数据库决赛-现场

决赛题目描述

题目要求选手实现一个或多个压缩算法，将指定的文件进行压缩，并能够解压后读取数据。
在数据库中，特别是OceanBase，底层存储为了节省空间，常常会对数据文件进行压缩。一个好的压缩算法，常常会节省很多成本。不过，因为数据存储都是为了读取，因此在考虑压缩率的同时，还需要考虑数据解压出来的效率，就是读取压缩数据的效率。
选手需要在miniob的基础上（可以在最原始的仓库代码中开始做，建议分支 miniob_competition），我们会创建一个表，并导入大量数据，选手需要对表数据文件和索引文件进行压缩。数据导入完成后，测试程序会随机选取一些数据进行查询。选手需要修改miniob代码，在一定时间范围内，完成数据导入和数据查询。

如何评判优胜

评判指标：压缩率

压缩率

压缩率是指压缩后文件大小，与原文件大小的比值。源文件大小是指当前未修改的miniob，导入表数据后，数据文件占用的磁盘空间大小（当前给出一个评估值）。压缩后的文件大小，是指选手对miniob保存文件数据的格式经过调整，保存的文件大小。获奖选手需要介绍自己使用的压缩算法及其原理。
当然，首先需要判断程序能够通过导入数据和查询数据的测试，即在一定时间内完成数据的导入和数据的查询，并且查询数据正确。

分数的计算

上传到英雄榜的分数，如果通过压缩率计算分数的话，排名是从小到大，因此为了得到一个从大到小排序的分数，英雄榜分数计算公式是
50000000000 ÷ 选手生成文件总大小
本项最终分数，按照之前性能分数的经验，仍然是 选手分数/最高分数*100。

测试流程

1. 清理测试环境。每次测试前，选手的测试环境，都会清空；
2. 拉取测试代码。代码尽量减少依赖，不需要提交build目录，拉取代码超时时间10分钟；
3. 编译；
4. 创建表；
5. 导入数据。注意超时时间；
6. 创建索引。同上；
7. 计算文件大小。会计算在当前程序默认的目录 miniob/db/sys 目录下的所有文件大小总和；
8. 测试查询数据；
9. 测试结束，上传结果。

所有带有超时时间限制的步骤，预留了一定的缓冲，所以出现超时，请检查是否优化不到位。

NOTE：查询测试时，会传入索引的值。
导入数据量：300万（条）
导入数据时间限制：30秒
创建索引：30秒
查询数据量：30000（条）
查询时间/正确性验证限制：20秒（大约3万条查询）

测试环境

CentOS 7.9
Linux kernel 3.10.0
GCC 8.3.0

表结构与数据来源

表结构

create table t1 (c1 int, c2 int, c3 int, v1 char, v2 char, v3 char, v4 char, v5 char, v6 char, v7 char, v8 char, v9 char);
第一个字段`c1`是严格单调递增的数字，c2和c3是随机生成的数字，可以使用int32_t存放。v1~v9，是随机生成的3字节字符串。注意，虽然是随机生成，但是有些列会具有一定规则。

导入数据的语句

load data infile '/home/test/game/final_miniob/etc/miniob-300w.table' into table t1;
NOTE：已经具备的功能，选手不需要实现。

创建索引

create index i1 on t1(c2);

测试查询的语句

select * from t1 where c2=`randint`;

查询语句都是等值查询，并且会使用索引。

查询输出格式

本次测试都是单表查询，查询结果中先输出表头，然后输出数据行。数据行输出以'\n'分隔每行，输出不区分先后顺序。输出格式示例：
c1 | c2 | c3 | v1 | v2 | v3 | v4 | v5 | v6 | v7 | v8 | v9
2945817 | 5905511 | 159221 | abc | abc | abc | abc | abc | abc | abc | abc | abc
表头每个字段之间、行数据每个字段之间，都使用' | '分隔（空格、竖线、空格）。
查询没有数据时，返回表头信息。

可以选择的压缩算法

本次比赛不允许使用通用压缩库和通用压缩算法，比如lz4、zstd、snappy等，只能选择以下指定的一种或多种压缩算法，压缩算法可以混合使用，并由自己实现。

- 差值编码（Delta Encoding）/ Double Delta
- 字典编码
- 哈弗曼编码

选手需要做的工作

选手们需要做的工作包括但不限于：

- 修改disk_buffer_pool,支持导入大量数据；
- 修改disk_buffer_pool，支持高效的查找页面；
- 修改bplus_tree，支持高效的导入数据；
- 考虑压缩算法和策略以及page数据的组织形式；
- 修改查询语句输出格式满足要求。

其它规则

测试查询数据时，禁止将数据直接放在内存，而不是buffer pool中，或类似的缓存手段。因为查询性能不是考察的目标，所以也不需要这么做；
禁止调整当前程序生成的目录结构与文件名称，禁止使用额外的文件或目录存储数据。即只允许在miniob/db/sys下创建文件；
符合以上禁止条件的，判断为作弊行为；
允许针对当前指定表结构做设计，可以根据表中每个字段的特征，选择合适的压缩算法。

测试答疑

提测平台提交测试后，测试结果可能会失败，比如出现提示“代码clone失败”、“编译失败”、“验证失败”等。
当前会有这些错误提示：

- branch不存在。请确认提测时填写内容是否准确；
- commit_id不存在。请确认提测时填写内容是否准确；
- 编译失败。请确认编译是否有问题。如果是clang编译器，需要切换为gcc编译器。gcc与clang在某些细节上表现会不一致。建议选手使用与测试后台相同的环境；

- git clone失败。先排查是否代码仓库邀请了 hnwyllmm 共建，代码平台是否为gitee。如果没有问题，可能clone超时，可以再次尝试。如果还有问题，联系工作人员辅助查找问题；
- git reset失败。通常会由于对应代码分支上没有对应的commit id；
- cmake失败。与编译失败类似，请尽量保证环境没有问题；
- 验证失败。出现验证失败的原因比较多，通常是由于选手代码执行过程出错，或某个环节执行超时导致的。比如导入表失败、构建索引失败、正确性验证失败。这些阶段超时也会上报“验证失败”。建议选手进行比较详细的测试，如果查不到原因，可以联系工作人员，工作人员能告知的信息只能到是哪一个环节出现问题，比如导入表失败、构建索引超时，但是没办法提供错误出现的具体原因；
- 其它异常。请直接联系工作人员。

选手遇到问题首先应该自查，如果自查无果可以联系工作人员。但是由于选手代码导致的问题，工作人员不会告知详细信息。比如由于表数据导入失败导致的“验证失败”，工作人员不会告知。