Item 130
git_comments:

1. Writable column vectors of the result columnar batch.
2. Initialize the missing columns once.
3. * * Put a `HiveDecimalWritable` to a `WritableColumnVector`.
4. * * Returns the number of micros since epoch from an element of TimestampColumnVector.
5. * * Put `HiveDecimalWritable`s to a `WritableColumnVector`.
6. * * To support vectorization in WholeStageCodeGen, this reader returns ColumnarBatch. * After creating, `initialize` and `initBatch` should be called sequentially.
7. Record reader from ORC row batch.
8. * * Initialize ORC file reader and batch record reader. * Please note that `initBatch` is needed to be called after this.
9. ORC File Reader
10. * * The default size of batch. We use this value for both ORC and Spark consistently * because they have different default values like the following. * * - ORC's VectorizedRowBatch.DEFAULT_SIZE = 1024 * - Spark's ColumnarBatch.DEFAULT_BATCH_SIZE = 4 * 1024
11. * * The memory mode of the columnarBatch
12. * * The column IDs of the physical ORC file schema which are required by this reader. * -1 means this required column doesn't exist in the ORC file.
13. `selectedInUse` should be initialized with `false`.
14. Vectorized ORC Row Batch
15. * * Initialize columnar batch by setting required schema and partition information. * With this information, this creates ColumnarBatch with the full schema.
16. * Licensed to the Apache Software Foundation (ASF) under one or more * contributor license agreements. See the NOTICE file distributed with * this work for additional information regarding copyright ownership. * The ASF licenses this file to You under the Apache License, Version 2.0 * (the "License"); you may not use this file except in compliance with * the License. You may obtain a copy of the License at * * http://www.apache.org/licenses/LICENSE-2.0 * * Unless required by applicable law or agreed to in writing, software * distributed under the License is distributed on an "AS IS" BASIS, * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. * See the License for the specific language governing permissions and * limitations under the License.
17. * * Return true if there exists more data in the next batch. If exists, prepare the next batch * by copying from ORC VectorizedRowBatch columns to Spark ColumnarBatch columns.
18. The result columnar batch for vectorized execution by whole-stage codegen.

git_commits:

1. **summary:** [SPARK-16060][SQL] Support Vectorized ORC Reader
   message: [SPARK-16060][SQL] Support Vectorized ORC Reader ## What changes were proposed in this pull request? This PR adds an ORC columnar-batch reader to native `OrcFileFormat`. Since both Spark `ColumnarBatch` and ORC `RowBatch` are used together, it is faster than the current Spark implementation. This replaces the prior PR, #17924. Also, this PR adds `OrcReadBenchmark` to show the performance improvement. ## How was this patch tested? Pass the existing test cases. Author: Dongjoon Hyun <dongjoon@apache.org> Closes #19943 from dongjoon-hyun/SPARK-16060. (cherry picked from commit f44ba910f58083458e1133502e193a9d6f2bf766) Signed-off-by: Wenchen Fan <wenchen@databricks.com>

github_issues:

github_issues_comments:

github_pulls:

1. **title:** [SPARK-20682][SQL] Support a new faster ORC data source based on Apache ORC
   body: ## What changes were proposed in this pull request? Since [SPARK-2883] (https://issues.apache.org/jira/browse/SPARK-2883), Apache Spark supports Apache ORC inside `sql/hive` module with Hive dependency. This issue aims to add a new and faster ORC data source inside `sql/core` and to replace the old ORC data source eventually. In this issue, the latest [Apache ORC 1.4.0]

(https://orc.apache.org/news/2017/05/08/ORC-1.4.0/) (released yesterday) is used. There are four key benefits. - **Speed**: Use both Spark `ColumnarBatch` and ORC `RowBatch` together. This is faster than the current implementation in Spark. - **Stability**: Apache ORC 1.4.0 has many fixes and we can depend on ORC community more. - **Usability**: User can use `ORC` data sources without hive module, i.e, `-Phive`. - **Maintainability**: Reduce the Hive dependency and can remove old legacy code later. The followings are two examples of comparisons in `OrcReadBenchmark.scala`. ``` Java HotSpot(TM) 64-Bit Server VM 1.8.0_131-b11 on Mac OS X 10.12.4 Intel(R) Core(TM) i7-3615QM CPU @ 2.30GHz SQL Single Int Column Scan: Best/Avg Time(ms) Rate(M/s) Per Row(ns) Relative ----- ---------- SQL ORC Vectorized 170 / 194 92.5 10.8 1.0X SQL ORC MR 388 / 396 40.5 24.7 0.4X HIVE ORC MR 488 / 496 32.3 31.0 0.3X Partitioned Table: Best/Avg Time(ms) Rate(M/s) Per Row(ns) Relative ---------- SQL Read data column 188 / 227 83.6 12.0 1.0X SQL Read partition column 98 / 109 161.2 6.2 1.9X SQL Read both columns 193 / 227 81.5 12.3 1.0X HIVE Read data column 530 / 530 29.7 33.7 0.4X HIVE Read partition column 420 / 423 37.4 26.7 0.4X HIVE Read both columns 558 / 562 28.2 35.5 0.3X ``` ## How was this patch tested? Pass the Jenkins tests with newly added test suites in `sql/core`.

github_pulls_comments:

1. **[Test build #76693 has finished] (https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/76693/testReport)** for PR 17924 at commit [`8bfd4bb`] (https://github.com/apache/spark/commit/8bfd4bb4aec6dc3a5e7f41ec8d5c2a25f14ab87b). * This patch **fails to generate documentation**. * This patch merges cleanly. * This patch adds the following public classes _(experimental)_: * `class OrcColumnarBatchReader extends RecordReader[Void, ColumnarBatch] with Logging ` * `class OrcRecordIterator extends Iterator[InternalRow] with Logging `
2. **[Test build #76695 has finished] (https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/76695/testReport)** for PR 17924 at commit [`4607e0e`] (https://github.com/apache/spark/commit/4607e0e0e55abda3e9af8bc7812d2d1216a66076). * This patch **fails to generate documentation**. * This patch merges cleanly. * This patch adds no public classes.
3. Retest this please.
4. **[Test build #76699 has finished] (https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/76699/testReport)** for PR 17924 at commit [`4607e0e`] (https://github.com/apache/spark/commit/4607e0e0e55abda3e9af8bc7812d2d1216a66076). * This patch **fails to generate documentation**. * This patch merges cleanly. * This patch adds no public classes.
5. **[Test build #76705 has finished] (https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/76705/testReport)** for PR 17924 at commit [`85ef731`] (https://github.com/apache/spark/commit/85ef73134b7b7450e0689e138339433a30b92dea). * This patch passes all tests. * This patch merges cleanly. * This patch adds no public classes.
6. From the current benchmark, seems the performance has not obvious improvement, compared with the vectorized Hive ORC reader #13775. Maybe with more efficient batch approach as @cloud-fan suggested, it can perform better. Besides performance, getting rid of Hive dependency on ORC datasource is a great advantage for this.
7. @cloud-fan and @viirya . Shall we remove the vectorized part from this PR? - The non-vectorized ORCFileFormat is mandatory and also the performance is better than the current one. - After merging `sql/core` ORCFileFormat, many people (including @viirya and me) can work together in parallel. How do you think about that?
8. @dongjoon-hyun It is good for me. We can reduce the size of this PR too and mitigate review job.
9. Yep. Since this is an approach adding new dependency on Apache ORC, the non-vectorized PR also will need more supports(or approval) from the committers. I'll wait for more opinions at the current status for a while.
10. Hi, @rxin , @marmbrus , @hvanhovell , @gatorsmile , @sameeragarwal . Could you give us your opinions on this approach in Spark SQL part, too?
11. Hi, All. For further discussion and easy comparison, I made another PR (#17943) except `ColumnarBatch`.
12. Retest this please.

13. **[Test build #76920 has finished] (https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/76920/testReport)** for PR 17924 at commit [`85ef731`] (https://github.com/apache/spark/commit/85ef73134b7b7450e0689e138339433a30b92dea). * This patch passes all tests. * This patch merges cleanly. * This patch adds no public classes.
14. Retest this please.
15. **[Test build #80604 has finished] (https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/80604/testReport)** for PR 17924 at commit [`85ef731`] (https://github.com/apache/spark/commit/85ef73134b7b7450e0689e138339433a30b92dea). * This patch passes all tests. * This patch merges cleanly. * This patch adds no public classes.
16. @dongjoon-hyun I have a question: does this orc data sources reader support a table contains multiple file format for example: table/ day=2017-09-01 RCFile day=2017-09-02 ORCFile ParquetFileFormat doesn't support this feature.
17. Hi, I didn't try that, but that's not a concept of Spark data source table. Please don't expect that. :)
18. BTW, the latest version is maintained in #17980. Recently, Spark Vector format is changed.
19. Please refer the superset in #17980 .

github_pulls_reviews:

1. IIRC, @viirya also has a PR for vectorized orc reader. In that PR, we simply wrap the orc column batch to expose spark column batch interfaces, instead of writing orc column batch to spark column batch. I think that approach is more efficient.
2. Oh, thank you for the comment. It sounds efficient. I'll take a look.
3. More specially, we wrap Hive's `ColumnVector` in a batch to expose Spark's `ColumnVector` for constructing Spark's `ColumnarBatch`. So we don't need to move data from one vector format to another vector format.
4. Btw, the PR is at #13775.
5. Will we keep current Hive ORC datasource even this is in Spark?
6. We need to keep both versions before complete transition and for safety. Instead, we can make configurable which file format is used for `orc` data source string, e.g, `USING ORC`.
7. So to avoid datasource name conflict, we may change Hive ORC datasource's shortName to other than "orc".

jira_issues:

1. **summary:** Import code from Hive
   description:

jira_issues_comments:

1. We need to have Hive refactored correctly so that this can happen smoothly.
2. GitHub user omalley opened a pull request: https://github.com/apache/orc/pull/23 ORC-1 Import of ORC code from Hive. This patch pulls the current Java code for the ORC reader and writer out of Hive. Under the java directory there are three modules: * storage-api - a temporary copy of hive's storage api until we release hive with the changes we need * core - the core reader and writer for the vectorized reader and writer * mapreduce - an implementation of InputFormat and OutputFormat that uses core to read and write row by row You can merge this pull request into a Git repository by running: $ git pull https://github.com/omalley/orc orc-1 Alternatively you can review and apply these changes as the patch at: https://github.com/apache/orc/pull/23.patch To close this pull request, make a commit to your master/trunk branch with (at least) the following in the commit message: This closes #23 ---- ----
3. Github user nahguam commented on the pull request: https://github.com/apache/orc/pull/23#issuecomment-214777817 Hi, what's the best place for comments? The diff is too big to comment directly so I'll list a couple here: 1. OrcRecordReader.next L94 - seems to be a dead-end? 2. How is one to use OrcStruct as an end user? getFieldValue and setFieldValue are package private. Previously you'd access via the StructObjectInspector, but we seem to have no ObjectInspectors or StructFields. FileInputFormat.setInputPaths(conf, path); OrcInputFormat<OrcStruct> inputFormat = new OrcInputFormat<>(); InputSplit[] splits = inputFormat.getSplits(conf, 1); RecordReader<NullWritable, OrcStruct> recordReader = inputFormat.getRecordReader(splits[0], conf,

null); NullWritable key = recordReader.createKey(); OrcStruct value = recordReader.createValue(); while (recordReader.next(key, value)) { // How do I interrogate value for it's fields' values? } recordReader.close(); 3. Perhaps for another ticket, but it would be nice to have a mechanism to access a struct's fields by name as well as by index. 4. Is there any particular reason that the value generic type is V extends Writable instead of OrcStruct?
4. Github user omalley commented on the pull request: https://github.com/apache/orc/pull/23#issuecomment-214806455 1. You're right that you can't actually pass null in to value. *smile* 2. I made the accessors public. 3. I added getters and setters that take the field name. 4. The input format doesn't need to have a struct as the root type. Look at TestOrcOutputFormat.testLongRoot for an example.
5. Github user nahguam commented on the pull request: https://github.com/apache/orc/pull/23#issuecomment-215073080 Excellent, Thanks! I've just been going over all the types and have a few more: 1. `VectorizedRowBatch.toUTF8` appears to be unused 2. `OrcMap` & `OrcList` - please could the constructors be public? 3. `OrcList` - please could we have another constructor so we can pass in the initialCapacity as per `ArrayList`? 4. `OrcUnion` - please could the class itself and the `set` method be public? I'm just looking at the Formats/RecordReader/RecordWriter now.
6. Github user omalley commented on the pull request: https://github.com/apache/orc/pull/23#issuecomment-215112415 1. The storage-api is a clone of Hive's until they release a version that has the bits that we need. So the method VectorizedRowBatch.toUTF8 is used by Hive. 2. Done 3. Done. 4. Done. I also moved TestOrcOutputFormat to a different package so that it can only access the public API, which should prevent similar problems. Thanks for your reviews.
7. The patch is the entire patch. The diff file is the diff between Hive's trunk + HIVE-11417 orc module against this patch's java/core. The only difference is the addition of the TypeDescription parser.
8. +1 for the patch. Assuming storage-api module will be removed from orc after next release of hive.
9. I just committed this. Thanks for the reviews nahguam and prasanthj!