**图计算综述**

# 引言

# 编程模型

# 计算泛型

# 现有框架

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 系统类型 | 系统 | 数据划分 | 编程模型 | 执行模型 | 特点 |
| **离线计算**  图数据是静态的 | Spark GraphX | 节点随机哈希 | Vertex-centric | BSP |  |
| Flink Gelly | 节点随机哈希 | Vertex-centric | BSP |  |
| Pregel | 节点随机哈希 | Vertex-centric | BSP | 鼻祖 |
| GraphLab | 节点随机哈希 | GAS | 异步执行 | C++编写 |
| PowerGraph | 节点随机哈希，Edge-Cut,  Vertex-Cut,  启发式v-cut | GAS | 混合 | 目前主流图计算系统中效率最高的 |
| Giraph | 节点随机哈希 | Vertex-centric | BSP | 建立在Hadoop平台上 |
| GraphChi(单机) | Edge-Cut,v排序 |  | 异步执行 |  |
| **增量计算**  图数据由原始数据和增量数据组成 | Percolator |  |  |  |  |
| Kineograph |  | Vertex-centric | 变化传播 |  |
| DryadInc |  |  | 结果缓存复用 |  |
| CBP |  |  |  | 建立在Hadoop平台上 |
| Incoop |  |  |  |
| IncMR |  |  |  |

# 科研总结

1. 前期的调研工作准备不够充分，现在已经存在很多系统，他们都或多或少的解决了各种问题，没有总结他们的特性、方法、技术和实现细节，就无法快速抓住重点。=> summary related works is important.

# 附录

## Spark GraphX

## Flink Gelly

## GraphLab

参考资料： 百度百科

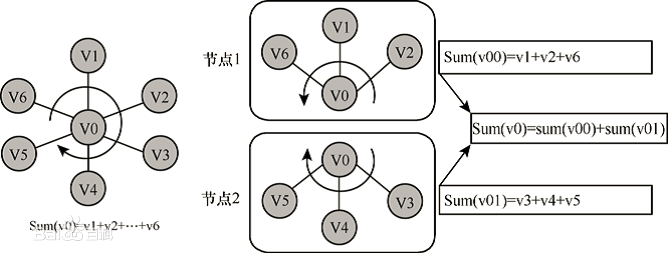
GraphLab 的设计目标是在集群或者多处理机的单机系统上实现大规模的机器学习算法。一般的机器学习类算法有以下两个特性。

● 数据依赖性很强。运算过程中参与计算的各个机器之间经常需要交换大量的数据。

● 流处理复杂。主要表现在整个处理过程需要反复地迭代计算，数据处理分支很多，很难实现真正的并行。

该框架是面向机器学习（ML）的流处理并行计算框架，可以运行在多处理机的单机系统、集群或是亚马逊的EC2 等多种环境下。框架的设计目标是，像MapReduce一样高度抽象，可以高效执行与机器学习相关的、具有稀疏的计算依赖特性的迭代性算法，并且保证计算过程中数据的高度一致性和高效的并行计算性能。

GraphLab 将数据抽象成Graph 结构，将算法的执行过程抽象为Gather、Apply、Scatter 三个步骤（类似于MapReduce 中的map 和reduce 阶段），其并行的核心思想是对图的顶点的划分，如下图 所示。



2012 年CMU 发布了GraphLab2，GraphLab2 在GraphLab1 的基础上对程序并行执行的性能有了较大的提升，GraphLab2 将程序的执行过程抽象为3 个基本的操作，即G（gather）、A（apply）、S（scatter），每个顶点每一轮迭代都要按照顺序经过gather→apply→scatter 这3 个阶段。

（1）gather 阶段

工作顶点从邻接顶点收集信息，对从邻接点收集的数据被GraphLab 进行求和运算。该阶段所有的顶点和边数据都是只读的。

（2）apply 阶段

各个从节点将gather 节点计算得到的求和值发送到master 节点上，master进行汇总得到总的和，然后Master 再根据业务需求执行一系列计算，更新工作顶点的值。该阶段顶点可修改，边不可修改。

（3）scatter 阶段

工作顶点更新了自己的值后，根据需要可以更新顶点相邻的边信息，并且通知依赖该工作顶点的顶点更新自己的状态。该阶段顶点只读，边数据可写。