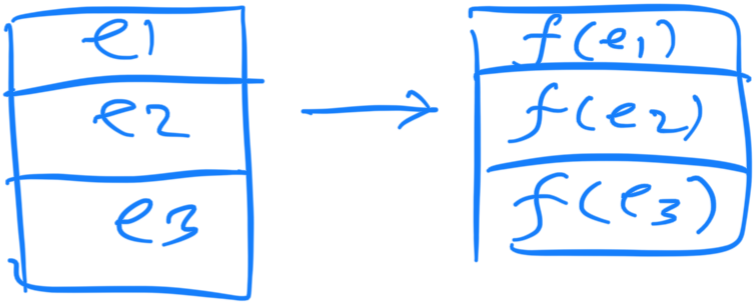
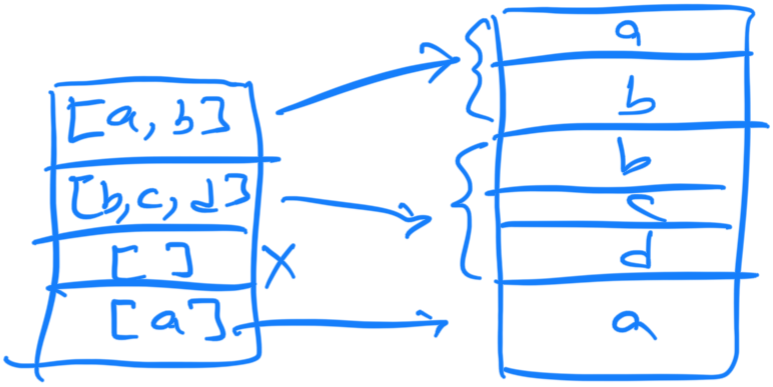
Summary of Mappers & Reducers

Mappers
- map() 1-to-1
- flatMap() 1-to-M
- mapPartitions(): M-to-1

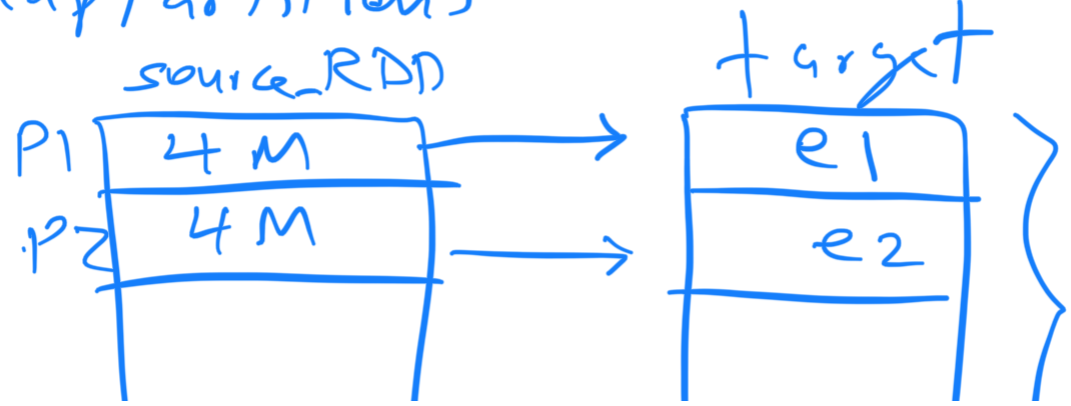
map()

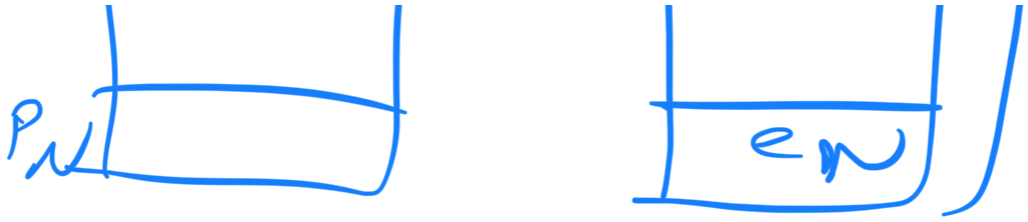


flatMap



mapPartitions





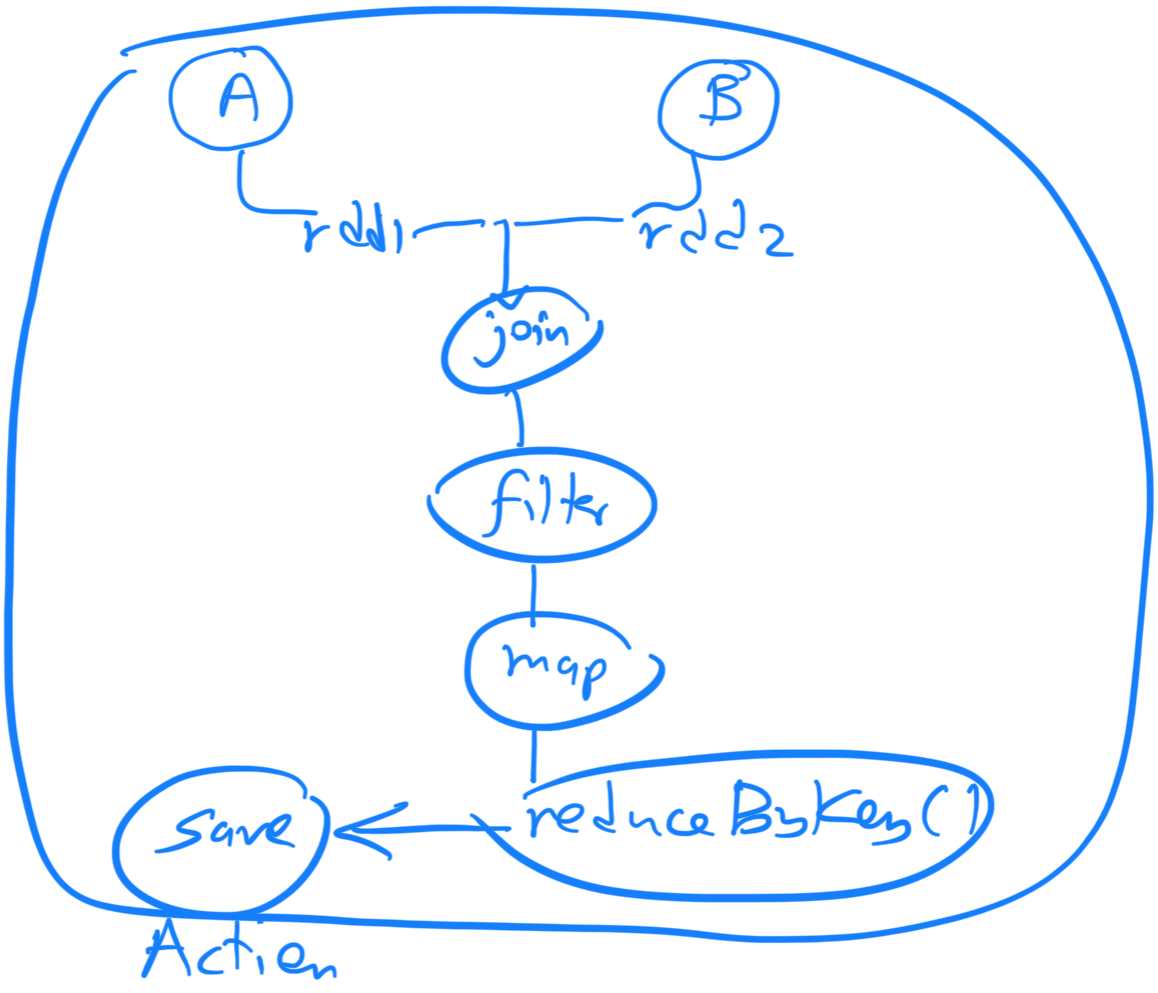
# of elements: 80,000,000,000

N = 20,000

each Partition will have ≈ 4000,000

target = source.mapPartitions(func)

Directed Acyclic Graph



Lazy Binding

R(a, b)

a	b
a_1	b_1
a_2	b_2
a_3	b_3 X

S(b, c)

b	c
b_1	c_1
b_1	c_2
b_2	d_1
b_2	d_2
b_4	d_9 X

join(R, S) =

(a_1,	b_1,	c_1)
(a_1,	b_1,	c_2)
(a_2,	b_2,	d_1)
(a_2,	b_2,	d_2)

← inner join

① map R

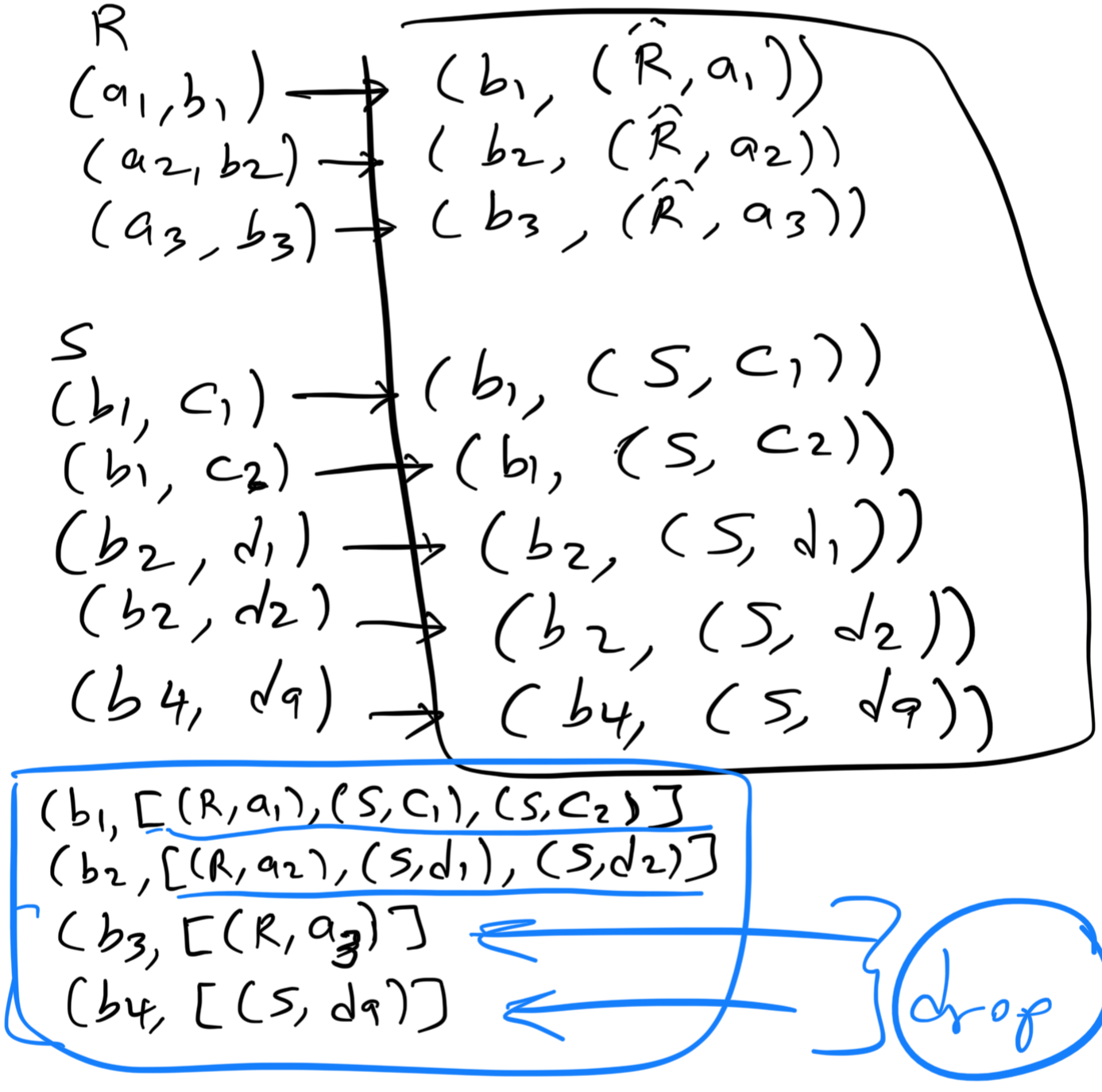
(K: record number, ignored; V: actual record)

```
map(K, V) {
  tokens = v.split(",")
   key = tokens[1]
   value = tokens[0]
   emit(key, ("R", value))
}
```

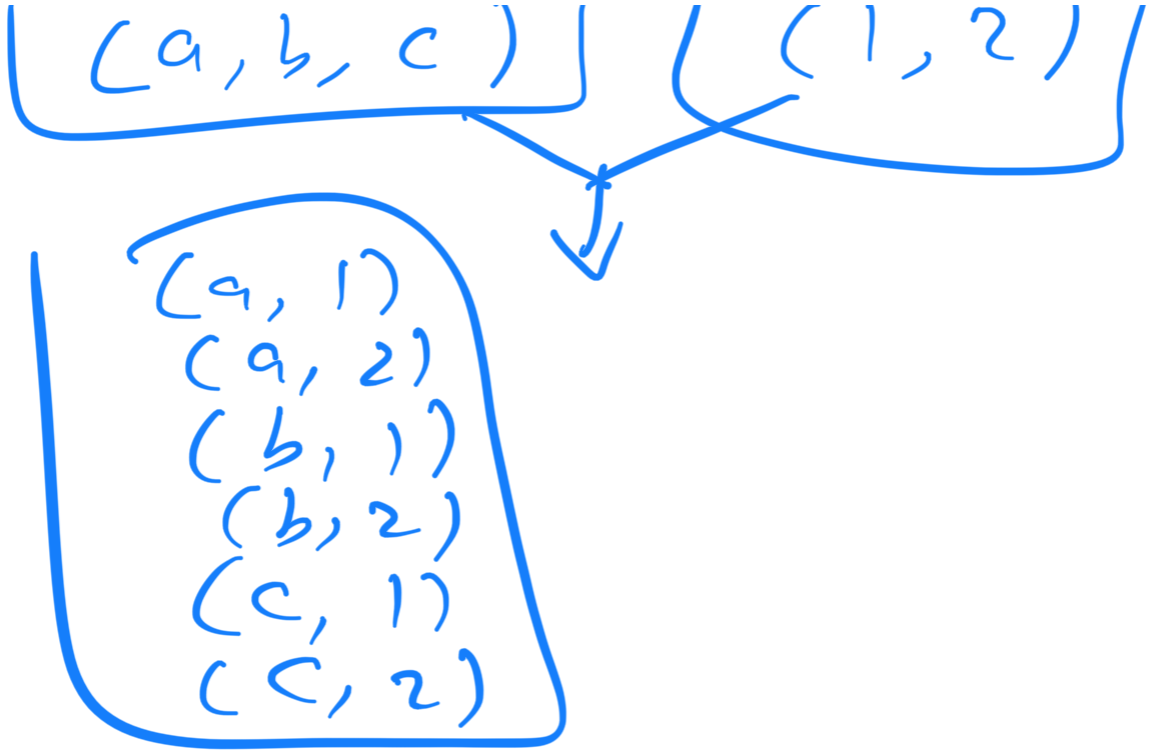
② map S

```
map(k, v) {
     tokens = v.split(",")
```

```
    key = tokens[0]
    value = tokens[1]
    emit(key, ("S", value))
}
```



Cartesian Product




```
reduce((K), values) {
  # K = actual key
  # values = [(R, v1)(R, v2), ..., (S, u1), (S, u2), ...
  RList = []
  SList = []
  for pair in values {
      relation = pair[0]
      value = pair[1]
      if (relation == 'R') {
          RList.append(value)
      } else {
          SList.append(value)
      }
  }
  # Join
```

```
# key=b1
# Rlist = (a,)
# Slist = (c1, c2)
if (len(R) == 0) {return}
if (len(S) == 0) {return}
# Len(R) > 0 & len(S) > 0
# find join by Cartesian Product
for (a in Rlist) {
    for (c in Slist) {
        emit (a, k, c)
    }
}
# } end of reduce()
```