키-값 데이터베이스에서 마스터 부하 조정을 위한 마스터-슬레이브-슬레이브 복제 모델 설계

Design of Master-Slave-Slave Replication Model to Balance Master Overhead for Key-value Database

저자
(Authors)

송경태, 박상현
Kyung-Tae Song, Sang-Hyun Park

출처
(Source)

한국정보기술학회논문지 15(2), 2017.2, 7-14 (8 pages)
Journal of Korean Institute of Information Technology 15(2), 2017.2, 7-14 (8 pages)

발행처
(Publisher)

한국정보기술학회
Korean Institute of Information Technology

URL

http://www.dbpia.co.kr/Article/NODE07111265

APA Style

송경태, 박상현 (2017). 키-값 데이터베이스에서 마스터 부하 조정을 위한 마스터-슬레이브-슬레이브 복제 모델 설계. 한국정보기술학회논문지, 15(2), 7-14.

이용정보
(Accessed)

연세대학교
165.***.121.254
2017/03/16 18:14 (KST)

저작권 안내

DBpia에서 제공되는 모든 저작물의 저작권은 원저작자에게 있으며, 누리미디어는 각 저작물의 내용을 보증하거나 책임을 지지 않습니다. 그리고 DBpia에서 제공되는 저작물은 DBpia와 구독계약을 체결한 기관소속 이용자 혹은 해당 저작물의 개별 구매자가 비영리적으로만 이용할 수 있습니다. 그러므로 이에 위반하여 DBpia에서 제공되는 저작물을 복제, 전송 등의 방법으로 무단 이용하는 경우 관련 법령에 따라 민, 형사상의 책임을 질 수 있습니다.

Copyright Information

Copyright of all literary works provided by DBpia belongs to the copyright holder(s)and Nurimedia does not guarantee contents of the literary work or assume responsibility for the same. In addition, the literary works provided by DBpia may only be used by the users affiliated to the institutions which executed a subscription agreement with DBpia or the individual purchasers of the literary work(s)for non-commercial purposes. Therefore, any person who illegally uses the literary works provided by DBpia by means of reproduction or transmission shall assume civil and criminal responsibility according to applicable laws and regulations.

Journal of KIIT. Vol. 15, No. 2, pp. 7-14, Feb. 28, 2017. pISSN 1598-8619, eISSN 2093-7571 7
http://dx.doi.org/10.14801/jkiit.2017.15.2.7

키-값 데이터베이스에서 마스터 부하 조정을 위한 마스터-슬레이브-슬레이브 복제 모델 설계

송경태* 박상현**

Design of Master-Slave-Slave Replication Model to Balance Master Overhead for Key-value Database

Kyung-Tae Song*, Sang-Hyun Park**

이 논문은 2015년도 정부(미래창조과학부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임 (NRF-2015R1A2A1A05001845).

요 약

키-값 데이터베이스 클러스터는 슬레이브 수가 늘어남에 따라 마스터의 부하가 증가하는 문제가 있다. 본 논문은 인메모리 키-값 데이터베이스 상에서 슬레이브 추가에 따른 마스터 성능의 부하 정도를 실험하고, 이를 해결할 수 있는 새로운 방안을 고안하였다. 고안한 방법은 상대적으로 처리 속도의 중요성이 덜한 슬레이브에서 마스터의 부하를 감당할 수 있도록 한다. 이는 기존의 마스터-슬레이브 구조가 아닌 마스터-슬레이브-슬레이브 복제 모델을 사용하여 가능하게 하였다. 본 논문은 실험에서 슬레이브 수를 1대에서 4대까지 변화시켜가며 새로운 방법의 효용성을 검증하였다. 실험 결과, 고안한 방법은 슬레이브 수가 4대인 경우를 기준으로 기존 방법보다 마스터의 성능을 약 300%까지 향상시켰다.

Abstract

In key-value database cluster, there is a problem that the overhead of the master increases as the number of slaves increases. This paper experiments the overhead of the master performance by adding new servers and proposes a new method to solve the problem. The proposed method delegates the load of the master to a slave which is relatively less important in processing speed. This can be done by using master-slave-slave replication model instead of the normal master-slave structure. In this paper, we verified the effectiveness of the new method with experiments by changing the number of slaves from 1 to 4. As a result, the devised method improved the performance of the master by about 300% compared to the original method when the number of slaves is 4.

Keywords
distributed database, load balancing, key-value database, load balancing, network latency, big data

* 연세대학교 컴퓨터과학과
** 연세대학교 컴퓨터과학과 교수(교신저자)
· 접 수 일: 2017년 01월 12일
· 수정완료일: 2017년 02월 12일
· 게재확정일: 2017년 02월 15일

Received: Jan. 12, 2017, Revised: Feb. 12, 2017, Accepted: Feb. 15, 2017
Corresponding Author: Sang-Hyun Park
Dept of Computer Science, Yonsei University, Sinchon-dong, Seodaemun-gu Seoul, 120-749, Korea
Tel.: +82-2-2123-5714, Email: sanghyun@cs.yonsei.ac.kr

연세대학교 | IP: 165.***.121.254 | Accessed 2017/03/16 18:14(KST)

Ⅰ. 서 론

관계형 데이터베이스는 다양한 기능을 제공하고 정형화된 데이터를 처리하는데 초점을 두고 있다. 그러나, SNS, 웹 서비스, 모바일 기기의 발전으로 빅데이터라고 불리는 다량의 비 구조적 데이터를 빠르게 처리할 수 있는 새로운 데이터베이스가 필요하게 되었다. 이를 위해 기존의 관계형 데이터베이스에서 부족했던 확장성을 지원하는 것이 필수적이었고, NoSQL이라고도 불리는 새롭게 등장한 데이터베이스들은 대부분 확장성을 지원한다. NoSQL 데이터베이스는 여러 종류가 있다. 키-값(Key-value) 데이터베이스, 문서형 데이터베이스, 컬럼형(Column-oriented) 데이터베이스, 그래프 데이터베이스 등이 그것이다. 이 중 키-값 데이터베이스는 기능성은 떨어지나 이식성이 좋고 성능이 뛰어나 여러 분야에서 사용된다. 키-값 데이터베이스는 인메모리(In-memory)에서도 쓰일 수 있고 디스크 기반으로도 쓰일 수 있다. 본 논문에서는 인메모리 키-값 데이터베이스 중 가장 널리 쓰이는 레디스(Redis)를 사용하여 클러스터(Cluster) 환경에서 슬레이브(Slave) 노드 수 증가가 마스터(Master)의 성능에 영향을 미치는 것을 확인 하였고, 이를 해결하기 위한 방법을 제시한다.

레디스는 다양한 데이터 타입을 지원하고, 클러스터 기능 및 복제 기능을 포함하여 여러 가지 기능을 제공한다. 이 중 클러스터 기능은 데이터베이스 시스템 구조에서 확장성을 가능하게 한다.

그림 1은 3대의 마스터 서버와 각각 1대의 슬레이브 서버가 연결되어 있는 클러스터 구조를 나타낸다. 레디스는 데이터를 저장할 때 키-값 형태의 해시 구조를 사용하여 저장한다. 해시 값은 최대 16383이고, 클러스터를 만들 때는 마스터 서버의 개수에 따라 각 마스터가 담당할 해시 슬롯(Hash Slot) 범위가 결정된다. 그림 1과 같이 마스터 서버가 3대라면 각 해시 슬롯에서 담당하는 해시 슬롯의 수는 약 5500개 이다. 그림 1에서 보이듯이 마스터 서버는 나머지 마스터들과도 연결되어 있고, 슬레이브 서버와도 연결되어 있어서 슬레이브 서버에 비해 상대적으로 부하가 크다. 또한, 이러한 구조에서 슬레이브 서버 수가 증가할 때마다 마스터가 전달해야 하는 데이터도 증가하여 마스터의 부하가 증가한다. 마스터의 부하는 전체 클러스터 시스템의 성능에 영향을 미치고, 병목현상이 발생할 수 있다. 그러므로 마스터의 부하를 줄이면 클러스터 환경의 성능을 향상 시킬 수 있다.

마스터의 부하를 줄이려는 대부분의 기존 연구들은 마스터 간의 데이터 부하를 균일하게 하는 것에 초점을 두었다[1]-[4]. 이러한 연구들은 데이터 편향을 막고 마스터들 사이에 균일한 데이터 분배를 이용하여 병목현상을 없애고 성능을 향상시켰으나, 슬레이브가 늘어남에 따라 발생하는 네트워크 부하 및 추가 작업에 따라 나타나는 성능 저하는 해결하지 못하였다.

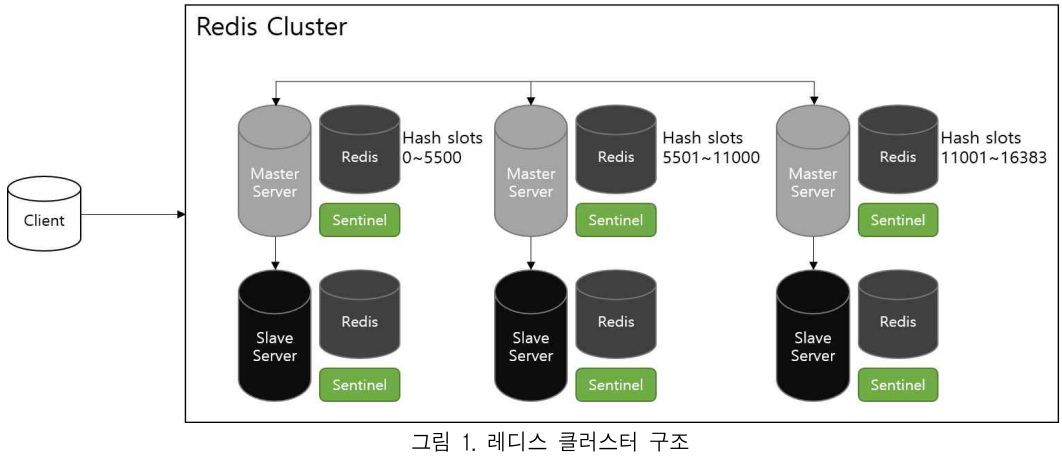

그림 1. 레디스 클러스터 구조
Fig 1. Redis cluster architecture

연세대학교 | IP: 165.***.121.254 | Accessed 2017/03/16 18:14(KST)

클러스터 환경에서 고가용성을 위해 슬레이브 서버를 추가하는 경우는 빈번하나 이로 인한 성능 저하를 해결하려는 연구는 아직 미미한 실정이다. 따라서 본 논문에서는 슬레이브 서버가 늘어남에 따라 발생하는 성능 저하를 측정하였고, 이를 해결하기 위하여 자동 마스터-슬레이브-슬레이브 복제 모델(MSSRM, Master-Slave-Slave Replication Model)을 제안한다. MSSRM은 클러스터에 새로운 슬레이브를 추가할 때 레디스의 장애극복 시스템인 레디스 센티널(Sentinel)을 이용하여 마스터의 부하를 분산시킨다. 마스터의 부하가 큰 경우 새로운 슬레이브가 마스터의 데이터를 복제하는 것이 아니라 자동으로 기존 슬레이브의 데이터를 복제하도록 한다. 이 과정은 레디스 센티널에서 주도하는데, 그 이유는 레디스 센티널에서 레디스 클러스터 전체의 마스터와 슬레이브에 대한 메타정보를 갖고 있기 때문이다.

본 논문의 기여도는 다음과 같다. 실험을 통하여 슬레이브 서버로 인한 마스터 서버의 비효율을 문제제기 하였고, 이를 위한 해결 방안을 제시하여 그 효용성에 대하여 기술한다.

본 논문의 구성은 다음과 같다. 제 2장에서 레디스 시스템의 구조적 특징을 기술하고 제 3장에서 기존 방법의 성능평가 및 문제점을 기술한다. 또한, 제시한 문제점을 해결하기 위해 MSSRM을 제안한다. 고안한 방법은 레디스 센티널을 이용하여 슬레이브에게 부하를 분산시키는 방법으로써, 시뮬레이션 성능 평가를 진행하였다. 마지막으로 제 4장에서 향후 발전 방향을 기술하고 결론을 맺는다.

II. 레디스 시스템의 구조적 특징

2.1 인메모리 키-값 데이터베이스

인메모리 키-값 데이터베이스 분야는 빠른 속도와 좋은 이식성을 제공하기에 다양한 분야에서 쓰이고 있고, 관련 기능을 향상시킬 수 있는 많은 연구들이 이루어지고 있다[5]-[8]. 인메모리 키-값 데이터베이스의 대표적인 플랫폼은 레디스와 Memcached, Aerospike가 있다[9]-[11]. 이 중 가장 먼저 등장한 것은 Memcached이다. Memcached는 이름과 같이 데이터베이스를 위한 메모리 캐시로써 등장하였고, 분산환경에서 빠른 데이터 처리를 보여주기에 트위터(Twitter), 페이스북(Facebook) 등 많은 서비스에서 Memcached를 사용한 바 있다[12][13]. 레디스는 기존의 Memcached보다 뛰어난 기능성을 보여 사용자들이 많이 사용하는 데이터베이스다. 마이크로소프트(Microsoft), 아마존(Amazon)의 트위치(Twitch) 서비스 등에서 사용 중 이고, 오픈소스 커뮤니티에서도 활발하게 개발되고 있다.

2.2 레디스 복제

레디스는 고가용성(High Availability)를 위해 마스터-슬레이브 복제를 사용한다. 복제 구조는 마스터에 장애가 생겼을 시 슬레이브가 마스터의 역할을 대신 하도록 하여 장애 상황에서도 서비스가 중단되지 않도록 한다. 슬레이브는 환경설정 파일의 slaveof를 이용하여 마스터의 주소와 포트를 입력하고, 레디스 서버를 실행하면 해당 마스터로 연결을 시도한다. 이 때 첫 연결 시 마스터는 마스터의 현재 데이터를 스냅샷(Snapshot)하는 RDB 파일을 만들어 슬레이브와 동기화를 진행한다. RDB는 레디스 DB의 약자로, 레디스의 물리적 로깅방식으로 저장되는 로그이다. RDB 파일은 기본적으로 디스크에 저장된 후 슬레이브로 전송되나 현재 레디스에서 디스크 저장 없이 전송하는 방법도 제공한다. 그러나 후자의 방법은 아직 불안정한 상태이기에 주로 디스크 저장을 한 후 전송하는 방법을 사용한다. 슬레이브는 전송 받은 RDB 파일을 이용하여 데이터베이스를 구축한다. 첫 연결 이후 마스터는 데이터베이스의 변경이 일어나는 명령을 처리할 때마다 슬레이브로 해당 명령을 전파한다. 슬레이브는 해당 명령을 받아 실행하여 마스터와 동일한 데이터를 유지한다.

슬레이브 또한 재귀적으로 또 다른 슬레이브의 마스터 역할을 할 수 있다. 이 때 기존 슬레이브에 연결된 또 다른 슬레이브를 2차 슬레이브라 칭한다. 만약 서비스 운영 중 마스터 노드에 문제가 생기면 1차 슬레이브가 마스터의 역할을 대신한다.

연세대학교 | IP: 165.***.121.254 | Accessed 2017/03/16 18:14(KST)

2.3 레디스 센티널

레디스 센티널은 장애 복구를 관리하는 조정자 역할의 어플리케이션이다. 레디스 센티널은 사용자로부터 센티널 명령어를 입력 받을 수 있고, 대다수의 명령은 센티널 내부에서 자체적으로 사용된다. 센티널은 현재 작동중인 모든 마스터와 슬레이브가 정상적으로 작동하는지 감시한다. 만약 마스터 서버에 장애가 발생하면 슬레이브의 권한을 마스터로 변경하여 서비스가 유지되도록 한다. 또한, 이전의 마스터 권한을 슬레이브로 변경하여 재 시작했을 때 슬레이브의 역할을 할 수 있도록 조정한다. 이러한 절차는 센티널이 레디스 서버의 환경설정 변경 명령을 내리는 방식으로 실행된다. 센티널 프로세스 또한 장애가 발생할 수 있기에 레디스를 운영할 때 여러 개의 센티널이 레디스 서버들을 감시하도록 한다. 이러한 다수의 센티널들이 판단을 내려 어떠한 슬레이브가 마스터로 승격 될 지 결정한다. 사용자가 설정한 정족수(Quorum)의 센티널이 특정 슬레이브 서버를 마스터로써 적합하다고 판단하게 되면 해당 슬레이브가 마스터로 승격된다. 레디스에서 권장하는 최소 센티널은 3대이고, 최소 정족수는 2이다.

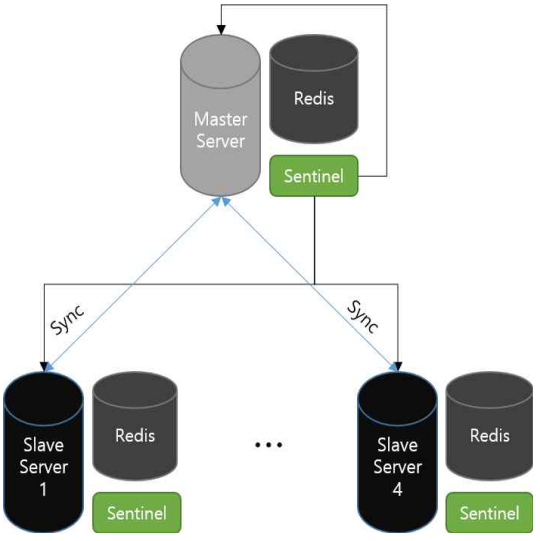

그림 2. 센티널을 이용한 레디스 마스터-슬레이브 장애복구 관리
Fig. 2. Master-slave failover managing with sentinel

그림 2는 하나의 마스터 서버에 4개의 슬레이브 서버가 연결되어 있는 구조를 나타낸다. 각각의 슬레이브들은 마스터 서버와 연결되어 동기화를 수행하며, 동기화가 끝난 이후 마스터에 들어오는 명령들을 동일하게 전달 받는다. 전달된 명령들은 레디스 인스턴스에 입력되어 데이터베이스에 저장된다. 그림에서 레디스 센티널은 각 서버에 존재하고 있는데, 레디스 프로세스 없이 센티널만 단독으로도 존재할 수 있다. 그림 2와 같이 센티널은 서버들의 메타정보를 활용하여 클러스터를 모니터링한다. 슬레이브의 각 센티널들 또한 다른 서버들에 대한 메타 정보를 알고 있지만, 그림의 직관성을 위해 표시하지 않았다. 레디스 센티널은 각 서버들의 메타정보를 이용하여 장애복구 시 서버들의 설정을 변경할 수 있다. 특히, 'slave of'라는 옵션은 레디스 서버가 특정 서버의 슬레이브로 등록되는 설정이다.

III. 마스터 부하 조정을 위한 MSSRM

본 절은 레디스 복제 시스템에서 슬레이브 수에 따른 마스터의 성능부하를 측정하기 위한 성능평가 도구 및 실험에 대해 설명한다. 이 후 실험을 통하여 슬레이브 수에 따른 마스터의 성능부하 정도를 보여준다. 또한, 마스터의 성능부하를 해결하기 위한 방법으로 슬레이브에서 마스터의 부하를 감당하게 하는 MSSRM을 제안한다.

3.1 슬레이브 수에 따른 마스터 성능부하 측정

마스터의 성능평가를 수행하기 위한 구조는 그림 2와 같다. 하나의 마스터 서버에 레디스 인스턴스와 센티널이 작동하고 있고, 마찬가지로 레디스 인스턴스와 센티널이 작동하고 있는 슬레이브 서버들을 1대에서 4대까지 연결하여 성능을 측정하였다. 성능평가에 이용한 하드웨어 및 소프트웨어 설정은 표 1과 같다. 실험에 사용한 서버 모두 동일한 하드웨어 및 소프트웨어 설정을 하였으며, 클러스터의 서버들은 1Gbps의 로컬 네트워크로 연결하였다. 실험에서 사용한 성능평가 도구는 memtier benchmark이다[14]. Memtier benchmark는 레디스 메인 개발자를

연세대학교 | IP: 165.***.121.254 | Accessed 2017/03/16 18:14(KST)

포함하여 여러 레디스 개발자들이 소속되어 있는 회사인 레디스 랩스(Redis Labs)에서 개발한 레디스 성능평가 도구이다[15]. Memtier benchmark는 파라미터 설정을 통해 성능평가 방법을 조절 할 수 있으며, 실험에서 사용한 파라미터 설정은 표 2와 같다.

실험에서 키 값의 크기를 유지하기 위해 키의 최소값(10,000,000)과 최대값(99,999,999)을 설정하였다. 본 실험에서는 쓰기 작업에 대해서만 실험을 진행하였다. 읽기 작업은 마스터 서버에서만 실행되고, 슬레이브 서버에게 전달하지 않기 때문에 슬레이브 수에 따른 지연 시간을 측정할 수 없기 때문이다.

표 1. 실험 환경
Table 1. Experimental environment

하드웨어 및 소프트웨어	종류 및 성능
OS	Linux Centos 7
Redis version	3.9.102
RAM	64GB
CPU	Intel i7 6700K
Storage	HDD WD 1TB Caviar Blue
Network	1Gbps Ethernet

표 2. Memtier benchmark 환경 설정
Table 2. Memtier benchmark environmental settings

파라미터(parameters)	값(value)
쓰레드 당 명령어 수 (requests)	10000
클라이언트 수 (clients)	1
쓰레드 수 (threads)	10
데이터 값 사이즈 (data size)	10KB
키 최소값 (key-minimum)	10000000
키 최대값 (key-maximum)	99999999
쓰기/읽기 비율 (ratio)	1:0

표 3. 슬레이브 수에 따른 마스터의 성능 평가
Table 3. Performance evaluation of the master according to the number of slaves

성능 지표 / 슬레이브 수	시간당 명령어 처리량	지연 속도 (msec)	시간당 데이터 처리량(MB/s)
1	9735	1.022	95.5
2	5618	1.774	55.1
3	3759	2.652	36.9
4	2821	3.537	27.7

표 3은 마스터에 연결된 슬레이브 수를 변화 시켜가며 Memtier benchmark를 수행한 결과이다. 각 실험에서 benchmark 설정은 표 2로 동일하게 적용하였다. 또한, 마스터의 성능 지표로써 시간당 명령어 처리량, 명령어에 대한 지연 시간, 시간당 데이터 처리량을 확인하였다.

표 3의 결과에서 슬레이브 수가 늘어남에 따라 마스터의 성능이 저하되는 것을 볼 수 있다. 이는 그림 5, 그림 6, 그림 7에서 'Master-Slave'로 표기되어 있는 항목으로도 확인할 수 있다. 특히, 슬레이브 수가 4인 경우 마스터의 성능 지표가 모두 약 350% 안 좋아지는 것을 볼 수 있다. 위의 실험 결과에서 확인 할 수 있듯이, 슬레이브 수가 증가함에 따라 마스터의 부하가 급격하게 증가함을 알 수 있다. 즉, 클러스터 전체의 성능을 저하시킨다고 할 수 있다. 이에 본 논문은 슬레이브 수 증가에 따른 마스터 부하를 감소시키기 위해 MSSRM을 제안한다.

3.2 MSSRM 및 시뮬레이션

MSSRM은 레디스 센티널에서 'slaveof' 옵션을 변경할 수 있다는 것에서 착안하였다. 마스터의 부하가 큰 경우 새로운 슬레이브를 기존 슬레이브의 2차 슬레이브로 연결하도록 한다. 즉, 새로운 슬레이브가 연결되었을 때 마스터-슬레이브-슬레이브(MSS, Master-Slave-Slave) 구조를 갖는다. MSS 구조와 그 절차는 그림 4와 같다. 전체적인 구조는 기존 클러스터 구조와 동일하다.

MSSRM은 센티널을 통해 관리된다. 기존 클러스터 복제 구조에서 새로운 슬레이브가 마스터에 접근하면 센티널에서 이를 인지한다. 마스터에 직접 연결된 슬레이브 서버가 너무 많아 부하가 크다고 판단하면 새로운 슬레이브 서버의 연결을 취소하고, 센티널에서 새로운 슬레이브 서버의 'slaveof' 설정을 변경한다. 센티널은 새로운 서버의 'slaveof' 값을 기존 1차 슬레이브로 변경하고, 결과적으로 새로운 서버는 2차 슬레이브가 되어 1차 슬레이브에게서 데이터를 전송 받는다. MSS 복제 방법은 클러스터의 전체적인 가용성과 내구성을 해치지 않으면서도 마스터의 부하를 줄였다는 점에서 기존 복제 방법보다 우수한 방법이라고 할 수 있다. 물론 기존

연세대학교 | IP: 165.***.121.254 | Accessed 2017/03/16 18:14(KST)

복제 방법에서도 2차 슬레이브를 설정 할 수 있지만 사용자의 판단에 의존해야 한다. 사용자의 판단에만 의존하는 방법은 슬레이브가 모두 마스터에게만 할당되었을 때 지속적으로 성능 저하가 발생할 수 있다. 또한, 이러한 성능 저하가 어디서 일어나는지 알기 힘들다. MSSRM은 자동으로 이루어지기 때문에 이러한 문제점을 막을 수 있다. MSSRM에서는 레디스 센티널에서 1차 슬레이브 적합 여부를 판단하여 부적합 시 2차 슬레이브로 할당한다는 점에서 차이가 있다. 1차 슬레이브 적합 여부는 마스터의 성능 저하 발생 정도에 따른다.

MSSRM의 효용성을 검증하기 위해 3.1 절에서 수행했던 방법과 동일한 설정을 이용하여 실험을 진행하였다. 설정은 마스터-슬레이브 구조에 2차 슬레이브로 슬레이브들을 연결하는 방법을 취하여 진행하였다. 성능평가로써 활용한 지표들 또한 3.1절과 동일하다. 실험은 시뮬레이션 형태로 진행되었으며, 시뮬레이션 결과는 그림 5, 그림 6, 그림 7에서 표현하였다.

그림 5-7에서 나타나 있듯 MSSRM을 사용했을 경우 슬레이브 증가에 따른 마스터 부하는 나타나지 않았다. 슬레이브 수에 따른 성능 차이는 무시할 수 있을 정도의 수준이었다. 그래프에서도 보이듯이 슬레이브 수가 늘어날수록 기존 방법과 MSSRM의 성능 차이가 증가하였고, 5대 이상의 슬레이브가 있을 경우에도 성능 차이가 증가할 것으로 예측된다.

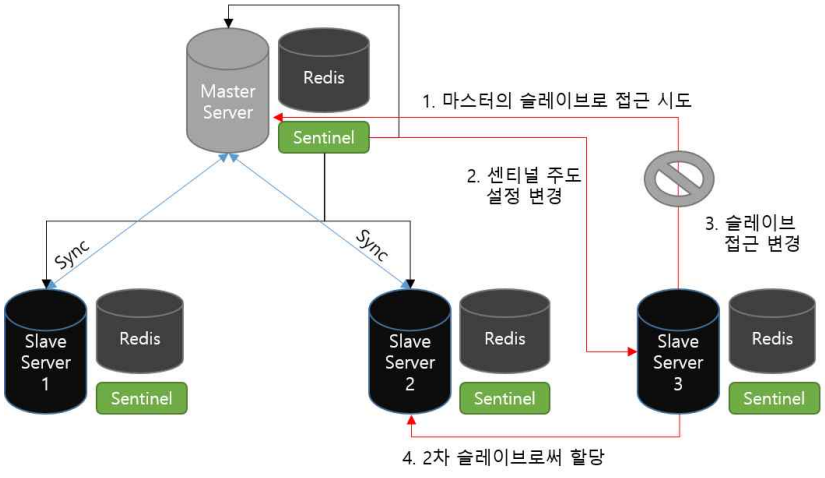

그림 4. 마스터-슬레이브-슬레이브 구조와 방법
Fig. 4. Master-slave-slave architecture and method

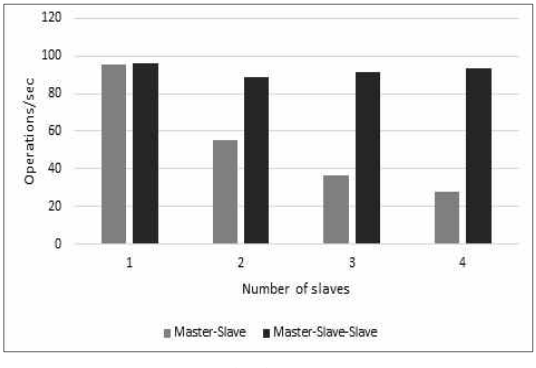

그림 5. 시간 당 명령어 처리량
Fig. 5. Processed operations per second

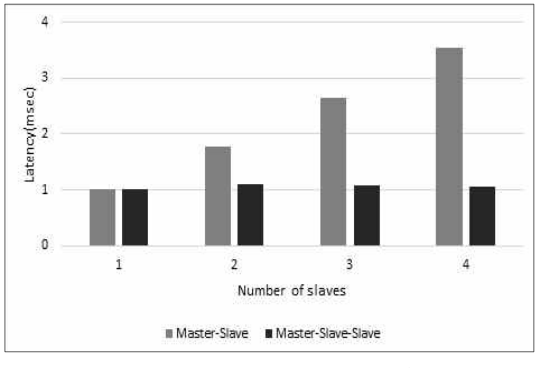

그림 6. 명령어 처리 지연 시간
Fig. 6. Latency for processing operations

연세대학교 | IP: 165.***.121.254 | Accessed 2017/03/16 18:14(KST)

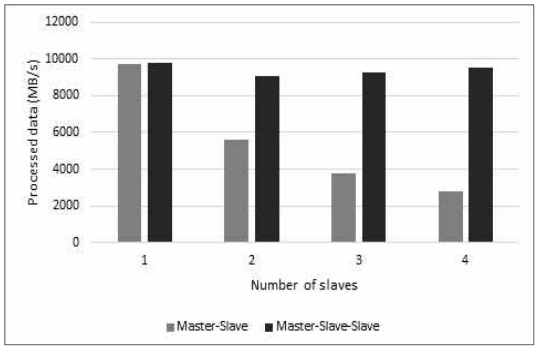

그림 7. 시간 당 데이터 처리량
Fig. 7. Processed data per second

IV. 결론 및 향후 과제

본 논문에서는 기존 레디스의 마스터-슬레이브 복제 방법의 성능평가를 분석하고 이에 대한 문제점을 발견하였다. 문제점은 슬레이브 수가 증가함에 따라 마스터의 성능 저하가 발생한 다는 것이었다. 이러한 문제는 클러스터 시스템이 중요시 되는 빅데이터 환경에서 전체 시스템의 성능 저하로 연결될 수 있기에 치명적이다. 이에 본 논문은 발견한 문제점을 해결하기 위해 마스터-슬레이브-슬레이브 복제 모델(MSSRM)을 제안하였으며 그 효용성을 검증하기 위해 시뮬레이션을 통한 기존 복제 모델과의 비교실험을 진행하였다. 실험은 레디스의 성능평가 도구인 memtier benchmark를 사용하였으며, 하나의 마스터에 슬레이브를 1대에서 4대까지 연결하여 비교하였다. 실험 결과, 기존 레디스의 복제 방법은 슬레이브 수가 늘어남에 따라 마스터의 성능이 저하되었고, 본 논문에서 제시한 MSSRM의 시뮬레이션 결과에서는 마스터 성능의 저하가 발생하지 않았다. MSSRM은 2차 슬레이브를 이용하기 때문에 2차 슬레이브 입장에서 마스터와 동기화가 늦어질 수 있다는 단점이 있다. 향후 연구에서 이와 관련된 실험 및 개선 방법을 연구할 예정이다. 또한, 레디스와 레디스 센티널에서 MSSRM을 자동으로 작동할 수 있도록 구현하고, 10G 네트워크를 적용한 실험과 비교를 진행할 예정이다.

References

[1] G. Cybenko, "Dynamic load balancing for distributed memory multiprocessors", Journal of parallel and distributed computing, Vol. 7, No. 2, pp. 279-301, Oct. 1989.

[2] Baran, E. Mesut, and Wu. Felix, "Network reconfiguration in distribution systems for loss reduction and load balancing", IEEE Transactions on Power Delivery, Vol. 4, No. 2, pp. 1401-1407, Aug. 2002.

[3] V. Cardellini, M. Colajanni, and S. Philip "Dynamic load balancing on web-server systems", IEEE Internet computing, Vol. 3, No. 3, pp. 28-39, May 1999.

[4] Q. Ye, B. Rong, Y. Chen, M. Al-Shalash, C. Caramanis, and J. G. Andrews, "User association for load balancing in heterogeneous cellular networks", IEEE Transactions on Wireless Communications, Vol. 12, No. 6, pp. 2706-2716, Apr. 2013

[5] H. Lim, D. Han, D. G. Andersen, and M. Kaminsky, "MICA: a holistic approach to fast in-memory key-value storage", In 11th USENIX Symposium on Networked Systems Design and Implementation, pp. 429-444, Apr. 2014

[6] Mitchell, Christopher, Y. Geng, and J. Li, "Using One-Sided RDMA Reads to Build a Fast, CPU-Ef ficient Key-Value Store", Presented as part of the 2013 USENIX Annual Technical Conference (USENIX ATC 13), pp. 103-114, Jun. 2013.

[7] S. Li, H. Lim, V. W. Lee, J. H. Ahn, A. Kalia, M. Kaminsky, D. Andersen, O. seongil, S. Lee, and P. Dubey, "Architecting to achieve a billion requests per second throughput on a single key-value store server platform", Proceedings of the 42nd Annual International Symposium on Computer Architecture, Vol. 43, No. 3, pp. 476-488, Jun. 2015.

[8] Dojin Choi, Bosung Kim, Insoo Bae, Yoonsik Kwak, and Seokil Song, "Spark based Distributed Processing Method for Sharing Big Traffic Data in Realtime", Journal of KIIT, Vol. 14, No .2, 107-114, Feb. 2016.

연세대학교 | IP: 165.***.121.254 | Accessed 2017/03/16 18:14(KST)

[9] Redis, https://redis.io/ [Accessed: Jan. 11, 2017]
[10] Memcached, https://memcached.org/ [Accessed: Jan. 11, 2017]
[11] Aerospike, http://www.aerospike.com/ [Accessed: Jan. 11, 2017]
[12] P. Saab, "Scaling memcached at Facebook," https://www.facebook.com/note.php?note_id=39391378919&ref=mf [Accesed: Jan. 11, 2017]
[13] Twitter Engineering, "Memcached SPOF Mystery", https://blog.twitter.com/2010/memcached-spof-mystery [Accesed: Jan. 11, 2017]
[14] Memtier Benchmark, https://github.com/RedisLabs/memtier_benchmark [Accessed: Jan. 11, 2017]
[15] Redis labs, https://redislabs.com/ [Accessed: Jan. 11, 2017]

저자소개

송 경 태 (Kyung-Tae Song)

2015년 2월 : 연세대학교 컴퓨터과학과(공학학사)
2015년 3월 ~ 현재 : 연세대학교 컴퓨터과학과(공학석사)
관심분야 : 빅데이터, 키-값 데이터베이스, 데이터베이스, SSD

박 상 현 (Sang-Hyun Park)

1989년 : 서울대학교 컴퓨터공학과 졸업(학사)
1991년 : 서울대학교 대학원 컴퓨터공학과 졸업(공학석사)
2001년 : UCLA 대학원 컴퓨터과학과 졸업(공학박사)
1991년 ~ 1996년 : 대우통신 연구원
2001년 ~ 2002년 : IBM T. J. Watson Research Center Post-Doctoral Fellow
2002년 ~ 2003년 : 포항공과대학교 컴퓨터공학과 조교수
2003년 ~ 2006년 : 연세대학교 컴퓨터과학과 조교수
2006년 ~ 2011년 : 연세대학교 컴퓨터과학과 부교수
2011년 ~ 현재 : 연세대학교 컴퓨터과학과 교수
관심분야 : 데이터베이스, 데이터마이닝, 바이오인포매틱스, 적응적 저장장치 시스템, 플래시 메모리 인덱스, SSD

연세대학교 | IP: 165.***.121.254 | Accessed 2017/03/16 18:14(KST)