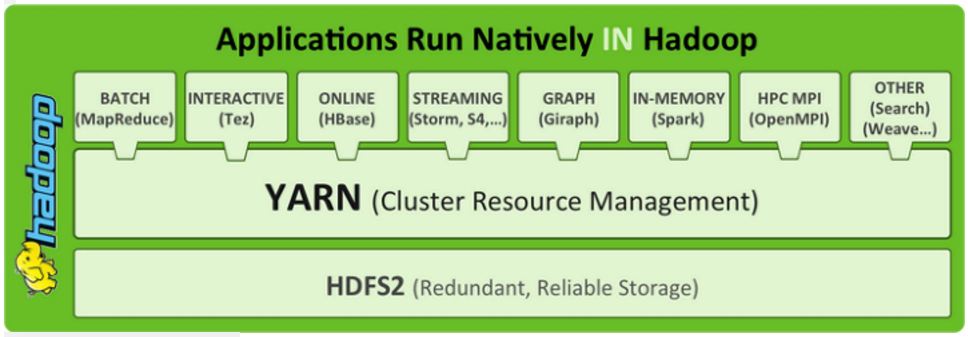
03-01 Yarn

2020년 7월 24일 금요일 오전 11:20

https://nive.tistory.com/218 참조(❤️ * 100)

아파치 하둡 얀은 리소스 관리와 컴포넌트 처리를 분리한 하둡 2.0에 도입된 아파치 소프트웨어 재단의 서브 프로젝트입니다. 얀은 맵-리듀스의 차세대 기술로서 **맵-리듀스의 확장성과 속도문제를 해소**하기 위해 새로 개발된 프로젝트입니다. 얀 이전의 맵-리듀스에서는 4000노드 이상의 클러스터에서 시스템 확장에 문제가 발생하여 야후(Yahoo)에서 새로 설계하였습니다.

가장 큰 변화로는 얀 자체로 맵-리듀스를 구동할 수 있으며, 추가로 다른 분산 처리 프레임워크(Tez, HBase, Giraph, Spark 등)를 사용자의 인터페이스 개발만으로 구동이 가능하게 되었습니다.



[하둡 2.0 스택]

얀(YARN)은 하둡 분산 파일 시스템(HDFS, Hadoop Distributed File System)의 상단에서 빅 데이터용 애플리케이션들을 실행하는 대용량 분산 운영체제 역할을 수행합니다. 얀(YARN)을 이용하면 안정적인 기반에서 배치 작업과 양방향 실시간 작업을 수행할 수 있습니다. 아파치 재단은 얀(YARN)을 '맵-리듀스 버전 2(MRv2)'로 명명했습니다.

YARN의 구조

얀(YARN)은 크게 리소스 매니저(Resource Manager), 노드 매니저(Node

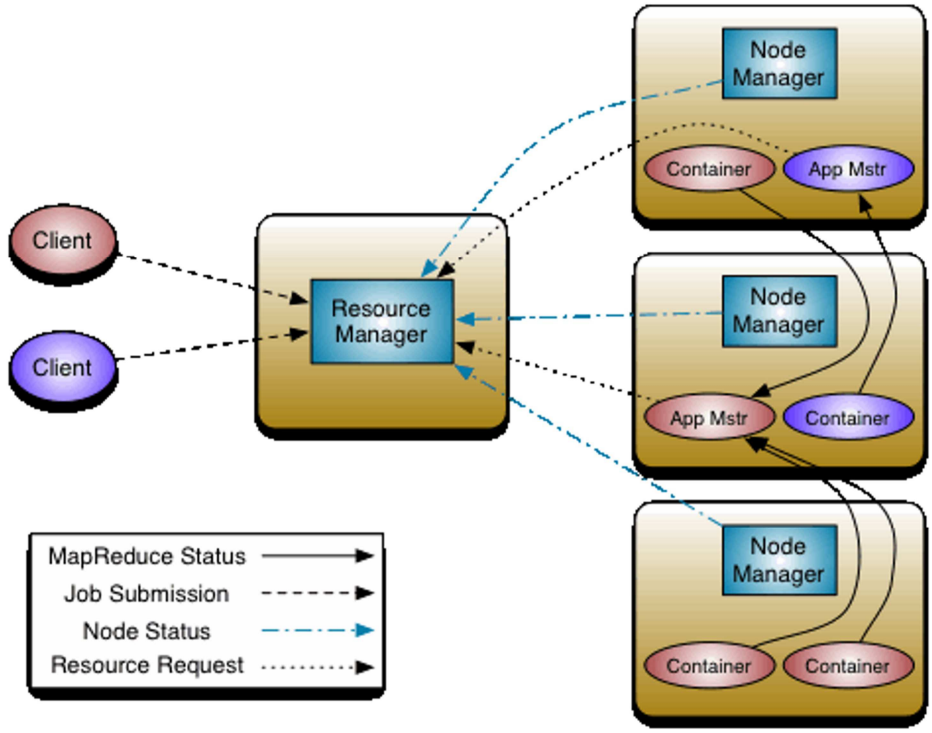
Manager), 애플리케이션 마스터(Application Master), 컨테이너(Container)로 구성되어 있습니다.

① 리소스 매니저(Resource Manager)
클러스터 전체를 관리하는 마스터 서버의 역할을 담당하며, 응용 프로그램의 요청을 처리합니다. 리소스 매니저는 클러스터에서 발생한 작업을 관리하는 애플리케이션 매니저(Applications Manager)를 내장하고 있으며, 응용 프로그램들 간의 자원(resource) 사용에 대한 경쟁을 조율합니다. 여기서 말하는 자원이란 CPU, 디스크(disk), 메모리(memory)등을 의미합니다.

② 노드 매니저(Node Manager)
노드당 하나씩 존재하며, 슬레이브 노드(slave node)의 자원을 모니터링(monitoring) 하고 관리하는 역할을 수행합니다. 노드 매니저는 리소스 매니저의 지시를 받아 작업 요구사항에 따라서 컨테이너를 생성합니다.

③ 애플리케이션 마스터(Application Master)
노드 매니저와 함께 번들로 제공되며, 작업당 하나씩 생성이 되며, 컨테이너를 사용하여 작업 모니터링과 실행을 관리합니다. 또한, 리소스 매니저와 작업에 대한 자원 요구사항을 협상하고, 작업을 완료하기 위한 책임을 가집니다.

④ 컨테이너(Container)
CPU, 디스크(Disk), 메모리(Memory) 등과 같은 속성으로 정의됩니다. 이 속성은 그래프 처리(Graph processing)와 MPI(Message Passing Interface: 분산 및 병렬 처리에서 정보의 교환에 대해 기술하는 표준)와 같은 여러 응용 프로그램을 지원하는데 도움이 됩니다. 모든 작업(job)은 결국 여러 개의 작업(task)으로 세분화되며, 각 작업(task)은 하나의 컨테이너 안에서 실행이 됩니다. 필요한 자원의 요청은 애플리케이션 마스터(Application Master)가 담당하며, 승인 여부는 리소스 매니저(Resource Manager)가 담당합니다. 컨테이너 안에서 실행할 수 있는 프로그램은 자바 프로그램뿐만 아니라, 커맨드 라인에서 실행할 수 있는 프로그램이면 모두 가능합니다.



[YARN의 전체 구성도]

리소스 매니저(Resource Manager)의 구조

리소스 매니저는 클러스터마다 존재하며, 클러스터 전반의 자원 관리와 작업(task)들의 스케쥴링을 담당합니다. 리소스 매니저는 클라이언트로부터 애플리케이션 실행 요청을 받으면 그 애플리케이션의 실행을 책임질 애플리케이션 마스터(Application Master)를 실행합니다. 또한 클러스터 내에 설치된 모든 노드 매니저(Node Manager)와 통신을 통해서 각 서버마다 할당된 자원과 사용중인 자원의 상황을 알 수 있으며, 애플리케이션 마스터들과의 통신을 통해 필요한 자원이 무엇인지 알아내어 관리하게 됩니다.

리소스 매니저(Resource Manager) 내부에는 여러 개의 컴포넌트들이 존재하며, 스케쥴러(Scheduler), 애플리케이션 매니저(Application Manager), 리소스 트랙커(Resource Tracker) 세개의 메인 컴포넌트가 있습니다.

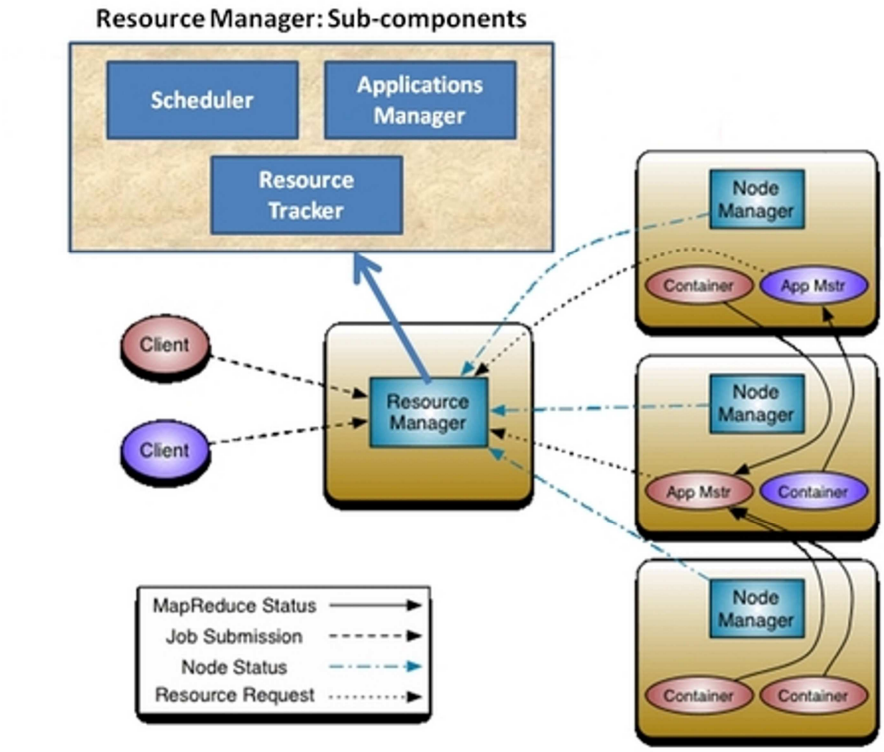
① 스케쥴러(Scheduler)

노드 매니저(Node Manager)들의 자원 상태를 관리하며 부족한 리소스들을 배정합니다. 스케쥴러는 프로그램의 상태를 검사하거나 모니터링 하지 않으며, 순수하게 **스케쥴링 작업**만 담당합니다. 스케쥴링이란 **자원 상태에 따라서 작업(task)들의 실행 여부를 허가해 주는 역할**만 담당하며, 그 이상의 책임은 지지 않습니다. 즉, 프로그램 오류나 하드웨어의 오류로 문제가 발생한 프로그램을 재 시작시켜주지 않으며, 프로그램에서 요구하는 리소스(CPU, Disk, 네트워크등)에 관련된 기능만 처리합

니다.

② 애플리케이션 매니저(Application Manager)
노드 매니저(Node Manager) 에서 특정 작업을 위해서 애플리케이션 마스터(Application Master)를 실행하고, **애플리케이션 마스터(Application Master)의 상태를 관리**합니다. 여기서 애플리케이션 마스터(Application Master)라는 용어가 나오는데 얀에서 실행되는 **하나의 작업(task)을 관리하는 마스터 서버**를 말합니다.

③ 리소스 트랙커(Resource Tracker)
컨테이너(Container)가 아직 살아 있는지 확인하기 위해서, 애플리케이션 마스터(Applications Master) 재 시도 최대 횟수, 그리고 노드 매니저(Node Manager)가 죽은 것으로 간주 될 때까지 얼마나 기다려야 하는지 등과 같은 **설정 정보**를 가지고 있습니다.



[리소스 매니저: 서브 컴포넌트]

노드 매니저(Node Manager)의 구조
노드 매니저는 **노드(컴퓨터) 당 한 개씩 존재**합니다. 해당 컨테이너(Application Container)의 리소스 사용량을 모니터링 하고, 관련 정보를 리소스 매니저에게 알리는 역할을 담당합니다. 애플리케이션 마스터(Application Master)와 애플리케이션 컨테이너(Application Container)로 구성되어 있습니다.

① 애플리케이션 마스터(Application Master)

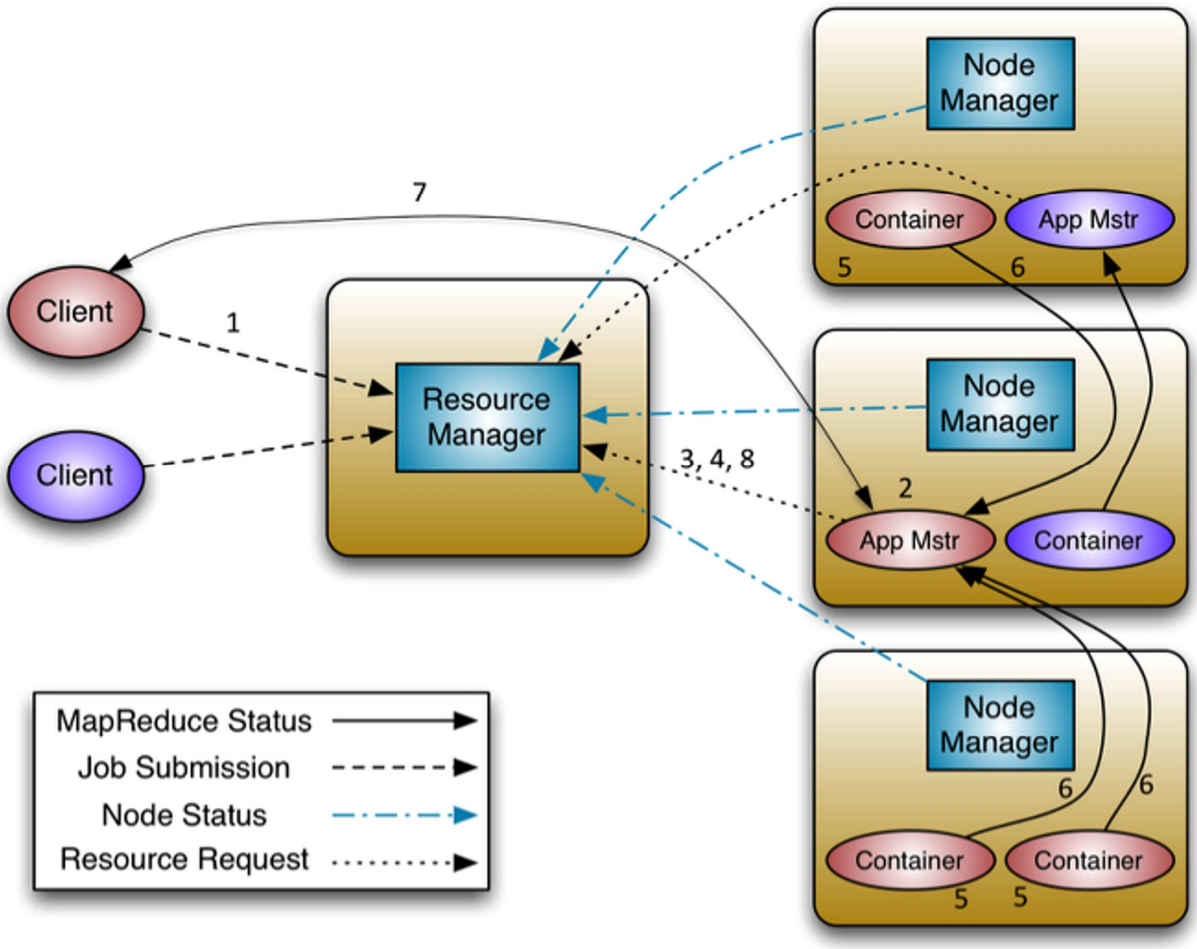
하나의 프로그램에 대한 마스터 역할을 수행하며, 스케쥴러로부터 적절한 애플리케이션 컨테이너(Application Container)를 할당 받고, 프로그램 실행 상태를 모니터링하고 관리합니다.

② 애플리케이션 컨테이너(Application Container)

프로그램에 할당된 자원을 나타냅니다.

얀의 작업 순서

다음 그림은 얀 클러스터(YARN Cluster)에서 응용 프로그램이 실행되는 과정을 보여줍니다.



[얀의 작업 순서]

① 클라이언트는Application Master 자체를 실행하는 필요한 데이터를 포함하는 응용프로그램을 제출한다.

② Resource Manager는 Container 할당을 책임지는 Application Master을 시작합니다.

③ Application Master가 Resource Manager에 등록되고 클라이언트가 Resource Manager과 통신할 수 있게 된다.

④ Application Master는 resource-request프로토콜을 통해 Resource Manager에게 적절한 리소스의 Container 를 요청한다.

⑤ Container 가 성공적으로 할당되면, Application Master는 Container 실행 스펙을 Node Manager에게 제공하여 Container 를 실행시킨다. 실행 스펙은 Container가 Application Master와 통신하기 위해 필요한 정보를 포함하고 있다.

⑥ 응용프로그램 코드는 Container 에서 실행되고 진행률, 상태 등의 정보를 응용프로그램-스펙 프로토콜을 통해 Application Master 에게 제공한다.

⑦ 클라이언트는 응용프로그램 실행 중의 상태, 진행률 등을 얻기 위해 응용프로그램-스팩 프로토콜을 통해 Application Master와 직접 통신한다.

⑧ 응용프로그램이 완료되고 모든 필요한 작업이 종료되면, Application Master는 Resource Manager의 등록을 해제하고 자신의 컨테이너를 다른 용도로 사용할 수 있도록 종료한다.