Combine by Example
OpenTelemetry Exponential Histograms
By Alex Van Boxel
Combine are just more Fn
OpenTelemetry
OpenTelemetry

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 = \( (2^{2^{\text{scale}}}) \)

\((\text{base}^{\text{index}}) \leq \text{value} < (\text{base}^{\text{index}+1})\)
<table>
<thead>
<tr>
<th>scale</th>
<th>-2</th>
<th>-1</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>base</td>
<td>16</td>
<td>4</td>
<td>2</td>
<td>1.414213562</td>
<td>1.189207115</td>
<td>1.090507733</td>
<td>1.044273782</td>
<td>1.021897149</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>16</td>
<td>4</td>
<td>2</td>
<td>1.414213562</td>
<td>1.189207115</td>
<td>1.090507733</td>
<td>1.044273782</td>
<td>1.021897149</td>
</tr>
<tr>
<td>2</td>
<td>256</td>
<td>16</td>
<td>4</td>
<td>2</td>
<td>1.414213562</td>
<td>1.189207115</td>
<td>1.090507733</td>
<td>1.044273782</td>
</tr>
<tr>
<td>3</td>
<td>4096</td>
<td>64</td>
<td>8</td>
<td>2.828427125</td>
<td>1.681792831</td>
<td>1.296839555</td>
<td>1.138788635</td>
<td>1.067140401</td>
</tr>
<tr>
<td>4</td>
<td>65536</td>
<td>256</td>
<td>16</td>
<td>4</td>
<td>2</td>
<td>1.414213562</td>
<td>1.189207115</td>
<td>1.090507733</td>
</tr>
<tr>
<td>5</td>
<td>1048576</td>
<td>1024</td>
<td>32</td>
<td>5.656854249</td>
<td>2.37841423</td>
<td>1.542210825</td>
<td>1.241857812</td>
<td>1.114386743</td>
</tr>
<tr>
<td>6</td>
<td>16777216</td>
<td>4096</td>
<td>64</td>
<td>8</td>
<td>2.828427125</td>
<td>1.681792831</td>
<td>1.296839555</td>
<td>1.138788635</td>
</tr>
<tr>
<td>7</td>
<td>268435456</td>
<td>16384</td>
<td>128</td>
<td>11.3137085</td>
<td>3.363585661</td>
<td>1.834008086</td>
<td>1.354255547</td>
<td>1.163724859</td>
</tr>
<tr>
<td>8</td>
<td>4294967296</td>
<td>65536</td>
<td>256</td>
<td>16</td>
<td>4</td>
<td>2</td>
<td>1.414213562</td>
<td>1.189207115</td>
</tr>
<tr>
<td>9</td>
<td>68719476736</td>
<td>262144</td>
<td>512</td>
<td>22.627417</td>
<td>4.75682846</td>
<td>2.181015465</td>
<td>1.476826146</td>
<td>1.21524736</td>
</tr>
<tr>
<td>10</td>
<td>109951162776</td>
<td>1048576</td>
<td>1024</td>
<td>32</td>
<td>5.656854249</td>
<td>2.37841423</td>
<td>1.542210825</td>
<td>1.241857812</td>
</tr>
<tr>
<td>11</td>
<td>1759218604441</td>
<td>4194304</td>
<td>2048</td>
<td>45.254834</td>
<td>6.727171322</td>
<td>2.593679109</td>
<td>1.610490332</td>
<td>1.269050957</td>
</tr>
<tr>
<td>12</td>
<td>2814749767106</td>
<td>16777216</td>
<td>4096</td>
<td>64</td>
<td>8</td>
<td>2.828427125</td>
<td>1.681792831</td>
<td>1.296839555</td>
</tr>
<tr>
<td>13</td>
<td>4.5036E+15</td>
<td>67108864</td>
<td>8192</td>
<td>90.50966799</td>
<td>9.51365692</td>
<td>3.084421651</td>
<td>1.75625216</td>
<td>1.325236643</td>
</tr>
<tr>
<td>14</td>
<td>7.20576E+16</td>
<td>268435456</td>
<td>16384</td>
<td>128</td>
<td>11.3137085</td>
<td>3.363585661</td>
<td>1.834008086</td>
<td>1.354255547</td>
</tr>
<tr>
<td>15</td>
<td>1.15292E+18</td>
<td>1073741824</td>
<td>32768</td>
<td>181.019336</td>
<td>13.45434264</td>
<td>3.668016173</td>
<td>1.915206561</td>
<td>1.383909882</td>
</tr>
<tr>
<td>16</td>
<td>1.84467E+19</td>
<td>4294967296</td>
<td>65536</td>
<td>256</td>
<td>16</td>
<td>4</td>
<td>2</td>
<td>1.414213562</td>
</tr>
<tr>
<td>17</td>
<td>2.95148E+20</td>
<td>17179869184</td>
<td>131072</td>
<td>362.038672</td>
<td>19.02731384</td>
<td>4.362030931</td>
<td>2.088547565</td>
<td>1.445180807</td>
</tr>
</tbody>
</table>

**Scale**
Scale

scale

2

1

0

-1

-2

scale
text
Protobuf

The OpenTelemetry Exponential Histogram

A deep dive into the Exponential Histograms original 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

```plaintext
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;
// 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);
}
Why not to use the pure protobuf as an accumulator?

Design Choice

No Proto as the accumulator
Mutable

/** Mutable Datapoint Accumulator for Exponential Histograms */
public final class ExponentialHistogramDatapointAccum implements 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?
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

```java
@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

```java
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

```java
@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

```java
@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;
}
```

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

```java
@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
From Double to Exp. Histogram

<table>
<thead>
<tr>
<th></th>
<th>SUM</th>
<th>COUNT</th>
<th>AVG</th>
<th>SIZE (by)</th>
<th>SIZE in (by)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>32</td>
<td>0</td>
</tr>
</tbody>
</table>
From Double to Exp. Histogram

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>SUM</td>
<td>4.1234</td>
</tr>
<tr>
<td>COUNT</td>
<td>1</td>
</tr>
<tr>
<td>AVG</td>
<td>4.1234</td>
</tr>
<tr>
<td>SIZE (by)</td>
<td>68 (32+36)</td>
</tr>
<tr>
<td>SIZE in (by)</td>
<td>4</td>
</tr>
</tbody>
</table>

![Histogram Diagram]

in: 4.1234
From Double to Exp. Histogram

<table>
<thead>
<tr>
<th>SUM</th>
<th>7.8176</th>
</tr>
</thead>
<tbody>
<tr>
<td>COUNT</td>
<td>2</td>
</tr>
<tr>
<td>AVG</td>
<td>3.9088</td>
</tr>
<tr>
<td>SIZE (by)</td>
<td>68 (32+36)</td>
</tr>
<tr>
<td>SIZE in (by)</td>
<td>8</td>
</tr>
</tbody>
</table>

in: 3.6942
From Double to Exp. Histogram

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>SUM</td>
<td>12.9287</td>
</tr>
<tr>
<td>COUNT</td>
<td>3</td>
</tr>
<tr>
<td>AVG</td>
<td>4.3096</td>
</tr>
<tr>
<td>SIZE (by)</td>
<td>72 (32+40)</td>
</tr>
<tr>
<td>SIZE in (by)</td>
<td>12</td>
</tr>
</tbody>
</table>

in: 5.1111
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

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>SUM</td>
<td>21.8207</td>
</tr>
<tr>
<td>COUNT</td>
<td>5</td>
</tr>
<tr>
<td>AVG</td>
<td>4.3641</td>
</tr>
<tr>
<td>SIZE (by)</td>
<td>72 (32+40)</td>
</tr>
<tr>
<td>SIZE in (by)</td>
<td>20</td>
</tr>
</tbody>
</table>

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

![Histogram diagram](image)
From Double to Exp. Histogram

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

in: 3.9999
From Double to Exp. Histogram

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>SUM</td>
<td>29.5572</td>
</tr>
<tr>
<td>COUNT</td>
<td>7</td>
</tr>
<tr>
<td>AVG</td>
<td>4.2225</td>
</tr>
<tr>
<td>SIZE (by)</td>
<td>72 (32+40)</td>
</tr>
<tr>
<td>SIZE in (by)</td>
<td>28</td>
</tr>
</tbody>
</table>

in: 3.7366
From Double to Exp. Histogram

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

in: 4.5434
From Double to Exp. Histogram

<table>
<thead>
<tr>
<th>SUM</th>
<th>37.0351</th>
</tr>
</thead>
<tbody>
<tr>
<td>COUNT</td>
<td>9</td>
</tr>
<tr>
<td>AVG</td>
<td>4.1150</td>
</tr>
<tr>
<td>SIZE (by)</td>
<td>72 (32+40)</td>
</tr>
<tr>
<td>SIZE in (by)</td>
<td>36</td>
</tr>
</tbody>
</table>
From Double to Exp. Histogram

<table>
<thead>
<tr>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>SUM</td>
<td>42.7585</td>
</tr>
<tr>
<td>COUNT</td>
<td>10</td>
</tr>
<tr>
<td>AVG</td>
<td>4.2759</td>
</tr>
<tr>
<td>SIZE (by)</td>
<td>76 (32+44)</td>
</tr>
<tr>
<td>SIZE in (by)</td>
<td>40</td>
</tr>
</tbody>
</table>

in: 5.7234
From Double to Exp. Histogram

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SUM</strong></td>
<td>42.7585</td>
</tr>
<tr>
<td><strong>COUNT</strong></td>
<td>11</td>
</tr>
<tr>
<td><strong>AVG</strong></td>
<td>5.1153</td>
</tr>
<tr>
<td><strong>SIZE (by)</strong></td>
<td>96 (32+64)</td>
</tr>
<tr>
<td><strong>SIZE in (by)</strong></td>
<td>44</td>
</tr>
</tbody>
</table>

in: 13.5101
From Double to Exp. Histogram

<table>
<thead>
<tr>
<th>SUM</th>
<th>42.7585</th>
</tr>
</thead>
<tbody>
<tr>
<td>COUNT</td>
<td>12</td>
</tr>
<tr>
<td>AVG</td>
<td>5.8141</td>
</tr>
<tr>
<td>SIZE (by)</td>
<td>96 (32+64)</td>
</tr>
<tr>
<td>SIZE in (by)</td>
<td>48</td>
</tr>
</tbody>
</table>
From Double to Exp. Histogram

<table>
<thead>
<tr>
<th>SUM</th>
<th>153.7686</th>
</tr>
</thead>
<tbody>
<tr>
<td>COUNT</td>
<td>32</td>
</tr>
<tr>
<td>AVG</td>
<td>4.8053</td>
</tr>
<tr>
<td>SIZE (by)</td>
<td>96  (32+64)</td>
</tr>
<tr>
<td>SIZE in (by)</td>
<td>128</td>
</tr>
</tbody>
</table>
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
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
Scale
Scale