PUBLICATIONS OF THE KOREAN ASTRONOMICAL SOCIETY
26: 37 ~ 44, 2011

ISSN 1225-1534 (Print)
DOI : 10.5303/PKAS.2011.26.1.037

중력렌즈 사건 측광 데이터베이스 및 프레임워크 개발
DEVELOPMENT OF THE PHOTOMETRY DATABASE AND FRAMEWORK FOR MICROLENSING EVENT

김동진, 이충욱, 김승리, 박병곤
한국천문연구원
D. -J. KIM, C. -U. LEE, S. -L. KIM, AND B. -G. PARK
Korea Astronomy and Space Science Institute, Daejeon 305-348, Korea
E-mail: keaton03@kasi.re.kr
(Received November 16, 2010; Accepted December 27, 2010)

ABSTRACT

We constructed a photometric database system which is optimally designed for microlensing events from KMTNet (Korea Microlensing Telescope Network) observation. We developed a framework software for the convenience of archiving, uploading, searching, and downloading of processed photometric data. From various tests for optimal data archiving engines, we found that the MyISAM storage engine shows the best performance. For the high performance of database system, data types of each field are carefully suggested from various combinations of tests especially to correct round-off errors. The developed framework provides the convenience of access to the database server using query forms via web pages, and displays the light curve of selected target for a quick view.

key words: instrument, data reduction; database: KMTNet

1. 서론

한국천문연구원은 2009년부터 2 m급 광시야 망원경과 초대형 CCD 카메라로 이루어진 외계행성 탐색시스템(Korea Microlensing Telescope Network, KMTNet) 개발을 수행해 오고 있다. 이 시스템은 지구형 외계행성 및 다양한 형태의 변광천체 발견을 위하여 칠레, 호주 및 남아프리카 공화국에 설치되어 우리은하 중심방향의 4° × 4° 영역을 24시간 연속 관측한다. 20 K × 20 K 크기를 가지는 초대형 CCD 카메라를 이용하여 하루에 생산되는 관측자료의 양은 약 600 GB 정도로 예측되며, 총 5년의 연구기간 동안 생산할 관측자료의 양은 대략 1.8 PB에 이른다. 이처럼 대용량의 관측자료를 생산하고 측광한 결과를 관리하기 위해서는 대용량의 데이터베이스 시스템 사용이 필수적이며, 자료의 효율적인 관리와 검색을 위해서는 시스템을 구성하는 각 하드웨어와 데이터베이스 엔진의 특성을 정확히 이해한 후 필요한 특성에 맞게 시스템을 구성하는 것이 매우 중요하다.

현재 우리은하 중심방향에서 미시 중력렌즈실험을 지속적으로 수행하고 있는 연구그룹은 크게 Optical Gravitational Lensing Experiment(OGLE, Udalski et al., 1992)과 Microlensing Observations in Astrophysics(MOA, Bond et al., 2001) 연구그룹이 있다. 이 두 연구그룹은 매일 관측 자료와 축적된 측광자료를 이용한 광도곡선을 홈페이지를 통하여 제공하고 있다. 특히 OGLE 그룹은 MySQL 데이터베이스를 사용하여 1996년부터 2000년까지 관측된 약 4천만 개의 별들에 대한 총 10억 개의 OGLE-II 측광 결과와 2001년부터 2009년까지 관측한 OGLE-III 변광성 자료를 제공하고 있다. OGLE-III의 경우 매년 3 TB의 관측 자료가 생성되었는데(Udalski et al., 2008), 이는 KMTNet이 단 5일 동안 생산하는 자료의 양과 동일하다. 이와 더불어 별빛가림 현상을 관측하여 외계행성을 탐색하는 Super Wide Angle Search for Planets(SuperWASP)도 하룻밤에 100 GB의 관측자료를 생산하며, MySQL 데이터베이스를 이용하여 3년 동안 관측된 100 TB의 관측자료와 30 TB의 측광자료를 관리하고 있다(Pollacco et al., 2006).

이러한 대용량 관측자료는 각각의 연구 목적에 맞는 측광파이프라인을 거친 후 결과를 데이터베이스에 저장한다. 매일 생산된 측광 자료가 누적될수록 데이터베이스 테이블의 크기 또한 증가하여 자료의 검색시간 또한 증가하게 된다. KMTNet은 다른 탐색 프로젝트에 비해

측광자료의 크기와 증가량이 방대하기 때문에 단일 데이터베이스 테이블을 사용하는 대신 일정한 크기의 관측영역별로 나누어진 여러 개의 데이터베이스 테이블을 이용하여 시스템을 구성하려는 전략을 가지고 있다(김동진 등, 2009).

이 연구에서는 여러 개의 데이터베이스 테이블을 이용하여 시스템을 구성할 때 필요한 대용량 저장매체의 선택과 자료의 입출력 및 검색 속도 등에 대한 충분한 실험을 통해 어떤 시스템의 구성이 효율적인지 알아보려 한다. 이를 위하여 저장매체와 데이터베이스의 특성을 조사하고, 각각의 조합을 통해 구성된 시스템의 성능을 비교하였다. 2장에서는 MySQL의 검색엔진, 자료형, 실제 측광자료를 이용한 성능 평가에 대하여, 3장에서는 KMTNet 측광파이프라인의 결과를 데이터베이스에 입력하기 위하여 개발한 데이터베이스 프레임워크의 구성에 대하여 기술한다.

2. 데이터베이스 구성

자료의 양이 많아지고 검색하는 사용자가 증가할수록 데이터베이스의 검색속도는 감소하게 된다. 이러한 문제가 발생하지 않기 위해서는 데이터베이스의 시스템을 효율적으로 구성하여야 한다. 효율적인 데이터베이스의 구성은 데이터베이스 저장엔진의 선택과 테이블 구조, 자료 형식, 저장장치의 설계로 이루어진다. 특히 데이터베이스의 선택은 매우 중요하다. 현재 대용량 데이터베이스를 구축할 수 있는 relational database management system(RDBMS)으로 ORACLE, MySQL, MS-SQL, CUBRID 등이 사용되고 있다. ORACLE과 MS-SQL은 이미 안정성과 빠른 속도가 검증되어 여러 기업에서 사용하고 있는 제품이다. 이미 김동진 등(2009)은 ORACLE을 이용하여 측광데이터베이스를 설계하고 모의실험을 수행한 바 있으나, 제품의 높은 가격으로 초기 구입비용과 유지보수 비용 문제가 있어 이 연구에서는 공개 데이터베이스로 널리 사용되고 있는 MySQL을 이용하여 시스템을 구성하고자 한다. MySQL은 사용자가 많아 운영에 관한 정보를 얻기 쉬운 장점이 있고 이미 OGLE, MOA, SuperWASP 등 여러 연구 그룹에서 사용하고 있는 안정성이 검증된 제품이다. 이 장에서는 MySQL의 MyISAM 저장엔진과 InnoDB 저장엔진의 특성에 대하여 정리하고, HDD와 SSD를 조합하여 구성한 저장매체의 성능비교 및 조합된 시스템의 성능에 대하여 기술한다.

2.1. 데이터베이스 저장엔진

데이터베이스 저장엔진은 데이터베이스 테이블을 관리하기 위한 핵심기능으로 저장엔진의 능력에 따라 데이터베이스의 검색 성능이 결정된다. MySQL은 여러 종류의 저장엔진을 지원하지만 그중 MyISAM과 InnoDB 저장엔진이 가장 널리 사용되고 있다.

MyISAM 저장엔진은 MySQL의 기본 저장엔진으로 블로그나 게시판처럼 한사람이 올린 글을 다른 많은 사람들이 접근할 때 최적의 성능을 발휘한다. 그러나 트랜잭션(transaction)[1] 기능이 없어 잘못된 작업을 수행할 경우 되돌리는 것(roll back)이 불가능하고 자료의 추가(insert), 갱신(update) 및 삭제(delete) 명령을 실행할 경우 테이블 자체를 잠그고(table lock) 작업을 하기 때문에 작업이 종료되고 잠금이 풀릴 때까지 검색이 불가능하여 검색 대기시간이 증가할 수 있는 단점이 있다. 그러나 검색 성능은 아주 우수하다.

InnoDB 저장엔진은 커밋(commit)[2]과 되돌리기(roll back) 기능이 포함된 트랜잭션과 외부키(Foreign Key)[3] 참조 등의 기능을 지원하고, 테이블 스페이스를 도입하여 저장장치 상의 폴더가 아닌 데이터베이스가 생성한 테이블 스페이스에 테이블을 생성할 수 있다. 추가, 갱신 및 삭제 등의 작업을 수행할 경우 MyISAM 저장엔진과 달리 테이블을 한줄 씩 잠그고(row level lock) 작업을 실행한다. 만약 100줄의 새로운 자료를 데이터베이스에 추가한다면 각각의 줄을 추가할 때마다 잠금 기능이 실행되어 작업 시간이 길어진다는 단점이 있지만 테이블 자체가 잠기지 않아 검색 대기시간이 짧은 장점이 있다. 검색 성능은 MyISAM 저장엔진에 비해 빠르지 않다.

2.2. 테이블 구성과 자료 형식

데이터베이스의 테이블은 하나 이상의 레코드로 구성되고 각 레코드는 필드로 구성된다. 이때 각각의 필드에 대한 자료형을 미리 지정하여야 한다. MySQL은 다양한 자료형을 제공하며 각각의 형식에 따라 데이터베이스 테이블의 크기가 달라진다. 수치를 저장하기 위한 자료형은 일반적으로 정확한값, 부동소수점, 비트의 세 형식으로 분류 된다(DuBois, 2009). 숫자를 표시하는데 있어 정확한 값은 ‘integer’와 ‘decimal’[4]이 있으며, 부동소수점은 ‘float’과 ‘double’이 있다.

---

[1] 요청한 작업을 완수하기 위한 기본 단위. 되돌리기 작업(roll back)은 트랜잭션을 기준으로 수행된다.

[2] 트랜잭션이 완료되었다고 판단되는 시점에서 종료를 요구하는 동작. 커밋 명령을 실행해야 갱신한 자료가 데이터베이스에 저장되고 lock이 해제된다.

[3] 테이블 사이에 관계를 관장하는 값으로 데이터의 일관성을 유지할 수 있도록 한다.

[4] decimal은 고정소수점 형으로 입력된 실수를 문자형으로 저장한다.

표 1. 측광레코드의 구성

필드명	자료형	크기
hjd	decimal(12,8)	7Byte
star_id	smallint(5)	2Byte
num_bad_pix	smallint(4)	2Byte
pf_flux	double(20,4)	8Byte
pf_flux_err	double(17,4)	8Byte
ap_flux	decimal(13,4)	6Byte
ap_flux_err	decimal(8,4)	4Byte
bkg	decimal(10,4)	7Byte
psf_chi2	decimal(14,4)	8Byte
fwhm	decimal(8,4)	7Byte

데이터베이스 테이블의 구성은 관측자료를 처리하여 얻는 측광정보의 내용에 따라 측광테이블과 좌표테이블 2가지로 구성하였다. 측광테이블을 구성하는 각각의 측광레코드는 하나의 천체에 대하여 각각의 관측시각에 대한 등급과 관련된 정보로 이루어지며, 관측자료를 영상차감 측광파이프라인으로 처리하여 얻는다. 한편 좌표테이블을 구성하는 각각의 좌표레코드는 영상차감 측광파이프라인에서 사용하는 기준영상을 측광하여 얻는다(Lee et al., 2010).

측광파이프라인을 통해 측광된 결과를 분석하여 측광레코드를 구성하는 각 필드의 자료형과 크기를 표 1에 정리하였다. 특히 저장공간을 절약하기 위해서 각각의 자료형은 실제 측광된 결과를 참조하였다. 테이블의 크기가 커지지 않도록 하기 위하여 각 필드의 범위와 정확도를 고려하여 최소의 크기가 되도록 결정하였다. 자료형에 나타난 각각의 숫자는 전체 자릿수와 소수점 이하 자릿수를 나타낸다.

좌표레코드는 관측영역 내에서 검출된 전체별의 위치 및 표준등급 정보를 가지며 각각의 필드와 자료형을 표 2에 정리하였다. 측광레코드와 좌표레코드에서 공통으로 사용되는 ‘star_id’처럼 값의 범위가 크지 않은 경우에는 ‘smallint’을 사용하였고, 좌표는 ‘time’을 사용하였다. 실수형 필드를 표현하기 위해서는 float, double, decimal을 사용할 수 있다. 그러나 이 중 float은 round-off error가 발생할 수 있기 때문에 사용에서 제외하고 double과 decimal을 사용하였다.

2.3. 데이터베이스의 입출력과 검색 모의실험

구성한 데이터베이스의 성능을 평가하기 위해 MOA 그룹이 2006년 3월 29일부터 2007년 9월 16일까지 관측한 R-band 자료를 사용하였다. 이충욱 등(2009)이 구성한 영상차감 측광파이프라인을 이용하여 관측자료를 처리하였다. 자료처리는 다중연산을 통한 원활한 자료처리 및 데이터베이스 테이블의 분산화를 위하여 2 K × 4 K 영역을 32개의 512 × 512영역으로 분할한 후 7개의 core를 사용하여 이루어 졌다. 측광과정이 모두 끝난 후 우리는 약 19만 개의 좌표레코드와 약 3억 개의 측광레코드를 얻을 수 있었고, 이 중에서 측광이 잘된 약 2,500만 개의 측광자료를 이용하여 데이터베이스의 입출력과 검색 실험을 수행하였다. 2,500만 개의 측광자료 개수는 512 × 512 크기로 나눈 영역에서 충분한 S/N비를 갖는 천체가 약 4,000개 측광된다고 가정할 때 KMTNet이 집중 관측하는 4° × 4° 영역에서 한 번 관측을 통해 생산되는 측광레코드에 해당한다(한정호, 2010). 만약 더욱 어두운 천체의 측광까지 포함시킨다면 측광데이터의 양은 더욱 증가할 수도 있지만, 밝기가 수십에서 수백 배씩 밝아지는 중력렌즈현상의 특성을 고려할 경우 적당한 양으로 생각된다.

표 2. 좌표레코드의 구성

필드명	자료형	크기
star_id	smallint(5)	2Byte
x	decimal(5,2)	2Byte
y	decimal(5,2)	2Byte
ra_dms	time	3Byte
dec_dms	time	3Byte
ref_mag	dicimal(5,2)	2Byte
ref_mag_err	decimal(5,2)	2Byte

표 3. 실험에 사용한 서버의 사양

	서버 사양
CPU	Intel Core i7 920 2.66 GHz
RAM	6 GB(DDR3)
HDD	500 GB SATA2 HDD×4 140 GB SSD×3
RAID Controller	Highpoint RocketRAID 2640×4
OS	CentOS 5.5(XFS filesystem)
Database	MySQL 5.0.77-4
PHP	PHP5 5.1.6-27

본 연구를 위해 구성한 하드웨어는 표 3과 같다. 저장장치의 구성에 따른 성능을 비교하기 위하여 HDD와 SSD 저장장치를 단독으로 사용할 때와 RAID-5 형식으로 구성할 때를 비교 실험하였다. SSD는 HDD와 달리 구동부가 전혀 없고 플래쉬 메모리만으로 이루어진 저장 장치로 HDD 대비 데이터 입출력 및 접근 속도가 매우 빠르나 용량대비 가격이 HDD에 비해 20배 ~ 30배 정도 비싸다는 단점이 있다. 한편 redundant array of independent disks(RAID)는 자료를 분할하여 여러 개의 저장장치에 병렬로 저장하는 방식으로 입출력을 분산시키기 때문에 전체적인 읽기/쓰기 성능이 향상되고 구성

표 4. 입출력 속도 벤치마크

구분	SSDx1	HDDx1	HDDx1 (spanning)	HDDx4 (RAID-5)
초당 요청개수 (Requests/sec)	1810.22	78.93	136.66	54.99
초당 처리량 (MB/sec)	28.285	1.2332	2.1353	0.87987

하는 RAID 형태에 따라 접근 속도와 데이터 보존 신뢰도가 구분된다. RAID-5는 자료의 패리티(parity) 정보를 모든 디스크에 분산 저장하는 방법으로 보존 신뢰도가 높으며 최소 3개의 디스크를 이용하여 시스템이 구성된다.

실험에 앞서 각 저장장치들의 파일 입출력 속도의 비교를 위하여 표 4에 Sysbench 프로그램을 이용한 벤치마크 결과를 나타내었다. 이 실험은 100 GB의 더미 데이터를 생성한 후 무작위로 읽기/쓰기를 반복하여 저장장치가 초당 처리할 수 있는 작업의 요청수와 초당 처리량이 얼마나 되는지 확인하였다. 이러한 파일 입출력 벤치마크는 I/O에 부하가 걸리는 상황에서 시스템이 얼마나 잘 작동하는지를 나타낸다. 보통의 경우 RAID-5 구성은 여러 디스크를 이용하여 분산 작업하기 때문에 단일 HDD를 사용할 때보다 빠른 속도를 보이지만, 이 테스트에서는 오히려 더 낮은 성능을 보였다. 그러나 메인보드의 SATA 컨트롤러에 연결한 단일 HDD와 RAID 컨트롤러에 spanning[5]한 단일 HDD의 결과와 비교하면 오히려 RAID 컨트롤러에 spanning한 단일 HDD가 약 80% 이상 높은 성능을 보였다. 이는 실험에 사용한 저가의 RAID 컨트롤러의 제어성능에 기인한 것으로 추측된다. 즉, 개별 HDD를 제어할 때에는 메인보드의 SATA 컨트롤러보다 우수하지만 RAID-5 구성에서는 제어 성능이 좋지 않아 이러한 속도차를 발생시킨 것으로 추측되며, 컨트롤러의 업그레이드를 통하여 성능이 향상되리라 기대한다. 한편 SSD의 경우 단일 HDD에 비해 약 20배 이상의 빠른 성능을 보였다.

우리는 데이터베이스에서 검색 속도를 빠르게 하기 위한 B-Tree index 작업을 사용했을 때와 사용하지 않았을 때 데이터베이스 저장엔진과 데이터의 자료형에 따라 생성되는 데이터베이스 테이블의 크기를 비교하여 표 5에 정리하였다. index 작업은 질의문에서 가장 사용빈도가 높은 'hjd', 'star_id' 필드에 대해 실시하였고, 정렬이나 검색 제한을 위해 'pf_flux', 'ap_flux' 필드도 추가하였다. index를 사용하지 않았을 경우 MyISAM이 InnoDB에 비해 테이블의 크기가 30% 작았고, index 사용 시에는 80% ~ 90%까지 차이가 발생하였다. 자료형

[5] spanning: RAID 카드에는 연결되어 있으나, RAID 구성에서 제외된 저장장치를 가리키는 용어.

표 5. 저장엔진과 자료형에 따른 테이블 크기 비교

index	저장엔진	자료형	테이블크기 (MB)	인덱스크기 (MB)	전체크기 (MB)
미사용	MyISAM	double	2,567.73	-	2,567.73
		decimal	2,025.92	-	2,025.92
		optimal	1,978.80	-	1,978.80
	InnoDB	double	3,444.00	-	3,444.00
		decimal	2,828.00	-	2,828.00
		optimal	2,567.73	-	2,567.73
사용	MyISAM	double	2,567.73	743.94	3,311.67
		decimal	2,025.92	918.49	2,944.41
		optimal	1,978.80	864.64	2,843.45
	InnoDB	double	-	6,184.00	6,184.00
		decimal	-	5,476.00	5,476.00
		optimal	-	5,360.00	5,360.00

에 따라 테이블의 크기도 변화를 보였다. 모든 자료형을 double형으로 사용했을 때에 비하여 decimal형을 사용하였을 경우 약 20% 정도 크기가 작았다. 한편 표 1과 같이 decimal형과 double형을 혼합하여 구성한 경우에 가장 작은 크기를 가졌으며, 이 실험을 통해 전체적으로 MyISAM 저장엔진이 InnoDB에 비해 동일한 공간에 더 많은 데이터를 저장할 수 있음을 확인할 수 있었다.

표 6은 동일한 실험을 단일 HDD와 SSD 저장장치를 이용하여 구성하였을 때와 RAID-5로 구성하였을 때에 대한 자료의 입출력 속도 및 검색 속도를 나타낸다. 단일 HDD에 데이터베이스 테이블을 설치하고 측광자료를 전송할 때 index를 하지 않으면 MyISAM 저장엔진이 InnoDB에 비해 약 10배 빨랐고, index를 할 때 MyISAM 저장엔진이 InnoDB보다 500배 더 빨랐다.

자료형에 따른 입력 속도는 모든 필드를 double로만 사용할 때 가장 빨랐고 decimal과 표 1에 정의된 자료형을 사용할 때 서로 속도가 비슷하였으며 double형에 비해 약 2배 정도 속도가 느렸다.

약 2,500만개의 측광자료 중 임의의 star_id에 해당하는 값을 검색 하였을 때 검색 속도는 index를 사용하지 않았을 경우 decimal과 표 1에 정의된 자료형을 사용할 때가 double 만을 사용했을 때 보다 약 30% 빨랐고, index를 사용한 경우 저장엔진과 자료형에 따른 검색속도의 차이는 발행하지 않았다.

단일 SSD를 사용하였을 경우 InnoDB 저장엔진의 성능이 크게 향상됨을 볼 수 있었다. InnoDB 저장엔진은 single HDD의 입력 속도와 비교하면 index를 하지 않았을 때 약 2배, index를 할 때 약 9배의 속도 향상이 있었지만, MyISAM 저장엔진은 단일 HDD의 입력 속도

표 6. 하드웨어에 따른 데이터베이스 자료 입력 및 검색시간

index	저장 엔진	자료 형식	HDD×1		SSD×1		SSD×3 (RAID-5)		HDD×4 (RAID-5)	
			Data upload Time(sec)	Searching Time(sec)	Data upload Time(sec)	Searching Time(sec)	Data upload Time(sec)	Searching Time(sec)	Data upload Time(sec)	Searching Time(sec)
X	MyISAM	double	63.61	31.55	63.61	11.58	63.50	9.02	63.72	9.53
		decimal	94.79	24.85	93.51	9.16	94.20	7.02	93.41	6.95
		optimal	92.07	24.94	90.88	8.99	93.37	8.04	92.37	8.48
	InnoDB	double	945.56	38.39	472.64	36.70	212.05	30.35	1,197.14	31.71
		decimal	836.72	34.97	422.7	33.37	271.28	27.89	1,056.99	28.38
		optimal	792.98	31.70	397.62	33.49	209.89	28.01	1,056.99	28.13
O	MyISAM	double	205.25	0.44	223.01	0.46	209.89	0.26	235.53	0.32
		decimal	266.06	0.34	289.19	0.42	271.28	0.27	285.07	0.29
		optimal	206.27	0.35	236.32	0.56	212.05	0.26	**234.57**	**0.30**
	InnoDB	double	-	-	19,465.40	0.78	-	-	96,099.28	0.35
		decimal	-	-	18,606.79	0.75	-	-	94,843.39	0.31
		optimal	128,084.08	0.36	18,176.70	0.76	19,254.84	0.29	94,124.25	0.32

와 비교할 때 index 작업유무에 관계없이 큰 차이가 없었다. MyISAM의 검색속도는 index 작업이 없는 경우 단일 HDD의 검색속도에 비해 2.7배 빨라졌고, index 작업을 수행한 경우에는 차이가 없었다. 그러나 InnoDB의 경우 index 유무에 관계없이 검색속도의 향상은 없었다. InnoDB 저장엔진은 자료를 입력할 때 한줄씩 잠그고 작업하기 때문에 입력하는 자료가 동일해도 MyISAM 저장엔진보다 더 많은 작업을 수행하게 되어 자료접근 속도가 빠른 SSD를 사용할 때 성능이 향상됨을 알 수 있었다.

3개의 SSD를 RAID-5로 구성하여 자료를 저장할 경우 MyISAM 저장엔진의 저장속도는 index 사용 유무와 관계없이 단일 SSD와 큰 차이를 보이지 않았다. InnoDB 저장엔진은 index를 사용하지 않았을 경우 단일 SSD 대비 80% 정도의 향상이 있으나 index를 사용한 경우에는 단일 SSD보다 10% 정도 느린 결과를 보였다. 검색속도는 index를 사용하지 않은 MyISAM의 경우 단일 SSD 대비 약 10%, InnoDB는 20% 정도 향상되었고 index를 사용한 경우 MyISAM은 20%, InnoDB는 40% 정도의 속도가 향상되었다.

4개의 SATA HDD를 RAID-5로 구성하여 사용하였을 경우 MyISAM 저장엔진은 index사용 유무에 관계없이 단일 HDD의 결과와 큰 차이가 없었다. InnoDB 저장엔진은 index를 사용하지 않고 자료를 입력할 경우 단일 HDD보다 20%의 속도 하락을 보였으나 index를 사용할 경우 속도가 20% 상승 하였다. 검색속도는 index를 사용했을 때 단일 HDD 보다 속도가 3배 빨라졌다. index를 사용하지 않고 자료를 입력할 때 속도가 감소한 원인은 표 4의 벤치마크에서 추정한 RAID 컨트롤러의 RAID-5 제어 성능이 높지 않았기 때문으로 분석된다.

HDD와 SSD를 RAID-5로 조합 구성하여 실험한 결과 MyISAM 저장엔진은 저장장치의 성능 및 index 유무와 관계없이 일정한 속도를 유지하였지만 InnoDB 저장엔진은 저장장치의 성능에 크게 의존하는 결과를 얻었다. 만약 실시간 자료처리를 통한 측광자료의 저장을 고려한다면 10분마다 생산되는 측광자료의 입력에 5시간 ~ 35시간이 소요되는 InnoDB는 적절한 선택이 될 수 없다. MyISAM 저장엔진의 경우에는 빠른 검색에 필요한 index를 수행하고도 4분 이내에 측광자료의 입력이 가능함을 확인하였다. 따라서 5년간 축적되는 180 TB 정도의 대용량 측광 데이터베이스를 구축하는 데 있어 한정된 예산을 고려한다면 여러 개의 HDD를 RAID-5를 이용하여 구성한 저장장치가 적당함을 확인하였다.

3. 프레임워크

24시간 연속 관측을 통해 생산되는 대용량의 측광자료를 측광데이터베이스에 기록하고 관리하기 위한 KMTNet 프레임워크를 개발하였다. 이는 지정된 시간에 측광파이프라인을 통하여 생산되는 측광자료를 수집하여 데이터베이스에 입력하거나 특정 천체에 대한 모든 측광레코드를 출력하기위해 개발된 프로그램으로 크게 'KMTNet_SW'와 'KMTNet_SEARCH'로 구성된다. 각 프로그램은 추가 기능 확장이 쉽도록 모듈로 구성하였다. 그림 1은 KMTNet 프레임워크의 구성도이다.

3.1. KMTNet_SW

KMTNet_SW는 총 5개의 폴더로 구성된다. 이중 'KMTNet_ENV'와 'SHEL'은 실행프로그램이 있는 폴더이고 'FTP', 'DROPBOX', 'ARR_DATA' 폴더는 측광자

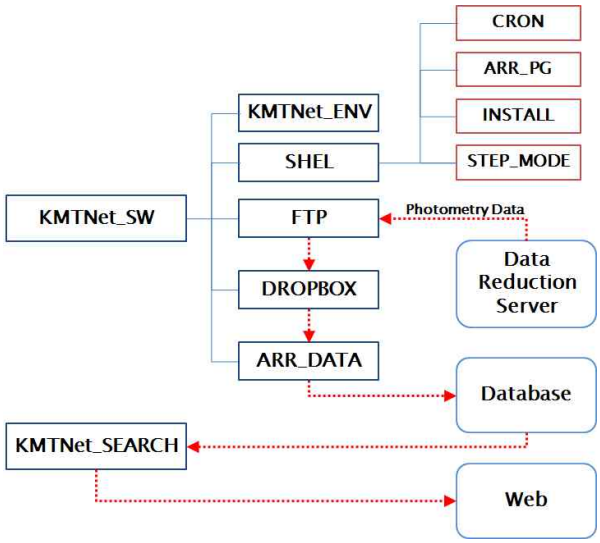

그림 1. KMTNet 프레임워크 구성도.

료를 수집하고 데이터베이스에 저장하는 공간이다. KMTNet_ENV 폴더는 KMTNet_SW에서 공통으로 사용하는 변수가 저장되어 있고, SHEL 폴더는 KMTNet_SW의 프로그램 파일들이 저장되어 있는 곳으로 프레임워크 설치를 위한 'INSTALL' 폴더, 자동수행을 위한 daemon 프로그램이 있는 'CRON' 폴더, 측광자료를 재배열하기 위한 'ARR_PG' 폴더, 수동 작업을 위한 'STEP_MODE' 폴더 등으로 구성된다.

KMTNet_SW는 기능상으로 측광자료의 수집과 데이터베이스 입력으로 구분되며, 측광자료의 수집은 측광자료처리 서버에서 측광 결과가 생산되면 KMTNet_SW의 FTP 폴더로 결과를 전송하여 목록을 작성한다. 측광자료의 저장을 위해서 프레임워크의 수집과정을 거치지 않고 직접 자료처리 서버가 데이터베이스에 접속하여 측광자료를 입력할 수도 있지만, 데이터베이스 테이블이 있는 하드디스크에서 측광자료를 입력하는 것이 데이터베이스 및 네트워크 사용에 안정적이므로 이 방법을 선택하였다.

1 Gbps의 네트워크 환경에서 자료처리 서버에서 직접 데이터베이스 서버에 접속하여 측광자료를 MyISAM 저장엔진 테이블에 입력하면 index 작업을 수행하지 않았을 경우 220초가 소요되었고 index 작업을 수행할 경우 약 320초가 소요되었다. 이는 프레임워크를 이용하여 측광자료를 수집하고 데이터베이스에 저장하는 시간과 거의 비슷하다. 그러나 전자의 경우 네트워크를 통해 데이터베이스에 자료를 입력하는 동안 네트워크 장애가 발생하면 데이터베이스에 불완전한 자료가 저장되고 이런 문제는 추후 테이블의 손상으로 연계되어 데이터베이스의 운영에 큰 차질을 초래할 수 있다. 따라서 안정적인 데이터베이스 운영을 위하여 프레임워크를 이용하여 데이터베이스 서버가 측광자료를 수집하는 방법을 택하였다.

표 7. FLAG 값과 의미

FLAG	의미
0	Photometry file exist
1	Photometry file transferring
2	Photometry file not exist
3	Photometry file moved
4	Arranging success
5	Arranging process failed
6	Data Upload Success
7	Upload Failed

FTP 폴더로 전송된 자료들은 CRON 폴더에 있는 daemon 프로그램의 감시를 받는다. 이 daemon 프로그램은 c-shell로 개발되었고 리눅스 서버의 'crontab'에 의해 매 1분 간격으로 실행되며, 관측 자료가 FTP 폴더에 들어왔는지 또는 전송되고 있는지를 확인한다. crontab 방식은 시간에 기반을 둔 Unix/Linux의 작업 스케줄러 프로그램으로 천리안 위성의 기상자료처리 프로그램(안명환 등, 2008)에서 비슷한 목적으로 구현되어 사용되고 있다. daemon 프로그램은 측광자료가 정상적으로 전송되었다고 판단하면 FTP 폴더에서 DROPBOX 폴더로 파일을 이동하고 이후 데이터베이스에 입력할 수 있는 형태로 관측자료를 가공하여 업로드할 측광자료의 무결점 검사를 수행한다.

현재 구축된 측광파이프라인을 통해 생성되는 파일은 각 별의 등급, 시각, 좌표 등 3가지 종류로 이를 각각의 테이블에 넣을 경우 관측 영역별로 테이블의 수가 많아지고 다중 테이블 검색을 하면 2번의 'join'이 이루어져야 한다. join은 복수의 테이블에서 자료를 검색할 때 사용하는 SQL 질의명령의 하나로, 사용하는 join의 수가 많아질수록 질의명령이 복잡해지고 검색 속도 또한 증가하므로 join의 사용량을 줄이는 것이 좋다. KMTNet 데이터베이스에서는 테이블의 용량을 고려하여 관측 시간값과 좌표값을 하나의 테이블을 사용하도록 결과 파일을 가공하였고 KMTNet_SEARCH에서 검색할 경우 join을 사용하지 않도록 하였다. 또한 자료처리 과정에서 출력되는 계산 불가능한 값을 '-999'로 표현하였다. 특히 연산과정에서 출력되는 'nan'과 같은 문자열은 데이터베이스의 float, double 및 decimal 타입 자료형에 입력되지 않기 때문에 이러한 가공 과정을 거치지 않는다면 자료 입력 과정에서 에러가 발생하게 된다. 또한 측광자료를 스크립트를 이용하여 일괄적으로 입력하려면 입력되는 모든 컬럼을 일정한 공백이나 큰 따옴표 등으로 구분해줘야 하는데, 이러한 과정이 포함된다.

모든 가공 과정을 거친 측광자료는 ARR_DATA 폴

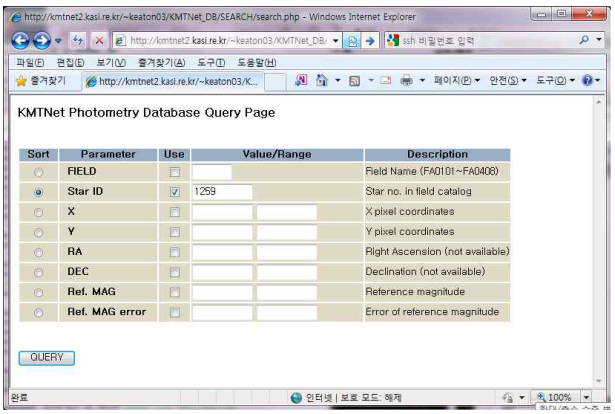

그림 2. KMTNet 검색 페이지.

더로 저장하고 DROPBOX 폴더에 있던 측광자료를 삭제한다. ARR_DATA 폴더에 가공된 측광자료가 저장되면 이 폴더를 감시하고 있던 daemon 프로그램이 관측자료를 데이터베이스로 입력한다. 이런 일련의 과정에 대한 'FLAG' 값을 데이터베이스에 출력하여 웹에서 현재 진행상황 및 상태를 파악할 수 있다. 표 7은 출력되는 각 FLAG 값의 정보를 나타낸다.

3.2. KMTNet_SEARCH

KMTNet_SEARCH는 웹페이지를 이용하여 측광데이터베이스의 자료를 빠르고 쉽게 검색하기 위한 프로그램으로 PHP로 개발하였다. 이 프로그램을 웹서버의 웹서비스가 가능한 폴더에 복사한 후 데이터베이스의 계정과 사용 테이블을 설정하여 사용한다. 그림 2는 KMTNet 프레임워크의 검색 페이지이다.

KMTNet_SEARCH는 주어진 관측영역을 확인한 후 이에 대응하는 데이터베이스 테이블을 연결하여 별의 ID, 좌표, 기준 등급 등을 참조 검색하여 원하는 천체의 측광레코드를 찾은 다음 측광자료를 화면에 출력하거나 또는 파일의 형태로 제공한다. 이 서비스를 통하여 각각의 연구자들은 자신의 연구실에서 측광데이터베이스에 접근하여 각 필드에 정보를 입력한 후 하단의 쿼리 버튼을 눌러 원하는 측광자료를 얻을 수 있다. 측광자료의 대략적인 모습을 보여주기 위하여 광도곡선을 화면에 출력한다. 한편 많은 양의 측광자료를 검색하고 연구해야 하는 연구자는 웹기반의 검색 방법 이외에 SQL 문법을 사용하는 C, FORTRAN 또는 Perl, Python 등의 스크립트를 이용하여 측광데이터베이스에 직접 접근하여 자료의 추출이 가능하다.

4. 결론 및 토의

우리는 중력렌즈현상 측광데이터베이스 시스템을 효율적으로 구성하기 위해 데이터베이스 저장엔진과 HDD 또는 SSD의 특성에 대하여 정리하고, 각 구성 요소의 조합을 통해 데이터베이스 시스템을 구성하고 이들의 성능을 비교 분석하였다. 저장매체의 불필요한 낭비를 막기 위하여 실제 측광자료의 값을 참조하여 결정한 각 필드의 자료형을 표 1에 정리하였다.

다양한 조합으로 이루어진 데이터베이스 시스템의 성능 비교실험을 통해 HDD를 RAID-5로 구성한 저장장치와 MyISAM 저장엔진을 사용한 데이터베이스 시스템의 효율이 가장 높음을 확인하였다. 한편 실험을 통하여 MyISAM 저장엔진은 데이터베이스를 구성하는 하드웨어 성능에 크게 의존하지 않았던 반면 InnoDB 저장엔진은 하드웨어 성능에 크게 의존함을 확인할 수 있었다.

KMTNet이 본격적으로 가동하게 되어 생산해 내는 양과 비슷한 측광 자료를 이용하여 측광자료를 데이터베이스에 저장하고 검색하는데 충분한지 검증하였다. KMTNet은 고유 연구주제와 관련하여 매 10분마다 한 번씩 측광자료를 생산하게 되는데, 이 자료를 MyISAM 저장엔진으로 index 작업을 수행하여 측광데이터베이스에 입력할 경우 약 4분 정도의 시간이 소요됨을 확인하였다. 따라서 KMTNet의 측광자료가 측광파이프라인을 이용하여 4분 ~ 5분 이내에 처리되어 생산된다면 실시간 측광자료의 서비스 또한 가능하리라 생각된다. 한편 이를 위해서는 여러 개의 CPU를 사용하는 다중연산 시스템이 필수적이라 생각된다.

측광자료를 데이터베이스에 자동으로 입력하고, 데이터베이스의 원할한 운영을 위한 프레임워크 소프트웨어를 개발하였다.

현재 모의실험을 통해 예측된 KMTNet의 전체 데이터베이스 크기는 약 180 TB이며 하루에 생산되는 데이터베이스의 크기는 약 240 GB로 예측된다. 데이터베이스의 크기가 계속 증가할수록 누적된 자료와 새로운 자료를 다시 index해야 하는 문제도 발생하리라 생각된다. 따라서 자료의 입력시간은 표 6에서 보인 시간보다 더 증가할 것으로 예상된다. 이러한 문제점을 해결하기 위해서 우리는 단일 데이터베이스 테이블을 사용하는 대신 데이터베이스 분산화를 위해 미리 정의된 관측영역별로 데이터베이스 테이블을 만들어 사용할 계획이다. 이와 더불어 시스템의 근본적 성능향상을 위한 MySQL 데이터베이스의 버퍼메모리 확장, 파라미터 튜닝 및 데이터베이스 병렬화 등의 다양한 시험이 추가적으로 요구된다.

이 연구에서는 중력렌즈 현상 검출에 초점을 둔 측광데이터베이스의 구성에 대해 실험과 개발이 이루어졌다. 따라서 소행성이나 지구접근천체 등 좌표가 항시 변하는 천체에 대해서는 측광자료의 정확한 입력과 출

력 및 검색이 어려운 실정이다. 이와 더불어 관측이 계속 진행되면서 앞으로 발견하게 될 초신성이나 신성, 감마선폭발체 등 빠른 분석을 요구하는 천체에 대해서는 별도의 특성화된 측광데이터베이스의 구성과 관리가 요구된다. 또한 수많은 측광자료를 체계적으로 분류하고 분석하는 자동화된 분석체계의 도입이 매우 중요하며 시급하다고 생각된다. 따라서 이에 대한 심도 깊은 연구가 뒤따라야 한다고 생각한다.

참고 문헌

김동진, 이충욱, 김승리, 박병곤, 이재우, 2009, KMTNet 자료처리 시스템 설계와 측광데이터베이스 구축, 천문학논총, 24, 83

안명환, 오성남, 최병철 등, 2008, 통신해양기상위성 전·후처리 시스템 및 인터페이스, 국립기상연구소 기술보고서

이충욱, 박병곤, 김승리, 이재우 등, 2009, 영상차감법을 이용한 대용량탐색자료처리 파이프라인 개발, 한국천문연구원 기술보고서(No. 20090238)

한정호, 2010, 개인서신

Bond, I. A., Abe, F., Dodd, R. J., et al., 2001, Real-Time Difference Imaging Analysis of MOA Galactic Bulge Observations During 2000, MNRAS, 327, 868

DuBois, P., 2009, MySQL (4th Ed.; Addison Wesley)

Lee, C. -U., Koo, J. -R., Kim, S. -L., Lee, J. W., Park, B. -G., & Han, C., 2010, Detection of Variable Stars in the Open Cluster M11 Using Difference Image Analysis Pipeline, JASS, in press

Pollacco, D. L., Skillen, I., Cameron, A. C., et al., 2006, The WASP Project and the SuperWASP Cameras, PASP, 118, 1414

Udalski, A., Szymanski, M., Kaluzny, J., Kubiak, M., & Mateo, M., 1992, The Optical Gravitational Lensing Experiment, Acta Astronomica, 42, 253

Udalski, A., Szymanski, M., SoszynskiI., & Poleski, R., 2008, The Optical Gravitational Lensing Experiment. Final Reductions of the OGLE-III Data, Acta Astronomica, 58, 69