分割文档中的信息的能力。

DOM Level 2 添加了名称空间支持。Level 2 扩展了 Level 1，允许开发人员检测和使用可能适用于某个节点的名称空间信息。Level 2 还增加了几个新的模块，以支持级联样式表、事件和增强的树操作。

当前正处于定稿阶段的 DOM Level 3 包括对创建 Document 对象（以前的版本将这个任务留给实现，使得创建通用应用程序很困难）的更好支持、增强的名称空间支持，以及用来处理文档加载和保存、验证以及 XPath 的新模块；XPath 是在 XSL 转换（XSL Transformation）以及其他 XML 技术中用来选择节点的手段。

DOM 的模块化意味着作为开发人员，您必须知道自己希望使用的特性是否受正在使用的 DOM 实现所支持。

DOM 推荐标准的模块性质允许实现者挑选将要包括到产品中的部分，因而在使用某个特定的特性之前，首先确定该特性是否可用可能是必要的。本教程仅使用 DOM Level 2 Core API，不过在着手您自己的项目时，了解如何能够检测特性是有所帮助的。

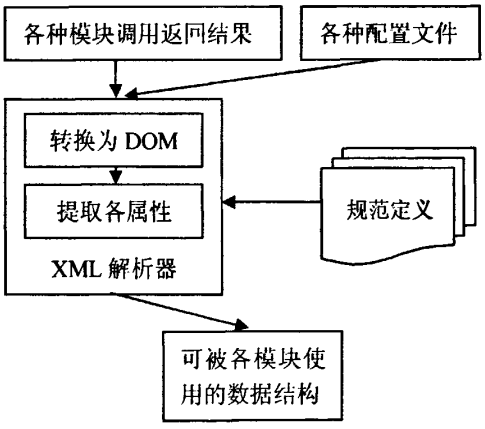

Fig.3.6 XML parsing process

图 3.6 XML 解析过程

如上图所示，本系统利用基于DOM的XML解析方法，用Python实现了XML解析器，解析器先将各种XML内容转换为DOM，然后以树型结构遍历所有属性，再根据规范定义理解其语义并将结果封装成可被各模块使用的数据结构，在这里，我们使用的是Python语言里的字典数据结构。

火龙果•整理
uml.org.cn

第四章　基于云计算的数据挖掘系统架构

4.1 目标系统模型

目标系统搭建于云计算之上透明的为各种终端用户提供用户界面的服务，也为基于本系统开发的应用程序提供开放接口的支持，用户既可以通过各种终端访问系统的用户界面来使用系统，也可以通过其他应用程序调用系统提供的开放接口来间接的使用系统提供的各种服务。而无论是哪种情况，用户都不需要了解系统的实现也不需要担心系统的计算与存储能力的不足而只需要关心选择什么样的算法处理什么样的数据，并最终以任务的方式部署给系统去执行而最终得到数据挖掘结果。此外，数据挖掘平台内部模块均通过用户界面与开放接口提供服务，三层模块按照抽象层次由低到高为算法模块、任务模块，其中开放接口开放的服务为所有外部可见服务，涉及系统管理等需高级权限使用的功能仅能通过系统用户界面调用以保障系统安全性，用户界面可直接调用开放接口实现外部可调用的服务。

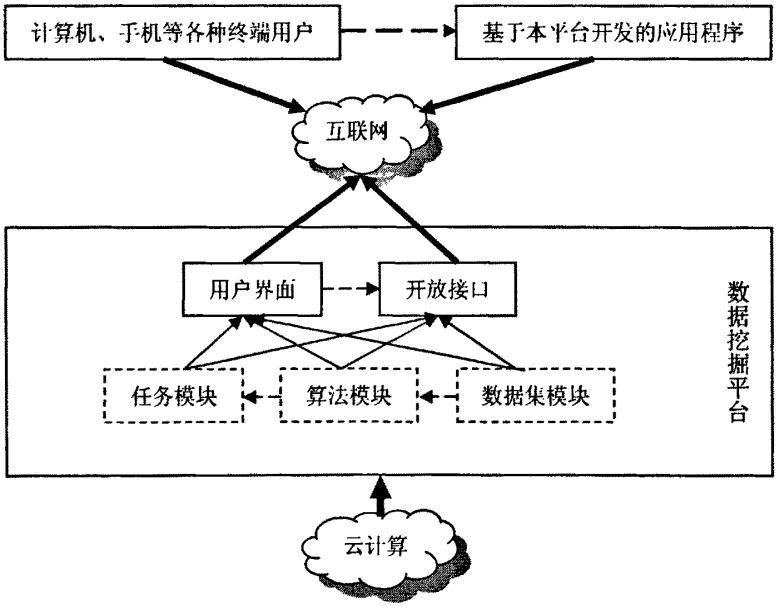

Fig.4.1　**Trarget System Model**

图 4.1　**目标系统模型**

火龙果•整理
uml.org.cn

4.2 功能层次框架设计

如图4.2所示，自底向上每层都透明的为其上层服务，最底层为云计算平台提供的应用程序接口，最顶层为用户界面与开放接口，通过调用开放接口，用户可以共享数据集、调用数据清洗与挖掘算法以及可视化算法，并可以方便的把它们集成到用户自己的应用中，从而实现平台的开放性。下面自底向上详细介绍中间各层提供的服务：

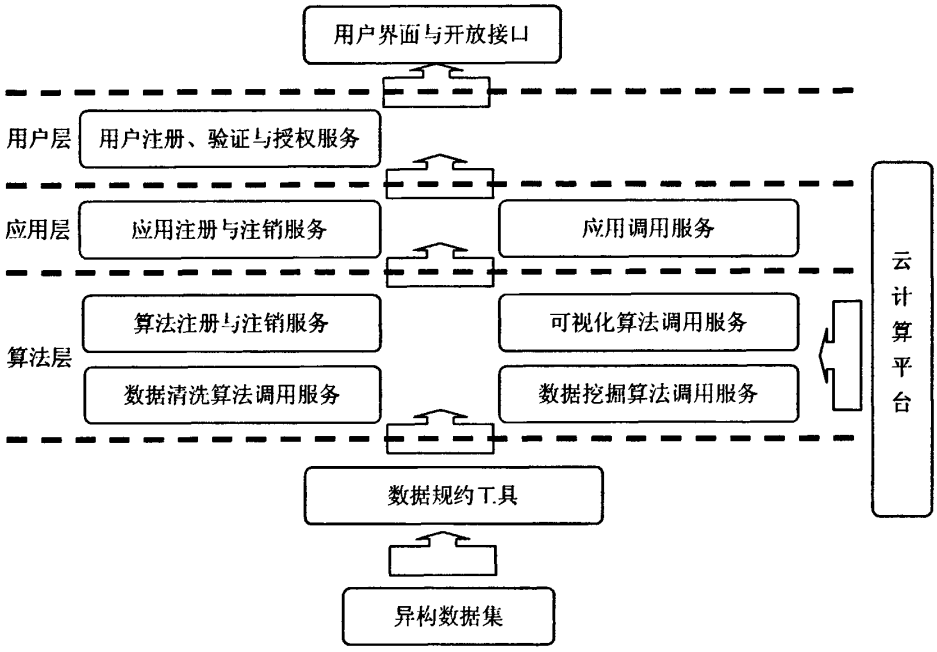

Fig.4.2 **Cloud Computing based Data Mining Platform architecture layers**

图 4.2 **基于云计算的数据挖掘平台分层架构**

4.2.1 算法层

此层使用下层提供的统一数据源实现各种算法调用及其管理接口，因为三类算法的执行顺序以及返回结果的不同而分开。

- 数据清洗算法调用服务：针对有噪声数据的数据集在执行数据挖掘算法之前的预处理方法调用接口，清洗之后的数据将通过数据层存入云计算平台提供的存储空间为接下来的数据挖掘服务。
- 数据挖掘算法调用服务：使用之前数据清洗过的或者其他不需要清洗的数据

火龙果•整理
uml.org.cn

进行数据挖掘的统一调用接口。

- 可视化算法调用服务：将数据挖掘的结果以表格或图形等方式呈现的调用接口。
- 算法注册与注销服务：算法管理模块，以插件方式管理各种算法模块。

4.2.2 应用层

此层抽象其下两层的操作，将整个数据挖掘流程所涉及的数据、算法及其之间关系与顺序描述为任务并提供以应用为单位的调用与维护接口。

- 应用调用服务：提供已注册应用的调用接口。
- 应用注册与注销服务：应用管理模块，以插件方式管理各种任务定义文件。

4.2.3 用户层

此层提供用户身份验证与授权功能。

- 用户注册、验证与授权服务：提供用户身份验证与授权接口，授权信息将作为调用下层各服务的通行证以保障平台的安全性。用户管理接口也由此服务提供。

以上各层各服务均以XML作为通讯语言[26]并基于表象化状态转变的Web服务[27]（RESTful Web Services）[28]形式内部调用以更好的支持各层的可伸缩性，并最终以开放接口形式对外开放。即用户可基于任意层做开发，将已有的服务导入其系统内，这大大增强了系统的开放性跟易用性，是之前数据挖掘平台架构所未有的。

4.3 设计要点

4.3.1 插件系统框架

插件是根据一定应用程序开发接口规范实现的程序。如图4.3，每个插件都由向上层提供服务的实现扩展点、业务逻辑、调用下层的新扩展点三部分组成，以上三部分又由一个负责模块管理的绑定包跟各种服务构成。绑定包包含一个服务说明接口跟各种服务调用接口，服务说明接口返回各服务的调用方法及规范被上层调用者使用并向用户提供具体服务的参数信息。绑定包的接口符合统一的规范，这样一旦插件被置于平台的特定目录，插件可被平台动态的辨识与加载。同时，算法插件的服务接口函数的参数使用的是上节提到的当前算法所能接受的原子类型数据的组

火龙果•整理
uml.org.cn

合，也就是说，算法的实现不再像以往那样针对特定个数、特定排列的数据的具体实现，而是在满足算法实现的前提下利用之前提到的抽象的数据来提供尽可能多的兼容性。虽然算法的实现难度相对提高了，但算法的复用性大大增强，一旦实现即可被平台的其他用户所使用来处理各种数据。

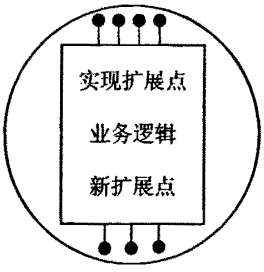

Fig.4.3 Structure of Plugin

图 4.3 插件结构

插件系统框架有两大优点：系统扩展性好与有利于模块化开发与开放的团队开发。出于以上目的，本文提出的平台基于OSGi进行了分层设计。OSGi[29] 是一个组件和服务模型，是用于需要长运行时间、动态更新和对运行环境破坏最小的系统。如图5，本文提出的插件系统框架由内核层、核心插件层、用户自定义层三层组成，除了负责调用云计算平台的资源来实现最底层架构的内核层外均是插件，凡是插件的部分都是可被替换的，核心插件层由系统的管理员负责开发与维护，而用户自定义插件层则对外开放由用户开发与维护，这极大增强了系统的扩展性与开放性。上一节提到为屏蔽数据异构性，本文提出的方法是把数据集定义解析为能被各种算法所识别的各种原子类型数据的组合，而这里的用户自定义插件层的插件均是需要用到各种数据集，所以用户自定义插件层的插件所提供的服务的参数均是符合之前提到的元数据定义文件的约束的，也就是说提供这些服务的函数参数是由元数据定义的数据类型与个数组合而成的。通过实现这种多层的插件框架的另一个优点是算法模块易于添加与修改，算法开发更灵活，复杂的数据挖掘过程可分解为多步实现。例如，某算法只实现数据分析部分，其执行结果由其子插件进行可视化，且可以实现多种可视化结果。具体插件机制如下：

1. 系统内核在主程序执行时遍历插件目录
2. 每个被主程序遍历得到的算法插件接口被动态加载到系统功能列表
3. 递归步骤1，此时的主程序为当前算法插件
4. 不再发现子插件时当前递归结束，否则继续递归过程3

火龙果•整理
uml.org.cn

注：每个算法插件模块的功能描述及所需参数可通过算法模块接口返回

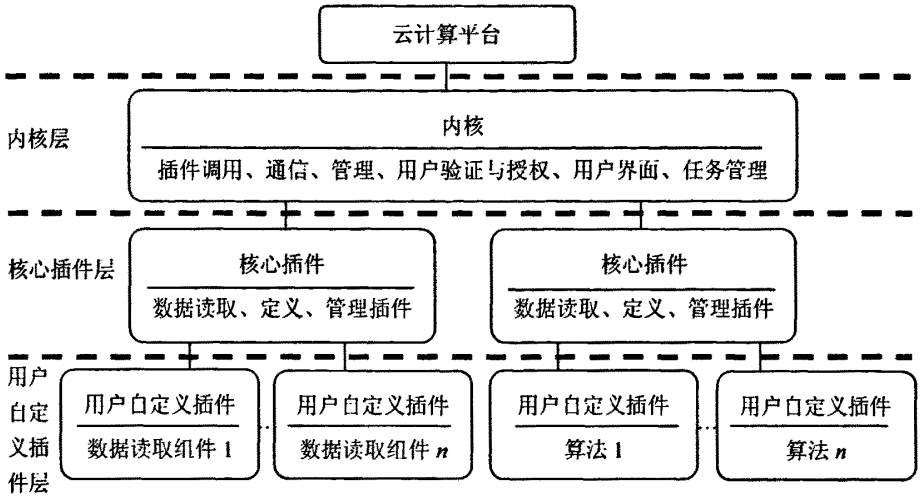

Fig.4.4 Data layer architecture

图 4.4 结构分层设计

4.3.2 开放接口设计

开放接口用于其他应用基于数据挖掘平台做开发时使用，通过这些接口，开发者可使用数据挖掘平台所提供的各种资源与服务。为了达到使开放接口高效、简单直观、具有高伸缩性等目的，本文提出的平台接口全部基于 REST。REST 是英文 Representational State Transfer 的缩写，中文翻译：表述性状态转移。

Roy T. Fielding 在他的博士学位论文[27]中定义了术语 REST。Roy 是许多基本 Web 协议的主要设计者，其中包括 HTTP 和 URIs。在论文中，Roy 首先定义一种方法论来谈论架构风格——高级、抽象的模式，来表达架构方法背后的核心理念。每一个架构风格由一系列的约束定义形成。架构风格的例子包括“没有风格”、管道和过滤器、客户端/服务器、分布式对象以及 REST。REST 中的资源所指的不是数据，而是数据和表现形式的组合，而由于他们的表现形式不同，所以被归为不同的资源。资源标识符就是 URI(Uniform Resource Identifier)，通过 URI 对全部资源进行唯一的标识。REST 是基于 Http 协议的，任何对资源的操作行为都是通过 Http 协议来实现。Http 把对一个资源的操作限制在 4 个方法以内： GET, POST, PUT 和

火龙果•整理
uml.org.cn

DELETE，这正是对资源 CRUD（Create, Read, Update, Delete）操作的实现。由于资源和 URI 是一一对应的，执行这些操作的时候 URI 是没有变化的，这和以往的 Web 开发有很大的区别。这极大的简化了 Web 开发，也使得 URI 可以被设计成更为直观的反映资源的结构，这种 URI 的设计被称作 RESTful 的 URI。REST 之所以可以提高系统的可伸缩性，是因为它要求所有的操作都是无状态的。由于没有了上下文(Context)的约束，做分布式和集群的时候就更为简单，也可以让系统更为有效的利用缓冲池。并且由于服务器端不需要记录客户端的一系列访问，也减少了服务器端的性能。

REST 在原有的 WEB 架构的三个规范：客户一服务器、无状态性、缓存基础上增加了三个新规范：统一接口，分层系统和按需代码。

REST 的设计准则：

REST 架构是针对 Web 应用而设计的，其目的是为了降低开发的复杂性，提高系统的可伸缩性。REST 提出了如下设计准则：

1. 网络上的所有事物都被抽象为资源（resource）；

2. 每个资源对应一个唯一的资源标识符（resource identifier）；

3. 通过通用的连接器接口（generic connector interface）对资源进行操作；

4. 对资源的各种操作不会改变资源标识符；

5. 所有的操作都是无状态的（stateless）。

基于以上准则，平台的接口被定义如下：

火龙果•整理
uml.org.cn

Table 4.1 Open APIs list

表4.1 开放接口列表

URI	操作	描述
/algorithms	GET	返回平台所有已实现算法名与算法 ID 及其基本描述信息
/algorithms/算法 ID	GET	返回此算法 ID 所对应的算法包括参数在内的详细描述信息
	POST	修改此算法 ID 所对应的算法描述信息，仅限管理员使用
	PUT	调用此算法 ID 所对应的算法
	DELETE	删除此算法 ID 所对应的算法，仅限管理员使用
/datasets	GET	返回平台所有已注册数据集名称与数据集 ID 及其基本描述信息
/datasets/数据集 ID	GET	返回此数据集 ID 所对应的数据集包括其物理数据所在位置、字段类型在内的详细描述信息
	POST	修改此数据集 ID 所对应的数据集描述信息，仅限管理员使用
	PUT	访问此数据集 ID 所对应的数据集
	DELETE	删除此数据集 ID 所对应的数据集，仅限管理员使用
/tasks	GET	返回所有的任务名称与任务 ID 及其基本描述信息
/tasks/任务 ID	GET	返回此任务 ID 对应的扩展功能的详细描述信息
	POST	修改此任务 ID 对应的内容，仅限任务所有者使用
	PUT	调用此任务 ID 对应的任务
	DELETE	删除此任务 ID 对应的任务，仅限任务所有者使用
/expansion	GET	返回所有的用户名称与用户 ID 及其基本描述信息
/user/用户 ID	GET	返回此用户 ID 对应的用户详细描述信息
	POST	修改此用户 ID 对应的用户描述信息，仅限用户自身使用
	PUT	此用户 ID 对应的用户登录与注销
	DELETE	删除此用户 ID 对应的用户，仅限管理员使用
/admin	GET	返回所有的扩展功能名称与扩展功能 ID 及其基本描述信息
/admin/管理功能 ID	GET	返回此管理功能 ID 对应功能的详细描述信息，仅限管理员使用
	POST	修改此管理功能 ID 对应的扩展功能描述信息，仅限管理员使用
	PUT	调用此管理功能 ID 对应的扩展功能，仅限管理员使用
/expansion	GET	返回所有的扩展功能名称与扩展功能 ID 及其基本描述信息
/expansion/扩展功能 ID	GET	返回此扩展功能 ID 对应的扩展功能的详细描述信息
	POST	修改此扩展功能 ID 对应的扩展功能描述信息，仅限管理员使用
	PUT	调用此扩展功能 ID 对应的扩展功能
	DELETE	删除此扩展功能 ID 对应的扩展功能，仅限管理员使用

火龙果•整理
uml.org.cn

如表 4.1，其中，GET 操作是无参数的，直接发送 HTTP 请求给服务器，而 POST、PUT、DELETE 操作均可传送参数，传送参数的方法这里不再赘述，调用接口时通过 GET 得到对应资源的描述获知使用该资源需给出的参数，针对不同的操作选择对应的方法并将参数发送给服务器即可。而这些开放接口即可以被系统自身所调用，也可以被二次开发用户直接调用嵌入其系统中，在这里，本系统也可以被看作是基于这些开放接口的一个二次开发的应用。所有开发者在获得权限的前提下都可以直接调用指定的接口将其服务嵌入当前的其他应用程序开发当中。

由于基于 REST 的这些接口具有无态的优点，在同一局域网内对一个服务的重复调用可被浏览器内置的缓冲所代替，这大大减轻了服务器的负担，另一方面，当用户量过大导致这些底层服务成为系统瓶颈时，系统可通过横向增加服务器数量来实现线性的吞吐量扩展，由于云计算平台可利用其强大的服务器资源池动态的实现这点，这使系统在理论上可以提供无穷大的吞吐能力满足系统在性能上的需求。

火龙果•整理
uml.org.cn

第五章　基于云计算的数据挖掘系统实现

基于上述框架，实现了一基于云计算的数据挖掘系统的原型，网址为：http://data-mining.appspot.com。其中两种经典的数据挖掘算法 ID3 决策树与 K-means 聚类以插件方式实现并带有基本的测试用数据集以验证数据规约工具以及本系统的有效性与执行效率。

5.1 系统开发环境

系统开发工具为 Eclipse3.4[30]，开发语言为 Python[31]，基于 Google 开放的云计算开发平台 App Engine SDK[32]开发。

Eclipse 是著名的跨平台的自由集成开发环境（IDE）。最初主要用来 Java 语言开发，但可通过插件使其作为其他计算机语言比如 C++和 Python 的开发工具。Eclipse 的本身只是一个框架平台，但众多插件的支持使得 Eclipse 拥有其他功能相对固定的 IDE 软件很难具有的灵活性。许多软件开发商以 Eclipse 为框架开发自己的 IDE。选用 Eclipse 作为开发工具是因为其易用、开源、免费与跨平台性，并且支持多种语言开发，可以方便本系统的协同开发。

Python 是一种面向对象的解释性的计算机程序设计语言，也是一种功能强大而完善的通用型语言，已经具有十多年的发展历史，成熟且稳定。选用 Python 作为开发语言是因为其可移植性与可嵌入性可为协同开发带来方便。

Google App Engine 是 Google 开放的云计算开发环境，通过 Google App Engine，即使在重载和数据量极大的情况下，也可以轻松构建能安全运行的应用程序。该环境具有动态网络服务、持久存储有查询、分类和事务、自动扩展和载荷平衡等特点。此外，Google App Engine 还提供一种功能完整的本地开发环境 App Engine 软件开发套件 (SDK)，包括可以在开发者的本地计算机上模拟所有 App Engine 服务的网络服务器应用程序。该 SDK 包括 App Engine 中的所有 API 和库。该网络服务器还可以模拟安全 Sandbox 环境，包括检查是否存在禁用模块的导入以及对不允许访问的系统资源的尝试访问。Python SDK 完全使用 Python 实现，可以在装有 Python 2.5 的任何平台上运行，包括 Windows、Mac OS X 和 Linux。您可以在 Python 网站上获得适用于您的系统的 Python。该 SDK 以 Zip 文件提供，安装程序可用于 Windows 和 Mac OS X。该 SDK 还包括可将在本地

火龙果•整理
uml.org.cn

开发的应用程序上传到 App Engine 的工具。创建应用程序的代码、静态文件和配置文件后，即可运行该工具上传数据。管理控制台是基于网络的界面，用于管理在 App Engine 上运行的应用程序。开发者可以使用它创建新应用程序、配置域名、更改应用程序当前的版本、检查访问权限和错误日志以及浏览应用程序数据库。

5.2 系统开发关键思想与技术

5.2.1 原型开发模型

综合投入风险产出比，系统采用原型开发模型[33]，一方面为将来的系统功能接口变更等需求改变提供便捷，一方面快速形成原型以方便在其基础上迭代开发。

原型开发过程如下：

(1)快速分析。

在分析人员与用户密切配合下，迅速确定系统的基本需求，根据原型所要体现的特征描述基本需求以满足开发原型的需要。

(2)构造原型。

在快速分析的基础上，根据基本需求说明尽快实现一个可行的系统。这里要求具有强有力的软件工具的支持，并忽略最终系统在某些细节上的要求，如安全性、坚固性、例外处理等等，主要考虑原型系统能够充分反映所要评价的特性，而暂时删除一切次要内容。

(3)运行原型。

这是发现问题、消除误解、开发者与用户充分协调的一个步骤。

(4)评价原型。

在运行的基础上，考核评价原型的特性，分析运行效果是否满足用户的愿望，纠正过去交互中的误解与分析中的错误，增添新的要求，并满足因环境变化或用户的新想法引起的系统要求变动，提出全面的修改意见。

(5)修改。

火龙果•整理
uml.org.cn

根据评价原型的活动结果进行修改。若原型未满足需求说明的要求，说明对需求说明存在不一致的理解或实现方案不够合理，则根据明确的要求迅速修改原型。

在这里，由于系统的用户是各种数据挖掘使用者，与二次开发者，用户需求相对复杂，难免有一定变更，遵循以上原则，可以使这种变更相对简单的实现。

5.2.3 基于 WSGI 规范的开发

WSGI（Python Web Server Gateway Interface）是一个种 WEB 组件的接口规范，也是 WEB 服务器以及 WEB 服务器与应用服务器交互的一种规范。WSGI 是 Python 应用程序或框架和 Web 服务器之间的一种接口，已经被广泛接受，它已基本达成它了可移植性方面的目标。WSGI 没有官方的实现，因为 WSGI 更像一个协议. 只要遵照这些协议,WSGI 应用都可以在任何服务器上运行，反之亦然。

基于 WSGI 规范开发有众多好处，本系统基于其规范开发的目的主要在于提高系统的可用性以及实现系统的跨平台性。首先，基于 WSGI 规范实现的 Web 比单纯的桌面应用程序或者是 C/S 模式的应用程序使用起来更加简便且使用方法多元化，它具备所有 B/S 模式的优点，用户可以简单的使用瘦客户端，通过各种浏览器访问系统，用户接入网络的终端可供选择的方式也多种多样，包括各种移动设备在内的智能终端都可以满足需求。另一方面，WSGI 规范由 Python 开发语言来实现，Python 作为一种跨平台开发语言，所开发的系统可以在各种主流操作系统上运行，这样无论是将开发好的系统部署到云计算平台还是在本地开发时使用各种操作系统都非常方便。

5.3 系统开发关键步骤

5.3.1 算法模块插件系统实现

插件系统的结构的设计借鉴了 OSGi 标准，通过模块的动态加载算法模块来实现对平台的动态扩展，同时也减轻了平台的系统资源消耗，因为算法模块仅在被用户调用时加载，执行完毕后则由系统自动回收资源，在实现上这借助了虚拟机机制，因为 Python 的虚拟机可以回收引用计数为 0 的资源释放它们。

具体实现方式如下：

火龙果•整理
uml.org.cn

系统根目录下 plugins 目录为所有算法模块插件文件所在目录，用户在每次调用 GET 方法调用/algorithms REST 接口时，名为 get_algorithms 的函数被自动调用，此函数遍历所有 plugins 目录下所有以*-plugin.py 格式命名的程序文件，并动态将其访问地址通过内置方法__import__将其指针加入 module 字典数据结构中，在用户调用指定算法时，算法调用模块通过访问对应指针的地址调用具体算法模块。

以上为算法模块插件的动态加载方式，而在具体访问某个算法模块插件时，每个算法模块又至少包括以下公有函数：

Table 5.1 Public functions of every algorithms plugin module

表 5.1 算法插件模块内部公有函数

方法名	方法作用
return_properties	返回一个字典类型的数据结构至少包括： name：算法名称 parameters：算法参数 comment：算法描述
perform	执行算法插件模块，返回为算法运行结果，运行结果可以是直接在 UI 上的输出，也可以是生成一个可被可视化算法进行进一步可视化处理的数据挖掘结果数据。
load_algorithms	检测子插件并动态加载它们到子功能列表
perform_plugin	执行子插件的 perform 方法
create_db	创建测试数据用例，用于算法测试

除了这些公有函数，每个算法都会有自身的算法实现相关函数以及测试用例相关的函数，所有这些函数组成了一个算法插件模块，而开发者在开发时只需遵循上述规范实现算法模块，则此模块即可被作为插件，动态嵌入到数据挖掘平台中来。

而算法模块包括数据集清洗算法、数据挖掘算法、数据挖掘结果可视化算法等。完整的执行过程是在原始数据集存在噪声与不规则数据时用户选用数据集清洗算法将数据集进行规整或者是从某些异构、分布式或者是其他数据源数据做数据转换，生成的规则的数据集再被数据挖掘算法进行处理得到数据挖掘结果数据，此数据最终被数据挖掘可视化算法进行可视化处理呈现给用户。每个环节的调用均是通过调用之前陈述的 REST 接口，而接口与接口全部通过 XML 格式来进行信息传递，这些信息也都是通过之前定义的 XML 解析器来解析语义并最终被各算法模块使用。随着数据挖掘平台各算法模块插件的不断实现，平台所能提供给用户的功能将不断增加。

火龙果•整理
uml.org.cn

5.3.2 数据集访问模块实现

数据集访问模块的实现与算法模块插件系统的实现有相同之处，数据集访问模块也是以插件方式维护与调用，在进行数据集访问时，所有数据集访问模块文件被遍历并被索引以待被调用，所不同的是在调用数据集访问模块之前，系统首先将数据集实例文件解析为各种元数据的组合与数据库类型及其物理访问地址，再将这些数据组合成字典类型结构作为参数传递给对应数据集访问模块的 perform 方法。该模块 perform 方法通过遍历传递过来的字典数据取得包括数据集物理存储地址、数据集物理结构在内的所有信息并返回数据挖掘算法所需的数据的查询结果。例如某数据集实例文件内容为：

```
<instance name="某商品数据库">

    <template name="商品模板 1"/>

    <datasource type="MSSQL" location="mssql03.land1.fr"
    database="db183881703 table="ERPdata1" username="jijun"
    password="aWUmVJVx"/>

</instance>
```

通过 template 元素的 name 值系统获得数据集模板名称并得到如下的数据集模板定义内容：

```
<template name="商品模板 1">

    <field name="编号" type="int" column="1"/>

    <field name="名称" type="string" column="2"/>

    <field name="价格" type="int" column="3"/>

    <field name="销售量" type="int" column="4"/>

    <field name="销售时间" type="time" column="5"/>
```

火龙果•整理
uml.org.cn


```
</template>
```

这个定义内容被 XML 解析器解析为数据库中各字段的位置与名称以及类型，这里的类型是引用的元数据定义，它们最终在元数据定义中被解析为与开发语言内部数据结构对应的数据类型。

而 datasource 元素的各种属性则直接用于访问 SQL Server[34]数据库，将上述参数传递到 perform 方法后最终的数据库引擎执行过程如下：

```
connString = 'Driver={SQL
server};server=mssql03.land1.fr;database=db183881703;UID=jijun;PWD=aWUm
VJVx;'

db = mx.ODBC.Windows. DriverConnect(connString)

c = db.cursor()

c.execute('select 编号,名称,价格,销售量,销售时间 from ERPdata1')

c.close()

return c.fetchall()
```

以上是基于 SQL Server 引擎的一个实例，而除了这种数据集直接访问方式之外，当前系统可采用数据转换的方式，将其他类型的数据先转换为 BigTable 类型然后通过内置的 BigTable 访问模块进行数据访问。具体实现方式是使用 Expando 类，方法如下：

```
class dataset_instance(db.Expando):

    dataset_name = db.StringProperty(required=True)
```

Expando 类是 db 命名空间的类，这里 dataset_instance 继承它并只设定了一个必须要有的字段 dataset_name 来作为标识符索引所有 BigTable 数据集的名称，通过这个名称可以执行查询语句从数据库返回需要的数据，在使用时数据集实例的其他属性将根据传递过来的元数据定义动态生成，这样也就可以适应任意类型的数

火龙果•整理
uml.org.cn

据库结构。通过上述两种方法，系统可以访问各种数据源的数据集，而新的数据源引擎也可以作为插件不断扩充进来。

5.4 当前系统简介

通过任意浏览器访问 http://data-mining.appspot.com 后，用户可选择浏览当前所有已实现的算法与当前已注册的数据集，通过后台登陆后，用户可注册数据集，而通过使用系统提供的算法上载工具，用户可上传当前各种新的算法实现。各种开放接口作为 WEB 服务可被用于二次开发，具体接口之前已经在列表中说明这里不再赘述。

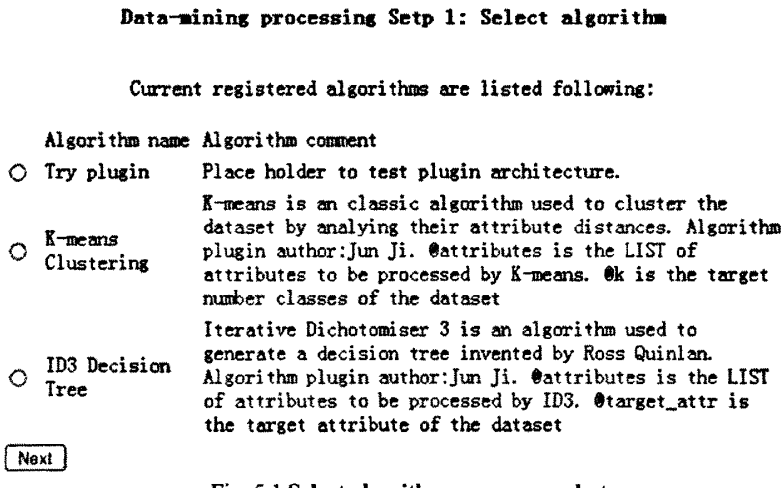

Fig. 5.1 **Select algorithm screen snapshot**

图 5.1 **算法选择界面**

火龙果•整理
uml.org.cn

Data-mining processing Setp 2: Select dataset and corresponding parameters

Current registered datasets are listed following:

Selected algorithm parameters

Algorithm name:	ID3 Decision Tree
target_attr:	education
attributes:	ALL

Dataset name	Dataset define	Dataset comment
◉ ID3_dataset1	<dataset name="ID3_dataset1"><template name="all_string"/><datasource type="BigTable" database="ID3_dataset1"/></dataset>	Test dataset for Decision Tree ID3 algorithms
○ ID3_dataset2	<dataset name="ID3_dataset2"><template name="all_string"/><datasource type="BigTable" database="ID3_dataset2"/></dataset>	Another Test dataset for Decision Tree ID3 algorithms download from http://archive.ics.uci.edu/ml/machine-learning-databases/adult/adult.data

Fig. 5.2 **Select dataset and corresponding parameters screen snapshot**

图 5.2 **数据集以及算法参数选择界面**

火龙果•整理
uml.org.cn


```
http://data-mining.appspot.com/perform
Decision Tree:
age
> 55
marital_status
single
income
high
purchase
will buy -> bachelors
low -> bachelors
married
income
high -> high school
low -> masters
< 18 -> high school
<18 -> high school
18 - 35
income
high -> bachelors
low -> high school
36 - 55
marital_status
single
income
high -> masters
low
purchase
will buy -> masters
married
income
low
purchase
won't buy -> bachelors
0:00:01.095643 second used. wrote:
target_attr=education
attributes=ALL
algorithm_name: ID3 Decision Tree
dataset_name: ID3_dataset1
dataset_instance.dataset_comment: Test dataset for Decision Tree ID3 algorithms
```

Fig. 5.3 **Result snapshot**

图 5.3 运行结果截图

如上图所示为系统截图，屏幕最上方为访问平台的网络地址，首先，用户进入的是算法选择界面，所有已实现的算法及其描述会列在其中，用户选择算法之后进入数据集选择页面，并选择对应的算法参数，而所有已注册数据集及其描述会列举在下方，用户指定某数据集为操作对象，然后点执行进入最后的运行结果页面，如图 5.3 所示，其内容为一个 ID3 决策树[35]，虽然可视化效果不理想但用户已可以理解其决策树结果，算法使用的数据集、算法的参数、执行时间等相关信息也被列出。如果用户想注册自己在网上的数据集或者导入其他类型数据库的数据，则需要登录后台页面然后编写该数据集的描述文件，由用户选择元数据列表与指定数据库引擎就可自动生成该数据集描述文件并自动注册的模块正在开发中将在不久开放。

火龙果•整理
uml.org.cn

5.5 实验结果与分析

为验证基于云计算的数据挖掘处理能力，本文使用加利福尼亚大学的机器学习数据集（http://archive.ics.uci.edu/ml/machine-learning-databases），从两种数据集分 4 组不同数量的分组分别在本地与云计算平台上运行。本地环境为 Google App Engine SDK，AMD 双核 1.6，2G 内存，其结果如下：

Table 5.2 Comparation between local host and cloud computing platform

表4.2 执行时间对比

算法名称	数据集数量	本地执行时间(s)	系统执行时间(s)
ID3 决策树	1000	15.18	3.26
	2000	57.91	7.12
	5000	238.53	21.68
	10000	1391.12	73.83
K-means 聚类	1000	2.30	0.52
	2000	4.81	1.20
	5000	13.56	7.23
	10000	32.16	13.62

可见，基于云计算的数据挖掘系统执行效率比单机系统高很多，数据量越大，这种优势越明显。

火龙果•整理
uml.org.cn

第六章　结论与展望

本文扩展了 Google App Engine 云计算开发平台，增加了数据规约功能，主要解决了异构数据访问问题与未存在数据类型访问问题，又在 Google App Engine 开发平台基础上设计与实现了一数据挖掘系统，通过实现 ID3 决策树算法与 K-means 聚类算法与相关实验验证了数据规约功能及云计算与数据挖掘相结合的高效性。

当前该平台的原型版本已实现并在不断完善中，将来的工作是把在当前系统的基础上迭代开发，持续完善系统的框架和规范，实现更多的数据挖掘算法、数据访问模块，增强用户界面的友好性，实现更多的开放接口供其他开发者使用。

火龙果•整理
uml.org.cn

参考文献

[1] J Han, M Kamber. Data mining concepts and techniques [M]. Morgan Kaufmann Publishers, 2001

[2] 邵峰晶，于忠清. 数据挖掘原理与算法[M]. 北京：中国水利水电出版社，2003

[3] Grossman R, Bailey S, Kasif S, Mon D, Ramu A, Malhi B. The preliminary design of papyrus: a system for high performance, distributed data mining over clusters, meta-clusters and super-clusters. International KDD'98 Conference, 1998, pp. 37-43

[4] Chattratichat J, Darlington J, Guo Y, Hedvall S, Köler M, Syed J. An architecture for distributed enterprise data mining. HPCN Europe 1999, Lecture Notes in Computer Science, 1593, 1999, pp. 573-582

[5] Stolfo SJ, Prodromidis AL, Tselepis S, Lee W, Fan DW, Chan PK. JAM: Java agents for meta-learning over distributed databases. International KDD'97 Conference, 1997, pp. 74-81

[6] M Cannataro, D Talia, P Trunfio. KNOWLEDGE GRID: High Performance Knowledge Discovery on the Grid. Lecture Notes In Computer Science, Vol.2242, Proceedings of the Second International Workshop on Grid Computing, 2001, pp.38-50

[7] Foster I, Kesselman C. Globus: a metacomputing infrastructure toolkit[J]. International Journal of Supercomputing Applications 11, 1997, pp. 115-128

[8] Foster I., Kesselman C. (eds.) The Grid: Blueprint for a Future Computing Inf., Morgan Kaufmann Publishers, 1999, pp.105-129

[9] M Cannataro, D Talia, P Trunfio. Distributed data mining on the grid[J]. Future Generation Computer Systems. Volume 18, Issue 8, October 2002, pp. 1101-1112

[10] M Cannataro, D Talia. KNOWLEDGE GRID An Architecture for Distributed Knowledge Discovery. CACM. Vol.46, No.1, pp. 89-93, January 2003

[11] 魏定国,彭宏.基于知识网格的数据挖掘[J].计算机科学. Vol.33, No.6, 2006

[12] A Weiss. Computing in the Clouds. netWorker, 11(4):16-25, Dec. 2007

[13] R Buyya, CS Yeo, S Venugopal. Market-Oriented Cloud Computing:Vision, Hype, and Reality for Delivering IT Services as Computing Utilities. Proceedings of the 2008 10th IEEE International Conference on High Performance Computing and Communications. Vol.00, pp. 5-13, 2008.

[14] H Attiya, J Welch. Distributed computing: fundamentals, simulations, and advanced topics[M]. Wiley-IEEE, 2004

火龙果•整理
uml.org.cn

[15] V Kumar. Introduction to parallel computing[M]. Addison-Wesley Longman Publishing Co. Boston, MA. 2002

[16] F Berman, G Fox, AJG Hey. Grid computing: making the global infrastructure a reality[M]. John Wiley and Sons. 2003

[17] P Barham, B Dragovic, K Fraser, S Hand, T Harris. Xen and the art of virtualization. ACM SIGOPS Operating Systems Review, Vol.37, Issue 5, pp.164-177. 2003

[18] CS Yeo, MD de Assuncao, J Yu, A Sulistio. tility Computing and Global Grids. Grid Computing and Distributed Systems Laboratory, The University of Melbourne, Australia, April 13, 2006

[19] D Patterson, A Brown, P Broadwell, G Candea. Recovery-oriented computing (ROC): Motivation, definition, techniques, and case studies. CSD-02-1175. March 15, 2002

[20] S Gilbert, N Lynch. Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services. ACM SIGACT News. Vol.33, Issue 2, pp.51-59. 2002

[21] Google App Engine, http://appengine.google.com [10 May 2009]

[22] Ming-Feng Chang, Min-Hsiung Hsu, Kuei-Hui Chen. An Introduction to Cloud Computing Service Platform-Google App Engine. Computer and communication. Vol.126, pp 24-33,2008.12

[23] Amazon Elastic Compute Cloud (EC2), http://www.amazon.com/ec2/ [10 May 2009]

[24] F Chang, J Dean, S Ghemawat, WC Hsieh, DA Wallach, etc. Bigtable: A Distributed Storage System for Structured Data. Proceedings of the 7th Symposium on Operating Systems Design and Implementation (OSDI'06). 2006. pp. 205 - 218

[25] Amazon Simple Storage Service, http://aws.amazon.com/s3/ [10 May 2009]

[26] 俞斌，熊齐邦.基于XML的网络配置管理的研究与实现方案[J]，计算机技术与发展,2007

[27] L Richardson, S Ruby. Restful web services[M]. O'Reilly. 2007

[28] GA Jan S Rellermeyer. Services everywhere: Osgi in distributed environments. In EclipseCon 2007, Santa Clara, CA, 2007

[29] RT Fielding, RN Taylor. Principled design of the modern Web architecture. ACM Transactions on Internet Technology, 2002

[30] Eclipse, http://www.eclipse.org/ [10 May 2009]

[31] Python, http://www.python.org/ [10 May 2009]

[32] Google App Engine SDK, http:// code.google.com/appengine/ [10 May 2009]

[33] JD Naumann, AM Jenkins. Prototyping: the new paradigm for systems development[M]. MIS Quarterly, 1982

火龙果•整理
uml.org.cn

[34] 张莉. SQL Server数据库原理及应用教程[M]. 北京：清华大学出版社，2002

[35] JR Quinlan. Induction of decision trees[M]. Machine learning, Springer, 1986

火龙果•整理
uml.org.cn

攻读学位期间的研究成果

[1] Jun Ji, FengJing Shao, RenCheng Sun, Neng Zhang and Guanfeng Liu, A TSVM based semi-supervised approach to SAR Image Segmentation, IITA-GRS 2008, 2008.12.24, index by EI.

火龙果•整理
uml.org.cn

致谢

在论文完成之际，首先感谢我的导师邵峰晶教授在毕业设计期间对我的悉心指导，是她凭借自己在科研领域敏锐的洞察力和先进的研究理念帮我加深了毕业设计的理论深度，挖掘出其中的学术价值。在三年的研究生学习过程中，邵老师一丝不苟的工作态度和严谨求实的治学风范给我留下了深刻的印象，在引领我学习和科研的同时，她也教会我们很多为人治学之道。我的毕业设计正是在邵老师的指导下才得以顺利完成。在此，向邵峰晶教授表示诚挚的谢意，谢谢老师对我的谆谆教导。

其次衷心感谢孙仁诚老师，在他的繁忙教学与博士学习之余，仍然细心指导我的课题研究进展与论文写作，使我能顺利完成硕士阶段的学习。在生活和做人上也给予我们这些师弟师妹们很多帮助和启示，指导和鼓励我们不断向前奋进。

此外，感谢身边所有关心我的同学和朋友，是他们在科研道路上给我帮助，在人生旅途中给我勇气，使我在青岛大学度过了精彩的三年，找到了积极的态度，经历了很多也收获了很多。

最后，感谢我的父母家人，他们对我的关爱和支持是我不断进步的动力。

火龙果•整理
uml.org.cn

一种基于云计算的数据挖掘平台架构设计与实现

作者： 纪俊
学位授予单位： 青岛大学