Astronomical Data Analysis Software and Systems XIII
ASP Conference Series, Vol. 314, 2004
F. Ochsenbein, M. Allen, and D. Egret, eds.

Compare: A Scaleable and Portable Catalog Cross-Comparison Engine for the NVO

Serge Monkewitz, John Good

Infrared Processing and Analysis Center, California Institute of Technology, Pasadena, CA

Abstract. We describe the architecture of a general cross-comparison engine capable of spatially matching sources in one astronomical source catalog with those in another. The software is highly modular and is written in portable C++. By performing many cross-comparisons of small sky regions in parallel, the software will scale to very large input catalog sizes. Support is provided for common catalog formats and data sources (e.g. local disk, database servers), and the addition of support for custom data formats and sources is simplified by the modular architecture employed. Hooks for customized source pre-processing and match-list post-processing are also available. Taken together, these attributes will make Compare a powerful package for cross-comparing astronomical catalogs on all scales and for cross-identifying sources between catalogs, allowing it to serve the needs of both large projects and individual astronomers. In particular, the package will be installed at San Diego Supercomputer Center, where it will perform cross-comparison between large-scale catalogs (such as MACHO and 2MASS) housed there. When complete, it will be a cornerstone compute service for the NVO. We have applied an early version of the package to the cross-comparison of the SDSS Early Data Release and the 2MASS 2nd Incremental Data Release catalogs, a computation central to the NVO Brown Dwarf demonstration project. Despite being performed sequentially, the comparison of 9.8 million SDSS sources to 0.5 million 2MASS sources completed in approximately 100 seconds when run on a 4 CPU Sun V480 with 16GB of memory.

1. Introduction

The **Compare** software package is a framework for performing spatial joins between two lists of astronomical source positions. For each source s in a primary catalog P, it finds all sources in a secondary catalog S within a given angular distance d_m of s. This will be referred to as the *match list* for s, and corresponds to a list of candidates in S which might be observations of the same astronomical object as s. In addition, the software finds all the sources $s_n \in P$ such that $\forall t \in S$, $dist(s_n, t) > d_m$ (the *primary no-match list*), as well as all sources $t_n \in S$ such that $\forall s \in P$, $dist(s, t_n) > d_m$ (the *secondary no-match list*). The process of cross-comparing two catalogs is a necessary first step when cross-identifying observations from different missions, but also has many other applications. It can

for example be used to pick a "best" observation from a cluster of observations, to merge or group observations in some way, or to help identify artifacts in a catalog.

2. Design Goals

The primary goals targeted by the **Compare** software package are high performance regardless of input catalog size, generality (the ability to process catalogs of arbitrary size and format on a variety of run-time platforms), and extensibility. The existing cross-comparison codes at IRSA[1] suffer from various limitations which render them incapable of meeting these goals. They were written for a fixed hardware platform and do not make any attempt to be portable; they are tied to specific input and output data formats; finally, they operate on fixed column sets, limiting their use to specific catalogs. This last limitation means that any processing requiring column values not initially retrieved requires an additional pass through the potentially very large input catalogs. **Compare** is designed to overcome all of these limitations.

3. Design

The software, written in portable C++, is partitioned into five major components:

Data Access: The component responsible for reading source positions (as well as any other requested fields). Implementations which read data from ASCII/binary table files and from Informix database tables are provided.

Source List Processing: A component which can filter sources, as well as modify or generate the data fields associated with each source.

Cross-Comparison This component computes match lists as well as primary and secondary no-match lists.

Match List processing This component allows for customizable match list filtering and processing.

Data Storage This component is responsible for storing match and no-match lists. Implementations which store data to ASCII/binary table files are provided.

These are illustrated in Figures 1 and 2. Each component implementation conforms to a simple interface, and communication between different components is limited to the consumption and production of sources and match lists. This makes it easy to add support for new input/output data formats, cross-comparison algorithms, and source or match list processing modules. Writing a working cross-comparison application becomes a matter of choosing and linking component implementations.

---

[1] `Infrared Science Archive: http://irsa.ipac.caltech.edu`

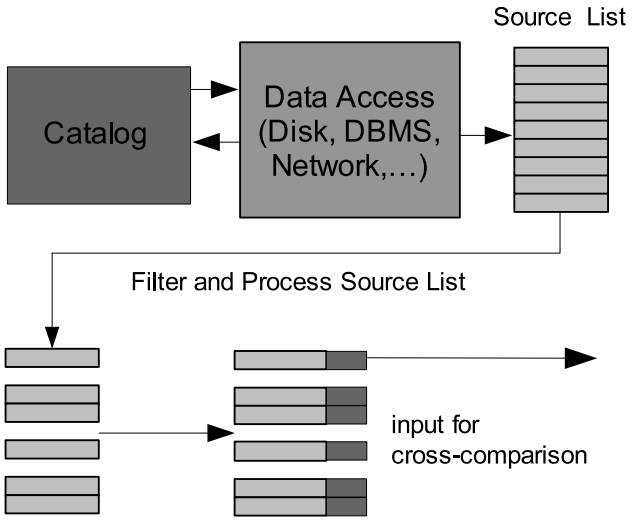

Figure 1. Data access and source list processing

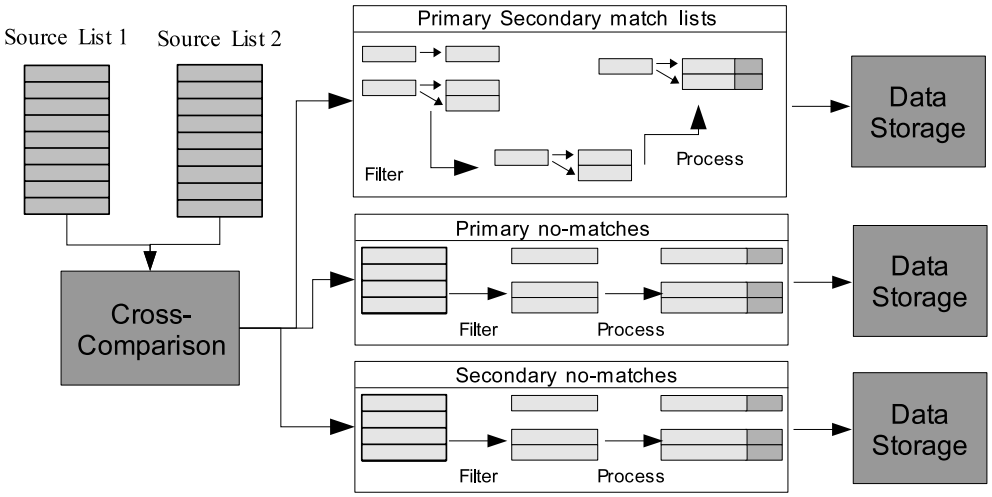

Figure 2. Cross-comparison, match list processing, and data storage

4. Implementation and Scaleability

The problem of scaling to very large catalog sizes is handled by splitting the sky into smaller disjoint regions which fit into machine RAM. A single **Compare** process is only capable of cross-comparing sources one region at a time; however, regions can be distributed across multiple **Compare** processes for simultaneous execution. It is important to note that the single largest performance bottleneck is I/O, so performance gains from parallelization will be modest unless I/O is also partitioned across multiple independent storage devices. Redundant I/O can be avoided by spatially indexing the input catalogs, and by performing all required source and match-list processing inside **Compare**. To summarize, high performance (when comparing large catalogs) requires high I/O bandwidth and spatially ordered data-access, a fast way to retrieve sources very close to a position, and a fast way to retrieve sources for larger regions of the sky.

4.1. Results

An early version of the **Compare** software was used by the NVO[2] Brown Dwarf Demonstration Project[3]. The prototype compared 9.8 million SDSS sources to 0.5 million 2MASS sources, finding 326020 source pairs within 3 arcseconds of each other in approximately 100 seconds. Roughly 80% of execution time was spent in I/O. Further confirmation that I/O is the major performance bottleneck for large-scale cross-comparisons is the fact that a series of simple SQL queries which retrieve the entire 2MASS working point source catalog (1.3 billion sources, 1.2TB of disk) take a total of around 4 days to complete.

5. Applications and Future Work

Compare was funded by the National Partnership for Advanced Computational Infrastructure as a demonstration project for Grid computing, and as such will be ported to the Tera-Grid as soon as it matures. Furthermore, spatial joins of the 2MASS, SDSS, USNO, and MACHO catalogs are planned, with results to be served as publicly available data sets. In the interest of promoting research, we would like to make source code for the software available to individual astronomers. In addition, there are several avenues of future development to explore. Firstly, support for more input formats (specifically the VOTable format) and data sources (RDBMSes other than Informix) is desirable. Secondly, performance could be improved by generating I/O code and data structures specific to a desired catalog column set at compile-time (or even run-time). Thirdly, allowing the use of SQL expressions to filter sources and matches (or to generate new column values) would allow astronomers unfamiliar with C++ to make wider and more efficient use of the software. Finally, including a module capable of determining whether or not the neighborhood of a source from one mission has been observed by another would be invaluable when processing primary and secondary no-match lists. Software availability is expected in early 2004.

---

[2] `http://us-vo.org`

[3] `http://irsa.ipac.caltech.edu/applications/WebCompare/nvodemo/`