Proactive Microservice Placement and Migration for Mobile Edge Computing

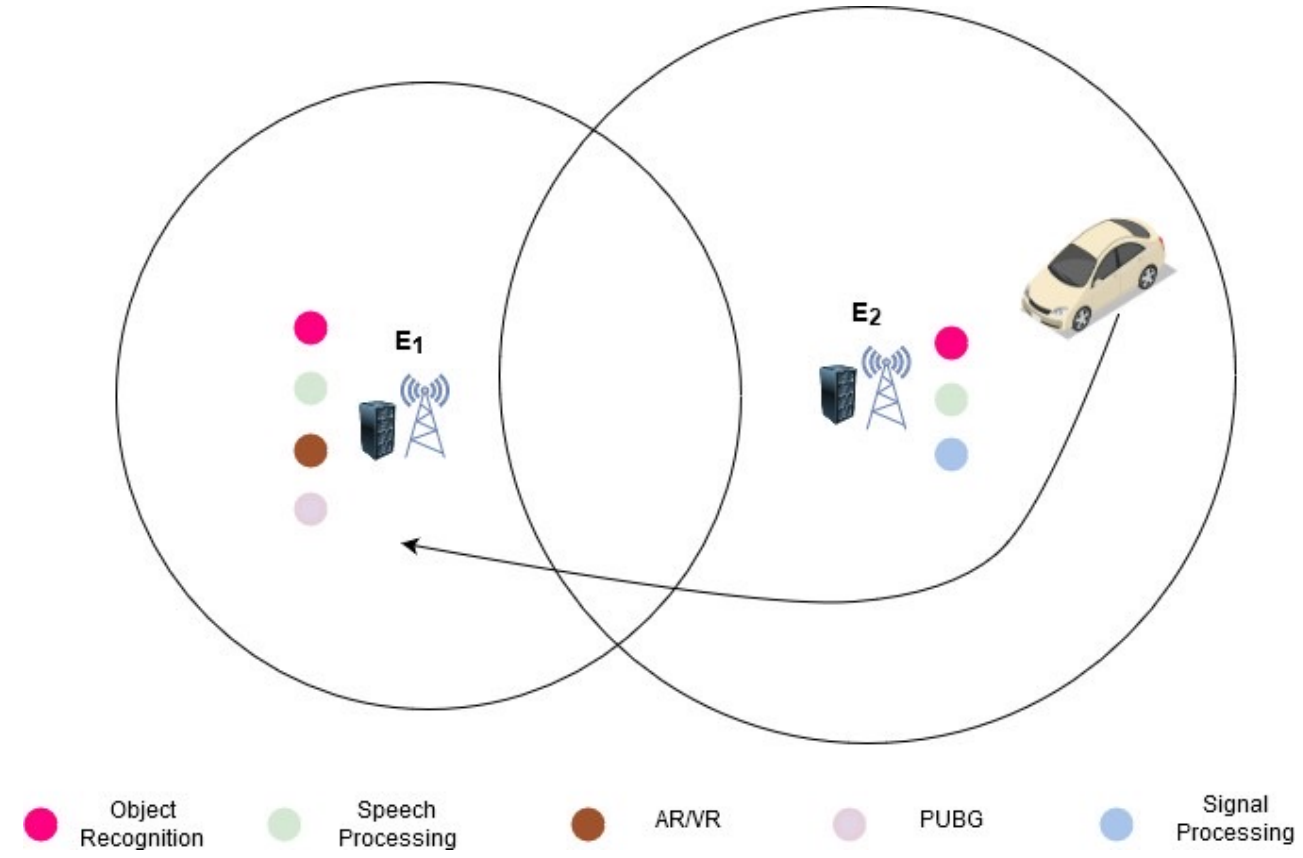
KAUSTABHA RAY & ANSUMAN BANERJEE
ADVANCED COMPUTING AND MICROELECTRONICS UNIT
INDIAN STATISTICAL INSTITUTE, KOLKATA

NANJANGUD C. NARENDRA
ERICSSON RESEARCH
BANGALORE

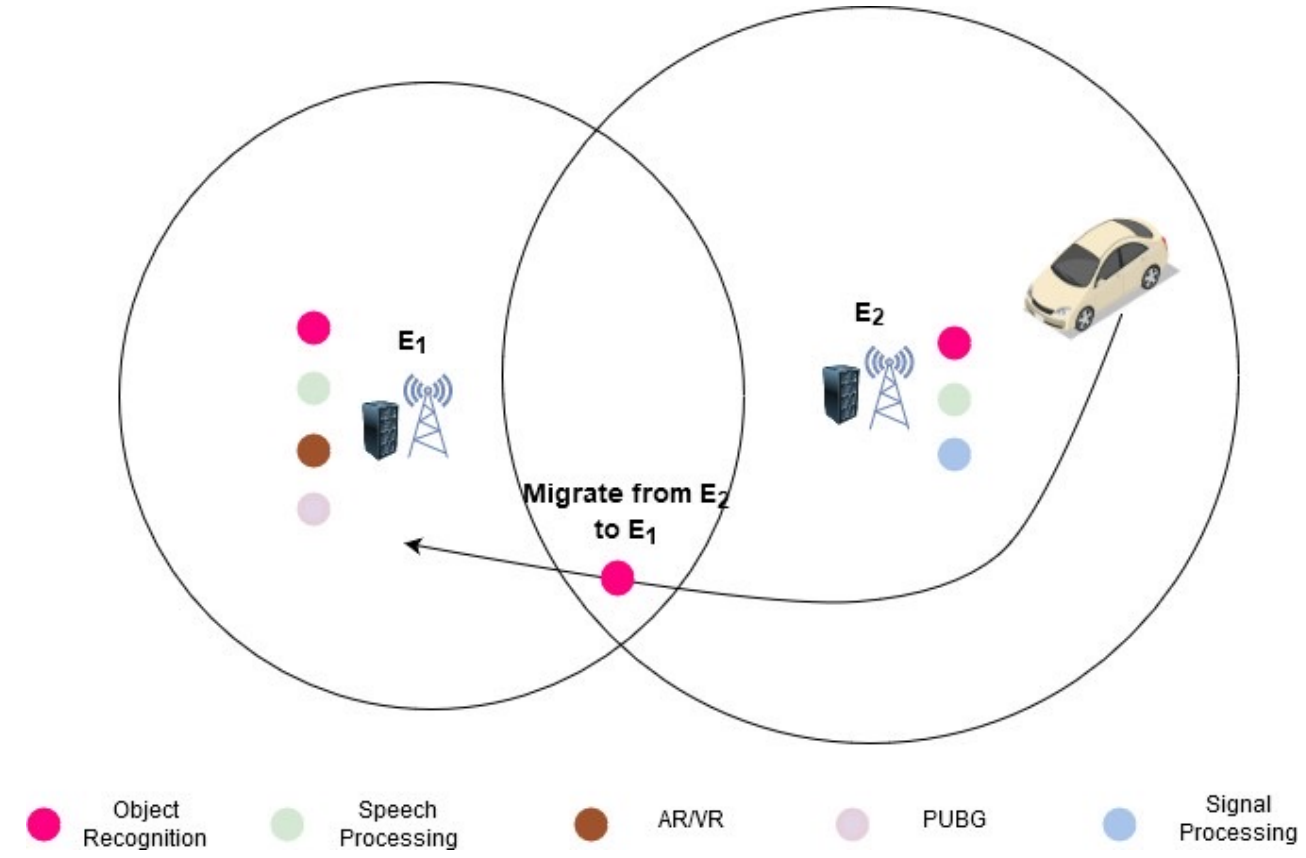
Outline

- Microservice Placement and Migration Problem
- Design of Proactive Microservice Placement and Migration Policy
- Experimental Evaluation

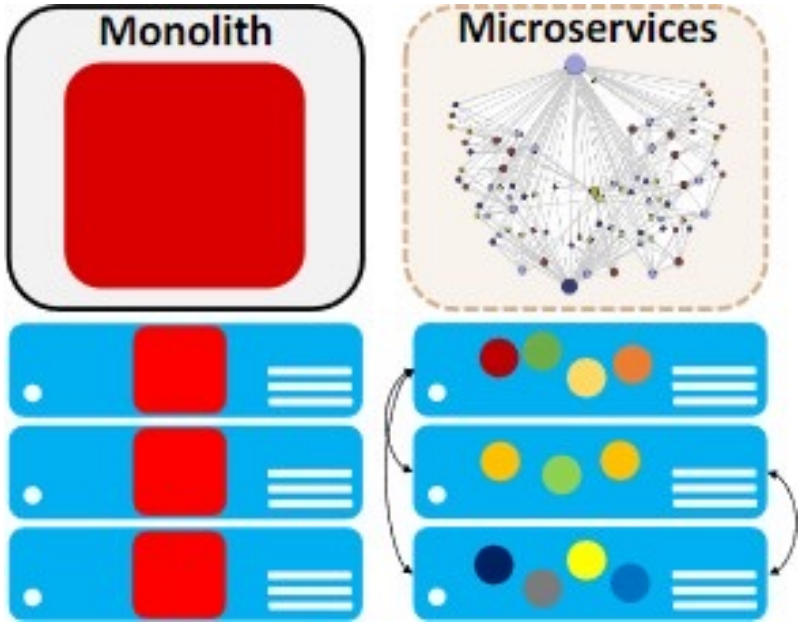
Service Placement and Migration

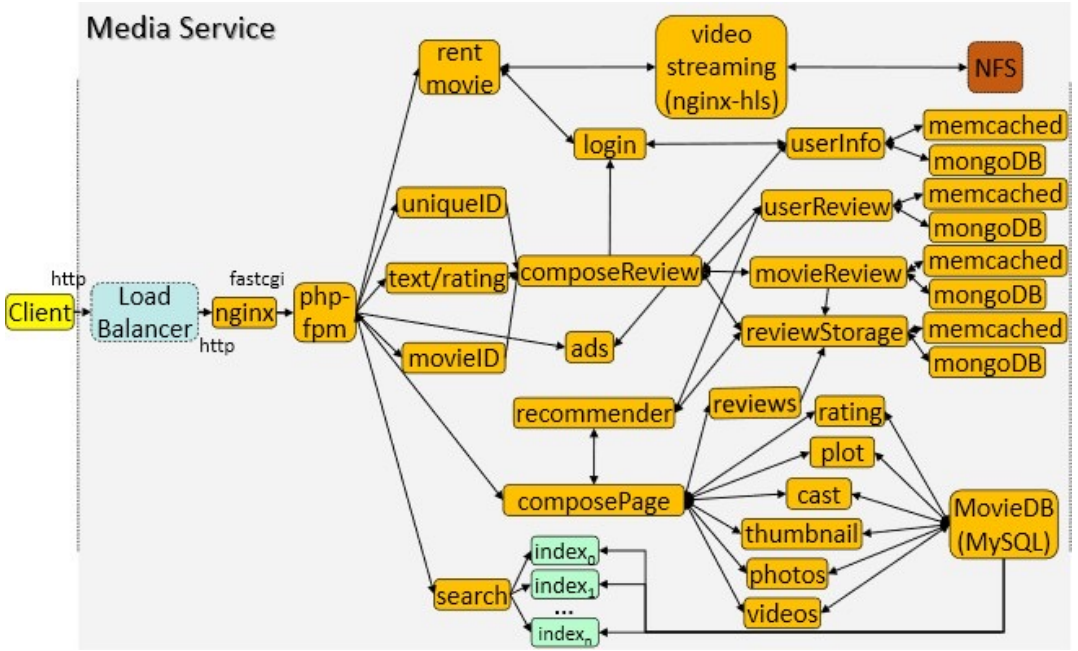


Service Placement and Migration



From Monoliths to Microservices



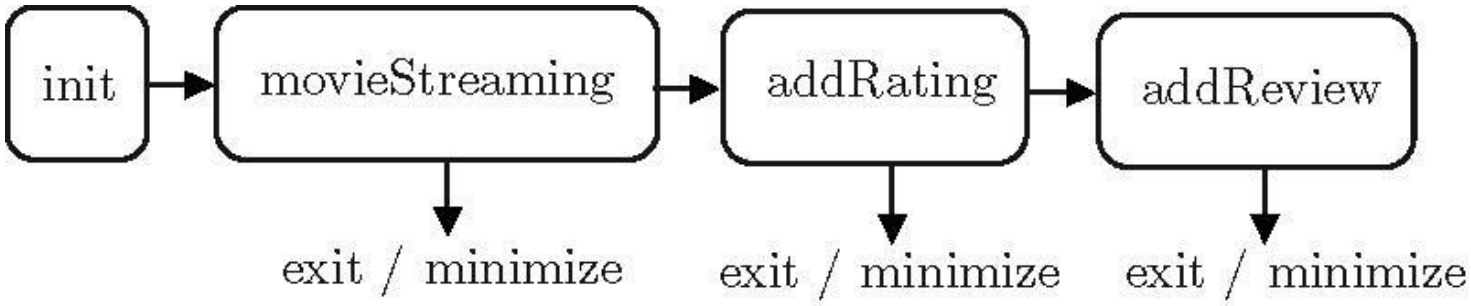


GAN, YU, ET AL. "AN OPEN-SOURCE BENCHMARK SUITE FOR MICROSERVICES AND THEIR HARDWARE-SOFTWARE IMPLICATIONS FOR CLOUD & EDGE SYSTEMS." *ASPLOS*. 2019

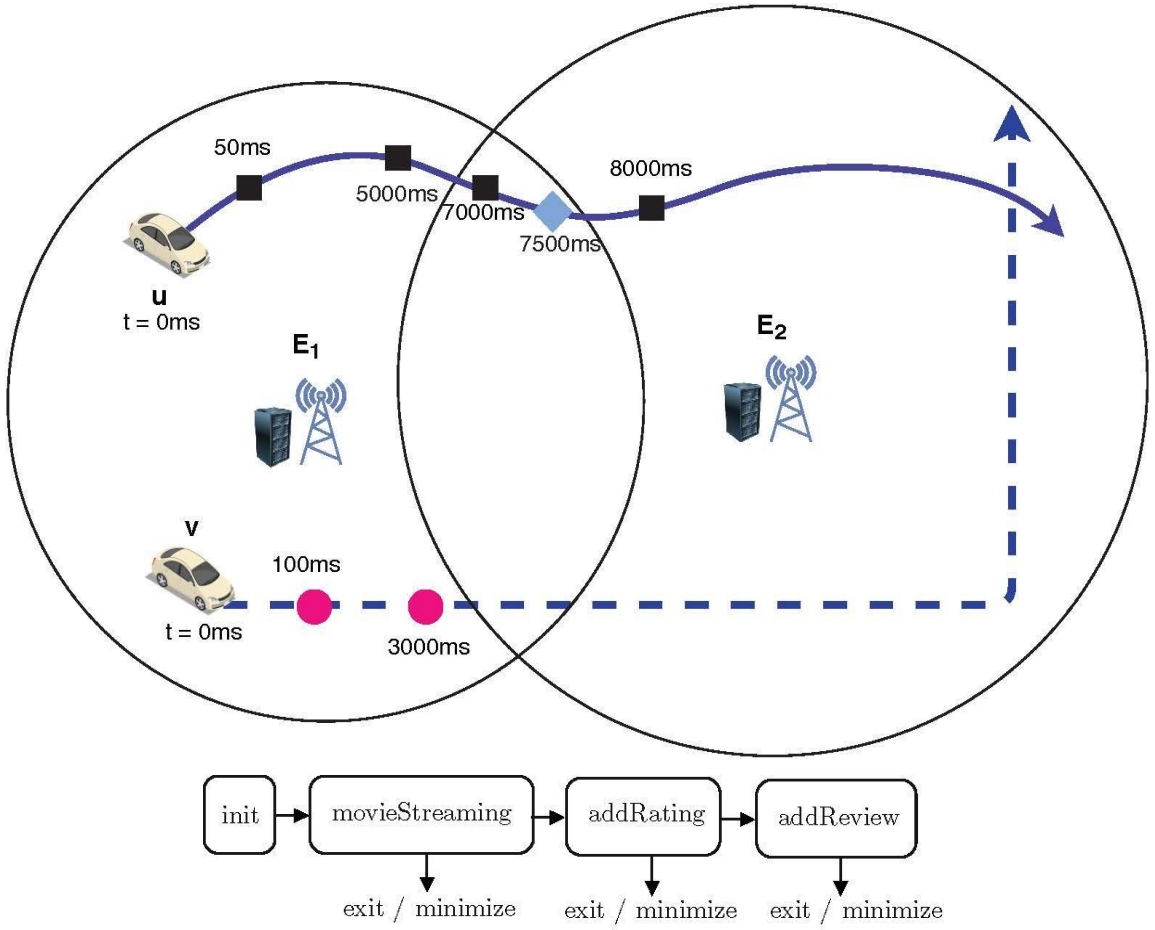
Novelty of this paper

- Considers the paradigm shift from monolithic services to microservices
- Formally model microservice placement and migration using Markov Decision Process (MDP)
- Presents a reinforcement learning based proactive microservice placement and migration strategy

A Motivating Example



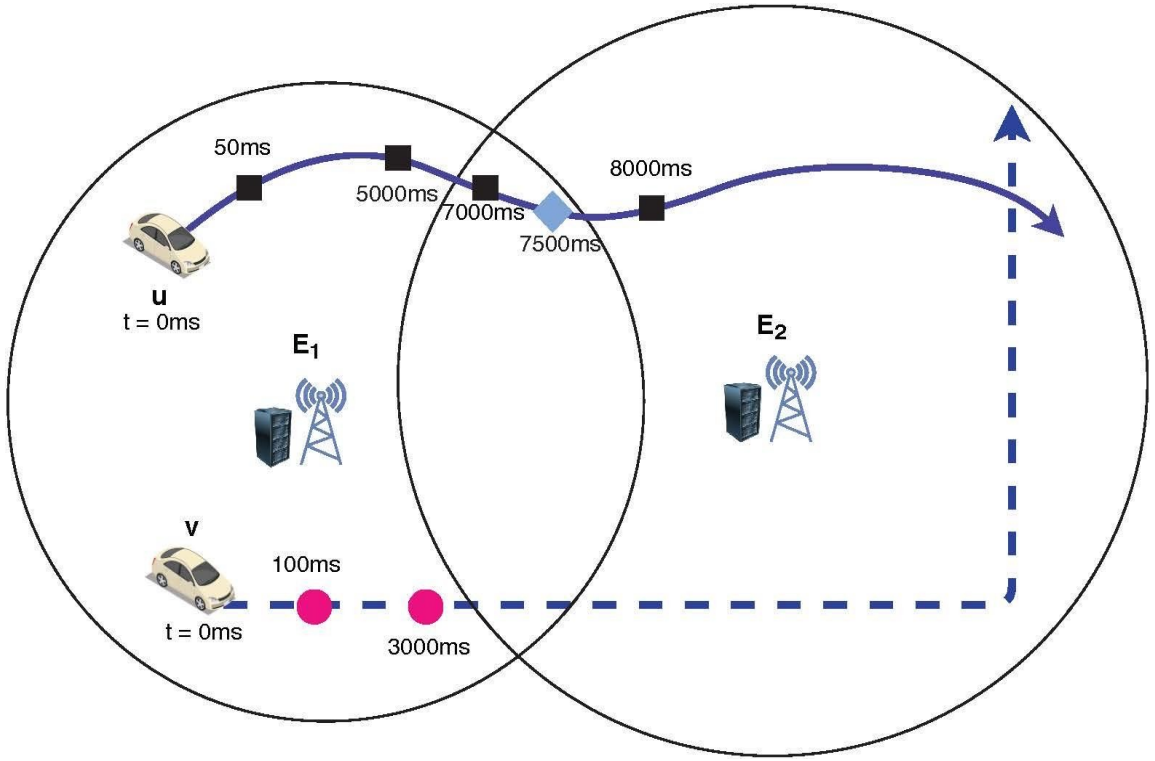
A Motivating Example

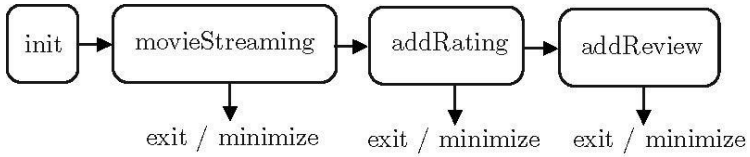


Time t	User Action	Server-Service State
$0ms$		No Services Deployed
$50ms$	$u \rightarrow$ movieStreaming	initialize movieStreaming
$75ms$		$E_1 \rightarrow$ movieStreaming
$100ms$	$v \rightarrow$ movieStreaming	$E_1 \rightarrow$ movieStreaming initialize new task for v
$110ms$		$E_1 \rightarrow$ movieStreaming, v_{task}
$3000ms$	v exits movieStreaming	$E_1 \rightarrow$ movieStreaming
$5000ms$	$u \rightarrow$ addRating	initialize addRating
$5025ms$		$E_1 \rightarrow$ addRating
$7000ms$	u minimizes addRating	$E_1 \rightarrow$ addRating
$8000ms$	$u \rightarrow$ addReview	initialize addReview
$8025ms$		$E_2 \rightarrow$ addReview

On-Demand Placement

A Motivating Example

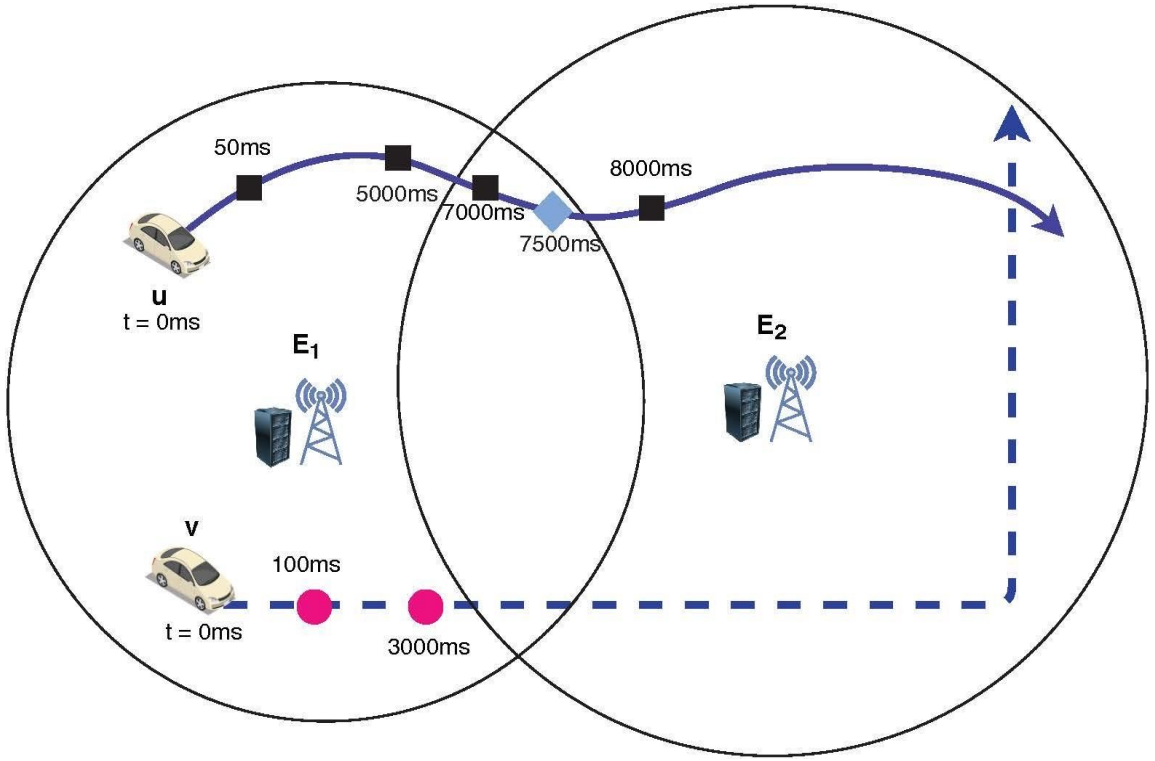


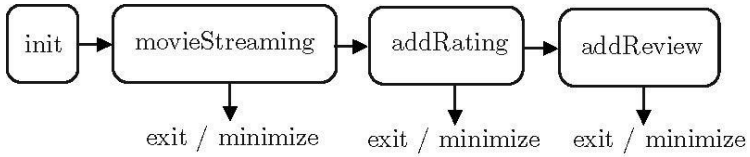


Time t	User Action	Server-Service State
$0s$		No Services Deployed
$50ms$	$u \rightarrow$ movieStreaming	initialize movieStreaming
$75ms$		$E_1 \rightarrow$ movieStreaming
$100ms$		$E_1 \rightarrow$ movieStreaming, addRating
$100ms$	$v \rightarrow$ movieStreaming	$E_1 \rightarrow$ movieStreaming, addRating initialize new task for v
$110ms$		$E_1 \rightarrow$ movieStreaming, addRating, addReview, v_{task}
$135ms$		$E_1 \rightarrow$ movieStreaming, addRating, addReview, v_{task}
$3000ms$	v exits movieStreaming	$E_1 \rightarrow$ movieStreaming, addRating, addReview
$5000ms$	$u \rightarrow$ addRating	$E_1 \rightarrow$ addRating, addReview
$7000ms$	u minimizes addRating	$E_1 \rightarrow$ addRating, addReview
$8000ms$	$u \rightarrow$ addReview	state-aware migrate addReview
$8010ms$		$E_2 \rightarrow$ addReview

Proactive Placement

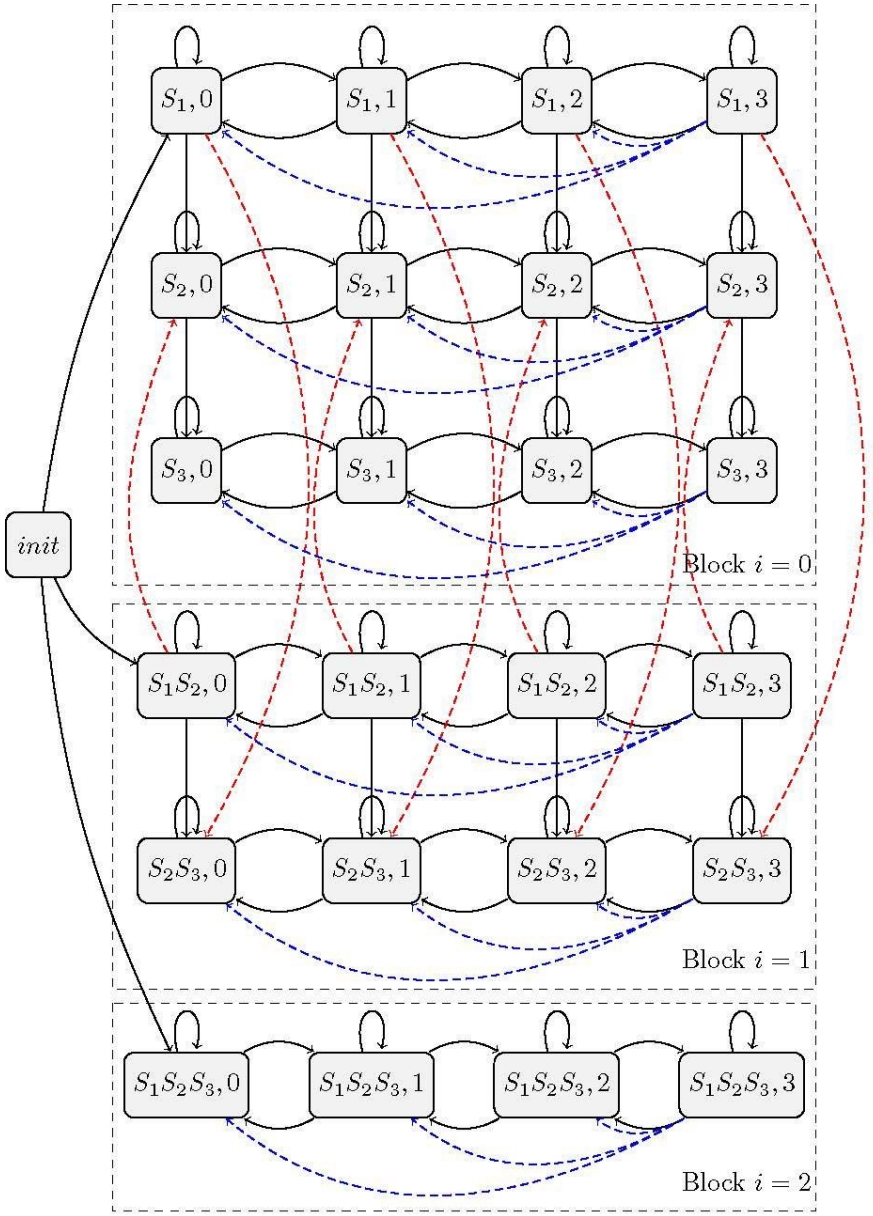
A Motivating Example





Time t	User Action	Server-Service State
$0s$		No Services Deployed
$50ms$	$u \rightarrow$ movieStreaming	initialize movieStreaming
$75ms$		$E_1 \rightarrow$ movieStreaming
$100ms$		$E_1 \rightarrow$ movieStreaming, addRating
$100ms$	$v \rightarrow$ movieStreaming	$E_1 \rightarrow$ movieStreaming, addRating initialize new task for v
$110ms$		$E_1 \rightarrow$ movieStreaming, addRating, addReview, v_{task}
$135ms$		$E_1 \rightarrow$ movieStreaming, addRating, addReview, v_{task}
$3000ms$	v exits movieStreaming	$E_1 \rightarrow$ movieStreaming, addRating, addReview
$5000ms$	$u \rightarrow$ addRating	$E_1 \rightarrow$ addRating, addReview
$7500ms$	$u \rightarrow$ addRating	migrate addRating, addReview
$7555ms$	$u \rightarrow$ addRating	$E_2 \rightarrow$ addRating, addReview
$8000ms$	$u \rightarrow$ addReview	$E_2 \rightarrow$ addReview

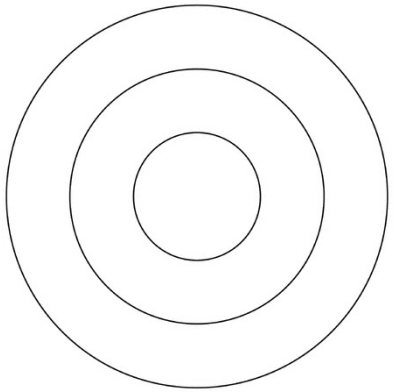
Proactive Placement + Migration



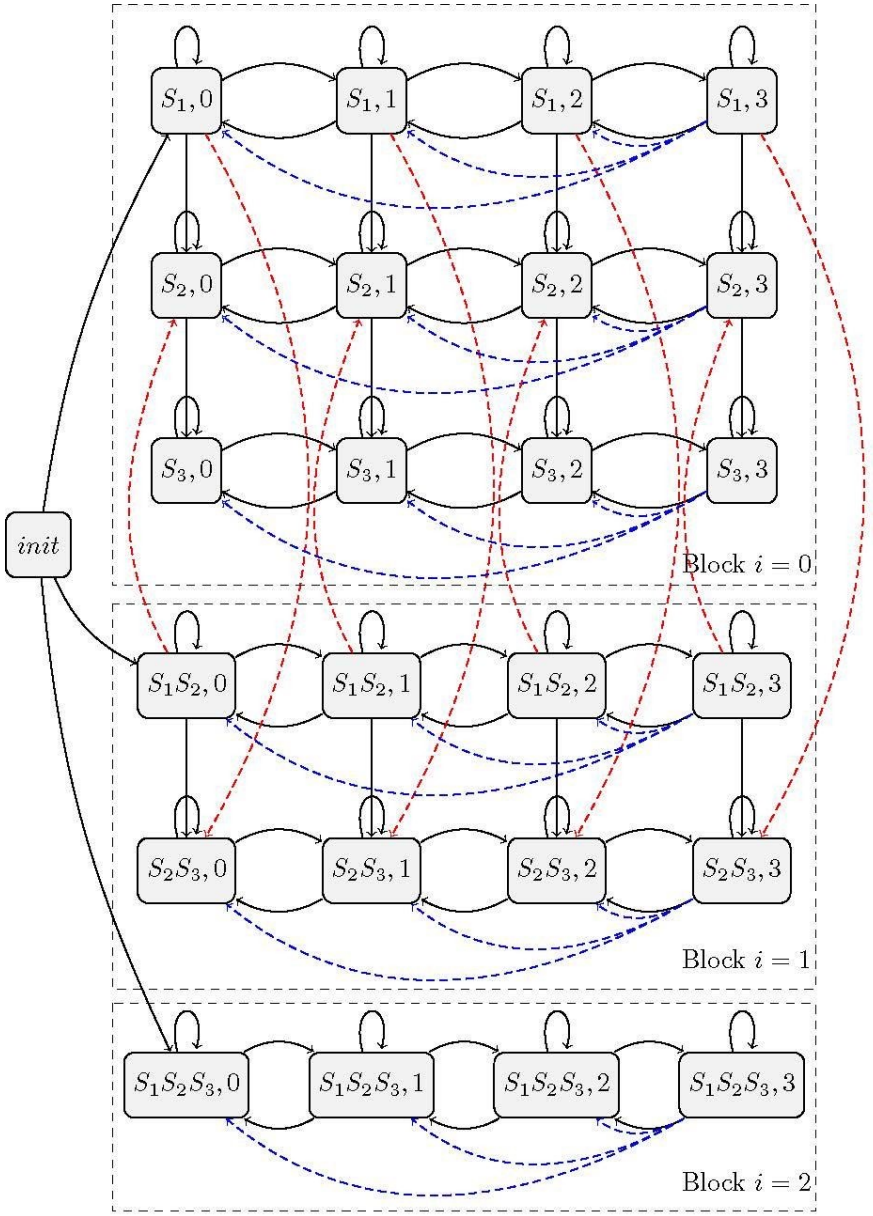
Formal Model

States Represent Proactive Placement of Microservices

Transitions Represent Movement and Choices of Proactive Placement



How distance is mapped

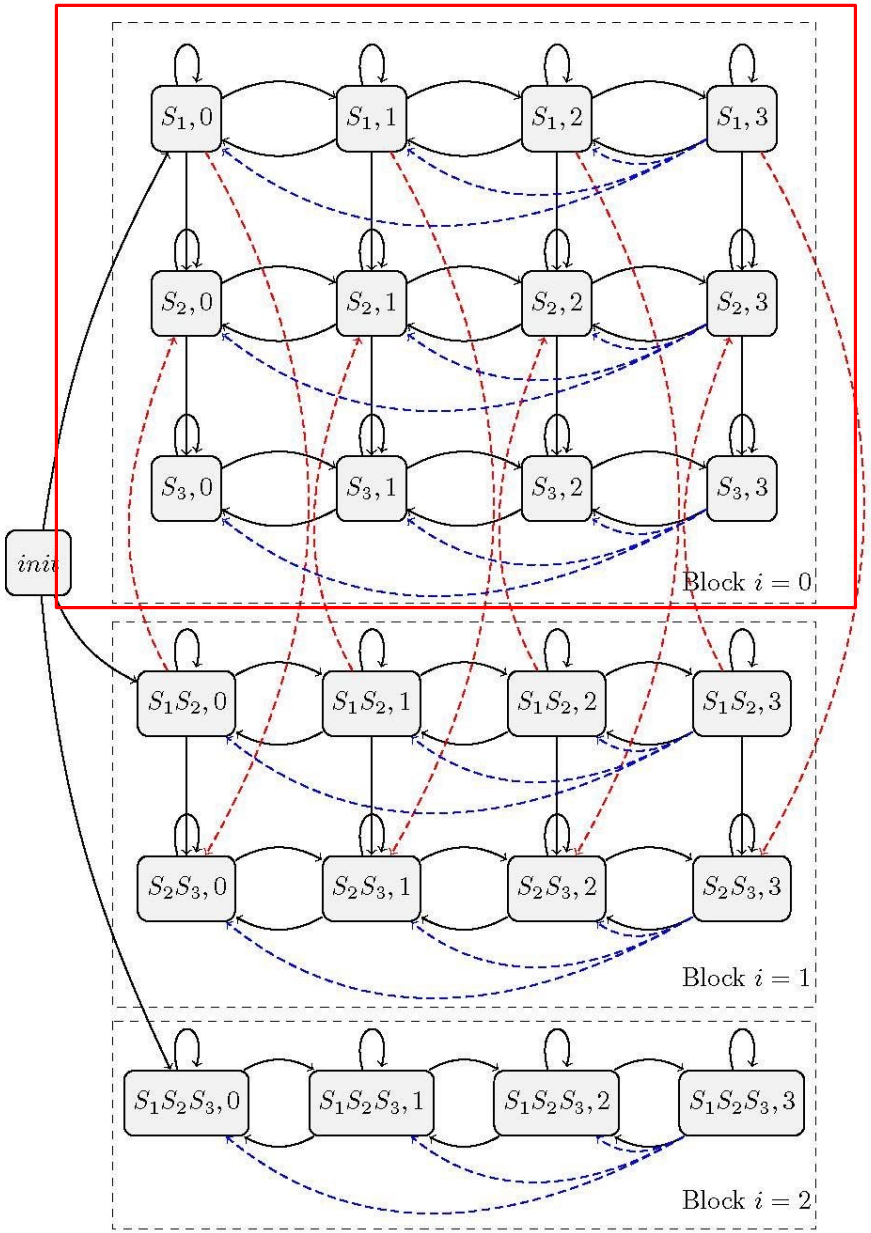


Formal Model

Blocks Represent *i* number of microservices to proactively deploy

Blocks i=0 to *i=n-1*

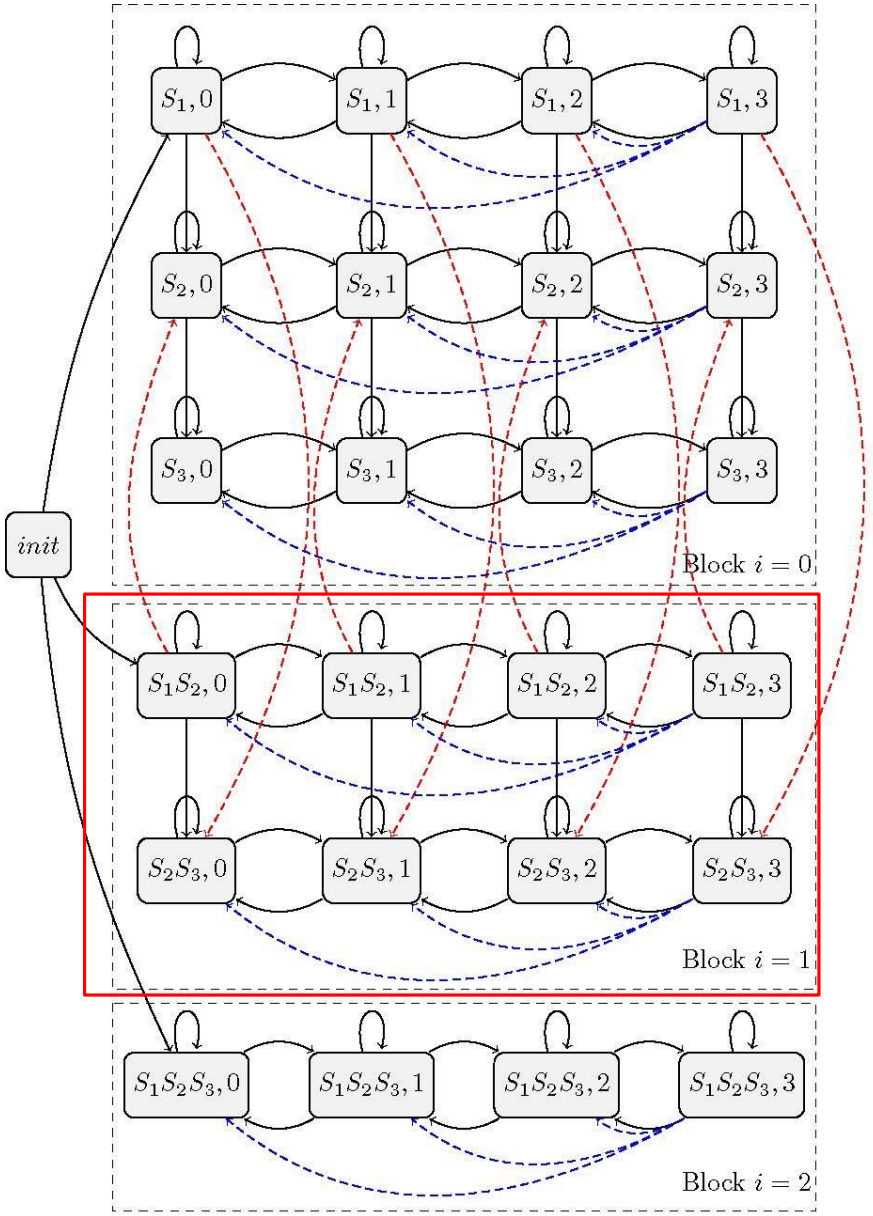
Why don't we have s1,s3?



Formal Model

Blocks Represent *i* number of microservices to proactively deploy

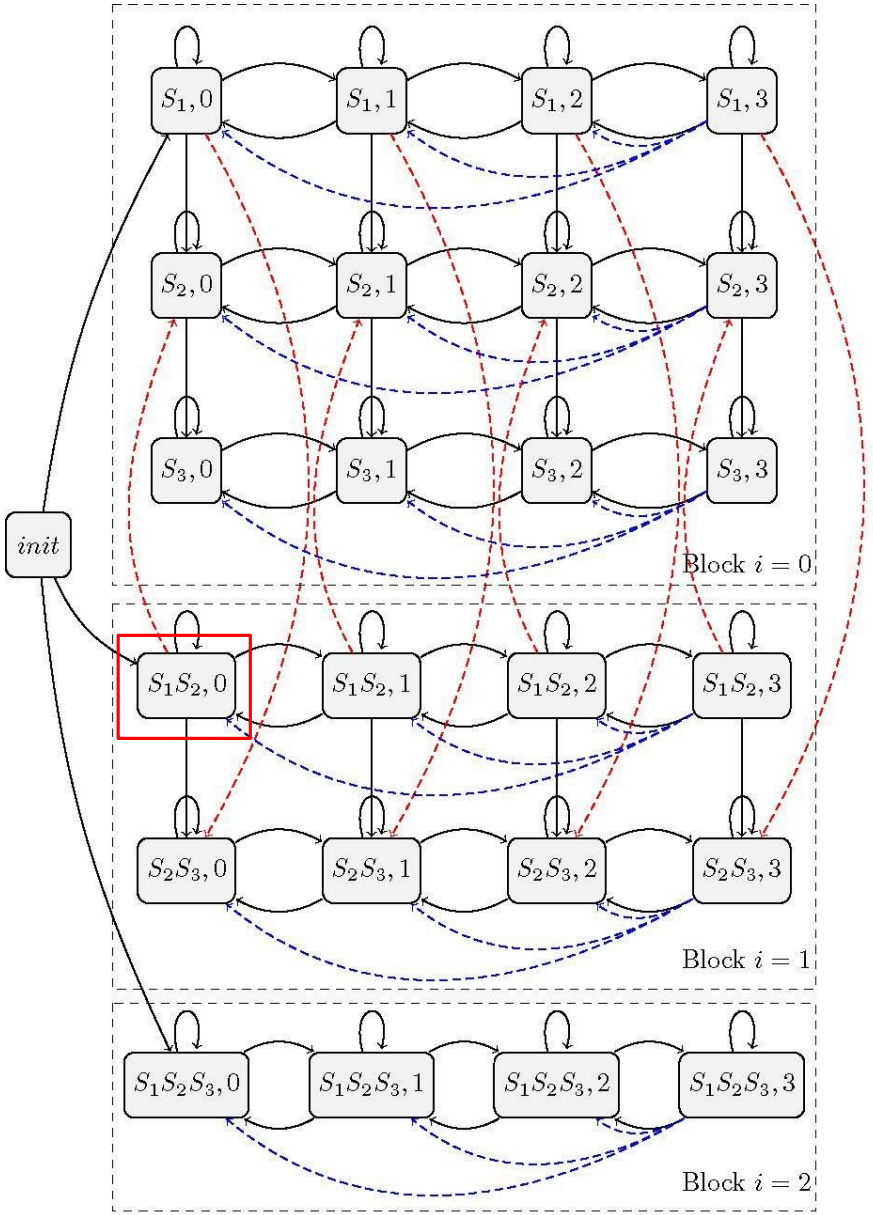
Blocks i=0 to *i=n-1*



Formal Model

Blocks Represent *i* number of microservices to proactively deploy

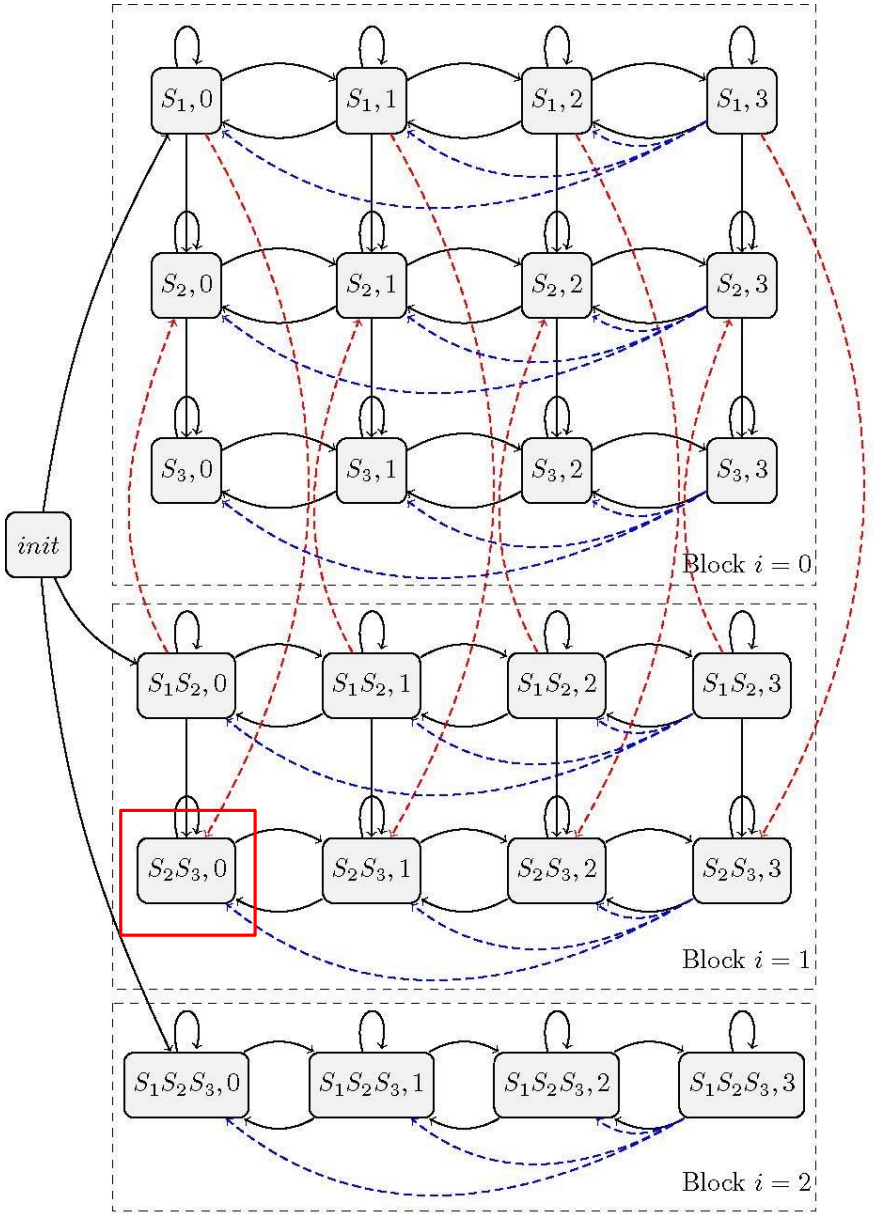
Blocks i=0 to *i=n-1*



Formal Model

Blocks Represent *i* number of microservices to proactively deploy

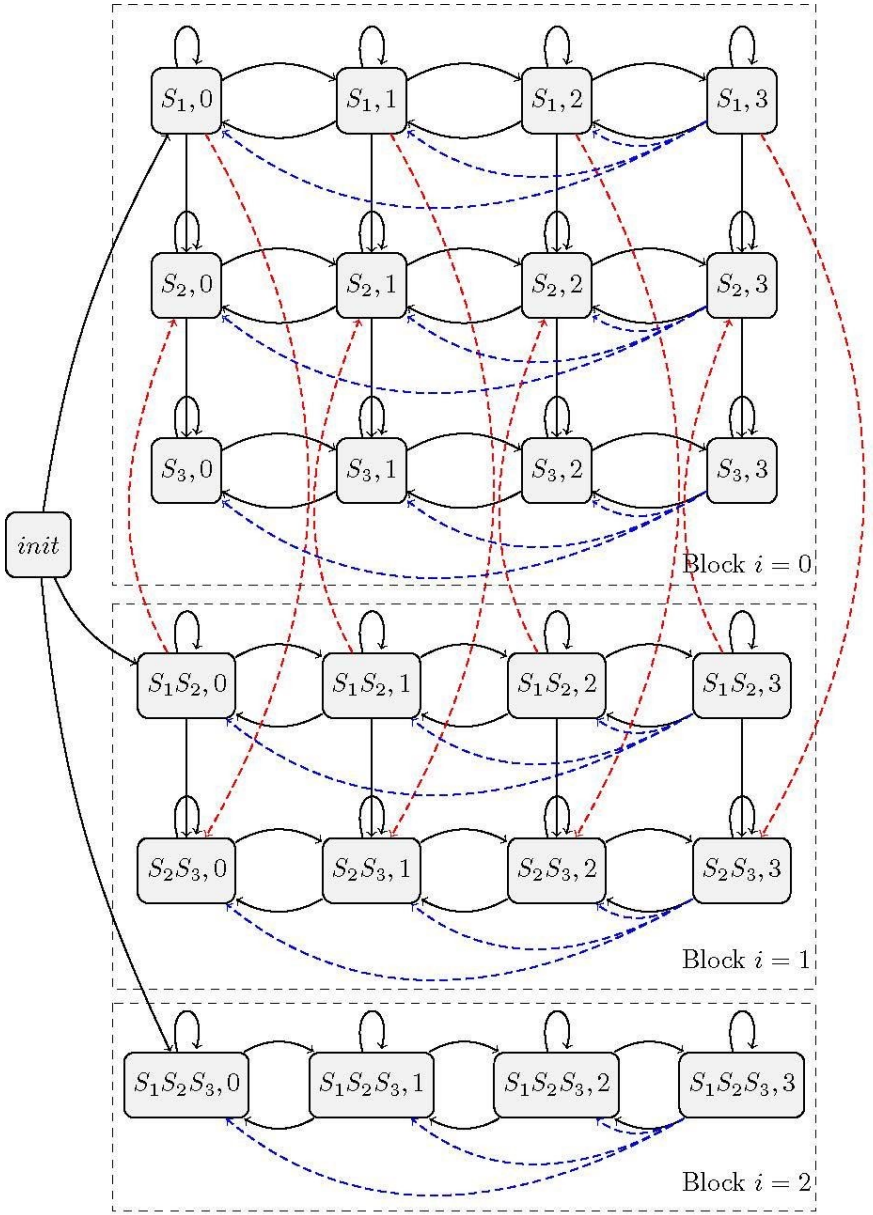
Blocks i=0 to *i=n-1*



Formal Model

Blocks Represent *i* number of microservices to proactively deploy

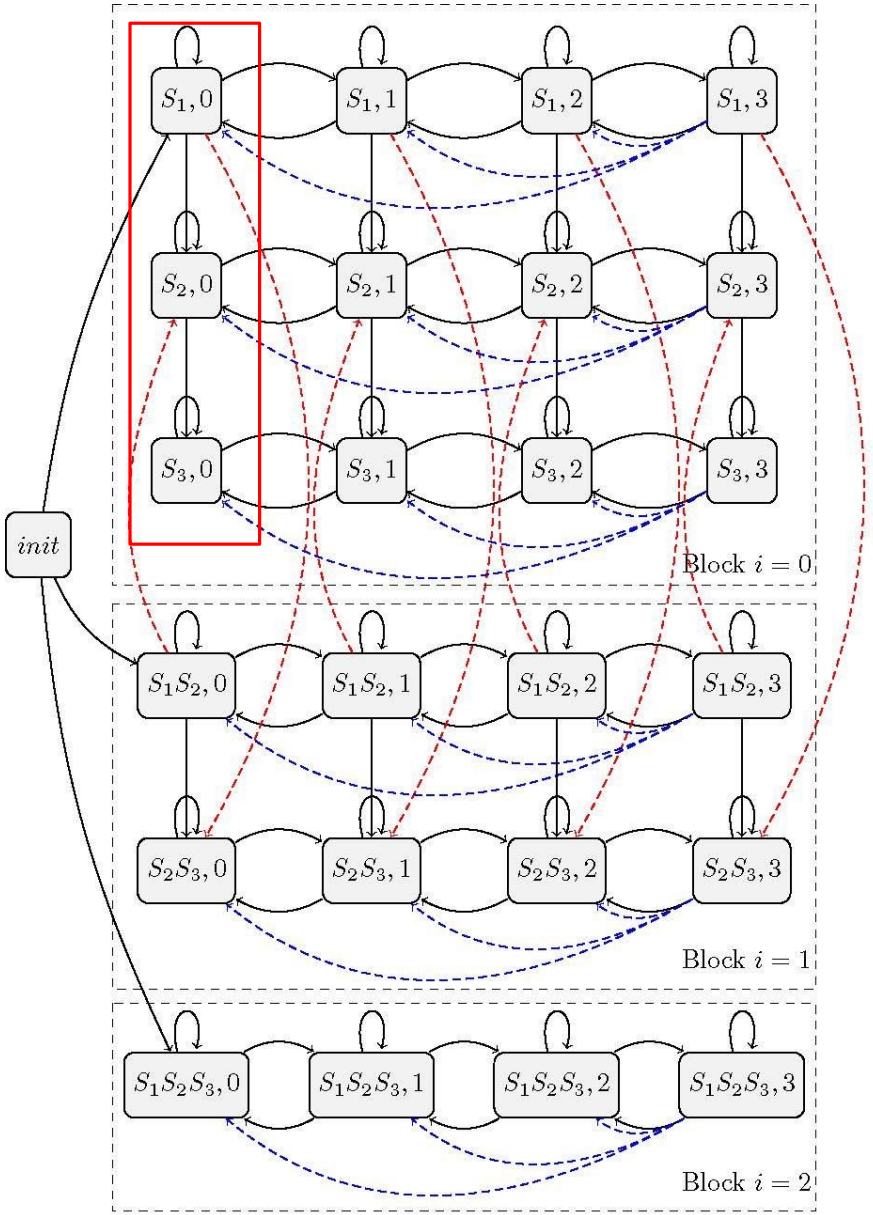
Blocks i=0 to *i=n-1*



Formal Model

Three types of transitions

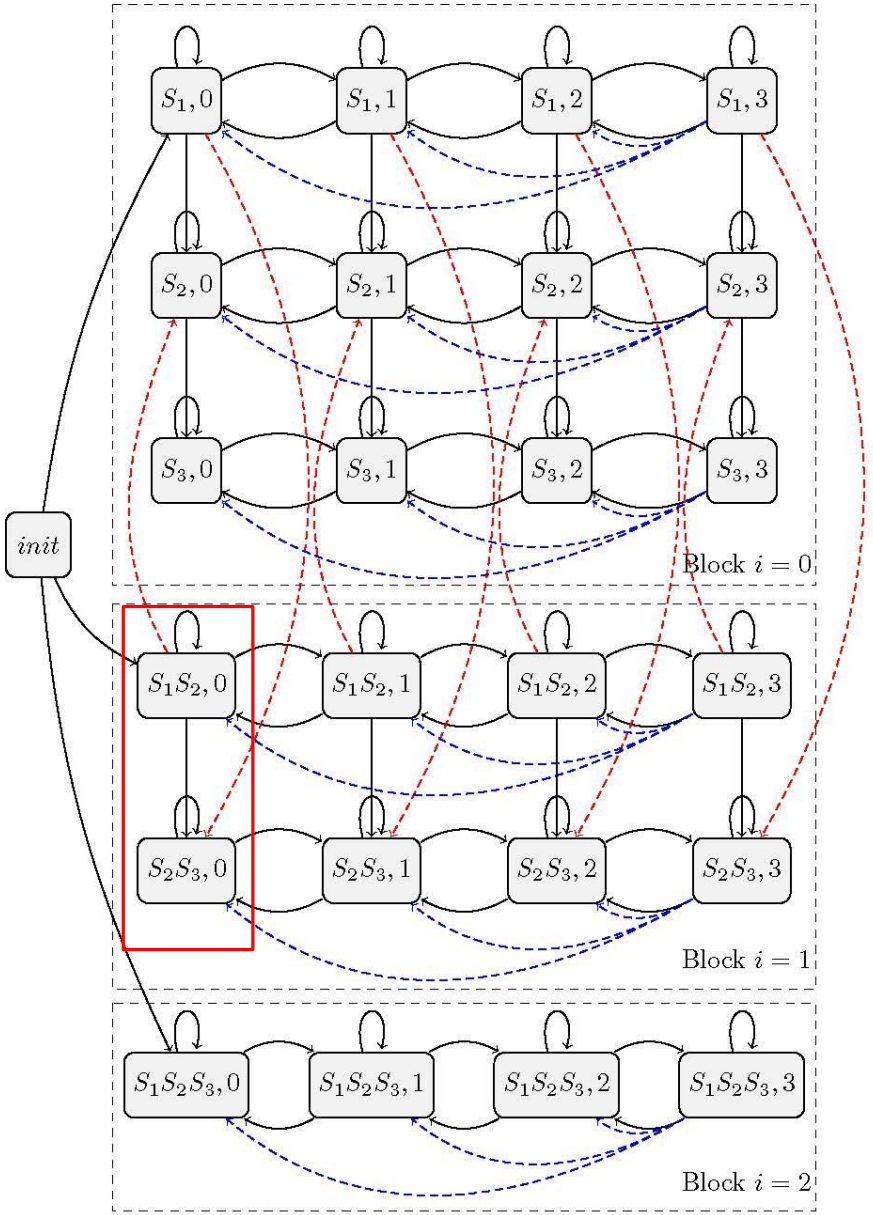
- User Movement
- Service Invocation in Application Topology
- Migration



Formal Model

Three types of transitions

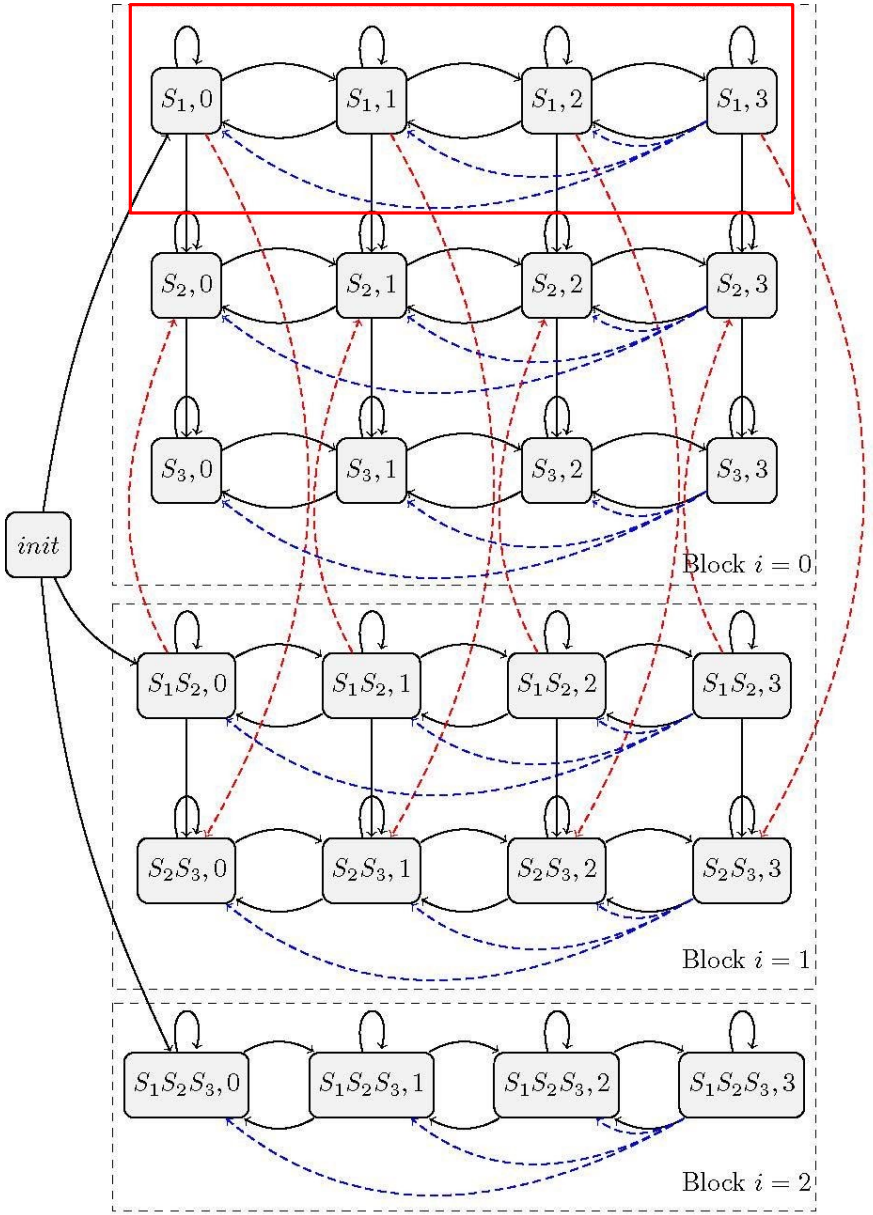
- User Movement
- Service Invocation in Application Topology
- Migration



Formal Model

Three types of transitions

- User Movement
- Service Invocation in Application Topology
- Migration



Formal Model

Three types of transitions

- User Movement
- Service Invocation in Application Topology
- Migration

Reinforcement Learning Solution

Algorithm 1: Dyna-Q

Initialize $Q(s,a)$ and $Model(s,a)$, $\forall s \in \mathcal{S}, \forall a \in A(s)$
while *true* **do**
  $s \leftarrow$ observe the application state
  $a \leftarrow \epsilon$-greedy(s,q)
  Observe the next state s' and the reward obtained
  Update $Q(s,a)$ using Equation 1
  Model $(s,a) \leftarrow r, s'$
  for $i = 0 \ldots n$ **do**
    $s \leftarrow$ random state previously observed
    $a \leftarrow$ random action previously taken in s
    $r, s' \leftarrow Model(s,a)$
    Update $Q(s,a)$ using Equation 1

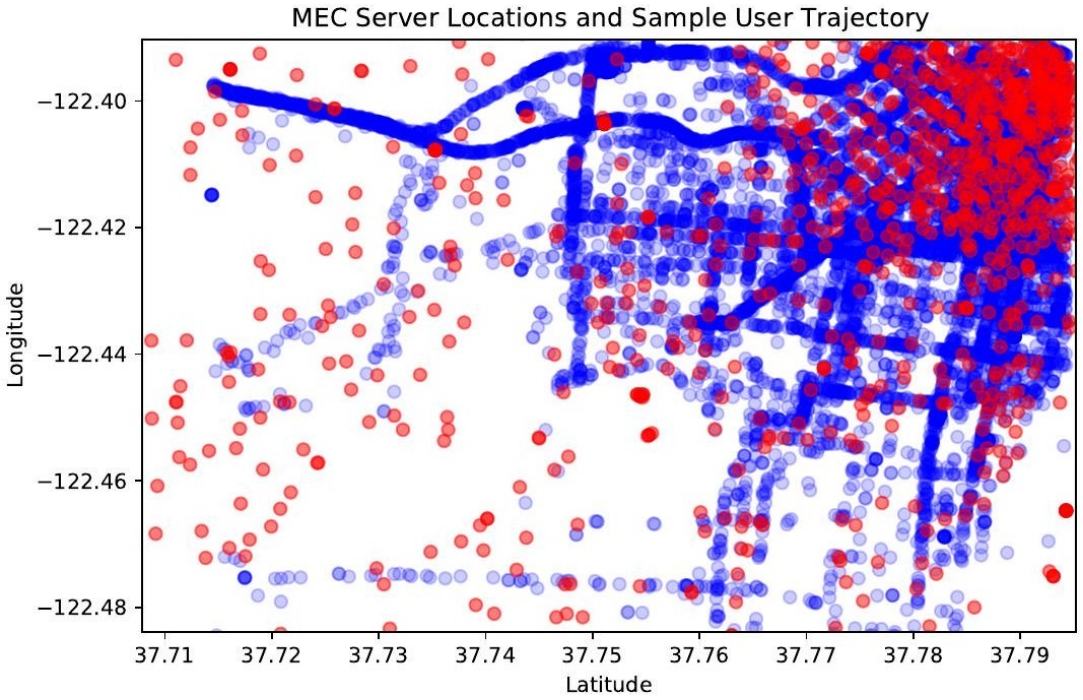
- Use Dyna-Q : Model Based and Model Free
- Simulation + Interaction
- Reward Function defined as a measure of prefetched and utilized services and prefetched and unutilized services
- Possible since we only transition upon service invocations

$$R = \sum_{\mu \in \mu_{used}} [\mu * c(\mu_{resources})] - \sum_{\mu \in \mu_{unused}} [\mu * c(\mu_{resources})]$$

Reinforcement Learning Solution

- Low Traffic some action receives a positive reward
- High Traffic same action may receive negative reward
- Variance leads to confusion
- Heuristic Solution : Three types of MDPs for each Application – ***{high, medium, low}*** – use the appropriate MDP depending on the traffic condition
- Capacity Constraint Heuristic : Allocate microservices greedily along the linear chain

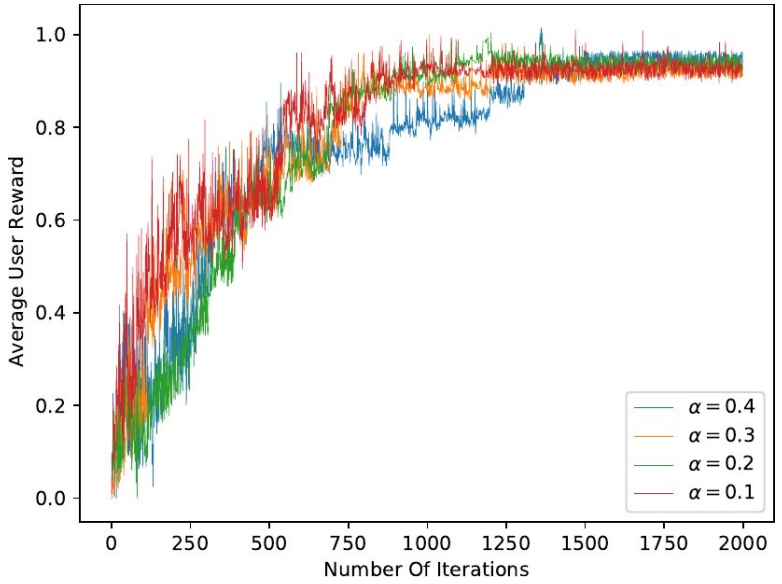
Dataset



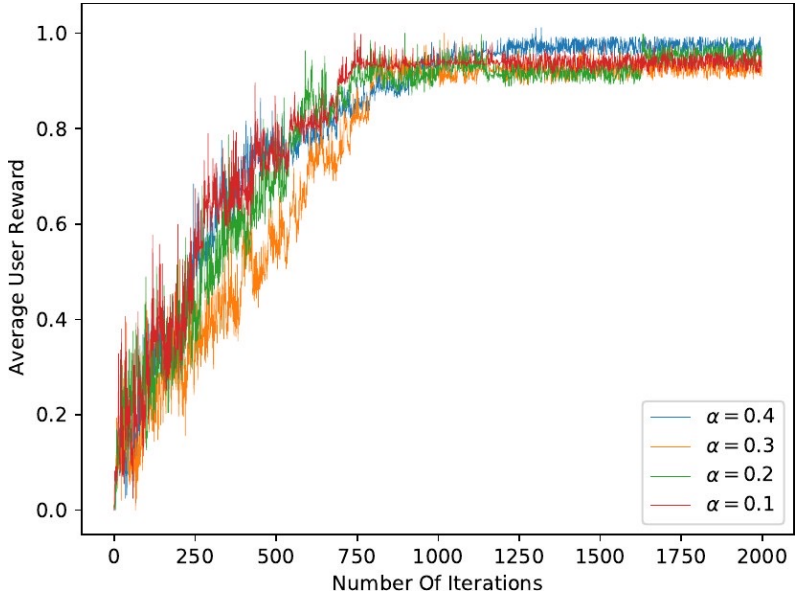
- San Francisco Taxi Dataset for User Trajectories https://crawdad.org/epfl/mobility/20090224/
- San Francisco Wireless Telecommunications Services Facilities Dataset for MEC Server Locations

  https://data.sfgov.org/Geographic-Locations-and-Boundaries/Existing-Commercial-Wireless-Telecommunication-Ser/aa26-h926
- Service invocations randomly generated along with representative timing for initialization from DeathStarBench suite

Accumulated Reward

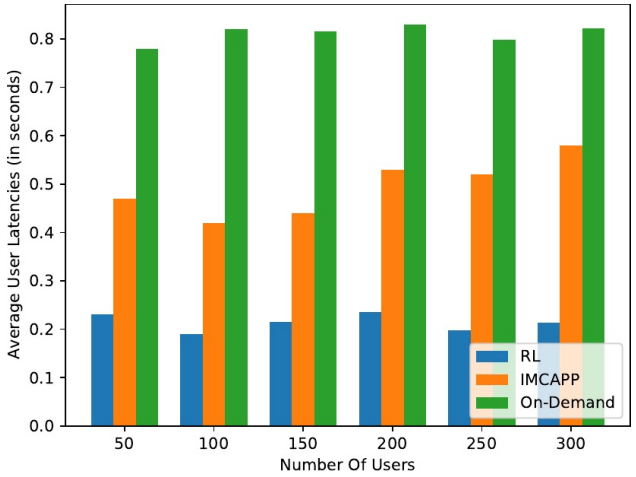


Users = 50

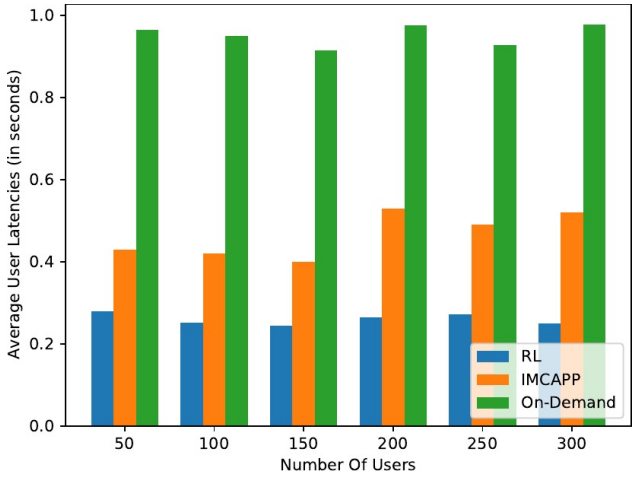


Users = 100

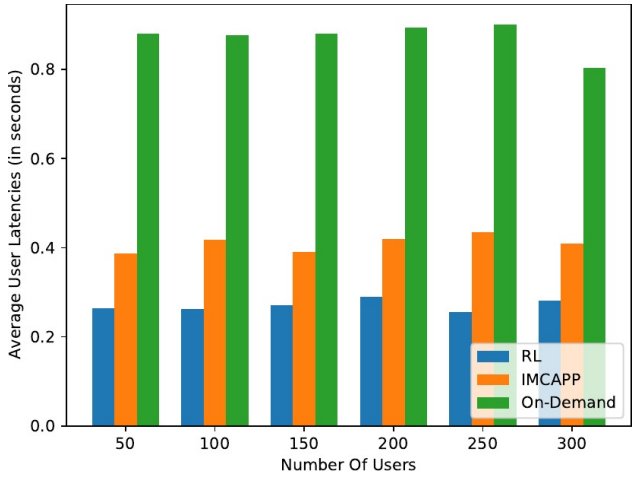
Average User Latency



4 Microservices

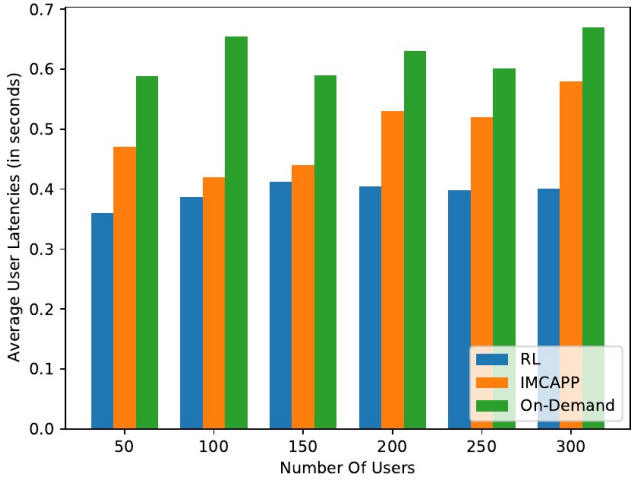


8 Microservices

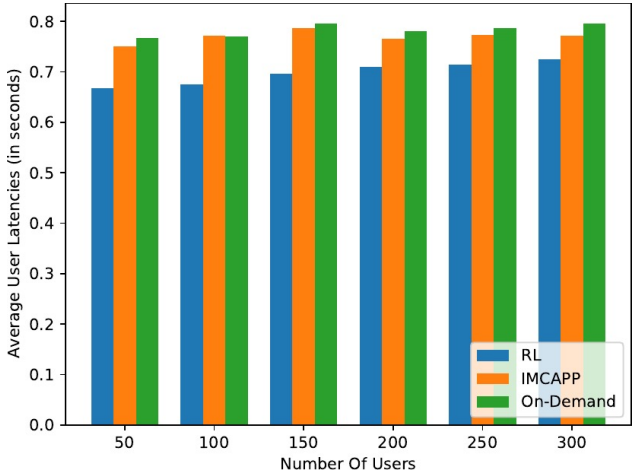


12 Microservices

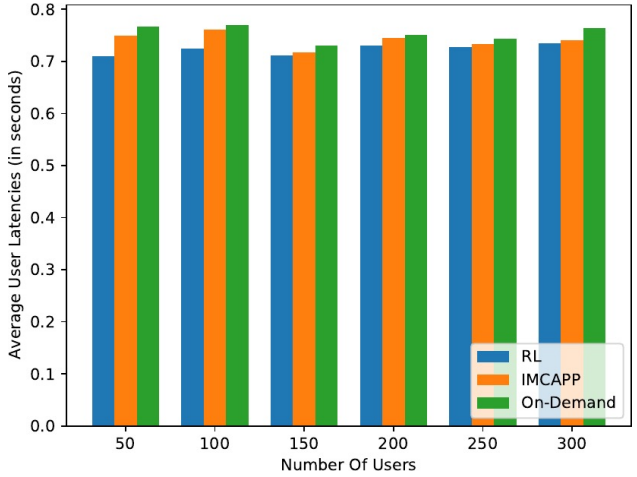
Average User Latency



Server Resources = 130%

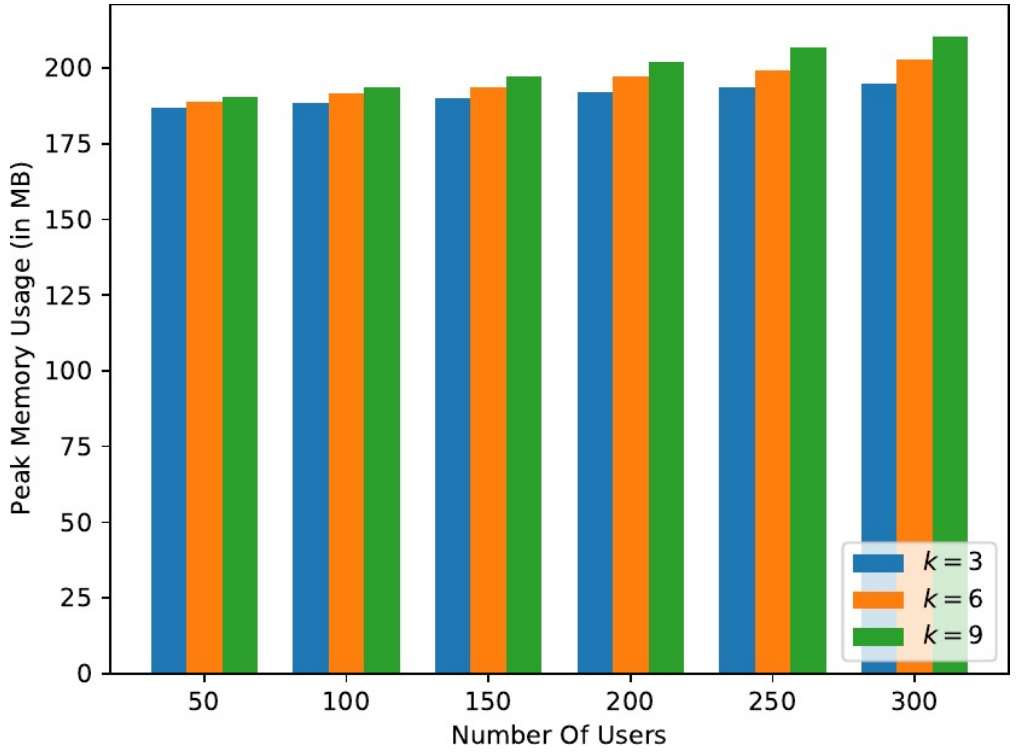


Server Resources = 100%



Server Resources = 65%

Memory Usage for Varying k



Thank You!