ACID or BASE? – the case of NoSQL

Peter Vanroose & Kris Van Thillo

TRAINING & CONSULTING

GSE DB2 Belgium

Joint User Group Meeting

IBM, Brussels, 12 June 2014

“*What’s next?*”

ACID or BASE?
– the case of NoSQL

1. NoSQL - what's in a name
2. NoSQL database arch
3. NoSQL database types
4. ACID or BASE?
5. The CAP theorem
6. Comm. NoSQL databases

ACID or BASE? – the case of NoSQL

Summary :

- an alternative to relational databases -- why?
- availability versus consistency: replication, distributed, ...
- key-value stores, columnar databases, document stores, ...
- commercial NoSQL implementations: a few examples

NoSQL - what's in a name

Wikipedia:

A NoSQL or Not Only SQL database provides a mechanism for

- storage and retrieval of data
- modelled otherwise than in relational database tables & relations
- motivations for this approach include:
  simplicity of design,
  horizontal scaling,
  finer control over availability,
  faster than in some RDBMS

NoSQL databases are finding significant, growing industry use in *big data* and *real-time web* applications.

Many NoSQL stores *compromise consistency* in favour of *availability* and *partition tolerance* ("CAP theorem")

Most NoSQL stores lack true *ACID transactions*

Term introduced 1998 by Carlo Strozzi (really meaning *no SQL*);

reintroduced 2009 by Eric Evans in the context of distributed DBs

ACID or BASE?
– the case of NoSQL

1. NoSQL - what's in a name
2. NoSQL database arch
3. NoSQL database types
4. ACID or BASE?
5. The CAP theorem
6. Comm. NoSQL databases

NoSQL and Big Data

Big Data:

- *3V* (Gartner): high-**V**olume, high-**V**elocity data with high **V**ariety
- enables decision making, insight discovery, process optimization

  ==> data *analysis* is central: data mining; statistical techniques

  ==> *distributed* analysis starts to make sense
- insight:
  - keep *all* data (sensor data, website clicks, blogs, ...)
  - in their *original* format (no Data Warehouse style ETL)
  - for potential later use (not yet decided)

    (pre-formatting destroys / biases information)
- as a consequence:
  - unstructured (or semi-structured, non-flat) data
  - no (or less) quality control or semantics during load
  - interpretation & value judgement: done by ad-hoc analysis step

ACID or BASE?
– the case of NoSQL

1. NoSQL - what's in a name
2. NoSQL database arch
3. NoSQL database types
4. ACID or BASE?
5. The CAP theorem
6. Comm. NoSQL databases

What's the problem with relational databases? 1.2

ACID or BASE?
– the case of NoSQL

1. NoSQL - what's in a name
2. NoSQL database arch
3. NoSQL database types
4. ACID or BASE?
5. The CAP theorem
6. Comm. NoSQL databases

P#1: must convert information from their natural representation into table(s)

P#2: must later reconstruct information from tabular representation

P#3: data must be modelled (semantics!) before storing it

P#4: a table column can only store similar data ("schema" is fixed)

P#5: relational systems may not scale well

P#6: joins between different systems (different identifiers): difficult

P#7: SQL dialects vary => difficult to port applications between databases

P#8: complex business rules are not easily expressible in SQL

P#9: approximate terms and fuzzy searches: not performing well

P#10: RDBMS don't store & validate complex documents efficiently

1. NoSQL - what's in a name
2. NoSQL database arch
3. NoSQL database types
4. ACID or BASE?
5. The CAP theorem
6. Comm. NoSQL databases

What's the problem with relational databases? (cont'd)

Hey, that rings a bell ...

"*Store your DB2 data as XML*":

- no need to convert back/forth to/from tabular representation
- no need to (re)interpret the XML structure when loading
- no need for predefined schema (columns & data types)
- let the reading application do the difficult work:

```
SELECT  coname, XMLQUERY('count($E//function[.="analyst"])'
                         PASSING b.employees as E        )
FROM    companies b
WHERE   XMLEXISTS('$E/employees/person[function="analyst"]'
                         PASSING employees AS E          )
```

where the content of XML column "employees" could be something like:

```
<?xml version="1.0" encoding="UTF-8"?>
<employees cono="32"><person><lname>Janssen</lname><fname>D.</fname>
    <address><street>Kortestraat</street><city>Leuven</city></address>
    <function>analyst</function></person>
<person>....<function>programmer</function></person></employees>
```

1. NoSQL - what's in a name
2. NoSQL database arch
3. NoSQL database types
4. ACID or BASE?
5. The CAP theorem
6. Comm. NoSQL databases

What's the problem with relational databases? (cont'd)

XML: still too rigid / too limited

How can we *store anything whatsoever*

and yet easily *find it back* and/or

***aggregate* on it (count/sum/avg/rank/top10/...)**

"*In search of a middle ground between file system & database*"
(Robert Greene, 2012)

Solutions (?) ==> NoSQL !

- ***schema-less* storage (=> dynamically add new attributes)**
- **but with *keys* & values (tuple store, ...) & possibly indexes**
- **using a *distributed* storage model (autonomous nodes; TCP/IP)**
- **with *replication* for fault-tolerance (redundancy across nodes)**

  ==> hence can afford "commodity hardware"; scales linearly
- **BUT which *guarantees* can such a setup provide us?**

NoSQL database architecture 2

schema-less storage 2.1

most NoSQL databases offer the possibility to work

- without a “schema”, i.e., a predefined structure
- or with dynamically changing schema’s

distributed (partitioned) 2.2

***scaling out* instead of scaling up:**

- “shared nothing” architecture: no common disk/memory/processor
- each participant is a cluster *node* (identity; network topology)
- node = both data and *analysis* jobs: work can be “threaded”

sharding, *replication*, fault-tolerance 2.3

a *shard* is a table partition, but isolated on a cluster node

- multiple nodes store the same partition (& allows read parallelism)

data (row) *versioning* 2.4

- may become crucial because of replication!

ACID or BASE?
– the case of NoSQL

1. NoSQL - what’s in a name
2. NoSQL database arch
3. NoSQL database types
4. ACID or BASE?
5. The CAP theorem
6. Comm. NoSQL databases

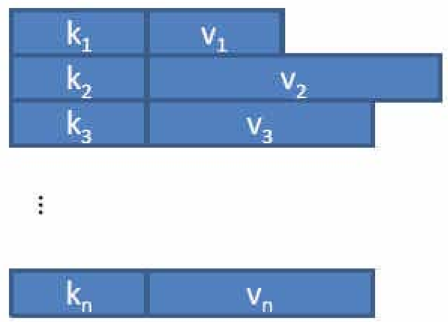
1. NoSQL - what's in a name
2. NoSQL database arch
3. NoSQL database types
4. ACID or BASE?
5. The CAP theorem
6. Comm. NoSQL databases

NoSQL database types 3

Key/Value Databases 3.1

- **values (data) stored based on programmer-defined** keys [hash table approach]
- **system is agnostic as to the semantics of the value**
- **requests are expressed in terms of keys**

  put(key, value)
  get(key): value
- **indexes are defined over keys**
  [some systems support secondary indexes over (part of) the value]



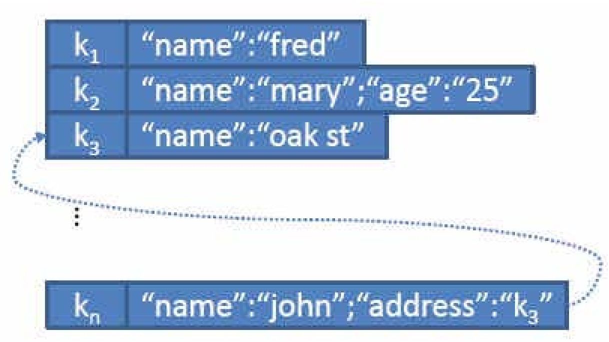
Examples: Berkeley DB, Oracle NoSQL, LevelDB, Dynamo, Memcached

1. NoSQL - what's in a name
2. NoSQL database arch
3. NoSQL database types
4. ACID or BASE?
5. The CAP theorem
6. Comm. NoSQL databases

NoSQL database types (cont'd)

Document Data Model 3.2

- **documents are stored based on a programmer-defined key**
  [a key-value store]
- **system is aware of the arbitrary document structure**
- **support for lists, pointers and nested documents**
- **requests expressed in terms of key (or attribute, if index exists)**
- **support for key-based indexes and secondary indexes**



Examples: MongoDB, CouchDB, RaptorDB, Riak, *IBM Lotus Notes*

NoSQL database types (cont'd)

Columnar Databases 3.3

[wide column store - 'Big Table' clones - cf. DB2 LUW with BLU]

- **stores data tables as sections of columns of data**
  [rather than as rows of data] [hybrid row/column structure]
- **data stored together with meta-data ('a map')**
  [typically including row id, attribute name & value, timestamp]
- **most often sparse storage**
- **"like just storing indexes, one per column"**

Peter	2013	75
Peter	2014	11
Peter		3
Eliza	2014	70
Eliza	2014	1
	2012	472
	2011	11
...		
	...	
		...

→

```
Eliza : [ 4, 5 ]
Peter: [ 1, 2, 3 ]
...
```

```
2011 : [ 7, ... ]
2012 : [ 6, ... ]
2013 : [ 1 ]
2014 : [ 2, 4, 5 ]
```

```
1   : [ 5 ]
3   : [ 3 ]
11  : [ 2, 7 ]
70  : [ 4 ]
75  : [ 1 ]
472 : [ 6 ]
...
```

or

```
(name, Eliza, v1, 4)
(name, Eliza, v1, 5)
(name, Peter, v1, 1)
(name, Peter, v1, 2)
(name, Peter, v1, 3)
(date, 2011, v1, 7)
...
(date, 2012, v1, 6)
...
(date, 2013, v1, 1)
(date, 2014, v1, 2)
(date, 2014, v1, 4)
(date, 2014, v1, 5)
(amount, 1, v1, 5)
(amount, 3, v1, 3)
(amount, 4, v1, 2)    *
(amount, 11, v1, 7)
(amount, 11, v2, 2)
(amount, ...)
```

Examples: Google Bigtable (2006), HBase, Hypertable, Cassandra

1. NoSQL - what's in a name
2. NoSQL database arch
3. NoSQL database types
4. ACID or BASE?
5. The CAP theorem
6. Comm. NoSQL databases

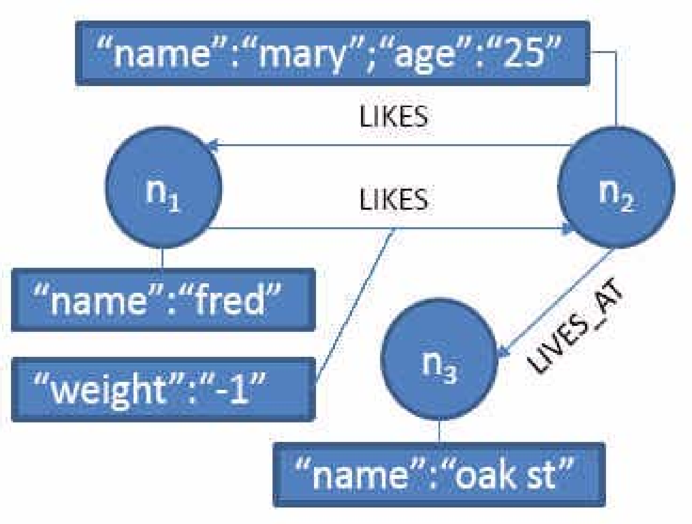
1. NoSQL - what's in a name
2. NoSQL database arch
3. NoSQL database types
4. ACID or BASE?
5. The CAP theorem
6. Comm. NoSQL databases

NoSQL database types (cont'd)

Graph Data Model

3.4

- **data is stored in terms of nodes and links**
  both can have (arbitrary) attributes
- **requests are expressed based on system id's (if no indexes exist)**
  secondary indexes for nodes and links are supported
- **SPARQL query language:** retrieve nodes by attributes and links by type, start and/or end node, and/or attributes



Examples: Neo4j, InfoGrid, IMS

Vendors are embracing NoSQL

1. NoSQL - what's in a name
2. NoSQL database arch
3. NoSQL database types
4. ACID or BASE?
5. The CAP theorem
6. Comm. NoSQL databases

... as they did with MDM, XML, OO, ... ??

(or is this different?)

- **Oracle [key value] :** BerkeleyDB, NoSQL DB
- **IBM:**

  [key value, columnar] : BigInsights / HBase (Linux; uses Hadoop)

  IBM DB2 LUW + BLU accelerator (ACID!)

  BlueRunner (Cassandra): email in the cloud

  [document] : IBM DB2 + MongoDB support ("*DB2 JSON*")

  [graph] : IBM DB2 + Triple-Graph Store option
- **Microsoft :** Azure [SaaS]
- **...**

ACID or BASE?

Transactions, consistency and availability

- **In a 'shared something' environment, ACID is wanted:**
  Pessimistic behaviour: force consistency at end of transaction!
  - **Atomicity:** all or nothing (of the *n* actions): commit or rollback
  - **Consistency:** transactions never observe or cause inconsistent data
  - **Isolation:** transactions are not aware of concurrent transactions
  - **Durability:** acknowledged transactions persist in all events
- **In a 'shared nothing' environment, BASE is implemented:**
  Optimistic behaviour: accept temporary database inconsistencies
  - **Basically Available** [guaranteed thanks to replication]
  - **Soft state** [it's the user's (application's) task to guarantee consistency]
  - **Eventually consistent (weakly consistent)**
    [database will be consistent in the longer run; 'stale' data is OK]

Why not have both? => consistency & availability & speed (through sharding)?

ACID or BASE?
– the case of NoSQL

1. NoSQL - what's in a name
2. NoSQL database arch
3. NoSQL database types
4. ACID or BASE?
5. The CAP theorem
6. Comm. NoSQL databases

The CAP theorem

Brewer's Conjecture (2000; proved in 2002; refined in 2012):

Real world data storage systems like to have three properties:

- [data] **Consistency** [all clients see the same data at the same time]
- [data] **Availability** [guaranteed server response: success or failure]
- **Partition tolerance** [nodes/messages may fail/get lost/unreachable]

Conjecture:

in a multi server/node/rac shared nothing environment
it is only possible to satisfy at most two of these requirements

C+A ~ "ACID": this needs a single, central server (with replication ?)

C+P: either "write N, read 1" or "write 1, read N" (maybe too slow ?)

A+P = "BASE": no strong consistency guarantees ...

(in reality: C, A, P are continuum; choices can be "ad hoc" !)

==> sacrifice consistency to gain faster responses in a more scalable manner

ACID or BASE?
– the case of NoSQL

1. NoSQL - what's in a name
2. NoSQL database arch
3. NoSQL database types
4. ACID or BASE?
5. The CAP theorem
6. Comm. NoSQL databases

CAP theorem: consequences

ACID (RDBMS)	BASE (NoSQL)
strong consistency	weak consistency (=> allow stale data)
isolation	last write wins
transaction	program managed
robust database	simple database
simple code (SQL)	complex code
available & consistent	available & partition-tolerant
scale-up (limited)	scale-out (unlimited)
shared-something (disk, mem, proc)	shared-nothing (parallellizable)

ACID or BASE?
– the case of NoSQL

1. NoSQL - what's in a name
2. NoSQL database arch
3. NoSQL database types
4. ACID or BASE?
5. The CAP theorem
6. Comm. NoSQL databases

The NRW notation: 5.2

1. NoSQL - what's in a name
2. NoSQL database arch
3. NoSQL database types
4. ACID or BASE?
5. The CAP theorem
6. Comm. NoSQL databases

where N = # replica's per item,
R = # reads (before declaring "success"),
W = # writes (before declaring "success"):

NRW=n1n: read-optimized strong consistency

cf DB2's logging mechanism

NRW=nn1: write-optimized strong consistency

cf DB2's buffer pool reading mechanism; recovery mechanisms

weak eventual consistency: when e.g. R+W <= N

suppose N=3, R=1, W=1:

- a certain data item is stored on nodes A, B and C
- client1 modifies the item through node A (& receives success msg)
- "eventually", nodes B and C will be updated
- client2 reads & modifies same item through node B (& success)
  BUT before node B got updated!
- conflict resolution ==> timestamps (versioning) needed
- clients *could* later be notified of the occurrence of this conflict

1. NoSQL - what's in a name
2. NoSQL database arch
3. NoSQL database types
4. ACID or BASE?
5. The CAP theorem
6. Comm. NoSQL databases

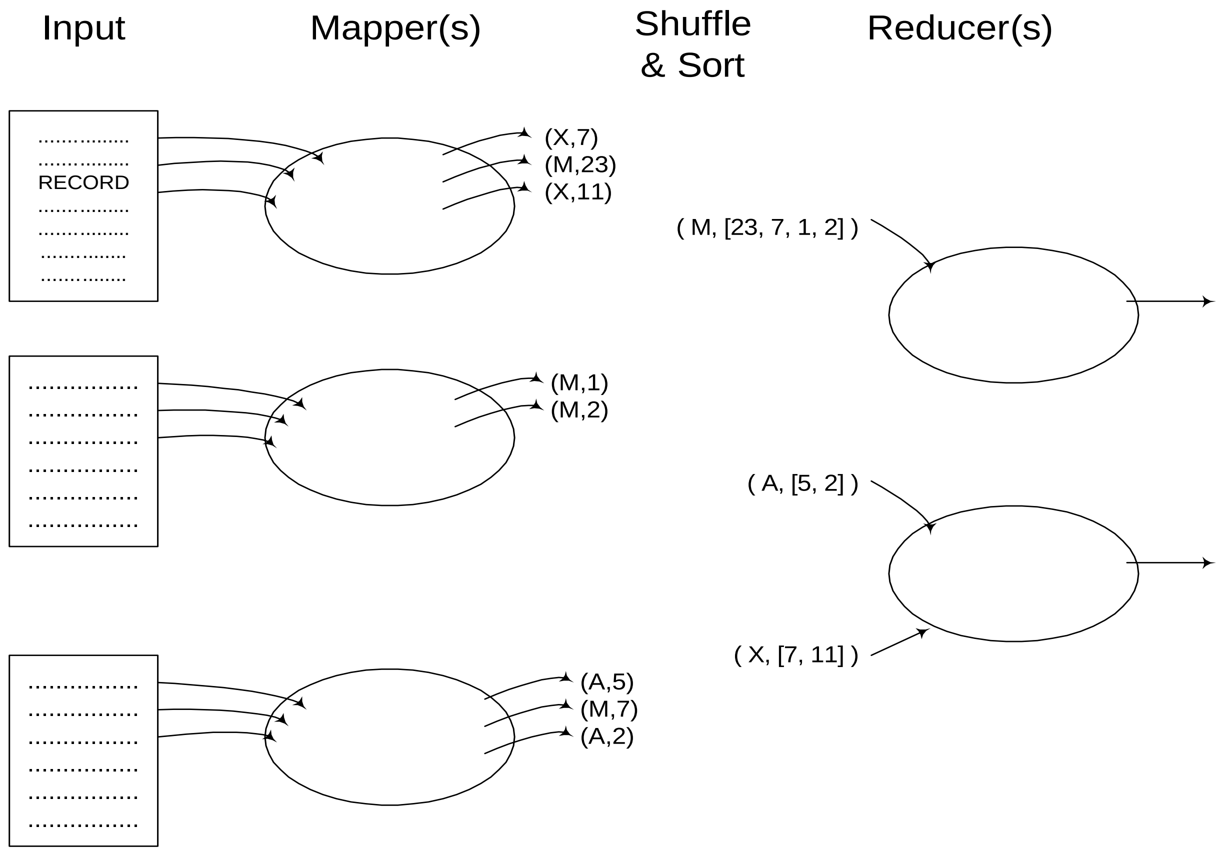
Framework example: MapReduce design patterns

5.3

Not just *data* is distributed, also the *application logic* must be

==> "*bring the program close to the data it is reading/writing*"

A **MapReduce** *framework* simplifies implementing parallel algorithms:



MapReduce design patterns (cont'd)

- Filtering ("WHERE"): done in the mappers
- Top-N filtering: needs ranking: pre-filtering in the mappers
- Distinct filter: use combiners; whole record as key, no value
- Summarization (count, sum, average, ...): combiners & reducer(s)
  with "GROUP BY": multiple reducers are possible
- Total-order sorting: is often not needed!
- Reduce-side join (is essentially a merge-scan join)
- Meta-patterns: job chaining

Further reading: see e.g.

"MapReduce Design Patterns", Donald Miner & Adam Shook, 2013

ACID or BASE? – the case of NoSQL

1. NoSQL - what's in a name
2. NoSQL database arch
3. NoSQL database types
4. ACID or BASE?
5. The CAP theorem
6. Comm. NoSQL databases

SQL vs NoSQL

	SQL	NoSQL
types	one 'logical' database, with somewhat distinct 'physical' impl.	many different types [columnar, key/value, document, ..]
history	1970	2000
storage	table/row/column a.k.a. file/record/field storage	it depends: records, documents ++ unstructured ++
schema	'static' schema's structure is pre-determined	'dynamic' or no schema ++ schema-free ++
scaling	vertical	horizontal ++ easier, cheaper ++
dvlpmnt model	initially: proprietary; later: open source	open source ++ agile ++
trans-actions	consistency: ACID ++ yes ++	consistency: BASE -- not always --
DML	++SQL++	OO; also SQL-like -- infancy --

other concerns:
security & access control; optimizer; check constraints; ...

ACID or BASE?
– the case of NoSQL

1. NoSQL - what's in a name
2. NoSQL database arch
3. NoSQL database types
4. ACID or BASE?
5. The CAP theorem
6. Comm. NoSQL databases

Hive 6.1

- Not really a database: rather “data warehouse software”
  - file based storage
  - maintains a “metastore”
- Built on top of Hadoop (Apache)
  - files are actually HDFS (distributed & replicated) fragments

    ==> **optimized for retrieve & for append**; update not supported
- Is itself an Apache project: free, open-source `http://hive.apache.org/`
- HiveQL:
  - SQL-style language
  - optimizer creates MapReduce source code (in Java)
- Command-line interface:

```
hive -f file.hive        ==> run script
hive -e 'cmd'            ==> execute a single Hive command
hive                     ==> interactive mode
```

ACID or BASE?
– the case of NoSQL

1. NoSQL - what’s in a name
2. NoSQL database arch
3. NoSQL database types
4. ACID or BASE?
5. The CAP theorem
6. Comm. NoSQL databases

HiveQL - statements

CREATE TABLE *tablename* (*col1* type, *col2* type) [PARTITIONED BY (col3 type)]
[ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t']
[STORED AS TEXTFILE] ;

SHOW TABLES ;

DESCRIBE *tablename* ;

LOAD DATA [LOCAL] INPATH 'filename' [OVERWRITE] INTO TABLE *tablename* ;

-- input data must already be in the correct format (incl. delimiters) for the table!

INSERT INTO dest [PARTITION col3=...] SELECT cols FROM src WHERE ...

SELECT *colnames*, expr FROM *tablename* WHERE cond ORDER BY expr LIMIT 10 ;

SELECT *colname*, SUM(expr) FROM *tablename* WHERE cond GROUP BY *colname* ;

FROM src INSERT OVERWRITE TABLE dest SELECT cols WHERE cond ;

FROM src INSERT OVERWRITE [LOCAL] DIRECTORY 'dirname' SELECT * WHERE ... ;

FROM tbl1 JOIN tbl2 ON (col1=col2) INSERT OVERWRITE TABLE t3 SELECT col1,col3 ;

Supported datatypes:

string int double decimal timestamp date ...

Examples:

SELECT * FROM t WHERE col < 7; -- filtering in parallel by the mappers

SELECT * FROM t ORDER BY col DESC LIMIT 10; -- pre-filtering; no global sorting!

SELECT col, MIN(val), COUNT(1) FROM t GROUP BY col; -- combiners & reducers

SELECT * FROM t1 LEFT OUTER JOIN t2 ON (t1.fkcol = t2.fkcol);

1. NoSQL - what's in a name
2. NoSQL database arch
3. NoSQL database types
4. ACID or BASE?
5. The CAP theorem
6. Comm. NoSQL databases

Hive - Interacting with Java

- **The Hive cli should be used as the principal user interface**
- **Hive allows calling Java Hadoop programs by writing and using Hive User Defined Functions (UDFs) in Java**
- **Additionally, there exists a JDBC interface to Hive**

All cluster behaviour is managed by HDFS

- **data nodes (3-fold replication)**
- **single name node, “master” (for metadata)**
- **clients transparently communicate with the cluster**
  through Java API
- **files are automatically split in blocks of 128 MB**
- **file compression on-the-fly**
- **supports *streaming* data access (“write-once, read-many”)**
- **nodes run Linux; access through TCP/IP**

1. NoSQL - what’s in a name
2. NoSQL database arch
3. NoSQL database types
4. ACID or BASE?
5. The CAP theorem
6. Comm. NoSQL databases

1. NoSQL - what's in a name
2. NoSQL database arch
3. NoSQL database types
4. ACID or BASE?
5. The CAP theorem
6. Comm. NoSQL databases

HBase

- Built on top of Hadoop (Apache) just like Hive
  - written in Java
  - files are actually HDFS (distributed & replicated) fragments
- Is itself an Apache project: free, open-source `http://hbase.apache.org/`
- Only uses HDFS, not MapReduce
- **Columnar** NoSQL database
- columns grouped in *families* which are are stored together
- For random read+write access to large datasets
  - "strongly consistent" writes: CP, not AP
  - still non-ACID: e.g. no rollbacks: each action is a transaction
- By design: no datatypes, no foreign keys, no indexes, no triggers
- table cells are just byte sequences (non-typed)
- keys: column name, row pointer, timestamp (=version)
- a table is essentially a **sorted** *map*, keyed by these three

HBase: an example

The “canonical” example: the *web table*

- table of crawled web pages (with attributes); row key = URL
- continuously accessed (globally) by analytics (MapReduce) jobs
- continuously accessed (randomly) to update content & attributes
- a webtable could contain 2 column *families*: `contents` & `anchor`
  - 1 column in the `contents` family: `contents:html`
  - several in `anchor`, e.g. `anchor:lang`, `anchor: www.apache.org`

```
{   "anchor:www.apache.org" : {
        "http://www.abis.be/html/index.html" : {
            "v1" : "<a href=\"http://www.apache.org/\">Apache</a>" ,
            "v2" : .....        } ,
        .....                               } ,
    "content:html" : {
        "http://www.abis.be/html/index.html" : {
            "v1" : "<html><head> .........."          }
```

(etc.)

1. NoSQL - what’s in a name
2. NoSQL database arch
3. NoSQL database types
4. ACID or BASE?
5. The CAP theorem
6. Comm. NoSQL databases

HBase in practice

- **command line interface: example:**

```
$ hbase master start
$ hbase shell
    hbase> list
        TABLE
        0 row(s) in 0.6830 seconds
    hbase> create 't1', 'f1'                -- creates table t1 with one column family f1
    hbase> list
        TABLE
        t1
        1 row(s) in 0.0480 seconds
    hbase> put 't1', 'row1', 'f1:a', 'val1'   -- puts value "val1" in cell  (row1,f1:a)
    hbase> scan 't1'
        ROW      COLUMN+CELL
        row1     column=f1:a, timestamp=1396745613547, value=val1
    hbase> get 't1', 'row1'
        COLUMN     CELL
        f1:a       timestamp=1396745613547, value=val1
    hbase> quit
```

- **The "real" HBase use is through a Java program (API)**

1. NoSQL - what's in a name
2. NoSQL database arch
3. NoSQL database types
4. ACID or BASE?
5. The CAP theorem
6. Comm. NoSQL databases

1. NoSQL - what's in a name
2. NoSQL database arch
3. NoSQL database types
4. ACID or BASE?
5. The CAP theorem
6. Comm. NoSQL databases

Cassandra

- **Columnar NoSQL database**
- **originally developed by Facebook (2007)**
  - **but Facebook moved to HBase for its messaging platform (2010)**
- **became an Apache incubator project in 2009**
  - **written in Java**
  - **v 2.0 : May 2014**
- **free, open-source** `http://cassandra.apache.org/`
- **commercial add-ons & support ("enterprise edition")**
  by Datastax `http://www.datastax.com/`
- **CQL:**
  - **SQL-style language**
  - **but no joins, no subqueries, no GROUP BY**
  - **only since v 2.0 there is built-in cursor support**
    for a plain "SELECT col1, col2 FROM t [WHERE cond]

1. NoSQL - what's in a name
2. NoSQL database arch
3. NoSQL database types
4. ACID or BASE?
5. The CAP theorem
6. Comm. NoSQL databases

Cassandra - architecture

- **nodes form a cluster (called a "ring", but any topology allowed)**
- **peer-to-peer network**
- **meta-data is exchanged once a second, using "*gossip*" protocol**
- **meta-data: SSTables (in-memory)**
  - **contains topology info and table "catalog" info**
  - **needs regular *compaction* (cf. REORG)**

    to ensure durability of transactions
- **partitioner**
  - **distributes data cells over nodes, based on *partition key* & hash**
- **the *client* is responsible for deciding the *consistency level***
  - **in NRW terminology: R & W decided by the user**

    ==> user chooses whether Cassandra is CP or AP
- **primary key updates not allowed; no foreign keys**
- **no transactions ==> no rollbacks**

Cassandra - table layout

- **columnar database:**
  - **each cell (called a "*column*") is a (name,value,timestamp) tuple**
  - **a *row* is a collection of columns, stored in alphabetic order**
    stored together on a single node
  - **a *column family* (or a *table*) is a collection of rows**
  - **a *key space* is a group of column families**
- ***indexes* are user-defined column families**

Used by:

- **Spotify (for their playlist data)**
- **eBay (for their fraud detection implementation)**
- **(earlier) Facebook (inbox search)**
- **IBM: BlueRumor (email system in the cloud; Jun Rao)**
- **Twitter**

ACID or BASE?
– the case of NoSQL

1. NoSQL - what's in a name
2. NoSQL database arch
3. NoSQL database types
4. ACID or BASE?
5. The CAP theorem
6. Comm. NoSQL databases

MongoDB

6.4

- **JSON-style documents (BSON)** [document-based queries]
- **schema-free**
  - written in C++ for high performance
  - full index support
  - memory mapped files
  - no transactions (but supports atomic operations)
  - not relational
- **scalability**

  replication - sharding
- **MongoDB = CP, optionally AP** [on top of CP]
- **'utilities' available:**
  - mongo<u>export</u> ; mongo<u>import</u> ; ...
- **language drivers available:** C, C++, Java, JavaScript, perl, PHP, Python, Ruby, C#, Erlang, Delphi, ... [*community supported*]
- **OS:** OS X, Linux, Windows, Solaris
- **Opens source, free -** commercial edition available

**ACID or BASE?
– the case of NoSQL**

1. NoSQL - what's in a name
2. NoSQL database arch
3. NoSQL database types
4. ACID or BASE?
5. The CAP theorem
6. Comm. NoSQL databases

MongoDB - Concepts and Structures

- **A Mongo deployment (server or instance) holds a set of databases**
  - a database holds a set of collections
  - a collection holds a set of documents
  - a document is a set of fields: key-value pairs (JSON - BSON)
  - key-value-pairs:
    a *key* is a name (string)
    a *value* is a basic type like string, integer, float, timestamp, binary, etc., an embedded document, or an array of values
  - a '*special pair*': _objectid - default artificial key

  'Lazy' - [most]
  - collections & databases created when first document inserted
- **collections can be 'capped'**

  need to be created before **they can be used!**

  [no deletes, limited updates tolerated]

  have a 'fixed' size

  db.createcollection('courseColCapped', ..., ...)

ACID or BASE?
– the case of NoSQL

1. NoSQL - what's in a name
2. NoSQL database arch
3. NoSQL database types
4. ACID or BASE?
5. The CAP theorem
6. Comm. NoSQL databases

MongoDB - Concepts and Structures (cont'd)

Document-oriented : collections **store** documents **in BSON format** [collection=?= table]

- **JSON-style documents: BSON (Binary JSON)**
- **support for 'non-traditional' data types:** Date **type and a** BinData **type**
  - can reference other documents
  - lightweight (*minimal spatial overhead*), traversable (*find data quickly*), efficient (*linked to C/C++ data types*) - VERY FAST
- **all documents belonging to one and the same collection** can have **heterogeneous data structures!** [remember: no schema's]
- **typically** [check version]**: 4MB document limit**

1. NoSQL - what's in a name
2. NoSQL database arch
3. NoSQL database types
4. ACID or BASE?
5. The CAP theorem
6. Comm. NoSQL databases

MongoDB - Concepts and Structures - JSON

1. NoSQL - what's in a name
2. NoSQL database arch
3. NoSQL database types
4. ACID or BASE?
5. The CAP theorem
6. Comm. NoSQL databases

Let's first introduce JSON...

JavaScript Object Notation

°) a collection of (nested) key-value pairs

°) supporting ordered lists

°) record oriented

... and then talk about BSON [Binary JSON]

- **an 'efficient' implementation of JSON**
- **efficient use of storage space**
- **increased scan-speed**
  [large elements in a BSON document are prefixed with a length field]
- **array indices explicitly stored**

MongoDB - Concepts and Structures - JSON


```
{
   "glossary": {
      "title": "example glossary",
        "GlossDiv": {
         "title": "S",
            "GlossList": {
            "GlossEntry": {
               "ID": "SGML",
                     "SortAs": "SGML",
                     "GlossTerm": "Standard Generalized Markup Language",
                     "Acronym": "SGML",
                     "Abbrev": "ISO 8879:1986",
                     "GlossDef": {
                  "para": "A meta-markup language, used to create DocBook.",
                          "GlossSeeAlso": ["GML", "XML"]
               },
                     "GlossSee": "markup"
            }
         }
      }
   }
}
```


1. NoSQL - what's in a name
2. NoSQL database arch
3. NoSQL database types
4. ACID or BASE?
5. The CAP theorem
6. Comm. NoSQL databases

MongoDB - startup

- **Installation**

    download, unzip, create data directory, create default config file, and get started!

- **Start the MongoDB 'server'**

./bin/mongod

[***bin\mongod.exe***]

- **Start MongoDB 'client' -** interactive JavaScript shell

./bin/mongo

[***bin\mongo.exe***]

[root@everest bin]# ./mongod --dbpath /data/db **--port** 27017 **--config** /etc/mongod.conf

1. NoSQL - what's in a name
2. NoSQL database arch
3. NoSQL database types
4. ACID or BASE?
5. The CAP theorem
6. Comm. NoSQL databases

MongoDB - basics

Basic commands - examples

use [db name]

show dbs
show collections

Basic operations

- **Insert operations** [sample]

```
> use coursedb
switched to db coursedb
> db.courseCol.insert({"Coursename":"DB2","Coursedur":3})
> db.courseCol.insert({"Coursename":"Oracle","Coursedur":5})
> db.courseCol.insert({"Coursename":"SQLServer","Coursedur":2})
> show collections
courseCol
system.indexes
```


ACID or BASE?
– the case of NoSQL

1. NoSQL - what's in a name
2. NoSQL database arch
3. NoSQL database types
4. ACID or BASE?
5. The CAP theorem
6. Comm. NoSQL databases

MongoDB - basics (cont'd)

1. NoSQL - what's in a name
2. NoSQL database arch
3. NoSQL database types
4. ACID or BASE?
5. The CAP theorem
6. Comm. NoSQL databases

- **Select operations** [sample]

```
> db.courseCol.find({"Coursename":"Oracle"})

{ "_id" : ObjectId("51a089ad17338b27674af7a2"), "Coursename" : "Oracle", "Coursedur" : "5" }

> db.courseCol.find({"Coursename":"Oracle"},{"Coursedur":1});

{ "_id" : ObjectId("51a089ad17338b27674af7a2"), "Coursedur" : "5" }

> db.courseCol.find({Coursedur:{"$gt":2}});

{ "_id" : ObjectId("51a08fc295ce664a0e633cfb"), "Coursename" : "Oracle", "Coursedur" : 5 }
{ "_id" : ObjectId("51a08fd795ce664a0e633cfd"), "Coursename" : "DB2", "Coursedur" : 3 }
```

conditional ops: $gt, $gte, ..., $and, $in, $or, $nor, ...
$limit, $offset, ..., $sort, ...

MongoDB - basics (cont'd)

- **...** [sample]

```
> db.courseCol.insert({"Coursename":"DB2","Coursedur":3, "Instructor" : "Kris"})

> db.courseCol.find({"Coursename":"DB2"});
{ "_id" : ObjectId("51a08fd795ce664a0e633cfd"), "Coursename" : "DB2", "Coursedur" : 3 }
{ "_id" : ObjectId("51a090dd95ce664a0e633cfe"), "Coursename" : "DB2", "Coursedur" : 3,
"Instructor" : "Kris" }

> db.courseCol.find({"Coursename":"DB2"},{"Instructor":1});
{ "_id" : ObjectId("51a08fd795ce664a0e633cfd") }
{ "_id" : ObjectId("51a090dd95ce664a0e633cfe"), "Instructor" : "Kris" }

> db.courseCol.find({"Instructor":"Kris"});
{ "_id" : ObjectId("51a090dd95ce664a0e633cfe"), "Coursename" : "DB2", "Coursedur" : 3,
"Instructor" : "Kris" }
>
```


1. NoSQL - what's in a name
2. NoSQL database arch
3. NoSQL database types
4. ACID or BASE?
5. The CAP theorem
6. Comm. NoSQL databases

MongoDB - basics (cont'd)

- **Update** [sample] - !! default !! - only the first doc is updated

```
> db.courseCol.insert({"Coursename":"DB2","Coursedur":3, "Instructor" : "Kris"})

> db.courseCol.find({"Coursename":"DB2"});
{ "_id" : ObjectId("51a09e6595ce664a0e633cff"), "Coursename" : "DB2", "Coursedur" : 3,
"Instructor" : "Kris" }

> db.courseCol.update({"Coursename":"DB2"},{$set : {"Coursedur":6}})
> db.courseCol.find({"Coursename":"DB2"});
{ "_id" : ObjectId("51a09e6595ce664a0e633cff"), "Coursename" : "DB2", "Coursedur" : 6,
"Instructor" : "Kris" }

> db.courseCol.update({"Coursename":"DB2"},{$set : {"CoursedurUSA":8}})
> db.courseCol.find({"Coursename":"DB2"});
{ "Coursedur" : 6, "CoursedurUSA" : 8, "Coursename" : "DB2", "Instructor" : "Kris", "_id" :
ObjectId("51a09e6595ce664a0e633cff") }
```

alternatives: $inc, $set, $push, $pushall, ...

1. NoSQL - what's in a name
2. NoSQL database arch
3. NoSQL database types
4. ACID or BASE?
5. The CAP theorem
6. Comm. NoSQL databases

MongoDB - basics (cont'd)

- **Remove** [sample]

```
> db.courseCol.remove()

db.courseCol.remove({"Coursedur" : {$lt : 7}})
> db.courseCol.find({"Coursename":"DB2"});
```


1. NoSQL - what's in a name
2. NoSQL database arch
3. NoSQL database types
4. ACID or BASE?
5. The CAP theorem
6. Comm. NoSQL databases

MongoDB - Indexes

- **full index support**
  [index on any attribute (including multiple, list/arrays, nested)]
  [blocking by default]
- **increase query performance**
- **indexes are implemented as “B-Tree” indexes**
  [unique or not] [asc, desc]
  [missing keys: null by default - sparse index]
- **as always: data overhead for inserts and deletes**
- **document TTL in index can be specified**
- **implementation:**

```
> db.courseCol.ensureIndex( {"Coursename" : 1 })
> db.courseCol.getIndexes()
[    {},
     {
          "v" : 1,
          "key" : {
               "Coursename" : 1
          },
          "ns" : "test.courseCol",
          "name" : "Coursename_1"
     }
]
```

ACID or BASE?
– the case of NoSQL

1. NoSQL - what’s in a name
2. NoSQL database arch
3. NoSQL database types
4. ACID or BASE?
5. The CAP theorem
6. Comm. NoSQL databases

MongoDB - Indexes (cont’d)

Limitations:

- **collections : max 64 indexes**
- **index key length max 1024 bytes**
- **queries can only use 1 index**
  [careful with concatenated indexes, careful with negation, careful with regexp]
- **indexes have storage requirements, and impact the performance of writes**
- **in memory sort (no-index) limited to 32 MB**

ACID or BASE? – the case of NoSQL

1. NoSQL - what’s in a name
2. NoSQL database arch
3. NoSQL database types
4. ACID or BASE?
5. The CAP theorem
6. Comm. NoSQL databases

MongoDB - Indexes - explain, caching

```
> db.courseCol.find({"Coursename":"Oracle"}).explain()
{
    "cursor" : "BtreeCursor Coursename_1",
    "isMultiKey" : false,
    "n" : 1,
    "nscannedObjects" : 1,              "nscanned" : 1,
    "nscannedObjectsAllPlans" : 1,   "nscannedAllPlans" : 1,
    "scanAndOrder" : false,              "indexOnly" : false,
    "nYields" : 0,                           "nChunkSkips" : 0,
    "millis" : 0,                            "indexBounds" : {
        "Coursename" : [
            [
                "Oracle",
                "Oracle"
            ]
        ]
    },
    "server" : "everest.abis.be:27017"
}
```

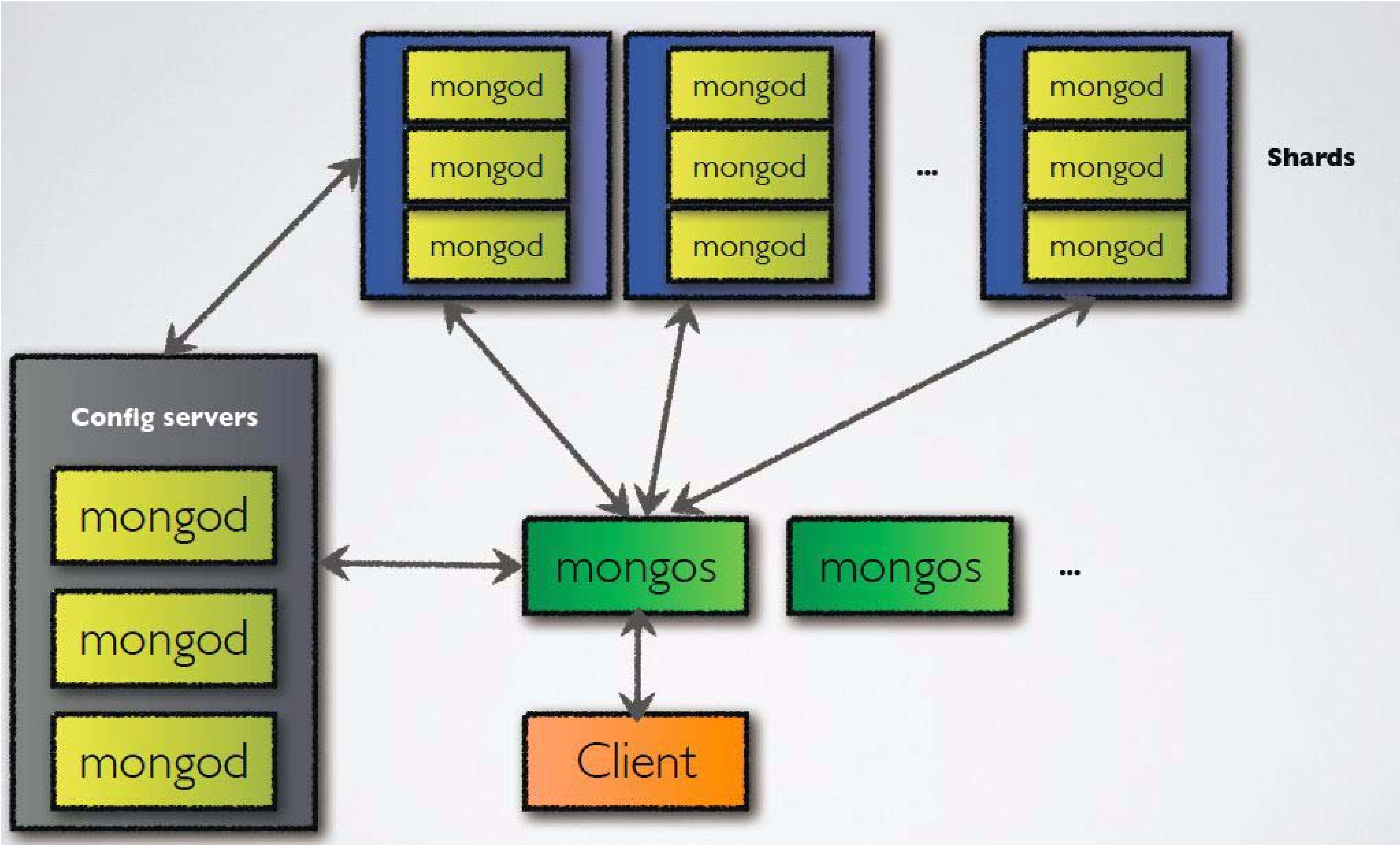
The Query Optimizer:

- **for each query "type", MongoDB periodically tries all useful idxes**
- **aborts the rest as soon as one plan wins**
- **the 'winning plan' is temporarily cached for each "type" of query**

Hints are supported.

1. NoSQL - what's in a name
2. NoSQL database arch
3. NoSQL database types
4. ACID or BASE?
5. The CAP theorem
6. Comm. NoSQL databases

MongoDB - Architecture revisited



- **data is stored on a** shard **in** chunks **of a specific** size [by default 64M]
- **MongoDB automatically** splits **and** migrates **chunks as needed**

1. NoSQL - what's in a name
2. NoSQL database arch
3. NoSQL database types
4. ACID or BASE?
5. The CAP theorem
6. Comm. NoSQL databases

MongoDB - Config servers

- **stored meta data:**
  store cluster chunk ranges and locations
- **can have only 1 or 3**
  [production: use 3 if not ...]
- **2PC commit (not a replica set)**

```
[root@everest bin]# ./mongod --configsvr --port 27019
[root@zion bin]# ./mongod --configsvr --port 27019
[root@bryce bin]# ./mongod --configsvr --port 27019
```

ACID or BASE?
– the case of NoSQL

1. NoSQL - what’s in a name
2. NoSQL database arch
3. NoSQL database types
4. ACID or BASE?
5. The CAP theorem
6. Comm. NoSQL databases

1. NoSQL - what's in a name
2. NoSQL database arch
3. NoSQL database types
4. ACID or BASE?
5. The CAP theorem
6. Comm. NoSQL databases

MongoS

- **acts as a router / balancer**
  installed next to the application server
  routes application requests to the data
  balances chunks
- **no local data (persists to config database)**
- **can have 1 or many**

[root@thegrand bin]# ./mongos --configdb everest:27019, zion:27019, bryce:27019

Start, add, enable shard(ing)

- **start the shard database** [can be an already running, non-sharded db]

[root@th bin]# ./mongod --shardsvr --dbpath /data/db **--port** 27018 **--config** /etc/mongod.conf

- **add the shard definition on MongoS**

> sh.addShard('xenophon:27018')

> sh.addShard('socrates:27018')

- **enable sharding**

> sh.enableSharding("coursedb");

> sh.shardCollection("coursedb.courseCol", {"coursedur":1})

MongoDB Sharding - chunks

- **based on** range-partitioning**!**
- **a chunk is a** section **of a range**
  - **a chunk is** split **once it exceeds the maximum size**
    [configuration, default 64M]

    There is no split point if all documents have the same shard key
  - **chunk split is a logical operation**
    [no data is moved]
  - **if split creates too large of a discrepancy of #chunks across shards: rebalancing starts**
    [configuration parameter]

ACID or BASE?
– the case of NoSQL

1. NoSQL - what's in a name
2. NoSQL database arch
3. NoSQL database types
4. ACID or BASE?
5. The CAP theorem
6. Comm. NoSQL databases

MongoDB Sharding - chunks (cont'd)

- **rebalancing:**
  - **balancer part of MongoS**
  - **migration - balancer lock:**
    - MongoS sends *moveChunk* to source shard
    - source shard notifies destination shard
    - destination shard claims the chunk shard-key range
    - destination shard pulls documents from source shard
    - destination shard updates config server - new location of copied chunks
  - **cleanup:**
    - **source shard deletes moved data**
      [waits for open cursors to either close or time out]
    - **MongoS releases balancer lock after old chunks are deleted**

ACID or BASE?
– the case of NoSQL

1. NoSQL - what's in a name
2. NoSQL database arch
3. NoSQL database types
4. ACID or BASE?
5. The CAP theorem
6. Comm. NoSQL databases

MongoDB Sharding - chunks (cont’d)

Shard key:

- **use a field commonly used in queries**
- **shard key is immutable; shard key values are immutable**
- **shard key requires index on fields contained in key**
- **shard key limited to 512 bytes in size**
- **things to think about:**
  [use your RDBMS skills]
  - cardinality
  - write distribution
  - query isolation
  - data distribution

MongoDB - About Replication

- **Why?**
  - **high availability**
    - if a node fails, another node can step in
    - extra copies of data for recovery
  - **Scaling reads = applications with high read requirements can read from replicas**
- **a** *replica set* **- a set of mongod servers**
  - **minimum of 3**
  - **election of a primary (consensus)**
  - **writes go to primary; secondaries replicate from primary**
- **define and start the replica set -**'named' **set**

mongod --replSet <name>

<name> uses a configuration file, listing the other servers in the set

1. NoSQL - what's in a name
2. NoSQL database arch
3. NoSQL database types
4. ACID or BASE?
5. The CAP theorem
6. Comm. NoSQL databases

MongoDB - About Replication - oplog

- **change operations are written to the** oplog **of the primary**
  - **a capped collection**
  - **must have enough space to allow new secondaries to catch up after copying from a primary**
  - **must have enough space to cope with any applicable** slaveDelay
  - **secondaries query the primary's oplog and apply what they find**

ACID or BASE?
– the case of NoSQL

1. NoSQL - what's in a name
2. NoSQL database arch
3. NoSQL database types
4. ACID or BASE?
5. The CAP theorem
6. Comm. NoSQL databases

MongoDB - About Replication - failover

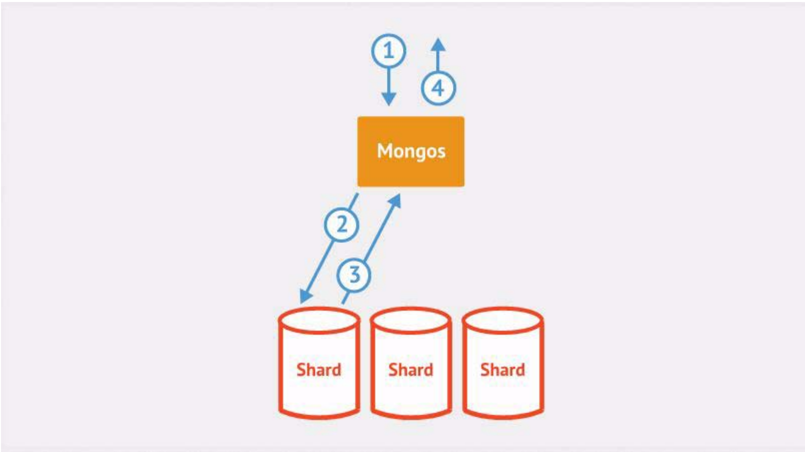
1. NoSQL - what's in a name
2. NoSQL database arch
3. NoSQL database types
4. ACID or BASE?
5. The CAP theorem
6. Comm. NoSQL databases

Failover:

- **replica set members monitor other set members [heartbeats]**
- **if primary not reachable, a new one is elected**
- **the secondary with the** most up-to-date oplog **is chosen** [priority can be set to influence election; secondaries can be banned from becoming primary]
- **if, after election, a secondary has changes not on the new primary, those are** undone**, and moved aside**
- **if you require a guarantee, ensure data is written to a majority of the replica set**

MongoDB - Request Routing

Targeted Queries

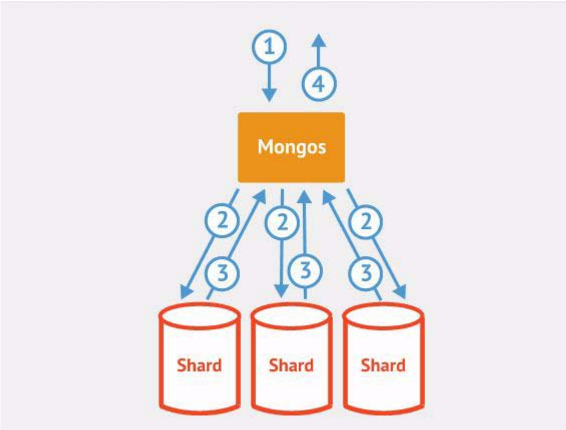


1. NoSQL - what's in a name
2. NoSQL database arch
3. NoSQL database types
4. ACID or BASE?
5. The CAP theorem
6. Comm. NoSQL databases

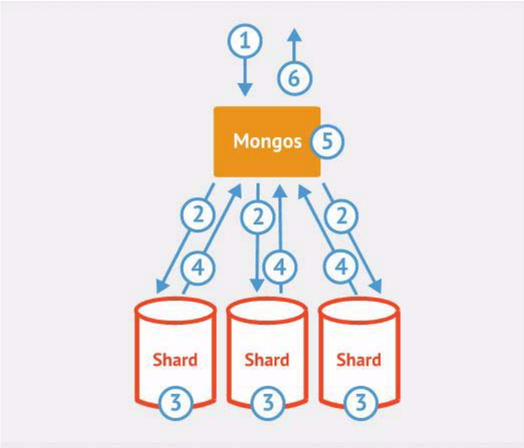
MongoDB - Request Routing (cont'd)

1. NoSQL - what's in a name
2. NoSQL database arch
3. NoSQL database types
4. ACID or BASE?
5. The CAP theorem
6. Comm. NoSQL databases

Scatter Gather Queries



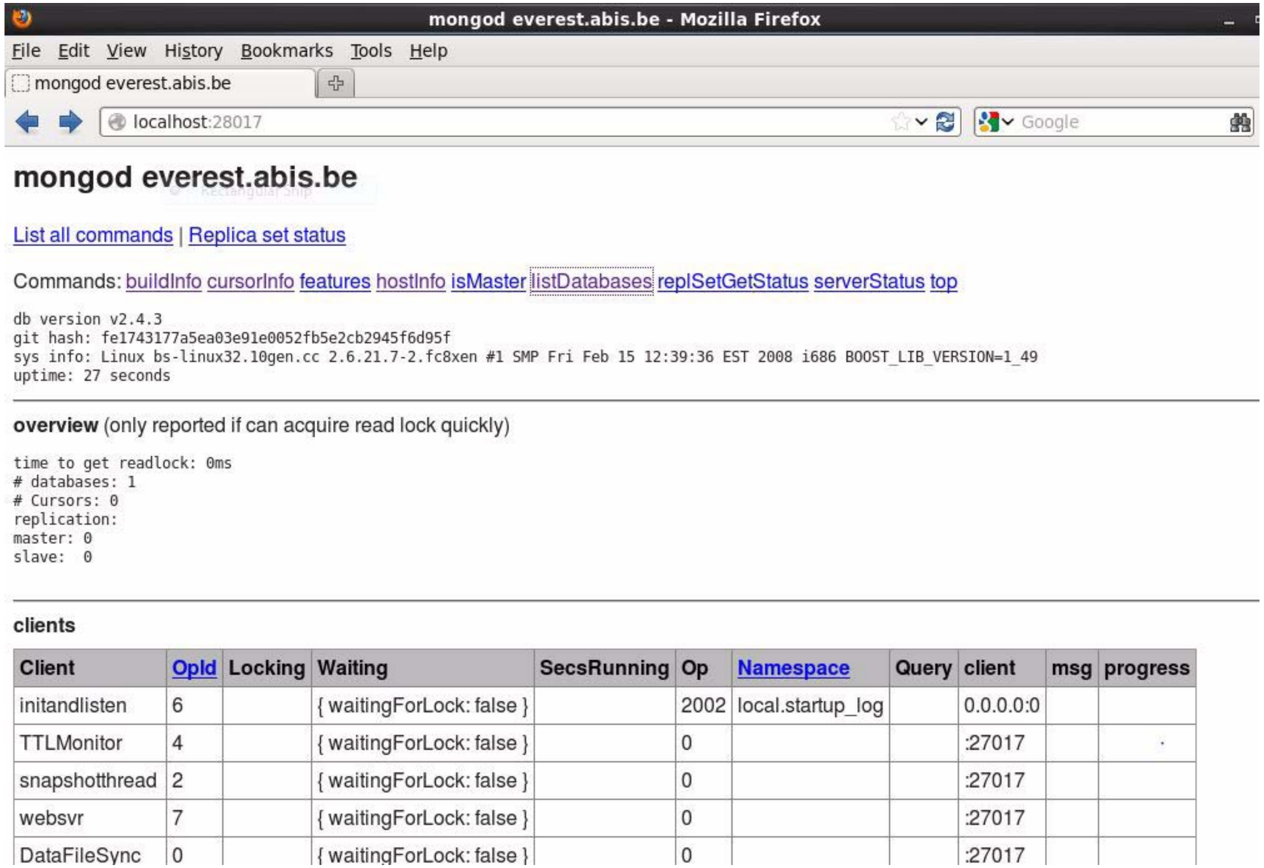
Scatter Gather Queries with Sort



MongoDB - REST interface

- **mongod provides a basic REST interface**
  [-- rest, default port 28017]

[root@everest bin]# ./mongod --dbpath /data/db --port 27017 --config /etc/mongod.conf --rest



mongod everest.abis.be

List all commands | Replica set status

Commands: buildInfo cursorInfo features hostInfo isMaster listDatabases replSetGetStatus serverStatus top

```
db version v2.4.3
git hash: fe1743177a5ea03e91e0052fb5e2cb2945f6d95f
sys info: Linux bs-linux32.10gen.cc 2.6.21.7-2.fc8xen #1 SMP Fri Feb 15 12:39:36 EST 2008 i686 BOOST_LIB_VERSION=1_49
uptime: 27 seconds
```

overview (only reported if can acquire read lock quickly)

```
time to get readlock: 0ms
# databases: 1
# Cursors: 0
replication:
master: 0
slave:  0
```

clients

Client	OpId	Locking	Waiting	SecsRunning	Op	Namespace	Query	client	msg	progress
initandlisten	6		{ waitingForLock: false }		2002	local.startup_log		0.0.0.0:0		
TTLMonitor	4		{ waitingForLock: false }		0			:27017		
snapshotthread	2		{ waitingForLock: false }		0			:27017		
websvr	7		{ waitingForLock: false }		0			:27017		
DataFileSync	0		{ waitingForLock: false }		0			:27017		

1. NoSQL - what's in a name
2. NoSQL database arch
3. NoSQL database types
4. ACID or BASE?
5. The CAP theorem
6. Comm. NoSQL databases

MongoDB - Other features...

- **GridFS**
  - **store files of any size** (exceeding binary storage data max size)
  - **GridFS leverages existing** replication **or** autosharding **that has been set up**
- **Map Reduce**
  - **queries** [jscript function] **run in all shards parallel** [one thread per node]
  - **flexible aggregation and data processing**
  - **often used**
- **Geospatial Indexing**

  two-dimensional indexing for location-based queries
  [find objects based on location? Find closest n items to x]

```
db.map.insert({location : {longitude : -40, latitude : 78}})
db.map.find({location : {$near : [ -30, 70]}})
```

ACID or BASE?
– the case of NoSQL

1. NoSQL - what's in a name
2. NoSQL database arch
3. NoSQL database types
4. ACID or BASE?
5. The CAP theorem
6. Comm. NoSQL databases

Questions, remarks, feedback, ... ?

TRAINING & CONSULTING

Thank you!

Peter Vanroose

ABIS Training & Consulting

pvanroose@abis.be

ACID or BASE?
– the case of NoSQL

1. NoSQL - what's in a name
2. NoSQL database arch
3. NoSQL database types
4. ACID or BASE?
5. The CAP theorem
6. Comm. NoSQL databases