Parallel Graph Computations: Foundation and Practice
Wenfei Fan
University of Edinburgh

Backgrounds

What is the right parallel graph computation model?
- **GRAPE**: automatically parallelizing graph computation
- **AAP**: adaptive asynchronous parallel

How to partition the graph under various circumstances?
- Dynamically re-partitioning a graph when adding or removing processors
- Partitioning the graph based on application scenarios
- Incrementalizing existing graph partitioners

How to efficiently carry out local computation?
- Graph contraction: making big graphs small
- Incrementalize existing graph algorithms

Parallel graph computation model

**GRAPE** programming model: plug and play
- Partial evaluation **PEval**
- Incremental evaluation **IncEval**
- Assemble partial results

**AAP**: each worker maintains parameters and dynamically adjusts:
- Its progress relative to other workers
- The staleness of its accumulated changes

Unique properties
- Dynamic model switch: reduce the total rounds
- Programming simplicity
- Ready to optimally utilize other platforms:
  - **MapReduce, BSP, AP, SSP** and **PRAM**
- Convergence guarantee: Church-Rosser

Outperform the state-of-the-art systems
- Outperforming **BSP, AP, SSP** by 4.3X, 14.7X and 4.7X on average
- 30X times faster than Petuum on collaborative filtering

Industrialized as **GraphScope** in Alibaba:
Open source: https://github.com/alibaba/GraphScope

Graph contraction

Graph contraction: making big graphs small
- Generically: support different classes of queries at the same time
- Losslessness: compute exact answer

Contraction scheme ($f_C, S, f_D$)
- Contraction function $f_C$: subgraphs to supernodes
- Synopsis function $S$: annotate supernodes
- Decontraction function $f_D$: restore supernodes to subgraphs

Algorithms:
- Contraction following deterministic order
- Bounded Incremental contraction

Incrementalization

Incremental algorithms: $A(G + \Delta G) = A(G) \oplus A_\Delta(G, \Delta G)$
Incrementalize batch algorithms
- Generating an incremental algorithm $A_\Delta$ from the batch algorithm $A$
- Reusing the original logic and data structures
- Providing performance guarantees

Fixpoint model for batch algorithm $A$:

$$ (D^{t+1}_A, H^{t+1}_A) = f_A(D^t_A, Q, G, H^t_A) $$
- Status $D_A$, status variable $H_A$, and step function $f_A$

Incrementalization:
- Initial scope function $h$: initializing status $D^0_A, H^0_A$ from $\Delta G$ and previous result
- Reusing the same $D_A, H_A$ and $f_A$
- Reiterating from $D^0_A, H^0_A$ until convergence

**Inheritance**: every fixpoint algorithm $A$ is incrementalizable

Relative boundedness: correct and bounded $h + Church$-Rosser $A$

Heterogeneous application

Heterogeneous entity resolution: match entities across relations and graphs
- Convert relations into canonical graphs $G_D$
- Parallelize parametric simulation across $G$ and $G_D$
- Integrate simulation with ML models to assess semantic closeness

Delivered

13 top-tier publications:
- 2022: VLDB [1], TKDE [2]
- Past: 2021 [3, 4, 5, 6, 7], 2020 [8, 9, 10, 11], 2019 [12], and 2018 [13]

To develop a package of solutions for parallel graph computations

Publications