CN43-1258/TP
ISSN 1007-130X

计算机工程与科学
COMPUTER ENGINEERING & SCIENCE

2012 年第 34 卷第 2 期
Vol. 34, No. 2, 2012

文章编号:1007-130X(2012)02-0104-07

SubCounter:一种基于语义簇聚的节点子集规模估计方法*

SubCounter: A Node Subset Size Estimation Approach Based on Semantic Clustering

郑　重,王意洁,马行空

ZHENG Zhong, WANG Yi-jie, MA Xing-kong

(并行与分布处理国防科技重点实验室,湖南 长沙 410073)

(National Laboratory for Parallel and Distributed Processing, Changsha 410073, China)

摘　要:为进一步改进性能,很多 P2P 应用需要系统中各节点子集的规模信息。已有的节点子集规模估计方法主要基于对已有的系统节点规模估计方法的直接应用。本文提出了 SubCounter——一种基于语义簇聚的节点子集规模估计方法。SunCounter 通过节点间周期性的邻居交换为每个节点维护一个语义簇聚的邻居列表,以保持每个节点与自己所属各子集其他节点的联系。基于这种联系,SubCounter 以反熵聚集的方式实现节点子集规模估计。实验结果表明,相比于已有方法,SubCounter 在节点所属子集较多的情况下具有更快的收敛速度,并且能够以更小的通讯与存储开销保证同样的估计精度和相似的鲁棒性。

Abstract: Many P2P applications need the size values of node subsets in the system to enhance performance. The existing subset size estimation approaches are based on applying the size estimation approach directly. This paper proposes SubCounter, a node subset size estimation approach based on semantic clustering. SubCounter maintains a semantic clustering neighbor list for each node by view exchange, so each node can keep contacts with others in the same subset. Based on the semantic clustering, SubCounter realizes the estimation of subset sizes, through anti-entropy aggregation. The experimental results show that compared with the existing approaches, SubCounter converges more quickly when each node belongs to many subsets simultaneously, and ensures the same precision and similar robustness with less communication and storage cost.

关键词:网络规模;规模估计;反熵聚集;P2P

Key words: network size; size estimation; anti-entropy aggregation; P2P

doi: 10.3969/j.issn.1007-130X.2012.02.020

中图分类号:TP393　　**文献标识码**:A

1　引言

随着 Internet 的迅速发展,各种 P2P 系统得到了广泛的应用。P2P 系统为提供高效的服务常常需要节点规模等系统全局信息,但 P2P 系统本身的分布特性又为收集维护系统全局信息带来了额外的困难。

* **收稿日期**:2010-12-01;**修订日期**:2011-02-25
基金项目:国家 973 计划资助项目(2011CB302601);国家自然科学基金资助项目(60873215);湖南省自然科学杰出青年基金项目(S2010J5050);高等学校博士学科点专项科研基金资助课题(200899980003)
通讯地址:410073 湖南省长沙市开福区德雅路 109 号并行与分布处理国防科技重点实验室
Address: National Laboratory for Parallel and Distributed Processing, 109 Deya Rd, Kaifu District, Changsha, Hunan 410073, P. R. China

针对 P2P 系统中的节点规模估计问题已有不少研究[1~10],这些研究关注的是如何高效地估计整个系统的节点规模。然而,对有的 P2P 应用而言,只掌握系统节点规模信息是不够的,为了达到更好的性能,系统中的节点可能还需要具有某种特性的节点子集的规模信息。对于 P2P 数据共享系统,如果每个节点掌握具有某类数据的节点规模信息,那么就可以利用该信息优化数据查询路由表以及查询跳步数的上限值。对于 P2P 数据分发系统,具有某种兴趣的节点规模信息同样可以帮助优化各种数据分发协议。这类需求的满足依赖能够高效地对系统中特定节点子集规模进行估计的分布式方法。尽管也可以利用节点规模估计方法来估计子集规模,但那需要为每个节点子集维护一个覆盖网,因而引入了较大的维护开销以及可扩展性问题。

针对 P2P 系统中节点子集规模估计的问题,本文提出了 SubCounter——一种基于语义簇聚的节点子集规模估计方法。SubCounter 根据节点的语义信息标记节点所属子集,但不限定语义的具体含义。无论是以共享的数据类型还是以关注的兴趣主题作为语义,对于 SubCounter 并没有本质区别,通称为主题(Topic)。本文对节点子集的定义即是具有某一主题的所有节点组成的集合,而同一节点可能同时属于多个子集。在 SubCounter 中,每个节点通过随机的邻居交换来为自己的每个主题维护一定数目的邻居,即语义簇聚。每个节点为自己的每个主题维护一个子集规模估计值,并通过与具有共同主题的邻居进行反熵聚集来逐步精化自己的估计值。

SubCounter 通过在维护邻居关系时附加语义信息来实现语义簇聚。这种簇聚使得每个节点只需维护一个固定大小的邻居列表便可保持与自己所属各子集中其他节点的联系。由于具有某个共同主题的两个节点往往还拥有其他的共同主题,因此节点与自己邻居的一次交互往往能够同时实现多个节点子集规模估计值的反熵聚集。因此,SubCounter 能够在获得反熵聚集的快速精确优点的同时,保证较少的存储、通讯开销以及较好的可扩展性。

2　相关工作

针对整个 P2P 系统中节点规模估计的问题,已有不少方法被提出。根据其进行规模估计的基本方式,这些方法大致可以分为两类:基于反熵聚集的方法[1~4]和基于随机采样的方法[5~10]。

在基于反熵聚集的方法中,每个节点保存一个与节点规模有关的聚集数值,所有节点聚集数值的平均值与节点规模成确定的反比关系。每个节点周期性地随机选择网络邻居交换聚集数值,然后双方再用彼此聚集数值的平均值更新各自保存的数值。通过这样不断消除节点间差异的反熵聚集过程,系统中每个节点保存的数值逐步收敛于平均值,然后每个节点便可利用该值推算出系统节点规模。该类方法要求系统中每个节点参与,而每个节点也都能通过这个过程获取越来越精确的规模估计值。

在基于随机采样的方法中,每个需要规模估计值的节点通过随机行走、gossip、广播等方式对系统中的节点信息进行随机采样,然后再利用采样获取的信息结合各种概率方法推算系统节点规模。该类方法并不一定要求系统中每个节点参与,且通常只有规模估计发起节点能够最终获取相应的估计值。尽管该类方法通讯开销相对较低,但由于通过概率推算进行估计,因而其得出的节点规模估计值一般不如基于反熵聚集的方法精确。

已有的 P2P 系统中对节点子集规模进行估计一般直接使用上述节点规模估计方法,典型的如 TERA[11]。节点规模估计方法实际上就是对单个覆盖网中的所有节点数目进行估计,而要利用它们进行节点子集规模估计意味着必须将同一子集的节点组织到同一个覆盖网中。在 TERA 中,每个节点需要为自己的每个主题独立地维护一组邻居,具有相同主题的节点就这样被组织到同一个簇覆盖网中。通过在这些簇覆盖网上分别使用已有的节点规模估计方法,TERA 中的每个节点便能够获得自己所属各簇的规模估计值。由于每个节点必须维护与其主题数一样多的邻居列表,因而这种方法的存储与通讯开销较大。而且当每个节点具有的主题数不断增加时,该种方法还将面临严重的可扩展性问题。

SubCounter 根据节点间的语义关系为每个节点维护一个单一的邻居列表,在避免为每个节点子集维护单独的覆盖网的同时,还保证了每个节点与自己所属各子集其他节点的联系。在此基础上,即使节点具有的主题数增长,SubCounter 中的每个节点仍能够以较低的存储与通讯开销通过反熵聚集获取自己所属各子集的规模估计值。

3 SubCounter 描述

3.1 基本思想

SubCounter 包括两个组成部分:邻居维护算法和反熵聚集算法。两者都是分布式的,运行在系统中的每个节点上。邻居维护算法的目的是实现语义簇聚,即根据语义信息以 gossip 的方式为每个节点维护一个邻居列表,使得节点在其每个主题上都有一定数目的邻居。如此,每个节点便可以与其每个主题对应的节点子集中的其他节点保持足够多的联系,从而能够与同属一个子集的节点实施反熵聚集以估计子集规模。反熵聚集算法通过在属于同一子集的节点间均匀散布数值总和 1,以使每个节点能够利用其聚集数值的倒数获得其所属子集的规模估计值。在周期性的反熵聚集交互中,节点与其邻居将彼此所有共有主题对应的聚集数值取平均值作为各自的新值。由于具有某一相同主题的节点可能还同时具有其他共同主题,因此 SubCounter 能够在一次聚集交互中进行多个子集的规模估计,从而节约通讯开销。

3.2 邻居维护算法

SubCounter 的邻居维护算法是基于文献[12]中提出的 gossip 框架设计的。但是,在 SubCounter 中,节点的邻居列表中的条目都附加上了语义信息,其具体内容包括:对应节点的地址端口信息;对应节点具有的主题(该信息的存储可以用压缩的位向量实现以节省存储开销);该条目的年龄。每个节点周期性地执行邻居维护算法,与某个邻居交换彼此的部分邻居以更新自己的邻居列表。下面是邻居维护算法的伪代码。

算法 1 procedure maintainNeighbors

```
/ * active thread * /
1  while (1)
2    wait(T_n)
3    q ← getOldestEntry(view)
4    sortViewForSemanticClustering(view, c−l)
5    sendBuffer ← getHeadEntry(view, l−1)
6    selfEntry ← < selfAddress, selfTopics, 0>
7    sendBuffer ← sendBuffer ∪ {selfEntry}
8    sendTo(q.address, sendBuffer)
9    recvFrom(q.address, recvBuffer)
10   view ← view − {q}
11   replaceEntry(view, sendBuffer, recvBuffer)
12   increaseAge(view)
/ * passive thread * /
1  recvFrom(pAddress, recvBuffer)
2  sortViewForSemanticClustering(view, c−l)
3  sendBuffer ← getHeadEntry(view, l)
4  sendTo(pAddress, sendBuffer)
5  replaceEntry(view, sendBuffer, recvBuffer)
```

其中,T_n 是邻居维护算法执行周期,*view* 是由三元组 *entry*=<*address*, *topics*, *age*>组成的邻居列表,*c* 是 *view* 中规定的条目数,*l* 是邻居交换条目数,*selfEntry* 代表本地节点的三元组条目,*selfAddress* 是本地节点的地址,*selfTopics* 是本地节点的主题集合。

邻居维护算法由 *maintainNeighbors* 例程实现,包括主动与被动两种线程,前者对应邻居交换发起节点的动作,后者对应响应节点的动作。发起节点选择年龄最大的条目对应的节点与之交换各自的部分邻居条目,然后再用新得到的邻居条目填充空位及替换交换出去的邻居条目,最后再为每个邻居条目的年龄加 1。响应节点的动作过程大致类似。发起节点在交换邻居时会注销交换对象对应的条目,同时创建一个对应自己的年龄为 0 的条目发送给对方。通过在系统中注销较老条目及注入新条目,邻居维护算法能够在邻居交换更新的过程中逐步剔除那些失效节点。

邻居维护算法实现语义簇聚的关键在于交换邻居时保留何种邻居以及交换何种邻居,这一选择通过例程 *sortViewForSemanticClustering* 实现,其伪代码见算法 2。

算法 2 *procedure sortViewForSemanticClustering*(*view*, *m*)

```
1  while (|view| > 0 & |reserved| < m)
2    topic ← argmin_{t∈selfTopics}{|{r ∈ reserved : t ∈ r.topics & ∃ v (v ∈ view & t ∈ v.topics)}|}
3    selectedEntry ← argmin_{v∈{v∈view:topic∈v.topics}}{v.age}
4    view ← view−{selectedEntry}
5    reserved ← reserved ∪ { selectedEntry}
6  exchanged ← view
7  view ←Ø
8  将 exchanged 中条目依次置入 view 前部,然后再将 reserved 中条目依次置入 view
```

其中,*view* 是由三元组 *entry*=<*address*, *topics*, *age*>组成的邻居列表,*m* 是保留邻居条目的数目,即不用于交换的邻居条目数,*selfTopics* 是本地节点的主题集合,*reserved* 是保留邻居条目集合,*exchanged* 是交换邻居条目集合。

保留邻居的数目等于邻居总数减去交换邻居数,即 $c-l$。sortViewForSemanticClustering 选择保留邻居的原则就是保证本地节点在自己的每个主题上都有尽可能同样数目的保留邻居,这样节点在邻居交换时就不至于失去与某些同主题的邻居的联系,而且还可以通过邻居交换逐步为自己的每个主题都寻找到一定数目的邻居。sortViewForSemanticClustering 在确定保留邻居后,将它们放置在邻居列表后部,以便邻居维护算法选择列表前部的条目用于交换。

3.3　反熵聚集算法

为实现反熵聚集,每个节点为自己的每个主题维护一个聚集数值及对应的聚集标识。聚集数值的倒数即代表对该主题对应子集规模的估计;聚集标识则由时戳与节点标识组成,用以区分关于同一主题的不同的聚集过程。聚集标识中的时戳越大则该聚集标识越大,若时戳相同,节点标识越大则聚集标识越大;聚集标识越大表示相应聚集过程的优先级越高。聚集过程的初始化以及聚集标识具体如何设置将在 3.4 节中介绍。

新节点加入系统后首先利用随机行走为自己的每个主题寻找邻居,若没有找到,则将对应主题的聚集数值置为 1,并设置相应标识。每个节点都周期性地执行反熵聚集算法,以更新自己的聚集数值。通过反熵聚集过程,每个节点的每个主题的聚集数值逐步收敛于所有同主题节点的聚集数值的平均值,于是每个节点关于每个主题的子集规模估计值也逐步收敛于精确值。算法 3 是反熵聚集算法的伪代码。

算法 3　procedure antiEntropyAggregation

```
                    /* active thread */
1     while (1)
2     wait(T_a)
3     targets ← getRandomSubset (view, k)
4     for each q ∈ targets do
5       sendBuffer ← {a ∈ aggInfo : a.topic ∈ q.topics}
6       sendTo(q.address, sendBuffer)
7       recvFrom(q.address, recvBuffer)
8       for each r ∈ recvBuffer do
9         a ← getAggInfoEntryByTopic(aggInfo, r.topic)
10          if (a == null) then continue
11          if (a.identifier > r.identifier) then
12            r.value ← 0
13          if (a.identifier < r.identifier) then
14            a.value ← 0; a.identifier ← r.identifier
15          a.value ← (a.value + r.value) / 2
                    /* passive thread */
1     recvFrom(pAddress, recvBuffer)
2     sendBuffer ← {a ∈ aggInfo : ∃ r(r ∈ recvBuffer & a.topic == r.topic)}
3     for each s ∈ sendBuffer do
4       a ← getAggInfoEntryByTopic(aggInfo, s.topic)
5       r ← getAggInfoEntryByTopic(recvBuffer, s.topic)
6       if (s.identifier > r.identifier) then
7         r.value ← 0
8       if (s.identifier < r.identifier) then
9         a.value ← 0; a.identifier ← r.identifier
10    sendTo(pAddress, sendBuffer)
11    for each s ∈ sendBuffer do
12      a ← getAggInfoEntryByTopic(aggInfo, s.topic)
13      r ← getAggInfoEntryByTopic(recvBuffer, s.topic)
14      a.value ← (a.value + r.value) / 2
```

其中,T_a 为反熵聚集算法执行周期,k 是一个周期中与之进行聚集交互的邻居数,$view$ 是由三元组 $entry=<address, topics, age>$ 组成的邻居列表,$aggInfo$ 是由三元组 $entry=<topic, value, identifier>$ 组成的聚集条目列表,$selfTopics$ 是本地节点的主题集合。

与邻居维护算法类似,反熵聚集算法也包括主动与被动两种线程,分别对应一次交互的发起者与响应者的动作。每个节点在一个周期中依次与 k 个随机选取的邻居进行反熵聚集交互。显然,k 越大估计值收敛越快,但通讯开销也越大,因此其值的选取需要在收敛速度与通讯开销间作一权衡。每个节点与邻居交换聚集值时只选择两者共同主题对应的聚集数值,然后将对方的值与自己的对应值取平均作为各自的新值。但是,节点在计算平均值以更新自己的值之前,首先需要判断彼此数值对应的聚集标识的大小。较小的聚集标识对应低优先级的聚集过程,因此相应节点需要放弃原聚集数值而切换到对方的高优先级的聚集过程,即原聚集数值在参与相应数值的更新前需先清零。参与交互的两节点也需用在交互中获取的较大聚集标识来更新自己的对应标识。

3.4　动态适应性

SubCounter 的邻居维护算法没有专门的节点失

效处理机制,而是利用邻居条目的年龄信息周期性地更新邻居列表以自动剔除失效节点。当节点需要主动退出时,只需直接退出,其他节点直接将其作为失效节点看待。新加入的节点从系统中的某已知节点发起并发随机行走,然后将各随机行走终止节点作为自己的初始邻居,并将代表自己的条目加入到这些终止节点的邻居列表中。在完成邻居列表的初始化后,新加入节点可反复执行若干次邻居维护算法以迅速实现语义簇聚。之后,新加入节点即可进入周期性的邻居维护以及反熵聚集过程。

为了实现精确的节点子集规模估计,系统中每个主题对应的分布在各节点的聚集数值之和应始终保持为1。节点失效会使系统损失失效节点的主题对应的聚集数值,从而影响反熵聚集算法的准确性。因此,SubCounter 每隔若干反熵聚集算法执行周期即按概率重启各主题对应的聚集过程,即每个节点每隔若干周期即逐个按概率将其各个主题对应的聚集数值置1,并用当前时间及自己的节点标识生成新的聚集标识。每个节点重启其某个主题对应的聚集过程的概率设为该节点对该主题对应子集规模的历史估计值的倒数。SubCounter 的反熵聚集算法中关于聚集标识的特定处理步骤的作用就是解决由上述随机重启可能导致的多个关于同一子集的聚集过程同时并存的问题。

4 性能评价

4.1 实验设置

TERA[11]通过分簇直接利用已有的节点规模估计方法来估计节点子集规模是目前的典型做法。为了评价 SubCounter 的性能,我们用 P2P 协议模拟器 PeerSim[13]实现了 SubCounter 与 TERA 以进行实验对比,比较内容包括估计值的收敛速度、通讯开销、存储开销、鲁棒性。TERA 中每个节点为自己的每个主题维护一组邻居,其维护方法也是基于节点间的邻居交换;TERA 中使用文献[2]的基于反熵聚集的节点规模估计方法。SubCounter 与 TERA 中的邻居维护算法及反熵聚集算法均按同样的时间间隔周期性执行,因此我们使用周期轮次来衡量实验时间,而两者的聚集重启周期间隔均设为25个轮次。SubCounter 中每个节点单个周期内与之进行反熵聚集交互的邻居个数记为 k,在实验中可调。SubCounter 中节点的邻居列表大小以及 TERA 中节点的全局的及关于每个主题的邻居列表大小均设为20;两者中节点在一次邻居列表维护中与邻居交换的邻居数均设为10。实验中节点总数设为5 000,主题总数设为100,每个节点具有的主题数记为 t,每个主题根据参数为0.5的 Zipf 分布随机生成,即每个特定主题被选中的概率正比于 $i^{-\alpha}$(i 为主题序号,α 为 Zipf 参数)。

4.2 收敛速度

为了衡量节点子集规模估计值的精确性,我们定义平均相对差错率(Mean Relative Error,简称 MRE),即 $MRE=\frac{1}{I+1}\sum_{i=0}^{I}\frac{1}{N_i}\sum_{j=1}^{N_i}\left|\frac{\varphi_{ij}-N_i}{N_i}\right|$,其中 $i\in[0,I]$表示主题编号,i 为0表示对应系统中所有节点的虚拟主题,N_i 表示主题 i 对应节点子集的实际规模,φ_{ij} 表示主题 i 对应的节点子集中的第 j 个节点对该子集规模的估计值。平均相对差错率越小,意味着系统中节点对各子集规模的估计越精确。当平均相对差错率小于0.5%,系统中绝大多数节点的估计值的精度已足以满足多数应用的需求。因此,我们用平均相对差错率经过多少轮次后小于0.5%来衡量收敛速度,并用为使平均相对差错率小于0.5%每个节点平均所需的通讯消息数来衡量通讯开销,这里的消息包括邻居交换以及反熵聚集的消息。下面我们首先通过调整参数 k 来检验 SubCounter 的性能。

表1列出了 t 为10时 SubCounter 在不同 k 值下的收敛速度与通讯开销。随着 k 逐步增大,SubCounter 收敛所需轮次逐步减少,但通讯开销逐步增大。这表明为了实现最佳性能,SubCounter 需根据应用需求通过调整参数 k 在收敛速度与通讯开销间做一权衡。在后面的实验中,我们均令 k 值等于 t 值。

表1 SubCounter 在不同 k 值下的性能

k	2	4	6	8	10	12
收敛速度	66	36	25	19	16	14
通讯开销	390.0	350.0	336.0	324.0	330.1	338.1

表2列出了 SubCounter 与 TERA 在不同 t 值下的收敛速度。随着 t 值增大,TERA 的收敛速度始终保持相对稳定,SubCounter 的收敛速度则逐渐加快,并在 t 大于10时超过了 TERA 的收敛速度。TERA 收敛速度保持稳定是因为各个子集分别并行地进行反熵聚集,单个节点所属子集数目的增加对平均相对差错率的收敛过程并无显著影响。SubCounter 的收敛速度之所以随 t 增大而加快,则是因为随着每个节点的主题数增多,节点间

共有的主题数也增多,SubCounter 通过语义簇聚而能够将更多子集的聚集交互包装在一次节点交互中。以上结果表明,SubCounter 的收敛速度在每个节点主题数较多的情况下优于 TERA,并表现出了更好的可扩展性。

表 2　不同 t 值下的收敛速度

t	6	8	10	12	14
SubCounter	26	20	16	13	10
TERA	15	15	16	16	16

4.3　维护开销

这里的维护开销包括通讯与存储开销。通讯开销用为使平均相对差错率小于 0.5% 每个节点平均所需的邻居列表维护及反熵聚集消息总数来衡量。SubCounter 的邻居条目与 TERA 的类似,而其独有的节点主题信息可以用压缩的位向量来存储,所以在邻居列表不大的情况下可以忽略不计。因此,我们直接用每个节点维护的邻居条目总数来衡量存储开销。

表 3 列出了 SubCounter 与 TERA 在不同 t 值下的通讯开销。随着 t 值增大,TERA 的通讯开销逐步增大,而 SubCounter 的通讯开销则逐渐减少,且始终低于 TERA。TERA 通讯开销的增大源自每个节点必须维护更多邻居列表所带来的消息开销,而 SubCounter 通讯开销减少的原因则与 4.2 节中其收敛速度加快的原因一致,即利用语义簇聚在一次交互中实现多个子集的反熵聚集。以上结果表明,在保证同样精度的前提下,SubCounter 的通讯开销低于 TERA,并表现出了更好的可扩展性。

表 3　不同 t 值下的通讯开销

t	6	8	10	12	14
SubCounter	350.1	342.0	330.1	312.0	270.1
TERA	407.0	519.0	675.0	795.1	915.1

表 4 列出了 SubCounter 与 TERA 在不同 t 值下的存储开销。随着 t 值增大,TERA 的存储开销逐步增大,而 SubCounter 的存储开销则保持在 20,且始终低于 TERA。TERA 中的每个节点必须维护一个全局邻居列表,此外还要为自己的每个主题分别维护一个邻居列表,因此在节点具有的主题数增多的情况下,其存储开销不断增大。SubCounter 通过实现语义簇聚保证节点与自己所属各个子集中的邻居保持联系,因而每个节点只需要维护一个固定大小的邻居列表。以上结果表明,在存储开销这一指标上,SubCounter 显示出远优于 TERA 的可扩展性。

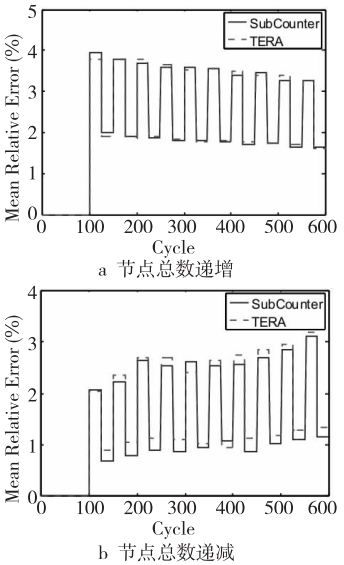
表 4　不同 t 值下的存储开销

t	6	8	10	12	14
SubCounter	20	20	20	20	20
TERA	140	180	220	240	280

4.4　鲁棒性

为衡量 SubCounter 及 TERA 的鲁棒性,我们考察它们在节点波动条件下提供的估计值的精确度。由于聚集过程周期性重启会导致节点聚集数值的周期性波动,为提供稳定可靠的估计值,两种方法均在每个聚集重启周期结束前根据节点当前的聚集数值计算更新相应的子集规模估计值。

图 1a 和图 1b 分别显示系统从第 100 轮开始每隔 50 个轮次增加及减少 100 个节点对 SubCounter 及 TERA 估计值的影响。



a 节点总数递增

b 节点总数递减

图 1　节点波动条件下的平均相对差错率

由图 1 可知,在节点总数周期性递增及递减的情况下,两种方法的平均相对差错率均产生了最大不超过 4% 的波动,且两种方法的曲线差别很小。图中平均相对差错率波动的周期性实际上源自子集规模估计值更新的周期性,也正是通过周期性地重启聚集过程及更新估计值,SubCounter 与 TERA 才能保证在动态的网络环境中使节点子集规模估计值及时反映实际情况。以上结果表明,在节点波动的条件下,SubCounter 提供的节点子集规模估计值的精确度与 TERA 的类似,即两种方法具有大致相当的鲁棒性。

5　结束语

本文提出了一种基于语义簇聚的节点子集规

模估计方法——SubCounter。SubCounter 基于节点邻居交换来实现语义簇聚,使得每个节点只需维护一个固定大小的邻居列表便可保持与自己所属各子集中其他节点的联系。在此基础上,SubCounter 利用反熵聚集来进行节点子集规模估计。实验结果表明,相对于已有方法,在每个节点同时属于较多子集的情况下,SubCounter 具有更快的收敛速度;在保证同样的估计精度和相似的鲁棒性的情况下,SubCounter 具有更小的通讯与存储开销。

参考文献:

[1] Montresor A, Jelasity M, Babaoglu O. Robust Aggregation Protocols for Large-Scale Overlay Networks[R]. Technical Report UBLCS-2003-16, Department of Computer Science, University of Bologna, 2003.

[2] Jelasity M, Montresor A. Epidemic-Style Proactive Aggregation in Large Overlay Networks[C]// Proc of ICDCS'04, 2004:102-111.

[3] Jelasity M, Kowalczyk W, van Steen M. An Approach to Massively Distributed Aggregate Computing on Peer-to-peer Network[C]// Proc of the 12th Euromicro Conference on Parallel Distributed and Network-Based Processing (PDP'04), 2004:200-207.

[4] Shafaat T M, Ghodsi A, Haridi S. A Practical Approach to Network Size Estimation for Structured Overlays[C]// Proc of the 3rd International Workshop on Self-Organizing Systems (IWSOS'08), 2008:71-83.

[5] Flajolet P, Martin G N. Probabilistic Counting[C]// Proc of the 24th Annual Symposium on Foundations of Computer Science (FOCS'83),1983:76-82.

[6] Przydatek B, Song D, Perrig A. Sia: Secure Information Aggregation in Sensor Networks[C]// Proc of the 1st Int'l Conf on Embedded Networked Sensor Systems, 2003:255-265.

[7] Kostoulas D, Psaltoulis D, Gupta I, et al. Decentralized Schemes for Size Estimation in Large and Dynamic Groups [C]// Proc of the 4th IEEE International Symposium on Network Computing and Applications (NCA'05), 2005:41-48.

[8] Massoulie L, Merrer E L, Kermarrec A, et al. Peer Counting and Sampling in Overlay Networks: Random Walk Methods[C]// Proc of the 25th Annual ACM Symposium on Principles of Distributed Computing, 2006:123-132.

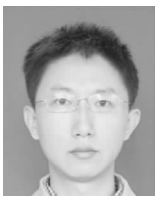
[9] Kennedy O, Koch C, Demers A. Dynamic Approaches to In-network Aggregation[C]// Proc of the 25th International Conference on Data Engineering (ICDE'09), 2009:1331-1334.

[10] Cardoso J C S, Baquero C, Almeida P S. Probabilistic Estimation of Network Size and Diameter[C]// Proc of the 4th Latin-American Symposium on Dependable Computing (LADC'09), 2009:33-40.

[11] Baldoni R, Beraldi R, Quema V, et al. Tera: Topic-Based Event Routing for Peer-to-Peer Architectures[C]// Proc of the 2007 Inaugural Int'l Conf on Distributed Event-Based Systems (DEBS'07), 2007:2-13.

[12] Jelasity M, Voulgaris S, Guerraoui R, et al. Gossip-Based Peer Sampling[J]. ACM Transactions on Computer Systems (TOCS), 2007, 25(3):8.

[13] PeerSim[CP/OL]. [2010-11-20]. http://peersim.sourceforge.net/.



郑重(1982-),男,江苏扬州人,博士生,CCF 会员(E200012085G),研究方向为网络计算和数据分发技术。**E-mail**:zhengzhong@nudt.edu.cn

ZHENG Zhong, born in 1982, PhD candidate, CCF member(E200012085G), his research interests include Internet computing, and data distribution technology.

王意洁(1971-),女,江苏镇江人,博士,教授,CCF 会员(E200006504S),研究方向为网络计算、海量数据管理和虚拟化技术。**E-mail**:wangyijie@nudt.edu.cn

WANG Yi-jie, born in 1971, PhD, professor, CCF member(E200006504S), her research interests include Internet computing, massive data management, and virtualization technology.

马行空(1987-),男,河南邓州人,博士生,CCF 会员(E200012091G),研究方向为网络计算和数据分发。**E-mail**:mxkong@gmail.com

MA Xing-kong, born in 1987, PhD candidate, CCF member(E200012091G), his research interests include Internet computing, and data distribution technology.