HowToUseConcurrencyAnalysisTools

Sound and dynamic concurrency analysis of Hadoop

The purpose of this page is to document how developers can obtain and leverage these tools in Hadoop.

Recently, SureLogic a company offering concurrency analysis tools for Java applications, has donated a license for the following products to the ASF.

- JSure: static analysis of concurrent applications
- Flashlight: dynamic concurrency analysis

It happened as a result of Yahoo! Hadoop team engaging with SureLogic to evaluate their products for Hadoop development. HDFS, MapReduce, and ZooKeeper teams were assessing the tools for three day back in October, 2009. Overall, the results were quite promising and produced some findings already.

Concurrency analysis tools

- The idea behind !JSure is simple yet highly effective: parts of the application source code are getting tagged with a special annotations. The parts in question identified by SureLogic analyzer as possibly problematic or even buggy. So having these tags (or annotations) in place will help the analyzer skip annotated spots and will only raise the warning about those left untouched. Such an approach might seems to be to troublesome because it literally requires a developer attention to the every bit of concurrent code which has even slight odor. However, it turns out to be a double-fold benefit: first, one pays some close attention and might even rethink particulars of their application's design and/or implementation; second, the annotations works as a form of design document virtually attached to the source code.
- Flashlight might be executed in conjunction with JSure or separately. It helps to find concurrency issues in the running code. Clearly, if all concurrency problems were to be found by static checks we won't have any concurrency bugs: it simply won't happen, because we would've been using static analyzers early in the game and detect the issues right on the spot. Unfortunately, it's worst than that. And that's why it is very important to know how your multi-threaded application behaves in the runtime. Highly concurrent ZooKeeper's development team found this tool to be quite useful.

How to get and install the tools

First of all you need to download and install the tools. Here's the instruction on how to do this. The good news - it shouldn't take more than 5 minutes to get you going. The bad news for non-Eclipse users - you need to get one first. Please refer to the page to find out how to run the tools.

All SureLogic tools have built-in feedback submission feature, so you might consider to share your comments/suggestions with the development team. If you've came across an issue with any of SureLogic tools you might want to file a ticket and monitor its progress. SureLogic has their own issue tracking system which is available here. Also, the discussion group is available.

Now, you'll need a license. Apache committers can download it from here. All you need is your Apache SVN password. Apache contributors, who don't posses SVN password (i.e. non-committers) need to email the request for a copy of the license to 'cos (at) apache (dot) org' or Tim Halloran from SureLogic and the license will be send to you. This slightly cumbersome process will be streamlined in the future.

Hadoop's projects will eventually automatically pull down annotations jar (aka "Promises") at the build time thus the source code with annotations should be compiled just fine. The patches like this are ready but will be applied later.

If you'd like to add SureLogic analysis to your Apache's project, you will need to configure your build to automatically download and use annotation jar file to make your compiler aware about these. It is simple enough to do thanks to Maven. You can take a look to to use its patch for your project as soon as it's committed.

Some examples of annotations and how to use them

HDFS, MapReduce, and ZooKeeper already has some SureLogic annotations in their source code. Here's some examples from HDFS.

First, you need to declare abstract annotation Region and declare all necessary RegionLocks:

```
@Regions({@Region("HeartbeatState"),
        @Region("DatanodeState")})
@RegionLocks({
        @RegionLock("HeartbeatLock is heartbeats protects HeartbeatState"),
        @RegionLock("DatanodeLock is datanodeMap protects DatanodeState")
})
```

Then you can start adding the specific lock annotations:

```
@RequiresLock("DatanodeLock")
private String newStorageID() {
  String newID = null;
  while(newID == null) {
    newID = "DS" + Integer.toString(r.nextInt());
    if (datanodeMap.get(newID) != null)
      newID = null;
  }
  return newID;
}
```

the annotation `@RequireLock` is telling the analyzer that the developer is aware about potential issue and he or she is pretty sure that the caller of the method holds named `DatanodeLock`. This also tells a uninitiated contributor about the pre-requisites which have to be satisfied before the method can be safely called.

Sometimes different methods might have different assumptions of invocation context. Thus one can use as many annotations as needed. E.g. further in the same class

```
@RequiresLock("HeartbeatLock")
private void updateStats(DatanodeDescriptor node, boolean isAdded) {
...
```

For more information about how to annotate your code check annotations JavaDoc

How it affects Hadoop development/QA process

SureLogic annotations are used in the source code of core sub-projects at the moment. It is not mandatory for contrib projects this it's up to contributors and committers to decide if they want to use concurrency analysis tools for their components.

SureLogic analysis is going to be included to the test-patch process. This said new patches are required not to raise SureLogic warnings level (similar to the requirements about FindBugs or javac).

Annotations retention policy

Annotations are part of the source code and needed for analysis purpose only. Thus, the promises will be needed only during compile time and for development. A production code won't include this jar file thus there won't be any impact on the final project's bits.

JIRA filing guidelines

If you have found and issue with your application using one of the SureLogic tools above please don't forget to add tag 'surelogic' to the JIRA. It will help to trace concurrency related issues in the future.