FP Complete

Hedera and Hedera ledger software

AUDIT REPORT

Customer Name: Hedera Hashgraph

Version: 1.0

Submitted to: Mehernosh Mody

Table of Contents

Prologue

This report is designated for **external distribution**, in accordance with the disclaimer below.

Disclaimer

This report is authorized for external distribution. This report is presented without warranty or guarantee of any type.

This report touches aspects of both the code itself and the architecture. Information for the architecture is gleaned from the whitepaper, website, and discussions with Swirlds members.

THIS REPORT IS UNCORRECTED AND INCOMPLETE, IS BASED ON INCOMPLETE INFORMATION, AND IS PRESENTED WITHOUT WARRANTY OR GUARANTY OF ANY TYPE.

This report is a copy of a work in progress. It lists the most salient concerns that have so far become apparent to FP Complete after a partial inspection of the engineering work. The inspection is ongoing, so further concerns are likely to arise. Corrections, such as the cancellation of incorrectly reported issues, may also arise. Therefore, FP Complete advises against making any business decision or other decision based on this report.

FP COMPLETE DOES NOT RECOMMEND FOR OR AGAINST THE USE OF ANY WORK OR SUPPLIER REFERENCED IN THIS REPORT.

This report focuses on the technical implementation as provided by the project's implementors, based on information provided by them, and is not meant to assess the concept, mathematical validity, or business validity of the project. This report does not make any assessment of the implementation or the project regarding financial viability, nor suitability for any purpose.

While this assessment when complete might be described as an "audit," no official standard exists for an audit of this nature. The word "audit" does not imply compliance with an accounting standard or other standard and is used informally here. FP Complete has not been given access to nor reviewed all aspects of the project and the engineering decision process underlying all the work. This report likely contains errors due to incomplete information as well as simple misunderstanding. This report may include references to problems that do not in fact exist. Meanwhile, the work referenced may or may not contain undetected or unreported problems. FP Complete has not had independent and unfettered access to all the relevant materials. Nor has a "whistleblower" or other process been provided such that any known problem could be reported and included herein.

Some technical decisions in the engineering work were made due to historic reasons, time constraints, budget constraints, or other constraints. Therefore, the presence of a concern or "flag" in this report does not imply improper conduct or lack of skill by the implementer or manager or any party.

NO ATTEMPT IS MADE OR IMPLIED TO JUDGE ANY PERSON, TEAM, COMPANY, OR OTHER PARTY.

Source Material

For this report, the FP Complete team has reviewed the Hedera and Hedera ledger software code bases. Both code bases were audited initially with further differential reviews upon new revisions of the code. The source materials consist of:

- The whitepaper, version 2.0, available at https://www.hedera.com/hh-whitepaper-v2.0-17Sep19.pdf
- The swirlds-open-review repository at https://github.com/hashgraph/swirlds-open-review up to the commit SHA c31f2d1c53a0985fb5cd8475b61bb9cd8a071b25. Explicitly excluded from this repository is the directory sdk/.
- The hedera-services repository at https://github.com/hashgraph/hedera-services up to commit SHA e4f825ce23d5e4429192c8bf218ea220064d548e, which resolves to the git tag v0.7.0-alpha1.
- Multiple versions of an architecture diagram provided to the audit team.

Furthermore, the audit explicitly excluded:

- Proofs in the technical whitepaper
- DevOps, infrastructure, and network architecture
- Technical leadership
- Applicability for business use-cases
- Hiring

Considerations

- The system currently assumes a fixed number of nodes (N). We have reviewed the code, using this assumption. We did not review the code on the basis that this assumption would soon change.
- We have assumed that blob storage will only store small blobs.
- After fixes, many previously identified issues have been resolved. For brevity, those issues are not included in this report.

Legend

This report classes findings into three categories:

- Yellow flag Potential problem without a clear or immediate exploit
- Red flag Confirmed issue which should be addressed immediately
- Green flag Aspect of the project demonstrating best practices

This report does not include previously resolved findings. Instead, we list unresolved and partially resolved flags that comment about production code. Furthermore, the past audit phases included test code and DevOps aspects, which are both excluded from both the public repositories listed above and this report.

Executive Summary

This report contains 0 red flag issues, 7 yellow flag issues, and 0 green flag issues. These issues remain partially resolved or unresolved at the time of writing.

Unresolved and Partially Resolved Findings

This section lists all partially resolved and unresolved findings.

Yellow flag: Task Queue interactions

Reproducible in commit: c31f2d1c53a0985fb5cd8475b61bb9cd8a071b25

Impact

Ungracefully handled failures can lead to degraded behavior when interacting with the Task Queue.

Situation

Part 1

Inside saveSignedStateToDisk:

```
boolean accepted = taskQueue.offer(new FileManagerTask(
        FileManagerOperation.WRITE,
        signedState,
        signedState.getLastRoundReceived(),
        snapshotTask,
        taskDesc));
```

1. The expression taskQueue.offer() returns a boolean indicating whether it can write to the queue or whether the queue is full.
2. If the queue is full, it doesn’t retry in this function, but logs the event.
3. It returns the boolean (false) to the caller to indicate that the queue is full.

If the task failed, the caller of this function won’t be made aware of it or be able to react to it.

Part 2

Meanwhile, in run, we have:

```
    while (true) {
        try {
            FileManagerTask task = taskQueue.take();
            ... run the task ...
        } catch (Exception e) {
            ... logging here ...
        }
    }
```

If running the task fails, such as writeSignedStateToDisk, an exception is caught, logged, but not reported back anywhere. Then we simply continue in the loop processing more tasks.

This seems like it could create an inconsistent state if any other code depends on that write succeeding.

Recommendation

Some possible approaches to these issues:

- If justified, include a written explanation for ignoring failures (e.g., explaining why losing these state updates is acceptable).
- Implement retrying logic, failing after N attempts.
- If writing to disk fails, it’s likely that there is a system failure occurring, or perhaps a permissions issue. In all cases, the software probably should not continue and instead should retreat up the call stack, performing any necessary cleanup.

Exception handling in asynchronous and concurrent programs is one of the more difficult scenarios to debug when errors occur. We hope that writing defensively to anticipate such issues can save substantial time later.

Yellow flag: Deprecated code

Reproducible in commit: c31f2d1c53a0985fb5cd8475b61bb9cd8a071b25

Impact

No immediate impact. This audit note discusses a code maintainability improvement.

Situation

The project contains several classes and methods marked as @Deprecated:

- the class com.swirlds.platform.Crypto
- the class com.swirlds.platform.Hash
- many methods of the class com.swirlds.platform.Utilities
- and others

According to the Javadoc of @Deprecated it has the following meaning:

> A program element annotated @Deprecated is one that programmers are discouraged from using, typically because it is dangerous, or because a better alternative exists.

As a result, Maven produces many warnings when building the project. The attached text file deprecations.txt lists some of these warnings.

Problem

Extensive use of @Deprecated reduces the visibility of related warnings and contradicts the idea of marking definitions as deprecated. Furthermore, having many ignored deprecation warnings could lead to other important warnings from the compiler being ignored.

Suggestions

We advise making proper use of the @Deprecated annotation: either add it less frequently or/and migrate the code currently using deprecated definitions to use better alternatives.

Resolution

The Swirlds team communicated:

- The com.swirlds.platform.Crypto and com.swirlds.platform.Hash, which involve deprecated code, are interrelated and will be addressed in future refactors.
- Due to the recent integration of the new Merkle reconnect logic that is waiting for deployment, the deprecated copyFrom methods will be removed in the future.

Yellow flag: Collection of minor code improvements

Reproducible in commit: c31f2d1c53a0985fb5cd8475b61bb9cd8a071b25

Impact

No direct impact. We make several suggestions to improve code examples with no impact on the code behavior.

Issue

Our suggestions for minor improvements are:

1. The FCSerializer defines markers that are written during serialization and used to distinguish between data areas while deserializing:

```
/**
 * Start delimiter for leaf keys
 */
private static final int KEY_S = 1_801_812_339; // 'k', 'e', 'y', 's'

/**
 * End delimiter for leaf keys
 */
private static final int KEY_E = 1_801_812_325; // 'k', 'e', 'y', 's'

/**
 * Start delimiter for leaf values
 */
private static final int VALUE_S = 1_986_096_243; // 'v', 'a', 'l', 's'

/**
 * End delimiter for leaf values
 */
private static final int VALUE_E = 1_986_096_229; // 'v', 'a', 'l', 's'
```

   Comments look to be incorrect and could be interpreted as e.g. KEY_S and KEY_E have the same value (corresponding to 'k', 'e', 'y', 's'). We suggest fixing the comments.

2. This is a snippet from Network.java

```
static String[] getOwnAddresses2() {
    if (ownAddresses == null)
        try {
            ownAddresses = computeOwnAddresses();
        } catch (SocketException e) {
            log.error(LogMarkers.LOGM_EXCEPTION, "", e);
        }
    return addresses;
}
```

Here, computeOwnAddress() contains the side effect of updating the class variable addresses. This side effect should be avoided or made explicit.

3. addDeserializedChildren() can be named set* instead of add* because it sets children in the range [0,children.size()-1], which makes it a set or replace operation instead of add.

4. In the method round a call to parentRound(x) is not necessary as it practically equals to rsp or rop (ifs before that line exclude cases when they are not equal).

5. This function in SyncConnection.java

```
static boolean connection(SyncConnection syncConnection) {
    return syncConnection != null && syncConnection.connected();
}
```

could be renamed to isConnected or hasConnection.

6. The SyncManager exposes some functions that are called when something needs to be reported to the SyncManager. We suggest using the function naming convention report* consistently. Currently, there is:

```
/**
 * Notifies the sync manager that there was a succesful sync
 */
void successfulSync() {
  ...
}

/**
 * Notify the sync manager that a node has reported that they don't have events we need. This mean
s we have probably
 * fallen behind and will need to reconnect
 *
 * @param id
 *      the id of the node who says we have fallen behind
 */
synchronized void reportFallenBehind(NodeId id) {
  ...
}
```

Using reportSuccessfulSync() as a name for the first function aligns the function names better.

7. The Marshal.java class does not seem required. Settings.java can be used directly instead.

8. The empty abstract class SubSetting.java can be removed or explained in comments.

9. In SwirldsAppLoader.java, a caught Exception can be replaced with the more specific MalformedURLException.

10. The AbstractHashgraph is an abstract class that Hashgraph extends from. However, this abstract class has only abstract methods and no member variables so it can be reasoned about as an interface instead of as an abstract class here.

11. The Constants class has a few constants defined. One of them is::

    ```
    public static final long MS_TO_NS = 1_000_000;
    ```

    This is used by TimestampStreamFileWriter and JVMPauseDetectorThread. However, in the codebase there are many more occurrences of ms-to-ns conversions where a hardcoded 1_000_000 is used instead of this constant. The same situation is seen for the constant SEC_TO_NS.

12. There's a contradiction between docs and implementation in one of the FCMap constructors. The docs state that an immutable copy is created, however, the new copy is set to be mutable and FCMap passed as a parameter is set to be immutable.

13. The method SyncUtils.writeUnknownEvents() lacks javadoc comments for two parameters.

Yellow flag: Outdated Java version

Reproducible in commit: c31f2d1c53a0985fb5cd8475b61bb9cd8a071b25

Impact

Using an unsupported Java version could result in missing important patches and fixes. This could compromise the Swirlds platform.

Situation

The project's root `pom.xml` contains the following:

```
<maven.compiler.source>12</maven.compiler.source>
<maven.compiler.target>12</maven.compiler.target>
```

The current code base uses Java 12 for the source and target versions. According to the page Oracle Java SE Support the current non-LTS version of Java is 14 (2 versions higher than the version 12 currently used).

Support for JDK version 12 ended in September 2019. Newer versions have been released since - with non-LTS versions released every six months.

Suggestions

If the Swirlds platform team has elected to use a non-LTS Java version, then we advise adding a workflow which periodically ensures timely migrations to newer non-LTS Java versions. Otherwise, we recommend standardizing on LTS Java versions.

Resolution

The Swirlds team added dual builds in their CI workflows, which builds the projects in JDK 12 and 14. Generally, this approach allows faster migrations to more recent Java versions in the future.

Yellow flag: Replace custom code with libraries

Reproducible in commit: c31f2d1c53a0985fb5cd8475b61bb9cd8a071b25

Impact

Increased risk to bugs in custom code, which can be replaced with battle-tested libraries.

Issue

There are a few places in the code-base where algorithms are implemented by custom logic instead of using libraries:

1. This includes two places where low-level code is used to parse .txt files:

    a. Parsing the settings.txt file.

    b. Parsing the config.txt file.

    In both examples, a library can replace the low-level code. We recommend using a widely used descriptive language, e.g. YAML, for setting and config files, which can be parsed by corresponding libraries, e.g. snakeYaml, eoYaml, or yamlBeans. Alternatively, CSV format and a CSV-parsing library could be considered. This provides the additional benefit that the parsed files can be easily validated too. For example, currently the Settings.java code only contains one simple validation in the code.

2. Parsing command line arguments is currently performed using custom low-level code, which trims and lowercases a string for example. A CLI library, e.g. picocli, can be used to increase the maintainability of the code, while reducing error-proneness.

Yellow flag: Shellcheck

Reproducible in commits:

- fccbbe7ed911628d73ff8ab07981246fac5639cf
- 1c46f8d30723a833d59b8bc2df1d069a1281ae60

Impact

No direct impact. Numerous shell scripts do not follow best practices, which may cause incorrect behaviour.

Situation

As part of the audit, the tool shellcheck was run to spot shortcomings in the shell scripts of the services-hedera and swirlds-open-review repositories. shellcheck provides a code for each finding of the form SCXXXX, which we are using in this issue as well. This issue outlines found issues with error-severity.

Issue

The following analysis was generated with the tool shellcheck and its invocation: shellcheck -S error $(find . -name '*.sh'). For completeness, the findings include occurrences of code that is out of scope, e.g. the /sdk directory.

hedera-services:

11 issues with error-severity were found, which fall into five distinct error codes:

```
SC1128: The shebang must be on the first line. Delete blanks and move comments.
SC2068: Double quote array expansions to avoid re-splitting elements.
SC2096: On most OS, shebangs can only specify a single parameter.
SC2145: Argument mixes string and array. Use * or separate argument.
SC2148: Tips depend on target shell and yours is unknown. Add a shebang or a 'shell' directive.
SC2199: Arrays implicitly concatenate in [[ ]]. Use a loop (or explicit * instead of @).
```

The full shellcheck output can be found in the attached asset shellcheck-services.txt.

swirlds-open-review:

In the incremental audit, only one issue with error-severity was found in commit 1c46f8d30723a833d59b8bc2df1d069a1281ae60:

```
In ./sdk/data/keys/generate.sh line 5:
if [[ -z "$@" ]]; then
         ^--^ SC2199: Arrays implicitly concatenate in [[ ]]. Use a loop (or explicit * instead of @).

For more information:
  https://www.shellcheck.net/wiki/SC2199 -- Arrays implicitly concatenate in ...
```

Recommendation

Since the above reports are already filtered by error-severity, we suggest fixing all of these. Furthermore, our recommendation is to integrate shellchecks into CI workflows so that dangerous scripting practices can be spotted and addressed consistently (and as early as possible).

Resolution

The reported finding in swirlds-open-review has been fixed in commit 2541f63b026bfcab523ef53b117404793b1102af.

Yellow flag: Thread Safety issues

Reproducible in commit: c31f2d1c53a0985fb5cd8475b61bb9cd8a071b25

Impact

Various thread safety issues could lead to undetermined correctness issues.

Problem

The static analysis tool Infer reports numerous thread safety warnings which are attached in the asset infer-thread-safety-warnings.txt.

Suggestions

The above referenced issues should be fixed. We recommend running the Infer tool as part of the CI so that these classes of error are caught sooner.

Resolution

The Infer tool has been integrated in internal CI workflows to capture the issues earlier and fix them. The reported warnings remain unresolved.