departamento de informática
FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Concurrency and Parallelism 2017-18
Home Work 2 — LinkedList

Objectives

By doing this homework you will learn how to make use of different techniques to properly synchronize the concurrent accesses to an IntSet implemented as a LinkedList. You are asked to both make the benchmark program work correctly in a concurrent environment and to evaluate, compare and contrast the performance of each of your implementations in different settings of workload and parallelism.

Important Dates

2017-11-01 *Home work 2 assignment document v1.0*

2017-11-27 @ 23:59 *Home work 2 code and report submission*

Introduction

In this homework assignment, you are asked to parallelize the implementation of a Java program that uses a linked list to implement a set-of-integers.

You are given a full Java program that compiles and works and that prints some information when executed. The printed information refers to both the phase of the benchmarking program under execution and some statistics about its behaviour.

To improve the quality of the information (statistics) reported, the benchmark starts running in a *warmup phase* for two seconds (where the list will be populated according to the workload specified). Once the warmup phase is concluded, it will switch to the *execution phase*, where statistics will be collected (the warmup phase is ignored in the statistics).

The implementation of the linked list you are given makes use of sentinel nodes and is correct when executed in a sequential setting. You are expected to study that sequential version and, once you understand how it works, develop multiple concurrent versions using different synchronization techniques.

For each of the concurrent versions of the list you develop, you are expected to run the benchmarking application under different conditions of concurrency and workload, and understand how and when each concurrent version performs well or bad, defining in which contexts one technique is preferable over the other(s).

Executing the program

The program accepts a large set of command line arguments, but you are expected to fiddle with just a few of them. To make your life easier, in the folder “scripts” there is a shell script “intset.sh” that accepts less arguments and launches the program providing default values for the remainder.

The “intset.sh” shell script accepts at most three arguments. The first argument is obligatory and indicates the running time (excluding the warmup phase) for the benchmark. The second

argument is the number of threads and defaults to 1. The third argument configures the workload by indicating the percentage of *write operations*, and defaults to 50%.

The shell script takes one to three arguments using the following syntax:

```
./scripts/intset.sh TIME [NTHREADS] [WRITE]
```

where the arguments are:

Argument	Semantics
TIME (*obligatory*)	Run the benchmark program for <TIME> miliseconds. Note that this time excludes the *warmup phase*.
NTHREADS (*optional*)	Execute the program with <NTHREADS> threads.
WRITE (*optional*)	Percentage of write (list modifying) operations. The remainder operations (100 – <WRITE>) will be read (non-modifying) operations.

Workplan

Fork the source GIT repository

Start by using the web interface of your favourite git hosting site to fork/import the original repository for Home Work 2 to your own account, naming your new private copy as below:

`cp-2017-18-gXX-hw2`

where "gXX" is the group number, available at:

https://docs.google.com/spreadsheets/d/114aBEtMMUFCiUZDKoTLhntoM-whJicoKO9sEN2BGl_Og/edit#gid=0

The original repository is here:
https://joaomlourenco@bitbucket.org/cp201718team/cp-2017-18-hw2_intsetlinedlist.git

To fork/import the repository you can use the following links for GitHub and BitBucket:

https://help.github.com/articles/importing-a-repository-with-github-importer/

https://confluence.atlassian.com/get-started-with-bitbucket/import-a-repository-861178561.html

Next, clone your new repository to create a local copy in your machine.

```
git clone URL_OF_YOUR_REPOSITORY
```

This is a Java program, so you may use your favourite editor (Eclipse?) to edit, compile and run the program.

If you prefer to compile and run the program form the command line (10 seconds, 4 threads, 25% of updates) , you can do it with the following commands (applied in the root of the cloned repository):

```
find . -name '*.java' -print | xargs javac -d bin
./scripts/intset.sh 10 4 25
```

Inspecting the source code

Open the source code in your favourite editor (e.g., eclipse) and study it. Almost for sure you will find some things which you are not familiar with. Talk with the teacher about them.

Start implementing the concurrent version(s)

We provide a file for each of the proposed concurrent versions, but these files are all replicas of the base sequential file (named '*IntSetLinkedList*'), where the class name and constructor was replaced appropriately.

Class / File	Linked List Version
IntSetLinkedList	Sequential (unprotected / unsynchronized) version of the Linked List. Works all right with one thread. Fails with two or more threads.
IntSetLinkedListSynchronized	Concurrent version of the Linked List protected with Synchronized statements.
IntSetLinkedListGlobalLock	Concurrent version of the Linked List protected with a global (exclusive) lock from the java.util.concurrency package.
IntSetLinkedListGlobalRWLock	Concurrent version of the Linked List protected with a global read-write lock from the java.util.concurrency package.
IntSetLinkedListPerNodeLock	Concurrent version of the Linked List implementing a hand-over-hand (fine-grain) locking methodology.
IntSetLinkedListOptimisticPerNodeLock	Variation of the '*IntSetLinkedListPerNodeLock*' that makes use of the *optimistic* strategy.
IntSetLinkedListLazyPerNodeLock	Variation of the '*IntSetLinkedListPerNodeLock*' or the '*IntSetLinkedListOptimisticPerNodeLock*' that makes use of the *lazy* strategy.
IntSetLinkedListLockFree	Concurrent version of the Linked List using a non-blocking strategy.

Do your implementation(s) in the order listed in the table above. We strongly suggest that you do not start the next implementation until you conclude the previous one.

Consider extending the validation

Can you make the validation stronger?

Get some results for the execution time with different workloads and number of threads.

For each version of the *IntSet* benchmark, run some tests with different workloads and number of threads. You are advised to use the same multiprocessor computer that was used for the Home Work 1 (available at IP address 10.170.138.270).

Remember that to reduce the impact of "noise" when measuring the execution time (i.e., the processor(s) got busy doing something else and not executing the program), each test should be executed multiple times.

Evaluation

Your submission will be evaluated considering:

- The quality of the work as described in the submitted report (85%)
- The quality of the submitted code (15%)

Report

You are expected to submit an original written report obeying to the following specifications:

1) Use a4paper size with **all margins at 2cm;**
2) Use font Times New Roman or similar **at size 11pt**;
3) Have at most 6 pages, including:
   - Up to 4 pages with text;
   - Up to 2 pages with charts/graphics;
   - Up to 1 page with Acknowledgments and Bibliography.
4) The report shall follow the following general specification, which should be adapted to better fit your needs and mind set;
   a) Abstract — A very short summary of your problem, your solution and your results.
   b) Introduction — A short section describing what is the context? What is the problem? What is it interesting and challenging in the problem? What are the general approaches/strategies you followed to have the program executing correctly in a multi-threaded setting?
   c) Approach — A longer section describing your solutions. Be simultaneously very concise and very precise on the information you provide in this section. IMPORTANT: By reading this section it must be clear which problems you had to deal with for each of the implemented solutions and how you handled each of them. After reading this section, another programmer should be able to re-implement your solution without looking at your source code.
   d) Validation — A short section describing which techniques did you use to validate your solution. How did you test your program? Why did the tests convince you that your solution was correct and well implemented? What would happen otherwise?
   e) Evaluation — A longer section describing which correctness and performance tests did you make (avoid tables and use charts/plots, unless the precise numeric values are important). You do not need many tests neither many charts/plots. You need to interpret/explain well the results you got (i.e., the behaviour you observed).
   f) Conclusions — A sort sections answering the following questions (or others you may find interesting). Did you got the results you were expecting? How does your validation and evaluation sections confirm that?
   g) Acknowledgments — Did you use some code snippet available on the web? Cite the source. Did get any help from someone other than your group mate? Please give him/her credit for it (the name, number, and what kind of help was given.)
   h) Bibliography — Which references did you use for preparing your work and the report. Bibliographic references must be cited somewhere in the text!
5) It is recommended that you write your report in LATEX (recommended means recommended, not obligatory!)

Submission

If you submit a buggy version (e.g., that does not compile, that stops abnormally with a run-time error, that does not respect the game rules, etc) will be penalized. So…

Before submitting remember to check carefully that your program is working correctly and following all the rules of the game (and of the heuristic used).

How to submit your project?

One of the group members created a project in BitBucket or GitHub by replicating the original repository. This repository must be *shared with both teaching staff members*, whose IDs are:

BitBucket — joaomlourenco, bernymac

GitHub — joaomlourenco, bernymac

Your repository must contain both the source code of your solution and the project report.

What to submit?

The repository must have the following structure (any other folders/files will be ignored):

```
cp-2017-18-gXX-hw2 -+-> README.txt
                   +-> report.pdf
                   +-> src/
```

Where:

- `cp-2017-18-gXX-hw2` — is your repository name (gXX = group number, e.g., "g27").
- `README.txt` — is a text file containing the name and number of both elements in the group.
- `report.pdf` — a PDF with your original written report.
- `src` — a directory (folder) with your source code.

Authors: João Lourenço and Bernardo Ferreira, 2017-18