DOI:10. 13471/j. cnki. acta. snus. 2020. 04. 21. 2020A014

一种面向雾计算的任务调度算法研究*

刘林东，邬依林

广东第二师范学院计算机学院,广东 广州 510303

摘　要：在雾计算环境中，为提高雾计算效率和给用户请求的任务分配合适的资源，需要对任务调度问题进行研究。面向流水线独立任务进行调度研究，首先基于经典Apriori算法提出一种雾计算环境中任务的分类I-Apriori算法；将得到的分类规则以及加权最早完成时间作为任务优先级选择的依据；对于同时达到的任务，优先选择出现在调度关系中的任务进行调度，其他雾结点则依据最早完成时间和加权链接数的高低进行调度。通过仿真实验对ITPS（improved task priority scheduling）算法的性能进行了评估，结果表明，ITPS算法在makespan及AWT（average waiting time）方面具有较好的性能。

关键词：雾计算；任务调度；关联规则；雾计算结点

中图分类号：TP391　　**文献标志码：**A　　**文章编号：**0529 -6579（2021）05 -0166 -09

Research of a task scheduling algorithm in fog computing

LIU Lindong, *WU Yilin*

School of Computer Science, *Guangdong University of Education*, *Guangzhou 510303*, *China*

Abstract: In the heterogeneous and distributed computing environment of fog computing, in order to improve the efficiency of fog computing and allocate appropriate resources to corresponding tasks, task scheduling problem needs to be studied. The scheduling of pipeline independent tasks is studied. Firstly, based on the traditional Apriori algorithm, a task classification algorithm I-Apriori algorithm in fog computing environment is proposed. Association rules generated by I-Apriori algorithm are combined with the weighted earliest completion time of tasks in the task set. Tasks appear in the association rules are selected to schedule first, other fog nodes are scheduled according to the earliest completion time and the number of weighted links. The performance of ITPS algorithm is evaluated by simulation experiments. The results show that ITPS algorithm has a good performance in makespan and AWT.

Key words: fog computing; task scheduling; association rule; fog computing node

计算密集型应用和资源受限型设备之间的矛盾[1-3]已成为雾计算中提供满意体验的瓶颈，需要通过任务调度来解决此问题。雾计算[4]相对于传统的分布式计算和云计算[5]，由数量众多的位置分布的雾结点和终端设备构成，一般由计算性能有限的设备组成，包括路由器、交换机、网关和终端设备等[1, 6-8]。由于带宽以及能耗因素影响，性能异构的物联网[9-12]设备（IOT）得到的数据可以边缘设备上进行计算、分析和调度，从而可以减轻云端的负担。

* **收稿日期：**2020 - 04 - 21　　**录用日期：**2020 - 05 - 29　　**网络首发日期：**2020 - 11 - 05

基金项目：广东省普通高校特色创新项目（自然科学类）（2018KTSCX163，2020KTSCX090）；广东省科技计划项目（2016A010106007，2016B090927010）；广东第二师范学院网络工程重点学科项目（ZD2017004）；广东第二师范学院计算机实践教学示范中心项目（2018sfzx01）

作者简介：刘林东（1978年生），男；**研究方向：**分布式计算、高性能计算；E-mail：hongox@163. com

通信作者：邬依林（1970年生），男；**研究方向：**大数据分析与处理；E-mail：lyw@gdei. edu. cn

现有的雾计算任务调度研究[13]，一般根据资源利用率、时间、能耗、成本、负载均衡等优化目标进行资源分配或任务分配。文献［14］提出了一种面向雾计算的多用户小区聚类资源优化策略，该策略以最大交付时延作为衡量服务质量的重要指标；文献［15］通过定义计算延迟以及通信延迟模型的方法，为雾计算结点的任务调度提供依据；文献［16］通过边缘计算和云计算之间的协作，达到功耗以及传输延迟间的负载平衡；文献［17］提出了一种启发式任务调度算法，达到调度跨度以及调度成本间的平衡。雾计算任务调度是一个NP完全问题[1, 18]。

文中对独立任务流水线任务调度问题进行研究，通过改进的分类挖掘算法建立雾计算任务调度模型以及相应的调度算法，以任务调度跨度和平均等待时间作为主要优化目标，从而提升雾计算用户服务质量以及任务调度吞吐量。

1　雾计算模型

雾计算由前端层、雾层以及云层三层架构构成[19]，如图1所示。前端层作为用户接口，一般由各种类型的物联网设备组成，向雾层发送任务请求；雾层包括m个异构的雾结点，向雾层发送任务请求，雾层离前端层很近，一方面用户可以直接利用雾层中的计算资源，另一方面，也可以减轻云层的计算压力；部署了多台服务器或者云结点的云层，相比较雾层计算能力更强，由于物理上远离前端层，用户请求一般不直接发送至云层。

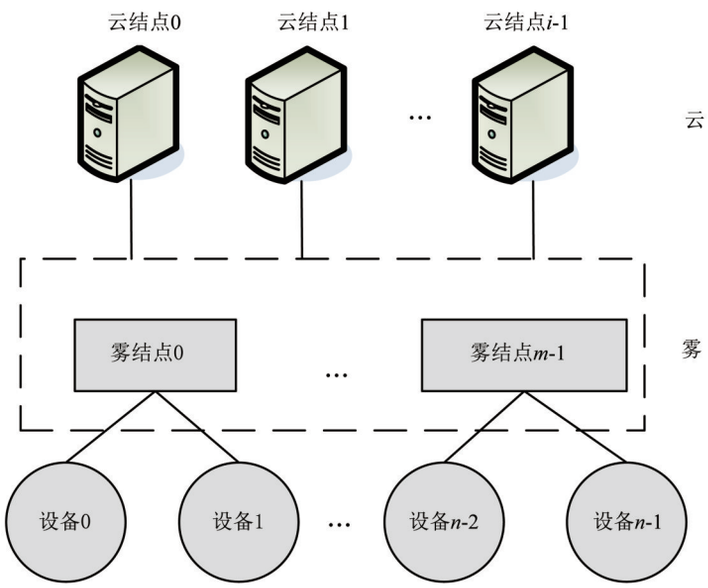

图1　雾计算系统架构

Fig. 1　System architecture of fog computing

2　I-Apriori分类挖掘算法

2.1　算法描述

经典Apriori算法[20-21]每次迭代产生频繁项集都需要去扫描事务数据库D，由于扫描数据库次数过多，造成生成频繁项集的算法效率较低。为改进经典Apriori算法效率低下的问题，提出一种分类挖掘I-Apriori算法。I-Apriori算法包括两个阶段。第一步：产生频繁1项集L_1，获取每个项集的事务标识TID，产生候选集1项集C_1，利用最小支持度阈值得到L_1；第二步：生成频繁项集L. 当L_{k-1}不为空时循环进行后面几项操作，1）候选k项集C_k直接通过L_{k-1}的自连接运算得到；2）利用候选k项集C_k直接得到项集计数；3）基于最小支持度阈值删除候选k项集C_k中不满足条件项集，得到频繁k项集L_k. 循环结束后最终得到频繁项集L. 算法描述如算法1所示。

算法 1：I-Apriori 算法

```
产生候选 1 项集 C1;
统计事务 D 中 TID 的数量 count;
循环对 C1 中每个项集 s{
  s. item-set=s;
  C1 中 s 的数量 s. count;
产生所有包含项集 s 的 TID 列表
如果 s. count<min_sup*count
  删除 C1 中的 s;
}
k=1; Lk=Ck, ;
while (Lk-1≠Ø){
  循环对 Lk-1 中的每个项集 l1{
    循环对 Lk-1 中的每个项集 l2{
      对 l1 和 l2 进行连接;
      直接统计 c. tid-list;
      直接统计 c. count;
    }
}
如果 c. count 大于最小支持度计数{
  把 c 添加 Ck 中;
  Lk=Ck;}
}
```

2.2 算法分析

利用 Java 分别实现经典 Apriori 算法和 I-Apriori 算法，基于 Intel Core（TM）i5-8265U 1.6 GHz CPU、8 GB 内存的硬件环境以及 Win10 系统生成频繁项集。

在最小置信度为 0.6 的实验条件下，通过对比 50~500 个事务数下产生频繁项集的执行时间。图 2（a）为最小支持度计数为 0.40 的情况下两种算法时间开销对比情况；图 2（b）为最小支持度计数为 0.45 的情况下两种算法的时间开销对比情况。通过实验得出，I-Apriori 算法在两种不同最小支持度下的时间开销均小于经典 Apriori 算法的时间开销，最小支持度为 0.40 的情况性能提升了 8.1%~34.4%，在最小支持度为 0.45 的情况性能提升了 7.6%~37.0%；总体上 I-Apriori 具有更好的性能。

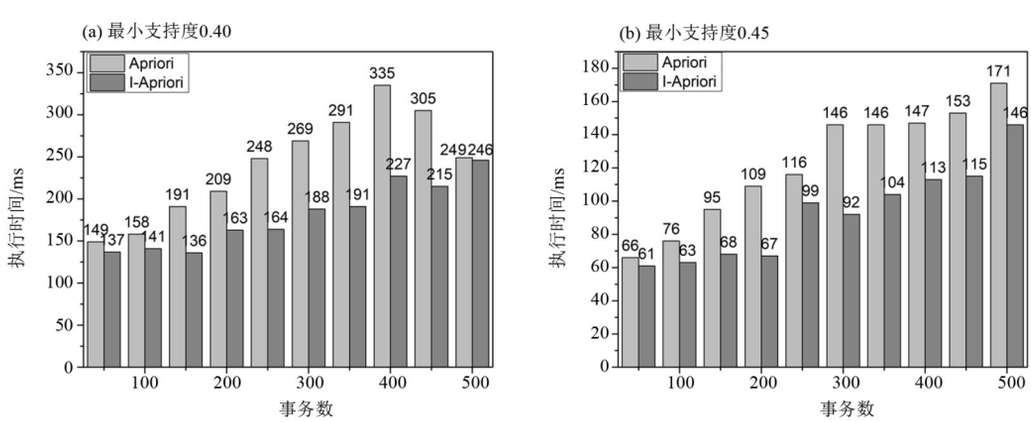

图 2　两种算法在不同最小支持度下执行时间对比

Fig. 2　Comparison of execution time of two algorithms under different minimum support

设 Apriori 算法中事务数为 n，项目数为 m，频繁项集迭代数为 k，则算法时间复杂度为 $O(k^4 \cdot m \cdot n)$；I-Apriori 算法的时间复杂度为 $O(m+n+k^3)$，I-Apriori 时间复杂度性能更优。

3　雾计算中任务调度

3.1　任务调度模型

针对雾计算环境中独立任务流水线任务调度问题，将 I-Apriori 算法融入到任务调度过程中。雾计算任务调度模型如图 3 所示，任务调度模型包括雾结点集 F、任务集 T、事务集 D、调度关系 R 以及 I-Apriori 和 ITPS 两种算法，在 F 和 T 间实现资源分配和任务调度。任务调度的过程是一个迭代更新的过程，每次迭代的基本流程首先利用 I-Apriori 算法对历史调度事务集 D 进行分类挖掘，得到 F 与 T 间的调度规则，然后利用 ITPS 算法读取调度规则，得到 F 与 T 间的调用关系 R，并循环将 R 更新到 D 中。

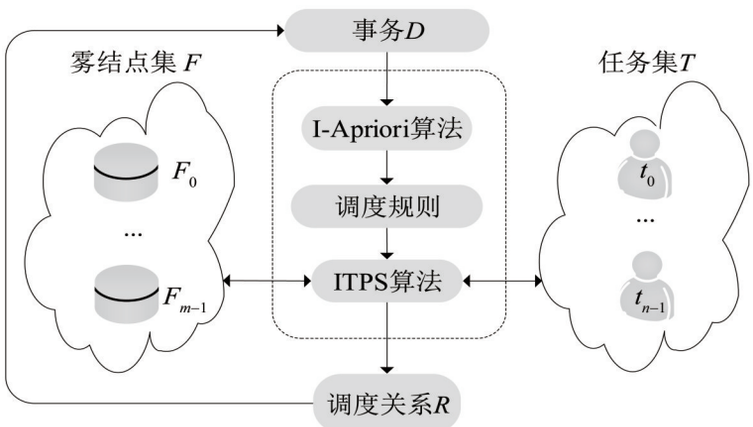

图 3　雾计算中任务调度模型

Fig. 3　Task scheduling model of fog computing

3.2　ITPS 调度算法

ITPS 任务调度算法（详见算法 2）主要包括 3 个步骤：

1）任务选择。循环选择任务集中的任务，首先选择到达时间 AT_i 最早的任务，若两个以上任务同时到达，则优先选择调度关系 R 中的任务；若任务到达时间均不相同，算出全部任务在雾计算资源上的最早完成时间，依据最早完成时间和加权链接数的从小到大进行调度；

2）雾计算结点选择。选择最早完成时间最小的雾计算结点进行任务调度；

3）任务调度。在雾结点上完成任务的执行，得出任务执行时间等调度数据，直到所有任务被调度完成为止。

3.3　调度过程

通过一个完整的任务调度案例分析 ITPS 算法的调度过程，任务调度过程包括任务初始化、初始事务集、I-Apriori 算法、任务调用关系以及 ITPS 算法调度等几个步骤。

3.3.1　任务初始化

设任务集 T 中包含 10（n=10）个独立流水线任务，每个任务 t_i（$0 \leqslant i \leqslant 9$）的到达时间如表 1 所示，其中 t_0，t_2 和 t_5，t_3 和 t_4 的到达时间相同；雾计算结点集 F 中包括 4（m=4）个性能异构的计算结点 F_j（$0 \leqslant j \leqslant 3$），任务 t_i 在雾计算结点 F_j 上的执行开销如表 2 所示，执行开销满足一致性模型条件，即任务 t_i 如果在雾计算结点 F_j 上的执行开销比雾计算结点 F_k 上的执行开销短，则任何其他任务也是如此。

3.3.2　初始事务集

初始事务集是任务集 T 中的任务 t_i 与雾计算结点集 F 的结点 F_j 之间的调度关系的集合，事务集中每一条记录为每一次任务调度的信息，“1”表示任务 t_i 以及雾计算结点 F_j 出现在某条事务 T_z 中，“0”表示事务 T_z 不包含任务 t_i 以及雾计算结点 F_j，设事务集中包含 10（z=10）条事务，初始事务集 D 如表 3 所示。

3.3.3　I-Apriori 算法

1）产生频繁项集。基于初始事务集 D 执行 I-Apriori 算法，以最小支持度 0.5 为例，得到 $\{F_1, F_0, F_3, t_2\}$

算法 2：ITPS 算法

输入：Time[n,m]，$R[t,c]$，任务集 TS
输出：任务调度跨度 makespan，任务平均等待时间 AWT
1 针对任务集 {　//循环对每个任务集进行调度
2　针对每个任务{
3　　针对每个雾计算结点{
4　　　计算 EFT[ ];
5　　}
6　　查找具有最小最早完成时间的任务 t_j;
7　　计算任务开始时间 ST[ ];
8　　按 EFT[ ]值对任务进行排序;
9　}
10　当任务集非空{
11　　针对雾计算结点
12　　　计算加权最小链接 nwc[ ];
13　　若多个任务同时到达
14　　　选择调度关系 R 中的任务
15　　否则
16　　　选择最早完成时间 EFT[ ]最小的任务 t_i;
17　　选择任务 t_i 最早完成时间 EFT［i］最小的雾计算结点 F_j 上调度;
18　　计算任务调度时间 WT;
19　　更 ST[ ]和 FT[ ];
20　　任务降序排列;
21　}
22　计算并输出 makespan 以及 AWT;
23 }

表 1　任务到达时间

Table 1　Arrival time of task　　ms

任务	t_0	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9
到达时间	20	0	20	35	35	20	22	40	38	50

表 2　任务集在处理机集上的执行开销矩阵

Table 2　Execution time of task set on fog node set　　ms

任务	F_0	F_1	F_2	F_3
t_0	100	111	90	113
t_1	110	121	105	128
t_2	78	86	70	92
t_3	66	73	62	75
t_4	120	135	109	140
t_5	145	155	138	160
t_6	199	210	186	215
t_7	103	108	99	110
t_8	90	98	82	101
t_9	210	225	201	230

表3　事务集

Table 3　Transaction set

事务	F_0	F_1	F_2	F_3	t_0	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9
T_1	1	1	1	1	1	0	1	0	0	1	0	1	0	1
T_2	0	1	0	1	0	0	0	1	0	0	0	0	1	0
T_3	1	1	1	1	0	1	1	0	1	0	0	1	0	0
T_4	1	0	1	1	0	0	0	0	0	1	0	1	1	0
T_5	0	1	0	1	0	1	1	0	0	0	0	0	1	0
T_6	0	0	0	1	0	0	0	1	0	0	0	0	0	0
T_7	1	1	0	1	0	0	0	1	0	1	0	1	0	0
T_8	0	1	1	0	1	0	0	0	0	0	0	0	1	0
T_9	1	1	1	1	0	0	0	1	1	0	0	1	1	1
T_{10}	1	1	1	1	0	1	1	0	0	0	1	0	1	0

以及$\{F_2,F_0,F_3,t_3\}$两个频繁项集；

2）产生关联规则。设最小置信度为0.7，利用生成的频繁项集，得到两个关联规则，分别为$t_2 \Rightarrow F_1 \vee F_0 \vee F_2$，$t_3 \Rightarrow F_2 \vee F_0 \vee F_3$.

3.3.4　任务调用关系　根据I-Apriori算法得出的两个关联规则，生成对应的任务调度关系。任务调用关系是任务集T中的任务t_i与雾计算结点集F中的结点F_j之间的强关系的集合，即出现在任务调度关系中的任务t_i优先被调度。t_i和F_j存在3种关系：① 若t_i与F_j未出现在关联规则中，则t_i行对应的值全部为0，其中t_0，t_1，t_4，t_5，t_6，t_7，t_8，t_9与F_j未出现在关联规则中；② 若t_i与F_j出现在关联规则中，则计算任务t_i在雾计算结点F_j上的置信度，t_2在F_0，F_1，F_2上的置信度分别为0.33，0.37和0.30，t_3在F_0，F_1，F_2，F_3上的置信度分别为0.33，0，0.35和0.32；③ 没有出现在关联规则中的雾计算结点，对应的位置值为-1，F_3未出现在t_2中。

3.3.5　ITPS算法调度　利用ITPS算法进行任务调度，以表1~2和任务调用关系作为输入，输出任务集任务调度跨度makespan以及平均等待时间AWT。

以任务集$\{t_0,t_1,t_2,t_7\}$作为调度对象，由于任务t_1最早到达，所以先对任务t_1进行调度，选择最早完成时间最小的雾计算结点F_2进行调度；接下来选择任务t_0和t_2，由于t_0，t_2任务同时到达，且任务t_2出现在关联关系R中，因此任务t_2被优先调度，然后选择最早完成时间最小的雾计算结点F_0进行调度；任务集中还有任务t_0和t_7未被调度，由于任务t_0早于任务t_7到达，因此选择任务t_0在F_1上调度；最后再将任务t_7调度到F_3，得到任务集$\{t_0,t_1,t_2,t_7\}$与雾计算结点集$\{F_0,F_1,F_2,F_3\}$的调度关系如图4所示。

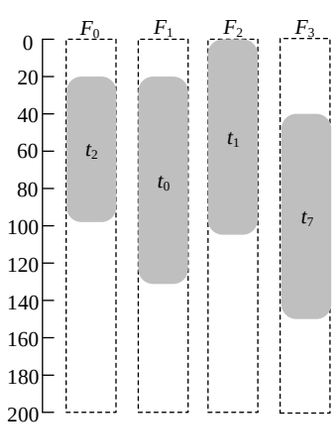

图4　任务与雾计算结点之间调度关系图

Fig. 4　Scheduling relationship between tasks and fog computing nodes

4 模拟结果与结果分析

为了验证ITPS算法的性能和效率，搭建基于SimGrid[22-24]模拟器工具包的仿真实验环境。实验环境中雾计算结点间通过高速网络连接，雾计算结点的计算性能异构，雾计算结点进行任务调度的同时也可以与其他雾计算结点进行通信，独立流水线任务分配到雾计算结点上执行直到任务执行完成。

实验中使用的计算机配置：Intel Core（TM）i5-8265U 1.6 GHz CPU、4 GB内存硬件环境、Ubuntu12操作系统以及SimGrid3.11工具包，在相同实验条件对ITPS算法以及wLC和wRR算法进行性能比较。

4.1 测试数据集

为测试各种算法的性能，需要给定任务调度算法的测试数据集，包括任务集、雾结点集、任务执行开销矩阵、事务集以及调度关系等。任务测试集中的任务数从50开始，每次递增50个任务，最大到1 000个任务，任务集中的任务与任务编号通过随机程序生成，任务集中每个任务的到达时间通过随机程序生成；雾结点测试数据集包括4个性能异构的结点；任务集在雾结点集上的执行开销通过随机程序生成；调度关系由I-Apriori算法根据初始事务集D得到；初始事务测试集由500个随机生成的事务构成。

4.2 实验结果分析

利用测试数据集，分别执行ITPS、wLC以及wRR任务调度算法，对不同任务数及不同任务到达时间对任务集进行调度，任务到达时间区间为[0,50]的调度情况，以及任务到达时间区间为[0,100]的调度情况图5所示。

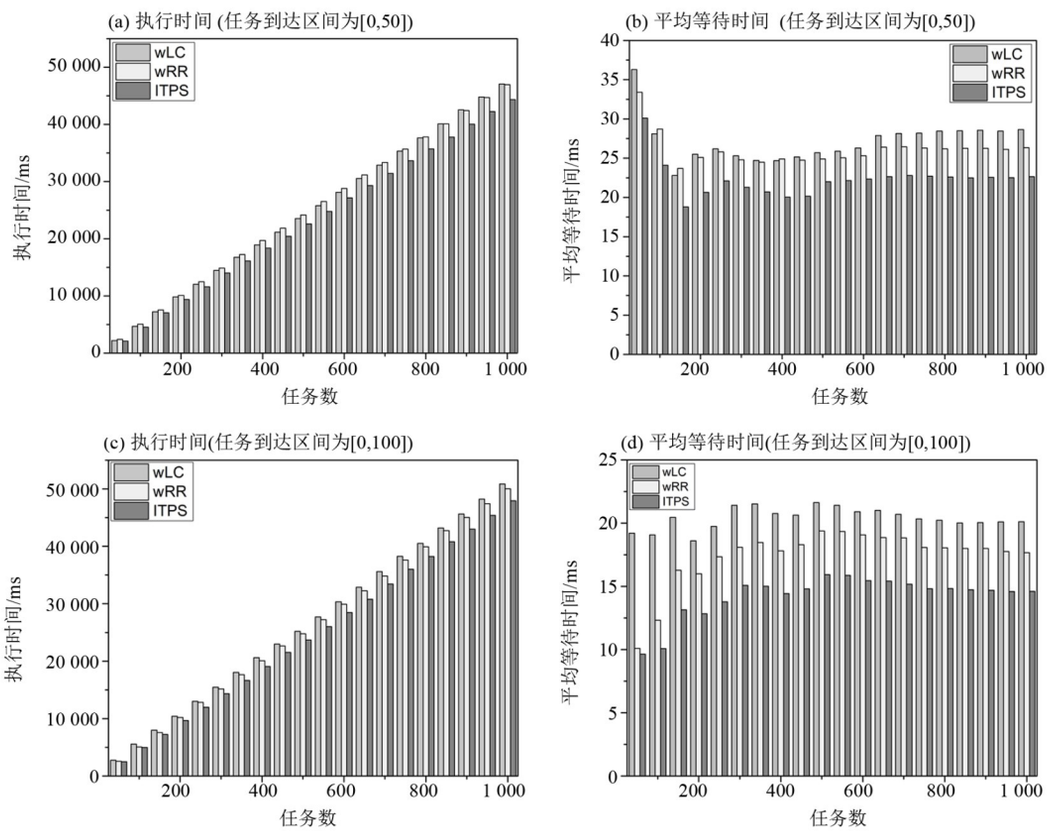

图5 不同任务数性能对比

Fig. 5 Performance comparison of different tasks

通过上述实验可以得到，ITPS算法在不同任务数以及不同任务到达时间条件下，任务调度跨度和平均等待时间均比wLC和wRR算法要短，而wRR算法的性能总体上要优于wLC算法。ITPS算法相比wLC算法的任务调度跨度缩短2.48%~10.62%，平均等待时间AWT值缩短14.97%~49.79%，任务越晚到达，IPTS的性能更优；ITPS算法相比wRR算法的任务调度跨度缩短2.19%~11.78%，平均等待时间AWT值缩

短4.55%~20.53%，任务越早到达，IPTS的性能更佳，总体上ITPS算法要优于wLC和wRR两种任务调度算法。

5　结　语

通过对经典Apriori算法的改进，提出一种雾计算环境中任务的分类I-Apriori算法，I-Apriori算法提升了分类挖掘的效率和性能；利用I-Apriori算法，面向雾计算环境下的任务调度问题，建立雾计算任务调度模型，并提出ITPS任务调度算法。ITPS算法在makespan及AWT具有更好的性能。

参考文献：

[1] 刘林东.分布式异构环境中任务调度算法研究[D].广州：华南理工大学，2019.
LIU L D. Researches on task scheduling algorithm in distributed heterogeneous environment [D]. Guangzhou: South China University of Technology, 2019.

[2] ZHU Q L, SI B J, YANG F F, et al. Task offloading decision in fog computing system [J]. China Communications, 2017, 14(11):59-68.

[3] MUKHERJEE M, SHU L, WANG D. Survey of fog computing: fundamental, network applications, and research challenges [J]. IEEE Communications Surveys & Tutorials, 2018,20(3): 1826-1857.

[4] PULIAFITO C, MINGOZZI E, ANASTASI G. Fog computing for the internet of mobile things: issues and challenges [C]// IEEE International Conference on Smart Computing, 2017.

[5] 刘林东, 邬依林. 多DAG任务调度算法[J]. 中山大学学报(自然科学版), 2019, 58(4): 99-107.
LIU L D, WU Y L. Multi-DAG task scheduling algorithms [J]. Acta Scientiarum Naturalium Universitatis Sunyatseni, 2019, 58(4): 99-107.

[6] PHAM X Q, HUH E N. Towards task scheduling in a cloud-fog computing system [C]//IEEE Network Operations & Management Symposium, 2016.

[7] YI S, HAO Z, QIN Z, et al. Fog computing: platform and applications [C]//The third IEEE Workshop on Hot Topics in Web Systems & Technologies, 2015.

[8] LYU X, REN C, NI W, et al. Distributed optimization of collaborative regions in large-scale inhomogeneous fog computing [J]. IEEE Journal on Selected Areas in Communications, 2018: 574-586.

[9] ZHANG Y H, ZHENG D, DENG R H. Security and privacy in smart health: efficient policy-hiding attribute-based access control [J]. IEEE Internet of Things Journal, 2018, 5(3): 2130-2145.

[10] HUANG M, LIU Y, ZHANG N, et al. A services routing based caching scheme for cloud assisted CRNs [J]. IEEE Access, 2018,6(1): 15787-15805.

[11] LIU X, DONG M, LIU Y, et al. Construction low complexity and low delay CDS for big data codes dissemination [J]. Complexity, 2018:1-19.

[12] NI L, ZHANG J, JIANG C, et al. Resource allocation strategy in fog computing based on priced timed Petri nets [J]. IEEE Internet of Things Journal, 2017,4(5): 1216-1228.

[13] ZENG D Z, GU L, GUO S, et al. Joint optimization of task scheduling and image placement in fog computing supported software-defined embedded system [J]. IEEE Transactions on Computers, 2016, 65(12): 3702-3712.

[14] OUEIS J, STRINATI E C, SARDELLITTI S, et al. Small cell clustering for efficient distributed fog computing: a multi-user case [C]//Vehicular Technology Conference, IEEE, 2016.

[15] INTHARAWIJITR K, IIDA K, KOGA H. Analysis of fog model considering computing and communication latency in 5G cellular networks [C]// IEEE International Conference on Pervasive Computing & Communication Workshops, 2016.

[16] DENG R, LU R, LAI C, et al. Optimal workload allocation in fog-cloud computing toward balanced delay and power consumption [J]. IEEE Internet of Things Journal, 2016,3(6):1171-1181.

[17] PHAM X Q, HUH E N. Towards task scheduling in a cloud-fog computing system [C]// IEEE Network Operations & Management Symposium, 2016.

[18] TANG C, WEI X, XIAO S, et al. A mobile cloud based scheduling strategy for industrial internet of things [J]. IEEE

Access, 2018,6(3): 7262-7275.

[19] DANG T, HOANG D. FBRC: optimization of task scheduling in fog-based region and cloud [C]//IEEE Trustcom/BigDataSE/ICESS, 2017:1109-1114.

[20] YANG J, HUANG H, JIN X. Mining web access sequence with improved apriori algorithm [C]// IEEE International Conference on Computational Science & Engineering, 2017.

[21] ZHANG S, DU Z, WANG J T L. New techniques for mining frequent patterns in unordered trees [J]. IEEE Transactions on Cybernetics, 2015,45(6):1113-1125.

[22] BRENNAND C A R L, DUARTE J M, SILVA A P. SimGrid: A simulator of network monitoring topologies for peer-to-peer based computational grids [C]// IEEE 8th Latin-American Conference on Communications (LATINCOM), 2016.

[23] DEGOMME A, LEGRAND A, MARKOMANOLIS G S, et al. Simulating MPI applications: the SMPI approach [J]. IEEE Transactions on Parallel and Distributed Systems, 2017, 28(8): 2387-2400.

[24] MOHAMMED A, ELELIEMY A, CIORBA F M. Towards the reproduction of selected dynamic loop scheduling experiments using SimGrid-SimDag [C]//IEEE 19th International Conference on High Performance Computing and Communications; IEEE 15th International Conference on Smart City; IEEE 3rd International Conference on Data Science and Systems, 2017.

（责任编辑 冯兆永）

声 明

本刊决定对以下论文进行撤稿。

1. “8”字图的匹配能级和Hosoya指标全排序. 2019,58(1):144-148

2. LDH-CO3-SO4-Cl晶须的合成及其结构. 2019,58(2):121-127

3. 次线性g-期望的性质及其应用. 2019,58(5):153-158

特此声明。

本刊编辑部

2021年9月25日