ACS Programming assignment 1
Resubmission

Anders Holst, wlc376

2020-12-18

This is my resubmission of programming assignment 1. This hand-in contains only new additions, and thus none of the original hand-in.

Please note: The feedback I received was not perfectly unambiguous as to which parts of the assignment I should resubmit, so I have done my best to infer it.

0 Implementation - resubmission

In my feedback, you noted that there was an error in running my tests non-locally.

I took a look at it and found that I had accidentally broken some of my code while cleaning up before submission.

The problem has been fixed and the tests should now run once again.

---

1 Resubmission: question 2a

- Feedback given: *Think about the benefits of using classes.*

I was given a hint to think about the benefits of using classes - in my answer I had focused on interfaces, but the same principle goes for the use of classes.

---

2 Resubmission: question 2b

- Feedback given: *Think about the benefits of ImmutableBook and ImmutableStockBook.*

By letting book store methods return immutable version of books, we ensure that clients are never manipulating originals.

---

3 Resubmission: questions 3a and 3b

- Feedback given: *The process of whole link could be regarded as a kind of naming service. And do we really need a naming server for naming service? Please give the function of the link.*

I had another look at the architecture and it occurred to me that the entire client/server communication is a naming service, allowing messages to be sent back and forth as though everything was executed locally.

The server and request is simply specified as a URL, and the communication follows send/receive semantics.

---

4 Resubmission: question 5b

- Feedback given: *Think about at-most-once semantics. Please point specifically where the web proxy should be deployed?*

In my original answer, I proposed that proxies be placed between clients and the bookstore server; more specifically, what I meant is that a proxy could be placed between the client HTTP proxy and the HTTP server.

This solution respects at-most-once semantics if the proxy is able to keep track of requests, such that it does not accidentally send the same client request to multiple book store servers, or to the same server multiple times.

---

5 Resubmission: question 6b

- Feedback given: *The line of reasoning is fine, but the answer should provide more details.*

In my original solution, I stated that there is a bottleneck in the fact that a single BookStore-HTTPServer handles all clients, and that scaling the number of clients would mean queues of incoming HTTP requests.

Another big point of consideration is, of course, the problem of resource locking. Unless the number of different books in stock also scale infinitely, then the increased number of clients will mean an increase in collisions in book accesses. These transactions would then be handled sequentially, even as the number of book store servers increase.

---

6 Resubmission: question 7a

- Feedback given: *What if the web proxy is not used to cache data?*

If there is no caching in the web proxies, then there is still the benefit that when one book store server is down, the web proxy can simply send the request to another, active server, if one such exists.

---

7 Resubmission: question 7c

- Feedback given: *What about the RPC and all-or-nothing semantics?*

One big problem with web proxies in this case is the fact that proxy cached resrouces might not be up to date. A proxy's response to an RPC might not be equivalent with what the server itself would have responded, violating the notion of the RPC.

With respect to all-or-nothing semantics: If the web proxy is given responsibility of any sort of request validation, then an outdated cache can be fatal, since it might reject otherwise valid requests on the basis that it is simply not aware of the existence of the book/s requested.
