****

硕士学位论文开题报告及论文工作计划书

|  |  |
| --- | --- |
| 学 号： | 1401394 |
| 姓 名： | 李玉国 |
| 导 师： | 张岩峰 |
| 学科类别： | 工学 □全日制专业学位 |
| 学科/工程领域： | 计算机应用技术 |
| 所属学院： | 信息科学与工程学院 |
| 研究方向： | 云计算，大数据处理 |
|  |  |
| 拟选题目： | 以边为中心的异步分布式图处 |
|  | 理模型与框架研究 |
| 选题时间： | 2015年 6月 1日 |

**东北大学研究生院**

**2015年 7月 17日**

填表说明

1、本表应在导师指导下如实填写。

2、学生在开题答辩前一周，将该材料交到所在学院、研究所。

3、按有关规定，没有完成开题报告的学生不能申请论文答辩。

4、全文正文均用小四号宋体，单倍行距，段前段后间距为0，如果页数不够，可以整页扩页，其他格式要求参见《东北大学硕、博士学位论文格式》。

一、前期工作基础（本节可以整页扩页）

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 课程学习及选题开题阶段，在导师指导下从事研究工作总结（不少于2000字）  自从2014年9月份开学以来，在研究方向方面的学习已经进行了将近一年。在这一年的时间里，在导师张岩峰副教授的指导下，在云计算、分布式计算、数据处理方面进行了一系列的学习和研究，收获很大，下面是我对研究工作的总结。  **1** 课程学习  在研究生的课程学习中，认真学习并完成了学位课程，又根据自己的兴趣和今后研究的需要，选修了一系列的相关课程。在学位课程的学习中，完成的课程有应用数理统计、随机过程、分布式协同计算、新一代互联网技术、英语等。根据课程跟自己研究的相关程度，重点学习了应用数理统计、随机过程和分布式协同计算这几门课程，并取得了良好的成绩。在选修课中，根据自己的兴趣和研究需要，选修了分布式数据库，分布式操作系统，高级人工智能等课程，分别涉及到了分布式数据存储原理、分布式系统结构、智能算法设计。这些课程对云计算、分布式计算、大数据处理方面的研究起到了基础性的作用。通过这些课程的学习，了解了很多云计算、分布式计算、大数据更深层次的基础知识，扩展了我的视野，对于其起源、发展、研究前沿有了一定的了解。  **2** 组会学习  研究生的培养目标就是使硕士研究生具有一定了创新能力和在某一领域取得一定学术成果，组会学习成为了我研究生阶段学习的最重要组成部分之一。组会的主要内容是：课题组的所有同学，每周固定时间组会，一起研读一篇研究方向内的比较新、比较有影响力的论文，由一位同学主讲。在组会的过程中，先听取主讲人对论文的讲解，然后听众与主讲人进行问题的讨论和交流。通过组会的学习，我了解到目前国际上在云计算、分布式计算、大数据存储、分布式算法等方面的研究成果，使自己的研究紧随研究前沿，更加有助于自己做出更有前瞻性的创新。  组会的论文选择方面，主要是一些顶级会议、期刊的高水平论文，紧紧地围绕着云计算、分布式计算、大数据处理方向。在此过程中了解了主流的分布式处理框架如Hadoop、Spark、Pregel、GraphLab、Giraph、PowerGraph、GraphX等系统的原理、实现以及优化。特别是Hadoop框架，Hadoop处理框架是目前应用的最广泛也是最通用的框架，有大量的论文都集中精力于Hadoop框架的优化工作，如Hadoop框架的负载均衡问题、容错问题、HDFS的小文件存储，除此之外，还有的工作对Hadoop框架的任务调度方式进行了优化，以更快的支持分布式迭代计算，典型的改进框架如Haloop、iMapReduce等等。在组会的过程中，还了解了一些应用广泛的图处理算法，如PageRank算法、SimRank算法、HITS算法、分布式单源最短路径算法等等，这些算法都是典型的分布式图处理迭代算法，由于这些算法具有广泛额代表性，很多的分布式框架均以尽可能的高效地实现这些算法为目标。通过学习这些算法的基本计算原理、过程，不但了解了分布式算法的特点，也了解了在设计和优化分布式计算框架时应该注意的问题。  下面是我在组会中汇报过的论文：   1. PageRank算法原理与实现 2. Mining Social-Network Graphs [Book:Mining of Massive Datasets] 3. Scalable Nonnegative Matrix Factorization with Block-wise Updates.[ECML PKDD’14] 4. GraphX: Graph Processing in a Distributed Dataflow Framework [OSDI’14] 5. PowerGraph:Distributed Graph-Parallel Computation on Natural Graphs [OSDI’12]   通过组会，学习了很多云计算、分布式计算、大数据方面的前沿知识，对于专业方向方面的研究起到了提升的作用。  **3** 参与项目  参与项目是研究生最重要的部分之一，组会学习主要是学习理论思想，只有将理论知识付诸到实践，通过实践来验证和改进，才能将学习融会贯通。在进入研究生阶段之初，就参加了张岩峰副教授的自然科学基金项目“云环境下基于BSP模型的大规模不动点迭代计算研究”。项目的主要工作是采用有效的方法提升大规模分布式图处理框架的性能。  首先，我们提出了DAIC（Delta Based Incremental Computation）计算模型，DAIC计算模型主要针对于一些特定的图处理迭代算法，采用累积差值的迭代方式：首先迭代的计算差值，然后将这些差值累积起来得到最终的结果，由于收敛的图顶点的迭代差值一定为零，所以可以在计算过程中剔除掉这些图顶点，有效的避免了已经收敛的图顶点继续参加运算，从而降低了整个分布式框架的计算量和通信量，另外，在DAIC计算模型基础之上，可以很容易的异步计算，使集群中的各个计算节点完全自治，无需任何同步过程，节省大量的同步开销，进而有效的提升了迭代算法的迭代效率。  其次，我们提出了优先级迭代计算。在DAIC计算模型的基础之上，我们发现，总是优先的调度迭代差值较大的图顶点参加运算，总是能够获得较快的收敛速度。我们也在理论上证明了这种方法的有效性。对于实际的社交网络，总是会出现80-20现象：少部分的人与特别多的人有关系，比如少数的明星，但是大部分人只是和自己圈子内的人有关联。这种现象就会导致处理这些数据的算法收敛不均匀，少部分顶点决定全局收敛速度。使用优先级迭代方法，可以优先的计算慢收敛的顶点，以到达均匀收敛的目的，因此优先级迭代计算有效的增快了迭代算法的收敛速度，进而提升了计算效率。  最后，我们实现了大规模异步图处理框架Maiter来支持DAIC计算模型和优先级迭代计算。Maiter框架采用C++实现，应用MPI通信协议，相对于Java语言和socket通信具有一定的优势。另外有效的支持异步计算、优先级迭代计算，使Maiter框架在计算性能上占有绝对的优势，对相关算法的实现，其性能优于目前主流的高性能图处理框架如Spark、GraphLab等。  在参与项目的过程中，我的主要工作是维护、优化、扩展Maiter框架。在维护方面，主要的工作是维护Maiter框架的Google Code主页，撰写一些关于Maiter原理、集群配置、使用方面的英文文档。在扩展方面，丰富了Maiter框架的API，使得Maiter框架可以支持交替计算的迭代算法，支持解决迭代计算中的TopK问题。在优化方面，参与了Maiter负载均衡处理的设计与实现。下面将自己负责的两项工作的研究和参与的一项研究中自己具体负责的部分工作进行一个简要的描述。  3.1 基于Maiter对迭代计算中TopK问题的解决  3.1.1TopK问题  在大规模数据处理中，常遇到的一类问题是，从海量数据中找出最大的前K个数，或者从海量数据中找出，出现频率最高的前K条记录，这类问题通常称为“TopK”问题，如：在搜索引擎中，统计搜索最热门的10个查询词；在歌曲库中统计下载率最高的前10首歌等等。  针对TopK这类问题，通常比较好的方案是：先将数据集按照hash方法分解成多个小数据集，然后使用trie树或者hash统计每个小数据集中的query词频，之后用小顶堆统计出每个数据集中出频率最高的前K个数，最后在从各个子数据集的TopK中求出最终全局的TopK。对于处理文本数据，获得其中的TopK信息，基本上按照上文提到解决方案来实现就可以了。但是对于大规模图数据，大多说情况下，我们需要对图数据进行不断的迭代计算，才能逐渐得到不断精确TopK信息，因此仅仅利用上述的解决方案是不够的。  3.1.2 迭代计算中TopK问题的现状研究  在MapReduce中，为了简化容错，每个MapReduce任务和工作的输出都在其被消耗前实体化到硬盘，因此对于迭代计算是非常低效的。为了提高系统利用率和快速得到在迭代计算中TopK信息，提出了一种改进的MapReduce体系--MapReduce Online，允许数据在操作之间用管道传送。在MapReduce Online中，下游数据元素可以在producer元素完成执行前开始消耗数据，这可以增加并行机会、提高利用率、减少响应时间。另外，由于mappers一产生数据后reducers 就开始处理，它们可以在执行工程中生成并改善其最终结果的近似值。简言之，MapReduce Online的思路就是允许部分数据先计算，从而估算出当前TopK信息；同时伴随着迭代计算的进行，前面得出的近似结果将会被不断的修正。可以看出，面对大规模图迭代计算的TopK问题，**其基本的解决思路是：基础的分治思想+迭代计算中异步化。**  3.1.3基于Maiter解决迭代计算中TopK问题的可行性研究  Maiter框架所实现的计算形式就是DAIC计算模型，也可以说DAIC计算模型是Maiter框架的精髓所在。DAIC计算模型创新的采用了累积迭代的计算方式。累积迭代的好处在于，每次只计算增量，但是那些已经收敛的图顶点的增量为零，所以可以让这些点不在参加计算，节省了计算和通信开销。基于DAIC计算模型的特性，实现了部分分布式迭代算法的异步计算，使得这些算法的计算不在需要同步过程，大大的提高了计算效率。DAIC计算模型在实现了异步计算的同时，DAIC计算模型实现了分布式迭代算法的优先级迭代计算，可以动态的决定每个顶点的计算优先级，让那些对全局收敛更重要的顶点优先的计算，极大的提高了分布式算法的收敛速度，进而提高整体的计算性能。  Maiter框架本身就是一个分布式图处理框架，所以可以很容易实现的对TopK任务的分治处理。Maiter本身又是一个支持异步计算的框架，同时也支持优先级的调度计算，因此它可以最大限度的利用最新的和最有利于算法收敛的数据信息。除此之外，Maiter本身还具有一些其他的优秀特性，例如差值累积的计算方式，可以有效的减少通信量和避免一些无用的计算。综合以上的分析，异步图处理框架Maiter非常适合用来解决TopK问题，其性能相对于MapReduce Online中的TopK将会有一个极大的提升  3.1.4 基于Maiter的解决方案  **（1）任务的分治化设计方案**   1. **分解：**一个大规模的数据集，通过Maiter的partition过程被均匀的分配到各个 worker上。 2. **解决**：每个worker周期的检测本地的数据，获得本地（局部）的TopK，并向master发送本地TopK信息。 3. **合并：**master上接收各个worker发送过来的局部TopK，最终产生全局的TopK。   **（2）有效提取TopK的设计方案**  Worker端：对于worker上的本地数据，是通过hash来存储，所以其数据是无序的。通过将本上千或者上万的数据进行全排序，其时间复杂度最低也要O(nlogn)，因此采用对本地数据进行全排序，显然是不可取的。因此考虑用近似查询结果来代替精确查询结果，通过随机取样技术，使得结果的近似准确，从而极大地减少查询代价。本文中选择采用采样技术来实现worker本地端TopK的提取，时间复杂度为O(n)。  Master端：master上接收来自各个worke发送过来的是有序的局部TopK，所以在master上采用归并排序来获得全局的TopK，时间复杂度为O(1)。  **（3）TopK-Online实现**   1. **TopK-Online的接口实现**   为了是实现TopK-Online功能，本功能模块为用户提供了相应的接口，来设置参数要显示的TopK个数和是否开启此功能。  本论文对提供给编程用户的类MaiterKernel进行了相应的改进，对该类的构造函数进行多态化，新定义一个构造函数，通过此构造函数用户可以设置TopK的显示个数和是否开启TopK功能。如果用户不定义TopK个数，框架将采用原来的构造函数，并将topK\_Num设置为零，系统在启动TopK功能是检查topK\_Num值，如果为零，TopK-Onlie功能择不启动。如果用户定义TopK个数，框架会将其值传递到信息表中，为后面框架实现TopK功能提供**参**数。   1. **Worker端TopK实现**   在Maiter中顶点数据是以hash的方式存储在一个vector<Bucket\_>（Bcuket\_是顶点的数据结构）中，数据是以无序的状态存在。下面将如何从上千上万的顶点中快速提取TopK的实现进行描述。为了更清晰的说明此实现，我们先定义一些变量，然后用伪代码的方式来描述此过程。  **变量定义：**  table.size：vector<Bucket\_>的大小  entities.size：实际存储的顶点数量  samples.size：采用样本的大小  topK\_Num：用户需要显示TopK的个数：  cut\_index:采样计算后的采样样本的阈值下标  thresh:阈值  thread\_index:样本阈值下标的最小值，其值越大采用的精确度越高  size\_multiple: samples容量不够时，容量扩充速度  heap\_sort: 堆排序函数  input：table, entities.size, samples.size, topK\_Num  output：lacal\_report  过程描述：  **If** table.size<= samples.size **then**  **If** entities.size<= samples.size **then**  lacal\_report **🡨** heap\_sort**(**entities**) ;**  **else**  lacal\_report**🡨** heap\_sort**(**entities**)[**0,1,…,topK\_Num**];**  **end**  **else**  samples **🡨 randomly select** samples.size records form table;  cut\_index **🡨** samples.size\*topK\_Num / entities.size;  **if** cut\_index<thread\_index **then**  samples.size**🡨** samples.size\* size\_multiple;  redo;  **else**  thresh🡨 samples[index].v2;  i 🡨 0;  **foreach** record r in table do  **if** r.priority >= thresh **then**  local\_report[i] 🡨( r.id, r.v2);  i🡨i+1;  **end**  **end**  **end**  **end**  Worker会周期性的调用上面这个函数，提取局部TopK，产生local\_report并将其发送给Master。   1. **Master端TopK的实现及结果显示**   全局TopK实现：接收各个worker的数据并经其存放在一个vector中，循环的扫描各个vector，通过归并排序，最终产生全局的TopK.  结果显示：Master会将全局TopK显示到指定的位置，并且会给用户提供当前TopK的可靠性（例如根据TopK的变化情况）。Master会周期性的执行以上操作，不断将实时TopK信息显示给用户。  3.1.5 实验  本次实验采用经典的PageRank算法，获取TopK的时间作为衡量系统性能的标准。实验环境：CPU-3.3GHZ-4Core，内存-4G；数据集：斯坦福大学的大规模网络数据集Google Webgrap，分别在Hadoop、Hadoop-TopK、Maiter、Maite-TopK四个框架中运行算法，对web-Google数据集处理，统计各个框架得到该数据集TopK的时间。运行时间如图1.1所示。  Time/s    图1.1 运行时间对比  从上表可以看出，Maiter-TopK的性能最佳，达到了预期的设计要求。   * 1. 对Maiter支持交替迭代算法的实现      1. 交替迭代算法描述   算法求解的参数有两个（如a，b），两个参数相互影响（，），算法交替计算这两个参数，通过不断的迭代计算，最终使两个参数收敛，得到参数的解。  3.2.2 交替迭代算法在Maiter中计算过程的抽象  参数说明：（发送给顶点j中参数1的消息），（发送给顶点j中参数2的消息），（顶点j中参数1的变化量到顶点h中参数2变化量的传递函数），（顶点j中参数2的变化量到顶点h中参数1变化量的传递函数），另外，分别表示当前参数的值和接收到消息中的参数差值累积。  3.2.3 实现  参照交替迭代算法在DAIC计算模型中的抽象，对Maiter框架进行改进，包括消息的传递、算法终止的判定等，并向用户提供支持交替迭代算法的API。该模块的实现中，对Maiter的数据结构重新设计，并按照交替迭代算法在DAIC模型中抽象进行相关的计算和消息的传播，具体的代码实现在本报告中就不在详细描述。  3.2.4 结果分析  对Maiter框架的应用进行了扩展，使其支持交替迭代算法的计算，并给出了计算模型的正确性证明。在实验验证阶段，对HITS（hyperlink-induced topic search）算法进行了DAIC计算模型的改进，但是对HITS算法的异步性改进中，还存在一定的问题，不能给出算法正确性的证明。但是在满足一定条件数据集的实验中，算法在Maiter中正确性可以得到保证，其实现的性能完全优于Hadoop中的实现。  3.3 Maiter动态负载均衡处理中的数据定位问题的解决  对于分布式系统来讲，系统中各个节点的负载出现不均衡将会直接影响到系统的整体性能。对应Maiter框架的负载均衡处理，我们采用动态的负载均衡策略来解决这个问题。下图1.2是Matier负载均衡处理流程图。在负载均衡处理的实现中，我参与了整个工作的设计和研究，并负责和实现了Maiter中数据快速定位的问题。    图1.2 Maiter负载均衡处理流程图  3.3.1 快速定位问题描述  集群中的每个数据都有其相应的定位信息。当一个图顶点向其他的图顶点发送消息的时候，该图顶点会给出消息的目的顶点的顶点标号，但是不会给出目的节点在那个计算节点上。为了在计算的过程中，发出的消息能够正确的到达目标图顶点，集群就必须维护每个顶点的定位信息，集群通过目的图顶点的标号，快速的判断出目标图顶点所在的位置，并正确的进行传输。在Maiter框架采用的是Hash定位的方式，具体采用Round Robin（俗称哈希取模法），其哈希函数为（p为机器个数）。这种方式实现非常的简单，也实现了对数据的快速定位，但是缺乏灵活性，非常不利于负载均衡处理中的数据迁移后的定位问题。  3.3.2 实现数据迁移后数据定位问题的研究  为了实现在负载均衡处理后，仍然实现对数据的快发定位的难题，我们必须对数据的定位方式进行重新的设计。问题研究过程中，对新定位方式的设计目标是：   1. 首先，能够像原来的定位方法一样简单有效。 2. 其次，要适应更改后的Maiter框架的数据管理方式。 3. 最后，不能增加太多的定位数据维护开销。   基于以上的的设计目标，本人设计了基于Hash的以数据块（虚拟桶）为单位的两级分布式数据定位方式。   * + 1. 两级定位的解决方案研究   **效率方面：**与简单的Hash定位方式相比，两级分布式数据定位方式要两次定位才能完成数据的定位。虽然简单的Hash定位多了一次哈希过程，但是却是可以优化的。比如消息在传出去之前都需要进行缓存。在实现的时候，可以将缓存实现成与数据块数量相应的桶。消息会通过一次哈希分配缓存在不同的桶内，每个桶中只会缓存一个数据块对应的数据。因此在消息分桶的时候采用的哈希函数与数据块划分时采用的哈希函数是一致的。每个桶会有一个标记，标记该桶中的消息属于哪个计算节点。那么在发送消息的时候，就可以直接发送。这样只通过一次哈希就能够找到对应的计算节点。  **在存储开销方面**：储存数据块与计算节点间的对应关系，需要一定的空间开销，但是数据划分时，划分的数据块个数并不会过多，至多是计算节点的十几倍，因此空间开销是常数级的。消息在分桶时所用的存储空间只是将原有的缓存改变了一下结构，并没有增加存储开销。   * + 1. 两级定位解决方案的实现   **两级定位过程：**下面通过这一个具体的例子来对其在迁移前和迁移后定位过程进行描述。假设顶点个数n=9，机器个数p=3，数据分片k=6，数据迁移前，其定位如下图1.3所示。    图1.3 迁移前数据定位过程  数据迁移前，其定位如下图1.4所示。    图1.4 迁移后数据定位过程  **两级定位的实现：**数据迁移以数据块为单位，因此数据迁移只会改变数据块与计算节点的对应位置，并且通过Hash仍然可以确定消息所属的数据块。因此在进行数据迁移的时候，只需要改变数据块与计算节点的对应关系就可以。由于数据迁移的决策是由决策决策节点发出的，那么在发出数据迁移决策的时候，决策节点只需采用广播的形式，通知所有的计算节点修改数据块与计算节点对应关系。  3.3.4实验分析  **分布式实验环境：**本实验使用的分布式集群包括4台计算节点，其中一台计算节点既是 Master 节点也是 Worker 节点，其余三台只是 Worker 节点。单机配置：CPU Intel I5-4690 3.3GHZ 4Core；Memory apacer 4G-DDR3。网络环境：1000M以太网。  **数据集：**  表1.1 实验数据集描述   |  |  |  |  | | --- | --- | --- | --- | | 数据集名称 | 顶点数量 | 边数量 | 直径 | | Journal | 4847571 | 68993773 | 16 | | Pokec | 1632803 | 30622564 | 11 | | Astro | 2473 | 9472 | 10 | | Gnutella | 6301 | 20777 | 9 |   使用有负载均衡处理的Maiter和未使用负载均衡处理的Maiter，分别对上边的数据集进行处理，运行PageRank算法和异步的SimRank算法。其运行时间如下图1.5所示。    图1.5 使用与不使用负载均衡处理运行时间对比图  从图1.5中可以看出，在出现负载不均衡问题时，启用本文提出的负载均衡机制，可以有效的提升 Maiter 框架的计算效率。结果中也显示，当任务的运行时间越长，负载均衡处理的效果也越明显。在运行时间较短时，也有一定的效果，但是有限。  **4** 学术交流  参加了学校组织的一系列的学术报告：  表1.2 参加报告一览表   |  |  |  |  | | --- | --- | --- | --- | | 报告名称 | 报告人 | 地点 | 时间 | | An Association Network for Computing Semantic Relatedness | 朱其立 教授 | 信息学院 | 2015.1.16 | | Asynchronous and Accumulative Distributed Computation for Massive Datasets | Professor Lixin Gao | 信息学院 | 2015.5.11 | | Urban Computing | 郑宇教授 | 信息学院 | 2015.5.21 | | The Internet of Things–The Ultimate ICT Revolution | Professor Joseph Sifakis | 汉卿会堂 | 2015.6.23 |   **5** 开题准备工作  在导师张岩峰教授的指导下，主要对两个方面的内容进行研究：其一，针对图数据具有的幂率分布的特性，对以边为中心的大规模异步图处理框架进行研究；其二，对大规模异步图处理框架进行增量计算的研究。图的幂率分布特性在应用图网络中（如社交网络、引用网络）都普遍存在，增量计算的需要也在应用中广泛的存在，对于这两项工作的研究都有着重大的现实意义和应用价值，同时它们也是分布式计算、大数据处理中很前沿的研究问题，因此对它们的研究也具有着重大的理论研究价值。在开题前期的准备工作中，对这两方面的内容进行了大量的学习和研究，对以边为中心的异步分布式图处理模型和框架进行的研究有了一个大概的解决方案，而对增量计算的研究仍然没有取得很大的进展，基于对毕业论文最终能否实现的考虑，在本次开题中将对以边为中心的异步分布式图处理模型和框架进行研究，而将对大规模异步图处理框架进行增量计算的研究作为可选研究内容。 |

二、选题依据（本节可以整页扩页）

|  |
| --- |
| 课题背景、选题依据、课题研究目的、理论意义和应用价值（工学硕士）/工程背景和实用价值（专业学位硕士）（不少于1000字）   1. 课题背景   **（1）图数据挖掘的巨大价值：**随着社会信息化程度的不断提高，数据采集和数据存储等技术的发展，各类应用所处理的数据量不断增大，人们探索世界的方式已经从计算科学转换到了数据科学。其中数十亿顶点规模的图数据大量涌现，如：社交网络、万维网、语义网等，采用机器学习和数据挖掘算法对这些大规模的图结构数据进行处理和分析具有着巨大的研究和应用价值。  **（2）分布式图处理框架的研究：**面对数亿级顶点如此大规模图数据，采用单机的方式对数据进行处理和分析，很显然已经无法满足当今的应用需求。目前人们通常利用分布式环境进行大数据的处理，在复杂的分布式环境中怎样来充分利用分布式环境中的各种资源必须依赖于分布式框架的管理。因此如何设计高效的高可用性的大数据处理框架，成为当今大数据领域的热点研究问题。图处理算法一般都具有数据间高耦合性、大量迭代计算等特性，分布式图处理框架也针对于这些特性，对如何提高分布式框架的性能进行研究，从具有高通用性但对图数据处理低效的同步处理框架发展到高效支持迭代计算的大规模图算法的异步类型的框架，计算模型从MapReduce到针对图数据处理的以顶点为中心的计算模型，再发展到并行化粒度更小的GAS计算模型。  我参与的国家自然科学基金项目“云环境下基于BSP模型的大规模不动点迭代计算的研究”的主要工作正是针对大规模异步处理框架的研究，我们提出了异步处理框架Maiter，并围绕着Maiter框架做一系列的研究，如何扩展Maiter的应用范围，如何优化Maiter提高其性能，旨在做出性能优越、可用性高的大规模异步图处理框架。  **（3）图数据的幂率分布特性：**随着计算机技术和互联网技术的发展，数亿顶点级别的网络已经大量涌现，而这些网络图往往具有着幂率分布（power-law degree）的特性，即很少一部分的顶点连接着大部分的边。例如，在社交网络微博中，明星、大V这样的用户，他们往往跟大量的用户都有交互，因此他们在网络图中往往有着数万甚至数百万级别的边，而绝大部分用户在图中的边往往不会超过数百条。目前，绝大多数的图处理框架（包括Maiter），都没有考虑到图数据幂率分布的特性。由于忽略了这个特性，而导致这些分布式框架的实际性能往往都没有达到预期的设计目标。同时Maiter也没有考虑图的这个特性，而导致在数据分片时任务量分配不均衡，在数据计算中图操作的并行化程度受限，导致框架的实际性能往往没有达到预期设计的目标。   1. **选题依据**   2.1图数据幂率分布特性与框架性能的紧密相关性  **（1）与计算节点负载的相关性：**在GAS计算模型中，在gather和scatter阶段中，存储量、通信量、计算量都跟顶点的度成线性关系，因此在幂率分布的图中，顶点运行时间会有很大的差别。那么从机器节点的高度来看负载情况，节点的负载就跟分配到该节点上边的数量成线性关系。在Maiter采用的是，随机哈希顶点来进行数据分片，只能保证各个机器节点上顶点的数量大致相等，但是无法保证节点上边的数量大致相等，因此会出现集群中各个机器节点负载不均衡的情况。  **（2）与图操作并行性的相关性：**图顶点之间可以并行的执行，但是顶点内无法并行操作。因此，图数据处理中要具有较高的并行化程度往往要依赖于图中顶点有着少量的邻接顶点，所以幂率分布的图中，图操作的并行性会很差。  **（3）与通信量的相关性：**在斜率很大的幂率分布中，导致机器节点之间消息的不对称性，甚至出现通信的瓶颈。  **（4）与存储量的相关性：**顶点的信息量跟它的度成线性关系，如果一个机器节点上出现较多度很高的顶点，就会导致分配到该节点上的数据量超过其单机内存的容量。  由于图幂率分布特性与框架性能之间的紧密相关性，在系统设计没有考虑到这个特性对框架影响，将会严重影响到框架的性能。因此，针对图幂率分布特性提出了以边为中心的异步分布式图处理模型，将能从从根本上消除这种特性带来的影响。  2.2 Maiter改进研究的可行性及价值性分析  **（1）可行性分析：**Maiter是针对于提高一类图处理迭代算法的计算性能而开发，Maiter的计算模型（DAIC）有着差值累积累积的特性，基于这种特性可以对框架实现很多的优化，例如Maiter的异步性执行、基于优先级调度的执行策略。基于DAIC计算模型差值累积的特性和Maiter中异步性、优先级计算的实现，再结合图幂率分布特性对Maiter进行进一步的改进和优化，有着重大的研究价值和很好的可行性。  **（2）价值性分析**：Maiter是现阶段性能非常优秀的一个大规模图处理框架，对于满足DAIC计算模型约束条件的算法，其在Maiter上实现的性能几乎优于所有的的分布式处理框架。因此对Maiter框架进行研究和改进，实现以边中心的异步图处理模型，对于大规模图处理框架和分布式计算技术的发展起到一定的推动作用。  **3** 课题研究目的  本课题的主要工作旨在通过研究图幂率分布的特性，提出了以边为中心的异步图处理模型，实现以边为中心的数据分片方法和以边为中心的计算模型，实现对图以边为粒度的操作，从根本上消除图幂率分布对图计算带来的影响。另一项工作就是，对框架Maiter的特性（尤其是Maiter计算模型中差值累积的特性）进行研究，实现一个以边为中心异步图处理模型的分布式框架Maiter+，消除图操作并行化低、通信量不对称、存储量不对称等问题。  **4** 理论意义  本课题中一个很重要的工作就是充分考虑图幂率分布的特性，利用差值累积的计算模型（DAIC）、优先级执行、异步执行等技术，对如何避免和解决上文中分析中提到的由于图幂率分布特性而导致的一系列问题进行理论的研究，提出以边为中心的异步分布式图处理模型，这对于分布式图处理技术的发展将会起到一定推动作用。  **5.**应用价值  在大规模图处理框架中，对于满足DAIC约束条件的算法，Maiter本身的性能就很大程度上优于目前大部分的大数据处理框架。本课题的研究中又将现实世界中大多数图具有幂率分布的特性考虑进来，并充分利用Maiter基于差值累积、优先级执行、异步化执行的特点对图操作进行优化，并基于Maiter实现一个以边为中心异步图处理模型的分布式框架Maiter+。Maiter+框架实现了副本顶点间的完全的并行计算，这将会很大程度上提升对图处理的性能，特别是具有幂率分布的图数据，性能的提升将会更加的明显，因此本工作具有很高的应用价值和现实意义。 |

三、文献综述（本节可以整页扩页）

|  |
| --- |
| 国内外研究现状、发展动态描述（不少于1000字）；所阅文献的查阅范围及手段，附参考文献（不少于10篇，其中近3年文献不少于5篇，英文文献不少于3篇，全部按照标准格式列出，并在文中顺序标注）   1. 大规模图处理框技术究现状   随着大数据时代的到来，人们探索世界的方式开始从计算科学发展到数据科学，对于大规模图处理技术的研究成为当前的一个研究热点。根据图计算中的计算范式，可以将其分为同步执行模型和异步执行模型。按处理单元分类，可以将其分为以点为中心的图处理模型、以边为中心的图处理模型、点边结合的图处理模型、以子图为中心处理模型、以路径为中心处理模型。下面本文将会对大规模图处理技术按照不同的种类进行研究。  1.1 同步与异步图处理模型的研究现状  1.1.1同步图处理模型  在当前同步执行的图处理框架中，主要包括MapReduce[1]和BSP[2]（Bulk Synchronous Parralle Computing）两种计算模型。  **（1）MapReduce计算模型：**MapReduce分布式计算框架最初是由Google公司于2004年提出，针对于处理大规模批处理的任务。自此之后，该计算模型被广泛的研究和使用，成为目前主流的的分布式计算模型，其中最成功的是Hadoop[3]对其的实现。但是该计算模型在大规模图数据进行处理时，显得并不是很高效。主要是由于对图的处理往往需要多次迭代计算以及数据随机访问等特点，因此MapReduce计算模型不是十分适合对大规模图数据进行处理。基于此模型实现的分布式图处理框架，除了Hadoop，还有很多实现框架，如适合进行迭代计算的iMapReduce[4]，Twister[5]，Spark[6]。  **（2）BSP计算模型:**针对于图数据处理中存在大量迭代计算等特点，Google在Pregel[2]中提出了整体同步并行计算模型（BSP）。它考虑到了图数据的结构信息，允许点对点的通信，对图的处理过程进行优化。BSP模型既包含宏观上的垂直结构，也包含微观上的水平结构。在宏观上，超级步之间按照时间轴依次执行；在微观上，超级步之内计算节点并发的执行计算子任务和消息的发送，在本次超级步结束时进行同步。遵循BSP模型的程序，对于容错处理、避免死锁、程序正确性保证等问题都能较好的解决，但是BSP模型也具有同步范式的缺点，资源利用率不高、网络通信量大等问题。对于图数据的处理，在计算过程中需要多次的迭代，BSP在每个超级步中都要进行一次同步，这严重影响了图处理框架的性能。基于BSP模型实现的分布式图处理框架，还有对Pregel开源实现的Giraph[7]，对Pregel进行进一步优化和改进的Pregelix[8]和基于优先级执行计算的PrIter[9]等框架。  1.1.2异步图处理模型  异步图执行模型是相对于同步执行而言，在计算中不需要进行任何的同步操作。它在图计算过程中，不需要进行数据的同步、而且更新的数据能够在本轮迭代中即可被使用，所以使得其算法的收敛速度块、系统吞吐量和执行效率都要明显高于同步模型。同时，异步执行模型存在着它的缺点，预测程序的执行和保证算法的正确性。异步处理框架的典型代表是Graphlab框架[10]。Graphlab框架主要用于大规模的图处，可以高效的实现分布式图算法。Graphlab在计算过程中以子图为中心，创新的使用不同的封锁机制来达到不同粒度的异步计算。在保证算法正确性的前提下实现了异步计算。  本文中进行研究和改进的Maiter[11]，同样也是一种异步的图处理框架。它创新的采用DAIC[11] (Delta Based Incremental Computation)计算模型，实现了完全的异步计算，即：各个计算节点完全自治，无需任何同步过程，完全去除了图处理算法在迭代计算中的同步开销。同时采用优先级迭代加快了算法的收敛速度，有效的提高了大规模图处理的效率。  1.2 从图处理单元来看图处理技术的研究现状  **（1）以点为中心的图处理模型：**以顶点为中心的图处理模型，是目前最主流采用的处理模型，该模型最典型的代表框架是Pregel[2]。它在对图的处理过程将顶点作为处理的基本单元，按照顶点对数据进行分片和对数据分布式的计算。在数据分片阶段，采用Edge-Cut的分片方法，往往采用哈希顶点的id，将图数据分割成若干份并将其分发给相应的机器节点。其典型特征是，顶点只出现在一个机器上，而边会可能出现在多个机器上。在图分布式计算阶段，一般采用GAS的计算模型（GAS是对以顶点为中心计算模型的细化）。在整个图处理过程中，图处理模型一直将顶点作为中心进行操作。  **（2）以边为中心的图处理模型：**随着人们对大规模图数据的认识，我们在实际应用中产生的大部分图数据都是自然图。这些图数据（如，web图、社交网络图）都有着幂率分布的特性，这使得以顶点为中心的图处理模型显得不是很高效。因此，以边为中心的处理模型被逐渐应用到分布式图处理框架中，其中该模型最典型的分布式框架是PowerGraph[12]。在该模型中，进行图数据的分片时是以边为中心，采用Vertex-Cut的分片方法，将图数据中的边均衡的分配到相应的机器上。该模型典型特征是，边只会出现在一个机器上，而顶点可能会出现在多个机器上。相对于以点为中心的图处理模型来讲，该模型是对实际应用考虑后，是对图处理模型的一次改进。  **（3）以点边结合的图处理模型：**综合以点为中心的图处理模型和以边为中心的图处理模型，而提出了点边相结合的图处理模型，实现该模型的分布式框架有GraphX[13]。在数据分片中，分别以点和边为中心对图数据进行切分。其典型特征是，顶点和边都只出现在一个机器上，没有副本的存在。相对于以边为中心图处理模型来讲，这种点边结合的图处理模型既具有了其优点，同时由于顶点没有副本而减少了对数据一致性维护的开销。  **（4）以子图为中心图处理模型：**由于在图数据中，数据记录之间存在着很强的耦合性，在图分布式计算过程中机器之间会进行信息的交互。在以顶点为中心的计算模型中（包括GAS计算模型），会产生大量的消息而影响图计算性能。为了减少消息的通信量，在GoFFish[14]中提出了以子图为中心的计算模型。这种计算模型可以有效的减少通信量，减少迭代计算的次数。  **（5）以路径为中心图处理模型：**基于图数据中，数据位置信息不足的问题，提出了一种以路径为中心的图计算模型，分布式框架PathGraph[15]就是对这一模型的一个实现。   1. 图数据分片问题的研究现状   在分布式环境下，对海量的图数据进行处理，首先面临的问题就是如何将数据比较均匀地分配到各个机器节上。对于非图数据来说，数据中的数据记录相互独立，这个问题的解决就非常的直观，只需将数据均衡的分配到各个机器上就可以了，数据切分算法不用考虑其他的约束条件。对于图数据来说，数据记录之间往往有着很强的耦合性。因此，对于图数据分片的问题，不仅仅要考虑机器之间任务量分配是否均衡的问题，还要考虑机器节点之间通信量的问题。在对图数据分片时，判定数据分片是否合理主要考虑两个因素：机器负载和网络通信总量。遗憾的是到目前为止，图数据分片问题尚未得到很好的解决。  从切分图数据的不同角度来看，图数据分片算法主要分为两类：切边法（Edge-Cut）和点切法（Vertex-Cut）[16]。  **（1）切边法：**切边法是图数据分片中最常见的一种思路，它通过切割图中的边将一个大的图分割成若干个子图，每个顶点仅出现在一个子图中。在分割过程中，要尽量将每个子图中分配到数量大致相等的顶点，也要尽量减少切割的边数，即在分割过程中要考虑负载和通信量两个因素。对于切边法的优化目标定义如下式（1）所示：    参数说明：表示边e是否被切割（切割为1，为切割为0），表示机器A上分配顶点的数量，表示不平衡因子（）。  **（2）切点法：**通过切割图中的顶点将一个大的图分割成若干个子图，每个边仅出现在一个子图中，一个顶点会出现在多个子图中。在分割过程中，要尽量保证每个子图中分配到数量大致相等的边，也要尽量减少切割的顶点数（顶点的副本数少），即在分割过程中要考虑负载和通信量两个因素。对于切点法的优化目标定义如下：  参数说明：表示顶点A副本个数，表示机器A上分配边的数量，表示不平衡因子（）。  **参考文献：**  [1] Dean. J, Ghemawat. S. MapReduce: simplified data processing on large clusters.  Communications of the ACM, 2004, 5(1):107-113.  [2] Malewicz G, Austern M, Bik. A, et al. Pregel: a system for large-scale graph processing// Proceedings of the 2010 ACM SIGMOD International Conference on Management of data. ACM, 2010:135--146.  [3] Shvachko K, Kuang H, Sanjay R, et al. The hadoop distributed file system[C]// Proceedings of the 2010 IEEE 26th Symp on Mass Storage Systems and Technologies, 2010:1-10.  [4] Zhang. Y ,Gao. Q,Gao. L, et al. iMapReduce: a distributed computing framework for iterative computation// Proceedings of the 2nd ACM Symposium on Cloud Computing(SOCC’11), 2011: 1112-1121.  [5] Ekanayake. J, Li. H, Zhang. B. Twister: a runtime for iterative mapreduce, Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing, 2010, 810-818.  [6] Zaharia. M, Chowdhury. M, Franklin. M. Spark: cluster computing with working sets //Proceedings of the 2nd USENIX conference on Hot topics in cloud computing, 2010:10-10.  [7] Giraph. //<http://giraph.apache.org/>.  [8] Bu. Y, Jia. J, Borkar. V, Carey. J, Condie. Tyson. Pregelix: Big(ger) Graph Analytics on A Dataflow Engine // Proceedings of the VLDB Endowment, 2015,2(8):161-172.  [9] Zhang. Y , Gao. Q, Gao. L, et al. PrIter: a distributed framework for prioritized iterative computations// Proceedings of the 2nd ACM Symposium on Cloud Computing(SOCC’12), 2012: 13-28.  [10] 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.  [11] Maiter Project, http://code.google.com/p/maiter, 2013-07-15.  [12] 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.  [13] Gonzalez. J, Xin .R, Dave. Ankur, Crankshaw. Daniel. GraphX: Graph Processing in a Distributed Dataflow Framework//11th USENIX Symposium on Operating Systems Design and Implementation, 2014: 605-606.  [14] GoFFish: A Sub-Graph Centric Framework for Large-Scale Graph Analytics, <http://arxiv.org/>.  [15] Yuan P, Zhang W, Xie C, et al. Fast Iterative Graph Computation: A Path Centric Approach// High Performance Computing, Networking, Storage and Analysis, 2014:401 - 412.  [16] 张俊林. 大数据日志录架构与算法[M]. 北京:电子工业出版社, 2014.9:279-284. |

四、研究内容（本节可以整页扩页）

|  |
| --- |
| 1．研究构想与思路、主要研究内容及拟解决的关键问题（不少于1000字）  1.1研究构想与思路  目前大部分的大规模图处理框架都没有考虑图幂率分布的特性，导致这些框架在处理符合幂率分布图数据的时候，出现负载不均衡、计算的并行性低、通信量瓶颈、存储过载等一系列的问题，导致框架的实际性能往往达不到预期的设计目标。如果能将对图的操作粒度从顶点细化到边，那么就能从根本上去除这种特性对图计算性能造成的影响，本开题报告中提出了以边为中心的异步图处理模型，并对该模型进行研究。之后，以大规模异步图处理框架Maiter为基础，实现一个以边为中心的异步图处理模型的分布式框架Maiter+。  研究的基本思路就是首先提出以边为中心的计算的图处理模型，对该模型的正确性、高效性等内容进行研究，得到一个较为成熟的计算模型。之后，在将该计算模型在一个分布式框架中是实现，对其性能进行实验验证，最终实现对图数据的高效处理。   1. 步骤一：提出以边为中心的异步分布式图处理模型； 2. 步骤二：实现一个支持以边为中心的异步图处理模型的分布式框架Maiter+。   1.2主要研究内容  面对现实世界中图幂率分布的特性和图数据记录之间的强耦合性，如果能将对图的操作以边为中心，那么就能从根本上去除这种特性对图计算性能造成的影响，因此我们提出了以边为中心的异步图处理模型。在以边为中心的异步图处理模型中，首先要保证该模型的正确性，并对模型的高效性给予证明，其次对数据如何进行有效的分片，如何实现更高效的优先级调度，这些都是要研究的内容。为了更好的支持满足以边为中心的异步图处理模型的算法的实现，需要开发一个实现该处理模型的分布式框架。本课题将以Maiter框架为基础，实现支持以边为中心的异步图处理模型的分布式框架Maiter+。  在今后的研究工作过程中，可能还会有新的难题或者优化方案出现（如以边为中心的计算模型），这些都是要研究和解决的问题。本课题研究工作将首先进行以下两方面的研究。   1. **研究内容1：以边为中心的异步图处理模型**   **高效的异步计算模型：**在以边为中心异步图处理模型中，同一个顶点可能出现在多个机器上，在现有的以边为中心的图处理框架中往往采用一次同步的方式，来同步副本顶点上的数据，来保证计算的正确性。但是，这在异步系统中，通过同步副本顶点的方式来保证计算的正确性，这显然是不可取的，因此我们必须提出更高效的异步计算模型。  **模型正确性与高效性的理论分析：**因为在以边为中心的异步图处理模型中，顶点可能会出现在多个机器中，这就改变了分布式图计算的过程。这对于异步计算模型来讲，保证其正确性是一个很大的挑战，必须给出严格的证明，否则对该模型之后的研究都会变得没有意义。同时，对该模型的高效性研究也是非常必要的，否则，就无法保证之后研究工作具有价值。  **按边分配的数据分割方法：**数据分片是否合理将直接影响到图分布式计算的性能，在以边为中心的异步图处理模型中采用切点法对数据进行分片。那么如何来衡量数据分片是否合理呢，具体来讲就是使得分配到各个机器上边的数量大致均衡和尽量减少顶点副本数量。从框架整体性能的高度来考虑，如果采用简单数据分片算法，在对集中式数据分片和分发中开销会很小，但是不利于第二阶段分布式计算的进行；如果采用负载的数据分片算法，会有利于第二阶段分布式图计算的进行，但是有时候导致在第一阶段中就付出很大的时间开销，得不偿失。因此，如何衡量这之间的平衡关系，或者找到新的更适合的数据分片算法，将是一个非常值得研究的问题。  **优先级调度策略：**在异步图处理模型中，基于优先级的调度策略可以有效的加快算法收敛的速度，因此对于它的研究对提升计算效率具有非常重要的意义。   1. **研究内容2：分布式计算框架Maiter+的设计与实现** 2. **数据结构设计及数据加载：**根据以边为中心的图处理模型，如何设计有效的数据结构。 3. **高效的迭代计算机制：**针对与以边为中心的图处理模型中Receive和Update两个核心操作，如何设计高效的迭代计算集中。 4. **高效且低开销的优先级计算机制：**在将以边为中心的图处理模型中优先级计算策略实现过程中，由于顶点及数据的分散和数据结构的无序性，如何高效、低开销的实现该机制是一个值得研究的问题 5. **高效的消息传递机制：**在以边为中心的异步图处理模型中，由于顶点副本的存在，使得消息传递机制变得负载，需要设计出高效的消息传递机制保证分布式图计算中消息传递的正确性。 6. **有效的迭代终止检测机制：**同样因为顶点副本的存在，使得顶点的值分散在不同的机器上，并且各个副本顶点操作的并行性，使得对于终止条件的判断变得困难，因此在Maiter+中需要对Maiter的终止检测机制进行研究和改进。   **1.3**拟解决的关键问题  **（1）数据分片均衡：**机器任务量分配的不均衡，将直接导致集群中机器之间负载的不均衡，以及从而带来的通信量不对称、计算并行性差、存储过载等问题，因此如何实现数据的均衡分配是本课题拟解决的关键问题之一。  **（2）以边为中心的异步图处理模型的正确性保证：**图计算中往往是以顶点为中心，而在以边为中心的模型中有副本顶点的存在，而且为了保证其高效性而要求各个顶点异步并行执行，因此证明该模型的正确性也是一个拟解决的关键性问题。 |
| 2. 拟采取的研究方法、技术路线、实施方案及可行性分析  2.1拟采取的研究方法与技术路线  任何理论都是以实践为基础的，本课题采用理论与实践相结合的方式，先设计基本的解决方案，然后利用实验对其验证，根据实验结果查找其不足，对其进行修改，然后再继续下一步的研究和优化。  在本硕士论文研究中的很多要解决的问题和研究的内容，都存在很多中的解决方案，而这些不同的解决方法在理论上，往往又很难严格的理论证明或者推导，因此本研究工作中采用理论研究与实验证明并行的研究方法来推进工作的进行。  2.2 实施方案  通过前期工作中对图幂率分布特性的理解和对Maiter框架思想及代码的掌握，现针对上一小节中的研究内容和拟解决的关键问题，制定了如下一个大概的实施方案，首先对提出高效的以边为中心的异步图处理模型，然后对模型的正确性和高效性进行理论分析，之后开始在Maiter框架上实现Maiter+。在实现Maiter+的过程中，从数据的加载、数据的定位、数据结构的设计、分布式迭代计算的实现、优先级调度的实现、消息传递的实现，到最后的终止检测阶段制定了一个具有很好的可行的实施方案。  2.2.1 DAIC模型的重新抽象  在采用GAS计算模型的框架中，往往是在消息收集阶段（Gather），每个顶点副本可以并发执行，但是为了保证算法的正确性，在进行Apply操作时需要进行数据的同步操作。这样的策略在异步模型中，很很难实现的，同时对于DAIC计算模型也是没有必要这样做的，因此我们对此提出了以边为中心异步图处理模型中高效的DAIC模型抽象。同在Maiter中DAIC抽象一样，也是分为Receive和Update两个阶段。下面是以机器p上的顶点j为例，描述其高效的模型，如下式4.1和4.2所示。  参数说明：表示机器p上顶点j接收到的消息；hp表示机器p上所有的顶点。  2.2.2 以边为中心异步图处理模型中DAIC模型的正确性及高效性理论分析  （1）正确性证明  DAIC（Delta-based Accumulate Iterative Computation）计算模型主要是针对迭代算法，那么传统的迭代算法的表示形式如式（1）所示，第k次迭代的结果依赖于若干k-1次迭代的结果。  DAIC计算模型约束条件一，要求上式(4.3)转化成如下形式公式(4.4)  因为满足交换律和结合律，上式（4.5）可以转换为式（4.6）  将新定义式(4.6)带入(4.4)得得到公式(4.7)：  (4.7)  DAIC计算模型约束条件二、三，要求运算⊕满足以下性质：   1. 对于方程g的分配律： 2. 交换律： 3. 结合律：   利用以上三个运算律，上式(4.7)可以进行如下的转化:  对上式（4.8）整理可得式(4.9)：  从上式(4.9)可以看出，，将顶点的计算转化到了对其副本顶点的操作。由上面的推导过程，可以看出也是一系列值累计的结果，并且因为其满足对方程g的分配律，所以将其拆分计算，仍可得到相同的计算结果。通过推导，证明了计算异步执行的正确性，保证了基于以边为中心的DAIC计算模型的正确性。  **参数说明：**上式中下标中p表示机器号，例如pi表示机器p上顶点i；m是机器中机器的个数，n是图数据中顶点的个数；g为消息传递函数。  **新计算模型收敛性和计算结果正确性的分析证明：**在同步计算过程中，消息是一跳一跳的进行传递。在异步计算过程中，各个顶点执行的速度是不一样的，现在我们将异步计算过程中消息被传递最多跳的步数k作为异步计算中“轮数”。  在同步系统中，在进行到第k轮时，每个顶点会收到所有距离它小于k步距离顶点所发送的消息，而在异步执行中，只会收到距离它小于k步距离顶点所发送的部分消息。让k1趋近与无穷大，那么同步计算中的值会收敛到一个值。同时，异步计算中其值会小于异步计算中的值，所以其值也会收敛。现在让k2=k1+，那么总会出现异步中的收集的消息会多余同步系统k1轮后所收到的消息。在异步计算中，k1步会收敛到一个值，k2也会收敛到一个值，这两个值是几乎相等的，且这两个值一个小于同步计算k1之后的值，另一个大于。因此根据夹逼准则，得到新的计算模型计算结果收敛，且正确。  **高效性分析：**首先该模型以边为中心进行数据的分片，使得各个机器上的边数大致相等，提高了机器之间计算的并行性，同时也消除了负载不均衡、通信量不对称、存储过载等问题。该模型中，副本顶点之间可以完全独立的并发执行，提高了顶点计算的并行化，在结合优先级执行的策略，将会更加有利于高效顶点的计算，进一步加快算法的收敛速度。  2.2.3 数据分片  **（1）切点法（Vertex-Cut）分片函数设计：**针对于分片算法复杂度的不同及对下一阶段分布式图计算影响的不同，制定了两个方向的解决方案。  **随机哈希：**对图中边的id进行hash，将边随机的分配到各个机器上。综合考虑分片是否均衡和顶点副本数尽量少特点。设计了如下随机Hash函数（4.10）：  **注：**P为机器的个数，此方法可以保证顶点的副本数不超过2。  在对图的结构进行研究后发现，根据高出度顶点和高入度顶点的占比将图分为三类，在本报告中将这三类图称作发散自然图（高出度顶点占比高），汇聚自然图（高入度顶点）和平衡自然图（高出度顶点和高入度顶点占比均衡）。针对这样的图结构，本文提出了带参数的哈希分片函数，来进一步保证任务量的均衡。  注：P=，p为机器的个数；此方法可以保证顶点的副本数不超过。  用户可以根据实际要处理的数据的特征，来调整哈希函数中参数，来尽可能的实现数据分片的均匀。但是这种调整可会增加顶点副本的个数，因此需要具体进一步的研究或者实验来证明那种策略最优。  **贪婪哈希（有启发信息的哈希）**：此方法一般在哈希边的时候，既考虑使数据分片均衡，也加入使顶点副本数的启发性信息，从而达到较好的效果。但是要想取得较好的效果，往往需要付出一些开销才能实现。下面给出一个，在PowerGraph论文中采用的方法。  对边edge分片时以下遵循以下规则：   1. Case1：如果集合和相交，那么edge被分配到交际中的某个机器上。 2. Case2：如何集合和非空且不相交，则从两个集合中选择一个负载最轻（边的数量最少）的一个机器，然后将edge分配到这个机器上。 3. Case3：如果集合和只有一个非空，则将edge分配到那个非空集合中的某个机器上。 4. Case4：如果集合和都为空，则将edge分配到机器中负载最轻（边的数量最少）的一个机器上。   2.2.4顶点定位问题解决方案  采用Vertex-Cut对数据进行分片中，由于是基于边进行分片，因此对于顶点的的定位成为一个难题。通过研究，现给出在采用有master顶点的机制下的一个解决方案。  **（1）master顶点的定位：**mirror顶点在进行数据交换时，要想master顶点发送消息，因此要解决master的定位问题。现给出的解决方法是，哈希顶点的id，在其对应的机器上创建一个该顶点作为master（如果数据分片中在改机器上顶点已经存在，则不需在进行操作）。所以，mirror顶点在向master发送消息时，只需要hash本身的id就能找到master的位置。  **mirror顶点的定位：**master在进行数据同步时，要向各个mirror发送消息，需要mirror的位置信息。本解决方案要求在数据分片是，一旦在一个机器上有新的顶点建立，就必须将其位置信息向master报告，这样master就有了各个mirror的位置信息。  2.2.5数据结构设计（Local State Table）及数据加载  在worker上的所有顶点及其邻接关系等信息都存在存放在本地的StateTable中，在Maiter+中，其数据结构如下图所示：    Pir  Data2  Data111111  Value  Key  is\_use  图4.1 StateTable数据结构图  **（1）数据结构说明：**数据结构中一共包括八项，Key是顶点的id；value是顶点的值；缓存接收到值变化量的累积，并用于本地邻接顶点的计算；用于缓存将要同步到其他副本的值变化量；Data1用于存储顶点在本地机器上的邻接顶点；对于Data2顶点，在Master顶点上该数据项用于副本顶点的所在机器的机器id，在Mirror顶点上，该数据项用于存储Master顶点所在的机器的机器id；Pir数据项存储顶点的优先级；is\_use记录该存储单元，是否被占用。  **（2）数据加载：**Key、Data1、Data2根据输入的数据进行初始化，Value、、根据用户编写的初始化接口函数进行初始化，Pir通过用户定义优先级计算函数接口计算得到。  2.2.6 分布式迭代计算  在以边为中心的Maiter+中，DAIC计算模型被抽象为两个阶段：Receive和Update。参照Maiter的实现方案，本课题也采用两个线程来实现这两个核心操作，现设计实施方案如下。  **（1）Receive线程：**在每个worker上Receive线程负责接收来自于顶点副本发送的数据，如果该消息的目的顶点是Master顶点，则将该消息的数值累加到和上；如果该消息的目的顶点是Mirror顶点，则只需该消息的数值累加到上。  **（2）Update线程：**worker上Receive线程负责计算、数据更新和消息发送等工作。首先，将累加到Value上，然后通过传递函数g计算出对下一跳顶点的贡献值，然后将自身的复制defaul\_value。最后进行消息的发送，将添加到目的顶点的和上。  2.2.7优先级调度  **（1）以边为中心的异步图处理模型的优先调度机制设计：**在图的迭代计算中，收敛速度的快慢往往取决于消息传播的速度，基于此，Maiter+将采用基于优先级的调度策略。顶点的优先级往往取决于当前顶点累积的变化量，但是由于顶点副本的存在，顶点累积的变化量是分散在多个机器上，如果要将这些变化量收集起来实现优先级队列的选取，那么将会付出一定开销，且由于Maiter框架的异步性使此过程不易实现。现从另外一个角度来看这个问题，Maiter计算的收敛速度取决于消息的疏通，对于各个机器来说选择本机中最有利于消息传播的顶点，就是充分发挥了本地计算资源，各个机器都最大化的将消息传播出去了，那么整个系统消息也就得到了最快的传播。因此，采用本地优先级队列，放弃使用全局优先级队列的解决方案，不但避免了开销，同时也更有利于算法的收敛。  **（2）优先级调度机制实现：**在Maiter+的数据结构中，数据是无序的，如何从大量的顶点中选取优先级高的顶点进入调度队列，我们拟采用**采样**的方法来实现优先队列的选取。通过分析，该采样方法的时间复杂度为（n为本地顶点的个数）。  2.2.8消息传递  **（1）消息格式：**Key—DeltaValue。Key为顶点id，DeltaValue为要传递的值。  **（2）消息传递机制设计：**在计算过程中，要进行两个方面的消息通信。一方面是，具有邻接关系的顶点之间的通信（实质上式本地通信）；另一方面，还要进行顶点副本之间的数据交换。消息传递机制的设计目标是：保证信息既不会丢失也不会被重复传递。针对这一设计目标，初步的设计方案如下：  **本地（本机）消息的处理：**在计算过程中，顶点通过data邻接表（只包括本地存储的边）将变化值传递到本地相关联的本地顶点上。首先，要将该值累加到本地目标顶点的中，以便目标顶点用于下一步的计算；同时也要将消息值累加到中，以便将其值传递到目标的顶点的副本上，用于副本顶点的计算。  **异地（不同机器）消息的处理：**由于Maiter的特性，在顶点副本之间既可以设定master，也可以不设定。这两种策略性能的优劣在无法给出理论性的证明，需要通过实验来进行验证，因此对于异地消息的处理在有Master和没有Master的情况下分别给出相应的消息处理机制。  **消息接收：**   1. 有master顶点机制：如果该点是master顶点，首先将接收到的异地消息累积到中，然后在将该值发送给除了发送方以外的其他所有顶点副本；如果该点不是master顶点，只需将消息值累积到中。在采用有master的顶点 2. 无master顶点机制：接收到消息后，将其累积到中即可。   **消息的发送：**   1. 有master顶点机制：如果是master顶点，则将发送给所有的其他副本（mirror），然后将清空；如果是mirror顶点，则将发送给它的master顶点，然后清空。 2. 无master顶点机制：将值发送其所有的副本顶点。   **（3）优化：**采用缓冲区的方法减少的消息量。在worker上，为每一个worker（除了自身）都创建一个对应的DeltaTable，将要发送的消息缓存在对应的DeltaTable表中，同时将相关的消息进行合并。最终DeltaTable达到一定的程度后（或者时间周期后），便将整个DeltaTable表的信息打包成一个消息发送到对应的worker上。  2.2.9终止检测  在顶点副本存在，各个副本并行进行计算的情况下，很难对当前顶点的累积的差值进行一个定量的估计。对DAIC计算模型进入深入的分析后，提出了下面的衡量计算进行程度的计算公式，公式（4.12）表示的是机器p上顶点j的状态信息。  实现：Worker周期性的收集所有顶点的和，并将其发送给Master进行汇总，判定算法是否终止。  2.3 可行性分析  在前期工作和开题报告中对该研究中可能遇到的问题和待研究的主要内容都了做了比较细致的分析，并制定了大致的实施方案，在技术上已经具有了很好的可行性。虽然数据分片函数目前仍然没有得到很好解决，但是即使采用最简单的随机哈希的方法进行数据的分片，在性能能相对于以前以点分片的方式，框架的性能也会得到很大的提升（参考PowerGraph的对比实验），因此在对框架的性能优化上也会具有很强的可行性。 | |

五、预期研究成果（本节可以整页扩页）

|  |
| --- |
| 对所研究的成果进行阐述，同时要对与前文研究内容的相关性及与前人（他人）研究成果的差异性进行描述  1 研究的成果进行阐述  （1）提出以边为中心的异步图处理模型   1. 基于DAIC模型的计算特性，提出实现副本顶点并行化的高效的异步计算模型。 2. 通过严格的数学推导和理论的分析，证明模型的正确性及高效性。 3. 提出合理性的数据分片算法，实现机器之间任务量的均衡分配，并一定程度上优化了顶点的副本数量。 4. 提出基于本地信息的优先级调度策略，相对于全局优先级调度策略，简化实现过程和复杂度，同时也更加有利于某些有助于加快收敛的顶点得到计算，加快消息的疏通。   （2）实现以边为中心的异步图处理模型的分布式框架Maiter+   1. 实现对Maiter+数据结构的设计 2. 基于双线程有效的实现了Receive和Update操作，实现高效的迭代计算机制。 3. 采用采用方法，低开销的实现优先队列的选择。 4. 基于缓存和合并的方法，实现一个高效的且正确的消息传递机制。 5. 提出合理的计算当前算法运行程度的方法，并实现迭代终止检测机制。   2 相关性  本研究工作的进行完全是依赖于前期的研究内容，同时也是对研究内容的进一步深化和研究改进。  3研究成果的差异性进行描述  在分布式图处理框架中，目前以中心的处理模型还有PowerGraph等框架。这些框架同本课题欲实现的Maiter+一样，都能有效的解决了任务量分配不均衡，通信量不对称、内存过载等问题，但是本课题提出的以边为中心的异步图处理模型是以DAIC为基础，因此还存在着以下差异。  （1）副本顶点间的并行性：在PowerGraph的GAS模型中，只有在gahter和scatter阶段顶点副本之间是独立的，在执行apply时需要进行数据的同步；而本课题的以边为中心的异步图处理模型中，顶点的副本之间可以完全独立的进行计算。  （2）优先级调度：本课题提出了基于顶点副本局部的优先级调度，这种调度方式比PowerGraph中的调度方式更加有效。  同时本课题中实现的Maiter+框架相对原来的Maiter框架，在负载均衡、通信量对称、计算的并行性等方面都有了很好的改进，另外Maiter+中实现了顶点副本的并行化操作，因此在优先级计算时，对于那些有利于收敛的顶点可以得到更快的计算，能更加进一步提高算法的收敛速度。 |

六、研究条件（本节不允许扩页）

|  |
| --- |
| 1．所需实验手段、研究条件和实验条件  （如果没有，可以空白）   * 1. 实验手段   **优化实验：**采用对不同解决方案，设计大量的对照善于，选出最佳的问题解决方案。  **对照实验1：**跟以边为中心的分布式框架（PowerGraph）进行对照实验，验证本课题中提出的图处理模型的高效性。  **对照实验2：**跟原有的Maiter分布式框架进行对照实现，验证本课题提出的图处理模型的有效性。   * 1. 研究条件和实验条件   单机硬件环境：CPU：Intel(R) Core(TM) i5 CPU；内存：4.00 GB；  集群配置：至少四个以上的计算节点。  数据集：满足幂率分布的图数据集  2．所需经费，包含经费来源、开支预算（工程设备、材料须填写名称、规格、数量）  （如果没有，可以空白）  (1)所需经费：大规模分布式集群实验租用amazon、实验材料、网费等。  (2)经费来源：国家自然科学基金项目“云环境下基于BSP模型的大规模不动点迭代计算的研究”（61300023）。 |

七、工作计划（本节不允许扩页）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 序号 | 阶段及内容 | 工作量估计  （时数） | | 起止日期 | 阶段研究成果 |
| 1 | 查询文献及资料，阅读论文 | 400 | | 2015.8-2015.10 | 熟悉相关技术 |
| 2 | 系统设计 | 350 | | 2015.11-2015.12 | 设计支持Vertex-Cut分片的解决方案 |
| 3 | 系统实现 | 400 | | 2015.12-2016.2 | 在异步处理系统上实现解决方案 |
| 4 | 性能测验 | 400 | | 2016.3-2016.4 | 在分布式集群上验证解决方案性能与改进 |
| 5 | 第二阶段研究工作中的系统设计 | 300 | | 2016.5-2016.6 | 设计提高计算并行性的解决方案 |
| 6 | 第二阶段研究工作中的系统实现与性能检测 | 600 | | 2016.7-2016.9 | 在异步处理系统上实现解决方案并进行实验验证 |
| 7 | 撰写毕业设计论文 | 400 | | 2016.10-2016.11 | 提交毕业论文 |
| 8 | 准备毕业答辩 | 250 | | 2016.12-2017.1 | 通过毕业答辩 |
|  |  |  | |  |  |
| 合计 | 3100 |