CONTROL: Continuous Output and Navigation Technology with Refinement On-Line

Ron Avnur, Joseph M. Hellerstein, Bruce Lo, Chris Olston, Bhaskaran Raman, Vijayshankar Raman, Tali Roth, and Kirk Wylie

Department of Electrical Engineering and Computer Sciences
University of California at Berkeley

{ronathan, jmh, brucelo, cao, bhaskar, rshankar, tali, wylie}@cs.berkeley.edu

ABSTRACT

The CONTROL project at U.C. Berkeley has developed technologies to provide online behavior for data-intensive applications. Using new query processing algorithms, these technologies continuously improve estimates and confidence statistics. In addition, they react to user feedback, thereby giving the user control over the behavior of long-running operations. This demonstration displays the modifications to a database system and the resulting impact on aggregation queries, data visualization, and GUI widgets. We then compare this interactive behavior to batch-processing alternatives.

1. INTRODUCTION

Despite significant advances in data processing over the years, many data-intensive applications still run in batch" mode: they require a long period of time to execute, and during that time they provide neither feedback nor control to the user. This state of affairs holds in a number of important domains, from the desktop (e.g. spreadsheets) to the back office (e.g. decision support) to new high-end applications (e.g. data mining and visualization tools). In each of these domains, batch processing is frustrating or even unacceptable for serious users. In order to investigate large data sets, users require "on-line" behavior: they need to get meaningful feedback while the system runs, and they need to be able to act on the feedback by controlling the system while it runs.

The goal of the CONTROL (Continuous Output and Navigation Technology with Refinement On-Line) project at UC Berkeley is to develop a set of core technologies for building applications that provide online behavior, and deploy those technologies in a variety of applications including database query processing, data mining, user-interface widgets and data visualization. We will demonstrate the following three applications of CONTROL:

1. Online Aggregation in a DBMS.
2. Online Data Visualization.
3. Gigantic Adaptive Dataset Graphical Element Tools (GADGETs)

We show the functionality of these applications by providing user interfaces that will allow visitors to participate in online queries and compare them with the batch-like alternatives. Based on this demonstration, we describe the core technologies that enable the applications to provide satisfactory online behavior.

In this text, we first discuss the demonstration of our algorithms for Online Aggregation. We then continue with an explanation of an Online Data Visualization demonstration, followed by a description of what we will show for GADGETs. We conclude briefly with an overview of the demonstration's theme.

2. ONLINE AGGREGATION IN A DBMS

Aggregation queries in relational database systems often require scanning and analyzing a significant portion of a database. In current relational systems such queries have batch behavior, requiring a long wait. The online version of the same queries reports a running estimate of the final aggregate in real time, along with a confidence measure during the processing of the query. The measure, or "confidence interval", says that with x% confidence, the current estimate is within an interval of size ε from the final answer. Reaching a small confidence interval quickly can allow the user to stop the query and go on to his/her next task. As a performance measure, online algorithms are designed to shrink the confidence interval faster while providing updated estimates at a satisfactory rate. To support this behavior, we demonstrate a new join algorithm called the Ripple Join [1], which allows the user to dynamically control the tradeoff between the rate at which the confidence interval shrinks (accuracy of the display), and the time between successive estimations (interactivity of the display). For GROUP BY queries, we would also like the user to see all the groups decrease their confidence intervals at similar rates, and also allow the user to control those

rates on the fly. New access methods Index Stride [3] and Online Permutation [2] provide this desired behavior.

We have implemented an Online Aggregation system by modifying the internals of Informix Universal Server to include Online Permutation, Index Stride and Ripple Join. Users at the demonstration will be able to submit queries and interact with the results using the Informix Metacube Explorer ROLAP tool which we have modified to serve as a front-end to our system. We describe the demonstration of each of these techniques below, beginning with the Stride algorithms, which are followed by Ripple Join.

2.1 Index Stride

Index Stride provides a user with simultaneously refining estimates for each group in a GROUP BY query. Without the algorithm, the distinct confidence intervals could not be refined concurrently, and the relative rates for the groups could not be controlled on the fly.

To understand the behavior of index stride, consider the following query:

SELECT ONLINE AVG(grade) FROM grades

GROUP BY major;

Index Stride first probes the index to find all the groups (majors). It then processes tuples from each group in a "round robin" schedule, improving each major's confidence interval and updating its running estimate. The user can manipulate the schedule at which different groups are processed by increasing, decreasing, or stopping the amount of time spent sampling each group as results arrive. The demonstration shows Index Stride and a typical Index Scan side by side to illustrate the improvement in usability that "even" group distribution provides over serial scanning. Unlike the Index Scan, Index Stride provides the user with the ability to choose which groups to favor at any given time.

2.2 Online Permutation

In the previous discussion, an index allows the algorithm to alternate between groups as tuples are fetched. However, if an index on the grouping columns does not exist we still desire the same query behavior. Moreover, the user should still be able to control the relative rates of groups on the fly. The Online Permutation algorithm uses a buffer and an auxiliary disk to permute the unordered tuples of a heap file on the fly, as described in [2]. The result is a simultaneous refinement of the different groups' confidence intervals, as if an index existed on the grouping column. Our demonstration shows the ordering of tuples in the input relation and how they are permuted to the desired output order. Like the Index Stride demonstration, we display how user preference affects the output ordering of tuples, and compare Online Permutation and Heap Scan to illustrate the usability improvement, while displaying an Index Stride as a reference.

2.3 Ripple Join

The traditional goal for join algorithms is to minimize the completion time. In contrast, the goal of online queries is to maximize the rate at which the confidence interval decreases. We have developed a family of Ripple Join algorithms, which improve the rate at which the confidence interval decreases for multi-table aggregation queries, and simultaneously provide updated estimates to the display in a timely fashion.

The central idea of Ripple Join comes from the fact that in order to generate a new confidence interval, a sample of one input relation must be fully joined with a sample of another input relation. To achieve this as quickly as possible, the ripple join alternates fetching from each of its input relations. When it fetches a new tuple from one relation, that tuple is combined with all previously seen tuples from the other relation. The ratio at which the join fetches from the two relations is critical to performance, and can be determined and modified dynamically by observing the statistical properties of the sets of tuples seen so far. The demonstration will use our implementation of Ripple Join in IUS to illustrate how Ripple Join reduces the confidence interval faster than the usual join algorithms, and allows the user to trade off the rate at which the confidence interval shrinks against the rate at which the display is updated.

3. ONLINE DATA VISUALIZATION

Visualizing large datasets is problematic because it takes a long time for all the data to be retrieved for display. Users must wait to begin interpreting the data until all the tuples have been retrieved and rendered. Many traditional techniques have batch-like characteristics. In contrast, Online Data Visualization allows the user to get intermediate results using density approximations while the remaining tuples are rendered.

If an R-Tree index exists for the spatial layout of the visualization, it can be used to provide the user with refining estimates of the final picture while the data tuples are being retrieved. Each internal node of the R-Tree provides a set of bounding rectangles for its child nodes. Rendering the internal nodes therefore gives a crude map of the dense areas of the visualization. While processing the tuples of the relation to draw them on the screen, Online Data Visualization also traverses the R-tree breadth-first to refine the density estimates. Each subsequent step down the tree provides a more accurate estimation of the final visualization. We will display a traditional rendering technique and our density refining technique side by side to illustrate Online Data Visualization as the clear usability winner.

4. GIGANTIC ADAPTIVE DATASET GRAPHICAL ELEMENT TOOLS (GADGETS)

Data from databases and other data sources is often displayed in user applications via graphical user interface "widgets". Unfortunately, these GUI widgets typically do not work for large data sets. We are developing a Java toolkit of Online GUI widgets (GADGETS) for use in applications managing massive or streaming data sources.

GADGETs are able to use the backend technology of online processing to handle large datasets and respond to the user s needs faster and more interactively than normal GUI widgets. With feedback about what the user wants, a GADGET can show results before it has all the data; as it receives more data from its source, it can begin populate itself with the preferred results first.

In addition, GADGETs utilize a more efficient storage method in order to be able to access the data faster. For our demonstration, we will build a GADGET that interacts with an online permutation source and is able to respond to the user browsing through the data, sorting it according to a certain column, or jumping to a certain part of the data. As user preference changes, the GADGET forwards that request to the online data source and thus begins to favor the desired values.

5. CONCLUSIONS

Batch processing is unattractive because it requires the user to wait while a large dataset is processed. In addition, the user may not require the fully accurate result if a "good" estimate is available much sooner. Online Aggregation, Visualization and GUI Widgets improve upon batch behavior by sending intermediate results to the user as soon as possible. By comparing the batch processing methods to our online algorithms we show that CONTROL improves the usability of large dataset processing.

6. ACKNOWLEDGMENTS

We would like to thank Chih-Po Wen and Paul Friedman of Informix for their help with the development of this software. We are also grateful to Mike Stonebraker, Cristi Garvey, and Robyn Chan of Informix for facilitating this work. Our research was sponsored by a grant from Informix Software Inc., a California MICRO grant, and a Sloan Foundation Research Fellowship.

7. REFERENCES

[1] Haas, P.J., Hellerstein, J.M. Join Algorithms for Online Aggregation. Submitted for publication.

[2] Hellerstein, J.M. Online Processing Redux. Data Engineering Bulletin, September 1997.

[3] Hellerstein, J.M., Haas, P.J., and Wang, H.J. Online Aggregation. In Proceedings of SIGMOD 1997 (Tucson, AZ, May 1997), ACM Press, 171-182.