Course Project Proposal: Cooperative storage management between a DBMS and a storage system

Overview

Declarative Programmable Storage is on the horizon for my own research. As an initial step that incorporates DBMS implementation, I would like to explore the integration of an existing DBMS with an existing storage system. This work would be to take the transactional storage manager in a DBMS and replace it with calls to a minimally modified storage API. The storage system, then, would manage storage of data directly on a block device while the DBMS would logically maintain indexes and metadata, but storage and relevant caching would be handled at a lower level.

Work Estimations and Milestones

Since there are roughly 5 weeks remaining in the course, and some portion (preferably all) of the project should be done before the end of the quarter, I break down the milestones and estimation of effort in this section.

Letter-based Estimate	Ideal deadline	Milestone description
A	11/09	VM on QEMU running the linux kernel with a minimally modified BtrFS
A	11/23	DBMS running in VM with calls directly to btrfs for storage
A	11/30	DBMS B-tree indexes also stored in btrfs storage with co-operative changes to btrfs for data storage
B	12/05	Benchmarks comparing DBMS on top of btrfs with DBMS communicating with btrfs directly

Details

Storage system

The storage system to be used is BtrFS. This is a file system based on the B-tree data structure. The motivation behind using BtrFS is to minimize impedance mismatch between the file system and the DBMS storage structures. Hopefully this will yield benchmark results that will be easier to interpret, and will serve as an important intermediate step before extending to other storage systems.

Data management system

I will use either postgres or Joy Alruraj's port of postgres to C++. As postgres was designed as a disk-based DBMS and was the base (somewhat) of Andy Pavlo's Peloton, it seems like a good DBMS to understand and support. It also seems like it may be more straightforward to work with than other typical options such as MySQL.

Queries and workloads to benchmark

A rough, high-level interest should be to benchmark: 1. write heavy workloads 2. read heavy workloads that span many nodes in the tree (both range and point queries) 3. balanced workloads that only hit hot data just

larger than RAM.

Storage interface contract

I will work on a layer between the DBMS calls to the storage manager and the file system API. Ideally this will satisfy the DBMS needs (the various ways in which it interacts with the storage manager), and make clear what types of operations the file system API does not sufficiently express. Changes to this interface will of course happen, but I would like to understand what the process looks like to get from storage interfaces as-is to a preferred storage interface.