

빅 데이터를 위한 개방형 DB 프레임워크 “하둡”의 이해

이제 하둡(Hadoop)이라는 단어를 처음 접하는 IT 전문가는 없을 것이다. 하지만, HDFS, 맵 리듀스, H베이스, 너츠, 피그, 하이브 등의 단어가 쏟아져 나오면, DB 전문가가 아닌 이상 갈피를 잡지 못할 것이다. 하둡이란 도대체 무엇인지 기본적인 개념을 이해하고, 이를 구성하는 요소와 하둡의 서브 프로젝트는 어떤 것들이 있는지, 그리고 하둡의 도입 효과는 어떠한지 알아보자.

- ❖ 하둡이란 무엇인가
- ❖ 하둡의 구성 요소와 하부 프로젝트
- ❖ 하둡의 효과
- ❖ 하둡의 향후 과제와 발전 방향

빅 데이터를 위한 개방형 DB 프레임워크, “하둡”의 이해

편집부 | ITWorld

선 사시대 때, 농경을 시작할 무렵 어느 한 부족이 농사를 짓기 위해 드넓은 평야에 다다랐을 때 이미 그 땅을 일궈 밀을 재배하고 있는 다른 부족이 있었다. 이 부족에게 평야의 일부를 빌려달라고 요청하자 너무 많은 대가를 요구했다. 그래서 그 평야를 놓고 싸우려는데, 다른 부족장이 대안을 제시했다. “평야를 지나가면 우리 능력으로도 일궈내지 못한 언덕이 있다. 그 언덕을 가져라.” 밀밭을 지나 언덕을 봤을 때 그 부족은 두 번 놀랐다. 한 번은 그 어마어마한 넓이에, 또 한 번은 땅이 너무 거칠어 제대로 일구기가 힘들다는 데에.

이것이 바로 빅 데이터이며, 7년 동안 이 땅을 일궈 밭을 만들어 낸 것이 바로 하둡(Hadoop) 프로젝트다. 일궈낸 밭에 무엇을 심고, 어떻게 가꿀지는 IT 업체와 각 기업들이 할 일이다. 여기서는 하둡이라는 밭에 대해, 그리고 무엇을 심을 지 알아보기로 하자. 어떻게 가꿀지는 데이터베이스 관련한 거의 모든 IT 업체들이 설명하고 있으니 여기서는 논외로 하겠다.

빅 데이터 시대의 대표 기술, 하둡

2008년 아래로 데이터 폭증 시대가 도래했다. 5년 만에 데이터 양이 44배에 이를 것으로 예측했고, 실제 이를 능가하는 수치를 기록했다. 이는 단순한 데이터량의 증가가 아니라 데이터의 다양성, 속도 등을 함께 아우르는 의미다. 특히 모바일 기기나 소셜 미디어 등에서 생산되는 비정형 데이터의 증가는 소비자의 행태를 바꾸고 있으며, 이는 산업계에서는 비즈니스 경쟁의 규칙조차 바꾸고 있다.

이제 빅 데이터는 피할 수 있는 것이 아니며, 누가 먼저 빅 데이터를 활용하느냐가 향후 비즈니스를 판



가름하는 척도가 된 것이다.

그러나 문제는 폭증하는 데이터를 저장, 분석, 활용하는 방법이 기존 방식으로는 거의 불가능하다는 것. 특히 기존 방식이 1970~80년대 RDBMS(Relational DataBase Management System), 90년대 DW(Data Warehouse) 시대였다면, 이제 빅 데이터 기술이 필요하게 된 것이다. 빅 데이터 저장 분석 기술을 대표하는 것이 바로 하둡이다.

검색에서 시작한 하둡, 클라우드 컴퓨팅과 만나 활기

하둡(Hadoop)은 지난 2005년 루센 개발자인 더그 커팅과 마이크 카파렐라가 구글의 맵 리듀스 알고리즘을 구현하면서 만들어졌다. 이후 커팅은 야후로 옮겨가 검색 서비스에 하둡 기술을 적용하는 프로젝트에 참여했으며, 최종적으로는 약 4만여 대의 서버에 걸쳐 하둡을 구현했다.

하둡이 널리 유명해지기 시작한 것은 클라우드 컴퓨팅과의 연계를 통해 상상을 초월하는 데이터 분석 성능을 제공하는 등의 가시적인 효과가 나타났기 때문이다. 하둡 프로젝트의 핵심 설계자인 톰 화이트가 저술한 <하둡 완벽 가이드>에 따르면, 전에는 시간이 너무 오래 걸려 결과를 얻을 수 없었던 문제들을 이제는 하둡으로 빠르게 해답을 얻을 수 있게 됐다. 대표적인 사례는 다음과 같다.

- **2008년 2월** – 뉴욕타임스는 1851년부터 1980년 12월 까지 130년 분량의 신문기사 1,100만 매를 ‘아마존 S3’에 저장하고, 하둡을 이용해 약 4테라바이트 크기의 데이터를 24시간 만에 변환했다. 이는, 일반 서버로 대략 14년이 걸리는 어마어마한 작업량이다.
- **2008년 4월** – 하둡은 ‘맵 리듀스로 테라바이트 데이터 소트하기’ 대회에서 2007년도 우승자의 297초에 비해 2/3분 수준인 209초 만에 정렬하며 우승했다. 이어

7년 간의 개발, 아파치 하둡 1.0 발표

2012년 1월 초 거의 7년에 이르는 긴 개발과 세부 조정 작업을 거쳐 아파치 하둡 데이터 프로세싱 프레임워크가 마침내 완전한 버전으로 발표됐다.

아파치 하둡 프로젝트 팀이 정식 1.0 버전을 발표한 것이다. 아파치 하둡 부사장 아룬 머시는 이번 버전을 특히 정식 1.0 버전으로 정한 것에는 세 가지 기능이 추가됐기 때문이라고 설명했다.

가장 대표적인 것이 엔드 투 엔드 보안이다. 하둡은 케베로스(Kerberos) 네트워크 인증 프로토콜을 사용해 네트워크 전체에 걸쳐 보안을 확보했다. 케베로스는 분산 컴퓨팅 환경에서 대칭 키를 이용해 사용자 인증을 제공하는 중앙 집중형 인증 방식으로, 신뢰있는 제 3의 컴퓨터가 서비스를 이용하려는 클라이언트의 사용자를 인증함으로써 가능해진다. 한 마디로 중앙 인증 서버가 클라이언트 사용자를 확인해 줌으로써 결과적으로 기업들은 민감하고 개인적인 데이터에 적용된 하둡을 신뢰할 수 있다는 것.

두 번째 기능은 WebHDFS REST(representational state transfer) API로, 많은 관리자와 프로그래머가 쉽게 이해할 수 있는 웹 기술을 사용해 하둡과 인터랙션을 할 수 있게 됐다. 이를 통해 기업의 하둡 적용이 한층 촉진될 것으로 보인다.

마지막으로 이번 버전은 처음으로 H베이스(HBase)를 완전하게 구동해 관리자들이 친숙한 관계형 데이터베이스 같은 구조로 데이터를 저장할 수 있다.

아룬 머시는 “사용자들은 이번 정식 1.0 버전이 오픈소스 커뮤니티의 지원을 받는다는 것을 확실히 알게 됐으며, 더 이상 어떤 기능을 위해 하둡의 어떤 버전을 사용해야 하는지에 대한 혼란은 없다”고 말했다.

2009년 5월, 야후는 하둡으로 62초 만에 1테라바이트를 정렬했다.

- **2009년 4월** – ‘1분 소트’ 대회에서 500기가바이트를 59초에(1,400개 노드에서) 정렬하며 우승했다. 또한 100 테라바이트를 173분에(3,400개 노드에서) 정렬했다.

데이터 분산 저장 분산 처리 프레임워크, 하둡

하둡은 대량의 자료를 처리할 수 있는 대규모 컴퓨터 클러스터에서 동작하는 분산 애플리케이션을 지원하는 오픈 자바 소프트웨어 프레임워크다. 원래 검색의 분산처리를 지원하기 위해 개발된 것으로, 아파치 루센의 하부 프로젝트였다. 분산처리 시스템인 구글 파일 시스템을 대체할 수 있는 하둡 분산 파일 시스템, 즉 HDFS(Hadoop Distributed File System)와 분산 처리 시스템인 맵 리듀스(Map Reduce)를 구현한 것이다. 이는 위키피디아에서 정의한 하둡의 의미를 조금 수정한 것이다.

한 마디로 얘기하면, 하둡은 대용량 데이터 처리 분석을 위한 대규모 분산 컴퓨팅 지원 프레임워크다. 하둡의 구성 요소 가운데 핵심 구성은 바로 저장과 처리(계산)이다. HDFS를 통해 분산 저장하고, 맵 리듀스를 통해 분산 처리한다는 것.

여기서 하둡의 가장 큰 특성이 나타난다. 바로 분산(Distributed)이다. 즉 분산 처리, 분산 저장이다. 하둡은 여러 개의 컴퓨터를 마치 하나인 것처럼 묶어주는 기술을 통해 저장 공간과 계산 능력을 늘린다.

하둡의 대표적 구성 요소, HDFS

HDFS는 하둡의 구성 요소 가운데 가장 알기 쉽다. 이름이 뜻하는 대로 분산형 파일시스템이다. HDFS란 하둡 네트워크에 연결된 아무 기기이나 데이터를 분산해 밀어넣는 것이다. 물론 여기에도 체계가 있어 그냥 닦치는 대로 배치하는 것은 아니다. 파일을 적당한 블록 사이즈(64MB)로 나눠서 각 노드 클러스터, 즉 각각의 개별 컴퓨터에 저장한다. 또한 데이터 유실의 위험이나 사람들이 많이 접속할 때의 부하 처리를 위해 각 블록의 복사본(Replication)을 만들어 둔다. 이런 형태는 RDBMS의 고도로 염격한

저장 인프라와 비교해 보면 아무 기기애나 닥치는 대로 밀어넣는 것이나 다름없다.

그렇지만 이런 유연성이야말로 하둡의 장점이라 볼 수 있다. RDBMS가 정교하게 제어하는 전용 기기들을 필요로 하는 것과는 달리, HDFS는 좋은 하드 드라이브가 거의 없는 범용 서버라도 얼마든지 활용할 수 있다. 또한 RDBMS 테이블에 데이터를 저장하는 데에 따른 막대한 관리 비용을 들이지 않고, 그대신 HDFS를 이용해 데이터를 다수의 기기들과 드라이브들에 저장하며 다수의 노드로 이뤄진 하둡 시스템에 데이터가 자동적으로 중복되게 만든다. 따라서 만약 하나의 노드에서 고장이 발생하거나 느려지더라도 여전히 그 데이터에 접근할 수 있다.

하둡 분산 파일 시스템은 다음과 같은 시스템에서 잘 동작하는 것을 목표로 하고 있다.

- **하드웨어 오동작** : 하드웨어 수가 많아지면 그 중에 일부 하드웨어가 오동작하는 것은 예외 상황이 아니라 항상 발생하는 일이다. 따라서 이런 상황에서 빨리 자동으로 복구하는 것은 HDFS의 중요한 목표다.
- **스트리밍 데이터 접근** : 일반 파일 시스템과 달리 반응 속도보다는 시간당 처리량에 최적화되어 있다.
- **대용량 데이터 집합** : 한 파일이 기가바이트나 테라바이트 정도의 크기를 갖는 것을 목적으로 설계됐다. 데이터 대역폭 총량이 높고, 하나의 클러스터에 수백 개의 노드를 둘 수 있다. 하나의 인스턴스에서 수천만 파일을 지원한다.
- **간단한 결합 모델** : 한번 쓰고 여러 번 읽는 모델에 적합한 구조다. 파일이 한번 작성되고 닫히면 바뀔 필요가 없는 경우를 위한 것이다. 이렇게 함으로써 처리량을 극대화할 수 있다.
- **자료를 옮기는 것보다 계산을 옮기는 것이 비용이 적게 든다** : 자료를 많이 옮기면 대역폭이 많이 들기 때문에 네트워크 혼잡으로 인해 전체 처리량이 감소한다. 가까운 곳에 있는 자료를 처리하게 계산 작업을 옮기면 전체적인 처리량이 더 높아진다.
- **다른 종류의 하드웨어와 소프트웨어 플랫폼과의 호환성** : 서로 다른 하드웨어와 소프트웨어 플랫폼들을 묶어 놓아도 잘 동작한다.

HDFS는 마스터/슬레이브(master/slave) 구조를 가진다. HDFS 클러스터는 하나의 네임 노드와 파일 시스템을 관리하고 클라이언트의 접근을 통제하는 마스터 서버로 구성된다. 게다가 클러스터의 각 노드에는 데이터 노드가 하나씩 존재하고, 이 데이터 노드는 실행될 때마다 노드에 추가되는 스토리지를 관리한다.

HDFS는 네임 스페이스를 공개해 사용자 데이터가 파일에 저장되는 것을 허락한다. 내부적으로 하나의 파일은 하나 이상의 블록으로 나누어져 있고, 이 블록들은 데이터 노드에 저장되어 있다. 네임 노드는 파일과 디렉터리의 읽기(open), 닫기(close), 이름 바꾸기(rename) 등 파일 시스템의 네임스페이스의 여러 기능을 수행한다. 또한, 데이터 노드와 블록들의 맵핑을 결정한다.

데이터 노드는 파일시스템의 클라이언트가 요구하는 읽기(read), 쓰기(write) 기능들을 담당한다. 또한 데이터 노드는 네임 노드에서의 생성, 삭제, 복제 등과 같은 기능도 수행한다. 네임 노드와 데이터 노드는 GNU/리눅스 운영체제를 기반으로 하는 일반 서버에서 실행하기 위해 디자인된 소프트웨어의 일부



다. HDFS는 자바(Java) 언어를 사용하므로 자바가 동작하는 어떤 컴퓨터에서나 네임 노드나 데이터 노드 소프트웨어를 실행할 수 있다. 이런 특성은 하드웨어나 관리 단계에서 엄청난 비용을 절약해준다.

한편 HDFS가 하둡과 함께 사용되는 일반적인 파일 시스템이긴 하지만, HDFS가 결코 유일한 하둡의 파일 시스템은 아니라는 점에도 주목할 필요가 있다. 기본적으로 하둡의 파일 시스템은 HDFS가 맞지만, 오픈 소스인 하둡은 누구에게나 공개되어 있기 때문에 파일 시스템으로 무엇을 쓰느냐는 해당 기업의 선택이라는 것이다.

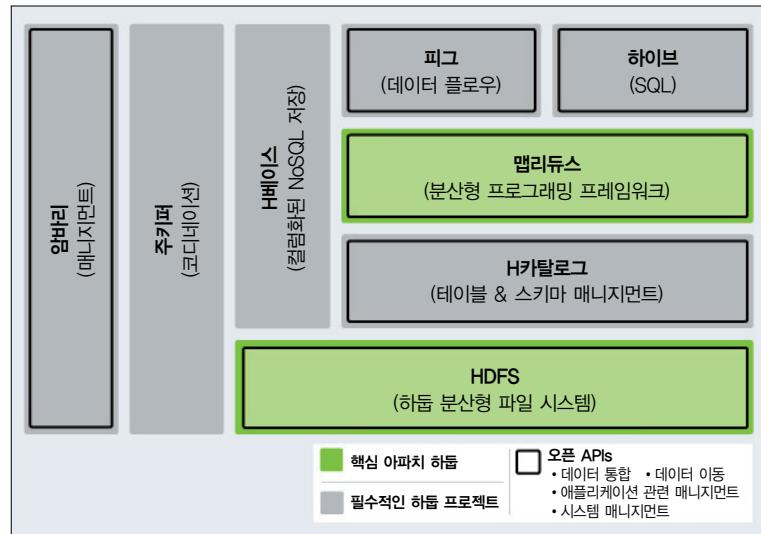
하둡이라는 밭은 누구나 다 사용할 수 있기 때문에 그 밭을 논으로 바꾸거나, 혹은 다시 나무를 심을 수도 있다는 얘기다. 예를 들어 아마존의 엘라스틱 컴퓨트 클라우드(Elastic Compute Cloud, EC2)의 경우에는 하둡에 HDFS 대신 자사의 S3 파일시스템을 채용했다. 데이터스택스(DataStax)의 브리스크(Brisk)는 자칭 하둡 배포판으로, HDFS 대신 카산드라 FS(Cassandra FS)로 대체했으며, 실시간 저장 및 분석 기능들을 통합하기 위한 하이브(Hive)의 데이터 쿼리 및 분석 기능까지 추가로 포함하고 있다.

분산 처리를 위한 프레임워크, 맵 리듀스

맵 리듀스(Map Reduce)는 원래 구글에서 분산 컴퓨팅을 지원하기 위해 2004년에 발표한 소프트웨어 프레임워크다. 이는 페타바이트 이상의 대용량 데이터를 신뢰할 수 없는 컴퓨터로 구성된 클러스터 환경에서 병렬 처리를 지원하기 위해 개발됐다. 중요한 것은 맵 리듀스가 프레임워크라는 점이다. 즉 분산처리를 위해 프레임워크를 만들어 두고 이에 맞춰 코딩을 하고 하둡 시스템에서 실행하면 자동으로 분산처리를 해 준다.

이 프레임워크는 함수형 프로그래밍에서 일반적으로 사용되는 맵(Map)과 리듀스(Reduce)라는 함수를

그림. 하둡의 구성 요소



기반으로 구성된다. 개념은 간단하다. 맵 함수에서 데이터를 처리하고, 리듀스 함수에서 원하는 결과값을 계산하는 것이다.

하둡 플랫폼에는 기본 요소인 HDFS와 맵 리듀스 와 함께 분산 데이터베이스인 H베이스(HBase), 검색엔진인 너치(Nutch), 관계형 대수 쿼리 언어 인터페이스인 피그(Pig), 데이터웨어하우징 솔루션인 하이브(Hive), 그리고 테이블 및 스토리지 관리 서비스인 H카탈로그 등이 포함된다(그림 참조).

하둡을 지원하는 하부 프로젝트들

하둡의 하부 프로젝트는 다음과 같은 에이브로(Avro), 주키퍼(ZooKeeper), 피그, H베이스, 하이브 등이 있다.

- 에이브로(Avro)** : 이기종 간 데이터 타입을 교환할 수 있는 체계를 제공하는 프레임워크로, 아파치 하둡 프로젝트의 서브 프로젝트로 시작된 후 아파치 메인 프로젝트로 승격됐다. 쓰리프트(Thrift)가 RPC(Remote Procedure Call) 요청을 안정적으로 처리하면서 이기종 간 RPC 호출을 지원하는 개념으로 접근했다면, 에이브로는 데이터 직렬화를 기본 개념으로 해 RPC 호출을 이기종 간에 가능케 하는 개념으로 접근한다. 따라서 에이브로의 이기

종간 데이터 접근은 RPC뿐만 아니라 파일에 데이터를 저장하는 등의 용도로도 사용할 수 있다.

• **주키퍼(ZooKeeper)** : 분산 환경에서 노드 간의 정보 공유, 락(Lock), 이벤트 등 보조 기능을 제공하는 프레임워크다. 아파치 하둡 프로젝트에서는 프로젝트 이름들이 코끼리(hadoop), 거북이(chukwa), 돼지(pig) 등과 같이 동물들 이름이 많다. 동물 사육사라는 의미대로 주키퍼는 하둡 분산 처리 시스템을 일괄적으로 관리해주는 시스템이다. 분산처리 환경에서는 기본적으로 서버가 몇 대에서 수백 대, 수천 대까지 갈 수도 있다. 이런 분산처리 환경에서는 예상치 못하는 예외적인 부분이 많이 발생하게 되는데, 주로 네트워크 장애, 일부 서비스 및 기능 중지나 장애, 서비스 업그레이드, 서버 확장 등에 문제가 발생할 수 있다. 이를 총체적으로 관리하는 것이 바로 주키퍼다.

주키퍼는 ▲ 하나의 클라이언트(하나의 서버)만 서비스를 수행하지 않고 알맞게 분산해 각각의 클라이언트들이 동시 작업할 수 있도록 지원하는 네임 서비스를 통한 부하 분산 ▲ 하나의 서버에서 처리된 결과가 또 다른 서버들과 동기화하는 분산 락 ▲ 액티브 서버가 예기치 못한 상황으로 문제가 발생해 서비스를 지속적으로 처리하지 못할 경우 대기 서버가 액티브 서버로 바뀌어서 기존에 액티브 서버가 하던 서비스를 처리하는 장애 상황 판단 및 복구 ▲ 각각의 다른 서버들을 통합적으로 관리, 환경 설정을 따로 분산하지 않고 주키퍼 자체적으로 관리하는 환경 설정 관리 등이 주된 기능이다.

• **피그(Pig)** : 하둡의 서브 프로젝트 가운데 하나인 피그는 대규모 데이터 셋에 대한 분석을 위해 야후에서 개발한 관계형 대수 쿼리 언어 인터페이스(Relational Algebra Query Language Interface)다. 병렬 컴퓨팅을 위한 고레벨 데이터 플로우 언어로서, 스크립트를 통해 맵 리듀스 기능을 수행하는 환경을 제공한다. 한 마디로 피그는 맵 리듀스용 프로그래밍 및 데이터 플로우 인터페이스.

• **H베이스(Hbase)** : HDFS를 지원하기 위한 구글의 빅테이블(BigTable)에 기반한 분산 데이터베이스 모델이다. 칼럼 기반 데이터베이스인 H베이스는 대량의 데이터를 온라인 상에서 활용할 수 있으며, 실시간 업데이트라는 점을 특징으로 한다. 또한 H 베이스는 SQL 형식의 DB가 아니라 NoSQL 형식으로, relational, joins, sophisticated query engine, 열 입력, SQL 등등 SQL 형식의 데이터 베이스 특징을 갖고 있지 않다. 기존 RDBMS는 단일 노드의 크기로 묶이기 때문에 과부하가 많은 반면, H베이스는 확장 가능한 데이터 저장을 지원한다. 하지만 RDBMS를 대체하기 위함이 아니다.

• **하이브(Hive)** : SQL(Structured Query Language)과 유사한 언어인 하이브 QL(Hive QL)을 이용해 데이터를 요약하고, 쿼리를 수행하고, 분석할 수 있는 데이터 웨어하우징 솔루션이다. 아룬 머시는 “하이브야말로 사람들이 소위 NoSQL 데이터 베이스에서 기대하는 것보다 하둡을 훨씬 쉽게 이용할 수 있게 만들어 줄 것”이라고 말했다. 하이브 QL를 이용해 데이터 분석가들은 RDBMS에서 이용하던 것과 거의 동일한 쿼리들을 가진 하둡 데이터베이스에서 정보를 빼낼 수 있다. 물론 SQL과 하이브 QL은 염연히 다르므로 하둡으로의 이전은 일정한 이행기를 거치게 되겠지만, 이 차이들은 그렇게 크지 않다.

또한 하이브 쿼리언어가 필요한 정보를 제공할 수 없다고 판명된 경우, 맵 리듀스 프로그래머들이 직접 그들이 만든 데이터 매퍼(data mapper)와 데이터 리듀서(data reducer)를 불러오게 할 예정이다. 단, 하이브를 활용할 때는 주의해야 할 점이 있다. 하둡은 일괄 처리 시스템(batch processing system)에 기반을 뒀기 때문에 짧은 응답시간을 약속할 수 없다. 하이브 쿼리들로 뜯기는 과정에서 아주 높은 지연시간을 가진다(몇 초가 아니라 몇 분이 될 수도 있다). 따라서 하이브는 실시간 프로세싱에 적합한 시스템이 아니다. 만약 실시간 분석이 필요하다면 H베이스를 이용하거나 아파치 카

산드라(Apache Cassandra)를 이용하는 편이 좋다. 카산드라는 오픈소스 분산형 데이터베이스 관리 시스템으로 실시간 요구들을 처리하는데 훨씬 적합하다.

- **H카탈로그(HCatalog)** : 하둡 데이터용 테이블 및 스토리지 관리 서비스

더그 커팅은 “하둡을 중심으로 성장한 프로젝트들의 전체 생태계가 있으며, 이는 계속해서 성장하고 있다. 하둡은 분산된 운영체제의 커널이며, 커널 주변의 다른 모든 구성요소들은 현재 개발 중에 있다. 피그와 하이브는 그런 것들의 좋은 예라 할 수 있다. 단순히 하둡만을 사용하는 사람은 없다. 그들은 여러 개의 다른 툴들도 사용하고 있다”고 설명했다.

하둡에 대한 오해

지금껏 하둡에 대해 설명했지만, 분명하게 알아둬야 할 것이 있다. 첫째 하둡은 빅 데이터에만 배타적으로 사용되는 시스템이 아니며, 둘째 진정한 의미의 NoSQL 툴도 아니다.

하둡이 비관계형 DBMS에 속하는 것은 맞지만, 그렇다고 SQL을 사용하지 못하는 것은 아니다. 이는 심지어 NoSQL의 의미로도 적절치 않다. NoSQL이란 SQL뿐 아니라 다른 쿼리 시스템도 사용될 수 있는 데이터베이스를 표현하는 용어다. 사실 하둡에서는 SQL과 유사한 쿼리언어들을 상당히 쉽게 이용할 수 있다.

그리고 빅 데이터에 관한 하둡의 개념도 수정돼야 한다. 많은 사람들은 하둡이라고 하면 방대한 양의 데이터부터 연상한다. 여기에는 물론 나름대로 합당한 이유가 있다. 하둡 스토리지는 누구라도 막대한 양의 데이터를 떠올릴만한 페이스북과 야후가 사용하고 있기 때문이다. 하둡의 얼리어댑터이자 엄청난 기여를 한 야후는 이미 5만 개의 노드로 구성된 하둡 네트워크를 설치했으며, 페이스북은 1만 개가 넘는 노드로 구성된 하둡 시스템을 갖췄다.

그러나 아파치소프트웨어재단(ASF)의 아파치 하



둡 부사장이자 호트웍스의 아키텍트 아룬 머시는 하둡 자체와 기업 내 하둡의 활용에 대해 이견을 제기했다. 머시는 “하둡의 활용은 빅 데이터를 훨씬 넘어선다”고 주장했다.

모든 데이터를 동등하게 저장

하둡의 가장 강력한 능력 가운데 하나는 바로 확장성이며, 야후와 페이스북은 하둡이 어떻게 확장될 수 있는지를 보여주는 훌륭한 예일 뿐이다. 그러나 하둡이 다른 방향으로 어떻게 확장할 수 있으며, 모든 규모의 기업들에게 의사결정에 필요한 분석 데이터를 제공하는 방법에 대해서는 별다른 이견이 없다.

머시의 설명에 따르면, 이전의 데이터 스토리지에는 비용이 많이 들었다. 5년 전만 해도 대기업과 중소기업들은 폭발적으로 증가하는 데이터를 계속해서 기록하고 스스로 관리해야 했다. 이메일, 검색 결과, 판매 데이터, 재고 데이터, 고객 데이터, 웹의 클릭 경로 등이 모든 정보들과 그 외의 더 많은 데이터들이 쏟아져 들어왔고, 이를 RDBMS에서 관리한다는 의미는 곧 막대한 비용을 뜻했다.

제대로 된 데이터 관리 기능과 비용 절감을 동시에 추구하는 기업은 들어오는 모든 이벤트 및 신호 데이터들을 일반적으로 더 작은 하위 집합들로 샘플링할 것이다. 머시는 이렇게 하향 샘플링(downsampling)된 데이터를 ‘히스토리컬 데이터(historical data)’라고

부른다. 이 데이터는 자동으로 특정 가정에 근거해 분류되며 그 중에서도 어떤 데이터가 다른 데이터들 보다 항상 더 중요할 것이라는 가정이 가장 우선적으로 작용할 것이다.

예를 들면 전자상거래 데이터에서 우선 순위는 신용카드 데이터가 제품 데이터보다 중요하며 제품 데이터는 클릭 경로 데이터보다 중요하다는 합리적인 가정이 수립될 것이다. 이렇게 하나의 주어진 가정 집합에 근거한 비즈니스 모델을 운영한다면 비즈니스 의사결정을 위한 정보들을 끌어내기가 별로 어렵지 않을 것이다. 하지만 이 정보들은 항상 초기 가정들에 입각해 있을 것이고 만약 그 가정 자체가 바뀌면 어떻게 될까?

데이터가 이전에 이미 하향 샘플링을 거쳤기 때문에 모든 원시 데이터(raw data)는 이미 오래 전에 사라져 버렸을 것이고, 어떤 새로운 비즈니스 시나리오든 스토리지에 남아있는 데이터를 사용해야 할 것이다.

게다가 RDBMS 기반 스토리지의 비용 부담으로 이 데이터는 종종 기관 내부에서 격리된 채로 저장된다. 영업부의 데이터는 영업부가, 마케팅부의 데이터는 마케팅부가, 회계부의 데이터는 회계부가, 그 외 여타 부서들도 마찬가지로 각 부서가 자신들의 데이터를 보관할 것이다. 따라서 비즈니스 모델과 관련된 결정들은 기업 전체가 아니라 기업의 각 부서들로 제한될 것이다.

머시는 “하둡을 이용하면 모든 데이터를 그냥 저장하기 때문에 아무런 가정도 필요하지 않다”고 주장했다. 이런 사실은 비록 하둡의 낮은 경제적 비용이라는 이점 뒤에 가려져 많이 주목받지 못하고 있지만, 아마 향후 하둡의 가장 큰 장점이 될 것이다. 머시는 또 “하향 샘플링은 어떤 특정 데이터가 다른 데이터들보다 더 크고 더 중요할 것이라는 가정을 토대로 한다. 그러나 하둡에서는 모든 데이터가 동등한 가치를 갖는다”라고 설명했다.

하둡에서는 모든 데이터의 가치가 동등하고, 각각이 비슷한 수준으로 이용될 수 있기 때문에, 비즈니스 시나리오들은 언제라도 아무런 제약 없이 원시

데이터로 원하는 작업을 진행할 수 있다. 게다가 이전에는 사일로(Silo)화 됐던 데이터들에도 동등하게 접근할 수 있으며, 이들을 공유함으로써 기업 비즈니스를 더 크게, 전체적으로 접근해 분석할 수 있을 것이다.

이제는 히스토리컬 데이터라는 것이 사라진다는 점에서 데이터가 어떻게 인식될 것인가의 변화는 실로 엄청나다. 게다가 데이터가 있는 그대로 저장될 수 있기 때문에 추출, 변환, 로드 연산 등과 관련된 데이터 관리 일들이 많이 줄어들 것이다.

비용 절감 효과는 ‘DB 계의 리눅스’

하둡의 여러 장점 가운데서도 저렴한 비용이 단연 가장 많은 주목을 받고 있다는 사실에는 반론의 여지가 없다. 아파치 소프트웨어 라이선스에 따라 전체 프레임워크가 오픈소스이기 때문에 기본 소프트웨어에 대한 라이선스 비용을 지불하지 않아도 된다.

하둡 프레임워크의 창안자 가운데 한 명인 더그 커팅을 채용한 상용 하둡업체인 클라우데라(Cloudera)는 오픈 코어 모델(open core model)을 사용하며, 따라서 하둡 기본 소프트웨어는 무료이지만 클라우데라의 연장 도구에는 라이선스 비용을 지불해야 한다.

머시가 야후의 하둡 팀에서 함께 근무했던 몇 명의 직원들과 2011년 초 공동 창업한 호튼웍스는 모든 소프트웨어를 무료로, 그리고 오픈소스로 제공하는 대신 교육과 지원 프로그램 등을 통해 수익을 올리고 있다.

그 이외에도 하둡에는 비용을 절약할 수 있는 여러 특성들이 있다. RDBMS와는 다르게 하둡은 고가의 하드웨어나 고성능 프로세서를 필요로 하지 않는다. 하둡 네트워크에 연결된 상용 서버면 충분하다. 즉 하둡 노드는 오로지 프로세서 하나, 네트워크 카드 하나, 하드 드라이브 몇 개만을 필요로 하는데 이를 총 합치면 약 4,000달러 정도 수준이다. 반면 RDBMS 시스템은 테라바이트 당 1만에서 1만 4,000달러 정도의 비용이 든다. 이렇게 큰 비용 차이는 하둡의 가치를 설명하는 결정적 요소다.

커팅은 “하둡은 상대적으로 새로운 기술이다. 사

람들은 하둡이 얼마나 유용한지 알아가고 있다. 개인적으로 하둡은 여전히 성장 중이며, 사람들은 더 많은 용도를 찾아내고 있는 중”이라고 말했다. 수 년간 소프트웨어는 하드웨어에 뒤쳐져 있었으며, 이제 그 차이를 따라잡고 있다. 기업들은 이제 구매한 하드웨어를 제대로 활용할 수 있는 소프트웨어인 하둡을 갖게 되었다. 그리고 이제 기업들은 이것을 이용해서 할 수 있는 모든 것을 찾아내고 있다.

하둡을 활용하는 용도는 각 산업마다 다른데, 금융 업계에서 사람들은 사기범죄 탐지에 사용하고, 신용 카드사들은 어떤 거래가 사기인지 알아보기 위해 사용하며, 은행들은 대출 문제에 있어 고객들의 신용을 평가하는데 쓴다. 유통업계는 재고를 분석하면서 광고를 분석해 장기적인 추세를 가늠한다. 정보 기관은 정보 분석에 활용하고 있다.

“써 보면, 압니다”

하둡에 대해 알아보고 싶은 DB 관리자가 있으면, 데이터 프레임워크로 구성된 오픈소스 소프트웨어를 다운로드받아 비교적 쉽게 시험해볼 수 있다. 하둡의 탁월한 기능은 국내 DB 관리자 가운데 얼리어답터

계층에 있는 이들은 이미 한번쯤 실험해 봤으며, 각 블로그를 통해 데이터 처리 속도에 대한 글들이 올라오고 있다.

머시는 “보통 한두 명의 엔지니어들이 하둡을 다운로드받아 하나의 노드 혹은 4~5개의 노드로 구성된 조그만 클러스터에 하둡을 설치한다”고 설명했다. 이후에는 거의 같은 패턴이라고. 먼저 하둡 클러스터를 사용하는 직원들이 그 가치를 알기 시작한다. 그러면 기업의 다른 부서에서도 그들만의 하둡 클러스터를 설치할 것이다.

결국에는 하둡의 효과가 크게 늘어나고 그 아래 깔려있는 분산형 파일시스템의 확장성 덕분에 각각 분리되어 있는 하둡 클러스터들이 대략 50~60 노드 정도로 이뤄진 커다란 단일 클러스터로 합쳐진다.

머시에 따르면, 야후에서 처음 하둡을 받아들일 때에도 바로 이런 과정을 거쳤다고. 각 부서들과 애플리케이션에게 하둡의 가치가 명료해지기만 하면, 하나의 커다란 하둡 네트워크에 모든 것들을 결합시키는 것이 분명 이상적이다. 물론 페이스북이나 야후처럼 각각 1만 개 혹은 5만 개의 노드로 이뤄진 대규모 시스템을 배치해야 할 기업은 많지 않을 것이다. 그러나 일반 원칙은 여전히 동일하다.

하둡의 향후 과제와 발전 방향

그러나 머시는 하둡을 활용하면서 그 효과는 지대하지만 기업이 하둡을 도입할 때에는 반드시 유념해야 할 몇 가지 제약 사항들이 있다고 지적했다.

우선 기업이 데이터에서 초 단위 이하(sub-second)로 상호 간에 보고하거나, 혹은 데이터를 다단계로 복잡한 트랜잭션에서 이용하고 있다면, RDBMS를 그대로 사용하는 편이 좋다. 하둡은 이런 영역에서는 그리 강점을 살리지 못하기 때문이다. 또한 데이터가 삽입 및 삭제를 통해 자주 갱신되고 바뀌는 경우에도 역시 하둡을 사용하지 않는 편이 좋다.

포레스터는 최근 한 보고서에서 하둡에 대해 상대적인 미성숙, 사용 애플리케이션의 부재, 기술의 부족 등으로 인해 기업들이 문제를 겪게 될 가능성이 있다고 지적했다. 또한 하둡 전문가의 부족 등으로



시장 확대에 어려움을 겪게 될 것으로 보인다.

이에 대해 커팅은 “하둡의 상대적인 미성숙은 ‘어린아이한테 어리다고 비난하는 것’과 같다. 하둡은 이제 걸음마를 시작했다. 다른 DBMS 기술들은 수십 년간 노하우가 축적됐으며, 그동안 애플리케이션 등 많은 부분을 개선해 왔다. 하둡은 아직 이런 단계에 이르지 못했다”고 반박했다.

또한 “하둡의 주변 소프트웨어는 성장 잠재력이 크다. 그렇다고 그것이 하둡이 해결할 수 없는 문제 가 없다거나 예전보다 지금에 이르러 더 많은 문제를 해결할 수 있다는 의미는 아니다. 누군가에게 있어 하둡은 이미 준비된 솔루션이지만 다른 누군가, 또는 일부 애플리케이션에 있어서 하둡은 앞으로 준비될 솔루션일 것”이라고 설명했다.

기업들은 현재 하둡에서 해결할 수 없는 문제를 확인하고 테스트를 진행해야 한다. 클러스터를 구성하고 특정 애플리케이션을 평가한 후, 얼마나 감당할 수 있는지, 그리고 하둡을 통해 얼마나 개선되는지 살펴봐야 한다.

커팅은 “개인적으로 최소한 초기 애플리케이션들

에 대해서는 테스트를 진행할 수 있을 것”이라며, “하둡 클러스터에 더 많은 데이터를 저장할수록 실제적인 시너지 효과가 발생한다”고 조언했다.

또한 커팅은 “현재 하둡은 미완성이라고 생각한다. 이제 막 걸음마를 시작한 어린 아이와 같다고 할 수 있다”며, “5년 정도가 지나면 하둡은 IT 인프라의 실제적인 구성 요소로 자리잡기 시작할 것”이라고 주장했다.

“빅 데이터와 관련해 현재 오라클과 마이크로소프트 등 업체들이 내놓는 것들은 실험적인 것들이다. 지금은 이런 실험적인 결과물을 이용해 현재 상황을 때우고 있는 것이다. 하지만 5년 뒤에는 실험의 단계를 벗어날 것이라 생각한다. 하둡도 마찬가지일 것”이라고 예측했다.

이 상황에서 기업들은 충분히 느슨하게 연결되어 발전하고 변화하도록 하면서 구성 요소별로 교체할 수 있는 결과물을 구축함으로써 단기간 내에 대대적인 변화를 가할 필요가 없는 상황을 유지해야 할 것이다. ITWORLD

Global IT Standard IDG

PC World, Computer World, CIO 등으로 잘 알려진 IDG는
90여 개국에서 180여 미디어를 발행하는 글로벌 테크놀로지 미디어로,
전 세계에 1억 4,000만 명의 독자를 대상으로
미디어, 리서치, 컨퍼런스, 이벤트 등
다양한 테크놀로지 관련 서비스를 제공하고 있습니다.



한국IDG(주) 서울시 종구 봉래동 1가 108번지 창화빌딩 4층 100-161

Tel : 02-558-6950 Fax : 02-558-6955 www.idg.co.kr twitter.com/ITWorldKR www.facebook.com/IDGKorea

IDG Korea Contacts

Sales Contact

김성일, Eddy Kim

Tel: + 82-2-558-6939 Mobile: + 82-10-2702-0360
Email: sungil_kim@idg.co.kr

이포원, Tony (Powon) Lee

Tel: + 82-2-558-6924 Mobile : + 82-10-2053-2413
Email : tony_lee@idg.co.kr

CIO Editorial Contact

천신웅, Brian Cheon

Tel: + 82-2-558-9882 Mobile: + 82-10-3221-7824
Email: psy_cheon@idg.co.kr

박해정, Jenny Park

Tel: + 82-70-7725-9955 Mobile: + 82-10-2810-4518
Email: jenny_park@idg.co.kr

IT World Editorial Contact

박재곤, Jay Park

Tel: + 82-2-558-9887 Mobile: + 82-10-2782-6722
Email: jay_park@idg.co.kr

이대영, Dy Lee

Tel: +82-70-7725-9951 Mobile: +82-10-6511-8982
Email: dy_lee@idg.co.kr

김현아, Hyuna Kim

Tel: + 82-2-558-6956 Mobile: + 82-10-8300-8511
Email: hyuna_kim@idg.co.kr

PR & Marketing Contact

김종민, Matt Kim

Tel: + 82-2-558-6079 Mobile: + 82-10-2408-4121
Email: matt_kim@idg.co.kr

박용학, Yong Park

Tel: + 82-2-558-6965 Mobile: + 82-10-4744-2823
Email: yongpark@idg.co.kr



IDG KOREA

4F ChangHwa Bldg., Bongnae-dong 1Ga, Jung-Gu, Seoul, Korea 100-161