State of the Go SDK 2022

Robert Burke
@lostluck
Previously, in Beam Go
Previously, in the Go SDK

- Handles Batch Basics
  - Global Windowing
  - ParDos
  - Iterable Side Inputs
  - Flatten
  - CoGBKs
  - CombineFns w/Lifting
  - Partition
  - User Metrics
  - Coders
    - Standard
    - Custom Go

- Announced in 2020
  - Cross Language Transforms
  - Bounded Splittable DoFns
  - Loopback mode
  - Katas
  - State Backed Iterables
  - Reshuffle
What’s missing?

TODO(you?)

- Map Side Inputs
- Advanced Windowing
- State and Timers
- DoFn Checkpointing
- Native SDF IOs
- Cross Language Wrappers
- Go SDK Expansion Service
- Testing Audit
- Go Generics
- …?
What’s new?

- Exit Experimental
  - Go Module Support
  - Default Beam Schema Coding
- Pipeline Results
- Map Side Inputs
- Advanced Windowing & Triggers
- Native Streaming Support
- New Cross Language Wrappers
- Testing Audit
- Worker Status
- Go Generics for Performance
- Load Tests
- Dataflow Support

Summer 2022
Go SDK
Exited
Experimental

- November 2021
- https://beam.apache.org/blog/go-sdk-release/
- Bounded Splittable DoFns
- Beam Programming Guide
- Default Schema Coding
- Go Module Support
Beam Programming Guide

https://beam.apache.org/documentation/programming-guide/

4.2.1.1. Applying ParDo

beam.ParDo applies the passed in DoFn argument to the input PCollection, as

// ComputeWordLengthFn is the DoFn to perform on each element.  
// ProcessElement is the method to execute for each element.

func (fn *ComputeWordLengthFn) ProcessElement(word string) { ... }

7. Data encoding and type safety

When Beam runners execute your pipeline, they often need to materialize the int elements to and from byte strings. The Beam SDKs use objects called Coder at

Note that coders are unrelated to parsing or formatting data when interacting with Beam. Formatting should typically be done explicitly, using transforms such as Partial

Standard Go types like int, int64, float64, []byte, and string and more are available. The coders are coders at

Note that coders do not necessarily have a 1:1 relationship with types. For example, the input and output data can use different Integer coders. A transform might have and Integer-coded output data that uses VarIntCoder.
Default Schema Coding

- Schemas are automatically inferred for user defined structs.
- Uses Beam Schema Row Encoding by default
- Compact Binary format is significantly more efficient than using JSON, the previous by default.
- Similar restrictions: fields must be Exported to be encoded
- No dynamic Row type at this time.
Go Module Support

import "github.com/apache/beam/sdks/v2/go/pkg/beam"

- go.mod rooted in sdks/ folder.
- v2.40.0 has a minimum language version of go 1.18
  - Doesn’t prevent users from adopting newer Go versions themselves
- Ensures SDK developers, SDK Users, and Repo Test automation all use the same dependency versions
Do More

- Triggers
- Cross Language Transforms
- Native Streaming
- Map Side Inputs
- Pipeline Results
Triggers

- Configure Aggregation Behavior
- Combines with Fixed, Sliding, and Session interval windowing strategies
- Use `beam.WindowInto` to define how your data is aggregated in EventTime
- Produce partial results
- How and when to handle late data

See
https://beam.apache.org/documentation/programming-guide/#triggers
Cross Language Transforms

- Found in beam/io/xlang/...
  - KafkaIO
  - BigqueryIO
  - JDBCIO
  - BeamSQL
  - DebeziumIO
- Automatic Java Expansion Service Startup

Learn more at
https://beam.apache.org/documentation/programming-guide/#multi-language-pipelines
Native Streaming

Author Unbounded Splittable DoFns in Go

- Self Checkpointing with Process Continuations
- Unbounded Restrictions for Splitting
- Watermark Estimation
- Bundle Finalization
- Custom Drain Trucation
Native Streaming

Writing a native Go streaming pipeline
Tuesday 16:15-16:40 CDT, Room 203
with Danny McCormick and Jack McCluskey
https://2022.beamsummit.org/sessions/native-go-pipeline/
func queryMetrics(pr beam.PipelineResult, ns, n string) metrics.QueryResults {
    return pr.Metrics().Query(func(r beam.MetricResult) bool {
        return r.Namespace() == ns && r.Name() == n
    })
}
Map Side Inputs

```go
ProcessElement(..., lookup func(K) func(*V) bool,...){
...
vals := lookup(key)
var val V
for vals(&val) { ... }
```
Performance

- Side Input Cache
- Generic Registration
- Load Tests!
Cross Bundle Side Input Cache

Caches inputs on the SDK worker side for cross bundle access.

Enable* with the `harnessopts` package:

```go
import "github.com/apache/beam/sdks/v2/go/pkg/beam/util/harnessopts"

harnessopts.SideInputCacheCapacity(keyCount)
```

*Requires runner state cache support
Generics for Performance

```go
import "github.com/apache/beam/sdks/v2/go/pkg/beam/register"

func init() {
    register.DoFn2x1[KeyType,ValueType,ReturnType](&myDoFn{})
    register.Function4x0(simpleDoFn4x0)
}
```
Load Tests!

Covers various Batch ParDo, GBK, CoGBK, Combine, and Side Inputs patterns.

To see them select the suite and then Batch and Go at metrics.beam.apache.org
Dataflow Support
Thank you for contributing!
What's next?

TODO(you?)

- State and Timers
  - GroupIntoBatches
- More
  - Native SDF IOs
  - Cross Language Wrappers
- Go SDK Expansion Service
  - Dynamic Row type
- Go
  - Faster!
  - Generics!
Related Talks

Oops, I wrote a Portable Beam Runner in Go
Tuesday 12:00-12:25 CDT, Room 203
with Robert Burke

Writing a native Go streaming pipeline
Tuesday 16:15-16:40 CDT, Room 203
with Danny McCormick and Jack McCluskey
https://2022.beamsummit.org/sessions/native-go-pipeline/
State of the Go SDK 2022

Robert Burke
@lostluck
Reference Links

Experimental Exit

Go v2.40