Lizard

A Linked Data Publishing Platform

Andy Seaborne
Epimorphics Ltd.
Outline

The (a) real world of service provision

What to do about (some of) it

How to do that
Who am I?

Andy Seaborne

Editor on SPARQL query

A committer on Apache Jena

At Epimorphics Ltd
This work

- Epimorphics
- Funding: InnovateUK*
- Users
  - For the discussion and encouragement

* Used to be the Technology Strategy Board.
UK Department for Business, Innovation & Skills
Example Services

http://environment.data.gov.uk/

http://landregistry.data.gov.uk/
Maximise usage

Publication not application
Running Services

Data publishing != Database backed web site

- Different traffic patterns
  - Expensive queries, less control
  - Bot multiplier effect

- “Admin”
  - SLAs: Heartbleed
Problem Statement

- Reacting to events
- Machine administration / SLAs
Goals

24x7 Operation

Consistency
About Consistency

Makes the system easier to use
  ○ For users
  ○ For operators

Each query sees an unchanging database

… that did exist; no “bit of this, bit of that”

Clients may conspire!
Apache Jena TDB

➢ Node Table
  ○ Inline values (integers, date/datetime, …)

➢ Indexes are covering
  ○ Range scans
  ○ All key, no value
  ○ No "triple table"
SPARQL Execution

\{ ?x :p 123 . \}

Convert to NodeIds

Look in POS to get all PO?, assign S to ?x

123 is an inline constant in TDB.

\{ ?x :p 123 . 
  ?x :q ?v . \}

A database join

Index join (Loop+substitution)

Index join (= loop) on

: x1 :q ?v

where : x1 is the value of ?x
TDB uses threaded B+Trees for indexes

- 8K blocks 100-way B+Tree
Choices

Query and Update

Indexes / B+Trees

Node table / Objects

Blocks

Key → Value Store
This Does Not Work (very well)

<table>
<thead>
<tr>
<th>Query and Update</th>
<th>Distribute the storage</th>
</tr>
</thead>
<tbody>
<tr>
<td>B+Trees</td>
<td>K-&gt;V store</td>
</tr>
<tr>
<td>Objects</td>
<td>Index access on query processor</td>
</tr>
<tr>
<td>Blocks</td>
<td></td>
</tr>
<tr>
<td>Key→Value</td>
<td></td>
</tr>
</tbody>
</table>

- Easy to do (pick a KV store of your choice)
- Impedance mismatch
  - Too much data moving about
  - Little parallelism
  - Bad cold-start
Distribute

➢ Distribute the indexes
  ○ With modified index access
➢ Distribute the nodes
➢ Comms : Apache Thrift
Clustered Node Table

➢ Node Table
  - $N$ replicas; Read $R$ / Write $W$
    - e.g. $W=N$ and $R=1$ =>
      Complete copies of node table on each data server
  - Can shard
  - Replaceable
    - Requirement: NodeId for naming
Clustered Indexes

Indexes

- Can shard by subject
- Replicas of each shard \((R=1, W=N)\)
- Compound access operations
Clustered Indexes

Index

Shard 1

Shard 2

Shard 3

Machine 1

Machine 2
Modified SPARQL Execution

- Different unit of index access
  - subject + several predicates
    - (subj, pred1, pred2, pred3, ...)

- Different join algorithms
  - Merge join
  - Parallel hash join
Configuration 1

Load Balancer (or RR-DNS)

Query server

Data server
- POS Copy 1
- PSO Copy 2

Data server
- POS Copy 1
- PSO Copy 2

Data server
- Node Copy 1

Data server
- Node Copy 2
Configuration 2

Load Balancer (or RR-DNS)

Query server

Data server
- POS
  Copy 1
- PSO
  Copy 2
- Node
  Copy 1

Query server

Data server
- POS
  Copy 1
- PSO
  Copy 2
- Node
  Copy 2
Status

Working prototype

Spin-off : TDB2
New Technology

- Copy-on-write indexes
- New transactional coordinator
- Apache Thrift encoded node table

Side effect: TDB2
  - Arbitrary scaling transactions
  - Transactional only
  - Space recovery
It looks kind of weird, are you sure you did this right?

200% sure

I'm not sure that's exactly how they wanted it...

Well I'm pretty sure I made it exactly as requested

I followed the specifications to the letter

Okay, I guess I'll send it to the client then

That's right. Send it to the client

To the letter? To the last bit