班级 3-0891

学号 03089013

西安电子科技大学

本科毕业设计论文

题　　目 虚拟计算平台资源调度系统的设计与实现

学　　院 计算机学院

专　　业 计算机科学与技术

学生姓名 王力

导师姓名 马建峰

摘　要

本文提出了一个基于贝叶斯分类的多算法混合虚拟资源调度系统（Bayes-classifier based Mixed-algorithm Virtual-resources Scheduling System，简称$BMVS^2$），设计了针对虚拟资源请求的分类划分机制，实现了虚拟计算平台下任务资源请求的最佳组态。

硬件技术的发展非常迅速，相比不断膨胀的需求，资源却是有限的，虚拟资源调度系统追求的目的就是寻找一种分配策略从而最大化的满足各个请求的需求，并使得该分配方案获得时空上的最优。在$BMVS^2$中，设计了一种基于数据挖掘分类技术的预测机制，以实现对任务请求的分类划分，获得更合理的组合方式。采用了模式完整下的一种最优分类器－贝叶斯分类器,更加快速地做出决策。设计了一个评价因子，对一次决策过程的结果进行比较，反馈给预测和分类模块，用于信息的存储交换，以便下次做出更好的决策。

$BMVS^2$解决了目前系统不具有学习和自适应能力的缺点，使系统同时具有描述状态信息和预测的能力，实现了资源请求的合理调度。测试结果表明，该方案可为虚拟计算平台提供更加灵活的决策支持。

关键词：虚拟化，资源分配，贝叶斯分类

ABSTRACT

A Bayes-classifier based Mixed-algorithm Virtual-resources Scheduling System ,namely $BMVS^2$,is proposed in this paper,which designed a classfication mechanism for virtual resources request,aiming to give a best configuration of task's resources request under the virtual computing platform.

Although the hardware technology develops very quickly,compared to growing demand, resources are still limited, therefore the pursuit of Virtual Resources Scheduling System is to find a scheduling policy that maximize the meet to the needs of individual requests, and so that the assignments given are spatial and temporal optimal. In the $BMVS^2$ system, designing a prediction mechanism based on data mining classification technology to achieve division of category to tasks' requests,and get a better and more reasonable combination. In order to make a fast decision,an optimal classifier under the circumstance of full pattern-Bayes classifier is adopted.For the adaptive characteristics, designing an evaluation factor, comparing the results to a decision-making process, thus the feedback will send to prediction and classification modules for exchanging storage-information, so that a better decision will be made in the next time.

$BMVS^2$ addresses the current systems' disadvantage of lacking of learning and adapting, making the system has the ability to describe the status information and to make a prediction,simultaneously,a reasonable scheduling of resources requests thereupon is achieved in a better way.Testing result show that virtual computing platforms are supported under the programme to make more flexible decisions.

Keywords: virtualization,resources allocation,Bayes classification

目　录

第一章　引言

随着计算机硬件技术的不断发展，以及用户应用需求的多样化，迫使应用提供商给出越来越多样化的解决方案。而这样会造成资源的浪费以及适应不同平台而造成的开销。而虚拟化技术的诞生恰好是为了解决这些问题的，而且虚拟化技术正在受到越来越多的关注，包括国际著名的IT企业Citrix、VMware，Google、Microsoft、Amazon等互联网公司也不断在这个领域推陈出新；IBM、DELL等传统IT 厂商也被迫转型，不断调整公司的战略计划、商业模式；AT&T、Verison等电信运营商作为传统的IDC和电信服务提供商业看到了机遇，一方面不断提升基础硬件，一方面不断拓展云计算虚拟化业务。可以预见，虚拟化技术将是继互联网之后，又一个给整个信息产业但来突破性发展的技术革新。

1.1　研究背景

虚拟化的概念最早由计算机科学家Christopher Strachey在一篇名为《Time Sharing in Large Fast Computers》[1] 的学术报告中提出。然后在20世纪60年代，为了共享昂贵的大型机资源，IBM公司发明了一种操作系统的虚拟机技术，并将该技术运用于VM/370系统中。而如今，硬件技术的飞速发展已经为虚拟化提供了更为有利的支持，它是构筑在互联网基础上的一系列技术，它将高速互联网、高性能计算机、大型数据库、移动终端设备等融为一体。而在虚拟化等基础上衍生出来的云计算，则更加让人们透明的使用计算资源、存储、数据等。

虚拟化所能提供的强大功能，必然在理论研究和应用实现上带来前所未有的挑战。虽然如此，在需求驱使的推动下，它在理论和实践中不断取得发展。包括中国在内，全世界很多国家都在对虚拟化、云计算投入巨大的人力财力，也产生了很多具有里程碑意义的研究结果。

互联网公司由于自身业务需求构建了基于分布式技术的可支撑海量数据信息处理和存储的统一IT基础设施；IT厂商考虑的则是如何保护企业已有的IT投资，主推虚拟化技术作为IT资源的整合手段，以帮助企业更平滑地过渡到云计算。不过，不论是分布式技术还是虚拟化技术，都只是云计算实现的基本技术手段，真正使云计算”落地“的核心技术还是如何实现虚拟化资源和分布式集群规模化及统一管理，以形成具备良好的弹性和可扩展性的资源池（计算和存储云），同时将应用基于互联网交付给用户（交付云）。这就像程控交换技术和软交换技术，它们都是实现电话业务的技术，在设备技术成熟后，更重要的是要通过部署和组网形成一张规模较大的交换网，并实现全网统一管理和运营，才能真正提供优质的语音业务[2]。因此，在虚拟化、分布式技术基本成熟的环境下，虚拟计算平台资源调度系统以及其核心－资源分配问题无疑是最重要，最具挑战性的研究方向之一。

目前，云计算是网络计算的最新形态，是网格计算、并行计算、分布式计算的发展结果，同时也是一种崭新的商业模式，备受业界关注，尤其是IaaS[2]（Infrastructure as a Service，即基础设施即服务）把业务部署在由大量服务器、存储设备、网络设备构建的资源池上。由于用户提交的请求有多个，并且需求不一，因此需要对有限的资源进行合理分配，才能满足应用的需求，同时达到节约资源的目的，这也是设计资源调度系统的初衷与目的。

1.2 研究目的及意义

云计算使计算分布在大量的分布式计算机上，对于用户来说，并不关心这些隐藏在“云”后面的架构，它就像一朵云一样。这样，企业就能将资源切换到需要的应用上，根据需求访问计算机和存储系统，而无需关心这些繁琐的资源管理，大大降低了企业信息化的复杂度。

在云计算平台中，资源以服务的形式按需分配，并且要保证用户满足服务等级协议（Service Level Agreement，简称SLA）。然而，由于资源是共享的，并且用户的需求有着动态的不均衡和平台差异，如果不能合理的分配这些有限的资源，势必会造成资源的浪费。除此之外，云计算平台同样需要满足动态负载平衡，以避免任务的频繁迁移带来的开销的同时尽可能高的提高资源的利用率。因此，从宏观上讲，如何管理资源以最大化的满足用户的需求成了一个有待解决的问题。

虚拟化技术最早是为了大型机系统提供虚拟化而服务的。随着信息技术的发展，虚拟化再次成为提高硬件利用率，整合企业计算资源的重要解决方案。它提供了一个高效的解决方案来管理云计算平台上的资源。从提交服务到虚拟机，映射到每个物理主机，再从解决平台和用户需求的不均衡到满足SLA，虚拟化技术都很好的给出了解决方案。此外，虚拟化技术还能够依据不同虚拟机（VM）和物理资源之间的关系改变这种映射，力求在整个系统内达到一种负载均衡。因此，虚拟化技术越来越多的使用在了云计算之中。但是，正是由于平台上资源的及不均衡，虚拟机必须动态地适应这些问题，以获得更好的性能。所以，细分下，如何分配虚拟机资源以实现高利用率是本文的研究重点。

在资源分配算法的研究过程中，产生了一些优秀的经典资源分配算法，如贪心（Greedy）、遗传（Genetic）、松弛线性规划（Relaxed LP）、装填算法（Vector Packing）[3]。但是这些算法也存在不足之处，并且大部分是针对CPU计算资源的最大化满足，本文由于设计的是一个系统，所以将不同的算法实现为不同的模块，根据预测机制自动调用不同的模块，实现资源的最大化利用，并在时空上达到最优。

1.3 国内外研究现状

本节就算法策略模块介绍国内外研究的现状，虚拟资源分配策略的研究一直是一个热点问题，尤其是近年来的一些发展。

在[4]中，圣何塞州立大学的项目提出了一种遗传算法，利用生物遗传理论，结合当前的计算机学科。由于考虑了历史信息，并结合当前的状态，可以提前计算部署后可能产生的影响，挑选一个影响最小的分配方案，以达到保证负载平衡的同时提高利用率，但是遗传算法有着致命的缺点，便是速度慢，有时也会有无解的情况产生。

在[5]中，研究员提出了基于概率分析的装填算法，从概率论角度研究了该模型。概率模型既具有描述性质，也具有预测性质。

在[6]中，美国劳伦斯国家实验室的研究者提出了一种原地进行的Θ(n)的二维装填算法，此算法考虑两个方面，即文件的负载和大小。并对此算法进行了严格的数学证明以及算法执行的伪代码部分。

在Xen发展的过程中，曾经出现了SEDF、BVT和Credit算法。SEDF简单的基于截止时间进行调度，该算法是一种动态优先级调整的算法，VCPU的优先级随着其绝对截止时间的变化而变化。但是SEDF算法在负载重的情况下表现出很多的时间错误，也不能对CPU以外的资源进行操作。BVT算法是一种公平性优先的调度算法，它将时间分为虚拟时间和实际时间，运行的时候用虚拟时间监控进程的执行时间，在一定的范围内，允许操作可以“借”未来的时间先过来运行。BVT算法优点在于可以公平均匀地将物理时间分配给各个虚拟机，满足I/O密集和低延迟的要求。但是BVT不支持NWC（non-working-conservation）模式，即每次运行时获得整个CPU；每个虚拟机只能借用自己的时间片，而不会剥夺其它虚拟机的时间片。Credit算法是Xen 3.0以来的默认调度算法，它是一种按比例公平共享的非抢占式调度算法，它可以全局管理多个物理CPU，从而将CPU公平高效地分配给各个虚拟CPU，它也可以很好的支持NWC模式。但是Credit算法的缺点是不能保证实时性，在I/O等对时延比较敏感的应用中，影响尤为重要。

之前的研究基本上都是局限于尽可能高的提高资源的利用率方面，而[7]中的DRF算法是MESOS平台上的调度策略，该算法是一种基于经济学博弈理论的公平分配策略，并在Facebook实际测试了2个月获得了预期的效果。该策略不同于以往满足某些资源的高需求的策略，它能刺激用户共享他们的资源（Sharing Incentive，即共享刺激），保证自己所获得的资源永远都是自我满足的（Envy-freeness性质）。该策略也是具有防策略（Strategy-proofness）性质的，即不能通过欺骗获得更多的资源。最后，DRF 满足帕累托最优（Pareto-efficiency）原则。该算法一经发表，就获得了state of the art 的称赞，并吸引了很多研究者的兴趣。但是此方法的初期版本也有缺陷，不能获得很高的资源利用率，甚至有时利用率不及FIFO的性能。

在DRF算法发布不久，[8]在第32届分布式计算系统国际会议（The 32nd International Conference on Distributed Computing Systems）上发表了一篇论文，提出了之前的算法仅考虑单资源的情况，却忽略了资源是有比例关系的，考虑了资源之间的依赖关系，提出了全新的启发式算法，并通过仿真验证了性能和效率的正确性，同时比较了DRF 算法，指出了DRF的缺陷。

此外，还有很多开源的云系统供研究者作为平台开发自己的云计算项目。比

如Eucalyptus[9]，Open Nebula[10]，Open Stack[11]等等。Eucalyptus使用贪心和循环算法，Open Nebula 使用队列、高级保留以及预取技术，而Open Stack中使用的是Chance Scheduler，即随机算法。开源平台的好处就是研究者可以根据自己的需要替换默认的调度策略。

作为虚拟化技术的领导厂商，VMware[12]在虚拟化方面有着雄厚的技术，VMware 分布式资源调度程序（Distributed Resource Scheduler，简称DRS）跨聚合到逻辑资源池中的硬件资源集合来动态地分配和平衡计算容量。

- 使资源优先于最重要的应用程序，以便让资源与业务目标相协调
- 自动、不间断地优化硬件利用率，以响应不断变化的情况
- 为业务部门提供专用的（虚拟）基础结构，同时让IT部门能够集中、全面地控制硬件
- 执行零停机服务器维护。

VMware DRS跨资源池不间断地监控利用率，并在多个虚拟机之间智能地分配可用资源。VMware DRS允许用户确定以下方面的规则和策略：决定虚拟机共享资源的方式与在多个虚拟机之间排定这些资源的优先级的方式。当虚拟机遇到负载增大时，VMware DRS将首先根据既定的资源分配规则和策略评估其优先级，如果合理，则分配更多资源。资源是通过以下两种方式分配给虚拟机的：将虚拟机迁移到具有更多可用资源的另一台服务器上；或者通过将其他虚拟机迁移到别的服务器上而在此服务器上为该虚拟机营造更大的“空间”。通过VMware VMtion将虚拟机实时迁移到不同的物理服务器是以对最终用户完全透明的方式完成的。

实际上，现有的研究不仅仅是上面所罗列的，国内外的研究者都在这个领域贡献着自己的智慧，提出更加高效的算法，开发更加有效的系统。

1.4 研究内容与组织框架

1.4.1 研究内容

总结毕设期间的研究，主要集中于以下几个方面：

1. 研究了虚拟化技术，分布式集群计算，借助Xen架构，分析了其中的原理与机制
2. 研究了虚拟资源调度系统的具体设计实现以及具体的应用
3. 通过阅读大量论文，研究了各种资源分配算法，分析了各个算法的适应场景以及优缺点
4. 通过测试，评估了本文提出的系统具的可行性和合理性

1.4.2　组织框架

本文的组织如下：

第一章为引言部分，是全文的铺垫部分，介绍了本文研究的背景、目的以及研究的意义所在，并分析了国内外此领域的研究现状

第二章首先介绍了本文研究背景中的虚拟化技术的发展，在第二节中对虚拟资源调度系统的发展和分类做了介绍和分析，然后介绍和分析了虚拟资源调度系统中的关键技术。

第三章提出了虚拟资源调度系统的架构设计，并对该系统进行了理论上的分析。

第四章通过测试验证了该系统，并对核心的调度算法进行了可行性与正确性的验证，总结了具体的应用环境。

第五章是本论文的结束语部分，对全文做一总结，同时针对设计的系统提出了一些改进的愿望，并对下一步要进行的工作做了简要的介绍。

第二章　虚拟化技术与资源调度系统

2.1　虚拟化技术

在计算机科学中，虚拟化（Virtualization）是一个表现逻辑群组或电脑资源的子集的进程，用户可以用比原本的组态更好的方式来存取这些进程。这些资源的新虚拟部份是不受现有资源的架设方式，地域或物理组态所限制。一般所指的虚拟化资源包括计算能力和资料储存[13]。

2.1.1　虚拟化技术的分类

按虚拟化的侧重，虚拟化技术可具体分为以下[13]：

- 虚拟机（Virtual machine或VM)，可以像真实机器一样运行程序的计算机的软件实现
  - 平台虚拟化，将操作系统和硬件平台资源分割开
    - 完全虚拟化，敏感指令在操作系统和硬件之间被捕捉处理，客户操作系统无需修改，所有软件都能在虚拟机中运行，例如IBM CP/CMS，VirtualBox，VMware Workstation
    - 硬件辅助虚拟化，利用硬件（主要是CPU）辅助处理敏感指令以实现完全虚拟化的功能，客户操作系统无需修改，例如VMware Workstation，Xen，KVM
    - 部分虚拟化，针对部分应用程序进行虚拟，而不是整个操作系统
    - 准虚拟化/超虚拟化（paravirtualization），为应用程序提供与底层硬件相似但不相同的软件接口，客户操作系统需要进行修改，例如早期的Xen
    - 操作系统级虚拟化，使操作系统内核支持多用户空间实体，例如Parallels Virtuozzo Containers，chroot，Solaris上的Zone
  - 应用程序虚拟化，在操作系统和应用程序间建立虚拟环境
    - 便携式应用程序，允许程序在便携式设备中运行而不用在操作系统中安装
    - 跨平台虚拟化，允许针对特定CPU或者操作系统的软件不做修改就能运行在其他平台上，例如Wine
    - 虚拟设备，运行于虚拟化平台之上，面向应用的虚拟机映像
    - 模拟器
- 虚拟内存，将不相邻的内存区，甚至硬盘空间虚拟成统一连续的内存地址

- 存储虚拟化，将实体存储空间（如硬盘）分隔成不同的逻辑存储空间
- 网络虚拟化，将不同网络的硬件和软件资源结合成一个虚拟的整体
  - 虚拟专用网络（VPN），在大型网络（通常是Internet）中的不同计算机（节点）通过加密连接而组成的虚拟网络，具有类似局域网的功能
  - 存储器虚拟化，将网络系统中的随机存储器聚合起来，形成统一的虚拟内存池
- 桌面虚拟化，在本地计算机显示和操作远程计算机桌面，在远程计算机执行程序和储存信息

2.1.2 虚拟化技术的发展

现在，由于虚拟化技术在x86服务器上的迅速普及已经引发了虚拟化技术的热潮，尤其是英特尔的VT^{TM}和AMD的AMD-V^{TM}技术对x86架构硬件虚拟化的支持。虚拟化技术最初只是应用在大型主机上，大型机上的虚拟分区技术最早可以追溯到上世纪六、七十年代。早在上世纪60 年代，IBM 公司就发明了一种操作系统虚拟机技术，允许用户在一台主机上运行多个操作系统，让用户尽可能地充分利用昂贵的大型机资源。

最早使用虚拟化技术的是IBM 7044 计算机。IBM之后，在上世纪60年代还开发了型号为Model 67的System/360主机。Model 67 主机通过虚拟机监控器(Virtual Machine Monitor，简称VMM)虚拟所有的硬件接口。1965年IBM公司的“M44/44X”计算机项目，定义了虚拟内存管理机制，用户程序可以运行在虚拟的内存中，对于用户来说，这些虚拟内存就好像一个个“虚拟机”，为多个用户的程序提供了独立的计算环境。

IBM提出的虚拟机技术，使一批新产品涌现了出来，比如：IBM360/40，IBM360/67，以及VM/370，这些机器在当时都具有虚拟机功能，通过VMM技术在物理硬件之上生成了很多可以运行独立操作系统软件的虚拟机实例。

由于虚拟化技术技术在商业应用上的优势，RISC服务器与小型机成为了虚拟化技术第二代受益者。1999年，IBM公司在AS/400 上提出了上“逻辑分区(LPAR)”技术和新的高可用性集群解决方案。在POWER管理程序上运行的AS/400 LPAR令单台服务器工作起来如同12个独立的服务器。而在2002年，IBM还更进一步，其AIX5L v5.2还首次包括了IBM实现的动态逻辑分区（DLPAR）。DLPAR允许在无需重启系统的情况下，将包括处理器、内存和其它组件在内的系统资源分配给独立的分区。这种在不中断运行的情况下进行资源分配的能力不仅令系统管理变得更加轻松，而且因为能够更好地使用资源而帮助降低总拥有成本。[14]

不过，尽管惠普、Sun公司也跟随IBM在自己的RISC服务器上提供了虚拟化技术，但由于真正使用大型机和小型机的用户还是少数，加上各家产品和技术之

间并不兼容，虚拟化技术仍旧不太被公众所关注。而现在，虚拟化技术的发展已经惠及到了x86架构。

此前，虚拟化技术在x86架构上进展缓慢的主要原因是x86架构本身不适合进行虚拟化，不过这个障碍已经由英特尔、AMD修改，x86处理器的指令集得到解决;还有一个原因是x86处理器的性能不足，这一原因也由于x86处理器在性能上的飞速提高得到了解决。由于x86架构的广泛普及，x86架构上的虚拟化技术也得到了比以前更大的关注。

随着x86平台上虚拟化技术的实现，首次向人们展示了虚拟化应用的广阔前景，因为x86平台可以提供便宜的、高性能和高可靠的服务器。更重要的是，一些用户已经开始配置虚拟化的生产环境，他们需要得到新的管理工具，从而随着虚拟化技术的发展而得到更大的收益。

不过，与已经有多年历史的UNIX服务器、大型主机上的虚拟化技术相比，x86服务器上的虚拟化仍旧处于早期阶段-根据英特尔的蓝图，在处理器当中集成硬件辅助虚拟化指令只是IA平台上的第一步，而在第二步则要实现I/O方面的虚拟化，直到最后实现整个IA平台的虚拟化。也就是说，目前的x86平台上，仅仅能够实现在处理器级别的虚拟化，在I/O以及其他方面的虚拟化还需要进一步的发展。不仅如此，x86架构上的虚拟化技术还无法完美实现虚拟分区之间动态迁移，而这些在UNIX平台、大型主机上早已不是问题。

自2006年以来，从处理器层面的AMD和Intel到操作系统层面的微软的加入，从数量众多的第三方软件厂商的涌现到服务器系统厂商的高调，我们看到一个趋于完整的服务器虚拟化的产业生态系统正在逐渐形成。这也使得在过去的一两年时间里，虚拟化开始成为广受关注的热点话题。整体看来，随着计算机新技术的飞速发展，虚拟化的前景和若干年前相比几乎彻底改变了，新的虚拟化平台前景十分乐观。

随着x86架构虚拟化的普及，高效的资源调度管理平台，整体解决方案，行之有效的资源调度、分配策略也成了研究的重点，也成为了该领域内比较重要的课题。

2.2　资源调度系统

2.2.1　资源调度系统的分类

资源调度系统按照不同的分类标准，可以很多种不同的调度系统。调度系统在具体的应用背景下可能需要特定的调度机制支持，因此有以下几类：

根据调度策略是脱机还是联机实现，调度系统可分为：

- ➤ 静态调度-调度策略在一次运行前已经确定
- ➤ 动态调度-调度策略在运行过程种实施，可根据系统的运行状态调整

根据系统分类，可分为：

- 单处理器调度-典型的应用如操作系统
- 集中式多处理器调度或者集中式仲裁
- 分布式调度-分布式管理系统

按照任务的可否抢占需求，系统可分为：

- 支持抢占的调度
- 不可抢占的调度

按照任务对时延的要求，可分为：

- 强实时调度
- 弱实时调度
- 非实时调度

2.2.2 典型的虚拟机系统

1. **Xen**

在工业界，Xen[15]已经渐渐成为一个实际存在的工业标准，并且已经有成功的商业案例，比如Citrix，拥有着包括Citrix XenServer，Citrix NetScaler，Citrix XenApp，Citrix XenDesktop等产品。Xen 是剑桥大学计算机系开发的一个开源虚拟机监控器项目，使用Xen 可以使单一物理主机上同时运行多个操作系统，并且具有很好的隔离性。Xen 可以对启动后的采用标准虚拟化技术的虚拟机进行资源的动态调度，而且可以给Guest OS 分配大于物理CPU 个数的VCPU。在Xen 中，不但可以映射不同的VCPU 到物理CPU，也可以调整CPU 运行队列来更改优先级。Xen的架构如图2.1，在虚拟层的基础上，Xen的Driver domain 相当于资源决策支持系统，它负责在监控器提供数据的基础上分配这些虚拟资源。

2. **VMware**

另外一个流行的虚拟化产品是VMware Infrastructure[16]，通过其Virtual Center可以提供资源管理功能，将处理器和内存资源分配给运行在服务器上的多个虚拟机，为CPU、内存、磁盘和网络带宽确定最小、最大和按比例共享的资源共享，在虚拟机运行的同时支持修改分配，使应用程序能够动态获得更多资源以适应应用峰值性能的需要，从而形成一个具有内置负载平衡能力的自我管理、优化和有效的虚拟机资源调度管理环境。并且，其中的VMware DRS可以跨资源池不间断地监控利用率，并根据反映业务需求和变化的优先级的预定义规则，在多个虚拟机之间智能分配资源。当某个虚拟机负载增加时，VMware DRS通过改变虚拟机在资源池中所在的物理主机来获得更多的资源，例如将该虚拟机迁移到比较空闲的一台机器上。VMware DRS的动态虚拟机迁移机制主要面对的是环境中主机

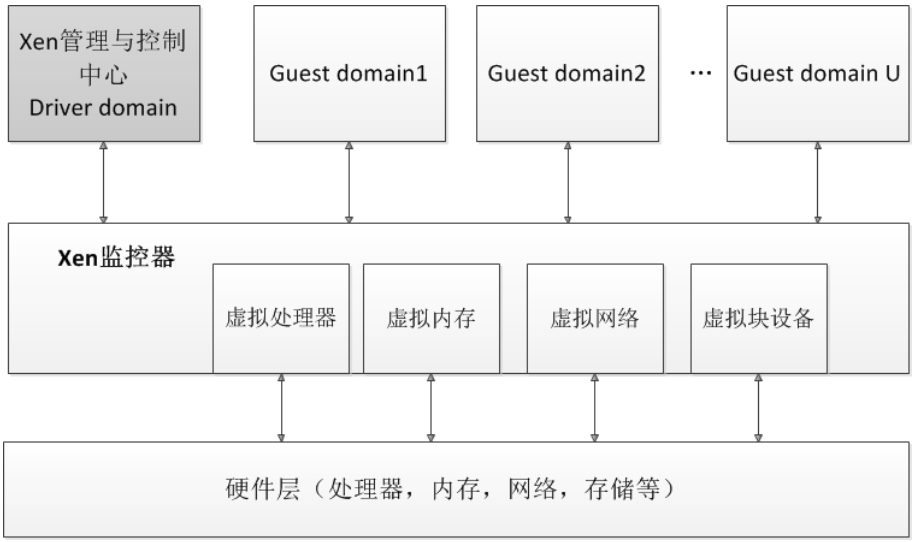

图 2.1 Xen的架构

之间的动态资源分配，而不是单机多CPU多核架构（SMP和NUMA，Non Uniform Memory Access非一致访问分布共享存储技术）中的动态资源分配，也没有提供对高性能计算任务的优化，还有VMware目前只能在自家的vCenter使用。VMware DRS的动态迁移如图2.2 所示。

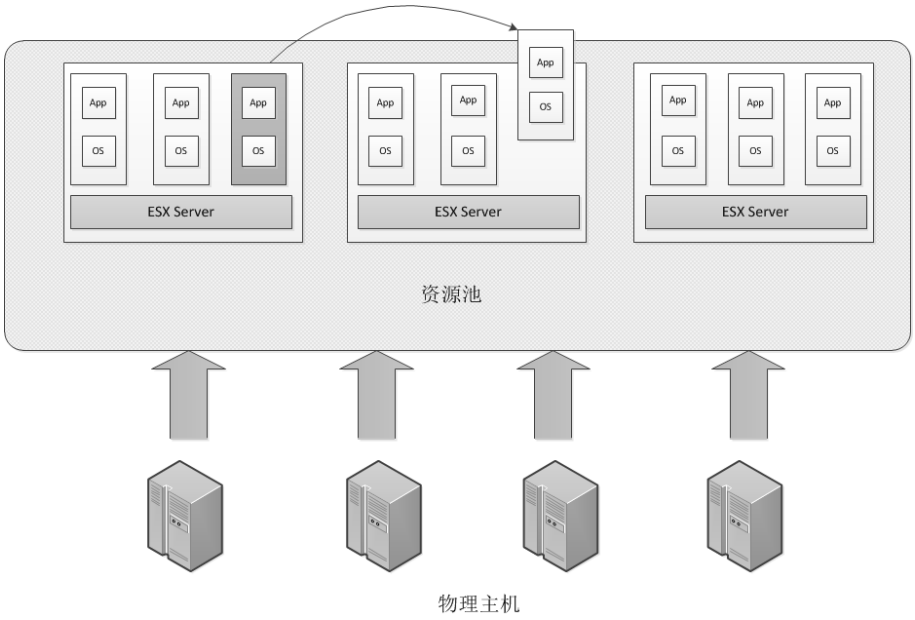

图 2.2 VMware DRS在虚拟机之前平衡资源

3. **Libvirt**

Libvirt[17]库是一种实现Linux 虚拟化功能的Linux API，它支持各种虚拟机监控程序，包括Xen 和KVM，以及QEMU 和用于其他操作系统的一些虚拟产品。前两款产品都是实际完整的架构，而讲到向外扩展计算（比如云计算），Libvirt可能是未曾听过的一个最重要的库之一。Libvirt 提供一种虚拟机监控程序不可知的API来安全管理运行于主机上的Guest OS。Libvirt 本身构建于一种抽象的概念之

上，它为受支持的虚拟机监控器实现常用功能－比如基于监控数据作出决策，提供通用的API支持。如图2.3 所示的是Libvirt的基本架构，表示了Libvirt的比较和用例模型，其中右图中深色标记的Libvirt为其在虚拟环境中的位置。

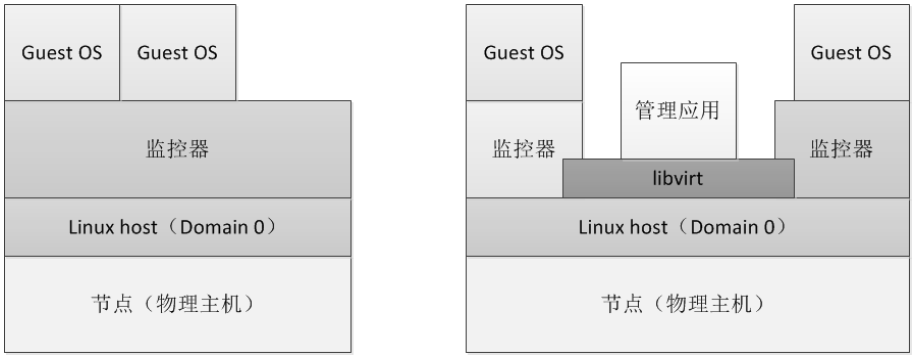

图 2.3 Libvirt的比较和用例模型

使用Libvirt，有两种不同的控制方式。第一种如图2.3右所示，其中管理应用程序和域位于同一节点上。管理应用程序通过Libvirt 工作，以控制本地域。当管理应用程序和域位于不同节点上时，便产生了另一种控制方式（如图2.4）。因此需要进行远程通信。该模式使用一种运行于远程节点上、名为libvirtd 的特殊守护进程。当在新节点上安装Libvirt 时该程序会自动启动，且可自动确定本地虚拟机监控程序并为其安装驱动程序。该管理应用程序通过一种通用协议从本地libvirt连接到远程libvirtd。据此，可以开发出管理分布式虚拟资源的调度系统。

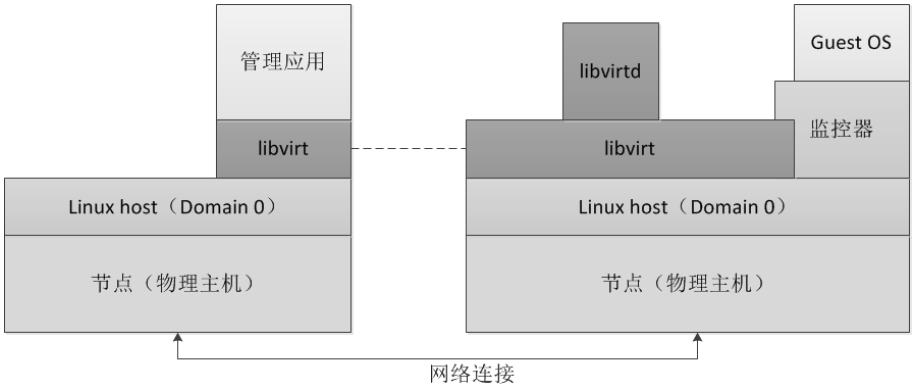

图 2.4 使用Libvirtd控制远程虚拟机监控程序

2.3 关键技术及分析

2.3.1 资源监控

虚拟资源的调度系统的基础是监控系统，它在整个系统中起着重要的作用，全面、合理、迅速的信息采集能够使决策模块全方面了解虚拟机的情况以及主机的性能，从而帮助上层决策系统做出相应的调度决策。虚拟机的信息监控模块需要考虑以下问题：

1. 信息采集的全面性

   能够监控多个虚拟机的CPU、内存利用率、虚拟网络负载和虚拟块设备负载等等。为了较准确判断虚拟机资源消耗和利用情况，需要能够实时监控虚拟机每个VCPU的资源利用率和运行状态信息。

2. 实时性和准确性

   数据更新要及时，数据信息一定要准确。

3. 实用性

   不禁能够监控虚拟机的资源消耗，还应该监控硬件资源的运行以及资源消耗情况

4. 资源开销

   信息监控的开销要小，不能影响系统的性能，要保证信息监测的执行不会抢占虚拟机的有限资源

2.3.2 资源调度策略

虚拟资源总是有限的，无限的资源仅仅是一种理想态，因此我们需要调度机制。资源调度机制是一个调度系统的核心，它负责从监控的数据中分析，然后基于预定的决策给出解决方案，决定这些虚拟资源的去向。一个调度系统的优秀与否与调度系统的是否高效有着直接的关系。

资源调度总是希望达到以下技术要求：

- 资源利用率

  最大化的利用资源是几乎每个调度系统首要考虑的问题。虚拟资源是有限的，资源请求又是不均衡的，如何合理的调度，使这些有限的资源以一个合理的组态最大化的利用是重点。

- 负载均衡

  负载均衡是希望实现集群中每个机器的负载在一个均衡的状态，平均分配客户的请求到各个虚拟机，以此解决快速获取资源，解决大量并发访问问题。

- 公平

  如何在满足自己需求的情况下最大化的获得公平性是DRF算法提出的观点。公平的情况下可以获得更好的均衡特性和防竞争性质。

- 实时动态调度

  虚拟机在运行的过程中，希望调度系统能够动态地在线调度任务，给这些任务分配资源，而不是静态地分配。

2.4 本章小结

本章详细介绍了虚拟化技术的分类与发展情况，对现有资源调度系统进行了归类，然后依据架构图分析了Xen的调度模块、VMware DRS 调度机制，并详细介绍了Libvirt。最后，对资源调度系统的关键技术进行了分析。

第三章　基于贝叶斯分类的多算法混合虚拟资源调度系统

3.1　$BMVS^2$的设计

3.1.1　系统结构设计

在科学技术的发展史上，系统工程与其他技术科学一样，它的产生和发展主要由需求所驱动。一个系统都有着它特定的架构，在该架构下，根据预定义的规则执行着任务流。

系统环境的复杂性是以其变化的不确定性和难以发现为主的，虚拟环境下，资源池有着从物理主机虚拟出来的庞大的资源，如虚拟CPU、虚拟内存等，而用户的需求也是变化不定的。在传统的虚拟机资源调度系统里，往往只有一种调度算法，或者一个特定时期的版本只有一种当时比较合理的算法，使用这种算法往往限制很多条件。而对于目前趋于更加复杂的管理与决策系统，必须把这些动态变化的因素作为一个整体来进行分析。

对于管理的任务或者调度的对象，以往都是着眼于用数学语言进行描述，而这种描述有时是不准确的。实际上，一个任务行为的确定，需要结合以往的历史信息，进行分类，然后选择一种当前最佳的算法。

面对复杂的环境和多变的请求，用传统的理论和方法愈加显得捉襟见肘，这是因为：

1. 传统的系统调度是建立在精确的模型之上的。而模型到数学的抽象，一方面虽然使问题得到简化，但有时也丢失了一些信息，或者对模型进行了某些限制。

2. 传统的调度系统一旦完成，在一段时间内的算法是固定的，并不能很好的适应多变的请求。

3. 传统的决策支持是在一个特定算法下，处理的能力有限，它不能感知，也丢弃了一些有用的历史信息，放弃了对这些信息进行融合、分析和推理，也就没有自我适应、自学习和自组织的能力。

基于此，本文提出了一种基于贝叶斯分类的多算法混合虚拟资源调度系统（Bayes-classifier based Mixed-algorithm Virtual-resources Scheduling System，简称$BMVS^2$），它的架构如图3.1所示：

3.1.2　系统的工作流程设计

根据所设计的虚拟资源计算平台调度系统的结构，该系统的执行流程如图3.2所示。

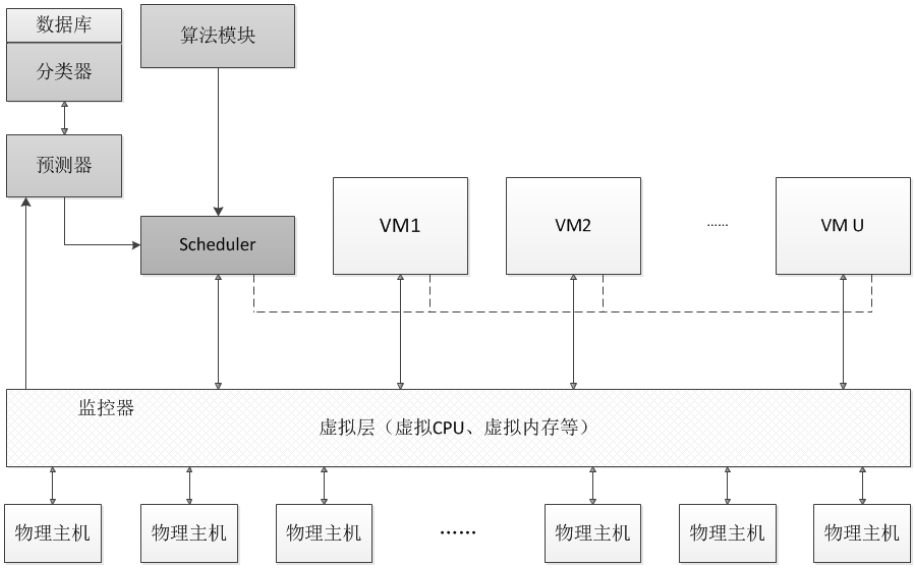

图 3.1 $BMVS^2$的架构

整个调度系统按照执行的过程主要分为两部分，一次决策过程和VM执行相关信息的搜集，分别如图3.2 的（a）和（b）所示。

1. 一次决策过程

   外界向系统提出请求，分配到每个VM上执行，监控器会获得运行过程中VM的任务请求状态信息，而预测器会从监控器获得这些信息，进行简单的处理后交由分类器进行分类。分类器使用Bayes决策，首次使用由于没有先验概率，故随机做出一个决策（或者首次使用训练收据集进行训练，获得先验概率），之后交由预测器，预测器会根据已有的算法模块，评价这次决策并选择最佳的算法进行调度，选择好算法后通知Scheduler进行具体的调度。将不同的任务请求分成不同的组态，在各个不同的组态内执行相应的分配策略。其具体的过程相当于在图3.3中的各种分类线上找到一条合理的，以形成一个更好的组态。

2. 信息保留

   一次决策过程的具体实施，必然会表现出性能的差异，通过收集这些信息，更改先验概率、分类标签等，预测器可以在下一次分类做出更加合理的决策。

3.2 $BMVS^2$的分析

3.2.1 系统结构分析

结合$BMVS^2$的架构图，该系统的各个部分的分析如下：

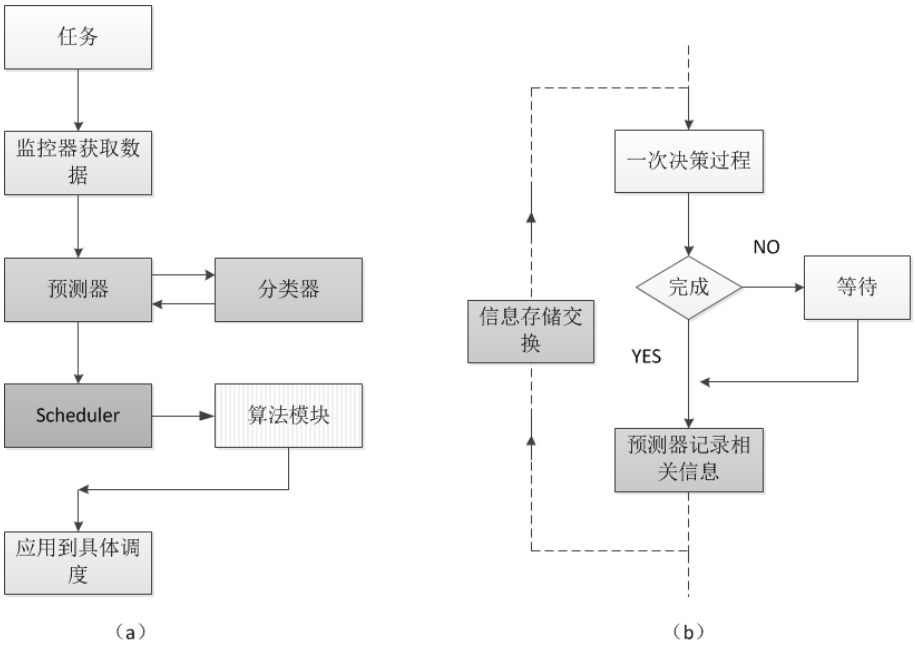

图 3.2 $BMVS^2$的工作流程

1. 硬件层

   硬件层是整个架构的基础，它们组成一个集群式的物理资源层，这些硬件资源可能是集中式的多台物理主机，也可能是分布在不同地域的分布式主机。

2. 虚拟层

   该层的作用是提供对硬件进行虚拟化，包括CPU、内存、I/O、块设备等等，同时，监控器也处于该层，监控位于该层上的VM 运行使用情况、任务请求情况，以及硬件的资源情况。

3. 知识发现

   虚拟化系统发展至今，已经出现了众多的调度算法，不乏优秀的资源调度策略，但是基本上是基于受限的数学模型抽象，拘泥于确定的模型。而随着智能系统的发展，虚拟资源调度系统下一步也会朝着更加智能的方向发展，而作为具有学习能力的系统，预测感知、分类决策、数据库模块是必不可少的。它们能够结合历史信息对当前环境状态做出一个最佳的估计。

4. 算法模块

   本系统挂载多个算法模块，由于目前已有的算法都有着特定的应用背景，比如针对CPU密集、I/O密集、计算公平、存储优先等等，而我们可以根据预测分类机制对任务请求实施组态的分类，每个组实施不同的分配策略，以达到当前的最佳期望。

5. Scheduler

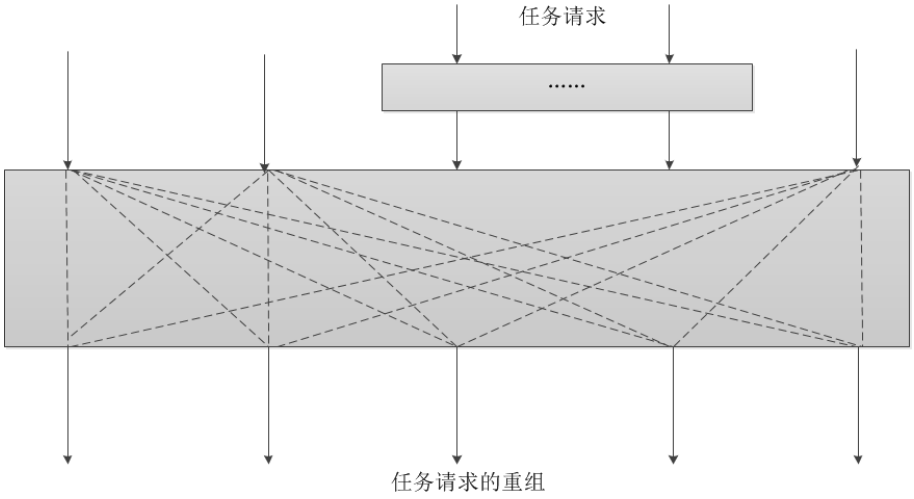

图 3.3 $BMVS^2$的任务请求重组机制

调度器是建立在分类基础之上的一个模块，在分类完成后，调度器完成对任务的划分，部署到不同的VM上，而具体到每个VM上的调度则由VM上的调度算法完成。

6. 反馈

输出的决策产生的影响直接或间接的反馈到决策输入端，这样便形成了一个闭环控制系统。为了实现闭环反馈控制，必须对决策产生的影响进行评价，由评价因子决定影响的优劣。

$BMVS^2$有着很明确的模块化思想，将所需的要素组合在一起，在特定的层上构成了一个具有特定功能的子系统，子系统作为通用性的模块与其它模块要素进行组合。模块化设计是绿色设计方法之一，具有高内聚、低耦合的特点，它已经从理念转变为较成熟的设计方法。将绿色设计思想与模块化设计方法结合起来，可以同时满足产品的功能属性和环境属性，同时满足后续的更改、添加、删除等等。

$BMVS^2$核心模块是预测分类模块，数据挖掘的任务就是从其目标上进行分类，为此，有如下定义：

定义 3.1. 数据挖掘就是对观测到的数据集（一般来讲是很庞大的）进行分析，目的是发现未知的关系和以数据拥有者可以理解并对其有价值的新颖方式来总结数据。

带有知识发现的数据挖掘就是从目标上进行分类[18]，分类是数据挖掘中非常重要的一种技术。不同的方法构造的分类器也不同，主要有决策树、贝叶斯、K-临近等等。其中贝叶斯分类器就是使用概率理论，以贝叶斯定理为基础，建立分类模型的一种学习方式。此外，分类器不会独立存在，大多会伴随**知识发现**（Knowledge Discovery，简称KD）而存在。

分类和预测是两个概念，两种方法都是数据挖掘中对数据分析的方法，但分类不等同于预测。所以在提出架构的时候将两者分离。

在具有模式的完整统计知识条件下，贝叶斯分类器是按照贝叶斯决策理论进行设计的一种最优分类器。分类器是对每一个输入模式赋予一个类别名称的模块，而贝叶斯分类器是各种分类器中分类错误概率最小或者在预先给定代价的情况下平均风险最小的分类器[19]。它的设计方法是一种最基本的统计分类方法。

3.2.2　决策分类模块的理论分析

Saridis认为智能控制是数学与语言描述和用于系统与过程的算法之间的结合[20]。为了解决资源调度问题，希望在调度的时候能够带有数据挖掘和机器学习的能力。

具有数据挖掘和机器学习的模块主要功能是：

- ➤ 接受上级数据并对此进行推理

  推理将不同的评价标准与所收集的数据结合，并从概率上评估一个决策。

- ➤ 规划

  主要是对动作进行操作，根据所选择的策略实现调度。为了计算概率，要用到先验概率、后验概率和最大似然估计等概念。

- ➤ 决策

  选择一个最佳的可能的决策。

- ➤ 反馈

  在完成一次决策后进行评估，通过学习更新概率。

- ➤ 存储交换

  更新存储在数据库中的信息。

为了从数学上给出原型系统的验证，本文首先对相关概念作出定义，然后对各个模块进行描述。

对于统计推理分类来说，数据分布的信息可以通过分析服从该分布的数据推理而来。给定一个数据集$X = \{x_1, x_2, \ldots, x_n\}$, 从而发现派生出这个数据集的总体分部属性。定理3.1中的贝叶斯法则，是一种以给定的数据集为依据或输入来估计某种属性的可能性方法。假定h_1和h_2是两个假设，则必有一个成立，但是不能同时成立。又假设x_i是可观测事件。

定理 3.1. 贝叶斯定理

$$P(h_1|x_i) = \frac{P(x_i|h_1)P(h_1)}{P(x_i|h_1)P(h_1) + P(x_i|h_2)P(h_2)} \quad \text{式(3-1)}$$

这里$P(h_1|x_i)$为**后验概率**，而$P(h_1)$是假设h_1的**先验概率**。$P(x_i)$是出现x_i的概率。$P(x_i|h_1)$是x_i元组满足给定假设的条件概率。

当存在m种不同的假设时，有：

$$P(x_i) = \sum_{j=1}^{m} P(x_i|h_j)P(h_j) \quad \text{式(3-2)}$$

因此，有

$$P(h_1|x_i) = \frac{P(x_i|h_1)P(h_1)}{P(x_i)} \quad \text{式(3-3)}$$

贝叶斯定理允许为给定的一个数据值指定一个假设的概率值$P(h_j|x_i)$。在实际问题中，x_i可以是属性值，也可以是其它数据标签。h_i可以是属性值、属性值的集合（如范围），甚至是一些属性值的组合。正如文献[21]所讨论的，有三种方法可以解决分类问题，即指定边界、利用概率分布、利用后验概率。而贝叶斯分类恰好符合这些方法，从而可以合理的进行分类问题的解决。

定义3.2给出了分类问题的定义。

定义 3.2. 给定一个由元组（条目，记录）组成的数据$D = \{t_1, t_2, \ldots, t_n\}$和一个类别集合$C = \{C_1, C_2, \ldots, C_m\}$，**分类问题**是指定一个映射$f : D \to C$，其中每个元组$t_i$被分配到一个类中，一个**类**$C_j$精确地包含了被映射到其中的元组，即$C_j$=$\{t_i|f(t_i) = C_j, 1 \le i \le n, t_i \in D\}$。

对一个目标元组进行分类时，需要利用从训练集中产生的条件概率和先验概率进行预测。这就需要将元组中不同的属性值对预测所起的作用组合起来。假设元组t_i有P个独立的属性值$\{x_{i1}, x_{i2}, \ldots, x_{ip}\}$。对于每个类$C_j$和属性$x_{ik}$，由描述阶段可得到$P(x_{ik}|C_j)$，进而可以估计出$P(t_j|C_j)$，即

$$P(t_i|C_j) = \prod_{k=1}^{p} P(x_{ik}|C_j) \quad \text{式(3-4)}$$

Algorithm1和2是分类决策的伪代码：

对于一次决策过程，本文提出一个计算公式，来评价一次分配过程，为下一次产生评价因子，首先有如下定义：

定义 3.3. 已知已经产生的分类$C = \{C_1, C_2, \ldots, C_n\}$，任务请求编号为$V = \{v_1, v_2, \ldots, v_m\}$，定义代价P作为评价标准，定义如下公式：

$$P = \frac{\sum_{i=1}^{m}\sum_{j=1}^{n} T(x_{ij})}{m} \quad \text{式(3-5)}$$

Algorithm 1 贝叶斯分类算法

Initialization:

$X = \{x_1, x_2, \ldots, x_n\}$　　▷ 待分类项

$C = \{C_1, C_2, \ldots, C_m\}$　　▷ 已有类别

Iteration:

while C_j and $j \leq m$ **do**

　Compute $P(C_j|x_i)$;

end while

if $P(C_k|X)$=max$\{P(C_1|X)$,$P(C_2|X)$,…,$P(C_m|X)\}$ **then**

　X$\in C_k$;

end if

Output:

X classified to C_k;　　▷ 最大似然

Algorithm 2 条件概率的计算

Initialization:

$C = \{C_1, C_2, \ldots, C_m\}$　　▷训练样本集

Iteration:

for all $j \in m$ **do**

　for all $i \in n$ **do**

　　Compute $P(x_i|C_j)$;

　end for

end for

Output:

$P(C_i|X)$;

其中，$T(x_{ij})$满足以下约束：

$$T(x_{ij}) = \begin{cases} 1, & v_j \text{ 属于类i} \\ 0, & \text{其它} \end{cases} \qquad \text{式(3-6)}$$

P值越高代表着决策的正确率越高，如果分类的P值比若干次的值都高，则进行存储交换，用以产生更好的决策。

由于监控器监控的数据有多种，比如CPU使用率、内存使用量、I/O带宽、网络带宽、请求向量等等，所以为了使不同的量纲能够进行比较，就要做出适当的变换，将不同的数据映射到[0,1]区间内。本文采用平移极差变换，假设n个虚拟机，R_i表示第i 个虚拟机的资源向量，R_i=$\{r_{i1}, r_{i2}, \ldots, r_{in}\}$，变换方法如下：

$$x'_{ik} = \frac{x_{ik} - \min\limits_{1 \leq i \leq n}\{x_{ik}\}}{\max\limits_{1 \leq i \leq n}\{x_{ik}\} - \min\limits_{1 \leq i \leq n}\{x_{ik}\}} \qquad \text{式(3-7)}$$

同理，任务请求的向量也使用平移极差变化得到归一化的值进行计算。

3.3 本章小结

本章分别从原型设计和原型分析两方面阐释了$BMVS^2$的具体细节。第一部分首先从架构图介绍了本系统，并详细介绍了系统的工作流程，继而引出第二节对系统从理论上进行分析。第二节着重从数学方面对分类器和预测器进行了精确的分析，并给出了其工作的伪代码，深入阐述了其实现环节，最后给出了评价准则。

第四章　$BMVS^2$的实现与测试

4.1　$BMVS^2$的实现

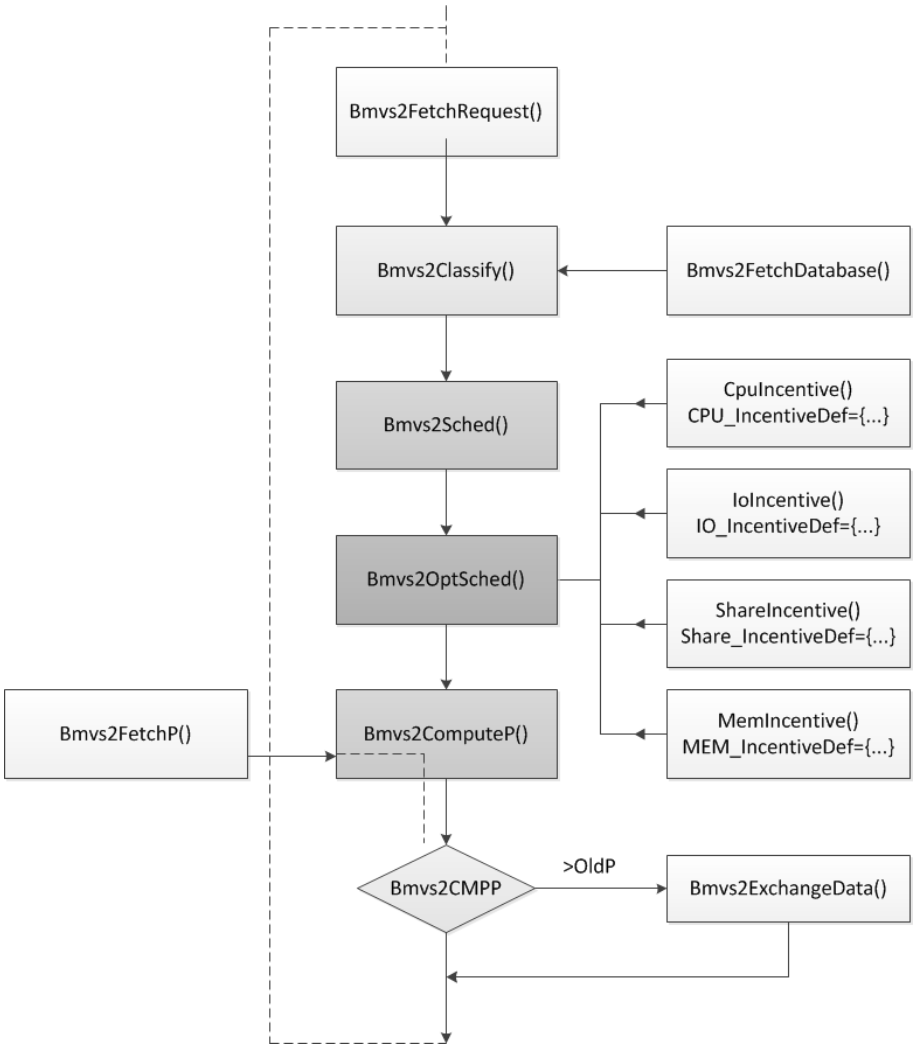

图 4.1 $BMVS^2$执行过程

$BMVS^2$的调度实现如4.1所示。执行一次这样的流程就相应的完成了一次任务请求的部署操作，任务请求的结构体如下所示，记录了任务的CPU、内存和I/O请求数据，BelongedClass成员代表的是最终划分的类别。

```
struct Request
{
        double CPU;/*CPU request*/
        double MEM;/*Memory request*/
        double IO; /*IO request*/
        int BelongedClass;
```

```
        /*
        remain area
        */
};
```

如流程图所示，该流程中的函数解释如下。

- **Bmvs2FetchRequest()**函数

首先，Bmvs2FetchRequest()函数从监控器获得任务请求数据，然后经过平移极差公式（3-7）进行归一化操作，返回一个经过归一化的请求数据。然后，归一化的数据传给分类器函数。

```
void Bmvs2FetchRequest(int ReqNum,Request *Req,Request *
   FormattedReq)
{
      //计算平移极差
      int CPU=Max(Req)-Min(Req);
      int IO=Max(Req)-Min(Req);
      int MEM=Max(Req)-Min(Req);
      for(int i=0;i<ReqNum;i++)
      {
            FormattedReq[i].CPU=(FormattedReq[i].CPU-min(
               Req.CPU))/CPU;
            FormattedReq[i].IO=(FormattedReq[i].IO-min(Req
               .IO))/IO;
            FormattedReq[i].MEM=(FormattedReq[i].MEM-min(
               Req.MEM))/MEM;
      }
}
```

- **Bmvs2Classify()**函数

Bmvs2Classify()获取已经预处理的数据，然后从Bmvs2FetchDatabase()函数获得分类标签，分类的结果是给每个人物请求的BelongedClass成员赋值。

```
      //分类
      O2 = NaiveBayes.fit(Data,Class,'dist',{'kernel','
         normal','kernel'});
      C2 = O2.predict(Data);
      cMat2 = confusionmat(Dlass,C2);
```

```
void Bmvs2Classify(int ReqNum,Request *FormattedReq)
{
```

```
    Classify(FormattedReq);
    for(int i=0;i<ReqNum;i++)
    {
        FormattedReq.BelongedClass=BelongedTo[i];
    }
}
```

• **Bmvs2Sched()**函数

BMVS2_Sched()获得分类的结果，对应各个分类预先设定的调度算法。

```
void Bmvs2Sched(int ReqNum,Request *FormattedReq)
{
    int CountA=0,CountB=0,CountC=0,CountD=0;
    for(int i=0;i<ReqNum;i++)
    {
        if(FormattedReq[i].BelongedClass==1)
        {
            A[CountA++].BelongedClass=i;
        }
        ...
        //继续划分其它三类的任务编号
        ...
    }
}
```

• **Bmvs2OptSched()**函数

Bmvs2OptSched()对已经分类好的任务列表调用相应的调度算法。由于任务编号已经在Bmvs2Sched()函数中完成分类记录，所以Bmvs2OptSched()函数直接调用这些算法库中对应的函数实施任务的部署，其中针对4类任务所使用的调度算法是已知的优秀算法，分别为CpuIncentive()、MemIncentive()、IoIncentive()和ShareIncentive()调度函数，分别代表针对CPU密集、内存密集、I/O密集和普通类任务，过程如下所示。

```
void Bmvs2OptSched()
{
    /*Task A,B,C,D already exist*/
    CpuIncentive();    /*Deploy A Type Task*/
    MemIncentive();    /*Deploy B Type Task*/
    IoIncentive();     /*Deploy C Type Task*/
    ShareIncentive(); /*Deploy D Type Task*/
}
```

• **Bmvs2ComputeP()**函数

根据公式（3-5），Bmvs2ComputeP()则计算出部署任务后的代价P，即正确分类到预估的类别中的比例，其代码如下所示。

```
void Bmvs2ComputeP(int ClassNum,int TaskNum)
{
        for(int i=0;i<ClassNum;i++)
        {
                for(int j=0;j<TaskNum;j++)
                {
                        Profit+=T(i,j);   /*公式3-6*/
                }
        }
        Profit=Profit/TaskNum;
}
```

• **Bmvs2FetchP()**和**Bmvs2CMPP()**函数

该函数用于从数据库中获取历史代价信息，供Bmvs2CMPP()函数比较，如果新的代价比历史的高，则调用Bmvs2ExchangeData() 函数实施存储交换操作。

```
void Bmvs2CMPP(double Profit)
{
        /*Fetch Profit from database*/
        int p=Bmvs2FetchP();
        /*Compare profit*/
        if(Profit>p)
        {
           /*bigger than,do exchange now*/
            Bmvs2ExchangeData();
        }
        /*else return*/
}
```

4.2 $BMVS^2$的测试

4.2.1 测试环境

本系统的测试环境如表4.1所示。其中Weka[23]的全名是怀卡托知识分析环境（Waikato Environment for Knowledge Analysis），是一款免费的，基于JAVA 环境下开源的机器学习（Machine Learning）以及数据挖掘（Data Minining）软件。Weka是新西兰怀卡托大学（University of Waikato）开发的数据挖掘实验平台，目前仍然在项目主页进行维护与更新。

表 4.1 $BMVS^2$测试环境

硬件平台	Lenovo® Y450 T6600 2.1GHz,2GB RAM
软件平台	Microsoft® Windows® 7 SP1 32bit（English） Weka Machine Learning Software V3 Microsoft® Visual Studio® 2010 Ultimate Matlab® R2012a

4.2.2　任务概述

本实验对三种不同的资源权值分布特征进行分类，假设监控的数据分别是R_i=$\{r_{i1}, r_{i2}, \ldots, r_{in}\}$，这三种权值[24]分别是：

（1）d_1=0.8，d_2=d_3=0.05

（2）d_1=0.6，d_2=d_3=0.1

（3）d_1=0.4，d_2=d_3=0.15

（4）d_1=0.2，d_2=0.2，d_3=0.3

（5）d_1=0.1，d_2=0.5，d_3=0.1

对于第（1）类，表示计算密集型的任务居多，如数值计算、数据挖掘分析等；第（2）、（3）类表示需求均衡类的任务表示，其中第（2）类CPU资源需求强于其它类的计算任务；第（4）类表示I/O型任务；第（5）类表示内存需求较高的任务，Weka的聚类分析如图4.2所示。

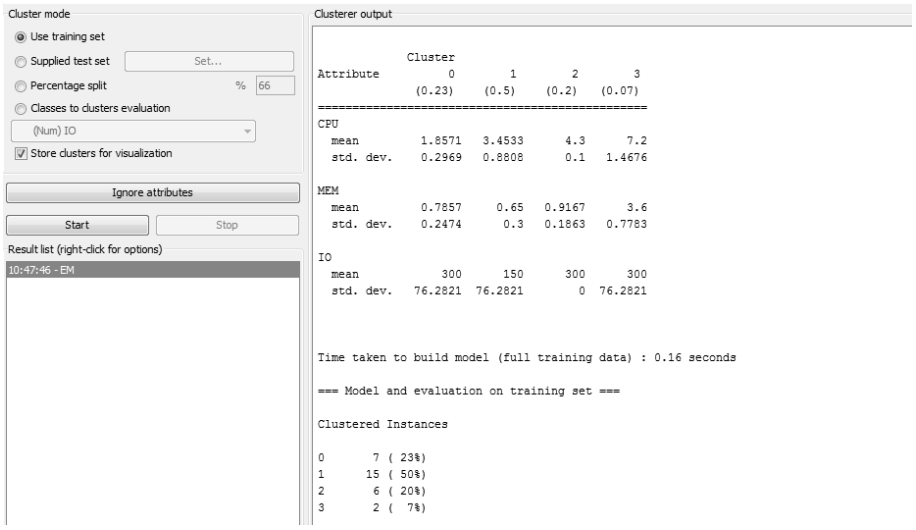

图 4.2 Weka聚类分析结果

4.2.3　测试结果

通过Weka聚类分析，我们获得一个较理想的、符合实际意义的数据分布图，其任务需求曲线如图4.3所示，其中图（a）表示的是CPU需求曲线，可以看出，

CPU需求大部分出去中间段，极端情况只有少量的分布；图（b）代表内存分布，相比CPU分布，内存处于中间部分的样本更多；I/O分布呈现出了较随机的分布，但是极端情况也很少发生，这与[7] 中Facebook实际测试分布图的情况大致相同，说明该图是符合实际情况的；图（d）用盒图表示了CPU、内存和I/O请求的大部分数据的分布情况。

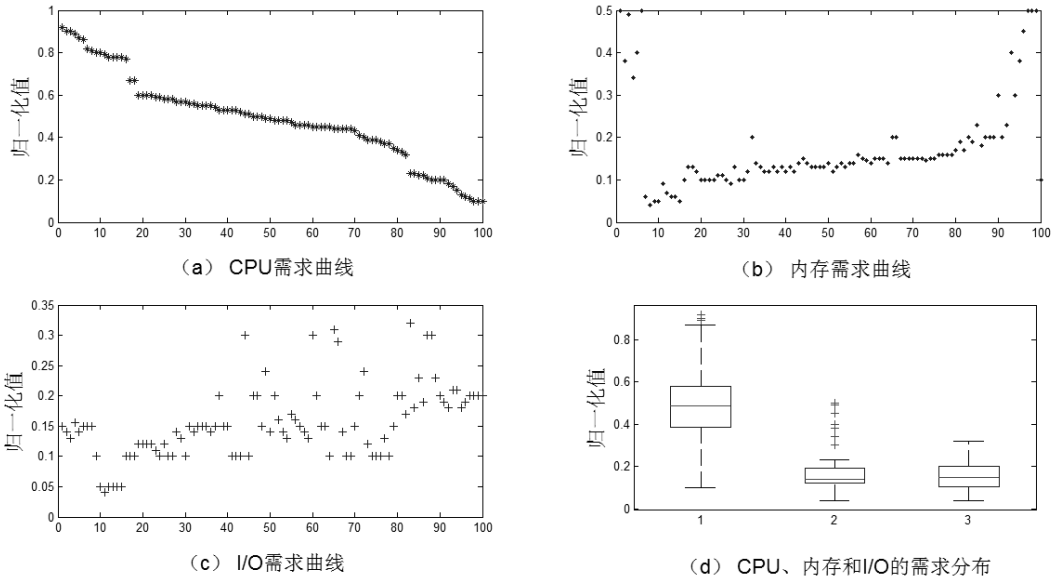

图 4.3 典型的虚拟环境下任务曲线

由如下公式可计算出分类的先验概率：

$$P(c)=\frac{NumOf(c)}{Total};c\in\{A,B,C\} \qquad \text{式(4-1)}$$

因此，P（A）=0.16，P（B）=0.05，P（C）=0.09，P（D）=0.70。测试中分别以1、2、3、4代表A、B、C、D类别，如图4.4 所示的是由100 组数据训练的结果。其中柱状图中的矩阵代表的是分类后的混淆矩阵，其中表示了正确分类与否的任务个数；以第4 类为例，从左向右依次为正确分类到第4类的66个样本，错误的分给第1类的1个、第2类2个、第3类1个；右图直观上给出了任务在空间上的分布情况。

如图4.5所示，测试100组任务请求数据而得到的数据。同4.4，该图也使用了矩阵和立体图给出详细的任务分布。

从图中可以看出，任务分布也遵从类似高斯分布的特征，对资源需求密集的任务属于少数，大多数任务属于普通类。利用公式（3-5）可以求得正确分类的百分比。同时，该矩阵还用来求得下一节定义的收益函数。

4.2.4 测试结果评估分析

分类后的任务请求，经过一个重新组合的过程，可以形成一个更优的组态。然后在此基础上重新分配任务请求的去向，如图4.6所示。

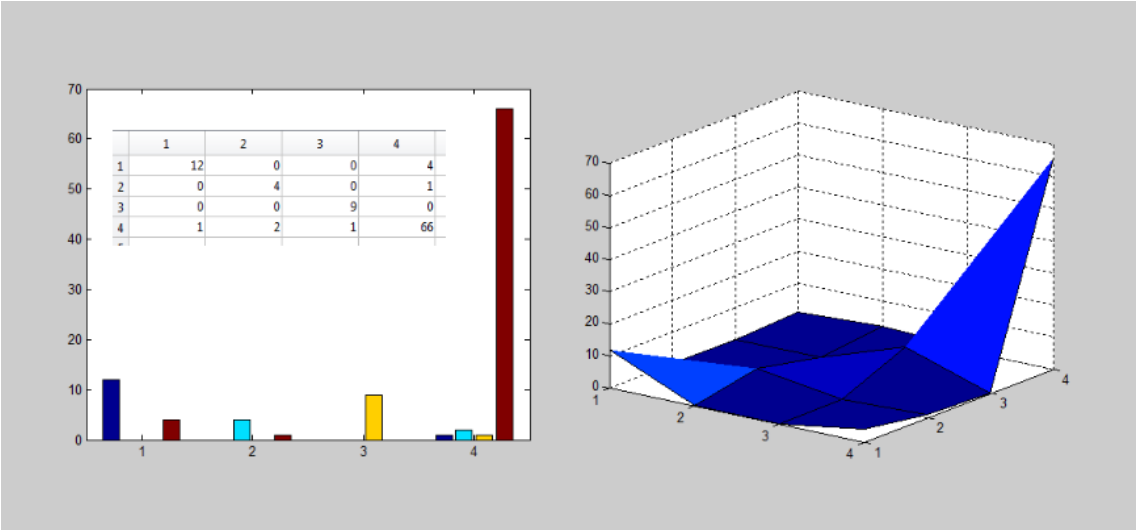

图 4.4 由测试数据集而得的分类结果

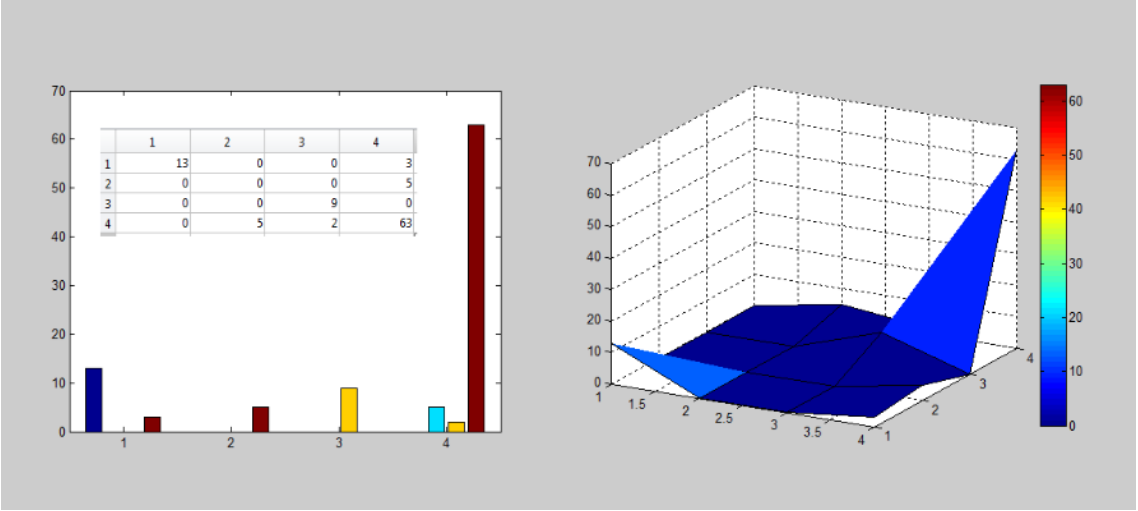

图 4.5 对100组任务请求数据集分类的结果

假设虚拟环境下有多种任务，也有多种任务分配算法。而目前的算法都有针对的特定应用环境，比如LRU解决CPU密集型、带有BOOST 特性的Credit算法具有针对I/O的优化、DRF算法具有公平的性质，能够最大化的共享资源、针对内存优化的调度算法。

定义 4.1. 每类算法对其特定的情况，定义其收益为1，否则收益为0.8，忽略请求列表的变更代价。假设有n个任务请求t=$\{t_1, t_2, \ldots, t_n\}$，收益函数定义如公式4-2所示。

$$Cost = \sum_{i=1}^{n} T(i) \qquad \text{式(4-2)}$$

其中：

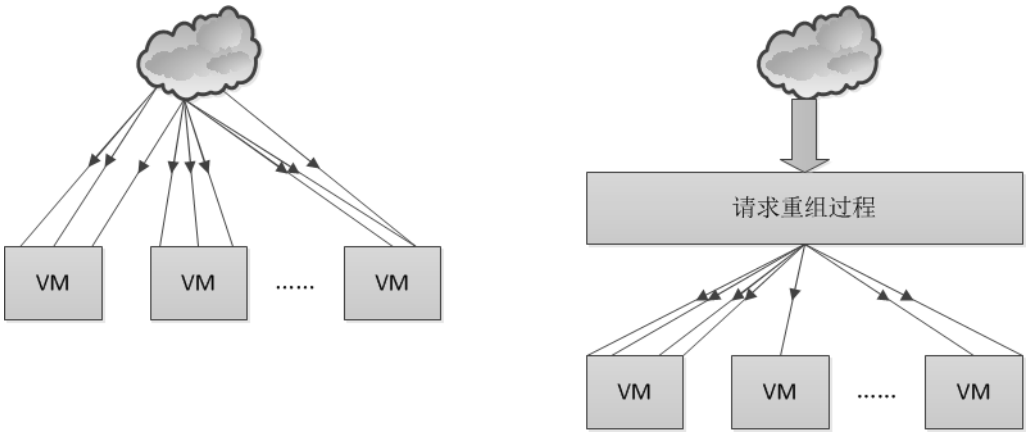

图 4.6 分类后的任务请求分配过程

$$T(i)=\begin{cases}1, & \text{适用于任务i的算法}\\ 0.8, & \text{其它}\end{cases} \qquad \text{式(4-3)}$$

则单一调度分配算法平台的收益和$BMVS^2$收益的对比如图4.7所示：

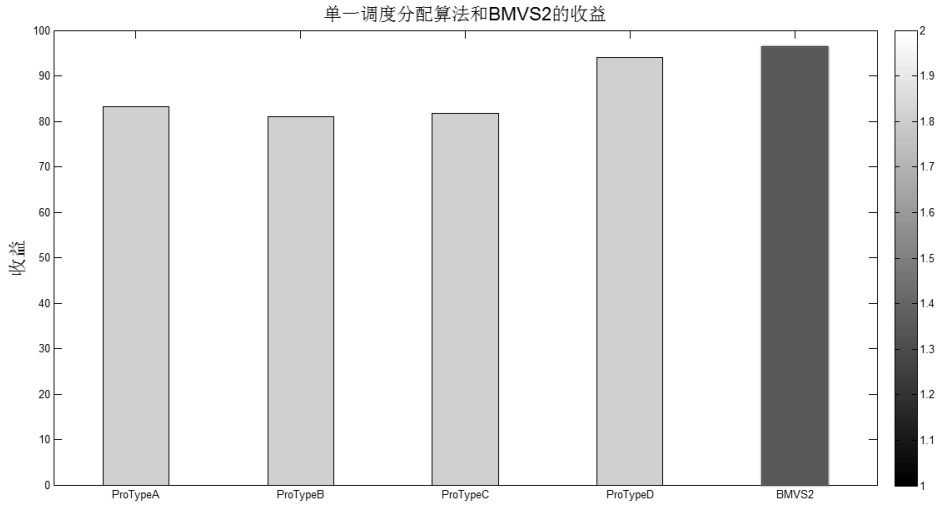

图 4.7 单一调度分配算法和$BMVS^2$的收益对比

如图4.7所示，$BMVS^2$有着更高的收益，代表着有更高的系统平均利用率。A、B、C、D分别有着83.2、81、81.8、94 的收益，而本文提出的$BMVS^2$有着96.4的收益，而且随着任务请求曲线（类高斯分布曲线）的逐渐平坦，$BMVS^2$会获得更高的收益，如图4.8所示。

4.2.5 结论

从实验的数据我们可以看出，根据不同的任务请求，结合不同算法的不同标准得到一个最优的部署是可能和合理的。而且这种需求是现实存在的，我们可以通过这种组合获得性能上的提升、体系架构的智能化发展等等。

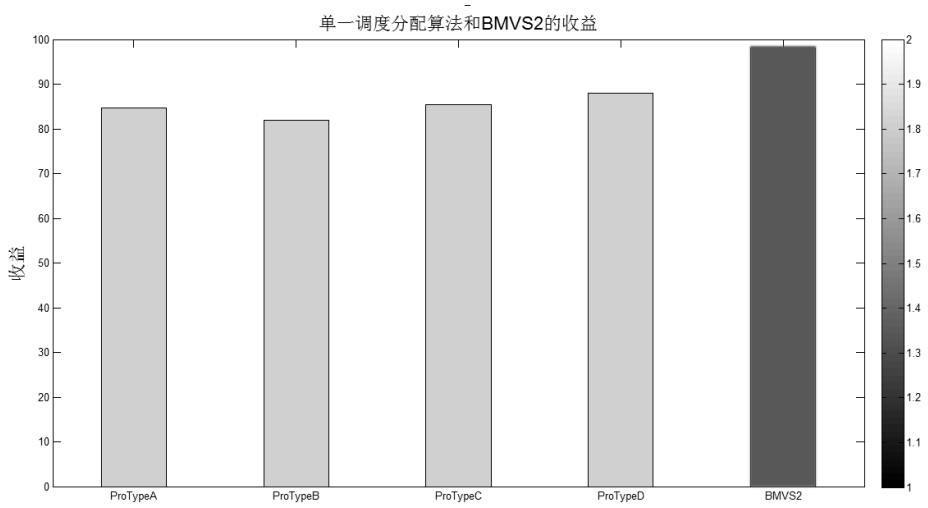

图 4.8 单一调度分配算法和$BMVS^2$的收益对比（随着资源分布的均衡）

通过实验可知，针对不同的应用背景，使用合适的算法进行调度是比原有策略更好的一个决策。

4.3　本章小结

本章分别介绍了实现测试的软硬件平台、测试过程。通过测试得到的数据和图，从实验角度分析了系统的可行性，并对本系统的核心模块－预测和分类模块同以往没有使用融合预测调度的系统收益进行了比较，评估了测试结果，并获得了预期的结果。最后对本次测试做出了总结。

第五章　总结与展望

5.1　论文所完成的工作

本文提出了一个基于贝叶斯分类的多算法混合虚拟资源调度系统$BMVS^2$，设计了针对虚拟资源请求的分类划分机制，实现了虚拟计算平台下任务资源请求的最佳组态。

资源有限的情况下，$BMVS^2$追求的目的就是寻找一种分配策略从而最大化的满足各个请求的需求，并使得该分配方案获得时空上的最优。在$BMVS^2$中，设计了一种基于数据挖掘分类技术的预测机制，以实现对任务请求的分类划分，获得更好的组合方式。

本文主要做了以下工作：

1. 对目前云计算、虚拟化技术的发展做了详尽的分析，结合国内外在当前领域的研究现状，总结了架构以及调度算法的缺点和待改进部分。

2. 针对虚拟资源调度在当前趋势下的新需求，融合了最先进的数据挖掘和知识发现功能，提出了一个具体的对应架构，该架构既具有描述性，也具有预测性。

3. 在所提出的基于贝叶斯分类的多算法混合虚拟资源调度系统的架构上，结合贝叶斯分类模型和反馈机制，对其进行了模型分析和流程分析。通过参考[22]，对核心模块进行了理论分析和代码的提出，进行了核心模块的流程控制。并在最后提出了一个评价因子，对一次决策过程给出计算，最为存储交换的衡量标准。

4. 利用开源数据挖掘软件Weka对数据进行聚类分析，然后利用Matlab进行了数据集训练和测试，在测试平台上实现了系统关键技术的分析。通过分析验证的结果，总结了具体应用背景下的策略选择。

本文的创新在于提出了一种不同于以往虚拟资源调度系统的架构，改变了传统调度平台的方式，在比传统架构更高的一层上建立融合了数据挖掘技术。目前的各个算法都有着有限的应用背景，而数据集存在着适应某些特定算法的特点。数据是可以被预处理和加工的，在执行一次决策之前，启发性地预测和分类，对于提高分类正确性有着重要的意义。

5.2　前景与展望

系统工程是一个庞大而复杂的集合体，它需要的理论支持和工程技术都是难以想象的，在完整实现环节上本系统还有很多工作要做，如何设计出更加有效的

调度系统还有待实践检验。此外，本原型系统也需要进行进一步的完善与实现，争取早日完成实体系统的部署。

信息技术时代，必将要求系统越来越智能化、自动化，并且具有学习能力和预测机制。与传统的系统相比，带有学习和知识发现的系统更能适应复杂的环境。可以满怀信心的说，这也是目前各个系统的发展方向，在如此广泛的应用领域，这种系统必将取得长足的发展，并推动该学科到一个新的高度。

致　谢

转眼间，我已经在美丽的西安电子科技大学度过了四个年头。四年，这是我人生中非常重要的时段，我有幸能够接触到这些不仅传授我知识、学问，而且从更高层次指导我的人生与价值追求的良师。他们使我坚定了人生的方向，获得了追求的动力，留下了大学生活的美好回忆，我可以肯定地说，我的大学是丰富的。

行文至此，我的这篇论文已接近尾声；本科的四年学习生活也即将敲响结束的钟声。离别在即，站在人生的又一个转折点上，心中难免思绪万千，一种感恩之情油然而生。

首先我要衷心的感谢我的导师马建峰教授，感谢确定导师后马老师一直悉心的指导。他渊博的专业知识和平易近人的态度，都令我钦佩。马老师对我们的要求都很高，令我们有一种很受重视的感觉，也有更大的动力。虽然公务繁忙，却也经常组织我们一块讨论，然后每次总会提出建设性的建议，对此我表示非常的感谢。

感谢我的师兄，卢笛、习宁、张涛、王一川，他们悉心指导我们解决毕设中遇到的问题，每周都组织我们去会议室讨论，甚至从老校区到新校区给我们指导，牺牲自己的学习时间来帮助我们最终完成毕设所要求的目标。

感谢我的父母，他们在家里辛勤的劳作，为的是我能够安心地学习，给自己一个更好的未来。

感谢实验室的各位组员，方祯、莫若、刘旭启、沈伟、刘志全、武磊、廉亚男、董超等，在平时遇到问题后，我们一起探讨，一起解决。不仅如此，我们也成了生活中的好朋友。

感谢生活了四年的舍友，我们来自不同的地方，有着不同的文化习俗，却相处的非常融洽，成了无话不说的好友。

时间的仓促及自身专业水平的不足，整篇论文还存在尚未发现的缺点和错误。恳请阅读此篇论文的老师、同学，多予指正，不胜感激!

.....

最后，感谢所有帮助和关心过我的人们，谢谢!

参考文献

[1] Strachey C. Time Shring in Large Fast Computers[A]. In International Conference on Information Processing(1959).

[2] 雷葆华，饶少阳，江峰等. 云计算解码[M]. 电子工业出版社(2011.4).

[3] Mark Stillwell,David Schanzenbach,Viven F. Resource Allocation Algorithm for Virtualized Service Hosting Platform[J]. Journal of Parallel and Distributed Computing(2010).

[4] Shailesh Sawant A Genetic Algorithm Scheduling Approach for Virtual Machine Resources in a Cloud Computing[J]. San Jose State University Master's Theses and Graduate Research(2011).

[5] Richard M.,Michael Luby. A Probabilistic Analysis of Multidimensional Bin Packing Problems. University of California at Berkeley,University of Toronto,University di Roma,ACM(1984).

[6] Ekow,Ali Pinar,Doron Rotem. A $\Theta(n)$ Approximation Algorithm for 2-Dimensional Vector Packing. Lawrence Berkeley National Laboratory,Sandia National Laboratory(1984).

[7] Ali Ghodsi, Matei Zaharia, Benjamin Hindman. Dominant Resource Fairness: Fair Allocation of Multiple Resource Types[R]. University of California, Berkeley .MESOS Project(2011).

[8] Xin Sun,Sen Su,Peng Xu. Multi-dimensional Resource Integrated Scheduling In a Shared Data Center[J]. 31st International Conference on Distributed Computing Systems Workshops(2011).

[9] Daniel Nurmi,Rich Wolski,Chris Grzegorczyk. The Eucalyptus Open-Source Cloud-Computing System. University of California,SB.

[10] Open Nebula. http://www.opennebula.org

[11] Openstack. http://openstack.org

[12] VMware. http://www.vmware.com/cn/

[13] 维基百科中文. http://zh.wikipedia.org/wiki/Wikipedia:虚拟化

[14] IBM虚拟化发展史. http://www-900.ibm.com/cn/itmanager/virtualization/solutions/virtualization/history.shtml

[15] Xen,University of Cambridge. http://www.cl.cam.ac.uk/research/srg/netos/xen/.

[16] VMware Infrastructure. http://www.vmware.com/products/vi/overview.html.

[17] Libvirt Virtualization API. http://libvirt.org/.

[18] Hand D,Mannila H,Smyth P. Principles of Data Mining[M]. (2002).

[19] 福永奎之介著,陶笃纯译. 统计图形识别导论[M]. 科学出版社(1978).

[20] Saridis G N,Valavanis K P. Mathematical Formulation for the Organization Level of Intelligence Machines[M]. Proc.Conference of Robotics and Automation,San Francisco,CA,Apr(1986,4:267-272).

[21] Ruby L,Kennedy ,Yuchun Lee. Solving Data Mining Problems Through Pattern Recognition[M]. Englewood Cliffs,N,J.:Prentice Hall.

[22] Margaret H.Dunham. Data Mining Introductory and Advanced Topics[M]. Chap3,4.Pearson Education,(2006).

[23] Weka. http://www.cs.waikato.ac.nz/ml/weka/.

[24] 刘伯成，陈庆奎. 云计算中的集群资源模糊聚类划分模型[J]. 《计算机科学》10A，卷38，(2011).