오픈 소스 분산 처리 환경 구축 가이드

빅 데이터 시대의 하둡 완벽 입문

제2판

Hadoop 徹底入門 第2版
(Hadoop TetteiNyuumon dail2han : ISBN 978-4-7981-2964-8)

Copyright © 2013 Kazuki Ohta / Masatake Iwasaki / Kousuke Saruta / Toru Shimogaki / Tatsurou Fujii / Shinichi Yamashita.
Original Japanese edition published by SHOEISHA Co., Ltd.
Korean translation rights arranged with SHOEISHA Co., Ltd.
in care of The English Agency (Japan) Ltd. through Danny Hong Agency.
Korean translation copyright © 2014 by J-PUB.

이 책의 한국어판 저작권은 대니홍 에이전시를 통한 저작권자와의 독점계약으로 제이펍에 있습니다.
신저작권법에 의하여 한국 내에서 보호를 받는 저작물이므로 무단전재와 무단복제를 금합니다.

오픈 소스 분산 처리 환경 구축 가이드

빅 데이터 시대의 하둡 완벽 입문 제2판

1쇄 발행 2014년 6월 18일

지은이 오오타 카스기, 이와사키 마사타케, 사루타 코우스케, 시모가키 토오루, 후지이 타츠로우, 야마시타 신이치
옮긴이 김완섭
펴낸이 장성두
펴낸곳 제이펍

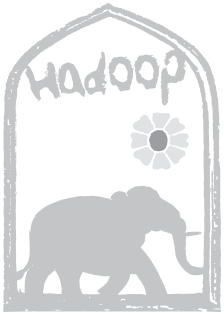
출판신고 2009년 11월 10일 제406-2009-000087호
주소 경기도 파주시 문발로 141 뮤즈빌딩 403호
전화 070 – 8201 – 9010 / **팩스** 02 – 6280 – 0405
홈페이지 www.jpub.kr / **이메일** jeipub@gmail.com

편집부 이민숙, 이 슬, 이주원 / **소통·기획팀** 현지환
본문디자인 북아이 / **표지디자인** 미디어픽스
용지 신승지류유통 / **인쇄** 한승인쇄사 / **제본** 광우제책사

ISBN 978-89-94506-96-8 (93000)
값 36,000원

※ 이 책은 저작권법에 따라 보호를 받는 저작물이므로 무단 전재와 무단 복제를 금지하며,
이 책 내용의 전부 또는 일부를 이용하려면 반드시 저작권자와 제이펍의 서면 동의를 받아야 합니다.
※ 잘못된 책은 구입하신 서점에서 바꾸어 드립니다.

제이펍은 독자 여러분의 책에 관한 아이디어와 원고 투고를 기다리고 있습니다. 책으로 펴내고자 하는 아이디어나 원고가 있으신 분께서는 책에 대한 간단한 개요와 차례, 구성과 저(역)자 약력 등을 메일로 보내주세요. **(보내실 곳: jeipub@gmail.com)**



오픈 소스 분산 처리 환경 구축 가이드

빅 데이터 시대의 하둡 완벽 입문

제2판

오오타 카스기, 이와사키 마사타케, 사루타 코우스케, 시모가키 토오루, 후지이 타츠로우, 야마시타 신이치 지음 /
하마노 켄이치로 감수 / 김완섭 옮김

Jpub 제이펍

※ 드리는 말씀

- 이 책에 기재된 내용을 기반으로 한 운용 결과에 대해 저자, 역자, 소프트웨어 개발자 및 제공자, 제이펍 출판사는 일체의 책임을 지지 않으므로 양해 바랍니다.
- 이 책에 등장하는 각 회사명, 제품명은 일반적으로 각 회사의 등록 상표 또는 상표입니다. 본문 중에는 ™, ©, ® 마크 등이 표시되어 있지 않습니다.
- 본문 중 일본 내의 실정에만 국한되어 있는 내용의 경우, 일부를 삭제하거나 국내 실정에 맞도록 변경하였으니 참고 바랍니다.
- 이 책의 소스 코드는 제이펍 홈페이지(www.jpub.kr)의 상세 소개 페이지에서 다운로드하실 수 있습니다.
- 책에 관한 문의사항은 옮긴이(jinsilto@gmail.com) 또는 제이펍 출판사(jeipub@gmail.com)로 연락주시기 바랍니다.

옮긴이 머리말 XII
머리말 XIV
베타리더 후기 XV

PART 1
Hadoop 기초 1

Chapter 1 Hadoop 기초 지식 3

1.1 Hadoop이란? 4
1.2 Hadoop 개요 4
1.3 Hadoop 적용 분야 15
1.4 Hadoop 시스템 구성과 아키텍처 19
1.5 Hadoop 적용 사례 30

Chapter 2 MapReduce 애플리케이션 활용 예 35

2.1 블로그 접속 수 집계 36
2.2 비슷한 사람을 찾아내자 41
2.3 검색 엔진 인덱스 작성 43
2.4 이미지 데이터 분산 처리 46
2.5 MapReduce로 구현할 수 있는 처리 특성과 활용 포인트 48

Chapter 3 Hadoop 도입 51

3.1 구축 환경 조건 52
3.2 실행 환경 구축 55
3.3 동작 확인 59
3.4 샘플 애플리케이션 실행 63

Chapter 4 HDFS 65

4.1 Hadoop의 파일 시스템 66
4.2 HDFS의 구조 68
4.3 HDFS의 파일 I/O 흐름 73
4.4 파일 시스템의 메타데이터 77
4.5 HDFS 설정과 시작/중지 81
4.6 SecondaryNameNode 92
4.7 CLI 기반 파일 조작 94

Chapter 5 MapReduce 프레임워크 109

5.1 MapReduce 처리 110
5.2 MapReduce 아키텍처 114
5.3 MapReduce와 HDFS의 관계 118
5.4 MapReduce 프레임워크 설정 123
5.5 MapReduce 프레임워크 시작과 정지 129
5.6 CLI 기반 MapReduce 관리 136

Chapter 6 Hadoop 애플리케이션 실행 143

6.1 테스트 애플리케이션 144
6.2 MapReduce 애플리케이션(자바) 144
6.3 HadoopStreaming 애플리케이션 150
6.4 Pig 애플리케이션 153
6.5 Hive 애플리케이션 160

Chapter 7 Hadoop 클러스터 구축 167

7.1 완전 분산 클러스터 168
7.2 환경 구축의 전제 조건 168
7.3 Hadoop 클러스터 환경 구축 169
7.4 웹 인터페이스를 통한 동작 확인 178
7.5 샘플 애플리케이션 실행 181

PART 2
MapReduce 애플리케이션 개발 185

Chapter 8 MapReduce 프로그래밍 기초 - 자바를 사용한 개발(1) - 187

8.1 자바를 사용한 MapReduce 개발 188
8.2 소스 코드 단계에서 프로그램 실행 190
8.3 Mapper 194
8.4 Reducer 197
8.5 Writable 199
8.6 InputFormat/OutputFormat 203
8.7 main 프로그램 207

Chapter 9 MapReduce 프로그래밍 응용 - 자바를 사용한 개발(2) - 215

9.1 MapReduce 프로그램 커스터마이즈 216
9.2 입력 데이터 조작을 제어한다 216
9.3 출력 데이터 제어 226
9.4 독자 데이터형을 정의한다 229
9.5 Shuffle 단계 동작 제어 231
9.6 MapReduce 애플리케이션 테스트 및 디버깅 235

Chapter 10 MapReduce 개발 팁 -자바를 사용한 개발(3) - 243

10.1 이 장에서 소개할 내용 244
10.2 압축 데이터 다루기 244
10.3 MapReduce 처리에 적합한 파일 포맷 247
10.4 한 번에 복수의 파일 포맷 처리 249
10.5 분산 캐시 이용 252
10.6 Map 태스크/Reduce 태스크 실행 Context 처리 254
10.7 처리 중에 발생한 이벤트를 집계 256
10.8 MapReduce 애플리케이션 설정을 제어 259

Chapter 11 HadoopStreaming 263

11.1 HadoopStreaming이란? 264
11.2 HadoopStreaming 애플리케이션 구성 264
11.3 HadoopStreaming 활용 예 270
11.4 HadoopStreaming 명령어 옵션 275

Chapter 12 데이터 흐름형 처리 언어 Pig 283

12.1 Pig란? 284
12.2 Pig 실행 방법 286
12.3 Pig의 데이터 관리 288
12.4 Pig에서의 처리 방법 290
12.5 함수 298
12.6 애플리케이션 개발 301
12.7 Pig Latin 처리 구현 303
12.8 사용자 정의 함수 구현 306
12.9 Pig 디버그 313
12.10 Pig를 활용하기 위한 포인트 315

Chapter 13 SQL 유사 인터페이스 Hive 319

13.1 Hive 개요 320
13.2 Hadoop과 Hive 321
13.3 HiveQL 324
13.4 효율적인 Hive 활용법 347

PART 3
Hadoop 클러스터 구축과 운영 359

Chapter 14 환경 구축 효율화 361

14.1 환경 구축과 설정 자동화 362
14.2 환경 구축 서버 설치 365
14.3 OS 자동 설치 373
14.4 Puppet을 사용한 환경 설정 377
14.5 PSSH를 이용한 운영 388
14.6 Hadoop 설정의 베스트 환경 392

Chapter 15 가용성 향상 395

15.1 고가용성의 기본 396
15.2 HDFS 고가용성 401
15.3 MapReduce 고가용화 413

Chapter 16 클러스터 모니터링 425

16.1 Hadoop 모니터링 426
16.2 Ganglia를 통한 Hadoop 클러스터 메트릭스 취득 437
16.3 Hadoop 클러스터 감시 445

Chapter 17 클러스터 운영 459

17.1 Hadoop 운영이란? 460
17.2 운영 시 구체적인 작업 462
17.3 정기적으로 실시해야 할 작업 474
17.4 Hadoop 클러스터 확장 479
17.5 클러스터 장애 대응 481

Chapter 18 복수 사용자에 의한 리소스 제어 493

18.1 리소스 제어 도입 494
18.2 기본 설정 495
18.3 스케줄러를 사용한 리소스 분배 제어 500

PART 4
Hadoop 활용 기술 521

Chapter 19 Hadoop 튜닝 523

19.1 튜닝이 필요한 케이스 524
19.2 MapReduce 잡 동작에 영향을 주는 설정 524
19.3 Map 태스크 튜닝 527
19.4 Reduce 태스크 튜닝 532
19.5 자바 VM 튜닝 535
19.6 OS 튜닝 545

Chapter 20 분산형 데이터베이스 HBase 551

20.1 HBase란? 552
20.2 HBase 데이터 모델 554
20.3 아키텍처 555
20.4 설치 방법 558

20.5 HBase Shell 563
20.6 MapReduce 잡 활용 572
20.7 HBase 테이블 설계 578

Chapter 21 **Fluentd 로그 수집기** **581**

21.1 로그 수집기 등장 배경 582
21.2 Fluentd란? 584
21.3 모든 로그를 JSON으로 587
21.4 Fluentd 설치 588
21.5 Fluentd 기본 590
21.6 HDFS에 기록하기 위한 설정 예 593
21.7 Fluentd 내부 구조 595
21.8 Fluentd HA 구성 598
21.9 Fluentd 튜닝 601
21.10 Fluentd 모니터링 602
21.11 Fluentd와 유사한 소프트웨어 604

Chapter 22 **YARN** **605**

22.1 YARN 기초 지식 606
22.2 YARN 아키텍처 611
22.3 Hadoop/MapReduce/
YARN과 기존 Hadoop/MapReduce의 차이 613
22.4 YARN을 이용한 MapReduce 실행 615

찾아보기 **629**

아마 최근 가장 주목을 받고 있는 IT 분야는 빅 데이터일 것이다. 많은 기업은 자신들이 가지고 있는 기존 데이터들을 어떻게 의미가 있는 정보로 바꿀지 고민하고 있다. 의미가 있는 정보란, 매일 같이(지금 이 순간에도) 발생하고 있는 대량의 데이터를 실시간으로 가공, 처리해서 해석 가능한 정보로 변환한 것이라 할 수 있다. 이 의미 있는 정보를 창출해 내기 위한 핵심 플랫폼이 바로 Hadoop(하둡)이다.

이 책은 Hadoop의 기존 개념과 구축 방법에 대해 상세히 설명하고 있을 뿐 아니라, 다양한 활용 방법과 각 설정 항목에 대해 상세한 설명으로 Hadoop을 어떻게 튜닝할지에 대한 비법도 알려준다. 저자들은 실제 Hadoop을 운영하면서 얻은 경험을 바탕으로 다양한 운영 상황에 대한 상세한 대처 방법을 설명하고 있으며, 독자들이 맞닥뜨릴 수 있는 거의 모든 문제에 대한 해답을 알려주고 있다 해도 과언이 아닐 정도로 완벽한 책이다.

이 책은 그야말로 Hadoop의 백과사전이라고 할 수 있다. 한 번만 읽고 책꽂이로 돌아가는 책이 아니라, 필요할 때마다 꺼내서 참고할 수 있는 내용이 다양하게 수록되어 있다. Hadoop을 운영하면서 문제가 발생할 때 이 책이 문제 해결 방법을 제시해 줄 것이다. 또한, 부하 증가나 데이터 증가로 튜닝이 필요할 때 이 책이 튜닝 비법도 알려줄 것이다. Hadoop의 A부터 Z까지 모든 것을 다루고 있는 백과사전으로, 대용량 데이터 처리를 위한 실전 참고서가 될 것이라 믿어 의심치 않는다.

게다가, 이 책은 Hadoop뿐 아니라 주변 연계 소프트웨어에 대해서도 상세히 설명하고 있다. Pig, Hive, Hbase, Fluentd 등에 대한 설치 방법과 Hadoop 연동 방법, 그리고 샘플 예제를 수록하고 있어서 Hadoop뿐 아니라 다양한 연계 시스템에 대한 지식을 얻을 수도 있다. 이 책만 제대로 마스터한다면 빅 데이터 시대의 IT 전문가로 거듭날 수 있을 것이다.

끝으로, 좋은 책을 번역할 기회를 주신 장성두 실장님께 감사의 말을 전한다. 그리고 늘 우리 가정을 지켜주시는 하나님께도 감사드리고, 나의 반쪽인 희운이에게도 사랑한다는 말을 전하고 싶다.

2014년 5월 네덜란드에서

옮긴이 **김완섭**

독자 여러분의 성원에 힘입어 《빅 데이터 시대의 하둡 완벽 입문(제2판)》을 출간하게 되었다. 1판을 낸 후 2판을 준비하는 동안 사용자층이 많이 넓어졌다. Hadoop 적용 사례가 늘어나고, Hadoop을 적극적으로 도입해 성공한 사례들이 늘어나고 있다. 또한, 이런 성공 사례가 바탕이 됨으로써 Hadoop 사용 분야가 날로 확대되고 있다.

그러나 아직 Hadoop을 잘 모르는 사람도 있고, 본격적으로 이용하려고 도입은 했으나 운영에 어려움을 겪고 있는 사람이 많은 듯하다. 그래서 이번 《빅 데이터 시대의 하둡 완벽 입문(제2판)》을 출판하면서 Hadoop을 전혀 모르는 사람들이 이해하는 데 좀 더 도움이 될 수 있도록 1장과 2장 내용을 보강하여 Hadoop에 대한 개요 부분을 좀 더 쉽게 구성했다. 또한, MapReduce 애플리케이션 개발 내용을 추가하여 초보자부터 고급 사용자까지 활용할 수 있도록 했다. 이 책의 특색인 운용성과 가용성 측면도 1판에 비해 보강했으며, 새로운 기술 이슈에 대해서도 추가하였다.

1부에서는 Hadoop을 사용하기 위해 이해해야 할 핵심 포인트들을 농축하여 수록했다. 혹시 책의 내용을 따라잡을 수 없게 된다면, 1부를 다시 읽어주기 바란다. 그리고 반드시 Hadoop을 실제로 동작해 보기 바란다. 후반부에 있는 장들은 각각 독립되어 있어 순서대로 읽어도 되고, 필요한 부분을 선택해서 먼저 읽어도 된다.

마지막으로, 이 책을 집필할 기회를 주신 분들께 감사의 마음을 전하는 바다. 아울러 책 집필을 위해서 많은 도움을 주신 분들과 가족들에게도 감사하다는 말을 전하고 싶다.

이 책이 Hadoop을 배우려는 분들과 Hadoop을 사용해서 개발 및 운영을 하고자 하는 분들께 조금이나마 도움이 될 수 있기를 바란다.

저자 일동

김인범(SK C&C)

Hadoop에 대해 이보다 더 자세한 가이드는 없을 것입니다. 필요한 정보가 충분히 수록되어 있고, 환경 설정 구성 및 운영 시 발생할 수 있는 상황에 대한 대처법도 적절하게 포함되어 있습니다. 그야말로 '완벽 가이드'로서 적극적으로 추천합니다.

윤재석(GDG Soongsil University)

이 책은 빅 데이터나 Hadoop과 같은, 말로만 들어도 어려운 내용에 한걸음 더 가까이할 수 있도록 도와주는 책입니다. 책 전반부에는 빅 데이터에 다가가기 위한 기본 개념을 제공한다면, 책 후반부에는 Hadoop으로 빅 데이터를 활용할 수 있는 다양한 방법을 예제로서 제시합니다. 그동안 어렵다는 이유로 빅 데이터와 Hadoop을 가까이할 수 없었다면, 이 책을 통해 조금씩 가까워질 기회를 마련해 보시기를 권합니다.

정연우(투씨에스지)

여태까지 읽은 Hadoop 입문서 중에서 가장 이해하기 쉽고, 짜임새가 잘 잡혀 있는 도서인 것 같습니다. 이 책으로 Hadoop과 Hadoop의 대표적인 에코시스템을 이해한 뒤, 나아가 Hadoop 심화 서적을 읽어 보고 응용해 본다면 더욱 빨리 배울 수 있을 것입니다.

조현석(다음 커뮤니케이션즈)

Hadoop을 처음 접할 때는 왠지 어려울 것 같아 걱정이 많았는데, 이 책은 친절한 선배가 옆에서 차근차근 알려주는 것처럼 비법을 친절히 전수해 준다는 느낌이 들었습니다. Hadoop 생태계 전반에 관한 개념을 잡는 데 도움이 될 뿐만 아니라 성능 튜닝에 대해서도 충분히 이해할 수 있었습니다.

최지웅(스윗트래커)

어느덧 Hadoop이라는 존재가 익숙해진 지도 꽤 많은 시간이 흘렀습니다. 그렇지만 쉽게 환경을 구축해 보거나, 대체 어느 분야에 어떻게 써야 할지 무척 막막했었는데요. 그런 의미에서 이 책은 그 궁금증을 해결하는 데 많은 도움이 되었습니다. 특히, 개념 설명 위주로 이루어진 3장까지는 반드시 정독할 것을 권합니다.

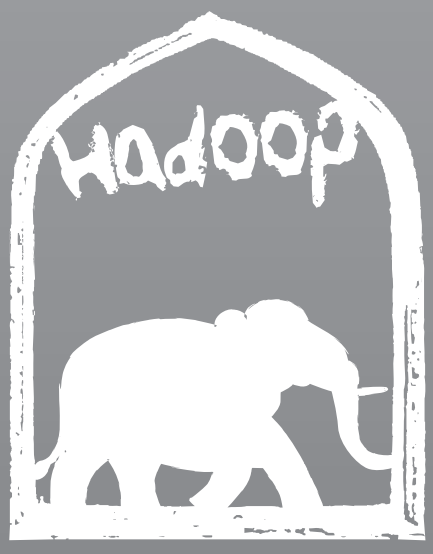
최해성(티켓몬스터)

Hadoop과 맵 리듀스에 관련된 내용을 예제를 중심으로 쉽게 이해할 수 있도록 만든 책이었습니다. 기본 개념, Hadoop 기본 구현과 클러스터를 구축하는 과정으로 넘어가는 부분, 그리고 클러스터를 모니터링하는 방법 등 아키텍처를 설계하는 데 필요한 전반적인 내용을 깔끔하게 설명해 줍니다. 일본 저자의 책은 믿을 수 있다는 제 견해를 다시 한 번 확신할 수 있게 해준 책이었습니다.

제이펍은 책에 대한 애정과 기술에 대한 열정이 뜨거운 베타리더들로 하여금
출간되는 모든 서적에 사전 검증을 시행하고 있습니다.

PART 1

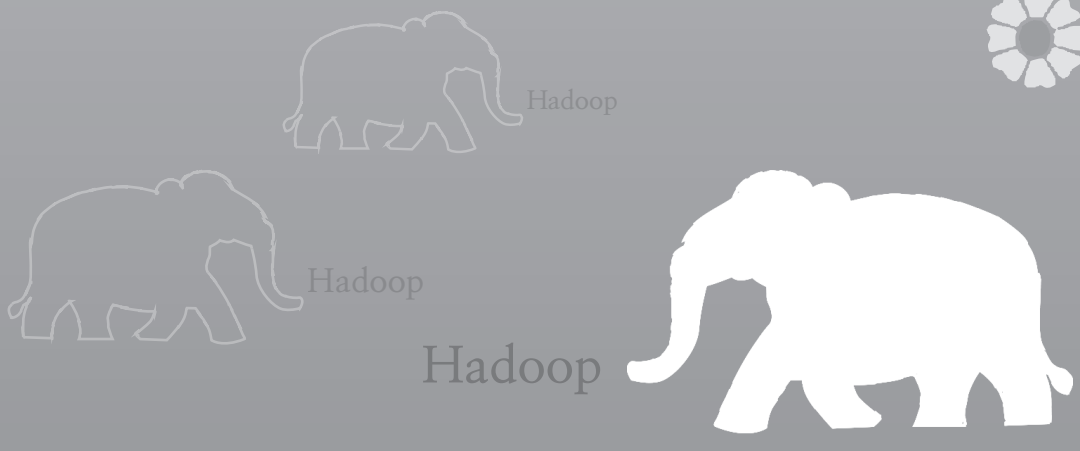
Hadoop 기초



Chapter

1

Hadoop 기초 지식



1.1 Hadoop이란?

IT 시스템 세계에서 'Hadoop(하둡, 이하 Hadoop)'이라는 소프트웨어 사용이 늘어나고 있다. 이 책을 가지고 있다는 것은 적어도 Hadoop에 관심을 가지고 있을 것이라 생각된다. 아직 Hadoop에 대해 잘 몰라도 소프트웨어 이름이라는 것 정도는 알고 있겠지만, '도대체 용도가 뭐야?', '뭐 하는 거지?' 하고 묻는 사람도 적지 않을 것이다. 어떤 데이터를 취급하기 위한 미들웨어라는 것은 알고 있더라도 '관계형 데이터베이스랑은 어떻게 달라?', '파일 시스템이랑 달라?', '처리를 위한 새로운 개념인가?' 등 다양한 의문이 많을 것이다.

이 장에서는 Hadoop 전체를 이해하기 위해서 Hadoop이 무엇인지, 어떤 구조로 이루어져 있고 어떤 식으로 사용하는지를 소개하겠다.

1.2 Hadoop 개요

우선 구체적으로 Hadoop이 무엇인지, 어떻게 등장하게 됐는지를 소개하고, 대규모 데이터 처리의 과제와 대응에 대해 설명하겠다.

1.2.1 Hadoop을 한마디로 정의하자면

■ 대량의 데이터를 처리하기 위한 병렬 분산 처리 소프트웨어

오늘날의 IT 시스템은 예전에는 생각할 수 없었을 정도의 대규모 데이터를 다루어야 하는 현실에 직면했다. 그 양은 몇 대의 데이터베이스 서버와 데이터 웨어하우스 서버로는 처리할 수 없을 정도다.

대량의 데이터를 처리할 때는 해결해야 할 몇 가지 문제가 존재하지만, 그중에서 피하기 어려운 것은 디스크 I/O 성능(throughput, 스루풋)이다. 데이터양이 적으면 메모리를 효율적으로 사용해서 해결할 수 있지만, 메모리 용량을 크게 상회하는 데이터를 다룰 때는 디스크 I/O가 직접적으로 영향을 받는다. 그러나 컴퓨터 역사를 살펴보면, CPU 성능 진화나 내장 메모리의 대용량화와 비교해서 디스크 I/O의 성능은 그다지 향상되지 않은 게 현

실이다. 이러한 배경을 바탕으로 사용이 늘고 있는 것이 바로 Hadoop이다.

Hadoop을 한마디로 표현하면 '대량의 데이터를 처리하기 위한 병렬 분산 처리 소프트웨어'라고 할 수 있다. '대량의 데이터'란 테라바이트(TeraByte, TB)나 페타바이트(PetaByte, PB) 급의 대용량 데이터 또는 수억, 수조 건의 데이터를 의미한다. '병렬 분산 처리'란 단어 하나하나를 보면 어려운 느낌이 들지만, 쉽게 표현하자면 하나의 처리를 복수의 서버로 동시 처리하는 것을 말한다. Hadoop에서는 10대~1,000대의 서버로 하나의 클러스터(cluster)를 구성하여 대용량 데이터를 저장하고 처리한다.

하이퍼바이저(Hypervisor) 등의 '가상화 기술'은 한 대의 서버를 가상으로 복수의 서버로서 인식하게 하는 기술이지만, 반대로 Hadoop은 복수의 서버를 묶어서 하나의 가상 서버로 사용할 수 있도록 하는 것이다.

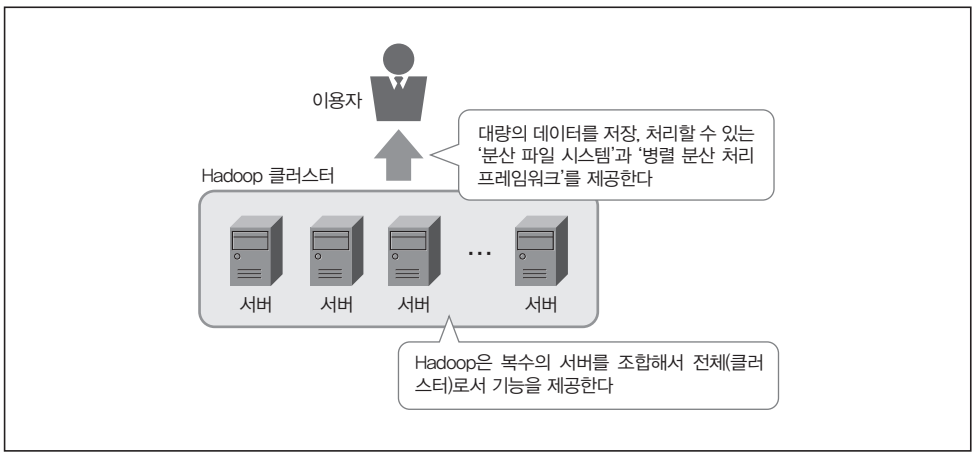


그림 1.1 Hadoop은 복수의 서버를 조합해서 하나의 클러스터로 구성

■ 분산 파일 시스템과의 강한 연계를 통해, 높은 스루풋 처리를 실현하는 분산 처리 소프트웨어

상세한 내용은 뒤에서 설명하겠지만, Hadoop은 크게 두 가지 구성 요소로 이루어져 있다. 하나는 분산 파일 시스템으로, 대용량 데이터를 복수의 서버에 저장하는 구조다. 복수의 서버를 조합해서 사용자에게 하나의 큰 파일 시스템을 제공한다. 다른 하나는 병렬 분산 처리를 실현하는 프레임워크다. 이것은 하나의 큰 처리(잡, job)를 복수의 단위(태스크, task)로 분할해서 실행하는 구조다. 이 분산 파일 시스템과 병렬 분산 처리 프레임워크가 공고히 연계되며, 높은 스루풋의 데이터 처리를 실현한다.

기술적인 관점에서 Hadoop을 한마디로 표현하면, '분산 파일 시스템과 강한 연계를 통해 높은 스루풋 데이터 처리를 실현하는 분산 처리 소프트웨어'라고 할 수 있다.

■ 자바(Java) 기반으로 개발되어 일반적인 서버에서 동작

Hadoop은 주로 자바를 개발 언어로 개발되었다. Hadoop을 설치할 하드웨어에 특수한 서버나 장비는 필요치 않다. IA(Intel Architecture) 서버같이 일반적으로 널리 사용되고 있는 서버와 네트워크 장비로도 충분한 효과를 볼 수 있다. '언제나 구할 수 있는 장비로도 구축이 가능하다.'는 점이 Hadoop의 매력 중 하나다. 자바로 개발되어 있기 때문에 다양한 OS에서 동작하지만, 보통은 리눅스 상에 구현하는 것이 일반적이다.

■ 서버를 추가하면 용량과 성능이 향상(스케일 아웃, Scale-out)

Hadoop에서는 클러스터를 구성하는 서버 대수를 추가하면 저장 용량을 확장하거나 분산 처리 성능을 향상시킬 수 있다. 서버 자체의 성능을 올리는 스케일 업(Scale-up) 방식으로는 성능을 올리는 데 한계가 있으므로, 서버 대수를 증설하는 스케일 아웃 방식이 가능하다는 것은 큰 이점이다. 실제 Hadoop의 처리 능력은 1,000대 정도로 구성할 때 제대로 된 효과를 볼 수 있다고 한다. 만약 두세 대로 구성한 클러스터라면 제대로 된 Hadoop의 매력을 느낄 수 없기 때문에 최소한 10대 정도를 이용하여 장점을 이끌어 낼 수 있도록 하는 것이 좋다.

■ 관계형 데이터베이스나 검색 엔진과는 다름

Hadoop은 '고속 데이터베이스 소프트웨어', '검색 엔진을 구현하는 소프트웨어'라고 오해를 사기도 한다.

Hadoop은 관계형 데이터베이스와는 전혀 다른 방식으로 만들어진 소프트웨어다. 관계형 데이터베이스는 정규화된 작은 데이터 세트에 대해 낮은 대기 시간을 가지고(수 밀리초나 수 초) 액세스하도록 만들어졌다. 하지만 Hadoop은 대량의 데이터를 대상으로 일괄 처리를 실행한다. 또한, 관계형 데이터베이스가 가지고 있는 트랜잭션(Transaction) 제어 등의 기능도 없다. 데이터를 누적해서 처리한다는 면에서는 같은 용도지만, 사전에 기능이나 용도가 크게 다르다는 것을 이해해 두는 것이 중요하다. 그리고 Hadoop은 검색 엔진도 아니다. 검색 인덱스를 만들 때도 자주 사용되나, 검색 요청에 바로 응답할 수 있는 검색 엔진 소프트웨어는 아니다.

■ 가장 활발한 오픈 소스 프로젝트

Hadoop은 Apache Software Foundation(아파치 소프트웨어 파운데이션)의 프로젝트로 Apache(아파치) 라이선스로 공개되어 있으며(그림 1.2), 현재 가장 활발한 오픈 소스 프로젝트 중 하나라고 할 수 있다. 대규모 데이터 처리를 필요로 하는 대기업 소속 엔지니어들이 직접 개발에 참여하고 있어서 나날이 진화를 거듭하고 있다.

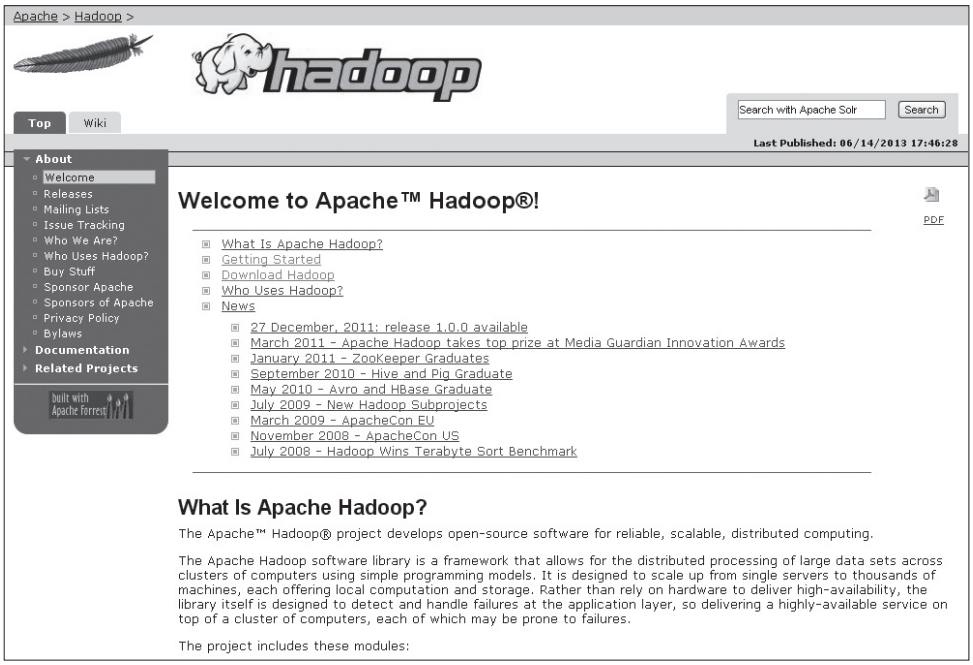


그림 1.2 Apache Hadoop 사이트(http://hadoop.apache.org/)

1.2.2 Hadoop의 용도

'대량의 데이터를 고속으로 처리할 수 있다.'라는 특성을 적용할 수 있는 분야 중 하나는 IT 자산 관리를 위해 사용되고 있는 일괄(Batch) 처리 변경이다. 하지만 반드시 기존 처리를 바꾸어야 하는 것은 아니다. Hadoop은 만능 미들웨어가 아니기에 기존의 모든 처리를 Hadoop으로 바꿀 수 있는 것은 아니다. 기존 시스템을 바꾸는 것뿐만 아니라, '지금까지 할 수 없어서 포기했던 것들이 가능해진다.'는 '새로운 가치'를 제공하는 것이 Hadoop의 역할이다. 실제로 Hadoop이 등장하면서 현실적으로 실현 불가능했던 처리가 가능해지거나, 높은 비용으로 인해 포기했던 처리를 현실적인 비용으로 구현할 수 있게 되었다.

이런 '새로운 가치'에 주목하여 Hadoop을 적극적으로 도입한 기업으로는 미국 야후(25,000대 규모), 페이스북(1,000대 규모) 등을 들 수 있다. 이 외에도 비자(VISA), JP 모건 체이스(MORGAN CHASE), 바이두(Baidu)[1], 차이나 모바일(China Mobile) 등이 있다.

일본의 경우는 웹 서비스를 제공하는 기업뿐만 아니라 통신, 소매업, 유통업, 금융기관 등도 도입하고 있다[2]. 지금은 로그 분석, 추천 엔진(Recommendation Engine), 검색 인덱스 생성 등에 주로 사용되고 있다. 이후에는 데이터 웨어하우스(Data warehouse)나 비즈니스 인텔리전스 분야에 활발히 적용될 것으로 예상된다.

1.2.3 Hadoop의 탄생

Hadoop이 어떻게 이 세상에 나오게 됐는지 살펴보도록 하자. 모두가 알고 있는 거대 기업 구글은 세상에 있는 모든 데이터를 대상으로 하는 검색 엔진을 제공하기 위해 기존 미들웨어로는 실현할 수 없는 시스템 기반이 필요했다. 그러나 세상 모든 웹 사이트에 존재하는 데이터를 수집하여 저장하기 위해서는 기존 저장 체계나 파일 체계로는 한계가 있었다. 뿐만 아니라, 수집한 대량의 데이터에서 검색용 인덱스를 생성하기 위해서는 기존 데이터베이스나 데이터 웨어하우스 구조로는 어려웠다. 그래서 구글은 대량의 웹 사이트 데이터를 저장하고 처리할 수 있는 미들웨어를 자체적으로 개발하여 사용하기 시작했다.

그 기술 중 일부가 2003년 〈The Google File System〉, 2004년 〈MapReduce: Simplified Data Processing on Large Clusters〉라는 두 가지 논문을 통해 발표되었다. 그때까지 철저히 베일에 가려져 있던 검색 엔진 기반 기술이 드디어 세상에 공개된 것이다. 이 논문을 통해 발표된 기술적 아이디어를 바탕으로 하여, 오픈 소스로 구현하기 시작한 것이 Hadoop 프로젝트의 시작이다[3]. 원래 이 아이디어는 구글이 검색 엔진의 핵심 기능[4]을 보완하기 위해 만들어 낸 것이다. 하지만 그 이외의 용도로도 폭넓게 사용되어 대규모 데이터를 높은 스루풋으로 처리할 수 있는 소프트웨어인 Hadoop을 진화시키는 발판이 되었다.

---

1 중국의 대표적인 인터넷 포털 회사다.

2 매년 개최되고 있는 'Hadoop Conference Japan'에서 다양한 사례들이 발표되고 있다. 발표 자료들도 많이 공개되었다.

3 논문에 쓰여져 있던 아이디어에는 구글이 특허를 가진 것도 있지만, 현재는 오픈 소스 프로젝트인 Hadoop이 그것을 사용할 수 있도록 허락하고 있다.

4 예를 들어, 구글은 PageRank라 불리는 값을 계산하고 있었다.

■ Hadoop과 구글 기술과의 관계

구글 논문을 통해 발표된 '구글 파일 시스템(Google File System)'은 대량의 데이터를 복수의 서버에 저장할 수 있는 파일 시스템이다. Hadoop에서 구글 파일 시스템에 해당하는 것이 HDFS(Hadoop Distributed File System)다.

한편 'MapReduce(맵리듀스. 이하 MapReduce)'는 구글 파일 시스템에 저장되어 있는 대량의 데이터를 분산 처리하기 위한 알고리즘 프레임워크다. Hadoop에서는 Hadoop MapReduce[5]가 이에 해당한다. 이 두 가지 외에도 연동 소프트웨어에 대한 아이디어도 논문을 통해 공개하고 있다. Hadoop 프로젝트에서는 HDFS와 MapReduce 뿐만 아니라 오픈 소스 개발도 함께 진행하고 있다.

표 1.1 구글 기술과 Hadoop 대응 관계

처리 내용	구글	Hadoop
분산 파일 시스템	GFS: Google File System	HDFS: Hadoop Distributed File System
분산 처리 프레임워크	MapReduce	Hadoop MapReduce
키 밸류형 데이터 스토어	BigTable	HBase

Hadoop 개발은 2005년부터 더그 커팅(Doug Cutting)을 중심으로 시작되었다. 더그 커팅은 원래 루씬(Lucene)이나 넛치(Nutch)라고 하는 오픈 소스 프로젝트 개발자였다. 루씬은 전문 검색 엔진이고, 넛치는 웹 크롤러(Web Crawler)로 루씬 상에서 동작하는 애플리케이션이다. Hadoop의 MapReduce는 넛치를 발전시키기 위해서 탄생했다고 보고 있다. 더그 커팅은 Hadoop을 오픈 소스로 개발하기 위해 노력했다. 그리고 '누구든 이 툴이 가진 우수함을 누릴 수 있다.'는 것을 강력히 주장했다.

대규모 병렬 분산 처리를 누구든지 쉽게 이용할 수 있고 수많은 사람들이 이용함으로 많은 피드백을 수집할 수 있다. 이렇게 수집한 피드백은 품질 향상으로 이어지고, 이런 과정을 반복할 수 있다는 것이 것이 오픈 소스화의 장점이다. 실제로 지금도 많은 개발자가 Hadoop 개선에 참여하고 있으며, 아주 빠른 속도로 진화하고 있다.

5 이후부터 MapReduce라고 지칭하는 경우, 모두 Hadoop MapReduce를 의미한다.

Hadoop 명칭의 유래

Hadoop이라는 명칭의 유래는 더그 커팅의 아들이 가지고 있던 코끼리 인형의 이름이었다고 한다. Hadoop이라는 단어 자체에 특별한 의미가 있는 것은 아니다. Hadoop의 노란색 코끼리 로고도, 코끼리 인형을 모티브로 만들어졌다.

■ 병렬 분산 처리 시스템

MapReduce가 실현하고 있는 기능은 병렬 분산 처리를 연구하던 기술자나 연구자들에게 있어서 새로운 기술이 아니었다[6]. 즉 예전부터 존재했던 처리 중 하나로, 해당 분야에서는 '상식'적인 부류에 속하는 기술이다. 하지만 기존 병렬 분산 처리 기술은 특정 하드웨어 환경에서만 동작하거나, 특정 소프트웨어 상에서만 동작하는 '고급형'이었다. 이런 기술은 매우 고가였으며, 일반인의 접근이 어려웠다. 그러나 Hadoop은 누구든지 기술을 사용할 수 있는 '범용적인' 형태로 제공하고 있다는 것이 큰 이점이다.

게다가, Hadoop이 값비싼 하드웨어가 아닌, '일반적인' 하드웨어에서도 동작하는 것 또한 큰 특징이다. 복수의 서버를 이용하는 분산 처리에서는 장비 고장에 따른 영향을 줄이기 위해서 비싼 장비를 사용하는 경우가 적지 않다. Hadoop 설치가 용이한 IA 서버를 전제로 한 설계는 많은 엔지니어와 기업들이 도입하여 비용 대비 효율을 높이고 있다. Hadoop에서는 일부 서버가 고장 나도 데이터를 잃거나 처리 실패가 일어나지 않도록 구성되어 있다.

또한, 분산 처리 애플리케이션 개발이라는 관점에서도 Hadoop이 그 문턱을 많이 낮추었다고 말할 수 있다. 병렬 분산 처리를 실현하려고 하면, 하나의 처리(잡)를 어떤 단위(태스크)로 분할할 것인지, 해당 태스크를 어떤 컴퓨터에서 실행할 것인지, 각 태스크 결과를 어떤 방식으로 모을 것인지, 도중에 서버 고장으로 태스크 실행에 실패했을 때 어떻게 복구할 것인지 등 많은 것을 고려해야 한다(그림 1.3).

6 함수형 언어에서 사용되는 map 함수와 reduce 함수에서 아이디어를 얻어 만들어졌다고 한다.

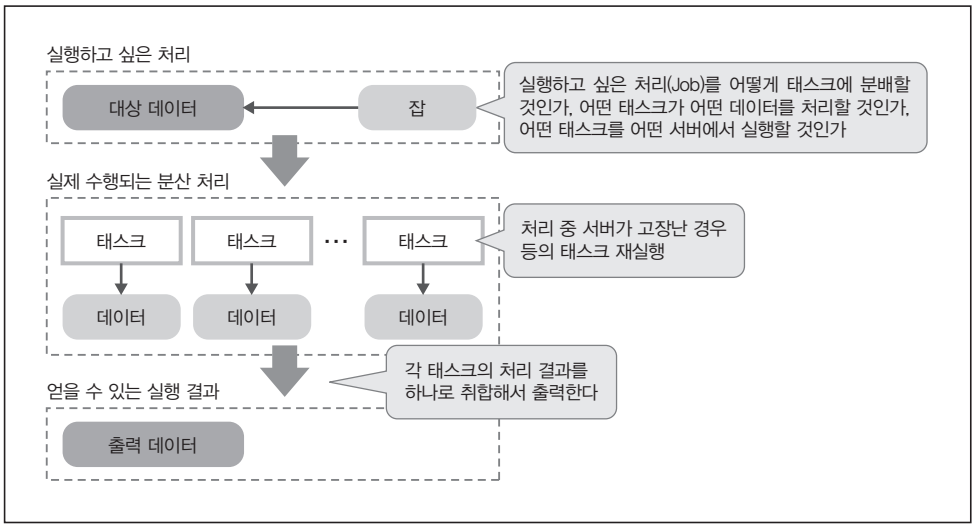


그림 1.3 분산 처리 문제(의 일부)를 자동적으로 해결해 준다

그러나 Hadoop에서는 MapReduce라는 프레임워크에 따라 애플리케이션을 개발하면 이런 문제를 자동적으로 해결할 수 있다. 개발자는 그저 개별 애플리케이션에 특화된 내용만을 기술하면 된다[7]. 이와 같은 '범용화'가 Hadoop 사용이 급속히 늘어난 이유 중 하나라고 볼 수 있다.

1.2.4 대규모 데이터 처리의 문제점과 대응

지금까지 '대규모 데이터 처리 필요성의 증가'라고 언급해왔는데, 실제 어느 정도의 데이터를 다룰 필요가 있으며 그로 인해 어떤 문제가 발생하고 있는 것일까? 세상에 공개되어 있는 숫자들을 통해서 확인해 보도록 하자.

■ 시스템이 취급하는 데이터 규모의 증가

'정보 폭발'이라는 키워드가 등장하고 시간이 꽤 흘렀지만 블로그/SNS/트위터라는 사용자 커뮤니케이션 서비스가 계속 등장하고, 유튜브(YouTube)를 시작으로 한 동영상 투고 사이트가 등장하면서 전 세계에서 발생하는 데이터양이 급격히 증가했다. 현재로서는 이용

7 일부의 각 태스크 간 처리 대상 데이터양이 많아지는 등, 애플리케이션 개발자가 해결하지 않으면 안 되는 문제도 있다.

가능한 정보량이 너무 많아, 관리조차 마음대로 되지 않는 상태다. 미국 시장 조사 회사인 IDC가 발행한 〈Digital Universe〉 2010년 보고서에 따르면, 2009년에 전 세계에서 생성된 데이터양은 전년도 대비 62% 증가한 0.8제타바이트(ZettaByte, ZB)(=8,000억 GB)라고 한다. 또한, 2020년에 전 세계에서 만들어지는 데이터양은 35ZB에 달할 것으로 예측하고 있어, 10년 동안 44배 가량 증가할 것이라 보고 있다[8].

제타바이트라고 하면 어느 정도의 숫자인지 실감이 나지 않을 수도 있다. 하지만 세계적으로 활약하는 기업들이 다루고 있는 데이터양의 관점에서 보면 NYSE(New York Stock Exchange)가 매일 1TB 데이터를 생성하고, 페이스북은 매일 20TB 이상의 압축 데이터를 만들어 내고 있으며, CERN: the European Organization for Nuclear Research(유럽 원자력 연구 기관)가 매일 40TB, 연간 15PB의 데이터를 생성하고 있다.

■ 대량의 데이터를 처리하기 위한 구조

대량의 데이터를 처리하려면, 우선 대상 데이터를 높은 처리량으로 읽어 들여야 한다. 하지만 통상적인 하드디스크 처리 성능,' 예를 들어 일반 SATA 디스크 하나로 읽어 들이는 성능은 실 속도로 약 70MB(메가바이트)/초 정도다. 1TB 데이터를 읽으려면 약 14,000초 = 약 3.9시간이 걸린다. SSD도 대용량화, 저가격화가 진행되고 있지만 기껏해야 수배 정도 빠른 성능이다.

따라서 대용량 데이터를 효율적으로 읽어 들이기 위해서는, 복수의 디스크에 데이터를 기록해 두고 각 디스크에서 병행하여 읽을 수밖에 없다. 예를 들어, 한 대의 SATA 디스크라면 70MB/초 처리량밖에 안 되지만, 40대로 동시에 읽어 들이면 2,800MB/초 처리량을 구현할 수 있다. 1TB 데이터도 약 350초면 된다. 만약 1,000대의 디스크를 사용한다면 14초만에 읽을 수가 있다.

Hadoop 분산 파일 시스템인 HDFS는 각 서버의 내장 디스크에 데이터를 저장한다. 이때 대용량 파일을 작은 단위(블록)로 분할하여 복수의 서버에 나누어 배치한다. 예를 들어, 블록(Block) 크기가 64MB고 데이터가 1GB(1,024MB)라고 하면, 16개 블록으로 분할하여 복수의 서버에 배치한다(그림 1.4). 서버가 충분히 있어서 16개 블록이 모두 개별 서버에 배치된다고 하면, 이 데이터를 읽어 들이기 위해서 서버 16대를 동시에 사용하는 것이다. 한 대의 디스크(서버)를 이용하는 경우와 비교하면 16배 스루풋을 실현할 수 있다. 이와 같

8 킬로10^3 → 메가10^6 → 기가10^9 → 테라10^{12} → 페타10^{15} → 엑사10^{18} → 제타10^{21}

이 Hadoop에서는 10대~1,000대 서버를 사용하여 대용량 데이터를 높은 스루풋으로 처리할 수 있다.

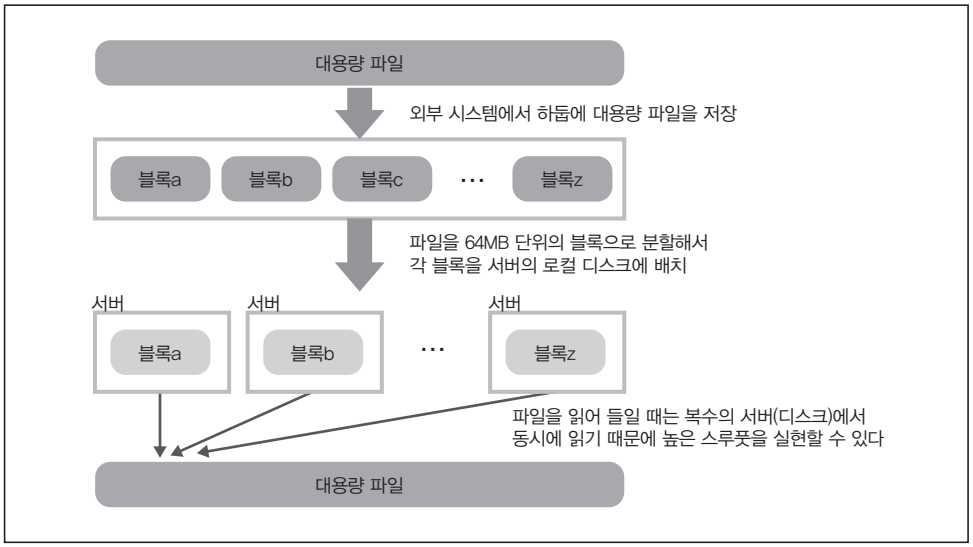


그림 1.4 HDFS는 대용량 파일을 블록에 분할해서 저장

HDFS는 높은 스루풋으로 데이터를 읽어 들일 수 있지만, 읽어 들인 데이터를 높은 스루풋으로 처리하기 위해서는 고민할 필요가 있다. 예를 들어 기존 데이터베이스 시스템같이 저장 장치와 서버를 네트워크로 연결하면, 해당 통신 회선이 가진 성능 이상으로 높은 스루풋을 구현하지 못한다(그림 1.5). 고속 채널도 있지만 그 역시 충분한 전송 속도라고는 할 수 없다.

따라서 읽어 들인 서버에서 데이터를 네트워크 전송하지 않고 처리하는 것이 관건이다. Hadoop에서는 각 서버에서 읽은 데이터를 가능한 한 그 서버 내에서 처리한다(그림 1.6).

16개 블록 데이터를, 16개의 개별 서버를 사용해서 처리하는 것이다. 물론 각 서버에서 처리한 결과를 네트워크를 통해 전송하고 하나의 결과로 합칠 필요가 있다[9].

9 일반적으로, 읽어 들이는 데이터 양보다 네트워크 전송 용량이 작다고 가정한다.

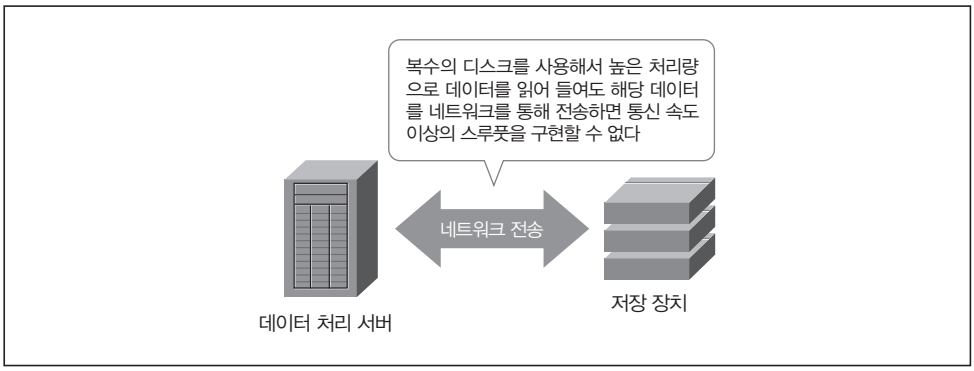


그림 1.5 저장 장치와 서버 간 데이터 전송

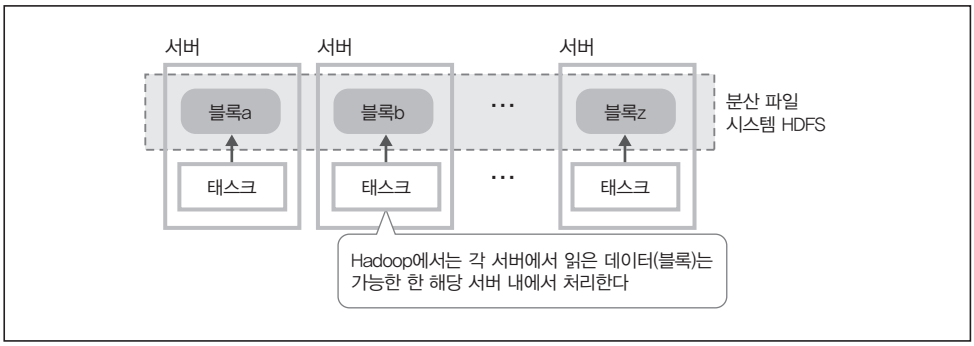


그림 1.6 Hadoop은 데이터 지역성(Locality)을 고려한 구조를 가졌다

Hadoop은 다음의 기능을 미들웨어로서 제공한다.

- 복수의 하드디스크나 서버를 동시에 이용할 수 있는 구조(높은 처리량으로 데이터를 읽어 들여 병렬 처리한다.)
- 분산 처리에 반드시 필요한 공통 기능 제공(태스크 분할, 실패 시 복원 등)

1.3 Hadoop 적용 분야

1.3.1 만능 제품이 아니다

'대용량 데이터를 처리하기 위해 병렬 분산 처리의 복잡함을 해소하고 확장성을 실현했다.'는 사실만 강조한다면, 기존 관계형 데이터베이스 관리 시스템(RDBMS)을 사용해서 데이터 관리를 한 사람에게는 마법같이 들릴 수도 있다. 하지만 Hadoop은 대용량 데이터에 특화된 미들웨어지 만능 제품은 아니다. 물론 Hadoop은 RDBMS도 아니기 때문에 오히려 대용량 데이터 처리를 위해 희생하고 있는 기능도 많다. 결론부터 말하면 Hadoop 적용 분야는 '테라바이트 또는 페타바이트 급의 데이터를 위한 일괄(Batch) 처리'다.

이 장에서는 기존 RDBMS와 비교해 가면서 Hadoop 적용 분야에 대해 설명하겠다. 그리고 RDBMS와 Hadoop을 연계한 구성에 대해 언급하고, 기존 시스템을 Hadoop으로 변경함으로써 비용 절감 효과를 얻을 수 있는지에 대해 설명한다.

1.3.2 RDBMS와 Hadoop 비교

Hadoop 특성을 설명할 때, 기존 RDBMS와 비교하면 이해하기 쉽다. 여기서는 표 1.2의 다섯 가지 관점에서 비교하겠다.

표 1.2 RDBMS와 Hadoop

	항목	RDBMS	Hadoop
1	다루는 데이터 크기	~ 수 테라바이트	테라바이트 ~ 페타바이트
2	데이터 조작	작은 데이터 참조, 변경	큰 데이터 삽입, 참조(변경은 없음)
3	응답 시간	빠름	느림
4	서버 대수와 성능 향상	여러 대의 서버로 스케일 업	수백 대~수천 대 서버로 스케일 아웃
5	데이터 구조	구조화 데이터	준 구조화 데이터

■ 1. 다루는 데이터 크기

RDBMS에서는 요건에 따라 다르긴 하지만, 하나의 인스턴스(instance)가 취급하는 데이터 크기는 수 기가바이트(GigaByte, GB)에서 수 테라바이트 정도다. OLTP 계열 백엔드(back-end)에서 사용하는 데이터베이스 서버의 경우 1TB 영역 전체를 갱신하는 방식은 드물며, 규모가 매우 큰 시스템이라 할 수 있다.

한편, Hadoop이 다루는 데이터 크기는 테라바이트~페타바이트 규모다. 물론 기가바이트 급 시스템도 Hadoop을 사용할 수 있지만, 그 정도는 기존 RDBMS로도 처리가 가능하다. 반대로 말하자면 RDBMS가 다루기 어려운 수백 기가바이트 이상 급이 아니라면, Hadoop을 굳이 사용할 이유가 없다.

■ 2. 데이터 조작

기존 RDBMS에 요구되는 것은 작은 데이터의 참조 및 갱신 처리가 메인이다. 하나의 트랜잭션으로 한 개 레코드를 참조하거나 갱신하는 작업이 많다. 데이터베이스의 디스크 I/O 단위인 블록 사이즈도 8KB, 16KB(킬로바이트, KiloByte) 정도 크기다. 이런 처리가 1초간 몇 번이고 실행된다.

한편, Hadoop에서는 대량의 데이터를 모아서 한 번에 처리한다. 1회 접속으로 수십 메가바이트에서, 크면 수백 테라바이트까지 처리할 수 있도록 설계되어 있다. 대량의 데이터를 한 번만 저장해 두고, 그 데이터를 참고하여 집계 처리 등을 한다. 한 번 저장된 데이터에 대한 변경 처리 개념은 존재하지 않는다. 오히려 한 번만 기록할 수 있다는 제약이, 기존 시스템에서는 어려웠던 높은 스루풋을 구현하도록 만든다[10].

■ 3. 응답 시간(Latency)

Hadoop과 RDBMS는 어떤 처리를 요구해서 그 결과를 얻기까지 걸리는 응답 시간의 특성이 다르다. 기존 RDBMS는 기본적으로 짧은 트랜잭션을 다루는 경우가 많기 때문에, 수 밀리초~ 수 초 정도의 짧은 응답 시간을 요구한다. 인덱스(Index)를 정의함으로써 필요한 데이터에만 접근하여 빠른 응답 시간을 구현하고 있다. 또한, 처리 요구에 빠르게 대응하기 위해 다양한 튜닝 작업도 가능하다(그림 1.7).

한편, Hadoop의 응답 시간이 느려서 온라인 처리에 적합하다고 말하기는 어렵다.

10 현재 Hadoop에는 추가 기록 기능이 있지만, 특정한 경우에만 사용할 수 있는 것으로, 범용성은 가지고 있지 않다.

Hadoop의 경우 처리를 분산하기 위한 전처리가 필요하기 때문에 하나의 잡(Job)을 실행시키기만 해도 최저 10~20초 정도의 오버헤드(overhead)가 발생한다. 이것에 실제 데이터 처리를 위한 시간을 더하면, 금세 수 분 정도의 처리 시간에 도달하게 된다. 때문에 RDBMS는 온라인 처리에 적합하고, Hadoop은 일괄 처리에 적합하다는 것을 이해할 필요가 있다.

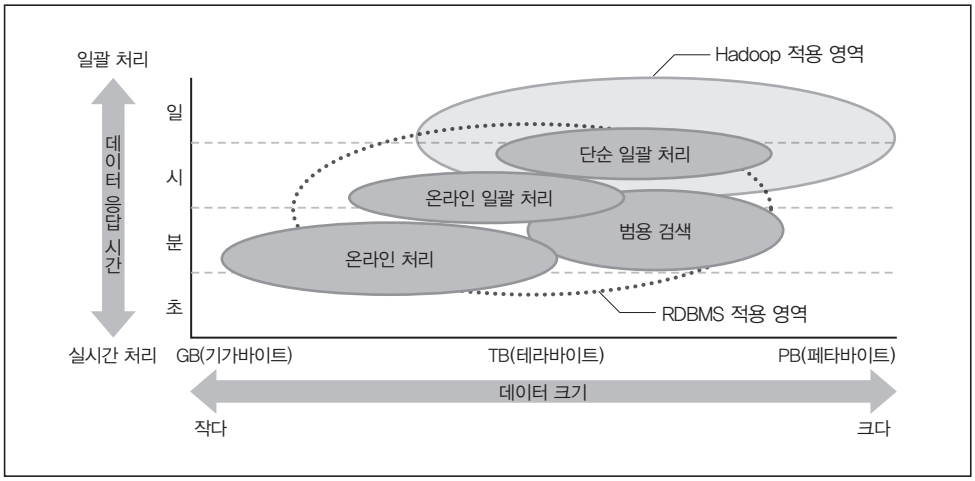


그림 1.7 RDBMS와 Hadoop 포지셔닝(Hadoop World NYC 2011, NTT 데이터 발표 자료 http://www.slideshare.net/cloudera/hadoop-world-2011-hadoops-life-in-enterprise-systems-ymasatani-ntt-data)

■ 4. 서버 대수와 성능 향상

기존의 RDBMS는 갱신 처리를 오류 없이 수행하기 위해 데이터 정합성이 매우 중요하다. 이를 위해 락(Lock)을 사용하여 변경 대상 데이터를 보호할 필요가 있다. 어떤 사용자가 특정 데이터에 락을 걸면, 해당 데이터에 접근할 수 있는 사람은 기본적으로 한 명밖에 없다. 동시 실행은 당연히 어렵다. RDBMS는 이런 개념을 바탕으로 하여 설계되었으므로 기본적으로 병렬 처리에 적합하지 않다.

RDBMS에서는 성능을 향상시키기 위해 어떻게 하나의 처리를 단시간에 끝낼 수 있을까에 주력한다. 때문에 성능 향상을 위한 기법은 '스케일 업', 즉 '서버 자체 성능을 향상'시키는 것이다. 다시 말하자면, 처리 능력이 높은 하드웨어를 사용해서 성능 향상을 꾀할 수밖에 없다는 것이다(그림 1.8). 하지만 스케일 업을 통한 성능 향상은 많은 비용이 필요하다.

스케일 업으로 처리 능력을 두 배 향상시키기 위해 필요한 비용은 통상적으로 원래 하드웨어 비용의 두 배 이상이 든다[11].

기존 RDBMS 가동 서버는 일반적으로 한 대다. 앞에서 말한 것과 같이, 다수의 서버를 사용해서 RDBMS를 구현하기 위해서는 상위 레벨에서 관리를 위한 다양한 처리를 설정해야 한다. 하지만 이런 작업은 기본적으로 그리 녹록지가 않다. 일부 상용 RDBMS이 복수의 서버를 사용한 클러스터 구성을 구현하고 있긴 하지만, 서버 간 동시 처리를 필요로 하기에 장비 추가에 따른 성능 향상에도 한계가 있다.

Hadoop은 확장성이 쉽다는 것을 대전제로 하고 있다. Hadoop 처리 모델인 MapReduce는 여러 대의 서버를 사용해 병렬 실행할 수 있다는 것을 가장 중시하고 있다. 당연히 성능 향상을 위한 기법은 '스케일 아웃', 즉 서버를 많이 추가하는 방식이다.

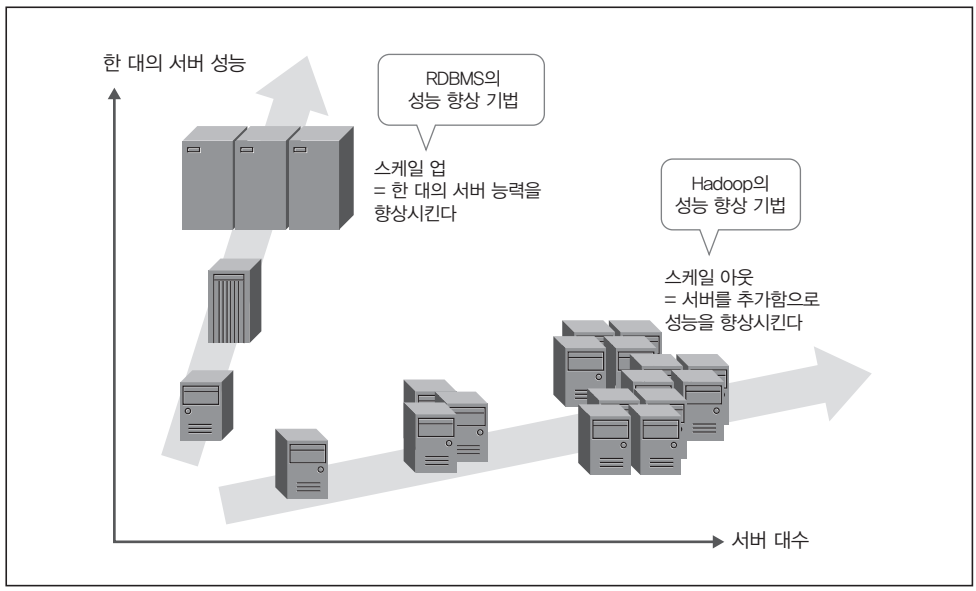


그림 1.8 스케일 업과 스케일 아웃

■ 5. 데이터 구조(구조화 데이터/준 구조화 데이터)

확장성 이외의 관점에서도 차이가 있다. RDBMS로 데이터 처리를 할 때는 데이터가 사전

11 10배 정도의 비용이 드는 경우도 드물지 않게 있다.

에 정의된 구조로 정규화될 필요가 있다. CREATE TABLE문을 실행하여 데이터를 저장하기 전에 데이터 저장 영역의 테이블 구조(스키마)를 결정해 두어야 한다. 이로 인해 설계에 비용과 시간이 들게 되고, 칼럼 추가나 삭제로 인해 구조가 변경되면, 어마어마한 비용이 또 들어가게 된다.

한편, Hadoop은 사전에 데이터 구조를 정의할 필요가 없다. 어떤 값을 키(Key)로 정할지 등 데이터에 의미를 붙이는 작업은 처리 실행 시점에 필요에 따라 해주면 된다. 데이터마다 구조가 다른 것을 허용하기 때문에 구조화가 덜 된 데이터라도 저장, 처리할 수 있다. 칼럼 추가나 삭제를 위해서 특별한 작업이 필요 없는 것이다.

이상, 여기까지의 설명을 통해 Hadoop의 특징을 다음과 같이 요약, 정리할 수 있다.

- 수백 기가바이트~수 테라바이트 이르는 대용량 데이터의 일괄 처리에 적합하다
- 확장성이 우수하다(서버 대수를 늘리면 저장할 수 있는 용량과 처리 성능이 향상된다.)
- 유연한 데이터 구조에 대응(처리 시점에 데이터 정의가 가능)

실제 시스템 구성에서는 기존 RDBMS와 Hadoop을 연동시키는 경우도 많다. 예를 들어, Hadoop으로 대용량 데이터를 기반으로 한 집계 결과를 출력한다고 하자. 해당 집계 결과를 검색하기 쉽도록 RDBMS에 로드한다. 원래 작은 데이터와 온라인 처리에 강하므로 Hadoop이 계산한 집계 결과를 보여주기 위해서 RDBMS를 사용하는 경우가 있다. RDBMS 이외에도 데이터 웨어하우스를 구현하는 제품이나 검색 엔진 소프트웨어 상에 Hadoop 실행 결과를 로드하는 경우도 있다.

1.4 Hadoop 시스템 구성과 아키텍처

이 절에서는 Hadoop 시스템 구성과 아키텍처의 개요를 설명한다. 상세 처리 흐름에 대해서는 뒤에서 따로 설명하고 있기 때문에 이 절에서는 대략적인 흐름을 잡을 수 있도록 안내하겠다.

1.4.1 Apache의 Hadoop 프로젝트

Hadoop에는 복수의 프로젝트가 존재한다[12]. Hadoop 커먼(Common)은 Hadoop의 각종 프로젝트를 지원하는 공통 유틸리티의 집합이다. 주요 프로젝트는 분산 파일 시스템 'HDFS'와 병렬 분산 처리 프레임워크 'MapReduce'다. 또한, 표 1.3에서 보여주고 있듯이 Hadoop을 사용하기 쉽도록 하는 각종 주변 제품 개발이 진행 중이다[13].

1.4.2 서버 구성

Hadoop 전체 구조를 이해하기 위해서 우선은 물리적 서버 구성에 대해 설명하겠다. Hadoop을 구성하는 서버는 클러스터 전체를 관리하는 '마스터(Master) 서버군'과 실제로 데이터를 저장하고 처리하는 '슬레이브(Slave) 서버군' 두 종류로 나뉜다. 이들 서버군이 협력하여 동작함으로써 하나의 Hadoop 클러스터를 구성한다.

표 1.3 Hadoop 주요 프로젝트

프로젝트명	설명
HDFS	높은 스루풋을 유지하면서 동시에 신뢰성을 추구하는 분산 파일 시스템
MapReduce	클러스터 환경에서 병렬 분산 처리를 수행하기 위한 프레임워크
Hive	Hadoop 조작을 쉽게 하기 위한 SQL 형식의 인터페이스
Pig	Hadoop 데이터 흐름을 기술하기 위한 스크립트 언어 방식 인터페이스
Sqoop	DBMS와 Hadoop 사이의 데이터 이동을 구현하는 커넥터(connector)
HBase	대용량 테이블을 관리하기 위해서 확장성을 추구한 키-밸류(Key-value) 방식 저장소
ZooKeeper	분산 클러스터 환경에서 동작하는 애플리케이션을 관리하기 위한 제품

현재 Hadoop은 크게 두 가지 계통이 존재한다. 버전 1.0대로 불리는 기존 아키텍처가 있고, 이와 병행하여 버전 2.0대의 새로운 아키텍처 개발이 진행되고 있다. 버전 2.0도 조금씩 활용되고 있긴 하지만, 아직 기존의 1.0대가 주류를 이루고 있다. 이 책에서는 1.0대 아

12 새로운 프로젝트가 계속 늘고 있는 추세다. http://hadoop.apache.org에서 프로젝트 리스트를 확인할 수 있다.

13 Hadoop 세계에서는 주변 소프트웨어군을 Hadoop 에코 시스템이라 부르기도 한다.

키텍처를 바탕으로 설명하겠다(그림 1.9)

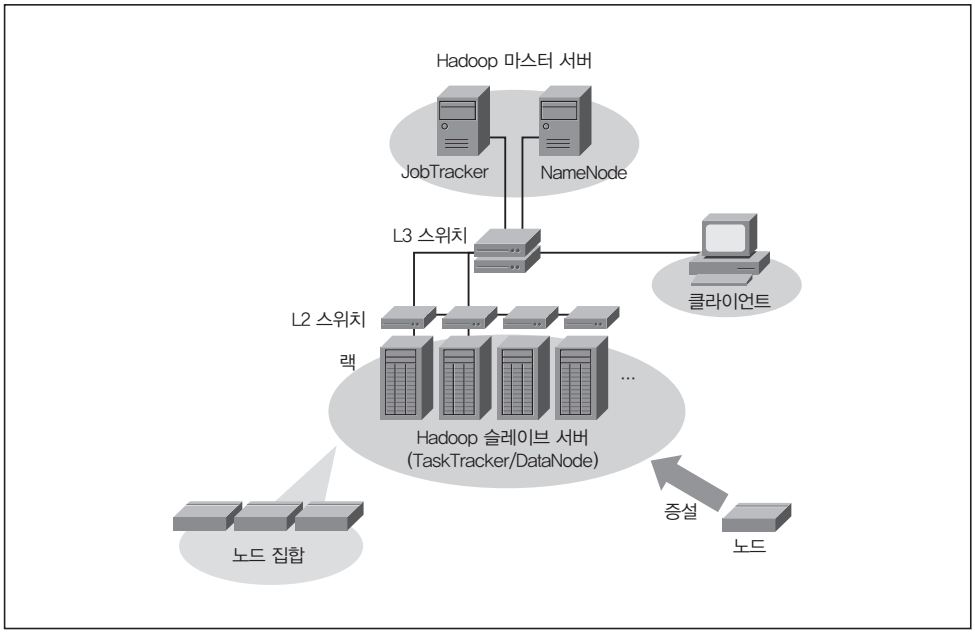


그림 1.9 Hadoop 서버 구성

Hadoop 분산 처리를 통한 '높은 스루풋'을 구현하기 위해서는 슬레이브 서버를 10대에서 수천 대 규모로 구성할 필요가 있다. 여러 대의 슬레이브 서버로도 Hadoop 클러스터를 구성할 수 있지만, 분산 처리에 따른 오버헤드가 분산 처리로 인한 성능 향상 효과보다 커지게 된다[14].

슬레이브 서버는 1U나 2U의 IA 서버로 구성되며, 랙(Rack)에 설치되는 것이 일반적이다. 서버 수가 20~30대를 넘으면 하나의 랙에 설치할 수 없기 때문에 다수의 랙에 나누어서 설치해야 한다. 서버 간 네트워크 접속에는 특별한 하드웨어가 필요치 않으며 랙 내부에서는 L2 스위치를, 랙 간 접속에는 L3 스위치를 배치하는 것이 일반적이다. 2013년 현재는 1Gbps NIC나 스위치가 주류를 이루고 있지만, 이후 가격이 떨어지면 10Gbps NIC와 스위치 사용이 일반적이 될 것이다.

14 여러 대의 클러스터로 검증은 가능하지만, 성능 향상이 목적인 경우는 추천하지 않는다.

한편, 마스터 서버는 슬레이브 서버보다 고성능, 고신뢰 장비를 사용하는 것이 일반적이다. 예를 들어, 2U의 IA 서버로 충분한 메모리양을 탑재하고 있는 장비를 이용한다. 슬레이브 서버는 RAID가 필요하진 않지만 SATA 디스크가 일반적이다. 반면, 마스터 서버는 SAS 디스크를 사용한 RAID 구성을 이용하기도 한다. 앞서 언급한 HDFS와 MapReduce 각각에 '마스터 서버'와 '슬레이브 서버'가 존재한다.

HDFS의 마스터 서버를 'NameNode'라고 부른다. 클러스터 전체에 걸쳐서 '데이터가 어디에 배치되어 있는지' 등의 메타(Meta)데이터를 관리한다. 한편, 슬레이브 서버를 'DataNode'라고 부른다. 실제 데이터를 읽고 쓰는 역할을 한다.

MapReduce의 마스터 서버를 'JobTracker'라고 부른다. 하나의 잡(job)을 태스크(Task)라 불리는 복수의 처리로 분할하여 각 슬레이브 서버에 할당한다. MapReduce의 슬레이브 서버를 'TaskTracker'라고 부른다. 할당된 태스크를 실행하고 결과를 반환한다.

각각의 마스터 서버인 NameNode, JobTracker는 각 한 대씩이다[15]. 소규모 Hadoop 클러스터에서는 NameNode와 JobTracker가 하나의 장비 내에서 동작하지만, 대규모 클러스터에서는 별도 장비를 사용한다.

슬레이브 서버는 여러 대로 구성되지만, DataNode와 TaskTracker가 같은 서버에 설치되는 것이 일반적이다. 각 DataNode가 읽어 들인 데이터를 같은 서버에 있는 TaskTracker로 처리할 수 있게 하기 위해서다. 이들 서버 외에도 데이터를 삽입하거나, 잡을 실행하기 위한 Hadoop 클라이언트 등도 있다.

1.4.3 HDFS: Hadoop 분산 파일 시스템

HDFS는 대용량 파일에 높은 처리량으로 접근할 수 있도록 설계된 분산 파일 시스템이다.

일반적인 파일 시스템으로 말하자면 리눅스의 ext4나 XFS 등을 들 수 있지만, 이들은 한 대의 서버 상에서 파일을 관리하기 위한 것이다. 분산 파일 시스템은 복수의 서버에서 규모가 큰 하나의 파일 시스템을 제공한다.

HDFS는 각 슬레이브 서버(DataNode)에 있는 ext4 같은 로컬 파일 시스템 상에 만들어지

15 HA 등 이중화 구성의 경우는 다수의 서버를 사용한다.

는 '오버레이 파일 시스템'이다. HDFS 상에 배치된 파일은 64MB 단위의 블록으로 분할되어 각 장비에 저장되지만, 각 블록 데이터는 ext4 같은 로컬 파일 시스템의 파일로 취급되는 것이다. HDFS에 대한 상세한 사항은 4장에서 설명하고, 여기서는 HDFS의 기본적인 특징만 소개하겠다.

■ HDFS의 편리성

HDFS의 편리성은 '투과성'과 '확장성' 두 가지 관점에서 설명할 수 있다.

- 투과성
  클라이언트 관점에서는 파일 시스템의 내부에서 복수의 서버가 동작하고 있다는 것을 고려할 필요 없이, ext4 같은 로컬 파일 시스템을 다루듯이 투과적으로 접근할 수 있다. 사용자는 파일이 어떻게 블록으로 분할되어 있는지 의식할 필요가 없다.

- 확장성
  HDFS는 슬레이브 서버 DataNode 대수를 늘려서, 용량과 기본적인 I/O 성능(스루풋)을 향상 시킬 수 있다. 디스크 용량이 부족하면 서버를 추가하기만 하면 된다[16]. 확장성이 보장된다는 점이 HDFS의 매력이라고 할 수 있다.

■ HDFS의 신뢰성

HDFS는 파일을 복수의 서버에 분할 배치 하지만 이것이 전부가 아니다. 하나의 파일을 복수의 '블록'으로 분할하고, 각각의 블록을 복수의 서버에 다중으로 기록하는 리플리케이션(replication) 기능을 가지고 있다(그림 1.10). 기본 설정에서는 각 블록이 세 개의 서버에 다중으로 저장된다[17].

하나의 서버가 고장 나서 블록에 접근할 수 없게 되더라도 다른 서버에 동일한 블록이 존재하기 때문에 분산 파일 시스템 전체적으로 데이터 소실 위험이 낮다.

마스터 서버인 NameNode가 고장 나면 HDFS 전체가 망가져 버린다(그림 1.11). HDFS의 가용성을 높이기 위해 NameNode의 HA 구성(이중화)을 할 수 있다[18].

16 기존 데이터를 서버 사이에 재배치해야 할 경우도 있다.

17 다중도(복제 수)는 설정에서 변경할 수 있다.

18 이전 버전에는 이중화 기능이 Hadoop에 없어서 DRBD나 Pacemaker 등을 조합하여 사용했다.

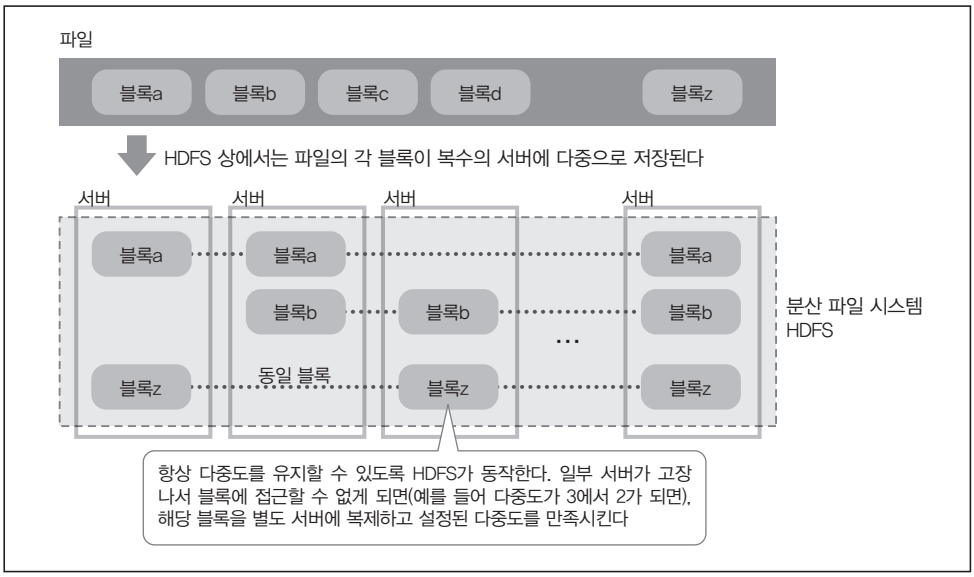


그림 1.10 HDFS에서는 각 블록을 복수의 서버에 다중으로 저장

■ 접근 패턴 제한

높은 스루풋을 실현하기 위해 HDFS에서는 데이터 접근 패턴을 제한하고 있다. 예를 들면, 다음과 같은 제약들이 있다.

- 기본적으로 연속적 스트림(stream) 읽기를 전제로 하고 있다. 랜덤 읽기 방식은 고려하지 않는다
- HDFS에서 데이터 기록은 한 번만 이루어지고, 그 이후는 읽기 처리만 가능하다. 데이터 변경은 불가능하다

일반적인 파일 시스템 상에서 데이터를 다루는 것과는 많은 차이가 있다. HDFS를 사용할 때는 이러한 제약 사항을 충분히 고려해야 한다.

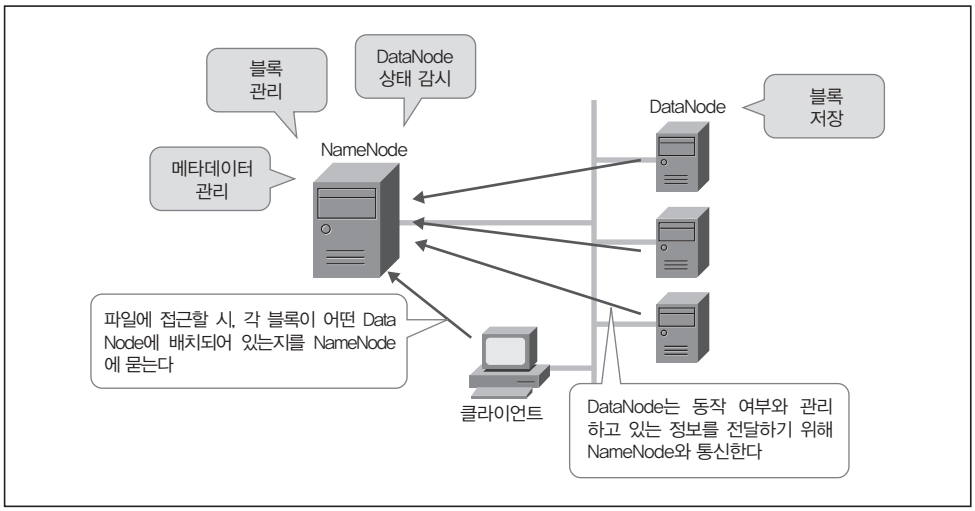


그림 1.11 NameNode는 HDFS 전체를 관리하기 때문에 만약 고장 나면 HDFS 전체가 망가져 버린다

1.4.4 Hadoop MapReduce 프레임워크

MapReduce는 대규모 데이터 집합을 처리하기 위한 프로그래밍 모델이다. 처리를 병렬로 실행하기 위해 하나의 잡을 독립된 태스크로 나누어서 실행한다.

■ 여러 사람이 전표 처리를 하는 상황을 예로 들어본다

MapReduce 처리가 어떤 구조인지 설명하기 전에 여러 사람이 종이 전표를 집계하고 있는 상황을 생각해 보도록 하자.

다음과 같은 상황을 가정해 보자.

- A, B, C 세 사람이 분담해서 집계 작업을 한다
- 취급 상품은 '상품a', '상품b', '상품c', '상품d', '상품e' 다섯 종류가 있다
- 전표에는 상품 판매 실적이 기록된다. 하나의 전표에는 전표 번호, 상품명, 수량이 기재되어 있다. 하나의 전표에는 한 종류의 상품만 기록된다
- 각 상품이 몇 개씩 판매되었는지 집계하고 싶다

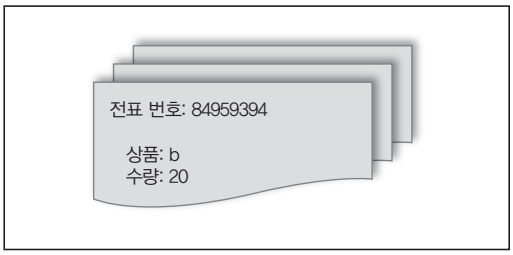


그림 1.12 전표 예시

이때 세 명이서 효율적으로 작업을 진행하기 위해 세 단계로 나누어 작업하는 방법이 있다(그림 1.13).

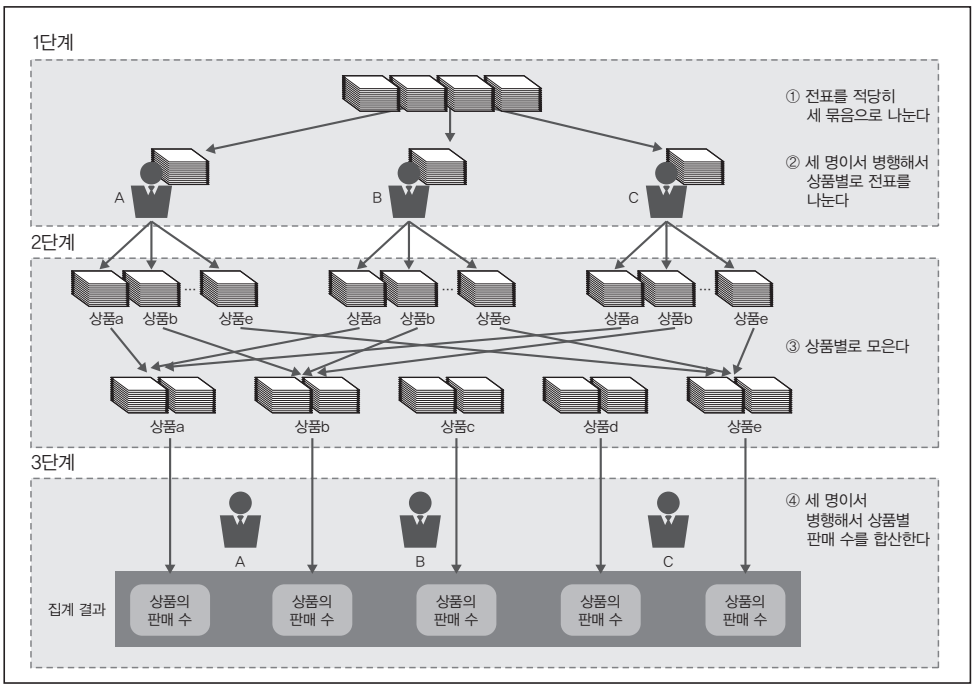


그림 1.13 다섯 상품의 판매 수를 세 명이서 집계하는 작업 예

우선 쌓여 있는 전표 뭉치를 적당히 세 그룹으로 나누어 세 명에게 나누어 준다①. 세 명은 전표를 한 장씩 보고, 상품 종류별로 분류한다. 예를 들어, A는 '상품a 전표', '상품b 전표'… '상품e 전표' 등 다섯 개 전표 그룹을 만들어 쌓아간다. 동일하게 B도, C도 각각 다섯 개 전표 그룹으로 분류한다②. 여기서 전표가 쌓인 그룹은 전체 열다섯 개가 된다.

다음으로 세 명이 만든 그룹을 상품별로 다시 정리한다고 하자. 'A가 분류한 상품a 전표', 'B가 분류한 상품a 전표', 'C가 분류한 상품a 전표'의 세 개 그룹을 하나의 그룹으로 정리한다. 동일하게 다른 상품들도 정리하면 다섯 개의 전표 그룹이 쌓이게 된다③.

마지막으로, 이 다섯 개 전표 그룹을 세 명이서 집계한다. 예를 들어, 상품a와 b는 A가, 상품c는 B가, 상품d와 e는 C가 담당한다. A는 우선 상품a의 전표를 순서대로 확인해서 몇 개씩 팔렸는지 합산한다. 그것이 끝나면 상품b 전표의 판매 수를 합산한다. 동일한 방법으로 B와 C도 작업을 진행한다④.

이를 통해 각 상품이 얼마나 판매됐는지를 세 명이 분담해서 집계할 수 있다. 실은 지금부터 설명하는 MapReduce는 이 3단계 작업을 범용화했다고 보면 된다.

몇 가지 주목해야 할 점이 있다. 우선은 각 단계에 있어서 A, B, C 작업은 독립된 작업이라는 것이다.

다른 한 가지는 사람 수가 바뀌어도 동일하게 분담할 수 있다는 것이다. 전표의 양이 많아서, 세 명이서 감당하기 어려우면 다섯 명이서 동일한 작업을 수행하면 된다.

첫 단계는 전표를 다섯 묶음으로 나누어 다섯 명이 분담하면 되고, 마지막의 합산 작업도 한 사람이 한 상품을 담당하면 된다[19].

MapReduce에서는 하나의 잡을 복수 태스크로 분배하고, 복수의 TaskTracker를 사용해 병렬로 실행하지만 결국 앞선 작업 방식과 같다.

■ MapReduce 처리 흐름

MapReduce는 크게 'Map(맵) 처리'와 'Reduce(리듀스) 처리'라 불리는 두 단계로 구성된다. 애플리케이션 개발자는 Map 처리와 Reduce 처리로 각각 어떤 동작을 할 것인지 정의한다. 반대로 말하면, Map 처리와 Reduce 처리 내용을 정의해 두면 이후부터는 MapReduce 프레임워크가 자동으로 병렬 분산 처리를 실행해 준다.

Map 처리는 주로 입력 파일을 한 줄씩 읽어서 필터링(filtering) 등의 처리를 하고, Reduce 처리는 데이터 집약을 맡는다. 앞의 예에서는 Map 처리는 '한 건의 전표에서 상품과 수량을 읽어 내고', Reduce 처리는 '상품별로 집약된 전표를 확인하고 수량을 합산하

19 여섯 명 이상이면 마지막 작업은 다섯 명이 하게 된다. 조금 고민해 보면 마지막 작업도 여섯 명 이상이 할 수 있다.

는' 것과 같다.

Map 처리와 Reduce 처리 사이에는 'Shuffle'이라 불리는 처리가 자동적으로 실행된다. 이것은 Reduce의 전처리에 해당하는 것으로, 사용자가 처리를 지정해 줄 필요는 없다.

어떤 처리든 데이터는 키(key)와 밸류(value)의 쌍으로 이루어지고, 해당 쌍의 집합을 처리한다. 입력 데이터도 출력 데이터도 이 키-밸류의 집합으로 구성된다. 예를 들어, 앞 예에서는 전표 번호라는 키와, 상품명과, 수량이라는 밸류를 가진 전표가 한 건의 데이터가 되는 것이다.

그러면 처리 흐름을 Map과 Reduce 순으로 살펴보도록 하자.

Map 처리

Map 처리에서는 입력 데이터 집합을 분할해서 각각을 Map 태스크에 할당한다①. 입력 데이터는 HDFS 상의 파일이며 데이터 분할은 MapReduce 프레임워크가 자동으로 수행한다[20].

MapReduce 프레임워크는 각 Map 태스크를 TaskTracker에 할당한다. 각 Map 태스크는 입력 데이터로부터 한 건씩 키-밸류 쌍을 꺼내서 사용자가 정의한 Map 처리를 수행하고, 처리 결과도 키-밸류 형태로 출력한다②. 앞 예에서는 전표 번호를 키, 상품과 수량을 밸류로 하는 전표 한 장씩을 입력하고, 키를 상품, 밸류를 수량으로 하는 데이터를 출력하고 있다고 보면 된다.

Shuffle 처리

Map 처리가 완료되면 MapReduce 프레임워크가 Map 처리 후 데이터를 정렬(sort)해서, 같은 키를 가진 데이터를 같은 장소에 모은다③. 이때 슬레이브 서버 간에 네트워크를 통한 전송이 발생한다.

이 Shuffle 처리는 MapReduce 프레임워크가 자동적으로 수행하기 때문에 사용자가 별도로 처리를 기술해 줄 필요가 없다. 단, Shuffle 처리에서 네트워크를 통한 데이터 전송이 발생한다는 것은 충분히 인식하고 있어야 한다. 전송 데이터양이 크면, Shuffle 처리가 처리 전체에 지장을 줄 수 있다. 따라서 Map 처리 결과의 데이터양도 고려해야 한다.

20 원칙적으로 HDFS 상의 블록 단위로 분할된다.

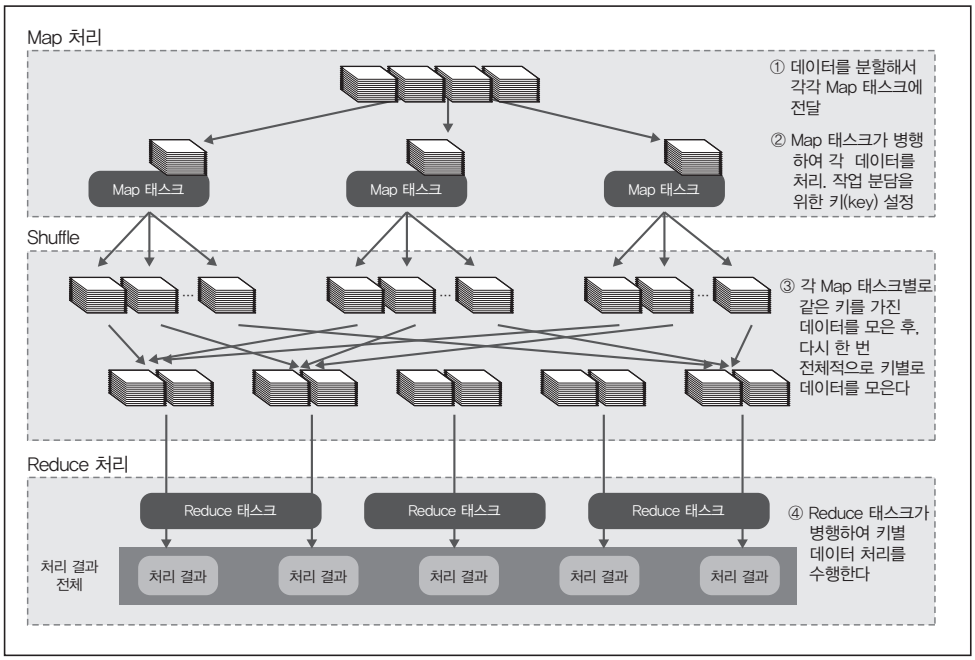


그림 1.14 MapReduce 처리 흐름 개요

Reduce 처리

Shuffle을 통해 키별로 모아진 데이터에 대해 Reduce 처리를 한다④. 키 수만큼 처리가 이루어지게 되는 것이다. 복수의 TaskTracker를 사용해서 병렬로 Reduce 처리를 할 수 있다. 앞선 예에서는 Shuffle로 동일 상품별로 데이터를 모으고, Reduce 처리로 밸류인 수량을 합산해서 상품별로 판매 수를 계산한다.

Map 처리도 Reduce 처리도 복수의 TaskTracker가 협력해서 동작한다. 이때, 예를 들어 어떤 데이터를 어떤 서버에 할당해서 처리할 것인지 고민할 필요가 없다. MapReduce 프레임워크가 자동으로 데이터를 분할해서 각 서버에 처리를 분배해 준다. 또한 어떤 서버에 처리를 할당했는데, 해당 서버가 고장이 나버린 경우의 처리에 대해서도 자동적으로 처리해 준다. 사용자가 해야 할 것은 Map 처리 내용과 Reduce 처리 내용을 정의하기만 하면 된다. MapReduce 처리의 상세 내용이나 프레임워크의 사용법에 대해서는 5장에서 설명하겠다.

데이터 변경 관점에서 보는 MapReduce

MapReduce의 각 단계별 처리에 있어서 데이터 이동 및 변경 형태를 일반적으로 기술하면 다음과 같다.

Map 처리: 입력 〈k, v〉* → 출력 〈k', v'〉*

입력 데이터는 키-밸류 조합인 〈k, v〉다. k는 키(key)를, v는 밸류(value)를 나타낸다[21]. 앞 예에서는 k는 전표 번호, v는 상품과 수량의 조합이 된다. 예를 들면 〈84959394, [상품b, 20]〉과 같다.

Map 처리에서는 이 입력 데이터에 대해 새로운 키-밸류 조합인 〈k', v'〉를 출력한다. 이 처리가 k별로 순차적으로 이루어진다. 이때 k'는 다음 Shuffle에서 데이터를 집약하는 단위가 된다. 예에서는 상품별로 데이터를 모으기 위해 k'를 상품, v'를 수량으로 취급한다. 예를 들어, 〈상품b, 20〉이 된다.

Shuffle: 〈k', v'〉** → 〈k', v'*〉*

Map 처리의 출력 데이터를 k'별로 모으고 밸류를 집약한다. 각 데이터의 밸류는 배열로 구성되어서 한 건의 데이터가 된다. 이때, v'의 배열을 정렬(sort)한다[22]. 입력이 〈상품b, 20〉, 〈상품b, 3〉, 〈상품b, 4〉일 때, 출력은 〈상품b, [3, 4, 30]〉이 된다. 이 처리가, 각 k'별로 이루어진다.

Reduce 처리: 입력〈k', v'*〉* → 〈k", v"〉*

Shuffle에서 k'별로 밸류가 집약된 데이터를 입력 데이터로, 새로운 키-밸류 조합 〈k", v"〉을 출력한다. 예에서는 〈상품b, [3, 4, 20]〉을 입력 데이터로 수량을 합산해서 〈상품b, 27〉을 출력한다. 이 예에서는 입력 키 k'와 출력 키 k"가 상품b로 동일하지만 달라도 상관없다. 또한, 이 처리가 각 k'에 대해 병렬로 실행된다.

1.5 Hadoop 적용 사례

이번 절에서는 Hadoop을 적용함에 따라 비즈니스 상 어떤 이점을 얻을 수 있는지에 대해 설명한다. 여기서는 우선 PoS 데이터를 사용한 고객 동향 분석 시스템을 예로 들어, Hadoop을 적용하면 무엇이 가능해지는지를 설명한다. 또한, 실제 Hadoop이 적용되고

21 *는 이들 데이터가 여러 개로 연속된다는 것을 의미한다.

22 이 단계 처리를 Shuffle&Sort라고 부르기도 한다.

있는 선행 사례들에 대해서도 몇 가지 소개하겠다.

1.5.1 Hadoop을 통해 변하는 것

Hadoop을 도입함으로 새로운 가치를 창출할 수 있다고 설명해 왔었다. 그럼 여기서 구체적인 예를 하나 들어보겠다.

어떤 사업 형태든 '매출을 늘리고 싶다.'는 생각은 공통적으로 하기 마련이다. 소매업계를 예로 들면, 매출을 늘리기 위해서 고객 동향을 충분히 파악한 후, 상품을 늘리거나 상품 배치를 바꾸고, 재고 수를 확보하기도 한다. 또한, 점포 수를 늘리거나 줄이기도 한다. 이런 대책을 실행하기에 앞서 고객 동향을 파악하게 되는데, 이때 정보원이 되는 것이 바로 PoS 데이터[23]다.

소매업계에서는 슈퍼마켓, 편의점, 약국 등을 시작으로 PoS 데이터를 수집해서 사용해 오고 있다. PoS 데이터를 해석하는 것으로 재고 관리나 판매 관리를 하고, 판매 경향을 발견하기도 한다. 여기서는 'PoS 데이터 해석에 Hadoop을 도입'하는 예를 들어 도입하면 어떤 변화가 발생하는지 살펴보자.

■ 예1: 처리 주기가 짧아짐

PoS 데이터를 집약하는 PoS 시스템에는 대량의 데이터가 축적된다. 대량의 데이터를 정리하고 해석하기 위해 매일 일괄 처리를 실행해야 한다. 하지만 다루는 데이터양이 테라바이트 급이 되면 일괄 처리에 많은 시간이 걸려서 하루에 한 번의 실행도 어려울 때가 있다.

여기서 데이터양이 동일한 상태에서 계산 처리 능력을 늘리는 방법을 생각해 보자. Hadoop이라면 서버 추가를 통해 처리 능력 향상을 간단히 실현할 수 있다. 처리 능력이 올라가면, 일괄 처리 시간도 당연히 짧아진다. 일괄 처리에 걸리는 시간이 짧아진다는 것은 짧아진 주기로 일괄 처리를 반복할 수 있다는 것을 의미한다. 결과적으로 보다 짧은 주기로 피드백을 얻을 수 있게 된다.

23 PoS(Point of Sales: 판매 시점 정보 관리)란 점포 판매 데이터다. 어떤 상품이 언제, 어디서, 얼만큼 판매되었는지 나타낸다.

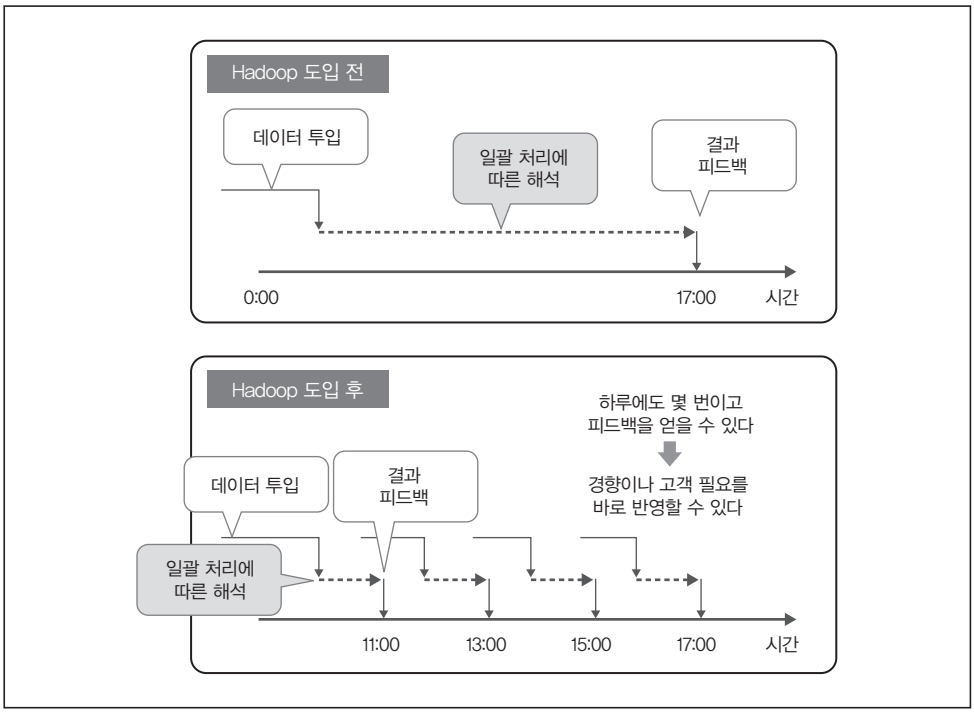


그림 1.15 일괄 처리 주기 향상

예를 들어, PoS 데이터의 일괄 처리로 잘 팔리는 상품과 부진한 상품을 발견하고 싶다면, 일괄 처리 주기를 15분에 1회 정도로 짧은 간격으로 실행함으로써 그날의 판매 경향을 거의 실시간으로 파악할 수 있다. 이 해석 결과를 바탕으로 각 점포가 재량껏 가격 조정 등의 대책을 세울 수 있게 된다.

■ 예2: 분석 범위 확대

앞서, 데이터양이 동일한 상태에서 처리 능력 향상을 통해 가능해지는 것들에 대해 설명했다. 이번에는 데이터양이 증가한 경우를 생각해 보자. PoS 데이터 경우, 종래는 한 달이 지나면 버렸던 데이터를 여러 해 동안 모아서 Hadoop을 사용해 처리할 수 있게 되었다. 이를 통해 1년간 '이 시기에는 판매가 어떤가?'라든가, '몇 년간 점점 ○○가 늘어나는 경향이 있다.' 등 어떠한 경향을 읽을 수 있게 된다. 그리고 이것을 상품 배치 전략에 활용하는 것이 가능해진다. 또한, PoS 데이터와 별도로 이벤트 정보를 연동함으로써 '○○사의 동향이 ××에 영향을 줄 가능성이 있다.'든가 '○○의 이벤트가 발생하면 ×× 경향이 있다.' 등을 파악할 수 있게 된다. Hadoop의 경우, 필요한 만큼 서버를 투입함으로써 종래에는 다

루기 어려웠던 데이터양을 처리할 수 있게 되어 보다 넓은 관점의 분석이 가능해진다.

1.5.2 Hadoop 애플리케이션 이용 사례

Hadoop은 이미 많은 기업에서 사용되고 있다. 주로 대량의 데이터를 실시간으로 처리하기 위해서다.

웹 접속 로그 통계

가장 많이 이용되는 분야 중 하나다. 사용자가 각 사이트에서 클릭한 이력을 모두 축적해 두었다가 사용자 접속 경향을 분석한다. 최근 사이트 접속량이 방대해지면서 접속 로그를 장시간에 걸쳐 보관해 둔다고는 하나, 처리 차체가 어려웠다. 이 때문에 지금까지는 로그를 그냥 버렸지만, Hadoop이 등장하면서 비로소 로그를 버리지 않고 활용할 수 있게 되었다.

전문 검색용 전치 인덱스 구축

전문 검색을 위한 인덱스 작성을 Hadoop을 통해 구현하는 예다. 어떤 단어가 어떤 문서에 존재하는가란 정보를 전치 인덱스라는 형태로 색인화한다. 웹 전체를 크롤(crawl)한 결과같이 입력하는 데이터양이 방대한 경우, 일반적인 인덱스로는 막대한 시간이 걸린다. 이때, Hadoop의 처리 성능 향상이 그 힘을 발휘한다. 검색 인덱스 엔진으로 Apache Solr[24]를 함께 이용하는 경우도 있다.

통계적 기계 번역

기계 번역이란 어떤 자연 언어를 다른 자연 언어로 변환하는 기술로, 예를 들면 영어를 한국어로 번역하는 것을 가리킨다. 번역 정확도를 높이기 위해 통계적 기법을 필요로 하지만, 이는 대량의 계산이 발생하기 때문에 Hadoop을 사용한 처리를 한다.

포맷 변환/이미지 변환

어떤 포맷의 데이터를 다른 포맷으로 변환하는 작업이 대량일 경우, 높은 처리 성능을 요구하는 경우가 있다. 이때, Hadoop을 사용할 수 있다. 예를 들어 대용량 동영상 데이터의 포맷을 변환할 경우, 데이터를 작은 크기로 나누어서 변환하고 변환 후에 데이터를 다시 결합시켜 한 개의 파일로 만드는 처리가 가능하다. 이미지 파일의 포맷 변환 예로서

24 http://lucene.apache.org/solr

뉴욕타임즈가 과거 130년간의 기사(1.5TB, 약 1,100만 장)를 PDF로 변환한 예가 있다. Amazon EC2 상에서 Hadoop 클러스터를 구축했고, 24시간 이내에 처리를 마칠 수 있었다고 한다.

부정값 및 이상값 검출

특정 패턴에 속하지 않는 이상값을 대량의 데이터에서 검출해내는 처리도 Hadoop으로 구현 가능하다. 미국 야후에서는 스팸 필터를 학습시키기 위해 Hadoop을 사용하고 있고, 비자(VISA)는 리스크 관리를 위한 스코링(scoring) 시스템 구성을 위해 사용하고 있다고 한다.

기계 학습

일정량의 데이터를 분석하고 그 데이터에서 의미있는 규칙이나 경향을 끄집어내는 처리다. 외부의 어떤 센서를 사용해서 정보를 축적하고 있는 경우, 기계 학습을 이용해서 분석하는 것으로 유익한 결과를 얻을 수도 있다. 음성 인식이나 문자 인식과 같은 패턴 인식 그리고 금융, 증권 시장의 동향 분석, 유전자 분석, 기상 예측, 교통 예측 등등, 다양한 분야에서 기계 학습을 사용하고 있다.

지금까지 든 예에서도 검색 인덱스를 기반으로 기계 학습을 사용함으로써 분석 정확도를 높인 경우가 많다. 기계 학습에서 다루는 데이터가 많은 경우는 대량의 계산이 필요하기 때문에 Hadoop이 최적의 솔루션이 될 수 있다.

지금까지 Hadoop 개요에 대해 설명했다. Hadoop에 대해서 조금이라도 이해할 수 있었다면 성공이다.

- Hadoop은 테라바이트, 페타바이트급의 데이터를 일괄 처리하는데 최적화되어 있으며, 서버를 추가하는 것으로 성능(확장성)을 향상시킬 수 있다
- HDFS라고 불리는 분산 파일 시스템과 MapReduce라 불리는 데이터 처리 프레임워크로 구성된다
- Hadoop은 어떤 처리에도 적용할 수 있는 만능 시스템이 아니다. RDBMS나 다른 미들웨어와 잘 조합하여 적재적소에 사용하는 것이 바람직하다

위 사항을 이해했다면, Hadoop 개요는 충분히 습득한 것이다. Hadoop은 새로운 미들웨어다. 때문에 그 능력을 최대한 끌어내기 위한 포인트가 존재한다. 다음 장부터는 Hadoop의 상세 구조를 설명하고, 그 능력을 최적으로 사용할 수 있는 기능들에 대해 깊이 있게 다루도록 하겠다.