大规模图数据分析系统

冯吕

Institute Of Computing Technology, Chinese Academy Of Sciences

fenglv19s@ict.ac.cn

2020 年 6 月 7 日

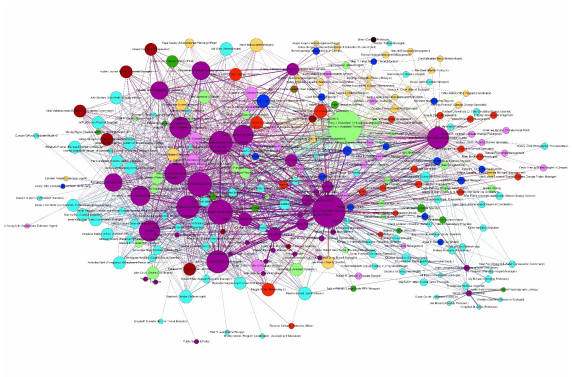
中国科学院大学

University of Chinese Academy of Sciences

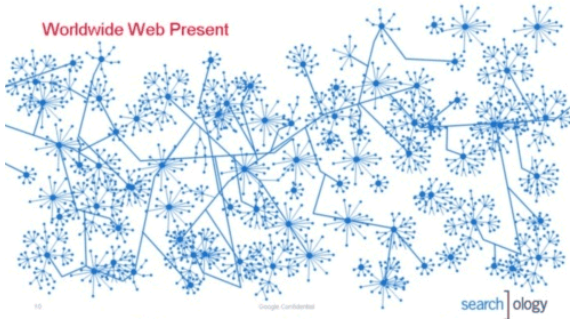
目录

背景

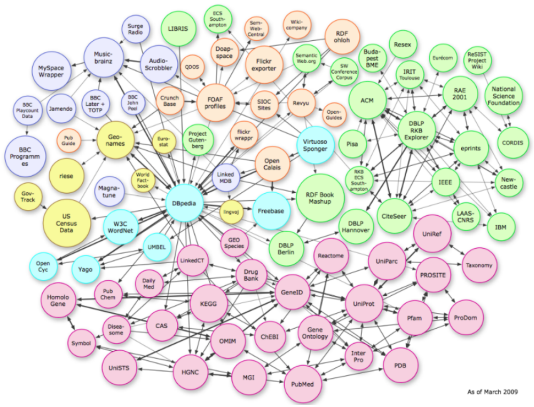
图结构数据广泛存在



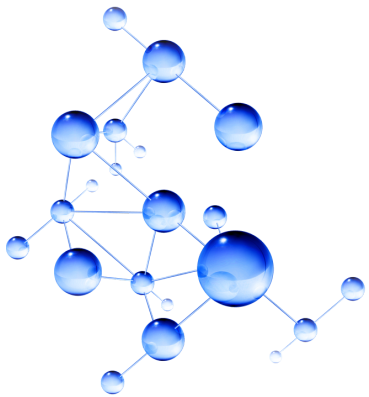
(a) 社交网络



(b) 因特网链接网络

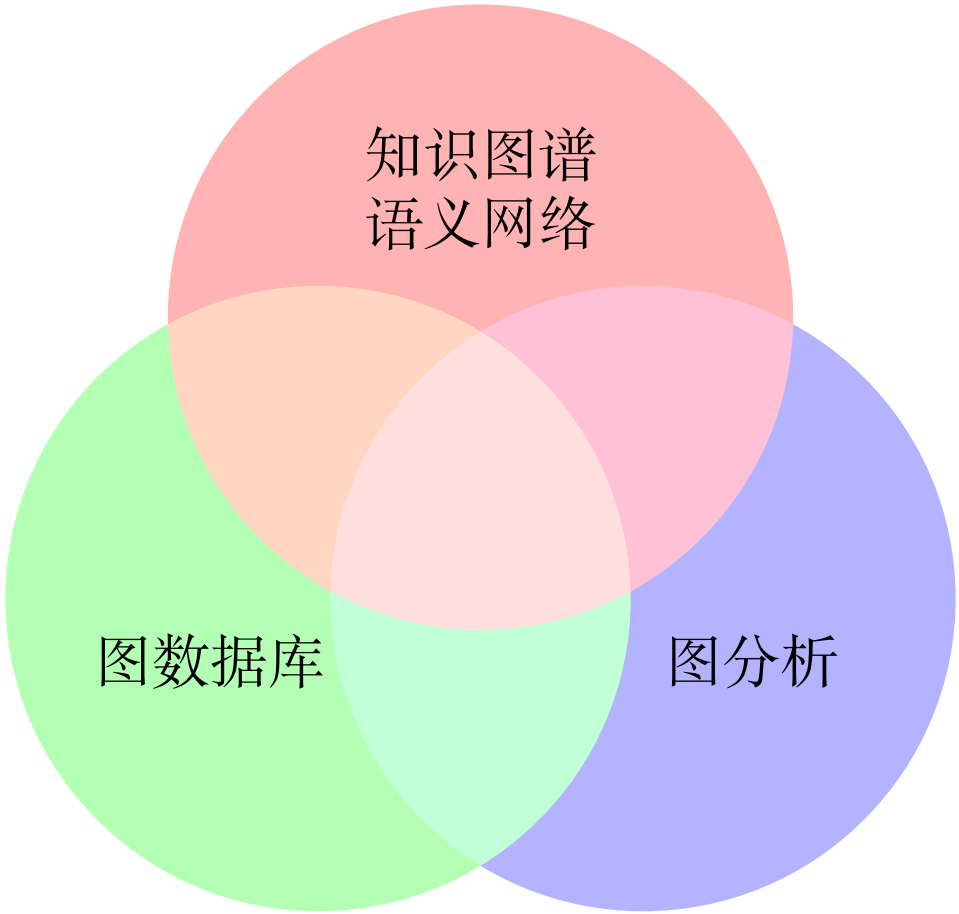


(c) 知识图谱

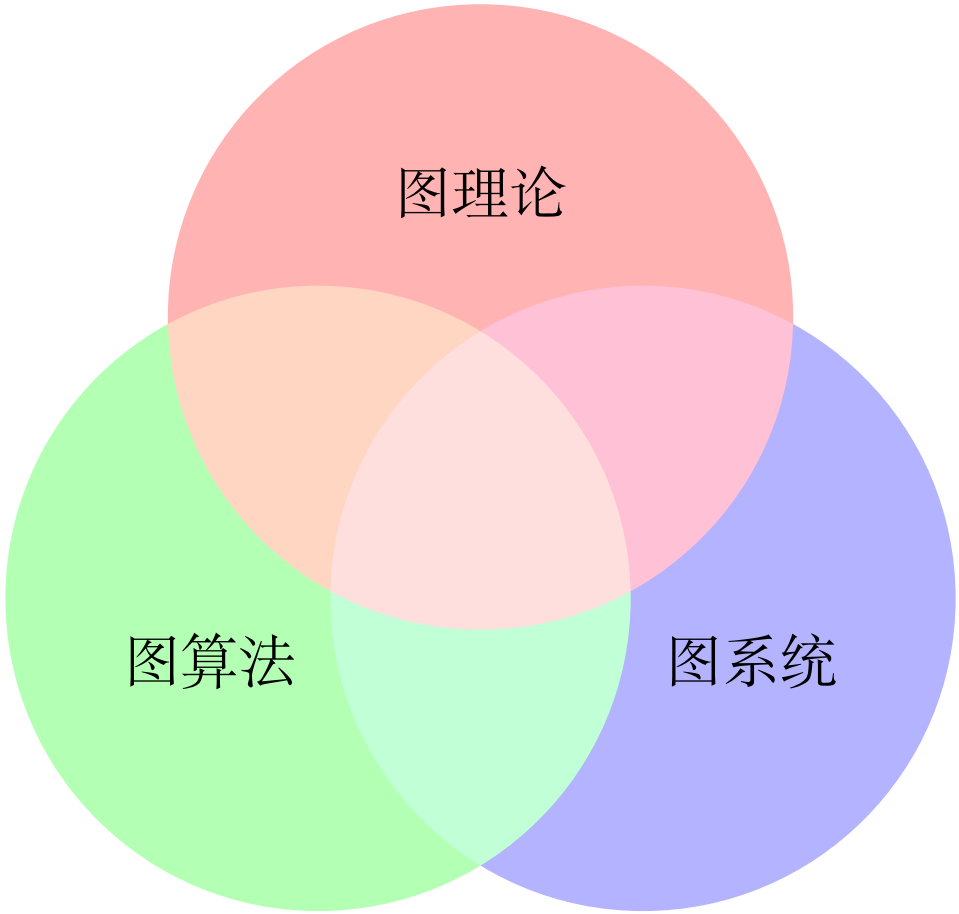


(d) 生物大分子结构网络

图相关研究较为分散

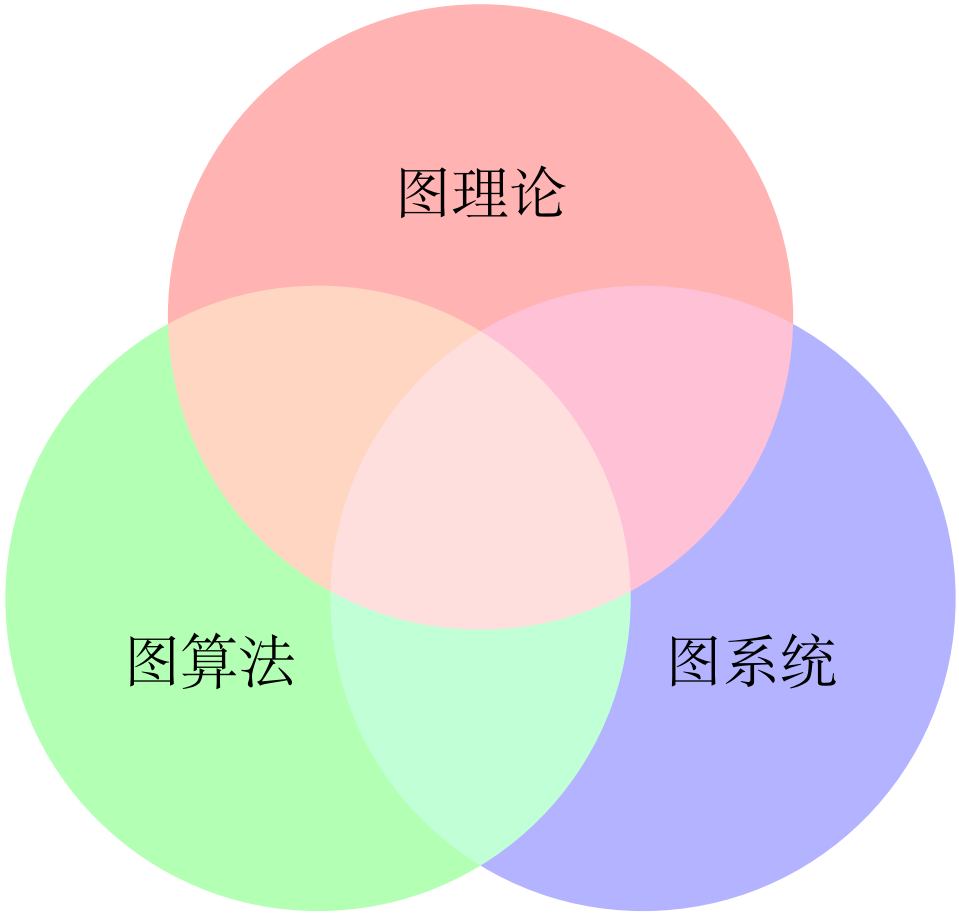


图相关研究较为分散



本次汇报主要涉及图分析系统领域

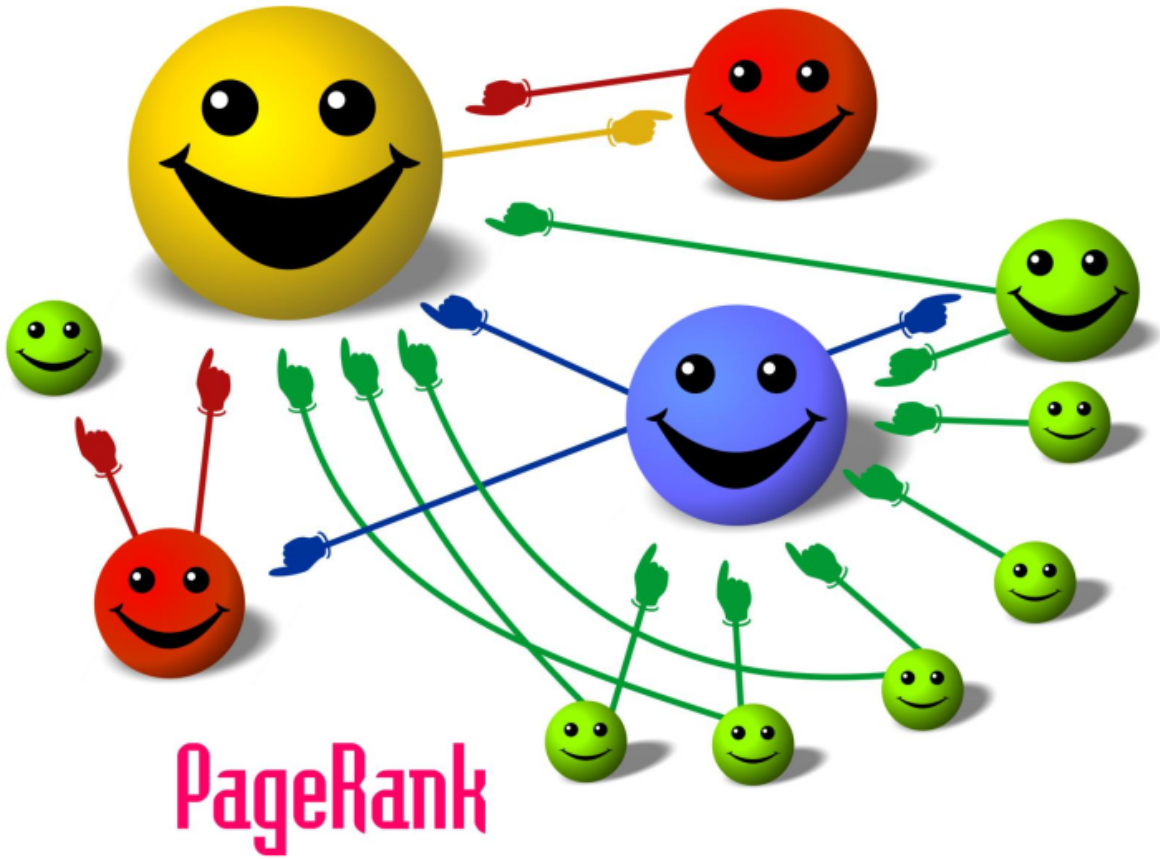
图相关研究较为分散



本次汇报主要涉及图分析系统领域

图分析任务

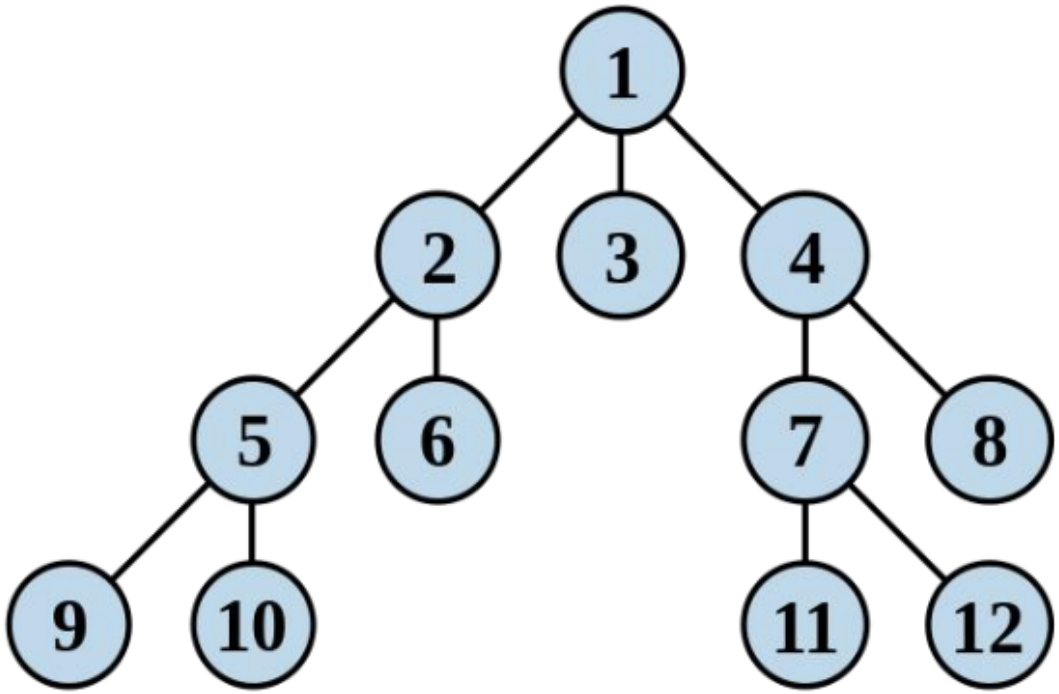
- 找出最“重要”的点



应用场景：搜索引擎等

图分析任务

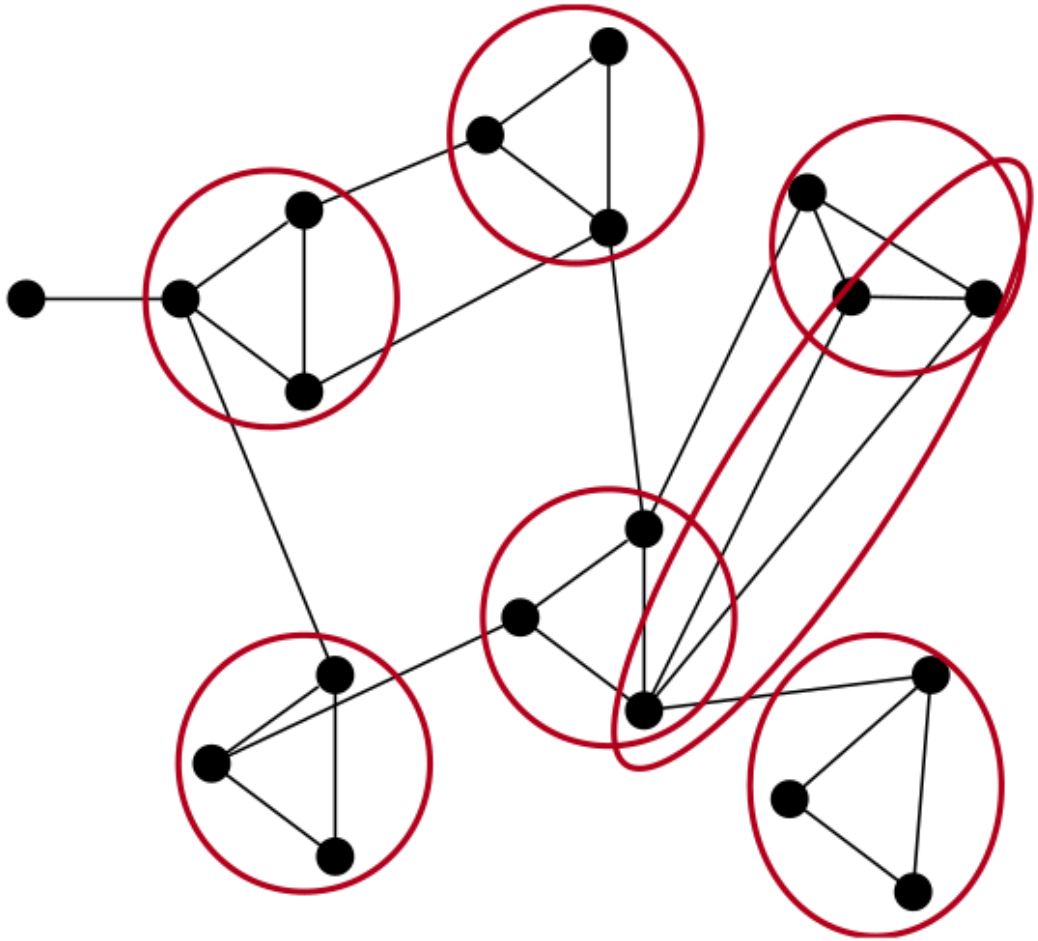
- 各类搜索、遍历算法



应用场景：查询引擎等

图分析任务

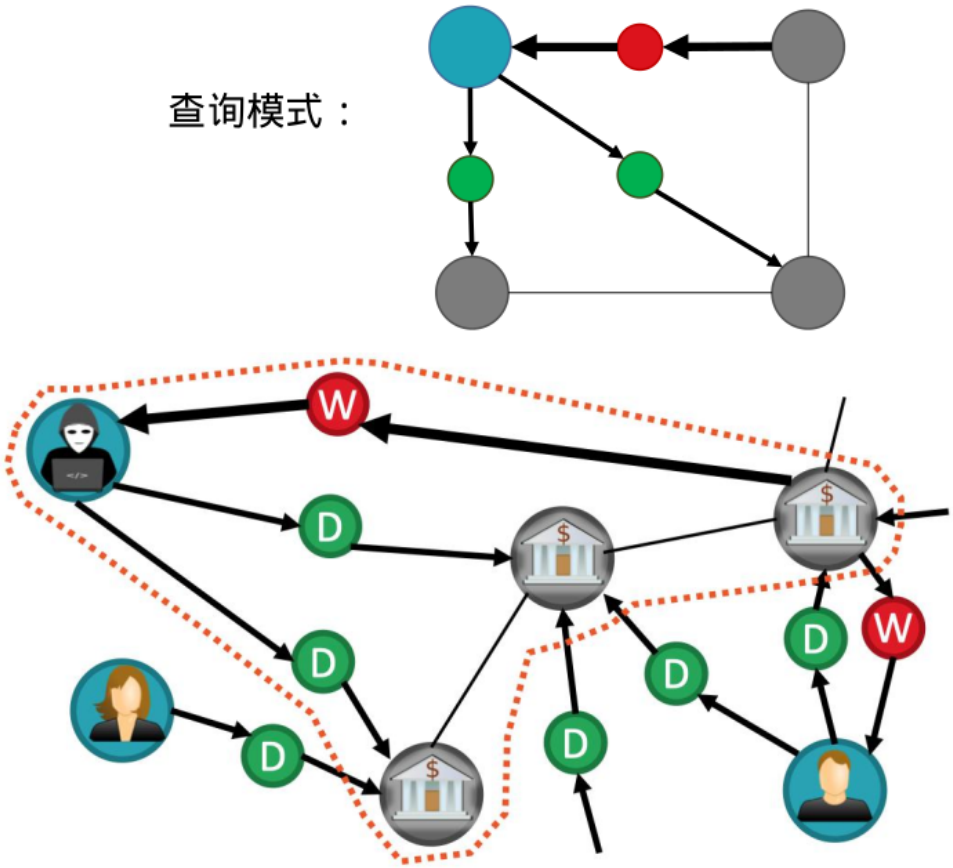
- 三角形计数



应用场景：推荐系统、异常检测等

图分析任务

- 子图匹配



应用场景：反欺诈、反洗钱等

图分析任务分类

图点边计算任务

- 遍历算法：
  - 宽度优先搜索
  - 单源最短路径
  - ···
- 计算算法：
  - 网页排序
  - 中心度计算
  - ···

图结构挖掘任务

- 子图统计：
  - 主题计数
  - 团块发现
  - 频繁子图挖掘
  - ···

图分析任务分类

图点边计算任务

- 遍历算法：
  - 宽度优先搜索
  - 单源最短路径
  - ···
- 计算算法：
  - 网页排序
  - 中心度计算
  - ···

图结构挖掘任务

- 子图统计：
  - 主题计数
  - 团块发现
  - 频繁子图挖掘
  - ···

图分析任务面临的挑战

- 现实世界中的图非常大：
  - 图数据也是一种大数据
  - 社交网络达到十亿级别点、百亿级别边；
  - 网页链接网络达到百亿级别点、千亿级别边；
- 图分析任务的特点：
  - 无结构数据 ⇒ 数据局部性差
  - 并行困难：
    - 任务划分 ⇒ 找到最优的任务划分是NP 困难的
    - 图的幂定律 ⇒ 难以实现负载均衡
    - 分布式系统的容错

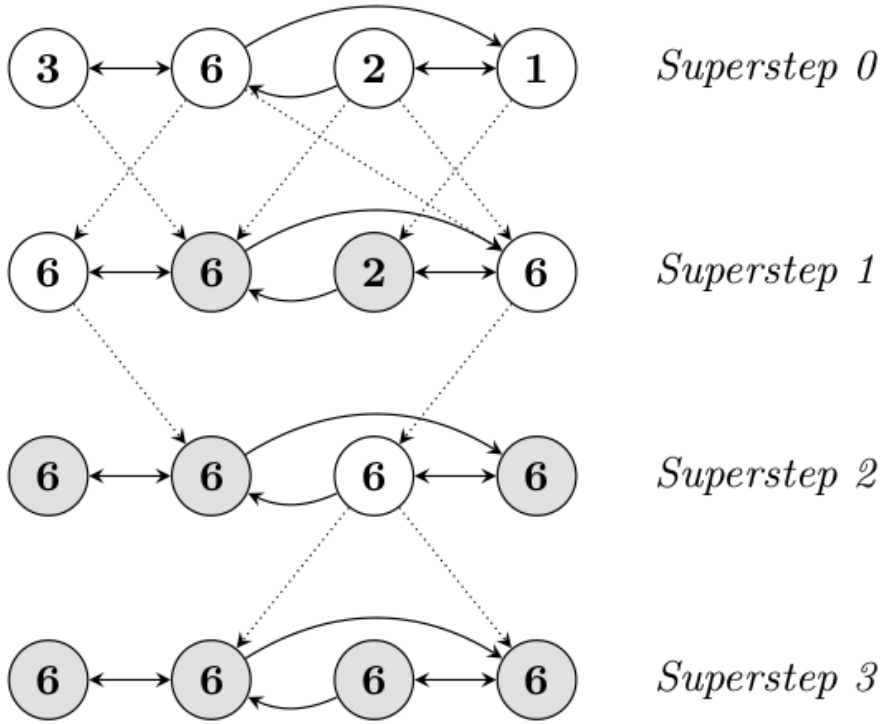
图分析任务面临的挑战

- 现实世界中的图非常大：
  - 图数据也是一种大数据
  - 社交网络达到十亿级别点、百亿级别边；
  - 网页链接网络达到百亿级别点、千亿级别边；
- 图分析任务的特点：
  - 无结构数据 $\Rightarrow$ 数据局部性差
  - 并行困难：
    - 任务划分 $\Rightarrow$ 找到最优的任务划分是NP 困难的
    - 图的幂定律 $\Rightarrow$ 难以实现负载均衡
    - 分布式系统的容错

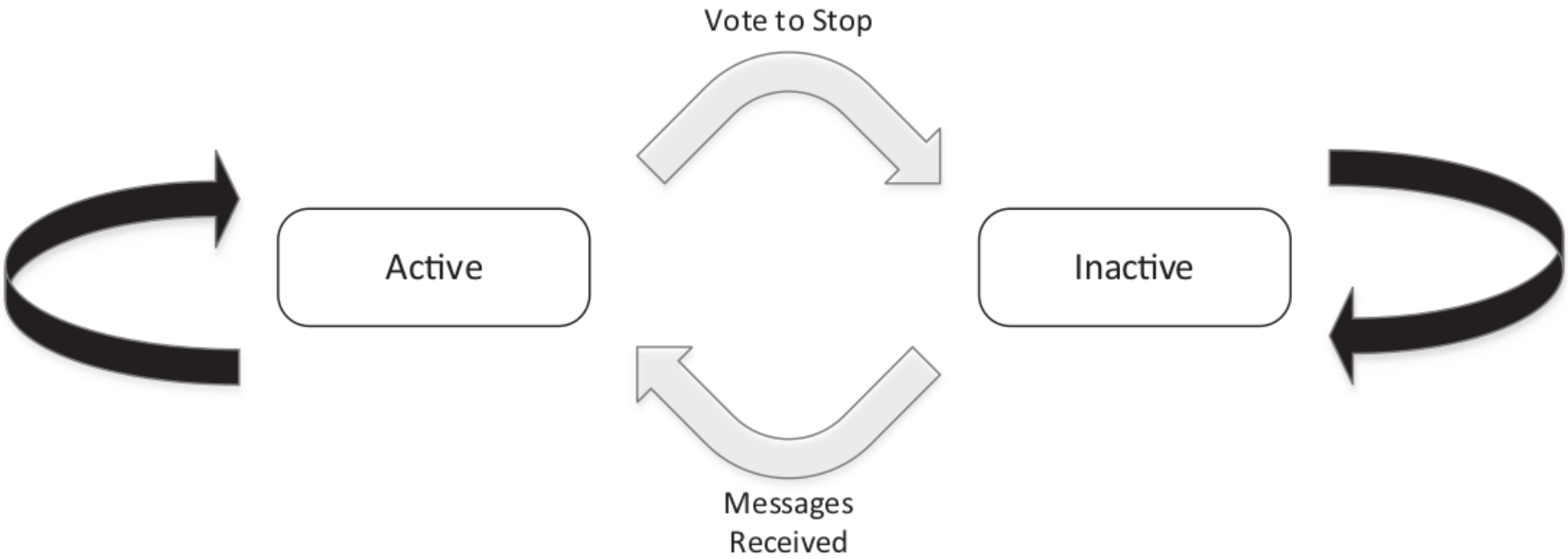
图分析系统介绍

Pregel

- 由 Google 提出的分布式图计算框架，采用Master-Worker架构
- 提出以点为中心的计算模型
- 采用BSP并行模型



最大值示例



节点状态机

- 初始时，所有节点均为活跃状态，在每轮迭代中，每个活跃节点执行用户定义的 Compute 函数
- 当所有节点均变为非活跃状态时候，算法终止

Pregel

图划分：

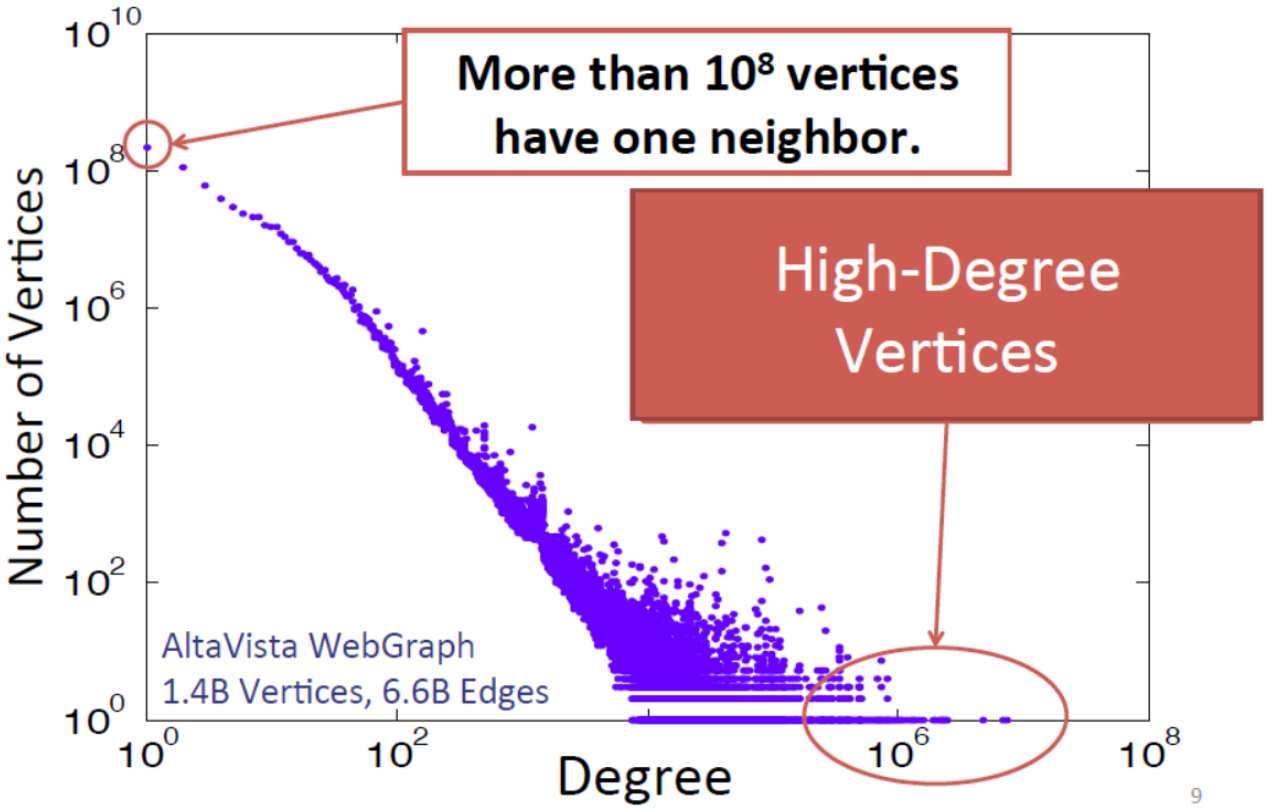
- 将图分为多个 partition，每个 partition 包含顶点集与其对应的所有出边
- 默认使用哈希函数对节点进行划分：$hash(VertexId)\%N$，N 为 partition 的数目

容错：

- 通过检查点机制实现容错

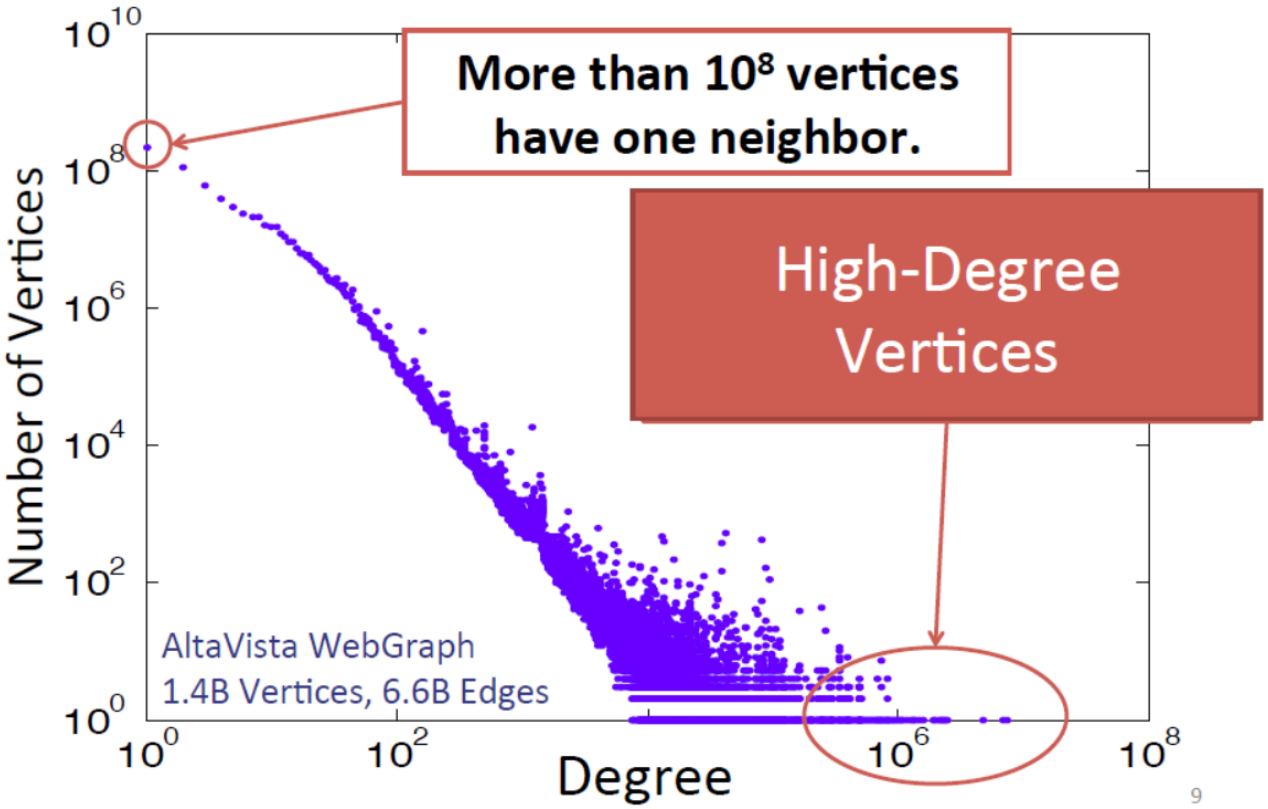
PowerGraph

大量自然图满足 Power Law



问题：按点进行图划分会导致图划分之间有大量的跨边，需要传递大量的消息，以及计算和负载不均衡

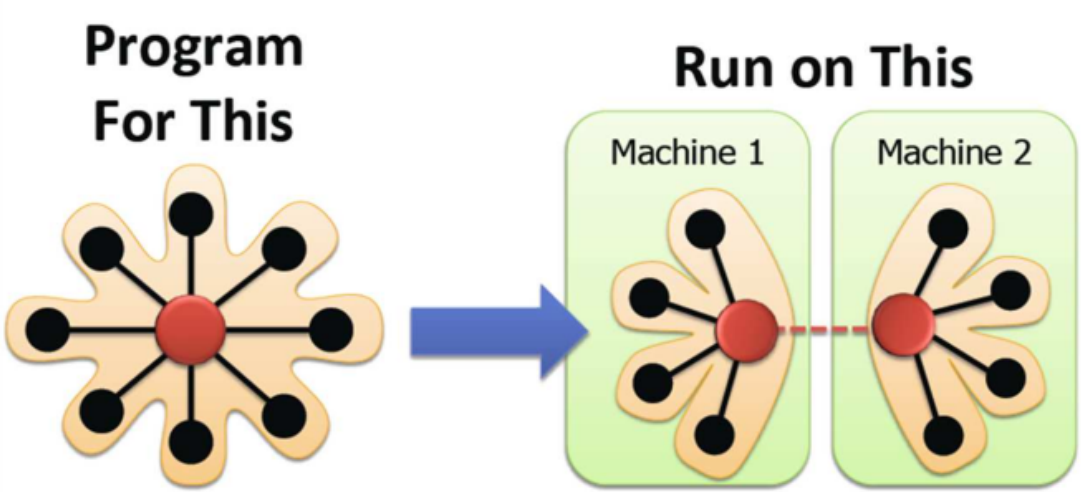
大量自然图满足 Power Law



问题：按点进行图划分会导致图划分之间有大量的跨边，需要传递大量的消息，以及计算和负载不均衡

PowerGraph：解决办法

- 把单一的 Compute() 函数分成三个用户函数GAS来执行
  - Gather
  - Apply
  - Scatter
- 对每个节点上的图数据建立 master 节点和 mirror 节点：实际效果即为把大度 (degree) 节点分裂成为了多个节点



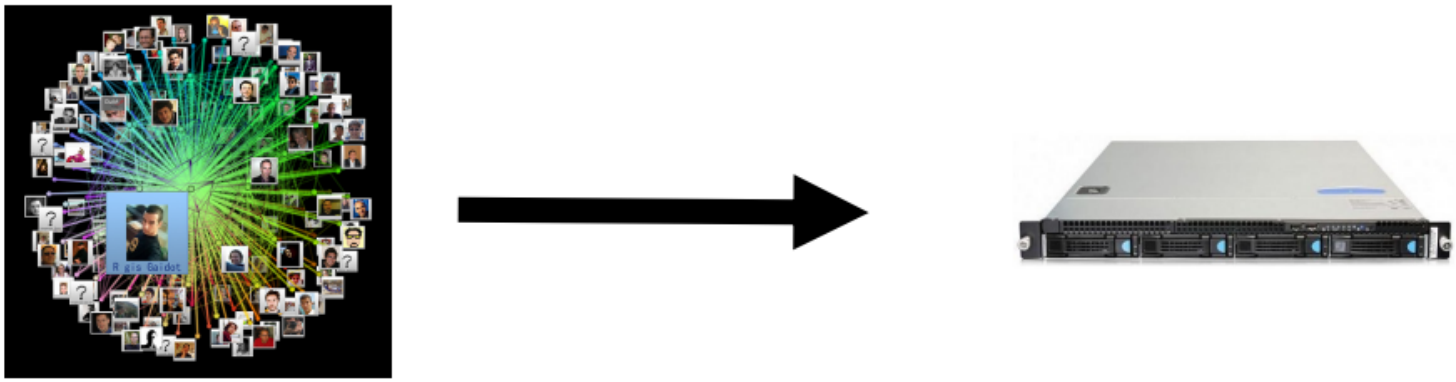
X-Stream

- 目标：在单机上处理大图，以磁盘作为内存扩展

- 磁盘的访问速度比内存慢很多，且以点为中心的模型会随机访问边数据 $\Rightarrow$ 大量的磁盘随机访问

X-Stream

- 目标：在单机上处理大图，以磁盘作为内存扩展

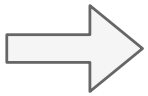


- 磁盘的访问速度比内存慢很多，且以点为中心的模型会随机访问边数据 $\Rightarrow$ 大量的磁盘随机访问

X-Stream：解决方法

- X-Stream 提出了以边为中心的模型，能够顺序访问边集，从而充分利用磁盘带宽

```
for each vertex v
  if v has update
    for each edge e from v
      scatter update along e
```



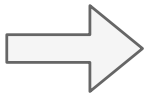
```
for each edge e
  if e.src has update
      scatter update along e
```

- 另一个问题：对顶点集的随机访问

X-Stream：解决方法

- X-Stream 提出了以边为中心的模型，能够顺序访问边集，从而充分利用磁盘带宽

```
for each vertex v
  if v has update
    for each edge e from v
      scatter update along e
```

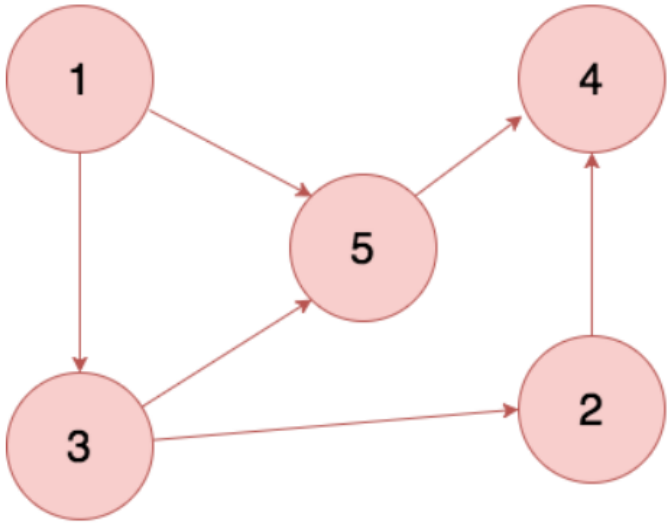


```
for each edge e
  if e.src has update
      scatter update along e
```

- 另一个问题：对顶点集的随机访问

X-Stream

- 问题：顶点集的随机访问



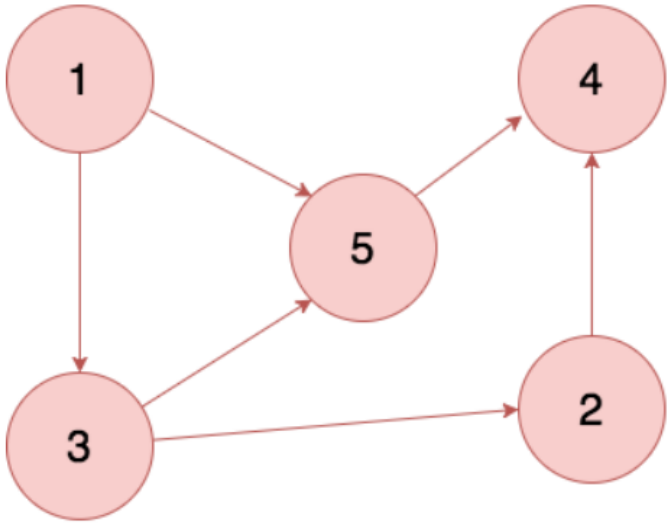
Vertices
1
2
3
4
5

Source	Destination
1	3
1	5
2	4
3	2
3	5
5	4

- 解决办法：将图划分为多个streming partition

X-Stream

- 问题：顶点集的随机访问



Vertices
1
2
3
4
5

Source	Destination
1	3
1	5
2	4
3	2
3	5
5	4

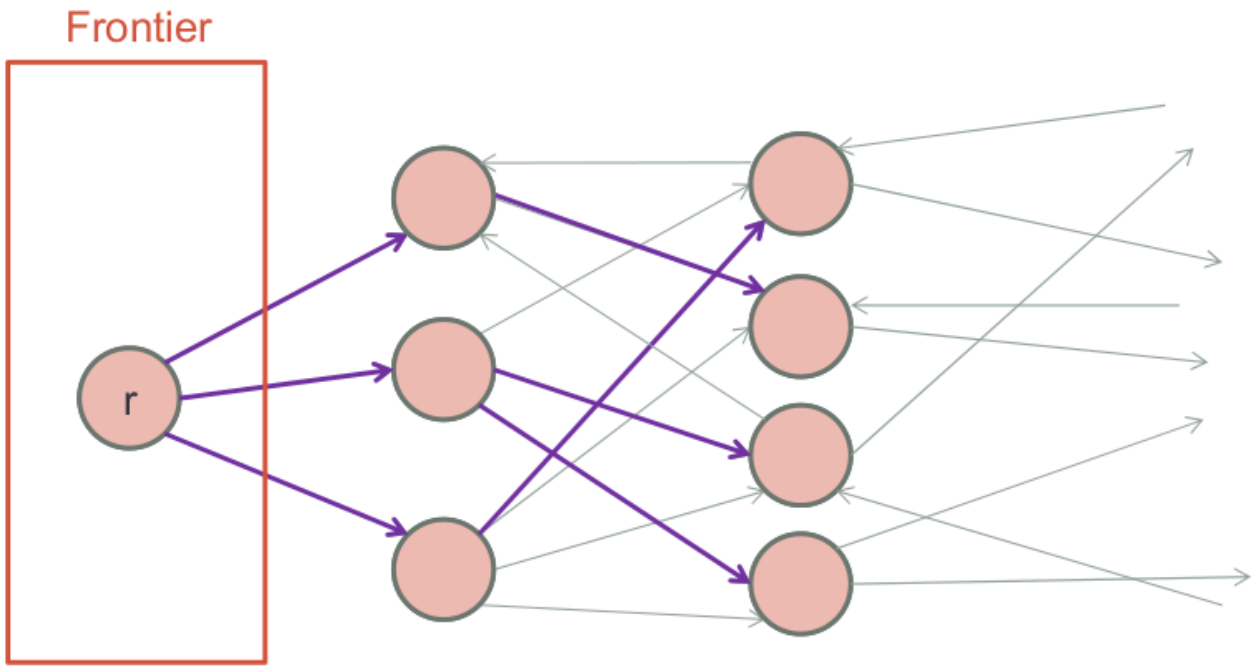
- 解决办法：将图划分为多个streming partition

Streaming Paritions

- Streaming partition 的构成：
  - 顶点集：RAM 能够容纳下的顶点子集
  - 边列表：所有源节点在该 partition 的顶点集中的边
  - 更新列表：所有目的节点在该 partition 的顶点集中的更新
- Streaming partitions 可以被并行处理
- 顶点 (随机访问)⇒ 内存，边 (顺序访问)⇒ 外存
- Parition 的数目对性能很重要
- Shuffle阶段：重排 Scatter 阶段产生的更新

Ligra

- 一个基于共享内存的轻量单机图计算框架：对于"frontier-based" 的图算法非常高效



从源点 r 开始计算 BFS 树，即从 r 能够到达的所有顶点

BFS 抽象

- 在一个顶点子集上进行操作 ⇐ VertexSubset
- 并行地将计算 map 到边集上，然后返回新的顶点子集 ⇐EdgeMap
- 并行地将计算 map 到顶点集上 ⇐ VertexMap

所有图遍历算法
工作方式均如此

因此，Ligra 抽象出了编程接口： VertexMap, EdgeMap

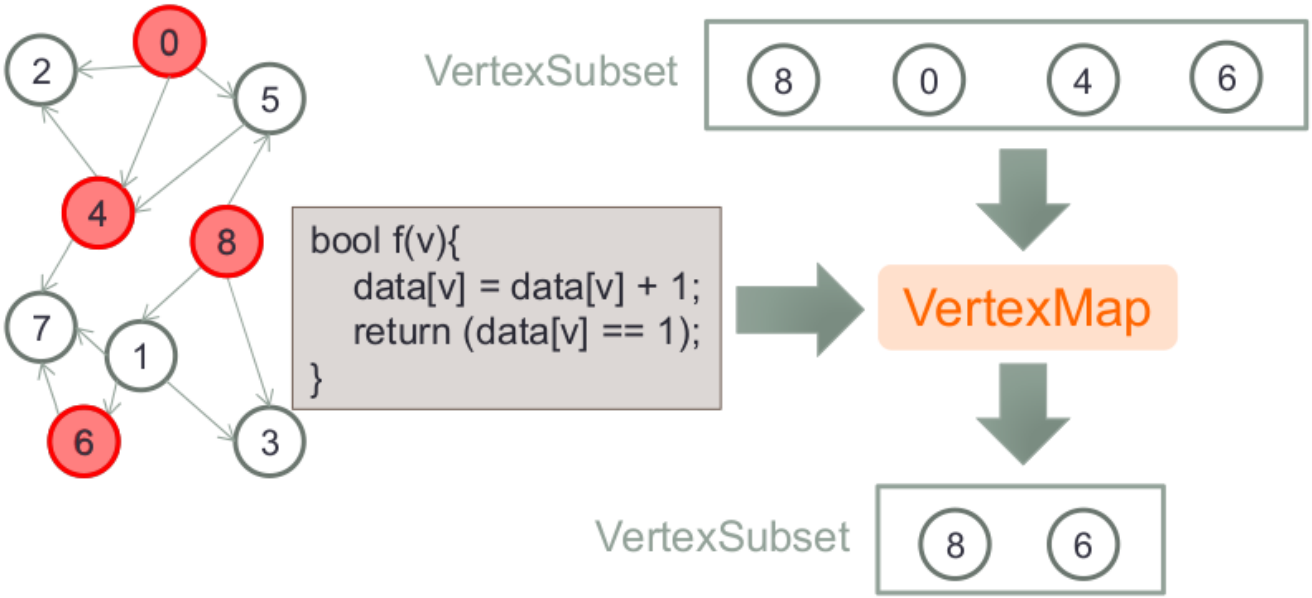
BFS 抽象

- 在一个顶点子集上进行操作 ⇐ VertexSubset
- 并行地将计算 map 到边集上，然后返回新的顶点子集 ⇐EdgeMap
- 并行地将计算 map 到顶点集上 ⇐ VertexMap

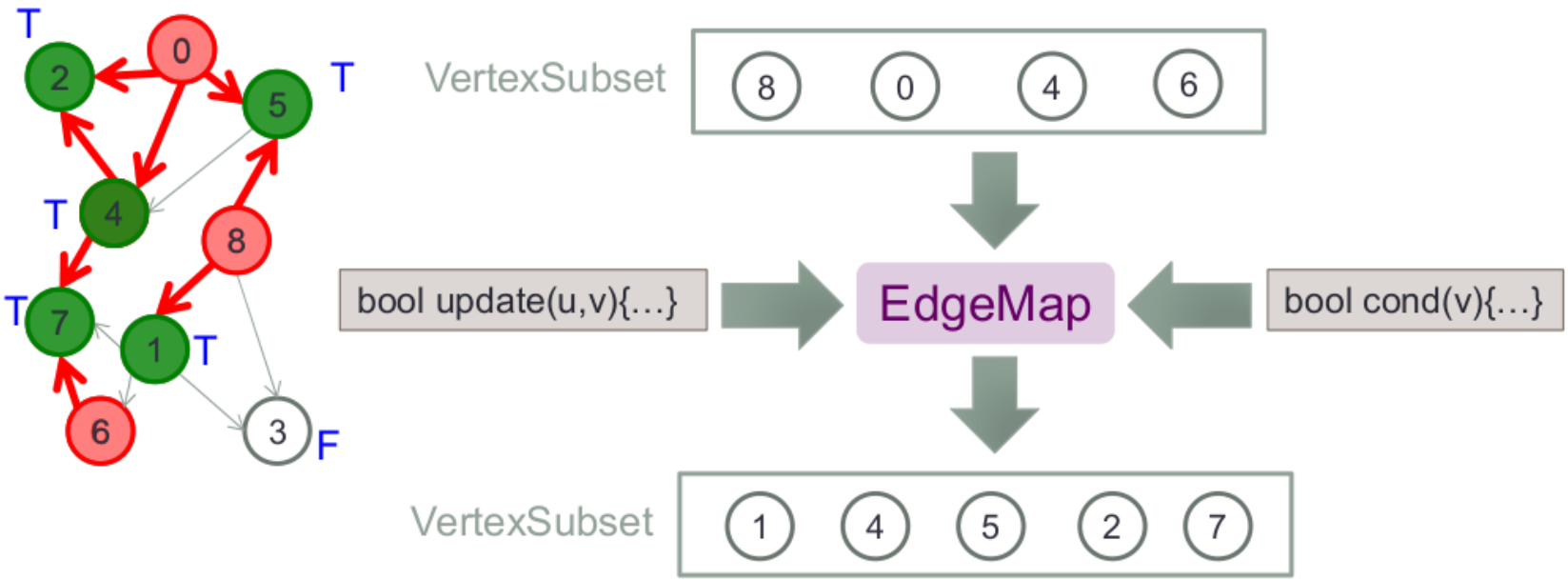
所有图遍历算法
工作方式均如此

因此，Ligra 抽象出了编程接口：VertexMap, EdgeMap

Ligra:VertexMap



Ligra:EdgeMap



EdgeMap: Push-based vs Pull-based

```
procedure EDGEMAP(G, frontier, Update, Cond):
    if (|frontier| + sum of out-degrees > threshold) then:
        return EDGEMAP_DENSE(G, frontier, Update, Cond);
    else:
        return EDGEMAP_SPARSE(G, frontier, Update, Cond);
```

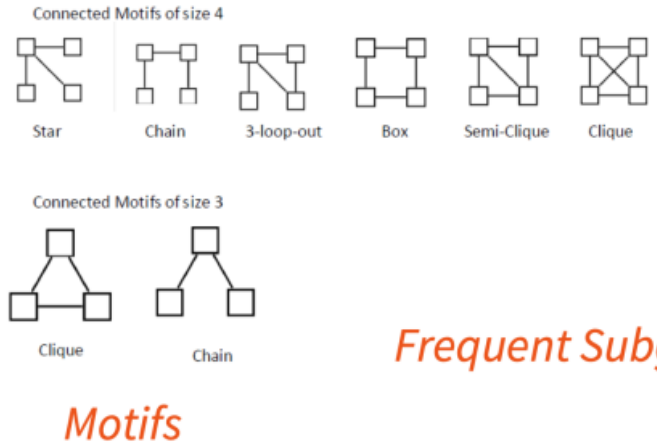
Loop through outgoing edges of frontier vertices in parallel

Loop through incoming edges of "unexplored" vertices (in parallel), breaking early if possible

- 在 Ligra 中，EdgeMap 能够根据frontier 的大小及其出边的数目在 push-based 和 pull-based 两种模式中进行切换

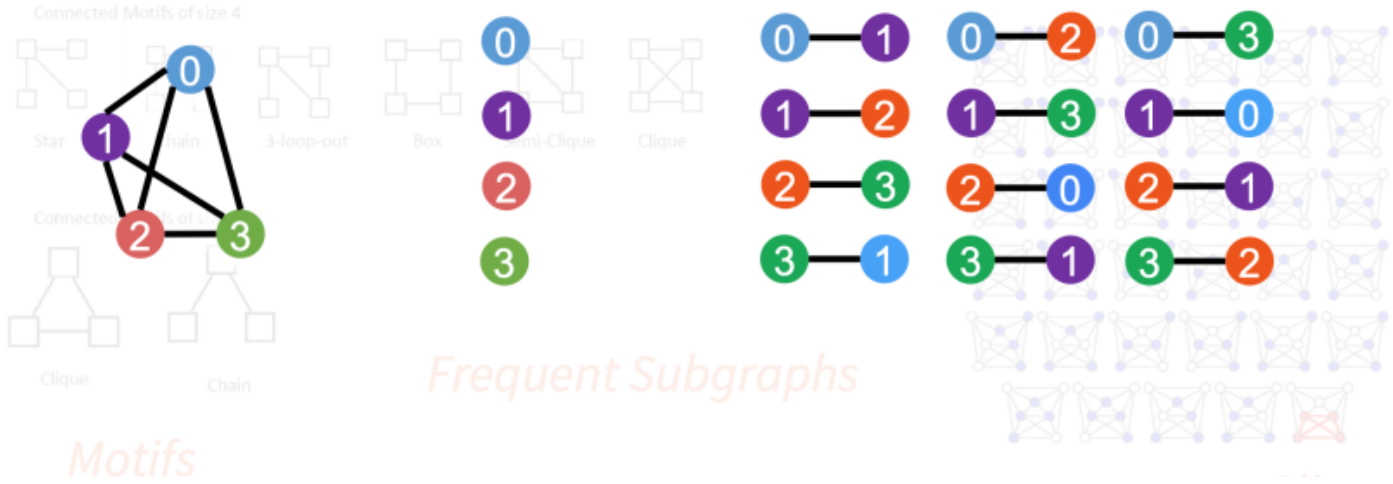
Arabesque

- 图结构挖掘任务：挖掘图中潜在的结构模式



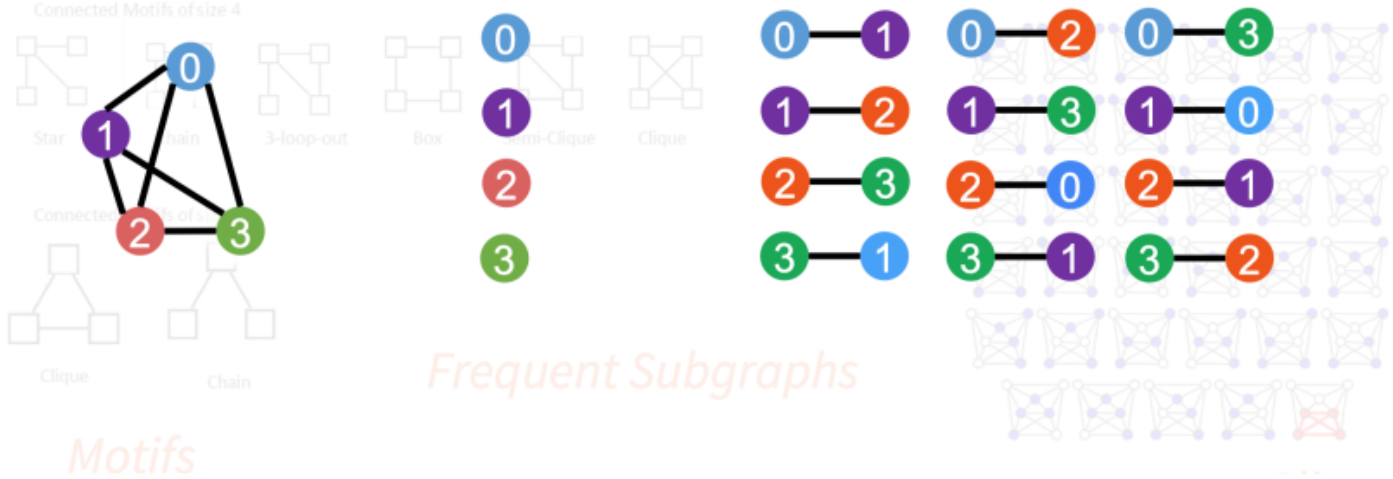
Arabesque

- 图结构挖掘任务：挖掘图中潜在的结构模式
- 标准方法：迭代扩展



Arabesque

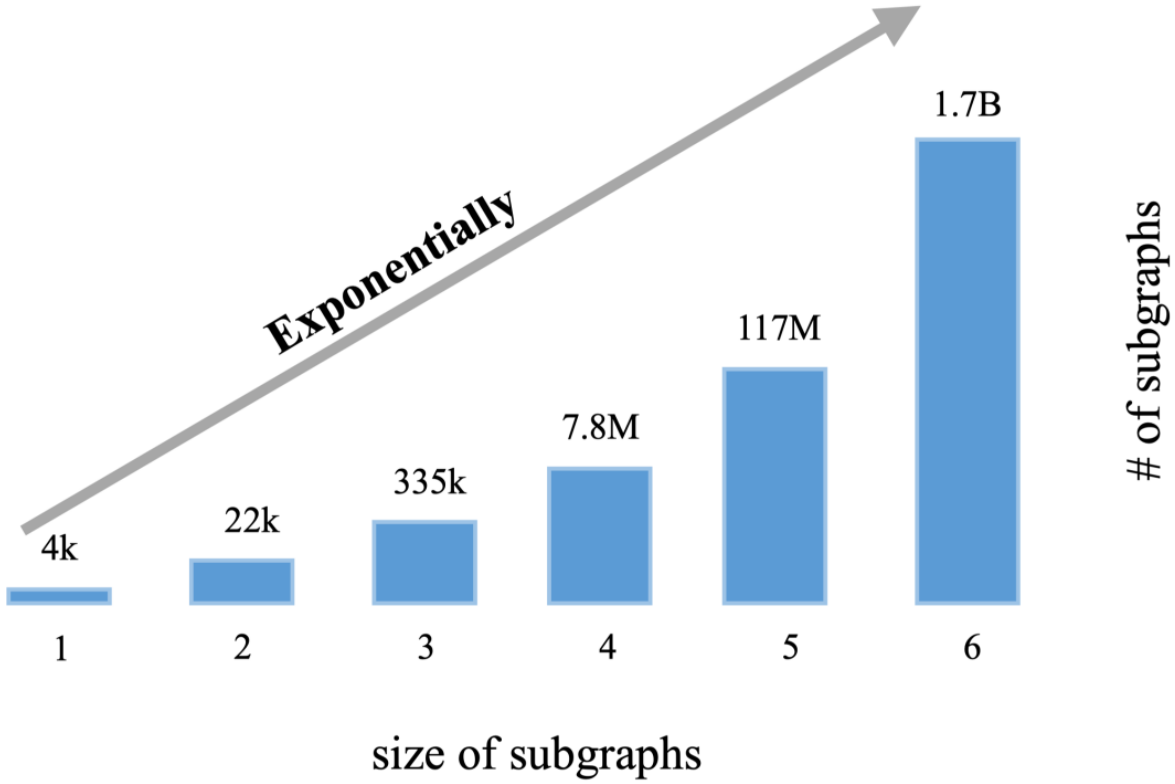
- 图结构挖掘任务：挖掘图中潜在的结构模式
- 标准方法：迭代扩展



- 问题:大量的中间数据

结构挖掘任务的挑战

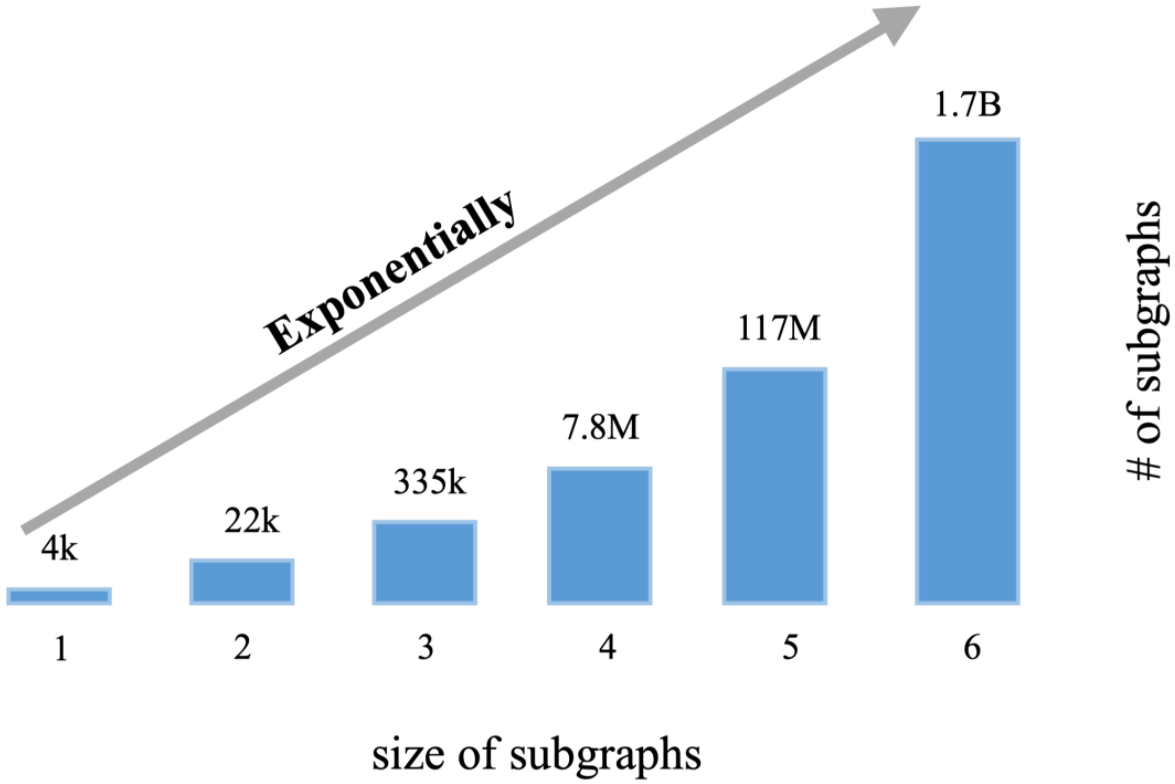
- 子图的数目随着子图的大小成指数增长



- 此外，高效的同构模式检测，生成完整且唯一的 embedding 集合，也是很大挑战

结构挖掘任务的挑战

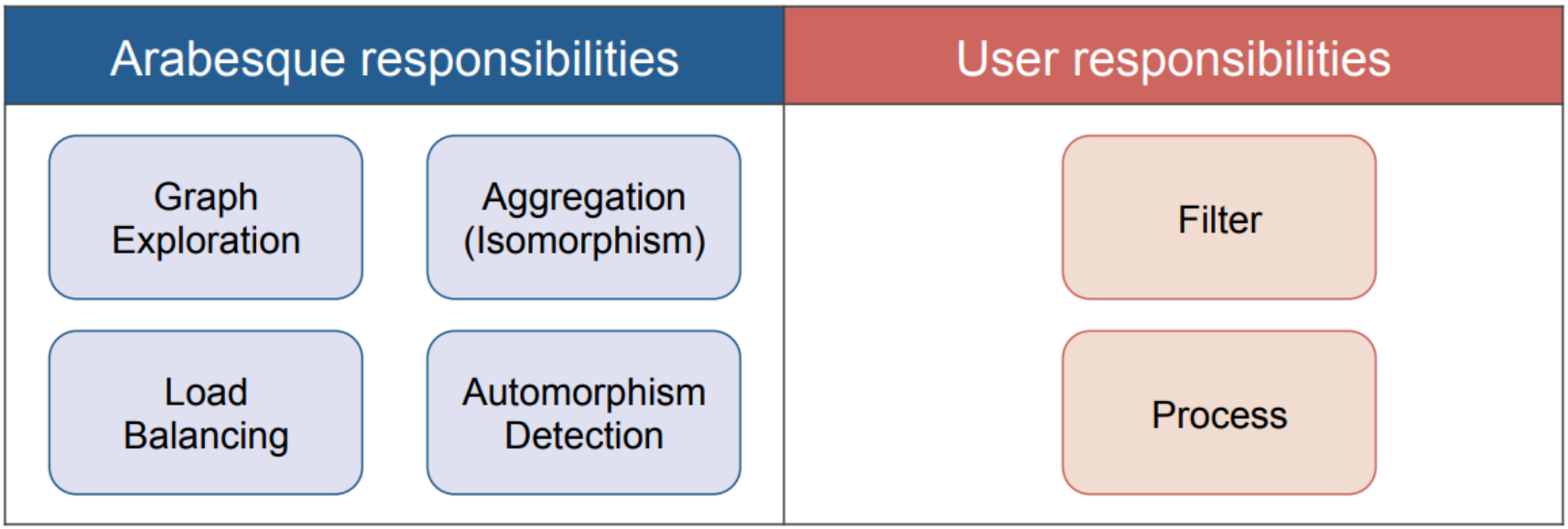
- 子图的数目随着子图的大小成指数增长



- 此外，高效的同构模式检测，生成完整且唯一的 embedding 集合，也是很大挑战

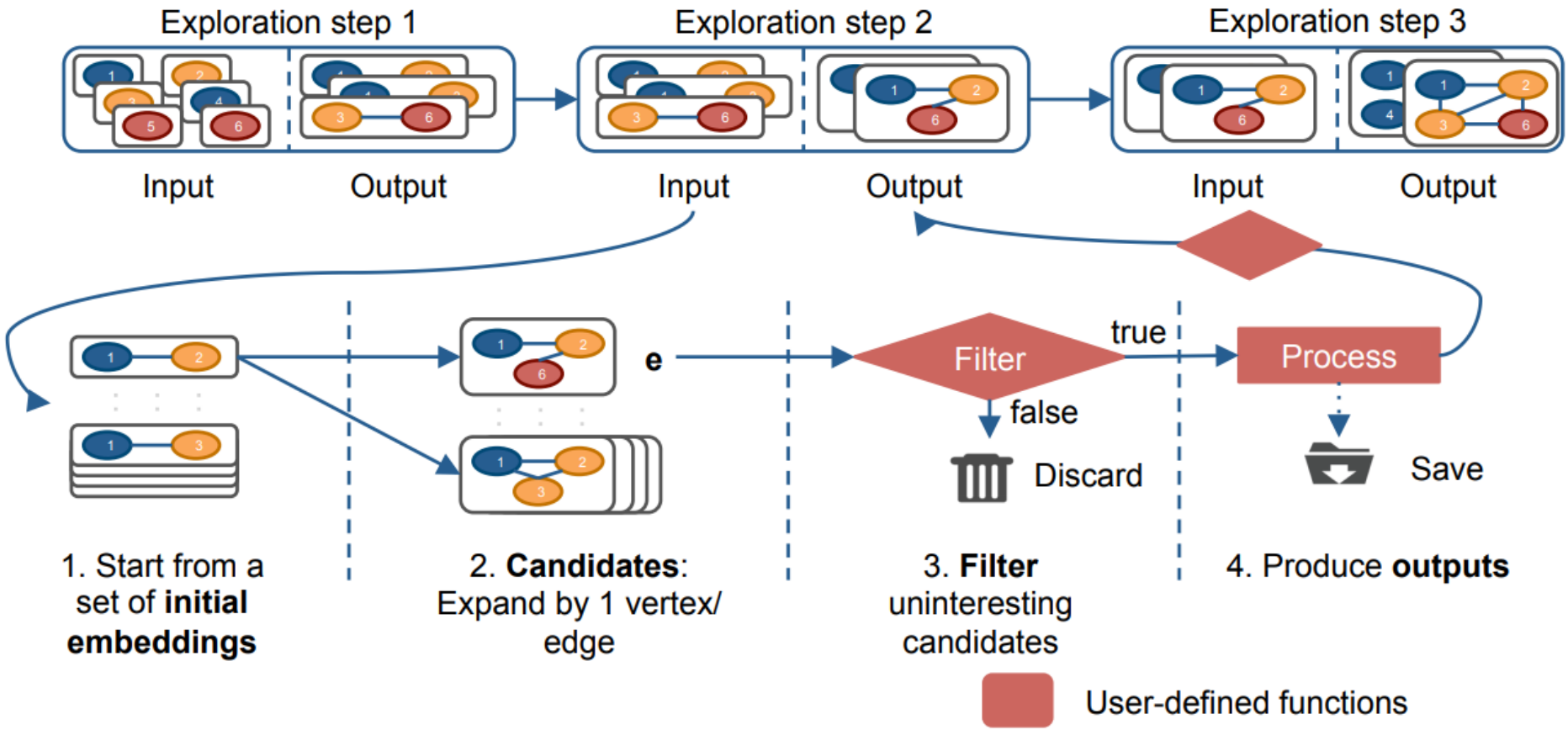
Arabesque

- 一个分布式的图挖掘系统
- 采用以子图为中心的的模型



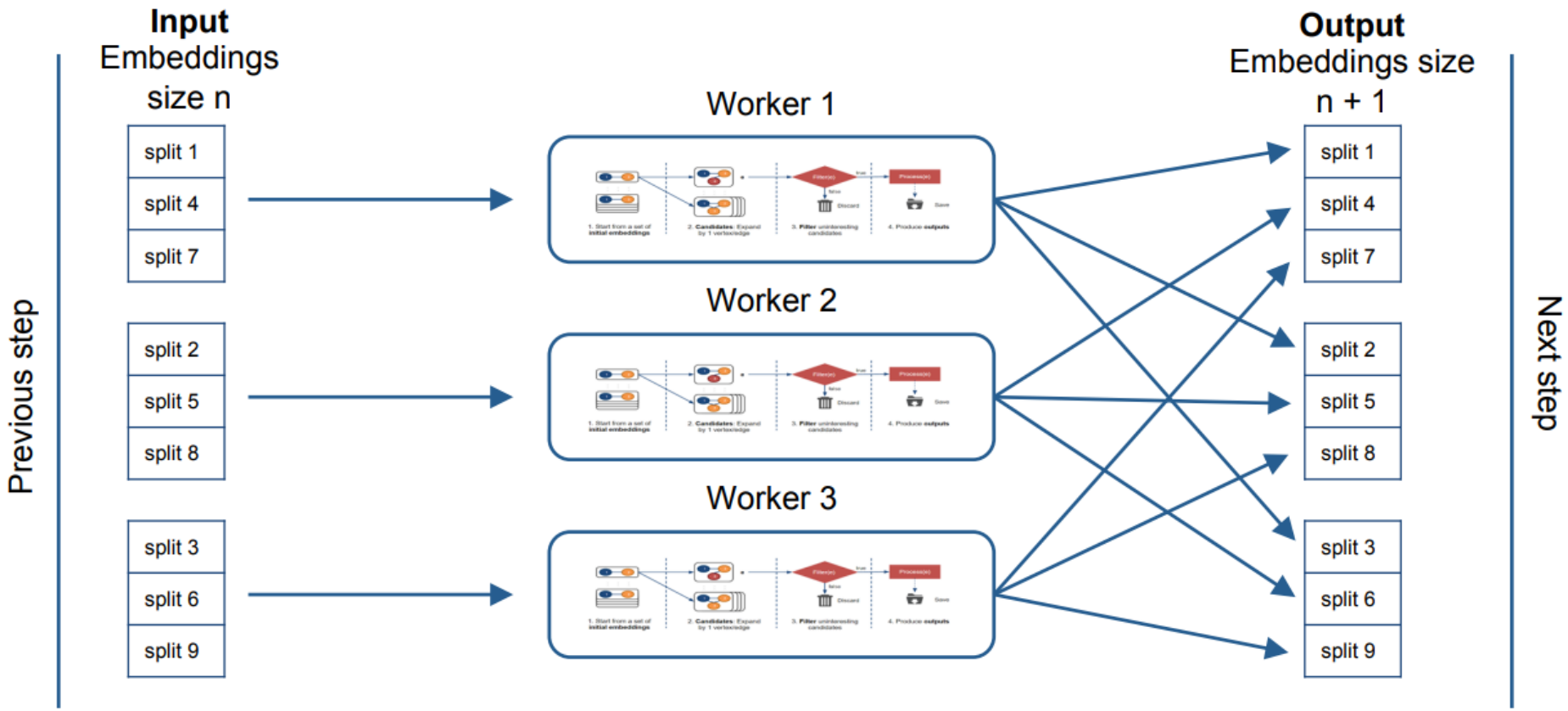
Arabesque：探索过程

- Filter-Process的计算模型



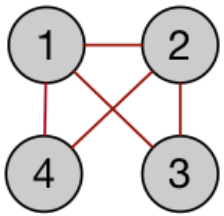
Arabesque：系统架构

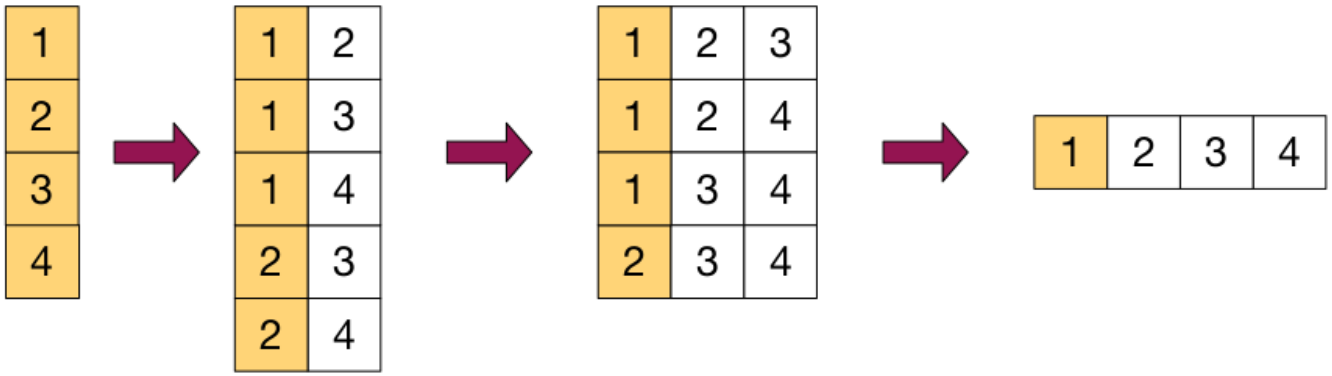
- Arabesque 实现于 Giraph 之上，采用 BSP 模型



Arabesque：生成 embedding

- Canonical embedding：通过首先访问 id 最小的节点，然后递归添加 id 最小的邻居得到

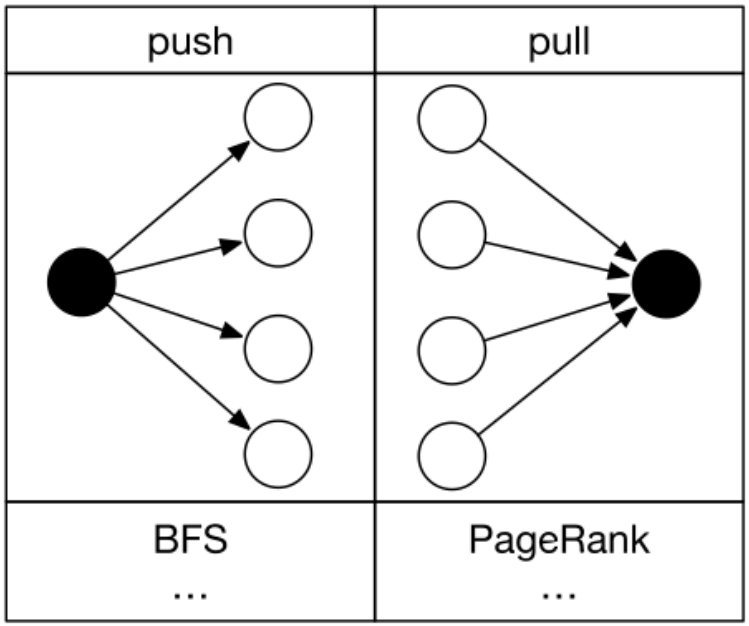




图分析系统的背后

图分析框架的计算模式

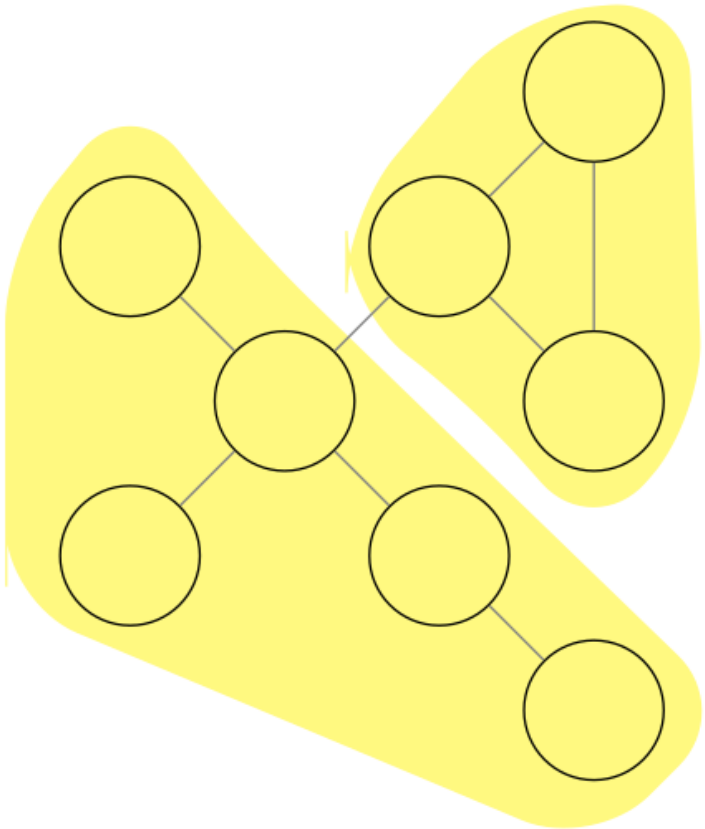
- 以点为中心的计算模式



- 缺点：以点为中心的计算模式会随机访问边数据，从而破坏数据的局部性
- 代表系统：Pregel

图分析框架的计算模式

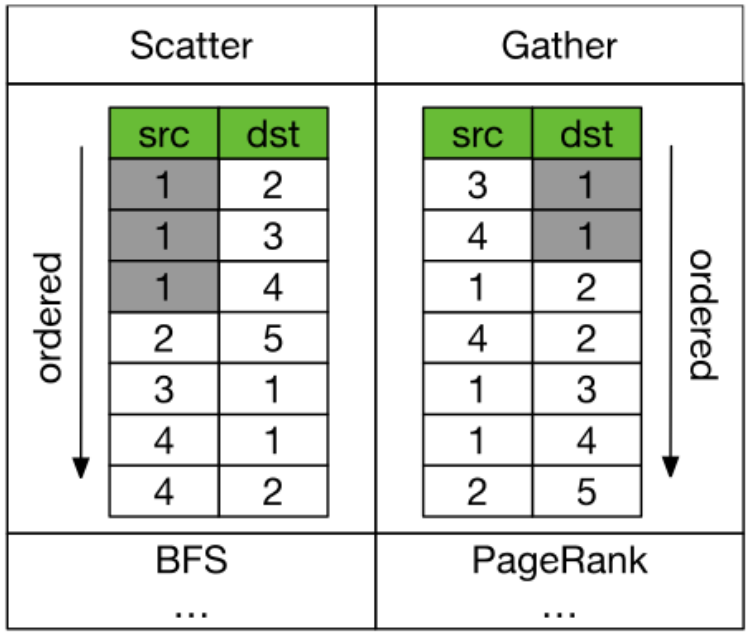
- 以 partition 为中心的计算模式



- 目的：减少节点之间的通信开销

图分析框架的计算模式

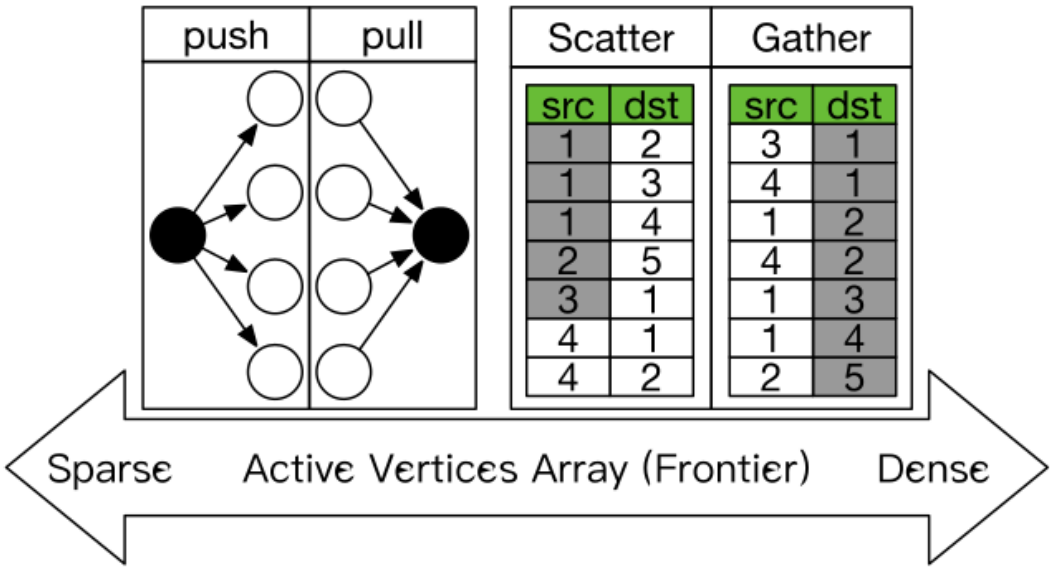
- 以边为中心的计算模式



- 缺点：以边为中心的计算模式会导致边数据的冗余遍历
- 代表系统：X-Stream

图分析框架的计算模式

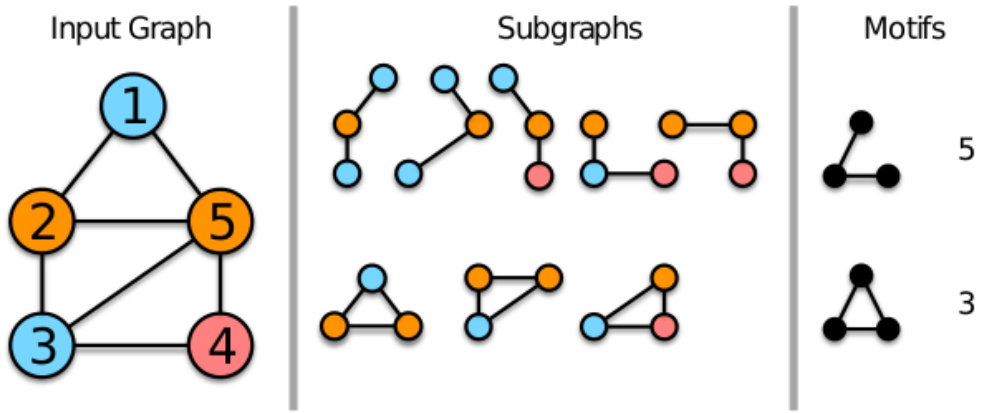
- 点边混合计算模式



- 代表系统：Ligra

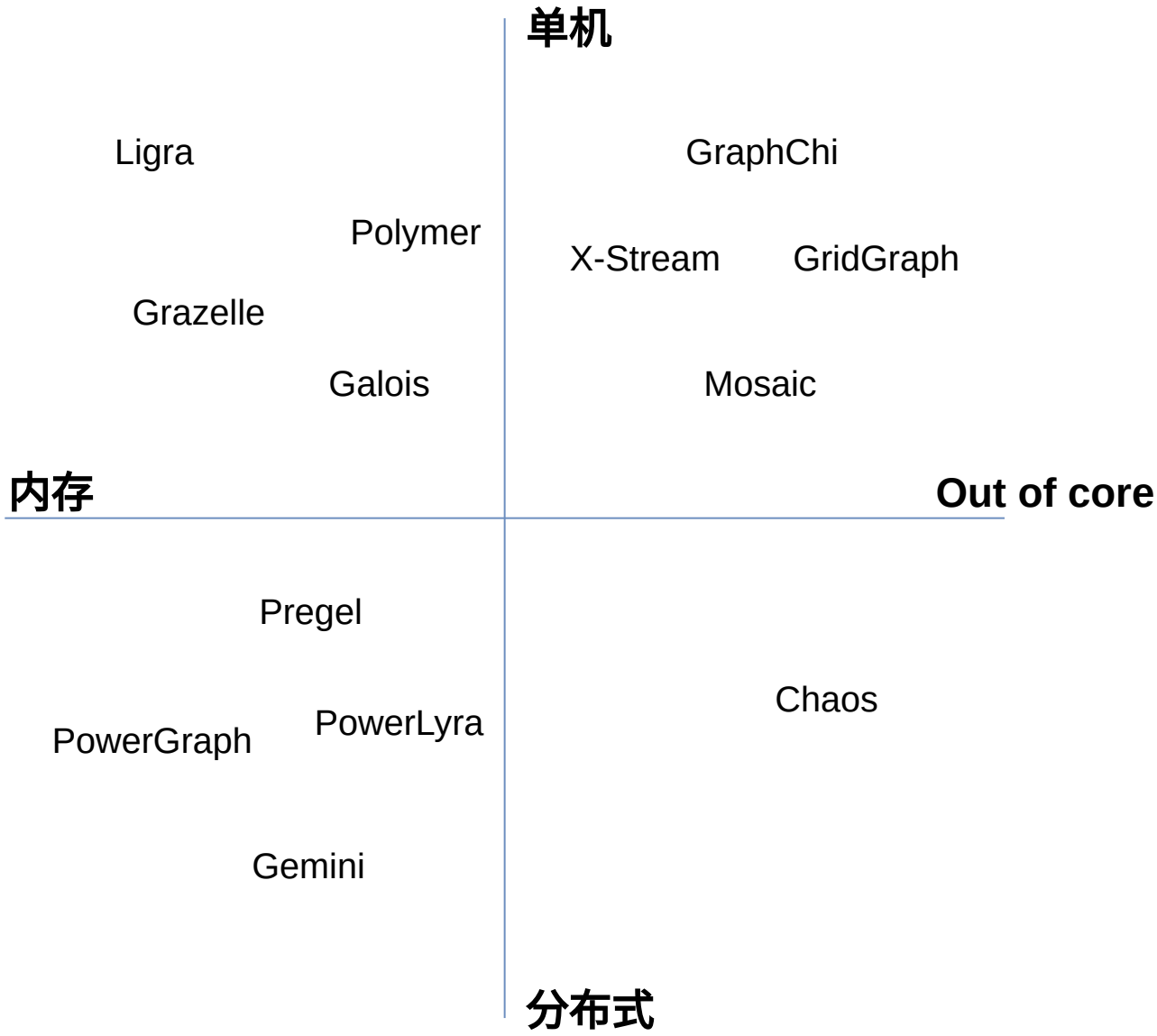
图分析框架的计算模式

- 图结构挖掘框架：以子图为中心的计算模式
  - 需要探索发现所有符合条件的子图
  - 以子图为单位过滤、统计挖掘需要的信息



- 代表系统：Arabesque、RStream

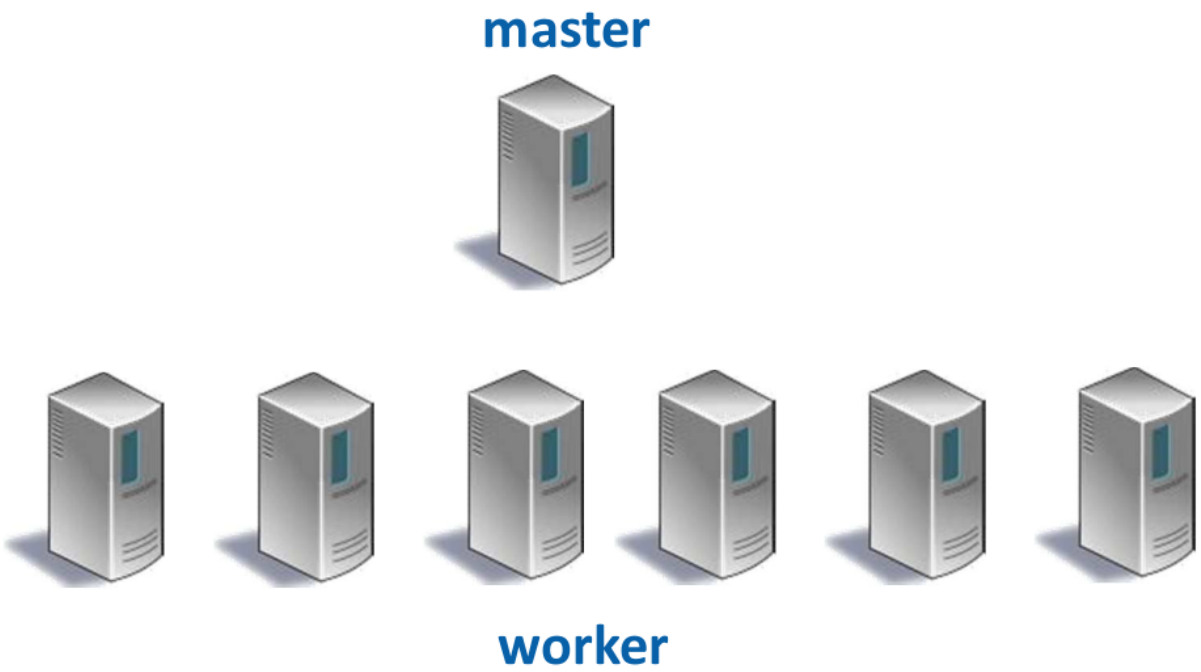
图分析框架的系统架构


单机
Ligra
GraphChi
Polymer
X-Stream
GridGraph
Grazelle
Galois
Mosaic
内存
Out of core
Pregel
Chaos
PowerGraph
PowerLyra
Gemini
分布式

图分析框架的系统架构

分布式共享内存架构：

- 通常采用worker-master机制：master 负责协调工作，worker 负责计算



- 优点：可扩展性好，能够处理超大规模的图
- 缺点：有很大的通信开销和启动开销，系统需要具备良好的容错机制

图分析框架的系统架构

单机共享内存架构：

- 多个线程都可以访问图数据
- 线程之间不用发送和接收消息
- 优点：与分布式集群相比，有较小的通信开销和启动开销，同时，系统的使用、部署简单，性能较好
- 缺点：系统需要良好的一致性模型，避免数据竞争；当图不断增大 $\Rightarrow$ Out of memory

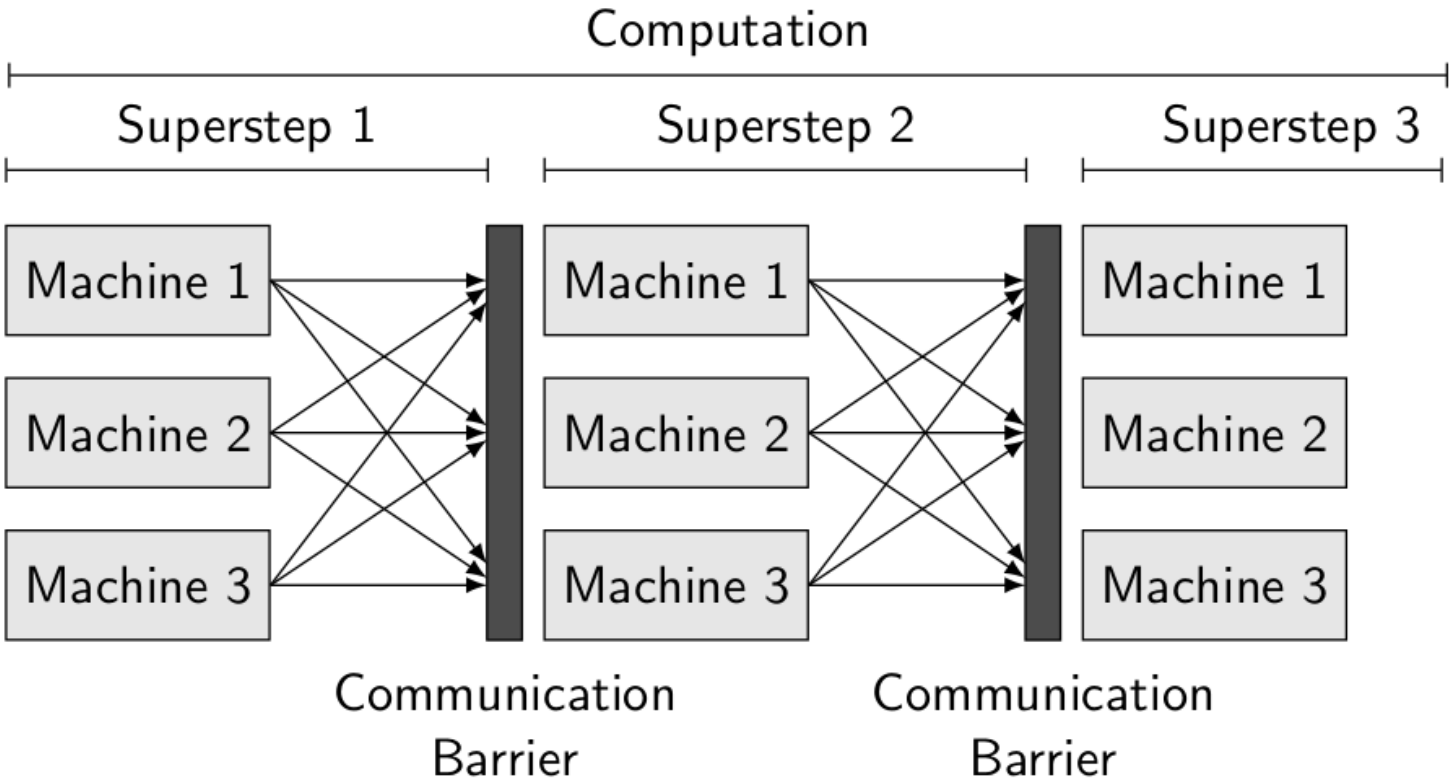
图分析框架的系统架构

单机外存架构：

- 使用外存作为内存的扩展
- 优点：能够在一台普通个人计算机上处理大图
- 缺点：IO 性能瓶颈；由于图数据的不规则，可能带来大量的磁盘随机访问；
  - IO 与计算重叠
  - 尽可能顺序 IO，减少随机 IO

图分析框架的并行模型

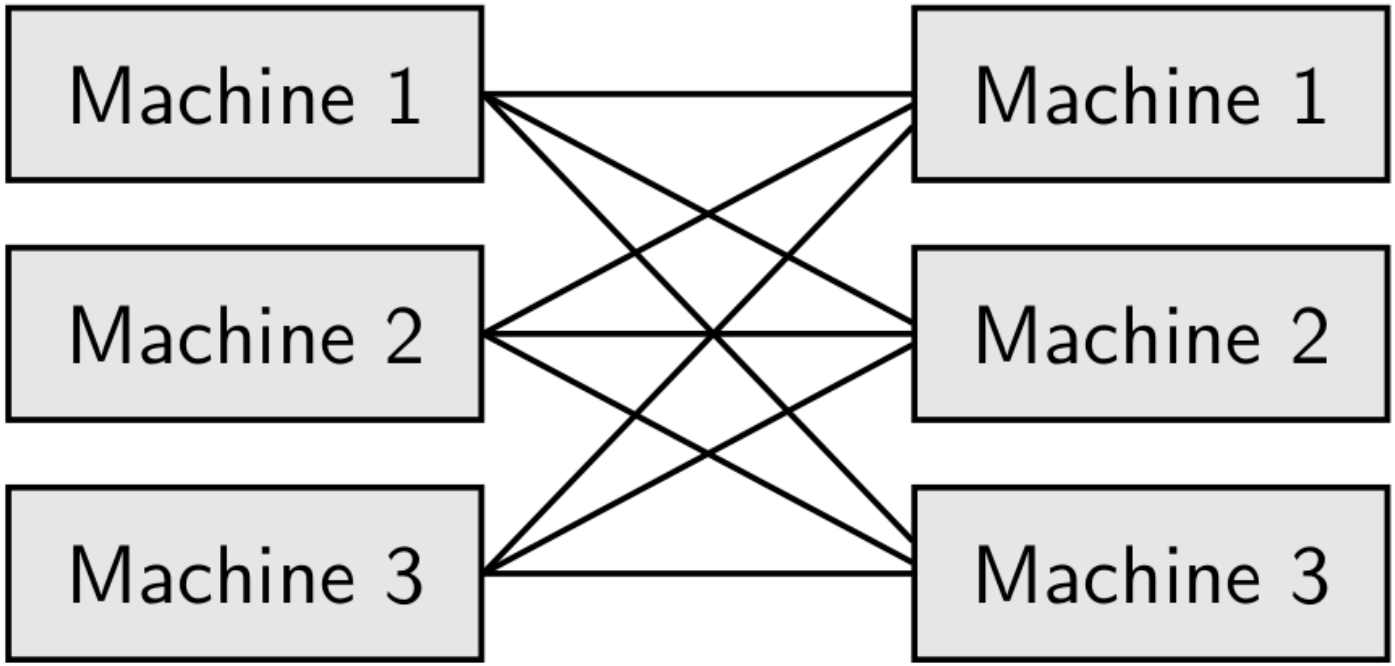
整体同步并行模型（BSP）：



- 计算过程由多个超步组成，超步内并行执行，超步间进行全局同步
- 优点：简化了图算法的分析，易于实现各种并行图算法
- 缺点：全局屏障 ⇒stragglers，超步的速度由性能最差的 worker 决定

图分析框架的并行模型

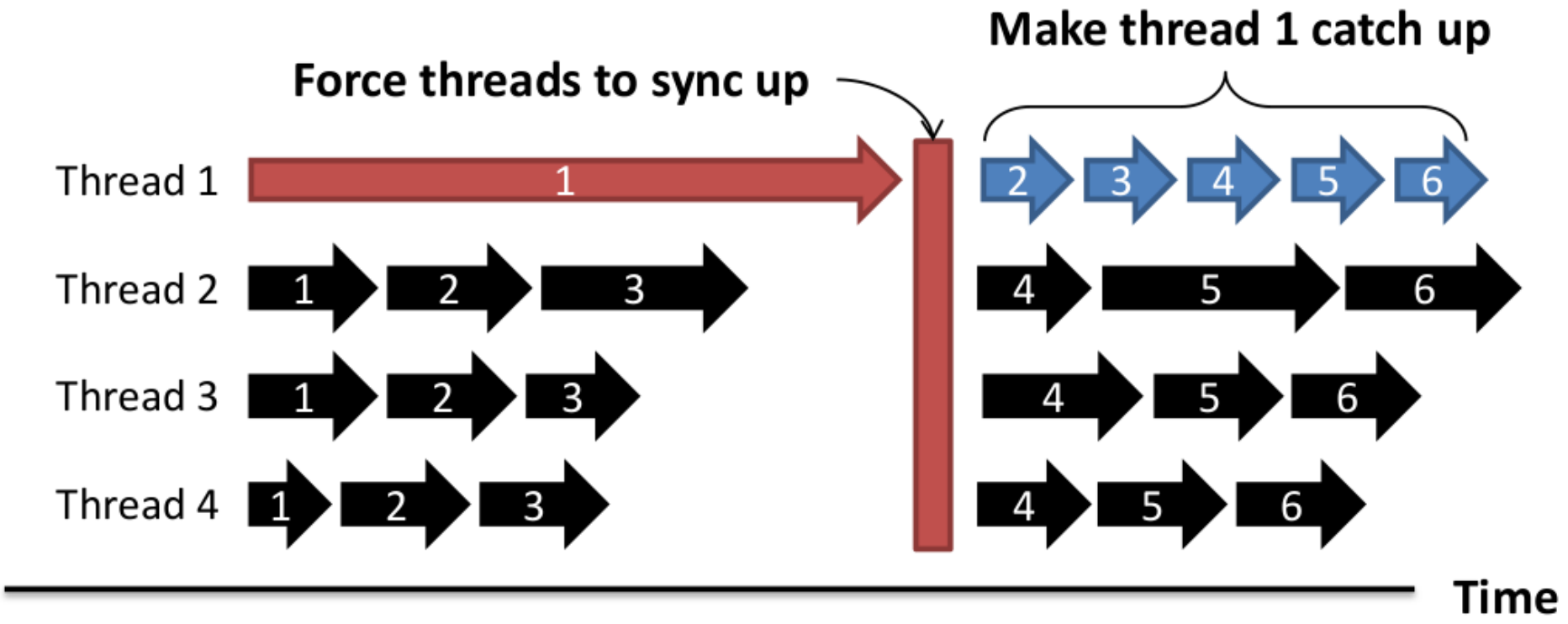
异步并行模型（AP）：



- 优点：没有全局同步屏障，能够消除 stragglers 问题
- 缺点：可能产生大量的“过时”计算，增加不必要的通信开销和计算开销，代码编写和调试困难，算法不一定收敛

图分析框架的并行模型

延迟同步并行模型（SSP）：允许“速度”较快的 worker “超过”较慢的 worker 最多 c 步

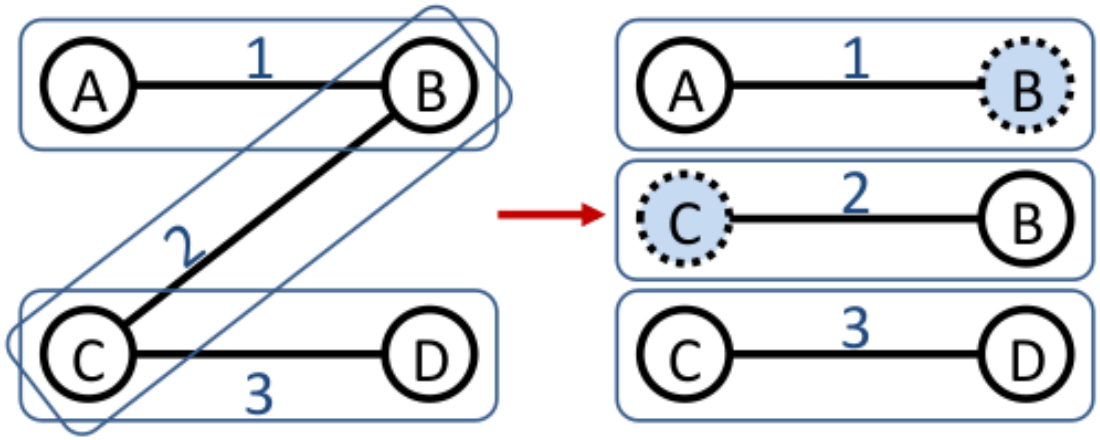


- 优点：结合了 BSP 和 AP 模型的优点，特别适合于分布式机器学习
- 缺点：同样会产生“过时”计算

图分析框架的数据划分方式

分布式内存图分析框架：

- 切点法：PowerGraph、Gemini等

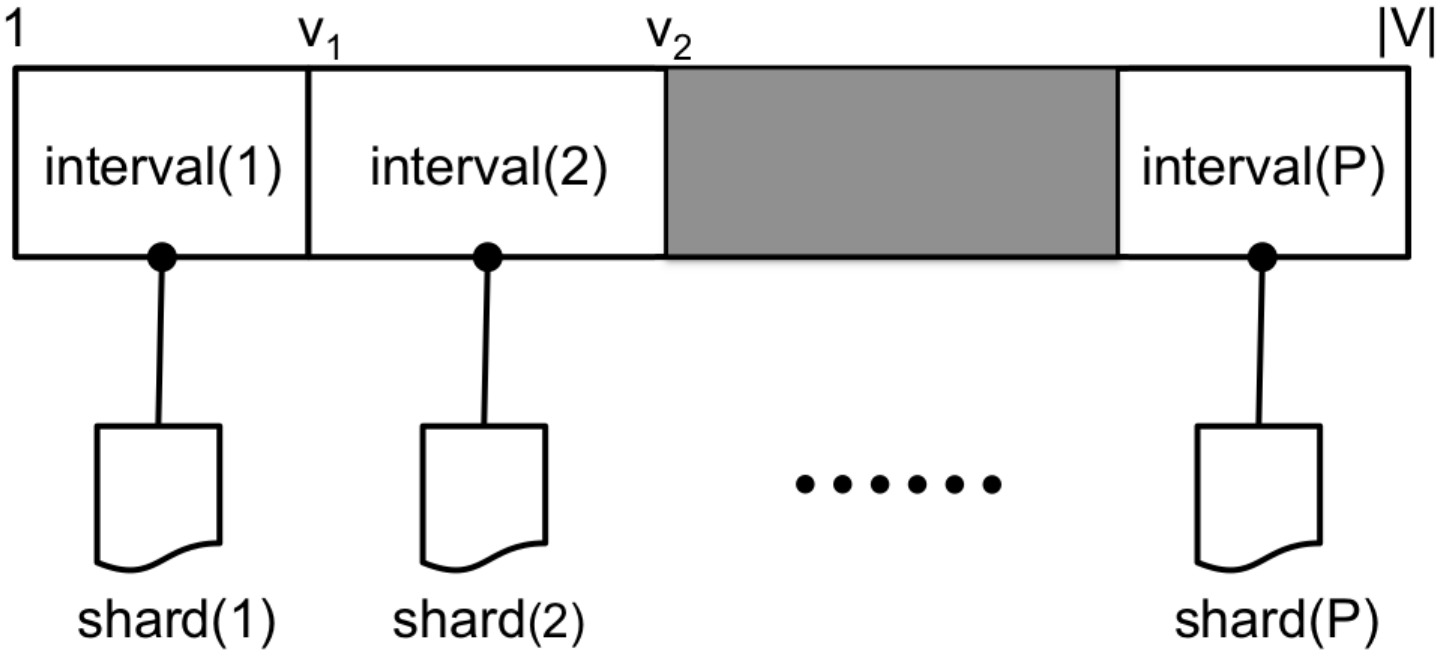


- 缺点：每轮计算会造成大量的通信开销，可能出现节点的计算资源浪费情况

图分析框架的数据划分方式

单机外存图分析框架：

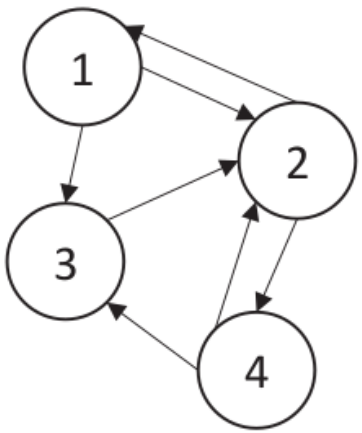
- 一维划分：划分边集数组，GraphChi等

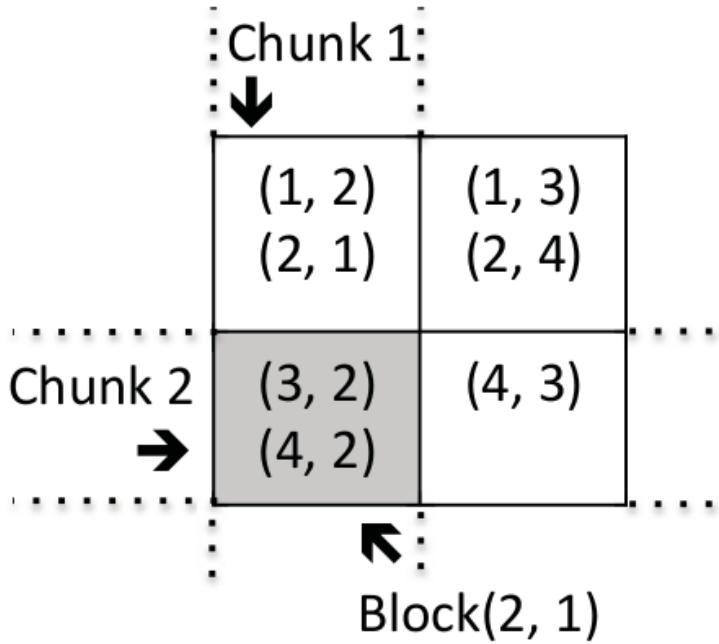


图分析框架的数据划分方式

单机外存图分析框架：

- 一维划分：划分边集数组，GraphChi等
- 二维划分：划分邻接表，GridGraph、Graphene等





- 缺点：加载外存导致计算等待，可能出现算力不足情况

总结

Scale up or Scale out

Scale up：

- 设计、开发、部署、维护简单，开箱即用

Scale out：

- 图不断变大（万亿边）⇒ 只有 scale out 才是可行的
- 复杂的图计算，导致即使图数据本身能够容纳在内存中，算法运行过程中内存依旧不够

近年来，考虑基于 SSD 扩展的单机图分析系统不断增加，成为当前的一大趋势，如FlashGrpah、Graphene、Basc、LUMOS 等

Scale up or Scale out

Scale up：

- 设计、开发、部署、维护简单，开箱即用

Scale out：

- 图不断变大（万亿边）⇒ 只有 scale out 才是可行的
- 复杂的图计算，导致即使图数据本身能够容纳在内存中，算法运行过程中内存依旧不够

近年来，考虑基于 SSD 扩展的单机图分析系统不断增加，成为当前的一大趋势，如FlashGrpah、Graphene、Basc、LUMOS 等

Scale up or Scale out

Scale up：

- 设计、开发、部署、维护简单，开箱即用

Scale out：

- 图不断变大（万亿边）⇒ 只有 scale out 才是可行的
- 复杂的图计算，导致即使图数据本身能够容纳在内存中，算法运行过程中内存依旧不够

近年来，考虑基于 SSD 扩展的单机图分析系统不断增加，成为当前的一大趋势，如FlashGrpah、Graphene、Basc、LUMOS 等

一些开放性问题

- 扩展性：系统如何扩展到超大规模图上
  - 具有数十亿节点，数千亿边的图越来越普遍
- 如何与数据科学工作流集成
  - 当前重点主要在单一计算上
  - 图分析应成为完整工作流的一部分，比如金融分析、诉讼分析等
  - 单一算法 $\Rightarrow$ 系统
- 图上的机器学习任务应受到更多关注：近年来较为热门的图神经网络

一些开放性问题

- 扩展性：系统如何扩展到超大规模图上
  - 具有数十亿节点，数千亿边的图越来越普遍
- 如何与数据科学工作流集成
  - 当前重点主要在单一计算上
  - 图分析应成为完整工作流的一部分，比如金融分析、诉讼分析等
  - 单一算法 $\Rightarrow$ 系统
- 图上的机器学习任务应受到更多关注：近年来较为热门的图神经网络

一些开放性问题

- 扩展性：系统如何扩展到超大规模图上
  - 具有数十亿节点，数千亿边的图越来越普遍
- 如何与数据科学工作流集成
  - 当前重点主要在单一计算上
  - 图分析应成为完整工作流的一部分，比如金融分析、诉讼分析等
  - 单一算法 $\Rightarrow$ 系统
- 图上的机器学习任务应受到更多关注：近年来较为热门的图神经网络

参考文献 I

[1] grzegorz malewicz, matthew h. austern, aart j. c. bik, james c. dehnert, ilan horn, naty leiser, and grzegorz czajkowski, pregel: a system for large-scale graph processing, sigmod' 10, june 6–11,2010, indianapolis, indiana, usa.
[2] amitabha roy, ivo mihailovic, willy zwaenepoel, x-stream: edge-centric graph processing using streaming partitions, sosp ' 13, nov 03-06 2013, farmington, pa, usa
[3] julian shun,guy e. blelloch, ligra: a lightweight graph processing framework for shared memory, ppopp' 13, february 23–27, 2013, shenzhen.
[4] carlos h. c. teixeira ∗ , alexandre j. fonseca, marco serafini,georgos siganos, mohammed j. zaki, ashraf aboulnaga, arabesque: a system for distributed graph mining, sosp' 15, october 4–7, 2015, monterey, ca, usa.
[5] joseph e. gonzalez,yucheng low,haijie gu,danny bickson,carlos guestrin, powergraph: distributed graph-parallel computation on natural graphs, osdi ' 12

参考文献 II

[6] Xiaowei Zhu, Wenguang Chen, Weimin Zheng, and Xiaosong Ma, Gemini: A Computation-Centric Distributed Graph Processing System, OSDI' 16
[7] Aapo Kyrola, Guy Blelloch, Guy Blelloch, GraphChi: Large-Scale Graph Computation on Just a PC, OSDI' 12
[8] Xiaowei Zhu, Wentao Han, and Wenguang Chen, GridGraph: Large-Scale Graph Processing on a Single Machine Using 2-Level Hierarchical Partitioning, USENIX ATC ' 15
[9] Narayanan Sundaram, Nadathur Satish, Md Mostofa Ali Patwary, Subramanya R Dulloor,Michael J. Anderson, Satya Gautam Vadlamudi, Dipankar Das and Pradeep Dubey, GraphMat: High performance graph analytics madeproductive, Proceedings of the VLDB Endowment, Vol. 8, No. 11
[10] Da Zheng, Disa Mhembere, Randal Burns,Joshua Vogelstein,Carey E. Priebe,Alexander S. Szalay, FlashGraph: Processing Billion-Node Graphs on an Array of Commodity SSDs, FAST ' 15

参考文献 III

[11] Hang Liu,H. Howie Huang, Graphene: Fine-Grained IO Management for Graph Computing, FAST' 17
[12] Keval Vora, L UMOS : Dependency-Driven Disk-based Graph Processing, ATC2019
[13] Kai Wang,Zhiqiang Zuo,John Thorpe,Tien Quang Nguyen,Guoqing Harry Xu, RStream: Marrying Relational Algebra with Streaming for Efficient Graph Mining on A Single Machine, OSDI' 18
[14] Daniel Mawhirter,Bo Wu, AutoMine: Harmonizing High-Level Abstraction and High Performance for Graph Mining, SOSP ' 19, October 27–30, 2019, Huntsville, ON, Canada
[15] Yanfeng Zhang, Qixin Gao, Lixin Gao, Cuirong Wang, Maiter: An Asynchronous Graph Processing Framework for Delta-based Accumulative Iterative Computation

参考文献 IV

[16] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2014, pp. 701–710.
[17] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "Line: Large-scale information network embedding," in Proceedings of the 24th international conference on world wide web. International World Wide Web Conferences Steering Committee, 2015, pp. 1067–1077.
[18] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2016, pp. 855–864.
[19] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A comprehensive survey on graph neural networks," arXiv preprint arXiv:1901.00596, 2019.

谢谢