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 26 tháng 2 năm 2024

ONE LOVE. ONE FUTURE.

Nội dung chính

HUST

- Kịch bản mô phỏng
- Triển khai mô phỏng và đánh giá
  - Kết quả mô phỏng
  - Đánh giá mô phỏng
- Kết luận

Nội dung chính

HUST

- Kịch bản mô phỏng
- Triển khai mô phỏng và đánh giá
  - Kết quả mô phỏng
  - Đánh giá mô phỏng
- Kết luận

Điều kiện thay đổi trạng thái

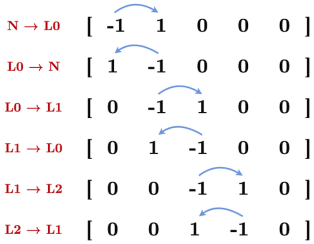
- Giới hạn sự thay đổi state chỉ giữa hai state liền kề, state chỉ thay đổi một mức mỗi lần.
- Biểu diễn ma trận đơn vị M_u theo cách hiểu khác (nói thêm ở vấn đề gặp phải):
  - -1: state nguồn, tương ứng với state bị thay đổi
  - 1 : state đích, tương ứng với state thay đổi tới
  - 0 : state không bị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

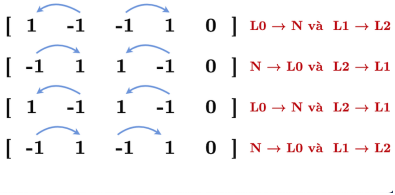
  $\longrightarrow$ Hai giá trị -1 và 1 luôn liền kề nhau (nếu có thay đổi ở hai vị trí state thì -1 và 1 sẽ được xếp theo từng cặp).

Điều kiện thay đổi trạng thái

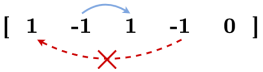
HUST

Trường hợp thoả mãn





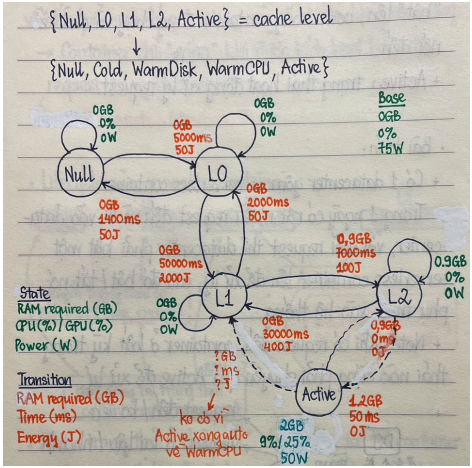
Trường hợp không thoả mãn



Điều kiện dừng

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

  → Container được đưa về state nào không sử dụng tài nguyên tương ứng gần nhất.



Kịch bản mô phỏng [Chỉnh sửa]

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 đến tuân theo phân phối Poisson với $\lambda = 100$ (100 request/h; 1h = 3600s).
- Mỗi 10 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.

Kịch bản mô phỏng [Chỉnh sửa]

HUST

- Công thức tính reward tối giản:

$$Reward = n_{request-in-L2} - \alpha \times t_{transition} - \beta \times Energy_{cost}$$

- Điều kiện:
  - $\alpha = 0.4$, $\beta = 0.02$
  - $n_{request-in-L2} \leq n_{total-container}$
  - Nếu $t_{transition} < t_{timeout}$ thì request được xử lý, nếu $t_{transition} \geq t_{timeout}$ thì request lỗi.
- Mỗi 10 giây, thực hiện thêm việc kiểm tra các container có đang chuyển trạng thái hay không (dùng ma trận `_transition_matrix`) và thực hiện đưa ra action phù hợp:
  - Nếu container đang chuyển trạng thái thì bỏ qua việc đưa ra action cho container này.
  - Nếu container không chuyển trạng thái (chuyển trạng thái đã xong) thì đưa ra action.

Nội dung chính

HUST

- Kịch bản mô phỏng
- Triển khai mô phỏng và đánh giá
  - Kết quả mô phỏng
  - Đánh giá mô phỏng
- Kết luận

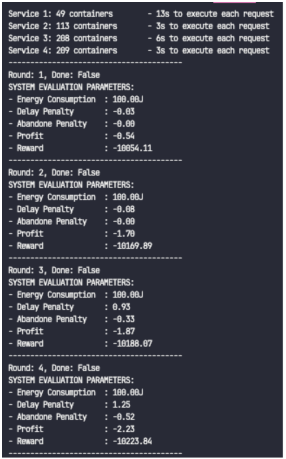
Nội dung chính

HUST

- Kịch bản mô phỏng
- Triển khai mô phỏng và đánh giá
  - Kết quả mô phỏng
  - Đánh giá mô phỏng
- Kết luận

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

- Kịch bản mô phỏng
- Triển khai mô phỏng và đánh giá
  - Kết quả mô phỏng
  - Đánh giá mô phỏng
- Kết luận

Nhận xét kết quả

- 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.

- Mô phỏng chạy bị lỗi, thỉnh thoảng mới chạy được =)))

  $\longrightarrow$ **Nguyên nhân:** Một số hàm và điều kiện được định nghĩa chưa chặt chẽ, sửa thêm.

Vấn đề gặp phải (1)

- Đang tìm hiểu cách tính reward theo cập nhật mềm, chưa áp dụng được trong mô phỏng.

  Cụ thể, mạng neural Q sẽ có một số tham số W và B môt tả các lớp trong mạng neural, khi huấn luyện mạng neural Q_{new} tương ứng có W_{new} và B_{new}. Với cách thực hiện cập nhật mềm, khi cập nhật $Q = Q_{new}$, thay vì đặt trực tiếp các tham số $W = W_{new}$ và $B = B_{new}$, ta sẽ đặt $W = \alpha W_{new} + (1 - \alpha)W$ và tương ứng đối với B, trong đó α là một siêu tham số kiểm soát mức độ di chuyển W, B đến W_{new}, B_{new}

  $\longrightarrow$ α được đặt rất nhỏ, khoảng 0.01, để thực hiện thay đổi dần dần đối với các tham số của mạng neural và làm thuật toán RL hội tụ đáng tin cậy hơn.

- Luồng thời gian xử lý có thể bị sai (chưa rõ).

Vấn đề gặp phải (2)

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

Nội dung chính

HUST

- Kịch bản mô phỏng
- Triển khai mô phỏng và đánh giá
  - Kết quả mô phỏng
  - Đánh giá mô phỏng
- Kết luận

Kết luận

- 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.
- Chỉnh sửa mô phỏng...

CẢM ƠN MỌI NGƯỜI ĐÃ LẮNG NGHE!