ConEx: A System for Monitoring Queries

Chaitanya Mishra
University of Toronto
cmishra@cs.toronto.edu

Maksims Volkovs
University of Toronto
m.volkovs@utoronto.ca

ABSTRACT

We present a system, ConEx, for monitoring query execution in a relational database management system. ConEx offers a unified view of query execution, providing continuous visual feedback on the progress of the query, and the status of operators in the query evaluation plan. It incorporates novel techniques to dynamically estimate important parameters affecting query progress efficiently. We describe the design and features of ConEx, and discuss its technology.

Categories and Subject Descriptors

H.5.2 [**User Interfaces**]: Graphical user interfaces (GUI)

General Terms

Design, Management

Keywords

Query monitoring, Progress estimation, Database adminstration

1. INTRODUCTION

Database systems are often utilized to execute complex queries that may run for long periods of time. However, they provide only limited feedback on the progress of query execution towards completion. Such information can be of great utility to users and system administrators. For example, one may choose to terminate the query or change the query parameters if one is informed that the query is long running. Similarly, a system administrator can use this information to identify poor plans, inadequate tuning and resource bottlenecks.

There has recently been increasing interest in the development of progress indicators for SQL queries [4, 3, 2, 1, 5]. A progress indicator provides feedback about the fraction of work done by a query. At a high level, these indicators define a *model of work*, under which they *measure* the amount of work completed by the query, and *estimate* the amount of work that remains to be done. The ratio of the work done to the estimated total work is a measure of the progress made towards completion.

Typically, the visual interface advocated for this task is a progress bar based interface. Such an interface provides information about the percentage of work completed and estimated work remaining. Although such a simple interface works well for tasks like file downloads, we argue that it is too simplistic to provide informative feedback about the progress of execution of a query. This is because queries are often executed in pipelined stages separated by blocking points. A progress bar is a sufficient interface if the entire process is pipelined (such as network file download). However, if the process is broken into stages, it becomes difficult to predict the amount of work to be done by future stages. In such a setting, a simple progress bar interface provides limited, and potentially incorrect feedback.

However, most database systems already provide a visual utility (EXPLAIN) which provides detailed information *before* query execution. The EXPLAIN interface displays the structure of the query execution plan, and provides cardinality estimates of the intermediate operators in the plan. While this is useful, the initial cardinality estimates often have errors due to incomplete statistics or wrong assumptions. Therefore, we often end up with an incorrect picture of how the query execution will proceed.

The ConEx (Continuous Explain) system merges progress indicator technology with an EXPLAIN interface in order to provide a continuous view of query execution. Along with displaying information regarding work completed and an estimate of the work remaining like any progress bar, it also updates an EXPLAIN tree with improved cardinality estimates and information about the currently executing segments of the query plan. Therefore, ConEx offers dynamic EXPLAIN functionality, which we argue is more informative to users and administrators. The ConEx system incorporates a lightweight online statistics collection and estimation framework [5] which has been developed specifically for aiding progress indicators.

2. DESIGN

2.1 Requirements

A query monitoring system like ConEx must satisfy some requirements and operate under certain constraints. It should provide continuous online feedback on the current state of query execution. It should also provide predictions on the future behaviour of the query. At the same time, the technology should be minimally intrusive to query execution. A monitoring framework like ConEx maintains a lot of information about the currently executing plan. Such maintenance incurs runtime overhead and it is imperative to keep such overheads minimal as they could affect query performance. The technology behind ConEx has been designed with the principle of minimal impact to query performance in mind.

Finally, ConEx aims to provide access to information regarding a query at different levels of detail. A non-technical user might be satisfied with just a progress-bar based interface which provides information about how much work has been performed by the query

SIGMOD'07, June 12–14, 2007, Beijing, China.
ACM 978-1-59593-686-8/07/0006.

and estimates of work remaining. On the other hand, a system administrator might wish to discover more details regarding the query such as incorrect cardinality estimates (which point to insufficient statistics) and data distributions. We therefore implemented an interface which provides higher level information, and also supports zooming in to access pipeline and operator level information about a query.

2.2 System Architecture

ConEx consists of two main components: A frontend graphical user interface, and a backend database engine. The two components communicate using a simple socket interface. For the backend, we modified the Postgresql 8.0 database engine to support progress estimation and communication with the ConEx frontend. We augmented the engine with our online statistics collection and estimation framework [5] to provide accurate observations on the current state of the query, and estimate future behaviour of the query. The ConEx frontend is implemented in Java. It allows a user to submit queries and view results. We now describe the design of the visual interface in more detail.

2.3 Visual Interface

The ConEx frontend provides online feedback about the current state of the query. It consists of different components, each of which displays information at a different level of granularity

2.3.1 Progress Indicator

The frontend incorporates a progress bar which displays the percentage of work completed, and estimated work remaining for the query. We adopt a cardinality based notion of query progress [2]. The measure of work done by a query is the sum over all operators of the number of tuples processed by each operator during the execution of the query. If we let N_i be the total number of tuples to be processed by operator i, and K_i be the number of tuples processed by the operator till this instant then:

$$progress = \frac{\sum_i K_i}{\sum_i N_i}$$

We report this value as the progress estimate of the work performed. Our online cardinality estimation techniques refine the estimated number of tuples to be processed by an operator (N_i) during runtime providing the user with an accurate picture of query progress. A salient feature of our technique is that it provides high accuracy estimates early in the execution of the pipeline. These estimates provably converge to the correct values as more data is processed by the pipeline [5].

2.3.2 Query Plan Viewer

The Plan Viewer component extends the traditional EXPLAIN tree to provide the user with a graphical view of the query execution plan. Each node in the plan is annotated with the number of tuples processed by the node, and the current cardinality estimate. These values are continuously updated during query processing. In addition, the plan is visually partitioned into pipelines (a set of operators that execute concurrently) , with each pipeline marked with a different colour. This provides a visual representation of query progress in the form of completed stages which complements the progress bar. The query plan viewer forms the main window of the ConEx frontend.

2.3.3 Operator Viewer

The progress indicator, and query plan viewer components described previously provide a global view of query progress. However, one might want to drill down at the operator level to view more details about query behaviour. In order to enable this functionality, we support user interaction with the Query Plan Viewer. When a user clicks on a node in the query plan tree, detailed information about that node is displayed in the side panels of ConEx. This information includes:

Operator Information: In the top panel of the operator viewer, we display information on the number of tuples processed, estimated number of tuples to be processed, and the initial optimizer estimate for the node. In addition, we display operator specific information as well.

Cardinality Convergence: In a different panel, we display information about the convergence of our online cardinality estimates, and how these estimates compare to the original optimizer cardinality estimates for the node. This information, in the form of a convergence graph, provides confidence guarantees on the quality of the estimates produced by our framework.

Histograms: In the bottom panel, we display histograms on particular columns for blocking operators. These histograms are displayed on the join column for a hash operator, a grouping column for an aggregation operator, and on sorting columns for sort operators. We display the frequencies of the most common values in the column, and the average frequency of the remaining values.

2.4 Additional Features

ConEx supports additional features that make it particularly useful for identifying poor query execution plans. These features enable analysis during and after query execution.

Alerts: Our online cardinality estimation techniques provide accurate cardinality estimates early in the execution of a pipelined segment of the query execution plan. If these cardinality estimates differ significantly from the optimizer cardinality estimates we highlight the plan red to flag the event that the optimizer estimates are incorrect.

Runtime Profiling: We segment the progress bar into colour coded segments with each segment representing a pipelined stage of query execution. This enables us to identify how much time was spent in each segment of query execution. Such runtime profiling can be used to tune the system and optimize query execution time.

For example, suppose we notice that the query spends a major fraction of its execution time in sequentially scanning a large table, even though only a small fraction of tuples are actually used in the query. This points to the need to construct an index on the table to make query execution more efficient. Our runtime profiling information can provide feedback to an index advisor utility which suggests indices to be created on tables.

2.5 Sample Query

Figure 1 shows a screenshot of ConEx in action. The query being executed is a join between 4 relations. The query plan viewer shows that the execution plan generated by the query optimizer is a bushy plan tree with 3 hash joins. Query execution proceeds in 5 stages, with relations being hashed and joins being performed before the final count is computed. Currently the query is in its second pipelined stage. The node currently being clicked on is a hash node. We can see that the associated join is a one pass join (0 batches) with 1092 hash buckets in use. The histogram shows that the data is almost uniformly distributed with only a few high frequency values. The join pipeline being currently executed (pipeline 2) is highlighted to signal that the optimizer estimates are off by a significant amount. In this case, the original cardinality estimate for the hash join is 371 tuples, while the actual number of tuples produced is more than 100K. The progress monitor informs us that

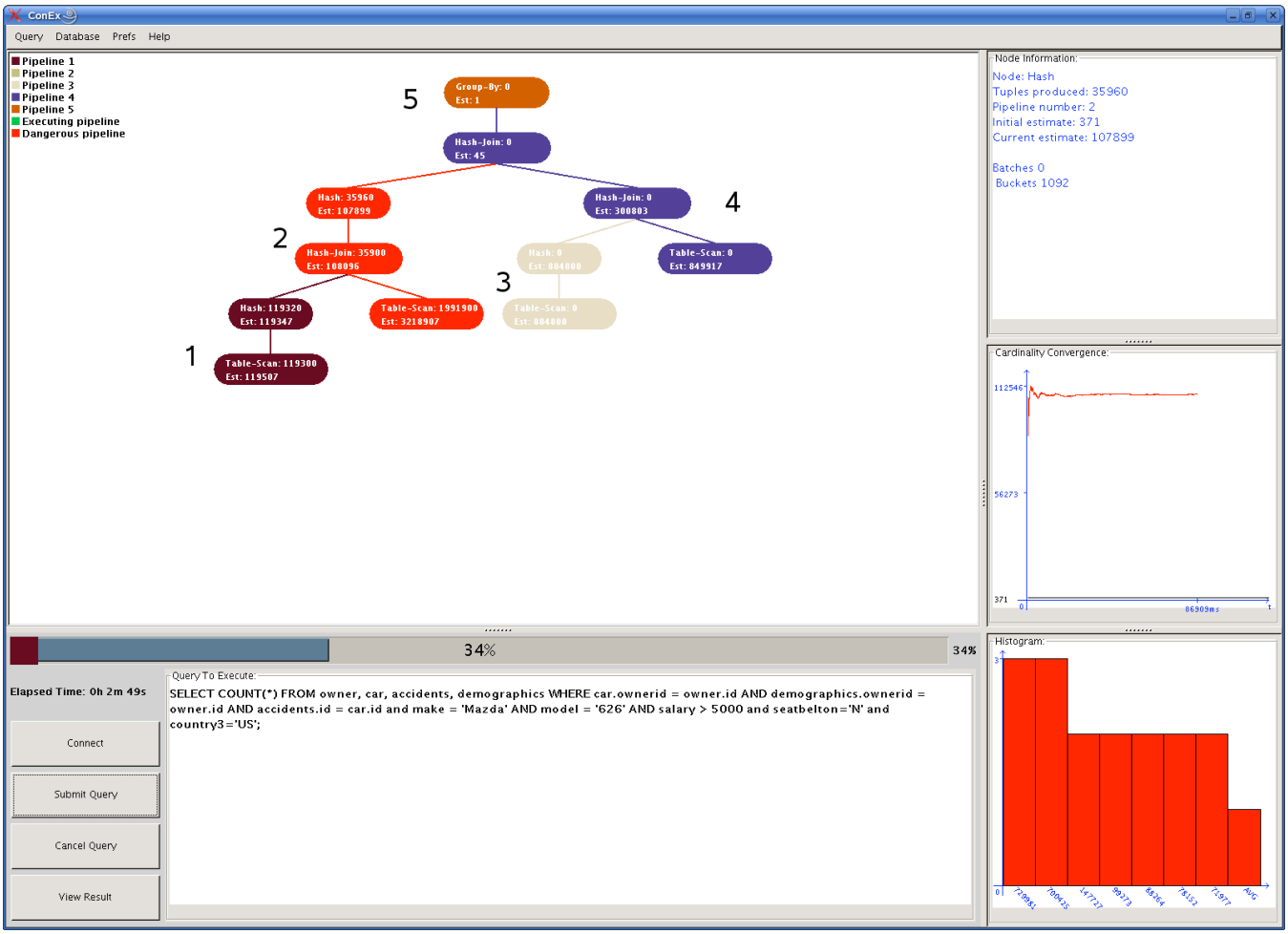

Figure 1: ConEx in Action

approximately 34% of query execution has completed. We can also observe the relative lengths of the coloured segments of the first and second pipelines. The first pipeline took a small time to execute and the work done by it is represented as the small coloured region at the left end of the progress bar. The second pipeline (which is still executing) takes comparatively more time than the fist one, and therefore represents more work done.

3. TECHNOLOGY

ConEx deploys an online cardinality estimation framework [5] developed in order to aid progress estimators. This framework monitors query execution at select points in the plan and collects statistics from the tuples processed at various points in a plan. The framework introduces a family of statistical estimators which utilize random samples and statistical summaries (such as histograms) to do online cardinality estimation at runtime.

4. DEMONSTRATION DESCRIPTION

Participants will be able to interact with ConEx using preloaded databases. The graphical interface is easy to use and the progress of various queries may be observed. The demonstration will highlight query instances where optimizer cardinality estimates are incorrect. We will demonstrate how ConEx obtains accurate cardinality estimates at runtime and show how this information serves to diagnose incorrect plans and insufficient statistics on tables. This will serve to show the utility of our system for long running queries. Participants will also be provided with information on the technology behind ConEx.

5. REFERENCES

[1] S. Chaudhuri, R. Kaushik, and R. Ramamurthy. When Can We Trust Progress Estimators for SQL Queries. *SIGMOD*, 2005.

[2] S. Chaudhuri, V. Narassaya, and R. Ramamurthy. Estimating Progress of Execution for SQL Queries. *SIGMOD*, 2004.

[3] G. Luo, J. Naughton, C. Ellman, and M. Watzke. Increasing the Accuracy and Coverage of SQL Progress Indicators. *ICDE*, 2004.

[4] G. Luo, J. Naughton, C. Ellman, and M. Watzke. Toward a Progress Indicator for Database Queries. *SIGMOD*, 2004.

[5] C. Mishra and N. Koudas. A lightweight online framework for query progress indicators. *ICDE*, 2007.