计算机科学
COMPUTER SCIENCE

去中心化云存储网络的存储任务分配算法

申圳, 赵成贵

引用本文

申圳, 赵成贵. 去中心化云存储网络的存储任务分配算法[J]. 计算机科学, 2022, 49(12): 17-21.

SHEN Zhen, ZHAO Cheng-gui. Storage Task Allocation Algorithm in Decentralized Cloud Storage Network [J]. Computer Science, 2022, 49(12): 17-21.

相似文章推荐（请使用火狐或 IE 浏览器查看文章）

Similar articles recommended (Please use Firefox or IE to view the article)

基于联邦学习的暖通空调系统故障检测与诊断

Fault Detection and Diagnosis of HVAC System Based on Federated Learning

计算机科学, 2022, 49(12): 74-80. https://doi.org/10.11896/jsjkx.220700280

基于联邦学习的Gamma回归算法

FL-GRM:Gamma Regression Algorithm Based on Federated Learning

计算机科学, 2022, 49(12): 66-73. https://doi.org/10.11896/jsjkx.220600034

基于联邦学习的车联网多维资源动态分配算法

Multi-dimensional Resource Dynamic Allocation Algorithm for Internet of Vehicles Based on Federated Learning

计算机科学, 2022, 49(12): 59-65. https://doi.org/10.11896/jsjkx.211000123

边缘场景下动态权重的联邦学习优化方法

Federated Learning Optimization Method for Dynamic Weights in Edge Scenarios

计算机科学, 2022, 49(12): 53-58. https://doi.org/10.11896/jsjkx.220700136

联邦学习激励机制研究综述

Survey of Incentive Mechanism for Federated Learning

计算机科学, 2022, 49(12): 46-52. https://doi.org/10.11896/jsjkx.220500272

计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.220700131

去中心化云存储网络的存储任务分配算法

申　圳　赵成贵
云南财经大学信息学院　昆明 650221
(stephen_zhen123@163.com)

摘　要　针对联邦学习客户端数据集的存储任务分配问题构建新型模型,为保证去中心化云存储网络的负载均衡,缩短存储数据上传/恢复时间,减少客户端存储总花费,提出了一种考虑客户端需求和全局负载的数据存储任务分配算法——URGL_allo(Allocation Based on User Requirements and Global Load)算法。在节点分配阶段考虑全局负载、拓扑属性及客户端关注的存储价格和数据恢复时间等节点资源,结合万有引力定律定义新的节点排序方法,选择最佳存储任务分配节点。在链路分配阶段,使用 Dijkstra 算法计算以客户端节点为中心到网络中其他节点的最短路径,并选择两节点间最短路径集合中带宽值最大的路径进行分配。仿真结果表明,相比基于随机策略的分配算法(Random_allo),所提算法的负载均衡指数、客户端存储总花费分别降低了 41.9%,5%,并且与基于链路带宽的贪婪算法的数据恢复时间相差不大,都稳定维持在(0,2]之间,是 Random_allo 算法的 1/20,在全局负载和服务质量上的综合表现优于对比算法。

关键词:联邦学习;去中心化云存储;存储任务分配;全局负载;客户端需求;节点排序
中图法分类号　TP311

Storage Task Allocation Algorithm in Decentralized Cloud Storage Network

SHEN Zhen and ZHAO Cheng-gui
School of Information Science,Yunnan University of Finance and Economics,Kunming 650221,China

Abstract　Constructing a novel model for the storage task allocation problem of federated learning client datasets,to ensure load balancing of decentralized cloud storage networks,shorten the storage data uploading and recovery time,and reduce the total client storage cost,a data storage task allocation algorithm——URGL_allo (allocation based on user requirements and global load) that considers client requirements and global load is proposed. In the node allocation phase,node resources such as global load,topological attributes,storage price and data recovery time concerned by clients are considered,and a new node ranking method is defined in conjunction with the law of gravity to select the best storage task allocation node. In the link allocation stage,the shortest path calculation is performed using Dijkstra's algorithm for the client node as the center to other nodes in the network,and the path with the largest bandwidth value in the set of shortest paths between two nodes is selected for allocation. Simulation results show that the proposed algorithm reduces the load balancing index and the total client storage cost by 41.9% and 5%,respectively,compared with the random policy-based allocation algorithm (Random_allo),and the data recovery time is not much different from that of the link bandwidth-based greedy algorithm,both of which are stably maintained between (0,2], which is 1/20 of Random_allo. The combined performance of global load and service quality is better than that of the comparison algorithm.

Keywords　Federated learning,Decentralized cloud storage,Storage task allocation,Global load,Client requirements,Node sequencing

1　引言

在联邦学习[1]中,客户端数据集的数量大小和内容质量极大地影响了模型训练的准确率[2],有关学者通过多次重复实验证明了增加客户端数据量相比不平衡校正方法[3]对模型性能提升的优越性[4],这说明在联邦学习中增加每个客户端训练的数据量是有意义的[5],但智能手机等边缘设备的存储空间限制了数据量的提升,去中心化云存储网络为解决保证用户隐私前提下联邦学习客户端存储空间的扩展问题提供了新的可能。

去中心化云存储网络[6]是依赖分布式哈希表类似技术[7-8],构建在提供存储空间的对等网络上的存储系统。存储

到稿日期:2022-07-13　返修日期:2022-09-30
基金项目:国家自然科学基金(61562089)
This work was supported by the National Natural Science Foundation of China(61562089).
通信作者:赵成贵(zcgui@126.com)

任务分配[9]指存储网络中的节点产生存储数据的请求后，根据分配策略将文件块分配到不同节点的过程。传统云存储分配策略中，存储设备的负载均衡是必须考虑的一点，而对于参与联邦学习的客户端，低存储总花费、较短的数据上传/恢复时间等是必须考虑的，但现有去中心化云存储网络中对存储任务分配的研究相对较少[10]。Benisi 等[11]最近对去中心化云存储网络做了一次全面的调查，在对未来的挑战中提到了关于订单延迟和下载存储文件速度的问题，优化订单(存储任务)分配算法是缓解这方面问题的方向之一，然而多数云存储框架中要么不考虑存储任务分配[12]，要么仅采用简单的基于链路带宽分配[13]或随机分配策略[14]，严重影响了存储网络的负载均衡、数据上传\恢复时间等存储网络性能，且不能满足用户的存储成本要求。直到 2019 年，Vakilinia 等[9]第一次将去中心化存储网络中的存储任务分配问题形式化，提出基于拍卖的方式进行存储任务分配，并提出了最小化智能合约计算成本算法，提升了部分系统的性能，但其存储任务分配主要考虑存储提供者的利益，对存储用户的利益缺乏考虑。

针对将联邦学习中客户端的数据集在去中心化存储网络中进行存储任务分配的问题，本文基于图理论构建去中心化云存储网络中的存储任务分配模型，并提出了一种考虑客户端需求和全局负载的数据存储任务分配算法，在保证负载均衡的前提下降低了客户端存储总花费和缩短了数据上传\恢复时间，从而使得客户端在参与联邦学习中获得更多收益，并获得更好的模型训练结果。

2 问题描述与模型建立

去中心化云存储系统底层存储节点分布与传统云存储不同，其底层基于 P2P 网络，因此将一定数量数据碎片分发到 P2P 网络中各个节点的过程，可以被模拟为在满足资源约束前提下的存储资源分配问题，与网络虚拟化中的资源分配问题又有所不同，其数据存储任务模拟成的虚拟网络全部为星型结构，且不着重考虑在映射时链路的约束等。因此，本文利用图理论根据问题特点构建问题模型，并提出了一种基于客户端要求和全局负载的分配策略来解决数据存储任务分配问题。

2.1 存储节点网络

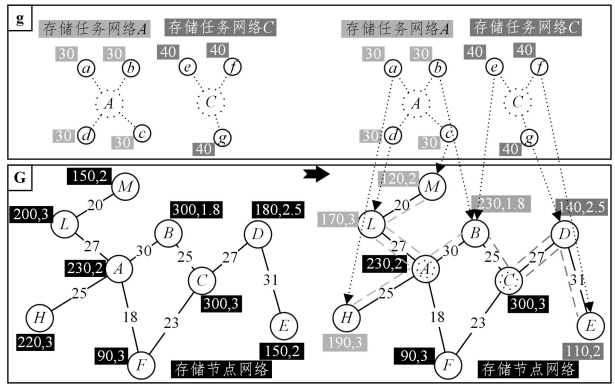
将底层存储节点网络的拓扑结构定义为加权无向图 $G_s=(N_s,L_s,An_s,Al_s)$，图 1 中 G 显示了图 G_s 的一个实例，其中 N_s 表示存储节点的集合；L_s 代表存储节点间链路的集合；存储节点和链路的属性分别用 An_s 和 Al_s 表示，每个存储节点 $n_s\in N_s$，其节点编号为 $No(n_s)$，提供的存储空间定义为 $s(n_s)$，存储空间单位价格为 $p(n_s)$，对每条存储节点间的链路 $l_s\in L_s$，其可提供的带宽定义为 $b(l_s)$。

2.2 数据存储任务网络

去中心化云存储系统进行数据存储任务分配时，为保证数据的安全性，采用纠删码数据冗余技术[14]，将待存储数据分割为 k 块并编码为大小相等的 m 块($m>k$)，将这 m 块待存储数据虚拟为一个星型网络，将其定义为加权无向图 $G_f=(No_f,N_f,L_f,Rn_f,Rl_f)$，图 1 中 g 显示了图 G_f 的一个实例，其中 No_f 表示提出存储任务的节点编号，即图 1 存储任务网络 A 的中心节点，N_f 表示所有任务节点集合；L_f 代表所有任务节点间的路径集合；节点和路径的属性分别用 Rn_f 和 Rl_f 表示，对于每个存储任务 $n_f\in N_f$，其需求的存储空间定义为 $s(n_f)$。图 1(左)给出了 1 个存储网络以及 2 个碎片网络示例，其中，链路附近的数字表示链路的带宽，节点附近括号中，前边的数字表示剩余的存储资源，后边的数字表示单位存储资源的价格。

2.3 数据存储任务分配问题描述

当 P2P 网络中的一个或多个客户端想要存储文件时，会以该客户端为中心，产生星型网络，该客户端节点的系统会根据客户端的存储需求为客户端进行存储任务分配，这个过程称为数据存储任务分配。可将该过程定义为 $G_f\rightarrow G_s$，该过程包括任务节点分配和链路分配两部分。分配过程中的唯一约束为存储节点提供的空间需要大于或等于存储任务节点的需求空间。如图 1 所示，当客户端 A 和客户端 C 要存储数据时，系统帮助客户端分别生成数据存储任务网络 A 和 C，并分别以 A 和 C 为中心进行分配，图 1(右)给出了一种可行的分配结果，其中数据存储任务的分配分别为 $\{a\rightarrow H,b\rightarrow B,c\rightarrow M,d\rightarrow L\}$，$\{e\rightarrow B,f\rightarrow E,g\rightarrow D\}$，相应存储任务节点间的链路分配结果分别为 $\{(A,a)\rightarrow(A,H),(A,b)\rightarrow(A,B),(A,c)\rightarrow(A,L,M)\}$；$(A,d)\rightarrow(A,L)\}$，$\{(C,e)\rightarrow(C,B)\rightarrow(C,g)\rightarrow(C,D),(C,f)\rightarrow(C,D,E)\}$。



注：图 1(左上)给出了分配前的存储任务网络，图 1(右)给出了存储任务分配后的可能结果

图 1 存储节点网络和数据存储任务网络示意图

Fig. 1 Schematic diagram of storage node network and data storage task network

2.4 评价指标

2.4.1 负载均衡指数

一个存储资源分布均衡的存储节点网络可以提升后续接受数据存储任务的能力，并为客户端个性化存储策略提供更好的支持。因此，在分配过程中需考虑存储节点网络中各节点的负载率，故定义整个存储网络的负载均衡指数为：

$$S_{\text{balance}}=\sqrt{\sum_{i\in M}\left(s(n_i)\%-\frac{\sum_{i\in M}s(n_i)\%}{|M|}\right)^2} \tag{1}$$

其中，$|M|$ 表示存储节点网络中的节点，S_{balance} 越小，存储节点网络的资源平衡越好；$s(n_i)\%$ 表示网络中各节点的负载率，其中 $s(n_i)\%=1-s(n_i)/s_{\max}(n_i)$，$s_{\max}(n_i)$ 表示存储节点可提供的最大空间，$s(n_i)$ 表示存储节点的剩余空间。

2.4.2　数据上传/恢复时间

在采用纠删码技术的数据分配方案中，平均文件丢失数量$(m-k)$时，可保证文件分配的安全性[15]，即仅需 k 个数据碎片便可恢复完整数据，数据的恢复时间为其中一组 k 个数据碎片被分配到各存储节点时传输时延的最大值，则该数据上传/恢复时间可以表示为：

$$t(No_f)=\max\left(\frac{D_f}{b_{ij}}\cdot q_{b_{ij}}\right) \tag{2}$$

其中，No_f 表示请求数据存储的客户端节点编号，D_f 表示该客户端生成的每个存储任务的数据大小，b_{ij} 表示存储节点 n_i 与 n_j 之间链路的带宽大小，$q_{b_{ij}}=0$ 表示数据不经过存储节点 n_i 和 n_j 之间的链路，$q_{b_{ij}}=1$ 表示数据碎片经过存储节点 n_i 和 n_j 之间的链路。

因此，最短数据上传/恢复时间为最优的一组 k 个数据碎片分配组合，使得目标函数最小：

$$T(No_f)=\min\left\{\max\left(\frac{D_f}{b_{ij}}\cdot q_{b_{ij}}\right)\right\} \tag{3}$$

2.4.3　客户端存储总花费

存储空间由存储网络中的各节点提供，各节点的存储价格由节点本身的存储环境确定，因此客户端存储总花费由存储任务分配到的存储节点组合与存储任务的数据大小决定，故客户端存储总花费可以定义为：

$$C_{\text{total}}=\sum_{i=1}^{m}D_f\times p_i \tag{4}$$

3　基于客户端需求和全局资源的存储任务分配算法

现有的基于随机的存储任务分配算法无法考虑全局网络的存储资源合理利用，并且不能满足客户端的个性化需求。在网络资源分配算法中，基于节点重要度的映射算法被广为研究[16]，受该算法以及万有引力定律的启发，本文针对星型网络，考虑客户端需求和全局资源，将存储任务分配分成节点分配和链路分配两阶段进行设计。

3.1　节点分配

3.1.1　拓扑属性

在节点网络中，拓扑属性可以从不同方面去度量节点的重要性以及相对影响，因此在考虑节点重要度排序的算法中，引入拓扑属性是必要的。

(1)度(Degree)：节点直连的链路条数。度越大，说明节点在网络中的连接程度越高。

$$D(n_i)=\sum_{n_j\in G,i\neq j}link(n_i,n_j) \tag{5}$$

其中，G 为 n_i 相邻的节点集合。

(2)节点链路强度(Node Link Strength)：节点直连的所有链路可提供的带宽和。链路强度越大，说明数据通过该节点时的传输速度越快。

$$NLS(n_i)=\sum_{n_j\in G,i\neq j}bandwidth(n_i,n_j) \tag{6}$$

其中，G 为 n_i 相邻的节点集合。

(3)节点与节点最短跳数(Node to Node Shortest Hops)：节点与节点之间最短路径的跳数。跳数越大，说明两节点相距越远。

$$NtN\_SH(n_i,n_j)=numOfNode(Dijkstra(n_i,n_j)),i\neq j \tag{7}$$

(4)节点与节点最小带宽(Node to Node Min Bandwidth)：节点与节点之间最短路径上的最小带宽值。带宽值越大，说明两节点之间数据传输的速度越快。

$$NtN\_MinBW(n_i,n_j)=minBandwidth(Dijkstra(n_i,n_j)),i\neq j \tag{8}$$

3.1.2　节点排序

在定义节点相对重要度时，我们考虑的是将根据客户端需求生成的以客户端为中心的星型网络合理地分布在存储网络中，以一个抽象半径的距离分布在客户端四周，以获得良好的数据恢复时间，则节点引力值便可用来衡量存储节点网络中其他节点 n_i 相对于客户端中心节点的重要性。受万有引力定律启发[17]，结合节点的资源丰富度以及拓扑属性，将相对客户端节点引力值定义为：

$$Inter\_nodal+F(No_fn_i)=g\frac{RV(No_f)RV(n_i)\cdot NtN\_SH(No_fn_i)}{t(No_fn_i)^2} \tag{9}$$

其中，g 是一个相对引力常量；$RV(No_f)$表示客户端中心 No_f 的资源量，由式(10)定义，用衡量客户端中心 No_f 的资源；$\sum s(G_f)$表示请求 G_f 的存储空间总需求；$RV(n_i)$表示存储节点 n_i 的资源量，由式(11)定义，用衡量存储网络中节点 n_i 的资源；$t(No_fn_i)$表示从节点 n_i 到客户端中心 No_f 的数据恢复时间，由式(12)定义。

$$RV\_C(No_f)=D(No_f)\cdot\frac{NLS(No_f)}{\sum s(G_f)} \tag{10}$$

$$RV\_E(n_i)=S(n_i)D(n_i)\cdot NLS(n_i) \tag{11}$$

$$t(No_fn_i)=\frac{D_f}{NtN\_MinBW(No_fn_i)} \tag{12}$$

每个节点提供的存储空间是有限的，为了使客户端的数据合理地分配在存储网络中，并且满足客户端的价格和数据恢复时间要求，将节点排序值定义为：

$$Node\_RM(No_fn_i)=\lambda\cdot\frac{Inter\_node\_F(No_fn_i)}{\sum Inter\_node\_F(No_fn_i)}+(1-\lambda-\gamma)\frac{Loadbalance(n_i)}{\sum Loadbalance(n_i)}+\gamma\cdot\frac{P(n_i)}{\sum P(n_i)} \tag{13}$$

其中，λ 为节点引力值权重，γ 为节点存储价格权重，$(1-\lambda-\gamma)$为节点负载权重，且 $0<\lambda,\gamma,(1-\lambda-\gamma)<1$。

3.2　链路分配

在进行链路分配时，为获得最短数据上传/恢复时间，结合 Dijkstra 算法，找到存储节点到客户端中心节点的多条最短路径集合 $Path(No_fn_i)$，再对 $Path(No_fn_i)$中的每条路径进行遍历，找到路径最小带宽值最大的路径进行分配，从而提升客户端上传/下载数据时的体验感。

3.3　URGL_allo 算法

URGL_allo 算法的具体过程如算法 1 所示。

算法 1　URGL_allo 算法

Input：$G_s=(Ns,Ls,An_s,Al_s)$，$G_f=(No_f,N_f,L_f,Rn_f,Rl_f)$

Output：result(F_N，F_L，$S_{balance}$，$T(No_f)$，C_{total})

1. 定义空数组 F_N，F_L 存储任务分配的对应节点和链路；
2. 定义 Count=0，记录已分配任务数量；
3. for 每个存储节点 $n_s\in N_s$ do
4. 　根据式(9)－式(13)计算每个存储节点 n_s 的负载率 Loadbalance(n_s)，以及每个存储节点 n_s 相对于客户端节点 No_f 的

Node_RM($No_f n_s$)和 Inter_nodal_F($No_f n_s$)；
5. end for
6. 将存储节点 $n_s \in N_s$ 按照 Node_RM($No_f n_s$)值逆序排列；
7. for 待分配的数据存储任务碎片 $n_f \in N_f$ do
8. for 排序后的存储节点 $n_s \in N_s$ do
9. if n_s 无任务分配 and $s(n_s) \geqslant s(n_f)$ do
10. 将该任务碎片 n_f 分配给存储节点 n_s：$F_N[n_f]=n_s$；
11. 已分配的任务数 Count 加 1；
12. 计算目前已分配的存储总花费 C_{total}；
13. 跳出当前循环；
14. end if
15. end for
16. end for
17. 根据式(1)实时计算负载均衡指数 $S_{balance}$；
18. If Count 等于待分配任务数 num(N_f) do 进行链路分配：
19. 计算客户端节点 No_f 到 F_N 中的节点 n_s 在图 G_s 中的最短路径集合 Path($No_f n_s$)；
20. for 待分配链路 $l_f \in L_f$ do
21. for 每条最短路径 $l_s \in$ Path($No_f n_s$) do
22. 从 Path($No_f n_s$)中选择节点 No_f 到 n_s 的最短路径中最小带宽值最大的路径 l_s，将其分配给 l_f：$F_L[l_f]=l_s$；
23. 计算数据上传/恢复时间为分配的所有节点到客户端节点传输的最大时间：

$$T(No_f)=\max[T(No_f), s(n_f)/b(l_s)]$$

24. end for
25. end for
26. else do
27. 标记该请求 G_f 被阻塞；
28. end if
29. return result(F_N, F_L, $S_{balance}$, $T(No_f)$, C_{total})

算法 1 包括节点分配和链路分配两部分，当一个节点产生存储任务时，首先进行节点分配，因计算节点排序值时需考虑节点间最短路径及最小带宽值，所以集中了大量计算，其时间复杂度为 $O(|N_s|(|N_s|+|L_s|)\log|L_s|)$，节点选择时间复杂度为 $O(|N_f||N_s|)$，最短路径集合已在节点分配时求出，链路分配的时间复杂度为 $O(|N_f||Path(No_f n_s)|)$。因此 URGL_allo 算法分配过程的时间复杂度为 $O(|N_s|(|N_s|+|L_s|)\log|L_s|+|N_f||N_s|+|N_f||Path(No_f n_s)|$。

4 仿真结果与分析

本节首先介绍了仿真实验环境，然后对仿真结果进行了分析。为了验证本文算法的优势，本文将基于链路带宽速度的贪婪算法(BWBased_allo)[13]和随机分配算法(Random_allo)[14]作为基线算法，将根据客户端需求提出的基于价格的贪婪算法(PriBased_allo)、基于全局存储资源的逐利算法(GRC_allo)[18]以及本文提出的基于客户端需求和全局资源的任务分配算法(URGL_allo)与之进行对比，来体现 URGL_allo 算法在解决客户端数据存储任务分配问题时的优越性。

4.1 实验环境搭建

本文使用 python 中的 NetworkX 框架来构建实验环境，其中生成一个节点数为 100 的存储网络，节点存储空间服从正态分布 $N(1636,300)$，节点单位存储价格服从正态分布 $N(2.5,0.15)$，节点间的连接率为 0.4，链路带宽大小服从正态分布 $N(2.5,0.44)$，运行总时间为 1 400 时间单位；生成 1 400 个星型存储任务网络，其节点数在区间[10,30]随机选择，数据存储任务总需求空间服从正态分布 N(22.5,5.5)，在区间[0,99]间随机生成节点编号作为客户端中心，平均每时间单位生成 1 个存储任务网络。其他参数设置为 $\lambda=0.1$，$\gamma=0.7$，$g=1\times10^{-6}$。

4.2 仿真结果分析

我们分别从存储网络负载均衡指数、数据上传/恢复时间、客户端存储平均花费这 3 方面对 5 种算法进行了对比。图 2 给出了网络负载均衡指数，可以看出 5 种算法的走势是类似的，相比 BWBased_allo 算法、Random_allo 算法、PriBased_allo 算法以及 GRC_allo 算法，URGL_allo 算法的网络负载均衡指数更低，虽然 BWBased_allo 算法的负载均衡指数前半段略低于本文算法，但是随着存储请求的增加，该基线算法的负载均衡指数超过了 URGL_allo 算法，并且两者的差值有逐渐增大的趋势。其中，本文提出的 URGL_allo 算法的负载均衡指数较 BWBased_allo 算法、Random_allo 算法、PriBased_allo 算法、GRC_allo 算法分别低 3.5%，41.9%，38.1%，19.6%，通过分析，原因是 URGL_allo 算法在进行节点分配时对存储网络中各节点的负载率进行计算，优先选择负载率低的节点进行分配，使得存储资源利用更加均衡，为后续的资源分配提供了更多的选择。

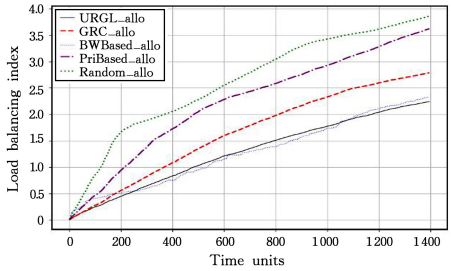


图 2 存储网络负载均衡指数

Fig. 2 Storage network load balancing index

图 3 给出了数据上传/恢复时间，可见 URGL_allo 算法和 BWBased_allo 算法全程在(0,2]范围内小幅度波动，PriBased_allo 算法和 GRC_allo 算法相比前两种算法数据恢复时间略长，在(1,6)范围内波动，而 Random_allo 算法则随着请求数的增加，波动范围逐渐扩大，最大波动到 40，URGL_allo 算法全程表现优异，与 BWBased_allo 算法相差不大，其数据恢复时间是 Random_allo 算法的 1/20。通过分析，其原因为 URGL_allo 算法在定义节点引力值时借鉴了万有引力定律，以数据传输时间为抽象半径，使得资源碎片以合适的时间半径分布在中心节点周围。

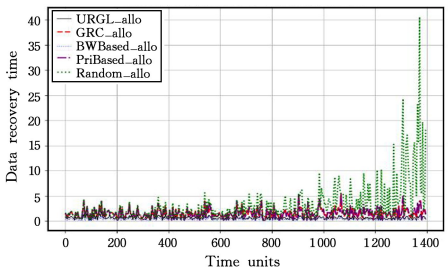


图 3 数据恢复时间

Fig. 3 Data recovery time

图 4 给出了客户端平均存储的总花费，5 种算法的客户端平均存储总花费曲线走势相同，其中 PriBased_allo 算法是基于存储价格的贪婪算法，在客户端存储总花费方面表现优异，但舍弃了数据上传/恢复时间和网络负载均衡，URGL_allo 算法的客户端存储成本虽略高于 PriBased_allo 算法，但相比 BWBased_allo 算法、Random_allo 算法和 GRC_allo 算法，其仍表现优异，分别降低了 6.4%，5.0%和 4.9%，且有良好的网络负载均衡和数据上传/恢复时间，综合指标优异，这是因为 URGL_allo 算法定义的节点排序值综合考虑了节点存储价格、负载率以及传输时间，使得在分配时综合表现优良。

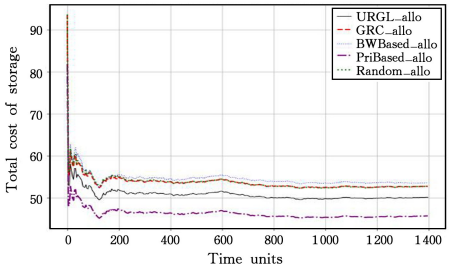


图 4　客户端存储平均花费

Fig. 4　Average client-side storage spend

结束语　本文针对联邦学习客户端数据集的存储任务分配问题构建新型模型，为使存储任务合理地被分配在去中心化云存储网络中，受万有引力定律启发，并结合客户端需求和全局负载，提出了一种新颖的节点排序算法以及链路选择算法。仿真结果表明，相比 BWBased_allo 算法、Random_allo 算法、PriBased_allo 算法以及 GRC_allo 算法，本文提出的 URGL_allo 算法在提出的 3 个指标上综合表现优异。但统一的权重参数设定无法满足客户端个性化需求，并且现存的去中心化存储网络框架形式各异，对算法的部署有不同的阻碍，因此，如何结合联邦学习的特点，搭建合适的去中心化云存储平台，并实现权重智能化选择的存储任务分配算法是下一步工作将要研究的问题。

参 考 文 献

[1] ZHANG C, XIE Y, BAI H, et al. A survey on federated learning [J]. Knowledge-Based Systems, 2021, 216: 106775.

[2] XIAO T, CUI T, ISLAM S R, et al. Joint content placement and storage allocation based on federated learning in F-RANs [J]. Sensors, 2020, 21: 215.

[3] LEMAÎTRE G, NOGUEIRA F, ARIDAS C K. Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning [J]. The Journal of Machine Learning Research, 2017, 18: 559-563.

[4] JUBA B, LE H S. Precision-recall versus accuracy and the role of large data sets[C]// Proceedings of the AAAI Conference on Artificial Intelligence. 2019: 4039-4048.

[5] MCMAHAN B, MOORE E, RAMAGE D, et al. Communicationefficient learning of deep networks from decentralized data [C]// Artificial Intelligence and Statistics. PMLR, 2017: 1273-1282.

[6] WEBVIEW-LAB. Decentralized storage, the disruptor of traditional cloud storage [EB/OL]. (2022-04-15) [2022-08-26]. http://www.yitb.com/article-19160.

[7] CANGIR O F, CANKUR O, OZSOY A. A taxonomy for Blockchain based distributed storage technologies [J]. Information Processing & Management, 2021, 58: 102627.

[8] MAIDSAFE. NET. Safenetwork[EB/OL]. (2014-03-26)[2022-08-26]. https://github.com/maidsafe/Whitepapers/blob/master/Project-Safe.md.

[9] VAKILINIA I, VAKILINIA S, BADSHA S, et al. Pooling approach for task allocation in the blockchain based decentralized storage network[C]// 2019 15th International Conference on Network and Service Management (CNSM). IEEE, 2019: 1-6.

[10] YUAN N H. Research on user-oriented software defined cloud storage resource allocation strategy[D]. Beijing: Beijing University of Posts and Telecommunications, 2017.

[11] BENISI N Z, AMINIAN M, JAVADI B Blockchain-based decentralized storage networks: A survey [J]. Journal of Network and Computer Applications, 2020, 162: 102656.

[12] PROTOCOL-LABS. Filecoin: A dicentralized storage networks [EB/OL]. [2022-07-06]. https://ipfs.netlify.app/tutorial/whitepapera.html.

[13] FU C H. The frame of decentralized storage system based on distributed ledger [D]. Chengdu: University of Electronic Science and Technology of China, 2020.

[14] STORJ LABS I. Storj: A Decentralized Cloud Storage Network Framework (Version 3.0) [EB/OL]. (2018-10-30) [2022-07-06]. https://www.storj.io/whitepaper.

[15] CONG L, LI R, WANG H, et al. Distributed data storage allocation scheme for the industrial internet of things [J]. Electric Power Information and Communication Technology, 2020, 18(7): 52-57.

[16] WU S, YIN H, CAO H, et al. Node ranking strategy in virtual network embedding: An overview [J]. China Communications, 2021, 18: 114-136.

[17] CAO H T. Research on virtual network embedding algorithm in network virtualization environment [D]. Nanjing: Nanjing University of Posts and Telecommunications, 2020.

[18] GONG L, WEN Y, ZHU Z, et al. Toward profit-seeking virtual network embedding algorithm via global resource capacity[C]// IEEE INFOCOM 2014 — IEEE Conference on Computer Communications. IEEE, 2014: 1-9.

SHEN Zhen, born in 1993, postgraduate, is a member of China Computer Federation. His main research interests include blockchain, SDN/NFV, federated learning and so on.

ZHAO Cheng-gui, born in 1974, Ph.D, professor. His main research interests include communication network and computer system architecture, distributed systems, algebraic graph theory and its applications, blockchain, federated learning and so on.

（责任编辑：喻藜）