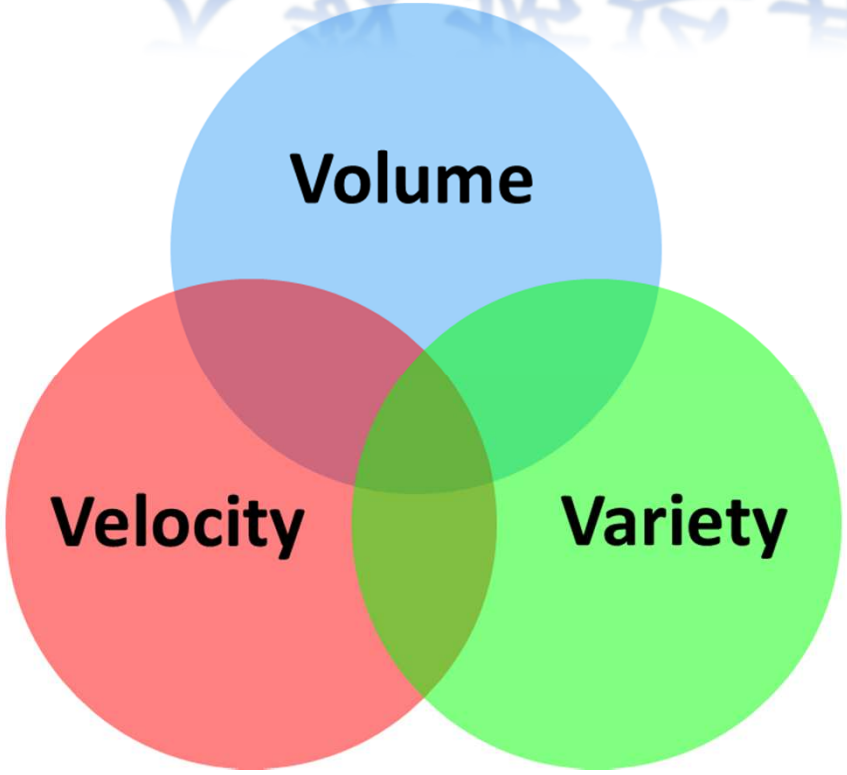
大数据系统与大规模数据分析

作业2：
大数据运算系统编程



陈世敏

中科院计算所
计算机体系结构
国家重点实验室

课程相关

- 成绩分配
  - 闭卷考试：45%
  - 作业1+作业2+作业3：30%
  - 大作业：20%
  - 课堂表现：5%

作业时间安排

周次	内容	作业
第4周，3/11	大数据存储系统1：基础，文件系统，HDFS	作业1布置
第5周，3/18	大数据存储系统2：键值系统	
第6周，3/25	大数据存储系统3：图存储，document store	
第7周，4/1	大数据运算系统1：MapReduce, 图计算系统	作业1提交 作业2布置
第8周，4/8	大数据运算系统2：图计算系统，MR+SQL	
第9周，4/15	大数据运算系统3：内存计算系统	大作业布置
第10周，4/22	最邻近搜索和位置敏感（LHS）哈希算法	作业2提交
第11周，4/29	数据空间的维度约化	
第12周，5/6	推荐系统	作业3
第13周，5/13	流数据采样与估计、流数据过滤与分析	
第14周，5/20	教育大数据的建模与分析	
第15周，5/27	期末考试	
第16周，6/3	大作业验收报告	大作业验收

作业2安排

- 成绩：占总成绩10%
- 时间
  - 发布：2020/4/1 (Wed)前
  - 上交：**2020/4/22 (Wed)，北京时间 6:59pm（共3周）**
  - 在课程系统中提交
    - 组号_学号_hw2.java 对应MapReduce程序
    - 组号_学号_hw2.cc 对应同步图运算程序
  - 晚交
    - 最晚：2020/4/29(Wed)，北京时间 6:59pm，将扣除20%成绩
    - 之后不再接收，作业2成绩为0
- 抄袭：课程总分为0！

分组（与作业1不同）

- 共分为**4个组**，每个组的作业题目有一定区别
- 分组方式如下
  - 组号 = (学号最右面6位数字) % 4
  - %是求余数
- 举例
  - 学号最右面6位数字=229032
  - 组号=229032 % 4 = 0
  - 所以是第0组

作业内容

- 目的
  - 学习Hadoop编程
  - 学习同步图运算的编程
- 分为两个部分（共10%）
  - Hadoop编程（5%）
    - 所有组的作业内容相同
  - 同步图运算编程（5%）
    - 分成4组，每个组实现不同的图运算

Hadoop编程

- 输入文件：文本文件
  - 每行格式
    - <source> ␣ <destination> ␣ <time>
    - 3个部分由空格隔开
    - 其中source和destination为两个字符串，内部没有空格
    - time为一个浮点数，代表时间（秒为单位）
    - 涵义：可以表示一次电话通话，或表示一次网站访问等
  - 输入可能有噪音
    - 如果一行不符合上述格式，应该被丢弃，程序需要正确执行
- MapReduce计算：统计每对source-destination的信息
- 输出
  - <source>␣<destination> ␣ <count> ␣ <average time>
  - 每一个source-destination组合输出一行（注意：顺序相反按不同处理）
  - 每行输出通话次数和通话平均时间(保留3位小数，例如2.300)

同步图运算

- Group 0: SSSP
- Group 1: KCore
- Group 2: Graph Coloring
- Group 3: Directed Triangle Counting

下载并安装GraphLite

- https://github.com/schencoding/GraphLite

下载GraphLite-0.20

根据GraphLite/**GraphLite-0.20**/README.txt

- 安装、编译graphlite
- 运行PageRank的例子
- 读一些header文件，了解具体的接口函数和类定义

SSSP

- Single Source Shortest Path
  - 给定一个顶点V0
  - 求V0到其它每个顶点的最短路
- 计算方法
  - 每个顶点Vertex Value记录当前已知的最短路长度
  - 初始化：V0: 0; 其它顶点: 无穷大
  - 迭代
    - 发送的消息：当前顶点的最短路长度+出边长度
    - 收到消息后，更新当前最短路长度值
- 输入：图，V0 (命令行参数)
- 输出：顶点ID，最短路长度

顶点ID: 最短路长度
顶点ID: 最短路长度
…

KCore

- KCore
  - 一个图G的 KCore 是G的子图
  - 这个子图的每个顶点的度≥K
- 计算方法
  - 每个顶点记录: is_deleted, current_degree
  - 如果顶点的度小于k，从图中删除该顶点，然后给邻居发送消息
  - 顶点收到消息后，得知被删掉的邻居顶点，更新自己的度
- 输入：无向图（有成对的有向边）
- 输出: KCore 子图中的所有顶点

  顶点

  顶点

  …

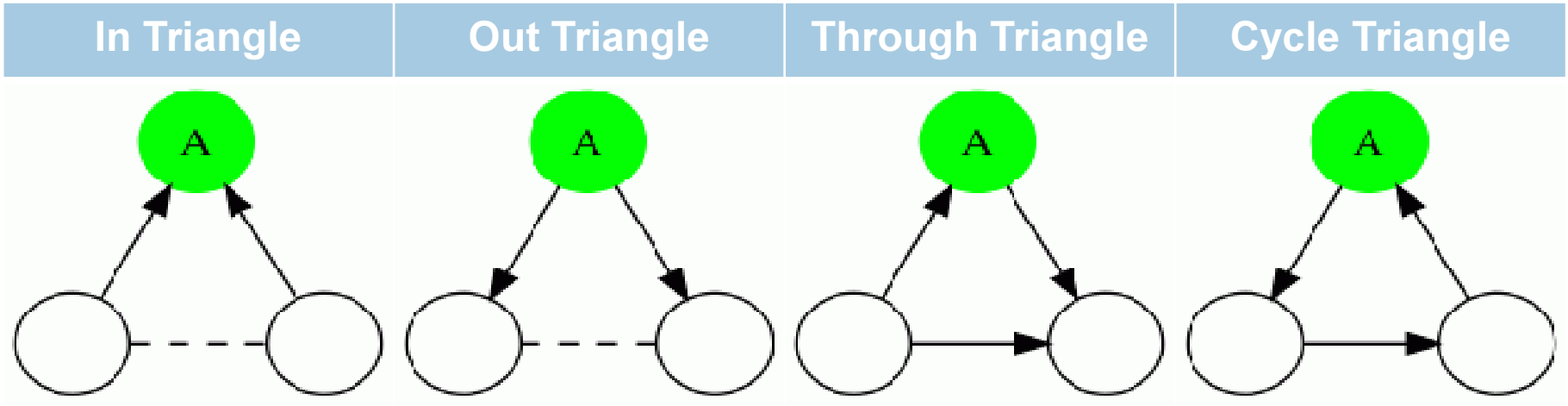
Graph Coloring

- Graph Coloring
  - 对图的顶点着色，相邻顶点不同颜色，给出一种着色方案。
  - 假设可用color数比实际需要的最小数大很多
- 计算方法
  - 每个顶点记录自己的color，初始为-1
  - Superstep = 0，顶点V0着色color=0，向邻居发送颜色编号
  - 接下来的 superstep 中，顶点收到消息后，统计邻居顶点的颜色，随机选择一个与之不冲突的颜色号着色
- 输入：无向图（有成对的有向边），命令行：V0，总color数
- 输出：

  顶点id： 颜色号
  顶点id： 颜色号
  …

Directed Triangle Counting (1)

- 有向图三角形计数：
  - 三角形类型（虚线表示箭头方向任意）



  - 从单个顶点角度，计算各类型三角形数
  - 累计上述计数，求各类型三角形的总数（会有多次计数）

Directed Triangle Counting (2)

- 思路
  - A知道每个邻居的所有邻居，就可以计算上述的in/out/through/cycle triangle个数
  - 如何获得邻居的邻居？
    - 每个顶点可以知道自己的out-neighbor
    - 经过一次超步通信，每个顶点可以知道自己的in-neighbor
    - 那么每个顶点都可以把in-neighbor和out-neighbor，发给邻居
  - 发送消息
    - 消息是定长的，可以发多条消息
  - 使用aggregate统计最终的triangle个数

Directed Triangle Counting (2)

- 输入：有向图

- 输出：in/out/through/cycle triangle个数

  in: 个数

  out: 个数

  through: 个数

  cycle: 个数