Lab 10 - Applying indexes to parabond

The goal of this lab is to experiment with database indexes.

Preparation
If you have not done the prior lab, **STOP**. Go back and complete that lab before starting this one.

This lab assumes you have started Mongo and initialized parabond and everything is setup to execute the parallel processing code..

Background
Indexes can dramatically speed up queries in a database. Without indexes, Mongo does a collection scan, from the first document to the one you're looking for--or the last one, if the one you're looking for isn't found. The complexity is O(n).

To index a collection, you tell Mongo which field or combinations of fields to index and it builds a lookaside collection that contains a binary tree to search the collection in O(log n) time.

Creating indexes in Mongo is very simple as you'll see below.

To test these ideas, you are going to run an application, `Par00.scala`, which prices bond portfolios using parallel collections. The portfolios are stored in Mongo. By default, Par00 prices 100 randomly selected portfolios of 100,000 portfolios and writes the runtime statistics in a diagnostic file, `Par00-dat.txt`. This file contains one record per run with six parameters:

Table 1. Runtime diagnostic output format

Parameter	Meaning
N	Number of portfolios
dt(1)	T_1
dt(N)	T_N
cores	Number of cores
R	Speed up
e	Efficiency

Note: Ser00.scala contains the serial algorithm. But then, how does Par00 calculate T_1 if Par00.scala doesn't invoke nor is it in any way connected to Ser00.scala? Look through Par00.scala and see if you can figure it out.

Tasks

Part I - Run a preliminary test.
You are ready to run an initial parallel processing test.

1. Create a new Run configuration as follows: Run > Application > + >

   Main class: `parascale.parabond.test.Par00`

   VM options: `-Ddir=`*`path`* `-Ddetails=true`

   where *path* is the directory path of a temporary folder to hold the runtime diagnostic report. For instance *path* defaults to c:\tmp\ on Windows.

   estimates R and e. In the console, it also outputs the

2. Right click on Par00 > Run > Par00.main.

3. Lookup portfolio 32075 using Mongo and check its price against the console output. They should be identical.

4. Open the file, `Par00-dat.txt`. Recalculate by hand R and e. Is the diags file right?

   Note: Do your calculations on the basis of "logical" cores, if there is such a thing because Par00 gets cores on the basis of `Runtime.getRuntime.availableProcessors`.

5. Download the lab spreadsheet and add T1, TN, and R values to the non-indexed columns for n=100 portfolios.

   Note: The spreadsheet contains computed values for R. However, you'll note that Table 1 has measured values of R. You may use either measured or computed values of R except for the gray cells. Leave the gray cells as computed values.

Part II - Run more tests.

1. Edit the Run configuration for Par00 and add this extra VM option:

   ```
   -Dn=1000
   ```

   Remove this option:

   ```
   -Ddetails=true
   ```

2. Run Par00, get the statistics from the diags file and add them to the spreadsheet for the non-indexed columns.

3. Run the test for `n=2000` portfolios and record the statistics in the spreadsheet.

Part III - Index the database.

1. In the Mongo shell, index the Bonds collection as

   ```
   db.Bonds.createIndex({id: 1});
   ```

   and Portfolios as:

   ```
   db.Portfolios.createIndex({id: 1});
   ```

   These commands tell Mongo, respectively, that for a query with *id* field, use indexes. It turns out that Par00 ONLY searches for bonds and portfolios using their respective unique id numbers.

2. To verify that indexes are in fact being used, run these commands:[1]

   ```
   db.Bonds.getIndexes();
   db.Portfolios.getIndexes();
   ```

Part IV - Rerun tests using indexes

1. Repeat the above test for 100; 1,000; and 2000 portfolios and record the results in the spreadsheet.

   Note: the "Indexed R" columns measure, respectively, the speedup in just the indexes alone. See the equation in the spreadsheet.

2. What general conclusions can you draw about the performance of indexes?

3. You may find that the efficiency drops with the indexed database. How would you account for this?

Deliverables

1. Answer the question about T_1 in the "Background" section and the questions in Part IV.2, 3. Type your responses in the shell.

2. Upload the diagnostics file, `Par00-dat.txt`.

[1] To remove indexes do `db.Bonds.dropIndexes();` and `db.Portfolio.dropIndexes();`.

3. Print a copy of the spreadsheet and upload it as a *PDF* -- not an *XLSX*.