Item 144
git_comments:

git_commits:

1. **summary:** [SPARK-25981][R] Enables Arrow optimization from R DataFrame to Spark DataFrame
   message: [SPARK-25981][R] Enables Arrow optimization from R DataFrame to Spark DataFrame ## What changes were proposed in this pull request? This PR targets to support Arrow optimization for conversion from R DataFrame to Spark DataFrame. Like PySpark side, it falls back to non-optimization code path when it's unable to use Arrow optimization. This can be tested as below: ```bash $ ./bin/sparkR --conf spark.sql.execution.arrow.enabled=true ``` ```r collect(createDataFrame(mtcars)) ``` ### Requirements - R 3.5.x - Arrow package 0.12+ ```bash Rscript -e 'remotes::install_github("apache/arrowapache-arrow-0.12.0", subdir = "r")' ``` **Note:** currently, Arrow R package is not in CRAN. Please take a look at ARROW-3204. **Note:** currently, Arrow R package seems not supporting Windows. Please take a look at ARROW-3204. ### Benchmarks **Shall** ```bash sync && sudo purge ./bin/sparkR --conf spark.sql.execution.arrow.enabled=false ``` ```bash sync && sudo purge ./bin/sparkR --conf spark.sql.execution.arrow.enabled=true ``` **R code** ```r createDataFrame(mtcars) # Initializes rdf <- read.csv("500000.csv") test <- function() { options(digits.secs = 6) # milliseconds start.time <- Sys.time() createDataFrame(rdf) end.time <- Sys.time() time.taken <- end.time - start.time print(time.taken) } test() ``` **Data (350 MB):** ```r object.size(read.csv("500000.csv")) 350379504 bytes ``` "500000 Records" http://eforexcel.com/wp/downloads-16-sample-csv-files-data-sets-for-testing/ **Results** ``` Time difference of 29.9468 secs ``` ``` Time difference of 3.222129 secs ``` The performance improvement was around **950%**. Actually, this PR improves around **1200%**+ because this PR includes a small optimization about regular R DataFrame -> Spark DatFrame. See https://github.com/apache/spark/pull/22954#discussion_r231847272 ### Limitations: For now, Arrow optimization with R does not support when the data is `raw`, and when user explicitly gives float type in the schema. They produce corrupt values. In this case, we decide to fall back to non-optimization code path. ## How was this patch tested? Small test was added. I manually forced to set this optimization `true` for _all_ R tests and they were _all_ passed (with few of fallback warnings). **TODOs:** - [x] Draft codes - [x] make the tests passed - [x] make the CRAN check pass - [x] Performance measurement - [x] Supportability investigation (for instance types) - [x] Wait for Arrow 0.12.0 release - [x] Fix and match it to Arrow 0.12.0 Closes #22954 from HyukjinKwon/r-arrow-createdataframe. Lead-authored-by: hyukjinkwon <gurwls223@apache.org> Co-authored-by: Hyukjin Kwon <gurwls223@apache.org> Signed-off-by: Hyukjin Kwon <gurwls223@apache.org>

github_issues:

github_issues_comments:

github_pulls:

1. **title:** [SPARK-25981][R] Enables Arrow optimization from R DataFrame to Spark DataFrame
   body: ## What changes were proposed in this pull request? This PR targets to support Arrow optimization for conversion from R DataFrame to Spark DataFrame. Like PySpark side, it falls back to non-optimization code path when it's unable to use Arrow optimization. This can be tested as below: ```bash $ ./bin/sparkR --conf spark.sql.execution.arrow.enabled=true ``` ```r collect(createDataFrame(mtcars)) ``` ### Requirements - R 3.5.x - Arrow package 0.12+ ```bash Rscript -e 'remotes::install_github("apache/arrow@apache-arrow-0.12.0", subdir = "r")' ``` **Note:** currently, Arrow R package is not in CRAN. Please take a look at ARROW-3204. **Note:** currently, Arrow R package seems not supporting Windows. Please take a look at ARROW-3204. ### Benchmarks **Shall** ```bash sync && sudo purge ./bin/sparkR --conf spark.sql.execution.arrow.enabled=false ``` ```bash sync && sudo purge ./bin/sparkR --conf spark.sql.execution.arrow.enabled=true ``` **R code** ```r createDataFrame(mtcars) # Initializes rdf <- read.csv("500000.csv") test <- function() { options(digits.secs = 6) # milliseconds start.time <- Sys.time() createDataFrame(rdf) end.time <- Sys.time() time.taken <- end.time - start.time print(time.taken) } test() ``` **Data (350 MB):** ```r object.size(read.csv("500000.csv")) 350379504 bytes ``` "500000 Records" http://eforexcel.com/wp/downloads-16-sample-csv-files-data-sets-for-testing/ **Results** ``` Time difference of 29.9468 secs ``` ``` Time difference of 3.222129 secs ``` The performance improvement was around **950%**. Actually, this PR improves around **1200%**+ because this PR includes a small optimization about regular R DataFrame -> Spark DatFrame. See https://github.com/apache/spark/pull/22954#discussion_r231847272 ### Limitations: For now, Arrow optimization with R does not support when the data is `raw`, and when user explicitly gives float type in the schema. They produce corrupt values. In this case, we decide to fall back to non-optimization code path. ## How was this patch tested? Small test was added. I manually forced to set this optimization `true` for _all_ R tests and they were _all_ passed (with few of fallback warnings). **TODOs:** - [x] Draft codes - [x] make the tests passed - [x] make the CRAN check pass - [x] Performance measurement - [x] Supportability investigation (for instance types) - [x] Wait for Arrow 0.12.0 release - [x] Fix and match it to Arrow 0.12.0

github_pulls_comments:

1. Let me leave a cc @felixcheung, @BryanCutler, @yanboliang, @shivaram FYI.
2. **[Test build #98508 has finished]
   (https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/98508/testReport)** for PR 22954 at commit

[`90011a5`](https://github.com/apache/spark/commit/90011a5ff48f2c5fa5fae0e2573fcdaa85d44976). * This patch **fails to generate documentation**. * This patch merges cleanly. * This patch adds no public classes.

3. **[Test build #98510 has finished] (https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/98510/testReport)** for PR 22954 at commit [`46eaeca`](https://github.com/apache/spark/commit/46eaeca5854281a5688badbe71981029096674a5). * This patch **fails to generate documentation**. * This patch merges cleanly. * This patch adds no public classes.
4. **[Test build #98512 has finished] (https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/98512/testReport)** for PR 22954 at commit [`614170e`](https://github.com/apache/spark/commit/614170e2187ebb904a767adb9289ac48300e533c). * This patch **fails SparkR unit tests**. * This patch merges cleanly. * This patch adds no public classes.
5. **[Test build #98514 has finished] (https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/98514/testReport)** for PR 22954 at commit [`b15d79c`](https://github.com/apache/spark/commit/b15d79c0d734d04a783470d793cf52e5c6c99b9d). * This patch **fails SparkR unit tests**. * This patch merges cleanly. * This patch adds no public classes.
6. So far, the regressions tests are passed and newly added test for R optimization is verified locally. Let me fix CRAN test and some nits.
7. Thanks, @felixcheung. I will address those comments during cleaning up.
8. For encryption stuff, I will try to handle that as well (maybe as a followup(?)) so that we support it even when that's enabled.
9. @felixcheung! performance improvement was **955%** ! I described the benchmark I took in PR description.
10. adding @falaki and @mengxr as well.
11. **[Test build #98595 has finished] (https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/98595/testReport)** for PR 22954 at commit [`8813192`](https://github.com/apache/spark/commit/881319298b844d934b1eb02994d7f1a85274efaa). * This patch **fails SparkR unit tests**. * This patch merges cleanly. * This patch adds no public classes.
12. **[Test build #98603 has finished] (https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/98603/testReport)** for PR 22954 at commit [`7be15d3`](https://github.com/apache/spark/commit/7be15d36989ba1a00a86bb5eeb0b3d6fb7da3c4c). * This patch **fails SparkR unit tests**. * This patch merges cleanly. * This patch adds no public classes.
13. **[Test build #98613 has started] (https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/98613/testReport)** for PR 22954 at commit [`2ddbd69`](https://github.com/apache/spark/commit/2ddbd694663b7ff0f5b3a8099ba87e4800397a57).
14. I have finished most of todos except waiting for R API of Arrow 0.12.0 and fixing some changes accordingly.
15. **[Test build #98615 has finished] (https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/98615/testReport)** for PR 22954 at commit [`2ba6add`](https://github.com/apache/spark/commit/2ba6addbcd52940ef989880bff69fe126a4dd2e1). * This patch **fails Spark unit tests**. * This patch merges cleanly. * This patch adds no public classes.
16. **[Test build #98614 has finished] (https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/98614/testReport)** for PR 22954 at commit [`0903736`](https://github.com/apache/spark/commit/0903736b46e1ff737577f7db3f29db67795701ee). * This patch **fails SparkR unit tests**. * This patch merges cleanly. * This patch adds no public classes.
17. retest this please
18. **[Test build #98628 has finished] (https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/98628/testReport)** for PR 22954 at commit [`2ba6add`](https://github.com/apache/spark/commit/2ba6addbcd52940ef989880bff69fe126a4dd2e1). * This patch passes all tests. * This patch merges cleanly. * This patch adds no public classes.
19. I don't know R well enough to review that code, but the results look awesome! Nice work @HyukjinKwon!!
20. Hey guys thanks for reviewing! Will address them soon.
21. Let me hide some comments that are addressed (it looks messy). Please make unhide if I mistakenly hide some comments that are not addressed yet.
22. **[Test build #98690 has finished] (https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/98690/testReport)** for PR 22954 at commit [`6f28aa5`](https://github.com/apache/spark/commit/6f28aa5c79854cc3df176e0fa0fa6f3e7d21a98a). * This patch **fails SparkR unit tests**. * This patch merges cleanly. * This patch adds no public classes.
23. **[Test build #98692 has finished] (https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/98692/testReport)** for PR 22954 at commit [`a81134e`](https://github.com/apache/spark/commit/a81134efe787bbd3b7a5b92d03b91e5f4e2f3cf6). * This patch **fails SparkR unit tests**. * This patch merges cleanly. * This patch adds no public classes.
24. Hm .. the CRAN passed in my local. Let me workaround for now.
25. **[Test build #98696 has finished] (https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/98696/testReport)** for PR 22954 at commit [`b2e0fc2`](https://github.com/apache/spark/commit/b2e0fc2d9e5e18b334b0177157ebd282b654c0a0). * This patch **fails some tests**. * This patch merges cleanly. * This patch adds no public classes.
26. **[Test build #98697 has finished] (https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/98697/testReport)** for PR 22954 at commit [`92419d0`](https://github.com/apache/spark/commit/92419d055c9f9eb4cbe1172158863866b4401b4b). * This patch **fails PySpark unit tests**. * This patch merges cleanly. * This patch adds no public classes.

27. **[Test build #98699 has finished] (https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/98699/testReport)** for PR 22954 at commit [`c3f47ce`](https://github.com/apache/spark/commit/c3f47ce04c82f6fb6efe6d55baa19ddc8cf50dd3). * This patch passes all tests. * This patch merges cleanly. * This patch adds no public classes.
28. Yea .. I will make the followup works right away after this one get merged. Thanks @felixcheung. Let me address the rest of comments, and wait for Arrow release. @BryanCutler BTW, do you know the rough expected timing for Arrow 0.12.0 release?
29. **[Test build #98713 has finished] (https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/98713/testReport)** for PR 22954 at commit [`d9d9f98`](https://github.com/apache/spark/commit/d9d9f982d26a5dd2141515e0c9089243b7b93554). * This patch passes all tests. * This patch merges cleanly. * This patch adds no public classes.
30. **[Test build #98755 has finished] (https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/98755/testReport)** for PR 22954 at commit [`954bc0e`](https://github.com/apache/spark/commit/954bc0eec206902cb8176338e1f72886f5b3c626). * This patch **fails due to an unknown error code, -9**. * This patch merges cleanly. * This patch adds no public classes.
31. retest this please.
32. **[Test build #98760 has finished] (https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/98760/testReport)** for PR 22954 at commit [`954bc0e`](https://github.com/apache/spark/commit/954bc0eec206902cb8176338e1f72886f5b3c626). * This patch passes all tests. * This patch merges cleanly. * This patch adds no public classes.
33. > @BryanCutler BTW, do you know the rough expected timing for Arrow 0.12.0 release? I think we should be starting the release process soon, so maybe in a week or two.
34. FYI the release tracker (quite useful!) https://cwiki.apache.org/confluence/display/ARROW/Arrow+0.12.0+Release
35. @HyukjinKwon it's released!
36. btw @HyukjinKwon would you be interested in a post about this, similar to ARROW-4262?
37. Sure!
38. Let me finish this one and then check the appropriate place to make the documentation (maybe around https://spark.apache.org/docs/latest/sql-pyspark-pandas-with-arrow.html?).
39. I was thinking a blog post in the Arrow project ;)
40. Gotya, yea, I am interested in it of course. I'll start to work on that after this PR merged.
41. **[Test build #101513 has finished] (https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/101513/testReport)** for PR 22954 at commit [`767af86`](https://github.com/apache/spark/commit/767af864a328b8dfb7bbcecea331e3f84ab45149). * This patch **fails due to an unknown error code, -9**. * This patch merges cleanly. * This patch adds no public classes.
42. retest this please
43. **[Test build #101551 has finished] (https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/101551/testReport)** for PR 22954 at commit [`767af86`](https://github.com/apache/spark/commit/767af864a328b8dfb7bbcecea331e3f84ab45149). * This patch passes all tests. * This patch merges cleanly. * This patch adds no public classes.
44. To cut it short, I think this PR is ready to go. I reran the benchmark, and updated PR descriptions. Few things to mention: 1. Arrow is not related on CRAN and looks it's going to take few months (see [ARROW-3204] (https://issues.apache.org/jira/browse/ARROW-3204)). So, for now, it should be manually installed. - It can be installed by `Rscript -e 'remotes::install_github("apache/arrow@apache-arrow-0.12.0", subdir = "r")'`. - I used maxOS Mojave 10.14.2 and faced some problems to fix at my env. Please connect me if you guys face some issue during installing this. If this is globally happening, I will document this somewhere. 2. Looks we can run the build via AppVeyor when it's on CRAN (see [ARROW-3204](https://issues.apache.org/jira/browse/ARROW-3204)). 3. We should remove the workarounds that I used to avoid CRAN check (see https://github.com/apache/spark/pull/22954#discussion_r250585248 and https://github.com/apache/spark/pull/22954#discussion_r250618871) Next items (im going to investigate first before filing JIRAs): 1. Im gonna take a look if we can do this Spark DataFrame -> R DataFrame too 2. Also, I'm going to take a look for R native function APIs like lapply and gapply and see if we can optimize this 3. Before Spark 3.0 release, I will document this. Hopefully, we can get rid of both workaround I mentioned above and Arrow is on CRAN before this.
45. **[Test build #101633 has finished] (https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/101633/testReport)** for PR 22954 at commit [`66b120b`](https://github.com/apache/spark/commit/66b120b3dbf3d0e6581429ec6cf10131452b5bb7). * This patch **fails Spark unit tests**. * This patch merges cleanly. * This patch adds no public classes.
46. **[Test build #101630 has finished] (https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/101630/testReport)** for PR 22954 at commit [`92eec4e`](https://github.com/apache/spark/commit/92eec4e0d63bb28a0d84d2761863f35751a3f448). * This patch **fails Spark unit tests**. * This patch merges cleanly. * This patch adds no public classes.
47. **[Test build #101632 has finished] (https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/101632/testReport)** for PR 22954 at commit [`854c9d8`](https://github.com/apache/spark/commit/854c9d83f1b1b41ffdac44ffd25072c2d5df9eb0). * This patch passes all tests. * This patch merges cleanly. * This patch adds no public classes.
48. @felixcheung and @shivaram, are you okay with this plan https://github.com/apache/spark/pull/22954#issuecomment-457214769 ? If so, I think we can go ahead.
49. Yes, install_github is fine. I'd say #2 gapply etc is higher priority. Sounds good to me.
50. Yea will do. Do you mind if we go ahead with this PR @felixcheung?
51. Sure.

52. Thanks. @felixcheung. Merged to master.
53. BTW, https://issues.apache.org/jira/browse/SPARK-26759 has subtasks for Arrow optimization (just FYI if anyone missed it)

github_pulls_reviews:

1. perhaps best to add a clearer error message?
2. is it possible to not depend on this withr?
3. btw, is it worthwhile to check the arrow package version?
4. I think you can use sparkR.conf
5. might need to give it a dir prefix to use - the tempfile default is not CRAN compliant and possibly some ACL issue
6. It's actually dependent on Arrow's API calls .. Those APIs were only added in few weeks ago. I will make it clean when R API of Arrow is close to release (at 0.12.0)
7. hmmm .. yea I will add the version check later when R API of Arrow is officially released to CRAN.
8. Note that this PR optimizes the original code path as well here - when the input is local R DataFrame, here we avoid `firstRDD` operation. In the master branch, the benchmark shows: ``` Exception in thread "dispatcher-event-loop-6" java.lang.OutOfMemoryError: Java heap space at java.util.Arrays.copyOf(Arrays.java:3236) at java.io.ByteArrayOutputStream.grow(ByteArrayOutputStream.java:118) at java.io.ByteArrayOutputStream.ensureCapacity(ByteArrayOutputStream.java:93) at java.io.ByteArrayOutputStream.write(ByteArrayOutputStream.java:153) at org.apache.spark.util.ByteBufferOutputStream.write(ByteBufferOutputStream.scala:41) at java.io.ObjectOutputStream$BlockDataOutputStream.drain(ObjectOutputStream.java:1877) ``` If I try this with `100000.csv` (79MB) record, it takes longer. To cut it short: **Current master**: ``` Time difference of 8.502607 secs ``` **With this PR, but without Arrow** ``` Time difference of 5.143395 secs ``` **With this PR, but with Arrow** ``` Time difference of 0.6981369 secs ``` So, technically this PR improves more then **1200%**
9. These workarounds will be removed when Arrow 0.12.0 is released. I did it to make CARN passed.
10. It's actually a bit odd that I need to manually require this package. Otherwise, it complains, for instance, here https://github.com/apache/arrow/blob/d3ec69069649013229366ebe01e22f389597dc19/r/R/on_exit.R#L20
11. I <3 4 digits!
12. ok
13. require1 sounds a bit like a hack though...
14. what's the advantage of adding this wrapper here - I've thinking to eliminate most of these if possible - and just use callJMethod on `ArrowConverters` say?
15. should this go by default with default parallelism ?
16. `1 : ceiling`? `1 : nrow`?
17. how is this slices computed? is it similar to https://github.com/apache/spark/blob/c3b4a94a91d66c172cf332321d3a78dba29ef8f0/R/pkg/R/context.R#L152
18. so is this an API missing in Arrow?
19. actually, I think if you use withr here it will call close_on_exit for you? (but when?)
20. is it always in the conf - I think you can also pass in a default value to sparkR.conf
21. can you check - I think `is` `is.x` doesn't something do the right thing when head(df, 1) and one of the field is `NA`
22. ? https://stat.ethz.ch/R-manual/R-devel/library/base/html/warning.html
23. unlink?
24. refactor this into a method?
25. ooops
26. nit: a `RDD` -> an `RDD`
27. Any idea what's going on with the `FloatType`? Is it a problem on the arrow side?
28. nit: `arrow` -> `Arrow`
29. Yup .. it is .. looks we shouldn't have this error from a cursory look in R API of Arrow. Maybe this can be gone when I use official R Arrow release version. Let me check it later.
30. Hmhmhm .. yea. What I was trying to do is to add SQL related codes called in R from JVM, into here when they are not official APIs in order to avoid: we change the internal API's signature within Scala, and it causes R test failure. I was trying to do the similar things within PySpark side. Let me keep it if you don't mind.
31. We should; however, it follows the original code path's behaviour. I matched it as the same so that we can compare the performances in the same conditions. If you don't mind, I will fix both in a separate PR.
32. This resembles PySpark side logic: https://github.com/apache/spark/blob/d367bdcf521f564d2d7066257200be26b27ea926/python/pyspark/sql/session.py#L554-L556
33. Hm, possibly yea. Let me try it. In this way, we could get rid of `require`.
34. Yea,I checked that it always has the default value. I initially left the default value but took it out after double checking.
35. Yup, let me try.
36. I suspect that it happens when `numeric` (which is like `1.0`) is casted into float type. I think it's related with casting behaviour. Let me take a look and file a JIRA there in Arrow side but if you don't mind I will focus on matching exact type cases for now ...
37. Yea, I at least managed to get rid of this hack itself. Will push soon.
38. I believe either way is fine.
39. Looks okay in this case specifically: ```r > any(sapply(head(data.frame(list(list(a=NA))), 1), is.raw)) [1] FALSE > any(sapply(head(data.frame(list(list(a=NA))), 1), function(x) is(x, "POSIXct"))) [1] FALSE >

any(sapply(head(data.frame(list(list(a=1))), 1), is.raw)) [1] FALSE > any(sapply(head(data.frame(list(list(a="a"))), 1), function(x) is(x, "POSIXct"))) [1] FALSE > any(sapply(head(data.frame(list(list(a=raw(1)))), 1), is.raw)) [1] TRUE > any(sapply(head(data.frame(list(list(a=as.POSIXct("2000-01-01")))), 1), function(x) is(x, "POSIXct"))) [1] TRUE ```

40. Let me try to reuse the R side slicing logic.
41. hmm, I see your point... but there could be hundreds of these wrapper we need add if we set as a practice, I'm guessing. a few problems with these wrappers I see: 1. they are extra work to add or maintain 2. many are very simple, not much value add 3. many get abandoned over the years - they are not called and not removed but I kinda see your way, let's keep this one and review any new one in the future.
42. ah, any idea that was done that way in python? this seems to be different from sc.paralleize which is what https://github.com/apache/spark/blob/c3b4a94a91d66c172cf332321d3a78dba29ef8f0/R/pkg/R/context.R#L152 is implementing (scala code in ParallelCollectionRDD)
43. ok
44. LG, I tested a few cases too
45. yes, just more consistent. I also don't know for sure why all other instances are calling unlink
46. maybe out of bit range? 53 bit https://stat.ethz.ch/R-manual/R-patched/library/base/html/double.html (numeric is double in R)
47. that will be good. circumventing CRAN check for method name is... problematic.. (there are other more hacky way too, but ..)
48. Not sure. I think the intention is the same. Let me stick to R's one for now.
49. does this fail? or just a way to inject a finally?
50. are we going to ask shane to install arrow/withr on the Jenkins machines?
51. future: consolidate the default here and inside makeSplits
52. nit: just `FALSE` is good
53. Just to inject the finally .. :-) ..
54. Maybe we should hold it for now .. because I realised R API for Arrow requires R 3.5.x and Jenkins's one is 3.1.x if I remember this correctly. Ideally, we could probably do that via AppVeyor first if everything goes fine.
55. Should we move `writeToTempFileInArrow` call into next `tryCatch`?
56. When `shouldUseArrow` is true, I think it means we already done using arrow? Maybe just `useArrow`?
57. Is this going to read a file in Arrow stream format? Or the comment should be fixed.
58. Yup, correct. Let me address other comments as well.
59. Oh if it's only when casting to a float, then maybe not that big of an issue. I just wanted to make sure a bug was filed for Arrow if the problem is there.
60. I think it's a bug because it always produces a corrupt value when I try to use `number` as explicit float types.
61. super nit if you will need additional commit: `... Pandas DataFrame," +` -> `... Pandas DataFrame, " +`
62. Oh yeah. I need to push additional commit when it's released. Will fix it together later.
63. FWIW, sparklyr and arrow implementation uses the same trick to avoid CRAN failure.
64. Assuming from ARROW-3204, Arrow is still not in the CRAN and looks it's going to take few months, and looks we can run the build via AppVeyor when it's on CRAN.
65. yes, published to CRAN not happening yet for Arrow. install_github should be fine? it's a one time thing

jira_issues:

1. **summary:** [R] Enable package to be made available on CRAN
   description: As of [ARROW-1325|https://issues.apache.org/jira/browse/ARROW-1325], the project contains a minimal working R package that takes advantage of R's built-in support for calling C++ code and uses Rcpp for added support. Though the exact public interface of the package hasn't been developed yet (so this issue may be running before we walk), one major feature in its development will be release to [CRAN|https://cran.r-project.org/], the official package repository for the R language. Completing this story would mean: * Getting source code to state where [R CMD CHECK|https://www.r-bloggers.com/how-i-learned-to-stop-worrying-and-love-r-cmd-check/] passes with 0 ERRORS, 0 WARNINGS, and 0 NOTES * Setting up build process for source and precompiled binary packages (see [data.table|https://github.com/Rdatatable/data.table] and [XGBoost|https://github.com/dmlc/xgboost] as examples of packages that do this) * Submission to CRAN and acceptance of the first release Distribution via CRAN would be a much more natural fit for most R users' workflows than the current build-from-source workflow, and comes with all the other benefits of package managers (e.g. version pegging, easy distribution of platform-specific binaries).

jira_issues_comments:

1. I agree that this software should be made available on CRAN. Because of GPL dependencies we (the Arrow PMC) may have to leave the actual distribution of binaries for this software to the R community to do as it wishes. We can develop the tools to assist with that here
2. +100! I can't wait to use Arrow R API!
3. A few folks working on R and Arrow got together today, we agree that the R Arrow package should be in CRAN as soon as possible. However, this is likely to take a few months since we have to first: # Have an appropriate build process for Windows which Jeroen is leading, see [rwinlib/arrow|[https://github.com/rwinlib/arrow].] # Proper appveyor automation. The UrsaLabs team is planning to hire a build engineer to help, in the meantime, Kirill might be able to help get this going to make sure Jeroen's work does not regress as Arrow 0.13 gets developed. # Have an appropriate build process for OS X. # The Arrow libraries need to get distributed in the official Debian repository to be allowed in CRAN. Wes mentioned that, Kouhei (hoping I got this right), might be able to help the project navigate this in the next few months. So, while we

completely agree with you that is imperative that Arrow gets in CRAN, in the meantime, we could still make use of Arrow in OS X and Linux using the remotes package (or devtools) by installing the Arrow runtime first, followed by: {code:java} remotes::install_github("apache/arrow", subdir = "r") {code}

4. The Homebrew lead maintainer has dropped arrow from homebrew because it isn't building: https://github.com/Homebrew/homebrew-core/commit/6cd7f61d2d07d5fd9f8747ea0f620169cc8a2434 . So this further complicates things for R bindings on MacOS. I am guessing that the build broke because the 0.11 tarball has been removed from the apache website, even though 0.12 hasn't even been announced yet? [https://www.apache.org/dist/arrow/]
5. AFAIK no one from this community has been maintaining the Homebrew recipe. So that will obviously have to change so that updating the Homebrew recipe is part of the release management process. No one did anything wrong here I'm creating a sub-task about Homebrew and attaching it as a child of this issue EDIT: opened ARROW-4300
6. I'll keep attaching child issues to this issue that are required (or related, like setting up CI for Windows) as part of the testing and packaging process. If there is anything necessary that is not in the issue tracker already please create new issues
7. I don't see an issue for distributing into Debian/Fedora so I'll open one and link here, will also request help from dev@arrow.apache.org.
8. Let this be closed when the package is finally on CRAN; it can't really be resolved until after the next Apache release has taken place anyway
9. Issue resolved by pull request 4908 [https://github.com/apache/arrow/pull/4908]