**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 scenario
- Incrementalizing existing graph partitioners

How to efficiently carry out local computation?
- Graph contraction: making big graphs small
- Incrementalizing 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

![Graph contraction](image)

### Graph contraction

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

### Incrementalization

Incremental algorithms: $\mathcal{A}(G \oplus \Delta G) = \mathcal{A}(G) \oplus \mathcal{A}_\Delta(G, \Delta G)$

- Incrementalize batch algorithms
- Generating an incremental algorithm $\mathcal{A}_\Delta$ from the batch algorithm $\mathcal{A}$
- Reusing the original logic and data structures
- Providing performance guarantee

#### Fixpoint model for batch algorithm $\mathcal{A}$

$$f_A(D^A_0, H^A_0) = f_A(D^A_1, Q, G, H^A_1)$$

- Status $D_A$, status variable $H_A$, and step function $f_A$

#### Incrementalization

- Initial scope function $f$: 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

**Incrementalizability**: every fixpoint algorithm $\mathcal{A}$ is incrementalizable

**Relative boundedness**: correct and bounded $h + \text{Church-Rosser} \mathcal{A}$

### Delivered

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

To develop a package of solutions for parallel graph computations

## Partitioning

**Dynamic scaling**
- Updating the partition $\Pi(n + k)$ from $\Pi(n)$ when add $k$ servers with minimal migration cost, subject to balance factor $b$ and replication factor $f$
- Tri-criteria optimization problem: NP-complete
- Approximation algorithms: extending consistent hashing on graphs

Application driven partitioning:
- Finding partition $\Pi(n)$ to minimize the cost $C(\mathcal{A})$ of algorithm $\mathcal{A}$
- Learned cost model: unifying computation and communication
- Hybrid partitioners: combining edge-cut and vertex-cut

**Incrementalization of partition algorithms**
- Tri-criteria optimization problem: NP-complete
- A generic approach based on fixpoint model

**Partition transparency for algorithm $\mathcal{A}$**
- Conditions: monotonic + robustness under vertex cut
- Example: SSSP, WCC, PageRank, etc.

$\mathcal{A}$ works for any partition without requiring any change to $\mathcal{A}$

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

---

**Publications**