UiT

THE ARCTIC
UNIVERSITY
OF NORWAY

INF-2202 (Fall 2015)

ASSIGNMENT #1

CONCURRENT B+TREES

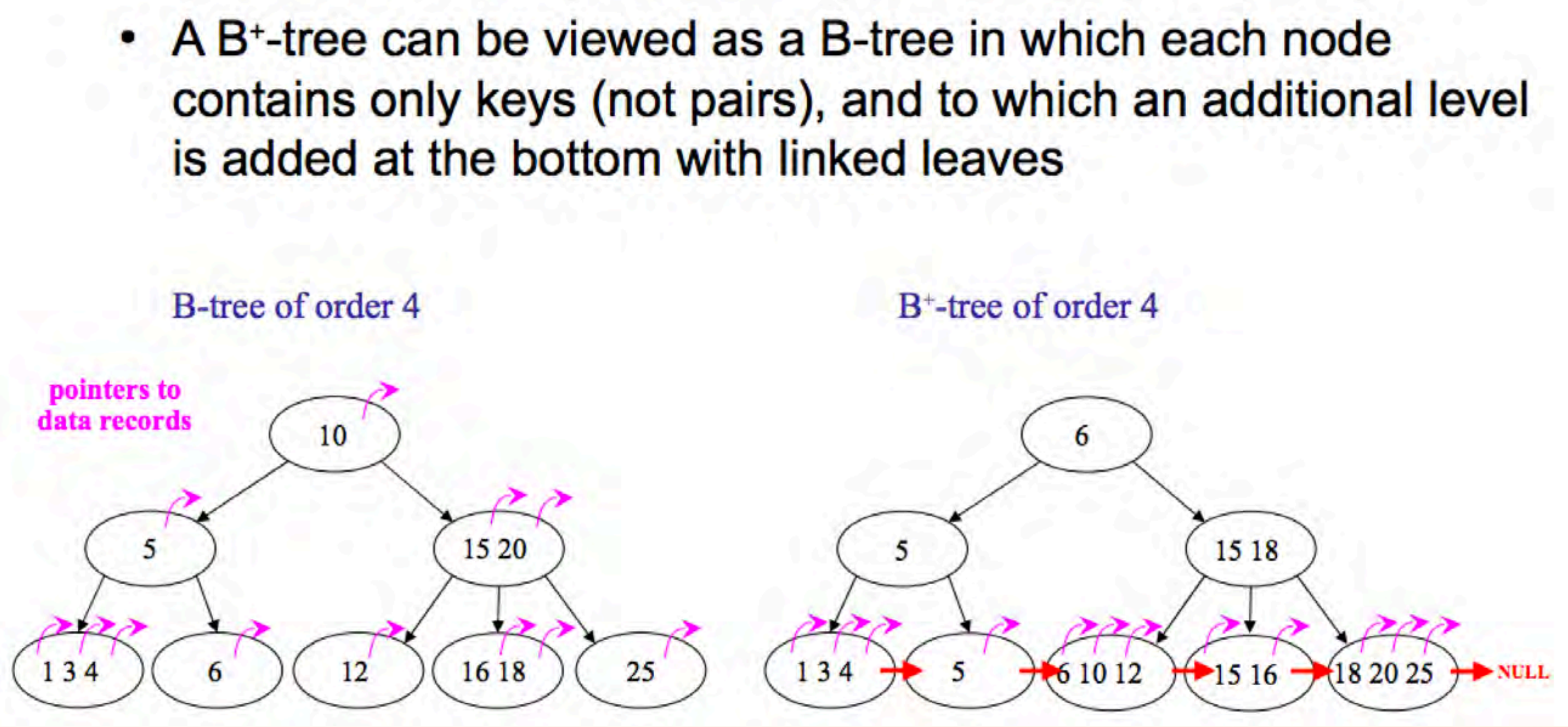
Ibrahim Umar & Lars Ailo Bongo

18.08.2015

Overview

- Your task is to transform the B+tree program which the code given as “precode/bpt.c” into a concurrent B+tree.
- You will find lecture notes related to B+tree inside the “btrees-notes” directory and papers related to concurrent B+tree inside the “concurrent-btrees-papers”.
- **Deadline: 14.09.2015, end of day**

B-tree vs. B+tree



- A B$^+$-tree can be viewed as a B-tree in which each node contains only keys (not pairs), and to which an additional level is added at the bottom with linked leaves

B-tree of order 4

B$^+$-tree of order 4

Source: http://stackoverflow.com/questions/870218/b-trees-b-trees-difference

About the precode (bpt.c)

- This file contains the single-threaded and complete B+tree implementation in C language

- Multi-threaded benchmark and correctness test have been provided inside the code

- Therefore, if your concurrency control implementation is correct, the benchmark and correctness test should run without any errors using multiple threads (e.g., no segmentation faults, all inserted keys are searchable).

- Note that the unmodified program benchmark and correctness test will always succeed if only using a single thread.

Compiling and running

- You can use any modern C (C99 and up) compiler to compile the code
- Run the compiled program with `-h` to display help and the list of accepted running parameters

```
$ gcc -g bpt.c -o bpt -lpthread –lm
$ ./bpt –h
```

- Use the `-t 1` switch to run the sequential and multi-threaded correctness test
- To use more than one thread, you will have to add the `-n <#threads>` parameter

Requirements and limitations

- Modify the B+tree Search, Insert, and Delete operations to support concurrency
- You may use any known techniques or invent one yourself ☺
- Use POSIX Thread API (pthreads.h) (i.e., `pthread_spin_lock`, etc.). Please contact your TA first if using other methods
- Changing the benchmark and test functions is not allowed
- The resulting program must be run using any number of threads
- Give a report on your method for achieving concurrency and also the parallel performance analysis (e.g., speedup, efficiency)
- Analysis should be done using the built-in multi-threads benchmarks using different combination of update ratios (`-u <x>`) and thread numbers (`-n <y>`)

GitHub workflow

1. Fork the assignment repository using GitHub
2. Create a directory in the forked repo using your UIT userID as the name (`<root>/abc123`)
3. Code(s) should be placed under `<root>/abc123/codes` while report is inside `<root>/abc123/report`
4. Work your solution and report using the forked private repo (but please do not make the forked repo public)
5. When done, create a "pull request" to this repo

Grading

- Based on your submitted code and report, a PASS or FAIL grade will be given

- Therefore, be sure to adhere to previously described requirements and limitations!

Disclaimer

- Please do not publicize or share your solution or codes anywhere without our permission

- Since this is an individual assignment, please refrain yourself to copy other students code(s).

- In contrary, group discussions and brainstorming for ideas are strongly encouraged