



电信大规模社交关系网络图数据挖掘研究

刘丽娇¹, 陶俊才¹, 肖晓军², 卢宇²

(1. 南昌大学信息工程学院计算中心 南昌 330029; 2. 广州优亿信息科技有限公司 广州 510630)

摘要: 电信技术的发展使得通话网络几乎覆盖了整个社会, 对电信运营商来说, 这个庞大的社会网络蕴藏着巨大的商机。尤其在日益激烈的市场竞争中, 客户关系的维护、客户潜在价值的挖掘以及有针对性地对客户进行营销等对电信图数据的挖掘, 受到了强烈关注。通过对比, 研究了当前针对类似电信网络的大规模社交关系网络图数据进行挖掘的分布式和单机图计算工具和框架, 并对单机图计算工具 Graphchi 的单机处理数据能力进行了可行性和可用性测试, 同时采用单机 Graphchi 对电信社交关系网络的数据进行了图挖掘研究和应用。

关键词: 电信图数据; 大规模图数据挖掘; Graphchi

doi: 10.11959/j.issn.1000-0801.2015035

Research on Large-Scale Social Network Graph Data Mining in Telecommunication

Liu Lijiao¹, Tao Juncail, Xiao Xiaojun², Lu Yu²

(1. Computing Center, Information Engineering College, Nanchang University, Nanchang 330029, China;

2. Guangzhou Usease Information Technology Co., Ltd., Guangzhou 510630, China)

Abstract: With the development of the telecommunications technology, telephone call network almost covered the whole society; the huge social network presents tremendous opportunities for telecommunication companies. Especially in the increasing fierce market competition, more and more telecommunication companies are concerned on maintaining customer relationship, mining customer potential value and marketing suitable target customers. The distribute and stand-alone graph computing frameworks and tools were researched for mining large-scale social network graph data like telecommunication network, and at the end the feasibility and usability of Graphchi in large-scale graph datasets were verified and a graph mining application was made with telecommunication data using stand-alone graph data processing tool—Graphchi.

Key words: telecommunication graph data, large-scale graph data mining, Graphchi

1 引言

对大规模数据特别是大规模图数据的分析和计算近年来获得了广泛关注。在电信领域, 现代通信设备的发展和普及使得电信数据不断扩增, 而如何挖掘并利用这些数据实现客户关系维系, 挖掘客户潜在价值以及为客户提供

个性化的服务项目, 在日益激烈的市场竞争环境中显得尤为重要。

电信网络数据可以映射成由顶点和边表达的抽象图数据结构, 从而形成社交关系网络。社交关系网络具有“小世界性”, 它可以反映网络的局部特征, 电信运营商通过分析每个客户群中成员的年龄、性别、职业、级别、爱好和兴

图1所示为抽象后的用户网络关系,利用图计算的PageRank算法可以分析出该网络中的关键人物,图中圆圈的大小表示对应的用户在关系网中的重要程度,边的粗细表示往来的频繁程度。

大规模图数据时代已然到来,而如何高效处理大规模图数据,成为一个新的挑战,设计简单可用的系统来分析处理现实世界中大规模的图已经成为当前面临的最迫切的问题。

主要有单机处理方式和并行分布式处理方式。本文介绍了几种大规模图数据运算的分布式工具和单机计算工具 Graphchi, 并且针对 Graphchi 的应用前景以及大数据图处理的可行性和可用性进行了实验和对比, 最后应用 Graphchi 对电信社交关系网络数据进行了挖掘实验。

2.1 分布式框架下的图计算工具

为了解决 MapReduce 在一些机器学习算法中性能瓶颈问题, Google 针对大规模图运算提出了 Pregel 框架, 它是严格的 BSP (bulk synchronous parallel) 模型 (BSP 模型, 即“大块”同步模型, 其概念由哈佛大学的 Valiant 和牛津大学的 Bill McColl 提出, 是一种异步 MIMD-DM 模型, 支持消息传递系统, 块内异步并行, 块间显式同步), 采用“计算-通信-同步”模式面向顶点的迭代方式完成机器学习的数据同步, 这种灵活的面向顶点的方法和高效的容错机制的设计模式可以描述一系列的算法, 并在有上千台的计算节点的集群中得以实现。

2015035-2

和批量的方式传递消息,通过共享内存的方式,有效地缓解了远程读取数据的延迟,提升了集群的性能,并且 Pregel 应用一组抽象的 API 隐藏了分布式编程的相关细节,展现给使用者一个易编程和易使用的大型图算法处理计算框架。

但是 Google 一直没有将 Pregel 的具体实现开源,外界对 Pregel 的模仿实现在性能和稳定性方面都未能达到工业级应用的标准。同时,在图计算中,由于图的顶点、边密度的不平衡性的特点,带来 BSP 模型的“木桶效应”(木桶效应是由美国管理学家彼得提出的,本文指的是先完成的任务需要等待后完成的任务,处理速度最慢的任务将成为整个系统的效率制约瓶颈)的限制,网络、计算机硬件中的差异性也会使这种现象更加明显。

2.1.2 Spark

Spark 是 UC Berkeley AMP 实验室开发的通用的并行计算框架,是 Pregel 的优化模型,它是基于 MapReduce 算法实现的分布式计算框架。Spark 拥有 MapReduce 所具有的优点,但不同于 MapReduce 的是,Spark 采用了一种弹性分布式数据集(resilient distributed dataset, RDD)的抽象数据结构,Spark 是一个基于内存计算的开源的集群计算系统。

RDD 是一个具有容错机制的特殊集合,它提供了一种抽象的数据架构,使用 RDD 逻辑转换而来的可重复使用的共享内存,而不再需要反复读写 HDFS,解决了 MapReduce 框架在迭代计算式中进行大量磁盘 I/O 操作的问题,这让数据分析更加快速,为构建低延迟的并行性大数据分析处理框架提供了稳定的基础。

同时,Spark 提供了 REPL(read-eval-print loop)的交互式查询以及函数式编程,支持围绕 RDD 抽象的 API,同时包括一套 transformation(转化)和 action(动作)操作以及针对大量流行编程语言的支持,比如 Scala、Java 和 Python。

在图计算方面,Spark 原生的 Bagel 以及 Graphx 提供了对于图操作的 API,为大规模的图计算提供了低延迟,负责优化交互式的大规模并行处理框架,但是 Spark 的磁盘索引是简单的静态机制,无法随着迭代状态的变化而动态优化。

2.1.3 Graphlab

Graphlab 是 CMU 的 Select 实验室提出的基于内存共享机制且面向机器学习的流处理并行框架,它的分布式处理是基于 MPI(message passing interface,消息传递接口)实现的,并且将数据抽象成图结构,它是以图的顶点为计算

单元的大规模图处理系统,支持稀疏的计算依赖异步迭代计算等,解决了 MapReduce 不适应需要频繁数据交换的迭代机器学习算法问题,是继 Google 的 Pregel 之后的第一个开源的大规模图处理系统。

Graphlab 的核心思想是“以图顶点的方式思考问题”,以最小化集群计算节点之间的通信量和均衡计算节点上的计算和存储资源为原则,对图的顶点进行切分。类似于 MapReduce 中的 map 和 reduce 过程,它将机器学习抽象成 GAS(gather(收集)、apply(运算)、scatter(更新))3个步骤,然后按该抽象模型设计顶点程序实现算法。在 gather 阶段,当前点收集邻接点和边的值,结合自身的值,进行简单的用户定义的 sum(求和)操作;在 apply 阶段,当前点根据 sum 得到的值及其前一时刻自身的值计算新的点值;scatter 阶段当前点利用自己的新值,结合邻接点/边前一时刻的值来计算邻接边的新值,并更新邻接边。

GraphLab 的算法被应用于很多推荐系统,也包括银行的欺诈检测和电脑网络中的入侵检测等领域。

2.1.4 PowerGraph

PowerGraph 是卡内基梅隆大学设计的一种强大的图计算分布式并行框架,它结合了 Graphlab 和 Pregel 关于图计算的优点,有效改善了 Pregel 和 Graphlab 等框架的并行化受限于顶点的邻居个数的问题。

现实世界中的图,都是典型的 Power-Law(幂律)分布图,其中少部分顶点连接到图中大部分的顶点上,这种图的划分对于并行的分布式框架来说是一个非常大的难题,并且图的划分效率直接影响系统的通信开销。一般的并行框架采用的是散列随机分配方案,但这种方案没有考虑局部性,划分完成后各任务负责的子图之间的强耦合性导致后续的迭代计算过程产生大量的消息通信,严重影响负载均衡。PowerGraph 使用了支持同步处理和异步处理机制的 GAS 模型,并且提出了一种 P-路顶点切割分区方案,在减少计算中通信量的同时保证了负载均衡,很好地解决了图的 Power-Law 问题。

2.2 单机图计算工具——Graphchi

除了以上介绍的分布式图计算框架外,还可以使用单机的图算法库,如 BGL/LEAD/NetworkX/JDSL/Stanford GraphBase/FGL 等进行图的挖掘和计算,但这种单机的方式由于内存限制的原因,对图本身的规模有了很大的限制^[2]。

为解决单机图计算的内存瓶颈问题,卡内基梅隆大学的 Select 实验室开发了 Graphchi,它是 Graphlab 的一个分



支,采用基于磁盘的以顶点为中心的计算模型,它可以在PC上进行大规模的类似于社会网络分析的图计算,而不需要分布式的集群和云服务,也不需要考虑内存的限制。

2.2.1 基于磁盘的计算

要想利用单机而不利用集群来并行地进行大规模的图计算,首当其冲面临的是存储问题。庞大的图数据在内存中处理上百万条边需要几十或几百吉字节的 DRAM,因为其价格昂贵,目前只对高端服务器有可用性,所以 Graphchi 将目光投向了价格低廉、容量大的磁盘作为其外部存储,用基于磁盘的计算模型减少内存的使用和随机存取问题。

然而,如何从磁盘上处理大规模的图数据是一个难题。为了处理这个问题,Graphchi 采用了新颖的 PSW (parallel sliding window,并行式滑动窗口)模型,从磁盘上处理大的图数据。

2.2.2 PSW 模型

Graphchi 采用了 PSW 模型从磁盘处理大的图数据,不同于分布式框架通用的 BSP 模型,PSW 模型能够异步处理存储在硬盘上的可扩展图数据,有效规避了“木桶效应”。

PSW 模型中,边的信息分区 *shard* 采用不相交子集(顶点集被分为 P 个子集 $interval(i)$)的形式关联存储,这种存储方式将每个子集以滑动窗口的形式分别从硬盘装入内存。Graphchi 分多次取节点子集 $interval(i)$,每次取 1 个,并且根据节点子集中的点信息构造子图进行计算。在第 p 次操作所需的子图数据载入后,每个节点并行地执行用户定义的更新函数,并更新节点,节点子集更新后的块文件将被写入磁盘。

图 2 表示 PSW 模型进行一次迭代的滑动窗口示意,顶点被分为 4 个不相交的子集,每个自己都关联一个分区,计算过程是构建一次子图顶点的子集。从内存的分区中读取顶点的入边,从每个滑动的分区中读取取出边,每个分区的最顶端为当前的滑动窗口。

2.2.3 Graphchi 基于 PSW 模型的改进

为了支持 Graphchi 的可扩展性,Graphchi 对 PSW 模型进行了改进,通过实现一个简化的、高效的 I/O 缓存树来

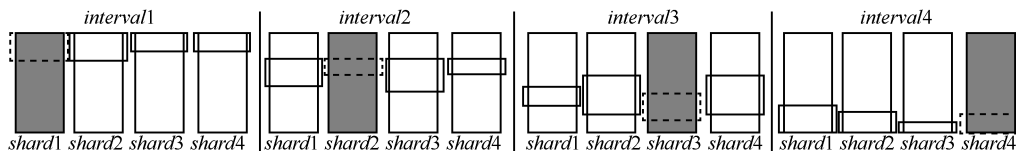


图 2 PSW 进行一次迭代的滑动窗口示意

支持图边的增加和删除,改进的 PSW 模型如图 3 所示。

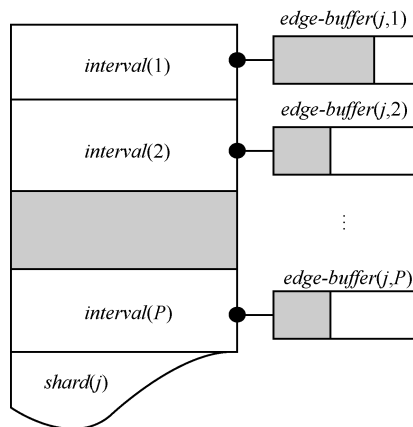


图 3 改进的 PSW 模型

图 4 显示了 Graphchi 执行 PSW 模型的流程。

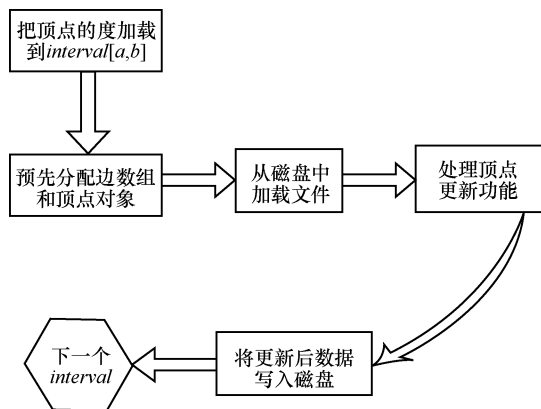


图 4 Graphchi 流程

3 Graphchi 应用前景

3.1 分布式图计算局限性

基于图的分布式框架通过云平台的计算资源处理上百万条边的图数据有很高的效率,但是利用分布式集群进行图计算仍然面临较高的硬件和技术要求,对于那些没有分布式专业背景、没有足够的硬件资源的人来说,仍然是个巨大的挑战。

首先,使用分布式框架时,使用者面临如何将强耦合性的图数据进行分割,部署到集群计算节点上的问题^[3]。其

次,图的分布式计算涉及复杂的处理过程,需要大量的迭代和数据通信,大多数分布式系统用到的是 BSP 模型,是一种同步计算模型,对于消息的处理容量有限,网络的延迟以及节点间的通信会造成“木桶效应”。再次,分布式框架处理需要计算耗时的大规模图数据时,重复计算以及系统故障使效率大大降低,同时系统的容错性也是制约运算效率和稳定性的关键瓶颈。最后,对于编程者来说,调试和优化分布式算法有很大的难度。

相对于复杂的分布式集群框架来说,简单的单机进行大规模的图计算,能够规避分布式框架的问题。使用者不需考虑强耦合性的图数据如何分割放置到分布式的集群节点中,也不需管理和部署众多的集群节点,并且可以减少分布式集群节点中的通信开销,规避网络延迟、“木桶效应”等问题。

例如,企业如果要在同一张图上计算多种任务(个性化推荐、图的社团发现等),在不同的国家、不同的利益集团都要计算同一个任务的情况下,企业要想提高运算速度,就必须增加集群节点,也就是说要增加成本。但是,如果一台机器上可以处理一个这样的大任务,企业可以为每台机器分配一个任务,每台机器之间无需互相通信,当增加机器数量时,吞吐量也随之增加,这样多种任务的处理将会变得非常简单、有效。

仅仅需要一台机器就可以对大规模的图数据进行分析处理和挖掘,这可以大大简化分布式集群处理框架的复杂性,如图 5 所示。

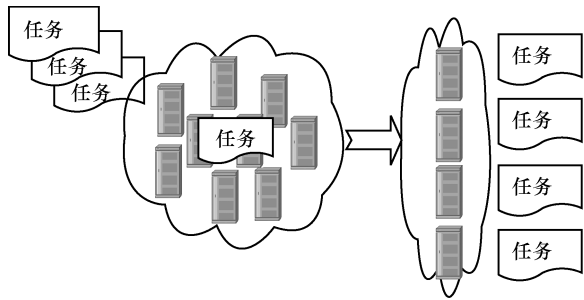


图 5 集群与单机处理多任务情况

本文对单机处理图数据技术 Graphchi 的发展、应用场景以及性能进行了研究,并进行了试验。

3.2 单机 Graphchi 应用前景

在图挖掘方面,Graphchi 实现了 PageRank、连通分支、社区发现等算法处理和分析现实世界中大规模的图数据;另外,应用在协同过滤算法的推荐系统中,Graphchi 从纷

繁复杂的信息中找出可向用户推荐的有价值的信息。不仅在图挖掘和协同过滤方面,Graphchi 还提供了通用的编程框架,支持使用者调用自己的算法对图进行分析和计算,这使得 Graphchi 使用起来更加灵活,也有更加个性化的可用性。

当前 Graphchi 中一些应用的算法设计还不尽完善,但是随着技术的发展以及应用的普及,Graphchi 因其在图计算方面独特的模型,其单机运行的简便、高可用和可观的运行效率,将在大规模图计算方面表现出越来越广阔的应用前景。

为了验证 Graphchi 在不同硬件环境下,不同数量级别社交网络图数据应用中的可行性和可用性,下文对不同数量级的数据在两种不同的环境进行了相应的测试,并且和其他分布式框架进行了对比。

4 Graphchi 的可行性、可用性评估实验

4.1 测试环境

- Intel (R) Core (TM)2Duo CPU T6600 @2.20 GHz、RAM 2 GB、Ubuntu11.04。
- Dell 服务器 QEMU Virtual CPU Version (cpu64-rhel6) 6 核 CPU、4 GB 内存(未特殊注明,本文中数据测试环境均为服务器环境)、CentOS 6.4。

4.2 数据集说明

本文采用的数据集来自斯坦福的 Snap 网站^[4]以及 Netflix 网站。测试的数据集为 Wiki、Twitter、Facebook、Friendster 等流行的社交网站,数据集大小为 40 MB~30 GB。

表 1 是对实验中使用到的测试数据集的说明,其中 $|V|$ 表示测试数据集的顶点数目, $|E|$ 表示测试数据集边的数目。

表 1 测试数据集

数据集	$ V $ 个	$ E $ 个
Wiki-Talk	2 394 385	5 021 410
LiveJournal	4 847 571	68 993 773
Facebook_sg	59 216 215	92 522 018
Wikipedia	27 154 798	601 038 302
Twitter-2010	61 578 415	1 400 000 000
com-friendster.ungraph	124 836 179	1 800 000 000
SmallNetflix ^[5]	95 526	545 177
Netflix	2 667 199	100 480 507

4.3 Graphchi 测试结果

图 6 表示的是 PageRank 和 CommunityDetection 两种算法对除 Netflix 数据集外所有数据集进行的测试,



X轴表示边集的数量,Y轴表示对应的运行时间。从图中可以看出,对于两种不同算法,随着数据集的增大,运行时间大体呈线性增长。

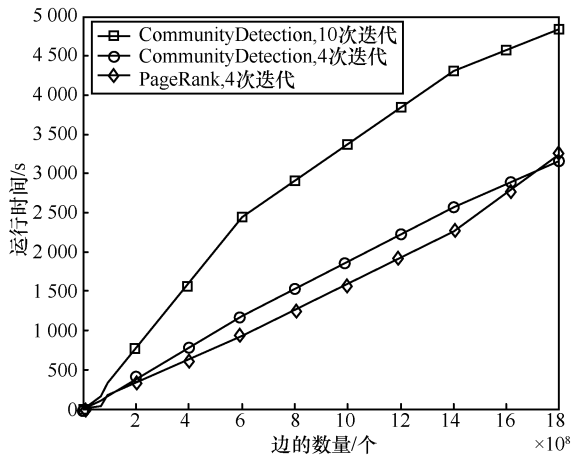


图6 两种算法随边规模增长运行时间变化

图7表示PageRank和CommunityDetection两种算法以及CommunityDetection分别在4次和10次迭代过程中,吞吐量随边数的变化。X轴为边集的数量,Y轴表示吞吐量(系统每秒处理边的数量)。Graphchi每秒可以处理的边的数量为 $0.2 \times 10^6 \sim 2 \times 10^6$ 个。

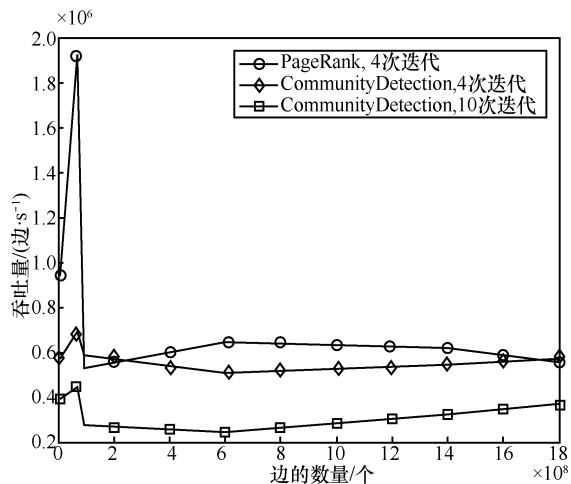


图7 两种算法吞吐量随边数变化

Graphchi测试Twitter 2010年所有的user-follower关系,14亿条边、4千万个顶点共20 GB的数据,PageRank算法需要46 min,CommunityDetection算法10次迭代需要70 min, Trianglecounting算法需要130 min;测试在线游戏Friendster,18亿个顶点、6千万条边共30 GB的数据集com-friendster.ungraph,PageRank算法4次迭代需要54 min。

可见,Graphchi可以在1 h左右完成对社交网络一年数据的分析。这种处理能力完全可以满足使用者对大规模图数据进行计算的需求,并且具有较好的吞吐量。

图8表示的是Graphchi测试两种数据集smallNetflx和Netflx协同过滤的7种算法进行6次迭代的运行时间。X轴表示7种协同过滤算法:SGD、ALS、RBM、SVD++、biasSGD、CCD++和PMF,Y轴对应的是各种算法的运行时间。

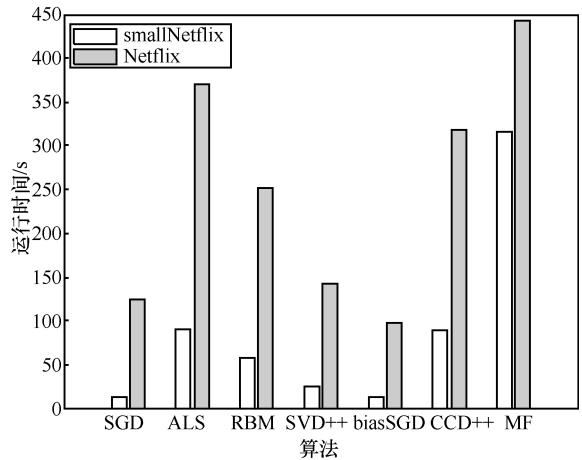


图8 协同过滤算法运行时间

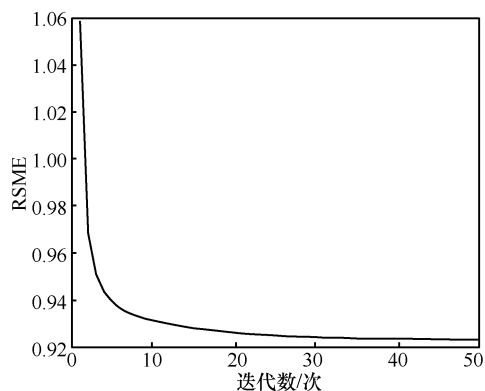
Graphchi在协同过滤中的运行时间最长为450 s,Netflx数据集的时间不超过300 s。

图9表示的是SGD算法运行50次迭代的运行时间以及RSME(root square mean error)均方差的变化曲线。迭代20次时,算法的RSME已经趋于稳定,无限接近于0.92,而此时的运行时间约为350 s。

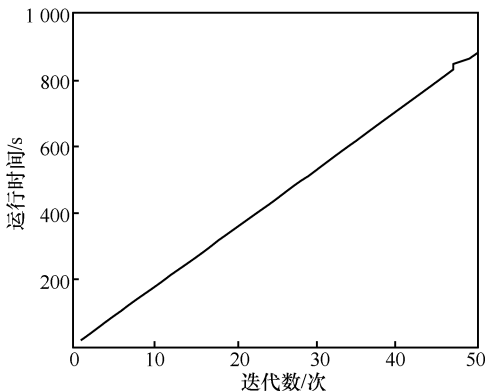
可见,Graphchi在协同过滤方面表现出良好的性能,可以在几百秒的时间内处理2 GB规模的数据。

图10表示的是PageRank、CommunityDetection和ConnectedComponents 3种算法,wiki-Talk和com-orkut两种测试集分别在2核CPU和6核CPU上运行时间的对比。X轴表示运行时间,Y轴表示3种算法以及两种数据集。从图10中可以看出,在相同数据集上6核CPU的运行时间要比2核CPU运行时间快了近10倍。

图11表示的是协同过滤的3种算法,Netflx测试集分别在2核CPU和6核CPU上运行时间的对比。X轴表示运行时间,Y轴表示协同过滤4种不同算法。Netflx数据集在6核CPU上的运行时间比在2核CPU上的运行时间快了5~10倍。



(a) SGD算法RSME随迭代数变化曲线



(b) SGD算法50次迭代运行时间变化曲线

图9 SGD算法RSME随迭代数变化曲线以及运行50次迭代的时间变化曲线

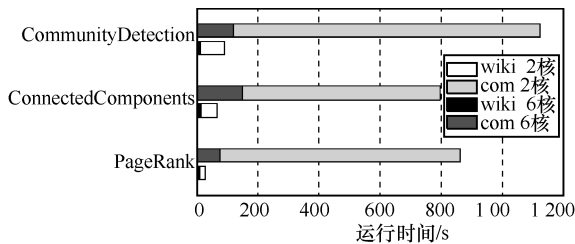


图10 3种算法两种数据集在不同核数CPU上运行时间对比

图11表示协同过滤4种算法在不同核数CPU运行时间的对比。

随着CPU数目的增加,运行速度也有明显的提升。相信在配置更高的单机上运行Graphchi将会有更加可观的性能。

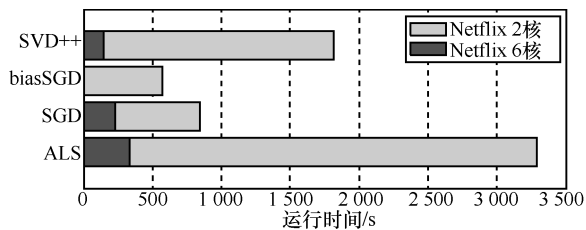


图11 协同过滤4种算法在不同核数CPU运行时间对比

4.4 可行性、可用性分析对比

本文对比了一些分布式的图处理框架,参考了一些其他文章的测试结果,见表2。

在有50个节点、100个CPU的Spark框架下,在Twitter-2010数据集上运行5次迭代的PageRank算法的时间比Graphchi在4核CPU的环境中运行相同数据集快了大约5倍。在有1636个节点的Hadoop框架运行Twitter-2010数据集的PageRank算法迭代一次,Graphchi比Hadoop快45倍,比Powergraph慢了155倍。与运行在AMD服务器上的Graphlab相比,用ALS算法测试NetfliX数据集,Graphchi运行时间是Graphlab的2.5倍。Trianglecounting算法测试Twitter-2010数据集在1636个节点的Hadoop环境,Graphchi比Hadoop快了3倍。

相对于Hadoop来说,Graphchi的大规模图数据方面的性能远优于Hadoop;在协同过滤方面,Graphchi和Graphlab性能相差不大;与性能较好的Spark相比,Graphchi的性能表现也在可以接受的范围内;对于性能强大的Powergraph,Graphchi性能还是有一些差距。

总体来说,Graphchi以单机运行方式进行图运算所表现出的性能可以和一些分布式的框架相媲美,虽然不及性能强大的Powergraph,但是这样的性能表现已经可以满足一定规模的图运算了。这样的性能表现已足以为成本不足、硬件设备配置不高的中小企业或者个人提供高可行、高可用的社交关系网络图数据分析和挖掘平台。

5 Graphchi 电信图数据挖掘应用

为验证Graphchi对电信大规模图数据的处理能力,本

表2 Graphchi和不同框架性能对比

算法 & 数据集	迭代数	对比结果	Graphchi
PageRank&Twitter-2010	5	Spark(50个节点100个CPU)486.6 s ^[6]	2 768 s(46.1 min)
PageRank&Twitter-2010	1	Hadoop, 1 636个节点, 423 min ^[7]	558 s(9.3 min)
		Powergraph, 64×8核, 3.6 s ^[8]	
Trianglecounting&Twitter-2010	—	Hadoop, 1 636个节点, 423 min ^[7]	7 821 s(130 min)
ALS&NetfliX, 参数 D=20	10	Graphlab(AMD 服务器), 4.7 min ^[9]	657 s(10.9 min)



文构造了电信通话清单数据约 20 GB,有 4 000 万个顶点、14 亿条边(已对数据进行匿名处理),格式见表 3。

表 3 电信数据格式

用户 1	用户 2
用户 2	用户 3
用户 2	用户 4
用户 3	用户 5
...	...
用户 m	用户 n

5.1 PageRank 算法挖掘核心人物

PageRank 算法是 Google 用于用来标识网页的等级/重要性的一种方法,是 Google 用来衡量一个网站好坏的唯一标准。它基于马尔科夫状态转移理论,通过网页的链入数对网页进行投票来得出重要性排名。

发展到目前,PageRank 算法也被广泛用于关键人物挖掘等社交关系网络分析中。本文应用 Graphchi 的 Pagerank 算法,对电信关系网络数据进行 Rank 值的计算,从而找出关键人物。

表 4 是采用 Graphchi 的 Pagerank 算法对电信数据集进行计算 Rank 值的排名前 10 的结果,在 4 000 万个用户中,标号为 1 653 的用户的重要性最高,为核心用户,应该对其重点挖掘和营销推广。

表 4 PageRank 前 10 名排名结果

排名	标号	Rank 值
1	1 653	7 788.59
2	21 407	4 524.67
3	1 786 295	3 931.42
4	1 803 885	3 624.54
5	5 299	3 193.47
6	2 581 430	3 077.34
7	3 806	2 323.41
8	3 154 485	2 068.3
9	3 676 640	1 857.47
10	3 653 185	1 803.97

5.2 CommunityDetection 算法进行社区发现

CommunityDetection 社区发现算法用于发现网络中的社区结构,也可以看作是一种聚类算法。同一社区之间的节点与节点之间的关系比较紧密,而社区与社区之间的关系比较稀疏。如果两者之间的联系越频繁,那么其社交关系就越紧密。如图 12 所示,可以找到 3 个关系紧密的社区。

表 5 为采用 Graphchi 的 CommunityDetection 算法对电信数据集进行社团发现的结果,共发现社区 1 733 613 个,最大社区有 35 558 616 个用户。运营商可以对每一个社团分析其相似特征,进行潜在客户挖掘以及后续的客户关系维护。

表 5 社团发现前 10 名社团大小

	标号	社区大小/个
1	56 569 558	35 558 616
2	57 092 875	1 802
3	57 613 117	698
4	57 613 117	612
5	57 614 778	596
6	57 615 142	590
7	57 599 599	589
8	57 600 140	589
9	57 600 474	589
10	57 606 609	578

6 结束语

电信技术的发展带来了大规模的电信数据,面对日趋激烈的市场竞争环境,电信运营商如何从通信数据抽象成大规模的图网络数据中挖掘有价值的信息,维护客户关系,进行针对性服务成了关注的焦点。

本文阐述了可以对大规模电信社交网络图数据进行挖掘和计算的几种分布式框架和单机计算框架。并且通过实验和对比,说明单机的 Graphchi 运行各算法在不同规模数据集所用的时间和其他可以运行这些算法的框架相比在合理的范围内,使用廉价的硬盘和普通的服务器就可以

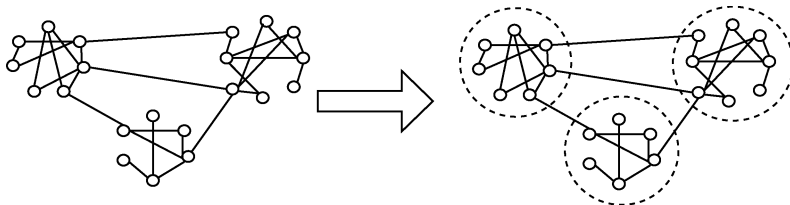


图 12 社区关系网络

实现大规模的图计算,并且有良好的性能,它可以像其他分布式框架一样,在解决大规模社交关系网络图数据时有很好的运行效率。

同时,Graphchi 简单、高可用的性能使其在解决其他分布式系统能解决的大规模电信社交关系网络图数据方面也有很高的运行效率,其在一定规模的图数据量上的应用前景不可限量。但是,随着当前信息时代数量的不断扩增,对图数据处理的需求越来越高,Graphchi 能否继续承载更高数据量的分析处理任务仍然是一个问号,本文也提到了并行分布式框架在超大规模的社交关系网络图数据挖掘中,表现出强大的处理能力和效率,相信并行处理将是超大规模社交关系网络图数据处理发展的必然趋势。

参考文献

- 1 于戈,谷峪,鲍玉斌等.计算环境下的大规模图数据处理技术.计算机学报,2011(10):1754~1755
Yu G, Gu Y, Bao Y B, *et al.* Large scale graph data processing on cloud computing environment. Chinese Journal of Computers, 2011(10):1754~1755
- 2 Malewicz G, Austern M H, Bik A J, *et al.* Pregel: a system for large-scale graph processing. Proceedings of SIGMOD ACM, Indianapolis, Indiana, 2010
- 3 杨苗苗,李跃辉,刘静等.基于Hadoop的电信频繁交往圈算法研究.电脑知识与技术,2013(9)
Yang M M, Li Y H, Liu Jing, *et al.* Research of algorithms about frequency telecom SNA based on Hadoop. Computer Knowledge and Technology, 2013(9)
- 4 Stanford Large Network Dataset Collection. <http://memetracker.org/data/>, 2014
- 5 SmallNetflix_mm. http://www.select.cs.cmu.edu/code/Graphlab/datasets/smallNetfli-x_mm, 2014
- 6 Stanton I, Kliot G. Streaming graph partitioning for large distributed graphs. Technical report, Microsoft Research, 2012
- 7 Kwak H, Lee C, Park H, *et al.* What is twitter, a social network or a news media. Proceedings of the 19th International Conference on World Wide Web, Raleigh, NC, USA, 2010: 591~600
- 8 Gonzalez J, Low Y, Gu H, *et al.* Powergraph: distributed graphparallel computation on natural graphs. Proceedings of the 10th USENIX Symposium on Operating Systems Design and Implementation (OSDI'12), Hollywood, CA, USA, 2012
- 9 Low Y, Gonzalez J, Kyrola A, *et al.* Graphlab: a distributed framework for machine learning in the cloud. Proceedings of the 26th Conference on Uncertainty in Artificial Intelligence (UAI), Catalina Island, USA, 2010
- 10 Zaharia M, Chowdhury M, Franklin M J, *et al.* Spark: cluster computing with working sets. Proceedings of HotCloud 2010, Boston, MA, June 2010
- 11 Seo S, Yoon E J, Kim J, *et al.* HAMA: an efficient matrix computation with the MapReduce framework. Proceedings of the IEEE 2nd International Conference on Cloud Computing Technology and Science, Washington, DC, USA, 2010
- 12 Aapo Kyrola, Guy Blelloch, Carlos Guestrin. Graphchi: large-scale graph computation on just a PC. Proceedings of the 10th USENIX Symposium on Operating Systems Design and Implementation (OSDI '12), Hollywood, CA, USA, 2012
- 13 Suri S, Vassilvitskii S. Counting triangles and the curse of the last reducer. Proceedings of the 20th International Conference on World Wide Web, Hyderabad, India, 2011: 607~614
- 14 Bertsekas D P, Tsitsiklis J N. Parallel and distributed computation: numerical methods. Prentice-Hall Inc, 1989
- 15 Leskovec J, Lang K, Dasgupta A, *et al.* Community structure in large networks: natural cluster sizes and the absence of large well-defined clusters. Internet Mathematics, 2009, 6(1):29~123
- 16 Zhu X, Ghahramani Z. Learning from labeled and unlabeled data with label propagation, 2002
- 17 Kang U, Chau D, Faloutsos C. Inference of beliefs on billion-scale graphs. Proceedings of the 2nd Workshop on Large-scale Data Mining: Theory and Applications, Washington, DC, USA, 2010
- 18 Gorton. Software architecture challenges for data intensive computing. Software Architecture, 2008

[作者简介]



刘丽娇,女,南昌大学硕士生,主要研究方向为数据挖掘、知识工程、软件工程等。

陶俊才,男,南昌大学教授,主要研究方向为软件工程、网络计算与系统集成、模式识别、知识管理与决策支持。

肖晓军,男,博士,广州优亿信息科技有限公司总经理,具有多年电信行业管理工作经历,主要研究方向为大数据、数据挖掘与电信行业应用等。

卢宇,男,硕士,广州优亿信息科技有限公司技术总监,主要从事大数据、机器学习与电信行业应用等相关研发工作。