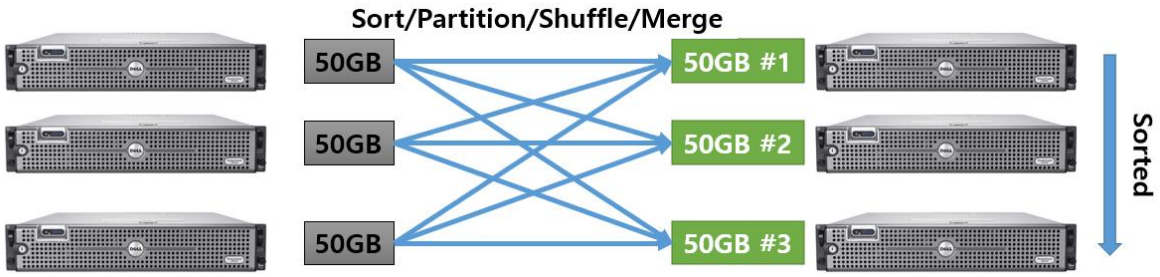
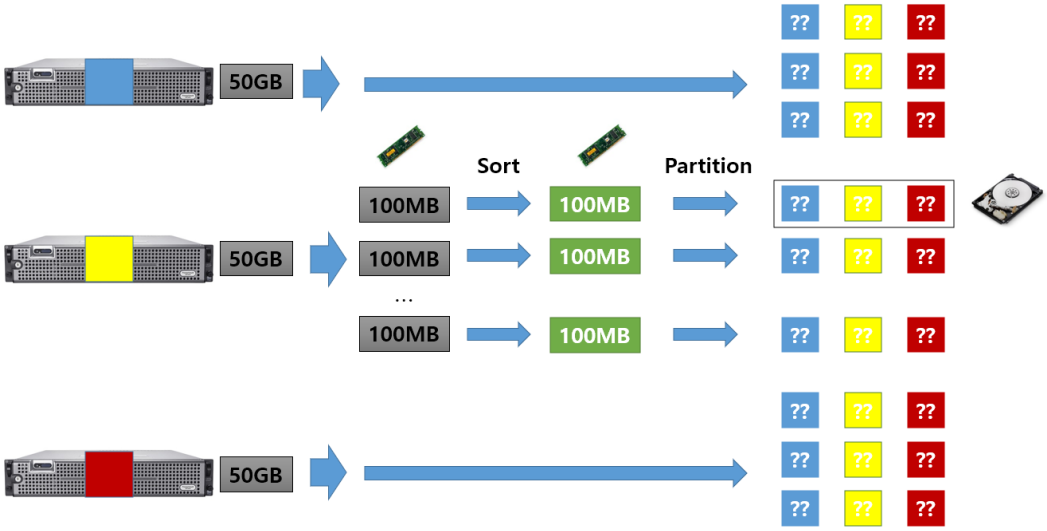
Distributed sorting을 이해하는데 있어 잘못된 개념을 파악한 것이 확인되어 이를 보완하여 다시 정리를 해보겠다.



전체적으로 sort->partition->shuffle->merge의 과정을 거친다.

이때 partition을 할 pivot 값들을 구하는 과정에서 master machine의 도움을 받는다.



위 과정에서 partition을 진행하기 전에 각 machine에서 sampling을 진행한다. 이는 sorting되기 전에 각 worker machine에서 1MB 이하의 데이터를 master machine에 보낸 뒤 master machine에서 sorting한 후에 worker machine 수 만큼 part를 나눠야 하기 때문에 worker machine 수-1 만큼의 pivot data를 추출하여 각 worker machine에 전달한다.

이후의 과정은 전과 같은 방식으로 진행된다. Part를 나눈 뒤 각 part를 하나의 machine에 각각 저장시킨 다음 disk 기반 merge를 진행한다.

정리를 하던 중에 하나의 궁금증이 생겼다.

교수님께서 알려주신 잘못된 implementation의 예시는 다음과 같다.

- Wrong implementation
  - Ex. Master collects all the input data from workers.
  - Ex. Master collects all the sorted data from workers and perform merging.

우리의 기존 구현 방식은 master가 worker의 data를 모두 가지거나 worker machine의 sorted된 data 전부를 input으로 받아 merge하는 방법이 아닌 disk-parallel sort를 진행한 다음 processor별로 pivot 값들을 각각 뽑은 다음 그 값들을 master machine에 보낸 다음 partition을 진행할 pivot data를 추출하는 과정을 고안했다.

이런 방식으로 진행하면 master machine에는 sorted 된 (전체 worker machine의 processor 수)만큼의 array가 각각 (worker machine수-1)만큼의 data를 가지고 sorted 된 채로 전달이 되면 master machine은 이를 merge하여 전체 partition에 쓰일 (worker machine 수-1)개의 pivot data를 worker machine에 각각 전달하면 된다.

이렇게 되면 1MB 이하의 data를 무작위로 sampling하는 것 보다 더 정확하게 partition을 나눌 수 있을 것이라 생각된다.

하지만 속도적인 측면에서는 두 방법의 차이가 있을 것이라 생각되어 어떤 방식으로 구현해야 할지 회의를 진행하고 있다.