Introduction to Data Science Module 3

Data Types and Storage

FIT1043 2017 Lecture 5

Monash University

Discussion: Python Language

- easy to learn
- flexible and multi-purpose
- great libraries
- well designed computer language
- good visualization for basic analysis

Discussion: Python versus R

- both are free
- R developed by statisticians for statisticians, huge support for analysis
- Python by computer scientists for general use
- R is better for stand-alone analysis and exploration
- Python allows for easier integration with other systems
- Python easier to learn and extend than R (better language)
- R currently less scalable.

See *In data science, the R language is swallowing Python* by Matt Asay, recent blog in *Infoworld*.

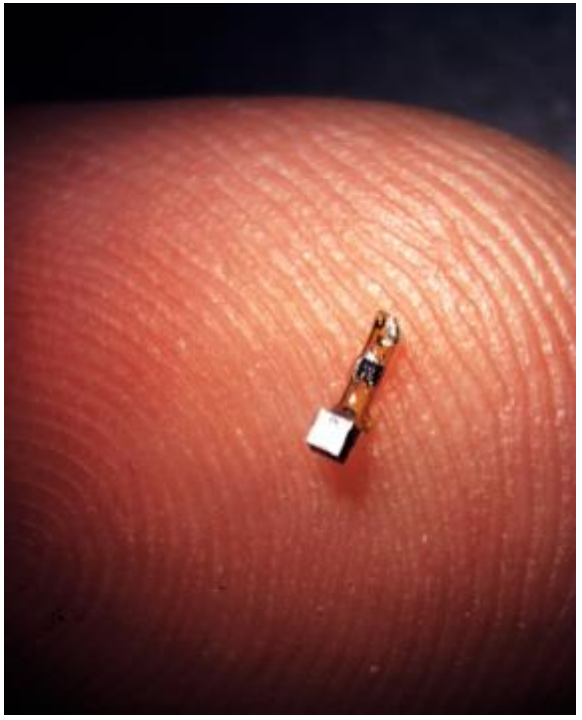
MARS Question
New Classes of Computing

Remember Bell's law ... new classes of computing every decade.

Can you suggest some new classes of computing?

Discussion:
New Classes of Computing

mind-reading or mind-control devices



in-body devices

NB. sounds like science fiction but we know R&D exists in all these areas!

Unit Schedule: Modules

Module	Week	Content
1.	1	Overview and look at projects
	2	(Job) roles, and the impact
2.	3	Data business models / application areas
3.	4	**Characterising data and "big" data**
	5	**Data sources and case studies**
4.	6	Resources and standards
	7	Resources case studies
5.	8	Data analysis theory
	9	Regression and decision trees
	10	Data analysis process
6.	11	Issues in data management
	12	GUEST SPEAKER & EXAM INFO

Big Data Processing
(ePub section 3.4)

processing data at scale, especially for analysis

- databases

  storing and accessing data

- distributed processing

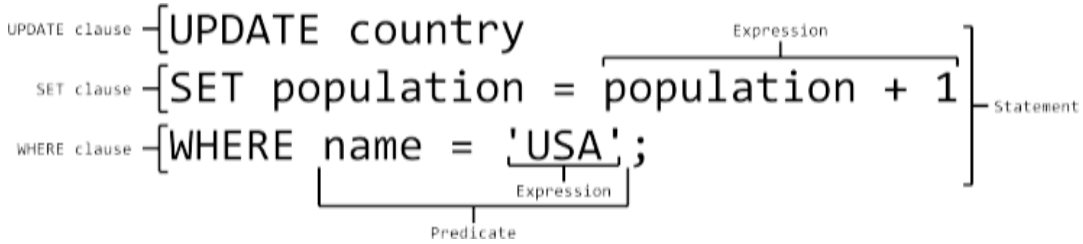
  breaking up computation to scale it up

Big Data Processing:
Databases

storing and accessing data

SQL Review

- Relational Database Management Systems (RDBMS)
- SQL ::= structured query language



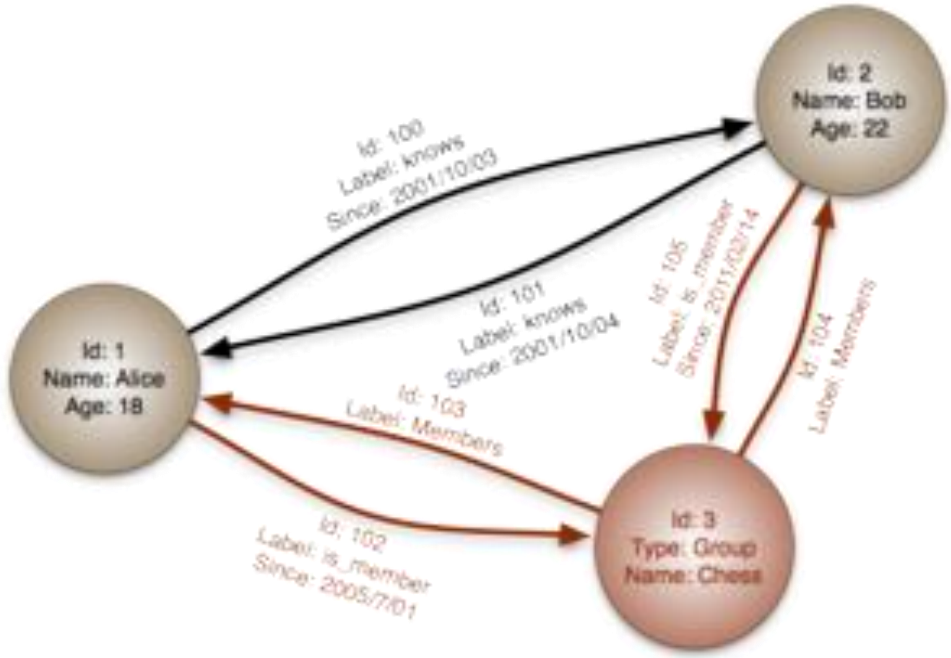
- rather like large scale set of Excel spreadsheets with better indexing and retrieval
- transaction oriented with support for correctness, distribution, ...

JSON Example

```
{
  "firstName": "John",
  "lastName": "Smith",
  "isAlive": true,
  "age": 25,
  "address": {
    "streetAddress": "21 2nd Street",
    "city": "New York",
    "state": "NY",
    "postalCode": "10021-3100"
  },
  "phoneNumbers": [
    {
      "type": "home",
      "number": "212 555-1234"
    },
    {
      "type": "office",
      "number": "646 555-4567"
    }
  ],
  "children": [],
  "spouse": null
}
```

- no fixed format
- semi-structured, key-value pairs, hierarchical
- “friendly” alternative to XML
- self-documenting structure

Graph Database Example



- stores graph, commonly as triples, subject, verb, object
- commonly used to store Linked Open Data

Database Background Concepts

Many NoSQL and SQL DBs offer:

- large scale, distributed processing
- robustness achieved
- general query languages
- some notion of consistency
  e.g. “eventually” as nodes spread updates

Beyond SQL Databases

Type	Notes
RDBMS	SQL
Object DB	navigate network
Doc. DB	JSON like, Javascript like queries
key-val cache	in-memory
key-val store	not in-memory but highly optimised
tabular key-val	relational-like, “wide column store”
graph DB	RDF, SPARQL,

SQL and Beyond SQL Databases (NoSQL)

- Use SQL database when:
  - data is structured and unchanging
- Use NoSQL database when:
  - Storing large volume of data with little to no structure
  - Data changes rapidly
- NoSQL databases offer a rich variety beyond traditional relational.

Big Data Processing:
Distributed processing

breaking up computation to scale it up

Overview: Processing

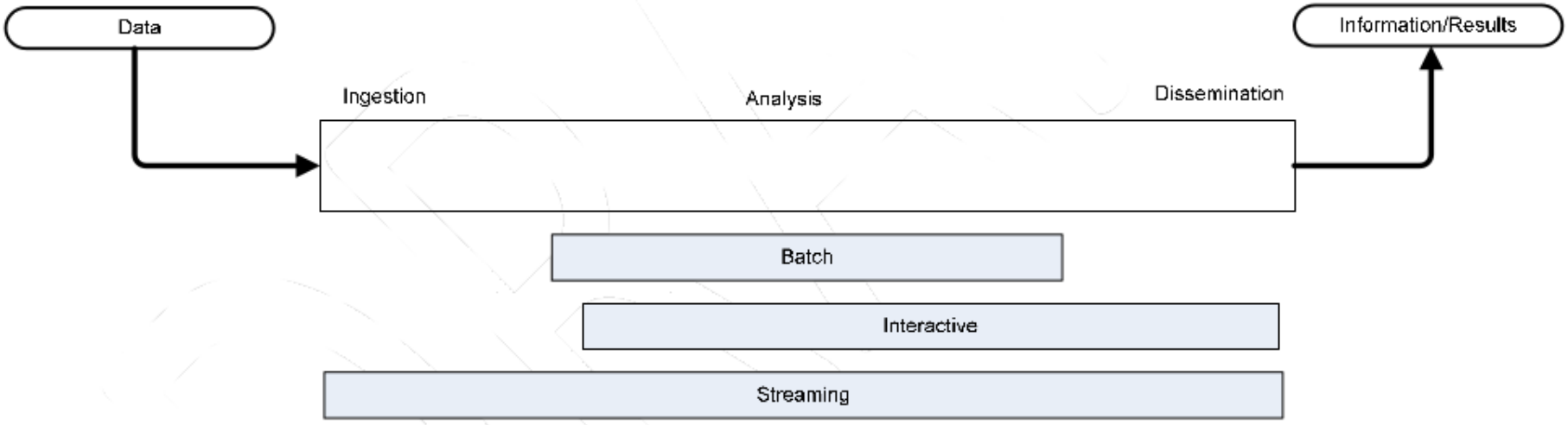


Figure 5: Information Flow

Interactive: bringing humans into the loop

Streaming: massive data streaming through system with little storage

Batch: data stored and analysed in large blocks, "batches," easier to develop and analyse

Processing Background Concepts

in-memory: in RAM, *i.e.*, not going to disk

distributed computing: across multiple machines

scalability: to handle a growing amount of work; to be enlarged to accommodate growth (not just “big”)

data parallel: processing can be done independently on separate chunks of data

yes: process all documents in a collection to extract names

no: convert a wiring diagramme into a physical design (optimisation)

MARS Question

Which one of the following tasks is very hard to make data parallel?

A. Face recognition in 1M images
B. Invert a large matrix
C. Looking for common 3-4 word phrases in a collection of documents

Distributed Analytics

- legacy systems provide powerful statistical tools on the desktop

  SAS, R, Matlab

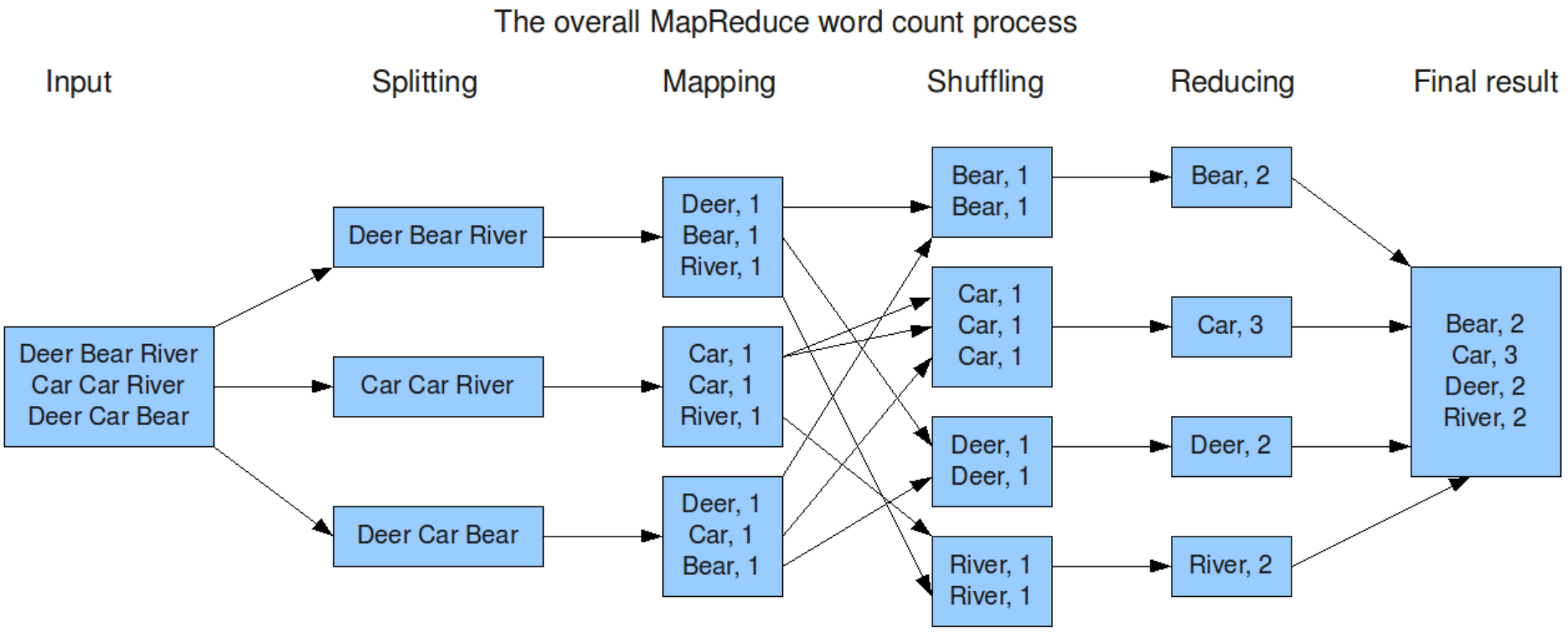
  but often-times without distributed or multi-processor support
- supporting distributed/multi-processor computation requires special redesign of algorithms

Map-Reduce

Simple distributed processing framework developed at Google

- published by Dean and Ghemawat of Google in 2004
- **intended to run on commodity hardware**; so has fault-tolerant infrastructure
- from a distributed systems perspective, is quite simple

Map-Reduce Example



for a simple word-count task: (1) divide data across machines (2) `map()` to key-value pairs (3) sort and `merge()` identical keys

Map-Reduce, cont.

- requires simple data parallelism followed by some merge ("reduce") process
- stopped using by Google probably in 2005
- Google now uses *"Cloud Dataflow"* (and *here*), available commercially, as open source

Hadoop

Open-source Java implementation of Map-Reduce

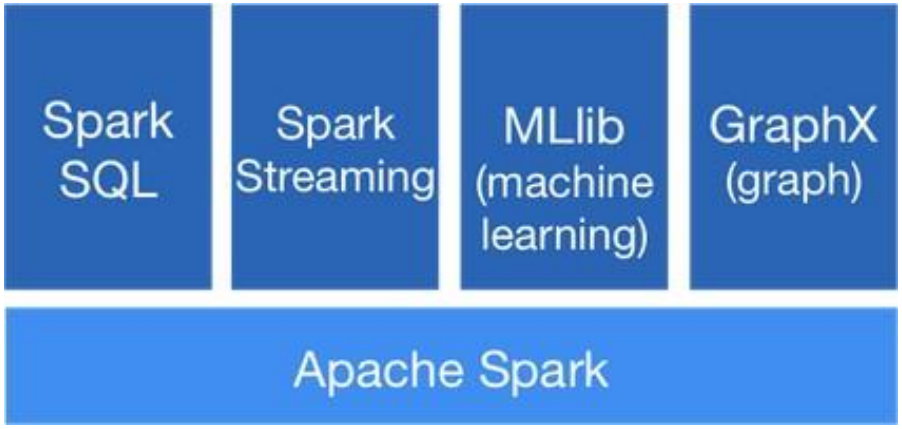
- originally developed by *Doug Cutting* while at Yahoo!
- architecture:

  Common: Java libraries and utilities

  MapReduce: core paradigm
- huge tool ecosystem
- well passed the peak of the hype curve

Spark

- another (open source) Apache top-level project at *Apache Spark*
- developed at *AMPLab* at UC Berkeley
- builds on Hadoop infrastructure
- interfaces in Java, Scala, Python, R
- provides in-memory analytics
- works with some of the Hadoop ecosystem



MARS Question

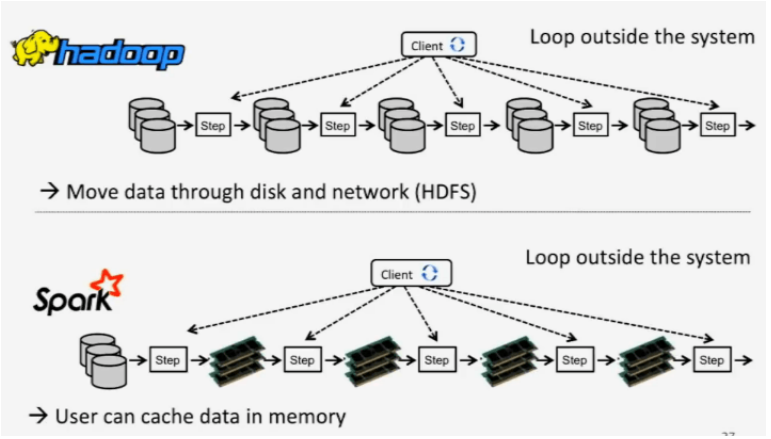
Which one of the following is suitable for real-time data processing?

A. Hadoop
B. Spark

Summary: Hadoop and Spark

- Hadoop provides an inexpensive and open source platform for parallelising processing:
  - based on a simple Map-Reduce architecture
  - not suited to streaming (suitable for offline processing)

- Spark is a more recent development than Hadoop
  - includes Map-Reduce capabilities
  - provides real-time, in-memory processing
  - much faster than Hadoop



Next: Module 4

Data Resources, Processes, Standards and Tools