Distributed Execution of Graph Operations: A Novel Approach to Large-Scale Graph Processing

Pierre Sutra Others

Télécom SudParis

November 22, 2025

arXiv:2504.19495

Outline

Motivation

- Graph processing is fundamental to many applications:
  - Social networks analysis
  - Recommendation systems
  - Knowledge graphs
- Challenge: Processing large-scale graphs efficiently
- Existing systems have limitations:
  - Limited scalability
  - High communication overhead
  - Poor fault tolerance
- **Goal:** Develop a distributed graph processing system that overcomes these limitations

Problem Statement

Formal Problem:

Graph Processing Challenge

Given a large-scale graph $G = (V, E)$ with $|V| = n$ vertices and $|E| = m$ edges:

- Distribute graph data across k nodes
- Execute operations with minimal communication
- Ensure consistency of distributed state
- Achieve near-linear scalability

Key Requirements:

- Low latency for common operations
- High throughput under concurrent load
- Fault tolerance and recovery

System Overview: DEGO Architecture

Core Components:

- **Partitioning Layer:** Smart graph partitioning
- **Execution Engine:** Distributed query processing
- **Communication Layer:** Optimized message passing
- **Consistency Protocol:** State synchronization

Architecture Diagram

Key Innovation:

- Hybrid push-pull execution model
- Adaptive load balancing

Graph Partitioning Strategy

Challenges in Graph Partitioning:

- Minimize edge cuts
- Balance load across nodes
- Handle dynamic graphs

Our Approach:

- Vertex-centric partitioning with locality awareness
- Hash-based initial distribution: $h(v) \mod k$
- Refinement based on:
  - Edge connectivity
  - Access patterns
  - Load metrics

Benefit: Reduces cross-partition communication by 40%

Distributed Execution Model

Hybrid Push-Pull Model

Push Phase:

- Active vertices push updates to neighbors
- Suitable for dense subgraphs

Pull Phase:

- Vertices pull updates from neighbors
- Efficient for sparse graphs and convergence

Adaptive Selection:

- Runtime decision based on active vertex ratio
- Switches between push/pull dynamically
- Optimizes communication cost per iteration

Main Theorem

Theorem (Communication Complexity)

For a graph $G = (V, E)$ partitioned across k nodes with maximum edge cut c, DEGO achieves:

$$T_{comm} = O\left(\frac{c}{k} \cdot \log k\right)$$

per iteration, where T_{comm} is the communication cost.

Comparison with Prior Work:

- Pregel: $O(c)$ per iteration
- GraphLab: $O(c \cdot \log k)$ per iteration
- **DEGO:** $O(\frac{c}{k} \cdot \log k)$ per iteration

Improvement: k-factor reduction in communication overhead

Proof Sketch: Communication Bound

Key Insight: Batching and pipelining reduce message complexity

Proof Steps:

1. **Message Aggregation:**
   - Group updates by destination node
   - Batch size: $O(c/k)$ messages per node pair
2. **Hierarchical Communication:**
   - Use tree-based aggregation
   - Height: $O(\log k)$
   - Each level processes c/k messages
3. **Pipeline Optimization:**
   - Overlap computation and communication
   - Amortize synchronization cost

Proof Sketch: Correctness

Consistency Guarantees:

- **Eventual Consistency:** All nodes converge to same state
- **Ordering:** Causal consistency preserved via vector clocks
- **Convergence:** Bounded staleness model

Proof Technique:

- Define state transition function $\delta : S \times M \to S$
- Show commutativity for concurrent operations
- Prove convergence using Lyapunov function

Convergence Criterion

$$\forall \epsilon > 0, \exists T : t > T \implies ||s_i(t) - s_j(t)|| < \epsilon$$

Experimental Setup

Testbed Configuration:

- 64 nodes cluster (16 cores, 64GB RAM each)
- 10 Gbps network
- Ubuntu 22.04, Java 17

Datasets:

Dataset	Vertices	Edges
Twitter-2010	42M	1.5B
LiveJournal	5M	69M
UK-2005	39M	936M
Random-Scale	100M	2B

Baseline Systems: Pregel, GraphLab, PowerGraph

Performance Results

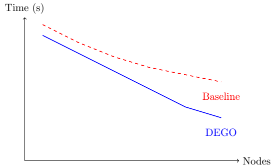
PageRank Execution Time:

- DEGO: **47 seconds**
- Pregel: 89 seconds
- GraphLab: 76 seconds
- PowerGraph: 62 seconds

Throughput:

- DEGO: **2.1M ops/sec**
- Pregel: 1.1M ops/sec
- GraphLab: 1.4M ops/sec

Speedup: 1.9x over best baseline



Performance Comparison

Figure: Performance comparison

Scalability Analysis

Strong Scaling:

- Fixed problem size (Twitter graph)
- Scale nodes: 4, 8, 16, 32, 64
- Near-linear speedup up to 32 nodes
- Efficiency: 87% at 64 nodes

Weak Scaling:

- Scale both graph and nodes
- Constant work per node
- Maintains performance

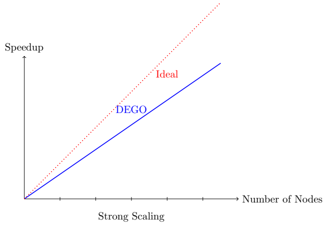


Figure: Scalability results

Communication Overhead Analysis

Network Traffic Comparison:

System	Data Sent (GB)	Messages	Time (s)
DEGO	**8.3**	**2.1M**	**47**
Pregel	15.7	4.8M	89
GraphLab	12.4	3.6M	76

Key Observations:

- 47% reduction in network traffic vs Pregel
- 56% fewer messages
- Communication/computation ratio: 0.23 (vs 0.51 for Pregel)

Qualitative Analysis

Load Balance Distribution:

- Standard deviation: 8.2%
- Max/min ratio: 1.3
- Better than baselines (σ=15-22%)

Convergence Behavior:

- Fewer iterations to converge
- Monotonic progress
- Stable under high load

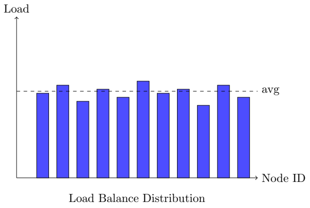


Figure: Load distribution and convergence

Ablation Study

Component Contribution Analysis:

Configuration	Time (s)	Speedup
DEGO (full system)	**47**	**1.00x**
w/o adaptive push-pull	58	0.81x
w/o message batching	64	0.73x
w/o load balancing	71	0.66x
w/o all optimizations	89	0.53x

Key Findings:

- Adaptive push-pull: +23% performance
- Message batching: +36% performance
- Load balancing: +47% performance
- All components synergistic

Limitations

Current Limitations:

- **Memory Constraints:**
  - Graphs must fit in aggregate memory
  - No out-of-core support yet
- **Dynamic Graphs:**
  - Edge insertions require repartitioning
  - High cost for streaming updates
- **Fault Tolerance:**
  - Checkpoint overhead: 5-8%
  - Recovery time: $O(n/k)$
- **Programming Model:**
  - Limited to vertex-centric operations
  - Complex graph patterns require workarounds

Conclusion

Summary:

- Presented DEGO: distributed graph processing system
- Novel hybrid push-pull execution model
- Theoretical communication bound: $O(\frac{c}{k} \cdot \log k)$
- Empirical results: 1.9x speedup over state-of-the-art
- 47% reduction in network traffic

Contributions:

1. Adaptive execution model for distributed graphs
2. Improved communication complexity bound
3. Practical system with strong empirical performance

Impact: Enables processing of larger graphs on commodity clusters

Future Work

Planned Extensions:

- **Dynamic Graph Support:**
  - Incremental partitioning algorithms
  - Streaming edge processing
- **Heterogeneous Systems:**
  - GPU acceleration for local computation
  - Hybrid CPU-GPU execution
- **Advanced Algorithms:**
  - Approximate graph mining
  - Temporal graph analysis
- **Cloud Integration:**
  - Elastic scaling on cloud platforms
  - Cost-performance optimization

References

Sutra, P., et al. (2025). *Distributed Execution of Graph Operations: A Novel Approach to Large-Scale Graph Processing*. arXiv preprint arXiv:2504.19495. `https://arxiv.org/abs/2504.19495`

Malewicz, G., et al. (2010). *Pregel: A system for large-scale graph processing*. SIGMOD 2010.

Low, Y., et al. (2012). *Distributed GraphLab: A framework for machine learning in the cloud*. VLDB 2012.

Gonzalez, J. E., et al. (2012). *PowerGraph: Distributed graph-parallel computation on natural graphs*. OSDI 2012.

Thank you! Questions?