Enhancements to Spark’s Extension Points API

Apache Spark has an extension points API to allow third party users to customize Spark in their deployments. In my previous blog (<https://developer.ibm.com/code/2017/11/30/learn-extension-points-apache-spark-extend-spark-catalyst-optimizer/>), I talked about the extension points and how to add an optimizer rule using this API. The extension points API is great and allows us to extend Spark in different ways for optimizations, for adding hooks to enable some functionality in our application. In this blog, I will talk about some enhancements that we developed for the Extension Points API.

Motivation:

When we use the SparkSessionExtensions API to add new optimizer rules, the rules are added in a specific place in the Optimizer. For most cases, this works great! The current API does not allow fine grained control on when the optimization rule will be exercised. In some cases, we want to add a rule in a specific place in the optimizer. Currently, the API does not give us that flexibility. This is what motivated us to enhance the API.

Enhancements:

There are two new API enhancements added to SparkSessionExtensions.

1. Inject a rule after or before an existing rule in a given existing batch in the

Optimizer.

*/\*\*  
 \* Inject an optimizer* `*Rule*` *builder into the* [[*SparkSession*]] *in a particular batch after  
 \* or before a specific existing rule in the batch.  
 \* If the batchName does not exist, or if the existing rule does not exist in the given  
 \* batch, then an error will be thrown. (fail fast)  
 \*/***def** injectOptimizerRuleInOrder(  
 builder: RuleBuilder,  
 batchName: String,  
 ruleOrder: Order.Order,  
 existingRule: String): Unit

This method can be used to inject a rule to an existing batch specified by the ‘batchName’ and you can specify whether the new rule is added before or after an existing rule name.

2. Inject a optimizer batch after or before a existing batch

We cannot currently add a batch to the optimization using the

SparkSessionExtensions API. So, this API adds support to add a batch of rules.

*/\*\*  
 \* Inject a batch of optimizer rules  
 \** ***@param batchName*** *- Batch Name to inject  
 \** ***@param maxIterations*** *- Iterations  
 \** ***@param existingBatchName*** *- Existing batch name in reference to which this batch is injected  
 \** ***@param order*** *- Specify the order, if before the existing batch or after the existing batch  
 \** ***@param rules*** *- New rules in the batch that will be added  
 \*/***def** injectOptimizerBatch(  
 batchName: String,  
 maxIterations: Int,  
 existingBatchName: String,  
 order: Order.Value,  
 rules: Seq[RuleBuilder]): Unit

This method can be used to add a batch of rules in a specific order with reference to an existing batch. You can specify if you want to add the batch before or after an existing batch name.

Code:

I have opened SPARK-26249 (<https://issues.apache.org/jira/browse/SPARK-26249>) for this functionality. The JIRA contains the design document and details on the expected behavior of this API. Please check out the code changes in this PR <https://github.com/apache/spark/pull/23206> if you are interested in trying out this functionality for your usecase. I would appreciate any feedback on the SPARK-26249 if it is useful for your scenarios.

I gave a presentation at the Apache Spark and AI Summit at San Francisco 2019 on how to extend Spark with customized optimizations. <https://www.youtube.com/watch?v=IlovS-Y7KUk>

The talk covers the extension points API in detail and also talks about the proposed enhancements that I cover in this blog. Please feel free to check it out if you are interested.