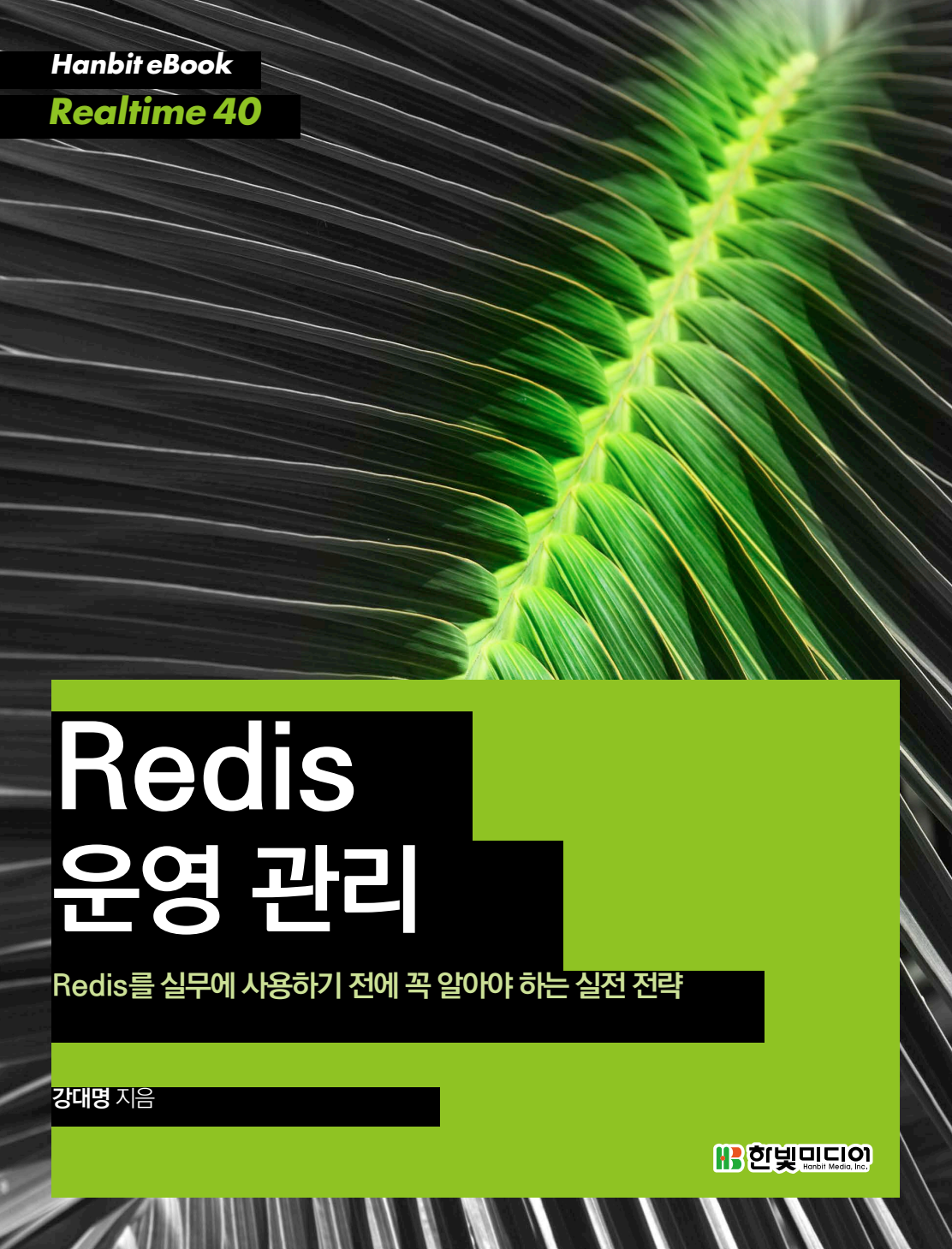
Hanbit eBook
Realtime 40
Redis
운영 관리
Redis를 실무에 사용하기 전에 꼭 알아야 하는 실전 전략
강대명 지음
한빛미디어
Hanbit Media, Inc.

Redis 운영 관리

Redis를 실무에 사용하기 전에 꼭 알아야 하는 실전 전략

지은이_ **강대명**

빅데이터와 클라우드에 관심이 많은 개발자다. 새로운 세상을 보러 어학연수를 다녀왔으며, 현재는 카카오에서 새로운 개발자 인생을 즐기고 있다.

Redis 운영 관리 : Redis를 실무에 사용하기 전에 꼭 알아야 하는 실전 전략

초판발행 2013년 8월 30일

지은이 강대명 / **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 978-89-6848-647-0 15000 / **정가** 9,900원

책임편집 배용석 / **기획** 김병희 / **편집** 안선화
디자인 표지 여동일, 내지 스튜디오 [밈], 조판 김현미
마케팅 박상용, 박주훈

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

Published by HANBIT Media, Inc. Printed in Korea

저자 서문

현재 우리는 수많은 서비스를 이용하고 있습니다. 이러한 서비스들은 우리가 모르는 사이에 수없이 생겨났다 사라지고 있어서, 서비스를 구축하는 개발자 입장에서는 모든 것을 스스로 개발할 수도 없고 함부로 오픈 소스를 사용하기도 힘듭니다. 그래서 잘 알려진 대형 서비스에서 많이 사용하는 기술이라면 안정적이거나 성능이 좋으리라고 믿게 됩니다. 그러다 보니 대형 서비스에서 이용하는 기술에 대한 관심이 늘게 마련이고, 개발하는 서비스에도 자연스레 이것을 이용하게 됩니다.

대형 서비스에서 이용하여 유명해진 기술의 대표적인 케이스가 Redis일 것입니다. 트위터, 인스타그램, 텀블러 등 비교적 큰 소셜미디어 업체에서 사용하고 있으니 안정성은 어느 정도 검증되었고, 성능이 빠르다는 점 역시 Redis를 선택하게 되는 이유입니다. 하지만 좋은 약은 또한 나쁜 독과도 일맥상통하는 법이지요. 중요한 서비스에 Redis를 사용하지만, 어떻게 사용할지를 제대로 알지 못한다면 도리어 큰 화를 입을 수 있습니다. 실제로 Redis를 사용하는 많은 사람이 잘못된 지식으로 인해 모든 데이터를 날리는 것은 물론, 잘못된 형태로 사용하는 경우도 비일비재합니다. 필자 스스로 그런 모든 부분을 알지도 못하거니와, 이 책에서 역시 그 부분에 대해 모두 설명하지는 못할 것입니다. 그렇지만 이 책이, 대표적인 경우에 한해서는 미리 알고 대비할 수 있는 작은 보험쯤은 되어줄 수 있으리라 생각합니다.

마지막으로 책을 집필하는 동안 함께 많은 시간을 보내주지 못한 사랑하는 아내와 9개월 뒤에 태어날 우리의 소중한 아이에게 감사하다는 말을 전합니다.

집필을 마치며

지은이 **강대명**

대상 독자 및 Redis 관련 웹 사이트

초급 | 초중급 | **중급** | 중고급 | 고급

이 도서는 Redis를 어느 정도 알고 있으며, Redis를 실무에 사용하고자 하는 개발자를 대상으로 한다. Redis 사용에 부담을 느끼고 있는 개발자들에게, Redis를 실무에 사용할 때 필요한 팁이나 문제 해결 방법 등 실전 전략을 알려줄 것이다.

- Redis 응용에 관심이 있는 중급 개발자
- Redis의 내부구조를 익혀서 장애 처리나 운영에 사용하려는 개발자

Redis 관련 웹 사이트

- Redis 홈 페이지 : http://redis.io
- Redis 기본 명령어 : http://redis.io/commands
- Redis 소스 코드 : http://redis.io/download
- Redis 소스 저장소 : https://github.com/antirez/redis

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

1 | Redis의 이해

1.1 Redis란 무엇인가?

최근에 회자되고 있는 대규모 서비스를 운영하는 업체(인스타그램[01], 트위터[02], 핀터레스트[03], 텀블러[04] 등)의 엔지니어링 블로그를 살펴보면, 공통으로 자주 등장하는 기술들이 있다. 키워드를 뽑아보면 NoSQL, Cache, Redis, Memcache, Sharding 등으로 모두 대용량 데이터 처리 관련 기술이다.

즉, 대규모 서비스를 운영하는 업체는 항상 데이터의 안정적인 저장과 빠른 처리를 원하는데, 이를 위해 위의 키워드와 관련된 기술들이 필요하다. 이 책에서 소개할 Redis('레디스'라고 읽음)를 어떤 사람은 속도를 빠르게 하기 위한 '캐시Cache 솔루션'이라고 정의하고, 어떤 사람은 NoSQL의 Key-Value 스토어로 분류하기도 한다.

어려운 의미를 이해하기 전에, '캐시 솔루션'과 'Key-Value 스토어'의 공통적인 특징부터 짚고 넘어가도록 하자. 이 둘의 특징은 사용하기 쉽고 속도가 빠르다는 것이다.

먼저 Redis 관련 공식 정보를 정리하면 표 1-1과 같다.

01 http://instagram.com/
02 https://twitter.com/
03 https://pinterest.com/
04 https://www.tumblr.com/

표 1-1 Redis 관련 공식 정보

항목	내용
홈 페이지	http://redis.io
소스 저장소	https://github.com/antirez/redis
개발자	살바토레 산필리포(Salvatore Sanfilippo)
최신 버전	Redis 2.6.14 (http://redis.io/download)
로고	

Redis의 기본 명령어는 "http://redis.io/commands"에서 볼 수 있다. 명령어 별로 상세하게 정리되어 있으니, 관련 웹 사이트를 참고하기 바란다(다만, Sentinel 관련 명령어와 최근 추가된 명령어에 대한 상세한 정보는 없다. 그래도 주요 명령어들은 상세히 정리되어 있으니 참고하라).

Redis의 주요 사용자들은 이미 대규모 서비스를 운영하고 있는 큰 기업들이다. 트위터, 인스타그램, 텀블러 등 해외 소셜미디어들을 비롯해 국내 IT 기업인 네이버 라인 , 카카오 등에서도 중요 서비스를 구성하는 데 Redis를 사용하고 있다.

표 1-2 대규모 서비스를 제공하는 기업들의 Redis 사용 정보

항목	내용
트위터	1억 5천만 명의 액티브 사용자를 위해서 800대의 Redis를 사용하고 있다.[05]
핀터리스트	오브젝트 캐싱 용도로 사용하고 있다. 작년 자료니 올해는 더 많은 장비를 사용하고 있을 것으로 예상된다.[06]
인스타그램	Redis가 핵심이라고 할 정도로 인스타그램 서비스의 중요한 부분(메인피드, 세션) 등을 처리하고 있다.[07]
텀블러	분산 서비스를 위해서 Redis를 많이 사용하고 있다.[08]
라인	수십억 개의 행을 저장하기 위해서 Redis를 사용하고 있다.[09]

또한 신규 서비스를 준비하는 국내의 스타트업이나 국외의 소셜 게임 업체도 Redis를 도입하거나 도입하려고 준비 중이다. 그렇다면 이런 업체들은 왜 Redis를 사용할까? 이 책에서는 이렇게 많은 업체에서 Redis를 선택하게 된 데 결정적인 역할을 한 Redis의 주요 특성과 이를 안정적으로 운영하기 위해 필요한 팁에 대해서 알아볼 것이다.

1.2 Redis의 주요 특성

Redis를 사용하기 전에 우선 Redis의 특성을 간략하게 살펴보자.

---

05 http://highscalability.com/blog/2013/7/8/the-architecture-twitter-uses-to-deal-with-150m-active-users.html

06 http://highscalability.com/blog/2012/2/16/a-short-on-the-pinterest-stack-for-handling-3-million-users.html
http://highscalability.com/blog/2012/5/21/pinterest-architecture-update-18-million-visitors-10x-growth.html

07 http://highscalability.com/blog/2012/4/9/the-instagram-architecture-facebook-bought-for-a-cool-billio.html

08 http://highscalability.com/blog/2012/2/13/tumblr-architecture-15-billion-page-views-a-month-and-harder.html

09 http://tech.naver.jp/blog/?p=1420

표 1-3 Redis의 주요 특성

항목	내용
Key-Value 스토어	단순 스트링에 대한 Key/Value(키/밸류) 구조를 지원한다.
컬렉션 지원	List, Set, Sorted Set, Hash 등의 자료 구조를 지원한다.
Pub/Sub 지원	Publish/Subscribe 모델을 지원한다.
디스크 저장 (Persistent Layer)	현재 메모리 상태를 디스크로 저장할 수 있는 기능과 현재까지의 업데이트 관련 명령을 저장할 수 있는 AOF 기능이 있다.
복제(replication)	다른 노드에서 해당 내용을 복제할 수 있는 마스터/슬레이브 구조를 지원한다.
빠른 속도	이상의 기능들을 지원하면서도 초당 100,000QPS(Queries Per Second) 수준의 높은 성능을 자랑한다.

1.2.1 Key-Value 스토어

기본적으로 Redis는 Key-Value 형태의 데이터 저장소다. 그래서 다음과 같이 간단한 명령을 이용하여 데이터를 저장할 수 있다.

```
$>set id:username “ username ”
$>set id:email test@test.com
$>get id:username
```

1.2.2 컬렉션 지원

Redis의 가장 중요한 특징을 꼽으라면, 주저하지 않고 ‘컬렉션 기능’이라고 말할 것이다. 일반적으로 사용하는 컬렉션은 하나의 서버 내부에서 동작한다. 그러나 Redis를 이용하면 이런 특성을 분산 서버 환경에서 처리할 수 있어서, 전체적인 서비스를 설계하거나 구현할 때 많은 이점을 얻을 수 있다.

1.2.3 Pub/Sub 지원

Redis는 Publish/Subscribe 기능을 지원한다. 서버 간에 통지가 필요할 때, 이 기능이 매우 유용하다.

1.2.4 디스크 저장

Redis의 특징 중 하나는 현재의 메모리 상태를 디스크에 저장할 수 있다는 것이다. Redis에는 현재 메모리 상태의 스냅샷을 남기는 'RDB 기능'과 지금까지 실행된 업데이트 관련 명령어의 집합인 'AOF'가 있다.

다만, 여기엔 주의해야 할 점이 있다. 스냅샷을 남기는 기능의 이름이 'RDB'이다 보니 데이터베이스라고 생각해서 데이터베이스의 모든 기능 역시 지원되지 않을까 기대하는 개발자가 있는데, 단순히 이름만 'RDB'일 뿐 메모리 내용을 저장하는 기능 이외에는 아무것도 지원하지 않는다. 이렇게 덤프한 내용은 다시 메모리에 올려서 사용할 수 있다.

AOF는 "Append Only File"의 약어로, set/del 등의 업데이트 관련 명령을 받으면 해당 명령어를 그대로 기록해둔다. Redis에서는 가능하면 이 두 개를 모두 사용하는 것이 좋다고 이야기하지만, 디스크를 사용해서 저장하는 만큼 성능 손실은 어느 정도 감수해야 한다.

1.2.5 복제

Redis는 마스터/슬레이브 리플리케이션Replication, 복제을 지원한다. 이를 통해 마스터에 장애가 발생하면 슬레이브로 서비스하거나 마스터의 부하가 많을 때에는 슬레이브를 이용해서 읽기를 처리할 수도 있다. 대규모 서비스에서 Redis를 저장소로 안정적으로 사용하려면 복제 기능을 반드시 이용해야 한다.

1.2.6 빠른 속도

Redis를 선택하는 가장 큰 이유는 '성능'이다. 초당 50,000~60,000QPS 이상의 처리 속도가 필요하다면, 어쩔 수 없이 Redis나 Memcached를 사용해야 한다.

1.3 Redis와 Memcached 비교

필자는 Redis 관련 강의를 여러 곳에서 했는데 강의 중에 "Redis와 Memcached를 비교해보아, 어떤 것이 더 좋은가요?"라는 질문을 많이 받아 왔다. 아주 단순하게 비교하자면, Memcached는 캐시 솔루션이고 이러한 Memcached에 저장소의 개념이 추가된 것이 Redis라고 할 수 있다. '캐시'는 빠른 속도를 위해서 어떤 결과를 저장해 두는 것을 의미하며, 또한 '데이터가 사라지면 다시 만들 수 있다'는 전제를 내포하고 있다. 그런데 저장소라는 개념이 추가되면 '데이터가 유지되어야 한다'는 특성을 가지게 된다. 즉, Redis는 이런 특성으로 말미암아 디스크에 저장하는 RDB나 AOF, 복제Replication가 추가된 것이라고 보면 된다.

이 외에도 Redis는 컬렉션이라는 자료구조(List, Hash, Set, Sorted Set)를 제공하므로, 개발의 생산성 측면에서 볼 때 'Redis'가 'Memcached'보다 좋다. 이상의 내용을 표 1-4로 간단히 정리했다.

표 1-4 Redis와 Memcached의 장단점 비교

기능	Redis	Memcached
속도	초당 100,000QPS 이상	초당 100,000QPS 이상
자료구조	Key-Value, List Hash, Set Sorted Set 지원	Key-Value만 지원
안정성	특성을 잘못 이해할 경우, 프로세스 장애 발생	장애 거의 없음
응답 속도의 균일성	Memcached에 비해서 균일성이 떨어질 수 있음	전체적으로 균일함

표 1-4에서 마지막 항목의 '응답 속도의 균일성' 부분을 조금 더 자세히 알아보자. 대규모 트래픽으로 인해 많은 데이터가 업데이트되면, Redis는 Memcached에 비해서 속도가 출렁인다. 이것은 Redis와 Memcached의 메모리 할당 구조가 다르기 때문에 발생하는 현상이다. Redis는 jemalloc을 사용하는데, 매번 malloc과 free를 통해서 메모리 할당이 이루어진다. 반면 Memcached는 slab 할당자를 이용하여, 내부적으로는 메모리 할당을 다시 하지 않고 관리하는 형태를 취한다. 이로 인해서 Redis는 메모리 프래그멘테이션fragmentation 등이 발생하며 이 할당 비용 때문에 응답 속도가 느려진다. 다만, 이는 극단적으로 봤을 때 발생하는 일이다. 대규모 서비스에서도 Redis를 많이 도입하는 것으로 보아, 일반적으로 스타트업 등에서 사용하기에는 거의 문제가 없다고 봐도 될 것이다.

여기서 잠깐_ jemalloc 메모리 할당자

'jemalloc'은 제이슨 에반스Jason Evans가 FreeBSD의 메모리 할당 성능을 개선하기 위해서 만든 메모리 할당자다. 메모리 할당/해제를 많이 하는 프로그램에서는, 메모리 할당자를 jemalloc로 바꾸는 것만으로도 성능을 10~20퍼센트 정도 향상할 수 있다. 이는 메모리 할당 정책에 따라서 공간이나 속도에 영향을 받게 되고(운영체제 시간에 배운 Best Fit, Worst Fit 등을 떠올려보기 바란다), 심지어는 실제로 메모리가 남지만 사용하지 못하는 경우도 발생할 수 있기 때문에 가능한 일이다.

성능을 높여주는 메모리 할당자로는 구글에서 개발한 'tcmalloc'과 'jemalloc'이 가장 유명하다. 'tcmalloc'과 'jemalloc'을 사용하지 않고 있다면, 메모리 할당자를 이 둘로 바꾸기만 해도 쉽게 성능을 어느 정도 향상할 수 있다. 이에 대해 필자가 자세히 설명하기보다는, KTH 개발 블로그의 문서를 보는 게 여러분에게 더 도움이 될 것이다.[10]

10 http://dev.kthcorp.com/2011/05/12/last-free-lunch-facebooks-memory-allocator-jemalloc/

1.4 Redis 빌드해서 사용하기

Redis를 사용하려면 먼저 이를 빌드해야 한다. 이 책에서는 Redis 2.8.0-rc1 버전을 사용하며, 리눅스에서 Redis를 사용한다 가정하고 빌드 방법을 설명할 것이다. 먼저 "http://redis.io/download"에서 'redis-2.8.0-rc1.tar.gz' 파일을 다운받는다.

> **NOTE_** Redis는 현재 공식적으로 리눅스와 FreeBSD, Mac 등을 지원하고 있다. 윈도우는 공식적으로 지원하지 않으며, 필요하다면 MS 연구소인 'MSOpenTech'에서 패치한 버전을 사용해야 한다. 윈도우에서의 사용 방법은 이 책에서 다루지는 않는다. MSOpenTech의 패치 버전은 "https://github.com/MSOpenTech/redis"에서 다운받을 수 있다.

① 다운로드

```
$> wget http://download.redis.io/releases/redis-2.8.0-rc1.tar.gz
```

② 압축해제

```
/$> tar zxvf redis-2.8.0-rc1.tar.gz
```

③ 빌드

Redis의 빌드는 정말로 쉽다. 해당 디렉터리로 이동해서 make만 실행하면 된다.

```
$> cd redis-2.8.0-rc1
$> make
$> make test
```

단, make test를 이용하여 빌드를 확인하려면 'tcl 8.5' 버전 이상이 설치되어 있어야 한다.

여기서 잠깐_ Redis 빌드가 안 돼요!

가끔씩 공식적으로 지원하는 운영체제(리눅스/FreeBSD 등)에서 Redis가 빌드되지 않는 경우가 있다. 이 문제는 크게 두 가지 경우로 나눌 수 있다.

① __sync_add_and_fetch_4 관련 이슈

Redis는 32bit와 64bit 컴퓨터에서 빌드할 수 있다. 보통 64bit에서도 32bit를 사용할 수 있는데, __sync_add_and_fetch_4 관련 이슈는 64bit 컴퓨터를 사용하면서 32bit로 빌드하려는 사용자에게 많이 발생하는 이슈다. 빌드할 때 "__sync_add_and_fetch_4 함수를 찾을 수가 없다"라는 메시지가 뜨는데, 이 문제를 해결하려면 Makefile의 CFLAGS 속성에 ' - march=i686' 또는 ' - march=native'를 추가해야 한다. 해당 명령이 i686 이전 CPU 명령에 추가되어 있지 않은데 해당 형태로 빌드되어서 발생하는 이슈다.

gcc 4.2 이전 버전일 경우, - march=native를 사용하면 현재 CPU에 맞는 형태로 자동으로 빌드해준다. 다만, Makefile을 써본 개발자라면 그냥 make 32bit CFLAGS="-march=i686" 등으로 설정해주면 되지 않느냐고 생각할 수 있는데, Redis에서는 이 형태가 32bit로 빌드할 때 제대로 동작하지 않는다(64bit에서는 제대로 동작한다). 해당 Makefile의 32bit 라벨을 살펴보면 그 이유를 알 수 있다.

```
32bit:
    @echo ""
    @echo "WARNING: if it fails under Linux you probably need to install libc6-
    dev-i386"
    @echo ""
    $(MAKE) CFLAGS="-m32" LDFLAGS="-m32"
```

보는 바와 같이 이미 CFLAGS를 덮어써버리기 때문에, 외부에서 설정해둔 값이 사라져서 인식하지 못한다.

② make를 할 때, 특정 헤더를 찾지 못하는 이슈

Redis에서는 최초 make를 실행하면 '.make-prerequisites'과 '.make-setting'을 미리 만들어 두는데, 이 파일이 있으면 deps 폴더에 있는 라이브러리를 다시 빌드하지 않는다. 그래서 빌드 과정 중에 뭔가 오류가 발생하면(주로 컴파일러가 설치되어 있지 않고 make를 시도하는 경우에 발생한다. 예전에는 aws로 인스턴스를 생성하면 gcc가 설치되지 않은 경우도 있었다). 그래서 이 경우에는 deps 폴더를 강제로 다 빌드해주거나, 'make distclean'이라는 명령을 이용하여 빌드한다. 다음과 같이 하면 빌드 오류를 해결할 수 있다.

```
$> make distclean
$> make
```