Emulator- Design and implementation decision support system for Workload Management

Why do we need an Emulator?

- Provides a measure of correctness and efficiency of the system before it is physically(or virtually) constructed, highlighting the constraints that might exist in the design and development.
- Emphasis on analyzing the designing stage using the Emulator rather than the post-construction stage of a project helps in reducing the overall time and cost.
- Measure scalability of computing resources for a functional and existing design efficiently.

RELATED WORK

1. IOTSim: A simulator for analysing IoT applications
    a. Defines an Application model, Network and storage model, and Big Data Processing model.
    b. Does not make a decision about the type of resource for each task that reduces the total execution time.
2. WLMS Emulator
    a. Defines the two properties of resources and tasks - Heterogeneity and Dynamism
    b. A late-binding strategy where scheduling decisions are delayed till the tasks need to be executed
    c. Performance comparison between binding (Early and Late) task to a scheduler of the given resource.
    d. Does not make a decision about the type of resource for each task that reduces the total execution time.
3. Baetyl

Definitions

1. **Task**: Stand-alone process that has well defined input, output, termination criteria, and resources requirements.

2. **Workload**: Set of tasks whose dependencies have been satisfied and can be concurrently executed.

3. **Resource:** A hardware system that can provide computational ability and storage memory.
    - **a.** Edge
        - i. Can solve latency issues
    - **b.** Fog/Cloudlet
    - **c.** Cloud
        - i. Can solve scalability issues

Mathematical Modelling (min use-case)

Workload : { $Task_1$, $Task_2$, ... , $Task_{n-1}$, $Task_n$ }

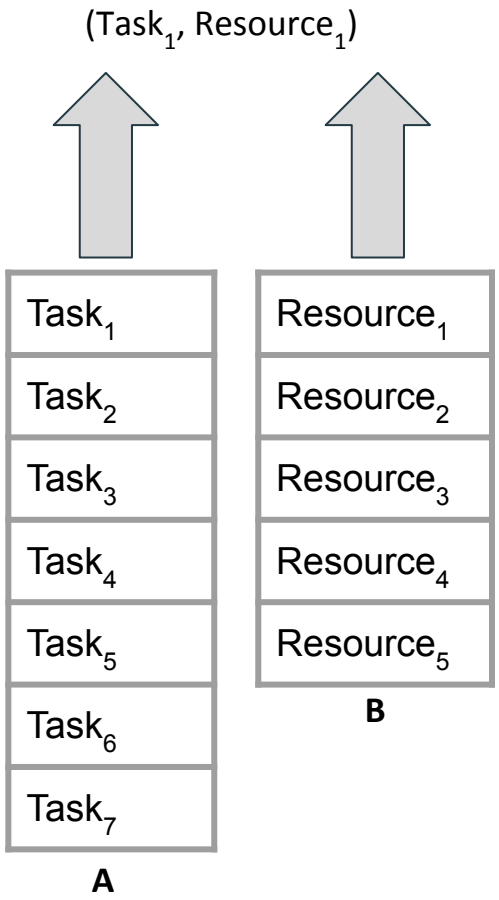
Resource : { $Resource_1$, $Resource_2$, ..., $Resource_{m-1}$, $Resource_m$}

Algorithm : { $Algorithm_1$, $Algorithm_2$, ..., $Algorithm_{p-1}$, $Algorithm_p$}

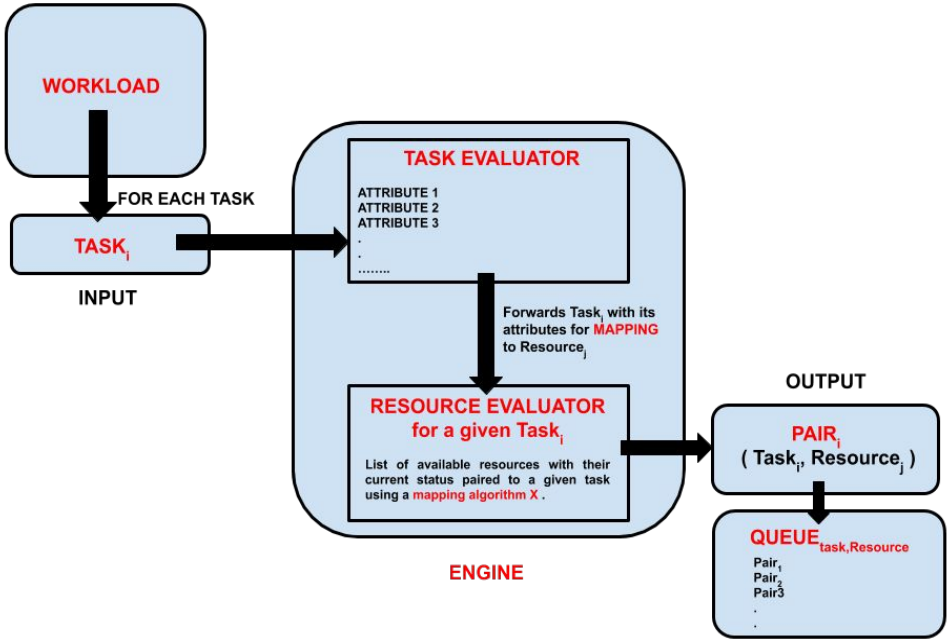
Engine : { $Algorithm_k$: $Task_i \Longrightarrow Resource_j$ } where i ε 1:N and j ε 1:M

For Instance,
Container A : Queue for Tasks
Container B : Priority Queue for Resources based on Computational Power

Algorithm: First Task mapped to the resource with the highest Computational Power



Reference Architecture



TASK EVALUATOR: Identifies the characteristics associated with a task that are presented by the user such as minimum execution time required, minimum memory required, or/and latency range for completion.

RESOURCE EVALUATOR: For instance, evaluates the processing speed, memory capacity, filesystem type at a given time and pairs a resource with the given task for execution of the task.

Emulator API - min-use case