Item 162
git_comments:

1. MAPREDUCE-6565: need to set configuration for SecurityUtil.
2. MAPREDUCE-6565: need to set configuration for SecurityUtil.

git_commits:

1. **summary:** MAPREDUCE-6565. Configuration to use host name in delegation token service is not read from job.xml during MapReduce job execution. Contributed by Li Lu.
   message: MAPREDUCE-6565. Configuration to use host name in delegation token service is not read from job.xml during MapReduce job execution. Contributed by Li Lu. (cherry picked from commit 8f6e14399a3e77e1bdcc5034f7601e9f62163dea) (cherry picked from commit 376a2439db9cd55ed1f2965cd876b811109990ba) (cherry picked from commit 05a570a1981597ab8d2615ed4752e63f0a21b32b)

github_issues:

github_issues_comments:

github_pulls:

github_pulls_comments:

github_pulls_reviews:

jira_issues:

1. **summary:** Configuration to use host name in delegation token service is not read from job.xml during MapReduce job execution.
   description: By default, the service field of a delegation token is populated based on server IP address. Setting {{hadoop.security.token.service.use_ip}} to {{false}} changes this behavior to use host name instead of IP address. However, this configuration property is not read from job.xml. Instead, it's read from a separate {{Configuration}} instance created during static initialization of {{SecurityUtil}}. This does not work correctly with MapReduce jobs if the framework is distributed by setting {{mapreduce.application.framework.path}} and the {{mapreduce.application.classpath}} is isolated to avoid reading core-site.xml from the cluster nodes. MapReduce tasks will fail to authenticate to HDFS, because they'll try to find a delegation token based on the NameNode IP address, even though at job submission time the tokens were generated using the host name.

jira_issues_comments:

1. Looks like no patch is available here. I propose to handle it in the DFSUtil.createKeyProvider method. // get useIp flag for KMS boolean useIp = conf.getBoolean(CommonConfigurationKeys.HADOOP_SECURITY_TOKEN_SERVICE_USE_IP, CommonConfigurationKeys.HADOOP_SECURITY_TOKEN_SERVICE_USE_IP_DEFAULT); LOG.debug("set "+CommonConfigurationKeys.HADOOP_SECURITY_TOKEN_SERVICE_USE_IP +" value from config." +useIp); SecurityUtil.setTokenServiceUseIp(useIp);
2. need more thought around this.. just realized that setting this flag caused hdfs delegation tokens cannot be looked up by ipc client.. the basic issue is that delegation tokens are populated by each individual client and they do not have a conform protocol to follow to use ip or host name. i will post more findings and suggestions later.
3. a quick place to fix the map reduce issue is to add the following line into the YarnClient.java LOG.info("YARN CHILD CHECK SECURITY SETTING USE_IP:"+job.get(CommonConfigurationKeys.HADOOP_SECURITY_TOKEN_SERVICE_USE_IP)); // get useIp flag for KMS boolean useIp = job.getBoolean(CommonConfigurationKeys.HADOOP_SECURITY_TOKEN_SERVICE_USE_IP, CommonConfigurationKeys.HADOOP_SECURITY_TOKEN_SERVICE_USE_IP_DEFAULT); LOG.debug("set securityutil token service use ip value from config." +useIp); SecurityUtil.setTokenServiceUseIp(useIp);

4. further thoughts on this fix.. the multi-home network setup for hadoop is getting some attention in industry. The security token design using toke + service name does not have updates to accommodate the complex network setup. HA This quick fix just get by with executing mapreduce jobs. However, I would suggest to create a new request to address the multi-home network and token handling in a more organized effort. Security package has very succinct log and it took me quite while to track down the issue. Properties under hadoop.security should be handled in a single code base to interact with various components. Credential token services such as retrieve/clone are handled by different packages in multiple components. Current code is really difficult to understand and manage.
5. I think MR over distributed cache shouldn't load configurations that included in MR tar ball. CC [~jlowe].
6. bq. I think MR over distributed cache shouldn't load configurations that included in MR tar ball. We explicitly have MapReduce load configurations that are in the tarball. We used to run without that and had the jobs pick up the configurations from the node, but we often ran into cases where rolling upgrades broke jobs on the cluster (e.g.: config changes to reference a plugin class that only exists in the new release, or old config setting breaks job running new release, etc.). Things got a lot simpler for us when we put the configs in the MR tarball and made sure those configs reflect the entire cluster. Added bonus: we don't need to push new confs to every node when the conf changes are just client-side (i.e.: not for the datanode/nodemanager). We just update our tarball in HDFS and every new job picks up the change. Back to the original reported issue: one reason not all security settings in job.xml aren't honored is because, unlike the *-site.xml files, job.xml is not added as a default Configuration resource. That means anything that simply instantiates a Configuration instance (e.g.: SecurityUtil, etc.) will not "see" job.xml settings. I'm not sure why job.xml wasn't added as a default resource. That may have been an intentional omission to avoid some problem, but part of me wonders if it should have been setup so that all Configuration objects start using job.xml settings if job.xml applies (i.e: we're using JobConf).
7. [~jlowe], thanks for your comments! For client settings, I agree that it would be convenient for loading configuration from tarball as we can honor different settings for different jobs (in kind of "batch" mode), especially from rolling upgrade prospectives. In addition, from usability prospective, I think client setting in job.xml has the highest priority to overwrite whatever conf in tar ball or not - given a bit overhead to change config setting in tarball. Isn't it? However, my next question is: do we think CommonConfigurationKeys.HADOOP_SECURITY_TOKEN_SERVICE_USE_IP is a client side setting? I assume we should keep this consistent in cluster level as mismatch between client and server setting will cause job get failed. If so, this actually belongs to a server-side setting, even it take effective in our current client side code.
8. bq. In addition, from usability prospective, I think client setting in job.xml has the highest priority to overwrite whatever conf in tar ball or not - given a bit overhead to change config setting in tarball. Isn't it? job.xml does override the tarball configs in almost every way _except_ this security setting because of the way that setting is loaded (i.e.: via Configuration instead of JobConf and job.xml is not a default resource). bq. If so, this actually belongs to a server-side setting, even it take effective in our current client side code. In practice that's essentially how it works in our setup. Since this property is currently not read from job.xml, it ends up using whatever is in the core-site.xml that's in the MR tarball. The MR tarball is created by the cluster admins and so the confs are consistent with the server settings for that cluster. A user could try overriding this property in their job.xml, but it wouldn't be used because job.xml is ignored for this specific config. (Yes, a user could provide their own, custom tarball with custom site files to override this setting, but they need to know what they're doing if they choose not to use the default MR tarball provided by the cluster admins.) Given this is a client-side setting that must be in sync with the server-side setting it'd be nice if this wasn't possible to get out-of-sync. For example, the server that granted the token could also communicate which setting must be used by the client later when the token is used, but that's a significant change with backward compatibility concerns. Of course there's also backwards compatibility concerns with adding job.xml as a default resource so that this property could be fetched from there instead of core-site.xml or core-default.xml.
9. bq. job.xml does override the tarball configs in almost every way except this security setting because of the way that setting is loaded To me this pretty much reveals the nature of a bug. Normally users would expect to have per-job configs override everything else, but this does not hold with the use ip setting. So one possible to fix this might be passing in the map reduce job configuration in security util, instead of using its own? In SecurityUtil, this limits the default behavior of use ip to be a newly created Configuration object, which may not be consistent with MR's job specific setting: {code} static { setConfigurationInternal(new Configuration()); } {code} And there is one API for this class to set the configuration used in SecurityUtil: {code} @InterfaceAudience.Public @InterfaceStability.Evolving public static void setConfiguration(Configuration conf) { LOG.info("Updating Configuration");

setConfigurationInternal(conf); } {code} So what we can do is to use the MR app's config to set this configuration?

10. bq. To me this pretty much reveals the nature of a bug. IMHO if we really want job.xml settings to override configs consistently then an easier, safer way to do that is making job.xml a default resource. Otherwise we're going to find more individual cases of this in the future where someone forgot to use the right Configuration type or pass the correct configs around. Good luck with code that's not MapReduce-aware, simply uses Configuration, and has no way of passing in a conf object. Either way we tackle this, we need to realize that the change will break scenarios that are working today where client-side configs do not have the proper value for hadoop.security.token.service.use_ip.
11. I probably won't agree it could be safe to let job.xml settings to override all configurations, especially serve-side configuration. Forget about MR tarball case, do we think this setting in job.xml should override what we have in local configuration? I think it is probably not - as I mentioned above, it won't have any benefit to give client flexibility to be different with what server settings. Isn't it? Base on that, back to MR tar ball case, if we think everything inside of MR tarball should be per job only, then only client configuration should work, but serve configuration (like case here) shouldn't get chance to override cluster setting. I am not fully agree that it should be cluster admin' job to create tarball and keep consistent for all configurations with cluster settings. In supportability prospective, making some server-side configurations to be transparent from client setting (job.xml or mr tar ball config) should be our job. Given that, I think the real problem now is: we should bypass server-side configurations in mr tar ball. Thoughts?
12. bq. it won't have any benefit to give client flexibility to be different with what server settings. Isn't it? Agreed, I think the best approach here long-term is to not have this be a config setting at all but rather derived from the communication session that retrieved the token in the first place. I also agree that it is going to be more dangerous than not for job.xml to override this particular setting (whether or not we're using an MR tarball). bq. if we think everything inside of MR tarball should be per job only This is definitely not the case, at least for our clusters. As I mentioned above, the MR tarballs are created by the cluster admins and therefore have the appropriate configs for that cluster. We do _not_ want the jobs to pick up configs from the nodes per the issues I described earlier. bq. I am not fully agree that it should be cluster admin' job to create tarball and keep consistent for all configurations with cluster settings. The tarball consist of the Hadoop jars and the *-site.xml files. Both of these are things admins of the cluster are expected to maintain and provide. Therefore I think it's completely appropriate that typically these tarballs are created and provided by admins. We're not running a bunch of different versions of the tarball for different types of jobs. We just have the main tarball for the cluster and users override various settings for their specific job via job.xml, just like a regular non-tarball job does. To be clear, I'm not saying everyone has to run it the way we do. However if you don't then there needs to be solutions for the types of rolling upgrade problems I pointed out above. Also I don't think the way we're using the tarball is an invalid setup, so therefore any proposed change needs to ensure this type of setup keeps working.
13. The more I think about this, the more I believe the original problem is describing an invalid setup. I can accomplish the same thing without a tarball by shipping my own custom core-site.xml that appears _before_ any core-site.xml provided by the admins of the cluster. I think we can agree that any such setup that fails due to a bad setting in the eclipsing core-site.xml is a fault of the user's setup and not a fault of the Hadoop framework itself. So I would argue the fault lies with construction of the job in question that failed. Either it used a faulty tarball that had a bad core-site.xml in it, or the core-site.xml was missing completely from the classpath (which may have been the case here).
14. bq. We just have the main tarball for the cluster and users override various settings for their specific job via job.xml, just like a regular non-tarball job does Our case is similar but slightly different. For every release, we generally only provide one tarball for all users' clusters. So every user can specify their cluster's own various settings without touch tarball that we handle over. Otherwise, taking care of multiple versions of tarball for the same release will be supportability disaster that we cannot afford. Can we make a consensus that everything in tarball's config is not final (unless we explicitly mark it as final)? If so, then may be we can simply go ahead to make job.xml as the highest priority without differentiating client/server settings. Any risk I am missing here?
15. **body:** bq. Can we make a consensus that everything in tarball's config is not final (unless we explicitly mark it as final)? I think the tarball is muddying the waters here. *-site.xml files should be treated the same whether they are in a tarball or not. It'd be tricky and messy to do otherwise. Essentially what we're asking is whether clients should be able to override _any_ non-final setting in the *-site.xml files with their job.xml setting, even if that setting is a "server side" property. bq. may be we can simply go ahead to make job.xml as the highest priority without differentiating client/server settings. Any risk I am missing here? Making job.xml a default resource accomplishes that proposal if we want to go that route. Note:

Usually job.xml contains the contents of the client-side *-site.xml files, so as long as those match the cluster we should be good there. The risk of this change is when the client's *-site.xml files do _not_ match what should be there. For the "server side" settings this has been working because we've been ignoring job.xml for those. Once we start using job.xml for even those properties, jobs that were working in the past because we ignored bad values will break. I don't know offhand how many other properties besides this one could suddenly change because we start using the client's version of the property in job.xml when we didn't before. I agree with [~gtCarrera9] that it's more consistent and less surprising to users if job.xml settings override any other settings in the job. However there are going to be some cases that break when we "fix" it. That's why I'm a bit hesitant, especially if this is going into 2.x.
label: code-design

16. **body:** bq. I think the tarball is muddying the waters here. *-site.xml files should be treated the same whether they are in a tarball or not. It'd be tricky and messy to do otherwise. Essentially what we're asking is whether clients should be able to override any non-final setting in the *-site.xml files with their job.xml setting, even if that setting is a "server side" property Agree. Whether clients are allowed to override any non-final setting is something I originally asking for. But later, I realized that if we disallow client to override "server-setting" - like case here, tar-ball configuration will become the final setting for "server-side" configuration. It also means MR tarball will become various according to cluster settings but not according to releases. Standing at support-ability prospective, we don't want our users to touch tarball settings because it is hard to control/monitor these settings by Ambari or other tools. Otherwise, we have to dig into tarball settings when weird things happens. In this sense, I think it could be simpler if we don't differentiate client-side and server-side setting in configuration loading. bq. The risk of this change is when the client's *-site.xml files do not match what should be there. For the "server side" settings this has been working because we've been ignoring job.xml for those. Once we start using job.xml for even those properties, jobs that were working in the past because we ignored bad values will break. I don't know offhand how many other properties besides this one could suddenly change because we start using the client's version of the property in job.xml when we didn't before. I think even before, our configuration loading doesn't guarantee we only loading client-side configuration rather than server-side configuration. Even for mapred-site.xml, there are history server related settings that belongs to server-side but you can override a different value in client side which just doesn't work though. So we still need to rely on code logic to bypass what settings belongs to server-setting in client side. Also, better documentation could be helpful for user to differentiate what's client-side and what's server-side. Our current tricky loading mechanism won't help on both side.
    label: code-design
17. I'm +1 for making all client-side settings override anything in any site file that isn't marked final for 3.x. I'm a bit hesitant for 2.x given the long-standing semantics for some of these properties, and in any case there needs to be a clear release note explaining to users what to expect with the change. Hmm, there might be a problem with adding job.xml as a default resource, and that has to do with relative ordering of when job.xml is added and other default resources like the *-site.xml files are added. The various site.xml files are only added as defaults when the related classes are loaded (e.g.: HdfsConfiguration, YarnConfiguration, JobConf, etc.) If we add job.xml as a default resource _before_ some of these classes are touched then some site files will override the job.xml files because they'll be loaded later. We can probably get the ordering right for all the site files provided by core Hadoop, but I'm worried about downstream projects that may have their own site files (e.g.: hive-site.xml). Client-side settings could be smashed by site settings if the ordering is not correct. job.xml would need to be the last default resource added, and we may not be able to guarantee that with arbitrary downstream code. Unfortunately without using the default resource feature of Configuration, I don't know of a straightforward way to get classes using plain ol' Configuration instances to see values set in job.xml. Any ideas here? We can fix individual instances like hadoop.security.token.service.use_ip in case-specific ways (i.e.: calling the SecurityUtil.setTokenServiceUseIp method for this property), but not all cases will have a straightforward fix. And we'd have to track them all down individually.
18. **body:** A quick patch to solely resolve the use_ip issue. We followed the pattern of UGI.setConfiguration to set configuration for SecurityUtil as well. This is my best-effort so far but I'm not very familiar with MR code base. So please do feel free to point out if there are more call sites needed. Also, to me adding an unit test in this case is non-trivial. Do we have any existing test infrastructure that can be helpful for this issue? Thanks!
    label: test
19. Thanks for all the attention you guys have paid to this issue. My initial hack works well with MR job, however when I did a test on Spark job submitted remotely to yarn cluster, the hack did not work. i did not open another jira for spark job because i did not quite dig into the spark job settings to see who it was

handled. Maybe you guys have more insight into the code and can figure out how to solve this problem for all YARN managed applications.

20. bq. Maybe you guys have more insight into the code and can figure out how to solve this problem for all YARN managed applications. I believe so far we've only seen this problems occurred on MR apps with specific ways of distributions (tarballs through distributed cache). No?
21. | (x) *{color:red}-1 overall{color}* | \\ \\ || Vote || Subsystem || Runtime || Comment || | {color:blue}0{color} | {color:blue} reexec {color} | {color:blue} 0m 13s {color} | {color:blue} Docker mode activated. {color} | | {color:green}+1{color} | {color:green} @author {color} | {color:green} 0m 0s {color} | {color:green} The patch does not contain any @author tags. {color} | | {color:red}-1{color} | {color:red} test4tests {color} | {color:red} 0m 0s {color} | {color:red} The patch doesn't appear to include any new or modified tests. Please justify why no new tests are needed for this patch. Also please list what manual steps were performed to verify this patch. {color} | | {color:green}+1{color} | {color:green} mvninstall {color} | {color:green} 7m 15s {color} | {color:green} trunk passed {color} | | {color:green}+1{color} | {color:green} compile {color} | {color:green} 0m 25s {color} | {color:green} trunk passed {color} | | {color:green}+1{color} | {color:green} checkstyle {color} | {color:green} 0m 17s {color} | {color:green} trunk passed {color} | | {color:green}+1{color} | {color:green} mvnsite {color} | {color:green} 0m 32s {color} | {color:green} trunk passed {color} | | {color:green}+1{color} | {color:green} mvneclipse {color} | {color:green} 0m 16s {color} | {color:green} trunk passed {color} | | {color:green}+1{color} | {color:green} findbugs {color} | {color:green} 0m 38s {color} | {color:green} trunk passed {color} | | {color:green}+1{color} | {color:green} javadoc {color} | {color:green} 0m 16s {color} | {color:green} trunk passed {color} | | {color:green}+1{color} | {color:green} mvninstall {color} | {color:green} 0m 22s {color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} compile {color} | {color:green} 0m 22s {color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} javac {color} | {color:green} 0m 22s {color} | {color:green} the patch passed {color} | | {color:red}-1{color} | {color:red} checkstyle {color} | {color:red} 0m 15s {color} | {color:red} hadoop-mapreduce-project/hadoop-mapreduce-client/hadoop-mapreduce-client-app: The patch generated 1 new + 90 unchanged - 1 fixed = 91 total (was 91) {color} | | {color:green}+1{color} | {color:green} mvnsite {color} | {color:green} 0m 27s {color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} mvneclipse {color} | {color:green} 0m 13s {color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} whitespace {color} | {color:green} 0m 0s {color} | {color:green} The patch has no whitespace issues. {color} | | {color:green}+1{color} | {color:green} findbugs {color} | {color:green} 0m 43s {color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} javadoc {color} | {color:green} 0m 14s {color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} unit {color} | {color:green} 8m 56s {color} | {color:green} hadoop-mapreduce-client-app in the patch passed. {color} | | {color:green}+1{color} | {color:green} asflicense {color} | {color:green} 0m 17s {color} | {color:green} The patch does not generate ASF License warnings. {color} | | {color:black}{color} | {color:black} {color} | {color:black} 22m 20s {color} | {color:black} {color} | \\ \\ || Subsystem || Report/Notes || | Docker | Image:yetus/hadoop:a9ad5d6 | | JIRA Patch URL | https://issues.apache.org/jira/secure/attachment/12840334/MAPREDUCE-6565-trunk.001.patch | | JIRA Issue | MAPREDUCE-6565 | | Optional Tests | asflicense compile javac javadoc mvninstall mvnsite unit findbugs checkstyle | | uname | Linux 4ebd81471565 3.13.0-95-generic #142-Ubuntu SMP Fri Aug 12 17:00:09 UTC 2016 x86_64 x86_64 x86_64 GNU/Linux | | Build tool | maven | | Personality | /testptch/hadoop/patchprocess/precommit/personality/provided.sh | | git revision | trunk / 0de0c32 | | Default Java | 1.8.0_111 | | findbugs | v3.0.0 | | checkstyle | https://builds.apache.org/job/PreCommit-MAPREDUCE-Build/6820/artifact/patchprocess/diff-checkstyle-hadoop-mapreduce-project_hadoop-mapreduce-client_hadoop-mapreduce-client-app.txt | | Test Results | https://builds.apache.org/job/PreCommit-MAPREDUCE-Build/6820/testReport/ | | modules | C: hadoop-mapreduce-project/hadoop-mapreduce-client/hadoop-mapreduce-client-app U: hadoop-mapreduce-project/hadoop-mapreduce-client/hadoop-mapreduce-client-app | | Console output | https://builds.apache.org/job/PreCommit-MAPREDUCE-Build/6820/console | | Powered by | Apache Yetus 0.3.0 http://yetus.apache.org | This message was automatically generated.
22. this.happens across board ....as long as job is submitted remotely
23. That's not exactly the same issue as we observed here. Can you try latest trunk to verify this issue? I would not be surprised if problems occur on all YARN apps after a fix in YarnClient, but I believe the issue discussed here is solely about MR.
24. Patch looks good to me, +1. It will address the specific issue for MapReduce. Other frameworks, such as Spark, will need to make similar modifications.
25. Patch LGTM too.

26. SUCCESS: Integrated in Jenkins build Hadoop-trunk-Commit #10910 (See [https://builds.apache.org/job/Hadoop-trunk-Commit/10910/]) MAPREDUCE-6565. Configuration to use host name in delegation token (junping_du: rev 8f6e14399a3e77e1bdcc5034f7601e9f62163dea) * (edit) hadoop-mapreduce-project/hadoop-mapreduce-client/hadoop-mapreduce-client-app/src/main/java/org/apache/hadoop/mapreduce/v2/app/MRAppMaster.java * (edit) hadoop-mapreduce-project/hadoop-mapreduce-client/hadoop-mapreduce-client-app/src/main/java/org/apache/hadoop/mapred/YarnChild.java
27. I have commit the patch to trunk, branch-2 and branch-2.8. Thanks [~gtCarrera9] for contributing the patch and [~jlowe] for review! Also, thanks [~cnauroth] for reporting the issue and others for comments.
28. [~junping_du], this seems to have broken the 2.8 build. Can you revert it from 2.8 until we can fix the patch?
29. Yes we need HADOOP-12954 prior to the fix here. Shall we back port that issue as well?
30. My bad... I forgot branch-2 and branch-2.8 has big gap already and I only compile on branch-2... Let me revert it from 2.8 first and I will check HADOOP-12954 if it need backport to branch-2.8.