Aurora: A New Model And Architecture For Data Stream Management

B.Tech Seminar Report

by

Shafeek T K
ETAHECS044

Department of Computer Science And Engineering
Government Engineering College, Thrissur
December 2010

Acknowledgment

I would like to express my sincere gratitude to **Prof. Manoj Kumar**, head of the department, for providing an opportunity to present this seminar and **Prof.K S Valsaraj**, for giving guidance and supports on topic selection and presentation.

I deeply thank **Mr.Ajay James**, **Mrs.Baby Syla**, seminar coordinators, for guiding the whole process. I am also thankfull to **Mr.Savyan PV**, and other faculty members constituting evaluation panel of seminar.

Its my privilege to thank all my classmates for being a constant source of encouragement and being patient listeners throughout the presenation of the seminar and Almighty for everything.

Shafeek T K
December 2010
Govt. Engineering College, Thrissur

Contents

List of Figures

Abstract

This paper introduces a new class of application called monitoring applications, These applications show differ substantially from conventional data processing. The fact that a software system should process and react to, inputs in continuous manner from many sources like sensors rather than from operators requires one to rethink about the fundamental architecture of a DBMS for this area of application. In this paper, I present Aurora, a new model for DBMS that is constructed at Brown University,Brandeis University , and M.I.T. I describe the basic system architecture, set of operators, optimization methods, and support for real- time operation.

Chapter 1

Introduction

The focus of Traditional DBMS on business data processing. They are designed to meet these businesss reqiurements. Aurora is a new model for DBMS which is used manage data streams. This model is mainly useful for monitoring applications. The question comes if there are any particular reasons why we need a new system instead of using the traditional DBMSs. This paper gives five reasons as following:

1. Wrong basic computation model. Traditional DBMSs are a human-active, DBMS-passive (HADP) model. A DBMS as a passive repository storing a large amount of data elements. Aurora system is a DBMS-active, human-passive (DAHP) model. The monitoring application gets the data from external data source. The system notifies humans when abnormal activity is detected.

   Traditional DBMSs store time-series information is a challenge. Traditional DBMSs care about only current value. Monitoring applications require both current records and some history of values.

2. Most monitoring applications are trigger-oriented.They cannot scale a large amount of triggers per table. Traditional DBMSs treat triggers and alerters as second-class citizens.

3. Traditional DBMSs can't answer approximate query.It assume that data are synchronized. Therefore, the results of querying are exact. But in data stream-oriented applications, data comes asynchronously and are often lost, stale, or intentionally dropping for processing reasons (e.g. load shed).

4. Monitoring applications also have real-time requirements. Traditional DBMSs assume that the applications have no real-time requirements.

1.1 Organization Of the report

Here after each chapter in this report describes the patrs and features of Auroa system.

1. Chapter 2 Aurora system model.
2. Chapter 3 Aurora optimization
3. Chapter 4 Run-time operation.
4. Chapter 5 SQuAl: the Aurora query algebra.
5. Chapter 6 Related Work and Conclusions.

Chapter 2

Aurora system model

Mainly there are two kinds of data sources in Aurora. Incoming valuse at regular or irregular coming from program of a computer. The other is from hardware devices like sensors. A unique source identifier is used for every source. A timestamp is given by Aurora system for incoming tuple. A data stream is made up with several such data sources. They look like as following:

StockID, Time,Price	ID, Time, Position
(MSF, 1:00, 20)	$(1, 2:00, 3)$
$(IBM, 1:00$,16)	(2, 2:00, 5)

Aurora system is mainly used to process incoming data streams in the way defined by an application administrator. The fundamental elements in Aurora system are boxes and arrows that are similar to those in process flow and work flow systems. Boxes represent processing operations. Data stream, in terms of tuples, flow through a loop-free, directed graph. Output streams are sent to applications, which must be implemented to process the asynchronous tuples in an output stream. Arrows represent a collection of streams with common schema. Fig.1 illustrated the high-level system model.

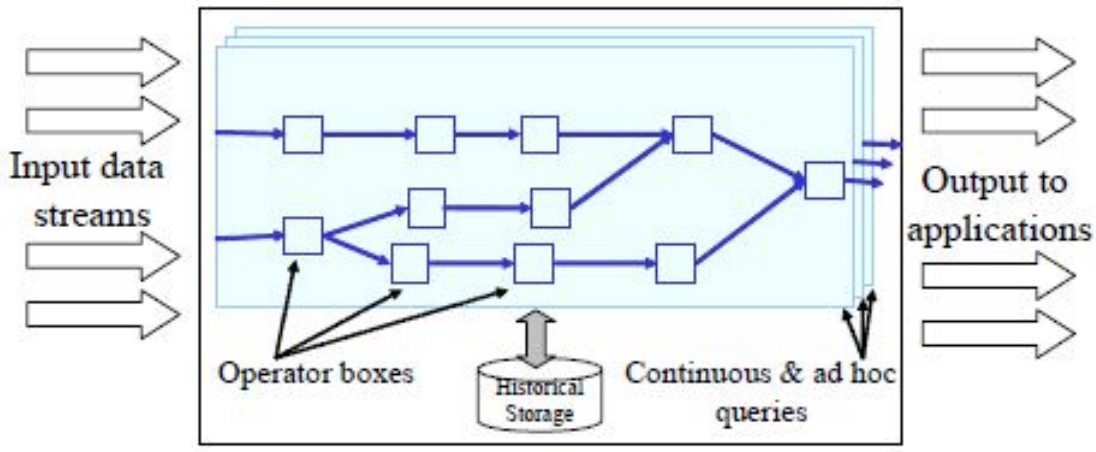

Figure 2.1: Aurora system model

Chapter 3

Query model

Each processes flows based on Quality-of-service(QoS) specifications each output in Aurora is related with graphs eith two-dimensionl QoS that show the utility of the output in terms of many performance-related and quality-related attributes.

Aurora supports three query models that all use the same building blocks. As we can see on the Fig.2, the topmost path is a continuous query. As data stream flows on boxes, it has been processed and no data elements are stored.

The dark circles on the input arcs to boxes b1 and b2 represent connection points. A connection point support dynamic modification to the network. The new boxes can be added or deleted from the connection point. A connection point can also cache data streams passed on it as persistent storage for a specified period. In Fig 2, the period is 2 hours. The middle path in Fig.2 represents a view. You can only see the path without the connected application. QoS specifications are also used to measure the importance of the view. An application can connect to the end of path at any time.

The bottom path represents an ad hoc query. An ad hoc query can initiate a search for a connection point and then attach itself to this connection point. An ad hoc query tries to find the earliest time T stored in the connection point and give the answers from T until the query is done.

Chapter 4

Aurora optimization

The main objective of query optimization in traditional DBMS is to minimize the iteration number. Aurora system has taken different strategies to optimize queries for better prfomance.

1. Computation requirements. Even if the amount of computation is very small for each operator to process those data stream, it is possible to have a large quantity of boxes.
2. Date rates may be high sometime
3. Dynamic changes may occur over time. The query optimization has to be done without offline.

4.1 Dynamic continuous query optimization

At the beginning, the user constructs an unoptimized Aurora network. Aurora system gathers runtime statistics such as the average cost of box execution and box selectivity during execution. Instead of optimizing the whole network, it selects a portion of the network to optimize. Furthermore, it will find all connection points that surround the subnetwork to be optimized.

1. **Inserting projections** is done not by application administrator but the operators. The optimizer will use operator signatures that describe the attributes that are used and produced by the operators.
2. **Combining boxes** is to combine two boxes into a single box to reduce some cost.

3. **Reordering boxes** is to change the orders of the boxes

4.2 Ad hoc query optimization

Aurora processes ad hoc queries in two steps by constructing two separate subnetworks. Each is attached to a connection point. The historical subnetwork runs first. The initial boxes in an ad hoc query pull information from the B-tree. Each node of the B-tree corresponds to the connection points. When the historical operation is done, Aurora changes mode to push-based mode to continue processing.

Chapter 5

Run-time operation

Aurora run-time network is to process floe of data(data stream) through a potentially large workflow diagram. In Fig.3 illustrates the basic architecture of Aurora . The duty of storage manager is to maintain the box queues and managing the buffer. The scheduler picks a box for execution, finds what processing is required, and passes a pointer to the multithreaded box processor. The QoS monitor and the load shedder work together. The QoS monitor continually monitors system performance and triggers the load shedder when it detects an overload situation and poor system performance. The load shedder then sheds load until the performance of the system reaches an acceptable level.

5.1 QoS data structures

There exist the three criteria to decide the QoS in Aurora system. They are Response, Tuple drops, and Values produced.

5.2 Storage management

Aurora Storage Manager (ASM) is used to store all tuples needed by Aurora network. There are two kinds of storage management. They are Queue management and Connection point management. Queue management manages storage for those tuples passed through an Aurora network. Connection point management arranges the storage for connection points.

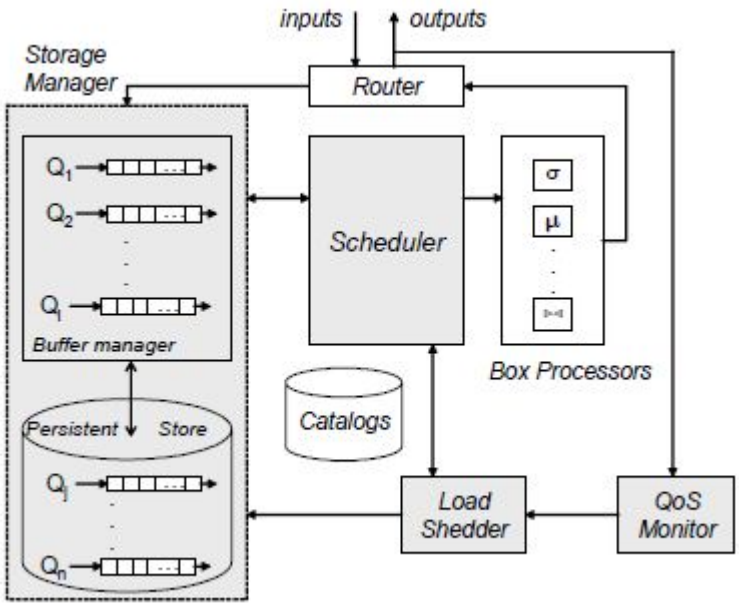

Figure 5.1: Aurora run-time architecure

5.3 Real-time scheduling

Scheduling in Aurora is a complex problem, which is related with several issues including large system scale, dynamic performance requirements, dependencies between box executions, and multiple accesses to secondary storage. Aurora system not only maximizes overall QoS but also tries to reduce overall tuple execution costs.

1. **Train scheduling** s used to describe a batch of mutiple tuples as input to a single box. Aurora system exploits two basic nonlinearities when processing tuples, and then reduce overall processing costs. (a). Interbox nonlinearity. The goal of the system is minimize the number of I/O operations processed per tuple. The system schedules the data stream to reduce tuple trashing when there is not enough buffer space. The system also schedules the tuples between two boxes without ASM intervening. (b). Intrabox nonlinearity. The goal is to minimize the number of box calls made per tuple by processing complete trains at once. It will reduce the overhead such as calls to box code and context switch.

2. **Priority assignment** The priority of an output is to meet the run- time requirement. Aurora currently considers two approaches for priority assignment. (a). A state-based approach assigns priorities to outputs based on their expected utility under the current system state and then picks for execution. The utility

of an output is decided by evaluating the lost of QoS if the execution of the output is deferred. (b) A feedback-based approach. It continuously monitors the performance of the system and dynamically reassigns priorities to outputs.

3. **Putting it all together** Superbox scheduling is to describe scheduling actions that push a tuple train through multiple boxes. In this case, The system asks storage manager to put all boxes into memory.

4. **Scheduler performance** There are some measurements against Aurora prototype.(a)The time used in the scheduler can be reduced by a factor of 0.48 when we shift from a box-at-a-time scheduling discipline to using tuple trains. (b) Adding a simple version of superbox scheduling decreases the time spent in the scheduler by an additional factor of 0.43. (c) The overall execution costs are also reduced.

5.4 Introspection

Aurora uses static and dynamic introspection techniques to predict and detect overload situations.

1. **Static analysis** is to determine if the system have enough computational power.

2. **Dynamic analysis** is to determine the system performance if the system is under expected condition, unpredictable, long-duration spikes in input rates.

5.5 Load shedding

When Aurora system detect an overload, it will decrease the tuple volume by load shedding. This can be done by dropping tuples, which is similar to dropping overflow packets in packet-switching networks. Aurora has two methods to reduce the volume, at the same time, without potential problems.

1. **Load shedding by dropping tuples**. For dynamic analysis, the effect of load shedding can use delay-based QoS. If the waiting time for output is beyond certain thresholds, then we continue the load-shedding process until the latency is acceptable. For static analysis, it can use drop-based QoS. We try to put the drop box as far upstream as possible until with hit the point where we find that the drop box will affects other output.

2. **Semantic load shedding by filtering tuples** It drops tuples in a more controlled way instead of randomly selected. Basically, it means that it drops less

important tuples by using filters. The effect of load shedding can use value-based QoS. The system observes the value ranges to create a filter predicate.

Chapter 6

SQuAl: the Aurora query algebra

Here is the basic operators used for process data stream in Aurora.

1. **Filter** is like a case statement.It can route input tuples to alternative streams. Filter(P1,P2 . . . , Pm)(S) P1,P2 . . . , Pm are predicates over tuples on the input stream, S

2. **Map** is similar to relational projection but can apply any functions to tuples. Map (B1 = F1, . . . , Bm = Fm)(S) B1, . . . , Bm are names of attributes and F1, . . . , Fm are functions over tuples on the input stream, S

3. **union** can merge two or more streams into a single outpur stream. Union (S1, . . . , Sn) S1, . . . , Sn are streams with common schema.

4. **BSort** is an approximate sort operator that takes the form: BSort (Assuming O) (S) O = Order (On A, Slack n, GroupByB1, . . . , Bm) A,B1, . . . , Bm are attributes and n is a nonnegative integer

5. **Aggregate** applies window functions to sliding windows over its input stream. This operator has the form: Aggregate (F, Assuming O, Size s, Advance i) (S) F is a window function O = Order (On A, Slack n, GroupBy B1, . . . , Bm) is an order specification over input stream S, s is the size of the window and i is an integer.

6. **Join** is a binary join operator that takes the form: Join (P, Size s, Left Assuming O1, Right Assuming O2)(S1, S2) P is a predicate over pairs of tuples from input streams S1 and S2, s is an integer, and O1 (on some numeric or time-based attribute of S1,) and O2 (on some numeric or time-based attribute of S2).

7. **Resample** is an asymmetric, semijoin-like synchronization operator. This operator takes the form:Resample (F, Size s, Left Assuming O1, Right Assuming

O2) (S1, S2) F is a window function over S1. s is an integer, A is an attribute over S1 , O1 (on some numeric or time-based attribute of S1) and O2 (on some numeric or time-based attribute of S2).

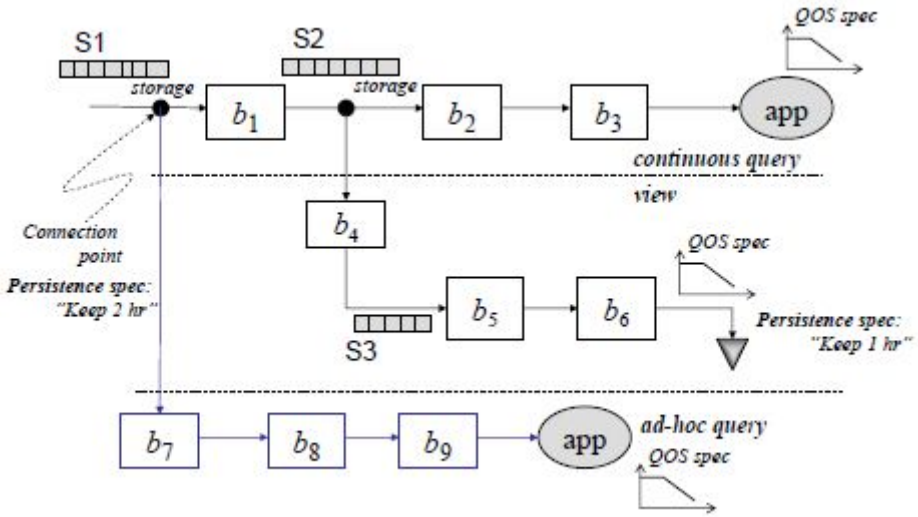

Figure 6.1: Aurora query model

Chapter 7

Related Work and Conclusion

Aurora system is related with the following fields.These are research fields: (1)active databases (monitoring conditions) (2) query indexing(3) continuous query(4)adaptive query processing techniques system (5) stream data query processing architectures (6) SEQ model (7) The Chronicle Data Model (8) materialized views.

Aurora system may benefit from and contribute to the following research fields: (1) temporal databases, main-memory databases and real-time databases (2) scheduling tasks in real-time and multimedia systems and databases (3) The congestion control problem in data networks and its load-shedding mechanism (4) approximate query answering (load shedding)

Aurora system can functionally correct, but there is no optimization and load shedding. The teams are working on competing scheduling algorithms and extending the functionality of ASM.

There are two further works for Aurora system: (1) will provide support for distribution. (2) will extend basic data and processing model to deal with missing and imprecise data values, which are common for the sensor-generated data streams

References

[1] Abadi, D., Carney, D., Cetintemel, U., Cherniack, M., Convey, C., Lee, S.,Stonebraker, M., Tatbul, N., and Zdonik, S., "Aurora: A New Model and Architecture for Data Stream Management", Journal of Very Large Data Bases(VLDB), Vol. 12 No.2, pp.120-139, August 2003 .

[2] R. Avnur and J. Hellerstein. Eddies: ContinuouslyAdaptive Query Processing. In Proc. of the 2000 ACM SIGMOD Intl. Conf. on Management of Data, Dallas,TX, 2000.

[3] D. Zhang, D. Gunopulos, V. J. Tsotras, B. Seeger: Temporal Aggregation over Data Streams Using Multiple Granularities. EDBT 2002: 646-663