Item 50
git_comments:

1. Package coderx contains coders for primitive types that aren't included in the beam model.
2. go:generate go install github.com/apache/beam/sdks/go/cmd/starcgen go:generate starcgen --package=coderx --identifiers=encString,decString,encUint32,decUint32,encInt32,decInt32,encUint64,decUint64,encInt64,decInt64,encVarIntZ,decVarIntZ,encVarUintZ,decVarUintZ,e
3. 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.
4. NewString returns a coder for the string type. It uses the native []byte to string conversion.
5. 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.
6. 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.
7. TestMain invokes ptest.Main to allow running these tests on non-direct runners.
8. 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.
9. **comment:** FixedReStream is a simple in-memory ReStream. Open returns the a Stream from the start of the in-memory ReStream. Close releases the buffer, closing the stream. Arguably this should be: reflect.ValueOf(v).Convert(to).Interface() but this isn't desirable as it would add avoidable overhead to functions where it applies. A user will have better performance by explicitly doing the type conversion in their code, which the error will indicate. Slow Magic vs Fast & Explicit.
   label: code-design

git_commits:

1. **summary:** [BEAM-3580] Use a custom coder for strings.
   message: [BEAM-3580] Use a custom coder for strings.

github_issues:

github_issues_comments:

github_pulls:

github_pulls_comments:

github_pulls_reviews:

jira_issues:

1. **summary:** Do not use Go BytesCoder to encode string
   description: We should not use the same built-in coder for two different types. It creates the need for conversions at inopportune times in the runtime. One option would be to a custom coder that shares encoding with bytes, given that bytes are length prefixed.
2. **summary:** Do not use Go BytesCoder to encode string
   description: We should not use the same built-in coder for two different types. It creates the need for conversions at inopportune times in the runtime. One option would be to a custom coder that shares encoding with bytes, given that bytes are length prefixed.
   label: code-design
3. **summary:** Do not use Go BytesCoder to encode string
   description: We should not use the same built-in coder for two different types. It creates the need for conversions at inopportune times in the runtime. One option would be to a custom coder that shares encoding with bytes, given that bytes are length prefixed.
4. **summary:** Do not use Go BytesCoder to encode string
   description: We should not use the same built-in coder for two different types. It creates the need for conversions at inopportune times in the runtime. One option would be to a custom coder that shares encoding with bytes, given that bytes are length prefixed.
5. **summary:** Do not use Go BytesCoder to encode string
   description: We should not use the same built-in coder for two different types. It creates the need for conversions at inopportune times in the runtime. One option would be to a custom coder that shares encoding with bytes, given that bytes are length prefixed.
   label: code-design
6. **summary:** Do not use Go BytesCoder to encode string
   description: We should not use the same built-in coder for two different types. It creates the need for conversions at inopportune times in the runtime. One option would be to a custom coder that shares encoding with bytes, given that bytes are length prefixed.
7. **summary:** Do not use Go BytesCoder to encode string
   description: We should not use the same built-in coder for two different types. It creates the need for conversions at inopportune times in the runtime. One option would be to a custom coder that shares encoding with bytes, given that bytes are length prefixed.

jira_issues_comments:

1. Seems that this breaks beam.Combine(s, combineFn, PCollection<string>) in certain cases. When entering Combine.addInput(), the value is of type []byte. It's then converted to the specified input type, which is supposed to be "string" [1]. But that type could be a universal type, e.g. when we define: "AddInput(a accum, val beam.T) accum". In that case, the conversion is a no-op, and we call AddInput() with []byte, causing a type mismatch. Using a separate coder that encodes string -> []byte, and decodes []byte -> string should solve this problem. I will try that and see if it works. This also makes me thinking whether we should substitute universal types in CombineFn to concrete ones before/at runtime?
   [1] [https://github.com/apache/beam/blob/master/sdks/go/pkg/beam/core/runtime/exec/combine.go#L219]
2. **body:** You're right though, a dedicated custom coder for strings would resolve this. In principle this would also be solvable through the User Defined Coders work. An alternative work around would be to do the converters, but those are likely unnecessary altogether if there was a dedicated coder for strings which would avoid using the converts on all the paths. That can be fixed by adding conversions prior to the invocation of the combine operations which isn't currently done via the invoker in all instances (such as the binary merge fast path), See here:
   [https://github.com/apache/beam/blob/master/sdks/go/pkg/beam/core/runtime/exec/fn.go#L132] Note: I'm likely going to be changing the combines soon so the code generator will be able to speed up combines for the AddInput and MergeAccumulators cases.

label: code-design

3. I'm in the process of fixing Top, and I ran into this, so I'm going to take this unless [~htyleo] has any objections. You'll get the review however once it's verified and tested properly
4. Sounds good. Sorry that I didn't get the time to fix this after your previous review.
5. **body:** There may be one or two no longer necessary []byte -> string conversions and vice versa that I missed, but they can be cleaned up as they're found.
   label: code-design