Keyed State

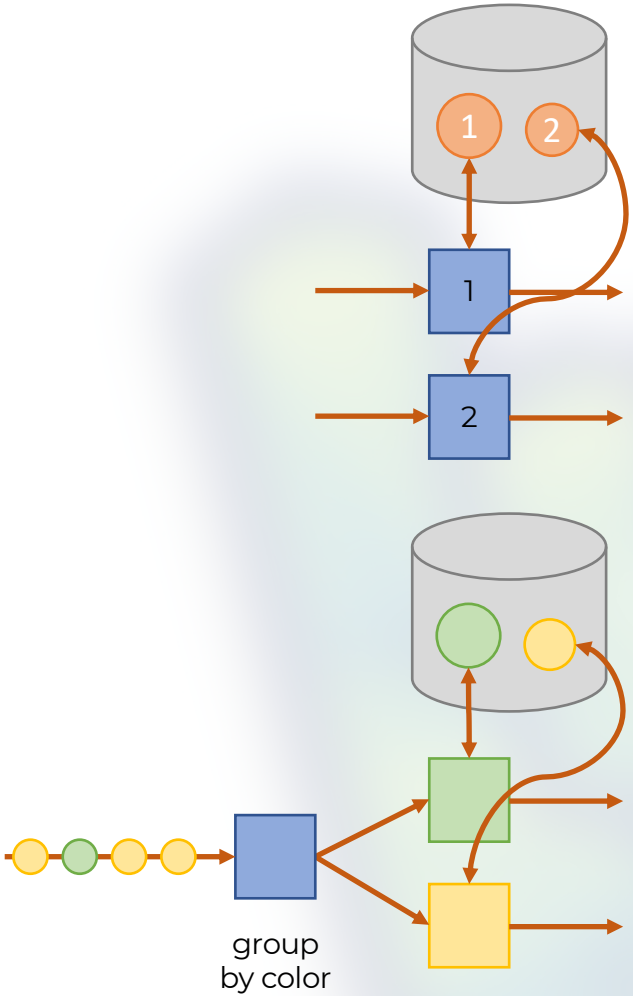
State Types

Operator state

- private to an operator task
- cannot be read/written by another operator task
- very rarely used

Keyed state

- one instance per key
- used in the vast majority of cases



Managing State

Variables do not scale to a distributed system

State variables

- essentially distributed variables managed by Flink
- need to be initialized in the lifecycle methods of rich functions

Value State

- a single value stored
- can be read (get) and written (update)

```
var stateCounter: ValueState[Long] = _ // a value state per key=userId

override def open(parameters: Configuration): Unit = {
  // initialize all state
  stateCounter = getRuntimeContext // from RichFunction
    .getState(new ValueStateDescriptor[Long]("events-counter", classOf[Long]))
}
```

Managing State

List State

- contains multiple values
- read as a Java iterator
- add an element, add an entire Java list
- overwrite

```
var stateEventsForUser: ListState[ShoppingCartEvent] = _

override def open(parameters: Configuration): Unit =
  stateEventsForUser = getRuntimeContext.getListState(
    new ListStateDescriptor[ShoppingCartEvent](
      "shopping-cart-events",
      classOf[ShoppingCartEvent]
  )
)
```

Map State

- contains a "hash map"
- read as a Java iterator of entries (get)
- check if a key is in the map
- get the value associated to a key
- insert/update the value for a key
- overwrite everything

```
var stateCountsPerEventType: MapState[String, Long] = _

override def open(parameters: Configuration): Unit = {
  stateCountsPerEventType = getRuntimeContext
    .getMapState(
      new MapStateDescriptor[String, Long](
        "per-type-counter",
        classOf[String],
        classOf[Long]
      )
    )
}
```

Flink rocks