Real-time Scheduling Simulator

Edoardo Sarri

Software Engineering for Embedded System
Project Work

Giugno 2025

Introduzione

Obiettivo

Simulare l'esecuzione di un taskset secondo un dato algortimo di scheduling e un protocollo di accesso alle risorse.

Introduzione

Output

Un file di log contenente la traccia di esecuzione, cioè una sequenza di coppie $< tempo, evento >$, dove i possibili eventi sono:

- Rilascio di un task.
- Acuisizione e rilascio di una risorsa da parte di un chunk.
- Completamento dell'esecuzione di un chunk o di un job di un task.
- Preemption su un task.
- Deadline miss di un job di un task.
- Fault di un chunk.

Introduzione

Capacità

- Generare una traccia di esecuzione.
- Genenare un dataset di tracce di esecuzione.
- Simulare Rate Monotonic con e senza risorse convidese insieme a Priority Ceiling Protocol.
- Simulare Earliest Deadline First senza risorse condivise.
- Rilevare eventuali deadline miss.
- Controllare la feasibility di un taskset dato il relativo algortimo di scheduling.

Introduzione

Capacità (fault injection)

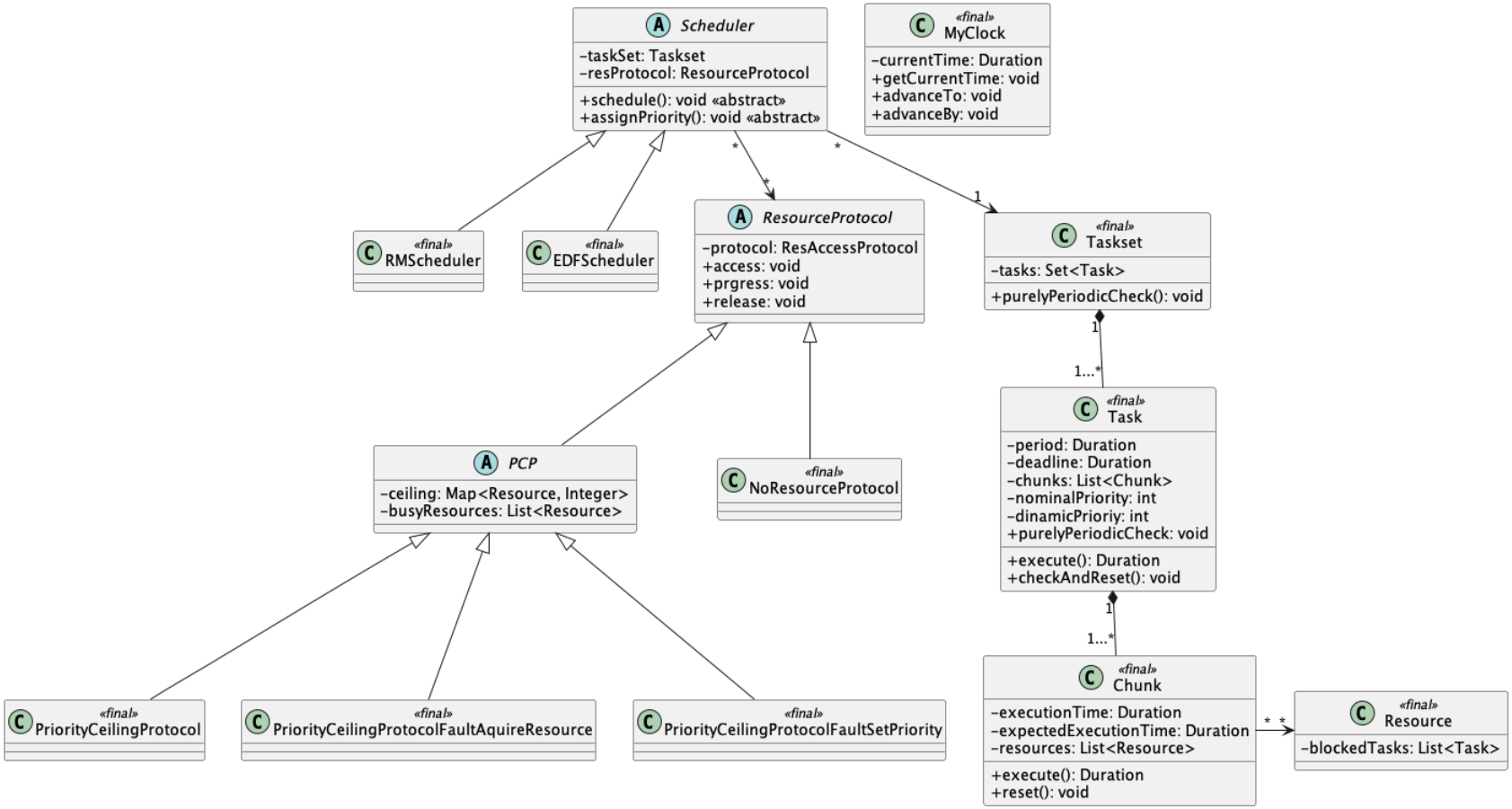
- Introdurre in modo stocastico e rilevare un additional execution time in un chunk.
- Introdurre un fault a livello del protocollo di accesso alle risorse per cui PCP imposta male la priorità dinamica dei task.
- Introdurre un fault a livello di chunk per cui esso non acquisisce (e rilascia) il semaforo della risorsa che userà.

Introduzione

Utilizzo

- All'interno del main devono essere definiti i componenti necessari: `Resource`, `Chunk`, `Task`, `TaskSet`, `Scheduler` e `ResourceProtocol`.
- Per avviare una simulazione chiamare il metodo `schedule` o `scheduleDataset` di uno `Scheduler`.
- I tempi devono essere passati e letti dal sistema in millisecondi. Il sistema li elabora in nanosecondi per una maggiore precisione.

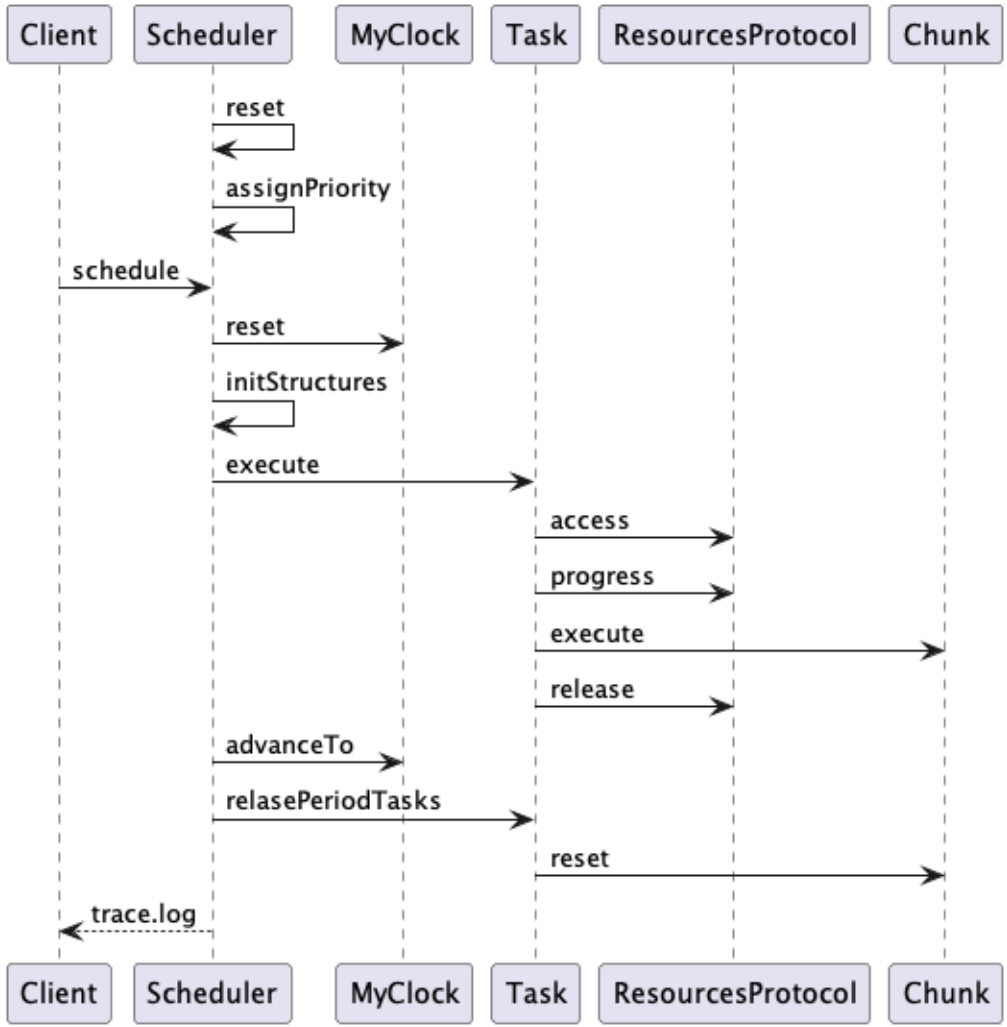
Analisi


Scheduler
-taskSet: Taskset
-resProtocol: ResourceProtocol
+schedule(): void «abstract»
+assignPriority(): void «abstract»
«final» MyClock
-currentTime: Duration
+getCurrentTime: void
+advanceTo: void
+advanceBy: void
«final» RMScheduler
«final» EDFScheduler
ResourceProtocol
-protocol: ResAccessProtocol
+access: void
+prgress: void
+release: void
«final» Taskset
-tasks: Set<Task>
+purelyPeriodicCheck(): void
«final» Task
-period: Duration
-deadline: Duration
-chunks: List<Chunk>
-nominalPriority: int
-dinamicPrioriy: int
+purelyPeriodicCheck: void
+execute(): Duration
+checkAndReset(): void
PCP
-ceiling: Map<Resource, Integer>
-busyResources: List<Resource>
«final» NoResourceProtocol
«final» Chunk
-executionTime: Duration
-expectedExecutionTime: Duration
-resources: List<Resource>
+execute(): Duration
+reset(): void
«final» Resource
-blockedTasks: List<Task>
«final» PriorityCeilingProtocol
«final» PriorityCeilingProtocolFaultAquireResource
«final» PriorityCeilingProtocolFaultSetPriority

Implementazione

Scheduler

- Gestione deterministica del tempo.
- Classe base che definisce la logica dello scheduling: *Template Method*.
- Classi concrete che implementano `assignPriority` e `addReadyTask`.



Implementazione

Resource Access Protocol

- Necessario `NoResourceProtocol` quando non si hanno risorse condivise.
- Implementa i metodi di accesso, progresso e rilascio.
- PCP usa due strutture dati: `ceiling` e `besyResource`.

Implementazione

Clock

- Gestione globale.
- Acceesso unificato tramite Singleton.
- Oggetti di tipo `Duration` di `java.time`.

Sampling

- Libreria `Sirio`.
- Aggiunto `ConstantSampler` per il campionamento di di un tempo costante.

Dataset

Earliest Deadline First (EDF)

- 5 task: 2 puramente periodici e 3 periodici.
- Fattore di utilizzo di 0,95.
- Chunk che campionano un additional ET da una uniforme.
- traceEDF.log

```
Task task1 = new Task(
    40,
    30,
    List.of(
        new Chunk(
            1,
            new ConstantSampler(new BigDecimal(4)),
            new UniformSampler(new BigDecimal(-1), new BigDecimal(2))),
        new Chunk(
            2,
            new ConstantSampler(new BigDecimal(4)),
            new UniformSampler(new BigDecimal(-1), new BigDecimal(1))),
        new Chunk(
            3,
            new ConstantSampler(new BigDecimal(4)),
            new UniformSampler(new BigDecimal(-1), new BigDecimal(1)))));
Task task2 = new Task(
    50,
    40,
    List.of(
        new Chunk(
            1,
            new ConstantSampler(new BigDecimal(4)),
            new UniformSampler(new BigDecimal(-1), new BigDecimal(1))),
        new Chunk(
            2,
            new ConstantSampler(new BigDecimal(5)),
            new UniformSampler(new BigDecimal(-1), new BigDecimal(1)))));
Task task3 = new Task(
    80,
    80,
    List.of(
        new Chunk(
            1,
            new ConstantSampler(new BigDecimal(7)),
            new UniformSampler(new BigDecimal(-1), new BigDecimal(2)))));
Task task4 = new Task(
    100,
    100,
    List.of(
        new Chunk(
            1,
            new ConstantSampler(new BigDecimal(6)),
            new UniformSampler(new BigDecimal(-1), new BigDecimal(3))),
        new Chunk(
            2,
            new ConstantSampler(new BigDecimal(6)),
            new UniformSampler(new BigDecimal(-1), new BigDecimal(3)))));
Task task5 = new Task(
    60,
    50,
    List.of(
        new Chunk(
            1,
            new ConstantSampler(new BigDecimal(5)),
            new UniformSampler(new BigDecimal(-1), new BigDecimal(1))),
        new Chunk(
            2,
            new ConstantSampler(new BigDecimal(5)),
            new UniformSampler(new BigDecimal(-1), new BigDecimal(2))),
        new Chunk(
            3,
            new ConstantSampler(new BigDecimal(5)),
            new UniformSampler(new BigDecimal(-1), new BigDecimal(1)))));
TaskSet taskSet = new TaskSet(Set.of(task1, task2, task3, task4, task5));
Scheduler edf = new EDFScheduler(taskSet, 500);
edf.scheduleDataset(10);
```

Dataset

Rate Monotonic (RM)

- 5 task puramente periodici.
- 2 risorse condivise.
- Fault injection sull'acquisizione di risorse.
- traceRM.log

```
Resource res1 = new Resource();
Resource res2 = new Resource();
Task task1 = new Task(
    35,
    35,
    List.of(
        new Chunk(
            1,
            new UniformSampler(new BigDecimal(2), new BigDecimal(2))),
        new Chunk(
            2,
            new UniformSampler(new BigDecimal(1), new BigDecimal(1.5)),
            List.of(res1)),
        new Chunk(
            3,
            new UniformSampler(new BigDecimal(0.5), new BigDecimal(1)))));
Task task2 = new Task(
    50,
    50,
    List.of(
        new Chunk(
            1,
            new UniformSampler(new BigDecimal(3), new BigDecimal(4))),
        new Chunk(
            2,
            new UniformSampler(new BigDecimal(3), new BigDecimal(3.5)),
            List.of(res2)),
        new Chunk(
            3,
            new UniformSampler(new BigDecimal(3), new BigDecimal(3.5)))));
Task task3 = new Task(
    80,
    80,
    List.of(
        new Chunk(
            1,
            new UniformSampler(new BigDecimal(4), new BigDecimal(5))),
        new Chunk(
            2,
            new UniformSampler(new BigDecimal(4), new BigDecimal(4.5)))));
Task task4 = new Task(
    110,
    110,
    List.of(
        new Chunk(
            1,
            new UniformSampler(new BigDecimal(5), new BigDecimal(5.5))),
        new Chunk(
            2,
            new UniformSampler(new BigDecimal(2), new BigDecimal(3)),
            List.of(res2))));
Task task5 = new Task(
    160,
    160,
    List.of(
        new Chunk(
            2,
            new UniformSampler(new BigDecimal(3.5), new BigDecimal(4)),
            List.of(res1))));
TaskSet taskSet = new TaskSet(Set.of(task1, task2, task3, task4, task5));
ResourcesProtocol pcp = new PriorityCeilingProtocolFaultAquireResource(0.2);
Scheduler rm = new RMScheduler(taskSet, pcp, 1500);
rm.scheduleDataset(10);
```

Real-time Scheduling Simulator

Edoardo Sarri

Grazie