Combine by Example

OpenTelemetry Exponential Histograms

By Alex Van Boxel

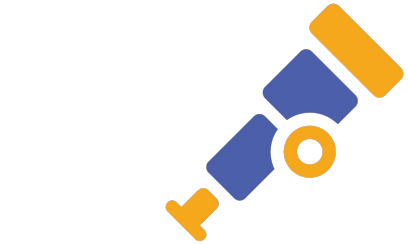
BEAM SUMMIT

Austin, 2022

Combine are just more Fn

OpenTelemetry

BEAM SUMMIT Austin, 2022



An observability framework for cloud-native software.

OpenTelemetry is a collection of tools, APIs, and SDKs. You use it to instrument, generate, collect, and export telemetry data (metrics, logs, and traces) for analysis in order to understand your software's performance and behavior.

Exponential

Histogram, what is it?

The magic of scale

Power of math

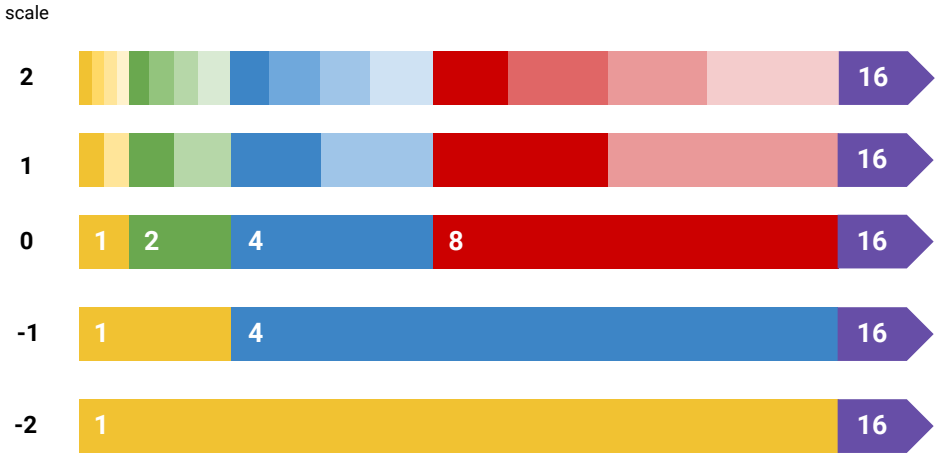
base = (2^(2^-scale))

(base^index) =< value < (base^(index+1))

Scale

scale	-2	-1	0	1	2	3	4	5
base	**16**	**4**	**2**	**1.414213562**	**1.189207115**	**1.090507733**	**1.044273782**	**1.021897149**
0	1	1	1	1	1	1	1	1
1	16	4	2	1.414213562	1.189207115	1.090507733	1.044273782	1.021897149
2	256	16	4	2	1.414213562	1.189207115	1.090507733	1.044273782
3	4096	64	8	2.828427125	1.681792831	1.296839555	1.138788635	1.067140401
4	65536	256	16	4	2	1.414213562	1.189207115	1.090507733
5	1048576	1024	32	5.656854249	2.37841423	1.542210825	1.241857812	1.114386743
6	16777216	4096	64	8	2.828427125	1.681792831	1.296839555	1.138788635
7	268435456	16384	128	11.3137085	3.363585661	1.834008086	1.354255547	1.163724859
8	4294967296	65536	256	16	4	2	1.414213562	1.189207115
9	68719476736	262144	512	22.627417	4.75682846	2.181015465	1.476826146	1.21524736
10	1099511627776	1048576	1024	32	5.656854249	2.37841423	1.542210825	1.241857812
11	1759218604441	4194304	2048	45.254834	6.727171322	2.593679109	1.610490332	1.269050957
12	2814749767106	16777216	4096	64	8	2.828427125	1.681792831	1.296839555
13	4.5036E+15	67108864	8192	90.50966799	9.51365692	3.084421651	1.75625216	1.325236643
14	7.20576E+16	268435456	16384	128	11.3137085	3.363585661	1.834008086	1.354255547
15	1.15292E+18	1073741824	32768	181.019336	13.45434264	3.668016173	1.915206561	1.383909882
16	1.84467E+19	4294967296	65536	256	16	4	2	1.414213562
17	2.95148E+20	17179869184	131072	362.038672	19.02731384	4.362030931	2.088547565	1.445180807

Scale



Protobuf

The OpenTelemetry Exponential Histogram

A deep dive into the Exponential Histograms original Protobuf

OpenTelemetry Metric Protobuf

```
// ExponentialHistogramDataPoint is a single data point in a timeseries that describes the
// time-varying values of a ExponentialHistogram of double values. A ExponentialHistogram contains
// summary statistics for a population of values, it may optionally contain the
// distribution of those values across a set of buckets.

message ExponentialHistogramDataPoint {

  // The set of key/value pairs that uniquely identify the timeseries from
  // where this point belongs. The list may be empty (may contain 0 elements).
  repeated opentelemetry.proto.common.v1.KeyValue attributes = 1;
```

OpenTelemetry Metric Protobuf

```
  fixed64 start_time_unix_nano = 2;    // StartTimeUnixNano is optional but strongly encouraged.

  fixed64 time_unix_nano = 3;    // TimeUnixNano is required.

  fixed64 count = 4;   // count is the number of values in the population.

  optional double sum = 5;    // sum of the values in the population. If count is zero then this field
  optional double min = 12;   // min is the minimum value over (start_time, end_time].
  optional double max = 13;   // max is the maximum value over (start_time, end_time].
}
```

OpenTelemetry Metric Protobuf

```
// scale describes the resolution of the histogram.
sint32 scale = 6;

// zero_count is the count of values that are either exactly zero or
fixed64 zero_count = 7;

// positive carries the positive range of exponential bucket counts.
Buckets positive = 8;

// negative carries the negative range of exponential bucket counts.
Buckets negative = 9;
```

OpenTelemetry Metric Protobuf

```
// Buckets are a set of bucket counts, encoded in a contiguous array
// of counts.
message Buckets {

  // Offset is the bucket index of the first entry in the bucket_counts array.
  sint32 offset = 1;   // Note: This uses a varint encoding as a simple form of compression.

  // Count is an array of counts, where count[i] carries the count of the bucket
  // at index (offset+i).  count[i] is the count of
  // values greater than or equal to base^(offset+i) and less than base^(offset+i+1).
  // This field is expected to have many buckets, especially zeros, so uint64 has been selected to ensure
  // varint encoding.
  repeated uint64 bucket_counts = 2;
}
```

Design Decisions

Implementing

CombineFn

Implementing a CombineFn

```
class CombineFn<InputT, AccumT, OutputT> {

    // Adds the given input value to the given accumulator, returning the new accumulator value.
    public abstract AccumT addInput(AccumT mutableAccumulator, InputT input);

    // Returns an accumulator representing the accumulation of all the input values accumulated in
    // the merging accumulators. (only the first accumulator can be modified)
    public abstract AccumT mergeAccumulators(Iterable<AccumT> accumulators);

    // Returns the output value that is the result of combining all the input values represented by
    // the given accumulator. (accumulator can be modified for efficiency)
    public abstract OutputT extractOutput(AccumT accumulator);
```

Implementing a CombineFn

```
class CombineFn<InputT, AccumT, OutputT> {

    // Adds the given input value to the given accumulator, returning the new accumulator value.
    public abstract AccumT addInput(AccumT mutableAccumulator, InputT input);

    // Returns an accumulator representing the accumulation of all the input values accumulated in
    // the merging accumulators. (only the first accumulator can be modified)
    public abstract AccumT mergeAccumulators(Iterable<AccumT> accumulators);

    // Returns the output value that is the result of combining all the input values represented by
    // the given accumulator. (accumulator can be modified for efficiency)
    public abstract OutputT extractOutput(AccumT accumulator);
```

Implementing a CombineFn

```
class CombineFn<InputT, AccumT, OutputT> {

    // Adds the given input value to the given accumulator, returning the new accumulator value.
    public abstract AccumT addInput(AccumT mutableAccumulator, InputT input);

    // Returns an accumulator representing the accumulation of all the input values accumulated in
    // the merging accumulators. (only the first accumulator can be modified)
    public abstract AccumT mergeAccumulators(Iterable<AccumT> accumulators);

    // Returns the output value that is the result of combining all the input values represented by
    // the given accumulator. (accumulator can be modified for efficiency)
    public abstract OutputT extractOutput(AccumT accumulator);
```

Design Choice

No Proto as the accumulator

Why not to use the pure protobuf as an accumulator?

Mutable

```
/** Mutable Datapoint Accumulator for Exponential Histograms */
public final class ExponentialHistogramDatapointAccum implementents Serializable {

      private final int scale;

      private final double base;

      ArrayList<Long> positiveBucket;

      long zeroBucket = 0;

      double sum = 0;
      long count = 0;
```

Design Choice

Output is an accumulator

Why is the Protobuf not the output?

Accum out

```
public abstract class ExponentialHistogramCombineFn<T>
            extends Combine.CombineFn<
                        T, ExponentialHistogramDatapointAccum, ExponentialHistogramDatapointAccum> {

      @Override
      public ExponentialHistogramDatapointAccum extractOutput(
                  ExponentialHistogramDatapointAccum accumulator) {
            return accumulator;
      }
}
```

Design Choice

Separate Accumulator to Proto **DoFn**

What about time?

Need for time

```
@ProcessElement
public void exponentialHistogramDataPointAccum2MetricFn(
            ProcessContext context, BoundedWindow window) {

      ExponentialHistogramDataPoint.Builder pbDataPointBuilder =
                  MetricUtil.add1MinuteBounds(
                              ExponentialHistogramDataPoint.newBuilder(),
                                    window.maxTimestamp().getMillis());
```

Design Choice

Put most of the logic in the **accumulator**

Accumulators are mutable anyway

Most of the logic in the Accum

```
public void addValue(double value) {
      if (value < 1.0) {
            increaseZeroCount(1);
      } else {
            increaseIndexCount(indexOf(value), 1);
      }
      count++;
      sum += value;
}

private int indexOf(double value) {
      Preconditions.checkArgument(value >= 1.0, "indexOf must not be called for values < 1.0");
      return (int) (Math.log(value) / Math.log(base));
}
```

Keep Combine simple

```
@Override
public ExponentialHistogramDatapointAccum mergeAccumulators(
      Iterable<ExponentialHistogramDatapointAccum> accumulators) {
      // Search for the lowest scale, so no unnecessary compression needs to happen mid-calculation
      int scale = preferredScale;
      for (ExponentialHistogramDatapointAccum accumulator : accumulators) {
            scale = Math.min(scale, accumulator.getScale());
      }
      ExponentialHistogramDatapointAccum mutable = new ExponentialHistogramDatapointAccum(scale);
      // Add all histograms using the add function in the accumulators
      for (ExponentialHistogramDatapointAccum accumulator : accumulators) {
            mutable.add(accumulator);
      }
      return mutable;
}
```

Keep Combine simple

```
@Override
public ExponentialHistogramDatapointAccum mergeAccumulators(
      Iterable<ExponentialHistogramDatapointAccum> accumulators) {
      // Search for the lowest scale, so no unnecessary compression needs to happen mid-calculation
      int scale = preferredScale;
      for (ExponentialHistogramDatapointAccum accumulator : accumulators) {
            scale = Math.min(scale, accumulator.
      }
      ExponentialHistogramDatapointAccum mutable = new ExponentialHistogramDatapointAccum(scale);
      // Add all histograms using the add function in the accumulators
      for (ExponentialHistogramDatapointAccum accumulator : accumulators) {
            mutable.add(accumulator);
      }
      return mutable;
}
```

Opportunity for optimization

Design Choice

Have an **abstract** Combine

Exponential Histogram takes a double

Abstract Combine

```
@Override
public abstract ExponentialHistogramDatapointAccum addInput(
            ExponentialHistogramDatapointAccum mutableAccumulator, T value);


public class DoubleToExponentialHistogramDataPointCombineFn
            extends ExponentialHistogramCombineFn<Double> {

      @Override
      public ExponentialHistogramDatapointAccum addInput(
                  ExponentialHistogramDatapointAccum mutableAccumulator, Double value) {
            mutableAccumulator.addValue(value);
            return mutableAccumulator;
      }
}
```

Abstract Combine

```
@Override
public abstract ExponentialHistogramDatapointAccum addInput(
            ExponentialHistogramDatapointAccum mutableAccumulator, T value);


public class ExponentialHistogramDataPointAccumCombineFn
            extends ExponentialHistogramCombineFn<ExponentialHistogramDatapointAccum> {

      @Override
      public ExponentialHistogramDatapointAccum addInput(
                  ExponentialHistogramDatapointAccum mutableAccumulator,
                  ExponentialHistogramDatapointAccum value) {
            mutableAccumulator.add(value);
            return mutableAccumulator;
      }
```

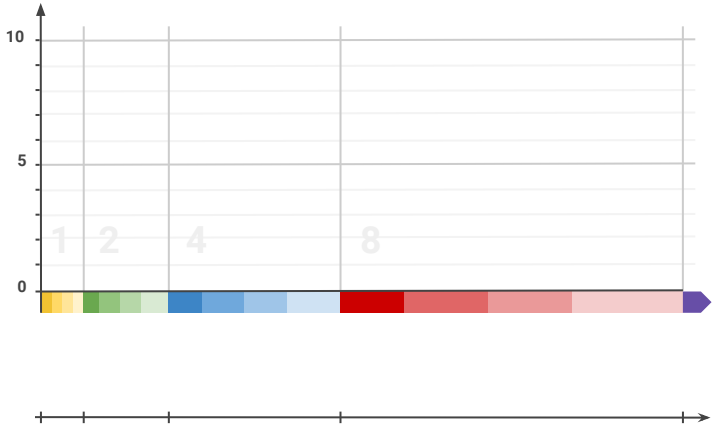
Combine in action

Combine

Doubles into Exponential Histograms

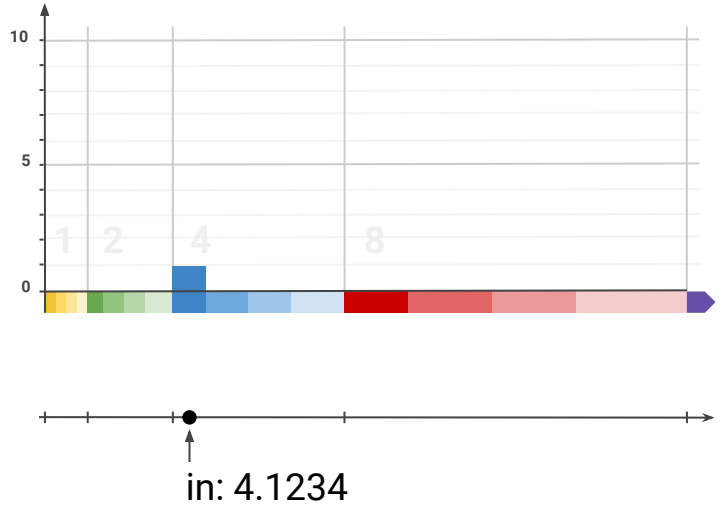
Visual guide to Exponential Histograms

From Double to Exp. Histogram



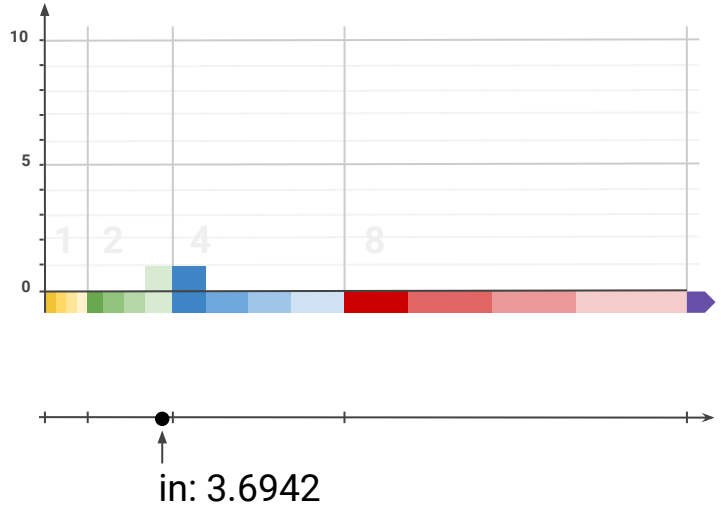
SUM	0
COUNT	0
AVG	0
SIZE (by)	32
SIZE in (by)	0

From Double to Exp. Histogram



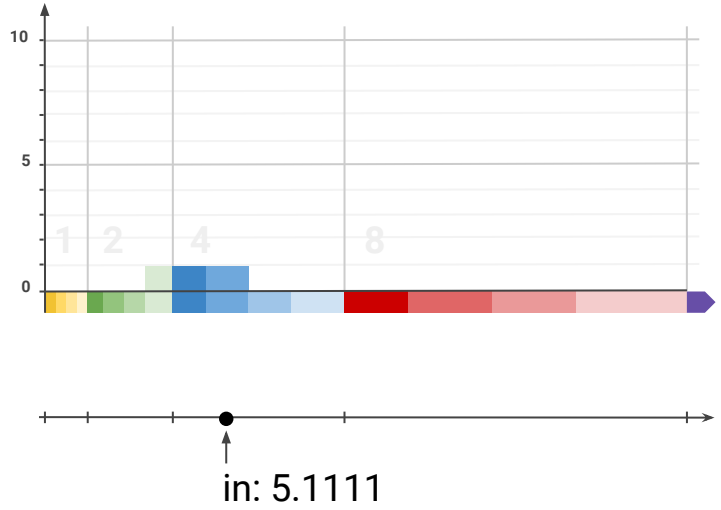
SUM	4.1234
COUNT	1
AVG	4.1234
SIZE (by)	68 (32+36)
SIZE in (by)	4

From Double to Exp. Histogram



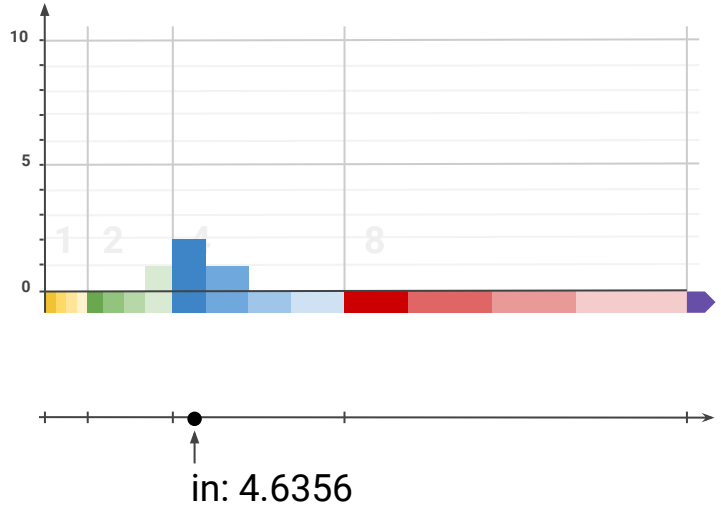
SUM	7.8176
COUNT	2
AVG	3.9088
SIZE (by)	68 (32+36)
SIZE in (by)	8

From Double to Exp. Histogram



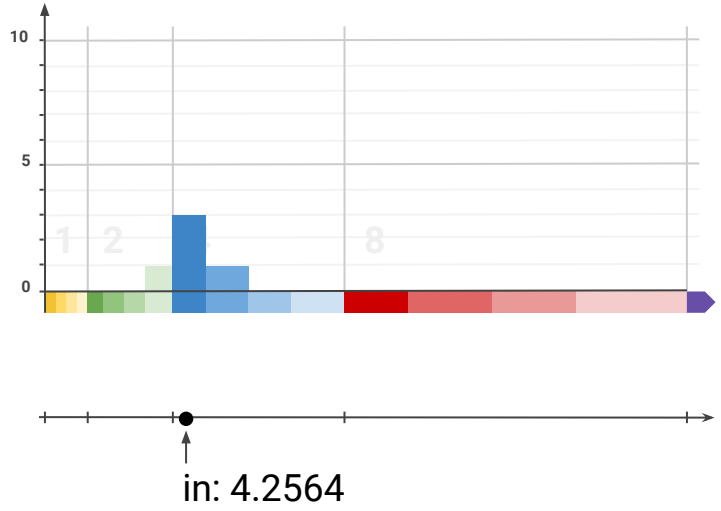
SUM	12.9287
COUNT	3
AVG	4.3096
SIZE (by)	72 (32+40)
SIZE in (by)	12

From Double to Exp. Histogram



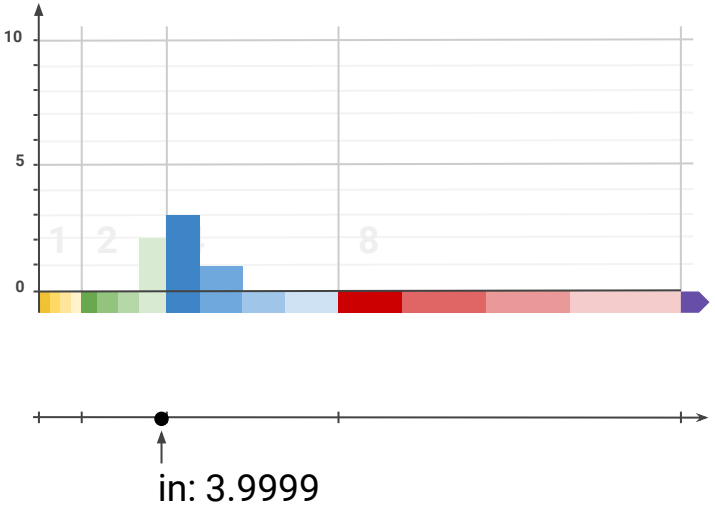
SUM	17.5643
COUNT	4
AVG	4.3911
SIZE (by)	72 (32+40)
SIZE in (by)	16

From Double to Exp. Histogram



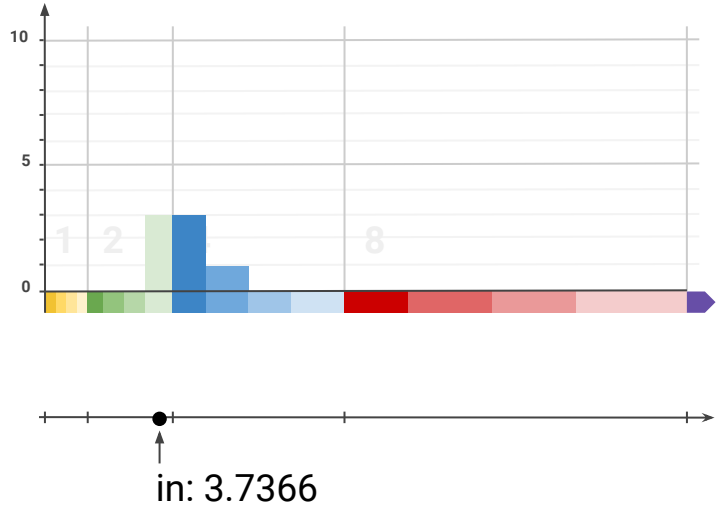
SUM	21.8207
COUNT	5
AVG	4.3641
SIZE (by)	72 (32+40)
SIZE in (by)	20

From Double to Exp. Histogram



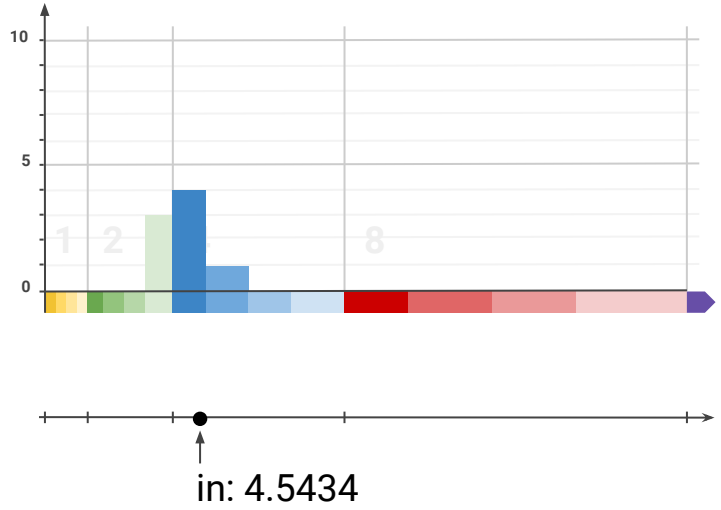
SUM	25.8206
COUNT	6
AVG	4.3034
SIZE (by)	72 (32+40)
SIZE in (by)	24

From Double to Exp. Histogram



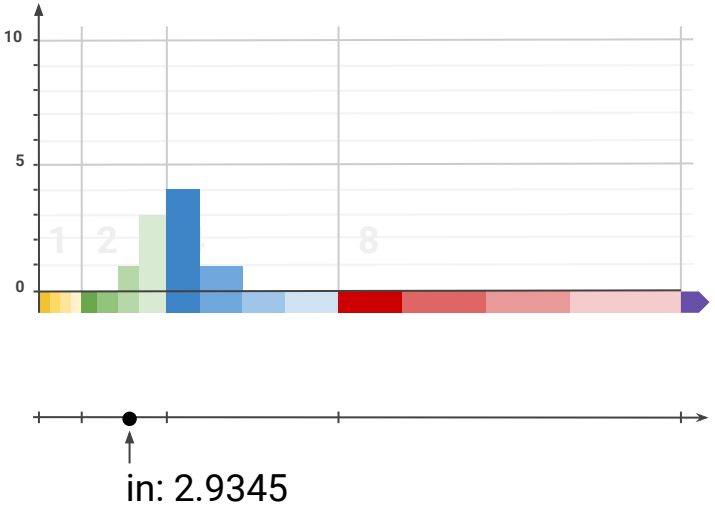
SUM	29.5572
COUNT	7
AVG	4.2225
SIZE (by)	72 (32+40)
SIZE in (by)	28

From Double to Exp. Histogram



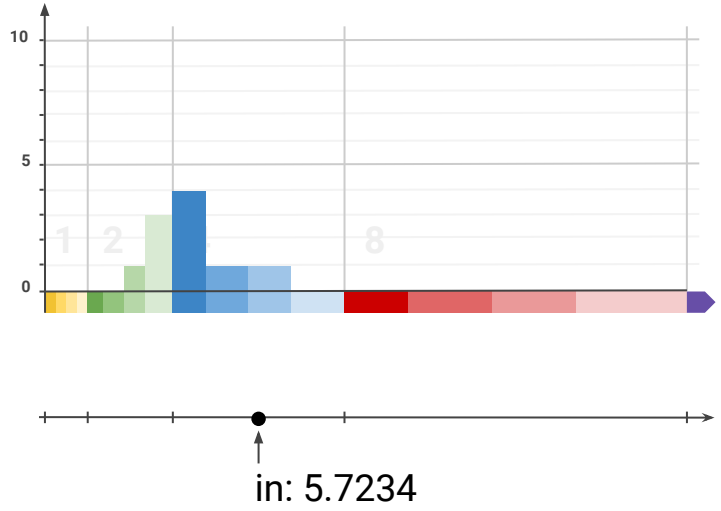
SUM	34.1006
COUNT	8
AVG	4.2626
SIZE (by)	72 (32+40)
SIZE in (by)	32

From Double to Exp. Histogram



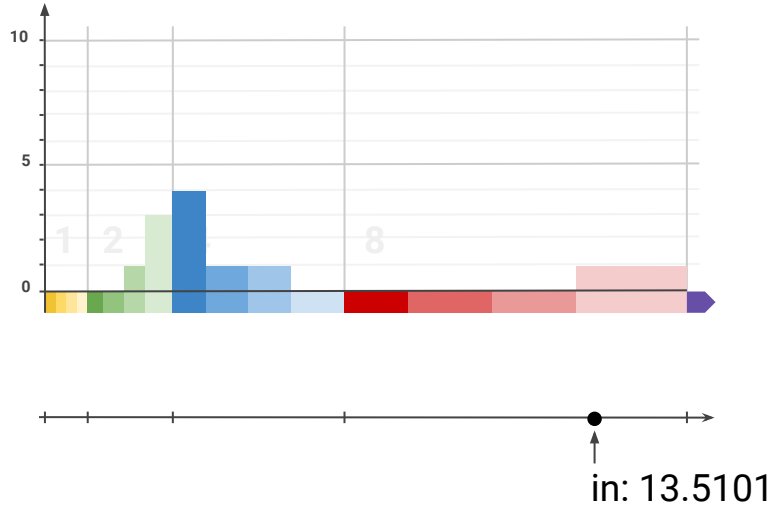
SUM	37.0351
COUNT	9
AVG	4.1150
SIZE (by)	72 (32+40)
SIZE in (by)	36

From Double to Exp. Histogram



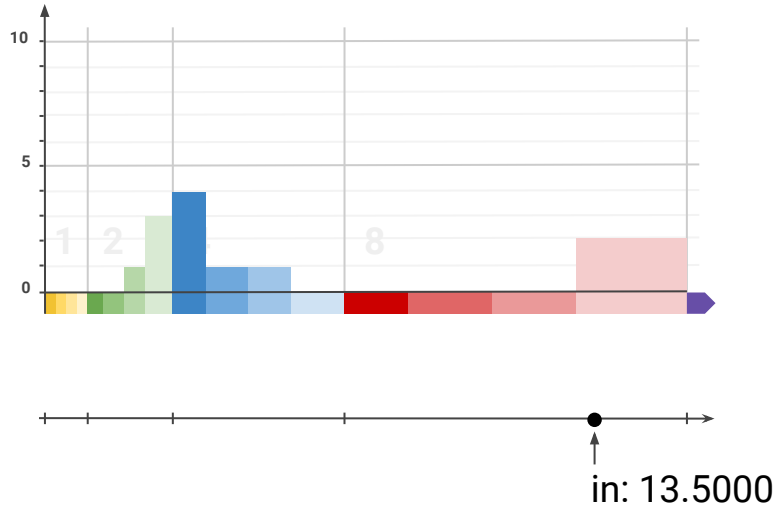
SUM	42.7585
COUNT	10
AVG	4.2759
SIZE (by)	76 (32+44)
SIZE in (by)	40

From Double to Exp. Histogram



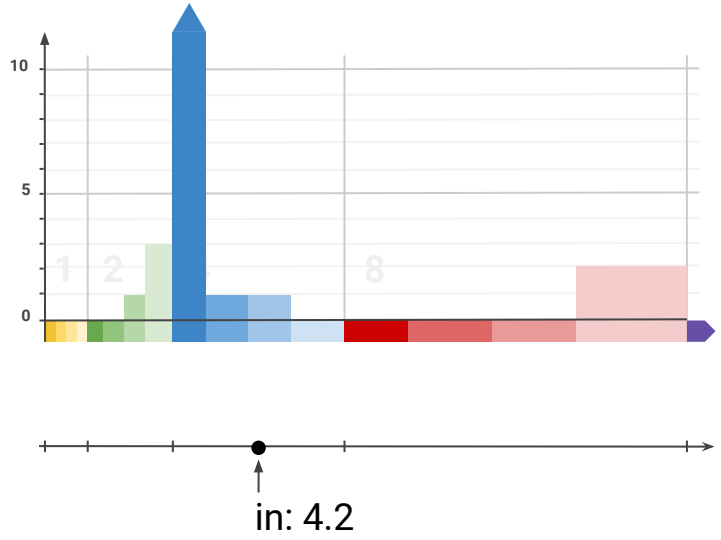
SUM	42.7585
COUNT	11
AVG	5.1153
SIZE (by)	96 (32+64)
SIZE in (by)	44

From Double to Exp. Histogram



SUM	42.7585
COUNT	12
AVG	5.8141
SIZE (by)	96 (32+64)
SIZE in (by)	48

From Double to Exp. Histogram



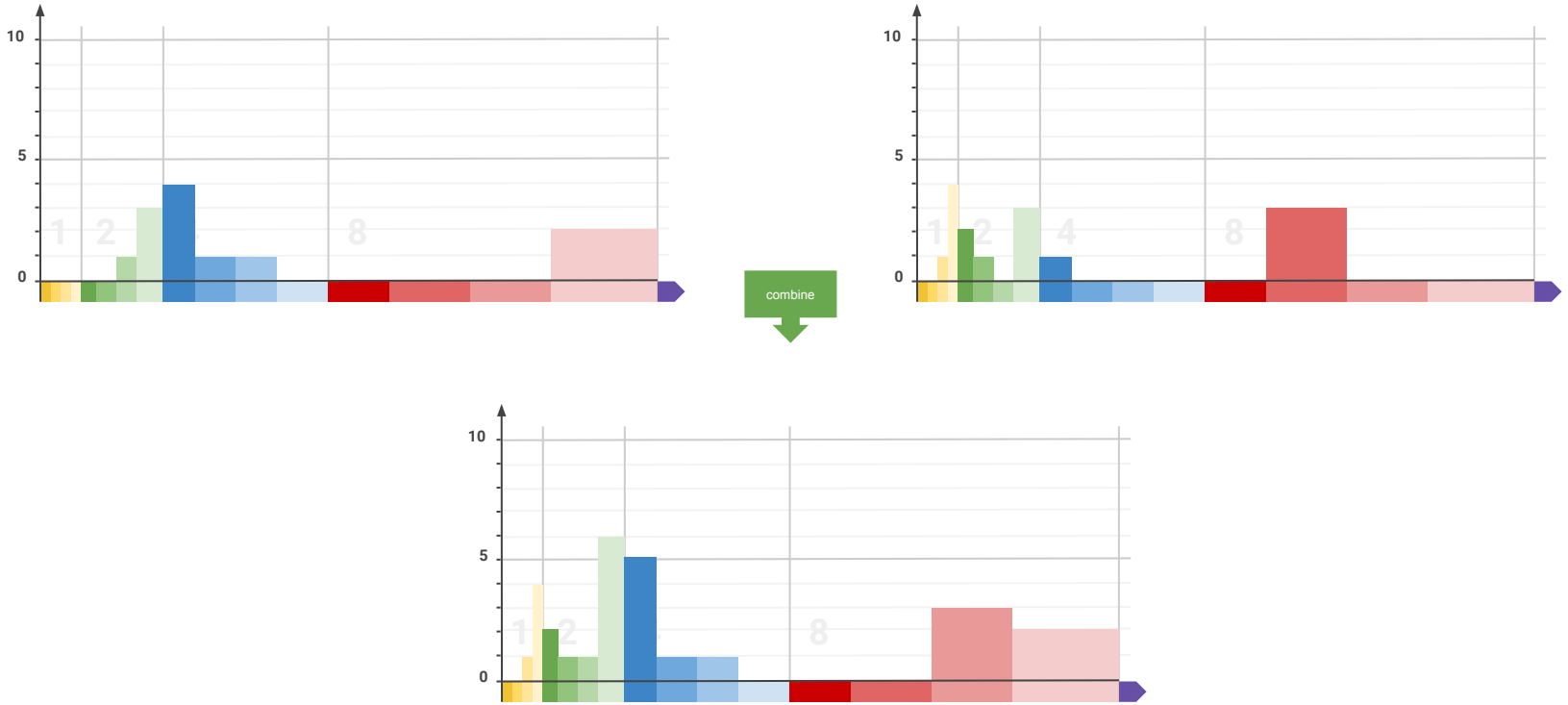
SUM	153.7686
COUNT	32
AVG	4.8053
SIZE (by)	96 (32+64)
SIZE in (by)	128

Combine

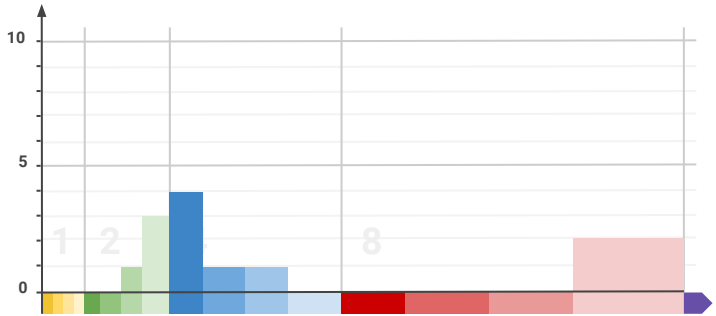
Exponential Histograms with each other

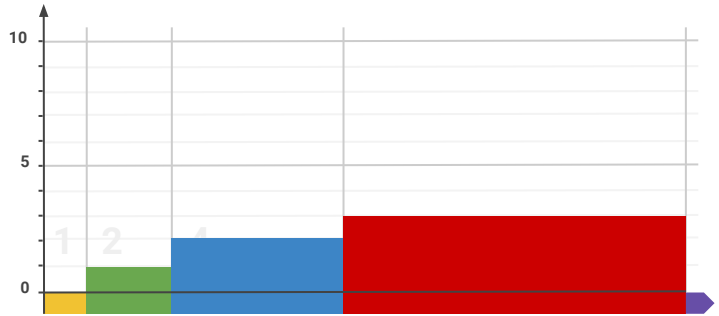
Visual guide to Exponential Histograms

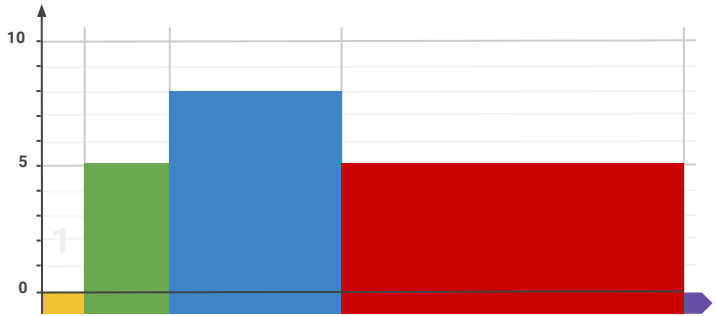
Combine with same scale



Combine with different scale





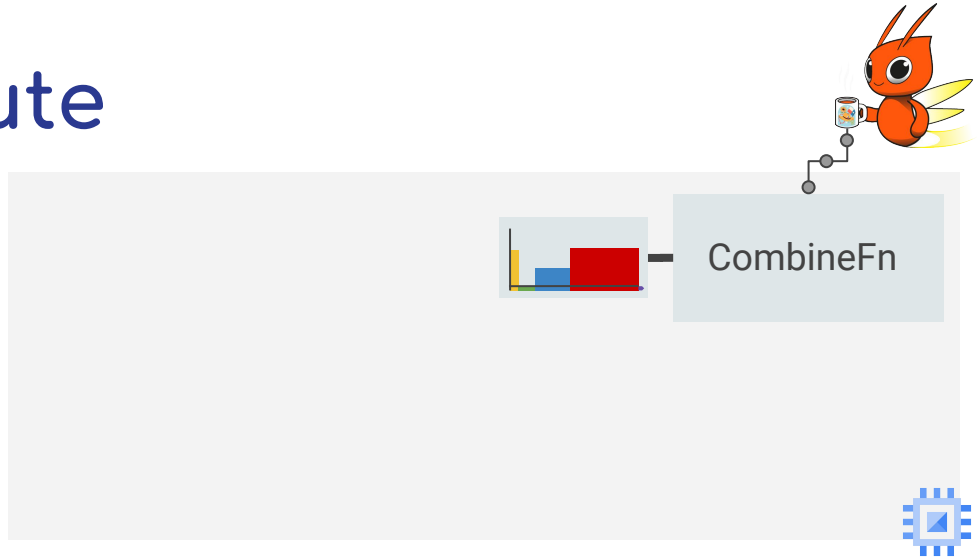


Combine

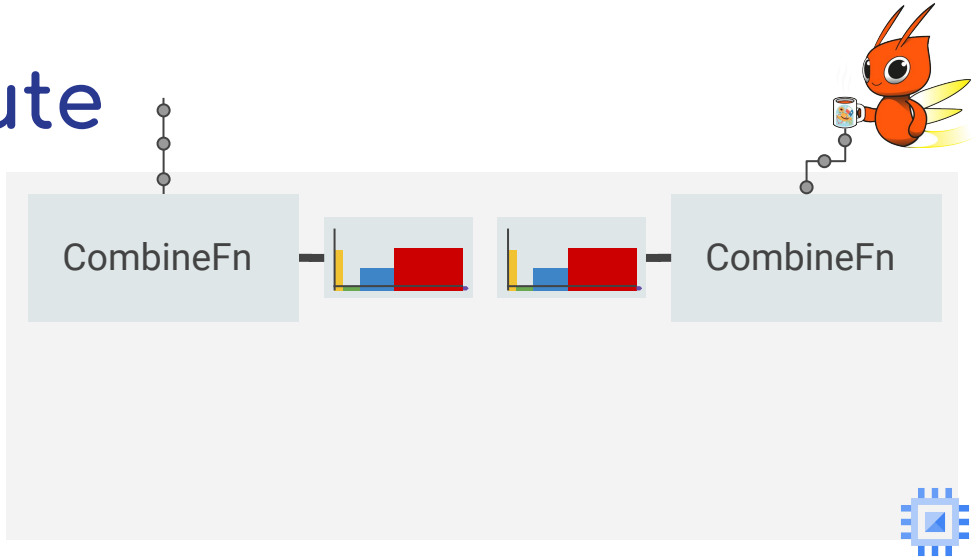
on the compute

Visual guide to Exponential Histograms

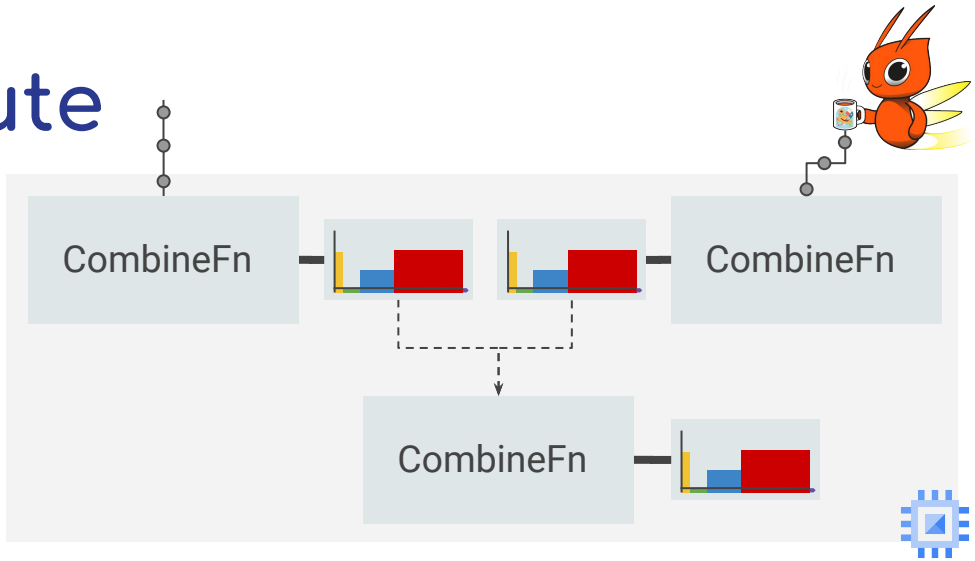
Combine on compute



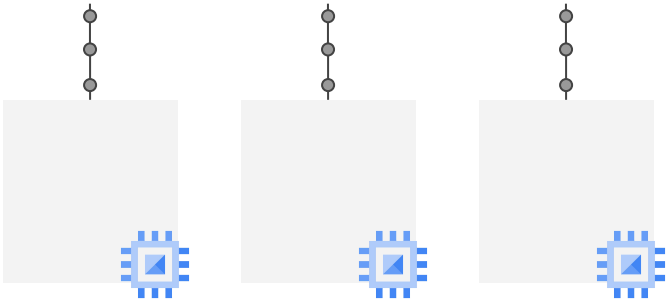
Combine on compute

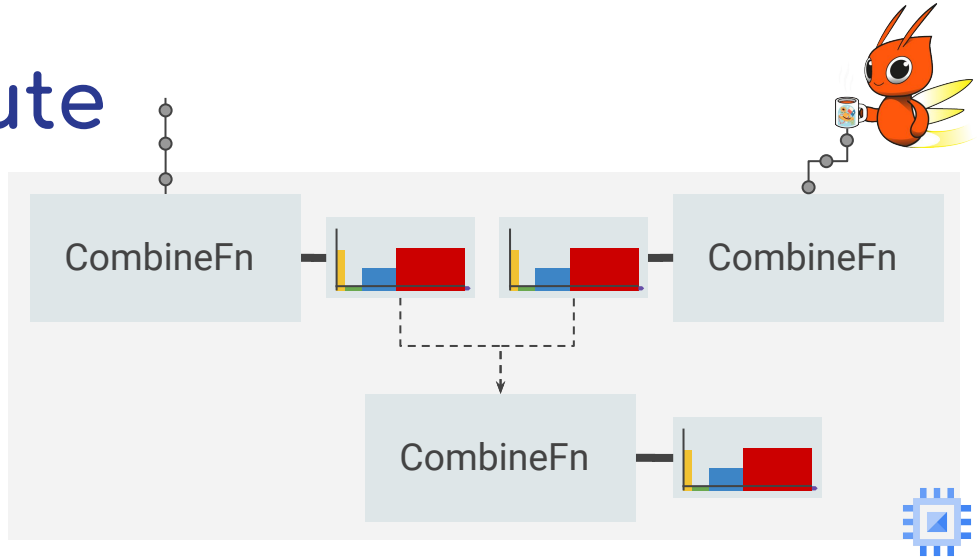


Combine on compute

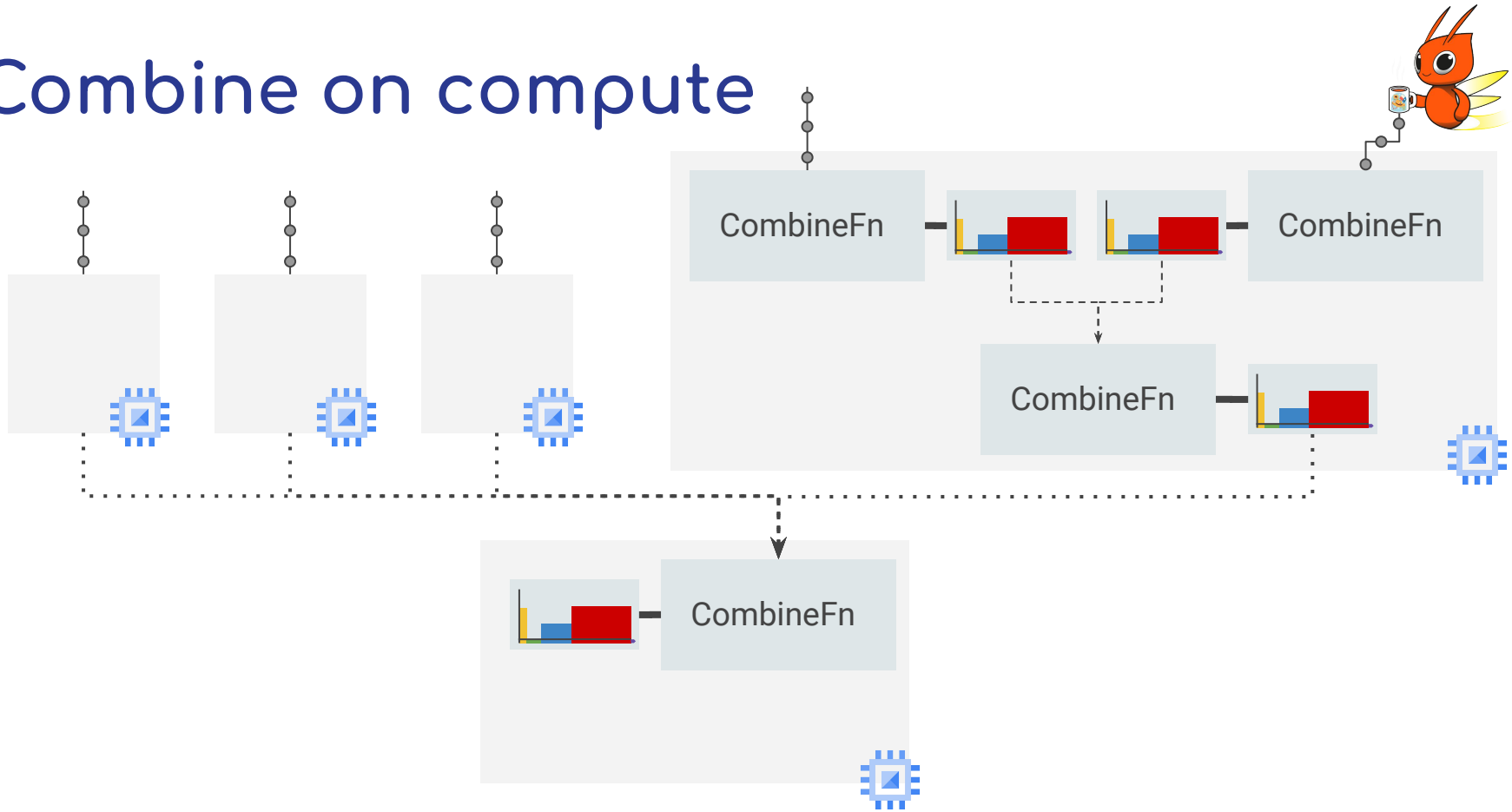


Combine on compute

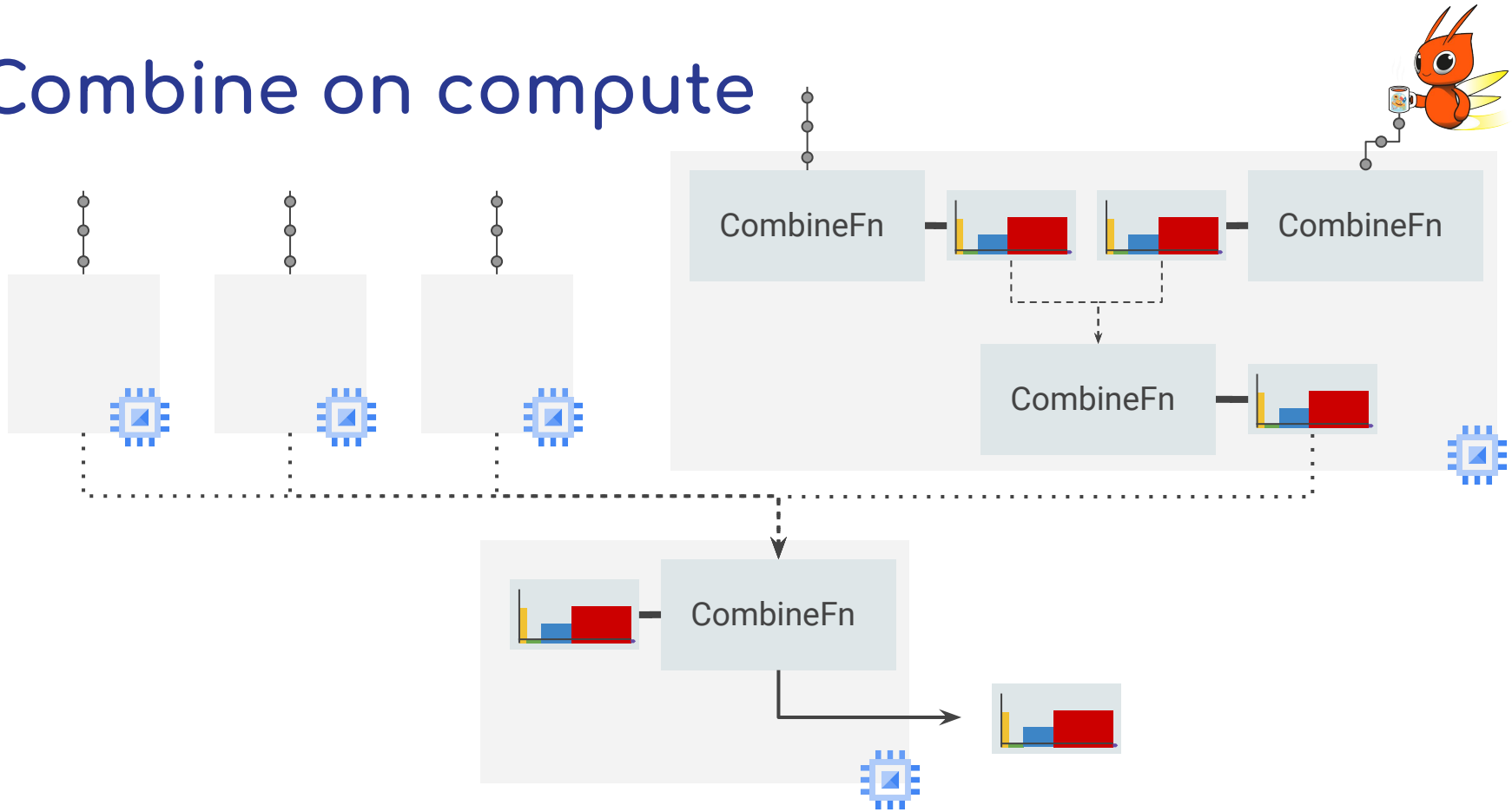




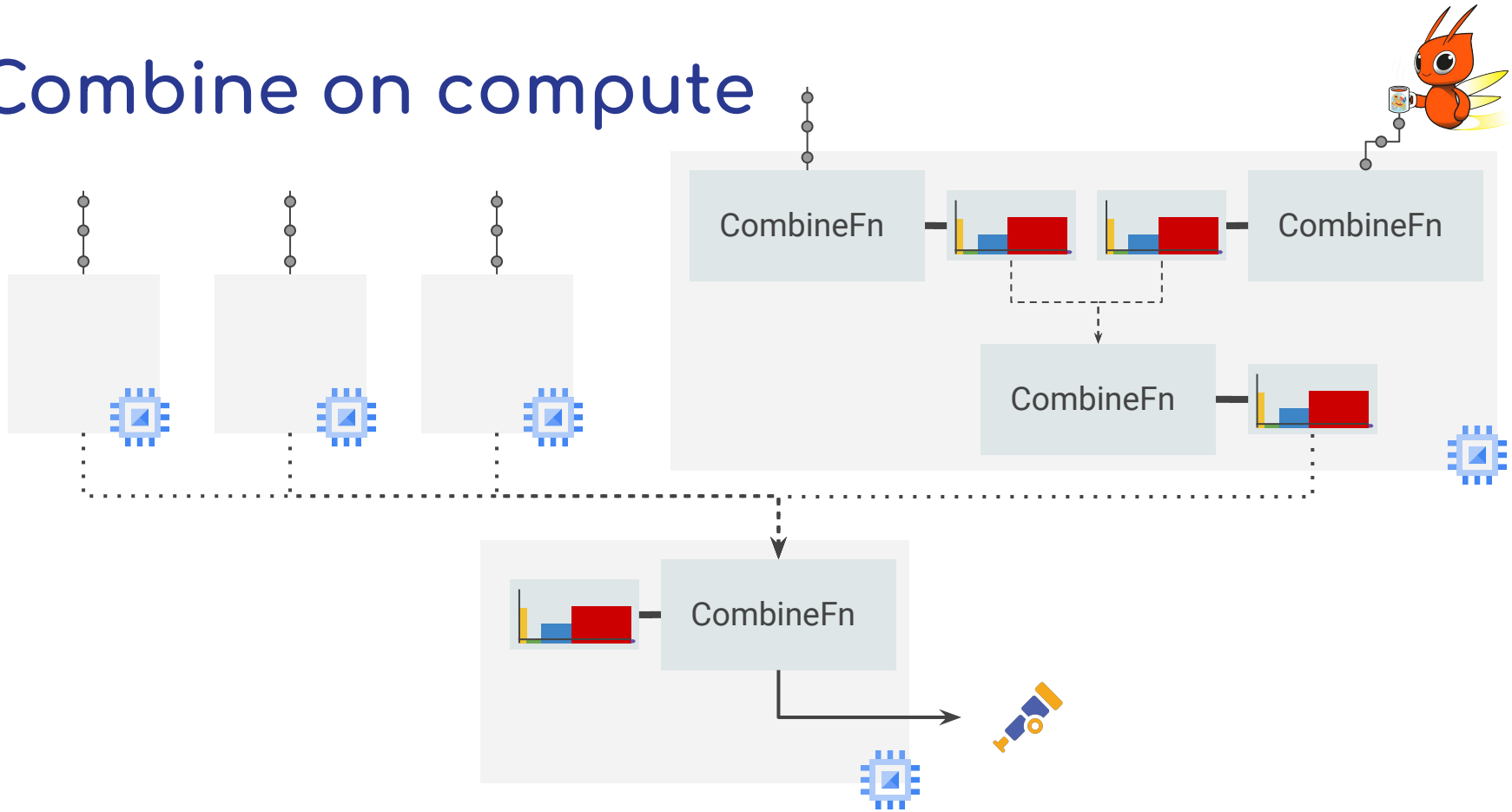
Combine on compute



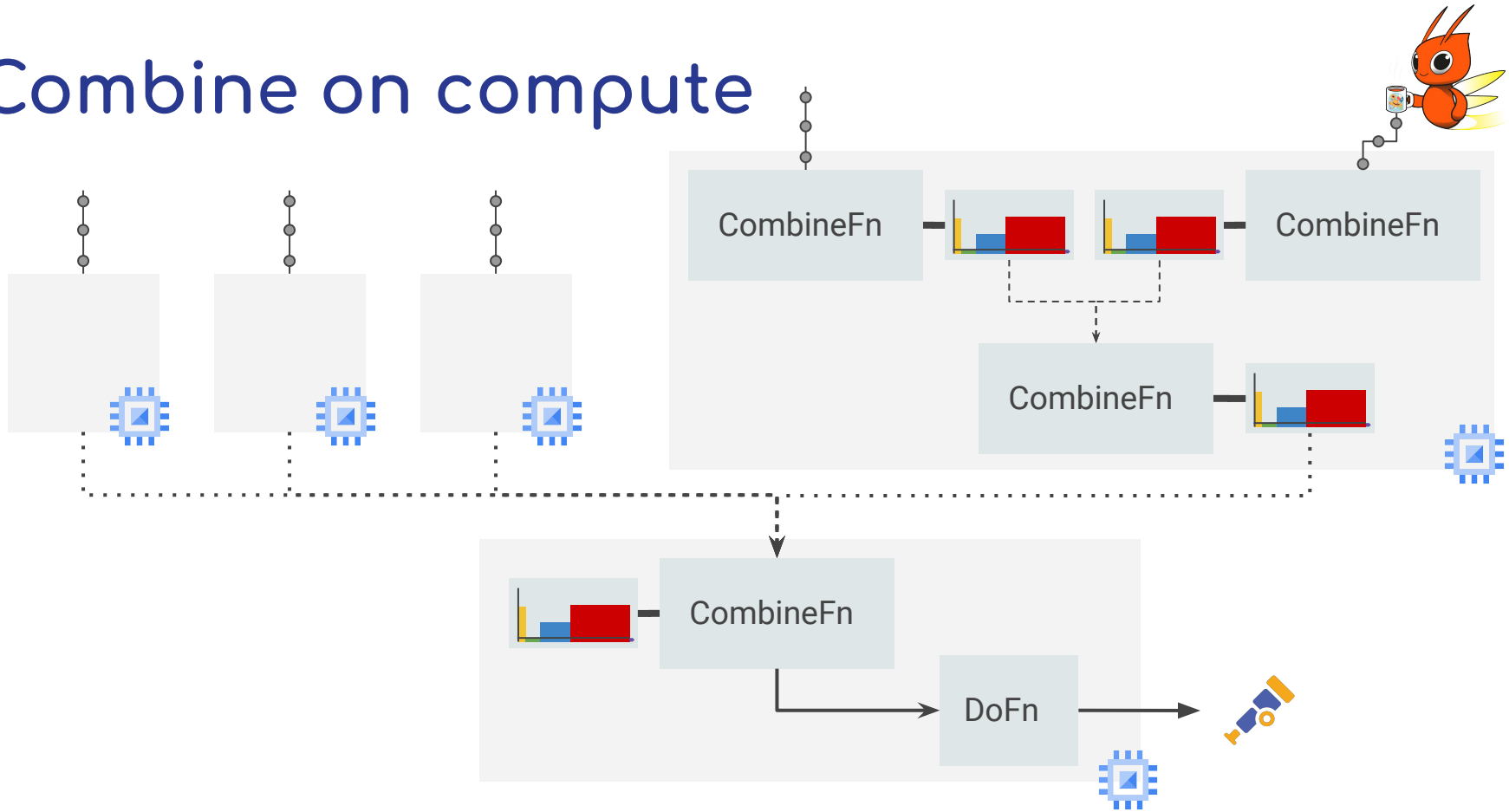
Combine on compute



Combine on compute



Combine on compute



Conclusion

BEAM SUMMIT Austin, 2022

Next...

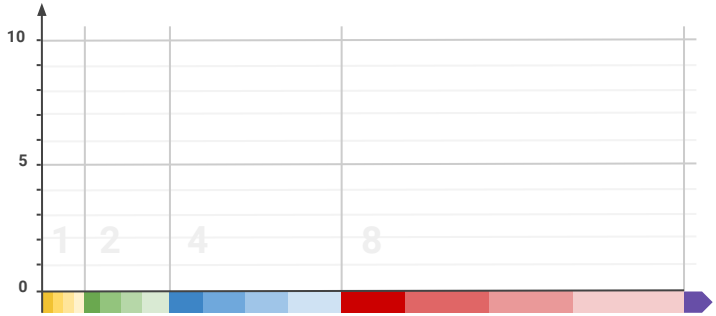
- Redo it in Go (as it’s now in GA)
- Full implementation
- Examplar support
- PromQL Beam (it’s just an idea)

Questions?

Maybe some contact info here?Twitter @alexvb
GitHub alexvanboxel
Or whatever you want

BEAM SUMMIT Austin, 2022

Scale



Scale