ISSN 1673-9418 CODEN JKYTA8
Journal of Frontiers of Computer Science and Technology
1673-9418/2013/07(12)-1083-10
doi: 10.3778/j.issn.1673-9418.1305031

E-mail: fcst@vip.163.com
http://www.ceaj.org
Tel: +86-10-89056056

IncGraph:支持实时计算的大规模增量图处理系统*

申 林+,薛继龙,曲 直,杨 智,代亚非
北京大学 计算机科学与技术系,北京 100871

IncGraph: Large-Scale Graph Processing System for Real-Time and Incremental Computing*

SHEN Lin+, XUE Jilong, QU Zhi, YANG Zhi, DAI Yafei
Department of Computer Science and Technology, Peking University, Beijing 100871, China
+ Corresponding author: E-mail: aaronlshen@gmail.com

SHEN Lin, XUE Jilong, QU Zhi, et al. IncGraph: large-scale graph processing system for real-time and incremental computing. Journal of Frontiers of Computer Science and Technology, 2013, 7(12): 1083-1092.

Abstract: With the increasing popularity of online social networks (OSNs), more and more related applications need a computation framework for processing large social graphs in a real-time manner. However, the existing graph processing systems, such as Google's Pregel, cannot achieve real-time performance due to processing the graph in global and batch manners. Therefore, this paper proposes a new graph computation model that restricts the computation to a local area when a node is updated. In essence, it transforms the traditional batch and global computations to a series of incremental and local computations. By this way, people achieve the goal of real-time computation, and avoid the overhead of duplicate computation on nodes that have not changed. Based on this new computation model, this paper designs IncGraph, a real-time large graph processing system. To achieve an efficient and scalable implementation, this paper uses graph partition techniques to exploit the computation locality in social graph, moreover, uses proactive computing trigger and inversing update chain to realize the real-time and reliable computing. Finally, this paper uses real social network data to validate the system performance. IncGraph can provide social network applications a more flexible computation framework.

Key words: graph processing system; incremental graph processing; graph partition; proactive computing trigger; inversing update chain

* The National Natural Science Foundation of China under Grant No. 61073015 (国家自然科学基金); the National Science and Technology Major Special Project of China under Grant No. 2010ZX03004-001-03 (国家科技重大专项).
Received 2013-05, Accepted 2013-07.
CNKI网络优先出版:2013-08-08, http://www.cnki.net/kcms/detail/11.5602.TP.20130808.1030.001.html.

摘 要:随着社交网络的流行,越来越多的相关应用要求能够实时地在大规模社会网络图上进行分析和计算。而目前的图处理系统,如Google的Pregel,是全局、批量处理的图处理系统,并不能实现对图的实时计算。因此,提出了一种新的图增量处理模型,当一个节点发生变化时,只需要以传播的方式更新局部范围内受影响节点。它本质上将传统的批量全局计算模型,转化成一系列的增量的、局部的图计算,保证对图变化的实时处理,并通过避免没有更新节点的重复计算来降低开销。基于这种新的图计算模型,设计了一个低开销、实时的图处理系统——IncGraph,它通过图切分技术将计算局部化,保证了计算的低开销,同时利用主动计算触发和反向链式更新技术,保证了计算的实时性和可靠性。利用真实的社交网络数据证明了IncGraph的低开销、实时性和扩展性。IncGraph的提出会为社交网络应用提供更为灵活的计算框架。

关键词:图处理系统;增量图处理;图切分;主动计算触发;反向链式更新

文献标志码:A **中图分类号:**TP393.0

1 引言

最近几年,人们越来越依赖社会网络(如Facebook、Twitter、新浪微博、人人网)来和朋友进行交互,以及了解最新的信息和新闻。例如Facebook、Twitter的访问量一直排名在最热门网站的前20。统计表明,人们在社交网络上的滞留时间要比在其他任何一个网络都要长[1]。

社会网络的流行成为构建实时图处理系统的一个重要需求。一方面,社会网络应用需要能够对大规模社会网络图进行实时分析,例如,社会网络需要依据网络拓扑图的实时变化进行朋友推荐、新鲜事排序、广告投放或者实时搜索,这对图运算的实时性提出了更高的要求。另一方面,目前流行的图计算模型Pregel[2]是全局的、批量的计算模型,由于社会网络的规模比较大,单次基于图的运算需要很长时间。因此,社会网络的实际应用需要一种实时性很高的图计算系统。目前国际上对这种实时图计算系统的研究刚刚起步,还没有具体的方案被提出。

考虑到图的更新是增量和局部的(图数据的大部分是不变的),本文提出了一种新的图计算模型来将传统的批量全局计算模型,转化为一系列增量的局部计算,保证对图变化的实时处理,并通过避免对没有更新的节点的重复计算来降低开销。基于这种新的模式,本文设计了大规模图的实时处理系统IncGraph。它利用了以下关键技术来保证系统的高效率和低开销:

(1)图切分技术。通过将社会网络图切分为多个尽可能独立的、连接紧密的子图,并把每个子图上的运算放在一台机器上,保证了计算的局部性,减少了跨机器间的通迅开销。

(2)主动计算触发技术。当图中节点发生更新时,主动触发被影响节点的计算,保证了系统只需要在更新时发生增量的局部计算。

(3)反向链式更新技术。通过正向传播计算,反向链式更新节点索引和数据的方式,保证了计算的低开销和可靠性。

最后,采用大规模在线社交网络(人人网)的真实数据进行验证,证明了系统的计算开销非常小,对结果的更新控制在毫秒级别。

本文的主要贡献如下:

(1)提出了一种新的增量的实时图处理模型。

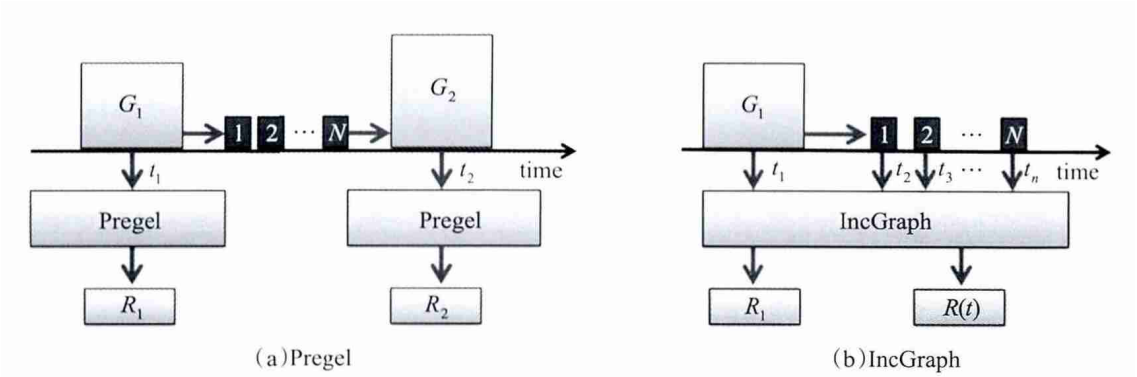
(2)基于新的模式设计了图的实时计算系统。

(3)利用实际的社会网络图进行验证。

本文组织结构如下:第2章介绍实时图计算模型;第3章给出基于模型设计的实时图处理系统;第4章对系统的实验评测结果进行分析;第5章和第6章为相关工作和总结。

2 模型

本章将详细介绍IncGraph的增量处理模型。首先,给出IncGraph的计算模型概览;其次,介绍增量图处理模型的设计及原理;最后,介绍模型的实现以及部分API(application programming interface),并以TrustRank为例介绍如何将一个计算模型抽象到IncGraph系统。



(a)Pregel　(b)IncGraph

Fig.1　Comparison of batching model and real-time model

图1　批处理与实时计算模型对比

2.1　计算概览

IncGraph是一个增量、实时的图处理模型，而当前广泛使用的图处理系统（如Pregel）均为批处理系统，其需要定期将全图重新计算，即使全图局部更改，也需全局计算，具有较大的开销和延时。与传统的批量图处理系统相比，IncGraph不需要每次全局地重新计算全图，只需将每次局部更新实时地输入系统，IncGraph就会将结果更新至最新状态。如图1所示，在Pregel系统中，对图 G_1 进行一次批处理得到结果 R_1，当经过时间 t 后，变成图 G_2，又需要一次重新全局计算。但在IncGraph中，只需在每个时间 t 处理局部的更新，并实时地输出最新的结果 $R(t)$。

因为IncGraph主要关注如何处理增量更新计算，所以在进入实时计算之前，需要将现有图的结果以离线批处理的方式计算出最新结果。离线批处理模块可以利用现有成熟模型来完成，如Pregel、Giraph或MapReduce等。当批处理完成后，IncGraph将进入实时守护状态，监控图的更新，并实时作出计算响应。

2.2　增量图处理模型

本文所定义的增量式图计算是指，随着图数据或图结构不断变化，系统可以根据最近变化实时计算最新结果。当图发生部分更新改变时，系统只需局部增量地计算由更新所引起的部分结果改变，并不需要对全图重新处理和计算，称这样的计算模式为增量式计算。当然，并不是所有图算法都能进行增量式计算，因此需要给出问题域空间，如图2所示。

现在详细介绍增量式图计算模型的细节。模型采用以节点为中心的计算模式，每个节点在逻辑上可

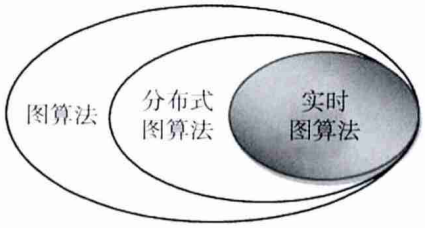


Fig.2　Problem domain of graph algorithm

图2　图算法问题域

以并行地执行由用户定义的程序，并将计算相关的结果保存在该节点上。开始，当图的某个节点发生变化时，该节点将会触发由用户自定义的程序，从而计算出该节点的最新值，该值保存在节点的state变量中。每次state变化都会对其他节点带来影响，这个影响值可由用户自定义计算，默认为state的变化，并保存在impact变量中。当impact足够大时，节点将该影响传播给其邻居节点，从而邻居节点被触发，也开始重新计算，整个过程直到传播停止为止。图3是以节点的角度来描述整个增量计算的流程。

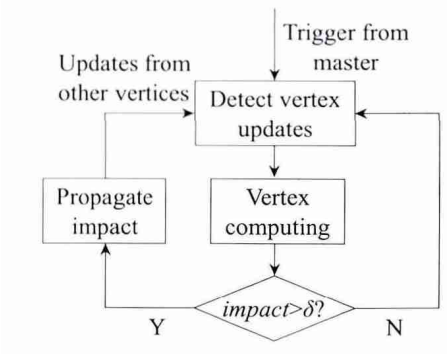


Fig.3　Flow of IncGraph model

图3　IncGraph计算模型流程

上述模型可以保证当节点有局部更新时，只会影响一部分节点的重新计算，其他未被影响传播到的节点一直保持守护状态。

2.3 API及模型实现

本节介绍IncGraph模型所提供的API及实现。

首先，IncGraph的基本计算接口与Pregel类似，这是因为其都需要通过节点的本地计算和与其他节点的通信来完成。除此之外，还需要定义发送影响值接口send_impact()、接收影响值接口receive_impact()和聚合影响值接口aggregate()，见图4。其中，send_impact()接口是当某节点进行重新计算后，发送其对邻居节点的影响值并触发邻居节点的操作；aggregate()接口是用来聚合所有接收到的影响值到当前自身影响值impact上；receive_impact()接口用于接收影响值，该接口是被动触发的，其主要功能是接收父节点发来的影响值，并将其聚合到现有影响值impact上。这里的聚合函数需要用户自定义，默认情况是相加。然后判断影响值的大小，如果大于一个阈值，就会触发用户自定义的计算程序进行重新计算，并将计算出的影响值发送给其所有邻居节点，从而触发邻居节点的计算，详细过程见算法1。该过程的算法复杂度与具体应用算法和更新变化的大小相关，其最优情况只需要$O(1)$的复杂度，最坏情况不超过一次批量计算复杂度$O(nm)$，n为节点数，m为迭代轮数。

```
Class Vertex{
//properties
   Type state;
   Type impact;
//methods
   send_impact(tar, send_impact);
   receive_impact(src, recv_impact);
   aggregate(recv_impact);
```

Fig.4 Vertex interface of IncGraph

图4 IncGraph的节点接口

算法1 IncGraph Vertex Computation

```
1. receive_impact(ipct)
2. impact=aggregate(impact, ipct)
3. If impact<delt:
4.    continue
5. Else:
6.    //user_defined_computation()
7.    For nbr in get_neighbor():
8.       send_impact(nbr, my_impact)
```

2.4 应用示例

本节以计算TrustRank[3]为例演示如何用IncGraph实现增量和实时计算。TrustRank是类似于PageRank的网络用户信誉值计算程序，其需要全局迭代计算。TrustRank计算公式如下所示：

$$T(u)=d\times\sum_{v}\frac{T(v)}{out\_deg(v)}+(1-d)\times I(u)$$

式中，$T(u)$表示用户u的TrustRank值；$I(u)$表示节点u的初始化TrustRank值，在TrustRank中会有一个可信种子集合S，对于集合内元素u，其初始值$I(u)$便是$1/|S|$，其他节点的初始值为0。应用到IncGraph模型中，每个节点的state便是当前该节点TrustRank值，节点的impact可定义为父节点的TrustRank值除以其出度，即$T(v)/out_deg(v)$。这里的聚合函数aggregate()只需要加法即可，impact的误差delt可以按照TrustRank迭代误差定义，一般取1E−8。

抽象成IncGraph计算模型后，就可以通过初始化计算，进入实时增量计算状态。

3 IncGraph系统架构

3.1 系统概览

IncGraph是一个实现大规模实时处理和增量计算模型的图处理系统。通过控制节点和工作节点的相互协作，以及运用图切分技术、主动计算触发技术和反向链式更新技术，完成系统实时、增量计算的功能。

3.2 核心组件介绍

如图5所示，IncGraph图处理系统主要由两部分组成：Coordinator（控制节点）和Worker（工作节点）。Coordinator是整个系统的指挥者和协调者，主要负责接受外部的更新和查询请求，图的划分和索引的建立，各个划分子图索引与机器的对应，以及索引的存储。Worker是实时增量图处理模型的具体实施者，主

要负责划分子图的计算，本地索引的建立以及索引和数据的存储等。此结构设计的基本思想是，通过Coordinator将图更新的任务分解，根据划分子图的规则，将图处理的任务分发到多个Worker，并监控其完成的进度和产生的数据。

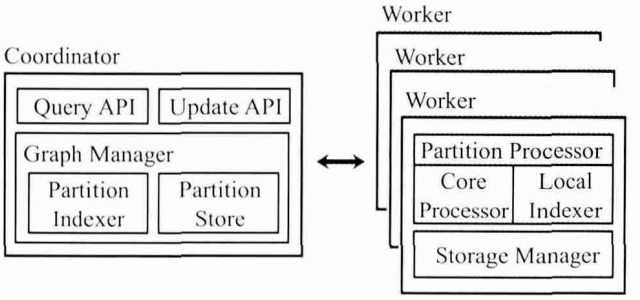


Fig.5　Architecture of IncGraph

图5　IncGraph系统架构

3.2.1　控制节点Coordinator

Coordinator由API与Graph Manager两部分组成。其中Query API和Update API是IncGraph对外访问的接口，负责接收来自外部的数据更新请求和数据查询请求，并返回相应的结果。

Graph Manager是Coordinator的核心组件。其中的Partition Indexer主要完成两方面的功能：

(1)在初始化状态中，当一个未处理的完整图提交到系统后，Partition Indexer会根据Worker的数目和图本身的拓扑结构将其划分成均匀的子图，子图所包含的具体数据将分发并存储到相应的Worker上，等待新节点的加入或者更新操作的触发，而子图(包括其中每个节点)与Worker的对应关系，作为全局索引存储在Partition Store中。

(2)在工作状态中，当系统接收节点更新/查询请求后，Partition Indexer找到需要计算的节点数据存放位置，即相应Worker的位置，并将请求转发到相应Worker上进行服务。

需要指出的是，Partition Indexer建立索引和查询索引的操作都是在内存中进行的，后台会有单独的异步线程将更新的索引定时写到磁盘上，防止内存突然失效时数据的丢失。相应的，在系统开始工作之前，Partition Indexer也会将索引从Partition Store中读入内存。

Graph Manager中的Partition Store主要负责划分子图索引的持久化存储，其中包括定时从内存中读取索引并写入磁盘的异步程序，以及一套本地文件系统/数据库的读写接口。

3.2.2　工作节点Worker

Worker由Partition Processor和Storage Manager两部分组成。

Partition Processor是Worker的核心组件。其中的Core Processor主要完成两方面的功能：

(1)在接收到Coordinator进行计算的命令后，Core Processor开始在内存中运行算法模型，并将更新的索引交由Local Indexer来处理。

(2)在计算过程中，如果某一个图节点的信息没有在Worker本地，Core Processor向Coordinator的Graph Manager提交索引查询请求，在索引返回后，Core Processor将其上的计算任务转发到相应的Worker上并等待其返回结果，等所有节点都处理完成，向Coordinator汇报结果(详细流程见2.3节)。

Partition Processor中的Local Indexer主要负责索引的更新，以及定期将内存中的索引交于Storage Manager写入磁盘作持久化存储。

Storage Manager主要负责存储更新后的划分子图的数据以及相应索引。类似Partition Store，其中也包括定期从内存中异步读取索引并写入磁盘的异步程序，以及本地文件系统/数据库(此系统中用的是HBase)的一套读写接口。

3.3　更新和查询协议

IncGraph基本的工作模式为更新数据和查询数据，相应的工作流程也主要介绍这两方面。

3.3.1　更新协议

为了更好地理解更新流程，本文使用模型中的网络拓扑进行举例。如图6所示，节点1、2、3、N是图中的四个节点，按照第2章的模型，当触发节点1的更新操作时，必须完成节点2、3和N的计算才能最终返回结果。假设节点1、2、3、N在初始情况下被Graph Manager切分到不同的Worker上，节点1在Worker 1上，节点2在Worker 2上，以此类推。接下来研究这种情况下系统是如何完成更新流程的(这是最复杂的情况，其他更新流程都可以看做此情况的特例)。

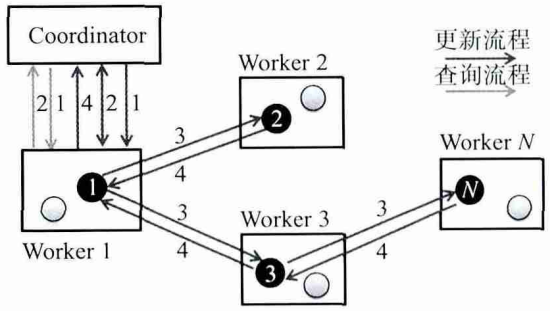


Fig.6 Chained update of IncGraph node

图6 节点链式更新示意图

协议1 更新协议

(1)当节点更新时(如图6中节点1),Coordinator记录图的更新,找到节点数据所在的Worker(如Worker 1),触发该Worker进行计算。

(2)Worker 接到命令,开始计算相关的数据,计算完成后,需触发其邻居节点进行计算。当发现节点不在本地,则需要向Coordinator查询。如Worker 1处理完节点1的数据后,就需要向Coordinator查询节点2和3的位置。

(3)得到邻居节点的位置后,即可向邻居节点所在的Worker发送计算消息,收到该消息的Worker执行相应节点的计算,并继续传播或根据条件终止计算。如Worker 1触发Worker 2和Worker 3对节点2和3的计算,计算传播到Worker N时停止。

(4)当每个Worker计算完成后,向上级Worker返回成功信息,直到Coordinator收到成功确认。例如,节点N的计算完成,Worker N向Worker 3返回成功。同样,Worker 3接收到成功信息后,更新本地索引,接着向Worker 1返回成功,直到返回到Coordinator。

3.3.2 查询协议

相比更新流程,查询流程比较直接,通过Coordinator上的全局索引找到节点数据存放的位置,读出即可。

协议2 查询协议

(1)当查询请求到达Coordinator时,Coordinator从索引中找到被查询节点所在的Worker,将查询请求转发到该Worker上。

(2)Worker收到查询请求后将被查询节点的数据返回给Coordinator,由Coordinator返回给查询客户端。

3.4 核心技术

本节主要介绍IncGraph系统设计中采用的几个关键技术,分别是图切分技术、主动计算触发技术和反向链式更新技术。

3.4.1 图切分

图切分技术主要目的有两个:一是为了保证各个机器的负载尽量均衡,这里的负载主要包括计算负载和存储负载。二是为了尽量保证图数据的局部特性,即将相关的紧密的节点存储在同一台机器上,从而可以让机器之间的切边(跨机器边)尽可能少,以减少跨机器的通信代价。目前,被广泛认可的切图算法主要是METIS[4],一个多层K路切图算法,可以保证每个部分均衡的前提下,尽可能减少划分的切边,其基本思想是将原始图进行粗糙化,变成一幅小图,从而进行切分,然后再反粗糙化实现全局切分。

在IncGraph中,图切分还有一个重要的功能是维护动态图的切分管理,比如节点或边的加入或删除。这里使用两个技术动态地维护图划分:一是贪心选择策略,即当有新的节点加入时,贪心地选择与该节点交集最大的机器,将该节点划入那台机器即可,其他的删除操作可以直接处理;第二是定期重新划分,长期的图更新操作会使得图的划分不均衡,造成机器负载不均,从而影响性能。因此,IncGraph会定期使用METIS重新对图数据进行切分,从而保证各个机器的负载均衡。

3.4.2 主动计算触发

增量实时计算的一个重要原则是要保证计算的低代价。在更新过程中,一般只有少部分节点需要进行计算,大部处于空闲状态。因此,在IncGraph中实现了主动计算触发技术,保证只有消息到达时,才触发计算,当计算完成后,将消息发送给其他节点,并进入休眠守护状态,直到等待下次消息的到来。通过主动触发技术,IncGraph可以保证在低代价的资源下实现增量计算。

3.4.3 反向链式更新

反向链式更新是为了实现系统增量处理功能。图中的每条边,正向传播的是计算请求(例如,寻找需要计算的节点,并将计算转发),反向传播的是更新结果。Worker只有在后续节点计算成功返回时,本地

索引才会更新。对图6所示的反向链式更新流程，当计算的节点数据不在本地时，Worker 1向Coordinator请求节点数据的索引，并将计算请求转发到Worker 2和Worker 3上。Worker 2完成计算即可返回成功结果，Worker 3与Worker 1情况相同，需要等待Worker N计算完成，才能返回成功结果。Worker 1需要等到Worker 2和Worker 3成功结果都返回，才向Coordinator汇报成功结果，并更新本地索引。如果Worker 3或者其他后续节点没有完成，Worker 1会保留相应状态，重新启动计算。

反向链式更新的好处是：

(1)链式计算与增量模型一致，只有当计算需要进一步迭代时，才增加Worker，保证当前状态最少的计算资源。

(2)如果采用正向更新，后续节点失败后需要回滚之前更新的索引，而反向更新很好地避免了该问题。

(3)某一个Worker将计算的状态汇报给其前继Worker，由其前继节点管理任务的启动、完成和重启，而不是交由Coordinator完成，避免集中式管理的复杂性和可能带来的性能瓶颈。

4　实验评测

本文用Java实现IncGraph的原型系统，并在分布式环境下作基本运行测试。本文将通过真实大数据来验证IncGraph的系统性能。

4.1　数据集与实验设置

本文实验数据来自人人网（http://www.renren.com）的真实图数据。人人网是中国最大的在线社交网络之一，有1.2亿注册用户，用户可以在网络中和朋友交互和分享信息。目前，人人网的许多应用都涉及到大规模图处理技术，比如好友推荐、安全系统、广告推荐、新鲜事分享等。

本实验主要通过人人网安全系统中一个常用算法（TrustRank）进行系统的性能验证。人人网利用TrustRank将其用户按照可信度排序，并将可信度低的用户加入可疑用户列表，并对他们进行监控，以便及早发现其有害行为。本实验是在人人网的好友申请图上计算TrustRank。好友申请图是单向无权图，用户A向用户B发送一条申请记录，则A指向B产生一条有向边。本实验中，共收集了人人网的北京大学子图，共约20万个节点，500万条申请记录（边）。

为了有效验证系统的性能，将500万条边按照时间顺序分成两部分，前400万作为系统的初始数据，输入系统，以离线的方式计算出所有节点的TrustRank值。然后用后100万条边的数据作为模拟的图数据变化输入，即模拟后100万条边的到来，从而验证系统的实时性和计算代价。

实验在云计算平台上完成，共使用50台虚拟节点和1台中心节点，所有机器均在同一集群内，机器间以高速设备互相连接。

4.2　实时更新代价

首先，验证图的更新引起的计算是否具有局部性，且检验每次更新计算所需要的系统计算代价。实验在50台分布式节点中进行，并持续增加100万条边来模拟图的动态更新过程。图7统计了每次图发生变化时所引起的重新计算的次数以及重新计算的节点个数的累计分布曲线，$F(x)$为更新次数小于x的节点数所占比例。从图7可以看出，95%的更新引起的重新计算的周围节点不超过30个，且有77%的更新，只引起周围一个节点的重新计算。这表明，系统可以用很小的计算代价维护TrustRank实时增量更新。

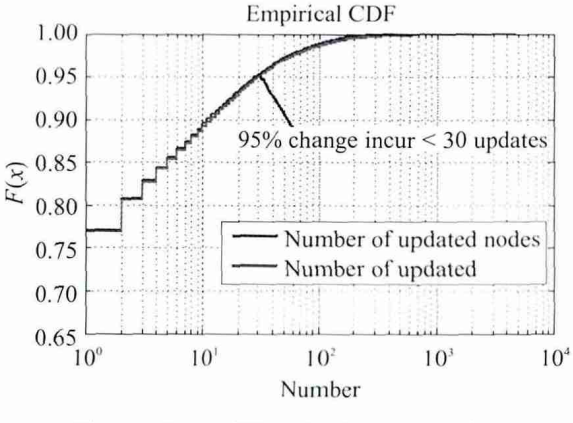


Fig.7　Cost of TrustRank computation

图7　实时计算TrustRank的代价

4.3　实时性

实时性同样是IncGraph的一个重要测评指标，对于一些有实时性要求的计算任务，其计算代价并不一定是首要考虑任务，但实时性会给结果造成非常大的影响，比如好友推荐、安全报警等。

在与上面相同的实验环境下，每一次图更新计算中，记录更新开始到所有节点完成所需要的时间，将该时间定义为系统的响应延时。图8所示为系统延时的累积分布图。

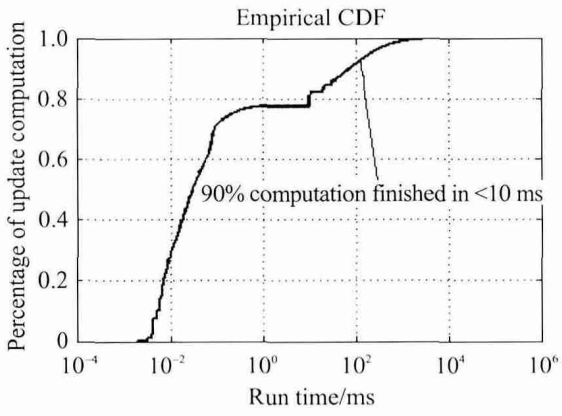


Fig.8 Instantaneity of IncGraph

图8 IncGraph系统的实时性

由图8可见，90%的更新计算可以在100 ms内完成，且大部分（约80%）的计算只需要在1 ms内完成。因此，可以认为系统的延时可控制在毫秒级别，可以满足实时性的大部分要求。

4.4 可扩展性

由于IncGraph是分布式计算系统，可扩展性是衡量系统质量的重要指标之一。本文将上述数据图采用两种常用切分方式，分别在5到50台机器上进行模拟实验。实验统计所有更新计算过程中的跨机器通信消息量，图9表示平均跨机器通信量随着机器数增长的变化。

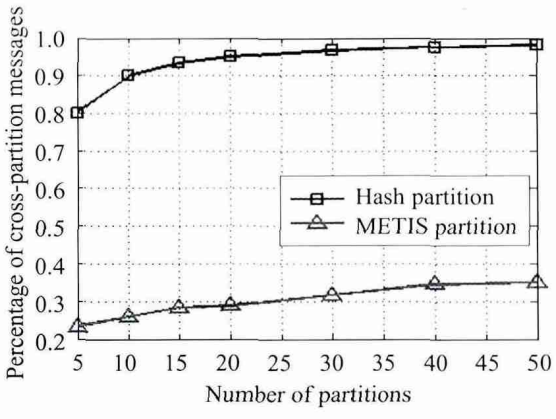


Fig.9 Percentage of cross-partition messages

图9 跨机器通信消息比例

由图9可以得知，无论是Hash切图还是METIS切图的方式，随着机器数的增加，跨机器间的通信量并没有增加很多，且低于线性增长的速度，因此可以说IncGraph具有非常好的可扩展性。还可以看出，通过METIS切图可以有效地降低机器间的通信量，平均降低68.2%。

5 相关工作

随着大规模在线社交网络等应用的快速发展，大规模图数据的快速处理成为研究的热点。本章将介绍一些关于分布式图数据处理的相关工作。

（1）大规模分布式图处理系统

首先提到的是Google在2010年提出的Pregel分布式图处理框架。Pregel系统借鉴了BSP模型[5]的基本思想，通过多个连续的超步来计算图数据，在每个超步中，每个节点并行地处理用户自定义的计算程序，并通过消息通信来传递每轮的计算结果。与Pregel类似的，有许多其他版本的GPS实现，诸如Giraph（http://giraph.apache.org/）、GPS[6]、GraphLab[7]、Picollo[8]等。Giraph是在Hadoop系统上层的抽象，通过将Pregel的每个工作节点实例化成Hadoop中的一个Mapper，避免了Mapper与硬盘之间的存储开销。类似的系统还有GraphLab发表的PowGraph系统[9]，主要解决Power-law图的计算性能问题。这些系统都是基于BSP的思想，只能进行离线批量处理。为了提高处理效率，最近一些工作[10-12]开始引入图切分和副本技术来保证图的计算局部性。然而以上系统均关注于批量图处理，对于更新和计算频繁的图应用，其计算代价较高。

（2）实时计算系统

Twitter在2011年提出了Storm（http://storm-project.net/）计算模型，并将其开源。类似系统有Yahoo! S4[13]。Storm是一套实时的流数据处理模型，其基本思想是将用户的一个计算请求抽象成一系列流水作业，将这一流水线上每一个计算程序都放在不同的分布式节点进行处理，当一个节点计算完成后，将结果传递给下一个计算节点，直到所有计算完成，最后一个节点会将结果输出。该系统不仅保证了计算结果的实时性，更节省了计算代价。但是，Storm系统只适用

于流式数据，对于基于图数据的一些复杂图算法，该类型的实时系统就无法处理。

（3）单机图处理系统

在OSDI 2012上，GraphLab发表了其处理大规模图数据的单机系统GraphChi[14]。该系统可以使用一台普通的PC机来完成大规模图数据的计算，为许多没有分布式环境的用户提供了可能。其基本思想是将大数据存储在硬盘中，用一个滑动的窗口将硬盘中的一部分图数据载入到内存中进行计算，通过多趟不断的迭代和滑动窗口，可以在较少的时间内完成图计算。另外，MSRA的Grace[15]系统也面向单机对多核计算进行优化。单机的图处理系统虽然效率较高，但其可扩展性差，对于大规模的实时图数据和频繁的更新应用均效率低下。

其他系统包括图数据库、共享内存的图处理系统等，这些系统或关注于离线处理的效率，或是优化具体的图算法，并没有关注如何实时地、增量式地处理大规模图算法。本文提出了IncGraph，能够将图算法问题域中一部分算法实现增量计算，降低离线计算的代价。

6　结束语

本文提出了一种新的图计算模型来将传统的批量图计算模型转化为增量的图计算，保证对图变化的局部处理，降低计算开销，同时实时地将图局部的更改反映在全局计算结果当中。此外，基于这种新的模型设计了大规模实时增量图处理系统IncGraph，利用了图切分技术、主动计算触发技术和反向链式更新技术等关键技术，保证了系统的高效率和低开销。未来将在系统上实现更多计算，进一步评测系统的性能。

References:

[1] Alexa top 500 global sites[EB/OL]. [2013-03]. http://www.alexa.com/topsites.

[2] Malewicz G, Austern M H, Bik A J C, et al. Pregel: a system for large- scale graph processing[C]//Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data (SIGMOD '10), Indianapolis, USA, Jun 6- 11, 2010. New York, NY, USA: ACM, 2010: 135-146.

[3] Gyöngyi Z, Garcia-Molina H, Pedersen J. Combating Web spam with TrustRank[C]//Proceedings of the 30th International Conference on Very Large Data Bases (VLDB '04), Toronto, Canada, 2004: 576-587.

[4] Karypis G, Kumar V. METIS—unstructured graph partitioning and sparse matrix ordering system[EB/OL]. [2013-03]. http://www.cs.umn.edu/~metis.

[5] Cheatham T, Fahmy A, Stefanescu D C, et al. Bulk synchronous parallel computing—a paradigm for transportable software[C]//Proceedings of the 28th Annual Hawaii International Conference on System Science (HICSS '95), Wailea, USA, Jan 3-6, 1995. Washington, DC, USA: IEEE Computer Society, 1995: 268-275.

[6] Salihoglu S, Widom J. GPS: a graph processing system[C]//Proceedings of the 25th International Conference on Scientific and Statistical Database Management, Baltimore, USA, Apr 2012.

[7] Low Yucheng, Gonzalez J, Kyrola A, et al. GraphLab: a new framework for parallel machine learning[J/OL]. arXiv: 1006.4990, 2010.

[8] Power R, Li Jinyang. Piccolo: building fast, distributed programs with partitioned tables[C]//Proceedings of the 9th USENIX Conference on Operating Systems Design and Implementation (OSDI '10),Vancouver, Canada, 2010. Berkeley, CA, USA: USENIX Association, 2010: 1-14.

[9] Gonzalez J E, Low Yucheng, Gu Haijie, et al. PowerGraph: distributed graph-parallel computation on natural graphs[C]//Proceedings of the 10th USENIX Conference on Operating Systems Design and Implementation (OSDI '12). Berkeley, CA, USA: USENIX Association, 2012: 17-30.

[10] Pujol J M, Erramilli V, Siganos G, et al. The little engine(s) that could: scaling online social networks[C]//Proceedings of the 2010 ACM SIGCOMM Conference (SIGCOMM '10), New Delhi, India, Aug 30-Sep 3, 2010. New York, NY, USA: ACM, 2010: 375-386.

[11] Mondal J, Deshpande A. Managing large dynamic graphs efficiently[C]//Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data (SIGMOD '12), Scottsdale, USA, May 20- 24, 2012. New York, NY, USA: ACM, 2012: 145-156.

[12] Yang Shengqi, Yan Xifeng, Zong Bo, et al. Towards effective partition management for large graphs[C]//Proceedings of the 2012 ACM SIGMOD International Conference on Man-

agement of Data (SIGMOD '12), Scottsdale, USA, May 20-24, 2012. New York, NY, USA: ACM, 2012: 517-528.

[13] Neumeyer L, Robbins B, Nair A, et al. S4: distributed stream computing platform[C]//Proceedings of the 2010 IEEE International Conference on Data Mining Workshops (ICDMW '10), Sydney, Australia, 2010: 170-177.

[14] Kyrola A, Blelloch G, Guestrin C. GraphChi: large-scale graph computation on just a PC[C]//Proceedings of the 10th USENIX Conference on Operating Systems Design and Implementation (OSDI '12). Berkeley, CA, USA: USENIX Association, 2012: 31-46.

[15] Prabhakaran V, Wu Ming, Weng Xuetian, et al. Managing large graphs on multi-cores with graph awareness[C]//Proceedings of the 2012 USENIX Annual Technical Conference (USENIX ATC '12), Boston, USA, 2012. Berkeley, CA, USA: USENIX Association, 2012: 4.

SHEN Lin was born in 1987. He is a master candidate at Peking University. His research interests include computer network and distributed storage system, etc.

申林(1987—),男,河南安阳人,北京大学硕士研究生,主要研究领域为计算机网络,分布式存储系统等。

XUE Jilong was born in 1988. He is a Ph.D. candidate at Peking University. His research interests include social network and distributed storage system, etc.

薛继龙(1988—),男,内蒙古乌兰察布人,北京大学博士研究生,主要研究领域为社会网络分析,分布式存储系统等。

QU Zhi was born in 1989. He is a master candidate at Peking University. His research interests include computer network and distributed storage system, etc.

曲直(1989—),男,北京人,北京大学硕士研究生,主要研究领域为计算机网络,分布式存储系统等。

YANG Zhi was born in 1982. He received his Ph.D. degree from Peking University in 2010. Now he is an assistant professor at Department of Computer Science and Technology, Peking University. His research interests include computer network and distributed storage system, etc.

杨智(1982—),男,内蒙古包头人,2010年于北京大学获得博士学位,现为北京大学计算机科学与技术系助理教授,主要研究领域为社会网络分析,分布式存储系统等。

DAI Yafei was born in 1958. She received her Ph.D. degree from Harbin Institute of Technology in 1993. Now she is a professor and Ph.D. supervisor at Department of Computer Science and Technology, Peking University, and the senior member of CCF. Her research interests include P2P computing, social network and distributed storage system, etc.

代亚非(1958—),女,黑龙江哈尔滨人,1993年于哈尔滨工业大学获得博士学位,现为北京大学计算机科学与技术系教授、博士生导师,CCF高级会员,主要研究领域为P2P系统,社会网络分析,分布式存储系统等。