**Group Activity 10; CS 3060**

**Names** of students in your group: Sidney Sanders, Amanda Collert, Quinci Drain, Emily Endlish

Points: 10

**Goal**: To make us familiar with Scala’s support for parallel computation. In particular, we will become familiar with the map-reduce style computation as available in Scala.

**Task 1**: (5 points) Consider the following Scala code, which is supposed to compute the product of the cubes of integers from 1 to 10, involving parallel execution. Fill in the blanks to complete the code. Hint: Refer to examples given on Slide 42-44 on Scala ppt 3.

L1. val list1 = (1 to 10).toList // make a list of 1 to 10

L2. val list2 = list1.par // make a par version of the list

L3. val list3 = list2.map(x => x \* x \* x ) // get list of cubes

L4. println(list3.reduce (\_\*\_) ) // compute and print the prod

map

1\*1\*1

2\*2\*2

3\*3\*3

.. …..

……

.. ….

n\*n\*n

reduce

op

with \*

1

2

3

..

..

n

final

result

map step runs in parallel

**Task 2**: (5 points) On Scala ppt 3 Slide 30 we have *code* to compare *serial* and *parallel* execution time of computing the size of multiple webpages. Also, on Scala ppt 3 Slide 43 we have *code* to compute sum of the size of multiple webpages in parallel fashion. Tweak these code snippets to do the following.

Compare *serial* and *parallel* execution time of computing the total *number of links* present in those webpages. Figure 1 and Figure 2 below show the schematic diagram of the serial and parallel computation.

Hint: basically you need to define a regex for a link, and count the number of matches as we did on Slide 17 in ppt 3.



Figure : Computation with regular collection (e.g., a list). All operations run serially.



Figure : Computation with Parallel collection (e.g., list.par). The map part runs in parallel fashion.