Parallelism and Concurrency

Midterm Solution

Wednesday, April 12, 2017

Exercise 1: Parallel Search (25 points)

```
def find(arr: Array[Int], value: Int): Option[Int] = {

  def findHelper(start: Int, end: Int): Option[Int] = {
    if(end - start <= THRESHOLD){
      var i = start
      while(i < end){
        if(arr(i) == value) return Some(i)
        i += 1
      }
      return None
    }else{
      val mid = (start + end)/2
      val (lRes, rRes) = parallel(findHelper(start, mid), findHelper(mid, end))
      lRes orElse rRes
    }
  }

  findHelper(0, arr.length)
}
```

Exercise 2: Parallel Word Splitting (25 points)

```scala
import scala.collection.parallel.ParSeq

def toWords(chars: ParSeq[Char]): Vector[String] = {

  type WordSeq = (Boolean, Vector[String], Boolean)

  val z: WordSeq = (false, Vector(), false)

  def f(s: WordSeq, char: Char): WordSeq = {
    val (spaceLeft, words, spaceRight) = s

    if (char.isWhitespace) {
      if (words.isEmpty) {
        (true, words, true)
      }
      else {
        (spaceLeft, words, true)
      }
    }
    else {
      if (words.isEmpty) {
        (spaceLeft, Vector("" :+ char), false)
      }
      else {
        if (spaceRight) {
          (spaceLeft, words :+ ("" :+ char), false)
        }
        else {
          (spaceLeft, words.init :+ (words.last :+ char), false)
        }
      }
    }
  }

  def g(s1: WordSeq, s2: WordSeq): WordSeq = {
    val (spaceLeft1, words1, spaceRight1) = s1
    val (spaceLeft2, words2, spaceRight2) = s2

    if (words1.isEmpty) {
      (spaceLeft1 || spaceRight1 || spaceLeft2, words2, spaceRight2)
    }
    else if (words2.isEmpty) {
      (spaceLeft1, words1, spaceRight1 || spaceLeft2 || spaceRight2)
    }
    else if (spaceRight1 || spaceLeft2)
      (spaceLeft1, words1 ++ words2, spaceRight2)
    else {
      (spaceLeft1, (words1.init :+ (words1.last ++ words2.head)) ++ words2.tail, spaceRight2)
    }
```

```
  }

  chars.aggregate(z)(f, g)._2
}
```

Exercise 3: Memory Models (25 points)

Question 1

```
1, 2
0, 1, 3
3
3
```

Question 2

```
0, 1, 2
0, 1, 2, 3
0, 1, 2, 3
3
```

Exercise 4: Lock-Free Banking (25 points)

```
class Account(initialAmount: Long) {
  private val amount = new AtomicLong(initialAmount)

  def getAmount: Long = amount.get

  def transfer(target: Account, n: Long): Unit = {
    if (n <= 0L)
      throw new IllegalArgumentException("n must be positive")

    val currentAmount = amount.get
    if (currentAmount < n)
      throw new IllegalStateException("Not enough money")

    if (currentAmount.compareAndSet(currentAmount, currentAmount - n))
      target.amount.addAndGet(n)
    else
      transfer(target, n)
  }
}
```