요약정리

- 일괄 처리
  - 유닉스 도구 및 이의 설계 철학이 어떻게 맵리듀스와 데이터플로 엔진에 녹아있는지
  - 설계 원리 : 입력은 불변 & 출력은 다른 프로그램의 입력. 복잡한 문제는 '한가지 일을 잘하는' 작은 도구를 엮어서 해결
- 인터페이스
  - 유닉스 환경에서의 프로그램 간 연결하는 단일 인터페이스 => 파일, 파이프
  - 맵리듀스의 인터페이스 => 분산 파일 시스템
  - 데이터 플로 엔진 => 자체 데이터 전송 메커니즘 (입출력 HDFS 사용)
- 분산 일괄 처리 프레임 워크의 해결해야할 2가지 문제
  - 파티셔닝
    - 매퍼 : 입력 파일 블록에 따라 파티셔닝
    - 리듀서 : 매퍼의 출력 재파티셔닝 & 정렬 => 사용자 지정 파티션 개수로 병합
    - (데이터 플로 엔진은 필요한 경우가 아니면 정렬 x)
  - 내결함성
    - 맵리듀스는 번번히 디스크에 기록
    - 데이터플로 엔진은 메모리에 상태 유지 (중간 상태를 최대한 구체화 x)
    - (결정적 연산자로 재계산 필요 데이터량 절약 가능)
- 맵리듀스의 조인 알고리즘 (=> MPP DB, 데이터플로 엔진 내부에서 사용)
  - 정렬 병합 조인
    - 각 입력이 조인 키를 추출하는 매퍼 통과 => 파티셔닝, 정렬, 병합 => 같은 키를 가지는 모든 레코드는 하나의 리듀서에서 호출 (=> 병합된 레코드 출력가능)
  - 브로드캐스트 해시 조인
    - 조인할 입력 둘 중 하나가 상대적으로 작은 경우, 파티셔닝하지 않고 해시 테이블에 모두 적재
  - 파티션 해시 조인

        - 조인 입력 두개를 같은 방식으로 파티셔닝 (같은 키, 해시함수, 파티션 수) => 각 파티션 별 독립적 해시 테이블 방식 사용
- 분산 일괄 처리 엔진은 의도적으로 제한된 프로그래밍 모델 제공
    - 매퍼/리듀서같은 콜백함수는 상태정보 x, 지정된 출력외 부수 효과 x …
    - why? 분산 시스템 내 발생하는 문제들을 추상화 아래로 숨길 수 있음
    - 문제발생해도 태스크는 안전한 재시도, 실패 태스크의 출력은 폐기 => 최종 출력이 결함이 생기지 않을 때와 동일 보장
    - => 내결함성 매커니즘 구현 필요 x (신뢰성의 시맨틱)
- 일괄 처리 작업의 특징
    - 입력을 수정하지 않고 읽어 출력 생산
    - 입력 데이터는 고정된 크기로 한정 (끝 판단 및 작업 종료 가능)
    - vs 스트림 처리 => 입력이 한정되지 않고 끝이 없음