INSTABASE

Distributed Workflow Processing

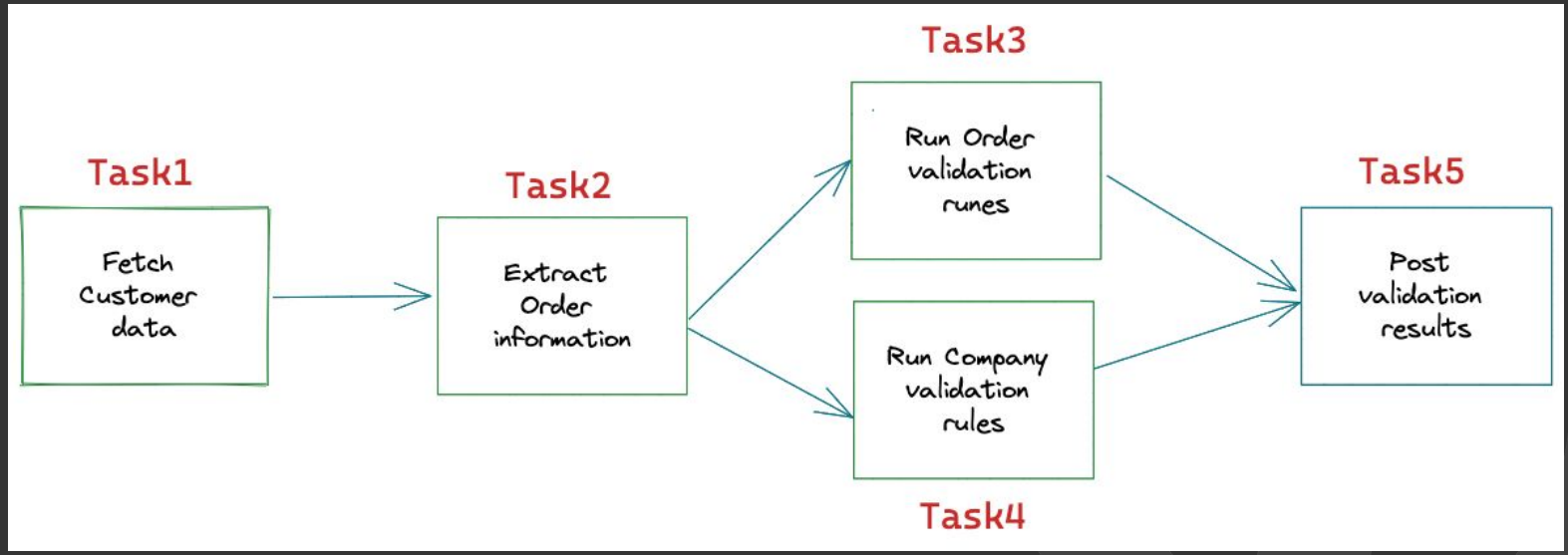
Problem Statement

Workflows are the mechanism by which businesses accomplish their work, processing data. Processing data involves a collection of tasks that may be executed in series or parallel. A workflow can be defined as any such combination of these tasks joined together to form a directed acyclic graph. These tasks could be independent or dependent on other tasks.

<Encipher/>

Coding the next generation of automation applications

Below is a sample workflow:



In the above diagram, we have 5 different tasks.

- Task1 - Fetch customer data
- Task2 - Extract order information
- Task3 - Run order validation rules
- Task4 - Run company validation rules
- Task5 - Post validation results

Task can be specified as:

```
{
      "name": "task3",
      "description": "Run Order validation rules",
      "cost": 20,
      "dependencies": [
        "task2"
      ]
 }
```

Each task can depend on one or more previous tasks and has a certain cost (time to execute) associated with it. Dependent tasks execute serially one after the other, while tasks in adjacent branches can be executed in parallel to optimize overall execution time.

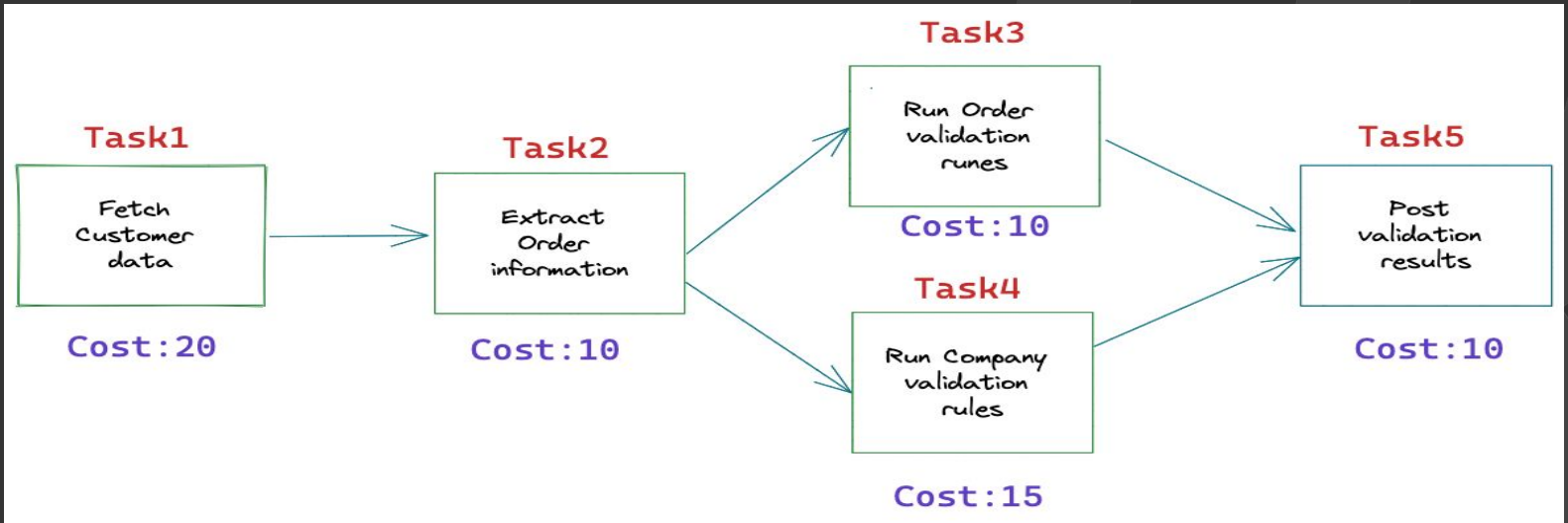
In the above example, Task1 and Task2 have to run serially one after the other. Task3 and Task4 can run in parallel after Task2 is completed. Task5 is dependent on both Task3 and Task4 and can only start after both are completed.

The goal of this problem is to design and implement a distributed workflow execution system which can execute any number of workflows in parallel.

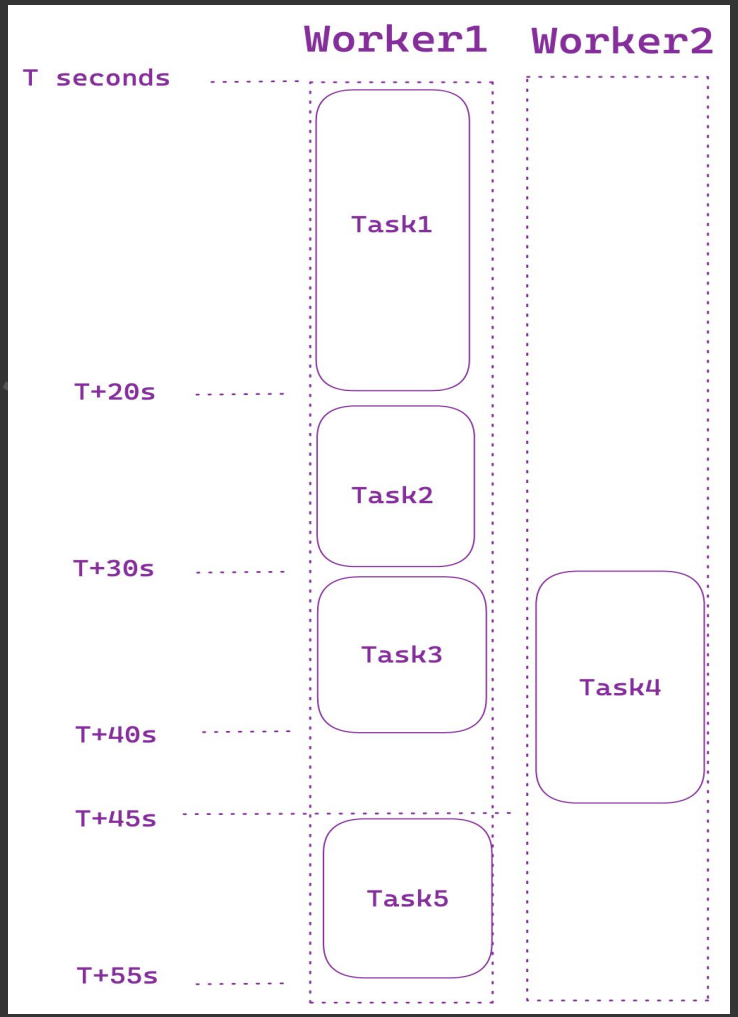
Example workflow run 1

Assume, we have 2 workers available in the workflow system and it needs to execute below workflow with tasks and costs indicated :

A possible representation for the execution of the above mentioned workflow maintaining task dependencies can be:



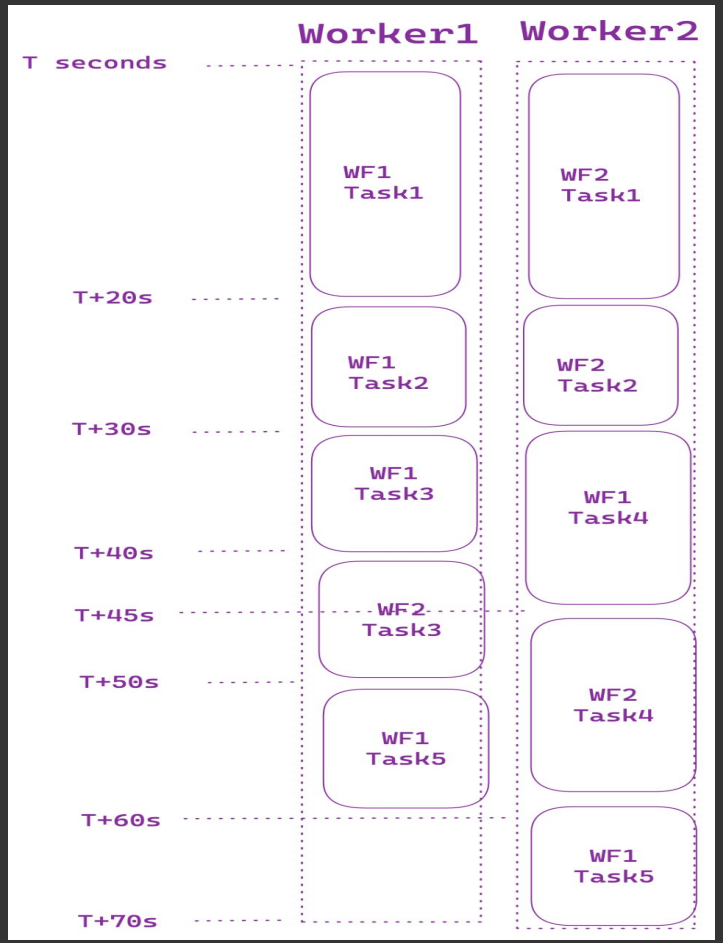
To be continued



Given workflow can be processed by the execution system in 55 seconds.

To be continued

Workflow Example 2



Assume, we have 2 workers running and we have 2 such workflows (WF1, WF2) - required to execute simultaneously. A possible representation for the execution can be as shown below where both workflows complete in 70 seconds.

Similarly we can have any number of workflows running at any time on the system.

To be continued

Workflow system should support:

- Executing workflows respecting task dependencies.
- Running workflows in parallel.
- Fairness - A workflow with more tasks / more time consuming tasks shouldn't hog all the resources. Minimize the median execution time for workflow execution across all workflows.
- Each input workflow would have a scheduled_at field, this field has to be respected when scheduling workflows in the execution system.

Sample Input / Output

Your code should define a function interface as defined below.

```
process_workflows(workflows, worker_count)
```

To be continued

Workflow Definition

JSON representation for a workflow is indicated below.

cost represents time spent (seconds) for executing each task.

scheduled_at represents the time when this workflow was added to the system (epoch time).

Example Workflow 1

```
{
 "name": "workflow1",
 "scheduled_at": 1641480759,
 "tasks": [
   {
     "name": "task1",
     "description": "Fetch customer data",
     "cost": 20,
     "dependencies": []
   },
```

```
{
    "name": "task2",
    "description": "Extract order information",
    "cost": 10,
    "dependencies": [
      "task1"
    ]
  },
  {
    "name": "task3",
    "description": "Run Order validation rules",
    "cost": 10,
    "dependencies": [
      "task2"
    ]
  },
  {
    "name": "task4",
    "description": "Run Company validation rules",
```

```
    "cost": 15,
    "dependencies": [
      "task2"
    ]
  },
  {
    "name": "task5",
    "description": "Post validation results",
    "cost": 10,
    "dependencies": [
      "task3",
      "task4"
    ]
  }
 ]
}
```

Example workflow 2


```
{
"name": "workflow2",
"scheduled_at": 1641480759,
"tasks": [
  {
    "name": "task1",
    "description": "Fetch customer data",
    "cost": 20,
    "dependencies": []
  },
  {
    "name": "task2",
    "description": "Extract order information",
    "cost": 10,
    "dependencies": [
      "task1"
    ]
  },
```

```
{
    "name": "task3",
    "description": "Run Order validation rules",
    "cost": 10,
    "dependencies": [
      "task2"
    ]
  },
  {
    "name": "task4",
    "description": "Run Company validation rules",
    "cost": 15,
    "dependencies": [
      "task2"
    ]
  },
  {
    "name": "task5",
    "description": "Run Shipping validation rules",
    "cost": 15,
    "dependencies": [
```

```
      "task2"
      ]
},
   {
     "name": "task6",
     "description": "Refine order data",
     "cost": 10,
     "dependencies": [
       "task3"
     ]
   },
   {
     "name": "task7",
     "description": "Refine company data",
     "cost": 15,
     "dependencies": [
       "task4"
     ]
   },
   {
     "name": "task8",
```

```
    "description": "Refine shipping data",
    "cost": 15,
    "dependencies": [
      "task5"
    ]
  },
  {
    "name": "task9",
    "description": "Post refinement results",
    "cost": 10,
    "dependencies": [
      "task6",
      "task7",
      "task8"
    ]
  }
 ]
}
```

Example workflow 3


```
{
 "name": "workflow3",
 "scheduled_at": 1641480799,
 "tasks": [
   {
     "name": "task1",
     "description": "Fetch customer data",
     "cost": 5,
     "dependencies": []
   },
   {
     "name": "task2",
     "description": "Extract personal customer information",
     "cost": 5,
     "dependencies": [
       "task1"
     ]
 }
 ]
}
```

The arguments passed to your function would be an array of workflows as defined above and the number of workers.

Sample call: *process_workflows(workflows, 5)*

Sample Output

The function should return a json with scheduled_at and completed_at timestamps for each workflow, as well as the the worker, started_at, completed_at for each task in the workflow.

started_at - represents the time when this task started executing. Any task in a workflow shouldn't start before the scheduled_at time of the workflow.
completed_at - represents the time when this task completed executing.

To be continued

Output Format


```
[
{
 "name": "workflow1",
 "scheduled_at": 1641480759,
 "completed_at": 1641480814,
 "tasks": [
   {
     "name": "task1",
     "worker": "w1",
     "started_at": 1641480759,
     "completed_at": 1641480779,

   },
   {
     "name": "task2",
     "worker": "w1",
     "started_at": 1641480779,
```

```
    "completed_at": 1641480789,

},
{
  "name": "task3",
  "worker": "w1",
  "started_at": 1641480789,
  "completed_at": 1641480799,

},
{
  "name": "task4",
  "worker": "w2",
  "started_at": 1641480789,
  "completed_at": 1641480804,

},
{
  "name": "task5",
  "worker": "w1",
```

```
 "started_at": 1641480804,
     "completed_at": 1641480814,

    }
 ]
}
]
```

Expectation

- Presentation / write up detailing the workflow system design - frameworks used, how system requirements are implemented.
- Design and concrete code implementation of orchestration component including:
  - Scheduling and execution of workflows and tasks.
  - Efficient execution of workflows at scale while maintaining task dependencies and ensuring fairness.
  - Minimize the median execution time for workflow execution across all workflows. Execution time is calculated as the difference between *completed_at* and *scheduled_at* time of the workflow.
- Ability to invoke an API / function which takes input workflows per specification provided above.
- Generate results using the sample workflows mentioned in the Sample Input/Output section. Consider example workflows 1, 2 and 3 as input with a worker count of 2. The generated output should be in the prescribed format and stored in an output.json file. This file is to be attached with your submission.

Considerations

- Advice to start at least 3-4 hours before the deadline for submission towards implementation, once the approach and design is finalized.
- Recovery and Resilience are out of scope for this problem solution.
- Pause and cancel workflows, workflow priorities are also out of scope for this problem.
- For implementation, any programming language is accepted. Also solutions using open source frameworks / libraries are also accepted, it would be ideal to indicate the rationality for the choices.

Guidelines

- For your submission, only attach code implementations and remove any binaries and output libraries from your submission zip file.
- Attach final result output json as part of your submission.
- Include details for software versions used, any third party libraries in a readme file. Also include instructions for setting up the solution and executing the solution.

Judgment Criteria:

- The jury evaluates each submission by means of the automated scoring function of the competition platform and/or manual inspection.
- A participant’s final score for a problem statement round is based on the overall submission of the code file, presentation, and video summary.
- The quality of the code will be considered in scoring, the jury will review the code to verify that the solution is original.
- The jury may determine the originality of code based on a review of the code alone and/or comparison against other code submissions.