Router-based stream processing implemented in P4

Sammy Moseley

Outline

1. Background on stream processing
2. Implementation using P4
3. Next steps

Stream Processing Applications

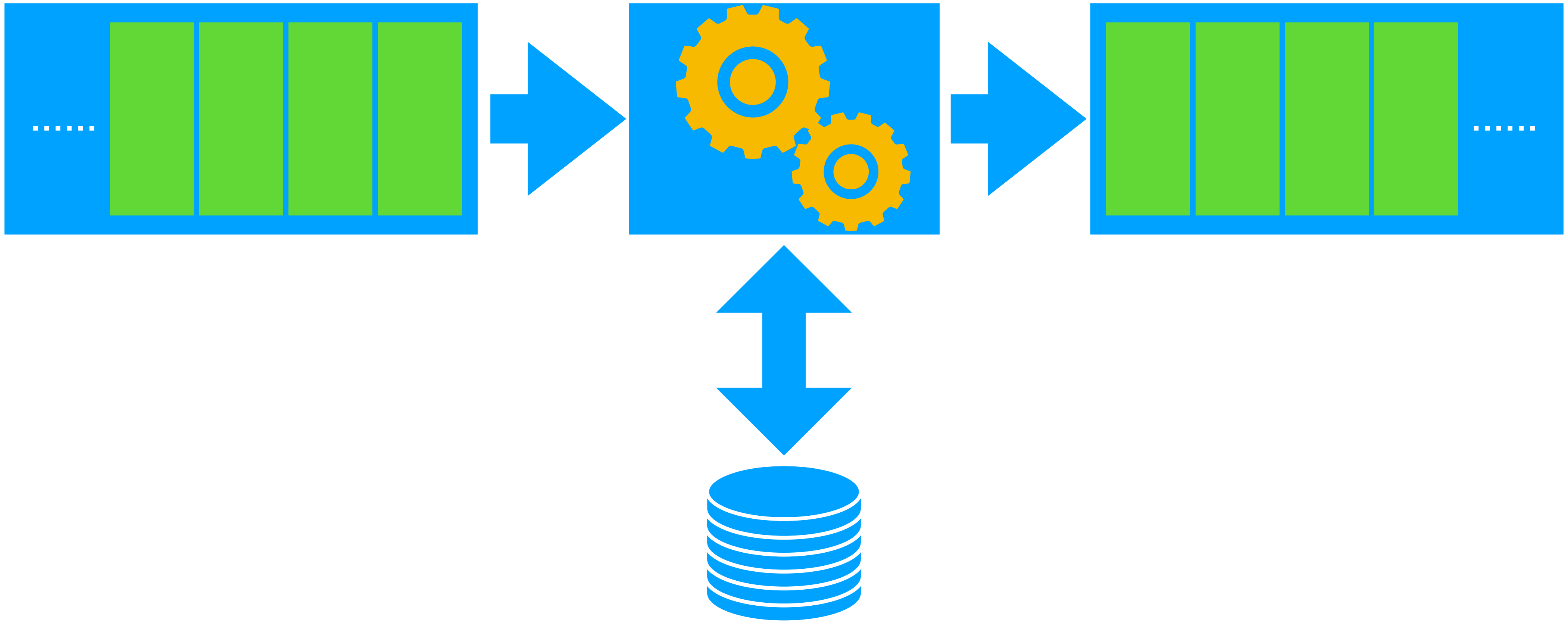
Friend/Follow Requests

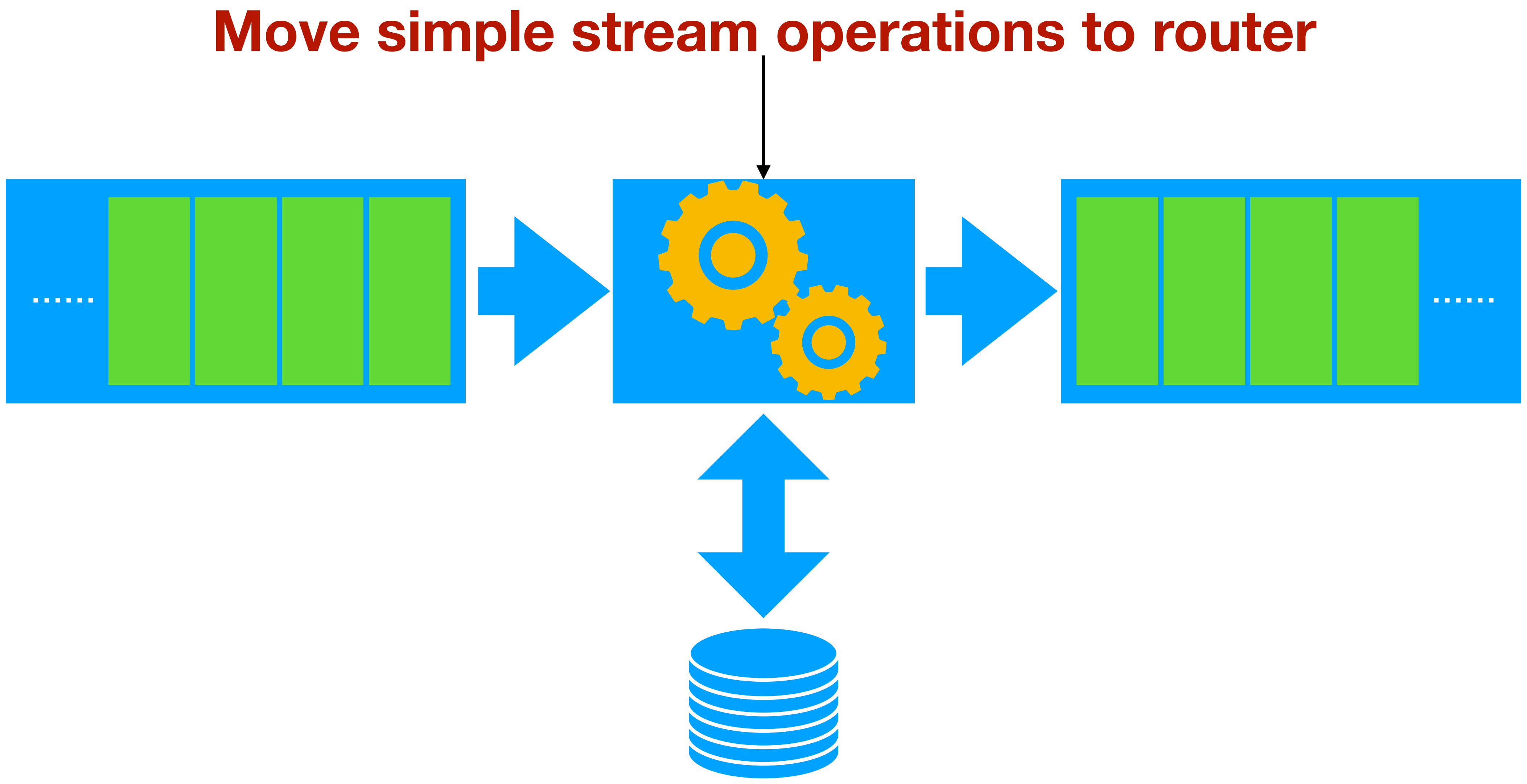
Messaging

Webpage view/post clicks

Payment Transactions

A Quick Background on Stream Processing



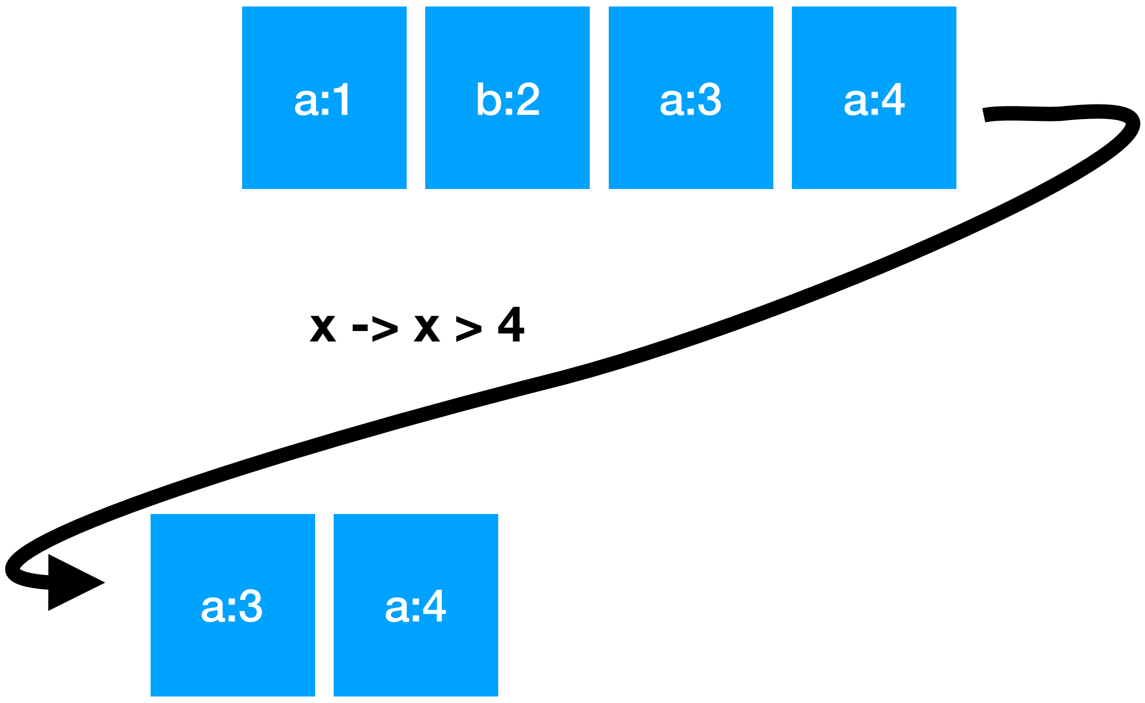

Move simple stream operations to router
......
......

Fundamental stream query operators

1. Filter
2. Map
3. Join
4. Aggregate
5. Zip

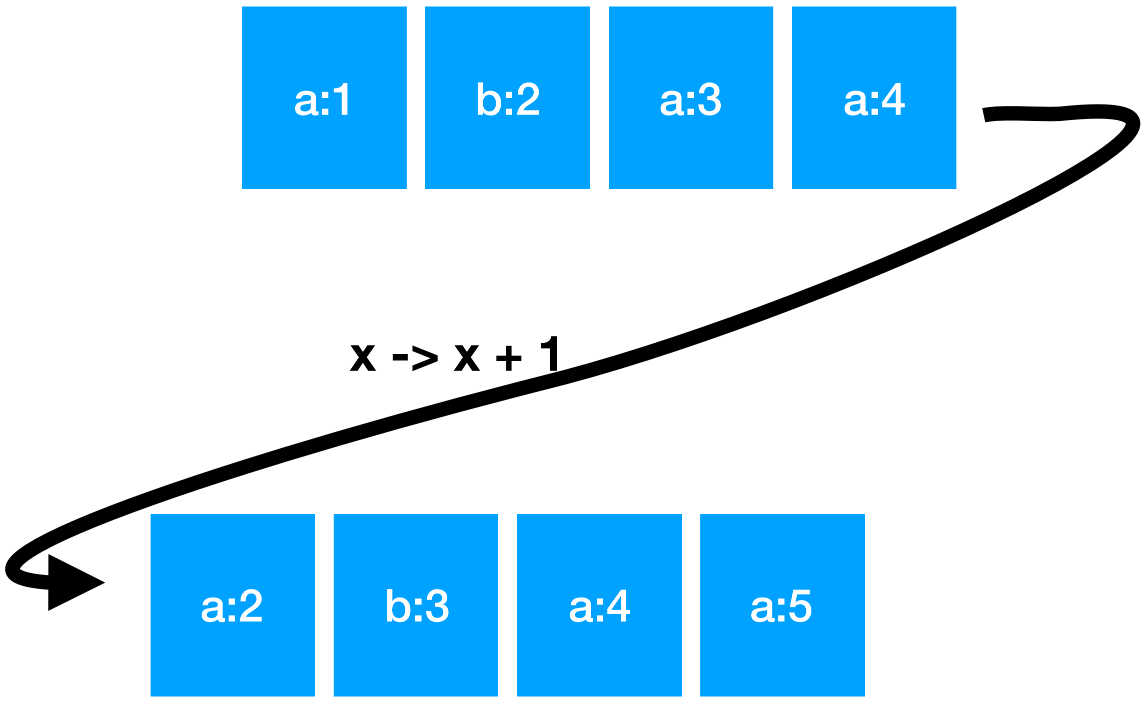
Fundamental stream query operators

1. **Filter**
2. Map
3. Join
4. Aggregate
5. Zip



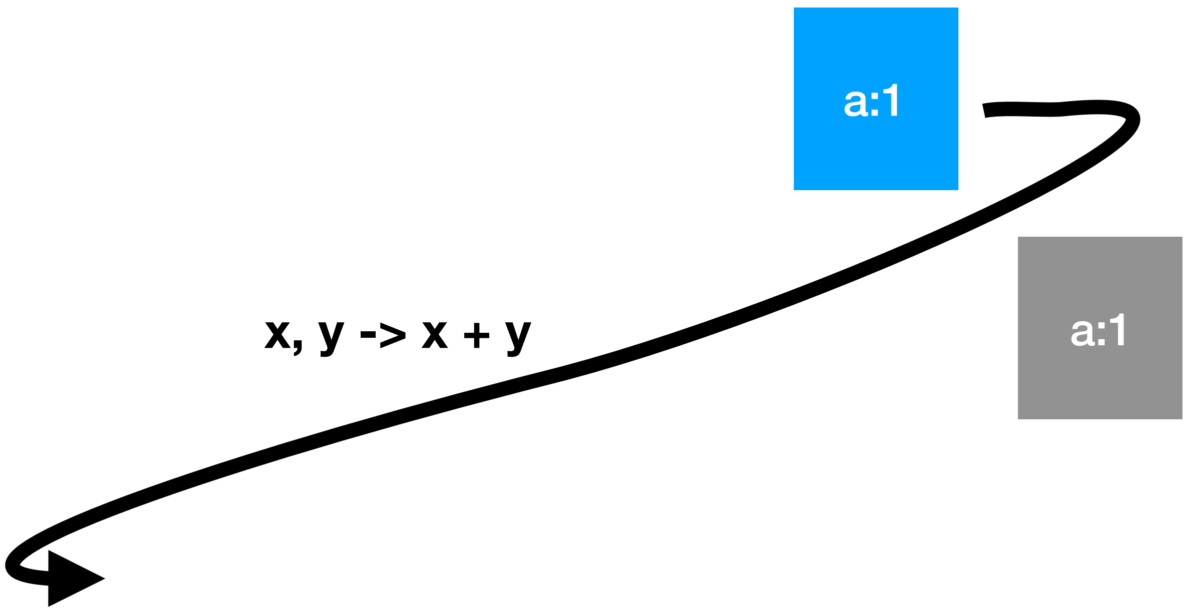
Fundamental stream query operators

1. Filter
2. **Map**
3. Join
4. Aggregate
5. Zip



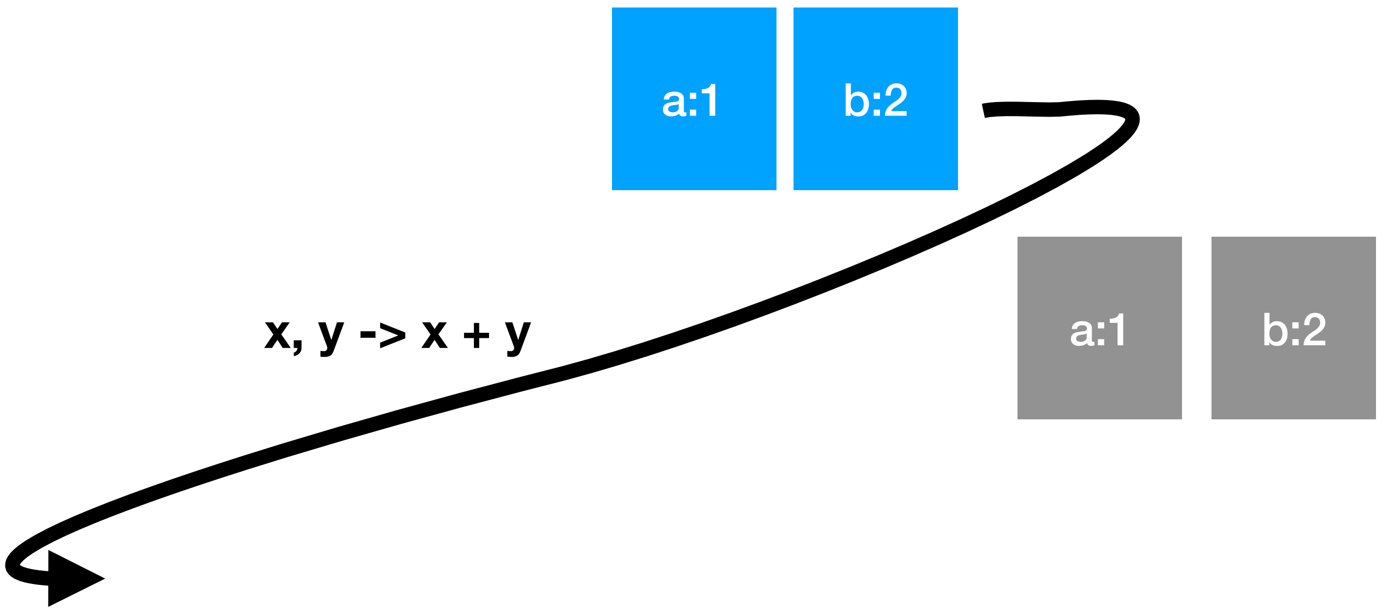
Fundamental stream query operators

1. Filter
2. Map
3. **Join**
4. Aggregate
5. Zip



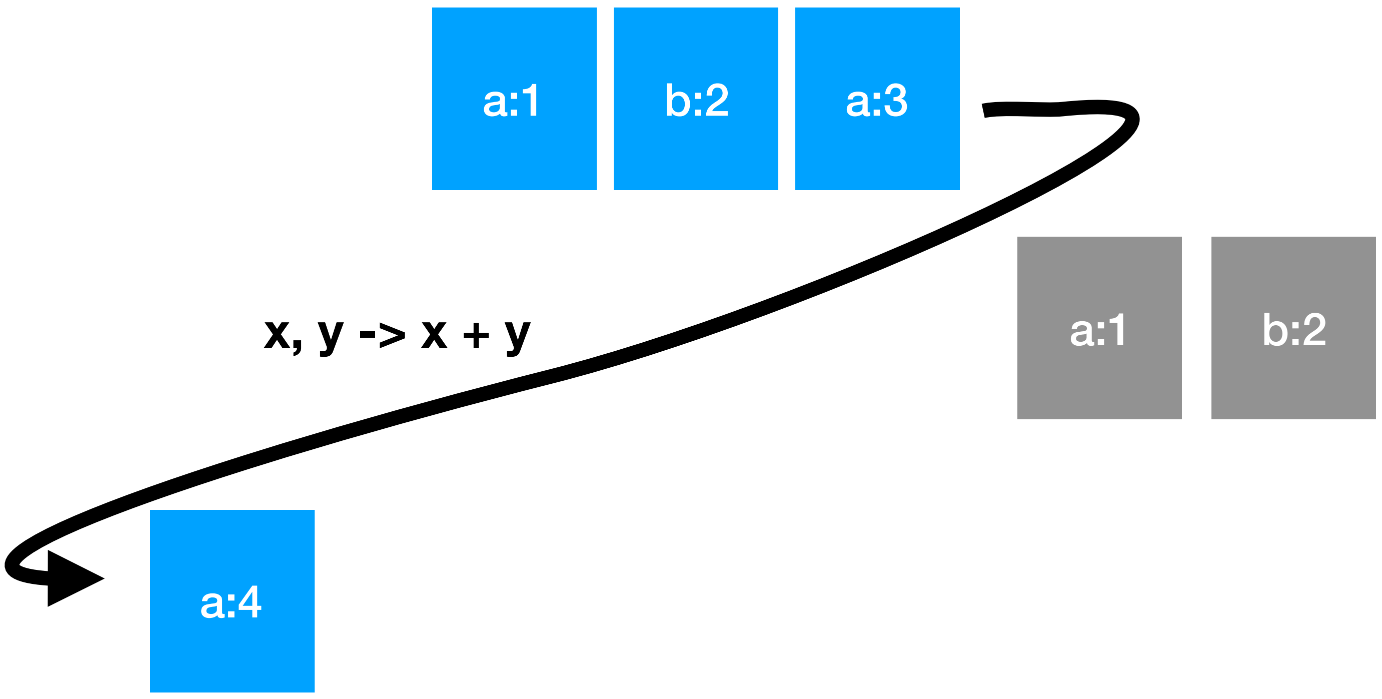
Fundamental stream query operators

1. Filter
2. Map
3. **Join**
4. Aggregate
5. Zip



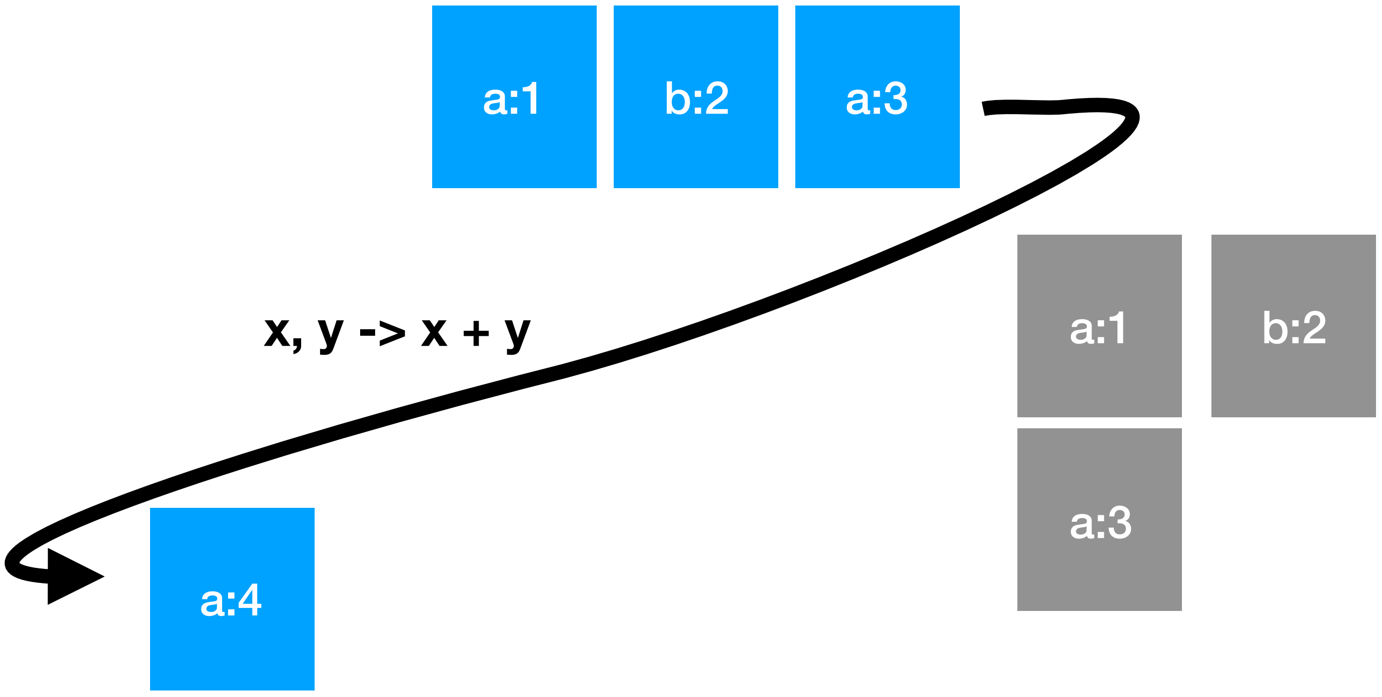
Fundamental stream query operators

1. Filter
2. Map
3. **Join**
4. Aggregate
5. Zip



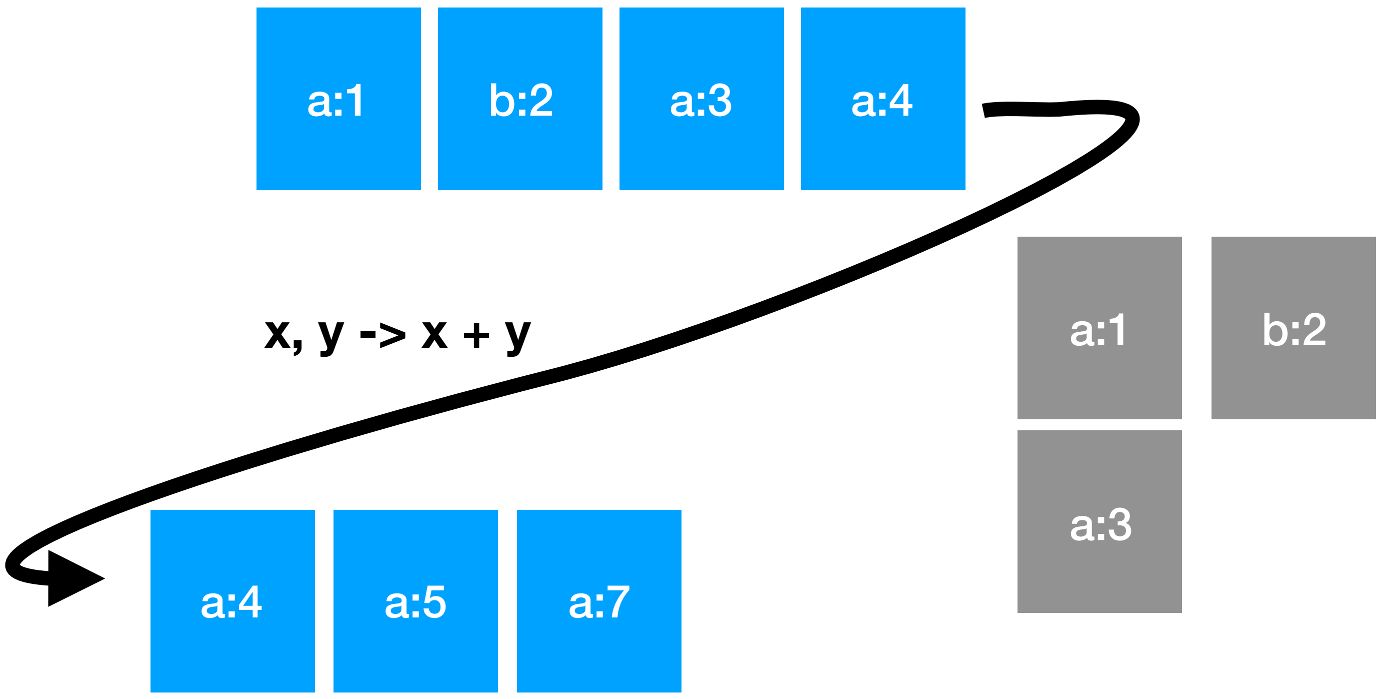
Fundamental stream query operators

1. Filter
2. Map
3. **Join**
4. Aggregate
5. Zip



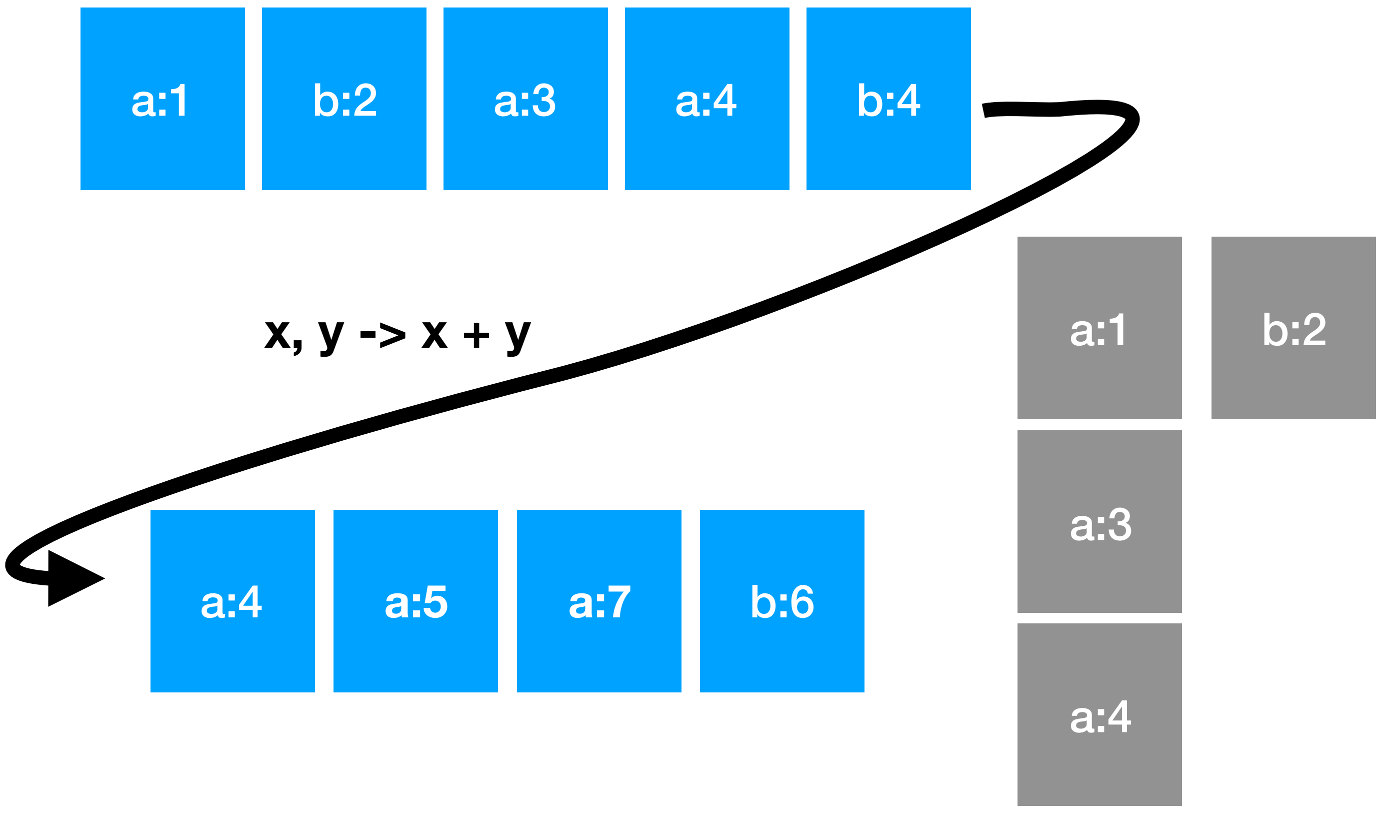
Fundamental stream query operators

1. Filter
2. Map
3. **Join**
4. Aggregate
5. Zip



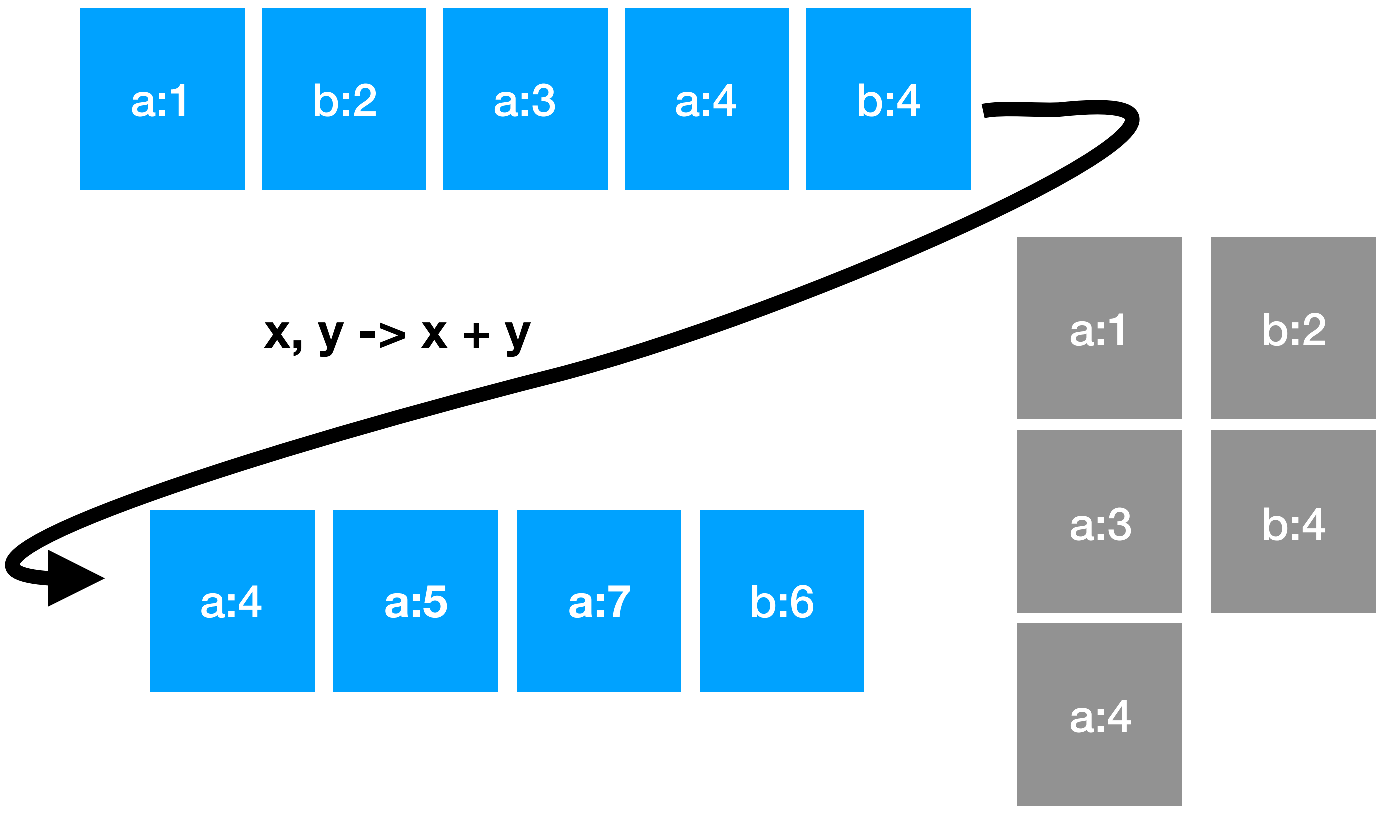
Fundamental stream query operators

1. Filter
2. Map
3. **Join**
4. Aggregate
5. Zip



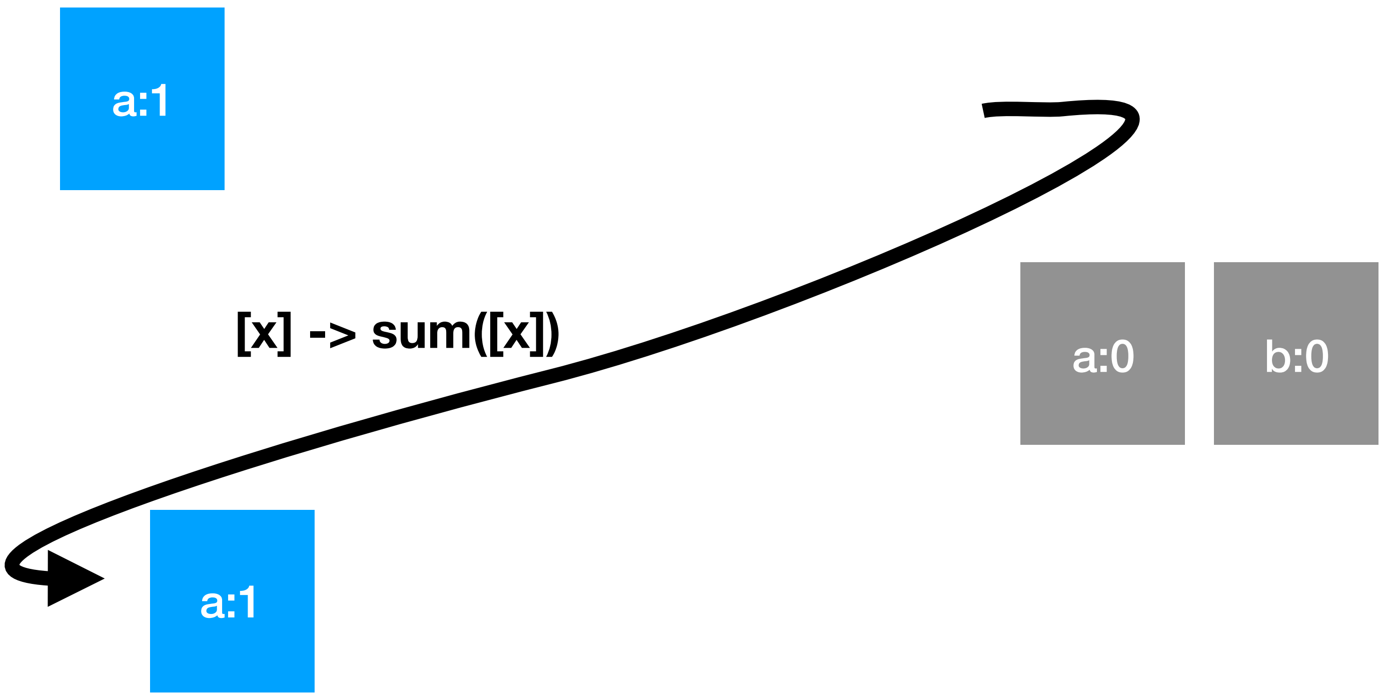
Fundamental stream query operators

1. Filter
2. Map
3. **Join**
4. Aggregate
5. Zip



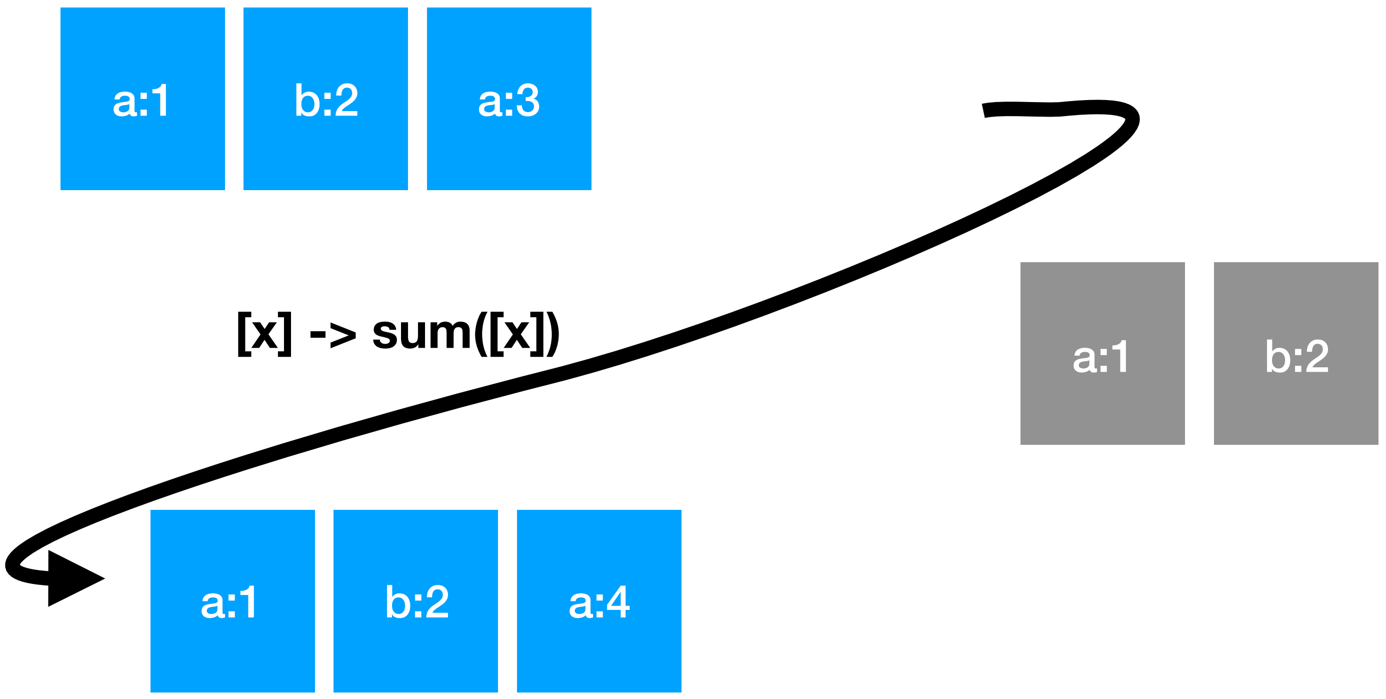
Fundamental stream query operators

1. Filter
2. Map
3. Join
4. **Aggregate**
5. Zip



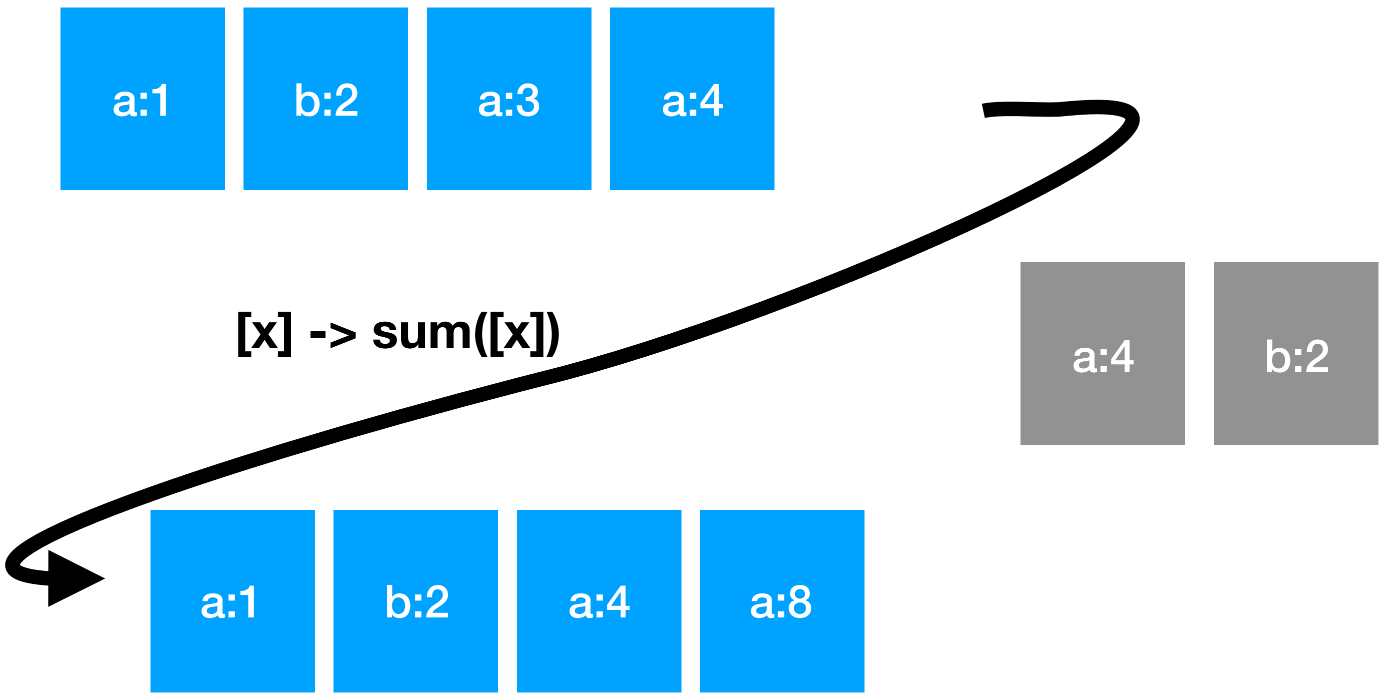
Fundamental stream query operators

1. Filter
2. Map
3. Join
4. **Aggregate**
5. Zip



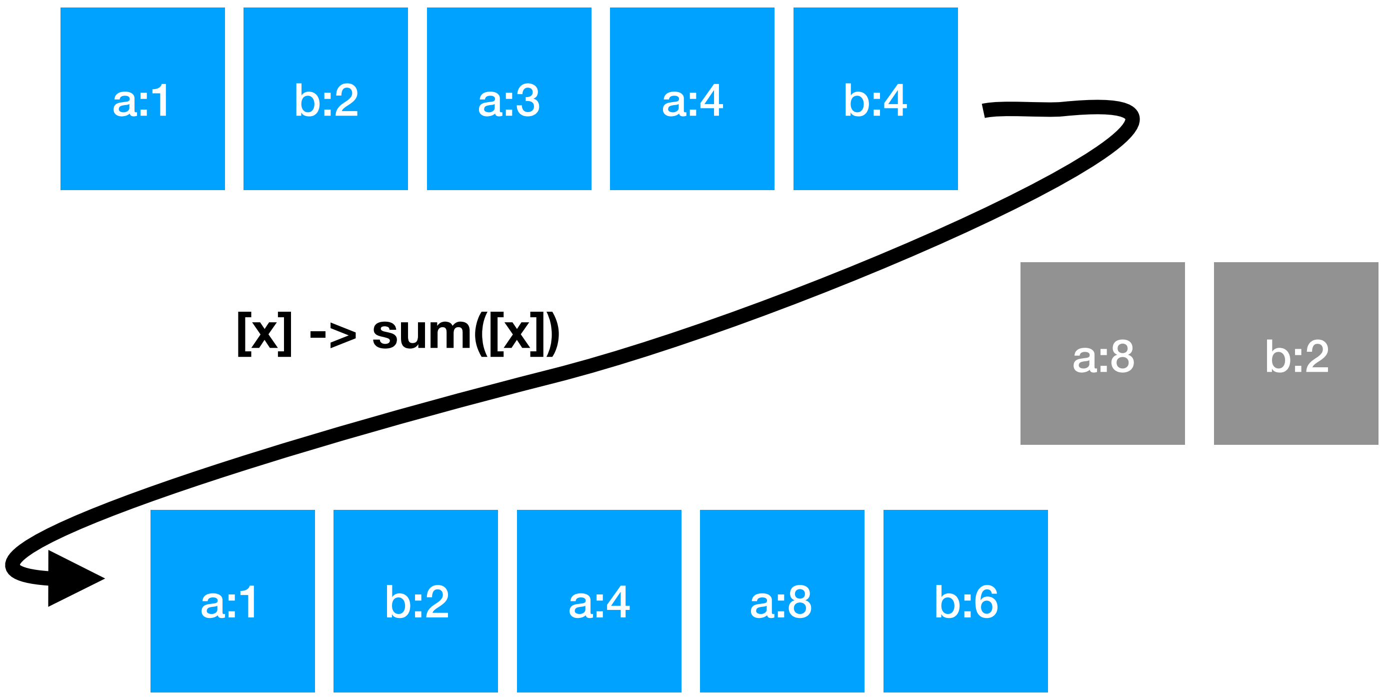
Fundamental stream query operators

1. Filter
2. Map
3. Join
4. **Aggregate**
5. Zip



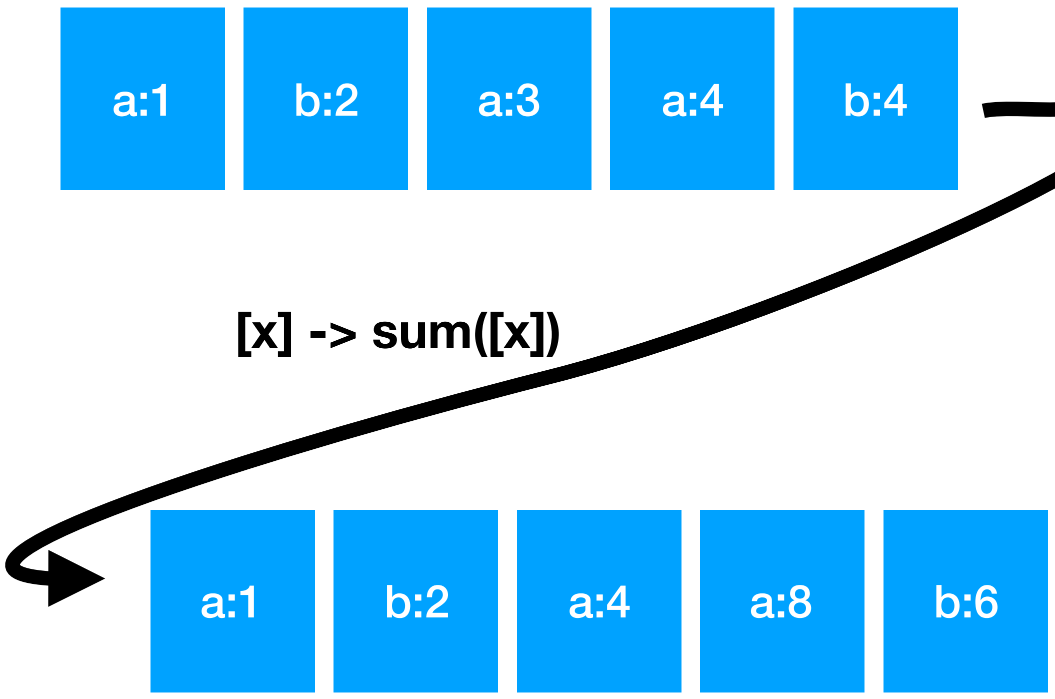
Fundamental stream query operators

1. Filter
2. Map
3. Join
4. **Aggregate**
5. Zip



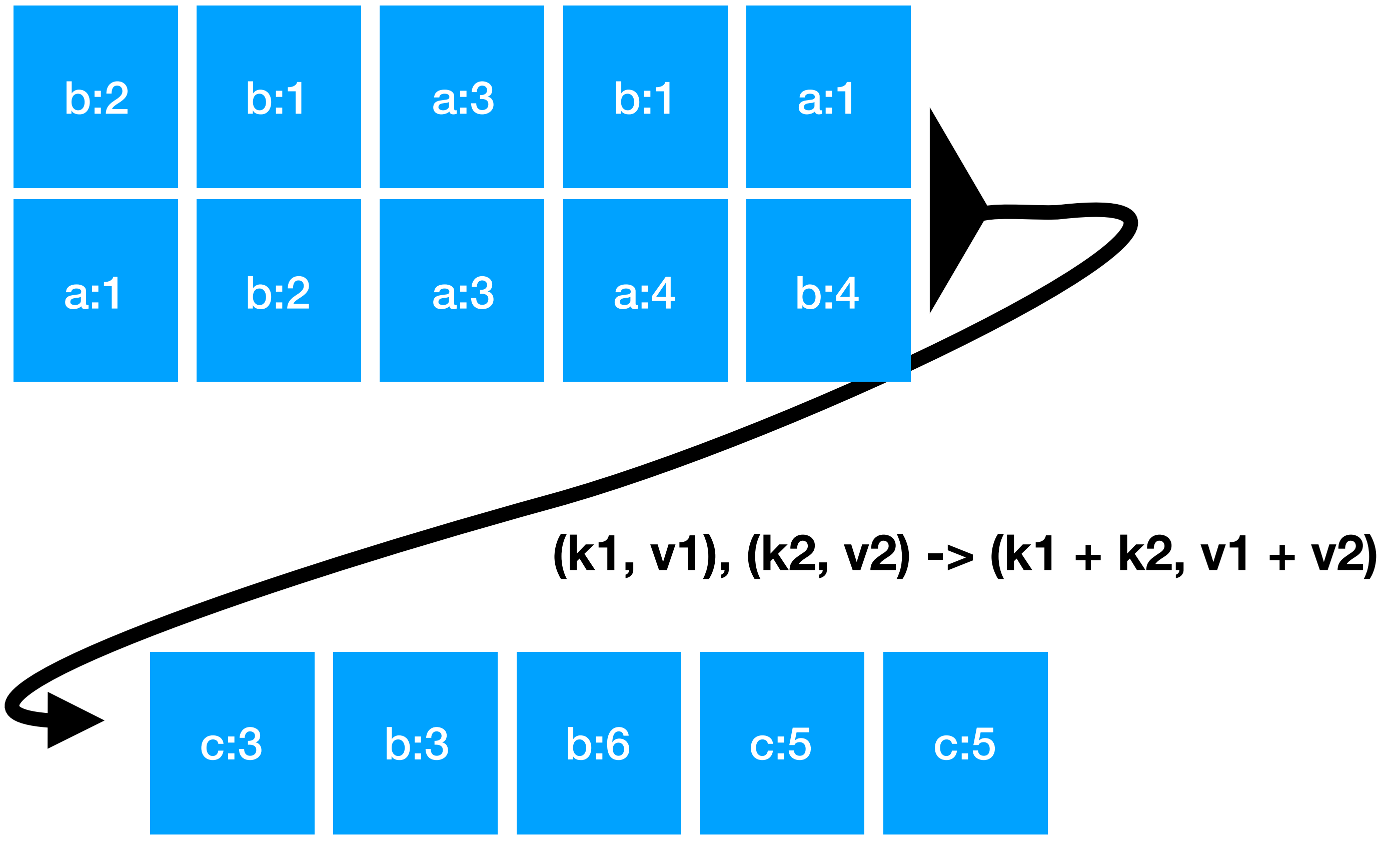
Fundamental stream query operators

1. Filter
2. Map
3. Join
4. **Aggregate**
5. Zip



Fundamental stream query operators

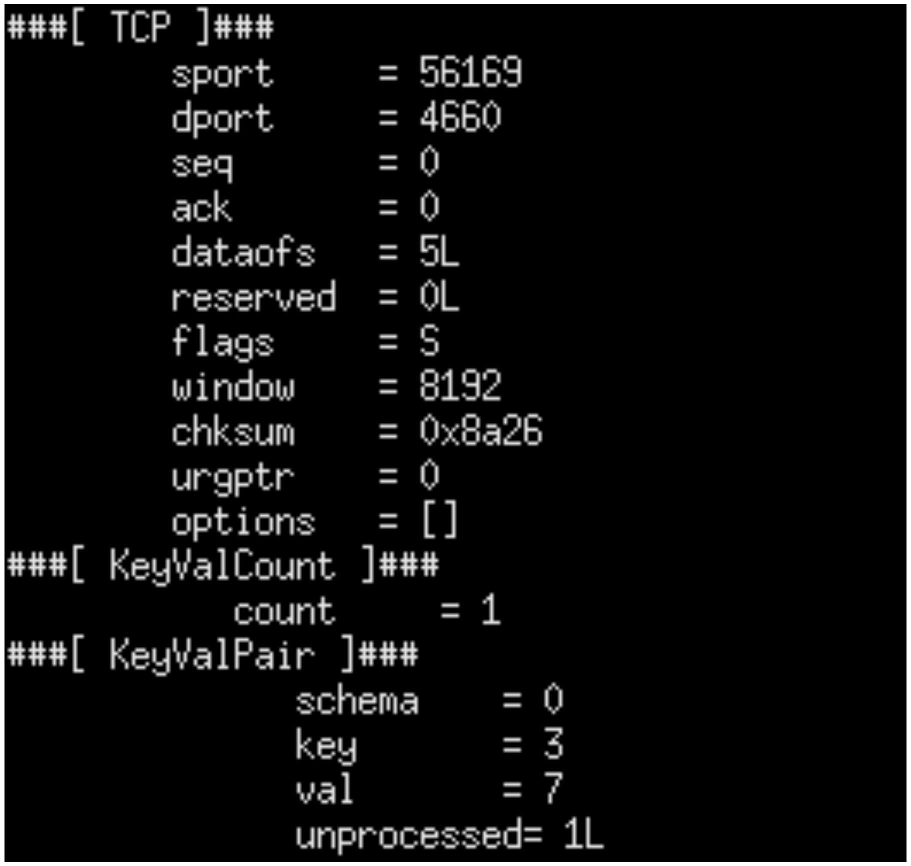
1. Filter
2. Map
3. Join
4. Aggregate
5. **Zip**



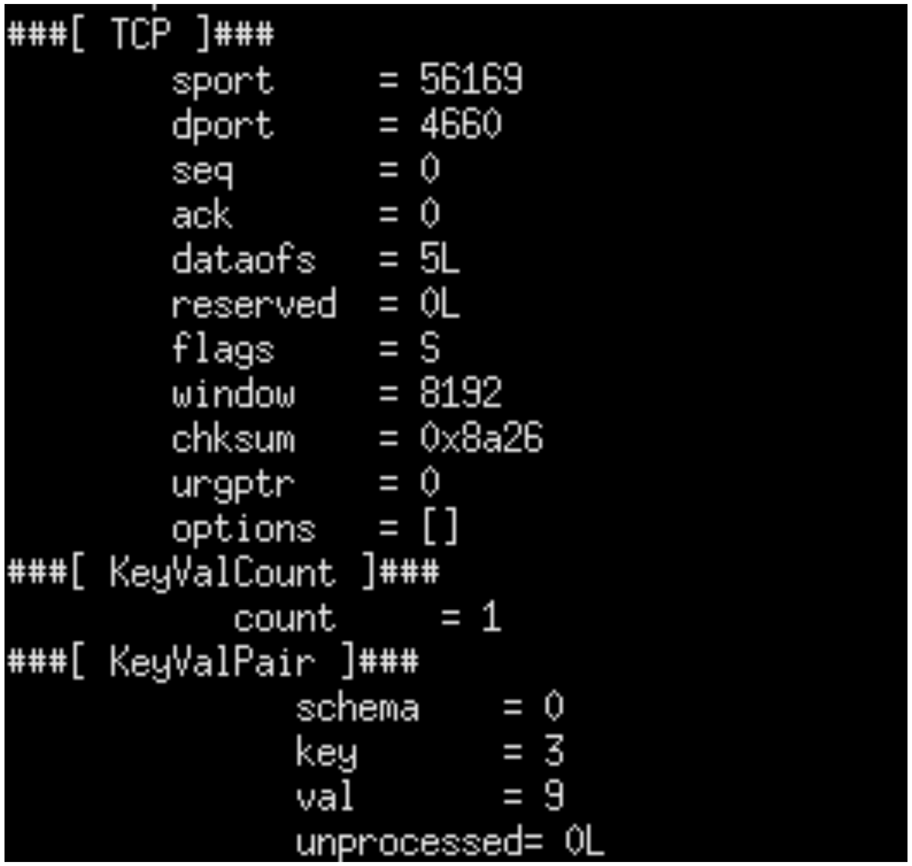
Fundamental stream query operators

1. Filter
2. **Map**
3. **Join**
4. **Aggregate**
5. Zip

Map operator in action

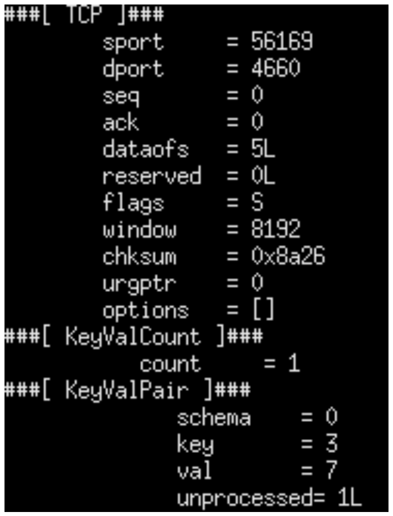


```
###[ TCP ]###
        sport     = 56169
        dport     = 4660
        seq       = 0
        ack       = 0
        dataofs   = 5L
        reserved  = 0L
        flags     = S
        window    = 8192
        chksum    = 0x8a26
        urgptr    = 0
        options   = []
###[ KeyValCount ]###
           count     = 1
###[ KeyValPair ]###
              schema    = 0
              key       = 3
              val       = 7
              unprocessed= 1L
```

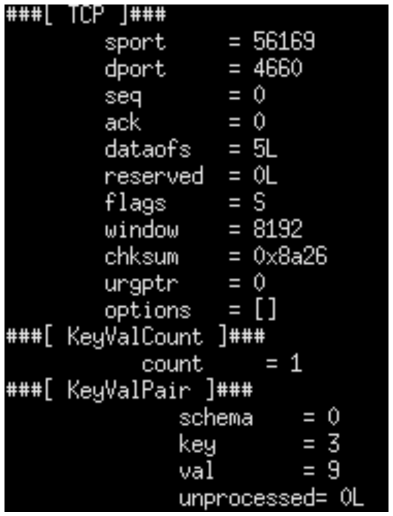


```
###[ TCP ]###
        sport     = 56169
        dport     = 4660
        seq       = 0
        ack       = 0
        dataofs   = 5L
        reserved  = 0L
        flags     = S
        window    = 8192
        chksum    = 0x8a26
        urgptr    = 0
        options   = []
###[ KeyValCount ]###
           count     = 1
###[ KeyValPair ]###
              schema    = 0
              key       = 3
              val       = 9
              unprocessed= 0L
```

Aggregate operator in action

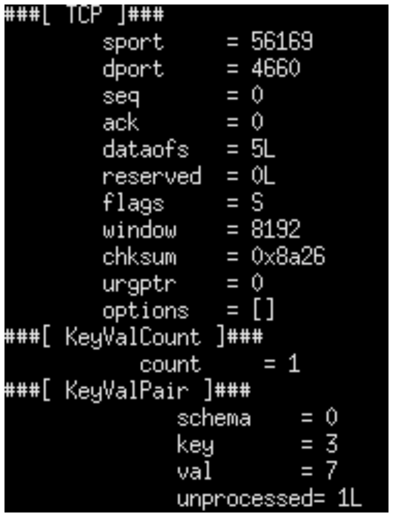


```
###[ TCP ]###
        sport     = 56169
        dport     = 4660
        seq       = 0
        ack       = 0
        dataofs   = 5L
        reserved  = 0L
        flags     = S
        window    = 8192
        chksum    = 0x8a26
        urgptr    = 0
        options   = []
###[ KeyValCount ]###
           count     = 1
###[ KeyValPair ]###
              schema    = 0
              key       = 3
              val       = 7
              unprocessed= 1L
```

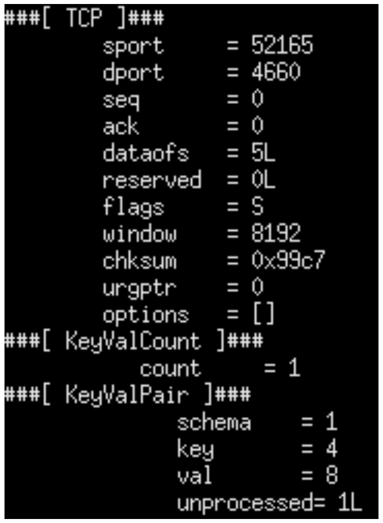


```
###[ TCP ]###
        sport     = 56169
        dport     = 4660
        seq       = 0
        ack       = 0
        dataofs   = 5L
        reserved  = 0L
        flags     = S
        window    = 8192
        chksum    = 0x8a26
        urgptr    = 0
        options   = []
###[ KeyValCount ]###
           count     = 1
###[ KeyValPair ]###
              schema    = 0
              key       = 3
              val       = 9
              unprocessed= 0L
```

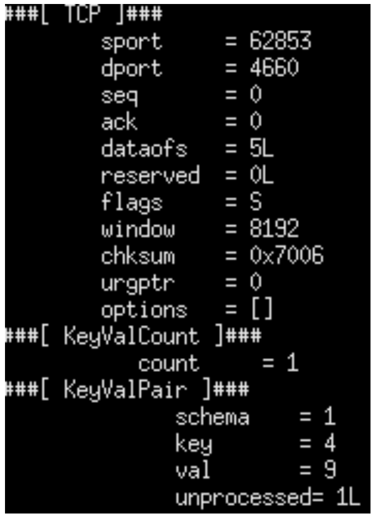
Aggregate operator in action



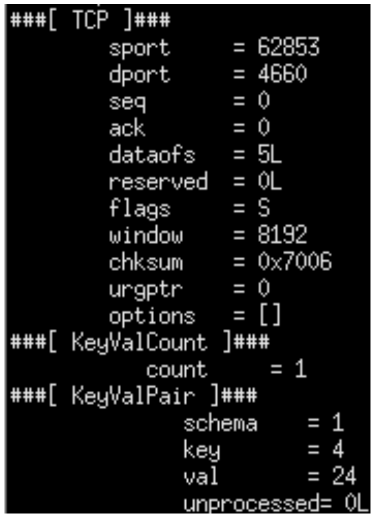
```
###[ TCP ]###
        sport     = 56169
        dport     = 4660
        seq       = 0
        ack       = 0
        dataofs   = 5L
        reserved  = 0L
        flags     = S
        window    = 8192
        chksum    = 0x8a26
        urgptr    = 0
        options   = []
###[ KeyValCount ]###
           count     = 1
###[ KeyValPair ]###
              schema    = 0
              key       = 3
              val       = 7
              unprocessed= 1L
```



```
###[ TCP ]###
        sport     = 52165
        dport     = 4660
        seq       = 0
        ack       = 0
        dataofs   = 5L
        reserved  = 0L
        flags     = S
        window    = 8192
        chksum    = 0x99c7
        urgptr    = 0
        options   = []
###[ KeyValCount ]###
           count     = 1
###[ KeyValPair ]###
              schema    = 1
              key       = 4
              val       = 8
              unprocessed= 1L
```

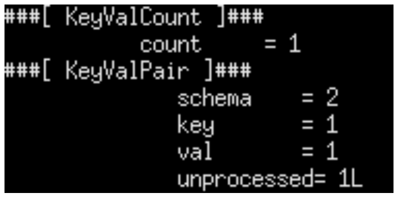


```
###[ TCP ]###
        sport     = 62853
        dport     = 4660
        seq       = 0
        ack       = 0
        dataofs   = 5L
        reserved  = 0L
        flags     = S
        window    = 8192
        chksum    = 0x7006
        urgptr    = 0
        options   = []
###[ KeyValCount ]###
           count     = 1
###[ KeyValPair ]###
              schema    = 1
              key       = 4
              val       = 9
              unprocessed= 1L
```



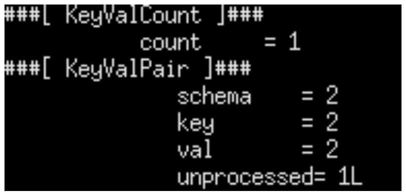
```
###[ TCP ]###
        sport     = 62853
        dport     = 4660
        seq       = 0
        ack       = 0
        dataofs   = 5L
        reserved  = 0L
        flags     = S
        window    = 8192
        chksum    = 0x7006
        urgptr    = 0
        options   = []
###[ KeyValCount ]###
           count     = 1
###[ KeyValPair ]###
              schema    = 1
              key       = 4
              val       = 24
              unprocessed= 0L
```

Join operator in action



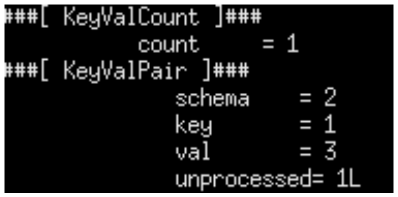
```
###[ KeyValCount ]###
          count     = 1
###[ KeyValPair ]###
             schema    = 2
             key       = 1
             val       = 1
             unprocessed= 1L
```

1:1



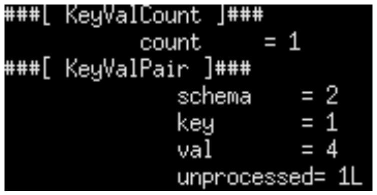
```
###[ KeyValCount ]###
          count     = 1
###[ KeyValPair ]###
             schema    = 2
             key       = 2
             val       = 2
             unprocessed= 1L
```

2:2



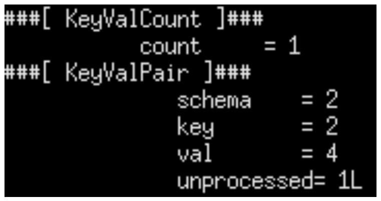
```
###[ KeyValCount ]###
          count     = 1
###[ KeyValPair ]###
             schema    = 2
             key       = 1
             val       = 3
             unprocessed= 1L
```

1:3



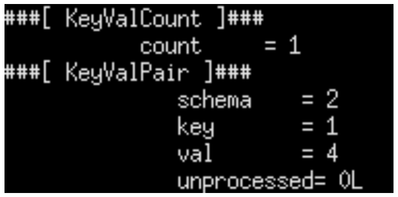
```
###[ KeyValCount ]###
          count     = 1
###[ KeyValPair ]###
             schema    = 2
             key       = 1
             val       = 4
             unprocessed= 1L
```

1:4



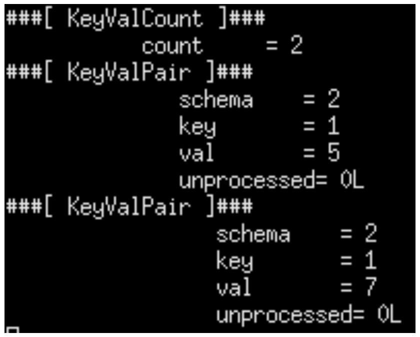
```
###[ KeyValCount ]###
          count     = 1
###[ KeyValPair ]###
             schema    = 2
             key       = 2
             val       = 4
             unprocessed= 1L
```

2:4



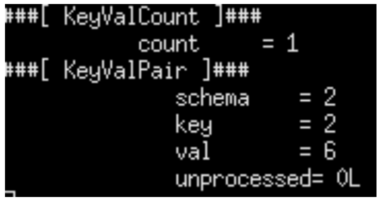
```
###[ KeyValCount ]###
          count     = 1
###[ KeyValPair ]###
             schema    = 2
             key       = 1
             val       = 4
             unprocessed= 0L
```

1:4



```
###[ KeyValCount ]###
          count     = 2
###[ KeyValPair ]###
             schema    = 2
             key       = 1
             val       = 5
             unprocessed= 0L
###[ KeyValPair ]###
                schema    = 2
                key       = 1
                val       = 7
                unprocessed= 0L
```

1:5, 1:7



```
###[ KeyValCount ]###
          count     = 1
###[ KeyValPair ]###
             schema    = 2
             key       = 2
             val       = 6
             unprocessed= 0L
```

2:6

Next steps

- Implement Stream QL compiler
- Compare to python stream processing implementation