UiT

NORGES
ARKTISKE
UNIVERSITET

EXAM PAPER

Exam in: INF-2202

Date: Thursday November 26th 2015

Time: 15:00 – 19:00

Place: Åsgårdvegen 9

Approved aids: None

The exam contains 5 pages included this cover page

Contact person: Lars Ailo Bongo

Phone: 92015508

NB! It is not allowed to submit rough paper along with the answer sheets

1) Concurrent hash table (20%)

In this question you will implement a concurrent hash table that resolves collisions using separate chaining (Figure 1).

Multiple threads will access the hash table at the same time. We divide the threads into two classes that you will use to prioritize access to the hash table: (i) *readers* can either check for the presence of a key, or read an entry, and (ii) *writers* can add new entries, modify existing entries, or delete entries.

Your solution should use pseudo code, and you are only allowed to use locks (mutexes) and condition variables.

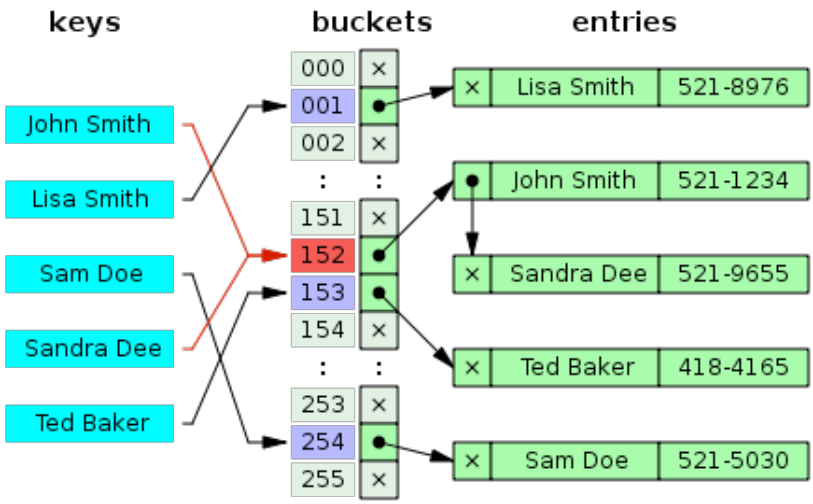

Figure 1: Hash collision resolved by separate chaining. [**Source:** Wikimedia]

Tip: the purpose of this question is to demonstrate your understanding of concurrent programming (and not hash table implementation). So you can, and should, use high-level pseudo-code functions.

a) Simple solution

Specify an interface for the hash table. Then use the interface functions, locks and (or) condition variables to implement a solution *where no two threads shall access the hash table at the same time.*

b) Readers preference

Modify your solution such that *no reader shall be kept waiting if the hash table is currently opened for reading.*

c) Writers preference

Implement a solution such that *no writer, once added to the wait queue, shall be kept waiting longer than absolutely necessary.*

d) No starvation

Implement a solution such that *no thread is allowed to starve*.

2) Sieve of Eratosthenes (15%)

This question is based on an exercise given by Rob Pike in a course at UiT in 2000.

Our goal is a program that prints primes. We observe that the integers 2, 3, 4, ... could be the successive values read from a channel. The first prime is the first integer in the series, 2.

Create a goroutine to print that value, then pass along, on a different channel, the integers that are non-zero modulo 2. That is, the goroutine copies the input stream to the output stream, removing the numbers that are zero modulo 2.

The next prime is the first integer that emerges from the output stream of the first goroutine, 3. Create a goroutine that prints that number and passes the rest along, removing numbers that are zero modulo 3.

The next prime is the first integer that emerges from the output stream of the second goroutine, 5....

In other words, the program is a dynamic chain of processes, each of which filters out values divisible by its prime.

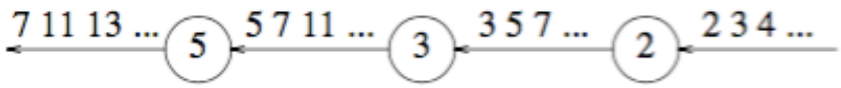

Figure 2: by Rob Pike.

This is the concurrent programming version of the Sieve of Eratosthenes.

a) Go

In Go, what is a *goroutine*? What is a *channel*?

b) Implementation

Implement the Sieve of Eratosthenes in Go.

3) Parallel programming laws (15%)

Gene Amdahl (16.11.1922 – 10.11.2015) proposed a famous law in parallel programming at a conference in 1967.

Figure 3: Gene and his wife, Marian, in front of the main house at Amdahl, Norway.
[Source: wikimedia]

a) Amdahl's law

Briefly describe, or formulate, Amdahl's law.

Use Amdahl's law to predict the speedup of the deduplication engine implemented in the second mandatory assignment. Please state any assumptions you make.

b) Gustafson's law

How does Gustafson's law differ from Amdahl's law?

c) Amdahl's law vs. Gustafson's law

For which of the three mandatory assignment programs (concurrent B+ tree, deduplication engine, PageRank) will the speedup prediction differ most for Gustafson's law compared to Amdahl's law? Briefly describe why.

4) Page Rank (50%)

PageRank is a way of measuring the importance of website pages. It is the algorithm you implemented in the third mandatory assignment.

Figure 4: Cartoon illustrating the basic principle of PageRank. [Source: wikimedia]

a) Parallelization

The parallelization process can be divided into four steps; decomposition, assignment, orchestration, and mapping. Give a brief description of each step.

b) Parallelization goals

Give a short description of the major performance goals for each of the four steps.

c) Spark

Why is Spark well suited to implement PageRank?

d) PageRank decomposition

Describe how you would decompose PageRank.

e) PageRank assignment

Describe how you would do assignment for PageRank. You can assume that you are using Spark.

f) PageRank orchestration and mapping using Spark

Describe, using pseudo code, how you would orchestrate a parallel implementation of PageRank using Spark.

Describe how you would do mapping using Spark (that is, the mapping stage of the parallelization process).

g) Amazon Web Services

Give a high-level description for how to use Amazon Web Services to evaluate the performance of PageRank implemented using Spark.