基于极大团的边缘云节点聚合算法

朱金彬[1,2] 武继刚[1,2] 隋秀峰[2]
(广东工业大学计算机科学与技术学院 广州 510006)[1]
(中国科学院计算技术研究所计算机体系结构国家重点实验室 北京 100190)[2]

摘 要 组合多个边缘云可以向用户提供更强大的云计算服务,在大量边缘云节点集合中选择适当的节点进行组合是一项具有挑战性的任务。该问题被建模成由云节点作为顶点、节点之间的链路作为边的资源拓扑图。云组合的构建过程等同于在该图中选择子图的过程,这是一个NP完全问题。子图的选择策略是决定云组合性能的重要因素,现有的minStar算法贪心地选择节点之间通信延迟最小的子图,将最优资源分配给当前用户,导致了局部最优和全局性能不良的问题。鉴于此,提出基于极大团的边缘云资源分配算法,提取图中的极大团并将其划分为若干互不重叠的规模较小的完全子图,以子图为单位构建资源块,以资源块为单位进行资源的分配。实验结果表明,与minStar算法相比,新算法将全局最大通信延迟降至原来的50%。

关键词 边缘云,云组合,极大团,资源块,全局最优
中图法分类号 TP301.6 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2018.04.008

Edge Cloud Clustering Algorithm Based on Maximal Clique

ZHU Jin-bin[1,2] WU Ji-gang[1,2] SUI Xiu-feng[2]
(School of Computer Science and Technology, Guangdong University of Technology, Guangzhou 510006, China)[1]
(State Key Laboratory of Computer Architecture, Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190, China)[2]

Abstract Effective combination of edge clouds can offer more powerful computing capacity, which is a promising research direction in cloud computing. It is a challenging work to select multiple edge clouds to combine, because a bad strategy will make a negative impact on the computational power of the obtained group. The problem is generally modeled as a resource topology graph in which cloud nodes are represented as vertices, and the links between nodes are represented as edges. The selection of edge cloud is equivalent to subgraph extraction of the resource topology graph, and this is a typical NP problem. The current minStar algorithm extracts the subgraph with the smallest communication delay between nodes, and then assigns the corresponding resources to the customer. It is a greedy strategy, resulting in local optimal and poor global performance. The proposed algorithm based on maximal clique divides maximal cliques into several smaller but complete subgraphs which do not overlap each other, then constructs resource block in unit of complete subgraph, and assigns the cloud resource in unit of resource block. Compared with minStar algorithm, simulation results show that the global maximum communication delay is reduced by 50% with the proposed algorithm.

Keywords Edge cloud, Cloud combination, Maximal clique, Resource block, Global optimization

1 引言

传统云计算的核心思想是,以高性能计算机和大容量存储设备为基础建立集中式云服务端,向用户提供强大的云计算和云存储服务[1-3]。用户通过网络将传统的本地计算或数据存储等任务全部或部分迁移至远程云端,以获取强大的云服务[4]。然而随着互联网应用的日益丰富,用户终端开始面临大数据的挑战,而传输海量数据到云端极大地增加了网络带宽的负载,造成较大的网络延迟,从而严重影响了用户的服务体验[5]。为了突破有限的网络带宽对云计算性能的限制,实现源数据的就近处理,边缘计算作为新兴的云计算范式应运而生[6-7]。目前,谷歌、微软等大型互联网公司已经将边缘云应用于实际生产环境[8]。

边缘计算模型通过整合大量的地理位置分布式和小规模服务器,以云服务本地化为目标,构建了分布式云计算体系[6-7],解决了传统云计算体系结构对网络依赖性强的不足。分布式云计算体系的架构如图1所示,配备高性能计算机和大容量存储设备的核心云被部署在远程位置,为边缘云提供强大的云服务。边缘云作为中间层,为用户提供灵活、可扩展的本地化云服务[9-10],利用靠近用户的优势直接对源数据进

到稿日期:2017-05-29 返修日期:2017-08-03 本文受国家自然科学基金项目(61672171),广东省教育厅重大科研项目(2016KZDXM052),广东省应用型科技研发专项(重点)(2015B010129014)资助。
朱金彬(1989-),男,硕士生,主要研究方向为云计算,E-mail:jinbinzhu@outlook.com;武继刚(1963-),男,博士,教授,主要研究方向为理论计算机科学、高性能计算、云计算,E-mail:asjgwucn@outlook.com(通信作者)。

行本地处理，将结果同步到核心云并反馈给用户。用户通过以太网或无线网络将全部或者部分计算和数据存储任务从本地迁移至边缘云，以获取低延迟、高可靠的云计算服务。

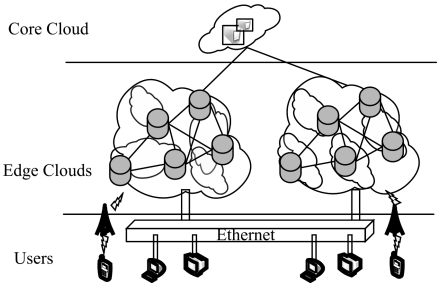

图 1　分布式云体系架构

Fig. 1　Distributed cloud architecture

边缘云作为分布式云计算体系，得到了学术界和工业界的普遍关注和研究[11]。相较于核心云，边缘云的特点是系统相互独立、计算和存储能力有限且资源分散，面对日益复杂的互联网应用，单个边缘云的服务能力很难满足用户对海量数据的处理及实时性的需求[12]。文献[13-16]基于分布式云计算体系结构，对边缘云进行组合，并在此基础上提出若干基于网络性能的边缘计算模型。

边缘云的组合是构建边缘计算模型的第一步，云组合中节点与节点之间的通信延迟对系统性能产生了至关重要的影响，且节点间的最大通信延迟决定了云组合的延迟。因此，选择不同的边缘云进行组合，直接影响了单个云组合的局部最大通信延迟，以及所有云组合的全局最大通信延迟，即边缘云的组合策略是影响云组合网络性能的决定性因素。然而，各边缘云在地理位置上的分散性，使得在大量边缘云节点集合中选择适当的节点进行组合极具挑战性。文献[13]已将边缘云的组合问题建模成由云节点作为顶点、顶点之间的链路作为边的资源拓扑图，云组合的构建过程等同于在该图中选择适当子图的过程。文献[13]以最小化组内局部通信延迟为目标，提出 minStar 算法来实现边缘云的组合，同时证明了以最小化组内节点间的最大通信延迟为目标的子图选择问题是 NP 完全问题。

本文以最小化所有云组合的全局最大通信延迟为目标，提出基于极大团的边缘云节点聚合算法，提取图中的极大团并将其划分为若干互不重叠的规模较小的完全子图，以子图为单位构建资源块，以资源块为单位进行资源的分配。

本文第 1 节总结了分布式云计算体系和边缘云组合的相关研究，重点分析了 minStar 算法的特点和全局性能不良的缺陷；第 2 节提出并设计了基于极大团的边缘云节点聚合算法，并分析算法的设计原理及时间复杂度；第 3 节进行仿真实验，结果证明新算法将全局最大通信延迟降至原来的 50%；最后总结全文并展望未来的工作。

2　相关工作

大数据时代，随着互联网应用的日益丰富和复杂，传统云计算利用具有超强服务能力的云计算中心，但集中式的任务处理方式已无法满足现代应用程序的需求，网络带宽成为云计算性能的瓶颈。而分布式云计算体系将资源分散、计算能力相对有限的边缘云进行整合，充分发挥边缘云靠近源数据和用户的优势，实现源数据的本地化处理。

2.1　分布式云计算体系

系统结构的设计直接影响了分布式云计算体系的性能。文献[17]基于小型数据中心提出的 CamCube 拓扑结构，相较于传统数据中心，可以实现更高的带宽和容错性。文献[18]将虚拟化服务器集群集成到边缘云，并考虑部分用户的移动性，实现了更高的性能和可靠性。雾计算是将云计算范式扩展到网络的边缘，从而实现新一代的应用和服务。文献[19]从雾计算低延迟、位置感知以及广泛的地域分布的角度出发，综述了雾计算在物联网应用中扮演的重要角色。针对系统结构的研究旨在提升分布式云计算体系的通用性、可扩展性及可靠性等，探索具有更高服务质量的分布式云体系结构。

任务迁移作为边缘云的核心技术，旨在解决网络设备的计算和存储能力受限以及访问核心云造成的高网络延迟的问题[6,20]。通过将用户端的计算、存储等任务迁移到附近资源相对丰富的边缘云执行，提升了用户终端的计算能力，节约了用户终端的能耗，并大幅降低了网络带宽的需求，节约了任务处理的成本。文献[21]依据子任务的相关性，将通信密集的子任务尽可能地迁移至同一个或距离较近的边缘云，以进一步降低网络延迟。

位置服务作为边缘云计算的亮点，通过定位用户位置探索边缘云资源的动态分配，实现用户随时随地按需获取丰富多样、灵活高效的云计算服务的目标[22]。另外，在边缘云计算带来便利的同时，数据安全与隐私保护的问题也日益突出并得到了大量的研究。身份认证是云端数据安全存储的重要保障，Chow 等设计了一种结合 TrustCube 和隐式认证的云认证平台[23]。

2.2　边缘云组合

边缘云计算模型将原有集中式云计算中心的部分或全部任务迁移至数据源附近。相较于核心云，边缘云的特点是系统相互独立、计算和存储能力有限且资源分散，面对日益复杂的互联网应用，单个边缘云的服务能力很难满足用户对海量数据的处理和实时性的需求。多个边缘云的有效组合，可以聚合多个云节点的计算核存储能力，为用户提供更优的云计算服务[13-16]。

通过网络组合多个边缘云，可以进一步扩展边缘云的计算和存储能力，为用户提供更加丰富的资源。然而，随着分布式云计算体系规模的不断扩大，边缘云的资源拓扑图日益复杂，边缘云的组合问题面临巨大的挑战。文献[13]首先将该问题建模成资源拓扑图的子图提取问题，然后设计了 minStar（包括 FindMinStar 和 MinDiameterGraph）算法。minStar 算法的核心思想是贪心地选择节点之间通信延迟最小的子图，将最优资源分配给当前用户。这对后到用户是极不公平的，且每次选择最优资源会导致陷入局部最优。另外，当该部分最优资源被耗尽之后，系统只能横跨该子图，将远距离的云节点组合并分配给后到用户，这极大地增加了组合内节点之间的通信延迟，导致了局部最优和全局性能不良等问题。

文献[14-15]继承了文献[13]选择通信延迟最小的子图的思想，并在此基础上进一步优化了边缘计算模型。文献[14]结合虚拟机之间的通信延迟，提出极小通信延迟的虚拟机分配算法。文献[15]提出任务重定向，将任务从负载过重

的边缘云迁移至负载相对较轻的边缘云，其同样是基于文献[13]所提的通信延迟最小的子图思想。而文献[16]默认由边缘云建模成的资源拓扑图是完全图，并在此基础上利用文献[13]所提思想实现了边缘云的组合。

与文献[13-16]不同，本文研究的背景是，由边缘云建模成的资源拓扑图是一般图。本文沿用文献[13]提出的模型，充分分析 minStar 算法全局性能不良的原因；在此基础上，提出基于极大团的边缘云节点聚合算法，以边缘云为单位进行均衡分配，降低了全局通信延迟，同时，基于团的边缘云节点聚合算法保障了网络负载的平衡。

3 问题的定义及组合方法

本文沿用文献[13]设计的模型，将分布式云计算体系中的边缘云子系统建模成一个资源拓扑图。其中，图中顶点代表边缘云节点，图中的边用来表示边缘云节点之间的链路，节点的权值表示该节点对应的边缘云配置的计算资源，边的权值表示该边所连接的两个边缘云之间的通信延迟。在分布式云计算体系中选择若干边缘云节点进行组合的过程，等价于从该资源拓扑图中选择一个子图的过程。在文献[13,24]中已经证明该问题是一个 NP 完全问题。

如图 2 所示，给定资源拓扑图 $G(V,E,w,d)$，其中 V 表示图中的顶点集合，$V=\{v_1,v_2,\cdots,v_n\}$，每个顶点表示分布式云计算体系中的边缘云节点，顶点上的权值表示边缘云节点上部署的资源数量，用 w 表示。E 表示图中的边集合，$E=\{e_{ij},0\leqslant i\leqslant n,0\leqslant j\leqslant n\}$，$e_{ij}$ 表示连接顶点 v_i 和 v_j 的边，且边上的权值表示它们之间的通信延迟，用 d 表示。设子集 $S\subseteq V$，则 $G(S)=(S,E\cap S\times S)$是图 G 的一个子图。$W(S)=\sum_{v_i\in S}w_i$，表示子图 $G(S)$中节点的权值之和，用以表示子图所对应的边缘云中的资源总数。$T(G)=\{S_1,S_2,\cdots,S_m\}$表示从图 G 中提取出的子图集合。

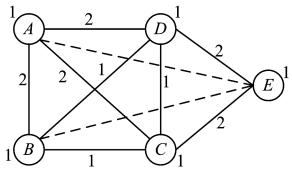

图 2 边缘云节点资源拓扑图

Fig. 2 Topological structure of edge cloud

定义 1(云组合的最大通信延迟) 云组合内任意两个边缘云节点之间通信延迟的最大者，即该云组合对应子图 $G(S)$中的最长边的边长，记作 $D(S)$，则

$$\begin{cases} DS=\max\{d_{ij}x_{ij}\} \\ x_{ij}\in\{0,1\} \\ v_i,v_j\in S \end{cases}$$

3.1 minStar 云组合算法

图 2 所示为某一分布式云计算体系中边缘云节点构成的资源拓扑图。文献[13]设计了启发式算法选择图中节点间通信延迟最小的子图，进而将子图对应的云组合资源分配给用户。首先，FindMinStar 算法获取用户申请的云资源数量 s，分别以每个节点作为起始节点，然后以此节点为中心，按边权值递增的顺序访问其邻接点，同时累加该节点及已访问的邻接点中的空闲资源，当累加得到的空闲资源满足用户需求时停止本次迭代任务，得到对应的子图，当所有节点的迭代过程完毕之后，得到若干子图集合。然后，MinDiameterGraph 算法在 FindMinStar 算法得到的子图集合中选择节点通信延迟最小的子图，进而将该子图对应的云组合资源分配给用户。

定理 1 设 G'是 minStar 算法得到的子图，l_x 为子图 G'最长边的边长，l 为图 G 最长边的边长，则 $l_x\leqslant 2l$。

证明：minStar 算法获取到的子图 G'如图 3 所示，其中，l 为图 G 的最长边的边长。根据三角不等式，$l_x\leqslant l+l'$，从而得到：$l_x\leqslant 2l$。

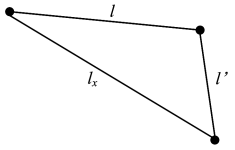

图 3 子图 G'

Fig. 3 Subgraph G'

由定理 1 可知，对于由 minStar 算法获得的云组合，其最大通信延迟为 $2l$。

3.2 基于极大团的云组合算法

文献[13]所提算法优先选择通信延迟最小的子图，导致局部性能较好，但全局性能较差。鉴于此，本文提出基于极大团的边缘云资源分配算法，目标是将该最大通信延迟降为 l。

根据上节的分析，如果系统优先选择子图$\{B,C,D\}$，则当该部分资源耗尽后，系统只能分配$\{A,E\}$给后到的用户，根据定理 1，分布式云计算体系的全局最大通信延迟会大幅增加。如图 4 所示，该资源拓扑图中包含两个极大团$\{A,B,C,D\}$和$\{C,D,E\}$，子图$\{C,D\}$是两个极大团重叠的部分。因此，子图$\{C,D\}$对应资源的耗尽是导致系统全局最大通信延迟大幅增加的直接原因。

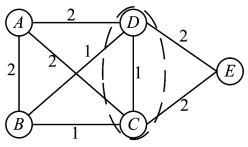

图 4 基于极大团的云联合模型

Fig. 4 Cloud combinatorial model based on maximal clique

因此，本文提出基于极大团的边缘云节点聚合算法。给定图 G，图 G 中的团是一个两两之间有边的顶点集合，它是一个完全子图。如果一个团不被任何一个更大的团所包含，则称该团为极大团。为了进一步方便且形象化地描述算法思想，提出如下定义。

定义 2(重叠度) 某一子图被几个极大团所覆盖或共用，用 SHD 表示。

例 1 如图 4 所示，子图$\{C,D\}$的重叠度为 2，因为它被极大团$\{A,B,C,D\}$和$\{C,D,E\}$所覆盖。子图$\{A,B\}$的重叠度为 1，因为它只被一个极大团所覆盖。

给定一个资源拓扑图，本文提出基于极大团的边缘云节点聚合算法，选择适当的子图($G(S)$)。算法的目标是：

$$\text{minimize } \max\{D(S_k)\}$$

$$\text{s.t.} \begin{cases} S_k \in T(G) & (1) \\ G(S) \text{ is a clique in } G & (2) \\ SHD_i = SHD_j, i,j \in S & (3) \\ W(S) \geqslant s & (4) \end{cases}$$

目标函数表示最小化所有云组合的全局最大通信延迟。约束条件(1)表示以子图为单位进行边缘云的组合；约束条件(2)强调该子图必须为一个完全子图，以保证任意组合中任意两个边缘云节点之间存在链路；约束条件(3)表示云组合内所有节点的重叠度都相同；约束条件(4)保证了分配给用户的资源块所包含的空闲资源满足用户的申请条件。

图 4 展示了本文所提算法的核心思想，即提取图中所有的极大团，并划分极大团为若干互不重叠的规模较小的完全子图，以子图为单位构建资源块，以资源块为单位分配边缘云资源。以图 4 为例进行说明，minStar 算法优先选择子图{B, C, D}，此时局部最大通信延迟为 1。然而，当顶点 C, D 对应的边缘云资源被耗尽后，系统只能组合剩余的边缘云，而此时通信延迟为 4，且全局最大通信延迟为 4。本文提出的算法，首先提取并划分图中所有的极大团，得到{A, B}，{C, D}和{E} 3 个子图，它们之间没有重叠；其次，优先分配非极大团重叠的子图(即子图{A, B}或{E})，此时全局最大通信延迟为 2。新算法避免了因资源耗尽而被迫分配横跨{C, D}的资源，大幅缩减了全局通信延迟。

显然，我们的分配策略保证了任意分配的两个边缘云节点之间均有链路连接，避免了定理 1 所论述的情况。因此，利用所提基于极大团的边缘云资源分配算法得到的所有组云组合的全局最大通信延迟，即为资源拓扑图中的最长边的边长。

4　启发式算法

本文提出的基于极大团的边缘云资源分配算法，共分 3 步完成资源的分配。首先，利用 ExpandMCE 算法提取资源拓扑图中所有的极大团，同时计算各节点的重叠度；然后，利用 CreateComponent 算法获得上述极大团集合，划分极大团集合为若干互不重叠的规模较小的完全子图，并将其组装成资源块；最后，按组内通信延迟递增的顺序，利用 Allocate-Component 算法接收用户提交的资源申请，并搜索符合条件的资源块分配给用户。

4.1　ExpandMCE 算法

(1)算法思想

ExpandMCE 算法是在经典的 Bron-Kerbosch 算法基础上，根据定义 1 中所述内容进行了适度改进。Bron-Kerbosch 算法是经典的极大团求解算法，它的基础形式是一个递归回溯的搜索过程。其核心思想是：首先，初始化集合 RES 为图 G 的全部顶点集合；初始化 $CANV$ 为空集合，其待扩展为极大团；VER 集合负责校验 $CANV$ 是否为极大团。其次，循环从 RES 集合中取出关键(度最大)顶点 v，将其加入 $CANV$，继续迭代 v 的邻接点集合。然后，若集合 RES 和 VER 均为空集，则集合 $CANV$ 中的顶点构成极大团。ExpandMCE 在此基础上设置了 SHD 集合，用于统计所得极大团中每一个节点的重叠度。具体过程是，在一次迭代过程中，每访问节点一次，便将该节点对应的 SHD 值加 1，当本次迭代过程结束时，如果所得子图是极大团，则返回该子图；如果不是极大团，则将各节点的 SHD 值回置为原值。最后，ExpandMCE 得到了图 G 的所有极大团和各节点的 SHD 值。

算法 1　ExpandMCE(RES, $CANV$, VER)

Input: RES(vertex set of graph G), CANV(candidates to construct a maximal clique), VER(auxiliary variable to verify the obtained maximal clique)

Output: maximal cliques attached with SHD

1. if CANV=VER=Ø then
2.　update and output RES and SHD
3. else
4.　drawback SHD one time
5. end if
6. sort(CANV ∪VER) and set the first vertex as a pivot
7. search neighbors of v, named N(v), and CANV′ ← CANV \ N(v)
8. for v∈CANV′ do
9.　CANV← CANV\v
10.　call ExpandMCE(RES∪v, CANV∩N(v, VER N(v)))
11.　set SHD and VER ← VER∪v
12. end for

(2)算法分析

引理 1　ExpandMCE 算法完成了枚举图 G 中极大团和各节点 SHD 的计算，其时间复杂度为 $O(3^{n/3})$。

证明：文献[24]已经证明将剪枝策略应用到 Bron-Kerbosch 算法的时间复杂度为 $O(3^{n/3})$。ExpandMCE 算法是在 Bron-Kerbosch 枚举极大团的过程中插入了 SHD 变量的求解，因此时间复杂度保持不变。

如图 4 所示，ExpandMCE 算法获得的结果是：{A, B, C, D}，{C, D, E}两个极大团以及顶点 A, B, C, D, E, F 对应的 SHD={1,1,2,2,1}。ExpandMCE 算法通过修改 Bron-Kerbosch，利用 SHD 统计各节点的重叠度，完成了基于极大团的边缘云节点聚合算法，即提取资源拓扑图中所有的极大团并计算各节点的重叠度。

4.2　CreateComponent 算法

(1)算法思想

CreateComponent 算法的功能是，将 ExpandMCE 算法获得的极大团划分为互不重叠的规模较小的完全子图。划分的依据是各顶点的重叠度，具体地，将一个极大团中重叠度相同的顶点划分到一组。然后，将各个分组封装成资源块(component)。资源块的数据结构如下：

```
struct component {
  members//块内节点集合
  total_resources//块内资源总数
  idle_resources//块内空闲资源数
  max_delay//块内最大通信延迟
}
```

其中，块内最大通信延迟表示该资源块内边缘云节点之间的最大通信延迟，表现在资源拓扑图中是子图内的最长边的边长。

算法 2　CreateComponent(MCL)

Input: MCL(maximal cliques)

Output: Components consisted of vertices in MCL

1. sort vertices in MCL by SHD

2. for cml∈MCL do
3. divide vertices which have the same SHD into a group
4. end for
5. for group∈groups do
6. encapsulate group to a component
7. end for

引理 2 CreateComponent 算法依据节点的重叠度划分极大团为资源块，时间复杂度为 $O(3^{n/3})$。

证明：文献[24]已经证明，如果图 G 含有 n 个顶点，则 G 最多含有 $3^{n/3}$ 个极大团。CreateComponent 算法对图 G 的极大团集合做了两次遍历操作，因此时间复杂度为 $O(3^{n/3})$。

如图 4 所示，CreateComponent 获得的结果是：$\{A,B\}$，$\{C,D\}$和$\{E\}$ 3 个资源块，其中每一个资源块中节点的重叠度相同，并且依据节点的重叠度划分极大团为资源块。

4.3 AllocateComponent 算法

为了方便资源的分配操作，设置了资源块组件，其封装了组合云的基本信息。资源的分配工作以资源块为单位进行。设计 AllocateComponent 算法完成资源的分配工作。AllocateComponent 算法的核心思想是贪心地选择块内通信延迟最小的资源块分配给用户。

算法 3 AllocateComponent(*COMs*,*REQ*)

Input:COMs(components that acquired from algorithm2),REQ(a request of one user)
Output:Component that allocated to the customer
1. sort components in ascending order of max_delay
2. for component∈components
3. if weight(resources)≥REQ then
4. allocate component
5. update component
6. break
7. end if
8. end for

引理 3 AllocateComponent 算法的时间复杂度为 $O(n\log n)$，其中 n 为资源块的个数。

证明：AllocateComponent 算法首先采用快速排序算法，按通信延迟递增的顺序对资源块进行排序；然后，遍历资源块，查找符合用户请求的资源块，并完成资源的分配。因此，算法的时间复杂度与快速排序算法保持一致。

5 仿真实验

本节通过两组仿真实验，随着用户数目和分配资源数量的增加，统计已分配资源的最大通信延迟。第一组实验的目的是，在用户数目增加的情况下，比较 minStar 算法和本文所提启发式算法的最大通信延迟。第二组实验的目的是，当用户在已获得所申请的资源后再次申请资源时，比较 minStar 算法和本文提出的算法所得最大通信延迟的增加量。

为客观地比较文献[13]和本文提出的算法，算法均使用 C 语言实现，实验环境为 AMD E2-7110 1500MHz CPU 4 核、4G DDR3 内存。资源拓扑图和用户申请的资源数目随机生成。其中，资源拓扑图的规模分为 3 类：10 个节点、25 个节点和 50 个节点。用户的资源申请批量式地到达系统，系统的资源总数为 2000 且平均分布在各个节点上。

图 5、图 6 和图 7 分别展示了在 3 种不同云节点规模下，随着用户数目的增加，所有用户申请到资源的最大通信延迟。在 3 种不同边缘云规模的环境下，实验结果均显示，当用户申请的资源数大约达到系统资源的 75%时，新算法的最大通信延迟最多降为原来的一半。另外，本文提出的启发式算法所得的结果相较于 minStar 算法所得结果，明显更稳定，不会出现通信延迟大幅上升的现象。这是因为，基于极大团的分配策略避免了定理 1 所论述的情况。

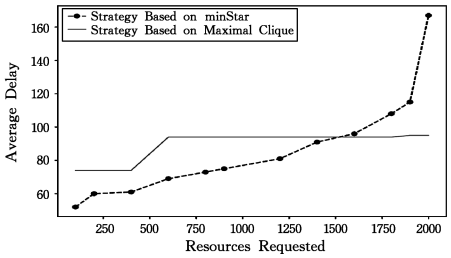

图 5 10 个节点的边缘云环境

Fig. 5 Edge cloud environment of 10 nodes

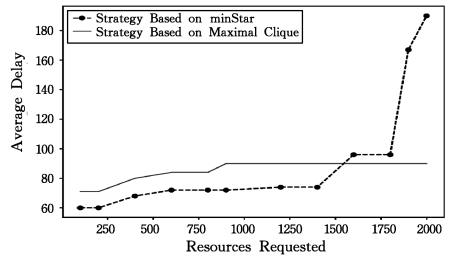

图 6 25 个节点的边缘云环境

Fig. 6 Edge cloud environment of 25 nodes

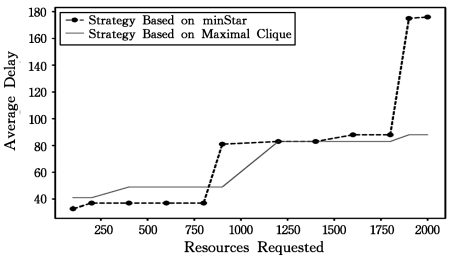

图 7 50 个节点的边缘云环境

Fig. 7 Edge cloud environment of 50 nodes

图 8 展示了本文第二组实验的结果。

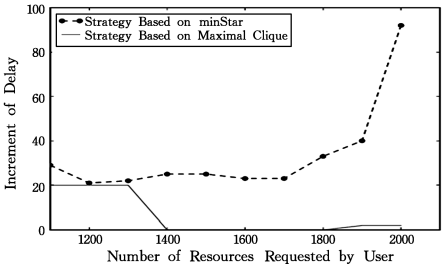

图 8 用户再次申请资源时最大通信延迟的增量

Fig. 8 Increment of maximum communication delay after user getting new resources

该组实验的目的是，在用户已经获取到第一次申请的资源后发起第二次资源请求时，比较两种算法所获得的资源最

大通信延迟的增加量。以 1000 为基准，即此时系统已经为用户分配了 1000 个单位的资源。对两种算法产生的最大通信延迟的增加量进行比较，实验结果显示，minStar 算法的最大通信延迟会有较大增加，尤其在资源消耗超过 1700 之后，最大通信延迟将持续大幅度地增加。而本文所提算法只是在前 30%的阶段中最大通信延迟有所增加，后续资源的分配几乎不会增加通信延迟。

结束语　随着物联网、移动互联网等新一代互联网应用范式的兴起和发展，传统集中式云计算已不能满足日益增强的用户需求，边缘云计算范式应运而生。为了进一步提升边缘云节点的可靠性、安全性及计算能力，云组合的思想被提出。新的边缘云节点聚合算法利用团中节点两两相连的特性，通过合理地避开极大团之间互相重叠的部分，提升了分布式云计算体系的全局性能。新算法将边缘云组合的全局最大通信延迟最大降至原来的 50%。

参 考 文 献

[1] FOSTER I, ZHAO Y, RAICU I, et al. Cloud computing and grid computing 360-degree compared[C]// Proceedings of the Grid Computing Environments Workshop. Piscataway, NJ: IEEE, 2008: 1-10.

[2] STANOEVSKA-SLABEVA K, WOZNIAK T. Grid and Cloud Computing: Cloud Basics-An Introduction to Cloud Computing [M]. Berlin: Springer, 2010: 47-61.

[3] ZHANG Q, CHENG L, BOUTABA R. Cloud Computing: State-of-the-Art and Research Challenges [J]. Journal of Internet Services and Applications, 2010, 1(1): 7-18.

[4] MARSTON S, LI Z, BANDYOPADHYAY S, et al. Cloud Computing—The Business Perspective[C]// Proceedings of the 47th Hawaii International Conference on System Sciences. New York: ACM, 2011: 1-11.

[5] GIURGIU I, RIVA O, JURIC D, et al. Calling the Cloud: Enabling Mobile Phones as Interfaces to Cloud Applications[C]// Proceedings of the 10th ACM/IFIP/USENIX International Conference on Middleware. New York: Springer, 2009: 83-102.

[6] SATYANARAYANAN M, BAHL P, CACERES R, et al. The Case for Vm-Based Cloudlets in Mobile Computing[J]. IEEE Pervasive Computing, 2009, 8(4): 14-23.

[7] VERBELEN T, SIMOENS P, DE TURCK F, et al. Cloudlets: Bringing the Cloud to the Mobile User[C]// Proceedings of the third ACM Workshop on Mobile Cloud Computing and Services. New York: ACM, 2012: 29-36.

[8] PAL R, LIN S H, GOLUBCHIK L. The Cloudlet Bazaar Dynamic Markets for the Small Cloud[J/OL]. [2017-05-25]. https://arxiv.org/pdf/1704.00845.pdf.

[9] LI C L, LIU Y P, LUO Y L. QoS-Based Resource Allocation Across Local and Public Cloud for Resource-Constrained Mobile Device[J]. Transaction on Emerging Telecommunications Technologies, 2016, 28(5): e3126.

[10] FERREIRA L L, ALBANO M, DELSING J. QoS-as-a-Service in the Local Cloud[C]// Proceedings of the 21st International Conference on Emerging Technologies and Factory Automation (ETFA'16). Piscataway, NJ: IEEE, 2016.

[11] OTHMAN M, MADANI S A, KHAN S U. A Survey of Mobile Cloud Computing Application Models[J]. IEEE Communications Surveys & Tutorials, 2014, 16(1): 393-413.

[12] LIU W Q, CAO J N, QIU X J, et al. Improving Performance of Mobile Interactive Data-Streaming Applications With Multiple Cloudlets[C]// Proceedings of IEEE International Conference on Cloud Computing Technology and Science. Piscataway, NJ: IEEE, 2014: 46-53.

[13] ALICHERRY M, LAKSHMAN T V. Network Aware Resource Allocation in Distributed Clouds[C]// Proceedings of the 31st Annual IEEE International Conference on Computer Communications (IEEE INFOCOM 2012). Piscataway, NJ: IEEE, 2012: 963-971.

[14] GAO R F, WU J, LAM S K, et al. Reducing Access Latency in Virtual Machine Assignments[C]// Proceedings of the 2nd International Conference on Data Science and Systems. Piscataway, NJ: IEEE, 2016: 616-622.

[15] JIA M, LIANG W F, XU Z C, et al. Cloudlet Load Balancing in Wireless Metropolitan Area Networks[C]// Proceedings of the 35st Annual IEEE International Conference on Computer Communications (IEEE INFOCOM 2016). Piscataway, NJ: IEEE, 2016: 1-9.

[16] YAO Y, CAO J, LI M L. Network-Aware Virtual Machine Allocation in Cloud Datacenter[C]// Proceedings of the 13th IFIP International Conference on Network and Parallel Computing (NPC'13). Berlin: Springer, 2013: 71-82.

[17] SHIN J Y, WONG B, SIRER E G. Small-world datacenters[C]// Proceedings of the 2nd ACM Symposium on Cloud Computing. New York: ACM, 2011.

[18] CESELLI A, PREMOLI M, SECCI S. Mobile Edge Cloud Network Design Optimization [J]. IEEE/ACM Transactions on Networking, 2017, PP(99): 1-14.

[19] BONOMI F, MILITO R, ZHU J, et al. Fog Computing and Its Role in the Internet of Things[C]// Proceedings of the first Edition of the MCC Workshop on Mobile Cloud Computing (MCC'12). New York: ACM, 2012: 13-16.

[20] CHUN B G, IHM S, MANIATIS P, et al. CloneCloud: Elastic Execution Between Mobile Device and Cloud[C]// Proceedings of the 6th Conference on Computer Systems. New York: ACM, 2011: 301-314.

[21] ZHANG W W, WEN Y G, WU D O. Collaborative Task Execution in Mobile Cloud Computing Under a Stochastic Wireless Channel[J]. IEEE Transactions on Wireless Communications, 2015, 14(1): 81-93.

[22] NIRJON S, LIU J, DEJEAN G, et al. COIN-GPS: Indoor Localization From Direct GPS Receiving[C]// Proceedings of the 12th Annual International Conference on Mobile Systems, Applications and Services. New York: ACM, 2014: 301-314.

[23] CHOW R, JAKOBSSON M, MASUOKA R, et al. Authentication in the Clouds: A Framework and Its Application to Mobile Users[C]// Proceedings of the 2010 ACM Workshop on Cloud Computing Security Workshop (CCSW'10). 2010: 1-6.

[24] TOMITA E, TANAKA A, TAKAHASHI H. The Worst-Case Time Complexity for Generating All Maximal Cliques and Computational Experiments [J]. Theoretical Computer Science, 2006, 363(1): 28-42.