

map task 0 (p.txt)

"bucket" →
(just some file)

mr_0_0	mr_0_1
(k ₁ , v ₁)	(k ₃ , v ₃)
(k ₂ , v ₂)	(k ₅ , v ₅)
(k ₄ , v ₄)	

output of map function: n_{out} (k,v) pairs

Let $n_{out} = 5$.

map task 1 (q.txt)

mr_1_0	mr_1_1
(k ₁ , v ₁)	(k ₃ , v ₃)
(k ₆ , v ₆)	(k ₅ , v ₅)
(k ₇ , v ₇)	(k ₉ , v ₉)

Example: p.txt contains "apple" 5 times. We're running WC.

So $(k_1, v_1) = ("apple", 5)$. q.txt also contains apple, but 3 times.

So $(k_1, v_1) = ("apple", 3)$. Note that the k_1 from map task 0 is the same as the one from map task 0.

Worker 1 now works on reduce task 0.

mr-0-0

(k₁, v₁)
(k₂, v₂)
(k₄, v₄)

mr-1-0

(k₁, v₁)
(k₆, v₆)
(k₇, v₇)

Concat all 6 (k,v) pairs and sort.

(k₁, v₁)
(k₁, v₁)
(k₇, v₇)
(k₂, v₂)
(k₆, v₆)
(k₄, v₄)

Reduce k₁: Take in (k₁, v₁) and (k₁, v₁) and produce (k₁, v₁).

Reduce k₇: Take in (k₇, v₇) and produce (k₇, v₇).
etc.

Note that the following scenario where a reduce task doesn't "fully aggregate" all the vals for a key is impossible.

Ex:

mr_0_0
(k ₁ , v ₁)
(k ₂ , v ₈)
(k ₄ , v ₄)

mr_0_1
(k ₃ , v ₂)
(k ₅ , v ₅)

mr_1_0
(k ₅ , v ₅)
(k ₆ , v ₆)
(k ₇ , v ₇)

mr_1_1
(k ₃ , v ₃)
(k ₁ , v ₁)
(k ₂ , v ₂)

Here, reduce task 0 won't be aware of (k₁, v₁).

This is impossible b/c the hash function and modulo will put (k₁, v₁) and (k₁, v₁) into the "same bucket" (i.e. if (k₁, v₁) goes into mr_0_i, (k₁, v₁) goes into mr_1_i).