유비쿼터스 교통환경에서 실시간 데이터 저장을 위한 시스템 구조 및 저장 알고리즘

백호현*, 이동욱*, 김재훈*,
백정희**, 이정우**
*아주대학교 정보통신전문대학원
**다음기술㈜
e-mail : *{hohyunny, dwlee, jaikim}@ajou.ac.kr
**{agnes, jwlee}@ntechs.com

System Architecture and Data Backup Strategy for Real-Time Data Processing in Ubiquitous Transportation Environment.

Ho-Hyun Baek*, Dong-Wook Lee*, Jai-Hoon Kim*,
Jung-Hee Baek**, Jung-Woo Lee**
* Graduate School of Information and Communication, Ajou University
**Next Technologies Co.,Ltd

요 약

본 논문에서는 유비쿼터스 지능형교통시스템(u-ITS: Ubiquitous Intelligent Transportation System) 에서 발생하는 다량의 데이터를 실시간 저장하기 위해 메인 메모리 데이터베이스(MMDB: Main Memory Database)와 디스크 상주 데이터베이스(DRDB: Disk Resident Database)를 사용하는 시스템 구조와 RM(Rate Monotonic)를 기반으로 데이터의 발생 주기에 따라 MMDB 와 DRDB 에 실시간 저장하는 알고리즘을 제안한다.

1. 서론

현재의 지능형교통시스템(ITS)은 도로상에 설치된 각종교통정보 수집장치를 통해 실시간 교통 정보를 수집한다. 수집된 데이터는 교통정보센터로 전송되어 가공되고, 가공된 정보는 도로 이용자에게 전송된다. 교통정보의 수집이 도로에 설치된 수집장치에 의해 이루어지기 때문에 수집되는 데이터는 수집장치의 범위 이내에 국한된다. 또한 수집되는 데이터가 검지기에 의존하기 때문에 차량의 상세한 정보를 얻을 수 없다. 이러한 한계를 극복하기 위해 u-ITS 에서는 모든 차량에, 차량의 상태 정보 및 운행 정보를 감지할 수 있는 센서를 설치하여 데이터를 수집하고 가공하여 제공한다 [1].

u-T 시스템은 데이터의 수집 단위가 도로가 아니라 각각의 차량이기 때문에 차량이 운행하는 위치의 모든 교통 정보를 수집할 수 있고 기존의 시스템에서는 수집할 수 없었던 차량의 상태 정보의 수집이 가능하며, 이러한 다양한 정보를 바탕으로 기존의 시스템보다 더욱 신속, 정확하게 다양한 교통정보를 관리하고 서비스할 수 있다. 그러나 차량 단위의 데이터 수집으로 기존 시스템에 비해 다량의 데이터가 수집되어 수집된 데이터의 저장 및 처리에 많은 시간과 자원이 필요하다. 교통의 특성상 수집된 데이터는 실시간 저장, 가공 및 제공 되어야 정보로서 의미를 가질 수 있으므로 다량의 데이터를 효율적으로 처리할 수 있는 시스템 구조에 대한 연구가 필요하다.

본 논문에서는 다량의 데이터를 실시간 처리하기 위해 수집된 데이터를 MMDB 를 이용해 실시간 저장하고, 가공할 수 있는 시스템 구조 및 효율적인 MMDB 메모리 관리를 위한 교통정보 저장 알고리즘 제안함으로써 모든 데이터를 Deadline 이내에 처리하고, MMDB 의 메모리를 효율적으로 사용한다.

2. 배경지식 및 관련연구

2.1 메인 메모리 데이터베이스(MMDB)

MMDB 는 일반적으로 사용되는 DRDB 와 달리 컴퓨터의 메인 메모리를 저장 매체로 사용하는 데이터

※ 본 연구는 건설교통부 교통체계효율화 사업의 연구비 지원(06 교통핵심 A01)에 의해 수행되었습니다.

베이스이다. 즉 데이터베이스의 일부 또는 전부를 메인 메모리에서 관리함으로써 디스크에 대한 접근 없이 메모리 접근 만으로 데이터를 처리한다.

MMDB 는 메모리의 특성상 낮은 접근시간과 높은 데이터 전송속도를 제공하고 메모리만을 관리하는 간단한 알고리즘을 사용해 메모리와 디스크 모두를 관리해야 하는 MMDB 에 비해 트랜잭션 처리 속도가 빠르다 [2].

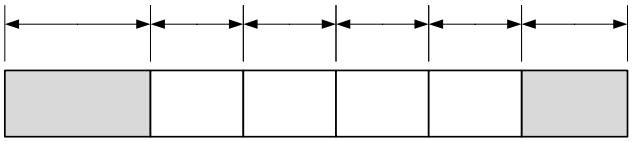
또한 MMDB 는 기본적으로 모든 데이터가 디스크에 존재한다고 가정하기 때문에, 각각의 레코드에 접근하기 위해서는 레코드의 논리적 주소를 물리적 주소로 변환하는 매핑(Logical-to-Physical Address Mapping) 작업이 필요하지만 MMDB 는 모든 데이터가 메모리에 존재하기 때문에 논리적-물리적 주소 변환 없이 레코드에 바로 접근 할 수 있는 성능상이 이점을 갖는다.

MMDB 가 DRDB 에 비해 월등한 성능상의 이점을 갖는 반면, 저장 공간의 크기가 작아 모든 데이터를 메모리에 상주시킬 수 없는 문제점이 있다. 이를 해결하기 위해 본 논문에서는 데이터의 특성에 따라 우선순위를 부여해 MMDB 와 DRDB 에 데이터를 분산해 저장하는 알고리즘을 제안한다.

2.2 MMDB 성능 분석

본 연구의 목적은 대량의 데이터를 실시간으로 저장하는 것이 목적이므로 MMDB 와 Disk DBMS 의 데이터 삽입(INSERT) 성능을 테스트하였다.

u-ITS 에서는 개별 차량의 위치정보가 주기적으로 수집되어 서버로 전송되며, 기존의 GPS 시스템의 위치정보 갱신 주기가 1 초인 것을 감안해 보면, u-ITS 는 GPS 의 주기보다는 빠를 것으로 예상되므로 매우 많은 데이터가 주기적으로 생성될 것임을 알 수 있다. 그러므로 삽입 테스트는 발생 빈도나 데이터 양이 가장 많을 것으로 예상되는 개별 차량의 위치정보 데이터를 이용하였고, 데이터 포맷은 (그림 1)과 같이 차량 고유번호, 위치정보, 속도정보, 생성시간으로 구성되면 전체 데이터 크기는 37byte 이다.



(그림 1) 위치정보 데이터

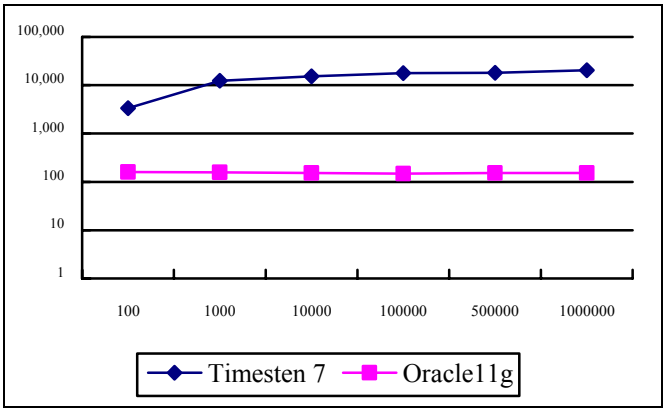
MMDB 와 DRDB 의 성능비교를 위한 삽입 테스트 프로그램은 리눅스 상에 자바(Java)와 JDBC(Java Database Connectivity) 를 이용하여 구현하였고, MMDB 는 Oracle 사의 TimesTen 7.03 버전, DRDB 는 같은 회사의 Oracle 11g 버전을 이용하였다 [3].

개별 차량의 위치 데이터는 데이터의 발생 주기가 중요한 요소이지만 많은 차량의 위치 정보를 수신하고 처리하는 서버에서는 주기성 보다는 단위시간당 데이터 처리 능력이 중요하므로 데이터는 일괄적으로 생성해서 저장 하였으며, 그 결과는 <표 1>과 같다.

<표 1> MMDB 와 DRDB 삽입성능 비교

	TimesTen 7		Oracle 11g	
데이터 수	시간(ms)	TPS*	시간(ms)	TPS*
100	0.03	3,333	0.63	159
1,000	0.08	12,500	6.42	156
10,000	0.65	15,385	65.89	152
100,000	5.62	17,794	677.97	147
500,000	27.33	18,295	3,292.37	152
1,000,000	48.87	20,462	6,577.49	152

메모리의 빠른 접근시간 덕분에 삽입 작업의 성능은 100 만 건의 데이터를 저장한 경우 MMDB 의 삽입 성능이 DRDB 보다 약 134 배 빠르다는 결과를 얻었다. 또한 삽입 데이터의 양이 증가할수록 MMDB 의 초당 트랜잭션 수(TPS) 는 증가하는 반면 DRDB 의 TPS 는 삽입 데이터의 양의 증가에 관계없이 비슷한 값을 나타냈다. 즉 데이터의 양이 증가할수록 MMDB 와 DRDB 의 성능상 차이가 커짐을 알 수 있다.



(그림 2) MMDB 와 DRDB 삽입성능 비교

2.3 관련연구

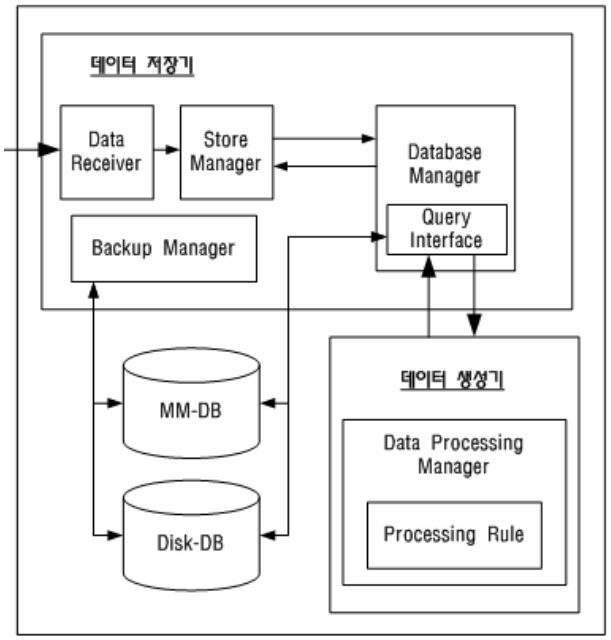
실시간 시스템에서 동시에 처리되어야 할 여러 가지 작업에 대해 각각 우선순위를 부여해 스케줄링 한다. 기존에 제안된 우선순위 부여 알고리즘은 First Come First Serve(FCFS), Earliest Deadline First(EDF), Least Slack Time First(LSF), Rate Monotonic First(RM) 등이 있다 [4].

FCFS 는 작업이 Release 된 순서대로 처리하는 방식으로 Deadline 정보를 사용하지 않으므로 실시간 시스템에 적합하지 않다. EDF 는 Deadline 이 임박한 작업을 우선 처리하는 방법이고, LST 는 Slack Time(작업을 처리하는데 예상되는 시간과 deadline 의 차이 즉 예상되는 완료시간부터 deadline 까지 여유시간) 이 작은 값을 가질수록 높은 우선순위를 부여하는 방법이다. RM 은 데이터 발생 주기가 짧을수록 즉, 데이터가 자주 발생할수록 높은 우선순위를 부여한다.

위에서 언급한 4 가지 우선순위 부여 알고리즘 이외에 시스템 상황에 따른 다양한 알고리즘이 존재한다. 실시간 시스템을 구현한 연구 중에는 Deadline, Period, Criticality 와 같은 특징들을 이용한 우선순위 부여 알고리즘을 사용한 예도 있다. [5]

3. 시스템 구조

주기적으로 발생하는 대량 데이터 삽입 트랜잭션을 빠르게 처리하기 위해 데이터 병목현상이 발생할 것으로 예상되는 데이터 수신, 데이터 저장을 담당하는 모듈은 Thread Pooling 기법을 이용하여 미리 생성되어 있는 인스턴스를 이용해 빠르게 작업을 시작할 수 있도록 구현하였다



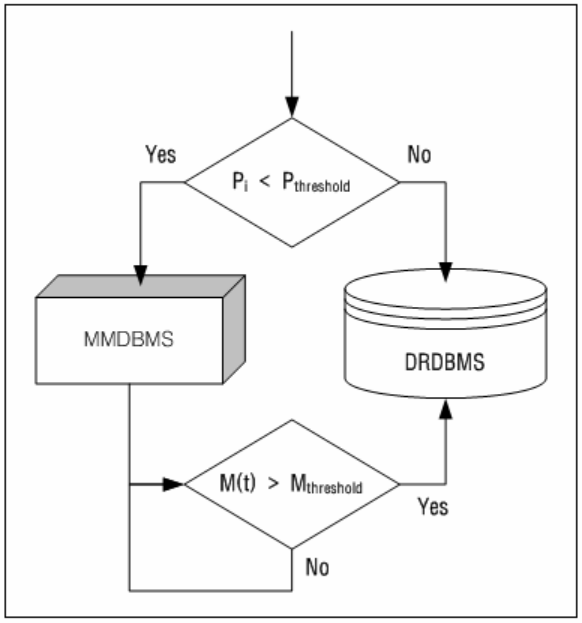
(그림 3) 시스템 구조도

1. 차량에서 생성된 데이터는 도로상에 설치된 데이터 수집기로 전송되며 데이터는 수집기에 의해 1 차적으로 Filtering 과 Aggregation 을 거쳐 데이터 저장기의 Data Receiver 로 전송된 후 Store Manager 내부의 메시지 큐로 전달된다.
2. Store Manager 는 내부의 메시지 큐에서 순서대로 메시지를 하나씩 꺼내 Database Manager 에 전달하고 Database Manager 는 전달된 데이터를 Query Interface 의 큐에 저장한다.
3. Query Interface 의 큐에는 데이터 삽입(INSERT) 작업뿐만 아니라 교통 데이터 생성을 위한 데이터 선택(SELECT) 작업도 저장된다. 데이터 선택 작업은 Data Processing Manager 에 의해 주기적으로 또는 사용자의 요청에 의해 생성된다.
4. Backup Manager 는 MMDB 에 누적된 데이터를 DRDB 로 백업하는 역할을 한다. 데이터 백업은 미리 정의된 백업전략(Backup Strategy)에 의해 이루어진다. 교통의 특성상 차량운행이 적은 새벽시간을 이용해 낮 동안에 누적된 데이터를 처리하는 방법이 일반적인 경우에는 적합하지만, 뜻하지 않은 시스템 이상이나 비정상적인 데이터 발생으로 인해 DRDB 의 저장영역에 Overflow 가 발생할 것으로 예상되면 그 즉시 데이터를 백업하는 것이 시스템 중단을 막을 수 있다. 또한 이러한 백업전략은 시스템의 고 가용성을 제공하기 위해 시스템 중단 없이 변경이 가능하다.

4. 저장 알고리즘

유비쿼터스 교통정보시스템에서 처리하는 데이터는 대부분 주기적으로 생성되므로 저장 알고리즘은 기본적으로 RM 알고리즘을 이용한다. RM 알고리즘에서는 데이터의 생성주기가 짧을수록 Deadline 이 짧다. 교통은 실시간 정보 제공이 중요하므로 발생 주기가 빠른 데이터는 그렇지 않은 데이터에 비해 높은 우선순위를 부여 받아 빠르게 저장되어야 한다.

여러 종류의 메시지가 주기적으로 수신되면, 수신된 i 타입 메시지의 발생주기(P_i)를 미리 정해진 발생주기 임계 값($P_{threshold}$)과 비교하여 작으면 MMDB 에 바로 저장하고 그렇지 않으면 DRDB 에 저장한다. 즉 (그림 4) 에서와 같이, 발생주기가 특정 주기 이상으로 빠른 데이터는 MMDB 에 저장하고 그렇지 않은 데이터는 DRDB 에 저장한다.



(그림 4) 실시간 저장 알고리즘

MMDB 는 DRDB 에 비해 처리속도가 빠른 대신 저장공간의 크기가 제한적이라는 문제가 있다. 그러므로 대량의 데이터를 MMDB 만으로 처리할 수 없고, 주기적으로 MMDB 의 데이터를 DRDB 로 백업해야 한다. 데이터 백업은 교통 데이터가 적게 발생되는 시간에 주로 이루어지는 것이 효율적이지만, 비정상적인 대량의 데이터 발생에 대비하기 위해 특정시각 t 에서의 메모리 사용량 M(t) 가 메모리 사용 임계 값 ($M_{threshold}$) 이상이 되면 그 즉시 MMDB 의 데이터를

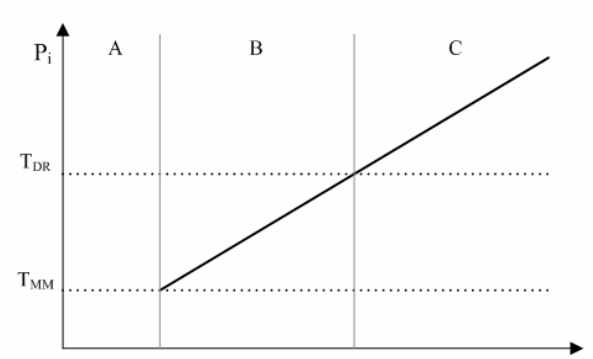
백업한다.

저장 작업이 MMDB 에서 수행될 것인지 DRDB 에서 수행될 것인지를 결정하는 $P_{threshold}$ 값은 시스템 상황에 맞게 동적으로 변한다. Database Manager 는 주기적으로 MMDB 와 DRDB 의 상태를 모니터링하고 그 결과(CPU 점유율, 저장영역 여유공간 등)를 바탕으로 $P_{threshold}$ 값을 결정한다. 만약 MMDB 저장영역이 부족하고 DRDB 의 경우 할당된 작업이 적어 여유가 있다면 $P_{threshold}$ 값을 높여 저장 작업이 DRDB 에서 처리될 수 있도록 한다.

5. 실시간 데이터 저장 성능향상

제안한 저장 알고리즘의 성능을 평가하기 위해 아래와 같은 내용을 가정하였다.

- 다양한 주기의 데이터가 존재한다.
- 데이터의 Deadline 은 데이터의 주기와 같다.
- 어떤 데이터 N 개에 대한 MMDB 의 처리시간(T_{MM}), DRDB 의 처리시간(T_{DR})은 일정하다.
- T_{MM} 보다 작은 주기를 갖는 데이터는 없다.



(그림 5) 메시지 주기와 처리 가능한 DBMS

위 가정하에 DRDB, MMDB 그리고 제안한 알고리즘을 이용하여 각각의 경우 Deadline 내에 작업이 완료되는지 확인했다.

(1) DRDB 만 사용할 경우, T_{DR} 이 데이터의 Deadline 보다 클 경우(B 구간), Deadline 내에 데이터를 처리할 수 없다.

(2) DRDB 와 MMDB 를 사용하고, 데이터 주기를 고려하지 않고 스케줄링을 하는 경우, 데이터의 주기 값이 $T_{MM} < P_i < T_{DR}$ 인 경우(B 구간) 해당 데이터가 DRDB 에서 할당될 경우 Deadline 내에 작업을 완료할 수 없다.

(3) MMDB, DRDB, 제안된 알고리즘을 사용할 경우, $P_{threshold}$ 값을 T_{MM} 값으로 설정하면, i 타입의 메시지(M_i)에 대해 $P_i < T_{DR}$ 를 만족하면 MMDB 에서 처리되고, 그렇지 않은 경우는 DRDB 에서 처리되므로 모든 메시지의 Deadline 을 지키면서 MMDB 의 메모리 자원을 효율적으로 사용한다.

6. 결론 및 향후 과제

u-ITS 환경은 개별 차량이 다양한 교통정보를 직접 생성하고 전송한다는 점에서 기존의 ITS 와 차이가 있다. 개별 차량이 데이터를 생성함으로써 대량의 데이터가 발생하므로 이를 실시간 저장하기 위한 시스템 구조와 알고리즘을 제안하였다.

시스템 구조는 크게 데이터 수신 모듈, 저장 모듈, 데이터베이스 관리 모듈, 백업 모듈, 데이터 생성 모듈로 나누어 지며, DBMS 는 MMDB 와 DRDB 를 같이 사용한다. 각 모듈은 Thread 기반으로 동작한다. 또한 데이터 수신 모듈이나, 데이터베이스 관리 모듈과 같이 병목현상이 예상되는 모듈은 Pooling 기법을 이용해 효율적인 처리가 이루어지도록 한다.

이후에, 우선 순위의 조절에 있어서 교통이라는 특수한 상황에 맞는 데이터 별 최적의 주기(P_i) 값과, 시스템 및 MMDB, DRDB 의 상태에 따른 주기 임계 값($P_{threshold}$) 조절 방법에 대한 연구 및 비주기적인 메시지의 중요도(Criticality) 할당 및 그에 따른 처리방법에 관한 연구가 계속되어야 할 것이다.

참고문헌

[1] 김태형, 강연수, 김원규, 오철, 박재용, 박정기, 고중협, 김준오, 김성민, “u-Transportation 교통정보 수집체계 개발”, 한국 ITS 학회지 1738-0901, 2007, 제 4 권 1 호, Page(s):17-24.

[2] Garcia-Molina, H., Salem, K., “Main memory database systems: an overview**”,** Knowledge and Data Engineering, IEEE Transactions on Volume 4, 6 Dec 1992, Page(s):509-516.

[3] Oracle Corporation, http://www.oracle.com

[4] Jane W. S. Liu, “Real-Time Systems”, Prentice Hall, 2000, Page(s):62-72, 115- 159.

[5] Krol Vaclav, Pokorny Jan “Design of V4DB - Experimental Real-Time Database System”, IEEE Industrial Electronics, IECON 2006 - 32nd Annual Conference on Nov. 2006, Page(s):126 -131.

[6] Kang Kyoung-Don, Sin Phillip H., Oh, Jisu, “A Real-Time Database Testbed and Performance Evaluation”, Embedded and Real-Time Computing Systems and Applications, 2007. RTCSA 2007. 13th IEEE International Conference, 21-24 Aug 2007, Page(s):319-326.

[7] Jing Huang, Gruenwald, L., “A data priority reload technique for real-time main memory databases”, Real-Time Systems, 1996., Proceedings of the Eighth Euromicro Workshop, 12-14 June 1996, Page(s):96 – 101.

[8] Son, S.H., “Real-time database systems: present and future “, Real-Time Computing Systems and Applications, 1995. Proceedings., Second International Workshop, 25-27 Oct 1995, Page(s):50-52.

[9] Abbott R., H. Garcia-Molina, “Scheduling Real-Time Transactions: A Performance Evaluation”, ACM Trans. on Database Systems, Sept 1992, vol. 17, no. 3, Page(s):513-560