DSAAS

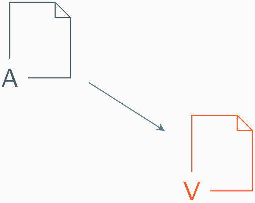
PROBLEM

Alice's structured data sets

Bob's structured data sets

PROBLEM

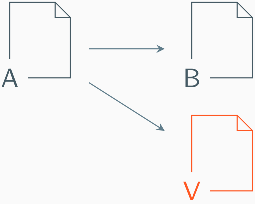
Alice's structured data sets



Bob's structured data sets

PROBLEM

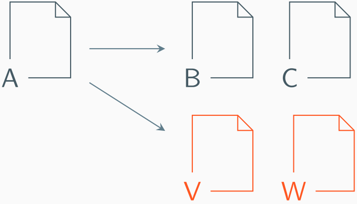
Alice’s structured data sets



Bob’s structured data sets

PROBLEM

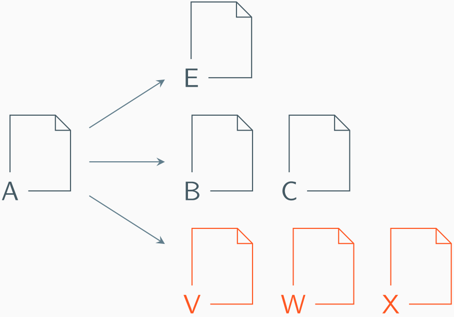
Alice's structured data sets



Bob's structured data sets

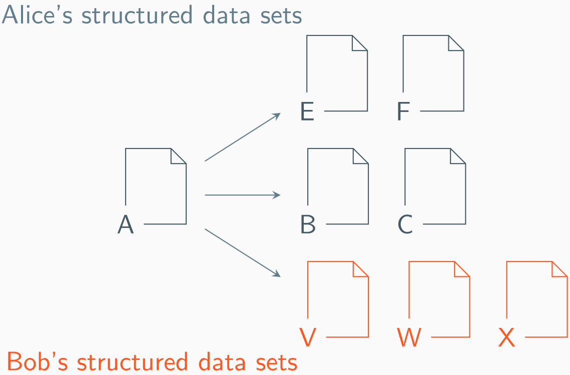
PROBLEM

Alice's structured data sets



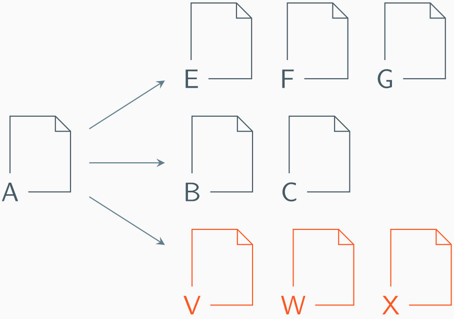
Bob's structured data sets

PROBLEM


Alice's structured data sets
E
F
A
B
C
V
W
X
Bob's structured data sets

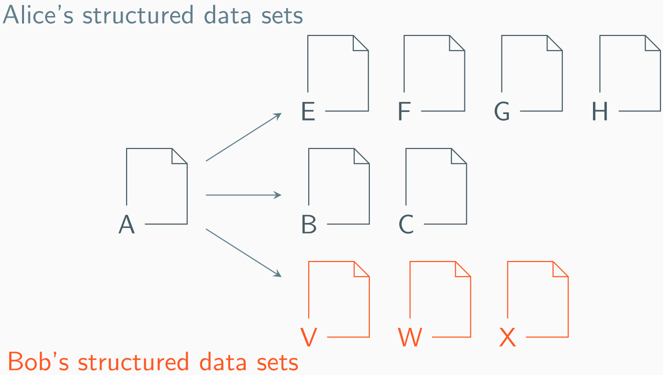
PROBLEM

Alice's structured data sets

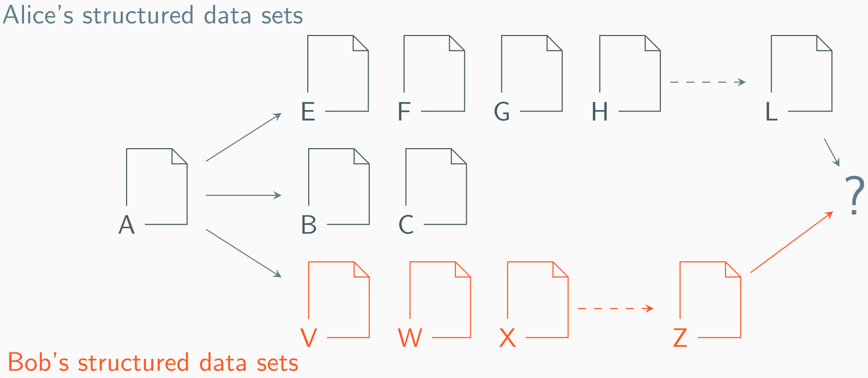


Bob's structured data sets

PROBLEM


Alice's structured data sets
E
F
G
H
A
B
C
V
W
X
Bob's structured data sets

PROBLEM


Alice's structured data sets
E
F
G
H
L
A
B
C
?
V
W
X
Z
Bob's structured data sets

PROBLEM

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

DSAAS

A Cloud Service for Persistent Data Structures

P. B. le Roux, S. Kroon and W. Bester

April 6, 2016

Stellenbosch University

OTHER USE CASES

- Scientific Reproducibility

OTHER USE CASES

- Scientific Reproducibility
- Debugging and Teaching Programming

OTHER USE CASES

- Scientific Reproducibility
- Debugging and Teaching Programming
- Session-based Interpreters

BACKGROUND

VERSION CONTROL

Ephemeral vs Persistent Data Structures

VERSION CONTROL

Ephemeral vs Persistent Data Structures

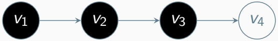
Types of persistence:

VERSION CONTROL

Ephemeral vs Persistent Data Structures

Types of persistence:

Partial Persistence

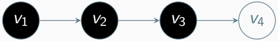


VERSION CONTROL

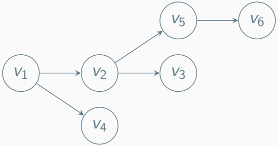
Ephemeral vs Persistent Data Structures

Types of persistence:

Partial Persistence



Full Persistence

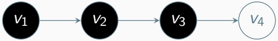


VERSION CONTROL

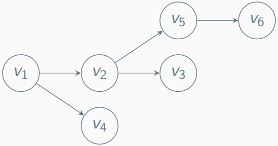
Ephemeral vs Persistent Data Structures

Types of persistence:

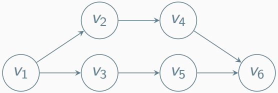
Partial Persistence



Full Persistence

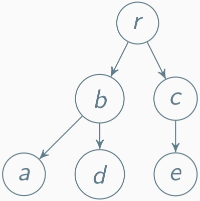


Confluent Persistence



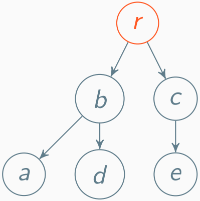
PATH-COPYING

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



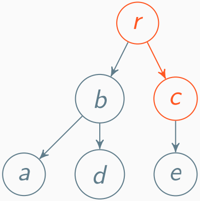
PATH-COPYING

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



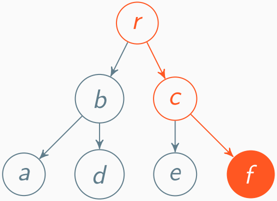
PATH-COPYING

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



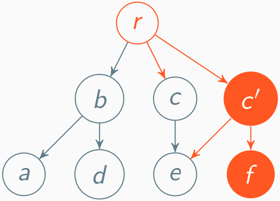
PATH-COPYING

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



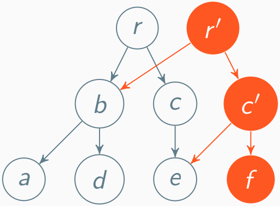
PATH-COPYING

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

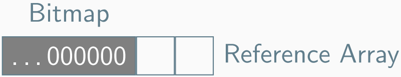


PATH-COPYING

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



HASHED ARRAY MAPPED TRIE (HAMT)



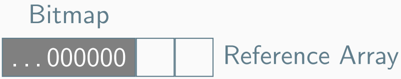
Bitmap

. . . 000000

Reference Array

Figure 1: An example of an HAMT. The grey blocks represent the bitmaps, and the white cells represent the array of references to key–value pairs stored in this trie.

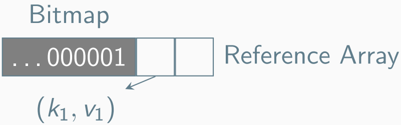
HASHED ARRAY MAPPED TRIE (HAMT)



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

Figure 1: An example of an HAMT. The grey blocks represent the bitmaps, and the white cells represent the array of references to key–value pairs stored in this trie.

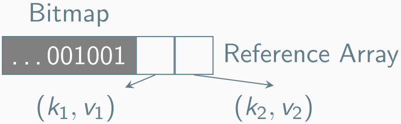
HASHED ARRAY MAPPED TRIE (HAMT)



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

Figure 1: An example of an HAMT. The grey blocks represent the bitmaps, and the white cells represent the array of references to key–value pairs stored in this trie.

HASHED ARRAY MAPPED TRIE (HAMT)

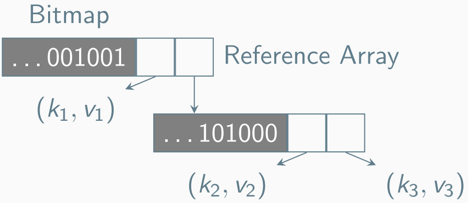


$$h(k_1) = 00000\ldots$$

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

Figure 1: An example of an HAMT. The grey blocks represent the bitmaps, and the white cells represent the array of references to key–value pairs stored in this trie.

HASHED ARRAY MAPPED TRIE (HAMT)

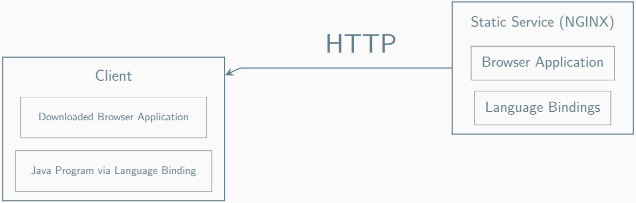


$$h(k_1) = 00000\ldots$$
$$h(k_2) = 00011\ 00011\ldots$$
$$h(k_3) = 00011\ 00101\ldots$$

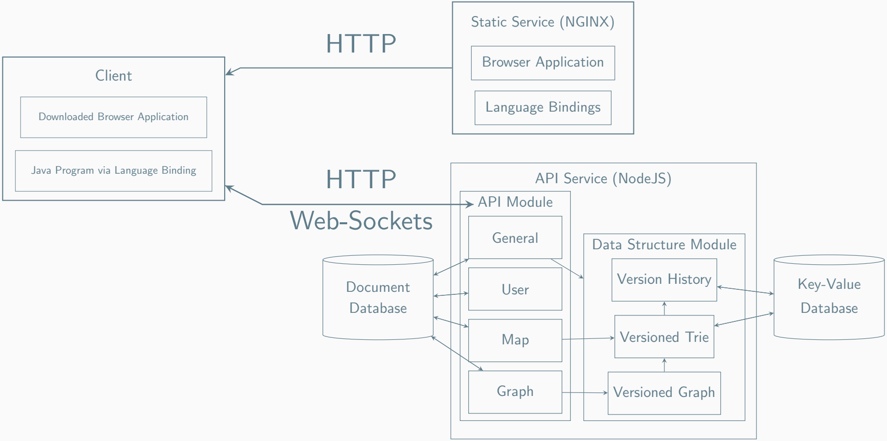
Figure 1: An example of an HAMT. The grey blocks represent the bitmaps, and the white cells represent the array of references to key–value pairs stored in this trie.

DEVELOPMENT

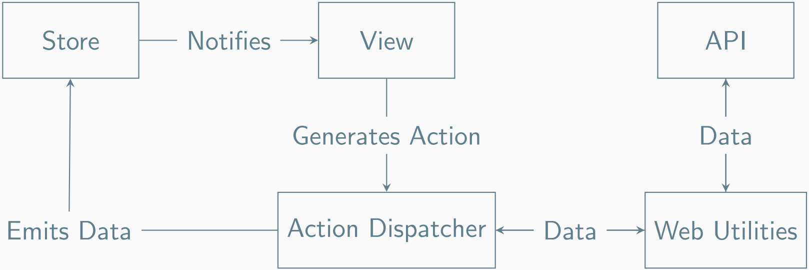
ARCHITECTURE OVERVIEW



ARCHITECTURE OVERVIEW



ADAPTED FLUX ARCHITECTURE



Store

Notifies

View

API

Generates Action

Data

Emits Data

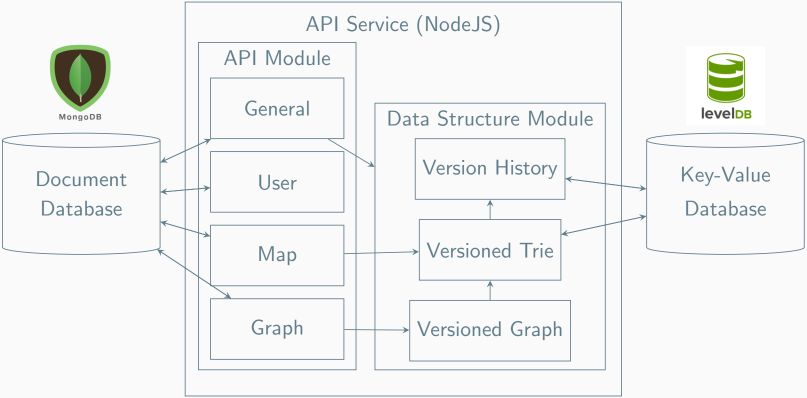
Action Dispatcher

Data

Web Utilities

LANGUAGE BINDINGS

BACK END

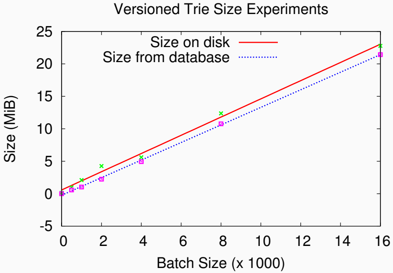

MongoDB
Document Database
API Service (NodeJS)
API Module
General
User
Map
Graph
Data Structure Module
Version History
Versioned Trie
Versioned Graph
levelDB
Key-Value Database

VERSIONED TRIE

- Based on the Hashed Array Mapped Trie (HAMT).
- Implemented on storage instead of in memory.
- Three-Way Merge operation for confluent persistence.
- Detecting transpositions through Zobrist hashing.

EVALUATION

EVALUATION



Insertion: 1 = adding 12 ephemeral data items

Removal: 1 = adding 10 ephemeral data items

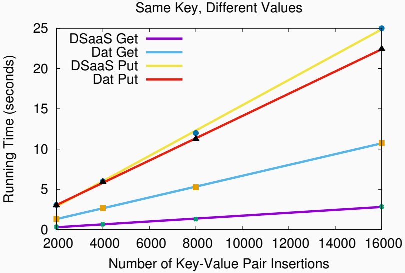
Merging: 16 000 x 16 000 elements = increase of 650 KiB 6000 ephemeral items

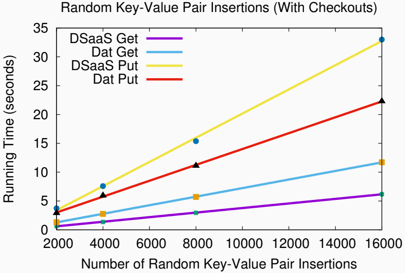
LATENCY

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

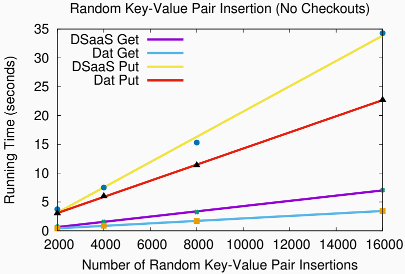
Table 1: The latency (in ms) for the *put* operation using the library binding to connect to a remote server and the localhost, and using JavaScript to test it on the core system.

COMPARED TO DAT


Same Key, Different Values


Random Key-Value Pair Insertions (With Checkouts)

COMPARED TO DAT



CONCLUSION

Questions?