

<Milestone talk>

Improving Hadoop Performance by Weakening Dependency

김동원@동원리더스아카데미
포항공대 프로그래밍언어 연구실

Major milestones

Studying
Hadoop

I'm here

Designing & Developing
New MapReduce
Runtime System



Modifying
Hadoop

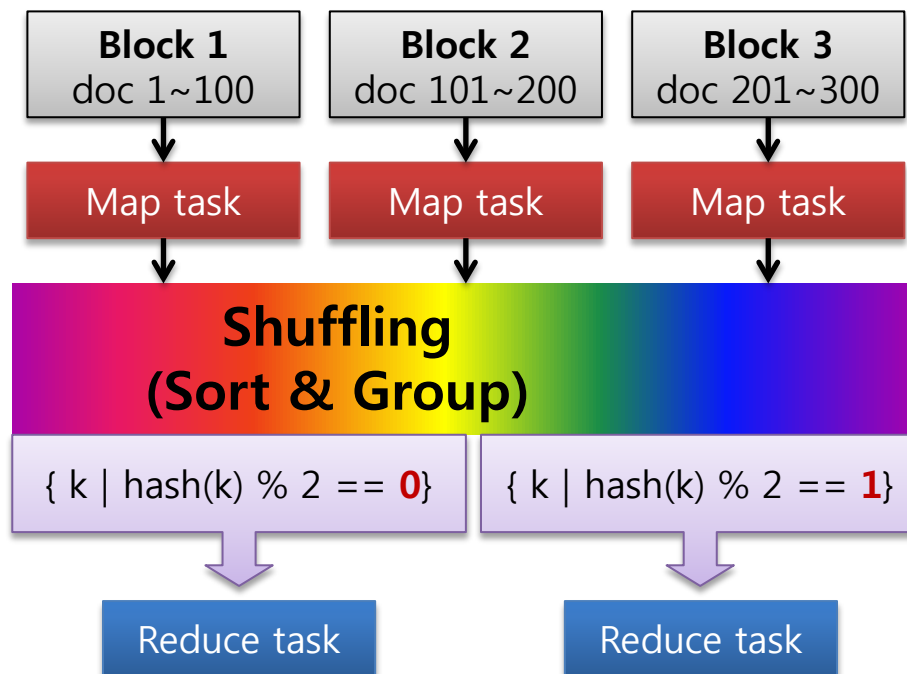


MapReduce review (1)

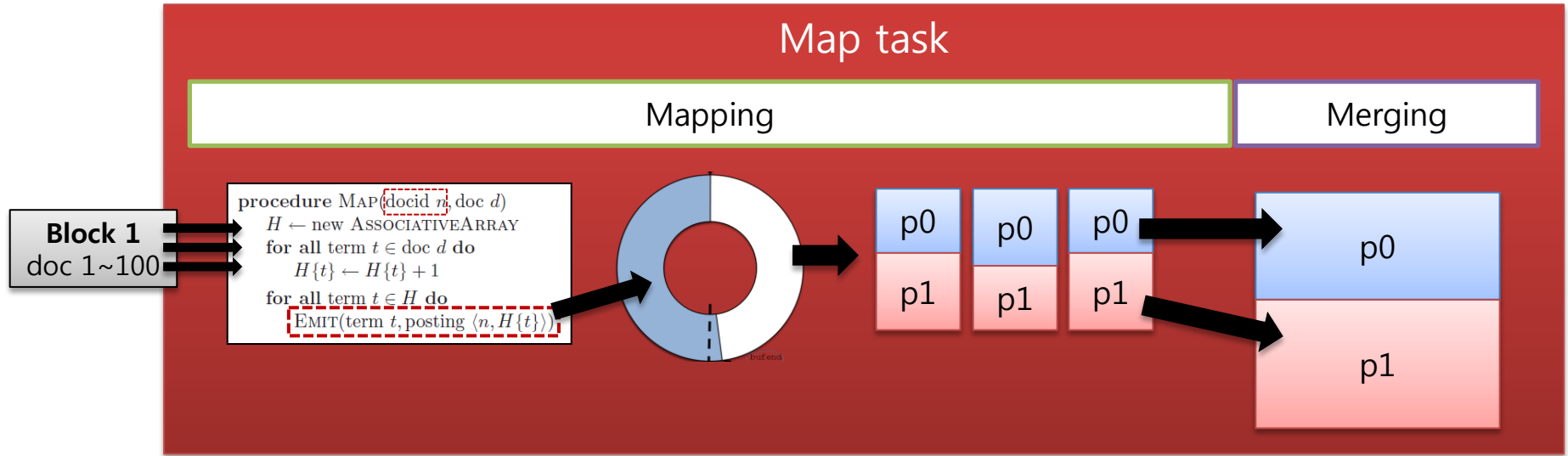
- Users specify the computation in terms of a **map** and a **reduce** function
- **MapReduce** runtime system automatically
 - + **parallelizes** the computation across large-scale clusters of machines
 - + **sort & group** intermediate pairs
 - + handles machine **failures**
- Hadoop is a representative MapReduce runtime system

```
procedure MAP(docid  $n$ , doc  $d$ )  
   $H \leftarrow$  new ASSOCIATIVEARRAY  
  for all term  $t \in$  doc  $d$  do  
     $H\{t\} \leftarrow H\{t\} + 1$   
  for all term  $t \in H$  do  
    EMIT(term  $t$ , posting  $\langle n, H\{t\} \rangle$ )
```

```
procedure REDUCE(term  $t$ , postings  $[\langle n_1, f_1 \rangle, \langle n_2, f_2 \rangle \dots]$ )  
   $P \leftarrow$  new LIST  
  for all posting  $\langle a, f \rangle \in$  postings  $[\langle n_1, f_1 \rangle, \langle n_2, f_2 \rangle \dots]$  do  
    APPEND( $P$ ,  $\langle a, f \rangle$ )  
  SORT( $P$ )  
  EMIT(term  $t$ , postings  $P$ )
```

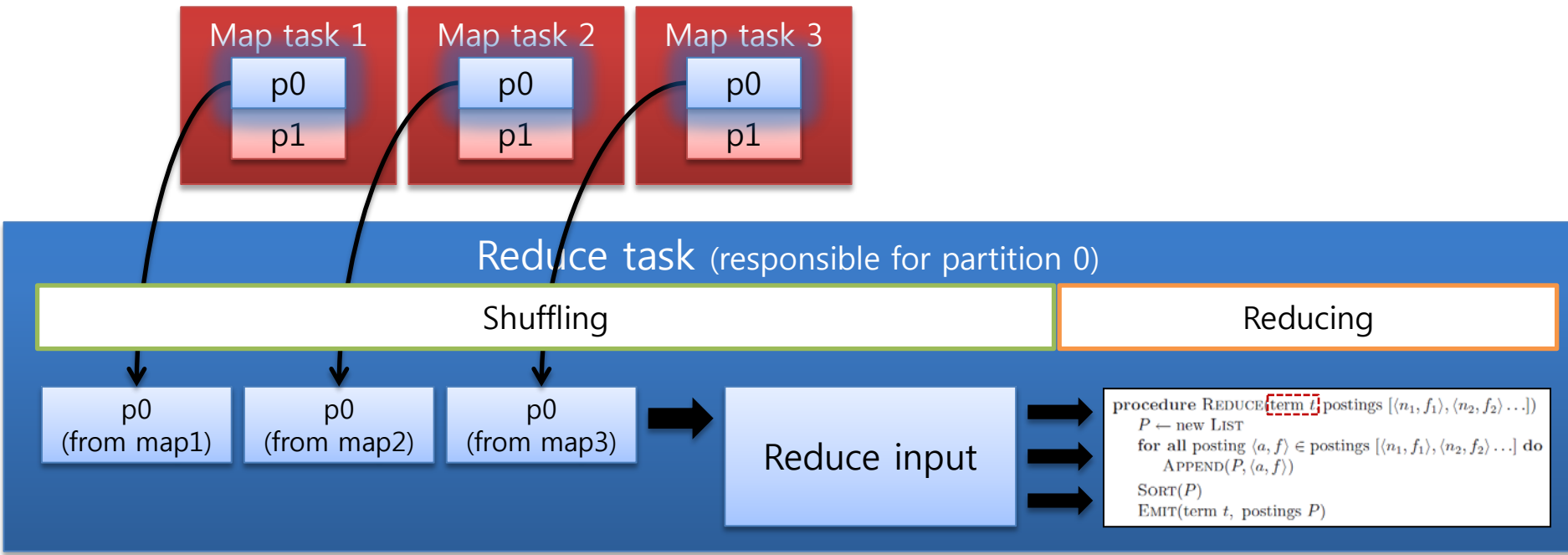


MapReduce review (2) – Map task



- **Map tasks** go through two phases
 - 1) Mapping
 - 2) Merging
- **Each map task** launches **its own merger**
 - To merge local spill files

MapReduce review (3) – Reduce task



- **Reduce tasks** go through two phases
 - 1) Shuffling
 - 2) Reducing
- **Reduce tasks** get their input from the output of **completed map tasks**
 - Reduce tasks are scheduled after a **pre-defined** number of map tasks are over

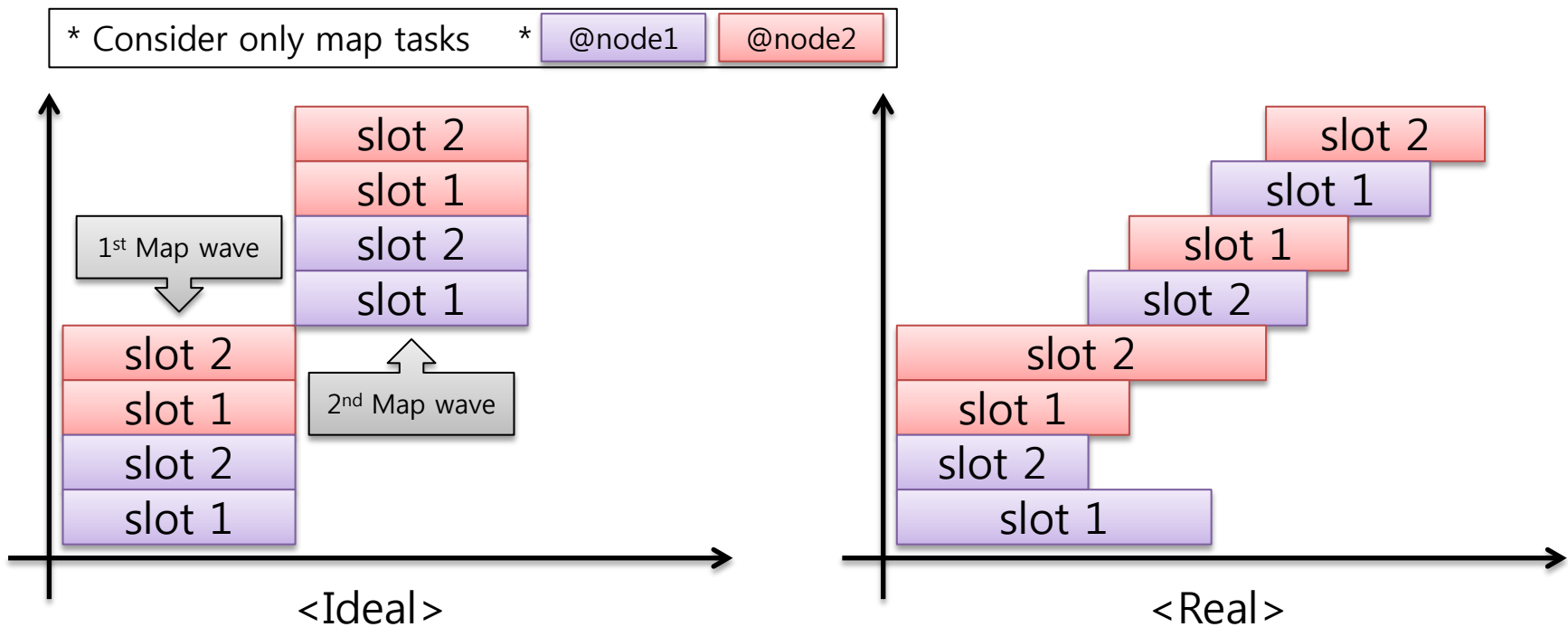
Improving Hadoop Performance by Weakening Dependency

- Goal
 - To make Hadoop run fast **regardless of # of map tasks**
 - **# of map tasks** has a significant impact on performance
- Challenge
 - To modify Hadoop design choices
 - **Each map task** launches **its own merger**
 - **Reduce tasks** get their input from the output of **completed map tasks**
- Solution
 - ??

More detail about Hadoop :

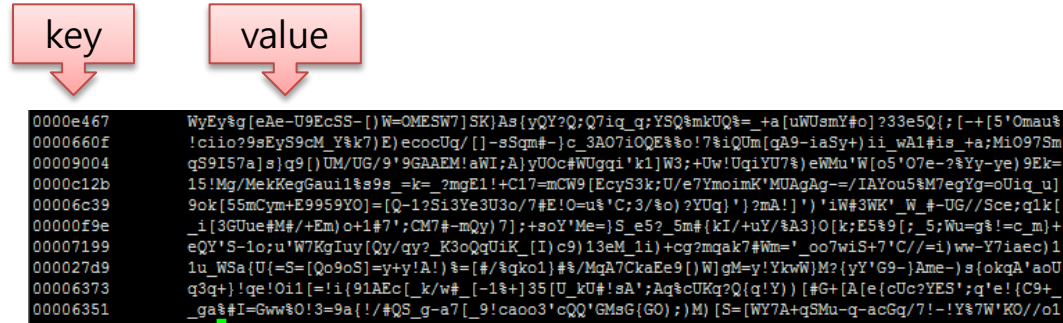
of simultaneous map tasks = # of **task slots**

- # of task slots = <user parameter>
 - If each of **2 nodes** has **2 map slots**, we can run **at most 4 map tasks** at an instant



of map tasks has a significant impact on performance

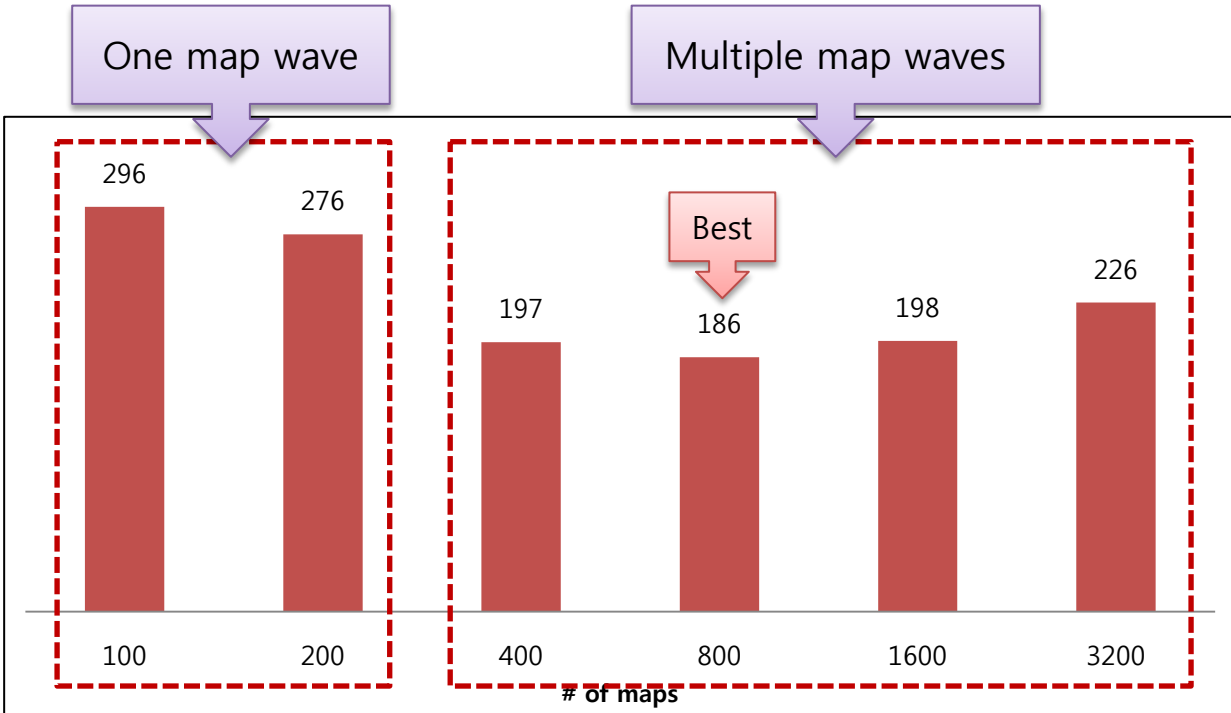
- 100G sorting
 - Identity map function
 - Identity reduce function
 - **20 nodes**
 - **12 map slots** per node
 - up to **240 map tasks** at an instant



<100G sorting input>

Block size	# blocks = # map tasks
1024 MB	100
512 MB	200
256 MB	400
128 MB	800
64 MB	1600
32 MB	3200

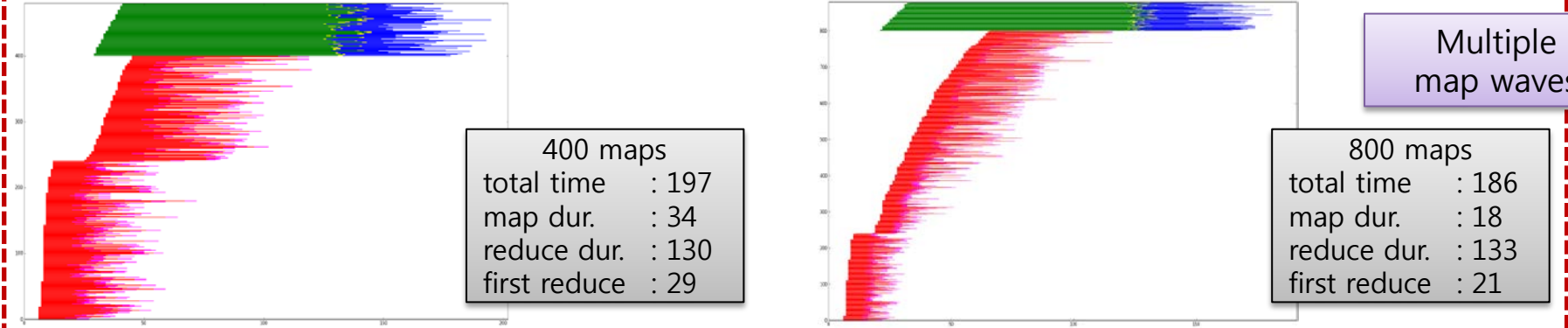
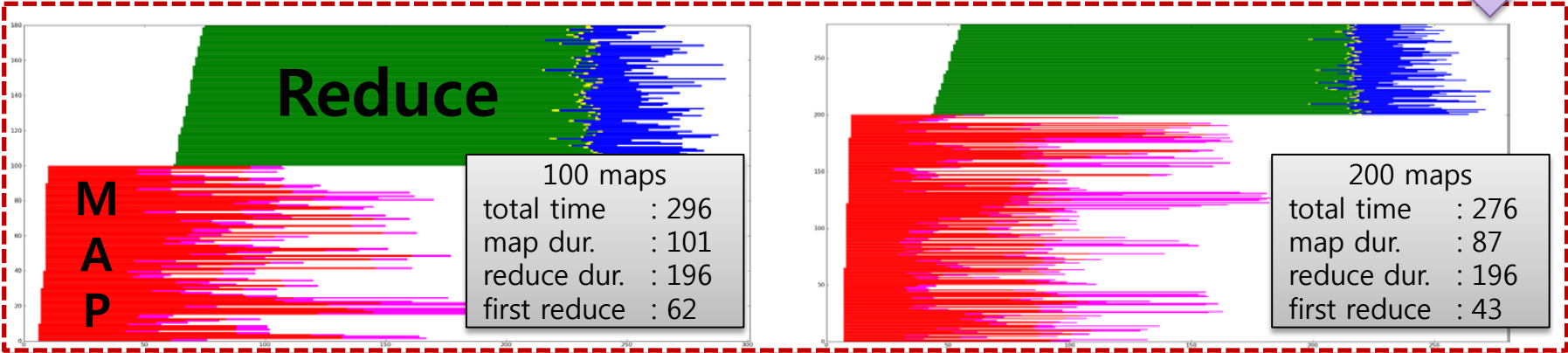
of reduce tasks : 80



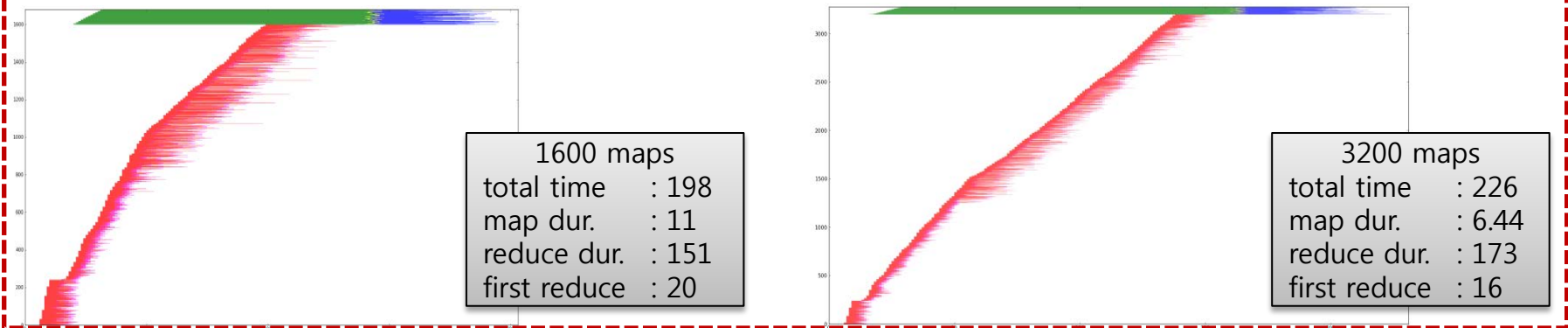
of map tasks has a significant impact on performance



One map wave

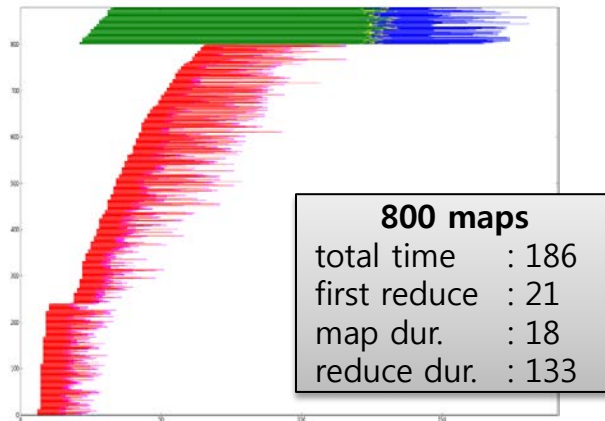
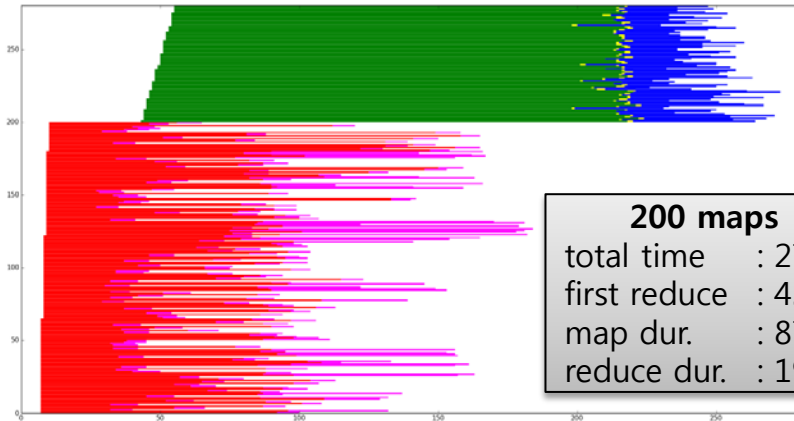


Multiple map waves




Why **one** map wave takes so long time?

200 map tasks vs. 800 map tasks



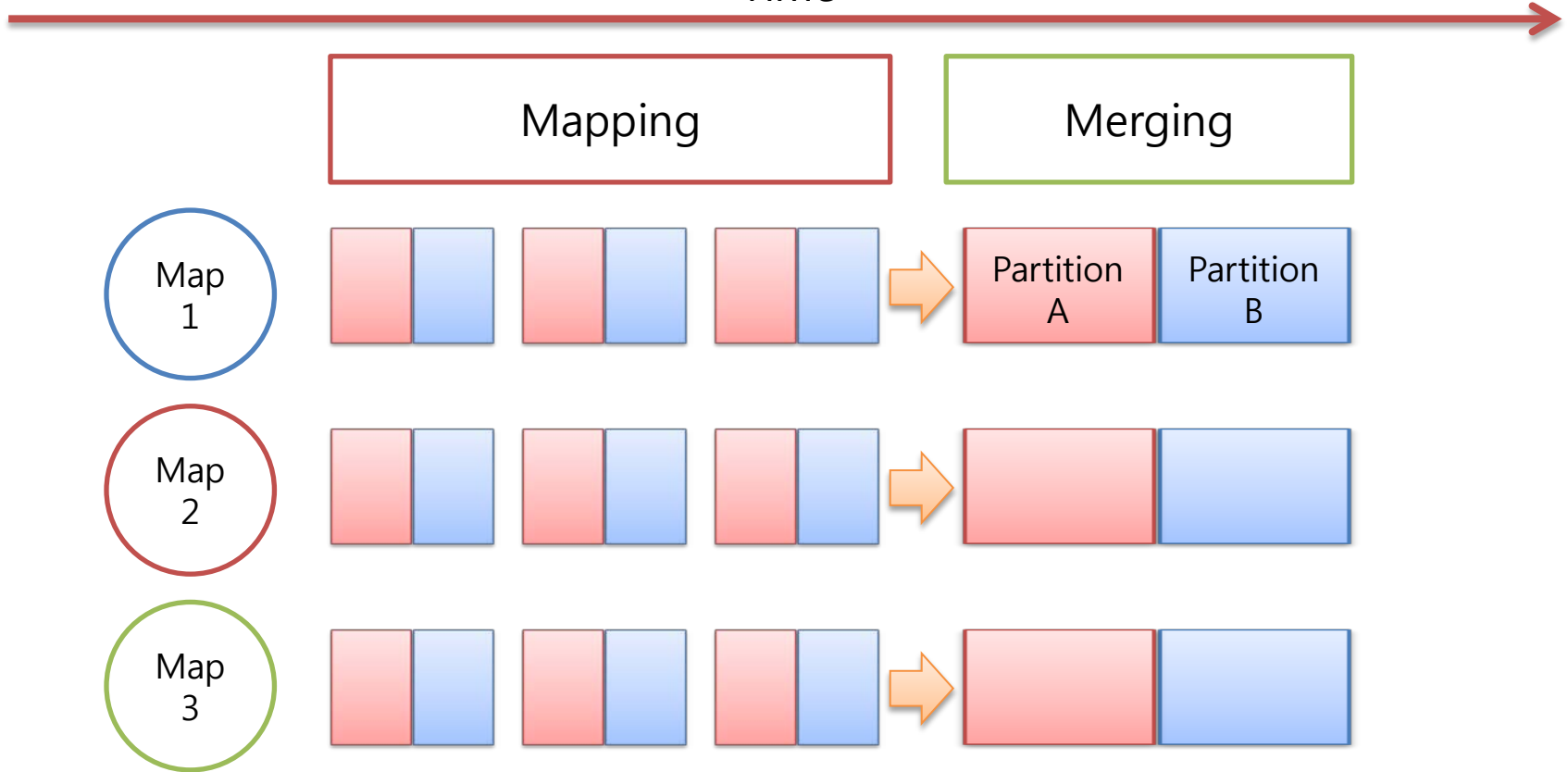
- What makes the difference (276 sec – 186 sec = 90 sec)?
 - 1) Beginning of reduce tasks (43 sec – 21 sec = 22 sec)
 - **Reduce tasks** get their input from the output of **completed map tasks**
 - **Each of the 200 maps** (left) takes longer than **each of the 800 maps** (right)
 - 2) Duration of reduce tasks (196 sec – 133 sec = 63 sec)
 - (with 800 maps) **smaller** map outputs are made continuously throughout the job
 - (with 200 maps) **mergers** are working around the same time
- 90 sec \doteq 22 sec + 63 sec

Improving Hadoop Performance by Weakening Dependency

- Goal
 - To make Hadoop run fast **regardless of # of map tasks**
 - **# of map tasks** has a significant impact on performance
- Challenge
 - To modify Hadoop design choices
 - **Each map task** launches **its own merger**
 - **Reduce tasks** get their input from the output of **completed map tasks**
- Solution 
 - Decouple **mergers** from map tasks

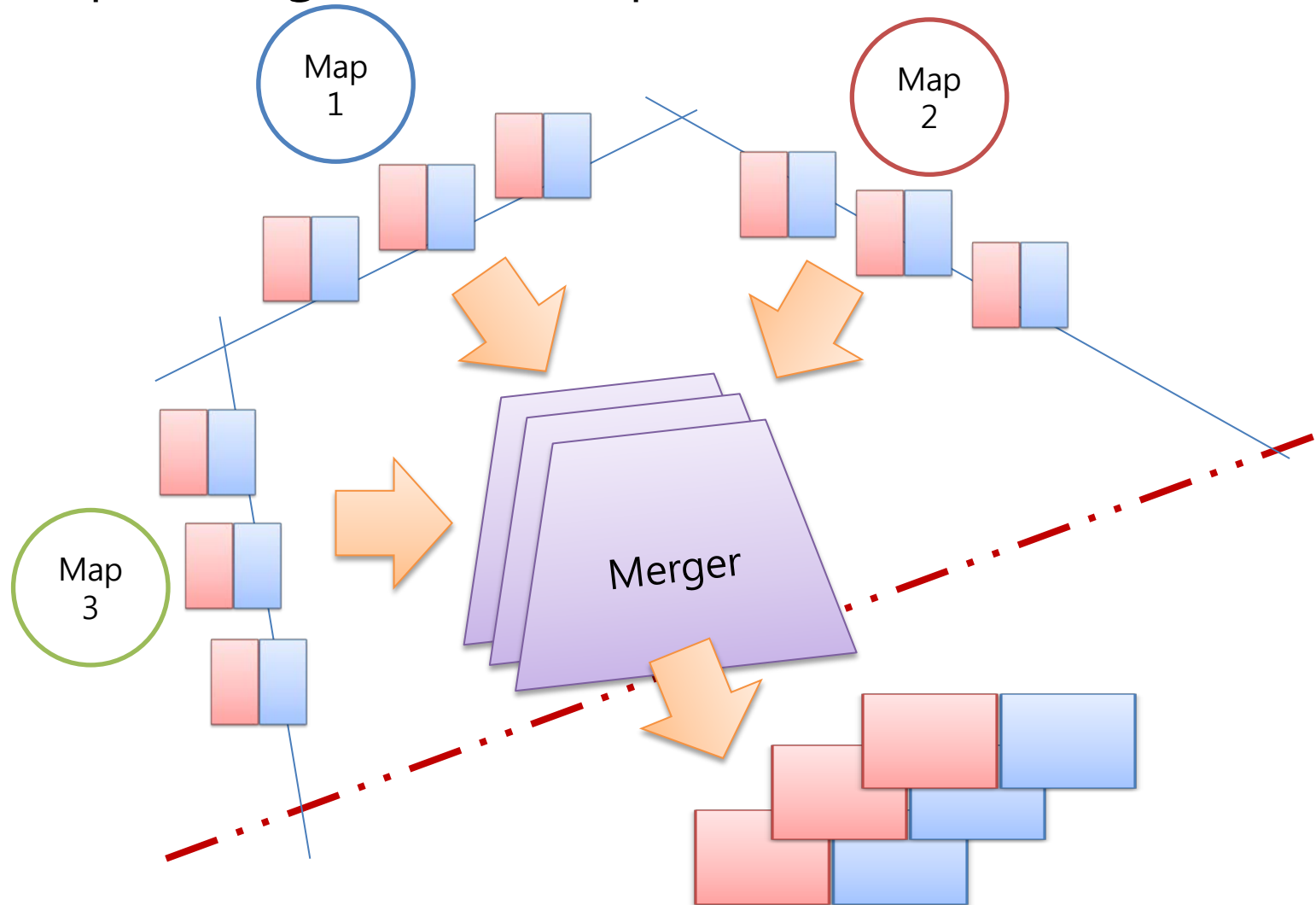
A Hadoop design choice – Each map task launches **its own** merger

Time



My idea –

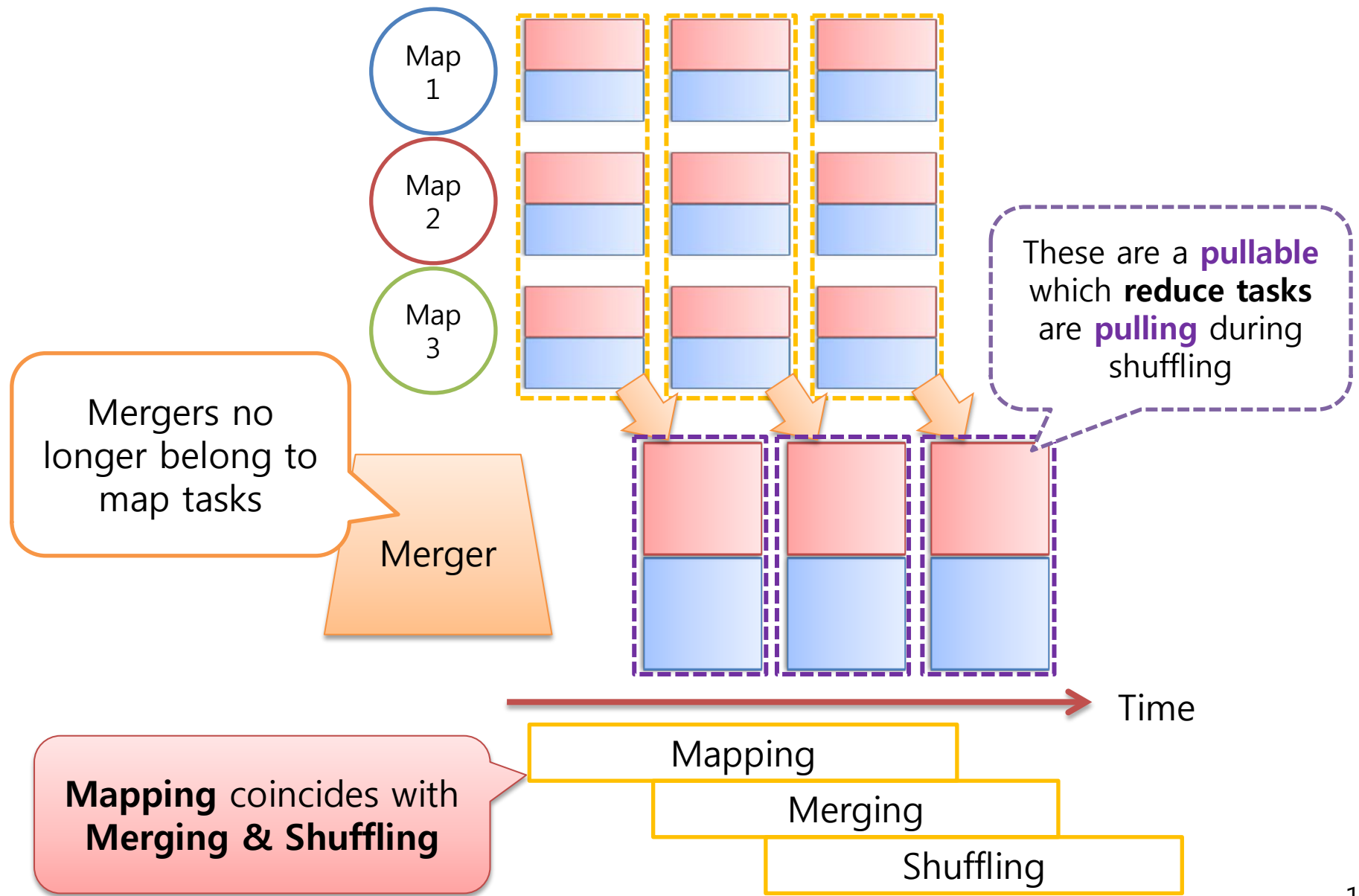
Decouple **mergers** from map tasks



- **Mergers** are working as soon as **some local spills** are available
- **Mergers** merge spill files regardless of their lineage

My idea –

Decouple **mergers** from map tasks



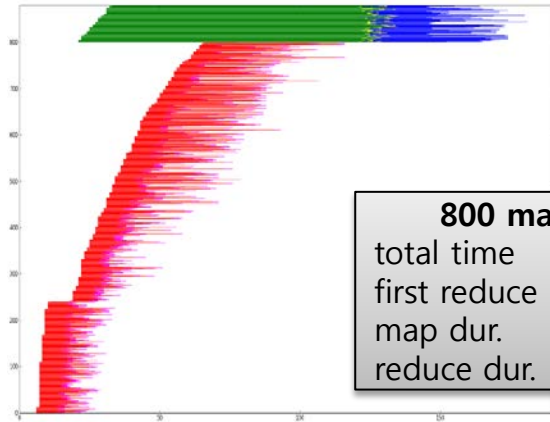
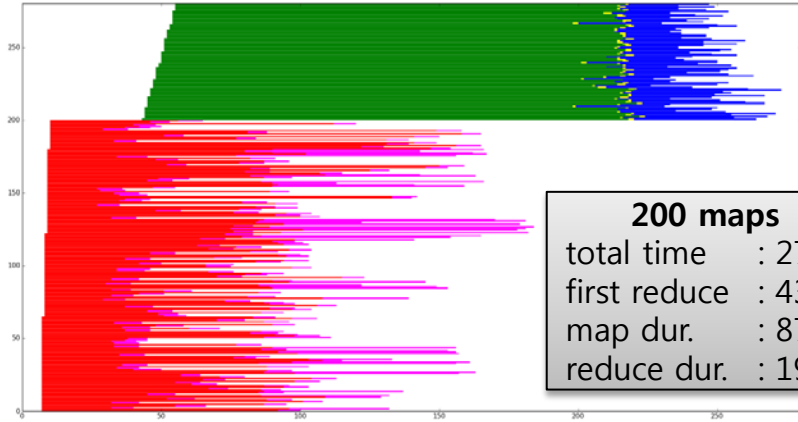
Expectations

- 1) Start shuffling **earlier** than before
 - as soon as some **pullables** are available
 - **not map outputs**

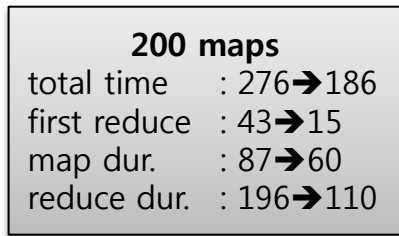
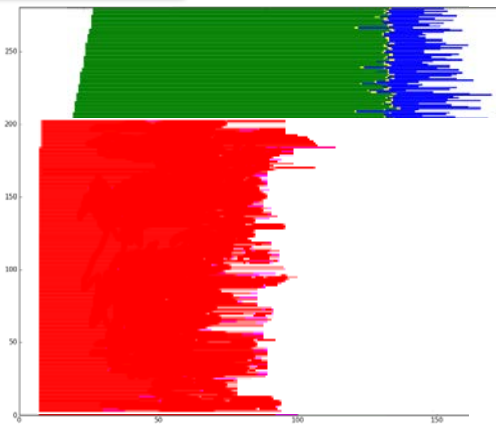
- 2) Have control over **when to merge** spills
 - to make **pullable copiers** work **constantly**

What if our expectations become reality?

Original



Goal



Advantages

- 1) Make Hadoop runs fast **regardless of # of map tasks**
 - Overlap different phases **regardless of # of map tasks**
 - ✓ Mapping
 - ✓ Merging
 - ✓ Shuffling

- 2) Ease the burden of choosing an appropriate value for # of maps

Conclusion

- Goal
 - To make Hadoop run fast **regardless of # of map tasks**
 - **# of map tasks** has a significant impact on performance
- Challenge
 - To modify Hadoop design choices
 - **Each map task** launches **its own merger**
 - **Reduce tasks** get their input from the output of **completed map tasks**
- Solution
 - Decouple **mergers** from map tasks