Orchestrating work with `mapPartitions`

What it does

- `mapPartitions(func)`
  - `func` runs once per partition
  - Receives an iterator over all elements in that partition
  - Must return or yield zero or more outputs

```
rdd2 = rdd.mapPartitions(func)
```

- `foreachPartition(func)`
  - Same execution model
  - **For side effects only** (writes, logging)
  - `func` returns nothing

Partition preparation

```
rdd = sc.parallelize(tasks, numSlices=num_partitions)
```

- Spark splits `tasks` into `num_partitions` slices
- Each slice → **one partition**
- Each partition → **a series of tasks**
- `mapPartitions(func)` is a **narrow transformation**
  - No shuffle
  - DAGScheduler creates `num_partitions` tasks
  - Each executor runs `func` **once per partition**

Common pitfall: nested partitions

```
sc.parallelize([partition1, partition2], numSlices=2)
```

What actually happens

- Partition 0 → `[partition1]`
- Partition 1 → `[partition2]`

Inside `mapPartitions`:

- Iterator yields **one element**
- That element is itself a list

Result:

```
Expected:   [task1, task2, ...]
Got:        [[task1, task2, ...]]
```

`yield` → streaming rows into a DataFrame

Generator mechanics (Python)

- A function containing `yield` becomes a generator
- Execution:
    1. Runs until first `yield`
    2. Emits value
    3. Suspends
    4. Resumes on next `next()`
- When code ends → `StopIteration`

Typical `mapPartitions + yield` pattern

Input

```
tasks = [a1, a2, b1, b2]
P0 = [a1, a2]
P1 = [b1, b2]
```

Function

```
def f(records_iter):
    for r in records_iter:
        yield Row(...)
```

Pipeline

```
rdd = sc.parallelize(task, 2).mapPartitions(f)
df = spark.createDataFrame(rdd)
```

- One `yield` → one output row
- No need to build lists
- Rows stream immediately back to Spark

Bonus - `yield from` (recursive generators)

Replaced

```
for child in children:
    for r in crawler(child):
        yield r
```

With

```
for child in children:
    yield from crawler(child)
```

Meaning: Forward all yielded values from the inner generator