Item 192
git_comments:

git_commits:

1. **summary:** [scala] Change ScalaAggregateOperator to use TypeSerializer
   message: [scala] Change ScalaAggregateOperator to use TypeSerializer This closes #263

github_issues:

github_issues_comments:

github_pulls:

1. **title:** [scala] Change ScalaAggregateOperator to use TypeSerializer
   body: Before it was using TypeSerializerFactory. This led to problems with the user code class loader and is not necessary.

github_pulls_comments:

1. I moved the cast and added a Validate.* check.
2. +1 LGTM
3. Looks good, will merge...

github_pulls_reviews:

1. Could we do explicit conversion from TypeSerializer to TupleSerializerBase instead of hard casting?
2. What do you mean by explicit conversion?
3. Ah sorry, I meant getIntputType call returns TypeSerializer and the AggregatingUdf constructor is taking TupleSerializerBase so could the constructor parameter takes TypeSerializer instead?
4. We could maybe. If we enhanced TypeSerializer to have a method createInstance(...) that can create an instance with the given field values. This is only possible for a small subset of the types that we support: The Java Tuple Types, since we control their constructors. And Scala Case Classes, since we assume that the fields are exactly the constructor arguments, as is customary for Case Classes. We would then need to add createInstance(...) methods to all other TypeSerializers that throw an Exception telling the user that instance creation with field values is not supported.
5. At least for now, could we change the signature of the constructor to TypeSerializer and add validate for the argument if it could be cast to TupleSerializerBase? So add check: Validate.isTrue(serializer instanceof TupleSerializerBase); Casting without proper check makes me nervous.
6. Ok, will do.

jira_issues:

jira_issues_comments: