Design of Data Models and Distribution Systems based on Big Data ecosystems HPCC and Hadoop with HBase

Braunmiller Nina
Big Data Engineering
Johannes Kepler University Linz
June 2023

Table of contents

1. HPCC

- **Definition:**
  - **H**igh-**P**erformance **C**omputing **C**luster
  - An ecosystem for processing Big Data relying on the concept of graph based clustering
- **Main idea:**
  - Handling Big Data by data modification during processing in memory instead of disks
  - Create graph like data clusters
- **Thor:**
  - Data cleaning, standardizing, enrichment
  - Information extraction
  - Indexing
- **Roxie:**
  - Querying process
  - Observing data patterns

1. HPCC

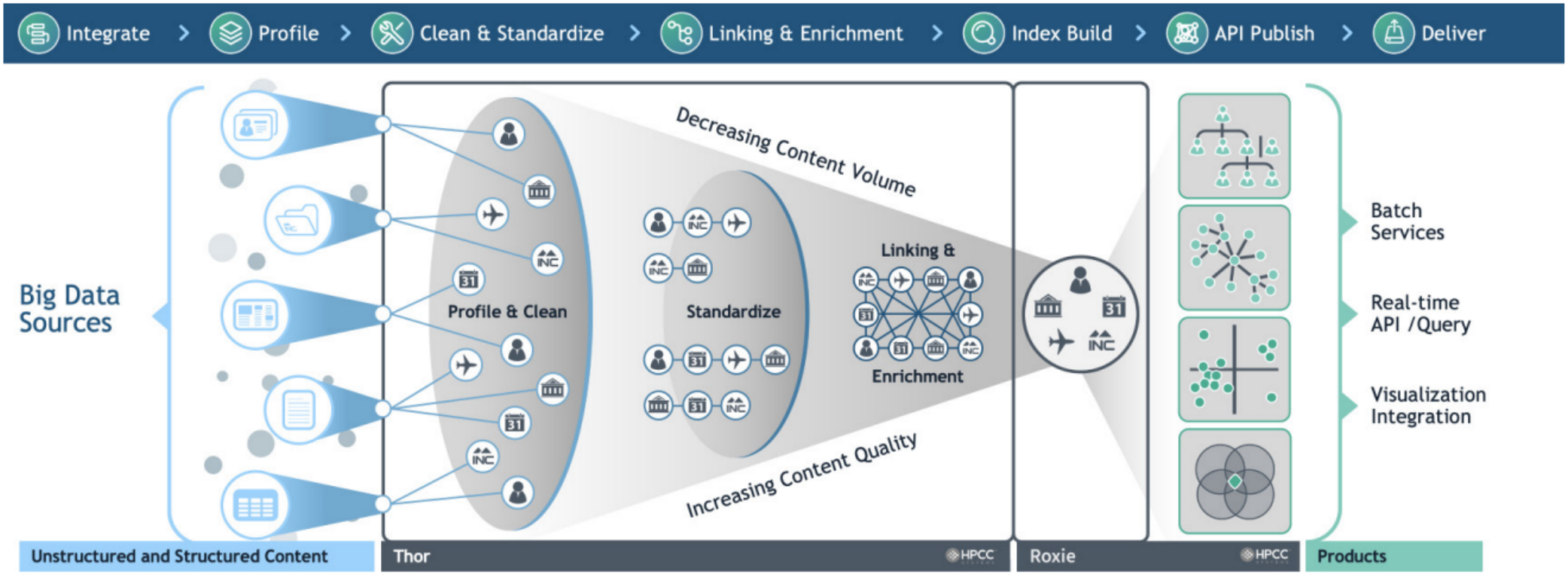


Fig. 1 Data modification in the course of processing.
Source: Taming the Data Lake: The HPCC Systems® Open Source Big Data Platform. (2022). https://cdn.hpccsystems.com/whitepapers/wp_introduction_HPCC.pdf

1. HPCC

- ✔ No data schema needed
- ✔ Join operations possible
- ✔ Own data-centric programming language: ECL
- ✔ Parallel computing over several servers

- ✗ Is beaten by Hadoop for 270 million sample units
- ✗ Became later open-source → less literature

2. Hadoop

- **Definition**
  - Ecosystem for handling Big Data
  - Parallel processing
  - Idea of using lots of different technologies

2. Hadoop

- **Storage organization:**

  Hadoop Distributed File System (HDFS)

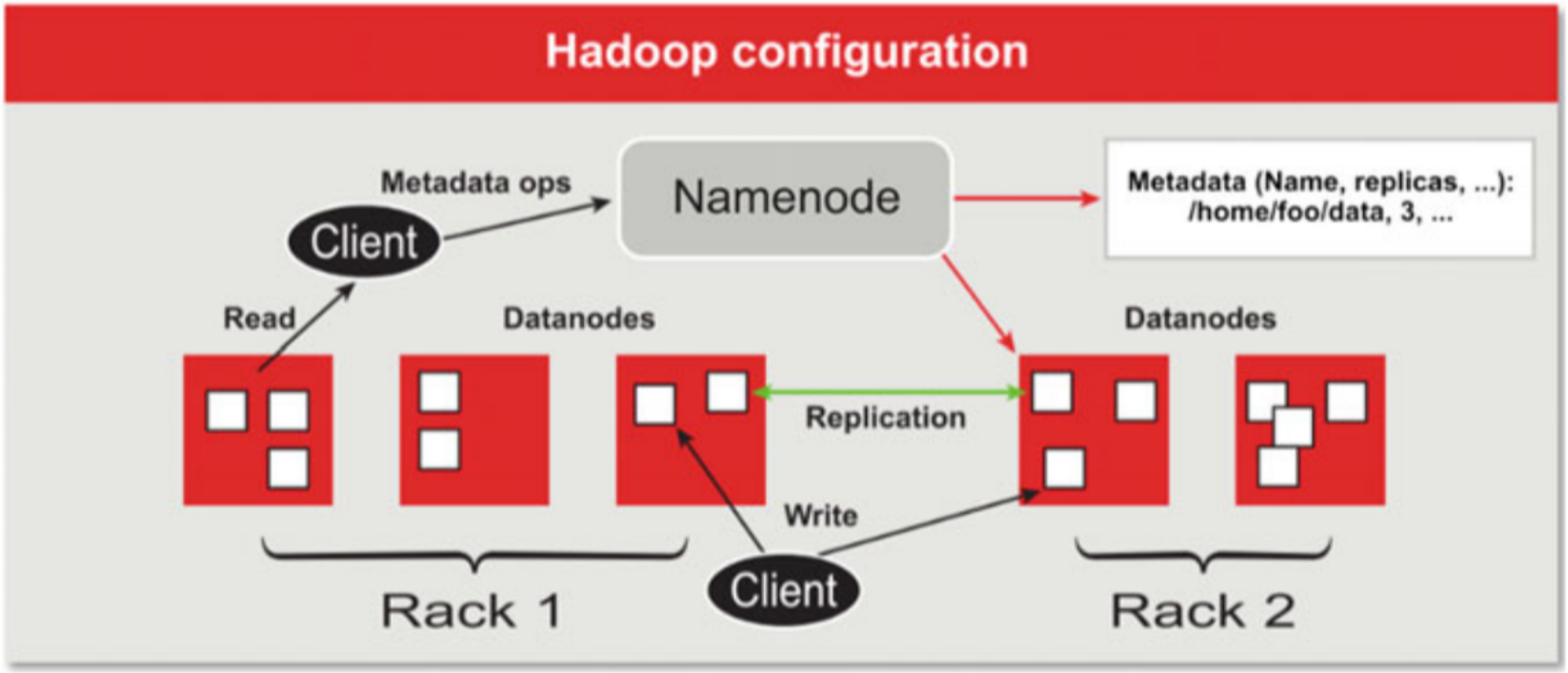


Fig. 2 Hadoop's storage architecture.
Source: Wiktorski, T. (2019). Hadoop Architecture. In X. Wu & L. C. Jain (Eds.), *Data-intensive Systems Principles and Fundamentals using Hadoop and Spark: Advanced Information and KnowledgeProcessing* (pp. 51–61). Springer Nature Switzerland AG. https://doi.org/https://doi.org/10.1007/978-3-030-04603-3

2. Hadoop

- **Query processing:**
  MapReduction

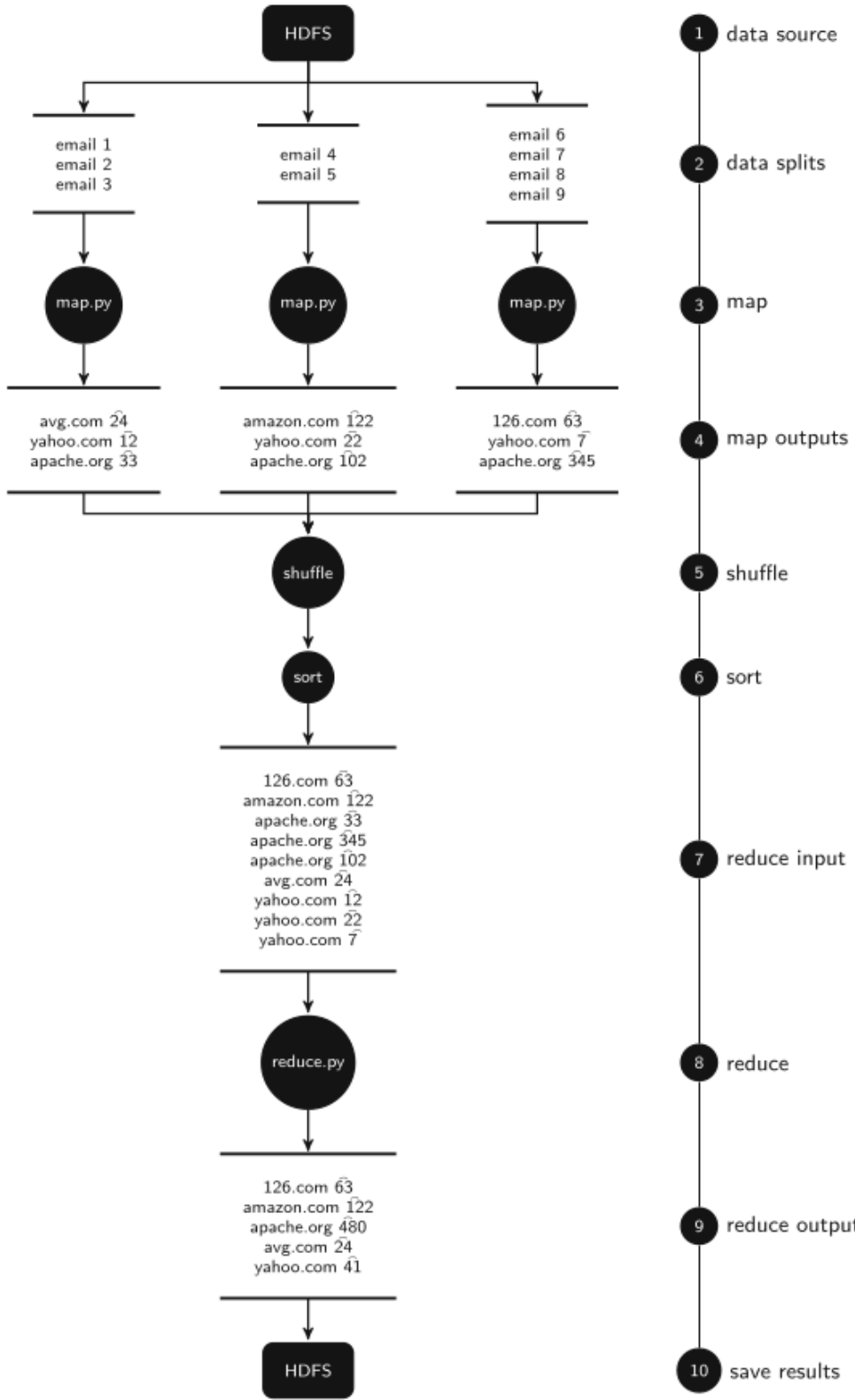


Fig. 3 An example for the MapReduce process.
Source: Wiktorski, T. (2019). Hadoop Architecture. In X. Wu & L. C. Jain (Eds.), *Data-intensive Systems Principles and Fundamentals using Hadoop and Spark: Advanced Information and KnowledgeProcessing* (pp. 51–61). Springer Nature Switzerland AG. https://doi.org/https://doi.org/10.1007/978-3-030-04603-3

3. Hadoop

- ✔ Big ecosystem with lots of technologies for different usage cases, e.g. machine learning
- ✔ Parallel data processing on storage and software level
- ✔ Simple but efficient underlying structure
- ✔ Is sped up by the usage of HBase as replacement to MySQL

- ✗ Different technologies with different programming languages, traits, and operating traits

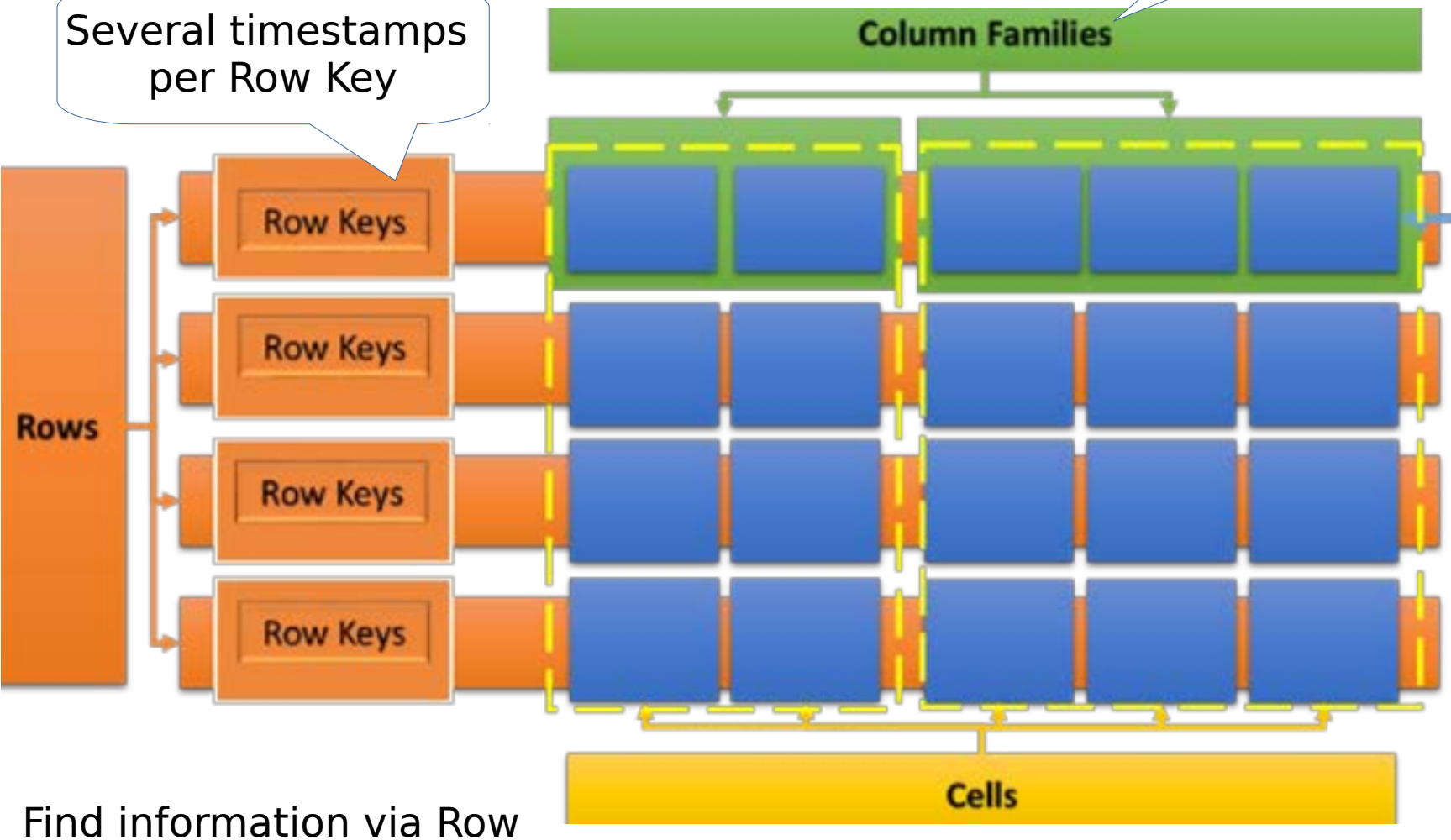
3. HBase

- **Definition:**
  - NoSQL database to handle Big Data
  - Column-based working with key-value pairs
  - Encoding in byte-array which can represent, strings, floats, ...

3.1 Architecture

One file per Column Family (CF)

Several timestamps per Row Key



Find information via Row Key, timestamp, and ColumnFamily:ColumnName

Fig. 4 The architecture of an HBase table .
Source: Shriparv, S. (2014). The Storage, Structure Layout, and Data Model of HBase. In A. Hussain, K. Colaco, P. Bisht, P. Kadam, J. Dharmaraj, & S. Mukherjee (Eds.), *Learning HBase: Learn the fundamentals of HBase administration and development with the help of real-time scenarios* (pp. 121–144). Packt Publishing. https://www.programmer-books.com/wp-content/uploads/2019/12/Learning-HBase.pdf

3.2 HBase – Design choices

- Aim of increasing performance for read and write request
- Fitting design to already given data
- Implementing Joins

3.2.1 HBase – Design choices – Column Family

- Single CF has an advantage compared to approach with multiple CFs.

  Reason: Only one file has to be searched through. Also writing advantage.

  → However, depending on the situation also multi-CF can be benficial, e.g. for very long columns and for certain querying techniques

  → also the number of built up CFs is important. Shouldn‘t be too much

- **Genetic Algorithm:** cost function with different weighted elements evaluates each CF combination. Train/Test/Evaluation sets.

3.2.2 HBase – Design choices – Key

- Keep column/CF/row key names short
- **Hot spot problem:** data load is not evenly distributed. New data is sorted to the latest region. Slower querying.

  Solutions:

  - Sort the new data according to timestamps. 60 regions for 60 minutes

    key = **timedRegionNumber** + random8bitString + currentTimestamp + uniqueDataRecordID
  - **facility information access protocol:**

    time-based data distirbution.

    Each entity with own PointID.

    Hashing used for region building.

    key = HashBinRegion + SerialNumber + LatestTimestampOfSequence
  - Consider frequenct query scenarios in the key design. E.g. have a watch at customer behavior.

    Secondary search key/index can help in finding values for second most common query type.

3.2.3 HBase – Design choices – Data Partitioning

- **Regions:** areas of rows. Those areas are sorted to different servers/DataNodes.

  → Parallel processing makes it possible to speed up reading and writing requests. MapReduce can make in the Mappers use of it when no overload (too many regions)
- Vertical partitioning delays region building → disadvantage when too strong
- **High Efficient Distributed Storage Middleware:**
  - Aim of data load balancing
  - Storage and query design. Redis database as cache for frequently visited data
  - Region size design
  - Data balancing between different servers. Heavy used servers get relaxed, unused servers get heavier load/more used regions.
  - Faster read and write processing

3.2.4 HBase – Design choices – Query design

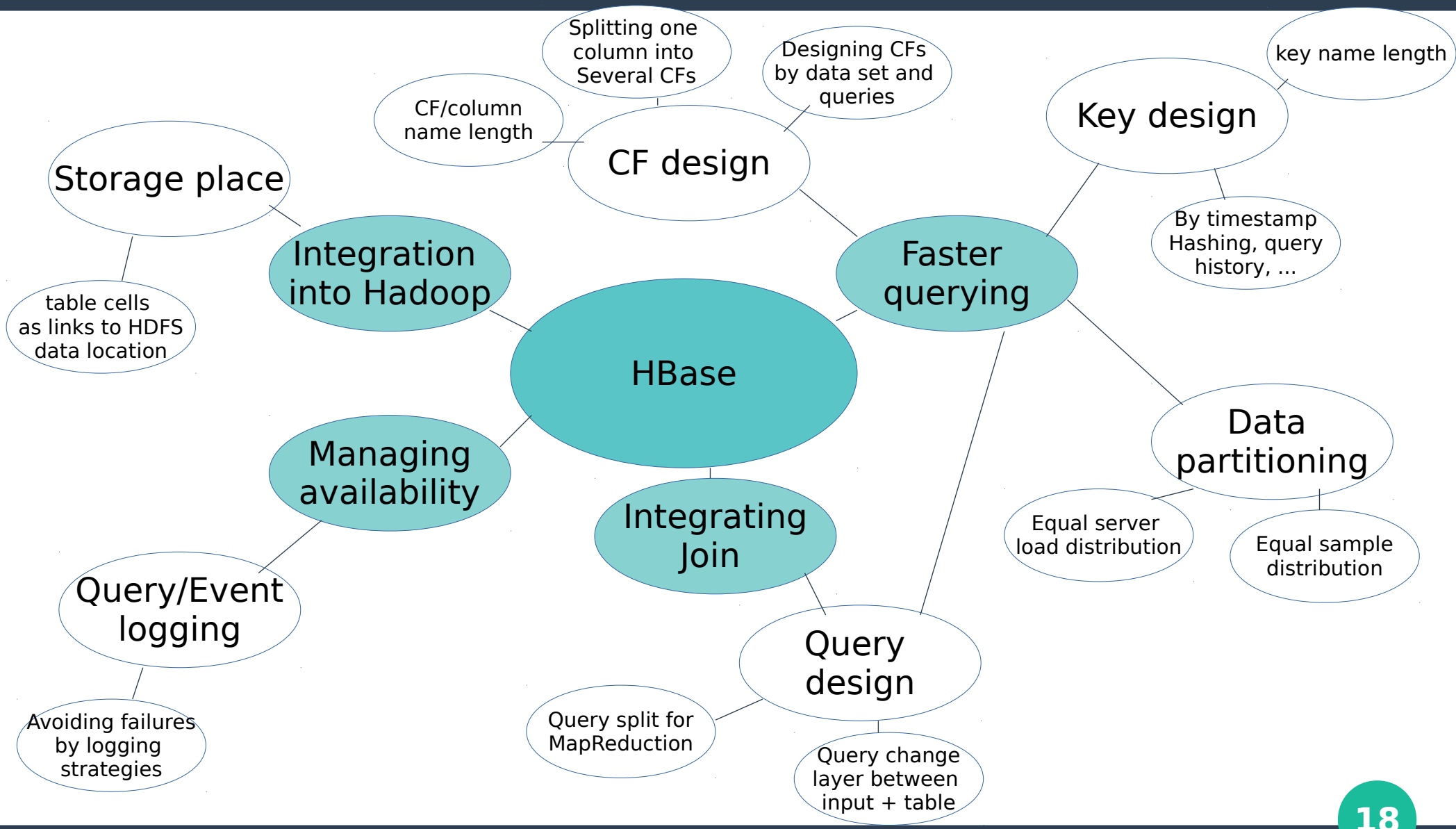
- **Faster processing**
  - "Indexed Random Access" (IRA): building regions based on secondary index. Splitting up query and applying Mappers in parallel to them.
  - "Full Source Scan" (FSS): parallel processing of table. Full table scan

    → faster than IRA
- **Handling Joins**
  - Make use of MapReduce. Different strategies in handling Join queries.
  - Split query by attributes

3.2.5 HBase – Design choices – Conclusion

- Heavy research
- Each design area is connected with others
- Papers often have similar approaches but implement them in a completely different way matching up the requirements of certain query types and data set characteristics.
- Successes in performance can be observed but those are always limited by different circumstances like the data set size, request type, and so on.

3.3 HBase Research map



Die Referenzen sind die gleichen wie in der Seminararbeit. Da die Präsentation nur die Seminararbeit überblickend erklärt, habe ich hier kein Referenzverzeichnis angegeben. Es kann aber gerne hinzugefügt werden.

Thank you for your attention!