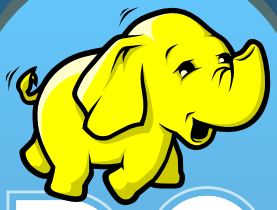


HADOOP

BUYER'S GUIDE

운영 환경을 위한 최적의 Hadoop
배포 프로그램을 선택하는 방법

Robert D. Schneider
Hadoop for Dummies 저자

ubuntu
Supported by Canonical

소개
빅 데이터, MapReduce 및 Hadoop
MapReduce의 탄생
Hadoop의 시작
Hadoop 관련 프로젝트
Hadoop 인프라 선택의 중요성
Hadoop 플랫폼 선택 시 주요 고려 사항
성능 및 확장성
성능 및 확장성을 위한 아키텍처 요구 사항
쓰기 스트리밍
확장성
실시간 NoSQL
신뢰성
신뢰성을 위한 아키텍처 요구 사항
고가용성
데이터 보호
재해 복구
관리 용이성
데이터 액세스
데이터 액세스를 위한 아키텍처 요구 사항
표준 파일 시스템 인터페이스 및 의미 체계(POSIX)
개발 도구
보안
주요 Hadoop 배포 프로그램 비교
간단 비교표
저자 소개

소개

핵심 비즈니스 업무를 위한 주요 애플리케이션의 필수 구성 요소로 Hadoop을 선택하는 기업이 점점 증가하고 있습니다. Hadoop은 더 이상 "과학 프로젝트" 혹은 주 업무와 분리된 “실험적 프로젝트”를 위한 플랫폼으로 간주되지 않습니다. 따라서 이제는 다른 주요 기술 제품과 마찬가지로 Hadoop 플랫폼도 신중한 선택이 요구됨을 의미합니다.

이 구매자 가이드는 향후 몇 년간 조직을 지탱해 줄 Hadoop 인프라를 선택할 때 참조할 수 있는 일련의 가이드라인을 제시합니다. 실제로 이 가이드는 Hadoop 플랫폼 평가 시 RFP에 포함시킬 수 있도록 작성되었습니다.

이 가이드에서는 먼저 배경 지식으로 빅 데이터, MapReduce 및 Hadoop에 대해 살펴본 다음, Hadoop 플랫폼의 선택이 중요한 이유에 대해 알아보겠습니다. 특히 다음과 같은 사항을 중심으로 각 Hadoop 옵션(지원하는 운영 체제 포함)을 평가하는 데 도움이 되는 몇 가지 권장 사항을 제시합니다.

- 성능 및 확장성
- 신뢰성
- 관리 용이성
- 데이터 액세스

이 가이드는 IT 리더, 데이터베이스 아키텍트, 시스템 관리자, 소프트웨어 개발자, 비즈니스 분석가 등, 조직에서 성공적인 빅 데이터를 구현하는 데 관련된 모든 사용자를 위해 작성되었습니다.

빅 데이터, MapReduce 및 Hadoop

Hadoop 플랫폼 선택 시의 고려 사항에 대해 알아보기 전 먼저 몇 가지 배경 지식에 대해 잠시 살펴보겠습니다. 5명에게 빅 데이터의 정의에 대해 물어보면 서로 다른 10가지의 답변을 듣게 될 것이며, 각 답변이 모두 그럴듯하게 들릴 것입니다.

말하자면 빅 데이터는 다음과 같이 어느 하나로 단정지을 수 없는 여러 특성을 가지고 있기 때문입니다.

- 대량의 정보 처리
- 다양한 데이터 형태 및 타입의 집합
- 다양한 생성 소스
- 장기간 보관
- 새로운 혁신적인 애플리케이션 지원

이러한 각 특성은 각기 다른 당면 과제를 제기하며, 빅 데이터에 내재되어 있는 모든 이점을 실현하려면 먼저 IT는 물론 전체 조직 차원에서 이러한 과제를 해결해야 합니다.

대량의 정보 처리. 한때는 1기가바이트의 데이터도 엄청난 것으로 간주되었지만, 이제는 테라바이트 혹의 페타바이트의 데이터에 대해서도 쉽게 말합니다.

여러 유형 및 형식의 데이터 포함. 정보는 더 이상 행과 열의 엄격히 정형화된 형식으로 한정되지 않습니다. 오늘날의 데이터는 매우 동적이며 정형화되어 있지 않거나 부분적으로 정형화되어 있어 기존 방식으로 처리하기가 매우 어렵습니다.

다양한 생성 소스. 기존의 트랜잭션 애플리케이션에서 생성된 정보를 처리하는 동시에 다음과 같은 새로운 소스에서 생성된 데이터에 대해서도 적응해야 합니다.

- 무선 장치
- 센서
- 행동 지표(예: Clickstreams, query 및 beacon logs)
- 기계 간의 상호 작용에 의해 생성된 통신 스트리밍
- 데이터 잔해: 모든 시스템 및 액세스 로그, 구성 변경 및 기타 컴퓨터 실행의 결과

트랜잭션 시스템의 정보를 처리하여 데이터 웨어하우스에 체계적으로 저장하는 이전의 접근 방식은 이처럼 광범위한 소스에서 대량의 데이터가 생성되는 시대에는 맞지 않습니다. 특히 데이터의 형식도 바뀔 수 있게 된 후로는 더욱 그렇습니다.

장기간 보관. 법률 및 규제 기관의 요구 사항 등으로 인해 이전에는 임시로 보관한 데이터를 이제 수년에서 수십년까지 보관해야 하므로 스토리지에 대한 수요가 폭발적으로 증가하고 있습니다.

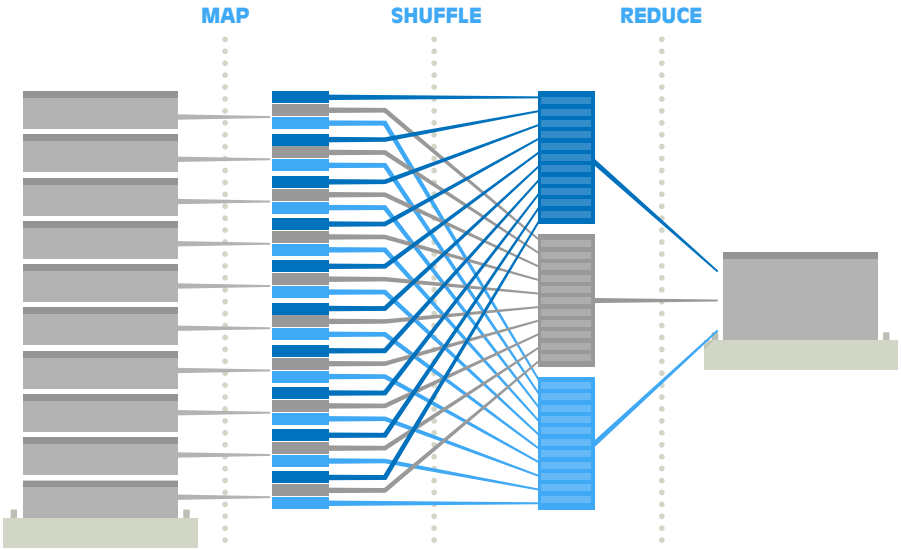
새로운 혁신적인 애플리케이션 지원. 많은 기업이 "데이터를 일단 저장한 후 어떻게 처리할 것인지는 나중에 결정"하는 접근 방식을 따르고 있습니다. 솔직히 빅 데이터에 관련된 모든 새로운 애플리케이션에 대해 IT가 예측하고 계획을 수립하기란 불가능에 가깝습니다. 대부분 데이터를 실험해 본 후에야 정확한 데이터의 가치를 확인할 수 있으므로 가능한 한 모든 데이터를 보관하는 이러한 방식은 매우 좋은 전략이라고 말할 수 있습니다.

MapReduce의 탄생

빅 데이터에 관한 모든 당면 과제를 해결하기 위해서는 장기적으로 대량의 정보를 처리할 수 있는 더 나은 방법이 필요하다는 것만은 분명합니다. 컴퓨터 과학에서는 오랫동안 분할-정복(divided and conquer) 알고리즘이 대량의 정보를 처리하는 데 매우 효과적이라고 알려져왔습니다. 문제는 지금까지의 경험을 통해 광범위한 문제에 대해 분할-정복(divided and conquer)을 적용하는 것이 매우 어렵다는 데 있습니다.

이와 달리 초기 Google 개발자들이 사용한 MapReduce라고 하는 프로그래밍 스타일은 대량의 데이터 처리에 매우 효과적일 뿐만 아니라 광범위한 알고리즘을 표현할 수 있다는 이점이 있습니다. 2004년, Google 개발자들은 이 프로그래밍 방법과 그 결과에 대해 설명한 논문을 발표했습니다.

이 논문에서 Google은 MapReduce를 이렇게 정의합니다. “MapReduce는 대량 데이터 조합의 처리 및 생성을 위한 프로그래밍 모델이자 관련 구현 방법입니다. 사용자는 키/값(key/value)의 쌍을 처리하여 중간 키/값(key/value) 쌍의 조합을 생성하는 map 함수 및 동일한 중간 키를 가진 모든 중간 값을 병합하는 reduce 함수를 지정합니다.”



MapReduce

실제 구현된 MapReduce는 상용 하드웨어 및 소프트웨어, 전용 기본 파일 시스템, 병렬 처리 방식이 포함된 상호 보완적 기술과 전략의 조합이라고 할 수 있습니다. 데이터가 상주하고 있는 시스템과 동일한 시스템에서 컴퓨팅이 수행되며, 하드웨어 장애 또는 기타 지연으로 인해 필요할 경우 전체 컴퓨팅의 개별 요소별로 재컴퓨팅된다는 점에서 많은 이점이 있습니다.

이는 아키텍처 면에서 매우 혁신적인 방식으로, 이전에 일반적인 개발자들이 병렬 처리를 구현하는 데 필요했던 엄청나게 복잡한 작업이 더 이상 필요 없게 되었습니다. 그러나 나중에 살펴보겠지만 MapReduce를 실제로 구현하는 과정에서 많은 기업이 몇 가지 큰 어려움에 봉착하게 됩니다.

HADOOP의 시작

기본적으로 MapReduce는 매우 간단 명료한 아키텍처지만, 올바르게 구현하려면 Google과 같이 체계적으로 조직된 대규모의 강력한 리소스를 필요로 합니다. 무엇보다 MapReduce는 기존 코드를 모두 버리고 대부분의 개발자에게 완전히 낯선 새로운 스타일로 처음부터 코드를 다시 작성해야 한다는 문제가 있습니다. 이러한 코드 표현 방식에 따른 제약은 MapReduce의 확장성을 위해 반드시 필요하지만 프로그래머를 교육하는 데 많은 어려움이 있습니다. 또한 구현 버전이 제공되지 않으므로 MapReduce를 도입할 경우 프로그램을 작성하기 전 먼저 전체 프레임워크를 구현해야 합니다.

다행히도 Doug Cutting 및 Mike Cafarella라고 하는 2명의 진취적인 엔지니어가 Nutch라고 하는 웹 크롤링 기술을 개발하는 과정에서 Google의 연구 논문에 영감을 받아 나중에 Hadoop이라고 명명되는 플랫폼의 토대를 만들기 시작했습니다.

이후 Cutting은 Yahoo!에 입사하여 Hadoop을 더욱 확장하는 데 기여했습니다. Hadoop이 더욱 정교해짐에 따라 Yahoo!는 일상 업무의 상당 부분을 수행하는 중요 애플리케이션 플랫폼으로 Hadoop의 활용 범위를 크게 확대했습니다. 2008년 초, Apache Software Foundation(ASF)은 Hadoop의 가치를 인정하고 이것을 최우선 오픈 소스 프로젝트로 격상시켰습니다.

원래 Hadoop은 Java로 구현되었으며 Linux 파일 시스템을 사용하여 데이터를 저장했습니다. 이러한 결정은 최초 구현 버전인 Nutch에서는 매우 효과적이었지만 기본 아키텍처로 Hadoop을 사용하려는 기업에게는 이로 인해 몇 가지 문제가 발생합니다.

그러나 전반적으로 Hadoop은 매우 성공적이었습니다. 아래에 몇 가지 주요 운영 분야의 활용 사례가 나와 있습니다.

> 많은 기업에서 이미 Hadoop에 데이터 웨어하우스 기반 데이터를 오프로드하고 있습니다. 일반적으로 Hadoop은 데이터 웨어하우스 솔루션 대비 10배 이상의 비용이 절감되는 효과가 있습니다.

금융 기관들은 보안 아키텍처의 핵심 요소로 Hadoop을 사용하여 실시간으로 피싱 행위 및 결제 사기를 예방하고 그 피해를 최소화하고 있으며, 데이터를 장기간 보관하고 보다 정교한 분석 및 forensics을 실행합니다.

한 온라인 광고 회사는 사용자에게 실시간 거래 기술을 제공하며 Hadoop을 활용해 페타바이트급 데이터를 저장 및 분석합니다. 또한 Hadoop 배포 프로그램으로 매일 900억 건의 실시간 광고 입찰을 처리합니다.

한 디지털 마케팅 정보 회사는 Hadoop을 사용해 1조7천억 개 이상의 인터넷 및 모바일 레코드를 매달 처리하여 배급식 및 맞춤형 디지털 마케팅 정보를 제공합니다.

기업은 적합한 Hadoop배포 프로그램을 사용함으로써 실시간으로 소비자에게 새로운 제품과 서비스를 제공할 수 있으며, 가용성 높은 Hadoop 클러스터에 저장된 데이터에 고급 기계 학습 및 통계 기법을 적용할 수 있습니다.

HADOOP 관련 프로젝트

Hadoop은 또한 다양한 하위 프로젝트에 대해 완벽한 환경을 제공합니다. 다음은 Apache 프로젝트의 몇 가지 예입니다.

워크플로우 예약	리소스 예약	데이터 상호 작용 및 분석
Oozie	YARN	Pig
데이터베이스	**데이터 웨어하우징**	Drill
HBase	Hive	**추출, 변환 및 로드(ETL)**
확장형 기계 학습		Flume
Mahout		Sqoop
		Scribe

HADOOP 인프라 선택의 중요성

많은 IT 조직에서 Hadoop 환경의 구성 요소를 확보, 구축 및 통합하는 데 어려움을 겪고 있으며, 그 결과 기본 비즈니스 업무를 수행하기보다 Hadoop 환경을 관리하는 데 더 많은 시간을 할애해야 합니다. 또한 Hadoop은 물론, 모든 Hadoop 관련 프로젝트도 지속적으로 변화함에 따라 이는 더욱 심각한 문제를 야기합니다.

더욱 쉽고 간편하게 Hadoop 환경을 완벽하게 구현할 수 있도록 몇몇 업체에서 종합적인 배포 프로그램을 제공하고 있으며 다음 3가지 모델 중 하나로 특징지을 수 있습니다.

1. **오픈 소스 Hadoop 및 지원.** 기본 뼈대인 오픈 소스에 유료 전문 지원과 서비스가 결합된 모델입니다. 이 전략의 예로 Hortonworks가 있습니다.
2. **오픈 소스 Hadoop, 지원 및 관리 혁신.** 오픈 소스 Hadoop에 IT 친화적인 도구와 유틸리티를 추가하여 IT 조직의 업무를 간소화합니다. 이 모델의 예로는 Cloudera가 있습니다.
3. **오픈 소스 Hadoop, 지원 및 아키텍처 혁신을 통한 가치 추가.** 먼저 파일 시스템 수준의 구성 요소로 Hadoop 아키텍처를 제공합니다. 그런 다음, 혁신을 위해 하나 이상의 구성 요소를 교체하고 나머지 오픈 소스 구성 요소를 패키징하여 Hadoop과의 호환성을 유지합니다. 이 모델의 예로는 MapR이 있습니다.

Hadoop 배포 프로그램의 3가지 모델

Hadoop 플랫폼 선택 시 주요 고려 사항

Hadoop 배포 프로그램의 도입은 조직 전반에 영향을 미치는 매우 중요한 결정이며, 초기 평가 기준을 수립할 때 관련된 모든 항목을 반영하기란 쉽지 않습니다. 특히 빅 데이터가 기업 환경에서 아직 초기 단계라는 점에서 더욱 그렇습니다.

Hadoop도 하나의 인프라 구성요소로써 애플리케이션 서버, 스토리지 및 데이터베이스와 같은 주요 자산을 선택할 때와 동일한 수준의 주의 및 감시가 필요합니다. Hadoop 환경 또한 다음과 같은 면에서 IT 포트폴리오에 포함된 다른 요소와 동일한 요구 사항을 충족해야 합니다.

- SLA
- 데이터 보호
- 보안
- 다른 애플리케이션과의 통합
- 전문 서비스
- 교육

먼저 Hadoop을 단일 솔루션이 아닌 여러 애플리케이션을 실행하는 플랫폼으로 간주해야 하며, 각 애플리케이션은 서로 함께 작동하여 가치를 극대화해야 합니다. 또한 특정 Hadoop 기술을 선택하여 강제로 적용하기 보다는 기업에서 비즈니스를 운영하는 방식에 적합한 솔루션을 찾아야 합니다.

다른 기본 소프트웨어 제품을 선택할 때와 같이 다음의 가이드라인을 사용하여 RFP를 작성하시기 바랍니다. 이해를 돕기 위해 다음 4가지 주요 카테고리로 분류했습니다.

- 성능 및 확장성
- 신뢰성
- 관리 용이성
- 데이터 액세스

지금부터 각 권장 사항에 대한 의미와 Hadoop 배포 프로그램에서 살펴보아야 할 사항을 설명하겠습니다. 또한 여러 예를 통해 이러한 기능이 실제 상황에서 어떤 부가 가치를 제공하는지에 대해 알아보겠습니다.

성능 및 확장성

초기 Hadoop은 현재의 주요 활용 사례와 비교하여 상대적으로 덜 민감한 웹의 크롤링과 색인 작업에 주로 사용되었습니다. 그러나 이제는 real-time 또는 Near real-time으로 유용한 결과를 제공해야 하는 작업과 관련된 Hadoop 프로젝트가 갈수록 증가하고 있습니다. 따라서 Hadoop의 성능에 대한 정의도 보다 엄격하게 바뀌었습니다. 초기에는 빠른 처리량이 기본 요건이었다면 지금은 여기에 빠른 응답 속도가 포함됩니다.

빠른 응답 속도가 중요해짐에 따라 Hadoop 플랫폼의 다음 두 가지 주요 속성이 주목을 받고 있습니다.

성능. 정보의 신속한 처리부터 해당 데이터를 즉시 분석에 사용할 수 있는지 여부, 실시간 애플리케이션의 응답 속도와 MapReduce 속도까지 모든 것을 의미합니다.

확장성. 노드와 테이블, 파일의 수 등, 관련된 모든 요소를 손쉽게 확장할 수 있는지를 나타냅니다. 또한 과도한 관리 및 비용 부담, 애플리케이션 로직에 대한 변경 등이 없어야 합니다.

이 섹션은 Hadoop 구현 환경의 실행 – 성능 및 확장과 직접적으로 연관된 여러 가지 기능을 설명하고 있습니다.

성능 및 확장성을 위한 아키텍처 요구 사항

Hadoop 환경에서 제공해야 할 기능은 표 1을 참조하십시오. 이 표에는 성능과 확장성에 영향을 미칠 수 있는 몇 가지 주요 아키텍처 전제 조건이 항목별로 정리되어 있습니다.

전제 조건	중요한 이유
C/C++와 같은 시스템 언어로 주요 구성 요소 구축	Apache 오픈 소스 Hadoop 배포 프로그램은 Java로 작성되어 이식성이 우수하지만, Java 자체의 오버헤드와 이미 알려진 garbage collection의 예측 불가능성 및 응답 속도로 인해 다른 프로그래밍 언어와 비교하여 성능이 저하될 수 있는 Java 관련 문제가 발생합니다. 다른 엔터프라이즈급 소프트웨어와 같이 성능 및 안정성을 위해 c/c++을 사용하는 것이 좋습니다.
소프트웨어 계층 최소화	일반적으로 시스템은 "구성 요소"의 수가 적을수록 좋습니다. Hadoop 및 관련 Apache HBase 프로젝트의 경우 HBase Master 및 RegionServer, Java 가상 머신 및 로컬 Linux 파일 시스템과 같은 일련의 분리된 계층 탐색에 따른 비효율성으로 인해 성능과 신뢰성이 떨어질 수 있습니다.
모든 빅 데이터 애플리케이션에 대한 단일 환경/플랫폼	주로 성능 문제로 인해 많은 Hadoop 구현 환경의 경우 관리자가 별도의 인스턴스를 만들어야 합니다. Hadoop 기술이 고객에게 필요한 모든 워크로드에 대해 충분한 처리량을 제공한다면 훨씬 더 효과적일 것입니다. 또한 데이터의 중복이 제거/감소되어 확장성이 향상될 것입니다.
주요 퍼블릭 클라우드 플랫폼의 유연성과 확장성 활용	개발자와 관리자 모두 방화벽, 그리고 Amazon Web Services 및 Google Compute Engine과 같은 주요 클라우드 환경 내부에 Hadoop을 실행할 수 있는 유연성을 필요로 합니다.

표 1: 성능 및 확장성 전제 조건

쓰기 스트리밍

일반적으로 Hadoop은 대량의 정보 처리 작업을 수행하므로 데이터의 적재 및 추출 작업이 최대한 효과적으로 수행되어야 합니다. 하지만 많은 Hadoop 배포 프로그램에서는 Flume 및 Scribe와 같은 기술을 사용하며 이는 복잡한 배치 혹은 부분 스트리밍 프로세스를 필요로 하고 있습니다. 더 큰 문제는 데이터의 양이

기가바이트 수준에서 테라바이트 또는 그 이상으로 증가함에 따라 이러한 내부적인 비효율성도 더욱 확대된다는 점입니다.

기존 NAS(Network Attached Storage)처럼 애플리케이션에서 Hadoop 클러스터를 액세스할 수 있는 표준 파일 인터페이스를 활용하는 Hadoop 구현 방법이 더욱 효과적입니다. 이를 통해 별도 데이터 스테이징이 없어도 애플리케이션 서버에서 바로 Hadoop 클러스터에 정보를 쓸 수 있습니다, 또한 Hadoop용 데이터가 도착과 동시에 자동으로 압축될 수 있으며, 다중 동시 병렬 연결을 통해 애플리케이션에 즉시 임의 읽기 및 쓰기 액세스가 제공됩니다. 이러한 즉각적인 상호 작용으로 앞서 설명한 Hadoop 기반의 실시간 의사 결정이 가능합니다.

온라인 게임 회사에서 Hadoop을 사용하여 수백만 명의 사용자와 수십억 건의 이벤트를 추적한다고 가정해 보겠습니다. 게임 사용자는 화면 전환이 빠르므로 매우 짧은 기간에 사용자에게 가상 상품을 소개할 수 있어야 합니다. 이 경우 스트리밍 데이터를 실시간 또는 거의 실시간으로 분석하여 적시에 제안 및 제공할 수 있다면 수익을 향상시킬 수 있을 것입니다. Apache Drill과 같은 프로젝트는 빠른 의사 결정을 제공하도록 설계되어 있지만 그러려면 원시 데이터 자체가 Hadoop 클러스터에 최대한 빨리 도착해야만 합니다.

확장성

IT 조직에서 Hadoop을 활용하려고 할 때 종종 몇 가지 어려운 문제에 직면하게 됩니다. 전보다 더 많은 하드웨어와 리소스가 필요하게 되어 예산 및 관리 시간에 부담이 발생하고, 한정된 컴퓨팅 자산을 최대한 활용하여 빅 데이터의 이점을 극대화할 수 있도록 하기 위해 고심해야 합니다.

확장 가능한 Hadoop 플랫폼은 이러한 요구 사항 간의 균형을 찾아 예산 한도 내에서 보다 쉽게 사용자 요구 사항을 충족할 수 있도록 해 줍니다.

앞서 설명했듯이 특정 Hadoop 인스턴스의 확장성은 어느 한 가지 기준으로 평가할 수 없으며 다음과 같은 여러 요소를 고려해야 합니다.

파일. Hadoop의 기본 아키텍처는 단일 NameNode로 구성됩니다. 이로 인해 Hadoop 클러스터에서 지원하는 파일 수가 비교적 적은 1억 ~ 1억 5천만 개로 제한되며, 파일 메타데이터에 사용할 수 있는 메모리의 양에도 영향을 미칩니다. 소규모 클러스터의 각 데이터 노드에 대한 블록 수의 제약으로 인해 사용 가능한 파일의 수도 제한됩니다. 따라서 분산된 메타데이터 아키텍처를 사용하여 단일 NameNode로 인한 병목 현상을 방지하고, 수십억 또는 수백억 개의 파일 및 테이블로 확장할 수 있는 Hadoop 플랫폼이 필요합니다.

노드의 수. 확장성을 평가할 또 다른 기준은 물리적 노드의 수입니다. CPU 또는 데이터 스토리지 요구 사항에 따라, 선택된 Hadoop 구현 환경을 1,000개 노드 이상으로 확장해야 할 수 있습니다.

노드 용량/집적도. 또한 스토리지 집약적인 사례를 위해서는 더 높은 디스크 집적도의 노드로 확장해야 합니다. 이를 통해 주어진 데이터 양을 저장하는 데 필요한 전체 노드의 수를 줄일 수 있습니다.

실시간 NoSQL

그 어느 때보다 많은 기업이 NoSQL 기반 솔루션을 사용하여 주요 비즈니스 업무를 수행하고 있습니다. 이러한 새로운 애플리케이션을 통해 우수한 신뢰성을 제공하고 RDBMS 기반 솔루션을 새로운 NoSQL 기반 솔루션에 적용하기 위해서는 기존 관계형 데이터베이스에 구축된 애플리케이션에 기대하는 것과 동일한 유형의 엄격한 SLA를 준수하는 것입니다. 예를 들어, NoSQL 솔루션의 응답 시간에 차이가 많을 경우 일관된 빠른 응답 속도가 필요한 핵심 비즈니스 업무에 적합한 솔루션이 아닙니다.

Apache HBase는 Hadoop에서 실행되도록 구현된 키-값 기반 NoSQL 데이터베이스 솔루션으로, 빅 데이터용 스토리지 및 실시간 분석 기능과 함께 Hadoop을 사용한 MapReduce 작업이라는 추가 이점을 제공합니다. 현재 전체 Hadoop 설치의 약 30~40%가 HBase를 사용하고 있는 것으로 추정됩니다. 그러나 Hadoop과의 통합이라는 이점에도 불구하고 HBase는 성능과 신뢰성의 여러 제약으로 인해 아직 기업에서 활발히 사용되지 않고 있습니다.

다행히 다음과 같은 몇 가지 혁신을 통해 대부분의 온라인 애플리케이션과 분석에 대한 엄격한 요구 사항을 충족할 수 있도록 HBase 애플리케이션에도 획기적인 변화가 일어나고 있습니다.

- 전체 계층의 수 감소
- Java 가비지 수집의 필요성 제거
- 수동 사전 분할 테이블의 필요성 제거
- 단일 NameNode가 아닌 전체 클러스터에 분산된 메타데이터
- 압축으로 인한 I/O 폭풍 방지

신뢰성

Hadoop에서도 다른 모든 유형의 엔터프라이즈 소프트웨어 시스템과 동일한 신뢰성을 제공할 수 있어야 합니다. 또한 기존 Legacy 환경을 관리하는 IT 관리자가 Hadoop 구현 환경도 관리할 수 있어야 합니다.

사용자와 관리자 모두에 대한 전반적인 부담을 줄이기 위해 모든 운영 시스템에서 발생하는 다양한 문제에 대처할 수 있어야 성공적인 Hadoop 인프라라고 할 수 있습니다. 또한 자동화된 대처 방법이 많을수록 신뢰성도 더욱 향상됩니다. 이 섹션에서는 가장 까다로운 환경에서 사용하도록 설계된 Hadoop 플랫폼의 여러 가지 특성에 대해 검토해 보겠습니다.

신뢰성을 위한 아키텍처 요구 사항

표 2에는 Hadoop 구현 환경의 신뢰성 향상을 위한 여러 가지 기본 원칙이 나와 있습니다.

전제 조건	중요한 이유
구성 요소의 수 감소	HBase 환경은 RegionServers 또는 HBase Master가 필요 없어야 합니다. Java 가상 머신의 제거 또한 매우 효과적입니다.

전제 조건	중요한 이유
수작업 감소	많은 Hadoop 관리자가 압축, 수동 관리 및 수동 사전 분할 등과 같은 작업 수행으로 인해 어려움을 겪고 있으며, 이러한 작업을 제거하면 전반적인 신뢰성이 향상될 수 있습니다.
데이터 무결성	데이터 무결성은 내부 체크섬, 복제 및 스냅샷과 같은 데이터 보호 기능을 통해 더욱 강화할 수 있으며 이에 대해서는 뒤에서 자세히 설명하겠습니다.
작업 최적화	런타임 환경은 임시 쿼리 등과 같은 소규모 작업이 사전 예약된 대규모 작업에 지장을 주지 않도록 최적화되어야 합니다.

표 2: 신뢰성을 위한 아키텍처 요구 사항

고가용성

고가용성(HA)이란 Hadoop 시스템에 부득이하게 하드웨어, 네트워크 및 기타 대규모의 복잡한 분산 컴퓨팅 환경 특유의 문제가 발생해도 사용자에게 계속 서비스를 제공할 수 있는 기능을 말합니다.

중요한 운영 애플리케이션에 적절한 수준의 가용성을 제공하기 위해서는 Hadoop 환경에 다음과 같은 HA 기능이 구현되어 있어야 합니다.

HA 기본 제공. 무엇보다 중요한 것은 HA 기능을 제공하기 위해 특별한 단계를 수행할 필요가 없이 플랫폼 자체의 기본 기능으로 제공되어야 한다는 점입니다.

메타 데이터. 클러스터에 대한 모든 메타 데이터가 포함된 단일 NameNode는 단일 장애 지점이 되며 HA를 통해 보호해야 합니다. 메타 데이터의 분산 및 장애 복구 기능을 제공하는 솔루션은 HA는 물론, 실제로 지원되는 파일의 수에 제한이 없다는 추가 이점을 제공합니다.

MapReduce HA. HA에 대해 고려해야 할 중요한 사항 중 하나는 장애 시 MapReduce 작업에 미치는 영향입니다. Job 또는 Task Tracker의 장애는 SLA 준수에 영향을 미칠 수 있습니다. 따라서 사용자 개입없이 HA에 의한 MapReduce의 자동화된 장애 복구로 서비스를 계속 제공할 수 있는 기능이 포함되어 있는지 확인해야 합니다.

NFS HA. 이 기능은 높은 처리량과 함께 NFS 기반 데이터의 처리 및 액세스에 대한 복원성을 제공합니다.

다중 장애 시의 복구 시간. 각 Hadoop 배포 프로그램을 차별화하는 요소 중 하나는 다중 장애로부터의 복구 기능을 포함하여 하드웨어, 사용자 또는 애플리케이션 오류 시 파일 복구에 걸리는 시간과 프로세스입니다. 노드 장애 또는 클러스터 재시작 후 파일과 테이블에 액세스하는 데 수초 또는 수분, 혹은 그보다 더 오래 걸리는지 알아야 합니다.

점진적 업그레이드. Hadoop 및 그 보조 기술의 발전에 따라 가동 중단 시간 없이 구현을 업그레이드할 수 있습니다.

데이터 보호

Hadoop을 사용하여 수익에 직접적인 영향을 줄 수 있는 중요한 비즈니스 의사 결정을 수행하는 기업이 증가하고 있습니다. 이에 따라 Hadoop에서 처리하는 데이터를 보호해야 할 필요성이 커지고 있습니다. 복제 및 스냅샷과 같은 검증된 기법은 관계형 데이터의 보호를 위한 기본 구성 요소로 오랫동안 사용되었으며, Hadoop의 정보를 보호하는 데도 중요한 역할을 수행합니다.

복제. 복제를 통해 상용 하드웨어에서 대량 데이터의 분산 처리 수행 시 일어날 수 있는 장애로부터 Hadoop의 데이터를 보호할 수 있습니다. 선택된 플랫폼에서 Hadoop의 파일 조각, 테이블 영역 및 메타데이터에 대해 최소 3번의 복제를 자동으로 수행하고, 최소한 하나의 사본을 다른 랙으로 전송해야 합니다.

스냅샷. Hadoop 스냅샷은 데이터 중복 없는 특정 시점 복구를 통해 사용자와 애플리케이션의 오류에 대한 추가 보장을 제공합니다. 가능하다면 선택한 Hadoop 플랫폼이 성능이나 확장성에 영향을 주지 않으면서 스냅샷이 실제 정보와 동일한 스토리지를 공유하도록 허용해야 합니다. 또한 스냅샷에서 바로

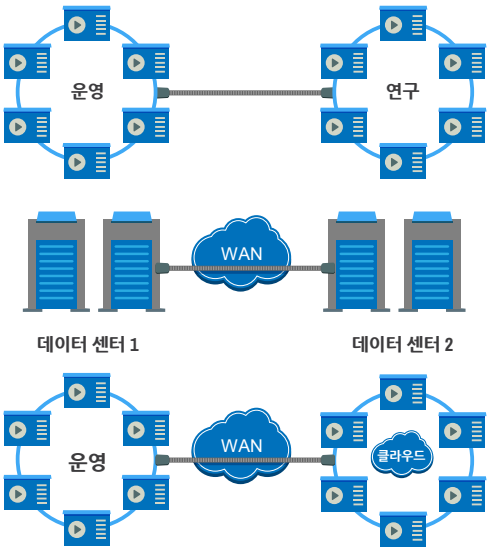
파일과 테이블을 읽을 수 있어야 합니다. 스냅샷은 단순한 데이터 보호, 그 이상의 역할을 수행합니다. 예를 들어, 데이터 과학자는 새 모델을 만드는 프로세스에 스냅샷을 사용할 수 있습니다. 동일한 스냅샷에 대해 서로 다른 모델을 실행하고 모델 변경에 따른 결과만
분리할 수 있습니다.

재해 복구

Hadoop은 특히 다음과 같은 이유로 비즈니스 운영에 심각한 지장을 초래할 수 있는 사고가 일어나기 쉽습니다.

- 일반적으로 상용 하드웨어에 배포됨
- 엄청난 양의 정보를 저장
- 데이터가 분산되며 네트워크는 산발적 중단이 발생하기 쉬움
- 오늘날의 IT 환경은 보안 침해 사고가 자주 발생함

이러한 이유로 응급 상황에 처할 가능성이 높으므로 가장 심각한 상황에서도 복구할 수 있도록 사전 예방적 조치로 미러링을 제공하는 것이 좋습니다.



미러링은 데이터 센터 간, 그리고 내부 인프라 및 클라우드 간 운영 환경을 동기화합니다.

미러링. Hadoop 미러링은 비동기적이어야 하며 WAN을 통해 변경된 사항에 한해 자동 압축된 블록 수준의 데이터 전송을 수행해야 합니다. 항상 데이터의 지역성과 일관성을 유지하면서 데이터는 물론 메타 데이터도 미러링해야 합니다. 이를 통해 사이트 장애 시 즉시 애플리케이션을 다시 시작할 수 있으며, 또한 다음과 같은 특성을 제공해야 합니다.

효율성	안정성	용이성
블록 수준(8KB) 델타	특정 시점 일관성	네트워크 문제의 원활한 처리
자동 압축	전체 체크섬	미러 볼륨 직접 액세스 (스탠바이 아님)
성능 저하 없음		볼륨 수준의 예약

관리 용이성

초기 Hadoop 개발 환경에서는 소스 코드 수준에서 Hadoop을 이해하고 있는 우수한 실력의 개발자가 여러 Hadoop 환경을 관리하는 것이 매우 흔했습니다. 개발자가 Hadoop 내부를 자세히 알고 있으며, 새로 시작하는 기업 환경이 흔히 그렇듯, 개발과 운영 업무를 모두 담당했기에 이러한 방법이 통했습니다. 그러나 기업에서 운영 팀이 Hadoop을 비롯한 수많은 시스템을 모두 능숙하게 다루는 것은 불가능하므로 일반적인 IT 운영 방식으로는 적합하지 않습니다. 총 소유 비용(TCO)은 IT 솔루션을 비교할 때 항상 중요하게 고려되는 항목이지만 특히 Hadoop 환경과 많은 관련이 있습니다.

관리 부담을 줄여줄 종합적이고 지능적인 도구를 제공하는 Hadoop 플랫폼을 선택해야 합니다. Hadoop이 성장함에 따라 Hadoop 배포 프로그램은 다음과 같은

주요 분야에 대해 더욱 우수한 품질의 다양한 관리 도구를 제공하게 될 것입니다.

관리

- 볼륨 기반 데이터 및 사용자 관리
- 중앙 집중식 노드 관리 및 문제 해결
- GUI 화면에서 쉽게 디스크 드라이브의 추가 및 삭제
- 서비스 중단 없는 단계적인 업그레이드
- 관리 작업의 자동화 및 예약 기능
- 데이터 및 작업 제어를 통한 멀티 테넌트 사용자 액세스

모니터링

- 디스크 장애 감지를 포함하여 애플리케이션부터 하드웨어 수준까지의 종합적인 Hadoop 클러스터 모니터링
- 상태, 메모리, CPU 및 기타 항목에 대해 각 노드별 색상 코드 기반 실시간 보기를 제공하는 알림, 경고 및 열 지도(heat map)
- REST API를 통한 서로 다른 오픈 소스와 상용 도구의 통합 및 맞춤형 대시보드 구축 기능
- Ganglia 및 Nagios와 같은 표준 도구를 통한 가시성

마지막으로 Hadoop 구현 환경은 수백 또는 수천 개의 노드로 확장하기 쉽습니다. 이러한 모든 노드를 구성, 배포 및 관리하려면 기계적으로 최대한 자동화되어야 합니다. 다행히 주요 운영 체제 업체는 지속적으로 더욱 향상된 구성 자동화 및 서비스 오케스트레이션 솔루션을 제공하기 위해 노력하고 있습니다. 예를 들어 Canonical의 Juju는 GUI 및 명령줄 인터페이스를 통해 분산 처리 환경의 모든 영역에 대한 관리를 자동화할 수 있습니다.

데이터 액세스

엄청난 양의 정보를 처리하는 것은 Hadoop 구현 환경의 시작 단계에 불과합니다. 데이터에 잠재된 가치를 최대한 활용할 수 있으려면 신속하고 안전하며 간편하게 정보를 처리 및 추출하고, 개발자가 검증된 도구와 기술을 사용하여 완벽한 기능의 애플리케이션을 구축할 수 있는 Hadoop 플랫폼이 필요합니다. 기존 애플리케이션을 쉽게 Hadoop의 데이터와 연결할 수 있다면 더욱 효과적일 것입니다.

이 섹션에서는 특정 Hadoop 플랫폼이 IT 환경의 다른 요소와 원활하게 상호 작용할 수 있는지 확인하는 방법에 대해 설명합니다.

데이터 액세스를 위한 아키텍처 요구 사항

Hadoop의 데이터 액세스 향상을 위한 권장 사항을 알아보기 전에 먼저 표 3에서 몇 가지 기본 전제 조건을 살펴보겠습니다.

전제 조건	중요한 이유
Hadoop 파일 시스템 접근에 대한 완벽한 API 제공	애플리케이션 개발 기능을 완벽하게 제공하면서 다른 Hadoop 구현 환경에 대한 이식도 용이해야 합니다. 그래야만 특정 업체의 기술에 구속되지 않습니다.
파일에 대한 완벽한 POSIX 읽기/쓰기/업데이트	소프트웨어 개발 유연성이 우수하고, 기존 애플리케이션이 Hadoop 데이터와 바로 작동 가능해야 합니다.
주요 리소스에 대한 개발자의 직접 제어	관리자가 영구 또는 임시 테이블을 만들지 않고도 응용 프로그램 워크플로우를 최적화할 수 있어야 합니다
보안, 엔터프라이즈급 검색	애플리케이션에 클릭 기록, 검색 알림 및 검색 관련성 조정과 같은 필수 기능을 포함할 수 있어야 합니다.

전제 조건	중요한 이유
포괄적인 데이터 액세스 도구	관리자와 개발자는 다음과 같이 필요한 작업에 따라 적합한 구현을 선택할 수 있습니다. ···> 로그 수집 및 집계를 위한 Apache Flume ···> Hadoop 및 관계형 데이터베이스, 데이터 웨어하우스 간 동시 가져오기/내보내기를 위한 Apache Sqoop ···> 클러스터 간 및 원격 데이터 소스와 Hadoop 간 분산 복사를 위한 distcp ···> Hive를 통해 Hadoop에서 데이터를 내보내기 위한 ODBC 및 JDBC

표 3: 데이터 액세스를 위한 아키텍처 요구 사항

표준 파일 시스템 인터페이스 및 의미 체계(POSIX)

POSIX 파일 시스템은 Hadoop에 임의 읽기/쓰기 작업은 물론, NFS 액세스를 제공하여 기본 HDFS의 일반적인 용도보다 훨씬 더 다양하게 Hadoop을 활용할 수 있습니다. 매우 복잡한 프로세스가 필요 없게 되어 작업이 간소화되는 효과도 있습니다.

관리자와 사용자는 엔터프라이즈 NAS와 같은 네트워크상에서 클러스터를 구축할 수 있습니다. 따라서 클러스터에서 바로 Windows Explorer, Mac Finder, IDE와 같은 브라우저 및 표준 Linux 파일 상호 작용 명령(예: ls, grep 및tail)을 사용할 수 있습니다. Hadoop 클러스터가 파일 시스템의 일부로 간주되므로 사용자는 Hadoop에 데이터를 끌어서 놓거나, 명령줄 인터페이스에서 Tab 키를 눌러 명령 자동 완성 기능을 사용할 수도 있습니다.

새 Hadoop 기반 애플리케이션 외, 작성된 프로그래밍 언어에 관계없이 다른 기존 또는 신규 솔루션이 파일 시스템을 사용하여 Hadoop에서 데이터에 액세스하고 데이터를 쓸 수 있습니다. POSIX 파일 시스템은 전용 연결 도구 없이 표준 도구를 사용하여 쉽게 관계형 데이터베이스와 데이터 웨어하우스에 대해 정보를 가져오거나 내보낼 수 있습니다.

예를 들어, 사용자가 표준 도구를 사용하여 NFS에 신속히 데이터를 로드할 수 있습니다. 데이터가 바로 스트리밍되어 저장을 위한 추가적인 작업이 발생되지 않아 전체 프로세스의 속도가 저하되지 않습니다.

개발 도구

개발자는 오랜 기간 관계형 데이터베이스와의 상호 작용에 일반적인 도구와 방식을 사용해 왔습니다. Hadoop이 제시하는 새로운 패러다임과 개념도 중요하지만 다음과 같은 기능을 통해 개발자의 생산성을 향상시킬 수 있는 플랫폼을 선택해야 합니다.

- 다운로드 및 맞춤 구성이 가능한 공개된 GitHub에 오픈 소스 구성 요소 제공
- 애플리케이션 구축 기간이 단축되도록 Maven 리포지토리를 통해 바이너리 제공
- 애플리케이션 구축 기간이 단축되도록 워크플로우 엔진 제공
- 클러스터의 데이터와 바로 작업할 수 있는 표준 개발 도구 제공
- 프로그래밍 언어에 제한 없이 기존의 비 Hadoop 애플리케이션과 라이브러리가 Hadoop의 데이터에 액세스하여 쓸 수 있도록 허용
- SQL과 유사한 상호 작용 쿼리 기능 제공

보안

하루도 빠짐 없이 데이터 침해 또는 기타 대규모 보안 침해 사건이 신문의 헤드라인을 장식하고 있으며, 이러한 사건 중에는 빅 데이터가 연루된 경우가 많습니다. Hadoop에 엄청난 양의 정보가 저장되며 데이터의 유형도 매우

광범위하다는 것을 고려하면 이러한 보안 사고가 발생하기 전에 사전 예방적 조치를 취해 데이터를 보호하는 것이 중요합니다. 그러나 일부 Hadoop 구현 환경은 너무 실험적이어서 보안 사고로부터 고객을 안전하게 보호하지 못하며, 때로는 보안 기능을 전혀 제공하지 않는 경우도 있습니다.

특정 기능에 주목하기 보다는 Hadoop 플랫폼이 다음과 같은 보호 기능을 포함하여 포괄적인 보안 솔루션을 제공하는지 확인해야 합니다.

- 파일, 디렉토리, 작업, 대기열, 관리 작업에 대한 정교한 권한 부여
- 테이블, 열 및 열 집단에 대한 액세스 제어 목록(ACL)
- 기본 및 타사 솔루션을 통해 Hadoop 클러스터 및 모든 외부 클러스터의 액세스 지점 간 유선 수준의 암호화
- 표준 인증 프로토콜(예: Kerberos, LDAP, Active Directory, NIS, 로컬 사용자 및 그룹, 기타 타사 인증 및 ID 시스템)
- 다른 모든 노드에 대한 직접 상호 작용을 차단하고 "게이트웨이 노드"를 통해서만 클러스터에 대한 단순한 보안 액세스 제공

주요 HADOOP 배포 프로그램 비교

거의 모든 기업이 빅 데이터를 활용할 수 있는 방법을 모색하고 있으며, 빅 데이터에 내재된 가치를 실현하기 위한 가장 효과적인 통로로 Hadoop을 선택하는 기업도 점점 증가하고 있습니다. 이는 즉, Hadoop이 기업의 핵심적인 역할을 수행하게 됨을 의미합니다. 따라서 성능과 확장성, 신뢰성 및 데이터 액세스의 용이성에 특히 주의하여 Hadoop 구현 환경을 신중하게 선택해야 합니다. 무엇보다 선택한 플랫폼이 기업의 업무 운영 방식과 잘 부합되는지 확인해야 합니다.

다음 페이지에는 주요 Hadoop 배포 프로그램 간의 몇 가지 차이점을 비교한 간단한 표가 나와 있습니다.

	Hortonworks	Cloudera	MapR
성능 및 확장성			
데이터 처리	배치	배치	배치 및 실시간
메타데이터 아키텍처	중앙 집중식	중앙 집중식	분산
HBase 성능	불균일한 응답 속도	불균일한 응답 속도	일관되게 낮은 응답 속도
NoSQL 애플리케이션	주로 배치 애플리케이션	주로 배치 애플리케이션	배치 및 온라인/ 실시간 애플리케이션
신뢰성			
고가용성	단일 장애 복구	단일 장애 복구	다중 장애에 대한 자동 복구
MapReduce HA	작업 재시작	작업 재시작	재시작 없이 지속적인 제공
업그레이드	업무 중단	업무 중단 불필요	업무 중단 불필요
복제	데이터	데이터	데이터 + 메타데이터
스냅샷	미사용 중인 파일만 일관성 보장	미사용 중인 파일만 일관성 보장	모든 파일과 테이블에 대해 특정 시점 일관성
재해 복구	지원 안 함	파일 복사 예약(BDR)	미러링
관리 용이성			
관리 도구	Ambari	Cloudera Manager	MapR Control System
볼륨 지원	지원 안 함	지원 안 함	지원
heat map, 경고, 알림	지원	지원	지원
REST API와의 통합	지원	지원	지원
데이터 및 작업 배치 제어	지원 안 함	지원 안 함	지원
데이터 액세스			
파일 시스템 액세스	HDFS, 읽기 전용 NFS	HDFS, 읽기 전용 NFS	HDFS, 읽기/쓰기 NFS(POSIX)
파일 I/O	추가만 가능	추가만 가능	읽기/쓰기
보안: ACL	지원	지원	지원
쓰기 수준 인증	Kerberos	Kerberos	Kerberos, 기본 인증

저자 소개

미국 실리콘 밸리에 거주하는 기술 컨설턴트이자 저술가인 Robert D. Schneider는 금융, 기술 및 공공 업종의 다양한 기업에 대해 데이터베이스 최적화, 분산 컴퓨팅 및 기타 전문 기술 분야에 대한 컨설팅을 제공해 왔습니다. IBM에서 출판한 Hadoop For Dummies를 비롯하여 지금까지 총 8권의 책을 저술했으며, 데이터베이스 기술, 클라우드 컴퓨팅, 빅 데이터, 데이터 분석, 서비스 중심 아키텍처(SOA) 등의 주제에 대해 수많은 기사를 발표했습니다. 전 세계 여러 기술 산업 행사의 주최자 및 발표자로 활동하고 있으며, www.rdschneider.com에 개인 블로그를 운영하고 있습니다.

ubuntu
Supported by Canonical

HADOOP

BUYERS GUIDE