GraphScope Flex: A Graph Computing Stack with LEGO-Like Modularity

GraphScope Team
graphscope@alibaba-inc.com

Extended Abstract

Real-world graph applications can exhibit significant variation in many aspects. Diverse graph workloads, such as graph analytics, graph traversal queries, graph pattern matching, and graph neural networks, may be involved, each of which can be tackled using different programming interfaces, such as Pregel[5], PIE[1], and FLASH[7] for graph analytics, or Cypher and Gremlin for graph queries. Furthermore, these applications may have different deployment modes, including an offline analytical pipeline that prioritizes low running time of a complex query, an online service that demands high query throughput, or a learning task that can leverage heterogeneous hardware resources, such as CPUs and GPUs. Moreover, the graph data can be stored in various formats, considering factors such as persistency, mutability, partitioning, and transactional guarantees. To further complicate the situation, these aspects can interact with each other in complex ways. For example, a fraud-detection scenario at Alibaba [2] may require a combination of graph analytics, traversal queries, and neural networks, working on an immutable distributed in-memory graph in an offline pipeline. On the other hand, an online interactive graph query system may require support for traversal queries and pattern matching on a mutable, persistent graph.

To address such diversities and complexities, we have developed GraphScope Flex, a graph computing stack with LEGO-like modularity, as shown in Figure 1. This stack comprises multiple components, which users can combine like LEGO bricks to create customized deployments that meet their specific graph computing needs. For example, in Figure 1, users can deploy systems for online graph business intelligence, offline analytical graph computation, and training GNN models, using orange, yellow, and green bricks, respectively. Furthermore, the blue frame in Figure 1 encloses the deployment of GraphScope Flex for solving a complex graph pipeline, such as the aforementioned fraud detection.

GraphScope Flex deployments are flexible yet highly performant. For instance, it has outperformed the other systems from 2× to magnitudes in both LDBC [3] SNB and Graphalytics benchmarks[1]. We highlight the techniques that enable this flexibility and efficiency below.

- The unified storage interface provides a layer of highly efficient graph access interface, which decouples different graph formats and media from computing engines. It handles the *read*-paths uniformly to a variety of graph data stores, with support for a number of indices and other "push-downs" to leverage the optimizations provided by the graph stores. Additionally, new types of graph stores can be easily integrated into GraphScope Flex.
- A Graph IR (intermediate representation) layer provides a language-agnostic representation that can be translated from graph queries written in Cypher and Gremlin. It extends relational algebra to include a set of graph-specific primitives such as getting neighbors and paths. Under the IR layer, a universal optimizer is developed. After optimization, the IR can guide code generation for either the Hiactor [4] for online high-QPS service, or Gaia [6] for data-parallel execution.
- An analytical runtime based on GRAPE [1], which uses a fragment-centric and extensible architecture to support multiple programming models efficiently.
- A GNN framework based on graph-learn [8] that can efficiently support distributed GNN training on industrial-scale graphs. It supports both Tensorflow and PyTorch as the training backend, and decouples sampling and training such that each part can be scaled independently.

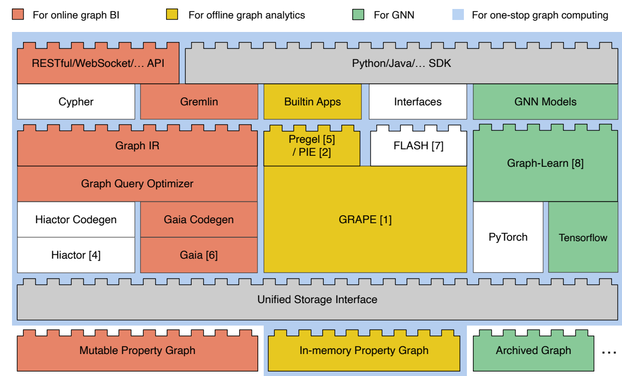

Figure 1. A LEGO view of GraphScope.

[1] https://github.com/alibaba/GraphScope/blob/Flex/Performance.md

References

[1] Fan, W., et al. Parallelizing sequential graph computations. In *SIGMOD 2017*.

[2] Fan, W., et al. Graphscope: A unified engine for big graph processing. *Proc. VLDB Endow. 14*, 12 (2021).

[3] Linked data benchmark council. http://ldbcouncil.org/.

[4] Li, S., et al. Hiactor: an open-source hierarchical actor framework. https://github.com/alibaba/hiactor, 2023.

[5] Malewicz, G., et al. Pregel: a system for large-scale graph processing. In *SIGMOD 2010*.

[6] Qian, Z., et al. Gaia: A system for interactive analysis on distributed graphs using a high-level language. In *NSDI 2021*.

[7] Xue, L., et al. Flash: A framework for programming distributed graph processing algorithms. In *ICDE 2023, to appear*.

[8] Zhu, R., et al. Aligraph: A comprehensive graph neural network platform. *Proc. VLDB Endow. 12*, 12 (2019).