



# 基于 Spatial-Spark 海量网络空间数据分析与挖掘

---

姓名: 高 峰

导师: 高井祥

孙久运

专业: 大地测量学与测量工程

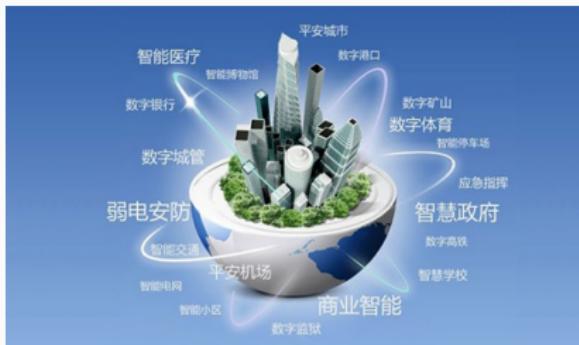
# 提纲

1. 绪论
2. 相关技术
3. Spatial-Spark
4. POI 空间分析
5. 人口流动网络分析
6. 总结

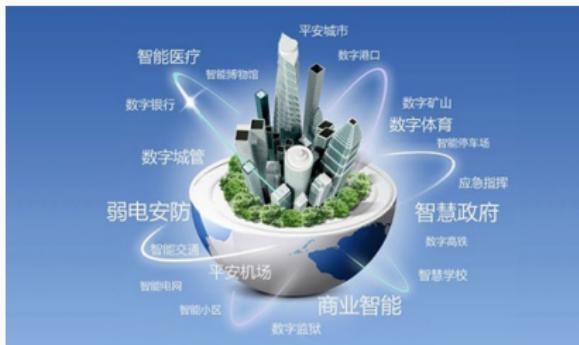
# 绪论

---

## 智慧城市： 数字城市、物联网、云计算



## 智慧城市： 数字城市、物联网、云计算



### 数字城市

- 空间信息快速获取技术
- 海量空间数据管理技术
- 空间信息可视化技术
- 空间数据分析挖掘技术
- 网络服务技术

## 移动互联网发展

LBS 应用 打车软件、O2O 软件、社交网络软件

# 绪论

移动互联网发展

LBS 应用 打车软件、O2O 软件、社交网络软件

意义

# 绪论

## 移动互联网发展

LBS 应用 打车软件、O2O 软件、社交网络软件

## 意义

- 对个人而言 (智慧生活)

## 移动互联网发展

LBS 应用 打车软件、O2O 软件、社交网络软件

## 意义

- 对个人而言 (智慧生活)
- 对商业公司而言 (智慧商业)

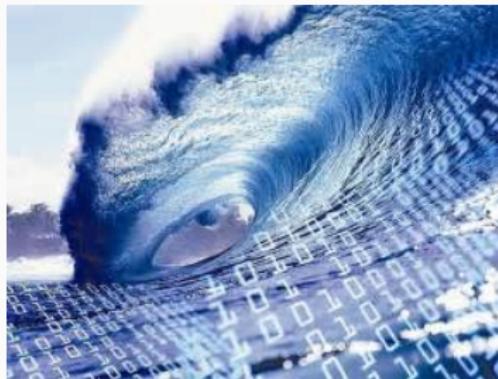
## 移动互联网发展

LBS 应用 打车软件、O2O 软件、社交网络软件

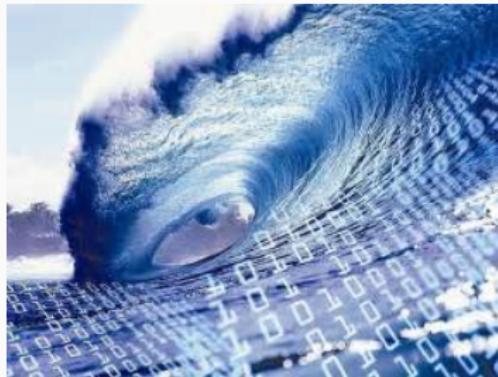
## 意义

- 对个人而言 (智慧生活)
- 对商业公司而言 (智慧商业)
- 对政府决策部门而言 (智慧政府)

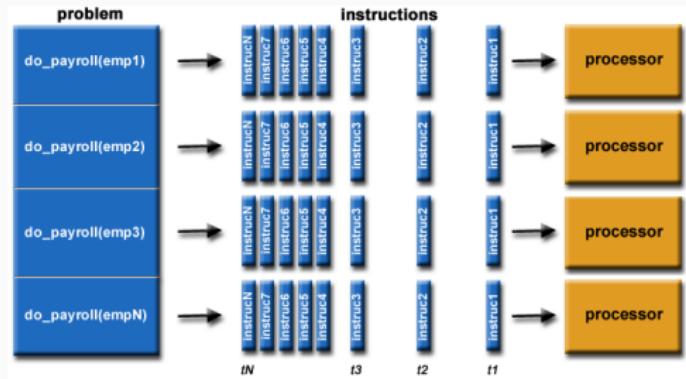
## 海量数据处理



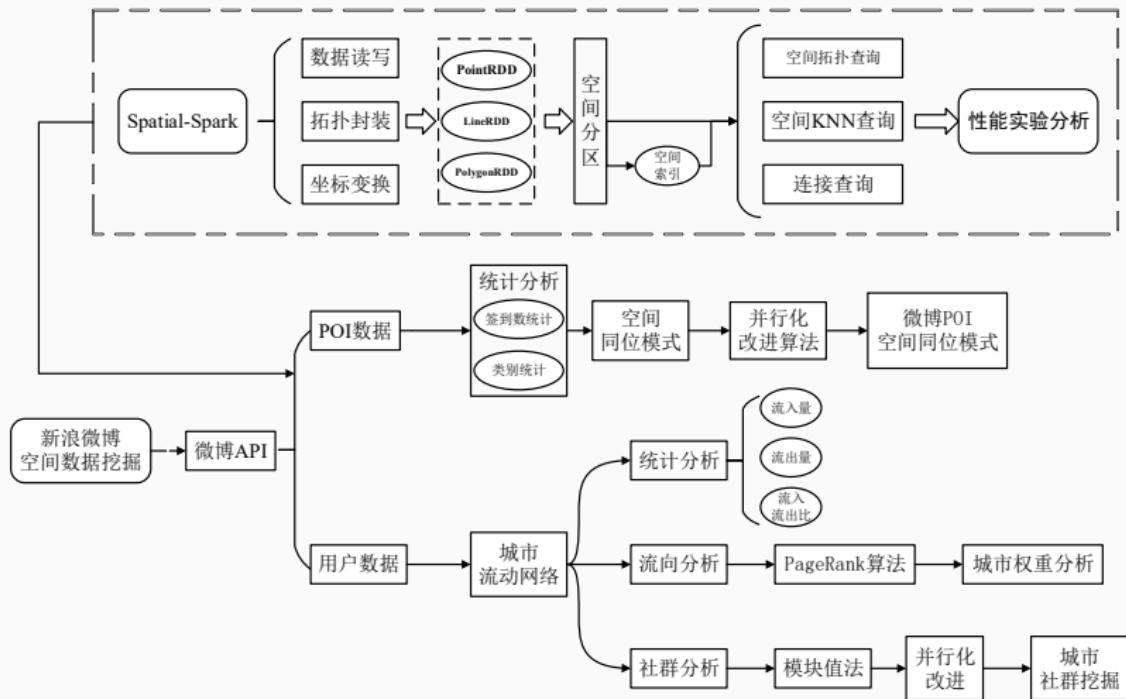
## 海量数据处理



## 并行计算



# 绪论



## 相关技术

---

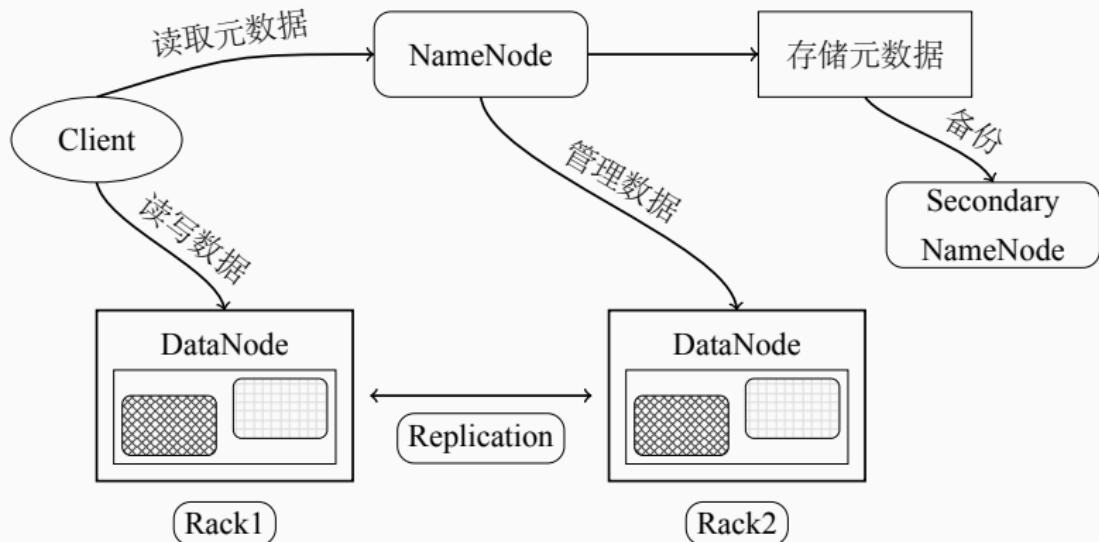
## 相关技术 (Hadoop)

**Hadoop** 是 Apache 基金会推出的大数据计算平台

# 相关技术 (Hadoop)

Hadoop 是 Apache 基金会推出的大数据计算平台

HDFS(Hadoop Distributed File System)

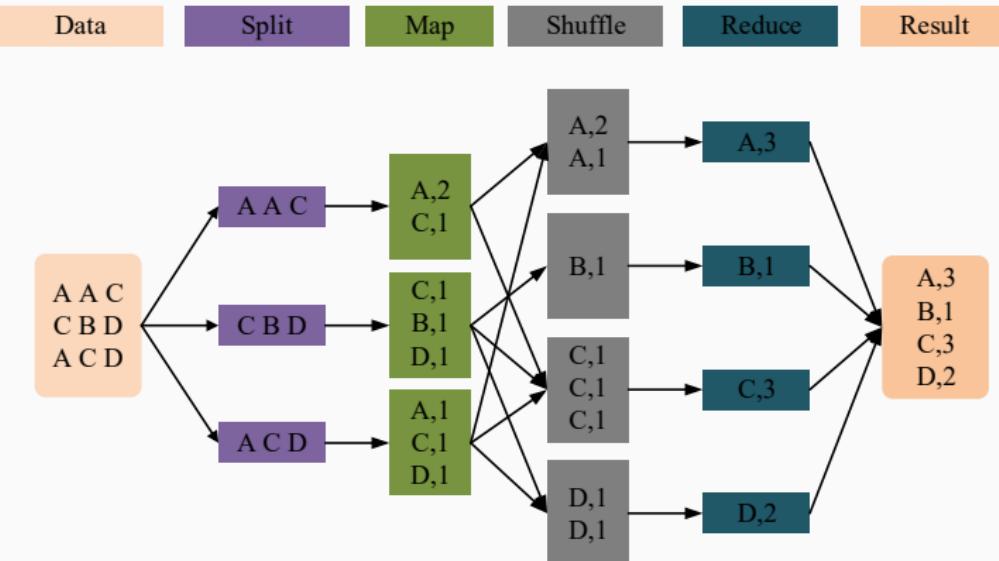


# 相关技术 (Hadoop)

## Hadoop MapReduce

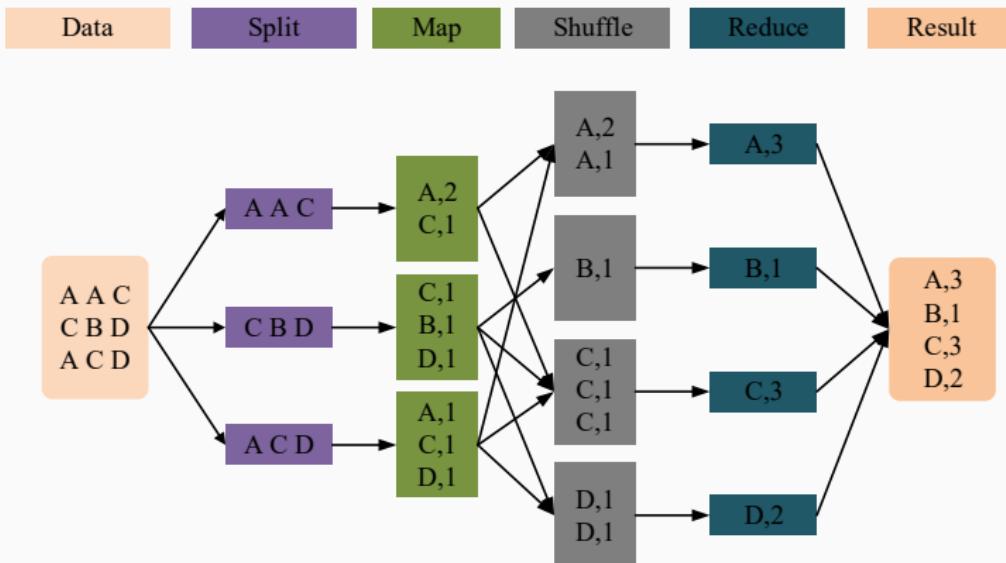
# 相关技术 (Hadoop)

## Hadoop MapReduce



# 相关技术 (Hadoop)

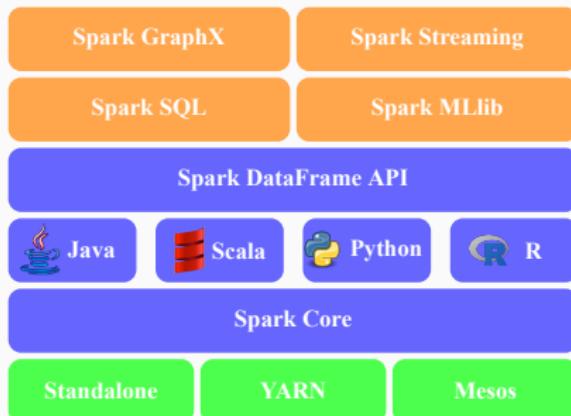
## Hadoop MapReduce



**缺陷:** 单点故障、算子抽象、IO 操作频繁

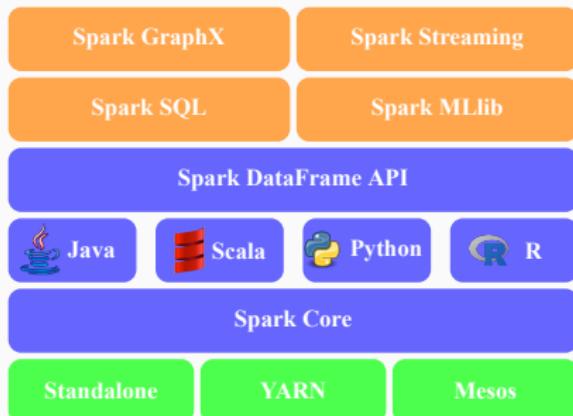
# 相关技术 (Spark)

## Spark 技术生态系统



# 相关技术 (Spark)

## Spark 技术生态系统



- **Spark SQL:** 类 SQL 语言查询结构化数据
- **Spark MLlib:** 常用机器学习算法 Spark 实现
- **Spark GraphX:** 分布式图及其计算 Spark 实现
- **Spark Streaming:** Spark 流应用计算框架

## 相关技术 (Spark)

弹性分布式数据集 (RDD) 是 Spark 的核心抽象，在内存中已被分区、只读的、并提供一组丰富的操作方式的数据集合

## 相关技术 (Spark)

弹性分布式数据集 (RDD) 是 Spark 的核心抽象，在内存中已被分区、只读的、并提供一组丰富的操作方式的数据集合

### RDD 算子

- Transformation
- Action

## 相关技术 (Spark)

Spark 一栈式解决方案的**优势**:

- 快速处理，内存计算
- 通用性，技术方案无缝集成
- 与 Hadoop 集群集成

## 相关技术 (Spark)

Spark 一栈式解决方案的**优势**:

- 快速处理，内存计算
- 通用性，技术方案无缝集成
- 与 Hadoop 集群集成

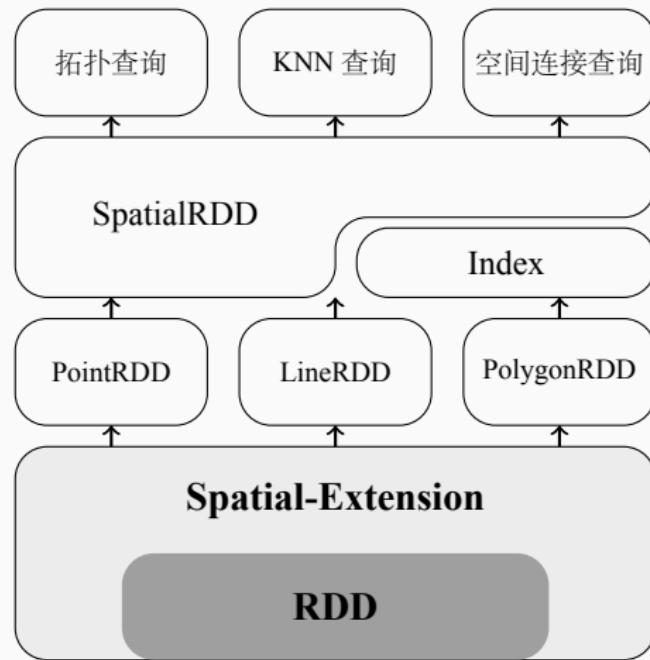
Spark 在处理海量空间数据**劣势**:

- 不支持空间数据类型
- 不支持空间操作
- 没有空间数据优化

# Spatial-Spark

---

## Spatial-Spark 体系结构



## Proj.4

- 空间椭球基础
- 空间参照系统
- 空间投影系统
- 坐标换算

## Proj.4

- 空间椭球基础
- 空间参照系统
- 空间投影系统
- 坐标换算

## JTS Topogy Suite

- 简单空间要素类型
- 空间数据分析和空间运算
- 空间数据读写

# RDD 空间扩展

PointRDD、LineRDD 和 PolygonRDD

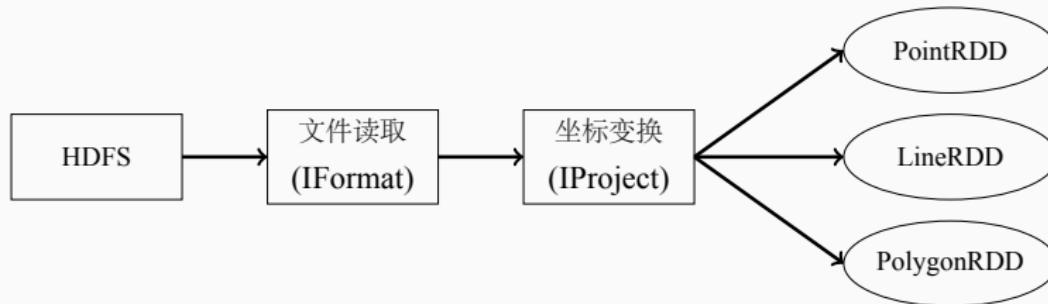
Spatial-RDD

# RDD 空间扩展

PointRDD、LineRDD 和 PolygonRDD

Spatial-RDD

- HDFS 中读取

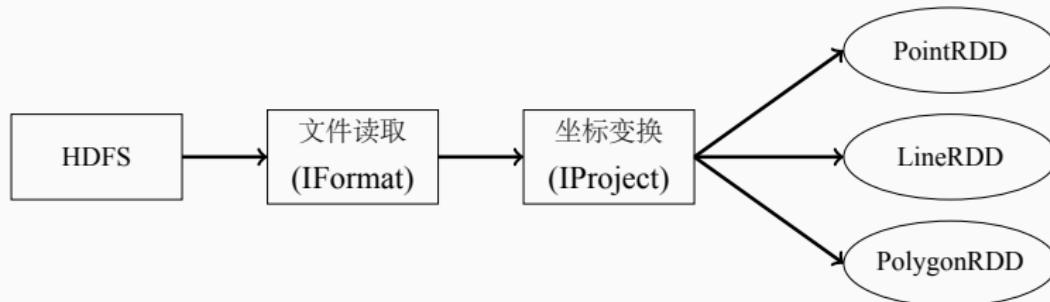


# RDD 空间扩展

PointRDD、LineRDD 和 PolygonRDD

Spatial-RDD

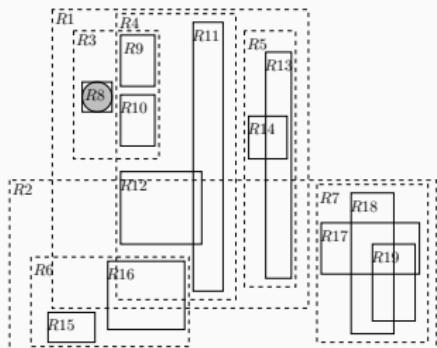
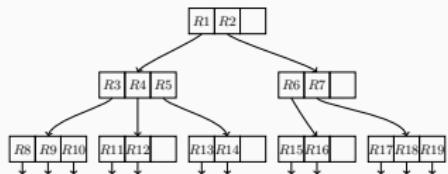
- HDFS 中读取



- Spatial-RDD 空间运算相互转换

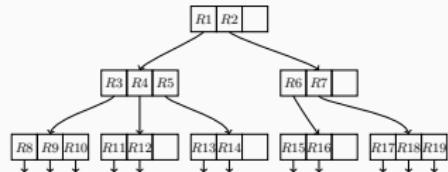
# 空间索引

## R 树空间索引

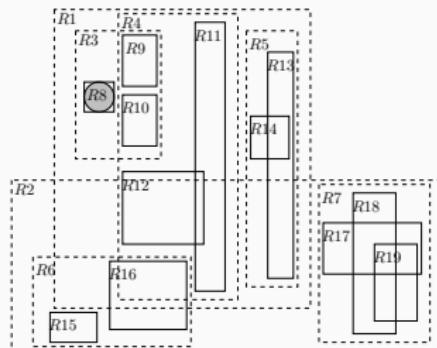


# 空间索引

## R 树空间索引

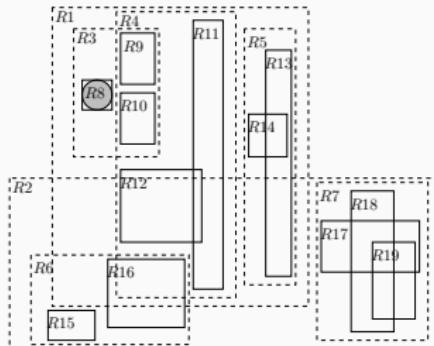
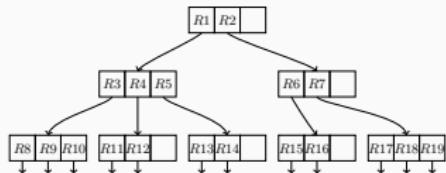


### ■ R 树插入



# 空间索引

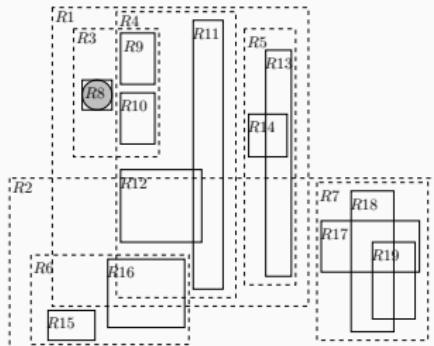
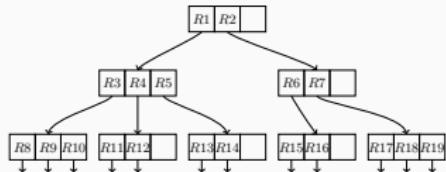
## R 树空间索引



- R 树插入
- R 树查询

# 空间索引

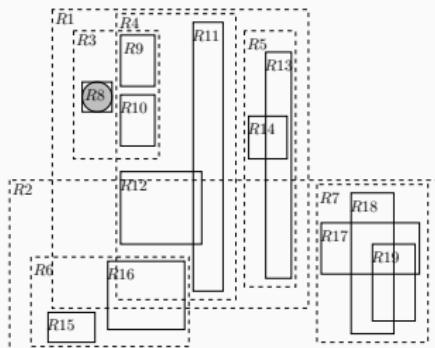
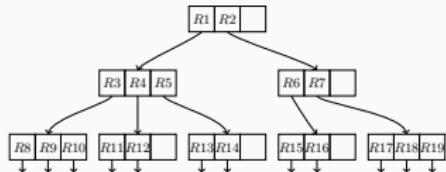
## R 树空间索引



- R 树插入
- R 树查询
  - 过滤

# 空间索引

## R 树空间索引



- R 树插入
- R 树查询
  - 过滤
  - 精选

# 空间查询

- 空间拓扑查询

支持 Spatial-RDD 之间的所有拓扑操作

# 空间查询

- 空间拓扑查询

支持 Spatial-RDD 之间的所有拓扑操作

- 空间 KNN 查询

借助优先级队列，筛选  $K$  个邻居；可借助索引加速空间查询

- 空间拓扑查询

支持 Spatial-RDD 之间的所有拓扑操作

- 空间 KNN 查询

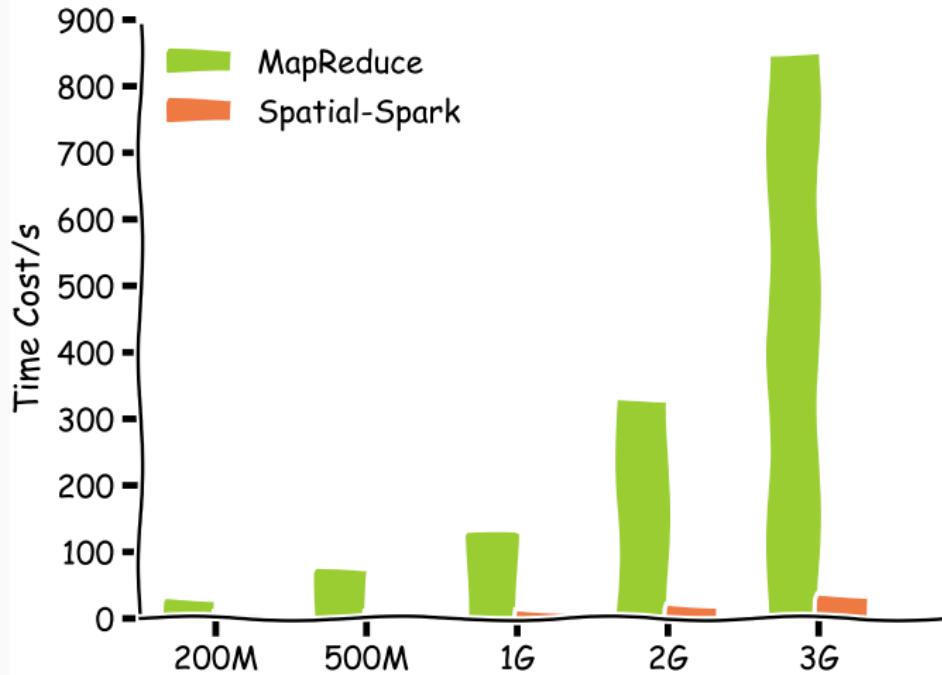
借助优先级队列，筛选  $K$  个邻居；可借助索引加速空间查询

- 空间连接查询

过滤步骤；精选步骤

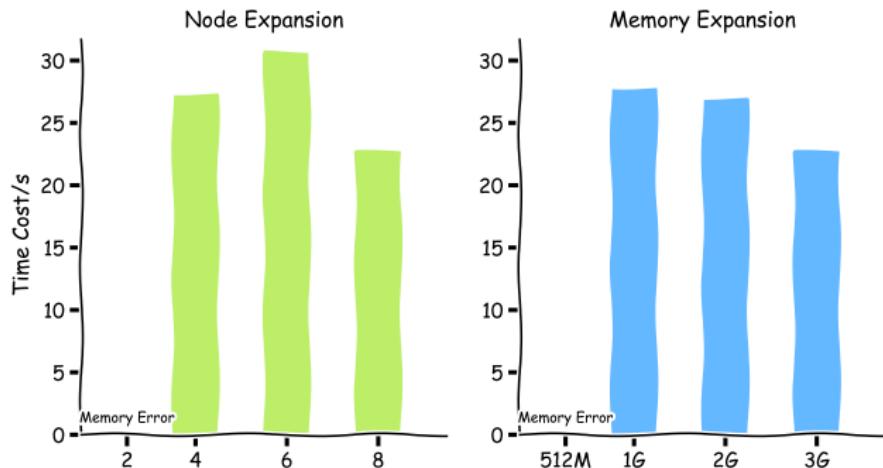
# 实验分析

MapReduce 与 Spatial-Spark 空间过滤筛选对比



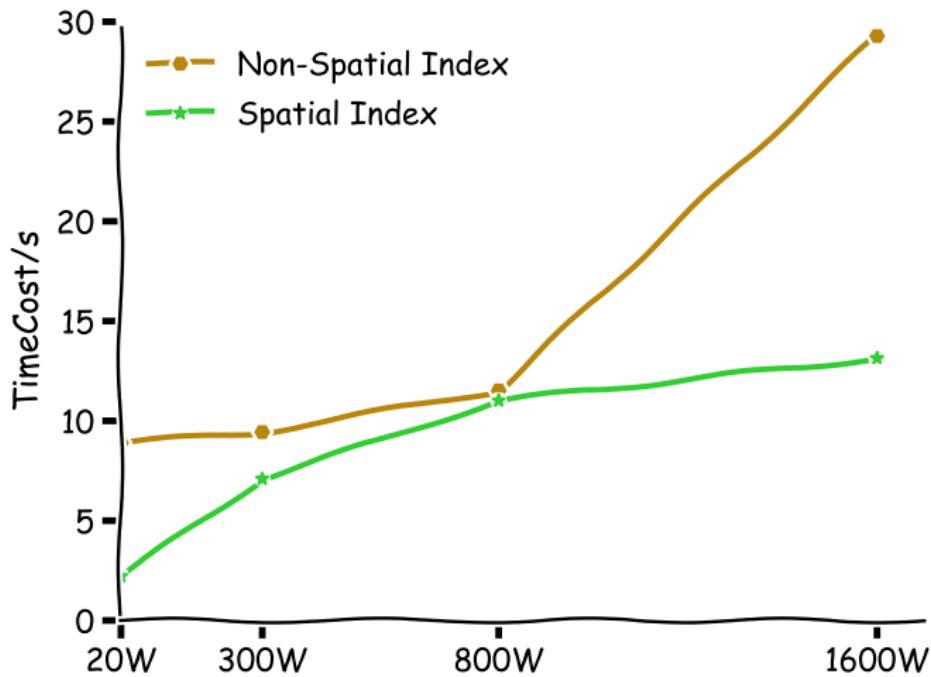
# 实验分析

## Spatial-Spark 集群扩展性能分析



# 实验分析

## Spatial-Spark 空间索引性能分析

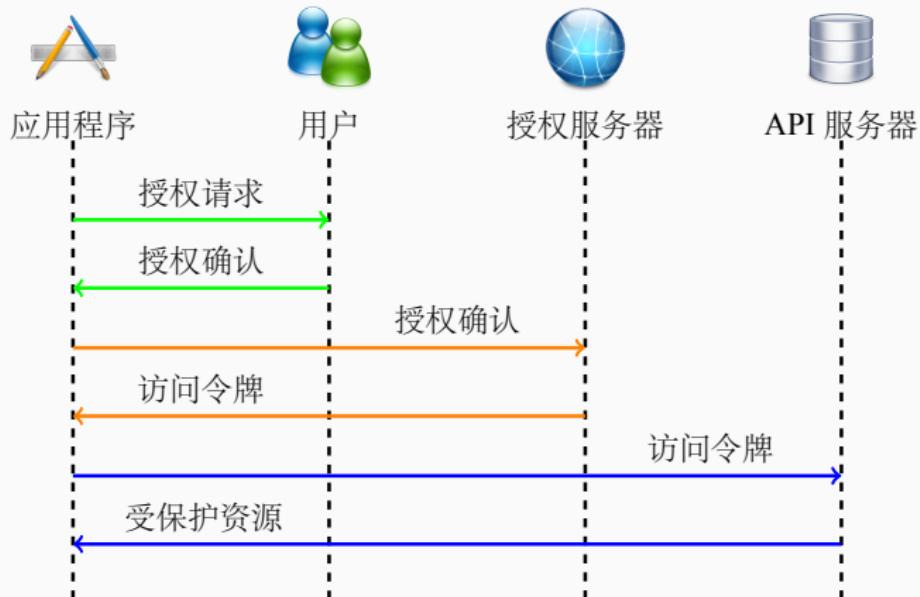


## POI 空间分析

---

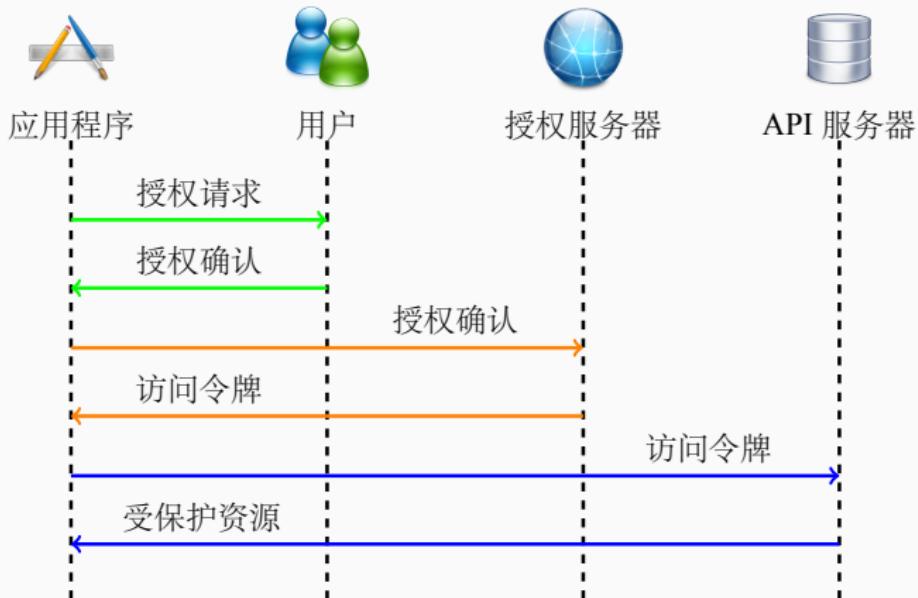
# 微博数据接口

新浪微博提供了 API 方便第三方程序访问微博丰富的数据资源



# 微博数据接口

新浪微博提供了 API 方便第三方程序访问微博丰富的数据资源



HTTP Get

# POI 空间分析 (获取)

POI 是新浪微博用户在使用过程中自行添加附近的热门位置

The screenshot shows a Weibo profile for a POI. At the top left is a thumbnail image of a modern building. To its right are three buttons: a white one with a pen icon and the text '写短评' (Write a short comment), a light blue one with a plus sign and the text '+ 关注' (Follow), and a light blue one with a diamond icon and the text '2915 赞' (Likes). Below these are five tabs: '主页' (Home), '我的周边' (Nearby), '相册' (Album), '签到的人' (People who checked in), and '附近热门' (Nearby hot spots). The '主页' tab is highlighted with an orange underline.

**简介**

类型: 高校中心点  
地点: 徐州市泉山区大学路1号  
简介: 51路

**此地热图**

1/50

**入驻商家**

中国矿业大学

11957 热图 | 2915 赞 | 24904 热议

+ 关注

**地图** 查看大图

# POI 空间分析 (获取)

place/nearby/pois

- **Lat:** 纬度
- **Long:** 经度
- **Range:** 查询半径

# POI 空间分析 (获取)

- 投影

Lambert 投影

place/nearby/pois

- **Lat:** 纬度
- **Long:** 经度
- **Range:** 查询半径

# POI 空间分析 (获取)

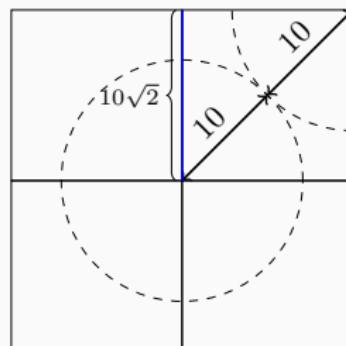
place/nearby/pois

- **Lat:** 纬度
- **Long:** 经度
- **Range:** 查询半径

- 投影

Lambert 投影

- 栅格化



# POI 空间分析 (获取)

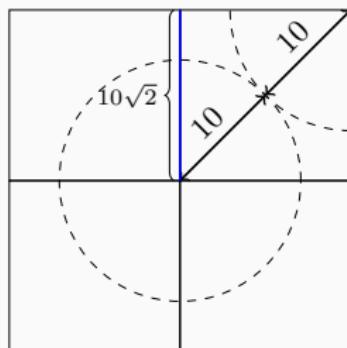
place/nearby/pois

- **Lat:** 纬度
- **Long:** 经度
- **Range:** 查询半径

- 投影

Lambert 投影

- 栅格化



- 坐标反算

适应 API 接口

# POI 空间分析 (统计)

热门签到地点: sortBy, take

# POI 空间分析 (统计)

热门签到地点: sortBy, take

1. 星光公益站
2. 浦东机场
3. 厦门高崎国际机场
4. 中关村
5. 深圳宝安机场
6. 丽江古城
7. 成都双流国际机场
8. 望京
9. 首都机场 T3 航站楼
10. 广州白云机场

# POI 空间分析 (统计)

热门签到地点: sortBy, take

1. 星光公益站
2. 浦东机场
3. 厦门高崎国际机场
4. 中关村
5. 深圳宝安机场
6. 丽江古城
7. 成都双流国际机场
8. 望京
9. 首都机场 T3 航站楼
10. 广州白云机场

热门签到类别: groupBy, reduce

# POI 空间分析 (统计)

热门签到地点: sortBy, take

1. 星光公益站
2. 浦东机场
3. 厦门高崎国际机场
4. 中关村
5. 深圳宝安机场
6. 丽江古城
7. 成都双流国际机场
8. 望京
9. 首都机场 T3 航站楼
10. 广州白云机场

热门签到类别: groupBy, reduce



## 关联规则算法

- 事务项  $T$ , 由集合  $I_1, I_2, \dots, I_m$  组成
- $P \in T, Q \in T$ , 规则  $P \rightarrow Q$
- 置信度约束  $c(P \rightarrow Q) \geq min\_c$
- 支持度约束  $s(P \rightarrow Q) \geq min\_s$

## 关联规则算法

- 事务项  $T$ , 由集合  $I_1, I_2, \dots, I_m$  组成
- $P \in T, Q \in T$ , 规则  $P \rightarrow Q$
- 置信度约束  $c(P \rightarrow Q) \geq min\_c$
- 支持度约束  $s(P \rightarrow Q) \geq min\_s$

## Apriori 算法

- 频繁项的子集必须也是频繁的
- 非频繁项的超集必定非频繁

# POI 空间分析 (同位模式)

## 单主题空间关联规则

$$P_1 \wedge P_2 \wedge \dots \wedge P_m \rightarrow Q_1 \wedge Q_2 \wedge \dots \wedge Q_n(s\%, c\%) \quad (1)$$

# POI 空间分析 (同位模式)

## 单主题空间关联规则

$$P_1 \wedge P_2 \wedge \dots \wedge P_m \rightarrow Q_1 \wedge Q_2 \wedge \dots \wedge Q_n (s\%, c\%) \quad (1)$$

## 多主题同位模式

$$R(a_1, b_1) \Leftrightarrow (\text{distance}(a_1, b_1) \leq d) \quad (2)$$

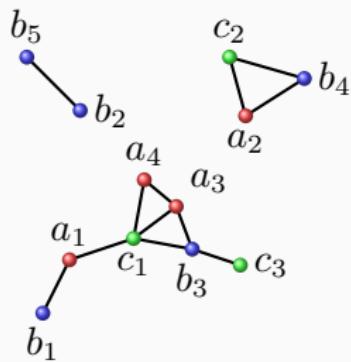
# POI 空间分析 (同位模式)

## 单主题空间关联规则

$$P_1 \wedge P_2 \wedge \dots \wedge P_m \rightarrow Q_1 \wedge Q_2 \wedge \dots \wedge Q_n(s\%, c\%) \quad (1)$$

## 多主题同位模式

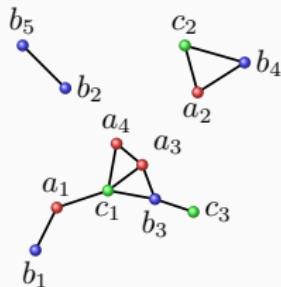
$$R(a_1, b_1) \Leftrightarrow (\text{distance}(a_1, b_1) \leq d) \quad (2)$$



# POI 空间分析 (同位模式)

## 参与率

$$PR(c, f_i) = \frac{|\pi_{f_i}(\text{table\_instance}(c))|}{|\text{table\_instance}(\{f_i\})|} \quad (3)$$



$$PR(\{A, B, C\}, A) = 2/4 = 0.5$$

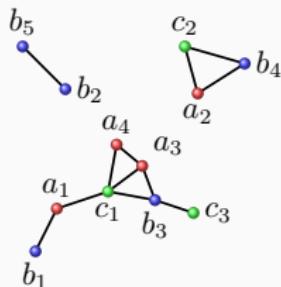
$$PR(\{A, B, C\}, B) = 2/5 = 0.4$$

$$PR(\{A, B, C\}, C) = 2/3 = 0.67$$

# POI 空间分析 (同位模式)

参与率

$$PR(c, f_i) = \frac{|\pi_{f_i}(\text{table\_instance}(c))|}{|\text{table\_instance}(\{f_i\})|} \quad (3)$$



$$PR(\{A, B, C\}, A) = 2/4 = 0.5$$

$$PR(\{A, B, C\}, B) = 2/5 = 0.4$$

$$PR(\{A, B, C\}, C) = 2/3 = 0.67$$

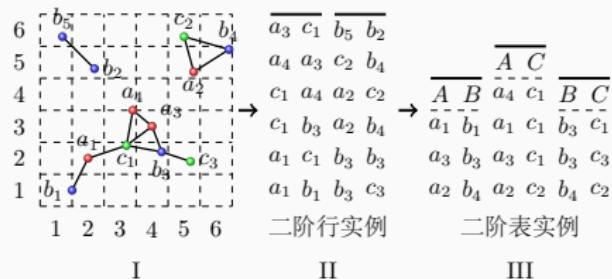
参与度

$$PI(c) = \min_{i=1}^k \{PR(c, f_i)\} \quad (4)$$

$$PI(\{A, B, C\}) = \min(0.5, 0.4, 0.67) = 0.4$$

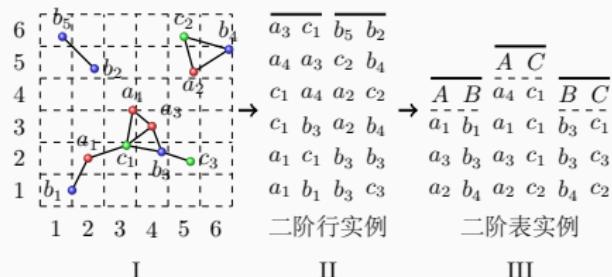
# POI 空间分析 (同位模式)

## Spatial-Saprk 二阶模式生成

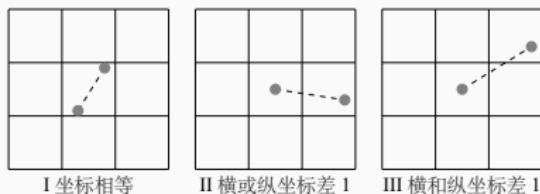


# POI 空间分析 (同位模式)

## Spatial-Saprk 二阶模式生成



## 键值相等判断



# POI 空间分析 (同位模式)

---

## 算法 co-location 算法

---

输入:

点集:points;

距离阈值:d;

参与度阈值:threshold;

输出:

空间同位模式: colocation patterns;

# POI 空间分析 (同位模式)

## 算法 co-location 算法

输入:

点集:points;

距离阈值:d;

参与度阈值:threshold;

输出:

空间同位模式: colocation patterns;

1:  $P_2 = \text{Spatial-Spark.join}(\text{points}, d, \text{threshold})$

2: while  $P_k$  is not empty and  $k < N$  do

# POI 空间分析 (同位模式)

## 算法 co-location 算法

输入:

点集:points;  
距离阈值:d;  
参与度阈值:threshold;

输出:

空间同位模式: colocation patterns;

- 1:  $P_2 = \text{Spatial-Spark.join}(\text{points}, d, \text{threshold})$
- 2: while  $P_k$  is not empty and  $k < N$  do
- 3: // Join 操作, RowPattern 重写
- 4:  $C_{k+1} = \text{gen\_candinate\_cocolation}(P_k, k)$  // $k+1$  阶候选集合

# POI 空间分析 (同位模式)

## 算法 co-location 算法

输入:

点集:points;  
距离阈值:d;  
参与度阈值:threshold;

输出:

空间同位模式: colocation patterns;

- 1:  $P_2 = \text{Spatial-Spark.join}(\text{points}, d, \text{threshold})$
- 2: while  $P_k$  is not empty and  $k < N$  do
- 3: // Join 操作, RowPattern 重写
- 4:  $C_{k+1} = \text{gen_candinate_cocolation}(P_k, k)$  //k+1 阶候选集合
- 5: // 距离阈值比较
- 6:  $C_{k+1} = \text{pruning}(C_{k+1}, d)$

# POI 空间分析 (同位模式)

## 算法 co-location 算法

输入:

点集:points;  
距离阈值:d;  
参与度阈值:threshold;

输出:

空间同位模式: colocation patterns;

- 1:  $P_2 = \text{Spatial-Spark.join}(\text{points}, d, \text{threshold})$
- 2: **while**  $P_k$  is not empty and  $k < N$  **do**
- 3:   // Join 操作, RowPattern 重写
- 4:    $C_{k+1} = \text{gen_candinate_cocolation}(P_k, k)$  //k+1 阶候选集合
- 5:   // 距离阈值比较
- 6:    $C_{k+1} = \text{pruning}(C_{k+1}, d)$
- 7:   // reduce 生成表实例
- 8:    $T_{k+1} = \text{gen_table_ins}()$ // 生成表实例

# POI 空间分析 (同位模式)

## 算法 co-location 算法

输入:

点集:points;  
距离阈值:d;  
参与度阈值:threshold;

输出:

空间同位模式: colocation patterns;

- 1:  $P_2 = \text{Spatial-Spark.join}(\text{points}, d, \text{threshold})$
- 2: while  $P_k$  is not empty and  $k < N$  do
- 3: // Join 操作, RowPattern 重写
- 4:  $C_{k+1} = \text{gen_candinate_cocolation}(P_k, k)$  //k+1 阶候选集合
- 5: // 距离阈值比较
- 6:  $C_{k+1} = \text{pruning}(C_{k+1}, d)$
- 7: // reduce 生成表实例
- 8:  $T_{k+1} = \text{gen_table_ins}()$ // 生成表实例
- 9: // threshold 筛选
- 10:  $P_{k+1} = \text{Select_Colocation_Pattern}(T_{k+1}, \text{threshold})$  // 筛选表实例

# POI 空间分析 (同位模式)

## 算法 co-location 算法

输入:

点集:points;  
距离阈值:d;  
参与度阈值:threshold;

输出:

空间同位模式: colocation patterns;

- 1:  $P_2 = \text{Spatial-Spark.join}(\text{points}, d, \text{threshold})$
- 2: while  $P_k$  is not empty and  $k < N$  do
- 3: // Join 操作, RowPattern 重写
- 4:  $C_{k+1} = \text{gen_candinate_cocolation}(P_k, k)$  //k+1 阶候选集合
- 5: // 距离阈值比较
- 6:  $C_{k+1} = \text{pruning}(C_{k+1}, d)$
- 7: // reduce 生成表实例
- 8:  $T_{k+1} = \text{gen_table_ins}()$ // 生成表实例
- 9: // threshold 筛选
- 10:  $P_{k+1} = \text{Select_Colocation_Pattern}(T_{k+1}, \text{threshold})$  // 筛选表实例
- 11:  $k = k + 1$  // 下一轮迭代

# POI 空间分析 (同位模式)

## 算法 co-location 算法

输入:

点集:points;  
距离阈值:d;  
参与度阈值:threshold;

输出:

空间同位模式: colocation patterns;

- 1:  $P_2 = \text{Spatial-Spark.join}(\text{points}, d, \text{threshold})$
- 2: while  $P_k$  is not empty and  $k < N$  do
- 3: // Join 操作, RowPattern 重写
- 4:  $C_{k+1} = \text{gen_candinate_cocolation}(P_k, k)$  //  $k+1$  阶候选集合
- 5: // 距离阈值比较
- 6:  $C_{k+1} = \text{pruning}(C_{k+1}, d)$
- 7: // reduce 生成表实例
- 8:  $T_{k+1} = \text{gen_table_ins}()$  // 生成表实例
- 9: // threshold 筛选
- 10:  $P_{k+1} = \text{Select_Colocation_Pattern}(T_{k+1}, \text{threshold})$  // 筛选表实例
- 11:  $k = k + 1$  // 下一轮迭代
- 12: end while

# POI 空间分析 (同位模式)

## 算法 co-location 算法

输入:

点集:points;  
距离阈值:d;  
参与度阈值:threshold;

输出:

空间同位模式: colocation patterns;

```
1:  $P_2 = \text{Spatial-Spark.join}(\text{points}, d, \text{threshold})$ 
2: while  $P_k$  is not empty and  $k < N$  do
3:   // Join 操作, RowPattern 重写
4:    $C_{k+1} = \text{gen_candinate_cocolation}(P_k, k)$  // k+1 阶候选集合
5:   // 距离阈值比较
6:    $C_{k+1} = \text{pruning}(C_{k+1}, d)$ 
7:   // reduce 生成表实例
8:    $T_{k+1} = \text{gen_table_ins}()$  // 生成表实例
9:   // threshold 筛选
10:   $P_{k+1} = \text{Select_Colocation_Pattern}(T_{k+1}, \text{threshold})$  // 筛选表实例
11:   $k = k + 1$  // 下一轮迭代
12: end while
13: return  $P_2, \dots, P_k$ 
```

# POI 空间分析 (同位模式)

上海市

(高等院校, 校园生活):0.556

(高等院校, 日本料理):0.472

(高等院校, 西餐厅):0.454

(高等院校, 甜品店):0.451

(高等院校, 培训机构):0.446

(高等院校, 医院):0.443

(高等院校, 糕饼店):0.441

(高等院校, 餐饮美食):0.432

武汉市

(高等院校, 校园生活):0.531

(高等院校, 图书馆):0.528

(高等院校, 糕饼店):0.384

(高等院校, 快餐厅):0.368

(高等院校, 四川菜):0.361

(高等院校, 火锅店):0.358

(高等院校, 连锁酒店):0.354

(高等院校, 医院):0.328

重庆市

(高等院校, 校园生活):0.306

(高等院校, ATM):0.238

(高等院校, 科研机构):0.224

(高等院校, 图书馆):0.224

(高等院校, 超市):0.217

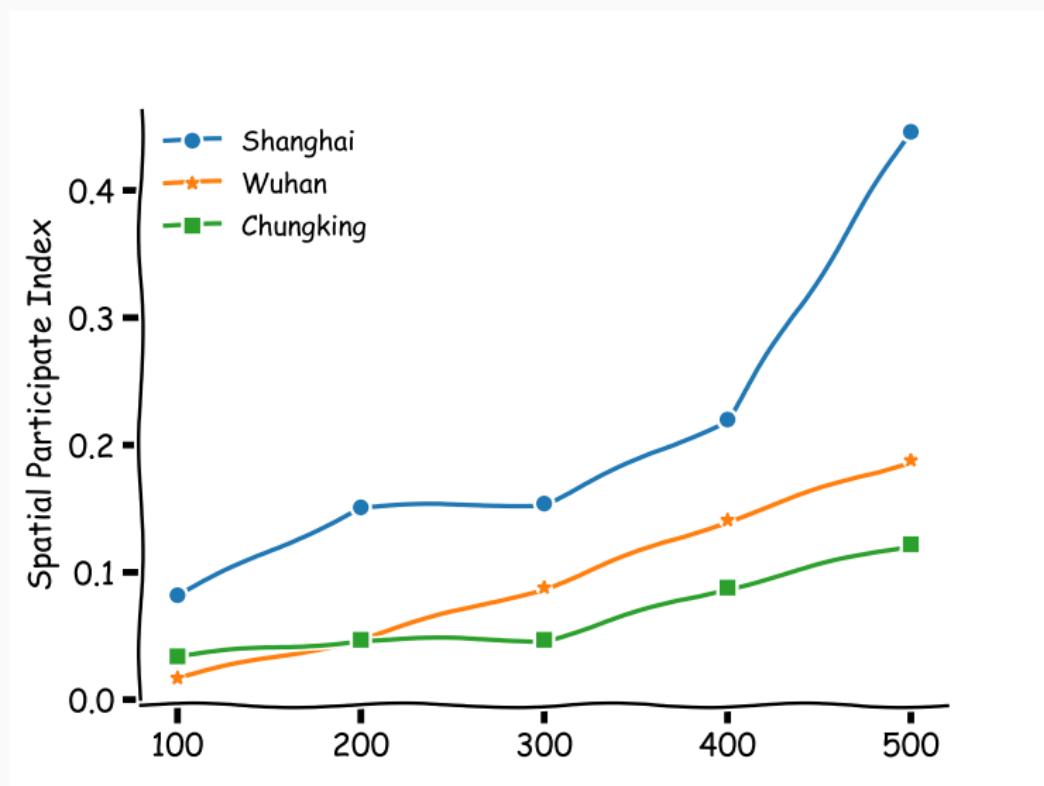
(高等院校, 特色餐厅):0.215

(高等院校, 电子卖场):0.210

(高等院校, KTV):0.205

# POI 空间分析 (同位模式)

(高等院校, 培训机构)



## POI 空间分析 (同位模式)

北京市 POI 空间模式,  $d = 500m$ , 空间参与度阈值为 0.6

# POI 空间分析 (同位模式)

北京市 POI 空间模式,  $d = 500m$ , 空间参与度阈值为 0.6

- 二阶 (29)  
(中餐厅, 校园生活), (校园生活, 医院), (甜品店, 中餐厅), (咖啡厅, 校园生活), (咖啡厅, 酒吧), (清真餐馆, 酒吧) ...

# POI 空间分析 (同位模式)

北京市 POI 空间模式,  $d = 500m$ , 空间参与度阈值为 0.6

- **二阶 (29)**

(中餐厅, 校园生活), (校园生活, 医院), (甜品店, 中餐厅), (咖啡厅, 校园生活), (咖啡厅, 酒吧), (清真餐馆, 酒吧) ...

- **三阶 (29)**

(中餐厅, 校园生活, 医院), (电影院, KTV, 美容美发店), (甜品店, 中餐厅, 美容美发店), (美容美发店, 酒吧, 甜品店), (中餐厅, 校园生活, 咖啡厅) ...

# POI 空间分析 (同位模式)

北京市 POI 空间模式,  $d = 500m$ , 空间参与度阈值为 0.6

- **二阶 (29)**

(中餐厅, 校园生活), (校园生活, 医院), (甜品店, 中餐厅), (咖啡厅, 校园生活), (咖啡厅, 酒吧), (清真餐馆, 酒吧) ...

- **三阶 (29)**

(中餐厅, 校园生活, 医院), (电影院, KTV, 美容美发店), (甜品店, 中餐厅, 美容美发店), (美容美发店, 酒吧, 甜品店), (中餐厅, 校园生活, 咖啡厅) ...

- **四阶 (19)**

(电影院, KTV, 咖啡厅, 美容美发店), (KTV, 中餐厅, 甜品店, 美容美发店), (甜品店, 中餐厅, 咖啡厅, 美容美发店), (甜品店, 咖啡厅, 美容美发店, 酒吧) ...

# POI 空间分析 (同位模式)

北京市 POI 空间模式,  $d = 500m$ , 空间参与度阈值为 0.6

- **二阶 (29)**

(中餐厅, 校园生活), (校园生活, 医院), (甜品店, 中餐厅), (咖啡厅, 校园生活), (咖啡厅, 酒吧), (清真餐馆, 酒吧) ...

- **三阶 (29)**

(中餐厅, 校园生活, 医院), (电影院, KTV, 美容美发店), (甜品店, 中餐厅, 美容美发店), (美容美发店, 酒吧, 甜品店), (中餐厅, 校园生活, 咖啡厅) ...

- **四阶 (19)**

(电影院, KTV, 咖啡厅, 美容美发店), (KTV, 中餐厅, 甜品店, 美容美发店), (甜品店, 中餐厅, 咖啡厅, 美容美发店), (甜品店, 咖啡厅, 美容美发店, 酒吧) ...

- **五阶 (7)**

电影院, KTV, 咖啡厅, 甜品店, 美容美发店), (KTV, 中餐厅, 咖啡厅, 美容美发店, 酒吧), (甜品店, 中餐厅, 咖啡厅, 美容美发店, 酒吧) ...

# POI 空间分析 (同位模式)

北京市 POI 空间模式,  $d = 500m$ , 空间参与度阈值为 0.6

- **二阶 (29)**

(中餐厅, 校园生活), (校园生活, 医院), (甜品店, 中餐厅), (咖啡厅, 校园生活), (咖啡厅, 酒吧), (清真餐馆, 酒吧) ...

- **三阶 (29)**

(中餐厅, 校园生活, 医院), (电影院, KTV, 美容美发店), (甜品店, 中餐厅, 美容美发店), (美容美发店, 酒吧, 甜品店), (中餐厅, 校园生活, 咖啡厅) ...

- **四阶 (19)**

(电影院, KTV, 咖啡厅, 美容美发店), (KTV, 中餐厅, 甜品店, 美容美发店), (甜品店, 中餐厅, 咖啡厅, 美容美发店), (甜品店, 咖啡厅, 美容美发店, 酒吧) ...

- **五阶 (7)**

电影院, KTV, 咖啡厅, 甜品店, 美容美发店), (KTV, 中餐厅, 咖啡厅, 美容美发店, 酒吧), (甜品店, 中餐厅, 咖啡厅, 美容美发店, 酒吧) ...

- **六阶 (1)**

(KTV, 中餐厅, 咖啡厅, 甜品店, 美容美发店, 酒吧)

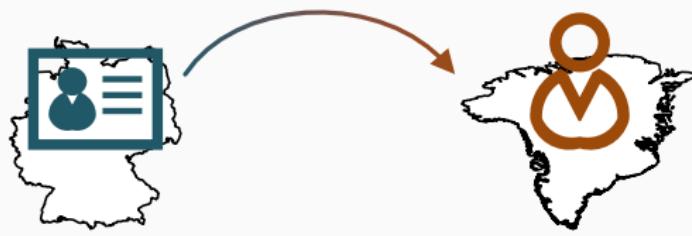
# 人口流动网络分析

---

# 人口流动网络分析 (获取)

人口流动

place/nearby/users

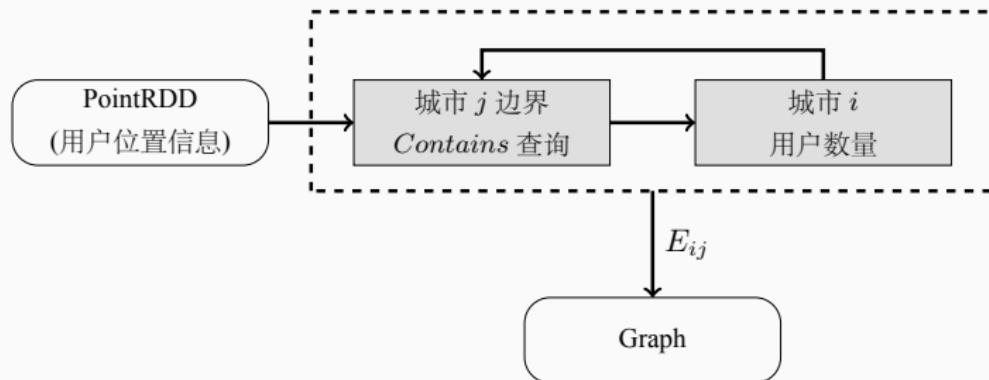


用户  
账户位置

用户使用  
微博位置

# 人口流动网络分析 (获取)

## 城市之间 GraphX 构建



# 人口流动网络分析 (统计)

## 人口流动指数

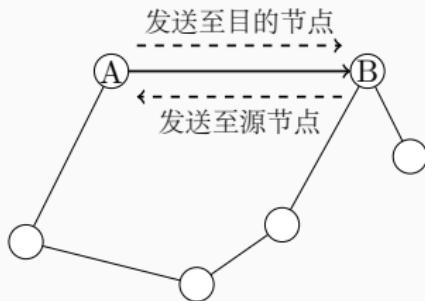
- 流入量:  $\delta_i$
- 流出量:  $\omega_i$
- 流入流出比:  
$$\psi_i = \delta_i / \omega_i$$

# 人口流动网络分析 (统计)

## 人口流动指数

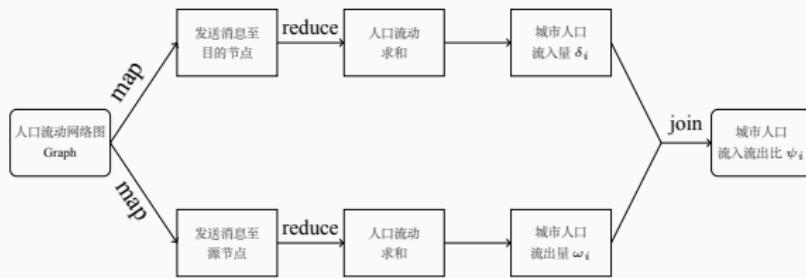
- 流入量:  $\delta_i$
- 流出量:  $\omega_i$
- 流入流出比:  
 $\psi_i = \delta_i / \omega_i$

Graph 并行化设计, aggregateMessages



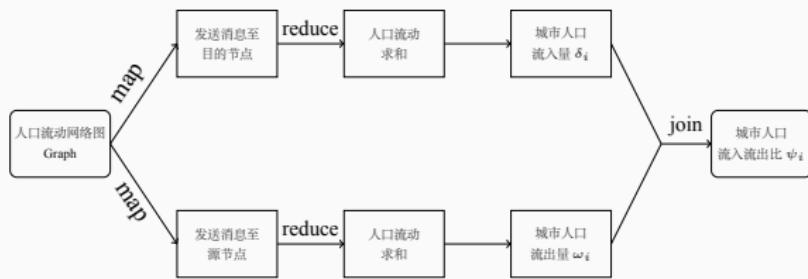
# 人口流动网络分析 (统计)

## aggregateMessages 流程

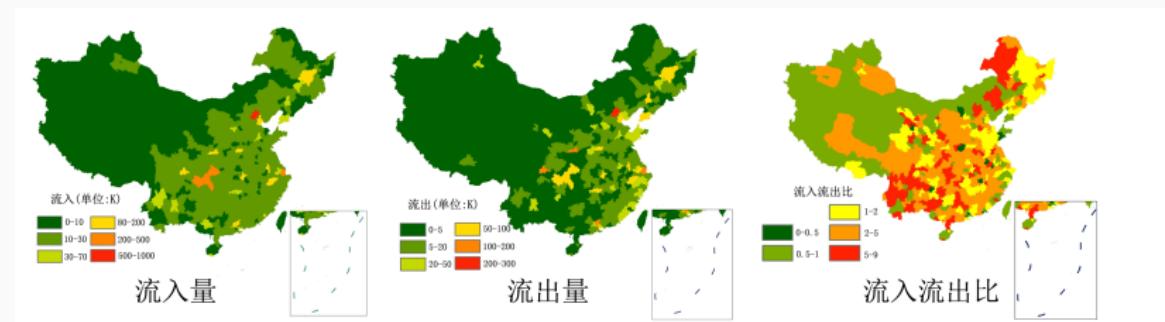


# 人口流动网络分析 (统计)

## aggregateMessages 流程



## 全国城市人口流动情况



# 人口流动网络分析 (统计)

人口流入流出比

# 人口流动网络分析 (统计)

## 人口流入流出比

- 大于 1

辽源、鄂州、西双版纳、毕节、永州、七台河、娄底、衡阳、  
白色、黄冈、松原、三亚、娄底、大理等

## 人口流入流出比

- 大于 1

辽源、鄂州、西双版纳、毕节、永州、七台河、娄底、资阳、  
白色、黄冈、松原、三亚、娄底、大理等

- 接近 1

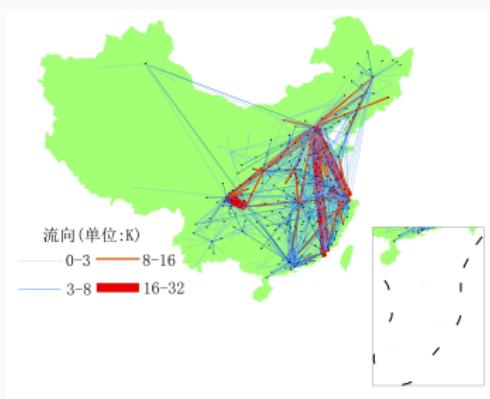
上海、北京、成都、武汉、哈尔滨、嘉兴、杭州、长沙，广  
州、西安、太原、郑州、沈阳、乌鲁木齐等

## 人口流入流出比

- 大于 1  
辽源、鄂州、西双版纳、毕节、永州、七台河、娄底、资阳、  
白色、黄冈、松原、三亚、娄底、大理等
- 接近 1  
上海、北京、成都、武汉、哈尔滨、嘉兴、杭州、长沙，广  
州、西安、太原、郑州、沈阳、乌鲁木齐等
- 小于 1  
东莞、马鞍山、芜湖、青岛、苏州、深圳、包头、宁波、无  
锡、唐山，石家庄、呼和浩特等

# 人口流动网络分析 (流向分析)

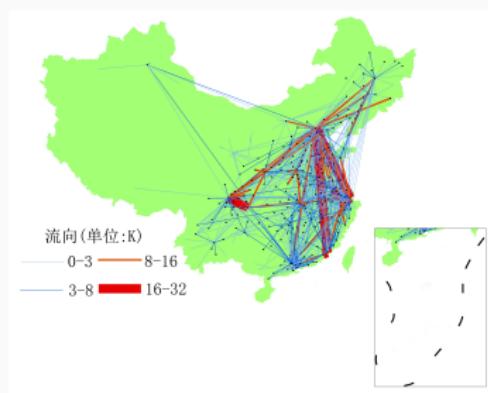
## 流向示意图



# 人口流动网络分析 (流向分析)

PageRank 计算网络节点的权重

## 流向示意图



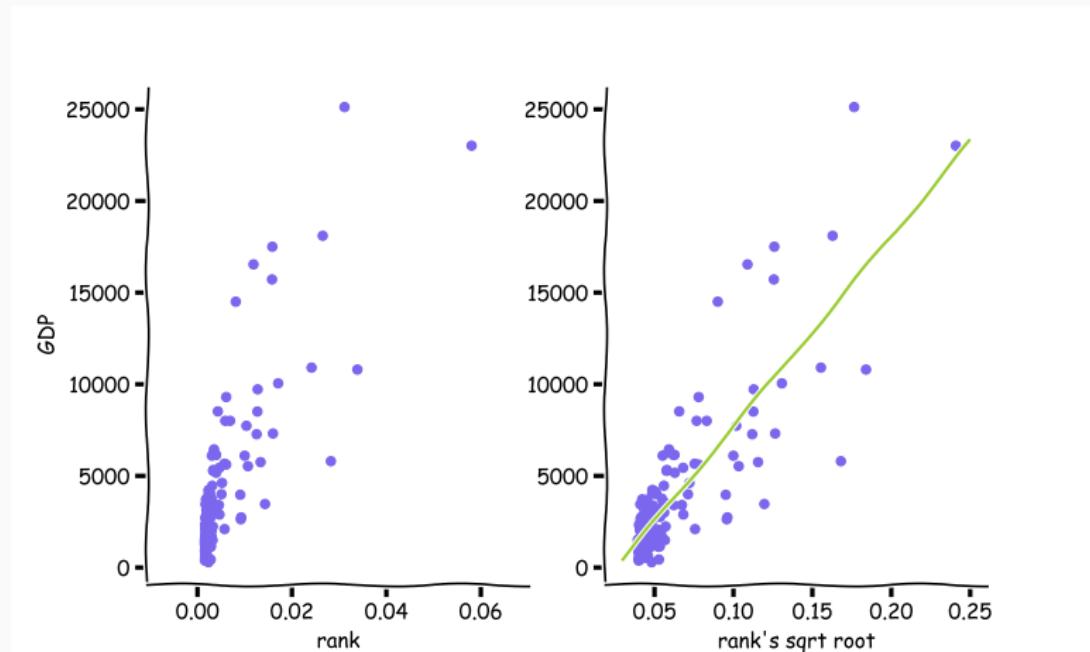
1. 北京 0.058
  2. 成都 0.033
  3. 上海 0.031
  4. 西安 0.028
  5. 广州 0.026
  6. 武汉 0.024
  7. 杭州 0.017
  8. 郑州 0.016
  9. 深圳 0.015
  10. 重庆 0.014
- ...

# 人口流动网络分析 (流向分析)

城市权重与城市 GDP 相关系数为 0.8

# 人口流动网络分析 (流向分析)

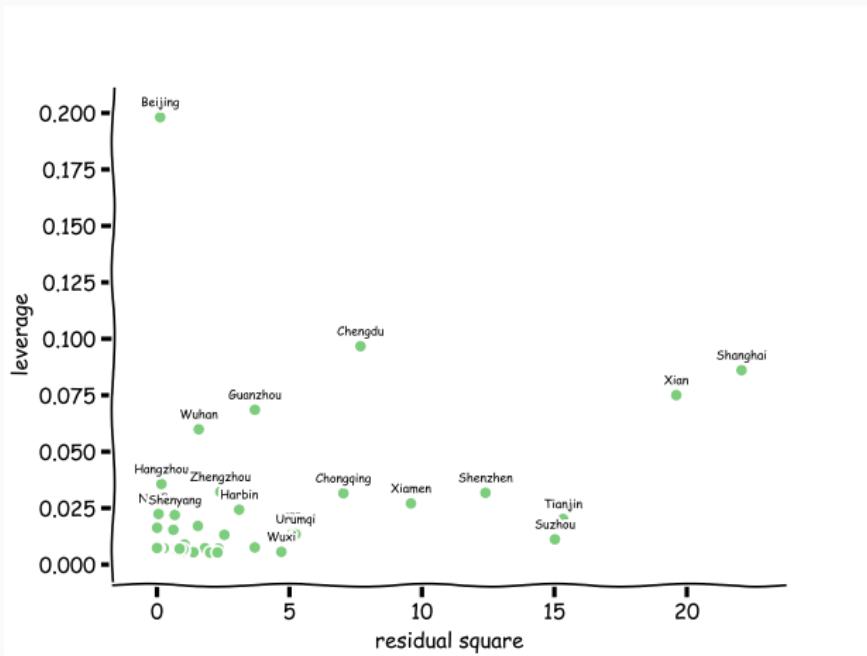
城市权重与城市 GDP 相关系数为 0.8



$$GDP = 1.037 \times 10^5 rank^{0.5} - 2669$$

# 人口流动网络分析 (流向分析)

## 残差平方与杠杆值散点图



# 人口流动网络分析 (流向分析)

## 结论

## 结论

### 1. 城市权重与城市经济发展相对一致性

城市的权重与城市 GDP 呈现一定的正相关关系，即城市的权重越高，城市的 GDP 数值会高，但也有异常

## 结论

### 1. 城市权重与城市经济发展相对一致性

城市的权重与城市 GDP 呈现一定的正相关关系，即城市的权重越高，城市的 GDP 数值会高，但也有异常

### 2. 城市权重在流动网络层级分布

北京为第一个层级；成都、上海、西安、广州和武汉为第二层级；杭州、郑州、深圳、重庆和厦门为第三个层级；哈尔滨、南京、长沙、沈阳和天津为第三层级；

## 结论

### 1. 城市权重与城市经济发展相对一致性

城市的权重与城市 GDP 呈现一定的正相关关系，即城市的权重越高，城市的 GDP 数值会高，但也有异常

### 2. 城市权重在流动网络层级分布

北京为第一个层级；成都、上海、西安、广州和武汉为第二层级；杭州、郑州、深圳、重庆和厦门为第三个层级；哈尔滨、南京、长沙、沈阳和天津为第三层级；

### 3. 东部城市与中西部城市权重的差异明显

东部城市群在人口流动网络中扮演重要的角色

# 人口流动网络分析 (社群挖掘)

## 模块值

$$Q = \frac{1}{2m} \sum_{vw} (W_{vw} - P_{vw}\delta(C_v, C_w)) \quad (5)$$

其中  $W_{vw}$  为网络中节点  $v$  和节点  $w$  之间的权重;  $m$  为权重之和;  $P_{vw}$  为零模型中节点之间的期望权重;  $C_v$  表示节点  $v$  所属的社群的类别, 如果节点  $v$  和节点  $w$  所属同一个社群, 则  $\delta(C_v, C_w)$  为 1, 否则为 0

## 模块值更新

$$\Delta Q_{AB} = Q_{AB} - Q_A - Q_B = \frac{1}{2m} \sum_{a_i \in A, b_j \in B} \left\{ \left( A_{ij} - \frac{k_i k_j}{2m} \right) \times \delta[r(i), r(j)] \right\} \quad (6)$$

# 人口流动网络分析 (社群挖掘)

改进

$$Q = \sum_c \left\{ \frac{\sum_{in}}{2m} - \left( \frac{\sum_{tot}}{2m} \right)^2 \right\} \quad (7)$$

其中  $\sum_{in}$  是社群  $c$  内部相互连接权重和,  $\sum_{tot}$  该社群  $c$  外部连接权重和

# 人口流动网络分析 (社群挖掘)

改进

$$Q = \sum_c \left\{ \frac{\sum_{in}}{2m} - \left( \frac{\sum_{tot}}{2m} \right)^2 \right\} \quad (7)$$

其中  $\sum_{in}$  是社群  $c$  内部相互连接权重和,  $\sum_{tot}$  该社群  $c$  外部连接权重和

$$\Delta Q = \left[ \frac{\sum_{in} + k_{i,in}}{2m} - \left( \frac{\sum_{tot} + k_i}{2m} \right)^2 \right] - \left[ \frac{\sum_{in}}{2m} - \left( \frac{\sum_{tot}}{2m} \right)^2 - \left( \frac{k_i}{2m} \right)^2 \right] \quad (8)$$

# 人口流动网络分析 (社群挖掘)

## 算法 社群挖掘算法

输入:

  网络图:graph;  
  最多迭代次数: maxIterations;  
  模块值变化阈值: modularityThreshold;

输出:

  模块值最大社群图: graph;

1: 初始化  $C_i, \{i = 1, 2, \dots, N\}$

2: iterate=0

3: while changeRate<modularityThreshold || iteration < maxIterations do

4:   changeRate=0

5:   for all  $i$  in graph.vertices do

6:      $Q_i = [ ]$

7:     for all  $j$  in  $i$  neighbors do

8:        $Q_i += \Delta Q(i, j)$  // 计算模块值增益

9:     end for

10:     $Q_{max}=\max(Q_i)$

11:    if  $Q_{max} > 0$  then

12:      graph=updategraph() // 如果最大模块值增益大于零, 合并顶点

13:      changeRate=  $Q_{max}$

14:    end if

15:   end for

16:   iterate += 1

17: end while

18: return graph

# 人口流动网络分析 (社群挖掘)

并行化设计， aggregateMessages 接口

# 人口流动网络分析 (社群挖掘)

并行化设计， aggregateMessages 接口

## 1. map 阶段

消息类，包含了顶点、所属社群和目标社群的权重

# 人口流动网络分析 (社群挖掘)

并行化设计， aggregateMessages 接口

## 1. **map** 阶段

消息类，包含了顶点、所属社群和目标社群的权重

## 2. **Reduce** 阶段

将发送到同一个顶点的消息按照模块增益值最大进行合并

# 人口流动网络分析 (社群挖掘)

并行化设计， aggregateMessages 接口

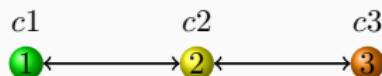
## 1. map 阶段

消息类，包含了顶点、所属社群和目标社群的权重

## 2. Reduce 阶段

将发送到同一个顶点的消息按照模块增益值最大进行合并

## 3. 连通量合并



# 人口流动网络分析 (社群挖掘)

并行化设计， aggregateMessages 接口

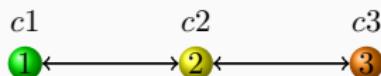
## 1. map 阶段

消息类，包含了顶点、所属社群和目标社群的权重

## 2. Reduce 阶段

将发送到同一个顶点的消息按照模块增益值最大进行合并

## 3. 连通量合并

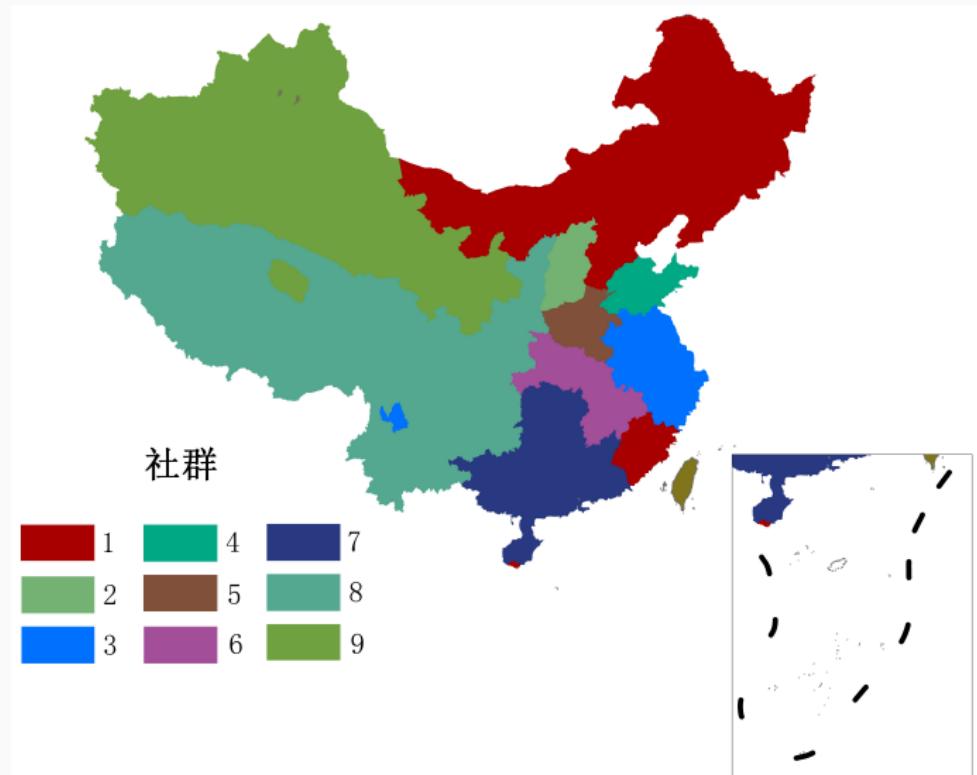


## 4. 图更新

与上一轮的 Graph 对象的顶点进行 joinVertice 操作

# 人口流动网络分析 (社群挖掘)

## 全国城市社群划分



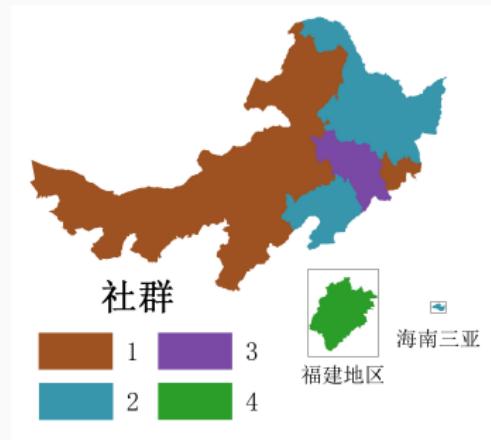
# 人口流动网络分析 (社群挖掘)

## 社群划分

1. 北京、天津、河北城市、东三省城市、福建城市、三亚等
2. 山东城市
3. 河南城市
4. 山西城市
5. 湖北城市、江西城市
6. 上海、江苏城市、安徽城市、浙江城市、丽江
7. 广东城市、广西城市、湖南城市等
8. 重庆、四川城市、贵州城市等西南城市
9. 新疆城市、甘肃城市等西北城市

# 人口流动网络分析 (社群挖掘)

## 子社群划分



## 结论

1. 城市之间人口流动以省份为划分的特征较为明显
2. 城市社群组成受地理空间位置影响较大，有明显的区域特征
3. 城市社群组成也存在不受地理位置影响的特殊情况，如福建省，三亚市与北方城市组成同一社群，丽江与东部省份城市为一个社群，从侧面表明其人口流动突破了地理空间的限制
4. 当对全国社群进行子社群进行社群挖掘，可以发现以省份为集聚现象越发明显

# 总结

---

- **Spatial-Spark**

Spatial-Spark 是高效的海量空间数据分析框架

- **POI 同位模式挖掘**

微博 POI 同位模式能够反映城市线下分布情况，尤其是商业实体分布

- **人口流动网络分析**

使用微博用户位置信息，能够快速获取构建城市人口流动网络

谢谢

欢迎各位老师批评指正