The Power of Spark

Spark is Central to Big Data Ecosystem and it's the first tool one should learn if one wants to work on Big data. It is the king of big data jungle

Why Learn Spark?

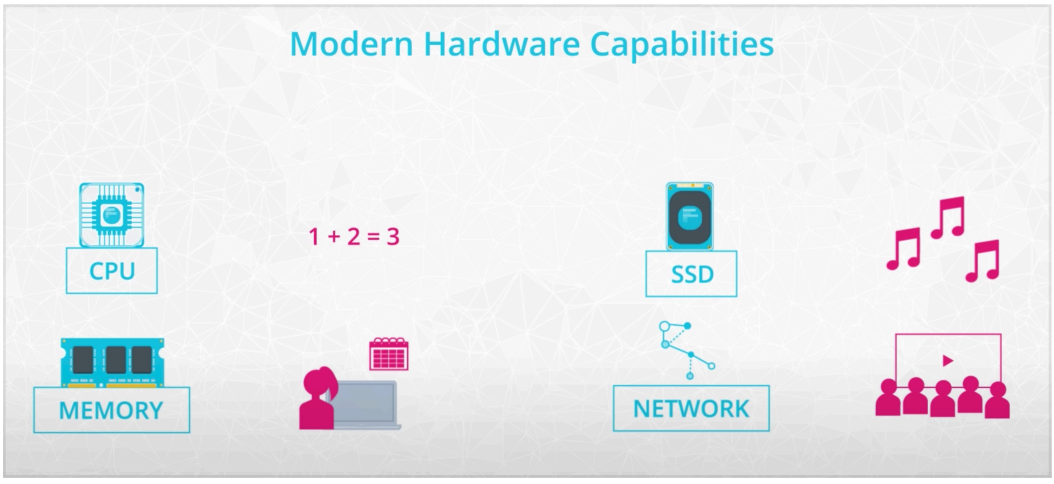
Spark is currently one of the most popular tools for big data analytics. You might have heard of other tools such as Hadoop. Hadoop is a slightly older technology although still in use by some companies. Spark is generally faster than Hadoop, which is why Spark has become more popular over the last few years.

There are many other big data tools and systems, each with its own use case. For example, there are database system like **Apache Cassandra** and SQL query engines like **Presto**. But Spark is still one of the most popular tools for analyzing large data sets.

Prerequisites

To be successful in this course, you should have the following skills and experience:

- Intermediate SQL skills and experience
- Intermediate Python programming skills
- Data modeling experience with relational and non-relational database design
- Amazon Web Services (AWS) knowledge and experience
- Knowledge of data warehouse architecture, dimensional modeling and creating OLAP cubes



Computing components that influence Big Data

Lesson Overview

In this lesson, you'll learn about big data and the implications for processing it with modern hardware, and how Spark fits into the big data ecosystem. Here is an outline of the topics we are covering in this lesson:

- What is big data?
- Review of the hardware behind big data

- Introduction to distributed systems
- Brief history of Spark and big data
- Common Spark use cases
- Other technologies in the big data ecosystem

Introduction to Big Data

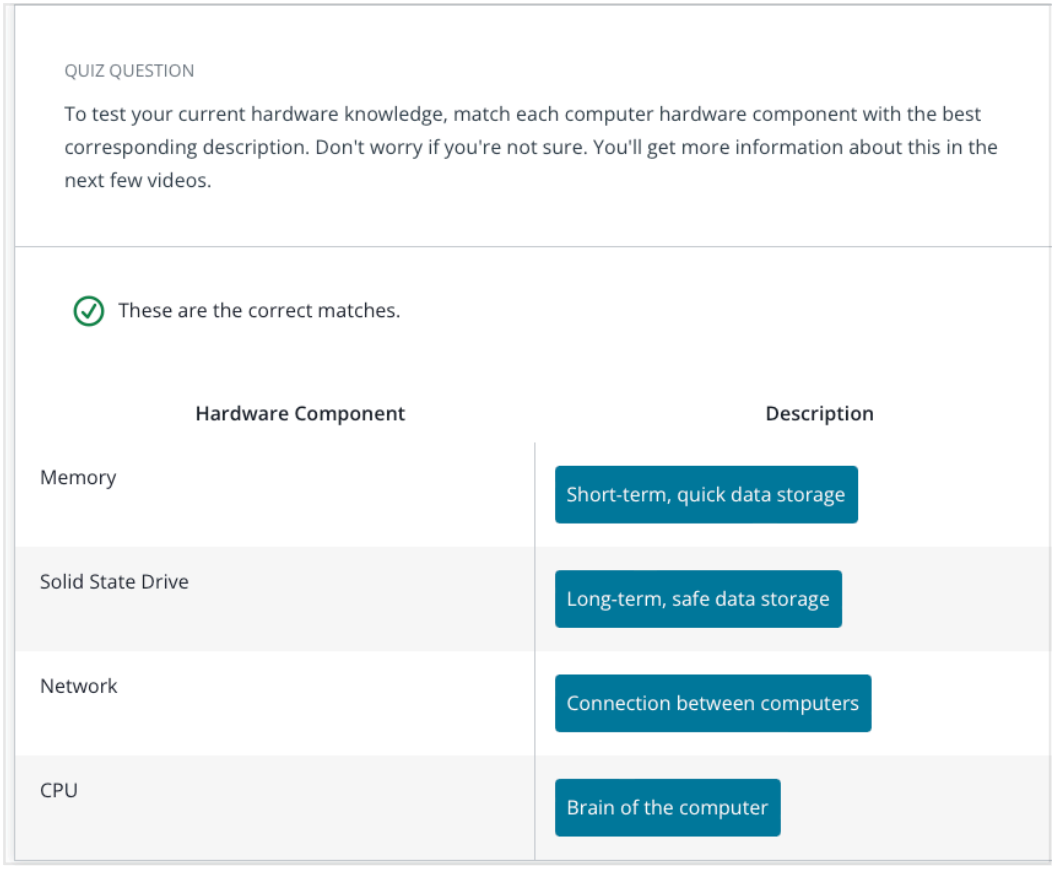
Big data is an overloaded term with no single definition, but most data scientists and engineers have a good sense of what we mean when we say, Spark is good for working with big data.

- You're working with big data when instead of using a single machine, it's easier to use a distributed system of multiple computers.

To understand when our data becomes big data, you need to be familiar with the capabilities of modern hardware.

- How long does it take for your CPU to add two numbers?
- How quickly can you look up an appointment if your calendar is already cached in your laptop's memory?
- How many seconds does it take to load your favorite song from your laptop's SSD storage
- How much data can you download from Netflix in a minute?

Knowing the rough answers to these types of questions is critical.



In the next few videos, you'll learn about four key hardware components. Understanding these components helps determine whether you are working on

a "big data" problem or if it's easier to analyze the data locally on your own computer.

CPU (Central Processing Unit)
The CPU is the "brain" of the computer. Every process on your computer is eventually handled by your CPU. This includes calculations and also instructions for the other components of the compute.

Memory (RAM)
When your program runs, data gets temporarily stored in memory before getting sent to the CPU. Memory is *ephemeral* storage - when your computer shuts down, the data in the memory is lost.

Storage (SSD or Magnetic Disk)
Storage is used for keeping data over long periods of time. When a program runs, the CPU will direct the memory to temporarily load data from long-term storage.

Network (LAN or the Internet)
Network is the gateway for anything that you need that isn't stored on your computer. The network could connect to other computers in the same room (a Local Area Network) or to a computer on the other side of the world, connected over the internet.

Other Numbers to Know?
You may have noticed a few other numbers involving the L1 and L2 Cache, mutex locking, and branch mispredicts. While these concepts are important for a detailed understanding of what's going on inside your computer, you don't need to worry about them for this course. If you're curious to learn more, check out **Peter Norvig's original blog post** from a few years ago, and **an interactive version** for today's current hardware.

QUIZ QUESTION

Rank the hardware component in order from fastest to slowest

These are the correct matches.

Speed	Hardware Component
Fastest	CPU
2nd Fastest	Memory (RAM)
3rd Fastest	Disk Storage (SSD)
Slowest	Network

The CPU is the brains of a computer. The CPU has a few different functions including directing other components of a computer as well as running mathematical calculations. The CPU can also store small amounts of data inside itself in what are called **registers**. These registers hold data that the CPU is working with at the moment.

For example, say you write a program that reads in a 40 MB data file and then analyzes the file. When you execute the code, the instructions are loaded into the CPU. The CPU then instructs the computer to take the 40 MB from disk and store the data in memory (RAM). If you want to sum a column of data, then the CPU will essentially take two numbers at a time and sum them together. The accumulation of the sum needs to be stored somewhere while the CPU grabs the next number.

This cumulative sum will be stored in a register. The registers make computations more efficient: the registers avoid having to send data unnecessarily back and forth between memory (RAM) and the CPU.

QUIZ QUESTION

A 2.5 Gigahertz CPU means that the CPU processes 2.5 billion operations per second. Let's say that for each operation, the CPU processes 8 bytes of data. How many bytes could this CPU process per second?

- ○ 312.5 million bytes per second
- ○ 3.2 billion bytes per second
- ✓ 20 billion bytes per second
- ○ I'm not sure how to calculate this

Nice work! 2.5 billion operations per second x 8 bytes per operation = 20 billion bytes per second

QUIZ QUESTION

Twitter generates about 6,000 tweets per second, and each tweet contains 200 bytes. So in one day, Twitter generates data on the order of:

(6000 tweets / second) x (86400 seconds / day) x (200 bytes / tweet) = 104 billion bytes / day

Knowing that tweets create approximately 104 billion bytes of data per day, how long would it take the 2.5 GigaHertz CPU to analyze a full day of tweets?

- ○ 0.19 seconds
- ○ 3.5 seconds
- ✓ 5.2 seconds
- ○ 47 seconds
- ○ 136 seconds

QUIZ QUESTION

What are the limitations of memory (RAM)?

- [x] RAM is relatively expensive
- [ ] You can only fit 8 GB of RAM onto a single computer
- [x] Data stored in RAM gets erased when the computer shuts down
- [ ] Operations in RAM are relatively inefficient compared to disk storage and network operations

While long-term storage like a hard drive disk is cheap and durable, it's much slower than memory.

- Loading data from magnetic disk can be 200 times slower.
- Even the newer solid-state drives known as SSDs, are still about 15 times slower.

Example:

- Processing an hour of tweets from memory would take about 30 milliseconds on my laptop. It will be closer to half a second if I had to load that data from my SSD storage, or four seconds if I had an older magnetic hard drive.

This difference between milliseconds and seconds may seem small, but once we're working with gigabytes or terabytes of data, it really adds up.

Spark was designed specifically to optimize our use of memory and avoid this problem.

Hardware: Networking

Unfortunately, the speed of our network has lagged behind the improvements in CPU memory and storage. As a result, moving data across the network from one machine to another is the most common bottleneck when working with big data.

- For example, the same hour of tweets that would take half a second to process from storage, would take 30 seconds to download from the Twitter API on a typical network.
- Network speeds depend on a lot of factors like if your roommate is using Netflix at the same time.
- It usually takes at least 20 times longer to process data, when you have to download it from another machine first.

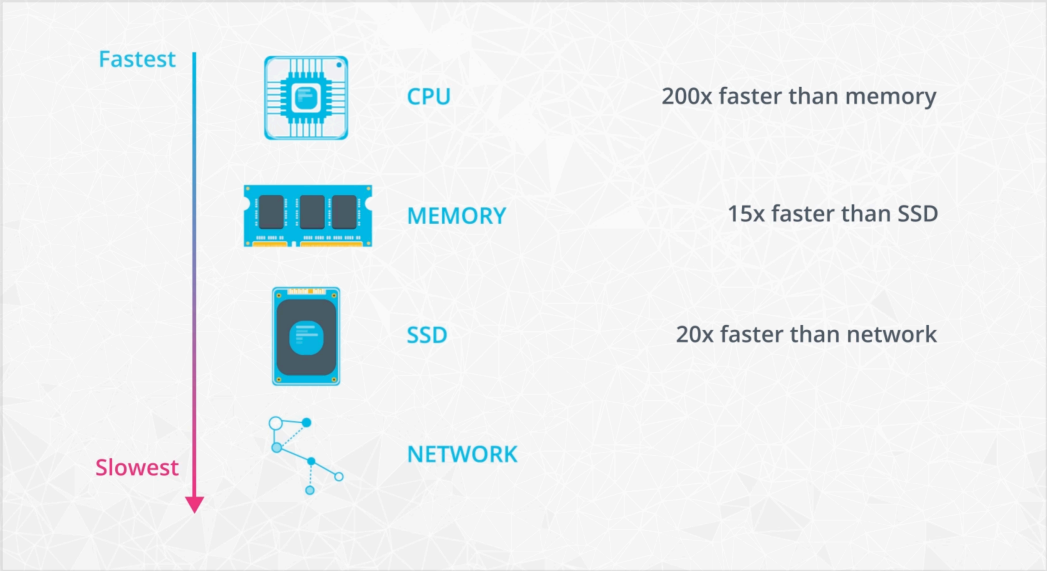
- Distributed systems try to minimize shuffling data back and forth across different computers.

Since Spark and any distributed technology for that matter uses the cluster of servers connected by a network, we can't entirely avoid moving data around.

QUIZ QUESTION

Which component is the biggest bottleneck when working with big data?

- [x] Transferring data across a network
- [ ] Reading to and writing from disk storage (SSD)
- [ ] Reading to and writing from memory (RAM)
- [ ] CPU operations



Fastest and slowest computing component

By going through the Twitter example, you're more familiar with the different aspects of hardware. Let's put all these numbers together.

- The fastest component of your hardware is the CPU
- Memory is about 15 times faster than long-term storage like SSDs.
- The network is the worst bottleneck.
- It's about 20 times slower to work with data when you have to download it first.

If you know these ratios, you'll know when you have big data, and when to use a distributed technology like Spark.

One of the first steps in every data problem is **extraction**, pulling out just the data you need.

- For example, your content team needs to figure out which artists were played the most in Canada in the last year.
- Every time a user listens to a song in the past year, the Sparkify app wrote a single record in a big log file, which you downloaded on your laptop overnight.

In this case, we only need the artist for Canadian users, so our processing is pretty simple. You filter out all the non Canadian users, and extract the artist field. Since Sparkify is still small, and you only have a few thousand users, this data is modest in size. The last year of data is only four gigs, so it can easily fit into the eight gigabytes of memory on your laptop. This means that you can load all the data sequentially in your memory, and you can write a simple Python program to have your CPU filter and extract the artists.

Since you're at an adventurous startup, willing to try anything, you spend a while thinking if there's an easier way to process the data.

- Perhaps, emailing a month of data to 12 of your co-worker's laptops. They could each run your Python program, and then email you back the results when they're done. But this wouldn't be worth it. Emailing the data across the network is the slowest part of this process.
- What if another co-worker was busy, and took an extra 20 minutes to start the program?

Even if you have 11 months of data, you don't know the most popular artist until you have all the results back. These are some of the problems you run into when you have distributed systems like Spark. In this case, it's easier to stick to Python processing on one computer. So, according to our definition, you don't have big data.

Over the next year, you and the rest of the Sparkify team work very hard to become a worldwide success with millions of customers streaming music every day. With several of the content deals expiring soon, the content team needs you to make the same report of the most popular artists played by your Canadian users. But after a year of stellar growth, the log of songs exploded to 200 gigabytes, way beyond the four gigs that was last year. So, what happens when you try using the same Python program that worked so well the previous year?

- When you run the program, which should read in records of song activity for all users and spit out the artists for the Canadian users, it seems to go well for a few seconds, but then your entire laptop seems to screech to a halt. After a few more minutes of this, you give up and kill the Python program, which releases your system from this slow

motion. What happened?

-

QUIZ QUESTION

What happened to your computer when you started to run the program with the larger data set? Keep in mind that in this current configuration, the data set is stored on your local computer and does not need to travel across a network (Select all that apply)

- [ ] It took too long to move data across the network
- [x] The CPU couldn't load data quickly from memory
- [x] The memory couldn't load data quickly from storage
- [ ] The CPU simply couldn't handle the larger data set
- [ ] The storage couldn't load data quickly from the network
- [ ] The CPU couldn't load data quickly from the network

If a dataset is larger than the size of your RAM, you might still be able to analyze the data on a single computer. By default, the Python pandas library will read in an entire dataset from disk into memory. If the dataset is larger than your computer's memory, the program won't work.
However, the Python pandas library can read in a file in smaller chunks. Thus, if you were going to calculate summary statistics about the dataset such as a sum or count, you could read in a part of the dataset at a time and accumulate the sum or count.
Here is an example of how this works.

Distributed systems, distributed programming, and distributed algorithms originally referred to computer networks, where individual computers are physically distributed within some geographical area and communicate with each other via message passing. An early example from the 70s, is ARPANET's Email Application.

- In a network topology, which is basically a graph, each node has its own private memory and processor and it communicates with the other nodes via messages.
- Some of you might have experienced with parallel computing using tools, such as OpenMP or OpenCL.
- You can think of parallel computing as a particular tightly coupled form

of distributed computing. The main difference is, that in parallel computing all processors may have access to a shared memory to exchange information.

Note: Great work! At a high level, distributed computing implies multiple CPUs each with its own memory. Parallel computing uses multiple CPUs sharing the same memory.

QUIZ QUESTION

Mark the statements that are true.

- [ ] Generally speaking, distributed computing assumes that multiple CPUs are sharing a single source of memory.
- [ ] Parallel computing is another way of saying distributed computing
- [x] In general parallel computing implies multiple CPUs share the same memory.
- [x] With distributed computing, each CPU has its own memory.
- [x] In distributed computing, each computer/machine is connected to the other machines across a network.

Hadoop Vocabulary

Here is a list of some terms associated with Hadoop. You'll learn more about these terms and how they relate to Spark in the rest of the lesson.

- **Hadoop** - an ecosystem of tools for big data storage and data analysis. Hadoop is an older system than Spark but is still used by many companies. The major difference between Spark and Hadoop is how they use memory. Hadoop writes intermediate results to disk whereas Spark tries to keep data in memory whenever possible. This makes Spark faster for many use cases.
- **Hadoop MapReduce** - a system for processing and analyzing large data sets in parallel.
- **Hadoop YARN** - a resource manager that schedules jobs across a cluster. The manager keeps track of what computer resources are available and then assigns those resources to specific tasks.
- **Hadoop Distributed File System (HDFS)** - a big data storage system that splits data into chunks and stores the chunks across a cluster of computers.

As Hadoop matured, other tools were developed to make Hadoop easier to work with. These tools included:

- **Apache Pig** - a SQL-like language that runs on top of Hadoop

MapReduce
- **Apache Hive** - another SQL-like interface that runs on top of Hadoop MapReduce

Oftentimes when someone is talking about Hadoop in general terms, they are actually talking about Hadoop MapReduce. However, Hadoop is more than just MapReduce. In the next part of the lesson, you'll learn more about how MapReduce works.

How is Spark related to Hadoop?

Spark, which is the main focus of this course, is another big data framework. Spark contains libraries for data analysis, machine learning, graph analysis, and streaming live data. Spark is generally faster than Hadoop. This is because Hadoop writes intermediate results to disk whereas Spark tries to keep intermediate results in memory whenever possible.

The Hadoop ecosystem includes a distributed file storage system called HDFS (Hadoop Distributed File System). Spark, on the other hand, does not include a file storage system. You can use Spark on top of HDFS but you do not have to. Spark can read in data from other sources as well such as **Amazon S3**.

Streaming Data

Data streaming is a specialized topic in big data. The use case is when you want to store and analyze data in real-time such as Facebook posts or Twitter tweets.
Spark has a streaming library called **Spark Streaming** although it is not as popular and fast as some other streaming libraries. Other popular streaming libraries include **Storm** and **Flink**. Streaming won't be covered in this course, but you can follow these links to learn more about these technologies.

MapReduce

MapReduce is a programming technique for manipulating large data sets. "Hadoop MapReduce" is a specific implementation of this programming technique.

The technique works by first dividing up a large dataset and distributing the data across a cluster. In the map step, each data is analyzed and converted into a (key, value) pair. Then these key-value pairs are shuffled across the cluster so that all keys are on the same machine. In the reduce step, the values with the same keys are combined together.

While Spark doesn't implement MapReduce, you can write Spark programs that behave in a similar way to the map-reduce paradigm. In the next section, you will run through a code example.

QUIZ QUESTION

In the map-reduce paradigm, what happens in the shuffle step?

- [ ] The data gets randomly shuffled for cross validation purposes.
- [ ] Calculations are done to find the sum of all values with the same key.
- [ ] Each value in the data set goes through some mathematical operation.
- [x] Data points with the same key get moved to the same cluster node.

QUIZ QUESTION

What operations does the map step of the MapReduce program perform?

- ( ) It creates a list of all the songs in the file
- (x) The code reads in the file one line at a time and the map steps outputs a tuple for each song play.
- ( ) It adds up all of the song plays and produces a total number of how many times each song was played.

QUIZ QUESTION

What operations does the reduce step of the MapReduce program perform?

- [x] The reduce step combines all tuples with the same key (the song name) and sums all the values of the tuple, which outputs the total song plays for each song.
- [ ] The reduce step combines all tuples with the same key (the song name) and and outputs how many unique songs there are in the tuples
- [ ] It reduces the contents in the file to one line so you can filter the tuples from the map step.

The Spark Cluster

When we talk about distributed computing, we generally refer to a big computational job executing across a cluster of nodes. Each node is responsible for a set of operations on a subset of the data. At the end, we combine these partial results to get the final answer. But how do the nodes know which task to run and demote order? Are all nodes equal? Which machine are you interacting with when you run your code?
Most computational frameworks are organized into a master-worker hierarchy:

- The master node is responsible for orchestrating the tasks across the cluster
- Workers are performing the actual computations

There are four different modes to setup Spark:

- Local mode: In this case, everything happens on a single machine. So, while we use spark's APIs, we don't really do any distributed computing. The local mode can be useful to learn syntax and to prototype your project.

The other three modes are distributed and declare a cluster manager. The cluster manager is a separate process that monitors available resources and makes sure that all machines are responsive during the job. There are three different options of cluster managers:

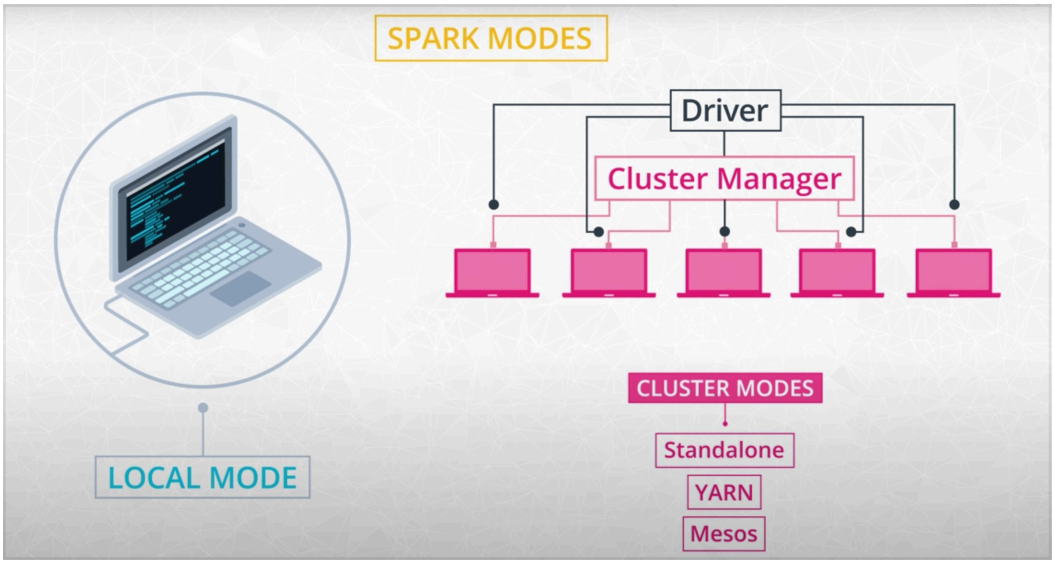
- Spark's own Standalone Customer Manager
- In this course, you will set up and use your own distributed Spark cluster using Standalone mode.

- In Spark's Standalone mode there is a Driver Process. If you open a Spark shell, either Python or Scala, you are directly interacting with the driver program. It acts as the master and is responsible for scheduling tasks
- YARN from the Hadoop project
- Another open-source manager from UC Berkeley's AMPLab Coordinators.

QUIZ QUESTION

In which cases would you use Local Mode?

- [ ] When you are working with a computer cluster that only has two machines
- [ ] For distributed computing across a cluster
- [x] When you are working with Spark installed on your own laptop
- [ ] There is never a reason to use local mode since you're not taking advantage of Spark's ability to work on a cluster: you should be using Standalone, YARN, or Mesos



Spark clusters in local mode and standalone modes

Spark Use Cases and Resources

Here are a few resources about different Spark use cases:

- Data Analytics
- Machine Learning
- Streaming
- Graph Analytics

You Don't Always Need Spark

Spark is meant for big data sets that cannot fit on one computer. But you don't need Spark if you are working on smaller data sets. In the cases of data sets that can fit on your local computer, there are many other options out there you can use to manipulate data such as:

- AWK - a command line tool for manipulating text files
- R - a programming language and software environment for statistical computing
- Python data libraries, which include Pandas, Matplotlib, NumPy, and scikit-learn among other libraries

Sometimes, you can still use pandas on a single, local machine even if your data set is only a little bit larger than memory. Pandas can read data in chunks. Depending on your use case, you can filter the data and write out the relevant parts to disk.

If the data is already stored in a relational database such as MySQL or Postgres, you can leverage SQL to extract, filter and aggregate the data. If you would like to leverage pandas and SQL simultaneously, you can use libraries such as SQLAlchemy, which provides an abstraction layer to manipulate SQL tables with generative Python expressions.

The most commonly used Python Machine Learning library is scikit-learn. It has a wide range of algorithms for classification, regression, and clustering, as well as utilities for preprocessing data, fine tuning model parameters and testing their results. However, if you want to use more complex algorithms - like deep learning - you'll need to look further. TensorFlow and PyTorch are currently popular packages.

Spark's Limitations

Spark has some limitation.
Spark Streaming's latency is at least 500 milliseconds since it operates on micro-batches of records, instead of processing one record at a time. Native streaming tools such as Storm, Apex, or Flink can push down this latency value and might be more suitable for low-latency applications. Flink and Apex can be used for batch computation as well, so if you're already using them for stream processing, there's no need to add Spark to your stack of technologies.

Another limitation of Spark is its selection of machine learning algorithms. Currently, Spark only supports algorithms that scale linearly with the input data size. In general, deep learning is not available either, though there are many projects that integrate Spark with Tensorflow and other deep learning tools.

Hadoop versus Spark

The Hadoop ecosystem is a slightly older technology than the Spark

ecosystem. In general, Hadoop MapReduce is slower than Spark because Hadoop writes data out to disk during intermediate steps. However, many big companies, such as Facebook and LinkedIn, started using Big Data early and built their infrastructure around the Hadoop ecosystem.

While Spark is great for iterative algorithms, there is not much of a performance boost over Hadoop MapReduce when doing simple counting. Migrating legacy code to Spark, especially on hundreds of nodes that are already in production, might not be worth the cost for the small performance boost.

Beyond Spark for Storing and Processing Big Data

Keep in mind that Spark is not a data storage system, and there are a number of tools besides Spark that can be used to process and analyze large datasets.

Sometimes it makes sense to use the power and simplicity of SQL on big data. For these cases, a new class of databases, know as NoSQL and NewSQL, have been developed.

For example, you might hear about newer database storage systems like HBase or Cassandra. There are also distributed SQL engines like Impala and Presto. Many of these technologies use query syntax that you are likely already familiar with based on your experiences with Python and SQL.

In the lessons ahead, you will learn about Spark specifically, but know that many of the skills you already have with SQL, Python, and soon enough, Spark, will also be useful if you end up needing to learn any of these additional Big Data tools.

Congratulations. You have completed the first lesson on Spark! In this lesson, we have discussed:

- What we mean by big data
- Spark's typical use cases, and you should, or shouldn't use Spark
- Tools surrounding Spark, in the big data ecosystem.

Now that you have a high-level understanding of the big data ecosystem, in the next lesson, we'll take a closer look at how to use Spark, to solve real-world big data problems.

Beyond Spark for Storing and Processing Big Data

Keep in mind that Spark is not a data storage system, and there are a number of tools besides Spark that can be used to process and analyze large datasets.

- HBase
- Cassandra
- Impala
- Presto