Performance Assessment of Large Astronomical Databases

A comparison of row-oriented vs column-oriented databases for the Vista Variables in the Via Lactea (VVV) Survey

Maria Patterson[1], Iraklis Klampanos[2], Ross Collins[3], Nicholas Cross[3], Mark Holliman[3], Robert Mann[3], and Malcolm Atkinson[2]

[1]Open Science Data Cloud NSF PIRE Fellow
[2]University of Edinburgh, School of Informatics
[3]University of Edinburgh, Royal Observatory, Edinburgh

We assess the relative performance of two database management systems (DBMS) for a large astronomical detection catalog from the VISTA Variables in the Via Lactea (VVV) Survey, a sub-survey of the Visible and Infra-Red Survey Telescope for Astronomy (VISTA). The VVV is a multi-epoch and multi-band survey aimed at detecting variable stars across the Galactic plane and bulge. The data collected have a detection table growing to over 1e11 rows and 132+ columns. For database performance testing, we use a subset of the data (1e7 rows) and compare the performance of MS SQL Server 2012, the currently implemented row-oriented DBMS, to MonetDB, a column-oriented DBMS. The two DBMSs are evaluated in three realistic tasks in terms of time efficiency and resource (disk I/O and CPU) use. We find that, in these three tasks, MonetDB significantly outperforms the existing MS SQL database.

1 Introduction

The success of large astronomical sky surveys such as the Sloan Digital Sky Survey (SDSS) has revolutionized data collection in observational astronomy. Small scale telescope observations performed by individual astronomers are becoming increasingly less common, in favor of dedicated surveys producing massive data sets. SDSS has produced 50 TB of data with an 18 TB object catalog of 357 million unique objects (Abazajian et al., 2009). The upcoming Large Synoptic Survey Telescope (LSST) will produce 60 PB with an object catalog of 20 billion rows (Ivezic et al., 2008). Accompanying the wealth of new data are the problems associated with handling very large data sets. Astronomers are becoming less limited by availability of data, but by the ability to quickly and easily handle large data sets and perform analysis.

This paper assesses the relative performance of two database management systems (DBMS) for data from the VISTA Variables in the Via Lactea (VVV) Survey (Saito et al., 2012) a public sub-survey of the Visible and Infra-Red Survey Telescope for Astronomy (VISTA) with the goal of detecting variable stars across the Galactic plane and bulge (see also Cross et al., 2012). The data collected are multi-epoch in five near-infrared filters, with a detection table expected to grow to over 10^{11} rows and 132+ columns. Using these detection data, we compare the performance of MS SQL Server 2012, the currently implemented row-oriented database management system, to MonetDB, a column-oriented DBMS. Currently, the VVV detection table is split both vertically and horizontally in order to more quickly process complete tasks such as whole-column recalibration updates, which are slow when achieved through row-by-row update statements. The goal of this research is to quantify the possible gain in time and resource use efficiency with the implementation of a column- oriented DBMS.

2 Experimental Setup

2.1 Data Sets

The largest table in the VVV database is the object detection catalog, which currently has 3×10^{10} rows of source detections and 132 columns of various measurements such as photometric or astrometric properties. For our analysis, we use a subsets of 1×10^{7} rows of this detection table to test relative DBMS performance. This small subset, which we will refer to as *vvvDetectionSmall*, is able to fit in memory. Further research with this experiment should also use larger subsets that are unable to fit in memory for a deeper analysis of scalability. We had planned to use various subsets of increasing size, but difficulties with virtual machine and disk failures and subsequent loss of data during the 6 week fellowship limited our time. We run the same SQL queries for this data set in both MS SQL Server 2012 and MonetDB with the same hardware configuration and assess the relative performance of the databases with the data set.

2.2 VMs and Hardware Configuration

We created two virtual machines through the KVM Virtualization platform, one running a Windows Server 2008 R2 operating system for MS SQL Server and one running Debian Linux jessie for MonetDB. Each VM has its own dedicated physical disk with 12 cores, 12 GB of memory, and a dedicated, physical 6 TB disk drive.

3 Methodology

3.1 System Monitoring

To acquire system and process performance statistics, we wrote and implemented a monitoring program that retrieves the relevant information via the Python module `psutil` (`http://code.google.com/p/psutil/`, version 1.0.1). Because `psutil` is cross-platform, it offers a consistent method for tracking process and system utilization information in both Windows and Linux. Throughout each test query at small time intervals, we retrieve the following CPU, I/O, and memory information, in addition to the total runtime of the task.

- system-wide CPU utilization in percentage
- system disk I/O statistics (number, size in bytes, and time in milliseconds of each read and write)
- disk I/O statistics (number and size in bytes) for every instance of the each database's server process
- number of threads used for each server process
- number of voluntary and involuntary context switches for each server process
- CPU user and system times for each server process
- memory for each server process (Resident Set Size (RSS) and Virtual Memory Size (VMS) in bytes)

The monitoring code is run through Python from the command line using BASH shell in Linux and Powershell in Windows and executes queries via `mclient` for MonetDB and `sqlcmd` for MS SQL Server.

3.2 Queries and Usage Patterns

We compare the relative performance of MS SQL Server and MonetDB with the small *vvvDetectionSmall* data table by monitoring three tasks described below. These tasks are currently used by the Vista Science Archive for the VVV Survey and are typical of tasks used for astronomical catalog data curation.

3.2.1 Bulk inserting data (*Task A*)

This task involves ingesting the data for an entire table of 132 columns and 10^7 rows for *vvvDetectionSmall* into each database. The data is read from a column-separated value file mounted in a shared location to each VM and loaded into the databases using the command BULK INSERT in MS SQL Server and the equivalent command COPY INTO in MonetDB. We reproduce the same schema used in the actual VVV Survey database with identical schemas for both test tables in the MS SQL Server and MonetDB, using a primary key of three attributes - multiframeID, extNum, and seqNum.

3.2.2 Inserting and updating seq. IDs (*Task B*)

This task involves two queries to add rows of new data to the existing object detection table, which might be done after a night's or month's worth of observations are completed. The current process for adding data to the VVV Survey detection table involves 1) inserting the new rows of data with the objID attribute as some negative number and then 2) updating these negative objIDs to provide sequential unique IDs for every detection.

3.2.3 Column updates for recalibration (*Task C*)

This task involves updating several columns for all detections in a single detector (i.e., updating several columns for all rows of a subset of the referenced primary key), which is necessary for the recalibration of photometric or astrometric measurements. We monitor a simulated photometric recalibration of 20 columns for two subsets of the detection tables equivalent to two types of detectors. The first is a pawprint ($\sim 10^4$ rows),

and the second is a tile ($\sim 10^6$ rows), determined by an appropriate WHERE clause selecting on a combination of multiframeID and extNum. These pawprint and tile detection subsets are chosen to be completely contained within the *vvvDetectionSmall* table, which is itself a subset of the *vvvDetection* table, ensuring that the same number of rows are updated in each table.

4 Results and Ongoing Work

In Table 1 below, we present a run-time summary of the two systems MS SQL Server vs MonetDB in each of the three tasks described above. MonetDB significantly outperforms MS SQL Server in terms of speed for each query tested. The tasks run in MonetDB are anywhere from a factor of 2-200 times faster than the same tasks run in MS SQL Server. Column updates in particular are much faster with MonetDB than MS SQL Server, which is expected given its column-oriented nature.

Figures 1-5 show CPU and I/O activity during the execution of the queries involved in each Task in MS SQL Server (blue) and MonetDB (red). The run time for each query is given along the x-axis. Given our results from the three tasks tested, MonetDB appears to be a faster and more efficient database management system for the VVV detection table data.

An ongoing analysis of the two DBMSs for use with the VVV survey can continue this work in a number of ways. First, an extended analysis with various larger data sets would be useful to test the scalability of each DBMS for databases too large to fit in memory, like the full VVV detection table. Additionally, this analysis was limited to testing of queries with one table. It would also be useful to compare each DBMS's performance across multiple tables using JOIN .

References

Abazajian, K. N., et al. 2009, ApJS, 182, 543

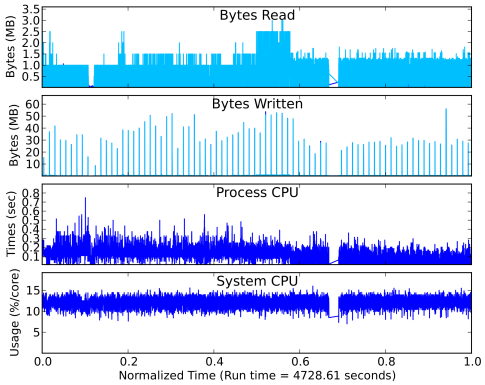
Cross, N. J. G., et al. 2012, A&A, 548, A119

Ivezic, Z., et al. 2008, ArXiv e-prints

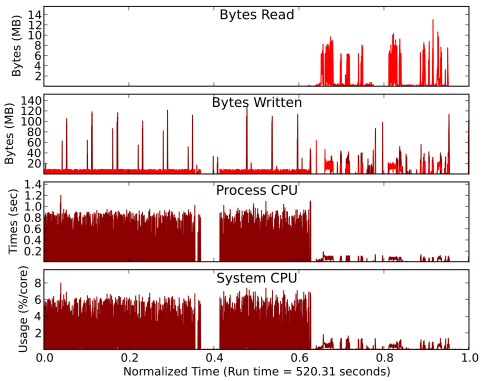
Saito, R. K., et al. 2012, A&A, 537, A107

	vvvDetectionSmall (10^7 *total rows*)				
	Task A	**Task B** (5.5×10^6 *affected rows*)		**Task C** (10^4 | 10^6 *affected rows*)	
	N/A	*Part 1*	*Part 2*	*Part 1*	*Part 2*
MS SQL	4729	786	75.2	22.7	57.7
MonetDB	520	416	2.6	0.1	7.0

Table 1: Run-time summary (seconds)

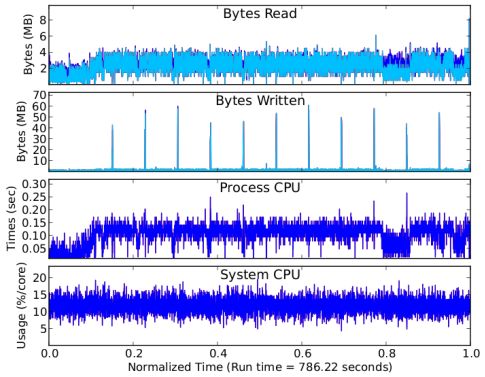


(a) I/O and CPU statistics for bulk loading all data into the MS SQL *vvvDetectionSmall* table.

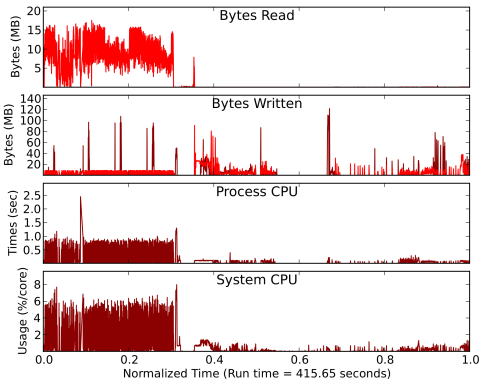


(b) I/O and CPU statistics for bulk loading all data into the MonetDB *vvvDetectionSmall* table.

Figure 1: CPU and I/O results for Task A

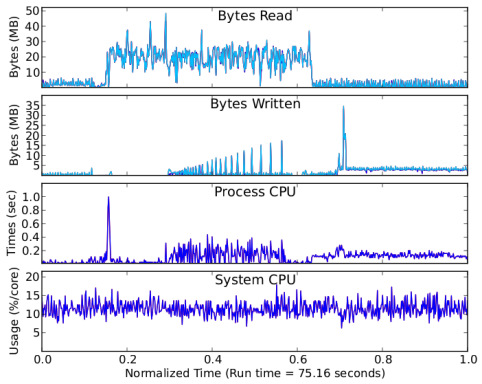


(a) I/O and CPU statistics for inserting new rows into the MS SQL *vvvDetectionSmall* table.

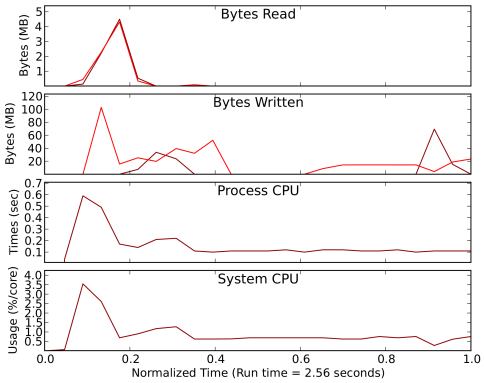


(b) I/O and CPU statistics for inserting new rows into the MonetDB *vvvDetectionSmall* table.

Figure 2: CPU and I/O results for Task B Part 1

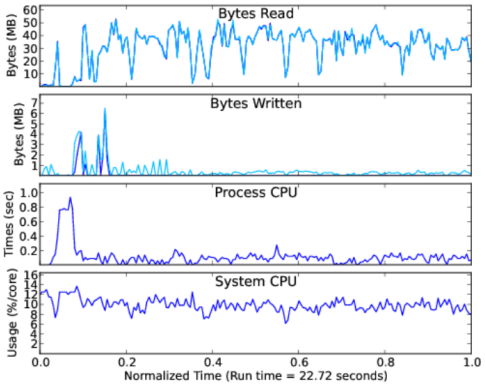


(a) I/O and CPU statistics for updating sequential IDs for new rows added to the MS SQL *vvvDetectionSmall* table.

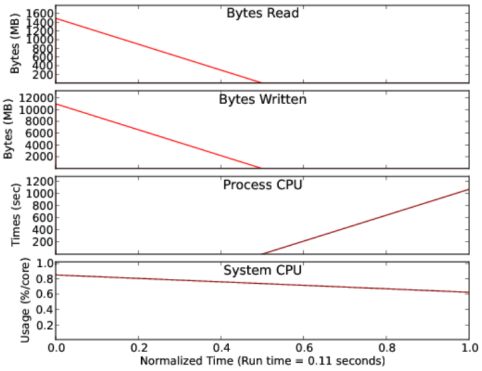


(b) I/O and CPU statistics for updating sequential IDs for new rows added to the MonetDB *vvvDetectionSmall* table.

Figure 3: CPU and I/O results for Task B Part 2

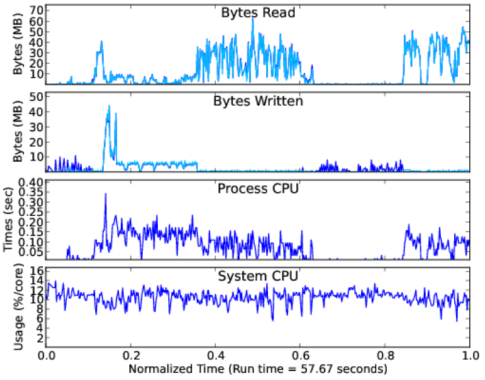


(a) I/O and CPU statistics for updating 20 columns for recalibration of one 'pawprint' (22,607 rows) in the MS SQL *vvvDetectionSmall* table.

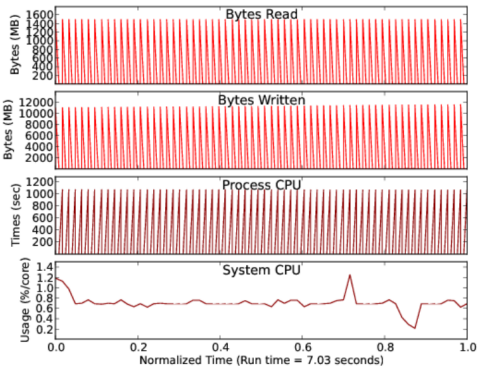


(b) I/O and CPU statistics for updating 20 columns for recalibration of one 'pawprint' (22,607 rows) in the MonetDB *vvvDetectionSmall* table.

Figure 4: CPU and I/O results for Task C Part 1



(a) I/O and CPU statistics for updating 20 columns for recalibration of one 'tile' (1,042,194 rows) in the MS SQL *vvvDetectionSmall* table.



(b) I/O and CPU statistics for updating 20 columns for recalibration of one 'tile' (1,042,194 rows) in the MonetDB *vvvDetectionSmall* table.

Figure 5: CPU and I/O results for Task C Part 2