[졸업 프로젝트]

Processing-in-Memory 활용 기술 개발

- 월별 보고서 -

한양대학교 컴퓨터소프트웨어학부
2021019961 장서연 2021097356 김한결

2024.06.20.

목차

1. 6월 계획

- 데이터에 관한 논의
- UPMEM-PIM을 활용한 sort-merge join 알고리즘 구현 시작

2. 진행 사항

6월은 5월에 마치지 못했던 데이터와 관련된 논의를 마무리하였고, 설계한 알고리즘의 디자인을 수정하여 최종 완성하였다.

2.1. 데이터에 관한 논의

첫 번째 논의는 input 데이터의 구조이다. UPMEM-PIM을 활용하지 않은 sort-merge join 알고리즘의 구현에서는 input으로 주어진 데이터의 구조를 미리 알고 그것에 맞는 알고리즘을 코드로 구현하였다. 이 과정은 sort-merge join의 동작 과정과 방식 및 구조를 파악하고 성능 개선점을 찾기 위함이다. UPMEM-PIM을 활용한 sort-merge join 알고리즘의 구현 단계에서는 input으로 주어진 데이터를 고정하지 않고 파일을 읽으며 해당 데이터를 파악할 수 있도록 한다. 데이터의 row를 읽으면 feature가 ‘,’으로 구분되어 있으므로 해당 데이터가 몇 개의 feature로 이루어져 있는지 알 수 있다. 또한 join column은 사용자가 입력하는 것을 설정하도록 한다. 사용자에게 첫 번째 table의 몇 번째 column과 두 번째 table의 몇 번째 column을 join할 것인지 두 개의 정수를 input으로 받는다.

두 번째 논의는 성능 측정을 위한 테스트 케이스 생성 방식이다. Python의 ‘random’ 모듈을 사용해 현재 시간 또는 운영 체제에서 제공하는 무작위 소스를 기반으로 한 난수를 생성하여 데이터로 사용한다.

2.2. UPMEM-PIM을 활용한 sort-merge join 알고리즘 디자인

지난 5월 보고서에서는 join 과정의 디자인이 완성되지 않았고, 이에 대해 두 가지 방식을 제안했다. 첫 번째는 정렬된 table들을 각각 쪼개 tasklet 단위로 비교하여 join하는 것이고, 두 번째는 UPMEM-PIM을 활용하지 않는 기존의 sort-merge join 알고리즘과 동일하게 전체 데이터를 sequential하게 join하는 것이다.

전자의 경우, T1과 T2를 모두 쪼개어 tasklet에 할당해 데이터를 검색, 비교하며 탐색한다. UPMEM-PIM에서는 모든 DPU 간의 통신은 호스트 CPU를 통해 DPU에서 CPU로 결과를 검색하고 CPU에서 DPU로 데이터를 복사하는 방식으로 이루어지며, takslet간의 통신은 불가하다. 그러므로 다른 DPU의 tasklet 단위 또는 같은 DPU의 tasklet 단위로 비

교를 하는 방식을 사용했을 때, 최악의 경우 T1의 가장 앞 부분 데이터가 T2의 가장 마지막 부분에 포함되어 있다면 CPU-DPU 이동과 DPU-CPU 이동이 매우 잦아져 cost가 커진다. 또한 lock을 사용하게 되면서 속도는 더욱 느려질 것이라고 예상된다.

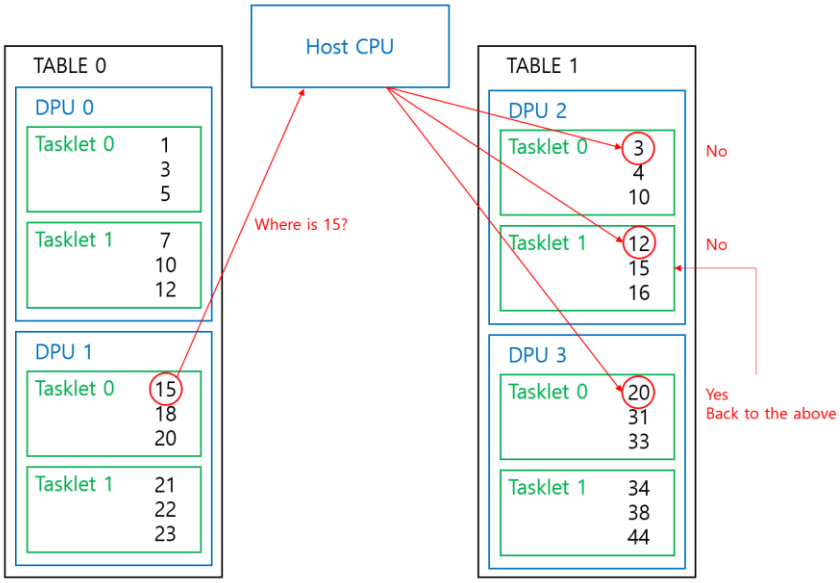

그림 1

수정 이전의 방식. tasklet 간 통신과 DPU 간 통신이 되지 않으므로 CPU와 DPU 간의 이동이 잦아진다.

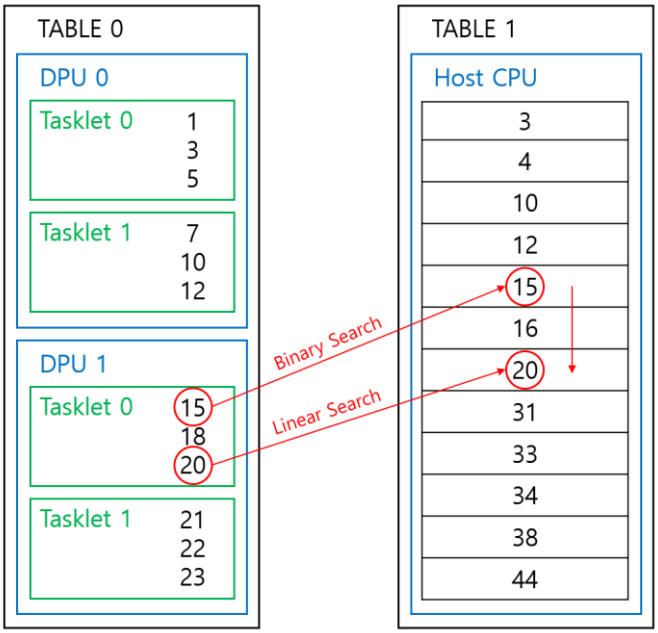

그림 2

수정 후 완성된 방식. CPU와 DPU 간의 이동이 줄어든다.

따라서 수정된 새로운 join 방식을 고안함으로써 UPMEM-PIM을 활용한 sort-merge join 알고리즘의 디자인을 완성했다. 정렬을 마친 두 table은 DPU 상에 위치하지 않고 메인 메모리 상에 존재한다. 이 중 하나의 table만 다시 tasklet 단위로 쪼개어 DPU를 할당한다. 이렇게 한다면 각 tasklet에서는 다른 table의 전체 데이터를 요청해야 하지만, 한 번의 CPU-DPU 이동만 요구된다. 이후 각 tasklet이 sequential하게 데이터를 읽으며 join을 수행하고 완료 시 메인 메모리에 올려 tasklet 순서대로 데이터를 병합한다.

추가적인 최적화 방법으로는 search 방식의 개선이 있다. sequential하게 데이터를 비교하지 않고, 현재 tasklet의 0번째 데이터가 다른 table내 어디에 위치하는지 binary search 등을 적용해 알아내면, 그 이후의 데이터는 sequential하게 탐색할 수 있다.

3. 7월 계획

- UPMEM-PIM을 활용한 sort-merge join 알고리즘 구현 시작