Jean Carlos Lerma Rojas 2259305
Juan Camilo Garcia Saenz 2259416
Jhojan Serna Henao 2259504

Informe Proyecto Final PFC 2024-I

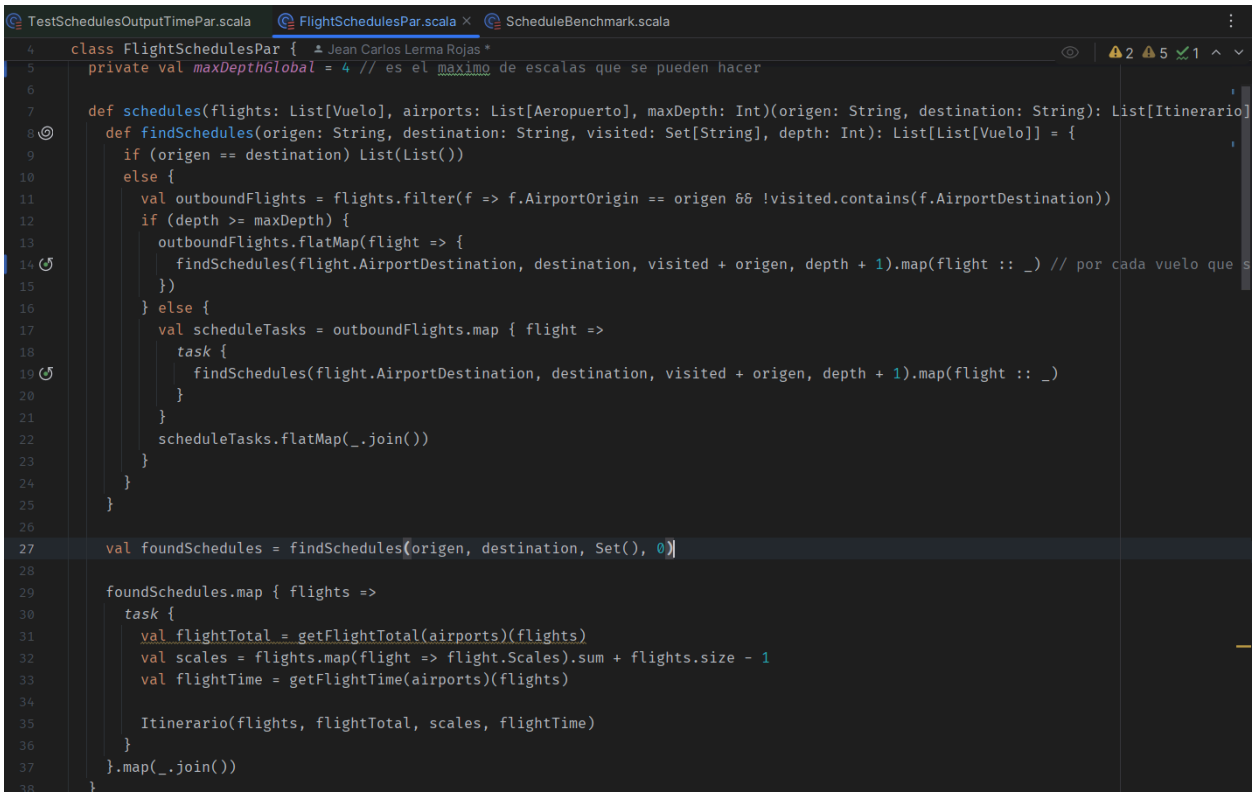
Solución al problema:
Para encontrar los mejores itinerarios de vuelo desde un aeropuerto de origen a un aeropuerto de destino, considerando múltiples criterios de optimización:

1. **Tiempo total de viaje**: Incluye el tiempo de vuelo y el tiempo de espera entre conexiones.
2. **Número de escalas**: Minimizar el número de conexiones entre vuelos.
3. **Tiempo de vuelo**: Solo el tiempo en el aire, excluyendo el tiempo de espera en las escalas.
4. **Horario de llegada**: Asegurar que el itinerario llegue antes de una hora específica.

La solución se implementa en la clase FlightSchedulesPar, que utiliza técnicas de paralelización para mejorar la eficiencia en la búsqueda de itinerarios de vuelo.

La solución planteada se basa en la paralelización de tareas y en aplicar límites de profundidad como criterio para hallar cuando es beneficioso realizar la paralelización y cuando la secuencial. Es decir, si la profundidad en el método de schedule es menor que maxDepth, entonces la búsqueda se realiza de manera paralela. En el caso paralelo, para cada vuelo que sale del aeropuerto de origen, se crea una tarea paralela que busca itinerarios desde el aeropuerto de destino de ese vuelo hasta el destino final. Estas tareas se crean utilizando la función task y se ejecutan en paralelo. Los resultados de estas tareas se combinan utilizando la función flatMap y la función join. Además, la creación de los itinerarios a partir de las listas de vuelos encontradas también se realiza en paralelo. Para cada lista de vuelos, se crea una tarea que calcula el tiempo total de vuelo, el número de escalas y el tiempo de vuelo, y crea un objeto Itinerario. Estas tareas se ejecutan en paralelo y los resultados se combinan utilizando la función map y la función join. Por lo tanto, la paralelización en FlightSchedulesPar se realiza a nivel de búsqueda de itinerarios y a nivel de creación de itinerarios.

Jean Carlos Lerma Rojas 2259305
Juan Camilo Garcia Saenz 2259416
Jhojan Serna Henao 2259504



Comparativas:

```
benchmark.run(benchmark.benchmarkSchedulesSeq)(benchmark.benchmarkSchedulesPar)
```

Tamaño de los datos de Vuelos	Solución Secuencial (ms)	Solución Paralela (ms)
15	0.0184	0.0102
40	0.0223	0.0105
100	0.0625	0.0525
500	0.0926	0.0483

```
benchmark.run(benchmark.benchmarkSchedulesTimeSeq)(benchmark.benchmarkSchedulesTimePar)
```

Tamaño de los datos de Vuelos	Solución Secuencial (ms)	Solución Paralela (ms)
15	0.0456	0.0224

Jean Carlos Lerma Rojas 2259305
Juan Camilo Garcia Saenz 2259416
Jhojan Serna Henao 2259504

40	0.0496	0.0261
100	0.0766	0.0248
500	0.0773	0.0464

```
benchmark.run(benchmark.benchmarkSchedulesScalesSeq)(benchmark.benchmarkSchedulesScalesPar)
```

Tamaño de los datos de Vuelos	Solución Secuencial (ms)	Solución Paralela (ms)
15	0.0399	0.0143
40	0.0671	0.0277
100	0.0706	0.0548
500	0.0912	0.0537

```
benchmark.run(benchmark.benchmarkSchedulesTimeLandSeq)(benchmark.benchmarkSchedulesTimeLandPar)
```

Tamaño de los datos de Vuelos	Solución Secuencial (ms)	Solución Paralela (ms)
15	0.0363	0.0303
40	0.0402	0.0177
100	0.0497	0.0211
500	0.0773	0.0653

```
benchmark.run(benchmark.benchmarkSchedulesOutputTimeSeq)(benchmark.benchmarkSchedulesOutputTimePar)
```

Tamaño de los datos de Vuelos	Solución Secuencial (ms)	Solución Paralela (ms)
15	0.0422	0.0272
40	0.1013	0.1149
100	0.0515	0.0289

Jean Carlos Lerma Rojas 2259305
Juan Camilo Garcia Saenz 2259416
Jhojan Serna Henao 2259504

500	0.0702	0.0735

Conclusiones:

Al hacer las comparaciones anteriores podemos observar que la versión paralela tuvo una mejor eficiencia que la versión secuencial, podemos decir que la implementación de la versión paralela fue la más apropiada ya que para tamaños de datos muy grandes la eficiencia del paralelo cada vez se vuelve más notoria. Incluso con datos pequeños la versión paralela tuvo un rendimiento mejor que el secuencial.

Al observar las comparaciones para datos pequeños nos damos cuenta de que el costo computacional del paralelo fue mínimo ya que los tiempos de ejecución de las dos versiones fueron muy similares. o la versión paralela fue más rápida

Podemos concluir que la versión paralela tuvo un mejor rendimiento en comparación con la secuencial siendo esta versión la más eficiente y se recomienda la paralela sobre todo para tamaños de datos muy grandes, aunque como consideración en ciertas casos o funciones los resultados fueron negativos para la versión paralela y esto pudo deberse a la complejidad de sus operaciones internas o factores externos del sistema.