40 return *RSM*;

综上所述,基于已得到的模型子集对象数组 *RS*、模型子集的物理倒排表 *RVS*,根据自动聚类模型子集的等检索代价分配,下面给出基于自动聚类模型集的静态并行检索的实现过程,如算法 4 所示.

算法 4. 基于自动聚类模型集的静态并行检索算法.

输入:*R*; //大规模流程模型集

RS; //*R* 通过自动聚类法得到模型子集数组

RVS; //模型子集的物理倒排表

q; //目标查询模型

m. //处理器个数

输出:R_qS. //模型库 *R* 中覆盖 *q* 的模型集

```
1  ResultList=∅;  //定义全局检索结果集
2  k=0;  //定义表示模型子集分组编号的全局变量
3  MS=GetCPU(m);  //得到具有 m 个处理器的数组
4  for each rs in RS do  //计算查询模型 q 与各模型子集的检索代价预估值
5  {  po=GetCenterPointProcess(rs,RVS);
6     qc=QC(rs,q);
7     add (rs,qc) to RPS;
8  }
9  RSM=GetOptimalGroup(RPS,m);  //将 n 个模型子集分成检索代价最接近的 m 组模型子集
10 for each cpu in MS do  //将 RSM 中 m 组模型子集分别分配给 m 个处理器
11 { k=k+1;
12 for each rs in RSM[k] do
13 { cpu.addTaskData(rs);}
14 }
15 for each cpu in MS do
16 { cpu.executeTask(QueryProcess(q));
17 } }
//返回归总后的查询结果集
18 if (AllCPUTaskIsFinished()=true )
19    RqS=ResultList;
20 return RqS;
```

在算法 4 中:第 1 行是定义一个全局的检索结果集 *ResultList* 用于存储各处理器的的检索结果,*ResultList* 的初始值为空;第 2 行定义一个全局变量 *k*;第 3 行用于获取一个具有 *m* 个处理器的处理器对象数组;第 4 行~第 8 行用于根据定义 1 计算查询模型 *q* 与各模型子集的检索代价预估值,并存于 *RPS* 集合中;第 9 行是根据 *RPS* 数组中每一模型子集的检索代价预估值,采用等检索代价的模型子集最优分配函数 *GetOptimalGroup*(*RPS*,*m*)将 *n* 个模型子集分成检索代价最接近的 *m* 组模型子集,存于集合 *RSM* 中;第 10 行~第 14 行用于将 *RSM* 中 *m* 组模型子集分别分配给 *m* 个处理器;第 15 行~第 17 行用于在 *m* 个处理器中执行检索任务 *QueryProcess*(*q*)(*QueryProcess* 的实现过程同上).最后,第 18 行~第 20 行用于返回归总后的查询结果集.

3.5 基于自动聚类模型集的动态并行检索算法

对于 *n* 个通过自动聚类法得到的模型子集,当采用动态方式来为 *m* 个处理器分配相应的模型子集时,每次尽可能将检索代价最高的模型子集分配给空闲的处理器,以保证 *m* 个处理器的并行检索效率达到最优.因此,基于已得到的模型子集对象数组 *RS*、模型子集的物理倒排表 *RVS*,根据自动聚类模型集的检索成本预估值,提出

了基于自动聚类模型集的动态并行检索算法,如算法 5 所示.

算法 5. 基于自动聚类模型集的动态并行检索算法.

```
输入:R;  //大规模流程模型集
     RS;  //R 通过自动聚类法得到的模型子集对象数组
     RVS;  //模型子集的物理倒排表
     q;  //查询模型
     m.  //处理器个数
输出:RqS.  //模型库 R 中覆盖 q 的模型集
1  ResultList=∅;  //定义一个全局检索结果集
2  k=0;  //定义一个表示模型子集计数的全局变量
3  MS=GetCPU(m);  //得到具有 m 个处理器的处理器对象数组
4  for each rs in RS do  //计算查询模型 q 与各模型子集的检索代价预估值
5  {  po=GetCenterPointProcess(rs,RVS);
6     qc=QC(rs,po,q);
7     add (rs,qc) to RPS;
8  }
9  for each cpu in MS do  //将 m 个模型子集分别分配给 m 个处理器,并执行检索任务
10   {  rs=GetMaxQC(RS,RPS);
11     cpu.addTaskData(rs);
12     cpu.executeTask(QueryProcess(q));
13     k=k+1;
14   }
15 while (AllCPUTaskIsFinished()!=true && k!=n)
16 {  cpu=GetFinishedCPU(MS);
17   if (cpu!=null)
18   {  rs=GetMaxQC(RS,RPS);
19     cpu.addTaskData(rs);
20     cpu.executeTask(QueryProcess(q));
21     k=k+1;
22   }
23 }
//返回归总后的查询结果集
24 if (AllCPUTaskIsFinished()=true )
25   RqS=ResultList;
26 return RqS;
```

在算法 5 中:第 1 行是定义一个全局的检索结果集 *ResultList* 用于存储各处理器的的检索结果,*ResultList* 的初始值为空;第 2 行定义一个全局变量 *k*;第 3 行用于获取一个具有 *m* 个处理器的处理器对象数组;第 4 行~第 8 行用于根据定义 1 计算查询模型 *q* 与各模型子集的检索代价预估值,并存于 *RPS* 集合中;第 9 行~第 14 行是根据 *RPS* 数组中模型子集的检索代价预估值,按从大到小的顺序将前面 *m* 个模型子集分别分配给 *m* 个处理器,并在分配完毕的同时,执行检索任务:*QueryProcess*(*q*)(*QueryProcess* 的实现过程同上);第 15 行~第 23 行用于监听处理器执行状况,并为每一空闲处理器分配一个没有被检索过的具有最大检索代价的模型子集,直到所有模型子集都检索完毕;最后,第 24 行~第 26 行用于返回归总后的查询结果集.

综上所述,对于上述 4 个并行检索算法,最希望达到的是:n 个模型子集的检索任务量在 m 个处理器间平均分配,又不会为每个处理器引入额外的工作量.如果能成功达到该目标,在 m 个处理器进行并行检索的查询效率就是在单机(单处理器)环境下相应检索算法的 m 倍.但事实上不可能达到该目标,因为在多处理器的并行检索中会引入一些其他代价,如模型子集(静态/动态)分配给各处理器的时间消耗、各子处理器与主程序(主处理器)的通信开销等.这些代价必然削减并行检索算法的查询效率.但不管怎样,当采用一定数目的处理器进行并行检索时,可以显著提高大规模流程模型集的检索效率.下面将给出相关评估与验证实验.

4 实验与评估

为了验证和评估本文的方法,我们基于陶瓷云科技服务平台框架研发了一个云工作流管理平台(cloud workflow management platform,简称 CWMP),其访问网址是http://platform.pasp.cn,通过陶瓷云科技服务平台的统一登录界面可以进入云工作流管理平台的操作界面.该平台主要包括 5 个工具:模型自动生成工具、模型导入与转换工具、模型可视化建模工具、模型查询工具及结果显示工具.模型自动生成工具用于自动生成满足实验需求的模拟流程模型.模拟流程的所有节点标签都是通过预先设定的方式从 WordNet 词库自动选取;每一节点的类型、汇合属性及分支属性也是按一定的规则随机分配的.模型自动生成工具中,这些用于生成控制流的规则都基本来源于文献[38].模型导入与转换工具用来导入其他格式(如 Petri-net,YWAL,BPMN,EPC 等[39,40])的模型,并转换成文献[3]中 CNet 模型.所有的 CNet 模型可以通过模型可视化显示工具(http://cbpm.pasp.cn)进行展示;最后的实验结果及结果分析可以过结果显示工具进行查看.

为了验证和评估算法 2(用 PQ-DESA 表示)、算法 3(用 PQ-DEDA 表示)、算法 4(用 PQ-DCSA 表示)、算法 5(用 PQ-DCDA 表示)的效率,文中以文献[3]中的算法(假定为算法 0,用 TS-QTCS 表示)为比较基准,在云工作流管理平台中,基于模拟生成的大规模流程模型库及真实的云工作流模型库做了大量验证实验.在这些实验中,所有模型子集的索引,如复合节点索引及标签索引,都是通过 Apache Lucene(the Web site of apache lucene, http://lucene.apache.org)进行管理和维护;另外,实验用服务器共 5 台,型号为曙光 A840-G20.每一服务器都采用服务器专用芯片组,配有 4 颗 AMD Opteron6320 CPU(2.8GHz,8 核,16M 三级缓存),256GB 内存,并集成 Intel 四口千兆网卡.

4.1 合成数据集上的实验验证

为了分析算法 2~算法 5 的查询响应时间与模型集大小之间的关系,并比较这 4 种算法与算法 0 的查询效率,基于模型自动生成工具生成的 6 个大规模流程模型集(它们的模型数分别 100 000~600 000)做了一些评估实验(实验中设定模型子集数 n 为 30,处理器个数 m 为 10),其实验结果如图 5 所示.

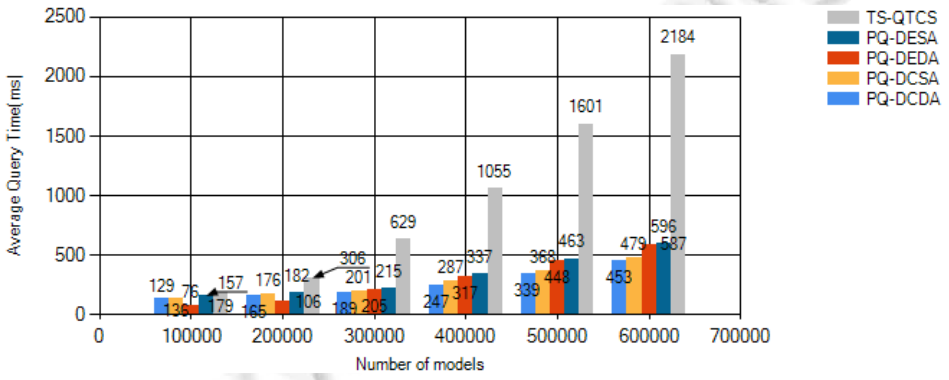

Fig.5 Comparion of query efficiency with the Algorithm 0 in the composite data set

图 5 在合成数据集上与算法 0 的查询效率比较

在图 5 中,随流程模型集规模的增大,算法 2(PQ-DESA)、算法 3(PQ-DEDA)、算法 4(PQ-DCSA)、算法 5(PQ-DCDA)及算法 0(TS-QTCS)都要花费更多查询响应时间.同时,对于同一大小的模型集,算法 2~算法 5 的平均查询响应时间明显少于算法 0 的平均查询响应时间,并且随流程模型数量的增多,算法 2~算法 5 将节省更多的查询响应时间.

根据上述实验结果可以得出,在大规模流程模型库中进行流程检索时,应用基于数据集分割的并行检索方法能进一步提高流程检索效率.

此外,为了量化上述 4 个并行检索算法(算法 2~算法 5)与单机(单处理器)环境下相应检索算法(也称串行检索算法,如算法 0)查询效率的比较结果,并且对比这 4 个并行检索算法的效率.下面给出一个大规模流程模型集的并行检索加速比的概念,如定义 3 所示:

定义 3(并行检索加速比). 假定大规模流程模型集为 R,对于同一查询请求,用 T_{pq} 表示采用并行检索算法 PA 在 R 中的查询响应时间,用 T_{sq} 表示采用串行检索算法 SA 在 R 中的查询响应时间.那么与串行检索算法 SA 相比,并行检索算法 PA 在 R 中的并行检索加速比可表示为 $PQS(PA,SA)$,其计算公式如公式(4)所示.

$$PQS(PA,SA)=T_{sq}/T_{pq} \tag{4}$$

对于定义 3,在理想状态(不考滤并行检索中模型子集分配、处理器间的通信开销及检索结果归并等代价)下,用 m 个处理器进行并行检索时,可以认为 $T_{pq}=T_{sq}/m$,此时,根据公式(4)可以得到 $PQS(PA,SA)=m$,称为并行检索的线性加速比,也即理想加速比.但在实际的流程并行检索中,由于 m 个处理器的并行处理引入了其他检索代价(用 T_c 表示其他检索代价的时间消耗),此时,$T_{pq}=T_{sq}/m+T_c$.根据公式(4),可以得到 $PQS(PA,SA)$也可由公式(5)来计算.

$$PQS(PA,SA)=\frac{T_{sq}}{T_{sq}/m+T_c}=\frac{m}{1+m\times(T_c/T_{sq})} \tag{5}$$

在实际的流程并检索中,T_c 不可能等于 0.因此根据公式(5),可以得出基于数据集分割的流程并行检索不可能获得线性加速比.

如图 5 所示,可以得出,与算法 0 相比,算法 2~算法 5 对不同规模流程集的并行检索加速比的值见表 1.

Table 1 Parallel retrieval acceleration ratio of Algorithm 2~Algorithm 5 compared with Algorithm 0

表 1 算法 2~算法 5 与算法 0 的并行检索加速比

模型数	算法 2	算法 3	算法 4	算法 5
100 000	1.14	1.25	1.32	1.39
200 000	1.68	1.79	1.74	1.85
300 000	2.93	3.07	3.13	3.32
400 000	3.13	3.33	3.68	4.27
500 000	3.46	3.57	4.35	4.72
600 000	3.66	3.72	4.56	4.82

根据表 1 所示:随流程模型集规模的增大,与算法 0 相比,算法 2~算法 5 的并行检索加速比也在增加.这是因为根据公式(5),随流程模型集规模的增大,T_{sq} 比 T_c 的增长会相对快些,也即多处理器并行中的其他开销时间相对变少了.因此,并行检索加速比会随流程模型集规模的增大而增加.

此外,为了对比算法 2~算法 5 的检索效率,本文用每一并行算法在不同规模流程模集下与同一串行算法(算法 0)的并行加速比的平均值(也即平均并行检索加速比)来衡量它们之间的检索效率.根据表 1 可以得到:与算法 0 相比,算法 2~算法 5 的平均并行检索加速比见表 2.

Table 2 Average parallel retrievable acceleration ratio

表 2 平均并行检索加速比

串行算法	算法 2	算法 3	算法 4	算法 5
算法 0	2.67	2.79	3.13	3.40

如表 2 所示,与算法 0 相比,4 个并行检索算法的平均加速比从大到小的顺序为:算法 5、算法 4、算法 3、

算法 2.这表明,在检索效率方面,基于自动聚类模型集的并行检索算法优于基于均匀划模型集的并行检索算法;而应用动态方式进行模型子集分配的并行检索算法优于应用静态方式进行模型子集分配的并行检索算法.

首先,基于均匀划分法模型集的算法在给处理器分配模型子集时,无论是静态分配还是动态分配,虽然都比较容易实现,但由于每一模型子集中各模型差异性较大(也即模型子集的模型特性多样化),对于同一查询模型,规模相同的各模型子集的检索代价也将各不相同;基于均匀划分法模型集的算法在分配时忽略它们之间检索代价的差异,这样难以保证多处理器并行检索的效率达到最优.而基于自动聚类模型集的算法,不管是静态分配还是动态分配,都通过检索代价的预估值(定义 1)来度量这种差异.在以静态方式进行模型子集分配时,按检索代价尽可能均匀分给各处理器;在以动态方式进行模型子集分配时,尽可能将检索代价高的模型子集先分给处理器进行处理,以减少处理器的空闲或等待时间,这样就可以充分发挥多处理器的并行性能,提高流程并行检索的效率.

其次,应用动态方式进行模型子集分配的算法也将明显优于应用静态方式进行模型子集分配的算法.这是因为静态分配方式虽可以减少各子处理器与主处理器(主程序)通信开销,但该方式可能会闲置部分处理器,造成处理器资源浪费,最终的检索时间会受限于任务最重、处理时间最长的处理器执行时间;而动态分配方式虽然会增加各子处理器与主处理器(主程序)通信成本,但当处理器间通信成本远小于大规模流程模型集检索的查询成本时,该方式会充分利用各处理器的并行检索能力,以提高大规模流程模型库检索的效率.

4.2 真实数据集上的实验验证

为了观察在确定的模型子集数下算法 2~算法 5 的查询响应时间与用于查询的处理器数目之间的关系,基于陶瓷云服务平台中的云工作流模型库做了相关比较实验(实验中设定模型子集数 n 为 30,处理器个数 m 分别取 4,6,8,10,12,14),其最后的实验结果如图 6 所示.

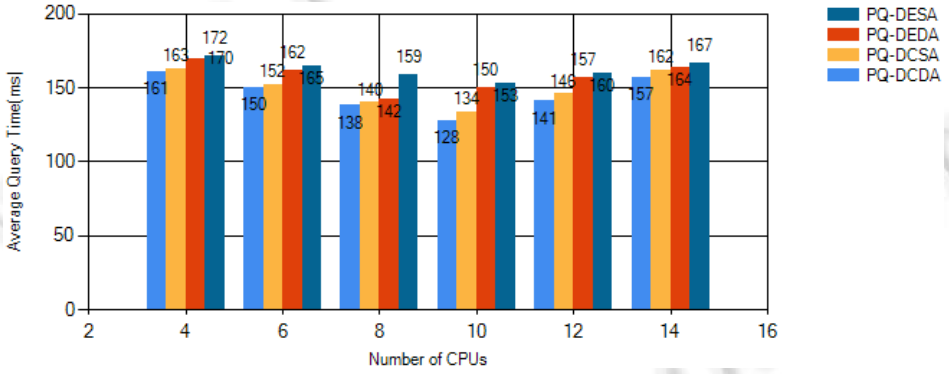

Fig.6 Relationship between the query response time and the number of parallel processors

图 6 查询响应时间与并行处理器数目之间的关系

如图 6 所示,在模型集规模及其合理划分的模型子集数确定的情况下,算法 2~算法 5 的查询响应时间总的趋势都是先随并行处理器的数目增多而减少,当并行处理器数达到一定数目后,再随并行处理器数目的增多而增加.这是因为当处理器达到一定数目后,并行处理器中的其他检索代价将花费越来越多时间,从而造成整个并行算法查询响应时间的增多.对于不同的算法,其查询响应时间达到最小值的处理器数目也并不一致,其中,算法 2、算法 4 及算法 5 的查询响应时间达到最小值的处理器数目比算法 3 要大,这表明算法 3 的查询响应时间对并行处理器数的变化最为敏感,算法 3 随并行处理器数的增多,最先达到查询响应时间的变化拐点.

此外,为了观察在确定的查询处理器数目下算法 2~算法 5 的查询响应时间与模型子集数之间的关系,本文基于陶瓷云服务平台中的云工作流模型库也做了相关比较实验(实验中设定处理器个数 m 为 10,模型子集数 n 分别取 10,20,30,40,50,60),其最后的实验结果如图 7 所示.

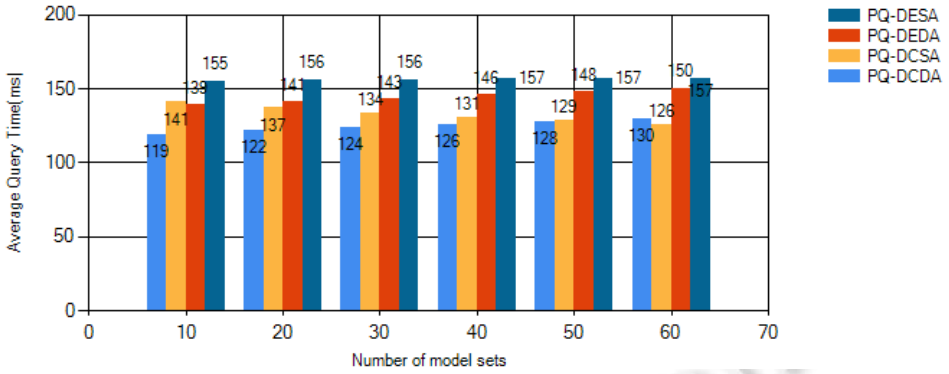

Fig.7 Relationship between the query response time and the number of model subsets

图 7 查询响应时间与模型子集数之间的关系

如图 7 所示,在模型集规模及并行处理器数目确定的情况下,算法 2 的查询响应时间受模型子集数变化的影响较少;算法 3、算法 5 的查询响应时间随模型子集数目增多而增加;而算法 4 的查询响应时间则随模型子集数目增多而减少.这是因为无论模型子集数是多少,算法 2 的模型子集分配结果都是按处理器数进行均分,最终的查询响应时间只受处理器数影响,与模型子集数无关;而算法 3、算法 5 在处理器数确定的情况下,其调度的通信开销与模型子集数紧密相关,模型子集数越多,处理器调度的通信开销越大,因此,算法 3、算法 5 的查询响应时间将会越多;对于算法 4,其检索代价的预估值随模型子集(聚类结果集)的增多将会获得更精确的估算值,也就可以更精确的将整个大规模流程模型集的总检索代价均分到确定数目的处理器上,这样可以使并行检索的性能达到最优化,算法 4 的查询响应时间将会减少.

综上所述,当设定合适数量的模型子集数及查询处理器数,基于数据集分割的流程并行检索算法都能进一步提高大规模云工作流模型库的检索效率.

5 结 论

为了提高大规模云工作流模型库检索的效率,本文提出了一个基于数据集分割的流程并行检索方法.该方法通过均匀划分法或自动聚类法对大规模流程模型库(集)进行模型子集的合理划分,然后应用前期工作中改进后的两阶段流程检索算法,基于两种模型划分集及两种子集分配方式提出了 4 种流程并行检索算法:基于均匀划分模型集的静态并行检索算法、基于均匀划分模型集的动态并行检索算法、基于自动聚类模型集的静态并行检索算法及基于自动聚类模型集的动态并行检索算法.为了评估我们所提出的方法,基于模拟生成的大规模流程模型库及真实的云工作流模型库上做了大量验证实验.实验结果表明,基于数据集分割的流程并行检索方法能进一步提高大规模云工作流模型库的检索效率.此外,在大规模云工作流模型库的检索中,基于自动聚类模型集的并行检索算法比基于均匀划分模型集的并行检索算法具有更好的检索效率;采用动态方式给处理器分配模型子集的并行检索算法比采用静态方式给处理器分配模型子集的并行检索算法具有更好的检索效率.

综上所述,本文提出的基于数据集分割的流程并行检索方法可以应用在大规模云工作流模型库的高效检索中.在下一步工作中,我们将应用 MapReduce 模型实现基于图结构/行为的流程检索算法的并行化,并将其应用在真正的分布式并行计算环境(如 Hadoop 生态系统)中,以进一步提高大规模云工作流模型库的检索效率.

致 谢 感谢清华大学软件学院的王建民老师、闻立杰老师、金涛老师与我分享他们所掌握的实验数据资源.

References:

[1] Baeyens T. BPM in the cloud. In: Proc. of the Int'l Conf. on Business Process Management. LNCS 8094, Berlin, Heidelberg: Springer-Verlag, 2013. 10–16.

[2] Chai XZ, Cao J. Cloud computing oriented workflow technology. Journal of Chinese Computer Systems, 2012,33(1):90−95 (in Chinese with English abstract).

[3] Huang H, Peng R, Feng ZW. Efficient and exact query of large process model repositories in cloud workflow systems. IEEE Trans. on Services Computing, 2018,11(5):821−832. [doi: http://doi.ieee computersociety.org/10.1109/TSC.2015.2481409]

[4] Song W, Xia X, Jacobsen HA, *et al*. Efficient alignment between event logs and process models. IEEE Trans. on Services Computing, 2017,10(1):136−149.

[5] Song W, Jacobsen HA. Static and dynamic process change. IEEE Trans. on Services Computing, 2018,11(1):215−231. [doi: 10.1109/ TSC.2016.2536025]

[6] Zhang ZW, Cui GM, Zhao S. IFOA4WSC: A quick and effective algorithm for QoS-aware service composition. Int'l Journal of Web and Grid Services, 2016,12(1):81−108.

[7] Song W, Jacobsen HA, Ye C, *et al*. Process discovery from dependence-complete event logs. IEEE Trans. on Services Computing, 2016,9(5):714−727.

[8] Zhang ZW, Cui GM, Deng SG, He Q. Alliance-Aware service composition based on quotient space. In: Proc. of the IEEE Int'l Conf. on Web Services (ICWS). San Francisco, 2016. 340−347.

[9] ter Hofstede AHM, Ouyang C, La Rosa M, Song L, Wang JM, Polyvyanyy A. APQL: A process-model query language. Lecture Notes in Business Information Processing, 2013,159:23−38.

[10] Beeri C, Eyal A, Kamenkovich S, *et al*. Querying business processes with BP-QL. Information Systems, 2008,33(6):477−507.

[11] Sakr S, Awad A. A framework for querying graph-based business process models. In: Proc. of the 19th Int'l Conf. on World Wide Web. ACM Press, 2010. 1297−1300.

[12] Dijkman R, Dumas M, García-Bañuelos L. Graph matching algorithms for business process model similarity search. In: Proc. of the Int'l Conf. on Business Process Management. Springer-Verlag, 2009. 835−842.

[13] Cao B, Yin JW, Chen HR. Levenshtein distance based process retrieval method. Computer Integrated Manufacting System, 2012, 18(8):1766−1773 (in Chinese with English abstract).

[14] Yan Z, Dijkman R, Grefen P. Fast business process similarity search with feature-based similarity estimation. In: Proc. of the Int'l Conf. on the Move to Meaningful Internet Systems. Springer-Verlag, 2010. 60−77.

[15] Qiao M, Akkiraju R, Rembert AJ. Towards efficient business process clustering and retrieval: Combining language modeling and structure matching. In: Proc. of the Int'l Conf. on Business Process Management. Springer-Verlag, 2011. 199−214.

[16] Dumas M, García-Bañuelos L, Dijkman RM. Similarity search of business process models. IEEE Data Eng.Bull, 2009,32(3):23−28.

[17] Kunze M, Weske MM. Metric trees for efficient similarity search in large process model repositories. Lecture Notes in Business Information Processing, 2010,66:535−546.

[18] Jin T, Wang J, Wen L. Efficiently querying business process models with BeehiveZ. In: Proc. of the Demo Track of the 9th Conf. on Business Process Management. 2011.

[19] Yan Z, Dijkman R, Grefen P. FNet: An index for advanced business process-querying. LNCS, 2012,7481:246−261.

[20] Jin T, Wang J, Rosa ML, *et al*. Efficient querying of large process model repositories. Computers in Industry, 2013,64(1):41−49.

[21] Ullmann JR. An algorithm for subgraph isomorphism. Journal of the ACM, 1976,23(23):31−42.

[22] Ryeng NH, Vlachou A, Doulkeridis C, *et al*. Efficient distributed top-*k* query processing with caching. In: Proc. of the Int'l Conf. on Database Systems for Advanced Applications. Springer-Verlag, 2011. 280−295.

[23] Jiang H, Cheng J, Wang D, *et al*. A general framework for efficient continuous multidimensional top-*k* query processing in sensor networks. IEEE Trans. on Parallel & Distributed Systems, 2012,23(9):1668−1680.

[24] Wang X, Shen D, Yu G. Uncertain top-*k* query processing in distributed environments. In: Proc. of the Distributed & Parallel Databases, 2015. 1−23.

[25] Wang B, Zhang G, Sun J. Large-Scale distributed parallel information retrieval technology. Information Technology Express, 2005, (2):1−9. (in Chinese with English abstract).

[26] Peng D. Parallel information retrieval and correlative technology. Research and Exploration of Education in China, 2007,(1):84−86 (in Chinese with English abstract).

[27] Chen GL, Sun GZ, Xu Y, Lu M. Methodology of research on parallel algorithm. Chinese Joural of Computers, 2008,31(9): 1493−1502 (in Chinese with English abstract).

[28] Liu Y, Chen L, Jing N, Liu L. Parallel top-*k* spatial join query processing on massive spatial data. Joural of Computer Research and Development, 2011,48(S3):163−172 (in Chinese with English abstract).

[29] Wu C. Parallel algorithm and optimization of top-*k* problems in information retrieval [Ph.D. Thesis]. Hefei: University of Science and Technology of China, 2011 (in Chinese with English abstract).

[30] Qi M. Spatial data retrieval and optimization research on shared memory parallel system [Ph.D. Thesis]. Hefei: University of Science and Technology of China, 2012 (in Chinese with English abstract).

[31] Zhu J. Large scale distributed parallel information retrieval technology. Computer CD Software and Application, 2013,(21): 305−306 (in Chinese with English abstract).

[32] Ge FJ. The study of text index construction for large-scale dynamic collection [MS. Thesis]. Harbin: Harbin Institute of Technology, 2008 (in Chinese with English abstract).

[33] Kaur S, Kaur U. A survey on various clustering techniques with *K*-means clustering algorithm in detail. Int'l Journal of Computer Science & Mobile Computing, 2013,2(4):155−159.

[34] Lv Z, *et al*. Parallel *k*-means clustering of remote sensing images based on MapReduce. In: Proc. of the Web Information Systems and Mining. 2010. 162−170.

[35] Surve AR, Paddune NS. A survey on hadoop assisted *K*-means clustering of hefty volume images. Int'l Journal on Computer Science & Engineering, 2014,6(3):113−117.

[36] Singhal A. Modern information retrieval: A brief overview. Bulletin of the IEEE Computer Society Technical Committee on Data Engineering, 2009,24(24):35−43.

[37] Dongen BV, Dijkman R, Mendling J. Measuring similarity between business process models. In: Proc. of the 20th Int'l Conf. on Advanced Information Systems Engineering. 2008. 405−420.

[38] Wynn MT, Verbeek HMW, Aalst WMPVD, *et al*. Reduction rules for YAWL workflows with cancellation regions and OR-joins. Information & Software Technology, 2009,51(6):1010−1020.

[39] Mendling J, Reijers HA, van der Aalst WMPVD. Seven process modeling guide-lines (7PMG). Information & Software Technology, 2010,52(2):127−136.

[40] van Dongen BF, de Medeiros AKA, Verbeek HMW, *et al*. The ProM framework: A new era in process mining tool support. Lecture Notes in Computer Science, 2005,3536:444−454.

附中文参考文献:

[2] 柴学智,曹健.面向云计算的工作流技术.小型微型计算机系统,2012,33(1):90−95.

[13] 曹斌,尹建伟,陈慧蕊.基于 Levenshtein 距离的流程检索方法.计算机集成制造系统,2012,18(8):1766−1773.

[25] 王斌,张刚,孙健.大规模分布式并行信息检索技术.信息技术快报,2005,(2):1−9.

[26] 彭达.并行信息检索及其相关技术.中国教育科研与探索, 2007,(1):84−86.

[27] 陈国良,孙广中,徐云,等.并行算法研究方法学.计算机学报,2008,31(9):1493−1502.

[28] 刘义,陈荦,景宁,等.海量空间数据的并行 Top-*k* 连接查询.计算机研究与发展,2011,48(S3):163−172.

[29] 吴超.信息检索中 top-*k* 问题的并行算法及优化研究[博士学位论文].合肥:中国科学技术大学,2011.

[30] 齐鸣.共享内存并行系统上空间数据检索及优化研究[博士学位论文].合肥:中国科学技术大学,2012.

[31] 朱江.大规模分布式并行信息检索技术.计算机光盘软件与应用,2013,(21):305−306.

[32] 葛付江.面向动态文档集的大规模文本索引构建技术的研究[硕士学位论文].哈尔滨:哈尔滨工业大学,2008.

黄华(1981－),男,湖南衡阳人,博士,副教授,CCF 专业会员,主要研究领域为服务计算,业务过程管理.

冯在文(1980－),男,博士,讲师,CCF 专业会员,主要研究领域为业务过程管理.

彭蓉(1975－),女,博士,教授,博士生导师,CCF 专业会员,主要研究领域为需求工程,软件工程,服务计算.