

Key Scheduler Operators for Project Reactor Reactive Types (Part 1)

Douglas C. Schmidt

d.schmidt@vanderbilt.edu

www.dre.vanderbilt.edu/~schmidt

Professor of Computer Science

**Institute for Software
Integrated Systems**

**Vanderbilt University
Nashville, Tennessee, USA**



Learning Objectives in this Part of the Lesson

- Recognize key Flux operators
 - Concurrency operators
 - Scheduler operators
 - These operators provide various types of threads & thread pools
 - e.g., `Schedulers.newParallel()`



Key Scheduler Operators for Project Reactor Reactive Types

Key Scheduler Operators for Project Reactor Reactive Types

- The `Schedulers.newParallel()` operator
 - Hosts a fixed-sized pool of single-threaded `ExecutorService`-based workers

```
static Scheduler newParallel  
(String name,  
 int parallelism)
```

Key Scheduler Operators for Project Reactor Reactive Types

- The `Schedulers.newParallel()` operator
 - Hosts a fixed-sized pool of single-threaded `ExecutorService`-based workers
 - The params (1) give a name for the scheduler & (2) indicate the # of pooled worker threads

```
static Scheduler newParallel  
    (String name,  
     int parallelism)
```

Key Scheduler Operators for Project Reactor Reactive Types

- The `Schedulers.newParallel()` operator
 - Hosts a fixed-sized pool of single-threaded `ExecutorService`-based workers
 - The params (1) give a name for the scheduler & (2) indicate the # of pooled worker threads
 - Returns a `Scheduler` suitable for parallel compute-bound operations

```
static Scheduler newParallel  
(String name,  
 int parallelism)
```



Key Scheduler Operators for Project Reactor Reactive Types

- The `Schedulers.newParallel()` operator
 - Hosts a fixed-sized pool of single-threaded `ExecutorService`-based workers
 - The params (1) give a name for the scheduler & (2) indicate the # of pooled worker threads
 - Returns a Scheduler suitable for parallel compute-bound operations
 - However, it detects & rejects use of blocking Reactor APIs



Class Schedulers

`java.lang.Object`

`reactor.core.scheduler.Schedulers`

```
public abstract class Schedulers
extends Object
```

`Schedulers` provides various `Scheduler` flavors usable by `publishOn` or `subscribeOn`:

- `parallel()`: Optimized for fast `Runnable` non-blocking executions
- `single()`: Optimized for low-latency `Runnable` one-off executions
- `elastic()`: Optimized for longer executions, an alternative for blocking tasks where the number of active tasks (and threads) can grow indefinitely
- `boundedElastic()`: Optimized for longer executions, an alternative for blocking tasks where the number of active tasks (and threads) is capped
- `immediate()`: to immediately run submitted `Runnable` instead of scheduling them (somewhat of a no-op or "null object" `Scheduler`)
- `fromExecutorService(ExecutorService)` to create new instances around `Executors`

See projectreactor.io/docs/core/release/api/reactor/core/scheduler/Schedulers.html

Key Scheduler Operators for Project Reactor Reactive Types

- The `Schedulers.newParallel()` operator
 - Hosts a fixed-sized pool of single-threaded `ExecutorService`-based workers
 - Can be used to create a custom parallel scheduler

Arrange to emit the random big integers in the "publisher" thread

```
Scheduler publisher = Schedulers
    .newParallel("publisher", 1);
Flux
    .range(1, sMAX_ITERATIONS)
    .map(Integer::toUnsignedLong)
    .subscribeOn(publisher)
    .map(sGenerateRandomBigInt)
    .filter(sOnlyOdd)
    .doFinally(() -> publisher
        .dispose())
    .subscribe(sink::next,
        error ->
            sink.complete(),
        sink::complete);
```

See [Reactive/flux/ex2/src/main/java/FluxEx.java](#)

Key Scheduler Operators for Project Reactor Reactive Types

- The `Schedulers.newParallel()` operator
 - Hosts a fixed-sized pool of single-threaded `ExecutorService`-based workers
 - Can be used to create a custom parallel scheduler
 - *Not* implemented via a “daemon thread”



Key Scheduler Operators for Project Reactor Reactive Types

- The `Schedulers.newParallel()` operator
 - Hosts a fixed-sized pool of single-threaded `ExecutorService`-based workers
 - Can be used to create a custom parallel scheduler
 - *Not* implemented via a “daemon thread”
 - i.e., an app won’t exit until this pool is disposed of properly & explicitly

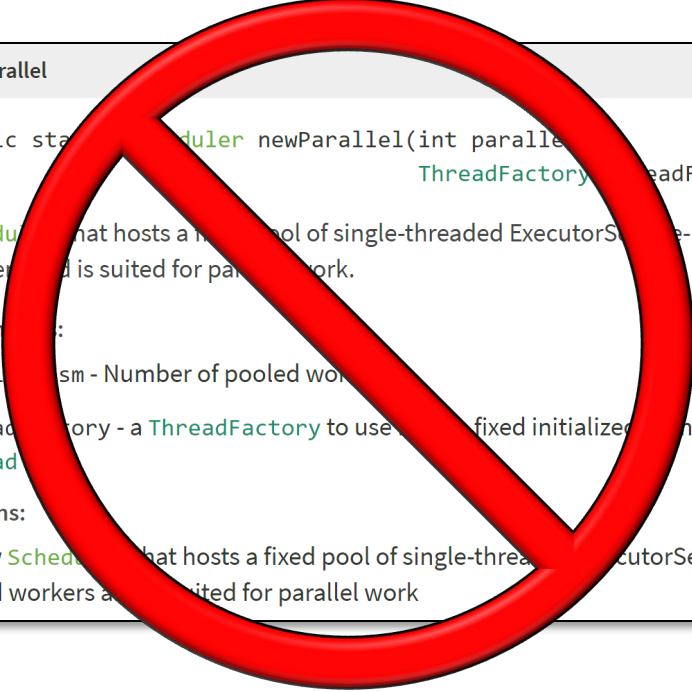
```
Scheduler publisher = Schedulers  
    .newParallel("publisher", 1);
```

```
Flux
```

```
.range(1, sMAX_ITERATIONS)  
.map(Integer::toUnsignedLong)  
.subscribeOn(publisher)  
.map(sGenerateRandomBigInt)  
.filter(sOnlyOdd)  
.doFinally(() -> publisher  
    .dispose())  
.subscribe(sink::next,  
    error ->  
        sink.complete(),  
    sink::complete);
```

Key Scheduler Operators for Project Reactor Reactive Types

- The `Schedulers.newParallel()` operator
 - Hosts a fixed-sized pool of single-threaded `ExecutorService`-based workers
 - Can be used to create a custom parallel scheduler
 - RxJava's `Schedulers` doesn't have an equivalent method



```
newParallel

public static Scheduler newParallel(int parallelism, ThreadFactory threadFactory)

Scheduler that hosts a fixed pool of single-threaded ExecutorService-based
workers and is suited for parallel work.

Parameters:
parallelism - Number of pooled workers
threadFactory - a ThreadFactory to use with a fixed initialized number of
Thread

Returns:
a new Scheduler that hosts a fixed pool of single-threaded ExecutorService-
based workers and is suited for parallel work
```

Key Scheduler Operators for Project Reactor Reactive Types

- The `Schedulers.newParallel()` operator
 - Hosts a fixed-sized pool of single-threaded `ExecutorService`-based workers
 - Can be used to create a custom parallel scheduler
- RxJava's Schedulers doesn't have an equivalent method
 - However, its `from()` method can be used in conjunction with Java's `Executor` framework

from

```
@NonNull  
public static @NonNull Scheduler from(@NonNull  
                                     @NonNull Executor executor)
```

Wraps an `Executor` into a new `Scheduler` instance and delegates `schedule()` calls to it.

If the provided executor doesn't support any of the more specific standard Java executor APIs, cancelling tasks scheduled by this scheduler can't be interrupted when they are executing but only prevented from running prior to that. In addition, tasks scheduled with a time delay or periodically will use the `single()` scheduler for the timed waiting before posting the actual task to the given executor.

Tasks submitted to the `Scheduler.Worker` of this `Scheduler` are also not interruptible. Use the `from(Executor, boolean)` overload to enable task interruption via this wrapper.

If the provided executor supports the standard Java `ExecutorService` API, cancelling tasks scheduled by this scheduler can be cancelled/interrupted by calling `Disposable.dispose()`. In addition, tasks scheduled with a time delay or periodically will use the `single()` scheduler for the timed waiting before posting the actual task to the given executor.

If the provided executor supports the standard Java `ScheduledExecutorService` API, cancelling tasks scheduled by this scheduler can be cancelled/interrupted by calling `Disposable.dispose()`. In addition, tasks scheduled with a time delay or periodically will use the provided executor. Note, however, if the provided `ScheduledExecutorService` instance is not single threaded, tasks scheduled with a time delay close to each other may end up executing in different order than the original `schedule()` call was issued. This limitation may be lifted in a future patch.

Key Scheduler Operators for Project Reactor Reactive Types

- The `Schedulers.newParallel()` operator
 - Hosts a fixed-sized pool of single-threaded `ExecutorService`-based workers
 - Can be used to create a custom parallel scheduler
- RxJava's `Schedulers` doesn't have an equivalent method
 - However, its `from()` method can be used in conjunction with Java's `Executor` framework, e.g.

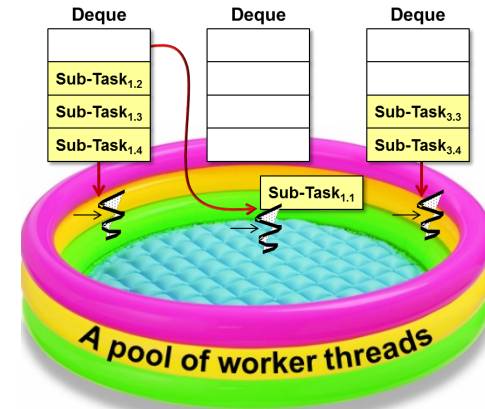
*Cached (Variable-sized)
Thread Pool*



*Fixed-sized
Thread Pool*

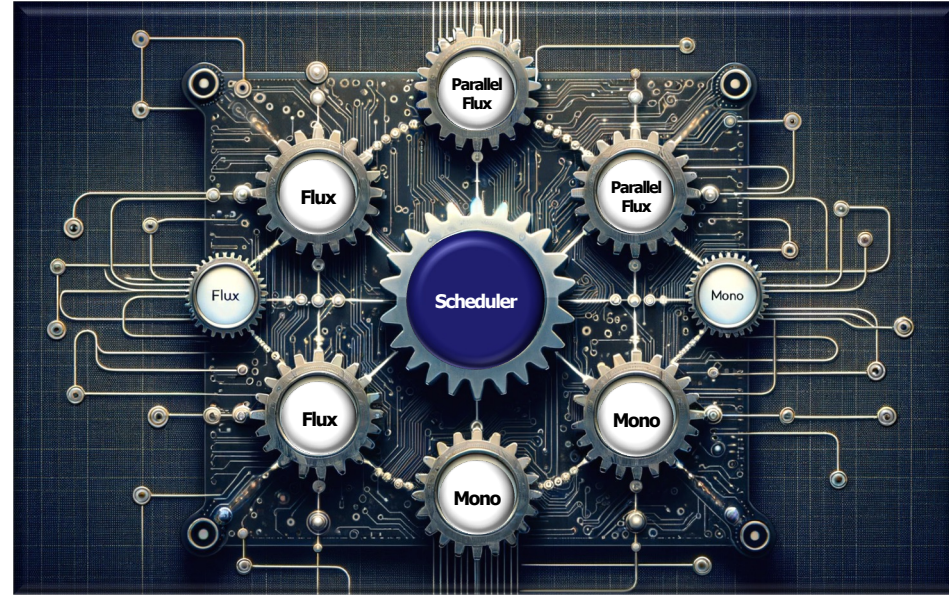


*Work-stealing
Thread Pool*



Key Scheduler Operators for Project Reactor Reactive Types

- The `Schedulers.newParallel()` operator
 - Hosts a fixed-sized pool of single-threaded `ExecutorService`-based workers
 - Can be used to create a custom parallel scheduler
 - RxJava's `Schedulers` doesn't have an equivalent method
- Project Reactor decouples Scheduler params from `Flux`, `ParallelFlux`, & `Mono` reactive types to enhance reuse



REDUCE
USE
CYCLE

End of Key Scheduler Operators for Project Reactor Reactive Types (Part 1)