Journal of the Korean Data & Information Science Society
2015, **26**(5), 1035–1045

http://dx.doi.org/10.7465/jkdi.2015.26.5.1035
한국데이터정보과학회지

In-memory data grid 기술을 활용한 택시 애플리케이션 성능 향상 기법 연구[†]

최치환[1] · 김진혁[2] · 박민규[3] · 권가은[4] · 정승현[5] · 프란코 나자레노[6] · 조완섭[7]

[16]충북대학교 바이오정보기술학과 · [2]충북대학교 빅데이터학과 ·
[34]충북대학교 비즈니스데이터융합학과 · [5]충북대학교 정보산업공학과 · [7]충북대학교 경영정보학과

접수 2015년 8월 8일, 수정 2015년 9월 21일, 게재확정 2015년 9월 24일

요 약

최근 빅데이터 분야에서 데이터를 메모리에 적재 후 빠르게 처리하는 인메모리 컴퓨팅 기술이 새롭게 부각되고 있다. 인메모리 컴퓨팅 기술은 과거 대용량 메모리와 다중 프로세서를 탑재한 고성능 서버에 적용 가능하였지만, 점차 일반 컴퓨터를 초고속 네트워크로 연결하여 분산·병렬처리가 가능한 구조로 변화하고 있다. 본 논문은 In-memory data grid (IMDG) 기술을 택시 애플리케이션에 접목하여 기존의 데이터베이스의 변경 없이 성능을 향상시키는 기법을 제안한다. IMDG 기술을 적용한 경우 기존의 데이터베이스 기반의 웹서비스에 비해 처리속도와 처리량이 평균 6∼9배정도 증가하며, 또한 부하량에 따른 처리량 변화의 폭이 매우 작음을 확인 하였다.

주요용어: 데이터베이스, 인메모리 데이터 그리드, 인메모리 컴퓨팅.

1. 서론

인메모리 컴퓨팅 (in-memory computing; IMC)은 기존에 주로 하드디스크에 저장하던 데이터를 메모리에 저장후 처리하는 컴퓨팅 기술을 의미한다. 주요 IMC 기술은 DRAM, FLASH, SSD (solid state disk)와 같은 하드웨어 기술뿐만 아니라 In-Memory Database, In-memory data grid, In-memory analysis, Complex event processing, In-memory messaging처럼 데이터의 저장 및 관리, 실시간 분석, 메시징을 모두 아우르는 소프트웨어 기술로서의 애플리케이션, 프레임워크, 플랫폼 등을 모두 포함한다. Figure 1.1은 1990년도와 2010년도의 기술발전 속도를 나타낸다. 메모리 기술의 발전과 보급이 확대되면서 단위 소자 당 생산 비용이 지속적으로 감소하고 있음을 확인 할 수 있다. 18개월마다 반도체 칩의 밀도가 2배로 늘어난다는 무어의 법칙과 더불어 전력소모, 동작속도 또한 발전하였다. 하지만 하드디스크 기술의 성장은 다른 하드웨어 기술 발전 속도에 비해 상대적으로 매우 뒤쳐져있다.

[†] 본 연구는 미래창조과학부 및 정보통신기술진흥센터의 대학ICT연구센터육성 지원사업의 연구결과로 수행되었음 (IITP-2015-H8501-15-1013).
[1] (362-763) 충청북도 청주시 서원구 충대로1, 충북대학교 바이오정보기술학과, 박사과정.
[2] (362-763) 충청북도 청주시 서원구 충대로1, 충북대학교 빅데이터학과, 석사과정.
[3] (362-763) 충청북도 청주시 서원구 충대로1, 충북대학교 비즈니스데이터융합학과, 석사과정.
[4] (362-763) 충청북도 청주시 서원구 충대로1, 충북대학교 비즈니스데이터융합학과, 석사과정.
[5] (362-763) 충청북도 청주시 서원구 충대로1, 충북대학교 정보산업공학과, 박사과정.
[6] (362-763) 충청북도 청주시 서원구 충대로1, 충북대학교 바이오정보기술학과, 박사과정.
[7] 교신저자: (362-763) 충청북도 청주시 서원구 충대로1, 충북대학교 경영정보학과.
E-mail: wscho@chungbuk.ac.kr

이러한 이유로 전반적인 시스템의 성능을 높이기 위해서는 CPU와 메모리 업그레이드가 우선적인 고려 대상이 되었다.

본 연구에서는 IMDG 기술을 활용한 택시 애플리케이션 성능 향상 기법을 제안한다. 일반적으로 IMDG 기술은 네트워크로 연결된 일반 컴퓨터를 이용하여, 하나의 컴퓨터에서 처리하기 어려운 데이터를 분산·병렬처리는 것이다. 또한 노드 증가에 따라 처리속도 및 처리량이 비례하여 증가하므로 데이터의 실시간 저장 및 분석에 적합한 기술이다. 택시 애플리케이션은 데이터의 실시간 저장 및 처리에 매우 민감하므로 IMDG기술을 적용하여 성능을 높이는 방법이 요구된다. 특히, 위치정보가 실시간으로 변하는 택시 데이터의 특성상 검색방법에 따라 검색속도, 처리량 차이가 상이함으로 적절한 검색방법을 찾는 것이 매우 중요하다.

본 논문의 구성은 다음과 같다. 제 2절에서 IMC 기술에 관하여 설명한다. 제 3절에서는 택시 애플리케이션 성능 향상 기법을 제안한다. 제 4절에서는 제안된 기법에 대하여 성능을 평가한다. 마지막으로 5절에서 결론을 맺는다.

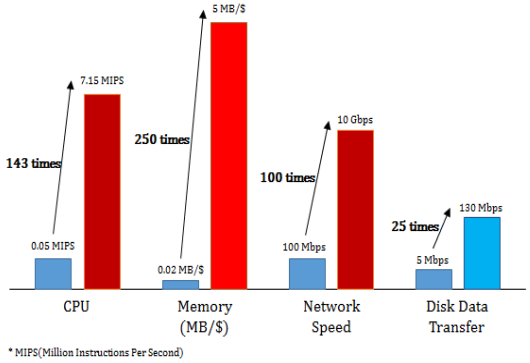

Figure 1.1 Growth rate comparison of technology between in 1990 and in 2010

2. 관련 연구

택시 애플리케이션은 동시에 많은 사람이 온라인에 접속하는 환경으로 높은 동시 수용성 및 실시간 처리를 필요로 한다. 2절에서는 높은 동시 수용성 및 실시간 처리를 위한 인메모리 컴퓨팅 관련 기술과 연구에 대하여 설명한다.

2.1. Big data and in-memory computing

빅데이터는 데이터의 용량 (volume), 생산 속도 (velocity), 그리고 다양성 (variety)을 세 가지 특징인 3V로 정의한다. 즉, 데이터가 생산되는 속도 (velocity)가 매우 빠르고, 데이터의 종류 (숫자, 문자, 음성, 영상 등)가 매우 다양하며 (variety), 용량 (volume)이 매우 방대하다는 것을 의미한다 (Kim과 Cho, 2013).

빅데이터를 위한 분산·병렬 처리 프레임워크로 아파치 하둡 (apache hadoop)이 대표적이다. 네트워크로 연결된 여러 서버에 데이터를 분산 저장한 후, 맵리듀스로 분산·병렬 처리하는 방식이다 (Park 등, 2013). 그러나 하둡은 대용량 데이터의 분석을 위한 일괄처리 (batch processing)에 적합하지만, 실시간 데이터 처리에는 적합하지 않다. 실시간 데이터 처리는 기존 디스크 기반의 데이터 저장 및 처리 방식을 넘어 고속의 메모리 기반 데이터 처리 기술이 적합하다.

IMC는 디스크에 저장하던 데이터를 메모리에 저장 및 처리하는 컴퓨팅 기술을 의미한다. Figure 2.1은 IMC 기술요소를 계층 구조적으로 나타낸다. IMC의 주요 기술로 DRAM, 다중프로세서 (multi-core) 등의 하드웨어와 IMDB, IMDG 등의 인메모리 데이터 관리 소프트웨어 그리고 CEP 등의 인메모리 응용 플랫폼이 포함된다.

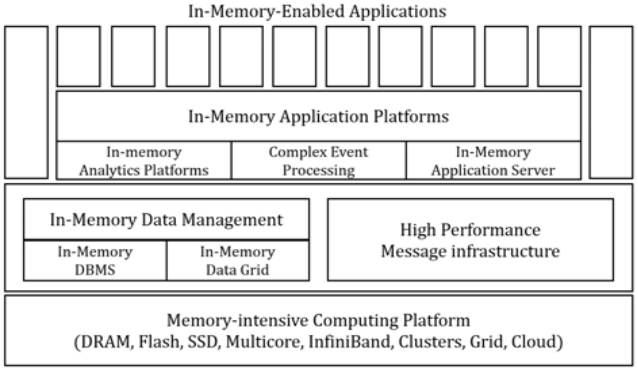

Figure 2.1 Gartner taxonomy of in-memory computing technologies

2.2. In-memory database

최근 기가비트 이더넷 네트워크의 발전과 점점 낮아지는 메모리 가격에 힘입어 인메모리 컴퓨팅 기술 발전에 큰 영향을 주고 있다. IMDB는 모든 데이터를 메모리에 상주시켜 디스크 I/O를 최소화하고 인덱싱 기법, 동시성제어, 회복 처리, 그리고 트랜젝션 처리를 최적화함으로써 디스크 기반 DBMS대비 빠른 처리가 가능하다 (Han 등, 2011).

일반적으로 성능향상을 위해서는 단일 시스템의 하드웨어 자원 업그레이드 방식인 수직확장성 (scale-up)과 다수의 노드로 구성된 분산 컴퓨팅 환경에서 노드를 증설하는 방식인 수평확장성 (scale-out)으로 구분된다. 전자는 하드웨어 자원의 업그레이드 가능성이 제한적인 단점이 있으며, 후자의 경우도 노드의 증설함에 따라 성능이 증가하지만 네트워크에 매우 민감하다는 단점이 존재한다.

IMDB의 수평확장성을 지원하는 소프트웨어로는 MySQL cluster, Oracle RAC, 그리고 Kairos-D 등이 있다. Han 등 (2011)은 대용량 데이터를 빠르게 처리하기 위해 IMDB 상용 제품인 Kairos 기반에 분산·병렬기술을 적용한 Kairos-D에 관한 연구다. 이중화 된 단일노드의 IMDB인 Kairos와 분산·병렬기술을 적용한 IMDB인 Kairos-D의 벤치마크에서 단일 노드에서 이중화 된 Kairos가 분산모듈이 추가된 Kairos-D에 비해 약 1.4% 가량 빠른 성능을 보였다. 이는 Kairos-D의 분산모듈이 성능에 미치는 영향력이 크지 않다는 의미한다. 반면, Kairos-D의 노드를 2대로 늘려 실험한 결과는 Kairos-D가 약 30% 빠른 성능을 보였다. 따라서 분산·병렬 기술 적용이 대용량 데이터 처리에도 적합하다는 결론을 도출하였다.

그러나 IMDB 기술도입은 기존의 시스템을 IMDB에 맞추어 재개발해야하는 어려운 일이다. 기존 시스템 변경을 최소화하면서 IMDB 기술을 시스템에 적용하기에는 상당한 어려움이 따른다. 즉, 기존 시스템에서 사용되던 트리거, 또는 프로시저와 같은 특정 작업을 수행하는 함수들을 IMDB에 맞추어 수정하거나 다시 개발해야한다는 단점을 갖는다.

2.3. In-memory data rid

IMDG는 높은 수평확장성과 메모리 기반의 빠른 데이터 처리 능력을 제공한다. IMDG는 분산 환경을 구성하기 위한 API (application programming interface)를 제공하며, 물리적으로 분리되어 있는 네트워크로 시스템을 연결하여 논리적으로 하나의 시스템처럼 메모리를 관리한다. G Kim (2006)은 그리드 컴퓨팅 기술로 대용량의 생물정보데이터를 분산 처리하는 연구를 제안하였다. 또한 Kim (2007)은 그리드 컴퓨팅 기술을 활용하여 OLAP (Online Analytical Processing) 질의를 향상시키는 방법인 HyperDB를 제안하였으며 질의평가를 위해 표준 TPC-R 벤치마크를 수행하여 성능을 입증하였다. 그 실험에 따르면 일정 수의 노드증가에 따라 응답시간은 현저히 짧아졌다. HyperDB는 IMDG 기술을 사용하지는 않지만, 여러 컴퓨터를 엮는 그리드 컴퓨팅 기술을 활용하여 성능향상을 꾀할 수 있었다.

Park (2015)는 key-value 저장소의 Redis, IMDG의 Hazelcast, 그리고 오픈소스 기반의 DBMS MariaDB 간의 성능비교 실험을 수행하였다. 그 결과 Hazelcast와 같은 IMDG 가 Redis에 비해 약 17.5% 정도 더 많은 양의 트랜잭션을 수행할 수 있는 것으로 나타났고, 또한 MariaDB에 비해서도 약 573배 정도 더 높은 처리량 (throughput)을 보였다. IMDG는 분산 환경 구성으로 노드 증가에 따라 더 좋은 성능을 보여주었다. IMDG의 우수한 성능을 보여주는 대목이다.

Arora와 Gupta (2014)는 제한된 확장성을 가진 전통적 방식의 DBMS 대신 IMDG를 적용해 수평확장성을 확보함으로써 업무관련 애플리케이션을 기반으로 하는 클라우드 시스템을 높였다. 또한 Bahl 등 (2012)은 빠른 성능과 확장성이 요구되는 GIS (Geographic Information System) 애플리케이션 개발을 위해 IMDG 기술을 적용하여 문제를 해결하였다. 이처럼 IMDG는 빠른 연산을 필요로 하는 문제, 확장성의 유연함을 필요로 하는 문제, 그리고 클라우드 시스템 성능향상을 위한 사례처럼 다양한 문제해결에 적용되고 있다.

Figure 2.2는 IMDG의 구조를 보여준다. 그림에서 'Application Logic'은 IMDG를 통해 데이터의 읽기/쓰기를 수행한다. IMDG의 데이터는 비동기 방식으로 데이터베이스에 저장하는 구조를 갖는다.

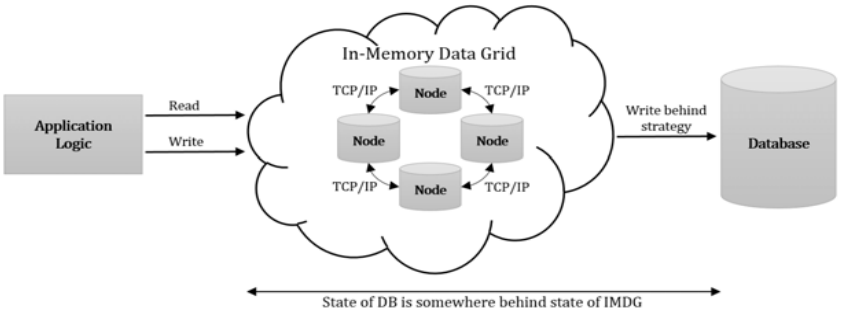

Figure 2.2 In-memory data grid asynchronous DB writing

3. 택시 애플리케이션 성능 향상 기법

본 절에서는 기존의 택시 애플리케이션의 구조 변경 없이도 성능 향상이 가능한 서버 애플리케이션 구조를 제안한다. 택시 애플리케이션은 실시간 처리를 요구하는 시스템이므로 일반적으로 고려해야하는 사항은 다음과 같다. i) 택시의 빈번한 위치정보 업데이트 ii) 고객의 주변택시 검색 iii) 고객의 택시 동시 호출 관리 iv) 개인정보 관리 등이다. 하지만 개인정보관리와 같은 보안 관련 사항과 택시동시호출관리 같이 코드수정이 불가피한 기능사항은 본 논문의 범위 밖이므로 고려하지 않는다.

3.1. 택시 애플리케이션의 구조 제안

일반적인 고려사항 중, 택시 애플리케이션에서 서버에 빈번하게 요청되는 부분은 i)택시의 위치정보 업데이트와 ii)고객의 주변택시 검색으로 한정할 수 있다. Figure 3.1은 요청을 '쓰기요청'과 '읽기요청' 두 가지로 단순화한 것이며, 컴포넌트에 대한 설명은 Table 3.1과 같다.

Table 3.1 Description of Components

Component	Description
Passenger	Passenger's application
Taxi	Taxi's application
WebServices	List of Web Services
DB	Database Management System
→	Request
IMDG	Instance of in memory data grid
Other Applications	Other Applications
Speedment	Data synchronizer; Synchronizing the changes in database to the instance of IMDG

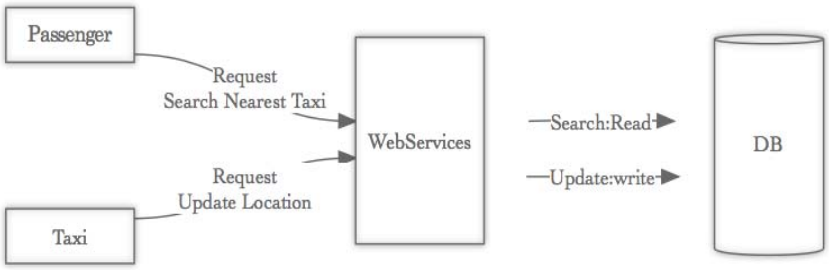

Figure 3.1 Simplified workflow diagram of read/write requests and processes using DBMS

Figure 3.2는 택시 애플리케이션의 요청/처리 과정을 도식화한 개념도로, 중간단계에 위치한 웹서비스가 택시와 고객의 요청을 DBMS에 접근하여 데이터 검색 및 업데이트 요청을 처리하는 구조이다. 이러한 구조의 단점은 고객과 택시의 요청이 갑자기 증가할 경우 웹서비스의 성능이 DBMS의 처리성능에 종속된다는 점이다. 성능의 종속구조 문제를 해결하기 위해서 이전 절에서 소개한 Park (2015)는 IMDG를 활용한 구조를 제안하였지만, 타 애플리케이션이 DBMS로 특정 데이터에 대해 업데이트를 할 경우 IMDG와 DBMS 간의 데이터가 서로 같지 않은 현상인 '데이터의 일관성' 문제가 제기될 수 있다.

이러한 한계점을 극복하기 위해 본 연구에서는 IMDG 기술을 활용하면서도 '데이터 의 일관성' 문제를 해결할 수 있는 기법을 기술한다. 뿐만 아니라 기존의 DBMS를 유지하면서 효율적으로 DBMS부하를 분산 할 수 있다. 이러한 요구사항을 충족시키기 위해 기존의 웹서비스 요청부분을 '읽기'와 '쓰기'로 단순화하고 '읽기'는 IMDG에서, '쓰기'는 DBMS에서 담당하여 처리하는 방법을 사용한다.

요청을 처리하기 위한 작업흐름은 다음과 같은 순서로 기술할 수 있다. 검색요청'읽기'의 경우 i) 고객의 주변 택시 검색 요청 ii) 웹서비스에서 IMDG에 주변택시 검색 요청 iii) 주변택시 검색 결과를 고객에게 전송 하는 순서로 요청이 처리된다. 업데이트요청'쓰기'의 경우 i) 택시 위치정보 업데이트 요청 ii) 웹서비스에서 DBMS에 택시 위치정보 업데이트 요청 iii) DBMS의 트랜잭션처리 후 결과를 전송하는 순서로 처리된다.

이러한 단순화된 시스템 구조를 사용할 경우 다음과 같은 장점이 있다. i) 읽기와 쓰기를 분리함으로써 읽기와 쓰기 요청을 동시에 처리하기 위해 발생하는 DBMS의 부하를 감소시킨다. ii) 읽기의 경우 IMDG의 유연한 수평적 확장성과 메모리의 빠른 처리 장점을 가진다. iii) 타 애플리케이션에서 DBMS로 트랜잭션 데이터를 요청을 처리하더라도 데이터의 일관성을 위반하는 것을 방지할 수 있다.

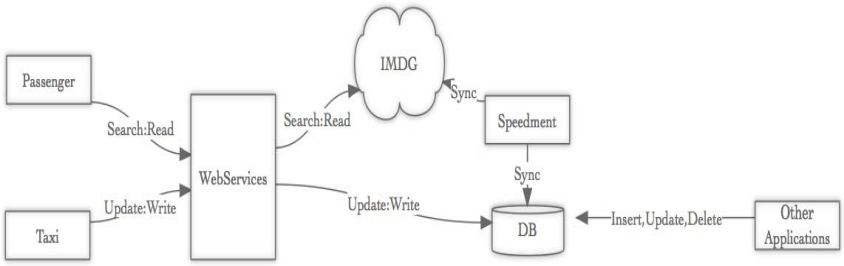

Figure 3.2 Simplified workflow diagram of seperating read/write processing using IMDG

3.2. 고객의 주변택시 검색 방법 제안

이전 절까지 전반적인 애플리케이션의 성능을 향상시킬 수 있는 구조에 대해 기술하였다. 본 절에서는 주변택시 검색 방법을 3가지로 나누어 설명하고, 각 방법의 장단점을 설명한다.

주변택시 검색은 총 3가지로 case1, case2, case3으로 분류하여 설명한다. 첫째 case1은 지구의 곡면 위의 거리 (Great Circle Distance) 를 이용하여 지점 P1과 지점 P2의 최단거리를 계산 후 주어진 거리 내에 존재하는지를 비교하는 방법이다. 둘째, case2는 주어진 지점에서 특정 거리를 계산하여 사각형의 대각선 경계지점 P1, P2 추정 후 범위 검색하는 방법이다. 마지막으로 셋째, case3은 첫 번째 방법과 두 번째 방법을 모두 사용하는 방법이다. 첫 번째 방법 Case1은 지구의 곡면 위의 거리 (Great Circle Distance)를 계산하는 방법으로 두 지점 간의 최단 거리를 계산하며 식 (3.1)로 표현가능하다 Gade (2010).

$$d = \arccos(\sin(lat_1) \cdot \sin(lat_2) + \cos(lat_1) \cdot \cos(lat_2) \cdot \cos(lon_1 \_ lon_2)) \cdot R \tag{3.1}$$

Figure 3.3은 두 지점간의 거리를 기반으로 택시의 거리에 따른 주변택시를 표기한 것으로 미리 정의된 거리보다 실제 거리가 먼 경우는 제외한다. 예를 들어 검은색은 정의된 거리안의 있으므로 주변택시에 해당하지만, 빨간색은 주변택시에 해당하지 않는다. 이 방법은 두 지점간 거리를 매우 정확하게 계산하지만 택시의 수가 증가함에 따라 연산량도 비례하여 증가하므로 택시가 많지 않은 경우에 고려해야하는 방법이다.

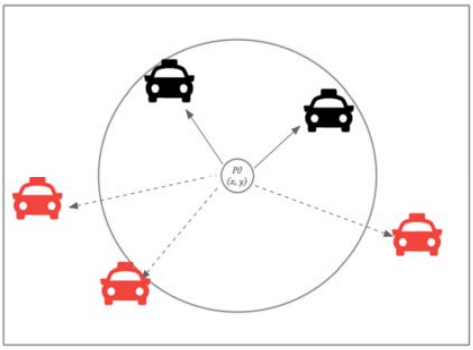

Figure 3.3 Case1 search method

두 번째 방법인 Case2는 범위검색을 위해 주어진 지점 P0에 대하여 지점 P1과 P2의 지점을 도출한 뒤, 택시가 두 지점 사이에 존재하는지 여부를 비교적 간단하게 비교 연산한다. Figure 3.4는 이 방법을

이용한 계산방식을 그림으로 나타낸다. 이 방법은 범위검색방식으로 주변의 택시 검색을 수행하므로 속도가 매우 빠르지만 정확한 거리를 측정하지 못한다. 주어진 지점 P0과 모든 택시와의 거리를 비교하지 않더라도 첫 번째 방법과 유사한 주변택시목록을 반환할 수 있으므로 정확하지 않지만 첫 번째 방법과 비슷한 결과를 원한다면 대안이 될 수 있다.

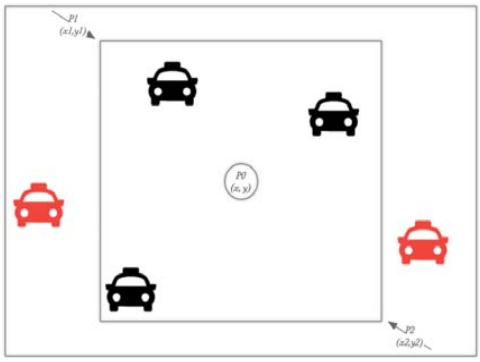

Figure 3.4 Case2 search method

세 번째 방법 Case3은 Case1과 Case2를 결합한 하이브리드 방법으로, Case2로 범위검색 후 얻어진 택시목록에 정확한 Case1의 거리연산을 추가적으로 수행함으로써 정확히 거리가 계산된 주변택시결과를 반환한다. 이는 택시가 많은 경우 Case1의 거리 계산량이 늘어나는 문제를 범위검색을 통해 주변택시목록을 반환받은 후, 지점 P0과 주변택시의 거리를 정확하게 계산을 하는 방식이다. Figure 3.5는 Case3의 검색 방법을 그림으로 표현한 것이다.

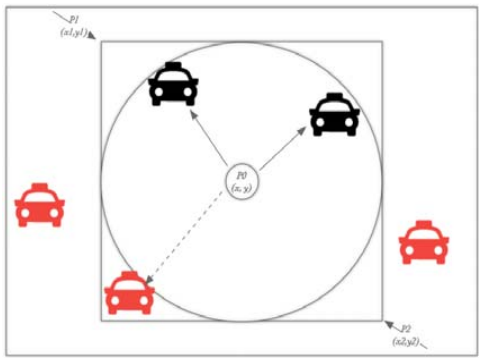

Figure 3.5 Case3 search method

4. 성능 평가

4절에서는 3.2절에서 제안한 방법에 대한 성능을 평가하고 결과를 기술한다. 성능 평가는 '읽기'요청에 대하여 3가지 다른 방법을 각각 DBMS와 IMDG에 적용한 후, 택시량 증가에 따른 요청 처리시간과 처리량을 측정하였다. 결과는 첫 번째, 두 번째, 세 번째 방식을 각각 CASE1, CASE2, CASE3으로 표기하였으며 실험 환경은 Table 4.1과 같다.

Table 4.1 Experiment environment

Item	Specification
DBMS	MySQL 5.5.28
IMDG	Hazelcast 3.3.5
CPU	intel i7-4770
Memory	24GB
HDD	2TB
Network	1gigabit Ethernet

Figure 4.1과 Figure 4.2는 택시의 수를 5만대씩 증가시키며 각각의 주변택시 검색방법에 따른 분당 처리량과 하나의 요청을 처리하는데 소요되는 시간을 측정한 것이다. Figure 4.1에서 DBMS와 IMDG의 검색방법에 따라 분당 처리량이 차이가 나는 것을 확인 할 수 있으며 IMDG의 처리속도가 전반적으로 빠른 것을 확인 할 수 있다.

Figure 4.2는 하나의 요청이 처리되는 평균 시간을 나타내며 각 방법별로 DBMS와 IMDG간 처리시간이 6배에서 9배 정도 차이가 있음을 확인할 수 있다. Table 4.2는 각 DBMS와 IMDG의 평균 처리시간을 나타낸다. DBMS의 경우 Case2와 Case3의 평균 처리 속도가 Case1에 비해 각각 57배, 46배 향상되었다. IMDG의 경우도 Case2와 Case3의 평균 처리 속도가 Case1에 비해 각각 43배, 48배 향상됨을 확인 할 수 있다.

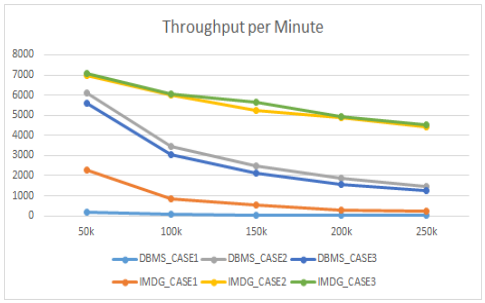

Figure 4.1 Throughput per minutes

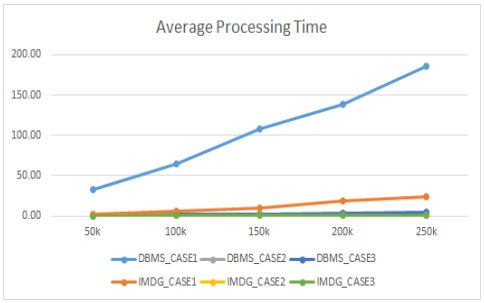

Figure 4.2 Average processing time

Table 4.2 Average processing time all of cases Unit(second)

	CASE1	CASE2	CASE3
DBMS	106.04	1.86	2.30
IMDG	12.13	0.28	0.25

Table 4.3은 택시의 수가 증가함에 따른 IMDG와 DBMS의 처리량 감소비율을 나타낸다. 택시의 수가 5만대씩 증가함에 따라 DBMS의 경우 처리량이 Case1은 평균 34%, Case2는 29%, Case3은 31% 감소하였으며, IMDG의 경우 Case1은 41%, Case2는 11%, Case3은 10% 감소하여 Case1을 제외한 나머지 Case2와 Case3에서 처리량 감소가 DBMS보다 낮음을 확인 할 수 있다.

Table 4.3 Decrease rate in average throughput

	CASE1	CASE2	CASE3
DBMS	34%	29%	31%
IMDG	41%	11%	10%

Figure 4.3은 Case1의 방법으로 택시를 5만대씩 증가시키며 DBMS와 IMDG의 평균 요청처리 시간을 비교한 것으로, IMDG와 DBMS 모두 택시량 증가에 비례하여 처리시간이 증가하였다. Figure

4.4는 그에 따른 처리량 변화를 나타내는 것으로, 택시량 증가에 반비례하여 처리량이 감소하였다.

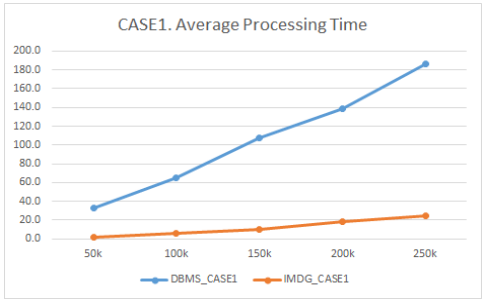

Figure 4.3 Case1 average processing time

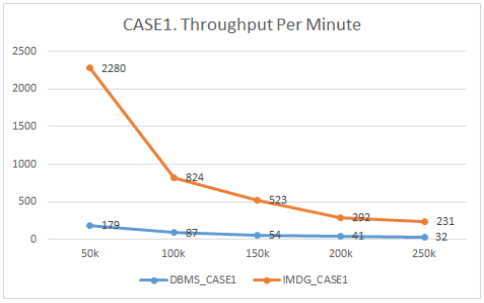

Figure 4.4 Case1 average throughput

Figure 4.5는 Case2 결과이며 Case1과 비교했을 때, 검색 속도는 DBMS의 경우 평균 약 10배 향상되었고, IMDG의 경우 약 43배정도 향상되었다. Figure 4.6은 그에 따른 처리량 변화를 나타내는 것으로, IMDG는 DBMS에 비해 감소변화량 폭이 작음을 확인 할 수 있다

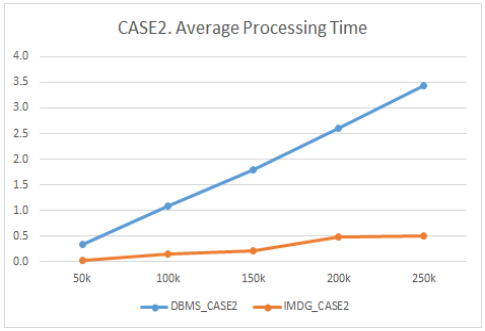

Figure 4.5 Case2 average processing time

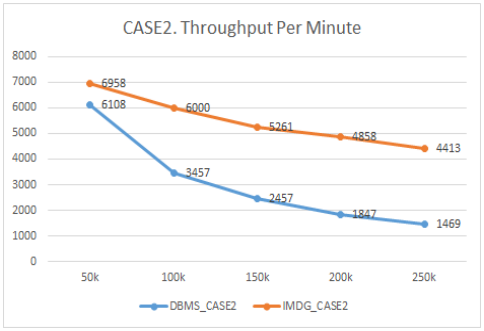

Figure 4.6 Case2 average throughput

.

Figure 4.7은 Case3 결과로 Case2와 비교했을 때, DBMS의 경우 1.8초에서 2.5초로 38% 처리시간이 증가하였고, IMDG의 경우 0.28초에서 0.25초로 10%의 처리시간 감소가 있었다. 연산량의 증가로 인해 IMDG 또한 처리시간이 증가해야함에도 처리시간이 감소한 것은 연산속도가 빨라진 것 보다는 네트워크 유동성을 따르는 IMDG의 특성 때문인 것으로 판단된다. Figure 4.8은 그에 따른 처리량 변화를 나타내며 Case2와 유사한 처리량을 보인다.

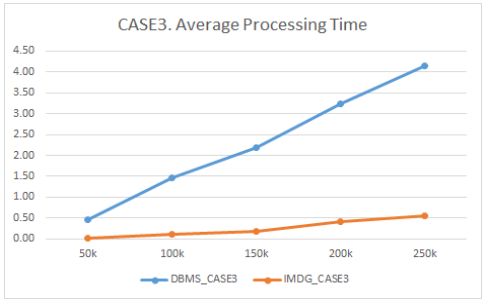

Figure 4.7 Case3 average processing time

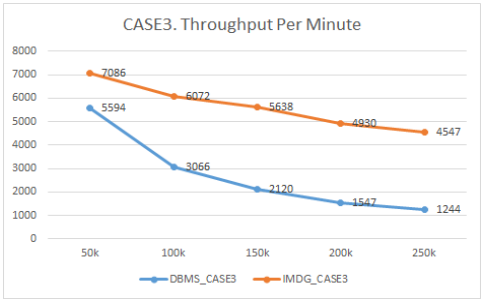

Figure 4.8 Case3 average throughpu

5. 결론 및 향후 연구 방향

본 연구는 IMDG 기술을 택시 어플리케이션에 접목하여 기존의 데이터베이스의 변경 없이 성능을 향상 시키는 기법을 제안하였다. 기존의 DBMS기반 택시 애플리케이션의 웹서비스를 '읽기 (검색)'과 '쓰기 (업데이트)' 처리요청으로 분류 후, '읽기' 처리요청은 IMDG로 그리고 '쓰기' 처리요청은 DBMS로 처리부를 분리하여 서비스 처리요청을 향상시켰다. '읽기' 처리요청은 총 3가지 방법으로 나누어 각각 DBMS와 IMDG의 처리속도와 처리량을 측정하였다. 그 결과, 같은 방법을 사용할 경우 IMDG의 성능이 우수함을 확인하였다.

또한 본 연구의 단순 '읽기/쓰기' 분리 처리구조는 기존 데이터베이스의 변경 없이도 IMDG 기술을 기존 시스템에 적용가능하다. 뿐만 아니라 '읽기'와 '쓰기'에 대한 부하를 분산하고, DBMS와 IMDG간 '데이터 일관성' 문제를 해결할 수 있다. 따라서 기존 데이터베이스 변경이 어렵거나 불가능한 제한적인 환경에서 전체 시스템의 성능을 높일 수 있다는데 의의를 갖는다.

하지만 DBMS의 경우 공간색인을 이용하여 성능을 향상시키는 여러 가지 알고리즘이 제안되었으나, IMDG의 경우 공간색인을 공식적으로 지원하지 않기 때문에 공간색인을 이용한 방법을 적용하지 못하였다는 한계점을 가진다. 이러한 한계점을 보완하기 위해 향후 연구는 IMDG의 분산 공간데이터에 대한 분산 공간색인을 적용방법 및 성능을 향상 연구를 진행할 예정이다.

별첨

본 연구에서 사용된 소스코드는 bitbucket에 공개되어 있습니다. 소스코드는 다음 주소에서 내려받을 수 있습니다.

`https://bitbucket.org/charisma0629/high_thoughput_and_scalable_imdg_server_git`.

References

Arora, I. and Gupta, A. (2014). Improving performance of cloud based transactional applications using in-memory data grid. *International Journal of Computer Applications*, **107**, 14-19.

Bahl, B., Sharma, V. and Rajpal, N. (2012). Boosting geographic information system's performance using in-memory data grid. *BVICAM's International Journal of Information Technology*, **4**, 468-473.

Gade, K. (2010). A non-singular horizontal position representation. *The Journal of Navigation*, **63**, 395-417.

Han, S., Jin, S. and Kim, Y. (2011). An architecture of a high performance distributed main memory database management system for massive data. *The Korean Institute of Information Scientists and Engineers*, **38**, 141-148.

Kim, T., Na, J. and Cho, W. (2006). PC-based hybrid grid computing for huge biological data processing. J*ournal of the Korean Data & Information Science Society*, **17**, 569-579.

Kim, T., Na, J. and Cho, W. (2007). HyperDB - A high performance data analysis system based on grid computing technology. *Journal of the Korean Data & Information Science Society*, **18**, 161-174.

Kim, Y. and Cho, K. (2013). Big data and statistics. *Journal of the Korean Data & Information Science Society*, **24**, 959-974.

Park, J., Lee, S., Kang, D. and Won, J. (2013). Hadoop and MapReduce. *Journal of the Korean Data & Information Science Society*, **24**, 1013-1027.

Park, M. (2015). *A method to enhance the real-time processing performance of traffic big data by applying in-memory data grid technology*, Master Thesis, Chungbuk National University, Cheongju.

Journal of the Korean Data & Information Science Society
2015, **26**(5), 1035–1045

http://dx.doi.org/10.7465/jkdi.2015.26.5.1035
한국데이터정보과학회지

Enhancing the performance of taxi application based on in-memory data grid technology[†]

Chi-Hwan Choi[1] · Jin-Hyuk Kim[2] · Min-Kyu Park[3] ·
Kaaen Kwon[4] · Seung-Hyun Jung[5] · Franco Nazareno[6] · Wan-Sup Cho[7]

[16]Department of Bio-Information Technology, Chungbuk National University
[2]Department of Bigdata, Chungbuk National University
[34]Department of BDC, Chungbuk National University
[5]Department of Information Industrial Engineering, Chungbuk National University
[7]Department of MIS / Business Data Convergence, Chungbuk National University

Received 8 August 2015, revised 21 September 2015, accepted 24 September 2015

Abstract

Recent studies in Big Data Analysis are showing promising results, utilizing the main memory for rapid data processing. In-memory computing technology can be highly advantageous when used with high-performing servers having tens of gigabytes of RAM with multi-core processors. The constraint in network in these infrastructure can be lessen by combining in-memory technology with distributed parallel processing. This paper discusses the research in the aforementioned concept applying to a test taxi hailing application without disregard to its underlying RDBMS structure. The application of IMDG technology in the application's backend API without restructuring the database schema yields 6 to 9 times increase in performance in data processing and throughput. Specifically, the change in throughput is very small even with increase in data load processing.

Keywords: Database, in-memory computing, in-memory data grid.

[†] This research was supported by the ITRC (Information Technology Research Centre) support program (IITP-2015-H8501-15-1013).
[1] Ph.D. student, Department of Bio-Information Technology, Chungbuk National University, Cheongju 362-763, Korea.
[2] Master student, Department of Bigdata, Chungbuk National University, Cheongju 362-763, Korea.
[3] Master student, Department of BDC, Chungbuk National University, Cheongju 362-763, Korea.
[4] Master student, Department of BDC, Chungbuk National University, Cheongju 362-763, Korea.
[5] Ph.D. student, Department of Information Industrial Engineering, Chungbuk National University, Cheongju 362-763, Korea.
[6] Ph.D. student, Department of Bio-Information Technology, Chungbuk National University, Cheongju 362-763, Korea.
[7] Corresponding author: Department of MIS / Business Data Convergence, Chungbuk National University, Cheongju 362-763, Korea. E-mail: wscho@chungbuk.ac.kr