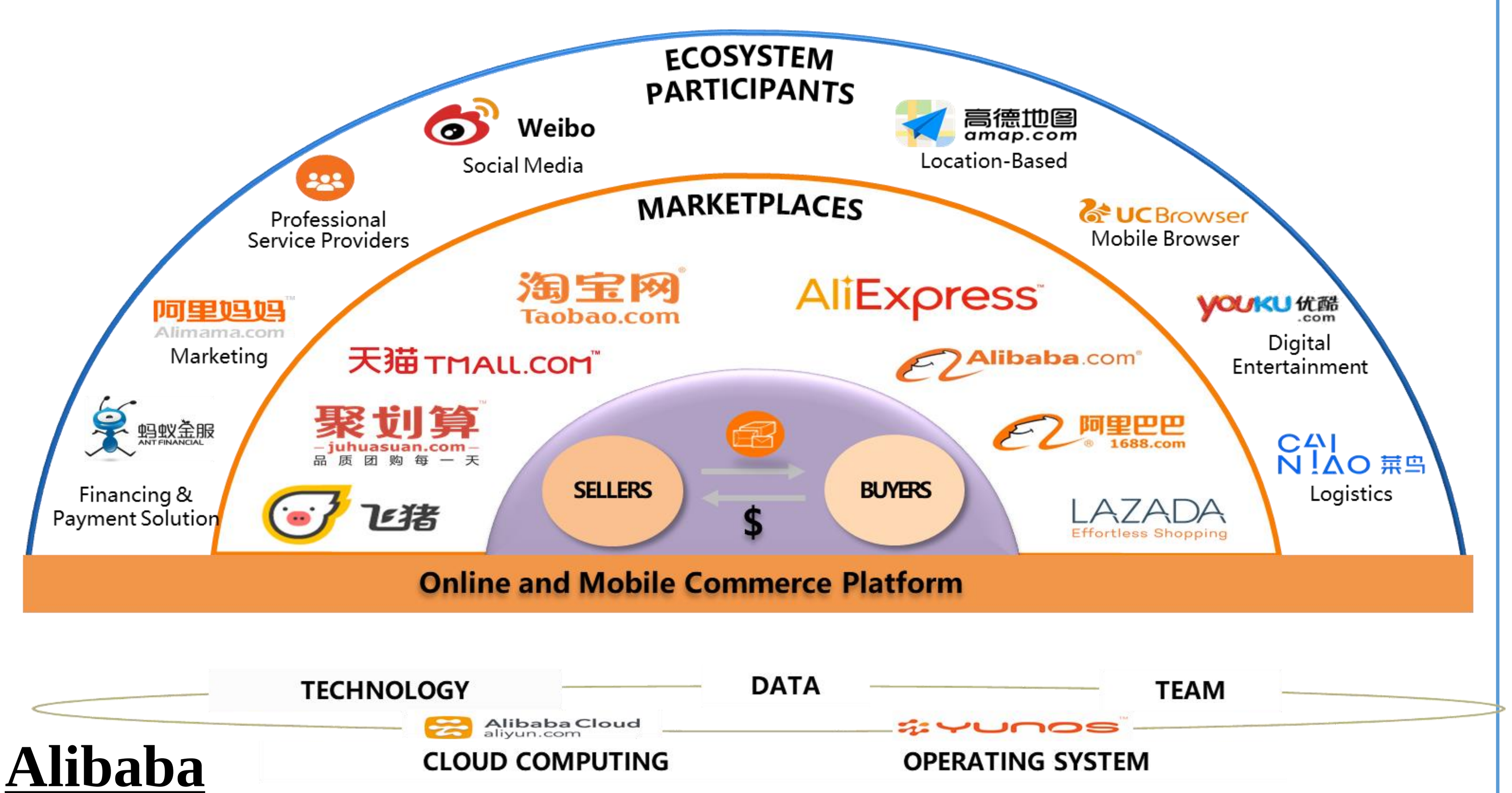
AliGraph: An Industrial Graph Neural Network Platform

Kun Zhao, Wencong Xiao, Baole Ai, Wenting Shen, Xiaolin Zhang, Yong Li, Wei Lin

PAI Team, Computing Platform, Alibaba Group

Graph Neural Network in Alibaba



Alibaba

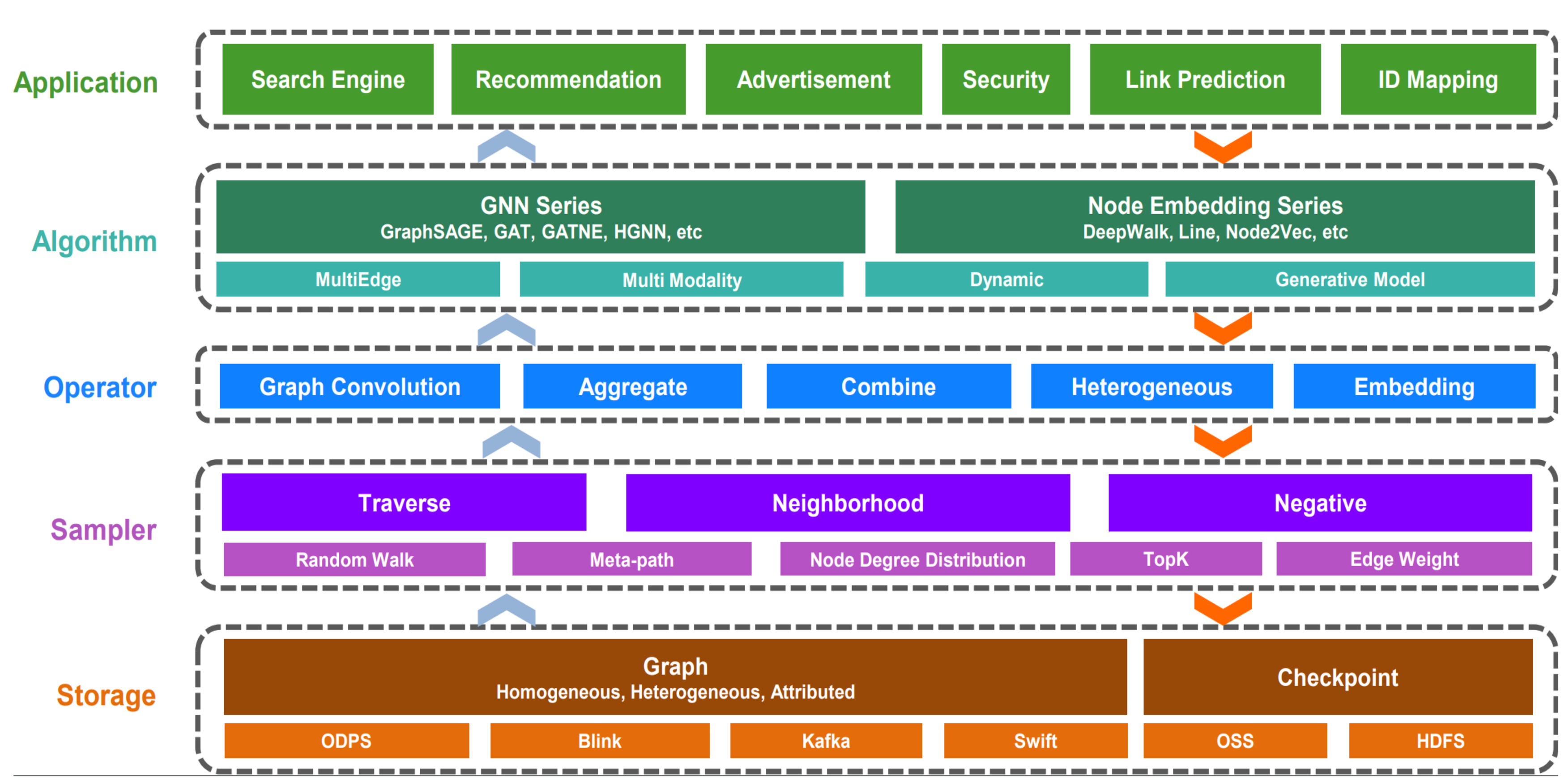
Leading e-commerce company: To make it easy to do business anywhere!

- **200+ Billions GMV** within one day!
- **Billions of orders** within one day!

Why Graph & GNN?

- Graph computing models are popular in Alibaba, as straightforward solutions to many practical problems
- Traditional **recommendation** and **CTR/CVR prediction** problems can be equivalently modeled with the attributed user-item graphs
- Graph neural network combines both deep learning and graph computing to integrate **end-to-end learning with inductive reasoning**, which is expected to solve the relational reasoning that deep learning cannot perform
- **More general** objective functions with global optimization; High-level proximity samples and modeling brings **more benefits** (e.g., more generalization and exploration, predictive graph changes)

AliGraph Overview

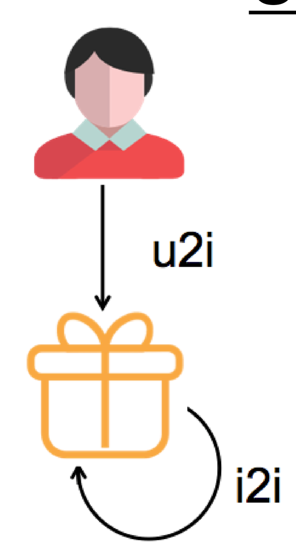


AliGraph is an industrial GNN platform, which bridges graph and neural network and aims for end-to-end solutions for researchers and developers. As an independent and portable system, the interfaces of AliGraph can be integrated with any tensor engine that is used for expressing neural networks. It provides an integrated development environment that empowers the whole procedure from data storage to application models, which largely reduces the cost of GNN exploration. Moreover, AliGraph shows excellent performance and scalability. It allows pluggable operators to adapt to the fast development of GNN community and outperforms existing systems an order of magnitude in terms of graph building and sampling.

Highlights

- Graph Data: Heterogeneous, tens of billions of edges, billions of nodes
- Distribution Scale: Thousands of servers
- Graph Building: Minutes
- Batch Sampling: Milliseconds

Unified Graph & Sampling Interface




```
from aligraph import *
g = Graph()
g.V("user").shuffle().batch(512)
 .outE("u2i").sample(10).by("EdgeWeight").inV()
 .outE("i2i").sample(5).by("TopK").inV()
```

Example for two-hop sampling on a heterogeneous graph

This example demonstrates how to implement it with Gremlin like API. The sampler starts from user vertices and then samples a batch of 512 user vertices randomly. For each sampled user vertex, sampling for 10 neighbor edges from user to item with probability proportional to the edge weight (EdgeWeightsampler) is performed. Thereby, a total of 5120 item vertices are selected. The second-hop sampling considers the outer edge set of each selected item vertex and samples the top 5 (TopKsampler) edges from item to item sorted by edge weight. In total, 31232 vertices are sampled, including 512 user vertices and 30720 item vertices, together with 5120 user-item edges and 25600 item-item edges.

Pluggable Operators

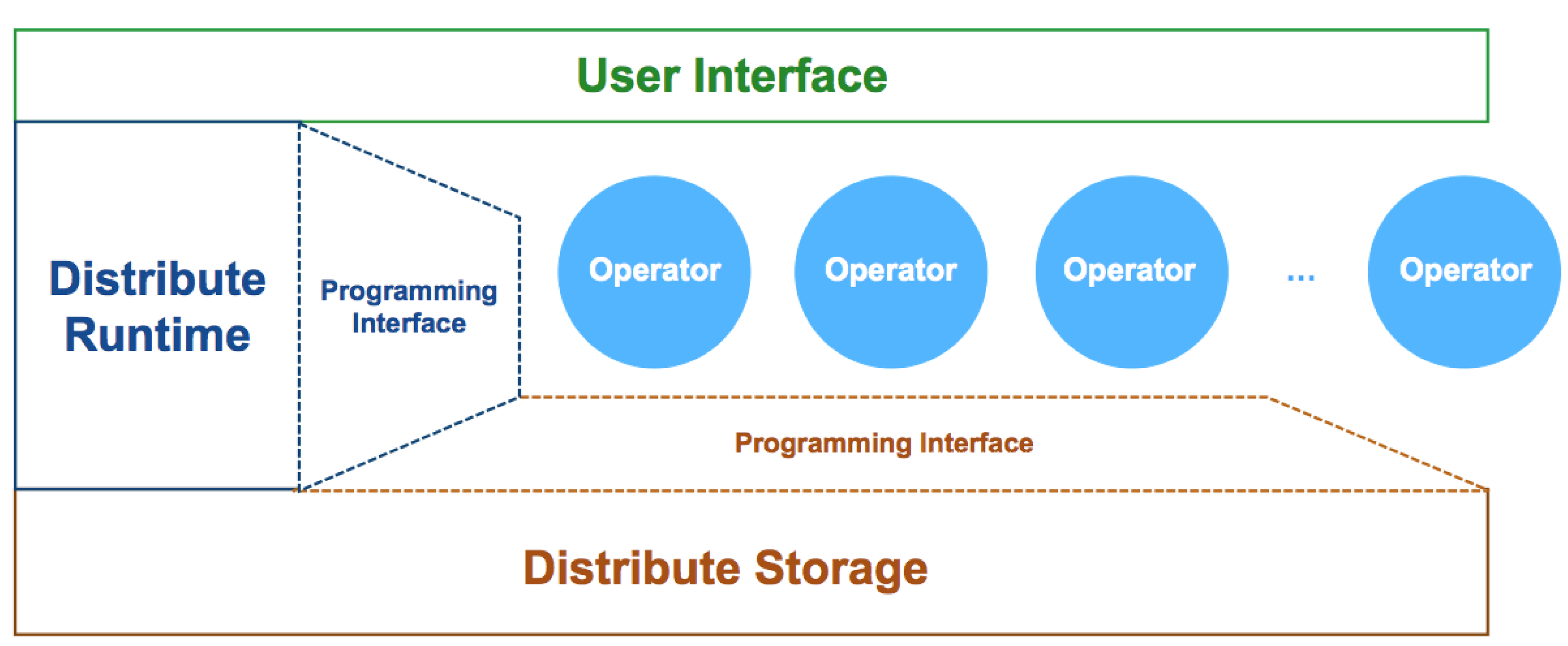
```
import aligraph
aligraph.DefineOp(Name="MySampler")
        .Param(type="int", shape=[])
        .Input(type="int", shape=[-1])
        .Output(type="int", shape=[-1])
        .Output(type="float", shape=[-1])
```

Define the schema of a customized sampler

```
g.V("user").shuffle().batch(512)
 .outE("u2i").sample(10).by("MySampler").inV()
```

An example of a user defined sampler

AliGraph simplifies the user efforts by hiding some system implementation details, such as RPC and message dispatching. First, the developer should define the schema of a sampler as illustrated above, including parameters, inputs, and outputs. Second, the abstract functions Process(), Map(), and Reduce() need to be implemented by the developer.



Lock-Free Multi-thread Graph Building

To construct a **heterogeneous** and **attributed** graph from raw data, three main phases are needed, including data reading and parsing, graph partitioning and dis-patching, and memory indexing. We propose a lock-free multi-thread method to reduce the time cost of them, which takes just **several minutes** to build a heterogeneous and attributed graph with billions of vertices and tens of billions edges.