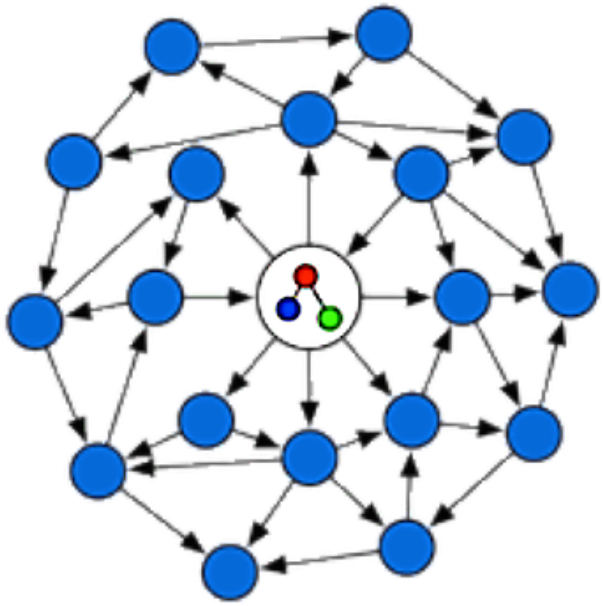
Chap. 9 图计算

王怡洋 副教授

大连海事大学 信息科学技术学院



On Graph Computing

内容提纲

*Chap. **9.1*** 图计算 简介

*Chap. **9.2*** Pregel 简介

*Chap. **9.3*** 图计算模型

*Chap. **9.5*** 的体系结构

本PPT是基于如下教材的配套讲义：

《大数据技术原理与应用——概念、存储、处理、分析与应用》（2017年2月第2版） 林子雨 编著，人民邮电出版社

9.1 图计算简介

- 9.1.1 图结构数据
- 9.1.2 传统图计算解决方案的不足之处
- 9.1.3 图计算通用软件

9.1.1 图结构数据

- 许多大数据都是以大规模图或网络的形式呈现，如社交网络、传染病传播途径、交通事故对路网的影响

- 许多非图结构的大数据，也常常会被转换为图模型后进行分析

- 图数据结构很好地表达了数据之间的关联性

- 关联性计算是大数据计算的核心——通过获得数据的关联性，可以从噪音很多的海量数据中抽取有用的信息
  - 比如，通过为购物者之间的关系建模，就能很快找到口味相似的用户，并为之推荐商品
  - 或者在社交网络中，通过传播关系发现意见领袖

9.1.2 传统图计算解决方案的不足之处

针对大型图计算问题，可能的解决方案及其不足之处具体如下：

- **（1）为特定的图应用定制相应的分布式实现：**通用性不好

- **（2）基于现有的分布式计算平台进行图计算：**在性能和易用性方面往往无法达到最优
  - 现有的并行计算框架像MapReduce还无法满足复杂的关联性计算
  - MapReduce作为单输入、两阶段、粗粒度数据并行的分布式计算框架，在表达多迭代、稀疏结构和细粒度数据时，力不从心

- **（3）使用单机的图算法库：**比如BGL、LEAD、NetworkX、JDSL、Standford GraphBase和FGL等，但是，在可以解决的问题的规模方面具有很大的局限性

- **（4）使用已有的并行图计算系统：**比如，Parallel BGL和CGM Graph，实现了很多并行图算法，但是，对大规模分布式系统非常重要的一些方面（比如容错），无法提供较好的支持

9.1.3 图计算通用软件

- 传统的图计算解决方案无法解决大型图的计算问题，因此，就需要设计能够用来解决这些问题的通用图计算软件

- 针对大型图的计算，目前通用的图计算软件主要包括两种：
  - 第一种主要是**基于遍历算法的、实时的图数据库**，如Neo4j、OrientDB、DEX和 Infinite Graph
  - 第二种则是**以图顶点为中心的、基于消息传递批处理的并行引擎**，如GoldenOrb、Giraph、Pregel和Hama，这些图处理软件主要是基于BSP（整体同步并行计算模型）实现的并行图处理系统

9.2 Pregel简介

•谷歌公司在2003年到2004年公布了GFS、MapReduce和BigTable，成为后来云计算和Hadoop项目的重要基石

•谷歌在后Hadoop时代的新“三驾马车”——Caffeine、Dremel和Pregel，再一次影响着圈子与大数据技术的发展潮流

•Pregel是一种基于BSP模型实现的并行图处理系统

•为了解决大型图的分布式计算问题，Pregel搭建了一套可扩展的、有容错机制的平台，该平台提供了一套非常灵活的API，可以描述各种各样的图计算

•Pregel作为分布式图计算的计算框架，主要用于图遍历、最短路径、PageRank计算等等

9.3 Pregel图计算模型

- 9.3.1 有向图和顶点
- 9.3.2 顶点之间的消息传递
- 9.3.3 Pregel的计算过程
- 9.3.4 实例

9.3.1 有向图和顶点

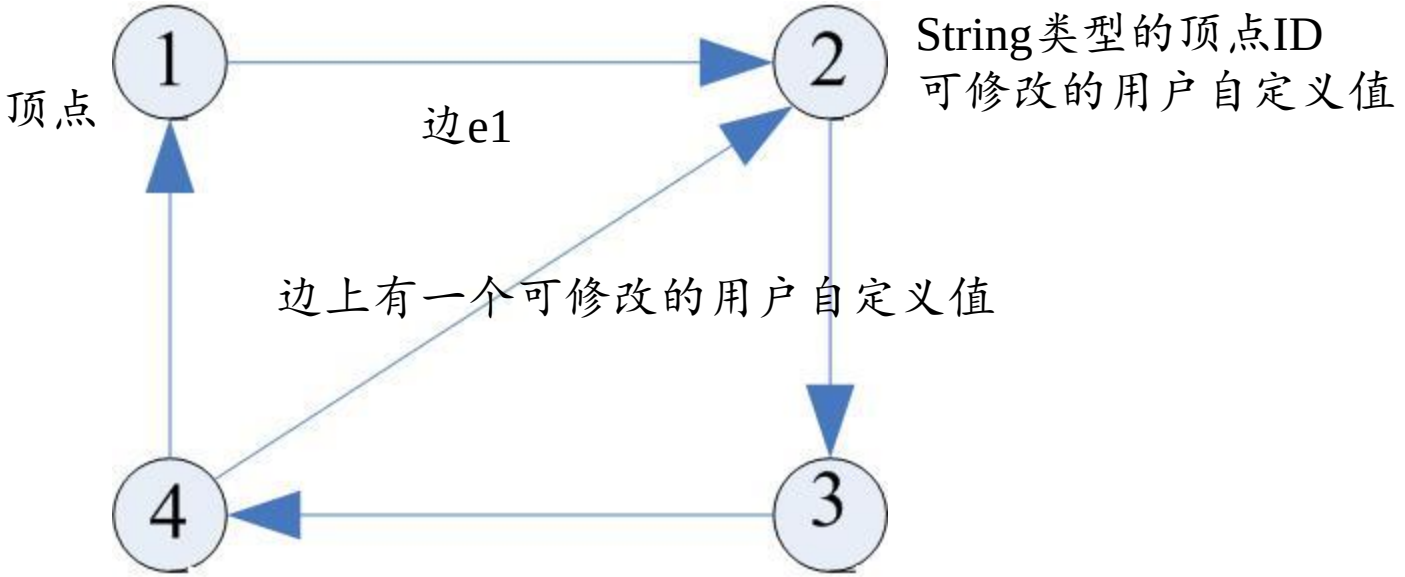
•Pregel计算模型以有向图作为输入

•有向图的每个顶点都有一个String类型的顶点ID

•每个顶点都有一个可修改的用户自定义值与之关联

•每条有向边都和其源顶点关联，并记录了其目标顶点ID

•边上有一个可修改的用户自定义值与之关联



9.3.1 有向图和顶点

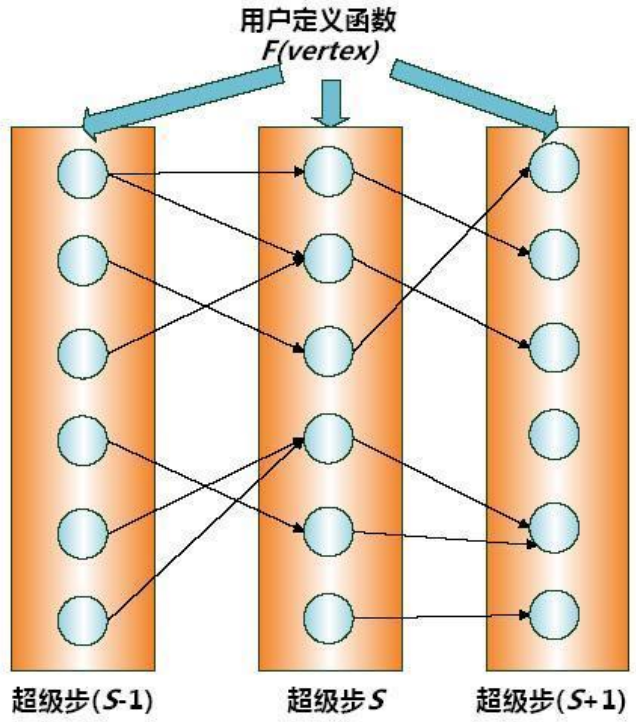
•在每个超步S中，图中的所有顶点都会并行执行相同的用户自定义函数

•每个顶点可以接收前一个超步(S-1)中发送给它的消息，修改其自身及其出射边的状态，并发送消息给其他顶点，甚至是修改整个图的拓扑结构

•在这种计算模式中，“边”并不是核心对象，在边上面不会运行相应的计算，只有顶点才会执行用户自定义函数进行相应计算

○ 表示顶点

⟶ 表示发送消息



9.3.2 顶点之间的消息传递

采用纯消息传递模型主要基于以下两个原因：

（1）消息传递具有足够的表达能力，没有必要使用远程读取或共享内存的方式

（2）有助于提升系统整体性能。大型图计算通常是由一个集群完成的，集群环境中执行远程数据读取会有较高的延迟；Pregel的消息模式采用异步和批量的方式传递消息，因此可以缓解远程读取的延迟

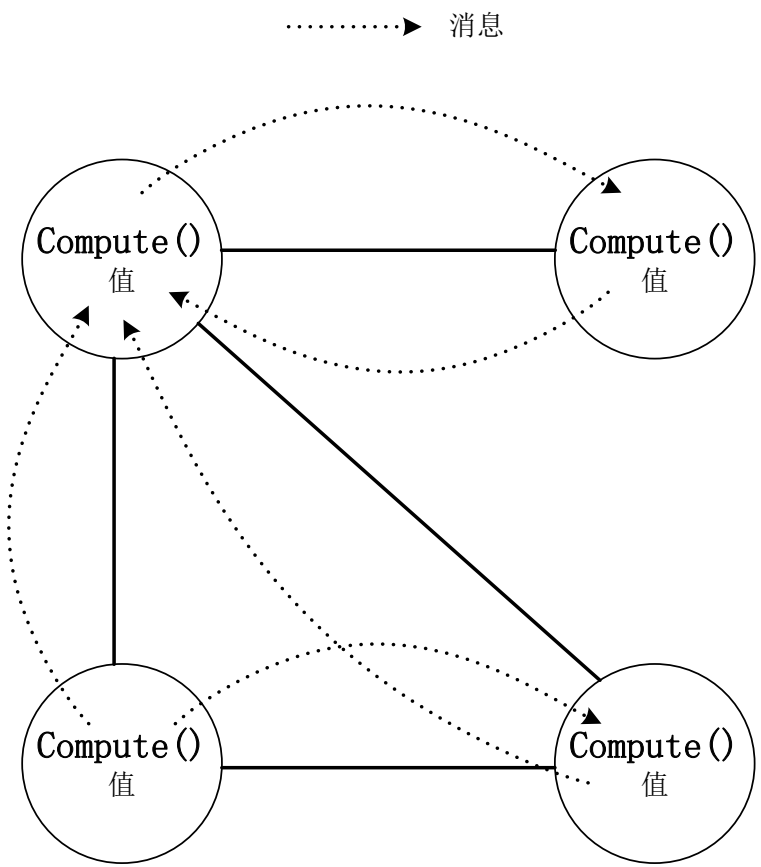


图9-2 纯消息传递模型图

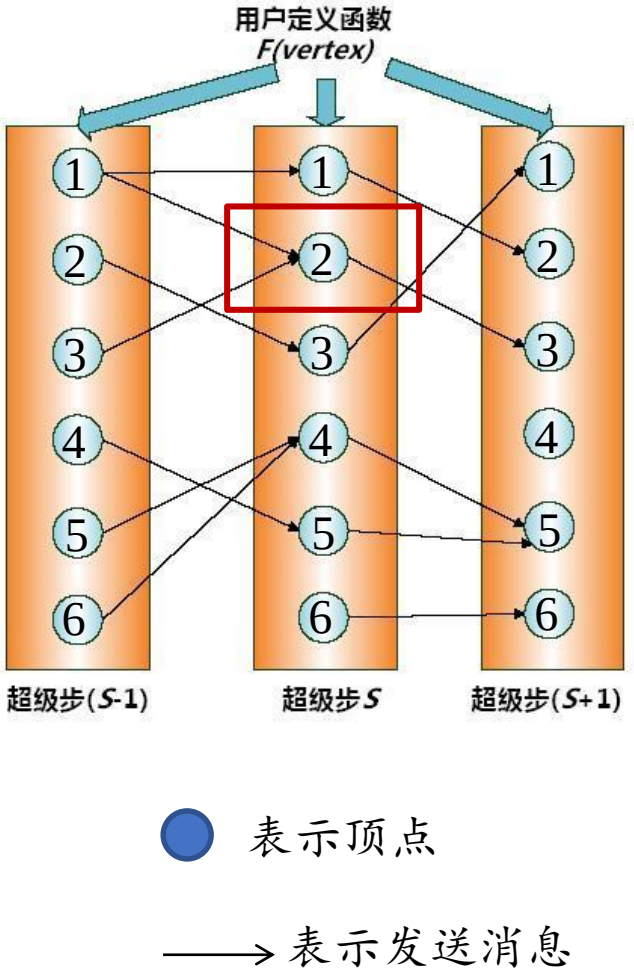
9.3.3 Pregel的计算过程

•Pregel的计算过程是由一系列被称为“超步”的迭代组成的

•在每个超步中，每个顶点上面都会并行执行用户自定义的函数，该函数描述了一个顶点*V*在一个超步*S*中需要执行的操作

•该函数可以读取前一个超步(*S*-1)中其他顶点发送给顶点*V*的消息，执行相应计算后，修改顶点*V*及其出射边的状态，然后沿着顶点*V*的出射边发送消息给其他顶点，而且，一个消息可能经过多条边的传递后被发送到任意已知ID的目标顶点上去

•这些消息将会在下一个超步(*S*+1)中被目标顶点接收，然后象上述过程一样开始下一个超步(*S*+1)的迭代过程



9.3.3 Pregel的计算过程

•在Pregel计算过程中，一个算法什么时候可以结束，是由所有顶点的状态决定的

•在第0个超步，所有顶点处于活跃状态，都会参与该超步的计算过程

•当一个顶点不需要继续执行进一步的计算时，就会把自己的状态设置为“停机”，进入非活跃状态

•一旦一个顶点进入非活跃状态，后续超步中就不会再在该顶点上执行计算，除非其他顶点给该顶点发送消息把它再次激活

•当一个处于非活跃状态的顶点收到来自其他顶点的消息时，Pregel计算框架必须根据条件判断来决定是否将其显式唤醒进入活跃状态

•当图中所有的顶点都已经标识其自身达到“非活跃（inactive）”状态，并且没有消息在传送的时候，算法就可以停止运行

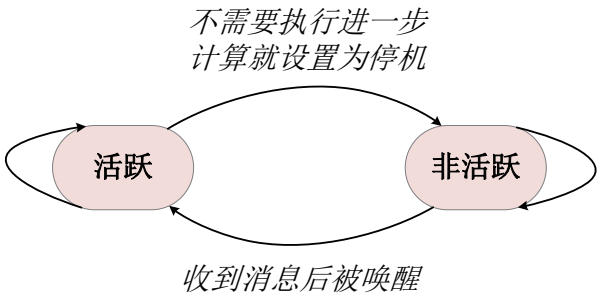


图9-3 一个简单的状态机图

9.3.4 实例

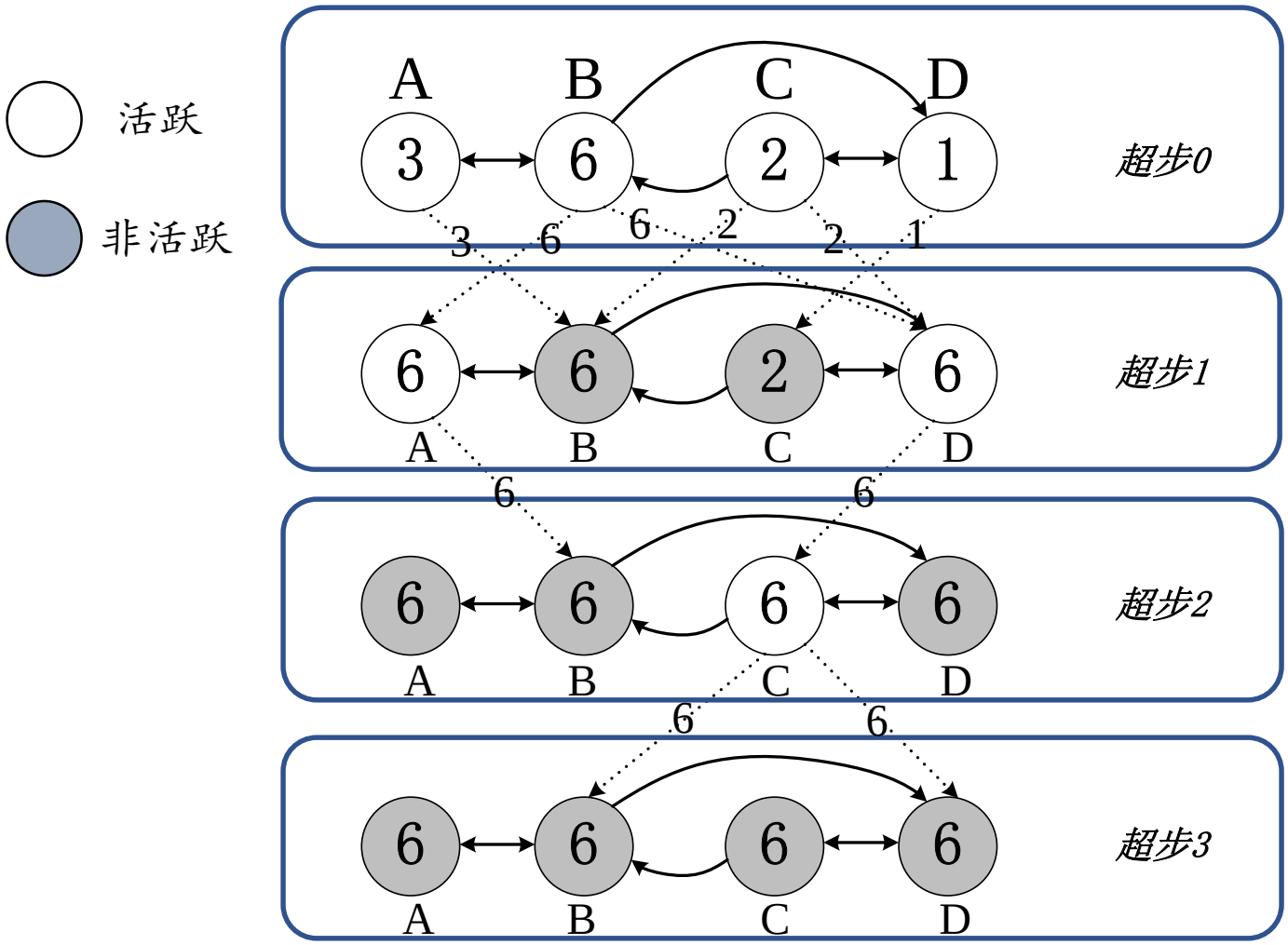


图9-4 一个求最大值的Pregel计算过程图

9.5 Pregel的体系结构

- 9.5.1 Pregel的执行过程
- 9.5.2 容错性
- 9.5.3 Worker
- 9.5.4 Master
- 9.5.5 Aggregator

9.5.1 Pregel的执行过程

•在Pregel计算框架中，一个大型图会被划分成许多个分区，每个分区都包含了一部分顶点以及以其为起点的边

•一个顶点应该被分配到哪个分区上，是由一个函数决定的，系统默认函数为hash(ID) mod *N*，其中，*N*为所有分区总数，ID是这个顶点的标识符；当然，用户也可以自己定义这个函数

•这样，无论在哪台机器上，都可以简单根据顶点ID判断出该顶点属于哪个分区，即使该顶点可能已经不存在了

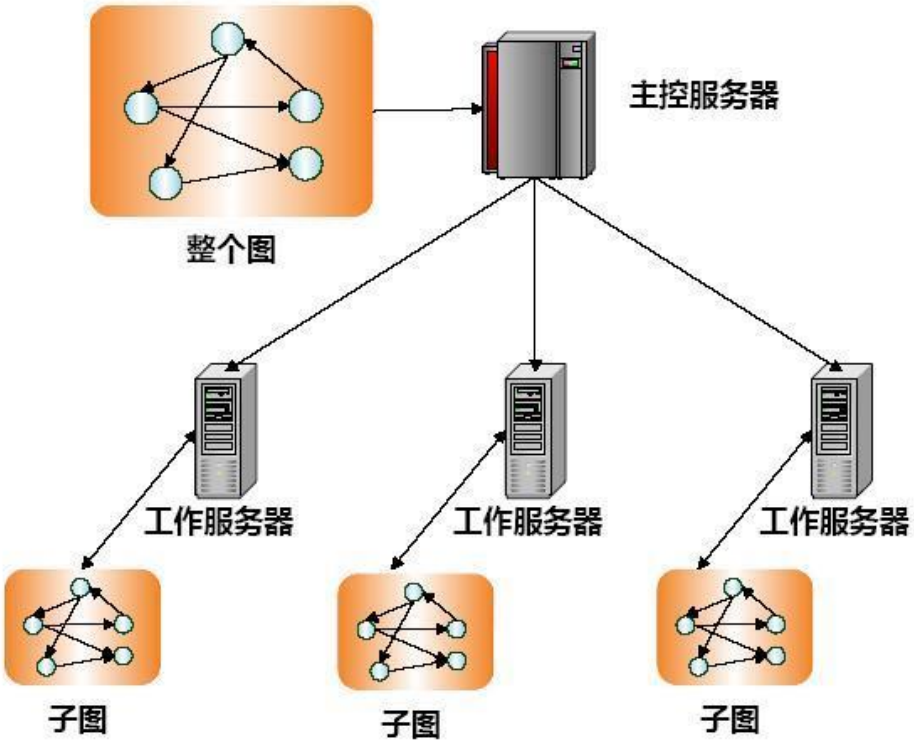


图9-6图的划分图

9.5.1 Pregel的执行过程

在理想的情况下（不发生任何错误），一个Pregel用户程序的执行过程如下：
（1）选择集群中的多台机器执行图计算任务，每台机器上运行用户程序的一个副本，其中，有一台机器会被选为Master，其他机器作为Worker。Master只负责协调多个Worker执行任务，系统不会把图的任何分区分配给它。Worker借助于名称服务系统可以定位到Master的位置，并向Master发送自己的注册信息。

（2）Master把一个图分成多个分区，并把分区分配到多个Worker。一个Worker会领到一个或多个分区，每个Worker知道所有其他Worker所分配到的分区情况。每个Worker负责维护分配给自己的那些分区的状态(顶点及边的增删)，对分配给自己的分区中的顶点执行Compute()函数，向外发送消息，并管理接收到的消息。

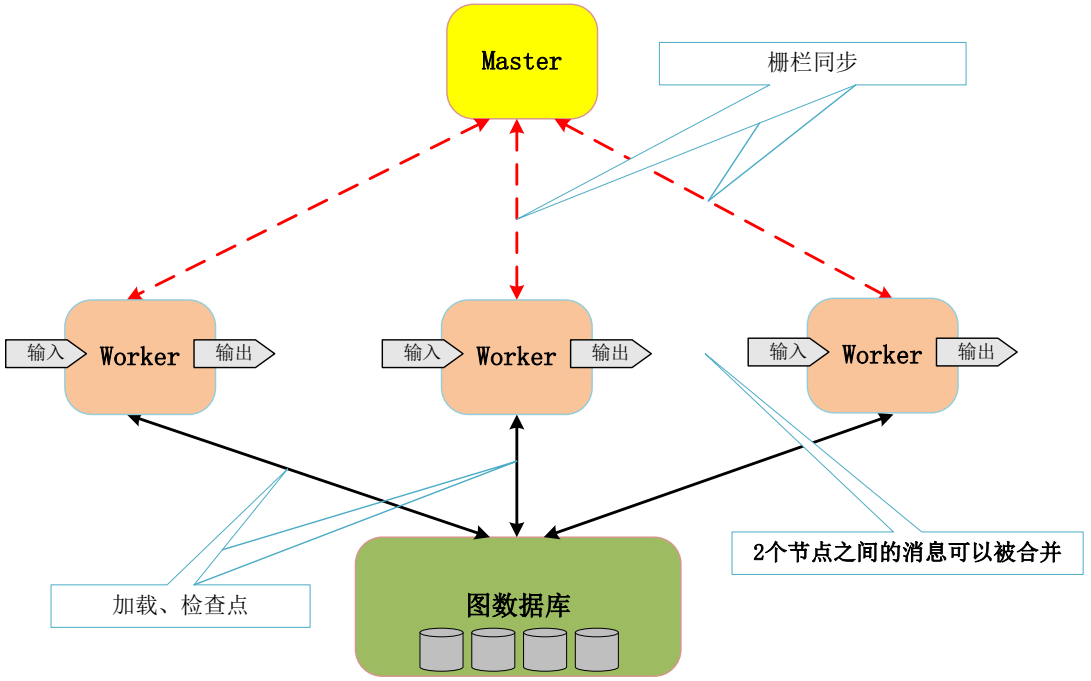


图9-7 Pregel的执行过程图

9.5.1 Pregel的执行过程

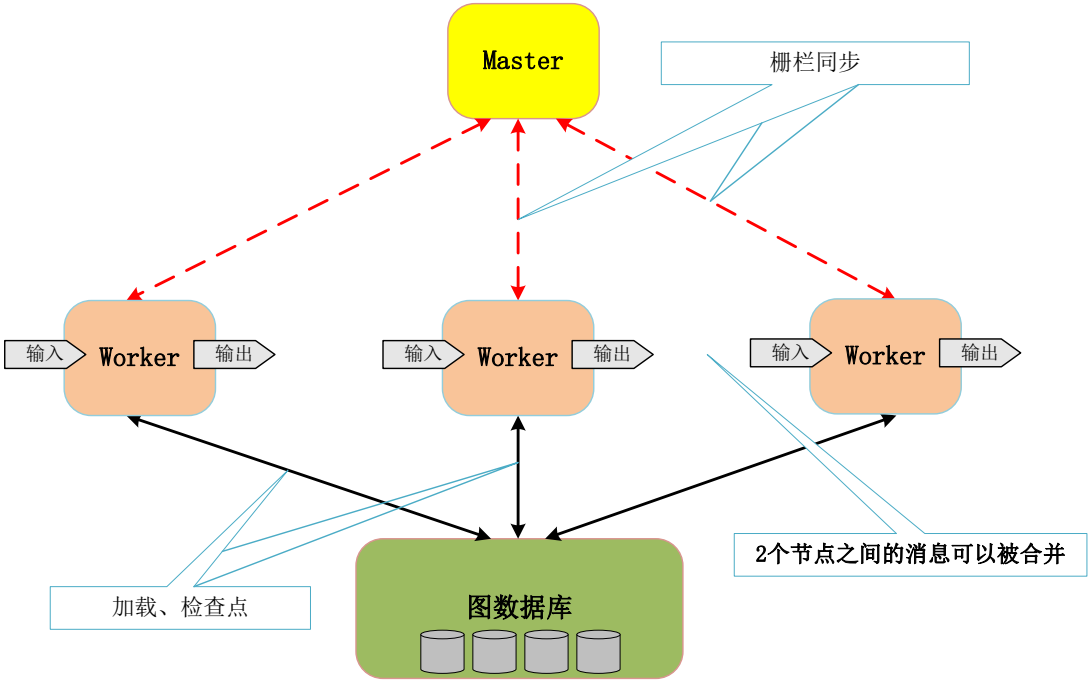


图9-7 Pregel的执行过程图

（3）Master会把用户输入划分成多个部分，通常是基于文件边界进行划分。划分后，每个部分都是一系列记录的集合，每条记录都包含一定数量的顶点和边。然后，Master会为每个Worker分配用户输入的一部分。如果一个Worker从输入内容中加载到的顶点，刚好是自己所分配到的分区中的顶点，就会立即更新相应的数据结构。否则，该Worker会根据加载到的顶点的ID，把它发送到其所属的分区所在的Worker上。当所有的输入都被加载后，图中的所有顶点都会被标记为“活跃”状态。

9.5.1 Pregel的执行过程

（4）Master向每个Worker发送指令，Worker收到指令后，开始运行一个超步。Worker会为自己管辖的每个分区分配一个线程，对于分区中的每个顶点，Worker会把来自上一个超步的、发给该顶点的消息传递给它，并调用处于“活跃”状态的顶点上的Compute()函数，在执行计算过程中，顶点可以对外发送消息，但是，所有消息的发送工作必须在本超步结束之前完成。当所有这些工作都完成以后，Worker会通知Master，并把自己在下一个超步还处于“活跃”状态的顶点的数量报告给Master。上述步骤会被不断重复，直到所有顶点都不再活跃并且系统中不会有任何消息在传输，这时，执行过程才会结束。

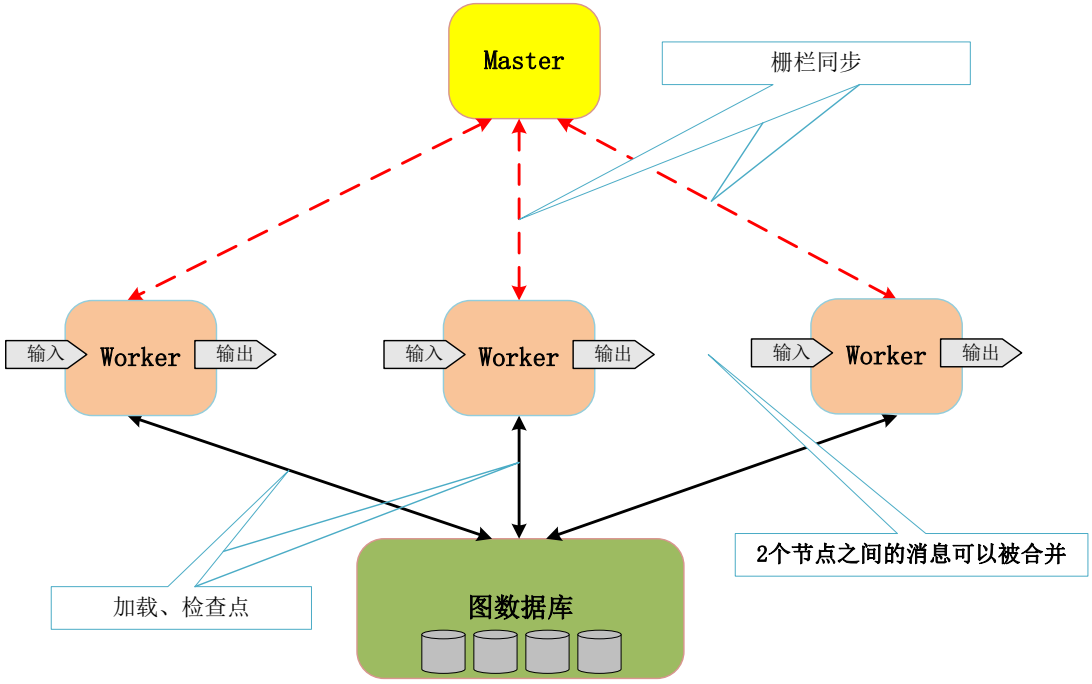


图9-7 Pregel的执行过程图

（5）计算过程结束后，Master会给所有的Worker发送指令，通知每个Worker对自己的计算结果进行持久化存储。

9.5.2 容错性

- Pregel采用检查点机制来实现容错。在每个超步的开始，Master会通知所有的Worker把自己管辖的分区的状态（包括顶点值、边值以及接收到的消息），写入到持久化存储设备

- Master会周期性地向每个Worker发送ping消息，Worker收到ping消息后会给Master发送反馈消息。如果Master在指定时间间隔内没有收到某个Worker的反馈消息，就会把该Worker标记为“失效”。同样地，如果一个Worker在指定的时间间隔内没有收到来自Master的ping消息，该Worker也会停止工作

- 每个Worker上都保存了一个或多个分区的状态信息，当一个Worker发生故障时，它所负责维护的分区的当前状态信息就会丢失。Master监测到一个Worker发生故障“失效”后，会把失效Worker所分配到的分区，重新分配到其他处于正常工作状态的Worker集合上，然后，所有这些分区会从最近的某超步S开始时写出的检查点中，重新加载状态信息

9.5.3 Worker

在一个Worker中，它所管辖的分区的状态信息是保存在内存中的。
分区中的顶点的状态信息包括：
•顶点的当前值
•以该顶点为起点的出射边列表，每条出射边包含了目标顶点ID和边的值
•消息队列，包含了所有接收到的、发送给该顶点的消息
•标志位，用来标记顶点是否处于活跃状态

在每个超步中，Worker会对自己所管辖的分区中的每个顶点进行遍历，并调用顶点上的Compute()函数，在调用时，会把以下三个参数传递进去：
•该顶点的当前值
•一个接收到的消息的迭代器
•一个出射边的迭代器

9.5.3 Worker

- 在Pregel中，为了获得更好的性能，“标志位”和输入消息队列是分开保存的
- 对于每个顶点而言，Pregel只保存一份顶点值和边值，但是，会保存两份“标志位”和输入消息队列，分别用于当前超步和下一个超步
- 在超步S中，当一个Worker在进行顶点处理时，用于当前超步的消息会被处理，同时，它在处理过程中还会接收到来自其他Worker的消息，这些消息会在下一个超步S+1中被处理，因此，需要两个消息队列用于存放作用于当前超步S的消息和作用于下一个超步S+1的消息
- 如果一个顶点V在超步S接收到消息，那么，它表示V将会在下一个超步S+1中（而不是当前超步S中）处于“活跃”状态

9.5.3 Worker

- 当一个Worker上的一个顶点V需要发送消息到其他顶点U时，该Worker会首先判断目标顶点U是否位于自己机器上
- 如果目标顶点U在自己的机器上，就直接把消息放入到与目标顶点U对应的输入消息队列中
- 如果发现目标顶点U在远程机器上，这个消息就会被暂时缓存到本地，当缓存中的消息数目达到一个事先设定的阈值时，这些缓存消息会被批量异步发送出去，传输到目标顶点所在的Worker上
- 如果存在用户自定义的Combiner操作，那么，当消息被加入到输出队列或者到达输入队列时，就可以对消息执行合并操作，这样可以节省存储空间和网络传输开销

9.5.4 Master

•Master主要负责协调各个Worker执行任务，每个Worker会借助于名称服务系统定位到Master的位置，并向Master发送自己的注册信息，Master会为每个Worker分配一个唯一的ID

•Master维护着关于当前处于“有效”状态的所有Worker的各种信息，包括每个Worker的ID和地址信息，以及每个Worker被分配到的分区信息

•虽然在集群中只有一个Master，但是，它仍然能够承担起一个大规模图计算的协调任务，这是因为Master中保存这些信息的数据结构的大小，只与分区的数量有关，而与顶点和边的数量无关

9.5.4 Master

- 一个大规模图计算任务会被Master分解到多个Worker去执行，在每个超步开始时，Master都会向所有处于“有效”状态的Worker发送相同的指令，然后等待这些Worker的回应

- 如果在指定时间内收不到某个Worker的反馈，Master就认为这个Worker失效

- 如果参与任务执行的多个Worker中的任意一个发生了故障失效，Master就会进入恢复模式

9.5.4 Master

- Master在内部运行了一个HTTP服务器来显示图计算过程的各种信息

- 用户可以通过网页随时监控图计算执行过程各个细节
  - 图的大小
  - 关于出度分布的柱状图
  - 处于活跃状态的顶点数量
  - 在当前超步的时间信息和消息流量
  - 所有用户自定义Aggregator的值

9.5.5 Aggregator

- 每个用户自定义的Aggregator都会采用聚合函数对一个值集合进行聚合计算得到一个全局值
- 每个Worker都保存了一个Aggregator的实例集，其中的每个实例都是由类型名称和实例名称来标识的
- 在执行图计算过程的某个超步S中，每个Worker会利用一个Aggregator对当前本地分区中包含的所有顶点的值进行归约，得到一个本地的局部归约值
- 在超步S结束时，所有Worker会将所有包含局部归约值的Aggregator的值进行最后的汇总，得到全局值，然后提交给Master
- 在下一个超步S+1开始时，Master就会将Aggregator的全局值发送给每个Worker