Simple & Transparent Resource Exchange And Management

STREAM: Course Project Plan

Isa Jafarov, Nan Jia, Alex Washburn, Rose Wong
AUTHORS

Saptarshi Debroy
INSTRUCTOR

2022 – 10 – 31
DATE

CSc – 85011
COURSE №

Computer Science
DEPARTMENT

Distributed and Cloud Computing
COURSE NAME

1 Introduction

The course project comprises the creation of a distributed "Science Broker" which manages requests for scientific resources and information. We will fulfill the project requirements by designing and implementation a service which receives requests for resources to satisfy a scientific job submission. Furthermore, the service will be implemented in a distributed manner, permitting the sharing of resources across collaborating scientific "domains."

2 Design

2.1 Overview: Utilize GENI multi-cloud topology

We will use GENI to simulate multiple collaborating scientific institutions. Each scientific organization will form a **`Domain`**. A **`Domain`** is a network topology of resources contained entirely within a "real" GENI site. Multiple **`Domain`** will be connected together within GENI to form a multi-could topology. Every **`Domain`** has an **`Endpoint`** facilitating **`User`** access to the entire multi-could topology which comprises the Science Broker Service. A **`User`** can submit a **`Job`** through an **`Endpoint`** by specifying the required resources.

Furthermore there will be an additional **`Domain`** containing containing the **`Broker`**. All resources across all **`Domain`**s are known to the **`Broker`**. Hence, the **`Broker`** handles scheduling, networking, and load balancing of submitted **`Job`**s.

2.2 Broker

The **`Broker`** agent/instance is the centralized manager of the Science Broker Service. Encapsulated within the **`Broker`** is the maintenance of an ACID database containing:

- A list of all incomplete **`Job`** submissions.
- A list of all resources across each **`Domain`**.
- A mapping of which resource(s) are allocated to which **`Job`**.
- A queue of pending **`Job`** submissions.

Additionally, the **`Broker`** executes a scheduling algorithm to determine which queued **`Job`** will be given which resource(s) and when a **`Job`** will be migrated.

2.3 Scheduling & Queuing

Testing possible scheduling algorithms that are:

- FCFS(first come first serve)
- SJF(shortest job first)
- RR(round robin)

Because we focus on monitoring our brokering network's resource management and performance. Those three algorithms have different advantages and disadvantages.

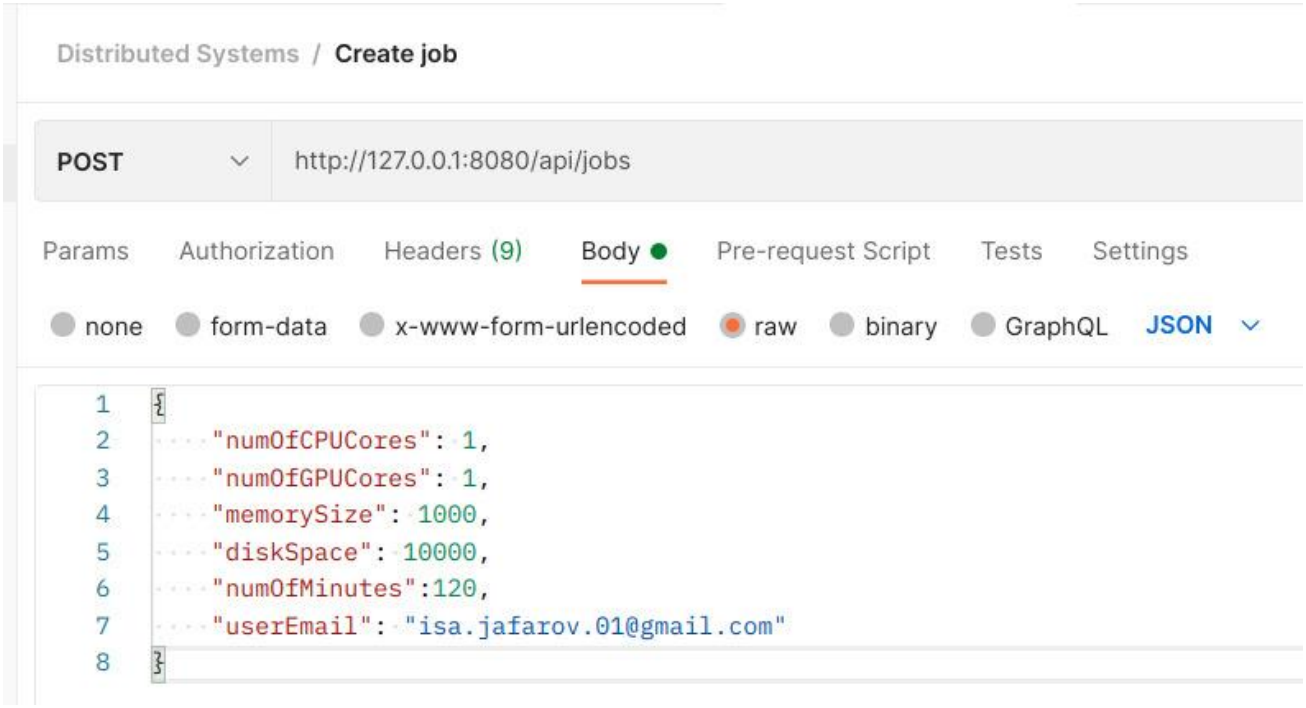

Figure 1: Example request received by "Back-end" **`Broker`**.

2.4 Domains

Within the multi-could topology, there are many **`Domain`**s. Each **`Domain`** contributes one or more computing resources to the multi-could topology. Additionally, each **`Domain`** locally hosts an **`Endpoint`** from which a **`User`** within the **`Domain`** can access the computing resources of the science brokering service by submitting a **`Job`**. Networked communication and computations across domains are dynamically mediated by the **`Broker`**.

2.5 Jobs

A **`Job`** contains the following information:

- **`Mail`**: String — An email address for the user, uniquely identifies user
- **`Time`**: $\mathbb{N}$ — Hard upper bound limit for job, user provides best effort
- **`Disk`**: MiB $\in \mathbb{Z}^+$ — Disk space requirements
- **`RAM`** : MiB $\in \mathbb{Z}^+$ — Memory requirements in MiB
- **`CPUs`**: $\mathbb{Z}^+$ — number of CPU cores/threads
- **`GPUs`**: $\mathbb{N}$ — GPU requirements
- **`Task`**: Binary — File of the executable to run
- **`Data`**: Array Binary — A list of data blobs to load into the disk space, total must be $\leq$ **`Disk`**

The **`Broker`** can process a **`Job`**, and decide which resources to allocate to fulfill the job request across the **`Domain`**s. When a **`Job`** is executed, it does so with a Docker container. This containerization is seamlessly performed by the science brokering service.

2.6 User Interface

The **User** submits a **Job** at an **Endpoint**. The **Endpoint** presents a User Interface (UI) to the **User**. The presented UI could be a hosted website, a terminal user interface (TUI), or a standalone graphical user interface (GUI). We will focus on a TUI for the initial implementation, with an HTML website UI as a stretch goal.

Required information of a **Job** is collected from the **User** by the **Endpoint** UI. Subsequently, the **Job** information is encoded as JSON by the **Endpoint** and forwarded to the **Broker**.

3 Work Delegation

3.1 Main task "umbrellas" to be completed:

- Backend
- Frontend
- Containerization
- Replication

3.2 Who will do what?

- Isa Jafarov: Backend Broker; centralized job/resource database, job queuing
- Nan Jia: Docker and Kubernetes startup scripts/containerization
- Alex Washburn: TUI **Endpoints**, scheduling algorithm
- Rose Wong: Backend Backup/Replication

4 Estimated Timeline

Week	Planned Work Items	Deliverable	Date
04 – 16	System design, GENI research	Project Plan	**04 – 23**
04 – 23	Frontend TUI, GENI experimentation, containerization	1st Presentation	**05 – 01**
04 – 30	Replication, backups, scheduling algorithm	— —	— —
05 – 07	Backend UI, job status feeback, resource utilization	2nd Presentation	**05 – 15**
05 – 15	Finalize everything!	Project Report	**05 – 22**

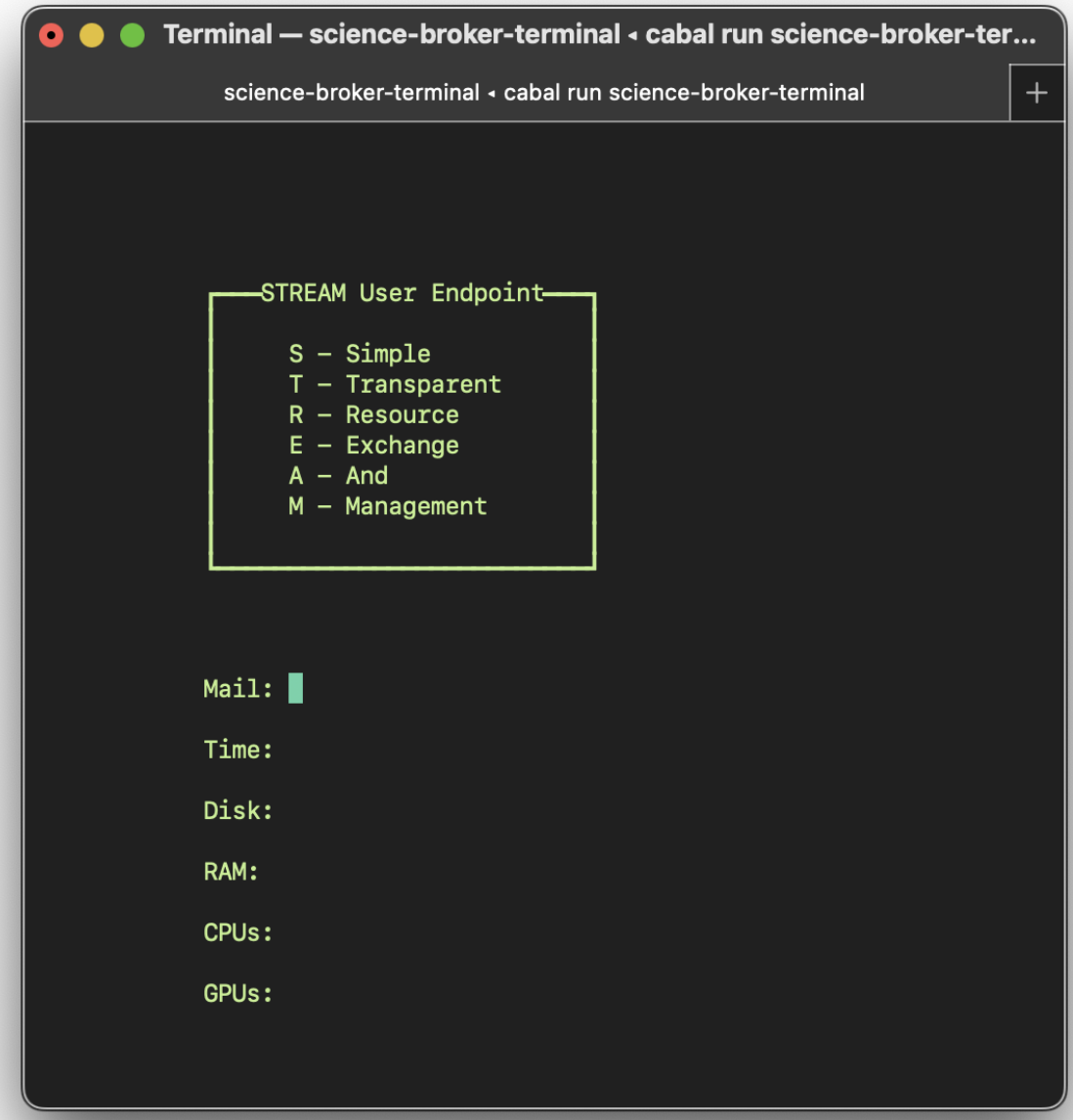

Figure 2: A mock-up of the “Front-end” Terminal User Interface.