Query processing for datacenter-scale computers

Spyros Blanas

The Ohio State University
blanas.2@osu.edu

Introduction

Quickly exploring massive datasets for insights requires an efficient data processing platform. Parallel database management systems were originally designed to scale only to a handful of nodes, where each node keeps recent ("hot") data in memory and has directly-attached hard disk storage for infrequently accessed ("cold") data. To keep pace with the growing data volumes, the research focus has been shifting towards rack-scale architectures. Rack-scale systems combine powerful nodes with directly-attached storage for "warm" data with hundreds of terabytes of network-attached storage for "cold" data. Exemplars include Oracle Exadata, the IBM PureData System and the Microsoft Analytics Platform System.

Processing even larger datasets quickly will inevitably require datacenter-scale computers. Although details of the hardware configurations of commercial datacenters are scarce, one can use scientific computers for data-intensive applications as a proxy. Compared to rack-scale architectures, the "hot" data path consists of a few petabytes of memory fragmented across ten thousand nodes, and proprietary network interconnects in unique topologies. The "cold" data path retrieves data through a parallel file system (such as Lustre) from many petabytes of network-attached storage.

We posit that the optimizations a DBMS performs individually within a node or across a handful of powerful nodes are insufficient when query processing becomes a datacenter-scale challenge. We argue that the opportunity for database systems research is to lead the transition from the cloud-inspired view of a datacenter as a collection of CPUs towards the view of a datacenter as a single entity that requires careful task orchestration for peak aggregate performance. We identify the following research directions towards realizing this vision.

Research opportunities

Accurately account for the cost of accessing local and distributed memory: Query optimization needs to accurately account for the cost of accessing in-memory data. In particular, cost models need to be sensitive to the in-memory access pattern and to the relative distance of the memory location. Distance has the potential to cleanly capture intra-node and inter-node effects in one metric. Intra-node effects of interest include costs to access data in different levels of the local cache hierarchy, as well as any local NUMA effects. Inter-node properties of interest are the effect of the network topology and link congestion on throughput.

Directly interface with the proprietary low-latency interconnect: Prior work has already shown that a query processing engine that uses TCP/IP for communication does not fully utilize a fast network. High-end computers take network performance one step further through proprietary interconnects that offer lower latency and higher throughput than commodity server networking, such as EDR InfiniBand. Proprietary interconnects can perform scatter/gather operations and synchronization directly in the network. In addition, high-end interconnects have hardware support for global address space programming models, which can offload costly virtual address translation and cache coherence operations to the network. To lever these unique network hardware features for performance, a query processing engine needs to more tightly interface with the network adapter than currently possible with InfiniBand verbs.

Curtail redundant I/O by reconstructing objects from cached data fragments in nearby memory: The limited I/O bandwidth to "cold" storage in a high-end computer prompts us to rethink the role of the buffer pool. The proximity of remote memory through the fast network interconnect necessitates a transition to a distributed buffer pool design, as originally proposed in the context of client/server database architectures. Furthermore, lineage information can transform the buffer pool from a passive responder to I/O requests to an active participant in the query that proactively places intermediate results in memory that is in the proximity of the consuming process.

Predict and ameliorate detrimental I/O interference: Cost models for directly-attached storage are commonly a linear function of the number of fundamental I/O actions, such as disk seeks. The implicit assumption is that if the DBMS eliminates such costly actions all requests will be processed near the peak throughput rate. The parallel file system of a datacenter-scale computer, however, is concurrently used by thousands of jobs. As multiple nodes target the same storage platform, the aggregate I/O pattern becomes more random, leading to highly-variable I/O costs due to interference and queuing. For query optimization to be effective, the DBMS needs to accurately estimate the current I/O interference and then extrapolate to the future. In addition, query evaluation algorithms that operate on "cold" data need to be made robust to I/O actions with highly variable latencies. Finally, the DBMS needs to judiciously consider when the synchronization overhead to coalesce small parallel I/O operations into large sequential requests would be preferable to coordination-free data processing.

Submitted to CIDR'17 January 8–11, 2017, Chaminade, CA, USA

ACM ISBN 978-1-4503-2138-9.
DOI: 10.1145/1235