Part 1

在开始之前，我们先回顾一下我们组要处理的数据集，我们选用的数据集基于收集了Github上的用户行为数据的GH Archive。

数据规模是balabal截至2020年1月，Github已有超过4000万用户与2800万个公开的repo。

如图示，这是一条事务记录，"id"字段是此事务唯一独特的序号，"type"字段指定了事务类型；"actor"字段记录了事务发起人的相关信息，包括其id, login等；repo字段包含了Fork对象的repo信息，包括id, name等；"payload"字段是有关此事件的详细信息；"public"为此repo是私有还是公开的标志；"create\_at"是事件发生的标准时间。除此之外，不同时间可能也会包含其他不同字段，如"org"记录了repo所属的机构（共有账户）。

这是我们从GH Archive download 下来的一个json文件。我们选择了2020年1月1日0点到1点这第一过小时的数据作为我们本次数据存储的内容。Download下来的文件一共是107MB,34423行。Json文件的每一行，就是对应于我们上一页ppt中的一个event事务数据。

但是我们发现payload的这一条过于冗长并且过于细节与我们研究方向联系不大，正如我们上次pre中说的，我们不把它加入到数据库存储的范围。

这是payload 的例子，显而易见其冗长性，于是我们做了一个预处理把所有payload字段删除。我们分别用c++和python写了预处理程序，处理1个小时107MB的数据我们用了大概2-3秒，我们最终的需要处理一个月，也就是30\*24\*107=77040MB 即75G数据，需要0.4小时，这是完全可以接受的。

删除payload字段后，文件大小变为16MB,也就是实际有效数据规模是16MB,而我们最终要处理的实际有效数据量预计是11.2GB。

预处理完就是数据的存入了。我们决定选用3个数据库系统进行数据管理，其中一个关系数据库与两个非关系数据库。

关系性数据库相对来说，我们都比较熟悉，我们选用的是使用比较广泛的mysql数据库。原因：mysql比较常见，在关系型数据库中具有代表性。对于预处理好的数据，整个存储读入的过程我们用了3分钟。

非关系型数据库我们选择了图数据库Neo4j 和文本数据库Redis。 为什么选择图形数据库Neo4j呢？因为上一次pre中就提到，对于我们对感兴趣的在该数据集上的应用目标的预想，我们当时就想找一个图形数据库来进行存储（实际上图形数据库对于处理社交网络十分擅长，所以我们认为它在这里会有不错的表现），把数据保存为图中的节点以及节点之间的关系。文件读入存储我们用了约96s。那为什么在选择了一个图形数据库后我们又选择了第三个非关系数据库——键值对存储数据库呢？因为我们考虑到我们的data类型里面，有很多string字符串，而键值对存储数据库Redis对这样的data类型很适用，直接建立一张key-value 的大表即可处理起来很方便，操作起来也很简单。利用Redis,对于预处理好的数据，整个存储读入的过程我们只用了50秒。

接下来第2part中，我们对三个数据库的性能进行初步分析。本part中我们先对三个数据库本身优点进行讨论，再将关系性与非关系性数据库对比来进行性能讨论。

关系性数据库Mysql，我们想到有下面5条优点：

1、稳定：mysql拥有一个非常快速而且稳定的基于线程的内存分配系统，可以持续使用面不必担心其稳定性；并且能支持大型的数据库， 可以方便地支持上千万条记录的数据库。作为一个开放源代码的数据库，可以针对不同的应用进行相应的修改；

2、user-friendly: 内容和逻辑容易理解，数据库建立很直观，二维表结构是非常贴近逻辑世界的一个概念

3、MySQL同时提供高度多样性：能够提供很多不同的使用者介面，包括命令行客户端操作，网页浏览器，以及各式各样的程序语言介面，例如C+，Perl，Java，PHP，以及Python。你可以使用事先包装好的客户端，或者干脆自己写一个合适的应用程序。MySQL可用于Unix，Windows，以及OS/2等平台，因此它可以用在个人电脑或者是服务器上。

4、方便使用：通用的SQL语言使得操作关系型数据库非常方便

5、易于维护：丰富的完整性大大减低了数据冗余和数据不一致的概率

关系性数据库Neo4j，我们想到有下面5条优点：

1、高性能：Neo4j可以使用图的遍历算法设计。即从一个节点开始，根据其连接的关系，可以快速和方便地找出它的邻近节点。这种查找数据的方法并不受数据量的大小所影响，因为邻近查询始终查找的是有限的局部数据，不会对整个数据库进行搜索。所以，Neo4j具有非常高效的查询性能，而且查询速度不会因数据量的增长而下降，即数据库可以经久耐用，并且始终保持最初的活力。

2、Neo4j的数据库有设计的灵活性。因为随着需求的变化而增加的节点、关系及其属性并不会影响到原来数据的正常使用，所以使用Neo4j来设计数据库，可以更接近业务需求的变化，可以更快地赶上需求发展变化的脚步。相比大多数使用关系型数据库的系统，为了应对快速变化的业务需求，往往需要采取推倒重来的方法重构整个应用系统。而这样做的成本是巨大的。使用Neo4j可以最大限度地避免这种情况发生。虽然有时候，也许是因为最初的设计考虑得太不周全，或者为了获得更好的表现力，数据库变更和迁移在所难免，但是使用Neo4j来做这项工作也是非常容易的，至少它没有模式结构定义方面的苦恼。

3、图数据库设计中直观明了的数据模型，从需求的讨论开始，到程序开发和实现，以及最终保存在数据库中的样子，它的模样似乎没有什么变化，甚至可以说本来就是一模一样的。这说明，业务需求与系统设计之间可以拉近距离，需求和实现结果之间越来越接近。这不但降低了业务人员与设计人员之间的沟通成本，也使得开发更加容易迭代，并且非常适合使用敏捷开发方法

4、图的数据结构直观而形象地表现了现实世界的应用场景

5、建模十分方便，使用简单框图就可以设计数据模型，方便建模

非关系性数据库Redis，我们想到有下面5条优点：

1、Redis不仅仅支持简单的key-value类型数据，还支持包括set,zset,list,hash,string这五种数据类型，操作非常方便。比如，如果你在做好友系统，查看自己的好友关系，如果采用其他的key-value系统，则必须把对应的好友拼接成字符串，然后在提取好友时，再把value进行解析，而redis则相对简单，直接支持list的存储(采用双向链表或者压缩链表的存储方式)。

2、Redis还支持 publish/subscribe, 通知, key 过期等等特性

3、由于是全内存操作，所以读写性能很好，性能极高 – Redis能读的速度是110000次/s（10w/s的频率）,写的速度是81000次/s 。

4、原子 – Redis的所有操作都是原子性的，同时Redis还支持对几个操作合并后的原子性执行。（事务）

5、内存数据库，速度快，也支持数据的持久化，可以将内存中的数据保存在磁盘中，重启的时候可以再次加载进行使用。

5的解释：Redis为了达到最快的读写速度将数据都读到内存中，并通过异步的方式将数据写入磁盘。所以redis具有快速和数据持久化的特征。如果不将数据放在内存中，磁盘I/O速度为严重影响redis的性能。在内存越来越便宜的今天，redis将会越来越受欢迎。 如果设置了最大使用的内存，则数据已有记录数达到内存限值后不能继续插入新值。

而且redis内存数据集大小上升到一定大小的时候，就会施行数据淘汰策略。

Mysql为什么需要3分钟呢？因为它还要检查是否满足约束，所以慢一些，但是

Neo4j 通过python接口与通过csv文件速度差距极大：前者远慢于后者。原因未知，怀疑是提供的python接口存在问题。但后者只用了96s体现了其优越的读入性能。

Redis为什么这么快呢？比图形数据库读入还快。因为它是内存数据库，具体点说它为了达到最快的读写速度将数据都读到内存中，并通过异步的方式将数据写入磁盘。所以redis具有快速和数据持久化的特征。如果不将数据放在内存中，磁盘I/O速度为严重影响redis的性能。在内存越来越便宜的今天，redis将会越来越受欢迎。

在仔细观察了数据集中所包含的各个数据及其对应意义后，我们对于该数据集上的应用目标有了初步的认识并提出了一些初步的设想以供后续参考。

关于下一阶段的想要实现的实验目标，我们计划是1、把完整的一个月数据读入存储三个数据库，并记录时间来进一步分析个数据库读入速度性能方面的性能。2、在同样的机器上对三个数据库做相同的增删改查工作，记录相关的实验时间以进一步分别分析增删改查方面的性能。3、我们将改变数据库结构，增添一条actor的属性“org”,并对比增加这一org所需的各个数据库的时间，以进一步进行性能分析。

以上便是我们本次报告的内容，感谢大家的聆听。