Case Study of Handling Scientific Queries on Very Large Datasets: The SDSS Science Archive

Aniruddha R. Thakar[1], Peter Z. Kunszt[1], and Alexander S.Szalay[1]

The Johns Hopkins University, Baltimore MD 21218, USA

Abstract. The SDSS Science Archive (SX) was designed to enable scientific data mining and interactive data exploration on the terabyte scale. It consists of a distributed object-oriented database that is accessible via a client-server interface. The lightweight SX GUI client can be run on any platform. SDSS queries are formulated in SXQL, an SQL-like query language with some object-oriented and astronomy extensions . The SX server combines a fully multithreaded query engine with a distributed parallel architecture, splitting the data among multiple hosts and allowing for parallel, scalable I/O and parallel data analysis. Each query is parsed into a query execution tree which is executed in parallel. Data on remote partitions are accessed in parallel locally by remote slave servers. This distributed and multithreaded design allows query execution to be optimized and dynamically load-balanced for any type of multi-processor architecture, from SMP machines to Beowulf-type clusters.

1 Introduction - Mining Large Datasets

The advent of multi-terabyte datasets in astronomy is creating a paradigm shift in the way that astronomers conceptualize, store and access data. The traditional tools used to access the data in current archives can no longer scale to the mind-boggling sizes of the archives expected to come online within the next decade or so. The Sloan Digital Sky Survey's Science Archive [1] is among the first of this new wave of multi-terabyte astronomical archives. Maintaining and accessing data in archives that are several orders of magnitude larger than current archives presents several challenges [2], some of which are listed below.

1.1 Data Mining Tools

Data mining (finding "needle in haystack") in these large archives requires specialized tools - the traditional tools currently used by astronomers just do not scale to these regimes. Searching through directories of FITS files is no longer a viable option! Furthermore, data integrity is crucial, given the size and complex nature of the data. Multiple versions (reruns) of data may need to be stored. Concurrent access by many users must be possible, including the ability to load new data while the archive is up and running.

1.2 A Virtual Observatory

The archives of the future will provide a "virtual sky" at your fingertips, and linking these archives together to enable unprecedented scientific discovery by combining data from more than one archive will give rise to a *Virtual Observatory*. This ambitious goal can only be realized if each of these archives is built with some basic capabilities and architectural elements that will allow it to be integrated into an eventual virtual observatory. Data abstraction is necessary so that metadata can be queried and exchanged. Standardized interfaces and tools must be used to make the archive interoperable with other archives. Efficient data mining must be facilitated for large user communities in which distributed and replicated datasets will be the norm. Fast and portable network access (e.g., from laptops) to the data is a must - web-based access will not be fast enough for power users of a virtual observatory. Also, each archive must be able to deploy on any kind of hardware.

2 The SDSS Science Archive

The SDSS Science Archive (abbreviated SX) is the final repository of the *processed* and *calibrated* data for the SDSS. All the *raw* data is stored in a separate archive - the Operational Database (OpDB). Both the SX and the OpDB employ a commercial Object-Oriented DataBase Management SSystem (DBMS) - **Objectivity** - as their data repository. Using a commercial object-oriented DBMS provides the following advantages:

- Persistence and data integrity - storing the data securely and allowing multiple users to access it concurrently, and updating the data without corrupting it.
- Binary compatibility across diverse hardware - allowing the data to be stored in a distributed environment on different hardware platforms.
- Transparent and configurable data organization - allowing optimization of the data storage hierarchy for maximum performance.
- Superior performance for complex data models - this is something that a relational DBMS would have difficulty in providing.
- Distributed database - the ability to spread the data across a network of data hosts (partitions).
- Automated replication - this is especially important for the SX, since **the Master Archive will be replicated at several sites**, including each of the SDSS institutions (mirror sites).

Fig. 1(a) shows the data flow for the SDSS data. The raw data from the OpDB will be processed in a pipeline and stored in the master archive, which will be replicated at several sites (local archives). Since Objectivity provides neither SQL-compatibility nor an object query language, and provides only very basic querying primitives, we have had to develop our own

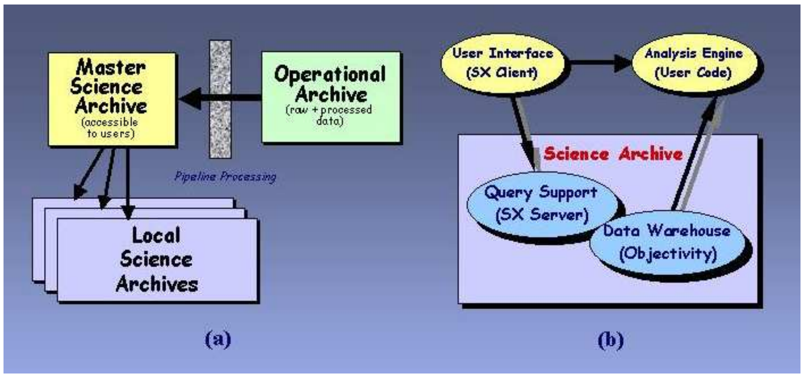

Fig. 1. (a) SDSS data flow, and (b) Client/Server view of the SDSS Science Archive.

query language as well as our own user interface and query agent. SX queries are formulated in **SXQL**, an SQL-like query language with object-oriented extensions and mathematical/astronomical macros. The SXQL query is first analyzed to yield an *a priori* cost-estimate. This allows the user to decide whether to continue with the query or not, and minimizes processing of very long queries.

2.1 The SX Client/Server Model

Fig. 1(b) shows the overview of the client/server architecture of the SX. The client is a lightweight, portable Tcl/Tk GUI that can be run on any hardware, including laptops. The client features a multi-windowing environment that enables users to submit SXQL queries to the SX query agent. It supports multiple concurrent user sessions and multiple concurrent queries within each session. It also monitors the progress of queries and allows queries to be aborted, suspended and resumed.

From the very outset, the SX software was designed with a high degree of parallelism inherent in it. The server, or SX query agent, is fully multi-threaded and distributed. It creates and manages multiple concurrent user sessions and executes user queries (see below for details). Query output can be directed back to the client (default) or remote targets specified by the user.

2.2 SX Architecture

Fig. 2 illustrates the SX software architecture. The SX query agent consists of several components, including the session manager or *Server*, the *Remote Slave Servers* to run distributed queries, the query *Engine* that executes each query in multi-threaded mode, and the *Parser* that parses the SXQL query. The SX was designed to meet the following data mining challenges:

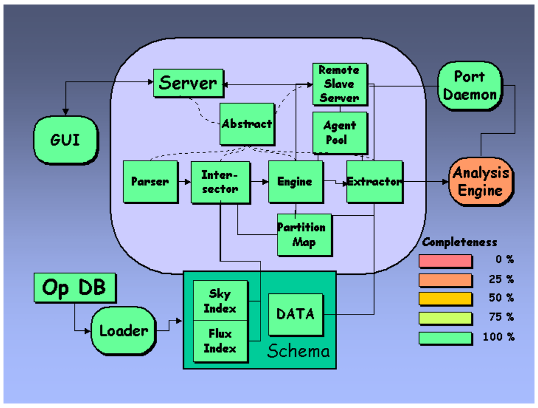

Fig. 2. The SDSS Science Archive software architecture.

- **Multi-Terabyte Dataset** - The total number of objects expected to be stored in the SX by the time the SDSS is done taking data is in excess of 200 million. The sheer size of the SX dwarfs all previous astronomical archives, and searching through this enormous dataset presents a daunting problem. Furthermore, although the SX data model contains only a handful of different object types, the hierarchical class structure and the interrelationships or *association links* between objects makes for quite a complex data structure. The large dataset also means that output volumes can be extremely large, and require binary output format with compression.
- **Distributed Dataset** - The SX will be a "federated" database, i.e., the data will be distributed across a cluster. To make the most of the distributed environment, each partition will be searched locally in parallel.
- **Portable Performance** - The primary goal is a parallel, scalable, I/O-bound archival system that is optimized for performance on both SMP (Symmetric Multi-Processing) and Beowulf-type cluster hardware environments.

2.3 The SX Data Model - Tag Classes

We have defined lightweight *tag* or *descriptor* object classes that encapsulate the most frequently requested fields. These fields are indexed using a k-d tree-based indexing scheme - the Hierarchical Triangular Mesh or HTM [3]. Furthermore, we use the inheritance mechanism available in an object-oriented database to define a hierarchy of subclasses of the main tag class so that searches can be specialized and narrowed considerably (see Fig. 3). The small size of the tag objects also means that caching is very efficient for these objects. The combination of these three features - the HTM, the tag subclasses, and the caching - yields superfast performance for the vast majority of searches, and objects of interest can first be selected using the tag classes and the rest of the object data can be retrieved through the links contained in the tag objects if desired.

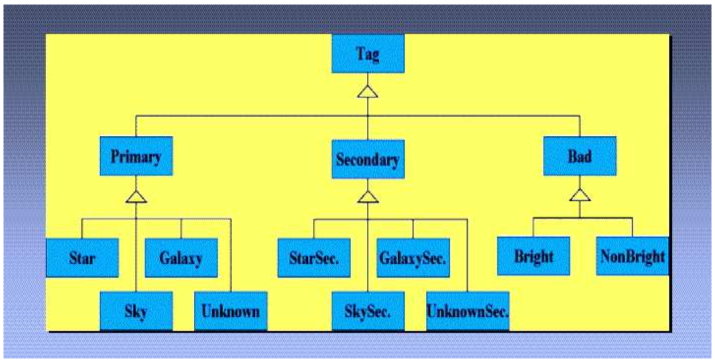

Fig. 3. The SX tag class hierarchy.

2.4 Server Multi-Threading

As mentioned above, the SX server incorporates multi-threading at several levels. This is illustrated in Fig. 4(a), and enables multiple simultaneous user sessions and multiple simultaneous queries within a user session. Fig. 4(b) shows an example of an SX Query Execution Tree - each node of which is executed as a separated thread. This parallel execution of nodes within a query maximizes SMP benefits: it yields I/O-bound performance on single processor and dynamical load-balancing among many processors. An ASAP data-push strategy is employed: query results are pushed up the tree as soon as they are available rather than at the end of the processing of each node.

The SX query primitives described below form the nodes of the Query Execution Tree. The Scoped Query nodes are the leaves, since they operate directly on the database.

- **Scoped Query** - A subset (scope) of the database is specified for the search, resulting in a list of database nodes to be searched. The query predicate is applied only to the selected type of objects within this scope.
- **Bag Query** - Query predicate is applied to each object in the given bag.
- **Association Query** - Select all objects linked to each object in the given bag through to-one or to-many associations and apply the query predicate to the selected objects.
- **Proximity Query** - Find all objects spatially close to each object in the given bag.
- **Set operations** - Union, intersection, and difference operations.

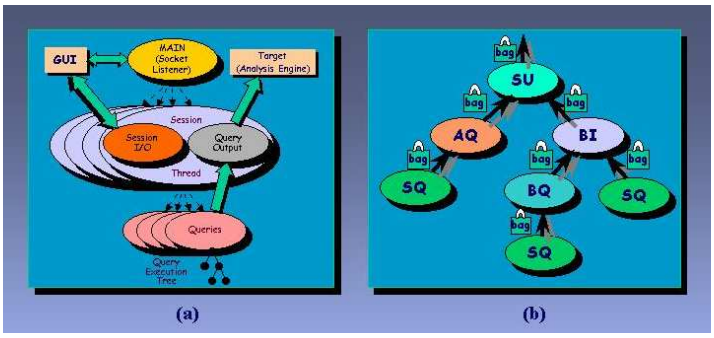

Fig. 4. (a) Local multi-threading in the SX server, and (b) a sample Query Execution Tree showing multi-threaded query execution (SQ=Scoped Query, BQ=Bag Query, AQ=Assoc. Query, SU=Set Union, BI=Bag Intersection).

Fig. 5 shows the parallel execution of queries in distributed mode. Remote slave servers running in parallel execute searches locally on remote partitions of the SX federation. Pointers to the objects selected on remote partitions are returned to the master server, which then performs extraction of the requested data from these objects. Currently the extraction is done entirely by the master, and can be a serious bottleneck for some queries. In the future the extraction will also be done in distributed mode.

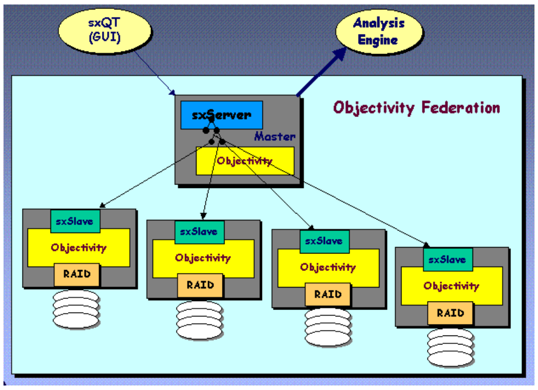

Fig. 5. Distributed query processing using remote multi-threading.

3 Performance Tuning

In spite of the design features described above, we have not achieved our goal of scalable, I/O-bound performance across diverse hardware. The multi-threaded performance of the Objectivity software has fallen well below our expectations, and early in our testing we uncovered a major performance bottleneck in Objectivity - opening individual objects was taking several thousand CPU cycles instead of a few hundred. This problem has not yet been corrected but is expected to be mostly resolved in the next release of Objectivity due out in the last quarter of 2000. With a beta release of this version, we have been able to achieve raw scanning speeds of nearly 100 Mb/s on Linux boxes that are capable of maximum speeds of 200 Mb/s in a cluster environment. Performance on an SMP SGI Origin 2000 has been disappointing by comparison, and we do not expect to do much better than 50 Mb/s on it in the near future.

We have performed a number of Objectivity benchmarks in order to squeeze the most out of the database software. We also log several usage statistics for the SX that help us to optimize the performance further, including the number of times that each object field is requested (this enables us to optimize the tag objects by including the highest impact fields in them) and the execution times of different types of queries in relation to the cost estimates. We also log usage by institution so that mirror sites can be deployed effectively in the future.

4 Future Enhancements

We plan to implement several enhancements in the near future with the aim of realizing even better performance and making the SX an archive template for a Virtual Observatory (VO).

1. **Member functions** - Frequently used functionality will be precompiled and stored in the archive as methods defined on objects.
2. **XML compliance** - We propose to use XML for communication between the client and server, and also use a scientific data DTD (document type definition) such as XSIL for object output/interchange.
3. **MPI-based distributed processing** - This will enable the SX to perform complex data analysis in a VO data-grid environment using MPI-based toolkits like Globus. Parallel extraction of object data will also be facilitated by MPI communication between nodes.
4. **Object cross-referencing with other archives** - We plan to employ parallel agent technology and fast linear methods for object lookup.

References

1. Szalay A.S. (1999) The Sloan Digital Sky Survey, *Computing in Science & Engineering*, Mar/Apr 1999, 54.
2. Szalay A.S., Kunszt P. Z., and Thakar A., Gray, J., Slutz, D., and Brunner, R.J. (2000) Designing and mining multi-terabyte astronomy archives: the Sloan Digital Sky Survey, Proceedings of the 2000 ACM SIGMOD on Management of Data, 451.
3. Kunszt P. Z., Szalay A.S. and Thakar A.R. (2000) Mapping the Sky using the Hierarchical Triangular Mesh, this proceeding.