CLOUD COMPUTING APPLICATIONS

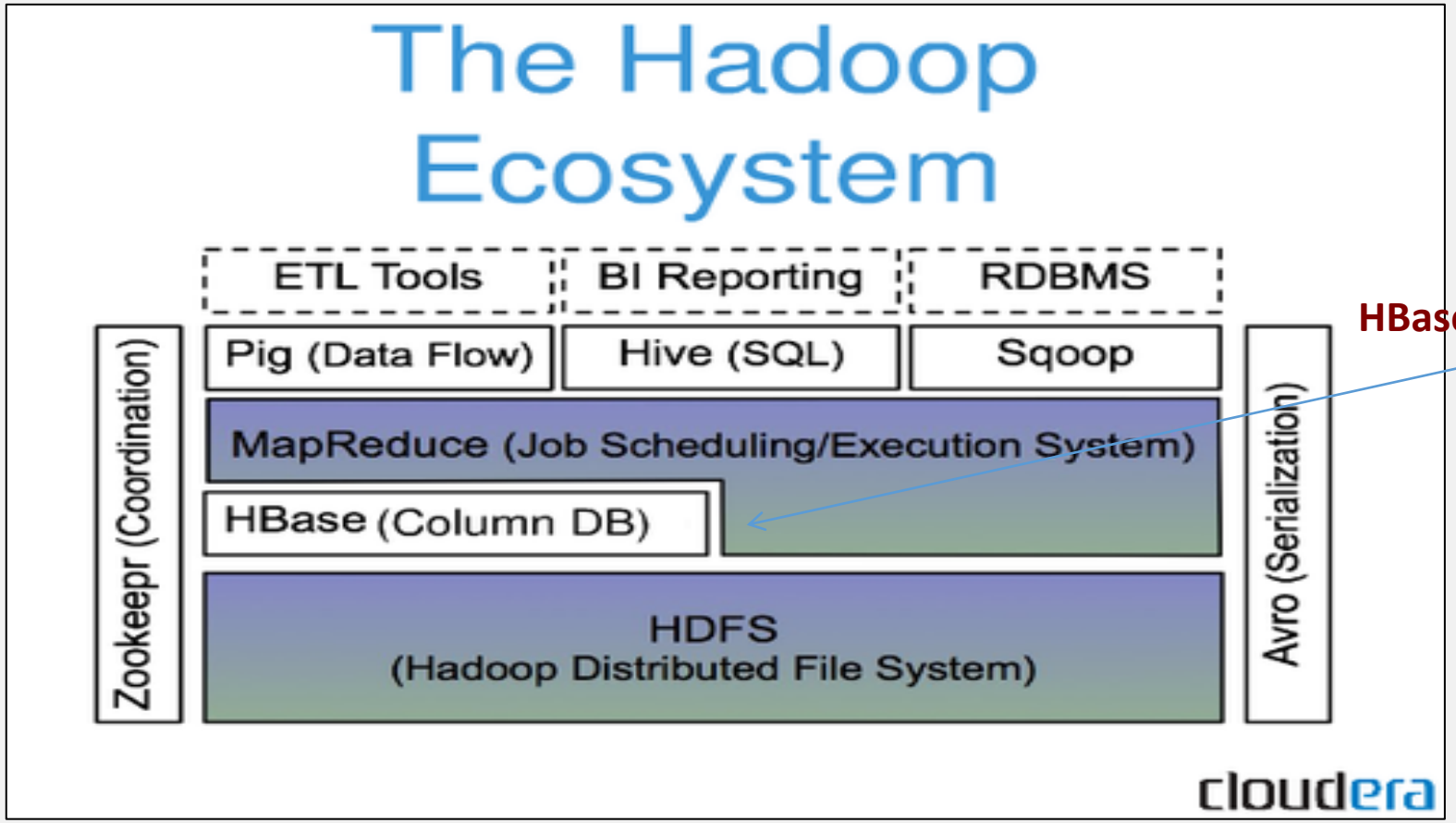
HBase Usage API

Prof. Roy Campbell

HBase: Overview

- HBase is a distributed column-oriented data store built on top of HDFS
- HBase is an Apache open source project whose goal is to provide storage for Hadoop Distributed Computing
- Data is logically organized into tables, rows, and columns

HBase:
Part of Hadoop's Ecosystem



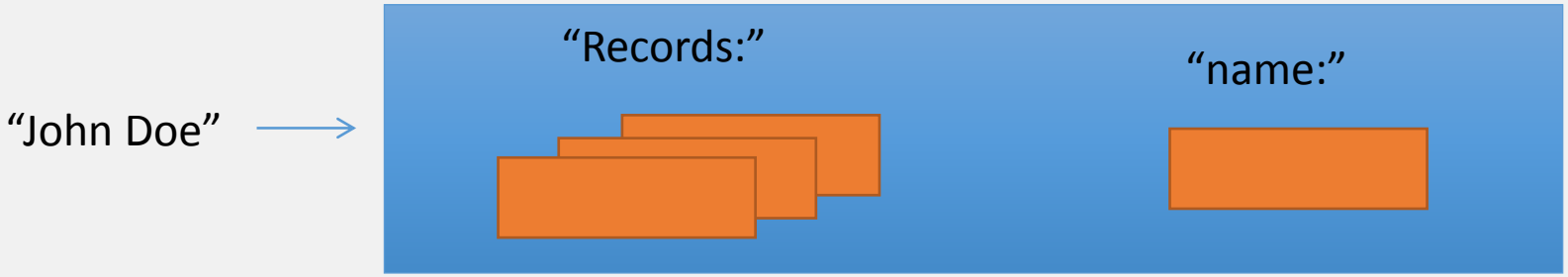
HBase is built on top of HDFS

HBase files are internally stored in HDFS

HBase vs. HDFS

- ***HDFS*** is good for batch processing (scans over big files)
  - Not good for record lookup
  - Not good for incremental addition of small batches
  - Not good for updates
- ***HBase*** addresses the above points
  - Fast record lookup
  - Support for record-level insertion
  - Support for updates

Data Model



- A table in Bigtable is a sparse, distributed, persistent multidimensional sorted map
- Map indexed by a row key, column key, and a timestamp
  - (row:string, column:string, time:int64) → uninterpreted byte array
- Supports lookups, inserts, deletes
  - Single row transactions only

Notes on Data Model

- HBase schema consists of several ***Tables***
- Each table consists of a set of ***Column Families***
  - Columns are not part of the schema
- HBase has ***Dynamic Columns***
  - Because column names are encoded inside the cells
  - Different cells can have different columns

"Roles" column family has different columns in different cells

Row key	Data
cutting	info: { 'height': '9ft', 'state': 'CA' } roles: { 'ASF': 'Director', 'Hadoop': 'Founder' }
tlipcon	info: { 'height': '5ft7, 'state': 'CA' } roles: { 'Hadoop': 'Committer'@ts=2010, 'Hadoop': 'PMC'@ts=2011, 'Hive': 'Contributor' }

Notes on Data Model

- The ***version number*** can be user-supplied
  - Does not have to be inserted in increasing order
  - Version numbers are unique within each key
- Table can be very sparse
  - Many cells are empty
- ***Keys*** are indexed as the primary key

Has two columns
[cnnsi.com & my.look.ca]

Row Key	Time Stamp	ColumnFamily contents	ColumnFamily anchor
"com.cnn.www"	t9		anchor:cnnsi.com = "CNN"
"com.cnn.www"	t8		anchor:my.look.ca = "CNN.com"
"com.cnn.www"	t6	contents:html = "<html>..."	
"com.cnn.www"	t5	contents:html = "<html>..."	
"com.cnn.www"	t3	contents:html = "<html>..."	

Rows and Columns

- Rows maintained in sorted lexicographic order
  - Applications can exploit this property for efficient row scans
  - Row ranges dynamically partitioned into tablets
- Columns grouped into column families
  - Column key = *family:qualifier*
  - Column families provide locality hints
  - Unbounded number of columns

HBase lookup example

```
Scanner scanner (T);
ScanStream *stream;
stream = scanner.FetchColumnFamily(“name”);
stream -> SetReturnAllVersions();
//Filter return sets using regex
scaanner.lookup (“John Doe”);
for (; !stream -> Done(); stream -> Next()) {
   printf (“%s %s %s \n”,
          scanner.RowName(),
          stream -> ColumnName(),
          stream – Value());
}
```

HBase lookup example

```
Table *T = OpenOrDie ("/hbase/myTable");

RowMutation rowMut (T, "John Doe");
rowMut.Set ("name:Jane Roe", "NAMES");
rowMut.Delete("name:Jack Public");
Operation op;
Apply (&op, &rowMut);
```