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

Informe Proyecto Final PFC 2024-I

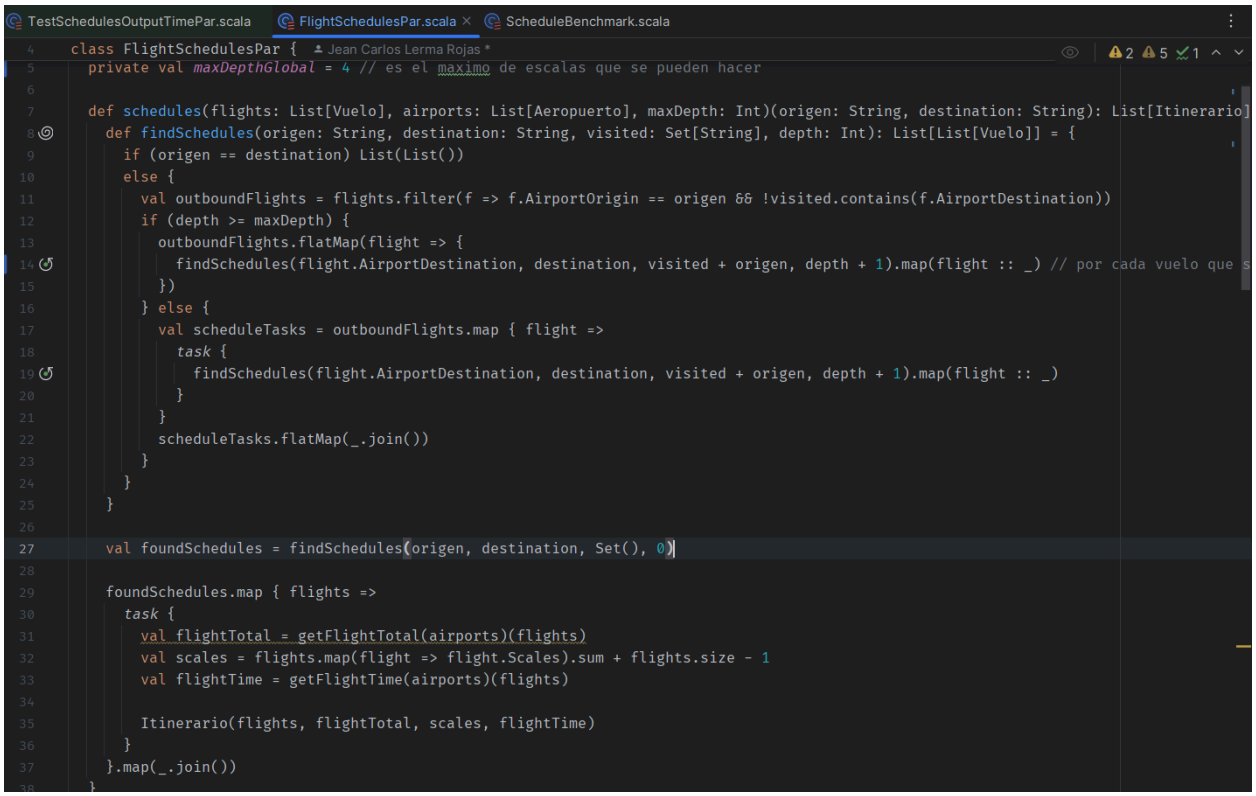
Solución al problema:
Para encontrar los mejores itinerarios de vuelo desde un aeropuerto de origen a un aeropuerto de destino, considerando múltiples criterios de optimización:

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.

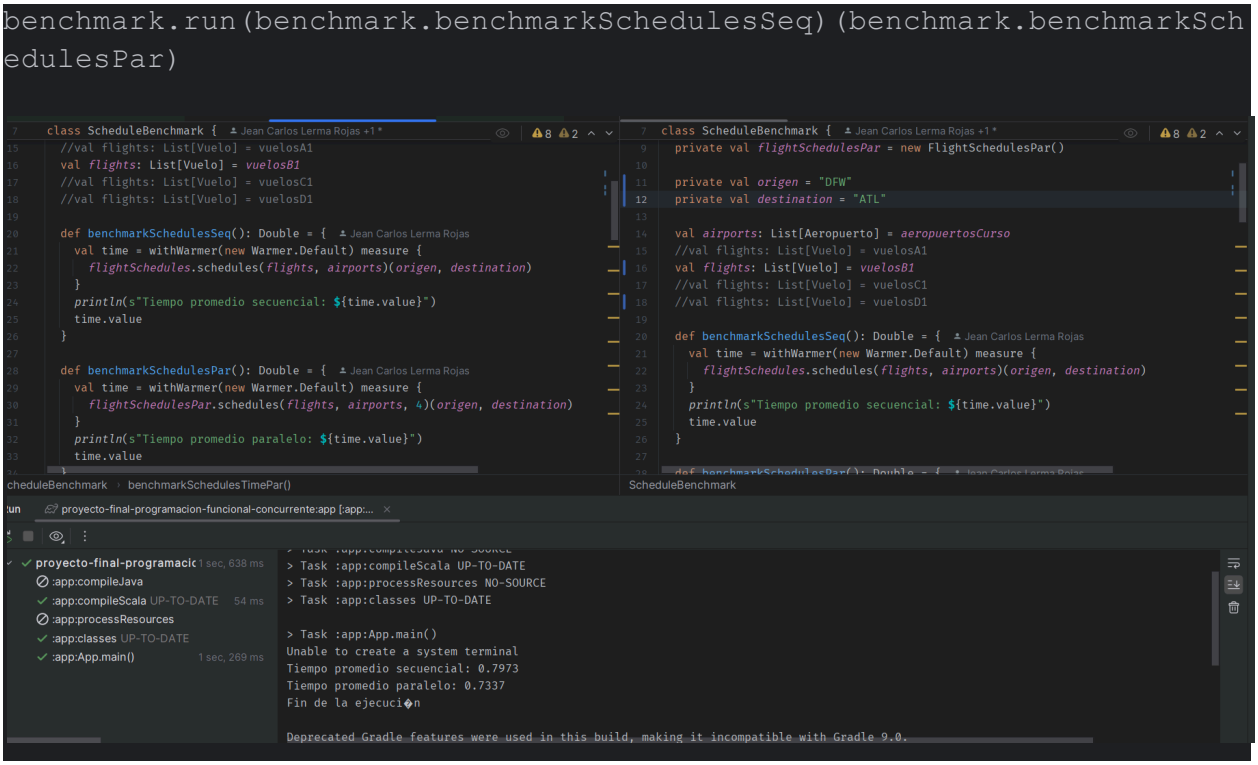
La solución planteada se basa en la paralelización de tareas y en aplicar límites de profundidad como criterio para hallar cuando es beneficioso realizar la paralelización y cuando la secuencial. Es decir, si la profundidad en el método de schedule es menor que maxDepth, entonces la búsqueda se realiza 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:

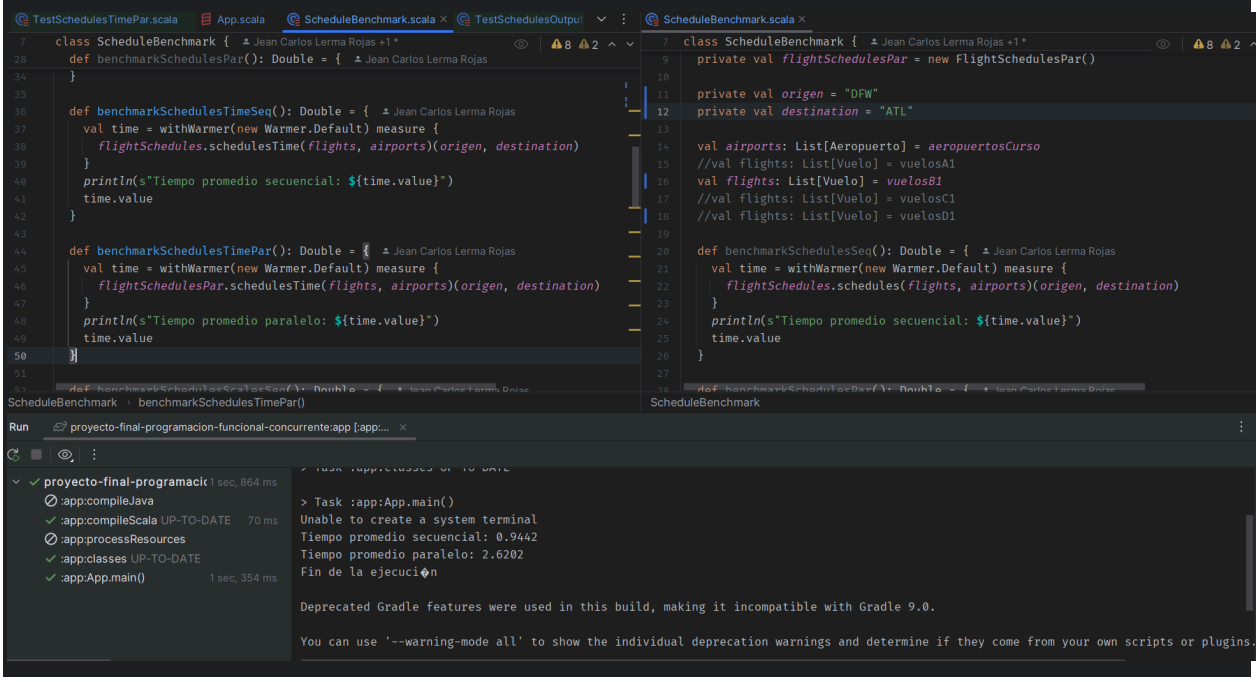
```
benchmark.run(benchmark.benchmarkSchedulesSeq) (benchmark.benchmarkSch
edulesPar)
```



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

Tamaño de los datos de Vuelos	Solución Secuencial (ms)	Solución Paralela (ms)
15	0.2685	0.2323
40	0.0383	0.0203
100	49.5613	19.4102

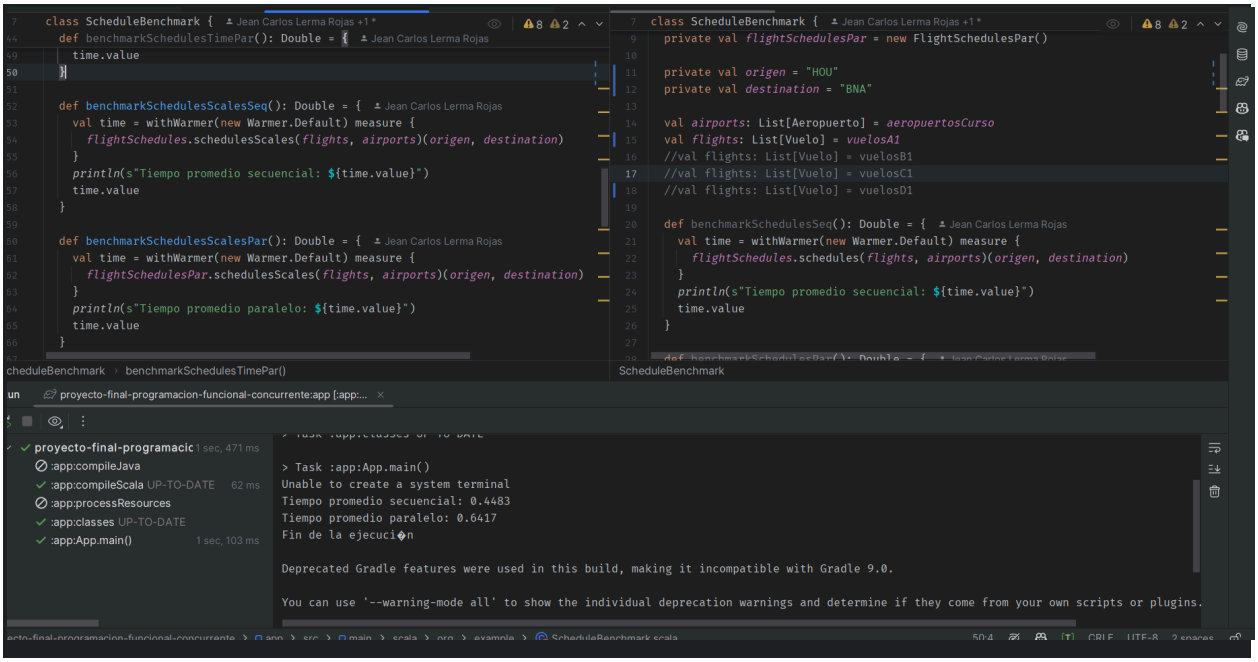
```
benchmark.run(benchmark.benchmarkSchedulesTimeSeq) (benchmark.benchmarkSchedulesTimePar)
```



Tamaño de los datos de Vuelos	Solución Secuencial (ms)	Solución Paralela (ms)
15	0.4559	0.2151
40	0.9537	1.0748
100	44.5055	29.827

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

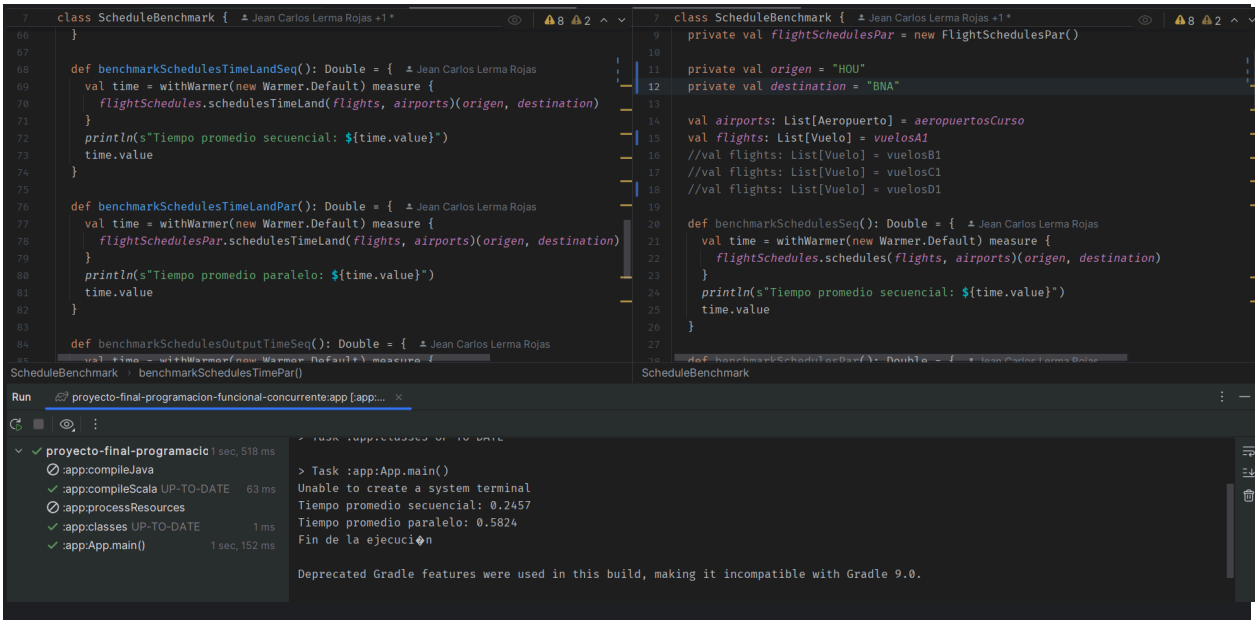
Jean Carlos Lerma Rojas 2259305
Juan Camilo Garcia Saenz 2259416
Jhojan Serna Henao 2259504



Tamaño de los datos de Vuelos	Solución Secuencial (ms)	Solución Paralela (ms)
15	0.4483	0.6417
40	1.4903	1.1823
100	19.9988	9.6963

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

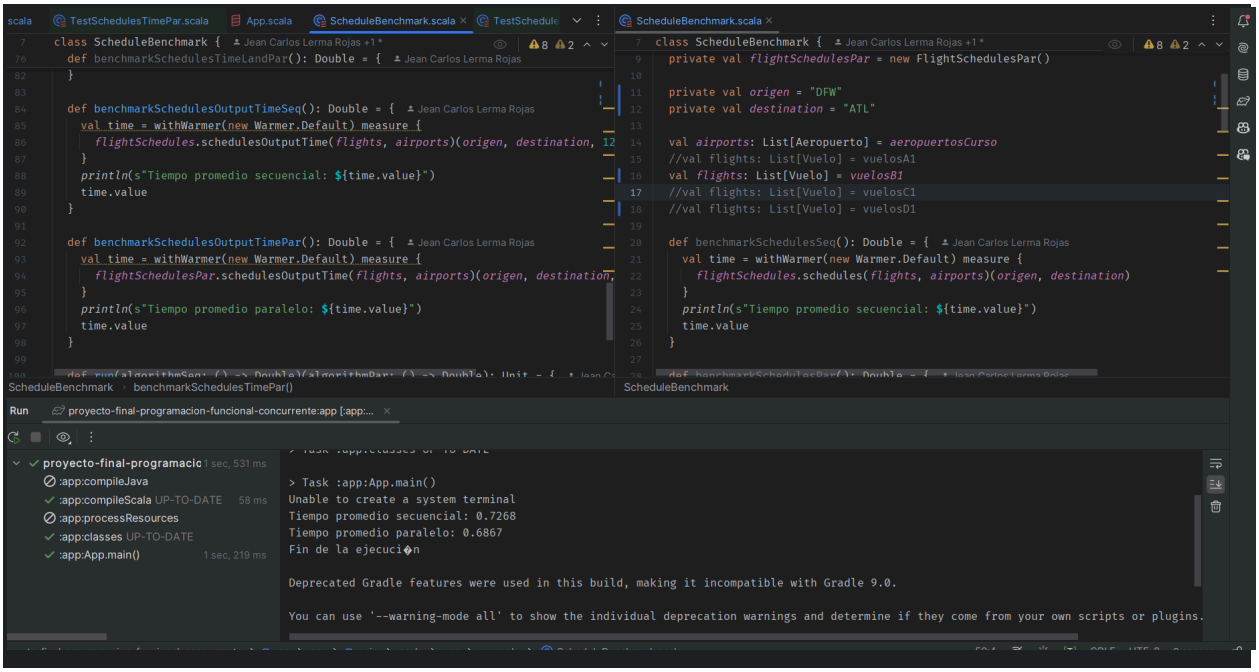
Jean Carlos Lerma Rojas 2259305
Juan Camilo Garcia Saenz 2259416
Jhojan Serna Henao 2259504



Tamaño de los datos de Vuelos	Solución Secuencial (ms)	Solución Paralela (ms)
15	0.2457	0.5824
40	0.043	0.0467
100	56.3106	28.5022

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

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



Tamaño de los datos de Vuelos	Solución Secuencial (ms)	Solución Paralela (ms)
15	0.4843	0.4615
40	0.4029	0.526
100	22.0562	14.0076

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 similar o mejor en comparación con la versión 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.