Item 295
git_comments:

1. * * Register callback and deduplicate it if any. * @param callBackName the name of callback. It should be identical. * @param callBack the callback passed in from upper layer, such as Hive.
2. * * Get the internal scale value of MemoryManger * @return
3. * * Get the registered callbacks. * @return
4. we do not really want to start a new thread here.
5. Verify Callback mechanism

git_commits:

1. **summary:** PARQUET-164: Add a counter and increment when parquet memory manager kicks in
   message: PARQUET-164: Add a counter and increment when parquet memory manager kicks in Add a counter for writers, and increment it when memory manager scaling down row group size. Hive could use this counter to warn users. Author: dongche1 <dong1.chen@intel.com> Author: dongche <dong1.chen@intel.com> Author: root <root@bdpe15.sh.intel.com> Closes #120 from dongche/PARQUET-164 and squashes the following commits: 9bcb1ba [dongche] Remove stats, and change returned callbacks map unmodifiable 3cbbeb9 [dongche] Merge remote branch 'upstream1/master' into PARQUET-164 bdef233 [dongche] Merge remote branch 'upstream1/master' into PARQUET-164 780be6ed [root] revert change about callable and address comments 11f9163 [dongche1] Merge remote branch 'upstream/master' into PARQUET-164 55549a5 [dongche1] Use callable and strict registerScallCallBack method. 74054aa [dongche1] Use Runnable as a generic callback 8782a02 [dongche1] Add a callback mechanism instead of shims. And rebase trunk b138b7f [dongche1] Merge remote branch 'upstream/master' into PARQUET-164 93a4678 [dongche1] PARQUET-164: Add a counter and increment when parquet memory manager kicks in

github_issues:

github_issues_comments:

github_pulls:

1. **title:** PARQUET-164: Add a counter and increment when parquet memory manager kicks in
   body: Add a counter for writers, and increment it when memory manager scaling down row group size. Hive could use this counter to warn users.

github_pulls_comments:

1. @dongche, rather than adding more shims what do you think about adding a callback mechanism? That would allow Hive or other systems to be notified and would be overall less code for both Parquet and Hive to maintain.
2. @rdblue , thanks for your review! Callback mechanism is a good idea. Not sure MapReduce has api supporting this, it seems only a job completed callback there. Then if we need to implement a callback mechanism by ourselves in Parquet, I think it should be a remote callback, since generating event code in Parquet executes in MapReduce tasks, and it has to notify Hive which run as a MR client. Not sure my idea is ok... Any suggestions? This patch use polling. After submitting jobs, Hive/other systems could use MR api to get counters of jobs from JobTracker, and check the "blockDownsize" counter monitored by Parquet memory manager in MR tasks.
3. I think a callback mechanism should be independent of the execution environment. For a given process, you should register callbacks with a memory manager and Parquet shouldn't provide anything else. To get this working in MR, you'd register the callback at the appropriate time in your OutputFormat. I'd say add the logic to [`getHiveRecordWriter`](https://github.com/apache/hive/blob/trunk/ql/src/java/org/apache/hadoop/hive/ql/io/parquet/MapredParquetOutputFormat.java#L85) in this case. This would need to handle ensuring only one counter callback is registered, so the callback API should probably provide named callbacks and a way to query whether a particular callback name is present.
4. **body:** Actually, the callback deduplication could easily be done in the MR OutputFormat too, so I could go either way. It wouldn't be a bad thing to be able to check what callbacks are already registered.
   label: code-design
5. **body:** Really clear! Thanks! I will update the patch later. One more thinking: It is fine in local mode. But if distributed, say Hive submits a job to write Parquet data, the instance of MemoryManger and the registered instance of Callback are both in a map (or reduce) task process. The HS2 of Hive, which needs callback to warn users, is in another process. In this distributed case, maybe a RPC is needed. How does this sound? Please correct me if I misunderstand the problem :)
   label: code-design
6. > The HS2 of Hive, which needs callback to warn users, is in another process. In this distributed case, maybe a RPC is needed. @brockn can correct me if I'm wrong, but I think the plan was to add the counter support in order to avoid RPC. Hive can monitor the counter as the job runs and give the user feedback as needed.
7. Hi @rdblue, sorry for the late response. I took vacation several days ago, and worked on Hive + Parquet vectorization after vacation... A new patch is updated, using the callback mechanism instead of add more shims. Could you please help to review it? Thanks
8. Hi, @rdblue , I like the suggestion of using a Runnable as a generic callback. Thanks! Patch updated.
9. **body:** Thanks @dongche! This looks good to me, although we might want to open up some of the MemoryManager methods for inspection in the future, so the callback can inspect the current scale value. That doesn't need to be part of this though. I'll commit this soon, but I'll leave it open for a little bit so @isnotinvain and @julienledem to have a chance to comment.
   label: code-design
10. I'd vote for making those two minor changes in this PR if you can. The callable could be: ``` public static class MemoryManagerStats { public final double scale; // in the future we could add more, w/o breaking our API } ``` and use a `Callable<MemoryManagerStats>` I think the `registerScaleCallBack` being strict about duplicates is a quick change and will catch mistake earlier rather than later.

11. Thanks for your review! @isnotinvain Code updated. Could you take a look again? Thanks.
12. @isnotinvain: did you want the `MemoryManagerStats` to be passed to the `Callable` or returned by it? I'd say it would be valuable to pass the stats into the callback, but `Callable` doesn't allow that, only returning something (something that the manager already as access to). I also wasn't sure if it was worth the trouble to create a special callback class with a scale argument, since the `MemoryManager` is a singleton. The callback can always inspect it without being handed data, though you might want to add accessors for scale and total allocation.
13. Hi @isnotinvain , @rdblue , are there any changes I could do for this patch? BTW, the Travis CI failed and seems not related with code changes (test passed locally). So I wanted to retry again. Is there a way to trigger Travis test without pushing code change? Thanks. It seems Travis is only triggered when submit or updated code in PR...
14. **body:** @dongche, I think you want to revert the change to use `Callable` instead of `Runnable` as I mentioned above. `Callable` doesn't allow you to pass objects, so you have to make the `MemoryManager` information public. Using a `Callable` to return it isn't very valuable, it would only be valuable to pass it to the callback, but you'd need to write a callback interface for that and I don't think that is necessary.
    label: code-design
15. Sorry, @rdblue is right about Callable, I forgot it's the return value. Runnable is fine, or an interface for the inverse of Callable would be fine too.
16. Thanks for the feedback. @rdblue @isnotinvain Patch updated.
17. LGTM, +1, though I'm not sure if we're still waiting for the 1.6.0 release before merging to master? @rdblue ?
18. @isnotinvain: yeah, I'd prefer not to merge anything until we do the rename and 1.7.0 branching
19. @dongche, the package rename has gone in and I'd like to get this merged now. could you rebase it on the current master? Thanks!
20. Sure. Rebased with current master at https://github.com/apache/parquet-mr
21. Address comments.

github_pulls_reviews:

1. **body:** I think it would be better to have a generic callback that is used for counters in Hive, rather than having an API here that is focused around counters. Ideally, the Callback would just be a Runnable. Naming should happen when the callback is registered: ``` java final Counter counter = ...; MemoryManager.registerScaleCallback("increment-hive-counter", new Runnable() { public void run() { counter.increment(1); } }); ```
   label: code-design
2. I think it makes sense to change Hive, but Parquet doesn't really need to count the number of scale-down events.
3. Do you want to use a `Callable<MemorManager>` or `Callable<MemoryManagerInfo>` or something like that so the callable can be passed things like the scale value etc.?
4. how about making this function void, using preconditions to assert that callBackName and callBack are not null, and throwing when two callbacks are assigned the same name?
5. Yes, it is good for callback to be able to get values of MemoryManager. Using `Callable<T>` is ok. Besides that, I was thinking public a method like `getCurrentScale` in MemoryManger, so that cackback could get wanted value in its implementation through MemoryManger instance. How does this sound? If this way is also ok, shall we work it in a follow on PR?
6. Nice! Thanks @isnotinvain I will update the code later.
7. catch `RuntimeException` not `Exception` `Exception` includes `InterruptedException` and other things you shouldn't catch by accident.
8. would we rather it be fatal?
9. please use `Preconditions.checkNotNull`
10. Could this return an UnmodifiableMap instead of the one used by the memory manager itself?
11. Now that we aren't passing stats to the callback, what is the value of this class? Why not just add a `getScale` method instead of a `getStats` method?

jira_issues:

1. **summary:** Warn when parquet memory manager kicks in
   description: In PARQUET-108 we implemented a memory manager for parquet. I think we should warn in the close() method if we had to adjust the memory down so that users know their memory is being adjusted down and block size will be less than expected.

jira_issues_comments:

1. Thinking about this more, we should increment a counter every time this kicks in. This will allow us to warn the user on the Hive side.
2. Implemented in [PR #119|https://github.com/apache/incubator-parquet-mr/pull/119].
3. **body:** We'll have to do the counter idea in a separate issue. I'd like to get the warning in before too many people start hitting this limit without knowing it.
   label: code-design
4. Merged PR #119.
5. **body:** I write a patch about the counter. Will do some cleanup and upload it tomorrow.
   label: code-design
6. Upload patch about the counter in https://github.com/apache/incubator-parquet-mr/pull/120
7. Merged #120. Thanks, Dong!