Part I: Scalable Data Access Models

MASTER SEP 2022-2023

Bart.Lamiroy@univ-reims.fr

Synchronization

Lock/semaphore synchronization

Centralized clock-based synchronization

NTP

Loose synchronization and conflict detection

- Vector Clocks
- Interval Tree Clocks
- Bloom Clocks

Vector clocks

Logical Relative Time

No Central Synchronization

Detects causality vs. concurrency conflict

General idea

- n Processes
- Each process *p* labels its local events with a n-dimensional timestamp called Vector Clock VC = $[c_1, c_2, \ldots c_n]$
- Each c_k contains the last known event of process *k* by process *p*.

Vector Clocks

Algorithm

- Vector Clocks are all set to *[0, 0, … 0]*
- When an event occurs at process *p* :
  - *p* increments c_p
  - *p* sends an event notification and its vector clock to all other processes
- When p receives an event notification from another process with Vector Clock *[* c_1, c_2, … c_n *]* it updates its local Vector *Clock [* l_1, l_2, … l_n*]* such that

$$l_i = max(l_i, c_i)$$

$$l_p = l_p + 1$$

Each entry c_k is a count of events in process *k* that causally precede event e

Vector Clocks

Vector Clocks define a partial order on events

If $VC_1 < VC_2$ then VC_2 is causally dependent on VC_1

(*i.e.* VC_1 strictly precedes VC_2)

Else there is a concurrency conflict between VC_1 and VC_2

(*i.e.* both chains of events rely mutually non-synchronized events)

$VC_1 < VC_2$ *iff* $c_i^{VC1} <= c_i^{VC2}$ for all *i* and there exists at least one *j* for which $c_j^{VC1} <= c_j^{VC2}$

UNIVERSITÉ DE REIMS CHAMPAGNE-ARDENNE

Vector Clocks : limitations

Fixed value of n

Inadequate for highly dynamic distributed setups

- Many appearing and disappearing nodes
- High volume of events and signaling

Adaptations :

- Interval Tree Clocks (http://gsd.di.uminho.pt/members/cbm/ps/itc2008.pdf)
- Bloom Clocks (https://arxiv.org/pdf/1905.13064v1.pdf)

UNIVERSITÉ DE REIMS CHAMPAGNE-ARDENNE

Assignment

Pick one of the following topics:

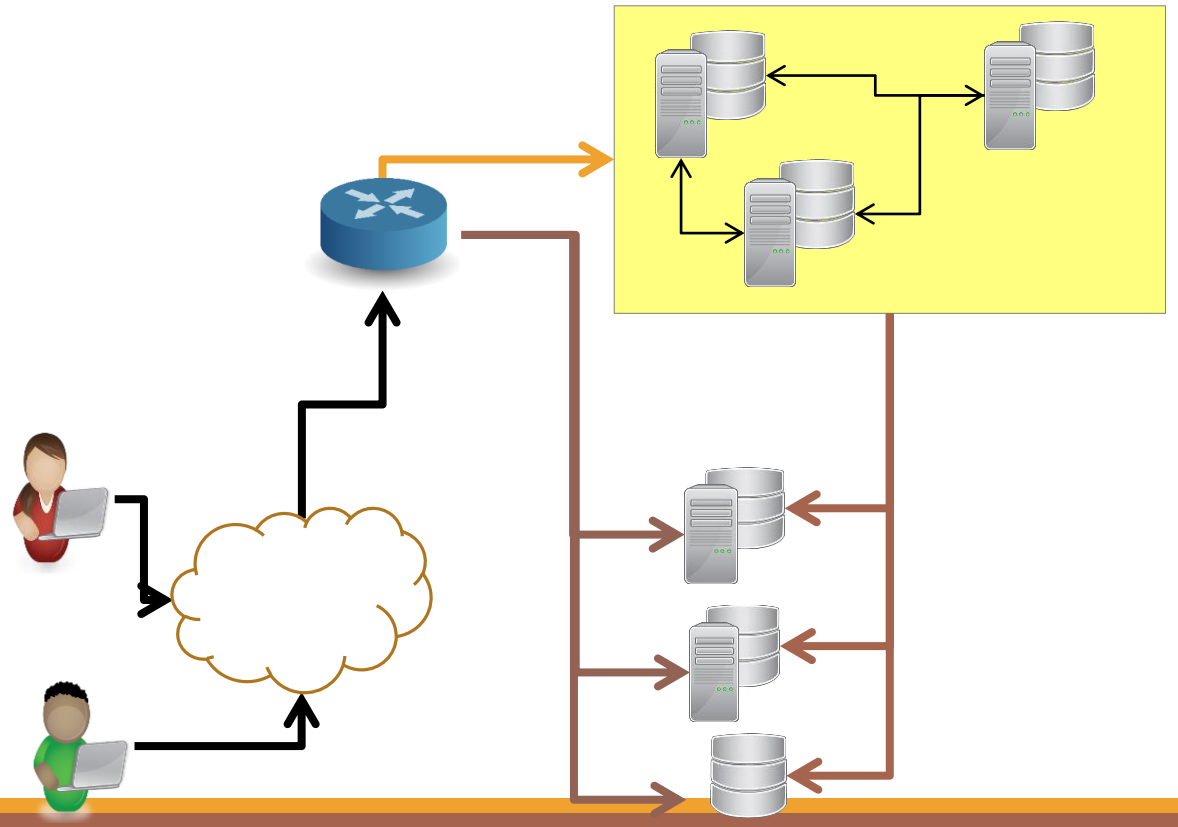
- Interval Tree Clocks
- Bloom Clocks
- Paxos
- Gossip Protocol
- Spread Toolkit
- Corosync

Submit a 4 page technical analysis of the studied protocol

Availability

How to Make Data Available in a Distributed Setup?

- *Where is my data ?*
- *Efficient access ?*
- *Failover ?*



Availability

How to Make Data Available in a Distributed Setup?

- Data split over multiple servers
- Servers can fail
- Servers added on the fly

How to allow efficient access to the data ?

How to maintain/guarantee availability ?

How to maintain coherence ?

What about the CAP theorem ?

Availability

Data Replication and Consistent Hashing

- Data split over multiple servers
- Servers can fail
- Servers added on the fly

How to allow efficient access to the data ?

How to maintain/guarantee availability ?

Key-Value Stores

Access to data through primary keys only

Query: key → record of data

Example: Amazon (Dynamo)

- Best seller list
- Shopping cart
- Customer preferences
- Session manager
- Product catalog
- Sales Rank

Document Stores

Extension of Key-Value Stores

- Value is semi-structured
- Often in standard encoding (XML, JSON …)
- Example (source Wikipedia):

```
{
      FirstName: "Jonathan",
      Address: "15 Wanamassa Point Road",
      Children: [
            {Name: "Michael", Age: 10},
            {Name: "Jennifer", Age: 8},
            {Name: "Samantha", Age: 5},
            {Name: "Elena", Age: 2} ]
}
```

- Query & Secondary indexing capabilities vary between interpretations

Document Stores

Examples

- Cassandra (Apache, Java, JSON)
- Couchbase (Apache, C/C++ Erlang, JSON)
- MongoDB (MongoDB, BSON)

Column Stores

Linked to relational data representations

- Each table column is 1 (virtual) file
- Better compression
- Optimized access for grouped operations
- Easier indexing

Examples:

- BigTable (Google)
- Hadoop (Apache)

Graph Database – Tuple Stores

Generally more normalized than other NoSQL

Mainly RDF oriented & SPARQL query language

Tuned to represent a R b or R(a,b) relations

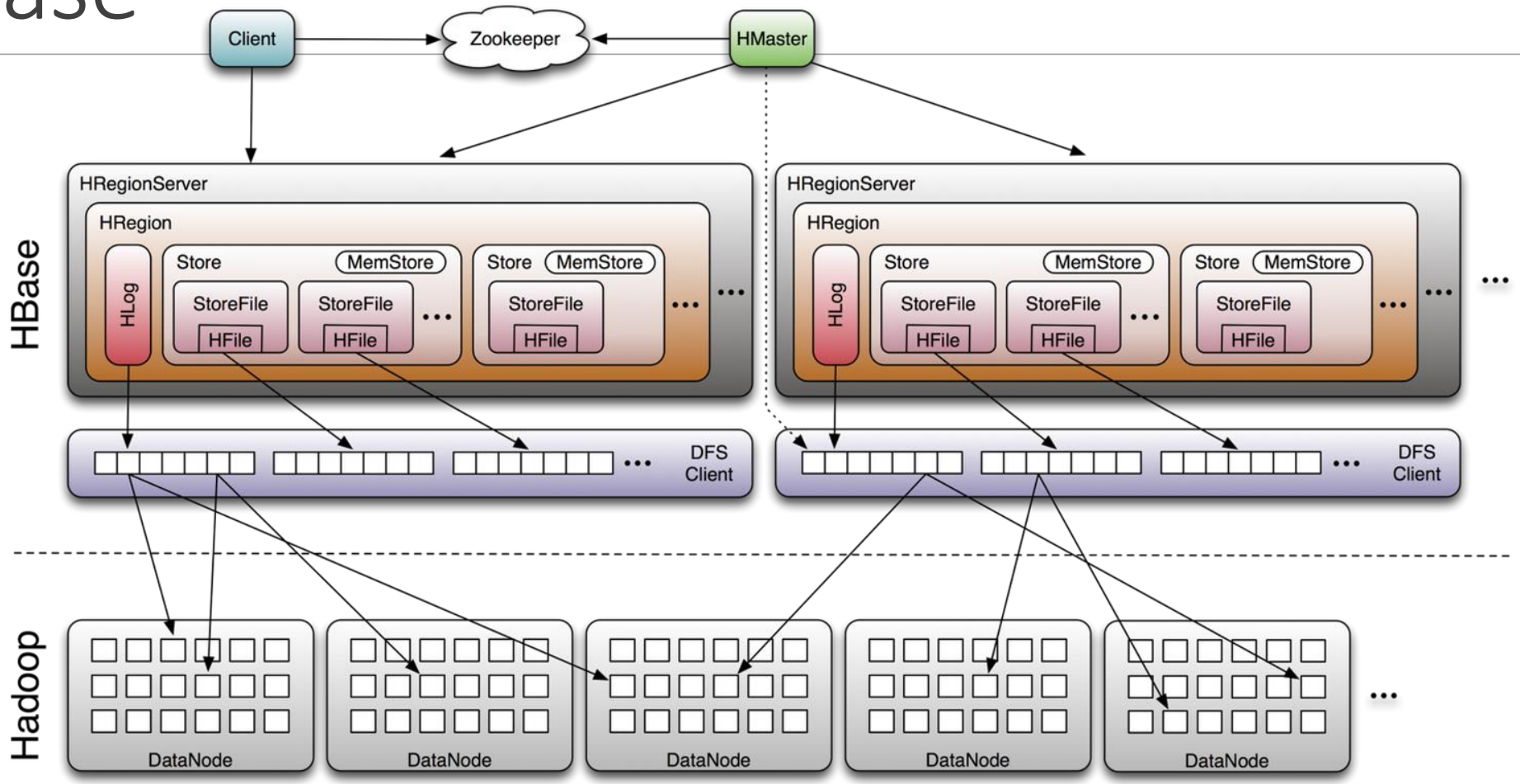
Examples

- Virtuoso
- AllegroGraph
- Giraph

HBase

- Tables are sharded horizontally into Regions
- Regions create vertical Stores for each column family
- Stores are stored into HFiles

HBase



Source: http://www.larsgeorge.com