Ning LIU, Dong-sheng LI, Yi-ming ZHANG, Xiong-lve LI, 2020. Large-scale graph processing systems: a survey. *Frontiers of Information Technology & Electronic Engineering*, 21(3):384-404. https://doi.org/10.1631/FITEE.1900127

Large-scale graph processing systems: a survey

Key words: Graph workloads; Graph applications; Graph processing systems

Corresponding author: Dong-sheng LI
E-mail: dsli@nudt.edu.cn
ORCID: https://orcid.org/0000-0001-9743-2304

Motivation

- Graph is a significant data structure that describes the relationship between entries.
- Graph applications are vastly different from traditional applications. It is inefficient to use general-purpose platforms for graph applications, thus contributing to the research of specific graph processing platforms.
- There is no complete analysis about comprehensive graph processing systems including the frameworks and implementation techniques rather than just one specific branch.
- There is no systematical categorization of graph workloads and applications.

Contents

- **Workloads and applications**
  - Traversal- and dense-computation-based workloads
  - Graph database, graph mining, and graph machine learning
- **Existing solutions**
  - General-purpose systems
  - Specialized systems: single-machine and distributed systems
- **Implementation techniques**
  - Programming model, partitioning strategy, communication model, execution model, and fault tolerance strategy
- **Recent advances and open challenges**

Workloads

- **Traversal-based workloads**: visit nodes in certain order to check or update information
- **Dense-computation-based workloads**: various computations are executed on several or all vertices with adjacent edges in each iteration

Table 1 Categories of graph workloads

Type	Algorithm(s)	Algorithm description	Service	Application
Traversal-based	BFS and DFS	Search	Link analysis	Transportation
	Dijkstra, Bellman Ford, SPFA, and APSP	Shortest path	Distance/path	Transportation
	Prim, MST, and Kruskal	Minimum spanning tree	Smooth traffic	Traffic project
Dense-computation-based	PageRank	Rank web page	Link analysis	Social network
	Connected component	Maximal connected subgraphs	Community detection	Social network
	Triangle counting	Counting triangles	Link analysis	Social network

BFS: breadth-first search; DFS: depth-first search; APSP: all pairs shortest path; MST: minimum spanning tree

Applications

- **Graph database** is a database that uses graph structures for semantic queries with vertices, edges, and properties to present and store data.
- **Efficient graph mining** algorithms contribute to inferring useful knowledge from structural graph data.
- **Graph machine learning**

Table 2 Categories of graph applications

Type	Research branch(es)	Algorithms or typical models
Graph database	Graph pattern matching	Neo4j, Oracle PGX, SAP HANA Graph, and Redis
Graph mining	Frequent subgraph mining	AGM, SPIN, gSpan, and CloseGraph
	Community detecting	FFSM, FSG, GREW, and CPM
	Influence maximization	CPMw, IS, FCM, EAGLE, IC, and LT
Graph machine learning	GNN	ChebNet and Neural FPs
	GCN	PATCHY-SAN and DCNN
	GAE	Neural FPs
	Graph autoencoder	DGCN and ARGA

GNN: graph neural network; GCN: graph convolutional network; GAE: generalized advantage estimator; CPM: clique percolation method; CPMw: CPM with weights; IS: iterative scan; EAGLE: agglomerative hierarchical clustering based on maximal clique; IC: independent cascade; LT: linear threshold; ARGA: adversarially regularized graph autoencoder

Existing solutions

1. General-purpose systems

General-purpose systems are usually general processing platforms used for data analysis on a mass scale. There are several disadvantages about systems modified from the MapReduce platform:

- Warm-up overhead
- Static data operating overhead
- Data storage inefficiency
- Shuffle overhead
- Poor programming interface

Existing solutions

2. Specialized systems

- **Single-machine shared-memory systems**

Single-node shared-memory systems with multiple cores support more than a terabyte of memory, which can fit graphs with tens or even hundreds of billions of edges.

- **Single-machine out-of-core systems**

With the sharp increase of graph data, the storage hierarchy of single-machine memory-shared systems is extended from random access memory (RAM) to external memory.

Existing solutions

2. Specialized systems

- **Distributed shared-memory systems**

A distributed system is composed of multiple processing nodes and each node has its own memory.

- **Distributed out-of-core systems**

Single-machine out-of-core systems can be expanded to distributed out-of-core systems for scalability.

Implementation techniques

1. Programming model

- **Vertex-centric model**

The basic computation unit is a vertex. User-defined programs are executed on vertices in each iteration, and the intermediate information is passing to the adjacent vertices along with edges for the next iteration.

- **Edge-/path-centric model**

Random access to any storage medium delivers less bandwidth than sequential access. The edge-/path-centric model reduces random access to out-edges during message passing.

- **Graph-centric model**

The graph-centric model partitions a graph into coarser-grained subgraph units, which are called blocks.

Implementation techniques

2. Partitioning strategy

- **Edge cut**

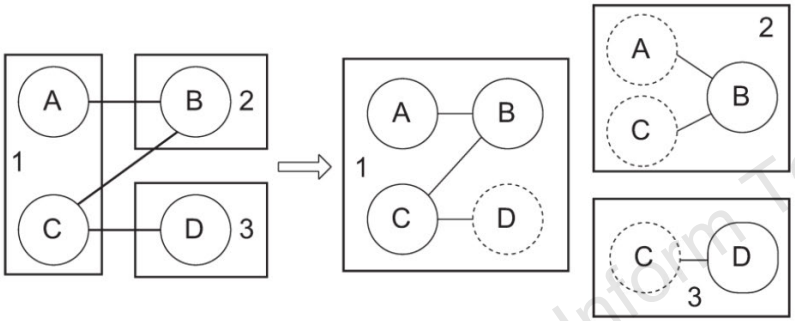

Fig. 2 An edge-cut example of a graph into three parts, where solid and dashed vertices are ghosts and mirrors, respectively (Gonzalez et al., 2012)

- **Vertex cut**

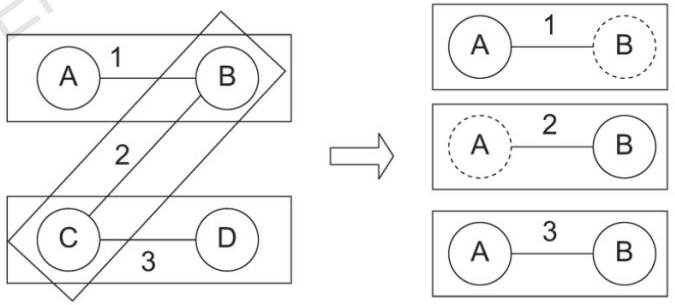

Fig. 3 A vertex-cut example of a graph into three parts, where solid and dashed vertices are ghosts and mirrors, respectively (Gonzalez et al., 2012)

Implementation techniques

2. Partitioning strategy

- **Parallel sliding windows cut**

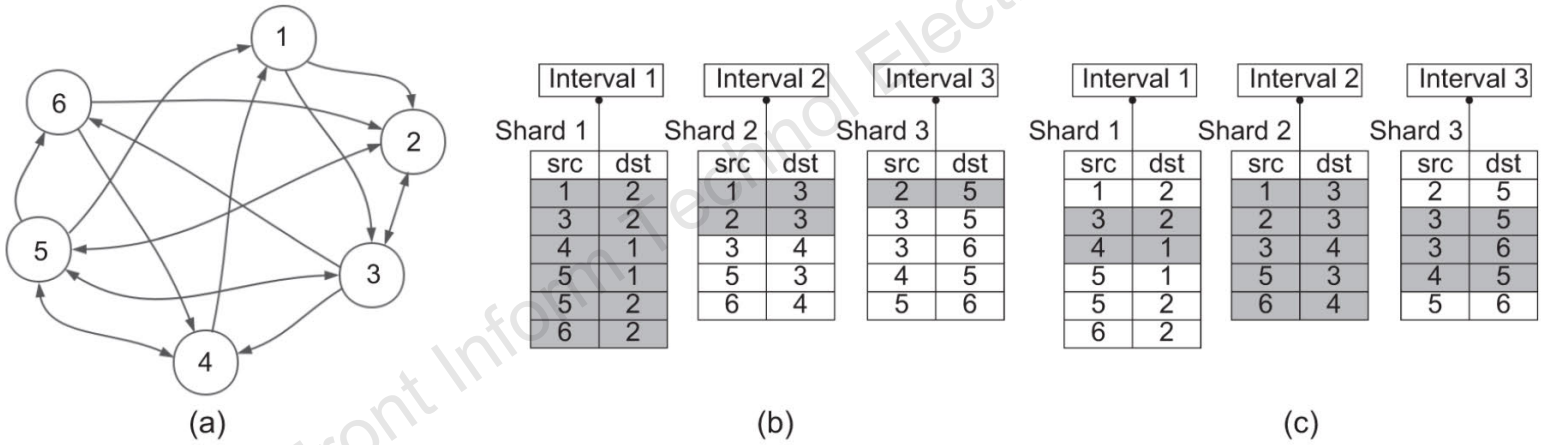

Fig. 4 Operation of PSW-cut on a toy graph: (a) the example graph with six vertices is partitioned into three equal intervals according to the destination; (b) PSW begins by executing interval 1 and the shaded parts are loaded into memory; (c) PSW moves to the next interval. Reprinted from Kyrola et al. (2012), Copyright 2012, with permission from the authors

Implementation techniques

2. Partitioning strategy

- **Grid cut**

Grid cut equally partitions vertices into P one-dimensional chunks based on the range. Edges are partitioned into a $P \times P$ two-dimensional block grid.

Implementation techniques

3. Communication model

- **Push style**

The push style communication model is source-vertex centric. The graph program traverses source vertices and executes user-defined update functions. Then messages will be sent to the neighbors through the out-edges. In other words, a vertex can learn its neighbors' values via only the messages that its neighbors push to it.

- **Pull style**

The pull style communication model is destination-vertex centric. The graph program traverses destination vertices and asks related messages from source vertices. The user-defined update function will be executed when all related messages have been pulled.

Implementation techniques

4. Execution model

- **Synchronous execution**

The synchronous execution of graph systems is often based on the bulk synchronous parallel (BSP) model.

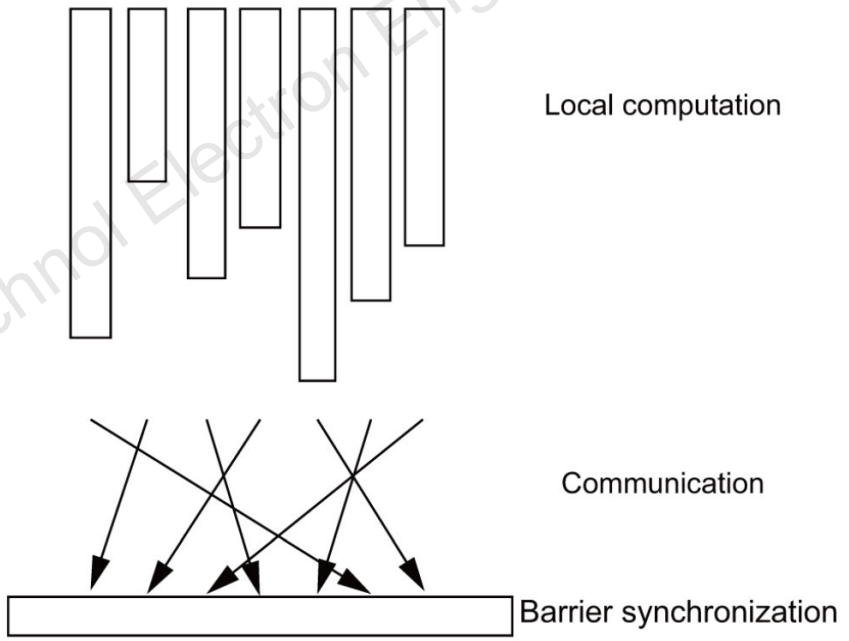

Fig. 6 Illustration of the processing of the bulk synchronous parallel model

Implementation techniques

4. Execution model

- **Asynchronous execution**

In the asynchronous model, the destination vertex can immediately execute the update function without waiting until all the other vertices have received messages. The asynchronous execution model works well to eliminate the bucket effect and reduce the synchronous overhead, but it increases the difficulty of programming and debugging.

Implementation techniques

5. False tolerance strategy

- **Checkpointing**

  Pregel [SIGMOD'10], Hama [CloudCom'11]

- **Asynchronous checkpointing**

  GraphLab [PVLDB'12]

- **Replication-based fault-tolerance mechanism**

  Imitator [TPDS'17]

- **Partition-based fault-tolerance mechanism**

  [PVLDB'14]

Recent advances and open challenges

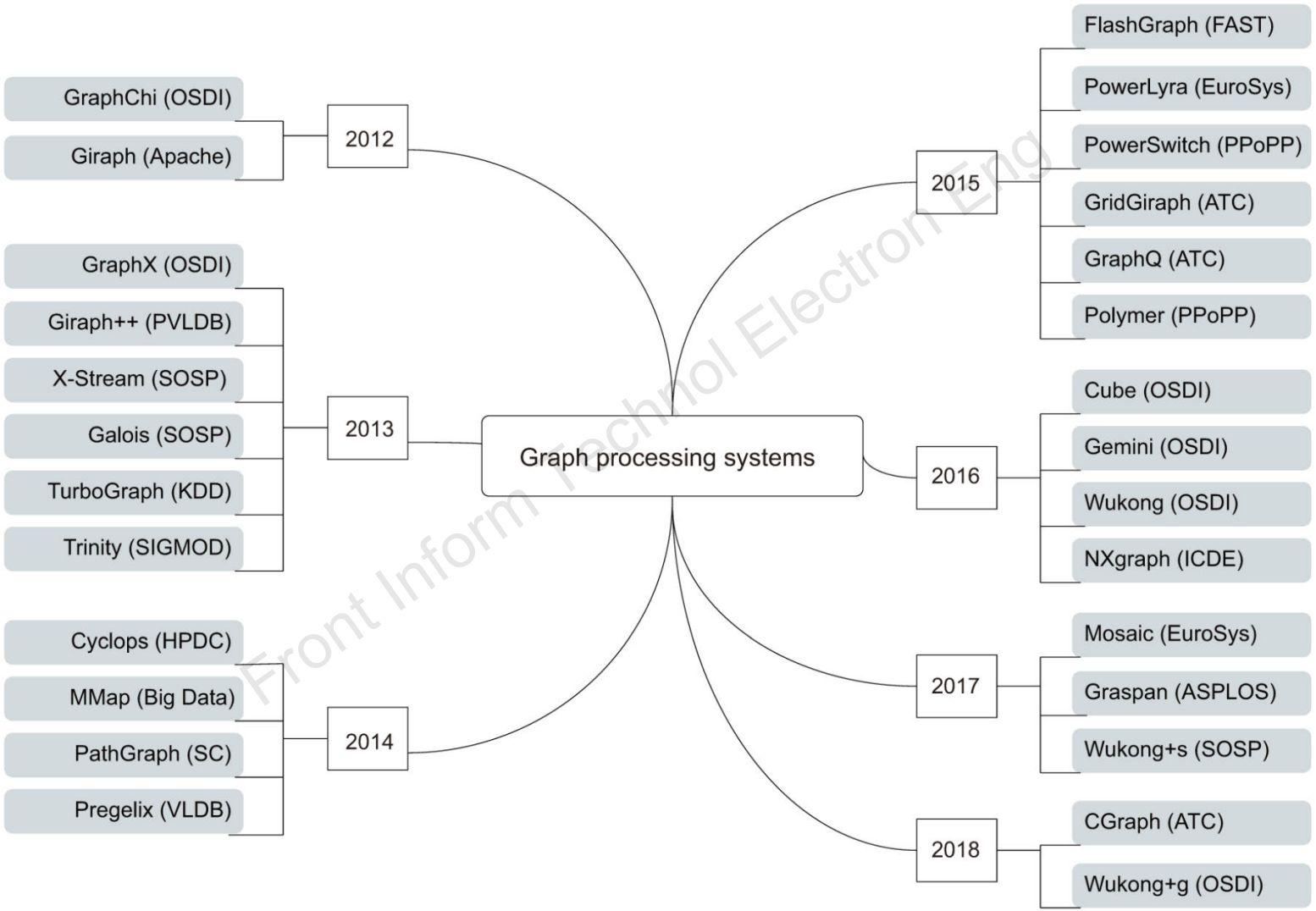

Fig. 7 Timeline representation of graph processing systems

Recent advances and open challenges

- **Massive graphs**
- **Dynamic graphs**
- **Remote direct memory access (RDMA)** can reduce communication overhead in distributed graph processing systems.
- **Emerging storages,** such as solid state drive (SSD), non-volatile memory (NVM), and DDR4, provide opportunity for improving the scalability of single-machine graph processing systems.
- **Graph deep learning**