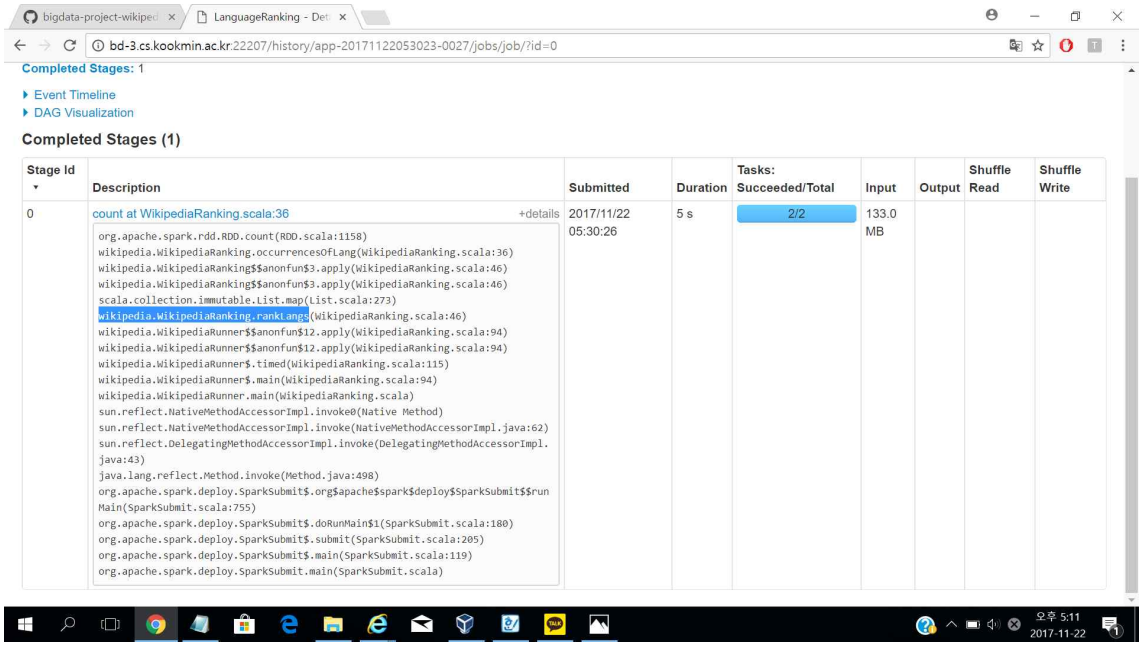
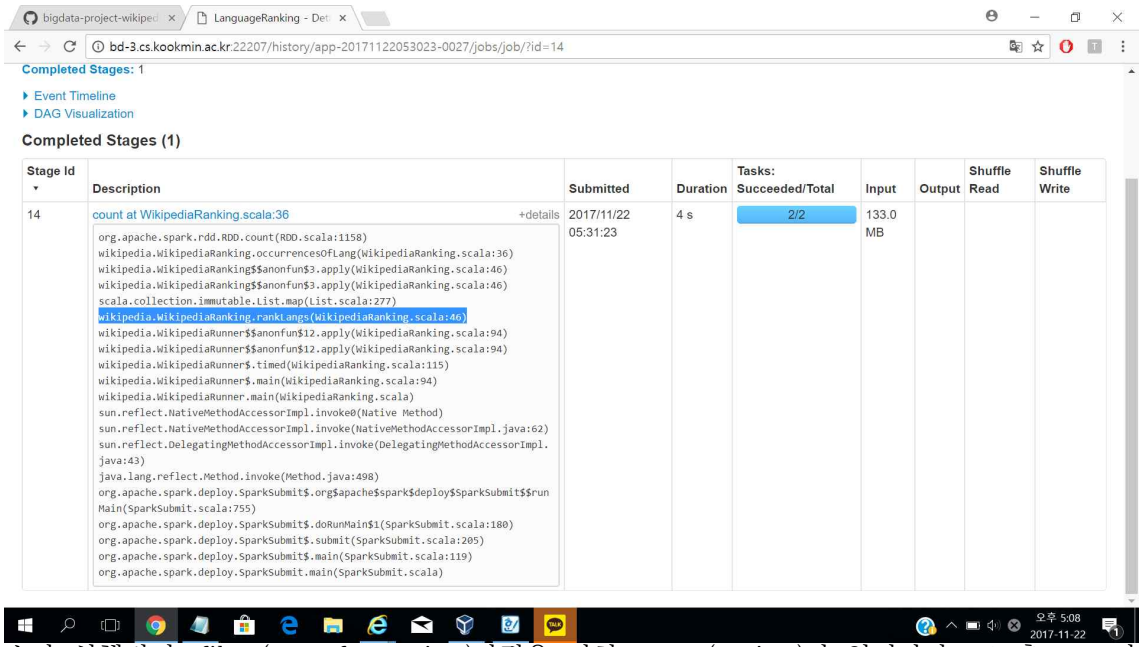



위의 스크린샷은 첫 번째 함수인 rankLangs를 보여주는 과정이다. ooccurrencesOfLang 함

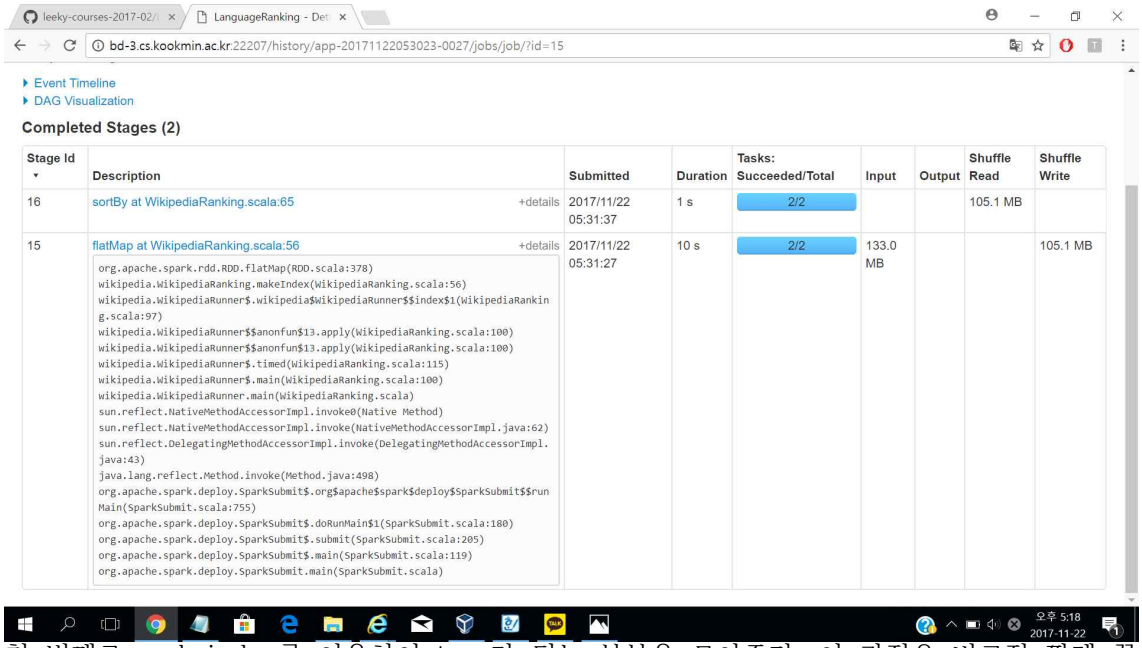



수가 실행되어, filter(transformation)과정을 거쳐 count(action)이 일어난다. 그 후 map과정과 sort과정이 이루어진다. 이때 작업이 모아서 이루어지는 것이 아니라 각각 이루어지기 때문에 여러 stage의 흐름을 보인다.

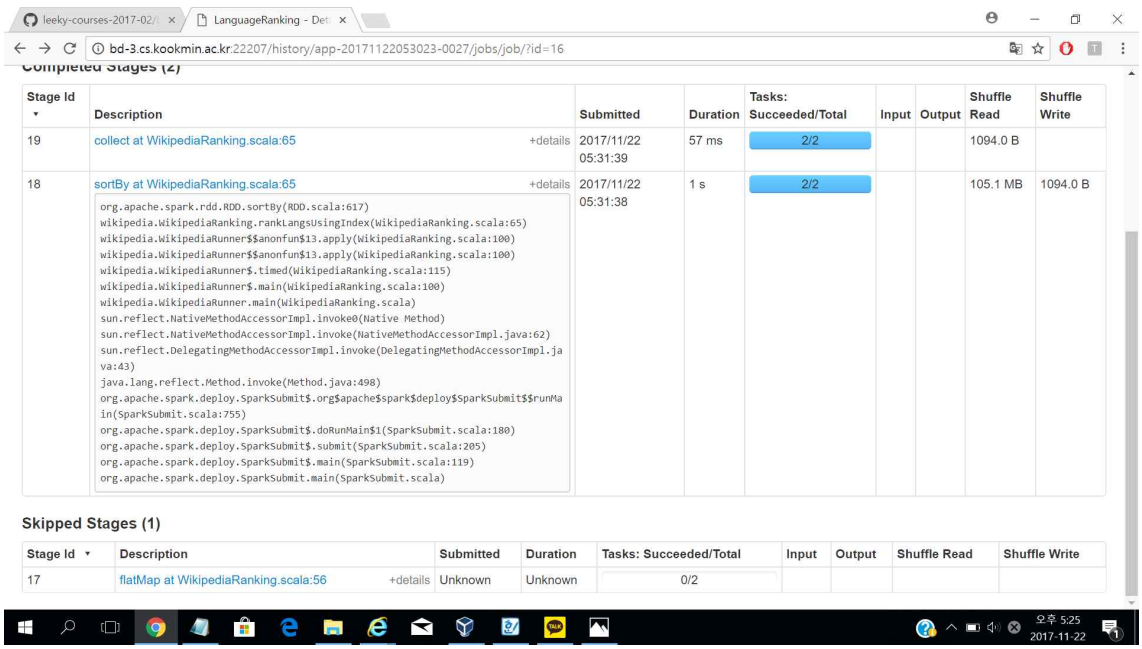
이를 자세히 설명하자면 다음과 같다.

스크린 샷에서 보이다시피, 0번 job부터 14번 job까지 rankLangs 함수가 이용됨을 볼 수 있다. Task3에서 보다시피, rankLangs의 시간이 가장 오래 걸리는데, 이는 데이터를 처리하는 과정이 더 많기 때문이다. job의 개수가 많으니 오래 걸리는 것은 당연한 부분이다. 데이터를 한번에 처리하는 것이 아닌 RDD를 계속해서 호출해주고(Transformation), action이 이루어지기 때문에, 이 부분에 대한 오버헤드가 크다.

두 번째 함수인 rankLangsUsingIndex는 첫 번째 함수보다는 시간이 적게 걸린다.
그 과정은 다음과 같이 나타난다.

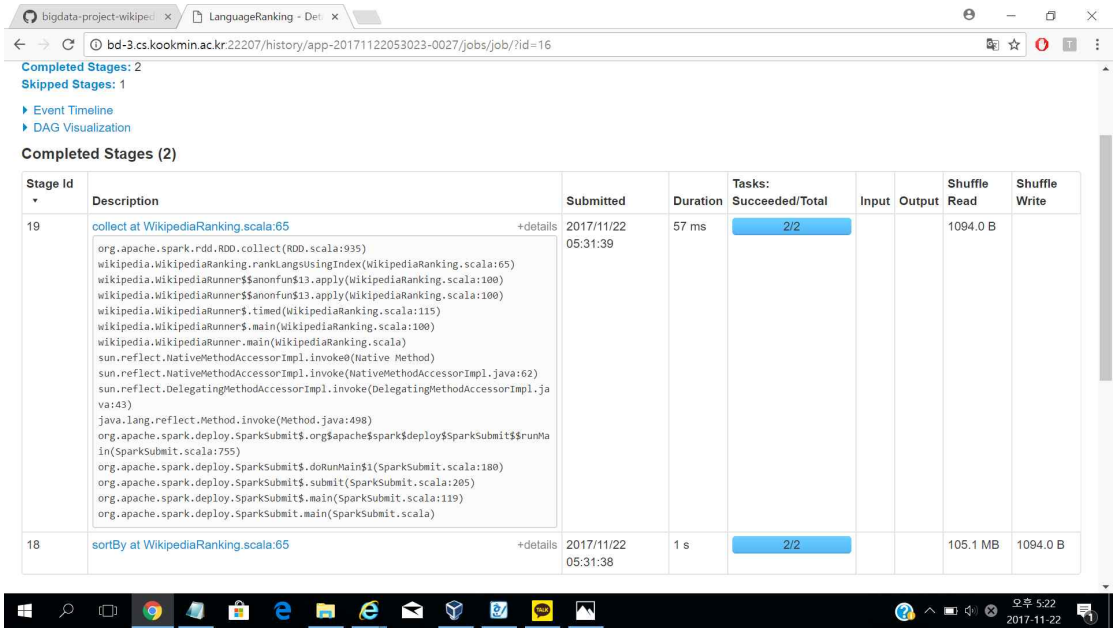



첫 번째로 makeindex를 이용하여 true가 되는 부분을 모아준다. 이 과정은 비교적 짧게 끝이 난다. 이 부분은 15번 job이다

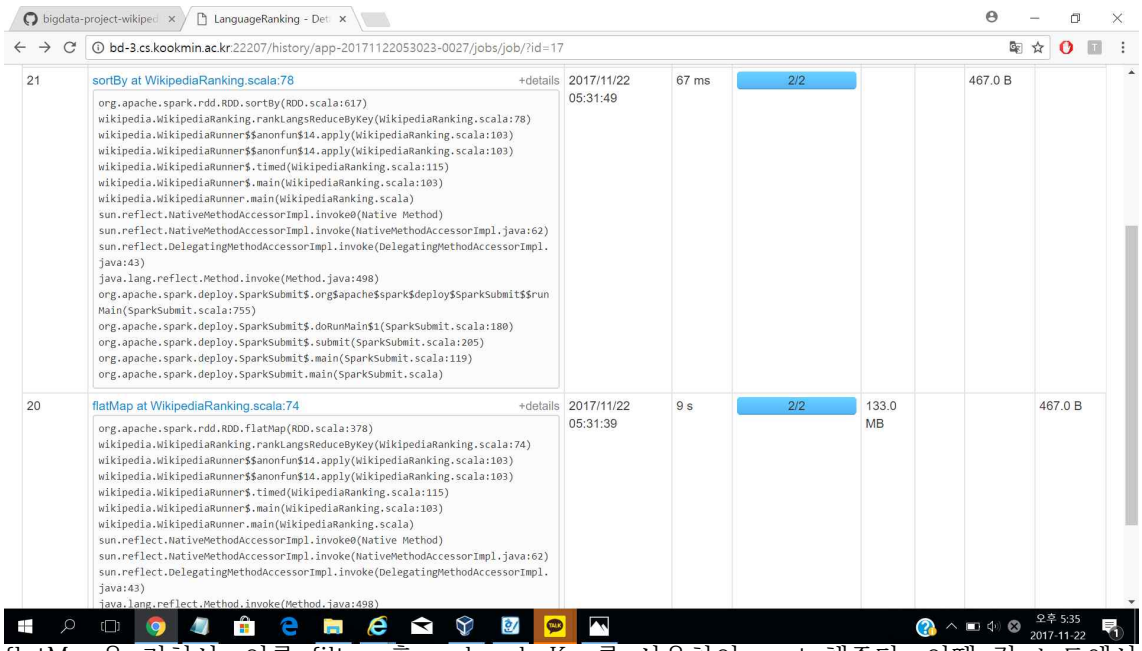



그 후 위의 사진과 같이 16~18 과정을 거쳐서(이 때 17번 flatMap은 스킵 된다. 따라서 성능이 더 우수하다,) sort까지 이루어진다. 마지막으로 collect(이때 시간이 가장 오래 걸린다) 해준 뒤 List로 만들어준다. 이는 아래의 19번 stage이다. 첫 번째 함수에 비하여 job이 월등히 작음을 볼 수 있다. 다른 점이 있다면, rankLangs의 경우에는 한번의 job동안 한번의 stage가 실행되지만, rankLangsUsingIndex는 한번의 job에 여러 가지 stage가 있어, job의

개수가 rankLangs에 비해 작다. 이 부분 때문에 실행시간의 차이가 나타난다. 마지막으로 collect(action)의 시간은 아래의 사진에 잘 나타나있다.

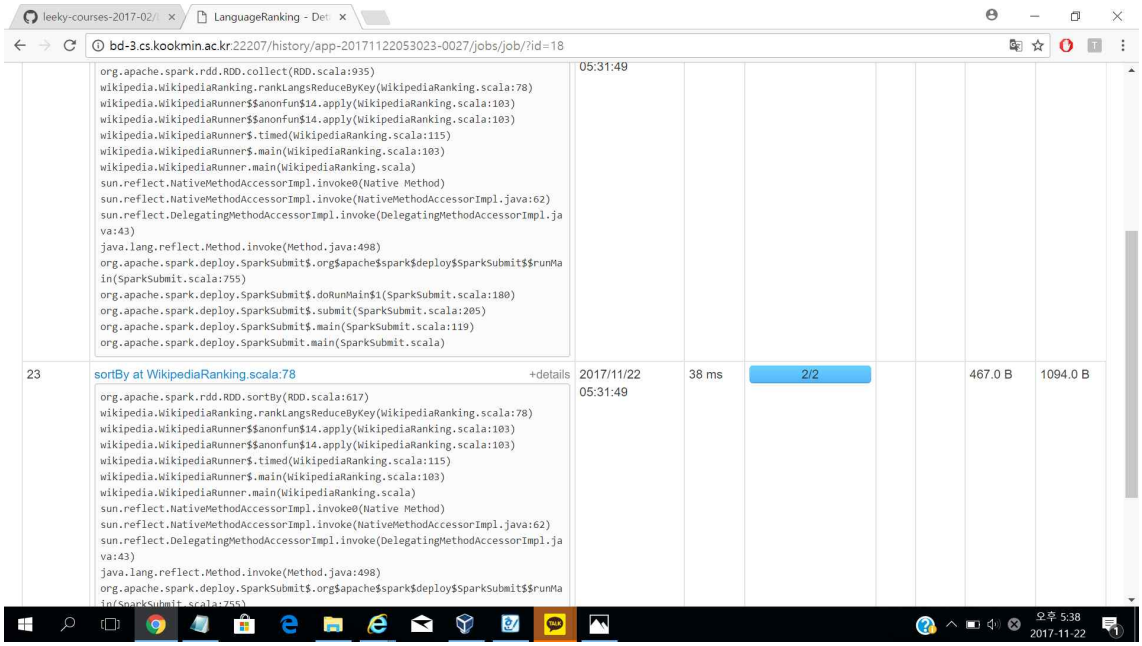



마지막 함수인 rankLangsReduceByKey이다. 진행과정은 다음과 같다.




flatMap을 거쳐서, 이를 filter 후 reducebyKey를 사용하여 sort 해준다. 이때 각 노드에서 중간 집계를 진행하기 때문에 stage수가 적다.

그 후 동일 한 과정이 반복 된다. 다시 말해 4 stage만에 모든 과정이 끝난 것이다. 각각에서 처리 하는 것이 아닌, 중간에 집계 후 그 집계 한 부분을 합친 것이기에 속도가 가장 빠르게 된다. 아래의 스크린샷에 최종적으로 마무리되는 과정과 시간이 나타나있다.

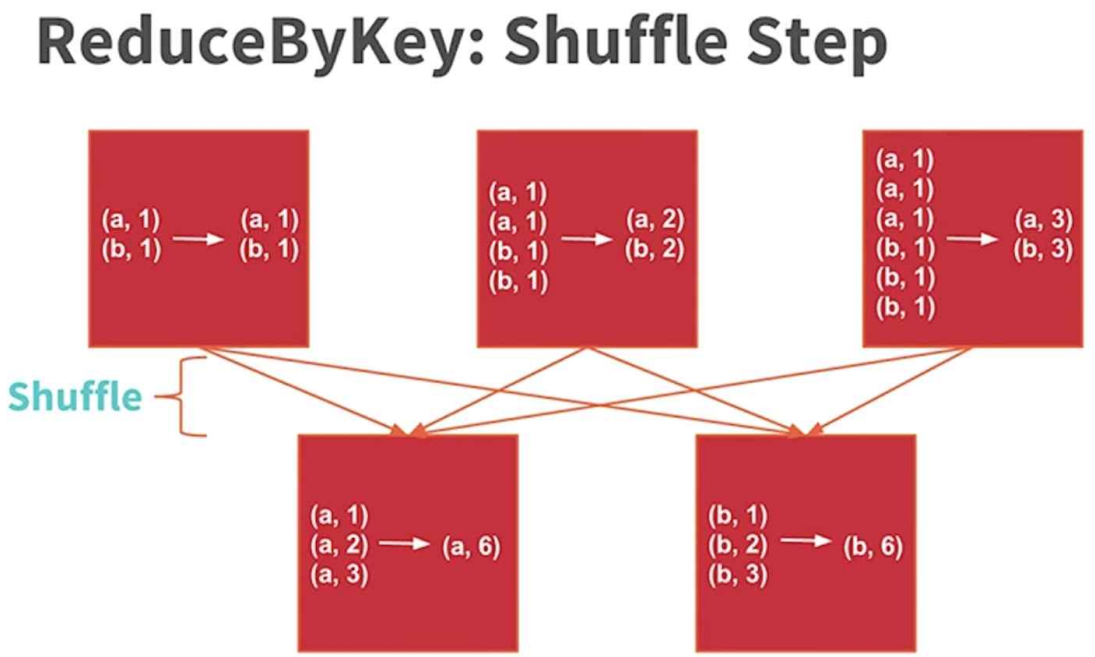


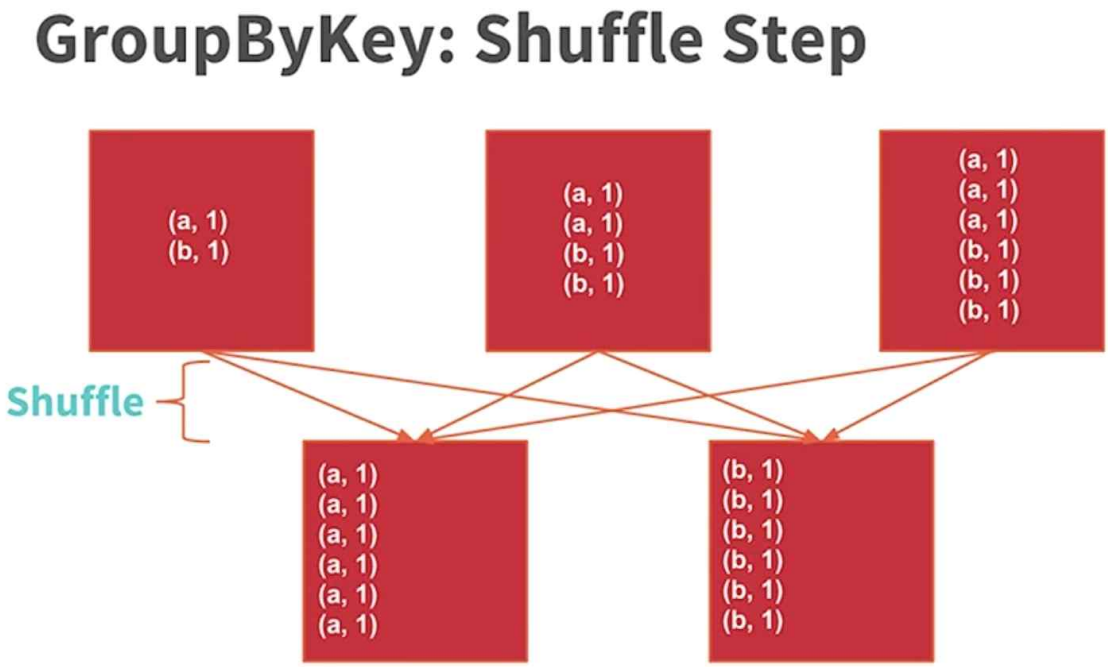

각 함수에 대한 자세한 설명은 다음과 같다.
rankLangs의 경우에는 langs을 map(transformation)하여, 이때 각각의 element(lang)과 이 element(lang)을 occurrencesOfLang에 사용하여 (k,v)RDD 타입으로 map해준다. 이때 occurrencesOfLang에서 langs의 element(lang)들은 filter(transformation)된 뒤 count되어 값으로 반환된다. 따라서 [element(lang), 와 occurrencesOfLang(lang)의 리턴값이](key,value) 형식으로 map되고 이를 sortBy로 descending order 해준다. 이는 List([String,Int]) 형태 이다.

rankLangsUsingIndex의 경우에는 우선 makeIndex를 이용하여 index값을 구해주게 된다. makeIndex에서는 WikipeaArticle 의 element(news)와 langs의 element(lang)을 이용하여 <key,value>형식(lang 과 그 lang이 나타나는 횟수) 꼴과 매우 유사하지만 다르다. 우선 key는 동일하지만 이때 value는 lang이 나타는 횟수가 아니라 true 값의 집합이다. 이것을 Iterable 하게 만들기 위해서는 groupbyKey가 실행되어야 한다.(이는 CompactBuffer(WikipediaArticle, WikipediaArticle, WikipediaArticle)꼴이다. 따라서 최종적으로 (lang, (article,article))형식이 반환된다. 이것이 index이다. 이것을 rankLangsUsingIndex에서 mapValues을 사용하여 <K,V>RDD형식으로 변환해준다. 그 후 sorting과 collect(action)그리고 toList 하여 최종적으로 우리가 필요로 하는 최종적인 List([String,Int])이 된다.

rankLangsReduceByKey는 우선 flatMap을 이용하여 element(article)들을 처리해준다. 이때 flatMap 안에서 element(article)과 langs의 element(lang)을 map 하여 (java,1),(python,1)...과 같은 flatMap을 만들어 준다. 그 후 rudeceByKey(transformation)를 사용하여 집계를 해주고(key 가 같다면 value를 합쳐준다.) 그 후 sortBy과 collect(action)그리고 toList 하여 최종적으로 우리가 필요로 하는 최종적인 List([String,Int])이 된다.

<보충> 우선 reduceByKey의 경우, 먼저 각 노드에서 중간 집계를 진행하고 이에 대한 결과를 동일한 키 값으로 전송된다.




반면, groupByKey는 각 노드에 있는 데이터에 대해 바로 Shuffle 과정을 거치게 되고 결과를 내보낸다. 따라서 groupByKey는 네트워크를 통해 전송되는 데이터의 양이 많아질 뿐만 아니라, Out of disk 문제가 발생할 수도 있다.
따라서 이런 오버헤드의 차이 때문에 reduceByKey가 더 우수한 성능을 보인다.