Journal of System Simulation

Volume 28 | Issue 3 Article 7

7-2-2020

QoS-aware Scheduling for Data Intensive Workflow

Wan Cong
College of Information Science and Engineering, Northeastern University, Shenyang 110004, China;

Cuirong Wang
College of Information Science and Engineering, Northeastern University, Shenyang 110004, China;

Wang Cong
College of Information Science and Engineering, Northeastern University, Shenyang 110004, China;

QoS-aware Scheduling for Data Intensive Workflow

Abstract

Abstract: The development of technology enables people to access resources from different data centers. Resource management and scheduling of applications, such as workflow, that are deployed on the cloud computing environment have already become a hot spot. A QoS-aware scheduling algorithm for data intensive workflow on multiple data center environment was proposed. Scheduling data intensive workflow on multiple data center environment has two characteristics: A large amount of data is distributed in different geographical locations, the process of data migration will consume a large amount of time and bandwidth; secondly, the data centers have different price and resources. *Data migration between data centers was mapped to a task of workflow*, models workflow with DAG, simplifying the DAG, and scheduling all the tasks of workflow using Simulated Annealing. *The experiment using CloudSim platform and Hadoop platform shows that this scheduling algorithm is effective*.

Keywords

workflow, big data, cloud computing, schedule

Recommended Citation

Wan Cong, Wang Cuirong, Wang Cong. QoS-aware Scheduling for Data Intensive Workflow[J]. Journal of System Simulation, 2016, 28(3): 549-558.

第 28 卷第 3 期 系统仿真学报© Vol. 28 No. 3
2016 年 3 月 Journal of System Simulation Mar., 2016

面向数据密集型工作流的 QoS-aware 调度算法

万聪，王翠荣，王聪
(东北大学 信息科学与工程学院，辽宁 沈阳 110004)

摘要：随着技术的发展，人们可以从不同的数据中心获得资源。跨数据中心的数据密集型工作流任务调度成为了一个热点问题。提出了一个在多数据中心环境下的数据密集型工作流任务调度算法。多数据中心环境中的数据密集型工作流计算有 2 个特点：(1)数据量大，而且分布在不同的地理位置，数据迁移的过程会消耗大量的时间和带宽；(2)数据中心具有异构性，提供资源的数量、种类和价格不同。针对这些特点，*算法将数据迁移的过程映射为一个数据迁移任务*，使用有向无环图(DAG)对工作流进行建模，对 DAG 进行了化简。算法利用模拟退火算法，将工作流执行时间和花费作为优化目标，计算出一个优化的调度方案。*分别使用 CloudSim 平台和 Hadoop 平台对算法进行了验证。*

关键词：工作流；云计算；大数据；调度
中图分类号：TP391 文献标识码：A 文章编号：1004-731X (2016) 03-0549-10

QoS-aware Scheduling for Data Intensive Workflow

Wan Cong, *Wang Cuirong*, *Wang Cong*
(College of Information Science and Engineering, Northeastern University, Shenyang 110004, China)

Abstract: The development of technology enables people to access resources from different data centers. Resource management and scheduling of applications, such as workflow, that are deployed on the cloud computing environment have already become a hot spot. A QoS-aware scheduling algorithm for data intensive workflow on multiple data center environment was proposed. Scheduling data intensive workflow on multiple data center environment has two characteristics: A large amount of data is distributed in different geographical locations, the process of data migration will consume a large amount of time and bandwidth; secondly, the data centers have different price and resources. *Data migration between data centers was mapped to a task of workflow*, models workflow with DAG, simplifying the DAG, and scheduling all the tasks of workflow using Simulated Annealing. *The experiment using CloudSim platform and Hadoop platform shows that this scheduling algorithm is effective*.

Keywords: workflow; big data; cloud computing; schedule

引言

随着软件，硬件和网络的发展，云计算正在成为最流行的研究热点。很多政府部门，研究机构和公司开始建立数据中心来提供云计算服务。很多领域的应用程序，例如数据挖掘[1]，实验仿真[2]甚至于军事训练[3]都被部署云上，其中一些应用程序是由多个子任务组成的，这些子任务的执行顺序取决于他们之间的控制与数据依赖。举个例子，很多社会网络网站(social networking sites，例如人人网)共享了他们的数据，所以跨越他们的数据中心的查

收稿日期：2014-07-15 修回日期：2014-10-23；
基金项目：国家自然科学基金(61300195)；
作者简介：万聪(1983-)，男，河北秦皇岛，讲师，博士生，研究方向为并行计算、云计算中调度问题；王翠荣(1963-)，女，河北迁安，教授，博士，研究方向为大数据处理、网络虚拟化。

询变的很常见。因此，一个云计算下的工作流管理系统是必须的。最近，一些应用于云计算中工作流管理系统和工作流调度算法被开发出来，例如基于 Hadoop 框架的 Nova[4] 和 GreenPipe[5]，还有一些工作流管理系统，例如 G-Hadoop[6] and Pegasus[7]，是应用于跨数据中心的数据流调度，但是这些系统缺乏对于服务质量的关注。对于数据密集型应用来说，数据的放置是一个关键问题。很多研究者关注于数据的备份[8]和不同数据之间的关系[9-10]，但是他们忽略了子任务之间的关系。

G-MR[11]是一个能够在地理分布的数据集上执行 MapReduce[12]任务的系统，该系统以运行时间和花费作为优化目标，计算出一串 MapReduce 任务的执行路径。LOSS 和 GAIN[13]算法以预算为约束条件，在分布式的计算环境中对工作流进行调度。在文献[14]中，研究者们提出了一个多约束条件下的科学工作流调度框架。在文献[15]中，研究者们关注于子任务的特异性。在文献[16]中，研究者们提出了一个任务备份的方案来提高系统的可靠性。

云计算的一个重要特点就是将地理位置分布的(Geodistribution)资源统一管理。很多部署在云平台上的应用也呈现出地理位置分布的趋势，特别是数据密集型应用，这一趋势更加明显。数据被部署在地理位置分布的多个数据中心，而不是过去的单一数据中心，原因可以总结为以下几点：

1. 基于用户分布的。将应用的使用者按照地理位置进行聚类，每个用户的数据都被保存在距离自己最近的数据中心。这样可以降低由距离带来的延时和网络消耗，常见于访问量较大的门户网站、大型多人在线网络游戏等等。

2. 基于所有者分布的。为了进行科学计算或者数据分析，经常需要使用多个数据源。这些数据源可能分别归属于不同的机构，出于安全性、隐私性或者便利性等方面的考虑，它们会被保存在机构自身搭建的私有云中。而这些私有云，往往是地理位置分布的。

3. 基于备份的。为了提高可用性，数据可能在多个数据中心进行备份。

本文首先描述了在多数据中心环境下，用户提交工作流的流程，将工作流中数据迁移的过程映射为一个子任务，又使用有向无环图对工作流进行建模并化简。本文还提出了一个调度算法(workflow on multiple data center environment algorithm，WMDCA)，算法利用模拟退火算法，将工作流执行时间和花费作为优化目标，计算出一个优化的调度方案。

1 架构

很多云提供者将存储、计算或者数据封装成服务供用户使用。在云提供者为地理位置分布的数据密集型应用提供服务的过程中，有两种比较典型的方式：第一，将带宽、内存、计算能力等资源封装成一些标准化的虚拟机，例如，亚马逊公司的 EC2；第二，建立并行计算的框架，对提供对外接口，用户可以上传自己的并行计算程序，例如 Hadoop。

在多数据中心环境下云平台提供服务的框架中，用户获得服务的流程如下所示：

1. 用户将任务提交到一个控制中心，提交的信息包括任务的描述，子任务的描述，对数据的需求情况等等。

2. 控制中心向能够提供资源的数据中心发出询问信息，收集各个数据中心的计算能力，收费等等信息。

3. 控制中心执行调度算法，根据任务信息和数据中心信息计算出一个调度方案。

4. 控制中心请求数据中心为子任务创建运行环境，并将子任务需要的数据迁移到制定的数据中心。

5. 数据中心执行子任务，并将输出的数据暂时保存在本地，等待控制中心的进一步指令。

6. 控制中心依次将所有子任务都执行完毕之后，将任务运行结果返回给用户，任务完成。

2 问题描述与建模

在多数据中心环境下，工作流的调度问题可以

用一个四元组<DC, T, D, A>来描述。四元组的定义如下所示：

定义 1 数据中心。用 DC={dc_i}来表示数据中心，数据中心提供计算能力作为服务，并对计算和网络使用计费。其中 dc_i=(cap_i, $band_i$, p_i, pb_i)，cap_i 代表数据中心提供的单个计算单元的计算能力，计算单元的可以是虚拟机，也可以是 Hadoop 框架中的槽位(Slot)，计算单元是数据中心提供资源的最小单位，用户可以一次申请多个计算单元；$Band_i$ 表示数据中心提供的带宽；p_i 表示计算单元的价格；pb_i 表示带宽的价格。

定义 2 工作流中的任务。用 T={t_i}来表示在工作流中运行的任务集合。每个任务的运行都需要若干数据集作为输入，执行结束后会输出若干数据集。其中 t_i=(dpre, dlast, workload, N)，dpre 代表任务 t_i 的前置任务集合，任务 t_i 需要 d_{pre} 中任务输出的数据集作为输入；dlast 代表任务 t_i 的后置任务集合，dlast 中的任务需要 t_i 输出的数据集作为输入；workload 代表任务的工作量；N 代表作业的并行程度，表示任务可以被分解为多少个并行处理的作业。

定义 3 数据集。用 D={d_i}来表示工作流运行过程中使用的数据集。数据集分为 2 类：第 1 类是原始数据集，在工作流建立之前就已经存在，被保存在某个数据中心；第 2 类是中间数据集，表示在工作流工作过程中被生产出来的数据。数据集 d_i=($length_i$, $position_i$)，其中 $length_i$ 表示数据集的大小；$position_i$ 表示数据集的位置。

定义 4 调度方案。用矩阵 ***A***=[assignij]来表示调度方案。如果子任务 t_i 被调度到数据中心 d_j，那么 $assign_{ij}$ 被赋值为 1，否则为 0。

$$\text{assign}_{ij}=\begin{cases}1 & \text{task}_i \text{ is assigned to } dc_j\\ 0 & \text{otherwise}\end{cases}$$

图 1 给出了一个工作流的例子。图中的 d_1, d_2, d_3, d_4 和 d_5 表示 5 个数据集，其中 d_1 为原始数据集，d_2~d_5 为中间数据集。图中 t_1, t_2, t_3 和 t_4 代表工作流中的 4 个子任务。图中的箭头表示任务和数据之间的依赖关系。例如，数据集 d_2 由任务 t_1 产生，任务 t_2 和 t_3 的执行需要依赖于 d_2，所以 t_2 和 t_3 必须等到 t_1 执行结束才能开始。

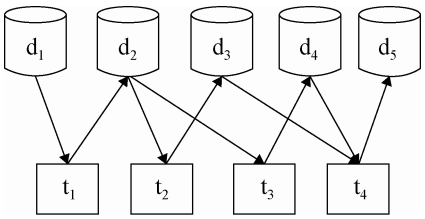


图 1 工作流示例图

假设有 3 个数据中心 dc_1，dc_2 和 dc_3，任务的调度方案如公式(1)所示。这表示任务 t_1 和 t_2 被调度到数据中心 dc_1，任务 t_3 被调度到数据中心 dc_2，任务 t_4 被调度到数据中心 dc_3。假设原始数据集 d_1 的初始位置在 dc_2。

$$A=\begin{bmatrix}1 & 0 & 0\\ 1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 1\end{bmatrix} \tag{1}$$

那么，如图 2 中的工作流执行过程如下：

1. 首先将数据集 d_1 由数据中心 dc_2 移动到数据中心 dc_1，然后开始执行 t_1，t_1 运行结束后产生的数据集 d_2 被直接保存在 dc_1。

2. 开始在 dc_1 执行任务 t_2，同时将数据集 d_2 移动到数据中心 dc_2。T_2 运行结束后产生的数据集 d_3 被直接保存在 dc_1。

3. 将数据集 d_3 移动到数据中心 dc_2 和 dc_3，等到数据集 d_2 和 d_3 都移动完毕，开始在 dc_2 上执行任务 t_3，t_3 运行结束后产生的数据集 d_4 被保存在 dc_2 中。

4. 将数据集 d_4 移动到数据中心 dc_3，等到数据集 d_3 和 d_4 都移动完毕，开始在 dc_3 上执行任务 t_4 ，t_4 运行结束后产生的数据集 d_4 被保存在 dc_3 中。

从上例可以看出，很多因素都会对工作流任务调度方案的成本造成影响：

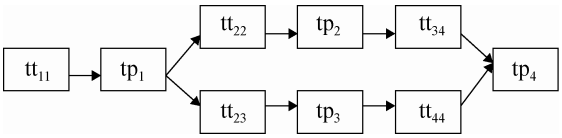
1. 数据集的大小。数据的移动需要占用带宽并花费时间，

2. 数据集和任务间的依赖关系。数据集和任

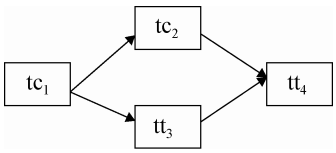
务之间可能存在多对多的关系

3. 数据中心的计算能力和定价方式。

为了在工作流执行过程中，保证一个较低的成本，调度算法应当综合考虑数量量、数据集和任务的关系(即工作流的结构)和数据中心的计算能力和定价方式。



(a) 化简前 DAG 图



(b) 化简后 DAG 图

图 2 工作流的 DAG 图

3 算法说明

为了更加清晰的表述问题，本算法将数据移动的过程映射为一个数据迁移任务，将原有的任务定义为运算任务。

定义 5 数据迁移任务，用 tt(task of transform)来表示一个数据迁移任务。如果数据 d_i 和任务 t_j 之间存在依赖关系，那么存在一个映射<d_i,t_j> tt_{ij}。Tt_{ij}=(dpre, dlast, length, source，dest)，其中 dpre 代表 tt 的前置任务集合，dpre 是运算任务的集合或者为空集；dlast 代表 tt 的后置任务集合，dlast 是一个运算任务的集合；length 表示数据集的大小；source 表示数据集初始的位置；dest 表示数据集要被移动到的数据中心。

将任务 tt 调度到某个数据中心就代表着将 tt 移动到某个数据中心，当 source=dest 时代表数据集不需要移动。

定义 6 运算任务，用 tp(task of process)来表示一个运算任务。运算任务是工作流中用户定义的原有任务，例如前例中 t_1 和 t_2。Tp=(dpre, dlast, workload, N)，其中 dpre 代表 tp 的前置任务集合，dpre 是一个 tt 的集合；dlast 是 tp 的后置任务集合，dlast 是一个 tt 的集合或者空集；workload 和 N 的含义与定义 6.2 一样。

将数据 d_i 和任务 t_j 之间的依赖关系用序偶<d_i,t_j>表示，对图 1 中的工作流进行整理，可以得到所有的数据迁移任务如表 1 所示。

表 1 数据迁移任务生成表

序偶	数据迁移任务	序偶	数据迁移任务
<d_1,t_1>	tt_{11}	<d_3,t_4>	tt_{34}
<d_2,t_2>	tt_{22}	<d_4,t_4>	tt_{44}
<d_2,t_3>	tt_{23}		

经过映射后得到了一个有向无环图(DAG)，结果如图 2(a)所示。

在调度过程中，每个运算任务和它的所有前置任务都必须被调度到相同的数据中心。因此，可以对图 2(a)中的 DAG 进行化简，化简的规则如下：

1. 每个运算任务 tp 与它的前置任务集合进行合并，合并任务用 tc 来表示；

2. 运算任务的所有后置任务都成为合并任务的后置任务；

3. 数据迁移任务的所有前置任务都成为合并任务的前置任务。

tp_1 将与它的前置任务 tt_{11} 合并；tp_2 与它的前置任务 tt_{22} 合并；tp_3 与它的前置任务 tt_{23} 合并；tp_4 与它的两个前置任务 tt_{34} 和 tt_{44} 合并。合并化简后的效果如图 2(b)所示。

用 $transTime_i$ 与 $transCost_i$ 分别表示一个 tt 任务花费的时间和钱数。它们的计算方式如(2)(3)所示。如果数据集的初始数据中心和目的数据中心相同，那么传输时间为 0，否则传输时间等于数据集的大小除以 2 个数据中心之间的网络带宽。网络带宽是初始数据中心带宽和目的数据中心带宽中较小的一个。传输费用等于传输时间乘以 2 个数据中心的网络使用费用。

$$transTime_i = \begin{cases} 0 & assign_{iposition} = 1 \\ \dfrac{length_i}{\min(band_{position}, \sum_{j=1}^{n} assign_{ij} \times band_j)} & otherwise \end{cases} \quad (2)$$

$$\text{transCost}_i = \text{transTime}_i \times (pb_{\text{position}} + \sum_{j=1}^{n} \text{assign}_{ij} \times pb_j) \quad (3)$$

用 proTime_i 与 proCost_i 分别表示一个 tp 任务花费的时和钱数。它们的计算方式如(4)(5)所示。计算任务的工作时间等于工作量除以数据中心单个计算单元的计算能力，再除以任务的并行度。花费等于工作时间乘以价格再乘以并行度。

$$\text{proTime}_i = \frac{\text{workload}_i}{N \times \sum_{j=1}^{n} \text{assign}_{ij} \times \text{cap}_j} \quad (4)$$

$$\text{proCost}_i = \frac{\text{workload}_i \times \sum_{j=1}^{n} \text{assign}_{ij} \times p_j}{\sum_{j=1}^{n} \text{assign}_{ij} \times \text{cap}_j} \quad (5)$$

用 cTime_i 与 cCost_i 分别表示一个 tc 任务花费的时间和钱数。它们的计算方式如(6)(7)所示。合并任务的工作时间=所有数据移动任务的时间+一个运算任务的时间，其中数据移动任务可以并行完成，所以以最大值为准。花费=所有数据移动任务的花费+运算任务的花费。

$$\text{cCost}_i = \sum \text{transCost}_j + \text{proCost}_i \quad (6)$$

$$\text{cTime}_i = \text{MAX}(\text{transTime}_j) + \text{proTime}_i \quad (7)$$

最后，用 Time 与 Cost 分别表示整个工作流花费的时间和钱数。它们的计算方式如(8)(9)所示。花费等于组成工作流的所有 tc 任务的花费之和。计算时间首先要求用 tc 任务的时间作为工作流 DAG 的节点权重，再求出 DAG 的关键路径，最后关键路径中节点权重的和就是整个工作流花费的时间。

$$\text{cost} = \sum \text{cCost} \quad (8)$$

$$\text{time} = \sum_{i \in \text{criticlePath}} \text{cTime}_i \quad (9)$$

算法的优化目标 U 的定义如(10)所示：

$$U = \text{time} \times \text{cost} \quad (10)$$

本问题是一个在分布式环境下进行工作流调度的问题，经过本文的建模，归结为一个时间和花费钱数的多目标优化问题。由于工作流调度问题的复杂性[17-18]，很多研究都采用了启发式算法。而且在解决多目标优化问题时，启发式算法也是常用的手段[19-20]。在本文中采用模拟退火算法来解决这一问题。本文中用到的模拟退火算法如下所述：

算法 1. 模拟退火算法

输入：temperature T; end temperature T_min; number of iterations L;DAG of workflow G; cooling coefficient α

输出：A

```
1 generate a initial A;  U=utility(A,G);
2 While T>T_min
3     While i<L
4         tmpA=getNeighbour(A);
5         tmpU=utility(tmpA,G);
6         IF tmpU<U
7             A=tmpA; U=tmpU;
8         End IF
9         Else
10            IF exp((U-tmpU)/T)>random;
11                A=tmpA; U=tmpU;
12            End IF
13        End Else
14        i++;
15    End While
16    T = T *α;
17 End While
```

第 1 步(第 1 行)，设置初始温度 T、终止温度 T_min、每个温度的迭代次数 L 和温度衰减系数 α。初始化调度方案 A 如(11)所示。根据公式(10)使用函数 utility(A，G)计算 U 的初始值，其中 G 代表用 DAG 建模的工作流。

$$\boldsymbol{A} = \begin{pmatrix} 1\,0 \cdots 0 \\ \vdots \ddots \vdots \\ 1\,0 \cdots 0 \end{pmatrix} \quad (11)$$

第 2 步(第 4~5 行)，使用函数 getNeighbour(A)生成一个 A 的邻居矩阵 tmpA。该函数随机选择 A

中的一行，在该行中随机选择一个元素置为 1，该行其他元素置为 0。计算 tmpU = utility(tmpA，G)。

第 3 步(第 6~13 行)，如果 tmpU<U，那么接受 tmpA 作为当前解，设置 U=tmpU，A=tmpA。如果 tmpU>U，那么将根据 Metropolis 接受准则来决定是否接受 tmpA。

第 4 步(第 14~15 行)，迭代次数加 1，如果当前温度的迭代结束则进入第五步，否则返回第 2 步。

第 5 步(第 16~17 行)，降低温度 T，如果 T<T_min 则结束算法，否则回到第 2 步。

本文的算法伪代码如下所示，代码一共分为了 3 个阶段，第 1~15 行为第 1 阶段，将每个数据集都映射为一个数据迁移任务，将工作流表示为如图 2(a)所示的有向无环图；代码第 16~29 行为第 2 阶段，对任务进行合并，将第一阶段的有向无环图进行化简，得到一个如图 2(b)所示的新图；第 30 行为第 3 阶段，调用算法 1 计算出一个调度方案。

算法 2. WDCMA 算法

输入：data center set DC={dc_i}; task set T={t_i}; data set D={d_i}

输出：A

1 For each t_i in T
2 generate tp_i for t_i ; // tp_i has the same attributes as task t_i
3 For each d_x in tp_i. d_{pre} ; // d_x is the data that task depends on
4 IF d_x in $t_j.d_{last}$; // if d_x is generate by the other task
5 generate tt_{ji} for d_x ; // d_x may be mapped to multiple tt
6 tt_{ji}.pertask = tp_i ;
7 tt_{ji}.lasttask =tp_j ;
8 End IF
9 Else // d_x exist before all task processed
10 generate tt_{ij} for d_x ;
11 tt_{ij}.pertask = tp_i ;
12 tt_{ij}.lasttask =tp_j ;
13 End Else
14 End For
15 End for
16 For each tp_i
17 generate tc_i for tp_i and tp_i.pretask ;
18 For each tt_x in tp_i.pretask ;
19 set previous task of tt_x as one of previous task of tc_i ;
20 End For
21 For each tt_x in tp_i.lasttask ;
22 IF tt_x has no last task
23 set tt_x as a part of tc_i ;
24 End IF
25 Else
26 set tt_x as one of previous task of tc_i ;
27 End Else
28 End For
29 End For
30 A=SA(tc);

4 实验

为了验证本文提出的算法，实验被分成了 2 个部分：(1)使用 CloudSim[21]环境进行了仿真实验; (2)使用 Hadoop 框架进行了真实环境的实验。

CloudSim 是由澳大利亚墨尔本大学开发的一款云计算仿真软件，可以对云环境中的资源分配和任务调度算法进行仿真实验。CloudSim 可以模拟云计算中涉及到的各种组件，如：

Datacenter 类模拟了一个数据中心，管理了一系列物理主机，每一台主机的配置都可以由用户进行设置。Datacenter 类以虚拟机的形式提供计算资源，处理虚拟机信息查询，负责虚拟机的创建和销毁。

VmAllocationPolicy 类封装了虚拟机的分配策略。当 Datacenter 类收到虚拟机请求时，会根据

VmAllocationPolicy 类中的策略来选择适合的物理主机来创建虚拟机。开发者可以根据特定的需求来自定义 VmAllocationPolicy 类中的分配策略。

CIS(Cloud Information Service)模拟了一个云计算平台信息服务中心，随着仿真的开始会自动创建。该类扮演了仿真平台服务黄页的角色，提供了数据中心的注册、查询功能。

DatacenterBorker 类可以表示使用云计算平台的用户。Borker 管理着一个用户的所有任务，它会为每个任务申请计算资源，管理任务的生命周期，直到所有的任务都被完成。

在 CloudSim 中进行仿真首先要建立数据中心并在 CIS 注册；当用户到来时，从 CIS 获得数据中心名单，然后选取一个数据中心创建虚拟机，提交任务；数据中心在创建虚拟机时，需要遵循虚拟机分配策略。

CloudSim 是一个开源的工具，为了让它可以模拟多数据中心环境下工作流的运行情况，本文对它进行了一些修改并添加了一些组件：

1. 创建了一个工作流生成器，工作流生成器可以按照配置文件，生成一系列具有依赖关系的任务。

2. 将 DatacenterBorker 分为了任务调度器和工作流执行引擎 2 个部分。调度器根据数据中心信息和任务信息制定一个调度方案，工作流执行引擎根据调度方案将任务提交到数据中心。另外，DatacenterBorker 还要记录完成任务的花费情况。

3. 本文假设在实际运行中，数据中心总是有充足的资源满足用户的需求。根据这一假设，重写了 Datacenter 类，不设置物理主机的数量，但是在创建虚拟机时，总有可用的虚拟机。

经过修改后，一个工作流的运行过程如图 3 所示：首先，工作流生成器会生成一个工作流，提交给任务调度器，然后任务调度器查询数据中心信息，生成一个调度方案，并将调度方案提供给工作流引擎。工作流引擎按照调度方案申请虚拟机并将工作流的子任务提交给数据中心，如果子任务可以并行运行，工作流引擎会同时申请多个虚拟机。

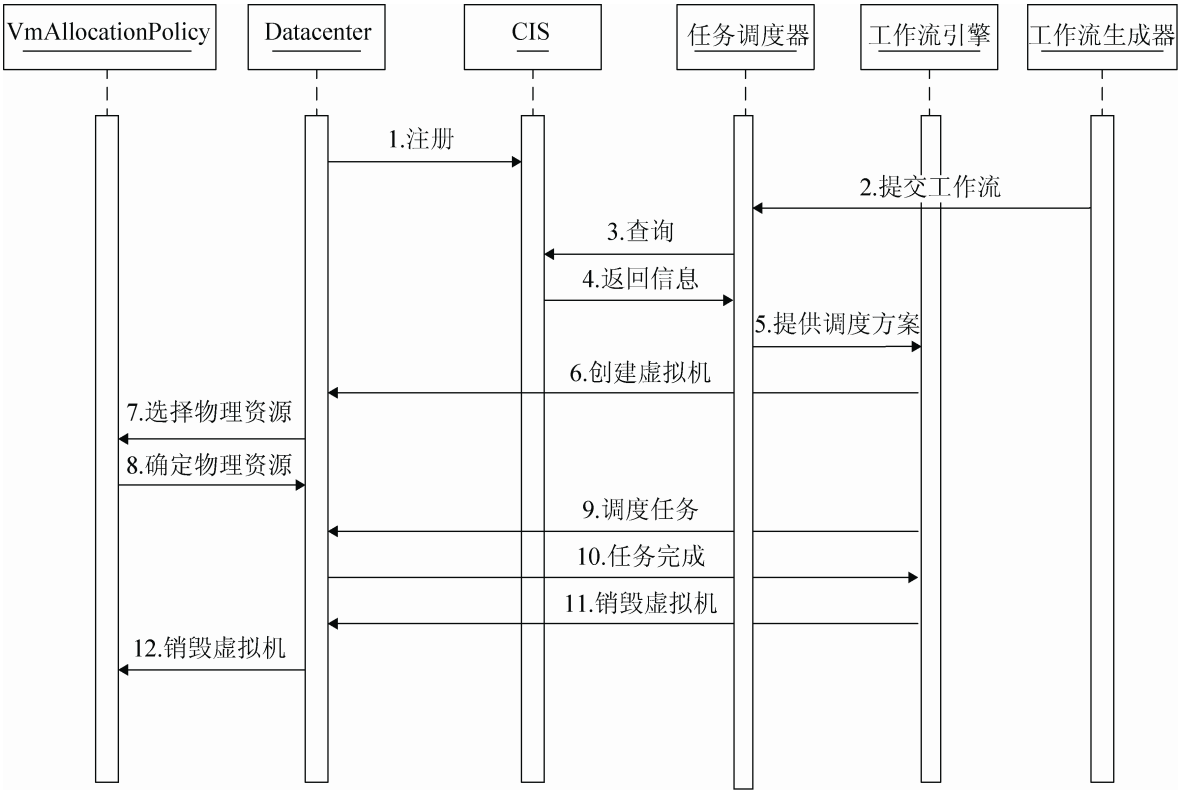


图 3 修改后的 CloudSim 工作流程图

仿真软件的修改和编译使用了 JDK1.6 和 eclipse 开发环境，运行的操作系统为 windows xp sp3，运行的硬件配置为 2GB 内存，2GHz 的双核处理器。

为了保证实验的可信性，每一个实验场景都运行了 30 次，取 30 次运行结果的平均值作为最终结果。为了说明算法的效果，本文选择 Rondom[22] 和 Greedy-data[18] 2 种任务调度策略与本算法进行了对比。Rondom 策略随机的将子任务调度到一个数据中心；Greedy-data 选择数据迁移量最小的数据中心。

首先设置数据中心的数量为 5 个，实验中每个工作流包含 10 个子任务，子任务之间的依赖关系为随机生成。在图 4~5 中显示，随着每个子任务工作量的增加，本算法的工作流执行时间和花费也明显低于对比算法。

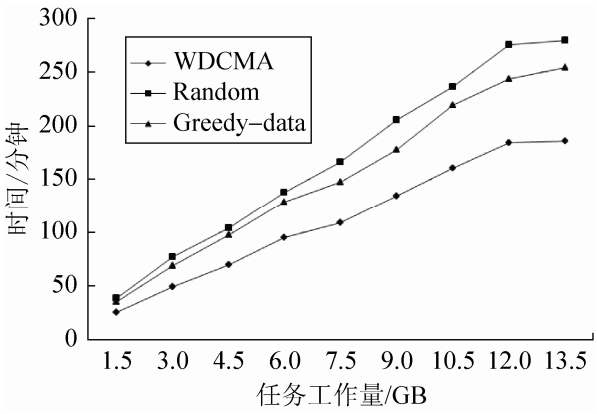


图 4 任务工作量对运行时间影响图

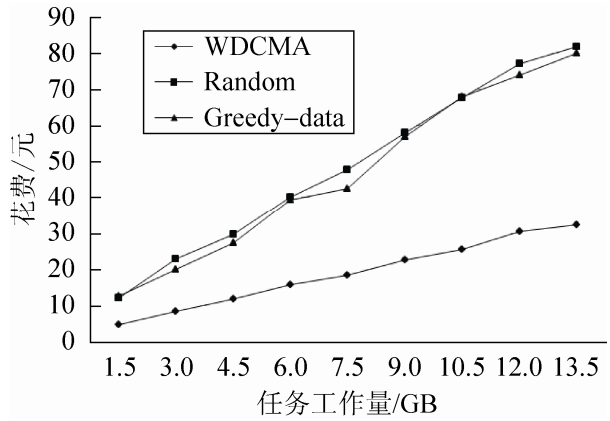


图 5 任务工作量对花费影响图

在之后的实验中设置数据中心的数量为 5 个，实验中工作流包含的每个子任务都需要处理 1.5GB 数据，子任务之间的依赖关系为随机生成。在图 6~7 中显示，随着工作流中子任务数量的增加，本算法的工作流执行时间和花费也明显低于对比算法。

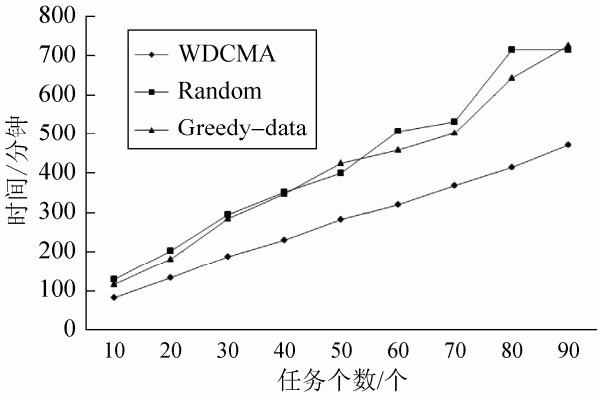


图 6 任务数量对运行时间影响图

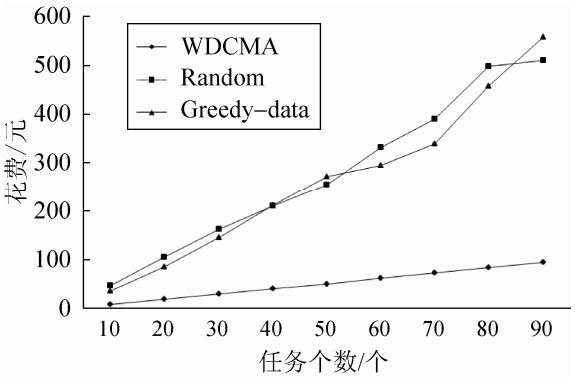


图 7 任务数量对花费影响图

为了进一步对算法的效果进行实验，本文建立了 3 个提供 Hadoop 计算服务的数据中心。集群 A，B，C 都由 4 台主机组成，每个集群都配置了 Hadoop 框架，其中 1 台主机作为 master，3 台主机作为 slave。主机使用了 ubuntu12 操作系统，硬件配置为 2GB 内存，2GHz 的双核处理器以及 250GB 的硬盘。每个 slave 节点配置了 2 个 map 槽位和 1 个 reduce 槽位。3 个集群各自接入交换机 a, b, c，交换机 a，b，c 和一台作为 controller 的主机接入交换机 d。工作流的请求会被提交到主机 controller，在 controller 中运行调度算法，将子任务分配到各

个集群。Controller 还要负责集群之间的数据迁移工作，在 Controller 中安装了一个数据迁移控制器，调用 HDFS 的接口来下载和上传文件。同时数据迁移控制器还负责管理不同集群的上传和下载带宽。

每个数据中心的参数配置如表 2 所示。

表 2 数据中心配置表

数据中心	计算能力/(MB/min)	价格/(分/min)	带宽/(MB/min)	网络价格/(分/min)
1	30	1	100	0.5
2	31	2	110	0.6
3	35	1.8	120	0.5

云计算中，跨数据中心的查询是一种常见的任务。SQL 语句“SELECT all FROM ((Table A INNER JOIN Table B ON A.x=B.x) INNER JOIN (Table C INNER JOIN Table D ON C.y=D.y) ON A.z=C.z) ORDER BY A.m”描述了一个查询任务。该任务可以分解为由多个 Hadoop 作业所组成的工作流。工作流可以分为 4 个子任务，任务 1 是表 A 和表 B 的连接操作，任务 1 的输出结果为表 E；任务 2 是表 C 和表 D 的连接操作，任务 2 的输出结果为表 F；任务 3 是表 E 和表 F 的连接操作，任务 3 的输出结果为表 G；任务 4 是对表 G 的一个排序操作，输出的结果为表 H。

在表 3 中描述了工作流运行过程中各个子任务用到的输入、输出数据的信息。表 A，B，C，D 是初始数据，在工作流开始之前就被分别保存在 3 个集群中。

表 3 数据描述表

表名	位置	数据大小/MB	描述
A	0	100	Initial data
B	1	110	Initial data
C	0	100	Initial data
D	2	120	Initial data
E	-	210	Table A join B
F	-	220	Table C join D
G	-	430	Table E join F
H	-	430	Sort result

表 4 显示了运行结果，本算法虽然并非运行时间最短，但是综合时间和花费的情况来看，效果优于对比算法。

表 4 运行结果表

算法	时间/min	花费	U
随机选择算法	6	65.3	391.8
贪婪数据迁移最小算法	5	199.4	997
多数据中心工作流调度算法	6	46.1	276.6

5 结论

本章提出了一个可以在分布式的数据中心进行工作流任务调度的算法。该算法使用有向无环图对工作流进行建模并化简，将数据中心之间的数据迁移映射为工作流中的一个子任务，以运行时间和花费作为调度的优化目标。本章使用 CloudSim 工具进行了仿真实验，使用 3 个 Hadoop 集群进行了实际运行实验。实验结果显示，算法的性能优于对比算法。

参考文献:

[1] 杨来, 史忠植, 梁帆, 等. 基于 Hadoop 云平台的并行数据挖掘方法 [J]. 系统仿真学报, 2013, 25(5): 936-944. (Yang Lai, Shi Zhongzhi, Liang Fan. Parallel Approach in Data Mining Based on Hadoop Cloud Platform [J]. Journal of System Simulation (S1004-731X), 2013, 25(5): 936-944.)

[2] 李伯虎, 柴旭东, 侯宝存, 等. 一种基于云计算理念的网络化建模与仿真平台-“云仿真平台” [J]. 系统仿真学报, 2009, 21(17): 5292-5299. (Li Bohu, Chai Xudong, Hou Baocun, et al. Networked Modeling & Simulation Platform Based on Concept of Cloud Computing-Cloud Simulation Platform [J]. Journal of System Simulation (S1004-731X), 2009, 21(17): 5292-5299.)

[3] 黄安祥, 冯晓文, 李劲松, 等. 基于云计算平台的航空兵训练仿真体系结构 [J]. 系统仿真学报, 2011, 23(增1): 106-109. (Huang Anxiang, Feng Xiaowen, Li Jinsong, et al. Aviation Simulation Architecture Based on Cloud Computing Platform [J]. Journal of System Simulation (S1004-731X), 2011, 23(S1): 106-109.)

[4] Olston C, Chiou G, Chitnis L, et al. Nova: Continuous Pig/Hadoop Workflows [C]// Proceedings of the SIGMOD'2011, Athens, Greece. USA: ACM, 2011: 1081-1090.

[5] Mao Y, Wu W, Zhang H, et al. GreenPipe: A Hadoop

Based Workflow System on Energy-Efficient Clouds [C]// Proceedings of the IPDPSW'2012, Shanghai, China. USA: IEEE, 2012: 2211-2219.

[6] Wang L, Tao J, Ranjan R, et al. G-Hadoop: MapReduce Across Distributed Data Centers for Data-Intensive Computing [J]. Future Generation Computer Systems (S0167-739X), 2013, 29(3): 739-750.

[7] Ahmed R, DeSmedt P, Kent W, et al. Pegasus: A System for Seamless Integration of Heterogeneous Information Sources [C]// Compcon Spring'91. Digest of Papers. USA: IEEE, 1991: 128-136.

[8] Du Z, Hu J, Chen Y. Optimized QoS-aware Replica Placement Heuristics and Applications in Astronomy Data Grid [J]. Journal of Systems and Software (S0164-1212), 2011, 84(7): 1224-1232.

[9] Fedak G, He H, Cappello F. BitDew: A Programmable Environment for Large-Scale Data Management and Distribution [J]. Journal of Network and Computer Applications (S1084-8045), 2009, 32(5): 961-975.

[10] Gu Y, Grossman R. Toward Efficient and Simplified Distributed Data Intensive Computing [J]. IEEE Transactions on Parallel and Distributed Systems (S1045-9219). 2011, 22(6): 974-984.

[11] Jayalath C, Stephen J, Eugster P. From The Cloud to The Atmosphere: Running MapReduce Across Data Centers [J]. IEEE Transactions on Computers (S0018-9340), 2014, 63(1): 74-87.

[12] Dean J, Ghemawat S, MapReduce: Simplified Data Processing on Large Clusters [J]. Communications of The ACM (S0001-0782). 2008, 51(1): 107-113.

[13] Sakellariou R, Zhao H, Tsiakkouri E, et al. Scheduling Workflows with Budget Constraints [C]// Integrated Research in GRID Computing. USA: Springer, 2007: 189-202.

[14] Fard M, Prodan R, Fahringer T. Multi-objective List Scheduling of Workflow Applications in Distributed Computing Infrastructures [J]. Journal of Parallel and Distributed Computing (S0743-7315), 2014, 73(3): 2152-2165.

[15] Adabi S, Movaghar A, Rahmani M. Bi-level Fuzzy Based Advanced Reservation of Cloud Workflow Applications on Distributed Grid Resources [J]. The Journal of Supercomputing (S0920-8542), 2014, 63(1): 175-218.

[16] Benoit A, Hakem M, Robert Y. Multi-criteria Scheduling of Precedence Task Graphs on Heterogeneous Platforms [J]. The Computer Journal (S0010-4620), 2010, 53(6): 772-785.

[17] Blythe J, Jain S, Deelman E, et al. Task Scheduling Strategies for Workflow-Based Applications in Grids [C]// Proceedings of the IEEE International Symposium on Cluster Computing and the Grid, 2005. USA: IEEE, 2005.

[18] Yu J, Buyya R. QoS-based Scheduling of Workflow Applications on Service Grids [C]// Proceedings of the 1st IEEE International Conference on e-Science and Grid Computing, 2005. USA: IEEE, 2005.

[19] 王凌, 郑大钟. 多目标优化的一类模拟退火算法[J]. 计算机工程与应用, 2002, 38(8): 4-5. (Wang Ling, Zheng Dazhong. A Class of Simulated Annealing Approach for Multi-objective Optimization [J]. Computer Engineering and Applications (S1002-8331), 2002, 38(8): 4-5.)

[20]谢涛, 陈火旺. 多目标优化与决策问题的演化算法 [J]. 中国工程科学, 2002, 4(2): 59-69

[21] Calheiros R N, Ranjan R, Beloglazov A, et al. CloudSim: A Toolkit for Modeling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms [J]. Software: Practice and Experience (S1097-024X), 2011, 41(1): 23-50.

[22] 刘少伟, 孔令梅, 任开军, 等. 云环境下优化科学工作流执行性能的两阶段数据放置与任务调度策略 [J]. 计算机学报, 2011, 34(11): 2121-2130. (Liu Shaowei, Kong Lingmei, Ren Kaijun, et al. A Two-step Data Placement and Task Scheduling Strategy for Optimizing Scientific Workflow Performance on Cloud Computing Platform [J]. Chinese Journal of Computer (S0254-4164), 2011, 34(11): 2121-2130.)