Extreme Computing

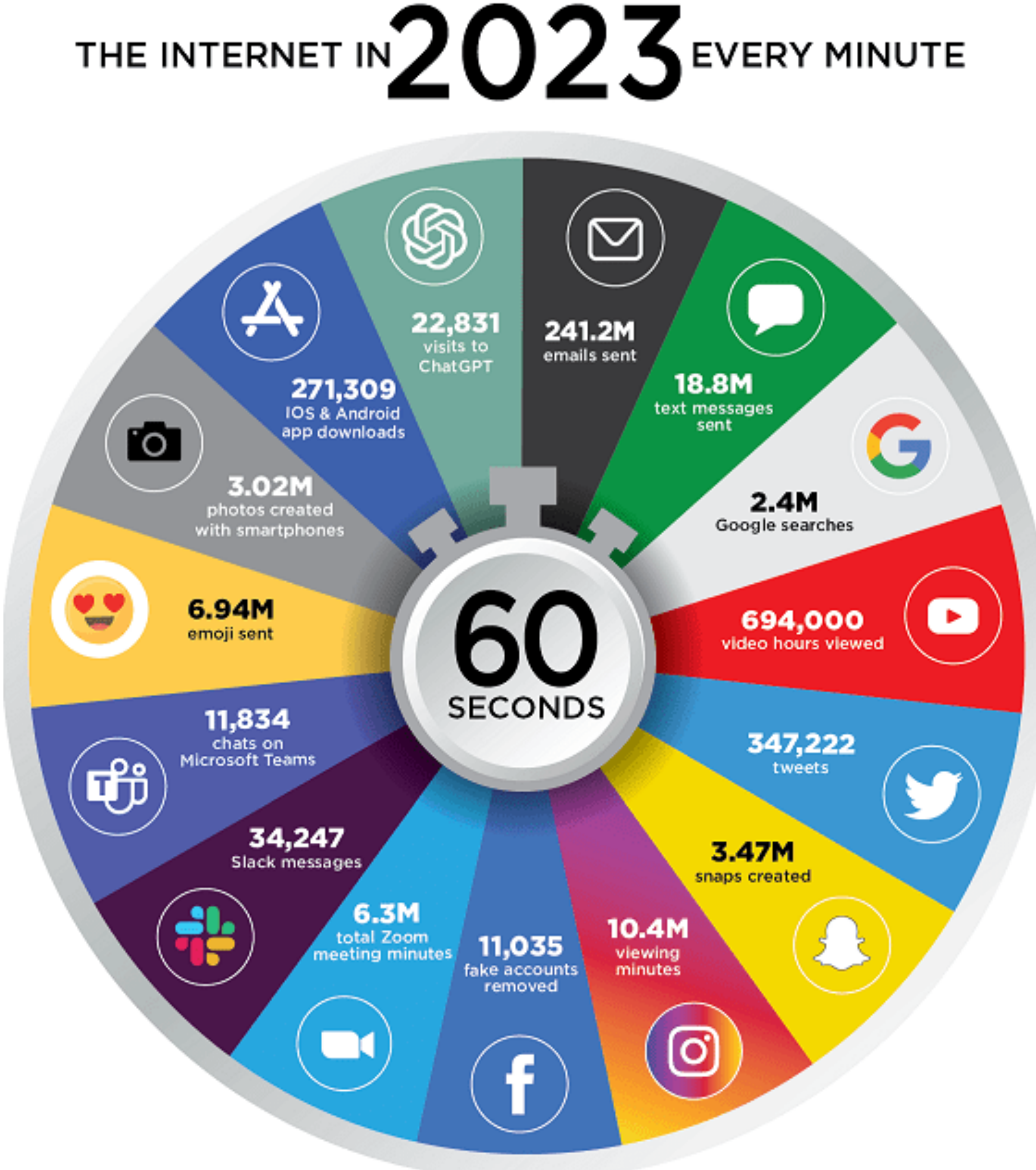
Distributed Data-Parallel Programming

THE UNIVERSITY *of* EDINBURGH

Amir Shaikhha, Fall 2023

Part 1

Programming Models/Languages



https://ediscoverytoday.com/2023/04/20/2023-internet-minute-infographic-by-ediscovery-today-and-ltmg-ediscovery-trends/

Mainstream Languages for Data Scientists

Mainstream Languages for Data Scientists (cont.)

Pros

✓Rapid Development

✓Large community

Cons

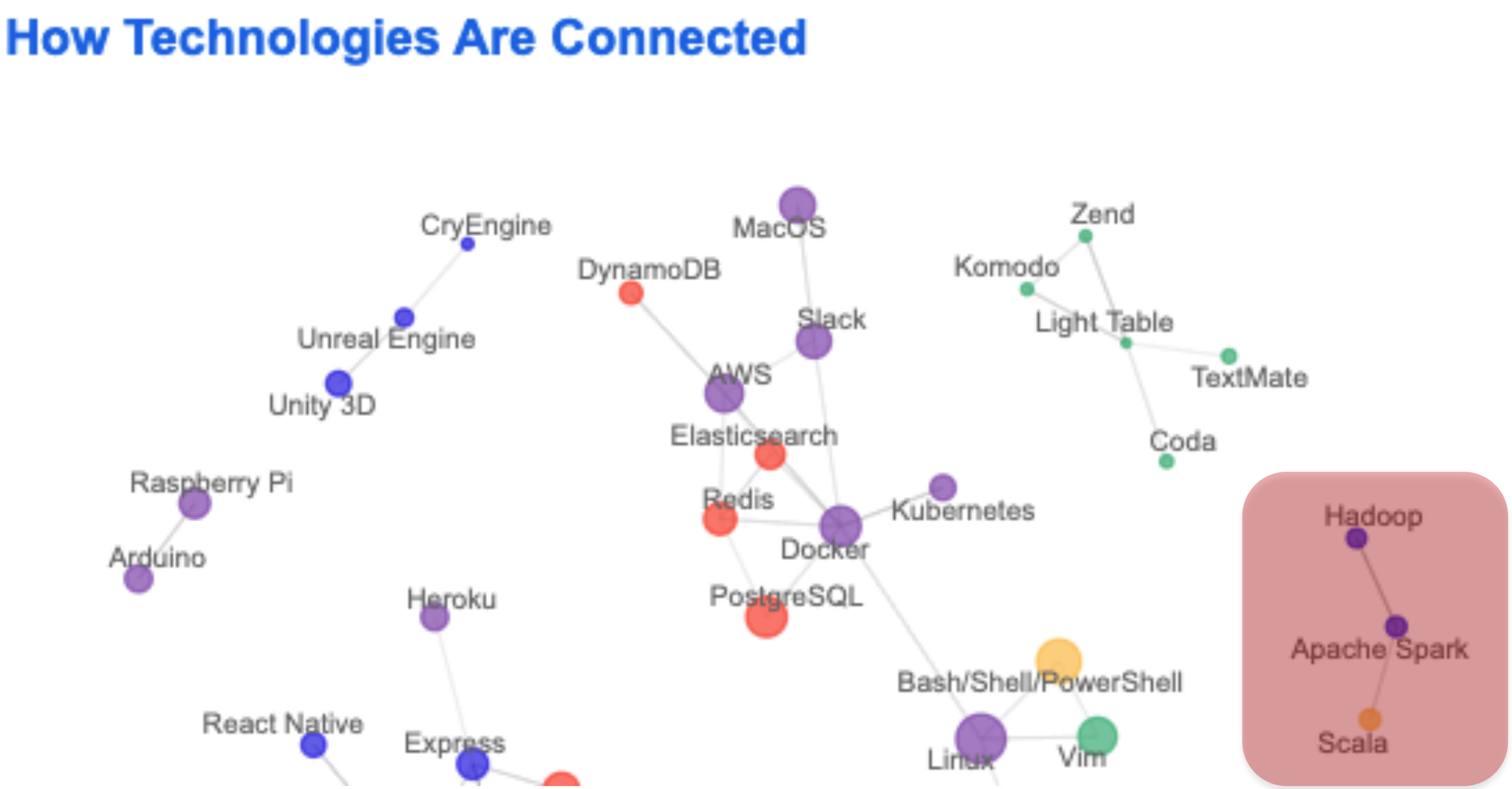
❖What to do with large datasets?

Rewrite from scratch ☹

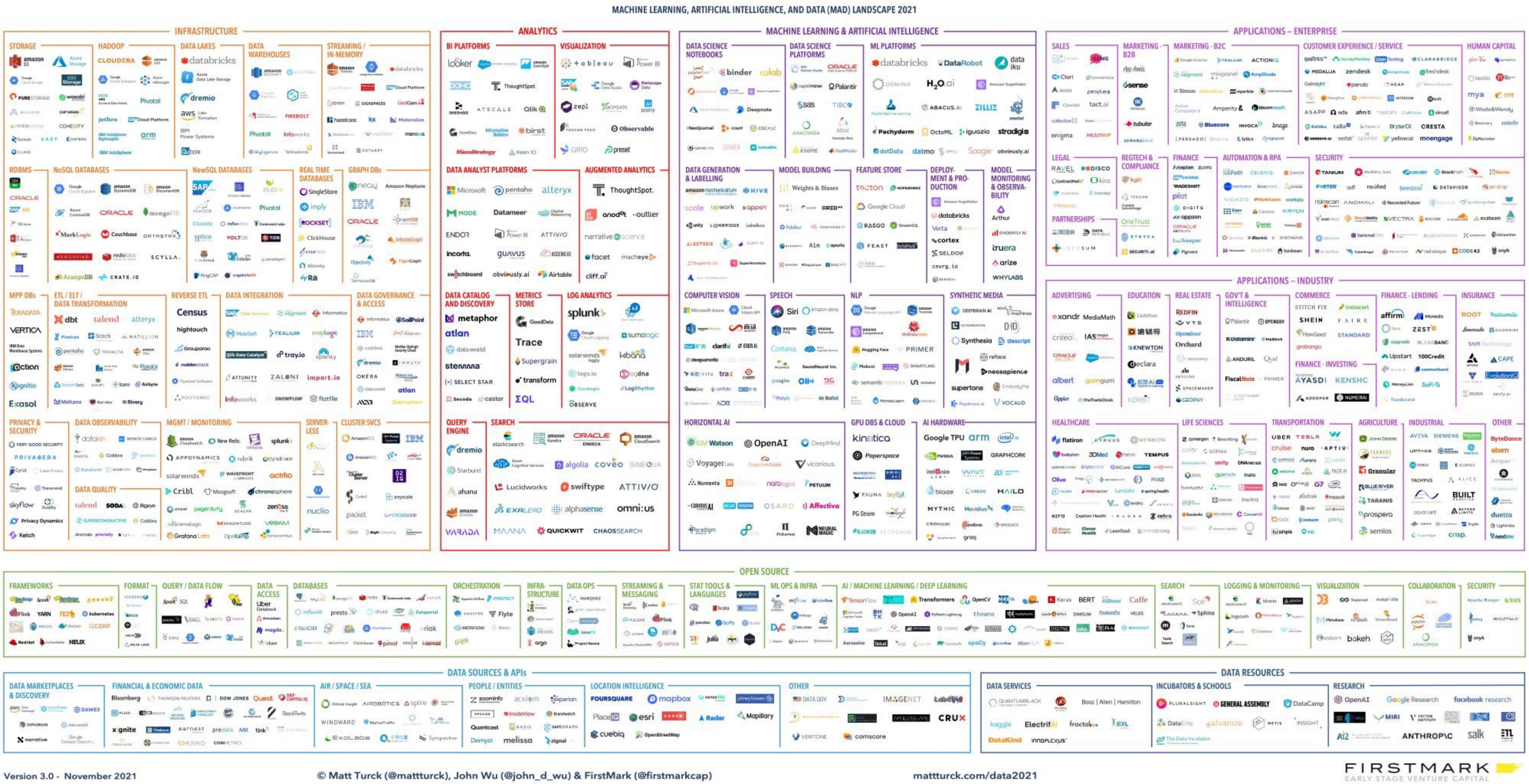
Is there any language without this issue?

Scala

Why Scala is related to BigData?



https://insights.stackoverflow.com/survey/2019


MACHINE LEARNING, ARTIFICIAL INTELLIGENCE, AND DATA (MAD) LANDSCAPE 2021
INFRASTRUCTURE
STORAGE
HADOOP
DATA LAKES
DATA WAREHOUSES
STREAMING / IN-MEMORY
RDBMS
NoSQL DATABASES
NewSQL DATABASES
REAL TIME DATABASES
GRAPH DBs
MPP DBs
ETL / ELT / DATA TRANSFORMATION
REVERSE ETL
DATA INTEGRATION
DATA GOVERNANCE & ACCESS
PRIVACY & SECURITY
DATA OBSERVABILITY
DATA QUALITY
MGMT / MONITORING
SERVERLESS
CLUSTER SVCS
ANALYTICS
BI PLATFORMS
VISUALIZATION
DATA ANALYST PLATFORMS
AUGMENTED ANALYTICS
DATA CATALOG AND DISCOVERY
METRICS STORE
LOG ANALYTICS
QUERY ENGINE
SEARCH
MACHINE LEARNING & ARTIFICIAL INTELLIGENCE
DATA SCIENCE NOTEBOOKS
DATA SCIENCE PLATFORMS
ML PLATFORMS
DATA GENERATION & LABELLING
MODEL BUILDING
FEATURE STORE
DEPLOYMENT & PRODUCTION
MODEL MONITORING & OBSERVABILITY
COMPUTER VISION
SPEECH
NLP
SYNTHETIC MEDIA
HORIZONTAL AI
GPU DBS & CLOUD
AI HARDWARE
APPLICATIONS – ENTERPRISE
SALES
MARKETING - B2B
MARKETING - B2C
CUSTOMER EXPERIENCE / SERVICE
HUMAN CAPITAL
LEGAL
REGTECH & COMPLIANCE
FINANCE
AUTOMATION & RPA
SECURITY
PARTNERSHIPS
APPLICATIONS – INDUSTRY
ADVERTISING
EDUCATION
REAL ESTATE
GOV'T & INTELLIGENCE
COMMERCE
FINANCE - LENDING
INSURANCE
FINANCE - INVESTING
HEALTHCARE
LIFE SCIENCES
TRANSPORTATION
AGRICULTURE
INDUSTRIAL
OTHER
OPEN SOURCE
FRAMEWORKS
FORMAT
QUERY / DATA FLOW
DATA ACCESS
DATABASES
ORCHESTRATION
INFRASTRUCTURE
DATA OPS
STREAMING & MESSAGING
STAT TOOLS & LANGUAGES
ML OPS & INFRA
AI / MACHINE LEARNING / DEEP LEARNING
SEARCH
LOGGING & MONITORING
VISUALIZATION
COLLABORATION
SECURITY
DATA SOURCES & APIs
DATA MARKETPLACES & DISCOVERY
FINANCIAL & ECONOMIC DATA
AIR / SPACE / SEA
PEOPLE / ENTITIES
LOCATION INTELLIGENCE
OTHER
DATA RESOURCES
DATA SERVICES
INCUBATORS & SCHOOLS
RESEARCH
Version 3.0 - November 2021
© Matt Turck (@mattturck), John Wu (@john_d_wu) & FirstMark (@firstmarkcap)
mattturck.com/data2021
FIRSTMARK
EARLY STAGE VENTURE CAPITAL

Mainstream Big Data models

How to store, manage and process Big Data by harnessing large clusters of commodity nodes

- MapReduce family: simpler, more constrained

HadoopDB

- Dataflow family: enables more complex processing & data, optimization opportunities

Google Pregel

Microsoft Dryad

The Hadoop Ecosystem

Storm, Flink (Streaming)	**Giraph, Hama** (Graph)	**Pig, Hive** (Query)	**Other Hadoop libraries**

Hadoop MapReduce
Distributed Batch Processing Framework

YARN
Cluster resource management

HDFS
Hadoop Distributed File System

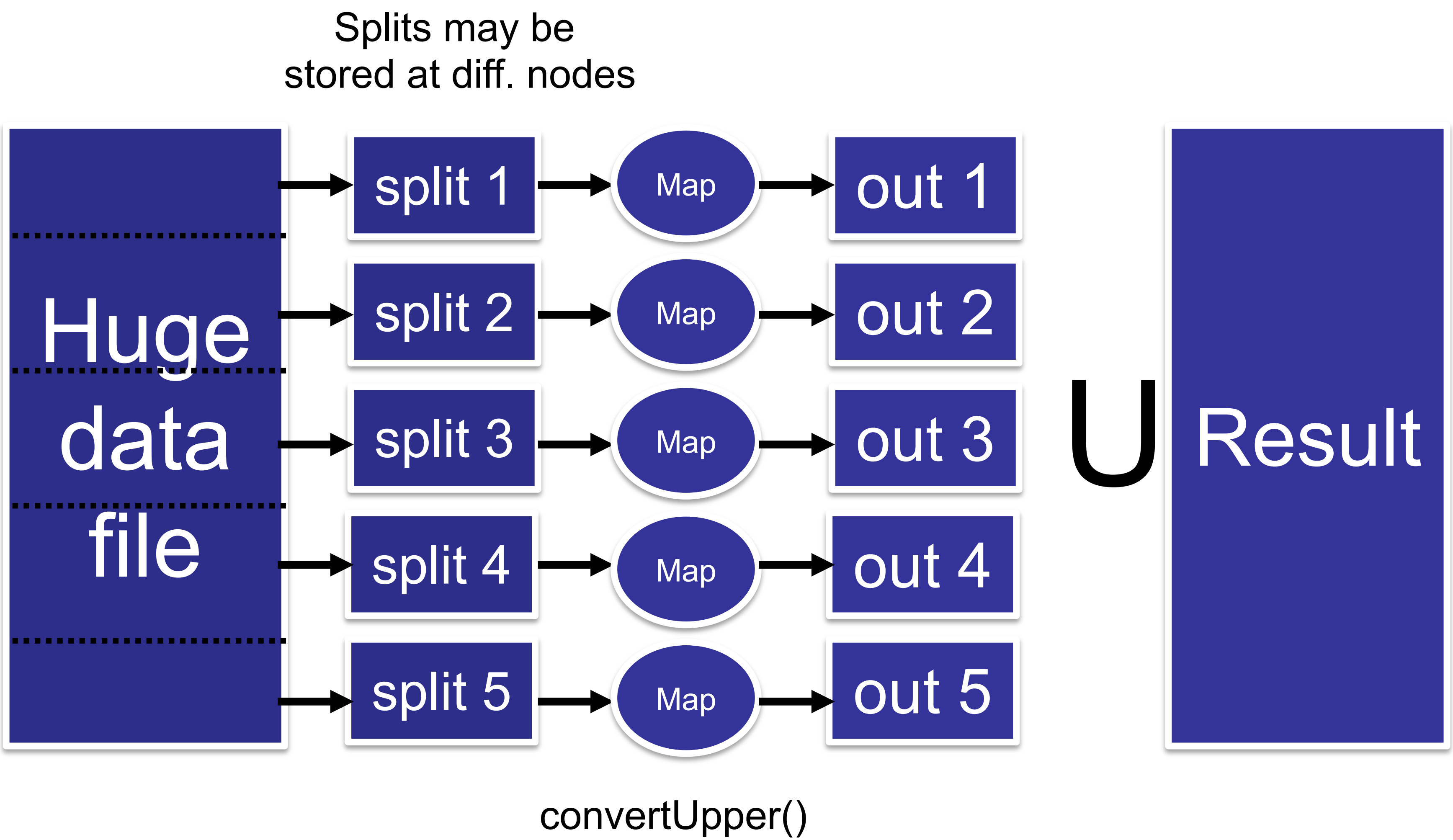
Spark Software Stack

Spark SQL (SQL)	**MLlib** (Machine Learning)	**GraphX** (Graph processing)	**Spark Streaming** (Streaming)	**Other Spark libraries**
Spark Core Processing Engine				
Mesos / YARN / Standalone Cluster Resource Management				
HDFS / Amazon S3 / OpenStack Swift / Cassandra Distributed File System & Storage				

PROGRAMMING MODELS

The vision…

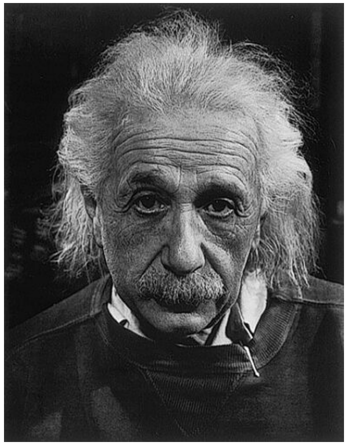
Sample function: convert all text to upper case



The vision (2)

More complicated: the word-count problem

- Huge file → extract frequencies of words
- Example



Einstein once said…

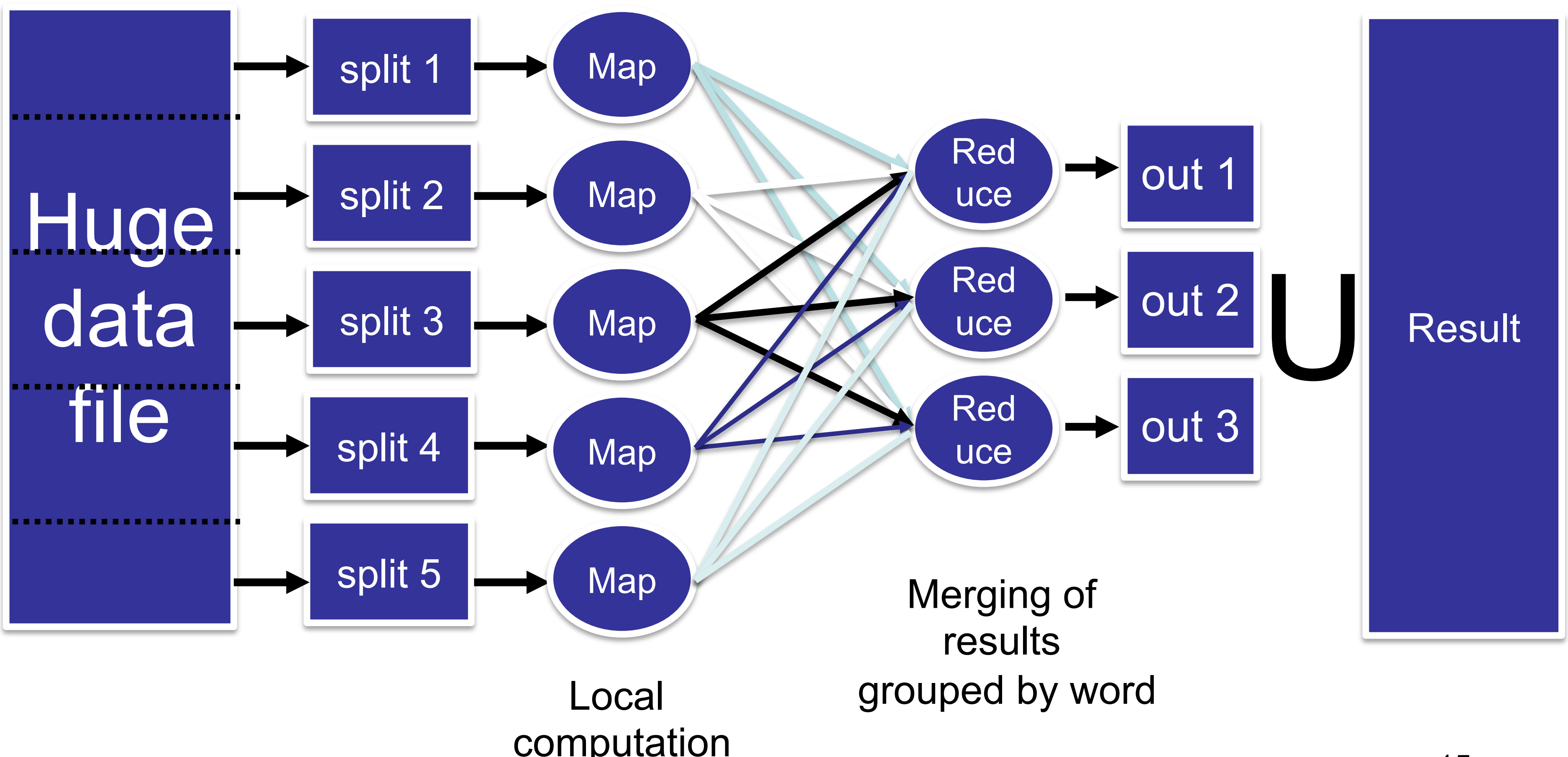
Logic will get you from A to B.
Imagination will take you everywhere.

Extracted frequencies:

- <Logic,1>, <will,2>, <get,1>, <you,2>, …

The vision (3)

Sample application: the word-count example



MapReduce programming model

- Data model: everything is a <key,value> pair
- Programming model - two core functions
  - Map(key,value): Invoked for every split of the input data. Value corresponds to the split.
  - Reduce(key,list(values)): Invoked for every unique key emitted by Map. List(values) corresponds to all values emitted from ALL mappers for this key.
- These are second-order functions
  - Map(key,value, MapperClassName)
  - Reduce(key,list(values), ReducerClassName)

→ parallelism and deployment handled by the system

MapReduce programming model (2)

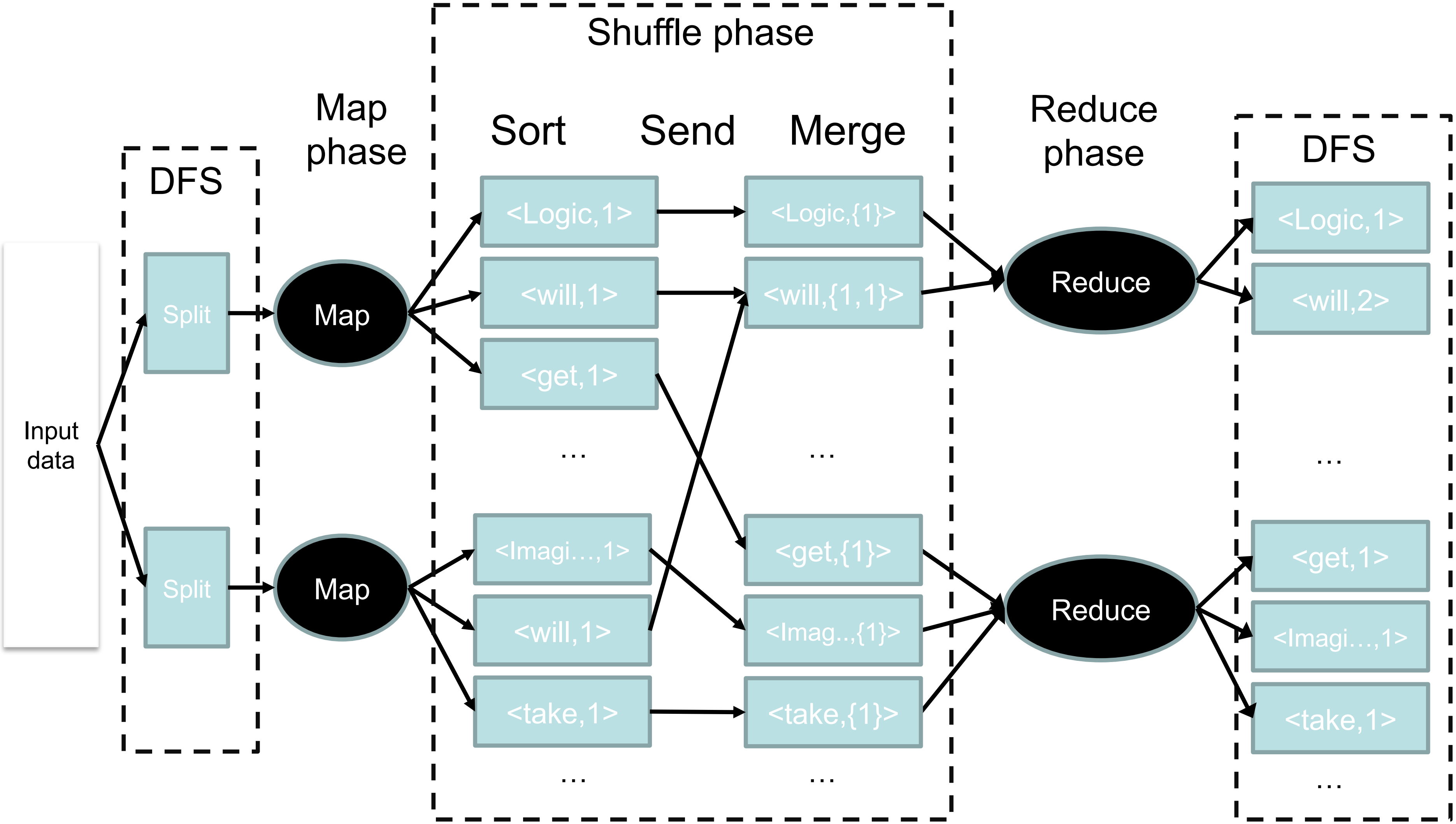
- The word-count problem
  - Input: Text file, broken in splits
  - Output: Frequency of each word observed in the file
  - Map(key,value): value: a split of the text file

```
for each word in value
   emit pair <word,+1>
```

  - Reduce(key,list(values)): Key: word, values: list of (+1's)

```
count=0
for each value in list(values)
   count+=value
emit pair<key,count>
```

MapReduce – under the hood

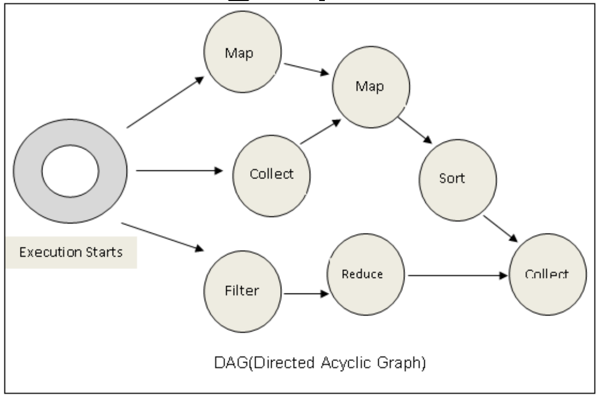



```
for each word in value
    emit pair <word,+1>
```

```
for each value in list(values)
    count+=value
emit pair<word,count>
```

Dataflow programming model

- MapReduce simple but weak for some reqs.
  - Cannot define complex processes
  - Everything file-based, no distributed memory
  - Procedural → difficult to optimize
- Dataflow
  - Processing expressed as a DAG, tree, graph with cycles, …
  - Vertices: processing tasks
  - Edges: Communication
    - DAG: Spark, Dryad
    - Tree: Dremel
    - Directed graph with cycles: Pregel



Spark DAG example

Dataflow programming model (2)

- Describing the processing tasks
  - Declarative languages, e.g., Dremel

```
SELECT DocId AS Id, COUNT(Name.Language.Code)
WITHIN Name AS Cnt FROM t
WHERE REGEXP(Name.Url, '^http');
```

  - Functional programming, e.g., Spark

```
val wordCounts = textFile.flatMap(line => line.split(" ")).
            map(word => (word, 1)).
            reduceByKey((a, b) => a + b)
wordCounts.collect()
```

  - Domain-specific languages, e.g., Pregel

for graph processing

```
class PageRankVertex
  : public Vertex<double, void, double> {
      public: virtual void Compute(MessageIterator* msgs) {
         const int64 n = GetOutEdgeIterator().size();
         SendMessageToAllNeighbors(GetValue() / n);
      }
  };
```

Why Spark? (1)

Scala **Collections** ⟷ Apache Spark

Similar API ☺

Which programming language is this?

```
Integer totalAgeReduce =
roster.stream()
    .map(Person::getAge)
    .reduce( 0, (a, b) -> a + b);


Map<String, List<String>> a = words
     .stream().collect(
          Collectors.groupingBy(w ->
sortChars(w)));
```

PLs that have a functional collection interface like Scala

C++, C#, F#, Clojure, Haskell, Java8, JavaScript, Perl, PHP, Python, Ruby, Scheme, Smalltalk, Standard ML, OCAML, …

See

https://en.wikipedia.org/wiki/Map_(higher-order_function)

Fault Tolerance

- Essential for scaling out
- The main reason behind the success of MapReduce in Google
- Requires writing intermediate data to disk

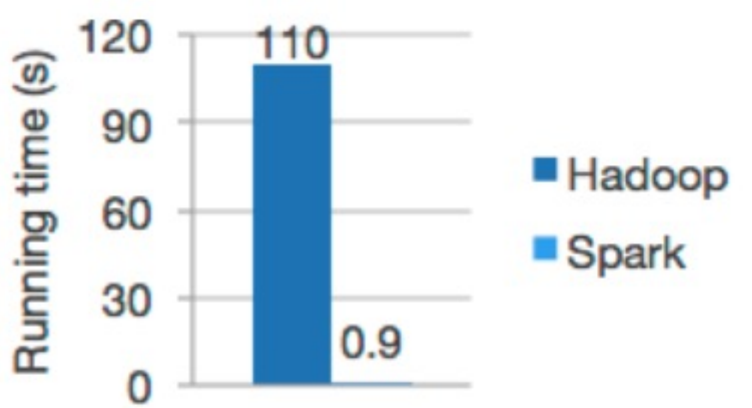
Fault Tolerance in Spark

- Data
  - Immutable
  - In-memory
- Operations = Functional transformations
- Fault tolerance = Replay operations

Why Spark? (2)

- Compared to Hadoop MapReduce, improves efficiency through:
  - General execution graphs
  - In-memory storage

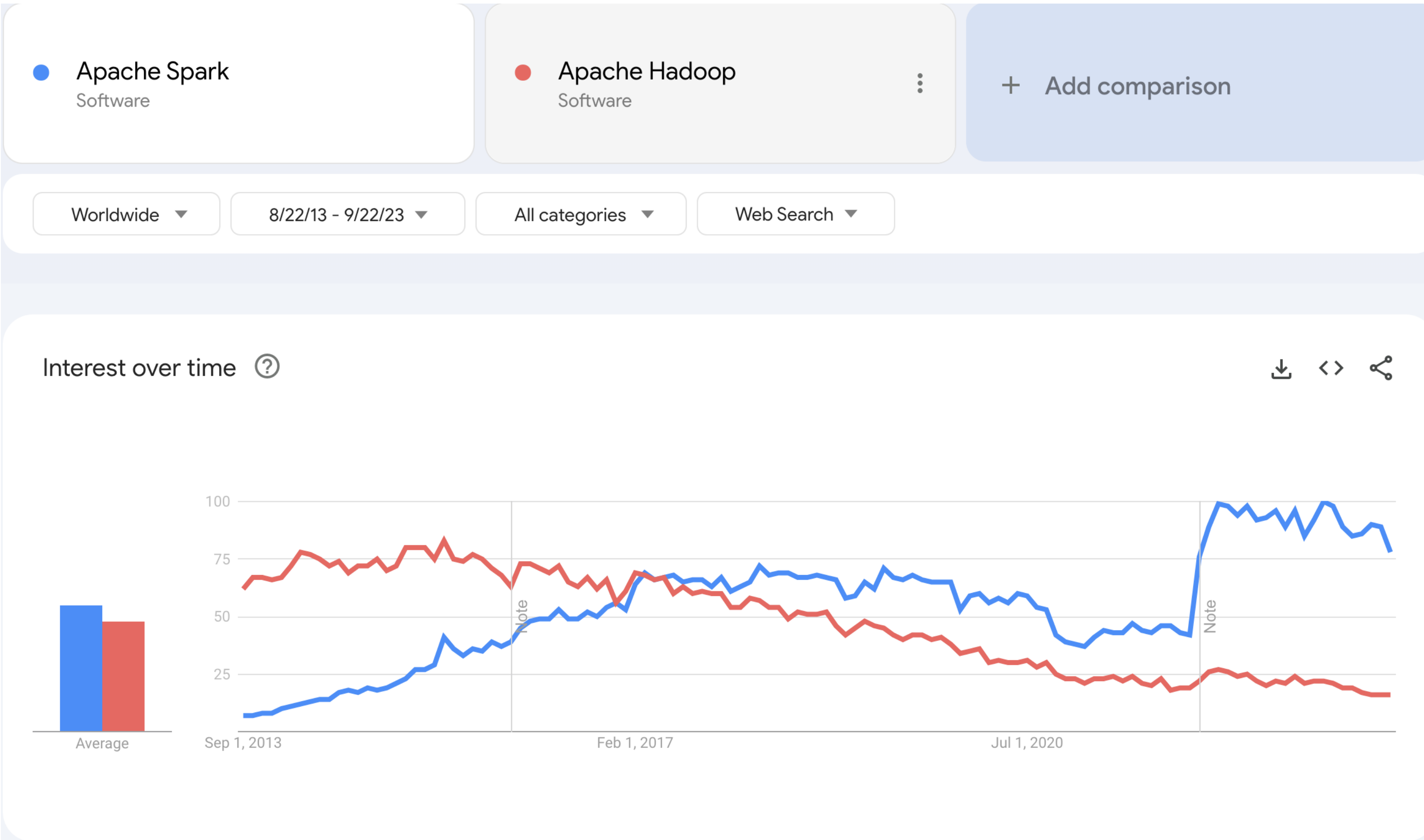
Up to 10 × faster on disk,
100 × in memory



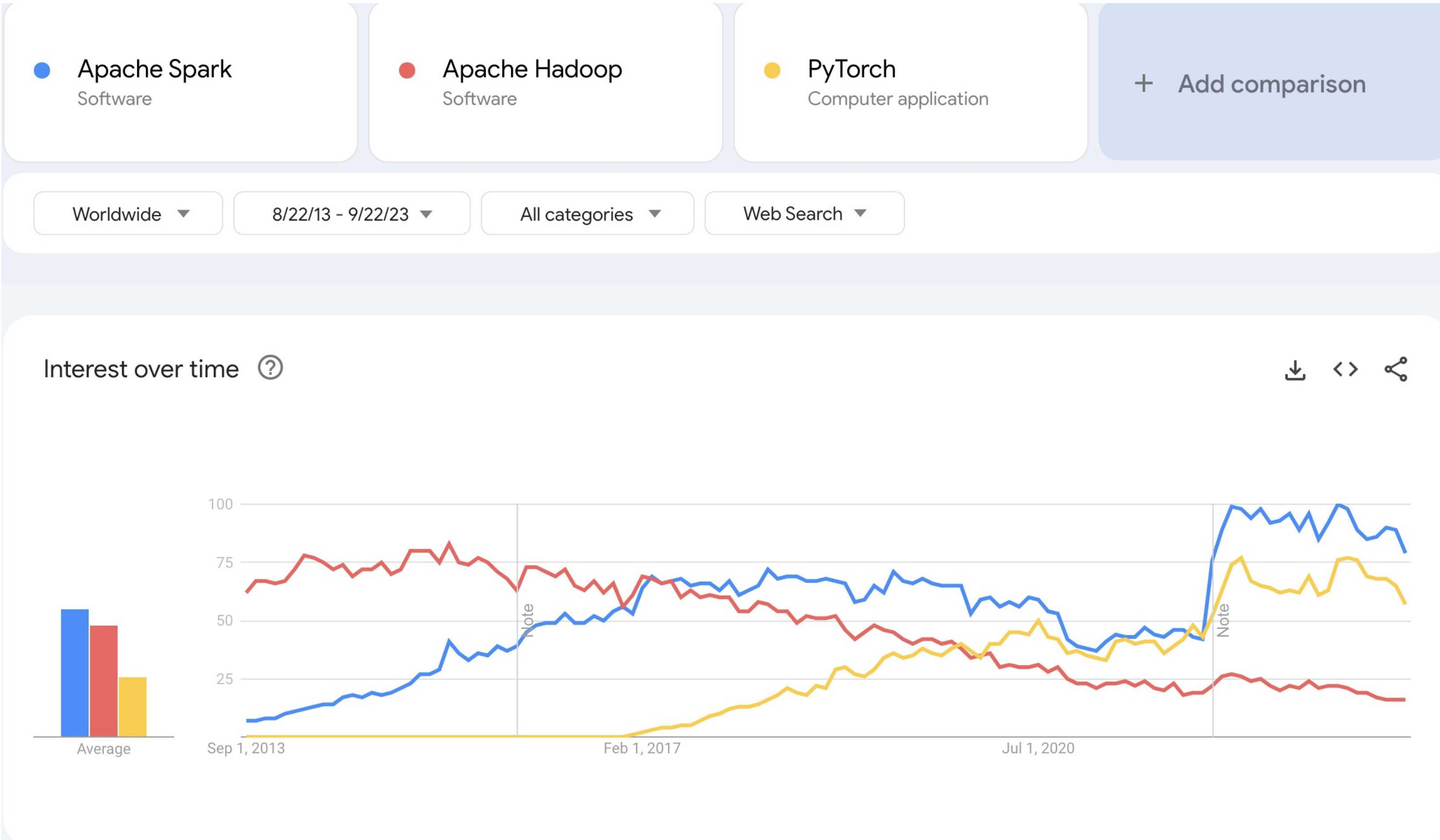
Logistic regression in Hadoop and Spark

http://spark.apache.org/

Why Spark? (3)


Apache Spark
Software
Apache Hadoop
Software
+ Add comparison
Worldwide
8/22/13 - 9/22/23
All categories
Web Search
Interest over time
100
75
50
25
Note
Note
Average
Sep 1, 2013
Feb 1, 2017
Jul 1, 2020

Why Spark? (4)


Apache Spark
Software
Apache Hadoop
Software
PyTorch
Computer application
+ Add comparison
Worldwide
8/22/13 - 9/22/23
All categories
Web Search
Interest over time
100
75
50
25
Note
Note
Average
Sep 1, 2013
Feb 1, 2017
Jul 1, 2020

Learn Scala

Scala School!

From ∅ to Distributed Service

Other Languages:

- 한국어
- Русский
- 简体中文

About

Scala school started as a series of lectures at Twitter to prepare experienced engineers to be productive Scala programmers. Scala is a relatively new language, but draws on many familiar concepts. Thus, these lectures assumed the audience knew the concepts and showed how to use them in Scala. We found this an effective way of getting new engineers up to speed quickly. This is the written material that accompanied those lectures. We have found that these are useful in their own right.

Lessons

Basics
Values, functions, classes, methods, inheritance, try-catch-finally. Expression-oriented programming

Basics continued
Case classes, objects, packages, apply, update, Functions are Objects (uniform access principle), pattern matching.

Collections
Lists, Maps, functional combinators (map, foreach, filter, zip, folds)

Pattern matching & functional composition
More functions! PartialFunctions, more Pattern Matching

Type & polymorphism basics
Basic Types and type polymorphism, type inference

https://twitter.github.io/scala_school/

QUESTIONS?