Hanbit eBook

Realtime 91

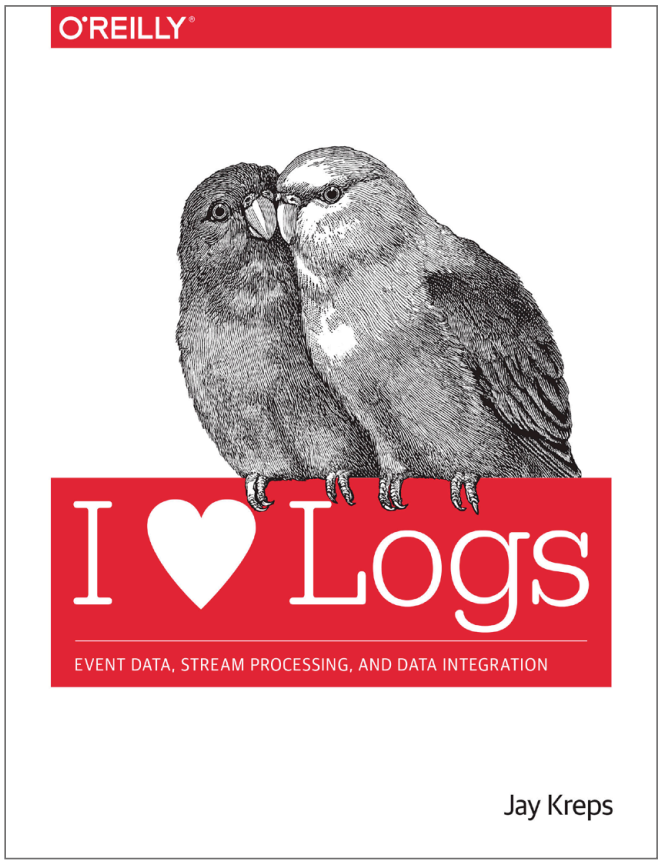
로그를 사랑해

데이터 처리/통합/시스템 구축을 위한 로그

I Heart Logs

제이 크렙스 지음 / 이일웅 옮김

O'REILLY® 한빛미디어 Hanbit Media, Inc.



이 도서는 O'REILLY의
I Heart Logs의
번역서입니다.

로그를 사랑해 데이터 처리/통합/시스템 구축을 위한 로그

초판발행 2015년 01월 30일

지은이 제이 크렙스 / **옮긴이** 이일웅 / **펴낸이** 김태헌
펴낸곳 한빛미디어(주) / **주소** 서울시 마포구 양화로 7길 83 한빛미디어(주) IT출판부
전화 02-325-5544 / **팩스** 02-336-7124
등록 1999년 6월 24일 제10-1779호
ISBN 978-89-6848-730-9 15000 / **정가** 7,000원

총괄 배용석 / **책임편집** 김창수 / **기획·편집** 김상민
디자인 표지 여동일, 내지 스튜디오 [밈], 조판 최송실
영업 김형진, 김진불, 조유미 / **마케팅** 박상용

이 책에 대한 의견이나 오탈자 및 잘못된 내용에 대한 수정 정보는 한빛미디어(주)의 홈페이지나 아래 이메일로 알려주십시오.
한빛미디어 홈페이지 www.hanbit.co.kr / **이메일** ask@hanbit.co.kr

지금 하지 않으면 할 수 없는 일이 있습니다.
책으로 펴내고 싶은 아이디어나 원고를 메일(ebookwriter@hanbit.co.kr)로 보내주세요.
한빛미디어(주)는 여러분의 소중한 경험과 지식을 기다리고 있습니다.

저자 소개

저자_ 제이 크렙스Jay Kreps

링크드인LinkedIn 수석 엔지니어인 제이 크렙스Jay Kreps는 데이터 인프라를 담당하는 아키텍처 팀장이다. 볼드모트Voldemort(분산 키-값 저장소), 카프카Kafka(메시징 시스템), 삼자Samza(스트림 처리 시스템) 등 몇몇 오픈소스 프로젝트의 원 저자다.

- Twitter: @jaykreps

역자 소개

역자_ **이일웅**

10년 넘게 국내 대기업/공공기관 SI 프로젝트에 참여한 자바 웹 개발자이자, 대규모 RESTful 웹 서비스, 플랫폼 간 데이터 연계 및 서버 부하 분산/튜닝 등 다양한 실무 경험이 풍부한 엔지니어다. 두 딸아이에게 좋은 아빠가 되기 위해 언제나 노력하고 있으며 시간이 나면 피아노곡을 연주한다.

『RESTful 자바 패턴과 실전 응용』(에이콘출판사, 2014.12)을 번역했고, 『Jasmin JavaScript Testing』, 『RESTful Java Web Service Security』의 번역을 마치고 출간을 기다리고 있다.

- email: leeilwoong@gmail.com
- homepage: http://www.bullion.pe.kr

역자 서문

정보 기술의 세계에서 데이터 인프라는 모든 시스템의 원천이자 심장부라고 할 수 있습니다. 웹, 모바일 앱, 점점 주목받기 시작한 사물 인터넷IoT 등 겉모습은 다르지만 어떤 IT 서비스도 그 핵심은 예나 지금이나 데이터라는 사실은 큰 변함이 없는 것 같습니다.

그런데 일관된 데이터를 가능한 한 신속하고 안전하게 제공할 책임을 가진 데이터 인프라 분야에서도 그간 많은 새로운 기술들이 출시되고 있습니다. 이 책의 저자인 제이 크렙스는 그러한 최신 기술을 선도하는 데이터 인프라 엔지니어의 일인으로, 분산 시스템에서 로그의 중요성을 설파하고 링크드인에서 근무할 당시 자신이 직접 로그를 응용하여 해결한 실전적인 경험과 노하우에 대해 한 편의 긴 수필 형식으로 작은 책 안에 담아냈습니다.

저자 자신이 직접 개발한 오픈소스를 비롯하여 워낙 다양한 기술과 제품이 언급되는 터라 일일이 참고 링크를 클릭하여 보기조차 벅찬 느낌도 있습니다만, 전문 인프라 엔지니어부터 애플리케이션 개발자에 이르기까지 실무자 여러분들에게 다소 신선한 레퍼런스가 될 것입니다.

또, 실제 성공적으로 구축한 사례만을 좇아 오픈소스 활용을 꺼리는 경향이 있는 국내 IT 분위기를 감안할 때, 기본에 보다 충실한 선진국의 엔지니어가 좌충우돌하며 체험한 수기를 적극적으로 참고할 필요가 있다고 생각합니다. 아니면, 저자의 표현처럼 보잘 것 없는humble 로그를 가지고 데이터 인프라 구축에 어떻게 활용하는지, 외국에서는 어떤 최신 기술을 어떻게 써서 문제를 해결했는지 동향 파악 정도로 가볍게 읽어 내려가도 좋을 듯싶습니다.

훌륭한 도서의 번역을 맡겨주신 한빛미디어 김창수 팀장님과 김상민 대리님 그리고 애써 짬을 내어 번역하느라 퇴근하고 집에 와서도 함께 해주지 못한, 사랑하는 제 아내와 두 딸, 제이, 솔이에게 고마움을 전합니다. 언제나 변함없는 믿음과 사랑으로 지켜봐 주신 부모님께도 진심으로 감사의 말씀 올립니다.

2015년 희망으로 새해를 시작하며

이일웅

한빛 eBook 리얼타임

한빛 eBook 리얼타임은 IT 개발자를 위한 eBook입니다.

요즘 IT 업계에는 하루가 멀다 하고 수많은 기술이 나타나고 사라져 갑니다. 인터넷을 아무리 뒤져도 조금이나마 정리된 정보를 찾는 것도 쉽지 않습니다. 또한 잘 정리되어 책으로 나오기까지는 오랜 시간이 걸립니다. 어떻게 하면 조금이라도 더 유용한 정보를 빠르게 얻을 수 있을까요? 어떻게 하면 남보다 조금 더 빨리 경험하고 습득한 지식을 공유하고 발전시켜 나갈 수 있을까요? 세상에는 수많은 종이책이 있습니다. 그리고 그 종이책을 그대로 옮긴 전자책도 많습니다. 전자책에는 전자책에 적합한 콘텐츠와 전자책의 특성을 살린 형식이 있다고 생각합니다.

한빛이 지금 생각하고 추구하는, 개발자를 위한 리얼타임 전자책은 이렇습니다.

1. **eBook Only - 빠르게 변화하는 IT 기술에 대해 핵심적인 정보를 신속하게 제공합니다.**

   500페이지 가까운 분량의 잘 정리된 도서(종이책)가 아니라, 핵심적인 내용을 빠르게 전달하기 위해 조금은 거칠지만 100페이지 내외의 전자책 전용으로 개발한 서비스입니다. 독자에게는 새로운 정보를 빨리 얻을 수 있는 기회가 되고, 자신이 먼저 경험한 지식과 정보를 책으로 펴내고 싶지만 너무 바빠서 엄두를 못 내는 선배, 전문가, 고수 분에게는 보다 쉽게 집필할 수 있는 기회가 될 수 있으리라 생각합니다. 또한 새로운 정보와 지식을 빠르게 전달하기 위해 O'Reilly의 전자책 번역 서비스도 하고 있습니다.

2. **무료로 업데이트되는, 전자책 전용 서비스입니다.**

   종이책으로는 기술의 변화 속도를 따라잡기가 쉽지 않습니다. 책이 일정 분량 이상으로 집필되고 정리되어 나오는 동안 기술은 이미 변해 있습니다. 전자책으로 출간된 이후에도 버전 업을 통해 중요한 기술적 변화가 있거나 저자(역자)와 독자가 소통하면서 보완하여 발전된 노하우가 정리되면 구매하신 분께 무료로 업데이트해 드립니다.

3. 독자의 편의를 위하여 DRM-Free로 제공합니다.

구매한 전자책을 다양한 IT 기기에서 자유롭게 활용할 수 있도록 DRM-Free PDF 포맷으로 제공합니다. 이는 독자 여러분과 한빛이 생각하고 추구하는 전자책을 만들어 나가기 위해 독자 여러분이 언제 어디서 어떤 기기를 사용하더라도 편리하게 전자책을 볼 수 있도록 하기 위함입니다.

4. 전자책 환경을 고려한 최적의 형태와 디자인에 담고자 노력했습니다.

종이책을 그대로 옮겨 놓아 가독성이 떨어지고 읽기 힘든 전자책이 아니라, 전자책의 환경에 가능한 한 최적화하여 쾌적한 경험을 드리고자 합니다. 링크 등의 기능을 적극적으로 이용할 수 있음은 물론이고 글자 크기나 행간, 여백 등을 전자책에 가장 최적화된 형태로 새롭게 디자인하였습니다.

앞으로도 독자 여러분의 충고에 귀 기울이며 지속해서 발전시켜 나가도록 하겠습니다.

차례 01

1 | 들어가며

이 책의 주제는 로그다. 수고스럽게 로그에 대한 책까지 쓰게 된 이유는 뭘까? 로그는 노시퀄NoSQL 데이터베이스에서 암호화폐cryptocurrency[01]에 이르기까지 다양한 시스템의 핵심적인 추상체abstraction다. 그런데 대부분의 엔지니어는 가끔 로그 파일을 테일링tailing 해보면서도 막상 로그 자체에 대해서는 별생각이 없다. 그래서 먼저 분산 시스템에서 로그의 작동 원리를 개략적으로 설명하고, 이러한 개념을 구체화하여 데이터 통합, 엔터프라이즈 아키텍처, 실시간 데이터 처리, 데이터 시스템 설계 등 다양한 용도로 활용되는 로그의 세계로 여러분을 안내하고자 한다. 또 링크드인LinkedIn에서 데이터 인프라 시스템 구축 업무를 수행하면서 터득한 지혜와 경험도 공유할 것이다. 우선 여러분이 이미 알고 있다고 생각하는 것부터 시작해보자.

1.1 로그란 무엇인가

많은 사람이 로그하면 [그림 1-1]과 같은 모습을 떠올릴 것이다.

프로그래머라면 대체로 이런 로그에 익숙하다. 텍스트 파일이 계속 순환되면서 차례로 쌓이는, 느슨한 구조의 요청, 에러 등 잡다한 메시지 더미 말이다.

내가 이 책에서 다루려는 로그는 프로그래밍적인 접근을 목적으로 한다는 점에서 이렇게 사람이 읽을 수 있는 퇴화한degenerative 형태의 로그와는 완전히 다르다.

사실 조금만 생각해봐도 기계에 쌓인 로그를 사람이 읽는다는 발상 자체가 시대착오적이라는 걸 알 수 있다. 서비스 가짓수가 늘어나고 서버 대수가 늘어나면 머지않아 관리하기 어려워질 테니 말이다. 로그는 운용 중인 장비들의 특성을 이해하기 위한 쿼리와 그래프의 입력값으로 쓰이는 것이 목적이다. 로마자 알파벳만 가득 찬

01 역자주_비트 코인과 같이 암호화 기술이 적용된 새로운 온라인 거래 수단을 말한다.

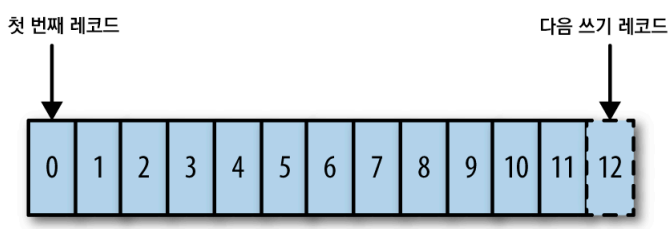
텍스트 파일은 곧 살펴볼 구조화 로그structured log로서 적합하지 않다.

[그림 1-1] 아파치 서버 로그 일부

```
jkreps-mn:~ jkreps$ tail -f -n 20 /var/log/apache2/access_log
::1 - - [23/Mar/2014:15:07:00 -0700] "GET /images/apache_feather.gif HTTP/1.1" 200 4128
::1 - - [23/Mar/2014:15:07:04 -0700] "GET /images/producer_consumer.png HTTP/1.1" 200 8(
::1 - - [23/Mar/2014:15:07:04 -0700] "GET /images/log_anatomy.png HTTP/1.1" 200 19579
::1 - - [23/Mar/2014:15:07:04 -0700] "GET /images/consumer-groups.png HTTP/1.1" 200 268;
::1 - - [23/Mar/2014:15:07:04 -0700] "GET /images/log_compaction.png HTTP/1.1" 200 4141(
::1 - - [23/Mar/2014:15:07:04 -0700] "GET /documentation.html HTTP/1.1" 200 189893
::1 - - [23/Mar/2014:15:07:04 -0700] "GET /images/log_cleaner_anatomy.png HTTP/1.1" 200
::1 - - [23/Mar/2014:15:07:04 -0700] "GET /images/kafka_log.png HTTP/1.1" 200 134321
::1 - - [23/Mar/2014:15:07:04 -0700] "GET /images/mirror-maker.png HTTP/1.1" 200 17054
::1 - - [23/Mar/2014:15:08:07 -0700] "GET /documentation.html HTTP/1.1" 200 189937
::1 - - [23/Mar/2014:15:08:07 -0700] "GET /styles.css HTTP/1.1" 304 -
::1 - - [23/Mar/2014:15:08:07 -0700] "GET /images/kafka_logo.png HTTP/1.1" 304 -
::1 - - [23/Mar/2014:15:08:07 -0700] "GET /images/producer_consumer.png HTTP/1.1" 304 -
::1 - - [23/Mar/2014:15:08:07 -0700] "GET /images/log_anatomy.png HTTP/1.1" 304 -
::1 - - [23/Mar/2014:15:08:07 -0700] "GET /images/consumer-groups.png HTTP/1.1" 304 -
::1 - - [23/Mar/2014:15:08:07 -0700] "GET /images/log_cleaner_anatomy.png HTTP/1.1" 304
::1 - - [23/Mar/2014:15:08:07 -0700] "GET /images/log_compaction.png HTTP/1.1" 304 -
::1 - - [23/Mar/2014:15:08:07 -0700] "GET /images/kafka_log.png HTTP/1.1" 304 -
::1 - - [23/Mar/2014:15:08:07 -0700] "GET /images/mirror-maker.png HTTP/1.1" 304 -
::1 - - [23/Mar/2014:15:09:55 -0700] "GET /documentation.html HTTP/1.1" 200 195264
```

이 책에서 이야기할 로그는 데이터베이스나 시스템 분야에서 커밋 로그commit log, 또는 저널journal이라 부르는 것과 좀 더 가깝고 일반적인 개념인데, [그림 1-2]에 표현한 것처럼 시간순으로 덧붙이기만 가능한append-only 레코드 시퀀스record sequence다.

[그림 1-2] 구조화 로그(0부터 시작하여 쓰인 순서대로 채번됨)



사각형 하나하나가 로그에 추가된 레코드를 나타내며, 레코드는 추가된 순서대로 저장되므로 왼쪽에서 오른쪽으로 읽힌다. 각 엔트리는 차례대로 로그 엔트리 번호log entry number가 채번되어 고유키로 된다. 레코드 내부의 내용이나 형식은 중요하

지 않다. JSON blob 같은 데이터가 쌓인다고 하면 구체적인 모습이 연상될 것이다. 물론, 다른 포맷의 데이터도 상관없다.

왼쪽 엔트리가 오른쪽 엔트리보다 먼저 생성된 것이므로 레코드의 순서는 곧 "시간"이고, 로그 엔트리 번호는 엔트리의 "타임스탬프timestamp"라고 할 수 있다. 레코드의 배열 순서가 곧 시간이라는 말이 처음엔 어색할 수 있겠지만, 특정한 물리적인 시간 관념과는 완전히 별개의 속성을 갖게 된다. 바로 이러한 속성이 분산 시스템에서 아주 중요한 핵심이 된다.

로그는 파일, 테이블과 크게 다르지 않다. 파일은 바이트의 배열이고 테이블은 레코드의 배열이며, 로그는 레코드가 시간순으로 정렬된 파일이나 레코드 중 하나다.

[그림 1-1]의 아파치 로그에서 모두 끝 부분에 추가되는 레코드 시퀀스라는 공통점이 있음을 알 수 있다. 하지만 여기서, 단순한 텍스트 파일이 아닌, 추상화된 데이터 구조로서 로그를 바라보는 것이 중요하다.

이쯤 되면 혹자는 "그렇게 간단한 걸 뭐하러 얘기하고 있는 거지?"라고 의문을 품을지 모르겠다. 덧붙이기만 가능한 레코드 시퀀스가 데이터 시스템과 무슨 상관이란 말인가? 정답은, "로그는 언제 무슨 일이 일어났는지 기록하는, 특정한 목적을 가지고 있다"는 것이다. 바로 이것이 여러 면에서 분산 데이터 시스템의 핵심이 된다.

1.2 데이터베이스에서의 로그

로그가 탄생하게 된 기원은 나도 잘 모르겠다. 이진 검색binary search을 처음 발명한 사람이 스스로 너무 간단한 알고리즘이라 발명품이라는 생각조차 하지 않았던 것과 비슷할 것 같다. 거슬러 올라가면 IBM System R[02] 시절에도 로그는 존재했다.

02 http://bit.ly/system-r

당시에 로그는 데이터베이스에서 장애가 발생하면 데이터 구조와 인덱스를 동기화시킬 용도로 사용했었다. 이를 좀 더 작은 단위로 세분화하고 오랫동안 보존할 목적으로, 데이터 구조에 어떤 변경을 하기 직전 변경될 레코드의 이력을 남겨두기 위해 데이터베이스에서 로그를 활용하기 시작했다. 로그는 발생한 일을 기록한 레코드고, 각 테이블이나 인덱스는 이러한 이력을 유용한 방향으로 투영시킨 결과물이다. 로그는 계속 보존되는 데이터이므로 시스템 장애가 발생할 때 다른 영속적인 persistent 데이터 구조를 복구하는, 가장 확실한 소스가 된다.

이후 로그는 데이터베이스의 ACID 속성(원자성atomicity, 일관성consistency, 고립성isolation, 지속성durability)을 구현하기 위한 용도에서 데이터베이스 간 데이터를 복제하는 도구로 사용 범위가 확대되었다. 데이터베이스에서 일어난 일들을 차례대로 기록한 시퀀스가 원격 사본 데이터베이스replica database와 동기화하기 위한 용도로 제격이었던 것이다. 오라클Oracle, 마이시퀄MySQL, 포스트그레SQLPostgreSQL, 몽고DBMongoDB에는 슬레이브slave 데이터베이스로 로그 덩어리를 전송하는 전용 프로토콜이 탑재되어 있다. 슬레이브 데이터베이스는 로그 데이터를 받아 자신의 로컬 데이터 구조를 마스터 데이터베이스와 일치시킨다. 오라클은 로그를 일반적인 데이터 구독 장치로 상품화시켜 자사 제품인 엑스스트림XStreams[03]이나 골든게이트GoldenGate[04]로 비오라클non-Oracle 시스템에서도 로그를 구독할 수 있게 하였고, 마이시퀄과 포스트그레SQL에도 이와 유사한 기능이 있다.

이 책의 전반에 걸쳐 로그의 용도는 크게 다음 두 가지로 분류할 수 있다.

1. 다른 사본으로 데이터를 전송하기 위한 발행자publisher/구독자subscriber 장치
2. 여러 개의 사본에 순서대로 업데이트를 적용하기 위한 데이터 동기화 장치

이처럼 로그가 처음 사용된 분야가 데이터베이스 내부인 관계로, 기계 판독 로그

03 http://bit.ly/xstreams
04 http://bit.ly/g-gate

machine readable log라는 개념이 널리 알려지지 않은 것 같다. 하지만 로그가 다양한 유형의 메시징이나 데이터 흐름, 실시간 데이터 처리를 가능케 한 이상적인 추상체다.

1.3 분산 시스템에서의 로그

데이터베이스에서 로그로 해결한 문제들(사본에 데이터를 분배하고 업데이트 순서를 맞추는)은 역시 분산 시스템에서도 가장 근본적인 문제 중 일부다.

분산 시스템에서 로그 중심적log-centric인 접근 방법은 기계 복제 원리machine replication principle라는 단순한 이론에서 비롯되었다.

> *2개의 똑같은 확정적*deterministic *프로세스가 똑같은 상태에서 시작하여 똑같은 입력을 똑같은 순서대로 받는다면, 똑같은 결과를 산출하고 똑같은 상태로 종료될 것이다.*

약간 생뚱맞은 소리처럼 들리겠지만, 차근차근 의미를 살펴보자.

'확정적'이라는 말은 프로세스가 타이밍에 의존하지 않고, 다른 대역 외out-of-band 입력이 결과에 영향을 미치지 않는 것을 뜻한다. 가령, 스레드thread의 실행 순서에 따라 결과가 바뀌는 멀티스레드multithread 프로그램이나 `gettimeofday()` 메서드의 호출 결과에 따라 실행 흐름이 분기되는 프로그램, 그 밖에 입력 소스가 비반복적non-repeatable인 경우는 역으로 비확정적non-deterministic이라 할 수 있다. 물론 100% 확정적이냐 하는 문제는 물리학의 근본까지 파고들어야 하겠지만, 이 책에서는 결과를 적절한 수학 함수로 나타낼 정도로 상태와 입력을 알고 있지는 못한다 해도 상관없다.

프로세스의 상태state란 처리 후 기계, 즉 메모리나 디스크에 남아있는 데이터를 가리킨다.

'똑같은 입력을 똑같은 순서대로 받는다'는 구절에서 뭔가 떠오르지 않는가? 바로 로그가 하는 일이다.

기계 복제 원리는 사실 아주 직관적이다. 2개의 확정적 코드에 똑같은 로그를 입력하면 똑같은 순서대로 똑같은 결과가 산출된다는 것이다.

이 원리를 분산 컴퓨팅에 어떻게 응용할지는 너무 뻔하다. 다수의 기계가 똑같은 일을 하도록 각 프로세스에 똑같은 로그를 입력하는 식으로 구현하는 것이다. 로그의 역할은 자신을 입력받아 처리할 각 사본이 동기화될 수 있도록 입력 스트림에서 모든 비확정성nondeterminism을 짜내는 것이다.

여기까지 이해가 되었다면 이 원리에 더는 심오하고 복잡한 건 없다. 한 마디로, "확정적인 처리는 확정적이다deterministic processing is deterministic"는 것이다. 그렇긴 하지만 이 한 마디가 분산 시스템을 설계하는 기본 토대가 되었다.

지금도 이 원리는 그대로다. 오래된 분산 컴퓨팅 시스템에서 전통적인 접근 방식을 답습한다 해도 마찬가지다.

그러나 이렇게 기본적인 설계 패턴에 내재한 의미가 아직도 제대로 받아들여지지 못하는 것 같고, 엔터프라이즈 아키텍처 기반의 애플리케이션에서는 더더욱 간과되는 경향이 있다.

이산적인discrete 로그 엔트리 번호가 사본의 상태를 나타내는 시계가 된다는 게 정말 멋지지 않은가? 번호 하나로 모든 사본의 상태를 표시할 수 있고, 해당 사본에서 가장 마지막으로 처리한 로그 엔트리의 타임스탬프가 되는 것이다. 서로 다른 사본이 같은 시각에 똑같은 상태로 맞춰지는 것이다. 따라서 로그와 결합한 타임스탬프로 사본의 전체 상태도 포착할 수 있다. 이렇게 기계의 로컬 시간이 아닌, 이산적이고discrete, 이벤트 구동event-driven적인 시간을 기준으로 다른 기계 간의 상태를 쉽게 비교할 수 있다.

1.3.1 로그 중심 설계의 다양성

기계 복제 원리는 로그에 기록할 대상에 따라 다양한 방법으로 응용할 수 있다. 예컨대, 서버 요청을 로깅하여 각 복제 프로세스가 독립적으로 처리하게 하거나, 처리 인스턴스를 하나 띄워두고 서비스 응답 과정에서 변경된 상태를 로깅한다. 이론적으로는 x86 기계어 명령이나 호출한 메서드명, 파라미터까지도 로깅해 두었다가 각 사본에서 실행할 수도 있다. 프로세스마다 입력을 처리하는 방법이 같다면, 모든 사본에서 실행 결과는 같을 것이다.

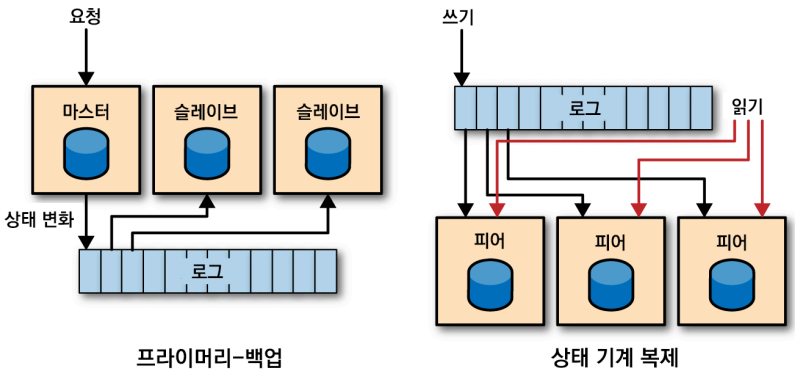
유사한 패턴을 두고도 업종에 따라 기술하는 방식은 조금씩 다르다. 데이터베이스 전문가들은 대개 물리적 로깅physical logging과 논리적 로깅logical logging을 구분한다. 테이블의 로우 단위row-based로 변경 내용을 기록하는 것이 물리적 로깅이라면, 논리적인 로깅은 변경된 로우 뿐만 아니라 변경을 일으킨 SQL 명령문(insert, update, delete)까지 로그에 포함하는 구문statement 로깅이다.

분산 시스템을 하는 사람들은 처리와 복제에 관한 접근 방식을 크게 두 가지로 구분한다. '상태 기계 복제 모델state machine replication model'은 활성-활성active-active 모델을 가리키며, 서버 요청을 로깅 후 각 복제 프로세스가 로깅된 순서대로 처리하는 방식이다. 이를 약간 변형시킨 것이 프라이머리-백업 모델primary-backup model로, 복제 프로세스 중 하나를 리더로 지정한다. 리더 프로세스가 접수 순서대로 요청을 처리하고 그 결과 발생한 상태 변화를 로깅하면, 다른 복제 프로세스들은 리더가 로깅한 상태를 스스로 적용하여 동기화하고, 리더가 잘못될 경우 언제라도 그 자리를 대신할 준비를 한다.

[그림 1-3] 프라이머리-백업 모델에서는 마스터로 선정된 노드가 모든 읽기/쓰기를 담당한다. 쓰기 처리는 로그에 남게 되고, 슬레이브는 이를 구독해서 마스터가 실행한 결과를 자신에게도 적용한다. 마스터 노드에 문제가 생기면 자동으로 슬레이브 중 하나가 새로운 마스터로 등극한다. 반면, 상태 기계 복제 모델에서는 모든

노드가 피어peer다. 로그에 뭔가 쓰이면 모든 노드가 쓰인 순서대로 각자 자기 자신을 업데이트한다.

[그림 1-3] 프라이머리-백업 모델에서 마스터 노드가 선정되어 모든 읽기/쓰기를 처리한다. 상태 기계 복제 모델에서는 모든 노드가 피어로 동작한다.



프라이머리-백업 상태 기계 복제

1.3.2 예제

간단한 예제를 통해 로그를 이용한 시스템 구축 방법을 비교해보자. 자, 초깃값이 0인 변수들이 있고 여기에 사칙연산 및 각 변숫값을 개별적으로 조회할 수 있는 계산기 서비스가 있다고 하자. 이 서비스에서 사용할 수 있는 명령어는 다음과 같다.

```
x? // x의 현재값을 조회한다
x+=5 // x에 5를 더한다
x-=2 // x에서 2를 뺀다
y*=2 // y에 2를 곱한다
```

계산기 서비스는 HTTP 프로토콜로 요청/응답이 이루어지는 원격 웹 서비스라고 가정하자.

서버가 하나뿐이라면 구현은 아주 간단하다. 어떤 순서로 요청이 들어오든지 변수

를 메모리나 디스크에 보관하고 값을 바꿔주기만 하면 된다. 하지만 서버가 한 대밖에 없으므로 무정지성fault tolerance이 떨어지고 서비스를 스케일아웃scale-out[05]할 방법이 없다(계산기 서비스가 엄청난 인기몰이 중이라고 해두자).

결국, 서버를 증설하고 상태와 처리 로직을 복제하는 것이 최선이다. 그런데 문제는 서버 간 동기화가 제대로 되지 않을 수 있다는 점이다. 이를테면 서버마다 다른 순서로 업데이트 명령이 수신된다든지(모든 연산이 가환적commutative[06]인 것은 아니므로), 아예 수신 자체가 되지 않거나 서버 응답이 없는 등 여러 가지 상황이 발생할 수 있다.

실제로 많은 사람이 쿼리와 업데이트를 원격 데이터베이스에 푸시push하여 문제를 해결하려고 할 것이다. 하지만 이렇게 한다고 문제가 다 해결되는 것은 아니며, 결국 데이터베이스의 무정지성이라는 새로운 문제에 봉착하고 말 것이다. 따라서 우리의 계산기 서비스에서는 일단 애플리케이션에서 로그를 사용하는 문제에만 집중하자.

로그를 이용한 해법으로 몇 가지가 있다. 상태 기계 복제 모델에서는 먼저 실행할 연산을 로그에 남기고 사본들은 순서대로 해당 연산을 반영한다. 그 결과, "x+=5", "y*=2" 같은 명령문들이 로그에 남게 될 것이다.

프라이머리-백업 모델에서는 먼저 프라이머리(리더/마스터) 사본을 선정하고 요청 순서대로 프라이머리가 명령어를 처리하면서 변경된 변숫값을 로깅한다. 따라서 로그에는 원래의 명령어가 아닌, "x=1"이나 "y=6" 같은 결괏값만이 남게 된다. 나머지 사본들은 백업(팔로워follower/슬레이브) 역할을 하는데, 프라이머리가 남긴 로그

05 역자주_ 처리 능력을 향상시키는 방법은 크게 스케일 업(scale-up)과 스케일 아웃(scale-out), 두 가지로 나눌 수 있는데, 스케일 업은 기존 서버를 하드웨어적으로 증강시키는 것을, 스케일 아웃은 접속 서버의 대수를 늘리는 것을 말한다. 분산 환경에서는 보통 스케일 아웃 방식으로 전체 처리 능력을 업그레이드한다.

06 역자주_ 연산의 순서를 바꾸어도 결과가 달라지지 않는 성질을 말한다. 실수 a, b에 대해 ab = ba이므로 곱셈 연산에 대해 가환적이지만, 행렬 A, B에 대해 AB = BA는 항상 성립하는 것은 아니므로 가환적이지 않다.

를 구독하여 자신의 로컬 저장소에 적용한다. 프라이머리가 잘못되면 백업 사본 중 하나가 프라이머리로 선출된다.

이 예제를 통해서 사본 간 데이터의 일관성을 유지함에 정렬 순서가 중요한 이유를 알아보았다. 덧셈과 뺄셈 명령어의 순서가 바뀌는 것이나 같은 변수에 대한 업데이트 순서가 바뀌는 것이나 결과가 달라지기는 매한가지다.

1.3.3 로그와 일치화

분산 로그는 일치화consensus 문제를 다루기 위한 데이터 구조다. 로그는 결국 다음에 추가될 값에 대한 일련의 결정 프로세스라고 표현할 수 있다. 로그를 생성하는 것이 알고리즘의 주된 응용 분야이긴 하지만, 팍소스Paxos 계열 알고리즘의 로그를 들여다보기 바란다. 팍소스에서는 일련의 일치화 문제로 로그를 모델링한 "멀티 팍소스multi-paxos"라는 프로토콜로 확장하여, 로그의 한 슬롯당 한 번씩 일치화를 수행한다. ZAB, RAFT, 뷰스탬프 복제Viewstamped Replication[07] 등은 동기화 로그를 분산시켜 보관하는 문제를 직접 다룬 프로토콜로서 로그는 훨씬 더 중요하다.

아마도 최근 2, 30년 동안 분산 컴퓨팅 이론이 실제 구현된 애플리케이션의 흐름을 따라가지 못한 이유로 우리가 바라보는 시각이 다소 편향된 것이 아닌가 싶다. 사실 일치화는 단순한 문제다. 컴퓨터 시스템에서 하나의 값만을 결정할 일이 드물고, 항상 시퀀스로 요청을 받아 처리하기 때문이다. 따라서 로그는 간단한 단일값 레지스터single-value register와는 달리, 더욱 자연 발생적인 추상체다.

알고리즘에 너무 골몰하다 보면 시스템에 정작 필요한 로그의 추상화 개념이 모호해질 수 있으므로 내부적인 구현이야 어떻든 상품화된 구현 재료라고 알고 로그에 대해서는 더 이상 파고들지는 말자. 우리가 해시 테이블을 이야기하면서 선형 탐사linear probing 같은 알고리즘들에 대해 논하지 않는 것과 같은 맥락이다. 로그는 그간

07 http://bit.ly/viewstamped

많은 사람이 수많은 알고리즘과 구현 방법을 동원하여 찾아낸, 가장 확실하고 성능이 뛰어난 인터페이스 상품이라고 하면 되겠다.

1.4 변경로그 101: 테이블과 이벤트는 동전의 양면과 같다

잠시 데이터베이스로 다시 돌아가자. 변경 로그와 테이블, 둘 사이에는 동전의 앞뒷면과 같은 흥미로운 양면성duality이 있다. 로그가 은행 시스템의 차/대변 계정 목록이라고 한다면, 테이블은 현재 계좌 잔액에 비유된다. 변경 로그만 있으면 테이블을 생성하고 현재 상태를 캡처하기 위해 변경 내용을 적용해볼 수 있다. 테이블에는 각 키(로깅 시각)별로 최종 상태가 기록된다. 원본 테이블을 생성할 뿐만 아니라, 이를 변형해서 어떤 종류의 테이블이라도 만들어낼 수 있기에 이런 관점에서 로그가 더 근본적인 데이터 구조라고 할 수 있다(그렇다, 비관계형 사고방식을 가진 사람들에게 테이블은 키별로 적재된 데이터 저장소일 뿐이겠지만).

이 프로세스는 역으로도 실행할 수 있다. 테이블을 업데이트하는 동안 기록된 변경 로그만 있으면 해당 테이블의 모든 변경된 내용을 상태에 반영할 수 있다. 거의 실시간에 가깝게 복제할 수 있다. 이런 점에서 테이블과 이벤트는 서로 양면적이다. 테이블은 정지된 상태의 데이터를 담아두고, 로그는 변경된 내용을 포착한다. 로그만 갖고 있으면 테이블의 최종 데이터는 물론이고 과거의 어느 버전의 데이터라도 되살릴 수 있다. 실로 테이블의 모든 과거를 복원시키는 백업 마술이다.

아마도 소스 코드의 버전 관리 시스템을 떠올리는 독자들도 있을 것이다. 실은 데이터베이스와 버전 관리 시스템 사이에는 밀접한 관계가 있다. 버전 관리 시스템에서 처리하는 문제가 분산 데이터 시스템에서 동시 다발적으로 발생하는 상태 변화를 분산시켜 유지하는 문제와 비슷하다. 버전 관리 시스템이 처리하는 연속적인 패치가 로그라면, 현재 체크 아웃된 소스 코드의 스냅샷snapshot은 테이블에 해당된다. 여타상태 유지stateful 분산 시스템과 마찬가지로, 버전 관리 시스템은 로그를 통

해 복제한다(PC에서 업데이트하면 서버에서 패치를 끌어와 로컬 스냅샷에 반영한다).

지금까지 언급한 내용 일부가 최근에 출시된 데이토믹Datomic[08]이라는, 로그 중심적인 데이터베이스[09]에 담겨있다는 사실을 알고 있는 독자들이 있을 텐데, 데이토믹은 벌써 10년 넘게 분산 시스템 및 데이터베이스 일부로 운용되어 온 시스템이므로 그들만의 고유한 개념은 아니다.

너무 이론적인 내용만 즐비하다고? 벌써 실망하지 마시길! 다음 장부터는 지극히 실전적인 내용 위주로 살펴볼 것이다.

1.5 다음 장에서는

다음 장부터는 분산 시스템의 내부 구조나 추상화된 분산 컴퓨팅 모델 따위는 접어두고, 여러분이 로그에 애정을 느낄 수 있도록 다음과 같은 내용을 알아볼 것이다.

데이터 통합

조직의 전체 데이터를 모든 스토리지와 처리 시스템에서 쉽게 접근하여 사용할 수 있게 함

실시간 데이터 처리

파생된 데이터 스트림 연산

분산 시스템 설계

로그 중심적인 설계로 시스템을 단순화하는 방법

앞의 세 가지 모두, 로그를 하나의 스탠드얼론standalone 서비스로 바라보는 인식에서 비롯되었다. 지속적persistent이면서 재연 가능한replayable 이력 데이터를 생성하

08 http://www.datomic.com/
09 http://bit.ly/log-centric

는, 로그의 단순한 기능만으로도 그 유용함은 충분히 입증되고도 남을 것이다. 다수의 기계를 확정적인 방식으로, 각자에게 맞는 비율로 이력을 재생시키는 로그의 능력이 핵심이다.