15-745 Project Milestone Report

Adarsh Sreedhar (adarshsr), Bhakti Chaudhari (bchaudha)

Major Changes

We do not have any major changes in our implementation, and believe that we should be able to successfully generate a compiler pass to replace RPC calls with Cache call, based on a simple expiry logic.

What you have accomplished so far

We have implemented a pass in Soot that rightly identifies the HTTP REST API in the generated IR (seen as interfaceinvoke type) for the Microservice, and replaces that with the Controller method call (seen as virtualinvoke type).

We have also written a simple Cache of a generic (K,V) type, and have come up with a logic of using the Comparator class and checking whether the required data exists in the cache, and if so retrieve that data, and if not, make the RPC call and store in the cache, all in the IR through Soot.

Meeting your Milestone

Our Milestone was to have a pass that successfully replaces RPC calls with a working cache and be able to test the accesses. While we mostly have all code in place required to achieve this, we are short of integrating our pass that inserts the wrapper with the generated Cache class.

With respect to the timeline, we are running one week behind schedule i.e. we have completed the expectations set for Week 3 (Developing a pass to insert instructions that replace the chosen RPC with cache calls), however Week 4 was integrating the compiler pass and adding logic of cache coherency, which we are currently progressing on.

Surprises

Framework:

There is very little documentation on Soot as well as questions on Stackoverflow or resources and examples. This is in contrast with LLVM which is more popular and therefore finding the right set of the calls to achieve necessary transformations in the IR was relatively easy. We are mostly going through the Source code itself to understand how to make any changes, and also rely on tools such as Intellisense for fast lookup of the methods.

Revised Schedule (Divided in Half-Weeks)

Week	Bhakti	Adarsh
Apr 14-17	Integrate the Cache logic and successfully add a wrapper in the Microservice IR	Check with Jerry on how to run the benchmarks to test out the microservice
Apr 18-22	Identify if the expiry-based cache logic is best or any other logic would suit this application	Collect and Analyse the results and the possible improvements in performance obtained
Apr 23-26	Attempt to generalize the pass to any Microservice within Train Ticket Application	Research on the effort required on cache coherency and possibly add placeholder code and expand based on available time.

Apr 27	Formulate Results and Report Writing	Preparation for Poster Presentations

Resources Needed

Since we are going to apply this pass on the Train Ticket Application, from our conversations with other members working on it, we believe that we should have the software, benchmarks and simulators in place. We were told the application takes about ~20GB of Memory, and so have been granted access on the Narwhal cluster to setup the application and run validations.