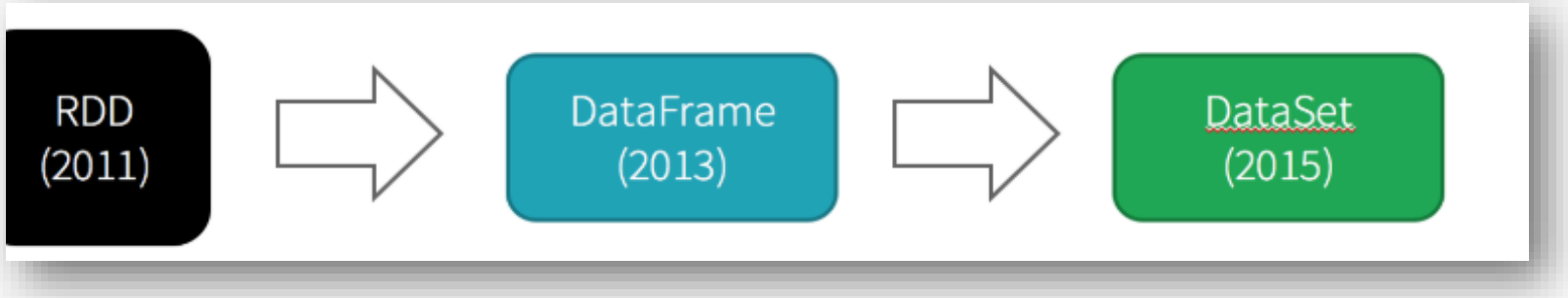
빅데이터 플랫폼 머신러닝 개발을 위한

Spark RDD APIs & example

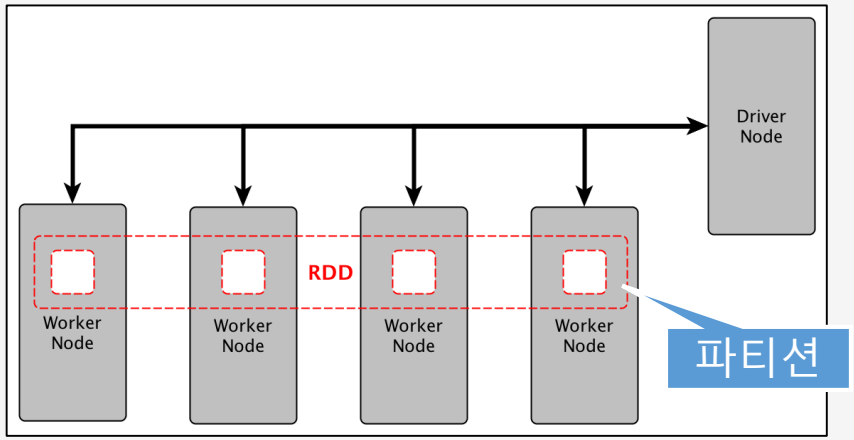
작성자 : 김진성

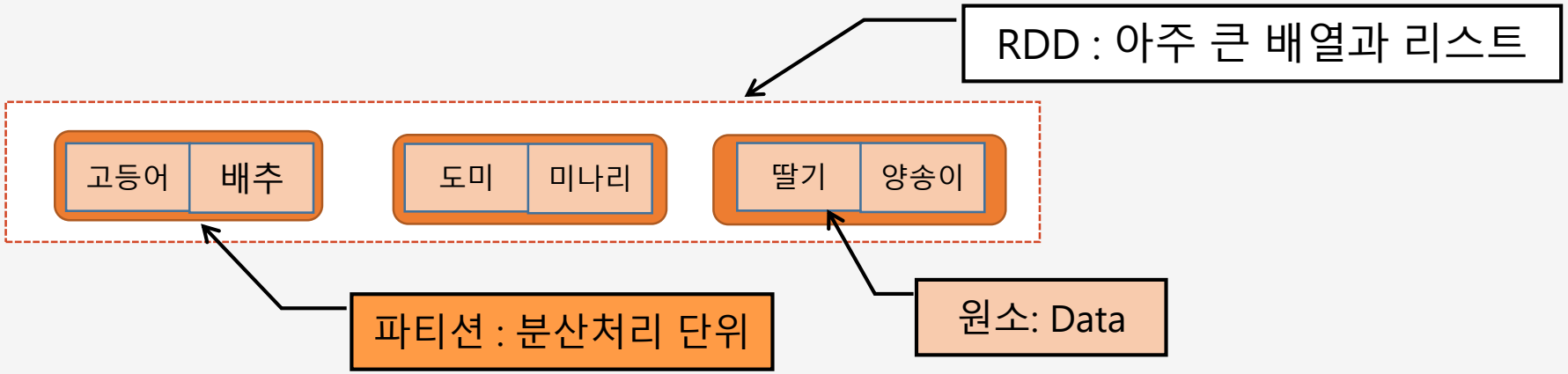
History of Spark APIs



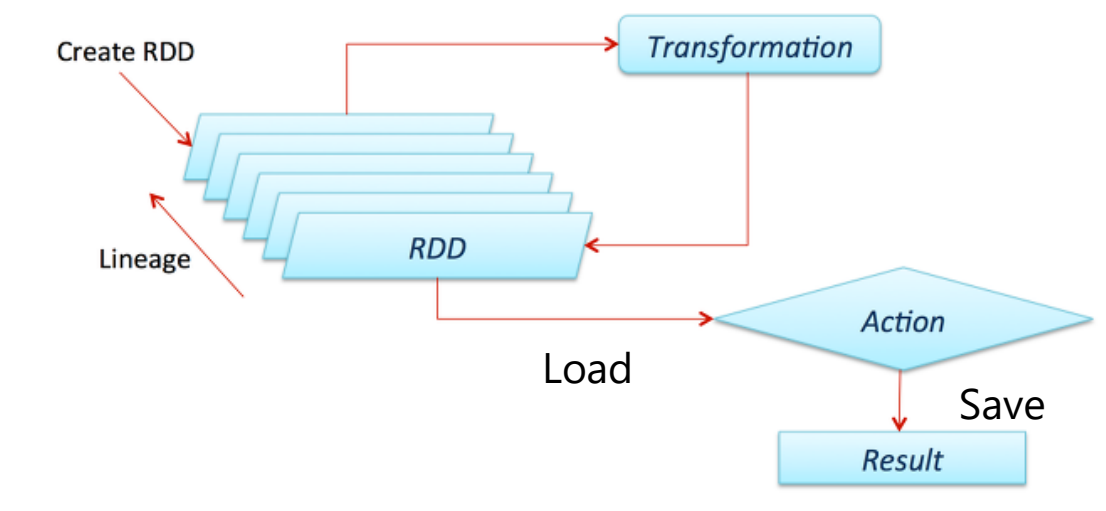
1. Spark RDD 개요

- RDD(Resilient Distributed Datasets)
  - ✓ 유연한 분산 데이터 셋 - 아주 큰 배열과 리스트 자료구조
  - ✓ Spark 빅데이터 처리를 위한 자료구조
  - ✓ 여러 대의 기계(분산 노드)에 저장된 변경 불가능한 Data 모음
  - ✓ 내부는 파티션이라는 단위로 분류(파티션 : 분산처리 단위)
  - ✓ 구성요소 : 파티션과 원소(element)





2. RDD 연산(RDD Operation)



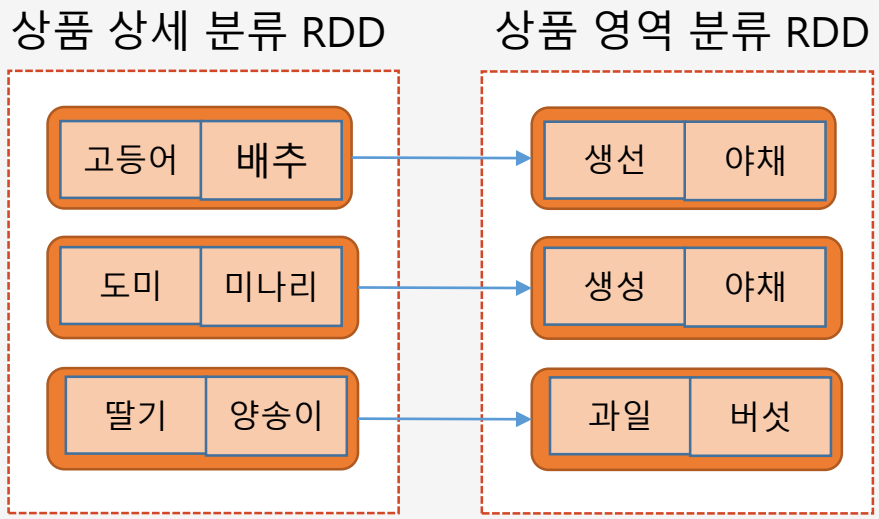
- Transformation : 기존 RDD를 가공 및 필터링하여 새로운 RDD 생성하는 연산
- Action : RDD를 가져오거나 외부 저장소에 저장하는 연산(RDD load/save)

3. Transformation 메서드

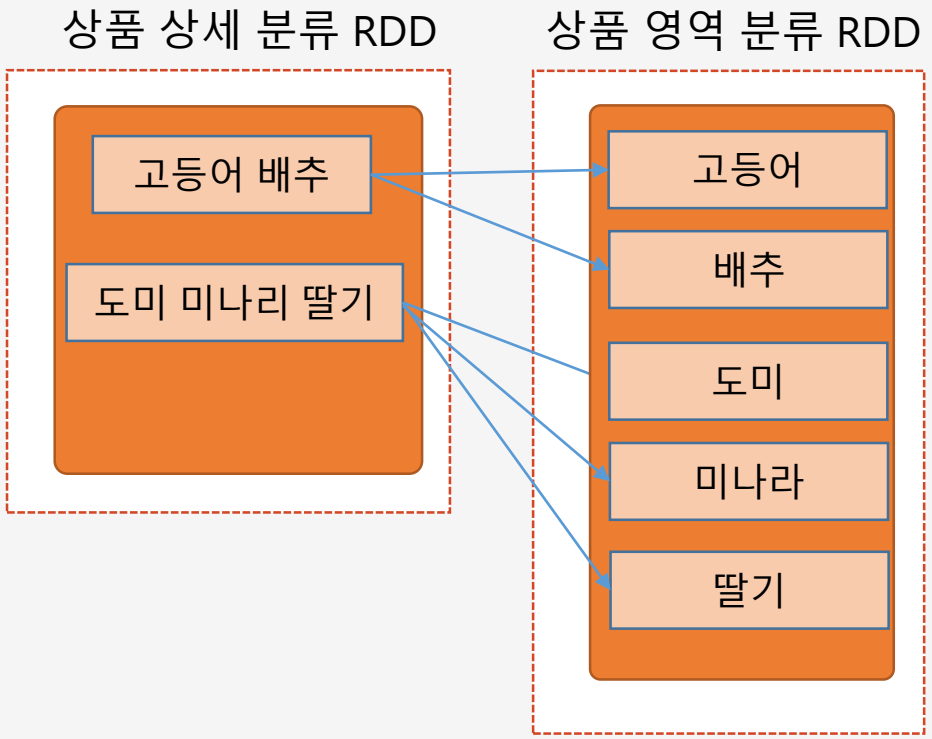
1. **map(func) :** RDD의 각 요소를 func 함수를 통해 전달하여 새 RDD 리턴
2. **flatMap(func) :** map과 비슷하지만 각 입력 항목을 여러 출력 항목으로 매핑
3. **filter(func) :** func이 true를 리턴하는 요소를 선택하여 새 데이터셋 반환
4. **reduceByKey(func) :** (k, v) 쌍의 데이터 집합에서 호출 될 때 주어진 감소 함수 func를 사용하여 각 키의 값이 집계되는 (k, v) 쌍의 데이터 집합을 반환
5. **union(other dataset) :** 합집합 반환(u3 = u1.union (u2) >>> u3.collect())
6. **intersection(other dataset) :** 교집합 반환
7. **distinct() :** 소스 데이터 세트의 유일한 요소만으로 반환(d2 = d1.distinct ())
8. **zip()** : 동일한 길이를 갖는 RDD 조합
9. **join(other dataset) :** 키-값 쌍 (k, v) 및 (k, w)의 데이터 집합에서 호출되면 모든 요소 쌍과 함께 (k, (v, w)) 쌍의 데이터 집합을 반환 : 동일한 키를 갖는 RDD 조합

3. Transformation 메서드

- map() 예 : 각 요소별 1:1 가공 연산

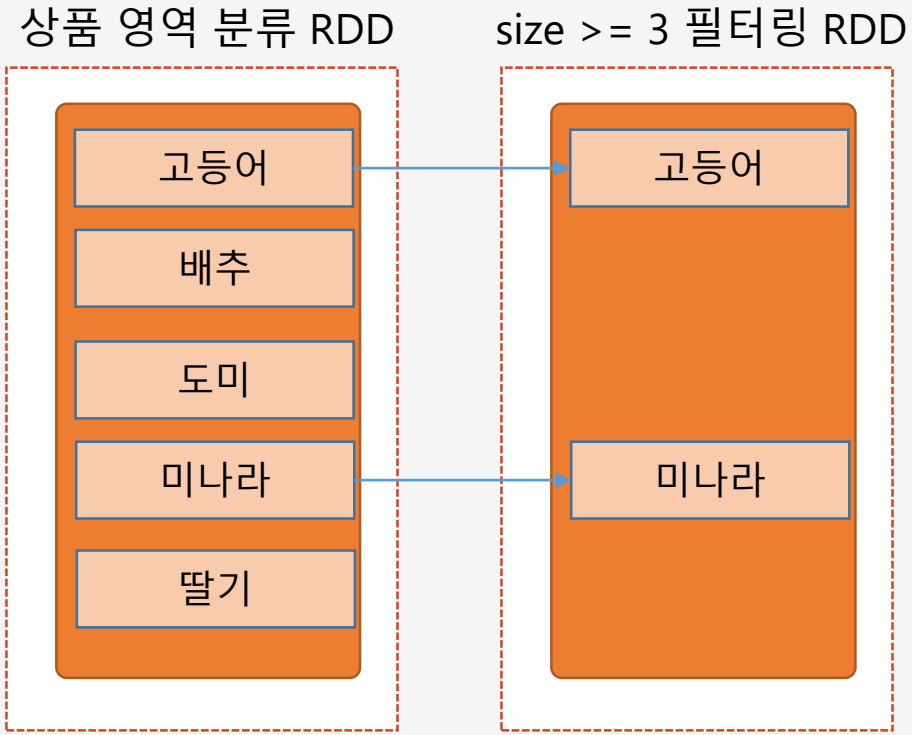


- flatmap() 예 : 각 요소별 1:N 가공 연산

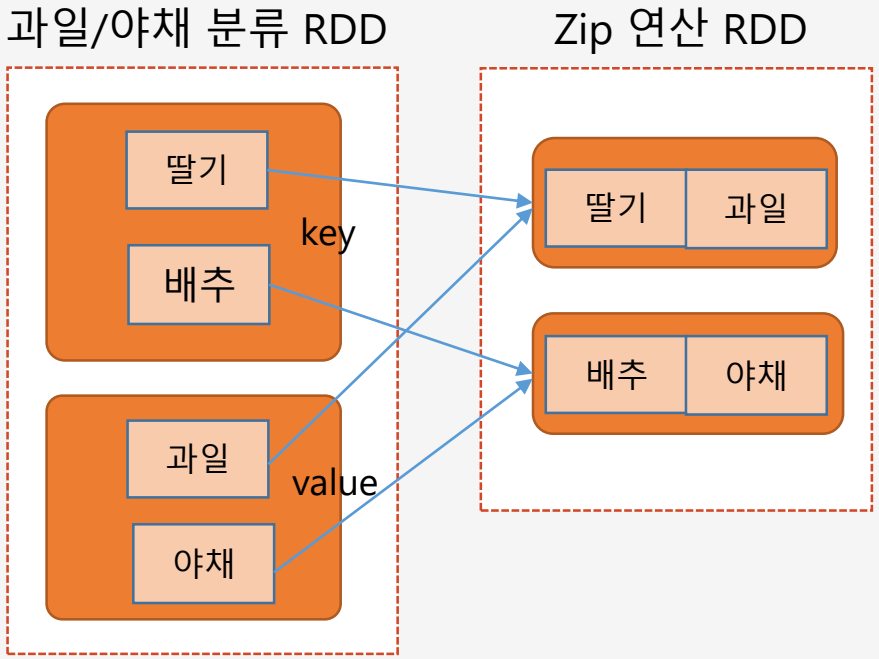


3. Transformation 메서드

- filter() 예 : 필터링 연산

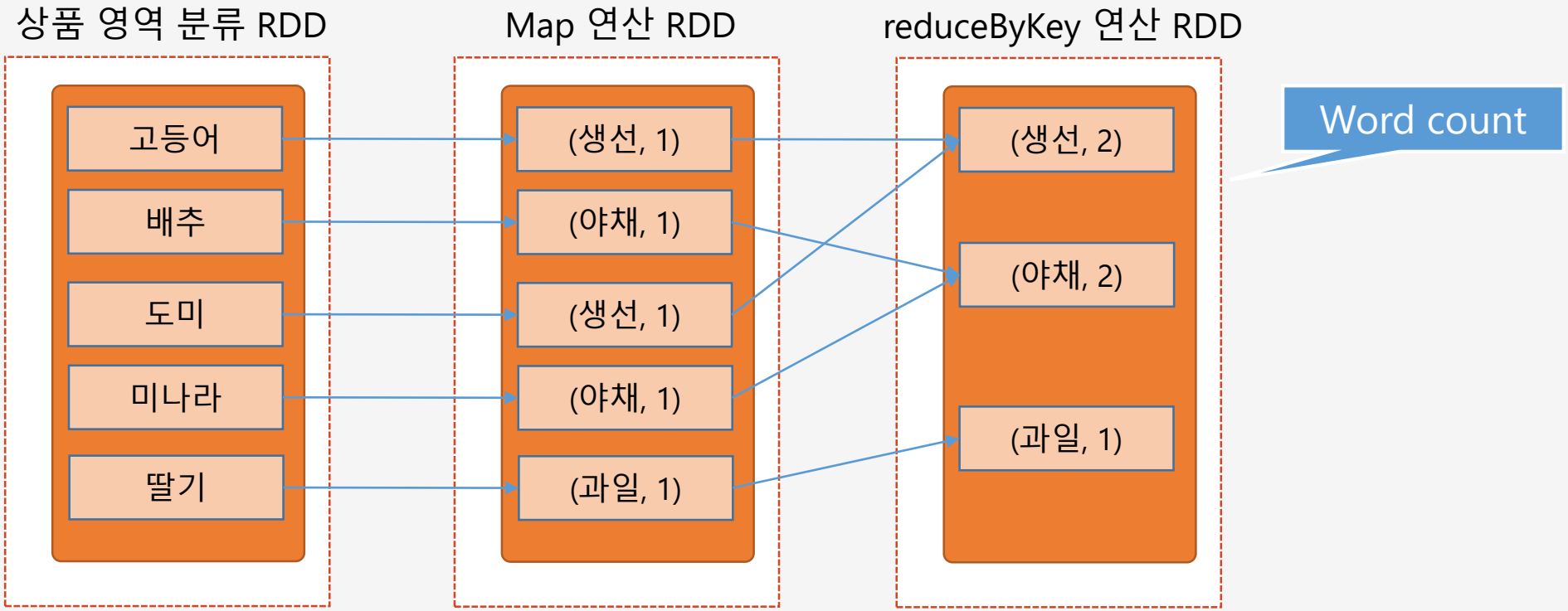


- zip() 예 : 파티션 수와 요소 수 동일 RDD 조합



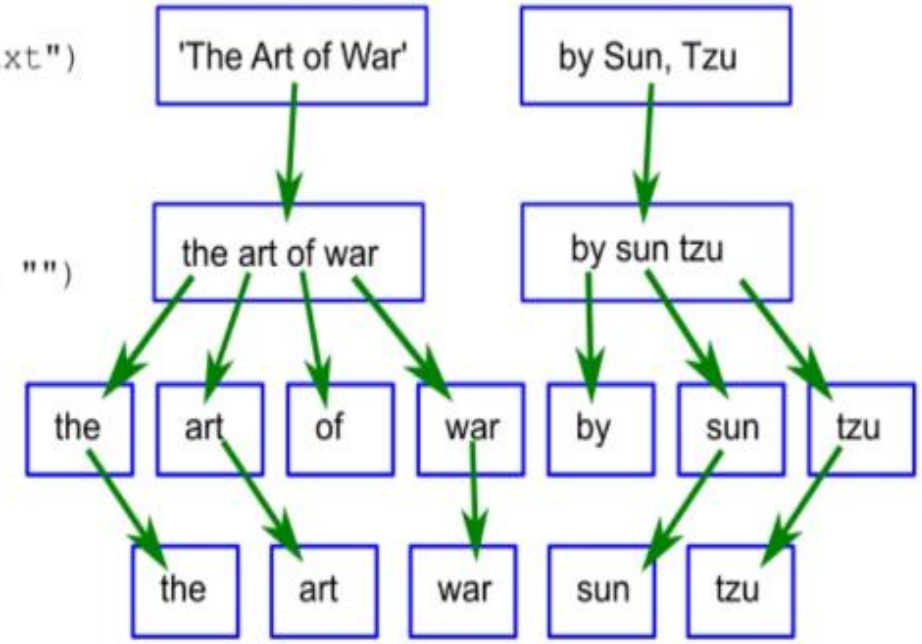
3. Transformation 메서드

- reduceByKey() 예 : 같은 key 기준으로 집약 처리 연산



3. Transformation 메서드

```
sc.textFile("artofwar.txt")
.map(
  _.toLowerCase
   .replaceAll("[^\\w ], "")
 .flatMap(_.split(" "))
   .filter(_.size >2)
```

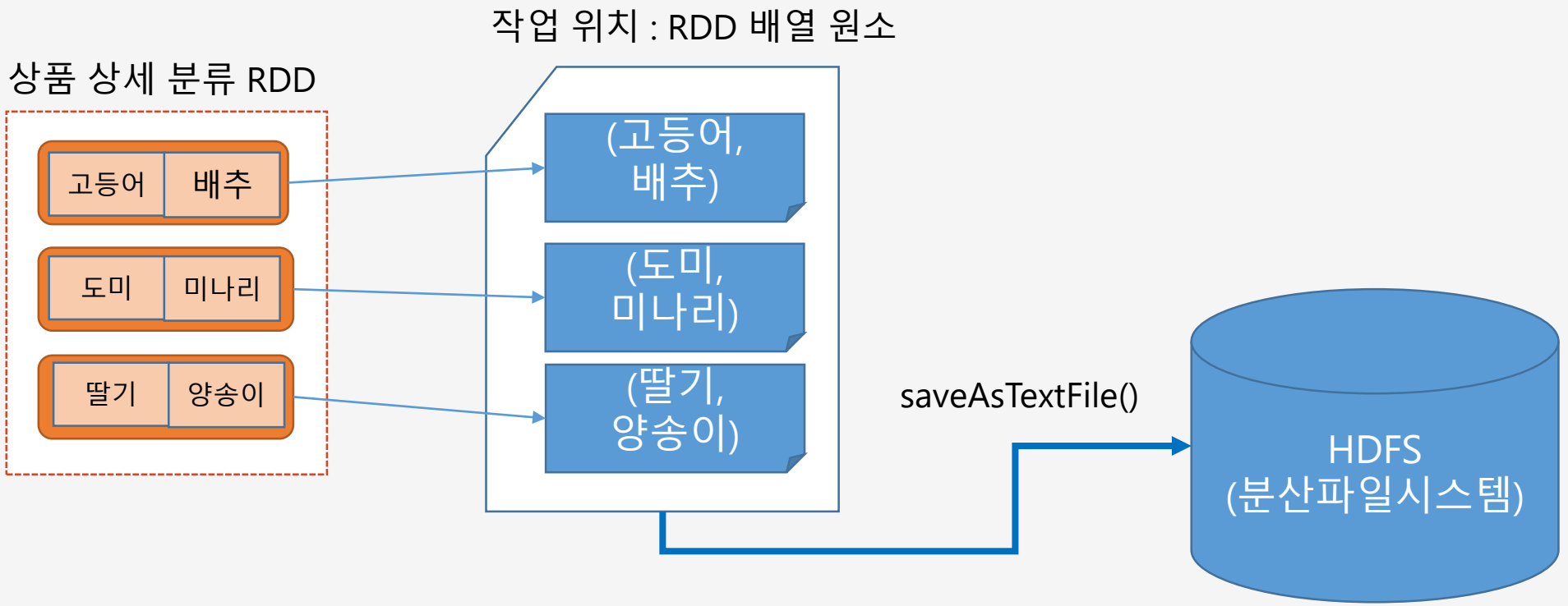


4. Action 메서드

1. **reduce(func) :** 두 개의 인수를 사용하고 하나를 반환하는 함수를 사용하여 데이터 집합의 요소를 집계
2. **collect() :** 데이터 셋의 모든 요소를 배열로 반환
3. **count() :** 데이터 세트의 요소 수를 반환
4. **first() :** 데이터 세트의 첫 번째 요소를 반환
5. **take(n) :** 데이터 집합의 처음 n 개 요소가 포함 된 목록을 반환
6. **countByKey() :** (k, v) 유형의 RDDS에 적용될 때 각 키의 개수와 함께 (k, int) 쌍을 반환
7. **saveAsTextFile(path) :** 데이터 파일의 요소를 로컬 파일 시스템, HDFS 또는 기타 Hadoop 지원 파일 시스템의 지정된 디렉토리에 **텍스트 파일**

4. Action 메서드

- collect() 예 : RDD 원소를 배열로 가져오는 연산
- saveAsTextFile() 예 : HDFS에 RDD 원소 저장



5. RDD 실습 : RDD 생성과 출력

```
object Step01_rddCreate {
    def main(arr: Array[String]) {

      val conf = new SparkConf()
          .setAppName("RDD Create")
          .setMaster("local")
      val sc = new SparkContext(conf)

      // 1. 파티션 지정 RDD 생성 : parallelize(data, 파티션수) 이용 : 2.1.3절 예제 2-6
      val rdd1 = sc.parallelize(1 to 100, 5) // data : vector 이용
      // RDD read -> 구분자 기준 원소 출력
      println(rdd1.collect().mkString(", ")) // 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, ..., 100

      val rdd1_2 = sc.parallelize(List("a", "b", "c", "d", "e")) // data : List collection
      println(rdd1_2.collect().mkString(", ")) // a, b, c, d, e


      // 2. 외부 저장 매체(file, HDFS) 데이터 이용 RDD 생성 : spark_home/README.md
      val rdd2 = sc.textFile("file:/C:/Spark/spark-2.4.5-bin-hadoop2.7/README.md")
      // ("file:/c:/localDir/file")

      // 텍스트 파일의 한 줄은 한 개의 RDD 구성요소
      println(rdd2.collect().mkString("\n")) // 줄바꿈 구분자 이용 줄 단위 출력

      sc.stop
  }
}
```