云计算环境下面向数据密集型应用的数据布局策略与方法

郑 湃 崔立真 王海洋 徐 猛

(山东大学计算机科学与技术学院 济南 250101)

摘 要 云计算环境下面向流程的数据密集型应用已被广泛应用于多个领域.面对多数据中心的云计算环境,这类应用在数据布局方面遇到了新的挑战,主要表现在如何减少跨数据中心的数据传输、如何保持数据间的依赖性以及如何在提高效率的同时兼顾全局的负载均衡等.针对这些挑战,文中提出一种三阶段数据布局策略,分别针对跨数据中心数据传输、数据依赖关系和全局负载均衡三个目标对数据布局方案进行求解和优化.实验显示,文中提出的数据布局策略具有良好的综合性能,特别是在降低流程执行过程中由跨数据中心数据传输所导致的时间开销方面,效果尤为明显.

关键词 云计算;流程;数据密集;数据布局;数据依赖

中图法分类号 TP393 **DOI 号**: 10.3724/SP.J.1016.2010.01472

A Data Placement Strategy for Data-Intensive Applications in Cloud

ZHENG Pai CUI Li-Zhen WANG Hai-Yang XU Meng

(*School of Computer Science and Technology, Shandong University, Jinan* 250101)

Abstract With the development of information technology, data-intensive applications in cloud have been used in more and more fields. Because of the decentralized data centers in cloud, these applications now are facing some new challenges in data placement which mainly include how to reduce the time cost of data movements between data centers, how to deal with the data dependencies, and how to keep a relative load balancing of data centers. This paper proposes a data placement strategy, the three stages of which address the three challenges above respectively. Simulation shows that the strategy can effectively reduce the time cost of data movements across data centers during the application's execution.

Keywords cloud computing; process; data-intensive; data placement; data dependency

1 引 言

随着信息技术的发展和普及,互联网逐渐成为一种计算平台.云计算是一种典型的网络计算模式,强调在虚拟计算环境下运行大规模应用的可伸缩性和可用性.基于云计算的大型网络应用呈现出分布、异构的特点和数据密集的趋势,如科学工作流系统,这类应用被称为数据密集型应用[1].目前数据密集型应用已被广泛应用于天文学[2]、高能物理学[3]以及生物信息学[4]等领域.这类应用的数据密集性主要体现在其处理的数据大小通常达 TB 甚至 PB 级,

收稿日期:2010-06-11.本课题得到国家自然科学基金(90818001)、山东省科技攻关计划(2008GG30001005,2009GG10001002)、高等学校博士学科点专项科研基金(200804221031)和山东大学自主创新基金(2009TS030)资助.**郑 湃**,男,1982 年生,硕士研究生,主要研究方向为云计算数据管理. E-mail:zhengpai_4u@163.com. **崔立真**(通信作者),男,1976 年生,博士,副教授,主要研究方向为软件与数据工程、工作流技术. E-mail: clz@sdu.edu.cn. **王海洋**,男,1965 年生,教授,博士生导师,主要研究领域为数据库应用. **徐 猛**,男,1978 年生,博士研究生,主要研究方向为软件与数据工程.

其中既有已存在的输入数据源，也有在对数据进行分析和处理的过程中产生的中间数据和最终结果数据，而通过使用流程管理技术，可以实现这类数据密集型应用的自动化执行. 数据密集型应用，尤其是那些面向流程的数据密集型应用，在利用云计算环境的过程中遇到了一些新的挑战，特别是在数据布局方面尤为突出. 在云计算环境下部署并执行数据密集型应用，需要多数据中心的协作，因此在多数据中心环境下如何为大量数据选择合适的存放位置变得至关重要，具体表现为：(1) 面对云环境下的多数据中心，这些应用在其执行过程中不可避免地需要进行跨数据中心的数据传输. 一方面数据规模巨大而数据中心间网络带宽有限，另一方面存在一些数据只能被存放于指定的数据中心而不能被移动，故数据中心间的数据传输成为一个挑战. (2) 这类应用的流程特性决定了其数据之间存在数据依赖关系. 在多数据中心环境下，合理的数据布局方案应力求保持这种数据间的依赖关系，这将利于降低流程执行过程中跨数据中心数据传输所导致的时间开销，进而提升执行效率. (3) 合理的数据布局方案，应在对应用执行效率进行优化的基础上，兼顾多数据中心间的全局负载均衡.

本文通过分析云计算环境下面向流程的数据密集型应用的特点，在全面考虑数据传输次数、数据集大小以及数据中心间网络带宽等因素的基础上，提出云计算环境下面向数据密集型应用的三阶段数据布局策略，该策略的 3 个阶段分别针对跨数据中心数据传输、数据间依赖关系和多数据中心间负载均衡三个目标对数据布局方案进行求解和优化. 该策略一方面解决了多数据中心环境下大规模数据的布局问题，降低了应用执行过程中跨数据中心数据传输所导致的时间开销；另一方面，对数据集间的数据依赖关系进行建模并依此对不同数据布局方案进行评价和调整，以使数据布局方案尽可能符合数据集间的依赖关系；此外，对不同的数据布局方案所对应的负载均衡状况进行量化分析并依此对数据布局方案进行筛选，从而在保证执行效率的基础上兼顾全局的负载均衡.

本文第 2 节介绍相关工作，并在此基础上进一步阐明本文与相关工作的差异与研究意义；第 3 节介绍云环境下数据密集型流程应用的数据布局问题，并对相关概念进行建模；第 4 节介绍本文所提的三阶段数据布局管理策略以及数据布局方案的求解与优化过程；第 5 节通过仿真实验将本文数据布局策略与其它同类策略进行对比，并对结果进行分析；最后一节，将对本文工作进行总结并对后续研究进行展望.

2　相关工作

本节首先简要介绍当前云计算环境下的数据管理系统；然后阐述几种数据密集型应用系统的数据管理策略；之后介绍目前关于数据依赖性的研究；最后，介绍当前针对云计算环境下数据密集型应用的数据布局问题的相关研究，并指出其局限性.

随着云计算日益受到重视，目前出现了一些云计算环境下的数据管理系统，例如 Google File System[5] 和 Hadoop①，二者均对用户隐藏了用于存储应用数据的基础设施. Google File System 主要针对 Web 搜索应用，而非云计算环境下的流程应用. Hadoop 则是一个更为通用的分布式文件系统，包括 Amazon 和 Facebook 在内的许多公司使用该系统. 当 Hadoop 文件系统接收到一个文件时，系统会自动将该文件分为若干块，并将每个块随机放置于某个集群中. 此外，Cumulus 项目[6] 提出了一个单数据中心环境的云架构. 然而，上述云数据管理系统并未针对云环境下数据密集型流程应用的数据布局问题进行相关研究.

常见的数据密集型流程应用系统都有其各自的数据管理策略. 在流程的构建阶段，这类策略主要针对数据的建模，例如 Kepler[3] 使用了一种面向角色的数据建模方法，用于网格环境下的大规模数据建模；Taverna[4] 和 ASKALON[7] 则分别采用了各自的流程定义语言来表示其数据流. 在流程的执行阶段，大部分系统采用某种数据网格来对数据进行管理，例如 Kepler 使用了 SRB[8] 系统，Pegasus[2] 和 Triana[9] 采用了 RLS 系统[10]. 数据网格的主要作用是为分布式环境下的数据密集型应用提供基础设施与服务，以实现对分布存储资源中海量数据集的访问、移动和修改[11]. 然而，无论在流程的构建阶段还是执行阶段，这些系统的数据管理策略都没有关注数据的存放布局，既没有考虑数据间的依赖性，也无法减少跨数据中心的数据移动.

当前已有一些研究注意到数据密集型流程应用中的数据依赖性问题. Filecules 项目[12] 基于依赖关系对文件进行分组，实验数据显示了其分组策略的可用性和有效性. BitDew[13] 将数据的依赖性定义为数据的一个属性，该属性由用户进行预定义. 然而，

① Hadoop：http：//hadoop.apache.org

云计算环境中的所有数据均位于各数据中心，因此由用户来定义数据间的依赖关系，对云计算环境下的数据密集型应用而言是不现实的.此外，这些研究虽然关注数据间的依赖性，但并未基于此来解决数据布局问题，也没有考虑跨数据中心数据传输的代价问题.

针对云计算环境下面向流程的数据密集型应用的数据布局管理问题，文献[14]提出了一种基于聚类矩阵的数据布局策略，用于多数据中心环境下数据布局方案的求解和优化.该方法首先将流程应用的所有输入数据构建为依赖矩阵，再利用BEA[15]算法对依赖矩阵进行聚类变换从而得到聚类矩阵，然后基于聚类矩阵对所有数据集组成的集合进行划分，并分别为所得每个划分分配存放位置.实验证明该策略能够减少云计算环境下流程应用执行过程中的跨数据中心数据传输的总次数.然而，数据传输总次数的减少并不一定意味着执行效率的提高，因为每次所传输数据量的大小并不一定相同，且遍布Internet的各数据中心两两之间的网络带宽也不一定相同，即执行过程中跨数据中心数据传输总次数的减少，并不能等同于数据传输所导致时间开销的减少.

综上所述，虽然针对云数据管理和流程密集型应用数据管理的研究较多，但关于云计算环境下数据密集型流程应用的数据布局问题的研究较少，当前少数针对该问题的研究也存在一定的局限性，即只关注跨数据中心数据传输的次数，而忽视了所传输数据量大小、各数据中心间的网络带宽差异等因素，因此难以对跨数据中心数据传输所导致的时间开销进行有针对性的优化并进而提高流程应用的执行效率.

3 问题描述与建模

本节将对云计算环境下数据密集型流程应用的数据布局问题的相关概念进行建模，具体包括云计算环境、面向流程的数据密集型应用、单次数据传输时间开销和全局数据传输时间开销.

3.1 云计算环境

云计算环境由多个分布的数据中心组成.由于本文研究重点为云计算环境下的数据布局问题，故本文仅关注其存储资源和网络带宽.

定义 1. 将云计算环境表示为多个分布式数据中心组成的集合 $DC=\bigcup_{i=1,2\cdots,|DC|}\{dc_i\}$.其中 dc_i 表示编号为 i 的数据中心，cs_i 表示数据中心 dc_i 的可用存储空间.DC 中各数据中心间网络带宽矩阵表示为

$$\boldsymbol{B}=\begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1|DC|} \\ b_{21} & b_{22} & \cdots & b_{2|DC|} \\ \vdots & \vdots & \cdots & \vdots \\ b_{|DC|1} & b_{|DC|2} & \cdots & b_{|DC||DC|} \end{bmatrix}$$

对 $\forall i,j=1,2,\cdots,|DC|$ 且 $i\neq j$，B 中元素 b_{ij} 表示数据中心 dc_i 和 dc_j 之间的网络带宽值.本文假设各数据中心间网络带宽值可知并忽略其实时波动.

3.2 面向流程的数据密集型应用

定义 2. 将面向流程的数据密集型应用定义为三元组 $P=\langle T,C,DS\rangle$.其中 T 是 P 中所有任务的集合；C 是各任务间的控制流的集合；DS 是 P 中所有数据集的集合.

对流程应用中的单个任务，本文只关注其输入数据集与输出数据集，定义如下.

定义 3. 将面向流程的数据密集型应用 $P=\langle T,C,DS\rangle$ 中的单个任务定义为二元组 $t_i=\langle IDS_i, ODS_i\rangle$，$i=1,2,\cdots,|T|$.其中 IDS_i 是任务 t_i 的输入数据集组成的集合；ODS_i 是任务 t_i 的输出数据集组成的集合.

有些数据集只允许被存放在指定的数据中心，本文将其称为固定位置数据集，并将 DS 中所有固定位置数据集组成的集合记为 DS_{fix}，对 $\forall d_i\in DS_{fix}$，记 d_i 的指定存放位置的数据中心编号为 $fix(d_i)$；而允许被存放于任意数据中心的数据集则被称为可变位置数据集，记 DS 中所有可变位置数据集组成的集合为 DS_{flex}.

按数据集的产生时间，又可将数据集分为初始数据集和生成数据集.作为流程应用的初始输入数据而在流程执行前已存在的数据集，本文将其称为初始数据集，并将 DS 中所有初始数据集组成的集合记为 DS_{ini}；而在执行过程中产生的数据集则被称为生成数据集，DS 中所有生成数据集组成的集合被记为 DS_{gen}.

对 DS 中的单个数据集，本文只关注其大小、固定存放位置和来源，定义如下.

定义 4. 将面向流程的数据密集型应用 $P=\langle T,C,DS\rangle$ 中的单个数据集定义为三元组 $d_i=\langle ds_i,lc_i,gt_i\rangle$，$i=1,2,\cdots,|DS|$.其中 ds_i 是数据集 d_i 的大小；lc_i 和 gt_i 的取值如下：

$$lc_i=\begin{cases}0, & d_i\in DS_{flex}\\ fix(d_i), & d_i\in DS_{fix}\end{cases},\quad gt_i=\begin{cases}0, & d_i\in DS_{ini}\\ 1, & d_i\in DS_{gen}\end{cases}.$$

结合图1中的两个简单示例说明本文对面向流程的数据密集型应用的建模.在图1(a)中：$T=$

$\{t_1,t_2,t_3,t_4\}$，以其中 t_3 为例，$t_3=\langle\{d_3,d_4\},\{d_5\}\rangle$；$DS=\{d_1,d_2,d_3,d_4,d_5,d_6,d_7\}$，以其中 d_6 为例，不妨设 d_6 是固定位置数据集必须被放置于 2 号数据中心且 d_6 大小为 2000GB，则 $d_6=\langle 2000,2,0\rangle$. 在图 1(b)中：$T=\{t_1,t_2,t_3,t_4,t_5,t_6\}$，以其中任务 t_3 为例，$t_3=\langle\{d_3,d_7\},\{d_6\}\rangle$；$DS=\{d_1,d_2,d_3,d_4,d_5,d_6,d_7,d_8,d_9,d_{10}\}$，以其中数据集 d_8 为例，不妨设 d_8 是可变位置数据集且大小为 500GB，则 $d_8=\langle 500,0,1\rangle$.

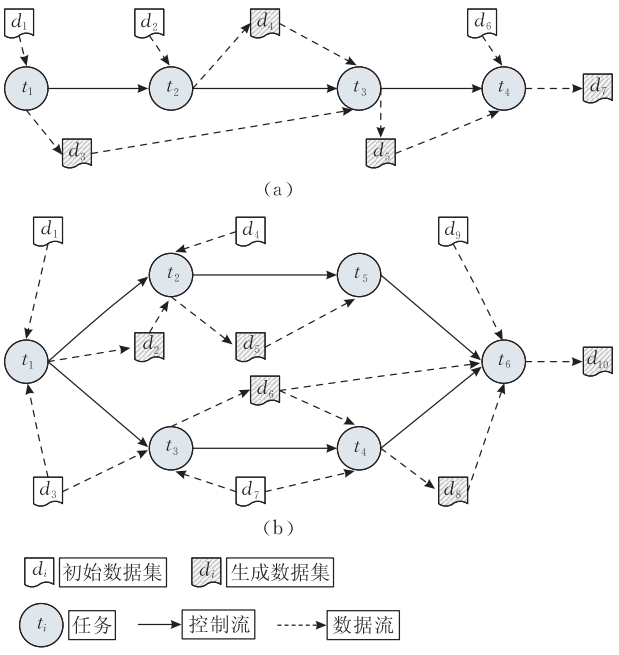

图 1　简单示例

3.3　单次数据传输时间开销

云计算环境下，流程中每个任务要开始执行须至少满足两个条件：(1) 该任务被调度至某个数据中心；(2) 该任务的所有输入数据集均位于本地. 因此若一个任务的多个输入数据集位于不同的数据中心，则须在任务开始执行前进行跨数据中心的数据传输. 关于任务调度策略已有许多相关研究且并非本文重点，故本文基于将任务调度至数据中心比进行跨数据中心数据传输所造成时间开销更小的假设，将任务调度策略简化为：将任务调度至那个使执行该任务所需跨数据中心数据传输所导致时间开销最小的数据中心.

考虑单个数据集单次跨数据中心传输的情况. 设源数据中心、目标数据中心和目标数据集分别为 dc_i、dc_j 和 d_k，则该次数据传输的时间开销可表示为

$$TimeCost_{\text{single}}(d_k,dc_i,dc_j)=\frac{ds_k}{b_{ij}}+C_{ij} \tag{1}$$

式(1)中 d_k，dc_i，dc_j 3 个参数依次表示目标数据集、源数据中心和目标数据中心. ds_k 表示数据集 d_k 的大小；b_{ij} 表示数据中心 dc_i 和 dc_j 间的网络带宽. 此外，在跨数据中心的数据传输过程中存在着请求、响应、建立连接、断开连接等步骤，也会造成一部分时间开销，将其记为 C_{ij}. 考虑到云计算环境下面向流程的数据密集型应用的数据规模巨大，而 C_{ij} 相对较小，故本文忽略 C_{ij} 所代表的那部分与被传输数据集大小无关的时间开销. 因此，使用下面式(2)可近似计算单个数据集单次跨数据中心传输所造成的时间开销

$$TimeCost_{\text{single}}(d_k,dc_i,dc_j)\approx\frac{ds_k}{b_{ij}} \tag{2}$$

3.4　全局数据传输时间开销

定义 5. 在数据中心集合 $DC=\bigcup\limits_{i=1,2,\cdots,|DC|}\{dc_i\}$ 已知的云计算环境中，面向流程的数据密集型应用 $P=\langle T,C,DS\rangle$ 的一个数据布局方案，是从集合 DS 到集合 DC 的映射，使对 $\forall d_i\in DS$，存在唯一 $dc_j\in DC$ 与之对应. 记 $P=\langle T,C,DS\rangle$ 的一个数据布局方案为 s，则有 $s=\bigcup\limits_{i=1,2,\cdots,|DS|}\{d_i\rightarrow dc_j\}$，其中 $dc_j\in DC$. 对 $\forall d_i\in DS$，记 $s(d_i)$ 为在给定数据布局方案 s 中数据集 d_i 所对应存放位置的数据中心编号.

定义 6. 在数据中心集合 $DC=\bigcup\limits_{i=1,2,\cdots,|DC|}\{dc_i\}$ 已知的云计算环境中，面向流程的数据密集型应用 $P=\langle T,C,DS\rangle$ 的一个数据布局方案 s 的全局跨数据中心数据传输时间开销，是指在 DS 中所有数据集按 s 进行布局的情况下，P 的执行过程中由跨数据中心数据传输所导致的时间开销.

若已知数据中心集合 DC、流程应用 $P=\langle T,C,DS\rangle$ 和数据布局方案 s，则可通过模拟流程应用的执行过程来近似计算数据布局方案 s 的全局跨数据中心数据传输所导致的时间开销. 求解给定布局方案的全局跨数据中心数据传输时间开销的近似计算算法将在 4.1.3 节中给出.

4　数据布局策略与求解方法

如图 2 所示，本文提出的数据布局求解方法的过程可分为 3 个阶段：第 1 阶段，基于遗传算法求得一组跨数据中心数据传输代价较小的数据布局方案；第 2 阶段，计算第 1 阶段所得方案集合中各方案的数据依赖关系破坏度，并依此对方案集进行调整；第 3 阶段，对第 2 阶段所得方案集中的各方案的数据中心负载均衡状况进行定量分析，并依此确定最终的数据布局方案.

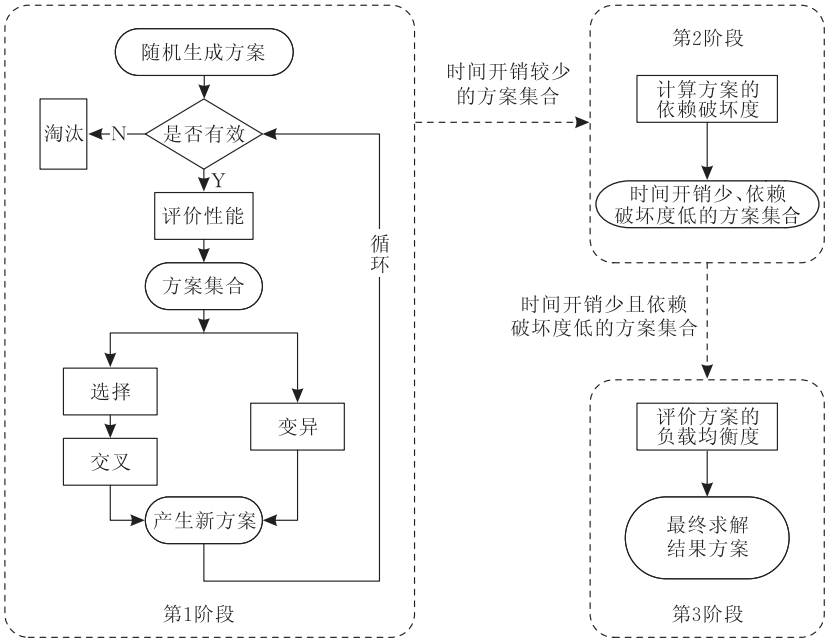

图 2 本文数据布局求解全过程流程图

4.1 数据布局求解第 1 阶段

在数据布局求解的第 1 阶段，基于遗传算法求解一组跨数据中心数据传输代价较小的数据布局方案.下面针对云计算环境下面向流程的数据密集型应用的数据布局问题，分别从编码规则、解的有效性保证和解的评价函数 3 个方面对数据布局方案求解过程的第 1 阶段进行介绍，并给出第 1 阶段求解算法.

4.1.1 编码规则

对云计算环境下的数据中心集合 DC，面向流程的数据密集型应用 $P=\langle T,C,DS\rangle$，数据布局方案求解第 1 阶段中解的编码规则如下：

对 $\forall dc_i\in DC$，用$\lceil\log_2^{|DC|}\rceil$位二进制数表示.具体的，对 $i=1,2,\cdots,|DC|-1$，dc_i 的编码为 i 的$\lceil\log_2^{|DC|}\rceil$位二进制数；而编号为$|DC|$的数据中心所对应的编码为$\lceil\log_2^{|DC|}\rceil$位 0.例如$|DC|=10$时，有如下关系：

数据中心	dc_1	dc_2	…	dc_9	dc_{10}
编码	0001	0010	…	1001	0000

对 DS 中的每个数据集 $d_i\in DS$，用一个长度为$\lceil\log_2^{|DC|}\rceil$的数据中心编码表示其存放位置.对每个数据布局方案，用$|DS|\times\lceil\log_2^{|DC|}\rceil$位二进制数表示：对 $i=1,2,\cdots,|DS|$，染色体左侧第 i 个长度为$\lceil\log_2^{|DC|}\rceil$的基因片段表示数据集 d_i 存放位置的数据中心编号.例如，当$|DC|=|DS|=10$时，“将数据集 d_i 存放于数据中心 dc_i”这一数据布局方案所对应的编码为

0001 0010 0011 0100 0101 0110 0111 1000 1001 0000.

然而并非每个长度为$|DS|\times\lceil\log_2^{|DC|}\rceil$的二进制数都一定能表示一个数据布局方案的有效解.4.1.2 节中将讨论解的有效性问题与解的有效性保证机制.

4.1.2 解的有效性保证

根据不同的失效原因可将无效解分为 3 类，下面分别说明，并给出对应的有效性保证机制.

第 1 类无效解.若解 s 的染色体中存在至少一个基因片段不能代表 DC 中的数据中心，则解 s 为第 1 类无效解.根据 4.1.1 节中的编码规则，当$|DC|$满足$\lceil\log_2^{|DC|}\rceil\neq\log_2^{|DC|}$时，则存在$|DC|$至$2^{\lceil\log_2^{|DC|}\rceil}-1$共$2^{\lceil\log_2^{|DC|}\rceil}-|DC|$个二进制数不能代表 DC 中的数据中心，记这些二进制数的集合为 $FS_{|DC|}$，记解 s 的染色体中所有$|DS|$个基因片段所对应的二进制数的集合为 PS_s，则有

$$PS_s\cap FS_{|DC|}\neq\varnothing,\text{则 } s \text{ 无效} \tag{3}$$

在初始布局方案的生成阶段、遗传过程中的交叉、变异阶段，用式(3)对每个新产生的解进行判断，若式(3)成立则淘汰该解.

第 2 类无效解.若解 s 所表示的数据布局方案使至少一个数据中心在流程开始执行前已处于超负荷状态，则解 s 为第 2 类无效解.为保证数据中心的正常运转，通常为数据中心设置一个经验参数 λ，若数据中心的使用率不小于 λ，则认为该数据中心处于超负荷状态.如 3.4 节所述，$s(d_i)$ 表示在解 s 所代表的数据布局方案中数据集 d_i 存放位置的数据

中心编号，则有

$$\exists \forall dc_i \in DC, \text{使} \frac{\sum_{s(d_j)=i} ds_j}{cs_i} \geq \lambda, \text{则 } s \text{ 无效} \quad (4)$$

在初始布局方案的生成阶段、遗传过程中的交叉、变异阶段，用式(4)对每个新产生的解进行判断，若式(4)成立则淘汰该解.

第 3 类无效解. 若在解 s 所表示的数据布局方案中，至少存在一个固定位置数据集的存放位置与其指定位置不同，则解 s 为第 3 类无效解.

$$\exists \forall d_i \in DS_{fix}, \text{使 } lc_i \neq 0 \text{ 且 } lc_i \neq s(d_i), \text{则 } s \text{ 无效} \quad (5)$$

本文对第 3 类无效解处理策略如下：设 $P=\langle T,C,DS\rangle$ 满足 $DS_{fix}\neq\varnothing$，则在生成初始解空间的过程中，将每个解的染色体中所有代表 $d_i\in DS_{fix}$ 存放位置的基因片段按 d_i 的下标升序依次与染色体左侧第一个代表 $d_i\in DS_{flex}$ 存放位置的基因片段交换位置，于是所有代表固定位置数据集位置的基因片段将按其下标升序依次分布于染色体左侧，而染色体的其余部分则表示所有可变位置数据集的存放位置. 通过对染色体进行上述基因片段的位置变换操作，可保证在交叉和变异阶段不产生第 3 类无效解.

记对 $P=\langle T,C,DS\rangle$ 的数据布局方案的解所进行的上述基因片段位置变换为 $Swap(P)$；记 $Swap(P)$ 的逆变换为 $Swap^{-1}(P)$.

4.1.3　数据布局方案的评价算法

本文所提的数据布局策略的根本目的在于降低云计算环境下面向流程的数据密集型应用在其执行过程中由跨数据中心数据传输所导致的时间开销，故本文以每个数据布局方案所对应的全局数据传输代价的近似值作为布局方案的评价函数，记数据布局方案的评价函数 $g(s)=TimeCost_{\text{total}}(DC,P,s)$.

对给定的数据中心集合 DC、流程应用 $P=\langle T,C,DS\rangle$ 和数据布局方案 s，可通过算法 1 求得 s 对应的数据传输时间开销的近似值，即 $g(s)$ 值.

算法 1.　数据布局方案的时间开销近似算法.

输入：$DC,B,P=\langle T,C,DS\rangle$，数据布局方案 s

输出：s 的时间开销近似值 $TimeCost_{\text{total}}(DC,P,s)$

主要步骤：

1. 初始化：令 $cost=0$，$TASK$ 为 P 中所有任务的集合；
2. 对 s 进行 4.1.2 节所述的 $Swap^{-1}(P)$ 变换；
3. 令 t 为 P 的第一个任务，并将其从 $TASK$ 中除去；
4. while ($TASK\neq\varnothing$)
5.　if (t 是分支任务)
6.　　for (每个分支)
7.　　记第 i 个分支对应的子流程为 P_i；
8.　　递归计算 $TimeCost_{\text{total}}(DC,P_i,s)$，结果记为 $cost_i$；
9.　$cost=cost+\max\{cost_i\}$；
10.　令 t' 为 t 对应的合并任务，将 t' 以及 t 与 t' 间所有任务从 $TASK$ 中除去，并令 t 为 t' 在 C 中的后继任务；
11.　else
12.　按 3.3 节所述任务调度策略为 t 选择执行位置，记该数据中心下标为 i；
13.　for(t 的 IDS 中每个满足 $s(d_j)\neq i$ 的数据集 d_j)
14.　$cost=cost+ds_j/b(s(d_j),i)$；
15.　令 t 为 t 在 C 中的后继任务；
16. Return ($cost$)；

算法 1 以“将任务调度至使执行该任务所需跨数据中心数据传输所导致时间开销最小的数据中心”为任务调度策略. 若使用其它任务调度策略，只需对算法 1 稍作修改.

算法 1 虽然只能近似计算给定数据布局方案的数据传输时间开销，但其结果能够反映不同数据布局方案所对应跨数据中心数据传输时间开销的差异与变化趋势，因此算法 1 结果的不精确性并不影响利用其对不同的数据布局方案进行评价和比较.

4.1.4　数据布局求解第 1 阶段算法

算法 2.　数据布局方案求解第 1 阶段算法.

输入：$DC,B,P=\langle T,C,DS\rangle$，数据布局方案 s

输出：时间开销较优的数据布局方案集合(CH)

主要步骤：

1. 初始化：定义变量 $popSize,genSize,maxGen$；
2. 令 $i=1$；
3. while ($i<=popSize$)
4.　生成随机布局方案 s；
5.　if (s 有效且不在 CH 中) //用式(3)～(5)进行判断
6.　　用算法 1 近似计算 s 的时间开销；
7.　　对 s 进行 4.1.2 节所述的 $Swap(P)$ 变换；
8.　　将 s 加入 CH，i++；
9. 令 $curGen=0$；
10. while ($curGen<maxGen$)
11.　for(CH 中的每个解 s)
12.　if ($random(0,1)<$变异率 mr)
13.　　在 s 中对应 DS_{flex} 的部分随机选择变异位置 p；
14.　　在位置 p 对 s 进行变异，记所得为 s'；
15.　　对解 s' 进行 $Swap^{-1}(P)$ 变换，记所得为 s''；
16.　　if (s'' 无效或 s' 在 CH 中)
　　//用式(3)～(4)进行判断
17.　　goto 13；
18.　　用 s' 替换 s，并用算法 1 重新计算 s 的时间开销；
19.　计算 CH 中每个解的被选中概率；

```
                                        //使用轮盘赌算法
20. 令 j=0;
21. while (true)
22.   依各解的被选中概率从 CH 中选两个不同解 s1,s2;
23.   if (random(0,1) <交叉率 cr)
24.     在 s 中对应 DS_flex 的部分随机选择交叉位置 q;
25.     在位置 q 对 s1 与 s2 进行交叉,记所得为 s1′,s2′;
26.     for (ns=s1′,s2′)
27.       对 ns 进行 Swap^-1(P)变换,记所得为 ns′;
28.       if (ns′有效且 ns 不在 CH 中)
                                        //用式(3)~(4)判断
29.         将 ns 加入 CH,并用算法 1 计算其时间开销;
30.         j++;
31.         if ( j==genSize) goto 32;
32. 将 CH 按各解对应时间开销的升序排序;
33. 从 CH 中除去后面 genSize 个解,curGen++;
34. 对 CH 中每个解进行 Swap^-1(P)变换;
35. Return(CH);
```

算法 2 返回的结果是一组跨数据中心数据传输时间开销较优的数据布局方案的集合,该集合中各方案按其所对应时间开销的升序排列.算法 2 所得即为本文数据布局方案求解方法第 1 阶段的结果,记为 $S_{(1)}$.

4.2 数据布局求解第 2 阶段

本文数据布局方案求解方法的第 2 阶段,将基于数据集之间的依赖关系对 $S_{(1)}$ 中的各方案进行评价和调整.下面给出数据集间的依赖度与数据布局方案的依赖关系破坏度的定义.

定义 7. 面向流程的数据密集型应用 $P=\langle T,C,DS\rangle$,对 $\forall d_i\in DS$,记 T_i 为 T 中所有以 d_i 作为输入数据集的任务的集合;对 $\forall d_i,d_j\in DS$,将 d_i 和 d_j 之间的数据依赖度记为 $denp(i,j)$,则对 $\forall i\neq j$ 有

$$denp(i,j)=\begin{cases}|T_i\cap T_j|\times\min\{ds_i,ds_j\}, & d_i,d_j\notin DS_{fix}\\ |T_i\cap T_j|\times ds_i, & d_i\notin DS_{fix}\text{ 且 }d_j\in DS_{fix}\\ |T_i\cap T_j|\times ds_j, & d_i\in DS_{fix}\text{ 且 }d_j\notin DS_{fix}\\ 0, & d_i,d_j\in DS_{fix}\end{cases}\tag{6}$$

定义 8. s 是面向流程的数据密集型应用 $P=\langle T,C,DS\rangle$ 的一个数据布局方案,s 的数据依赖破坏度记为 $denp\_damage(s)$,则有

$$denp\_damage(s)=\sum_{\forall d_i,d_j\in DS}denp(i,j)\times dist(i,j)\tag{7}$$

其中 $dist(i,j)=\begin{cases}0, & \text{若 } s(i)=s(j)\\ 1, & \text{若 } s(i)\neq s(j)\end{cases}$.

本文数据布局方案求解方法第 2 阶段的具体步骤如下:

(1) 记 $S_{(1)}$ 中各数据布局方案的全局数据传输时间开销最小值为 $timeCost_{\min}$,在 $S_{(1)}$ 中选择所有数据传输时间开销值小于 $timeCost_{\min}\times(1+\lambda_d)$ 的方案,记这些方案组成的集合为 $S^*_{(1)}$.这里的 λ_d 是一个百分比参数,可根据具体需求调整,本文实验中取 $\lambda_d=0.5\%$.

(2) 使用式(6)与式(7)计算 $S^*_{(1)}$ 中的每个方案的数据依赖破坏度值,并按其升序对 $S^*_{(1)}$ 中所有数据布局方案进行排序,所得结果即为本文数据布局方案求解方法第 2 阶段的结果,记为 $S_{(2)}$.

4.3 数据布局求解第 3 阶段

在本文数据布局方案求解方法的第 3 阶段,将分别计算 $S_{(2)}$ 中各方案的负载均衡性能,并依此对各布局方案进行评价和筛选.

(1) 记 $S_{(2)}$ 中各数据布局方案数据依赖破坏度最小值为 $damage_{\min}$,在 $S_{(2)}$ 中选择所有数据依赖破坏度值小于 $damage_{\min}\times(1+\lambda_l)$ 的方案,记这些方案所组成的集合为 $S^*_{(2)}$.这里的 λ_l 是一个百分比参数,可根据具体需求调整,本文实验中取 $\lambda_l=1\%$.

(2) 对 $S^*_{(2)}$ 中的每个方案,分别计算其所对应的各数据中心使用率的标准差,其中标准差最小的方案即为本文数据布局方案求解的最终结果.

5 仿真实验

5.1 实验设置与环境

本节将对本文提出的三阶段数据布局策略(下文简称本文策略)与文献[14]提出的聚类数据布局策略(下文简称聚类策略)进行对比实验.对每个随机生成的测试流程,分别使用本文策略与聚类策略求得各自的数据布局方案,然后基于两个方案对流程进行仿真执行,并记录该过程中的跨数据中心数据传输次数和时间开销.5.2 节图中所标注的数据,均为运行 1000 个参数相同的随机生成测试流程所得结果的平均值;图 3 与图 5 中所标注的时间开销单位为模拟时间单位.

实验环境为 Inter(R) Core(TM)2 Duo 2.93GHz,RAM 3GB,硬盘 320GB,100MB 网络带宽.

5.2 实验结果和分析

如图 3 所示,随着输入数据集个数的增加,两种数据布局策略所对应的跨数据中心数据传输所导致的时间开销都呈明显上升趋势;本文策略对应的时间开销明显低于聚类策略,且增速较缓.

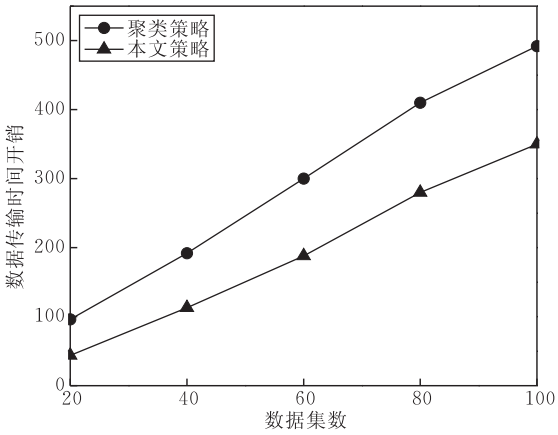

图 3　数据集数变化对时间开销的影响

如图 4 所示，随着输入数据集个数的增加，两种数据布局策略所对应的跨数据中心数据传输次数都呈明显上升趋势；与聚类策略相比，本文策略所对应的数据传输次数略多，但二者差别非常小.

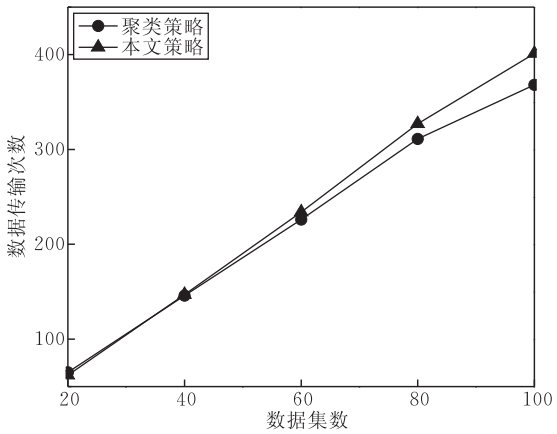

图 4　数据集数变化对移动次数的影响

如图 5 所示，随着数据中心个数的增加，两种数据布局策略所对应的跨数据中心数据传输所导致的时间开销都呈缓慢上升趋势；本文策略对应的时间开销明显低于聚类策略.

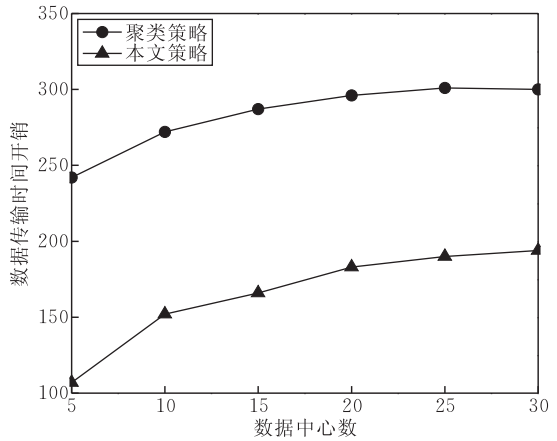

图 5　数据中心数变化对时间开销的影响

如图 6 所示，随着数据中心个数的增加，两种数据布局策略所对应的跨数据中心数据传输次数都呈上升趋势；与聚类策略相比，本文策略所对应的数据传输次数略多，但二者差别非常小.

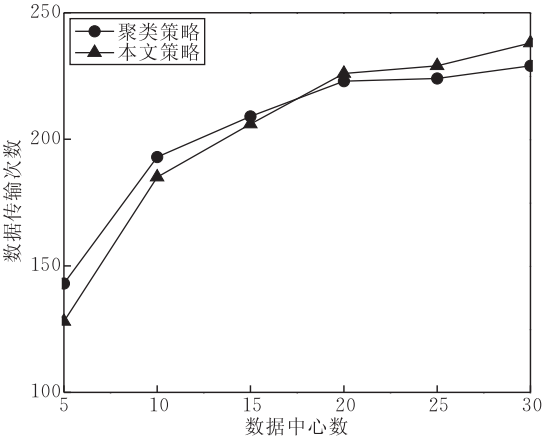

图 6　数据中心数变化对移动次数的影响

综上所述，对于流程应用执行过程中跨数据中心数据传输的次数，特别是在数据集数较多以及数据中心数较多的情况下，聚类策略所对应的传输次数较本文策略略少，但二者差距较小；而对于流程应用执行过程中跨数据中心数据传输所导致的时间开销，本文策略明显优于聚类策略，且随着数据集数量的增加，本文策略的优势有逐渐扩大的趋势.

6　结论与未来工作

本文首先指出，当前云计算环境下面向流程的数据密集型应用在数据布局管理方面所遇到的挑战，特别是跨数据中心数据传输所造成的不可忽视的时间开销；然后通过对该问题进行分析、建模，并在充分考虑数据移动次数、数据集大小以及网络带宽等因素的基础上提出了兼顾时间开销、数据依赖性和负载均衡三方面指标的数据布局策略与方法；之后，通过与其它数据布局策略的对比实验，说明本文的数据布局策略具有较好的性能，尤其是在降低跨数据中心数据传输导致的时间开销方面，本文策略的优势十分明显.

未来我们计划在以下方面进行进一步的研究：首先，计划对本文数据布局方案求解过程第 1 阶段中初始布局方案生成的环节进行优化，即基于一定的约束条件生成较优的初始方案，以提高求解效率. 此外，本文目前主要针对云环境下单个流程应用的数据布局问题进行研究，未来计划研究云计算环境下多个数据密集型流程应用的数据布局管理策略.

参 考 文 献

[1] Deelman E, Chervenak A. Data management challenges of data-intensive scientific workflows//Proceedings of the IEEE International Symposium on Cluster Computing and the Grid (CCGRID). Lyon, France, 2008: 687-692

[2] Deelman E, Blythe J, Gil Y, Kesselman C, Mehta G, Patil S, Su M H, Vahi K, Livny M. Pegasus: Mapping scientific workflows onto the grid//Proceedings of the European Across Grids Conference(AxGrids). Nicosia, Cyprus, 2004: 11-20

[3] Ludascher B, Altintas I, Berkley C, Higgins D, Jaeger E, Jones M, Lee E A. Scientific workflow management and the Kepler system. Concurrency and Computation: Practice and Experience, 2005, 18(10): 1039-1065

[4] Oinn T, Addis M, Ferris J, Marvin D, Senger M, Greenwood M, Carver T, Glover K, Pocock M R, Wipat A, Li P. Taverna: A tool for the composition and enactment of bioinformatics workflows. Bioinformatics, 2004, 20(17): 3045-3054

[5] Ghemawat S, Gobioff H, Leung S T. The google file system. ACM SIGOPS Operating Systems Review, 2003, 37(5): 29-43

[6] Wang L, Tao J, Kunze M, Castellanos A C, Kramer D, Karl W. Scientific cloud computing: Early definition and experience//Proceedings of the 10th IEEE International Conference on High Performance Computing and Communications(HPCC). Dalian, China, 2008: 825-830

[7] Wieczorek M, Prodan R, Fahringer T. Scheduling of scientific workflows in the ASKALON grid environment. SIGMOD Record, 2005, 34(3): 56-62

[8] Baru C, Moore R, Rajasekar A, Wan M. The SDSC storage resource broker//Proceedings of the IBMCentre for Advanced Studies Conference. Toronto, Canada, 1998: 1-12

[9] Churches D, Gombas G, Harrison A, Maassen J, Robinson C, Shields M, Taylor I, Wang I. Programming scientific and distributed workflow with Triana services. Concurrency and Computation: Practice and Experience, 2006, 18: 1021-1037

[10] Chervenak A, Deelman E, Foster I, Guy L, Hoschek W, Iamnitchi A, Kesselman C, Kunszt P, Ripeanu M, Schwartzkopf B, Stockinger H, Stockinger K, Tierney B. Giggle: A framework for constructing scalable replica location services//Proceedings of the ACM/IEEE Conference on Supercomputing. Baltimore, Maryland, USA, 2002: 1-17

[11] Venugopal S, Buyya R, Ramamohanarao K. A taxonomy of data grids for distributed data sharing, management, and processing. ACM Computing Survey, 2006, 38(1): 1-53

[12] Doraimani S, Iamnitchi A. File grouping for scientific data management: Lessons from experimenting with real traces//Proceedings of the 17th International Symposium on High Performance Distributed Computing. Boston, MA, USA, 2008: 153-164

[13] Fedak G, He H, Cappello F. BitDew: A programmable environment for large-scale data management and distribution//Proceedings of the 2008 ACM/IEEE Conference on Supercomputing. Austin, Texas, USA, 2008: 1-12

[14] Yuan D, Yang Y, Liu X, Chen J J. A data placement strategy in scientific cloud workflows. Future Generation Computer Systems, 2010, to appear

[15] McCormick W T, Sehweitzer P J, White T W. Problem decomposition and data reorganization by a clustering technique. Operations Research, 1972, 20(5): 993-1009

ZHENG Pai, born in 1982, M. S. candidate. His current research interests focus on data management for cloud computing.

CUI Li-Zhen, born in 1976, Ph. D., associate professor. His current research interests include software and data engineering, workflow management.

WANG Hai-Yang, born in 1965, professor, Ph. D. supervisor. His current research interests focus on database application.

XU Meng, born in 1978, Ph. D. candidate. His current research interests include software and data engineering.

Background

This work is supported by the National Nature Science Foundation of China under grant No. 90818001, Key Technology R&D Program of Shandong Province under Grant No. 2008GG30001005 and No. 2009GG10001002, Specialized Research Fund for the Doctoral Program of Higher Education under grant No. 200804221031, and Independent Innovation Foundation of Shandong University under grant No. 2009TS030.

Faced with the development of cloud computing, data placement which is a critical aspect of data management for data-intensive applications in cloud is gaining more and more attention. The primary challenge of data placement is the inevitable data movement between distributed data centers in cloud.

The main objective of this paper is to provide a data placement strategy to reduce the time cost of data movement between distributed data centers while taking data dependency and load balancing into consideration. Compared with an existing data placement strategy which focuses on the reduction of the data movement during the application's execution, our strategy can reduce the time cost of this data movement more efficiently.