gstore jena sesame virtuoso

gstore 和其他 RDF 数据库的测试与对比

目录

背景介绍

gstore 是一个基于图的数据库系统，比较好地保持了原有 RDF 数据（RDF 的介绍请参见附录[1]）的图结构。它的数据模型是带标签的有向图，每个顶点对应着一个 subject 或者 object。对于每个 sparql 查询（目前 gstore 只支持无更新的查询操作），gstore 也会将其表示为一个图，从而将一个 sparql 查询转化为一个子图同构问题。gstore 在 RDF 图上建立 vs-tree 索引来加速匹配过程，先过滤得到候选集，再作 join 操作得到最终结果集。

（关于 gstore 的详细介绍、使用方法、API 和源码等，请参见附录[2]）

我们在此选用了 jena（附录[3]）、sesame（附录[4]）、virtuoso（附录[5]）数据库来与 gstore 共同作测试并对比，需要对比的内容主要是加载数据集的时间、加载数据集后的数据库大小、单个查询的时间比较、单个查询的结果匹配。（内存消耗除了特别情况，在此不予比较）

2015 年 10 月 9 日

gstore jena sesame virtuoso

在测试之前需要约定规范，以及其他准备操作。必须先确保所有数据集和对应查询正确无误（能够正常地运行结束，而不会崩溃），至少必须满足这几个 dbms 的格式要求。如 gstore 不支持对谓词（predicate）的查询，jena 不支持使用 prefix 声明和使用^^连接字符串的数据集，sesame 不支持无前缀的 subject 等等。对于这些情况，我们需要修改原数据集或者查询来满足所有数据库的共同标准。（更多的格式问题请参见附录[8]）

另外，在测量查询时间时，不能把加载数据库索引的时间（即 offline 时间）也考虑在内，而只应考虑单个查询消耗的时间（即 online 时间），并在每个数据库引擎每测试完一整个数据集时清空缓存。

我们会在 WatDiv、LUBM、BSBM、DBpedia 这些数据集上作查询，其中有些是网站的后台数据，有些是人工生成，查询有些是程序生成（对于人工数据集），有些是我们编写。（关于这些数据集和查询的具体情况，请参见附录[6]）

实验环境是一台拥有 82G 内存 7T 硬盘的 linux 服务器，安装了 centos 2.6.32-573.3.1.el6.x86_64 系统。为满足测试要求，glibc 库至少是 2.14 版本。

几种数据库的版本(均为开源版本)：gstore 为最新版，jena 为 3.0.0 版，sesame 为 2.7.0 版，virtuoso 为 7.2 版本。

程序和功能

首先为 gstore 扩充了一个 gtest 功能，可以对某个数据集和相应查询作测试，也可以对某种类型的所有数据集和相应所有查询作测试，具体使用方法可通过./gtest -h 得知。（其他功能以后再扩充）

另外，编写了一个统一测试程序 full_test.sh，该脚本枚举了所有数据集和所有查询，分别用不同的数据库查询引擎进行测试，并最终生成一系列 tsv 文件。（这些 tsv 文件可直接导入 excel 表格中，须设置仅以 tab 键分割）

在使用 full_test.sh 时，需要注意的是，根据实际情况更改脚本中 dbms 的位置和各数据集的位置，且数据集的放置必须遵循规范：如 WatDiv 等类型的数据集全部放在同一文件夹下（/media/wip/common/data），且每种类型数据集的文件夹中，database 文件夹下放置该类型的各个数据集（以.nt 命名），query 文件夹下放置针对该类型数据集的所有查询（以.sql 命名）

2015 年 10 月 9 日

gstore jena sesame virtuoso

因为 gstore 和 jena 都对外部命令行有比较好的支持，所以在 full_test.sh 中直接调用相关命令，然后用 shell 命令对输出进行分析即可。针对其他数据库如 sesame 和 virtuoso，可以直接在其相应控制台操作，也可以另行根据相应 API 编写程序输出结果，并最终在 full_test.sh 中嵌入相关命令并分析结果。（注意 full_test 是 shell 脚本，第一行指定的解释器需要根据实际情况确定，可能是/bin/bash，也可能是/usr/bin/bash）

测试结果

测试结果保存在与 full_test.sh 同一目录下的 load.log/，time.log/和 result.log/中，时间均以 ms 为单位，空间以 kb 为单位。

其中，load.log/下是 size.tsv 和 time.tsv 文件，size.tsv 文件对应着各种数据集导入各种数据库引擎后的数据库大小（但 virtuoso 是将每个数据集存为同一数据库的一个表，本质上是关系数据库，和 sesame 一样不方便统计大小），而 time.tsv 文件对应着各种数据集导入各种数据库引擎的时间。

在 time.log 中，是以各数据集命名的文件如 watdiv_60.nt.tsv，每个文件对应者该数据集的各个查询在不同数据库引擎中消耗的时间。

在 result.log 中，也是以各数据集命名的文件如 watdiv_60.nt.tsv，但文件的列不再是数据库引擎而是数据库引擎的对比如 gstore_jena，对于每个查询，两个数据库引擎的查询结果若匹配，则在对应位置填 Y，否则填 N。（但 sesame 的控制台中只会输出查询所消耗的时间，不会输出查询结果，所以该数据库引擎无法据此比较）

（结果统一采用 tab 键作为唯一分隔符，具体细节参见附录[7])

对比分析

对 result.log 分析后可以发现，在各个数据库引擎上，所有数据集的所有查询结果都是匹配的（不会输出结果的话则忽略），这说明至少在这些数据集和这些查询上，gstore 的正确性是无误的。

对 load.log 分析后可以发现（因为 sesame 和 virtuoso 中无法获取生成的数据库大小，所以这里的空间开销仅比较 gstore 和 jena），gstore 的 load 过程无论是时间开销还是空间开销

gstore jena sesame virtuoso

一般都要大很多，基本上都要差一个数量级。

对 time.log 分析后可以发现（只关注运行时间相差 20%以上的情况），各种查询的时间比较相对复杂，基本上比较复杂的查询（比如有很多待查变量）gstore 会表现更好。但存在少数查询输出结果太多，gstore 将会有大问题（比如崩溃），为了统一测试的需要在本次测试中忽略这些查询。

就 WatDiv 系列的数据集而言，运行 C3.sql 时 gstore 表现更好，而在运行 F3.sql 时 jena 等数据库表现更好。在运行其他查询时，各种数据库引擎的效率相似，查询时间没有数量级的差别。

就 LUBM 系列的数据集而言，运行目前已有的那些查询时，各种数据库引擎消耗的时间大体相同。

就 BSBM 系列的数据集而言，在 self1.sql，self3.sql，self8.sql 这些查询上，gstore 都要比 jena 等数据库慢。但在 self7.sql 这个查询上，gstore 要比其他几个数据库引擎都快上不少。

就 DBpedia 系列的数据集而言，在 q3.sql 和 q4.sql 这些查询上，gstore 不如 virtuoso 等几个数据库，在其他查询上 gstore 也没有表现出明显优势。

关于内存消耗没有具体统计，大概观察后发现，gstore 的内存消耗高于 jena、sesame、virtuoso 这些数据库引擎，并且越是大的数据集和复杂的查询这个问题就越明显。

结论与展望

gstore 的优势：可以处理广泛的 RDF 数据集（其他数据库引擎在处理各种 N-Triples 数据集时会有很多格式规范问题，参见附录[8]）；在某些特别复杂的查询上非常高效；基于图模型而非关系模型，还有很大的优化空间。

功能方面的差距：只支持 N-Triples 格式的 RDF 数据集；内存和磁盘开销非常大；只能处理无更新的查询操作，并且不能处理聚集查询，也不支持对谓词的查询；所有中间过程的结果全部放在内存，并且输出时采用 streambuf 机制，因此若查询的结果集非常大系统将崩溃；命令行支持不全面也不友好，无命令/路径补全机制，也无法记录历史命令。

设计和代码层面的问题：编译过程有大量的重定义警告；代码可读性不算特别好，注释较少；某些功能相似的类被多处定义，而某些地方功能还很不完整。

数据集和查询方面的问题：很多数据集没有基本的说明，查询数量和质量也不够，某些数据集还不是特别规范。

未来的方向和计划：

gstore jena sesame virtuoso

1. 整理并扩充数据集和查询。目前使用的是 WatDiv、LUBM、BSBM、DBpedia 这四个，但收集的数据集还包括 yago2、yago3 和 DBLP 等。另外几个数据集因为格式问题导致目前无法用于测试，查询也还没有，之后可以修正数据集并补上查询。

2. 添加对聚集查询的支持（如 union 操作和数值型查询等，要注意性能优化），完善命令行。可以在服务器上安装使用 readline 库，借以为 gstore 实现命令行补全和历史命令搜索等功能，并考虑为每个命令提供帮助信息。

3. 修正 bug，改善设计。首先需要修改负责输出结果的代码，不应该继续使用 streambuf。将几个功能极其相似的类（如 Bstr）合并为一个或定义父子类，另外实现一个 Result 类负责中间结果的转存。还可以实现一些通用的数据结构和更多的辅助算法，均可以作为库使用。有条件时可以考虑重构现有代码（应在一个头文件中 typedef 定义所有类型）。

附录

[1] RDF 格式参考：http://www.w3school.com.cn/rdf/

[2] gstore 详细情况：https://github.com/Caesar11/gStore

[3] jena 官网：http://jena.apache.org/

[4] sesame 官网：http://www.rdf4j.org/

[5] virtuoso 官网：http://virtuoso.openlinksw.com/

[6] 数据集和查询：以 guest 账户访问 59.108.48.36(SSH 协议)，在/media/wip/common/data/文件夹下

[7] 测试程序和结果：gtest　　full_test　　xlsx 结果　　tsv 结果

[8] 格式问题：format_question

2015 年 10 月 9 日