UiT
NORGES
ARKTISKE
UNIVERSITET

EXAMINATION PAPER

Exam in: inf-2202 Concurrent and System Level Programming

Date: Friday 22.11.2013

Time: Kl 09:00 – 13:00

Place: Åsgårdvegen 9

Approved aids: None

The exam contains 5 pages included this cover page

Contact person: Lars Ailo Bongo

Phone: 92015508

UNIVERSITETET I TROMSØ

1) Sleeping barber (25%)
The sleeping barber problem is a classical inter-process communication and synchronization problem.

The analogy is based upon a hypothetical barber shop with one barber. The barber has one barber chair and a waiting room with a number of chairs in it. When the barber finishes cutting a customer's hair, he dismisses the customer and then goes to the waiting room to see if there are other customers waiting. If there are, he brings one of them back to the chair and cuts his hair. If there are no other customers waiting, he returns to his chair and sleeps in it.

Each customer, when he arrives, looks to see what the barber is doing. If the barber is sleeping, then the customer wakes him up and sits in the chair. If the barber is cutting hair, then the customer goes to the waiting room. If there is a free chair in the waiting room, the customer sits in it and waits his turn. If there is no free chair, then the customer leaves. [1]

The problem can be implemented as follows shown on the next page:

[1] The text is from http://en.wikipedia.org/wiki/Sleeping_barber_problem

```
#
# Sleeping barber
#

numberOfSeats = 5      # Number of seats in waiting room
waitingCustomers  = 0  # Number of waiting customers
someoneInChair = False # True if someone is sitting in barber chair

def barber():
    while true:
        if someoneInChair == False:
            # Spin until customer arrives
            while someoneInChair == False:
                pass # spin
            cutHair()
            someoneInChair = False

       if waitingCustomers > 0:
            waitingCustomers = waitingCustomers – 1
            someoneInChair = True
            cutHair()
            someoneInChair = False


def customer():
    if someoneInChair:
        if waitingCustomers == numberOfSeats:
            # Leave
            return False # Hair could not be cut
        else:
            waitingCustomers = waitingCustomers + 1
            # wait until hair is cut (not implemented)
            return True # Hair was cut
     else:
        someoneInChair = True
        # wait until hair is cut (not implemented)
        return True # Hair was cut
```

1) Sleeping barber continued

a) Assume that the code implemented a wait-until-hair-is-cut function. Describe at least two synchronization errors that may occur if the above code is run on a multi-core processor with a single barber and multiple customers.

b) Implement a solution for the sleeping barber problem using the synchronization mechanisms in pthreads or Python threads. In the solution the customer should wait until his hair is cut.

c) Implement a solution the sleeping barber problem using Go.

2) Spam filter (50%)

You are to implement a spam filter that works as follows:

1. Periodically dump ten thousands of emails to a file.
2. The spam filter read an email from the file, and for each email:
    a. Check if the sender is in a whitelist. If so, no further checks are necessary
    b. Check if the sender is in a blacklist. If so, the mail is marked as spam.
    c. For each word in the subject field, check if the word is in a spam word list. If more than N words are in the spam word list, mark the email as spam
3. Write the non-spam messages to an output file
4. Write the spam messages to a spam file

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

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

c) Describe how you would *decompose* the spam checker.

d) Describe how you would do *assignment* for the spam checker.

e) Give pseudo-code for *orchestration* of a parallel spam checker on a shared memory computer using Go.

f) Describe how you would experimentally evaluate the performance of the spam filter.

g) Based on your experience implementing and evaluating the compression engine in mandatory assignment 2; what do you expect to be the major performance limitations?

3) Grep (25%)

The grep utility searches any given input files, selecting lines that match one or more patterns. By default, a pattern matches an input line if the regular expression (RE) in the pattern matches the input line without its trailing newline.[2]

a) Give pseudo code for grep implemented using MapReduce. You can assume that you have a function for doing pattern matching.

b) What are the advantages of using MapReduce rather than pthreads, Python threads, or Go?

c) What is the expected speedup of a MapReduce implementation over a sequential implementation? Please list all assumptions you make about important factors such as input file size, number of cores, and so on.

[2] From the grep man page