自适应云端的大规模导出子图提取算法

郭 鑫[1] 董坚峰[2] 周清平[1]
(吉首大学软件服务外包学院 张家界 427000)[1] (武汉大学信息资源研究中心 武汉 430072)[2]

摘 要 针对现有云计算平台资源随机调配与传统导出子图挖掘效率较低等问题,进一步提升云计算平台中资源整合利用效率与大规模导出子图挖掘效率,提出了一种自适应云端的大规模导出子图提取算法,以解决资源优化利用与海量图挖掘等问题。首先介绍了云计算概念与导出子图挖掘相关概念以及问题描述;接着根据MapReduce并行处理模型设计了一种自适应任务动态分配算法SAC_TA(Self Adaptive Cloud Dynamic Allocation),它根据计算任务自适用分配系统资源以达到成本消耗的最优;并设计出自适应云端框架,然后基于自适应云端提出了大规模导出子图挖掘算法SFGFF(SAC_TA、Find_VE、G_F1、FindPartFG、FindAllFG),它共分为4个阶段的挖掘,将所有算法应用到自适应云端中可构成整个导出子图挖掘体系;最后在人工模拟数据与真实环境数据下进行了试验,结果表明,自适应云端运行良好,算法有效可行,具有较高的加速比与运行效率,能有效满足大规模频繁导出子图挖掘的需求。

关键词 大数据,数据挖掘,云计算,导出子图,子图同构
中图法分类号 TP311 **文献标识码** A

Large Scale Induced Subgraphs Mining Algorithm on Self Adaptive Cloud

GUO Xin[1] DONG Jian-feng[2] ZHOU Qing-ping[1]
(School of Software Service Outsourcing, Jishou University, Zhangjiajie 427000, China)[1]
(Center for Studies of Information Resources, Wuhan University, Wuhan 430072, China)[2]

Abstract Aiming at the current puzzles of random resource allocation of cloud computing platform and lower mining efficiency of traditional induced subgraph, promoting the efficiency of resource integration and using of cloud computing platform and large-scale induced subgraph mining, the paper put forward an algorithm of large-scale induced subgraph extraction for self-adaption cloud to solve the problems of resource optimal utilization and massive graph mining. The paper firstly introduced the relevant concepts and problem description of cloud computing and induced subgraph mining, then designed an algorithm SAC_TA of self-adaption task dynamic allocation according to MapReduce parallel processing model, which can comput task self-adaption allocation system resources to reach the optimum of cost wasting, meanwhile designed the self-adaption cloud framework. On the basis of the framework, the paper put forward the massive induced subgraph mining algorithm SFGFF, which includs four stages of mining. And while applying all the algorithms to self-adaption cloud, the whole induced subgraph mining system can be constructed. The experimental result of manual simulation data and real environment data shows that the self-adaption cloud runs well and the algorithms are efficient and feasible, and have higher speed-up ratio and operating efficiency to satisfy the demand of massive frequent induced subgraph mining.

Keywords Big data, Data mining, Cloud computing, Induced subgraph, Subgraph isomorphism

1 引言

在大数据[1,2]环境下,数据表现形式多种多样,从简单的集合、序列、文本等到音视频数据、社交网络媒体、感知数据等,数据的规模越来越大,复杂度也越来越高。图作为复杂数据类型之一,在许多应用场景中扮演着重要的角色,如社交网络好友推荐、计算网络分析、生物医药数据分析等,这些领域中的数据都可以抽象成图,并用于进一步分析。

在图数据库中挖掘有用的知识已经成为热点研究课题之一,国内外专家学者们也提出了很多相关算法,并且这些算法效果良好。在频繁子图挖掘方面,文献[3]提出了一种在不确定图中挖掘频繁子图模式的算法,其利用基于Apriori性质的期望支持度与深度优先搜索策略来挖掘频繁子图,降低了子图同构复杂度。然而在大多数情况下,图挖掘算法产生的大量频繁子图往往是无用的,譬如在一个简单化合物集合CA中,通过频繁子图挖掘可以得到接近100万个频繁子图,这些

到稿日期:2013-07-25 返修日期:2013-12-06 本文受湖南省工业支撑计划重点项目(2012GK2006),湖南省教育厅科学研究项目(12C0291,11C1051),生态旅游湖南省重点实验室开放基金项目(JDSTLY201206),湖南省图书馆学会2013-2014年度重点课题(XHZD1007)资助。
郭 鑫(1984-),男,硕士,讲师,主要研究方向为数据挖掘与并行计算等,E-mail:jianghai079@126.com;董坚峰(1977-),男,博士,讲师,主要研究方向为数据挖掘与信息管理等;周清平(1966-),男,博士,教授,主要研究方向为云计算与量子计算等。

输出子图很庞大，分析利用依然很困难，同时也将需要较大的时间成本。因此，文献[4,5]分别提出了针对频繁闭子图的挖掘算法，减少获得全部频繁子图所需的子图同构次数，降低了计算量。文献[6]提出了极大子图挖掘算法，采用标号图属性信息，定义判断图同构条件，以减少同构次数。文献[7]提出了挖掘频繁跳跃模式，可以大幅度减少输入模式数量并且具有抗噪声与干扰的能力。在导出子图挖掘方面，文献[8]提出了频繁导出子图挖掘算法，基于顶点扩展策略，提出两种操作减少子图生成个数，从而缩减子图搜索空间，提高算法运行的效率。文献[9]提出了一种基于快速指数与组合边界的导出子图挖掘算法，给出特定的导出子图同构算法，提高同构效率。这些算法都是为了解决特殊频繁子图生成问题，降低子图生成数量，在一定程度上提高了图挖掘效率。

但在实际应用中，数据量往往是巨大的，并且结构复杂多变，传统的串行图挖掘算法虽然可以解决问题，但是运行效率较低，特别是在大规模的图挖掘应用中表现尤为明显。传统的串行图挖掘算法主要存在两种性能的瓶颈：一种是候选子图的生成，另外一种是候选子图的支持度计算等。这两个过程都需要进行大量的循环扫描与子图匹配操作，而子图同构又是一个 NP 完全问题，计算复杂程度很高，特别是在大规模的复杂图挖掘应用中，子图同构需要消耗大量的用户时间。因此，传统的串行图挖掘算法已无法满足用户的需求，将云计算技术应用于图挖掘算法中是一种可行的解决方法，通过改变图挖掘串行执行方式与增加算法所需的计算资源来提高图的挖掘效率。

云计算[10]是并行计算、分布式计算等先进计算机技术与网络技术的产物，具有普遍适用性与高效性，可以应对海量数据挖掘带来的挑战，解决传统数据库面临的大规模数据处理等问题。云计算平台中大规模数据处理模型主要包括基于 MapReduce 的分布式编程模型[11]与基于消息通信的 BSP 模型[12]。Google 的 Pregel[13]与 Yahoo 的 Giraph[14]是 BSP 模型的代表，但此类模型仍然处于系统研发阶段，应用并不成熟。Hadoop[11]与 HOP[15]系统是 MapReduce 模型的代表，同时还包括基于 Hadoop 的扩展：Haloop、Prlter、Twister 及各个行业的云模型[16-19]等。MapReduce 模型的执行流程简单，主要包括：数据输入、任务分配、键值对的生成、消息发送、中间数据的调用、Reduce 函数的执行与最终结果输出等过程。在应用时无需考虑分布式存储、网络通信与容错处理等因素，只需考虑任务的分配策略与 MapReduce 函数对的设计。因此本文将基于该模型提出资源优化分配的自适应云端及大规模导出子图提取算法。本文的工作如下：

1)改进现有云计算平台中任务随机分配策略，考虑到云计算处理时资源调度的复杂性，提出一种启发式任务动态分配算法，根据应用自适用分配计算资源以达到成本消耗的最优，并设计出包含两个任务调度队列的自适应云端架构。

2)为了解决传统频繁子图挖掘算法在大规模数据应用中的性能瓶颈问题，提出一种基于 MapReduce 模型的导出子图挖掘算法，其共分为 4 个主要步骤，采用并行与串行相结合的方式，并将算法应用于自适应云端中，构成整个导出子图挖掘框架。

3)通过大量实验验证了自适应云端与挖掘算法的可行性，在大规模数据挖掘应用中，本文算法运行效率提升显著。

2 问题描述与相关概念

自适应云端的大规模导出子图提取是指在给定图数据库中挖掘出满足一定支持度阈值的导出子图。在介绍具体的算法之前首先给出相关概念的定义。

定义 1(标号图[20]) 令五元组 $G=(V,E,\Sigma,\lambda,C)$ 表示一个标号图 G，其中 V 表示标号图的顶点集合，E 表示标号图边的集合，Σ 为类标记的集合，映射 $\lambda:(V\cup E)\rightarrow\Sigma$ 表示 λ 每个顶点与每条边分配一个标记，C 表示标号图的类别。

定义 2(子图同构与真子图[20]) 给定两个标号图 G 与 G'，图 G 到 G' 的子图同构存在一个单射函数 $f:V\rightarrow V'$，满足如下条件：

1) $\forall u\in V,\lambda(u)=\lambda'(f(u))$

2) $\forall(u,v)\in E,(f(u),f(v))\in E'$ 且 $\lambda((u,v))=\lambda'((f(u),f(v)))$

如果 G 到 G' 存在子图同构，则称 G 为 G' 的子图，G' 为超图，记为 $G\subseteq G'$，若 $G\subseteq G'$ 且 $G\neq G'$，则称 G 为 G' 的真子图。

定义 3(导出子图) 设 G 为 G' 的子图，满足 $V\subset V'$ 和 $E\subseteq E'$，且 E 包括 E' 中 V 顶点间的所有边，则称 G 为 G' 的导出子图。

定义 4(标号图数据库) 令 $GDB=\{G_1,G_2,\cdots,G_n\}$ 表示一个标号图的数据库，在图数据库中每个图分配一个 id 号，每个 id 号对应一个图的 String 表示。

定义 5(图的支持度) 给定图数据库 GDB 与图 G，图 G 在 GDB 中的支持度 Sup_G 定义为：

$$Sup_G=\frac{|\{G'\in GDB|G\subseteq G'\}|}{GDB}$$

其中，G 子图同构于 G'，即包含所有图 G 的原图数量与总数量的比值定义为图的支持度。

定义 6(频繁导出子图) 给定图数据库 GDB 与最小支度阈值 s，G 为频繁导出子图当且仅当 G 为导出子图，且满足 $Sup_G\geqslant s$。

图 1 给出了包含 3 个标号图 G_1、G_2、G_3 的图数据库 GDB，图 G 为 G_1 的导出子图，同时也是 G_3 的导出子图，图 G 在数据库 GDB 中的出现次数为 2，若最小的支持度阈值设置为0.5，根据支持度的定义可知，图 G 的支持度为 $2/3>0.5$，因此图 G 为频繁的导出子图。

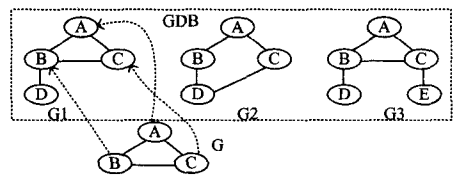

图 1 导出子图与图数据库

本文研究的问题可以准确描述为：在给定大规模图数据库 GDB 与最小支度阈值 s 的情况下，设计一个资源利用最优的自适应云端框架并基于 MapReduce 模型设计出频繁导出子图提取算法，并将此算法应用于该平台中，并行挖掘出所有的频繁导出子图，以提高图挖掘的效率。

3 自适应云端的大规模导出子图提取算法

3.1 自适应云端

自适应云端是基于云计算平台的一种改进，以适应大规

模数据挖掘的需求，并进一步提升云计算平台中资源整合利用的效率。传统基于 MapReduce 并行处理模型的云计算平台中 Master 节点会对输入的任务进行分割预处理，将分割后的任务抽象成一系列的(Map-Reduce)操作对，其中，Map 主要完成数据的过滤操作，Reduce 主要完成数据的聚焦操作，数据的输入输出模式均以键值对的形成存储；同时，用户在进行算法设计时只需要考虑 Map 与 Reduce 函数对的设计，MapReduce 并行处理模型会自动完成计算结点的分配、资源的整合等，并进行并行处理，最后将计算得到的结果发送给 Master 结点并反馈给用户。在这一过程中，云平台高度的自治为任务的顺利完成提供保障，各个计算结点会将状态参数定时发送给主控结点以证明当前是否空闲，主控结点会根据各个计算结点反馈的信息编制状态信息表，并进行任务的分配，但是这种任务的分配只是简单的随机分配，并没有考虑到整个系统的资源最优化调配，譬如：如何选择最佳计算结点来完成指定的任务、如何根据任务的计算量分配计算结点的数量等。现有的云平台虽然高度自动化，但并没有做到资源的高度最优化利用，在许多实际应用过程中，平台中很多计算结点处于空闲状态或者低效率的利用状态，这是不合理的，因此，本文将提出一种基于自适应云端的任务处理平台，以解决资源低效率利用的问题。

任务调度的最优化问题包括任务完成时间最小化、资源成本消耗的最小化、整体反应时间最小化、计算性能最大化与资源利用最大化等，在这些最优化问题中，要想达到所有的最优是很困难的，但是可以通过改进现有平台设计达到一种或者几种最优是可行的。因此，本文将基于云计算平台设计一种任务动态分配的架构，根据应用自适用分配计算资源来达到成本消耗的最优，这个平台称为自适应云端。提出的大规模导出子图提取算法将基于这个平台进行设计，并应用于实际案例中。

考虑到云计算平台在任务处理时的资源调度复杂性，在设计自适应云端时，将集中考虑资源调度所需的输入数据文件类型、分配的子任务数量、平台中计算结点数量、资源调度成本、数据文件使用成本、回收数据文件成本等主要影响因素，其它因素将忽略，首先给出自适应云端中资源调度的总成本目标函数的参数定义，如表 1 所列。

表 1　参数定义表

PARAMETER	EXPLAIN
$r\in[1,R]$	数据文件种类
$i\in[1,I]$	计算结点数量
$j\in[1,J]$	任务数量
B_j	完成任务的预算时间
x_{rij}	计算结点 i 在起始阶段时，分配给子任务 j 的文件资源 r 的数量
d^t_{rij}	时间段 t 内，子任务 j 对文件资源 r 的需求量
A^t_{rij}	时间段 t 内，在计算结点 i 中分配给子任务 j 文件资源 r 的数量
F^t_{rij}	时间段 t 内计算结点 i 中子任务 j 中回收文件资源 r 的数量
f_{ri}	计算结点 i 中调度文件资源 r 的时间成本
δ^t_{rij}	计算结点 i 中，子任务 j 使用资源 r 的时间成本
β^t_{rij}	计算结点 i 中，从子任务 j 回收资源 r 的时间成本
X(P)	云计算环境中文件资源调度时间成本
Y(P)	任务分配成本
Z(P)	使用与回收文件资源时间成本

根据上述参数设置，自适应云端的资源调度的总成本目标函数可以定义为：

$$Cost(P)=\min\{X(P)+Y(P)+Z(P)\} \tag{1}$$

式中，X(P)、Y(P)、Z(P)分别表示为：

$$X(P)=\sum_{ri} f_{ri}Y_{ri} \tag{2}$$

$$Y(P)=\sum_{rijt} N(\sum_{\tau=1}^{t} A^{\tau}_{rij}-\sum_{\tau=1}^{t-1} F^{\tau}_{rij}+x_{rij}) \tag{3}$$

$$Z(P)=\sum_{rijt}(\delta^t_{rij}A^t_{rij}+\beta^t_{rij}F^t_{rij}) \tag{4}$$

约束条件为：

$$\sum_{\tau=1}^{t} A^{\tau}_{rij}+x_{rij}\geqslant\sum_{\tau=1}^{t-1} F^{\tau}_{rij},\ \forall r,i,j,t \tag{5}$$

$$\sum_{j}(\sum_{\tau=1}^{t} A^{\tau}_{rij}-\sum_{\tau=1}^{t-1} F^{\tau}_{rij}+x_{rij})\geqslant d^t_{rj},\ \forall r,j,t \tag{6}$$

$$\sum_{rit} N(\sum_{\tau=1}^{t} A^{\tau}_{rij}-\sum_{\tau=1}^{t-1} F^{\tau}_{rij}+x_{rij})+\sum_{rit}(\delta^t_{rij}A^t_{rij}+\beta^t_{rij}F^t_{rij})\leqslant B_j,\ \forall j \tag{7}$$

在式(2)中，如果计算结点 i 在任务分配时使用过文件资源 r，则 $Y_{ri}=1$，否则为 0。式(3)为任务分配成本，对于给定的计算结点，首先用分配的资源数量减去回收的数量再加上初始分配的数量最后乘以时间，得到任务分配成本。式(4)将计算结点 i 中使用资源的时间成本与对应的回收资源成本相加得到使用与回收文件资源时间成本。在计算时，考虑到现有计算结点数据资源调度的高效性与稳定性，假设所有调度时间成本为 1，即 $\delta^t_{rij}=1$，$\beta^t_{rij}=1$。式(5)表示对于计算结点 i，分配的资源的数量与初始资源数量必须大于等于回收资源的数量，以保证 $Y(P)$不会出现负数。式(6)表示在一定时间段内，计算结点 i 的资源需求量应大于等于预先设定的资源需求量，在实际运行操作时，根据节点计算情况设置 d^t_{rj}，以满足此条件。式(7)表示子任务分配成本与子任务使用回收资源时间成本之和应小于等于完成子任务的预算时间。

资源调度的精确求解方法主要包括基于图论与整数规划等，该问题被证明是一个 NP 完全问题，精确求解此类问题是比较困难的，需要消耗巨大的系统资源与时间成本，因此本文提出一种基于启发式的近似求解算法来解决云计算平台中的资源调度问题。传统的启发式搜索算法包括基于专用处理器网络、基于处理器数量约束、基于计算集群无限制与基于任务复制等，本文基于自适应云端的资源调度总成本函数与启发式技术提出一种任务分配的算法 SAC_TA，并基于该算法设计出自适应云端的框架结构。

算法 1　SAC_TA

输入：计算任务列表 T_List，参数设置表 1，最小成本阈值 m

输出：最优任务分配方案 TA

1)系统初始化操作函数 InitialOp()，将计算任务列表 T_List 放入到任务调度队列 TL 中，根据参数设置表 1 设置相关参数，其中 J=Length(TL)，B_j 为用户给定的预计时间成本，r=1 表示只有一种输入文件类型。

2)根据式(1)设计任务分配方案 AS=(A_{rij}，F_{rij}，X_{rij})，并假设所有的资源调度时间成本为 1，即设置：$f_{ri}=1$，$\delta^t_{rij}=1$，$\beta^t_{rij}=1$。

3)在初始分配方案中随机生成(A_{rij}，F_{rij}，X_{rij})，然后根据式(5)－式(7)判断生成的方案是否有效，如果有效则根据式(1)－式(4)分别计算 X(P)、Y(P)、Z(P)以及 Cost(P)，否则重新生成分配方案。

4)取其中一个任务分配方案为最优方案，并设定 Cost(P)为最小消耗时间成本 MinCost。

5)设计一个随机轮流选择与轮盘赌选择相结合的算法 ChooseOp()，以尽可能地使得搜索结点与最优结果相接近，并且避免过早地陷入到局部最优。先由轮盘赌根据生成概率 F(C_i)生成两种分配方案

AS_1 和 AS_2，如果无效就重新生成。

6)设计一个多点交叉操作函数 CrossverOp()，并将 AS_1 和 AS_2 进行随机多角度交叉操作，交叉点设置为 Min(Length(AS_1)，Length(AS_2))的三次方根，得到两个新分配方案：AS_3、AS_4。

7)设计一个更新操作函数 UpdateOp()，对新的分配方案进行随机更新，以增强方案的多样性与扩展搜索空间。

8)对两种新的分配方案分别计算 Cost(P)，并与 MinCost 进行对比，如果小于 MinCost 则进行替换，将新的方案作为最优与次优方案，循环执行步骤 4)－8)，直接达到循环结束条件。

9)循环结束条件为：①最优与次优方案 AS_1 和 AS_2 满足最小成本阈值 m，或者②达到一定的循环次数，譬如 10000 次，则取最小 Cost(P)对应的方案作为最优方案 TA。

将传统的云计算平台与上述算法相结合，设计一种自适应云端框架，并将这些框架应用于大规模导出子图的挖掘中，框架如图 2 所示。

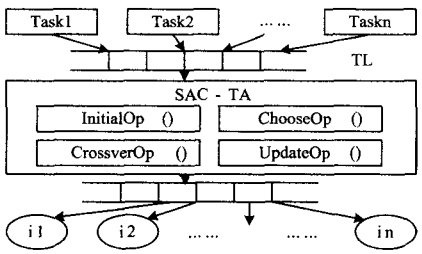

图 2　自适应云端框架图

在自适应云端中，通过两个任务调度队列将大规模的计算任务与相关资源进行缓存，并按照算法进行智能调配。

3.2　算法基本思想

在给定的图数据库 GDB 中进行频繁子图挖掘需要解决图的编码、候选子图的生成、候选剪枝与候选子图支持度计算等问题。图的编码主要采用邻接矩阵、DFS 编码与规范化标记等，必须实现图的唯一标识。候选子图的扩展主要包括边的扩展与点的扩展两种，可以使用逐层搜索与深度优先搜索的方式进行扩展，每次引入一条边或者是一个新的顶点。在候选子图的生成过程会产生大量冗余的候选子图，而计算候选子图的支持度又需要进行大量的数据库扫描与子图同构操作，系统消耗很大，因此对候选子图进行剪枝可以提高挖掘效率，可利用图的反单调性对图进行剪枝。支持度的计算主要涉及子图同构操作，采用哈希表与 k^k 搜索树等减少同构复杂性。

频繁导出子图挖掘算法与传统的频繁子图挖掘算法类似，首先扫描原始图数据库，根据用户给定的最小支持度阈值找出频繁 1 子图与频繁边 $e_1, e_2, \cdots, e_n$，即包含单一结点的图与单一边，并删除非频繁结点，然后对频繁边 $e_1, e_2, \cdots, e_n$ 进行深度优先的扩展生成候选导出子图 G，并计算候选子图的支持度 Sup_G，若 $Sup_G < s$ 即非频繁，则根据反单调性质，图 G 的所有超图也是非频繁的，因此停止对图 G 所有后裔的扩展，否则将候选子图添加到频繁导出子图集中，再由此频繁导出子图进行新一轮的边扩展，并以此类推进行相同循环，直到找出所有的频繁导出子图。在这个过程中，候选导出子图支持度的计算需要频繁扫描数据库，因此耗费大量的系统资源。本文将基于该过程设计新的在大规模数据环境下的导出子图提取算法，首先给出算法所需的性质与引理。

性质 1　若导出子图是非频繁的，那么包含它的所有导出子图也是非频繁的。

证明：假如导出子图 G_1 的支持度为 $s1$，小于用户给定的最小支持度 s，那么导出子图 G_1 是非频繁的，即 $Sup_{G_1} < s$，增加一条频繁边到子图 G_1 中，所得到的新的子图 G_2 在整个图数据库中出现的次数也不会多于原导出子图 G_1 的出现次数，因此，G_2 也是非频繁的，即 $Sup_{G_2} < s$。

引理 1　给出频繁导出子图 G 及其超图 $G'(G \oplus e)$，支持度分别为 Sup_G 与 $Sup_{G'}$，则有 $Sup_G \geqslant Sup_{G'}$。

证明：根据性质 1 易推知。

性质 2　给定导出子图 G，如果存在映射函数 f，使得 $\forall (u_1, v_1), (u_2, v_2) \in E(G)$，满足 $f(u_1) = u_2, f(v_1) = v_2$，且有 $e_1, e_2 \in E(G), e_1, e_2 = (u, v), E(G') = \{G - e_1\}, E(G'') = \{G - e_2\}$，则子图 G' 与 G'' 子图同构。

证明：根据条件，由对任意边的结点有 $f(u_1) = u_2, f(v_1) = v_2$ 可知导出子图 G 是对称的，删除对称图中结点相同的两条边，图中结点保持不变，图的结构也不会发生改变，生成的子图也将是同构的。如图 3 所示，图 G 是对称的，当删除图 G 的两条边生成图 G_1 和 G_2 时，生成的图也一定是子图同构的。

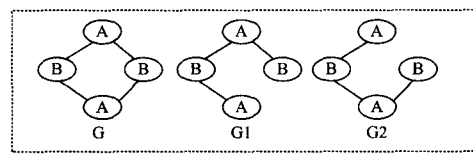

图 3　对称图与子图同构

根据上述性质与引理可以有效减少子图同构操作次数与数据库扫描的次数，提高了算法运行的效率，特别是在大规模的图挖掘应用中，效果显著。

自适应云端的大规模导出子图提取算法主要包括如下几个阶段：

第一阶段　并行扫描图数据库，得到边集合 *E_List* 与结点的集合 *V_List*。

第二阶段　串行构建基于哈希表结构的频繁边集 *FE_List* 与频繁 1 子图 *F1_List*。

第三阶段　并行挖掘局部频繁导出子图。

第四阶段　并行提取全局频繁导出子图。

3.3　算法设计

选取一个计算主机作为主控结点 Driver，控制整个程序的运行。第一阶段挖掘边集与 1 子图，算法描述如下：

算法 2　Find_VE

输入：图数据库 GDB

输出：边集合 E_List 与结点的集合 V_List

Map 操作　在云端中，自适应分配一定数量的计算结点，并行扫描图数据库 GDB，将图进行格式化〈id,G〉，其中 id 为图标号，G 为图字符串表示，分别统计结点与边的数量并作为中间输入结果，结果保存在〈e|v,num〉的键值对中，将此键值对发送到 Reduce 操作。

Reduce 操作　扫描键值对，按节点与边的标识进行升序排列，将相同标识的结点与边进行合并，并分别添加到边集合 E_List 与结点的集合 V_List 中。

在得到所有的边集合与点集合后，通过主控节点 Driver 设计串行算法得到全局的频繁边集 FE_List 与频繁 1 子图 F1_List，算法描述如下：

算法 3 G_F1

输入：包含结点与边信息的 E_List 与 V_List，最小支持度阈值 s

输出：频繁边集 FE_List 与频繁 1 子图 F1_List

1)创建 F1List：ArrayList〈Pair〈int，string〉〉，并对其进行初始化设置。

2)循环扫描边集合 E_List 与结点的集合 V_List，统计每个结点与边的计数，根据最小支持度阈值 s 将所有频繁的点与边添加到 FE_List 与 F1_List。

3)对 FE_List 与 F1_List 按支持度升序，并根据性质 1 与引理 1，将所有非频繁的点与边删除，减少搜索空间。

在得到全局的频繁结点与频繁 1 子图后，自适应云端并行扫描图数据库 GDB，设计算法挖掘局部的频繁导出子图，算法描述如下：

算法 4 FindPartFG

输入：图数据库 GDB，最小支持度阈值 s，频繁边集 FE_List 与频繁 1 子图 F1_List

输出：频繁导出子图集合 PartFG_List，候选图集合 Te_List

Map 操作　并行扫描图数据库 GDB，将数据块映射到不同的计算结点上，每个结点获得 FE_List 与 F1_List 的信息，以频繁结点 id 为键，频繁结点与边为值作为键值对发送到 Reduce 操作。

Reduce 操作　循环扫描键值对，读取 FE_List，进行深度优先扩展(G′◇e)生成候选导出子图，扫描存储图的哈希链表，计算导出子图当前的支持数 n，在子图同构过程中，根据性质 2 判断是否为对称图子图，若是则取消子图同构操作，减少子图同构的次数，根据最小支持度阈值 s，若 $n \geqslant s \times N_{GDB}$，则将 G′加入到 PartFG_List，否则加入到 Te_List。

显然，导出子图集合 PartFG_List 中所有的导出子图都是频繁的，候选图集合 Te_List 的导出子图在局部是非频繁的，但是有可能全局频繁，所以第四阶段将从 Te_List 提取发现的频繁导出子图，将剩余的频繁导出子图添加到最终的频繁集中。算法描述如下：

算法 5 FindAllFG

输入：图数据库 GDB，最小支持度阈值 s，频繁导出子图集合 PartFG_List，候选图集合 Te_List

输出：频繁导出子图集合 FG_List

Map 操作　将候选图集合 Te_List 按结点数量升序，频繁导出子图集合 PartFG_List 按结点数量从小到大依次添加到 FG_List，以最小候选图为键，其超图为值作为键值对〈G，G⊕e〉传递给 Reduce 操作。

Reduce 操作　循环扫描〈G，G⊕e〉，按最小候选图降序排列，并将候选导出子图添加到哈希链表 HL 中，并构成一个子图向量结构，扫描子图向量与图数据库，计算候选导出子图的支持度，并判断是否大于最小支持度阈值，若大于则添加到 FG_List 指定位置中，否则进入下一次迭代。

通过上述基于 MapReduce 的频繁导出子图挖掘算法设计与自适应云端架构的设计，可以有效解决面向大规模的导出子图挖掘应用，整个算法设计体系可以概括为一个云应用平台，如图 4 所示。

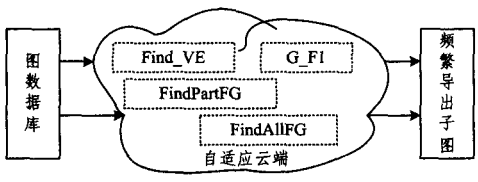

图 4　图挖掘框架

4　实验

4.1　实验环境

实验验证 SFGFF(SAC_TA、Find_VE、G_F1、FindPartFG、FindAllFG)算法的可行性及运行效率等。在学校的云计算实验室搭建实验所需的软硬件环境，因为实验室有其它计算任务，所有实验所得结果可能会与理想状态有一定的出入，但不会影响算法的整体效率。

搭建实验环境的硬件设施如表 2 所列。

表 2　硬件设施表

硬件名称	硬件描述	数量
机架服务器	2×E5560 QC 2.8Hz 处理器，8GB PC3-8500DDR3 内存，2×146GB 10k 硬盘	10 台
刀片服务器	高性能 HS22 刀片服务器，2×E5560 QC 2.8Hz 处理器，8GB PC3-8500DDR3 内存，2×146GB 10k 硬盘	10 台
光纤交换机	中心存储光纤交换机含 24＊4Gb 端口 短波 SFP 模块	2 台
存储阵列	DS5300 存储服务器，28TB 光纤硬盘	1 台
交换机	Cisco 6509，96 电口交换机，双冗余电源	1 台

实验所需的软件环境为 ubuntu linux 11.10 操作系统，Hadoop 版本为 0.20.203，Hadoop Studio，Java 版本为 1.6.x，Eclipse 开发环境。

4.2　实验设计与分析

实验共分为两大部分，第一部分人工生成模拟图数据库进行测试。按复杂程度：0.2、0.5、0.8 及数据库的大小：10M、25M、45M 生成图数据库 GDB_1、GDB_2、GDB_3。在不同的平台下进行实验，一个是单机的基于深度优先搜索的频繁导出子图挖掘算法 CISM[8]，另一个是基于自适应云端的大规模导出子图提取算法 SFGFF。图数据库生成参数设置如下：

GDB_1：D10MT40L500i22V6E6C0.2

GDB_2：D25MT60L800i22V8E8C0.5

GDB_3：D45MT80L1000i22V12E12C0.8

实验结果如图 5 所示。

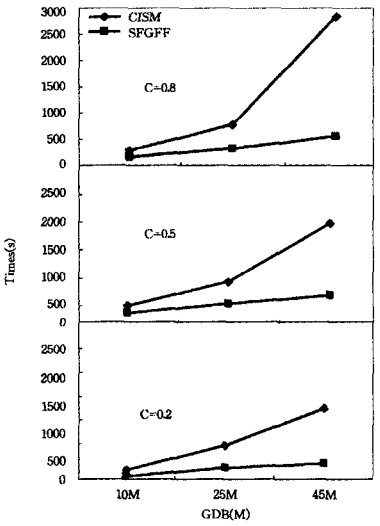

图 5　两种算法的测试结果

上述实验结果图表示在 3 个不同规模与复杂度的图数据库中挖掘频繁导出子图的时间效率，图的横坐标表示数据库

的规模，纵坐标表示实验时间，从下往上 3 个图分别表示不同复杂程度数据库的实验情况。可以看出，SFGFF 算法运行效率较 CISM 有了较大提高，特别是在数据规模较大时，效率提升明显；但是在小规模数据挖掘时，效率差别不大，因为在数据库量较小的时候，自适应云端无法充分利用所有的计算机资源，并且通过主控节点 Driver 可以发现，在所有的 19 个计算结点中，只有不到 1/3 的结点在运行，并且这 1/3 结点 cpu 的使用率不足 30%。因此，为了更好地体现整个自适应云端的加速性能，将数据规模进一步调高，设置如下：$D1=800M$、$D2=500M$、$D3=200M$，在不同的数据库复杂度下：$C=0.2$、$C=0.5$、$C=0.8$，以及在不同的集群结点数量情况下进行实验，加速性能如图 6 所示。

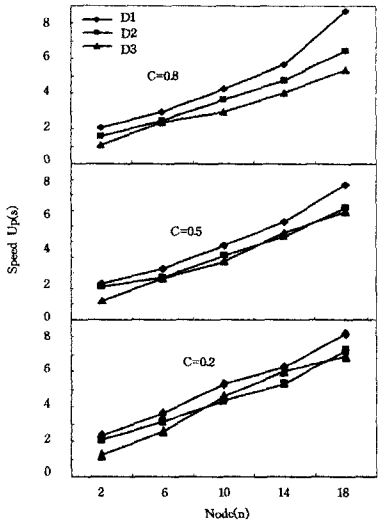

图 6　加速比测试结果

从图中可以看出，随着数据规模的增大，SFGFF 算法的加速性能越来越好，这是因为在自适应云平台中，任务的智能调度、并行挖掘中键值对的合理设置、Driver 结点对全局的控制及高性能硬件的环境为算法顺序地运行提供了良好的保障，这样可以充分地发挥系统潜力，数据规模越大，性能发挥越有优势，加速性能也就越好。并且通过对 Driver 的监测发现，只有当计算机结点与 cpu 的利用率达到 60%以上时，自适应云平台才能体现出它的性能优势。可以注意到，随着计算结点数的增多，算法加速性能越好，但非几何增加，因为结点数目越多，结点之间的通信消耗就越大，数据交换的代价也就越大，同时实验室有其它计算任务在进行，这些都会影响到系统的加速性能，但我们采用的 4GB 光纤交换机与光纤 28TB 光纤硬盘以及系统资源的分配已将这种影响降低至最小。

实验的第二部分在真实数据集中进行实验。实验数据来自于 NCI-HIV 化合物数据，可以从 http://dtp.nci.nih.gov/ 下载得到，在 44000 多个 NCI-HIV 化合物数据当中，把人体 CEM 细胞保护程度分成 3 种类别：强烈保护、一般保护与没有保护，其中强烈保护的有 422 个，一般保护的有 1081 个，剩下的都是没有保护的。不同类别的化合物其复杂程度是不一样的，如强烈保护化合物节点数与边数最大值分别为 203 和 231。一般保护化合物节点数与边数最大值分别为 268 和 289。首先从所有数据中随机抽取 3 种规模的化合物数据：500，800，1200，并进行预处理操作，对化合物的顶点与边进行规范化命名处理，并转化成图的标准表示，在相同的支持度阈值下，分别采用 CISM 与 SFGFF 进行挖掘，得到的实验结果如图 7 所示。

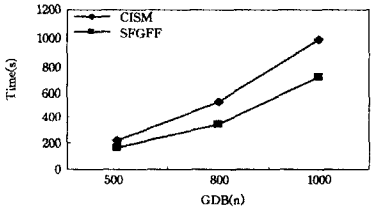

图 7　真实数据测试结果

从实验图可以发现，在真实环境下，本文提出的自适应云端导出子图提取算法 SFGFF 要优于普通的导出子图挖掘算法 CISM，但性能差距并不是很大，这是因为本文的算法更适合于大规模导出子图挖掘应用，这与模拟数据的实验结果是一致的。

最后针对 3 种类别的 NCI-HIV 化合物数据，在不同的支持度阈值下，分别采用两种算法进行实验，得到的实验结果如图 8 所示。

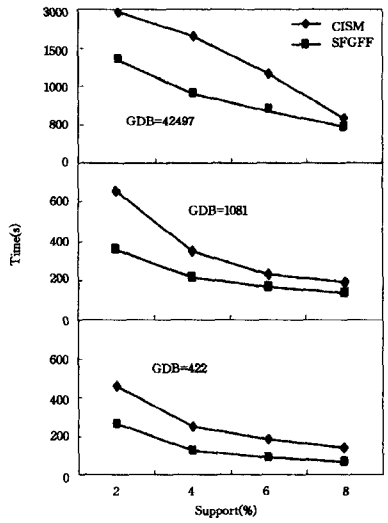

图 8　真实数据测试结果

从图 8 可以看出，当支持度小于 3%时，本文的算法 SFGFF 要明显高于 CISM，支持度越小，产生的频繁导出子图也就越多，所需的计算也就越多，自适应云端可以有效提高此过程的效率。有趣的是，与前面的模拟数据实验结果对比后发现，在真实环境中，数据规模越小，算法所需的运行时间反而越多，这是因为实际数据要比模拟数据复杂许多，一个化合物结构包括几十甚至上百个结点与边，结构越复杂，子图同构所需的时间也就越多，并且呈指数级增长。尽管如此，本文算法要比 CISM 算法的运行效率提升达一倍以上。

综上，自适应云端的大规模导出子图提取算法在人工模拟环境与真实环境中都具有可行性与较好的运行效率，特别是在大规模数据挖掘应用中，运行效率提升显著。

结束语　本文提出了一种基于自适应云端的大规模导出子图提取算法。首先给出了相关概念与问题的描述，其次根据现有云计算平台资源调度状况，提出一种基于启发式搜索的任务分配算法 SAC_TA，以达到成本消耗的最优，并基于该算法设计出自适应云端的框架结构。再次将传统的导出子

（下转第 198 页）

[13] Horowitz R, Varaiya P. Control design of an automated highway system [J]. Proceedings of the IEEE, 2000, 88(7): 913-925

[14] Glavaski S, Papachristodoulou A, Ariyur K. Safety verification of controlled advanced life support system using barrier certificates [C]// Hybrid Systems: Computation and Control, Lecture Notes in Computer Science. 2005: 3414: 306-321

[15] Loos S M, Renshaw D W, Platzer A. Formal verification of distributed aircraft controllers [C]// Hybrid Systems: Computation and Control. Philadelphia, PA, USA, 2013: 125-130

[16] Prajna S. Optimization-based Methods for Nonlinear and Hybrid Systems Verification [D]. California Institute of Technology, 2005

[17] Glover W, Lygeros J. A stochastic hybrid model for air traffic control simulation. LNCS 2993[C]// Hybrid Systems: Computation and Control. Heidelberg: Springer-Verlag, 2004: 372-386

[18] Kouskoulas Y, Renshaw D W, Platzer A. Certifying the safe design of a virtual fixture control algorithm for a surgical robot [C]// Hybrid Systems: Computation and Control. Philadelphia, PA, USA, 2013

[19] Manna Z, Pnueli A. Temporal Verification of Reactive Systems [M]. Springer-Verlag, 1995

(上接第 160 页)

图挖掘算法并行化，提出了基于 MapReduce 模型的大规模导出子图提取算法，其共分为 4 个阶段，每个阶段对应于一个算法：Find_VE、G_F1、FindPartFG、FindAllFG，并将算法应用于自适应云平台中，构成整个挖掘体系。最后通过大量实验可以表明，算法具有可行性、良好的加速性能与运行效率。

未来的工作主要包括：

1)继续优化自适应云端与导出子图提取算法，以进一步提高挖掘性能。

2)从真实环境实验中可以发现，子图同构问题已经严重制约了算法的运行效率，虽然可以通过云计算平台增加计算资源，以达到提高算法运行效率的目的，但并不能从根本上解决子图同构时间复杂度高的问题，特别是在复杂图数据环境下表现尤为明显，因此子图同构问题也将是我们未来持之以恒研究的工作之一。

3)现在是大数据时代，改进现有算法，考虑将自适应云端应用于其它数据类型的挖掘中。

参 考 文 献

[1] 覃雄派，王会举，杜小勇，等. 大数据分析—RDBMS 与 MapReduce 的竞争与共生[J]. 软件学报，2012，23(1)：32-45

[2] TDWI Checklist Report: Big Data Analytics[OL]. http://tdwi.org/research/ 2010/08 Big-Data-Analytics. aspx

[3] 邹兆年，李建中，高宏，等. 从不确定图中挖掘频繁子图模式[J]. 软件学报，2009，20(11)：2965-2976

[4] Zou Xiao-hong, Chen Xiao, Guo Jing-feng, et al. An improved algorithm for mining Close Graph[J]. ICIC Express Letters Journal of Research and Surveys, 2010, 4(4): 1135-1140

[5] 薛冰，张俊峰，郑超. 基于分割图集的频繁闭图挖掘算法[J]. 计算机应用研究，2011，28(1)：61-64

[6] Guo Jing-feng, Chai Ran, Li Jia. Top-down algorithm for mining maximal frequent subgraph[J]. Advanced Research on Industry, Information System and Materials Engineering, 2011, 204-210: 1472-1476

[7] 刘勇，李建中，高宏. 从图数据库中挖掘频繁跳跃模式[J]. 软件学报，2010，21(10)：2477-2493

[8] 刘文艳. 基于深度优先策略的频繁导出子图挖掘算法[D]. 西安：西安电子科技大学，2009

[9] Gupta S, Raman V, Saurabh S. Maximum r-Regular Induced Subgraph Problem: Fast Exponential Algorithms and Combinatorial Bounds[J]. SIAM Journal on Discrete Mathematics, 2012, 26(4): 1758-1780

[10] Lenk A, Klems M, Nimis J, et al. What's inside the cloud? An Architectural Map of the Cloud Landscape[C]// Proceedings of the 2009 ICSE Workshop on Software Engineering Challenges of Cloud Computing. 2009: 23-31

[11] Son J, Choi H, Chung Y D. Skew-tolerant key distribution for load balancing in MapReduce[J]. IEICE Transaction on Information and Systems, 2012, 95(2): 677-680

[12] Valiant Leslie G. A bridging model for parallel computation[J]. Communication of the ACM, 1990, 33(3): 103-111

[13] Grzegorz M, Matthew A H, Bik Art J C, et al. Pregel: A system for large-scale graph processing[C]// Proceedings of the SIGMOD. Indianapolis, Indiana, USA, 2010: 135-145

[14] Avery C. Giraph: Large-scale graph processing infrastruction on Hadoop[C]// Proceedings of the Hadoop Summit. Santa Clara, 2011

[15] Tyson C, Neil C, Peter A, et al. MapReduce Online[C]// Proceedings of the NSDI. San, Jose, California, USA, 2010: 33-48

[16] Islam S, Gregoire J C. Giving user an edge: A flexible cloud model and its application for multimedia[J]. Future Generation Computer Systems, 2012, 28(6): 823-832

[17] Samba A. Logical data models for cloud computing architectures [J]. IT Professional, 2012, 14(1): 19-26

[18] Huang H, Wang L Q. P&P: A combined Push-Pull model for resource monitoring in cloud computing environment[C]// Proceedings of the 2010 IEEE 3rd International Conference on Cloud Computing. Miami, Florida, USA, 2010: 260-267

[19] Tsai Wei-Tek, Sun X, Balasooriya J. Service-oriented cloud computing architectued[C]// Proceedings of the 2010 Seventh International Conference on Information Technology: New Generations. Las Vegas, NV, USA, 2010: 684-689

[20] 王桂娟，印鉴，詹卫许. GC-BES：一种新的基于嵌入集的图分类方法[J]. 计算机科学，2012，39(6)：155-158