ADASS XIII
ASP Conference Series, Vol. XXX, 2004
F. Ochsenbein and M. Allen eds.

Batch Query System with Interactive Local Storage for SDSS and the VO

William O'Mullane[1], Jim Gray[2], Nolan Li[1], Tamás Budavári[1], María A. Nieto-Santisteban[1], Alex Szalay[1]

Abstract. The Sloan Digital Sky Survey science database is approaching 1TB in size. While the vast majority of queries normally execute in seconds or minutes, this prompt execution time can be disproportionately increased by a small fraction of queries that take hours or days to run either because they require non-index scans of the largest tables or because they request very large result sets. In response to this, a job submission and tracking system has been developed with multiple queues. The transfer of very large result sets from queries over the network is another serious problem. Statistics suggested that much of this data transfer is unnecessary; users would prefer to store results locally in order to allow further cross matching and filtering. To allow local analysis, a system was developed that gives users their own personal database(MYDB) at the portal site. Users may transfer data to their MYDB, then perform further analysis before extracting it to their own machine.

1. Sloan Digital Sky Survey - SkyServer

The SkyServer[3] has been public since June 2001. This is coded in ASP on a Microsoft.Net server and backed by a SQL Server database. The current database, Data Release 1 (DR1), is over 1TB (with indexes) and subsequent releases will bring this to at least 6TB of catalog data in SQL Server. In fact there will be up to 50 TB of pixel and catalog data and more of this may be put in the database e.g. the points of all SDSS[4] spectra have recently been loaded in a separate database. Hence the database could become much larger than 6TB. The SkyServer site allows interactive queries in SQL[5]. The results of some of these queries are large; we have seen as many as 12 million rows downloaded in an hour the site is averaging 2M hits per month. Considering this is running on a $10k server the site performs extremely well. However in certain circumstances

[1]The Johns Hopkins University

[2]Microsoft Research

[3]http://skyserver.sdss.org

[4]Sloan Digital Sky Survey

[5]Structured Query Language

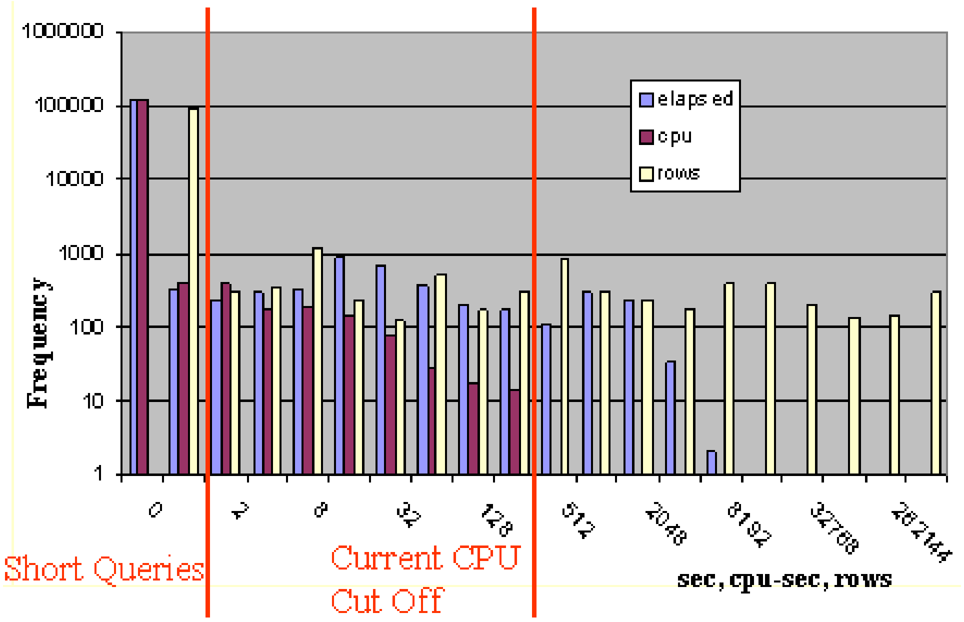

Figure 1. Log-log plot of Frequency of queries using n seconds, n cpu seconds, or returning n rows over 100 hour period

we have experienced problems. Complex queries can swamp the system and erroneous queries may run for a long time but never complete.

1.1. SkyServer Statistics

Generally, the execution time of result set sizes follow a natural power law 1. Hence there is not particularly obvious point at which queries could be cut off. All queries currently run at the same priority- there is no ranking or "nice" system built into SQL Server (or any other DB products). While this may not be a problem in itself, long queries can slow down the system, causing what should be quick queries to take much longer. Some long queries are returning data to a user over an Internet connection; frequently time is spent transferring the data, not extracting it, but a database resource is still tied up. We have seen as many as twelve million rows(20GB) downloaded in one hour. These large transfers are unnecessary; this data is often used only to make comparisons against a small local set.

2. Batch System

We have developed a system[6] to address these problems, particularly to partition queries and allow rudimentary load balancing across multiple machines, guarantee query completion/abortion, provide local storage for users, and separate

[6] http://skyservice.pha.jhu.edu/devel/casjobs

data extraction/download from querying. This will be pertinent for the virtual observatory as the SkyNode protocol begins to mature (Yasuda et al. 2003, Budavári et al. 2003)

2.1. Queues

We have set up multiple queues based on query length. Jobs in the shortest queue are executed instantly, while jobs in all other queues are executed sequentially (limited simultaneous execution is allowed). Query execution time is strictly limited by the time assigned to a particular queue, and queues are executed on separate machines mirroring the same data. Hence there will no longer be ghost jobs hanging around for days, nor can a long query hinder execution of a shorter one; a job may take only as long as the limit of its queue, and different types of jobs are executed on different machines.

2.2. Query Estimation

There is a query estimator in SQL server however its accuracy is questionable and for our first beta of this system we have decided not to actively determine query execution time. This responsibility is left to the user; they decide how long they think their query will take and choose the appropriate queue accordingly. As mentioned previously queries exceeding queue time limit will be canceled. We do provide autocomplete functionality that will move the query to the next queue if it times out in its original queue. In a future release we may use the statistics gathered on jobs to develop a heuristic for estimating query lengths and automatically assigning them to queues.

2.3. MYDB - Local Storage

Queries submitted to the longer queues must write results to local storage, known as MYDB, using the standard into syntax e.g.

```
select top 10 * into MYDB.rgal from galaxy where r < 22 and r >21
```

The MYDB idea is similar to the AstroGrid MySpace (Walton et al. 2003) notion. We create a SQL Server database for the user dynamically the first time MYDB is used in a query. Upon creation, appropriate database links and grants are made such that the database will work in queries on any database. Since this is a normal database the user may perform joins and queries on tables in MYDB as with tables in any other database. The user is responsible for this space and may drop tables from it to keep it clear. We have initially assigned each user 100MB but this is configurable on a system and per user basis.

2.4. Groups

Some users may wish to share data in their MYDBs. Any user with appropriate privileges may create a group and invite other users to the group. Any invited user will have to accept to be part of the group. A user may then publish any of his MYDB tables to the groups of which he is a member. Other group members may access these tables by using a pseudo database name consisting of the word *group* followed by the *id* of the other user followed by the table name e.g. if the *adass* user published the table *rgal* and you were in a group with *adass* you may access this table using *GROUP.adass.rgal*.

2.5. Import/Export Tables

Tables from MYDB may be requested in FITS, CSV, or VOTable[7] format. Extraction requests are queued as a different job type and have their own processor running. The file extraction is done on the server and a URL to the file location is put in the job record upon completion. On the Web site group tables also appear in the extraction list. Currently a user may also upload a CSV file of data to an existing table in MYDB. Having the table created before upload avoids problems of intended types. We hope the ability to upload data and the group system will reduce some of the huge downloads from our server.

2.6. Jobs

Apart from the short jobs, everything in this system is asynchronous and requires job tracking. This is simply done by creating and updating a row in a Jobs table in the administrative database. However this also requires users to be identified and associated with the jobs. Identification is also required for resolution of MYDB. A user may list all previous jobs and get the details of status, time submitted, started, finished etc. The user may also resubmit a job.

Ferris Wheel A future experiment we would like to try is to batch full table scan queries together. Theoretically we may piggy back queries in SQL Server so that a single sequential scan is made of the data instead of several. Ideally we would like to not have to wait for a set of queries to finish scanning to join the batch. Rather we would like some set of predefined entry points where a new query could be added to the scan. Conceptually one may think of this as a ferris wheel where no matter which bucket you enter you will be given one entire revolution.

3. SOAP Services

We have found that SOAP services provide a very clean API for any system. In this system the Web site sits upon a set of SOAP services. Any user may access these services directly using a SOAP toolkit in their preferred programming language. At JHU we have tried Python and Java (AXIS) clients for webservices successfully. Others have written Perl clients. More information on this is available at the IVOA Web site[8].

References

Budavári, T., et al. 2003, this volume, [P2.18]

Yasuda, N., et al. 2003, this volume, [P3.10]

Walton, A., et al. 2003, this volume, [O7.5]

---

[7] http://www.us-vo.org/VOTable/

[8] http://www.ivoa.net/twiki/bin/view/IVOA/WebgridTutorial