云计算环境下实训教学平台动态迁移策略①

廖大强

(南华工商学院, 广州 510507)

摘 要: 云服务环境下最大特点是按需交付, 通过虚拟化技术将相关资源构建统一调度池, 并且按照用户需求为用户提供服务, 因此, 云服务具有并行计算、开放性以及按需交付特性. 对于实训教学平台来说, 在云计算环境下需要面对各种用户需求, 如请求任务各种各样, 实验任务类型不尽相同, 设备资源存在较大差异, 通过虚拟化技术来实现规范化管理何资源共享, 对云资源进行调度来才能有效满足用户需求, 为此, 在本文中提出了云计算环境下实训教学平台动态迁移策略. 策略设计了三层协同资源调度机制来实现对资源和任务管理, 重点研究了任务分割、资源划分、资源调度策略等, 在此基础上对系统进行仿真实验, 验证云计算环境下实训教学平台动态迁移策略可行与有效性.

关键词: 云计算; 实训教学平台; 任务分割; 资源划分; 动态迁移

Dynamic Migration Strategy of Practical Teaching Platform Based on CloudComputing Environment

LIAO Da-Qiang

(Nanhua College of Industry and Commerce, Guangzhou 510507, China)

Abstract: The biggest feature of Cloud service environment is on-demand delivery, through virtualization technology will be related to build a unified scheduling of resources, and in accordance with the needs of users to provide services, therefore, cloud service has parallel computing, open and on-demand delivery characteristics. For training teaching platform, in the cloud computing environment need to face a variety of users' needs, such as the request of a variety of tasks, the type of experimental tasks are not same, there is a big difference between the device resources, through virtualization technology to achieve standardized management of the resource sharing, cloud resource scheduling can effectively meet the needs of users. Dynamic migration strategy is designed to achieve the management of resources and tasks in the three layers of collaborative resource scheduling mechanism. This research focuses on the task partitioning, resource allocation, resource scheduling strategy, etc., based on the design of the prototype system, the system simulation experiments, is conducted to verify the feasibility and effectiveness of the dynamic migration strategy of training platform in cloud computing environment.

Key words: cloud computing; practical teaching platform; task segmentation; resource dynamic migration

1 引言

在云服务环境下, 对于实验任务来说, 在不同时间所需要资源是不尽相同, 可能在某一个时刻需要计算资源, 在另外时刻需要存储资源, 通过占据相应资源来实现任务执行, 在实训教学平台中用户实验任务也具有较大差异, 甚至是不同学科交叉, 因此, 在进行云服务实训教学平台资源调度时候, 首先是需要对资源规范以及整合, 进行不同功能资源等级划分, 实现实训教学平台资源统一调度与管理, 为后续调度提供良好基础环境, 这是实训教学平台首先需要解决问题[1]. 此外, 在对用户需求满足上需要考虑不同用户对不同资源需求, 特别是在实训教学平台资源需求,

① 基金项目:南华工商学院科研课题阶段性成果(15K03)

收稿时间:2015-11-19;收到修改稿时间:2015-12-20 [doi:10.15888/j.cnki.csa.005231]

需要将资源域任务高效分配, 所分配资源可以有效满足实训教学平台任务, 完成实验任务. 为此, 在云服务环境下, 对于实训教学平台中任务需要进行分割, 通过分割就可以更好实现资源调度与匹配, 提高任务执行效率. 在云服务环境下, 实训教学平台也要考虑各个应用负载均衡, 通过对资源合理调度实现资源动态均衡, 因此合理资源调度策略是实现实训教学平台重要技术手段[2].

柴亚辉针对资源调度提出基于云计算计算机与软件实验资源管理模型, 介绍一个基于云计算技术的计算机实验资源管理系统与方案[3]. 此外, 陈传波提出云服务传统资源调度的其他模型和策略例如遗传算法(GA)、蚁群算法(ACO)、粒子群优化算法(PCO)和模拟退火算法(SA)[4], 利用上述算法就可以实现对资源调度与共享, 其来源于自然现象与规律, 在面对复杂环境资源调度时候, 存在一定缺陷与不足[5]. 如对于遗传算法来说, 在进行资源调度最优解时候, 仅仅是提供一个实现通用框架, 蚁群算法主要是通过信息素正反馈机制来来实现最优解, 但是并没有与实际状况结合, 无法很好适应复杂云服务资源环境[6]; 粒子群优化算法在进行最优解时候非常依靠粒子社会行为, 在实际应用时候存在灵活性不足问题[7]; 模拟退火算法在进行云服务资源调度最优求解时候收到温度和材料影响非常大, 并且在有限自治域中[8].

在本文中将从任务分割、资源划分、资源调度三个角度来研究实训教学平台, 实现对资源最优利用, 提高资源调度效率, 满足用户对于实验活动需求.

2　资源调度模型设计

在云服务环境下, 实验中任务在不同时刻所需求资源是不尽相同, 因此, 对于实训教学平台来说如何更好实现资源调度来实现用户需求, 提高资源调度效率, 是云服务环境下资源调度所需要解决问题.

2.1 传统调度策略

对于资源调度, 在以往策略是在一个实验任务完成后将释放资源调度给下一个实验任务, 对于上述传统资源调度模式具体如图 1 所示.

从上面关于传统资源调度模型可以看到, 在这个策略没有很好与云服务环境进行有效结合, 并且进行云服务资源调度求解策略中或多或少存在不足和缺陷, 例如适应性与灵活性不足, 无法很好满足实际环境, 因此, 需要对传统云服务资源调度策略进行改进, 使得更好满足云服务环境[9].

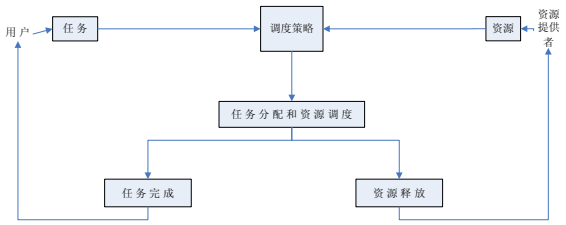

图 1　传统资源调度模型

2.2 实训教学平台资源调度

在面向实验需求资源调度其流程具体如图 2 所示.

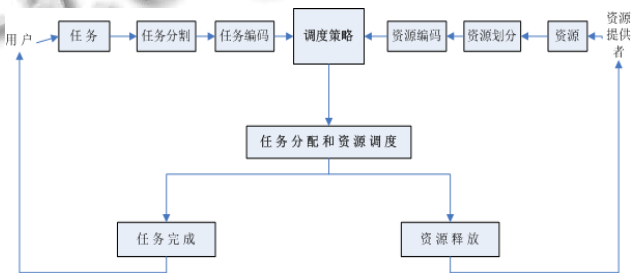

图 2　实训教学平台资源调度流程

实训教学平台资源调度模型可以看到, 其主要包括了几个重要内容, 将模型划分为任务分割、任务编码、资源划分、资源编码四项内容[10]. 首先是从用户需求处得到所需要完成任务, 接着对上述任务进行分割操作, 得到不同大小任务, 对分割之后任务进行编码, 形成具有先后顺序任务队列 M; 对于资源来说, 也是如此, 首先从资源提供方将资源进行划分, 并且对分块之后资源进行编码, 形成具有一定等级功能资源队列 N. 接下来是通过任务分配和资源调度策略, 将资源 N 有效分配给任务队列 M, 并且这个分配是合理, 可以提高任务完成, 实现任务和资源高效匹配, 这样就可以较好提高资源利用率[11].

1)资源调度目标. 在云服务环境下面向实训教学平台资源调度, 其设计目标就是实现在不同时段资源可以被不同任务利用, 提高资源利用, 实现任务和资源高效匹配.

2)资源分类. 在实训教学平台中所需要利用资源可以使多种多样, 不仅仅是 CPU 计算资源, 还有内存、存储以及数据库服务资源等等.

3)资源调度模型. 在云服务环境下资源调度中,

其实现策略是将任务与资源进行一定优化处理, 可以利用资源调度策略与模型实现资源域任务高效匹配映射. 在云服务环境下实训教学平台资源调度模型可以表示如图 3 所示.

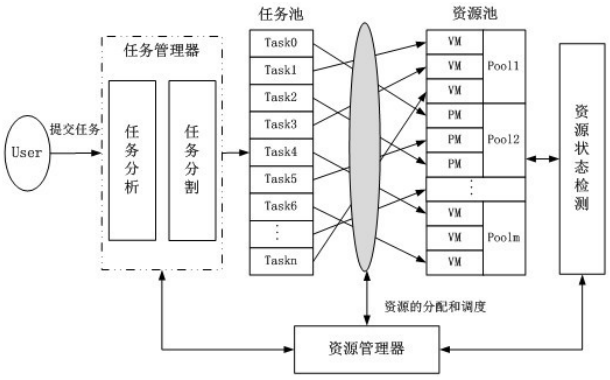

图 3 实训教学平台资源调度具体模型

在上述实训教学平台资源调度模型中, 主要包括了几个重要内容任务管理器、任务池、资源池、资源管理器以及资源状态检测, 其实现过程具体如下:

Step1. 首先是用户将所需要完成实验任务提交给任务管理器, 任务管理器在接收到相关任务数据之后, 对任务进行分析;

Step2. 根据一定分割策略将所提交任务进行分割操作, 得到大小不一任务数据;

Step3. 对上述任务进行编码, 在编码时候设定一定优先级顺序, 形成具有一定优先级别任务队列, 并且将任务提交到任务池中等待调度;

Step4. 将资源调度策略应用到上述任务队列与资源池匹配中, 使得不同实训教学平台任务可以得到所需资源, 实现高效资源调度与匹配;

Step5. 实训教学平台任务队列在得到资源后执行完成 .

2.3 实训教学平台任务调度

2.3.1 任务定义与描述

首先需要对任务进行定义, 在本文研究中任务主要值得是在一定时间内所完成具有优先级别实验集合. 在本文所研究云服务环境下实训教学平台中任务具有以下特点:

1)在一定时间范围;

2)任务主要有一系列实验所组成, 并且每个实验不尽相同;

3)对任务可以进行分割, 经过分割之后获取得到不可分割原子任务;

4)经过分割之后不同任务在时间上具有先后顺序.

在实训教学平台中任务在执行过程中主要有两种依赖关系, 分别为并行执行和顺序执行.

在进行任务执行时候, 其中每一个实验任务都有其生命周期, 其生命周期主要包括了六种状态, 不同生命周期状态可以相互进行转换, 在进行任务执行中需对任务生命周期进行管理, 在对生命周期状态转换中其过程如图 4 所示.

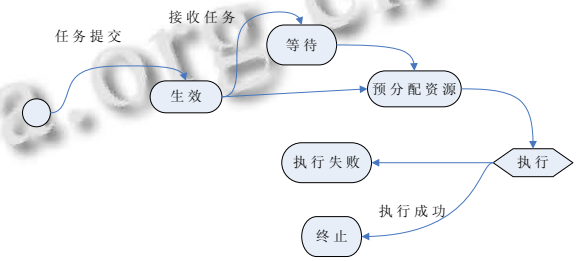

图 4 任务生命周期

状态 1: 对于任务来说, 其完成需要一定资源, 用户将其提交到云服务资源调度中, 那么任务此时就处于生效状态;

状态 2: 任务在提交之后, 对任务与资源进行匹配, 如果匹配成功符合其所需要资源, 任务就降入到预分配资源状态, 如果没有符合任务所需要资源, 那么任务就进入到等待状态;

状态 3: 任务管理器在接收到任务之后就进入到任务等待状态;

状态 4: 对任务队列中相关任务如果具有运行资源, 那么其就降入到预分配资源状态;

状态 5: 任务在获取得到相关资源后, 那么任务就进入到执行状态;

状态 6: 任务在得到所需要调度资源并且任务可以执行, 那么任务在执行完成之后就进入到终止状态; 如果不是话, 任务就进入到执行失败状态.

上述是任务在进行转换过程中各个任务状态, 但是对于原子任务来说, 上述状态中任务执行失败状态是不包括, 如果原子任务在资源调度过程中, 出现执行失败情况, 那么任务将重新提交, 使得任务重新进入到提交状态, 使得原子任务可以继续等待预分配资源. 对于实验任务及其属性来说, 可以通过如下方式来进行描述, 具体如表 1 所示.

表 1　实验任务及其属性描述表

项目	描述
任务序列	$T=\{T_1,T_2,T_3,...,T_n\}$
子任务	$T_i=\{T_{i,1},T_{i,2},T_{i,3},...,T_{i,m}\}$
子任务不同属性需求	$T_{i,j}=\{TP_{i,j},TR_{i,j},TST_{i,j},TC_{i,j}\}$
不同原子任务其资源分配与并行程度	$TP_{i,j}\begin{cases}0,\text{没并行任务}\\k,\text{有k个并行任务}\end{cases}$
对 CPU、内存、存储和应用程序资源需求	$TR_{i,j}=\{TOS_{i,j},TCPU_{i,j},TMR_{i,j},TSC_{i,j},TA$

首先是实验任务集合. 在实训教学平台中, 对于实验任务来说, 其任务序列可以表示为一系列集合 Ti. 在上述实验集合系列中, 实验任务序列号表示 n. 对于集合来说, 具有一定优先级别, 按照任务优先级进行资源调度, 并将其掺入到任务执行队列中.

其次是子任务集合. 对于实验任务来说, 可以对其进行分割成为若干个子任务, 通过分割就可以不同功能和要求子任务, 在上述子任务集合中, 其中 i 表示是任务序号, j 表示是子任务时间顺序, 一般情况下, 如果任务优先级别相同那么将按照时间先后顺序进行执行. 对于子任务来说, 主要包括了原子任务并行程度、资源需求数、任务执行时间、子任务生命周期状态, 对上述子任务不同属性需求. 在上述任务属性描述中, $TP_{i,j}$ 是并行程度, $TR_{i,j}$ 是资源需求数, $TST_{i,j}$ 是完成所需要时间, $TC_{i,j}$ 是生命周期状态.

接着对实验任务中子任务来说, 对于不同原子任务其资源分配与并行程度都不尽相同, 可以对实验任务子任务属性进行赋值, 如对原子任务 $T_{i,j}$ 来说, 对其并行程度表示为 $TP_{i,j}$.

与上述类似是, 对原子任务 $T_{i,j}$ 来说, 在对 CPU、内存、存储和应用程序资源需求可以通过如下集合来描述. 在原子任务属性集合中, 通过不同赋值来实现对云服务环境下实验任务进行定义, 其中 $TOS_{i,j}$ 取值有两个, 分别为{0, 1}, 其中 0 表示是实训教学平台中任务执行需要 Windows 环境, 1 表示是实训教学平台中任务执行需要 Linux 环境.

对于不同原子任务、应用程序资源、任务资源需求、内存需求其集合可以表示如下表 2 所示.

表 2　不同任务资源集合表示

功能	集合
原子任务 $T_{i,j}$ 对应用程序资源需求	$TAPP_{i,j}=\{TAPP_{i,j,1},TAPP_{i,j,2},...,TAPP_{i,j,m}\}$
应用程序资源进行合并需求集合	$RTYPE=\{RTYPE_2,RTYPE_2,...,RTYPE_t$
任务资源需求	$RTS_{i,s}=\{TR_{i,1},TR_{i,2},...,TR_{i,m}\}$
任务 Ti 对于计算资源需求 RTCPUi, m	$RTCPU_{i,m}=\sum_{j=1}^{m}TCPU_{i,j}$
任务 Ti 对于内存需求 RTMEi, m	$RME_{i,m}=\sum_{j=1}^{m}TME_{i,j}$
任务 Ti 对于存储需求 RTSCi, m	$RTSC_{i,m}=\sum_{j=1}^{m}TSC_{i,j}$
任务 Ti 经过分割之后得到 m 个原子任务, 对于应用程序容量表示为 RAPPi, m	$RTAPP_{i,m}=\{m,\underbrace{0,0,...,0}_{p-1\text{个}0}\}$
应用程序容量可以计算总时间 TETi, m	$TET_{i,m}=\sum_{j=1}^{m}TST_{i,j}$

2.3.2 实验任务分割

在对虚拟平台中任务进行分割中, 可以根据任务功能特点、执行特点、执行时间段以及所需求资源数, 按照上述不同需求进行分割, 经过分割之后就可以获取得到子任务, 并且子任务是不可分割原子任务, 其实现过程可以描述为如图 5 所示.

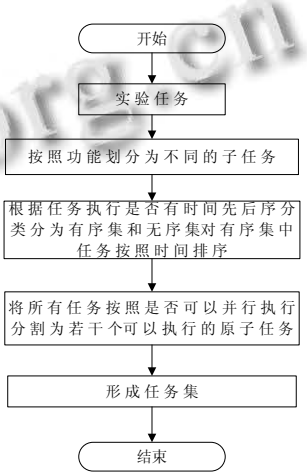

图 5　任务分割流程图

在分割过程中其流程具体如下:

Step1. 按照实训教学平台中功能进行分割, 在分割时候结合分割后子任务在执行顺序不同, 将其划分为有序任务集和无序任务集; 在分割时候, 对无序任务集来说, 主要存储是子任务集是可以动态调整子任

务, 有序任务集则是具有时间顺序执行子任务, 按照时间来进行排列;

Step2. 对属于同一个功能类型子任务进行继续分割, 使得经过分割之后任务不可再分割原子任务, 并且将其提交到实验队列中;

Step3. 按照上述要求对其他用户任务进行分割, 完成全部分割过程, 对于任务分割.

前面对任务分割过程进行了详细介绍, 将其分为几个阶段, 主要依据是按照功能类型不同进行任务分割, 在实现分割过程中, 需要遵循如下原则, 具体如下:

(1)经过分割之后原子任务, 在进行任务执行时候, 在同一个时间范围内, 其对于资源需求也是并行;

(2)在分割后所得到全部原子任务资源需求需要小于资源池中资源配置.

(3)此外, 在对实验分割过程中, 分割之后实验任务需要与资源池中节点配置相当, 否则话, 可能会造成资源闲置或大量资源迁移.

(4)对于分割实验任务来说, 对于应用程序需求数与种类需要符合下面关系, 其可以描述如下公式:

$$\{RTYPE=\{RTYPE_1,RTYPE_2,RTYPE_3,...,RTYPE_p,\}\}\subseteq \{OTYPE=\{OTYPE_1,OTYPE_2,OTYPE_3,...,OTYPE_q\}\}$$

2.3.3 资源定义与描述

前面对云服务环境下实验任务分割进行分析和定义, 接下来需要对云服务环境下资源进行划分, 在这个过程中首先需要资源定义.

在云服务环境下资源主要包括了两个方面内容, 分别为: 物理资源和虚拟资源. 首先是物理资源, 在云服务环境下其资源定义为物理主机所包括 CPU、内存、存储容量、应用程序等, 可以承载实验运行所需要环境. 其次是虚拟资源, 在这部分主要是物理服务器、存储等经过虚拟化所得到资源池, 可以满足实验需求, 主要体现为云服务中虚拟机. 从这个角度来说, 物理主机或者虚拟机可以看作是云服务环境下资源节点. 在云服务环境下资源中物理资源和虚拟资源具有如下特点:

(1)对于资源中无论是物理资源还是虚拟资源都可以进行构建, 通过构建得到资源池, 并且可以对资源池进行划分, 以此来满足实验任务需求;

(2)在由物理资源和虚拟资源所构成资源池中, 对于物理主机或虚拟机其功能是类似, 也都可以资源调度;

(3)对于构建资源池其大小或容量不尽相同, 其中节点配置也不一样;

(4)对资源池中节点, 都包括运行所需要基本配置, 如 CPU、内存、存储容量、应用程序, 可以为外部用户提供调度服务;

(5)对于处于资源池中资源, 经过调度和配置后可以实现负载均衡.

与任务类似, 对于资源来说, 也是有其生命周期, 在完成相关资源调度与任务后, 可以释放和回收, 并且处于生命周期中不同状态也是可以相互转换, 主要包括了六种生命周期状态, 其相互转换过程可以如图 6 所示.

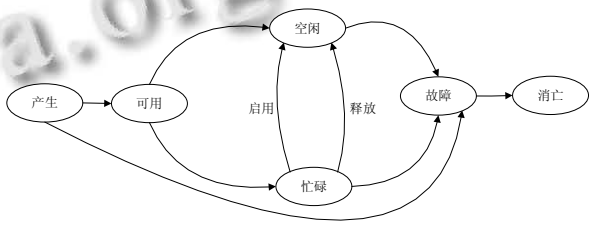

图 6 资源节点生命周期图

状态 1: 资源产生状态, 主要是初始化阶段, 将资源从源节点中取出就进入到产生状态;

状态 2: 如果某资源在完成初始化配置之后, 就可以进入到可用状态, 等待资源调度;

状态 3: 资源处于可以被调度状态, 如果调度为资源分配相应任务, 那么就处于忙碌状态, 否则话, 就处于空闲状态;

状态 4: 如果资源处于空闲状态, 经过节点调度之后就可以转化为忙碌状态, 如果资源被释放之后就处于空闲状态;

状态 5: 在运行过程中如果资源出现问题, 那么资源状态就转变为故障状态, 无法为外面提供服务, 经过一定修复之后处于故障状态也可以转化为可用状态, 如果无法修复话, 就删除此资源节点.

接下来对资源进行描述, 在完成资源状态描述后, 对资源池中资源进行描述, 在这个过程中, 资源以及资源池可以分为如下几种情况, 具体如表 3 所示.

表 3 资源描述结果表

项目	描述
物理服务器资源	$DataCenter=\{PH_1,PH_2,PH_3,...,PH_N\}$
功能资源池	$ResourcesCluster=\{RP_1,RP_2,RP_3,...,RP_M$

由物理机资源池或虚拟机资源池来组成	$RP_i=\{VM_{i,1},VM_{i,2},VM_{i,3},...,VM_{i,k}\}$ 或 $RP_i=\{PM_{i1},PM_{i2},PM_{i3},...,PM_{ik}\}$
物理特征若干资源属性(CPU、内存、存储容量、应用程序、运行状态、故障率)	$R_{i,j}=\{CPU_{i,j},ME_{i,j},SC_{i,j},APP_{i,j},RS_{i,j},FR_{i,j}$
$CPU_{i,j}$ 份额值和处理器个数	$CPU_{i,j}=\{CPUA_{i,j},CPUB_{i,j}\}$
资源节点统计种类	$OTYPE=\{OTYPE_1,OTYPE_2,OTYPE_3,...,OTYPE_q\}$
应用程序配置	$APP_{i,j}=\{APP_{i,j,1},APP_{i,j,2},APP_{i,j,3},...,APP_{i,j,s}\}$

在云服务中, 其资源池主要是由物理资源和虚拟资源构成, 组成资源集群, 在这个资源集群中包括了 M 个功能资源池.

对于资源池可能是由物理机资源池或虚拟机资源池来组成, 对于资源池中第 i 个资源池中第 j 个资源节点, 其物理特征主要了若干资源属性, 包括了 CPU、内存、存储容量、应用程序、运行状态、故障率等特征状态, 关于资源节点中各个特征, 其中 $CPU_{i,j}$ 包括了两个部分, 分别为份额值和处理器个数, 分别起到不同作用. 其中份额值主要功能是进行设定资源池中 CPU 大小, 处理器个数主要是用来提升其计算速度.

CPU 中取值来说, 可以设定不同取值, 其中取值设定为 0 话, 可以取得 2000 个 CPU 份额, 属于比较高份额; 此外, 还可以设定 1, 那么其取值份额为 1000 个, 属于正常范围; 例如, 在某一个虚拟主机上, 其 CPU 大小为 8GHz, 在上面运行两个应用, 如果设定虚拟机份额为正常话, 那么 CPUBi, j 取值是在[1, 8]范围.

此外, 对于资源节点上应用程序也可以进行配置, 应用程序进行资源节点统计, 应用程序种类集合中, 其中 q 表示是种类数量.

接着在资源池中容量、资源序列、内存容量、存储容量、应用程序容量可以通过如下表方式来表示, 具体如表 4.

表 4　源池中各个指标

功能	公式
某一个资源池中资源节点	$RP_i=\{VM_{i,1},VM_{i,2},VM_{i,3},...,VM_{i,}$
RP_i 所拥有资源序列 $RCS_{i,s}$	$RCS_{i,s}=\{R_{i,1},R_{i,2},R_{i,3},...,R_{i,s}\}$
RP_i 所拥有 CPU 总份额值 $RCPUA_{i,s}$	$RCPUA_{i,s}=\sum_{j=1}^{s}CPUA_{i,j}$
RP_i 所拥有内存容量 $RME_{i,s}$	$RME_{i,s}=\sum_{j=1}^{s}ME_{i,j}$
RP_i 所拥有存储容量 $RSC_{i,s}$	$RSC_{i,s}=\sum_{j=1}^{s}SC_{i,j}$
RP_i 所拥有应用程序容量 $RAPP_{i,s}$	$RAPP_{i,s}=\left\{s,\underbrace{0,0,...,0}_{q-1\text{个}0}\right\}$

一般来说, 对于资源节点来说, 如果其配置较好话, 那么将拥有较多应用程序、较多存储容量、较大 CPU 和内存, 在资源池中, 在众多资源节点中, 其最大配置表示为 MaxRi, s, 评判标准是首先是对应用程序进行评判, 接着是对 CPU 值比较, 最后是对内存和存储容量比较, 如果较多话, 那么其属于最大资源节点 MaxRi, s. 对于最小配置刚好与最大配置评判标准是相反, 首先比较是存储容量和内存, 然后是对 CPU 和应用程序种类, 如果比较结果是最小, 那么就属于最小配置.

按照上述对最大配置和最小配置评判标准, 设定某一个资源池其中有虚拟机数量为 s, 其配置具体如表 5 所示.

表 5　源池资源节点配置

资源节点	CPU(份额)	内存(MB)	存储(G)	应用程序
1, 2, 3	0	1024	25	OTYPE1
4	0	1024	30	OTYPE1
5, 6	1512	25	25	OTYPE1
7	2	256	15	OTYPE1
8, 9, …, s	2	256	20	OTYPE1

根据对最大资源配置与最小资源配置评判标准, 可以看到最大资源配置资源节点为 VM i, 4, 最小资源配置 VM i, 7. 接下来是对 M 个资源池中最大配置与最小配置计算, 可以通过如下集合来表示, 具体如下:

$$MaxR=Max\{MaxR_{1,x},MaxR_{2,x},...,MaxR_{m,x}\}$$

$$MinR=Min\{MinR_{1,x},MinR_{2,x},...,MinR_{m,x}\}$$

其中, x 为变量, 指第 m 个资源池所拥有虚拟机数量.

2.3.4 资源池负载均衡

在实训教学平台中, 对于不同实验任务其所需配置资源不尽相同, 为此, 需要对资源池中相关节点进行负载均衡, 使得其中不同节点利用率达到较高水准, 在本文研究中, 是基于由 vCenter 组件 DRS 来实现资源负载均衡, 通过对资源池中 CPU、内存、存储配置实现资源高效匹配与最大程度利用.

下面对资源池负载均衡介绍其实现过程, 主要是通过对 CPU、内存与存储预留值来实现对资源池负载均衡. 首先设定虚拟机在某一个时间段范围内对原子

任务执行, 那么第 k 个虚拟机 CPU 预留值、内存预留值、资源池中存储预留值可以表示如表 6 所示.

表 6 第 k 个虚拟机资源描述

项目	描述
CPU 预留值	$RCPU_{i,k} \le CPU_{i,k} - TCPU_{j,m}$
内存预留值	$RME_{i,k} \le ME_{i,k} - TME_{j,m}$
资源池中存储预留值	$RSC_{i,k} \le SC_{i,k} - TSC_{j,m}$

在云服务环境下通过 vCenter 组件 DRS 来对资源池中 CPU、内存、存储预留值设定, 使得可以虚拟机在满足需求情况下可以保证正常运行, 并且可以通过组件将其他资源动态迁移到尚未达到高峰利用率不高虚拟机上, 这样就可以在一定程度上使得资源池中相关应用和服务达到资源负载均衡.

前面对资源进行了定义, 根据对实验具体任务以及对任务分割方式, 对资源中资源进行划分, 通过划分实现资源集群, 在这个过程中主要是按照功能不同将其划分为若干功能资源池, 在划分后每一个资源池中都包括不同虚拟机或物理主机, 对于资源集群进行划分具体实现过程如下:

Step1. 按照操作系统类型将其划分为两个资源类型, 对于其中设定配置决定因子表示为 OSQ, 对于某一个资源池来说 i, 如果是 windows 类型, 那么中 i≤OSQ; 否则话, 就为中 i>OSQ;

Step2. 根据资源对某实验任务完成情况, 根据资源节点对功能任务不同划分为若干功能资源池;

Step3. 对上述划分功能资源池进行并行执行, 如果对某实验是并行执行话, 那么就可以得到特性功能资源池.

Step4. 继续对上述功能池进行划分得到若干子资源池.

前面按照功能类型实现对资源划分, 在进行资源划分中其遵循如下几个原则:

(1)经过划分之后某一个资源节点至少可以完成对某个原子任务执行.

(2)在资源池中资源如果具有并行执行功能, 那么其资源具有比较高功能级别, 并且如果并行程度越高话, 那么其功能级别就越高.

(3)在进行资源划分过程中, 对于 CPU 划分需要保证在同一个资源池中节点其 CPU 个数是一样.

(4)对于划分每一个资源节点中配置需要有一定比例关系, 如对于资源节点中 CPU 份额越多话, 那么所配置内存、存储需求也就越大.

(5)处于同一个资源池中节点之间可以实现负载, 如果出现空闲话, 那么就可以调度到其他节点中.

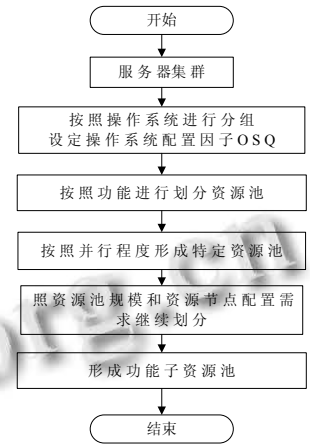

图 7 资源划分流程图

3 资源调度模型实现

下面对资源调度进行详细模型设计与实现.

3.1 任务分割编码

前面对实训教学平台中实验任务进行分割, 经过分割可以得到具有优先级别序列队列, 设定对于某个任务 T 来说, 经过分割之后可以得到 5 个子任务, 可以表示为:

$$\{T_{i,1}, T_{i,2}, T_{i,3}, T_{i,4}, T_{i,5},\}$$

在上述子任务序列中，其中第 2, 3 个任务与组合是无序任务集，可以表示为 $\{T_{i,2}, T_{i,3}\}$， 其中第 4, 5 个子任务是属于有序任务集可以表示为，$\{T_{i,1}, T_{i,4}, T_{i,5}\}$。其中对于无序任务集来说，其中第 2 个无序子任务可以进一步分割得到原子任务，分割得到 3 个原子任务 $T_{i,2'}$，有序任务集中第 1 个子任务可以分割得到 2 个并行原子任务，对于上述子任务在并行处理时候，其执行时间顺序表示为 $T_{i,5}$、$T_{i,1}$、$T_{i,4}$，具体如图 8 示。

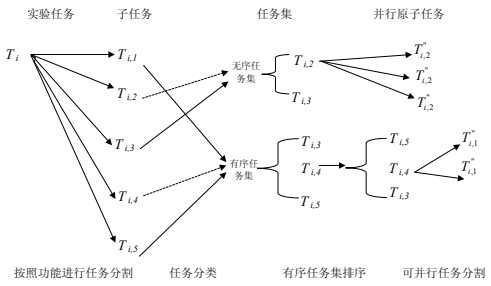

图 8 于实训教学平台所建立任务以及子任务模型图

具体任务分割集合可以表示为如表 7.

表 7　务分割集合统计表

对于任务来说划分成若干子任务	$T_i=\{T_{i,1}',T_{i,2}',T_{i,3},T_{i,4},T_{i,5}\}$
不同子任务之间并行处理	$TP_i=\{2,3,1,1,1\}$
具有一定序列任务集合	$TN_i=\{T_{i,5},T_{i,1}',T_{i,4},\}$
处于无序状态任务集合	$TU_i=\{T_{i,2}',T_{i,3}\}$

根据上述所建立各个任务在完成任务分割操作后建立无序和有序任务模型关系, 其具体如图 9.

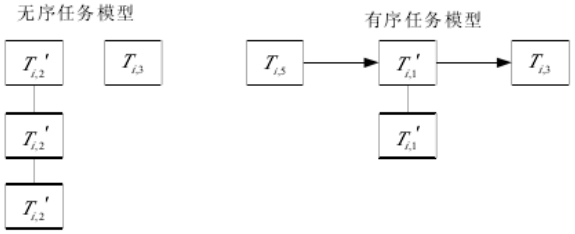

图 9　验任务有序和无序任务模型

接下来进行任务编码规则. 在对任务进行编码过程中主要有二个规则, 下面对任务编码规则进行介绍.

规则一: 对原子任务等级进行有理数规则编码, 其中原子任务时间序列可以通过有理数整数部分来表示, 在有理数编码如果整数部分是 0 话, 那么任务为无序原子任务, 在任何时间范围都可以启动, 如果大于 0 话, 表示植原子任务优先级别, 数值越大那么优先级别就高; 在有理数小数部分则表示是并行处理任务数量. 从上述对任务编码规则来看, 如果有理数是大于 1 话, 那么实验任务在运行过程是按照时间顺序来执行原子任务; 否则话, 在实验任务运行过程中, 则是随机选择执行.

通过上述对实验任务等级编码, 对实验任务 T 来说, 其编码结构示意图如图 10.

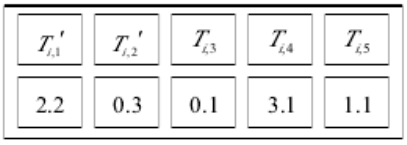

图 10　实验任务优先级编码

在某一个实验任务 T 中, 设定任务有 6 个子任务, 上述子任务可以表示如下:

$$T=\{T_1,T_2,T_3,T_4,T_5,T_6\}$$

在上述子任务集合中, 无序任务集为*{T1, T3}*，有序任务集表示为*{T5, T2, T4, T6}*, 在有序任务集中其中第 2 和第 4 任务是可以并行处理.

对实验任务中有序任务和有序任务其表示模型具体如图 11 所示.

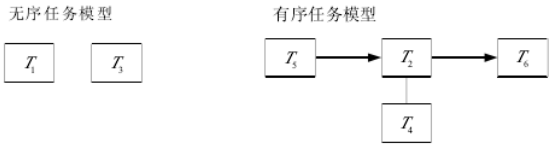

图 11　T 无序任务和有序任务模型

规则二: 在这个对实验任务分割规则中主要依据是对子任务进行有理数编码. 对子任务等级进行有理数规则编码, 其中子任务时间序列可以通过有理数整数部分来表示, 在有理数编码如果整数部分是 0 话, 那么任务为无序子任务, 在任何时间范围都可以启动, 如果大于 0 话, 表示植子任务优先级别, 数值越大那么优先级别就高; 在有理数小数部分则表示是并行处理任务数量.

从上述对任务编码规则来看, 如果有理数是大于 1 话, 那么实验子任务在运行过程是按照时间顺序来执行子任务; 否则话, 在实验子任务运行过程中, 则是随机选择执行.

通过上述对实验任务等级编码, 对实验任务 T 来说, 其编码结构示意图如图 12 所示.

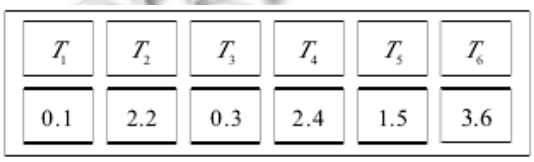

图 12　实验任务 T 优先级编码

3.2 资源池构建

① 对资源编码首先需要对资源池模型定义, 设定资源集群中构建了 7 个资源池为 $RP_i=(i\in[1,7]),\{RP_1,RP_2,RP_3,RP_4,RP_5,RP_6,RP_7\}$, 在上述资源集群中集合中具体包括资源节点数量分别为: {5, 7, 4, 5, 6, 2, 3}, 其中资源 2, 4, 6 所执行功能是一样.

前面资源池模型其结构具体如图 13 所示.

② 资源编码规则

规则三: 对资源池功能序列进行有理数规则编码,

其中功能顺序可以通过有理数整数部分来表示, 资源池序列编号则是通过有理数小数表示, 通过上述资源编码规则就可以将资源池呈现出来, 具有相同功能资源池, 具有相同有理数整数部分, 实现对资源池编码.

通过上述规则可以将资源集群中功能进行编码呈现, 具体如图 14 所示.

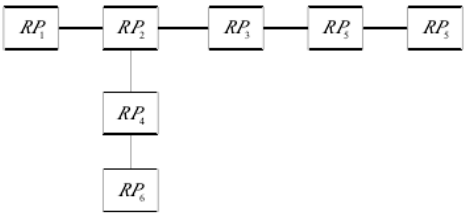

图 13 资源池模型图

RP_1	RP_2	RP_3	RP_4	RP_5	RP_6	RP_7
1.1	2.2	3.3	2.4	4.5	2.6	5.7

图 14 资源池功能等级编码

按照有理数对资源池编码规则, 可以将资源池 RPi 进行划分, 经过划分之后得到了 6 个资源队列, 可以表示如下:

$$\{VM_{i,1},VM_{i,2},VM_{i,3},VM_{i,4},VM_{i,5},VM_{i,6},\}$$

在上述表示资源池队列中, 其中第 2, 3, 4 资源节点可以实现执行相同功能并行处理资源节点.

经过对资源节点并行化处理之后, 就可以得到如下所示资源节点队列模型, 具体如图 15 所示.

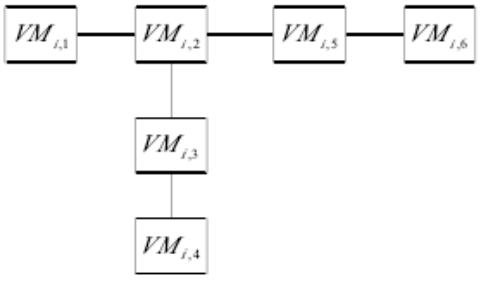

图 15 资源节点队列模型图

规则四: 对资源节点队列功能序列进行有理数规则编码, 其中功能顺序可以通过有理数整数部分来表示, 资源节点队列序列编号则是通过有理数小数表示, 通过上述资源编码规则就可以将资源节点队列呈现出来, 具有相同功能资源节点队列, 具有相同有理数整数部分, 实现对资源节点队列编码.

通过上述规则, 可以将资源集群中功能进行编码呈现, 具体如图 16 所示.

$VM_{i,1}$	$VM_{i,2}$	$VM_{i,3}$	$VM_{i,4}$	$VM_{i,5}$	$VM_{i,6}$
1.1	2.2	2.3	2.4	3.5	4.6

图 16 资源节点队列功能等级编码

3.3 实验资源调度实现

在云服务环境下对任务与资源调度中, 需要根据一定调度策略来实现, 通过调度完成对任务与资源高效匹配与映射, 使得实验任务可以分配到相应资源来完成, 为此, 在进行任务与资源调度时候需要遵循如下策略:

(1)最大匹配原则, 也就是说在对资源调度时候选择原子任务中并行程度比选定资源池中资源节点个数; 原子任务中并行程度最大需求和配置比功能相同能够执行并行任务资源节点配置小.

(2)在实训教学平台资源调度中, 如果选定某个资源比任务需求小无法得到满足时候, 进行资源搜索, 选择其中最邻近资源进行资源调度.

(3)在对实验任务分配资源过程中, 对操作系统选择与配置, 如果是仅仅满足 windows 操作系统环境话, 那么就从资源池中第一个资源开始分配; 如果是 linux 操作系统需求, 就从最后一个资源开始分配, 如果是混合性需求, 那么就设定决定因子 *OSQ*, 通过 *OSQ* 来实现对任务资源分配.

(4)在对实验任务分配资源过程中, 首要是满足应用程序邱, 首先是判断资源池中资源节点是否可以满足应用程序运行, 如果可以话, 接着在比较其他因素如 CPU 值等, 如果满足, 那么就选择资源节点作为调度对象, 将其调度到任务中进行原子任务执行.

(5)在实验任务与资源调度中, 对于其中任务来说, 如果其并行程度大于 1, 也就是说需要进行并行处理, 在开始执行前, 需要将满足需求资源节点进行处理, 使得其先处于等待任务执行状态, 在完成之后释放资源.

(6)在实验任务中根据优先级进行分配资源, 如果优先级为非 0 原子任务, 在执行资源调度时候, 只要

出现空闲资源就对其分配相应资源，分配完毕后实验任务执行无法占用；如果优先级为 0 原子任务，在执行资源调度时候，只要最邻近资源池资源有空闲时资源就对其分配相应资源，分配完毕后实验任务执行无法占用.

在云服务环境下，对实验任务与资源调度器实现过程具体如下，设定在实验任务 T 中经过分割之后得到有 *m* 个实验子任务，图 17 为资源调度流程图. 对上述实验子任务调度具体如下:

Step1. 根据实验任务分割子任务需求进行资源预分配，进行预分配可以表示为，

$$T_i = \{T_{i,1}, T_{i,2}, T_{i,3}, ..., T_{i,m}, \};$$

Step2. 云服务资源搜索，选最邻近资源池作为资源调度中资源池，具体表示为:

$$RP_j = \{VM_{j,1}, VM_{j,2}, VM_{j,3}, ..., VM_{j,k}, \};$$

Step3. 根据优先级任务调度. 在对任务进行编码基础上将其中整数部分为 1 子任务选作任务调度对象，将其选进到资源调度器中，并对其预分配资源，；接着对编码中小数部分调度处理，小数部分和该子任务应用程序需求选取最邻近资源池中节点来执行，并将资源节点状态重置为忙碌状态，在调度任务完成之后，资源释放后，资源云状态就转换为空闲；最后任务调度器中任务其生命周期为终止状态;

Step4. 继续对任务编码中整数部分为 1 任务进行调度，实现对资源预分配，按照 Step3 步骤进行资源调度;

Step5. 接下来对子任务进行资源调度，满足资源节点需求;

Step6. 在进行任务调度中，按照最邻近功能资源池中进行搜索，满足需求进行依次分配;

Step7. 按照上述步骤，对任务与资源进行调度，一直到全部原子任务都完成，处于终止状态，那么任务就完成.

在上述对任务与资源调度中，对于任务 Ti 执行所花费时间表示为 COST，其计算公式具体表示如下:

$$COST_i = \sum_{x=1}^{m} (TE_{i,x} - TS_{i,x})$$

此时，对于资源池中虚拟机 CPU 预留值可以表示如下:

$$PCPU_{i,k} = CPU_{i,k} - TCPU_{j,m}$$

4　仿真结果

前面对实训教学平台中资源调度进行设计，根据一定调度策略来实现对实验任务与实验资源高效匹配映射，为此，对实现资源调度进行性能测试，是否符合用户需求. 在对资源调度进行性能方面测试中，可以通过实训教学平台中具有代表性制图操作. 在实训教学平台中，用户在进行制图操作过程中需要较大资源消耗，如果数据量大话，制图操作需要花费较长时间，因此，利用实训教学平台资源调度来实现对制图操作中图片进行任务分割实现任务碎片化，通过调度相关计算资源以及内存资源来实现高效处理，可以在一定程度上降低用户等待处理时间，提高处理工作效率. 首先对制图实验进行分割处理之后得到具有可执行原子任务，分割得到 5 个原子任务，接着是对资源划分，其中功能资源池划分为 5 个，由 5 个服务器资源节点做构成. 首先是制图任务进行分割和优先级编码. 制图图片任务 T_1，分割后并行原子任务集:

$$T_1 = \{T_{1,1}, T_{1,2}, T_{1,3}, T_{1,4}, T_{1,5}\}$$

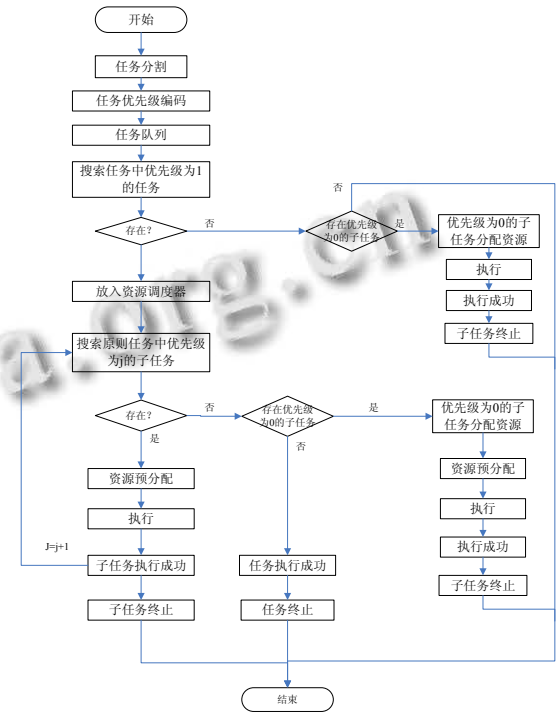

图 17　资源调度流程图

任务并行程度表示:

$$TP_1 = \{5\}$$

实验任务 T_1 原子任务队列模型:

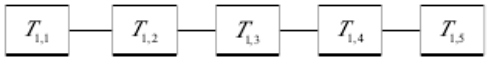

图 18 原子任务队列模型

优先级编码:

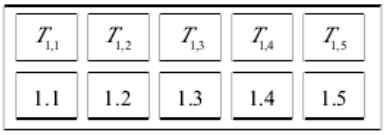

图 19 优先级编码

制图资源池 RP_1 拥有物理机资源表示:

$$RP_1=\{PM_{1,1},PM_{1,2},PM_{1,3},PM_{1,4},PM_{1,5}\}$$

资源池 RP_1 资源节点队列模型:

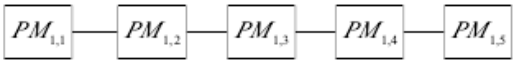

图 20 资源节点队列模型图

功能等级编码:

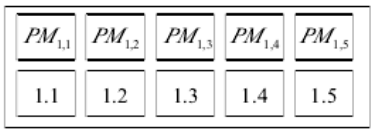

图 21 功能等级编码

接下来是资源调度. 资源调度通过调度策略实现制图任务和资源匹配, 具体模型图如 22.

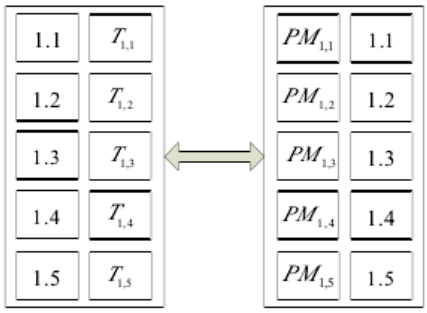

图 22 制图任务和资源匹配模型图

4.1 仿真环境

为了更好达到测试目标, 在本次测试中采取两种模式来对制图实验进行测试, 一种是通过单机模式来实现制图, 另外一种是通过集群模式来实现制图, 对于这两种制图实验与测试具体如下:

①单机制图模式实验环境配置具体如下:

表 8 单机制图模式实验环境配置

项目	配置
服务器	PowerEdge™ M1000e 模块化刀片，一共 1 片
每片 CPU	2xIntel Xeon E5CPU 处理器; 主频 2.40GHz，一共 6 核
每片内存	64GB
每片硬盘	465.25 GB*2
软件	Maya 2011
操作系统	Windows 2008 R2 Server Standard，64 位

② 集群制图模式实验环境配置具体如下.

表 9 单机制图模式实验环境配置

项目	配置
服务器	PowerEdge™ M1000e 模块化刀片，一共 5 片
每片 CPU	2xIntel Xeon E5CPU 处理器; 主频 2.40GHz，一共 6 核
每片内存	64GB
操作系统	VMware ESXi 5.1.0
	Windows 2008 R2 Server Standard，64 位
每片硬盘	465.25 GB*2
操作系统	VMware ESXi 5.1.0
	Windows 2008 R2 Server Standard，64 位

4.2 实验数据

(1)两种制图模式实验

图 23 为制图实验在不同计算模式下时间对比.

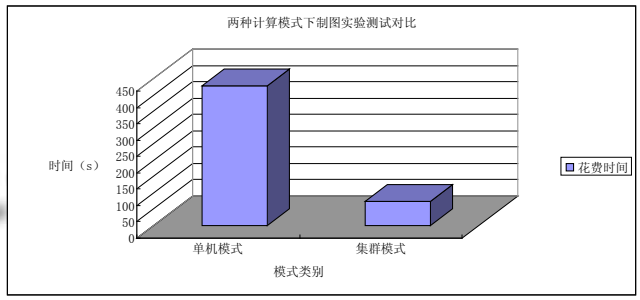

图 23 制图实验在不同计算模式下时间对比

在按照前面实验环境与配置实现两种制图实验测试, 经过实验测试, 在单机模式下, 完成上述制图图片所需要花费时间为 430s, 如果采取集群模式对前面同样图片进行制图实验所需要花费时间仅仅为 72s, 在制图工作效率上提高了大概 80%, 具有较高效率, 通过对制图实验进行任务分割, 使得实验任务可以并行处理, 可以极大提高实验执行速度.

(2)多任务调度性能实验

上面是对单任务情况下对制图实验进行性能方

面测试, 在云服务环境下实训教学平台中, 对于资源调度很多应用是在多用户多任务, 在此环境下资源调度直接关系到资源调度质量和性能, 为此, 需要设计在多用户多任务请求环境下, 对实验资源资源调度, 通过反复实验来验证资源调度性能.

在多任务资源调度中通过测试记录任务执行完成时间情况, 为此需要配置三种实验和环境, 在不同条件下进行制图图片实验, 设计三个制图图片任务, 经过实验得到三种不同任务其执行完成时间, 具体如图 24 所示.

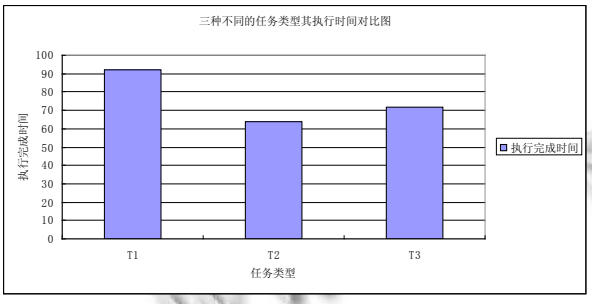

图 24　三种制图任务执行完成时间

三种制图其并行程度不同, 分割成为不同原子任务, 在相同资源节点下进行制图图片测试, 其中三种不同实验任务: 设计 T1 由 5 个可以并行执行原子任务组成, T2 则由 2 个, T3 由 3 个, 按照上述三种不同制图图片任务进行实验测试, 在不同并行程度下执行不同原子任务, 其执行完成时间是不尽相同. 接下来如果在不同调度策略下, 分别采取不同模式, 在传统模式和引入调度策略模式下查看任务执行情况, 并且记录任务执行完成时间.

根据前面对不同模式下任务执行时间对比中可以看到, 在引入了调度策略之后, 可以使得相同任务其花费时间较少, 可以有效提高任务执行效率; 随着任务增加, 在传统模式下任务执行完成时间呈现出线性增加趋势, 在引入调度策略模式下, 任务执行完成时间增长缓慢, 具体结果如图 25 所示.

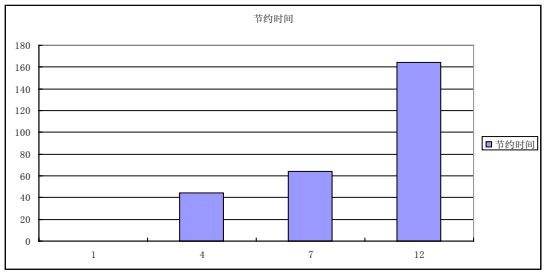

图 25　任务执行时间节约时间比较

因此, 比传统模式时间更加节约. 这是由于, 在引入调度策略模式下, 可以对任务进行分割从而得到具有不同优先级别原子任务, 可以分配得到具有功能等级资源编码, 实现高效任务与资源匹配映射, 可以有效降低任务执行时间.

4.3 结果与分析

传统模式和引入调度策略模式下任务执行时间对比结果具体如图 26 所示.

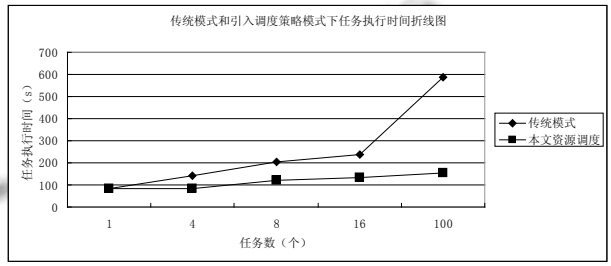

图 26　传统模式和引入调度策略模式下执行时间折线图

除了任务执行完成时间, 对实训教学平台 CPU 利用率经过采集和统计后, 分析在不同模式下完成制图实验测试所花费计算资源, 其结果具体如图 27 所示和图 28 所示.

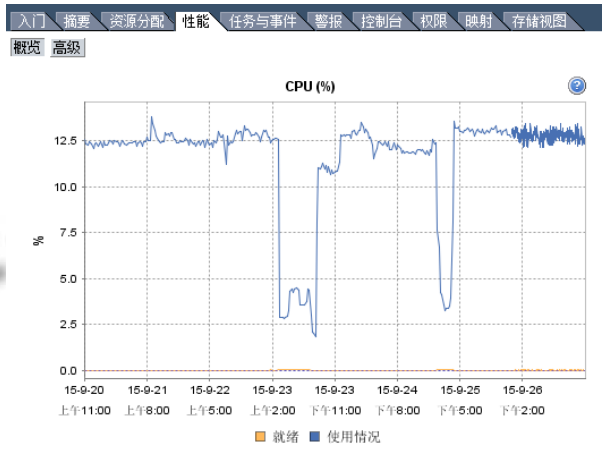

图 27　传统模式资源池 CPU 利用率

首先是在传统模式下 CPU 利用率, 在这种模式下对实训教学平台制图, 在完成同样制图效果所花费计算资源不多, 导致其利用率不高, 花费时间较多, 无法实现并行计算. 在引入了资源调度策略后, 可以实现并行计算与处理, 能有效调度相关资源, 使得资源利用率有效提高, 因此, 整体 CPU 利用率比传统模式有了较明显提高, 具体如图 28 所示.

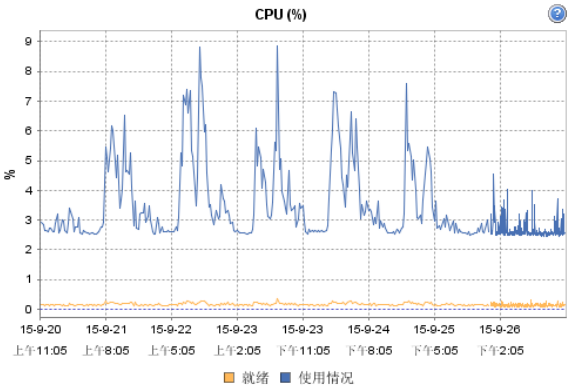

图 28 引入调度策略模式资源池 CPU 利用率

前面对两种模式下任务执行时间与整体 CPU 利用率进行对比实验测试, 在不同对资源调度策略下进行制图实验处理, 并对结果进行分析, 验证了资源调度策略有效性与可行性.

5 结束语

针对传统实训教学平台资源调度策略不足与缺陷, 对实训教学平台动态迁移过程中需要解决技术问题进行分析, 提出实训教学平台动态迁移策略中任务分割模型和资源划分模型, 对资源调度关键技术进行研究, 设计了任务分割、资源划分和任务、资源编码方式, 对分割后任务进行优先级编码、划分后资源进行功能等级编码, 实现任务与资源高效匹配与映射, 设计了云服务环境下实训教学平台动态迁移模型, 对系统进行仿真实验, 从对两种模式下任务执行时间与整体 CPU 利用率进行对比实验测试, 验证所提出资源调度策略可行性.

参考文献

1 刘筱兰,张薇,程惠华等.实训教学平台室类型及发展趋势.计算机应用研究,2015,43(8):149−155.

2 Shu WN, Wang JQ. Min-Min chromosome genetic algorithm for load balancing in grid computing. International Journal of Distributed Sensor Networks, 2014(1): 73−79.

3 柴亚辉,涂春萍,刘觉夫等.基于云计算计算机与软件实验资源管理.实验室研究与探索,2015,26(2):177−184.

4 He XS, Sun XH, Laszewski G. QoS guided Min-Min heuristic for grid task scheduling. Journal of Computer Science and Technology, 2014, (4)

5 陈传波,张立峰,陈南平.一个远程实训教学平台教学平台研究.华中科技大学学报(自然科学版),2014,46(12):157−163.

6 廖大强,邹杜,印鉴.一种基于优先级网格调度算法.计算机工程,2014,40(10):11−16.

7 黄晨晖,林泳琴.基于云计算虚拟计算机实验室研究与实现.实验室研究与探索,2013,27(11):69−74.

8 廖大强.云计算环境下高职院校教学资源有效应用研究.电子测试,2015,17:80−81.

9 董秋生,黄文,马骏涛等.服务器虚拟化技术在实验室服务器整合中应用.计算机与网络,2014,35(1):76−79.

10 廖大强.云计算在高职院校实验教学服务中分析与应用.数字技术与应用,2015(6):199−200.

11 康辰,朱志祥.基于云计算技术网络攻防实验平台.西安邮电大学学报,2014,49(3):89−94.