14 并行图算法

- 大纲
  - 一些重要和基本的图算法的并行公式
    - 图论在CS中扮演着重要角色
    - 它提供了一种简单而系统的方法来模拟许多问题
    - 许多问题可以用图来表示，并且可以使用标准的图算法来解决
- 最小生成树：Prim's 算法
  - 最小生成树
    - 无向图 G 的生成树是 G 的子图，它是包含 G 的所有顶点的树。
    - 在加权图中，子图的权重是子图中边的权重之和。
    - 加权无向图的最小生成树 (MST) 是具有最小权重的生成树
  - Prim's 算法
    - Prim 寻找 MST 的算法是一种贪心算法。
    - 首先选择一个任意顶点，将其包含到当前 MST 中。
    - 通过将最接近当前 MST 中已经存在的顶点之一的顶点插入到当前 MST 中来增长当前 MST。
  - 并行化
    - 该算法在 n 次外部迭代中工作 - 很难同时执行这些迭代。
    - 内循环比较容易并行化
    - 令 p 为进程数，令 n 为顶点数。
    - 邻接矩阵以一维块的方式划分，距离向量 d 相应地划分。
    - 在每个步骤中，处理器选择本地最近的节点，然后进行全局归约以选择全局最近的节点。
    - 这个节点被插入到 MST 中，并且选择广播到所有处理器。 每个处理器在本地更新它的 d 向量部分。
    - **单词循环的时间，n个循环还要×n**：$n^2/p + nlogp$

      > $O(n/p + \log p)$.
      >
      > 计算　　→通信
- 单源最短路径：Dijkstra’s算法
  - 对于加权图 G = (V,E,w)，单源全汇最短路径问题是找到从顶点 v 到 V 中所有其他顶点的最短路径
  - Dijkstra 的算法类似于 Prim 的算法。 它维护一组已知最短路径的节点。

- 它使用当前最短路径集中的节点之一基于最接近源的节点来增长该集合。
- 并行化
  - 使用一维块映射对加权邻接矩阵进行分区
  - 每个进程在本地选择离源最近的节点，然后进行全局归约以选择下一个节点。
  - 该节点被广播到所有处理器并更新 l 向量。
  - 性能和Prim一样

- all-pair最短路径——图中任意两点的最短路径
  - 执行n次单源，时间复杂度是$O(n^3)$
  - 两个并行化策略
    - （源分区）在不同的处理器上执行 n 个最短路径问题中的每一个，或者
    - （源并行）使用最短路径问题的并行公式来增加并发性
  - Dijkstra 算法
    - 分析：
      - 源分区
        - 使用p个进程，p小于等于n，顶点数
        - 不需要进程间的通信，假设每个进程中都有一个邻接矩阵副本

          The parallel runtime is $\Theta(n^3/p)$

          The maximum #proc is $\Theta(n)$

          - **with runtime is $\Theta(n^2)$**
      - 源并行
        - p大于n

          Using previous results, this takes total time:

          $$T_P = \overbrace{\Theta\left(\frac{n^3}{p}\right)}^{\text{computation}} + \overbrace{\Theta(n\log p)}^{\text{communication}}.$$

          $T_o = pW_p - W$

          $= pT_p - W = O(pn\log p)$

          - **($\log p$ is a pessimistic estimate of the broadcasting time)**

          The maximum #proc is $\Theta(n^2/\log n)$

          - **with runtime $\Theta(n \log n)$** $= \frac{O(n^3)}{O(n^2/\log n)}$
          - **when the efficiency is not degraded**

          $$S = \frac{\Theta(n^3)}{\Theta(n^3/p) + \Theta(n\log p)}$$

          $$E = \frac{1}{1 + \Theta((p\log p)/n^2)}$$

          令$T_o = W$ #proc最大

          . Grama et al., “Introduction to Parallel Computing,” Addison Wesley, 2003

- Floyd's算法PPT26，并行

> > $$T_P = \overbrace{\Theta\left(\frac{n^3}{p}\right)}^{\text{computation}} + \overbrace{\Theta\left(\frac{n^2}{\sqrt{p}}\log p\right)}^{\text{communication}}.$$

- 连通图PPT34
  - 先将图划分给每个进程，找子图，之后合并