Advanced grouping & aggregation

Iñigo San Jose Visiers
Technical Solutions Engineer @ Google

Agenda

What’s a combiner?

Built-in and Custom Combiners

Combiner Interface

Demo

What’s a Combiner?

Definition

Types

Combiner definition

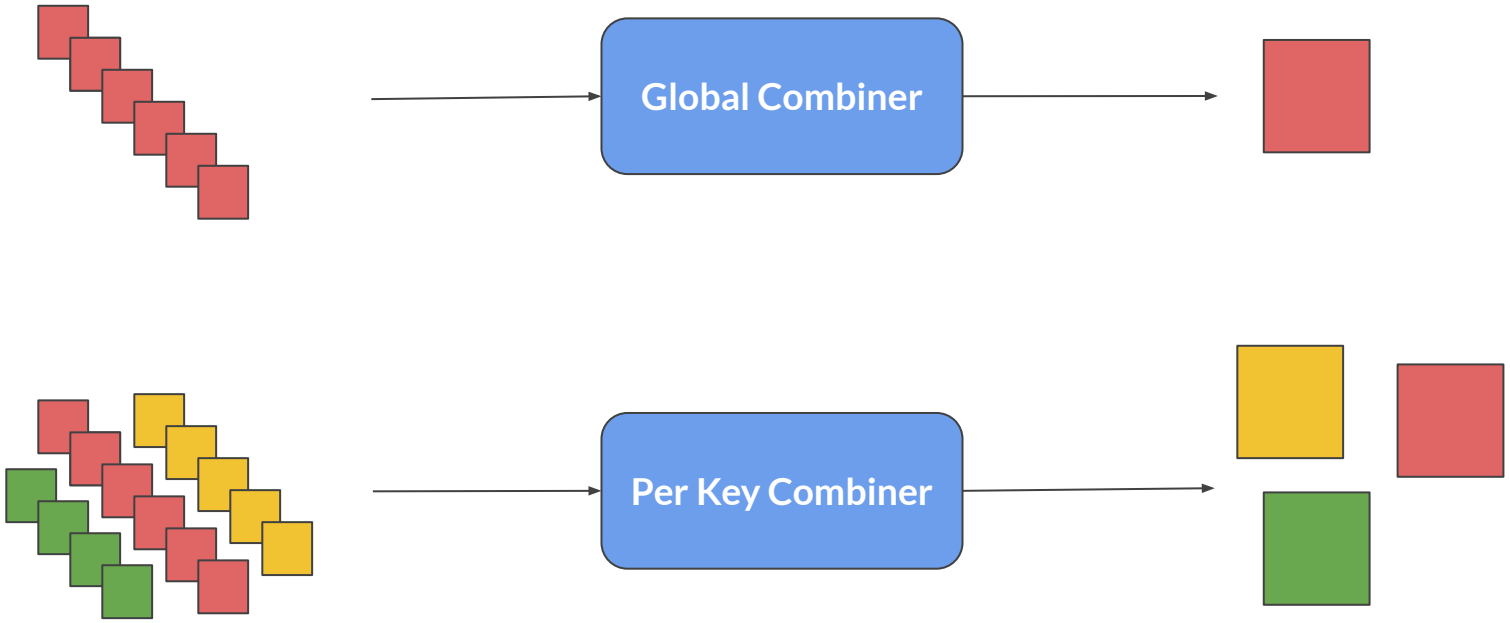
Combiners are *PTransforms* that aggregate a *PCollection*; this is, from multiple input elements, they output one element or one element per key.

Types of Combiner

Global Combiners aggregate the input *PCollection* into one output element (per window).

Per Key Combiners aggregate a *PCollection* of key values into one output element per each key (per window).

Types of Combiner



Built-in and Custom Combiners

Built-in Combiners

Custom Combiners

Combiner Lifting

Combiners and windows

Built-in Combiners

Apache Beam comes with some already-built Combiners.

Some of them (in Python API):

- **Count** outputs the total number of elements in a *PCollection*.
- **Mean** outputs arithmetic mean of a *PCollection*.
- **Top** outputs the *n* largest/smallest of a *PCollection* given a comparison.
- **ToList** a global CombineFn that condenses a *PCollection* into a single list**.**
- **ToDict** converts a *PCollection* of tuples into a dictionary.
- **Latest** outputs the latest element to arrive to the *PCollection*.
- *More here*.

Custom Combiners

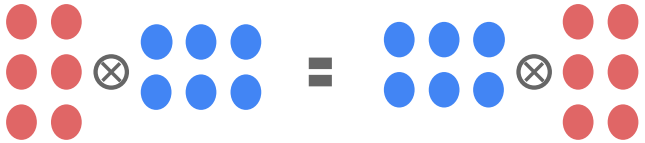
We can create our own *Combiners* using our own logic. We can do it both globally and per key:

- ***CombineGlobally***
- ***CombinePerKey***

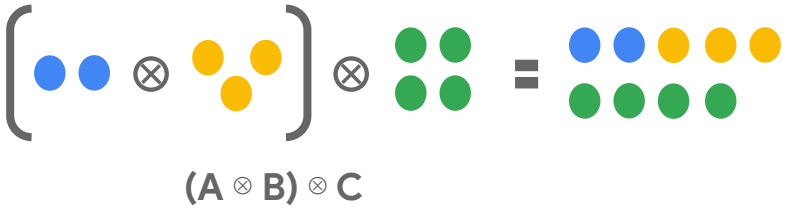
Custom Combiners

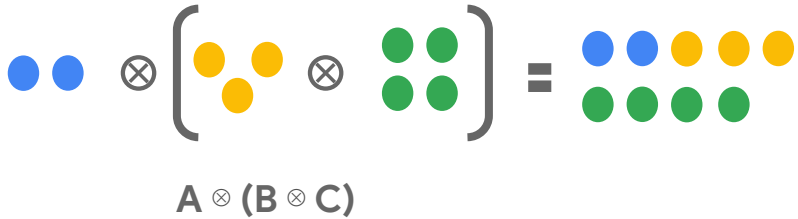
When applying a ***Combine*** transform, a function with the logic for combining elements or values needs to be provided. The function should be commutative and associative.

Commutative



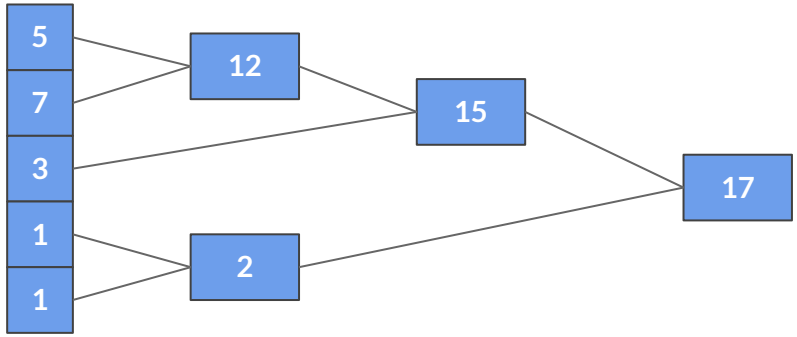
Associative



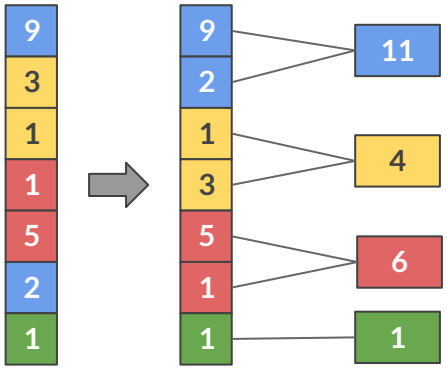


Custom Combiners

CombineGlobally(sum)



CombinePerKey(sum)



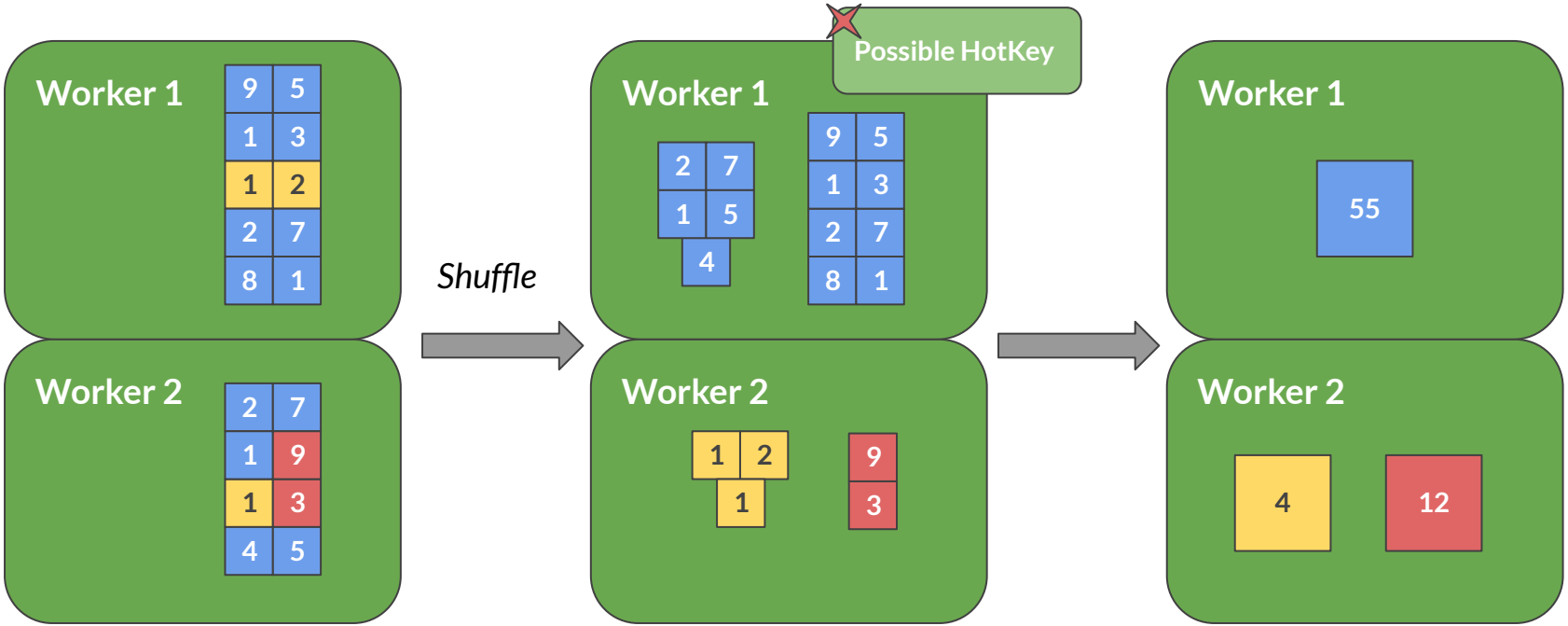
Combiner Lifting

Combiner Lifting is an optimization some runners (as Dataflow) have that make the combine operation to happen within the workers before shuffling.

Combiner Lifting not only would speed up the combine, but can help avoid *hot key* issues.

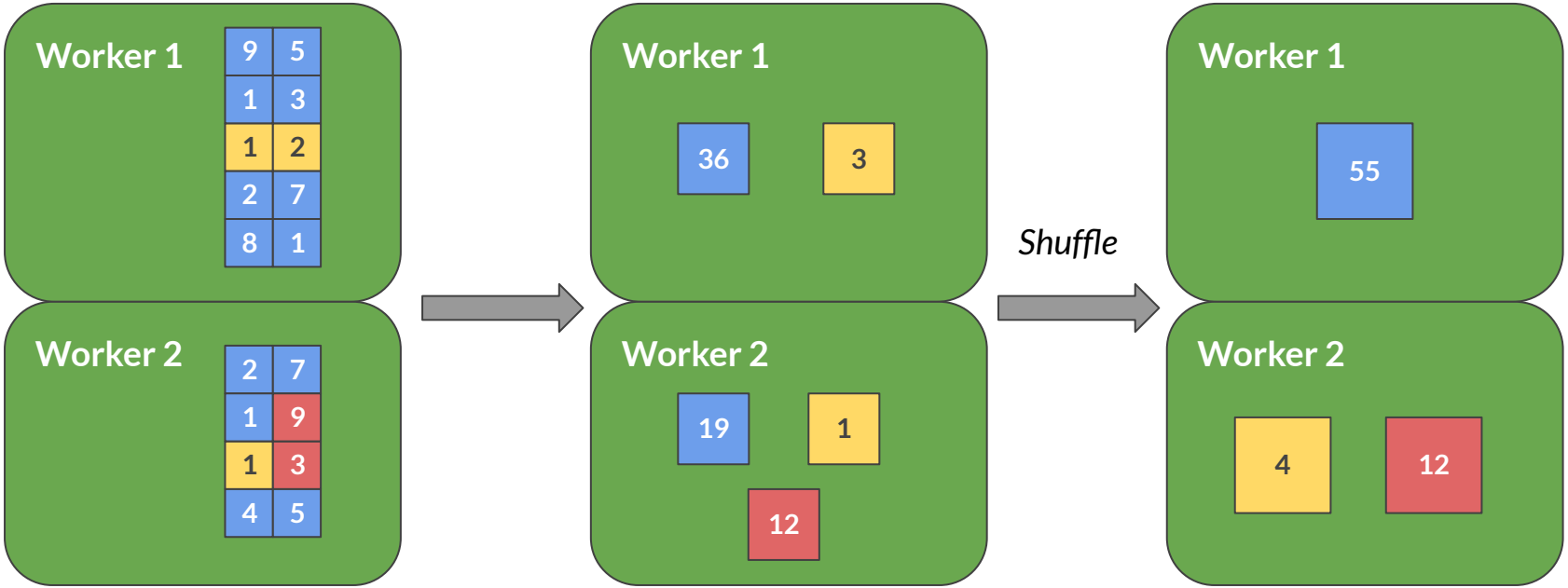
Combiner Lifting

Without Combiner Lifting



Combiner Lifting

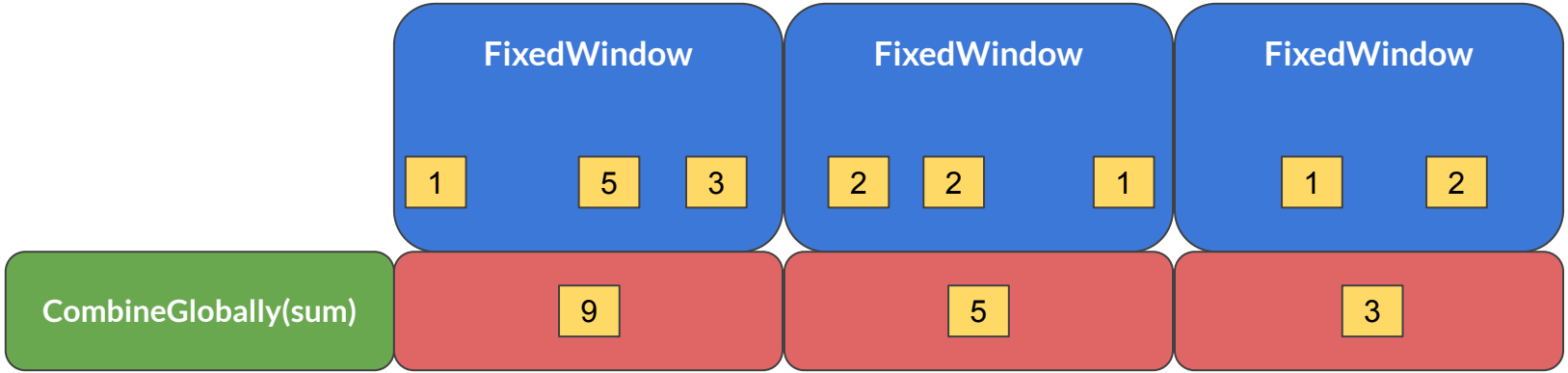
With Combiner Lifting



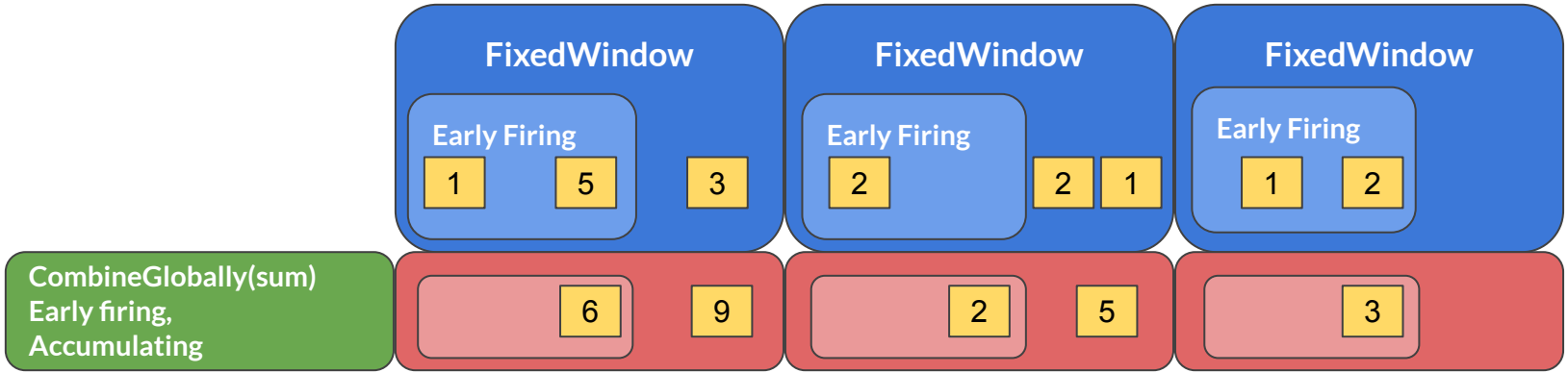
Combiners and Windows

Combiners also work on a window and trigger basis. Windows and/or triggers are needed when our input data that needs to be aggregated is *unbounded*.

Combiners and Windows



Combiners and Windows



Combiner Interface

Motivation

Operations

Motivation

We need the *Combiner Interface* when our *Combiner* requires a more sophisticated accumulator, must perform additional pre- or post-processing, might change the output type, or takes the key into account.

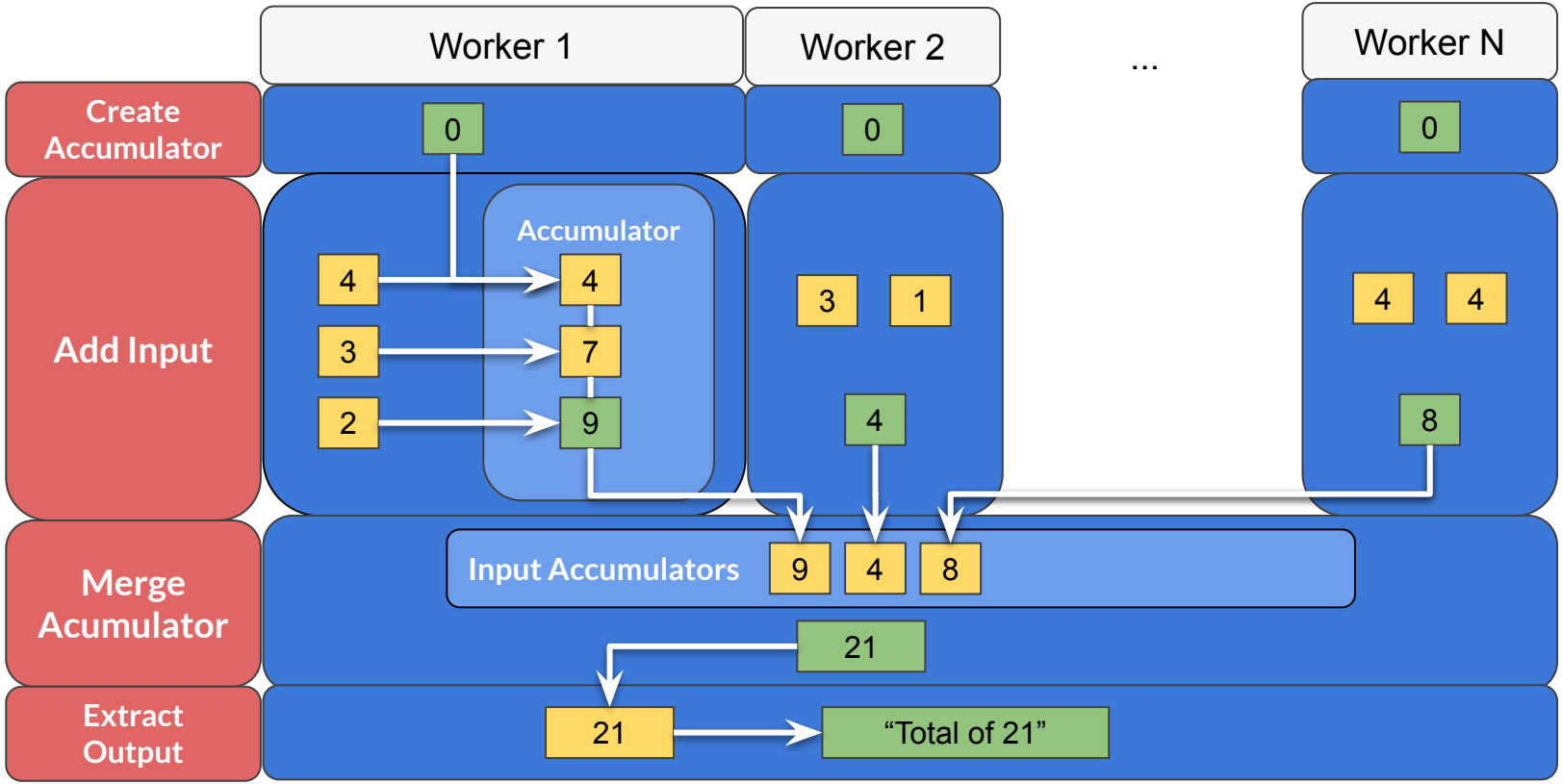
The *Combiner Interface* give us more control over the elements and how they are combined.

Operations

The ***Combiner Interface*** has a four operations workflow:

1. **Create Accumulator:** creates a “local” accumulator
2. **Add input:** adds the input element to the accumulator, returning the updated accumulator
3. **Merge Accumulators:** takes all accumulators and merges them into a single accumulate
4. **Extract Output:** final computation, returns the final value of the combiner.

Operations


Worker 1
Worker 2
...
Worker N
Create Accumulator
Add Input
Merge Acumulator
Extract Output
0
0
0
Accumulator
4
3
2
4
7
9
3
1
4
4
4
8
Input Accumulators
9
4
8
21
21
“Total of 21”

Demo

Thank you!

Questions?