Scheduler Simulator

Marco Ronzani, Alessandro Sassi

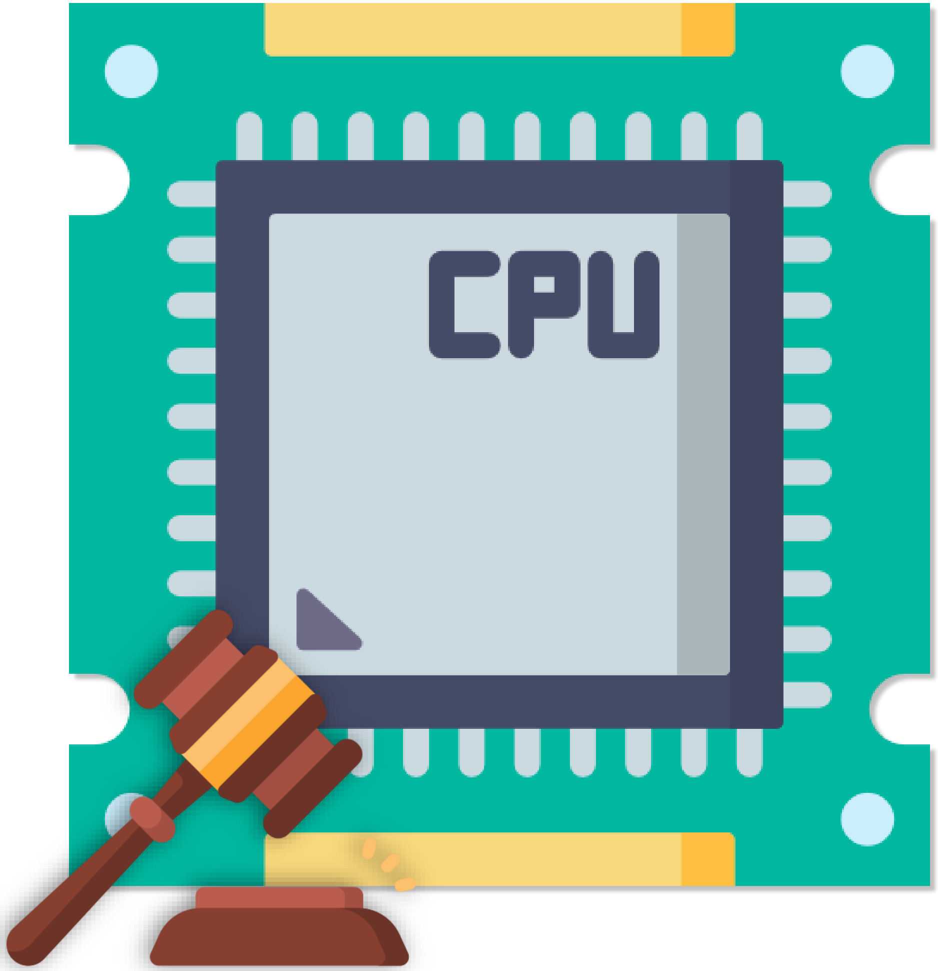
Advanced Operating Systems Project

2023-2024

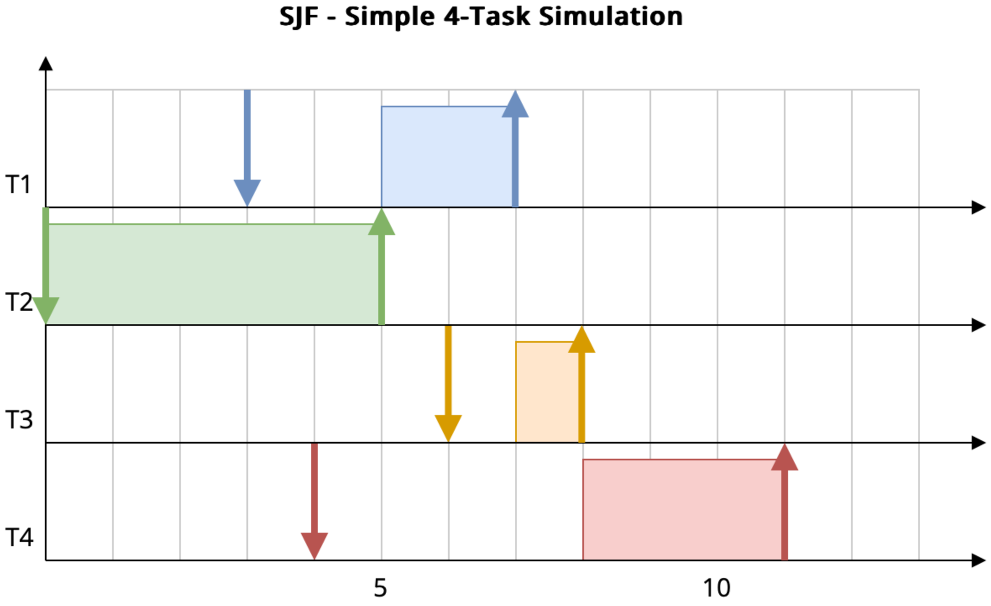
POLITECNICO
MILANO 1863

What is a Scheduler?

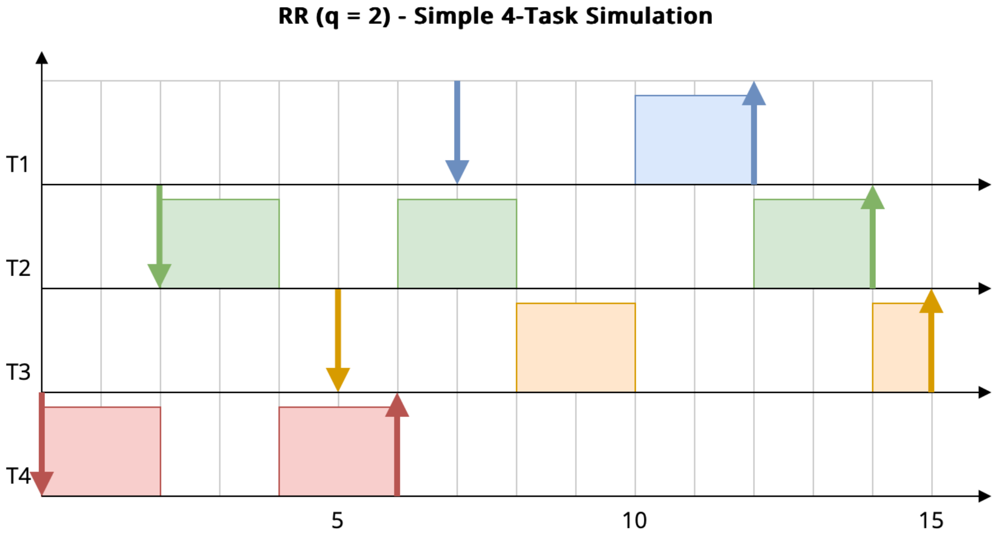
- A **CPU scheduler** is the component of an OS responsible for managing the allocation of **CPU processing time** among various **tasks**
- Objective is to optimize system performance, ensuring **fairness**, **responsiveness**, and **throughput**
- Selects which **process to execute** next from the ***ready queue*** based on scheduling algorithms such as *Round-Robin*, *Shortest-Job-First*



Schedule Example



SJF - Simple 4-Task Simulation

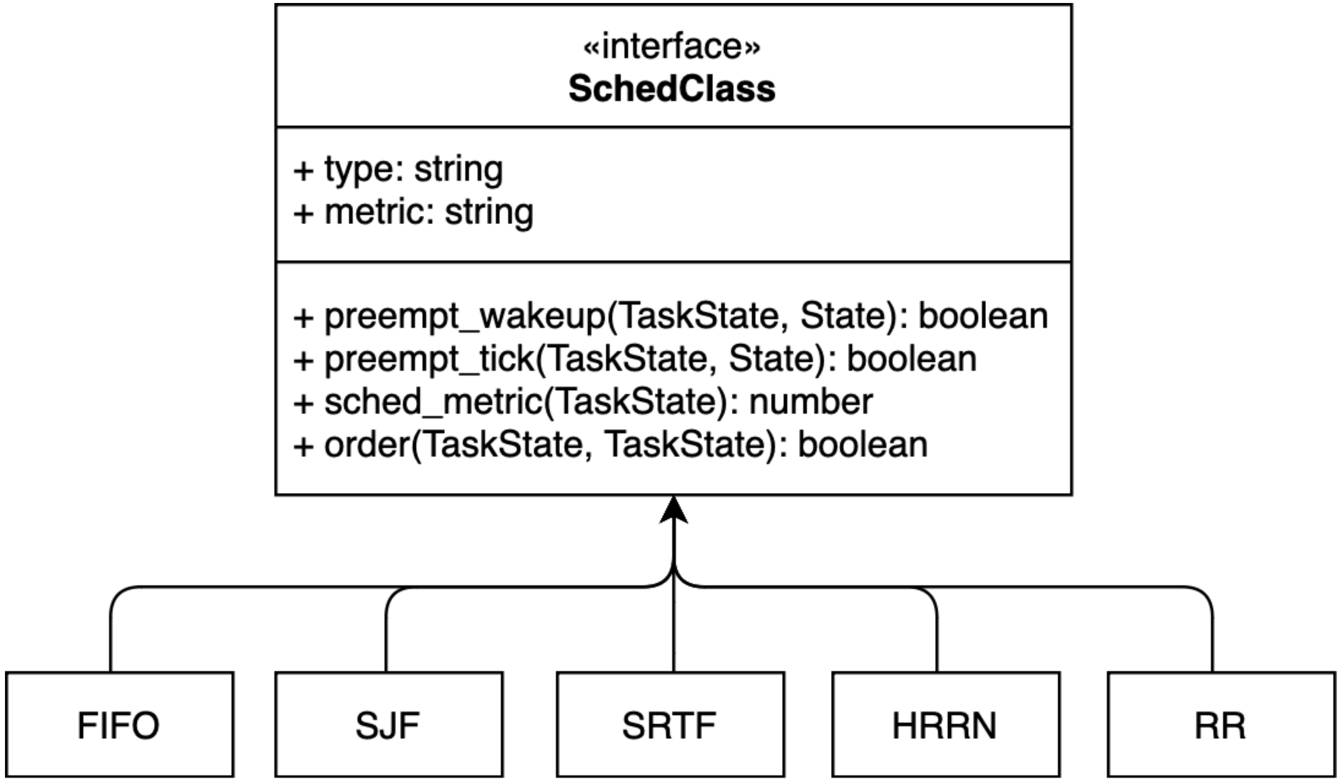


RR (q = 2) - Simple 4-Task Simulation

What is our Scheduler Simulator

- A simulator to enable the rapid **inspection and comparison** of different scheduling algorithms
- Extension of an existing **discrete-time scheduler simulator** made for CFS
- Add support for
  - **FIFO** (First-In, First-Out)
  - **SJF** (Shortest Job First)
  - **SRTF** (Shortest Remaining Time First)
  - **HRRN** (Highest Response Ratio Next)
  - **RR** (Round- Robin)

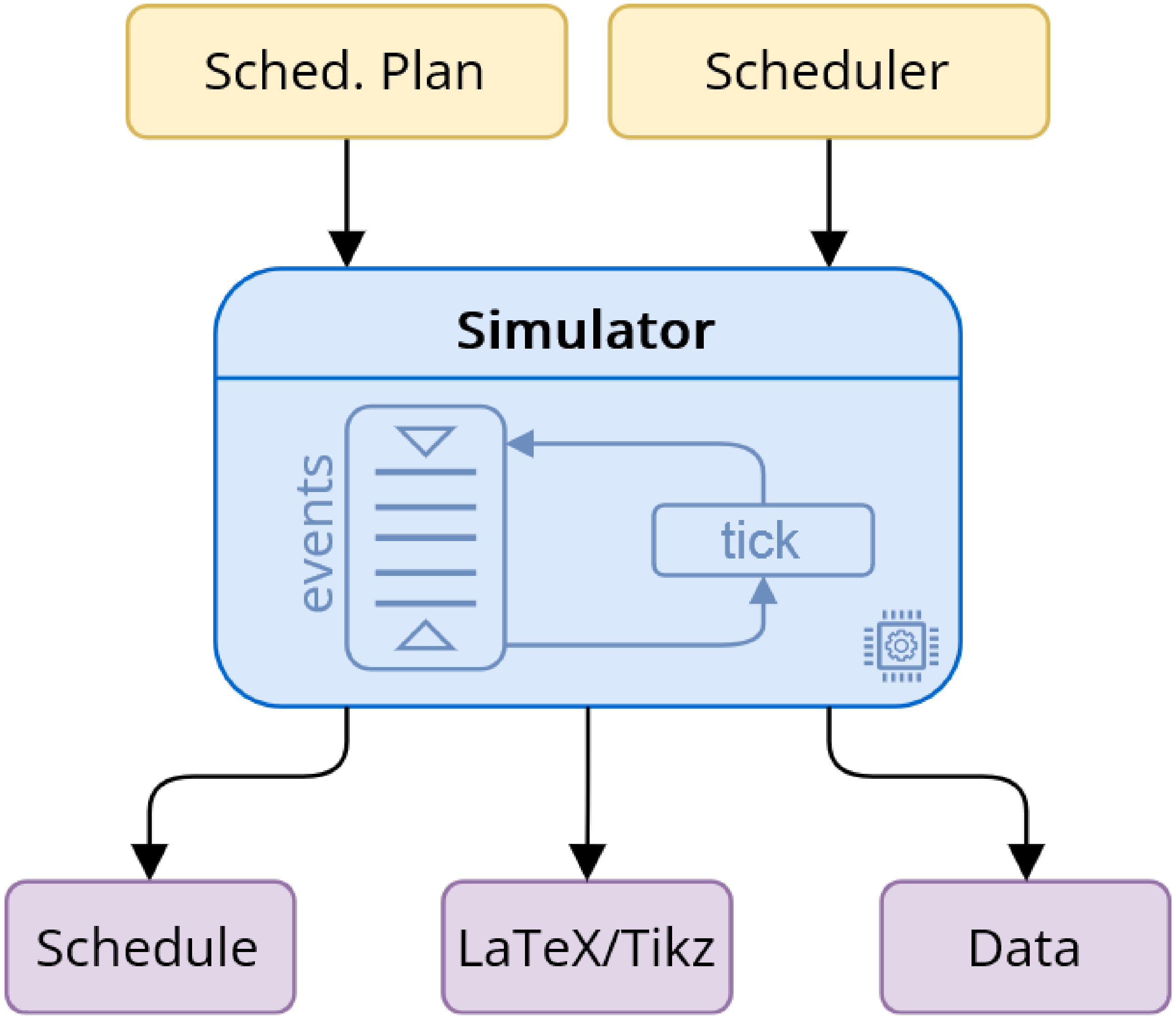
Design: Extensible Architecture


«interface»
SchedClass
+ type: string
+ metric: string
+ preempt_wakeup(TaskState, State): boolean
+ preempt_tick(TaskState, State): boolean
+ sched_metric(TaskState): number
+ order(TaskState, TaskState): boolean
FIFO
SJF
SRTF
HRRN
RR

- Enables the addition of new schedulers with **minimal disruption** to the existing code-base
- Algorithms implemented in their own **scheduler class**
- **Common interface** between scheduling algorithms and the simulation engine
- We applied the **Strategy Pattern**

Design: The Simulator

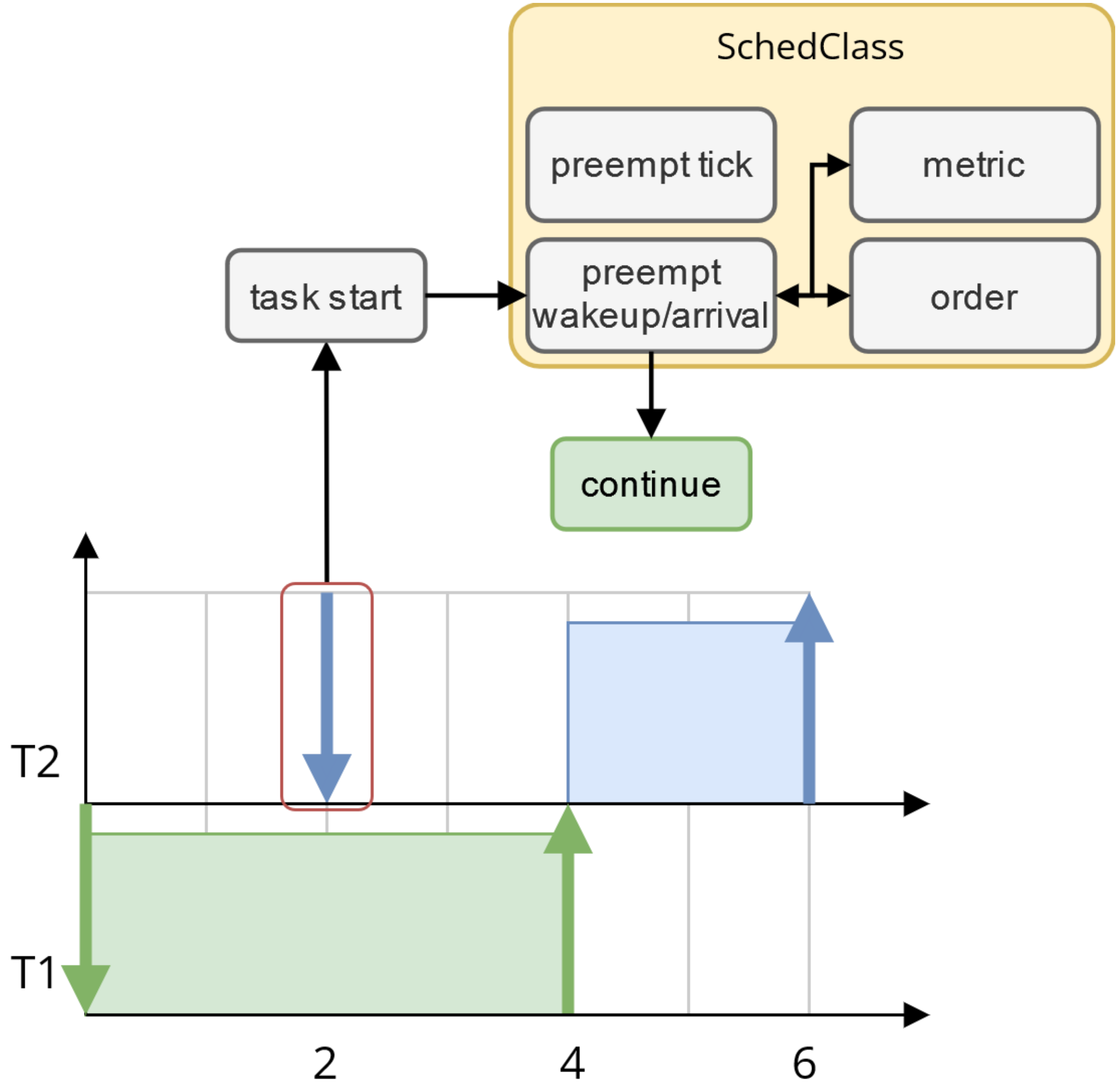
- One **simulator in common** to all schedulers
- **Event-driven** and simulator
- Events:
  - *tick*
  - *task start*
  - *task sleep*
  - *task wakeup*
- **SchedClass methods** dictate the simulator's behaviour



Event Example: Task Start

1. Task t_1 is running
2. Task t_2 arrives at $t = 2$, causing a ***task start*** event
3. The event:
   1. Adds the task to the runqueue
   2. Initializes the task's statistics
   3. Calls the current SchedClass's ***preempt*** method to decide on preemption
4. In this case, t_1 continues

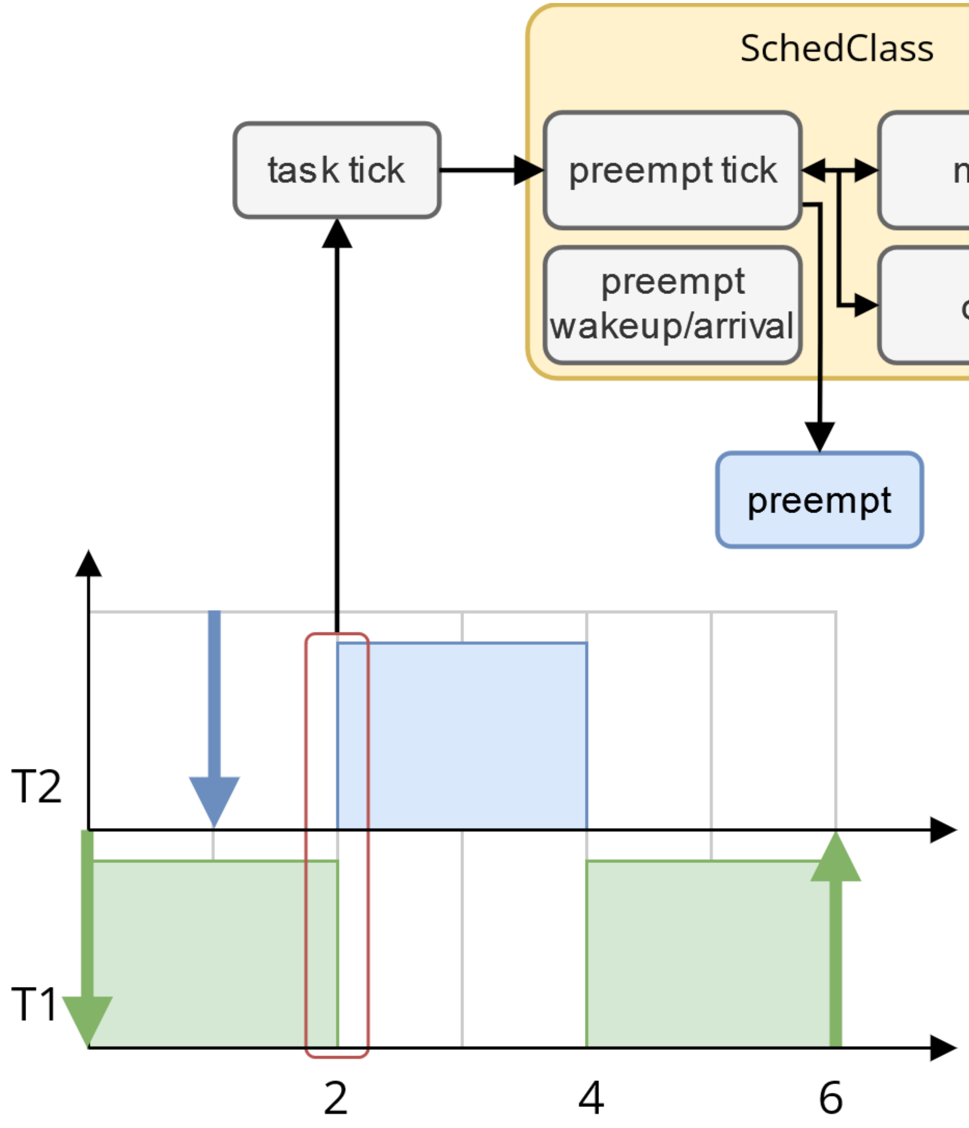
This is how the SchedClass interacts with the simulation.



Event Example: Task Tick

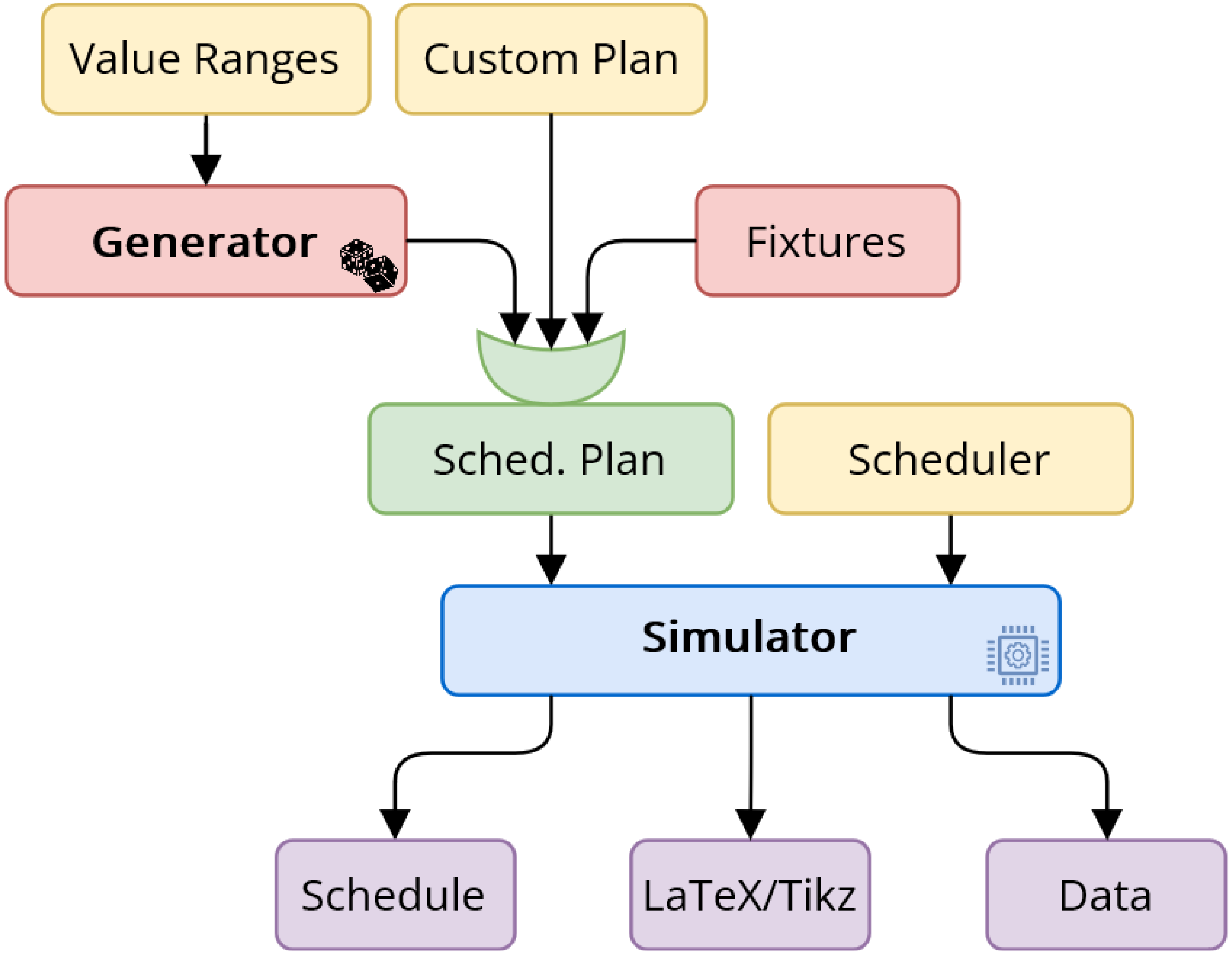
1. Task t_1 is running ($quantum = 2$)
2. Task t_2 waiting
3. The ***task tick*** event at $t = 2$:
   1. Updates task statistics
   2. Checks for events on the task
   3. Calls the current SchedClass's ***preempt*** method to decide on preemption
4. In this case, t_1 is preempted

This is how the SchedClass interacts with the simulation.

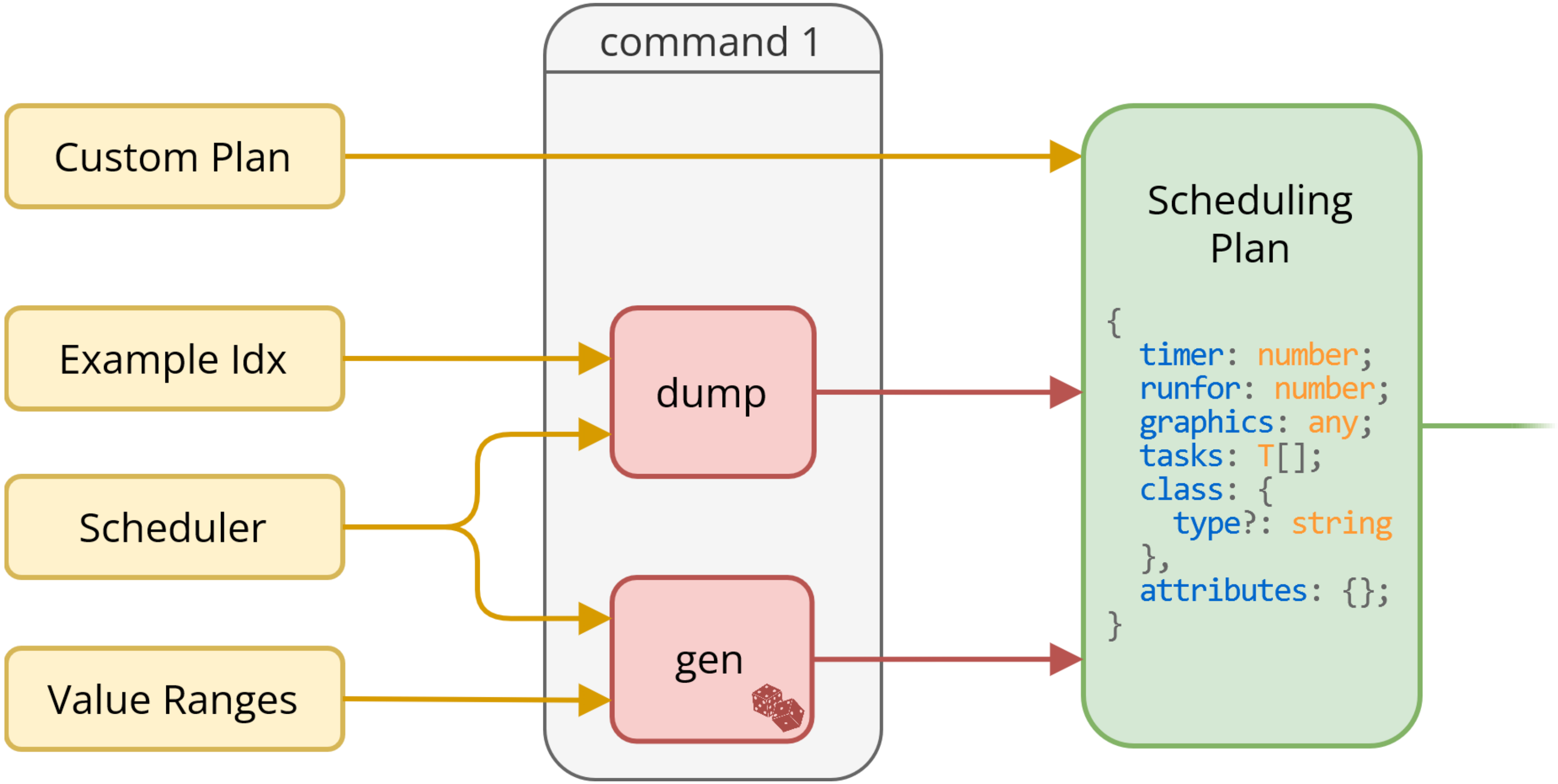


Scheduling Plan Generator

- We developed an **automated generator** of scheduling plans
- Selects parameter values within default or user-specified ranges
- Used to **discover of edge cases** and improve to the simulator



Functionalities: Details



command 1

Custom Plan

Example Idx

Scheduler

Value Ranges

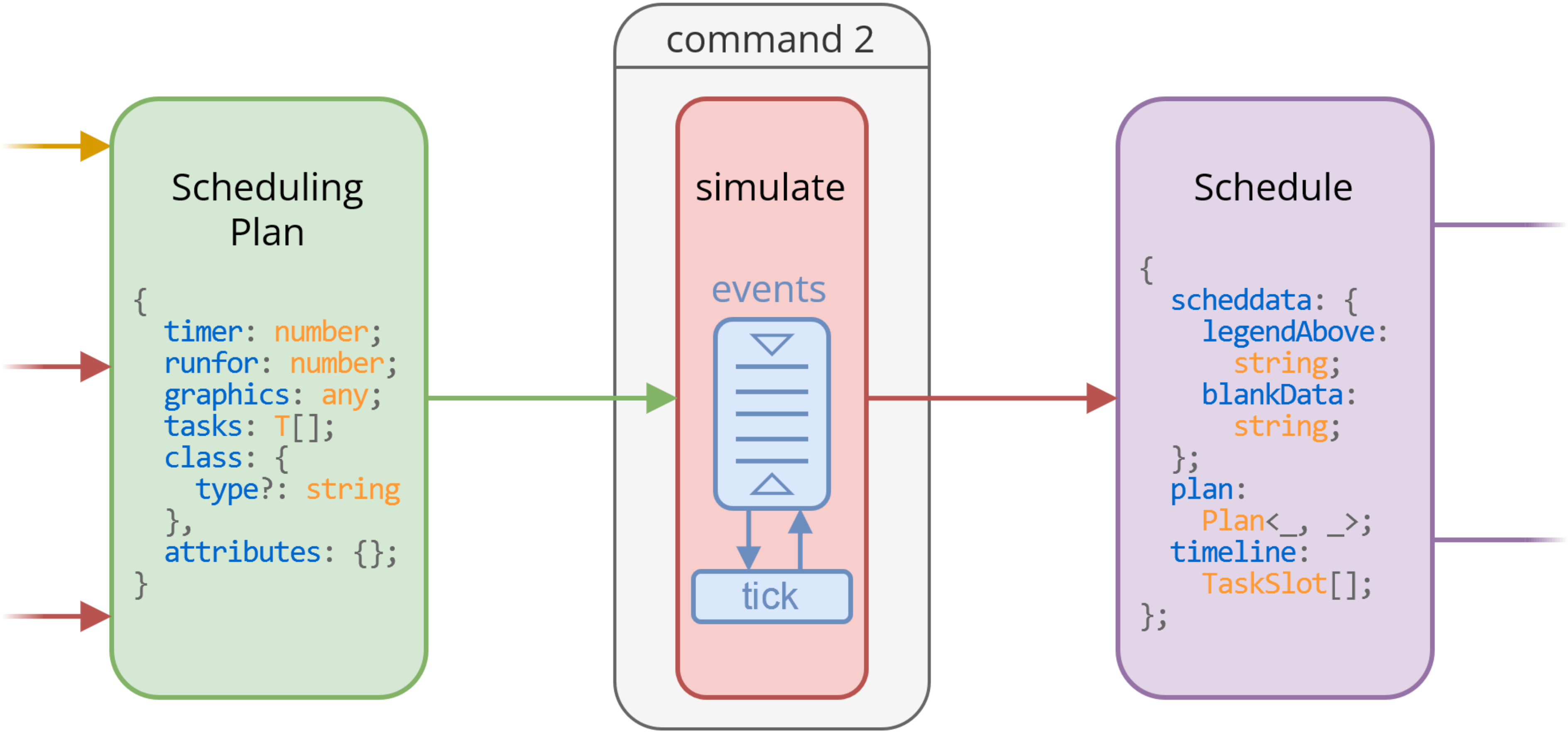
dump

gen

Scheduling Plan

```
{
  timer: number;
  runfor: number;
  graphics: any;
  tasks: T[];
  class: {
    type?: string
  },
  attributes: {};
}
```

Functionalities: Details



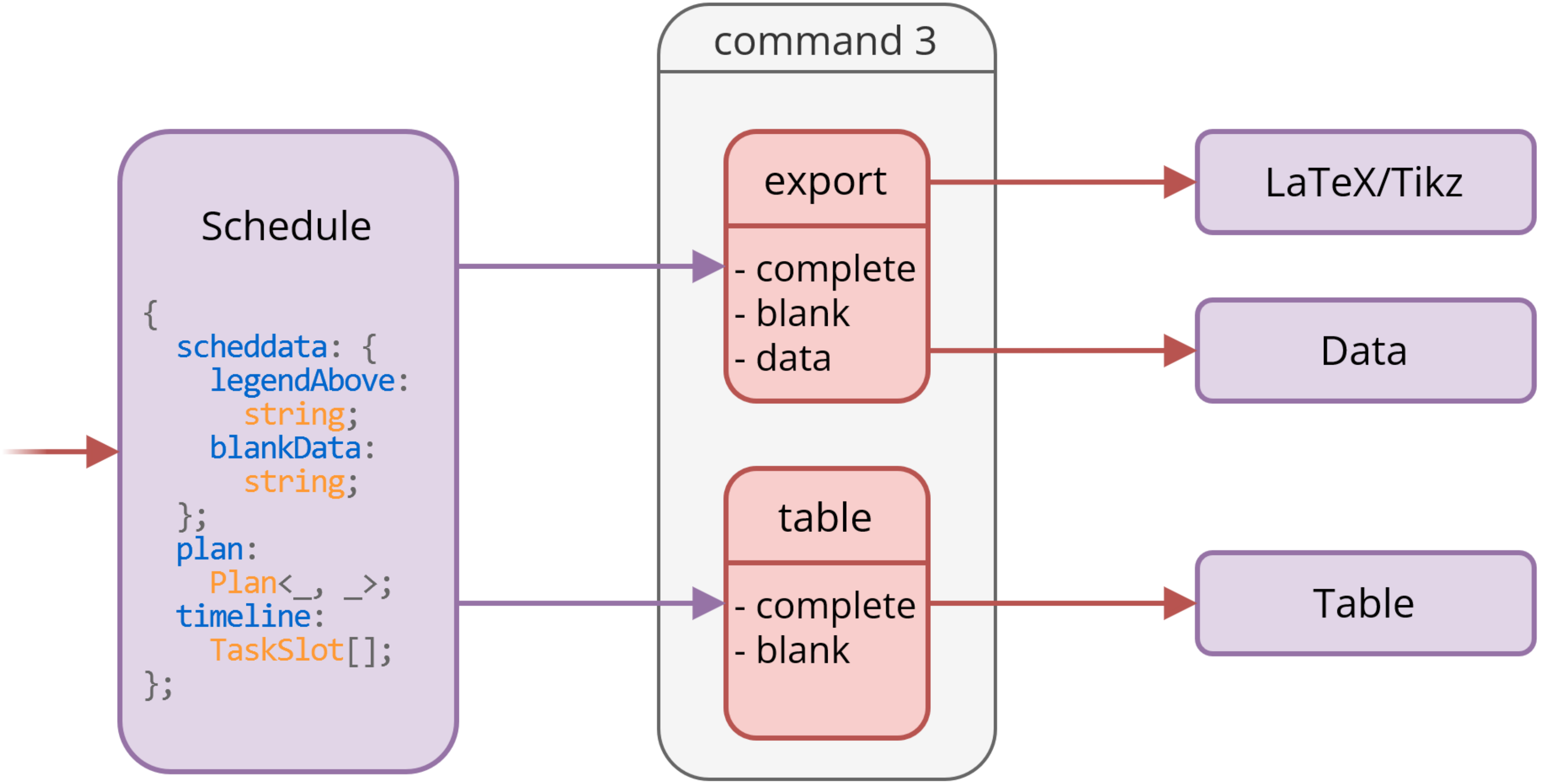
Scheduling Plan

```
{
  timer: number;
  runfor: number;
  graphics: any;
  tasks: T[];
  class: {
    type?: string
  },
  attributes: {};
}
```

Schedule

```
{
  scheddata: {
    legendAbove:
      string;
    blankData:
      string;
  };
  plan:
    Plan<_, _>;
  timeline:
    TaskSlot[];
};
```

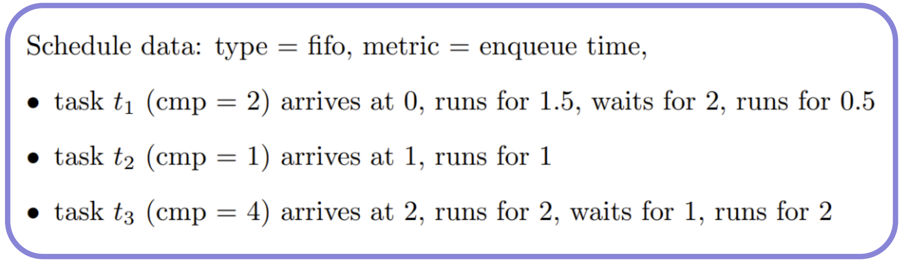
Functionalities: Details



```
{
  scheddata: {
    legendAbove:
      string;
    blankData:
      string;
  };
  plan:
    Plan<_, _>;
  timeline:
    TaskSlot[];
};
```

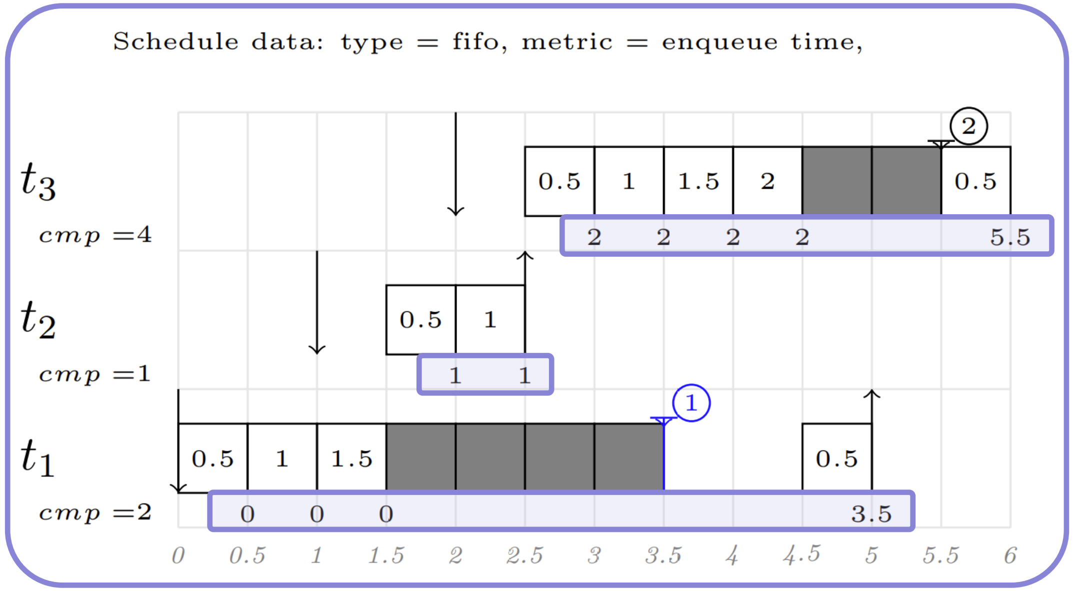
Other Additions & Improvements

- Data fixed and translated to **English**
- LaTeX generation with **generic metrics**
- Improved **layout** of the schedule graph
- New schedule **summary table**
- **Fixes** to CFS
- **Additional tests** to account for edge cases
- **Containerization** of the application

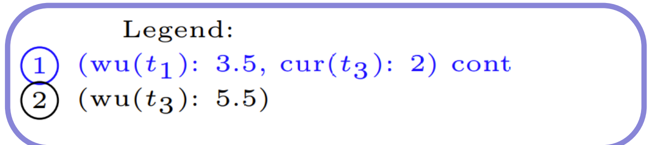


Schedule data: type = fifo, metric = enqueue time,

- task t_1 (cmp = 2) arrives at 0, runs for 1.5, waits for 2, runs for 0.5
- task t_2 (cmp = 1) arrives at 1, runs for 1
- task t_3 (cmp = 4) arrives at 2, runs for 2, waits for 1, runs for 2



Schedule data: type = fifo, metric = enqueue time,



Legend:

① (wu(t_1): 3.5, cur(t_3): 2) cont

② (wu(t_3): 5.5)

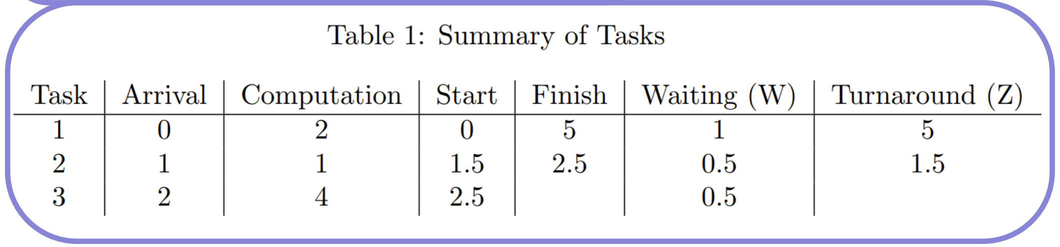


Table 1: Summary of Tasks

Task	Arrival	Computation	Start	Finish	Waiting (W)	Turnaround (Z)
1	0	2	0	5	1	5
2	1	1	1.5	2.5	0.5	1.5
3	2	4	2.5		0.5	

Frameworks & Runtimes

Bun.JS
Runtime

TypeScript
Source Code Language

Docker
Container Environment

WebKit
Debugger