gtrpc

Caching Scheme Evaluation in a RPC server

How to run

- `thrift --gen cpp server.thrift` to generate the skeleton code (don't need to)
- `export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/lib` for linking
- run `make` to compile the clients and server
- server executable: `bin/ProxyRPC_server <cache type> <cache size>`, type 0 for EmptyCache, 1 for RandCache, 2 for FIFOCache, 3 for LRUCache and cache size is in KB
- client executable: `bin/client <server-ip>` - executes an example run to test the sever. It accesses 100 different urls in sequence twice
- client executable: `bin/get <server-ip> <url to fetch>` - execute a single run of RPC call and fetches the given url through proxy server at the provided ip

Files

- `gen-cpp/get.cpp` - client example, executes an example run to test the sever. It accesses 100 different urls in sequence twice
- `gen-cpp/client.cpp` - client example, execute a single run of RPC call and fetches the given url through proxy server at the provided ip
- `gen-cpp/ProxyRPC_server.cpp` - server example
- `gen-cpp/cache.cpp` - implements all the cache, read `doc/report.pdf` for more details
- `gen-cpp/curl.h` & ``gen-cpp/curl.cpp` - uses curl class to fetch document from a given url
- `python script/run.py` - used to run experiments on localhost, the geerated file are present in `data/` folder containg the response time. The hit rate is visible on terminal
- use `sudo fuser -k 9090/tcp` if port is already being used

Resources

- Data: https://docs.google.com/spreadsheets/d/1U7titiaqbHlA-lGv7PLrT_NCyfi-8m4OO9-cHXbfJLk
- code: https://github.com/mangalaman93/gtrpc
- report: doc/readme.pdf, doc/report.pdf