Introducing Ordered List States

Shunping Huang
SWE @ Google

BEAM SUMMIT

September 4-5, 2024

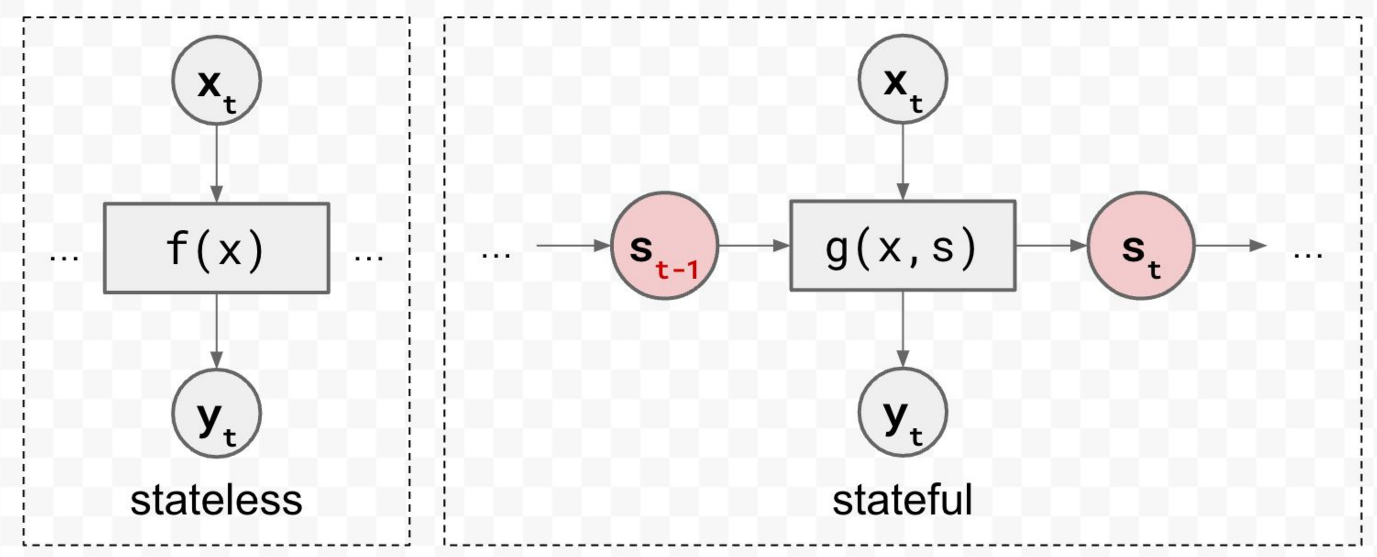
Sunnyvale, CA. USA

Agenda

- States and how they are handled in Beam
- OrderedListStates and the APIs
- Walkthrough on a simple example and pseudo code
- Demo (if time permits)

What Are States?

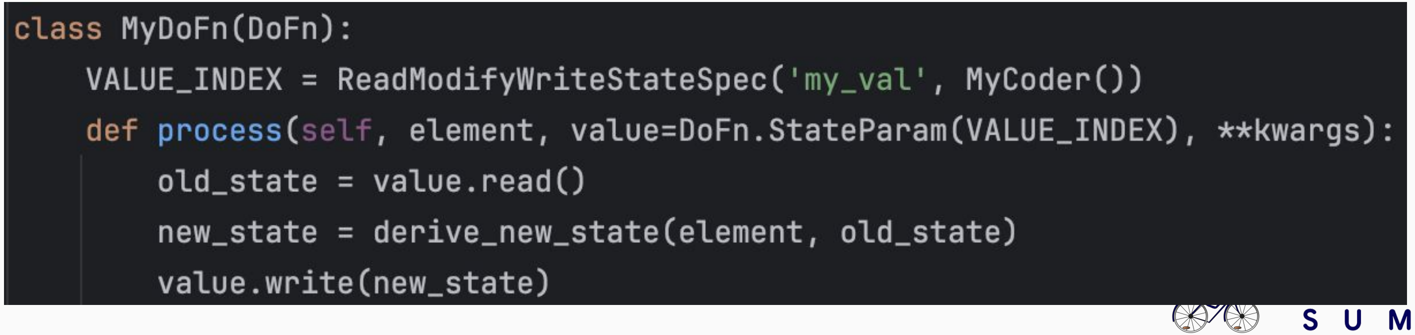
- Stateless vs Stateful



- States
  - the information derived from the **past** events that the operation executor has to remember in order to process the current event.

What Are States in Beam?

- Data structures (state classes) for users to store different types of states
- Properties
    - Must-have: unique identifier, element coder
    - Per-key-and-window
    - Accessed by a single thread
    - Standardized APIs, but runner-dependent implementation
- An example stateful DoFn in Python



```
class MyDoFn(DoFn):
    VALUE_INDEX = ReadModifyWriteStateSpec('my_val', MyCoder())
    def process(self, element, value=DoFn.StateParam(VALUE_INDEX), **kwargs):
        old_state = value.read()
        new_state = derive_new_state(element, old_state)
        value.write(new_state)
```

True or False

“We can use ValueState to represent **anything** we want to store as a state.”

- TRUE technically …

- But practically …

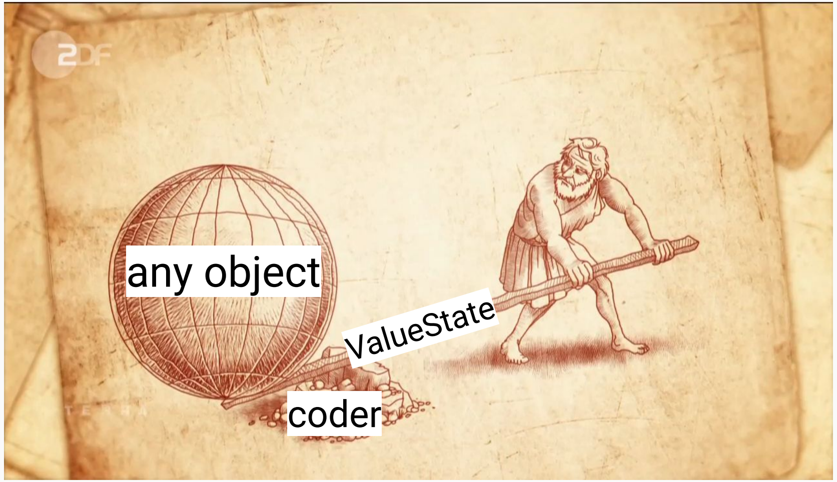
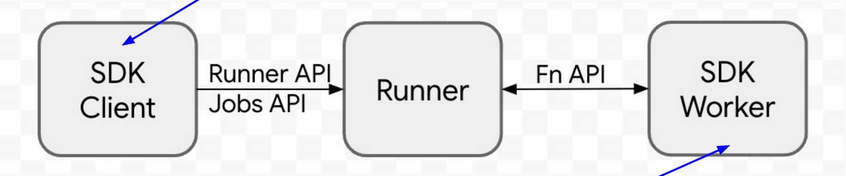


Image from : https://www.artistsjourney.com/blog/creativity-moving-the-earth-with-archimedes-stravinsk

How States Are Handled in Beam

```
def process(self, e,
    my_state=DoFn.StateParam(INDEX))
  …
```



SDK Client

Runner API

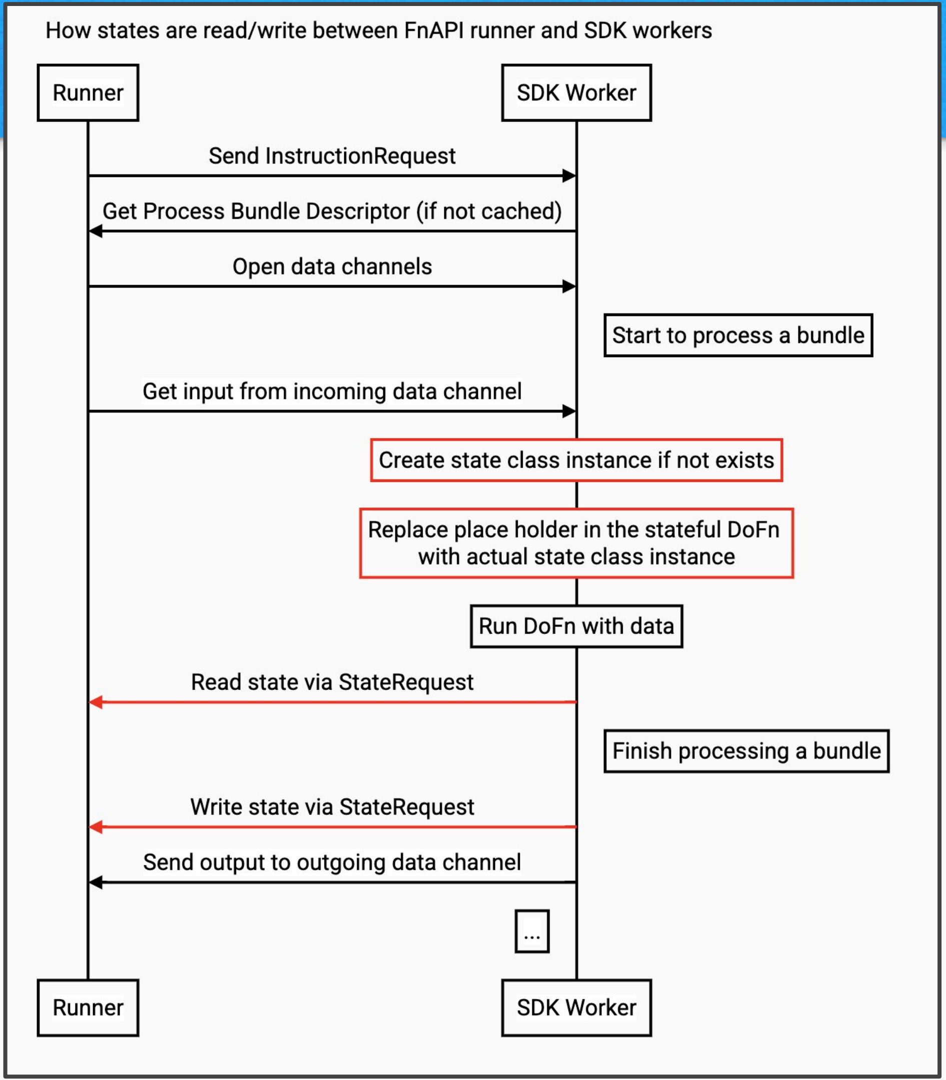
Jobs API

Runner

Fn API

SDK Worker

```
.process(element, actual_state_object)
```



How states are read/write between FnAPI runner and SDK workers

Runner

SDK Worker

Send InstructionRequest

Get Process Bundle Descriptor (if not cached)

Open data channels

Start to process a bundle

Get input from incoming data channel

Create state class instance if not exists

Replace place holder in the stateful DoFn with actual state class instance

Run DoFn with data

Read state via StateRequest

Finish processing a bundle

Write state via StateRequest

Send output to outgoing data channel

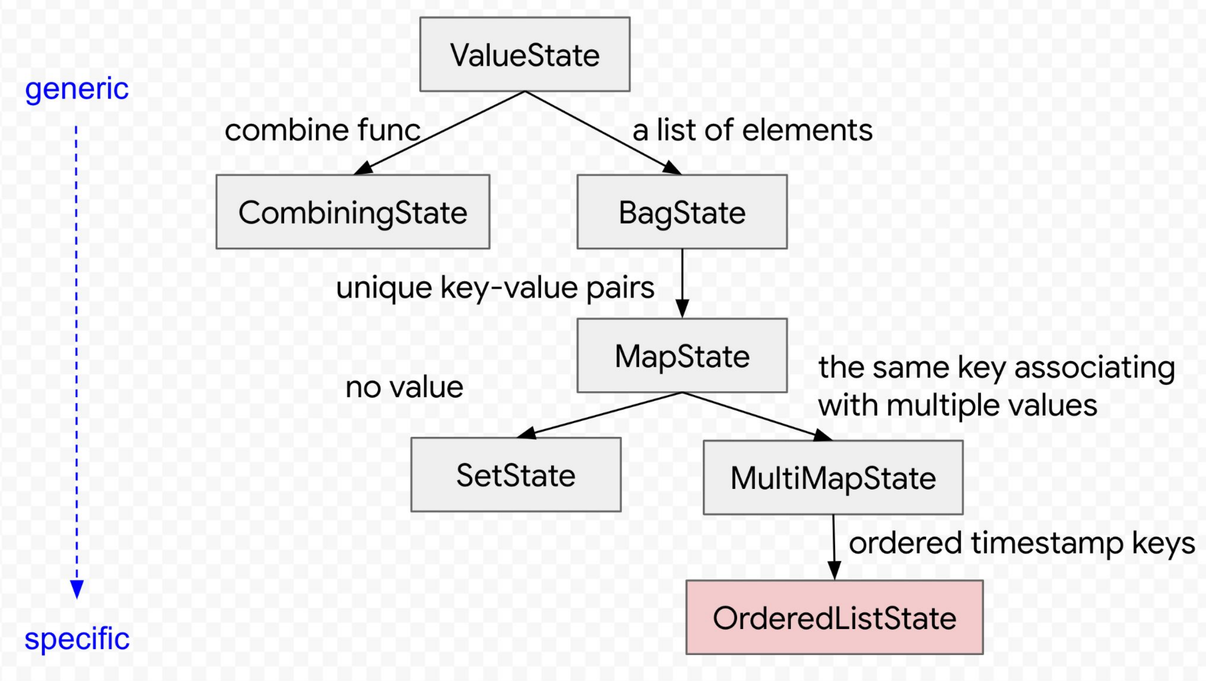
...

Runner

SDK Worker

Choosing the Right State Type

- From generic to specific.
- To store a list of elements, `BagState` will be a better choice than `ValueState`.
- Similarly, to store a list of elements ordered by their timestamps, `OrderedListState` will be the best choice.

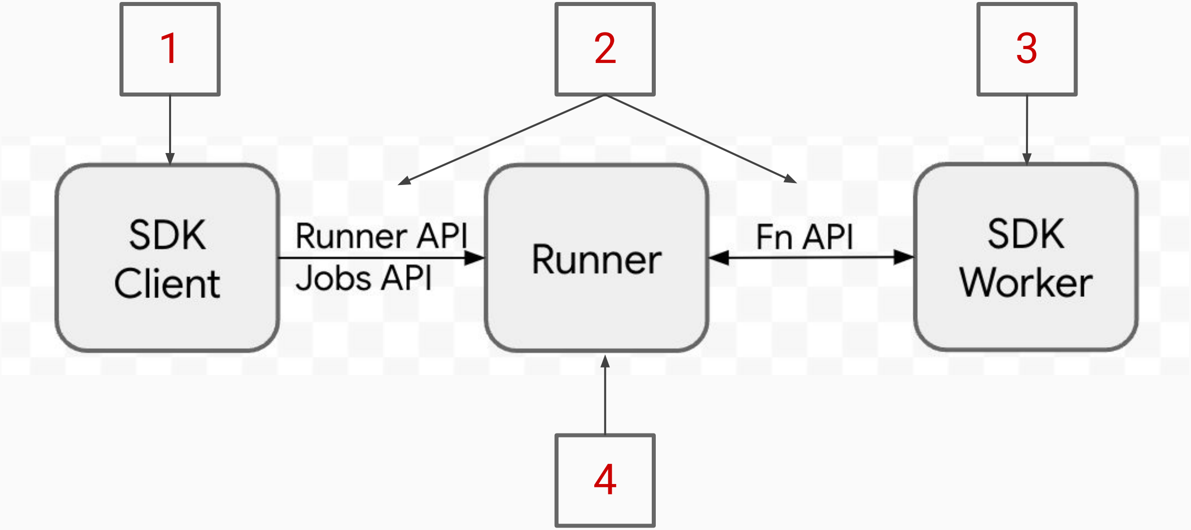


OrderedListState APIs

- `add()`: insert key-value pairs into the state where keys are int64 or timestamp
- `read()`: read every element
- `clear()`: remove every element

- `read_range(lo, hi)`: read elements whose key falls into range [lo, hi) from the state
- `clear_range(lo, hi)`: remove elements whose key falls into range [lo, hi) from the state

Implementation



1. Client APIs

2. Protos
   a. RunnerAPI (StateSpec)
   b. FnAPI (StateKey, StateGetRequest, etc.)

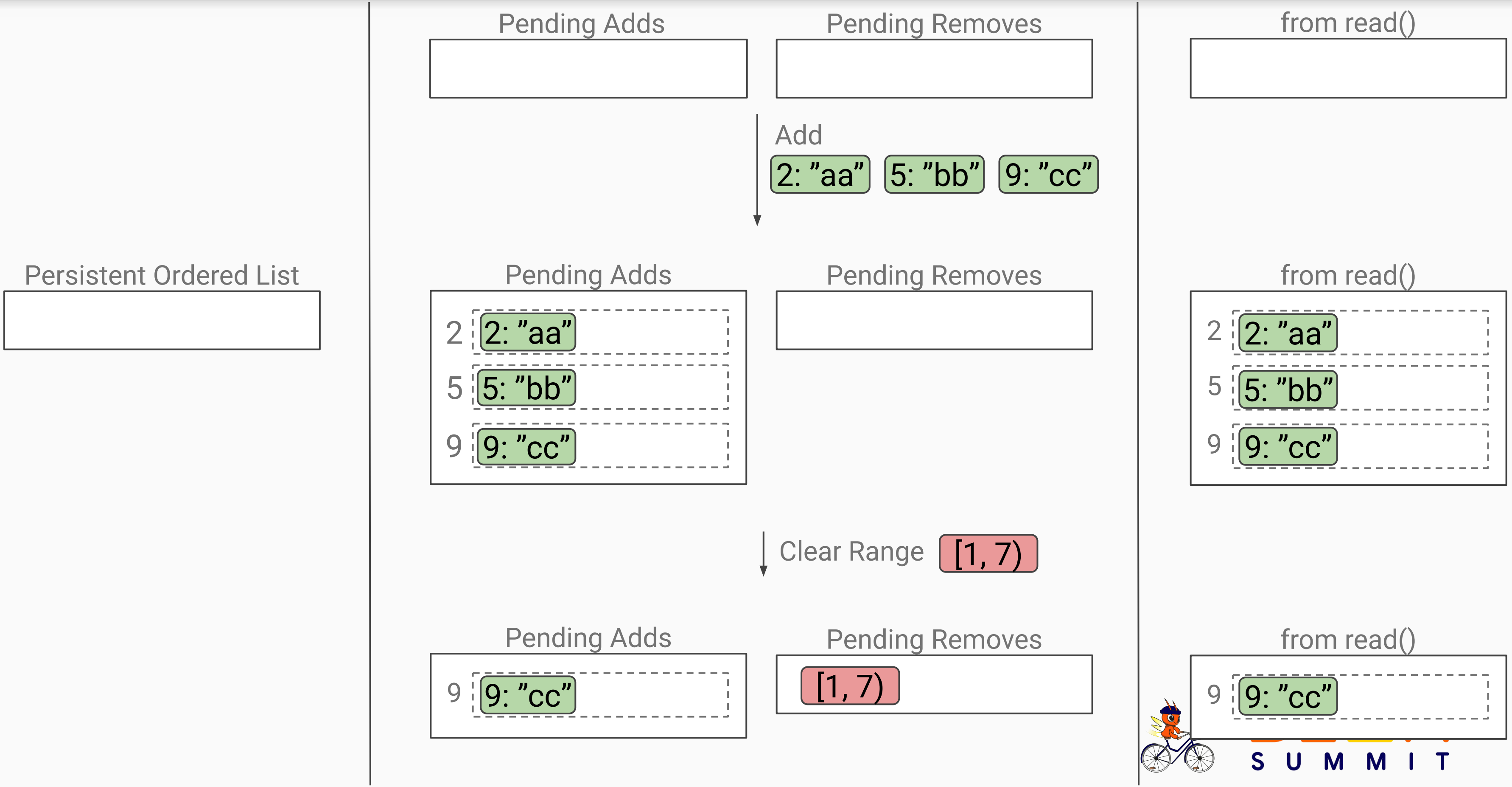
3. SDK implementation

the actual implementation that is used to replace the state placeholder when workers calling DoFns

4. Runner

stores states and responds to state requests

Simple Walkthrough (When Persistent Ordered List is empty)



Persistent Ordered List

Pending Adds

Pending Removes

from read()

Add

2: "aa" 5: "bb" 9: "cc"

Pending Adds

2 2: "aa"

5 5: "bb"

9 9: "cc"

Pending Removes

from read()

2 2: "aa"

5 5: "bb"

9 9: "cc"

Clear Range [1, 7)

Pending Adds

9 9: "cc"

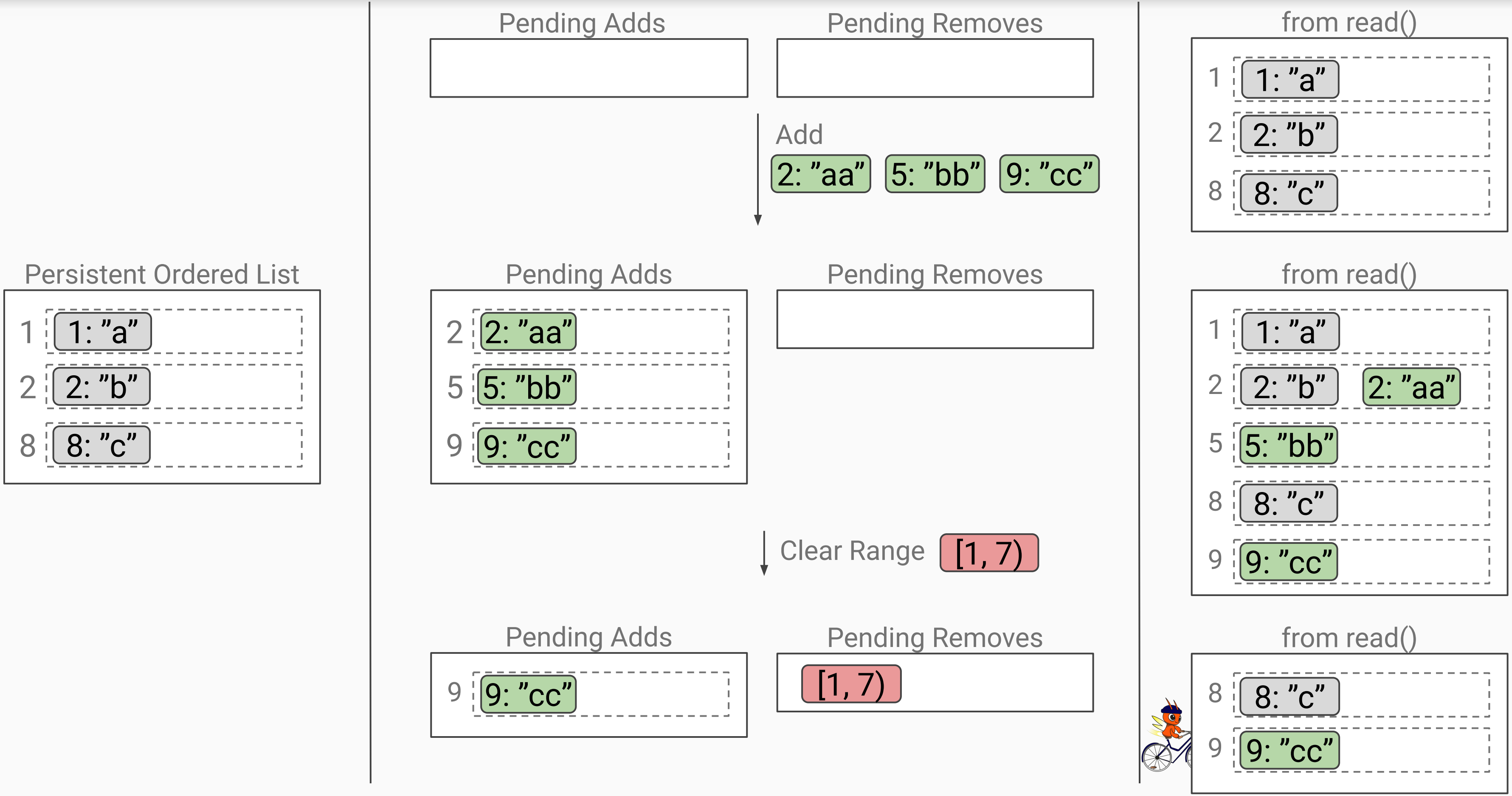
Pending Removes

[1, 7)

from read()

9 9: "cc"

Simple Walkthrough (When Persistent Ordered List is NOT empty)



Pending Adds

Pending Removes

Add 2: "aa" 5: "bb" 9: "cc"

from read()

1 1: "a"

2 2: "b"

8 8: "c"

Persistent Ordered List

1 1: "a"

2 2: "b"

8 8: "c"

Pending Adds

2 2: "aa"

5 5: "bb"

9 9: "cc"

Pending Removes

from read()

1 1: "a"

2 2: "b" 2: "aa"

5 5: "bb"

8 8: "c"

9 9: "cc"

Clear Range [1, 7)

Pending Adds

9 9: "cc"

Pending Removes

[1, 7)

from read()

8 8: "c"

9 9: "cc"

Pseudo Code

```
func add(k, v)
    add (k,v) to pending_adds
```

```
func clear_range(lo, hi)
    remove (k, v) from pending_adds where k in [lo, hi)
    add [lo, hi) to pending_removes
```

```
func read_range(lo, hi)
    pending_adds_in_range = {any items in pending_adds whose keys are in [lo, hi)}
    if persistent_items is {}
        return pending_adds_in_range
    else
        persistent_items_in_range_after_removal = {any items in persistent_items whose keys are in [lo, hi), but not in any ranges of pending_removes}
```

Demo

https://github.com/shunping/beam-summit-2024-demo/

References & Further Reading

- Portable OrderedListState (Design Doc) By Boyuan Zhang
- So, You Want to Write a Beam SDK? By Robert Bradshaw
- Stateful processing with Apache Beam By Kenneth Knowles
- Github PRs
  - OrderedListState support in Java: https://github.com/apache/beam/pull/30317
  - OrderedListState support in Python (WIP): https://github.com/apache/beam/pull/32326

Thank you!

Questions?

Shunping Huang

shunping@google.com

Runner Capability Matrix for States

State Type	Java Direct Runner	Python Direct Runner (Legacy)	Python FnAPI Runner	Go Prism Runner	Dataflow Legacy Runner	Dataflow Runner V2
ValueState	✅	✅	✅	✅	✅	✅
CombiningState	✅	✅	✅	✅	✅	✅
BagState	✅	✅	✅	✅	✅	✅
SetState	✅	✅	✅	✅	✅	
MapState	✅			✅	✅	
MultimapState	✅			✅	✅	
OrderedListState	✅		⭐		✅	⭐