그림으로 공부하는 IT 인프라 구조

絵で見てわかるITインフラの仕組み
(E de Mite Wakaru IT infra no Shikumi : 2573-2)

Copyright © 2012 by Yasushi Yamazaki, Yoshiko Minawa, Yohei Azekatsu, Takahiko Sato, Keiji Oda.
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 © 2015 by J-PUB

이 책의 한국어판 저작권은 대니홍 에이전시를 통한 저작권사와의 독점 계약으로 제이펍에 있습니다.
저작권법에 의해 한국 내에서 보호를 받는 저작물이므로 무단전재와 복제를 금합니다.

초판 1쇄 발행 2015년 7월 20일

지은이 야마자키 야스시, 미나와 요시코, 아제카츠 요헤이, 사토 타카히코
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** 민지환, 현지환
표지디자인 미디어픽스 / **본문디자인** 북아이
용지 신승지류유통 / **인쇄** 해외정판사 / **제본** 광우제책사

ISBN 979-11-85890-26-5 (93000)
값 25,000원

※ 이 책은 저작권법에 따라 보호를 받는 저작물이므로 무단 전재와 무단 복제를 금지하며,
이 책 내용의 전부 또는 일부를 이용하려면 반드시 저작권자와 제이펍의 서면동의를 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)**

그림으로 공부하는 IT 인프라 구조

야마자키 야스시 / 미나와 요시코 / 아제카츠 요헤이 / 사토 타카히코 지음
오다 케이지 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.
- 이 책에서 사용하고 있는 제품 버전은 독자의 학습 시점이나 환경에 따라 책의 내용과 다를 수 있습니다.
- 본문 중 일본 내의 실정에만 국한되어 있는 내용이나 그림은 일부를 삭제하거나 국내 실정에 맞도록 변경하였으니 참고 바랍니다.
- 3계층형 시스템의 구성에 관한 그림을 제이펍의 이 책 소개 페이지에서 별도의 PDF 파일로 제공하고 있습니다. 4~5장 학습과도 연계되어 있으니 꼭 다운로드하시기 바랍니다.
- 책 내용과 관련된 문의사항은 옮긴이나 출판사로 연락해 주시기 바랍니다.
  - 옮긴이: jinsilto@gmail.com
  - 출판사: jeipub@gmail.com

차례

옮긴이 머리말 xv
머리말 xvi
저자 소개 xix
감수자 소개 xxi
베타리더 후기 ... xxii

CHAPTER 1 인프라 아키텍처를 살펴보자 1

1.1 | 시작하며 2
칼럼 궁극의 아키텍처와 최적의 아키텍처 3

1.2 | 집약형과 분할형 아키텍처 4
1.2.1 집약형 아키텍처 4
1.2.2 분할형 아키텍처 7

1.3 | 수직 분할형 아키텍처 10
1.3.1 클라이언트-서버형 아키텍처 10
1.3.2 3계층형 아키텍처 12
칼럼 웹은 클라이언트-서버형을 대체할 수 있을까? 14

1.4 | 수평 분할형 아키텍처 14
1.4.1 단순 수평 분할형 아키텍처 15
1.4.2 공유형 아키텍처 16
칼럼 가상화 진행 상황 18

1.5 | 지리 분할형 아키텍처 19
1.5.1 스탠바이형 아키텍처 19
1.5.2 재해 대책형 아키텍처 20
1.5.3 클라우드형 아키텍처 22
칼럼 기술은 대물림되고 있다 23

CHAPTER 2 서버를 열어 보자 25

2.1 | 물리 서버 26
2.1.1 서버 외관과 설치 장소 26
2.1.2 서버 내부 구성 29

2.2 | CPU 31

2.3 | 메모리 33

2.4 | I/O 장치 36
칼럼 코드네임의 정체 36
2.4.1 하드 디스크 드라이브(HDD) 36
2.4.2 네트워크 인터페이스 39
2.4.3 I/O 제어 40
칼럼 조립 PC 추천 43

2.5 | 버스 43
2.5.1 대역 44
2.5.2 버스 대역 45
칼럼 불사조 InfiniBand 48

2.6 | 정리 49

CHAPTER 3 3계층형 시스템을 살펴보자 51

3.1 | 3계층형 시스템의 구성도 52

3.2 | 주요 개념 설명 53

3.2.1 프로세스와 스레드 53

칼럼 막대 인간의 모험 57

3.2.2 OS 커널 58

칼럼 커널은 결코 견고하지 않다 62

3.3 | 웹 데이터 흐름 63

3.3.1 클라이언트 PC부터 웹 서버까지 64

3.3.2 웹 서버부터 AP 서버까지 68

칼럼 데이터와 함께 전달되는 당신을 향한 마음 71

3.3.3 AP 서버부터 DB 서버까지 71

칼럼 RDBMS와 KVS의 소리 없는 전쟁 75

3.3.4 AP 서버부터 웹 서버까지 75

3.3.5 웹 서버부터 클라이언트 PC까지 76

3.3.6 데이터 흐름 정리 77

칼럼 높은 하늘을 날다 — 조감도 78

CHAPTER 4 인프라를 지탱하는 기본 이론 79

4.1 | 웹 데이터 흐름 80

4.1.1 직렬/병렬이란? 80

4.1.2 어디에 사용되나? 83

4.1.3 정리 85

칼럼 병렬과 병행 86

4.2 | 동기/비동기 86

4.2.1 동기/비동기란? 86

4.2.2 어디에 사용되나? 88

4.2.3 정리 92

칼럼 C10K 문제 93

4.3 | 큐 **94**
4.3.1 큐란? 94
4.3.2 어디에 사용되나? 95
4.3.3 정리 100

4.4 | 배타적 제어 **101**
4.4.1 배타적 제어란? 101
4.4.2 어디에 사용되나? 103
4.4.3 정리 105
칼럼 멀티 프로세서 시스템에서는 배타적 제어가 어렵다 106

4.5 | 상태 저장/상태 비저장 **107**
4.5.1 상태 저장/상태 비저장이란? 107
4.5.2 자세히 살펴보자 109
4.5.3 어디에 사용되나? 110
4.5.4 정리 113

4.6 | 가변 길이/고정 길이 **113**
4.6.1 가변 길이/고정 길이란? 113
4.6.2 어디에 사용되나? 116
4.6.3 정리 118

4.7 | 데이터 구조(배열과 연결 리스트) **119**
4.7.1 데이터 구조(배열과 연결 리스트)란? 119
4.7.2 어디에 사용되나? 121
4.7.3 정리 123

4.8 | 탐색 알고리즘(해시/트리 등) **124**
4.8.1 탐색 알고리즘(해시/트리 등)이란? 124
4.8.2 어디에 사용되나? 125
4.8.3 정리 130

CHAPTER 5 인프라를 지탱하는 응용 이론 133

5.1 | 캐시 134
5.1.1 캐시란? 134
5.1.2 어디에 사용되나? 135
5.1.3 정리 136

5.2 | 끼어들기 139
5.2.1 끼어들기란? 139
5.2.2 자세히 보자 140
5.2.3 어디에 사용되나? 141
5.2.4 정리 143

5.3 | 폴링 144
5.3.1 폴링이란? 144
5.3.2 어디에 사용되나? 145
5.3.3 정리 147

5.4 | 핑퐁 149
5.4.1 핑퐁이란? 149
5.4.2 어디에 사용되나? 151
5.4.3 정리 156

5.5 | 저널링 156
5.5.1 저널링이란? 156
5.5.2 어디에 사용되나? 157
5.5.3 정리 159
칼럼 변화는 항상 순식간에 일어난다 162

5.6 | 복제 163
5.6.1 복제란? 163
5.6.2 어디에 사용되나? 164
5.6.3 정리 166

5.7 | 마스터-슬레이브 168
5.7.1 마스터-슬레이브란? 168
5.7.2 어디에 사용되나? 170
5.7.3 정리 171

5.8 | 압축 172
5.8.1 압축이란? 172
5.8.2 자세히 보자 173
5.8.3 어디에 사용되나? 175
5.8.4 정리 178

5.9 | 오류 체크/오류 수정 178
5.9.1 오류 체크/오류 수정이란? 178
5.9.2 자세히 보자 179
5.9.3 실수나 오류를 검출하려면 180
5.9.4 어디에 사용되나? 181
5.9.5 정리 183

CHAPTER 6 시스템을 연결하는 네트워크 구조 185

6.1 | 네트워크 186

6.2 | 【기초】 계층 구조란? 187
6.2.1 회사를 계층 구조에 비유하다 187
6.2.2 계층 구조는 역할 분담 188
6.2.3 계층 모델의 대표적인 예 — OSI 7계층 모델 189
6.2.4 계층 구조는 네트워크 외에도 존재한다 190

6.3 | 【기초】 프로토콜이란? 191
6.3.1 사람끼리의 의사소통도 프로토콜 192
6.3.2 컴퓨터에서는 프로토콜이 필수 불가결이다 193
칼럼 표준화 단체에 대해서 194
6.3.3 프로토콜은 서버 내부에도 존재한다 195

6.4 | TCP/IP를 이용하고 있는 현재의 네트워크 196
6.4.1 인터넷의 발전과 TCP/IP 프로토콜 196
6.4.2 TCP/IP 계층 구조 197

6.5 | 【레이어 7】 애플리케이션 계층의 프로토콜 HTTP 199
6.5.1 요청과 응답의 구체적인 내용 200
6.5.2 애플리케이션 프로토콜은 사용자 공간을 처리 202
6.5.3 소켓 이하는 커널 공간에서 처리 203
칼럼 한 번 잡으면 놓아주지 않는다 204

6.6 | 【레이어 4】 전송 계층 프로토콜 TCP 205
6.6.1 TCP의 역할 206
칼럼 인터넷의 주인은 누구? 207
6.6.2 커널 공간의 TCP 처리 흐름 208
6.6.3 포트 번호를 이용한 데이터 전송 210
6.6.4 연결 생성 210
6.6.5 데이터 보증과 재전송 제어 212
6.6.6 흐름 제어와 폭주 제어 214

6.7 | 【레이어 3】 네트워크 계층의 프로토콜 IP 216
6.7.1 IP의 역할 217
6.7.2 커널 공간의 IP 처리 흐름 217
6.7.3 IP 주소를 이용한 최종 목적지로의 데이터 전송 219
칼럼 IP 주소 고갈과 IPv6 221
6.7.4 라우팅 222
칼럼 IP 헤더에서 체크섬이 사라진 날 224

6.8 | 【레이어 2】 데이터 링크 계층의 프로토콜 이더넷 224
6.8.1 이더넷의 역할 225
6.8.2 커널 공간의 이더넷 처리 흐름 225
6.8.3 동일 네트워크 내의 데이터 전송 228
6.8.4 VLAN 229

6.9 | TCP/IP를 이용한 통신 이후 231

6.9.1 네트워크 스위치 중계 처리 231
6.9.2 최종 목적지의 수신 확인 233
칼럼 NIC의 끼어들기와 패킷 처리 235

CHAPTER 7 무정지를 위한 인프라 구조 237

7.1 | 안정성 및 이중화 238
7.1.1 안정성이란? 238
7.1.2 이중화란? 239
칼럼 장애 보호 240

7.2 | 서버 내 이중화 241
7.2.1 전원, 장치 등의 이중화 241
7.2.2 네트워크 인터페이스 이중화 242

7.3 | 저장소 이중화 248
7.3.1 HDD 이중화 248
7.3.2 버스 이중화 254

7.4 | 웹 서버 이중화 256
칼럼 장애 괴담 첫 번째 이야기, '벌써 시간이 다 됐어?' 256
7.4.1 웹 서버의 서버 이중화 256
7.4.2 서버 이중화 259
칼럼 DSR(Direct Server Return) 264

7.5 | AP 서버 이중화 265
7.5.1 서버 이중화 265
7.5.2 DB 연결 이중화 267

7.6 | DB 서버 이중화 270
7.6.1 서버 이중화(액티브-스탠바이) 270
7.6.2 서버 이중화(액티브-액티브) 274
칼럼 장애 괴담 두 번째 이야기, '진단 때문에 죽었다' 279

7.7 | 네트워크 장비 이중화 279
7.7.1 L2 스위치 이중화 279
7.7.2 L3 스위치 이중화 283
7.7.3 네트워크 토폴로지 286
칼럼 장애 괴담 세 번째 이야기, '브로드캐스트 스톰' 289

7.8 | 사이트 이중화 290
7.8.1 사이트 내 이중화 전체 구성도 290
7.8.2 웹사이트 이중화 291

7.9 | 감시 293
7.9.1 감시란? 293
7.9.2 생존 감시 294
7.9.3 로그 감시 295
7.9.4 성능 감시 296
7.9.5 SNMP 297
7.9.6 콘텐츠 감시 299

7.10 | 백업 301
7.10.1 백업이란? 301
7.10.2 시스템 백업 302
7.10.3 데이터 백업 304

7.11 | 정리 305

CHAPTER 8 **성능 향상을 위한 인프라 구조** 307

8.1 | 응답과 처리량 308
8.1.1 성능 문제의 두 가지 원인 308
칼럼 가장 중요한 응답 시간은? 311
8.1.2 응답 문제 312
8.1.3 처리량 문제 314

8.2 | 병목 현상이란? 315

칼럼 CAP 정리를 익혀서 초능력자가 되자 317

8.2.2 병목 현상은 어떻게 해결하는가? 318

8.2.3 병목 지점은 반드시 존재한다 319

칼럼 병목 현상의 숙명의 적, 데이터베이스 321

8.3 | 3계층형 시스템 그림을 통해 본 병목 현상 321

8.3.1 CPU 병목 현상 예 322

칼럼 여유가 있는 노련한 시스템 324

칼럼 C는 자바보다 빠르다? 333

8.3.2 메모리 병목 현상 예 334

8.3.3 디스크 I/O 병목 현상 예 338

칼럼 아이들을 공원에서 놀게 하자 341

칼럼 ORDER(N) - 일인분 나왔습니다 344

8.3.4 네트워크 I/O 병목 현상 344

칼럼 대역이 가장 중요한 것일까? 347

8.3.5 애플리케이션 병목 현상 예 350

8.4 | 정리 354

찾아보기 355

옮긴이 머리말

네트워크 구조를 설계하는 인프라 엔지니어나 소프트웨어 혹은 시스템을 개발하는 소프트웨어 엔지니어들이 공통으로 안고 있는 문제가 있다. 어떤 사람은 숲을 보고서 큰 그림을 그릴 수 있지만 나무를 보지 못해서 상세한 구조를 이해하지 못하는 문제가 그것이다. 반대로, 또 어떤 사람은 나무를 보고서 개별 시스템 컴포넌트나 장비에 대한 지식은 해박하지만 숲을 보지 못해서 전체적인 구조를 설계하지 못한다.

이 책은 숲(전체 구조)을 보여주고 나무(개별 요소)에 대해서도 상세히 알려 준다. 시스템 및 인프라 구성이라는 거시적 내용부터 CPU나 스레드 단위의 미시적 내용까지 다루고 있어서 실용성이 높은 책이라고 할 수 있다. 또한, 거시적/미시적 관점의 인프라뿐만 아니라 인프라를 지탱하고 있는 배경 이론과 실용 기술까지 다루고 있어서 핵심 알고리즘 이론도 다시 짚어 보고 넘어가는 책이다.

특히 머릿속에서만 맴돌던 이론이나 개별 기술, 설계 방식, 성능 관련 이슈들을 그림을 통해 명확하게 설명하고 있어서 여기저기 흩어져 있던 지식이 마치 퍼즐 맞추듯이 자신의 지식으로 만들어지는 것을 느낄 수 있을 것이다.

안정성과 성능, 기본 컴퓨터 이론과 네트워크 이론을 한 번에 잡고 싶은 입문자와 어느 정도 지식은 있지만 막상 업무에 적용하려면 자신이 제대로 알고 있는 건지 확신이 들지 않는 사람들에게 꼭 추천해 주고 싶은 책이다. 특히 컴퓨터 구조가 알고 싶은 사람, 프로세스와 스레드의 차이가 아직도 이해가 안 가는 사람, 웹 시스템에서 데이터가 어떻게 흐르는지 눈으로 보고 싶은 사람, OSI 7계층을 아무리 읽어도 이해하지 못하겠다는 사람, 해시와 트리의 차이는 알지만 어떤 것을 언제 사용해야 하는지 헷갈리는 사람이라면 이 책이 좋은 선생이 될 것이다.

2015년 7월

역자 **김완섭**

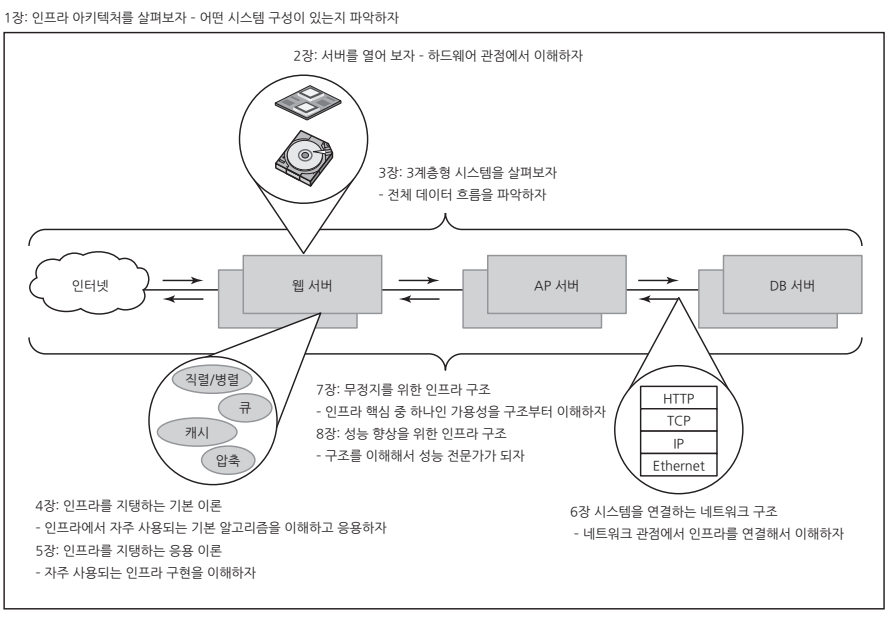
머리말

여러분이 일상에서 접하고 있는 IT 시스템은 그 구조가 매우 고도화되면서 복잡해지고 있다. 이런 복잡성을 지탱하기 위해 엔지니어의 기술 전문성이 중시되고 있으며, 결과적으로 엔지니어가 IT 시스템 전반을 살펴볼 기회가 적어지고 있다. 또, 요소 기술의 전문성이 높아짐으로써 자기 분야 외의 기술을 다룰 기회가 없어지고 있다.

하지만 현장에서는 IT 시스템 기획 및 설계 단계뿐만 아니라 성능 튜닝이나 문제 해결 등에 IT 시스템 전반을 조감하고 여러 기술을 조합해서 파악하는 능력이 요구되고 있다. IT 시스템의 세부 요소들을 보면 실제로는 평범한 상식에 의해 지탱되고 있는 것이 많다. 각 전문 영역에서 사용되고 있는 기술 요소나 이론이 실제로는 다른 영역에서도 이용되고 있다.

이 책은 평범한 상식을 기반으로 'IT 인프라'라는 영역 전체를 미시적, 거시적 관점에서 알기 쉽도록 그림을 통해 설명하고 있다. IT 인프라에 대한 상식(급소)을 이해함으로써 IT 시스템에 대한 이해가 깊어질 것이다. 또, 전문 분야 외의 새로운 기술을 접하더라도 그 본질을 이해할 수 있는 기초 체력을 단련할 수 있을 것이다.

이 책은 다음과 같이 구성돼 있다.



이 책은 IT 관련 일을 시작해서 5년차 정도까지의 엔지니어를 대상으로 하고 있다. 자신의 업무 분야에 대해서 식견이 깊어지면서 IT 인프라 전반에 대해 배우고 싶은 사람에게 추천한다.

애플리케이션 개발 엔지니어나 프로그래머도 일상에서 사용하고 있는 알고리즘이 IT 인프라와 밀접한 관련이 있다는 것을 알 수 있는 기회가 될 것이다. 또, 베테랑 엔지니어도 이전에 몰랐던 사실들을 새롭게 접할 수 있을 것이다. '예외가 있긴 하지만 기본적으로는 이런 식으로 생각하면 이해하기 쉬워요'라는 방침을 가지고 집필했다.

감사의 글

이 책을 집필하면서 많은 분께서 도움을 주셨다. 다시 한 번 감사드린다. 일부이지만 이름을 게재할 수 있도록 허락해 주신 분들의 이름을 기재해 보았다.

- 네트워크 기술 구현에 관한 검토와 조언을 주신 이토추 테크노솔루션즈의 하야시 신야 님
- 웹 서버, AP 서버 구조에 대한 검토와 조언을 주신 일본 오라클의 사쿠마 야스히로 님
- 서버 구조에 대한 검토와 조언을 주신 일본 오라클의 마루야마 타케시 님
- 커널 내부 동작에 대한 검토와 조언을 주신 이시바시 켄이치 님
- 그림을 예쁘게 만들어 주신 호리 아키코 님
- 서버 사진 촬영을 허락해 주신 일본 오라의 야치다 노리히토 님, 오소네 아키라 님, 사와후지 코가 님

저자 소개

이 책은 일본 오라클에 재직하고 있는 컨설턴트들이 집필했다.

야마자키 야스시(山崎 泰史) 1장, 3장, 8장 집필

인프라 컨설턴트. 요코하마에서 태어났고 미국 텍사스에서 자랐다. 오라클 데이터베이스를 오른손에, 잔재주를 왼손에 들고서 IT 인프라 속에 사는 도깨비들과 매일 같이 싸우고 있다. IT 관련 일에 종사하는 분들이 조금이라도 일찍 퇴근할 수 있도록 노력하고 있다.

미나와 요시코(三縄 慶子) 2장, 5장 일부, 7장 집필

도쿄 출생의 문과생. 이토추 테크노솔루션즈㈜에 신입사원으로 입사한 후 현재는 일본 오라클에서 근무 중이다. 서버의 중저음을 자장가처럼 생각하면서 일상을 보내고 있다. IT 시스템에 대한 사랑을 관철하면서 인프라 ➡ 데이터베이스 ➡ 업무 애플리케이션으로 담당 업무를 바꾸는 중이다. 현장에서 평생 일하는 것을 목표로 열심히 일하고 있다.

아제카츠 요헤이(畔勝 洋平) 4장, 5장 일부, 8장 일부 집필

오사카 출생으로 나라, 와카야마에서 자랐다. 대학 졸업 후에는 오사카의 한 벤처에서 웹 디자이너로 일했다. 그 후 프리랜서 엔지니어로 경험을 쌓았고, 주식회사 와곤에서 DBA로 일했다. 《그림으로 공부하는 OS/저장소/네트워크》 집필로 인한 인연으로 일본 오라클로 전직했다. 사물을 분해하고 재조립해서 그 뒤에 숨겨져 있는

보편적인 원리 탐구를 즐긴다.

사토 타카히코(佐藤 貴彦) **4장 일부, 5장 일부, 6장**

컨설턴트라기보다는 만물상에 가깝다. 나라 첨단과학기술 대학원대학에서 네트워크를 전공했고, 인프라가 가진 재미에 눈을 뜨고 있다. 졸업 후에는 일본 오라클에 입사해서 데이터베이스를 중심으로 한 인프라부터 애플리케이션까지 폭넓게 다루고 있다. 몸 쓰는 것을 좋아해서 요즘은 암벽타기에 열중하고 있다.

감수자 소개

오다 케이지(小田 圭二)

최근에는 컨설턴트 육성에 힘을 쏟고 있다. 이 책도 그 일환이다. 주요 저서 및 관련서로는 《나만 알고 싶은 오라클 실무 테크닉》, 《그림으로 공부하는 시스템 구축을 위한 오라클 설계》, 《그림으로 공부하는 오라클 구조》, 《그림으로 공부하는 OS/저장소/네트워크》, 《44개 안티 패턴으로 배우는 DB 시스템》 등이 있다.

베타리더 후기

권영철(바이널아이)

회사 업무에만 빠져서 IT 인프라 구조를 막연하게만 생각했었는데, 막상 책을 보고 나니 그리 간단하지 않다는 것을 알았습니다. 이 책을 읽고 난 후 IT 인프라에 대한 이해를 넓히고 회사 업무를 해나간다면 큰 도움이 될 것 같습니다.

김민수(프리랜서)

다소 막연했던 인프라와 서버 아키텍처라는 분야를 매우 짜임새 있게 정리할 수 있는 기틀을 잡아주는 책이었습니다. 비교적 최신의 동향까지도 놓치지 않게 일러주고, 조금씩 단계를 나아갈 때마다 하나의 그림이 완성되어 가는 듯한 희열이 있더군요. 자칫 지루한 내용으로 흐를 수 있는 부분도 재미있는 예시와 유머러스한 일화로 독자의 집중도를 잃지 않게 하려는 소소한 배려도 돋보였습니다. 좋은 책을 선정하신 제이펍과 역자께도 감사의 말씀을 드립니다.

도경원(네이버)

인프라라는 거대한 숲을 모두 보기는 힘듭니다. 이 책은 거대한 숲의 나무 하나하나에 대해 자세히 설명하지는 않지만, IT 인프라를 한번 훑어보는 데는 큰 도움이 되었습니다.

송영준(줌인터넷)

자료구조, 네트워크, 시스템 엔지니어링 등에 필요한 내용이 다양한 그림과 함께 실생활에서 찾을 수 있는 재미있는 비유로 잘 설명되어 있습니다. 다른 책에서 보였던 이해하기 힘든 설명과 지루한 구성에 힘들어했던 독자들에게 단비와 같은 책입니다.

이아름

IT 인프라스트럭처라는 용어가 생소하던 때가 있었지만, 이제는 반드시 공부해야 하는 주제가 된 것 같습니다. 베타리딩하면서 느낀 이 책의 장점은 글로만 보면 이해하기 어려운 부분을 그림으로 풀어내어 잘 설명한 점이라고 생각합니다.

이재빈(연세대학교)

IT 회사 입사를 준비하고 있는 분이나 종사하고 분들 모두에게 전반적인 IT 시스템 구성을 쉽게 파악할 수 있도록 도와주는 책인 것 같습니다. 더불어 생산성의 극대화를 위해서는 어떻게 인프라를 구축해야 하는지에 대한 질문에 힌트를 얻을 수도 있을 것 같습니다.

제이펍
베타리더스

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

CHAPTER

1

인프라 아키텍처를 살펴보자

먼저 대표적인 인프라 아키텍처를 소개한다. 역사뿐만 아니라 각각의 구조가 생겨난 이유를 생각하면서 읽도록 하자. 또한, 어떤 구조든 반드시 장점과 단점이 존재한다는 것을 이해하도록 하자.

1.1 시작하며

갑작스런 질문이긴 하지만, '인프라(infra)'라고 하면 무엇이 떠오르는가?

전기, 수도, 가스 등 가정에서 이용하는 것이나 지하철, 버스처럼 공공 목적의 인프라를 떠올릴 수 있을 것이다. 인프라를 우리말로 하면 '기반'이란 뜻으로, 여러분의 생활을 지탱하는 바탕이나 토대란 의미다. 인프라 구조 자체는 복잡하지만, 전문가에 의해 관리되고 있어서 사용자는 그 구조를 이해하지 않고도 간단히 이용할 수 있다는 특징이 있다.

'IT 인프라'도 마찬가지다. IT의 기반이 되는 것으로서 이 역시 여러분의 생활을 지탱하고 있다. 일상에서 사용하고 있는 인터넷 검색 엔진을 생각해 보자. 검색 키워드를 입력하고 검색 버튼을 누르면 많은 검색 결과를 얻을 수 있다. 이런 방대한 데이터는 어떻게 관리되고 있는지 생각해 본 적이 있는가? 이것을 지탱하고 있는 것이 IT 인프라다.

그러면 이번 장의 제목이기도 한 '인프라 아키텍처'란 무엇일까?

아키텍처란, 직역하면 '구조'라는 의미다. 여기서는 기차를 예로 들겠다. 다양한 기차 종류가 있지만, 그 구조 자체는 거의 같다. 전기로 움직이거나, 여러 객차가 연결돼 있고, 내부에는 좌석이나 손잡이가 있다. 즉, 기차의 '구조' 또는 '아키텍처'가 확립돼 있어서 이미 공통화돼 있다.

'인프라 아키텍처'는 IT 인프라의 '구조'를 의미한다. 인터넷 검색 시스템이나 항공회사 티켓 발권 시스템, 편의점의 계산대 등 모두 이용 방법이나 사용자가 다르지만 IT 인프라 위에서 동작하고 있다. 그리고 이 '인프라 아키텍처'는 실은 놀라울 정도로 닮아 있어서 거의 같은 구조를 가지고 움직이고 있다.

이번 장에서는 현재 IT 업계에서 주류가 되고 있는 인프라 아키텍처에 대해 그림 1.1에 있는 순서대로 설명해 가도록 하겠다.

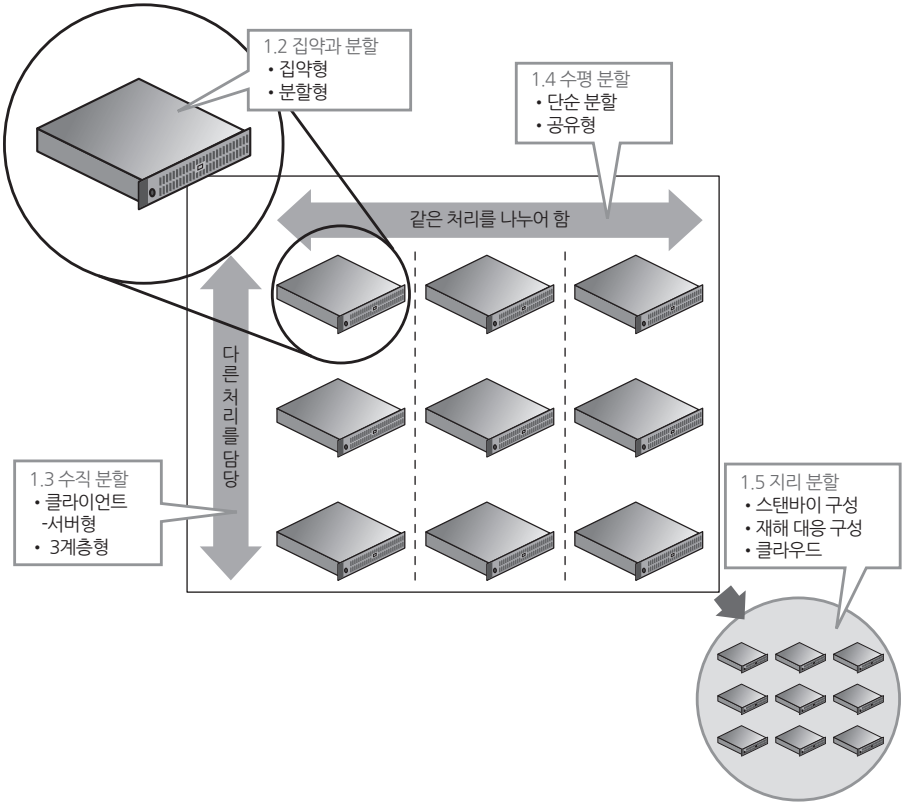


그림 1.1 이번 장에서 소개하는 인프라 아키텍처

칼럼 **궁극의 아키텍처와 최적의 아키텍처**

'이 아키텍처 누가 설계한 거야?!'

'1주일만 기다려 주세요. 진정한 아키텍처를 보여 드리겠습니다.'

이런 대화는 시스템 설계 현장에서 자주 볼 수 있는 광경이다.

이 책에서는 다양한 아키텍처 구성과 기술 구성 요소 및 이중화, 성능 등에 대해 설명하고 있다. 읽다 보면 의문이 생길 것이다. '어떤 시스템이든지 요건은 거의 비슷하다. 그러면 하나의 만능 아키텍처를 가지고 모든 프로젝트에 적용할 수는 없을까?', '궁극의, 최적의 아키

텍처 하나만 있다면 굳이 설계를 하지 않아도 괜찮지 않을까?'

대답은 'NO'다. 왜냐하면 아키텍처나 설계 요소에는 반드시 장점과 단점이 공존한다. 장점만 있다면 가장 좋은 것을 취하면 되지만, 단점은 가장 영향력이 적은 것으로 선택하는 것이 어렵기 때문에 반드시 취사선택 상황이 발생한다.

그중에서도 가장 제약이 많은 것이 시스템 도입 비용이다. 예를 들어, 100만 명이 사용하는 대규모 웹 서비스의 예산과 사내에서 10명이 사용하는 시스템의 예산은 차이가 크다. 하지만 중요도라는 관점에서는 이용자에겐 양쪽 시스템이 다 중요하다. 많은 현장에서 비용 제약이 심해지고 있는 가운데, 시스템의 가장 중요한 장점은 살리고 단점을 최소화하도록 설계하는 것이 중요하다.

'설계? 다 똑같은 거 아닌가?'라는 얘기를 들어도 개의치 말고, 시스템에 따라 중요한 사항을 잘 찾아서 최적의 설계를 하도록 하자.

1.2 집약형과 분할형 아키텍처

IT 인프라는 컴퓨터로 구성된다. 기본적인 구성 방식에는 '집약형'과 '분할형'이 있다. 각각의 장단점을 비교해 보도록 하자.

1.2.1 집약형 아키텍처

옛날 영화에 등장하는 컴퓨터 장면에서는 방 하나에 가득 차는 기계와 거대한 카세트 테이프 같은 장치, 청색, 녹색, 황색 등으로 깜빡이는 램프, 그리고 안경을 낀 과학자들이 등장한다. 기업에서 사용하는 컴퓨터는 이 정도로 거대하진 않지만, IT 시스템의 여명기에는 대형 컴퓨터를 이용해서 모든 업무를 처리하는 형태가 대부분이었다(그림 1.2).

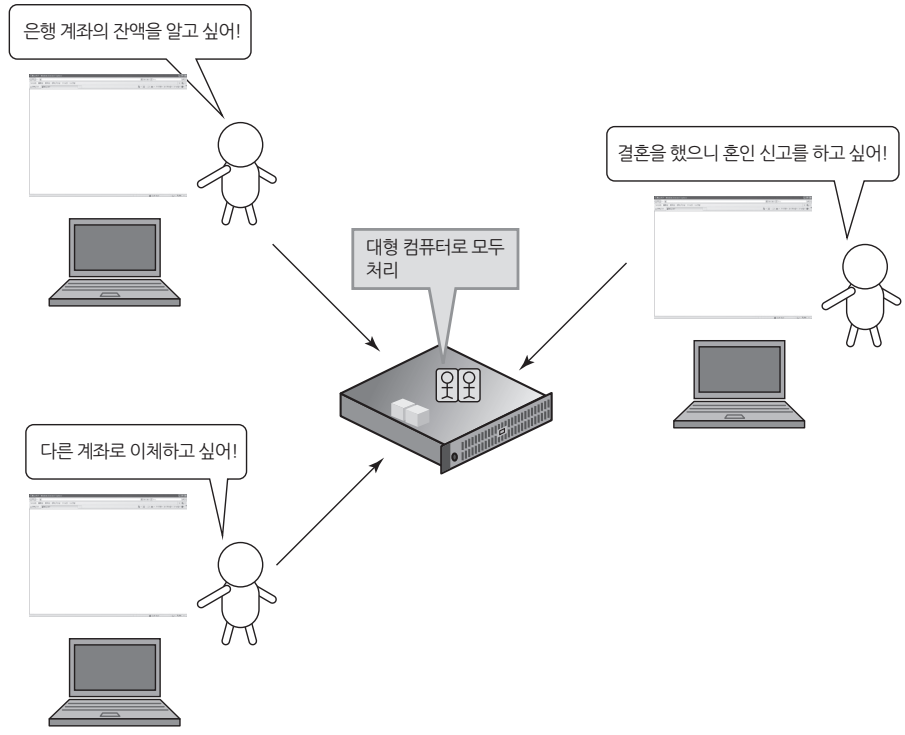


그림 1.2 집약형 아키텍처

이런 대형 컴퓨터는 '범용 장비', '호스트', '메인 프레임' 등으로 불렸다. 시스템 아키텍처라는 관점에서는 하나의 컴퓨터로 모든 처리를 하기 때문에 '집약형'이라고 할 수 있다. 집약형의 최대 장점은 구성이 간단하다는 것이다.

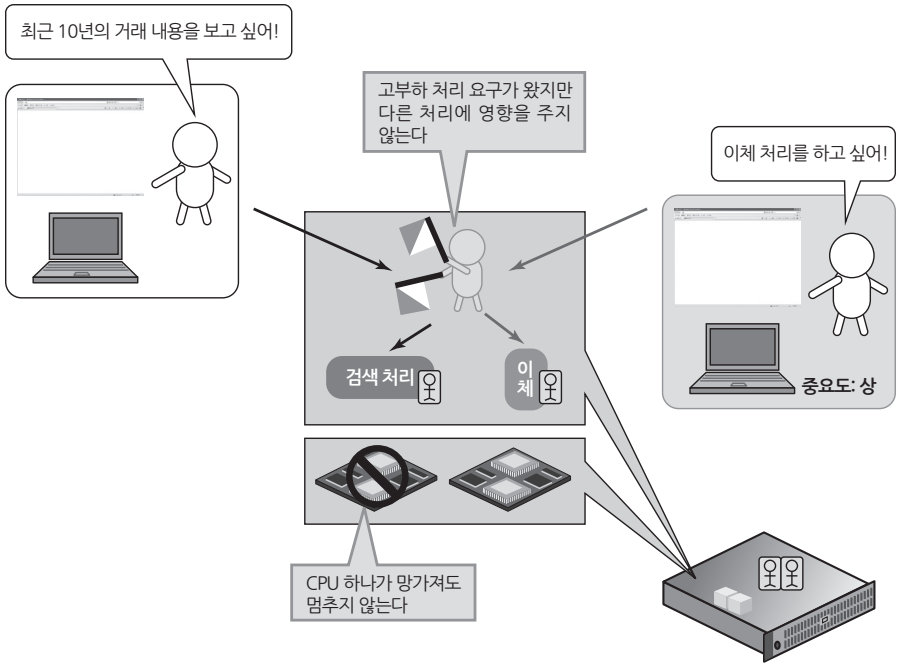


그림 1.3 대형 컴퓨터에 대한 고민

집약형 아키텍처에서는 해당 기업의 주요 업무를 모두 한 대로 처리하기 때문에 장비 고장 등으로 업무가 멈추지 않도록 여러 고민을 하고 있다. 예를 들어, 그림 1.3에 있는 것처럼 컴퓨터를 구성하는 주요 부품은 모두 다중화돼 있어서 하나가 고장 나더라도 업무를 계속할 수 있다. CPU를 포함한 구성 요소에 대해서는 2장에서 자세히 다루도록 한다.

또, 복수의 서로 다른 업무 처리를 동시에 실행할 수 있도록 유한 리소스 관리를 하고 있다. 이를 통해 하나의 처리가 실수로 대량의 요청을 보내더라도 다른 처리에 영향을 주지 않도록 되어 있다. 한 대의 컴퓨터라고 하지만, 그 안에 마치 여러 사람이 동거하고 있는 모습이라 할 수 있다.

많은 기업에서 아직까지 사용되고 있으며, 주로 '기간 시스템'이라 불리는 기업 내 핵심 업무 시스템에서 이용하고 있는 경우가 많다. 예를 들어, 은행이라면 '계정 시스템'이 여기에 해당된다.

단, 대형 컴퓨터는 도입 비용 및 유지 비용이 큰 경향이 있다. 또, 대형 컴퓨터의 파워가 부족하면 다른 한 대를 별도로 구매해야 해서 비용이 매우 많이 들며, 확장성에도 한계가 있다는 단점이 있다. 현재는 가격이 싸고 확장성이 높은 분할형이 주로 사용되고 있다. 이 구조에 대해서는 다음 절에서 설명하겠다.

장점

- 한 대의 대형 컴퓨터만 있으면 되므로 구성이 간단하다
- 대형 컴퓨터의 리소스 관리나 이중화에 의해 안정성이 높고 고성능이다

단점

- 대형 컴퓨터의 도입 비용과 유지 비용이 크다
- 확장성에 한계가 있다

1.2.2 분할형 아키텍처

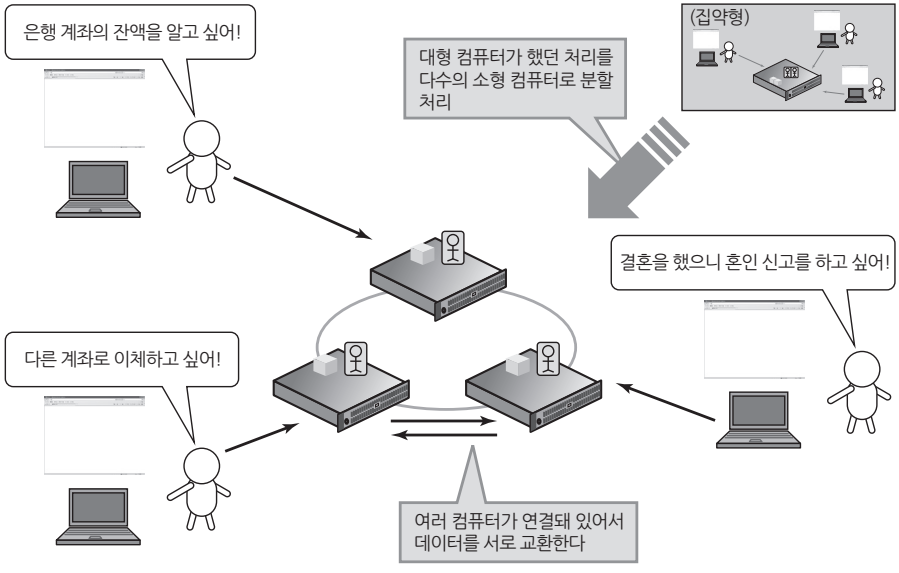


그림 1.4 분할형 아키텍처

분할형 아키텍처는 그림 1.4처럼 여러 대의 컴퓨터를 조합해서 하나의 시스템을 구축하는 구조다.

대형 컴퓨터는 안정성이 높고 고성능이지만, 현재는 소형 컴퓨터라도 충분한 성능을 낼 수 있다. 또, 대형 컴퓨터와 비교해서 소형 컴퓨터 구입 비용이 압도적으로 저렴하다. 그 가격 차이가 100배가 되는 경우도 있다. 단, 안정성은 한 대의 컴퓨터만으로는 대형 컴퓨터에 미치지 못한다. 이 결점을 보완하기 위해 분할형 아키텍처에서는 여러 대의 컴퓨터를 이용해 한 대가 고장 나도 안정성을 담보하고 있다.

분할형 아키텍처는 표준 OS나 개발 언어를 이용하기 때문에 '오픈 시스템'이라고도 부른다. 또, 여러 대의 컴퓨터를 연결해서 이용하기 때문에 '분산 시스템'이라 부르는 경우도 있다[1].

분할형의 장점은 개별 컴퓨터의 안정성이 낮아도 괜찮기 때문에 저가 장비를 이용해서 전체적인 비용을 줄일 수 있다는 것이다. 또한, 더 많은 컴퓨터를 이용해서 시스템 전체 성능을 향상시킬 수 있어서 확장성이 좋다는 특징도 있다.

하지만 서버 수가 늘어나면 이를 운영하기 위한 구조가 복잡해지는 경향이 있다. 또, 서버가 망가지면 영향 범위를 최소화하기 위해 서버별 역할을 세세하게 검토해야 한다. 서버를 분할하는 일반적인 방식으로는 수직형과 수평형의 두 가지가 있다. 이에 대해서는 다음 절에서 설명하겠다.

장점

- 낮은 비용으로 시스템을 구축할 수 있다
- 서버 대수를 늘릴 수 있어서 확장성이 높다

단점

- 대수가 늘어나면 관리 구조가 복잡해진다
- 한 대가 망가지면 영향 범위를 최소화하기 위한 구조를 검토해야 한다

1 분산 시스템이라는 용어는 '연결돼 있다'는 의미만 있는 것이 아니라 다른 의미도 있다. 흥미가 있는 사람은 찾아보도록 하자.

물리 서버와 논리 서버의 차이

분할형 아키텍처에서 이용되는 컴퓨터를 '서버'라고 한다. 서버라는 용어는 컴퓨터 자체(하드웨어)를 가리키는 경우도 있고, 컴퓨터에서 동작하고 있는 소프트웨어를 가리키는 경우도 있다.

서버라는 용어는 원래 '특정 역할에 특화된 것'을 의미한다. 레스토랑의 웨이터를 서버라고 부르는 경우도 있다. 웨이터의 역할은 주문 접수나 음식을 내오는 것에 특화돼 있으며 요리는 하지 않는다.

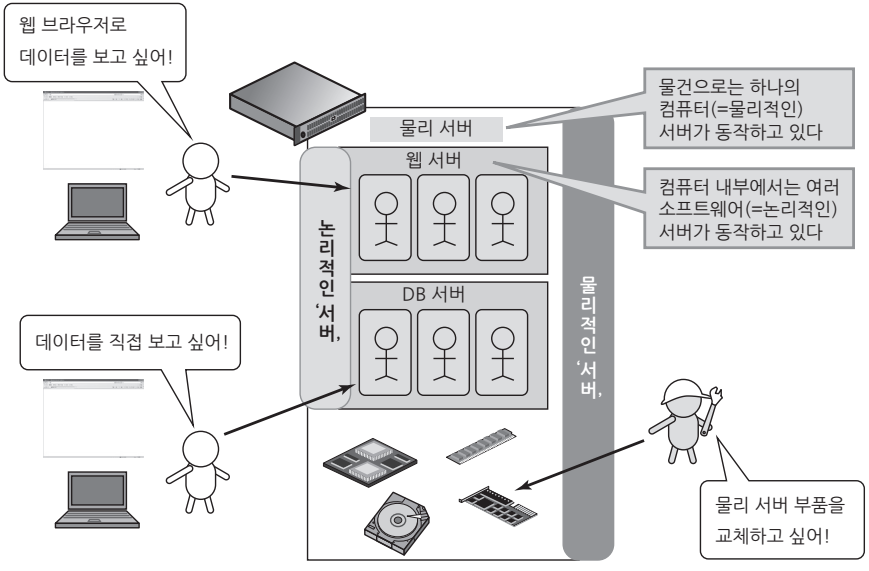


그림 1.5 서버라는 용어가 지니는 다양한 의미

그림 1.5를 보자. 예를 들어, 인터넷에 접속했을 때 사용자 입력 및 HTML 생성을 담당하는 것은 서버에서 동작하는 소프트웨어인 '웹 서버'다. 대량의 데이터를 저장해서 요청에 따라 데이터를 제공하는 것은 데이터베이스 기능을 제공하는 'DB 서버'다.

컴퓨터 자체를 가리키는 경우는 '물리 서버'라고 부른다. 특히 인텔의 x86 서버는 인텔 아키텍처(Intel Architecture)를 채용하고 있어서 'IA 서버'라 부른다. 이 IA 서버는 여러분이 사용하고 있는 일반 PC와 기본적인 아키텍처가 같다. 서버를 연상하

기 어려운 사람은 '모니터가 없는 PC'라고 이해하면 된다. 모양은 DVD 플레이어와 비슷하기 때문에 이것을 연상해도 괜찮다.

웹 서버나 DB 서버가 하나의 물리 서버에서 동작하는 경우도 있으며, 각각 별도의 물리 서버에서 동작하는 경우도 있다.

1.3 수직 분할형 아키텍처

분할형에서는 서버 분할 방식, 즉 역할 분담을 고려해야 한다. 각각의 서버가 전혀 다른 작업을 하는 것인지, 아니면 비슷한 작업을 하는 것인지에 대한 관점이다.

이번 절에서는 서버별로 다른 역할을 담당하는 '수직 분할형 아키텍처'를 소개한다. 수직형이라고 표현하는 것은, 특정 서버 측면에서 봤을 때 역할에 따라 '위' 또는 '아래' 계층으로 나뉘기 때문이다.

1.3.1 클라이언트-서버형 아키텍처

클라이언트-서버형은 수직 분할형의 한 예다. 그림 1.6처럼 업무 애플리케이션, 미들웨어, 데이터베이스 등의 소프트웨어를 '물리 서버' 상에서 운영하고 있다. 이들 소프트웨어에 '클라이언트' 또는 '단말'이라 불리는 소형 컴퓨터가 접속해서 이용하는 형태다. Client/Server의 앞글자를 따서 'C/S'라고 부르기도 한다.

클라이언트-서버형의 특징은 클라이언트 측에 전용 소프트웨어를 설치해야 한다는 것이다. 클라이언트(단말)에 주로 PC가 이용되지만, 최근에는 스마트폰이나 태블릿도 단말로 사용되고 있다. 예를 들어, 주식 판매 시스템은 그래프 표시나 주가 흐름 분석은 PC에서 하고, 필요에 따라 서버에서 주가 데이터를 취득하는 방식이 많다. 이 방식에서 서버 측은 데이터 입출력만 하면 되기에 처리당 부하가 낮아서 많은 PC가 동시에 요청을 보내도 문제가 없다.

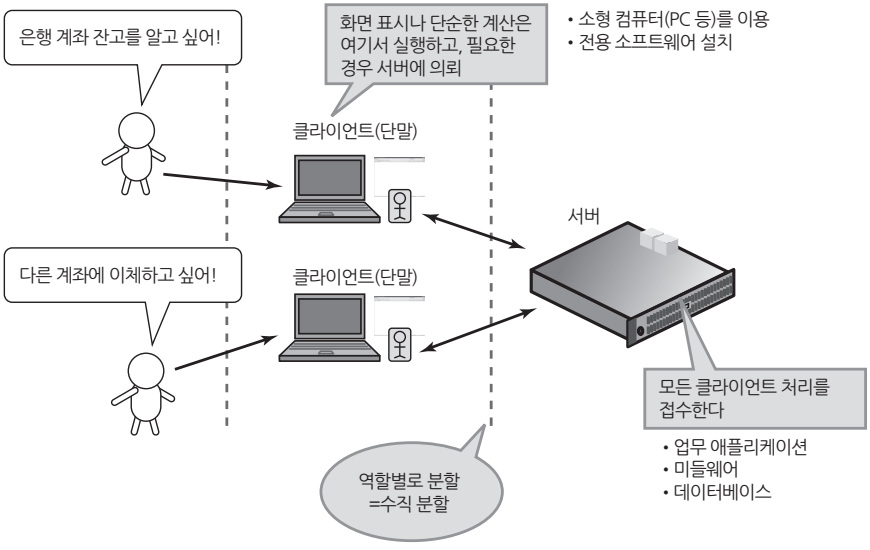


그림 1.6 클라이언트와 서버의 역할 분담

하지만 기업 IT 시스템에서는 업무 애플리케이션 기능 추가나 버그 수정 등으로 반드시 정기적인 업데이트가 필요하다. 클라이언트-서버형에서는 업무 애플리케이션 갱신 시마다 클라이언트 측 소프트웨어도 업데이트해야 한다. 여러분의 PC에도 마이크로소프트의 윈도우즈 업데이트나 오라클의 자바 업데이트 등이 자주 발생할 것이다. 하지만 귀찮아서 무시하고 넘어가는 경우가 많을 것이다. 이용자 측면에서는 마음껏 사용할 수 없다는 불편함이 발생하고, 이용자가 반드시 업데이트한다는 보장도 없어서 시스템 위험 요소가 될 수 있다. 또한, 서버에 처리가 집중되면 확장성에 한계가 발생할 가능성이 있다. 이런 단점을 개선하려고 한 것이 3계층형이다.

장점

- 클라이언트 측에서 많은 처리를 실행할 수 있어서 소수의 서버로 다수의 클라이언트를 처리할 수 있다

단점

- 클라이언트 측의 소프트웨어 정기 업데이트가 필요하다
- 서버 확장성에 한계가 발생할 수 있다

1.3.2 3계층형 아키텍처

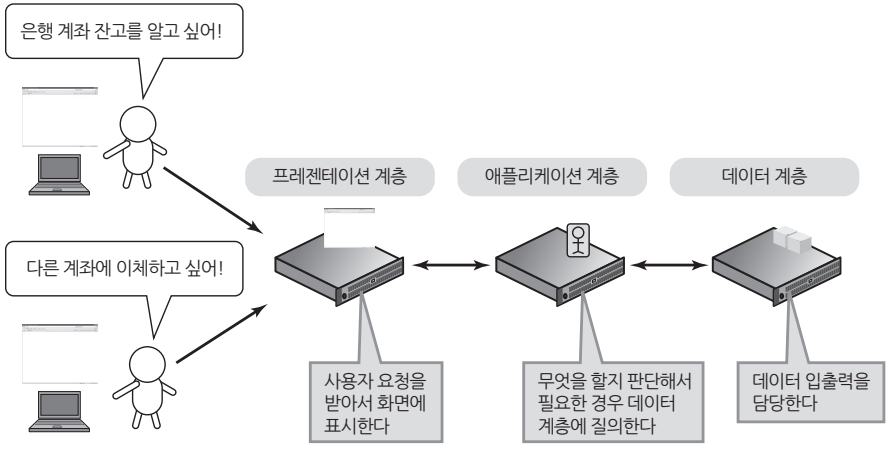


그림 1.7 3계층 서버의 역할 분담

3계층형은 수직 분할형의 한 가지 예인데, 클라이언트-서버형을 발전시킨 것이다. 그림 1.7과 같이 '프레젠테이션 계층', '애플리케이션 계층', '데이터 계층'의 3층 구조로 분할돼 있어서 3계층형이라고 부른다.

각 계층의 역할은 명확히 구분돼 있다.

프레젠테이션 계층

- 사용자 입력을 받는다
- 웹 브라우저 화면을 표시한다

애플리케이션 계층

- 사용자 요청(request)에 따라 업무 처리를 한다

데이터 계층

- 애플리케이션 계층의 요청에 따라 데이터 입출력을 한다

3계층 시스템에서는 사용자가 웹 브라우저를 통해 시스템에 접속한다. 예를 들어, 인터넷 검색 시스템에서는 사용자가 웹 브라우저에 입력하는 화면은 프레젠테이션 계층의 웹 서버에 먼저 전달된다. 웹 서버는 그 요청을 뒤에 있는 애플리케이션 계층의 애플리케이션 서버(AP 서버)에 전달한다. AP 서버는 검색 키워드를 바탕으로 무엇을 검색해야 할지 판단해서 뒤에 있는 데이터 계층의 데이터베이스(DB 서버)에 데이터를 요청한다. 이 처리 흐름은 3장에서 자세히 소개할 것이다.

클라이언트-서버형에 비해 특정 서버에 부하가 집중되는 문제가 해결된다는 것이 장점이다. 또한, 업무 애플리케이션 갱신에 따른 클라이언트 업데이트가 필요 없으며, 사용자는 웹 브라우저만 준비하면 된다.

게다가, 이 아키텍처에서는 모든 처리가 AP 서버나 DB 서버를 이용하지 않아도 된다. 예를 들어, 이미지 파일만 읽으면 되는 경우라면 웹 서버만으로도 처리를 완료해서 결과를 반환할 수 있으므로 다른 서버에 부하를 주지 않는다. 단, 시스템 전체 구조가 클라이언트-서버형보다 복잡하다.

장점

- 서버 부하 집중 개선
- 클라이언트 단말의 정기 업데이트가 불필요
- '처리 반환'에 의한 서버 부하 저감

단점

- 구조가 클라이언트-서버 구성보다 복잡하다

여러분이 일상에서 사용하고 있는 인터넷 사이트, 모바일 사이트, 사내 업무 시스템의 대부분이 이 3계층 구조를 채용하고 있다. 엔지니어로서 활약하고 있는 독자들도 업무상의 이유로 이런 시스템을 많이 다루고 있을 것이다. 이 책에서도 3계층형 아키텍처를 기본형으로 해서 다양한 알고리즘과 그 특성을 설명하고 있다.

칼럼 **웹은 클라이언트–서버형을 대체할 수 있을까?**

웹 시스템에서 많이 이용되는 것은 3계층형이다. 이 아키텍처는 클라이언트–서버형의 약점을 보완하기 위해 탄생했다. 사람들은 이것이 궁극의 아키텍처 탄생의 신호라고 생각했었다.

여러분은 어떤 웹 브라우저를 이용하고 있는가? 브라우저 세계에서는 경쟁이 심해서 기술 혁신이 빠르게 진행되고 있다. 특히 최근의 경향은 브라우저 다기능화다. 여러분들도 동영상, 게임, 엑셀 등의 애플리케이션을 브라우저에서 이용해 본 적이 있을 것이다.

한 예로, 브라우저와 통신을 위한 '웹 소켓(web socket)'이라는 기술이 있다. 기존 웹에서는 브라우저가 웹 서버에 요청을 보내서 그것을 받아 처리하는 구조였지만, 이 방식에서는 요청이 없어도 웹 서버가 브라우저에 데이터를 전달(푸시)할 수 있다. 이를 통해 브라우저가 더 많은 것을 실행할 수 있는 형태로 진화되고 있다.

브라우저는 단순히 '인터넷 콘텐츠 표시용 소프트웨어'가 아닌, '다기능(리치(Rich)) 단말'이 되고 있다. 이것은 3계층 아키텍처 상에서 클라이언트–서버형 시스템을 동작시키고 있는 것과 같다. 이처럼 제약이 생긴 발상은 그 제약을 제거함으로써 기존의 방식으로 되돌아가는 경우가 많다. 자주 쓰이는 표현을 빌자면, '역사는 반복된다'라고 할 수 있다.

예를 들어, 집약형에서 분할형으로 패러다임이 이동했지만, 최근 하드웨어 제조사는 역으로 '하드웨어와 소프트웨어 일체형' 제품을 판매하고 있다. 이것은 일단 분할한 것을 요소 기술 그대로 다시 재결합하는 형태다.

반복되고 있는 것처럼 보여도 더 나은 아키텍처를 향한 진화는 계속되고 있다.

1.4 수평 분할형 아키텍처

앞 절에서는 서버별로 다른 역할을 하도록 시스템을 수직으로 확장하는 구조를 소개했다. 하지만 더 높은 확장성을 실현하려면 다른 하나의 축으로 분할하는 것이 필요하다.

여기서 설명하는 '수평 분할형 아키텍처'는 용도가 같은 서버를 늘려나가는 방식이다. 서버 대수가 늘어나면 한 대가 시스템에 주는 영향력이 낮아져서 안정성이 향상

된다. 또한, 처리를 담당하는 서버 대수가 늘어나면 전체적인 성능 향상도 실현할 수 있다.

참고로, 수식 분할형과 수평 분할형은 배타적인 관계가 아니다. 대부분의 시스템이 이 두 가지 방식을 함께 채택하고 있다.

1.4.1 단순 수평 분할형 아키텍처

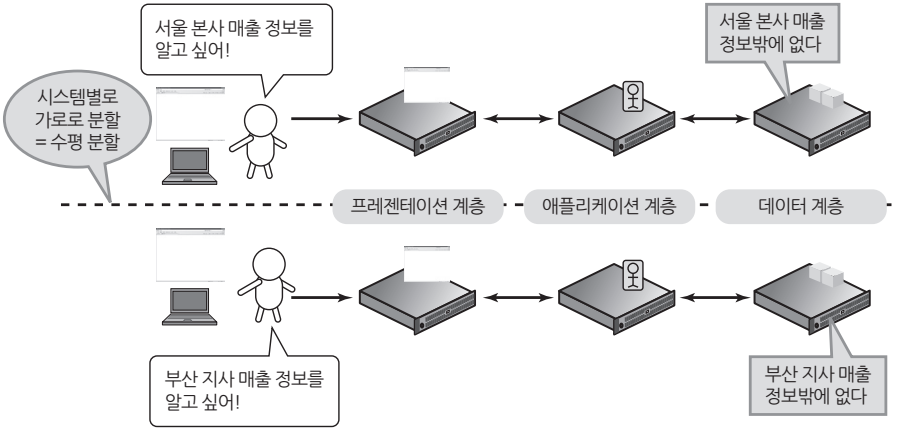


그림 1.8 같은 기능을 가진 복수의 시스템으로 단순 분할한다

그림 1.8과 같은 수평 분할에서는 서울 본사와 부산 지사 시스템이 완전히 분할돼 있다. 서울에서 부산 지사 정보를 알고 싶으면 부산 지사 측 시스템에 접속하면 된다. 수평 분할을 'sharding(샤딩)'이나 'partitioning(파티셔닝)'이라 부르기도 한다.

이 구성에서는 시스템이 둘로 분할됨으로써 시스템 전체 처리 성능을 두 배로 향상시킬 수 있다. 또한, 두 개의 독립된 시스템이 생성되기에 서울 측 시스템에 장애가 발생하더라도 부산 측 시스템에는 전혀 영향을 주지 않아 독립성이 향상된다.

하지만 서울과 부산에서 같은 업무 애플리케이션을 이용하고 있다면, 애플리케이션 업데이트를 양쪽 시스템에 매번 실시해 주어야 한다. 데이터도 서울과 부산이 따로 보유하고 있어서 양쪽 데이터를 동시에(일원화해서) 이용할 수 없다.

또, 서울과 부산의 이용자 수가 비슷하다면 괜찮지만, 이용자의 대부분이 서울 측 시스템을 이용하고 있다면, 서울 측 시스템에 과부하가 걸리고 부산 측 시스템은 놀고 있는 상태가 된다. 이런 상태에서는 시스템 처리 성능이 두 배가 됐다고 말하기 어렵다.

이 구조는 거래상으로 멀리 떨어진 시스템에 자주 이용된다. 또, 공장처럼 각 거점이 완전히 독립된 운영을 하고 있는 경우에도 적합하다. 예를 들어, 많은 사용자가 있는 SNS 웹 서비스에서는 사용자 ID를 기준으로 서버를 분할(sharding)하는 경우가 있다.

장점

- 수평으로 서버를 늘리기 때문에 확장성이 향상된다
- 분할한 시스템이 독립적으로 운영되므로 서로 영향을 주지 않는다

단점

- 데이터를 일원화해서 볼 수 없다
- 애플리케이션 업데이트는 양쪽을 동시에 해 주어야 한다
- 처리량이 균등하게 분할돼 있지 않으면 서버별 처리량에 치우침이 생긴다

1.4.2 공유형 아키텍처

일반 기업 시스템이라면 아주 사이가 나쁘지 않는 한 서울과 부산에서 다른 애플리케이션을 이용하는 경우는 드물다. 공유형에서는 단순 분할형과 달리 일부 계층에서 상호 접속이 이루어진다.

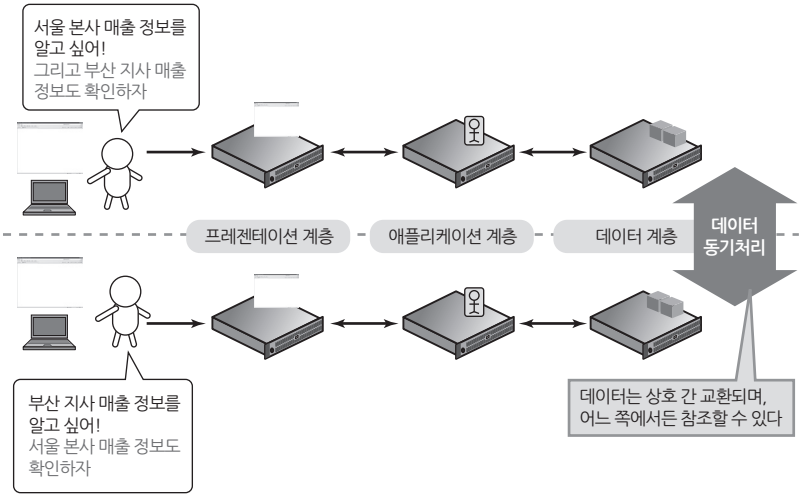


그림 1.9 데이터 계층을 상호 접속한다

그림 1.9는 데이터 계층의 상호 접속 예다. 오라클 제품인 Oracle Database의 클러스터 기능(Real Application Clusters, RAC)에서는 어떤 DB 서버에 접속해도 같은 결과를 얻을 수 있도록 내부적으로 데이터 교환을 하고 있다.

이 구성에서는 서울 측 시스템에서도 필요에 따라 부산 시스템의 데이터에 접속할 수 있다. 물론, 반대도 가능하다. 데이터 계층은 데이터 저장고의 역할도 하기 때문에 보안이 필요한 데이터가 저장되는 경우도 많다. 데이터가 각지에 흩어져 있는 것보다 한곳에서 집중적으로 관리하는 것이 보안상이나 관리상 유리한 경우가 있다. 또한, 이 구성에서는 데이터를 모아서 참조할 수 있다. 예를 들어, 본사의 상품 관리부가 각 지점의 상품 정보를 참조할 수 있다는 장점이 있다.

장점

- 수평으로 서버를 늘리기 때문에 확장성이 향상된다
- 분할한 시스템이 서로 다른 시스템의 데이터를 참조할 수 있다

단점

- 분할한 시스템 간 독립성이 낮아진다
- 공유한 계층의 확장성이 낮아진다

칼럼 가상화 진행 상황

집약형과 분할형의 장단점에 대해 소개했다. 처음에 설명한 것처럼 '궁극의 아키텍처'는 존재하지 않지만, 어느 것이 좋은지에 대한 의문을 가지기 시작했을 것이다.

대형 컴퓨터는 높은 비용이 문제였지만, 지금은 저가 서버도 충분한 성능을 가지고 있기 때문에 시스템 전체를 한 대로 집약해도 별다른 문제가 없을 수 있다. 하지만 서버 한 대에 지나치게 집약하면 확장성 문제가 불거질 수 있다. 또한, 저가 서버는 안정성에 문제가 있을 수 있다.

집약과 분할형 양쪽의 장점을 취하는 접근법이 '가상화'다. 물리 서버를 가상화 기능으로 여러 대의 가상 서버로 분할하는 방식이다. 그림 A에 있는 예처럼, 두 대의 웹 서버와 두 대의 AP 서버, 총 네 대의 가상화 서버를 하나의 물리 서버에서 운영하고 있다. 또한, 물리 서버를 이중화해서 한 대가 망가지더라도 계속해서 운영할 수 있다(이중화에 대해서는 7장에서 상세히 설명하겠다).

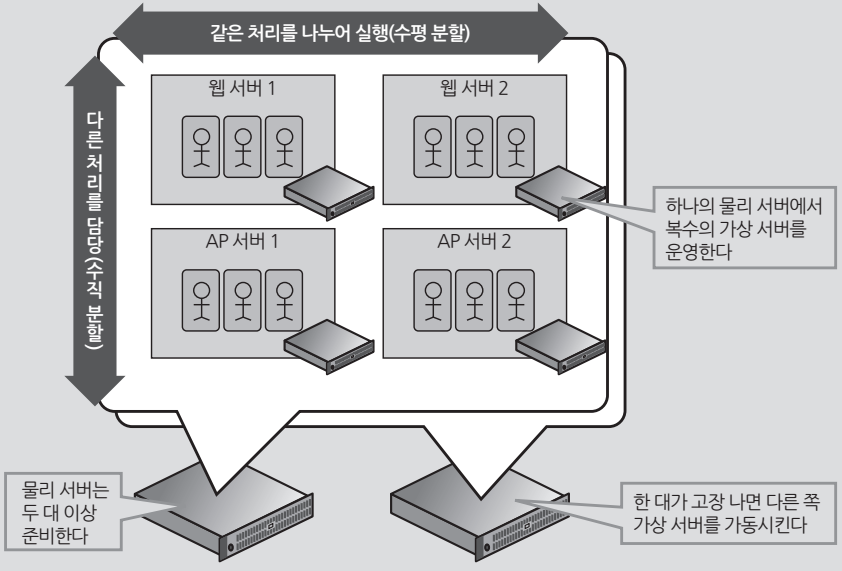


그림 A 가상화 구성

이 구성의 특징은 두 가지다. 먼저, 물리 서버가 두 대이기 때문에 안정성이 높다. 다른 한 가지는 가상 서버를 다른 환경으로 옮기기가 쉬워서 물리 서버 성능이 한계에 이르더라도 확장성을 보장할 수 있다는 것이다.

현재는 대부분의 기업형 업무 시스템이 이런 가상화 구성을 채택하고 있다.

1.5 지리 분할형 아키텍처

지금까지 서버를 수직 또는 수평으로 분할하는 아키텍처에 대해 설명했다. 이들 아키텍처를 조합함으로써 목적에 적합한 구성을 만들 수 있다.

이번 절에서는 업무 연속성 및 시스템 가용성을 높이기 위한 방식으로 지리적으로 분할하는 아키텍처에 대해 알아본다.

1.5.1 스탠바이형 아키텍처

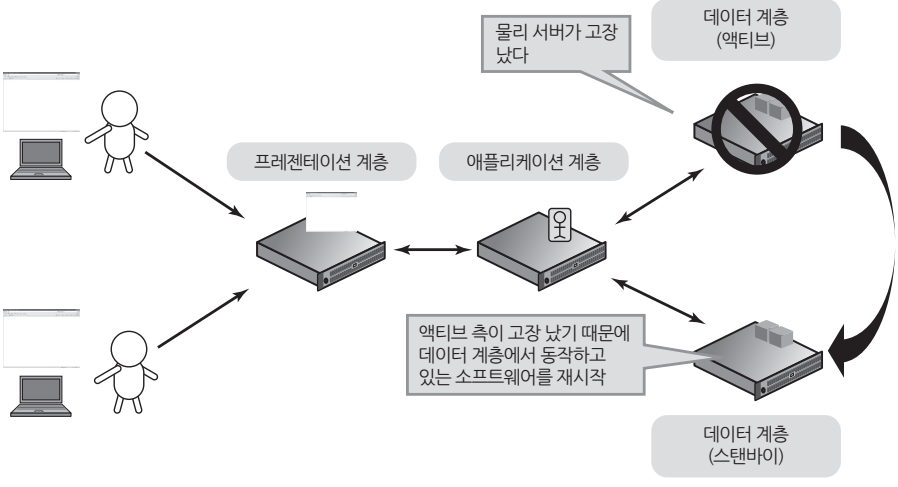


그림 1.10 액티브-스탠바이(active-standby) 구성

그림 1.10은 스탠바이 구성, HA(High Availability) 구성, 액티브-스탠바이 구성 등으로 불리는 형태다. 물리 서버를 최소 두 대를 준비하여 한 대가 고장 나면 가동 중인 소프트웨어를 다른 한 대로 옮겨서 운영하는 방식이다. 이때 소프트웨어 재시작을 자동으로 하는 구조를 '페일오버(failover)'라고 한다. 좀 아는 척하는 사람은 '페일' 또는 'F/O'라고 부르기도 한다(페일오버에 대해서는 7장에서 자세히 다룬다).

이 방식에서는 물리 서버 고장에 대처할 수 있지만, 보통 때는 페일오버 대상 서버(스탠바이)가 놀고 있는 상태가 되기 때문에 리소스 측면에서 낭비가 발생한다. 투자 효과가 낮다면 CIO(Chief Information Officer)에게 꾸중을 들을 수 있다. 이 문제를 해결하기 위해 스탠바이를 따로 두지 않고, 양쪽 서버를 동시에 교차 이용(한쪽이 고장 나면 다른 한쪽이 양쪽을 처리)하는 경우도 많다.

참고로, 물리 서버가 아닌 가상화 서버를 이용하고 있는 경우는 서버 상의 소프트웨어뿐만 아니라 가상 서버별로 다른 물리 서버에 페일오버하는 방식도 선택이 될 수 있다.

1.5.2 재해 대책형 아키텍처

최근 들어 재해 발생이 현실적인 문제가 되고 있다. 인프라 아키텍처에서도 재해에 대응하기 위한 재해 복구(disaster recovery) 구성을 취하는 일이 잦아지고 있다. 예를 들자면, 특정 데이터 센터(사이트)에 있는 상용 환경에 고장이 발생하면 다른 사이트에 있는 재해 대책 환경에서 업무 처리를 재개하는 것을 가리킨다.

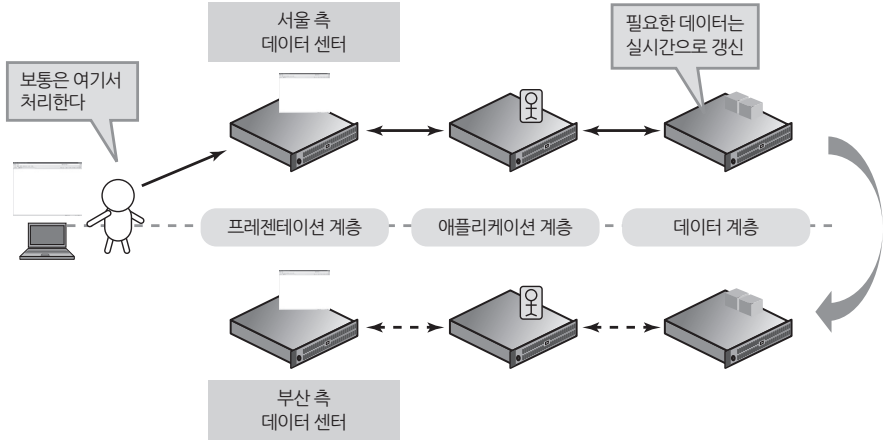


그림 1.11 평상시의 재해 대책 사이트 데이터 반영

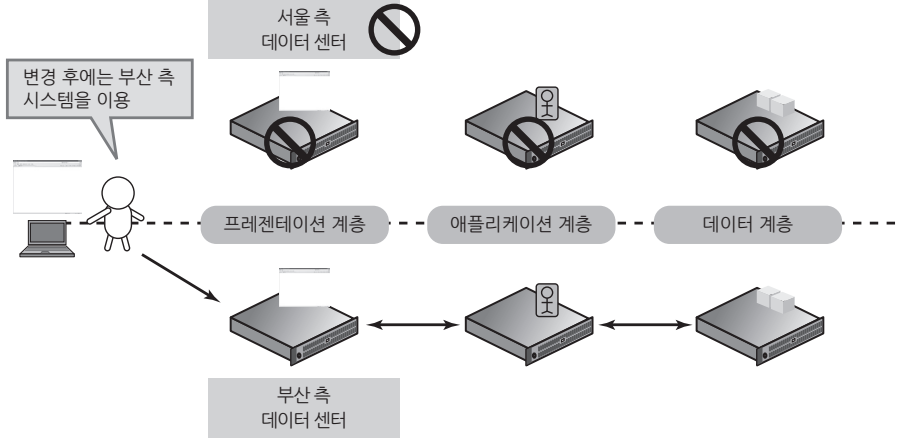


그림 1.12 재해 발생 시의 이용

그림 1.11처럼 서버 장비를 최소 구성 및 동시 구성으로 별도 사이트에 배치하고 소프트웨어도 상용 환경과 동일하게 설정한다. 재해가 발생하면 그림 1.12와 같이 전혀 다른 사이트에 있는 정보를 이용하게 된다.

여기서 문제가 되는 것은 애플리케이션 최신화와 데이터 최신화다. 특히 데이터는

매일 갱신되기 때문에 어느 정도 실시간성을 유지해서 사이트 간 동기 처리를 해야 한다. 저장소 장비 기능, OS 기능, 데이터베이스 기능 등 동기 처리를 위한 방법은 여러 가지가 있다. 각각의 비용, 대상 데이터, 동기 연장 특성 등을 고려해서 결정할 필요가 있다.

1.5.3 클라우드형 아키텍처

클라우드라는 용어는 이미 일반화돼 있어서 독자들도 한 번 정도는 들어 본 적이 있을 것이다. 이 책은 클라우드 해설서가 아니므로 인프라 아키텍처에 해당하는 클라우드만 간단히 설명하겠다.

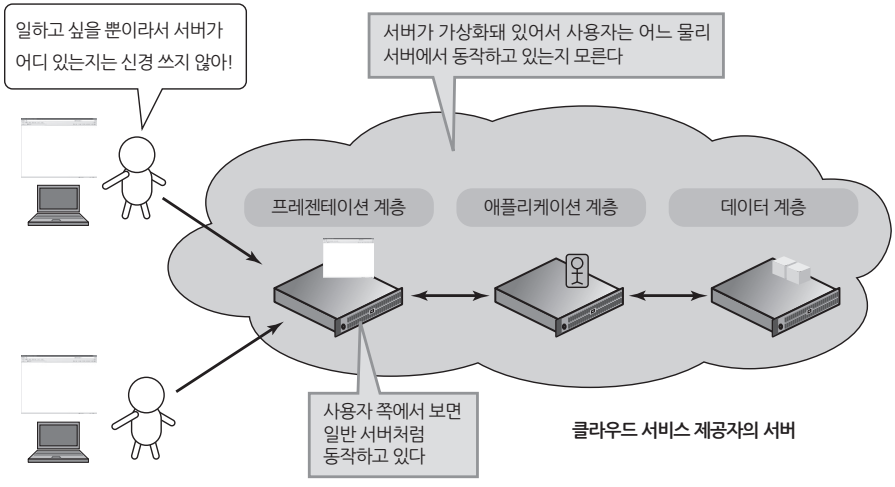


그림 1.13 시스템 전체를 클라우드에 배치하는 경우

그림 1.13이 클라우드형 아키텍처다. 3계층형 시스템의 일부 또는 전부가 클라우드 서비스 제공자가 보유하고 있는 물리 서버에서 동작한다. 대부분의 가상화 기술에서는 서버가 가상적인 존재로 동작한다.

클라우드 세계에서는 어떤 계층을 클라우드 상에 배치하는가에 따라 명칭이 달라진다.

SaaS(Software as a Service) 모델에서는 서버뿐만 아니라 애플리케이션을 포함한 업무 시스템을 클라우드 서비스 회사가 제공하기 때문에 사용자는 인프라를 의식할 필요가 없다. SaaS의 대표적인 예로 'Yahoo! 메일'이나 'Gmail' 등의 웹 메일 서비스가 있다. 여러분들도 이 서비스를 이용할 때 인프라를 의식한 적이 없을 것이다.

한편, PaaS(Platform as a Service), IaaS(Infrastructure as a Service), DBaaS(Database as a Service) 등에서는 일반적인 3계층형 시스템을 구성하는 서버의 일부 혹은 전부를 클라우드 상의 리소스로 대체한다. 이 방식을 이용하면 인프라 설계자는 인프라를 의식해서 그 장점과 단점을 고려한 설계를 해야 한다. 예를 들어, 물리 서버를 구입하는 것에 비해 비용이 낮고 바로 사용할 수 있다는 장점이 있지만, 보안 문제나 클라우드 서비스 제공자와의 네트워크 연장 문제 등이 있을 수 있다.

양쪽의 장점만을 가져와서 사내 특정 부서에서는 '사내 클라우드'를 구성하고, 사내 각 업무 시스템에는 PaaS/IaaS/DBaaS 형태로 제공하는 경우도 있다.

칼럼 **기술은 대물림되고 있다**

하드웨어나 소프트웨어 관련 새로운 기술이 매일같이 등장하고 있지만, 의외로 기본적인 구조는 바뀌지 않고 있다.

예를 들어, 멀티 프로세스 시스템, 가상 기억 시스템, 파일 시스템 같은 기능은 특별히 의식하지 않고 공기처럼 사용하고 있지만, 원래는 범용 기계 시대에 개발된 구조다. 범용 기계의 OS, 상용 유닉스, 리눅스, 윈도우즈 등의 OS는 사용법이나 외형은 다르지만, 그 핵심에는 공통점이 존재해서 과거부터 있었던 기술을 계승하고 있는 듯이 보인다.

현대 컴퓨터는 70년 전에 노이만[2]이 고안한 원리를 기본적으로 계승하고 있다고 볼 수 있다. 나는 옛날 기술서 등을 읽고 기술 근간이나 설계 사상 등을 조사하는 것을 즐긴다. 취미는 아니지만 근본이 되는 설계 사상 등을 알면 시야가 넓어져서 도움이 된다.

2 존 폰 노이만(John von Neumann)은 미국의 과학자다. 대부분의 컴퓨터 아키텍처는 노이만이 고안한 방식을 채택하고 있다고 한다.

CHAPTER

2

서버를 열어 보자

이번 장에서는 하드웨어 장비를 소개하고 그 내부에서 데이터가 어떻게 흐르고 있는지 설명한다.

2.1 물리 서버

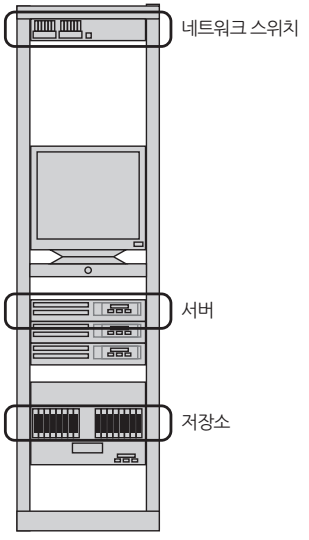
2.1.1 서버 외관과 설치 장소

앞 장에서는 대표적인 시스템 아키텍처를 소개했다. 아키텍처 전체를 구상할 때는 먼저 서버라는 단위로 생각한다는 것을 이해했을 것이다. 이번 장에서는 물리 서버 내부 구조에 대해 더 자세히 살펴보도록 한다.

여러분은 데이터 센터나 서버실에 들어가 본 적이 있는가? 서버가 대량으로 설치돼 있고, 서버에서 나오는 열기를 식히기 위해 실내 온도를 항상 낮게 설정할뿐더러 빛도 들어오지 않는 환경이다. 서버에겐 친절하지만 사람에겐 그렇지 못한 장소로, 그림 2.1과 같다.

그림 2.1 서버실 예(출처: senticus/Shutterstock.com)

서버는 랙(rack)이라는 것에 장착된다. 랙에는 서버 외에도 HDD가 가득 장착돼 있는 저장소나 인터넷 및 LAN을 연결하기 위한 스위치 등도 탑재돼 있다.

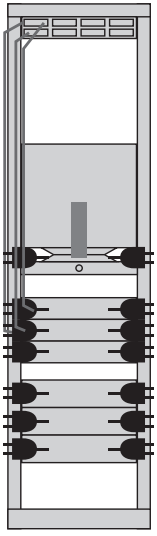


〈랙 구조도〉

그림 2.2 랙 구조

어째서 랙에 서버가 딱 맞게 설치되는지 신기하게 생각한 적이 없는가? 사실은 서버 랙에도 규격이 있는데, 대부분의 랙은 폭이 19인치다. 또, 높이는 한 눈금에 약 4.5cm로 40~46개 눈금으로 이루어져 있다. 이 한 눈금을 1U라고 하며, 서버 높이는 이 단위를 따르고 있다. 이 때문에 2U 서버는 2눈금(높이 약 9cm)의 서버임을 의미한다.

전원이나 네트워크 케이블 배선 등은 모두 랙 뒷면에서 연결된다(그림 2.3).



〈랙 뒷면〉

그림 2.3 랙 뒷면의 모습

서버 설치 시에 중요한 정보는 다음과 같다.

- 서버 크기(U)
- 소비 전력(A)
- 중량(Kg)

다음은 대표적인 서버 아키텍처 중 하나인 인텔의 CPU를 사용한 IA 서버에 대한 설명이다.

먼저, 서버 사진을 보자(그림 2.4). 이것은 오라클이 판매하는 Sun Server X3-2L(구 Sun Fire X4270 M3)라는 모델의 서버 전면 사진이다. 일반적인 서버는 이렇게 옆으로 긴 형태다. 또한, 옆에 랙 마운트 레일(rack mount rail)이라는 것이 있어서 장롱 서랍처럼 설치할 수 있다.

전면에는 HDD나 전원 버튼 등이 있다. HDD는 교체하기 쉽게 되어 있어서 손으로 당겨 꺼낼 수 있다.

그림 2.4 서버 전면

2.1.2 서버 내부 구성

이런 서버는 위쪽 뚜껑을 열 수 있다. 그림 2.5는 Sun Server X3-2L 사진이다.

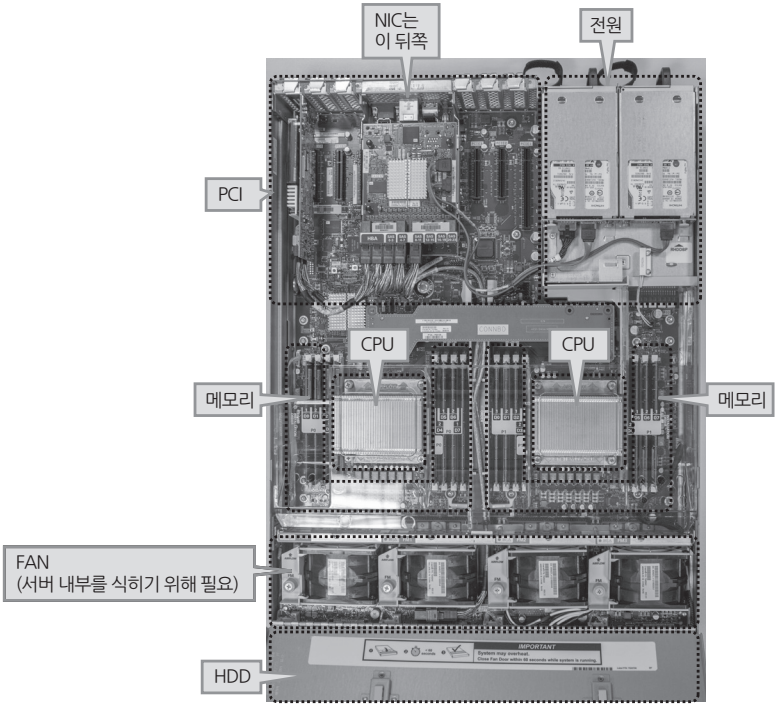


그림 2.5 서버 내부. 컴포넌트는 PC와 동일

PC 부품과 같은 종류가 들어 있는 것을 알 수 있다. 각 부품이 어떻게 연결돼 있는지 그림으로 표현하면 다음과 같다(그림 2.6). 이것은 Sun Fire X4170 M2 구성도다. 이 서버는 Oracle Exadata Database Machine X2-2에 사용된다.[1]

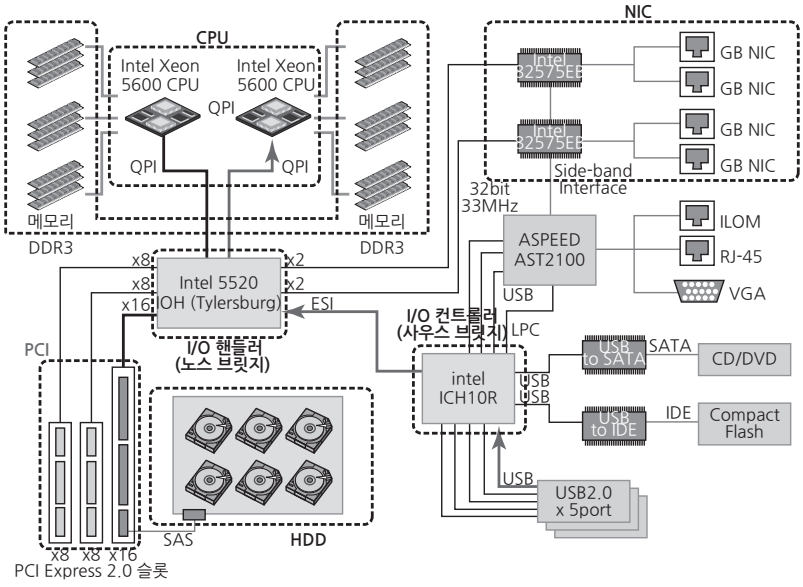


그림 2.6 컴포넌트들은 버스(bus)로 연결된다

CPU, 메모리, HDD 등의 위치 관계를 알 수 있을 것이다. 컴포넌트를 연결하는 선을 '버스(bus)'라고 한다.

그림 왼쪽 상단을 보자. CPU가 두 개 연결돼 있고 그 옆에 메모리가 배치돼 있다. CPU와 메모리는 물리적으로 직접 연결되는 것을 알 수 있다.

왼쪽 하단에는 PCI Express 슬롯이라는 것이 있다. 이것은 외부 장치를 연결하는 곳이다. Sun Fire X4170 M2에서는 1슬롯을 이용해서 HDD(디스크)가 연결돼 있다. 계속해서 오른쪽 상단을 보자. 이 서버에서는 네트워크 인터페이스 네 개를 갖추고

1 이번 장에선 Sun Fire X4170 M20를 이용해서 설명한다. 집필 도중에 최신 제품이 나왔기 때문에 사진은 최신 서버를 실었다.