

대분류/20
정보통신

중분류/01
정보기술

소분류/02
정보기술개발

세분류/09
빅데이터플랫폼구축

능력단위/05

NCS 학습모듈

빅데이터 처리시스템 개발

LM2001020905_17v1



교육부

NCS 학습모듈은 교육훈련기관에서 출처를 명시하고 교육적 목적으로 활용할 수 있습니다. 다만 NCS 학습모듈에는 국가(교육부)가 저작재산권 일체를 보유하지 않은 저작물들(출처가 표기되어 있는 도표, 사진, 삽화, 도면 등)이 포함되어 있으므로 이러한 저작물들의 변형, 복제, 공연, 배포, 공중 송신 등과 이러한 저작물들을 활용한 2차 저작물의 생성을 위해서는 반드시 원작자의 동의를 받아야 합니다.

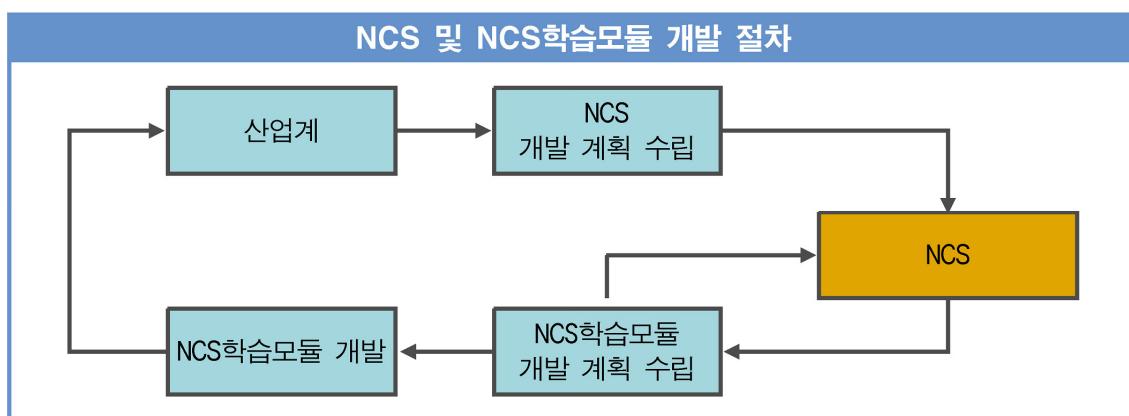
NCS학습모듈의 이해

* 본 NCS학습모듈은 「NCS 국가직무능력표준」 사이트(<http://www.ncs.go.kr>)에서 확인 및 다운로드 할 수 있습니다.

I

NCS학습모듈이란?

- 국가직무능력표준(NCS: National Competency Standards)이란 산업현장에서 직무를 수행하기 위해 요구되는 지식·기술·소양 등의 내용을 국가가 산업부문별·수준별로 체계화한 것으로 산업현장의 직무를 성공적으로 수행하기 위해 필요한 능력(지식, 기술, 태도)을 국가적 차원에서 표준화한 것을 의미합니다.
- 국가직무능력표준(이하 NCS)이 현장의 ‘직무 요구서’라고 한다면, NCS학습모듈은 NCS의 능력단위를 교육훈련에서 학습할 수 있도록 구성한 ‘교수·학습 자료’입니다. NCS학습모듈은 구체적 직무를 학습할 수 있도록 이론 및 실습과 관련된 내용을 상세하게 제시하고 있습니다.

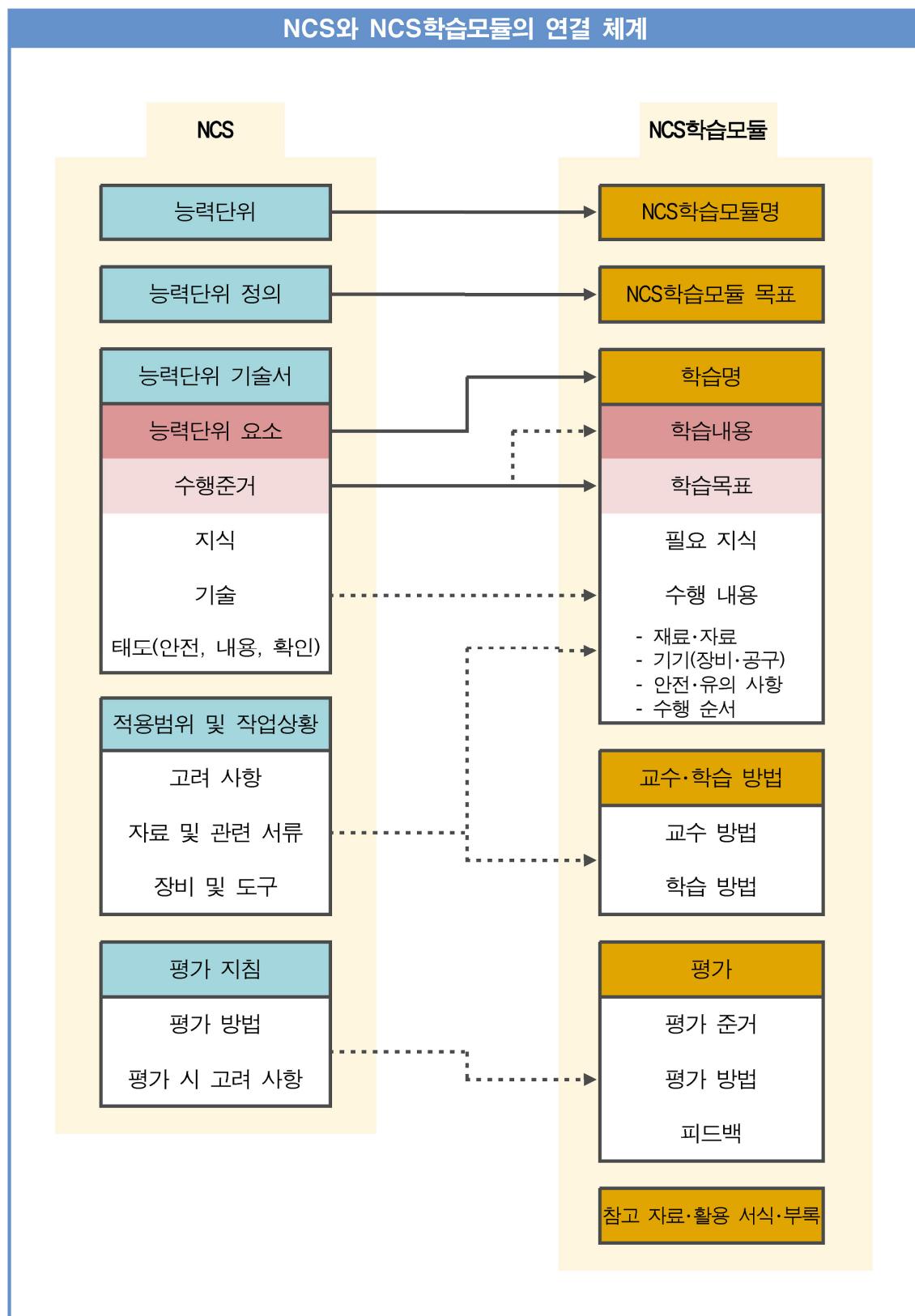


- NCS학습모듈은 다음과 같은 특징을 가지고 있습니다.

첫째, NCS학습모듈은 산업계에서 요구하는 직무능력을 교육훈련 현장에 활용할 수 있도록 성취목표와 학습의 방향을 명확히 제시하는 가이드라인의 역할을 합니다.

둘째, NCS학습모듈은 특성화고, 마이스터고, 전문대학, 4년제 대학교의 교육기관 및 훈련기관, 직장교육기관 등에서 표준교재로 활용할 수 있으며 교육과정 개편 시에도 유용하게 참고할 수 있습니다.

- NCS와 NCS학습모듈 간의 연결 체계를 살펴보면 아래 그림과 같습니다.



II NCS학습모듈의 체계

- NCS학습모듈은 1. NCS학습모듈의 위치, 2. NCS학습모듈의 개요, 3. NCS학습모듈의 내용 체계, 4. 참고 자료, 5. 활용서식/부록으로 구성되어 있습니다.

1. NCS학습모듈의 위치

- NCS학습모듈의 위치는 NCS 분류 체계에서 해당 학습모듈이 어디에 위치하는지를 한 눈에 볼 수 있도록 그림으로 제시한 것입니다.

[NCS-학습모듈의 위치]		
대분류	문화 · 예술 · 디자인 · 방송	
중분류	문화콘텐츠	
소분류	문화콘텐츠제작	
세분류		
방송콘텐츠제작	능력단위	학습모듈명
영화콘텐츠제작	프로그램 기획	프로그램 기획
음악콘텐츠제작	아이템 선정	아이템 선정
광고콘텐츠제작	자료 조사	자료 조사
게임콘텐츠제작	프로그램 구성	프로그램 구성
애니메이션 콘텐츠제작	캐스팅	캐스팅
만화콘텐츠제작	제작계획	제작계획
캐릭터제작	방송 미술 준비	방송 미술 준비
스마트문화 콘텐츠제작	방송 리허설	방송 리허설
영사	야외촬영	야외촬영
	스튜디오 제작	스튜디오 제작

학습모듈은

NCS 능력단위 1개당 1개의 학습모듈 개발을 원칙으로 합니다. 그러나 필요에 따라 고용단위 및 교과단위를 고려하여 능력단위 몇 개를 묶어 1개 학습모듈로 개발할 수 있으며, NCS 능력단위 1개를 여러 개의 학습모듈로 나누어 개발 할 수도 있습니다.

2. NCS학습모듈의 개요

○ NCS학습모듈의 개요는 학습모듈이 포함하고 있는 내용을 개략적으로 설명한 것으로 **학습모듈의 목표**,

선수 학습, **학습모듈의 내용 체계**, **핵심 용어**로 구성되어 있습니다.

학습모듈의 목표	해당 NCS 능력단위의 정의를 토대로 학습목표를 작성한 것입니다.
선수 학습	해당 학습모듈에 대한 효과적인 교수·학습을 위하여 사전에 이수해야 하는 학습모듈, 학습 내용, 관련 교과목 등을 기술한 것입니다.
학습모듈의 내용 체계	해당 NCS 능력단위요소가 학습모듈에서 구조화된 체계를 제시한 것입니다.
핵심 용어	해당 학습모듈의 학습 내용, 수행 내용, 설비·기자재 등 가운데 핵심적인 용어를 제시한 것입니다.

제작계획 학습모듈의 개요

학습모듈의 목표

본격적인 촬영을 준비하는 단계로서, 촬영 대본을 확정하고 제작 스태프를 조직하며 촬영 장비와 촬영 소품을 준비할 수 있다.

학습모듈의 목표는

학습자가 해당 학습모듈을 통해 성취해야 할 목표를 제시한 것으로, 교수자는 학습자가 학습모듈의 전체적인 내용흐름을 파악하도록 지도할 수 있습니다.

선수학습

제작 준비(LM0803020105_13v1), 섭외 및 제작스태프 구성(LM0803020104_13v1), 촬영 제작(LM0803020106_13v1), 촬영 장비 준비(LM0803040204_13v1.4), 미술 디자인 협의하기(LM0803040203_13v1.4)

선수학습은

교수자 또는 학습자가 해당 학습모듈을 교수·학습하기 이전에 이수해야 하는 교과목 또는 학습모듈(NCS 능력단위) 등을 표기한 것입니다. 따라서 교수자는 학습자가 개별 학습, 자기 주도 학습, 방과 후 활동 등 다양한 방법을 통해 이수할 수 있도록 지도하는 것을 권장합니다.

학습모듈의 내용체계

학습	학습 내용	NCS 능력단위 요소 코드번호	요소 명칭
1. 촬영 대본 확정하기	1-1. 촬영 구성안 검토와 수정	0803020114_16v3.1	촬영 대본 확정하기
2. 제작 스태프 조직하기	2-1. 기술 스태프 조직 2-2. 미술 스태프 조직 2-3. 전문 스태프 조직	0803020114_16v3.2	제작 스태프 조직하기
3. 촬영 장비 계획하기	3-1. 촬영 장비 점검과 준비	0803020114_16v3.3	촬영 장비 계획하기
4. 촬영 소품 계획하기	4-1. 촬영 소품 목록 작성 4-2. 촬영 소품 제작 의뢰	0803020114_16v3.4	촬영 소품 계획하기

핵심 용어는

해당 학습모듈을 대표하는 주요 용어입니다. 학습자가 해당 학습모듈을 통해 학습하고 평가받게될 주요 내용을 알 수 있습니다. 「NCS 국가직무능력 표준」사이트 (www.ncs.go.kr)의 색인(찾아보기) 중 하나로 이용할 수 있습니다.

핵심 용어

촬영 구성안, 제작 스태프, 촬영 장비, 촬영 소품

3. NCS학습모듈의 내용 체계

○ NCS학습모듈의 내용은 크게 **학습**, **학습 내용**, **교수·학습 방법**, **평가**로 구성되어 있습니다.

학습	해당 NCS 능력단위요소 명칭을 사용하여 제시한 것입니다. 학습은 크게 학습 내용, 교수·학습 방법, 평가로 구성되며 해당 NCS 능력 단위의 능력단위 요소별 지식, 기술, 태도 등을 토대로 내용을 제시한 것입니다.
학습 내용	학습 내용은 학습 목표, 필요 지식, 수행 내용으로 구성되며, 수행 내용은 재료·자료, 기기(장비·공구), 안전·유의 사항, 수행 순서, 수행 tip으로 구성한 것입니다. 학습모듈의 학습 내용은 실제 산업현장에서 이루어지는 업무활동을 표준화된 프로세스에 기반하여 다양한 방식으로 반영한 것입니다.
교수·학습 방법	학습 목표를 성취하기 위한 교수자와 학습자 간, 학습자와 학습자 간 상호 작용이 활발하게 일어날 수 있도록 교수자의 활동 및 교수 전략, 학습자의 활동을 제시한 것입니다.
평가	평가는 해당 학습모듈의 학습 정도를 확인할 수 있는 평가 준거 및 평가 방법, 평가 결과의 피드백 방법을 제시한 것입니다.

학습 1	촬영 대본 확정하기
학습 2	제작 스태프 조직하기
학습 3	촬영 장비 계획하기
학습 4	촬영 소품 계획하기

2-1. 기술 스태프 조직

학습 목표 •프로그램 제작에 적합한 기술 스태프를 조직할 수 있다.

필요 지식 /

① 기술 스태프의 구성
프로그램의 장르에 따라 구성하는 기술 스태프는 많은 차이가 있다. 같은 장르의 프로그램이라도 그 형식이나 내용, 규모에 따라서 구성되는 기술 스태프의 종류와 인원수는 천차만별이다.

1. 스튜디오 프로그램
토크쇼, 종합 구성, 예능과 같은 스튜디오 프로그램은 부조 정설과 스튜디오를 사용하여 제작하기 때문에 많은 기술 스태프가 필요하다.

학습은
해당 NCS 능력단위요소 명칭을 사용하여 제시하였습니다. 하나의 학습은 일반교과의 '대단원'에 해당되며, 학습모듈을 구성하는 가장 큰 단위가 됩니다. 또한 하나의 직무를 수행하기 위한 가장 기본적인 단위로 사용할 수 있습니다

학습 내용은
NCS 능력단위요소별 수행준거를 기준으로 제시하였습니다. 일반교과의 '종단원'에 해당합니다.

학습 목표는
학습 내용을 이수할 때 학습자가 갖춰야 할 행동 수준을 의미합니다. 따라서 수업시간의 과목 목표로 활용할 수 있습니다.

필요 지식은
해당 NCS의 지식을 토대로 학습에 대한 이해와 성과를 제고하기 위해 반드시 알아야 할 주요 지식을 제시하였습니다. 필요 지식은 수행에 꼭 필요한 핵심 내용을 위주로 제시하여 교수자의 역할이 매우 중요하며, 이후 수행 순서와 연계하여 교수·학습으로 진행할 수 있습니다.

수행 내용 / 기술 스태프 구성표 작성하기

재료 · 자료

- 방송프로그램 제작 기획서 및 방송 대본, 콘티(continuity), 제작 일정, 운용표
- 장비 및 시설, 제작 시설 배정 의뢰서 및 배정표, 방송 기술 스태프 데이터베이스(DB) 자료

기기(장비 · 공구)

- 컴퓨터 등

안전 · 유의 사항

- 프로그램의 내용과 제작 방법을 분석하고, 각 스태프들의 역할을 신중하게 검토한다.

수행 순서

- 방송 대본이나 콘티(continuity), 큐 시트를 분석하고, 프로그램의 내용적 특성, 제작 과정에 대한 자료를 수집한다.
- 프로그램 제작 방법을 결정한다.
 - 스튜디오 녹화를 할 것인가, 야외 촬영을 할 것인가 검토한다.

수행 tip

- 스태프의 결정은 스태프 간의 흐름을 중요시하여 선정해야 프로그램의 질을 향상시킬 수 있다.

수행 내용은

해당 학습모듈에서 제시한 내용 중 기술(skill)을 습득하기 위한 실습과제로 활용할 수 있습니다.

재료 · 자료는

수행 내용을 수행하는데 필요한 재료 및 준비물로 실습 시 활용할 수 있습니다.

기기(장비 · 공구)는

수행 내용에 필요한 기본적인 장비 및 도구를 제시하였습니다. 제시된 기기 외에도 수행에 필요한 다양한 도구나 장비를 활용할 수 있습니다.

안전 · 유의사항은

수행 내용을 수행하는 데 있어 안전 상 주의해야 할 점 및 유의사항을 제시하였습니다. 실습 시 유념해야 하며, NCS의 고려사항도 추가적으로 활용할 수 있습니다.

수행 순서는

실습 과제의 진행 순서로 활용할 수 있습니다.

수행 tip은

수행 내용에서 실습을 용이하게 할 수 있는 아이디어를 제시하였습니다. 수행 tip은 지도상의 안전 및 유의사항 외에 전반적으로 적용되는 주안점 및 수행 과제 목적에 대한 보충설명, 추가사항 등으로 활용할 수 있습니다.

학습2 교수 · 학습 방법

교수 방법

- 방송 프로그램의 기술적 요소, 미술 구성 요소, 특수 촬영에 대해 설명한다.
- 방송 프로그램 제작에서 각 기술 스태프의 역할에 대해 설명한다.
- 방송 프로그램을 분석하고 필요한 기술 스태프를 구성할 수 있도록 지시한다.

학습 방법

- 방송 프로그램의 기술적 요소, 미술 구성 요소, 특수 촬영에 대해서 알아본다.
- 프로그램 제작에 필요한 기술 스태프의 역할을 이해하고, 기술 스태프 구성표를 작성한다.

교수 · 학습방법은

학습 목표를 성취하는 데 필요한 교수 방법과 학습 방법을 제시하였습니다.

교수 방법은

해당 학습 활동에 필요한 학습 내용, 학습 내용과 관련된 자료명, 자료 형태, 수행 내용의 진행 방식 등에 대하여 제시하였습니다. 또한 학습자의 수업 참여도 제고 방법 및 수업 진행상 유의사항 등도 제시하였습니다. 선수학습이 필요한 학습을 학습자가 숙지하였는지 교수가 확인하는 과정으로 활용할 수도 있습니다.

학습 방법은

해당 학습 활동에 필요한 학습자의 자기 주도 학습 방법을 제시하였습니다. 또한 학습자가 숙달해야 할 실기 능력과 학습 과정에서 주의해야 할 사항 등도 제시하였습니다. 학습자가 학습을 이수하기 전 반드시 숙지해야 할 기본 지식을 학습하였는지 스스로 확인하는 과정에 활용할 수 있습니다.

학습2 평가

평가 준거

- 평가자는 학습자가 학습 목표를 성공적으로 달성하였는지를 평가해야 한다.
- 평가자는 다음 사항을 평가해야 한다.

학습 내용	학습 목표	성취수준 상 중 하
기술 스태프 조직	프로그램 제작에 적합한 기술 스태프를 조직할 수 있다.	
미술 스태프 조직	프로그램 제작에 적합한 미술 스태프를 조직할 수 있다.	
전문 스태프 조직	프로그램 특수 촬영을 위한 전문 스태프를 조직할 수 있다.	

평가 방법

- 사례 연구

학습 내용	평가 항목	성취수준 상 중 하
기술 스태프 조직	프로그램에서 기술적 요소의 파악 여부	
	기술 스태프의 역할 파악 여부	
	프로그램에 필요한 기술 스태프 구성표 작성 능력	

피드백

- 사례 연구
 - 프로그램을 선택하여 기술 스태프, 미술 스태프, 전문 스태프 구성표를 예시와 같이 작성하였는지 개인별 능력을 평가한 후, 그 결과를 모든 학습자에게 공유하도록 한다.

평가는
NCS 능력단위의 평가 방법과 평가 시 고려사항을 준용하여 작성합니다. 교수자와 학습자가 평가 항목별 성취 수준 확인 시 활용할 수 있습니다.

평가 준거는
학습자가 학습을 어느 정도 성취하였는지 평가하기 위한 기준을 제시하고 있습니다. 학습 목표와 연계하여 단위 수업 시간에 평가 항목 별 성취수준을 평가하는 데 활용할 수 있습니다.

평가 방법은
NCS 능력단위의 평가 방법을 참고하였으며, 평가 준거에 따른 평가 방법을 2개 이상 제시합니다. 평가 방법의 종류는 포트폴리오, 문제해결 시나리오, 서술형 시험, 논술형 시험, 사례 연구, 평가자 체크리스트, 작업장 평가 등이 있으며, NCS 능력단위 요소 별 수행 수준을 평가하는 데 가정 적절한 방법을 선정하여 활용할 수 있습니다.

피드백은
평가 후에 학습자들에게 평가 결과를 피드백하여 학습 목표를 달성하는 데 활용할 수 있습니다.

4. 참고 자료

참고 자료

- 교육부(2013). 섭외 및 제작스태프 구성(LM0803020104). 한국직업능력개발원.

참고자료는
해당 학습모듈에 제시된 인용 자료의 출처를 제시하였습니다. 교수·학습의 과정에서 참고로 활용할 수 있습니다.

5. 활용 서식/부록

활용 서식

스튜디오 기술 스태프 구성표

직종	이름	연락처	소속	특이사항	비고
기술감독 조명감독					

부록

[디지털 텔레비전 방송프로그램 음량 등에 관한 기준]
제정 2014. 11. 29. 미래창조과학부 고시 제2014-87호
제1창 총칙

제1조(목적) 이 고시는 방송법 제70조의2제1항에 따라 방송사업자가 디지털 텔레비전 방송 프로그램 및 방송광고의 음량을 일정하게 유지하기 위해 필요한 사항을 규정함을 목적으로 한다.

활용 서식은
평가 서식, 실습 시트 등 교수·학습 시 활용할 수 있는 다양한 서식들로 구성하였습니다. 수행에서 평가에 이르기까지 필요한 서식을 해당 모듈의 특성에 맞춰 개발하거나 기준의 양식을 활용하여 제시하였습니다.

부록은
활용 서식 이외에 교수·학습 과정에서 참고할 수 있는 자료가 있는 경우 제시하였습니다.

[NCS-학습모듈의 위치]

대분류	정보통신
중분류	정보기술
소분류	정보기술개발

세분류	능력단위	학습모듈명
SW아키텍처	빅데이터 플랫폼 요구 사항 분석	빅데이터 플랫폼 요구 사항 분석
응용SW엔지니어링	빅데이터 플랫폼 아키텍처 설계	빅데이터 플랫폼 아키텍처 설계
임베디드SW엔지니어링	빅데이터 수집 시스템 개발	빅데이터 수집 시스템 개발
DB엔지니어링	빅데이터 저장 시스템 개발	빅데이터 저장 시스템 개발
NW엔지니어링	빅데이터 처리 시스템 개발	빅데이터 처리 시스템 개발
보안엔지니어링	빅데이터 분석 시스템 개발	빅데이터 분석 시스템 개발
UI/UX엔지니어링	빅데이터 품질 관리 시스템 개발	빅데이터 품질 관리 시스템 개발
시스템SW엔지니어링	빅데이터 플랫폼 테스트	빅데이터 플랫폼 테스트
빅데이터플랫폼구축		
핀테크엔지니어링		
데이터아키텍처		

차 례

학습모듈의 개요	1
학습 1. 빅데이터 처리 시스템 설계하기	
1-1. 빅데이터 처리 시스템 설계	3
• 교수 · 학습 방법	23
• 평가	24
학습 2. 빅데이터 처리 시스템 구성하기	
2-1. 빅데이터 처리 시스템 구성	26
• 교수 · 학습 방법	56
• 평가	57
학습 3. 빅데이터 처리 시스템 수행 모듈 개발하기	
3-1. 빅데이터 분산처리 수행 모듈 개발	59
3-2. 빅데이터 실시간 수행 모듈 개발	83
3-3. 빅데이터 이벤트 처리 수행 모듈 개발	91
• 교수 · 학습 방법	105
• 평가	106
참고 자료	109

빅데이터 처리 시스템 개발 학습모듈의 개요

학습모듈의 목표

저장된 데이터를 처리 목적에 따라 크기, 종류, 저장 구조를 고려하여 처리 및 가공하기 위한 분산 처리, 실시간 처리, 이벤트 처리 모듈을 개발할 수 있다.

선수학습

애플리케이션 요구 사항 분석(2001020219_16v4), 빅데이터 플랫폼 요구 사항 분석(2001020901_17v1), 빅데이터 수집 시스템 개발(2001020903_17v1), 빅데이터 저장 시스템 개발(2001020904_17v1), 빅데이터 플랫폼 테스트(2001020908_17v1)

학습모듈의 내용체계

학습	학습 내용	NCS 능력단위 요소	
		코드번호	요소 명칭
1. 빅데이터 처리 시스템 설계하기	1-1. 빅데이터 처리 시스템 설계	2001020905_17v1.1	빅데이터 처리 시스템 설계하기
2. 빅데이터 처리 시스템 구성하기	2-1. 빅데이터 처리 시스템 구성	2001020905_17v1.2	빅데이터 처리 시스템 구성하기
	3-1. 빅데이터 분산 처리 수행 모듈 개발	2001020905_17v1.3	분산 처리 수행 모듈 개발하기
3. 빅데이터 처리 시스템 수행 모듈 개발하기	3-2 빅데이터 실시간 수행 모듈 개발	2001020905_17v1.4	실시간 수행 모듈 개발하기
	3-3. 빅데이터 이벤트 처리 수행 모듈 개발	2001020905_17v1.5	이벤트 처리 수행 모듈 개발하기

핵심 용어

분산 처리, 실시간 분산 처리, 이벤트 처리, Apache Hadoop, HDFS, SQL-on-Hadoop, Apache Kafka, Apache Spark

학습 1

빅데이터 처리 시스템 설계하기

학습 2

빅데이터 처리 시스템 구성하기

학습 3

빅데이터 처리 시스템 수행 모듈 개발하기

1-1. 빅데이터 처리 시스템 설계

학습 목표

- 빅데이터 관련 기술, 제품, 분석을 고려하여 처리 데이터의 크기, 종류, 저장 구조에 따른 빅데이터 처리 방식을 결정할 수 있다.
- 데이터 처리 모델을 설계하기 위하여 데이터 요구 사항을 도출하고 데이터 유형별로 분류하여 특성을 파악할 수 있다.
- 데이터의 크기, 종류, 저장 구조 특성에 부합하는 데이터 처리 모델 및 제약 조건을 설계할 수 있다.
- 완료된 데이터 처리 모델 설계에 대한 결과의 타당성을 검증할 수 있다.

필요 지식 /

① 빅데이터 처리 시스템의 필요성

기존 데이터 웨어하우스(DW, Data Warehouse)에서 대량의 집계(통계)를 수행하려면 너무 많은 시간이 필요하였다. 이러한 점을 극복하기 위해 데이터 마트(Data Mart)를 생성하여 BI와 같은 도구로 빠르게 데이터를 탐색하고 있다.

하지만, 최근 비즈니스 요건에서는 데이터를 저장 즉시 분석하고자 하는 요건이 늘어나고 있으며 그에 따라 데이터 마트를 생성하는 시간도 줄일 필요성이 나타났다.

빅데이터 처리 시스템은 수집 및 저장된 데이터를 빠르게 탐색 및 집계할 수 있도록 하여 비즈니스 요구 사항을 더 적극적으로 반영할 수 있다.

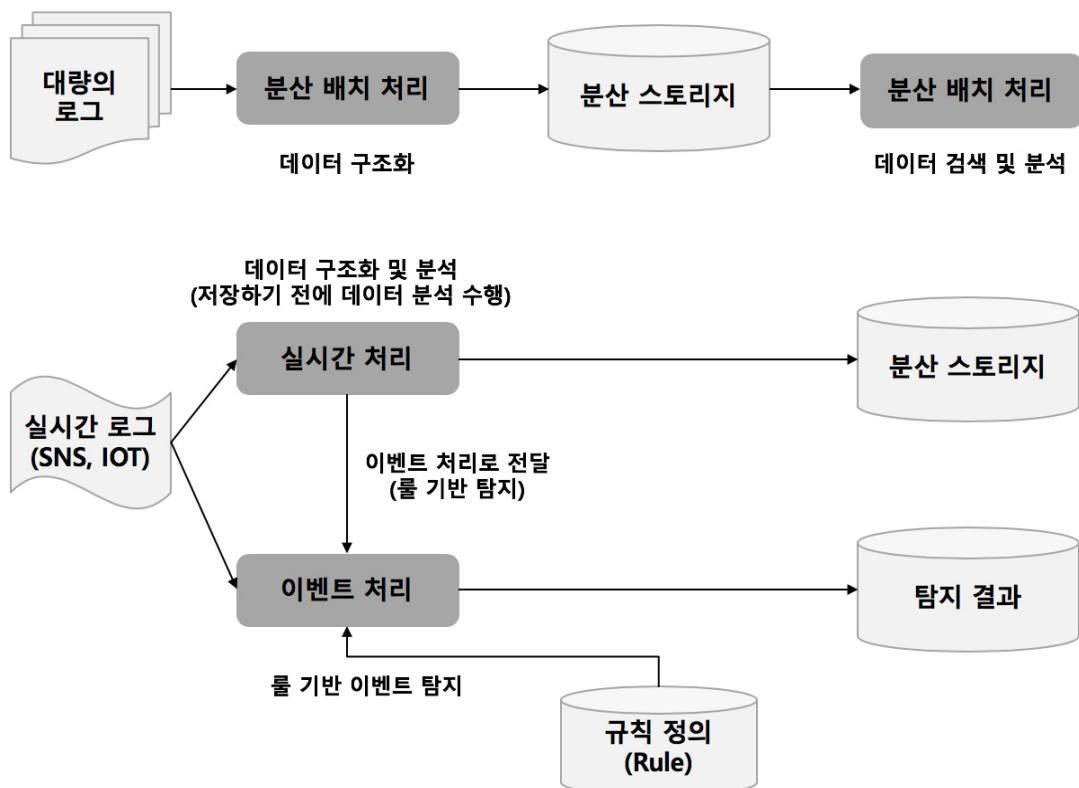
② 빅데이터 처리 시스템

빅데이터 처리 시스템은 기존 기술로 처리하지 못하는 대량의 데이터를 분산 스토리지에 저장하고, 저장된 대량의 데이터를 빠르게 조회 및 분석할 수 있도록 분산 환경에서 병렬로 실행하여 빠르고 안정적으로 처리할 수 있는 시스템이다. 주요 처리 방식으로 대량의 정적인 파일을 한 번에 처리하는 분산 배치 처리 방식과 실시간으로 끊임없이 유입되는 스트리밍 데이터(SNS, IOT 센서 등)를 유실 없이 빠르게 처리하는 실시간 처리 방식, 그리고 실시간으로 유입되는 데이터에서 이벤트(현재 온도, 습도, 구매 정보 등)를 정의하고, 이벤트를 탐지하는 규칙(Rule)을 작성하여 규칙 기반의 이벤트를 탐지하는 이벤트 처리

방식이 있다.

1. 빅데이터 처리 유형

빅데이터 처리 유형은 처리할 데이터 유형에 따라서 크게 정적 배치 처리와 실시간 처리로 분류할 수 있고, 실시간 처리는 대량의 데이터 처리·분석을 위한 실시간 처리와 룰(Rule) 기반의 이벤트 처리 /탐지 방식으로 분류할 수 있다. 특히 실시간 데이터를 처리하면서, 동시에 이벤트 처리와 연동하여 정해진 룰에 맞는 이벤트를 탐지하도록 구성할 수 있다.



[그림 1-1] 빅데이터 처리 유형

2. 빅데이터 처리 유형의 특징 비교

<표 1-1> 빅데이터 처리 유형의 특징 비교

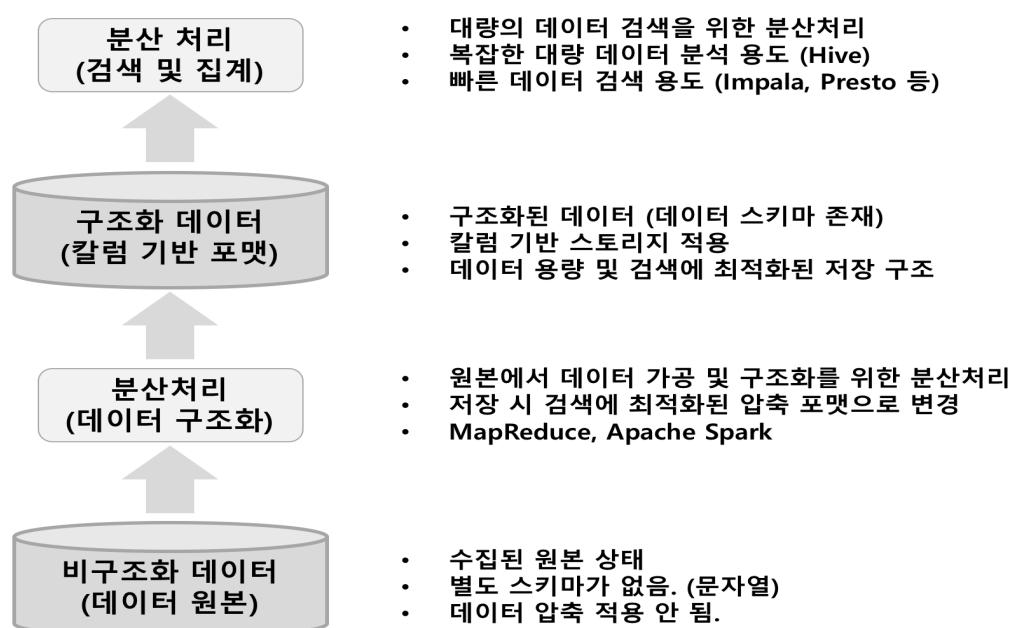
구분	배치 처리	실시간 처리	이벤트 처리
처리 주기	긴 주기 (1일, 일주일 등)	아주 짧은 주기 (밀리세컨드 단위)	이벤트 발생 시
데이터 유형	특정 기간 동안 저장된 정적인 데이터	끊김 없이 유입되는 동적인 데이터	특정 사건 발생 또는 상태 변경 시
데이터 예시	일주일간 기록된 웹 로그	SNS, 장비 센서 로그	온도 센서, 카드 결제 정보

구분	배치 처리	실시간 처리	이벤트 처리
데이터 특징	대량의 고정된 데이터	유입량의 변화가 짧은 주 기 동안 수집된 데이터	이벤트 유형에 따라 동적인 데이터
데이터 보관	분산 파일 시스템 기반의 안정적인 저장소	분산 메시지 큐에 임시 보관	를 엔진에서 처리 가능한 용량만 보관
장애시 재처리	전체 프로세스 처음부터 재시작	실시간 처리 로그 확인하여, 이후 데이터 처리	를 엔진 장애 시 기존 이벤트 정보 유실.
핵심 성능	대량 데이터 안정적 처리(처리량 최대화)	유실 없는 빠른 처리	이벤트 기반을 탐지 성능
핵심 기술	Tez, Apache Spark, Apache Hive, Presto	Apache Spark Streaming, Apache Storm, Apache Flink	Drools Fusion, Esper

③ 분산 배치 처리

저장된 대량의 데이터를 분산하여 처리하는 방식으로, 많은 자원(cpu, memory, disk, network)을 사용하여 안정적으로 결과를 생성한다. 보통 1개월에서 몇 년간 저장된 데이터에서 다양한 통계를 추출하거나, 시각화를 위한 BI에 연동하기 위한 데이터 마트를 생성하는 용도로 활용한다.

분산 배치 처리가 필요한 단계는 수집된 대량의 비구조화된 원본 데이터를 구조화하고 압축 저장 포맷으로 변환하는 단계와 이미 저장된 데이터에서 분산 처리를 통해 빠르게 대량의 데이터를 조회할 수 있도록 하는 단계로 나뉜다. 각 단계에 따라 최적의 기술이 존재하며, 요구 사항에 맞는 기술을 선택하여 적용한다.



[그림 1-2] 분산 배치 처리 단계

④ 컬럼 기반 스토리지(Column-oriented storage)

기존 관계형 데이터베이스는 행 기반으로 데이터를 저장 및 조회하도록 구성되었다. 이는 데이터를 행단위로 저장하여 데이터의 잣은 입력/수정에 최적화(OLTP, OnLine Transaction Processing)하도록 구성하였다.

행(row) 기반 스토리지

id	Name	Address	Age	Zip
1032123	Sm3ith	984 A ST	43	12345
2932336	Jane	1234 MAIN ST	25	56779
8900284	John	76 Queen ST	76	2321

[그림 1-3] 행 기반 스토리지 데이터 저장 예시

하지만 대부분의 빅데이터 처리는 한번 저장되면, 다양한 분석을 위한 읽기에 최적화되어 구성되어야 하므로, 기존 행 기반 스토리지 방식은 성능이 저하된다. 따라서, 다음 그림과 같이 칼럼(열) 기반으로 데이터를 저장하여 저장 공간을 줄이고, 선택된 칼럼만 조회할 수 있어 읽기 성능을 향상해 준다. 또 동일한 칼럼에 유사한 데이터가 많은 경우(Age와 같이 1에서 100이 거의 대부분인 경우) 압축률이 수십 배 이상 높은 효과를 가져온다.

열(column) 기반 스토리지

id	Name	Address	Age	Zip
1032123	Sm3ith	984 A ST	43	12345
2932336	Jane	1234 MAIN ST	25	56779
8900284	John	76 Queen ST	76	2321

[그림 1-4] 칼럼 기반 스토리지 데이터 저장 예시

⑤ 실시간(스트림) 처리

실시간(스트림) 데이터는 배치와 달리 끊임없이 유입되는 데이터를 뜯하며, 실시간(스트림) 처리는 프로그램의 종료 없이 항상 동작하면서 유입되는 데이터를 처리하는 방식이다. 실시간 데이터의 특성상 유입되는 데이터가 균일하지 않으며, 특정 시점에 급증하는 현상이 있으므로, 실시간(스트림) 처리는 이를 안정적으로 분산 처리할 수 있도록 구성해야 한다.

1. 실시간 처리 핵심 기술 요소

<표 1-2> 실시간 처리 핵심 기술 요소

기술 요소	설명	대표 기술
메시지 큐 (Message Queue)	데이터를 수집하여 전송하는 모듈과 이를 처리하는 모듈 간의 종속성을 제거하여 데이터가 유실 없이 안정적으로 처리할 수 있도록 하는 기술	
	즉, 수집한 데이터를 메시지 큐로 전송하고, 메시지 큐는 이를 안정적으로 분산하여 보관하고, 이를 처리하는 시스템이 처리할 수 있는 용량만큼 처리할 수 있도록 한다. 이를 통해 실시간으로 급증하는 메시지의 유입을 안정적으로 처리 가능	Apache Kafka

기술 요소	설명	대표 기술
실시간 분산 처리	실시간으로 유입되는 대량의 데이터를 빠르고 안정적으로 처리할 수 있도록 구성된 분산 처리 시스템	Apache Spark Apache Storm Apache Flink
In-Memory Cache	빠르게 유입되는 데이터에서 관련 정보를 추출하기 위한 용도로 사용하는 메모리 캐시 사용자 아이디나 머신 아이디를 키로 하여 세부 정보를 추출하는 용도로 활용	Redis
메모리 기반 저장소	실시간으로 처리한 대량의 데이터를 빠르게 저장하고, 검색할 수 있는 메모리 기반의 저장소 메모리 용량에 맞추어 일정 기간(몇 시간 또는 며칠)의 데이터만 저장 및 검색 원본 데이터는 분산 파일 시스템에 저장하여 유실 방지	MemSQL, VoltDB
시계열 저장소	실시간 데이터의 특성상 시간 순서대로 생성되는 데이터를 처리하며, 이러한 시계열 데이터에 최적화된 데이터 구조를 제공하여 데이터 저장 및 검색	Elasticsearch InfluxDB

2. 활용 및 기대 효과

<표 1-3> 실시간 처리 기술의 활용 및 기대 효과

구분	설명
활용 방식	기존 단일 프로그램 방식으로 처리 불가능한 대량의 실시간 메시지를 처리하는 경우 대량의 실시간 메시지를 분산하여 처리 가능 (공장의 센서 정보, SNS 메시지 등)
기대 효과	데이터 저장 전에 실시간으로 데이터를 처리 및 분석하여 비즈니스에서 의미 있는 결과를 추출하는 용도로 활용 (최근 5분 동안 많이 구매된 상품 목록 Top 10 등) 룰 엔진과 연동하여 비정상 거래 및 행위를 실시간으로 탐지하는 용도 (카드 거래 이력이 1분 동안 5건 이상 거래 등)

⑥ 이벤트(Event) 처리

1. 이벤트의 정의

이벤트는 특정 시스템 내에서 발생한 어떤 액션을 말하며, 특정 상태에서 다른 상태로의 변화로도 정의할 수 있다. 예를 들어 문자 메시지, SNS 포스트, 주식 시장의 주가 정보, 교통 정보, 기상 정보 등의 데이터도 이벤트가 될 수 있다. 또 측정값이 사전의 정의된 시간, 온도 등의 기타 값의 임계값을 넘었을 때 ‘상태의 변화’를 이벤트로 정의할 수 있다.

(1) 이벤트 타입

이벤트 타입은 개념적으로 정의된 이벤트를 시스템에서 처리할 수 있도록 프로그래밍 방식으로 표현하는 것을 말한다. 모든 이벤트는 자체 속성을 가지고 있는데, 이를 스키마 형식으로 표현한 것을 이벤트 타입이라고 한다. 이러한 이벤트 타입을 정의하는 포맷으로는 POJO(Plain Old Java Object), Json, XML, CSV 등이 있다.

```
package org.myapp.event;
public class OrderEvent {
    private String itemName;
    private double price;

    public OrderEvent(String itemName, double price) {
        this.itemName = itemName;
        this.price = price;
    }
    public String getItemName() {
        return itemName;
    }
    public double getPrice() {
        return price;
    }
}
```

[그림 1-5] POJO 포맷의 이벤트 타입 예시 (주문 이벤트)

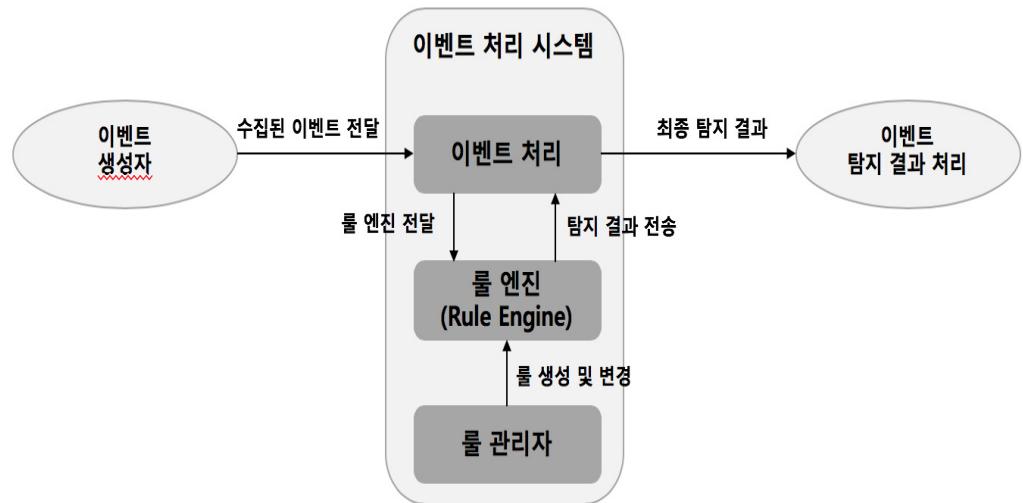
2. 이벤트 처리 시스템

이벤트 처리는 순차적으로 발생하는 이벤트의 흐름을 추적하고, 이로부터 결과를 도출하는 시스템이다. 복합 이벤트 처리(Complex Event Processing, CEP)는 좀 더 복잡한 상황의 이벤트에서 패턴을 추론하기 위하여 복수의 이벤트를 기반으로 패턴을 탐지하는 이벤트 처리 방법이다. CEP의 목표는 복합적인 이벤트에서 의미 있는 정보(고장 감지, 불법 접근 등)를 도출하고, 이에 대해 빠르게 대응한다.

예를 들어, 리테일 체인점이 판매를 늘리기 위한 방법으로 일 년에 1,000달러 이상 소비한 중요 고객들이 지속적으로 재방문하도록 유도하기 위해 CEP 기술을 사용하여 높은 가치의 고객이 프로그램에 참여하자마자 시스템이 그 고객에게 추가 할인 혜택을 제공하도록 할 수 있다. 또 신용 카드 회사들은 사기를 방지하기 위해 CEP를 활용하여, 사기 범죄의 유형이 발견될 때, 회사의 중대한 손실을 겪기 전에 신용 카드를 차단할 수 있다.

(1) 이벤트 처리 시스템 구성도

외부에서 수집된 데이터(센서, SNS, 카드 결제 정보 등)를 이벤트 타입으로 변환하여 이벤트 처리 시스템으로 전달하면 이를 룰 엔진에서 정의된 규칙에 따라서 새로운 이벤트를 탐지하게 된다. 여러 개의 이벤트가 복합적으로 연관되어 있다면 최종 이벤트 탐지 결과를 출력하게 된다.



[그림 1-6] 이벤트 처리 시스템 구성도

(2) 이벤트 처리 시스템 구성 요소

<표 1-4> 이벤트 처리 기술의 구성 요소

구성 요소	설명	대표 기술
이벤트 생성자	외부에서 생성되는 데이터를 수집하여 이벤트 타입으로 변환하고, 이벤트 처리 시스템으로 전달 제품에 따라 별도의 이벤트 처리 시스템 없이 이벤트 생성자에서 직접 이벤트 처리 가능	Esper(라이브러리 제공 방식)
이벤트 처리	다양한 이벤트를 수신하여 룰 엔진에서 정의한 규칙에 해당하는지 결과 확인 다수의 이벤트 생성자에게서 받은 이벤트를 비즈니스 요건에 따라 처리 및 룰 엔진 연동	Drools Kie Execution Server
룰 엔진	다양한 룰을 내부 메모리에서 관리하며, 특정 기간(최근 5분 또는 최근 3건) 동안 입력된 이벤트와 현재 입력된 이벤트 간의 관계 및 룰 기반의 탐지 수행	Esper Drools Fusion
룰 관리자	사용자가 웹 UI 기반으로 룰을 직접 정의하거나, 동적으로 변경 및 삭제	Drools Workbench

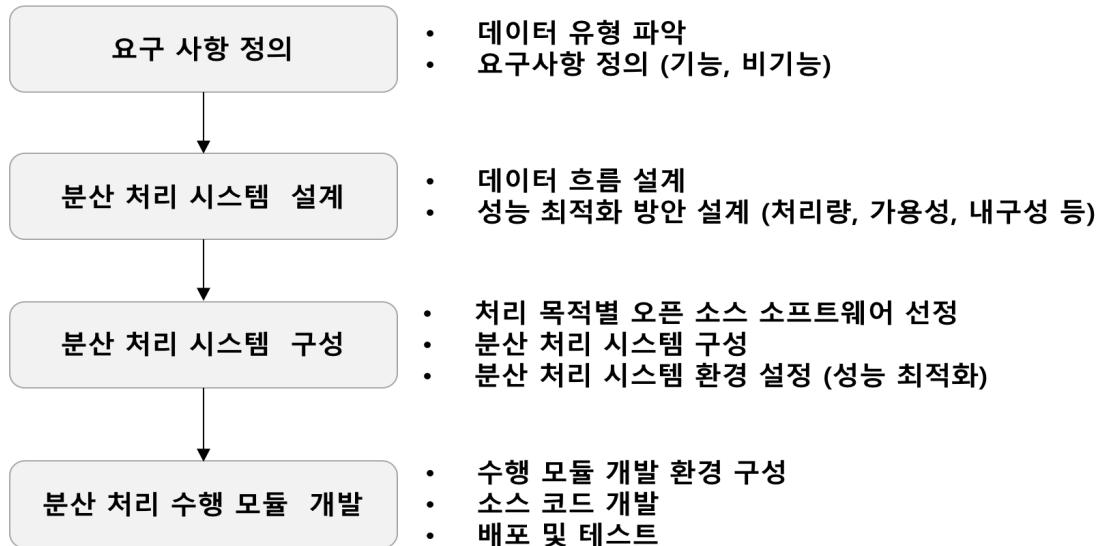
⑦ 아파치 하둡 에코 시스템(Apache Hadoop Ecosystem)

다수의 컴퓨터에서 대량의 데이터를 분산 저장하고, 이를 병렬로 처리할 수 있도록 하는 기술인 아파치 하둡(Apache Hadoop)을 기반으로 쿼리 엔진, 실시간 처리, 머신러닝 등의 기능을 수행하는 오픈 소스들의 집합이다.

⑧ 빅데이터 분산 처리 시스템 구축 절차 이해

빅데이터 처리 시스템을 효율적으로 구축하기 위해 소프트웨어 개발 방법론의 기본 절차를 따르면서, 빅데이터 분산 처리에 필요한 특징을 요구 사항 및 설계에 반영하여야 한다. 또 설계된 구성을 기반으로 다양한 소프트웨어를 활용하여 최적의 시스템을 구성하고, 이를 기반으로 실제 처리 모듈을 개발하여 배포하는 절차를 이해하는 것이 중요하다.

1. 구축 절차



[그림 1-7] 빅데이터 분산 처리 시스템 구축 절차

2. 단계별 주요 활동

<표 1-5> 단계별 주요 활동

단계	주요 활동	산출물
요구 사항 정의	데이터 유형을 파악하여, 설계할 시스템의 특징을 확인한다. 데이터 정의서	
분산 처리 시스템 설계	사용자 요구 사항을 기반으로 필요한 기능을 도출하고, 분산 처리 시스템의 성능(처리량, 가용성, 내구성 등) 요구 사항 정의서 건을 정의한다.	데이터 흐름도
분산 처리 시스템 구성	사용자의 기능 요구 사항을 만족하는 데이터 처리 흐름을 설계한다. 각 단계별로 데이터 관점에서 입력/ 처리 /출력의 흐름을 명확하게 표현한다.	기능 명세서
	설계된 기능을 구현하는데 최적의 소프트웨어를 선정하고, 이를 기반으로 분산 처리 시스템을 설치한다. 성능 요구 사항에 만족하기 위한 소프트웨어 환경 설정	시스템 구성도

단계	주요 활동	산출물
	을 조정하고, 이를 시스템에 반영한다.	
	최종 설치된 시스템 구성이 정상 동작하는지 테스트한다.	
	선정된 오픈 소스 소프트웨어를 기준으로 요구 사항을 개발된 수행 모듈 만족하는 프로그램을 개발한다.	수행 모듈
분산 처리 수행 모듈 개발	처리할 데이터 특성에 따라 배치 분산 처리 수행 모듈, 실시간 처리 수행 모듈, 이벤트 처리 수행 모듈을 활용하여 개발한다.	
	최종 개발된 수행 모듈을 배포하고, 테스트한다.	

수행 내용 / 빅데이터 처리 시스템 설계하기

재료 · 자료

- 처리할 데이터의 유형, 주기 등이 작성된 명세서
- 비즈니스 요건이 정의된 산출물(인터뷰 결과서 등)
- 기존 운영 시스템 또는 신규 시스템의 아키텍처 산출물

기기(장비 · 공구)

- 컴퓨터, 프린터, 인터넷
- 사무 자동화 소프트웨어

안전 · 유의 사항

- 빅데이터 처리 시스템을 설계할 때 오픈 소스 기술에 집중하는 것이 아닌, 최대한 단순한 구조로 요구 사항을 만족할 수 있도록 유의하여 설계한다.
- 선택한 오픈 소스 소프트웨어가 빅데이터 분산 처리의 핵심 요건인 분산 처리, 가용성 제공, 데이터 유실 방지(비즈니스 요건에 따라 유실 허용 가능)를 지원할 수 있는지 유의하여 검토해야 한다.
- 일부 서버의 장애가 전체 빅데이터 처리 시스템의 장애로 전파되지 않도록 서버(메모리, CPU 등) 자원의 여유분을 고려하여 시스템을 설계한다.
- 빅데이터 처리 시스템의 일부 서버에서 발생할 수 있는 장애 상황을 고려하여, 다른 서버에 과도한 부하가 전파되지 않으면서, 시스템의 가용성이 확보될 수 있도록 시스템의 서버 규모를 산정한다.

수행 순서

① 처리할 데이터 유형 및 특징을 파악한다.

데이터 유형에 따라서 최적의 소프트웨어를 선택하고, 데이터 처리를 위한 흐름을 설계할 수 있도록 특징을 정리한다.

1. 데이터 수집 유형을 확인한다.

처리할 데이터가 특정 주기(1일, 1개월 등)별로 한 번에 대량의 처리가 필요한 데이터인

지, 실시간으로 처리해야 할 데이터인지 확인한다.

(1) 배치 데이터 특징을 확인한다.

기업이 오랜 기간 동안 저장해 온 대용량의 데이터 파일이며, 많은 파일로 분할되어 저장되어 있고 각 파일의 크기가 큰 경우가 많다.

수행 tip

- 너무 작은 크기(1MB 이하)의 파일이 많으면 분산 처리의 성능이 저하된다.
- 작은 파일들은 병합하여 하나의 큰 파일로 생성하는 것이 분산 처리에 효과적이다.

(2) 실시간 데이터 특징을 확인한다.

소셜 미디어(facebook 등), IoT 센서 장비, 통신 장비 등 실시간으로 생성 및 유입되는 데이터는 배치 방식의 수집이 불가능하다. 따라서 지속적으로 네트워크를 통해서 수집되며, 데이터의 유입량이 일정하지 않고 불규칙하게 변화한다.

수행 tip

- 실시간 데이터의 안정적인 전송이 가장 중요하며, 이에 가장 큰 영향을 주는 것이 네트워크 대역폭이다.
- 실시간 데이터 특성상 처리 중 실패에 대한 재처리가 어려우므로, 메시지 큐(Apache Kafka 등)를 활용하여 유실을 방지한다.

2. 데이터 정의서를 작성한다.

처리할 데이터의 유형과 크기, 처리 주기 등을 세부적으로 확인하고, 각 데이터 유형별로 목록으로 기록한다.

(1) 데이터 정의서 항목별 작성 내용을 확인한다.

처리할 데이터에 대하여 항목별로 구체적인 내용을 확인하고, 데이터 처리에 필요한 정보를 정리한다.

<표 1-6> 항목별 작성 내용

항목	작성값	확인할 내용	예시
유형	배치	데이터의 처리 주기를 확인한다.	배치
	실시간	실시간으로 유입되는 데이터인지 확인한다.	실시간
위치	데이터 위치	배치 유형인 경우 실제 파일의 위치 또는 데이터가 저장된 데이터베이스 경로	
		파일이 일 단위, 시간 단위, 분 단위로 생성되는지 확인하고, 파일명 생성 규칙도 정의한다.	/data/temp/test_yyy y_mm_dd_ss.csv
		실시간 유형인 경우 유입되는 경로	http://open-api/

항목	작성값	확인할 내용	예시
		Open API(SNS, Web Service 등), 메시지 큐 서버(Apache Kafka 등)	
처리 주기	처리 시간	배치 유형인 경우 데이터를 처리할 시간을 확인한다.	매일 02:00
		실시간 유형의 경우 최소 대기시간을 지정 한다. 즉, 정해진 초 동안 유입되는 데이터 를 한 번에 처리한다.	
사이즈	데이터 크기	배치유형의 경우 전체 대상 데이터의 크기 를 확인한다. 파일이 여러 개로 분할되어 있는 경우, 개 별 파일의 평균 크기를 계산한다.	ROW SIZE: 10 KB FILE SIZE: 100 MB
		실시간 유형의 경우 일반적으로 초당 유입 량을 확인한다. 초당 유입량을 확인하고, 처리 주기에 어느 20MB 정도의 데이터가 유입되는지 평균값으로 계산할 수 있다.	
데이터 건수	처리할 데이터	배치 유형인 경우 전체 데이터 건수와 파일 별 평균 데이터 건수로 구분하여 확인한다.	100,000,000건
		실시간 유형의 데이터가 주로 유입되는 구 간의 전체 데이터 건수를 초 기준으로 나 누어 평균적인 유입 건수를 확인한다.	10,000건 / 초

수행 tip

- 실시간 데이터의 평균 유입량은 대략적인 처리량을 예측하는 데 사용되고, 실제 시스템에 영향을 미치는 요소는 초당 최대 유입량이다.
- 365일 중 특정 일자나 시간에 최대 몇 건의 유입량이 발생하는지 확인해야 아키텍처를 설계할 때에 서버 확장성, CUP, MEMORY, NETWORK 대역폭 등 을 결정하는 데 도움이 된다.

(2) 데이터 정의서를 작성한다.

처리할 데이터의 구체적인 사항을 사전에 파악하고, 이를 참조하여 기능 · 비기능 요구 사항 및 제약 조건을 정의한다.

<표 1-7> 데이터 정의서(예시)

데이터 목록	유형	위치	처리 주기	처리 시간	사이즈	건수
상품 거래 로그	배치 파일	폴더명	1일 1회	01:00	500kB	1,000,000
Web Server log	실시간	API	실시간	실시간	1kB	10,000/초

수행 tip

- 데이터 정의서에 정의된 내용은 빅데이터 처리의 기반이 되는 데이터이므로, 각 담당자와의 인터뷰 및 실제 데이터를 직접 확인하여 작성하도록 한다.
- 프로젝트 요구 사항에 따라 데이터의 다양한 특징을 추가하여 관리한다.(파일 포맷 등)

② 데이터 유형에 따른 요구 사항을 정의한다.

데이터 정의서에 명시된 데이터 유형 및 특징을 처리하기 위해 필요한 기능적 요구 사항과, 성능 및 안정성의 수준을 정의하기 위한 비기능적 요구 사항을 정의한다.

1. 요구 사항 목록을 확인한다.

빅데이터 플랫폼 요구 사항 분석(2001020901_17v1)에서 정의된 요구 사항 목록에 업무에 필요한 모든 기능 및 비기능 항목을 확인한다.

<표 1-8> 요구 사항 목록(예시)

ID	기능	품질 속성	설명	중요도	...
R-F-001	센서 로그 분산 저장	기능	일 단위 분산 처리로 수집된 센서 로그를 분산 파일 시스템에 저장한다.	상	
R-F-002	원본 데이터 구조화	기능	비정형 원본 데이터를 구조화하고, 컬럼 스토리지 형식의 포맷으로 변환하여 저장한다.	중	
...
R-P-001	처리 시간	비기능	하루 100만 건의 로그 정보를 1시간 내에 처리해야 한다.	상	

수행 tip

- 비기능 요구 사항은 테스트가 가능한 수준으로 명확한 기준을 정의해야, 이를 기반으로 완성된 시스템을 평가할 수 있다.

2. 기능 요구 사항을 정의한다.

분산 처리를 위해 필요한 기능 요구 사항을 항목별로 구분하여 구체적으로 정의한다. 모든 내용은 관련된 이해관계자가 이해하기 쉽도록 작성되어야 하며, 전체 시스템의 기능을 포괄적으로 포함되도록 기능을 정의해야 한다.

또 데이터 유형에 따라 빅데이터 분산 처리에 필요한 적합한 기능적 요구 사항이 정의되었는지 확인한다. 예를 들어 이벤트 처리에 필요한 기능 요구 사항에 복합 룰(rule) 정의 기능을 포함하여 정의한다.

3. 비기능 요구 사항을 정의한다.

빅데이터 분산 처리의 품질(성능, 보안, 안정성 등)을 만족할 수 있는 비기능적 요구 사항을 정의한다. 빅데이터 분산 처리 시스템은 대량의 데이터를 처리하는 데 최적화된 시스템으로 사용 목적에 따라서 성능과 안정성, 보안 요건을 다르게 적용해야 한다. 빅데이터를 안정적으로 최적의 성능으로 분산 처리하기 위해 고려해야 할 요소들이 있다. 주로 처리량(Throughput) 향상, 지연(Latency) 최소화, 가용성(Availability) 보장, 내구성(Durability) 보장과 같은 비기능 요건을 처리 유형에 따라 조정해야 한다.

<표 1-9> 처리 유형별 비기능 요구 사항 예시

비기능 요건	분산 배치 처리	실시간 처리	이벤트 처리
처리량	높음. (대량의 데이터 일시에 처리)	중간 (짧은 주기의 데이터 처리)	중간 (이벤트가 발생할 때 처리)
데이터 지연	높음. (분산 처리에 많은 시간 소요)	낮음 (지연 없이 빠른 처리)	낮음. (이벤트 발생 시점에 처리)
가용성	낮음. (장애가 발생할 때 재 시작)	높음. (서버 간 데이터 복제로 장애 데이터 처리)	낮음. (룰 엔진이 하나의 서버에서만 동작)
내구성	데이터 유실 없음. (기존 적재된 데이터 처리)	유실 가능 (메시지 큐를 통해 유실 방지)	유실 가능 (룰 엔진이 장애가 발생할 때 이벤트 유실 가능)

수행 tip

- 대량의 데이터를 빠르게 처리하는 목적을 위해서는 메모리 기반의 처리 기술을 이용하여 중간 단계 데이터 유실을 감수하고, 처리 성능을 극대화할 수 있다.
- 네트워크를 통한 외부 접속이 가능한 환경에서는 처리 시스템의 실행에 접근 권한을 부여하여 사용자별 처리를 제한하도록 설계할 수 있다.

③ 요구 사항에 따른 빅데이터 처리 모델을 설계한다.

명시된 기능 및 비기능 요구 사항을 만족하는 빅데이터 처리 방식을 선택하고, 이를 구현하기 위한 처리 흐름을 설계한다.

1. 분산 처리(배치) 시스템을 설계한다.

저장된 대량의 데이터를 빠르고 안정적으로 처리할 수 있는 시스템을 설계하기 위해 분산 처리 유형에 따라 데이터 흐름을 설계한다.

(1) 분산 처리 데이터 처리 흐름을 설계한다.

분산 처리를 실행하기 위한 처리 흐름을 기능 요구 사항을 만족할 수 있도록 구체적으로 도식화하여 설계한다. 처리할 대상 데이터의 입력에서 가공 및 저장 단계를 순차적인 흐름으로 작성하여 이를 기반으로 시스템을 구현할 수 있도록 작성한다.

정해진 포맷이 정해져 있는 것은 아니며, 기능 요구 사항 및 비기능 요구 사항을 만족할 수 있도록 데이터 흐름을 세부 기능의 흐름으로 표현한다.



[그림 1-8] 분산 처리 데이터 흐름도 예시

(2) 단계별 처리 기능을 상세히 작성한다.

위의 데이터 처리 흐름도에서 도출된 기능별로 실제 개발이 가능한 수준으로 구체화하여 명세서로 작성한다. 주요 작성 항목으로는 각 단계에서 입력/처리/출력 내용을 세부적으로 명시하여 프로그램 개발에 필요한 자료구조 및 함수를 설계할 수 있도록 한다. 쿼리 엔진(sql-on-hadoop)을 이용하는 경우는 SQL 구문의 실행 순서를 결정할 수 있도록 내용을 구성한다.

<표 1-10> 단계별 기능 명세 예시

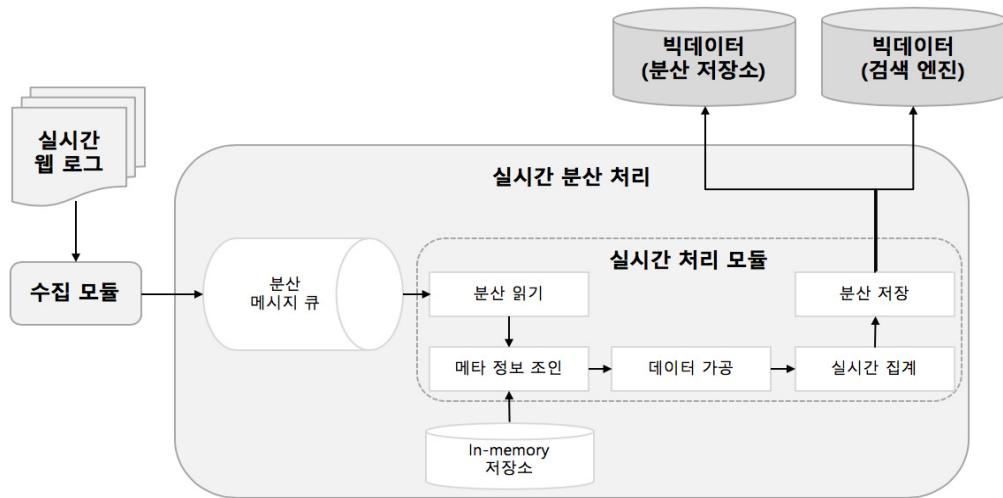
기능	데이터 입력	데이터 가공	데이터 출력
분산 읽기	웹 로그(csv 파일) 칼럼 정보(사용자 ID, 요청 URL, 응답 코드 등)	데이터 파싱(입력 포맷 준수) 데이터 필터링(NULL 확인 등)	가공된 데이터를 다음 단계 전달 규칙에 맞게 파싱된 데이터만 전달됨.
분산 가공	읽기에서 파싱된 데이터	데이터 포맷 검증(문자열, 숫자 등) 데이터 정합성 검증(자리 수, 카테고리 데이터 등) 데이터 구조화(스키마 적용)	구조화된 데이터 전달
...			
분산 쓰기	구조화된 데이터	스키마 생성 데이터 저장 포맷 적용	분산 스토리지 저장

2. 실시간 처리 시스템을 설계한다.

지속적으로 유입되는 데이터를 유실 없이, 안정적으로 처리하기 위해 필요한 기능 및 비기능 요구 사항을 확인하고, 이를 지원할 수 있는 시스템을 설계한다.

(1) 실시간 데이터 처리 흐름을 설계한다.

실시간 데이터 처리를 실행하기 위한 처리 흐름을 기능 요구 사항을 만족할 수 있도록 구체적으로 도식화하여 설계한다. 처리할 대상 데이터의 입력에서 가공 및 저장단계를 순차적인 흐름으로 작성하여 이를 기반으로 시스템을 구현할 수 있도록 작성한다. 실시간 처리는 급증하는 메시지 유입량을 고려하여, 처리 용량 이상의 메시지를 보관할 수 있는 분산 메시지 큐를 반드시 적용해야 한다. 또 실시간 처리량 및 지연을 최소화하기 위하여 메모리 기반의 데이터 저장소 및 캐시를 적극적으로 활용하여 처리 성능을 향상할 수 있다.



[그림 1-9] 실시간 처리 데이터 흐름도 예시

(2) 단계별 처리 기능을 상세히 작성한다.

위의 데이터 처리 흐름도에서 도출된 기능별로 실제 개발이 가능한 수준으로 구체화하여 명세서로 작성한다. 주요 작성 항목으로는 메시지 큐에 저장할 메시지의 크기 및 보관 용량을 명시하고, 인메모리(in-memory)에 저장하여 조인할 데이터의 세부 칼럼 및 조인할 키 정보를 작성한다. 또 실시간 데이터를 처리할 주기를 정의하여 메시지 지연을 최소화하거나 초당 처리 용량을 최대화할 수 있다.

<표 1-11> 단계별 기능 명세 예시

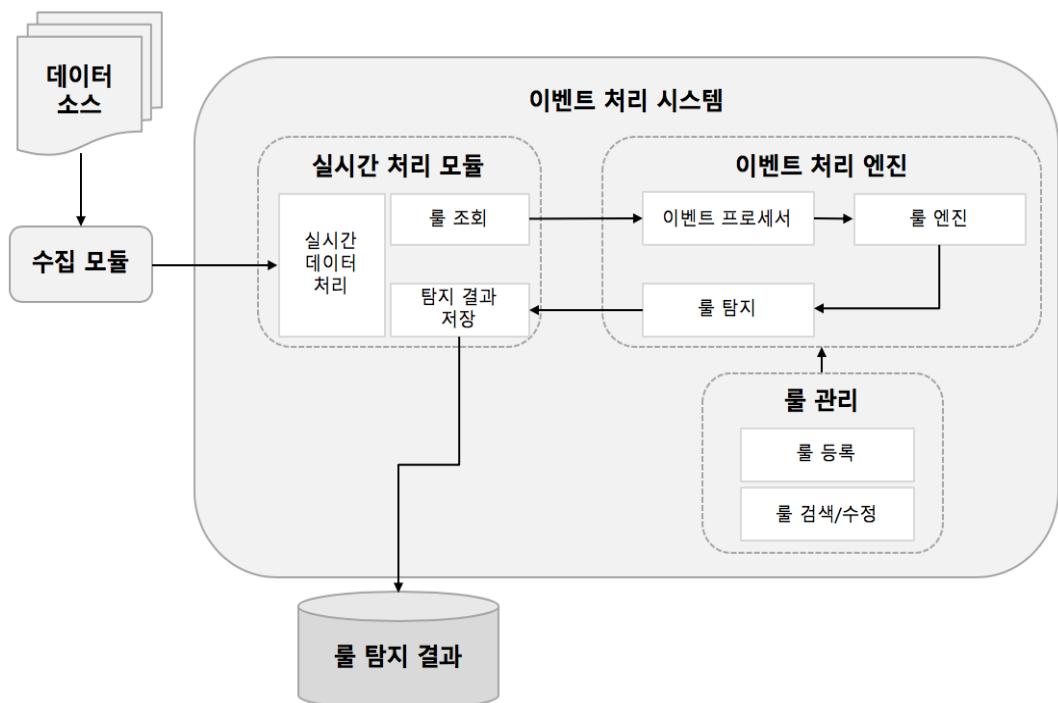
기능	데이터 입력	데이터 가공	데이터 출력
분산 메시지 큐	수집된 데이터	유실 없이 데이터 분산 보관 데이터 쓰기/읽기 성능 보장	빠른 읽기 성능 제공
인 메 모 리 (in-memory) 저장소	실시간 데이터와 결합할 정보 (예를 들면 사용자 ID를 키로 저장 된 정보)	없음. 빠른 조회를 지원하는 저장소 (사용자 ID를 KEY로 조회)	저장된 데이터 조회
메타 정보 조인	메시지 큐 데이터 인메모리(in-memory) 데이터	사용자 ID를 기준으로 두 개의 입력 데이터를 조인하여 정보를 결합	사용자 ID로 결합된 데이터
...			
실시간 집계	실시간 수집된 데이터	최근 1분/5분 등을 기준으로 특정 칼럼값을 집계 실시간 분산 저장소에 저장 가능 한 데이터에 한하여 집계 가능	집계된 요약 정보

3. 이벤트 처리 시스템을 설계한다.

실시간 대용량 데이터를 처리하는 관점에서 실시간으로 유입되는 데이터에서 패턴을 찾고, 탐지할 수 있도록 다양한 유형의 이벤트에서 특정 패턴을 발견할 수 있는 규칙(Rule)을 정의하고, 이를 기반으로 실시간 유입되는 데이터에서 가치를 찾거나, 위험 요인을 사전에 감지하여 대응할 수 있도록 시스템을 설계한다.

(1) 이벤트 데이터 처리 흐름을 설계한다.

예시로 방의 온도 센서로부터 온도 이벤트(데이터)를 실시간으로 전달받고, 최근 1분 동안 평균 온도(40도)가 넘고, 습도가 60% 이상이면 자동으로 온도를 낮추는 데이터 흐름을 설계해 보자.



[그림 1-10] 이벤트 처리 데이터 흐름도 예시

(2) 단계별 처리 기능을 상세히 작성한다.

위의 데이터 처리 흐름도에서 도출된 기능별로 실제 개발이 가능한 수준으로 구체화하여 명세서로 작성한다. 주요 작성 항목으로는 생성 및 처리할 이벤트와 각 이벤트에 포함될 속성(예를 들어 온도 이벤트는 현재 온도, 거래 이벤트는 거래금액, 거래지점 등)을 정의하고, 의미 있는 정보를 찾거나, 비정상 행위를 탐지할 수 있는 룰을 구체적으로 정의한다.

<표 1-12> 이벤트 처리 단계별 기능 명세 예시

모듈	기능	상세 명세
실시간 처리 모듈	실시간 데이터 처리 를 실행	실시간 유입되는 데이터에서 이벤트를 데이터를 추출한다. 룰에 따라 정의된 속성을 가지도록 이벤트를 생성한다.
	룰 탐지 결과 저장	생성된 이벤트를 이벤트 처리 엔진으로 전달하여, 룰에서 이벤트를 감지할 수 있도록 한다.
이벤트 처리 모듈	이벤트 프로세서 를 엔진	이벤트 처리 엔진에서 특정 룰에 탐지된 이벤트가 있으면, 이를 수신한다. 수신된 이벤트 탐지 결과를 처리하기 위해 저장소 또는 이벤트 처리 시스템으로 전달한다.
	이벤트 탐지	수신 받은 이벤트를 룰 엔진에서 분석할 수 있는 형식으로 처리한다.
룰 관리 모듈	룰 등록	다수의 룰을 관리하며, 이벤트 추출, 이벤트 패턴 매칭, 이벤트 상관관계 분석, 이벤트, 복합 이벤트 구성 등을 수행한다. 룰에서 탐지된 결과 이벤트가 다른 룰의 입력으로 연결되는 복합 이벤트 처리를 지원한다. 정의된 룰에 탐지된 이벤트 결과는 이벤트 탐지로 전달한다.
	룰 검색/수정	탐지된 이벤트 정보를 이벤트를 호출한 시스템으로 전달해 준다.
		이벤트를 탐지할 룰을 등록한다. 제품마다 룰을 정의하는 방식이 다르므로, 해당 제품의 룰 규격에 맞게 룰을 정의한다.
		등록된 룰의 탐지 로직이 변경된 경우, 이를 동적으로 수정할 수 있도록 지원한다.

수행 tip

- 이벤트 처리 모듈은 별도로 개발하지 않고, 오픈 소스 및 상용 제품이 내부적으로 지원하는 기능이므로, 실제 구현 범위에서는 제외한다.
- 실시간 처리량이 크지 않은 경우에는 실시간 처리 모듈을 웹 서버 방식으로 구성하여 이벤트를 처리할 수도 있다.

④ 빅데이터 분산 처리 시스템의 테스트 계획을 작성한다.

데이터 유형 및 업무 목적별로 구성된 빅데이터 분산 처리 시스템이 목적한 방식대로 동작하는지 확인하기 위한 테스트 계획을 수립한다. 빅데이터 플랫폼 테스트(2001020908_17v1)를 참고하여 빅데이터 플랫폼 테스트 항목을 정의한다.

분산 처리(배치)의 핵심 기능 및 비기능 요구 사항에 명시된 항목을 중심으로 테스트 항목, 테스트 내용, 검증 방식을 기술하기 위한 테스트 계획서를 작성한다.

<표 1-13> 테스트 계획서 예시

테스트 항목	유형	테스트 내용	검증 방식
일 단위 웹서버 로그 배치 처리 기능	기능	/data/logs/ 디렉터리에 저장된 데이터를 분산 처리하여, 결과 데이터를 Hadoop HDFS의 정해진 디렉터리에 처리된 결과값이 정상적으로 저장되는지 확인한다.	HDFS의 정상적으로 저 장 여부 확인
초당 처리량	비기능	700 GB의 데이터를 분산 처리하여, 구조화 데이터로 변환하는 초당 처리량을 측정한다.	초당 100,000건 이상의 성능을 유지하는지 확인
데이터 유실 여부	비기능	데이터 분산 처리 과정에서 비정상으로 종료할 경우, 재처리 로직에서 데이터 유실이 발생하는지 확인한다.	입력된 건수와 최종 결과 건수가 동일한지 확인
서비스 가용성	비기능	분산된 서버 중 일부 서버에장애가 발생한 경우, 전체 서비스에 영향이 없는지 확인한다.	특정 서버의 강제 종료 후, 서비스 정상 동작 여부 확인

학습 1 교수 · 학습 방법

교수 방법

- 빅데이터 분산 처리에 필요한 기술(오픈 소스 중심으로)을 분산 처리, 실시간 처리 영역으로 구분하여 핵심 특징을 사례를 통해 전달한다.
- 빅데이터 처리 방식을 설계하기 위해 필요한 데이터 유형(배치, 실시간) 및 유형별 특징을 확인하는 방법을 전달한 후, 사례를 조별로 조사하여 토의한 후 발표하는 토의식 프로젝트 수업을 실시한다.
- 데이터 유형별로 요구 사항(기능, 비기능)을 식별하고 빅데이터 처리 시스템을 설계할 수 있도록 특정 사례(온라인 쇼핑몰 결제 정보 처리, SNS 이력 정보 처리 등)를 중심으로 설계 절차 및 산출물 작성 방식을 지도한다.
- 설계된 빅데이터 분산 처리 시스템의 기능 및 성능을 명확하게 테스트할 수 있도록 테스트 내용과 검증 방식을 사례를 통하여 전달한다.

학습 방법

- 빅데이터 분산 처리에 필요한 기술 영역별 핵심 개념을 이해하고, 주요 내용은 기술 구성도 및 핵심 기능과 함께 노트에 정리한다.
- 조별로 빅데이터 처리 방식을 설계하기 위해 필요한 데이터 유형(배치, 실시간) 및 유형별 특징을 확인하는 방법과 사례를 조사하여 서로 토의한 후 발표를 통해 내용을 공유하고 숙지하도록 한다.
- 조별로 가상의 요구 사항(일별 판매량 Top 10 통계 생성, SNS 시간별 Top 10 키워드 추출 등)을 선정하고, 요구 사항을 만족할 수 있는 시스템을 설계하고, 결과를 공유 및 토의를 통해 지식을 체득한다.
- 조별 또는 개인이 설계한 빅데이터 분산 처리 시스템의 기능 및 성능을 누구나 검증할 수 있도록 테스트 방법 및 검증 방식을 명확하게 이해한다.

학습 1 평 가

평가 준거

- 평가자는 학습자가 학습 목표를 성공적으로 달성하였는지를 평가해야 한다.
- 평가자는 다음 사항을 평가해야 한다.

학습 내용	학습 목표	성취수준		
		상	중	하
빅데이터 처리 시스템 설계	- 빅데이터 관련 기술, 제품, 분석을 고려하여 처리 데이터의 크기, 종류, 저장구조에 따른 빅데이터 처리 방식을 결정할 수 있다.			
	- 데이터 처리 모델을 설계하기 위하여 데이터 요구 사항을 도출하고 데이터 유형별로 분류하여 특성을 파악할 수 있다.			
	- 데이터의 크기, 종류, 저장 구조 특성에 부합하는 데이터 처리 모델 및 제약 조건을 설계할 수 있다.			
	- 완료된 데이터 처리 모델 설계에 대한 결과의 타당성을 검증할 수 있다.			

평가 방법

- 서술형 시험

학습 내용	평가 항목	성취수준		
		상	중	하
빅데이터 처리 시스템 설계	- 빅데이터 관련 기술, 제품, 분석을 고려하여 처리 데이터의 크기, 종류, 저장 구조에 따른 빅데이터 처리 방식을 선택할 수 있는 능력			
	- 데이터 처리 모델을 설계하기 위하여 데이터 요구 사항을 도출하고 데이터 유형별로 분류하여 특성을 파악할 수 있는 능력			
	- 데이터의 크기, 종류, 저장 구조 특성에 부합하는 데이터 처리 모델 및 제약 조건을 설계할 수 있는 능력			
	- 완료된 데이터 처리 모델 설계에 대한 결과의 타당성을 검증할 수 있는 능력			

• 평가자 체크리스트

학습 내용	평가 항목	성취수준		
		상	중	하
빅데이터 처리 시스템 설계	- 빅데이터 관련 기술, 제품, 분석을 고려하여 처리 데이터의 크기, 종류, 저장 구조에 따른 빅데이터 처리 방식을 선택할 수 있는 능력			
	- 데이터 처리 모델을 설계하기 위하여 데이터 요구 사항을 도출하고 데이터 유형별로 분류하여 특성을 파악할 수 있는 능력			
	- 데이터의 크기, 종류, 저장 구조 특성에 부합하는 데이터 처리 모델 및 제약 조건을 설계할 수 있는 능력			
	- 완료된 데이터 처리 모델 설계에 대한 결과의 타당성을 검증할 수 있는 능력			

피드백

1. 서술형 시험

- 처리할 데이터의 크기, 종류, 저장 구조에 대하여 기초적인 개념을 이해하고 있는지 평가하고, 피평가자가 잘못 이해한 개념에 대하여 정확한 개념을 피드백해 준다.
- 빅데이터 처리에 필요한 기술 요소의 역할 및 개념을 이해하고 있는지 평가하여 결과를 피드백해 준다.
- 빅데이터 처리 모델의 설계 과정을 평가하는 것으로 제시한 목표를 달성했는지를 평가하며, 평가 결과가 일정 수준 이하인 학습자에게는 학습 목표에 미달된 사항과 보충해야 할 부분에 대하여 피드백해 준다.

2. 평가자 체크리스트

- 처리할 데이터의 크기, 종류, 저장 구조에 대한 학습 내용을 체크리스트로 평가한 후, 미흡한 개념에 대하여 정확한 개념을 피드백해 준다.
- 빅데이터 처리에 필요한 영역별 기술 요소에 대한 학습 내용을 체크리스트로 평가한 후, 부족한 항목에 대해서는 핵심 내용을 설명해 준다.
- 빅데이터 처리 모델 설계를 위한 절차에 대한 학습 내용을 체크리스트로 평가한 후, 누락되거나 부족한 절차에 대해서 핵심 내용을 설명해 준다.

학습 1	빅데이터 처리 시스템 설계하기
학습 2	빅데이터 처리 시스템 구성하기
학습 3	빅데이터 처리 시스템 수행 모듈 개발하기

2-1. 빅데이터 처리 시스템 구성

학습 목표

- 빅데이터 처리 시스템 설계서에 따라 빅데이터 처리 하드웨어와 소프트웨어를 설치하고 처리 시스템 개발 환경 설정을 할 수 있다.
- 빅데이터 처리 시스템을 효율적으로 사용하기 위해 사용자 혹은 작업 종류에 따라 시스템 자원을 조정할 수 있다.
- 주어진 테스트 절차에 의해 데이터 처리 시스템 설치 완료 여부를 확인할 수 있다.

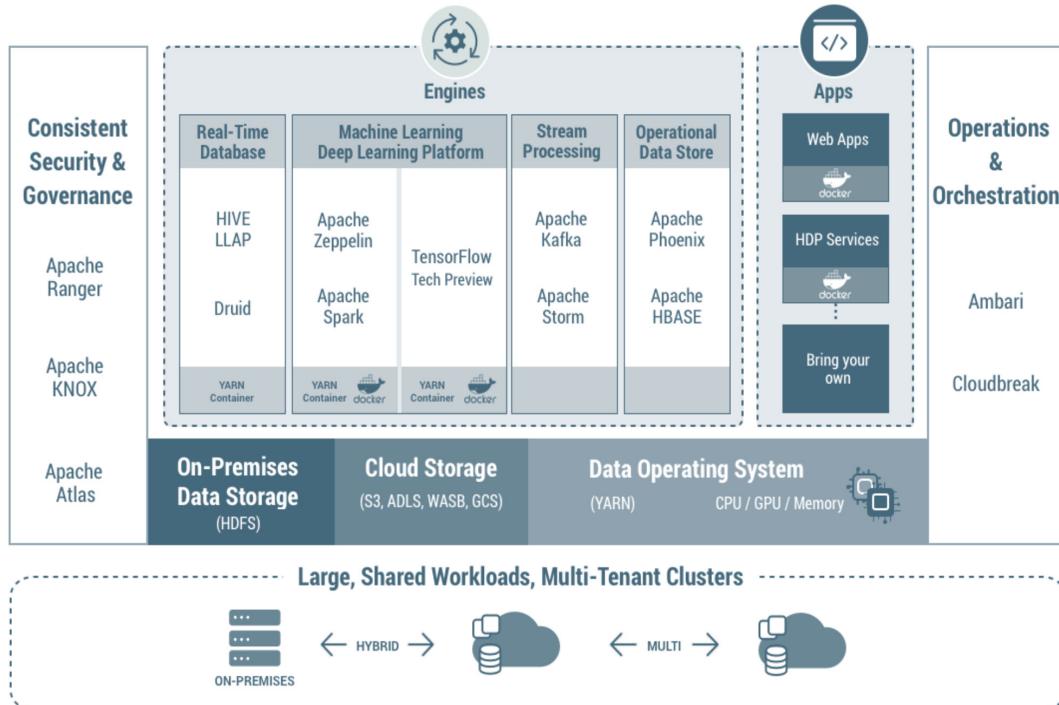
필요 지식 /

① HDP(Hortonworks Data Platform)

HDP는 기업이 정형 및 비정형 데이터로부터 통찰력을 확보할 수 있도록 방대한 멀티소스 데이터 세트를 분산 저장하고 처리할 수 있는 오픈 소스 프레임워크이다. HDP는 민첩한 애플리케이션 배포, 머신러닝 및 딥러닝 워크로드, 실시간 데이터 웨어하우스, 보안, 거버넌스 기능을 제공한다.

1. HDP 기술 아키텍처

HDP 기술 아키텍처는 다양한 인프라 환경(On-Premises, Cloud)에서 설치 가능하며, 빅 데이터 분산 처리에 필요한 핵심적인 오픈 소스 소프트웨어 스택을 데이터 수집/처리/저장/시각화 단계별로 제공한다. 또 이렇게 다양한 오픈 소스 소프트웨어를 웹 화면에서 손쉽게 설치할 수 있는 Ambari라는 도구를 제공하여, 설치의 복잡성을 줄여 준다. 또 설치된 오픈 소스의 상태 및 자원 상황을 쉽게 모니터링할 수 있는 기능도 제공한다.



출처: Hortonworks 공식 홈페이지(<https://ko.hortonworks.com>). 2019. 08. 09. 스크린샷.
[그림 2-1] Hortonworks HDP 아키텍처

2. HDP 제공 오픈 소스 소프트웨어 목록 및 특징

(1) HDP 버전별 오픈 소스 지원 목록



출처: Hortonworks 공식 홈페이지(<https://hortonworks.com/products/data-platforms/hdp/>). 2019. 08. 09. 스크린샷.

[그림 2-2] Hortonworks Data Platform 지원 오픈 소스

(2) HDP 핵심 오픈 소스 이하

<표 1-1> 오픈 소스 소프트웨어 기능 설명

오픈 소스 SW	기능 설명	관련 SW
Hadoop / YARN	분산 환경에서 파일을 저장하고, 분산 처리 시스템으로 대용량의 데이터를 처리/분석할 수 있는 프레임워크	HDFS, MapReduce
Oozie	Hadoop 기반의 다양한 분산 처리 작업 들을 하나의 워크플로로 연결하여 실행할 수 있게 해 주는 도구	
Hive	SQL 방식으로 대용량의 데이터를 검색 및 분석할 수 있는 기술로 내부적으로 Tez 엔진을 이용해 분산 파일을 처리	Impala, Presto
Spark	MapReduce와 달리 메모리 기반으로 대용량 데이터를 처리할 수 있는 분산 처리 프레임워크로 Spark SQL, Spark Streaming 등 다양한 처리 엔진을 제공	Apache Flink Apache Storm
Kafka	대량의 메시지를 수집하는 모듈과 소비하는 모듈 사이에 메시지 버퍼 역할을 수행하는 분산 메시지 큐 기술로 기존 메시지 큐와 달리 디스크 기반의 메시지 큐를 제공하여 실시간 처리에서 핵심적인 오픈 소스	
Zeppelin	대용량 데이터를 검색 및 분석하기 위한 Web UI 기반의 노트북이라는 개념을 이용하여 데이터를 처리 및 시각화할 수 있는 기술	Jupyter Notebook
Zookeeper	분산 서버에서 실행되는 오픈 소스들의 상태를 관리하고, 주요 값의 동기화를 유지하여 분산된 서버들의 가용성을 높여 주는 기술	
HBase	HDFS 기반의 칼럼 기반의 분산 데이터베이스로 대용량의 데이터에 빠르게 검색 및 수정이 가능	

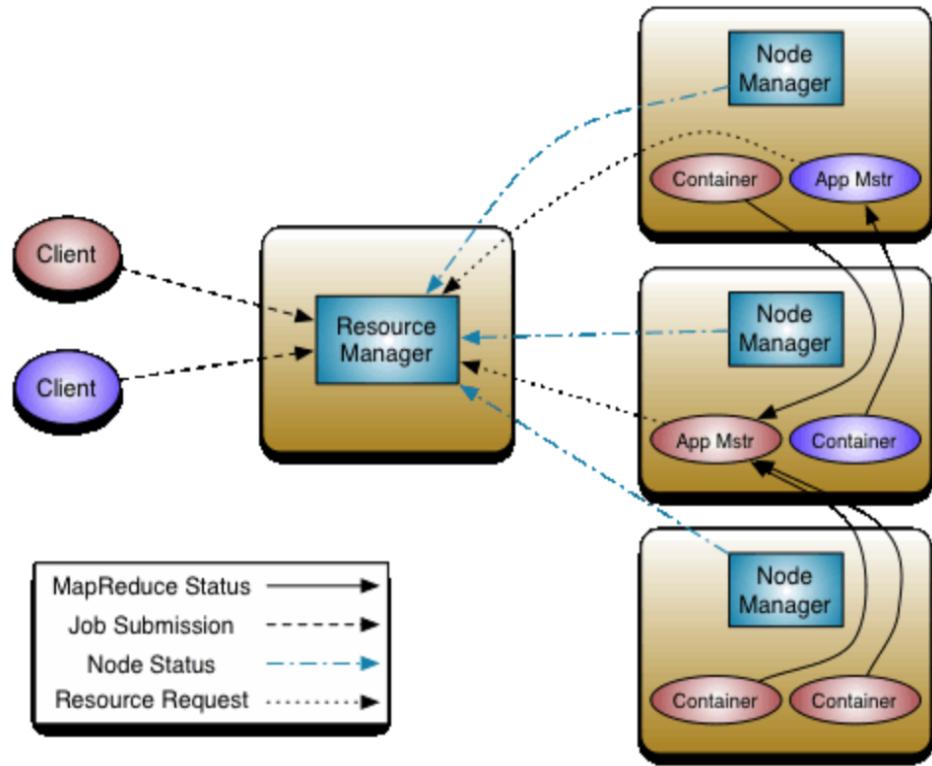
(3) Hadoop 클러스터 자원 관리 핵심 기술인 YARN 이해

YARN(Yet Another Resource Negotiator)은 Hadoop V2.0에서 도입한 클러스터 리소스 관리 및 애플리케이션 라이프사이클 관리를 위한 아키텍처로, 기존 Hadoop V1.0에서 자원관리 영역을 별도로 분리하여 개발된 기술이다. YARN에서 자원 관리는 리소스 매니저와 노드매니저, 애플리케이션 라이프사이클 관리 기능은 애플리케이션 마스터와 컨테이너가 담당하도록 구성하였다.

(가) YARN 구성도 및 자원 할당 흐름

클라이언트가 리소스 매니저에 애플리케이션을 제출하면, 리소스 매니저는 비어 있는 노드에서 애플리케이션 마스터를 실행하고, 애플리케이션 마스터는 작업 실행을 위한 자원을 리소스 매니저에 요청하여 자원을 할당받아 각 노드에 컨테이너를 동작시키고, 주어진 작업을 진행한다. 컨테이너는 실제 작업이 실행되는 단위이며, 컨테이너에서 작업이 종료되면 결과를 애플리케이션 마스터에게 알리고 애플리케이션 마스터는 결과를 클라이언트에게 전달한다.

이션 마스터는 모든 작업이 종료되면 리소스 매니저에 알리고 자원을 해제하는 방식으로 동작한다.



출처: Apache Hadoop 공식
홈페이지(<https://hadoop.apache.org/docs/current/hadoop-yarn/hadoop-yarn-site/YARN.html>). 2019. 08. 20. 스크린샷.
[그림 2-3] Apache Hadoop YARN 구성도

(나) YARN 핵심 구성 요소

<표 1-2> YARN 주요 구성 요소

구성 요소	기능 설명
리소스 매니저	Resource Manager(RM) Hadoop 클러스터의 자원을 관리하는 역할 애플리케이션 매니저에게 자원을 할당하거나 회수
애플리케이션 매니저	ApplicationMaster(AM) 클러스터에 배포된 애플리케이션을 실행 · 조정하는 프로그램 모든 애플리케이션은 담당하는 AM이 항상 존재
노드 매니저	NodeManager(NM) Hadoop 클러스터의 개별 컴퓨팅 노드, 즉 인프리를 관리 AM으로 배포된 애플리케이션 컨테이너의 라이프사이클을 관리하고, 각 컨테이너의 자원을 모니터링 RM에게 현재 컨테이너의 상태 정보를 공유
컨테이너	컨테이너의 기본 개념은 Memory, CPU Core, Disk 등의 물리적 자원 그 자체를 말하며 단일 노드에서도 여러 컨테이너가 존재 가능

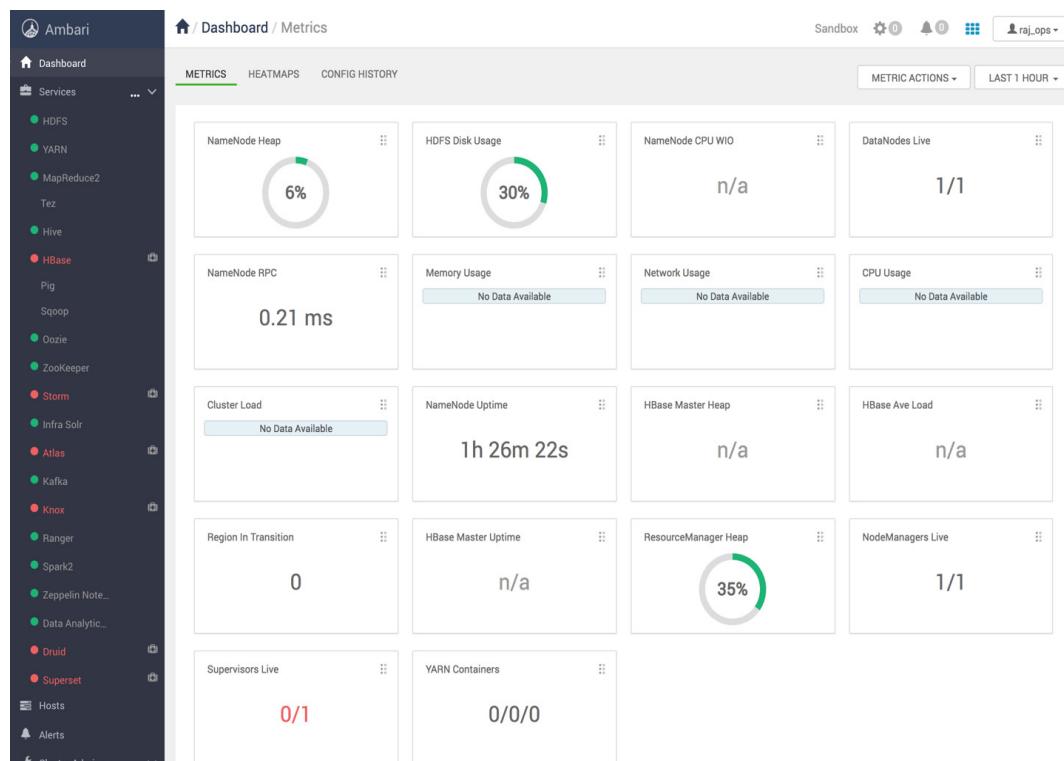
3. HDP Sandbox Image

일반적으로 빅데이터 시스템을 구성에 필요한 오픈 소스 소프트웨어(Apache Hadoop, Apache Spark, Apache Kafka, Apache Hive 등)를 설치하기 위해서는 복잡한 설치 과정과 다수의 서버가 필요하다. 하지만 대부분의 개발자는 노트북이나 데스크톱 컴퓨터를 이용해 개발 환경을 구성하게 되므로, 이러한 빅데이터 개발 환경을 구성하는 것이 쉽지 않다. HDP Sandbox는 HDP에서 제공하는 모든 오픈 소스를 설치한 상태로 이미지를 제공하여, 누구나 쉽게 빅데이터 개발 환경을 구성할 수 있도록 지원하는 가상 이미지이다.

- Sandbox Image는 외부와 독립된 가상의 환경을 제공하여 Sandbox 내부의 동작 및 장애가 다른 시스템에 영향을 주지 않도록 방지하는 용도로 활용됨.
- HDP Sandbox Image를 실행하기 위해 노트북/데스크톱 컴퓨터의 메모리(10g 이상), CPU(4core 이상)가 권장됨.

4. Ambari로 설치된 오픈 소스 소프트웨어 모니터링

Apache Ambari는 손쉬운 웹 UI 및 REST API 사용을 제공하여 아파치 하둡(Apache Hadoop) 클러스터의 관리 및 모니터링을 간소화한다. Ambari는 HDP 내에 포함되어 있으며, 클러스터를 모니터링하고, 구성을 변경할 수 있도록 하는 서비스이다.

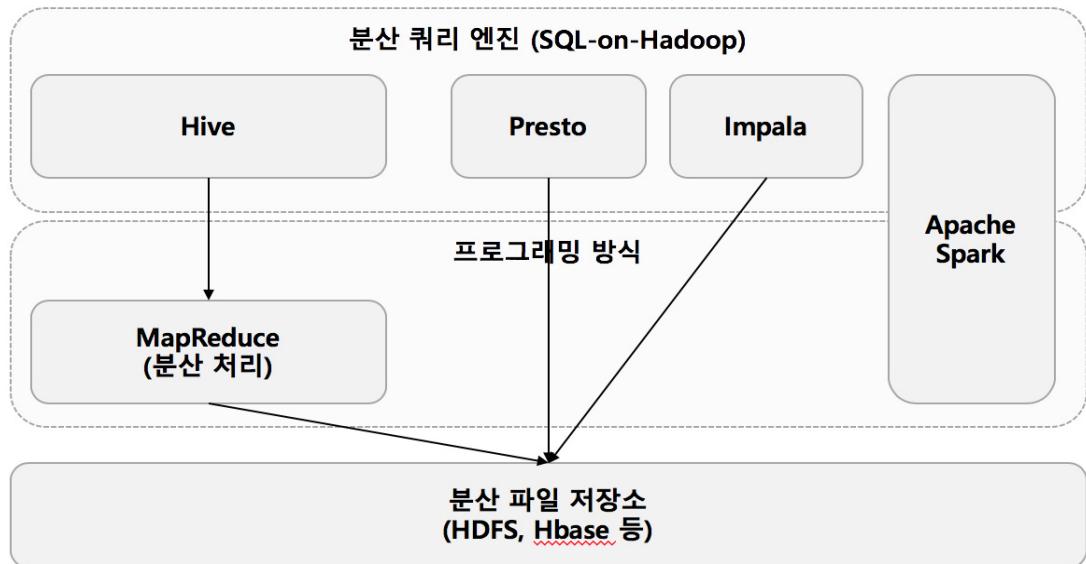


출처: Hortonworks HDP나 ambari 설치화면. 스크린샷.
[그림 2-4] Ambari 화면 예시

② 분산 처리 방식의 유형 비교

아파치 하둡(Apache Hadoop)의 HDFS(Hadoop Distributed File System)에 저장된 데이터를 분산 처리하는 방법은 크게 두 가지로 분류된다. 프로그래밍을 통해 데이터에 접근하여 처리할 로직을 직접 코드로 구현하는 방식(MapReduce, Apache Spark 등)과 SQL 구문을 통해 저장된 데이터를 처리하는 분산 쿼리 엔진 방식(sql-on-hadoop)이 존재한다.

1. 분산 처리 기술 간 관계



[그림 2-5] 분산 처리 기술 간 관계

2. 분산 배치 처리 구현 방식

분산 배치 처리 구현 방식은 크게 프로그래밍 방식과 쿼리 엔진(sql-on-hadoop) 방식으로 구분되며, 데이터 처리 목적에 따라 구분하여 활용한다.

<표 1-3> 분산 배치 처리 구현 방식

	프로그래밍 방식	분산 쿼리 엔진
처리 데이터	비구조화된 데이터 수집 데이터 원본	구조화된 데이터 칼럼 방식의 저장포맷
활용 방식	비구조화된 데이터를 구조화하여 저장 칼럼 기반 스토리지로 저장 포맷을 변환하여 조회 성능 향상	구조화되어 저장된 데이터의 검색 기능 대량의 데이터 통계 및 변환
개발 언어	Java, Scala, Python	SQL
주요 기술(오픈 소스)	MapReduce, Apache Spark	Apache Hive, Impala, Presto
장점	다양한 데이터 처리 가능 프로그래밍 로직을 통해 업무에	SQL 문법만 이해하면, 누구나 빅 데이터 분산 처리 가능

	프로그래밍 방식	분산 쿼리 엔진
	최적화된 처리 기능 구현 가능 프로그램 품질에 따라 최적의 성능 보장 개발 난이도 높음.	쉬운 구현 방법 운영 및 유지 보수 쉬움.
단점	비숙련자가 개발할 경우 성능 저하 및 오류 발생 가능성 높음. 운영 및 유지 보수 어려움.	SQL에서 지원하지 않는 기능은 실행 불가

(1) 프로그래밍 방식

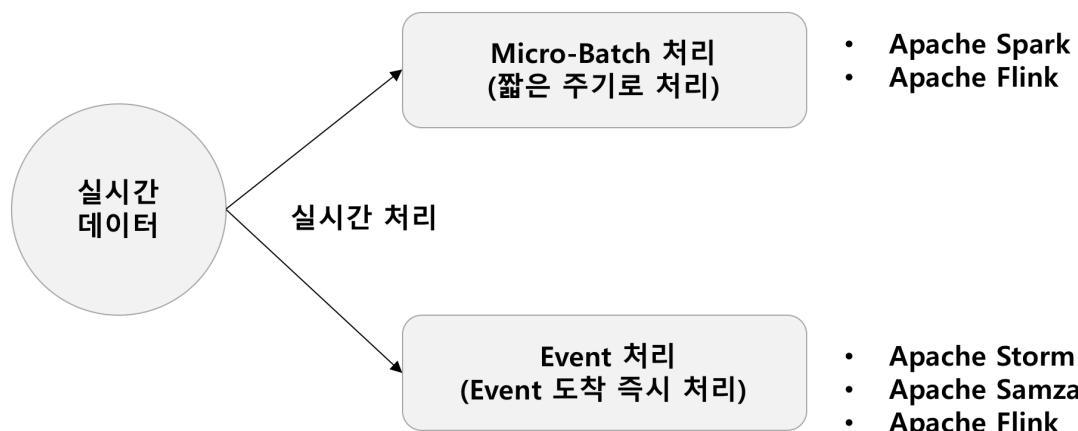
프로그래밍 방식은 초기 빅데이터 분산 처리에서 활용한 방식으로 개발자가 직접 분산 처리에 필요한 함수 및 기능적 절차를 프로그래밍을 통해 정의하고, 이를 분산하여 처리한다. 개발자의 숙련도 및 역량에 따라 성능 차이가 크고, 간단한 기능을 위해서도 높은 수준의 개발자가 필요하다. 다만, 숙련된 개발자는 최적의 성능을 보장하는 복잡한 비즈니스 로직을 제공하는 장점이 있다.

(2) 분산 쿼리 엔진(sql-on-hadoop)

분산 쿼리 엔진(sql-on-hadoop)은 분산 스토리지(hadoop 등)에 분산 저장된 데이터를 SQL 방식으로 조회할 수 있도록 지원하는 기술이다. 대용량의 안정적인 집계를 위한 방식(Hive)과 빠른 조회 성능을 지원하는 용도의 방식(Impala, presto 등)이 있다.

③ 실시간 처리 기술 유형 비교

실시간 처리 기술 유형은 실시간 데이터를 처리하는 방식에 따라 크게 두 가지 유형으로 분류된다. 실시간 이벤트 발생 즉시 처리하는 방식과 짧은 주기 동안 저장된 데이터를 빠르게 처리하는 Micro-batch 처리 방식이 있다. 초기 트위터에서 이벤트 처리 방식으로 실시간 트위트 메시지를 처리하는 Apache storm의 개발을 시작으로 최근에는 Apache spark라는 micro-batch 방식의 실시간 처리 기술을 많은 기업에서 활용하고 있다.



[그림 2-6] 실시간 처리 기술 유형

유사한 기능의 다양한 실시간 분산 처리를 지원하는 오픈 소스가 존재하며, 이들의 특징을 고려하여 시스템에 적용 여부를 결정해야 한다.

<표 2-4> 실시간 분산 처리 기술의 유형별 특징

기술	특징 및 활용	활용 기업
Storm	Mili second 이하의 빠른 처리가 필요한 요건에 중요한 경우. 처리량보다 빠른 이벤트 탐지가 중요한 경우 활용	Twitter, Yahoo!, TheWeatherChannel
Samza	Storm과 유사하면서, 자체 key-value store를 가지고 상태 관리를 지원하므로, 다양한 실시간 비즈니스가 필요한 경우 활용	LinkedIn, Yahoo!, Metamarkets
Flink	Spark과 유사하지만, min-batch가 아닌 sub-second의 빠른 처리가 중심인 비즈니스에 활용	Alibaba, ericsson, otto group
Spark	데이터 처리량을 높이는 mini-batch 기반의 처리와 상태 관리를 지원하면서, sql/ml/graph 분석을 위한 package를 지원하여 다양한 비즈니스에 적용 가능	Yahoo!, NASA JPL, eBay, Baidu

④ 이벤트 처리 기술의 유형

이벤트 처리의 특징은 데이터를 저장하기 전에 처리 및 탐지를 하고, 이후에 저장하는 방식이다. 또 입력 이벤트에 대한 정해진 응답을 제공하는 방식이 아닌, 이벤트 탐지 조건에 만족하는 반응을 하는 방식이다. 실시간 복합 이벤트 처리를 지원하는 다양한 제품들이 존재하며, 오픈 소스를 중심으로 기술의 특징을 확인한다.

1. 이벤트 처리의 유형

(1) ESP(Event Stream Processing)

ESP(Event Stream Processing)는 일반적인 모든 이벤트에 대한 처리를 하는 방식이다. 예를 들어 지난 5분간 온도 센서의 평균 온도를 계산하는 것이다.

(2) CEP(Complex Event Processing)

CEP(Complex Event Processing)는 ESP의 기능에 패턴 매칭을 추가한 개념으로, 이벤트 간의 인과관계를 정의하여 복합적인 패턴을 탐지할 수 있다. 예를 들어, 최근 5분간 실내 온도가 평균 3도 이상 오르고, 실내 먼지 센서값이 10% 이상 상승하는 경우 이벤트를 탐지한다.

2. 대표적인 오픈 소스 Esper와 Drools 비교

(1) Esper

Esper는 EpserTech가 개발한 복합 이벤트 처리 시스템이며 가장 유명한 오픈 소스 중 하나이다. 이 시스템은 이벤트 기반 반응형 프로그램의 개발자에게 Java와 C# 인

터페이스를 제공하고 있다. 또 관계형 데이터베이스를 사용했던 개발자들이 쉽게 이벤트 스트림 처리에 접근할 수 있도록 개발되었다.

(2) Drools

JBoss Community의 LGPL 라이센스로 제공하는 Drools Fusion CEP 는 사실 오픈 소스 BPM인 jBPM의 구성 요소이다. 이 오픈 소스는 Esper와는 다르게 보험사 등에서 사용하는 rule을 기반으로 동작한다. Esper와 차별성이라면 복잡한 rule을 지원하는 것이다. 고성능 처리를 목표로 하지는 않지만 복잡한 규칙을 실시간 이벤트 처리하는 데 있어서 최적의 대안이다.

(3) Drools와 Esper의 핵심 특징 비교

두 가지 이벤트 처리 오픈 소스 소프트웨어의 특징을 비교하고, 요구 사항 및 제약조건을 충족할 수 있는 제품을 선택한다.

<표 2-5> Esper vs Drools 특징 비교

구분	Drools	Esper
복합 이벤트 처리	가능	가능
Java 연동	가능	가능
룰 서버 제공	가능	상용 버전 지원
타임 윈도 지원	지원	지원
룰 정의	자체 룰 스크립트 정의	SQL 방식
룰 관리 도구	지원	상용 버전 지원
고 가용성 구성	미 지원	지원
라이센스	LGPL	GPLv2

수행 내용 / 빅데이터 처리 시스템 구성하기

재료 · 자료

- 빅데이터 처리 시스템을 설치할 서버 정보
- 빅데이터 처리 시스템 설계서
- 기존 운영 시스템 또는 신규 시스템의 아키텍처 산출물

기기(장비 · 공구)

- 리눅스 서버
- 컴퓨터, 프린터, 인터넷
- 사무 자동화 소프트웨어

안전 · 유의 사항

- 해당 사항 없음.

수행 순서

① 빅데이터 처리 시스템 개발 환경 구성에 필요한 소프트웨어를 설치한다.

1. 빅데이터 처리 시스템 개발 환경을 구성할 컴퓨터를 준비한다.

본 학습모듈에서는 Windows OS가 설치된 노트북/데스크톱 컴퓨터를 기준으로 빅데이터 처리 시스템 개발 환경을 구성한다. 개발 환경을 설치할 하드웨어 사양은 다음 표를 참고 한다.

<표 2-6> 개발 환경을 구성할 컴퓨터 하드웨어 사양

자원 구분	최소 사양	권장 사양
CPU	2CORE 이상	4CORE 이상
Memory	10G 이상	15G 이상
Disk	70G 이상	100G 이상

2. 빅데이터 처리 시스템에 필요한 소프트웨어를 설치 및 실행한다.

빅데이터 처리 시스템을 구성하기 위해서는 수많은 오픈 소스 소프트웨어를 유형(수집, 처리, 저장, 시각화, 보안 등)별로 직접 다운로드하고, 리눅스 명령어를 이용하여 직접 설치해야 한다. 하지만, 이러한 설치 방식은 전문 개발자도 실수를 많이 하는 복잡하고 어

려운 과정이므로 본 교재에서는 필요한 대표적인 오픈 소스 소프트웨어가 이미 설치되어 정상적으로 실행된 상태의 가상 환경(Sandbox)인 Hortonworks HDP Sandbox를 이용하여 설치한다. 또 가상의 개발 환경에 접속할 수 있는 접속 프로그램인 putty라는 프로그램을 설치하여 개발 환경에 접속하여 테스트를 수행한다.

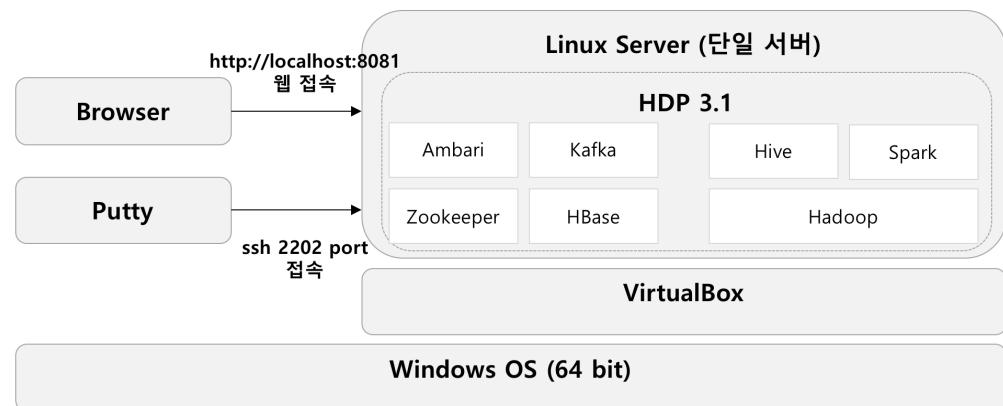
(1) 설치할 소프트웨어 목록을 확인한다.

<표 2-7> 설치할 소프트웨어 목록 및 설명

목록	설명	특징
VirtualBox	가상 머신(Virtual Machine)이란, 본인의 운영 체제 안에서 프로그램 형식으로 별도의 운영 체제(OS)를 사용하고 싶을 때 사용하는 것으로 소프트웨어 기술을 이용하여 본인의 컴퓨터에 가상의 컴퓨터를 만들어 주는 기술이다	윈도 피시에 다양한 운영 체제(리눅스 등) 설치 가능
HDP 3.0	Hortonwork에서 빅데이터 처리/분석에 필요한 핵심 오픈 소스 소프트웨어를 패키지하여 설치 및 모니터링할 수 있도록 제공하는 플랫폼이다	VirtualBox에서 실행 가능한 Sandbox로 제공
Putty	ssh와 telnet 같은 프로토콜로 윈도 및 리눅스에 접속할 수 있는 터미널이다	

(2) 분산 처리 시스템 개발 환경을 확인한다.

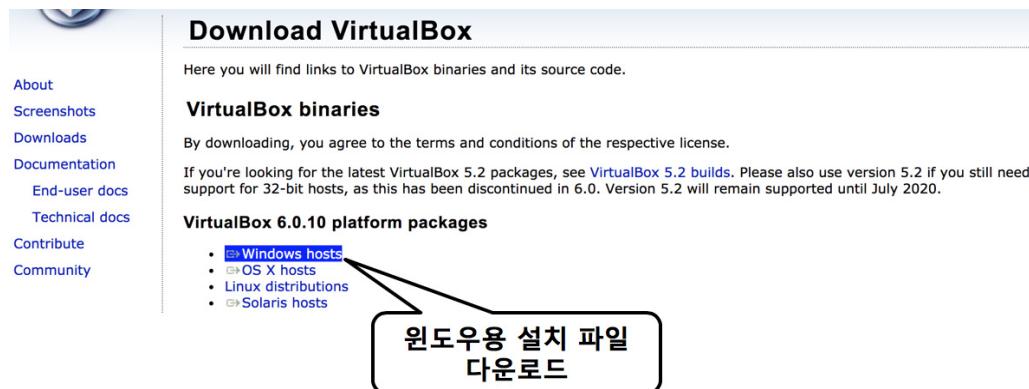
본 교재에서 구성할 개발 환경의 전체 구성도는 다음과 같다. VirtualBox에 HDP 3.0 Sandbox를 설치하여 빅데이터 처리 시스템 개발 환경을 실행한 후, Putty를 이용하여 내부 서버에 접속하여 프로그래밍을 하거나, 웹 브라우저를 통해 설치된 오픈 소스 소프트웨어의 자원 및 서비스 상태를 모니터링한다.



[그림 2-7] Hortonworks Data Platform 지원 오픈 소스

(3) VirtualBox를 설치한다.

공식 다운로드 페이지(<https://www.virtualbox.org/wiki/Downloads>)에서 사용자의 운영 체제에 맞는 설치 파일을 다운로드한다. 본 교재에서는 윈도용 VirtualBox 6.0.10 버전을 다운로드한다. 다운로드한 후 설치를 진행하여 소프트웨어를 설치한다.



출처: VirtualBox 공식 홈페이지(<https://www.virtualbox.org/wiki/Downloads>). 2019. 08. 09. 스크린샷.

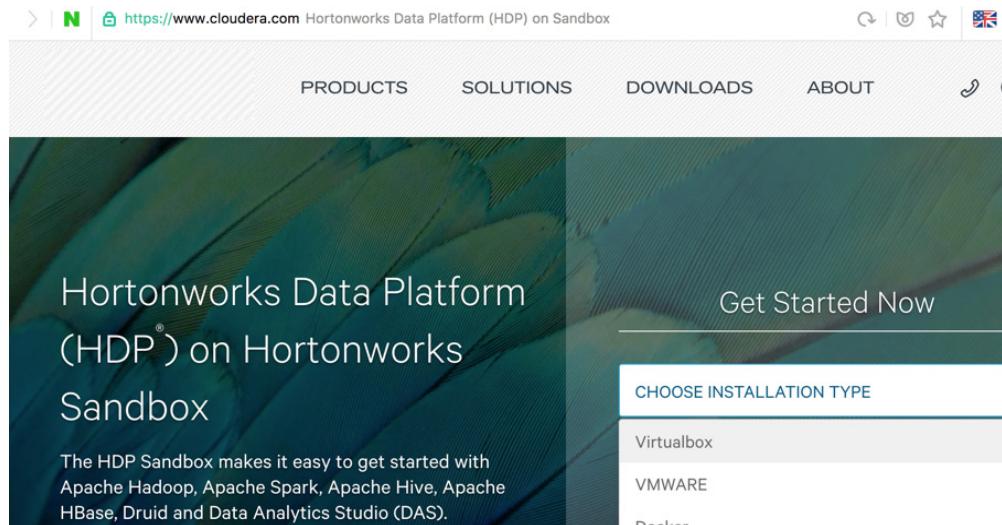
[그림 2-8] VirtualBox 공식 다운로드 페이지 화면

(4) HDP 3.0을 다운로드한다.

Hortonworks 공식 다운로드

페이지(<https://www.cloudera.com/downloads/hortonworks-sandbox/hdp.html>)에서

VirtualBox용 HDP3.0 SandBox를 다운로드한다.



출처: Hortonworks 공식 다운로드 페이지
(<https://www.cloudera.com/downloads/hortonworks-sandbox/hdp.html>). 2019. 08. 09.
스크린샷.

[그림 2-9] Hortonworks HDP Sandbox 다운로드 페이지

수행 tip

- HDP 3.0 SandBox는 총 20G 용량이므로, 인터넷 환경
이 빠른 곳에서 다운로드받는 것을 권장한다.

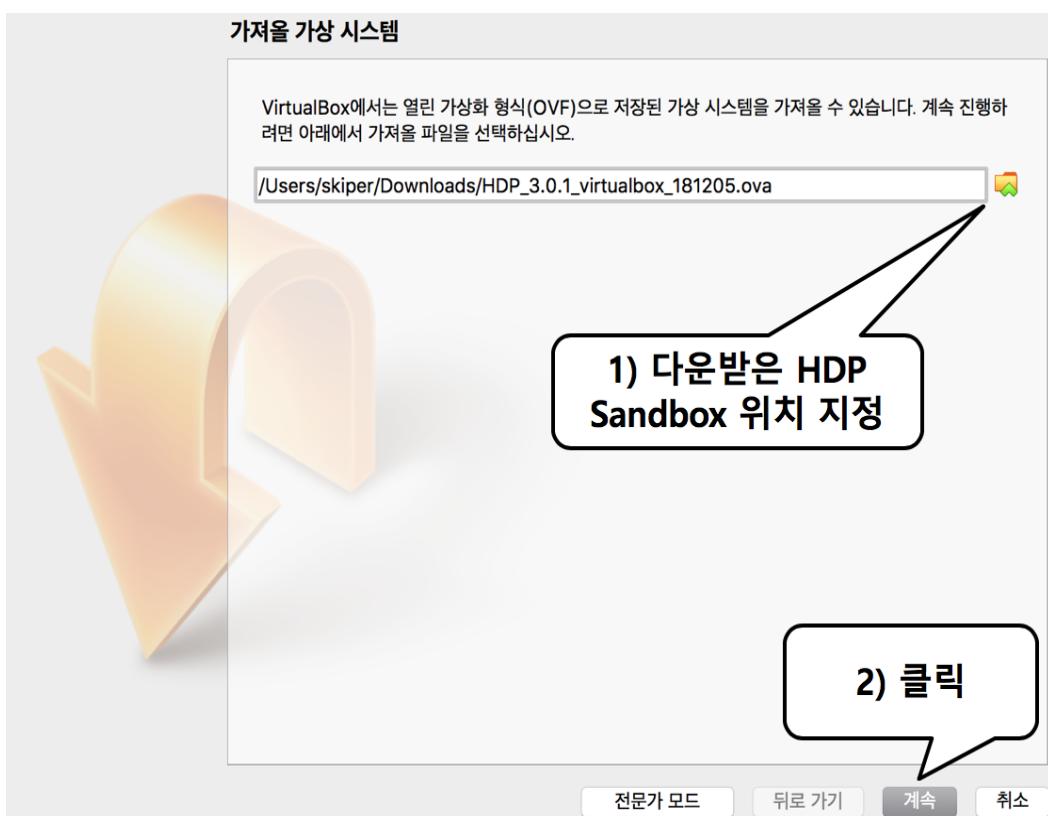
② VirtualBox를 이용하여 빅데이터 처리 시스템 개발 환경(HDP 3.0 SandBox)을 구성한다.

VirtualBox를 실행하여 다운로드한 HDP 3.0 SandBox를 실행하여, 빅데이터 처리 시스템

환경을 구성한다.



[그림 2-10] VirtualBox에서 가져오기 클릭

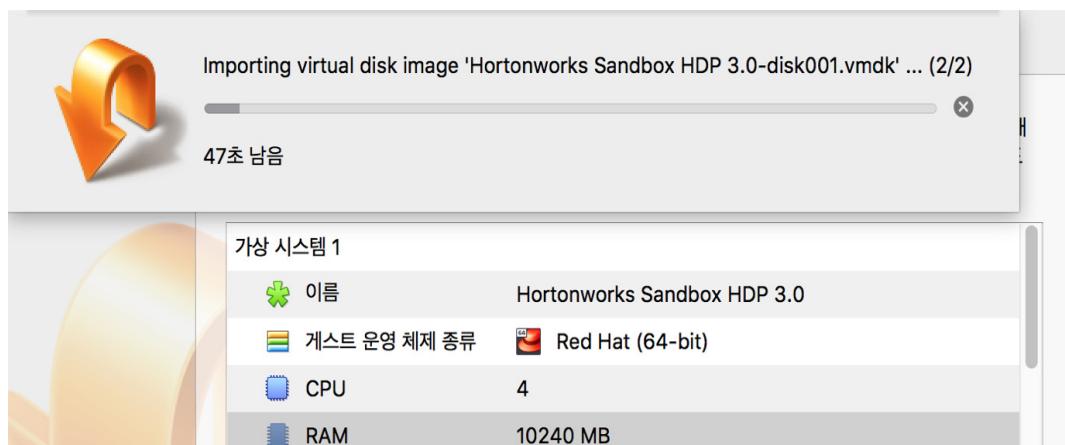


[그림 2-11] 다운로드한 HDP 3.0 Sandbox 경로 지정 후 “계속” 버튼 클릭

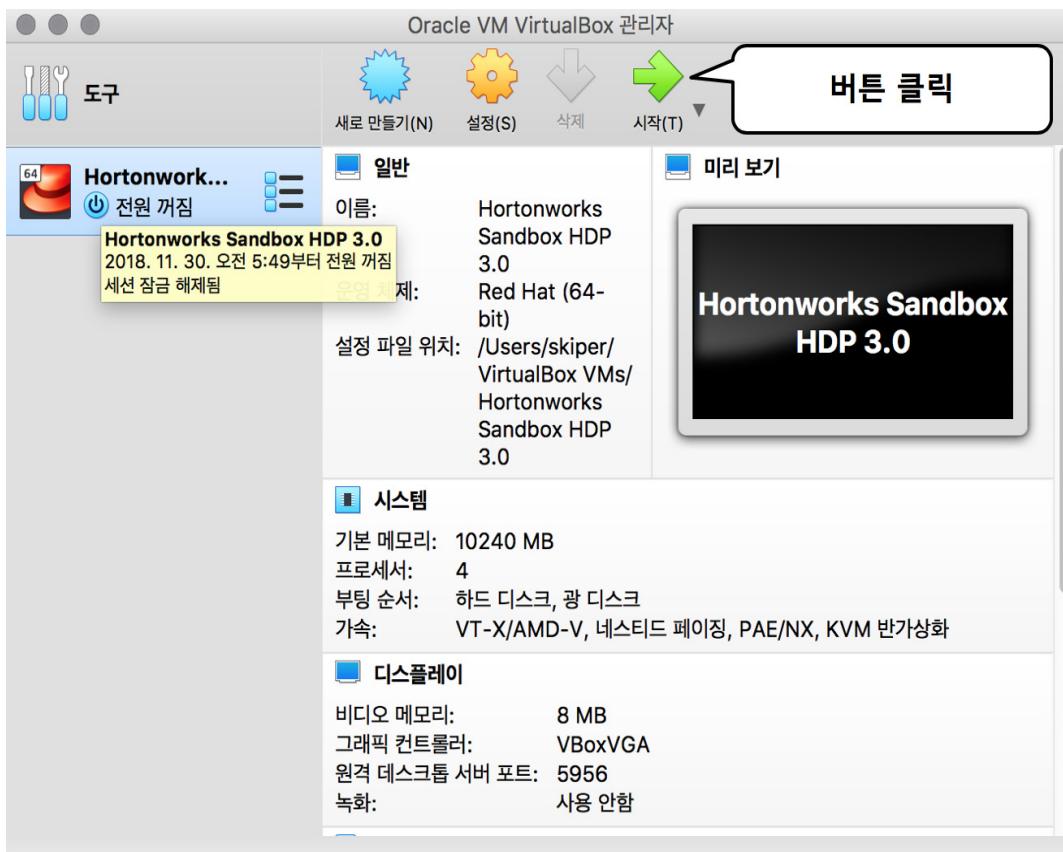
가상 시스템 설정



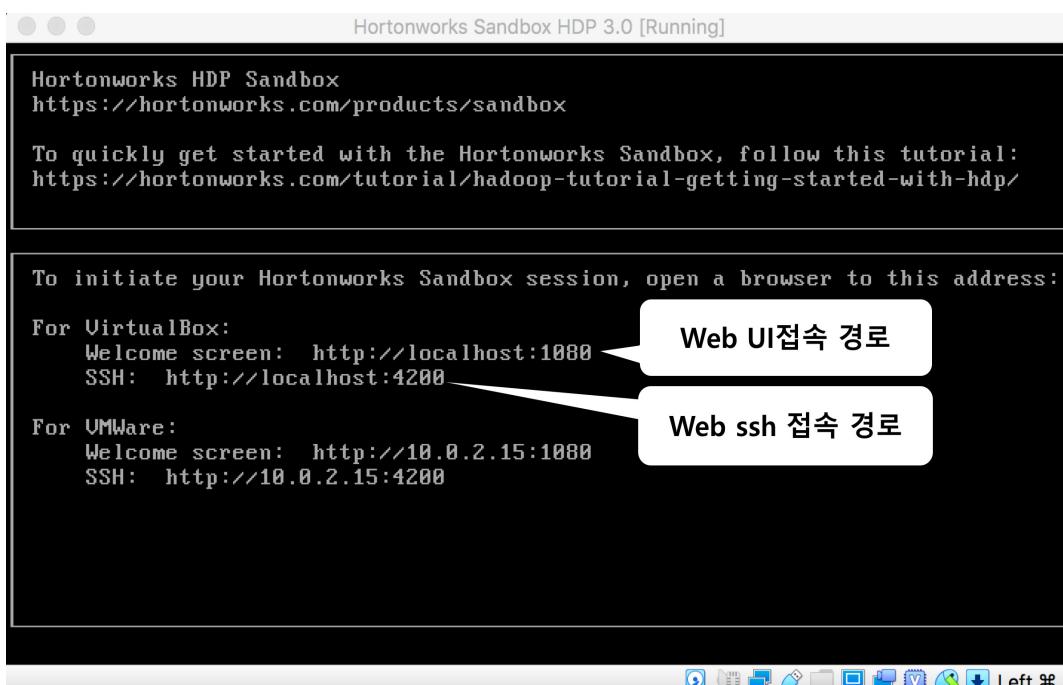
[그림 2-12] RAM의 최소 사이즈를 10G(10240 MB)로 변경 후 “가져오기” 클릭



[그림 2-13] 지정한 HDP SandBox를 VirtualBox에 가져오는 중



[그림 2-14] “시작” 버튼을 클릭하여 HDP가 설치된 가상의 서버 실행



[그림 2-15] 설치 완료 화면

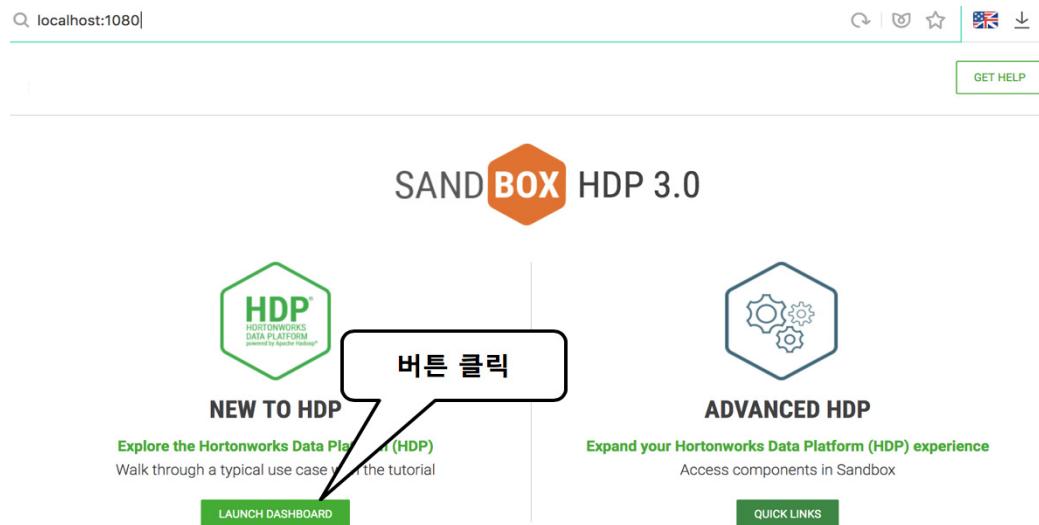
수행 tip

- VirtualBox를 설치한 데스크톱의 사양에 따라 10~30분까지 시간이 소요된다. 될 수 있는 대로 고사양의 데스크톱에 설치하는 것이 효율적이다.
- Web UI 접속 경로와 Web ssh 접속 경로를 따로 기록하여, 시스템 모니터링 및 서버 접속을 할 때에 해당 경로로 접속한다.

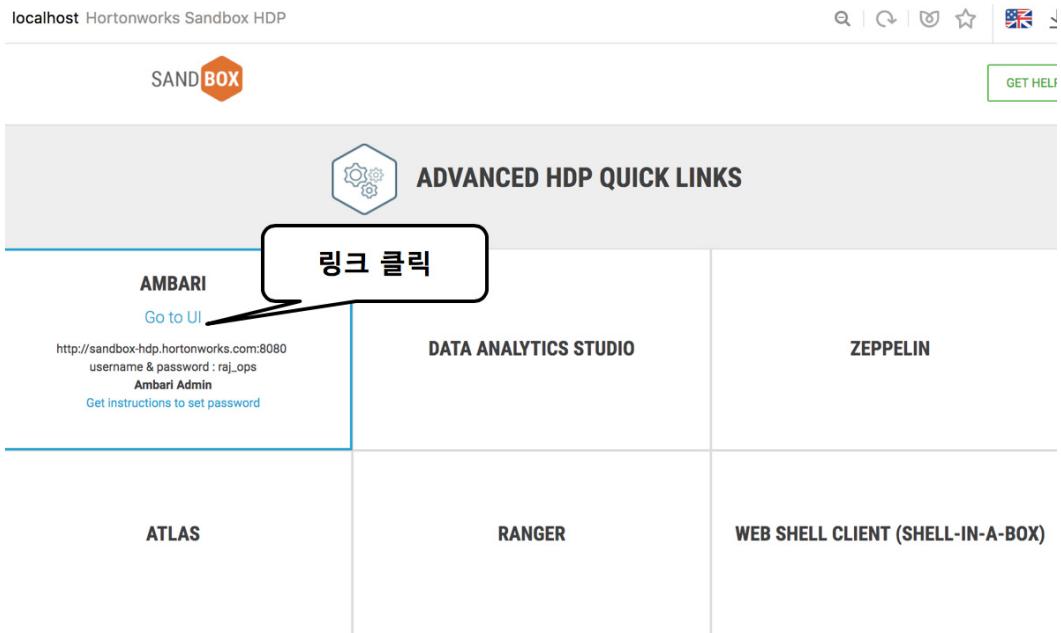
③ 설치한 빅데이터 처리 시스템인 HDP의 정상 동작 여부를 테스트한다.

1. Web UI를 통한 HDP 설치 정보를 확인한다.

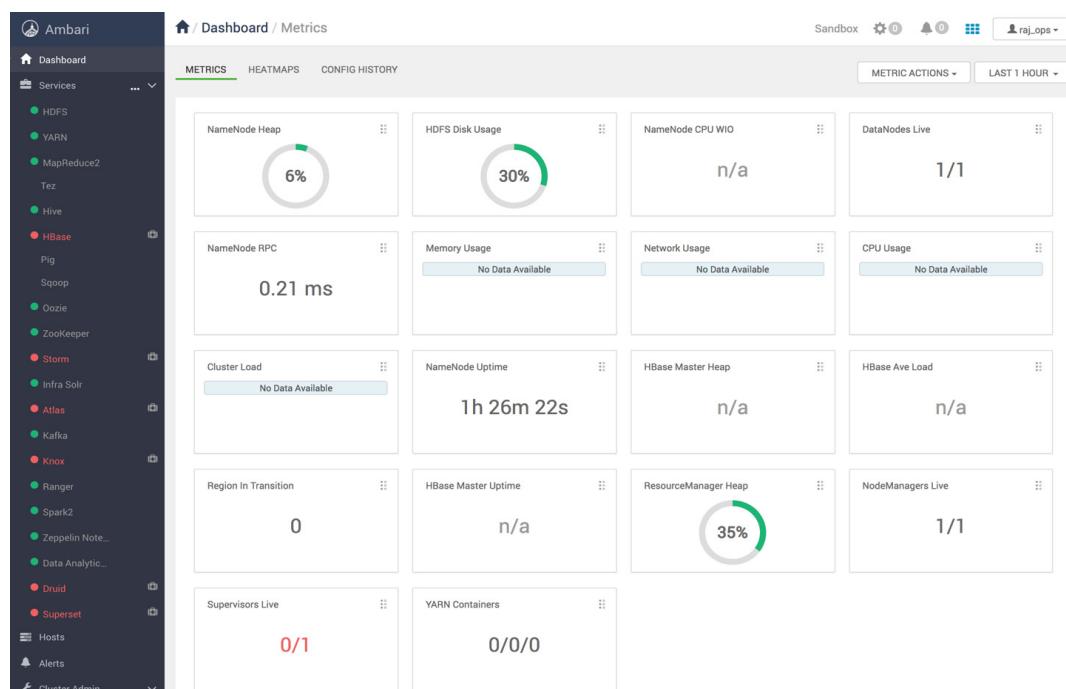
VirtualBox에 HDP를 설치 완료 후 화면에 표시되는 “Welcome Screen”의 접속 경로 (<http://localhost:1080>)으로 접속하여 HDP에서 제공하는 서비스를 확인한다.



[그림 2-16] HDP 서비스 링크 클릭



[그림 2-17] Ambari 링크 클릭



출처: 스크린샷, HDP의 Ambari 화면

[그림 2-18] Ambari에서 제공하는 HDP 서비스 모니터링 화면

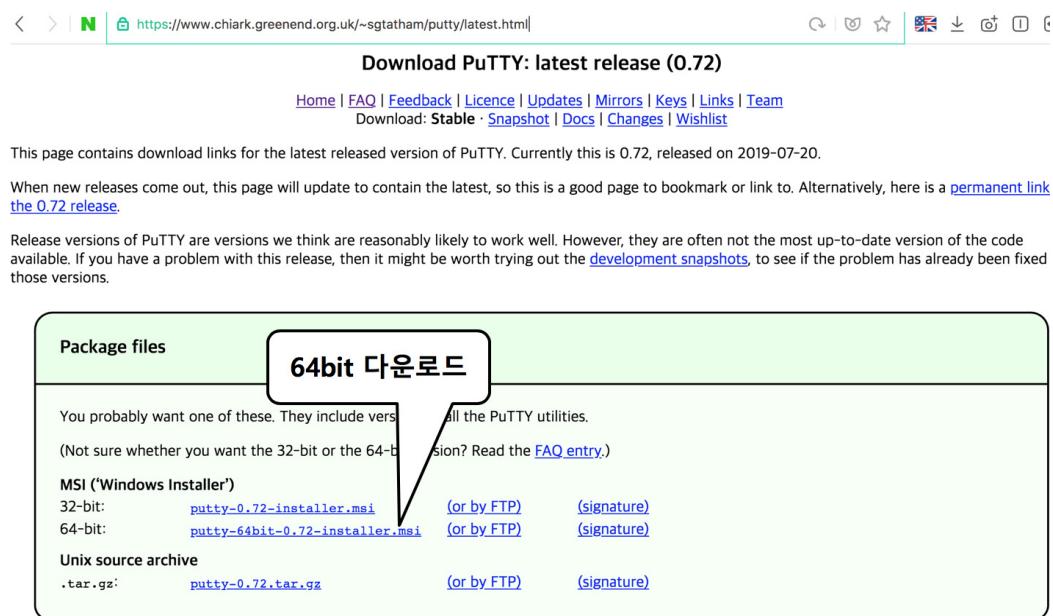
수행 tip

- 초기 VirtualBox 설치 직후에는 많은 서비스가 오류 상태로 보인다. 이는 서비스가 시작되는 데 시간이 걸리기 때문이고, 10분 정도 지나면 정상(녹색)으로 표시된다.
 - 설치가 안 된 서비스(붉은색)는 사용자가 추가로 설치할 수 있다.

2. SSH 연결을 통한 오픈 소스 소프트웨어 기능을 확인한다.

SSH 연결 도구인 putty를 이용해서 VirtualBox의 HDP가 설치된 서버에 접속하고, 설치된 오픈 소스 소프트웨어인 Hadoop, Spark의 동작 여부를 간단한 명령어를 통해 확인한다.

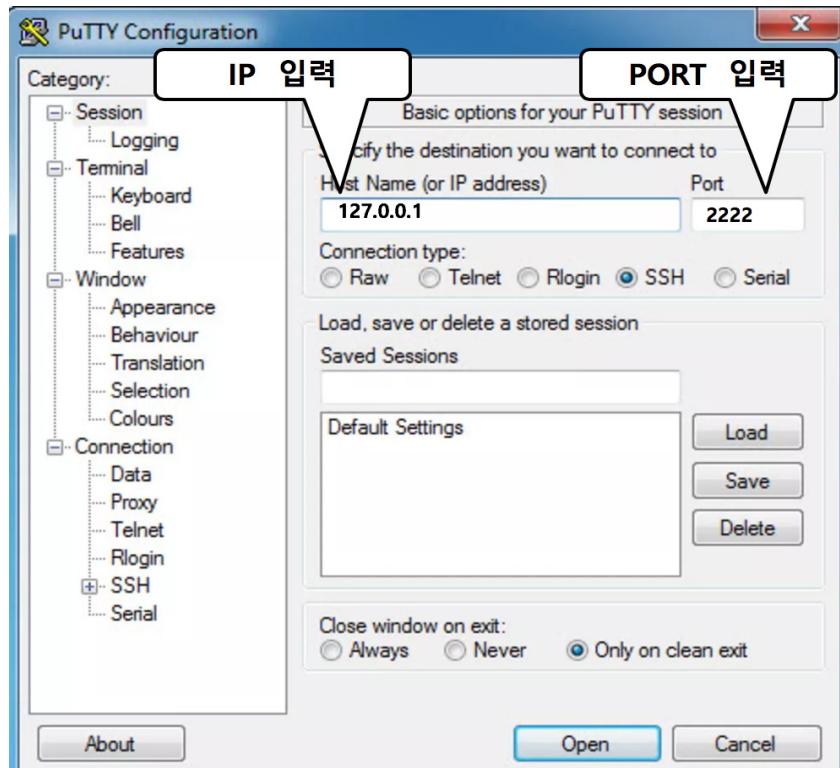
(1) Putty 설치하기



출처: Putty 공식 다운로드 페이지
(<https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html>). 2019. 08. 09. 스크린샷.
[그림 2-19] Hortonworks HDP Sandbox 다운로드 페이지

(2) Putty로 HDP 서버에 접속한다.

putty를 이용하여 HDP가 설치된 서버에 접속할 수 있는 설정을 하고, “Open” 버튼을 클릭하여 HDP 서버에 접속한다.



[그림 2-20] Putty 접속 설정하기

```
Warning: Permanently added '[localhost]:2222' (ECDSA) to the list of known hosts
.
root@localhost's password:
You are required to change your password immediately (root enforced)
Last login: Thu Aug 22 09:05:48 2019
Changing password for root.
(current) UNIX password:
New password:
Retype new password:
[root@sandbox-hdp ~]#
```

[그림 2-21] Putty로 SSH 접속한 화면

초기 root 계정의 패스워드는 hadoop로 설정되어 있으며, 접속하면 패스워드를 변경하도록 되어 있다. 새로운 패스워드를 두 번 입력하면 로그인이 완료된다.

수행 tip

- 신규 패스워드를 입력할 때 기존 패스워드와 유사하지 않으면서, 특수 기호가 포함된 여섯 자 이상으로 설정해야 패스워드가 변경된다.

(3) HDFS 기능을 확인한다.

설치된 분산 파일 시스템인 HDFS에 정상적으로 접속되고, 디렉터리 및 파일이 조회되는지 확인한다. 다음 명령어로 최상위 루트(/) 디렉터리에 포함된 파일과 디렉터리를 조회해 본다.

(가) 리눅스 명령어(Command Line)로 확인하기

```
[root@sandbox driver_data]# hadoop fs -ls /
Found 15 items
drwxrwxrwx  spark    hadoop    0 B      2019. 8. 14. 오전 5:56:56   0
drwxrwxrwx  - yarn    hadoop    0 2019-08-13 17:27 /app-logs
drwxr-xr-x  - hdfs   hdfs     spark  0 2016-10-25 07:54 /apps
drwxr-xr-x  - yarn    hadoop    0 2016-10-25 07:48 /ats
drwxr-xr-x  - hdfs   hdfs     root   0 2016-10-25 08:01 /demo
```

[그림 2-22] hdfs 명령어로 디렉터리 및 파일 조회 화면

(나) Web UI에서 확인한다.

Web UI에서 제공하는 화면을 통해서 HDFS의 디렉터리 및 파일 목록이 정상적으로 조회되는지 확인한다. 웹 브라우저에서 <http://localhost:50070> 경로를 입력하면 Hadoop의 Web UI에 접속하고, Utilities > Browse the File system을 클릭하면 HDFS에 저장된 파일 및 디렉터리 목록을 확인할 수 있다.

	Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
drwxrwxrwt	yarn	hadoop	0 B	Nov 30 2018	0	0 B	app-logs	
drwxr-xr-x	hdfs	hdfs	0 B	Nov 30 2018	0	0 B	apps	
drwxr-xr-x	yarn	hadoop	0 B	Nov 30 2018	0	0 B	ats	
drwxr-xr-x	hdfs	hdfs	0 B	Nov 30 2018	0	0 B	atsv2	
drwxr-xr-x	hdfs	hdfs	0 B	Nov 30 2018	0	0 B	hdp	
drwxr-----	livy	hdfs	0 B	Nov 30 2018	0	0 B	livy2-recovery	
drwxr-xr-x	mapred	hdfs	0 B	Nov 30 2018	0	0 B	mapred	
drwxrwxrwx	mapred	hadoop	0 B	Nov 30 2018	0	0 B	mr-history	
drwxr-xr-x	hdfs	hdfs	0 B	Nov 30 2018	0	0 B	ranger	
drwxrwxrwx	spark	hadoop	0 B	Aug 22 19:15	0	0 B	spark2-history	

[그림 2-23] Hadoop Web UI에서 확인한 HDFS 파일 목록 예시

(4) Apache Hive 기능을 확인한다.

putty를 이용하여 ssh로 접속하고, 리눅스 명령어로 hive를 실행한다. 실행된 hive 환경에서 데이터베이스 테이블을 생성하고, 정상적으로 테이블이 생성되었는지 확인한다.

```
[root@sandbox-hdp ~]# hive
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/hdp/3.0.1.0-187/hive/lib/log4j-slf4j-impl-2.10.0.jar]
SLF4J: Found binding in [jar:file:/usr/hdp/3.0.1.0-187/hadoop/lib/slf4j-log4j12-1.7.25.jar]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]
Connecting to jdbc:hive2://sandbox-hdp.hortonworks.com:2181/default;password=hive;serviceNamespace=hiveserver2
19/08/22 10:09:13 [main]: INFO jdbc.HiveConnection: Connected to sandbox-hdp.hortonworks.
Connected to: Apache Hive (version 3.1.0.3.0.1.0-187)
Driver: Hive JDBC (version 3.1.0.3.0.1.0-187)
Transaction isolation: TRANSACTION_REPEATABLE_READ
Beeline version 3.1.0.3.0.1.0-187 by Apache Hive
```

[그림 2-24] hive 명령어를 실행한 화면

```
0: jdbc:hive2://sandbox-hdp.hortonworks.com:2> show TABLES;
INFO : Compiling command(queryId=hive_20190822100922_ef09c593-3282-4b9b-acbb-e514577d9ffb):
INFO : Semantic Analysis Completed (retrial = false)
INFO : Returning Hive schema: Schema(fieldSchemas:[FieldSchema(name:tab_name, type:string, s:null)]
INFO : Completed compiling command(queryId=hive_20190822100922_ef09c593-3282-4b9b-acbb-e514577d9ffb):
INFO : Executing command(queryId=hive_20190822100922_ef09c593-3282-4b9b-acbb-e514577d9ffb):
INFO : Starting task [Stage-0:DDL] in serial mode
INFO : Completed executing command(queryId=hive_20190822100922_ef09c593-3282-4b9b-acbb-e514577d9ffb):
INFO : OK
+-----+
| tab_name |
+-----+
|          |
+-----+
|          | Transaction isolation: TRANSACTION_REPEATABLE_READ
+-----+
No rows selected (0.636 seconds)
```

[그림 2-25] show TABLES; 명령어로 현재 테이블 목록 조회 화면

```
0: jdbc:hive2://sandbox-hdp.hortonworks.com:2> CREATE TABLE test(foo INT, bar STRING);
INFO : Compiling command(queryId=hive_20190822100954_97662224-4344-4e33-b57f-c5c839d02831)
)
INFO : Semantic Analysis Completed (retrial = false)
INFO : Returning Hive schema: Schema(fieldSchemas:null, properties:null)
INFO : Completed compiling command(queryId=hive_20190822100954_97662224-4344-4e33-b57f-c5c839d02831)
)
INFO : Executing command(queryId=hive_20190822100954_97662224-4344-4e33-b57f-c5c839d02831)
)
INFO : Starting task [Stage-0:DDL] in serial mode
INFO : Completed executing command(queryId=hive_20190822100954_97662224-4344-4e33-b57f-c5c839d02831)
INFO : OK
No rows affected (1.859 seconds)
```

[그림 2-26] CREATE TABLE test(foo INT, bar STRING); 명령어로 테이블 생성 화면

```
0: jdbc:hive2://sandbox-hdp.hortonworks.com:2> show TABLES;
INFO : Compiling command(queryId=hive_20190822101000_035b0733-def2-482e-bcaf-c6b842683037)
INFO : Semantic Analysis Completed (retrial = false)
INFO : Returning Hive schema: Schema(fieldSchemas:[FieldSchema(name:tab_name, type:string, s:null)]
INFO : Completed compiling command(queryId=hive_20190822101000_035b0733-def2-482e-bcaf-c6b842683037)
INFO : Executing command(queryId=hive_20190822101000_035b0733-def2-482e-bcaf-c6b842683037)
INFO : Starting task [Stage-0:DDL] in serial mode
INFO : Completed executing command(queryId=hive_20190822101000_035b0733-def2-482e-bcaf-c6b842683037)
INFO : OK
+-----+
| tab_name |
+-----+
|          |
+-----+
|          | Semantic Analysis Completed (retrial = false)
+-----+
| test    | Returning Hive schema: Schema(fieldSchemas:null, properties:null)
```

[그림 2-27] show TABLES; 명령어로 생성한 테이블 조회 화면

```

0: jdbc:hive2://sandbox-hdp.hortonworks.com:2> describe test;
INFO : Compiling command(queryId=hive_20190822101403_3f7d7530-1603-4fef-8ada-91bb780f5df3)
INFO : Semantic Analysis Completed (retrial = false)
INFO : Returning Hive schema: Schema(fieldSchemas:[FieldSchema(name:col_name, type:string, a(name:data_type, type:string, comment:from deserializer), FieldSchema(name:comment, type:s properties:null)
INFO : Completed compiling command(queryId=hive_20190822101403_3f7d7530-1603-4fef-8ada-91bb780f5df3)
INFO : Executing command(queryId=hive_20190822101403_3f7d7530-1603-4fef-8ada-91bb780f5df3)
INFO : Starting task [Stage-0:DDL] in serial mode
INFO : Completed executing command(queryId=hive_20190822101403_3f7d7530-1603-4fef-8ada-91bb780f5df3)
INFO : OK
+-----+-----+-----+
| col_name | data_type | comment |
+-----+-----+-----+
| foo      | int       |          |
| bar      | string    |          |
+-----+-----+-----+
2 rows selected (0.33 seconds)

```

[그림 2-28] describe test; 명령어로 생성한 test 테이블의 속성(칼럼) 정보 확인 화면

(5) Apache Spark 서비스 및 기능을 확인한다.

Apache Spark 클러스터가 정상적으로 동작하고 있는지 Web UI에 접속한 후, 기본 제공되는 간단한 Spark 프로그램을 실행하고, Spark-shell을 이용하여 spark 함수를 직접 실행해 본다. 그리고 실행된 작업들이 Spark Job 목록에 정상적으로 기록되는지 확인한다.

(가) Web UI에서 확인한다.

웹 브라우저에서 localhost:1808로 접속하면, Apache Spark History 페이지가 보인다. 아직까지 Apache Spark 프로그램을 실행하지 않아서 작업 중 또는 작업 완료된 목록이 표시되지 않는다.

History Server

Event log directory: hdfs:///spark2-history/

Last updated: 2019-08-22 19:17:13

Client local time zone: Asia/Seoul

No completed applications found!

Did you specify the correct logging directory? Please verify your setting of spark.history.fs.logDirectory. It is also possible that your application did not run to completion or did not stop the SparkContext.

Show incomplete applications

출처: 스크릿샤. HDP로 설치한 spark history server의 메인화면

[그림 2-29] 웹 브라우저에서 Apache Spark History Server 접속 화면

(나) 리눅스 명령어에서 예제 프로그램을 실행한다.

putty를 이용하여 ssh로 접속하고, 리눅스 명령어로 Apache Spark를 설치할 때 제공되는 샘플 프로그램을 실행해 본다. 이 샘플 프로그램은 수학의 파이값을 대략적으로 계산하는 프로그램이다. 다음 명령어로 실행한다.

```
spark-submit --master yarn-client --class org.apache.spark.examples.SparkPi
/usr/hdp/3.0.1.0-187/spark2/examples/jars/spark-examples_2.11-2.3.1.3.0.1.0-
```

187.jar 10

```
[root@sandbox-hdp ~]# spark-submit --master yarn-client --class org.apache.spark.examples.SparkPi  
ples/jars/spark-examples_2.11-2.3.1.3.0.1.0-187.jar 10  
SPARK_MAJOR_VERSION is set to 2, using Spark2  
Warning: Master yarn-client is deprecated since 2.0. Please use master "yarn" with specified depl  
19/08/22 10:24:47 INFO SparkContext: Running Spark version 2.3.1.3.0.1.0-187  
19/08/22 10:24:47 INFO SparkContext: Submitted application: Spark Pi  
19/08/22 10:24:47 INFO SecurityManager: Changing view acls to: root  
19/08/22 10:24:47 INFO SecurityManager: Changing modify acls to: root  
19/08/22 10:24:47 INFO SecurityManager: Changing view acls groups to:  
19/08/22 10:24:47 INFO SecurityManager: Changing modify acls groups to:  
19/08/22 10:24:47 INFO SecurityManager: SecurityManager: authentication disabled; ui acls disable  
Set(root); groups with view permissions: Set(); users with modify permissions: Set(root); groups  
19/08/22 10:24:48 INFO Utils: Successfully started service 'sparkDriver' on port 33735.  
19/08/22 10:27:08 INFO YarnScheduler: Removed TaskSet 0.0, whose tasks have all completed, from po  
19/08/22 10:27:08 INFO DAGScheduler: ResultStage 0 (reduce at SparkPi.scala:38) finished in 32.06  
19/08/22 10:27:08 INFO DAGScheduler: Job 0 finished: reduce at SparkPi.scala:38 took 33.022923 s  
Pi is roughly 3.1397671397671396
```

결과 출력

```
19/08/22 10:27:17 INFO ShutdownHookManager: Shutdown hook called  
19/08/22 10:27:17 INFO ShutdownHookManager: Deleting directory /tmp/spark-5371aa31-d4e6-4e75-bb70  
19/08/22 10:27:17 INFO ShutdownHookManager: Deleting directory /tmp/spark-5b70334c-66f5-4b27-9893
```

[그림 2-30] Spark 샘플 프로그램을 실행한 화면

실행한 명령어의 세부 내용을 보면 —master yarn-client는 spark를 yarn이라는 자원 관리자를 통해서 실행한다는 의미이다. —class는 실제 실행할 프로그램 명칭을 의미한다. 또 그 뒤의 파일명은 실행할 파일의 경로를 의미하고, 마지막 10은 프로그램을 실행할 때 전달되는 입력값이다. 구체적인 설명은 분산 처리 수행 모듈 개발하기에서 다루도록 한다.

(다) spark-shell 명령어로 테스트한다.

putty로 접속한 터미널에서 spark의 주요 함수를 직접 실행하도록 spark-shell로 spark 실행 환경에 접속하여 주요 기능을 테스트한다. 다음 명령어로 spark 실행 모드로 접속한다.

spark-shell —master local

```
[root@sandbox-hdp ~]# spark-shell --master local  
SPARK_MAJOR_VERSION is set to 2, using Spark2  
Setting default log level to "WARN".  
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).  
Spark context Web UI available at http://sandbox-hdp.hortonworks.com:4040  
Spark context available as 'sc' (master = local, app id = local-1566471392411).  
Spark session available as 'spark'.  
Welcome to  
    ____  
   / _ \_\_  ___ / /_\_\_  
  _\ \ V \ - V \ - ` \_ / ' \_ /  
 /_ / . \^_,/_ / / \_ \_\_ /  
  \_ \_ Components  
version 2.3.1.3.0.1.0-187  
The command to start the Apache Spark S  
  
Using Scala version 2.11.8 (OpenJDK 64-Bit Server VM, Java 1.8.0_191)  
Type in expressions to have them evaluated.  
Type :help for more information.  
Install On Ubuntu  
scala>
```

2.1. Create a new RDD

[그림 2-31] spark-shell로 spark 실행 환경에 접속한 화면

실행 명령어의 —master local은 로컬 환경에서 spark를 실행한다는 의미이다. 즉, spark를 현재 서버에서만 실행한다는 의미이다.

수행 tip

- spark의 분산 처리 성능 효과를 보기 위해서는 다음 명령어 형식으로 실행한다.
- spark-shell --master yarn-client --executor-cores 3 —number-executors 4
- master yarn-client로 hadoop의 분산 서버에서 동작하도록 지정한다.
- —number-executors 4로 네 개의 프로세스가 동작하도록 지정하고,
- --executor-cores 3 각 executor 프로세스별로 세 개씩의 cpu 코어를 할당한다.

1~100까지 숫자 배열 생성
10개의 파티션으로 저장

```
scala> val parallelSeq = sc.parallelize(1 to 100, 10)
parallelSeq: org.apache.spark.rdd.RDD[Int] = ParallelCollectionRDD[0] at parallelize at <console>:24
1~100에서 홀수만 추출
scala> val parallelOdd = parallelSeq.filter(n => n % 2 == 1)
parallelOdd: org.apache.spark.rdd.RDD[Int] = MapPartitionsRDD[1] at filter at <console>:25
추출된 홀수값을 더함
scala> val sum = parallelOdd.sum
sum: Double = 2500.0
```

[그림 2-32] spark-shell로 1~100까지 숫자를 생성하고, 홀수만 더한 값 출력 화면

(라) spark 실행 이력을 확인한다.

웹 브라우저에서 localhost:1808로 접속하여, Apache Spark 프로그램을 실행한 이력이 표시되는지 확인한다.

Spark History Server							
3.1.3.0.1.0-187							
Event log directory: hdfs://spark2-history/							
Last updated: 2019-08-22 19:38:08							
Client local time zone: Asia/Seoul							
Search: <input type="text"/>							
App ID	App Name	Started	Completed	Duration	Spark User	Last Updated	
local-1566469901799	Spark shell	2019-08-22 19:31:41	2019-08-22 19:34:56	3.3 min	root	2019-08-22 19:34:56	
local-1566469793178	Spark shell	2019-08-22 19:29:52	2019-08-22 19:30:48	56 s	root	2019-08-22 19:30:48	
application_1566465026373_0002	Spark Pi	2019-08-22 19:24:47	2019-08-22 19:27:08	2.4 min	root	2019-08-22 19:27:08	

실행한 spark job 이력

[그림 2-33] spark history server에서 실행된 spark job을 확인하는 화면

(6) Apache Kafka 기능을 확인한다.

putty로 HDP 서버에 접속하여 Apache kafka에 topic을 생성하고, 생성된 topic에 데

이터를 전송하고, 전송된 데이터를 정상적으로 가져오는지 확인한다.

```
/usr/hdp/current/kafka-broker/bin/kafka-topics.sh --create --zookeeper localhost:2181 --replication-factor 1 --partitions 1 --topic test_topic  
[root@sandbox-hdp ~]# /usr/hdp/current/kafka-broker/bin/kafka-topics.sh --create --zookeeper localhost:2181 --replication-factor 1 --partitions 1 --topic test_topic  
WARNING: Due to limitations in metric names, topics with a period ('.') or underscore('_') could collide. To avoid issues it is best to use either, but not both.  
Created topic "test_topic".
```

[그림 2-34] Kafka topic(test_topic)을 생성하는 화면

```
/usr/hdp/current/kafka-broker/bin/kafka-topics.sh --list --zookeeper localhost:2181  
[root@sandbox-hdp ~]# /usr/hdp/current/kafka-broker/bin/kafka-topics.sh --list --zookeeper localhost:2181  
ATLAS_ENTITIES  
ATLAS_HOOK  
consumer_offsets  
test_topic
```

[그림 2-35] 생성된 topic(test_topic)을 확인하는 화면

```
/usr/hdp/current/kafka-broker/bin/kafka-console-consumer.sh --bootstrap-server sandbox-hdp.hortonworks.com:6667 --topic test_topic  
[root@sandbox-hdp ~]# /usr/hdp/current/kafka-broker/bin/kafka-console-producer.sh --broker-list sandbox-hdp.hortonworks.com:6667 --topic test_topic  
>my kafka test message
```

[그림 2-36] kafka-console-producer 명령어로 topic_test에 메시지를 전달하는 화면(my kafka message를 입력)

```
/usr/hdp/current/kafka-broker/bin/kafka-console-consumer.sh --bootstrap-server sandbox-hdp.hortonworks.com:6667 --topic test_topic  
[root@sandbox-hdp ~]# /usr/hdp/current/kafka-broker/bin/kafka-console-consumer.sh --bootstrap-server sandbox-hdp.hortonworks.com:6667 --topic test_topic  
my kafka test message
```

[그림 2-37] kafka-console-consumer 명령어로 test_topic에서 메시지를 수신하는 화면(my kafka message가 출력된다.)

(7) Hadoop Application 실행 목록을 조회한다.

아파치 하둡(Apache Hadoop)에서 실행되는 모든 분산 처리 모듈은 아래의 웹에서 정보를 기록한다. 성공 및 실패 원인 파악 및 실행 속도 등을 확인할 수 있다.

ID	User	Name	Application Type	Queue	Priority	StartTime	FinishTime	Status	FinalStatus	Running Containers	Allocated CPU	Allocated Memory	% of Queue	Progress	Tracking UI	Blacklisted Nodes
application_1565716898955_0002	root	Hive-b043113f-93aa-426f-b281-974387513983	TEZ	default	0	Wed Aug 14 02:32:48 2019	Wed Aug 14 02:35:08 +0900 2019	FINISHED	SUCCEEDED	N/A	N/A	N/A	0.0	0.0	History	N/A
application_1565716898955_0001	root	Hive-3f71e999-37ff-4370-8581-7fb4e253585e	TEZ	default	0	Wed Aug 14 02:27:24 2019	Wed Aug 14 02:37:39 +0900 2019	FINISHED	SUCCEEDED	N/A	N/A	N/A	0.0	0.0	History	N/A

출처: HDP로 설치한 Hadoop 모니터링 화면

[그림 2-38] Hadoop All Application Web UI 화면(8088 PORT)

③ 설치한 빅데이터 처리 시스템인 HDP에 설치된 소프트웨어의 자원을 조정한다.

빅데이터 처리 시스템의 처리 유형 및 목적에 따라 관련 오픈 소스 소프트웨어의 하드웨어 자원 할당을 최적화한다. 또 비기능 요구 사항에 명시된 처리량, 데이터 지연, 가용성, 내구성 등의 요건을 만족하기 위한 소프트웨어 설정도 최적화한다.

1. YARN 설정 정보를 확인하고 목적에 따라 최적화한다.

웹 브라우저에서 <http://localhost:8080>으로 접속하여 왼쪽 메뉴의 “YARN”을 선택하고, 오른쪽 화면에서 “CONFIG”를 클릭한다. YARN의 메모리 설정은 `yarn-site.xml` 파일을 수정하거나, Ambari Web UI에서 확인 및 변경이 가능하다.

(1) Ambari Web UI를 통한 YARN 설정 정보를 확인한다.

[그림 2-39] Ambari YARN 설정 화면

수행 tip

- 기본값은 yarn-default.xml에 정의되어 있다.
- 이 기본값을 변경하려면, yarn-site.xml에 새롭게 정의한다. 기존 yarn-default.xml은 수정할 수 없다.
- HDP로 설치하면 /etc/hadoop/yarn-site.xml에 파일이 존재한다.

(2) YARN 주요 설정 정보를 변경한다.

<표 2-8> YARN 주요 설정 정보

설정값	설명
yarn.nodemanager.resource.memory-mb	각 노드(서버)에서 컨테이너에 설정할 수 있는 메모리 총량 노드의 OS에 필요한 메모리를 제외하고 설정(예를 들어 메모리가 32G인 경우 운영 체제용 4G를 제외하고 28G를 설정)
yarn.nodemanager.resource.cpu-vcores	각 노드(서버)에서 컨테이너에 설정할 수 있는 CPU의 개수
yarn.scheduler.maximum-allocation-mb	하나의 컨테이너에 할당 가능한 메모리의 최댓값(기본 8G)
yarn.scheduler.minimum-allocation-mb	하나의 컨테이너에 할당할 수 있는 메모리의 최솟값(기본 1G)
yarn.nodemanager.vmem-pmem-ratio	실제 메모리 대비 가상 메모리 사용 비율 mapreduce.map.memory.mb * 설정값의 비율로 사용 가능 메모리를 1G로 설정하고, 이 값을 10으로 설정하면 가상메모리 10G 사용 가능
yarn.nodemanager.vmem-check-enabled	가상 메모리 사용량 제한(true: 메모리 사용량을 넘어서면 컨테이너 종료)
yarn.nodemanager.pmem-check-enabled	물리 메모리 사용량 제한(true: 메모리 사용량을 넘어서면 컨테이너 종료)

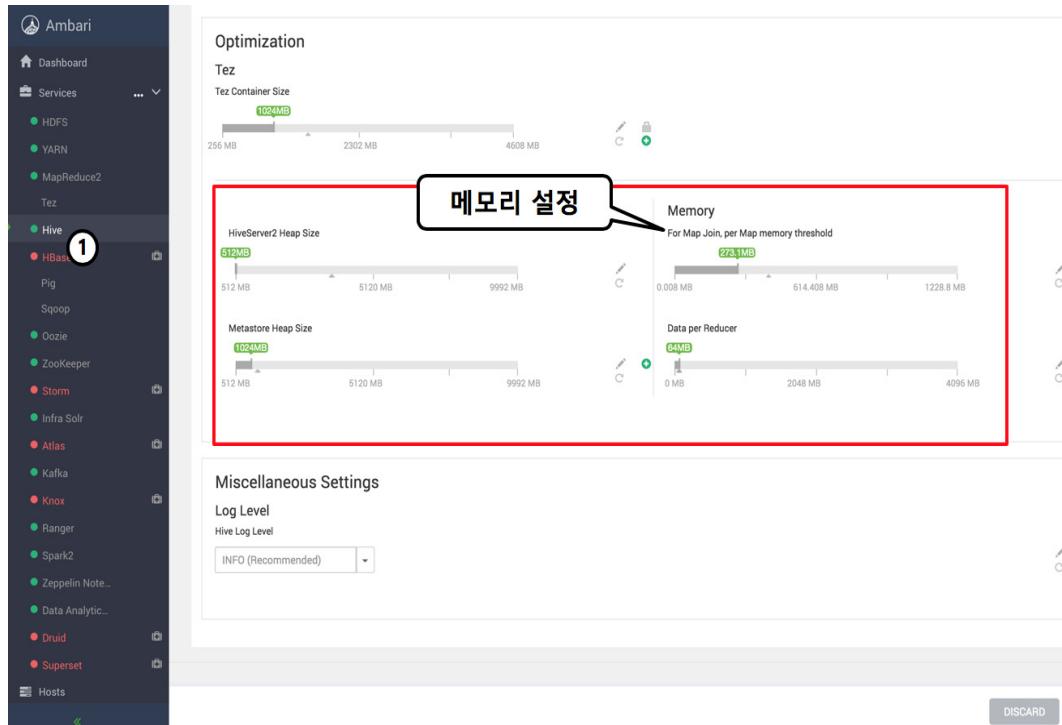
(3) 수정한 설정값을 시스템에 반영한다.

yarn-site.xml에 변경할 값을 수정 및 저장한 후, hive를 재시작한다.

2. Hive 설정 정보를 확인하고 목적에 따라 최적화한다.

웹 브라우저에서 <http://localhost:8080>으로 접속하여 왼쪽 메뉴의 “Hive을 선택하고, 오른쪽 화면에서 ”CONFIG“를 클릭한다. Hive 설정은 hive-site.xml 파일을 수정하거나, Ambari Web UI에서 확인 및 변경 가능하다.

(1) Ambari Web UI에서 Hive 설정 확인 및 변경한다.



[그림 2-40] Ambari Hive 설정화면

(2) Hive 주요 설정 정보를 변경한다.

<표 2-9> Hive 주요 설정 정보

설정값	설명
hive.execution.engine	Hive SQL을 실행하는 엔진을 선택한다.(tez 기본)
hive.tez.container.size	Tez 엔진에서 사용하는 컨테이너 메모리 사이즈(MB)
hive.tez.java.opts	Tez 엔진에서 사용하는 java 프로세스 메모리 사이즈(MB)
hive.vectorized.execution.enabled	true로 설정 시 벡터화 처리를 이용하면 한번에 1024개의 row를 처리하여 개선 가능(filter, 집계 등에 활용)
hive.cbo.enable	true로 설정하면 하이브 메타의 정보를 이용하여 쿼리의 성능 향상
hive.stats.autogather	

(3) 수정한 설정값을 시스템에 반영한다.

(가) Hive 실행 환경에서 set 명령어로 설정을 적용한다.

현재 사용자가 접속한 session에서만 적용되는 설정이며, session(현재 터미널)이 종료되면 해당 값이 원래의 값으로 복구된다.

```

0: jdbc:hive2://bdavm11:2181/default> set hive.exec.max.dynamic.partitions.pernode=2000;
No rows affected (0.007 seconds)
0: jdbc:hive2://bdavm11:2181/default> set hive.exec.max.dynamic.partitions.pernode;
+-----+
|          set          |
+-----+
|  hive.exec.max.dynamic.partitions.pernode=2000  |
+-----+
1 row selected (0.028 seconds)

```

[그림 2-41] hive 실행 환경에서 설정 변경 화면 예시

(나) Hive 환경 설정 영구적으로 적용하기

hive-site.xml에 변경할 값을 수정 및 저장한 후, hive를 재시작한다.

3. Spark 설정 정보를 확인하고 목적에 따라 최적화한다.

웹 브라우저에서 <http://localhost:8080>으로 접속하여 왼쪽 메뉴의 “Spark2”를 선택하고, 오른쪽 화면에서 ”CONFIG“를 클릭한다. Spark2 기본 설정은 spark-default.conf 파일을 수정하거나, Ambari Web UI에서 확인 및 변경 가능하다.

(1) Ambari Web UI에서 Spark 설정을 확인 및 변경한다.

The screenshot shows the Ambari Web UI interface. On the left, there's a sidebar with various service icons. The 'Spark2' icon is highlighted with a red circle labeled '1'. At the top right, there are tabs for 'SUMMARY' and 'CONFIGS', with 'CONFIGS' being active and highlighted with a red circle labeled '2'. The main content area displays configuration groups like 'Advanced livy2-client-conf', 'Advanced livy2-conf', etc. A large red box highlights the 'Advanced spark2-env' section. Inside this section, there are several configuration parameters: 'hive_kerberos_keytab', 'hive_kerberos_principal', 'spark_daemon_memory' (which has a tooltip explaining it's 'Memory for Master, Worker and history server (default: 2G)'), 'Spark Log directory', 'Spark PID directory', and 'spark_thrift_cmd_opts'. A callout bubble points to the 'spark_daemon_memory' input field. At the bottom right of the configuration panel, there are 'DISCARD' and 'SAVE' buttons.

[그림 2-42] Ambari Spark 설정 화면

(2) Spark 주요 설정 정보를 변경한다.

<표 2-10> spark-default.conf 주요 설정 정보

설정값	설명
spark.driver.cores	Driver 프로세스에게 할당할 코어 수
spark.driver.memory	Driver 프로세스에게 할당할 메모리 사이즈(기본 1G)
spark.executor.memory	Executor 프로세스당 사용할 메모리 용량 JVM 메모리 설정 형식과 동일(예: 512m, 2g)
spark.executors.cores	애플리케이션에 사용될 코어 개수에 대한 설정들

설정값	설명
	Yarn에서 각 Executor에 설정된 개수의 코어를 할당
spark.cores.max	모든 Executor들이 사용할 총 코어 개수의 최댓값 지정
spark.speculation	ture로 설정 시 느린 Task들을 다른 노드에 복사하여 재실행하여 처리 성능 향상

(3) 수정한 설정값을 시스템에 반영한다.

Apache Spark의 설정을 적용하는 방법은 여러 가지가 존재하며, 사용자가 아무런 설정을 변경하지 않으면 자동으로 기본 설정(spark-defaults.conf)이 적용된다.

(가) Spark 프로그래밍에서 SparkConf 객체를 이용하여 설정값을 적용한다.

실행할 spark application을 개발하는 시점에 필요한 설정 정보를 SparkConf라는 객체를 이용하여 설정한다. 다음 예시는 spark executor의 개수를 네 개로 설정하고, 각 executor에 다섯 개의 cpu core를 할당하는 코드이다.

```
SparkConf conf = new SparkConf()
    .set("spark.executor.instances", "4")
    .set("spark.executor.cores", "5");
```

[그림 2-43] SparkConf로 executor 설정하는 화면 예시

(나) 기본 제공되는 spark-submit 명령어로 spark application 실행 시점에 적용한다.

작성된 코드를 빌드한 실행 파일을 실행하는 시점에 spark-submit 명령어의 인자로 spark 설정을 동적으로 지정하여 실행한다.

```
spark-submit \
--class <main-class> \
--master <master-url> \
--deploy-mode <deploy-mode> \
--conf <key>=<value> \
... # other options
<application-jar> \
<application-arguments>
```

```
spark-submit \
--class org.apache.spark.examples.SparkPi \
--master spark://207.184.161.138:7077 \
--deploy-mode cluster \
--executor-memory 20G \
--total-executor-cores 100 \
/path/to/examples.jar \
1000
```

[그림 2-44] spark-submit 실행 화면 예시

수행 tip

- spark-submit과 spark-defaults.conf에서 사용하는 설정 변수명이 서로 다른 경우가 있으므로, spark 공식 홈페이지를 참고하여 설정한다.

(다) 기본 설정을 영구히 적용한다.

spark-defaults.conf를 수정하고, 저장한 후에 spark cluster를 다시 시작한다.

학습 2 교수 · 학습 방법

교수 방법

- 교수자가 빅데이터 분산 처리에 필요한 오픈 소스 소프트웨어의 특징을 유형별로 비교하여 학습자가 이해할 수 있도록 정보를 전달한다.
- 학습자가 직접 빅데이터 분산 처리에 필요한 오픈 소스 소프트웨어를 정상적으로 설치하고 실행할 수 있도록 실습을 통해 학습하게 한다.
- 학습자가 설치된 오픈 소스의 환경 설정 정보를 이해할 수 있도록, 각 설정값들의 동작 방식 및 특징에 대하여 학습하게 한다.
- 교수자는 설치된 오픈 소스 소프트웨어를 제대로 테스트하고 있는지 관찰하고, 피드백을 제공한다.

학습 방법

- 빅데이터 분산 처리에 필요한 오픈 소스 소프트웨어의 유형별 특징을 이해하고, 궁금한 점을 질문한다.
- 실습할 때 설치할 오픈 소스 소프트웨어에 대한 자료를 인터넷으로 검색하면서, 설치 방법 및 필요한 환경 설정 정보를 조사한다.
- 설치한 오픈 소스의 환경 설정 파라미터에 대하여 실습을 통해 직접 변경하면서, 동작방식 및 성능에 미치는 영향을 확인하면서 학습한다.
- 실습을 하면서 실행한 명령어나, 오류 발생 시 원인 및 해결 방법 등을 별도 문서로 정리하여 학습한다.

학습 2 평 가

평가 준거

- 평가자는 학습자가 학습 목표를 성공적으로 달성하였는지를 평가해야 한다.
- 평가자는 다음 사항을 평가해야 한다.

학습 내용	학습 목표	성취수준		
		상	중	하
빅데이터 처리 시스템 구성	- 빅데이터 처리 시스템 설계서에 따라 빅데이터 처리 하드웨어와 <u>소프트웨어</u> 를 설치하고 처리 시스템 개발 환경 설정을 할 수 있다.			
	- 빅데이터 처리 시스템을 효율적으로 사용하기 위해 사용자 또는 작업 종류에 따라 시스템 자원을 조정할 수 있다.			
	- 주어진 테스트 절차에 의해 데이터 처리 시스템 설치 완료 여부를 확인할 수 있다.			

평가 방법

- 문제해결 시나리오

학습 내용	평가 항목	성취수준		
		상	중	하
빅데이터 처리 시스템 구성	- 빅데이터 처리 시스템 설계서에 따라 빅데이터 처리 <u>소프트웨어</u> 를 설치하고 처리 시스템 개발 환경 설정을 할 수 있는 능력			
	- 빅데이터 처리 시스템을 효율적으로 사용하기 위해 사용자 또는 작업 종류에 따라 시스템 자원을 조정할 수 있는 능력			
	- 주어진 테스트 절차에 의해 데이터 처리 시스템 설치 완료 여부를 확인할 수 있는 능력			

• 평가자 체크리스트

학습 내용	평가 항목	성취수준		
		상	중	하
빅데이터 처리 시스템 구성	- 빅데이터 처리 시스템 설계서에 따라 빅데이터 처리 소프트웨어를 설치하고 처리 시스템 개발 환경 설정을 할 수 있는 능력			
	- 빅데이터 처리 시스템을 효율적으로 사용하기 위해 사용자 또는 작업 종류에 따라 시스템 자원을 조정할 수 있는 능력			
	- 주어진 테스트 절차에 의해 데이터 처리 시스템 설치 완료 여부를 확인할 수 있는 능력			

피드백

1. 문제해결 시나리오

- 빅데이터 처리 소프트웨어의 설치 절차 숙지 여부, 설치 절차에 따라 정상적인 개발 환경 구성 여부 등에 대하여 평가하고, 성취 수준이 낮은 학습자들에게는 주요 절차를 다시 설명해 준다.
- 설치된 빅데이터 처리 소프트웨어의 정상 동작을 테스트한 결과가 정상적으로 출력되는지 확인하고, 오류가 발생한 경우 원인 및 해결 방안에 대한 가이드를 제공한다.

2. 평가자 체크리스트

- 빅데이터 처리 소프트웨어 설치에 대한 학습 내용을 체크리스트로 평가한 후 미흡한 부분에 대해서 정확한 설치 절차를 설명해 준다.
- 설치된 빅데이터 처리 소프트웨어의 정상 동작을 테스트하는 방법을 체크리스트로 평가한 후, 미흡한 부분에 대해서 정확한 테스트 방법을 설명해 준다.

학습 1	빅데이터 처리 시스템 설계하기
학습 2	빅데이터 처리 시스템 구성하기
학습 3	빅데이터 처리 시스템 수행 모듈 개발하기

3-1. 빅데이터 분산 처리 수행 모듈 개발

학습 목표

- 분산 처리 모델에 따라 대규모 데이터를 병렬 처리하는 분산 처리 수행 모듈을 작성할 수 있다.
- 분산 처리를 수행하기 위하여, 스크립트를 작성하고 등록할 수 있다.
- 주어진 테스트 절차에 의해서 데이터 분산 처리 완료 및 성공 여부를 테스트하고, 실패 시 문제점을 파악하고 처리하는 모듈을 수정할 수 있다.

필요 지식 /

① 분산 처리 오픈 소스 기술 비교(MapReduce vs Apache Hive)

초기 MapReduce 기반의 프로그래밍을 통해 분산 데이터를 처리하면서, 숙련된 고급 개발자 수준의 프로그래밍 역량이 필요하였고, 프로그래밍 오류 또는 비숙련된 코드 구현으로 인한 성능 저하 같은 이슈가 발생하였다. 이를 해결하기 위해 다수의 개발자, 분석가가 활용하는 SQL을 이용하여 분산된 데이터를 처리할 수 있는 기능이 등장하게 되었다. 이 때 Apache Hadoop에 저장된 데이터를 SQL 쿼리로 처리할 수 있는 기능을 제공하게 되면서, SQL-on-Hadoop이라는 기술로 명명되게 되었고, 이후 다양한 Sql-on-Hadoop 기술들이 등장하게 된다. 여기서는 대표적인 기술 요소 간의 특징을 확인해 본다.

<표 3-1> MapReduce vs Hive 특징 비교

구분	MapReduce	Apache Hive
데이터 저장소	HDFS만 지원	HDFS, HBase, 외부 테이블 등 지원
개발 언어	Java, Python, Scala 등	HQL(SQL을 지원하는 쿼리 언어)
처리 데이터	구조화 및 비구조화 데이터 처리	구조화 데이터만 처리
데이터 처리 프레임워크	Java 기반의 Map Reduce 프로그램	HQL(내부적으로 MapReduce 프로그램으로 변환하여 최적화)
개발 난이도	어려움.	쉬움.
처리 성능	개발자 역량에 따라 성능 최적화 가능	자체 최적화 알고리즘을 통해 최적화
운영/유지 보수	어려움(변경 사항 적용에 시간 소요)	쉬움(SQL 구문만 수정하여 배포)

② SQL-on-hadoop 기술 비교

분산 저장된 데이터를 SQL 기반으로 처리 · 분석할 수 있는 기술은 점차 발전하고 있다. 초기 Apache Hive와 같이 대량의 데이터를 한 번에 분석하기 오랜 시간 동안 분산 처리를 진행하는 방식에서 최근 빠른 데이터 검색이 가능한 대화형(Interactive) SQL을 지원하는 기술이 많이 활용되고 있다. 또 Apache Spark의 경우 프로그래밍 방식의 배치 처리와 SQL-on-hadoop 방식의 대화형 쿼리를 모두 지원하는 특징을 가진다.

<표 3-2> SQL-on-hadoop 기술 비교

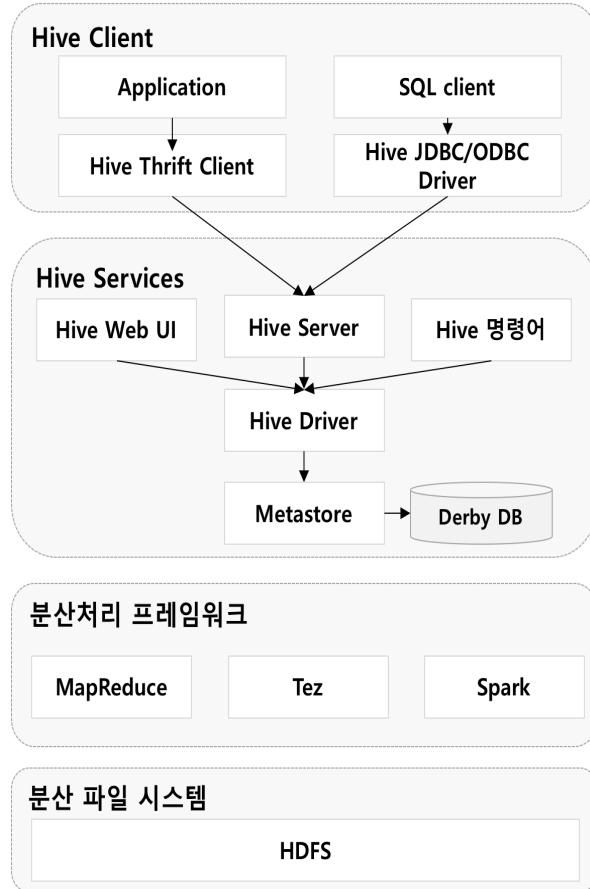
구분	Hive	Presto	Apache Spark
데이터 집계	지원	지원	지원
처리 성능	높음	높음	높음
ETL 지원	지원	일부 지원	지원
대량 배치 처리	가능	부적합	가능
빠른 검색	검색 속도 느림.	가능	일부 가능
BI 연동	가능	가능	별도 설정 필요
SQL 지원	지원	지원	지원

③ Apache Hive의 이해

Apache Hive는 하둡에서 동작하는 데이터 웨어하우스(Data Warehouse) 인프라 구조로서 데이터 요약, 질의 및 분석 기능을 제공한다. 초기에는 페이스북에서 개발되었지만 넷플릭스 등과 같은 회사에서 사용되고 있으며 개발되고 있다. Hadoop HDFS 또는 HBase와 같은 데이터 저장 시스템에 저장되어 있는 대용량 데이터 집합들을 분산 방식으로 검색 및 분석할 수 있다. HiveQL이라고 불리는 SQL 같은 언어를 제공하며 맵 리듀스의 모든 기능을 지원한다. 쿼리를 빠르게 하기 위해 비트맵 인덱스를 포함하여 인덱스 기능을 제공한다.

1. 아키텍처 구성도

Apache Hive는 Meta Store라는 저장소에 데이터에 대한 정보를 관리하고, HiveQL을 입력 받아서 MapReduce 또는 Tez 프로그램으로 변환하여 최적화된 분산 처리를 하는 구조이다. Apache Hive version 3에서는 쿼리 엔진을 Tez로 교체하면서, 더 이상 MapReduce를 지원하지 않는다.



[그림 3-1] Apache Hive 3 아키텍처 구성

2 Apache Hive 핵심 구성 요소

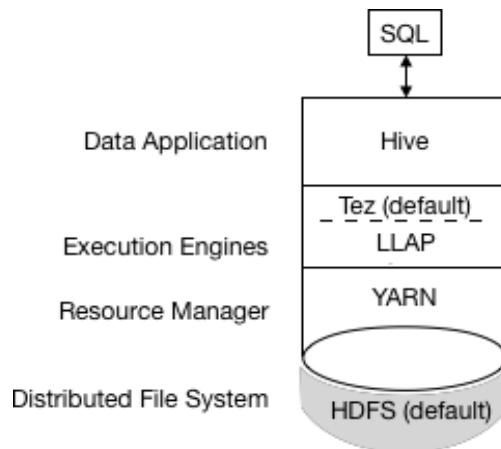
<표 3-3> Apache Hive 핵심 구성 요소

구성 요소	설명
Hive Client	Hive는 다양한 개발 언어(java, c++, python 등)으로 개발된 애플리케이션이 JDBC 또는 ODBC 드라이버로 hive에 접속할 수 있도록 지원한다.
Hive Services	SQL 기반의 HiveQL을 hive에서 실행할 수 있도록 하는 서비스로 세 가지 방식으로 실행이 가능하다.
Hive Driver	Hive Web UI, Hive 명령어에서 보내는 SQL을 Hive Driver에서 Metastore의 정보를 기반으로 분산 처리 프레임워크로 전달하여 실행한다.
Metastore	다양한 Hive 요청을 수신하여, SQL 쿼리 구문을 파싱하고, 최적화된 실행 계획을 수립하여 분산 처리 프레임워크로 전달한다.
분산 처리 프레임워크	Hive의 모든 메타정보를 저장하는 중앙 저장소로 테이블 구조, 칼럼명, 칼럼 타입 등의 정보를 관리한다.
분산 파일 시스템	Hive의 실행 계획에 따라 실제 분산 처리를 수행하는 엔진으로, MapReduce, Tez, Spark를 지원한다. Hive 3.0부터는 기본으로 Tez를 활용한다.
분산 파일 시스템	Hive에서 SQL을 이용하여 탐색할 데이터를 보관하고 있는 분산 파일 저장소이다.

- JDBC(Java DataBase Connectivity)란 Java 프로그램에서 DB에 접속할 수 있도록 하는 프로그래밍 API
- ODBC(Open DataBase Connectivity)는 MS가 만든, 데이터베이스에 접근하기 위한 표준 규격

3. Apache Hive SQL 실행 절차(Hadoop 환경)

(1) Hive SQL 실행 흐름



출처: Hortonworks 공식
[홈페이지\(<https://docs.hortonworks.com>\)](https://docs.hortonworks.com). 2019.
 08. 09. 스크린샷.
 [그림 3-2] Apache Hive 3 아키텍처 구성

(2) Hive에서 입력받은 SQL 실행 시 처리 절차

사용자가 작성한 SQL 문장이 분산 환경에서 실행되는 과정은 <표 3-4>와 같다.

<표 3-4> Esper vs Drools 특징 비교

순서	실행 내용
1	사용자가 입력한 SQL 구문을 Hive에서 컴파일한다.(Tez 엔진 전달)
2	Tez 엔진은 전달받은 컴파일된 SQL을 실행한다.(실행 계획 최적화 진행)
3	Tez가 실행 요청한 작업에 필요한 자원을 YARN을 통해 할당하고, 할당된 자원에서 Tez Job을 실행한다.
4	작성된 SQL 내용에 따라 HDFS에 데이터를 업데이트하거나 Hive Metastore 정보를 수정한다.
5	분산 처리 된 최종 작업 결과를 사용자에게 리턴한다.

④ MapReduce vs Tez vs Apache Spark 기술 특징 비교

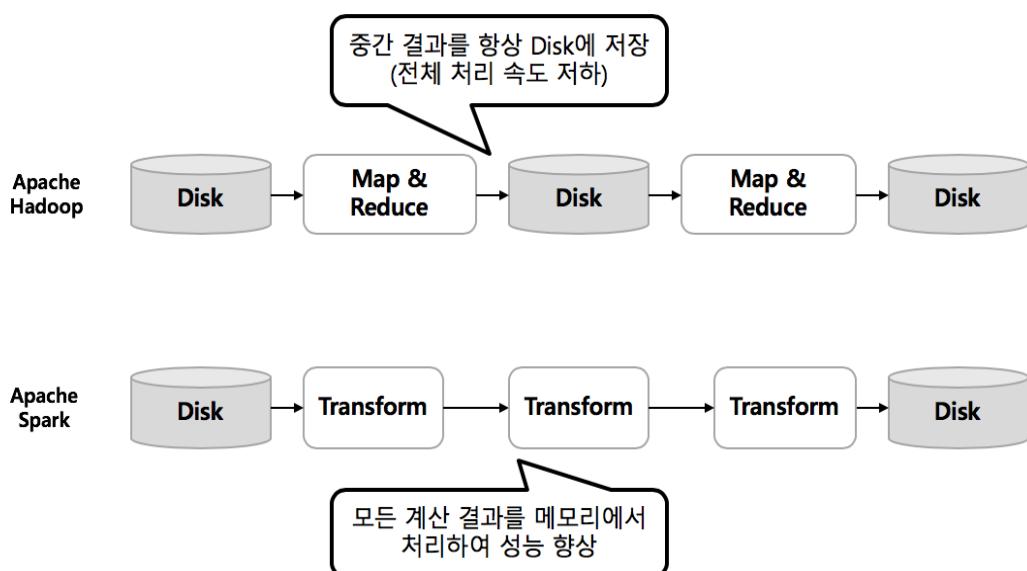
MapReduce 기반의 분산 처리 방식은 HDFS에 저장된 데이터를 읽어 들여, 각 Map, Reduce 단계별로 발생하는 임시 중간 결과를 다시 HDFS에 저장하고, 읽어 들이는 방식으로 대량의 데이터를 안정적으로 처리하는 방식이다. 하지만 이러한 방식은 DISK I/O가 너

무 많이 발생하여 처리 성능이 전반적으로 저하되는 결과를 유발하게 된다. Apache Spark는 중간 단계 결과물을 Memory에 저장하여 데이터 처리 속도를 향상시키는 메모리 기반 분산 처리 기술이다.

최근 머신러닝과 같이 동일한 데이터를 반복적으로 처리하는 분석 요구 사항을 만족하기 위해서는 Apache Spark와 같은 기술을 이용하여 대량의 데이터를 효율적으로 처리할 수 있어야 한다.

1. 동작 방식 비교

Apache Spark은 처리할 데이터를 메모리 기반으로 처리하므로 각 처리 단계별 데이터 이동이 빠르고, 처리 성능이 높다.



[그림 3-3] MapReduce vs Spark 동작 방식 비교

2. 핵심 특징 비교

주요 영역별로 핵심적인 특징의 차이를 확인한다.

<표 3-5> MapReduce vs Tez vs Apache Spark 특징 비교

구분	MapReduce	Tez	Apache Spark
데이터 처리	배치 처리 전용	배치 처리 전용	배치 처리+실시간 처리
처리 성능	Disk I/O 사용으로 Apache Spark보다 느림.	실행 계획 최적화로 성능 향상(MapReduce 대비)	Memory 기반으로 빠름.
기능 분류	데이터 처리 엔진	데이터 처리 엔진	데이터 분석 엔진
머신러닝 지원	별도 머신러닝 라이브러리 활용	별도 머신러닝 라이브러리 활용	자체 머신러닝 API 제공
사용성	다소 어려움.	다소 어려움.	다수 API로 쉬운 편
개발 언어	Java, C, Python, Ruby	Java, C, Python, Ruby	Java, Python, Scala, R

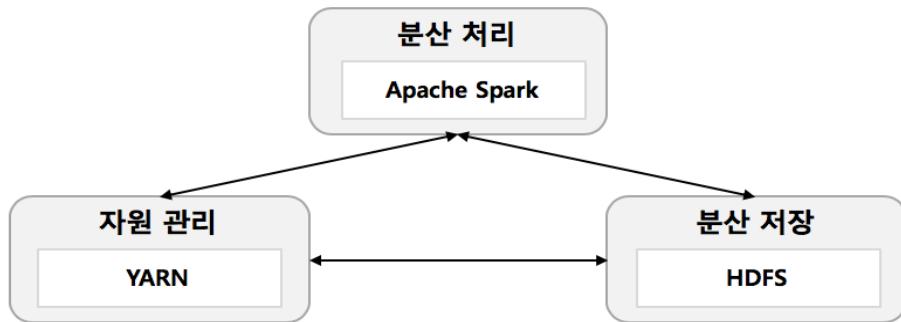
구분	MapReduce	Tez	Apache Spark
	등 다수	등 다수	
SQL 지원	불가	불가	지원

⑤ Apache Spark의 이해

빅데이터를 처리하는 용도로 개발된 오픈 소스 분산 병렬 처리 플랫폼으로 메모리 기반의 처리 엔진을 이용하여 대용량 배치 처리와 실시간 대용량 처리까지 지원하는 특징과 분산 환경에서 머신러닝 및 SQL 분석을 위한 기능을 제공한다.

1. Apache Spark와 아파치 하둡(Apache Hadoop)의 관계

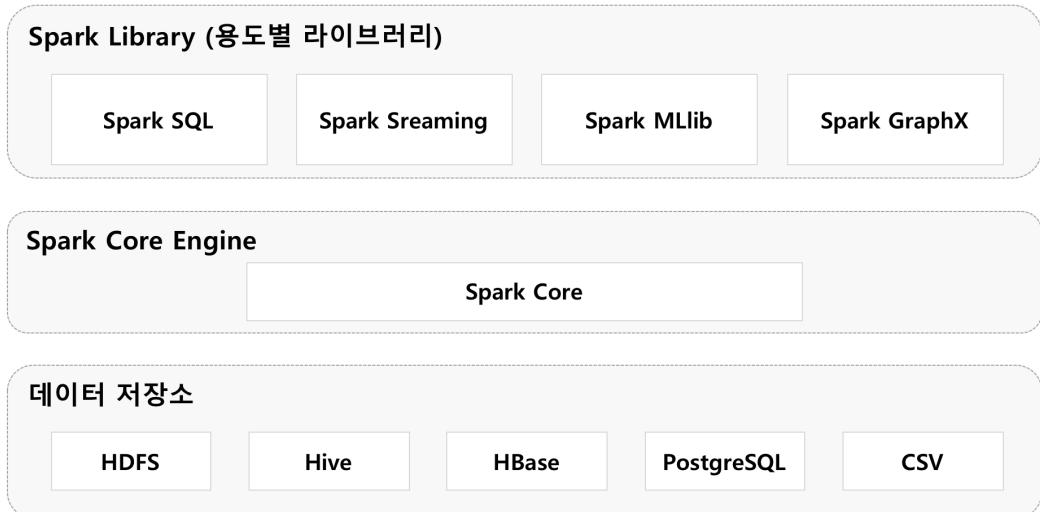
Spark는 Hadoop의 처리 엔진으로 동작하며, Hadoop에서 제공하는 동적 자원을 할당받아 분산 데이터를 처리하도록 구성한다.



[그림 3-4] 아파치 하둡(Apache Hadoop)과 아파시 스파크(Apache Spark)의 관계도

2. Apache Spark의 컴파넌트 구성도

Spark는 분산 처리 엔진인 spark core를 기반으로 실시간 처리를 위한 Spark Streaming, SQL 엔진인 Spark SQL, 분산 머신러닝 학습을 위한 Spark MLlib, 그래프 관계 분석을 위한 Spark GraphX 라이브러리를 지원한다. 하고자 하는 업무 목적에 따라 라이브러리를 선택하여 빅데이터를 처리 및 분석할 수 있다.



[그림 3-5] Spark의 컴포넌트 구성

3. Apache Spark 처리를 위한 자료 구조(RDD)

Spark는 분산된 데이터를 효율적으로 처리하기 위한 자체 자료 구조인 RDD(Resilient Distributed Dataset)를 제공한다. Spark 프로그래밍에서는 데이터를 읽어 오는 순간부터 RDD로 변환하고, 이 변환된 RDD를 다시 가공하여 새로운 RDD를 생성하는 방식으로 처리를 반복하여 최종 결과를 생성한다. 또 RDD는 다른 자료 구조와 달리 한 번 생성된 내부 데이터값이 변경되지 않는 불변성을 가지며, 실제 최종 데이터를 생성하는 시점까지 계산을 지연하는 특징을 가진다.

(1) RDD 구조 예시

HDFS에서 파일의 목록을 저장한 파일을 읽어 온 경우, Spark에서는 RDD 구조로 변환하여 저장하게 된다. 다음 예시를 보면 먼저 데이터를 분산해서 처리할 수 있도록 여러 개의 partition(예시에는 두 개)으로 분산하고, 각 partition에 실제 데이터를 할당 한다. 그리고 이후 모든 처리는 RDD를 기반으로 분산 처리 된다.



[그림 3-6] RDD 구성 예시

(2) RDD 처리를 위한 함수 구성

RDD에서는 변환(transformation)과 실행(Action) 두 가지의 처리를 할 수 있다.

<표 3-6> RDD 함수 유형

구분	변환(Transformation)	실행(Action)
동작 방식	RDD를 가공하여 새로운 RDD를 생성하는 함수	기존 RDD의 데이터를 기공하지 않고, 데이터를 기반으로 원하는 결과를 출력하는 함수
지원 함수	filter(데이터 필터링) map(데이터 변경 처리) join(같은 키를 가지는 데이터를 합침.)	saveAsTextFile(RDD 내용을 파일로 저장) count(RDD의 데이터 수를 계산)

4. Apache Spark 클러스터 구성도

Spark는 여러 대의 서버로 구성된 클러스터 환경에서 분산 처리하는 플랫폼이다. 이러한 분산된 서버 자원을 관리하기 위해서는 자원 관리 기능이 필요한데, Spark에서는 다양한 자원 관리 시스템을 지원한다.

(1) 클러스터 자원 관리 시스템

<표 3-7> 자원 관리 시스템

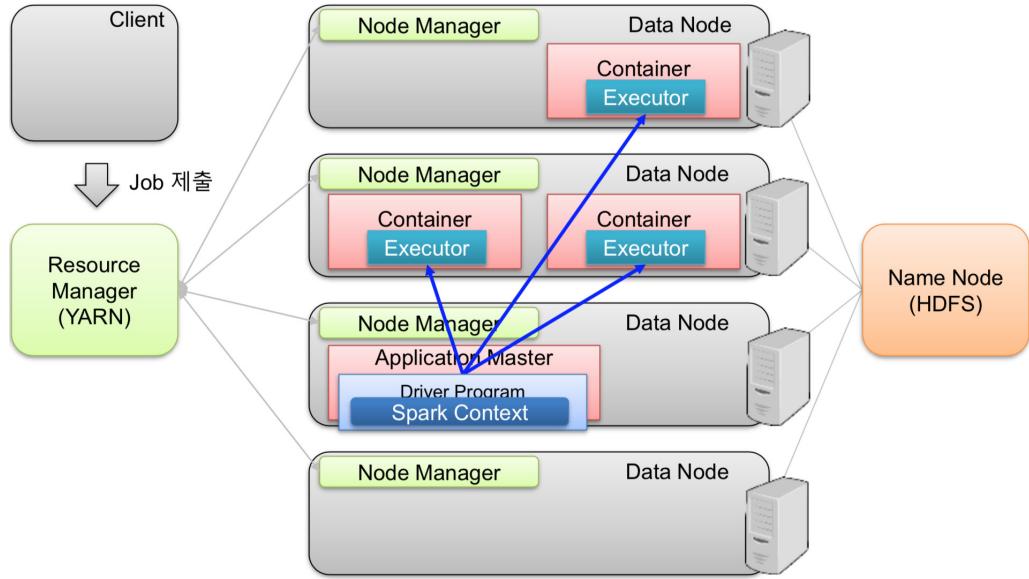
자원관리	설명
YARN	Hadoop의 분산 처리를 위한 자원 관리 시스템으로 HDFS를 사용할 경우 데이터가 존재하는 곳에 프로그램을 실행시키는 지역성을 고려하여 처리 성능이 높은 장점이 있다.
Mesos	YARN과 동일한 목적의 클러스터 관리 시스템으로 분산 처리에 할당되는 자원(cpu, memory 등)을 동적으로 조정 가능한 장점이 있다.
Standalone	Spark에서 기본으로 제공되는 클러스터 관리 시스템으로 별도의 클러스터 관리 시스템 없이 쉽게 이용 가능하다.

(2) YARN을 이용한 Spark 애플리케이션 실행 구조 이해

클러스터 관리 시스템을 이용하면 각 서버는 마스터 또는 워커(Data) 노드로 동작하게 된다. 마스터 노드는 클러스터 내의 계산 리소스를 관리하고, 워커 노드는 cpu, memory를 애플리케이션에 제공하고 실제 처리를 담당한다.

(가) 애플리케이션 실행 구조도

다음 예시는 사용자가 개발한 spark 애플리케이션을 spark-submit으로 실행한 경우, 클러스터에 배포되는 구성을 설명하고 있다.



[그림 3-7] Spark의 애플리케이션 실행 구조

(나) Spark 애플리케이션 실행 절차

<표 3-8> Spark 배포 및 실행 순서

순서 실행 내용	
1	사용자가 개발한 spark 애플리케이션을 클라이언트가 클러스터로 배포한다.(spark-submit 명령어 활용)
2	클라이언트는 애플리케이션을 배포함과 동시에 애플리케이션이 워커노드에서 실행에 필요한 자원(memory, cpu core 등)을 지정한다.
3	애플리케이션 실행에 필요한 자원이 확보되면 애플리케이션의 시작점인 드라이버 프로그램이 구동된다.
4	실행할 때 cluster-mode를 이용하여 드라이버 프로그램의 실행 위치를 지정 할 수 있다.(위 예시는 클러스터에서 드라이버 프로그램이 구동된다.)
4	Executor에서 실제 RDD를 처리하는 분산 작업이 task라는 단위로 실행된다.

(다) Spark 애플리케이션 실행 프로세스 구성 요소

<표 3-9> spark 애플리케이션 구성 요소

구성 요소	설명
Driver Program	spark 애플리케이션의 동작을 실행 및 제어하는 로직을 처리하는 프로그램이다. RDD의 변환에서 실제 결과 출력까지의 흐름을 기술하고 있으나, 실제 작업은 Driver Program에서 실행되지 않고, Executor에서 실행된다.
Executor	Driver Program에서 정의한 RDD 처리 로직을 실제로 수행하며, 자원 관리 시스템에 의해서 필요한 만큼의 자원을 할당받아서 분산 처리를 수행한다.

(라) Spark 실행 명령어인 spark-submit 실행 옵션 이해

<표 3-10> spark-submit 실행 옵션 설명

옵션	상세 설명
--master	접속할 Cluster Manager의 정보를 입력한다.(다음과 같은 방식) - spark://host:port - mesos://host:port
--deploy-mode	[Client Mode] Driver가 spark-submit을 실행한 서버에서 실행된다. 기본은 client 모드로 설정된다.
--class	[Cluster Mode] Cluster의 작업 머신들 중에서 Driver가 결정된다.
--name	Java나 Scala 프로그램을 실행할 때 “main” 이 들어 있는 클래스를 지정한다.
--jars	실행할 Apache Spark jar 파일의 경로를 지정한다.(관련된 모든 라이브러리를 포함하도록 jar 파일을 생성)
--num-executors	분산하여 실행할 executor의 개수를 지정한다.
--executor-memory	executor가 쓸 메모리를 사이즈를 지정한다. (“512m” , “15g” 등)
--executor-cores	하나의 executor에 할당할 cpu core 개수를 지정한다.
--driver-memory	Driver가 쓸 메모리를 사이즈를 지정한다. (“512m” , “15g” 등)

5. Apache Spark 개발 환경 이해(spark-shell vs spark application)

Spark 애플리케이션 실행하는 방법은 크게 spark-shell에서 함수를 직접 호출하여 데이터를 처리하는 방법과 프로그래밍을 통해 컴파일된 실행 파일(jar 파일)을 생성하여 실행하는 방식이 있다.

<표 3-11> spark-shell vs spark application

구분	spark-shell	spark 애플리케이션
동작 방식	console에서 명령어 실행 방식	java/scala 같은 개발 언어로 코딩한 후, compile을 통해 생성된 실행 파일(jar)을 실행
결과 출력	각 명령어를 실행할 때 결과 출력 대화식(상호 작용, Interactive)	코딩이 완료된 전체 애플리케이션의 결과 출력
지원 언어	Python, Scala	python, scala, java,
장점	명령어 실행으로 빠른 결과 확인 간단하고 실행이 쉬움.	복잡한 비즈니스 로직이나, 다양한 라이브러리를 활용한 계산에 적합
단점	복잡한 로직 구현이 불편함.	단순한 데이터 조회 및 검증 불편
실행 방법	spark-shell 명령어 실행	spark-submit 명령어 실행

6. Apache Spark 핵심 기능

<표 3-12> Apache spark 핵심 기능

기능	상세 설명
쉬운 구현	개발자들이 쉽게 구현할 수 있도록 추상화된 레벨의 API를 제공하여, 빠르게 시스템을 구축 가능
가용성	여러 대의 서버에서 분산 처리할 수 있으며, 노드 장애 시 다른 노드에서 자동으로 데이터를 처리하여 서비스 가용성이 높음.
Fault Tolerance	Exactly-once 수준의 메시지 전달을 보장하여, 데이터의 유실 또는 중복 없이 데이터 전달이 가능함.
Spark 패키지 연동	Spark 프로젝트에서 제공하는 Machine Learning, SQL, GraphX 등의 라이브러리를 쉽게 연동하여 활용 가능.
빠른 성능	다양한 비즈니스 요건을 빠르고 쉽게 대응 가능
빠른 성능	자체 분산 메모리 자료 구조인 RDD를 이용하여 대량의 데이터를 분산하여 빠르게 처리 가능

⑥ Apache Spark 애플리케이션 빌드 관리 툴, sbt 이해

Spark application은 소스 코드(java, scala)를 컴파일하고 최종 실행 파일인 jar파일로 패키징 해야 한다. sbt는 이러한 컴파일에서 라이브러리 의존 관계를 관리하며 실제 패키징된 실행 파일을 생성해 주는 통합 관리 도구이다.

1. sbt 설치 및 테스트 방법

sbt는 설치하는 서버의 운영 체제(OS, Operating System)에 따라 설치 명령어가 다르므로, 현재 서버의 운영 체제를 먼저 확인한다. 본 교재에서는 Linux(CentOS 7.0)을 기준으로 설치 방법을 가이드

한다.(<https://www.scala-sbt.org/release/docs/Installing-sbt-on-Linux.html> 참고)

```
[root@]# curl https://bintray.com/sbt/rpm/rpm | sudo tee /etc/yum.repos.d/bintray-sbt-rpm.repo
% Total    % Received % Xferd  Average Speed   Time     Time      Current
          Dload  Upload Total   Spent    Left Speed
100  160    0  160    0     0   143      0 --:--:--  0:00:01 --:--:--  143
#bintray-sbt-rpm - packages by from Bintray
[bintray-sbt-rpm]
name=bintray-sbt-rpm
baseurl=https://sbt.bintray.com/rpm
gpgcheck=0
repo_gpgcheck=0
enabled=1
```

[그림 3-8] sbt 설치 리파지토리 다운로드(설치 정보가 저장된 파일)

```
[root@]# sudo yum install -y sbt
```

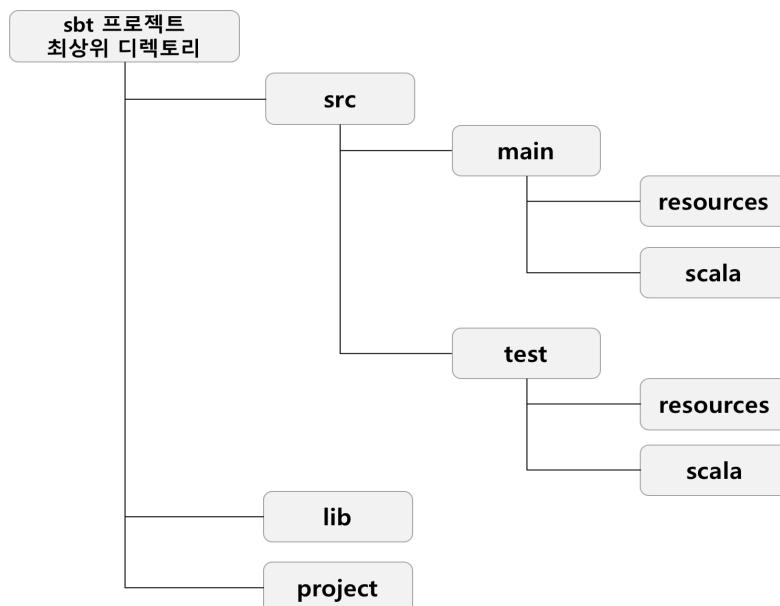
[그림 3-9] linux 설치 명령어로 sbt 설치하기

```
[dpcore@b ~]$ sbt about
[info] Loading project definition from /home/dpcore/test/sbt-build/project
[info] Loading settings for project sbt-build from build.sbt ...
[info] Set current project to sbt-build (in build file:/home/dpcore/test/sbt-build/)
[info] This is sbt 1.2.8
```

[그림 3-10] sbt version 확인하기(현재 버전은 1.2.8, scala 버전은 2.12.7)

2. sbt 프로젝트의 디렉터리 구조

sbt는 디렉터리 구조에 따라서 소스 코드의 위치를 자동으로 인식한다. 즉, 어떤 디렉터리 명(java, scala)가 존재하는지에 따라서 소스 코드를 빌드하는 방법이 자동으로 설정된다. sbt에서 사용하는 디렉터리 구조와 각 디렉터리의 역할을 살펴본다.



[그림 3-11] sbt 프로젝트 디렉터리 구조

각 디렉터리의 역할은 다음 표와 같다.

<표 3-13> sbt 프로젝트 디렉터리별 상세 역할

디렉터리	상세 역할
src/main/scala	scala로 작성한 소스 코드를 저장하는 디렉터리
src/main/resources	최종 생성된 실행 파일이 참조하는 파일들을 저장하는 디렉터리
src/test/scala	scala로 작성된 main/scala의 함수를 테스트하는 소스 코드를 저장하는 디렉터리
src/test/resources	테스트용 실행 파일이 참조하는 파일들을 저장하는 디렉터리
lib	생성된 spark 애플리케이션이 참조(의존)하는 라이브러리를 저장하는 디렉터리.(sbt에서 의존 관계를 설정하면 별도로 이 디렉터리에 저장하지 않는다. 여기는 sbt에 의존 관계를 지정하지 않은 라이브러리만 별도 저장)
project	sbt 설정과 관련된 추가 설정 정보를 저장하는 디렉터리

디렉터리	상세 역할
build.sbt	sbt 빌드에 필요한 필수 설정값을 저장하는 파일
target	sbt build 또는 package 실행 시 최종 실행 파일을 저장하는 디렉터리

3. sbt 프로젝트 설정방법

sbt로 spark 애플리케이션을 빌드하려면 build.sbt 파일에 필요한 빌드 정보를 작성하고 저장해야 한다. 또 최종 실행 파일을 jar로 생성하기 위해서는 plugin.sbt라는 파일에 관련 내용을 작성하고 저장한다.

(1) build.sbt 설정 방법

build.sbt는 sbt 프로젝트 최상위 디렉터리 아래에 생성하고, 필요한 설정값을 Key/Value 방식으로 프로젝트 빌드에 필요한 정보를 작성한다.

<표 3-14> sbt 빌드 관련 주요 설정 정보

설정 키	내용
name	생성할 프로젝트 명칭
version	생성할 프로젝트의 버전
scalaVersion	현재 scala 컴파일러 버전(sbt about 명령어로 확인 가능)
libraryDependencies	현재 프로젝트 소스 코드에서 사용(의존)하는 라이브러리 목록 정의 사용하는 라이브러리가 테스트 단계에서만 필요한지, 실제 배포될 때 포함되어야 할지 지정
assemblyOption in assembly	sbt-assembly 플러그인 옵션으로 최종 jar 파일을 생성할 때 scala 라이브러리도 필요하게 된다. 하지만 이 라이브러리가 배포하려는 서버에 존재하는 경우, 이를 포함할지 여부를 설정(includeScala = false)

(2) plugins.sbt 설정 방법

sbt-assembly 플러그인을 활용하려면 plugins 디렉터리 아래에 plugins.sbt를 생성하고 다음 내용을 추가한다.

```
addSbtPlugin("com.eed3si9n" % "sbt-assembly" % "0.14.0")
```

4. sbt 프로젝트 생성 및 실행

sbt 프로젝트를 관리할 디렉터리(simple-sbt)를 생성하고, scala 소스 코드를 저장할 디렉터리도 함께 생성한다. 생성된 scala 디렉터리에 샘플 scala 코드(Hello.scala)를 작성한다.

```
[dpcore@      test]$ mkdir simple-sbt/
[dpcore@      test]$ cd simple-sbt/
[dpcore@      simple-sbt]$ mkdir -p src/main/scala/
[dpcore@      simple-sbt]$ vi src/main/scala/Hello.scala
```

[그림 3-12] sbt 프로젝트 디렉터리 생성

Hello.scala에서 “Hello, World”라는 문자열을 출력하는 메인 함수를 작성하고 파일을 저장한다.

```
package com.ncs.sbt

object Main extends App {
    println("Hello, world")
}
```

[그림 3-13] Hello.scala 내용

sbt 프로젝트 빌드를 위한 설정 정보를 build.sbt 파일에 작성하고, 프로젝트 최상위 디렉터리 아래에 저장한다.

```
[dpcore@      simple-sbt]$ vi build.sbt

name := "my-simple-sbt"
version := "0.1"
scalaVersion := "2.12.7"
```

[그림 3-14] build.sbt 설정 파일 내용

작성한 소스 코드(Hello.scala)를 빌드하기 위한 명령어(sbt compile)를 실행한다.

```
[dpcore@      simple-sbt]$ sbt compile
[info] Loading project definition from /home/dpcore/test/simple-sbt/project
[info] Loading settings for project simple-sbt from build.sbt ...
[info] Set current project to my-simple-sbt (in build file:/home/dpcore/test/simple-sbt)
[info] Executing in batch mode. For better performance use sbt's shell
[info] Updating ...
[info] Done updating.
[info] Compiling 1 Scala source to /home/dpcore/test/simple-sbt/target/scala-2.12/classes
[info] Done compiling.
[success] Total time: 6 s, completed Aug 26, 2019 3:40:41 PM
```

[그림 3-15] sbt compile 실행 화면

compile 완료된 실행 파일을 실행하는 명령어(sbt run)을 실행하여, 작성한 소스 코드의 결과 문자(Hello, world)가 출력되는지 확인한다. 정상 출력되면 sbt 프로젝트가 제대로 설정된 것을 확인 가능하다.

```
[dpcore@l      simple-sbt]$ sbt run
[info] Loading project definition from /home/dpcore/test/simple-sbt/project
[info] Loading settings for project simple-sbt from build.sbt ...
[info] Set current project to my-simple-sbt (in build file:/home/dpcore/test/simple-sbt/)
[info] Packaging /home/dpcore/test/simple-sbt/target/scala-2.12/my-simple-sbt_2.12-0.1.jar
[info] Done packaging.
[info] Running com.ncs.sbt.Main
Hello, world
[success] total time: 1 s, completed Aug 26, 2019 3:25:11 PM
```

출력 결과

[그림 3-16] sbt run 실행 화면

수행 내용 / 분산 처리 수행 모듈 개발하기

재료 · 자료

- 빅데이터 처리 시스템 설계서
- 분산 처리 수행 모듈을 개발할 서버 정보
- 분산 처리 수행 모듈을 개발할 개발 환경

기기(장비 · 공구)

- 리눅스 서버
- 컴퓨터, 프린터, 인터넷
- 사무 자동화 소프트웨어

안전 · 유의 사항

- 해당 사항 없음.

수행 순서

① 빅데이터 분산 처리를 위한 오픈 소스 소프트웨어를 선정한다.

대상 데이터가 비구조화된 원본 데이터인 경우, 이를 검색 및 분석하기 위해 구조화 데이터로 변환 및 칼럼 스토리지로 저장하는 분산 처리 수행 모듈 개발에 최적화된 Apache Spark를 선정하고, 구조화된 데이터를 빠르고 정교한 분석을 위하여, SQL-on-hadoop 기술인 Apache Hive를 선정하여 수행 모듈을 개발한다.

② Apache Spark 애플리케이션을 이용한 분산 처리 수행 모듈을 개발한다.

대표적인 대용량 데이터 분산 처리 기술인 Apache Spark를 이용하여 데이터를 처리하는 프로그램을 작성하고 분산 환경에서 실행한다.

1. 분산 처리 수행 모듈의 설계 내용을 확인한다.

개발할 분산 처리 수행 모듈을 개발하기 위하여 설계된 문서를 참고한다. 입력 데이터와 처리를 통해 생성할 데이터 등의 입출력 내용을 확인한다. 본 교재에서는 대량의 텍스트 파일에서 특정 문자열이 나타난 횟수를 집계하는 예시로 개발하도록 한다.

2. 분산 처리 수행 모듈 개발을 위한 데이터를 준비한다.

(1) Apache Hive를 실행하기 위해 ssh 연결로 서버에 접속한다.

VirtualBox로 Hortonworks의 HDP Sandbox를 실행한 상태에서 해당 서버로 접속하기

위하여 ssh로 접속한다. HDP Sandbox로 설치된 서버는 초기 root 계정의 비밀번호가 hadoop이다. 처음 접속하면 root 비밀번호를 변경해야 접속이 가능하다.

```
>ssh root@localhost -p 2222  
Warning: Permanently added '[localhost]:2222' (RSA) to the list of known hosts.  
root@localhost's password:  
Last login: Tue Aug 13 20:27:10 2019 from 10.0.2.2  
[root@sandbox ~]#
```

[그림 3-17] ssh 연결 화면(port 2222)

수행 tip

- root 비밀번호는 분실하게 되면, 서버에 접속할 방법이 없으므로, 반드시 다른 저장 매체에 기록하여 분실하지 않도록 대안을 마련한다.

(2) Apache Spark에서 읽어 올 데이터를 HDFS에 업로드한다.

본 교재 실습에서는 HDP를 설치할 때 제공되는 Apache SPark의 “README.md” 파일을 이용하여 데이터를 읽고, 처리하는 수행 모듈을 작성한다.

```
[dpcore@      simple-spark]$ hadoop fs -put /usr/hdp/3.1.0.0-78/spark2/README.md /  
[dpcore@      simple-spark]$ hadoop fs -ls /  
Found 17 items  
-rw-r--r--  3 dpcore hdfs  2019-06-17 17:49 /LICENSE_hdfs  
-rw-r--r--  3 dpcore hdfs  2019-06-17 17:26 /LICENSE_hdfs  
-rw-r--r--  3 dpcore hdfs  3809 2019-08-26 17:33 /README.md
```

[그림 3-18] hdfs에 파일을 업로드하고 확인하는 화면

```
[dpcore@      simple-spark]$ hadoop fs -cat /README.md  
# Apache Spark  
  
Spark is a fast and general cluster computing system for Big Data. It provides  
high-level APIs in Scala, Java, Python, and R, and an optimized engine that  
supports general computation graphs for data analysis. It also supports a  
rich set of higher-level tools including Spark SQL for SQL and DataFrames,  
MLlib for machine learning, GraphX for graph processing,  
and Spark Streaming for stream processing.
```

[그림 3-19] hdfs에 업로드된 파일의 내용 확인하는 화면

3. 분산 처리 수행 모듈인 Apache Spark 애플리케이션을 개발한다.

(1) 애플리케이션 개발을 위한 코드를 작성한다.

sbt기반의 프로젝트를 개발하기 위하여, 프로젝트 디렉터리를 생성하고, 소스 코드 디렉터리에 코드를 작성할 파일(SimpleApp.scala)을 생성한다.

```
[dpcore@      test]$ mkdir -p simple-spark/src/main/scala/  
[dpcore@      test]$ vi simple-spark/src/main/scala/SimpleApp.scala
```

[그림 3-20] 프로젝트 디렉터리 생성 및 scala 파일 생성

분산 처리를 수행할 코드를 다음과 같이 작성한다.

```

import org.apache.spark.sql.SparkSession

object SimpleApp {
    def main(args: Array[String]) {
        val logFile = "/README.md"
        1 val spark = SparkSession.builder.appName("Simple Application").getOrCreate()
        2 val logData = spark.read.textFile(logFile).cache()
        3 val numAs = logData.filter(line => line.contains("a")).count()
        4 val numBs = logData.filter(line => line.contains("b")).count()
        5 println(s"Lines with a: $numAs, Lines with b: $numBs")
        6 spark.stop()
    }
}

```

HDFS 경로

[그림 3-21] spark 프로그램 소스 코드

위 코드의 주요 내용은 다음과 같다.

<표 3-15> SimpleApp.scala 코드 실행 순서

순서	내용
①	SparkSession 객체를 이용하여 spark 세션을 생성하고, 이력에 표시될 애플리케이션 명칭을 지정한다.
②	생성된 spark 세션으로 HDFS에 저장된 “README.md” 파일을 읽어온다. 이때 Spark의 내부 자료 구조인 RDD로 변환되고, 이후부터 RDD를 처리하는 spark 함수를 실행할 수 있다. 또 대용량 데이터인 경우 hdfs의 block 크기만큼 분할하여 분산 처리를 하게 되어, 성능을 향상시킨다.
③	분산되어 읽어온 RDD의 데이터를 각 서버의 executor에서 필터링한다. 여기서는 “a” 문자열이 포함된 횟수를 계산한다.
④	분산되어 읽어온 RDD의 데이터를 각 서버의 executor에서 필터링한다. 여기서는 “b” 문자열이 포함된 횟수를 계산한다.
⑤	최종 action 함수가 실행되면서 분산된 executor에서 실행된 결과값을 하나로 조합하여 출력하게 된다. 즉, “README.md”에 포함된 a, b 문자열의 총 횟수를 출력한다.
⑥	사용한 spark 세션을 종료한다.

(2) sbt로 수행 모듈의 빌드하기 위한 설정을 한다.

sbt의 build.sbt에 프로그램 빌드에 필요한 기본 정보(프로젝트 명, scala 버전, 사용할 spark 라이브러리 및 버전 등)를 작성한다.

```

[dpcore@      test]$ cd simple-spark/
[dpcore@      test]$ vi build.sbt
name := "Simple Project"
version := "1.0"
scalaVersion := "2.11.8"
libraryDependencies += "org.apache.spark" %% "spark-sql" % "2.3.2"

```

사용하는 spark 버전 확인

[그림 3-22] sbt 프로젝트 환경 설정

현재 구성된 sbt 프로젝트의 구성을 확인한다.

```
[dpcore@      simple-spark]$ find .
.
./src
./src/main
./src/main/scala
./src/main/scala/SimpleApp.scala
./build.sbt
```

[그림 3-23] sbt 프로젝트 디렉터리 및 파일 확인

(3) sbt로 수행 모듈을 빌드 및 패키징(jar 파일 생성)한다.

sbt package 명령어를 사용하면 compile 명령어를 자동으로 실행하여 최종 실행 파일을 생성한다.

```
[dpcore@      test]$ sbt package
[info] Updated file /home/dpcore/test/project/build.properties: set sbt.version
[info] Loading project definition from /home/dpcore/test/project
[info] Updating ProjectRef(uri("file:/home/dpcore/test/project/"), "test-build")
[info] Packaging /home/dpcore/test/target/scala-2.12/simple-project_2.12-1.0.jar
[info] Done packaging.
[success] Total time: 342 s, completed Aug 26, 2019 5:13:47 PM
[dpcore@      test]$ ls target/scala-2.12/
resolution-cache  simple-project_2.12-1.0.jar
```

[그림 3-24] sbt package 명령어로 실행 파일 생성 및 확인 명령어

4. 개발한 분산 처리 수행 모듈인 Spark 애플리케이션을 실행하고 이력을 확인한다.

(1) spark-submit으로 spark 애플리케이션을 실행한다.

spark-submit 명령어로 개발한 분산 처리 수행 모듈을 배포 및 실행한다. 다음 예시에서는 local 서버에서 cpu 2 코어만 이용하여 실행하도록 하였다. 실제 분산 환경에서는 spark master에서 spark 애플리케이션을 다수의 서버에서 실행하도록 자원을 할당해야 한다. 자세한 spark-submit 명령어는 필요 지식의 “spark-submit 명령어 이해”를 참고한다.

```
[dpcore@bdavm09 simple-spark]$ spark-submit \
> --class "SimpleApp" \
> --master local[2] \
> target/scala-2.11/simple-project_2.11-1.0.jar
19/08/26 18:10:47 INFO SparkContext: Running Spark version 2.3.2.3.1.0.0-7
19/08/26 18:10:47 INFO SparkContext: Submitted application: Simple Application
19/08/26 18:10:47 INFO SecurityManager: Changing view acls to: dpcore
19/08/26 18:10:47 INFO SecurityManager: Changing modify acls to: dpcore
19/08/26 18:10:47 INFO SecurityManager: Changing view acls groups to:...
Lines with a: 61, Lines with b: 30
```

결과 출력

[그림 3-25] spark-submit 실행 후 결과 출력 화면

(2) Apache Spark History Web UI에서 실행 결과를 확인한다.

spark 애플리케이션이 실행된 이력을 web에서 조회하기 위하여, localhost:18081 url

을 웹 브라우저에서 접속한다. 수행 모듈이 실행된 결과가 화면에 출력된다.

App ID	App Name	Attempt ID	Started	Completed	Duration
local-1566810388581	Simple Application		2019-08-26 18:06:26	2019-08-26 18:06:43	17 s
application_1564534589507_0058	Spark shell		2019-08-26 17:49:41	2019-08-26 17:54:04	4.4 min

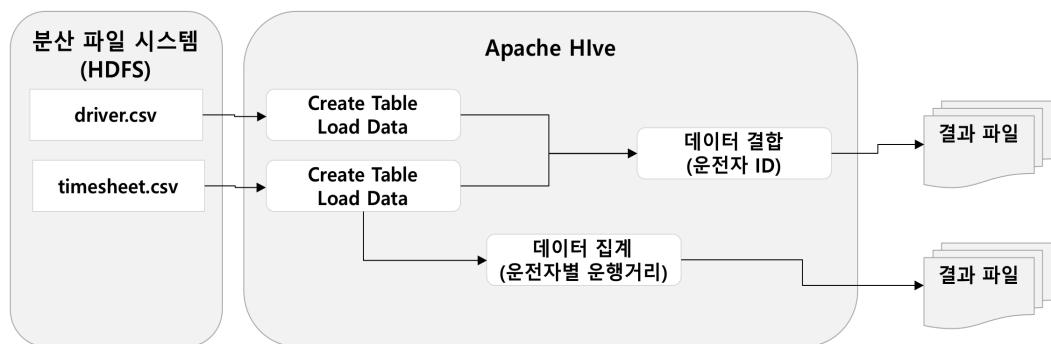
[그림 3-26] spark history server web ui 화면

③ Apache Hive를 이용한 분산 처리 수행 모듈을 개발한다.

대표적인 SQL 방식의 분산 처리 기술인 Apache Hive를 이용하여 분산 처리하는 수행 모듈을 개발하도록 한다.

1. 분산 처리 수행 모듈의 설계 내용을 확인한다.

개발할 분산 처리 수행 모듈을 개발하기 위하여 설계된 문서를 참고한다. 입력 데이터와 처리를 통해 생성할 데이터 등의 입출력 내용을 확인한다. 다음 예시를 기반으로 Hive 수행 모듈을 개발한다.



[그림 3-27] 데이터 처리 흐름도 예시

2. 분산 처리 수행 모듈 개발을 위한 데이터를 준비한다.

(1) Apache Hive를 실행하기 위해 ssh 연결로 서버에 접속한다.

VirtualBox로 Hortonworks의 HDP Sandbox를 실행한 상태에서 해당 서버로 접속하기 위하여 ssh로 접속한다.

```
>ssh root@localhost -p 2222
Warning: Permanently added '[localhost]:2222' (RSA) to the list of known hosts.
root@localhost's password:
Last login: Tue Aug 13 20:27:10 2019 from 10.0.2.2
[root@sandbox ~]#
```

[그림 3-28] ssh 연결 화면(port 2222)

(2) 실습할 데이터를 다운로드한다.

분산 처리 수행 모듈 개발을 위해 필요한 샘플 데이터들을 웹에서 다운로드한다.

```

wget
https://raw.githubusercontent.com/hortonworks/data-tutorials/master/tutorials/hdp/
how-to-process-data-with-apache-hive/assets/driver_data.zip
>wget https://raw.githubusercontent.com/hortonworks/data-tutorials/master/tutorials/hdp/how-to-p
rocess-data-with-apache-hive/assets/driver_data.zip

```

[그림 3-29] 샘플 파일을 다운로드받는 명령어

```

>unzip driver_data.zip
Archive:  driver_data.zip
  creating: driver_data/
  inflating: driver_data/.DS_Store
  creating: __MACOSX/.csv_truck_event_text_partition.csv
  creating: __MACOSX/driver_data/
  inflating: __MACOSX/driver_data/.DS_Store
  inflating: driver_data/drivers.csv
  inflating: __MACOSX/driver_data/_drivers.csv
  inflating: driver_data/timesheet.csv
  inflating: __MACOSX/driver_data/_timesheet.csv
  inflating: driver_data/truck_event_text_partition.csv
  inflating: __MACOSX/driver_data/_truck_event_text_partition.csv
>ls driver_data
drivers.csv          timesheet.csv          truck_event_text_partition.csv

```

[그림 3-30] 다운로드한 샘플 파일의 압축을 해제하고, 디렉터리를 조회하는 명령어

```

>head timesheet.csv
hive.exec.max.dynamic.partitions=2000
driverId,week,hours-logged,miles-logged
10,1,70,3300
10,2,70,3300
10,3,60,2800
10,4,70,3100
10,5,70,3200
10,6,70,3300
10,7,70,3000
10,8,70,3300
10,9,70,3200

```

[그림 3-31] timesheet.csv 파일 내용 확인하는 명령어

```

>head drivers.csv
hive.exec.max.dynamic.partitions=2000
driverId,name,ssn,location,certified,wage-plan
10,George Vetticaden,621011971,244-4532 Nulla Rd.,N,miles
11,Jamie Engesser,262112338,366-4125 Ac Street,N,miles
12,Paul Coddin,198041975,Ap #622-957 Risus. Street,Y,hours
13,Joe Niemiec,139907145,2071 Hendrerit. Ave,Y,hours
14,Adis Cesir,820812209,Ap #810-1228 In St.,Y,hours
15,Rohit Bakshi,239005227,648-5681 Dui-Rd.,Y,hours
16,Tom McCucht,363303105,P.O. Box 313- 962 Parturient Rd.,Y,hours
17,Eric Mizell,123808238,P.O. Box 579- 2191 Gravida. Street,Y,hours
18,Grant Liu,171010151,Ap #928-3159 Vestibulum Av.,Y,hours

```

[그림 3-32] drivers.csv 파일 내용 확인하는 명령어

(3) Apache Hive에서 처리하기 위해 Apache HDFS로 저장한다.

Apache Hive가 분산 처리를 실행하기 위해서는 데이터가 분산 파일 시스템인 Apache HDFS에 저장되어 있어야 한다. 다운로드한 두 개의 파일을 HDFS에 저장한다.

```

> hadoop fs -copyFromLocal /root/driver_data/drivers.csv /
> hadoop fs -copyFromLocal /root/driver_data/timesheet.csv /

```

```
[root@sandbox driver_data]# hadoop fs -ls /
Found 15 items
drwxrwxrwx  - yarn  hadoop      0 2019-08-13 17:27 /app-logs
drw-r-xr-x  - hdfs hadoop      0 2016-10-25 07:54 /apps
drw-r-xr-x  - yarn  hadoop      0 2016-10-25 07:48 /ats
drw-r-xr-x  - hdfs hadoop      0 2016-10-25 08:01 /demo
-rw-r--r--  1 root  hdfs     2043 2019-08-13 20:57 /drivers.csv
drw-r-xr-x  - hdfs hadoop      0 2016-10-25 07:48 /hdp
drw-r-xr-x  - mapred hadoop    0 2016-10-25 07:48 /mapred
drwxrwxrwx  - mapred hadoop    0 2016-10-25 07:48 /mr-history
drw-r-xr-x  - hdfs hadoop      0 2016-10-25 07:47 /ranger
drwxrwxrwx  - spark  hadoop    0 2019-08-13 20:57 /spark-history
drwxrwxrwx  - spark  hadoop    0 2016-10-25 08:14 /spark2-history
-rw-r--r--  1 root  hdfs     26205 2019-08-13 20:54 /timesheet.csv
drwxrwxrwx  - hdfs hadoop      0 2016-10-25 08:11 /tmp
-rw-r--r--  1 root  hdfs   2272077 2019-08-13 20:55 /truck_event_text_partition.csv
```

[그림 3-33] Apache HDFS에 저장된 파일 목록 확인

3. Apache Hive로 분산 처리 수행 모듈(sql script)을 개발한다.

(1) 데이터를 저장할 테이블 생성 및 데이터를 로딩한다.

실습을 위해 두 개의 테이블을 생성하고, 각각의 테이블에 Apache HDFS에 저장된 데이터를 로딩하는 기능을 SQL로 개발한다.

```
hive> CREATE TABLE IF NOT EXISTS drivers (driverId String, name String, ssn String, location String,
  > wageplan String)
  > COMMENT 'Drivers detail'
  > ROW FORMAT DELIMITED
  > FIELDS TERMINATED BY ','
  > LINES TERMINATED BY '\n'
  > STORED AS TEXTFILE;
OK
Time taken: 0.445 seconds
```

[그림 3-34] drivers 테이블을 생성하는 스크립트 화면

테이블을 생성에 필요한 칼럼 명, 칼럼 타입, Apache HDFS에 저장된 데이터를 읽어오기 위한 구분자(Delimiter) 등을 명시하여 테이블을 생성한다.

```
hive> select * from drivers LIMIT 10;
OK
driverId          name        ssn       location      certified
10    George Vetticaden 621011971  244-4532 Nulla Rd.      N
11    Jamie Engesser   262112338  366-4125 Ac Street    N
12    Paul Coddin     198041975  Ap #622-957 Risus. Street   Y
13    Joe Niemiec    139907145  2071 Hendrerk Ave.    Y
14    Adis Cesir     820812209  Ap #810-1228 In St.      Y
15    Rohit Bakshi   239005227  648-5681 Dui- Rd.      Y
16    Tom McCucht   363303105  P.O. Box 313- 962 Parturient Rd.  Y
17    Eric Mizell    123808238  P.O. Box 579- 2191 Gravida. Street  Y
18    Grant Liu      171010151  Ap #928-3159 Vestibulum Av.   Y
Time taken: 0.342 seconds, Fetched: 10 row(s)
```

[그림 3-35] 데이터가 구조화되어 Apache Hive에 저장되었는지 확인하는 스크립트 화면

```
hive> CREATE TABLE timesheet (driverId INT, week INT, hours_logged INT , miles_logged INT)
  > COMMENT 'Timesheet detail'
  > ROW FORMAT DELIMITED
  > FIELDS TERMINATED BY ','
  > LINES TERMINATED BY '\n'
  > STORED AS TEXTFILE;
OK
Time taken: 0.488 seconds
```

[그림 3-36] timesheet 테이블을 생성하는 스크립트 화면

```
hive> LOAD DATA INPATH '/timesheet.csv' OVERWRITE INTO TABLE timesheet;
Loading data to table default.timesheet
Table default.timesheet stats: [numFiles=1, numRows=0, totalSize=26205, rawDataSize=0]
OK
Time taken: 0.982 seconds
```

[그림 3-37] 생성된 timesheet 테이블에 데이터를 로딩하는 스크립트 화면

```
hive> select * from timesheet LIMIT 5;
OK
NULL    NULL    NULL    NULL
10      1       70      3300
10      2       70      3300
10      3       60      2800
10      4       70      3100
Time taken: 0.289 seconds. Fetched: 5 row(s)
```

[그림 3-38] 데이터가 구조화되어 Apache Hive에 저장되었는지 확인하는 스크립트 화면

(2) 저장된 데이터를 처리하기 위한 수행 모듈을 개발한다.

분산 파일 시스템인 Apache HDFS에 저장된 데이터를 분산 처리하여 검색할 수 있는 기능을 개발한다.

```

hive> SELECT driverId, sum(hours_logged), sum(miles_logged) FROM timesheet GROUP BY driverId;
Query ID = root_20190813212953_a03db886-658a-4dc6-935e-072e1e365e5e
Total jobs = 1
Launching Job 1 out of 1

Status: Running (Executing on YARN cluster with App id application_1565716898955_0009)

CREATE TABLE timesheet (driverId INT, week INT, hours_logged INT, miles_logged INT)
COMMENT 'timesheets detail'
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','

LOAD DATA INPATH '/timesheet.csv' OVERWRITE INTO TABLE timesheet
TIMESHEET LIMIT 5;

-----  

    VERTICES      STATUS TOTAL COMPLETED RUNNING PENDING FAILED KILLED  

-----  

Map 1 ..... SUCCEEDED   1     1     0     0     0     0  

Reducer 2 ..... SUCCEEDED   1     1     0     0     0     0  

-----  

VERTICES: 0/02 [=====] 100% ELAPSED TIME: 0.87 s  

-----  

OK  

NULL    NULL    NULL  

10      3232   147150  

11      3642   179300  

12      2639   135962  

13      2727   134126  

14      2781   136624  

15      2734   138750  

16      2746   137205  

-----  

UNPARTITIONED  

query first group  

sum of the hours  

driverId, sum  

-----  

SELECT driverId, sum(hours_logged), sum(miles_logged) FROM timesheet

```

[그림 3-39] 데이터가 구조화되어 Apache Hive에 저장되었는지 확인하는 스크립트 화면

두 개의 테이블에 저장된 데이터를 공통된 키를 기준으로 조인하여 하나의 결과를 추출하는 기능을 개발한다.

```

hive> SELECT d.driverId, d.name, t.total_hours, t.total_miles from drivers d
    > JOIN (SELECT driverId, sum(hours_logged)total_hours, sum(miles_logged)total_miles FROM timesheet GROUP BY driverId ) t
    > ON (d.driverId = t.driverId)
    > ;
Query ID = root_20190813213528_c23961ed-7401-45e9-8dad-d3dd1e80c3b2
Total jobs = 1
Launching Job 1 out of 1

Status: Running (Executing on YARN cluster with App id application_1565716898955_0009)
CREATE QUERY TO JOIN THE DATA (DRIVERID, NAME, HOURS_LOGGED, MILES_LOGGED)

-----  

      VERTICES  STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED  

Map 1 ..... SUCCEEDED   1       1       0       0       0       0  

Map 3 ..... SUCCEEDED   1       Logged 1       0       0       0  

Reducer 2 ..... SUCCEEDED   1       1       0       0       0       0  

VERTICES: 0/3/03 [—————>] 100% ELAPSED TIME: 5.70 s
-----  

      OK  

10     George Vetticaden 3232 147150  

11     Jamie Engesser 3642 179300  

12     Paul Coddin 2639 135962  

13     Joe Niemiec 2727 134126  

-----  

The resulting data looks like:
```

[그림 3-40] 테이블을 조인하여 결과를 추출하는 화면

4. 개발된 분산 처리 수행 모듈의 실행 결과를 확인한다.

작성한 분산 처리 수행 모듈의 실행 결과 정보를 구체적으로 확인하기 위하여, All Application 목록을 조회할 수 있는 웹에 접속하여 정보를 확인한다.

The screenshot shows the Apache Hadoop 'All Applications' page. On the left, there's a sidebar with 'Cluster Metrics' and 'Scheduler Metrics' sections, and a 'Scheduler' dropdown menu. The main area displays a table of applications. The first application listed is 'application_1565716898955_0009', which is in the 'RUNNING' state. The second application listed is 'application_1565716898955_0008', which is in the 'FINISHED' state. Both applications are of type 'TEZ' and were submitted by 'root'. The table includes columns for ID, User, Name, Application Type, Queue, Application Priority, StartTime, FinishTime, State, FinalStatus, Running Containers, Allocated CPU, Allocated Memory MB, % of Queue, % of Cluster, Progress, Tracking UI, and Blacklisted Nodes.

ID	User	Name	Application Type	Queue	Application Priority	StartTime	FinishTime	State	FinalStatus	Running Containers	Allocated CPU	Allocated Memory MB	% of Queue	% of Cluster	Progress	Tracking UI	Blacklisted Nodes
application_1565716898955_0009	root	HIVE-cc75d6d3-5884-4d44-972b-e865e81f5140	TEZ	default	0	Wed Aug 14 06:29:32 +0900 2019	N/A	RUNNING	UNDEFINED	2	2	750	33.3	33.3		ApplicationMaster	0
application_1565716898955_0008	root	HIVE-cc75d6d3-5884-4d44-972b-e865e81f5140	TEZ	default	0	Wed Aug 14 05:53:20 +0900 2019	Wed Aug 14 06:03:29 +0900 2019	FINISHED	SUCCEEDED	N/A	N/A	N/A	0.0	0.0		History	N/A

출처: HDP로 설치한 Hadoop 모니터링 웹 화면. 스크린샷.

[그림 3-41] 아파치 하둡(Apache Hadoop)의 분산 처리 수행 모듈이 실행된 결과 조회 화면

5. Apache Hive로 개발된 sql script를 스케줄러에 등록하여 실행한다.

대량의 배치 데이터를 분산 처리하는 작업은 많은 컴퓨터 자원과 시간을 필요하게 된다.

따라서 개발된 수행 모듈을 시스템의 스케줄러(crontab)나 Hadoop workflow(oozie)와 같은 스케줄러에 등록하여 작업을 주기적으로 실행하며 관리한다.

3-2. 빅데이터 실시간 수행 모듈 개발

학습 목표

- 실시간 처리를 위하여 스키마, 컬럼, 조건, 처리 방식을 결정하고 실시간 수행 모듈을 작성할 수 있다.
- 실시간 처리 중 데이터 정합성 유지를 위한 로깅, 검증, 예외 처리 공통 모듈을 작성할 수 있다.
- 주어진 테스트 절차에 의해서 데이터 실시간 처리 완료 및 성공 여부를 테스트하고, 실패 시 문제점을 파악하고 처리하는 모듈을 수정할 수 있다.

필요 지식 /

① 실시간 처리 기술 유형인 이벤트 기반 처리와 Micro Batch 방식의 기술 비교

실시간으로 유입되는 데이터를 처리하는 대표적인 방식으로 메시지 지연을 최소화하기 위해서 이벤트가 도착하는 즉시 처리하는 이벤트 기반 실시간 처리 방식과 아주 짧은 주기로 데이터를 모아서 한 번에 처리하여 처리량을 최대화하는 Micro Batch 방식의 실시간 처리 기술로 분류된다. 이 두 가지 방식을 대표하는 기술인 Apache Storm과 Apache Spark Streaming 기술의 특징을 비교하고, 요구 사항에 맞게 기술을 활용한다.

<표 3-16> Apache Storm vs Apache Spark Streaming

구분	Apache Storm	Apache Spark Streaming
지원 언어	Java, Closure, Scala	Java, Scala, Python
처리 모델	이벤트 기반(즉시 처리)	Micro-Batch(짧은 주기)
처리 단위	Tuple(이벤트)	DStream(짧은 주기의 데이터)
장애 관리	Zookeeper로 장애 모니터링 및 재시작	YARN, Mesos 등의 지원 관리자가 장애 관리
지연	낮음.	높음.(배치 주기에 따라 다름.)
처리량	낮음.	높음.(배치 주기에 따라 다름.)
활용	빠른 메시지 처리가 필요한 업무	실시간 처리할 데이터가 많은 업무

② 대표적인 실시간 처리 기술인 Apache Spark Streaming 기술 이해

실시간으로 유입되는 대용량의 데이터(SNS, Log, Image 등)를 빠르고 안정적으로 분산 처리할 수 있도록 설계된 메모리(RDD) 기반의 스트리밍 설루션이다.

1. 스트리밍(Stream) 데이터의 이해

고정된 저장소에 저장된 데이터가 아닌 “지속적으로 생성되어 유입되는 데이터”를 말한다. 특히 유입되는 주기가 매우 짧으며, 유입량의 변화량도 큰 특징이 있다. 크리스마스 이브나 세계적인 이슈에 트위터 메시지 전송량이 급증하는 경우를 볼 수 있다. 이러한 스트리밍 데이터는 IoT 센서, SNS 메신저, 웹서버 로그, 카드 결제 이력 등을 통해 생성된다.

2. 스트리밍 데이터 처리의 이해

연속적으로 입력되는 스트리밍 데이터를 짧은 주기로 모아서 한 번에 처리하는 방식을 스트리밍 데이터 처리라고 한다. 이때 데이터는 특정 저장소에 저장되지 않고, 메모리상에서 처리하여 처리 성능을 높인다. 스트리밍 데이터 처리의 최종 목적은 저장된 후 결과를 계산하는 것이 아닌, 저장 전에 원하는 결과를 계산/처리하여 외부에서 활용하는 것이다.

3. Apache Spark Streaming 기술 아키텍처

다양한 오픈 소스 소프트웨어(Kafka, Flume, Twitter 등)로부터 유입되는 데이터를 Spark에서 제공하는 추상화된 API를 이용해 복잡한 알고리즘을 쉽게 처리할 수 있으며, 그 결과를 분산 저장 장치에 저장하는 아키텍처를 제공한다.



출처: Apache Spark 공식 홈페이지(<https://spark.apache.org>). 2019. 08. 09. 스크린샷.

[그림 3-42] Apache Spark streaming 개념 아키텍처

4. Apache Spark Streaming 처리 메커니즘

끊임없이 유입되는 데이터를 짧은 주기(0.1초 ~ 수 초) 구간으로 수집하고, 이렇게 데이터(DStream)을 RDD라는 자료 구조로 변환한 후, 여러 개의 서버에서 분산 처리한다. 즉, 연속적으로 입력되는 데이터를 일정 주기(batch interval)별로 잘라서, Spark Engine에서 분산 처리하는 방식이다.



출처: Apache Spark 공식 홈페이지(<https://spark.apache.org>). 2019. 08. 09. 스크린샷.

[그림 2-43] Apache Spark streaming 데이터 흐름

이렇게 특정 주기로 입력되는 데이터는 RDD로 처리되는데, 연속되는 RDD의 집합을 DStream이라고 한다.

<표 3-17> DStream vs RDD 비교

구분	DStream	RDD
자료 구조	RDD의 연속된 집합	Spark 자체 자료 구조
데이터 처리	DStream 단위 처리 함수	RDD 자체 처리 함수
윈도 함수 지원	가능	불가
활용	DStream 결합 및 상태 업데이트	RDD 기반 복잡한 로직 실행

5. Apache Spark Streaming 활용 시 장점

<표 3-18> Apache Spark Streaming 장점

장점	상세 설명
'높은 처리량'	기존 Spark Core Engine의 배치 처리 성능을 유지하여, 다른 오픈 소스 대비 초당 처리량이 높다.
지연 최소화	아주 짧은 주기로 데이터를 반복적으로 처리하여 데이터 전달의 지연이 발생하지 않는다.
쉬운 구현	스트림 데이터 처리를 위한 특별한 지식 없이 기존 스트림을 처리하기 위한 구현이 가능하다
집계 및 통계 처리	실시간으로 입력되는 데이터에서 다양한 집계 및 통계 데이터를 생성할 수 있다.
다양한 라이브러리 연동	Spark MLlib를 활용하여 실시간 데이터에 대한 머신러닝 및 통계 생성이 가능하다.

③ Spark Streaming 애플리케이션 개발을 지원하는 프로그래밍 언어의 특징 이해

분산 처리를 지원하는 다양한 프로그램 언어들이 있으며, 개발 용이성, 코드의 간결성, 유지 보수성, 성능, 개발자의 언어 선호도 등을 고려하여 수행 모듈 개발에 적합한 개발언어를 선택한다.

<표 3-19> 프로그래밍 언어의 특징 비교

구분	Scala	Java	Python
성능	높음	높음	낮음
구현 용이성	어려움	보통	쉬움
학습 시간	오래 걸림.	오래 걸림.	빠른 학습 가능
데이터 분석 지원	지원	미지원	지원
빅데이터 오픈 소스 프레임워크 지원	활용	활용	활용
운영 환경 적용	가능	가능	성능 저하

수행 내용 / 실시간 수행 모듈 개발하기

재료 · 자료

- 빅데이터 처리 시스템을 설치할 서버 정보
- 빅데이터 처리 시스템 설계서
- 기존 운영 시스템 또는 신규 시스템의 아키텍처 산출물

기기(장비 · 공구)

- 리눅스 서버
- 컴퓨터, 프린터, 인터넷
- 사무 자동화 소프트웨어

안전 · 유의 사항

- 해당 사항 없음.

수행 순서

① 빅데이터 실시간 처리를 위한 오픈 소스 소프트웨어를 선정한다.

대용량 실시간 데이터를 빠르고 안정적으로 처리하기 위한 오픈 소스 소프트웨어로는 Apache Spark Streaming이 가장 많은 관심을 받고 있다. 즉, 오픈 소스 커뮤니티의 개발자가 많이 참여하고, 많은 기업들이 운영 시스템에 적용하면서 버그 수정 및 성능/기능을 개선하고 있다.

② Apache Spark Streaming 애플리케이션으로 실시간 처리 수행 모듈을 개발한다.

1. 실시간 처리 수행 모듈의 설계 내용을 확인한다.

처리해야 할 데이터가 메시지의 자연 없이 빠르게 전달되어야 하는 업무인지, 실시간 처리량을 최대화하여 가능한 많은 실시간 데이터를 처리해야 하는 업무인지 요구 사항 정의서 및 설계서를 확인한다. 본 교재에서는 실시간으로 유입되는 문자열에서 1초 단위로 단 어별 카운트를 계산하여 출력하는 실시간 처리 수행 모듈을 개발한다.

2. 실시간 수행 모듈을 개발할 프로그래밍 언어를 선정한다.

실시간 처리 성능이 중요한 경우는 성능이 높은 scala, java와 같은 언어를 선택하고, 빠르

게 다양한 요구 사항을 구현 및 활용하기 위해서는 python과 같은 개발 언어를 선택한다.

3. Python을 이용하여 Apache Spark Streaming 수행 모듈을 개발한다.

코드의 내용이 직관적이고, 구현이 빠른 python 언어를 활용하여 실시간 수행 모듈을 개발한다.

(1) Apache Hive를 실행하기 위해 ssh 연결로 서버에 접속한다.

Apache Spark streaming 코드를 개발하고, spark-submit 명령어로 job을 분산 환경에서 실행하기 위해서 Apache Spark 명령어 실행이 가능한 HDP Sandbox로 ssh를 이용해서 접속한다.

```
>ssh root@localhost -p 2222  
Warning: Permanently added '[localhost]:2222' (RSA) to the list of known hosts.  
root@localhost's password:  
Last login: Tue Aug 13 20:27:10 2019 from 10.0.2.2  
[root@sandbox ~]#
```

[그림 3-44] ssh 연결 화면(port 2222)

(2) python 언어를 이용하여 spark streaming 소스 코드를 작성한다.

코드의 내용이 직관적이고, 구현이 빠른 python 언어를 활용하여 실시간 수행 모듈을 개발한다. 서버에서 직접 python 코드를 실행해야 하므로, 서버의 특정 디렉터리(본 실습에서는 /tmp) 아래에 코드를 작성한다.

```
>cd /tmp  
>vi /spark-streaming-demo.py
```

[그림 3-45] 소스 코드를 작성하는 명령어

```
from pyspark import SparkContext  
from pyspark.streaming import StreamingContext  
  
# Create a local StreamingContext with two working threads and a batch  
① sc = SparkContext("local[2]", "NetworkWordCount")  
② ssc = StreamingContext(sc, 2)  
    , 3333)  
# Create a DStream  
③ lines = ssc.socketTextStream("sandbox.hortonworks.com", 3333)  
  
# Split each line into words  
④ words = lines.flatMap(lambda line: line.split(" "))  
  
# Count each word in each batch  
⑤ pairs = words.map(lambda word: (word, 1))  
⑥ wordCounts = pairs.reduceByKey(lambda x, y: x + y)  
    , 1000)  
# Print each batch  
⑦ wordCounts.pprint()  
  
⑧ ssc.start()          # Start the computation  
    , 1000)  
ssc.awaitTermination() # Wait for the computation to terminate
```

[그림 3-46] spark-streaming-demo.py 소스 코드

(3) 소스 코드(spark-streaming-demo.py)를 이해한다.

Apache spark 홈페이지에서 제공하는 개발 가이드(<https://spark.apache.org/streaming/>) 및 API를 참고하여 위와 같이 동작 가능한 수행 모듈을 개발한다. 위 수행 모듈은 2초 단위로 실시간 유입되는 데이터를 처리하며, 유입되는 데이터는 네트워크 port 3333 번을 통해서 수집된다.

입력되는 메시지는 한 줄씩 처리하여, 공백을 기준으로 단어를 분리한 후, 단어별로 숫자를 1씩 매핑하는 구조로 생성한다. “(단어, 1)”과 같은 형식으로 데이터를 가공한 후, 2초 동안 유입된 데이터 전체에서 동일한 단어가 있을 경우 숫자를 더해 준다. 즉, 학교라는 단어가 3번 나왔다면 “(학교, 3)”과 같은 형식으로 출력한다.

<표 3-20> spark-streaming-demo.py 코드 실행 순서

순서	내용
①	SparkSession 객체를 이용하여 spark 세션을 생성하고, 이력에 표시될 애플리케이션 명칭과 사용할 자원(local[2])을 지정한다. 위에서 생성한 SparkContext를 기반으로 StreamingContext를 생성한다. 이때 반복 처리할 주기를 전달하는데, 위 코드에서는 2초를 지정했다.
②	이제 스트림 처리를 위한 진입점이 구성되었고, 이후에 이를 통해서 스트림 데이터를 수신하게 된다.
③	위 코드에서는 네트워크 port를 통해서 데이터를 수신하도록 구현하였다. 3333 port로 데이터가 입력되면 DStream으로 변환되어 처리되게 된다.
④	전달받은 데이터를 공백(“ ”) 기준으로 단어를 분리하도록 처리한다.
⑤	분리된 단어 단위로 1을 매핑한 데이터 세트를 생성한다.
⑥	생성된 단어들에서 동일한 단어는 합산하여 최종 단어별 카운트를 한다.
⑦	최종 처리된 결과를 화면에 출력한다.(단어별 카운트 수)
⑧	1~7번까지의 과정을 반복하는 스트림 처리를 시작한다.

4. 개발된 실시간 수행 모듈을 실행한다.

python으로 개발된 spark streaming 모듈을 분산 환경에서 실행하기 위해서는 “spark-submit”이라는 명령어를 통해서 실행해야 한다.

/usr/hdp/current/spark2-client/bin/spark-submit /tmp/spark-streaming-demo.py 명령어를 실행하면 다음과 같은 메시지가 출력된다. 아직 유입되는 로그 메시지가 없으므로, 처리 결과가 출력되지 않는다는 것이다.

```
[root@sandbox ~]# /usr/hdp/current/spark2-client/bin/spark-submit /tmp/spark-streaming-demo.py
/usr/hdp/2.5.0.0-1245/spark2/python/lib/pyspark.zip/pyspark/sql/context.py:477: DeprecationWarning: HiveCon
se SparkSession.builder.enableHiveSupport().getOrCreate() instead.
19/08/13 22:18:34 INFO SparkContext: Running Spark version 2.0.0.2.5.0.0-1245
19/08/13 22:18:34 INFO SecurityManager: Changing view acls to: root
19/08/13 22:18:34 INFO SecurityManager: Changing modify acls to: root
19/08/13 22:18:34 INFO SecurityManager: Changing view acls groups to: root
19/08/13 22:18:34 INFO SecurityManager: Changing modify acls groups to: root
19/08/13 22:18:34 INFO SecurityManager: authentication disabled; ui acls disabled; users th view permissions: Set(); users with modify permissions: Set(root); groups with modify permissions: Set()
19/08/13 22:18:35 INFO Utils: Successfully started service 'sparkDriver' on port 36732.
19/08/13 22:18:35 INFO SparkEnv: Registering MapOutputTracker
19/08/13 22:18:35 INFO SparkEnv: Registering BlockManagerMaster
19/08/13 22:18:35 INFO DiskBlockManager: Created local directory at /tmp/blockmgr-e0f0d3de-ef4c-4a01-b5f5-b
19/08/13 22:18:35 INFO MemoryStore: MemoryStore started with capacity 366.3 MB
19/08/13 22:18:35 INFO SparkEnv: Registering OutputCommitCoordinator
19/08/13 22:18:35 INFO log: Logging initialized @1839ms
19/08/13 22:18:35 INFO Server: jetty-9.2.z-SNAPSHOT
19/08/13 22:18:35 INFO ContextHandler: Started o.s.j.s.ServletContextHandler@699ee369{/jobs,null,AVAILABLE}
19/08/13 22:18:38 INFO TaskSchedulerImpl: Removed TaskSet 4.0, whose tasks have all completed, from pool
19/08/13 22:18:38 INFO DAGScheduler: ResultStage 4 (runJob at PythonRDD.scala:441) finished in 0.114 s
19/08/13 22:18:38 INFO DAGScheduler: Job 2 finished: runJob at PythonRDD.scala:441, took 0.136110 s
-----
Time: 2019-08-13 22:18:38
-----
```

[그림 3-47] spark-submit 실행 화면

5. 실시간 유입될 데이터를 생성하여 정상 작동을 확인한다.

netcat을 이용하여 spark driver가 실행되는 서버의 3333 포트로 메시지를 전달하는 명령어를 실행한다. 정상적으로 처리된다면 이전에 실행한 실시간 수행 모듈에서 전달한 메시지를 공백 단위로 분리하고, 각 단어별로 발생 빈도를 카운트하여 로그로 출력한다.

```
[root@sandbox ~]# nc -l sandbox.hortonworks.com 3333
Hello from the sandbox team!
```

[그림 3-48] netcat(nc) 명령어로 실시간 수행 모듈이 동작하는 서버의 3333 포트로 메시지 전송하는 화면

```
19/08/13 22:13:14 INFO DAGScheduler: ResultStage 32 (runJob at
19/08/13 22:13:14 INFO DAGScheduler: Job 16 finished: runJob at
-----
Time: 2019-08-13 22:13:14
-----
```

(u'sandbox', 1)
(u'from', 1)
(u'the', 1)
(u'Hello', 1)
(u'team!', 1)

At this point you should be
For example, if we type the
following output in the run

[그림 3-49] netcat에서 전달된 메시지를 분리하여 단어별 카운트를 출력한 화면

수행 tip

- 서버의 IP가 아닌 hostname을 사용하는 경우, 해당 서버의 /etc/hosts 파일에 hostname0 등록되어 있는지 확인한다.
- hostname0 netcat과 spark streaming 모듈이 3333 포트에 접속하지 못하고 에러가 발생한다.

③ Apache Spark Streaming의 환경설정을 조정한다.

Apache Spark streaming은 분산 처리에서 발생하는 수많은 로그를 파일 및 콘솔로 출력을 한다. 하지만, 운영 환경에서 이러한 로그들은 문제(ERROR, WARNING)를 파악하는데 어려움을 주고, 불필요한 로그 크기 가로 서버의 저장 공간을 낭비하게 된다. 이 로깅 설정을 변경하여 WARNING 수준의 메시지만 출력하도록 설정한다.

```
vi /usr/hdp/current/spark2-client/conf/log4j.properties
```

```
# Set everything to be logged to the console
log4j.rootCategory=INFO, console
log4j.appender.console=org.apache.log4j.ConsoleAppender
log4j.appender.console.target=System.err
log4j.appender.console.layout=org.apache.log4j.PatternLayout
log4j.appender.console.layout.ConversionPattern=%d{yy/MM/dd HH:mm:ss} %p %c{1}: %m%n

# Settings to quiet third party logs that are too verbose
log4j.logger.org.eclipse.jetty=WARN
log4j.logger.org.eclipse.jetty.util.component.AbstractLifeCycle=ERROR
log4j.logger.org.apache.spark.repl.SparkIMain$exprTyper=INFO
log4j.logger.org.apache.spark.repl.SparkILoop$SparkILoopInterpreter=INFO
```

[그림 3-50] 로그와 관련된 설정 파일

위 설정에서 log4j.rootCategory=INFO, console을 WARN수준의 로그만 출력하도록 변경
log4j.rootCategory=WARN,console하여 파일을 저장한 후, 다시 spark-submit을 실행하면 이전과 달리 로그 메시지가 훨씬 줄어든 것을 확인할 수 있다.

```
[root@sandbox ~]# /usr/hdp/current/spark2-client/bin/spark-submit /tmp/spark-streaming-demo.py
/usr/hdp/2.5.0.0-1245/spark2/python/lib/pyspark.zip/pyspark/sql/context.py:477: DeprecationWarning: SparkSession.builder.enableHiveSupport().getOrCreate() instead.
/usr/hdp/2.5.0.0-1245/spark2/python/lib/pyspark.zip/pyspark/sql/context.py:477: DeprecationWarning: SparkSession.builder.enableHiveSupport().getOrCreate() instead.
-----
Time: 2019-08-13 22:23:42          log4j.logger.org
-----          log4j.logger.org
```

[그림 3-51] spark-submit 실행 시 출력 화면

3-3. 빅데이터 이벤트 처리 수행 모듈 개발

학습 목표

- 스트림 데이터의 특정 패턴에 대한 쿼리를 작성할 수 있다.
- 이벤트 감지와 처리를 하기 위한 규칙을 정의하고, 규칙에 따라 이벤트 처리를 수행 할 수 있는 모듈을 작성할 수 있다.
- 주어진 테스트 절차에 의해서 데이터 이벤트 처리 완료 및 성공 여부를 테스트하고, 실패 시 문제점을 파악하고 처리하는 모듈을 수정할 수 있다

필요 지식 /

① 룰 엔진

룰 엔진은 실시간으로 입력되는 다수의 이벤트 간의 관계 및 규칙을 기반으로 특정 조건을 만족하는 이벤트를 빠르게 탐지할 수 있도록 처리하는 엔진이다.

② 룰 정의 방식

룰 정의 방식은 사용자가 직접 룰을 코딩할 수 있도록 자체 스크립트를 제공하는 방법과 SQL 기반의 룰을 이용하는 방법이 있다. 좀 더 복잡한 관계를 탐지하기 위해서는 코드로 룰을 정의할 수 있는 방법이 효과적이다.

③ JBoss Drools

JBoss Drools는 다양한 이벤트에서 룰 기반으로 복합적인 패턴을 가지는 이벤트를 탐지할 수 있는 룰 엔진을 제공한다. 기존 프로그래밍 방식에서는 간단한 룰(예를 들어 알람을 온도 40도에서 45도로 변경 등)의 변경에도 프로그램을 변경하고 빌드한 후에 다시 배포해야 하며, 이로 인해 시스템의 가용성과 비즈니스 요건을 적시에 만족하기 어려운 구조였다. 룰 엔진은 동적으로 룰을 변경하여도 프로그램의 중지 없이 바로 변경된 룰이 적용되므로 빠른 비즈니스 요구 사항 적용이 가능하다.

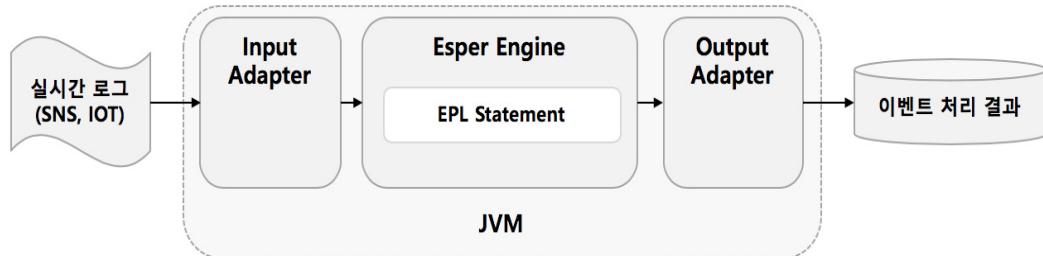
④ Esper

Esper는 실시간 스트림 데이터의 복잡한 이벤트 처리에 최적화된 오픈 소스 룰 엔진이다. Apache Spark Streaming이나 Apache Storm과 같은 실시간 처리 기술은 유입되는 데이터의 단순 가공 및 집계 등의 처리는 지원하지만, 입력되는 데이터의 복잡한 패턴을 탐지하고, 이벤트를 처리하는 기능(CEP, Complex Event Processing)을 제공하지 않는다. Esper는 복잡한 이벤트 처리에 필요한 다양한 조건과 이벤트 발생 조건을 룰로 쉽게 정의할 수 있다.

1. Esper 아키텍처 구성도

Esper는 JVM(Java Virtual Machine)상에서 동작하는 java 기반 프로그램이다. 다양한 입력 데이터를 Input Adaptor를 통해서 수신하고, Esper Engine에서 정의한 이벤트 처리

룰에 따라서 패턴을 탐지한다. 그리고 최종 결과를 Output Adapter에서 외부 저장소로 저장하는 구성이다.



[그림 3-52] Esper 아키텍처 구성도

2. Esper 구성 요소

<표 3-21> Esper 구성 요소

구성 요소	설명
Input Adaptor	다양한 외부 데이터를 수집할 수 있도록 제공한다. Socket, HTTP, csv 등의 Adapter를 제공한다.
Esper Engine	Input Adapter에서 수신한 데이터를 이벤트로 변환 또는 복잡한 이벤트 패턴을 탐지하는 엔진이다. 다양한 이벤트에 대한 분석을 실행할 때 SQL과 유사한 EPL 구문을 활용한다.
EPL	Event Processing Language Esper Engine에 저장된 이벤트 데이터를 SQL 구문을 이용하여 쉽게 탐지할 수 있는 이벤트 처리 언어이다.
Output Adapter	Esper Engine에서 탐지된 이벤트 결과를 외부 저장소에 저장한다.

3. 이벤트 처리의 핵심 구문, EPL 문법 이해

EPL 구문은 esper engine 내부에서 생성되고 저장되어 관리되며, 이벤트가 입력되면 EPL을 실행하여 결과를 전달한다.

(1) 간단한 select 구문

select 구문은 입력된 이벤트에서 어떤 속성을 추출할 것인지 정의한다. 다음 예시는 SK라는 회사에서 최근 30초간의 평균 주가를 추출하는 구문이다.

```
select avg(price) from StockTick#time(30 sec) where symbol='SK'
```

(2) 복잡한 조건의 EPL 문법

복잡한 조건의 데이터를 추출하기 위해서는 단순 데이터 조회가 아닌 where 구문을 통한 필터링 또는 group by를 통한 집계 등의 질의를 통해 추출 가능하다.

```
select select_list
from stream_def [as name] [, stream_def [as name]] [,...]
[where search_conditions]
[group by grouping_expression_list]
[having grouping_search_conditions]
[output output_specification]
[order by order_by_expression_list]
[limit num_rows]
```

출처: Esper 공식
홈페이지(<http://esper.espertech.com/release-7.1.0/esper-reference>). 2019. 08. 09.
스크린샷.
[그림 3-53] EPL 문법 가이드

(3) 시간 구간 지정하기

이벤트 탐지를 위해 특정 타임 구간(20초, 1분, 1시간 등) 동안의 데이터를 기준으로 계산할 수 있다.

```
10 seconds
10 minutes 30 seconds
20 sec 100 msec
1 day 2 hours 20 minutes 15 seconds 110 milliseconds
0.5 minutes
1 year
1 year 1 month
```

출처: Esper 공식
홈페이지(<http://esper.espertech.com/release-7.1.0/esper-reference>). 2019. 08. 09.
스크린샷.
[그림 3-54] Time Period 가이드

(4) 이벤트 선언 방식

이벤트를 선언하는 방식은 Java Object, Map 객체, XML 등 다양한 방식을 지원한다. 보통 java 프로그램 내부에서 이벤트를 활용하는 경우에는 Java Object나 Map 객체를 많이 활용한다.

```

public class MyEvent {
    private Map props = new HashMap();
    private Object[] array = new Object[10];

    public void setProps(String name, Object value) {
        props.put(name, value);
    }

    public void setArray(int index, Object value) {
        array[index] = value;
    }
}

```

출처: Esper 공식
홈페이지(<http://esper.espertech.com/release-7.1.0/esper-reference>). 2019.

08. 09. 스크린샷.

[그림 3-55] Java Object 이벤트 정의 예시

위와 같이 정의된 이벤트값을 수정하기 위해서는 다음과 같은 EPL 구문을 활용한다.

update istream MyEventStream set props('key') = 'abc', array[2] = 100

```

Map<String, Object> event = new HashMap<String, Object>();
event.put("carId", carId);
event.put("direction", direction);

```

출처: Esper 공식 홈페이지(<http://esper.espertech.com/release-7.1.0/esper-reference>).

2019. 08. 09. 스크린샷.

[그림 3-56] Map 객체 이벤트 정의 예시

(5) 이벤트 Schema 활용

EPL을 이용하여 이벤트를 정의하고, 정의된 이벤트에 데이터를 추가할 수 있다.

ParkingEvent를 정의한다.

create map schema ParkingEvent as(carId string, driverName string)

ParkingEvent를 다른 이벤트(CarArrivalEvent)에서 생성한다.

insert into ParkingEvent select carId, 'jim' as driverName from CarArrivalEvent

⑤ Java 개발 환경 구성 이해

Java 언어로 애플리케이션을 개발하기 위해서는 다양한 개발 도구를 활용하는 것이 효율적이며 개발 생산성을 높여 준다. 일단 java 언어를 실행하기 위해서는 jdk(java development kit)가 필요하고, java 프로그래밍에 필요한 다양한 환경을 지원하는 개발 환경(IntelliJ 등)이 필요하다.

1. JDK(Java Development Kit)

Java 프로그래밍을 위해 필요한 컴파일러 및 실행 환경을 제공하는 도구이며 오라클에서 관리하고 있다. 오라클

홈페이지(<https://www.oracle.com/technetwork/java/javase/downloads/index.html>)에서

다운로드할 수 있다.

2. IntelliJJ 개발 환경

JetBrains사에서 제작한 상용 자바 통합 개발 환경으로 Java, Scala 등 개발 언어를 지원하며, 애플리케이션을 빠르고 효율적으로 개발할 수 있는 다양한 기능과 플러그인을 제공한다. JetBrains 홈페이지(<https://www.jetbrains.com/idea/>)에서 다운로드할 수 있다.

3. Apache Maven

Java 프로젝트의 빌드 및 배포를 자동화하는 빌드 도구로 소스 코드를 컴파일하고 패키지해서 배포하는 전체 과정을 자동화해 준다. 이러한 빌드 및 배포에 관련된 설정은 java 프로젝트의 상위 디렉터리의 pom.xml에 작성하여 관리한다.

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
                               http://maven.apache.org/xsd/maven-4.0.0.xsd"
          modelVersion="4.0.0">

    <groupId>ncs</groupId>
    <artifactId>simple-esper</artifactId>
    <version>1.0-SNAPSHOT</version>
    <dependencies>
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>3.8.1</version>
            <scope>test</scope>
        </dependency>
    </dependencies>
</project>
```

[그림 3-57] pom.xml 설정 예시

pom.xml에서 사용되는 주요 설정값은 다음 표와 같다.

<표 3-22> Maven pom.xml 설정 정보

설정 정보	설명
<modelVersion>	POM 모델의 버전이다. 별도 수정 없이 사용한다.
<groupId>	java 프로젝트를 생성할 때에 프로젝트가 속한 조직명(회사 또는 단체)을 입력한다.
<artifactId>	생성할 java 프로젝트를 식별하는 고유 명칭을 지정한다.
<version>	프로젝트의 현재 버전을 입력한다.
<dependency>	현재 java 프로젝트에서 사용하는 라이브러리 정보를 정의한다. 여기서 정의된 라이브러리는 maven 중앙 저장소에서 검색하여 자동으로 다운로드하여서, java 프로그램이 동작될 수 있도록 한다.

수행 내용 / 이벤트 처리 수행 모듈 개발하기

재료 · 자료

- 빅데이터 처리 시스템을 설치할 서버 정보
- 빅데이터 처리 시스템 설계서
- 기존 운영 시스템 또는 신규 시스템의 아키텍처 산출물

기기(장비 · 공구)

- 리눅스 서버
- 컴퓨터, 프린터, 인터넷
- 사무 자동화 소프트웨어

안전 · 유의 사항

- 해당 사항 없음.

수행 순서

① 빅데이터 이벤트 처리를 위한 오픈 소스 소프트웨어를 선정한다.

실시간 이벤트를 SQL 기반으로 쉽고 빠르게 처리하는 엔진은 Esper가 최적의 오픈 소스 소프트웨어이다. Drools는 자체 룰 스크립트를 제공하나, 별도의 문법을 학습해야 한다.

② Esper를 이용한 이벤트 처리 수행 모듈을 개발한다.

Esper 라이브러리를 이용하여 이벤트 처리가 필요한 애플리케이션에 통합하거나, 이벤트 처리를 위한 별도의 모듈로 개발할 수 있다. 본 교재에서는 이벤트 처리용으로 별도의 자바 애플리케이션을 개발한다.

1. 이벤트 처리 수행 모듈의 설계 내용을 확인한다.

개발할 이벤트 처리 수행 모듈을 개발하기 위하여 설계된 문서를 참고한다. 입력 데이터와 처리를 통해 생성할 데이터 등의 입출력 내용을 확인한다. 본 교재에서는 온도 센서에서 입력되는 현재 온도 이벤트가 입력될 때마다, 이전 3초간의 평균 온도를 계산하여 출력한다.(이상 온도 감지를 위한 예시)

2. 이벤트 처리 수행 모듈을 개발할 프로그래밍 언어를 선정한다.

선정한 Esper는 java 기반 개발 API를 지원하므로 java 언어를 활용하여 이벤트 처리 수행 모듈을 개발한다.

3. Java 개발 환경을 구성한다.

Java 프로그래밍에 필요한 jdk, IntelliJ를 다운로드받고 설치한다.

(1) JDK를 설치한다.

다운로드 링크(<https://www.oracle.com/technetwork/java/javase/downloads/index.html>)에서 현재 운영 체제에 맞는 JDK를 다운로드하고 설치한다. 자세한 내용은 많은 인터넷 자료에서 가이드하고 있으므로 참고하여 설치한다.

(2) IntelliJ를 설치한다.

JetBrains 홈페이지(<https://www.jetbrains.com/idea/>)에서 무료 버전인 CE(Community Edition)을 다운로드하여 설치한다. 별도의 설정이 없으며 현재 운영 체제에 맞는 버전을 다운받아서 압축을 해제한다.

4. Java를 이용하여 이벤트 처리 수행 모듈을 개발한다.

(1) Esper 버전 7.0.0을 다운로드한다

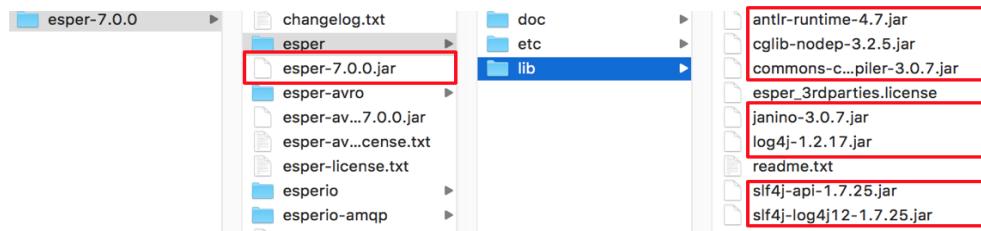
이벤트 처리 수행 모듈을 개발하기 위해서는 esper에서 제공하는 다양한 이벤트 처리용 라이브러리가 필요하다. 본 교재에서는 esper 버전 7.0.0을 이용하여 개발한다. 다운로드 페이지(<http://esper.espertech.com/distributions/>)에서 “esper-7.0.0.tar.gz” 파일을 다운로드한다.



[그림 3-58] esper 7.0.0 버전 다운로드 화면

다운로드한 파일의 압축을 해제하면 다음과 같은 파일이 보이고, 이 중 이벤트 처리 수행 모듈에서 사용할 라이브러리는 박스로 표시한 목록이다.

esper-7.0.0.jar, antlr-runtime-4.7.jar, cglib-nodep-3.2.5.jar,
commons-compiler-3.0.7.jar, janino-3.0.7.jar, log4j-1.2.17.jar, slf4j-api-1.7.25.jar,
slf4j-log4j12-1.7.25.jar



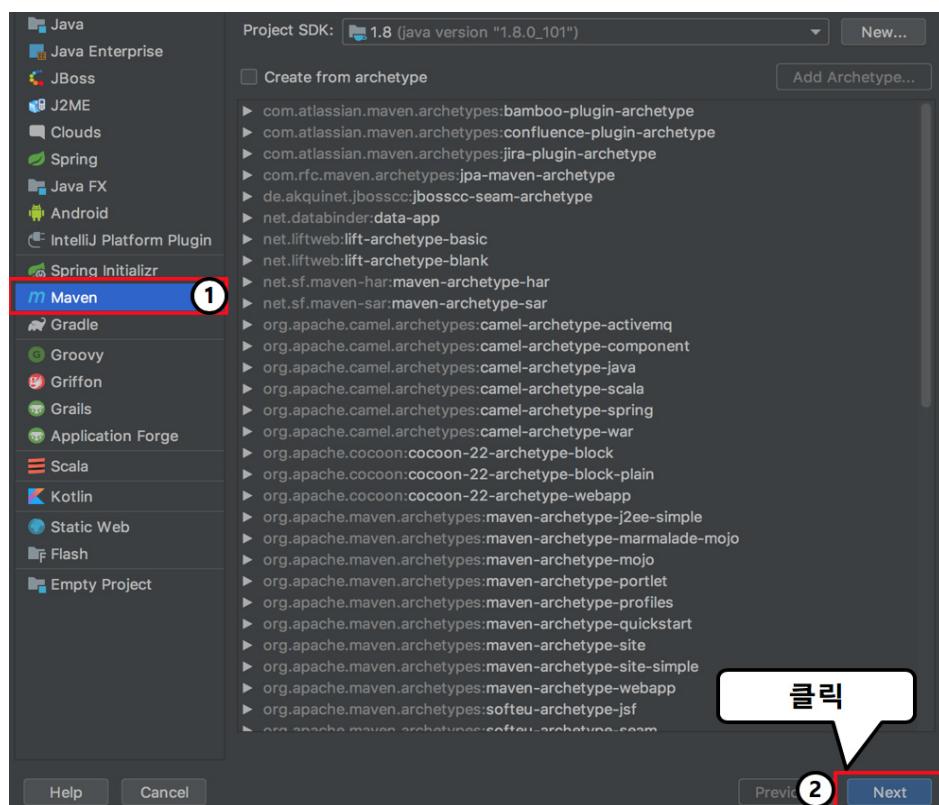
[그림 3-59] esper-7.0.0.tar.gz 파일 압축 해제 및 활용할 라이브러리 파일

(2) IntelliJ로 maven 프로젝트를 생성한다.

java 애플리케이션의 빌드 및 배포 관리하기 위해 maven 프로젝트를 생성한다.

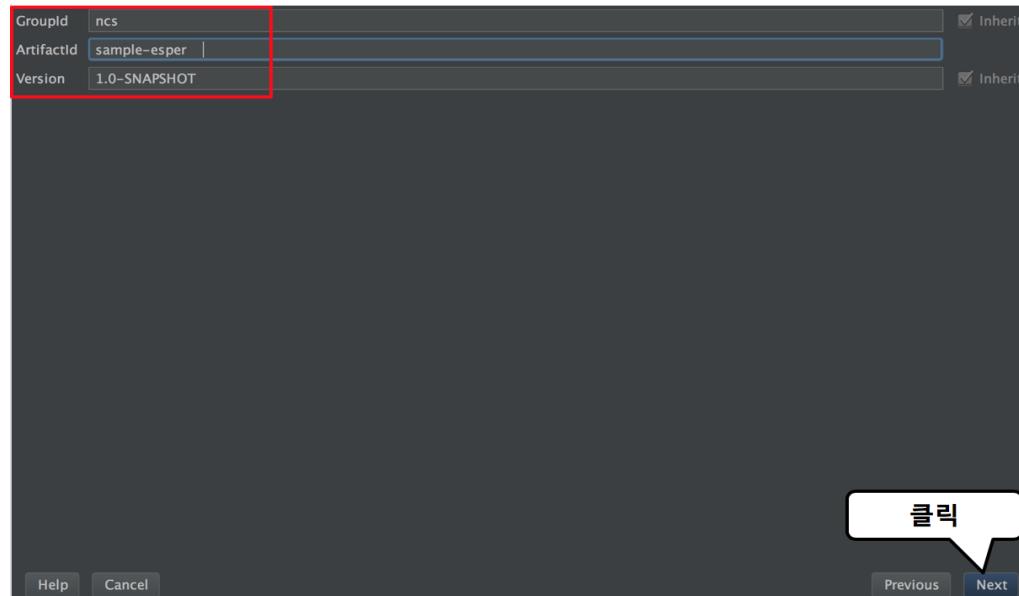


[그림 3-60] IntelliJ 신규 프로젝트 생성 화면

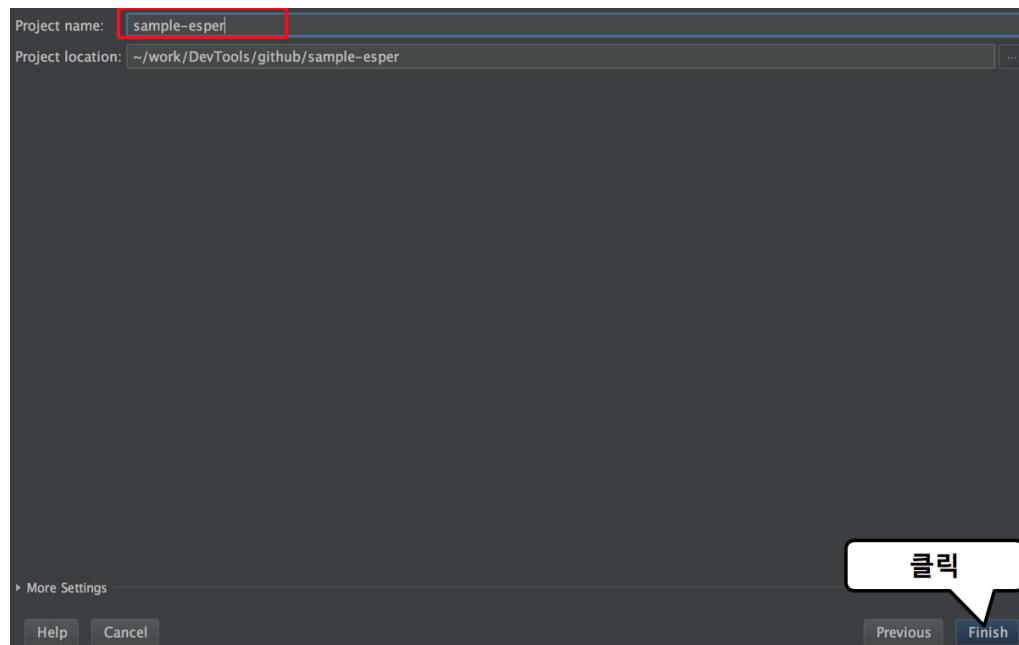


[그림 3-61] Maven 프로젝트 지정 후 “Next” 클릭

생성할 프로젝트의 GroupId, ArtifactId, Version 정보를 입력한다. 이 정보를 기반으로 maven의 pom.xml이 생성된다.



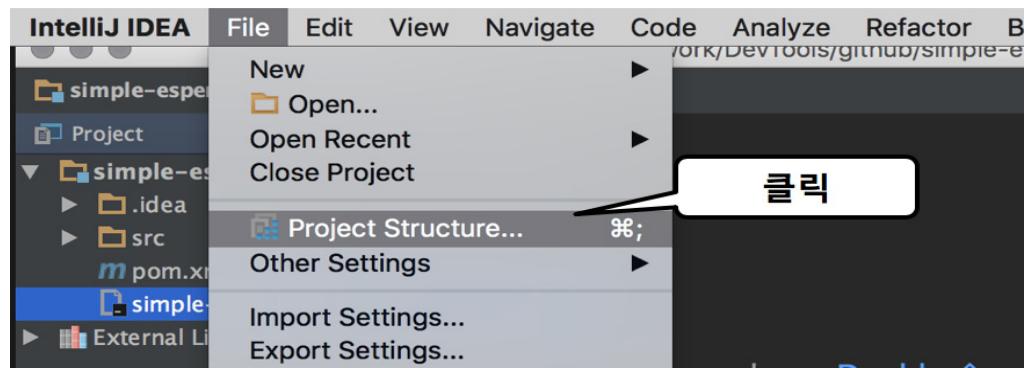
[그림 3-62] 프로젝트 기본 정보 입력 후 “Next” 클릭



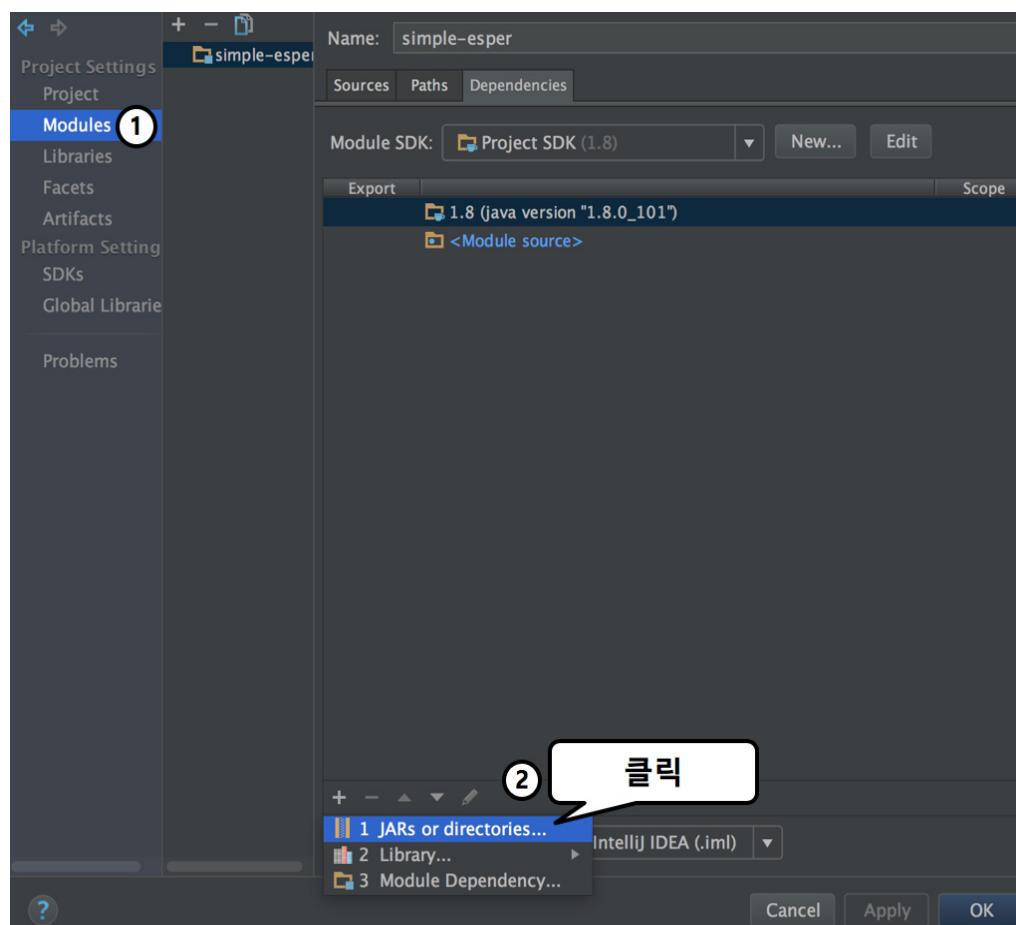
[그림 3-63] 생성할 프로젝트 이름 입력 후 “Next” 클릭

(3) esper에서 제공하는 라이브러리를 설정한다.

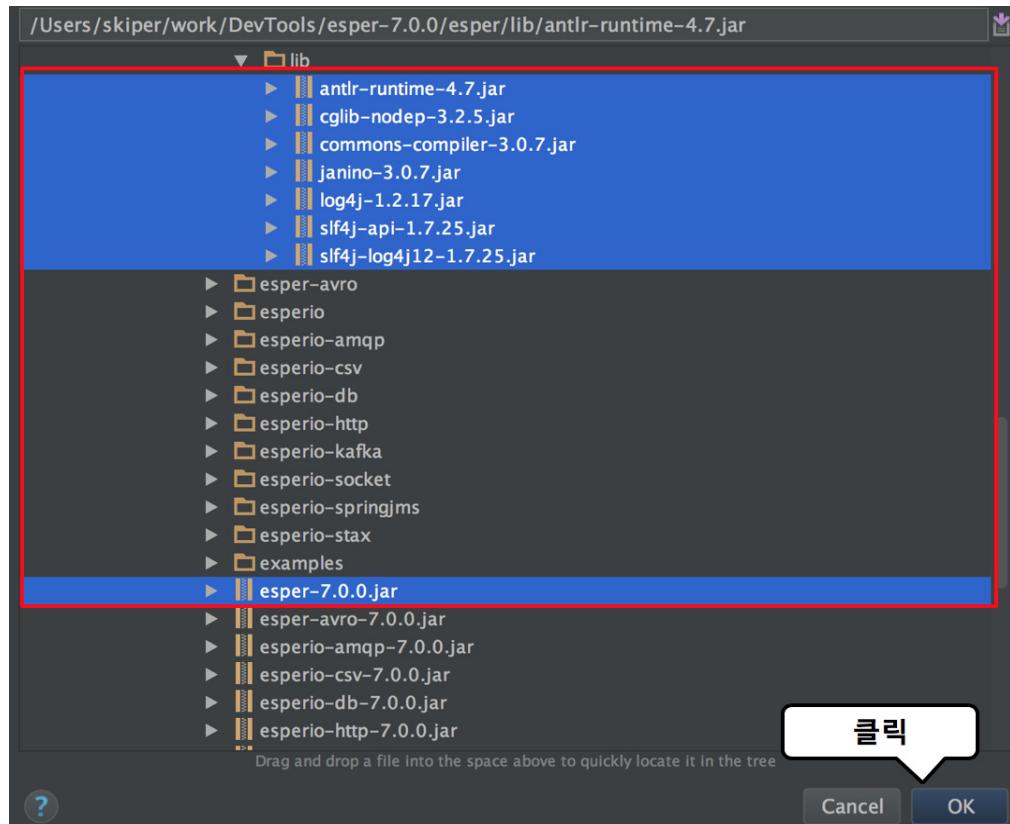
압축을 해제한 esper 디렉터리에서 필요한 esper 라이브러리를 연결한다.



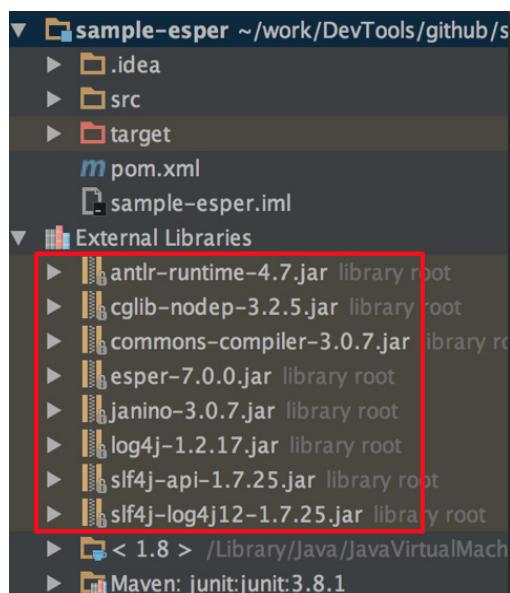
[그림 3-64] File > Project Structure 클릭 화면



[그림 3-65] Module 클릭 후 “JAR or directories” 클릭 화면



[그림 3-66] 압축을 푼 esper 디렉터리에서 위 파일을 선택하고, OK 클릭 화면



[그림 3-67] 가져온 라이브러리가 표시된 화면

프로젝트 화면에 위 라이브러리가 표시되면 정상적으로 라이브러리를 연동한 것이다.

(4) Maven 빌드 설정을 한다.

java 애플리케이션에서 사용할 의존 라이브러리나 빌드에 필요한 설정을 추가한다. 본 교재에서는 기본 pom.xml을 변경 없이 사용한다.

```

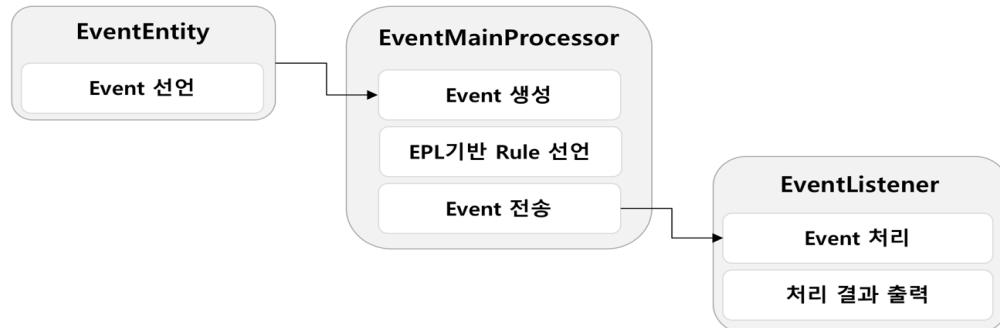
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
                               http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <groupId>ncs</groupId>
    <artifactId>simple-esper</artifactId>
    <version>1.0-SNAPSHOT</version>
    <dependencies>
    </dependencies>
</project>

```

[그림 3-68] maven pom.xml 설정 화면

(5) Java 애플리케이션을 개발한다.

설계서에 명시된 기능을 만족하는 애플리케이션을 구현하기 위해 필요한 클래스를 생성하고 함수를 구현한다. 본 교재에서는 EventEntity 클래스에 선언된 객체를 EventMainProcessor 클래스에서 생성하고, EventListener로 전송한다. 또 EventListener에서는 EventMainProcessor에서 EPL로 정의한 룰을 기반으로 수신받은 이벤트를 처리하여 결과를 출력하는 역할을 한다.



[그림 3-69] 이벤트 처리 수행 모듈의 클래스 관계 예시 화면

<표 3-23> 각 클래스의 역할

클래스	역할
EventEntity	이벤트를 Java Object 방식으로 선언하여, 속성(item, price)를 가진다. 이후 모든 이벤트 처리는 이 클래스를 기준으로 처리된다.
EventMainProcessor	EPL 룰을 정의하고, 이를 EventListener에 등록한다. 외부에서 입력받은 데이터를 SampleEvent로 변환하여 생성한 후 EventListener로 전달한다. EPL: select item, count(*), avg(price) from EventEntity.win:time(3 sec) 설명: 최근 3초간의 EventEntity의 입력 개수(count(*))와 평균 price 를 출력하는 EPL 구문이다.
EventListener	EventMainProcessor에서 등록한 룰에 따라서 수신된 이벤트 (EventEntity)를 처리하고 결과를 출력한다.

```

public class EventEntity {
    private String item = null;
    private Double price = null;

    public EventEntity(String item, Double price) {
        super();
        this.item = item;
        this.price = price;
    }

    public String getItem() { return item; }
    public void setItem(String item) { this.item = item; }
    public Double getPrice() { return price; }
    public void setPrice(Double price) { this.price = price; }
}

```

[그림 3-70] EventType.java 화면

```

import com.espertech.esper.client.EventBean;
import com.espertech.esper.client.UpdateListener;

public class EventListener implements UpdateListener {
    public void update(EventBean[] newEvents, EventBean[] oldEvents) {
        EventBean event = newEvents[0];
        System.out.println(
            "EventName : " + event.get("item") +
            ", Count : " + event.get("count(*)") +
            ", Average Price : " + event.get("avg(price)"));
    }
}

```

[그림 3-71] EventListener.java 화면

```

import com.espertech.esper.client.Configuration;
import com.espertech.esper.client.EPRuntime;
import com.espertech.esper.client.EPServiceProvider;
import com.espertech.esper.client.EPServiceProviderManager;
import com.espertech.esper.client.EPStatement;

public class EventMainProcessor {
    public static void main(String[] args) {
        Configuration config = null;
        EPServiceProvider service = null;
        EPStatement stat = null;
        String epl = null;
        EventListener listener = null;
        EPRuntime runtime = null;

        config = new Configuration();
        config.addEventType("EventEntity", EventEntity.class.getName());
        service = EPServiceProviderManager.getDefaultProvider(config);

        epl = "select item, count(*), avg(price) from EventEntity.win:time(3 sec)";
        stat = service.getEPAdministrator().createEPL(epl);

        listener = new EventListener();
        stat.addListener(listener);
    }

    runtime = service.getEPRuntime();
    for (int i = 1; i <= 20; i++) {
        runtime.sendEvent(new EventEntity("MyEvent-" + i, 10.0));
        try {
            Thread.sleep(300);
        } catch (InterruptedException ex) {
            ex.printStackTrace();
        }
    }
}

```

SampleEvent 클래스를
이벤트로 등록

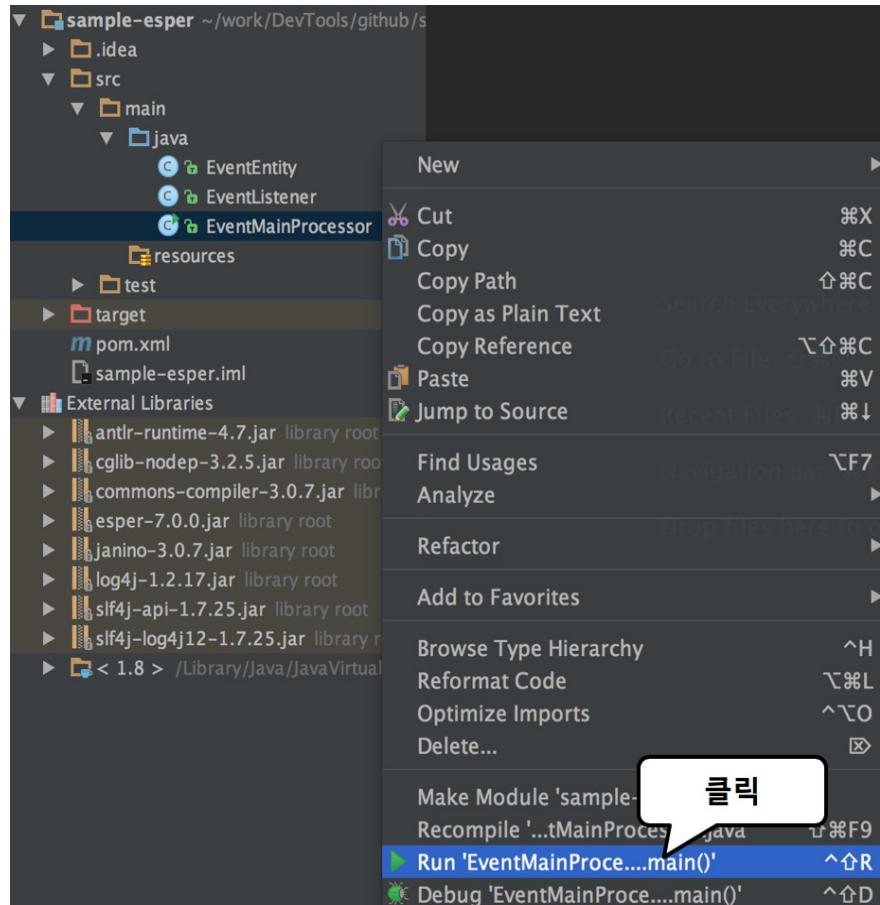
EPL을 정의하고
Listener에 등록

SampleEvent를
생성하고 전달

[그림 3-72] EventMainProcessor.java 화면

5. 이벤트 처리 수행 모듈을 실행하여 정상 작동을 확인한다.

EventMainProcessor 클래스의 메인 함수를 실행하여 esper 기반의 이벤트 처리 엔진이 정상적으로 출력되는지 확인한다.



[그림 3-73] EventMainProcessor 클래스의 메인 함수 실행 화면

```

EventName : MyEvent-1, Count : 1, Average Price : 10.0
EventName : MyEvent-2, Count : 2, Average Price : 10.0
EventName : MyEvent-3, Count : 3, Average Price : 10.0
EventName : MyEvent-4, Count : 4, Average Price : 10.0
EventName : MyEvent-5, Count : 5, Average Price : 10.0
EventName : MyEvent-6, Count : 6, Average Price : 10.0
EventName : MyEvent-7, Count : 7, Average Price : 10.0
EventName : MyEvent-8, Count : 8, Average Price : 10.0
EventName : MyEvent-9, Count : 9, Average Price : 10.0
EventName : MyEvent-10, Count : 10, Average Price : 10.0
EventName : MyEvent-11, Count : 11, Average Price : 10.0
EventName : MyEvent-12, Count : 10, Average Price : 10.0
EventName : MyEvent-13, Count : 10, Average Price : 10.0
EventName : MyEvent-14, Count : 10, Average Price : 10.0
EventName : MyEvent-15, Count : 10, Average Price : 10.0
EventName : MyEvent-16, Count : 10, Average Price : 10.0
EventName : MyEvent-17, Count : 10, Average Price : 10.0
EventName : MyEvent-18, Count : 10, Average Price : 10.0
EventName : MyEvent-19, Count : 10, Average Price : 10.0
EventName : MyEvent-20, Count : 10, Average Price : 10.0

Process finished with exit code 0

```

[그림 3-74] 실행 결과 출력 화면

총 20개의 이벤트가 입력되었고, 각 price의 값이 10이므로 평균값도 10이 정상적으로 출력되었다.

학습 3 교수 · 학습 방법

교수 방법

- 교수자는 학습자가 분산 처리 수행 모듈을 매뉴얼에 따라 적절히 개발을 하는지 관찰하고, 학습자가 부족한 부분에 대하여 피드백을 제공한다.
- 학습자가 분산 처리 수행 모듈 개발에 필요한 프로그램 개발 환경을 정상적으로 설치 및 설정할 수 있도록, 설치 매뉴얼에 따라 가이드하고 피드백을 제공한다.
- 교수자는 개발이 완료된 수행 모듈이 정상적으로 동작하는지 테스트한 결과를 확인하고, 피드백을 제공한다.
- 교수자는 개발된 수행 모듈의 테스트가 실패한 경우, 원인을 파악하고 문제를 해결할 수 있도록 가이드한다.

학습 방법

- 학습자는 분산 처리 수행 모듈을 적합한 개발 언어로 개발할 수 있도록 인터넷의 자료를 통해 내용을 학습한다.
- 분산 처리 수행 모듈 개발을 위한 프로그램 개발 환경을 직접 설치 및 설정하고, 간단한 샘플 프로그램 동작을 통해 컴파일 및 디버깅 기능을 학습한다.
- 학습자는 테스트 계획서에 명시된 테스트 방법에 따라서 수행 모듈을 테스트하고, 테스트 결과에 따라 수행 모듈의 기능을 수정할 수 있도록 분산 처리 수행 모듈에 대한 이해를 높인다.
- 학습자가 개발한 수행 모듈의 동작에 오류가 발생하거나, 원하는 결과값이 출력되지 않았을 경우, 설계 단계의 처리 로직이 프로그램에 정상적으로 반영되었는지 디버깅 및 로그를 통해 확인할 수 있도록 학습한다.

학습 3 평 가

평가 준거

- 평가자는 학습자가 학습 목표를 성공적으로 달성하였는지를 평가해야 한다.
- 평가자는 다음 사항을 평가해야 한다.

학습 내용	학습 목표	성취수준		
		상	중	하
빅데이터 분산 처리 수행 모듈 개발	- 분산 처리 모델에 따라 대규모 데이터를 병렬 처리하는 분산 처리 수행 모듈을 작성할 수 있다.			
	- 분산 처리를 수행하기 위하여, <u>스크립트</u> 를 작성하고 등록할 수 있다.			
	- 주어진 테스트 절차에 의해서 데이터 분산 처리 완료 및 성공 여부를 테스트하고, 실패했을 때 문제점을 파악하고 처리하는 모듈을 수정할 수 있다.			
빅데이터 실시간 수행 모듈 개발	- 실시간 처리를 위하여 스키마, 컬럼, 조건, 처리 방식을 결정하고 실시간 수행 모듈을 작성할 수 있다.			
	- 실시간 처리 중 데이터 정합성 유지를 위한 로깅, 검증, 예외 처리 공통 모듈을 작성할 수 있다.			
	- 주어진 테스트 절차에 의해서 데이터 실시간 처리 완료 및 성공 여부를 테스트하고, 실패했을 때 문제점을 파악하고 처리하는 모듈을 수정할 수 있다.			
빅데이터 이벤트 처리 수행 모듈 개발	- 스트림데이터의 특정 패턴에 대한 쿼리를 작성할 수 있다.			
	- 이벤트 감지와 처리를 하기 위한 규칙을 정의하고, 규칙에 따라 이벤트 처리를 수행할 수 있는 모듈을 작성할 수 있다			
	- 주어진 테스트 절차에 의해서 데이터 이벤트 처리 완료 및 성공 여부를 테스트하고, 실패했을 때 문제점을 파악하고 처리하는 모듈을 수정할 수 있다.			

평가 방법

- 문제해결 시나리오

학습 내용	평가 항목	성취수준		
		상	중	하
분산 처리 수행 모듈 개발	- 분산 처리 모델에 따라 대규모 데이터를 병렬 처리하는 분산 처리 수행 모듈을 작성할 수 있는 능력			
	- 분산 처리를 수행하기 위하여, 스크립트를 작성하고 등록할 수 있는 능력			
	- 주어진 테스트 절차에 의해서 데이터 분산 처리 완료 및 성공 여부를 테스트하고, 실패했을 때 문제점을 파악하고 처리하는 모듈을 수정할 수 있는 능력			
실시간 수행 모듈 개발	- 실시간 처리를 위하여 스키마, 컬럼, 조건, 처리 방식을 결정하고 실시간 수행 모듈을 작성할 수 있는 능력			
	- 실시간 처리 중 데이터 정합성 유지를 위한 로깅, 검증, 예외 처리 공통 모듈을 작성할 수 있는 능력			
	- 주어진 테스트 절차에 의해서 데이터 실시간 처리 완료 및 성공 여부를 테스트하고, 실패했을 때 문제점을 파악하고 처리하는 모듈을 수정할 수 있는 능력			
이벤트 처리 수행 모듈 개발	- 스트림 데이터의 특정 패턴에 대한 쿼리를 작성할 수 있는 능력			
	- 이벤트 감지와 처리를 하기 위한 규칙을 정의하고, 규칙에 따라 이벤트 처리를 수행할 수 있는 모듈을 작성할 수 있는 능력			
	- 주어진 테스트 절차에 의해서 데이터 이벤트 처리 완료 및 성공 여부를 테스트하고, 실패했을 때 문제점을 파악하고 처리하는 모듈을 수정할 수 있는 능력			

- 평가자 체크리스트

학습 내용	평가 항목	성취수준		
		상	중	하
분산 처리 수행 모듈 개발	- 분산 처리 모델에 따라 대규모 데이터를 병렬 처리하는 분산 처리 수행 모듈을 작성할 수 있는 능력			
	- 분산 처리를 수행하기 위하여, 스크립트를 작성하고 등록할 수 있는 능력			
	- 주어진 테스트 절차에 의해서 데이터 분산 처리 완료 및 성공 여부를 테스트하고, 실패했을 때 문제점을 파악하고 처리하는 모듈을 수정할 수 있는 능력			
실시간 수행 모듈 개발	- 실시간 처리를 위하여 스키마, 컬럼, 조건, 처리 방식을 결정하고 실시간 수행 모듈을 작성할 수 있는 능력			
	- 실시간 처리 중 데이터 정합성 유지를 위한 로깅, 검증, 예외 처리 공통 모듈을 작성할 수 있는 능력			

학습 내용	평가 항목	성취수준		
		상	중	하
이벤트 처리 수행 모듈 개발	종, 예외 처리 공통 모듈을 작성할 수 있는 능력			
	- 주어진 테스트 절차에 의해서 데이터 실시간 처리 완료 및 성공 여부를 테스트하고, 실패했을 때 문제점을 파악하고 처리하는 모듈을 수정할 수 있는 능력			
	- 스트림데이터의 특정 패턴에 대한 쿼리를 작성할 수 있는 능력			
	- 이벤트 감지와 처리를 하기 위한 규칙을 정의하고, 규칙에 따라 이벤트 처리를 수행할 수 있는 모듈을 작성할 수 있는 능력			
	- 주어진 테스트 절차에 의해서 데이터 이벤트 처리 완료 및 성공 여부를 테스트하고, 실패했을 때 문제점을 파악하고 처리하는 모듈을 수정할 수 있는 능력			

피드백

1. 문제해결 시나리오
 - 분산 처리 수행 모듈 개발에 필요한 사전 환경, 스크립트 작성 방법, 실행 결과를 확인하는 방법 등에 대해서 평가하고, 성취 수준이 낮은 학습자들에게는 주요 절차를 다시 설명해 준다.
 - 실시간 수행 모듈 개발에 필요한 프로그램 작성 방법, 분산 환경에서 실행 방법, 테스트 성공 여부 등에 대해서 평가하고, 성취 수준이 낮은 학습자들에게는 주요 절차를 다시 설명해 준다.
 - 이벤트 처리 수행 모듈 개발에 필요한 통합 개발 환경 구성 방법, 룰 스크립트 작성 방법, 룰 엔진의 이벤트 탐지 결과 등에 대해서 평가하고, 성취 수준이 낮은 학습자들에게는 주요 절차를 다시 설명해 준다.
2. 평가자 체크리스트
 - 학습자가 분산 처리 수행 모듈을 개발한 과정을 체크리스트로 평가한 후 미흡한 부분에 대해서 정확한 설치 절차를 설명해 준다.
 - 실습 과정에서 학습자가 개발한 수행 모듈의 개선할 사항 등을 정리한 후 학습자에게 피드백해 준다.
 - 학습자의 수행 모듈의 아이디어나, 기능의 차별성이 높은 것들은 학습자들에게 공유하여 서로의 의견을 논의할 수 있도록 한다.

참고자료



- 김강원(2017). 『실무로 배우는 빅데이터 기술』. 위키북스.
- 사루다 고스케 · 도바시 마사루 · 요시다 고요 · 사사키 도루 · 쓰즈키 마사요(2017). 『아파치 스파크 입문』. 김진용(역). 한빛미디어.
- 서형석.(2019). 『 빅데이터 - 하둡, 하이브로 시작하기』. 위키북스.
- 최중현 · 조은선 · 이강우(2015). 「복합 이벤트 스트림 질의 처리 성능 개선을 위한 질의 전 처리 도구」, 『정보과학회』, 21(8).
- SW공학센터(2014). 「SW아키텍처 참조 모델, 빅데이터 플랫폼 설계 및 구현」.
- Reference. Spark Shell Commands to Interact with Spark-Scala. Retrieved from <https://data-flair.training/blogs/scala-spark-shell-commands/>
- Reference. Get started on Hadoop with these tutorials based on the Hortonworks Sandbox. Retrieved from <https://hortonworks.com/tutorial/how-to-process-data-with-apache-hive/>
- Apache Hadoop 공식
홈페이지(<https://hadoop.apache.org/docs/current/hadoop-yarn/hadoop-yarn-site/YARN.html>). 2019. 08. 20. 스크린샷.
- Apache Spark 공식 홈페이지(<https://spark.apache.org>). 2019. 08. 09. 스크린샷.
- Hortonworks 공식 홈페이지(<https://ko.hortonworks.com>). 2019. 08. 09. 스크린샷.
- Hortonworks 공식 다운로드
페이지(<https://www.cloudera.com/downloads/hortonworks-sandbox/hdp.html>). 2019. 08. 09. 스크린샷.
- Putty 공식 다운로드
페이지(<https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html>). 2019. 08. 09. 스크린샷.
- VirtualBox 공식 홈페이지(<https://www.virtualbox.org/wiki/Downloads>). 2019. 08. 09. 스크린샷.

NCS학습모듈 개발이력

발행일	2019년 12월 31일
세분류명	빅데이터플랫폼구축(20010209)
개발기관	(사)한국정보통신기술사협회, 한국직업능력개발원
	박미화((주)투이컨설팅)* 김준범(SK주식회사)
	김영옥(현대자동차) 유병곤((주)안세기술)
	박상원(SK주식회사) 정종호(삼일공업고등학교)
	송창용(한솔인티큐브) 최남대(한국전산감리원)
집필진	이상아(한국정보통신기술사협회)
	검토진
	이주희((주)유알피시스템)
	임세훈((주)케이티디에스)
	하석재(주식회사 투에이치큐브)

*표시는 대표집필자임

빅데이터 처리 시스템 개발(LM2001020905_17v1)

저작권자 교육부

연구기관 한국직업능력개발원

발행일 2019. 12. 31.

※ 이 학습모듈은 자격기본법 시행령(제8조 국가직무능력표준의 활용)에 의거하여 개발하였으며, NCS통합포털사이트(<http://www.ncs.go.kr>)에서 다운로드 할 수 있습니다.



www.ncs.go.kr