스마트홈 환경에서 센서 데이터 처리율 향상을 위한 기계학습 기반 캐싱 시스템 설계

송진수*, 이필원*, 신용태**
*숭실대학교 컴퓨터학과
**숭실대학교 컴퓨터학부
iko153@soongsil.ac.kr, pwlee@soongsil.ac.kr, Shin@ssu.ac.kr

A Study on Machine Learning-Based Caching System for Improving Sensor Data Processing in Samrt Home Environment

Jin-Su Song*, Pil-Won Lee*, Yong-Tae Shin**
*Dept. of Computer Science, Soong-Sil University
**Dept. of Computer Science and Engineering, Soong-Sil University

요 약

최근 초연결화를 근간으로 한 스마트 홈 구성을 위해 스마트 홈 내부에 센서를 탑재한 디바이스가 증가하고 있으며, 이를 효과적으로 사용하기 위해 빅데이터 처리 시스템이 활발하게 도입되고 있다. 그러나 기존 빅데이터 처리 시스템은 분산노드에 할당되기 전 모든 요청이 클러스터 드라이버로 향하기 때문에 동시에 많은 요청이 발생하는 경우 분할 작업을 관리하는 클러스터 드라이버에 병목현상이 발생함에 따라 네트워크를 공유하는 클러스터 전체의 성능감소로 이어진다. 특히 작은 데이터 처리를 지속적으로 요청하는 스마트 홈 디바이스에서 지연율이 더 크게 나타난다. 이에 본 논문에서는 동시간에 빈번한 요청이 발생하는 스마트 홈 환경에서 효과적인 데이터 처리를 위한 기계학습 기반 캐싱 시스템을 설계하였다.

1. 서론

4차 산업혁명의 기술이 발전하며 인공지능, 사물인터넷, 클라우드 컴퓨팅, 빅데이터와 같은 정보통신기술과 공간의 융합에 관한 연구가 활발하게 진행되고 있다. 사물 인터넷과 인공지능이 접목된 주거시설인 스마트 홈 시스템은 집 내부의 모든 디바이스가 네트워크로 연결되어 거주자의 생활패턴에 맞춰 실내안전, 전력관리, 헬스케어 등 다양한 분야에서 도움을 준다[1]. 초연결화를 근간으로 한 스마트 홈 구성을 위해 스마트 홈 내부에 센서를 탑재한 디바이스가 증가하고 있으며, 각각의 디바이스는 실시간으로 방대한 양의 데이터를 생성한다[2]. 수집된 데이터는 빅데이터 처리 시스템을 통해 데이터의 연관관계를 도출하여 거주자의 편의성을 향상시키는 지능화된 스마트 홈 구축에 활용된다.

빅데이터 처리 시스템은 분산 클러스터 구조로 연결되어 있으며 실질적인 데이터 처리는 클러스터 드라이버를 통해 분산된 노드에서 이루어진다. 그러나 분산노드에 할당되기 전 모든 요청이 클러스터 드라이버로 향하기 때문에 동시에 많은 요청이 발생하는 경우 분할 작업을 관리하는 클러스터 드라이버에 병목현상이 발생함에 따라 네트워크를 공유하는 클러스터 전체의 성능감소로 이어진다. 특히 작은 데이터 처리를 지속적으로 요청하는 스마트 홈 디바이스에서 지연율이 더 크게 나타난다. 따라서 동시간에 빈번한 요청이 발생하는 스마트 홈 환경에서 효과적인 데이터 처리 시스템이 필요하다.

본 논문의 구성은 다음과 같다. 2장에서는 기존에 분산 클러스터 기반으로 사용 중인 인-메모리 빅데이터 처리 시스템과 NoSQL 중 인-메모리로 구성된 데이터베이스를 살펴보고 시계열 연관도 분석 기법을 살펴본다. 3장에서는 본 논문에서 제안하는 스마트 홈 환경에서 센서 데이터 처리율 향상을 위한 기계학습 기반 캐싱 시스템을 제시한다. 마지막 4장에서는 결론 및 향후 연구 과제를 제시한다.

2. 관련연구

2.1. 인-메모리 기반 분산 시스템 스파크

스파크는 빅 데이터 처리를 위한 인-메모리 기반의 대용량 데이터 고속처리 엔진을 보유하고 있는

시스템으로서, 기존의 디스크 방식 빅데이터 시스템에 비해 처리시간이 100배 이상 단축되어 실시간 빅데이터 처리에 보편적으로 사용되고 있다[3]. [그림 1]은 스파크의 구성이다.

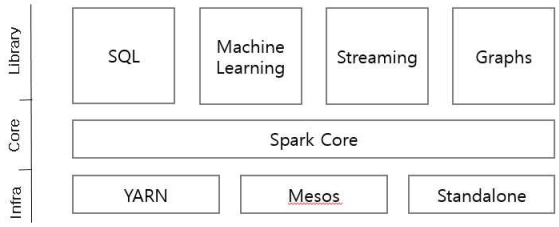

그림 1. 인-메모리 기반 분산 시스템 스파크
Fig 1. In-memory based distributed system spark

스파크는 인프라, 스파크 코어, 라이브러리로 구성되어 있다. 인프라는 스파크 실행 및 스케줄러 리소스 관리를 하고 그 위로 메모리 기반의 분산 클러스터 컴퓨팅 환경의 스파크 코어가 실행된다. 스파크는 빅데이터를 SQL로 처리하는 Spark SQL, 실시간 전송되는 데이터를 처리하는 Spark Streaming, 머신러닝을 위한 MLib, 그래프 데이터 프로세싱이 가능한 GraphX 라이브러리를 통해 실시간 데이터 처리를 용이하게 하고, 머신러닝 알고리즘을 활용할 수 있다[4,5] 스파크 클러스터의 구조는 크게 드라이버, 클러스터 매니저, 워커 노드로 구성되어 있다. [그림2]은 스파크의 클러스터 구조를 나타낸다.

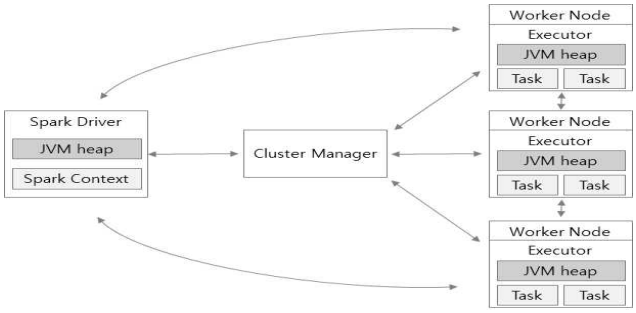

그림 2. 스파크 클러스터 구조
Fig 2. A spark structure diagram

스파크는 드라이버 프로세스와 다수의 워커 노드가 네트워크로 연결된 분산 처리 시스템이다. 드라이버 프로세스는 입력에 대한 응답, 워커 노드의 작업과 관련된 스케줄링 역할을 수행하고 워커 노드에서는 드라이버 프로세스가 할당한 작업을 수행한다.

클러스터 매니저는 스파크 컨텍스트와 워커 노드 사이에 중계자 역할을 하며 다수의 워커 노드가 작업을 공유할 수 있게 한다[5,6]. 워커노드는 요청된 작업을 수행하기 위해 다수의 태스크를 워커 노드에 분산하여 처리한다[4,5].

2.2. 연관도 분석 기법

머신러닝은 학습 시스템에 따라 지도학습, 비지도 학습, 강화학습 세 가지로 분류한다. 그 중 비지도 학습 알고리즘인 연관성 규칙은 다수의 품목 간의 관계를 수치화하여 연관 규칙을 찾아내는 기법이다. 기존의 데이터를 특별한 변형 없이 사용할 수 있어 다양한 분야에서 두 개 이상의 품목 간의 관련성과 규칙을 탐색하는데 활용된다. 연관규칙의 평가 기준은 지지도, 신뢰도, 향상도가 있다. 지지도는 항목 집합 X와 Y가 동시에 발생할 비율을 의미하며, 식(1)과 같이 정의된다.

$$Support_{(X \Rightarrow Y)} = (X \cap Y) \quad (1)$$

신뢰도는 항목집합 X가 포함된 비율 중 항목 집합 X와 Y가 동시에 포함된 비율을 의미하며, 식(2)와 같이 정의된다.

$$Confidence_{(X \Rightarrow Y)} = P(Y \mid X) = \frac{P(X \cap Y)}{P(X)} \quad (2)$$

향상도는 실제 발생 확률을 각 항목집합의 발생이 독립적일 경우 그 거래가 동시에 발생할 예상 기대확률로 나눈 것을 의미하며, 식(3)과 같이 정의된다.

$$Lift_{(X \Rightarrow Y)} = \frac{P(Y \mid X)}{P(Y)} = \frac{P(X \cap Y)}{P(X)P(Y)} \quad (3)$$

향상도의 값이 크면 두 항목이 동시에 발생한 확률이 예상확률보다 더 크므로 향상도의 값이 큰 경우에 의미 있는 규칙이다. 최소 지지도 값과 최소 신뢰도 값을 모두 만족하고 향상도의 값이 큰 경우 두 항목 집합의 규칙을 강한 연관규칙으로 판단한다[7].

2.3. 레디스

레디스는 오픈소스 프로젝트로 메모리 기반의 Key/Value 저장소다. 레디스는 인-메모리 데이터베이스로 일반 디스크 저장소를 사용하는 데이터베이스에 비해 빠른 속도로 데이터를 처리하며, 성능은

서버의 품질에 따라 다르나 평균 초당 2만~10만회를 수행한다. 레디스는 인-메모리 데이터베이스의 특성상 메모리상에서 데이터를 관리하며 일정 시간이 지나면 데이터의 손실을 막기 위해 메모리에 있는 데이터를 디스크 파일로 저장한다[8,9].

3. 제안하는 기계학습 기반 캐싱 시스템

제안하는 기계학습 기반 캐싱 시스템은 스파크의 단점을 보완하기 위해 빈번하게 사용되는 데이터와 연관도가 높은 데이터를 예측하여 센서와 가까운 캐시서버에 미리 저장한다. 캐시 서버에 사용되는 연관도 분석 알고리즘은 거주자의 위치, 사용된 센서, 시간을 분석하기 때문에 캐시 메모리의 세 가지 지역성인 시간 지역성, 공간 지역성, 순차 지역성을 모두 만족하며 캐시 서버의 활용율을 향상시킨다. 캐시 서버를 사용하면 빅 데이터 서버에 접근하는 횟수가 줄어들어 서버의 자원이 보다 효율적으로 사용되고 데이터 처리 성능이 향상된다.

3.1. 제안하는 기계학습 기반 캐싱 시스템 구성

제안하는 기계학습 기반 캐싱 시스템은 스마트 홈 디바이스를 기준으로 빅 데이터 서버와 캐싱 서버 두 가지로 구성된다. [그림3]은 제안하는 캐싱 시스템의 구조도이다.

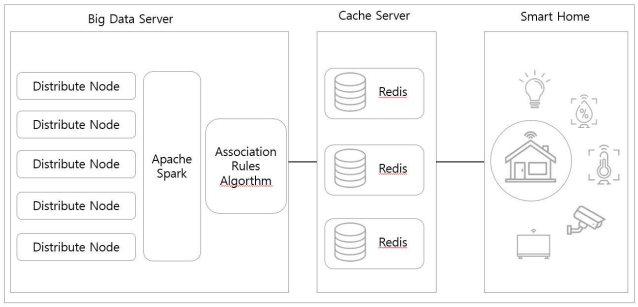

그림 3. 제안하는 기법의 구조도
Fig 3. A schematic diagram of the proposed technique

스마트 홈 디바이스는 사용자의 활동을 기록할 수 있는 다양한 센서를 탑재하고 사용자의 활동 및 특정 행동에 대한 지속시간을 측정한다. 빅 데이터 서버는 아파치 스파크를 사용하며 클러스터 드라이버와 다수의 분산노드로 구성되며 수집된 사용자 기록을 분산노드에 나눠서 저장한다. 빅데이터 서버에 저장된 사용자기록은 연관도 분석을 통해 항목별 연관도가 높은 순으로 캐시 서버에 저장한다.

3.2. 연관도 분석 알고리즘

수집된 데이터에서 센서 이름과 측정된 시간을 추출하여 시간에 따른 거주자의 위치와 비교하고 분석한다. 분석된 결과는 가장 연관도가 높은 장소 및 센서 순으로 정렬하여 반환한다. 다음 [그림4]은 캐시 데이터 적재를 위해 센서별 연관도를 구하는 알고리즘을 나타낸다.

기계 학습 기반 연관도 분석 알고리즘

```
# 최소 지지도, 연관 규칙 반환 함수
def association_rules(device_item, min_support):
# 지지도, 빈도수 계산
item_stats=freq(device_item).to_frame("freq")item_sta
ts['support']  = item_stats['freq'] /
device_count(device_item) * 100
# 최소 지지도보다 작은 품목은 제외
qualifying_items = item_stats[item_stats['support']
>= min_support].index order_item
=device_item[device_item.isin(qualifying_items)]
# 2개 미만의 정보는 제외
device_size= freq(device_item.index)
qualifying_device= device_size[device_size >=
2].index device_item=
device_item[device_item.index.isin(qualifying_device)]
# 빈도수, 지지도 계산
item_stats =
freq(device_item).to_frame("freq")item_stats['support'
]  = item_stats['freq'] / device_count(device_item) *
100
# 품목에 대한 generator를 생성
item_pair_gen=get_item_pairs(device_item)
# 품목집합의 빈도수, 지지도 계산
item_pairs=freq(item_pair_gen).to_frame("freqAB")
item_pairs['supportAB'] = item_pairs['freqAB'] /
len(qualifying_device) * 100
# 최소 지지도를 만족하지 못하는 품목 집합을 제외
item_pairs= item_pairs[item_pairs['supportAB'] >=
min_support]
# 계산된 연관 규칙을 지표와 함께 테이블로 생성
item_pairs =
item_pairs.reset_index().rename(columns={'level_0':
'item_A', 'level_1': 'item_B'}) item_pairs =
merge_item_stats(item_pairs, item_stats)
item_pairs['confidenceAtoB'] =
item_pairs['supportAB'] / item_pairs['supportA']
item_pairs['confidenceBtoA'] =
item_pairs['supportAB'] / item_pairs['supportB']
item_pairs['lift']              =
item_pairs['supportAB'] / (item_pairs['supportA'] *
item_pairs['supportB'])
# 향상도 정렬하여 결과를 반환
return item_pairs.sort_values('lift', ascending=False)
```

그림 4. 기계학습 기반 연관도 분석 알고리즘
Fig 4. Machine Learning-Based Association Algorithm

4. 결론

IoT 환경의 발전으로 네트워크로 연결된 다수의 센서에서 실시간으로 거주자의 활동정보를 수집하는

스마트 홈 환경은 효과적인 빅 데이터 처리 시스템을 활용하기 위해 다수의 센서에서 생성되는 데이터를 효율적으로 처리하는 것이 중요하다.

이에 본 논문에서는 스마트 홈 환경에서 센서 데이터 처리율 향상을 위한 기계학습 기반 캐싱 시스템을 제안하였다. 제안하는 시스템은 스파크와 레디스를 결합하여 구성하고 연관도 분석 알고리즘을 통해 캐시 서버에 데이터를 적재한다.

향후 기계학습 기반 캐싱 시스템의 구축을 통한 보다 구체적인 성능 분석이 필요하다.

Ackowledgement

"이 논문은 2017년도 정부(과학기술정보통신부)의 재원으로 정보통신기술진흥센터의 지원을 받아 수행된 연구임(No. 2017-0-00724, 셀룰러 기반 산업 자동화 시스템 구축을 위한 5G 성능 한계 극복 저지연, 고신뢰, 초연결 통합 핵심기술 개발)"

참고문헌

[1] M.Raisul Alam, M. B. I. Reaz, M.A. Mohd Ali, "A Review of Smart Homes – Past, Present, and Future,"IEEE Transactions on Systems Man and Cybernetice Part C (Applications and Reviews), 2012.

[2] Yongtak Yoon, Kyuchang Lee, Nathall Silva, Kijun Han. (2020). Adaptive Sensor Data Transmission Scheduling Scheme to Configure Samrt Home Network. KIISE Transactions on Computing Practices, 26(10), 458-462.

[3] Hyeopgeon Lee, Youngwoon Kim, Kiyoung Kim. (2019). Study of In-Memory based Hybrid Big Data Processing Scheme for Improve the Big Data Processing Rate. Journal of Korea institute of Information,Electronics, and Communication Technology, 12(2), 127-134.

[4] Keungyeup Ji, Youngmi Kwon. (2020). Performance Comparison of Python and Scala APIs in Spark Distribured Cluster Computing System. Korea Multimedia Society, 23(2), 241-246.

[5] IT World Glossary|Spark, http://www.itworld.co.kr/news/92835 (accessed April 9, 2015

[6] Real-world Python Workloads on Spark:Standalone Clusters, https://becominghuman.ai/real-world-python-workloads-on-sparkstandalone-clusters-2246346c7040 (accessed November 1, 2019).

[7] Heechang Park, Kwanghyun Cho. (2005). Waste Database Analysis Joined with Local Information Using Association Rules. ,Journal of The Korean Data Analysis Society 7(3), 763-772.

[8] J. M. Choi, D. W. Jeong, J. S. Yoon, and S.Lee, "Digital forensics investigation of redisdatabase," IPS Trans. Comp. and Commun.Sys., vol. 5, no. 5 pp. 117-126, May 2016.

[9] Bomin Seo, Byeongsoon Jang, Hyeungseok Oh, Hyunju Park. (2019). RESTful, Redis Vased API Server Platform Design for Automatic API Generation and Data Processing Performance. The Journal of Korean Institute of Communications and Information Sciences, 44(5), 895-903.