질의 처리율 가속화를 위한 멀티 CSD 기반 문서형 데이터베이스 설계

유형선[1], 김기환[1], 유정현[1], 변홍수[1], 이명철[2], 최진춘[2], 김영재[1]
[1]서강대학교 컴퓨터공학과, [2]한국전자통신연구원 스마트데이터연구실
{hsyoo, lewis461, jhryu, byhs, youkim}@sogang.ac.kr, {mclee, jcchoi}@etri.re.kr

Designing a Multi-CSD-based Document Database to Improve Query Throughput

Hyungsun Yoo[1], Kihwan Kim[1], Junghyun Ryu[1], Hongsu Byun[1], Myungcheol Lee[2], Jinchun Choi[2], Youngjae Kim[1]
[1]Sogang University and [2]ETRI

요 약

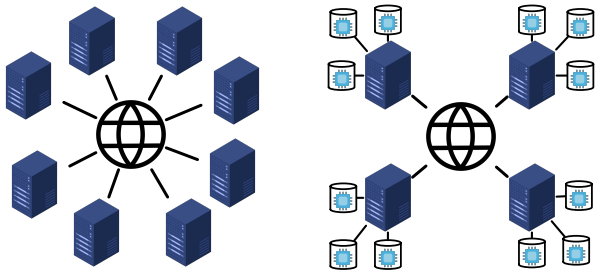
기존 분산데이터베이스는 네트워크 병목, 일관성 유지를 위한 오버헤드 등의 제약으로 Scale-out 확장성의 한계에 직면한다. 본 연구는 최근 활발하게 연구되고 있는 계산 가능한 저장 장치인 Computational Storage Drive(CSD)와 대표적인 문서형 데이터베이스인 CouchDB 내의 기능을 활용한 멀티 CSD 기반 문서형 데이터베이스 설계를 바탕으로 데이터베이스의 Scale-out의 한계를 극복하여 성능 향상을 목표로 한다. CSD 장치 내의 샤드 개수 할당 최적화를 통해 장치 내 병렬성을 확대시킨 결과, 질의 처리 속도가 기존 Host 시스템에 비해 약 1.27배 향상되었다. 또한 기존 CouchDB는 장치 간 성능 차이를 고려하지 않고 데이터를 균등하게 분배하기 때문에 분산 처리의 이점을 활용할 수 없는 한계가 존재하였으나, Host 장치와 CSD 장치의 성능 차이에 따른 부하 분산 가중치를 부여하여 기존 Host 시스템의 처리 속도에 비해 약 1.97배 향상을 기대할 수 있다.

1. 서론

In-Storage Processing (ISP)은 연산을 호스트 CPU가 아닌 스토리지 내에서 연산을 수행하여 데이터 처리 가속화가 가능하다. 최근 ISP 기술이 탑재된 상업적 Computational Storage Drive (CSD)가 출시되었으며, 대표적으로 FPGA 가속기가 장착되거나 ASIC 기반 프로세서를 사용하여 운영체제를 구동하는 방식으로 나뉜다. FPGA 가속기를 탑재한 CSD를 사용하기 위해서는 낮은 수준의 커널 프로그램이 필요한 반면, 운영체제를 구동하는 CSD는 기존 리눅스 서버와 완전히 동일한 환경을 제공하고 TCP/IP over NVMe를 통한 호스트와 통신으로 유연한 프로그래밍이 가능하다. 한편, 방대한 양의 트랜잭션에 대응이 가능한 Cassandra [1], PolarDB [2]와 같은 분산데이터베이스는 클러스터를 구성하는 노드를 추가하여 성능 Scale-out이 가능하지만 네트워크 병목, 일관성 유지를 위한 오버헤드와 같은 제약으로 인해 확장성 한계에 직면한다.

본 연구에서는 데이터베이스의 Scale-out 확장성 한계를 극복하기 위해 단일 노드 내부의 추가자원인 CSD를 활용하여 각 노드 데이터베이스의 성능 Scale-up을 목표로 한다. 이를 위해 호스트와 멀티 CSD기반 클러스터 형태의 데이터베이스 설계를 제안하고, 대표적인 문서형 데이터베이스인 CouchDB를 사용하여 프로토타입 하였다. 본 연구에서 제안한 멀티 CSD 기반 데이터베이스는 호스트와 CSD의 성능 차이를 고려한 부하 분산 가중치 알고리즘을 적용하여 기존 호스트 시스템에서 작동하는 데이터베이스에 비하여 1.97배 높은 처리량을 보였다.

본 연구는 2021년도 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원(No.2021-0-00136, 다양한 산업 분야 활용성 증대를 위한 대규모/대용량 블록체인 데이터 고확장성 분산 저장 기술 개발)과 2024년 과학기술정보통신부 및 정보통신기획평가원의 SW중심대학사업 지원을 받아 수행되었음(2024-0-00043)



(a) 전통적인 분산 DB (b) CSD 활용 클러스터 DB

그림 1: 클러스터를 활용한 데이터베이스 구성 방법 비교

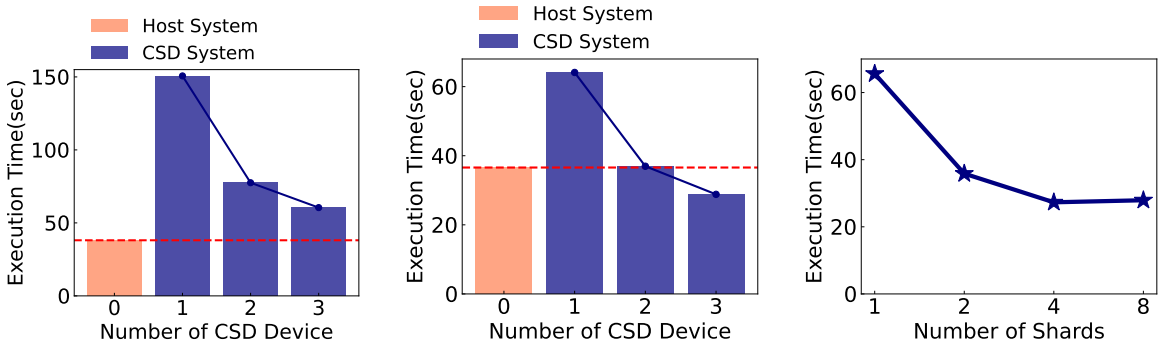
2. 배경 지식 및 연구 동기

2.1 In Storage Processing & Computational Storage Drive

ISP(In-Storage Processing) 기술은 저장장치 내 자원을 활용해서 대용량 데이터에 대한 작업을 처리하는 최신 기법이다. ISP는 에너지 효율성을 증대시킬 뿐만 아니라 호스트와 저장장치 간 데이터 이동량을 감소시킨다 [3]. ISP 기법을 구현한 대표적인 저장장치로는 Computational Storage Drive (CSD)가 있다. CSD는 저장장치 내에 연산 능력을 부여하여 데이터 처리 성능을 향상시킨다. 삼성전자의 SmartSSD [4], NGD Systems의 Newport CSD [5], ScaleFlux의 Computational Storage [6] 등 다수의 저장장치 제조사에서 상업적 CSD를 출시한 바 있다. CSD는 운영체제 탑재 가능 여부에 따라 두 가지로 분류된다. FPGA 가속기를 활용하여 사전에 정의된 작업을 수행하는 SmartSSD [4]가 존재하고, ARM 프로세서를 탑재하여 독립적인 운영체제를 구동하는 Newport CSD [5]가 존재한다. 특히 Newport CSD는 유연한 프로그래밍을 통해 다양한 작업을 수행하여 ISP의 장점을 극대화할 수 있다.

2.2 CouchDB

Apache 소프트웨어 재단에서 개발한 오픈 소스 NoSQL 데이터베이스 CouchDB [7]는 분산 환경에서 효율적으로 데이터를



(a) (b) (c)

그림 2: (a) 노드당 1개의 샤드가 할당된 Host 시스템과 CSD 시스템의 수행 시간 비교, (b) 노드당 4개의 샤드가 할당된 Host 시스템과 CSD 시스템의 수행 시간 비교, (c) 단일 CSD 저장 장치 내의 샤드 개수별 수행시간 변화 추이

저장하고, 관리하기 위해 고안되었다. CouchDB에서 데이터베이스는 JSON 형식의 데이터를 문서 단위로 저장하고, HTTP 프로토콜을 통한 RESTful API를 이용하여 문서의 추가, 편집, 삭제를 지원한다. CouchDB는 다중 버전 동시성 제어(Multi Version Concurrency Control)를 통해 읽기 작업 중 쓰기 요청으로 인해 발생할 수 있는 Locking 문제를 방지하고, 데이터 일관성을 유지한다.

Cluster Mode: CouchDB의 클러스터 모드는 데이터베이스의 확장성과 가용성을 향상시키기 위해 여러 노드를 연결하여 하나의 클러스터를 형성한다. 시스템을 분산 환경에서 확장 가능하게 운영할 수 있도록 도와주는 기능을 제공하며 여러 노드를 추가하여 시스템의 용량을 확장할 수 있다.

Sharding & Replication: CouchDB는 클러스터 모드에서 데이터를 여러 개의 조각으로 나누어 저장하는 샤딩(Sharding)과 데이터 손실 방지 및 특정 노드 장애에 대처하기 위해 복제(Replica) 기법을 이용한다. 데이터베이스 생성 시 매개 변수를 통해 샤드와 복제본의 수를 지정할 수 있다. 각 샤드는 독립적인 키 범위를 가지며, 문서의 고유 ID의 해시값에 따라 속하는 샤드로 할당된다. 분할된 샤드에 대해 지정된 복제본의 개수에 따라 여러 노드에 샤드 복제본을 저장하여 데이터 손실이 발생 시 복제본을 참조하여 데이터 일관성을 유지한다.

2.3 연구 동기

기존에 고확장성 및 분산 처리를 위한 클러스터 데이터베이스 설계로서 Cassandra [1], HBase [8], Aurora [9], PolarDB [2], AlloyDB [10] 등이 제안된 바 있다. 이러한 분산 스토리지 시스템에서는 노드 간 네트워크 연결로 인해 오버헤드가 발생하며 스위칭, 라우팅 등으로 인한 네트워크 지연 시간이 필연적으로 동반된다 [11]. 또한 네트워크 트래픽 관리를 위해 추가적인 CPU 싸이클을 소모한다 [12].

본 연구에서는 네트워크로 연결된 전통적인 확장 방식이 아닌 단일 노드 내에서의 계산 성능 확장과 노드 내 소규모 sub-cluster 설계를 통한 데이터베이스 가속화 가능성을 탐구하고자 한다. 그림 1 은 전통적인 분산 DB 클러스터와 CSD를 활용한 클러스터의 차이를 나타낸다. CSD를 활용한 구조를 채택하게 될 경우 단일 노드 내의 데이터베이스의 성능 Scale-up을 통한 가속화로 기존과 같은 성능을 보장하면서 네트워크를 경유하는 노드 수를 줄일 수 있다.

3. 실험 및 평가

3.1 실험 환경 설정

본 연구는 Newport CSD 3개가 장착된 서버를 사용하여 시스템을 구성하였다. 상세한 스펙은 표 1과 표 2에 나타나있다. 본 연구에서 평가한 시스템 설정은 다음과 같다.

- Host 시스템(Host Only) : 호스트 서버의 저장장치와 CPU를 사용하여 단일 CouchDB 인스턴스를 실행하였다.
- CSD 시스템(Host & CSD Cluster) : 호스트 서버와 서버에 부착된 3개의 CSD를 사용하여 클러스터 모드로 CouchDB를 실행하였다.

기본적으로, Filtering 질의는 128KB 크기의 문서를 100,000개 가지고 있는 데이터베이스에 대해 수행하였다.

3.2 샤딩을 통해 병렬성을 확대한 CSD 시스템 성능 평가

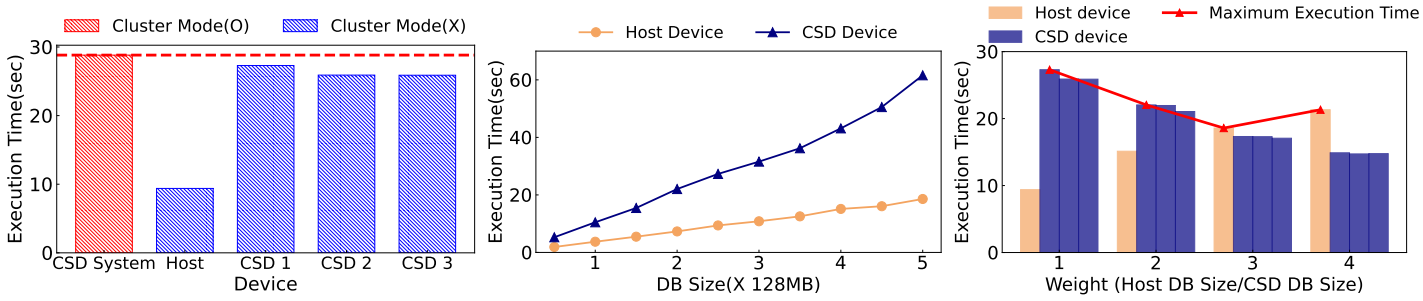
그림 2(a)는 한 개의 질의에 대해 Host 시스템과 Cluster mode를 이용하여 노드 당 1 샤드가 배정된 CSD 시스템 간의 처리속도 차이를 보여준다. CSD 시스템에서 가용한 CSD 장치의 수가 증가할수록 처리속도가 증가하지만, Host 시스템의 처리속도에는 미치지 못하는 것을 확인할 수 있다. 이는 CSD 내의 1 샤드 할당으로, 병렬성을 충분히 활용하지 못한 이유 때문이며, 가용한 CSD 장치의 수가 증가하더라도 처리속도가 선형적으로 증가하

표 1: 호스트 서버 상세 설명

CPU	AMD EPYC™ 7352, 48 Cores (96 Threads), 2.3GHz (Up to 3.2GHz)
Socket	2 NUMA Node
Memory	256 GB (64 GB × 4) DRAM (DDR4)
OS	Centos 7.92.2009 (Core) / Linux Kernel 4.14
Storage	Dell 1.92 TB SSD SAS 24Gbps ISE RI 512e

표 2: Newport CSD 상세 설명

Storage Capacity	8 TB	
Host Interface	Single Port PCIe Gen3x4 (U.2)	
NAND Flash Memory	Toshiba Flash Memory	
In-Storage Processing Engine	ARM Cortex-A53, 1.0 GHz, 4 Cores	
	Memory	8 GB DRAM (DDR4)
	OS	Linux Kernel 4.14



(a) (b) (c)

그림 3: (a) Cluster mode를 사용한 CSD 시스템의 수행 시간과 사용하지 않은 CSD 시스템의 장치별 수행 시간 비교, (b) 데이터베이스 크기에 따른 장치별 수행 시간 변화 추이, (c) 부하 분산이 적용된 CSD 시스템의 수행 시간 예상 수치

지 않는다. 그림 2(c)는 CSD 장치에 25,000개의 문서를 가진 데이터베이스를 할당하였을 때, 샤드 개수에 따른 질의 처리 속도 변화를 보여준다. CSD 장치 내의 할당된 샤드의 개수가 증가할수록 노드 내 병렬성이 확대되어 처리 속도가 증가한다. 하지만, 샤드 개수가 4개에서 8개로 증가되었을 때 수행 시간의 감소가 이뤄지지 않는다는 점을 확인할 수 있고, 이는 노드 내 병렬성의 한계가 존재하여 성능 향상에 긍정적인 영향을 끼치는 샤드 개수의 최대 임계값이 존재함을 보인다. 그림 2(b)는 CSD 내 병렬성을 충분히 확보하기 위하여 샤드 개수를 4개로 설정하고, 가용한 CSD 장치의 수에 따른 처리 속도 변화를 보여준다. 가용한 CSD 장치의 수가 같을 때, 그림 2(b)는 그림 2(a)에 비해 각각 2.35배, 2.17배, 2.05배의 성능 향상을 보인다. 또한, 가용 CSD 장치의 수가 3개일 때, Host 시스템의 처리 속도보다 1.27배 향상되었다.

3.3 Computing Power에 따른 부하 분산을 적용한 CSD 시스템 성능 평가

그림 3(a)는 Cluster mode를 사용하지 않고, 장치 별로 25,000개의 문서로 구성되어 있는 데이터베이스를 관리할 때의 장치별 질의 수행 시간을 측정한 것이다. CSD 장치들 중 최대 수행 시간은 Host 장치의 수행 시간보다 약 2.9배 느리다. 이러한 장치 별 수행 시간의 차이는 Host 장치와 CSD 장치의 성능 차이가 존재함을 시사한다. 그림 3(b)는 데이터베이스 크기에 따른 수행 시간 변화를 보인다. 그래프의 기울기는 데이터베이스 크기 변화에 따른 질의 처리 수행 시간을 의미하는데, 각 지점에서의 기울기는 일관되게 Host 장치가 CSD 장치에 비해 약 3배 낮은 수치를 보인다. 또한, 그림 3(a)의 붉은색 점선은 총 100,000개의 문서로 구성되어 있는 데이터베이스에 대해 CSD 시스템의 질의 처리 시간을 표기한 것이다. 빨간색 점선과 CSD 장치들 중 최대 수행 시간의 차이를 비교하면, CSD 시스템의 질의 처리 시간은 가장 늦게 질의 처리를 완료한 저장장치에 의해 결정되는 것을 알 수 있다. 기존 Cluster mode는 이러한 장치 별 성능 차이를 고려하지 않고 데이터를 균등하게 분배하고 있으나, 부하 분산 가중치를 3으로 설정하여 데이터를 배치한다면 가장 느린 장치의 성능을 향상시켜 최적의 성능이 나올 것을 예측할 수 있다. 이에 따라, 본 연구에서는 Computing power를 고려한 부하 분산 알고리즘이 적용된 시나리오를 상정하여 실험을 진행한다. 그림 3(c)는 가중치 설정에 따른 질의 처리 수행 시간 예측값을 제공한다. 앞선 실험에서 예측하였듯, 가중치를 3으로 설정했을 때가 처리 성능의 최대이고, Host 시스템 대비 약 1.97배 성능 향상을 보였다.

4. 결론

본 연구에서는 분산 데이터베이스 클러스터에서 단일 노드 Scale-up 을 통한 가속화를 위해 CSD 활용 기법의 가능성을 탐구하였다. 샤딩에 대한 고려없이 단순히 3개의 CSD를 활용한 시스템은 Host 시스템의 성능에 미치지 못했으나, CSD 장치 내의 샤드 개수 할당 최적화를 통해 병렬성을 확대한 결과 CSD 시스템은 Host 시스템에 비해 약 1.27배 빠른 질의 처리 속도를 보였다. 다만, 여전히 데이터의 균등 분배로 인한 성능 향상의 한계를 발견했다. 본 연구에서는 Computing power 차이를 고려한 부하 분산 알고리즘이 적용된다면, CSD 시스템이 Host 시스템 대비 최대 1.97배 성능 향상이 가능함을 시사하는 예측값을 확인하였다.

참고 문헌

[1] Avinash Lakshman and Prashant Malik, "Cassandra - A Decentralized Structured Storage System," *ACM SIGOPS Operating Systems Review*, vol. 44, no. 2, 2010.

[2] WeiCao, ZhenjunLiu, PengWang, SenChen, CaifengZhu, SongZheng, Yuhui Wang, and Guoqing Ma, "Polarfs: An ultra-low latency and failure resilient distributed file system for shared storage cloud database," *Proceedings of the VLDB Endowment*, 2018.

[3] Rajeev Balasubramonian, Jichuan Chang, Troy Manning, Jaime H. Moreno, Richard Murphy, Ravi Nair, and Steven Swanson, "Near-Data Processing: Insights from a MICRO-46 Workshop," *IEEE Micro*, vol. 34, no. 4, pp. 36–42, 2014.

[4] Samsung Electronics, "SmartSSD." `https://semiconductor.samsung.com/us/ssd/smart-ssd/`, 2022.

[5] NGD Systems, "Newport CSD." `https://www.ngdsystems.com/solutions`, 2022.

[6] Scaleflux, "Scaleflux CSD." http://https://scaleflux.com/, 2022.

[7] Apache Foundation, "CouchDB." https://couchdb.apache.org, 2005.

[8] Mehul Nalin Vora, "Hadoop-HBase for Large-Scale Data," *International Conference on Computer Science and Network Technology*, 2011.

[9] Alexandre Verbitski, Anurag Gupta, Debanjan Saha, Murali Brahmadesam, Kamal Gupta, Raman Mittal, Sailesh Krishnamurthy, Sandor Maurice, Tengiz Kharatishvili, and Xiaofeng Bao, "Amazon aurora: Design considerations for high throughput cloud-native relational databases," *Proceedings of the 2017 ACM International Conference on Management of Data, SIGMOD Conference 2017*, 2017.

[10] Google Cloud, "AlloyDB for PostgreSQL under the hood: Intelligent, database-aware storage." https://cloud.google.com/blog/products/databases/alloydb-for-postgresql-intelligent-scalable-storage, 2022.

[11] M. GabrielHaas and ViktorLeis, "Exploiting directly-attached nvme arrays in DBMS," *10th Conference on Innovative Data Systems Research, CIDR 2020, Amsterdam, The Netherlands*, January 12-15, 2020.

[12] Qizhen Zhang, Philip A. Bernstein, Badrish Chandramouli, Jiasheng Hu and Yiming Zheng, "DDS: DPU-optimized Disaggregated Storage," *Proceedings of the VLDB Endowment*, vol. 17, no. 11, 2024.