一种分布式网络中轮廓推荐的有效方法

黄震华*①② 张 波③
①(同济大学电子与信息工程学院 上海 201804)
②(同济大学嵌入式系统与服务计算国家教育部重点实验室 上海 201804)
③(上海师范大学信息与机电工程学院 上海 200234)

摘 要：当底层数据的容量以及轮廓推荐指令个数增大时，轮廓推荐的时间代价将呈指数级增长，从而严重影响其推荐效率。为此，基于超对等分布式网络(SPA)，该文提出预存储 w 个轮廓快照来高效处理系统中 u 个轮廓推荐指令的分布式网络轮廓推荐算法(EMSRDN)。EMSRDN 算法充分考虑 SPA 网络的数据存储和通信特性，利用 map/reduce 分布式计算模型，通过初始快照集启发式构造来快速产生最优 w 个轮廓快照。理论分析和仿真实验表明，该算法具有有效性和实用性。

关键词：分布式网络；轮廓推荐；Map/reduce 分布式计算；信息服务
中图分类号： TP393 **文献标识码**： A **文章编号**：1009-5896(2015)05-1214-06
DOI: 10.11999/JEIT140615

An Efficient Method for Skyline Recommendation in Distributed Networks

Huang Zhen-hua①② Zhang Bo③
①(*School of Electronics and Information*, *Tongji University*, *Shanghai* 201804, *China*)
②(*Key Laboratory of Embedded System and Service Computing*, *Tongji University*, *Shanghai* 201804, *China*)
③(*College of Information, Mechanical and Electrical Engineering*, *Shanghai Normal University*, *Shanghai* 200234, *China*)

Abstract: Based on distributed networks of the Super-Peer Architecture (SPA), this paper proposes Efficient Method for Skyline Recommendation in Distributed Networks (EMSRDN), to handle u skyline recommendation instructions by prestore w skyline snapshots. The EMSRDN method fully considers the characteristic of storage and communication of SPA networks, and uses the map/reduce distributed computation model. The EMSRDN algorithm can fast produce the optimal w skyline snapshots through the phase of heuristically constructing the initial set of snapshot. The detailed theoretical analyses and extensive experiments demonstrate that the proposed EMSRDN algorithm is both efficient and practical.

Key words: Distributed networks; Skyline recommendation; Map/reduce distributed computation; Information service

1 引言

轮廓推荐技术是近年来信息服务方向的一个研究重点和热点[1]，这主要是因为它在许多领域有着广泛的应用，如：大数据分析、城市导航系统、多标准决策支持以及高维数据可视化等[2]。给定对象集合 $\boldsymbol{\mathcal{R}}=\{o_1,o_2,\cdots,o_n\}$，其中每个对象 $o_i(i\in[0,n])$具有 δ 个维度 $F=\{d_1,d_2,\cdots,d_\delta\}$，每个维度衡量它的一个子特征(比如距离，价格等)，那么维度空间 $U\subseteq F$ 上的轮廓推荐就是在 $\boldsymbol{\mathcal{R}}$ 中找出一类对象集合 $\boldsymbol{\mathcal{R}}'$，它满足如下条件：$\boldsymbol{\mathcal{R}}'\subseteq\boldsymbol{\mathcal{R}}$ 且 $\boldsymbol{\mathcal{R}}'$ 中每个对象不会在 U 所有维度上的取值均差于 $\boldsymbol{\mathcal{R}}$ 中的某一对象。显然，通过轮廓推荐，用户只需考虑属于轮廓对象集合的对象，而不必关心那些被过滤掉的对象，这样用户就可以在小规模的轮廓对象集合上对自己感兴趣的对象进行选择。不难看出，δ 个维度的对象集合，最多拥有 $2^\delta-1$ 个维度空间上的轮廓推荐。

近些年，国内外学者对轮廓推荐技术进行了深入的研究，并取得了一定的成果。文献[3]首次提出轮廓推荐的概念，并提出两个可行的推荐算法：块嵌套循环(Block Nested Loop, BNL)算法以及分区覆盖(Divide and Conquer, DC)算法。假定对象全集为 $\boldsymbol{\mathcal{R}}$，其中 BNL 算法通过 $O(|\boldsymbol{\mathcal{R}}|^2)$次对象间的比

2014-05-12 收到，2015-01-12 改回
国家自然科学基金(61272268, 61103069)，教育部新世纪优秀人才支持计划(NCET-12-0413)，国家 973 计划项目(2014CB340404)，霍英东教育基金会高等院校青年教师基金(142002)和同济大学中央高校基本科研业务费专项资金资助课题
*通信作者：黄震华 huangzhenhua@tongji.edu.cn

较来找出返回完整的轮廓对象集合，而 DC 算法使用递归分区的方法来获取推荐结果。文献[4]在 BNL 算法的基础上提出排序过滤轮廓(Sort Filter Skyline, SFS)算法，先对对象进行排序，再进行比较的推荐。SFS 算法能够有效减少 BNL 算法中对象间比较的次数，然而，它增加了排序的时间开销。文献[4]从理论上给出在均匀分布情况下，BNL 算法、DC 算法以及 SFS 算法的时间开销，并提出轮廓线性消除排序法(Linear Elimination Sort for Skyline, LESS)。LESS 算法的时间复杂度可降为 $O(|\boldsymbol{\mathcal{R}}|\log_2|\boldsymbol{\mathcal{R}}|+\delta|\boldsymbol{\mathcal{R}}|)$，其中 δ 为对象的维度个数。文献[5]将轮廓推荐与聚类算法相结合，来获取轮廓对象集合中最具代表性的 k 个对象，从而进一步精炼用户可选择对象的范围。文献[6]基于可能世界实例模型[7]给出两个处理不确定数据上轮廓推荐的有效方法：自底向上法(Bottom-Up Algorithm, BUM)和自顶向下法(Top-Down Algorithm, TDM)，其中 BUM 算法基于 R-树索引结构通过定界(bounding)、剪枝(pruning)和提纯(refining) 3 个阶段来快速返回概率超过预定阈值 p 的所有不确定数据上的轮廓对象；而 TDM 算法将所有不确定数据对象组织为 1 棵分区树，并利用分区树 3 个有效的性质来降低对象各可能世界实例间的比较次数，从而加快返回轮廓对象的速度。

随着分布式网络的深入应用，文献[8]首次考虑在超对等架构(Super Peer Architecture, SPA)[9]的分布式网络中实施轮廓推荐技术，并通过传输扩展轮廓对象集合来降低数据传输的代价。SPA 架构是目前使用较广的一种分布式网络，因为传统客户/服务(Client/Server, C/S)模型的网络架构能够方便地升级为 SPA 架构的分布式网络。文献[10]在文献[9]的基础上，增加了多维路由索引机制(Multidimensional Routing Indices, MRI)来降低参与轮廓推荐的网络节点数量，从而进一步降低数据传输的代价。文献[11]在识别文献[8,9]性能缺陷的基础上，提出了有效预处理分布式网络中子空间轮廓推荐的有效方法(Efficient Preprocessing of Subspace Skyline Queries in Distributed Networks, EPSSQDN)。EPSSQDN 算法基于布隆过滤器(Bloom Filter, BF)技术[11]来缩减数据传输代价；同时基于正规格结构[12]来索引网络节点上的数据对象，并且使用格间的支配关系来有效降低数据对象间的比较次数，从而提高轮廓推荐的计算效率。文献[13]基于非共享策略，围绕降低网络反应延迟与通信负荷的目标，提出了一种两阶段分布式算法(Two-Phase Distributed Algorithm, TPDA)，并对算法的关键实现环节，如协调-远程节点间的通信、轮廓推荐增量的计算等进行优化，使算法在通信负荷与反应延迟上达到较好的综合性能。

随着大容量廉价磁盘的出现，使得在 SPA 分布式网络采用预存储 w 个轮廓快照 $SN=\{s_1,s_2,\cdots,s_w\}$ 来高效处理系统中 u 个轮廓推荐指令 $IN=\{I_1,I_2,\cdots,I_u\}$，成为克服现有方法性能缺陷的首选技术。基于此，本文从优化 SPA 分布式网络中 u 个轮廓推荐指令在 w 个轮廓快照间的分配出发，提出了有效的分布式网络轮廓推荐算法(Efficient Method for Skyline Recommendation in Distributed Networks, EMSRDN)。EMSRDN 算法利用 map/reduce 分布式计算模型，通过初始轮廓快照集启发式构造来快速产生最优的 w 个轮廓快照。理论分析和仿真实验表明，本文所提的 EMSRDN 方法具有有效性和实用性。

2　问题描述

不失一般性，本文假定SPA分布式网络中的计算节点为N_c，并拥有λ个存储节点$N_g^{(1)},N_g^{(2)},\cdots,N_g^{(\lambda)}$，其中$u$个轮廓推荐指令$IN=\{I_1,I_2,\cdots,I_u\}$，在$N_c$上提交，而候选的$\gamma(\gamma>w)$个轮廓快照分布式存储于$N_g^{(1)},N_g^{(2)},\cdots,N_g^{(\lambda)}$上。首先，本文给出SPA分布式网络中轮廓推荐的代价模型。从轮廓快照 s 获取轮廓推荐指令 I 结果的时间代价 t_{I}^{s}，由两部分组成：(1)将s从磁盘调入内存的时间开销t_s，(2)由 s 计算产生 I 所对应结果的CPU代价$t_{s\to I}$。第(1)部分时间代价t_s比较容易获得，如式(1)所示：

$$t_{\mathrm{s}}=\frac{\mathrm{size(s)}}{\mathrm{block\_size}}\cdot t_{\mathrm{I/O}}^{\mathrm{block}} \tag{1}$$

其中 size(s) 为轮廓快照 s 的大小，block_size 为内存块大小，而 $t_{\mathrm{I/O}}^{\mathrm{block}}$ 为传输一个内存块的时间开销。

下面，给出第(2)部分的时间代价 $t_{s\to I}$。不失一般性，假定轮廓推荐指令 I 的维度空间为 V，并令$v=|V|$。

定理 1[14]　假定轮廓快照 s 在维度空间 V 上满足联合分布函数 $F(\bar{x})$和联合密度函数 $f(\bar{x})$，其中$\bar{x}=(x_1,x_2,\cdots,x_v)$，那么 I 结果的期望值 $E(s,v)$可表示为

$$|s|\times\int_{[0,1]^v}f(\bar{x})(1-F(\bar{x}))^{|s|-1}\mathrm{d}\bar{x} \tag{2}$$

定理 2[14]　假定轮廓快照 s 在维度空间 V 上满足联合分布函数 $F(\bar{x})$和联合密度函数 $f(\bar{x})$，其中$\bar{x}=(x_1,x_2,\cdots,x_v)$，那么第(2)部分的时间代价 $t_{s\to I}$ 可表示为

$$\sum_{x=2}^{|s|}E(x-1,v)\times E(x-1,v+1)/x-1 \tag{3}$$

本文所要解决的问题就是，如何从候选的γ个轮廓快照中挑选并预存储最优的$w(\gamma > w)$个轮廓快照SN=$\{s_1,s_2,\cdots,s_w\}$，使得计算代价comCost(IN)最小。

3 EMSRDN算法

本文发现，从候选的γ个轮廓快照中精确挑选最优的$w(\gamma > w)$个轮廓快照，需要遍历指数级个数的轮廓快照组合空间，从而使得获取精确最优w个轮廓快照是非多项式困难(NP-hard)问题。因此在本节中，将提出一种快速获取近似最优方案ASN的有效算法EMSRDN。

EMSRDN算法的核心思想是利用map/reduce分布式计算模型，通过初始轮廓快照集启发式构造以及基于遗传算法的轮廓快照集深度优化这两个阶段来快速产生最优的w个轮廓快照，其伪代码如表1所示。

map函数和reduce函数具体实施过程如表2和表4所示。

OPTIMIZATION的具体实施过程如表3。

4 实验评估

这一节通过具体的实验来评估本文EMSRDN算法的优化率和运行时间。

表1 EMSRDN算法

算法1 EMSRDN

输入 候选γ个轮廓快照$\mathrm{SN}'=\{\mathrm{sn}_1,\mathrm{sn}_2,\cdots,\mathrm{sn}_\gamma\}$，$u$个轮廓推荐指令$\mathrm{IN}=\{\mathrm{I}_1,\mathrm{I}_2,\cdots,\mathrm{I}_u\}$。

Begin

(1)构造SN′所对应的输入键值对集合

KY_SN=$\{<$'sn'+i, $\mathrm{sn}_i>|\ i\in[1,\gamma]\}$

(2)构造IN所对应的输入键值对集合

KY_I= $\{<$'I'+j,$I_j>\ |j\in[1,\ u]\}$

(3)将KY_SN分割成m份$\mathrm{KY_SN}_1,\mathrm{KY_SN}_2,\cdots,\mathrm{KY_SN}_m$;

(4)将KY_I分割成m份$\mathrm{KY_I}_1,\mathrm{KY_I}_2,\cdots,\mathrm{KY_I}_m$;

(5)For λ=1 to m Do /* m为用户参数 */

(6) $\mathrm{SI}_\lambda\leftarrow\mathrm{KY\_SN}_\lambda\cup\mathrm{KY\_I}_\lambda$;

(7) $\{<\mathrm{sn}_i, I>|\mathrm{sn}_i\in\mathrm{KY\_SN}_\lambda\wedge I\in\mathrm{KY\_I}_\lambda\}\leftarrow$**map**($\mathrm{SI}_\lambda$); /* sn_i为$\mathrm{KY_SN}_\lambda$中的每个轮廓快照，I为$\mathrm{KY_I}_\lambda$中能够通过sn_i来获取结果的指令 */

(8) $f(\mathrm{sn}_i)\leftarrow i \bmod n$; /*$f$为分割函数，$n$为执行reduce函数的工作计算机数量*/

(9) For λ=1 to n Do /*并行处理*/

(10) $\{<\mathrm{sn}_i, \mathrm{SI}_i>\}\leftarrow$**reduce**($\{<\mathrm{sn}_i, I>\}$); /*$\mathrm{sn}_i$为$\mathrm{KY_SN}_\lambda$中的每个轮廓快照，$\mathrm{SI}_i$为$\mathrm{KY_I}_\lambda$中通过$\mathrm{sn}_i$来获取结果的指令集合 */

(11) ASN$\leftarrow\{\mathrm{sn}_i|\mathrm{SI}_i\neq\varnothing\}$;

(12) Return ASN。

End

表2 map函数

算法2 map函数

输入 <轮廓快照标识符，轮廓快照实体>组成的键值对(key-value)输入集KY_SN, <轮廓推荐指令标识符，轮廓推荐指令实体>组成的key-value输入集KY_I。

输出 <轮廓快照实体，轮廓推荐指令实体>组成的中间key-value集合KY_SNI。

Begin

(1) KY_SNI←∅;

(2) mapCost←userCost/m;

(3) rootS←KY_I所包含所有轮廓推荐指令实体的根轮廓快照;

(4) IF mtCost({rootS})>mapCost Then Return Null; /*mtCost({rootS})为轮廓快照集{rootS}维护与传输代价之和，计算见式(1)*/

(5) Else

(6) For ∀<I_id, I_ent>∈KY_I Do

(7) KY_SNI←KY_SNI∪{<rootS, I_ent>};

(8) $\mathrm{KY\_SNI}^{(1)}\leftarrow$**OPTIMIZATION**(KY_SNI, KY_SN, KY_I); /*初始轮廓快照集启发式构造 */

(9) Return KY_SNI。

End

表3 OPTIMIZATION实施过程

算法 3 OPTIMIZATION(KY_SNI, KY_SN, KY_I)

输入 <轮廓快照实体，轮廓推荐指令实体>组成的集合KY_SNI，<轮廓快照标识符，轮廓快照实体>组成的key-value输入集KY_SN，<轮廓推荐指令标识符，轮廓推荐指令实体>组成的key-value输入集KY_I。

输出 <轮廓快照实体，轮廓推荐指令实体>组成的集合$\mathrm{KY_SNI}^{(1)}$。

Begin

(1) TempS←{rootS};

(2) For ∀<I_id, I_ent>∈KY_I Do

(3) $\mathrm{sn}\leftarrow\min\limits_{\mathrm{sn}\in\mathrm{SN}} t^{\mathrm{sn}}_{\mathrm{I\_ent}}$; /* $t^{\mathrm{sn}}_{\mathrm{I_ent}}$为从sn获取I_ent结果的时间代价，计算见式(1)-式(3) */

(4) KY_SNI←KY_SNI∪{<sn, I_ent>}−{<rootS, I_ent>};

(5) TempS←TempS∪{sn};

(6) $\mathrm{KY_SNI}^{(1)}\leftarrow$KY_SNI;

(7) Return $\mathrm{KY\_SNI}^{(1)}$。

End

4.1 实验环境设置

本文的实验环境由30台PC机组成3层SPA分布式网络架构，每台PC机的配置为4核i5-3450 CPU, 4 G内存和500 G硬盘，操作系统为CentOS Linux 6.4。

表4　reduce函数

算法4　reduce函数

输入　<轮廓快照实体，轮廓推荐指令实体>组成的中间key-value集合KY_SNI。

输出　<轮廓快照实体，轮廓推荐指令实体集合>组成的key-value集合KY_S。

Begin

(1)　SN←KY_SNI所包含轮廓快照实体组成的集合；

(2)　For $\forall sn \in SN$ Do $I^{(sn)} \leftarrow \varnothing$;

(3)　For $\forall <sn, I> \in KY\_SNI$ Do $I^{(sn)} \leftarrow I^{(sn)} \cup \{I\}$;

(4)　KY_S←∅;

(5)　For $\forall sn \in SN$ Do $KY\_S \leftarrow KY\_S \cup \{<sn, I^{(sn)}>\}$;

(6)　Return KY_S.

End

计算节点包含10台PC机组成的集群，其中1台PC机选为控制计算机(Master)，这10台PC机构成Hadoop平台，其版本号为1.0.3。而其余两层为20个分布式存储节点，共分配20台PC机。实验中，本文在计算节点上产生200个轮廓推荐指令，在每个存储节点上产生100个轮廓快照，总计2000个候选轮廓快照。

与EMSRDN比较的方法是OPTIMAL算法，通过指数级时间复杂度的穷举来获取精确最优的轮廓快照集合；每一类实验分为2组：(1)固定计算节点上轮廓推荐指令的个数为100，而每个存储节点上轮廓快照的个数在20～100间变化；(2)固定每个存储节点上轮廓快照的个数为50，而计算节点上轮廓推荐指令的个数在40～200间变化。所有算法的代码编译采用JDK 1.6。

4.2 EMSRDN算法的优化率评估

本小节通过实验评估EMSRDN算法的优化率。图1(a)和图1(b)分别给出这两个算法分别在两组实验中优化率的评估结果。

在图1评估算法优化率的实验中，本文以最优算法(OPTIMAL)为基准，因为它获取的轮廓快照集合是精确最优的，即将该精确轮廓快照集合的优化率定为100%。我们从图1可以看出，EMSRDN算法的优化率接近于最优算法。例如在图1(a)中，当每个存储节点上的轮廓快照数等于100时，EMSRDN算法的优化率为37.5%；在图1(b)中，当计算节点上轮廓推荐指令数等于80时，EMSRDN算法的优化率为68.3%。

4.3 EMSRDN算法的运行时间评估

本小节通过实验评估EMSRDN算法的运行时间。图2(a)和图2(b)分别给出这两个算法分别在两组实验中优化率的评估结果。

虽然在图1中，最优算法获取的轮廓快照集合的优化率略高于EMSRDN算法，然而在图2(a)和图2(b)评估算法运行时间的实验中，我们可以发现最优算法在每种实验环境下的运行时间都是非常巨大，这主要是因为最优算法为了获取精确的轮廓快照集合，需要遍历所有可能的轮廓快照组合空间，因此需要指数级的时间开销，而EMSRDN算法只需要多项式时间开销即可返回近似最优的轮廓快照集合，而无需遍历所有可能的轮廓快照组合空间。例如在图2(a)中，当每个存储节点上的轮廓快照数等于100时，最优算法的运行时间为79824.6 s，而算法EMSRDN的运行时间仅为35.9 s；在图2(b)中，当计算节点上轮廓推荐指令数等于200时，最优算法的运行时间为49652.5 s，而算法EMSRDN的运行时间仅为20.4 s。

因此，综合图1和图2的实验评估，可以得出本文的EMSRDN算法能够很好平衡轮廓快照集合的优化率与运行时间，而且具有很好的可扩展性。

4.4 SPA分布式网络中的轮廓推荐代价评估

目前常用的SPA分布式网络中轮廓推荐的算法有两个，即EPSSQDN[11]和TPDA[13]。因此，在这一小节中，通过实验来对比本文EMSRDN算法与EPSSQDN, TPDA算法的时间代价。基础数据库由文献[3]的数据生成器产生，包含1×10^6个对象，每个对象具有5个浮点型属性。轮廓指令个数固定为100。实验分为两组：(1)固定每个存储节点上轮廓快照的个数为80，底层数据库的对象个数在2×10^5～1×10^6变化；(2)固定底层数据库的对象个数为8×10^5，而每个存储节点上轮廓快照的个数在20～100间变化。在实验中，EPSSQDN和TPDA算法的轮廓推荐时间为直接从底层数据库中获取轮廓推荐结果的时间开销，而EMSRDN算法的轮廓推荐时间由两部分组成，一部分是由EMSRDN算法获取最优轮廓快照集合的时间开销，另一部分是由轮廓快照集合获取轮廓推荐结果的时间开销。图3(a)和图3(b)分别给出这3个算法分别在两组实验中轮廓推荐时间代价的评估结果。

从图 3 可以看出，本文 EMSRDN 算法在每一种实验环境下的轮廓推荐时间开销均小于目前常用的 EPSSQDN 和 TPDA 算法，这主要是因为轮廓推荐过程是 CPU 和 I/O 敏感的，算法输入的数据量大小直接影响轮廓推荐的时间开销，EMSRDN 算法虽然比EPSSQDN和TPDA算法多花费了轮廓快照的选择时间，然而相对于轮廓计算的时间，这部分开销所占的比重较小，因此 EMSRDN 算法比 EPSSQDN 和 TPDA 算法效率更高。例如在图 3(a)中，当底层数据库的对象个数为 1×10^6 时，EMSRDN

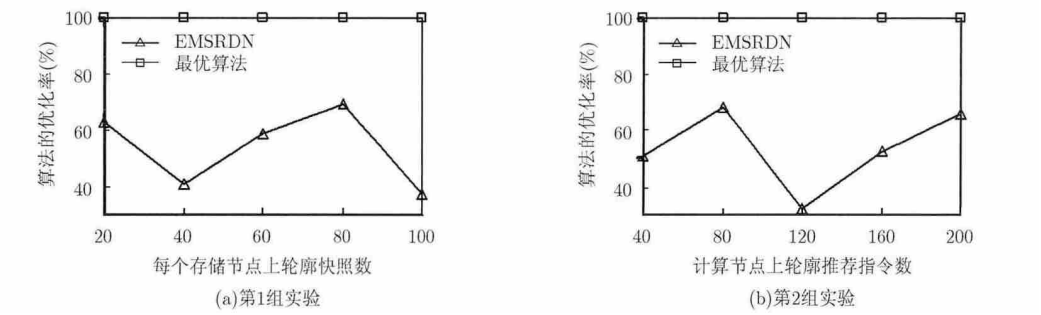

图 1 算法优化率实验评估

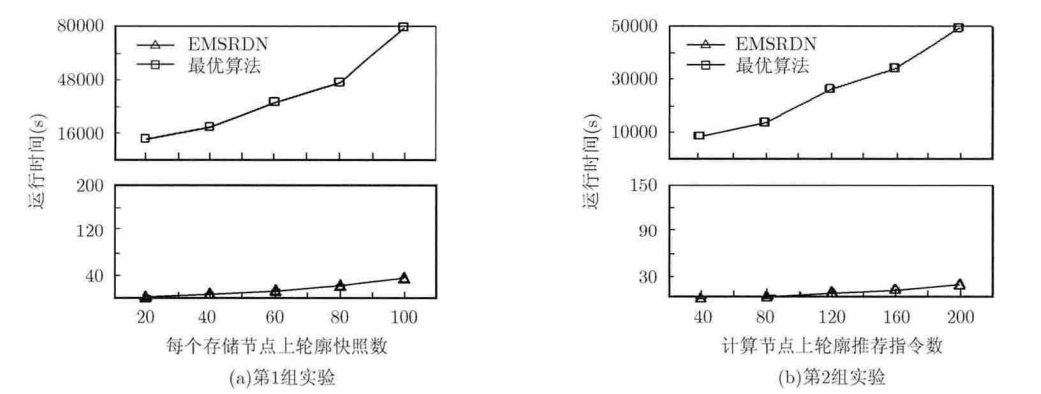

图 2 算法运行时间实验评估

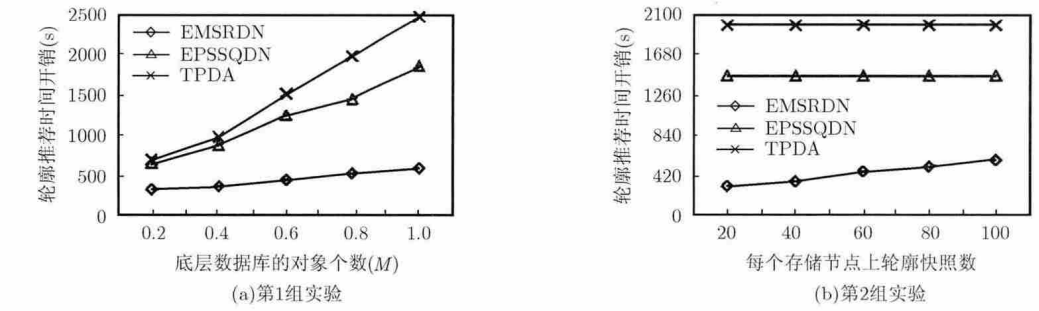

图 3 轮廓推荐时间代价实验评估

算法的的轮廓推荐时间为 604.5 s，而 EPSSQDN 和 TPDA 算法则分别需要 1868.5 s 和 2480.2 s；在图 3(b)中，当每个存储节点上轮廓快照个数为 20 时，EMSRDN 算法的的轮廓推荐时间为 304.5 s，而 EPSSQDN 和 TPDA 算法则分别需要 1462.1 s 和 1994.8 s。另一方面，我们在图 3(b)中还可以发现，对于每个实验环境EPSSQDN和TPDA算法的轮廓推荐时间都是相同的，这主要是因为这两个算法的输入是底层的基础数据，与轮廓快照的个数无关。

5 结束语

传统C/S架构的网络能够方便地升级到SPA体系架构的分布式网络，因此研究“在 SPA 架构的分布式网络中有效进行轮廓推荐”是一个很有意义的工作。本文分析了现有工作存在的主要性能缺陷，并给出一种在 SPA 分布式网络中，进行轮廓推荐的有效方法 EMSRDN。EMSRDN 算法不以底层细粒度的数据为输入参数，而采用预存储 w 个轮廓快照来高效处理系统中的 u 个轮廓推荐指令，并且利用 map/reduce 分布式计算模型，通过初始轮廓快照集启发式构造来快速产生最优的 w 个轮廓快照。理论分析和仿真实验表明，本文所提的 EMSRDN 方法具有有效性和实用性。

参 考 文 献

[1] Ma L and Zhu M. Skyline query for location-based recommendation in mobile application[C]. Proceedings of the International Workshops on Web-Age Information Management, Beidaihe, China, 2013: 236–247.

[2] Wu J, Chen L, Xie Y, *et al*.. Modelling and exploring

historical records to facilitate service composition[J]. *International Journal of Web and Grid Services*, 2014, 10(1): 54–79.

[3] Borzsony S, Kossmann D, and Stocker K. The skyline operator[C]. Proceedings of the 17th International Conference on Data Engineering, Heidelberg, Germany, 2001: 271–285.

[4] Godfrey P. Skyline cardinality for relational processing[C]. Preceedings of the International Symposium on Foundations of Information and Knowledge Systems, Wilheminenburg Castle, Austria, 2004: 78–97.

[5] Huang Z, Xiang Y, Zhang B, *et al*.. A clustering based approach for skyline diversity[J]. *Expert Systems with Applications*, 2011, 38(7): 7984–7993.

[6] Pei J, Jiang B, Lin X, *et al*.. Probabilistic skylines on uncertain data[C]. Proceedings of the 33rd International Conference on Very Large Data Bases, Vienna, Austria, 2007: 15–26.

[7] Fitting M. Possible world semantics for first-order logic of proofs[J]. *Annals of Pure and Applied Logic*, 2014, 165(1): 225–240.

[8] Vlachou A, Doulkeridis C, Kotidis Y, *et al*.. SKYPEER: efficient subspace skyline computation over distributed data[C]. Proceedings of the 23rd International Conference on Data Engineering, Istanbul, Turkey, 2007: 416–425.

[9] Ghafarian T, Deldari H, Javadi B, *et al*.. CycloidGrid: a proximity-aware P2P-based resource discovery architecture in volunteer computing systems[J]. *Future Generation Computer Systems*, 2013, 29(6): 1583–1595.

[10] Doulkeridis C, Vlachou A, Nørvåg K, *et al*.. Multidimensional routing indices for efficient distributed query processing[C]. Proceedings of the 18th ACM Conference on Information and Knowledge Management, Hong Kong, China, 2009: 1489–1492.

[11] 黄震华, 向阳, 孙圣力, 等. 超对等网络中的轮廓查询优化[J]. 电子学报, 2013, 41(8): 1515–1520.
Huang Zhen-hua, Xiang Yang, Sun Sheng-li, *et al*.. Optimizing skyline queries in SPA distributed networks[J]. *Acta Electronica Sinica*, 2013, 41(8): 1515–1520.

[12] Zhao L, Yang Y Y, and Zhou X. Continuous probabilistic subspace skyline query processing using grid projections[J]. *Journal of Computer Science and Technology*, 2014, 29(2): 332–344.

[13] Trimponias G, Bartolini I, Papadias D, *et al*.. Skyline processing on distributed vertical decompositions[J]. *IEEE Transactions on Knowledge and Data Engineering*, 2013, 25(4): 850–862.

[14] Chaudhuri S, Dalvi N N, and Kaushik R. Robust cardinality and cost estimation for skyline operator[C]. Proceedings of the 22th International Conference on Data Engineering, Atlanta, USA, 2006: 1–10.

黄震华：　男，1980 年生，博士，副教授，研究方向为信息服务、数据挖掘和大数据分析等.

张　波：　男，1978 年生，博士，副教授，研究方向为信息论、语义计算和模式识别等.