Designing A Low-Latency Cuckoo Hash Table for Write-Intensive Workloads Using RDMA

Tyler Szepesi, Bernard Wong, Ben Cassell, Tim Brecht

Cheriton School of Computer Science
University of Waterloo

April 13, 2014
In-Memory Distributed Systems
TCP Data Transfer
RDMA Data Transfer
Current State of the Art: Pilaf/FaRM
Current State of the Art: Pilaf/FaRM

Client

Server

Application

Shared Data

Client

Client

Client
One-Sided RDMA for Reads and Writes

Client

Server

Application

Shared Data

Client

Client

Client
Nessie

- Cuckoo hash table where clients manage both reads and writes

- Design:
  - One-sided RDMA for reads and writes of key-value pairs
  - Checksum to detect corrupted reads
  - RDMA compare-and-swap to create atomic writes
  - Lock-free to eliminate the possibility of deadlock

- Advantage: Full benefits from one-sided RDMA

- Tradeoffs compared to the state of the art:
  - More roundtrips per operation
  - Added complexity
Cuckoo Hash Table

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td></td>
</tr>
</tbody>
</table>
Cuckoo Hash Table

Write
Key: Dan
Value: Bacon

Dan
Primary Hash
3
Secondary Hash
6

Key | Value
---|---
1  |  
2  |  
3  |  
4  |  
5  |  
6  |  
7  |  
8  |  
9  |  
10 |  
11 |  
12 |  
13 |  
14 |  

Cuckoo Hash Table

Write
Key: Dan
Value: Bacon

Primary Hash

Secondary Hash

Dan

Key Value
1
2
3
4
5
6
7
8
9
10
11
12
13
14

Dan
Bacon

3
6
### Cuckoo Hash Table

**Write**
- **Key:** Ben
- **Value:** Apple

#### Primary Hash
- Ben

#### Secondary Hash
- 10
- 13

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dan</td>
<td>Bacon</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>10</td>
<td>11</td>
</tr>
<tr>
<td>12</td>
<td>13</td>
</tr>
<tr>
<td>14</td>
<td></td>
</tr>
</tbody>
</table>

12
Cuckoo Hash Table

**Write**
- Key: Ben
- Value: Apple

```
Ben
Primary Hash
```

```
Ben
Secondary Hash
```

```
Key | Value
--- | ---
1   | 
2   | 
3   | Dan
4   | Bacon
5   | 
6   | 
7   | 
8   | 
9   | 
10  | Ben
11  | Apple
12  | 
13  | 
14  | 
```
Cuckoo Hash Table

**Write**

Key: Tim
Value: Eggs

```
Key Value
1
2
3
4
5
6
7
8
9
10
11
12
13
14
Dan Bacon
Ben Apple
Tim
Primary Hash
Secondary Hash
3
10
3
```

![Diagram of Cuckoo Hash Table with keys and values mapped through primary and secondary hash functions.](Diagram.png)
Cuckoo Hash Table

Write
Key: Tim
Value: Eggs

Primary Hash
Tim
Secondary Hash

Key | Value
---|---
1  | 2  
3  | 4  
5  | 6  
7  | 8  
9  | 10 |
11 | 12 |
13 | 14 |
### Cuckoo Hash Table

**Write**

- **Key:** Tim
- **Value:** Eggs

#### Table:

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tim</td>
<td>Eggs</td>
</tr>
<tr>
<td>Dan</td>
<td>Bacon</td>
</tr>
<tr>
<td>Ben</td>
<td>Apple</td>
</tr>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td></td>
</tr>
</tbody>
</table>

#### Primary Hash

- **Tim**

#### Secondary Hash

- **Tim**
- **10**
<table>
<thead>
<tr>
<th>Index Table</th>
<th>Data Table</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Nessie
# Nessie

## Index Table

<table>
<thead>
<tr>
<th>Index</th>
<th>V#</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

## Data Table

<p>| |</p>
<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Index Table</td>
</tr>
<tr>
<td>-------------</td>
</tr>
<tr>
<td>Index</td>
</tr>
<tr>
<td>V#</td>
</tr>
<tr>
<td>N Bytes</td>
</tr>
<tr>
<td>64 Bits</td>
</tr>
</tbody>
</table>

- Index Table:
  - 64 Bits

- Data Table:
  - 1
  - 2
  - 3
  - 4
  - 5
  - 6
  - 7
  - 8
  - 9
  - 10
  - 11
  - 12
  - 13
  - 14

- N Bytes
**Nessie: Write**

**Write**
- Key: Tim
- Value: Eggs

<table>
<thead>
<tr>
<th>Index Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
</tr>
<tr>
<td>6</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Data Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key</td>
</tr>
<tr>
<td>------</td>
</tr>
<tr>
<td>Sam</td>
</tr>
<tr>
<td>Tim</td>
</tr>
<tr>
<td>Dan</td>
</tr>
<tr>
<td>Ben</td>
</tr>
</tbody>
</table>

Key: Tim
Value: Eggs
**Nessie: Write**

**Write Data**

**Write**

Key: Tim
Value: Eggs

**Index Table**

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
<th>Valid</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tim</td>
<td>Carrot</td>
<td>T</td>
</tr>
<tr>
<td>Dan</td>
<td>Bacon</td>
<td>T</td>
</tr>
<tr>
<td>Tim</td>
<td>Eggs</td>
<td>F</td>
</tr>
<tr>
<td>Ben</td>
<td>Apple</td>
<td>T</td>
</tr>
</tbody>
</table>

**Data Table**

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
<th>Valid</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sam</td>
<td>Pizza</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6</td>
</tr>
<tr>
<td></td>
<td></td>
<td>7</td>
</tr>
<tr>
<td></td>
<td></td>
<td>8</td>
</tr>
<tr>
<td></td>
<td></td>
<td>9</td>
</tr>
<tr>
<td></td>
<td></td>
<td>10</td>
</tr>
<tr>
<td></td>
<td></td>
<td>11</td>
</tr>
<tr>
<td></td>
<td></td>
<td>12</td>
</tr>
<tr>
<td></td>
<td></td>
<td>13</td>
</tr>
<tr>
<td></td>
<td></td>
<td>14</td>
</tr>
</tbody>
</table>
Nessie: Write

**Read Index Table**

**Write**
- Key: Tim
- Value: Eggs

**Index Table**
- 10
- 6
- 1
- 5

**Data Table**
- Sam | Pizza | T  | 1
- 2
- 3
- 4
- 5
- Tim | Carrot | T | 6
- Dan | Bacon | T | 7
- Tim | Eggs  | F | 9
- Ben | Apple | T | 10
- 11
- 12
- 13
- 14
Nessie: Write

Read Data at Primary

Write
Key: Tim
Value: Eggs

Index Table

Data Table
Key | Value | Valid
--- | --- | ---
Sam | Pizza | T
1
2
3
4
5
Tim | Carrot | T
6
7
Tim | Eggs | F
8
9
Ben | Apple | T
10
11
12
13
14

Tim Primary Secondary

Key Value Valid
**Nessie: Write**

**Read Data at Secondary**

**Write**

Key: Tim
Value: Eggs

**Index Table**

<table>
<thead>
<tr>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>5</td>
</tr>
</tbody>
</table>

**Data Table**

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
<th>Valid</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sam</td>
<td>Pizza</td>
<td>T</td>
</tr>
<tr>
<td>Dan</td>
<td>Bacon</td>
<td>T</td>
</tr>
<tr>
<td>Tim</td>
<td>Eggs</td>
<td>F</td>
</tr>
<tr>
<td>Ben</td>
<td>Apple</td>
<td>T</td>
</tr>
</tbody>
</table>

Tim

Primary

Secondary

Read Data at Secondary
Nessie: Write

Update Index Table

Write
Key: Tim
Value: Eggs

Index Table

Data Table
Key | Value | Valid
--- | --- | ---
Sam | Pizza | T
Tim | Carrot | T
Dan | Bacon | T
Tim | Eggs | F
Ben | Apple | T

Tim
Primary
Secondary
Nessie: Write

**Make Data Valid**

**Write**

Key: Tim
Value: Eggs

**Index Table**

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
<th>Valid</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sam</td>
<td>Pizza</td>
<td>T</td>
</tr>
<tr>
<td>Dan</td>
<td>Bacon</td>
<td>T</td>
</tr>
<tr>
<td>Tim</td>
<td>Eggs</td>
<td>X</td>
</tr>
<tr>
<td>Ben</td>
<td>Apple</td>
<td>T</td>
</tr>
</tbody>
</table>

**Data Table**

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
<th>Valid</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tim</td>
<td>Carrot</td>
<td>T</td>
</tr>
<tr>
<td>Dan</td>
<td>Bacon</td>
<td>T</td>
</tr>
<tr>
<td>Tim</td>
<td>Eggs</td>
<td>X</td>
</tr>
<tr>
<td>Ben</td>
<td>Apple</td>
<td>T</td>
</tr>
</tbody>
</table>

Tim

Primary

Secondary
Nessie: Write with Conflict

Client 1 | Nessie | Client 2
---|---|---

Index Table

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
<th>Valid</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sam</td>
<td>Pizza</td>
<td>T</td>
</tr>
<tr>
<td>Tim</td>
<td>Carrot</td>
<td>T</td>
</tr>
<tr>
<td>Dan</td>
<td>Bacon</td>
<td>T</td>
</tr>
<tr>
<td>Ben</td>
<td>Apple</td>
<td>T</td>
</tr>
</tbody>
</table>

Data Table

Key: Tim  Value: Eggs
### Data Table

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
<th>Valid</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sam</td>
<td>Pizza</td>
<td>T</td>
</tr>
<tr>
<td>Tim</td>
<td>Carrot</td>
<td>T</td>
</tr>
<tr>
<td>Dan</td>
<td>Bacon</td>
<td>T</td>
</tr>
<tr>
<td>Tim</td>
<td>Eggs</td>
<td>F</td>
</tr>
<tr>
<td>Ben</td>
<td>Apple</td>
<td>T</td>
</tr>
</tbody>
</table>

### Index Table

<table>
<thead>
<tr>
<th>10</th>
<th>6</th>
<th>1</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tim</td>
<td>Carrot</td>
<td>Eggs</td>
<td></td>
</tr>
</tbody>
</table>

### Diagram

- Client 1 → Nessie → Client 2
- Write Data

- Index Table
- Data Table

- Key: Tim
- Value: Eggs

---

[31]
Nessie: Write with Conflict

Data Table

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
<th>Valid</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sam</td>
<td>Pizza</td>
<td>T</td>
</tr>
<tr>
<td>Tim</td>
<td>Carrot</td>
<td>T</td>
</tr>
<tr>
<td>Dan</td>
<td>Bacon</td>
<td>T</td>
</tr>
<tr>
<td>Tim</td>
<td>Eggs</td>
<td>F</td>
</tr>
<tr>
<td>Ben</td>
<td>Apple</td>
<td>T</td>
</tr>
</tbody>
</table>

Index Table

- Key: Tim
  - Value: Eggs

Write Data

Client 1  Nessie  Client 2

Read Indices
### Data Table

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
<th>Valid</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sam</td>
<td>Pizza</td>
<td>T</td>
</tr>
<tr>
<td>Tim</td>
<td>Carrot</td>
<td>T</td>
</tr>
<tr>
<td>Dan</td>
<td>Bacon</td>
<td>T</td>
</tr>
<tr>
<td>Tim</td>
<td>Eggs</td>
<td>F</td>
</tr>
<tr>
<td>Ben</td>
<td>Apple</td>
<td>T</td>
</tr>
</tbody>
</table>

### Index Table

<table>
<thead>
<tr>
<th>Key Value Valid</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
</tr>
<tr>
<td>6</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>5</td>
</tr>
</tbody>
</table>

### Write Data

- Key: Tim
- Value: Eggs

### Nessie: Write with Conflict

Client 1: Write Data
Client 2: Read Indices
Nessie: Read Primary
Client 2: Read Indices
Nessie: Write with Conflict

Client 1  Nessie  Client 2

Write Data  
Read Indices  
Read Primary  
Read Secondary

Index Table

Data Table

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
<th>Valid</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sam</td>
<td>Pizza</td>
<td>T</td>
</tr>
<tr>
<td>Tim</td>
<td>Carrot</td>
<td>T</td>
</tr>
<tr>
<td>Dan</td>
<td>Bacon</td>
<td>T</td>
</tr>
<tr>
<td>Tim</td>
<td>Eggs</td>
<td>F</td>
</tr>
<tr>
<td>Ben</td>
<td>Apple</td>
<td>T</td>
</tr>
</tbody>
</table>

Key Value Valid
10  6  1  5

Client 1  Client 2  Nessie
Write Data
Read Indices
Read Primary
Read Secondary
## Nessie: Write with Conflict

### Index Table

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
<th>Valid</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sam</td>
<td>Pizza</td>
<td>T</td>
</tr>
<tr>
<td>Dan</td>
<td>Bacon</td>
<td>T</td>
</tr>
<tr>
<td>Tim</td>
<td>Carrot</td>
<td>T</td>
</tr>
<tr>
<td>Tim</td>
<td>Eggs</td>
<td>F</td>
</tr>
<tr>
<td>Ben</td>
<td>Apple</td>
<td>T</td>
</tr>
</tbody>
</table>

### Data Table

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
<th>Valid</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Diagram

- **Write Data**
- **Read Indices**
- **Read Primary**
- **Read Secondary**
- **Delete Tim**

**Arrows**:
- Client 1 → Nessie
- Nessie → Client 2
- Delete Tim (arrows indicating deletion)

**Key Values**:
- Tim: Eggs
- Client 1: 1
- Client 2: 2
Nessie: Write with Conflict

Data Table

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
<th>Valid</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sam</td>
<td>Pizza</td>
<td>T</td>
</tr>
<tr>
<td>Ken</td>
<td>Cake</td>
<td>T</td>
</tr>
<tr>
<td>Tim</td>
<td>Carrot</td>
<td>T</td>
</tr>
<tr>
<td>Dan</td>
<td>Bacon</td>
<td>T</td>
</tr>
<tr>
<td>Tim</td>
<td>Eggs</td>
<td>F</td>
</tr>
<tr>
<td>Ben</td>
<td>Apple</td>
<td>T</td>
</tr>
</tbody>
</table>

Index Table

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
<th>Valid</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Tim</td>
<td>F</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Client 1 Data

Write Data
Read Indices
Read Primary
Read Secondary
Delete Tim
Write Ken
**Nessie: Write with Conflict**

### Diagram

- **Client 1**
  - Write Data
  - Read Indices
  - Read Primary
  - Read Secondary
  - Write Index

- **Nessie**
  - Write Ken
  - Delete Tim

- **Client 2**
  - Write Ken
  - Write Data

### Data Table

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
<th>Valid</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sam</td>
<td>Pizza</td>
<td>T</td>
</tr>
<tr>
<td>Ken</td>
<td>Cake</td>
<td>T</td>
</tr>
<tr>
<td>Tim</td>
<td>Carrot</td>
<td>T</td>
</tr>
<tr>
<td>Dan</td>
<td>Bacon</td>
<td>T</td>
</tr>
<tr>
<td>Tim</td>
<td>Eggs</td>
<td>F</td>
</tr>
<tr>
<td>Ben</td>
<td>Apple</td>
<td>T</td>
</tr>
</tbody>
</table>

### Index Table

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

**Notes:**
- The diagram illustrates the process of adding entries to a data table and the potential for conflict between simultaneous updates.
- The key-value pairs are updated by clients, and the validity of each entry is indicated.
- The conflict occurs when two clients attempt to update the same key simultaneously.

---

37
Nessie: Write with Conflict

Client 1 | Nessie | Client 2
---------|--------|---------
Write Data
Read Indices
Read Primary
Read Secondary
Delete Tim
Write Ken
Write Index
Make Valid

Data Table

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
<th>Valid</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sam</td>
<td>Pizza</td>
<td>T</td>
</tr>
<tr>
<td>Ken</td>
<td>Cake</td>
<td>T</td>
</tr>
<tr>
<td>Tim</td>
<td>Carrot</td>
<td>T</td>
</tr>
<tr>
<td>Dan</td>
<td>Bacon</td>
<td>T</td>
</tr>
<tr>
<td>Tim</td>
<td>Eggs</td>
<td>F</td>
</tr>
<tr>
<td>Ben</td>
<td>Apple</td>
<td>T</td>
</tr>
</tbody>
</table>

Index Table

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
<th>Valid</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

38
Problem:
Any client can modify the index table part way through a write operation.
**Problem:**
Any client can modify the index table part way through a write operation.

**Solution:**
1) Abort the operation as soon as a conflict is detected.
2) Retry the operation.
Nessie: Handling Conflicts

Data Table

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
<th>Valid</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sam</td>
<td>Pizza</td>
<td>T</td>
</tr>
<tr>
<td>Ken</td>
<td>Cake</td>
<td>T</td>
</tr>
<tr>
<td>Tim</td>
<td>Carrot</td>
<td>T</td>
</tr>
<tr>
<td>Dan</td>
<td>Bacon</td>
<td>T</td>
</tr>
<tr>
<td>Tim</td>
<td>Eggs</td>
<td>F</td>
</tr>
<tr>
<td>Ben</td>
<td>Apple</td>
<td>T</td>
</tr>
</tbody>
</table>

Client 1
- Write Data
- Read Indices
- Read Primary
- Read Secondary

Nessie
- Write Tim
- Delete Tim
- Write Ken

Client 2
# Nessie: Handling Conflicts

**Index Table**

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
<th>Valid</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sam</td>
<td>Pizza</td>
<td>T</td>
</tr>
<tr>
<td>Ken</td>
<td>Cake</td>
<td>T</td>
</tr>
<tr>
<td>Tim</td>
<td>Carrot</td>
<td>T</td>
</tr>
<tr>
<td>Dan</td>
<td>Bacon</td>
<td>T</td>
</tr>
<tr>
<td>Tim</td>
<td>Eggs</td>
<td>F</td>
</tr>
<tr>
<td>Ben</td>
<td>Apple</td>
<td>T</td>
</tr>
</tbody>
</table>

**Data Table**

- **Key**: Tim
- **Value**: Eggs
- **Valid**: F

### Nessie Diagram

- **Client 1**: Write Data, Read Indices, Read Primary, Read Secondary, CAS Index, FAIL expected 5
- **Client 2**: Delete Tim, Write Ken
- **Nessie**
  - Write Data
  - Read Indices
  - Read Primary
  - Read Secondary
  - CAS Index

```
<table>
<thead>
<tr>
<th>Key</th>
<th>Client 1</th>
<th>Client 2</th>
<th>Nessie</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tim</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ken</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tim</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sam</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dan</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ben</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```
Nessie: Handling Conflicts

Client 1  Nessie  Client 2

Write Data
Read Indices
Read Primary
Read Secondary

Delete Tim
Write Ken

CAS Index
FAIL expected 5
Abort

Data Table

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
<th>Valid</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sam</td>
<td>Pizza</td>
<td>T</td>
</tr>
<tr>
<td>Ken</td>
<td>Cake</td>
<td>T</td>
</tr>
<tr>
<td>Tim</td>
<td>Carrot</td>
<td>T</td>
</tr>
<tr>
<td>Dan</td>
<td>Bacon</td>
<td>T</td>
</tr>
<tr>
<td>Tim</td>
<td>Eggs</td>
<td>F</td>
</tr>
<tr>
<td>Ben</td>
<td>Apple</td>
<td>T</td>
</tr>
</tbody>
</table>

Index Table

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Write
Key: Tim
Value: Eggs
Nessie: Migration

**Migrate**
Key: Dan

**Index Table**

| 10 | 1  | 5 | 11 |

**Destination**

**Source**

**Data Table**

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
<th>Valid</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dan</td>
<td>Bacon</td>
<td>T</td>
</tr>
<tr>
<td>Sam</td>
<td>Pizza</td>
<td>T</td>
</tr>
<tr>
<td>Ben</td>
<td>Apple</td>
<td>T</td>
</tr>
<tr>
<td>Ken</td>
<td>Cake</td>
<td>T</td>
</tr>
<tr>
<td>Dan</td>
<td>Bacon</td>
<td>T</td>
</tr>
</tbody>
</table>

**Index Table**

| 10 | 1  | 5 | 11 |

**Data Table**

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
<th>Valid</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dan</td>
<td>Bacon</td>
<td>T</td>
</tr>
<tr>
<td>Sam</td>
<td>Pizza</td>
<td>T</td>
</tr>
<tr>
<td>Ben</td>
<td>Apple</td>
<td>T</td>
</tr>
<tr>
<td>Ken</td>
<td>Cake</td>
<td>T</td>
</tr>
<tr>
<td>Dan</td>
<td>Bacon</td>
<td>T</td>
</tr>
</tbody>
</table>

**Index Table**

| 10 | 1  | 5 | 11 |

**Data Table**

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
<th>Valid</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dan</td>
<td>Bacon</td>
<td>T</td>
</tr>
<tr>
<td>Sam</td>
<td>Pizza</td>
<td>T</td>
</tr>
<tr>
<td>Ben</td>
<td>Apple</td>
<td>T</td>
</tr>
<tr>
<td>Ken</td>
<td>Cake</td>
<td>T</td>
</tr>
<tr>
<td>Dan</td>
<td>Bacon</td>
<td>T</td>
</tr>
</tbody>
</table>

**Index Table**

| 10 | 1  | 5 | 11 |

**Data Table**

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
<th>Valid</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dan</td>
<td>Bacon</td>
<td>T</td>
</tr>
<tr>
<td>Sam</td>
<td>Pizza</td>
<td>T</td>
</tr>
<tr>
<td>Ben</td>
<td>Apple</td>
<td>T</td>
</tr>
<tr>
<td>Ken</td>
<td>Cake</td>
<td>T</td>
</tr>
<tr>
<td>Dan</td>
<td>Bacon</td>
<td>T</td>
</tr>
</tbody>
</table>

**Index Table**

| 10 | 1  | 5 | 11 |

**Data Table**

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
<th>Valid</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dan</td>
<td>Bacon</td>
<td>T</td>
</tr>
<tr>
<td>Sam</td>
<td>Pizza</td>
<td>T</td>
</tr>
<tr>
<td>Ben</td>
<td>Apple</td>
<td>T</td>
</tr>
<tr>
<td>Ken</td>
<td>Cake</td>
<td>T</td>
</tr>
<tr>
<td>Dan</td>
<td>Bacon</td>
<td>T</td>
</tr>
</tbody>
</table>
Nessie: Migration

Data Table

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
<th>Valid</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dan</td>
<td>Bacon</td>
<td>T</td>
</tr>
<tr>
<td>Sam</td>
<td>Pizza</td>
<td>T</td>
</tr>
<tr>
<td>Ben</td>
<td>Apple</td>
<td>T</td>
</tr>
<tr>
<td>Ken</td>
<td>Cake</td>
<td>T</td>
</tr>
</tbody>
</table>

Index Table

10
1
5
11

Client 1  Nessie  Client 2
Nessie: Migration

Client 1  Nessie  Client 2

Copy Data

Data Table

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
<th>Valid</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dan</td>
<td>Bacon</td>
<td>T</td>
</tr>
<tr>
<td>Sam</td>
<td>Pizza</td>
<td>T</td>
</tr>
<tr>
<td>Dan</td>
<td>Bacon</td>
<td>F</td>
</tr>
<tr>
<td>Ben</td>
<td>Apple</td>
<td>T</td>
</tr>
<tr>
<td>Ken</td>
<td>Cake</td>
<td>T</td>
</tr>
</tbody>
</table>

Index Table

10
1
5
11
Nessie: Migration

Client 1 → Nessie → Client 2

Copy Data

Write Dan

Index Table

Data Table

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
<th>Valid</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dan</td>
<td>Bacon</td>
<td>T</td>
</tr>
<tr>
<td>Dan</td>
<td>Tofu</td>
<td>T</td>
</tr>
<tr>
<td>Sam</td>
<td>Pizza</td>
<td>T</td>
</tr>
<tr>
<td>Dan</td>
<td>Bacon</td>
<td>F</td>
</tr>
<tr>
<td>Ben</td>
<td>Apple</td>
<td>T</td>
</tr>
<tr>
<td>Ken</td>
<td>Cake</td>
<td>T</td>
</tr>
</tbody>
</table>
Nessie: Migration

Data Table

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
<th>Valid</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dan</td>
<td>Bacon</td>
<td>T</td>
</tr>
<tr>
<td>Dan</td>
<td>Tofu</td>
<td>T</td>
</tr>
<tr>
<td>Sam</td>
<td>Pizza</td>
<td>T</td>
</tr>
<tr>
<td>Dan</td>
<td>Bacon</td>
<td>F</td>
</tr>
<tr>
<td>Ben</td>
<td>Apple</td>
<td>T</td>
</tr>
<tr>
<td>Ken</td>
<td>Cake</td>
<td>T</td>
</tr>
</tbody>
</table>

Index Table

| 10  |
| 6   |
|x2   |
| 5   |
| 11  |

Copy Data

Update Dest.

Write Dan
## Nessie: Migration

### Data Table

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
<th>Valid</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dan</td>
<td>Bacon</td>
<td>T</td>
</tr>
<tr>
<td>Dan</td>
<td>Tofu</td>
<td>T</td>
</tr>
<tr>
<td>Sam</td>
<td>Pizza</td>
<td>T</td>
</tr>
<tr>
<td>Dan</td>
<td>Bacon</td>
<td>F</td>
</tr>
<tr>
<td>Ben</td>
<td>Apple</td>
<td>T</td>
</tr>
<tr>
<td>Ken</td>
<td>Cake</td>
<td>T</td>
</tr>
</tbody>
</table>

### Index Table

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dan</td>
<td>Bacon</td>
</tr>
<tr>
<td>Dan</td>
<td>Tofu</td>
</tr>
<tr>
<td>Sam</td>
<td>Pizza</td>
</tr>
<tr>
<td>Dan</td>
<td>Bacon</td>
</tr>
</tbody>
</table>

### Diagram

- **Client 1**
  - Copy Data
  - Update Dest.
  - CAS Index
  - FAIL expected 1

- **Nessie**
  - Write Dan

- **Client 2**
  - Migrate Key: Dan

---

49
Nessie: Migration

Data Table

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
<th>Valid</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dan</td>
<td>Bacon</td>
<td>T</td>
</tr>
<tr>
<td>Dan</td>
<td>Tofu</td>
<td>T</td>
</tr>
<tr>
<td>Sam</td>
<td>Pizza</td>
<td>T</td>
</tr>
<tr>
<td>Dan</td>
<td>Bacon</td>
<td>F</td>
</tr>
<tr>
<td>Ben</td>
<td>Apple</td>
<td>T</td>
</tr>
<tr>
<td>Ken</td>
<td>Cake</td>
<td>T</td>
</tr>
</tbody>
</table>

Index Table

<table>
<thead>
<tr>
<th>Key Value Valid</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
</tr>
<tr>
<td>6</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>11</td>
</tr>
</tbody>
</table>

Abort
Performance Tradeoffs

Expected number of roundtrips for a write operation in Nessie:

<table>
<thead>
<tr>
<th>Load Factor</th>
<th>Roundtrips</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.10</td>
<td>2.5</td>
</tr>
<tr>
<td>0.25</td>
<td>3.5</td>
</tr>
<tr>
<td>0.50</td>
<td>5.8</td>
</tr>
<tr>
<td>0.75</td>
<td>9.9</td>
</tr>
<tr>
<td>0.90</td>
<td>13.3</td>
</tr>
</tbody>
</table>
Nessie’s Design Complexity

- Contention over the index table creates complexity

- Our solution mirrors that of Hardware Transactional Memory
  - Optimistically attempt operation
  - Check for conflicting operations
  - Abort and retry if a conflict is detected

- NIC support of HTM would greatly improve Nessie
  - NIC handles detection and reporting of conflicts
  - Fewer roundtrips
Hardware Transactional Memory

XBEGIN
    Read Index Table
    Write to Index Table
    ...
    Write to Index Table
XEND
HTM + RDMA

XBEGIN
Read Index Table
Write to Index Table
...
Write to Index Table
XEND
Summary

- Low latency is critical for in-memory distributed systems

- Nessie:
  - Exclusively RDMA reads, writes, and CAS operations
  - Achieves low-latency
  - Shared lock-free data

- HTM in conjunction with RDMA can greatly simplify Nessie

- We are excited to build Nessie and explore HTM