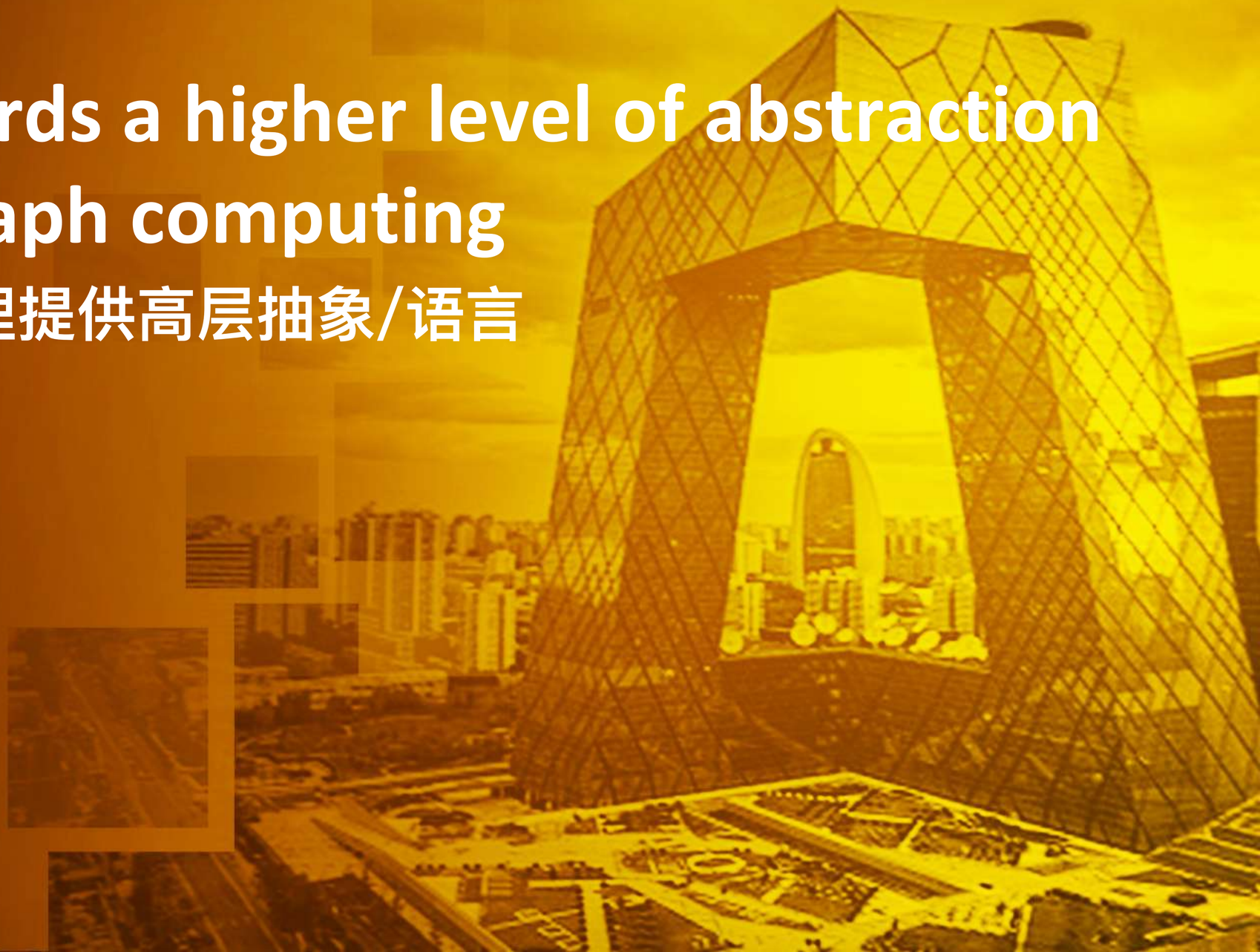
FLASH: Towards a higher level of abstraction in parallel graph computing

为并行图数据处理提供高层抽象/语言



Zhengping Qian (钱正平)

Senior Staff Engineer

Alibaba

FLINK FORWARD

Graph scenarios at Alibaba

阿里图场景

Graph technologies and challenges

图处理技术与挑战

Project Flink FLASH

FLASH项目

On-going research

更多研究工作

FLINK FORWARD

Graph scenarios at Alibaba

阿里图场景

Graph technologies and challenges

图处理技术与挑战

Project Flink FLASH

FLASH项目

On-going research

更多研究工作

Graph analytics for better decisions

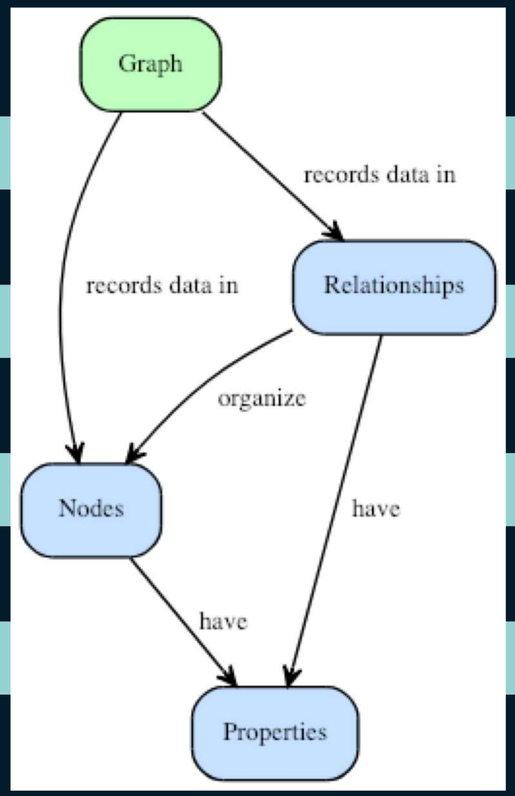
数据间的复杂关联与结构特 → 更精准和可靠的决策

Graph

Nodes

Properties

Relationships

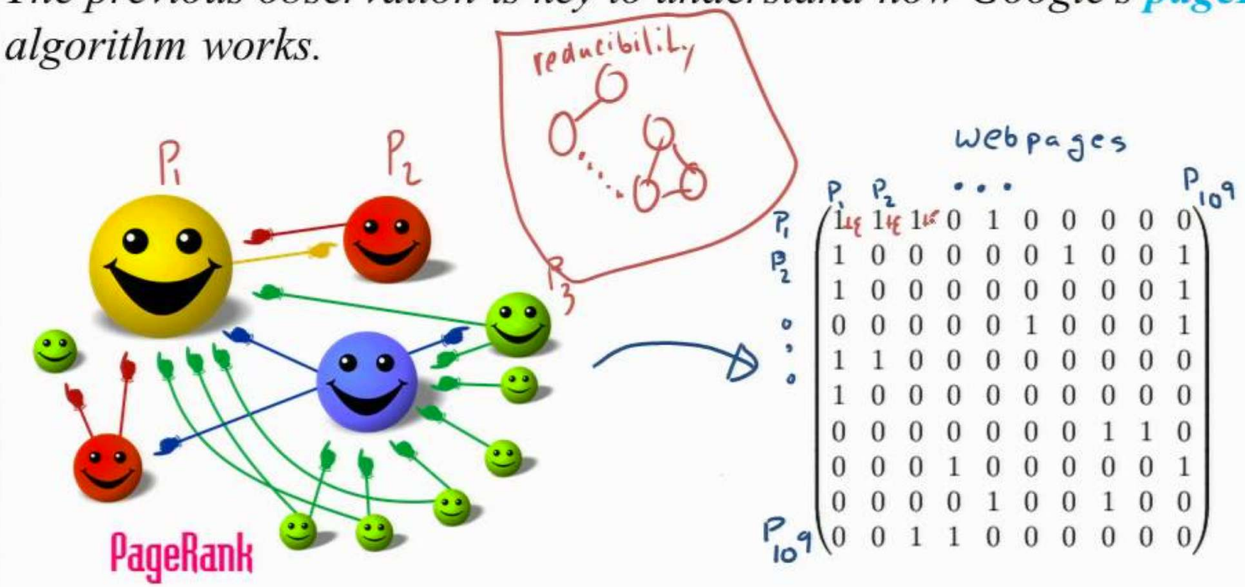


Machine Learning

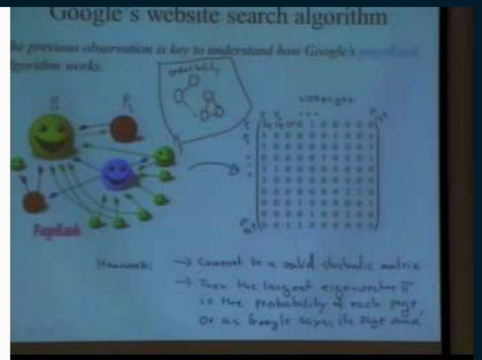
Better Decision

Google's website search algorithm

The previous observation is key to understand how Google's ***pageRank*** *algorithm works.*



Homework:
→ Convert to a valid stochastic matrix
→ Then the largest eigenvector Π is the probability of each page, or as Google says: its page rank

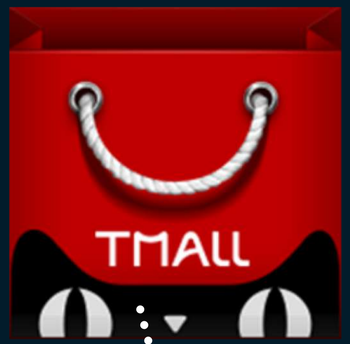


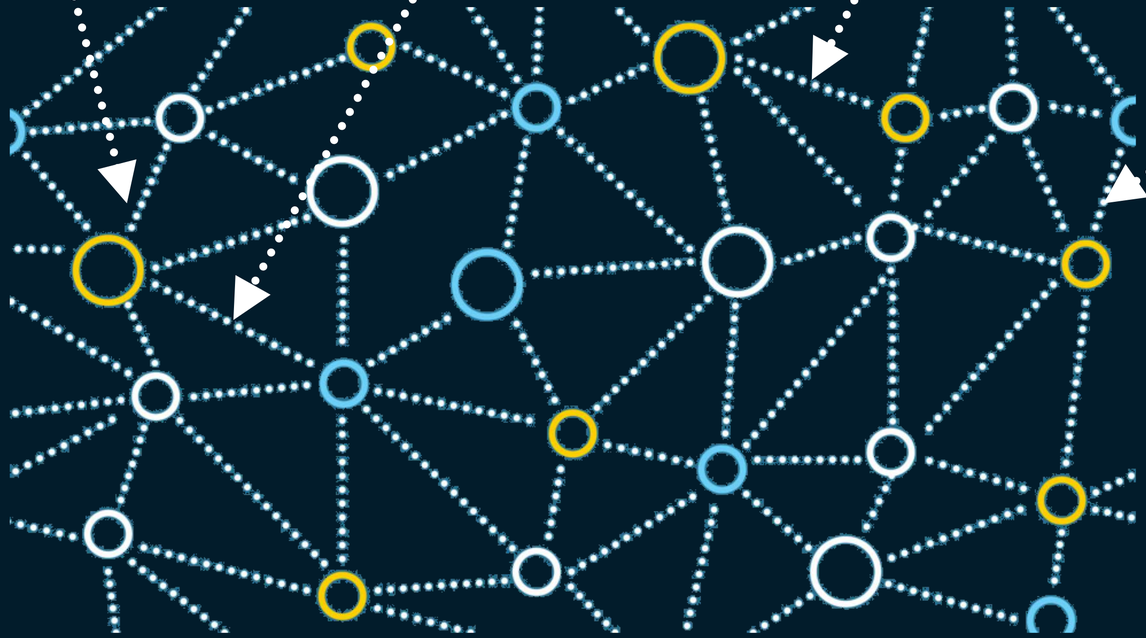
UBC Computer Science

CPSC 340
Nando De Freitas

Sep 12 2012
15:24

https://www.artificial-intelligence.video/undergraduate-machine-learning-4-introduction-to-probability-linear-algebra-and-pagerank





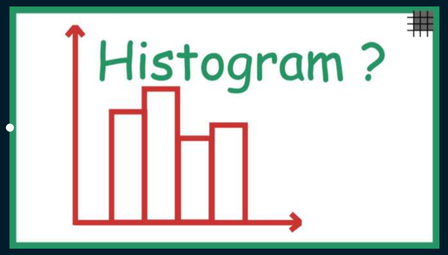
Graph matters at Alibaba, too

阿里丰富的图场景

- Billions of vertices (几十亿的点)
- Hundred billions of edges (成百上千亿的边)
- Real-time updates (实时更新)
  - e.g., 320K Tmall transactions/s in 2017

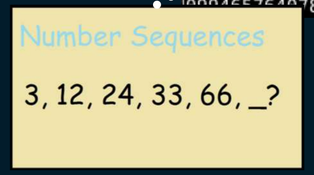
Graph data processing paradigms
图数据处理范式

See
看

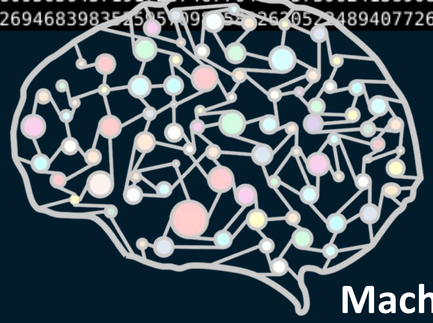


Get statistics
特征统计

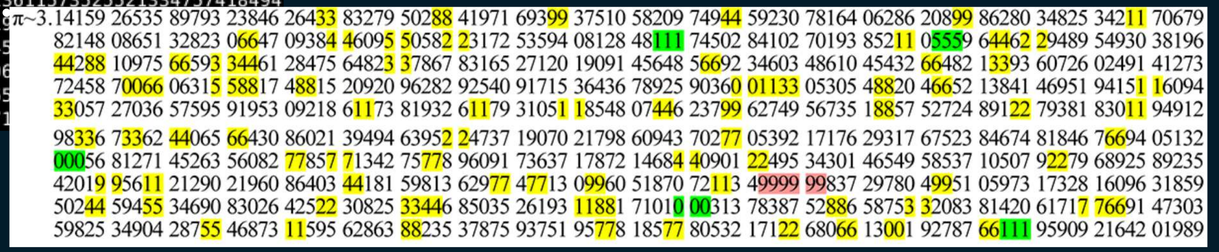
Insights
洞见



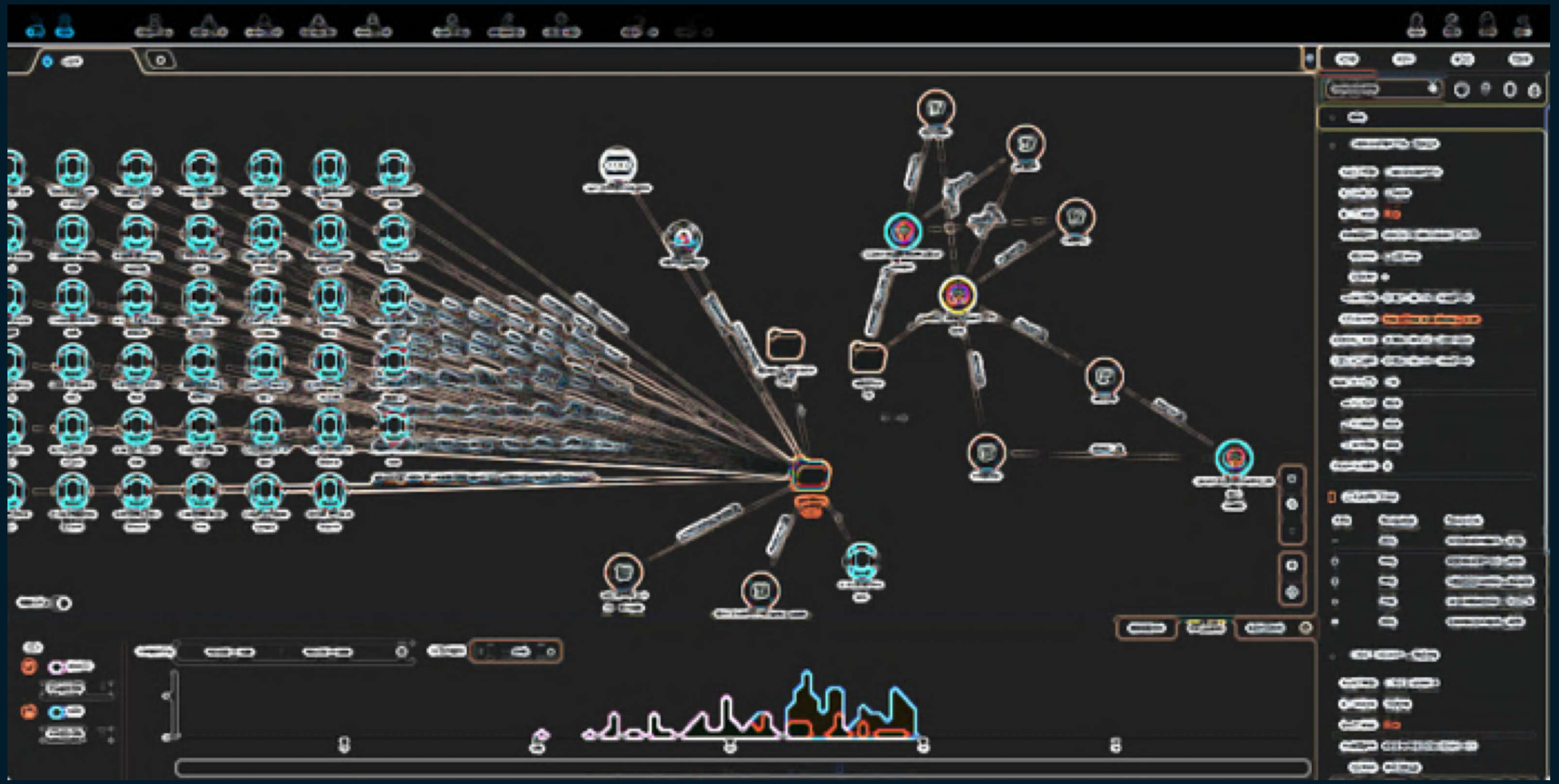
On-line inference
在线预测



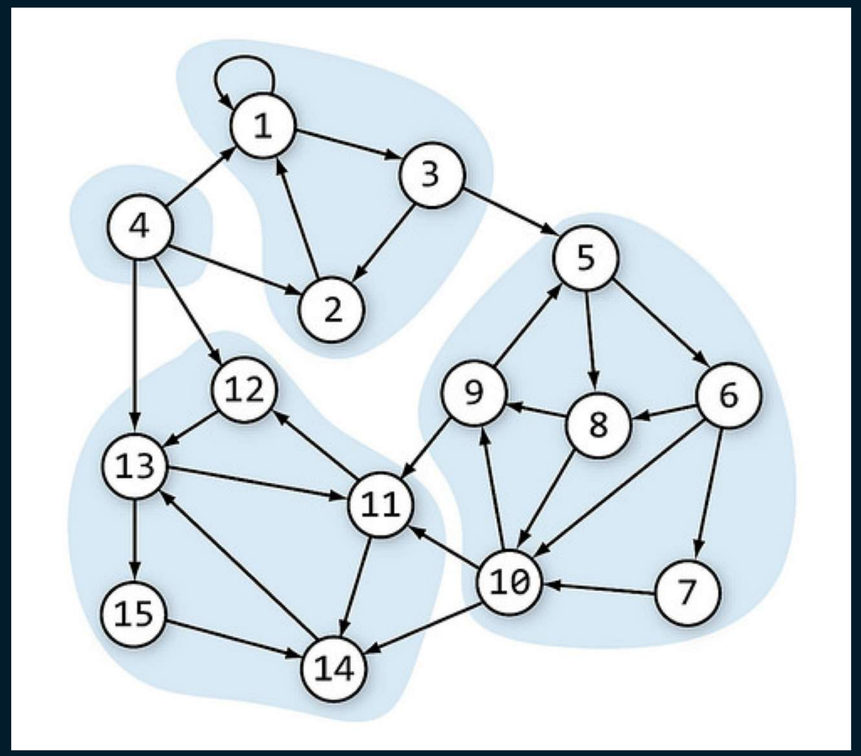
Machine learning
机器学习



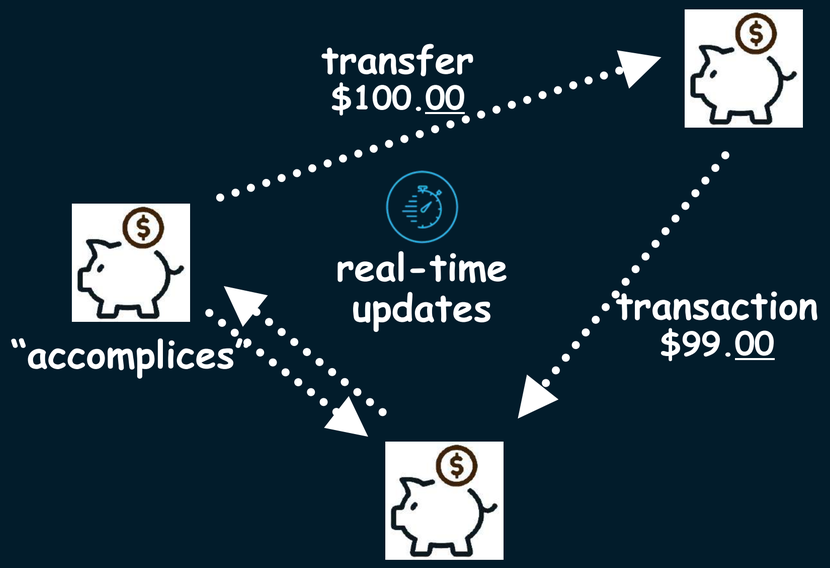
Pattern mining
模式匹配



Graph algorithms
图算法

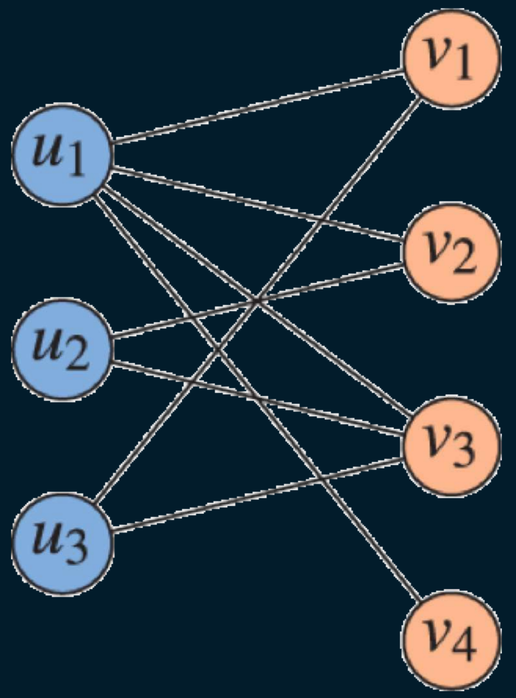


Pattern-based fraud detection
基于子图模式的异常检测



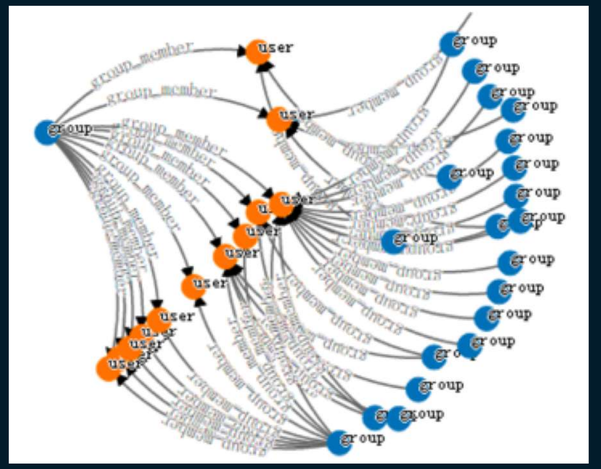
Click farming-simple

刷单1



Click farming-complex

刷单2

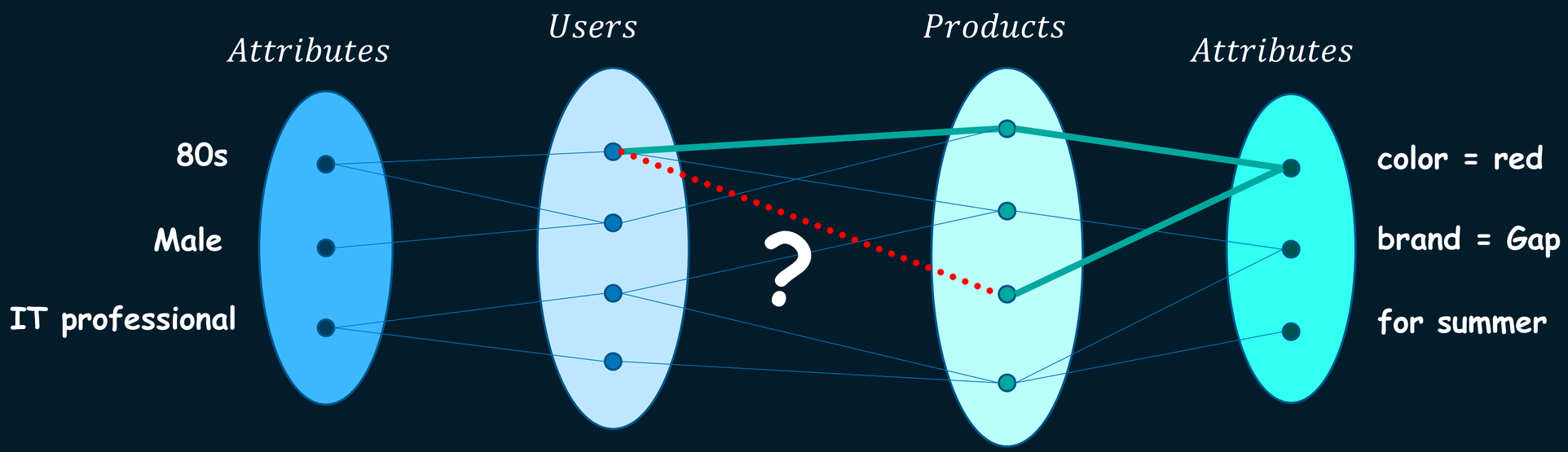


On-line gambling

线上赌博

Community-based recommendation in Taobao

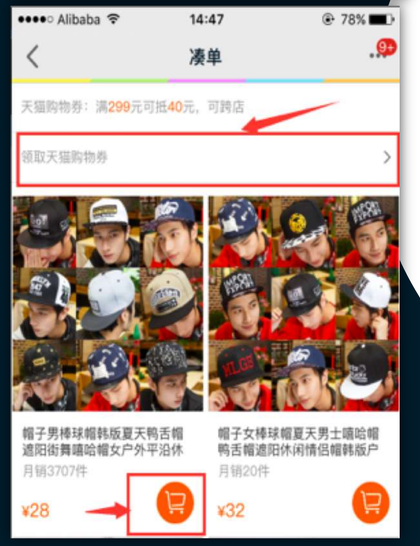
基于图的机器学习应用于淘宝推荐

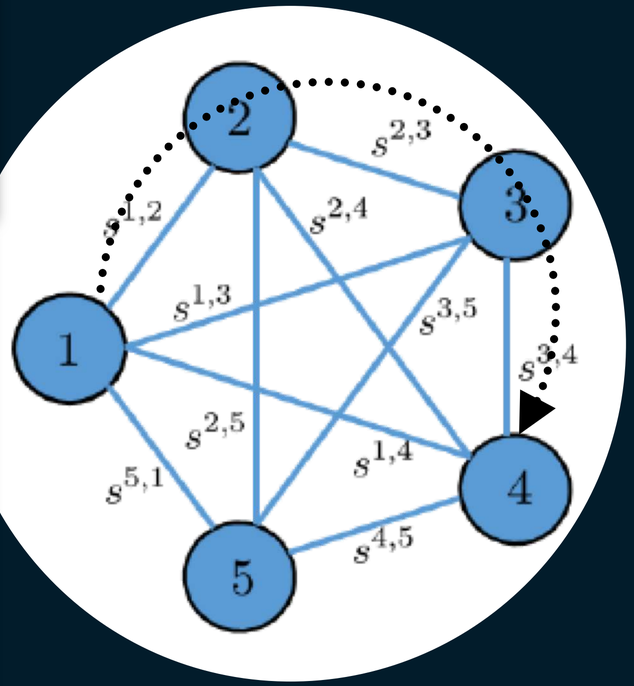


We encode the following “intuitions” 图结构信息有利于解决冷启动/结果可解释性等难题，也能引入多样性！

- *Reflect each user/item’s uniqueness*
- *Utilize structural information (transaction or viewing edges)*
- *Utilize user/item attributes (handling cold start problem in CF)*
- *Introduce nice “surprise” via graph exploration*

On-line knowledge graph reasoning
基于知识图谱的在线推理





Bundle recommendation

凑单推荐

Knowledge-powered intelligent service

认知服务

FLINK FORWARD

Graph scenarios at Alibaba

阿里图场景

Graph technologies and challenges

图处理技术与挑战

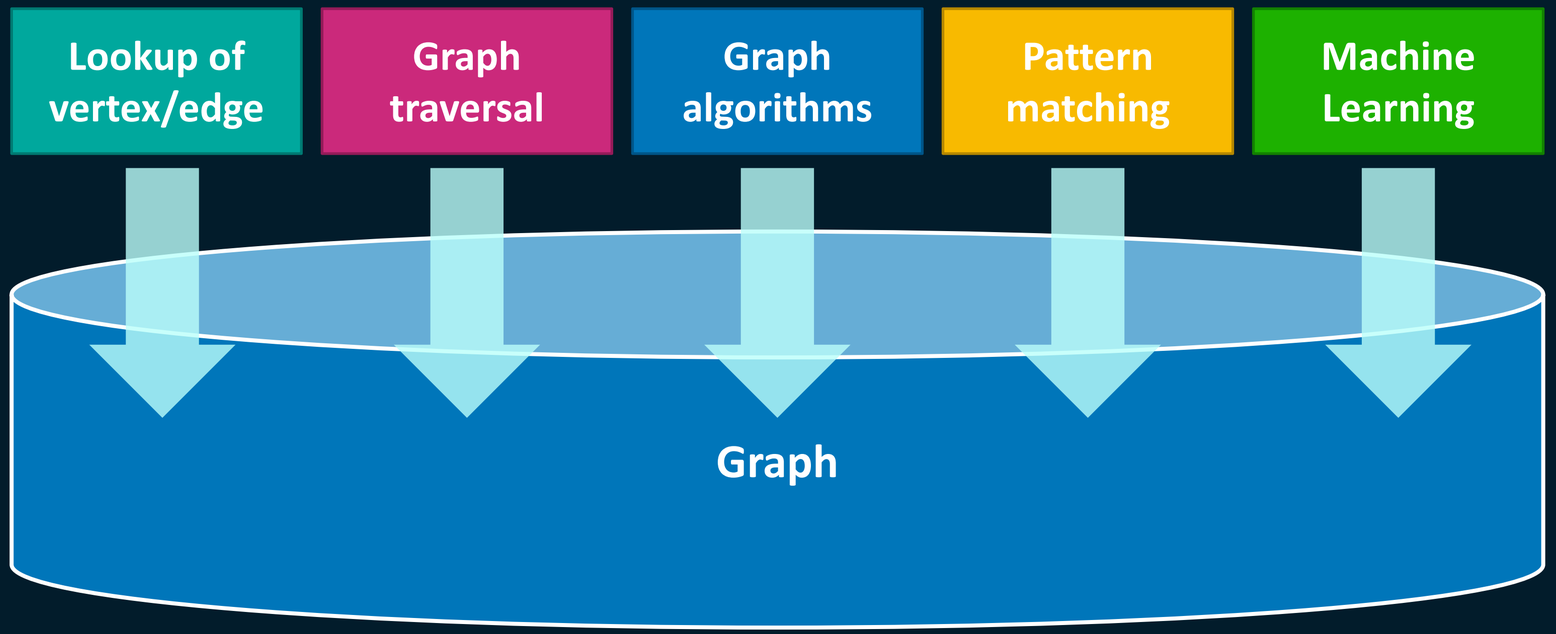
Project Flink FLASH

FLASH项目

On-going research

更多研究工作

Vision
愿景

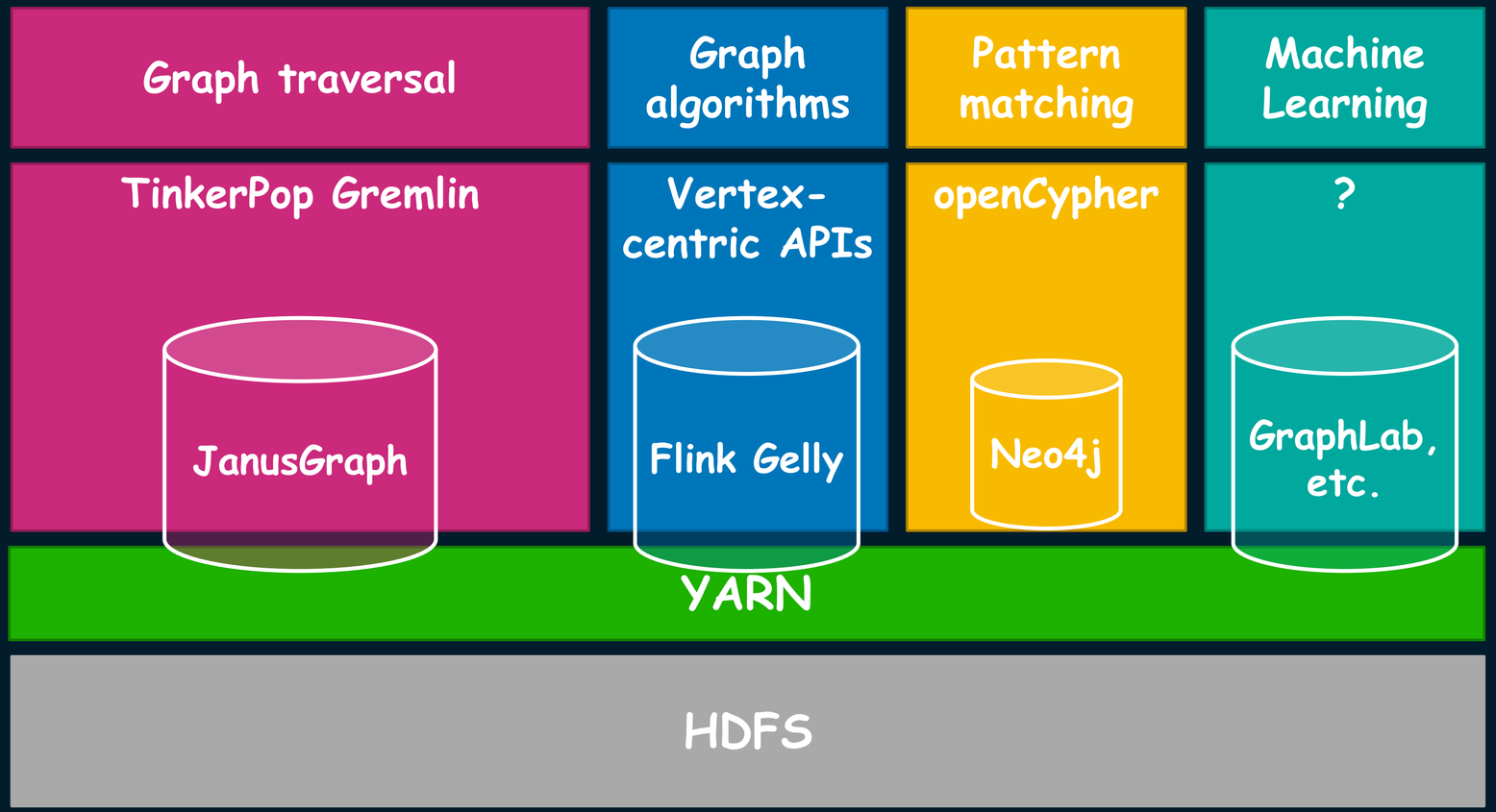


Store any graph, run any algorithms, and serve the results on-line as needed

存储任何图，使用任何分析工具和构建解决方案，甚至在线服务

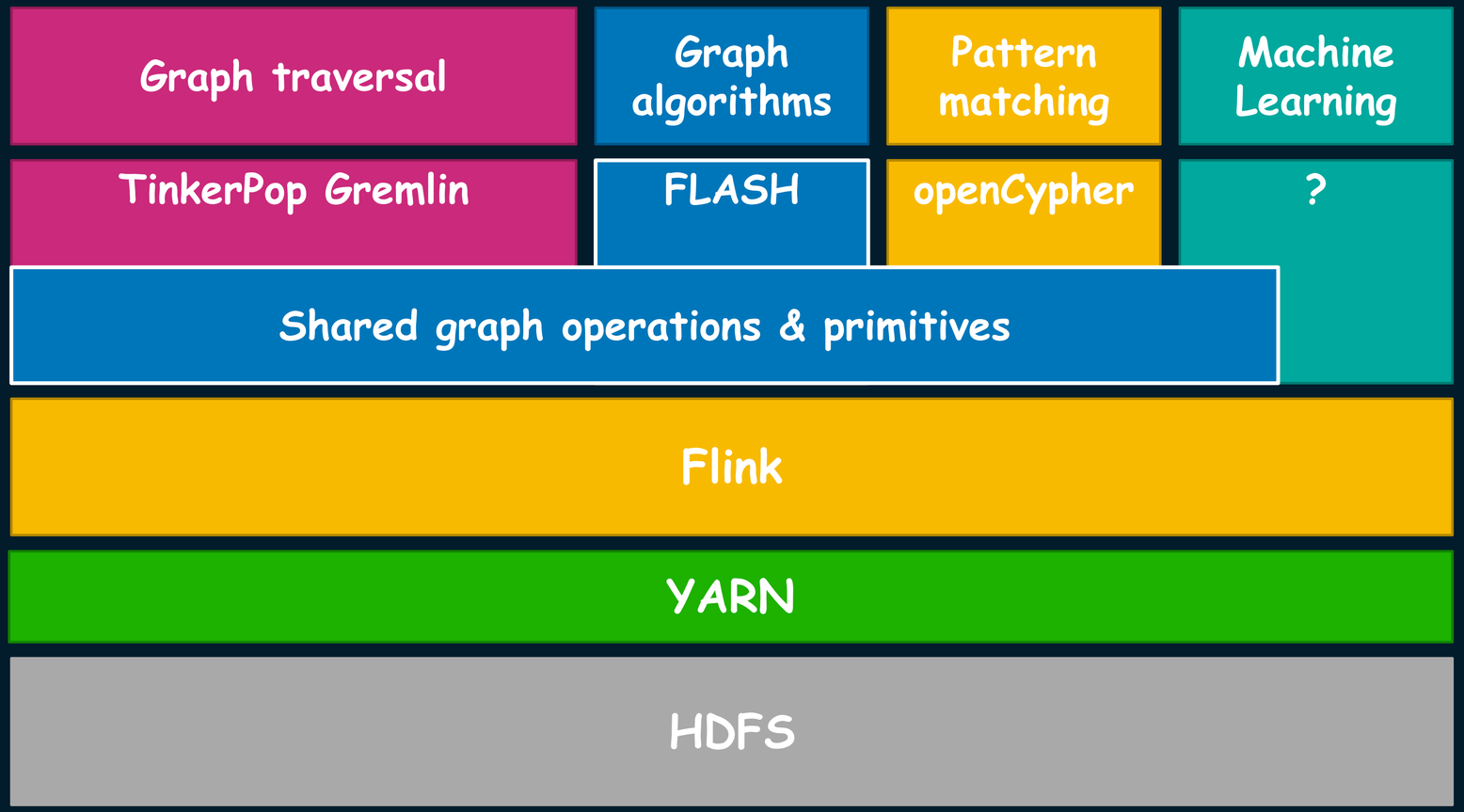
Build an extensible stack to allow “play-and-play” existing graph technologies

构建可扩展的服务体系，借力现有图技术生态



Build an extensible stack to allow “play-and-play” existing graph technologies

构建可扩展的服务体系，借力现有图技术生态



FLINK FORWARD

Graph scenarios at Alibaba

阿里图场景

Graph technologies and challenges

图处理技术与挑战

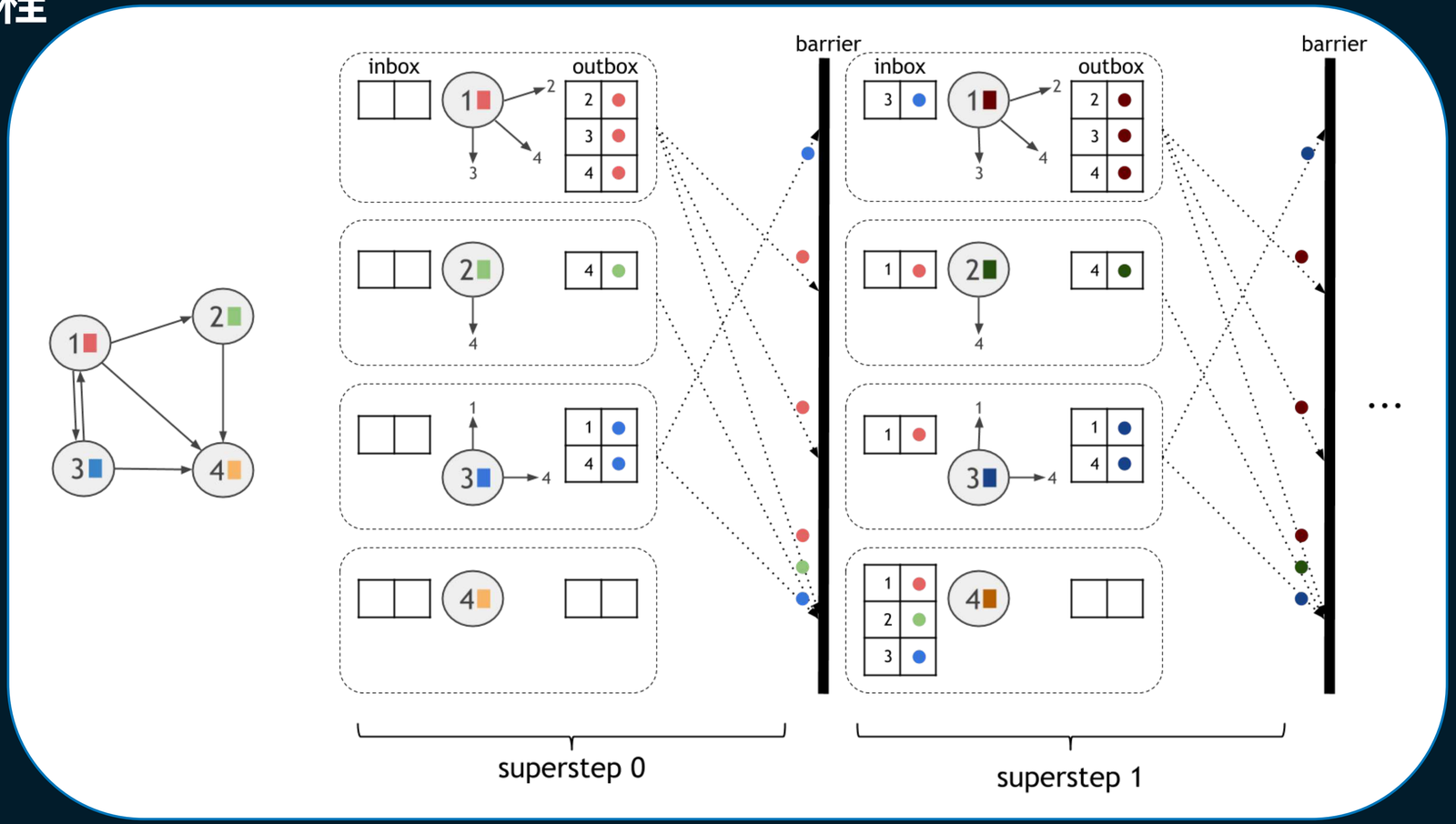
Project Flink FLASH

FLASH项目

On-going research

更多研究工作

Vertex-centric programming
面向顶点的图编程



Arijit Khan, "Vertex-Centric Graph Processing: Good, Bad, and the Ugly", in EDBT 2017

FLASH (FiLter, locAl, puSH)

FLASH (main query language)

- B = A.FlashOperator (arguments...), where A and B are both sets of vertices + control flow (e.g., while loop)

Basic Operators

- B = A.**Filter**(condition):

  Filter vertices in A which satisfy the given condition

- B = A.**Local**(operation):

  Apply some local operations (aggregation e.g.) to A

- B = A.**Push**(neighbors, attributes):

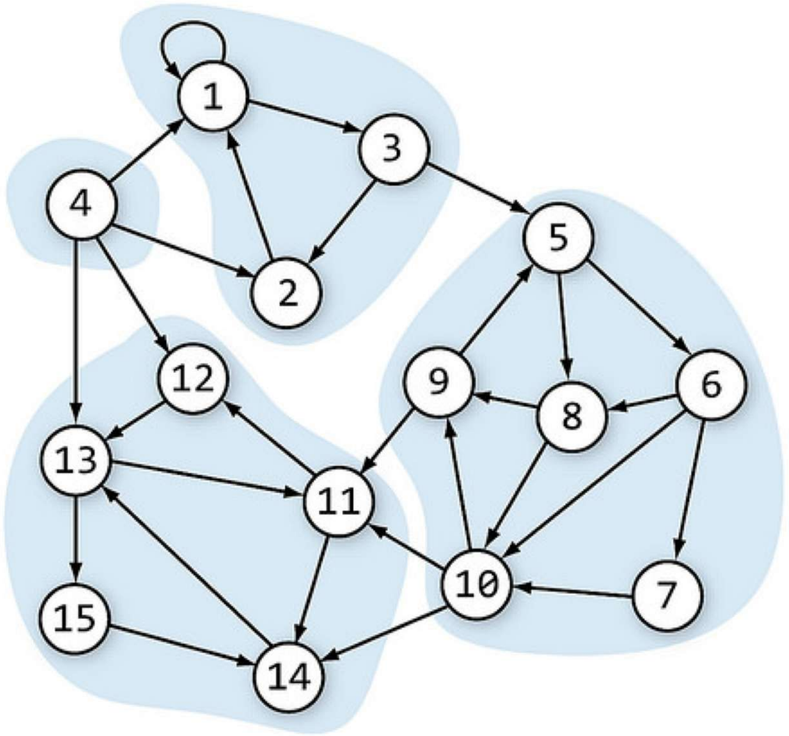
  For each vertex in A, push the given attributes to all its neighbors. The output set contains all vertices' neighbors.

Auxiliary Operators

- **Set** (Union, Intersect, etc.), **Order** (Asc, Desc, TopK) and **Group** (by Key)

An example: Connected Component (*CC*)

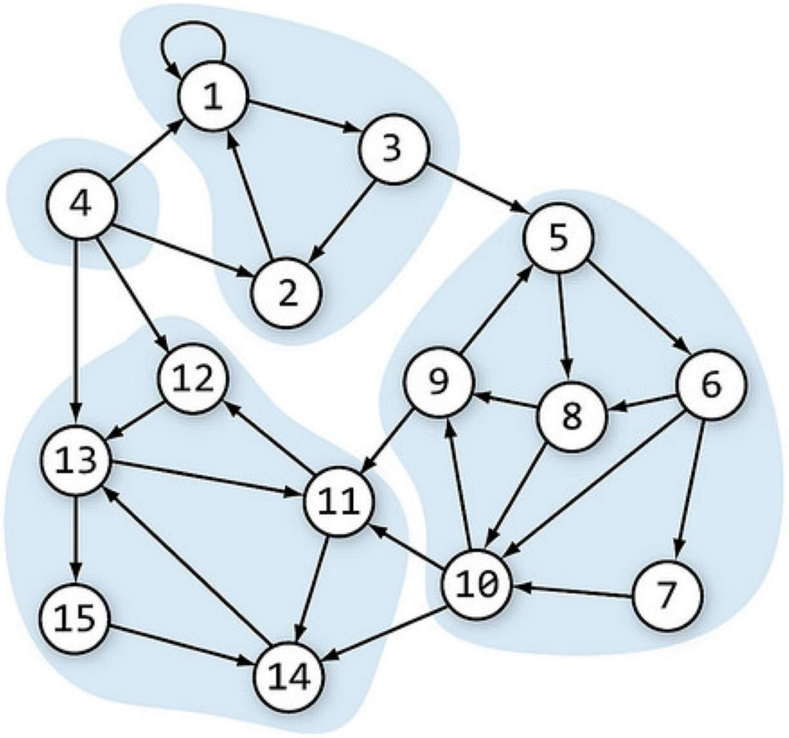
算法示例：连通分量



- BFS, DFS: $O(|V|+|E|)$ time
- 1: **for** every edge $e \in E$ **do**
  2: **if** two endpoints v and w of e are in different CCs **then**
  3: Let $\mu(v)$ and $\mu(w)$ be the component label of v and w
  4: **for** every $u \in V$ **do**
  5: **if** $\mu(u) = \mu(v)$ or $\mu(u) = \mu(w)$ **then**
  6: $\mu(u) = \min(\mu(v), \mu(w))$

An example: Connected Component (*CC*)

算法示例：连通分量



- BFS, DFS: $O(|V|+|E|)$ time
- 1: **for** every edge $e \in E$ **do**
  2: **if** two endpoints v and w of e are in different CCs **then**
  3: Let $\mu(v)$ and $\mu(w)$ be the component label of v and w
  4: **for** every $u \in V$ **do**
  5: **if** $\mu(u) = \mu(v)$ or $\mu(u) = \mu(w)$ **then**
  6: $\mu(u) = \min(\mu(v), \mu(w))$

```
A = V.local(cid = id);

while (A.size>0)
  A = A.push("adj", "cid", "preid=min(cid)")
       .filter(preid < cid).local(cid = preid);

return V.{id, cid};
```

FLASH VS vertex-centric programming
与面向顶点编程的比较 (*CC*)

```
A = V.local(cid = id);

while (A.size>0)
  A = A.push("adj", "cid", "preid=min(cid)")
       .filter(preid < cid).local(cid = preid);

return V.{id, cid};
```

```
public class ConnectedComponents<K, VV extends Comparable<VV>, EV>
        implements GraphAlgorithm<K, VV, EV, DataSet<Vertex<K, VV>>> {

    private Integer maxIterations;

    public ConnectedComponents(Integer maxIterations) {
        this.maxIterations = maxIterations;
    }

    @Override
    public DataSet<Vertex<K, VV>> run(Graph<K, VV, EV> graph) throws Exception {

        // get type information for vertex value
        TypeInformation<VV> valueTypeInfo = ((TupleTypeInfo<?>) graph.getVertices().getType()).getTypeAt(1);

        Graph<K, VV, NullValue> undirectedGraph = graph
                .mapEdges(new MapTo<>(NullValue.getInstance()))
                .getUndirected();

        return undirectedGraph.runScatterGatherIteration(
                new CCMessenger<>(valueTypeInfo),
                new CCUpdater<>(),
                maxIterations).getVertices();
    }

    public static final class CCMessenger<K, VV extends Comparable<VV>>
            extends ScatterFunction<K, VV, VV, NullValue>
            implements ResultTypeQueryable<VV> {

        private final TypeInformation<VV> typeInformation;

        public CCMessenger(TypeInformation<VV> typeInformation) {
            this.typeInformation = typeInformation;
        }

        @Override
        public void sendMessages(Vertex<K, VV> vertex) throws Exception {
            // send current minimum to neighbors
            sendMessageToAllNeighbors(vertex.getValue());
        }

        @Override
        public TypeInformation<VV> getProducedType() {
            return typeInformation;
        }
    }

    public static final class CCUpdater<K, VV extends Comparable<VV>>
            extends GatherFunction<K, VV, VV> {

        @Override
        public void updateVertex(Vertex<K, VV> vertex, MessageIterator<VV> messages) throws Exception {
            VV current = vertex.getValue();
            VV min = current;

            for (VV msg : messages) {
                if (msg.compareTo(min) < 0) {
                    min = msg;
                }
            }

            if (!min.equals(current)) {
                setNewVertexValue(min);
            }
        }
    }
}
```

How about other algorithms? Gremlin? Cypher?

别的算法也能表达？和Gremlin和Cypher语言比较呢?

https://www.uts.edu.au/staff/lu.qin

Algorithms checked
已验证可表达算法

3. Gremlin to FLASH
3.1 Between Vertices
3.2 Duplicate Edge Detection
3.3 Duplicate Vertex Detection
3.4 Recommendation
3.5 Traversal Induced Values
3.6 Lowest Common Ancestor on Trees
3.7 Maximum Depth on Trees
3.8 Pagination
3.9 If-Then Based Grouping
3.10 Connected Component
3.11 Cycle Detection
3.12 Shortest Path
3.13 Degree Centrality
3.14 Betweenness Centrality
3.15 Closeness Centrality
3.16 Eigenvector Centrality
3.17 Time-based Indexing
3.18 Determine Followers in a Follow Graph
3.19 Determine Co-workers of Softwares for Each Person
3.20 Detect Students with Conflicting Schedule
3.21 Group Neighbor Labels

4. Graph algorithms in FLASH
4.1 Induced Subgraph
4.2 Breadth-First Graph Traversal (BFS)
4.3 PageRank Computation
4.4 Graph Keyword Search
4.5 Connected Component
4.6 Single Source Shortest Path (Weighted Graph)
4.7 Core Graph Decomposition
4.8 Strongly Connected Component
4.9 Maximal Independent Set
4.10 Minimal Vertex Cover
4.11 Ego-Network Computation
4.12 Triangle Enumeration
4.13 Directed Triangle Enumeration
4.14 Clustering Coefficient Computation
4.15 Maximal Matching
4.16 Minimal Dominating Set
4.17 All-pair Shortest Path
4.18 Minimal Graph Coloring
4.19 Structural Clustering
4.20 Closeness Centrality
4.21 Betweenness Centrality
4.22 Graph Eccentricity
4.23 k-NN Friend Recommendation
4.24 Depth of Each Vertex on a Tree
4.25 Euler-Path on a Tree
4.26 Rooting an Undirected Tree
4.27 Depth-First Traversal (DFS) on a Tree
4.28 Post-order Traversal on a Tree
4.29 Number of Descendants on a Tree
4.30 Reachability Querying on a Tree
4.31 Lowest Common Ancestor Querying on a Tree
4.32 Minimum Spanning Tree
4.33 Maximal Clique Enumeration
4.34 Maximum Flow
4.35 Directed Cycle Detection
4.36 Degeneracy Graph Ordering
4.37 Topological Sort
4.38 Graph Simulation
4.39 Graph Dual-Simulation
4.40 Bipartiteness Testing
4.41 Minimum Steiner Path
4.42 Minimum Group Steiner Path
4.43 Minimum Steiner Tree
4.44 Minimum Group Steiner Tree
4.45 Undirected Depth-First Traversal (DFS)
4.46 Directed Depth-First Traversal (DFS)
4.47 Bridge Detection
4.48 Bi-connected Component
4.49 Cut Vertex Detection
4.50 HITS algorithm
4.51 Arithmetic Expression Evaluation
4.52 Tree Function Computation
4.53 Truss Graph Decomposition
4.54 Undirected Basic Cycle Computation
4.55 Personalized PageRank
4.56 k-Edge Connected Component
4.57 ECC Graph Decomposition
4.58 Steiner Component
4.59 2-Hop Labelling for Reachability
4.60 2-Hop Labelling for Shortest Path
4.61 k-Path Cover
4.62 Maximum Bipartite Matching
4.63 Maximum Weight Bipartite Matching
4.64 Stable Bipartite Matching
4.65 Tree Graph Decomposition
4.66 Core-Tree Graph Decomposition
4.67 Double k-Core
4.68 (k,r)-Core
4.69 Graph Diameter
4.70 Subgraph Matching
4.71 Subgraph Enumeration
4.72 Motif Enumeration
4.73 Ear Decomposition
4.74 Densest Subgraph
4.75 Locally Densest Subgraph
4.76 Frequent Pattern Mining
4.77 SimRank
4.78 Vertex Connectivity
4.79 k-Vertex Connected Component
4.80 Contraction Hierarchy Computation
4.81 Core Graph Hierarchy Computation
4.82 Sparse Certification Computation
4.83 Multi-Source Breadth-First Search using Bitmap
4.84 Pushbox Game using Dynamic Programming
4.85 A* based Shortest Path
4.86 Eight Number Game using A* Algorithm
4.87 Undirected Graph Cut
4.88 Gomory-Hu Tree Construction
4.89 Frequent Subgraph Mining in a Graph Database
4.90 Influence Maximization
4.91 Strong/Weak Tie Computation
4.92 Directed Minimum Spanning Tree
4.93 Shortest Path in a Temporal Graph
4.94 Semi-Clustering on a Graph
4.95 Multi-Criteria Social Graph Partitioning
4.96 Perfect Matching Testing on a Bipartite Graph
4.97 Perfect Matching on a Bipartite Graph
4.98 Minimum Vertex Cover in a Bipartite Graph
4.99 Planarity Testing
4.100 Longest Path in a Directed Acyclic Graph
4.101 Minimum k Disjoint Path Computation
4.102 Core Decomposition in an Uncertain Graph
4.103 Graph Pattern Matching (with Reachability Edges)
4.104 Eulerian Circuit Computation
4.105 Graph Summary
4.106 Graph Coarsening
4.107 Bidirectional BFS
4.108 Graph Voronoi Diagram Partitioner
4.109 Collaborative Filtering
4.110 Maximal Bipartite Matching
4.111 Matrix Factorization
4.112 Random Walk in All Vertices (Node2Vec)
4.113 k-means
4.114 Maximal bi-clique enumeration
4.115 Belief Propagation
4.116 FM Sketch
4.117 Stochastic Gradient Descent (SGD)

5. Cypher to FLASH
5.1 Query Friends
5.2 Les Personnages
5.3 UEFA Euro 2016
5.4 Winter Network
5.5 Fitness and Nutritional Recommendations
5.6 The Cantina Bar
5.7 NBA Sneakers
5.8 FIS Alpine Skiing Seasons
5.9 ClimbingDB (social network climbing database)
5.10 NHL Team Ranking Model
5.11 England Senior Squad
5.12 Pokémon X & Y
5.13 Travel Helper
5.14 NBA Playoff Prediction
5.15 Running Competition
5.16 Beer & Breweries GraphGist
5.17 OMA
5.18 Movie Recommendations with k-NN and Cosine Similarity
5.19 Interpreting Citation Patterns in Academic Publications
5.20 Organization Learning
5.21 Competency Management
5.22 Drug repurposing by hetnet relationship prediction
5.23 Roads, Nodes and Automobiles
5.24 Restaurant Recommendations
5.25 Supply Chain Management
5.26 Bank Fraud Detection
5.27 Rebels financial system
5.28 Tor Network Graph
5.29 Finding Influencers in a Social Network
5.29.1 Finding User Counts
5.29.2 Looking at forwarded messages
5.29.3 Finding Conversations
5.30 Network versions
5.31 Entitlements and Access Control

An optimized version of *CC*

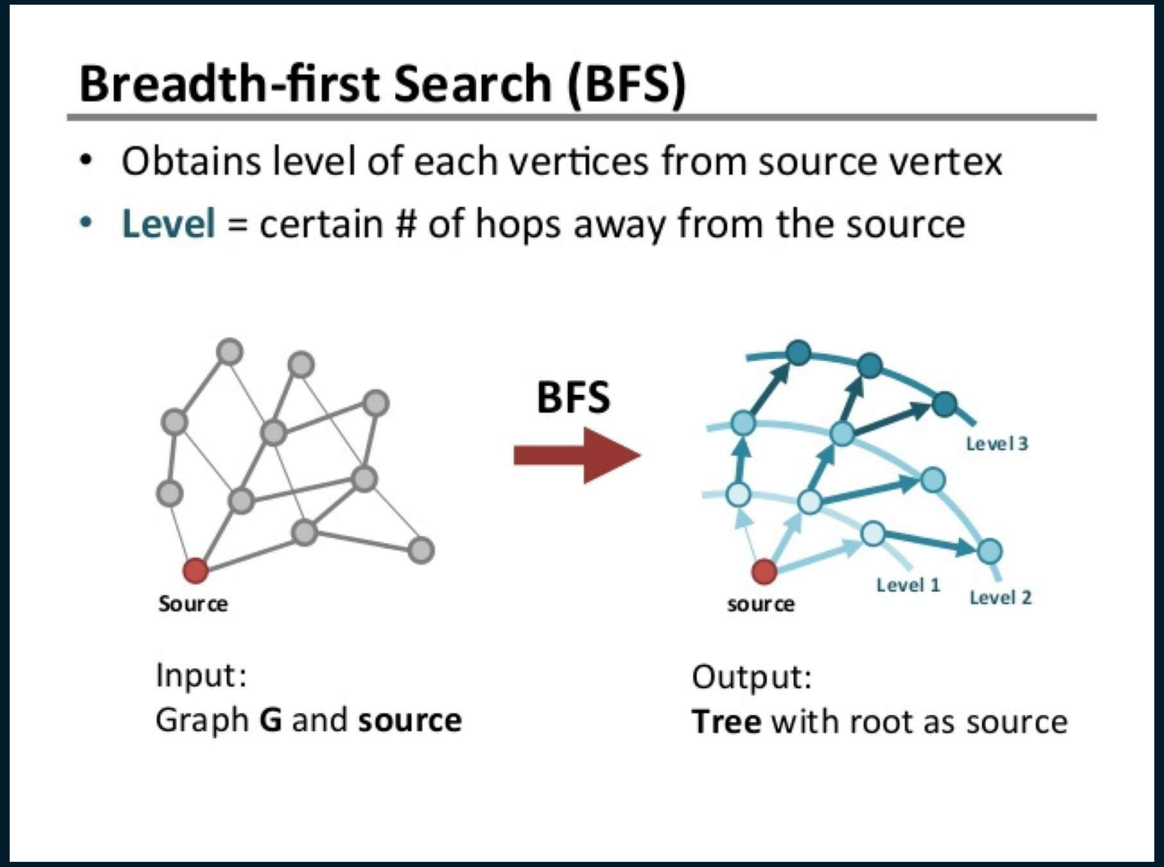
表达复杂算法优化

```
V.local(p = (min(adj,id) == id ? min(adj):min(adj,id)));
do {
    star();
    V.filter(s == 1).push("adj", "id", "f").push("f", "p", "ap").local(pm = min(p, ap))
      .push("p", "pm", "f").local(p = min(f));
    star();
    V.filter(s == 1).push("adj", "id", "f").push("f", "p", "ap").local(pm = min(ap(it != p)))
      .filter(pm != null).push("p", "pm", "f").local(p = min(f));
    A = V.push("p", "id", "f").push("f", "p", "g").filter(p != g).local(p = g);
} while(A.size > 0)
return V.{id, p};

procedure star() {
    V.local(s = 1).push("p", "id", "f").push("f", "p", "g").filter(p==g).local(s = 0).push("g").local(s = 0);
    V.push("p", "id", "f").filter(s == 0).push("f").local(s = 0);
}
```

System optimizations made possible: example of push/pull

允许更多的系统优化：push/pull示例

```
BFS(s)

A = V.filter(id == s).local(dis = 0);
while(A.size > 0) {
    A = A.push("out", "dis", "predis")
          .filter(!hasAttr("dis"))
          .local(dis = min(predis) + 1);
}
return V.{id, dis};
```

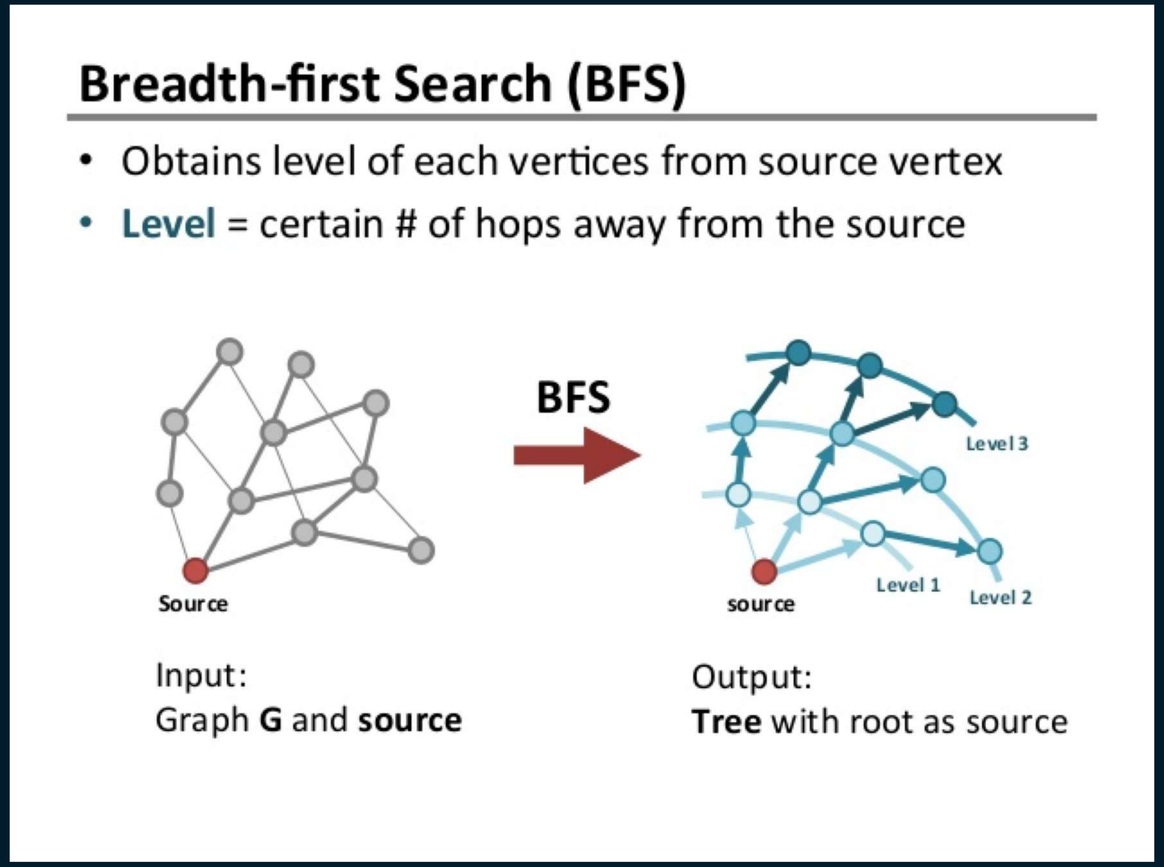


https://www.slideshare.net/yuichiroyasui/graph500-bigdata2013

System optimizations made possible: example of push/pull

允许更多的系统优化：push/pull示例

```
BFS(s)

A = V.filter(id == s).local(dis = 0);
while(A.size > 0) {
    if (A.size < threshold) {
        // TODO: switch to pull mode
        continue;
    }
   A = A.push("out", "dis", "predis")
        .filter(!hasAttr("dis"))
        .local(dis = min(predis) + 1);
}
return V.{id, dis};
```



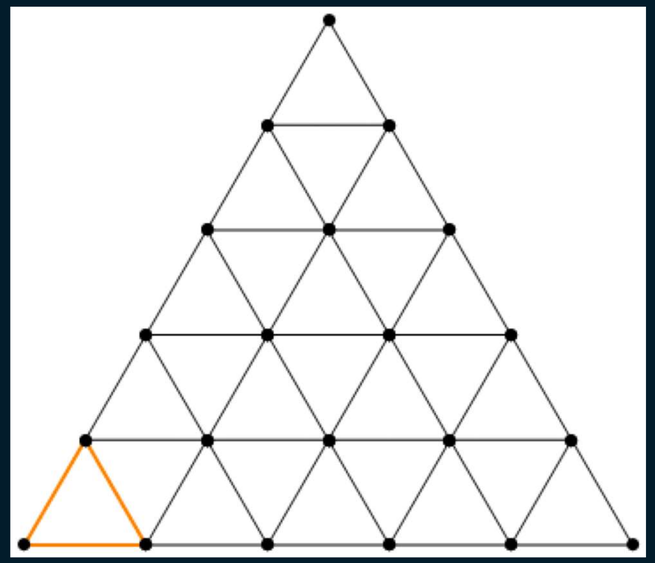
https://www.slideshare.net/yuichiroyasui/graph500-bigdata2013

System optimizations made possible: example of (vertex) reordering

允许更多的系统优化：(顶点)顺序优化示例

Triangle enumeration

```
A = V.local(t = both).local(t = append(t,id))
     .push(both, t, tList)
     .local(t = append(tList,id))
     .local(t=filter(t, it.0<it.1 && it.1 < it.2 && contain(both, it.0));
return {t};
```

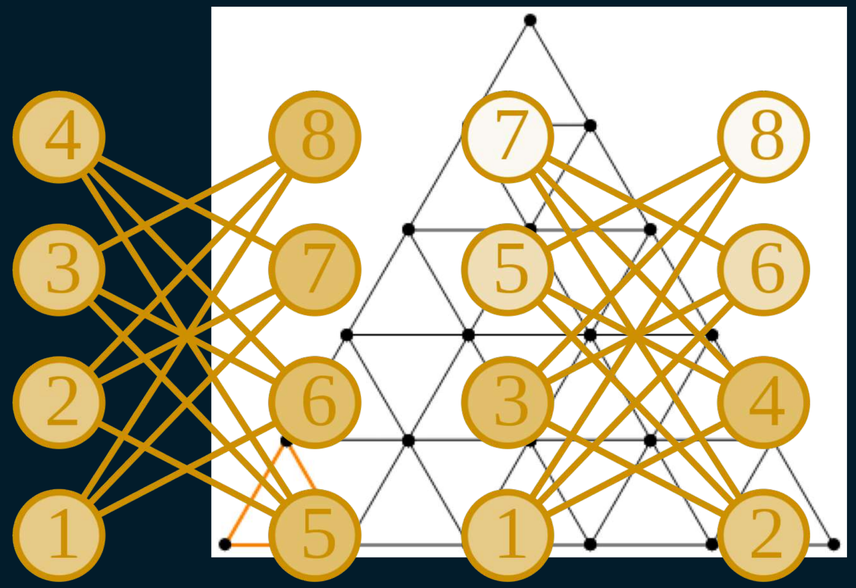


System optimizations made possible: example of (vertex) reordering

允许更多的系统优化：(顶点)顺序优化示例

Triangle enumeration

```
A = V.local(t = filter(both,it.id > id)) // t.0>t.1
    .local(t = append(t,id))
    .local(tout = filter(both, id > it. id)) // t.1>t.2
    .push(tout, t, tList)
    .local(t = append(tList,id))
    .local(t=filter(contain(both, it.0));
return {t};
```



https://en.wikipedia.org/wiki/Greedy_coloring

Preliminary evaluation (VS Flink Gelly)

与Flink Gelly进行初步的易编程/性能比较

Algorithms	LOCs (Gelly)	LOCs (FLASH)	Soc (Gelly)	Soc (FLASH)	Twitter (Gelly)	Twitter (FLASH)	UK (Gelly)	UK (FLASH)	Friendster (Gelly)	Friendster (FLASH)
CC	52	8	-	-	9m49s	9m10s	45m57s	49m55s	-	-
SSSP	53	7	-	-	5m31s	5m34s	13m13s	13m14s	-	-
PageRank	51	7	-	-	11m13s	11m22s	31m20s	31m29s	-	-
Community Detection	93	11	-	-	27m48s	26m19s	1h24m	1h41m	-	-
Triangle Counting	194	5	-	-	-	-	-	-	45m19s	55m28s
Jaccard Similarity	299	6	7m11s	10m43s	-	-	-	-	-	-

Environment: 16 nodes with 20GB memory and 4 cores each

FLINK FORWARD

Graph scenarios at Alibaba

阿里图场景

Graph technologies and challenges

图处理技术与挑战

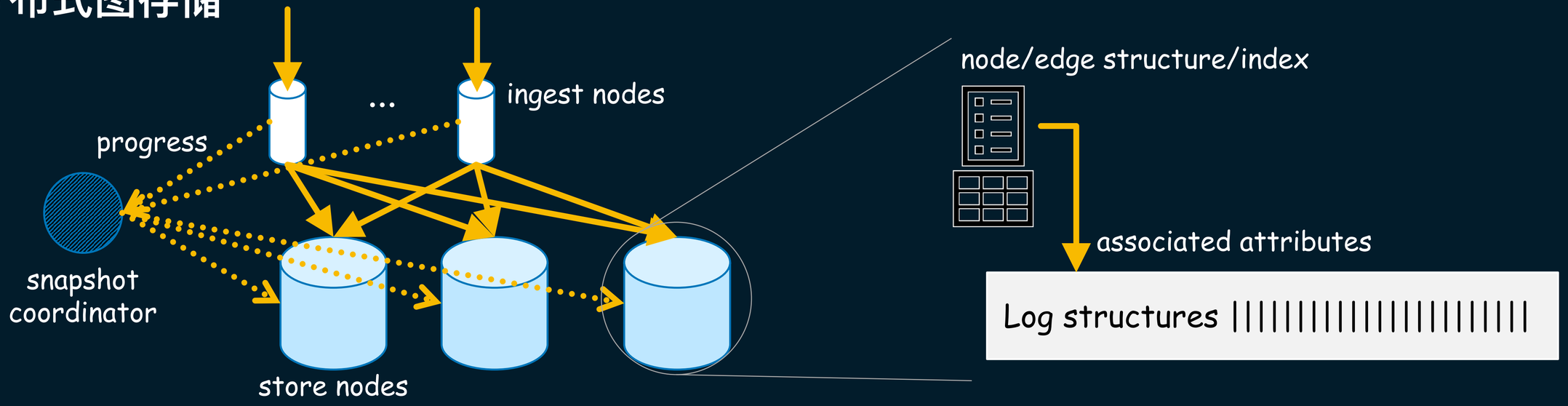
Project Flink FLASH

FLASH项目

On-going research

更多研究工作

Distributed graph store
分布式图存储



Key design choices 将图分区存放到分布式节点，支持快照隔离和多副本容错

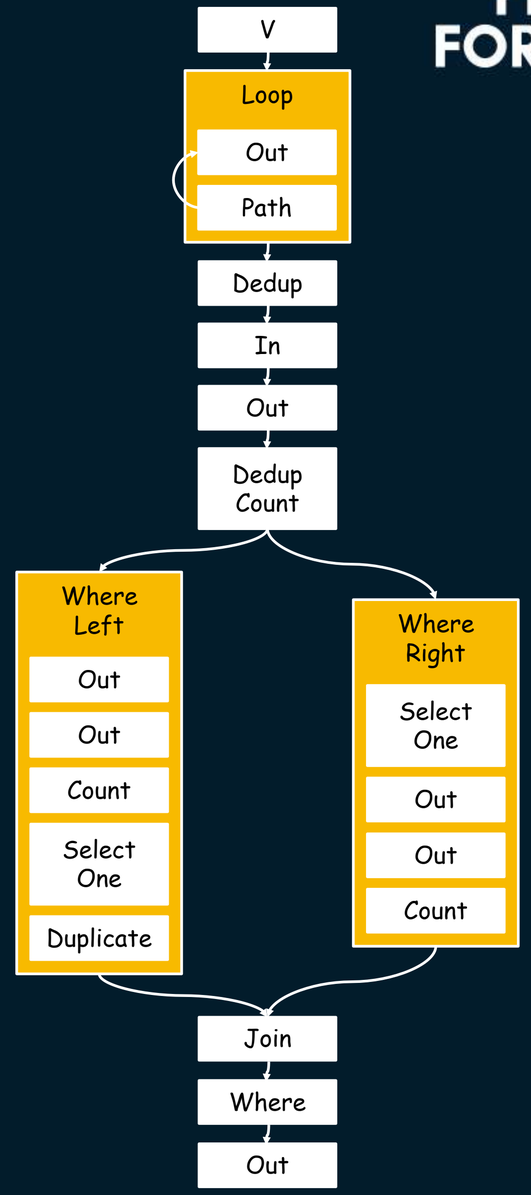
- *Partitioning graph data across distributed memory/SSDs*
- *Real-time updating with snapshot isolation*
- *Fault tolerance using replication*

Distributed and parallel graph traversal
分布式并行图遍历

Many graph operations can be expressed as graph traversals
许多图操作可以用遍历实现

- An example query in Gremlin (from LDBC Social Network Benchmark)

```
g.V("2146078220").as("root")
 .repeat(out("person_knows_person").simplePath()).emit().times(2).dedup().as("friends")
 .in("post_hascreator_person").as("post")
 .out("post_hastag_tag").as("tag")
 .where("tag", neq("root"))
 .by(out("person_knows_person").out("person_hasinterest_tag").count())
 .by(out("person_knows_person").out("person_hasinterest_tag").count())
 .select("friends", "post")
```



Compiling Gremlin to Flink dataflow by (re)using FLASH primitives
将Gremlin查询翻译成FLASH原语/数据流

- Precompiles operators to minimize scheduling overhead

Concluding remarks
小结/展望

http://www.hykleen.co.uk/contract_services.php

- Graph analytics is the next disruption in big data processing
- 我相信，有效的图数据分析和应用能够带来业务革新

- Better tools are still needed!
- 今天的工具和系统（面对复杂图场景）依然不够好！

- Driven by real scenarios (at Alibaba), there are great opportunities in redesigning key and higher-level abstractions for graph data processing
- 在真实场景的驱动下，我们有机会重新思考和设计更好的系统抽象


THANKS