Data Stream Management: Efficient Data Processing in Network Traffic

Mrs.Swapna Vanguru, Assistant Professor(CSE), ACE Engineering College(Autonomous),Hyderabad
Ms.Shubhangi Mahule, Assistant Professor(CSE), ACE Engineering College(Autonomous),Hyderabad
Mrs.P.Swaroopa, Assistant Professor(CSE), ACE Engineering College(Autonomous),Hyderabad

Abstract— Due to the sheer volume of data and complex processing, most current network traffic management applications process the collected data offline. Data after loaded into data warehouse it can be processed into offline and online. Offline Processing is appropriate for applications using decision-support type queries. like Capacity Planning and Provisioning or determining pricing plans. However, many other traffic management applications would benefit from online data processing. Like Tracking changes in network topology and traffic distribution online would enable congestion cause detection, Adaptive intra-domain and inter-domain policies.

Index Terms— Sheer volume of data, Network, Topology, Congestion, Congestion Cause Detection, Intra domain, Inter domain

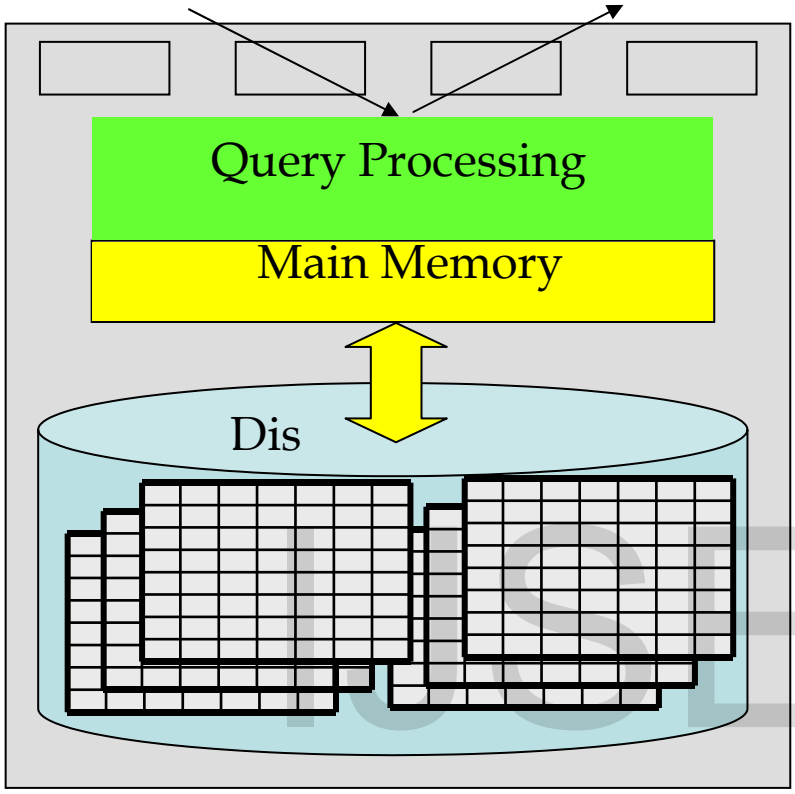
IJSER

1. INTRODUCTION

A Data Stream Management System(DSMS) is a computer software system to manage continuous data streams. Which is very similar to a Database Management System(DBMS). However, designed for static data in conventional databases. A DSMS also offers a flexible query processing so that the information needed can be expressed using queries. However, in contrast to a DBMS, a DSMS executes a continuous query that is not only performed once, but is permanently installed. Therefore, the query is continuously executed until it is explicitly uninstalled. Since most DSMS are data- driven, a continuous query produces new results as long as new data arrive at the system. This basic concept is similar to complex event processing so that both technologies are partially coalescing.

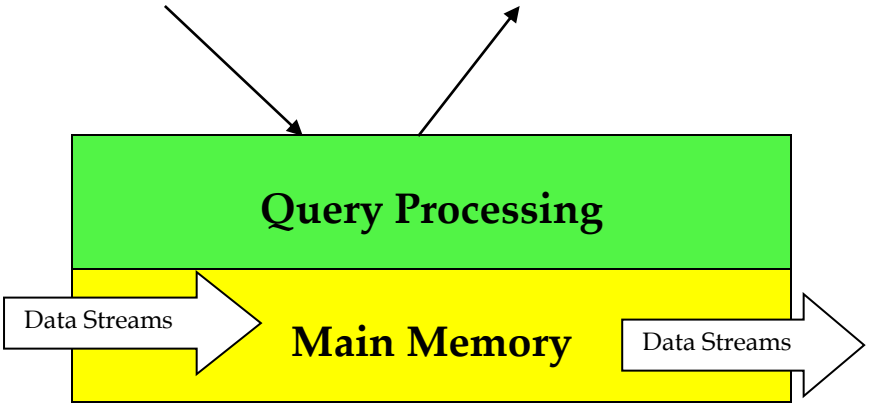
1.1 Handle Data Streams in DBS

SQL Query **Result**



DSMS

Register CQs Result(stored)



1.2 Query Processing

Since there are a lot of prototypes, there is no standardized architecture. However, most DSMS are based on the query processing in DBMS by using declarative languages to express queries, which are translated into a plan of operators. These plans can be optimized and executed. A query processing often consists of the following steps.

2. Formulation of continuous queries

The formulation of queries is mostly done using declarative languages like SQL in DBMS. Since there are no standardized query languages to express continuous queries, there are a lot of languages and variations. However, most of them are based on SQL, such as the Continuous Query Language (CQL), Stream SQL and ESP. There are also graphical approaches where each processing step is a box and the processing flow is expressed by arrows between the boxes.

The language strongly depends on the processing model. For example, if windows are used for the processing, the definition of a window has to be expressed. In Stream SQL, a query with a sliding window for the last 10 elements looks like follows:

```
SELECT AVG(price) FROM example stream [SIZE 10 ADVANCE 1 TUPLES] WHERE value > 100.0
```

IJSER

This stream continuously calculates the average value of "price" of the last 10 tuples, but only considers those tuples whose prices are greater than 100.0.

In the next step, the declarative query is translated into a logical query plan. A query plan is a directed graph where the nodes are operators and the edges describe the processing flow. Each operator in the query plan encapsulates the semantic of a specific operation, such as filtering or aggregation. In DSMSs that process relational data streams, the operators are equal or similar to the operators of the Relational algebra, so that there are operators for selection, projection, join, and set operations. This operator concept allows the very flexible and versatile processing of a DSMS.

2.1 Optimization of queries

The logical query plan can be optimized, which strongly depends on the streaming model. The basic concepts for optimizing continuous queries are equal to those from database systems. If there are relational data streams and the logical query plan is based on relational operators from the Relational algebra, a query optimizer can use the algebraic equivalences to optimize the plan. These may be, for example, to push selection operators down to the sources, because they are not so computationally intensive like join operators.

Furthermore, there are also cost-based optimization techniques like in DBMS, where a query plan with the lowest costs is chosen from different equivalent query plans. One example is to choose the order of two successive join operators. In DBMS this decision is mostly done by certain statistics of the involved databases. But, since the data of a data streams is unknown in advance, there are no such statistics in a DSMS. However, it is possible to observe a data stream for a certain time to obtain some statistics. Using these statistics, the query can also be optimized later. So, in contrast to a DBMS, some DSMS allows to optimize the query even during runtime. Therefore, a DSMS needs some plan migration strategies to replace a running query plan with a new one.

IJSER

2.2 Transformation of queries

Since a logical operator is only responsible for the semantics of an operation but does not consist of any algorithms, the logical query plan must be transformed into an executable counterpart. This is called a physical query plan. The distinction between a logical and a physical operator plan allows more than one implementation for the same logical operator. The join, for example, is logically the same, although it can be implemented by different algorithms like a Nested loop join or a Sort-merge join. Notice, these algorithms also strongly depend on the used stream and processing model. Finally, the query is available as a physical query plan.

2.3 Execution of queries

Since the physical query plan consists of executable algorithms, it can be directly executed. For this, the physical query plan is installed into the system. The bottom of the graph (of the query plan) is connected to the incoming sources, which can be everything like connectors to sensors. The top of the graph is connected to the outgoing sinks, which may be for example a visualization. Since most DSMSs are data-driven, a query is executed by pushing the incoming data elements from the source through the query plan to the sink. Each time when a data element passes an operator, the operator performs its specific operation on the data element and forwards the result to all successive operators.

3. Network Traffic Management:

The problem of obtaining the best possible network performance in the growing Internet has given rise to the need for efficient network *traffic management*. Broadly, traffic management can be divided into three tasks:

IJSER

(1) *collecting* data, e.g., network topology and utilization data, router con- figuration data;

(2) *processing* the collected data, e.g., to detect problems such as link failure, to determine the best ways to optimize network performance;

(3) *deploying* mechanisms for controlling network traffic.

We focus on tasks (1) and (2) in the context of a large Internet Service Provider (ISP) such as AT&T or Sprint.

Data collected to enable traffic management applications in an ISP includes: network packet and flow traces; active measurements of packet delay, loss, and throughput; router forwarding tables and configuration data; and *Simple Net- work Management Protocol* (SNMP) data maintained by various network elements (e.g., routers, switches).

Since networks need to be running all the time, much of this data is collected continuously (although on different time scales) and results in very large and fast-growing databases. For example, packet traces collected in the Sprint IP backbone amount to 600 Gigabytes of data per day.

Different kinds of processing must be performed on the collected data to enable sophisticated traffic management applications.

The *network topology* is maintained by joining SNMP data and/or configuration data from different network elements. Statistics of *link and router utilization* are maintained by aggregating packet traces or from SNMP data. *Packet losses*, *per-hop* and *end-to-end delays*, and *network throughput* are measured either by joining packet traces collected from multiple points in the network, or by using a dedicated system that generates network traffic to measure these parameters online. The recently proposed *model for traffic demands* in an IP backbone network is computed by joining four different data sources—network flow traces, router forwarding tables and configuration data, and SNMP data.

IJSER

4. Challenges in Data Processing

Due to the sheer volume of data and complex processing, most current network traffic management applications process the collected data offline. The data is typically loaded into a centralized file system or data warehouse (e.g., *Daytona* [2]) and processed using software toolkits. Offline processing is appropriate for applications using decision-support type queries, e.g., capacity planning and provisioning or determining pricing plans [2]. However, many other traffic management applications would benefit from online data processing. For example, tracking changes in network topology and traffic distribution online would enable congestion cause detection, adaptive intra-domain and inter-domain routing policies (e.g., adjusting OSPF weights and BGP policies [4]), resource allocation mechanisms for guaranteed application-level quality of service (QoS), admission-control and traffic-policing, and detecting denial-of-service attacks.

For some of these applications (e.g., network topology maintenance, congestion cause detection, traffic demand computation) data can still be collected and processed in a central place, but they would benefit from a system that per- forms its processing in a continuous fashion, requiring efficient online data processing techniques. Other applications might require distributed online computation. For example, admission-control and traffic-policing require complex processing over incoming packet streams at different routers (or different machines dedicated to data collection and processing). Collecting and processing data in a central place is difficult due to the real-time response requirements of these applications and the additional load that would be placed on the network. One solution is to maintain a table capturing aggregate properties of the incoming packet stream at each router and transfer (only) significant updates to this table to a central place for further processing. It is essential to keep the table as well as the additional processing overhead at each router within reasonable limits.

IJSER

The structure of an ISP further motivates the need for distributed processing. The *back-bone routers* of an ISP are significantly more crucial and sensitive than its *access routers*. Therefore, it is preferable to distribute the tasks of data collection and (online) processing to machines monitoring the links between the access and backbone routers or, to a lesser extent, among the access routers, rather than directly among the backbone routers.

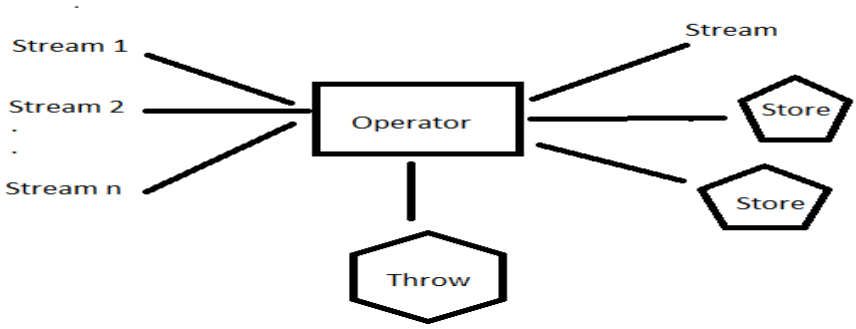


Figure 1: Architecture of a data processing operator in a DSMS.

IJSER

5. A Data Stream Management System

To integrate data collection and processing, and to enable on- line (as well as offline) processing, we propose the use of a *Data Stream Management System* (DSMS) for deploying traffic management applications. The difference between a file system or Database Management System (DBMS) and a DSMS is simple: current file systems or DBMSs expect all data to be managed within some form of persistent *data set*; in a DSMS, the concept of a *data stream*, possibly un- bounded, is as important as a conventional stored data set. By nature, a stored data set is appropriate when significant portions of the data are queried again and again, and updates are small and/or relatively infrequent. In contrast, a data stream is appropriate when the data is changing constantly (often exclusively through insertions of new elements), and it is either unnecessary or impractical to operate on large portions of the data multiple times.

To integrate data collection and processing, and to enable online(as well as offline) processing. I propose the use of a Efficient Data Stream Management System (E-DSMS) for deploying traffic management applications. One solution is to maintain a table capturing aggregate properties of the incoming packet stream at each router and transfer only significant updates to this table to a central place for further processing.

It is essential to keep the table as well as the additional processing overhead at each router within reasonable limits. The structure of an ISP further motivates the use for distributed processing.

The Efficient Data Stream Management System (E-DSMS) is to build a complete DSMS, with functionality and performance similar to that of a traditional DBMS, but which allows some or all of the data being managed to come in the form of unbounded, possibly very rapid, data streams. For details of the envisioned system. Data processing operators in a DSMS have the general architecture shown in Figure 1. Each operator works on a set of input data streams and produces an output stream (the *Stream* component in Figure 1).

IJSER

Data may be saved in the *Scratch* component for future use, or it may be discarded (the *Throw* component in Figure 1). Current results are stored in the *Store* component and/or sent to the output stream, which could serve as an in- put to another operator. These operators are designed particularly for online processing over continuous data streams with bounded amounts of storage. For instance, operators with very rapid or unpredictable input streams could adapt and provide approximate answers in real-time (e.g., by sampling the streams). In general, the data stream processing model of *one-pass computation* using bounded amounts of storage also provides a scalable alternative for offline data processing over fast-growing databases. Processing traffic management applications over a distributed traffic management infrastructure further motivates the use of data stream processing.

Let us revisit some of the network traffic management applications discussed in Sections 1 and 2 supposing we have the support of a DSMS. The collected data is fed to the DSMS as data streams and the applications are enabled using data processing operators based on the general architecture.

For example, one operator is a *network topology monitor* operating on a continuous feed of router up/down status and con- figuration data. Since this operator needs to join data from configuration files of different routers, some data would need to be saved in *Scratch*. *Store* would contain the current network topology, while updates to the topology could be streamed to other operators (e.g., an *intra-domain route monitor*).

As a second example, consider traffic demand computation in an IP backbone network . This application can use an *inter-domain route monitor* operator which maintains the set of outbound links carrying traffic from the backbone to specific external IP addresses. The inter-domain route monitor operates on a continuous feed of router forwarding tables (or BGP advertisements) and configuration data. The general operator architecture also captures the needs of applications that maintain aggregate properties of the in- coming packet stream at routers (e.g., admission-control and traffic-policing). The DSMS can also be used to integrate and improve offline processing (e.g., decision-support type queries) by applying the same operators to data streams generated from the collected data.

IJSER

References

[1] S. Babu and J. Widom. Continuous queries over data streams. Technical report, Stanford University Database Group, March 2001. Available at http://dbpubs.stanford.edu/pub/2001-9.

[2] R. Caceres et al. Measurementand analysis of IP network usage and behavior. *IEEE Communications Magazine*, 38(5):144– 151, 2000.

[3] N. G. Duffield and M. Grossglauser. Trajectory sampling for direct traffic observation. In *Proc. of the 2000 ACM SIGCOMM*, pages 271–284, September 2000.