Large scale graph processing systems: survey and an experimental evaluation

Cluster Computing 2015

Omar Batarfi, Radwa El Shawi, Ayman G. Fayoumi ,
Reza Nouri, Seyed-Mehdi-Reza Beheshti, Ahmed Barnawi, Sherif Sakr

Graph scale

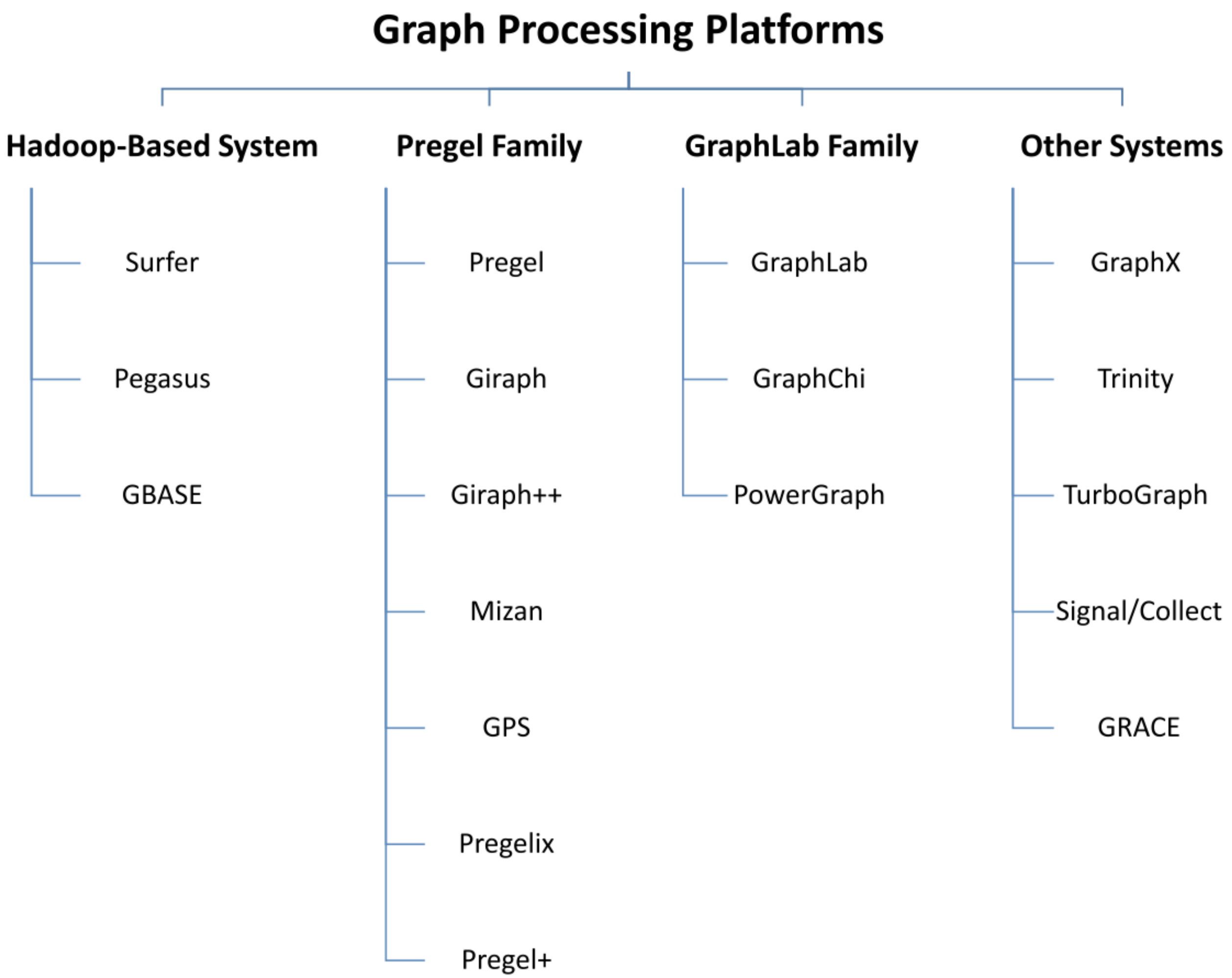
- Scale: millions to billions of nodes and edges
  - Facebook social network graph:
    - 1 billion+ users (nodes)
    - 140 billion+ friendship relationships (edges)
- The size of a single node/edge can be very different due to the various attributes size or nodes/edges
- **An Estimation of large-scale graph size: 1 GB to 500 TB**
  - ~0.5 GB/million nodes, ~0.20 GB/million edges
  - 1 million nodes/edges: ~0.5 GB nodes & ~0.20 GB edges
  - 500 billion nodes/edges: ~250 TB nodes & ~100 TB edges

Table 1 Characteristics of the used graph datasets

Dataset name	Number of nodes	Number of edges	Size on disk	GB/million nodes	GB/million edges
Wikitalk	2,394,385	5,021,410	1 GB	0.42	0.20
Amazon	21,365,698	140,015,189	18 GB	0.84	0.13
Citation	3,774,768	16,518,948	43 GB	11.39	2.60
Friendster	65,608,366	1,806,067,135	120 GB	1.83	0.07
LUBM 30K	12×10^8	3×10^9	700 GB	0.58	0.23
LUBM 40K	2×10^9	5×10^9	950 GB	0.48	0.19
LUBM 50K	28×10^8	7×10^9	1.2 TB	0.44	0.18

platforms

- General-purpose platforms (such as MapReduce) are bad
  - No direct support for iterative graph algorithms
  - To detect a fix point (termination condition), an extra task might be required on each iteration
- **Specialized platforms**
  - *Pregel* family
  - *Graphlab* family
  - Others



Algorithms

- Characteristics of graph algorithms
  - **Iterative**
  - **Need to traverse the graph**
- Typical graph algorithms
  - **PageRank**: rank nodes based on their incoming/outgoing edges
  - **Shortest Path**: find the path between 2 nodes where the sum of weights is minimal
  - **Pattern Matching**: find certain structures (e.g. path, star)
  - **Triangle Count**: counts the number of triangles
  - **Connected Component**: find the subgraphs in which any two vertices are connected

Pregel family

- Pregel:
  - Google's pioneer work in this area
  - Published in 2010
- Distributed & computations are **totally in memory**
- Iteration -> superstep
- Address scalability issue
  - **Bulk Synchronous Parallel (BSP**): synchronization barrier on each superstep
  - **Message passing interface (MPI)**
- Vertex-centric approach
  - **Locality**: Each vertex & its neighbors are in the same node
  - A vertex can: execute a function/send messages to others/change states (active/inactive)
  - Termination: no active vertices & no messages being transmitted
- Pregel family
  - Apache Giraph: Java implementation of pregel
  - GPS: another Java implementation
  - Pregelix: set-oriented, iterative dataflow

GraphLab family

- GraphLab
  - Shared memory
- GAS (Gather, Apply, Scatter) processing model
  - Gather: a vertex collects info of its neighbors
  - Apply: performs computation
  - Scatter: update adjacent vertices and edges
- Comparison
  - GAS : pull-based; a vertex request info of all neighbors
  - MPI: push-base; a vertex receives info from neighbors
- Two modes:
  - Synchronous model (BSP): communication barriers
  - Asynchronous model: using distributed locking; no communication barriers orsuperstep
- GraphLab family
  - PowerGraph: avoid the imbalanced workload caused by high degree vertices in power-law graphs
  - Trinity: memory-based; distributed
  - Signal/Collect: vertex-centric; two operations for a vertex (signal/collect)
  - **Graphchi**

GraphLab family cont'd

- **Graphchi**:
    - Out-of-core: using secondary storage in a single machine
    - Parallel Sliding Window (PSW):
        - Goal: **decreases non-sequential accesses** on disk
        - It partitions the graph into shards
        - In each shard, edges are sorted by the source IDs
    - Selective scheduling:
        - Converge faster on some parts of the graph
        - "some parts" -> the change on values is significant
    - Pros
        - It avoids the challenge of finding efficient graph cuts
            - Now with zone-based devices, partitioning is needed again
        - It avoids cluster management, fault tolerance etc.

    - **Out-of-Core + SMR**

Other systems

- **TurboGraph**
  - Out-of-core
  - Processing billion-scale graphs using modern hardware -> **parallelism**
    - Multicore CPU: multiple job at the same time
    - **FlashSSD: multiple I/O requests in parallel using multiple flash memory packages**
  - A parallel model called *pin-and-slide*: column view of the matrix-vector multiplication
  - Two types of thread pools
    - Execution thread pool
    - Asynchronous I/O callback thread pool
  - Steps
    - Restrict computation to a set of vertice -> identify the corresponding pages
    - -> pin those pages in the buffer pool
    - -> processing completes for a page -> swtich unpinned -> can be evicted now
    - -> Parallel asynchronous I/O request to the FlashSSD for pages which are not in the buffer pool
  - The system can slide the processing window one page at a time

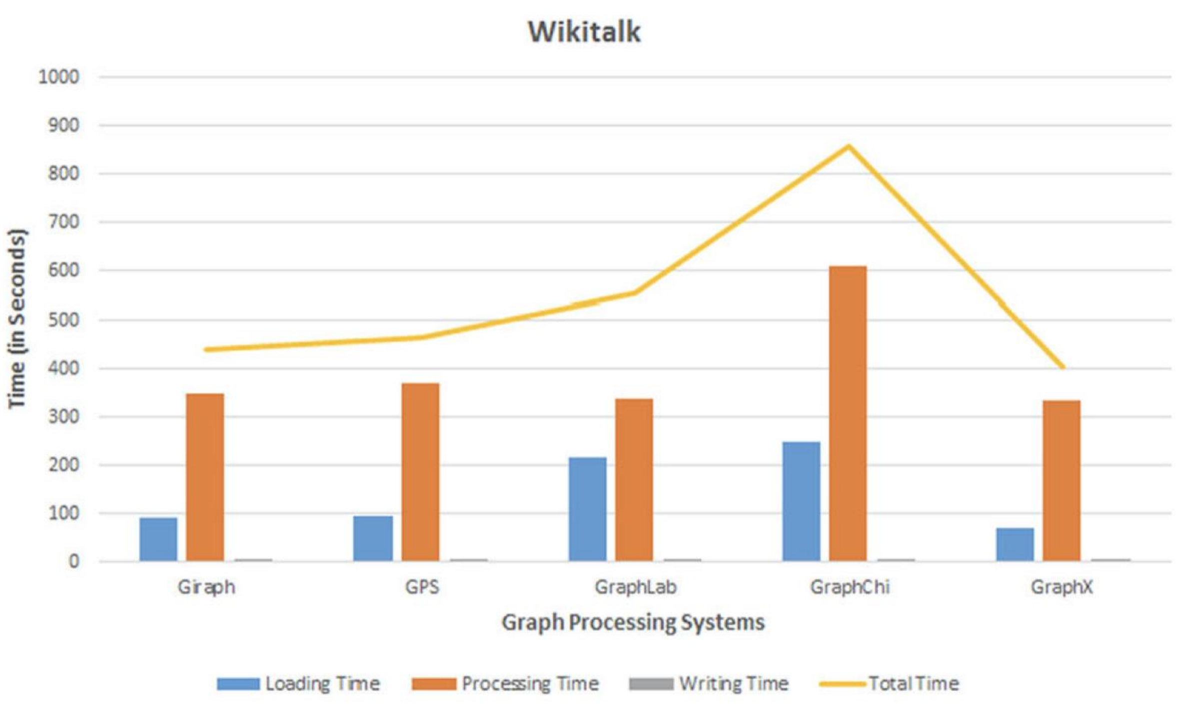
  - **Multiple channel SSD**
  - **Extreme-large-scale graph that does fit into memory**
  - **CMR -> SMR/Zone named SSD**

Other systems

- **GRACE**
  - Out-of-core
  - Batch-style graph programming frameworks
  - Providing a high level representation for graph data
    - Separating application logic from execution policies.
  - Combine synchronous programming with asynchronous execution

Experiments

- **Perforamance metrics**
  - **Reading Time**, Processing Time, Writing Time, Total Execution Time, CPU Utilization, RAM Usage, Network Traffic
- Deployed on Amazon AWS cloud services



(a) Wikitalk Dataset.

The execution times metrics for the PrageRank algorithm for all systems using the different datasets