Yiqing Huang – SEC 01 (NUID 001525629)

Big Data System Engineering with Scala
Spring 2022
Assignment No. 7

Task
Web Crawler

- **Implement the 3 ‘to be implemented’ parts in *WebCrawler.scala*.**
    1. **WebCrawler.wget.getURLs(ns: Node): Seq[Try[URL]]**
    2. **WebCrawler.wget(u: URL): Future[Seq[URL]] for-comprehension**
    3. **WebCrawler.wget(us: Seq[URL:): Future[Seq[Either[Throwable, Seq[URL]]]]**
- **Implement the 2 ‘to be implemented’ parts in MonadOps.scala**
    1. **MonadOps.mapFuture**
    2. **MonadOps.sequence**
- `[Optional]` **Suggestions on how to improve the web crawler**

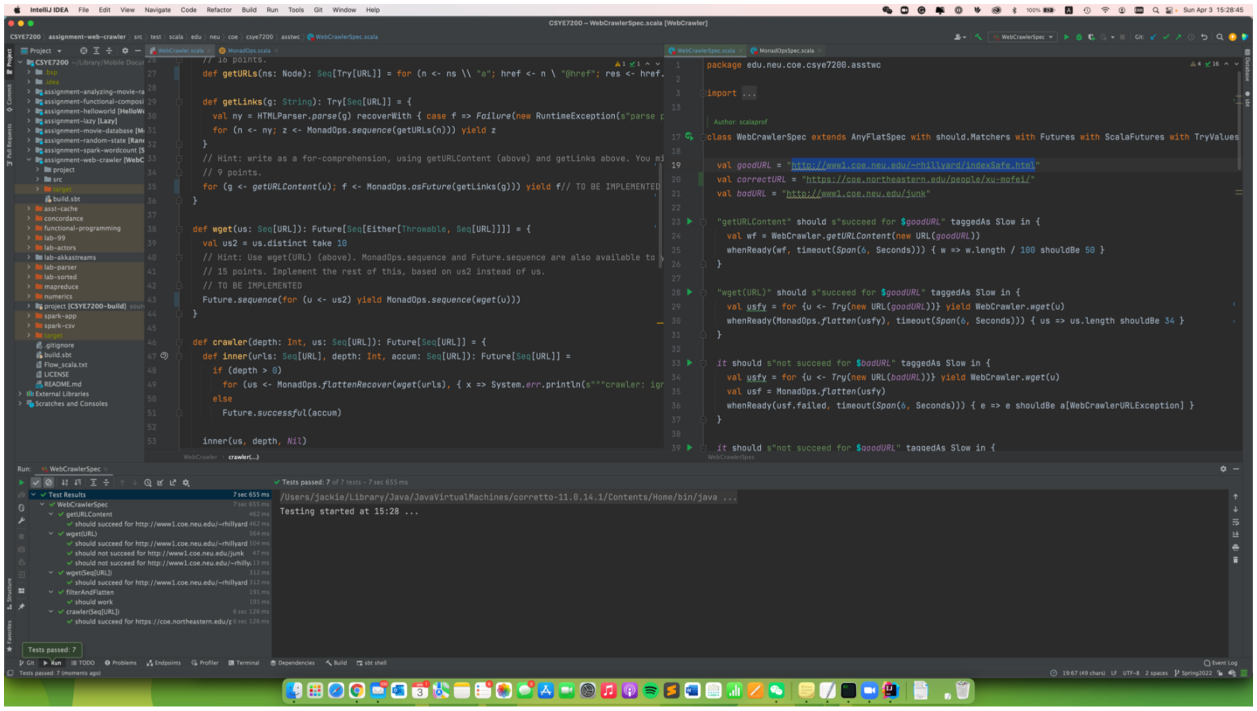
Task
Suggestions

I suggest refactoring the code by embedding an open-source toolkit called *akka* and using *Apache Kafka* as a buffer and load distributor for the crawling jobs. Firstly, *Akka* provides multiple high-performance programming models for concurrency (e.g., actor-based concurrency). Secondly, it provides an asynchronous, non-blocking HTTP client that enables a high degree of parallelism while keeping resource requirements to a minimum. Moreover, thanks to its high-level encapsulation, the whole web-crawling process can be simplified. As a result, our code can be much more readable and easier to develop and maintain.

According to a case study of the improvement of *PayPal*’s web crawler. It shows that by using the *akka* toolkit, the performance has increased tenfold for the reason that *akka* uses a much lower number of threads and achieves an incredible CPU utilization rate of around 90%. (*PayPal scales to billions of daily transactions with Scala and Akka Platform from Lightbend*)

Solution/Unit test (screenshot)

A. Unit Test



```scala
    // 16 points.
    def getURLs(ns: Node): Seq[Try[URL]] = for (n <- ns \\ "a"; href <- n \ "@href"; res <- href.

    def getLinks(g: String): Try[Seq[URL]] = {
      val ny = HTMLParser.parse(g) recoverWith { case f => Failure(new RuntimeException(s"parse p
      for (n <- ny; z <- MonadOps.sequence(getURLs(n))) yield z
    }
    // Hint: write as a for-comprehension, using getURLContent (above) and getLinks above. You mi
    // 9 points.
    for (g <- getURLContent(u); f <- MonadOps.asFuture(getLinks(g))) yield f// TO BE IMPLEMENTED
  }

  def wget(us: Seq[URL]): Future[Seq[Either[Throwable, Seq[URL]]]] = {
    val us2 = us.distinct take 10
    // Hint: Use wget(URL) (above). MonadOps.sequence and Future.sequence are also available to y
    // 15 points. Implement the rest of this, based on us2 instead of us.
    // TO BE IMPLEMENTED
    Future.sequence(for (u <- us2) yield MonadOps.sequence(wget(u)))
  }

  def crawler(depth: Int, us: Seq[URL]): Future[Seq[URL]] = {
    def inner(urls: Seq[URL], depth: Int, accum: Seq[URL]): Future[Seq[URL]] =
      if (depth > 0)
        for (us <- MonadOps.flattenRecover(wget(urls), { x => System.err.println(s"""crawler: igr
      else
        Future.successful(accum)

    inner(us, depth, Nil)
```

```scala
package edu.neu.coe.csye7200.asstwc

import ...

 Author: scalaprof
class WebCrawlerSpec extends AnyFlatSpec with should.Matchers with Futures with ScalaFutures with TryValues

  val goodURL = "http://www1.coe.neu.edu/~rhillyard/indexSafe.html"
  val correctURL = "https://coe.northeastern.edu/people/xu-mofei/"
  val badURL = "http://www1.coe.neu.edu/junk"

  "getURLContent" should s"succeed for $goodURL" taggedAs Slow in {
    val wf = WebCrawler.getURLContent(new URL(goodURL))
    whenReady(wf, timeout(Span(6, Seconds))) { w => w.length / 100 shouldBe 50 }
  }

  "wget(URL)" should s"succeed for $goodURL" taggedAs Slow in {
    val usfy = for {u <- Try(new URL(goodURL))} yield WebCrawler.wget(u)
    whenReady(MonadOps.flatten(usfy), timeout(Span(6, Seconds))) { us => us.length shouldBe 34 }
  }

  it should s"not succeed for $badURL" taggedAs Slow in {
    val usfy = for {u <- Try(new URL(badURL))} yield WebCrawler.wget(u)
    val usf = MonadOps.flatten(usfy)
    whenReady(usf.failed, timeout(Span(6, Seconds))) { e => e shouldBe a[WebCrawlerURLException] }
  }

  it should s"not succeed for $goodURL" taggedAs Slow in {
```

```
✓ Tests passed: 7 of 7 tests – 7 sec 655 ms

✓ Test Results                                                      7 sec 655 ms
  ✓ WebCrawlerSpec                                                  7 sec 655 ms
    ✓ getURLContent                                                 462 ms
      ✓ should succeed for http://www1.coe.neu.edu/~rhillyard       462 ms
    ✓ wget(URL)                                                     564 ms
      ✓ should succeed for http://www1.coe.neu.edu/~rhillyard       504 ms
      ✓ should not succeed for http://www1.coe.neu.edu/junk         47 ms
      ✓ should not succeed for http://www1.coe.neu.edu/~rhilly      13 ms
    ✓ wget(Seq[URL])                                                312 ms
      ✓ should succeed for http://www1.coe.neu.edu/~rhillyard       312 ms
    ✓ filterAndFlatten                                              191 ms
      ✓ should work                                                 191 ms
    ✓ crawler(Seq[URL])                                             6 sec 126 ms
      ✓ should succeed for https://coe.northeastern.edu/p           6 sec 126 ms

/Users/jackie/Library/Java/JavaVirtualMachines/corretto-11.0.14.1/Contents/Home/bin/java ...
Testing started at 15:28 ...

Tests passed: 7
Tests passed: 7 (moments ago)
```

Project Source

https://github.com/eikcaj16/CSYE7200/tree/Spring2022/assignment-web-crawler