GraphLab and its distributed versions

Mu Li

Adopted slides from Joseph and Yuchen
The **GraphLab Abstraction**

- A user-defined **Vertex Program** runs on each vertex
- **Graph** constrains *interaction* along edges
  - Directly **read** and **modify** the state of adjacent vertices and edges
- **Parallelism**: run multiple vertex programs simultaneously
GAS Decomposition

**Gather (Reduce)**
Accumulate information about neighborhood

*User Defined:*
- \( \text{Gather}(Y, \cdots) \rightarrow \Sigma \)
- \( \Sigma_1 + \Sigma_2 \rightarrow \Sigma_3 \)

**Apply**
Apply the accumulated value to center vertex

*User Defined:*
- \( \text{Apply}(Y, \Sigma) \rightarrow Y' \)

**Scatter**
Update adjacent edges and vertices.

*User Defined:*
- \( \text{Scatter}(Y') \rightarrow \)

Parallel Sum
\( \cdots + \cdots \rightarrow \Sigma \)
GraphLab is **Asynchronous**

The **scheduler** determines the order that vertices are executed.

Scheduler can **prioritize** vertices.
GraphLab is **Serializable**

- Automatically ensures **serializable** executions
The GraphLab Framework

Graph Based
Data Representation

Update Functions
User Computation

Consistency Model
Distributed Graph

Partition the graph across multiple machines.
Distributed Graph

- Ghost vertices maintain adjacency structure and replicate remote data.
Ensuring Race-Free Code
How much can computation overlap?
Edge Consistency via Graph Coloring

Vertices of the same color are all at least one vertex apart. Therefore, All vertices of the same color can be run in parallel!
Distributed Locking

Edge Consistency can be guaranteed through locking.

: RW Lock
Challenges of High-Degree Vertices

- Asynchronous Execution requires heavy locking (GraphLab)
- Sends many messages (Pregel)
- Touches a large fraction of graph (GraphLab)
- Edge meta-data too large for single machine
- Sequentially process edges
- Synchronous Execution prone to stragglers (Pregel)
• Split High-Degree vertices

• New Abstraction $\rightarrow$ **Equivalence** on Split Vertices
Distributed Execution of a PowerGraph Vertex-Program

Gather
Apply
Scatter
Things didn’t covered

• Checkpoint
• Graph partition
• Experiments
  – There is no standard benchmark
  – Do not take serious about
    • It is highly related to data, workload, and system
    • comparison with map/reduce
    • Scalability