인-메모리 데이터 처리 기반 S/W의 등장

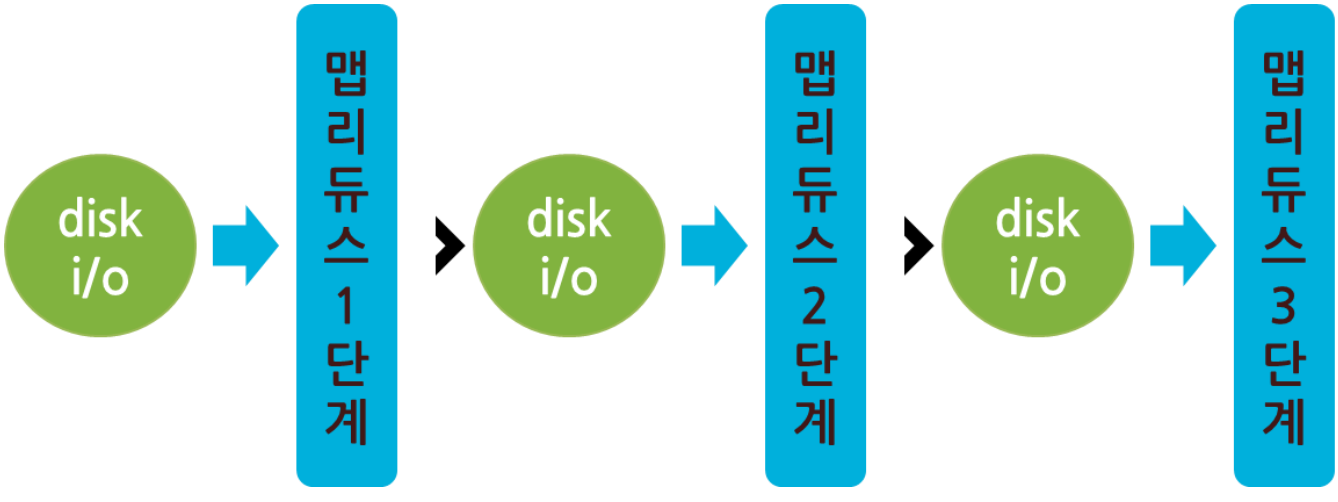
1 하둡의 문제점

① 대용량 일괄 배치 처리에는 효율적

② 맵리듀스 사용 시 데이터 저장·전송 단계에서 많은 디스크 입출력과 네트워크 트래픽 발생

1. 실시간 데이터 처리에는 비효율적
2. 비동기적으로 발생하는 데이터 처리에는 비효율적
3. 반복 작업이 많은 경우에는 비효율적

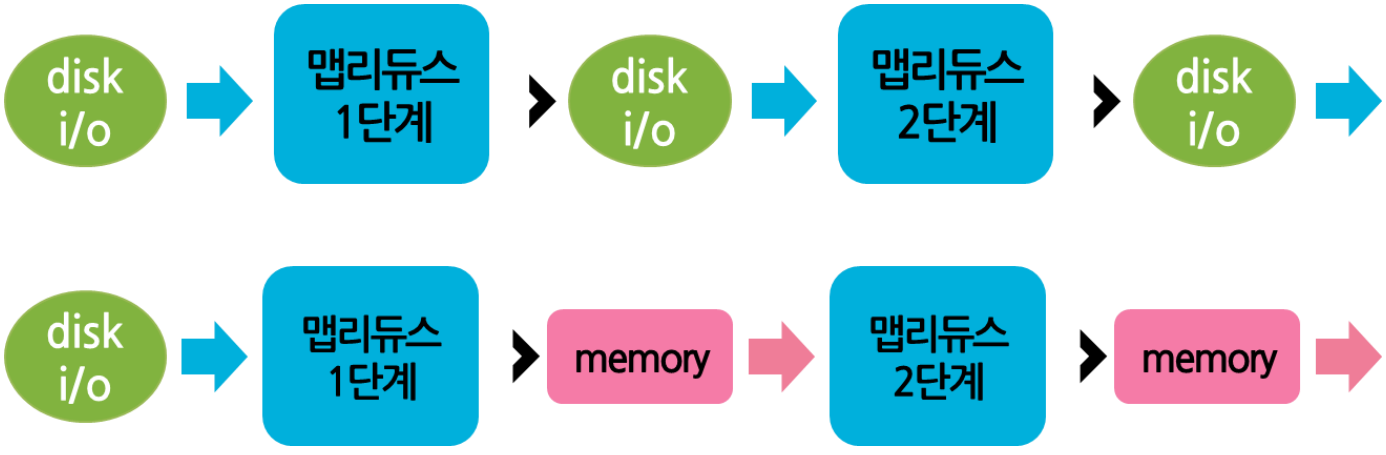
③ 여러 번 맵리듀스 수행에 잦은 디스크 입출력 필요
→ 성능 저하 발생



인-메모리 데이터 처리 기반 S/W의 등장

2 하둡의 문제점 해결 방안

① 디스크 입출력 방식을 인-메모리 데이터 처리 방식으로 전환



디스크 입출력

최초 데이터 로드와 최종 결과 저장 시에만 사용

중간 결과

메모리에 분산 저장하고 병렬 처리 구조로 변경

효과

기존 맵리듀스 디스크 입출력 방식보다 평균 10 ~ 100배 정도의 속도 향상

인-메모리 데이터 처리 기반 S/W의 등장

3 인-메모리 데이터 처리 기반 S/W 등장

① 디스크 입출력 방식을 인-메모리 데이터 처리 방식으로 전환

인-메모리 데이터 처리 기반 S/W의 등장

4 스파크 개요

UC 버클리의 AMP 랩에서 개발

인-메모리 방식의 분산 처리 시스템

개발 이유

기존 디스크 입출력에 대한 지연 시간 개선을 위해

메모리를 사용하여 반복적인 작업이나 스트리밍 데이터를 효율적으로 처리하기 위해

인-메모리 데이터 처리 기반 S/W의 등장

4 스파크 개요

UC 버클리의 AMP 랩에서 개발

인-메모리 방식의 분산 처리 시스템

개발 이유

기존 디스크 입출력에 대한 지연 시간 개선을 위해

메모리를 사용하여 반복적인 작업이나 스트리밍 데이터를 효율적으로 처리하기 위해

2014년 5월 버전 1.0 출시

2016년 7월 버전 2.0 출시

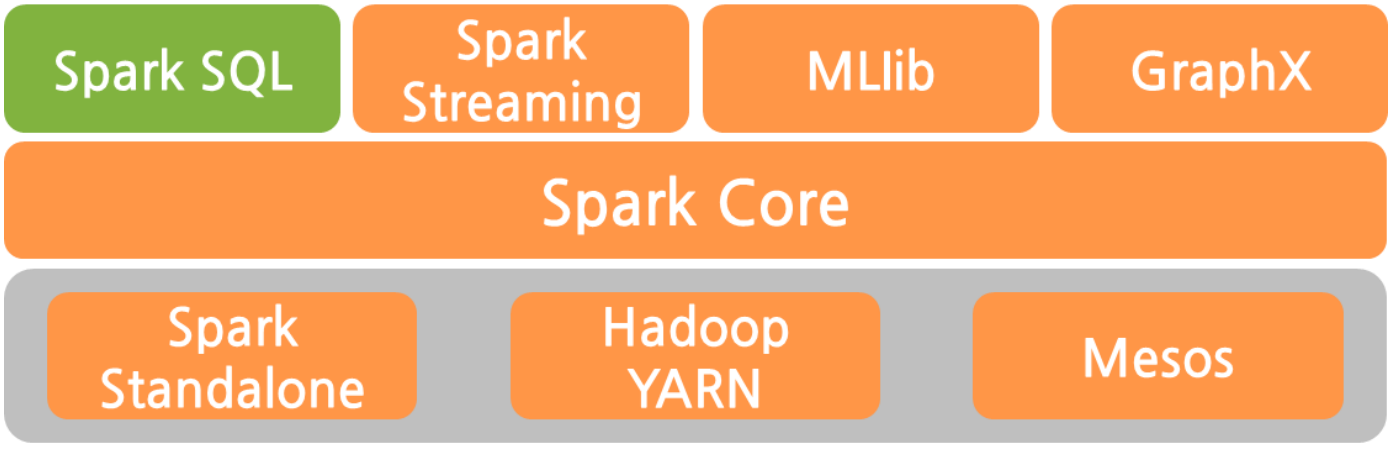
2017년 7월 버전 2.2 출시

인-메모리 데이터 처리 기반 S/W의 등장

5 스파크 주요 요소 및 기능

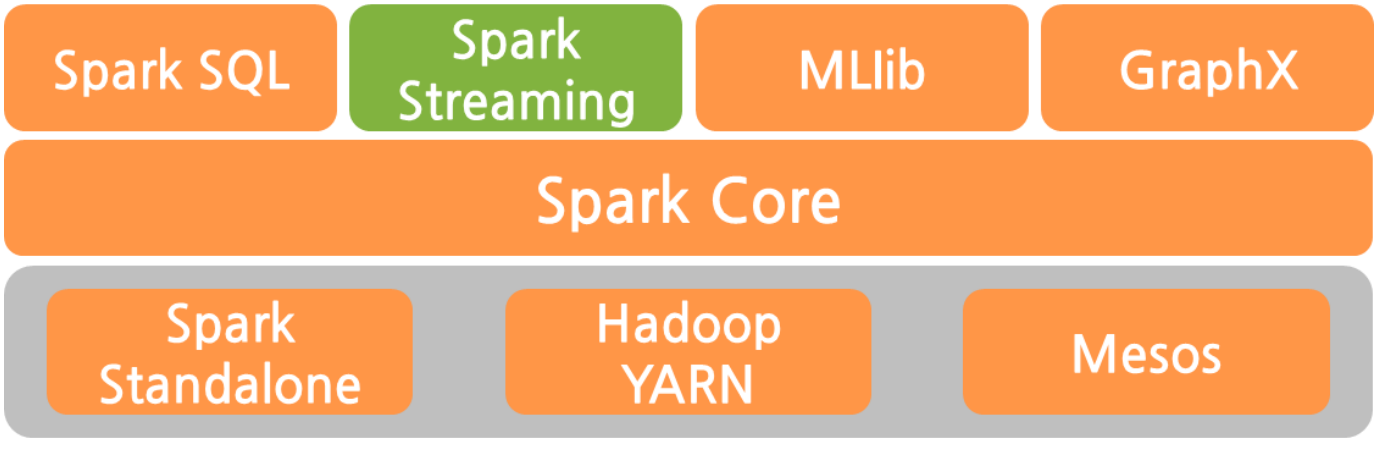
① 스파크 SQL

✓ SQL 기반으로 쿼리 수행



② 스파크 스트리밍

✓ 데이터 스트림을 개별 세그먼트로 나눈 후 각 세그먼트의 데이터를 스파크 엔진으로 처리함

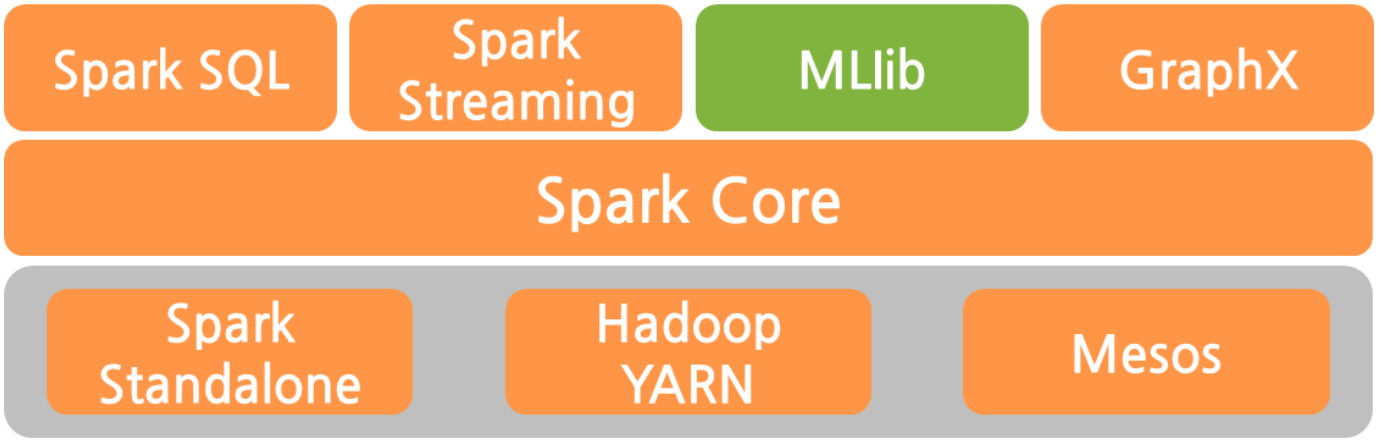


인-메모리 데이터 처리 기반 S/W의 등장

5 스파크 주요 요소 및 기능

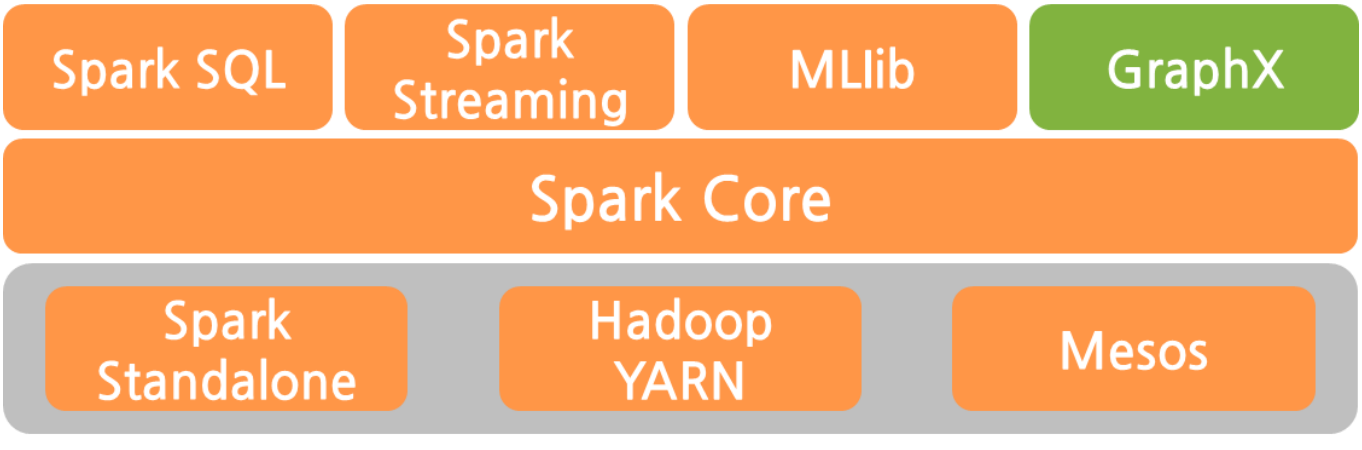
③ 스파크 MLlib

✓ 머신 러닝 라이브러리(선형 및 로지스틱 회귀, 서포트 벡터 머신, 의사 결정 트리, 랜덤 포레스트, k-평균 군집화, SVD 등)



④ 스파크 GraphX

✓ 그래프 라이브러리(페이지랭크, 레이블 전파, 삼각 계수 등의 그래프 알고리즘 지원)

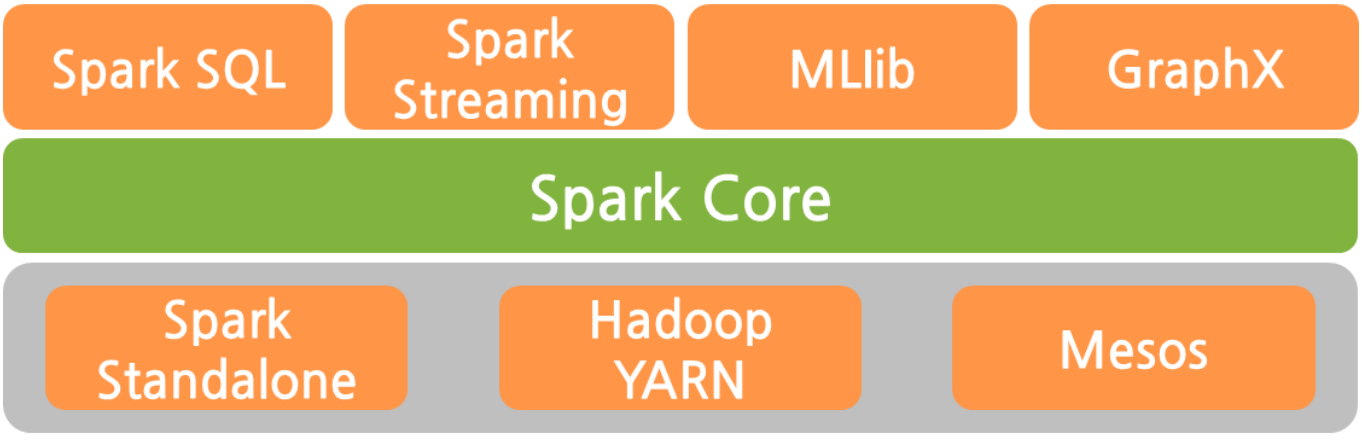


인-메모리 데이터 처리 기반 S/W의 등장

5 스파크 주요 요소 및 기능

5 스파크 코어

- ✓ 스파크 전체의 기초가 되는 분산 작업 처리, 스케줄링, 입출력 기능, API 인터페이스(자바, 스칼라, 파이썬, R 등) 기능 제공



6 스파크 작업 처리

- ✓ 스파크 단독으로 운영 가능
- ✓ 하둡 자원 관리를 담당하는 YARN이나 자체 개발한 Mesos을 통해 처리