SPONSORED BY

IBM

GEEK 가이드

증가하는 Redis NoSQL 서버 클러스터 제어

LINUX™ JOURNAL

목 차

후원사 소개 4

개요 5

Redis 사용 10

단일 서버 Redis 15

다중 서버 Redis 및 복제 18

IBM의 CAPI 기술 21

CAPI가 적용된 Redis 사용 24

여러분에게 필요한 기술인가요? 26

결론 28

REUVEN M. LERNER은 웹 개발자, 컨설턴트 겸 강사이며, 오랫동안 Linux Journal에서 컬럼니스트로 활동해 왔습니다. 최근에 Northwestern 대학에서 학습 과학 분야의 박사 과정을 이수했습니다. 블로그와 Twitter 계정을 운영하고 있으며, http://lerner.co.il.에서 뉴스레터를 받아볼 수 있습니다. Reuven은 이스라엘 모디인에서 아내, 세 자녀와 함께 살고 있습니다.

GEEK 가이드:

전 세계 기술인을 위한 미션 크리티컬 정보.

후원사 소개

IBM은 뉴욕 아몽크에 본사를 둔 세계적인 통합 기술 및 컨설팅 기업입니다. 170여 개국에서 기업, 정부 및 비영리 조직의 문제를 해결하고 경쟁 우위를 제공하고 있습니다. 혁신은 IBM 전략의 핵심입니다.

IBM은 소프트웨어, 시스템 하드웨어, 광범위한 인프라, 클라우드 및 컨설팅 서비스를 개발 및 판매합니다. 예를 들어, IBM Power Systems는 미션 크리티컬 애플리케이션을 위해 개방형 기술을 기반으로 구축되었습니다. 이를 통해 빅데이터용으로 설계되었으며, 변화하는 비즈니스 요구에 맞게 최적화되고, 보안되고, 조정된 서버를 제공합니다.

현재 IBM은 클라우드, 빅데이터 및 분석, 모바일, 소셜 비즈니스, 보안이라는 5가지 성장 이니셔티브에 주력하고 있습니다. IBMer는 비즈니스 컨설팅, 기술 및 R&D 전문성을 발휘하여 모든 기업 및 정부에 동적 통찰력을 제공하는 인게이지먼트 시스템을 지원하기 위해 전 세계 고객과 협력하고 있습니다.

증가하는 Redis NoSQL 서버 클러스터 제어

REUVEN M. LERNER

개요

1990년대 중반에 처음 웹 애플리케이션 개발을 시작했을 때는 세션 전체에서 데이터를 유지하려면 파일을 사용해야 하고 그것이 가능하다고 생각했습니다. 하지만 곧 파일시스템 사용은 좋은 생각이 아니며 대신 서버의 관계형 데이터베이스를 사용해야 한다는 것을 깨달았습니다. 데이터베이스를 사용함으로써 여러 서버에서 같은 데이터를 액세스할 수 있게 되었습니다. 그뿐만 아니라 관계형 데이터베이스를 통해 사용자의 의도를 명확히 보여주는 데이터 유형(예: 숫자 또는 텍스트 스트링)을 사용하여 적절한 형식으로 데이터를 저장할 수 있습니다.

Redis에 푹 빠진 사람은 저뿐만이 아닙니다. Redis는 사용 편의성과 빠른 실행 속도에서 최고의 조합을 선사하며, 제공하는 기능 또한 다양합니다.

20년이 지난 지금은 개발자라면 누구나 데이터베이스가 답이라는 것을 알고 있습니다. 이제 문제는 데이터베이스를 사용해야 하느냐가 아니라 어떤 데이터베이스 기술을 사용해야 하느냐입니다. SQL과 NoSQL 사이의 논쟁이 이어지고 있습니다.

SQL 지지자는 ACID 규정 준수를 위해 설정된 기준, 정규화, 관계형 데이터베이스에 쏟아 부은 수년간의 노력과 시간을 강조합니다. NoSQL 지지자는 지금과 같은 웹 규모 애플리케이션의 시대에 관계형 데이터베이스는 구시대적이며, 유연하고, 스키마가 없는, 복제 가능한 데이터베이스를 사용해야 한다고 주장합니다.

저는 일반적으로 SQL 지지자에 속합니다. 관계형 데이터베이스가 역할을 잘 수행하고 있으며, 시간이 지나면서 점점 더 정교해지고 있다고 생각합니다. NoSQL이 잘못되었다는 뜻은 아니지만, NoSQL에 회의적인 것은 사실입니다.

하지만 저의 NoSQL 회의론에 하나의 중요한 예외 사항이 있습니다. 바로 Redis입니다. Redis에 푹 빠진 사람은 저뿐만이 아닙니다. Redis는 사용 편의성과 빠른 실행 속도에서 최고의 조합을 선사하며, 제공하는 기능 또한 다양합니다. Redis로 작업 하면서 한 번도 실망해본 적이 없습니다.

비즈니스와 개발자 사이에서 Redis는 얼마나 인기가 있을까요? 오픈 소스 제품의 사용률을 추적하기는 쉽지 않지만, 일부 지표에 따르면 Redis의 인기가 점점 더 상승하는 것으로 보입니다. DB-Engines(**http://db-engines.com**)의 데이터베이스 인기도 순위(웹 사이트, 소셜 네트워크 및 직업 게시판의 언급을 관찰)에서 Redis는 2014년 5월 13위에서 2015년 5월 현재 10위로 순위가 상승했습니다. Redis 소프트웨어의 상용 공급업체인 Redis Labs의 최근 발표에 따르면 2015년 1분기에 신규 고객이 33% 증가했다고 합니다. Redis는 또한 GitHub에 올려져 있습니다. GitHub는 프로젝트에 관심이 있거나 프로젝트에 기여하고자 하는 사람들에게 통찰력을 제공하는 호스팅 서비스입니다. 본 문서 작성 당시 Redis 프로젝트의 팔로워는 1,365명이었고 기여자는 70명이 넘었습니다. 반면에 Memcached의 팔로워는 449명이었고 기여자는 71명이었습니다.

비즈니스에서는 왜 Redis를 사용할까요? 그리고 어떻게 사용할까요? 대부분 장바구니나 세션 정보 같은 사용자 데이터의 캐싱을 위해 Redis를 사용하고 있습니다. 그리고 페이지 뷰와 소셜 네트워크 링크를 위한 정교한 카운터로도 사용할 수 있습니다. 이 외에도 데이터의 부정 사용을 발견하기 위해 최근 데이터에서 추세를 파악하는 데 사용할 수도 있습니다. 그뿐만 아니라 애플리케이션의 외부에 있는 메시지 큐의 형태로 게시-구독 프로토콜을 구현하기 위해 사용할 수 있습니다.

Redis는 유일한 키-값 스토어도 아니고 오픈 소스 라이센스하에 처음으로 출시된 데이터베이스도 아닙니다. 오랫동안 많은 개발자가 Memcached라는 이와 유사한 시스템을 사용했습니다.

또한, Redis는 놀라울 만큼 빠릅니다. 고속 데이터 캐시로서 사용할 수 있으므로 관계형 데이터베이스보다 빠르게 사용자 프로파일과 장바구니를 저장할 수 있습니다.

이름에서 알 수 있듯이 Memcached는 데이터를 메모리에만 저장했습니다. 따라서 Memcached 서버에 장애가 발생하거나 재부팅해야 하는 경우, 콘텐츠는 손실됩니다. 이와는 달리 Redis는 정보를 정기적으로 디스크에 저장하여 정전이나 다른 문제가 발생해도 데이터가 전혀 손실되지 않거나 일부만 손실됩니다.

Redis는 다양한 데이터 유형을 제공하여 텍스트 스트링 뿐만 아니라 스트링 목록, 스트링 세트(각 요소가 고유한), 저장된 스트링 세트 및 해시 테이블을 저장하고 조작할 수 있습니다. Redis를 사용하여 사용자의 현재 구매를 추적하려는 경우, 사용자의 고유 ID에 연결된 해시 테이블을 사용하면 됩니다. 서버를 방문한 여러 IP 주소나 방문한 사용자 ID 세트, 시스템에 사용자가 입력한 검색어를 확인하려는 경우, Redis 세트가 손쉽게 이러한 작업을 처리할 수 있습니다.

또한, Redis는 놀라울 만큼 빠릅니다. 고속 데이터 캐시로 사용할 수 있으므로 관계형 데이터베이스보다 빠르게 사용자 프로파일과 장바구니를 저장할 수 있습니다. 몇 년 전 금융 산업의 프로젝트에 참여했을 때 통화 가격을 지속적으로 추적해야 했고 그

때 Redis를 사용하여 추적하던 통화의 가장 최근 값을 저장할 수 있었습니다. 소스에서 빈번하게 검색해야 하는 이름-값 쌍을 쉽게 매핑할 수 있었으며 데이터베이스보다 안정적이고 빨랐습니다.

Redis는 클라이언트-서버, 키-값 스토어입니다. 즉, 네트워크 프로토콜을 통해 연결할 수 있으며, 해시 테이블처럼 작동합니다. 데이터는 키를 통해 저장되고 액세스되며, 키는 고유한 값이어야 합니다. 따라서 다음과 같이 키-값 쌍을 저장할 수 있습니다.

```
a = 1
```

또는

```
105 = reuven@lerner.co.il
```

이는 Redis의 모든 키가 고유해야 한다는 의미이며 키-값 스토어에도 적용되는 내용입니다. Redis는 여러 데이터베이스를 제공하며, 각 데이터베이스에는 번호가 붙는데 이 번호는 서버에 연결할 때 지정할 수 있습니다. 하지만 연결하려는 데이터베이스는 데이터를 읽거나 쓸 때가 아니라 연결하는 시점에 명시합니다. 그렇지만 키 이름을 계획할 때 키 이름이 겹치는 경우가 발생하지 않도록 할 수 있습니다.

예를 들어, 소프트웨어를 "foobar"라고 부르는 경우 모든 키 이름이 "foobar:"로 시작하도록 정할 수 있습니다. 콜론(:)은 Redis 키 이름에서 허용하는 항목으로, 특정 Redis 인스턴스 내에 일정 수준의 이름 공간을 생성하는 데 도움이 됩니다.

Redis 사용

최신 프로그래밍 언어 대부분은 Redis 클라이언트를 사용합니다. Redis-cli로 알려진 명령행 클라이언트를 사용하면 다른 언어를 통하지 않고 Redis와 직접 상호 작용할 수 있습니다. 이러한 인터페이스 내에서 다양하지만 상당히 기억하기 쉬운 Redis 명령을 사용할 수 있습니다. 예를 들어, 키-값 쌍 a:b를 다음과 같이 redis-cli를 사용하여 저장할 수 있습니다.

```
set a b
```

그런 다음, "a"와 연결된 값을 검색할 수 있습니다.

```
get a
```

다음 명령을 사용해 값을 수정할 수도 있습니다.

```
append  a c
```

이제

```
get a
```

명령이

```
bc
```

값을 반환하는 것을 볼 수 있습니다.

이와 같은 방법으로 목록, 세트, 저장된 세트 및 해시 테이블을 조작할 수 있습니다. 데이터 유형별로 명령이 다른데, 일반적으로

목록은 L로, 세트는 S로, 해시 테이블은 H로, 저장된 세트는 Z로 시작합니다. Redis는 원자적(atomic)으로 동작하므로, 다른 사람이 읽고 있는 세트를 수정할 때 무슨 일이 발생할지 걱정할 필요가 없습니다.

Redis 데이터 유형의 장점 중 하나는 원하는 작업 형태를 지원한다는 것입니다. 예를 들어, 일종의 카운터가 필요하다고 해보겠습니다. Redis는 Se별로 카운터 유형을 지원하지 않지만, Redis 스칼라 유형으로 정수(숫자만 포함하고 있는 스트링)를 저장하는 경우에는 incr 및 decr 명령을 사용하여 값을 늘리거나 줄일 수 있습니다. 예를 들어,

```
set  a  10
incr a
```

명령은

```
11
```

을 반환합니다. 그리고 다시

```
get a
```

명령을 입력하면

```
11
```

값을 얻게 됩니다. 마찬가지로 Redis 목록을 스택 및 큐처럼 사용하면 매우 유용합니다. lpush, rpush, lpop 및 rpop 명령을 조합하면 됩니다. 실제로 이러한 명령 중 하나를 호출하여 목록을 생

성할 수 있습니다. 지정한 목록이 존재하지 않는 경우 Redis에서 이를 생성합니다. 예를 들어, 다음과 같습니다.

```
lpush    numbers    1
lpush    numbers    2
lpush    numbers    1000
rpush    numbers    555
```

이제 인덱스 0(시작)에서 인덱스 -1(끝)까지 숫자 목록을 요청할 수 있습니다.

```
lrange  numbers 0 -1
```

Redis는 다음과 같이 응답합니다.

```
1) "1000"
2) "2"
3) "1"
4) "555"
```

물론 목록을 정렬할 수도 있습니다.

```
sort numbers
```

그러면 다음과 같이 정렬됩니다.

```
1) "1"
2) "2"
3) "555"
4) "1000"
```

세트는 목록과 유사하지만 순서가 없으며 고유한 요소를 갖는다는 점은 다릅니다. 예를 들어, "사용자" 세트에 여러 명의 사용자를 추가하려는 경우 다음과 같이 sadd 명령을 사용하면 됩니다.

```
sadd users  reuven
sadd users  root
sadd users  atara
sadd users  shikma
asdd users  amotz
```

그리고 "관리자" 세트에 3명을 추가합니다.

```
sadd administrators reuven
sadd administrators root
sadd administrators joeshmoe
```

이제 Redis에게 사용자가 아닌 사람이 관리자 목록에 있는지 물어볼 수 있습니다.

```
sdiff  administrators users
```

또한, Redis가 두 세트간의 상호 작용을 찾게 할 수도 있습니다.

```
sinter  administrators users
```

저는 지난 몇 년 동안 사용한 모든 프로그래밍 언어 중에서 세트를 가장 좋아합니다. 이로 인해 Redis를 더 선호하게 되어 많은 곳에 적용하게 되었고 이제 추적에도 사용하려고 합니다.

사용자, IP 주소 및 기타 데이터를 추적할 때 세트는 간편성과 유용성에서 비교할 수 없을 만큼 뛰어납니다.

사용자, IP 주소 및 기타 데이터를 추적할 때, 세트는 간편성과 유용성에서 비교할 수 없을 만큼 뛰어납니다. 데이터를 정렬해야 할 뿐 아니라 더불어 데이터가 고유해야 한다면, Redis에서 제공하는 "sorted set" 데이터 유형을 사용하는 것이 좋습니다.

마지막으로 Redis가 키-값 스토어(해시 테이블로도 알려짐)임을 고려할 때, Redis 데이터 유형 중 하나가 해시 자체인 것이 이상하게 보일 수도 있습니다. 하지만 한편으로는 Redis에서는 가장 적합하며 매우 유용한 기능이기도 합니다.

예를 들어, 다음과 같이 입력할 수 있습니다.

```
hset  person  first_name Reuven
hset  person last_name Lerner
hset person  email  reuven@lerner.co.il
```

그런 다음 hget 명령을 사용하여 이러한 필드 중 하나를 검색할 수 있습니다.

```
hget person  email
```

그러면 다음 값이 반환됩니다.

```
"reuven@lerner.co.il"
```

hexists 명령은 해시 이름과 키 이름을 가져가서 키가 존재하면 1을 반환하고, 존재하지 않으면 0을 반환합니다. 존재하지 않는 키를 검색하는 경우, Redis에서는 예외 처리를 하거나 오류를 일으키지 않고 "nil" 값을 반환합니다.

Redis 데이터 유형에 대해서는 아직 설명할 것이 많습니다. Redis가 제공하는 풍부한 기능은 Redis가 단순히 오래된 키-값 스토어가 아님을 보여줍니다. Redis가 문서 데이터베이스인지 객체 데이터베이스인지 말할 수 없더라도, Redis는 단지 키별로 하나의 단순 값을 저장하는 것 이상의 역할을 하고 있습니다.

단일 서버 Redis

Redis를 설치하고, 구성하고, 실행하는 것은 생각보다 간단합니다. 대부분 몇 분 만에 다운로드하고, 설치하고, 실행할 수 있습니다.

예를 들어, Ubuntu를 구동하는 Linux 시스템에서 다음 명령을 실행하여 Redis 서버와 redis-cli 프로그램(다른 명령행 도구 포함)을 설치할 수 있었습니다.

```
sudo apt-get install redis-server redis-tools
```

설치되면 다음 명령을 통해 시작할 수 있습니다.

```
sudo service redis-server start
```

Ubuntu 패키지의 Redis 서버 구성은 "demonized"로 구성됩니다. 즉, 시작할 때 서버가 백그라운드에서 실행됩니다. 이는 서버

실행 시 일반적으로 선호하는 방식입니다. 디버깅할 때는 포그라운드 실행이 편리하기도 하지만 그 외에는 부적절한 방식입니다.

간단한 Redis 서버를 실행하는 경우, 별다른 항목이 없습니다. 구성 파일이 있어서 이를 수정할 수도 있지만, 서버 구성은 단순 명료한 프로세스입니다. 구성 파일은 일반적인 UNIX 스타일 형식으로서, 이름을 설정한 다음 적절한 값을 설정하며 주석 줄은 해시 표시(#)로 시작합니다.

구성은 잘 정리된 몇 가지 문서 섹션으로 되어 있습니다. 첫 번째는 기본 구성 및 네트워킹으로서, Redis가 청취해야 할 포트와 확인해야 할 IP 주소가 무엇인지 알려줍니다.

두 번째 구성 항목은 Redis가 현재 데이터 세트(RAM에 있는)를 디스크에 저장 시 어떻게 처리해야 하는지 알려줍니다. 이전 버전은 2초 간격으로 데이터를 디스크에 저장했습니다. 대부분의 경우 충분했으나, 이는 2초간의 데이터는 손실될 수 있다는 것을 의미합니다.

AOF(append-only file)로 알려진 다른 시스템은 2초 간격으로 완전한 데이터 세트를 작성하는 대신, 데이터 스트림을 연속으로 디스크에 작성함으로써 인 메모리 데이터 세트의 변경 사항을 표시합니다. 시스템이 다시 시작될 때, Redis가 AOF를 재생하여 종료되기 전 상태로 돌아갈 수 있습니다. Redis 설명서에서는 두 시스템을 함께 사용하여 안정성을 최대화할 것을 권장 합니다.

Redis에게 데이터 백업을 어떻게 지시해야 할까요? RDB나 AOF 중 하나 또는 둘 다 사용해야 할까요?

Redis 구성 파일에서 파일이 저장되는 위치뿐만 아니라 스냅샷을 디스크에 저장하는 시점을 제어하는 규칙도 설정할 수 있습니다. "저장" 구성은 저장 빈도를 2개의 숫자로 지시하여 얼마나 많은 키가 변경되었는지에 따라 Redis가 정보를 디스크에 작성하기까지 얼마나 기다려야 하는지 알려줍니다. 제 구성 파일의 기본 명령행은 다음과 같습니다.

```
save    900 1
save    300 10
save    60 10000
```

이는 키가 1개 변경되었으면 900초 후에, 키가 10개 변경되었으면 300초 후에, 키가 10,000개 변경되었으면 60초 후에 Redis가 디스크에 스냅샷을 저장해야 한다는 뜻입니다. 다시 말해 변경된 키의 수가 증가하면 Redis가 디스크에 저장하는 빈도를 높인다는 의미입니다.

Redis에게 데이터 백업을 어떻게 지시해야 할까요? RDB나 AOF 중 하나 아니면 둘 다를 사용해야 할까요? 중요한 데이터라면 두 방법을 모두 사용하여 안정성을 최대화하는 것이 좋습니다. 하지만 Redis를 캐시로만 사용하고 모든 데이터가 다른 장소에 존재

하는 경우, Redis가 지속성을 제공하면 더 좋겠지만, 꼭 필요한 것은 아닙니다. AOF는 실행하지 않고 RDB 스냅샷만 사용하여 좀 더 느슨하게 진행해도 무방합니다.

아마도 Redis를 구성할 때 알아야 할 가장 중요한 점은 대부분의 경우 시스템을 튜닝할 필요가 전혀 없거나 약간의 튜닝만 하면 된다는 것입니다. 설치하고, 구성을 일부 변경하기만 하면 준비가 완료됩니다. 이는 분명 지금까지 제가 Redis 작업을 즐거워하는 이유 중의 하나입니다.

다중 서버 Redis 및 복제

애플리케이션이 증가하면서 애플리케이션의 데이터에 대한 수요도 증가했습니다. Redis는 대량의 클라이언트를 처리할 수 있습니다. 완전한 데이터베이스라기보다는 키-값 스토어이기 때문입니다. 그럼에도 불구하고 Redis에도 한계가 있습니다. 일부 NoSQL 데이터베이스와는 달리 샤딩 지원이 처음이라(본 문서 작성 당시) 아직 불안정할 수 있습니다. 샤딩이 없으면 전체 데이터 세트가 단일 컴퓨터의 메모리에 존재합니다. 즉, 데이터를 읽거나 쓰려는 클라이언트가 너무 많으면 해당 단일 컴퓨터가 병목 지점이 될 수 있습니다.

마스터-슬레이브 복제가 이를 해결하는 솔루션이 될 수 있습니다. 다른 마스터-슬레이브 데이터베이스 구성과 마찬가지로 실제로 변경이 발생하는 컴퓨터를 "마스터"로 지정합니다. 이러한 구성에서는 마스터 컴퓨터만 쓰기 요청을 수락하게 됩니다. 쿼리의 대부분이 읽기라는 가정에 따라 읽기를 하나 이상의 슬레이브 머신에 분배함으로써 로드를 분산합니다. (사실 Redis에서 쓰기 요청을 수락하도록 구성할 수 있지만, 슬레이브에 작성한 데이터가

영구적이지 않다는 것을 고려하면 이러한 사용 사례는 매우 드물 것입니다. 앞으로는 이 기능이 없어질 가능성이 큽니다.)

Redis의 마스터-슬레이브 복제는 쉽게 설정할 수 있습니다. 먼저, Redis를 구동하는 컴퓨터 2대(단순히 테스트만 하려는 경우) 또는 각각 다른 디렉터리와 포트에서 Redis 인스턴스 2개가 실행되고 있는 컴퓨터 1대가 필요합니다. 슬레이브에서 다음과 같이 구성 지시문을 추가합니다.

```
slaveof IP PORT
```

여기서, IP와 포트가 마스터 Redis 서버의 IP 주소와 포트로 설정되어야 합니다. 마스터 Redis 서버에서 인증 암호를 요구하도록 설정한 경우, 클라이언트의 연결 호출에 암호를 추가해야 합니다.

설정이 완료되면, 마스터-슬레이브 복제가 문제없이 작동할 것입니다. 마스터에서 읽거나 마스터에 쓸 수 있고 슬레이브에서 읽을 수 있습니다. 슬레이브 서버는 여러 대 설정할 수 있으며, 활성 슬레이브가 최소(min-slaves-to-write) 임계값보다 높을 때만 마스터가 쓰기 요청을 수락하도록 지정할 수도 있습니다.

Redis의 마스터-슬레이브 복제는 많은 조직에게 적합한 솔루션이 될 수 있습니다. 하지만 조직이 확장하면서 마스터-슬레이브로는 충분하지 않을 때가 있습니다. 이런 경우 Redis 클러스터로 전환하게 됩니다. 마스터-슬레이브 복제에서 각 컴퓨터(마스터와 모든 슬레이브)는 완전한 데이터 세트 복사본을 갖게 됩니다. Redis 클러스터에서 데이터가 "샤딩"된다는 것은 여러 대의 서버로 자

동 분할된다는 뜻입니다. 6대의 Redis 서버가 있는 경우, 하나의 서버에 모든 데이터를 저장하지 않고, 여러 서버에 걸쳐 데이터가 분산됩니다. 이를 통해 데이터 가용성이 향상될 뿐 아니라 하나 이상의 클러스터 노드를 사용할 수 없게 되도 하나 이상의 남아 있는 서버를 통해 데이터에 액세스할 수 있습니다.

Redis는 뛰어난 성능과 안정성으로 잘 알려져 있습니다. 하지만 클러스터링이 Redis를 분산 시스템으로 만들면서 이러한 분산 시스템에 내재된 일부 문제가 Redis의 성능과 안정성에 영향을 줄 수 있습니다. 예를 들어 다른 Redis 노드가 다른 쓰기 쿼리(write query)를 성공적으로 수신하면, 이미 Redis 클라이언트가 승인한 쓰기 쿼리의 데이터가 무효화되는 경우가 발생할 수 있습니다. Redis 노드가 증가하거나 감소할 수도 있고, 여러 노드에 걸쳐 데이터가 약간 다르거나 일관성이 없을 수도 있습니다.

클러스터 소프트웨어가 Redis 3.0.0의 일부로 2015년 4월 1일에 처음 출시되었으므로, Redis 클러스터링에 대해 아직 모르는 것이 많습니다. 얼마나 안정적일까요? 오류를 얼마나 쉽게 처리할까요? 클러스터를 최대 1,000개 노드까지 확장하려면 어떻게 해야 할까요?

그 동안 Redis가 일구어온 놀라운 발전을 깎아 내리거나 Redis 클러스터 사용을 방해하려고 이런 이야기를 하는 것이 아닙니다. Redis 클러스터는 새로운 기술이고, 모든 새로운 기술이 그렇듯이 (특히 분산 컴퓨팅의 요소가 추가될 때는) 최소한 약간의 테스트 기간이 필요합니다.

칩 아키텍처와 Redis가 무슨 관련이 있는지 궁금하실 겁니다. 그 답은 POWER8 아키텍처에서 지원하는 CAPI(Coherent Accelerator Processor Interface)에 있습니다.

IBM의 CAPI 기술

많은 Redis 사용자가 관심을 가질만한 또 다른 새로운 혁신은 IBM에서 비롯되었습니다. IBM은 오랜 기간 Power 아키텍처를 개발 및 홍보해왔고, POWER8이 최신 시리즈입니다.

칩 아키텍처와 Redis가 무슨 관련이 있는지 궁금하실 겁니다. 그 답은 POWER8 아키텍처에서 지원하는 CAPI(Coherent Accelerator Processor Interface)에 있습니다.

기본 아이디어는 다음과 같습니다. Redis는 일반적으로 모든 데이터를 메모리에 저장합니다. 따라서 대용량 데이터 세트가 있는 경우 상당히 고가인 RAM을 확장하는 데 많은 투자를 해야 합니다. 최신 버전인 Redis 3.0에서는 이에 대한 대안으로 Redis 클러스터 생성을 선택할 수 있습니다. 하지만 머신이 여러 개 있어야 하므로 이 또한 큰 비용이 들 수 있습니다.

이상적인 솔루션은 저렴한 RAM이 탑재된 단일 머신을 사용하는 것이겠지만, 짧은 시일 내에 실현될 것 같지는 않습니다. RAM처럼 빠르지는 않지만 하드 디스크보다는 빠른 플래시 메모리가 있지만, 문제는 지연 시간입니다.

컴퓨터와 스토리지가 커뮤니케이션하는 일반적인 방법은 원하는 Redis 성능을 지원하기에는 너무 느립니다.

IBM의 CAPI는 외부 하위 시스템에 프로세싱을 오프로드할 수 있게 해주는 인터페이스를 제공합니다. 이러한 오프로드 프로세싱은 컴퓨팅 집약적 알고리즘의 실제 프로세싱을 보완하는 데 사용하거나 고속 스토리지에 사용할 수 있습니다. 본 사용 사례에서 CAPI는 CPU가 외부 플래시 스토리지 시스템과 직접 상호 작용하게 해줌으로써 지연 시간을 줄이고 이를 통해 응답성 면에서 플래시를 전형적인 스토리지 시스템보다 RAM과 더 비슷하게 만들어 줍니다. RAM처럼 빠르지는 않지만, 상당히 빠르고 너무 비싸지 않은 플래시는 합리적인 절충안이 될 수 있습니다.

IBM 엔지니어가 배포한 "NoSQL용 데이터 엔진 – IBM Power Systems Edition" 백서에 따르면, Power Systems는 NoSQL용 특수 API를 제공합니다. 다양한 유형의 NoSQL 데이터베이스가 있다는 것을 감안할 때 제가 "NoSQL API"에 회의적이었다는 것을 인정해야 할 것 같습니다. 백서를 보면 IBM은 Redis와 같은 키-값 스토어를 지원하는 특수 API를 제공합니다. API는 현재 MongoDB를 비롯한 다른 NoSQL 데이터베이스는 지원하지 않습니다. 하지만 IBM 직원과의 대화에서 IBM이 추가 지원에 관심이 있다는 것을 확인했고, 향후에 다른 NoSQL 오퍼링을 기대해도 좋을 것 같습니다.

이러한 새로운 기능은 상당히 하드웨어에 특화되어 있으며, 기본적인(아웃오브박스) Redis 설치에서는 이를 활용하기 어렵습니다. 따라서 IBM은 고성능 Redis 호스팅 및 소프트웨어를 제공하는 회사인 Redis Labs와 파트너십을 맺고 이러한 하드웨어에서 작동하는 Redis 특별 버전을 만들었습니다.

하지만 최소한 저에게 가장 흥미로운 부분은 제공된 시스템이 RAM과 플래시의 사용 비율을 사용자에게 맡긴다는 것입니다.

Redis Labs의 창립자 겸 CTO인 Yiftach Shoolman은 프로그래머의 관점에서는 Redis Labs이 구현한 API와 표준 Redis API 간에 차이가 없다고 말했습니다.

기본 아이디어는 IBM 하드웨어를 Redis Labs 소프트웨어와 함께 사용하면 빠른 속도의 대용량 Redis 인스턴스를 생성할 수 있다는 것입니다. 하지만 최소한 저에게 가장 흥미로운 부분은 제공된 시스템이 RAM과 플래시의 사용 비율을 사용자에게 맡긴다는 것입니다. 사용자는 IBM 시스템을 RAM을 많이 사용하고 플래시는 적게 사용하도록 구성하여 비싸지만 매우 빠른 시스템으로 만들 수 있습니다. 이와 반대로 플래시를 많이 사용하고 RAM을 아주 적게 사용할 수도 있습니다. 최종 균형은 사용자의 몫입니다. 웹 기반 제어판을 통해 POWER8 시스템에서 RAM과 플래시의 비율을 구성하면 됩니다. IBM에서 제공한 백서에 따르면 스토리지의 80%가 플래시일 때 사용자는 최적의 성능을 누릴 수 있다고 합니다. 제어판에서 간편하게 RAM과 플래시 비율을 변경할 수 있으므로, IBM 스토리지 솔루션의 많은 고객 및 잠재 고객이 성능과 비용 간의 최적점을 찾기 위해 다양한 조합을 테스트해 볼 것으로 예상합니다.

CAPI가 적용된 Redis 사용

IBM은 Geek Guide 작성을 위해 제가 POWER8 CAPI 인터페이스와 Redis를 사용하는 시스템에 액세스할 수 있게 해주었습니다. 구체적으로 다음 3개의 머신에 액세스할 수 있었습니다.

- Red Hat Enterprise 6.6 구동 Linux 가상 머신.
  이 머신은 PHP, Apache와 함께 Redis Enterprise 서버를 모니터링하고 구성하는 데 사용할 수 있는 웹 애플리케이션을 실행하고 있습니다.

- Ubuntu 14.10 구동 POWER8 시스템으로 20 코어와 256GB 메모리 탑재. Redis Enterprise Edition이 실행되고 있는 시스템 상에 있습니다. 이 시스템에는 플래시 스토리지 유닛에 연결된 CAPI 어댑터가 포함되어 있습니다.

- 세 번째 시스템은 IBM FlashSystem 840였습니다. 자체 IP 주소가 있고 웹 애플리케이션을 통해 모니터링할 수 있지만, 스토리지 시스템이므로 운영 체제를 실행하지 않습니다. 이 부분이 우리가 직접적으로 관심이 있는 부분입니다.

상기 구성은 조직에서 IBM의 플래시 솔루션을 사용할 때 예상할 수 있는 구성의 작은 복제판입니다. 물론 이러한 고성능 Redis 솔루션을 사용하려는 조직은 웹 서버를 하나가 아니라 여러 대 사용할 것입니다. Redis가 서버를 제공한다는 것을 고려하면, 위의 간단한 구성과 조직에서 사용할 만한 구성의 유일한 차이는 서버의 수입니다. 각 서버는 적절한 Redis 클라이언트를 갖추고 플래시 서버로 연결될 것입니다.

IBM 시스템에서 실행하는 버전은 일반 오픈 소스 Redis 버전이 아니라 Redis Labs의 시스템입니다. 구성만 다를 뿐이므로 클라이언트 측면에서 보면 동작은 동일해야 합니다. 하지만 Redis Labs 시스템은 Redis 2.8.12를 실행하므로 Redis 3.0.0 기능은 사용할 수 없습니다.

위에서 설명했듯이 일반적인 Redis 시스템은 단일 프로세스를 실행하지만, Redis Labs 시스템은 여러 개의 별도 프로세스를 사용합니다. 각 Redis 인스턴스에는 자체 구성 파일이 포함되어 있어 인스턴스가 시작될 때 이를 참조하게 됩니다. 하지만 이 구성 파일을 직접 변경하는 것이 아니라, 구성 파일 변경에 백엔드 API를 사용하는 웹 기반 GUI를 통해 변경해야 합니다. 제가 사용한 시스템에는 2개 서로 다른 유형의 Redis 서버가 실행되고 있었습니다. 하나는 "redis-server", 다른 하나는 "redis-server-big"이라고 부르고 있어, Redis 인스턴스가 RAM과 플래시 중 어디에서 실행되고 있는지 알 수 있었습니다. 웹 기반 제어판을 사용하여 각 메모리 유형에서 실행되는 Redis 서버 인스턴스를 추가하거나 제거할 수 있었습니다.

Redis Labs가 제공하는 웹 제어판은 Redis 설치를 손쉽게 모니터링하고 구성할 수 있게 해줍니다. 특히 다중 머신 클러스터로 작업할 때 간편합니다. 웹 제어판은 구성 기능을 한곳으로 모아서 로드 밸런싱 필요에 따라 Redis 서버 인스턴스를 추가 및 제거할 수 있습니다. 텍스트 기반 구성 파일에 익숙한 저로서는 이러한 파일과 Redis Labs Enterprise 클러스터에서 사용하는 구성 제어판 간의 매핑을 이해하는 데 시간이 좀 걸렸습니다. 하지만 Redis 인스턴스를 추가하고 제거하는 작업이 이보다 간단할 수는 없을 것입니다.

Redis Labs 제어판은 모니터링 기능도 제공합니다.

또한, RAM과 플래시간의 비율을 변경하는 것도 제어판의 플래시 시스템에서 슬라이더를 움직이기만 하면 됩니다. 따라서 슬라이더를 움직여서 시스템이 어떻게 동작하는지 보면서, 여러 가지 다른 구성을 테스트해 볼 수 있습니다.

RAM/플래시 분할 구성은 하나는 RAM용으로 다른 하나는 플래시용으로 2개의 Redis 인스턴스를 구성한다는 뜻입니다. Redis 클라이언트는 단일 포트에 연결되며, 어느 백엔드를 사용하는지 스토리지는 어디 있는지 클라이언트는 알지 못합니다. 구성된 각 Redis 시스템(RAM 및 플래시)은 Redis 지속성(persistence) 모델(AOF 또는 스냅샷) 중 하나를 쓰도록 별도로 설정할 수 있습니다. 스냅샷 지속성의 경우, Redis 데이터베이스의 콘텐츠를 디스크에 저장하는 빈도를 지정할 수 있습니다.

Redis Labs 제어판은 모니터링 기능도 제공합니다. 2개의 노드(RAM 및 플래시)가 수행하는 초당 작업 수를 볼 수 있으며, 이를 최근 몇 분, 하루 또는 1년으로 나누어 볼 수도 있습니다. 이를 통해 Redis 머신을 지속적으로 모니터링하면서 RAM과 플래시 비율을 재조정해야 하는지를 파악할 수 있습니다.

여러분에게 필요한 기술인가요?

저는 하드웨어보다는 소프트웨어에 더 익숙하다는 걸 말씀드려야 할 것 같습니다. 하지만 새로운 Power 지원 Redis 시스템 개발에 참여한 몇몇 IBM 엔지니어와 이야기를 나누면서 깊은 인상을 받

았습니다. 기본적으로 I/O 시스템을 짧은 지연 시간의 고대역폭 커뮤니케이션 채널에 연결한다는 것은 고속 I/O를 위해 RAM을 사용하지 않아도 된다는 뜻입니다.

향후에 오픈 소스 버전의 Redis에서도 이러한 아키텍처를 활용할 수 있을지 또는 데이터베이스(SQL 또는 NoSQL)에서는 활용할 수 있을지 IBM 엔지니어에게 물어보았습니다. 상용 Redis Labs 구현이 CAPI 연결을 적용한 첫 번째 시스템이고, 향후에 다른 시스템에도 적용할 예정이라고 합니다. Redis보다 Memcached를 선호하는 사용자를 위해 Redis Labs에서 Power Systems용 Memcached 버전도 지원합니다(어떤 장점이 있는지 잘 모르겠지만). 현재 상용 Redis Labs 구현은 이러한 기술을 활용할 수 있는 첫 번째 시스템입니다.

이 시점에서 과연 고려할 만한 가치가 있느냐는 질문을 던지게 됩니다. 제가 보고 검토한 바로는 충분히 고려할 만한 가치가 있으며, 균형 유지는 비즈니스 의사 결정이 될 것입니다. 만약 설치된 Redis 규모가 너무 커서 고속 솔루션을 찾아야 하는데 RAM을 고려하기엔 비용 부담이 된다면, Redis 클러스터와 IBM 오퍼링 중에 선택해야 합니다. 여러 대의 서버를 구매하고 관리하는 것보다 단일 서버인 IBM 제품과 Redis Labs 상용 라이선스를 구매하는 것이 더 유리합니다. IBM이 제공한 정보에 따르면, POWER 기반 Redis 스토어는 스토리지 크기가 증가하면서 더 큰 ROI를 제공하여 40TB 플래시 스토리지 시스템의 경우 최대 3배까지 절약할 수 있습니다.

또한, IBM 제품을 사용하면 여러 대의 서버 및 스토리지 시스템을 단일 서버 및 단일 스토리지 유닛으로 교체하여 유지 관리가 간편하고 랙 공간과 전력을 적게 소비하게 됩니다.

Redis 시스템을 위해 여러 개의 대형 스토리지 시스템을 운영하고 있다면, IBM의 새로운 제품을 살펴보고 비교해볼 만한 가치가 있습니다.

Redis 클러스터와 IBM 오퍼링은 출시된 지 얼마 되지 않아 아직 많은 조직에서 사용하고 있지 않습니다. 시장에서 두 시스템이 어떻게 성장해가는지 지켜보는 것도 흥미로울 것입니다. 대규모로 Redis 시스템을 운영하는 조직 중 IBM과 Redis Labs의 솔루션을 사용하여 시간과 비용을 절약하는 조직이 생길 것으로 예상합니다.

결론

Redis는 오랫동안 인기 있고, 안정적이며, 기능이 다양하고, 속도가 빠른 키-값 스토어로 성장해왔습니다. 현재 사용하는 NoSQL 솔루션 중 가장 인기 있는 솔루션이라고 해도 과언이 아닙니다. 본 안내서에서 설명했듯이, Redis의 데이터 유형은 다양한 기능을 제공하고, 인 메모리 속도와 영구 온 디스크 스토리지의 조합은 타의 추종을 불허합니다. 그러나 Redis 사용이 증가하면서 기업에서 점점 더 큰 용량의 데이터 세트에 이를 사용하게 되었고, 한계에 부딪히게 되었습니다. 마스터-슬레이브는 여전히 모든 데이터가 단일 컴퓨터의 메모리에 유지될 수 있다고 가정합니다. 하지만 Redis 클러스터는 새로운 솔루션으로 여러 대의 머신을 구성하고 유지 관리해야 합니다. IBM과 Redis Labs의 새로운 오퍼링은 대규모 Redis 시스템을 운영하는 조직에 작은 공간, 간편한 관리 및 쉬운 구성을 약속합니다. RAM/플래시 조합과 유용한 웹 기반 GUI를 통해 가능한 일입니다. 현재 대규모 Redis 시스템을 운영하고 있고 어떻게 확장을 계속할 수 있을지 고민이라면, IBM의 새로운 제품을 고려해볼 것을 권장합니다.■