Title of the Paper

First Last Name*
University of California, Berkeley
November 7, 2009

Abstract Write your abstract here.
Keywords: Time-inconsistency, hyperbolic discounting
JEL Classifications Numbers: D11, D42, L12, L16, L66, L67.

* 508-1 Evans Hall #3880, Berkeley, California 94720-3880. Email: xxx@berkeley.edu

Acknowledgements

Abstract

Contents

List of Figures

No table of figures entries found.

Chapter 1

Introduction

Mini TOC

Chapter 2

Preliminaries

Mini TOC

Chapter 3

Finding good Factorization Trees

Mini TOC

Chapter 4

Serialization of Data Factorizations

Mini TOC

Chapter 5

Distributed Query Processing in FDB

Mini TOC

Chapter 6

Experimental Evaluation

In this section we will present experimental evaluation for the main contributions of this project, namely the *COST* function for finding good f-trees (see Chapter 3), the serialization techniques explained in Chapter 4 and D-FDB, the distributed query engine as presented in Chapter 5.

6.1 Datasets and evaluation setup

This section contains information regarding datasets used and the evaluation setup used to record the reported times and sizes.

6.1.1 Datasets

We used two different datasets throughout the development and evaluation of the above contributions, both described below.

1. *Housing*

   This is a synthetic dataset emulating the textbook example for the house price market.

   It consists of six tables:

   - *House* (postcode, size of living room/kitchen area, price, number of bedrooms, bathrooms, garages and parking lots, etc.)
   - *Shop* (postcode, opening hours, price range, brand, e.g. Costco, Tesco, Sainsbury's)
   - *Institution* (postcode, type of educational institution, e.g., university or school, and number of students)
   - *Restaurant* (postcode, opening hours, and price range)
   - *Demographics* (postcode, average salary, rate of unemployment, criminality, and number of hospitals)
   - *Transport* (postcode, the number of bus lines, train stations, and distance to the city center for the postcode).

   The scale factor s determines the number of generated distinct tuples per postcode in each relation: We generate s tuples in *House* and *Shop*, $log2(s)$ tuples in *Institution*, $s/2$ in *Restaurant*, and one in each of *Demographics* and *Transport*. The experiments that use the *Housing* dataset will examine scale factors ranging from 1 to 15.

2. *US retailer*

   The dataset consists of three relations:

   - *Inventory* (storing information about the inventory units for products in a location, at a given date) (84M tuples)
   - *Sales* (1.5M tuples)
   - *Clearance* (370K tuples)
   - *ProMarbou* (183K tuples)

6.1.2 Evaluation setup

The reported times for the *COST* function and the serialization techniques were taken on a server with the following specifications:

- Intel Core i7-4770, 3.40 GHz, 8MB cache
- 32GB main memory
- Linux Mint 17 Qiana with Linux kernel 3.13

The experiments to evaluate the distributed query engine D-FDB were run on a cluster of 10 machines with the following specifications:

- Intel Xeon E5-2407 v2, 2.40GHZ, 10M cache
- 32GB main memory, 1600MHz
- Ubuntu 14.04.2 LTS with Linux kernel 3.16

6.2 COST function – Finding good f-trees

TODO

6.3 Serialization of Data Factorizations

In this section, we evaluate each serialization technique examined and described in Chapter 4. The factorizations we use to evaluate the serialization techniques are the result of applying *NATURAL JOIN* on all the relational tables of the two datasets, *Housing* and *US retailer*.

6.3.1 Correctness of serialization

The correctness test of each serialization was done both in-memory and off-memory (using secondary storage). For equality comparison between two factorizations we use a special function *toSingletons()* that traverses the factorization, encoding the singletons into a string representation that contains *a)* attribute name, *b)* value and *c)* attribute ID in text format, thus creating a huge string that contains the whole data of the factorization.

For the *in-memory* tests we performed the following steps:

1. Load the factorization from disk, let's call it *OriginRep*
2. Serialize it in memory writing into a memory buffer (array of bytes)
3. Deserialize the buffer into a new instance of a factorization, let's call it *SerialRep*

4. Check that the fields of *SerialRep* have valid values
5. Use the *toSingletons()* method and create the string representation for *OriginRep* and *SerialRep* and compare the two strings for equality. This ensures that not only we recover the same number of singletons properly but also that the IDs and values of those singletons are preserved during serialization and de-serialization, even with problematic datatypes like floating point values.

For the *off-memory* tests we performed similar steps as in-memory with an extra additional test to further prove correction.

1. Load the factorization from disk, let's call it *OriginRep*
2. Serialize it to a file on disk (binary file mode)
3. Open the file in read mode and de-serialize it into a new instance of a factorization, let's call it *SerialRep*
4. Check that the fields of *SerialRep* have valid values
5. Use the *toSingletons()* method and create the string representation for *OriginRep* and *SerialRep* and compare the two strings for equality.
6. Enumerate the tuples encoded by the factorizations *OriginRep* and *SerialRep* into two files. Compare the two files for equality using the standard command line tool *diff*.

Chapter 7

Conclusions and Future Work

Mini TOC

A player faces a dynamic optimization problem of 5 periods. Let a_t denotes the player's action in period t,

$$a_t \in \{P, N\} \tag{1}$$

We denote the vector of action choices by $\boldsymbol{a} = (a_1, a_2, a_3)$. Playing in a period yields an immediately consumption level of x at a certain future cost, to be paid at period 4, while not playing yields no consumption and incurs no cost, so

$$x_t = \begin{cases} x & if\ a_t = P \\ 0 & if\ a_t = N \end{cases} \tag{2}$$

The player observe x in period 1 before she pick her action.
Let C_s denotes total cost for playing s games and S_t the number of games played up till and including time t.

This paper.[1] Theoretically, ...
The issue of ...
This paper is organized as follows. The next section presents ... Then, Section 3 discusses

[1] Ashraf et. al [1] uses a ...

the ... Section 4 analyzes the ... Concluding remarks are offered in Section 5.

References

Ashraf, Nava, Dean Karlan and Wesley Yin. "Tying Odysseus to the Mast: Evidence from a Commitment Savings Product in the Philippines." Quarterly Journal of Economics. Vol. 121, No. 2, pp. 635-672. May 2006.