

DSAAS

DSAAS

A Cloud Service for Persistent Data Structures

Pierre le Roux, Steve Kroon and Willem Bester

Sunday 24th April, 2016

Stellenbosch University

<http://cs.sun.ac.za/~kroon/dsaas>

PROBLEM

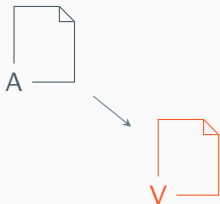
Alice's structured data sets



Bob's structured data sets

PROBLEM

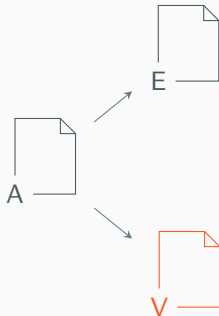
Alice's structured data sets



Bob's structured data sets

PROBLEM

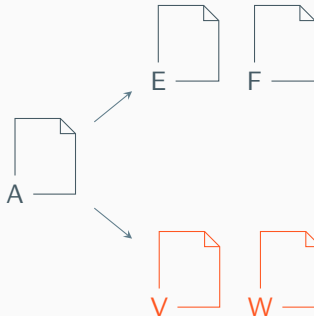
Alice's structured data sets



Bob's structured data sets

PROBLEM

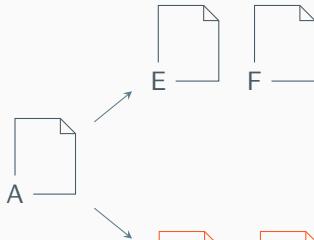
Alice's structured data sets



Bob's structured data sets

PROBLEM

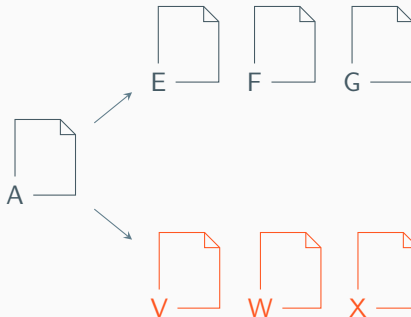
Alice's structured data sets



Bob's structured data sets

PROBLEM

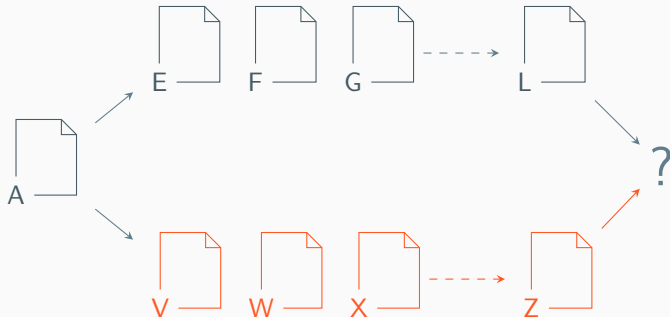
Alice's structured data sets



Bob's structured data sets

PROBLEM

Alice's structured data sets



Bob's structured data sets

Collaboration on structured data can be difficult, time-consuming, error-prone and frustrating.

OVERVIEW

A prototype *cloud* service for using automatically version controlled data structures.

$(\text{version}, \text{key}) \Rightarrow \text{value}$

OVERVIEW

A prototype *cloud* service for using automatically version controlled data structures.

$(\text{version}, \text{key}) \Rightarrow \text{value}$

Layers of Service

Language Bindings
API
Data Structures
Version Control
Data storage

EXAMPLE

```
http://dsaas.pbit.co.za/workbench/graph/pierre/  
SimpleFriendsGraph/
```

BACKGROUND



Ephemeral vs Persistent Data Structures

Ephemeral vs Persistent Data Structures

Types of persistence:

VERSION CONTROL

Ephemeral vs Persistent Data Structures

Types of persistence:

Partial Persistence



VERSION CONTROL

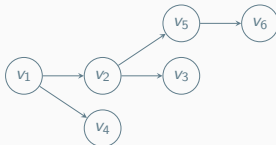
Ephemeral vs Persistent Data Structures

Types of persistence:

Partial Persistence



Full Persistence



VERSION CONTROL

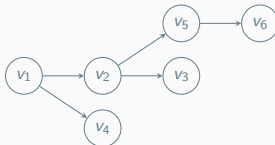
Ephemeral vs Persistent Data Structures

Types of persistence:

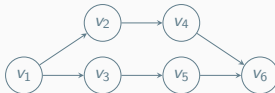
Partial Persistence



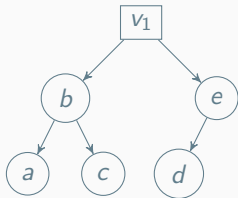
Full Persistence



Confluent Persistence

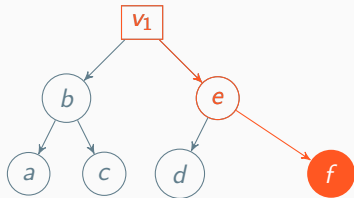


Achieve full persistence using a technique called *path-copying*



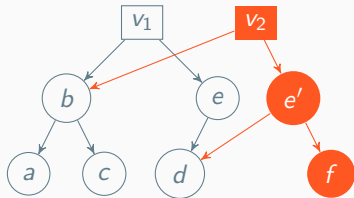
PATH-COPYING

Achieve full persistence using a technique called *path-copying*



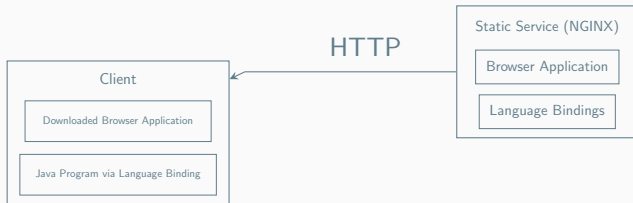
PATH-COPYING

Achieve full persistence using a technique called *path-copying*

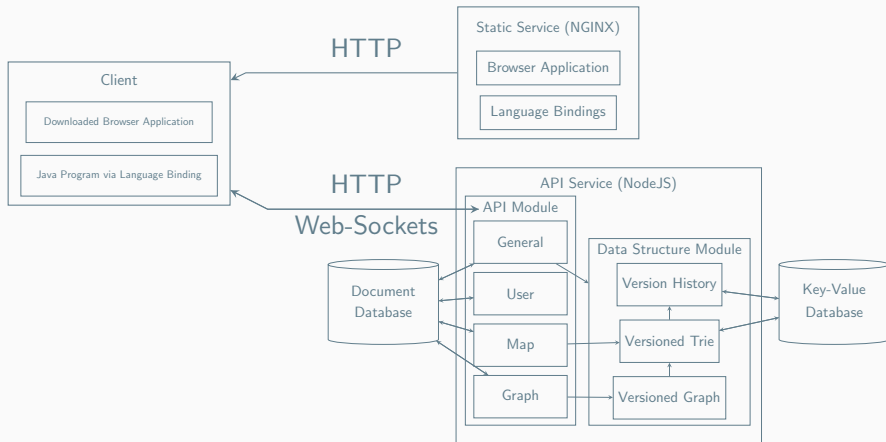


DEVELOPMENT

ARCHITECTURE OVERVIEW



ARCHITECTURE OVERVIEW



BROWSER APPLICATION

DSaaS Pierre Be...

Workbench

Download Java Binding Tutorial

Filter

Graphs Maps Both

Add Data Structure + Access Tokens



pierre

MyMap map	Fork	Permissions	Export	Info	Remove
BST graph	Fork	Permissions	Export	Info	Remove
BST2 graph	Fork	Permissions	Export	Info	Remove
BST3 graph	Fork	Permissions	Export	Info	Data structures received

DataStructures As A Service

DSaaS Pierre Be...

Version History Property Viewer Graph Editor pierre/SimpleFriendsGraph/2539a..45a8e



[Merge Track]

DSaaS Pierre Be...

Version History pierre/exp1/ac58..a0f7



[[Merge Track]]

Add Key-Value Pair + Filter Keys

CSAHR-10395-1-1-1140209

payee: STANDARD LIFE
fundingSourceDescription: Payroll Clearing Fund
city: PORTLAND
amount: 56113.71

GSFPM-55652-1-1-1140197

payee: ALL METRO DOOR & DOCK SERVICES LLC
fundingSourceDescription: General Fund-Operations
city: DENVER
amount: 1719.51

EAPPS-21542-1-14-1140219

DSaaS Pierre Be...

Editing nodes in pierre / SimpleFriendsGraph / 2539a11ef6ae3539c8376955c90745693bd2df726bd2d205e

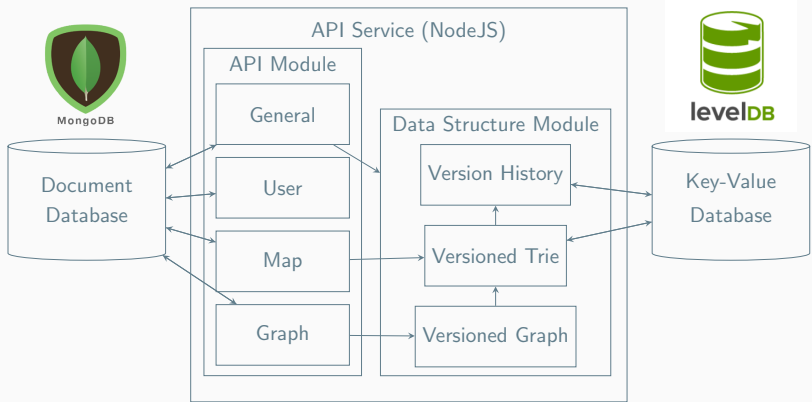
pierre

< Back Save Remove

name	pierre	-
number	0123	-

+ -

BACK END



VERSIONED TRIE

- Based on the Hash Array Mapped Trie (HAMT).
- Implemented on storage instead of in memory.
- Three-way merge operation for confluent persistence.
- Detects transpositions using Zobrist hashing.

HASH ARRAY MAPPED TRIE (HAMT)



An example of an HAMT. The white cells represent the array of references to key-value pairs stored in this trie.

HASH ARRAY MAPPED TRIE (HAMT)



$$h(k_1) = 00000 \dots$$

An example of an HAMT. The white cells represent the array of references to key-value pairs stored in this trie.

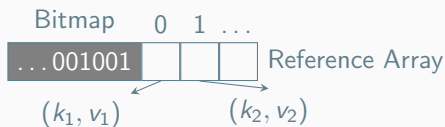
HASH ARRAY MAPPED TRIE (HAMT)



$$h(k_1) = 00000\dots$$

An example of an HAMT. The white cells represent the array of references to key-value pairs stored in this trie.

HASH ARRAY MAPPED TRIE (HAMT)

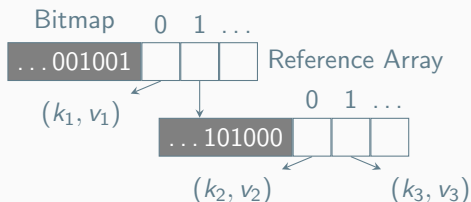


$$h(k_1) = 00000 \dots$$

$$h(k_2) = 00011 \ 00011 \dots$$

An example of an HAMT. The white cells represent the array of references to key-value pairs stored in this trie.

HASH ARRAY MAPPED TRIE (HAMT)



$$h(k_1) = 00000 \dots$$

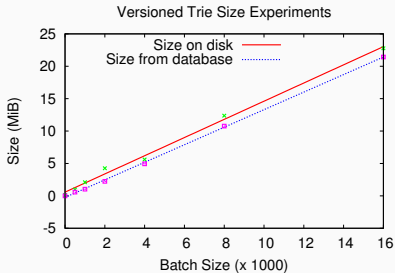
$$h(k_2) = 00011 \ 00011 \dots$$

$$h(k_3) = 00011 \ 00101 \dots$$

An example of an HAMT. The white cells represent the array of references to key-value pairs stored in this trie.

EVALUATION

EVALUATION



Insertion: 1 \approx adding 12
ephemeral data items

Removal: 1 \approx adding 10
ephemeral data items

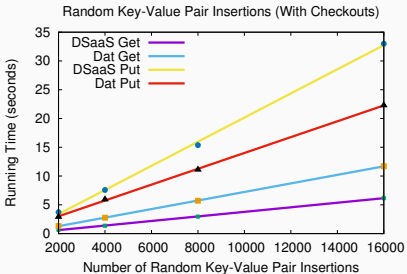
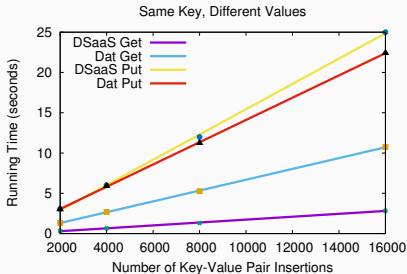
Merging: 16 000 \times 16 000
elements \approx increase
of 650 KiB (\approx 6000
ephemeral items)

LATENCY

Remote Server (Library Binding)	206
Localhost (Library Binding)	7.6
Core (JavaScript)	2

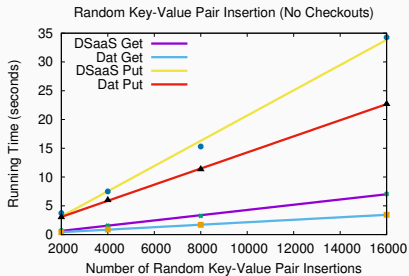
The latency (in ms) for the *put* operation using the library binding to connect to a remote server and the localhost, and using JavaScript to test it on the core system.

COMPARED TO DAT



Dat is available at <http://dat-data.com/>

COMPARED TO DAT



Dat is available at <http://dat-data.com/>

CONCLUSION

Questions?

