5-8-cn

1 本节我们讨论并行图计算模型
2 数据处理系统提供大数据计算处理能力和应用开发平台。
从计算架构的角度，将数据处理系统分为算法层、计算模型层、计算平台和引擎层等。
与大数据相关的计算算法包括机器学习算法和数据挖掘算法。
计算模型是指不同类型的大数据在不同场景下的处理方式，
包括批处理、流计算、结构化数据的大规模并发处理(MPP)模型、内存计算模型和数据流图模型。
就计算平台和引擎而言，通常具有代表性的大数据处理平台有 Hadoop、Spark、storm、Pregel 等，
本节我们讨论并行图计算模型
3
通过 Hadoop 和 Spark 可以完成数据的并行处理，提高计算性能。那么，单节点处理能力之外的大图形呢?

图并行计算可以解决这个问题，代表是谷歌 Pregel, Apache graph lab 和 Apache Giraph。
2010 年谷歌发表的一篇论文首次概述了 Pregel。
它是一个用于大规模图形处理的系统(想想数十亿个节点)，它是 Apache Giraph 的灵感来源，Facebook 在内部使用它来分析他们的社交网络，Apache Spark 的 GraphX 库为 Pregel 计算提供了一个 API。
GraphLab 项目是由卡耐基梅隆大学的 Carlos Guestrin 教授在 2009 年发起的。它是一个使用 Apache 许可证的开源项目。虽然 GraphLab 最初是为机器学习任务开发的，但它在广泛的其他数据挖掘任务中取得了巨大成功;在数量级上超越其他抽象。
Apache Giraph 是为高可伸缩性而构建的迭代图形处理系统。
这两个系统的灵感都来自于 Leslie Valiant 介绍的分布式计算的批量同步并行模型。
Giraph 在基本的 Pregel 模型之外增加了几个特性，包括主计算、共享聚合器、面向边的输入、内核外计算等等。
4 大多数图处理算法可以用“遍历”和“变换”的组合来表示。
在“遍历”的情况下，它可以表示为包含很多段的序列的路径。
每个段都包含从节点到弧的遍历，然后是从弧到节点的遍历。
图变换的主要目的是修改图。
这包括修改现有节点和弧的属性，创建新的弧/节点和删除现有的弧/节点。
修改逻辑由用户定义的函数提供，该函数将应用于所有活动节点。
5 最基本的(原子)单元是一个“节点”，它包含其属性、向外弧线(及其属性)以及向外弧线所指向的节点 id(仅为 id)。 节点还有一个逻辑收件箱来接收发送给它的所有消息。
Pregel 可以被认为是一个广义的并行图变换框架。

6 整个图被分解成多个“分区”，每个分区包含大量节点。
分区是一个执行单元，通常有一个与之关联的执行线程。
一台“worker”机器可以承载多个“分区”。 让我简单描述一下这个过程
第一: 处理模型为所有活动节点都将被执行
当没有更多活动节点和没有更多在传输消息时，整个处理完成。
Superstep 超步执行是，
1)收到来自收件箱的邮件，
2)修改 node 和 arc 属性。
3)停止自我直到收到新消息。
4)向其他节点发送消息，使其激活;
5)删除现有的或创建新的弧线。
执行模型基于 BSP (Bulk Synchronous Processing)模型。

在这个模型中，有多个处理单元以“Superstep”超步的顺序并行进行。
在每个“Superstep”超步中，每个处理单元首先接收前面的“Superstep” “超步”传递给它们的所有消息，
然后操纵它们的本地数据，并可能将它打算发送给其他处理单元的消息排队。
这在所有处理单元之间是异步和同时发生的。
排队的消息将被传递到指定的处理单元，但直到下一个“Superstep” “超步”才会被看到。
当所有处理单元完成消息传递(因此是同步点)时，下一个“Superstep”超步可以开始了，
如此循环，直到达到终止条件。
注意，根据图算法，将节点分配到分区可能会对整体性能产生影响。
Pregel 提供了一个默认的分配，其中 partition = nodeId % N，但用户可以在需要时覆盖这个分配算法。
通常，将近邻节点放在同一个分区中是个好主意，这样这些节点之间的消息就不需要流到网络中，从而减少通信开销。
当然，这也意味着遍历相邻节点都发生在同一台机器中，从而阻碍了并行性。
当上下文节点非常多样化时，这通常不是问题。
7
在每一个超步骤中，
它分为计算和通信两个阶段。
在计算阶段，节点完成各自的计算任务;在通信阶段，顶点发送和接收消息。
如果顶点收到消息，它将在下一个超步中处于活动状态，如果没有收到消息，顶点将在下一个超步中处于非活动状态。
8
完整的执行过程如图所示:
基本处理单元是与每个分区相关联的“线程”，运行在一个 worker 的内部。
每个 worker 从它的 inQ 中收到来自上一个“superstep”的消息
并将消息发送到目标节点所在的对应分区。之后，在分区的每个节点上调用用户定义的“compute()”函数。
注意，**每个分区只有一个线程，因此分区中的节点是按顺序执行的** 而且执行的顺序是不确定的。
“master”在协调超步的执行上起着中心作用。
在知道所有 worker 都完成了上一个步骤之后，这标志着对所有 worker 来说一个新的超步的开始。
它还会对每个 worker 发出 ping 信号，以了解他们的处理状态
并定期向所有 worker 发出“检查点”命令，然后将其分区保存到持久的图存储中。
Pregel 没有定义或强制要求图存储模型，所以任何持久机制都应该工作得很好。
在开始阶段有一个“加载”阶段，每个分区开始时都是空的，并读取图存储的一个切片。
对于从存储中读取的每个节点，如果函数返回相同的节点，将调用"partition()"函数并加载当前分区中的节点，
否则，该节点将排队到分配给该节点的另一个分区。
通过检查点机制来实现故障恢复，在检查点机制中，每个 worker 被指示定期(在超步开始时)将其内存中的图分区保存到图存储中。
如果检测到 worker 已经死亡(没有响应来自 master 的“ping”消息)，
master 将指示幸存的 worker 接管失败 worker 的分区。
整个处理过程将返回到前一个检查点，并从那里再次进行(即使是正常的 worker 也需要重做前一个处理)。
在 Pregel 中还可以进行进一步优化，以减少网络带宽的使用。
可以使用用户定义的“combine()”函数将发送到同一节点的消息组合在一起，该函数必须具有关联性和交换性。
这类似于 Map/Reduce 模型中的 combine()方法。
此外，每个节点还可以在“compute()”末尾发出一个“聚合值”。
Worker 将调用一个用户定义的“aggregate()”函数，该函数将所有节点的聚合值聚合到一个分区级别的聚合值，并一直聚合到主节点。

最后的聚合值将在下一个超步中提供给所有节点。
只需聚合值即可计算出各节点的汇总统计值，并协调各处理单元的进度。
9
1 对于图来说，超越 MapReduce 移动数据
许多问题本质上形成了一个图表，如网络、交通和社交媒体。然而，作为一种常见的大规模分布式处理管道，MapReduce 并不适合执行这样的任务。MapReduce 要求数据块独立处理。换句话说，每项工作都需要知道计算所需的所有信息。这种独立性当然不是图形的情况，作业可能需要先前的计算或来自邻近作业的信息才能进行计算。为了解决这个问题，谷歌的 Pregel 架构采用了一个消息传递系统，创建了一个“大规模图形处理”框架。
另一个问题是 MapReduce 为了处理数据而移动数据，例如通过 shuffle。
这种方法需要在机器之间移动图分区，从而产生较高的网络开销。
当 MapReduce 作业被链接到实现迭代图算法时，也会出现类似的问题。
因此，“MapReduce 需要将图的整个状态从一个阶段传递到下一个阶段——通常需要更多的通信和相关的序列化开销。”
最后，“协调链式 MapReduce 的步骤会增加编程的复杂性，这是 Pregel 在超步上的迭代所避免的。”
MapReduce 不是迭代的，它可以处理单个迭代，需要用户自己处理迭代，即多个 MapReduce 作业的链接。

对于大规模运行的真实世界的图算法来说，这种情况可能会很快变得复杂，不仅要实现，而且要在出错时进行调试。
用户算法被分布到许多机器上，其中一台成为 Master。
Master 将图划分为多个图分区，并将它们分配给 worker。
一个 worker 可以有多个图分区，其中包含一组顶点和它们对应的传出边，这些传出边可能指向图中的任何顶点，可能存储在另一个 worker 上。
设置完成后，master 命令每个 worker 执行一个超步。
在超步骤的开始，工作人员将其当前状态保存到持久存储中以实现容错。
worker 然后 compute( ) 图形分区中活动顶点的用户函数，以异步发送和接收消息。
当 worker 完成计算时，它会告诉 master 它已经完成了计算，以及在下一个超步中有多少顶点是活动的。
对于消息传递，为超步 S 和 S+1 维护两个队列，以便在计算当前超步的同时接收消息。
在超步 S 发送的任何消息都在超步 S+1 接收，以避免在消息级别锁定。
对于接收到的消息的顺序没有保证。
在计算时，一个 worker 可以根据顶点存储的位置向包括自己在内的其他 worker 发送消息。
当达到某个消息缓冲区大小阈值时，消息将以单个网络消息的形式发送，以减少网络流量。
如果用户指定了可选的组合器，则流入和流出的消息进行组合，分别减少网络使用量和存储。
例如，如果用户对它们的值的和感兴趣，可以将几个消息组合成一个消息。
用户可以指定聚合器 combiner 在每个超步收集结果。worker 向聚合器实例提供值，以生成单个本地值的聚合 Aggr。
在超步的最后，worker 使用基于树的约简将部分约简的聚合器局部值合并为全局值，并将其交付给 master。
当所有的 worker 完成一个超步时，master 启动下一个超步的执行。
这个过程一直重复，直到没有活动顶点为止。
顶点在投票停止后变为不活动的，并且没有传入的消息来再次唤醒它们。
当所有顶点都不活动时，算法的执行将终止。
10
应用数据算法层算法，结合批处理、流处理、海量并行处理，实现内存计算
或者图计算模型，我们可以处理大数据问题，但是我们很难自己手动实现所有的算法和计算模型，
计算平台和计算引擎层可以为我们提供平台和引擎，包括所需的工具，库，以促进实现复杂的算法和计

算模型。
计算平台和引擎是指为大数据计算和分析提供技术标准、计算架构以及一系列开发技术和工具的开发集成环境。
目前比较有代表性的计算平台有:Hadoop、Spark、Storm以及基于一系列大数据计算技术的商业平台。
11
该表概述了批处理、流计算、大规模并行处理、内存计算、图计算和交互计算等技术的计算模型和平台
计算模式、代表性产品、存储系统、计算模型、计算平台和关键技术
12
本节我们讨论了并行图计算模型，今天的内容就到这里，谢谢大家