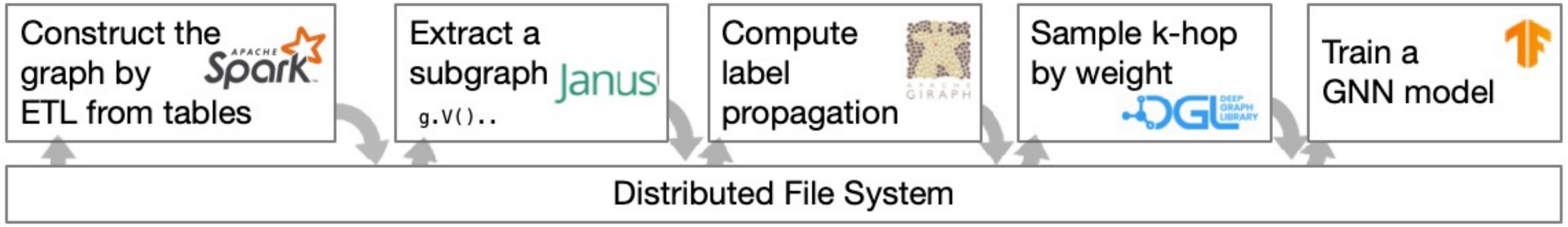
GraphScope Flex: A Graph Computing Stack with LEGO-Like Modularity

Wenyuan Yu

Alibaba Damo Academy

Applications across Graph Analytics, Traversal, and Learning

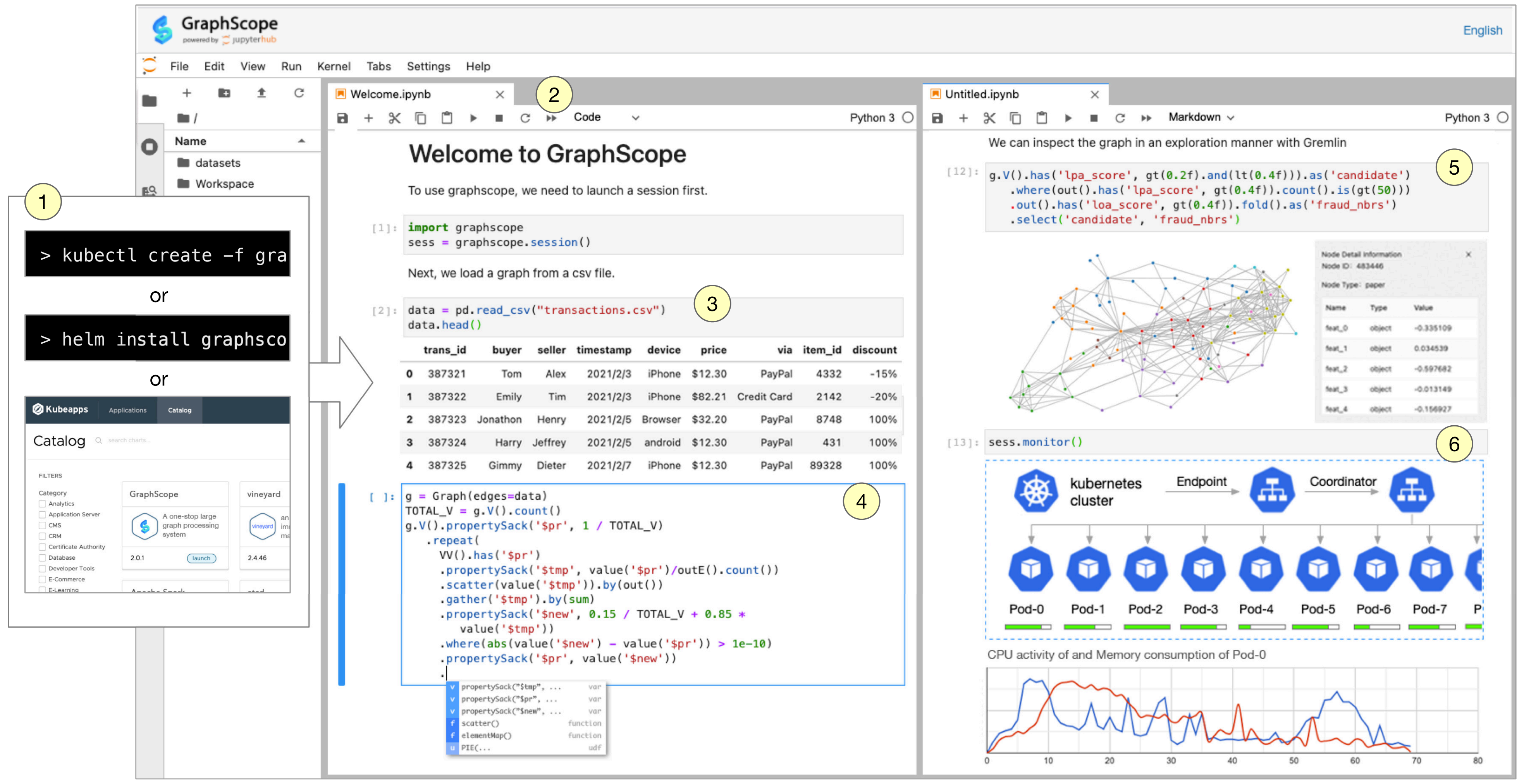
- More and more graph applications require more than just one type of workloads
- A simplified workflow for fraud-detection in Alibaba:
  - Construct a property graph from raw data using SQL;
  - Extract a subgraph using Gremlin;
  - A label-propagation algorithm for identifying fraudulent entities;
  - Graph sampling to conduct k-hop sampling by weight;
  - Train a GNN model using TensorFlow or PyG



Real life graph applications often involve multiple types of graph computations.

FLASHBACK: GraphScope: a Unified Engine for Big Graph Processing

Wenfei Fan, Tao He, Longbin Lai, Xue Li, Yong Li, Zhao Li, Zhengping Qian, Chao Tian, Lei Wang, Jingbo Xu, Youyang Yao, Qiang Yin, Wenyuan Yu, Jingren Zhou, Diwen Zhu, and Rong Zhu: GraphScope: A Unified Engine for Big Graph Processing, VLDB2021.



- A simple and unified **programming interface (Gremlin + Python)**;
- A **distributed dataflow runtime** that enables a separate optimization (or family of optimizations) for each graph operation in one carefully designed coherent framework.
- An **in-memory data store** that automatically manages the representation, transformation, and movement of intermediate data.
- We adopt the language integration approach advocated by Python to integrate the graph operators into a general-purpose high-level programming interface. This approach allows us to **seamlessly combine** GraphScope with other data processing systems

However, real-life graph applications are even more diverse and complex

- Multiplicity of Workloads: Graph analytics, interactive queries, pattern matching, and Graph Neural Networks (GNNs)
- Variety in Data Storage and Organization: Whether it's on-disk or in-memory, mutable or immutable, distributed or transactional?
- Range of Programming Interfaces: GQL, openCypher or Gremlin? Pregel, Gather-scatter, GraphBLAS or PIE? pyG or DGL
- Diverse Deployment Modes and Performance Needs: Offline data analytical tasks? Online services?

Real-life graph applications are diverse and complex

SPARQL
Gremlin
GraphQL
Graph Query Languages
openCypher
GQL
Higher query throughput? (e.g. snb interactive)
Lower latency for complex queries? (e.g. snb BI)

PageRank
WCC
CDLP, …
Graph Analytics
Pregel
Programming Models
GAS
PIE
GraphBLAS
Ligra

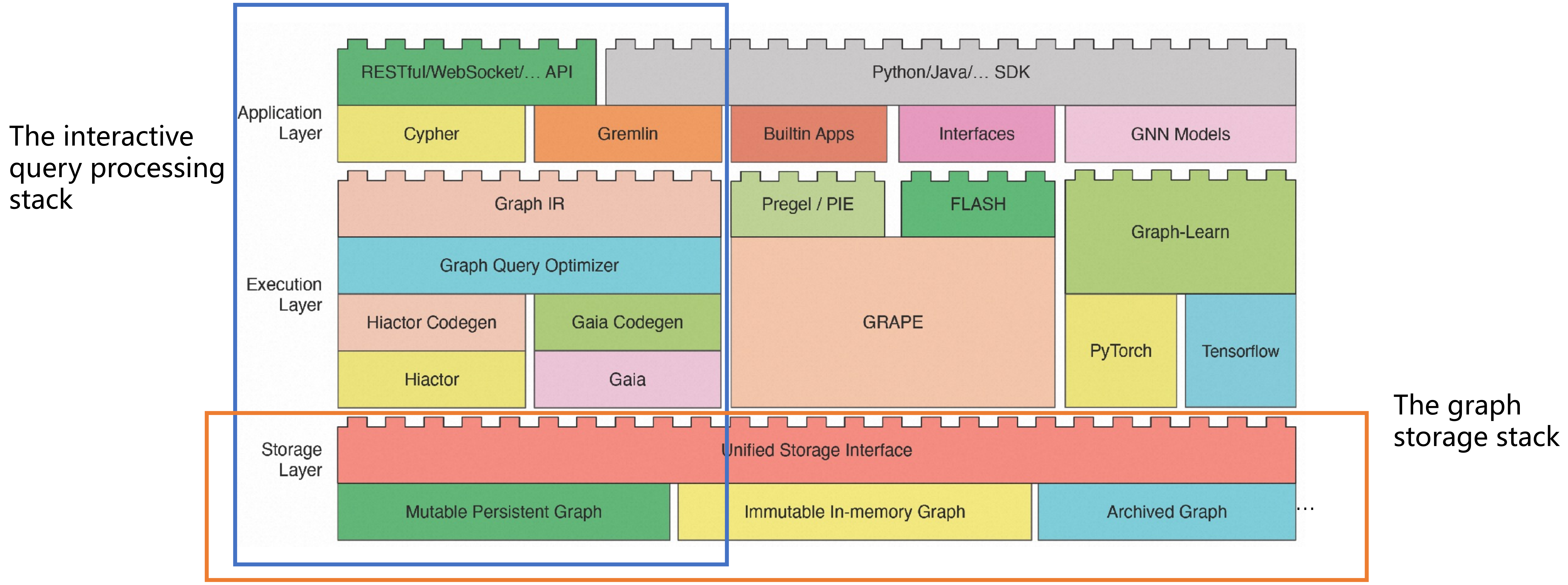
ML with Graphs
Graph Embedding
GNNs
Random Walk
Sampling-based GNNs
Subgraph GNNs
Wholegraph GNNs
Inference?
Training?

CRUD?
Read only?
Transactional?
In-memory?
On-disk?
Archival?
Static graph?
Streaming?
Property Graph?
RDF?
GPU?
Partitioned?
V-Cut?
E-Cut?
Hybrid-Cut?

KG
Hyper Graph
…

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

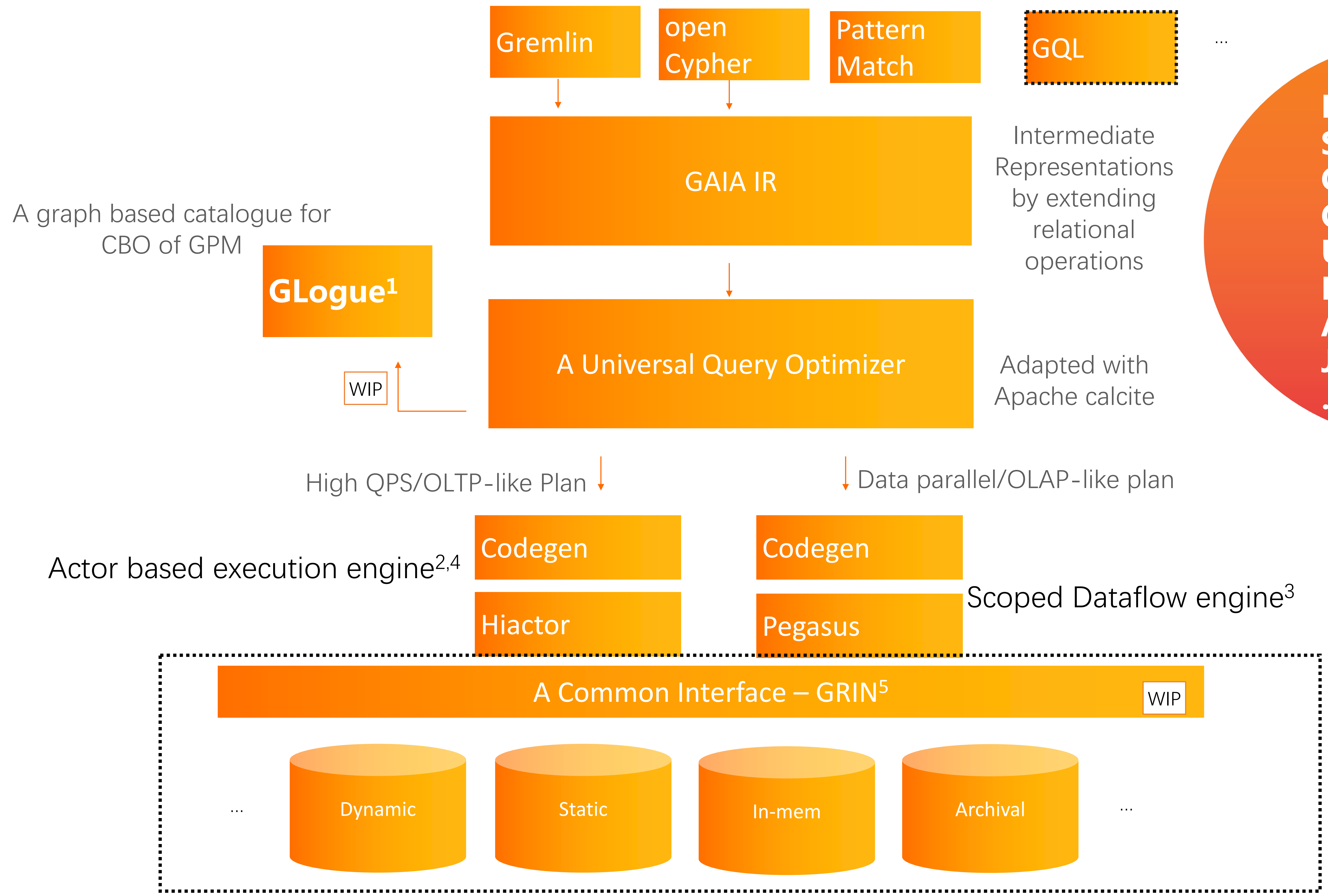
To address such diversities, we are developing the GraphScope Flex. It follows a modular and disaggregated design, where components are like LEGO bricks and user can easily make their customized builds and deployments.

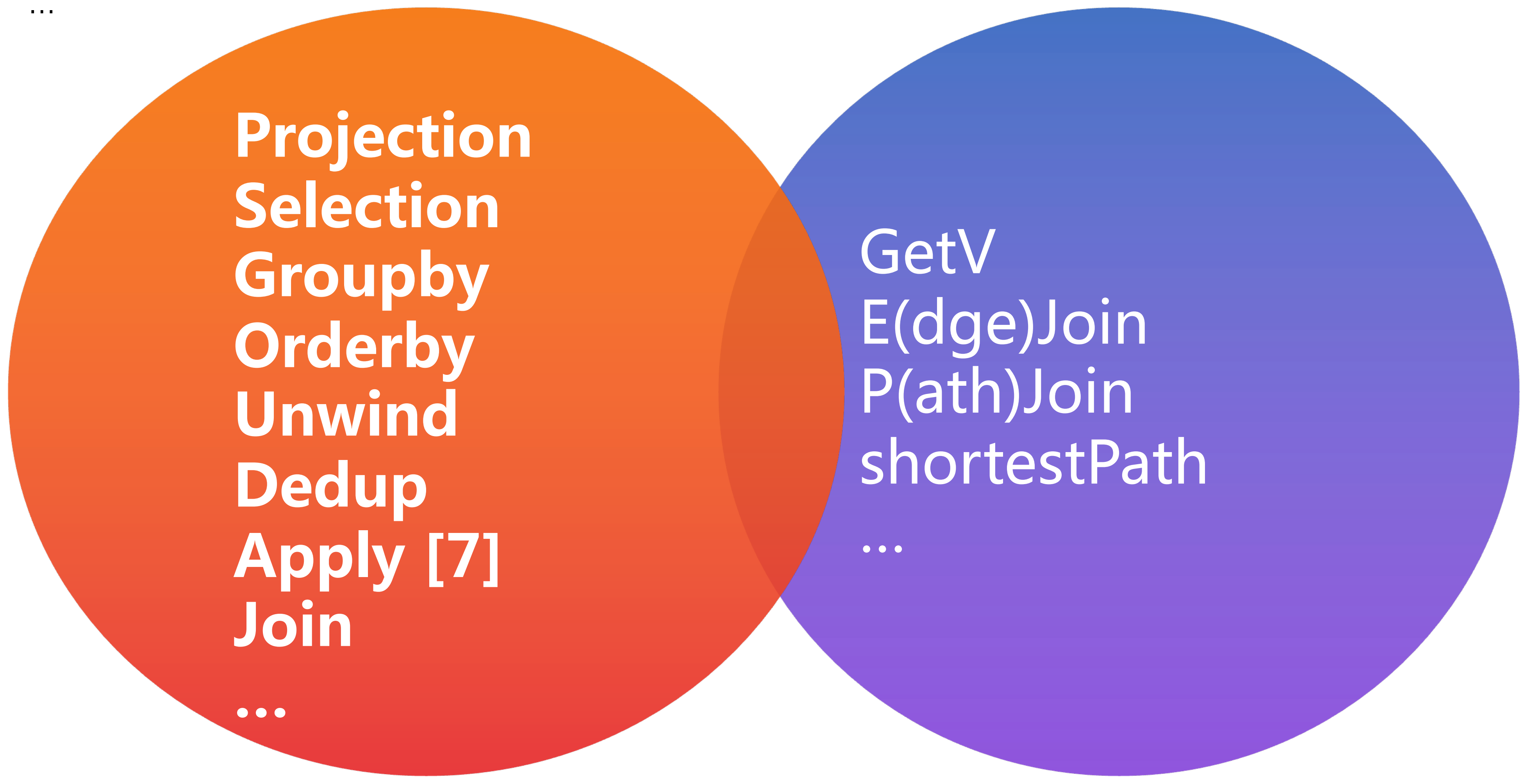


The interactive query processing stack

- How to support more query languages?
  - Gremlin
    - 100+ Steps…
  - Cypher, GQL, …
- How to effectively optimize graph queries?
- How to support more types of workloads? Higher QPS or Data-parallel?

Our approach..

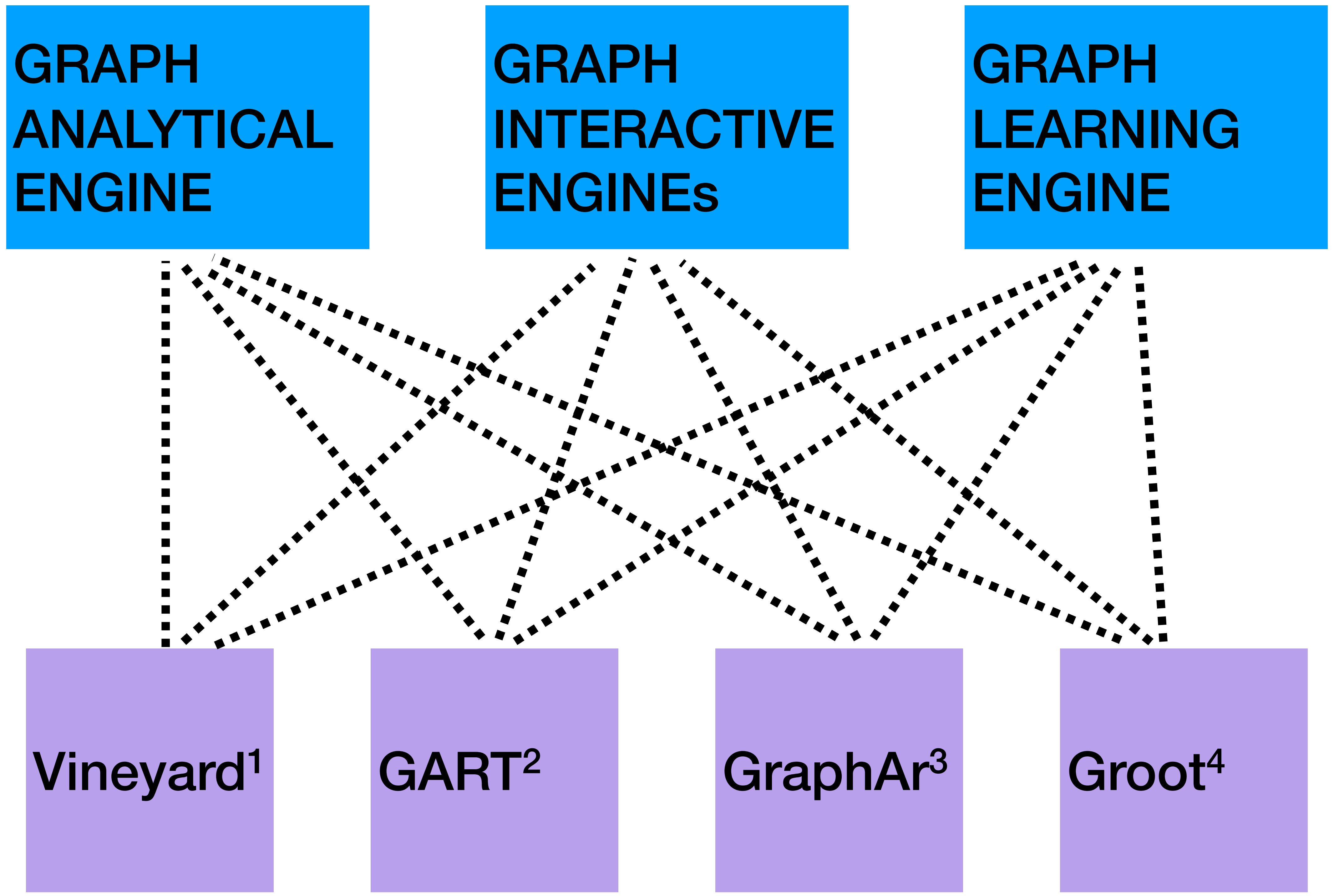


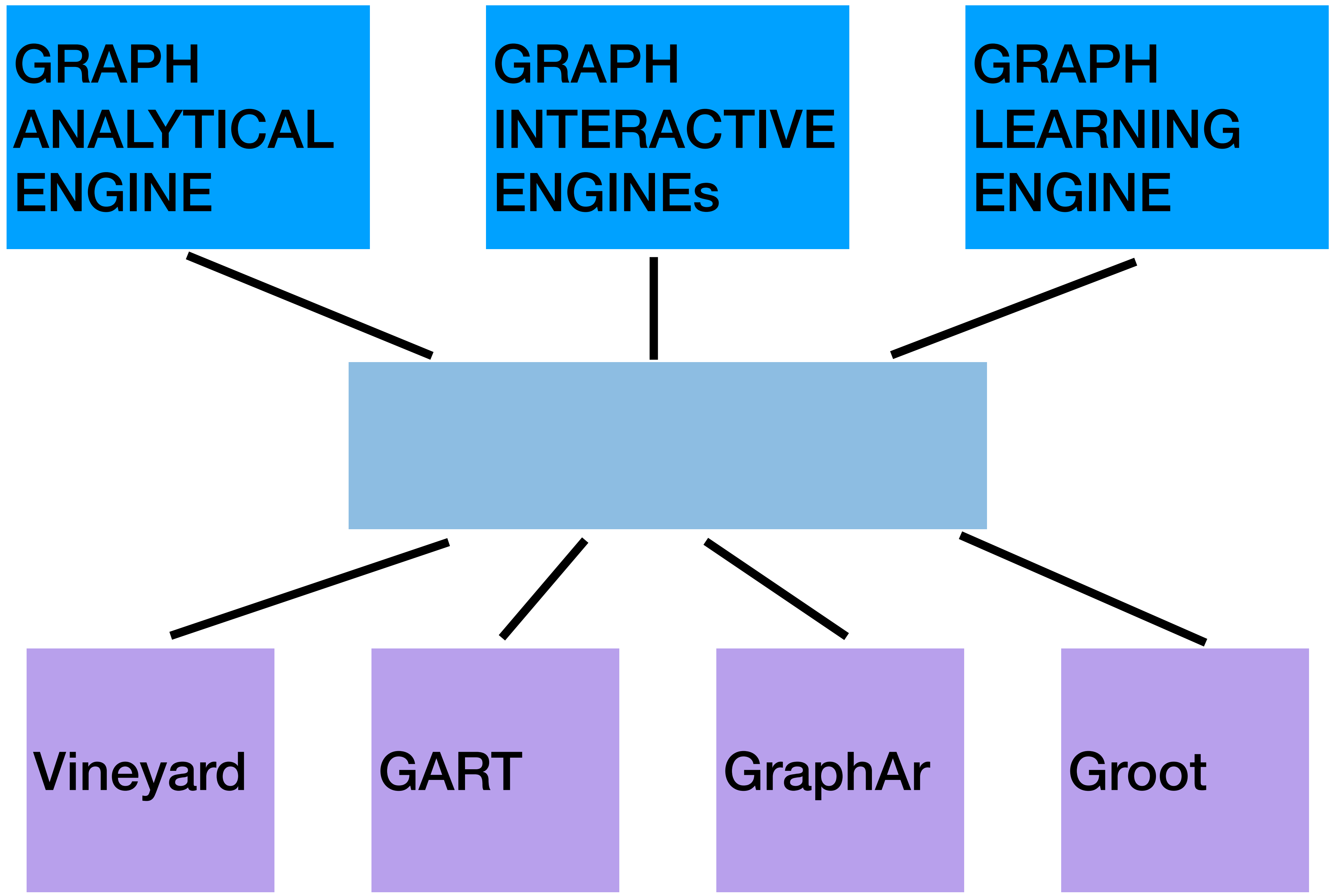


1. GLogS: Interactive Graph Pattern Matching Query At Large Scale. ATC 23
2. Banyan: a scoped dataflow engine for graph query service. VLDB 22
3. GAIA: A System for Interactive Analysis on Distributed Graphs Using a High-Level Language. NSDI 21
4. https://github.com/alibaba/hiactor
5. https://github.com/GraphScope/GRIN

How computing engines interact with storage engines in GraphScope?

With or without a common interface

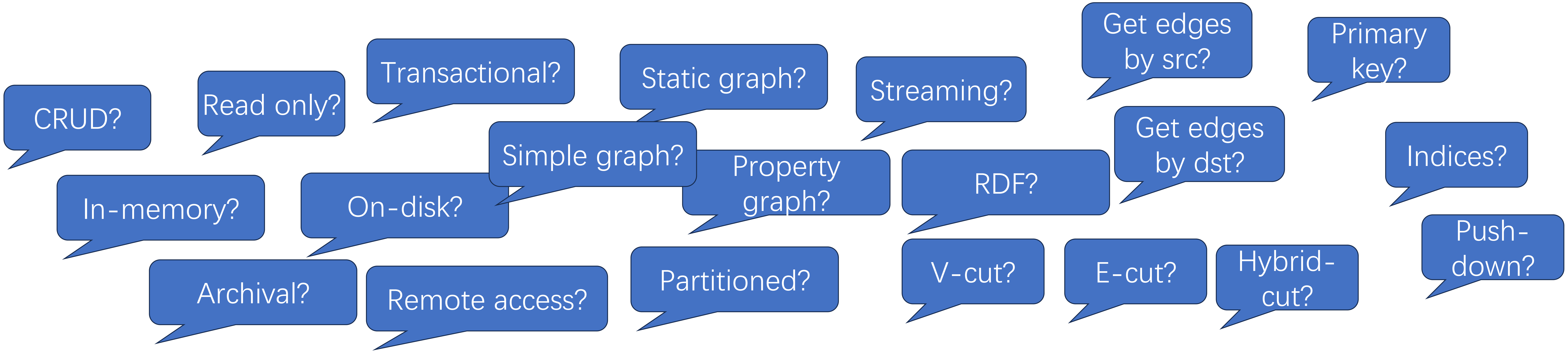




1. Vineyard: Optimizing Data Sharing in Data-Intensive Analytics. SIGMOD 23
2. Bridging the Gap between Relational OLTP and Graph-based OLAP. ATC 23
3. Graph Archive format (shown later)
4. A rocksDB based distributed on-disk graph storage

Understanding the complexity of graph storage abstraction is crucial

Graph storages can be diverse. The requirements of computing engine accessing the data are different as well.



The design of GRIN

- GRIN is a proposed standard **g**raph **r**etrieval **in**terface in GraphScope
- The goal is to simplify the integrations between different computing engines and storage engines from M * N to M + N
- To achieve the goal:
  - It only supports the read-path over an immutable graph/snapshot. (no WRITEs at the moment)
  - Using a trait abstraction for graph elements (V, E, …), inspired by POSIX (e.g. a FD can and cannot do sth with it). API is written in C, which makes GRIN portable to engines written in different programming languages like Rust, Java and C++
  - GRIN defines a set of handles such as vertex, edge,, and abstracts the operations (e.g., getting the adjacent edges of a vertex) as a set of APIs in different header files.
  - C Macros and a YAML file to tell computing engines what features are supported by a storage.
  - The handles and APIs are defined The APIs must be well-abstracted and low-level to avoid introducing excessive performance loss.

GRIN is still a work in progress

- The three computing engines (analytical, interactive and learning) in GraphScope are being rewrited to adapt their graph retrieval layer using GRIN APIs,.
- 3 (out of 5) storage engines are being adapted to provide their GRIN implementations, namely Vineyard[1], GART[2] and GraphAr in GraphScope

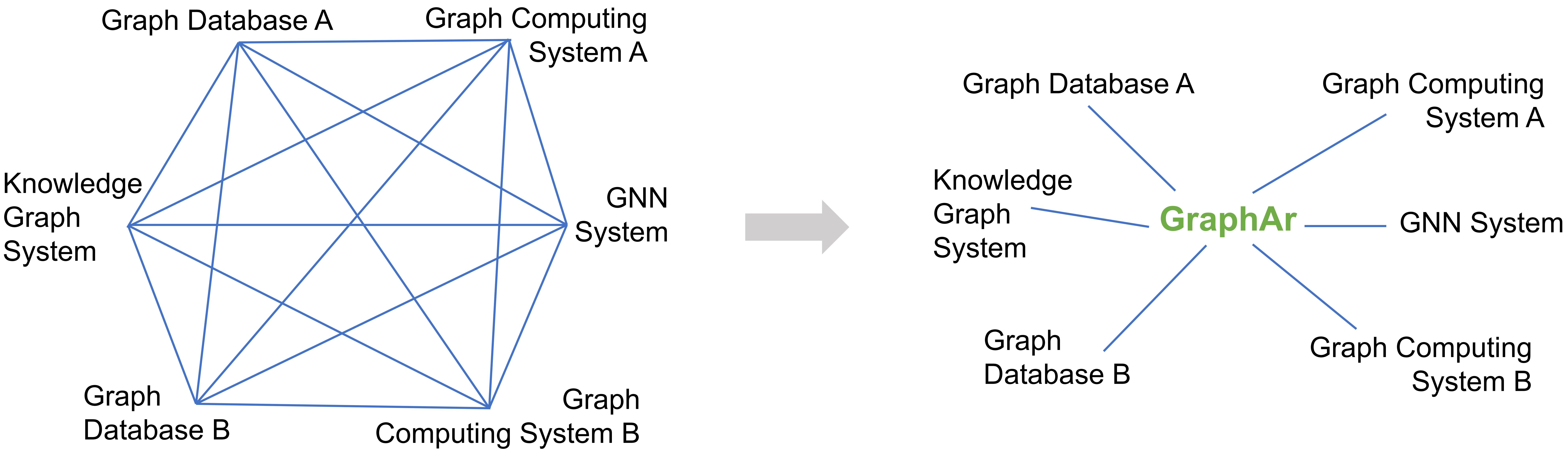
- The preliminary results shows that the performance overhead of GRIN is always less than 10%, and sometimes the performance is better if the original integration without GRIN is not designed or implemented carefully.

- Watch https://github.com/GraphScope/GRIN for progress.
- Further ahead, we plan to make GRIN support more external graph storage and provide a way to abstract a graph from other type of storages (tabular, …) while easier to use.

1. Vineyard: Optimizing Data Sharing in Data-Intensive Analytics. SIGMOD 23
2. Bridging the Gap between Relational OLTP and Graph-based OLAP. ATC 23

GraphAr: An Open Source File Format for Archiving and Exchanging Graph Data

GraphAr (short for "Graph Archive") is a project that aims to make it easier for diverse applications and systems (in-memory and out-of-core storages, databases, graph computing systems, and interactive graph query frameworks) to build and access graph data conveniently and efficiently.



Objectives

GraphAr is designed to serve two main scenarios:

- As a standardized file format for importing, exporting and archiving of the graph data which can be used by diverse existing systems, reducing the overhead when various systems co-work.
- As a direct data source for graph processing applications.

The GraphAr project provides:

- The GAR file format: a standardized system-independent file format for storing graph data.
- A set of libraries for reading, writing and transforming GAR files (presently available in C++ and Spark).
- Examples of how to use GraphAr to write graph algorithms, or collaborate with existing systems like GraphScope.

Features of GraphAr

- The file format supports the property graphs and different representations for the graph topology (COO, CSR and CSC).
- It is compatible with existing widely-used file formats including ORC, Parquet (and less ideally CSV).
- Apache Spark can be utilized to generate, load and transform GraphAr files.
- It is convenient for use in a variety of single-machine/distributed graph processing systems, databases, and other downstream computing tasks.
- It enables users to conveniently perform operations without modifying the payload files, such as appending new vertices, adding new properties, or constructing a new graph with a set of selected vertices and edges.

GraphAr File Format – Vertices

Physical table of vertices

- label: person, chunk size: 500
- property groups: (id), (firstName, lastName, gender)

	id
0	933
1	6597069767117
…	…
499	13194139534267

./vertex/person/id/chunk0

	firstName	lastName	gender
0	Mahinda	Perera	male
1	Eli	Peretz	female
…	…	…	…
499	Asha-Rose	Chung	male

./vertex/person/firstName_lastName_gender/chunk0

	id
500	15393162788965
501	15393162789614
…	…
903	32985348834100

./vertex/person/id/chunk1

	firstName	lastName	gender
500	Hans	Becker	male
501	Adi	Cohen	female
…	…	…	…
903	Bruno	Oliveira	male

./vertex/person/firstName_lastName_gender/chunk1

GraphAr File Format – Edges

Physical table of edges

- label: person-knows-person, type: CSR
- chunk size: 1024, property group: (creationDate)

	offset
0	0
1	3
…	…
499	3772
	3778

./edge/person_knows_person/ordered_by_source/offset/chunk0

source	destination
0	87
…	…
164	829

./edge/person_knows_person/ordered_by_source/adj_list/part0/chunk0

creationDate
2010-07-30T15:19:53.298+0000
…
2010-06-11T19:23:42.146+0000

./edge/person_knows_person/ordered_by_source/creationDate/part0/chunk0

source	destination
164	30
…	…
269	565

./edge/person_knows_person/ordered_by_source/adj_list/part0/chunk1

creationDate
2010-05-16T17:41:47.623+0000
…
2011-12-22T17:56:13.491+0000

./edge/person_knows_person/ordered_by_source/creationDate/part0/chunk1

source	destination
269	321
…	…
499	628

./edge/person_knows_person/ordered_by_source/adj_list/part0/chunk2

creationDate
2012-01-04T13:29:11.784+0000
…
2012-08-03T01:00:51.312+0000

./edge/person_knows_person/ordered_by_source/creationDate/part0/chunk2

	offset
500	0
501	1
…	…
903	2848
	2848

./edge/person_knows_person/ordered_by_source/offset/chunk1

source	destination
500	623
…	…
637	704

./edge/person_knows_person/ordered_by_source/adj_list/part1/chunk0

creationDate
2012-04-21T19:08:41.647+0000
…
2012-08-10T02:49:19.288+0000

./edge/person_knows_person/ordered_by_source/creationDate/part1/chunk0

source	destination
638	375
…	…
793	884

./edge/person_knows_person/ordered_by_source/adj_list/part1/chunk1

creationDate
2012-06-26T02:41:08.148+0000
…
2012-03-10T06:07:41.141+0000

./edge/person_knows_person/ordered_by_source/creationDate/part1/chunk1

source	destination
793	662
…	…
901	252

./edge/person_knows_person/ordered_by_source/adj_list/part1/chunk2

creationDate
2012-02-19T06:42:02.399+0000
…
2012-08-13T10:11:20.606+0000

./edge/person_knows_person/ordered_by_source/creationDate/part1/chunk2

GraphAr File Format – Meta Files

GraphInfo: ldbc_sample.graph.yml

```yaml
name: ldbc_sample
vertices:
  - person.vertex.yml
edges:
  - person_knows_person.edge.yml
version: gar/v1
```

VertexInfo: person.vertex.yml

```yaml
label: person
chunk_size: 100
prefix: vertex/person/
property_groups:
  - properties:
      - name: id
        data_type: int64
        is_primary: true
    prefix: id/
    file_type: csv
  - properties:
      - name: firstName
        data_type: string
        is_primary: false
      - name: lastName
        data_type: string
        is_primary: false
      - name: gender
        data_type: string
        is_primary: false
    prefix: firstName_lastName_gender/
    file_type: csv
version: gar/v1
```

EdgeInfo: person_knows_person.edge.yml

```yaml
src_label: person
edge_label: knows
dst_label: person
chunk_size: 1024
src_chunk_size: 100
dst_chunk_size: 100
directed: false
prefix: edge/person_knows_person/
adj_lists:
  - ordered: true
    aligned_by: src
    prefix: ordered_by_source/
    file_type: csv
    property_groups:
      - prefix: creationDate/
        file_type: csv
        properties:
          - name: creationDate
            data_type: string
            is_primary: false
  - ordered: true
    aligned_by: dst
    prefix: ordered_by
    file_type: csv
    property_groups:
      - prefix: creati
        file_type: csv
        properties:
          - name: crea
            data_type:
            is_primary
version: gar/v1
```

Future of GraphAr

- It is currently open-sourced at https://github.com/alibaba/GraphAr

- Support more file formats, more standard and user-defined data types.
- More graph features: RDF, time-series
- Encoding optimizations.
- Complete Spark suite to transform create GraphAr files.
- Integrations with popular graph database, such as Neo4j, Nebula, TuGraph, PyG …
- Explore the use GraphAr for data lake of graphs.

- We aim to make GraphAr vendor-neutral (e.g., Apache Foundation) when it matures.
- Current contributors: Alibaba Damo Academy, Zhejiang Lab and Nebula Graph
- New contributors are welcome!

Conclusion

- GraphScope Flex is an on-going efforts to make our graph computing stack more composable to tackle diverse graph applications. Areas covered by this talk:
  - A new query evaluation framework for a core subset of openCypher, Gremlin and GQL.
    - Multiple language frontends
    - An IR for graph queries (recursion not supported yet)
    - A query optimizer based on Apache Calcite with a graph catalogue Glogue for CBO.
    - Execution engines for query throughput and data parallel queries.
  - A new storage layer:
    - A common interface GRIN: https://github.com/GraphScope/GRIN
    - A graph format for archiving graph data: (aim for Apache Incubator) https://github.com/alibaba/GraphAr

Scan to learn more from GraphScope github repo: