The Cloud and Data Analytics

the cloud

Cloud computing refers to the provision of computing resources over the internet

key concept: virtualization

Virtualization refers to the ‘in-memory’ cloning of a hardware resource

Properties of virtualization

as your needs grow

Scalable: the machine can grow or contract as requirements increase or decrease

Pay per use: the user only pays for what they end up using

Transferable: applications can be easily transferred and are not dependent on hardware

Redundancy: cloud services have built in redundancy guaranteeing continuous service

why?

- Instantaneous time to market
  - ✓ just sign up for a cloud service
  - ✓ no need to buy equipment, hire staff
- Pay per use and no upfront cost
  - ✓ typically pay for time
  - ✓ 10 cpu for 1 hr = 1 cpu for 10 hours
- Cost matching with arbitrary loads
  - ✓ no need to buy infrastructure for max usage
- High fault tolerance

why?

bigger machine, larger storage

- Manage large (many terabytes) datasets
  - ✓ data can sit on multiple servers
  - ✓ redundancy can reduce risk of losing data
- Analyze large (big) data
  - ✓ multi core machines
  - ✓ persistent intermediate computations
  - ✓ group jobs for maximum efficiency
- Cost efficiency
  - ✓ scale up for JIT machine usage

Clustered File Systems

A file system that is stored across multiple servers

* scalable: data can grow horizontally
* transparent to the client
* location independence
* concurrency level consistency

same results regardless where the data are stored at.

Clustered File Systems work in tandem with virtualized machines.

Hadoop

scalable, fault tolerant system capable of processing large datasets across a cluster of servers

Hadoop components

Cluster manager

Cloud dataproc

Compute engine

Compute engine

pyspark
spark
scala
yarn
mapreduce

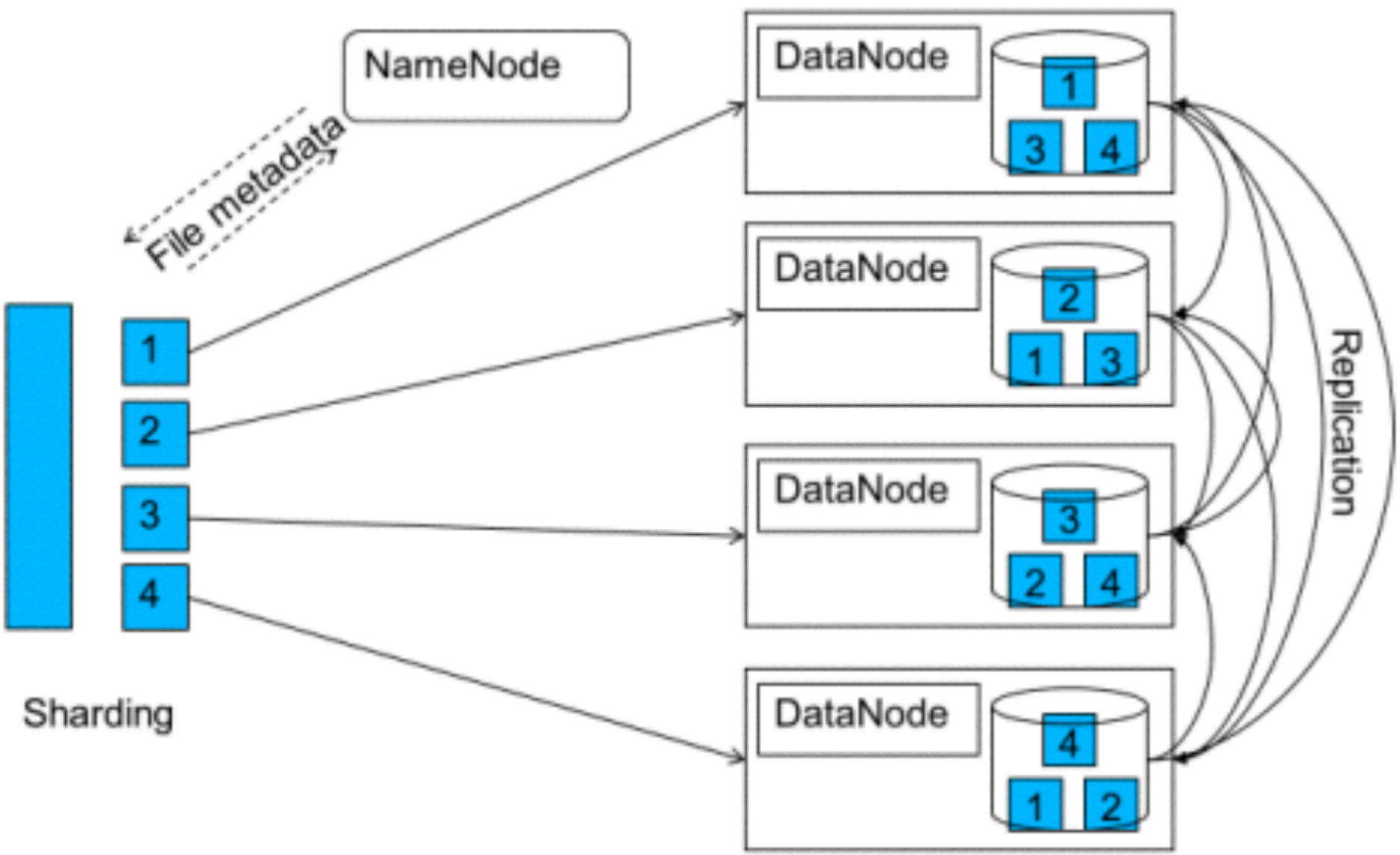
Distributed file system

hdfs

Hadoop Distributed File System (HDFS)

1 datanode = 1 machine

4 blocks of data



will lost data, only if 3 or more DataNodes failed.

Hadoop 2 features

- Block size
  - ✓ data can sit on multiple servers
  - ✓ redundancy can reduce risk of losing data
- Replication factor
  - ✓ multi core machines
  - ✓ persistent intermediate computations
  - ✓ group jobs for maximum efficiency
- Name node
  - ✓ Meta data (directory data) about the file system
- Data node
  - ✓ A machine that contains data (many per hdfs)

Distributed functional programming

F.P. makes Big Data processing possible

Distributing functions across a distributed dataset and combining the results

* stateless functions
* chained function calls
* combining results (worker/master model)

within a function, you should not be changing the DataSet at all. You just create the new data.

Distributed functional programming

MapReduce

map: apply a function to data stored in each cluster

to different data sets

reduce: combine the outputs of each map to get some "aggregated" result

key/value: data is always in the form of (key, value) pairs

data node 1

data node 2

map

“a man, a plan”

“a canal, panama”

(‘a’,1)
(‘man’,1)
(‘a’,1)
(‘plan’,1)

(‘a’,1)
(‘canal’,1)
(‘panama’,1)

reduce

(‘a’,2)
(‘man’,1)
(‘plan’,1)

(‘a’,1)
(‘canal’,1)
(‘panama’,1)

reduce

(‘a’,3)
(‘man’,1)
(‘plan’,1)
…

types of programming languages

imperative programs: set of sequential instructions that tell the program how to convert inputs into outputs through a series of state transitions. Imperative programs may be procedural or object-oriented

functional programs: problems are decomposed into sets of functions. Each function creates new objects and functions may be composite. Functional programs can be expressed mathematically and are “provable”.

20 —> 4 groups of 5 sentences each —> use MapReduce

Much faster for a large DataSet, even though with a more convoluted program.

functional programs

1. a function can be passed as arguments to, or returned as return values from, other functions
   ```
   map(max,((1,2),(3,4)))
   ```
2. a function has no side-effects - i.e., it makes no changes to any data outside its own scope
3. a program can be represented using a mathematical expression (a necessary condition for provable programs)
   ```
   kp2 = map(lambda x:(x,1),text2.lower().split())
   fkp2 = filter(lambda x: len(x[0])>4,kp2)
   filter(f1,f2) where f1 is the condition and f2 is the map function
   ```
4. functions create new instances of data and never update existing instances (referential transperancy)
5. function expressions are not evaluated unless necessary (lazy evaluation)

do not generate anything unnecessary

otherwise, need too much storage for replicates of data

the cloud and functional programming

1. functional programming makes it easy to thread programs because:
   - each piece of code has no side effects
   - does not change existing data
2. functional programming is better at dealing with large datasets
   - the entire data need not be read into memory
   - lazy evaluation ensures that only necessary evaluation is done


APACHE
Spark
TM

What is Spark

Apache Spark is a fast and general purpose computing platform (not a language)

It is an extension of the Hadoop Map-Reduce model that allows interactive data analysis

Spark uses in-memory datasets for fast computation

Spark integrates with Hadoop clusters

Spark uses multiple data types including Hadoop, SQL servers, csv files, JSON files and streams

Spark has APIs for Python, Java, C++, Scala, SQL

Spark and Machine Learning

Makes ML scalable (no need to rewrite code as the dataset grows)

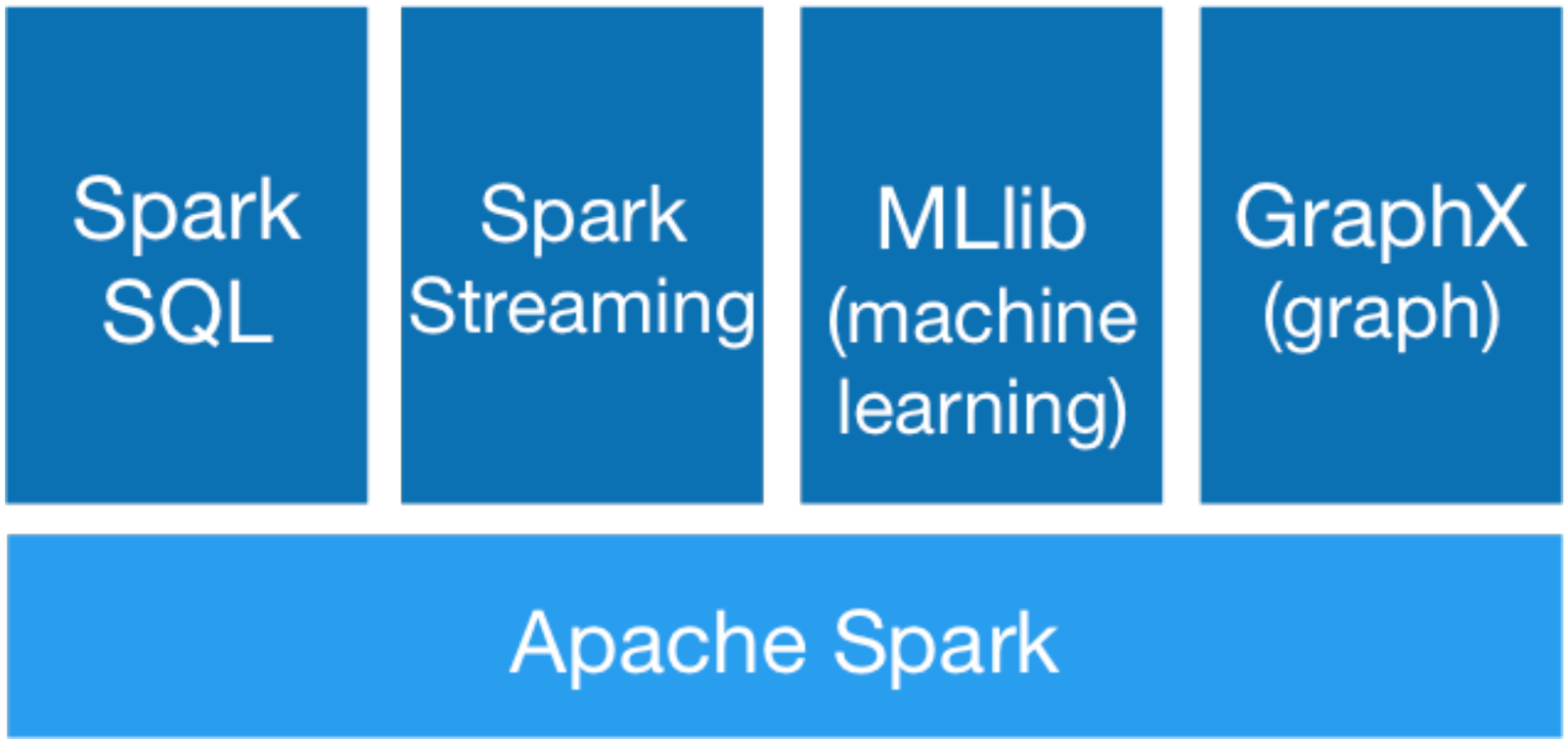
Contains powerful ML libraries

Has libraries for Streaming ML for Streaming Data (realtime data)

Provides easy to set up pipelining support

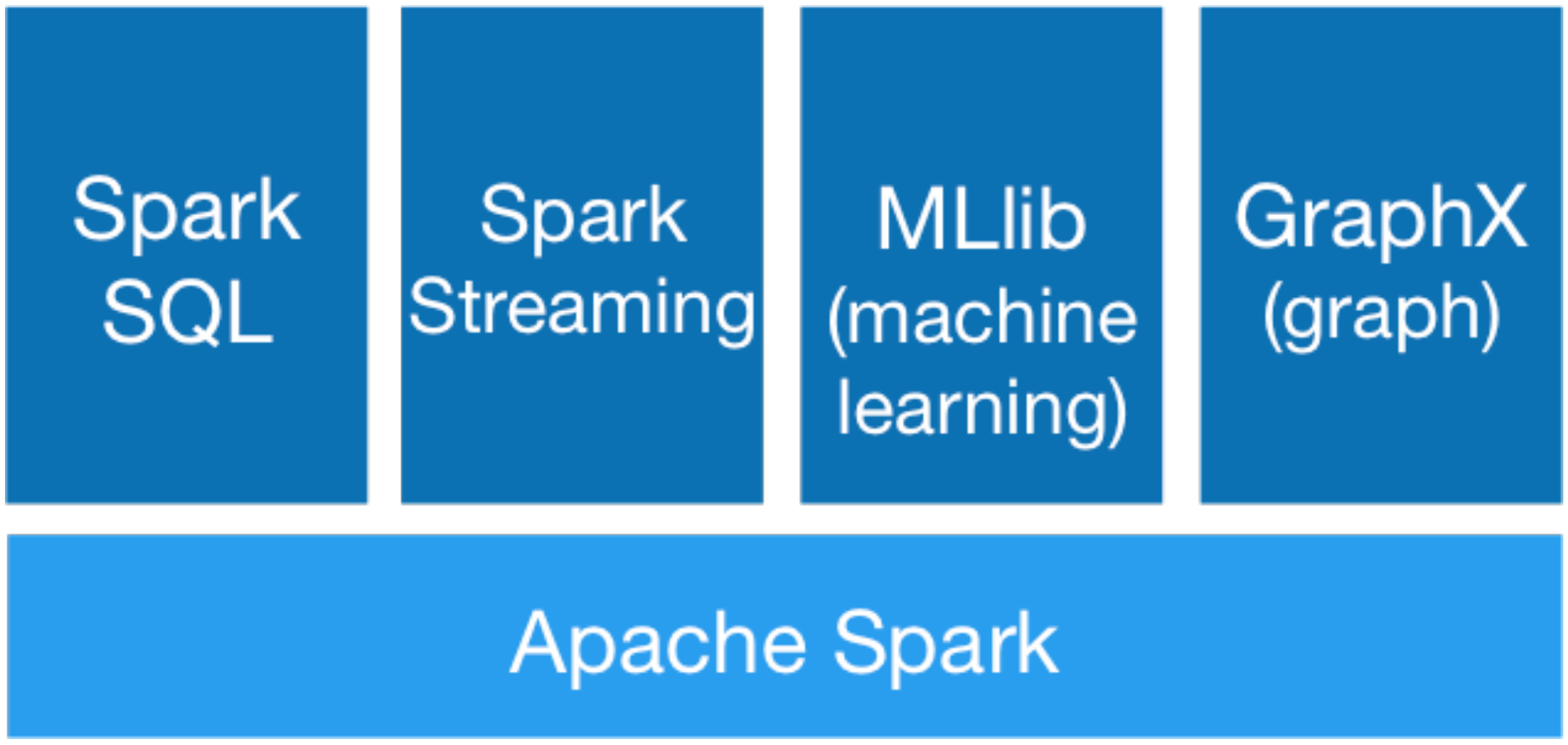
Can be integrated with Python

the spark stack



Source: http://spark.apache.org/

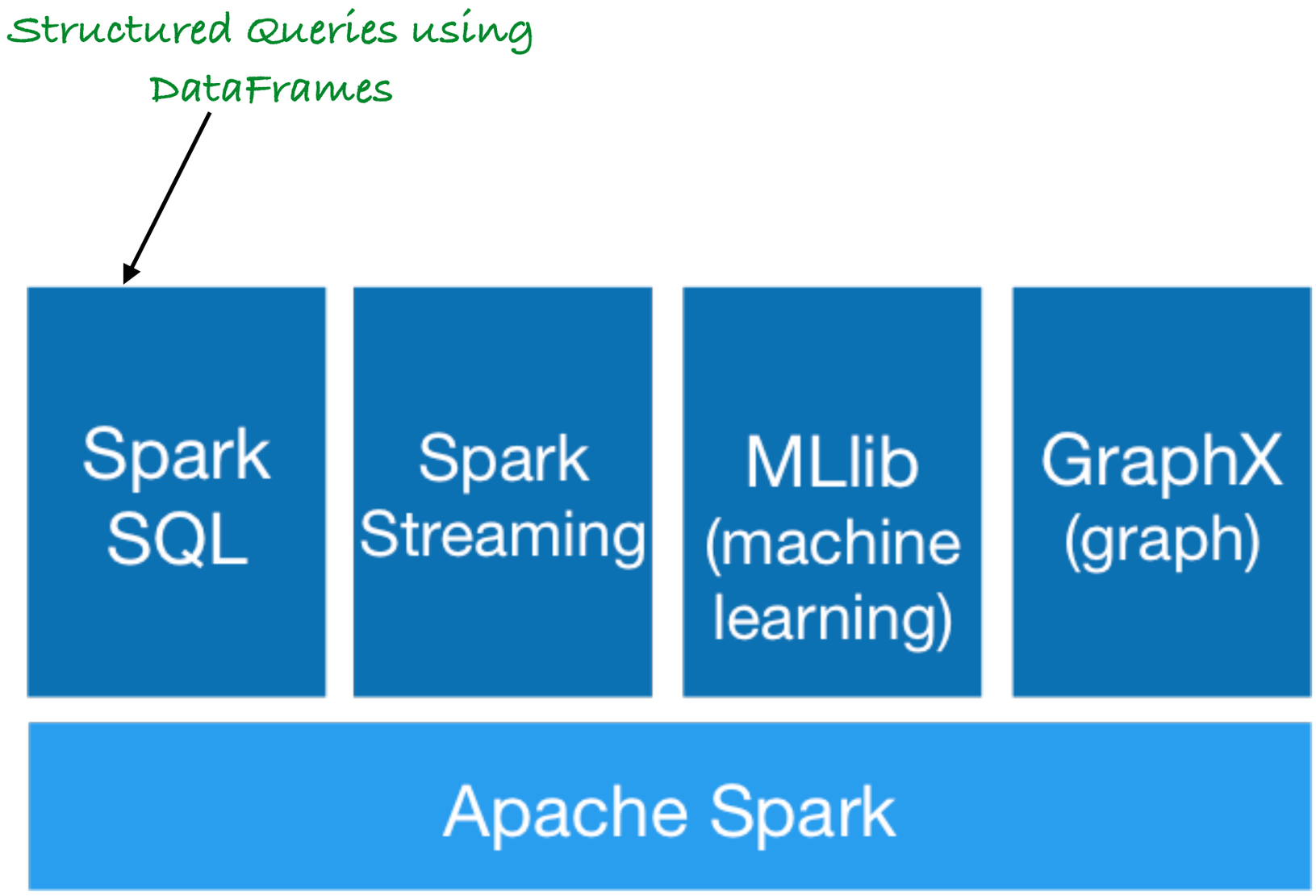
the spark stack



Spark core: task scheduling, memory management, fault recovery, storage, RDDs

Source: http://spark.apache.org/

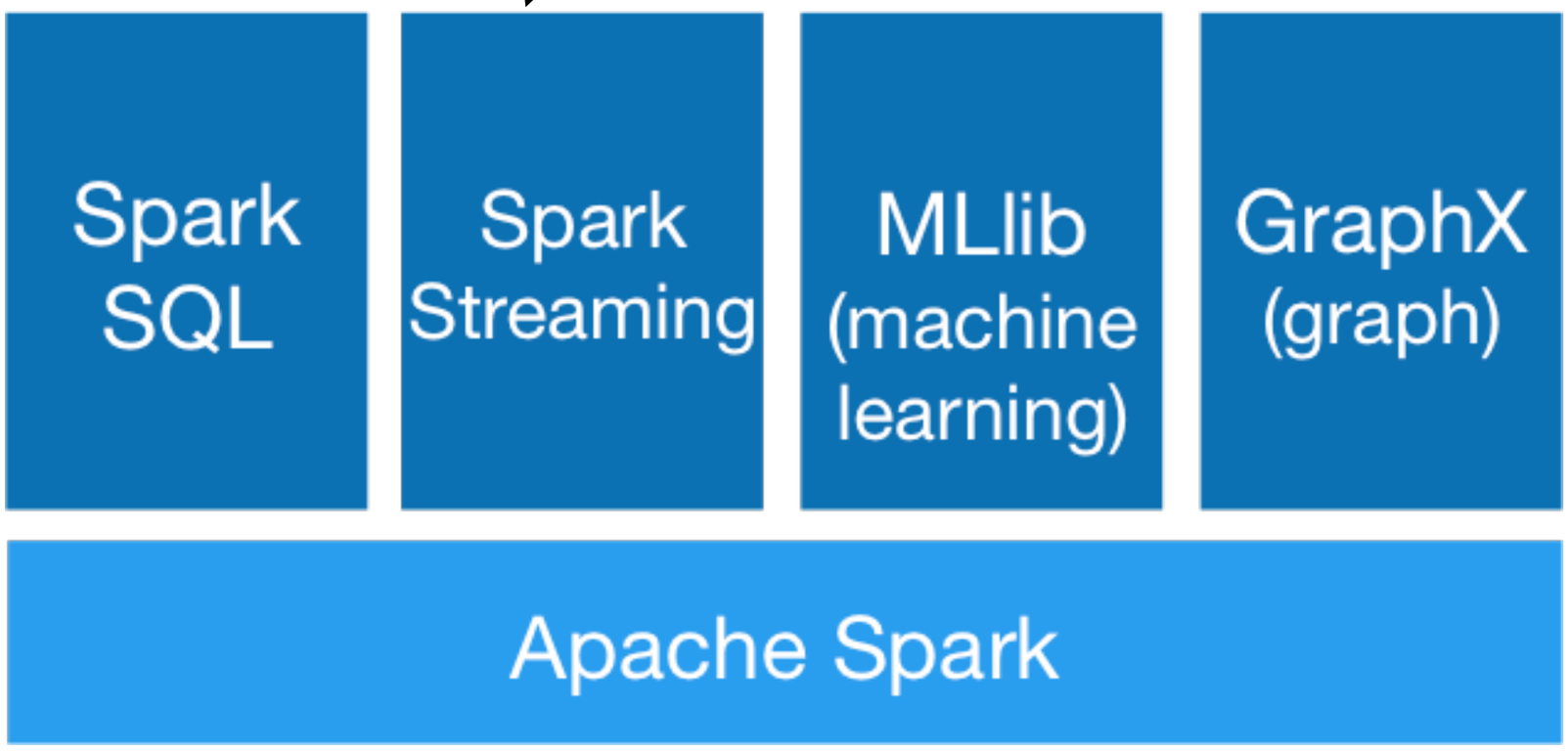
the spark stack



Source: http://spark.apache.org/

the spark stack

Live input data stream processing capability (log file streaming, message streaming, Twitter)



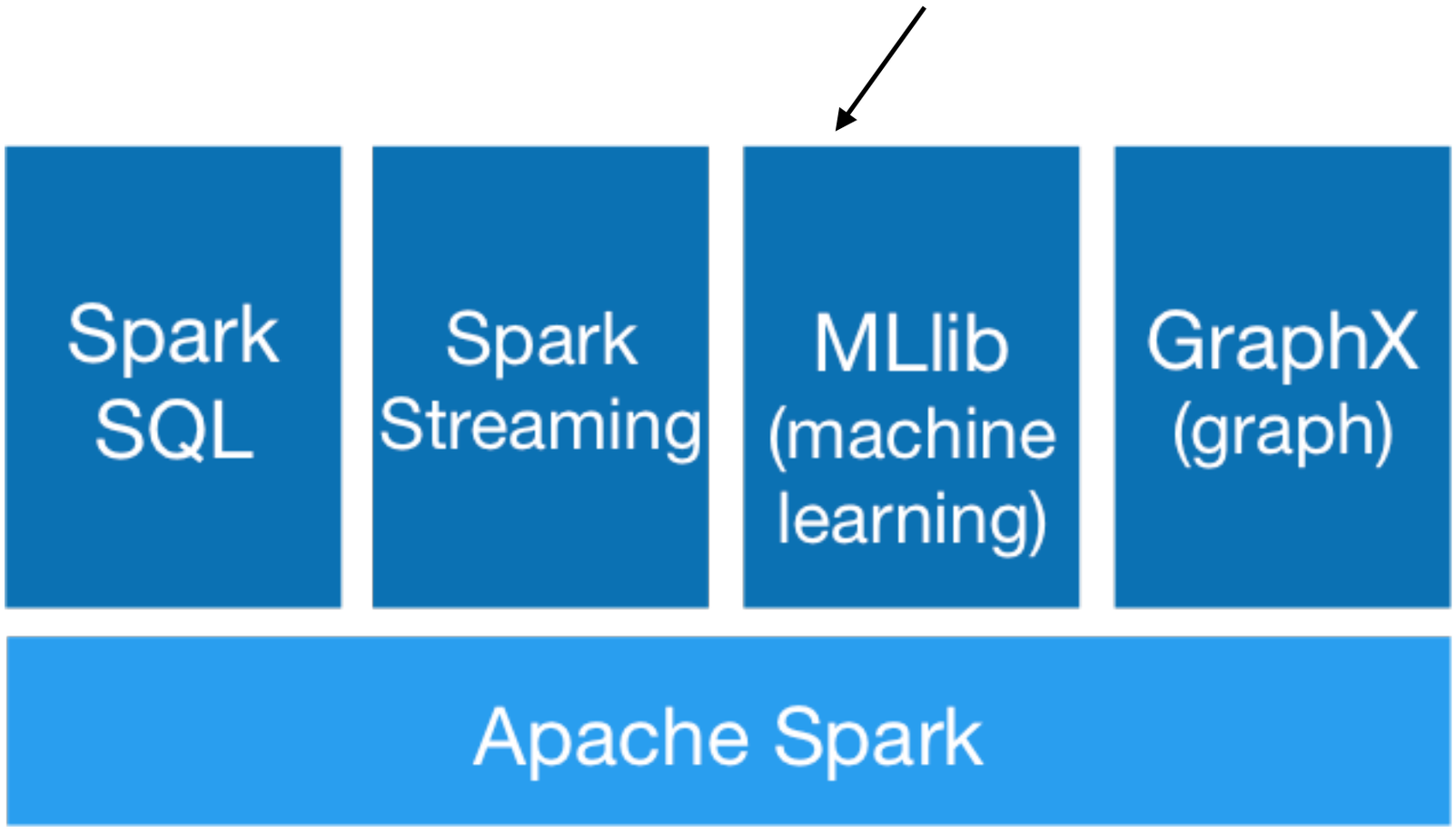
Source: http://spark.apache.org/

the spark stack

machine learning libraries:

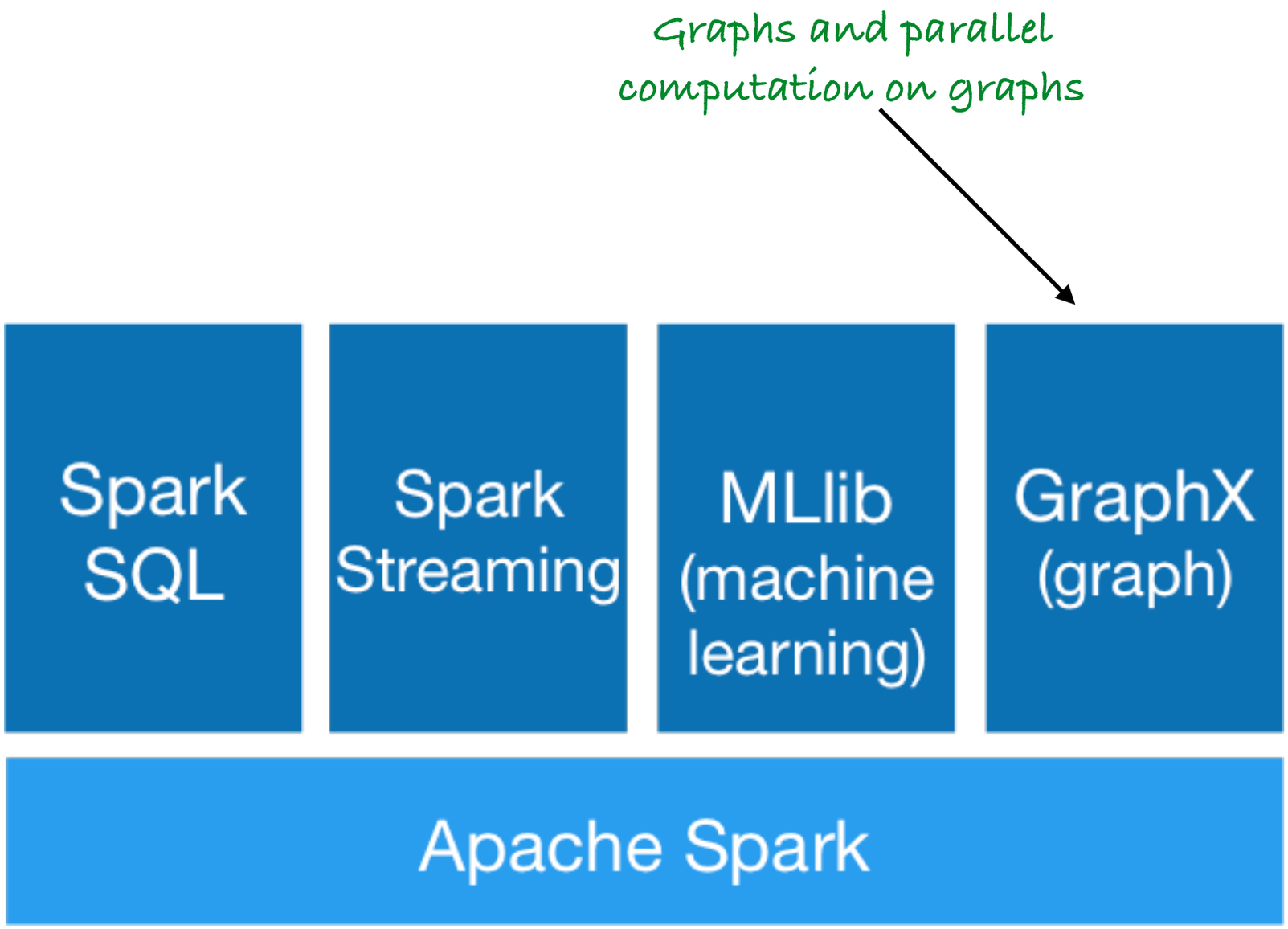
http://spark.apache.org/mllib/

classification, regression, collaborative filtering, clustering



Source: http://spark.apache.org/

the spark stack



Source: http://spark.apache.org/

Spark is becoming the
language of choice for
Data Science

This course: Analytics on the Cloud

* Scala: Our primary platform for functional programming

* Apache Spark: Cluster computing framework for analytics
  * Spark ML, Spark Streaming, Spark SQL, GraphX

* PySpark: Apache Spark integration with Python for building applications

* The cloud: Basics. Our focus is on analytics!
  * PaaS
  * SaaS

* Platform: Mostly local machines but also the google cloud platform