livelessons
video instruction from technology experts

Lesson 4: Spark Internals

4.1 Introduction to Distributed Systems

What we have Accomplished

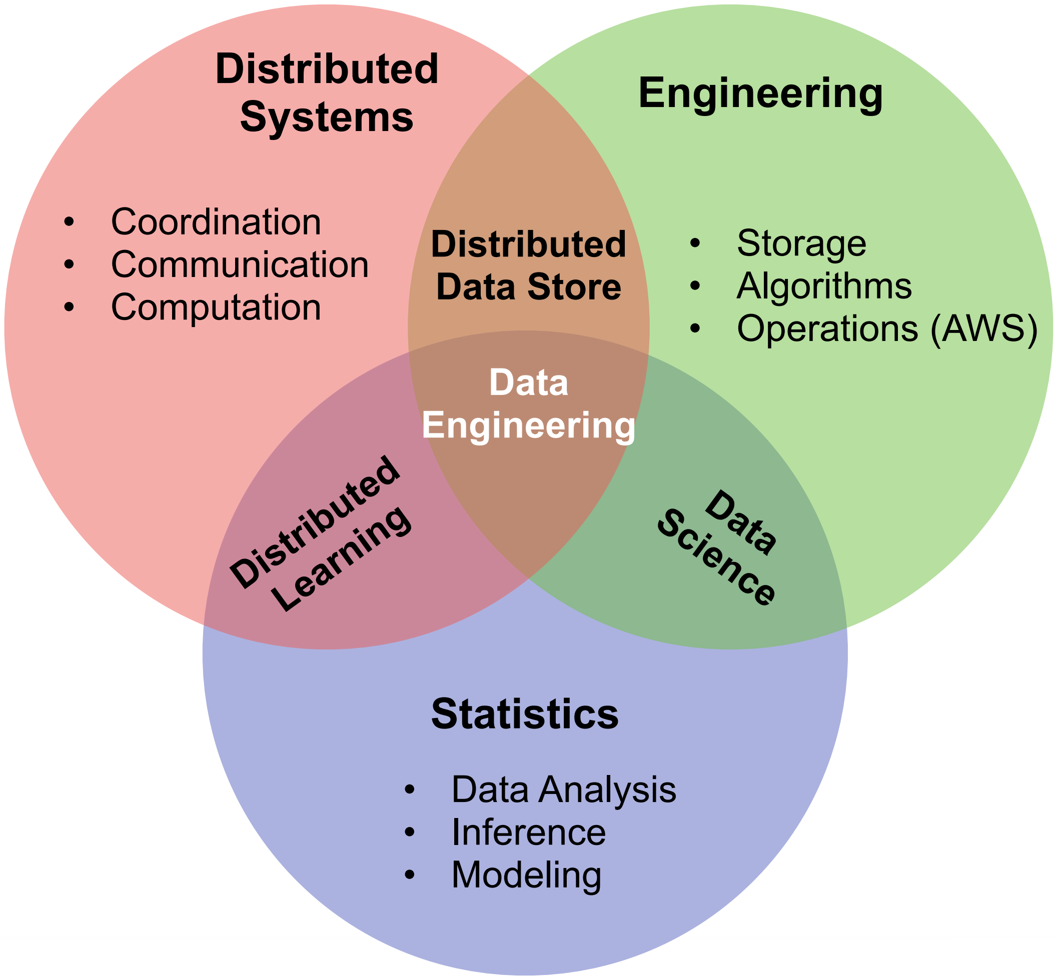
- Setup a local Spark environment and learned the basics of the framework
- Experienced the tradeoffs of each of the various Spark programming APIs
- Built a complete end-to-end application with Spark

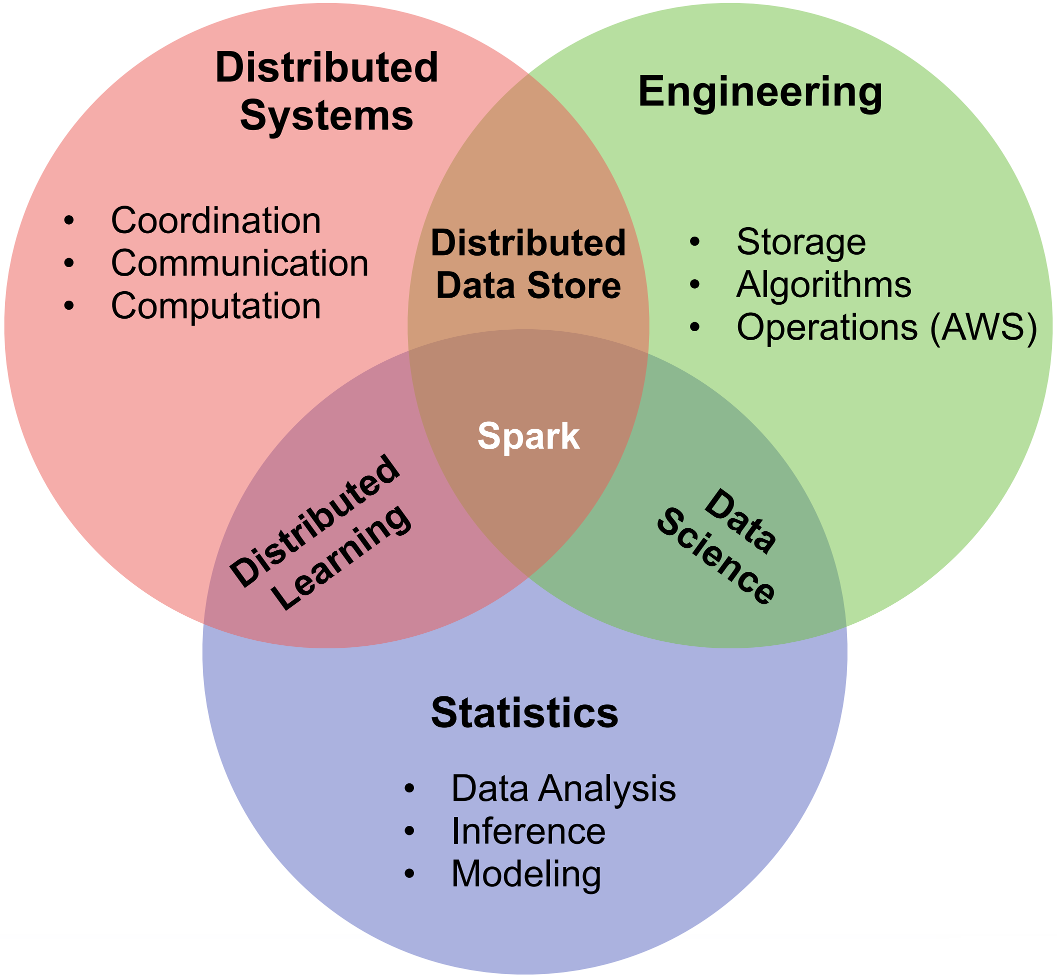
The Data Engineer lives in the liminal space between distributed systems (CS theory), engineering (operations/infrastructure), and statistics. While it is not crucial for them to understand each of these domains wholly, being able to synthesize concepts from each is essential for success…

- Jonathan Dinu

And Spark lives exactly at this nexus! Making it one of the most powerful frameworks for data scientists and engineers alike!

- Jonathan Dinu


Distributed Systems
• Coordination
• Communication
• Computation
Engineering
• Storage
• Algorithms
• Operations (AWS)
Distributed Data Store
Data Engineering
Distributed Learning
Data Science
Statistics
• Data Analysis
• Inference
• Modeling


Distributed Systems
• Coordination
• Communication
• Computation
Engineering
• Storage
• Algorithms
• Operations (AWS)
Distributed Data Store
Spark
Distributed Learning
Data Science
Statistics
• Data Analysis
• Inference
• Modeling

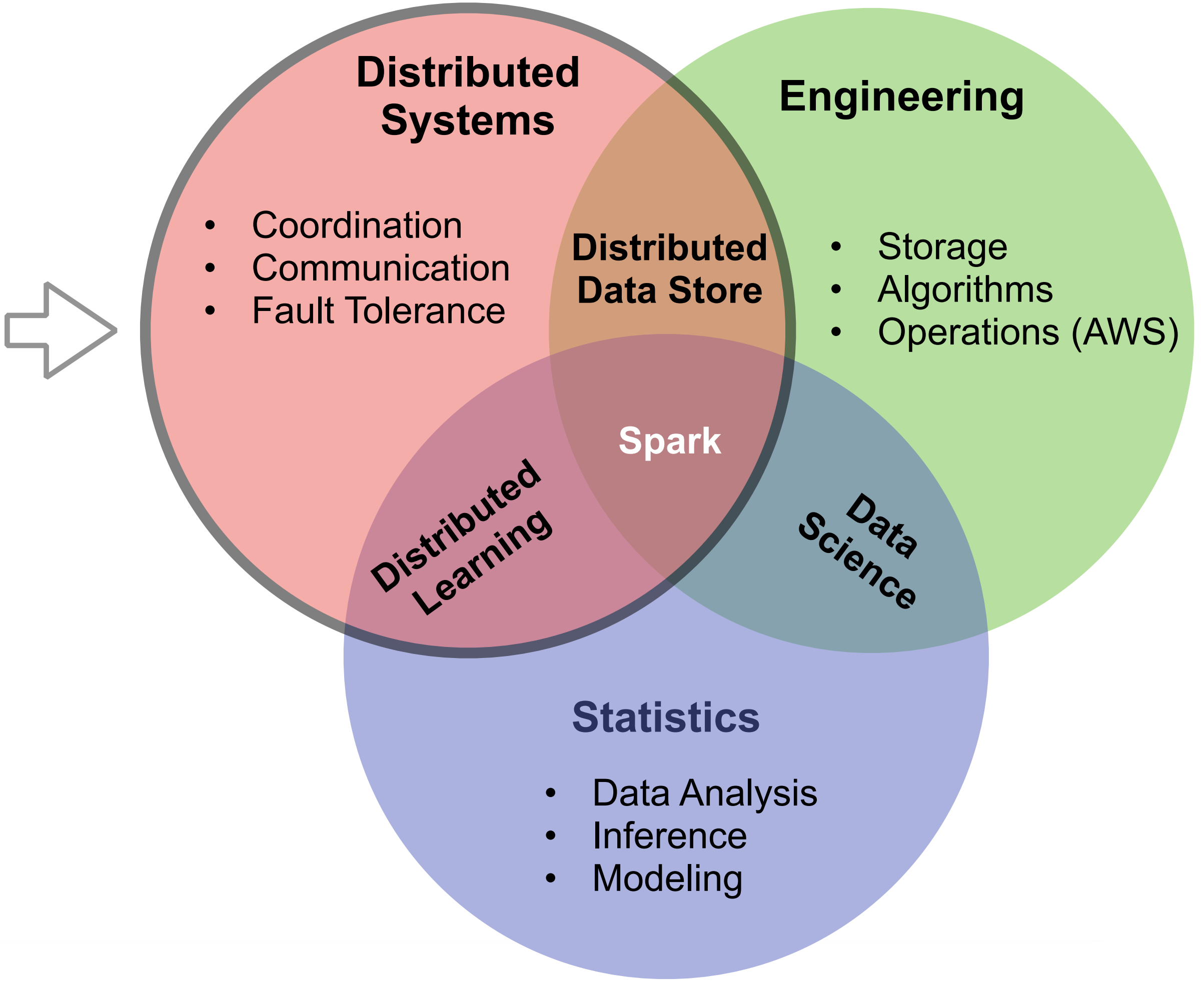
So why do I need to know this?

- **Logic:** if you understand the system you are programming, you can reason about your code much more effectively.

- **Debugging:** if you understand the system you are programming, you can fix your code much faster.

- **Performance:** if you understand the system you are programming, you can write smarter (and more optimal) code.

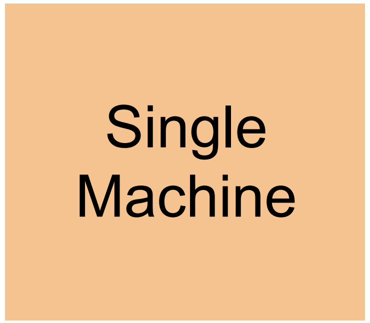
We are Here


Distributed Systems
• Coordination
• Communication
• Fault Tolerance
Engineering
• Storage
• Algorithms
• Operations (AWS)
Distributed Data Store
Spark
Distributed Learning
Data Science
Statistics
• Data Analysis
• Inference
• Modeling

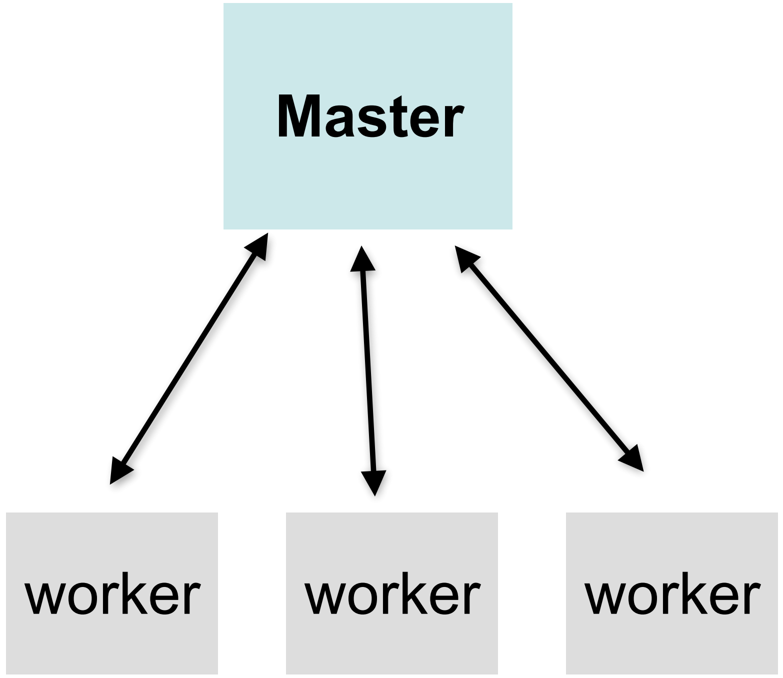
Distributed programming is the art of solving the same problem that you can solve on a single computer using multiple computers.

- “Distributed systems: for fun and profit”, Mikito Takada

Local



Distributed



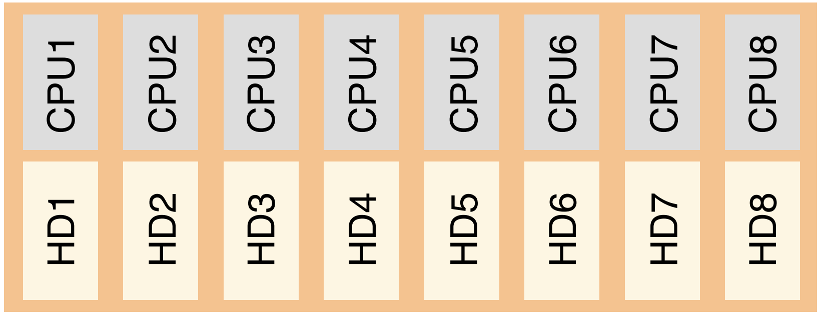
Why Distribute?

- Problem no longer fits on a single machine (storage)
- Problem need to be solved faster (computation)
- Often a combination of the two.

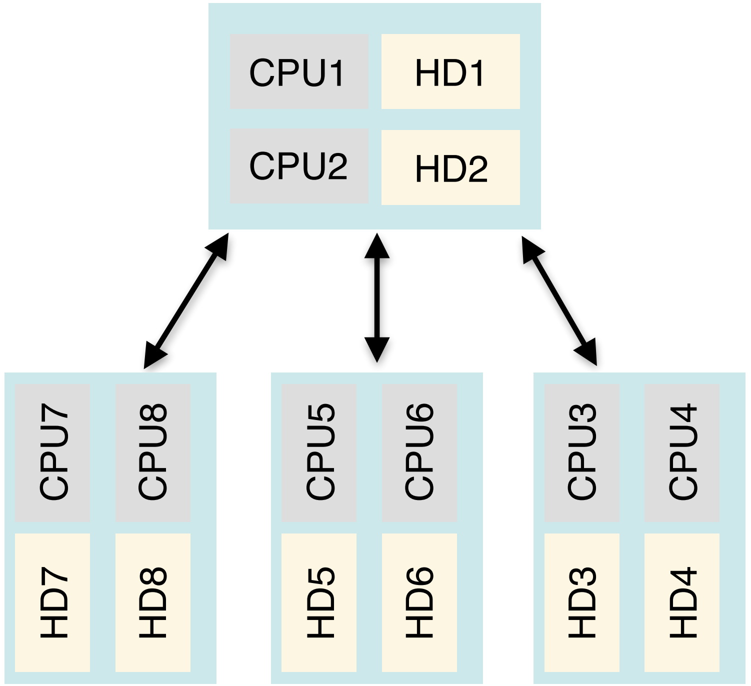
Availability is also often sought after and achieved by distributing a system but in the context of Spark we will concern ourselves with storage and computation since its programing model can be thought to be always available.

How to Distribute

Local



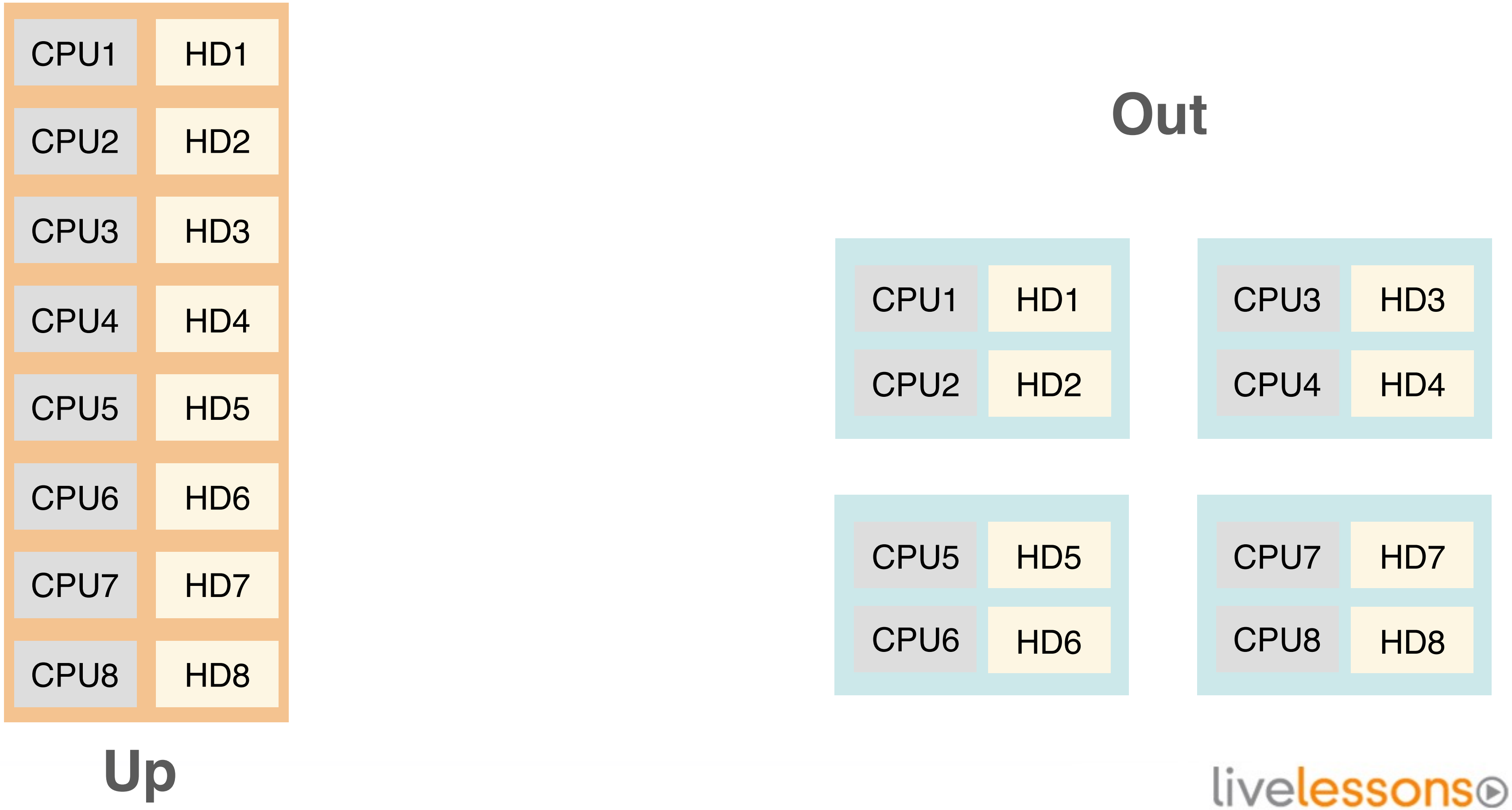
Distributed



Storage: Hard drives

Computation: CPUs

Scale Up versus Out



Problem Solved?

- **Coordination:** what happens when and to whom? And how do you synchronize these events
- **Communication:** how do the different nodes in your system talk to one another? And how do you talk to your nodes?
- **Fault Tolerance:** is your system robust to network and machine failures (because they will happen)?