火龙果•整理
uml.org.cn

基于Graphx的大规模用户图计算

淘宝技术部——数据挖掘与计算
明风

火龙果•整理
uml.org.cn

目录

- Graphx简介和特性
- 图计算场景
- 整体模型，流程和算法
- 调优与改进
- 性能和技巧
- 总结

火龙果•整理
uml.org.cn

Graphx的发展

0.5
- 2012-10-24
- Bagel

0.8
- 2013-09-25
- Graphx-Branch

0.9
- 2014-03-03
- Graphx-Alpha

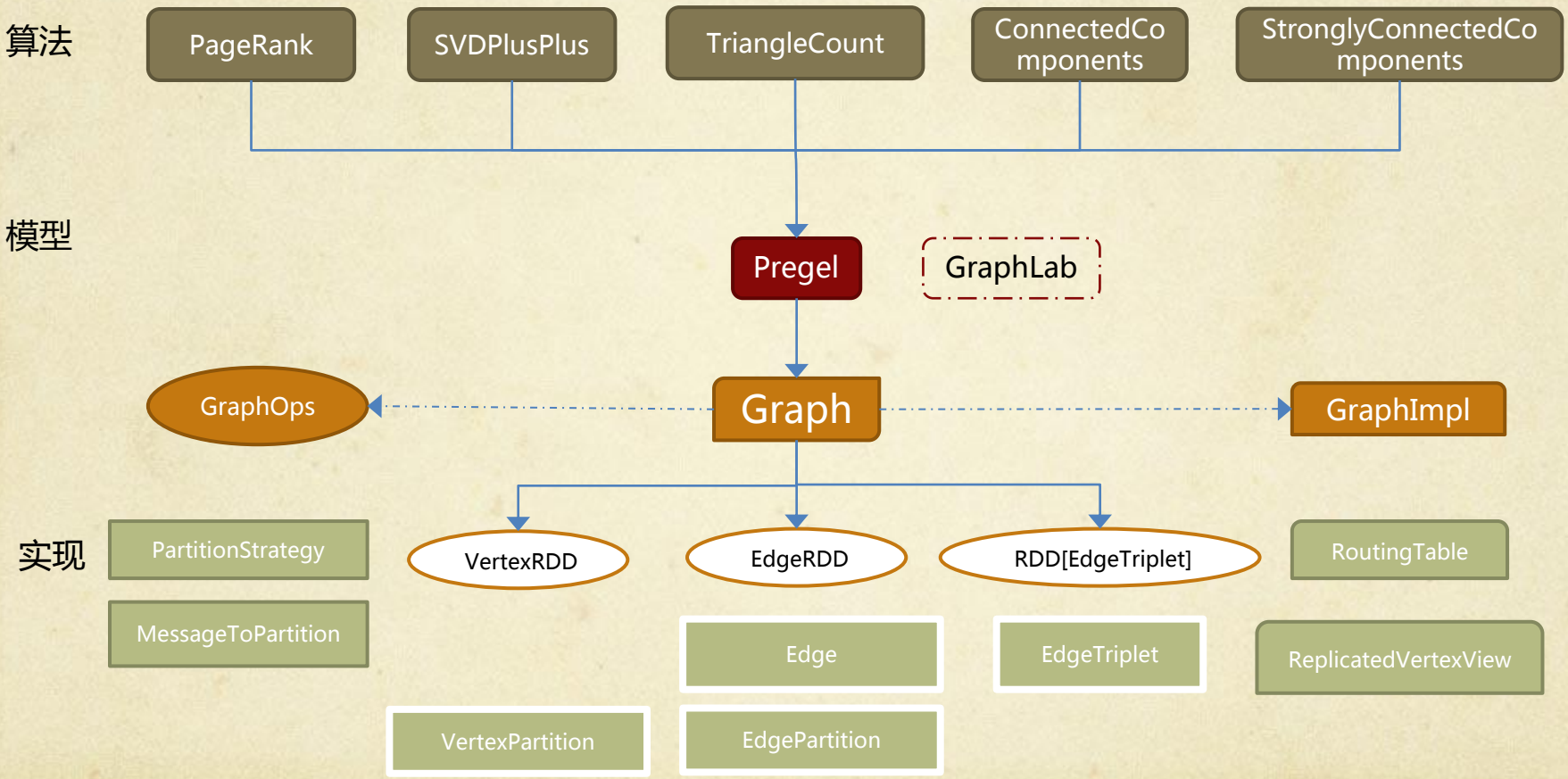
特性
- 离线
- 批量
- 同步

竞争对手
- GraphLab
- Giraph

合作伴侣
- Neo4j
- Titan

火龙果•整理
uml.org.cn

Graphx架构



火龙果•整理
uml.org.cn

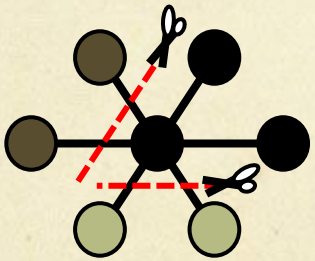
关键点

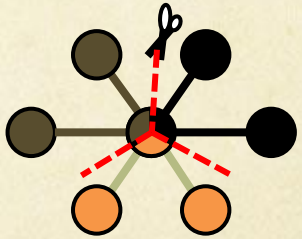
- 3个核心的RDD
  - Vertices
  - Edges
  - Triplets
- 3个特性
  - 不变性——Immutable
  - 分布性——Distributed
  - 容错性——Fault-Tolerant

火龙果•整理
uml.org.cn

边分割和点分割

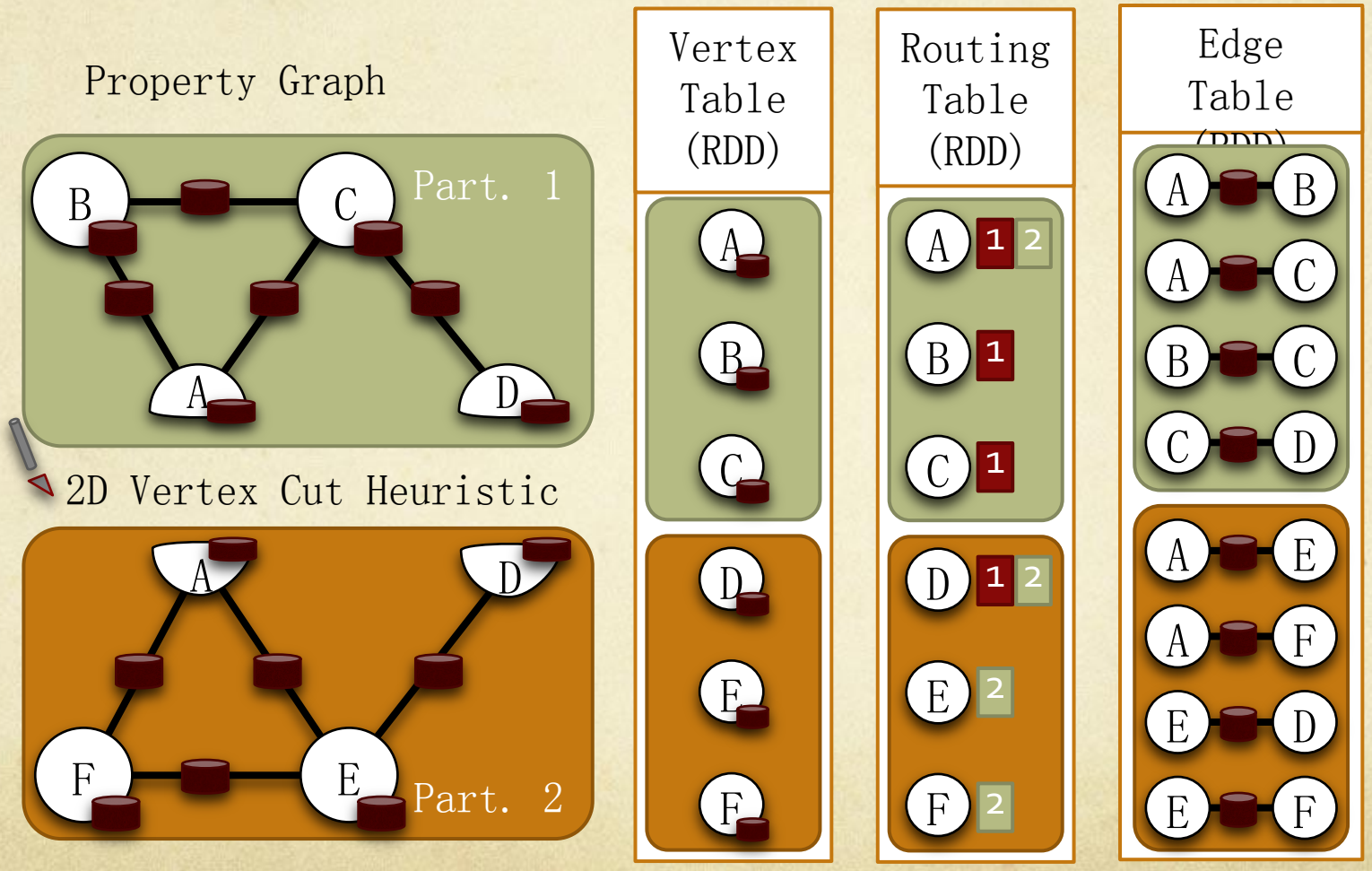
- VertexPartition & EdgePartition
- Routing Table





火龙果•整理
uml.org.cn

点分割实现



火龙果•整理
uml.org.cn

Graphx主要的接口

基本信息接口

- numEdges
- numVertices
- degrees(in/out)

转换操作

- mapVertices
- mapEdges
- mapTriplets

结构操作

- reverse
- subgraph
- mask
- groupEdges

联边聚合接口

- mapReduceTriplets
- collectNeighbors

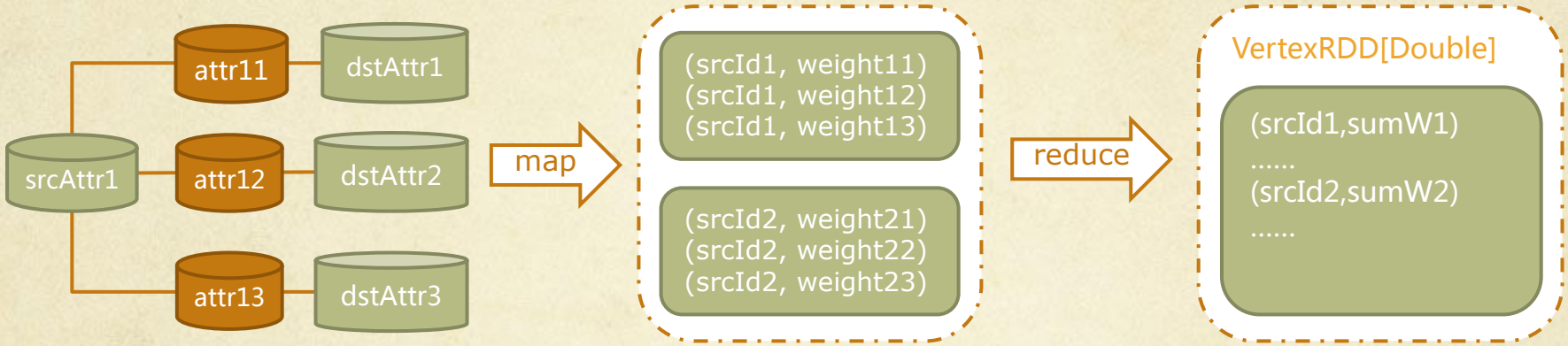
缓存操作

- cache
- unpersistVertices

火龙果•整理
uml.org.cn

重要接口

- mapReduceTriplets




```
val sumWeight: VertexRDD[Double] =
    g.mapReduceTriplets(
            triplet => {
                Array((triplet.srcAttr.id, triplet.attr.weight)).iterator
            },
            (a, b) => a + b
    )
```

MAP

REDUCE

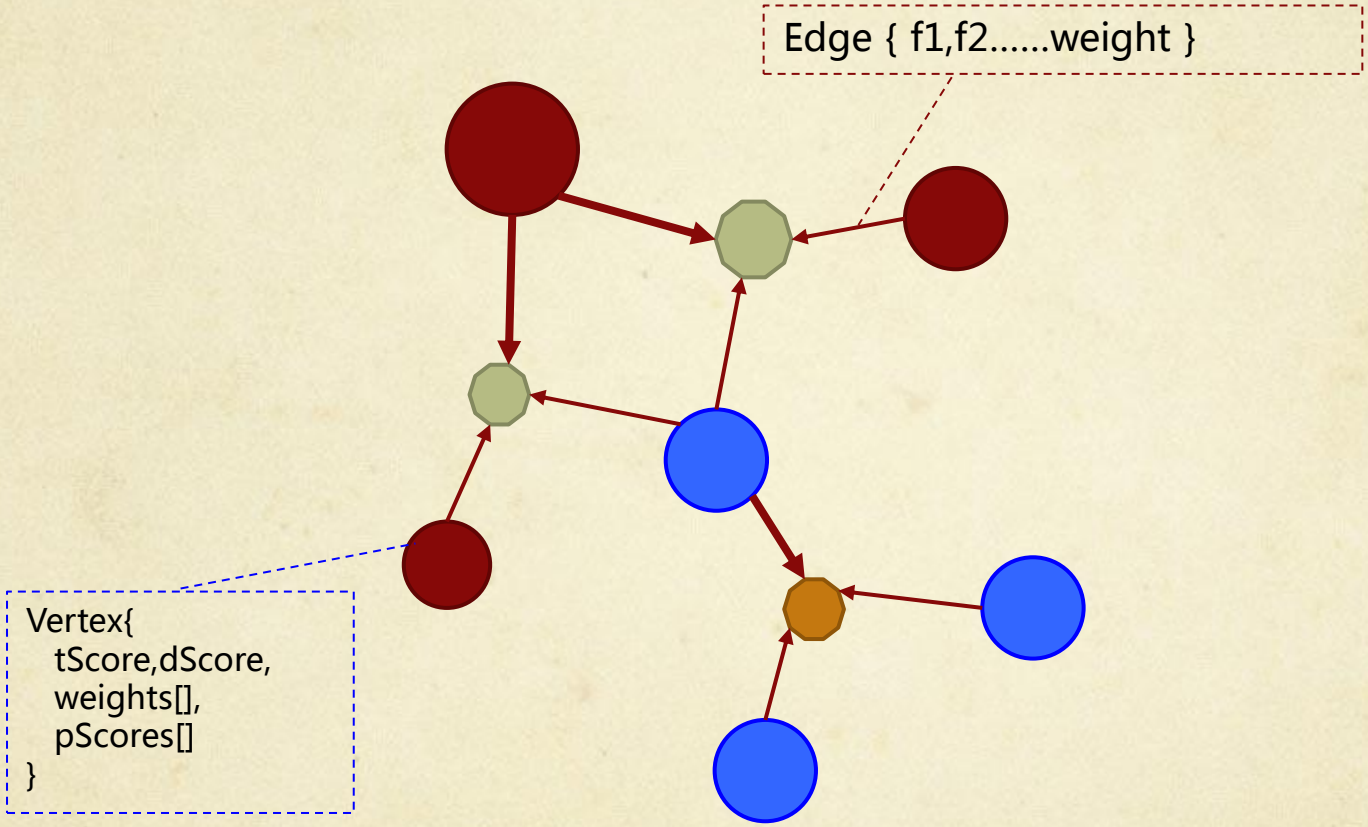
火龙果•整理
uml.org.cn

用户图计算的场景

- 基于度分布的中枢节点发现
- 基于最大连通图的社区发现
- 基于三角形计数的关系衡量
- 基于随机游走的用户属性传播
- ……

火龙果•整理
uml.org.cn

用户能量模型



Supervised Random Walks: Predicting and Recommending Links in Social Network

火龙果•整理
uml.org.cn

模型设计

1. 点能量传播

正能量，负能量都会向周围的点传播，而能量从点i到点j传播的比例是：边（j-i）在点i的所有边的权重和的占比

$$A_{ij} = \frac{(1+e^{-w(x_{ij}-q)})^{-1}}{\sum_j (1+e^{-w(x_{ij}-q)})^{-1}}$$

2. 训练集求AUC

标签集能量传播之后，在训练集上计算AUC

$$\mathrm{AUC}(w,q) = \frac{\sum_{i\in PT}\sum_{j\in NT} S(P(i)-P(j))}{|PT||NT|}$$

3. AUC求偏导

对AUC求偏导，得到每个维度的权重和偏移量

$$\frac{\partial AUC(w,q)}{\partial w} = \frac{\sum_{i\in PT}\sum_{j\in NT} \frac{\partial S(d_{ij})}{d_{ij}}\left(\frac{\partial P(i)}{\partial w} - \frac{\partial P(j)}{\partial w}\right)}{|PT||NT|}$$

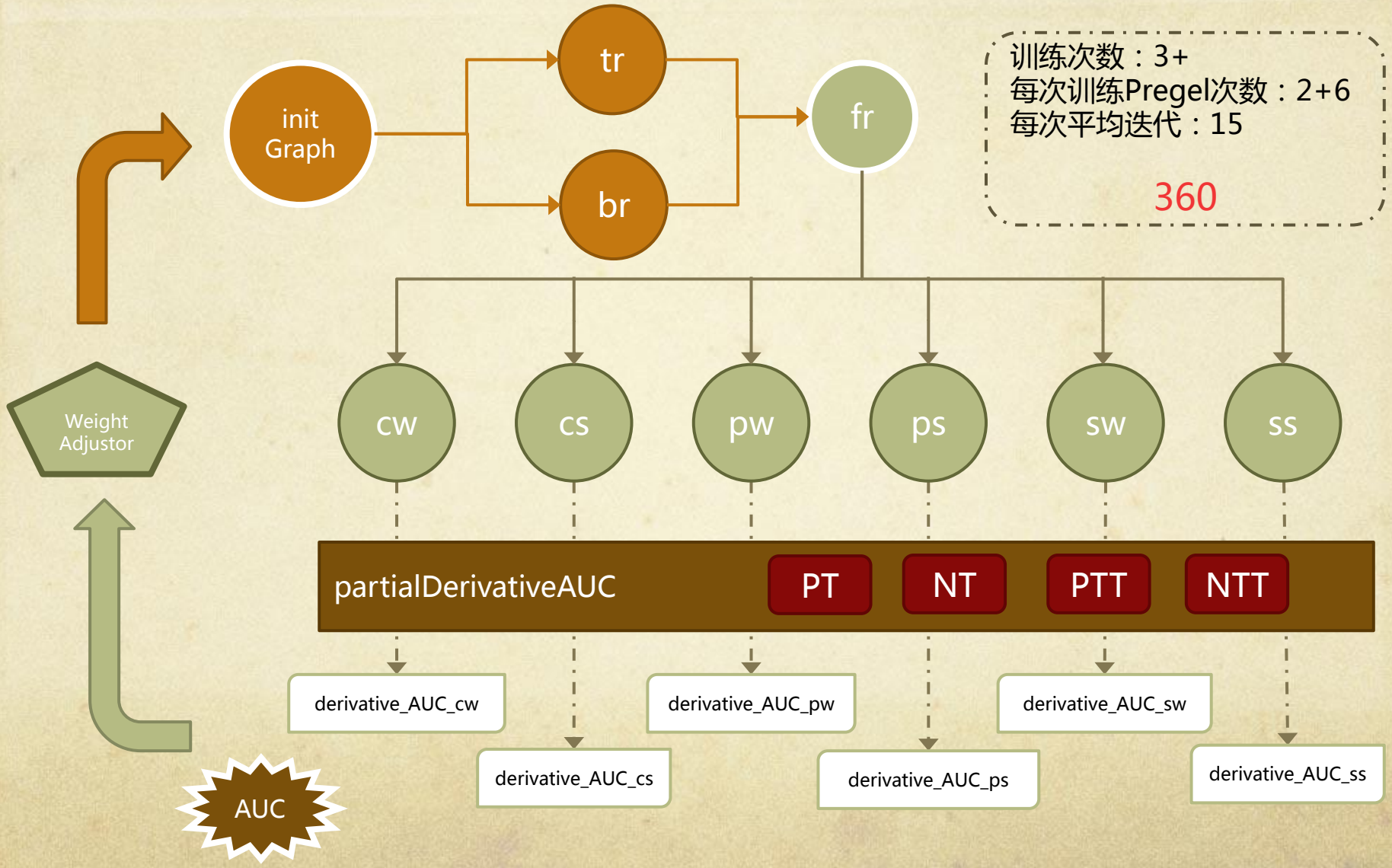
4. 更新边的权重

使用新的权重调节因子，对图的所有边的权重，再做一次更新

$$S(x;b) = \frac{1}{1+e^{-bX}}$$

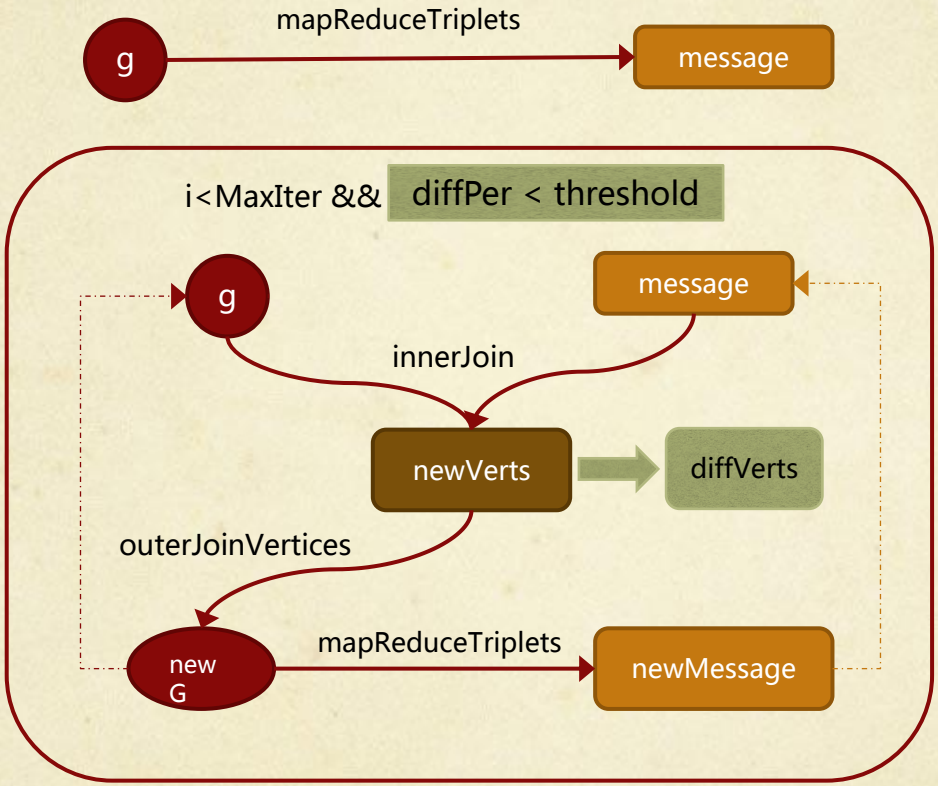
火龙果•整理
uml.org.cn

模型训练



火龙果•整理
uml.org.cn

基于Pregel的随机游走



内存释放：unpersistVertices

火龙果•整理
uml.org.cn

3个函数

- **点消息处理**

```
vprog{
  v.score = msg + (1 - ALPHA) * v.weight
}
```

- **发送消息**

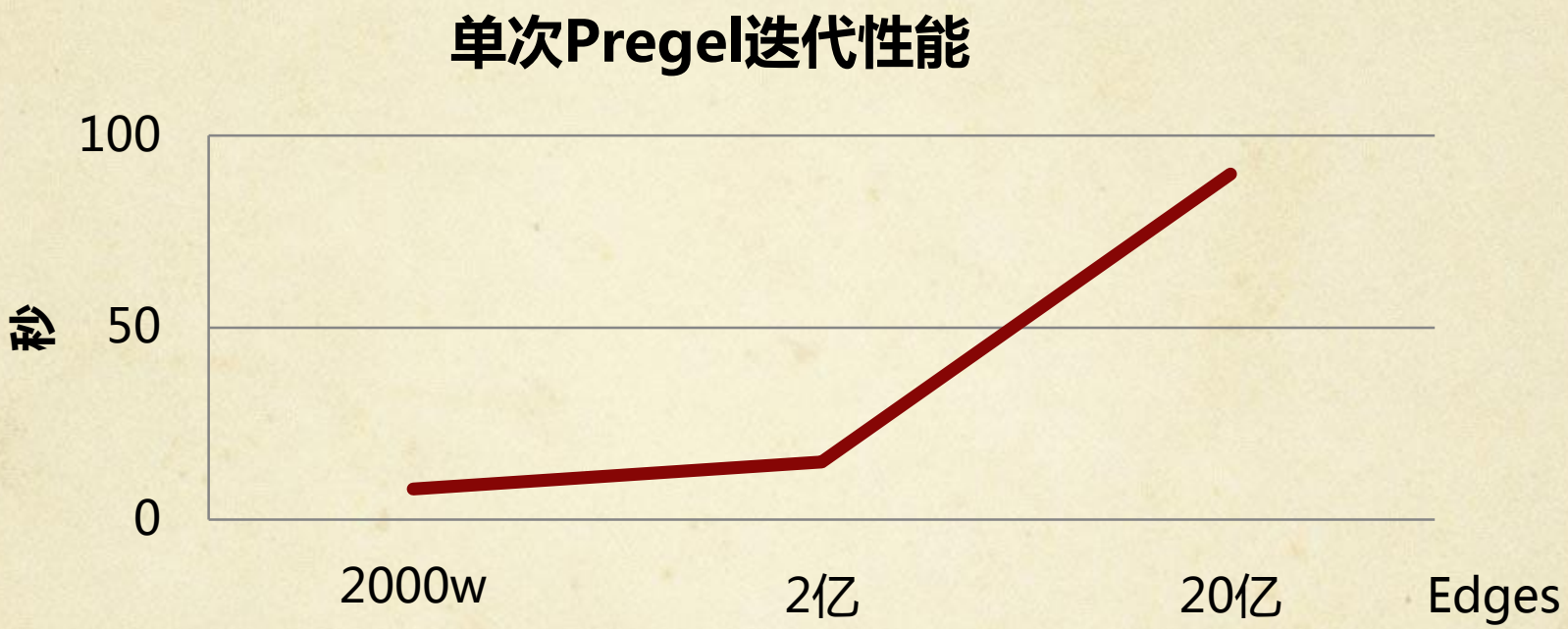
```
sendMsg{
   (destId, ALPHA * edgeTriplet.srcAttr.score *
edgeTriplet.attr.weight)
}
```

- **消息合并**

```
mergeMsg{
        v1+v2
}
```

火龙果•整理
uml.org.cn

Graphx的性能参考



运行环境

200 Worker
5 Core/Per Worker
30G/Per Worker

火龙果•整理
uml.org.cn

Graphx使用技巧

循序渐进

- 使用Sample，逐步增大数据量
- 从边生成点，再去关联点属性

```
val edgeRdd = loadEdges(edgefile).sample(false, 0.01, 1)
val verticesRdd= extractVerticesFromEdge(edgeRdd)
Graph(verticesRdd, edgeRdd)
```

火龙果•整理
uml.org.cn

Graphx使用技巧

释放内存

- 保留旧图的引用，并尽快释放

```
prevG = g
 g = g.outerJoinVertices(newVerts) {……}.cache()
 ( ……do some action to g…… )
 prevG.unpersistVertices(blocking=false)
```

火龙果•整理
uml.org.cn

Graphx使用技巧

异常定位

- 加入带时间点的count，作为调试锚点

```
g = g.outerJoinVertices(newVerts) {……}.cache()
if (debug_mode){
    logWithTime(g.vertices.count())
}
```

火龙果•整理
uml.org.cn

Graphx的配置建议

- spark.default.parallelism
- num-workers * worker-cores
- master-memory
- spark.storage.memoryFraction=0.5

火龙果•整理
uml.org.cn

加入我们

- 我们需要
  - 复杂网络算法师
  - Spark攻城师

- 联系方式
  - 微博: @明风Andy

火龙果•整理
uml.org.cn

谢谢

Q&A