Towards Effective Partition Management for Large Graphs

Shengqi Yang, Xifeng Yan, Bo Zong and Arijit Khan

Computer Science, UC Santa Barbara
{sqyang, xyan, bzong, arijitkhan}@cs.ucsb.edu
Motivation – Challenges

Google: > 1 trillion indexed pages

Web graph

De Bruijn: billions of vertices

Biological graph

Facebook: >800 million active users

Social network

20 billion RDF tuples

Linked data: RDF graphs
Motivation – Existing solutions

- Memory-resident solution
  - Running on single server.
  - Difficult/Impossible to accommodate the content of an extremely large graph.
  - Low concurrency.

- Simple distributed solution (e.g., MapReduce)
  - Running on commodity cluster.
  - High concurrency and enough memory space.
  - Chained MapReduce
    - Communication and serialization overhead.
    - Programming complexity
Pregel (GraphLab, GPS, etc)

- Distribution model: graph partitioning.
- Computation model: run on each partition/vertex simultaneously.
- Communication model: message passing
Pregel: Pros and Cons

Pros
- Designed for iterative jobs
  - Graph algorithms: shortest distance, PageRank, etc.
  - DM/ML tasks: K-Means, belief propagation, Parallel Gibbs sampling, etc.
- Scalable, robust, simple.

Cons
- Graph partitioning: **one partition can not fit all**!
- Query on graph
  - Unbalanced workload
  - Inter-machine communication
Graph query

- Graph query pattern
  - (a) random/complete
  - (b) internal
  - (c) cross-partition

- Unbalanced workload
- Inter-machine communication
Objectives

- Graph query processing
  - Solving the problems facing graph partitioning (Pregel)
    - Workload balancing (replication)
    - Communication reduction
  - Graph partition management strategy
    - Evolving query workload.
- Sedge: a Self Evolving Distributed Graph Processing Environment
Outline

- Partitioning techniques
  - Complementary partitioning
  - On demand partitioning
  - Two-level partition management
- System architecture
- Experiments
- Conclusions
Complementary partitioning: repartition the graph with region constraint.

These two sets of partitions will run independently.
Complementary partitioning

- Iteratively repartition the graph
- Pros
  - Effective communication reduction
  - Workload balancing
- Cons
  - Space limitation
  - Can not adapt to dynamic workload

5/25/2012 UCSB
On demand partitioning

- Blocks
  - **Goal**: coarsen a graph
  - **Method**: disjoint 1-hop region

- Query profiling
  - **Blocking**: use blocks to track cross-partition queries.
  - **Advantages**:
    - Query generalization.
    - Profiling with fewer features.
On demand partitioning

- **Envelope**: a set of blocks that covers a cross partition query.
- **Envelope Collection**: put the maximized number of envelopes into a new partition wrt. space constraint.
Intention: combine similar envelopes sharing many common blocks.

Algorithm:
1) **Similarity search** (nearest neighbor search).
   *Locality Sensitive Hashing (LSH): Min-Hash*, in $O(n)$
2) **Envelope combining**
   1. Cluster the envelopes in the same bucket produced by Min-Hash.
   2. Combine the clusters with highest benefit.
Two-level partition architecture

- Primary partitions. e.g., A, B, C and D. They are interconnected in *two-way*
- Secondary partitions. e.g., B’ and E. They are connected with primary partitions in *one-way*
Sedge: System Architecture
SP²Bench (Schmidt et. al., ICDE ’09)

- Employ the DBLP library as its simulation basis.
- 100M tuples (11.24GB): 6 partitions.
- Query templates:
  - 5 of 12: Q2, Q4, Q6, Q7, Q8.
  - E.g. Q4: Given a journal (J), select all distinct pairs of article author names for authors (?A) that have published papers (?P) in the journal.

Cluster environment

- 31 nodes (connected by a gigabit Ethernet).
  - 4 GB RAM and quad-core 2.60GHz Xeon Processors
  - 1 master, 30 workers.
Effect: complementary partitioning

Query workload: $10^4$ queries with random start

Log scale!
Evolving query workload

- 5 Static replication
- 5 Comp. part.
- 4 Comp. part. + On demand part.
Datasets
- UK web graph: 30M vertices, 956M edges.
- Twitter: 15M users, 57M edges.
- Bio graph: 50M vertices, 51M edges.
- Synthetic graph: 0.5B vertices, 2 Billion edges (by R-MAT, Chakrabarti et. al., *SDM’04*).

Query workload
- Mixture of general graph queries
  - neighbor search (2~3 hops).
  - random walk (5~10 steps).
  - random walk with restart (5~10 steps, 10% restart).
Scalability
Cross-partition queries vs. Avg. response time

Improvement in avg. response time

% of cross-partition queries in the workload

Web
Twitter
Bio
Syn.
Conclusions

- Partitioning techniques
  - Complementary partitioning
  - On demand partitioning
- Two-level partition management

Project home: [http://grafia.cs.ucsb.edu/sedge](http://grafia.cs.ucsb.edu/sedge)
- Documents and sample datasets
- Source code
Thank You!