Load Management and High Availability in the Medusa Distributed Stream Processing System

Magdalena Balazinska, Hari Balakrishnan, and Michael Stonebraker

Massachusetts Institute of Technology
http://nms.lcs.mit.edu/projects/medusa/

{mbalazin,hari,stonebraker}@csail.mit.edu

ABSTRACT

Medusa [3, 6] is a distributed stream processing system based on the Aurora single-site stream processing engine [1]. We demonstrate how Medusa handles time-varying load spikes and provides high availability in the face of network partitions. We demonstrate Medusa in the context of Borealis, a second generation stream-processing engine based on Aurora and Medusa.

1. INTRODUCTION

Over the past few years *stream processing* has emerged as an important area for data management. Many single-site continuous query processing engines have been developed, including Aurora [1], STREAM [2], TelegraphCQ [4], and NiagaraCQ [5]. Our work is in the area of *distributed* stream processing, with the goal of developing a system comprised of many single-site query processors working in concert. *Distributing* stream processing across multiple machines and sites has many advantages:

1. Distribution allows stream processing performance to be incrementally scaled to handle increasing input loads.
2. Distribution enables high availability (HA) because the processing nodes can monitor and take over for each other when failures occur.
3. Geographic and administrative distribution is inherent in certain stream processing applications such as financial services, network health monitoring, and wireless sensor networks. The system can gain in efficiency by leveraging this distribution. The composition of stream feeds from different participants also enables the creation of complete end-to-end services from these feeds.
4. Distribution and collaboration between participants in different administrative domains allows participants to cope with sudden load spikes without individually having to maintain the computing, network, and storage resources required for peak operation.

Medusa is a distributed stream-processing system that aims to provide the above benefits. Our work focuses in particular on (1) wide-area distribution and collaborations between autonomous participants, (2) load management, and (3) high availability.

SIGMOD 2004 June 13-18, 2004, Paris, France.

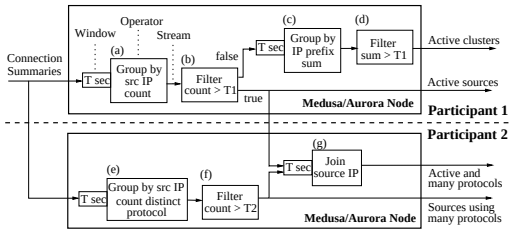

Figure 1: Example of a distributed stream-processing query.

Figure 1 illustrates a query for a network intrusion detection application. Suspicious hosts are identified by monitoring the number of connection attempts and protocols used by any given IP source within a short period of time. The query crosses node and administrative boundaries. We use this query in the Medusa demonstration.

2. LOAD MANAGEMENT

Medusa assumes a federation of autonomous participants, each operating one or more nodes with computing, storage, and network resources. Autonomous participants do not collaborate for the benefit of the whole system, but rather aim to maximize their own benefit. In Medusa, we therefore adopt an agoric model to create the right incentives for participants to handle each other's load.

Many computational economies have been proposed in the past (e.g., [7, 12]), but their deployment has remained limited due to their complexity, excessive overhead, and overly optimistic assumptions on participants' level of collaboration. To address these issues, in Medusa, we instead propose an approach based on pairwise *contracts*.

Medusa's approach is called the *bounded-price mechanism* [3]. Participants negotiate pairwise contracts offline. These contracts set tightly bounded prices for migrating each unit of load and specify the set of tasks that each participant is willing to execute on behalf of its partner. At runtime, a participant moves load to another only if it has a contract with that participant and the cost of processing a task locally is larger than the payment the participant would have to make to its partner for the same processing.

In contrast to other approaches, this mechanism has a low runtime overhead and remains stable under variable load conditions, minimizing load migrations. Participants have tight control over their collaborations, prices, and the tasks they move. Participants can also easily extend Medusa contracts to further customize their load management agreements by adding, for instance, performance and availability requirements.

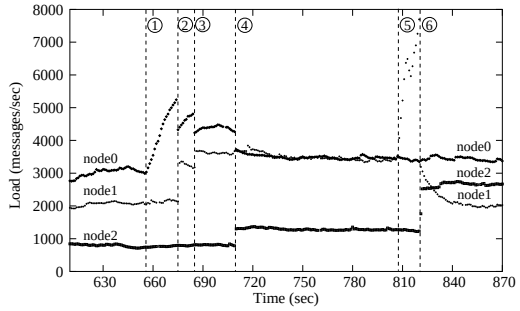

Figure 2: Load at three Medusa nodes running the network intrusion detection query over network connection traces.

Medusa does not distribute load optimally but it guarantees *acceptable allocations*: i.e., either *no* participant operates above its capacity, or, if the system as a whole is overloaded, then *all* participants operate at or above capacity. We argue that autonomous participants value privacy and service customization, as offered by Medusa, more than optimal load balance. An important result of the Medusa mechanism is that only small contract price-ranges are required for the system to always converge to an acceptable allocation. Our demonstration explains how and why this works.

Medusa relies on *remote definitions* to move operators between nodes. Remote definitions specify how operators on different nodes map onto one another. At runtime, when a path of operators needs to be moved, the origin node simply instantiates the operators remotely and diverts the incoming streams to the appropriately named inputs on the new node. As we demonstrate, remote definition is fast and lightweight compared to process migration.

Figure 2 shows the result of running Medusa on three machines, each one executing the query shown in Figure 1. Each machine processes traces of network connections collected at MIT and an ISP in Utah. As load varies ((1) and (5)), nodes exchange load ((2), (3), (4), and (6)) and converge to acceptable allocations. We use a similar setup in the demonstration.

3. HIGH AVAILABILITY

High availability is an important goal for many Medusa applications. A multi-site deployment can improve the availability of a single-site system because sites can monitor and take over for each other [9, 11]. Wide-area queries, however, are also vulnerable to network failures and more importantly to network partitions. Medusa addresses this second problem, which has received little attention in the distributed stream processing community.

Designing a distributed system that is always available, provides consistent access to replicated data, and is resilient to network partitions is known to be impossible [8]. Because stream processing applications tolerate non-deterministic results and value low processing-latency and availability, approaches that favor availability over consistency are more appropriate in this domain. Existing approaches (e.g., [10]), however, are not designed to work-well with the high replica update rate common in stream-processing applications. Furthermore, in stream processing, in addition to reconciling replica state, we also need to correct the information previously sent by each replica to its clients.

To achieve availability in face of network partitions and minimize information loss and inconsistencies at the client, Medusa proposes the following approach. Each replica of a query-network fragment is also a different version of its primary. Versioning allows the system to keep track of all replicas and allows client applications that can hear from multiple partitions to disambiguate the duplicate information.

When a partition heals, the system has multiple copies of identical query-network fragments. Each copy could be in a different state due to the unavailability of different inputs during the partition. To reconcile replica states, the system could wait for the convergence to occur naturally or it could rollback all queries to a state before the partition and reprocess all the data. With the firs approach, clients lose information, while the overhead of the second approach is prohibitive in practice.

To reconcile replica states with less overhead and without information loss, Medusa selects one of the replicas to reprocess input streams. Medusa then compares the new output stream with that produced by other replicas and either rolls-back the output stream alone or produces selected updates. Medusa's network partition handling capabilities leverage stream versioning, time travel and revision tuples, three of the new stream processing features introduced by Borealis, the second generation distributed stream processing engine developed at Brown, Brandeis, and MIT.

To show how Medusa handles network partitions, we disconnect the network cable between two machines running the intrusion detection query and show that the query continues within each partition albeit missing a fraction of the input data. Reconnecting the machines, we show how Medusa reconciles the inconsistencies.

4. REFERENCES

[1] D. J. Abadi, D. Carney, U. Çetintemel, M. Cherniack, C. Convey, S. Lee, M. Stonebraker, N. Tatbul, and S. Zdonik. Aurora: A new model and architecture for data stream management. *VLDB Journal*, 12(2), Sept. 2003.

[2] B. Babcock, S. Babu, M. Datar, R. Motwani, and J. Widom. Models and issues in data stream systems. In *Proc. of the 2002 ACM PODS Conf.*, June 2002.

[3] M. Balazinska, H. Balakrishnan, and M. Stonebraker. Contract-based load management in federated distributed systems. In *Proc. of the First NSDI Symp.*, Mar. 2004.

[4] S. Chandrasekaran, A. Deshpande, M. Franklin, and J. Hellerstein. TelegraphCQ: Continuous dataflow processing for an uncertain world. In *Proc of the CIDR Conf.*, Jan. 2003.

[5] J. Chen, D. J. DeWitt, F. Tian, and Y. Wang. NiagaraCQ: A scalable continuous query system for Internet databases. In *Proc. of the 2000 ACM SIGMOD Conf.*, May 2000.

[6] M. Cherniack, H. Balakrishnan, M. Balazinska, D. Carney, U. Çetintemel, Y. Xing, and S. Zdonik. Scalable distributed stream processing. In *Proc of the CIDR Conf.*, Jan. 2003.

[7] B. N. Chun. *Market-Based Cluster Resource Management*. PhD thesis, University of California at Berkeley, 2001.

[8] S. Gilbert and N. Lynch. Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services. *ACM SIGACT News*, 33(2), 2002.

[9] J.-H. Hwang, M. Balazinska, A. Rasin, U. Çetintemel, M. Stonebraker, and S. Zdonik. A Comparison of Stream-Oriented High-Availability Algorithms. Technical Report CS-03-17, Brown University, Oct. 2003.

[10] K. Petersen, M. Spreitzer, D. Terry, M. Theimer, and A. Demers. Flexible update propagation for weakly consistent replication. In *Proc of the ACM SOSP*, Oct. 1997.

[11] M. Shah, J. Hellerstein, and E. Brewer. Highly-available, fault-tolerant, parallel dataflows. In *Proc. of the 2004 ACM SIGMOD Conf.*, June 2004.

[12] M. Stonebraker, P. M. Aoki, W. Litwin, A. Pfeffer, A. Sah, J. Sidell, C. Staelin, and A. Yu. Mariposa: a wide-area distributed database system. *VLDB Journal*, 5(1), Jan. 1996.