Cross-Matching Very Large Datasets

María A. Nieto-Santisteban, Aniruddha R. Thakar, and Alexander S. Szalay

Johns Hopkins University

***Abstract*—The primary mission of the National Virtual Observatory (NVO) is to make distributed digital archives accessible and interoperable in such a way that astronomers can maximize their potential for scientific discovery by cross-matching multi-wavelength data between multiple catalogs on-the-fly. While cross-matches between small datasets are possible at present, large-scale cross-matches that involve all or a large fraction of the sky cannot be performed on demand. This is a serious deficiency that prevents the NVO from realizing its mission and hinders its widespread acceptance by the community. In this paper, we analyze the issues and requirements that an environment aiming to enable large-scale astronomical science needs to consider, and describe a workbench environment where astronomers can cross-correlate, analyze and compare vast amounts of data and share their results with others. We focus on catalog data managed by commercial Relational Database Management Systems and analysis tasks expressed in SQL. We describe an indexing algorithm named *Zones* and its crucial role in enabling parallelization and cross-matching of very large datasets.**

***Index Terms*— Cross-Match, Parallelization, Very Large Databases.**

I. Introduction

The astronomical cross-match problem involves identifying and comparing light sources belonging to different observations of the same sky region. Such observations may come from telescopes with different wavelength and detection limit capabilities or may correspond to the same telescope but have been taken at different times and under different conditions. Some times one is interested in finding objects surrounding a given object or position. In other cases, the goal is to perform one-to-one matches in order to combine physical properties or study the temporal evolution of the source. When the only attributes under consideration are positional we speak of a spatial cross-match. This spatial identification is not easy. Due to different errors, instrument sensitivities and other peculiarities of the data acquisition and calibration process, which are unique to each archive and survey project, the same object may have slightly different coordinates on different catalogs. Such imprecision makes astronomical cross-matching especially hard. We are dealing with a kind of cross-match that requires performing a fuzzy spatial join across multiple datasets. Searches and cross-correlations usually involve millions or billions of objects making spatial indexing a fundamental component in any environment aiming to provide serious data exploration and scientific analysis.

This work is supported by the NASA Applied Information Systems Research Program (NASA AISR 2006-09) and Microsoft Research.

II. The Astronomical Problem

Astronomy, like many other sciences, is facing a data avalanche. Current and future instruments are producing data at rates that will break the Petabyte barrier in the next decade. Astronomical datasets are spread all over the world. Formats, naming schemas, data structures, etc. are very diverse. Often times, the diversity is unavoidable since projects are usually funded based on their unique characteristics. However, motivated by the wish to share, initiatives such as the National Virtual Observatory (NVO)[5] and its partners in the International Virtual Observatory Alliance (IVOA)[4] are establishing protocols to help communication, minimize diversity, and in this way, maximize the number of new scientific discoveries. These characteristics - high distribution, diversity, and very large datasets - make the cross-match and data analysis to be a big challenge.

III. Open SkyQuery

Cross-matching requires all the data to be in the same place at some given point in time. In order to solve the distribution problem and provide universal access to the data, we created OpenSkyQuery.net [10], a web portal that allows querying and cross-matching distributed astronomical datasets. Some of these datasets contain a billion or more entries, and for each entry, tens or even hundreds of columns may be stored. Services implement a standard interface defined by the IVOA community to facilitate communication between services and between services and the portal. *Open SkyQuery* knows about these data services by querying another web application, the NVO Registry. Users may upload small personal catalogs to the portal for cross-matching with the big public datasets. Users pose their queries in ADQL, the Astronomical Data Query Language. ADQL looks very much like SQL, the standard Structured Query Language used by many database systems. ADQL does not support all features of SQL, but it adds several important language constructs specialized for astronomy. These include a REGION function that allows users to constrain their queries to a particular region on the sky, and a cross-match function, XMATCH, that allows finding objects in different datasets that are spatially coincident to within some tolerance.

We have addressed two of the main issues with the development of *Open SkyQuery*. First, we solve the high distribution problem by creating an environment that allows discovery, integration, and universal data access. Second, the data diversity impact is minimized because services follow a standard interface protocol which hides the publisher's internal data structures.

Unfortunately, dealing with very large datasets requires a much higher effort. Providing access to very large datasets brings issues such as asynchronism and staging of results. When querying very big datasets, results are likely to be very big as well. Retrieving these results to a browser takes usually more time than users are willing to wait in front of their computers. Not to mention the lack of usability of displaying thousands or even millions of rows on a screen. Moving big volumes of data between services is expensive and slow usually.

We also face an optimization challenge. *Open SkyQuery* follows a very basic optimization process which consists of counting the number of objects within a given region and ordering the cross-match workflow from the service hosting the smallest number of objects to the service with the biggest number of objects. This fairly simple mechanism works reasonably well for most use cases. However as an example, when the footprints of the datasets are substantially different, this approach might not be efficient. When the datasets are small, it does not matter usually, but when we are trying to cross-match big datasets covering big areas of the sky, having knowledge of footprints and density distributions can help to improve the performance of the system significantly.

Finally, although *Open SkyQuery* is conceived primarily as a service to cross-match distributed astronomical datasets, it aims to provide individual access to the datasets as well. ADQL queries can be formulated through the portal to individual data services. As described above, ADQL is a restricted subset of SQL to accommodate to a number of different DBMS. Although ADQL serves the purpose it was intended for, we lose a lot of the native SQL expressivity and programmability at the same time.

IV. CasJobs

The Catalog Archive Server Jobs System (CasJobs) [1] is a web application created to solve some of the problems that the Sloan Digital Sky Survey (SDSS) Catalog Archive Server (CAS) portal, *SkyServer* [11], was facing due to the increasing size of the SDSS catalog databases along with a high demand to access the data [9].

CasJobs addresses the problems by creating a workbench environment where users can run almost unlimited queries against the SDSS databases in batch mode. Query results can be stored on the server-side in the user's personal relational database (MyDB) where users can perform further analysis and refinements using the fast filtering and searching tools that the DBMS offers. Users may upload and download data to and from their MyDB. They can analyze or cross-correlate these data inside MyDB or against the main SDSS databases. CasJobs grants full power to create new tables, indexes, functions, and stored procedures on the user's MyDB. CasJobs also offers a tool that let user look at execution query plans. All these capabilities provide a very rich programmability environment to implement complex scientific workflows [6]. Additionally, CasJobs creates a collaborative environment where users can form groups and share their data with others. Finally, one of the main benefits of CasJobs is that provides a framework where users can bring their SQL code to the DBMS server side, which is quite a unique feature. CasJobs is by no means exclusive to SDSS and has been successfully installed to work with other astronomy projects such as the Galex Evolution Explorer (GALEX), Palomar-Quest, and outside the astronomy context, Ameriflux, a project which collects levels of carbon, water, energy and nitrogen from micrometeorological tower sites in the Americas to do environmental and climatic change studies.

With the development of CasJobs, we have addressed some of the remaining issues related to the management of very large datasets. Users have asynchronous access and can stage their data at their own databases systems on the server side. Data transfers between client and server are significantly reduced and happen only when strictly necessary. Users have the full expressivity of SQL to formulate queries and create complex workflows against their own databases and/or the SDSS databases. Using a CasJobs system holding the three largest catalogs such as the United States Naval Observatory B (USNOB), Two Micron All Sky Survey (2MASS), SDSS, etc., we are (almost) ready to set up an environment where we can cross-match very large datasets. The remaining issue we need to address now is the inherent computational complexity that involves cross-matching big datasets spatially.

V. Zones

Zones is an indexing algorithm to efficiently support spatial queries on the sphere using a generic RDBMS.

Each position on the sphere can be defined by two angles (α, δ). These two angles are known as *Longitude* and *Latitude* when referring to Earth's coordinates or *Right Accession (R.A.)* and *Declination (DEC)* when working in the Astronomy context. The *Zones* algorithm is equally valid to work with points on either the terrestrial or celestial sphere.

The basic idea is to map the sphere into stripes of a certain height called zones. Each object at position (α, δ) is assigned to a zone by using the fairly simple formula

$$zoneID = \lfloor \delta + 90.0 / h \rfloor, \tag{1}$$

where h is the zone height and δ is the Declination, both in degrees. Objects within a zone are then stored on disk ordered by zoneID and Right Ascension using a traditional B-tree index. The storage constraint optimizes the data access as it minimizes the number of I/O operations.

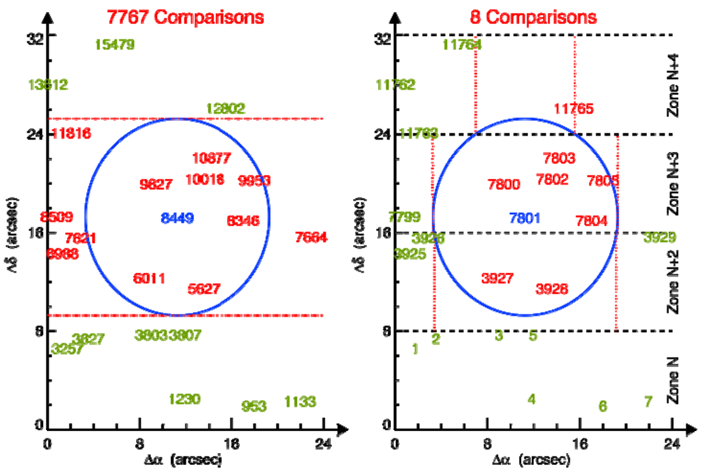

Fig. 1. Declination vs ZoneID and Right Ascension. The numbers in the figures are indexes indicating their order on the disk. Fewer comparisons are required when data is ordered by ZoneID and Right Ascension.

Spatial searches, and most specifically circular searches, can be efficiently performed by using dynamically computed bounding boxes (B-tree ranges), followed by a careful (expensive) distance test that examines all members within the bounding box and discards false positives. Fig. 1. shows the number of comparisons required to perform a 4 arcsec search around some arbitrary object using simulated data ordered by declination, versus data zoned in 8 arcsec stripes and ordered by ZoneID and Right Ascension. The details of how to implement the Zones algorithm are discussed in [2].

Assuming we have a function implementing a circular search, the simplest approach to cross-match two datasets, A and B, would be to execute a series of circular searches in dataset B centered on object from dataset A. However, when the cardinality of A is very big, order of a few millions of rows, calling a search function sequentially turns out to be extremely slow. Luckily, looking at Fig. 1., one can easily see that if the goal is cross-matching, we can load two zones and perform the whole comparison in a single step instead of executing individual little circular searches,. As a matter of fact, using this batch-oriented technique we managed recently to run a 1-degree cross-match between SDSS (350 million objects) and a sparse grid (50, 000 objects) in 7 hours instead of the few years that the sequential execution was predicting.

Finally, Zones also provides a simple way to partition the data, and by extension making easy the parallelization of the cross-match. Actually, we can apply partitioning and parallelization at two different levels; at the CPU level by using a multi-core processor, as well as, distributing the data or workload to different servers where the jobs run in parallel.

At the CPU level and assuming the cross-match distance is smaller than the zone height, we can divide the work in cross-match tasks such as: a) A.zoneID - 1 = B.zoneID, b) A.zoneID = B.zoneID, and c) A.zoneID + 1 = B.zoneID. Since there are no dependencies, we can run in parallel all pairs {(0, 1), (1, 2), …} in different CPUs, followed by pairs {(0, 0), (1, 1), …} and {(1, 0), (2, 1), …} In principle the order is irrelevant, but the data is likely to be in the cache if it happens in this order and therefore the I/O activity will be reduced. At the moment, we have not experimented with the multi-core parallelization approach very much. Our partitioning and parallelization research and experiments have focused mainly in distributing the data and workload across multiple servers. However, the multi-core analysis is in the short-term plan.

In the following section we describe one of our experiments cross-matching very big datasets in a cluster of DBMS servers.

VI. Partitioning and Parallelization Tests

We used a vertical partition (subset of attributes) of the SDSS Data Release 3, USNOB, and 2MASS catalogs to run large-scale access and cross-match queries. SDSS DR3 contains about 142 million objects covering a non-contiguous area of 5,282 squared degrees. In order to simplify our cross-match experiments, we used most of the 2MASS (28,445,694 objects) and USNO (117,698,363 objects) area that overlaps with SDSS DR3 [Fig. 2.]. The next task was to distribute the workload in a homogeneous way. This step implied zoning the data and assigning zone subsets to different servers in the cluster [Fig. 3.]. At this point, the databases were replicated across a cluster of servers.

The most efficient approach to query a single catalog is to distribute the workload homogenously so all nodes process the

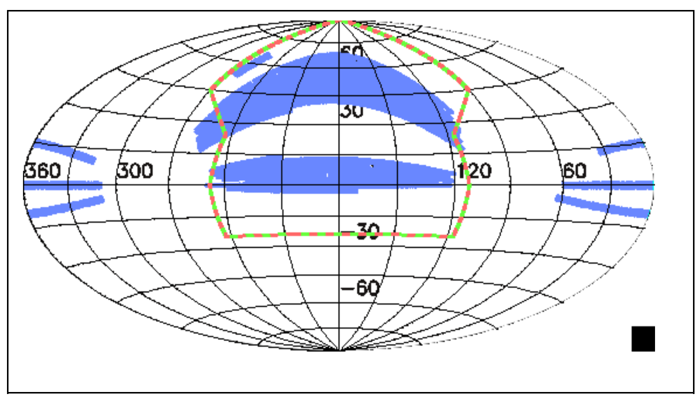

Fig. 2. SDSS DR3, 2MASS and USNOB test-case footprints.

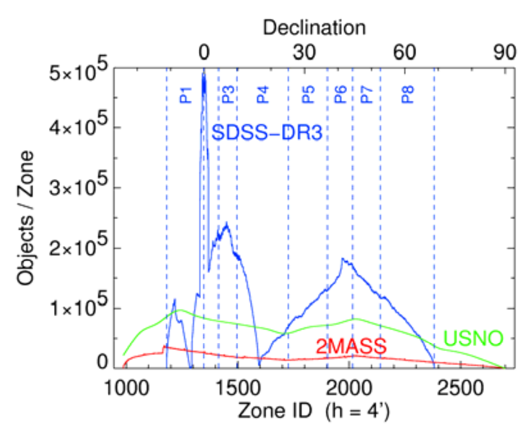

Fig. 3. Zone distribution of SDSS DR3 data.

the same amount of data. But cross-match queries require choosing a catalog leading the partitioning. What is the best partitioning choice is not always as simple as it might seem. For example, a reasonable choice could be to make the catalog with the smallest number of objects lead the partitioning. The assumption is that minimizing the size of the dataset works best because we reduce the number of operations on each server. This would be 2MASS in our test case. Looking at Fig. 3., we can see that making 2MASS the leading partitioning catalog to do a cross-match with SDSS would be a terrible choice. While certain servers would be overloaded, others would basically remain idle. How to automatically decide what is the best distribution approach is not easy, if not impossible, without having precise information about the catalog densities and coverage footprints.

Performance Analysis

We run the same cross-match procedures between SDSS DR3 and 2MASS using 1, 2, 4, and 8 servers to measure scalability. Fig. 4. shows that scalability is certainly possible in terms of CPU time. This is directly related to a good workload distribution. On the top right, the initial high slope for the one and two-server cases is due to I/O as the bottom graph shows. However, the basically flat line plotting total elapsed time indicates that by using four or more servers we can speed up the cross-match computation linearly. It is worth noticing the small difference between *Total CPU* and *Total Elapse* indicating good usage of the CPU.

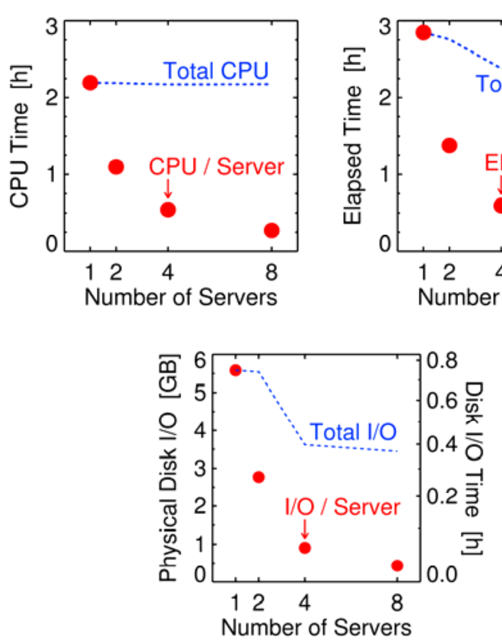

Fig. 4. SDSS vs 2MASS Cross-Match Performance. These tests were performed under the following hardware and software configuration: Super Micro Rack mounted Cluster (10 nodes), Processor: Xeon 2.7 GHz (x4), 2 GB Memory, Disks: 3ware Escalade 7000 ATA RAID, Microsoft Windows Server 2003 Enterprise, Microsoft SQL Server 2000.

VII. Conclusion

In this paper, we analyzed the issues and requirements that an environment aiming to enable large-scale astronomical science needs to consider and described a workbench environment, CasJobs, where astronomers can cross-correlate, analyze and compare vast amounts of data and share their results with others. Using this type of workbenches we can bring the multi-billion cross-match problem down to the few hours range. Such a framework allows astronomers to implement sophisticated analysis workflows using the full expressivity of SQL. DBMS native optimization procedures can be applied to improve the performance of individual and distributed queries. The price to pay though is that data need to be centralized and managed under a homogenous DBMS framework. Although, this centralization might not seem a very attractive idea, the alternative is to continuously move the data on-demand. A framework aiming to provide an environment where to cross-correlate, analyze and compare vast amounts of data will need mechanisms such as UDT (UDP-based Data Transfer), a high performance network transfer protocol, to move the data between archives quickly. There are always limits to how much data can be hosted in a single site or by a single organization. Nonetheless, it will be always important to analyze the cost of moving the same data many times versus the cost of hosting the data permanently.

We have also described Zones, an algorithm capable of cross-matching very large datasets efficiently and facilitating parallelization. The tests and results presented here demonstrate that by zoning, partitioning, and parallelizing the workload and using RDBMS technologies, we can reduce linearly the response time. Efficient cross-matching is becoming critical to projects such as the Large Synoptic Survey Telescope (LSST) where millions of objects need to be cross-matched in few seconds.

Acknowledgment

The authors would like to acknowledge Jim Gray (Microsoft Research) as a co-author in this paper. Sadly, we could not find Jim [12] to review it and give his blessing.

References

[1] CasJobs http://casjobs.sdss.org/CasJobs/
[2] J. Gray, M. A. Nieto-Santisteban, and A. Szalay, “The Zones Algorithm for Finding Points-Near-a-Point or Cross-Matchin Spatial Datasetes,” *Microsoft Technical Report:* MSR-TR-2006-52, April 2006.
[3] R. L. Grossman, M. Mazzucco, H. Sivakumar, Y. Pan, and Q. Zhang, “SABUL, Simple Available Bandwidth Utilization Library for high-speed wide area networks.” *Journal of Supercomputing*, to appear.
[4] International Virtual Observatory Alliance, http://ivoa.net
[5] National Virtual Observatory, http://us-vo.org
[6] M. A. Nieto-Santisteban, J. Gray, A. Szalay, J. Annis, A. R. Thakar, and W. O’Mullane, “When Database Systems Meet the Grid,” *in Proc. ACM CIDR 2005,* Asilomar, CA, January 2005.
[7] M. A. Nieto-Santisteban, A. Szalay, A. R. Thakar, and J. Gray, “LSST, the Spatial Cross-Match Challenge,” *Astronomical Data Analysis Software and Systems XVI O1.2 ASP Conference Series*, Vol. XXX, 2007, in press.
[8] R. A. Shaw, F. Hill and D. J. Bell, eds.

[9] W. O'Mullane, N. Li, M. A. Nieto-Santisteban, A. Thakar, A. Szalay, and J. Gray, "Batch is back: CasJobs, serving multi-TB data on the Web," *in Proc. 2005 IEEE International Conference on Web Services (ICWS 2005).* Orlando, FL, July 2005.

[10] Open SkyQuery, http://openskyquery.net

[11] SkyServer, the SDSS Catalog Archive Server, http://skyserver.sdss.org

[12] Tenacious Search, http://www.openphi.net/tenacious/