Write up of ASG4: adding aliases to the HTTP server

Yujia Li
CruzID: yli302

Write up questions:

- **Explain the difference between fully resolving a name (to an httpname) when the name is created and the approach that you're taking for this assignment. Give an example of when it might be useful.**

  Give httpname aliases can let filename more meaningful human. Httpname is a 40 hexadecimal characters which is necessary for server database, and it is meaningless for people. But if we map meaningful aliases to a httpname, it is easier to remember and operate names. For example, when I test my server, I could let '4MBfile' be an alias, and map to an httpname whose data size is 4MB. In this situation, I don't need to type 40 hex char to call this httpname, I could simply call '4MBfile' and this file's name can tell me size of data of this file.

- **Was it easier to modify your existing KV store code from Assignment 3 as compared to getting it to work the first time?**

  It is easier to modify kv store code from asg3 than getting it to work the first time. When write kv store code for name mapping (asg4), I don't need to consider how to use hash function and linear probing in kvs. I move the code from asg3 to asg4 and changed it work for name mapping. After finishing code, kvs for name mapping got work without debugging. Because of that, I think asg4 is the easiest assignment.

- **What did you learn about system design from this class? In particular, describe how each of the basic techniques (abstraction, layering, hierarchy, and modularity) helped you by simplifying your design, making it more efficient, or making it easier to design.**

Thinking in system design helped a lot. During write design doc, I always consider apply abstraction, layering, hierarchy and modularity in my code.
Abstraction: When I designed KVS interfaces, I design the arguments of KVS functions the same as pread and pwrite functions. In this way, I can hide the complexity concurrency problem of pread and pwrite, and make it easier to understand.
Layering and hierarchy: In my design doc, I draw several diagrams to explain the layering and hierarchy to me and other people. Thinking in layering and hierarchy let me keep a clean clue of how to implement server and client step by step. Especially when I first implement kvs in server. If I didn't think in layering and hierarchy, I would mess up the order and spend lots of time to sort out the solution.
Modularity: By separating code into different modula, my program become easier to debug, and save lot of time. Asg4 is a great example in modularity. In asg4, for PUT and GET request, I needed to find a valid httpname first. When I got a valid httpname, I could pass it to kvs file module which had already been done in asg3. After ensuring there is no bug in find httpname, my PUT and GET can work well, because kvs file module was bugfree. I don't need to look around my whole code to find where lead to this bug. Bugs only restricted in their module.

TEST:

All test shell script in test directory pass, include send big file, recv big file, send small file, recv small file, send then recv big file, send tehn recv small file. And all these operations with aliases. A test for infinity loop of alias handler (alias chain). All shell script run client at least 10 times as the same time. So can be consider as multithread.

No known bug.