PAGraph:一种面向异步计算模型的图分区方法

摘要：异步的分布式图处理框架对于迭代计算是否的高效，其性能往往优于同步的分布式图处理框架。但是当前的数据分片的方法，都是基于同步的图处理模型，而并没有一种准对于异步图处理模型而设计的分区方面，这在很大程度上限制了异步模型的高效性。本文，将基于异步图处理模型的特性（收敛速度的快慢取决于消息的疏通性），提出了一种基于流通量的以边为中心和以边点结合的分区方法 PAGraph（Partiting For Asynchronous Graph Processing Model），实现消息在高流通性的同时，又协调负载均衡、通信量、计算量等影响框架性能的因素。为实现对 PAGraph 分区数据的有效处理，基于 DAIC 及开源框架 Maiter 实现了 MR-DAIC 计算模型和 Maiter+计算框架。通过实验，可以看到 PAGraph 明显的提高了消息在各个分区中的流通性，运行时间也提高了近 *%。

关键字:异步图计算模型；流通量；PAGraph；MR-DAIC；Maiter+；

图可以表达复杂的结构和丰富的语义，其迭代分析算法在社交网络、Web、时空和科学数据计算等诸多领域都获得了广泛的应用[21-22]。大数据时代，图的规模不断增大，这对大规模图数据实现高效的迭代计算带来了巨大的挑战。为了应对这一挑战，相继出现了一些支持迭代计算的分布式图处理框架，如 Pregel[14]、Spark[13]、GraphLab[12]，Maiter[1]，PowerGraph[2] 和 GraphX[15]等。

分布式图处理框架主要分为两类：同步框架和异步框架。它们的区别在于迭代计算超级步之间是否需要一个同步操作，前者需要，而后者不需要。因此，可以总结出异步框架相对于同步框架具有以下几点特性：

1）异步框架不需要同步操作，避免了同步操作开销；

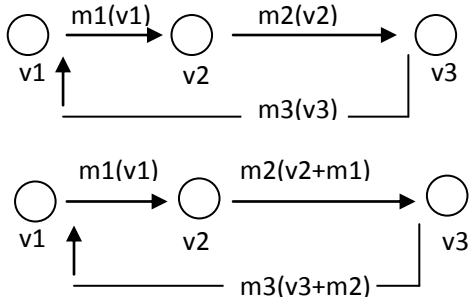
2）异步框架能够消除集群中最慢的机器对集群整体性能的影响；

3）消息传递，在同步框架中具有阻塞性，在异步框架中具有即时性，使算法获得更快的收敛速度；

4）异步框架可以实现灵活的调度策略，往往会采用本地迭代计算多次后，再与集群进行消息的交互的调度方式，减弱网络阻塞带来的影响。

综上，在迭代计算中异步框架相对于同步框架往往会具有更高的性能，因此对异步图处理框架进行研究或许是实现大规模图数据高效计算的一个突破口。

在对海量的数据进行分布式迭代计算之前，图划分是必须的一步。由于图计算通常按照拓扑结构访问数据， 所以每次迭代处理均会引入巨大的通信开销， 这成为制约分布式处理性能的关键因素[16]。同时，集群的负载是否均衡也会直接影响分布式图处理框架的性能[17]。图划分中既要实现通信量，又要实现负载均衡，因此图分区算法是一个经典的 NP-Compete 问题[18] 。但是，由于图划分对迭代计算的重要性，图分区方法仍然得到了广泛的关注和研究。



在同步框架中，消息的传递过程是传递一跳，同步一次，如图 1（a）所示。在异步框架中，消息的传递没有阻塞，即到即用，如图 1（b）所示，如果采用本地迭代计算多次再通信一次的调度策略，那么消息的流通性的重要性将会成倍的提高。在异步框架中具有较好的消息流通性，将会使算法获得更快的收敛速度。

但是，当前的图分区方法通常只会权衡通信量和负载均衡这两个因素来实现对图的划分，而没有考虑异步框架中消息传递的特性。对图 G，现分别使用随机 hash 顶点的分区方法（Edge-Cut）、PowerGraph 中的贪婪分区方法（Vertex-Cut）和考虑流通性的分区方法实现对图 G 的划分，如下图 2 所示。

从图 2 中可以看到，考虑流通性的分区方法获得了最好的消息流通性。基于以上分析，本文提出一种基于边的流通量的分区方法 PAGraph。本分区方法综合消息的流通性、通信量、负载均衡三个因素的权衡实现对图的划分，提高异步图处理框架的性能。

本文中，选择异步计算模型 DAIC 作为支持 PAGraph 分区方法的计算模型，来最大程度的体现消息的流通性。综上，本文主要贡献总结如下：

1）对同步的图计算模型和异步的图计算模型进行了研究和分析，发现消息的流通性在异步计算模型中的重要性。

2）基于边的流通量，提出了一个针对于异步图处理框架的图分区方法 PAGraph。

3）改写了 DAIC 计算模型中的通信模型，并基于 Maiter 框架实现了支持对多副本顶点的异步图处理框架 Maiter+。

4）通过实验，说明了 PAGraph 图分区方法的高效性。

本文第 1 节介绍相关工作，第 2 节介绍基于流通量的分区方法 PAGraph 和优化计算负载后的 PAGraph，第 3 节实现支持 PAGraph 分区方法的开源框架 Maiter+，第 4 节展示实验结果和分析，最后第 5 节总结全文。

1 相关工作

本节，将简要的介绍一种高效的异步框架 DAIC（delta-based accumulative iterative computation）；介绍和分析当前几个典型的图分区方法，并说明它们在体现消息流通性方面的缺乏。

1.1 DAIC 计算模型

使用传统的迭代计算求解问题，是通过不断的用上一次迭代计算的结果去更新当前值，而 DAIC 计算模型是基于上一次结果的变化量来更新当前的结果，同时基于 DAIC 计算模型的四个约束条件[]，对消息达到的次序没有要求（基于交换律），这就形成了 DAIC 计算模型的异步特性。下面以 PageRank 为例，详细的说明 DAIC 计算模型及其原理。

更新函数：

$$R_j^k = d \cdot \sum_{\{i|(i\to j)\in E\}} \frac{R_i^{k-1}}{|N(i)|} + (1-d),$$

传统计算模型

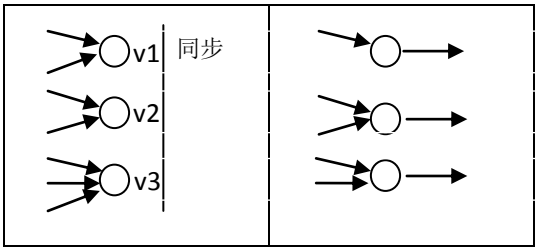
$$v_j^k = g_{\{1,j\}}(v_1^{k-1}) \oplus g_{\{2,j\}}(v_2^{k-1}) \oplus \ldots \oplus g_{\{n,j\}}(v_n^{k-1}) \oplus c_j$$

DAIC 计算模型

$$\begin{aligned} v_j^k = & g_{\{1,j\}}(v_1^{k-2}) \oplus g_{\{1,j\}}(\Delta v_1^{k-1}) \oplus \ldots \oplus \\ & g_{\{n,j\}}(v_n^{k-2}) \oplus g_{\{n,j\}}(\Delta v_n^{k-1}) \oplus c_j. \end{aligned}$$

又因为运算符满足交换律和如下的计算公式，因此消息的传递没有任何的同步操作。

$$\begin{cases} v_j^k = v_j^{k-1} \oplus \Delta v_j^k, \\ \Delta v_j^{k+1} = \sum_{i=1}^{n} \oplus g_{\{i,j\}}(\Delta v_i^k). \end{cases}$$



通过上图，看出传统的计算方式，需要进行一次全局的同步后（收到 m1-m7），消息才能被使用和传递。DAIC 计算中，消息既不需要全局顶点的同步也不需要顶点本身接受消息的同步（GraphLab 需要），消息到达就可以被立即的使用和传递。

1.2 图分区方法

对图数据进行分布式的处理分为两个阶段：数据分区和迭代计算。数据分区在减少通信量、保证负载均衡、提高消息的流通性方面扮演着一个至关重要的角色，同时它也被公认为是 NP-hard 问题，近来特别是近两年对图分区算法研究变得越来越得到科研工作者的关注。按照分区粒度的不同，现有的分区方法可以主要分为两类：Vertex-Cut 和 Edge-Cut。

Edge-Cut：以点为数据分区的粒度，对图中的边进行切割，是最直观也是最容易实现的方式。其中 hashing 方法最为流行，近两年之前几乎所有的图处理框架都采用这种分区方法。这种方法，它对输入的每个顶点 v，h(v) mod |P|获得顶点 v 的目标分区。这种方法可以被很容易的实现，但是会切割

大量的边导致很高的通信量。另一类就是贪婪性的分区方法，Fennel [5]是其中一个性能不错的方法。对于输入的一个顶点 v，根据它目的顶点在当前分区中的个数来决定顶点 v 的目标分区。它相对于 hashing 实现有些复杂，但是大大降低了切割边的数量。

Vertex-Cut：在 Edge-Cut 中分区方法，能够保证每个分区上的顶点个数是均衡的，表面上是实现了各个分区之间的负载均衡。在图顶点的度分布均衡的图数据中这种均衡是成立，但是在顶点的度分布不均衡（skewed Graphs）的情况下上面的负载均衡是不成立的。实际上，计算的负载、通信负载、存储负载都跟边的数量呈线性关系，因此 Edge-Cut 分区方法在顶点度分区扭曲的图结构中表现的很差，而 Vertex-Cut 分区方法被广泛的采用来解决这一问题，实现分区之间的负载均衡。

PowerGraph 中，提出一种随机 hash 和贪婪分区两种方法，这两种方法都有效的解决了负载均衡的问题。后者相对于前者，可以有效的减少顶点副本的数量，减少在迭代计算过程中的计算量。

PowerGraph 的分区方法相对 Edge-Cut 方法既实现了负载的均衡，也减少了通信量，但是它产生了大量的计算顶点，增加了计算开销。切割顶点相对于切割边更容易将图分割，同时切割度高的顶点相对于切割度低的顶点也会更容易的将图分割[6, 7, 8, 9]，因此在 DBH[11]、HDRF[3]、PowerLyra[4]等分区方法中都侧重于切割度高的顶点。DBH 和 HDRF 这两种方法对一条边进行切割时，会优先选择将高度顶点切割，但是效果并不是很有效。在 PowerLyra,它采用先对低度顶点以顶点为粒度，在对高度顶点以边为粒度进行分区的策略，这种方法需要一次预处理操作，不能对流数据进行直接的操作。

下面再来看，流通性在以上各个分区方法中的体现。在 Egde-Cut 的贪婪性分区方法中，为了减少通信量，它会选择关系紧密的一些点放在一个分区中，但是这对于体现流通性是粗粒度的一种分区方法。在 Vertex-Cut 的贪婪性分区方法中，虽然它们实现了以边为粒度分区，但是在体现流通性上是不全面的，下面以 PowerGraph 中贪婪分区方法为例，对消息流通性进行分析。

给定一条 e∈E，源顶点为 s，目的顶点为 d（A(v)表示顶点 v 当前分配到的分区集合）：

Case 1：如果 s、d 都没有被分配过，那么从所有分区中选择负载最小的那个分区作为 e 的目标分区。

Case 2：如果，s 或 d 中只有一个顶点被分配（假设 s 被分配到了分区集合 A（s）中），那么从 A(s)中选择负载负载最小的分区作为 e 的目标分区。

从体现流通性角度来看，e 被分配 A(s)中的不同分区往往会有不同的流通性，如果 Pi∈A(s)，且 s 在 Pi 的分区的入边的数量最多，那么 e 被分配到 Pi 会使消息获得最好的流通性。

Case 3：如果 s、d 都被分配且 A(s)∩A(d)为空集，则会从集合 A(s)∪A(d)中选择负载最小的分区作为 e 的目标分区。

从体现流通性角度来看，从 A(s)选择 s 入边数量最多，从 A(d)中选择 d 出边数量最多所在的分区作为目标分区，将会更有利于消息的流通。

Case 4：如果 s、d 都被分配且 A(s)∩A(d)不为空集，则会从集合 A(s)∩A(d)中选择负载最小的分区作为 e 的目标分区。

从体现流通性角度来看，从集合 A(s)∩A(d)中选择消息流通性最好的分区作为目标分区，将会更有利于消息的流通。

以上的三种方法都是从直观上选择度高的顶点来切割，而不切割度低的顶点，这是不严格的，一种粗滤的方法。实际上，一个顶点是否应该被切分，取决于切割它带来的计算开销与通信量、流通量的权衡。同时以上的这些分区方法，都没有充分考虑消息的流通性，抑制了异步计算模型的优势，基于以上原因，本文中提出了基于消息流通性，一个针对于异步计算模型的图分区方法 PAGraph。

1.3 相关术语定义

顶点 v，在以边为粒度的分区方法分割后在各个分区的分布如下图：

图展示各中顶点

master 顶点：副本顶点中的一个，负责各个副本顶点之间消息的传负责顶点内部各递。接收各个 mirror 顶点的消息，并将其转发给相应的 mirror。

mirror 顶点：master 顶点以外的所有副本顶点，负责收集本地消息，并将其发送给 master 顶点，同时也接收 master 发送过来的消息。

计算顶点：具有出边信息的顶点，既要负责相关的计算任务，也负责消息的收集和转发。

通信顶点：没有出边信息的顶点，只是负责消息的收集和转发，没有计算任务。

在本文中，master 顶点一定是计算顶点也是一个 mirror 顶点。有出边的 mirror 顶点是计算顶点，而没有出边的 mirror 顶点是通信顶点。

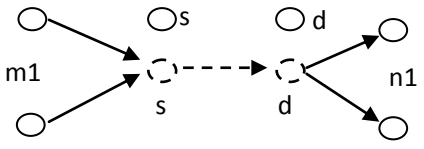
2 分区方法

异步图计算模型中的迭代计算是一个顶点之间相互影响直到各个顶点值收敛的过程。在这个过程中，顶点通过它的边传递消息给它的邻居顶点并对其产生作用，单位时间内流过边的消息量越大，那么迭代计算也就会得到更快的收敛。边是消息流通的最基本单位，因此本文中的 PAGraph 就是一种基于流通量以边为粒度的分区方法。

本小节，将详细描述边的流通量的含义和两个版本的 PAGraph，一种基于流通量以边为粒度的分区方法和一种改进型的以边点结合的分区方法。

2.1 边的流通量

流通性是表示单位时间（一定的迭代次数）内顶点之间通过边，交换消息的多少。本文中为了在分区过程中体现消息的流通性，引入一个概念--边的流通量。在分区过程中，计算待分配边在各个分区中流通量，流通量值越大说明该边在该分区中的流通性越好。



现对边(s,d)进行分配，s 会通过它的入边和 s 的副本顶点接受消息（in_messege），然后再由 d 通过它的出边和 d 的副本顶点将消息发送出去（out_messege），那么就将 out_messege 称为边(s,d)的流通量。

in_messege=m1/s.indegree + [(s.indegree-m1)/s.indegree]*(1/net_cost)

out_messege= in_messege*[n1/d.outdegree + [(d.outdegree-n1)/d.outdegree]*(1/net_cost)]

其中 m1 是指向 s 的顶点在本分区中的数量，s.indegree 是 s 顶点的入度；n1 是 d 指向的顶点在本分区中的数量，d.outdegree 是 s 顶点的出度；net_cost 是网络通信的开销。s.indegree 和 d.outdegree 的值需要对数据进行一次遍历才能得到，为了避免预处理的开销，本文使用已经分配数据的信息获取顶点的局部度数，来近似等于顶点真正的度数[3]。

2.2 分区方法 PAGraph

在分布式计算环境下，消息的流通性将直接影响迭代计算的收敛速度，因此最大化消息的流通性成为异步计算模型分区方法所考量的指标。

边是消息流通的基本单位，因此采用以边为粒度的分区方法可以最大程度的最大化消息的流通性。在以边为粒度的 PAGraph 中，给定一条边 e(s,d)，计算边 e 在各个分区中流通量的值，选择流通量最大的分区作为边 e 的目的分区。同时考虑到边的通信量、负载均衡对迭代计算过程中的影响，制定了如下的启发式分区函数：

score[s,d]=

a*score[transitivity]+b*score[communication]+c*score[load]+d*score[prio]

score[s,d]表示 e 在某一分区 P_i 中的得分，其中，a,b,c 是描述各个项重要性的系数。对于流通量的分值，它是通过 2.1 中边的流通量计算公式计算得到。对于通信量项的分值计算，判断源顶点 s 和目的顶点 d 是否存在于分区 p 中，存在则得 1 分，取值范围是{0,1,2}。

对于负载均衡项分值计算，其计算公式如下：

socere[load]={(load_max-load_i)/(constant+load_max-load_min)}
constant 是一个常数，load_max，load_min 是当前所有分区中负载的最大值和最小值，load_i 是当前计算分区 P_i 负载，取值范围（0,1）,该数据分片在集群中的负载相对越小,那么该分片的 s 就会越大。
对于优先级项，它用于实现倾向于切割高度点的策略，计算公式如下（以 s 顶点为例）：

本文中对于分区 P_i 的负载如下定义：
Load[P_i]=vertics[P_i]+ratio*edges[P_i],
vertics[P_i]表示分区 P_i 的顶点数量，edges[P_i]表示分区 P_i 中边的数量，ratio 是图数据中顶点数量与边的数量的比值。

2.3 优化计算负载的 VEPAGraph

2.2 小节中实现的分区方法在消息的流通性、通信量、负载均衡等方面都得到了不错的效果，但是它引入了大量的计算顶点，导致计算量大量增加。特别是在边的密度不大的图数据中，这将会大大的降低分布式图处理框架的性能。

在 PowerGraph 的分区方法中也同样存在着这种问题，Powerlyra 为解决这一问题，对低度顶点采用 Edge-Cut 方法，对高度顶点 Vertex-Cut 的方法，减少分区中计算顶点的数量。对于采用广度优先遍历的边输入顺序，本文提出了以边点结合的 PAGraph 分区方法，采用了更加灵活、定量计算的方式来决定对计算顶点的生成。读入一个顶点及其出边，首先进行一次预分区，按照 2.2 中方法对每条进行分区，然后根据预分区的结果计算每个分区的得分，并将该分值与一个描述计算负载的阈值比较，如果大于阈值，预分配到该分区中边被真正的分配，如果小于阈值，则对该边进行重新分配。下面将详细描述该过程。

给定一个顶点 source 及其目的顶点列表{des1,des2,...,desn}：

1)先计算每个分区公共部分的得分，包括该分区的负载均衡得分，source 顶点副本的得分和计算负载得分。公式
score[计算负载]={threshhold source 是计算顶点,0 source 不是计算顶点}

2）为每条边计算私有部分的得分，包括目的顶点的副本得分和流通量得分，然后加上公共部分得分，为每条边选择出最佳的预分区。公式

3）根据预分区结果，计算每个分区的得分 score，分区的公共部分得分+预分配的到该分区中每条边的私有的得分。,score[part]大于 threshhold[增加一个计算顶点带来的开销]，则该分区中的边分配成功,如果小于 threshhold[计算负载]的分区,则执行步骤 4)。如果,最大的 score 还是小于 threshhold 那也要将最大的这个分片，保证分区能够进行下去。

4)将分配失败的分区中的边,进行重新的分配,回到步骤 1)。

在对图进行分片切割时,切割高度顶点会切割更少的顶点,更容易的将整个图分割,所以本文中也采用类似于了 PowerLyra 中先分配低度顶点再分区高度顶点的分区策略,不同的是，PowerLyra 中对低/高度顶点需要执行两种不同的操作、实现较复杂，且它对于低度顶点的分区是以顶点为粒度；而在 PAGraph 中对低/高度顶点都使用一种方法易于实现，且它是以边为粒度，能够进行更精确计算和定量计算。同时，PAGrahp 也可以不区分低度高度顶点直接对数据进行分区，这样可以减少预处理操作，同时也对分区效果有一点影响。

3 计算模型

本小节，将对多副本顶点的 DAIC 计算模型(MR-DAIC)和不同顶点采用的不同处理方式进行描述。

3.1 MR-DAIC 计算模型

MR-DAIC 像其他的 vertex-program 的图处理模型一样，都是并行的计算每一个顶点 v∈V，完成对整个有向图的处理。不同的是，在多副本的图结构中，每个分区都构成了一个完整的子图，网络通信发生在顶点内的副本之间。在这种图结构中，所有的顶点都被称为副本，为了实现副本之间高效、正确的信息交换，会选择其中一个副本作为 master 顶点，其余副本都被称为 mirror 顶点。在 PowerGraph、PowerLyra 中，它们都是随

机的选择一个副本作为 master 顶点，而在 Maiter+中为了提高消息的流通性，会选择出度最高的副本作为 master 顶点。对于任意一个在分区 p 上的顶点 Vj，MR-DAIC 计算模型描述如下。

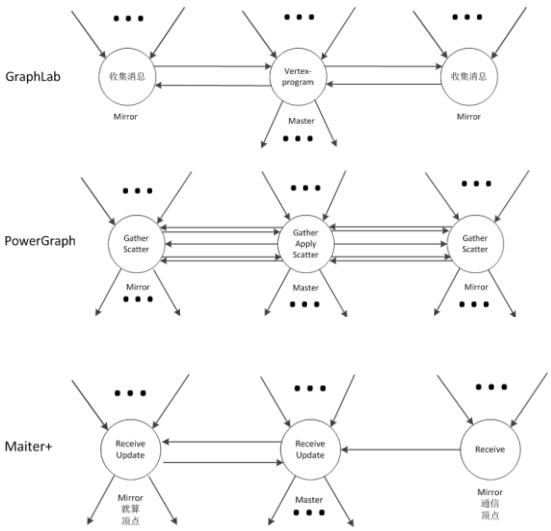
对于本地消息-LM:

receive: 每当顶点 j 收到消息 LM(j,delta),则
delta(j)+= delta;
update:
if(j 是 master 顶点){
  value(j)+= delta(j)
  for j_mi in mirrors{
    send NM(p,j_m,j,delta(j)-offset[j_m]);
  }
}else if(j 是 mirror 顶点){
  send
  NM(p,master,j,delta(j) -offset[j_m]);
}
for d in link{
  send LM(d,g{j,d}(delta(j)));
}
Clear(delta(j));
Clear(offset[]);

对于网络消息-NM:

receive: 每当顶点 j 收到消息 NM(s_part,d_part,j,delta),则
delta(j)+=delta;
offset[s_part]+=delta;
update:
if(j 是 master 顶点){
  value(j)+=delta(j);
  for j_mi in mirrors{
    send NM(p,j_m,j,delta(j)-offset[j_m]);
  }
}
for d in link{
  send LM(d，g{j,d}(delta(j)));
}
Clear(delta(j));
Clear(offset[]);

LM (j, delta)：表示本地消息，j 表示目的顶点，delta 为传送的值；NM(s_part,d_part,j, delta)：表示网络消息，part 表示分区号，j 表示目的顶点，delta 表示传送的值；g{j,d}(delta)用于计算顶点 j 传递给顶点 d 的值；Clear()用于将 delta 值清空到默认值。



3.2 不同顶点的计算

Master 顶点与 Mirror 通信顶点：对于这两种顶点之间的处理，本文采用类似于在 GraphLab[12]中计算模型。在 MR-DAIC 计算模型中，Mirror 通信顶点只进行 receive 操作，负责收集本地的消息，并将该消息发送给 master 顶点。Master 顶点执行 receive、update 操作：收集本地消息和来自 mirror 的消息，执行更新 value 值的操作。在这种处理方式中，mirror 通信顶点只需要执行 receive 操作，降低了计算开销；mirror 顶点和 master 顶点之间的通信是单向的，master 顶点无需给 mirror 顶点发送消息，降低顶点之间的通信量。

Master 顶点与 Mirror 计算顶点：对于这两种顶点之间的处理，本文采用类似于在 PowerGraph 中 GAS 计算模型。在 GAS 中各个顶点并行执行 Gather 操作，之后 mirror 将消息汇总到 master，经过一次同步操作后 master 顶点执行 Apply 操作，将结果传递到 mirror 后，各个顶点再并行执行 Scatter 操作。在 MR-DAIC 中，各个顶点并行的执行 receive、update（mirror 顶点不执行更新 value 的操作），没有任何的同步操作。副本之间消息的交互完全独立于上述的操作，mirror 计算顶点将本地收集的消息发送给 master 顶点，master 顶点将消息汇总后在转发给相应的 mirror 顶点。可以看到，MR-DAIC 相对于 PowerGraph 中的 GAS，副本顶点之间不需要任何的同步操作，减少了同步开销提

高了计算的并行性，同时也看到通信量也得到了明显减少。

4 实验

为了验证 PAGraph 对消息流通性及收敛速度的提高，本小节将用 PAGraph 分区方法同其他几个典型的分区方法进行比较，比较它们在消息流通性方面的差异和运行时间的快慢。

5 总结

对异步计算模型（特别是 DAIC）和同步计算模型进行研究，发现了消息流通性在异步计算模型中特殊性，对收敛速度的影响。本文，基于边的流通量提出了一种针对于异步计算模型的图分区方法 PAGraph。基于异步计算模型 DAIC 实现了支持多副本的 MR-DAIC，并基于 DAIC 的开源实现框架 Maiter 实现了支持 MR-DAIC 的分布式图处理框架 Maiter+。

同时，现实世界中图数据的顶点的度服从扭曲的分布（如幂率分布），使得现有的一些分区方法产生的分区出现负载不均衡等问题，本文的分区方法 PAGraph 也很好的解决了这些问题。

下一步工作，本文中的分区方法采用启发式函数确定分区，只需对数据遍历一遍，时间复杂度为 o(n)，目前分区方法是使用单机实现，下面的工作将实现分布式版本。启发式函数各个项的影响因子在本文中缺乏深入的分析，后面的工作将会对此进行研究和对分区方法的优化。

参考文献

[1]MaiterProject,http://code.google.com/p/maiter, 2013-07-15.

[2]Gonzalez. J, Low. Y, Bickson. Danny, Guestrin. Carlos. PowerGraph: Distributed Graph-Parallel Computation on Natural Graphs//10th USENIX Symposium on Operating Systems Design and Implementation (OSDI ’12), 2012:22-25.

[3] Fabio Petroni，Leonardo Querzoni，Khuzaima Daudjee. HDRF: Stream-Based Partitioning for Power-Law Graphs. CIKM’15, October 19-23, 2015

[4]Y.C.R.Chen, J.Shi and H.Chen. **Powerlyra**:Differentiated graph computation and partitioning on skewed graphs. In Proceedings of the 10th ACM SIGOPS European Conference on Computer Systems, 2015

[5]C. Tsourakakis, C. Gkantsidis, B. Radunovic, and M. Vo-jnovic. **Fennel**: Streaminggraph partitioningformassive scale graphs. In WSDM, pages 333–342, 2014.

[11]DBH C. Xie, L. Yan, W.-J. Li, and Z. Zhang. Distributed power-law graph computing: Theoretical and empirical analysis. In Advances in Neural Information Processing Systems, 2014

[12]Low. Y, Bickson D, Gonzalez J, et al. Distributed **GraphLab:** a framework for machine learning and data mining in the cloud //Proceedings of the VLDB Endowment, 2012, 5(8): 716-727.

[6] D. S. Callaway, M. E. Newman,S.H. Strogatz,and D.J.Watts.Network robustness and fragility:Percolation on random graphs. Physical review letters, 85(25):5468,2000.

[7] R. Cohen, K. Erez, D. Ben-Avraham, and S. Havlin. Resilience of the internet to random breakdowns.Physical review letters, 85(21):4626, 2000.

[8] R. Cohen, K. Erez, D. Ben-Avraham, and S. Havlin.Breakdown of the internet under intentional attack.Physical review letters, 86(16):3682, 2001.

[9] S. N. Dorogovtsev and J. F. Mendes. Evolution of networks.Advances in physics, 51(4):1079–1187, 2002.

[10] Asyn-SimRank:一种可异步执行的大规模 SimRank 算法

[13] Spark

[14]Pregel

[15]GraphX

[16] OnFlyP：基于定向边交换的分布式在线大图划分算法。王志刚 谷 峪 鲍玉斌 于戈

[17]KARYPIS,G.,AND KUMAR, V. Multilevel k-way partitioning scheme for irregular graphs. J. Parallel Distrib. Comput. 48, 1(1998), 96–129.
[18]
[19] PELLEGRINI, F., AND ROMAN, J. Scotch: A software packagefor static mapping by dual recursive bipartitioning of process and architecture graphs. In HPCN Europe (1996), pp. 493–498.
[21] YuGe, GuYu, Bao Yu-Bin, Wang Zhi-Gang. Large scale graph data processing on cloud computing environments. Chinese Journal of Computers, 2011, 34(10): 1753-1767(in Chinese)
(于戈，谷峪，鲍玉斌，王志刚. 云计算环境下的大规模图数据处理技术. 计算机学报，2011,34（10）：1753-1767)
[22] Li Xian-Tong，Li Jian-Zhong，Gao Hong. An efficient frenquent subgraph mining algorithm. Journal of Software, 2007，18(10):2469-2480(in Chinese)
（李先通，李建中，高宏. 一种高效频繁子图挖掘算法. 软件学报，2007，18(10):2469-2480）