Fault-tolerant, Load-balancing Queries in Telegraph

Mehul A. Shah and Sirish Chandrasekaran
U. C. Berkeley
{mashah, sirish}@cs.berkeley.edu

Users frequently perform tasks with database systems that involve long-running, data-intensive queries. Examples of such queries include: decision support queries, data mining queries [4], and warehouse loads [7]. Queries across wide-area sources with high-latencies and large input sizes are another example that can run for days [6].

With current systems, these long-running queries can be tedious because the longer they run, the more likely they are to encounter faults. Software bugs or human error are sources of failures in the local-area. Unpredictable data arrival rates and source unavailability are sources of volatility in wide-area queries [5]. Frequent faults and restarts can prevent such queries from making any progress.

Administrators often speed up long-running, local-area queries by parallelizing them across a shared nothing cluster. While failures are also a concern in this environment, unexpected load imbalances are even more likely [2]. For example, inaccurate cost estimates may overload a node causing it to thrash. These imbalances are difficult to anticipate and may reduce or even negate the benefits of parallelism.

To address these issues, we built a parallel query processor in Telegraph that handles local failures and gracefully adapts to runtime perturbations in a cluster. In addition, it adapts to volatility from wide-area sources.

1. ARCHITECTURE

To achieve its resilience, Telegraph employs operator redundancy, the *Flux* operator, and Eddies [3]. Telegraph relies on the Flux operator coupled with operator redundancy for fault-tolerance and load-balancing in the local-area. Telegraph mirrors the transient state of operators to tolerate node failures. Flux, a generalized exchange operator, encapsulates logic for dynamic workload partitioning, redundant data shipping, and coordinating state migration to balance load and handle recovery. Telegraph uses an Eddy, a mechanism that continuously reorders operators at runtime, for adapting to wide-area volatility.

2. DEMONSTRATION

We demonstrate fault-tolerant, load-balancing queries in Telegraph. Our hardware platform consists of an array of laptops connected by an Ethernet network. We run a few long-running queries on this cluster and introduce faults and non-uniformities into the execution environment to demonstrate the following:

Tolerance to Node Failure. We disconnect a computing node during a large-scale decision support query. This failure causes Telegraph to rebalance the work among the remaining nodes, and continue the query at a proportionally slower pace. We show that this adaptive phase, in which the query is vulnerable to a second, fatal failure, is short.

Efficient Use of Available Resources. We then reconnect fresh nodes to the network during the same query. The system rebalances the work to make use of additional resources, and the query progresses at its original rate.

Load Balancing. Finally, on one of the compute nodes, we simulate two different types of load: contention for the CPU by adding an additional process, and increased memory pressure by revoking available memory. In each of these cases, the system detects an imbalance and adjusts the workload partitioning to use more resources on the remaining nodes.

Tolerance to Wide-Area Volatility. We run a relational query over simulated web sources. We introduce delays and burstiness in the input sources by adjusting the simulation parameters. Telegraph adapts in two ways: by altering the order in which operators are applied [3], and by processing postponed work in the operators [5].

We conclude our demonstration with a "deep web"[1] crawl to simultaneously illustrate all of the above features. A "deep web" crawl is a transitive closure query that runs over web sources with form-based query interfaces. For example, imagine entering a single street address into the online white pages to find all people that live on that street. One can take the last names from that result set and reenter them into the white pages to find more addresses. After numerous iterations, this procedure will have traversed a significant fraction of the white pages. This algorithm is a transitive closure computation for the starting street address. We can use similar queries to crawl the information stored behind forms in specialized databases on the web, i.e. the "deep web". We execute a "deep web" crawl, simulating the web sources and introducing faults describe above to demonstrate the query engine's resilience.

ACM SIGMOD 2001 May 21-24, Santa Barbara, California, USA

3. REFERENCES

[1] BrightPlanet.com. The Deep Web: Surfacing Hidden Value, White Paper. *http://www.lexibot.com/*.
[2] Remzi Arpaci-Dussea et. al. Cluster I/O with River: Making the Fast Case Common. *IOPADS*, May 1999.
[3] Ron Avnur and Joseph M. Hellerstein. Eddies: Continuously Adaptive Query Processing. *SIGMOD*, 2000.
[4] S. Sarawagi, S. Thomas, and R. Agrawal. Integrating Mining with Relational Database Systems: Alternatives and Implications. In *SIGMOD*, pages 343–354, 1998.
[5] T. Urhan, M. Franklin, and L. Amsaleg. Cost-Based Query Scrambling for Initial Delays. In *SIGMOD*, 1998.
[6] Telegraph Project. Federated Facts and Figures. *http://fff.cs.berkeley.edu*.
[7] Wiburt Labio et. al. Efficient Resumption of Interrupted Warehouse Loads. *SIGMOD*, 2000.