



# SmartCat

DATA → KNOWLEDGE → POWER

# When DataFrames fail, resort to mapPartitions

aka “What to do when Spark can’t handle it”

Matija Gobec  
matija.gobec@smärtcat.io  
@mad\_max0204

Berlin Buzzwords, June 2018

**SmartCat.io**

# Why this talk?



Complex problem with an easy solution  
(or not)



Logically correct vs Correct

# Logically correct solution

- Data analysis process (data exploration)
- Get to the correct output ASAP
- Validate result
- Run on subset (if possible)
  - Quick iterations
  - Tends to eliminate data issues



# Correct solution

- Correct result
- Repeatable performance
- Scalable and optimized
- Proper data handling





Let's dig in

# spark.read.jdbc

What in the world?

Specify `columnName` for proper distribution

`numPartitions` defines parallelism

Query for bounds (`upper` and `lower`)

```
val df = (spark.read.jdbc(url=jdbcUrl,  
    table="some_table",  
    columnName="some_column",  
    lowerBound=1L,  
    upperBound=100000L,  
    numPartitions=100,  
    connectionProperties=connectionProperties))
```



# (Re)partitioning

`df.repartition(100)` - repartition to a number of partitions

`df.repartition(100, col("some_column"))` - repartition and split on column

`df.coalesce(10)` - coalesce (narrow dependency)



# (Re)partitioning

Partition by a join field

Repartition is expensive but can help down the road

coalesce vs repartition

HashPartitioner

Number of partitions divisible by available cores

(usually from 2 to 10 x cores)



# Data locality

**PROCESS\_LOCAL** - data is in the same JVM

**NODE\_LOCAL** - data is on the same node (HDFS, local executor)

**RACK\_LOCAL, ANY** - data is not on the executors

**NO\_PREF** - no locality preference



# Data locality

## Tasks (6)

Index ▲	ID	Attempt	Status	Locality Level	Executor ID / Host
0	26688	0	SUCCESS	PROCESS_LOCAL	1 / 10.10.43.102
1	26689	0	SUCCESS	PROCESS_LOCAL	2 / 10.10.43.102
2	26690	0	SUCCESS	PROCESS_LOCAL	0 / 10.10.43.102
3	26691	0	SUCCESS	PROCESS_LOCAL	1 / 10.10.43.102
4	26692	0	SUCCESS	PROCESS_LOCAL	2 / 10.10.43.102
5	26693	0	SUCCESS	PROCESS_LOCAL	0 / 10.10.43.102



# Work distribution

## Summary

	RDD Blocks	Storage Memory	Disk Used	Cores	Active Tasks	Failed Tasks	Complete Tasks	Total Tasks	Task Time (GC Time)	Input	Shuffle Read
<b>Active(4)</b>	209	178.0 KB / 1465.2 MB	0.0 B	6	1	0	6030	6031	1.4 m (2.3 s)	0.0 B	0.0 B
<b>Dead(0)</b>	0	0.0 B / 0.0 B	0.0 B	0	0	0	0	0	0 ms (0 ms)	0.0 B	0.0 B
<b>Total(4)</b>	209	178.0 KB / 1465.2 MB	0.0 B	6	1	0	6030	6031	1.4 m (2.3 s)	0.0 B	0.0 B

## Executors

Executor ID	Address	Status	RDD Blocks	Storage Memory	Disk Used	Cores	Active Tasks	Failed Tasks	Complete Tasks	Total Tasks	Task Time (GC Time)	Input	Shuffle Read	Shuffle Write
2	10.10.43.102:63674	Active	52	44.3 KB / 366.3 MB	0.0 B	2	0	0	2010	2010	28.2 s (446 ms)	0.0 B	0.0 B	0.0 B
1	10.10.43.102:63676	Active	52	44.3 KB / 366.3 MB	0.0 B	2	0	0	2010	2010	28.6 s (928 ms)	0.0 B	0.0 B	0.0 B
0	10.10.43.102:63675	Active	52	44.3 KB / 366.3 MB	0.0 B	2	1	0	2010	2011	28.2 s (941 ms)	0.0 B	0.0 B	0.0 B
driver	10.10.43.102:63664	Active	53	45.1 KB / 366.3 MB	0.0 B	0	0	0	0	0	0 ms (0 ms)	0.0 B	0.0 B	0.0 B



# Work distribution

## Summary

	RDD Blocks	Storage Memory	Disk Used	Cores	Active Tasks	Failed Tasks	Complete Tasks	Total Tasks	Task Time (GC Time)
<b>Active(4)</b>	148	126.0 KB / 1465.2 MB	0.0 B	6	6	0	15720	15726	2.9 m (2.5 s)
<b>Dead(0)</b>	0	0.0 B / 0.0 B	0.0 B	0	0	0	0	0	0 ms (0 ms)
<b>Total(4)</b>	148	126.0 KB / 1465.2 MB	0.0 B	6	6	0	15720	15726	2.9 m (2.5 s)

## Executors

Executor ID	Address	Status	RDD Blocks	Storage Memory	Disk Used	Cores	Active Tasks	Failed Tasks	Complete Tasks	Total Tasks	Task Time (GC Time)	Ir
2	10.10.43.102:63674	Active	37	31.5 KB / 366.3 MB	0.0 B	2	2	0	5240	5242	57.5 s (503 ms)	0
1	10.10.43.102:63676	Active	37	31.5 KB / 366.3 MB	0.0 B	2	2	0	5240	5242	57.7 s (1.0 s)	0
0	10.10.43.102:63675	Active	37	31.5 KB / 366.3 MB	0.0 B	2	2	0	5240	5242	57.0 s (1.0 s)	0
driver	10.10.43.102:63664	Active	37	31.5 KB / 366.3 MB	0.0 B	0	0	0	0	0	0 ms (0 ms)	0





# Broadcasting variables

Keeps a copy of data on each worker

Reduces the size of a serialized task as data is on the worker already

Optimizes join operations



# Checkpointing

Truncate lineage graph

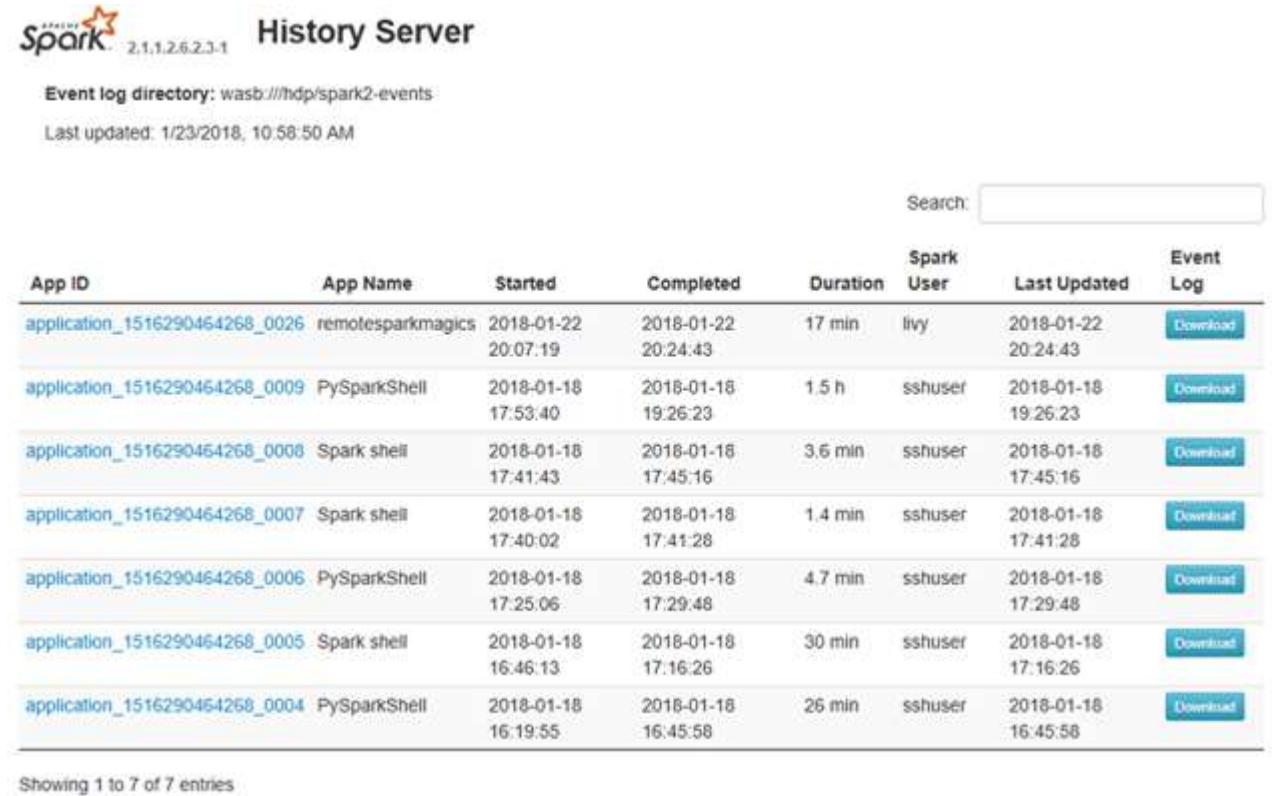
Reliable - HDFS storage (`sc.setCheckpointDir`)

Local - node local temp storage



# Metrics

Spark WebUI is your friend  
Use Spark History Server  
JSON metrics over REST API  
Collect JMX metrics (Dropwizard)



The screenshot shows the Spark History Server interface. At the top left is the Spark logo with version 2.1.1.2.6.2.3.1. The title is "History Server". Below the title, it says "Event log directory: wasb:///hdp/spark2-events" and "Last updated: 1/23/2018, 10:58:50 AM". There is a search bar on the right. The main content is a table with 8 columns: App ID, App Name, Started, Completed, Duration, Spark User, Last Updated, and Event Log. There are 7 rows of data, each with a "Download" button in the Event Log column. Below the table, it says "Showing 1 to 7 of 7 entries".

App ID	App Name	Started	Completed	Duration	Spark User	Last Updated	Event Log
<a href="#">application_1516290464268_0026</a>	remotesparkmagics	2018-01-22 20:07:19	2018-01-22 20:24:43	17 min	livy	2018-01-22 20:24:43	<a href="#">Download</a>
<a href="#">application_1516290464268_0009</a>	PySparkShell	2018-01-18 17:53:40	2018-01-18 19:26:23	1.5 h	sshuser	2018-01-18 19:26:23	<a href="#">Download</a>
<a href="#">application_1516290464268_0008</a>	Spark shell	2018-01-18 17:41:43	2018-01-18 17:45:16	3.6 min	sshuser	2018-01-18 17:45:16	<a href="#">Download</a>
<a href="#">application_1516290464268_0007</a>	Spark shell	2018-01-18 17:40:02	2018-01-18 17:41:28	1.4 min	sshuser	2018-01-18 17:41:28	<a href="#">Download</a>
<a href="#">application_1516290464268_0006</a>	PySparkShell	2018-01-18 17:25:06	2018-01-18 17:29:48	4.7 min	sshuser	2018-01-18 17:29:48	<a href="#">Download</a>
<a href="#">application_1516290464268_0005</a>	Spark shell	2018-01-18 16:46:13	2018-01-18 17:16:26	30 min	sshuser	2018-01-18 17:16:26	<a href="#">Download</a>
<a href="#">application_1516290464268_0004</a>	PySparkShell	2018-01-18 16:19:55	2018-01-18 16:45:58	26 min	sshuser	2018-01-18 16:45:58	<a href="#">Download</a>

Showing 1 to 7 of 7 entries



# Event timeline

## Details for Stage 10265 (Attempt 0)

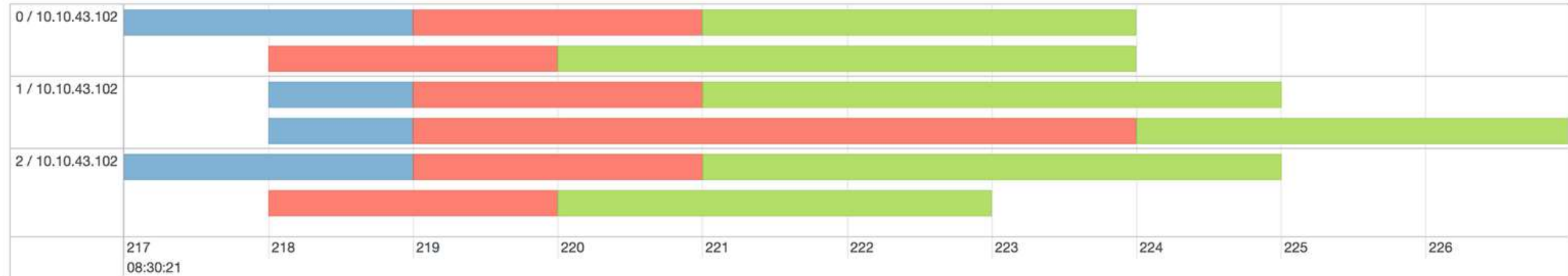
Total Time Across All Tasks: 21 ms

Locality Level Summary: Process local: 6

- ▶ DAG Visualization
- ▶ Show Additional Metrics
- ▼ Event Timeline

Enable zooming

- Scheduler Delay
- Task Deserialization Time
- Shuffle Read Time
- Executor Computing Time
- Shuffle Write Time
- Result Serialization Time
- Getting Result Time



# Storage tab

RDD Name ▲	Storage Level	Cached Partitions	Fraction Cached	Size in Memory	Size on Disk
MapPartitionsRDD	Memory Serialized 1x Replicated	4	100%	67.1 KB	0.0 B
MapPartitionsRDD	Memory Serialized 1x Replicated	4	100%	59.5 KB	0.0 B
MapPartitionsRDD	Memory Serialized 1x Replicated	4	100%	66.1 KB	0.0 B
MapPartitionsRDD	Memory Serialized 1x Replicated	4	100%	65.9 KB	0.0 B
MapPartitionsRDD	Memory Serialized 1x Replicated	4	100%	61.7 KB	0.0 B
MapPartitionsRDD	Memory Serialized 1x Replicated	4	100%	63.4 KB	0.0 B
MapPartitionsRDD	Memory Serialized 1x Replicated	4	100%	76.4 KB	0.0 B
MapPartitionsRDD	Memory Serialized 1x Replicated	4	100%	61.8 KB	0.0 B
MapPartitionsRDD	Memory Serialized 1x Replicated	4	100%	70.9 KB	0.0 B
MapPartitionsRDD	Memory Serialized 1x Replicated	4	100%	66.1 KB	0.0 B
MapPartitionsRDD	Memory Serialized 1x Replicated	4	100%	58.7 KB	0.0 B
MapPartitionsRDD	Memory Serialized 1x Replicated	4	100%	59.9 KB	0.0 B
MapPartitionsRDD	Memory Serialized 1x Replicated	4	100%	69.5 KB	0.0 B
MapPartitionsRDD	Memory Serialized 1x Replicated	4	100%	71.8 KB	0.0 B
MapPartitionsRDD	Memory Serialized 1x Replicated	4	100%	55.5 KB	0.0 B
MapPartitionsRDD	Memory Serialized 1x Replicated	4	100%	65.3 KB	0.0 B
MapPartitionsRDD	Memory Serialized 1x Replicated	4	100%	61.6 KB	0.0 B
MapPartitionsRDD	Memory Serialized 1x Replicated	4	100%	62.3 KB	0.0 B



# RDD storage details

**Storage Level:** Memory Deserialized 1x Replicated

**Cached Partitions:** 2

**Total Partitions:** 2

**Memory Size:** 11.2 KB

**Disk Size:** 0.0 B

## Data Distribution on 4 Executors

Host	Memory Usage
192.168.19.244:65021	5.5 KB (366.3 MB Remaining)
192.168.19.244:65009	0.0 B (366.3 MB Remaining)
192.168.19.244:65020	0.0 B (366.3 MB Remaining)
192.168.19.244:65018	5.7 KB (366.3 MB Remaining)

## 2 Partitions

Block Name ▲	Storage Level	Size in Memory
rdd_1_0	Memory Deserialized 1x Replicated	5.5 KB
rdd_1_1	Memory Deserialized 1x Replicated	5.7 KB



But what if all fails?

# What is Apache Spark?

**Apache Spark™** is a unified analytics engine for large-scale data processing.

Apache Spark

Computer software



Apache Spark is an open-source cluster-computing framework. Originally developed at the University of California, Berkeley's AMPLab, the Spark codebase was later donated to the Apache Software Foundation, which has maintained it since. [Wikipedia](#)





# Let's use it

Spark is a distributed computing framework

Spark provides resource management

Spark provides controllable work distribution

Spark is highly configurable

Why not?



# MapPartitions function

```
/**  
 * Return a new RDD by applying a function to each partition of this RDD.  
 *  
 * `preservesPartitioning` indicates whether the input function preserves the partitioner, which  
 * should be `false` unless this is a pair RDD and the input function doesn't modify the keys.  
 */  
def mapPartitions[U: ClassTag](  
    f: Iterator[T] => Iterator[U],  
    preservesPartitioning: Boolean = false): RDD[U]
```



# MapPartitions function - indexed

```
/**  
 * Return a new RDD by applying a function to each partition of this RDD, while tracking the index  
 * of the original partition.  
 *  
 * `preservesPartitioning` indicates whether the input function preserves the partitioner, which  
 * should be `false` unless this is a pair RDD and the input function doesn't modify the keys.  
 */  
def mapPartitionsWithIndex[U: ClassTag](  
  f: (Int, Iterator[T]) => Iterator[U],  
  preservesPartitioning: Boolean = false): RDD[U]
```



# MapPartitions example

```
def mapFunction(args): Iterator[Row] => Iterator[Row] = iterator => {  
  // Processing based on the partition data  
  // Example:  
  //   val ids = iterator.toList.map(row => row.getInt(0))  
  //   Processor.doWork(ids)  
  iterator  
}
```



# Other usages

Applying custom processing code

Running ML pipeline

Using third party libraries

Misc



How about them numbers?

# Results

## PROs

Processing time dropped from ~96 hours to ~3.5hours (28 times)

Adding new entities doesn't significantly impact complexity/performance

Additional control which we leveraged down the road

We managed to use all the resources

## CONs

We managed to use all the resources

Raised code complexity



# Q&A



# Thank you

Matija Gobec  
matija.gobec@smartcat.io  
@mad\_max0204

SmartCat.io