Petrolink Challenge

André Sampaio

January 28, 2018

1 Approach

Considering the challenge description, the tasks to perform are:

1. Read data from the provided SQLite database;
2. Perform the Minimum Curvature Method calculations to determine the experienced progress: *i)* Determine Beta, RF; *ii)* Determine North, East and TVD; *iii)* With the Values from *ii*), calculate new coordinates. Repeat from step 1;
3. Output the results in different formats.

1.1 Proposed Architecture

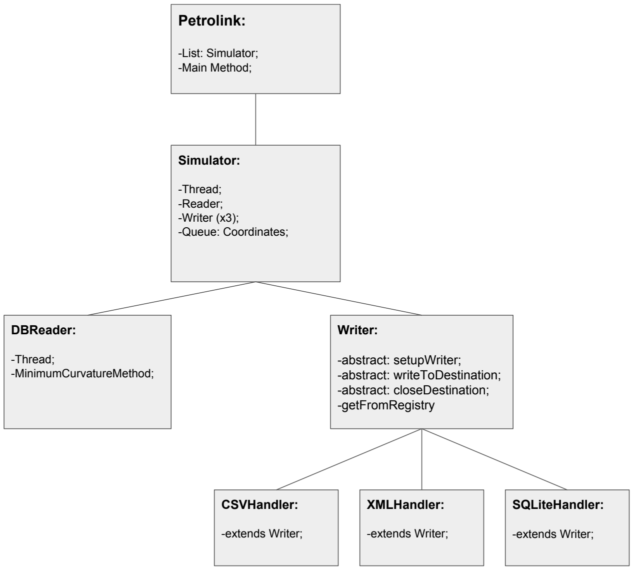

Figure 1: Simplified Class Diagram

Figure 1 provides a very basic representation of the implemented Class Diagram. The Petrolink class allows the system to perform various simulations at the same time, with each Simulator working independently.

Each Simulator contains a DBReader, which is a Thread (it needn't be in the scenario where no concurrency is considered however it was still implemented that way). Each Simulator also contains a writer logic (responsible for writing results) considering three different threads (one per type of destination file) and a shared queue, where the Reader inserts the calculated results and, ”at the same time”, the writers extract data so they may save results in the requested formats. This means that actions can happen separately/concurrently, as logically demonstrated in Figure 2.

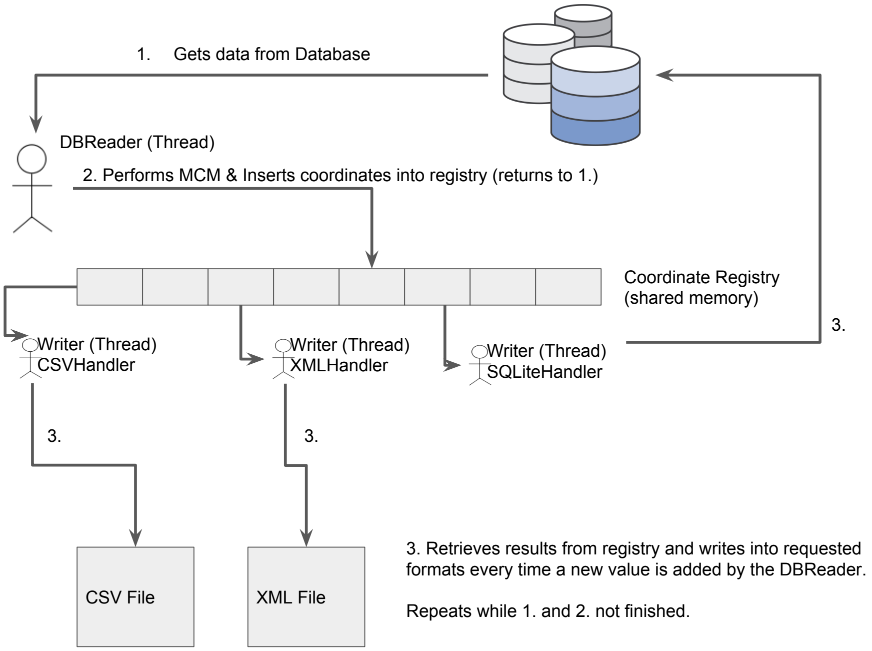
1. Gets data from Database
DBReader (Thread)
2. Performs MCM & Inserts coordinates into registry (returns to 1.)
Coordinate Registry (shared memory)
3.
Writer (Thread) CSVHandler
Writer (Thread) XMLHandler
Writer (Thread) SQLiteHandler
3.
3.
CSV File
XML File
3. Retrieves results from registry and writes into requested formats every time a new value is added by the DBReader.
Repeats while 1. and 2. not finished.

Figure 2: Sequence Logic

2 Performance

While the (time) gains from implementing the system in a concurrent manner don't seem big, with a bigger dataset the difference would expand. On average, a 100ms time gain was noticed if no management is done on the way the SQLite Handler performs its ”Insert” duties. However, should inserts be grouped in bundles, time gains can be seen much more clearly. Bundles of 2 reduce the time in half, etc. For instance, with a 200 unit bundle where only one database interaction is required for inserting, the process is completed approximately in half a second.