认识图计算和图数据库

什么是图

图（Graph）是一种非常直观表达事物及其关联关系的数据结构，基本元素是“点”和“边”，点表示一个事物，边就表示他们之间具有一定关系

比如下面这张图，它的点有公司、员工、项目，边即他们之间的关系——公司和员工之间是雇佣关系、员工和员工之间可以有好友关系、项目和员工之间也可以有参与关系。也就是说我们可以用图的方式来把事物和它们的关系抽象出来。

这是一张结构比较简单的图。随着点和边不断增加，图数据越来越多越来越复杂，逐渐形成更丰富的网状结构。比如一些金融交易图，它的规模可能会非常大，超过 10 亿个点，有千亿甚至万亿边。可以想象，要真正处理这些图还是很有挑战的。
为什么需要图计算

把计算机想象成大脑，我们要解决两个关键问题：数据的存储和分析。

传统的数据存储采用关系型数据库，其结构是“表结构”（想象一下 Excel 表格）。比如一家银行的客户转账信息可能包含交易人员、交易金额、交易时间，记录在一张表格里。假如交易人 A 和 B 之间有直接转账关系，这种直接关系用关系型数据库是不难发现的。但对于非直接关系，关系型数据库就较难“穿透”多个点来发现了，即便可以处理，查询速度也可能非常慢。我们用信用卡套现来举例。

首先是简单的直接套现模式。如左侧图显示，一个人办了一张信用卡，他其实不是真的想去还款，他找了一个商店，这个商店提供一个非法的服务就是信用卡套现。那么他通过信用卡付款，把 2020 元钱转到这个商店。这个商店直接就把其中的 2000 元钱返回给付款人，就完成了一次套现。这样的一种套现是非常简单的，我们通过对这个个体，对这个商店的收款记录和付款记录做分析，就可以识别出套现行为。

但右边这张图就复杂了很多。我们可以看到，右上角的这个人，他还是通过信用卡付款，付了 2020 元钱给了商店。这个时候，商店没有直接把钱退给付款的人，而是由一个个人付了 2000 元钱给到一个第三人。这个个人和商店之间，我们可以通过一些分析发现，他实际上拥有这个商店，所以我们把这种关系叫做同人关系。就是店和人虽然看起来是不同的实体，但其实他们之间有一个非常强的关联。那么他付款给的第三人也不是最开始刷卡的人，而是刷卡人的一个亲友，店主付款到了刷卡人亲友的银行卡上。那这样的一个套现模式就比左边的复杂很多了。我们把这种模式叫做多跳闭环模式。

要分析这种多跳闭环模式，就需要找出复杂的关联关系，而不能只对这个个体进行分析。但是大家可能会说，你画的这张图很简单呀，我一眼就能看出来，这是一个闭环，这个坏人我很快就能抓住。

但实际生活中情况可能会更加复杂，有更多其它交易和关系，就没那么容易看出来了。如下图所示，右边这张图可能会有千亿条甚至万亿条边，怎么很快地在这个图

上把环找出来，这就对整个分析技术——复杂的关联分析技术提出了非常高的要求，性能成为了关键。

什么时候要用图计算

随着数据量和深度的增加，如果我们用传统的关系数据库的方法去分析的话，那就可能非常非常慢，难以在有效的时间内计算出结果。而图计算技术直接将事物与其关系像制作地图一样定位存储下来，直接支撑对事物和关系的各种查询和计算——这与我们大脑对信息的处理模式很像，大脑本身也可以建模成一个图。由于提供了对关联数据最直接的表达，以及图模型对异构数据天然的包容力，可以很好的解决目前遇到的关联数据分析问题。

由此可见，关系型数据库的设计擅长回答“已知”的问题，而图数据库可以回答超出设想的“未知”问题。相较于关系型数据库，图数据库是真正注重“关系”的数据库。

我们刚才举的是金融方面的例子，但是图计算的用途远远不限于金融行业。在互联网、工业领域、医药、公共卫生、公共安全等领域都有很多的应用。如绘制用户社交关系图谱进行社交影响力排名、好友推荐；构建设备关系网络图谱实现物联网建模分析、供电网络建模分析等。

随着互联网和 5G 时代数据指数级增长，数据之间的关系越来越复杂，企业管理和分析数据面临更高难度。越来越多的企业管理者们开始关注以图为代表的技术来更智能地使用数据，Google、Facebook 等科技巨头也早就在通过图数据库的力量来支撑主要业务应用。

国际知名咨询公司 Gartner，每年都会发布各种技术趋势的报告。在 2021 年的《十大数据分析技术趋势》报告中，Gartner 提到了“Graph relates everything”。这是一个非常有趣的双关，即图连接万物，一方面表示了图的本质，就是把各种事物连起来，另一方面也表达了图会在数据分析的各个领域得到广泛应用。Gartner 预测到 2025 年，图技术在数据和分析创新中的占比将从 2021 年的 10%上升到 80%。

蚂蚁集团开源图数据库 TuGraph，成立图计算开源委员会

9 月 1 日，2022 世界人工智能大会“新一代图智能技术发展与实践论坛”上，蚂蚁集团图计算负责人陈文光宣布开源蚂蚁集团高性能图数据库 TuGraph 单机版，并成立图计算开源技术委员会，中国工程院院士郑纬民、陈纯分别担任主席、副主席，5 位业界知名专家担任委员。
TuGraph 由蚂蚁集团和清华大学共同研发，是图数据库权威测试世界纪录保持者，也是世界上有测试纪录的“最快”的图数据库。

随着 TuGraph 的开源，图数据领域将迎来一款性能卓越、功能丰富、生态完备的开

源产品。

开发者可以聚焦应用层，轻松打造属于自己的图数据，从而提升行业整体技术应用水位。TuGraph 开源采用 Apache2.0 协议，在 Github 和 Gitee 上进行托管。

图数据库区别于关系型数据库，基于图模型，使用点边来表示、存储、处理数据，拥有灵活的数据抽象模型，能够更好地表达出“关系”的概念。

蚂蚁 TuGraph 是一套分布式图数据库系统，可以支持万亿级边上的实时查询。此次开源的 TuGraph 单机版，同样具备完备的图数据库基础功能和成熟的产品设计，可以轻松支持 TB 级别数据和百亿级别大图，足以满足大多数业务场景需求。相较于市场上常见的开源产品，TuGraph 单机版的性能高 10 倍以上。

蚂蚁集团 2015 年开始自主研发分布式图数据库、流式图计算等图相关技术，2016 年发布自研分布式图数据库，并应用于支付宝。至今 TuGraph 已应用于蚂蚁内部 150 多个场景，包括在线支付的实时链路，以支付宝风险识别能力提升近 10 倍、风险审理分析效率提升 90%的成绩，验证了其高可靠性。

就在上个月，LDBC（关联数据基准委员会）发布最新图数据库 SNB 测试结果，TuGraph 在功能完整性、吞吐率、响应速度等层面全球领先。

目前，蚂蚁集团已形成了一套以图数据库为底座、同时包含流式图计算，离线图学习的大规模图计算系统。

蚂蚁集团图数据库负责人洪春涛表示，图技术是未来大数据、人工智能和高性能计算产业发展的关键所在，它很有可能会成为下一代的数据底座。蚂蚁集团愿意通过开源持续输出核心技术优势，推动图数据库更广泛的应用生态，携手行业抢占技术高地，不断探索技术的可能性。

技术解读 | TuGraph 图分析引擎技术剖析

图分析引擎又称图计算框架，主要用来进行复杂图分析，是一种能够全量数据集运行快速循环迭代的技术，适用场景包括社区发现、基因序列预测、重要性排名等，典型算法有 PageRank、WCC、BFS、LPA、SSSP。
TuGraph 图数据管理平台社区版已于 2022 年 9 月在 Github 开源，本文将对 TuGraph 图分析引擎的技术进行剖析。

1 TuGraph 图分析引擎概览
TuGraph 的图分析引擎，面向的场景主要是全图/全量数据分析类的任务。借助 TuGraph 的 C++ 图分析引擎 API ，用户可以对不同数据来源的图数据快速导出一个待处理的复杂子图，然后在该子图上运行诸如 BFS、PageRank、LPA、WCC 等迭代式图算法，最后根据运行结果做出相应的对策。 在 TuGraph 中，导出和计算

过程均可以通过在内存中并行处理的方式进行加速，从而达到近乎实时的处理分析，和传统方法相比，即避免了数据导出落盘的开销，又能使用紧凑的图数据结构获得计算的理想性能。

根据数据来源及实现不同，可分为 Procedure、Embed 和 Standalone 三种运行模式。其中 Procedure 模式和 Embed 模式的数据源是图存储中加载图数据，分别适用于 Client/Server 部署，以及服务端直接调用，后者多用于调试。

Standalone 模式的数据源是 TXT、二进制、ODPS 文件等外部数据源，能够独立于图数据存储直接运行分析算法。

TuGraph 图计算系统社区版内置 6 个基础算法，商业版内置了共 34 种算法。涵盖了图结构、社区发现、路径查询、重要性分析、模式挖掘和关联性分析的六大类常用方法，可以满足多种业务场景需要，因此用户几乎不需要自己实现具体的图计算过程。

算法类型	中文算法名	英文算法名	程序名
路径查询	广度优先搜索	Breadth-First Search	bfs
	单源最短路径	Single-Source Shortest Path	sssp
	全对最短路径	All-Pair Shortest Path	apsp
	多源最短路径	Multiple-source Shortest Paths	mssp
	两点间最短路径	Single-Pair Shortest Path	spsp
	网页排序	Pagerank	pagerank
	介数中心度	Betweenness Centrality	bc
	置信度传播	Belief Propagation	bp
	距离中心度	Closeness Centrality	clce

重要性分析	个性化网页排序	Personalized PageRank	ppr
	带权重的网页排序	Weighted Pagerank Algorithm	wpagerank
	信任指数排名	Trustrank	trustrank
	sybil检测算法	Sybil Rank	sybilrank
	超链接主题搜索	Hyperlink-Induced Topic Search	hits
关联性分析	平均集聚系数	Local Clustering Coefficient	lcc
	共同邻居	Common Neighborhood	cn
	度数关联度	Degree Correlation	dc
	杰卡德系数	Jaccard Index	ji
图结构	直径估计	Dimension Estimation	de
	K核算法	K-core	kcore
	k阶团计数算法	Kcliques	kcliques
	k阶桁架计数算法	Ktruss	ktruss
	最大独立集算法	Maximal independent set	mis
	弱连通分量	Weakly Connected Components	wcc

社区发现	标签传播	Label Propagation Algorithm	lpa
	EgoNet算法	EgoNet	en
	鲁汶社区发现	Louvain	louvain
	强连通分量	Strongly Connected Components	scc
	监听标签传播	Speaker-listener Label Propagation Algorithm	slpa
	莱顿算法	Leiden	leiden
	带权重的标签传播	Weighted Label Propagation Algorithm	wlpa
模式挖掘	三角计数	Triangle Counting	triangle
	子图匹配算法	Subgraph Isomorphism	subgraph_isomorphism
	模式匹配算法	Motif	motif

表1.1 TuGraph内置算法

2 功能介绍
2.1 图分析框架 图分析框架作为图分析引擎的“骨架”，可以联合多种模块有效的耦合协同工作。一般分为预处理、算法过程、结果分析三个阶段。

预处理部分用于读入数据及参数进行图构建及相关信息的存储统计，并整理出算法过程所需的参数及数据。

算法过程会根据得到的数据通过特定的算法进行逻辑计算，并得到结果数据。 结果分析部分根据得到的结果数据进行个性化处理（如取最值等），并将重要的信息写回和打印输出操作。

2.2 点边筛选器 点边筛选器作用于图分析引擎中的 Procedure 和 Embed 模式。对于图存储数据源可根据用户需要和实际业务场景对图数据进行筛查，选择有效的点

边进行图结构的构建。 2.3 一致性快照 TuGraph 中的 Procedure 和 Embed 模式能够提供数据“快照”，即建立一个对指定数据集的完全可用拷贝，该拷贝包括相应数据在某个时间点（拷贝开始的时间点）的镜像。由于 OLAP 的操作仅涉及读操作而不涉及写操作，OlapOnDB 会以一种更紧凑的方式对数据进行排布，在节省空间的同时，提高数据访问的局部性。 2.4 块状读写模块 块状读写模块作用于图分析引擎中的 Standalone 模式，用于对不同外部数据源的数据进行高效读入，同时也包含对内部算法处理后的图数据结果写回。 2.5 参数模块 参数模块作用于分析引擎中的 Standalone 模式，用于对图的一般信息（如数据来源，算法名称，数据输入、输出路径，顶点个数等）以及根据不同数据来源、不同算法所配置的不同信息参数进行接受和整理，传输给图算法及各个模块，同时将最终结果模块化展示。

3 使用示例
由前文所述可知，图分析引擎分为 Standalone、Embed 和 Procedure 模式，现在以 bfs 算法为例分别介绍他们的使用方式。 3.1 Procedure 模式 Procedure 模式主要用于 Client/Sever 的 TuGraph 运行时，图算法的加载和调用。 在 TuGraph/plugins 目录下执行 bash make_so.sh bfs 即可在 TuGraph/plugins 目录下的到 bfs.so 文件，将该文件以插件形式上传至 TuGraph-web，输入参数后即可执行。 示例： 在 TuGraph/plugins 编译.so 算法文件

bash make_so.sh bfs

将 bfs.so 文件以插件形式加载至 TuGraph-web 后，输入如下 json 参数：

即可得到返回结果。
输出内容解释： num_edges: 表示该图数据的边数量 num_vertices: 表示该图数据顶点的数量 prepare_cost: 表示预处理阶段所需要的时间。预处理阶段的工作：加载参数、图数据加载、索引初始化等。 core_cost: 表示算法运行所需要的时间。 found_vertices: 表示查找到顶点的个数。 output_cost: 表示算法结果写回 db 所需要的时间。 total_cost: 表示执行该算法整体运行时间。 3.2 Embed 模式 该种方式主要用于 TuGraph 在后台程序中对预加载的图存储数据进行算法分析，多用于快速调试。在 TuGraph/plugins 目录下对 embed_main.cpp 文件完善，补充数据名称、输入参数、数据路径等信息，示例如下：
保存后在 TuGraph/plugins 目录下执行 bash make_so.sh bfs 即可在 TuGraph/plugins/cpp 目录下的到 bfs_procedure 文件，bash make_embed.sh bfs

在 TuGraph/plugins 文件夹下执行./cpp/bfs_procedure 即可得到返回结果。

3.3 Standalone 模式 Standalone 模式可以独立于图存储运行，直接从文本文件或 ODPS 读取 Edgelist 形式的图数据。在 TuGraph/build 目录下执行 make bfs_standalone 即可得到 bfs_standalone 文件,该文件生成与 TuGraph/build/output/algo 文件夹下。运行：在 TuGraph/build 目录下执行./output/algo/

bfs_standalone --type [type] --input_dir [input_dir] --vertices [vertices] --root [root] --output_dir [output_dir]

- [type]：表示输入图文件的类型来源，包含 text 文本文件、BINARY_FILE 二进制文件和 ODPS 源。
- [input_dir]：表示输入图文件的文件夹路径，文件夹下可包含一个或多个输入文件。TuGraph 在读取输入文件时会读取[input_dir]下的所有文件，要求[input_dir]下只能包含输入文件，不能包含其它文件。参数不可省略。
- [vertices]：表示图的顶点个数，为 0 时表示用户希望系统自动识别顶点数量；为非零值时表示用户希望自定义顶点个数，要求用户自定义顶点个数需大于最大的顶点 ID。参数可省略，默认值为 0。
- [root]：表示进行 bfs 的起始顶点 id。参数不可省略。
- [output_dir]：表示输出数据保存的文件夹路径，将输出内容保存至该文件中，参数不可省略。

示例：在 TuGraph/build 编译 standalone 算法程序

在 TuGraph/build/output 目录下运行 text 源文件

得到运行结果：

结果参数解释同上。

4 小结

综上，图分析引擎可以高效、快速的处理多种来源的数据，其并行的图构建方式保证了内存占用小的特点。此外，图分析引擎也具有易于安装部署、灵活性高、耦合程度低、易于上手等对用户友好特性，可以帮助用户结合具体业务解决问题。

TuGraph开源JAVA客户端工具TuGraph-OGM，无缝对接JAVA开发生态

TuGraph 图数据库提供了 JAVA、C++、Python 等多种语言的 SDK 支持，方便客户在各种场景下使用。用户使用 SDK 向TuGraph服务器发送Cypher请求，服务器则以 JSON形式返回数据。近日，TuGraph 推出了一款面向 JAVA 客户端用户的开发工具 TuGraph-OGM (Object Graph Mapping)，为用户提供了对象操作接口，相较 Cypher/JSON 接口应用起来更加便捷。

OGM 类似于关系数据库中的 ORM（Object Relational Model），可以将数据库返回的数据自动映射成 JAVA 中的对象，方便用户读取，而用户对这些对象的更新操作也可以被自动翻译成 Cypher 语句发送给服务器。这样即便是完全不懂 Cypher 的用户，也可以通过操作对象与数据库进行交互，大大降低了图数据库的使用门槛。

TuGraph-OGM 同时也兼容其他开源产品 OGM 工具如 Neo4j-OGM，方便用户将工程在不同数据库与 TuGraph数据库间无缝迁移。本文将对 TuGraph-OGM 进行全面的介绍。

0 映射原理
TuGraph-OGM 将 JAVA 对象映射为图的对象，类映射为点，类的属性映射为图中的属性，类中的方法映射为操作 TuGraph 的查询语句。

以电影场景为例，对演员、电影、导演之间的关系进行数据化，就形成了非常典型的图数据。举一个简单的示例，演员Alice在1990年和2019年分别出演了两部电影《Jokes》和《Speed》，其中《Jokes》的导演是Frank Darabont。

以图的思维来看，演员、导演、电影可以被映射为三种不同的节点，而出演、执导可以被映射为两种边，映射结果如上图所示，将数据存入图数据库后，相关的开发人员就可以使用各类图查询语言对数据进行查询。

但对非图数据库相关的开发人员来说，这个例子中的演员、导演、电影作为实体，同样可以映射为类中的对象，而与实体相关联的对象可以通过集合存储，这是大多数开发人员熟悉的领域。

1 TuGraph-OGM架构
TuGraph-OGM可被看做一个"翻译器"，主要功能是将开发人员对JAVA对象的一些操作翻译为TuGraph可理解的图查询语言Cypher请求，并将该操作返回的结果，再次翻译为JAVA对象。架构图如下所示：

3 使用示例
详细示例请参考tugraph-ogm-integration-tests 在pom.xml中引入依赖

3.1 构建图对象

首先需要通过注解标明图中的实体。

@NodeEntity：该注解标明的类为节点类。

@Relationship：用于标明边，同时@Relationship中可指定label与边的指向。

@Id：用于标明identity，是OGM中数据的唯一标识。

3.2 与TuGraph建立连接

当前TuGraph-OGM提供了RPC driver用于连接TuGraph，具体配置如下所示：

3.3 通过OGM进行增删改查

OGM支持对TuGraph的实体执行CRUD 操作，同时支持发送任意TuGraph支持的Cypher语句，包括通过CALL调用存储过程。

CREATE

在完成图对象的构建后，即可通过类的实例化创建节点，当两个节点互相存储在对方的集合（该集合在构建时被标注为边）中，就形成了一条边，最后使用session.save方法将数据存入数据库。

注意：TuGraph数据库为强schema类型数据库，在创建实体前需要该数据的label已经存在，且新建过程中需要提供唯一的主键。

DELETE

使用session.delete方法删除节点，同时会删除与节点相关联的所有边。

UPDATE

修改已创建的节点的属性，再次调用session.save方法会对节点进行更新。

MATCH

session.load方法用于根据节点id查找节点。 session.loadALL方法用于批量查找节点，支持通过多个节点id查找节点、查找某一类型的所有节点、带有filter的查询。filter查询需要新建Filter，传入参数ComparisonOperatorx0;可选为：EQUALSx0;、GREATER_THANx0;、LESS_THAN

QUERY WITH CYPHER

OGM支持通过queryForObject、query方法向TuGraph发送Cypher查询，由于Cypher查询的灵活性，需要用户自行指定返回结果格式。

session.queryForObject方法：需要在方法第一个参数处指定返回类型，可设定为某一实体类或数字类型。

session.query方法：Cypher查询的返回结果被存储为Result类型，其内部数据需要用户自行解析，以下方代码为例，传入数据库的Cypher为CREATE查询，返回结果createResult可被解析为QueryStatistics，可获取到此次查询被创建的节点与边的数目。

关于TuGraph

高性能图数据库 TuGraph（https://github.com/TuGraph-family/tugraph-db） 由蚂蚁集团和清华大学共同研发，经国际图数据库基准性能权威测试，是 LDBC-SNB 世界纪录保持者，在功能完整性、吞吐率、响应时间等技术指标均达到全球领先水平，为用户管理和分析复杂关联数据提供了高效易用可靠的平台。

历经蚂蚁万亿级业务的实际场景锤炼，TuGraph 已应用于蚂蚁内部150多个场景，助力支付宝2021年资产损失率小于亿分之0.98。关联数据爆炸性增长对图计算高效处理提出迫切需求，TuGraph 已被成熟应用于金融风控、设备管理等内外部应用，适用于金融、工业、互联网、社交、电信、政务等领域的关系数据管理和分析挖掘。

2022年9月，TuGraph 单机版开源，提供了完备的图数据库基础功能和成熟的产品设计，拥有完整的事务支持和丰富的系统特性，单机可部署，使用成本低，支持TB级别的数据规模。

蚂蚁高性能图数据库 TuGraph-DB 的技术思考与实践

在近日举行的 DTCC 2022 第十三届中国数据库技术大会-图数据技术与应用创新专场，蚂蚁集团图数据库负责人洪春涛博士分享了蚂蚁高性能图数据库 TuGraph-DB 的技术思考和实践，以下为演讲内容要点回顾。

图计算的优势

图计算是一种高效的抽象计算方法，可以方便地处理复杂的多维数据。我们将员工和公司之间的关系画成图，这样可以快速查询员工的信息，例如员工的工作情况、与其他员工的关系以及与哪些公司有联系。相比之下，在关系数据库中，我们需要分别建立员工信息表、公司信息表以及员工和公司之间的关系表。这就把整个数据切成了很多张二维表，在实际应用系统中经常能够看到几百上千张表。这样会使得系统变得更加复杂，需要基于多张表去做推断，对人或机器来说都会带来挑战。

对比来看什么是简单查询和复杂查询：

图中表格的前两行属于比较简单的查询，比如某个员工的工龄，直接找到对应的那一行然后取出来就可以；比如找出所有的雇员数大于 5 的公司，可能在雇佣关系表中就能找出来，在关系数据库里这些都属于常规的操作。

但如果查询再复杂一点，我想知道员工 A 和员工 C 之间有什么关系，不一定是一跳的关系，可能包含在同一家公司工作，也可能包含参与了同一个项目，甚至还包含他们有一个共同好友，这些关系就是多种多样的，想用 SQL 列举所有可能的关系其实就没有那么容易了。如果确定知道这里可能有几种关系，还能靠 SQL 穷举出来，但随着表越来越多，里面的关系可能有几百上千的时候，再想去穷举就非常难了。

最后还有一种更复杂的查询，比如想找员工 A 和员工 E 的所有关系，可能包括 A 认识 B，B 认识 C，C 认识 E，相当于在做一个不定长跳数的查询，在 SQL 里面就非常难写了，相信绝大多数人都难以写出这种查询。

业界有句话，所谓的关系数据库实际上并不擅长处理关系。例如，员工 A 和 E 之间的所有关系实际上就是我们要查询的关系，但在关系数据库中处理这种关系查询实

际上是不够友好的。这是图数据库的优势所在，它们更擅长处理复杂的关系数据。我们发现，大部分浅显的数据信息都可以比较容易地挖掘出来，但要想更深入地利用数据产生更多的价值，就必须挖掘更深层次的信息，这时就一定会遇到这些复杂的关系，这也是为什么图数据库越来越受欢迎的原因。

为什么图数据库开始流行

图数据库的概念最早出现在七八十年代初期，但当时为什么人们没有选择使用图数据库，而是选择了关系数据库呢？我认为主要原因是，当时的计算机并不那么强大，使用二维表格形式的关系数据库对计算机来说更友好。我们知道，计算机最擅长的事情就是重复的劳动，重复的任务。如果我们要在一张二维表中找一行数据，我们可以一行一行查找，或者使用二分查找（如果数据是树状有序的）。每一层都是重复的算法在运行，这对计算机来说是一个非常规整的数据，容易处理。但如果把数据抽象成一张图，那么难度就会大很多，对计算机来说会更难处理。

举个例子，一个员工可能与其他人有各种不同的关系，比如朋友关系、雇佣关系、参与项目关系，每种关系的类型都不同，对应的数据也都不同。此外，一个点上的边数也非常不规律，有的人可能只有几个好友，而在微博上，一个普通人可能有两三百个粉丝，而大 V 账号可能有数百万甚至上亿的粉丝。这样一来，存储数据时，普通人和大 V 之间的差距就非常大了，对计算机来说，处理这两种数据的差异也会很大。

我们知道，在七八十年代，计算机相对来说很弱，如果那个时候使用图来表示数据，整个处理和查询的难度就会大很多。所以，人们选择使用关系数据表的形式表示数据。事实上，那个时候有人做过图数据库，但并没有流行起来，原因就是性能相对关系数据库来说差得太多了。

我们知道，所有的事物，尤其是电脑，一直在不断进步。这包括硬件和软件方面的改进。如今的硬件和几十年前的硬件完全不是一个概念，而现在的软件优化也大不相同。随着这些改进，我们会发现对当前的电脑和计算机系统来说，使用图数据库带来的额外开销可能不是很大的问题。

举个例子，我们会发现，过去人们都使用低级编程语言编写程序，但随着时间的推移，有了高级语言。比如最开始、最原始的电脑可能是用纸袋和机器码写程序，后来有了汇编，再后来有 C 语言、C++，现在很多人都直接写 Python。虽然 Python 程序的执行速度可能较慢，但是写的很快，而用用 C++或者汇编去写得写半天，对于编写程序到最终得出结果的整个过程来说，使用 Python 会更方便。

计算机编程语言的发展是从低级向高级演变的，数据抽象也是一样。我们认为，未来的数据抽象一定会对人更友好一些，而不是专注于对机器更友好。如图所示，编程语言的发展是从低级语言向高级语言转变的，我们也认为数据抽象层次也会慢慢从低层次表格抽象向高层次图表抽象发展。

图计算在蚂蚁的应用

自 2015 年开始，蚂蚁实际上投入了大量资源来研究图计算，研究如何在蚂蚁的业务中使用图计算。例如数据血缘应用。在对业务的处理过程中，我们需要较好地追

踪这些数据的流转路径，如果修改了一份源数据会对下游数据产生什么影响，会对最终业务产生什么影响。为了更好地追踪，我们使用图数据库来存储数据。

另一个比较有趣的应用场景就是程序分析。相信几乎所有互联网公司内部都有大量的程序，因此，我们需要管理这些程序，并在每次提交代码时了解将会对哪些内容产生影响。为此，蚂蚁负责程序分析的团队会分析这里的图数据。例如，定义一个变量 A，然后使用变量 A，“定义”与“使用”之间就会有一条边，使用关系会存储在图数据库中。目前我们的图中已经有超过 200 亿条边，这是一个非常大的数据量。我们需要对这些数据进行存储、查询和分析，这是蚂蚁公司内部非常多的图数据场景之一。

举个例子来说明优惠券反套现的场景：满额返券是一种比较常见的促销方式，比如购物满 2000 元就可以享受 100 元的优惠。这种情况下，如果正常消费，用户花费 2000 元，通过返券省下 100 元。但是有些人会想办法注册假商铺，进行虚假交易，目的是把平台补贴的优惠券套出来。因此当用户去买东西，平台要去付补贴的过程，我们需要去实时检测一下会不会有可疑的资金交易情况。

蚂蚁有很多业务需要研究图计算系统和图数据库等技术来满足需求，因为这些业务需要对大量的点边进行分析，数据量超过了 100TB，基本上已经达到了 PB 级别。我们需要对这些图进行实时查询，吞吐率大约在百万级别。由于需要对用户的付款进行实时判断，所以需要比较低的延迟，大约在 20 毫秒的级别。如果延迟太长，会导致用户体验很差，比如付款需要等待 5 秒才能完成，这样就会非常麻烦。

图计算系统建设中的问题与挑战

在建立蚂蚁图计算系统的过程中，我们遇到了各种各样的问题。为了解决这些问题，我们与学术界和许多研究界的同事一起合作，并发表了许多相关的学术论文，包括 EuroSys 等。然而，我们在建立系统的过程中发现，目前的图计算仍处于较早期的阶段，因此许多标准尚未成形。这对我们来说是一个棘手的问题。例如，在关系型数据库中，查询语言基本上就是 SQL，但在图数据库中，仅查询语言就有许多种，包括 Gremlin、G-SQL 等等。这导致了市场的碎片化，人们学习和使用的成本也很高。

在建立图计算系统的过程中，我们也遇到了许多挑战。为了分担较大的通信量，需要将图数据分布到多台机器上，但这会导致边的信息在不同机器之间传递，造成大量的通信。此外，单次查询所涉及的数据量也比较大，例如五跳查询涉及的点数就已达到 10 的五次方，图中还存在一些非常大的点。同时，用户对图计算系统的需求也十分多样，既有快速查询的需求，也有对复杂算法（如社区发现）的需求，单一系统很难满足这些不同的需求。

TuGraph 技术优势

蚂蚁自己开发了一套图计算系统 TuGraph，既能解决图数据的存储问题，也能解决流式计算、离线计算和图学习的问题。目前，超过 100 个业务线和 300 多个场景都在使用这套系统。这套系统在 2021 年获得了世界互联网大会领先科技成果奖。

在 TuGraph 中，性能是一个重要的因素，因为图数据集的体积很大，如果性能不佳

就会浪费机器资源，导致许多情况下无法完成任务。比如，希望业务的查询能在几十毫秒内返回结果，但是如果做的性能不好，几秒钟才能返回结果，就无法作为在线查询使用。因此，我们是非常对性能是很重视的，其中在 LDBC-SNB 标准测试中（类似于数据库领域性能标准测试 TPC-C），TuGraph 仍然是世界纪录的保持者。

TuGraph 的整个图存储是建立在完美哈希的基础上的，这是我们与其他图系统的一个重要区别。目前，大多数图系统使用的是基于数的存储，但数的问题在于永远存在一个 LogN 的查找操作。然而，在图中可以看到，不同的顶点之间实际上是无序的，不需要有顺序，所以顶点这个级别实际上是基于哈希的，理论上，顶点的读取是最优的。

此外，TuGraph 还参与了许多标准的定制，整个系统在尽量往标准化的方向去做。

除了为内部提供服务，我们还向外提供服务，主要是因为，作为一个系统，如果只为有限的客户提供服务，就很容易构建成一个专有系统。我们希望这是一个标准化、开放的系统，所以我们也在对外提供图计算系统的产品和服务。目前，我们也有很多外部客户，包括金融、工业、互联网以及政企领域。

开源开放，共建发展

整个图计算系统目前仍处于较早期的阶段，我们认为还有很多工作要做，包括提升应用性、性能和降低成本。所有的系统都会有这些问题。但是，如果希望普及，我们认为最重要的是有健康的生态，来推动图计算系统的发展，需要有更多的用户和更多的场景使用这个系统。

所有的计算机系统都需要去有一个更开放、更大的生态才能促进发展。蚂蚁有一句话叫做“成熟一个、开放一个”，一个系统成熟以后，我们就会试着开放出去，让更多的人去用。今年 9 月，我们已经在 GitHub 上开源了 TuGraph 中的单机版图数据库，以及一个离线图分析引擎 TuGraph Compute。分布式图数据库和流式图计算现在已经包含在我们的商业化版本中，包括一站式图研发平台。我们计划在未来迭代更多更丰富的系统功能，希望能做得更好。

TuGraph 开源版特色

为什么要去开源单机版而不是分布式版本？主要是考虑到它的部署和使用成本比分布式版本要低得多，同时功能也很完整、独立。我们希望这样可以让许多刚开始使用图数据库或有使用图数据库解决问题的想法的人，可以先尝试用我们的单机版图数据库。因为它的部署非常简单，如果跑起来没有问题，那么再考虑是否需要分布式版本。如果确实需要，我们可以再跟进这个问题。

我们的单机版图数据库已经能够支持 TB 级别的数据，我们内部也有很多情况使用单机版图数据库。在单台机器上，我们最大的数据量也达到了 2TB 多，在线上运行，能够处理百亿级别的点边。事实上，大多数用户使用单机版图数据库都是足够的。由于单机版的图数据库很容易优化，我们对它进行了极致的优化，因此单机版图数据库在性能上可以满足绝大多数场景的需求。此外，它的系统特性也很全面，包括高可用性、多图支持、权限管理、日志记录等，它可以被看作是一个成熟、易用的图数据库，类似于 MySQL。

图中所列出的开源版 TuGraph 几个特性包括：

- 单机版图数据库能够处理数据量几个 TB 的数据，前提是磁盘足够大。
- 成本很低，因为是单机版，部署和运维都很容易。
- 性能很好，我们对其进行了大量优化。TuGraph 的 LDBC-SNB 测试目前是世界第一，大家可以在 GitHub 上获取测试 SNB 的条款并进行测试。
- 单机版图数据库是一个非常易用的完整系统，我们提供了导入导出工具和查询语言。此外，还提供了底层 API，用户可以使用它来编写复杂的程序。

我们的开源版本的目标主要有三点：

首先，我们希望提供一个免费的图数据库产品，能够让更多的人使用图数据库，尝试用它来解决问题。

其次，我们希望促进图数据库标准的成形。目前图数据库的差异太大，每个数据库都有所不同，我们希望通过提供一个参考答案来帮助大家达成趋同。这样大家就可以根据我们提供的设计来判断哪些特征合理，如果觉得合理就可以遵循这个设计，慢慢地大家就会逐渐靠近。假如所有产品在主要特征上保持一致，这样所有人的学习成本就会降低。

最后，基础研究性问题可以不断优化发展，包括存储方面的问题，例如哈希可能是理论上最优的，但是是否还有其他需要调整的东西？目前没有一个很好的研究性平台让大家去进行这些尝试和研究，我们希望提供的开源 TuGraph-DB 能成为这些研究人员的对比基线，促进研究的发展。

TuGraph 企业版特色

除了开源版本，我们也继续提供商业版本。这个版本包含一个分布式图数据库，以及离线计算引擎和流式图计算功能。此外，我们还提供了 TuGraph Platform 一站式图平台，包括运维、可视化等功能。在这个平台上，用户可以在图数据库中执行流式计算，并在线写回数据库。这种方式通常用于实时查询结果，因为流式计算的时间可能比较长，但用户可以立即查询到较早的结果。这对于在线业务来说非常重要。

商业化产品还提供私有化部署，也可以通过一体机的方式部署硬件，并将很快推出云上部署方案，这样大家就可以在云上体验我们的产品。

总结

蚂蚁在图计算方面投入了大量资源，并在众多业务场景中磨练出了一整套在线查询、流式计算、离线分析以及图学习的体系。目前，我们已经在 GitHub 上开源了单机版（https://github.com/TuGraph-family），同时也提供企业版来满足不同用户需求。

陈文光：AI时代需要怎样的数据处理技术?

大家好，我是清华大学/蚂蚁技术研究院陈文光，今天为大家带来《AI 时代的数据处理技术》主题分享。

我们身处一个以信息技术为核心驱动力的大数据时代。从下面这张图，我们可以看出，数据量和数据生成的速度在飞速增长。与此同时，新的产生数据的形式在产生，数据模态也在不断增长，不仅包括自然语言，还有声音、图像、视频等等多种形式。最近非常流行的多模态大模型包括具身智能、触觉等新的数据形态。
丰富的数据形态要求我们要对数据做有效处理。模仿“马斯洛需求层次理论”，数据处理也有一个层次，从最底下的收集数据、存储数据到做一般的数据查询处理，更上面的层次是现在越来越接近于用 AI 方式处理数据，甚至最后还能生成很多内容。

1.AI时代数据处理新需求
这样的大模型崛起的时代也引发了对数据处理的新需求。最近，Meta 出了一个新模型 LLaMA-3，效果非常好，它实现了在十几万亿的 Token 上面做的训练，而我们之前很多模型可能只是在 4T 的或者几个 T 的 Token上面做训练。

那么，如何获得增加的这部分 Token？实际上，这需要从很多网上低质量的数据中做大量的数据处理，清洗出来可用的高质量数据，如果想让大模型的能力进一步增长，实际上需要数据处理做很多的工作。

另一方面，大模型直接应用在生产服务场景下，本身还存在很多缺陷，比如幻觉问题、上下文长度的问题。目前的多数超长上下文大模型并不能完整记录真正领域的知识。为了满足需求，向量数据库和大语言模型结合起来，提供高质量的服务。
从数据服务的角度来讲，向量数据库是一种使用嵌入的方式表达知识，再用另外索引的方式快速找到相应知识的方式，它和大模型配合才能获得很好的效果。所以大模型的发展和崛起，对数据库领域也提出了很多新需求。

2.ai时代数据库发展趋势
在这样的趋势下，我今天想分享三个观点，也是未来的数据库面临的三个比较重要的发展趋势：
（一）在线离线一体化

这张图是企业常见的在线、离线两个链路。

- 上面是在线链路，一个数据请求会先经过预处理，再通过训练好的模型做推理，比如风控、分类等等，再把结果反馈到 KV 里，直接服务用户的请求。
- 下面是离线链路，收到数据请求后，我们要想办法处理，去更新模型。经过一段时间后再把模型更新到在线链路上。

在线、离线两个链路分开，在生产中会遇到一些比较严重的问题，主要就是在线、离线不一致。

比如在离线链路上做了各种仿真模拟，但是当把策略、模型上传到在线链路时，会

出现与离线链路仿真效果不一样的情况。造成这种现象的根本原因，就是我们通过不同的数据链路把新数据接入进来，离线链路处理的数据与在线链路不一致。

怎么解决这个问题？最好的解决方案，就是只有一套数据。如果能够做到一个系统既能够做事务处理，可以支持事务在数据上面原子化地做更新，同时还可以在这一批数据上做后续分析型的业务。也就是说用一份数据给在线离线链路的一致性打下基础。

这里面也有非常多的技术问题。一般来说，事务处理行存会比较好，分析一般是列存比较好，当一套系统需要同时支持行存和列存的时候，需要什么样的存储结构？另外，事务处理对于优先级、延迟/尾延迟、吞吐率要求比较高。那么在系统里如何调度不同优先级的请求，这里涉及到很多相关技术。

在蚂蚁的图风控中，也有另外一个场景。刚才我们讲到，可以通过数据库本身的 HTAP 引擎解决在线/离线一致性的问题，如果没有这样的混合系统，应该如何实现两份数据达到在线离线一致性？下面以图风控方案中的在线离线一体化为例，给大家介绍。

TuGraph DB（分布式图数据库），是一个支持 事物处理 的图数据库。TuGraph Dataflow （流图计算系统），可以看作是一个支持图语义的 Flink。

在我们原来的方案中，这两个系统采用不同的查询语言，一个是我们自定义的 GQuery 语言，另外一个是基于 Java 的支持 Gremlin 语言。这两个系统的数据通过 TuGraph Dataflow 处理完成后，一条线通过 TuGraph DB 去做在线链路，另外一个经过存储去完成后续的离线分析，这时就会出现数据不一致的情况。

这个问题如何解决呢？首先需要让数据保持一致性。数据虽然是两份，但是在 TuGraph DB 和存储之间新增了一条数据同步链路，就是通过从 Binlog 中读取数据，保证两份数据的一致性，防止出现两边写，一边写成功、一边写失败，而导致的数据不一致。当把在线数据里已经处理完成的数据同步至离线数据，这时数据的一致性是有保证的。

另外，我们把这两个系统的查询语言和语义都统一起来，都使用国际标准图查询语言 ISO-GQL，同样一套查询语言在两个系统上用同一个语义支持，在进行后续的策略分析时，数据和查询语言的语义是一致的，可以达到更好的一致性。

这里也存在非常复杂的情况。图数据库的基本功能是从一个点扩出去很多点，但是有些点的邻居非常多，可能有几十万、上百万个，所以我们会限制每个点扩展的点数，比如只扩两百个。但同时还需要在两个系统中保证不仅只扩两百个点，这两百个点都是一样的，才能保证数据一致性。所以想要在两个系统中要保证数据一致性，需要花费相当大的精力。

（二）向量数据库和关系型数据库一体化

向量数据库和大语言模型的结合有非常重要的作用，如果一个企业要用大语言模型做服务，既要部署语言模型又要部署向量数据库，同时企业的很多数据又保存在关系型数据库中。

这样一个多系统复杂混合的部署，开发、部署、维护非常困难。因为涉及到多个系统之间的依赖性，软件版本、系统之间的交互也会存在很多的问题。如果能够把这些功能做到一起，就能够实现一致性的管理。

在关系型数据库中，可以通过一些插件支持向量数据库的语义，同时在调用查询引擎的时候，将数据分到不同的链路上执行，从用户的角度就可以实现只部署一个系统，使用一套语言，完成相关工作。

蚂蚁集团有一套内部的 VSAG 的向量库，实现了主流向量数据库的相关功能，而且在实际生产中已经得到应用。向量数据库最有名的是 FaceBook 开源的 FAISS 系统。

VSAG 和 FAISS 之间有什么区别？FAISS 功能非常强大、性能非常好，对 GPU 也做了很多优化，但是相对来说提供了很多底层功能，这就需要通过调整各种参数、配置，从中得到一个对应用比较合适的配置。

而蚂蚁集团的 VSAG 库更多从开发者和产品应用性的角度出发，默认把很多基础配置的事情都做好了，而且在 CPU 上也实现了很多优化，提供了近似于开箱即用的功能。
在 OceanBase 里，以插件的方式集成了 VSAG 功能，可以在 OceanBase 里使用 VSAG 向量化的功能，用一套系统达到这样的效果。
（三）数据处理与 AI 计算一体化

有人可能会问，数据处理不就是 SQL 吗？AI 是神经网络层面的东西，AI 与 SQL 为什么会结合到一起？我举一个例子。大家知道世界上有很多的网页，网页上面有很多内容，内容量非常大，远超几十 T、几百 T。但是在这些海量内容中，很多内容质量很低，如何从中提取出高质量的内容？FaceBook 提出了一套 CCNet 的流程，下图的 CCNet 流程展示了数据处理和 AI 的模型在这一过程中的融合试用。
第一步， CCNet 对网页的原始数据进行解析，在 HTML 的网页中抓取内容，这里涉及到解析等工作。

第二步是删冗，删冗也可以被认为是一个 JOIN，因为抓取网页内容中可能用到了别的网页内容，语料里面有冗余不利于最后的训练，即对每段话都做一个哈希，和过去已有的内容对比，是相同还是不相同。解析与删冗是非常典型的数据处理过程。

第三步，做语言分类，需要经过神经网络模型判断网页的语言。

接下来，通过一些 AI 模型对数据做分词、质量评估，后面的过滤、分桶工作，又回到数据处理。

在这个应用里，数据处理和 AI 计算处于交叠的状态，不是一次数据处理之后都交给AI 完成后续的处理，这是一个复杂的来回交互的链路。

那这种情况下，什么样的系统可以支撑这样复杂交互的服务？现在的 AI 和大数据生态基本是割裂的生态：

- AI 用 Python，主要用 GPU ；
- 大数据基本上是用 CPU ，用基于 Java 的 Spark 实现。

另一方面，在很多小数据的处理上，Python 已经展现出非常强大的性能，像 Pandas 这样的系统，在单机数据的处理上提供了非常方便的接口。

当 AI 逐渐成为主流计算形态的时候，数据应该如何与 AI 融合?

由于 Spark 是基于 Java 的生态，当我们如果把大模型处理交给 Spark 去做，它产出的结果要通过文件系统、或者其他传输方式交给 AI 的 Python 程序，Python 处理完之后可能还有一些后续处理。在刚才的例子里面，数据处理和 AI 计算之间会有多次的交互，对整个系统的开发、调试、部署、维护带来非常大的问题。

有人尝试把数据处理和 AI 结合起来。2019 年，英特尔出了一个系统“BigDL”，在 Spark 里面把神经网络的描述、优化器、训练方式把这些东西加进去。当时只支持了 CPU，而且是基于 Java 的。我们可以认为，这种方式是试图把 AI 融入到大数据的生态里面。

我们反过来看，如何把大数据的生态往 AI 的方向牵引？这其实是 Spark 的 Python 化。

上图是 2022 年在DataBricks Summit上讲的。这是一个分布式的 PySpark，就是 Python 接口的 Spark系统。当时 PySpark 的使用率已经达到了整个 Spark 使用率的近 50%，很多人已经愿意用 PySpark 了。但是 PySpark 还存在一个问题：它的性能很差。

Python 是一个动态语言，在编译时不知道它的类型，动态时才知道，所以它的性能很差，比 Java 的 Spark 还要慢一半。所以虽然 PySpark 对编程非常友好，很多人也习惯用，但是性能不太好。因此我们在处理大量数据的时候，希望能够避免这一问题。

所以，我们提出一个愿景，融合数据处理和 AI 生态。

我认为还是要基于 Python，因为 AI 是主要的计算形式，所以整个数据处理应该围绕 AI 建设。从编译优化的角度来讲，我们希望把 PySpark 做很多的优化。这件事是可以做的，我们最近也有了一些成果。在删冗部分，通过把 PySpark 做相关优化，基本上性能可以提升一倍多，可以达到我们的性能预期。

未来，这个生态不只是编程要融合，底层的硬件也要融合，数据和 AI 结合以后，底层的硬件生态也要支持 GPU、弹性任务调度，最后可以达到“一次编写到处执行”的效果。

3月30日，TuGraph 社区 Meetup “图数据库智能化建设与探索”在北京顺利举办，探讨大模型时代下的图数据库智能化。请看精彩回顾 👇

01. 技术分享 | TuGraph 计算引擎模型推理系统设计与优化
“虽然传统的基于迭代的算法能够解决我们现实生活中的大多数问题，但随着业务需求的不断发展和现实问题的日渐复杂化，这些算法往往难以满足某些具体的需求。

尤其是当规模和维度日益增长、数据越发密集时，我们很难利用这种传统的方法去提取到更加关键的一些信息，或者说是我们从人的视角上更难理解的一些信息。因此，由于图结构在表达能力上的优势，结合机器学习分析技术，图算法近年来引起了广泛关注，并在业界落地和取得了较好的商业价值。”

02. 技术分享｜TuGraph-DB兼容Neo4j客户端：Bolt协议设计与实现
“兼容Neo4j客户端的最大优势在于生态支持。以客户端为例，Neo4j官方自身支持五种编程语言的客户端，社区又贡献了两种，共计七种语言的客户端得以直接使用。此外，还有一系列与上下游生态相接的组件，如与Apache Spark或Apache Kafka的连接，都有现成的代码可供利用。在编程框架方面，特别是Java，例如OGM（Object-Graph Mapping，对象图映射）以及一些业务开发框架，如Spring，这些所需的相关代码都已现成，无需重新编写。这种做法极大地节约了研发资源，我们可以将这些资源重新投入到提升数据库本身能力上。”

03. 技术分享｜知识图谱语义框架SPG及图谱推理
“当前，我们正处于图谱技术发展的第三阶段，这一阶段的核心是将图谱与大型模型相结合。目标转向了知识的标准化、跨领域数据的联通与复用。随着这个阶段的深入，简单地在推理过程中融入文本概念和信息，或者是加入交易与社交的实体关系，已经不能明显提升推理效果了。关键的做法应当是结合实体信息的多元素特征进行深度协作，从而更精准地关联相关性，揭示那些稀疏的实体间关系，并实现意义解释的密集化。”

04. 技术分享｜CStore Compaction模块的设计与优化
“TuGraph Analytics本质上是一款图分析OLAP数据库。CStore作为一个单机版存储引擎，提供了坚实的存储基础。同时，RocksDB也可以作为TuGraph Analytics的存储基础。我们采用LDBC提供的通用社交网络图数据集进行了基准测试，测试涉及让TuGraph Analytics分别连结RocksDB以及我们自有版本的CStore进行分析。在同步与异步compaction（数据压缩整理）两种方式下进行了读写性能测试：同步方式意味着数据写入完成后进行compaction，完成之后再进行读性能测试；异步方式则是写入和compaction同时进行，写入完成后立即测试读性能。在这两种情境下，使用CStore的TuGraph Analytics的读性能超过了使用RocksDB的三倍以上。”

05. 社区规划｜TuGraph 社区技术路线

最后是展望未来环节，TuGraph 开源负责人范志东与大家分享了大模型时代的图计算要做些什么，包括Q2即将推出的开源数据分析工具 OSGraph，Q3即将开源的 TuGraph 研发平台 TuGraphMaker，结合大模型的“与图对话”工具 ChatTuGraph 等项目。
06. 圆桌讨论｜图技术、图生态、图智能在自由讨论环节，TuGraph 布道师戚仕鹏邀请了几位 TuGraph 的老朋友一起聊聊图技术、图生态、图智能。包括中国开源先锋人物 、华为产业发展专家、Rust 技术专家马全一老师，北京大学前沿交叉学科研究院数据科学博士庞悦，蚂蚁集团知识图谱专家王少飞，以及TuGraph 开源负责人范志东。各位老师就为什么开源、图技术的未来与学术热点、图与AI等话题进行了精彩讨论。2024年，TuGraph 将努力更贴近客户，更拥抱开源，更关注生态。欢迎大家继续关注！

权威报告：蚂蚁集团TuGraph跻身中国图数据库市场“领导者”象限

蚂蚁技术AntTech TuGraph 2023年07月18日 09:25

7月15日，全球领先的IT市场研究和咨询公司IDC发布了最新的市场研究报告《IDC MarketScape：中国图数据库市场厂商评估，2023》。报告显示，蚂蚁集团自研的企业级图数据管理平台TuGraph跻身"领导者"象限。

（IDC发布的“中国图数据库市场，2023”象限）

IDC在报告中列举了蚂蚁集团TuGraph的五大优势。第一，TuGraph研发七年多来支持了蚂蚁集团300多种应用，在大量不同场景中长时间使用，产品成熟度、丰富度等方面具有优势。第二，具备业界鲜有的在线、流式、离线"三线一致"计算能力，覆盖了毫秒级延迟、分钟级和小时级等不同引擎，满足不同场景的性能需求。第三，支持每秒千万级查询的超高吞吐，毫秒级超低延迟。第四，生态工具“一站式图研发及可视化平台 TuGraph-Platform”降低了用户使用门槛。第五，在其他非金融场景，如社交推荐、数据血缘管理、故障归因分析等场景广泛使用，应用体系成熟。

本次报告IDC主要评估了中国市场上12家典型的企业级图数据库厂商，类型覆盖互联网厂商、云服务厂商、大数据厂商等。IDC针对入选图数据库厂商的产品能力（Capabilities）和技术战略（Strategies）两个维度，考察了目前的 产品技术能力、市场和生态以及未来发展战略等，评估了16个细分能力指标和4个细分战略指标，包含72个指标评估项，并配以不同权重。该报告为企业发展图计算和投资选型提供了有力支持。

图数据库，即以图（Graph）数据结构来进行存储和分析的数据库。与传统数据库相比，图数据库擅长关系分析，能更好地管理和组织数据，开发上层智能模型，同时也能实现海量数据的高并发、低延迟分析处理，提高数据变现的商业价值。当前图数据库应用主要集中在欺诈检测、人际关系分析和预测分析等领域。

IDC调研发现，数字经济、产业数字化转型进入深化发展阶段，企业对于业务逻辑开发和关系挖掘需求增强，图数据库市场受大型央国企数字化转型政策的推动明显。95%的企业认为图数据库是重要的数据管理工具，超过 65%的厂商认为在业务上图数据库优于其他选择。整体来看，图数据库的使用仍处于早期阶段，仍然缺乏统一标准范式，技术生态环境弱势，低操作门槛的图计算平台仍有较大缺口。

蚂蚁集团从2015年开始布局图技术，与清华大学合作研发了高性能图数据库TuGraph，扛住了蚂蚁万亿级业务的高性能要求。TuGraph在功能完整性、吞吐率、响应时间等技术指标上处于全球领先水平，两次打破国际图数据库基准性能测试世界纪录，成为LDBC-SNB世界纪录保持者。TuGraph在2021年帮助支付宝实现了资产损失率小于亿分之0.98的目标。

目前，蚂蚁集团已经开源了TuGraph系统中的单机版图数据TuGraph-DB和流式图计算引擎TuGraph-Analytics。TuGraph-DB提供了完备的图数据库基础功能和成熟的产品设计，具备完整的事务支持和丰富的系统特性，可在单机上部署，使用成本低，支持TB级别的数据规模。而TuGraph-Analytics是业界首个工业级流式图计算引擎，能够在超大规模图上进行流式复杂计算。

未来，蚂蚁将持续加强技术开放，为业界带来先进的图计算技术能力，与行业携手建设图计算技术生态。

蚂蚁图数据库再获LDBC权威测试世界第一

近日，国际权威图数据库测试机构国际关联数据基准委员会（LDBC）公布了行业通用的社交网络基准测试（LDBC SNB）最新结果。蚂蚁集团图数据库TuGraph打破官方审计测试纪录，再次拿到世界第一，这一纪录较LDBC早前公布的最高纪录吞吐量提升了52%，也超过了两年前由TuGraph保持的世界纪录1倍以上。

据LDBC官方发布的报告，在本次测试中，TuGraph在不同规模的数据集下均表现优异，在最大数据规模300G的数据集（8亿个结点，53亿条边）上，TuGraph的吞吐率较上一次官方纪录提升了52%，在系统事务性、可恢复性、正确性、稳定性等方面均达到官方标准，体现了TuGraph高并发低延迟的强大性能优势。
为了更加贴近真实场景使测试更加严谨，TuGraph还采用了Client/Server部署，将客户端和服务器分别部署在两台服务器上，在更严苛的条件下（固有网络延迟与网络波动）完成了本次测试。
蚂蚁集团也是LDBC最新的金融图数据测试基准Finbench的发起人和主要建设者。
关于LDBC和SNB测试
LDBC，即“关联数据基准测评委员会”（Linked Data Benchmark Council），是全球公认的图数据库领域基准指南制定者与测试机构，与TPC并称为国际数据库行业两大权威技术组织。
SNB，即社交网络基准测试 （Social Network Benchmark），是由LDBC开发的面向图数据库的基准测试（Benchmark）之一。SNB测试由于更贴近现实系统，同时包含了读写任务，简单和复杂查询，规定了系统的响应时间，更能体现系统的综合性能，是目前图数据行业最成熟和通用的性能测试。
LDBC SNB测试由指定的第三方机构进行，从数据导入到结果验证均由第三方在云平台上执行，最终结果由LDBC执行委员会进行审计并公布，最大限度的保证了结果的可信性。同时，SNB还公布了测试过程所用的程序和脚本，以及测试过程中产生的详细结果，进一步确保了测试的可复现性。
关于TuGraph
蚂蚁集团图数据库TuGraph是基于图模型的一站式数据存储和分析系统，擅长处理大规模关联数据的管理和分析，如社交关系、物流服务、设备管网、金融交易等场景，数千倍优化分析性能，天然具备数据可视化展示。
TuGraph拥有业界领先的集群规模和性能，是蚂蚁集团金融风控能力的重要基础设施，显著提升了欺诈洗钱等金融风险的实时识别能力和审理分析效率，提供了稳定的决策支持能力，其中，支撑支付宝的重要风险识别能力提升了近10倍，风险审理分析效率提升90%。
TuGraph 已被成熟应用于安全风控、信贷风控、知识图谱、数据血缘、资金分析、流量归因分析、会员关系等场景，并面向金融、工业、政务服务等行业客户。

TuGraph：从清华到蚂蚁

蚂蚁集团图计算TuGraph（原名GeaGraph），是蚂蚁集团与清华大学联合研发的大规模图计算系统，构建了一套包含图存储、图计算、图学习、图研发平台的完善的图技术体系，拥有业界领先规模的图集群，是图数据库基准性能测试LDBC-SNB世界纪录保持者。
TuGraph是蚂蚁集团金融风控能力的重要基础设施，显著提升了欺诈洗钱等金融风险的实时识别能力和审理分析效率，并面向金融、工业、政务服务等行业客户。
清华时期

早在2010年前后，清华大学计算机系高性能所就开始图计算相关技术及系统研究。于2016年成功研发的双子星图计算系统比业界常用的开源图计算引擎GraphX性能提高了约100倍，获得了业界的广泛关注。

费马时期

为了推动技术的广泛应用，2016年从事图计算研究的清华师生成立了费马科技有限公司。费马科技在推进双子星系统应用的同时，进一步开发出了具有国际领先性能的图数据库产品TuGraph，能够支持完整的图数据库事务，服务了搜狗搜索、京东金融、以及大型国有银行、国家级能源企业在内的不同行业用户，并在2020年通过了国际图数据库标准组织LDBC的认证测试，认证成绩是第二名的7.6倍。

蚂蚁时期

蚂蚁集团在很多领域具有科技领先能力，国内乃至全球最大的用户量和峰值交易量使得蚂蚁集团对图计算有着丰富的应用需求，利用图计算技术处理支付宝的反欺诈、反套现等难题，可以比传统技术更加适用。从2015年起，蚂蚁集团开始自主研发分布式图数据库、流式图计算等图计算技术系统，并在内部应用中获得了良好的效果。

2020年，蚂蚁集团进一步整合了自有的图计算技术系统GeaBase，以及清华大学和费马科技的产品和技术，升级形成了一套完整的图计算系统GeaGraph（后统一采用TuGraph命名）。这套系统集成了各方原有优势，经过多年技术积累和大规模实战打磨，无论从功能的完整性，还是吞吐率、响应时间等性能指标，都达到了世界领先水平。

2023开放原子全球开源峰会，蚂蚁图计算平台开源业内首个工业级流图计算引擎

6月11日，2023开放原子全球开源峰会在北京开幕。本次峰会以“开源赋能，普惠未来”为主题。在高峰论坛上，蚂蚁技术研究院院长、图计算负责人陈文光宣布开源TuGraph 图计算平台核心成员——工业级流式图计算引擎 TuGraph Analytics。

去年9月，蚂蚁集团开源了 TuGraph 图计算平台中的图数据库 TuGraph DB。这次开源是 TuGraph 图计算平台的又一次开源升级，进一步加大了蚂蚁在图计算基础软件领域的开放力度，也是通过开放协同促进科技创新的实际行动。

图（Graph）是一种抽象的数据结构，由顶点和边构成。图计算是一种以图结构建模的算法模型，可对大规模数据进行关系挖掘和复杂计算，实现知识推理和事件溯源。图计算目前已广泛应用在金融、政务、医疗等领域，备受全球研发机构和顶尖科技公司关注。流式图计算是一种将流式计算和图计算结合的交叉创新，融合了流式计算的高度实效性和图计算的灵活性，攻坚难度极高。据了解，蚂蚁从2015年开始探索图计算，布局了图数据库、流式图计算引擎、图学习等相关技术，打造了世界规模领先的图计算集群，于业界首创了工业级流式图计算引擎，多次问鼎图数据库行业权威测试 LDBC 世界冠军并保持世界纪录。此次开源的工业级流式图计算引擎是蚂蚁从 2017 年开始布局打造，经过五年多工业级应用大考，流式图计算做到了在千亿数据规模的“图”上秒级延迟计算，是蚂蚁风控的核心基础技术，成功解决了金融场景风险分析难、识别率低、时效性差等业界难题。

图计算是下一代人工智能关键核心技术。中国工程院院士郑纬民曾指出，“高性能图计算是当前全球人工智能竞争的战略性制高点，我们要加快攻克技术、突破产业瓶颈，防止在高性能图计算这一关键技术领域再被卡脖子”。而开源是共享科技成果，加速先进技术落地的最快路径。陈文光强调，开源是蚂蚁的核心技术战略，也是面向数字化未来可持续创新的动力。此次流式图计算引擎开源，是延续蚂蚁开源核心

基础技术的实际动作，希望通过开放成熟的图计算技术，服务更广阔的数字化产业，向世界输出中国科技公司的前沿技术影响力。未来，蚂蚁集团也愿意携手行业伙伴共同突破技术创新，繁荣开源生态。据了解，蚂蚁开源聚焦于数据库、云原生、中间件等基础软件领域，积累了近100个社区头部开源项目、近1600个开源仓库、9大核心开源项目，如“2022世界人工智能大会镇馆之宝”隐语隐私计算技术栈、分布式数据库 OceanBase、行业首个通过商用密码产品认证的密码学技术“铜锁”等自研核心技术。

TuGraph Analytics (别名：GeaFlow) 是蚂蚁集团开源的流图计算引擎，支持万亿级图存储、图表混合处理、实时图计算、交互式图分析等核心能力，目前广泛应用于数仓加速、金融风控、知识图谱以及社交网络等场景。

特性

- 分布式实时图计算
- 图表混合处理（SQL+GQL语言）
- 统一流批图计算
- 万亿级图原生存储
- 交互式图分析
- 高可用和Exactly Once语义
- 高阶API算子开发
- UDF/图算法/Connector插件支持
- 一站式图研发平台
- 云原生部署

实时能力

相比传统的流式计算引擎比如Flink、Storm这些以表为模型的实时处理系统而言，GeaFlow以图为数据模型，在处理Join关系运算，尤其是复杂多跳的关系运算如3跳以上的Join、复杂环路查找上具备极大的性能优势。
GeaFlow简介

GeaFlow起源

早期的大数据分析主要以离线处理为主，以Hadoop为代表的技术栈很好的解决了大规模数据的分析问题。然而数据处理的时效性不足， 很难满足高实时需求的场景。以Storm为代表的流式计算引擎的出现则很好的解决了数据实时处理的问题，提高了数据处理的时效性。 然而，Storm本身不提供状态管理的能力， 对于聚合等有状态的计算显得无能为力。Flink 的出现很好的弥补了这一短板，通过引入状态管理以及Checkpoint机制，实现了高效的有状态流计算能力。
随着数据实时处理场景的丰富，尤其是在实时数仓场景下，实时关系运算(即Stream Join) 越来越多的成为数据实时化的难点。Flink虽然具备优秀的状态管理能和出色的性能，然而在处理Join运算，尤其是3度以上Join时， 性能瓶颈越来越明显。由于需要在Join两端存放各个输入的数据状态，当Join变多时，状态的数据量急剧扩大，性能也变的难以接受。 产生这个问题的本质原因是Flink等流计算系统以表作为数据

模型，而表模型本身是一个二维结构，不包含关系的定义和关系的存储， 在处理关系运算时只能通过Join运算方式实现，成本很高。
在蚂蚁的大数据应用场景中，尤其是金融风控、实时数仓等场景下，存在大量Join运算，如何提高Join 的时效性和性能成为我们面临的重要挑战，为此我们引入了图模型。图模型是一种以点边结构描述实体关系的数据模型，在图模型里面，点代表实体， 边代表关系，数据存储层面点边存放在一起。因此，图模型天然定义了数据的关系同时存储层面物化了点边关系。基于图模型，我们实现了新一代实时计算 引擎GeaFlow，很好的解决了复杂关系运算实时化的问题。目前GeaFlow已广泛应用于数仓加速、金融风控、知识图谱以及社交网络等场景。
技术架构

GeaFlow整体架构如下所示：

- DSL层：即语言层。GeaFlow设计了SQL+GQL的融合分析语言，支持对表模型和图模型统一处理。
- Framework层：即框架层。GeaFlow设计了面向Graph和Stream的两套API支持流、批、图融合计算，并实现了基于Cycle的统一分布式调度模型。
- State层：即存储层。GeaFlow设计了面向Graph和KV的两套API支持表数据和图数据的混合存储，整体采用了Sharing Nothing的设计，并支持将数据持久化到远程存储。
- Console平台：GeaFlow提供了一站式图研发平台，实现了图数据的建模、加工、分析能力，并提供了图作业的运维管控支持。
- 执行环境：GeaFlow可以运行在多种异构执行环境，如K8S、Ray以及本地模式。

应用场景

实时数仓加速

数仓场景存在大量Join运算，在DWD层往往需要将多张表展开成一张大宽表，以加速后续查询。当Join的表数量变多时，传统的实时计算引擎很难 保证Join的时效性和性能，这也成为目前实时数仓领域一个棘手的问题。基于GeaFlow的实时图计算引擎，可以很好的解决这方面的问题。 GeaFlow以图作为数据模型，替代DWD层的宽表，可以实现数据实时构图，同时在查询阶段利用图的点边物化特性，可以极大加速关系运算的查询。
实时归因分析

在信息化的大背景下，对用户行为进行渠道归因和路径分析是流量分析领域中的核心所在。通过实时计算用户的有效行为路径，构建出完整的转化路径，能够快速帮助业务看清楚产品的价值，帮助运营及时调整运营思路。实时归因分析的核心要点是准确性和实效性。准确性要求在成本可控下保证用户行为路径分析的准确性;实效性则要求计算的实时性足够高，才能快速帮助业务决策。 基于GeaFlow流图计算引擎的能力可以很好的满足归因分析的准确性和时效性要求。如下图所示：
GeaFlow首先通过实时构图将用户行为日志转换成用户行为拓扑图，以用户作为图中的点，与其相关的每个行为构建成从该用户指向埋点页面的一条边.然后利用流图计算能力分析提前用户行为子图，在子图上基于归因路径匹配的规则进行匹配计算得出该成交行为相应用户的归因路径，并输出到下游系统。
实时反套现

在信贷风控的场景下，如何进行信用卡反套现是一个典型的风控诉求。基于现有的套现模式分析，可以看到套现是一个环路子图，如何快速，高效在大图中快速判定套现，将极大的增加风险的识别效率。以下图为例，通过将实时交易流、转账流等输入数据源转换成实时交易图，然后根据风控策略对用户交易行为做图特征分析，比如环路检查等特征计算，实时提供给决策和监控平台进行反套现行为判定。通过GeaFlow实时构图和实时图计算能力，可以快速发现套现等异常交易行为，极大降低平台风险。

深入解读TuGraph计算引擎模型推理系统

TuGraph计算引擎模型推理系统将基于迭代计算的图计算框架与模型推理系统相结合，推理系统可自定义推理依赖环境，图迭代计算与推理链路实现隔离。基于共享内存的跨进程通信方式，提高了推理数据交换效率，满足流图近线推理的时效性。在蚂蚁集团内部的实际应用场景中，大幅缩短了模型推理上线的链路与开发时间，用户迭代模型版本更方便快捷。

1. 图算法概述

在计算机科学中，图是一种表示实体（节点或顶点）以及实体之间关系（边）的数据结构。图模型可以天然地描述网络结构，能更清晰地表达复杂的数据关系和依赖，简化关联数据的理解和分析。在不同的场景下，图中点边具备不同的语义信息。比如在资金交易场景下，每个人可以抽象成一个点表示，人与人之间的转账关系可以抽象成一条边表示。如下图，通过图数据模型反映出各个实体之间的资金往来关系，让数据的关联分析更加直观和高效。

资金交易图谱示例

在图数据模型上可以执行多种图算法，如社区检测，最短路径匹配，环路检测算法等。通过点边上的迭代计算，探索图模型中各个实体之间的关系。探索过程不依赖于数据的线性结构，从而便于识别隐藏的模式和关联关系。在主流迭代图算法中，节点通过消息传递的方式进行通信。每次迭代，节点可以接收来自它们邻居的消息，处理这些消息，然后决定是否发送新的消息给其他节点。迭代算法中，每个节点有一个状态，每次迭代它们都有可能更新这个状态直至收敛。例如，在PageRank算法中，每个节点的状态是其PageRank值，这个值在迭代过程中会随着邻居的值的更新而更新。

图迭代算法解决了经典的图计算问题，但随着业务需求的复杂度提升，基于迭代的图算法存在着表达能力不足、自适应性能力差、异质图处理难度大等缺点。近年来随着深度学习的研究和应用的发展，以图神经网络（Graph Neural Networks，GNNs）为代表的一类神经网络算法，被设计用来捕获图中实体（节点）和关系（边）间的复杂模式。图神经网络能够结合节点特征和图的结构来学习节点和边的表示，相比之下，传统的迭代图算法通常不会直接从原始特征中学习，而更多地专注于结构特征。依赖于深度学习的天然优势，GNNs具有更强的表示学习能力，可以自动从数据中学习复杂的模式，这使得 GNNs 能够更好地处理多任务学习和迁移学习等问题。在社交网络分析、知识图谱、生物分子网络、推荐系统以及交通网络等领域，得到广泛应用。

2. 流图推理简介

TuGraph计算引擎（TuGraph Analytics[1]）是蚂蚁集团开源的大规模分布式实时图计算引擎（流图引擎），实现了流批一体的图计算模型，支持了丰富的图计算算法。TuGraph Analytics的流图计算能力，能处理连续输入的数据流，并支持增量的计算模式，极大得提高了数据的计算效率和实时性。TuGraph Analytics解决了业界大规模数据关联分析的实时计算问题，已广泛应用于数仓加速、金融风控、知识图

谱以及社交推荐等场景。
随着业务场景中问题复杂度的提升，基于传统的迭代图算法已无法满足业务的实际需求。例如在反洗钱场景中，利用图神经网络算法处理复杂的交易关系，能够捕获到节点的局部图结构信息。通过聚合邻接节点的特征信息，每个交易节点都可以感知到周边图网络结构的信息。类似的图神经网络等AI模型的推理逻辑，是无法基于传统的图迭代计算模式直接高效地表达的。
受上述问题启发，我们思考是否可以将TuGraph Analytics的流图计算能力与图神经网络等深度学习模型相结合，开发一套基于流图计算的模型推理系统。最终期望的推理系统具备如下能力：

- 对于图算法工程师，在图迭代计算过程中，能够方便地使用机器学习模型的推理能力。
- 对于AI算法工程师，可以通过TuGraph Analytics分布式流式计算的能力实现实时的模型推理。

众所周知，在深度学习为代表的数据科学领域，Python已经成为数据分析、模型训练和推理框架的主流开发语言，并提供了丰富的开发库和框架生态。而以Hadoop全家桶为代表的大数据计算引擎领域，基于Java语言开发的系统仍占据一席之地，当然TuGraph Analytics也在其中。这种语言差异带来的“互操作性”成本，使得相当一部分大数据和AI生态组件无法轻松地融合，这也是TuGraph Analytics支持图推理需要亟待解决的问题。
3. 系统设计
我们对业内的跨Python & Java语言的方案进行了充分的调研，通过深入对比现有的跨语言交互方案的性能与效率，最终决定将模型推理任务运行于Python原生环境中以发挥出最佳的性能。
1. OONX
OONX是一个开发的生态系统，为不同的机器学习框架之间提供一个标准的模型表示格式。它使得开发人员能够在不同的框架、工具、运行时环境之间以一种标准方式交换模型，从而简化了模型的迁移和部署。
优点
缺点

框架互操作性	转换成本高
生态系统支持	更新滞后
优化推理	版本兼容性
规范化模型格式	性能不一致

2. Jython
以Jython为代表的方式，主要思想是在运行的宿主虚拟机上，使用宿主语言重新编写实现。
优点
缺点

Java集成	版本管理复杂
跨平台	支持库有限
线程	更新滞后
-	支持Python3有限

3. Py4j
Py4j桥接库为代表的方式，以Socket通信模型为基础，实现Python和Java互相访问对象，方法，提供两个程序相互通信的能力。
优点
缺点

跨语言交互	网络传输
动态代理	部署分发难度大
支持复杂类型	版本难兼容
API使用简易	运行时环境依赖复杂

4. Web服务化
Web服务化是一种将机器学习模型部署成网络服务，调用者通过相应的api获取模型推理结果。
优点
缺点

扩展性好	性能差
简易且轻量	不适合计算密集型场景
社区支持	无状态管理
机器学习类库易集成	并发连接有限

在TuGraph Analytics模型推理系统的架构设计中，核心部分是通过C++原生语言建立起来的一座桥梁，实现Python环境和Java虚拟机之间高效的数据交互和操作指令的传递。通过使用C++作为媒介语言，我们不仅能够利用其接近硬件的执行效率，确保数据交互的性能，还能够保证在两个虚拟环境之间数据交换的计算精度和稳定性。基于共享内存的设计允许Python和JVM进程各自独立运行，既保证了运行环境的安全隔离，又能实现数据的高效共享。
4. 技术原理
TuGraph Analytics模型推理系统工作流中，Driver端（即控制节点）发挥着至关重要的角色。该节点运行在Java虚拟机进程，是整个推理流程的控制中心。Driver端初始化了一个非常关键的组件——InferenceContext对象，InferenceContext对象被设计为模型推理流程的核心，在JVM环境中创建并负责加载和预处理用户提供的模型文件和环境依赖信息。在模型推理任务之前，InferenceContext会详细检查并准备好模型文件，确保能够正确加载到预期的执行环境中。InferenceContext也负责初始化和配置与模型推理相关的虚拟环境，确保正确的Python环境或其他必要的运行时库得以安装和配置。
如图所示，由流式数据源源不断的触发图迭代计算与模型推理工作。TuGraph计算引擎提供了DeltaGraphCompute计算接口，用户可自主定义增量图数据的处理逻辑，并更新历史的图存储(Graph Store)。通过TuGraph计算引擎模型推理系统，增量图迭代的中间计算结果，经过推理前置数据处理接口，并基于共享内存的跨进程通信方式，将处理后的数据流输入到推理进程，完成推理工作后的结果参与后续图迭代计算逻辑。下文将详细介绍各个数据接口的使用。
4.1 计算推理隔离
在TuGraph Analytics模型推理系统的架构中，集群的工作负载分配给多个worker节点。每个worker节点上运行着两个关键进程：负责图数据迭代计算的Java进程，以

及执行模型推理的Python进程。为了充分利用计算系统的资源，推理进程在没有接收到推理请求时，会进入睡眠状态。这样的设计不仅减少了系统资源的占用，而且降低了系统的整体能耗。当推理请求到达时，推理进程会被立即唤醒，接收和执行新的推理任务。借助睡眠与唤醒机制以及智能的任务调度策略，可以保证系统能够以高效、稳定、节能的方式运行，同时满足了大规模图数据处理和实时推理的需求。

在每个worker工作节点下，按照不同的推理作业级别划分基础的虚拟环境，从而保证一个wroker节点也可以支持不同推理任务，支持标准的requirements.txt管理推理依赖库。

在图迭代计算进程和推理进程之间通过数据队列实现双边数据的交互，通过在数据包的头文件中插入参数个数，长度等信息，推理进程在连续若干次收到空消息包后，将自动进入睡眠状态，释放cpu等资源。图迭代计算进程调用推理接口时，推理进程将快速退出睡眠状态，接收输入数据并完成推理流程。

4.2 跨进程数据交换

对于推理数据的交换部分，底层通过C++开发共享内存管理模块，实现两个进程之间的数据交互。在推理初始化阶段，由InferenceContext对象开辟进程共享内存，Java进程负责创建并初始化推理（Python）进程，通知推理进程共享内存的地址信息，并映射到相应的进程。如图，Java进程和推理进程均采用C++作为桥梁语言，实现共享内存中数据的流动操作。

在推理系统的性能测试阶段，我们发现推理进程读写进程时，接口的调用开销不容忽视。常规的理解认为C++能够优化Python的执行效率，但前提是Python的执行内存足够复杂，优化执行内容的收益远大于接口的调用开销。然而，在我们系统设计中，共享内存的读写接口只是操作了内存地址，实现读写指针的移动。因此，接口的调用开销也是影响推理性能的关键因素，为此，我们充分调研了业界主流的方案。

解决方案

描述

CPython	C语言实现Python及其解释器（JIT编译器）
ctypes	Python标准库
SWIG	开发工具，封装native程序接口
Pybind11/Boost.python/ Nanobind	轻量级头文件库
Cython	Python的超集，使用C语言特性，静态编译

如图所示，我们对比了多种Python调用C链接库的方案，性能是第一要素，因此选择Cython作为推理进程和底层内存交互的工具。Cython是一个编程语言，是Python语言的一个超集，它将/C++的静态类型系统融合在了Python中，允许开发者可以在Python代码中直接使用C语言的特性，从而提高程序的执行效率。Cython将Python源代码翻译为C或C++代码，然后将其编译为二进制代码，能够显著提高数值计算和循环场景的代码执行性能。

4.3 推理接口设计

上文介绍了采用Cython作为推理进程内存管理的链接工具，如下为TuGraph Analytics模型推理系统的内存管理接口设计，提供了初始化，读和写三个接口。

1. 初始化接口：负责共享内存地址的映射和读指针的初始化。

2. 读接口：数据bytes的长度作为输入参数，直接在内存端上移动相应长度返回数据

段，并移动到读指针。
3. 写接口：将bytes和bytes长度写入到共享内存，并移动至写指针。
```
@cython.final
cdef class InferIpc:

    cdef MmapIPC * ipc_bridge;
    cdef uint8_t* read_ptr;

    def __cinit__(self, r, w):
        self.ipc_bridge = new MmapIPC(r, w)
        self.read_ptr = self.ipc_bridge.getReadBufferPtr()

    cpdef inline bytes readBytes(self, bytesSize):
        if bytesSize == 0:
            return b""
        cdef int readSize
        cdef int len_ = bytesSize
        with nogil:
            readSize = self.ipc_bridge.readBytes(len_)
        if readSize == 0:
            return b""
        cdef unsigned char * binary_data = self.read_ptr
        return binary_data[:len_]

    cpdef inline bool writeBytes(self, bytesBuf, length):
        cdef bool writeFlag
        cdef int len_ = length
        cdef char* buf_ = bytesBuf
        with nogil:
            writeFlag = self.ipc_bridge.writeBytes(buf_, len_)
        return writeFlag

    def __dealloc__(self):
        del self.ipc_bridge
```
如下为用户实现推理的Java接口，同其它图迭代计算接口一样，需要推理的时候直接调用该接口，将图迭代的中间结果inputs发送到推理进程并返回模型结果。
```
public interface GraphInferContext<OUT> extends Closeable {
    OUT infer(Object... inputs);
}
```
5. 最佳实践
我们以PageRank任务结合群组打分模型推理流程为例，演示具体的操作流程。
5.1 数据处理
定义推理数据前后置处理逻辑如下：
```
import abc
import json
import sys
import os
```

```
import torch

class MyInference(TransFormFunction):
    def __init__(self):
        super().__init__(2)

    def transform_pre(self, *args):
        return args[0], args[1]

    def transform_post(self, *args):
        return args[0]
```

5.2 图迭代推理

定义图迭代计算结合推理逻辑如下：

```
    public static class PRVertexCentricComputeFunction implements
        IncVertexCentricComputeFunction<Integer, Integer, Integer, Integer> {

        private IncGraphComputeContext<Integer, Integer, Integer, Integer>
graphContext;
        private IncGraphInferContext<String> inferContext;

        @Override
        public void init(IncGraphComputeContext<Integer, Integer, Integer,
Integer> graphContext) {
            this.graphContext = graphContext;
            this.inferContext = (IncGraphInferContext<String>) graphContext;
        }

        @Override
        public void evolve(Integer vertexId,
                           TemporaryGraph<Integer, Integer, Integer> temporaryGraph) {
            long lastVersionId = 0L;
            IVertex<Integer, Integer> vertex = temporaryGraph.getVertex();
            HistoricalGraph<Integer, Integer, Integer> historicalGraph =
graphContext
                .getHistoricalGraph();
            if (vertex == null) {
                vertex = historicalGraph.getSnapShot(lastVersionId).vertex().get();
            }

            if (vertex != null) {
                List<IEdge<Integer, Integer>> newEs = temporaryGraph.getEdges();
                List<IEdge<Integer, Integer>> oldEs =
historicalGraph.getSnapShot(lastVersionId)
                    .edges().getOutEdges();
                if (newEs != null) {
                    for (IEdge<Integer, Integer> edge : newEs) {
                        graphContext.sendMessage(edge.getTargetId(), vertexId);
```

```
            }
        }
        if (oldEs != null) {
            for (IEdge<Integer, Integer> edge : oldEs) {
                graphContext.sendMessage(edge.getTargetId(), vertexId);
            }
        }
    }

}

@Override
public void compute(Integer vertexId, Iterator<Integer> messageIterator) {
    int max = 0;
    while (messageIterator.hasNext()) {
        int value = messageIterator.next();
        max = max > value ? max : value;
    }
    IVertex<Integer, Integer> vertex =
graphContext.getTemporaryGraph().getVertex();
    IVertex<Integer, Integer> historyVertex =
graphContext.getHistoricalGraph().getSnapShot(0).vertex().get();
    if (vertex != null && max < vertex.getValue()) {
        max = vertex.getValue();
    }
    if (historyVertex != null && max < historyVertex.getValue()) {
        max = historyVertex.getValue();
    }
    graphContext.getTemporaryGraph().updateVertexValue(max);
}

@Override
public void finish(Integer vertexId, MutableGraph<Integer, Integer,
Integer> mutableGraph) {
    IVertex<Integer, Integer> vertex =
graphContext.getTemporaryGraph().getVertex();
    List<IEdge<Integer, Integer>> edges =
graphContext.getTemporaryGraph().getEdges();
    if (vertex != null) {
        mutableGraph.addVertex(0, vertex);
        graphContext.collect(vertex);
    } else {
        LOGGER.info("not found vertex {} in temporaryGraph ", vertexId);
    }
    if (edges != null) {
        edges.stream().forEach(edge -> {
            mutableGraph.addEdge(0, edge);
```

```
            });
        }
        List<String> inferInput = new ArrayList<>();
        inferInput.add(String.valueOf(vertexId));
        inferInput.add("param2");
        String infer = this.inferContext.infer(inferInput.toArray(new Object[]
{0}));
      }
   }
```

5.3 创建作业

在Console作业管理平台创建一个HLA任务，上传图迭代计算jar包，模型文件和依赖管理文件。

5.4 配置参数

配置相关参数，启动运行作业即可。

```
"geaflow.infer.env.enable":"true",
// 初始化虚拟环境等待时间
"geaflow.infer.env.init.timeout.sec":120,
// 是否接收日志
"geaflow.infer.env.suppress.log.enable":"true"
```

6. 总结

通过将AI模型推理引入TuGraph Analytics流图计算系统，让我们能够对图数据进行深度地分析和预测。利用最新的机器学习和深度学习技术，TuGraph Analytics图计算引擎不仅可以对图数据进行分类和回归分析，还可以预测未来趋势，从而在多个维度上提供决策支持。

希望通过以上的介绍，可以让大家对TuGraph Analytics模型推理系统有个比较清晰的了解，非常欢迎大家加入我们社区（https://github.com/TuGraph-family/tugraph-analytics），一起构建图数据上的智能化分析能力！

Q: TuGraph 的边是否支持索引?

A: TuGraph 在引擎层支持边索引，可通过存储过程使用。Cypher的边索引功能正在开发支持中

Q: TuGraph 单机的QPS是多少?

A: 不同数据规模，不同查询操作的QPS差异较大，比如LDBC SNB典型图操作超过1.2万，参考测试结果：https://www.tugraph.org/blog?id=0

Q: 可视化文件 build 后如何更新到 tugraph 服务?

A: 可视化文件打包后，需要进行以下操作进行替换。

- 登录 tugraph 服务所在的服务或 docker 容器内。
- 通过 lgraph_server --help 查看服务启动的配置文件所在目录。通常情况：`/usr/local/etc/lgraph.json`
- 查看 `/usr/local/etc/lgraph.json`文件中 web 的配置目录。通常情况：`/usr/local/share/lgraph/resource`
- 将可视化打包后生成的文件夹中的内容全部替换到配置目录下 `/usr/local/share/lgraph/resource`
- 重新启动 tugraph 服务

Q：如何通过 npm run dev，连接已有的 tugraph 服务?
A：启动之前，需要修改文件`.env.development`中的'VUE_APP_REQUESTURL'的配置项。然后在通过`npm run dev`进行启动。
示例：

```
NODE_ENV = development VUE_APP_TITLE = TuGraph(dev)
VUE_APP_REQUESTURL = http://localhost:7070/
```

Q：client 目前有哪些编程语言，是否支持 node js?
A：目前主要支持的编程语言有 c++,python,java；目前不支持 node js。使用 node 作为主要开发语言的用户，可以使用 tugraph 提供的 restful api 来调用。建议使用 Cypher 来封装调用接口。后续版本 restful api 将不再进行更新维护，只会保留登录、登出、刷新 token、cypher 调用这几个常见的 api。

Q：python client 是否支持 pip install？client 在哪里进行引用?
A：目前 python client 不支持 pip 进行安装。client 在目录https://github.com/TuGraph-db/tugraph-db/tree/master/src/client。

Q: TuGraph 可以对接那些常用数据库?
A: TuGraph通过DataX可以实现大部分主流数据库的导入导出，支持的数据库包括 MySQL、Oracle、Hive 等。具体参考https://github.com/TuGraph-db/DataX

Q：如何加载存储过程或算法包?
A：加载方式有两种：

- 第一种：通过可视化页面的插件模块，通过交互操作完成加载。
- 第二种：通过 cypher 语句实现存储过程的加载。

```
CALL
db.plugin.loadPlugin(plugin_type::STRING,plugin_name::STRING
,plugin_content::STRING,code_type::STRING,plugin_description
::STRING,read_only::BOOLEAN) :: (::VOID)
```

Q：如何调用或执行存储过程?
A：可以使用 cypher 进行存储过程的执行或调用。

```
CALL
db.plugin.callPlugin(plugin_type::STRING,plugin_name::STRING
,param::STRING,timeout::DOUBLE,in_process::BOOLEAN) ::
(success::BOOLEAN,result::STRING)
```

Q:开源内置的算法包在哪里?
A：代码地址https://github.com/TuGraph-db/tugraph-db/tree/master/plugins

Q：如何使用 docker 镜像安装?
A：

- 确认本地是否有 docker 环境，可使用`docker -v`进行验证。如果没有请安

装 docker，安装方式见 docker 官网文档https://docs.docker.com/install/ 。

- 下载 docker 镜像，下载方式可使用`docker pull tugraph/tugraph-runtime-centos7`，也可以在官网下载页面进行下载https://www.tugraph.org/download[注：下载的文件是*.tar.gz 的压缩包，不用解压]。
- 如果使用 docker pull 下载的镜像则不用导入镜像。如果使用官网下载的压缩包，则要使用`docker load -i ./tugraph_x.y.z.tar`[注：x.y.z 是版本号的代替符，具体数值根据自己下载的版本进行改写]
- 启动 docker 容器`docker run -d -p 7070:7070 -p 9090:9090 --name tugraph_demo tugraph/tugraph-runtime-centos7 lgraph_server`[注：具体的镜像名称 tugraph/tugraph-runtime-centos7 要以本地实际镜像名称为准，可用过 docker images 命令查看]

Q：rpm 包和 deb 包安装后，启动 lgraph_server 服务。提示缺少'liblgraph.so'报错?
A：此问题主要是环境变量导致，需要配置环境量。
示例：

```
export LD_LIBRARY_PATH=/usr/local/lib64
```

Q：是否支持不定长边的条件查询?
示例：

```
 MATCH p=(v)-[e:acted_in|:rate*1..3]-(v2) WHERE id(v) IN [3937] AND e.stars = 3 RETURN p LIMIT 100
```

A：目前还不支持不定长边的过滤查询。目前的代替方案只能是分开写。上面的示例，就需要从 1 跳到 3 跳都写一遍。

Q：如何查询最短路径，shortestPath 函数如何使用?
A：使用示例如下（示例图谱：MovieDemo）

```
 MATCH (n1 {name:'Corin Redgrave'}),(n2 {name:'Liam Neeson'})
     CALL algo.allShortestPaths(n1,n2) YIELD nodeIds,relationshipIds,cost
         RETURN nodeIds,relationshipIds,cost
```

详尽使用方案请参考官网文档https://www.tugraph.org/doc?version=V3.3.0&id=10000000000658658。

Q：查询语句 Where 后使用 and 进行拼接查询速度较慢，语句应如何优化改进?
示例：

```
 MATCH (n1),(n2) CALL algo.allShortestPaths(n1,n2)
     YIELD nodeIds,relationshipIds,cost
```

```
WHERE id(n1) IN [0] AND id(n2) IN [3938]
    RETURN nodeIds,relationshipIds,cost
```

A：目前 cypher 查询引擎正在优化中。现阶段语句改写可以通过 with 向下传递进行优化。
示例：

```
 MATCH (n1) where id(n1) in [0] with n1
 MATCH (n2) where id(n2) in [3938] with n1, n2
     CALL algo.allShortestPaths(n1,n2) YIELD
nodeIds,relationshipIds,cost
         RETURN nodeIds,relationshipIds,cost
```

Q：如何查询任意跳的边?
A：使用*..
示例：

```
 MATCH p=(a)-[*..]-(b) WHERE id(a) IN [3] AND id(b) IN [19]
RETURN p
```

Q：报错"User has reached the maximum number of tokens"后，怎么做?
A：这表明当前账号Token数量已达上限10000个。解决方法如下，任选其一：

1. 登出不使用的Token。
2. 重新启动TuGraph服务，会清空所有Token。
3. Token有效期默认为24小时，24小时后会自动失效并删除。