



Optimizing Apache Spark

## The Five Most Common Performance Problems

# The 5 Most Common Performance Problems (The 5 Ss)

## Five Basic Problems

The most egregious problems fall into one of five categories:

- **Spill**: The writing of temp files to disk due to a lack of memory
- **Skew**: An imbalance in the size of partitions
- **Shuffle**: The act of moving data between executors
- **Storage**: A set of problems directly related to how data is stored on disk
- **Serialization**: The distribution of code segments across the cluster

# The 5 Most Common Performance Problems (The 5 Ss)

## Five Basic Problems – Why it's hard

- Root sourcing problems is hard when one problem can causes another
- **Skew** can induce **Spill**
- **Storage** issues can induce excess **Shuffle**
- Incorrectly addressing **Shuffle** can exacerbate **Skew**
- Many of these problems can be present at the same time
- To better illustrate this problem...  
let's take a quick look at how we benchmark our experiments



Optimizing Apache Spark

The Five Most Common  
Performance Problems

Benchmarking

# The 5 Most Common Performance Problems (The 5 Ss)

## Benchmarking

There are generally three common approaches to benchmarking:

- The **count()** action
- The **foreach()** action with a do-nothing lambda
- A **noop** (or no operation) write

We can see how these three strategies differ with our [Spark UI Simulator](#)

# The 5 Most Common Performance Problems (The 5 Ss)

## Benchmarking – In Action, Part 1

See [Experiment #5980](#)

- Compare **Step B-1** and **Step B-2**
  - Note the total duration
  - Why did **Step B-1** take 2x longer than **Step B-2**?
- See **Step C**, the **count ()** operation
  - Note the duration
  - Note that the Python and Scala samples are nearly identical
  - Note the number of jobs
  - Why is there one less job as compared to **Step B-2**?

# The 5 Most Common Performance Problems (The 5 Ss)

## Benchmarking - In Action, Part 2

See [Experiment #5980](#)

- See **Step D**, the **foreach ()** action with a do-nothing lambda
  - Note the total duration (esp compared to the **count ()** action)
  - Compare the Scala a Python versions
  - Why is the Python version significantly slower than the Scala version?
- See **Step E**, the **noop** write.
  - Note the total duration of both the Python and Scala

# The 5 Most Common Performance Problems (The 5 Ss)

## Benchmarking – Review

- About **Step B-1** and **Step B-2**
  - Loading the schema in **Step B-1** and not **Step B-2** provided a side effect
- About the **count()** action
  - Count is optimized - doesn't process all the data
  - Metadata & columnar reads affect execution
- About the **foreach()** action
  - Simulates processing of every record
  - The serialization side effect is quite significant in Python
- About the **noop** with a schema - it just works as expected!



