PART B. GEAR WORKSHOP III
LARGE SCALE GRAPH ANALYSIS

Today's topics

- Introduction to the scalable Graph Analysis
- Pregel: "Think like a vertex!"

This material is built based on


GRAPH?

- Graph algorithms are becoming increasingly important for solving many problems
- Graphs provide a flexible abstraction for describing relationship between discrete objects
  - A Graph consists of a finite set of vertices and a set of edges
    - Adjacency Matrix
    - Adjacency List

http://www.cs.colostate.edu/~cs535
Graph analysis applications in Biology

- Protein-protein interaction networks
  - E.g. LCMA (Local Clique Merging Algorithm) iteratively identifies local cliques and merges them if they overlap significantly
- Gene regulatory networks
  - E.g. Deriving a gene regulatory network from observed gene expression levels
- Metabolic networks
  - E.g. Identifying common patterns in graph-based representations of metabolic pathways
- Tissue modeling
  - E.g. Classifying tissue samples by segmenting tissue images to identify cells

Graph analysis applications in Network Security

- Network vulnerability analysis
  - E.g. Specifying the pre- and post-conditions of attacks using attack graphs
  - E.g. Defining a suite of metrics to qualify the vulnerability of a network based on its attack graph
- Malware detection
  - E.g. Analyzing download graphs to identify droppers, malicious programs that download other programs to a host machine
  - E.g. Estimating the probability of a file being malicious using bipartite graph of machine and files
- Botnet detection
  - E.g. Identifying a P2P botnet in a communication graph
- Anomaly detection
- Video Surveillance

Graph analysis applications in Software Engineering

- Identifying and locating software bugs
  - E.g. Mining software behavior graphs
- Detect prediction
  - E.g. Mining software dependency graph
- Software plagiarism detection
  - E.g. GPLAG: mining the software dependency graphs

Graph analysis applications in Social Science

- Identifying important users
  - E.g. Modeling Twitter users based on the Authority scores using PageRank
- Identifying interesting posts
  - E.g. Using the HITS algorithm
  - Science, Finance, Linguistics, etc.

Challenges in the Large-scale graph processing

- Voluminous graph will be partitioned across the cluster
- Ingression time
  - Time that the system to load and partition the graph before starting the actual execution
- Communication cost
  - The amount of network transfer required during the executing the algorithm
- Load balancing
  - Quality of load distribution
- Balanced graph partitioning is known as an NP-complete problem

Graph partitioning: Edge-cut

- Vertices are equally distributed among partitions
- Then, edges are distributed across partitions
  - Edges along with the corresponding vertices are replicated and passed according to the requirement between partitions
  - Communication cost associated with edge-cut algorithms is directly proportional to the number of edges cut
- Both edge data and vertex data are passed to between partitions

http://www.cs.colostate.edu/~cs535
Graph partitioning: Edge-cut: Example
- Both edge data and vertex data are passed to between partitions

Example
Vertices are partitioned with a hash function:
(vertex id) % n
Where, n is the total number of partitions
Suppose that n is 2. Vertices 1, 3, and 5 will be grouped in the same partition. 2 and 4 will be grouped separately.

If both source and destination of edge are in the same partition, that edge will be grouped in the same partition. Otherwise, it will follow the destination.

Graph partitioning: Vertex-cut
- Edges are equally distributed
- Then, vertices are cut and replicated across the partitions
  - Vertex data is passed between the partitions
  - Communication cost is directly proportional to the number of the vertex replicas
  - Load balancing factor is determined by the number of edges assigned to each of the partitions
- Passes just vertex data

Types of Graph Process Models
- Vertex-centric (Edge-cut)
  - Synchronous: Giraph, GraphLab, BlogelVertex
  - Asynchronous: GraphLab
  - These systems cannot handle the Power Law like uneven distribution
- Edge-centric (Vertex-cut)
  - PowerGraph
  - GraphX
- Block-centric: Blogel-Block (Blogel-B)
- MapReduce and its extensions: Hadoop
- Relational: Vertica
- Streaming Graph: Flink, Gelly

Graph analysis at Google?
- MapReduce tasks
  - Google’s 80% of data analysis
  - Large-scale web search indexing
  - Clustering problems for Google News
  - Producing reports for popular queries (e.g., Google Trend)
  - Processing of satellite imagery data
  - Language model processing for statistical machine translations
  - Large-scale machine learning problems
  - Back-up/restore

- The other 20%?
Graph analysis at Google?

- MapReduce tasks
  - Google's 80% of data analysis
  - Large-scale web search indexing
  - Clustering problems for Google News
  - Produce reports for popular queries (e.g., Google Trend)
  - Processing of satellite imagery data
  - Language model processing for statistical machine translations
  - Large-scale machine learning problems
  - Back-up/restore
- The other 20%? Graph Analysis

Graph analysis at Google?

- Large graph analysis
  - Graph algorithms
    - PageRank
    - Shortest path
    - Connected components
    - Clustering techniques
  - Graph data
    - Web graph
    - Transportation routes
    - Citation relationships
    - Social networks

What is so unique about "processing large graphs"?

- Poor locality of memory access for graph algorithms
- Very little work per vertex
- A changing degree of parallelism over the course of execution

MapReduce is NOT great for graph processing

- Many iterations are needed for parallel graph processing
- Materializations of intermediate results at every MapReduce iteration causes performance bottleneck

Single Source Shortest Path (SSSP)

- Find shortest path from a source node to all target nodes
- If you have a single processor machine?
  - Dijkstra's algorithm

GEAR Workshop II | Large Scale Graph Analysis
Pregel: "Think Like A Vertex!"
Understanding Graph Process in the Cluster

http://www.cs.colostate.edu/~cs535
GEAR Workshop II | Large Scale Graph Analysis
Pregel: “Think Like Vertex!”
Understanding Graph Process in the Cluster
Finding SSSP using Dijkstra’s Algorithm
with a Single Node
Using MapReduce

Adjacency Matrix

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>10</td>
<td>2</td>
<td>1</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>3</td>
<td>9</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Adjacency list:
- A: [(B, 10), (C, 5)]
- B: [(C, 2), (D, 1)]
- C: [(B, 3), (D, 9)]
- D: [(B, 4), (C, 5)]
- E: [(A, 7)]

Map output: <E, 1>, <D, 11>, <C, 5>, <B, 10>, <A, 7>, (B, 10), (C, 5)>

= Map input for next iteration

Reduce input: <E, 1>, <D, 11>, <C, 5>, <B, 10>, <A, 7>, (B, 10), (C, 5)>

Reduce input

Map input: <nodeID, <dist, adj list>>
- A: (A, 7), (D, 6)
- B: (E, 4)
- C: (B, 3), (D, 9)
- D: (E, 4)
- E: (A, 7), (D, 6)

Reducer: Find the minimum distance <nodeID, dist>
- A: (A, 7), (D, 6)
- B: (E, 4)
- C: (B, 3), (D, 9)
- D: (E, 4)
- E: (A, 7), (D, 6)

Flush to local FS

http://www.cs.colostate.edu/~cs535
Using MapReduce

Reduce input <nodeID, <dist, adj list>>
- Map input for next iteration
  - A, 0, <(B, 10), (C, 5)>
  - B, 8, <(C, 2), (D, 9)>
  - C, 5, <(B, 3), (D, 9)>
  - D, 11, <(E, 4)>
  - E, 7, <(A, 7), (D, 6)>

Keep going..

GEAR Workshop II | Large Scale Graph Analysis
Pregel: “Think Like Vertex!”
Understanding Graph Process in the Cluster
Finding SSSP using Dijkstra’s Algorithm
using Pregel

Bulk Synchronized Parallel Model

Input
Superstep: A Sequence of Iterations
Output

Inspired by Valiant’s Bulk Synchronous Parallel model (1990)

Computation Model

- Data partitioning: Edge-cut
  - Vertices are distributed evenly
- Superstep: the vertices compute in parallel
  - Each vertex
    - Receives messages from the previous superstep
    - Executes the same user-defined function
    - Sends messages to other vertices
    - Maintains the topology of the graph if need be
    - Votes to halt if it has no further work to do
- When to terminate?
  - All vertices are simultaneously inactive
  - Voting to halt
  - There are no messages in transit

Computation Model

- Input to the Pregel computation
  - A directed graph
    - Vertex
      - String vertex ID
      - Associated user-defined value
    - Edge
      - Associated with the source vertex
      - User-defined value and a target vertex ID

http://www.cs.colostate.edu/~cs535
**Computation Model (3/3)**

- Computation in the vertex
  - Executes the same user-defined function
  - Modifies the state
  - Sometimes changes the outgoing edges
  - Receives/send message
  - Mutate topology
  - There is no computation associated with the edges

---

**Vertex State Machine**

- States
  - Active
  - Inactive
- Transition to the "active" state
  - "Any" message is received
- Transition to the "inactive" state
  - The process is voted to "halt"

---

**Output of a Pregel program**

- Set of values explicitly output by the vertices
  - Often a directed graph isomorphic to the input
  - E.g. clustering algorithm
  - Creates small set of disconnected vertices selected from a large graph
  - E.g. graph mining algorithm
  - Generates aggregated statistics mined from the graph

---

**Message Passing (1/2)**

- Vertices communicate directly with one another by sending messages
  - Message value
  - Name of the destination vertex

  - A vertex can send any number of messages in a superstep

  - There is no guaranteed order of messages in the iterator.
    - However,
      - Message is delivered reliably
      - There will be no duplicate

---

**Message Passing (2/2)**

- Common usage pattern
  - A vertex \( V \) to iterate over its outgoing edges and sending a message to the destination vertex of each edge

  - Destination vertex need not be a neighbor of \( V \)
    - E.g.: A vertex can learn the identifier of a non-neighbor from a message received earlier
    - E.g.: implicitly vertex info is distributed

  - If destination does not exist, user-defined handler will be executed.
    - Create the missing vertex or remove the dangling edge

---

http://www.cs.colostate.edu/~cs535

Spring 2019 Colorado State University, page 8
SSSP using parallel BFS in Pregel

If you did not update your value, inactive the state.

If you did not update your value, inactive the state.

If you did not update your value, inactive the state.

If you did not update your value, inactive the state.

http://www.cs.colostate.edu/~cs535
SSSP using parallel BFS in Pregel

SSSP using parallel BFS in Pregel

SSSP using parallel BFS in Pregel

http://www.cs.colostate.edu/~cs535  Spring 2019 Colorado State University, page 10