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

刘少伟[1)] 孔令梅[2)] 任开军[1)] 宋君强[1)] 邓科峰[1)] 冷洪泽[1)]

[1)](国防科学技术大学计算机学院 长沙 410073)

[2)](中国人民解放军 78046 部队 成都 610011)

摘 要 云环境中跨数据中心科学工作流的高效执行通常面临数据交互量大的问题.文中给出基于相关度的两阶段高效数据放置策略和任务调度策略:即在工作流建立阶段根据数据依赖关系图把关系紧密型数据集尽可能放置到同一数据中心;而后任务调度策略在运行阶段将任务调度到数据依赖最大的数据中心执行,并将新产生数据集放置到相关度最高的数据中心.实验表明,该策略能有效减少跨数据中心科学工作流执行时的数据传输量,从而能有效提升科学工作流的执行效率,并能减少资源的租赁费用.

关键词 云计算;科学工作流;数据放置;数据相关;任务调度

中图法分类号 TP316 **DOI 号**: 10.3724/SP.J.1016.2011.02121

A Two-Step Data Placement and Task Scheduling Strategy for Optimizing Scientific Workflow Performance on Cloud Computing Platform

LIU Shao-Wei[1)] KONG Ling-Mei[2)] REN Kai-Jun[1)] SONG Jun-Qiang[1)] DENG Ke-Feng[1)] LENG Hong-Ze[1)]

[1)](*School of Computer*, *National University of Defense Technology*, *Changsha* 410073)

[2)](78046 *unit PLA*, *Chengdu* 610011)

Abstract Scientific workflows in collaborative cloud environments are becoming more and more popular. There is an urgent need to address the problem of large amount of data transfer across geo-distributed data centers during workflow execution. By utilizing data dependencies, we propose a two-stage data placement strategy and a task scheduling strategy for efficient workflow execution. With our strategy, the most related datasets can be placed into the same data center based on the data dependence between them at workflow build-time; then the tasks are scheduled to their most closely related data centers for execution and the newly-generated data sets are put into the data center that has the most dependency with them at workflow runtime. The experimental results show that the proposed strategy can significantly reduce the volume of data transfer among different data centers, and hence improve the performance of running scientific workflows and cut down the cost of doing science on the clouds as well.

Keywords cloud computing; scientific workflow; data placement; data dependence; task scheduling

---

收稿日期:2011-08-29;最终修改稿收到日期:2011-09-13.本课题得到国家自然科学基金项目"动态网络环境下服务快速合成与优化执行的算法研究"(60903042)、国家"八六三"高技术研究发展计划项目"基金地球系统模式一体化集成开发环境及示范应用研究"(863-2010AA012404)资助.**刘少伟**,男,1987 年生,硕士研究生,中国计算机学会(CCF)会员,主要研究方向为高性能计算、云计算、科学工作流.E-mail: liushaowei@nudt.edu.cn. **孔令梅**,女,1981 年生,硕士,工程师,主要研究方向为网络安全、科学工作流、云计算、粗糙集. **任开军**,男,1975 年生,博士,副研究员,主要研究方向为高性能计算、云计算、科学工作流、Web 服务合成. **宋君强**,男,1962 年生,研究员,博士生导师,主要研究领域为数值天气预报、大型并行应用软件、CPU/GPU 异构混合计算. **邓科峰**,男,1985 年生,博士研究生,主要研究方向为高性能计算、云计算、科学工作流. **冷洪泽**,男,1982 年生,博士研究生,主要研究方向为并行计算、数值天气预报.

1 引 言

在众多科学研究领域中,例如高能物理学、生物信息学、大气科学等,科学计算过程往往由成千上万个步骤构成,这往往需要对 TB 甚至 PB 量级的数据进行分析和处理.在过去,科学家通常使用简单的方法(例如 Perl 脚本语言)编排任务以及管理数据,但是这种方式不仅耗时而且容易出错.针对这一问题,科学工作流系统[1]开始受到关注并被用来进行自动化科学任务的编排、执行、监控以及追踪[2].随着问题求解规模的增大,当今大型科学工作流通常需要在复杂的分布式计算机系统上执行,例如超级计算机、分布式集群系统以及网格系统等.然而,构造这样的系统往往需要付出异常昂贵的代价,申请访问这些系统也需要复杂耗时的过程.云计算[3]技术提供共享基础架构的方法,它通过虚拟技术将分布在不同地理位置的计算资源和存储资源虚拟成一个资源池,用户需要使用时申请资源,使用完成后释放资源,从而使得资源可以重复利用.通过这种方式,云计算中心可以提供高性能的计算资源和海量的存储资源,而且成本低廉,使用简单.

随着云计算技术的深入发展和不断成熟,其高效、灵活、可定制的特点为解决科学工作流运行过程中遇到的难题提供了一种新思路.许多地理上分离的科学研究机构都有单独的私有云,每个私有云都可以提供一部分存储资源和计算资源,可以将它们看作是独立的数据中心,这些数据中心通过互联网技术形成更大的云计算平台.研究人员使用该平台时,需要将数据集上传到云计算平台.由于数据集规模可能非常庞大、数据中心之间存在带宽限制且部分数据集只能存放于指定的数据中心,研究人员不可能将所有数据集上传到同一数据中心或为每个数据中心上传所有数据集,而需要将不同数据集分别上传到不同数据中心,从而使得科学工作流的多个子任务可以并行执行.由于科学工作流任务间存在较强的数据依赖关系,其执行往往需要频繁对跨数据中心的数据集进行传输和访问,不合理的数据放置和任务调度策略容易导致数据中心间数据传输量和访问量过大,一方面增加用户使用云资源的费用外,另一方面也严重影响了科学工作流的执行效率,因此研究基于云环境的高效数据放置策略和任务调度策略对减少跨数据中心数据传输量、提升科学工作流执行性能、减少用户费用等方面具有重要意义.但是,现阶段已有方法还不能很好解决这一问题,我们在相关工作中给予了详细分析.

本文通过分析科学工作流数据集之间的依赖关系、数据集和数据中心之间的相关度以及任务和数据中心之间的相关度,提出了一种云平台下基于相关度的两阶段高效数据放置策略.该策略首先根据数据依赖关系图在工作流建立阶段将关系紧密的数据集放置到同一个数据中心,将关系松散的数据集放置在不同数据中心,这样保持了不同数据中心数据集之间的低耦合性和同一数据中心数据集之间的高内聚性;之后任务调度策略在运行阶段将任务调度到数据依赖最大的数据中心执行,并将新产生数据集放置到相关度最高的数据中心.实验表明,本文提出的策略不但极大地减少了数据中心间的数据移动量,提高了工作流任务的并行执行效率,同时也节省了用户的云资源使用费用.

本文的贡献主要表现在以下几个方面:在科学工作流建立阶段,提出了基于数据依赖的初始化数据布局方法,充分挖掘数据相关性,使得数据布局尽可能的符合使用规则,同一个任务所需数据集最大规模地聚集在一个数据中心;在科学工作流运行阶段,提出一种相应的任务调度策略,将任务调度到所需数据集规模最大的数据中心上,减少数据中心之间的数据传输量,加快科学工作流执行速度.针对任务执行过程中产生的中间数据集,本文利用数据集之间的关系,通过量化计算将它们放置到合适的数据中心,使得后续的任务调度与执行能够快速展开.

本文第 2 节介绍相关工作,说明本研究的意义以及与相关工作的差异;第 3 节介绍科学工作流在云环境下的执行流程,并给出相关的符号定义,然后分析数据放置策略需要考虑的各种因素;第 4 节详细阐述基于相关度的两阶段数据放置策略;第 5 节通过模拟实验将本策略与其它数据放置策略进行了对比,并对结果进行了详细分析;第 6 节对全文进行总结并对以后的研究方向进行展望.

2 相关工作

科学工作流的数据放置是一个非常重要且富有挑战性的问题,目前已有部分学者对此进行了研究与探索.现有科学工作流一般都有自身所属的数据管理系统,如 Pegasus 工作流系统使用的数据放置

策略[4-5]：它首先预先分配数据到执行任务的计算单元，这样可以加快任务的执行速度，降低任务等待时间；然后动态地删除那些不会被后续任务使用的数据，以减少存储开销.但这种策略只是保证了数据传输的可靠性和有效性，并没有考虑到云计算环境下因为数据交互引起的跨数据中心之间的传输开销.

为了减少数据传输开销，文献[6]采用副本机制，它使用改进后的贪婪算法和经过优化的遗传算法计算副本的最佳放置策略，并利用基于 Web 服务的数据网络系统 ADPPS(Astronomy Data Processing Pipeline System)产生工作流来进行实验验证.但数据集副本机制增加了存储开销，该策略主要针对网络环境下多节点之间的数据传输而非针对云计算平台上多数据中心之间的数据传输，并没有考虑到数据之间存在相关性和依赖关系.

另外一些研究对数据依赖进行了分析，如 BitDew[7] 由用户定义数据间的依赖关系，但并没有利用数据间的依赖关系减少传输开销.Gu 等人[8]设计和实现一种分布式文件系统 Sector/Sphere，系统中数据集是规模庞大的若干未分块(non-block)的文件集合；Sphere 通过设置目录和文件树将文件按照数据局部性原则聚合起来，同时使用高速传输协议 UDT[9] 和文件副本减少传输延迟，实验结果表明，该系统比 Hadoop① 处理数据要快 2～4 倍，但 Sphere 只是根据任务来聚集数据，并没有对数据之间的关系进行仔细分析利用.Nephele 项目[10]是现有的第一个数据处理框架，注重发掘在任务的调度、执行过程中 IaaS 云环境下资源的动态分配，有效地减少了资源使用开销，却没有减少数据传输开销.值得注意的是，以上的几种数据管理策略均不适用于云计算环境下数据密集型的工作流.

针对云计算环境下数据密集型科学工作流问题，Yuan 等人在文献[11]中提出了一种基于聚类矩阵的数据放置策略，用于多数据中心之间数据集的放置.该方法的数据放置策略分为两步，在科学工作流建立阶段，先利用所有数据集之间的关系构建一个相关度矩阵，然后通过 BEA 算法对相关度矩阵进行变换得到聚类矩阵，然后通过该矩阵将所有数据集划分为 K 个集合，每一个集合内部的数据集都是高内聚的，集合之间的数据集则是低耦合的；在科学工作流执行阶段，在考虑存储条件满足的情况下，新产生的数据集被放置在与它相关度最大的数据中心上.实验表明，该方法可以有效减少跨数据中心之间的数据移动次数.但这种方法并未考虑移动的数据大小，如果移动次数较少，但所移动的数据太大，传输开销不一定降低，导致科学工作流的执行效率反而下降.

3　科学工作流形式化描述和问题分析

3.1　相关模型和符号定义

我们首先形式化定义了科学工作流的运行环境和参数模型.

定义 1.　数据中心设为 $DC=\bigcup_{i=1,2,\cdots}\{dc_i\}$，其中，$dc_i=\langle cap_i, cs_i, \lambda_{ini}\rangle$ 表示编号为 i 的数据中心，cap_i 表示 dc_i 的计算能力，并用执行同一任务所需的时间的倒数来量化表示，并假设该值保持不变；cs_i 表示 dc_i 的存储空间大小；λ_{ini} 表示在科学工作流建立阶段，数据中心可以使用的存储空间的比例[11].因为科学工作流执行过程中产生的中间数据有可能规模庞大，因此在原始数据分配阶段要留有一定的空间来存储中间数据，所以 $0<\lambda_{ini}<1$.λ_{ini} 是一个经验值，它的大小取决于科学工作流的性质.

定义 2.　原始数据集设为 $DS_{ini}=\{d_1, d_2, d_3\cdots\}$ 表示在科学工作流建立时所存在的数据集，即所有原始输入.中间数据集设为 $DS_{gen}=\{d_1, d_2, d_3\cdots\}$，表示在科学工作流执行过程中所产生的数据集.

固定数据集和非固定数据集分别设为 FD 和 NFD.FD 表示必须放置在固定数据中心的数据集，这是因为某些数据需要特定数据中心的特定设备才能处理，或者某些数据具有私有性和产权性.NFD 表示没有固定数据中心的数据集，这是相对 FD 而言的.

定义 3.　$T=\{t_1, t_2, t_3\cdots\}$ 表示在科学工作流上运行的任务集，每一个任务执行都需要若干数据集作为输入.

定义 4.　$d_i=\langle T_i, s_i, dc_i, fix\_flag, depLink\rangle$ 表示科学工作流中编号为 i 的数据集.其中 $T_i=\{t_1, t_2, t_3\cdots\}$ 表示使用 d_i 的任务集合；s_i 表示数据集的大小；dc_i 表示 d_i 所对应的数据中心；fix_flag 为 true 表示 d_i 是固定数据，反之则为非固定数据；$depLink$ 是一个链表，将在定义 5 中详细解释.

3.2　实例分析与问题说明

科学工作流在云平台中的运行过程可以分为以

① Hadoop, http://hadoop.apache.org/(accessed 2011.07.04)

下几步：

(1)地理上分布在不同地区的若干数据中心构成了科学工作流运行的云计算环境；

(2)使用者运行科学工作流时，需要向云平台申请资源，并对云平台进行定制；

(3)使用者获取了所需要的运行平台后，上传输入数据并设置科学工作流的运行方式；

(4)在科学工作流的建立阶段，按照特定的数据放置策略将使用者的输入数据集放置在合适的数据中心，然后根据一定的任务调度策略将科学工作流中的子任务调度到合适的数据中心执行；

(5)在科学工作流的执行阶段，大量的任务可能产生大量的中间数据集，这些中间数据集也需要放置到合适的数据中心，直到科学工作流运行完毕.

图 1(a)给出了一个科学工作流的例子，该科学工作流包含 5 个子任务$\{t_1,t_2,t_3,t_4,t_5\}$，5 个输入数据集$\{d_1,d_2,d_3,d_4_f,d_5\}$和一个中间数据集$\{d_6\}$，其中$\{d_4_f\}$是$dc_2$上的固定数据集，不能移动.

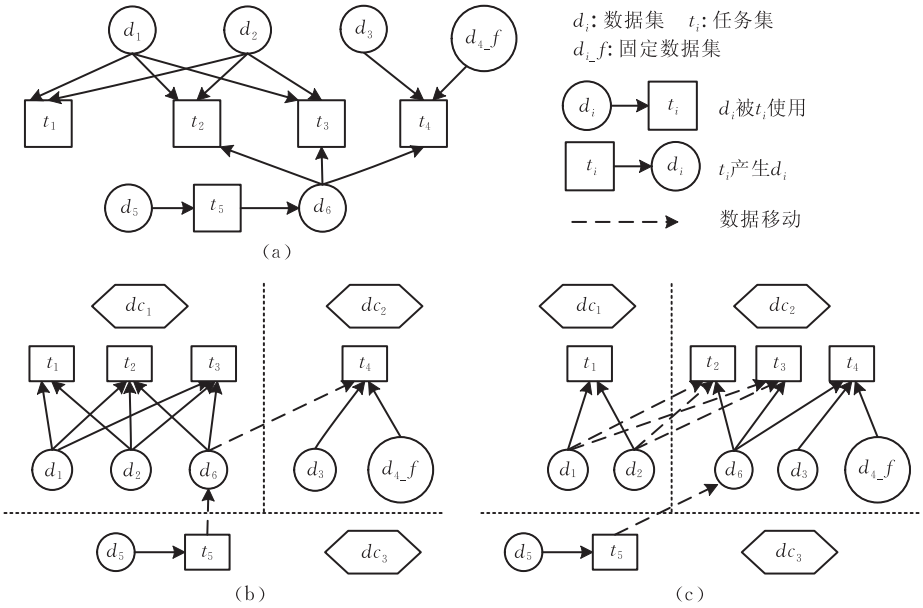

图 1 一个简单的科学工作流

以图 1 为例，科学工作流的数据放置策略需要考虑以下几点：

(1)数据相关对科学工作流的影响. 科学工作流运行中数据集和任务之间并不是一对多或者多对一的关系，而是多对多的关系，即一个数据集可能会被多个任务同时使用，一个任务也可能调用多个数据集. 从图 1(a)中可以看出，$\{d_1,d_2\}$同时被任务$\{t_1,t_2,t_3\}$使用，$\{d_3,d_4_f,d_6\}$同时被任务$\{t_4\}$使用，$\{d_5\}$被任务$\{t_5\}$使用并且任务执行完毕后会产生一个中间数据$\{d_6\}$. 由于移动数据比调度任务代价更大，即相比之下调度任务的开销几乎可以忽略，所以需要将关系紧密的数据集尽量放置到同一个数据中心.

(2)数据集大小对科学工作流的影响. 图 1(b)是按照文献[11]中的数据放置策略对产生的中间数据集$\{d_6\}$进行放置. 因为$\{d_6\}$和$\{d_1,d_2\}$同时被两个任务$\{t_2,t_3\}$使用，$\{d_6\}$和$\{d_3,d_4_f\}$同时被任务$\{t_4\}$使用，所以$\{d_6\}$产生后放置在dc_1上，科学工作流总的数据移动次数是两次. 图 1(c)使用的数据放置策略则将$\{d_6\}$放在dc_2上，科学工作流总的数据移动次数是 5 次. 但如果考虑数据大小，比如$\{d_1=1\text{MB},\ d_2=1\text{MB},\ d_6=30\text{MB}\}$，那么图 1(b)中数据移动次数虽然是两次，但数据移动量是 60 MB，而图 1(c)的数据移动次数是 5 次，但数据移动量是 34 MB，总的数据移动量大幅减少. 所以数据集大小对于科学工作流的数据放置策略和任务调度有着很大影响.

(3)固定数据集对科学工作流的影响. 因为固定数据集只能放置在特定的数据中心，无法向外传输，一旦任务使用到固定数据集，该任务一定会被调度到这个数据中心上执行. 如图 1(b)、(c)所示，t_4使用了固定数据集d_4_f，所以只能在dc_2上执行，这不仅影响了原始数据集d_3的放置，还影响了中间数据集d_6的放置. 所以固定数据集通过影响任务调度间接地影响了其它数据集的放置.

(4)数据中心的计算能力、存储能力对科学工

作流的影响.由于各个数据中心隶属于不同的组织机构,其计算能力、存储能力可能差异较大.合理的数据放置策略也要也要将这两个因素考虑到,即在存储空间足够的前提下,向计算能力强的数据中心放置尽量多的数据集,以加快科学工作流的执行速度.

由于数据移动开销对科学工作流性能影响较大,因此合理的数据放置策略应该努力减少数据移动量,本文针对这种情况提出了一种基于相关度的数据放置策略,该策略综合考虑了数据相关度(即数据之间被相同任务使用的多少)、数据大小、固定数据集、数据中心的计算能力和存储能力,有效提升了科学工作流的执行效率.

4　基于相关度的两阶段数据放置与任务调度策略

基于相关度的数据放置策略包括建立阶段的数据放置策略和运行阶段的数据放置策略两部分.

4.1　建立阶段数据放置策略

在科学工作流建立阶段,该策略对所有的原始输入数据集在逻辑上进行预分配,这样做可以优化数据分配方案,防止前期出现不合理分配的情况.预分配主要从局部性考虑,使得子任务在调度后,所需使用的数据集都尽量在本地数据中心上存储.预分配首先把所有的数据集分为两类:固定数据集(有固定数据中心的数据集)和非固定数据集(无固定数据中心的数据集),然后对非固定数据集中的数据属性进行研究,考虑数据集大小的前提下分析它们之间的数据相关度,然后根据数据中心的计算能力和存储能力对数据集进行预分配.所有原始数据集预分配完成以后,根据结果把数据集物理上分配到相应的数据中心.在这种方式下,数据相关度大的数据集原则上会分到同一个数据中心,拥有较强计算性能的数据中心原则上能分到更多的数据集.

假定科学工作流总共使用 m 个数据中心.

定义 5.　数据集相关度[12]设为

$$dep_{ij}=\begin{cases}count(T_i\cap T_j)\times\min\{s_i,s_j\}, & d_i,d_j\in NFD\\ count(T_i\cap T_j)\times s_i, & d_i\in NFD,d_j\in FD\\ count(T_i\cap T_j)\times s_i, & d_i\in FD,d_j\in NFD\\ 0, & d_i,d_j\in FD\end{cases}$$

表示数据 d_i 和数据 d_j 的相关度大小,其中 $count(T_i\cap T_j)$ 表示共同使用 d_i 和 d_j 的任务数量.dep_{ij} 与数据集大小有关.由定义 4 知,数据集 d_i 有一个属性 $depLink$,此处定义为 $depLink(i)=\{\langle d_i,dep_{ij}\rangle\mid j\neq i\}$,根据 dep_{ij} 大小形成一个降序链表.

定义 6.　预分配数据中心设为 DC_k,$k=1,2,\cdots,m$,假定 DC_k 的存储空间值为相应的 dc_k 存储空间大小.在科学工作流建立阶段,需要先将原始数据集逻辑上分配到 DC_k,$k=1,2,\cdots,m$,然后按照 $DC_k\rightarrow dc_k$ 的映射原则,将逻辑上的分配方案在物理的数据中心上实现.

定义 7.　待分配数据集集合设为 DC_{wait}.在科学工作流建立阶段,该集合中存放的是与其它任何数据集相关度均为 0 的数据集;在科学工作流运行阶段,该集合中存放在后续过程中需要分配的数据集.

如图 2 所示,在科学工作流建立阶段需要对所有的原始数据进行预分配,其流程大致如下:

科学工作流建立阶段预分配算法
输入:原始的输入数据集 DS_{ini},任务集 T,数据中心 $DC=\cup_{i=1,2\ldots}\{dc_i\}$
输出:数据集对数据中心的映射

```
1  Divide all datasets into NFD and FD;
2  FOR (each d_i∈FD) d_i→DC_k, k=1,2,…,m
3  FOR (each d_i∈NFD) calculate d_i's depLink
4  FOR (each d_i∈NFD)
5      d_j=d_i.depLink→first;  /*d_j是d_i相关度最大的数据集*/
6      IF(d_j∈FD)
7          DC_j=d_j.dc;
8          IF (DC_j.freespace>d_i.size) d_i→DC_j;
9          ELSE d_j=d_i.depLink→next;
10             i--; continue;          /*goto 6*/
11     ELSE IF (d_j∈NFD)
12         IF (d_j.dc!=null)
13             DC_j=d_j.dc
14             IF (DC_j.freespace>d_i.size) d_i→DC_j;
15             ELSE d_j=d_i.depLink→next;
16                  i--; continue;     /*goto 6*/
17         ELSE
18             DC_k=getDataCenterByCapacity();
19             while (DC_k.freespace<d_i.size)
                   DC_k=getDataCenterByCapacity();
20             d_i→DC_k;
21         ELSE PUSH d_i to DC_wait;
               /*d_i和其它数据集相关度都为0*/
22 END FOR
23 FOR (each d_i in DCwait)
24     DC_k=getDataCenterByCapacity();
25     while (DC_k.freespace<d_i.size)
           DC_k=getDataCenterByCapacity();
26     d_i→DC_k;
27 END FOR
```

图 2　科学工作流建立阶段预分配算法

第 1 步(语句 1～5).算法首先将所有的原始数据分为两类,固定数据集 FD 和非固定数据集 NFD,由于固定数据集的数据中心是不能变更的,因此可以将它们预分配到相应的逻辑数据中心 DC_k.而对于在 NFD 集合中的每一个 d_i,按定义 5 计算与其它所有数据集的相关度,并按照相关度的

大小降序排列，加入到它的属性链表 $depLink$ 中，链表中第一个元素即是相关度最大的数据集.

第 2 步(语句 6～22). 判断 d_i 的 $depLink$ 中相关度最大的数据集 d_j 是固定数据集还是非固定数据集. 如果 d_j 是固定数据集且对应的数据中心为 DC_j，若存储空间足够则将 d_i 预分配到 DC_j 上，否则 $d_j=d_i$. $depLink\rightarrow next$，然后重新分析计算. 如果 d_j 是非固定数据集并且已经分配到数据中心 DC_j，则处理方法同上；如果 d_j 是非固定数据集并且还没有分配到数据中心，则根据各个数据中心的计算能力属性 cap_i，找出计算能力最强的数据中心 DC_k，若存储空间足够则将 d_i 预分配到 DC_k 上，不足则查找下一个计算能力强的数据中心，直到将 d_i 放置. 如果与 d_i 相关度最大的数据集不存在，即所有的数据集和它的相关度都为 0，则将 d_i 放置到 DC_{wait} 上，等到最后阶段分配.

第 3 步(语句 23～27). 对 DC_{wait} 中的数据集进行预分配. 对于每一个 $d_i\in DC_{\text{wait}}$，根据各个数据中心的计算能力属性 cap_i，找到计算能力最强的数据中心 DC_k，存储空间足够则将 d_i 预分配到 DC_k 上，不足则查找下一个计算能力强的数据中心，直到将 d_i 放置为止.

预分配完成后，将逻辑数据中心 DC_i 中的数据集映射到相应的物理数据中心 dc_i，这就实现了科学工作流建立阶段的数据放置.

4.2 运行阶段数据放置与任务调度策略

在科学工作流执行阶段，子任务执行可能会产生大量的中间数据集，这些数据集可能被其它或后续的子任务使用，由于这些中间数据集规模可能非常庞大，且数据中心之间存在带宽限制，因此仍需要对这些数据集进行合理放置. 基于相关度的数据放置策略将中间数据集放置到与它相关度最大的数据中心上，如果该中心存储空间不足，则按照该策略中的 Adjustment 算法对全局数据集进行调整.

定义 8. 数据集 d_k 和数据中心 dc_m 的相关度 $dc\_dep_{mk}=\sum_{i=1}^{N} count(T_i\cap T_k)\times s_k$. 其中 T_i 表示数据中心 dc_m 上数据集 d_i 所需使用的任务集，N 表示 dc_m 上数据集的个数，T_k 指使用数据集 d_k 的任务集集合，s_k 表示数据集 d_k 大小.

定义 9. 调度任务 t_k 在数据中心 dc_m 上执行引起的传输开销设为

$transCost_{mk}=$

$[size(DS_k)-size(DS_k\cap DS_m)+size(DS'_{\text{gen}}-DS'_m)]$，

其中 $size(DS)$ 表示集合 DS 中所有数据集大小之和，DS_k 是任务 t_k 所需使用的数据集，DS_m 包含数据中心 dc_m 上的所有数据集，DS'_{gen} 表示任务 t_k 执行完毕后产生的数据集，DS'_m 表示 DS'_{gen} 中应该放在 dc_m 上的数据集.

传输开销中 $[size(DS_k)-size(DS_k\cap DS_m)]$ 表示 t_k 在 dc_m 上执行需要从其它数据中心调入的数据集大小，$size(DS'_{\text{gen}}-DS'_m)$ 表示 t_k 在 dc_m 上执行完毕后产生的中间数据集向其它数据中心发送的数据集大小.

如果 $transCost_{hk}=\min_{m=1}^{k}(transCost_{mk})$，将 t_k 调度到 dc_h 上执行所引起的传输开销是最低的.

科学工作流运行阶段数据放置与任务调度算法如图 3 所示.

科学工作流运行阶段数据放置算法
输入：已存在的数据集和任务集，数据中心
输出：数据集对数据中心的映射

```
1  FOR (each t_i∈T)
2     Schedule t_i to dc_j where transCost_ji is minimum;
3     Execute t_i on dc_j, generate new tasks T and new datasets DC_gen;
4     Update T;
5     FOR (each d_k∈DC_gen)
6           Calculate dc_dep_nk between d_k and dc_n, n=1,2,…,m;
7           Choose dc_h where dc_dep_hk is maximum;
8           Release all unused datasets on dc_h;
9           IF (dc_h.freespace<d_k.size)
10                Adjustment();
11          ELSE  d_k→dc_h;
12    END FOR
13 END FOR
```

图 3 科学工作流运行阶段数据放置算法

科学工作流运行的时候，从任务集合中选取任务 t_i，根据定义 9 将 t_i 调度到合适的数据中心执行，执行完毕后若产生新的任务和新的数据集，则首先更新任务集合，然后给新产生的数据集选择合适的数据中心放置.

对新产生的中间数据 d_k，根据定义 8 计算其与所有数据中心的相关度 dc_dep_{mk}，选择相关度最大的数据中心分配，如果该数据中心存储空间不足，表明科学工作流已经运行了一段时间，出现了负载不均衡，因此需要对所有数据集进行重新调整.

调整算法如图 4 所示，详细过程如下：

第 1 步(语句 1～31). 对所有数据中心上的数据集进行预分配. 预分配过程和科学工作流建立阶段算法相似，只在两种特殊情况下略有不同. 一种情况如语句 19～23 所示，如果与非固定数据集 d_i 相关度最大的数据集 d_j 未分配，且 d_i 归属物理数据中心 dc_k，则将 d_i 放置在 DC_k 上，若 d_i 也未分配，则按照数据中心的计算能力为 d_i 寻找合适的数据中心放置；另一种情况如语句 27～30 所示，对于 DC_{wait} 中的每一个数据集，处理方法和前一种情况相同.

第 2 步(语句 32～38). 预分配完成以后，对所有逻辑数据中心 DC_i 和物理数据中心 dc_i 上的每个数据集 d_k 进行对比：若 $d_k\in DC_i$ 且 $d_k\in dc_i$，则不需要移动；若 $d_k\in DC_i$ 且 $d_k\in dc_j(j\neq i)$，则将 d_k 从 dc_j 移动到 dc_i.

调整算法. *Adjustment*()
输入：当前数据集，当前任务集，数据中心dc_i, i=1,2,…,m
输出：将所有数据集映射到dc_i, i=1,2,…,m

```
1   Adjustment()
2     Divide all datasets into NFD and FD;
3     FOR (each d_i→FD)   d_i→DCk, k=1,2,…,m
4     FOR (each d_i→NFD) calculate d_i's depLink;
5     FOR (each d_i→NFD)
6       d_j=d_i.depLink→first;/*d_j是d_i相关度最大的数据集*/
7       IF(d_j→FD)
8         DC_j=d_j.dc
9         IF   (DC_j.freespace>d_i.size)   d_i→DC_j;
10        ELSE d_j=d_j.depLink→next;
11               i--;    continue; /*goto 9*/
12      ELSE IF (d_j→NFD))
13             IF(d_j.dc!=null)
14               DC_j=d_j.dc
15               IF (DC_j.freespace>d_i.size) d_i→DC_j;
16               ELSE d_j=d_i.depLink→next;
17                    i--;    continue; /*goto 9*/
18             ELSE
19               IF(d_i.dc_i!=null), d_i→DC_i;
20               ELSE DC_k=getDataCenterByCapacity();
21                 while(DC_k.freespace<d_i.size)
22                   DC_k=getDataCenterByCapacity();
23                 d_i→DC_k;
24           ELSE  PUSH d_i to DC_wait; /*d_j=null*/
25    END FOR
26    FOR(each d_i→DC_wait)
27      IF(d_i.dc_i!=null), d_i→DC_i;
28      ELSE DC_k=getDataCenterByCapacity();
29           while(DC_k.freespace<d_i.size)
                 DC_k=getDataCenterByCapacity();
30         d_i→DC_k;
31    END FOR
32    FOR(each DC_i, i=1,2,…,m) /*检查映射关系DC_i→dc_i*/
33      FOR(each d_k→DC_i)
34        IF (d_k.dc==dc_i) Continue;
35        ELSE  d_k→dc_i;
36      END FOR
37    END FOR
38  END Adjustment()
```

图 4　调整算法

5　实验分析

5.1　实验环境和设置

为了验证基于相关度的数据放置策略效果，在“天河”集群上建立了一个包含 80 个节点的测试平台，每一个节点包含一个 Intel Xeon E5540 2.53GHz 的四核 CPU. 为了模拟云计算平台，在每个节点上安装 Xen 并在上面创建了虚拟集群以模拟数据中心；为每一个 CPU 核创建一个带有存储空间的计算实体，每个数据中心包含 16 个计算实体，于是共有 20 个数据中心；为了对数据进行管理，在每个数据中心上安装了 Apache HDFS，并运行 SwinDeW-C (Swinburne Decentralised Workflow for Cloud)[13]用来解释和执行工作流.

由于实际的科学工作流(如大气科学工作流)不能全方位检测基于相关度的数据放置策略策略的效果，只能部分验证，因此本文将采用模拟的、可定制的科学工作流来测试基于相关度的数据放置策略. 通过分别改变科学工作流的数据集和任务的数量来控制科学工作流的复杂度；通过改变上界与下界来控制数据集大小的取值范围；同样，固定数据集的比例和数据中心的数量也可以进行调整. 实验过程中，为了保证结果的可靠性，每一个科学工作流在保持配置和云平台环境不变的情况下，运行 300 次后取平均值作为测试结果.

为了说明本文所提数据放置策略的效果，实验对比了 3 种数据放置策略，分别是 Random、Cluster 和本文所提出的基于相关度的数据放置策略.

Random 策略：输入数据集在建立阶段随机的放到其中一个数据中心，如果是固定数据集则放置到指定的数据中心；运行阶段，如果空间足够，产生的中间数据集则存放在本地数据中心，否则随机放置到其中一个数据中心. 在网格、集群系统中，产生的中间数据集就是存放在本地或者随机放在存储空间富余的结点上.

Cluster 策略：在文献[11]中提到的数据放置策略. 在建立阶段，把所有的输入数据分为 K 个数据集合，把这 K 个数据集合放置到合适的数据中心；在运行阶段，把新产生的数据集放置到合适的数据中心.

基于相关度的数据放置策略：本文所提的数据放置策略，在文章后续部分所有图示中用 Data-dependence 表示该策略

5.2　测试结果及分析

5.2.1　数据集数量变化对结果的影响

图 5 显示的是当数据集数量增加时，数据移动

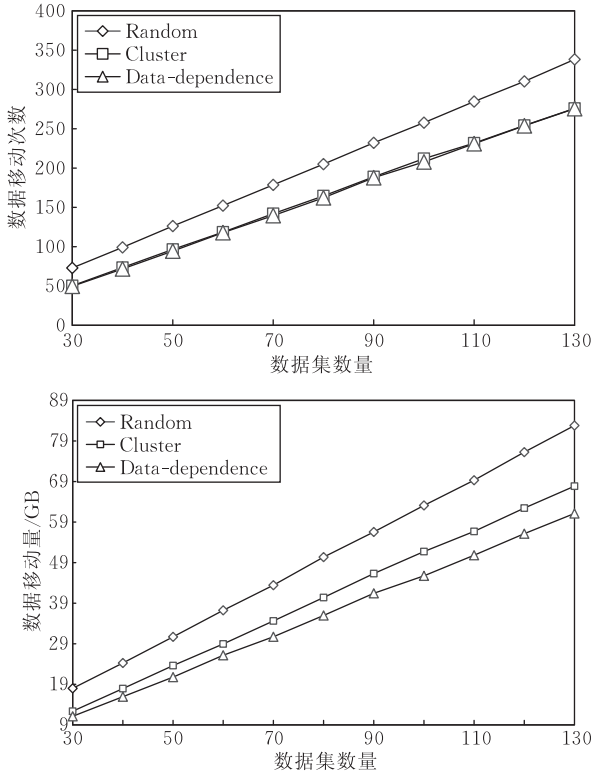

图 5　数据集数量变化对数据移动次数和移动量的影响

次数和数据移动量的变化趋势.实验设定如下:科学工作流任务量 N 和数据集数量 N 取相同的值,数据集的变化范围设为 1～500 MB,固定数据集比例为 20%,数据中心为 15 个.

实验结果表明,随着数据集数量的增多,本文所使用的策略和 Cluster 策略在数据移动次数上相差无几,相比 Random 策略,数据移动次数明显减少;数据移动量方面,3 种策略随着数据集规模增加而呈现出近似的线性增长,Random 策略效果最差,Cluster 策略次之,基于相关度的数据放置策略则比 Cluster 策略在数据移动量少约 10%.

原因分析:Cluster 策略按照最大数量的原则将数据集聚集在同一个数据中心,而基于相关度的数据放置策略按照最大流量的原则将数据集聚集在同一个数据中心;同时,在运行阶段 Cluster 策略将任务调度到包含数据个数最多的数据中心,基于相关度的数据放置策略则将任务调度到数据量最多的数据中心,所以任务执行时,基于相关度的数据放置策略引起的数据传输量就会明显减少.

5.2.2 数据集大小取值范围变化对数据移动量的影响

图 6 表示数据集大小幅度改变时数据移动量的变化趋势.实验设定如下:科学工作流任务量为 80,数据集数量为 80,固定数据集比例为 20%,数据中心为 15 个,数据集大小的平均值是 250 MB.

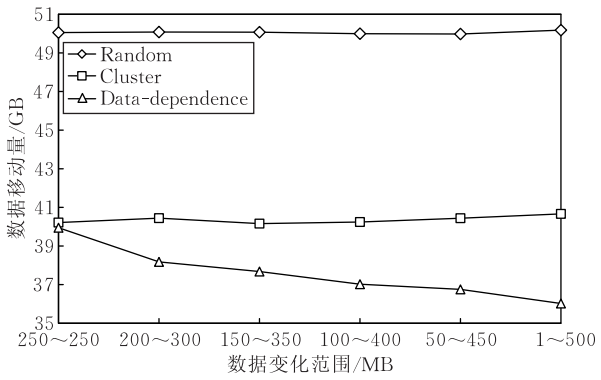

图 6 数据集大小变化幅度对数据移动量的影响

实验结果表明,当数据平均值相同时,Random 和 Cluster 策略的数据移动量变化幅度很小,在某一平均线上下浮动,而基于相关度的数据放置(Data-dependence)策略则不同,数据集大小的变化幅度越大,数据移动量越少.

原因分析:若数据大小变化幅度很大,则 Data-dependence 策略在大多数情况下会选择移动规模小的数据集,这样就降低了数据移动量.

5.2.3 固定数据集比例的改变对数据移动量的影响

图 7 表出的是固定数据集比例改变时数据的移动量的变化趋势.实验设定如下:科学工作流任务量为 80,数据集个数为 80,数据中心为 15 个,数据集的变化范围是 1～500 MB.

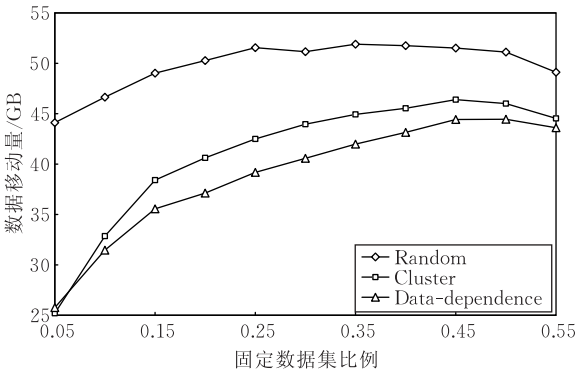

图 7 固定数据集比例对数据移动量的影响

实验结果表明,随着固定数据集比例上升,3 种策略的数据移动量都是先增后减.从图 7 中可以看出 Random 策略数据移动量最大,Cluster 比起 Random 策略性能有很大提高,而相比 Cluster,当固定数据集比例超过 15%后 Data-dependence 策略又有较大提高.

原因分析:随着固定数据集比例的上升,因为固定数据集的因素,某些任务只能调度到特定数据中心执行,如果这些任务使用了非固定数据集,那么非固定数据集的移动次数会不断上升,导致传输的开销逐渐增大,导致 3 种数据放置策略的数据移动量都呈上升趋势;当固定数据集达到一定比例后,非固定数据集的数量越来越少,3 种数据放置策略传输开销都在减少.在 Data-dependence 策略中,若任务中存在固定数据集,则该任务中使用的非固定数据集很有可能与固定数据集放置在同一个数据中心,相比 Cluster 明显减少了数据移动量.

5.2.4 数据中心数量变化对数据移动量的影响

图 8 给出的是当数据中心数量改变时数据移动

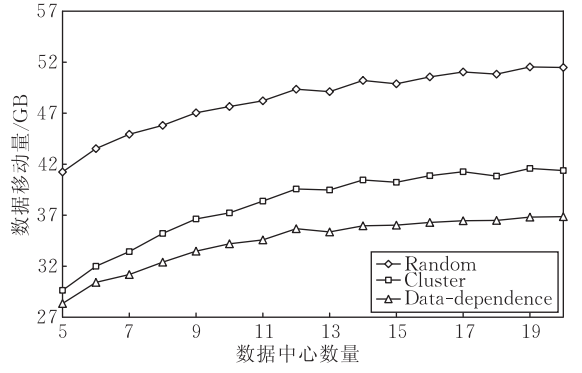

图 8 数据中心数量对数据移动量的影响

量的变化趋势. 实验设定如下：科学工作流任务量为80，数据集数量为80，固定数据集比例20%，数据集的变化范围是1～500 MB.

实验结果表明，随着数据中心数量的增多，3种策略的数据移动量都在逐步上升. 可以明显看出，Random效果最差，Data-dependence策略比起Cluster也有很大的性能提升.

原因分析：随着数据中心数量的增多，平均每个数据中心分得的数据集将减少，任务执行时调用其它数据中心数据集的可能性增加，导致数据传输量上升. Data-dependence策略要优于Cluster策略，再次表明考虑数据集大小的计算调度方式可以减少数据传输流量.

5.3 小　结

当数据集的变化范围为1～500 MB、固定数据集比例为20%及数据中心为15时，随着数据集数量的增多，本文所使用的基于相关度的数据放置(Data-dependence)策略相比Cluster策略在数据移动量方面减少约10%；当任务量为80、数据集数量为80、固定数据集比例为20%、数据中心为15及数据集大小的平均值是250 MB时，Random策略和Cluster策略的数据移动量分别在50 GB和40 GB上下浮动，而基于相关度的数据放置策略的数据移动量则呈递减趋势，数据集大小上下限之差每增大100 MB，数据移动量减少2%～3%；当任务量为80、数据集个数为80、数据中心为15及数据集的变化范围是1～500 MB时，Cluster比起Random策略数据移动量有了大幅度减少，而相比Cluster策略，基于相关度的数据放置策略在固定数据集比例在15%～40%时，数据移动量减少约10%～11%；当任务量为80、数据集数量为80、固定数据集比例20%及数据集的变化范围是1～500 MB时，随着数据中心数量的增加，基于相关度的数据放置策略比起Cluster策略在数据移动量上减少7%～11.5%，而且随着数据中心数量的不断增多，这个比率将持续增长.

综合以上实验结果，基于相关度的数据放置策略在降低数据传输开销方面要明显优于Cluster策略和Random策略. 所以，基于相关度的数据放置策略可以有效提高科学工作流的执行效率.

6　结论与展望

本文首先对科学工作流在云平台上的执行过程做出了分析，指出了多个数据中心之间的数据传输开销是制约科学工作流执行性能的一个瓶颈，并在此基础上提出了一种基于相关度的数据放置策略，该策略综合考虑数据相关、数据集大小、固定数据集比例及数据中心计算能力. 通过与其它数据放置策略进行实验对比，结果表明这种数据放置策略可以有效提升科学工作流的执行效率，减少跨数据中心之间的数据传输流量.

下一步准备在本文工作的基础上进行副本机制的研究. 注意到云计算平台收费的特点，在综合考虑计算代价和存储代价的基础上加入副本机制，进一步提升科学工作流的执行性能.

参　考　文　献

[1] Lin C, Lu S Y, Fei X B, Chebotko A, Pai D, Lai Z Q, Fotouhi F, Hua J. A reference architecture for scientific workflow management systems and the view soa solution. IEEE Transactions on Service Computing, 2009, 2(1): 79-92

[2] Ren K J, Chen J J, Xiao N, Song J Q. Building quick service query list (QSQL) to support automated service discovery for scientific workflow. Concurrency and Computation: Practice & Experience, 2009, 21(16): 2099-2117

[3] Weiss A. Computing in the cloud. ACM Networker, 2007, 11: 18-25

[4] Chervenak A, Deelman E, Livny M, Su M H, Schuler R, Bharathi S, Mehta G, Vahi K. Data placement for scientific applications in distributed environments//Proceedings of the 8th IEEE/ACM International Conference on Grid Computing. Washington, USA, 2007: 267-274

[5] Singh G, Vahi K, Ramakrishnan A, Mehta G, Deelman E, Zhao H N, Sakellariou R, Blackburn K, Brown D, Fairhurst S, Meyers D, Berriman G. B, Good J, Katz D S. Optimizing workflow data footprint. Scientific Programming, 2007, 15(7): 249-268

[6] Du Z H, Hu J K, Chen Y N, Cheng Z L, Wang X Y. Optimized QoS-aware replica placement heuristics and applications in astronomy data grid. The Journal of Systems and Software, 2011, 84(7): 1224-1232

[7] 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, USA, 2008: 1-12

[8] Gu Y, Grossman R. Toward efficient and simplified distributed data intensive computing. IEEE Transactions on Parallel And Distributed Systems, 2011, 22(6): 974-984

[9] Gu Y, Grossman R. UDT: UDP-based data transfer for high-speed wide area networks. Computer Networks, 2007, 51(7): 1777-1799

[10] Warneke D, Kao O. Exploiting dynamic resource allocation for efficient parallel data processing in the Cloud. IEEE Transactions on Parallel and Distributed Systems, 2011,

22(6): 985-997

[11] Yuan D, Yang Y, Liu X, Chen J J. A data placement strategy in scientific cloud workflows. Future Generation Computer Systems, 2010, 26(8): 1200-1214

[12] Zheng Pai, Cui Li-Zhen, Wang Hai-Yang, Xu Meng. A data placement strategy for data-intensive applications in cloud. Chinese Journal of Computers, 2010, 33(8): 1472-1480(in Chinese)

(郑湃, 崔立真, 王海洋, 徐猛. 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. 计算机学报, 2010, 33(8): 1472-1480)

[13] Liu X, Yuan D, Zhang G F, Chen J J, Yang Y. SwinDeW-C: A peer-to-peer based cloud workflow system//Furht B, Escalante A eds. Handbook of Cloud Computing. Berlin, Germany: Springer, 2010: 309-332

LIU Shao-Wei, born in 1987, M. S. candidate. His current research interests include high-performance computing, cloud computing and scientific workflow.

KONG Ling-Mei, born in 1981, M. S., engineer. Her current research interests include network security, scientific workflow, cloud computing and rough sets.

REN Kai-Jun, born in 1975, Ph.D., associate professor. His current research interests include Web service composition, workflow, cloud computing and high performance computing.

SONG Jun-Qiang, born in 1962, professor, Ph. D. supervisor. His current research interests include numerical weather prediction, large-scale parallel applications and CPU/GPU hybrid heterogeneous computing.

DENG Ke-Feng, born in 1985, Ph. D. candidate. His current research interests include high-performance computing, cloud computing and scientific workflow.

LENG Hong-Ze, born in 1982, Ph. D. candidate. His current research interests include parallel computing and numerical weather prediction.

Background

With the development and progress of cloud computing technology, the data-intensive applications running on cloud-supported platform have received more and more attention. Especially, scientific workflow based on cloud computing technology can facilitate interdisciplinary scientists to collaboratively solve scientific problems by providing more advanced computing techniques. However, running scientific workflows not only requires to access TB-scale or PB-scale datasets but also produces a large volume of derived datasets including intermediate and final results. As such, the total volume of involved datasets by a scientific workflow might become extremely large and it's impossible to upload all of the datasets to only one data center, or each data center. In this scenario, datasets-transmission across distributed data centers cannot be avoided during scientific workflow's execution. Further, considering the limited bandwidth and storage capacity of data center, the datasets need to be reasonably assigned to the corresponding data centers before the workflow's execution in order to minimize the total data transferring across data centers and guarantee the higher efficiency of executing a scientific workflow. Nevertheless, we have noticed that little existing work has well addressed this issue. This paper presents an efficient data placement strategy. Compared with other existing ones, the proposed data placement strategy can effectively reduce data traffic among data centers, improve the execution efficiency of scientific workflows, and save the cost of leasing the cloud resources.

This work is partially supported by the National Nature Science Foundation of China under Grant No. 60903042 and the National High Technology Research and Development Program (863 Program) of China under Grant No. 863-2010AA012404. The main aim of these projects is to improve the efficiency and reliability of running workflows on distributed high performance platforms by utilizing cloud computing technologies.

Prior to this work, our research group had published more than 40 papers in high quality journals and at prestigious international conferences, including IEEE Transactions on Services Computing (TSC), Concurrency and Computation: Practice & Experience(CCPE), Journal of Supercomputing (JSC), KSII Transactions on Internet and Information Systems, and Grid 2011, CACHES 2011, CSCWD 2011, HPCC 2010, ICWS 2009, SCC 2008, etc.. Especially, Dr. Ren as a co-author of this paper has received Excellent Doctoral Dissertation Award from PLA and CCF which was also a product of these projects and has tutored the basic idea of this paper.