**中期报告**

演讲人：毛骏源

What

**项目概述**

基于2022年项目x-Wowkiddy和x-TOBEDONE进行各方面优化，主要包括在一定程度上解决数据一致性问题，尝试选用更好的数据库改进框架，以期获得更好的性能和功能，实现高效的图文件系统，并改进网页端可视化，实现三维可视化以更好地体现文件的相关性。

**图数据库**

图数据库（graph database，GDB）是一种使用图结构进行语义查询的数据库。

使用图数据库进行数据之间关系的表示非常直观，对于存储高度相连的数据非常有效。

在文件系统中使用图数据库进行存储（即图文件系统），可以为普通的文件系统赋予图结构具有的增加文件间关联，并查找内容相关文件的特点。

分布式图数据库在图数据库的基础上增加了分布式部署的功能。分布式图数据库兼顾了图文件系统方便用户快速搜索、模糊搜索、查找相关文件的特点，以及分布式文件系统的海量文件存储、云存储的特点。

**打标技术**

在图文件系统中，标签若以人工标注，则对于大量文件数据，将是非常繁重的工作量，于是通常采用深度学习进行标签的提取。具体来说，打标就是从不同格式，如文字、图片、视频等文件中提取出其内容关键词，深度学习一般是基于多分类来实现这一过程的。幸而现在有成熟的深度学习模型可供标签的提取，我们只需要布置一个打标服务器即可完成上传文件的打标任务。

**可视化**

文件系统的可视化主要基于文件系统结构。例如我们日常使用的文件系统结构多基于B+树，则文件之间关系的展现形式也就基于树形结构，即我们常见的文件夹-文件结构。

演讲人：毛骏源

对于图数据库而言，由于其逻辑结构为图结构，则自然的使用图的展现形式。例如图数据库中的节点表示文件，边表示文件之间的关联性

**如ppt图片所示**

**DisGraphFS**

DisGraphFS是一个分布式图文件系统，他将图结构的思想应用到分布式文件系统上面，使之兼具图文件系统方便用户快速搜索，模糊搜索，查找相关文件的特点，以及分布式文件系统的海量文件的存储、云存储的特点。这个项目的计算集群使用Ray,Ray是一个高性能分布式计算框架，存储使用的是JuiceFS，这是一个共享文件系统

大体框架如图所示(ppt图片)

包括中央索引服务器，存储集群，网页端，计算集群，客户端五个模块

**TOBEDONE**

该项目是对DisGraFS改进，搭建了完全意义上的存储集群，简化用户的操作。采用了dontpanic中的纠删码技术，以提高存储系统的可靠性。

搭建了分布式监控，通过监控保证服务正确稳定地运行。分布式监控是部署在分布式系统内的监控组件，可以监视和显示集群中各节点的状态信息，它有运行在各个节点的进程，可以采集不同节点之间的通信消息，采集各个节点的资源利用率，将数据汇总到一个数据库，进行分析处理后以直观的图形化界面进行呈现。

相较于前面的DisGraphFS，TOBEDONE的系统架构共有六个模块

* 索引服务器：与存储集群交互，获取存储节点的状态信息
* 图数据库：维护存储文件的图结构
* 分布式存储集群：文件切片后的存储位置
* 分布式计算集群：由 Ray 支持的打标计算集群
* 分布式监控：监控各模块状态、资源占用情况，可远程唤醒与关闭存储节点
* 网页端：呈现给用户的界面，用户可进行文件或文件夹操作，查看图结构

大体框架见ppt图片（翻页）

这张ppt左边是抽象后的结构

具体来说

* \* 图数据库仍然采用neo4j，在DisGraFS上做了改进。
* \* 分布式计算集群完成三个动作：接收文件、打标、发送打标信息。用户上传的文件，也会传一份给计算集群，计算集群检测到有新文件传来，就会调用打标操作，打标就是计算集 群的“计算”内容，多台电脑共同承担这些计算任务，这些任务的分发和负载均衡是依靠 Ray 来实现。
* 增加监控模块
* \* 网页端 增加了文件删除、重命名和共享功能，并优化了图结构界面
* \* 索引服务器：改为获取storage的状态信息，通过与storage通讯实现

Ppt的右边是ray模块和neo4j模块的细节

总结：DisGraphFS 相比，减少了客户端，DisGraphFS的客户端，就是要用户把存储的文件夹安装并挂载到本地，而TOBEDONE省去了客户端。另外增加监控模块。

但是仍然存在问题，主要在于数据一致性问题以及可视化、

(ppt翻页)

**数据一致性**

**数据一致性定义** 在分布式环境中,数据一致性是指数据在多个副本之间是否能够保持一致的特性。

数据一致性强包括强一致性和最终一致性

强一致性：数据的实际状态所体现的一致性；在任何情况下数据只有两种状态，所有副本一致或者不一致。在某些条件下不一致的状态是暂时的，还会转换到一致的状态

最终一致性：存储系统保证在没有新的更新的条件下，最终所有的访问都是最后更新的值。不保证在任意时刻任意节点上的同一份数据都是相同的，但是随着时间的迁移，不同节点上的同一份数据总是在向趋同的方向变化。简单说，就是在一段时间后，节点间的数据会最终达到一致状态。

Why

**数据一致性问题**

TOBEDONE等图文件系统存在的问题：未做到数据的一致性。

存在数据一致性问题

分布式系统的数据一致性问题主要是由于分布式系统中的多个节点之间需要进行数据共享和同步，而节点之间的数据和操作是不一致的，从而导致数据不一致的问题。

* 数据竞争

分布式系统中数据竞争的问题通常发生在多个进程或节点之间共享数据的时候。由于分布式系统中的各个节点通常是独立的，它们可能会同时访问和修改同一个数据对象，从而导致数据的不一致性和错误的结果。

* 操作顺序

多个节点之间的操作顺序不一致，可能会导致数据不一致。例如，分布式架构中的服务A调用服务B，发了两个请求，一个插入操作一个删除操作，本来是先插入再删除。但是很可能俩请求过去了，集群部署的情况下落在了不同机器上，可能插入请求因为某些原因执行慢了一些，导致删除请求先执行了，此时因为没数据所以没有啥效果没有啥影响；接着这个时候插入执行完了，把应该被删除的数据插入进去了，出现了错误。

* 网络延迟

由于网络延迟的存在，节点之间的数据传输可能会发生延迟，从而导致数据不一致。

* 丢失数据

在分布式系统中，可能会出现数据丢失的情况，从而导致数据不一致。例如，当一个节点发生故障或者网络连接中断时，可能会导致数据丢失，从而导致数据不一致。

因此，解决数据一致性问题的十分必要的

(翻页)

**可视化：**

TOBEDONE等项目的可视化上也有需要改进的地方。

常见的图均能用二维结构展示，关系较为复杂的图也不例外。但是，二维的图大多不能直观清晰的展现节点之间的复杂逻辑关系：例如将文件系统中大量的顶点之间所有的关系全部用连线表示，则最终的图将会极其复杂与混乱，这对于我们进行文件逻辑关系的理解是非常不利的。

如ppt图片所示

根据调研，目前的图文件系统中集成的可视化工具并不能做到对可视化的最佳展现。为此，我们计划实现对图文件系统可视化的重构。

首先，我们计划实现图文件系统的多维关联，从不同的角度对文件之间的关联进行重新定义：例如时间、地点、标题、内容等多个方面之间的关联，均可作为可视化展示中的某一维度，而在某一维度上的关联代表了两文件在该维度中具有逻辑上的相关性。

其次，我们计划实现图文件系统的多维展示，使用一些可视化工具对上述实现的多维关联进行展示。最终我们期望达到的效果是选定文件，能够按不同的维度展现出与之相关联的其他文件。

How

**总体的技术路线**

1.复现往年项目x-TOBEDONE/DisGraphFS 并在此基础上研究架构

2.可视化优化

3..在往年项目基础上解决我们提出的一致性问题

4.尝试使用更好的图数据库框架

5.进一步探索做其他改进，并对项目分析评价

复现往年项目—》正在做

**数据一致性**

为了保证图数据库，分布式文件系统间的数据一致性，采用消息中间件（MQ）传递操作信息。事务消息是 Apache RocketMQ 提供的一种高级消息类型，支持在分布式场景下保障消息生产和本地事务的最终一致性。我们使用事务消息在可视化平台与网页端，图数据库，分布式文件系统间传递信息，从而保证事务操作与消息发送要么都成功，要么都失败。

在我们的架构中，用户在网页端操作，将操作信息以普通消息的形式发往图数据库。图数据库作为生产者，执行本地事务，将操作信息以事务消息的形式发送给文件系统，文件系统收到消息后执行操作。具体步骤即为：

1. 图数据库将操作信息发送至Apache RocketMQ服务端。
2. Apache RocketMQ服务端将消息持久化成功之后，向图数据库返回Ack确认消息已经发送成功，此时消息被标记为"暂不能投递"，这种状态下的消息即为半事务消息。
3. 图数据库开始执行本地事务逻辑，即具体操作。
4. 图数据库根据本地事务执行结果向服务端提交二次确认结果（Commit或是Rollback），服务端收到确认结果后处理逻辑如下：
   * 二次确认结果为Commit：服务端将半事务消息标记为可投递，并投递给文件系统。
   * 二次确认结果为Rollback：服务端将回滚事务，不会将半事务消息投递给文件系统。
5. 在断网或者是文件系统应用重启的特殊情况下，若服务端未收到发送者提交的二次确认结果，或服务端收到的二次确认结果为Unknown未知状态，经过固定时间后，服务端将对图数据库发起消息回查。
6. 图数据库收到消息回查后，需要检查对应消息的本地事务执行的最终结果。
7. 图数据库根据检查到的本地事务的最终状态再次提交二次确认，服务端仍按照步骤4对半事务消息进行处理。

事务消息保证文件系统与图数据库的数据一致性：若图数据库执行操作不成功，则不会改动文件系统；若图数据库不能正常发送消息给文件系统，则不会进行操作。文件系统将操作结果发送回网页端。

**打标技术**

针对不同类型的文件打标思路如下：

文字将使用提取关键词的算法得到标签；语音将先转化为文字，再调用文字打标部分；图片将使用卷积神经网络的多分类实现，我们有现成的API可以调用；视频打标将截取视频片段，划归为图片打标。

我们可以直接用上面打标的成功可视化

上面说到将文件映射为向量，并且映射结果具有“聚类”的可视化方式，本思路启发于自然语言处理中的词向量模型。

词向量模型将词映射为指定维数的向量，并且语义相似的词具有相似的向量。而我们将文件视为其标签的词向量的集合，于是就将文件映射到了向量空间，并且继承了词向量的“聚类“性质，并且可以根据余弦相似度进行模糊的语义查找。

**图数据库**

项目中所使用的图数据库大多有可视化api，可以直接利用这些api进行可视化的展现。例如neo4j与可视化工具neovis.js之间可以进行直接连接，从而进行图数据库中数据的可视化。

此外还有备选方案：根据调研，目前已知图数据库可将数据输出为csv文件进行保存，将该csv文件导出至可视化工具中也可以实现文件的可视化。这样做的优点是可选的可视化工具较多，更容易实现多维的可视化；因为大多数的可视化工具可能不支持图数据库接口，但支持csv文件的输入。但该做法也有缺点，就是我们需要手动将数据进行导出和导入，提升工作量的同时，性能也会相对降低。

目前已有多种分布式图数据库，如JanusGraph。如果将这类分布式图数据库部署在项目中，需要对框架进行较大的改变。因为工作量较大，且性能的提升未知，目前我们还未确定是否要更换数据库。如有时间与精力，更换数据库是我们的可选项之一。