CodeReviewChecklist

Code Review Checklist

Here is a list of things to check during code reviews. Once the review is complete (which means all suggestions from previous reviews have been satisfactorily incorporated and you believe the patch is ready to commit) then please set the **Reviewed** flag on the issue's Jira. **Please do NOT change this list without first discussing the change on the core-dev@ mailing list.**

Coding Style

new code:

- follows Sun's code conventions except indentation is 2 spaces, not 4

changes to existing code:

- maintains existing style

Documentation

internal javadoc:

- accurate, sufficient for future maintainability

public javadoc:

- accurate, sufficient for developers to code against
- follows standard javadoc conventions
- loggers and logging levels covered if they do not follow our conventions (see below)
- system properties, configuration options, and resources covered
- illegal arguments are properly documented as appropriate
- package and overview javadoc are updated as appropriate

Coding

- implementation matches what the documentation says
- logger name is effectively the result of `Class.getName()`
- class & member access - as restricted as it can be (subject to testing requirements)
- appropriate `NullPointerException` and `IllegalArgumentException` argument checks
- asserts - verify they should always be true
- look for accidental propagation of exceptions
- look for unanticipated runtime exceptions
- try-finally used as necessary to restore consistent state
- logging levels conform to Log4j levels
- possible deadlocks - look for inconsistent locking order
- race conditions - look for missing or inadequate synchronization
- consistent synchronization - always locking the same object(s)
- look for synchronization or documentation saying there's no synchronization
- look for possible performance problems
- look at boundary conditions for problems
- configuration entries are retrieved/set via setter/getter methods
- implementation details do NOT leak into interfaces
- variables and arguments should be interfaces where possible
- if `equals` is overridden then `hashCode` is overridden (and vice versa)
- objects are checked (`instanceof`) for appropriate type before casting (use generics if possible)
- public API changes have been publically discussed
- use of static member variables should be used with caution especially in Map/reduce tasks due to the JVM reuse feature

Tests

- **unit tests exist for bug fixes and new features, or a rationale is given in Jira for why there is no test**
- unit tests do not write any temporary files to `/tmp` (instead, the tests should write to the location specified by the `test.build.data` system property)
- `org.apache.hadoop.dfs.MiniDFSCluster`, `org.apache.hadoop.mapred.MiniMRCluster` and `org.apache.hadoop.yarn.server.MiniYARNCluster` are used to start servers as needed (servers are not directly instantiated)

Jira

- the **Incompatible change** flag on the issue's Jira is set appropriately for this patch
- for incompatible changes, major features/improvements, and other release notable issues, the **Release Note** field has a sufficient comment