Julius-Maximilians-
UNIVERSITÄT
WÜRZBURG

Institute of Computer Science
Chair of Communication Networks
Prof. Dr. Tobias Hoßfeld

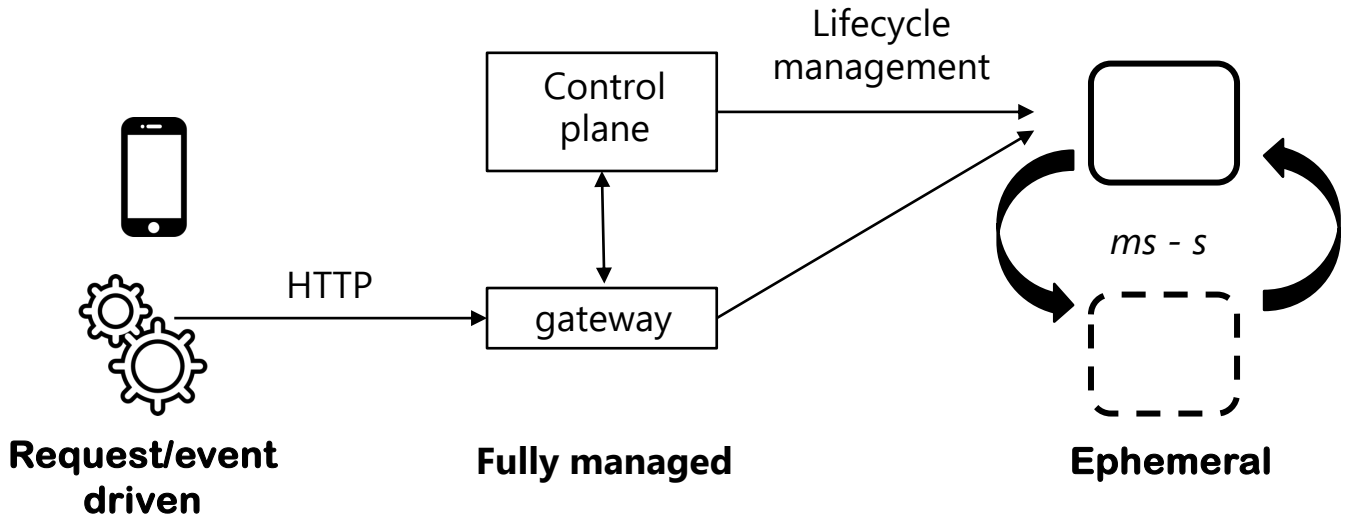
QoS-aware and sustainable deployment of Serverless

Kien Nguyen

informatik.uni-wuerzburg.de/comnet

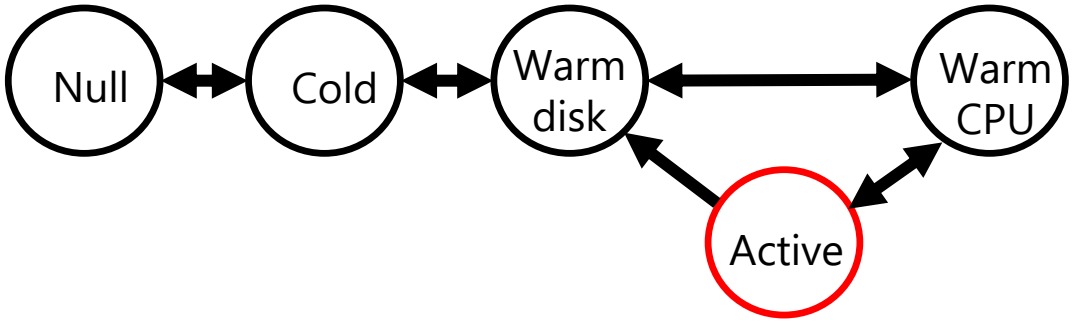
Introduction

- Serverless is a **use case** of container that is



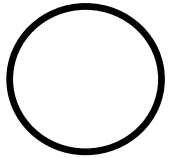
- Research question skip
  - What is the **performance** and **resource demand** of serverless in **different** environments?
  - How to optimize **performance** and **resource demand** of serverless?

Serverless container lifecycle



State	Description
Null	Idle, nothing is related to the service
Cold	The abstraction of service has been deployed
Warm disk	Service and Image are available
Warm CPU	Container is available, code in container is running and waiting for incoming connection
Warm Mem	Container is available but in “paused”/sleep state
Active	Code is processing the request

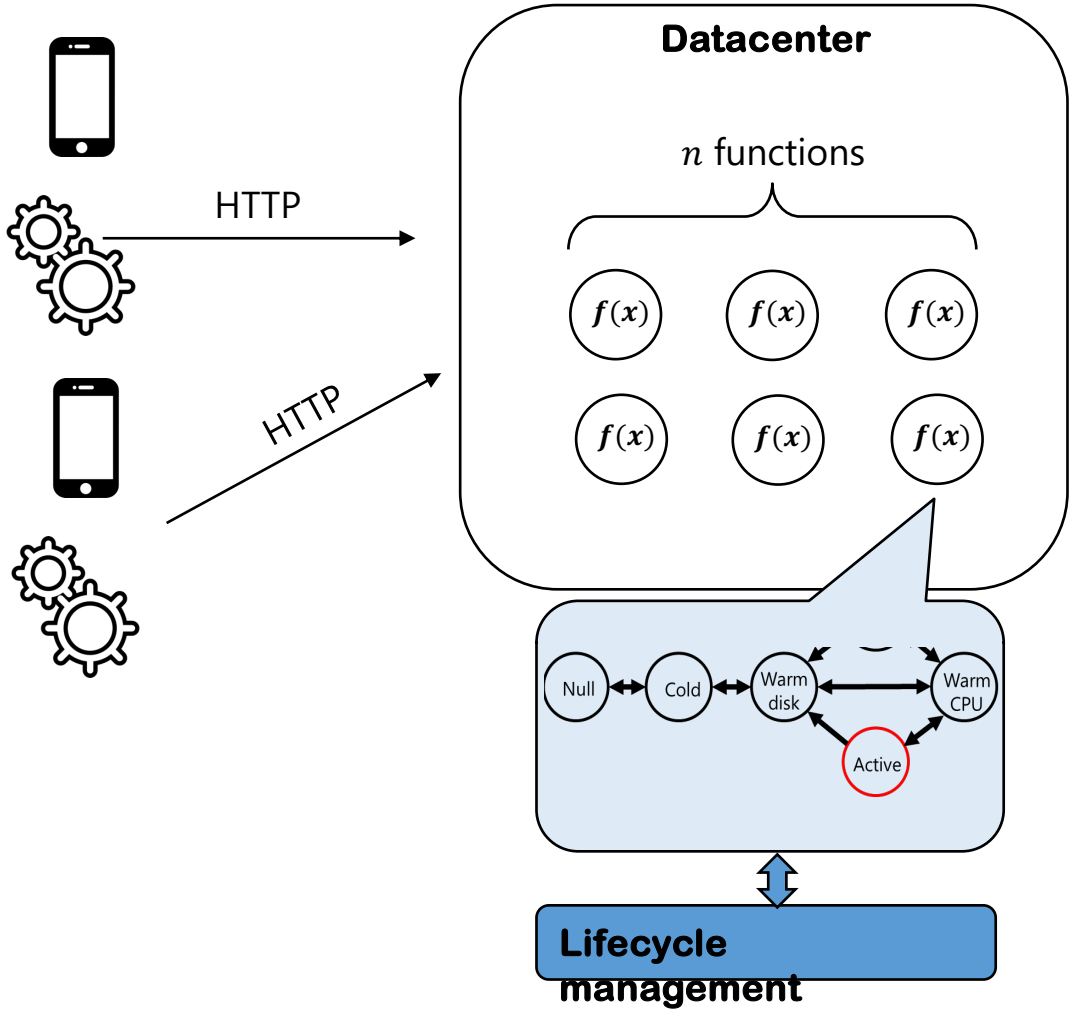
- Moving from one state to another costs:
  - Time
  - Energy



- Staying at one state costs:
  - Resource (RAM)

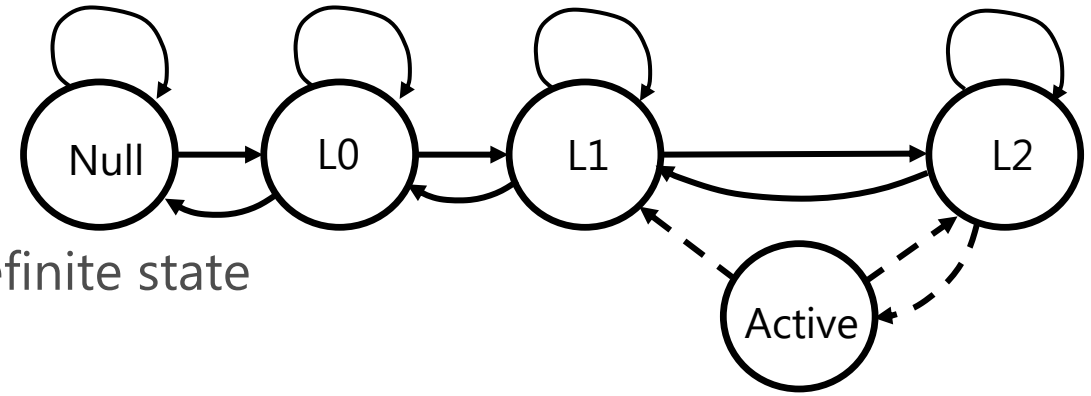
Problem and Methodology

- Scenario
  - DC with n functions
  - Functions are serverless flexible lifecycle
- Challenges
  - ~~How to minimize latency caused by moving from intermediate states to Active?~~
  - ~~How to minimize energy consumption caused by moving back and forth between states?~~
  - How to maximize revenue for operators?
- Methodology
  - Applying RL to prepare serverless functions in advance



Simplified state machine

- Simplified state-machine
  - L0, L1, L2 indicate cache level
  - Active may mean nothing because it's not an indefinite state



- Profiling results
  - Costs of staying at one state and move to another states

Transition	RAM required (GB)	Time (ms)	Energy (J)
Null -> L0	0	5000	50
L0 -> L1	0	50000	2000
L1 -> L2	0.9	7000	100
L2 -> Act	1.2	50	0
Act -> L2	0.9	0	0
L2-> L1	0	30000	400
L1 -> L0	0	2000	50
L0 -> Null	0	1400	50

State	RAM required (GB)	CPU	Power (W)
Base	0	0	75
Null	0	0	0
L0	0	0	0
L1	0	0	0
L2	0.9	0	0
Act	1.2		50

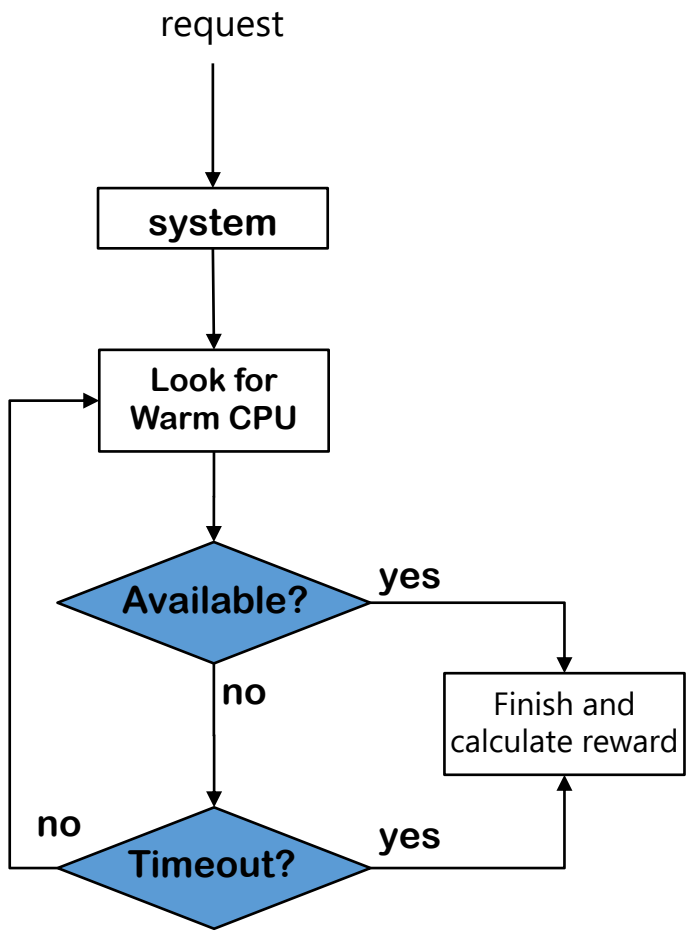
Traffic model

- System has N types of service*, I functions per service.
  - Each instance i can serve only one request of type n
  - Total number of instances is: $N * I$
  - *We must have multiple services, otherwise the system will keep everything at Warm CPU

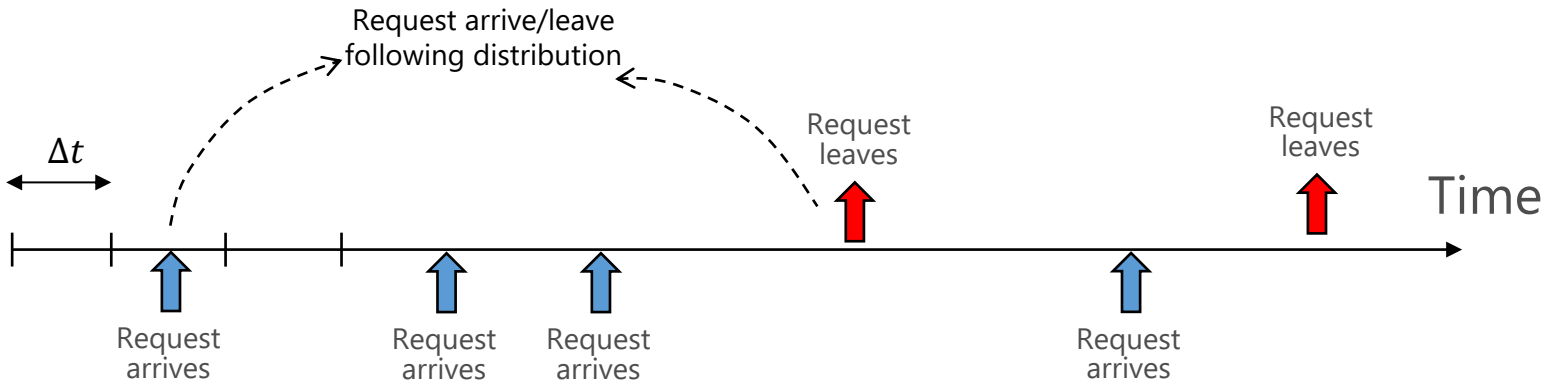
- Online problem
  - Arrival rate following Poisson distribution, for ex: $\lambda = 100 \sim 100 req/hour$
  - One request of type n arrives, one instance i of n must be turned to Active
  - Assume the each request of type N requires $t_{processing}$ seconds

Workflow

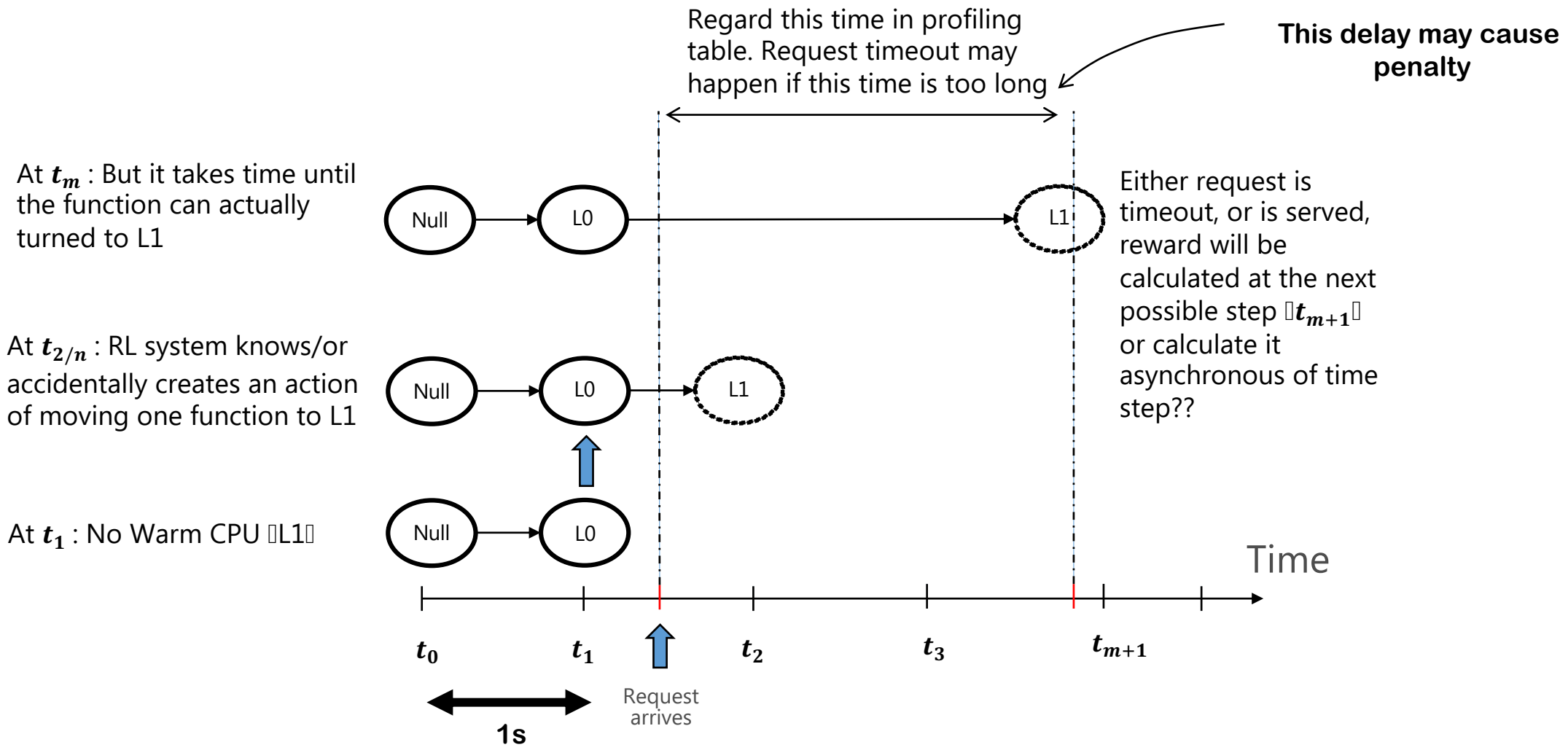
- The number of instances is fixed and equals $N * I$.
- DRL makes decision every Δt
  - Δt must be > 1min, because longest action is 50s
- When request arrives, the following operates:
  - System looks for Warm CPU functions to host the request
  - Request can wait for a fixed time_out before marked as failure
  - Reward will be calculated, and model will be updated



Ideally: during this timeout, RL must realize it needs to turn on a function to Warm CPU



Workflow: example of unavailable Warm CPU



Power & Energy consumption model

- Power is the sum of power consumption of all servers on the Cloud, example of power cons. of one server is: P = $P_{idle} + \sum P\ of\ services$
  - P_{idle} is the constant power of the server at idle state
  - $\sum P\ of\ services$ is power cons. of services running on top of the server

- Now we have M servers and N services that **are Wcpu/Active**, so how many servers are running?
  - $\sum S_{running} = (\sum_1^M RAM)/(\sum_1^N RAM\ or\ CPU\ demand\ in\ Active/Wcpu)$
  - Only running servers are calculated for power consumption

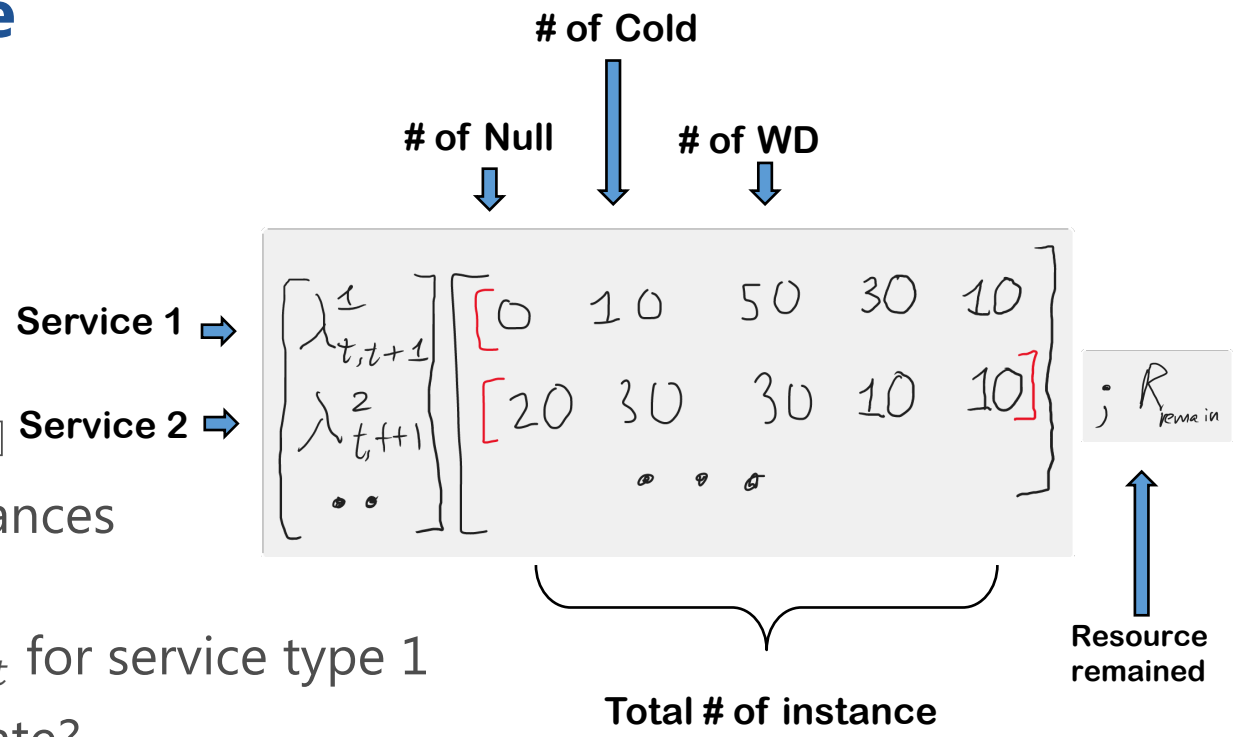
- assume n are Active, m are in-active, u are moving from one state to another. Then Energy cost during Δ_t is:
  - $E(\Delta_t) = [\sum_1^{S_{run}} P_{idle} + \sum_1^n P_{active}] * \Delta_t + \sum_1^u E\ of\ processes$*
  - *this energy is calculated only when processes finish within Δ_t
  - m in-actives are assumed to not cosume any energy they do consume resources as shown in next slide though

Resource consumption model

- A system has M different servers, but we sum their resource in a pool: ex. RAM capacity of Cloud is 1000GB, same for CPU
- Assume that n functions are in Warm CPU, m in Active
- RAM consumption: $\sum RAM = n * RAM_{wc} + m * RAM_{active}$
- CPU consumption: $\sum CPU = m * CPU_{active}$
- If either RAM or CPU are full (> 100% capacity) then no more functions can be turned to Warm CPU or Active

System model: representation of state

- State M_s
  - Represented as $m * n$ matrix
  - Row represents type of service
  - Column represents number of instances that are staying at a certain Mode （Null, Cold, etc.）
  - Sum of all element in a row is number of instances that belong to a certain Service
  - $\lambda^1_{t,t+1}$ is the sum of arrival requests between Δ_t for service type 1
  - Should we put leaving requests also in the State?



  - Add Active state and lambda of previous time window
    - Embed it inside the State Matrix

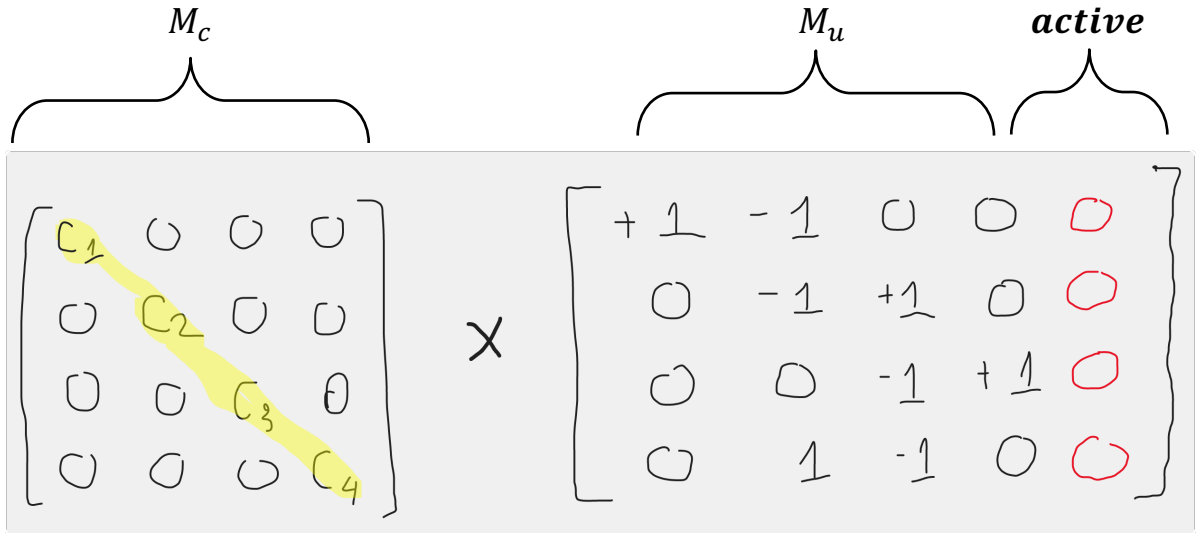
System model: representation of action

- Action M_a
  - Is the multiplication of two matrices
    - Coefficient, diagonal matrix M_c
    - Unit matrix M_u
- Let $I = \{1, \dots, m\}, J = \{1, \dots, n\}$, e_{ij} is an element in the matrix
- Constraints for M_a
  - Sum of each row in M_u must equal 0

$$\sum_{j=1}^{n} e_{mj} = 0, \quad \forall m \in I$$

  - Absolute value of $e_{ij} \in M_u$ must equal or lower than 1

$$|e_{ij}| \leq 1, \quad \forall e_{ij} \in M_u$$



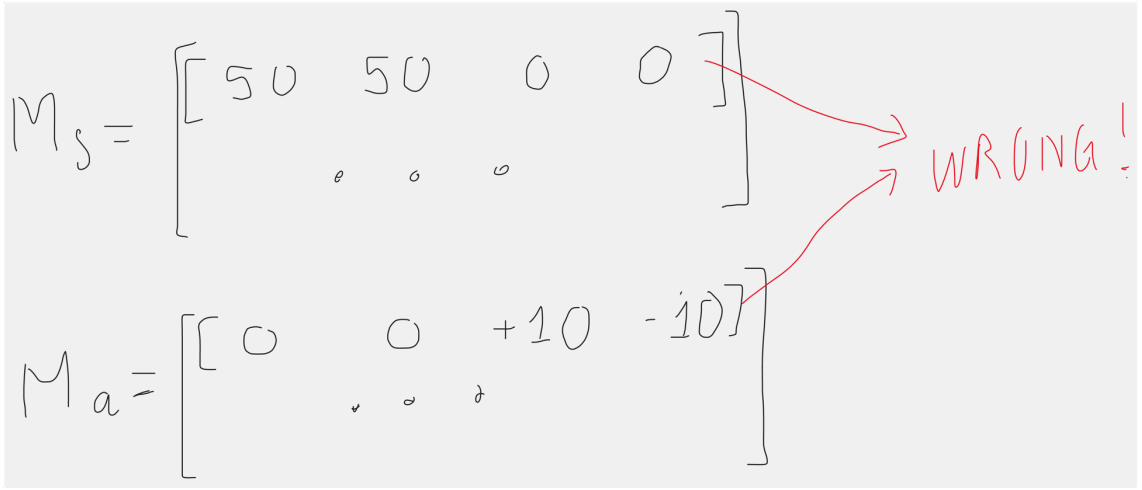
System model: representation of action

- continue Constraints for M_a
  - To assure no negative value of next State

$$M_{s+1} = M_s + M_a$$

$$e_{ij} \geq 0 \quad ; \forall e_{ij} \in M_{s+1}$$

  - Otherwise, this will happen



System model: representation of action

- continue Constraints for M_a
  - Denote C_{cap} as the maximum RAM of the system
  - To assure resource consumption will not exceed the Capacity C_{cap}, the column that represents Warm CPU only mode that consumes RAM must not exceed C_{cap}. Let $M_r = M_{mx2}$as the matrix of the column of Warm CPU and Active mode

$$\sum_{i=1}^{I} \left[M_r(M_a) + M_r(M_s) \right]_{i,1} \leq C_{cap}$$

  - Last column of M_u must equals zero. Let $M_{ra} = M_{mx1}$as the matrix of the column of Active mode

$$e_{ij} = 0; \forall e_{ij} \in M_{ra}$$

Reward model

- Accepted request λ_{acc} leads to profit:

$$Profit = \lambda_{acc} \times RAMused \times \mathrm{Exec_{time}} \times \$perUnit \ (1)$$

$$Reward = Profit - Energy_{cost} - Penalty_{cost}^{delay} - Penalty_{cost}^{abandone}$$

- $Profit$ follows AWS equation.
  - $RAMused$ is GB, 500MB = 0.5
  - $\$perUnit = 0.0000166667\$$
  - $\mathrm{Exec_{time}}$ is fixed and can serve as the coefficient value, ex. 2 means 2s
- $Energy_{cost}$ may follow state energy price 30 cents/kWh
- $Penalty_{cost}^{abandone}$ is - $Profit$, abandonment rate follows: $5.8\% \times t \ (t > 2)$ □2□ [1]
- $Penalty_{cost}^{delay}$ is regarded to "non-Active" states, regardless of execution time
  - $Penalty_{cost}^{delay} = \alpha + \beta \times t$ [2] , α is used to make sure $Penalty_{cost}^{delay}$ always greater than 0, t is counted when counter > 2s. Counter starts when a request arrives.
  - $\alpha = 5\% \ of \ Profit, \beta = 0.34 \Rightarrow maxPenalty_{cost}^{delay} = 10\% \ of \ profit$ [3] since □2□ causes $\max t_{abadone} \sim 15s$

[1] Video Stream Quality Impacts Viewer Behavior: Inferring Causality Using Quasi-Experimental Designs
[2] SLA-Based Resource Provisioning for Hosted Software-as-a-Service Applications in Cloud Computing Environments
[3] Performance-based service-level agreement in cloud computing to optimise penalties and revenue

How to code now?

- Focus on the non-RL environmental variables first! Hiep's simulator
  - Request arrival: Possion follows traffic
  - Request live time: Exponential ~ 2s

- Then go to modeling equations
  - Energy, resource and reward functions

- Then go to RL variables
  - States, actions, rewards, etc.