Introduction

We are interested in the possibility of a "smart" storage server, a machine that is aware of the individual hardware storage devices it possesses. A tree data structure is used to internally represent both hardware devices and low level operations of the devices.

Under this scheme, leaf nodes will represent the hardware devices themselves. Intermediate nodes will represent operations that can be performed on the devices. Intermediate node operations can be composed of other operations. For instance, an abstract hashmap "put" operation can be composed of both a call to the "put" operation on a balanced tree and a call to the "set" operation on an array (see Figure 1).

The tree will contain a special root node representing the API of the machine. It exposes a set of operations to be performed on the machine’s hardware storage devices.

Motivations

Flexibility

In the recent years there have been the introduction of several new commodity hardware storage devices. Shingled Magnetic Recording (SMR) is one such example. Given that different storage devices have different strengths and weaknesses, work has been done in designing storage schemes that leverage the strengths of different hardware storage devices (see Soundararajan 2010, "Extending SSD Lifetimes with Disk-Based Write Caches") to yield better performance and longer device lifetimes.
One motivation for our model is its flexibility in supporting arbitrary hardware and functional requirements. A system designer would be able to specify, for instance, that a particular HDD is a write cache for a particular SSD. Successfully implementing this kind of framework would facilitate the implementation of new storage system designs and the modification of existing ones.

An interesting area to explore would be the possibility of a declarative language for expressing tree designs. The programmer declares the set of hardware devices, the set of operations and how they are composed, and the root level API, and the framework translates the declaration into the underlying code.

**Custom API’s**

This model is able to easily represent custom API’s as the set of functions exposed through the root node of the tree. Our model would be able to support arbitrary data storage API’s, e.g. graph database, relationship database, document store, etc. Both the API and its implementation can be tailored specifically for the hardware devices of the underlying machine.
Performance Modeling

The fact that the tree models the hardware and a set of defined operations on the hardware could potentially allow for a novel way of measuring the performance of the system. Furthermore, software could be developed to model the effects of modifying the system (e.g. removing a device, adding a device, implementing operations in different ways). This would offer systems designers valuable insight into the hardware architecture of their system and give them an accurate way of modeling alternative designs.

Automated Discovery of New Designs

If, under this scheme, it is possible to model the performance of a system’s hardware architecture, then the process of modeling alternative designs could be automated. Then, the process of modeling and running simulations on alternative designs would surface better architectures. Ideally, algorithms could be developed such that this process yields optimal designs given a set of constraints and requirements.

Deliverables

My project will focus on exploring the possibility and implementation of custom API’s as well as the generation of code expressing the tree model. The latter entails implementing a synthesizer that generates code for arbitrary operators, assuming that the foundation operators (native hardware level operators) have already been implemented.

Thus, my deliverables will include:

1. Working code to tree synthesizer
2. Implementations of various custom API’s