
TRƯỜNG ĐẠI HỌC
BÁCH KHOA HÀ NỘI
HANOI UNIVERSITY
OF SCIENCE AND TECHNOLOGY

# Mô phỏng thuật toán Reinforcement Learning cho Serverless

Future Internet Laboratory

Nguyễn Phạm Trung Hiếu

Ngày 24 tháng 12 năm 2023

ONE LOVE. ONE FUTURE.

# Nội dung chính

HUST

- Chi tiết mô phỏng
  - Kịch bản mô phỏng
  - Mô hình hoá trình mô phỏng
  - Triển khai các tham số môi trường
- Triển khai mô phỏng và đánh giá
  - Kết quả mô phỏng
  - Đánh giá mô phỏng
- Kết luận

# Nội dung chính

HUST

- Chi tiết mô phỏng
  - Kịch bản mô phỏng
  - Mô hình hoá trình mô phỏng
  - Triển khai các tham số môi trường
- Triển khai mô phỏng và đánh giá
  - Kết quả mô phỏng
  - Đánh giá mô phỏng
- Kết luận

# Nội dung chính

HUST

- Chi tiết mô phỏng
  - Kịch bản mô phỏng
  - Mô hình hoá trình mô phỏng
  - Triển khai các tham số môi trường
- Triển khai mô phỏng và đánh giá
  - Kết quả mô phỏng
  - Đánh giá mô phỏng
- Kết luận

# Kịch bản mô phỏng

HUST

- Hệ thống khởi tạo ngẫu nhiên số lượng container và thời gian xử lý request ở mỗi service.
- Hệ thống có số service là 4 (`size = 4`), tức ma trận container có 4 hàng.
- Hệ thống nhận request vào mỗi $\Delta t = 10s$, thời gian tồn tại của request là $2s$.
- Mỗi 1 giây, hệ thống sẽ tính toán reward và các giá trị info (lượng RAM sử dụng, năng lượng tiêu thụ, profit...).
- Mỗi 1 giây, thực hiện kiểm tra lượng container ở trạng thái WarmCPU có trong mỗi service và so sánh với lượng request:
  - Nếu $n_{WarmCPU} \geq n_{request}$ thì $n_{WarmCPU} -= n_{request}$ rồi đặt $n_{request} = 0$, sau đó chờ thời gian xử lý, tính toán các tham số đánh giá, rồi cộng trở lại lượng WarmCPU vừa trừ.
  - Nếu $n_{WarmCPU} < n_{request}$ thì $n_{request} -= n_{WarmCPU}$ rồi đặt $n_{WarmCPU} = 0$, lượng request dư ra được đưa vào hàng chờ, sau đó xử lý tương tự như trường hợp trên.

# Điều kiện dừng



- Khi hệ thống chạy được một khoảng thời gian là `container_lifetime = 43200`,
- Hoặc thiếu tài nguyên (phần trăm sử dụng CPU hay GPU lớn hơn 1),
- Hoặc xảy ra hiện tượng tràn RAM...

# Điều kiện thay đổi trạng thái



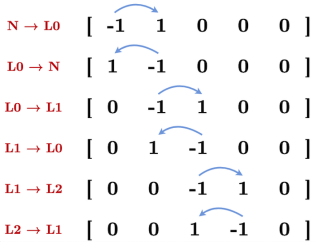
- Giới hạn sự thay đổi state chỉ giữa hai state liền kề, state chỉ thay đổi một mức mỗi lần.
- Biểu diễn ma trận đơn vị $M_u$ theo cách hiểu khác (nói thêm ở vấn đề gặp phải):
  - -1: state nguồn, tương ứng với state bị thay đổi
  - 1 : state đích, tương ứng với state thay đổi tới
  - 0 : state không bị thay đổi

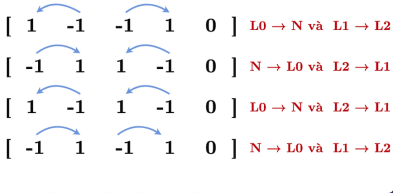
  $\longrightarrow$ Hai giá trị -1 và 1 luôn liền kề nhau (nếu có thay đổi ở hai vị trí state thì -1 và 1 sẽ được xếp theo từng cặp).

# Điều kiện thay đổi trạng thái

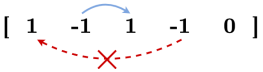
HUST

## Trường hợp thoả mãn




## Trường hợp không thoả mãn

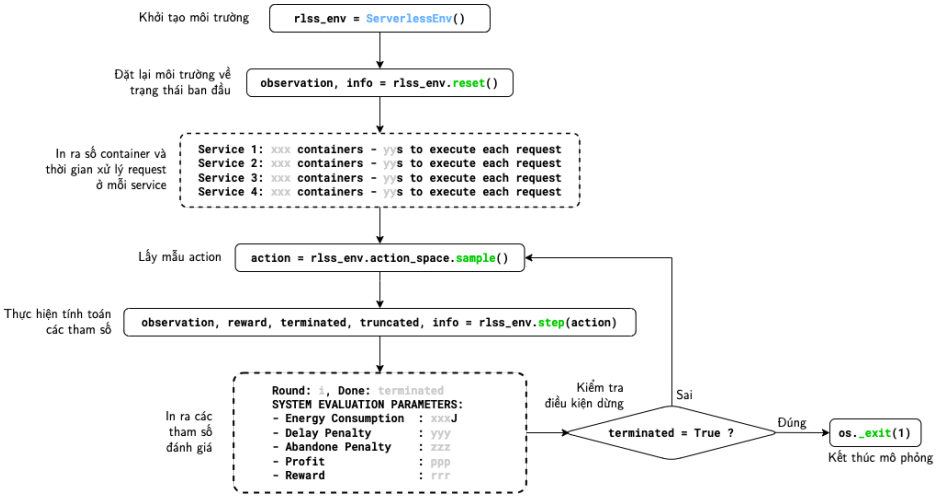



# Nội dung chính

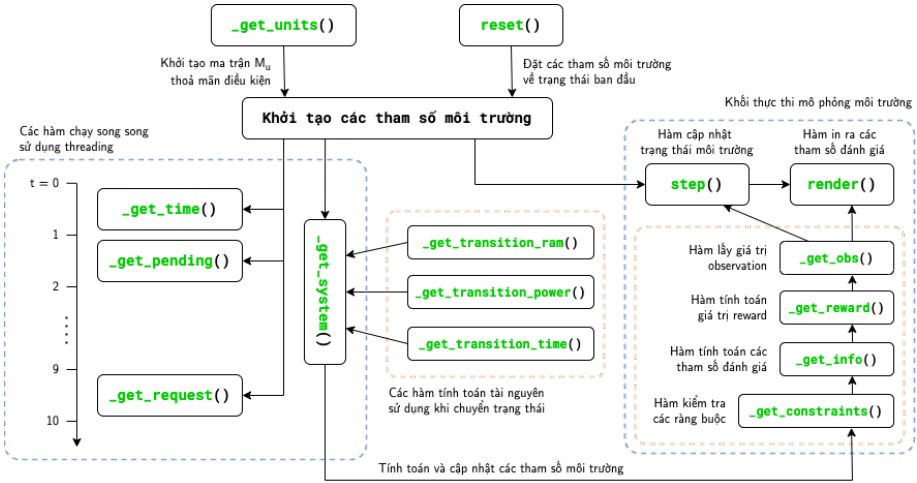
HUST

- Chi tiết mô phỏng
  - Kịch bản mô phỏng
  - Mô hình hoá trình mô phỏng
  - Triển khai các tham số môi trường
- Triển khai mô phỏng và đánh giá
  - Kết quả mô phỏng
  - Đánh giá mô phỏng
- Kết luận

# Quy trình mô phỏng



# Mối liên hệ giữa các hàm



# Nội dung chính

HUST

- Chi tiết mô phỏng
  - Kịch bản mô phỏng
  - Mô hình hoá trình mô phỏng
  - Triển khai các tham số môi trường
- Triển khai mô phỏng và đánh giá
  - Kết quả mô phỏng
  - Đánh giá mô phỏng
- Kết luận

# Triển khai các tham số môi trường

HUST

Các tham số chính của môi trường:

```
self.size = size  # The number of services
self.num_states = 5  # The number of states in a container's lifecycle
self.num_resources = 3  # The number of resource parameters (RAM, GPU, CPU)
self.min_container = 16
self.max_container = 256

self.timeout = 2  # Set timeout value = 2s
self.container_lifetime = 43200  # Set lifetime of a container = 1/2 day
self.limited_ram = 64  # Set limited amount of RAM (server) = 64GB
self.limited_request = 128
```

# Triển khai các tham số môi trường

HUST

Không gian observation và action của môi trường:

```
self.observation_space = spaces.Dict({
    "execution_times":   spaces.Box(low=1,high=10, shape=(self.size, 1), dtype=np.int16),
    "request_quantity":  spaces.Box(low=0,high=self.limited_request, shape=(self.size, 1),dtype=np.int16),
    "remained_resource": spaces.Box(low=0,high=self.limited_ram, shape=(self.num_resources, 1),dtype=np.int16),
    "container_traffic": spaces.Box(low=0,high=self.max_container, shape=(self.size, self.num_states),dtype=np.int16),
})

self._action_coefficient = spaces.Box(low=0, high=0, shape=(self.size, self.size), dtype=np.int16)
self._action_unit = spaces.Box(low=-1, high=1, shape=(self.size, self.num_states), dtype=np.int16)
self.action_space = spaces.Tuple((self._action_coefficient, self._action_unit))

# Set the main diagonal elements of _action_coefficient to be in the range [0, self.max_container]
np.fill_diagonal(self._action_coefficient.low, 0)
np.fill_diagonal(self._action_coefficient.high, self.max_container)

# Set the last column of _action_unit to be always zero and the sum of the elements in a row of _action_unit = 0
self._get_units()
```

# Triển khai các tham số môi trường

HUST

Một số tham số và ma trận khởi tạo khác:

```
self.current_time = 0  # Start at time 0
self.transition_ram = 0
self.transition_time = 0
self.transition_power = 0
self._custom_request = np.random.randint(0, 64, size=(self.size, 1))
self._pending_request = np.zeros((self.size, 1), dtype=np.int16)
self._ram_required_matrix = np.array([0, 0, 0, 0.9, 2])
self._action_matrix = np.zeros((self.size, self.num_states), dtype=np.int16)
self._exectime_matrix = np.random.randint(2, 16, size=(self.size, 1))
self._request_matrix = np.zeros((self.size, 1), dtype=np.int16)
self._resource_matrix = np.ones((self.num_resources, 1), dtype=np.int16)
self._resource_matrix[0, 0] = self.limited_ram
self._container_matrix = np.hstack((
    np.random.randint(self.min_container, self.max_container, size=(self.size,
1)),
    np.zeros((self.size, self.num_states-1), dtype=np.int16)
))
```

# Nội dung chính

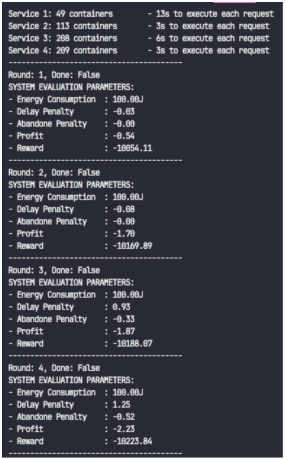
HUST

- Chi tiết mô phỏng
  - Kịch bản mô phỏng
  - Mô hình hoá trình mô phỏng
  - Triển khai các tham số môi trường
- Triển khai mô phỏng và đánh giá
  - Kết quả mô phỏng
  - Đánh giá mô phỏng
- Kết luận

# Nội dung chính

# Kết quả mô phỏng

HUST




Số container và thời gian xử lý mỗi request của từng service

Kết quả của vòng học thứ nhất
Done: False

Kết quả của vòng học thứ hai
Done: False

Kết quả của vòng học thứ ba
Done: False

Kết quả của vòng học thứ tư
Done: False

Mô phỏng dừng lại khi giá trị `Done: True` (`terminated = True`)

# Nội dung chính

HUST

- Chi tiết mô phỏng
  - Kịch bản mô phỏng
  - Mô hình hoá trình mô phỏng
  - Triển khai các tham số môi trường
- Triển khai mô phỏng và đánh giá
  - Kết quả mô phỏng
  - Đánh giá mô phỏng
- Kết luận

# Nhận xét kết quả



- Kết quả reward nhận được bị âm, giảm dần qua các vòng học

  $\longrightarrow$ **Nguyên nhân:** Hệ số $a$ và $b$ trong công thức tính reward đang đặt là 100.

$$Reward = a \times Profit - b \times Energy_{cost} - Penalty_{delay} - Penalty_{abandone}$$

- Đôi lúc mô phỏng chỉ chạy được 1-2 vòng học rồi dừng lại.

  $\longrightarrow$ **Nguyên nhân:** Điều kiện xác định `terminated` chưa chặt chẽ, hoặc do lập trình sai.

- Kết quả mô phỏng chưa đánh giá được vấn đề cho bài toán, vì chưa kết hợp hệ thống mô phỏng request đến mà chỉ mới cho random, hoặc do lập trình sai.

# Vấn đề gặp phải (1)

**Ví dụ:**

$$M_u = \begin{array}{c} \begin{array}{ccccc} \text{N} & \text{L0} & \text{L1} & \text{L2} & \text{A} \\ \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \end{array} \\ \begin{bmatrix} 0 & 1 & -1 & 0 & 0 \\ -1 & 1 & 0 & 0 & 0 \\ -1 & -1 & 1 & 1 & 0 \\ -1 & 0 & 0 & 1 & 0 \end{bmatrix} \end{array} \begin{array}{l} \leftarrow \text{Service 1} \\ \leftarrow \text{Service 2} \\ \leftarrow \text{Service 3} \\ \leftarrow \text{Service 4} \end{array}$$

- Đối với ma trận đơn vị $M_u$, ta có các phần tử $u_{ij} \epsilon \{-1, 0, 1\}$ lần lượt là các action quay state trước đó, không thay đổi state và chuyển đến state kế tiếp.
- Đối với service 1, giả sử ta có $x_1$ container đang ở trạng thái L0 sẽ được chuyển đến state kế tiếp, và $x_1$ container đang ở trạng thái L1 sẽ được chuyển về state trước đó.

# Vấn đề gặp phải (1)



**Ví dụ:**

- Khi đó, khi cập nhật ma trận state $M_{s+1} = M_s + M_a$, giả sử có sẵn 50 container ở L0 và 40 container ở L1, thì số container sau khi cập nhật sẽ thành $(50 + x_1)$ L0 và $(40 - x_1)$ L1.

  $\rightarrow$ Chỉ hiểu được $x_1$ container chuyển từ L1 về L0, còn phần chuyển đến state kế tiếp của L0 thì không thể hiện được.

  $\rightarrow$ Vấn đề xảy ra tương tự với service 2 và service 4.

- Ở service 3, không thể xác định được container chuyển từ state nào đến state nào?

  $\rightarrow$ Việc tính toán các tham số đánh giá nếu chuyển nhiều state trong một `step` rất khó khăn.

$\longrightarrow$ *Đề xuất đặt các điều kiện như kịch bản mô phỏng.*

# Vấn đề gặp phải (2)



- Tính $Penalty_{delay}$ và $Penalty_{abandone}$ chưa đúng? Hiểu sai? (Cần hỏi lại)
- Tính toán năng lượng $Energy_{cost}$ chưa đúng? Lập trình sai?
- Tính toán các tham số chuyển state chưa đúng?

# Nội dung chính

HUST

- Chi tiết mô phỏng
  - Kịch bản mô phỏng
  - Mô hình hoá trình mô phỏng
  - Triển khai các tham số môi trường
- Triển khai mô phỏng và đánh giá
  - Kết quả mô phỏng
  - Đánh giá mô phỏng
- Kết luận

# Kết luận

HUST

- Link Github mô phỏng:
  *https://github.com/owofuyuki/reinforcement-learning-for-serverless*
- Kết quả mô phỏng chưa đánh giá được vấn đề cho bài toán, vì chưa kết hợp hệ thống mô phỏng request đến mà chỉ mới cho random, hoặc do lập trình sai.
- Chỉnh sửa mô phỏng...

# CẢM ƠN MỌI NGƯỜI ĐÃ LẮNG NGHE!