Project Report - Implementing a Sorted File

Arpan Banerjee

UFID: 9359-9083

[arpanbanerjee@ufl.edu](mailto:arpanbanerjee@ufl.edu)

Krutantak Patil

UFID: 5615-6343

Krutantakb.patil@ufl.edu

***1) Steps to compile and run - code, tests and gtests***

*(*Assuming .bin files would be provided to us in dbfiles folder)

1. **make main** – Command to build the main program.
2. **./main** – Command to run the main executable.
3. **make test.out** – Command to build the tests.
4. **./test.out** – Command to run the tests.
5. **make bigQTest.out** – Command to build the gtests.
6. **./gtestScript.sh** – Command to run the gtests. (gtest folder contains data required for running 4 test cases written in BigQTest.cc file)

***2) BigQ file implementation***

**Data members**

1. **int runlength** – Used to store length of each run.
2. **File runFile** – Used to store records and pages in the file.
3. **OrderMaker sortOrder** – Pointer to page holding recent record additions to the file.
4. **Pipe \*in** – Pointer to produce records and add to pages from .bin files.
5. **Pipe \*out** – Pointer to consume sorted and merged records from .bin files.

**Member functions**

1. **BigQ** ()
   1. *Default constructor used in BigQTest.cc to initailize BigQ object for testing*.

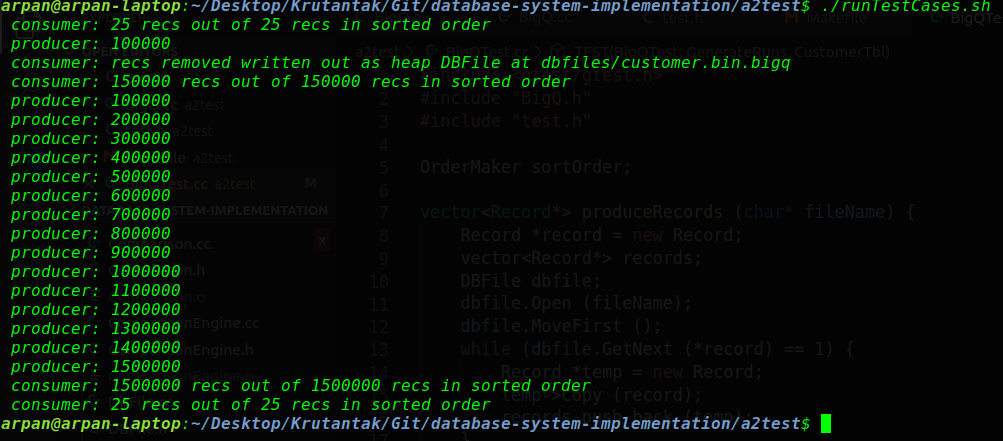
1. **BigQ** (Pipe &in, Pipe &out, OrderMaker &sortorder, int runLength)

* 1. Paramaterized constructor called to read data from input pipe, sort it using two-pass multiway merge algorithm and then store using output pipe.
  2. A worker thread is created using pthread in this constructed.

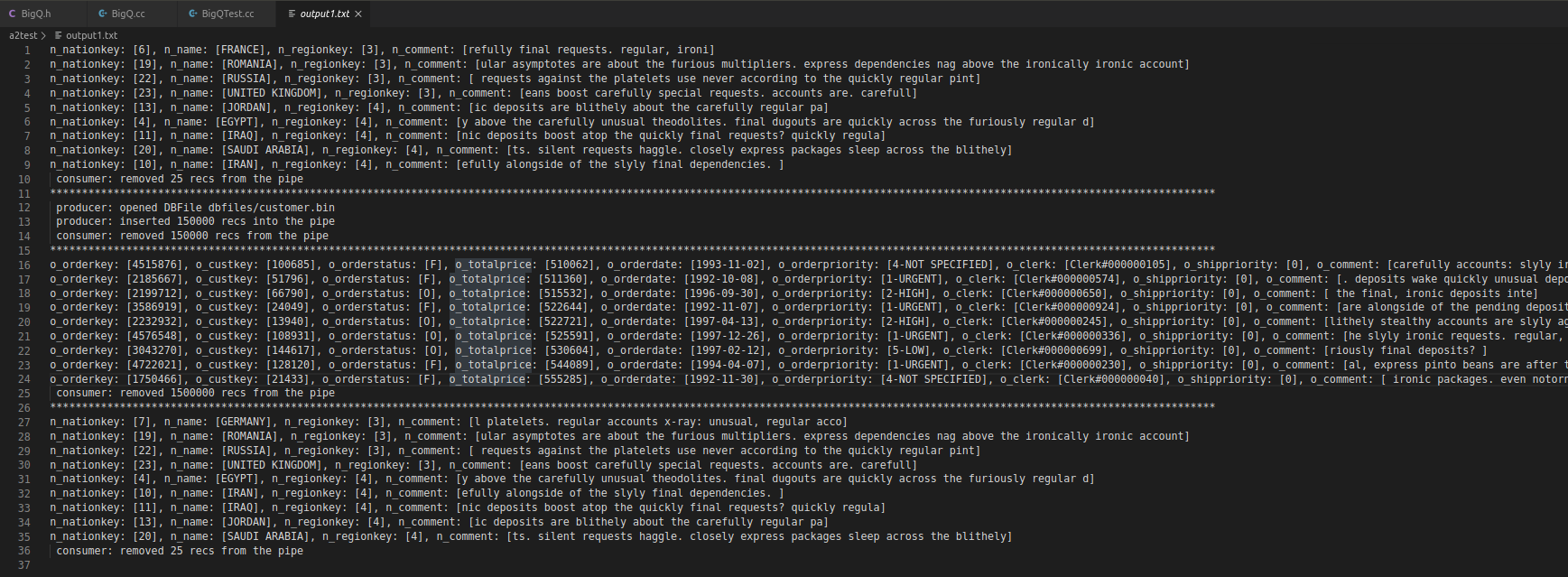
1. **worker** (void \*workerThread)
   1. It is a worker routine which first reads data from input pipe and computes sorted runs. Then, using priority queue, it performs k-way merging and stores back using the output pipe.
   2. runCount is initialized to 0.
   3. overflow map is used to store overflow papes.
2. **initFile** (File &runfile)
   1. This function is used to create and open an in memory file used to stored sorted runs.
3. **tpmmsPhase1** (Pipe \*in, OrderMaker \*sortOrder, int &runCount, int runLength, File &runFile, map<int,Page\*> &overflow)
   1. Its the phase 1 of Two-Pass Multiway Merge Sort Algorithm.
   2. Records are retrieved from the inpute pipe and added to pages until the page is full. Once the page is full, the next record is added to a new page and page counter is incremented by 1.
   3. If, either there are no more records in the inpute pipe or page count equals runLength, then pages sorted and stores in in-memory file as runs.
4. **generateRuns** (vector<Record \*> &records, int &runCount, int runLength, File &runFile, map<int,Page\*> &overflow)
   1. Records are sorted using Phase1Compare operator and stored as runs.
   2. After sorting and generating runs, if, pageCount is equal to or more than runlength, then pages are stored in the overflowmap.
5. **tpmmsPhase2** (Pipe \*out, OrderMaker \*sortOrder, int runCount, int runLength, File &runFile, map<int,Page\*> overflow)
   1. Its the phase 2 of Two-Pass Multiway Merge Sort Algorithm.
   2. Priority queue is used to merge k-runs.
   3. Runs are merged using Phase2Compare operator and then written to output pipe.
   4. K-way merging algorithm is used to read runs and merge them
6. **Phase1Compare**(OrderMaker \*inSortOrder)
   1. It is used to sort records using CNF order and ComparisonEngine class.
7. **Phase2Compare**(OrderMaker \*inSortOrder)
   1. It is used to merge records using CNF order and ComparisonEngine class.

***3) ./test.out results for 1GB data***

1. **./runTestCases.sh**



1. **output1.txt**

******

***5) gTest results:***

1. **TEST(BigQTest, GenerateRuns\_OrderTbl) –** 
   1. This function produces records from gtests/orders.bin file, and test whether runs are created and written to in-memory file. Here we check it for records for multiple and overflow pages.
2. **TEST(BigQTest, GenerateRuns\_NationTbl) –** 
   1. This function produces records from gtests/nation.bin file, and test whether runs are created and written to in-memory file. Here we check it for records for a single page.
3. **TEST(BigQTest, Phase1Compare) –** 
   1. This function assert sorting of records using Phase1Compare operator. Records are produced from gtests/actual.bin file and then sorted. Here we match the sorted record vector with the expected record vector produced using gtests/expected.bin file.
4. **TEST(BigQTest, Phase2Compare) –** 
   1. This function assert sorting of records using Phase2Compare operator. Records are produced from gtests/actual.bin file and then sorted. Here we match the sorted record vector with the expected record vector produced using gtests/expected.bin file. Since the sorting techinque is not consistent, we ASSERT\_FALSE here.

(ensure that gtest folder contains actual.bin, expected.bin, nation.bin, & orders.bin files required for testing BigQ.cc file)

