StreamCorder: Fast Trial-and-Error Analysis in Scientific Databases

Etzard Stolte, Gustavo Alonso
Swiss Federal Institute of Technology, Department of Computer Science
ETH Zentrum, Zürich, Switzerland
{stolte, alonso}@inf.ethz.ch

1. Introduction

Scientific databases managing continuous observations of natural phenomena are currently storing multiple TBs of data and will soon reach the PBs range. Given their size, a very important part of the system load comes from scientists sifting through long stretches of noise in search of relevant events. This is a *trial-and-error* procedure where a scientist retrieves a data interval, processes it, determines whether the result of the analysis is acceptable, repeats the analysis if that is not the case or proceeds with the next data interval. In general, the large volume of necessary data-shipping and -processing inhibits interactive analysis. As a result, scientists are forced to coordinate lengthy offline event searches with online evaluation sessions.

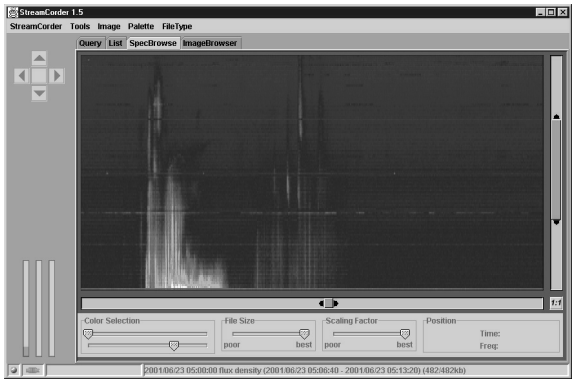

Figure 1. StreamCorder spectrogram analysis (full quality and resolution)

We have implemented a tool for fast trial-and-error analysis in the HESSI Experimental Data Center (*HEDC*)[1], a scientific database that will manage the high-energy solar observations generated by the High Energy Solar Spectroscopic Imager (*HESSI*)[2] satellite and that already houses an extensive collection of radio observations. The goal of HEDC is to facilitate access to the data and the creation of derived data products. It also aims at providing a convenient way to share scientific results in the area of solar astrophysics.

HEDC is implemented as a 3-tier architecture, where the intermediate layer relays data- and processing-requests by clients to an Oracle 8.1.6 DBMS and a number of analysis servers. HEDC currently runs on a SUN Enterprise Server with 2 GB RAM, two 450 MHz processors and a SUN A1000 RAID with 1.5 TB on-line storage space. In three years, HEDC is expected to manage multiple TBs of raw-data and millions of data products, which are mostly images.

We provide a "very" fat Java client for trial-and-error analysis, the *StreamCorder*[3] (see fig. 1). It includes dynamically loaded, context sensitive modules that access core services such as stream management, request queues and local analysis programs. The current context is kept across all modules. Currently available modules support browsing and download of all datatypes stored in HEDC, allow local and remote processing and offer administrative tools. During trial-and-error analysis, the StreamCorder coordinates the asynchronous download, caching, decoding and analysis of the data. A local database transparently caches query results and manages downloaded files. The local DBMS schema and the structure of the local file-system archive are identical to the ones on the server. Thus, in combination with semantic caching, global tuple identifiers and synchronization, offline work is possible. Through wavelet encoded view building (section 2) and client side data-quality and -resolution adaptation (section 3) fast trial-and-error analysis is practical.

2. View -generation and -encoding

During data-loading all *observations* of relevant satellite instruments are identified and extracted. Currently, these include the 18 photon detectors and the two positioning sensors, which together constitute around 70% of the raw-data.

[1] http://www.hedc.ethz.ch/
[2] http://hesperia.gsfc.nasa.gov/hessi/
[3] http://www.hedc.ethz.ch/release/

Every observation is a time-tagged tuple with less than 5 *attributes*. A cleaning, sorting and calibration step prepares attribute values for encoding and analysis. Next, a subset of $d+1$ attributes is selected for all tuples inside consecutive time ranges. For each time range, d-dimensional arrays are built. The size of each dimension is determined by the maximum sensitivity of the corresponding instrument. Outside of solar events only a few observations will be made, creating very sparse arrays or pre-computed *views*.

Figure 2. Low-quality/-resolution spectrogram (5% coefficients, 1/4 resolution)

The d-dimensional view is further divided along all dimensions into a number of d-dimensional *segments*. The optimal segment size depends on a multitude of system parameters influencing both file-handling overhead and query selectivity. All segments are wavelet encoded (using a commercial wavelet package) and stored inside file-system archives. Finally, summary information is extracted and, together with the file references, imported into the DBMS. During query processing, this information is used to match data requests to the appropriate view file(s).

3. Quality and resolution adaptation

During trial-and-error analysis encoded data segments are streamed to the client for processing, and data products are uploaded to the server for reuse. The StreamCorder adapts both data-quality and -resolution to speed up processing (see fig. 3). *Data-quality* depends on the percentage of the segment file used during decoding. Partially downloaded segment files yield approximated views of lossy to lossless quality and are the basis of *low-quality* data products. *Full-quality* views produce genuine, full-quality data products. Typical HESSI data and HEDC analysis algorithms yield already an acceptable data product with as little as 5% of the view coefficients (see fig. 1 and 2). Download time is thus shortened, while decoding and analysis time remain practically unchanged. Data-quality adaptation requires no changes to the analysis algorithms. *Data-resolution* depends on the decoded view dimension size. Most HEDC analysis algorithms accept variable size input data. The higher the degree of aggregation, the shorter the decoding time. As the cost of many important HEDC analyses is proportional to the size of the input data, analysis time is also much reduced. Due to the complexity of astrophysical analysis algorithms, it is not possible to provide generic error estimation routines. Therefore, either specific error routines or an interactive quality calibration mechanism is needed for every analysis.

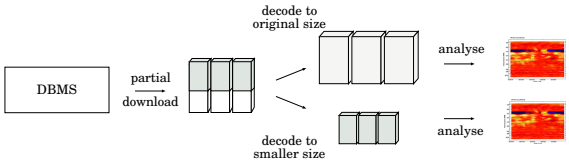

Figure 3. Adaptive decoding

Figure 4 displays the average overall handling time of spectrogram analyses for 55 view segment sizes with increasing data-quality. Data-resolution increases in 5 steps. An adequate spectrogram based on 5% of the segment coefficients and 1/4 of the resolution executes three times faster than the standard, full-quality/-resolution analysis (dark horizontal line). During trial-and-error analysis, 1/2 to 1/4 resolution is acceptable for most HEDC algorithms. Download time (over 100 Mb/s ethernet) is insignificant. Most of the processing is offloaded to the client, while the server focusses on query execution and file serving.

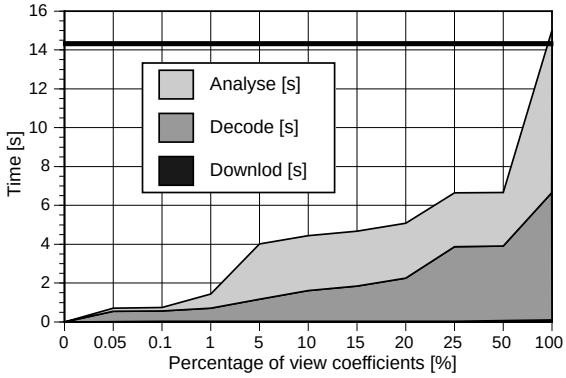

Figure 4. Fast trial-and-error analysis

4. Conclusions

We have implemented a client/server system for fast trial-and-error analysis. The server streams wavelet encoded views to the clients, where they are cached, decoded and processed. Low-quality decoding is beneficial for slow network connections. Low-resolution decoding greatly accelerates decoding and analysis. Depending on system resources, cached data and analysis requirements the user may alter minimum analysis quality at any time.