

Distributed Autonomous Virtual Resource Management in Datacenters Using Finite-Markov Decision Process

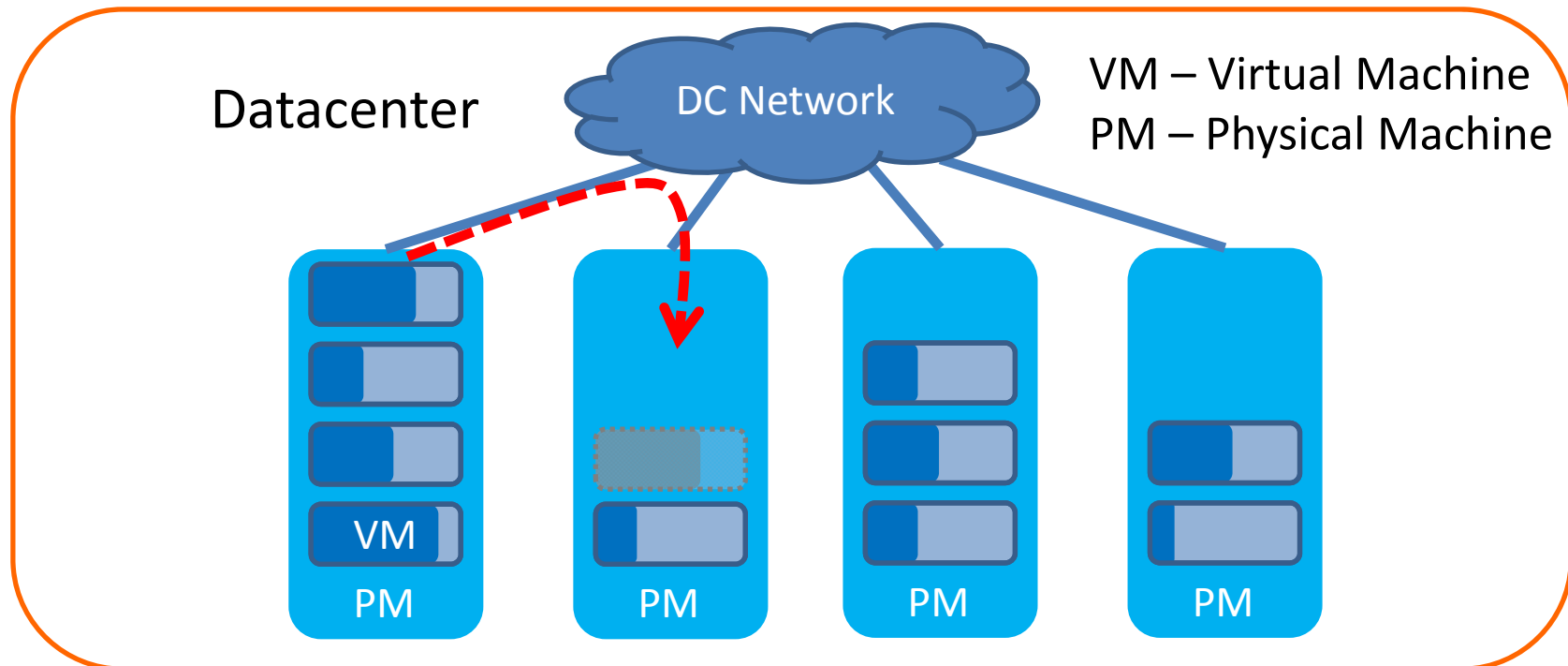
Liuhua Chen, Haiying Shen and Karan Sapra

Department of Electrical and Computer Engineering
Clemson University

CLEMSON[®]
UNIVERSITY



Load Balancing is Critical For IaaS Cloud



- ✓ Increase resource utilization
 - ✓ Reduce response time / SLA violations
- Increase profit

Existing LB Methods Have Limitations

Reactive Methods

Reactively perform VM migration upon the occurrence of PM overloads

Proactive Methods

Predict VM resource demand in a short time for sufficient resource provision or load balancing

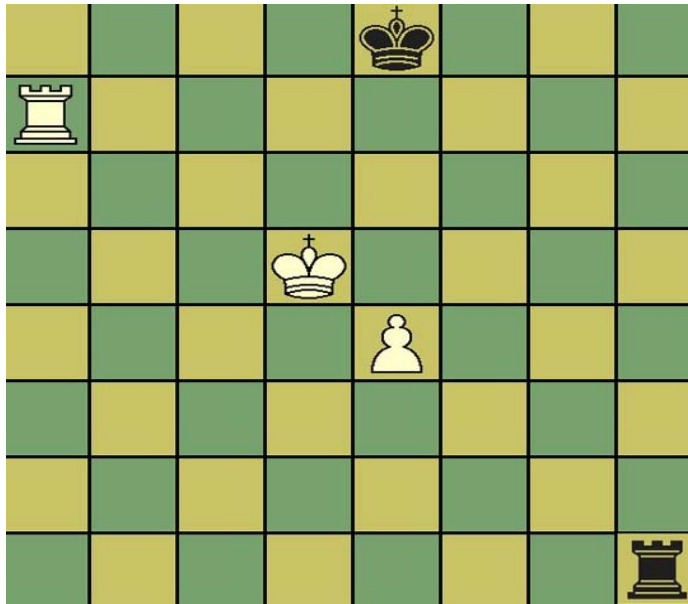
- × Selecting migration VMs and destination PMs generates:
 - high delay
 - high overhead
- × Cannot guarantee a long-term load balance state

Requirements for High Performance LB

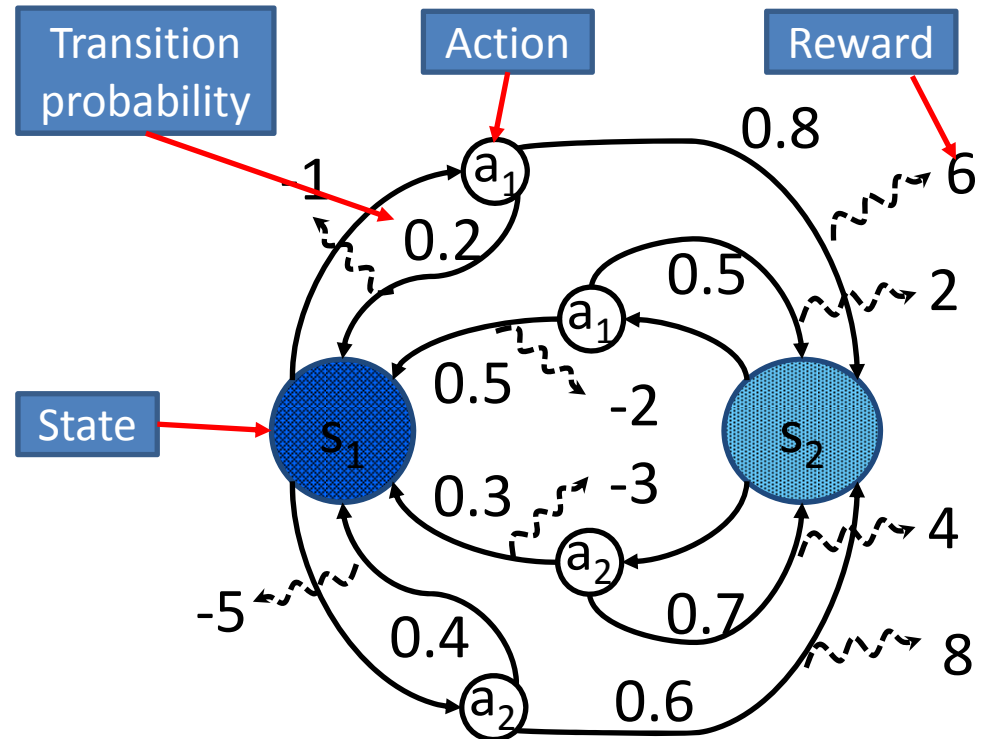
- ✓ Proactively handle the potential load imbalance problem
- ✓ Generate low overhead and delay for load balancing
- ✓ Maintain a long-term load balance state

Solution for High Performance LB

○ Solution

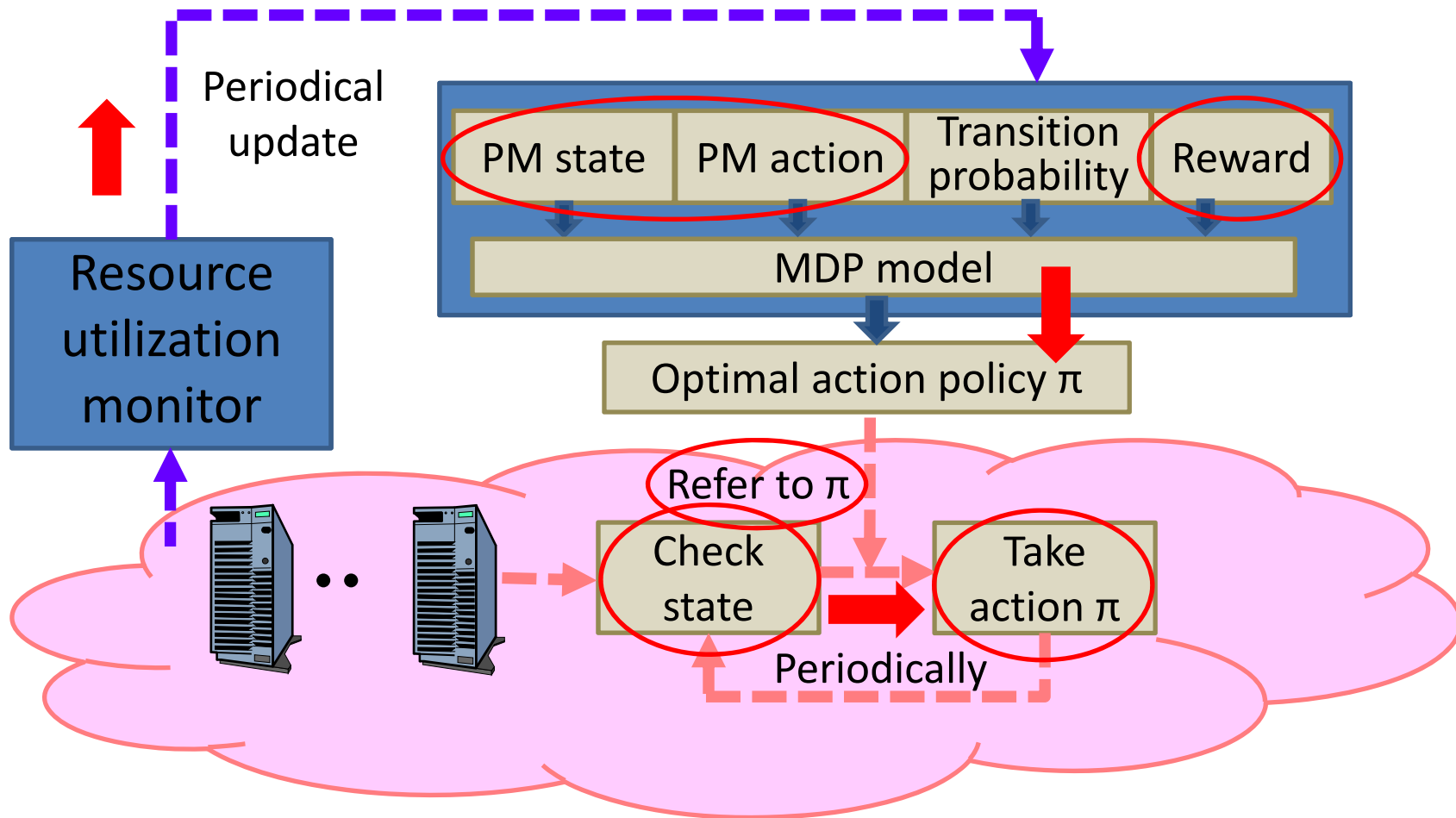


- ✗ Consider current situation
- ✗ Consider next step
- ✓ Consider many steps



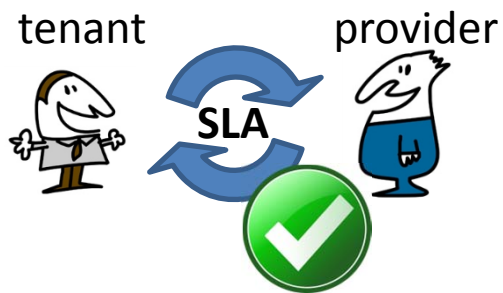
- ✓ Finite-Markov Decision Process (MDP)

MDP-based Load Balancing



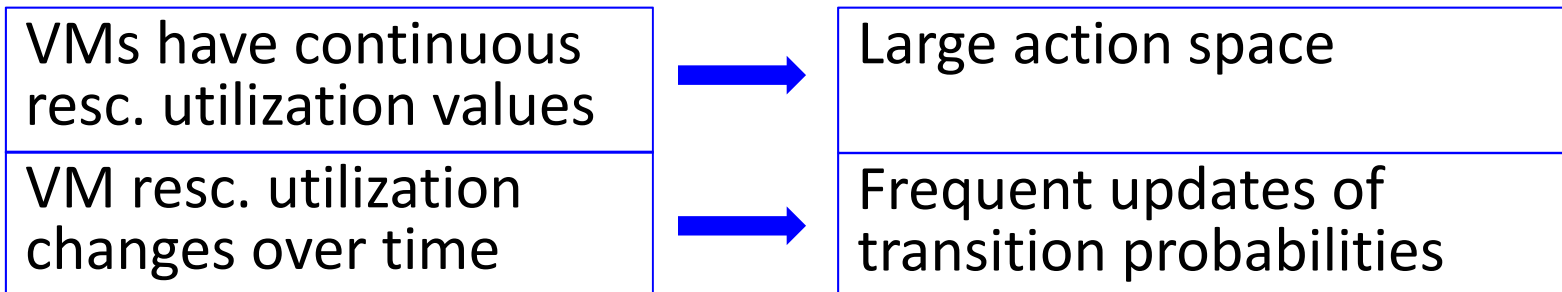
Advantages of MDP-based Load Balancing

- Achieves long-term load balance hence reduces SLA violations
- Reduces the overhead and delay of load balancing
- Builds one MDP used by all PMs

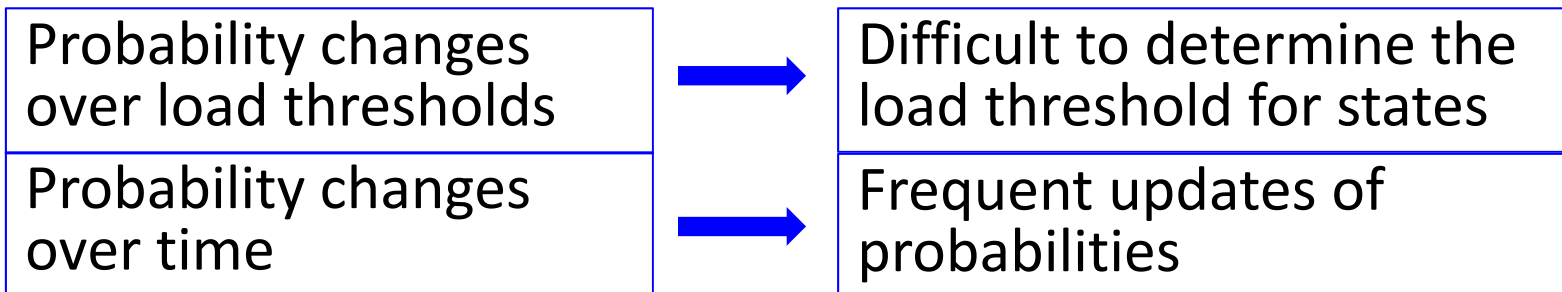


Challenges of the MDP Design


1. MDP components must be well designed for low overhead



2. Transition probabilities in the MDP must be stable

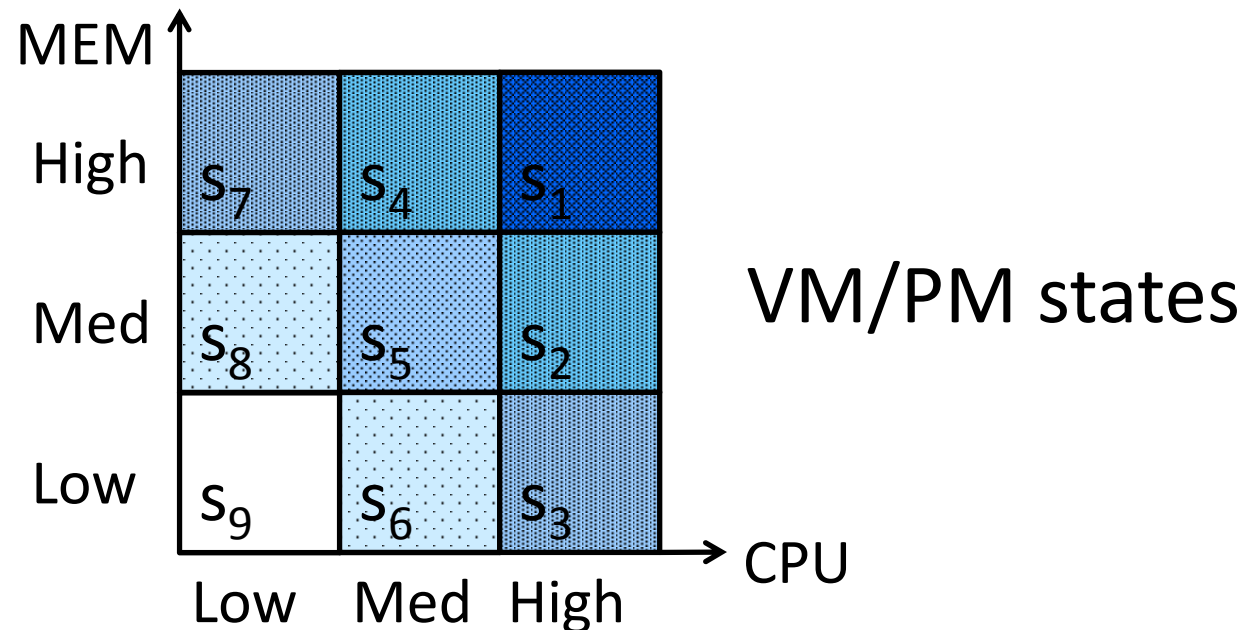


Solution to Challenge 1

- Action: moving out a specific VM
- 
 - needs to record the state transitions of a PM for moving out each VM
 - generates a prohibitive cost due to many VMs
 - is not accurate due to time-varying VM load

Solution to Challenge 1

- Action: moving out a VM state (high, med, low)
 - uses a PM load state as a state
 - records the transitions between PM load states by moving out a VM in a load state



Solution to Challenge 1

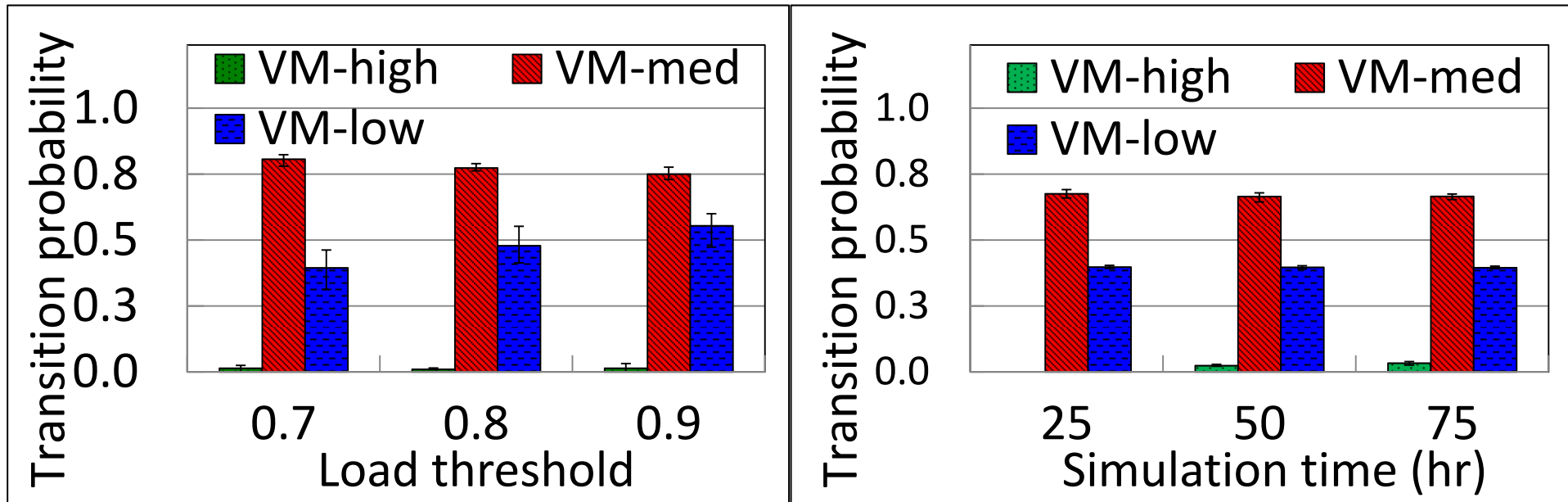
- Action: moving out a VM state (high, med, low)
 - uses a PM load state as a state
 - records the transitions between PM load states by moving out a VM in a load state
- Advantages:
 - the total number of VM-states in the action set does not change
 - each VM-state itself does not change, so the associated transition probability does not change



Challenge 2

- MDP's transition probabilities should be stable
 - otherwise, it cannot accurately provide guidance
 - must be updated very frequently to keep the transition probabilities accurate
- We have studied transition probabilities on VM migrations based on traces (Google Cluster trace and PlanetLab VM trace)
 - stable under slightly varying load threshold
 - stable over time

Trace Study on Transition Probabilities

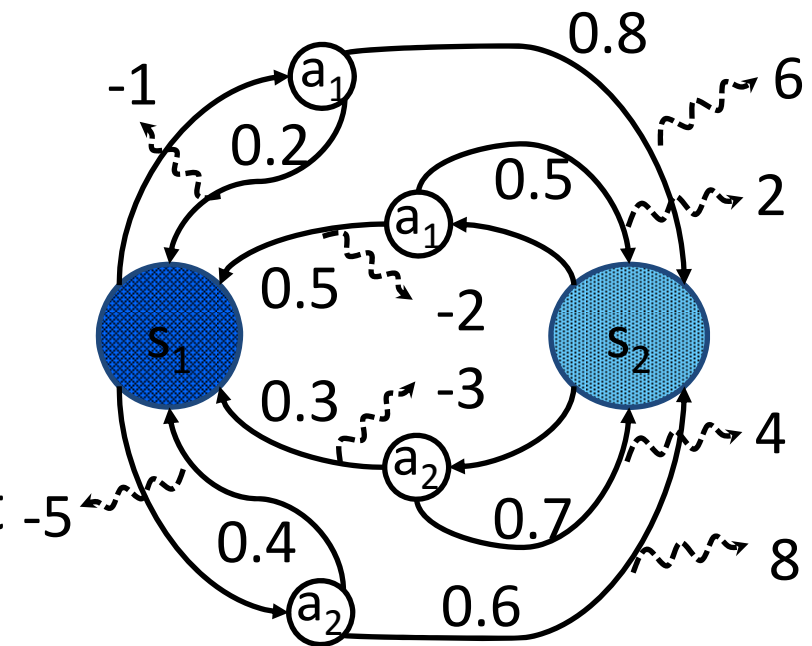


The probabilities that PM-high \rightarrow PM-medium by taking different actions, using Google Cluster trace

Stable under varying threshold and over time

MDP Components

- **State:** classification of resource utilization of a PM
- **Action:** a migration of VM in a certain state (VM-State)
- **Transition probability:** the probability that state s will transit to state s' after taking action a
- **Reward:** given after transition to state s' from state s by taking action a

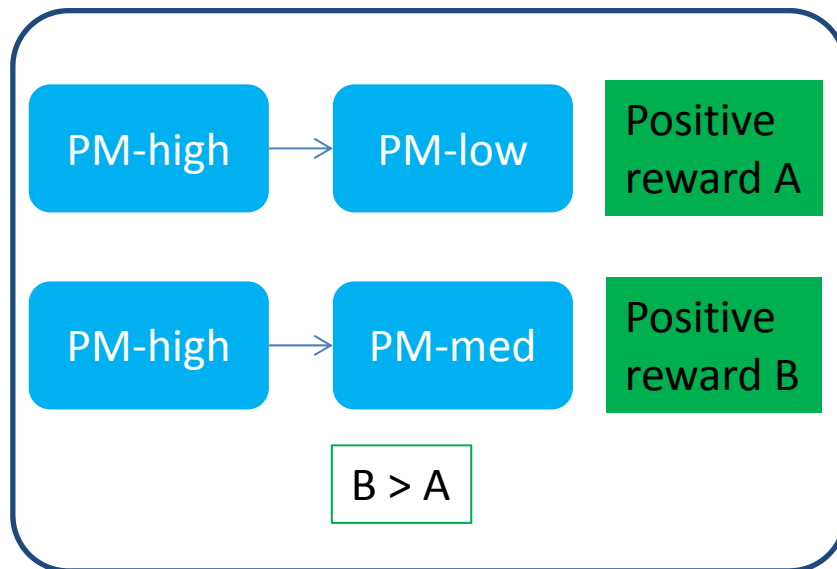


MDP model

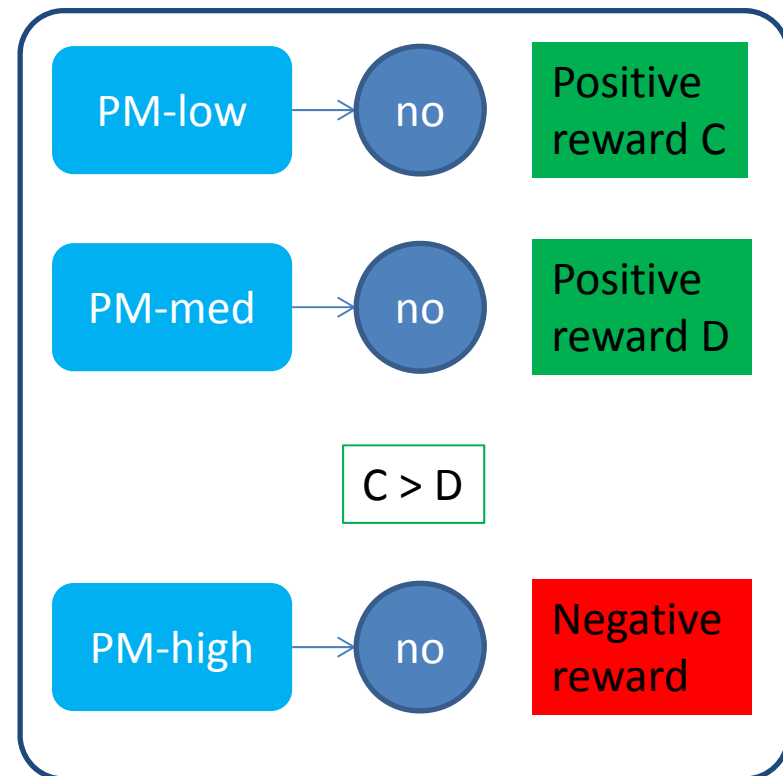
Rewarding Policy

- Goal: avoid heavily loaded state of PMs while constrain the number of VM migrations

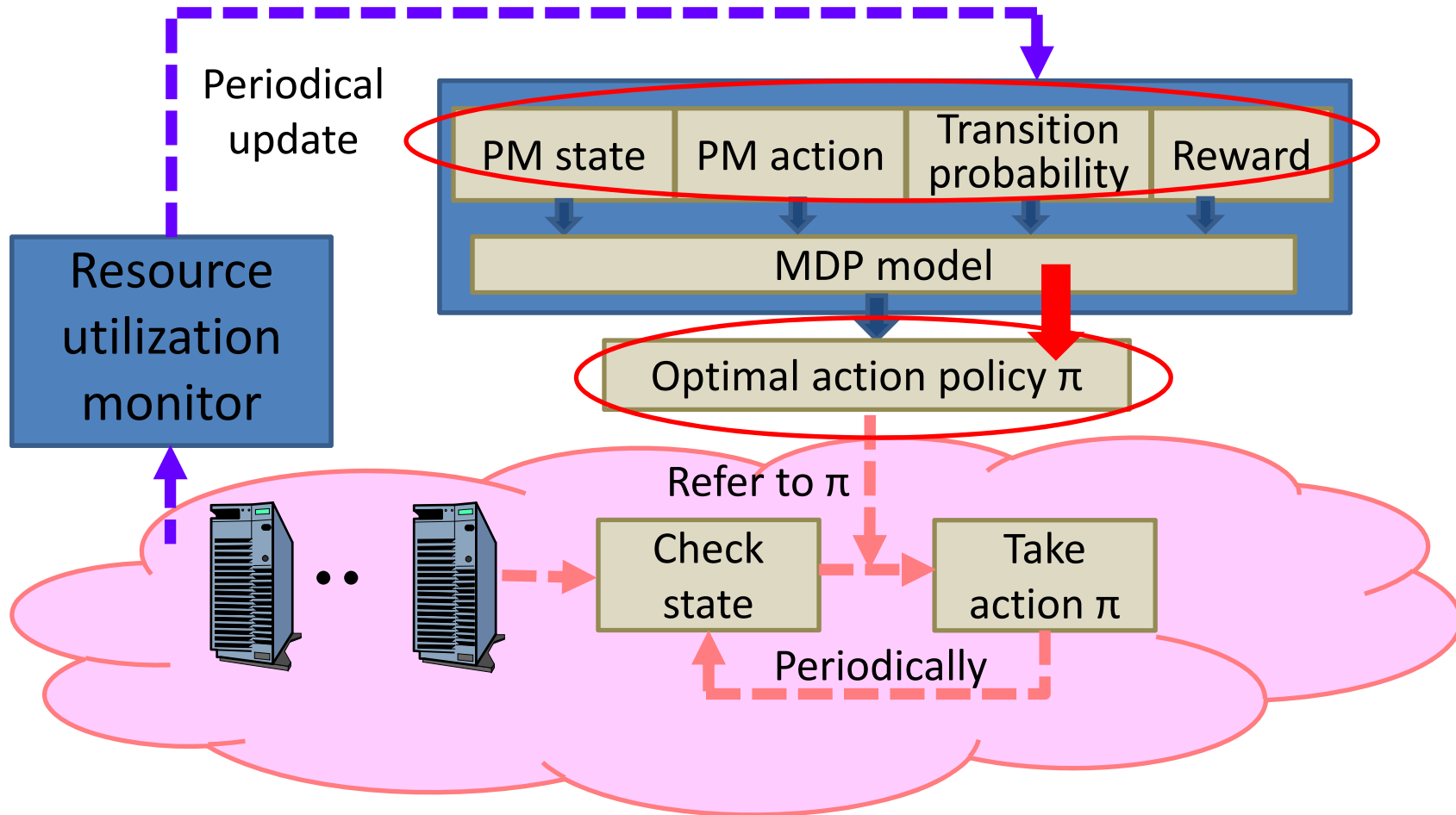
Rewards for state changes



Rewards for no actions



MDP-based Load Balancing



Optimal Action Determination

Value-iteration algorithm: find an action for each specific state that can quickly lead to the maximum reward

$$\pi(s_i) = \arg \max_a \left\{ \sum_j (P_a(s_i, s_j)(R(s_i, s_j) + V(s_j))) \right\}$$

$$V(s_i) = \sum_j \underbrace{P_{\pi(s_i)}(s_i, s_j)}_{\text{Transition probability}} \underbrace{(R_{\pi(s_i)}(s_i, s_j))}_{\text{Reward}} + \underbrace{V(s_i)}_{\text{Value for a specific state, calculated based on rewards}}$$

Transition probability

Reward

Value for a specific state,
calculated based on rewards

Basic idea:

- Set $V=0$
- Iteratively update V by the equations
- Update optimal actions for states in each iteration

The number of iterations indicates how many steps are considered in the future

Experimental Setup

Simulator: CloudSim

Traces:

PlanetLab VM trace

Google Cluster trace

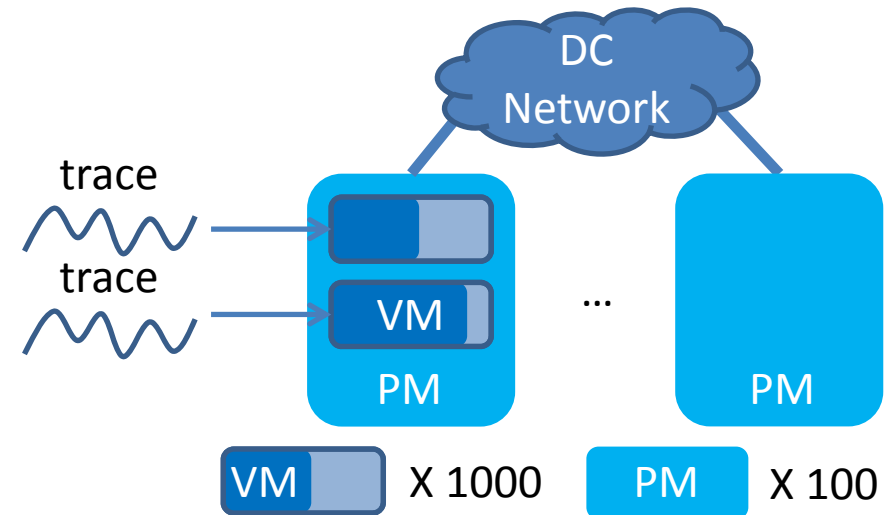
Load balancing is conducted every 300 seconds for 30 times

Implement two versions:

MDP uses the MDP model for migration VM selection;

MDP* uses the model for both migration VM selection and destination PM selection.

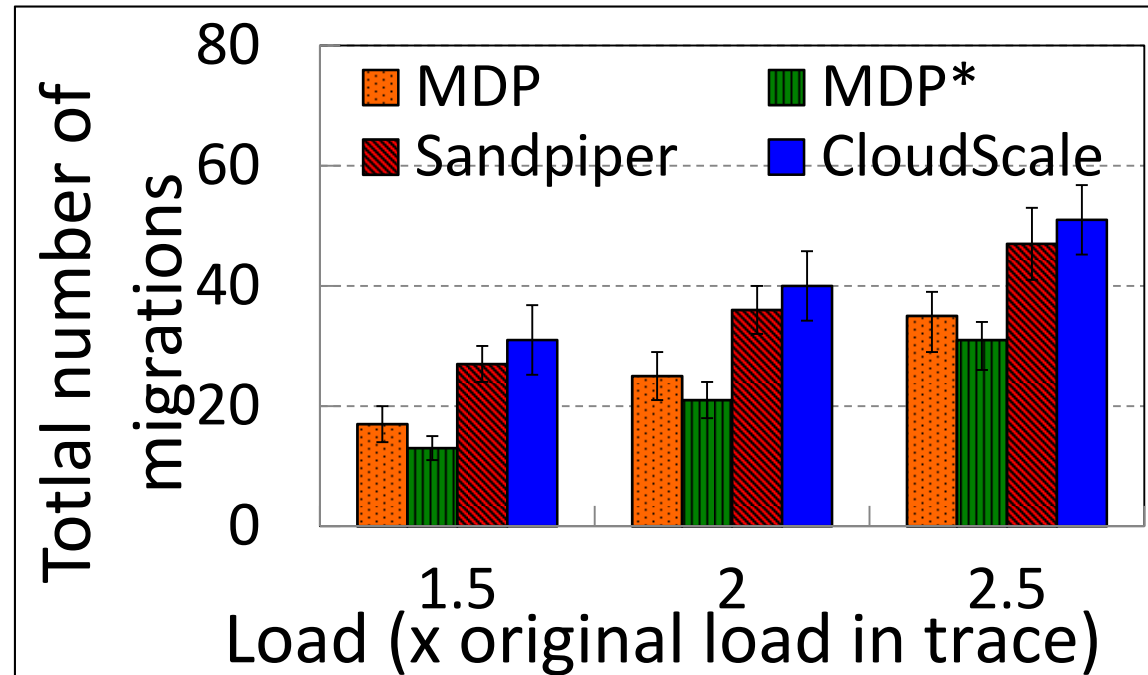
Comparison methods: Sandpiper [1], CloudScale [2]



[1] T. Wood, P. J. Shenoy, A. Venkataramani, and M. S. Yousif. Black-box and gray-box strategies for virtual machine migration. In Proc. of NSDI, 2007.

[2] Z. Shen, S. Subbiah, X. Gu, and J. Wilkes. CloudScale: Elastic resource scaling for multi-tenant cloud systems. In Proc. of SOCC, 2011.

The Number of VM Migrations

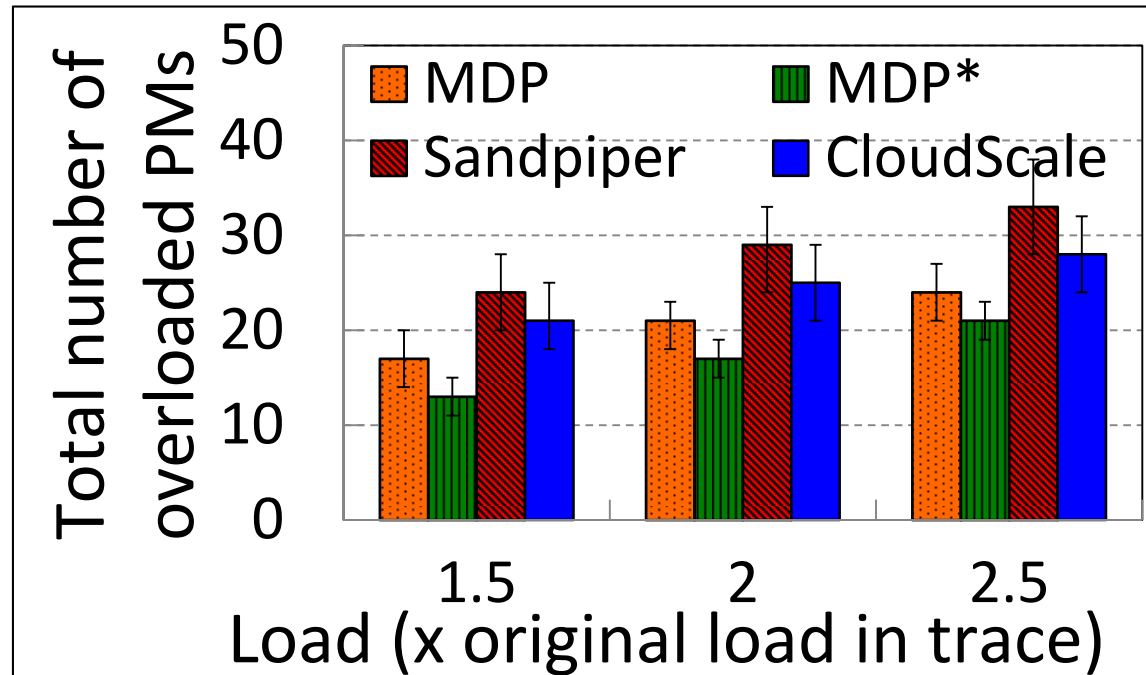


Result: $MDP^* < MDP < Sandpiper < CloudScale$

Conclusion: MDP* has the least number of migrations

→ the lowest load balancing overhead

The Number of Overloaded PMs

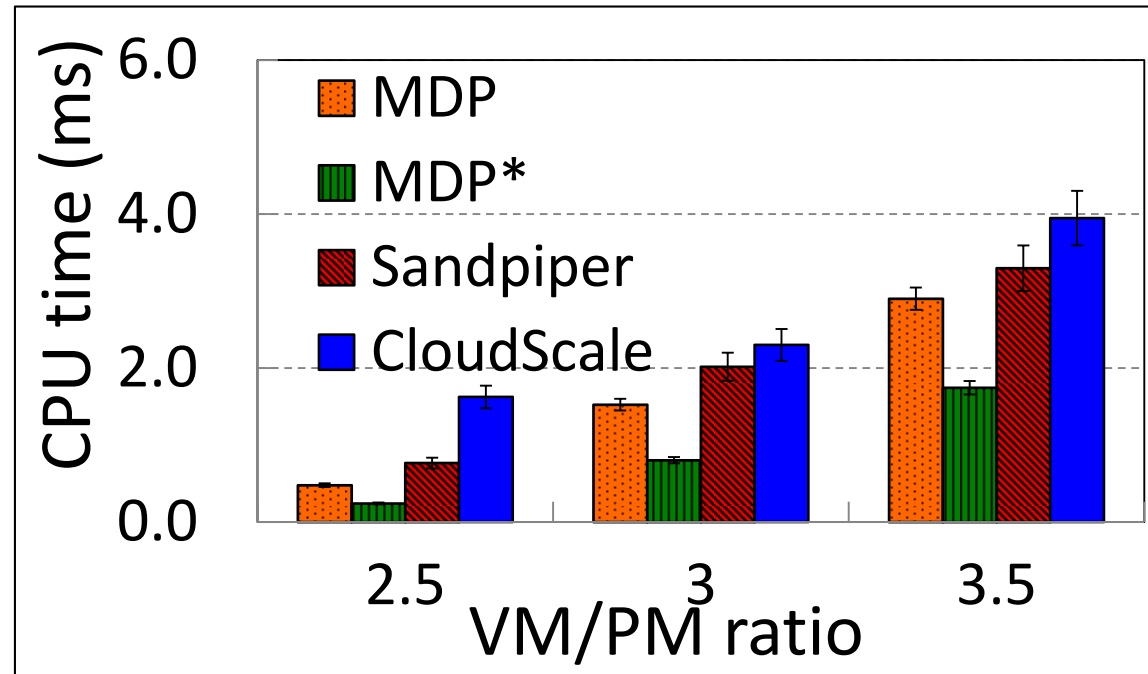


Result: $MDP^* < MDP < Sandpiper < CloudScale$

Conclusion: MDP* has the least number of overloads

→ the longest time for the load balance state

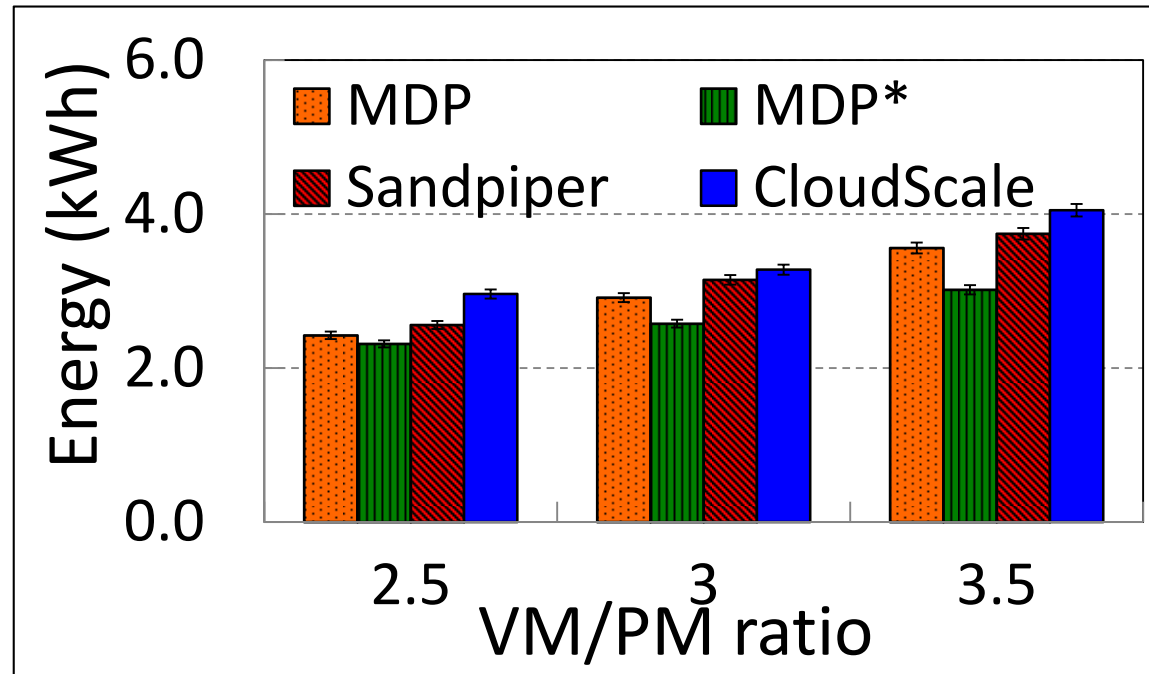
The CPU Time Consumptions



Result: $MDP^* < MDP < CloudScale < Sandpiper$

Conclusion: MDP* is the fastest in load balancing

Total Amount of Energy Consumptions

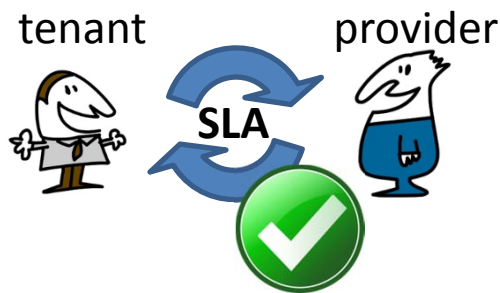


Result: $MDP^* < MDP < Sandpiper < CloudScale$

Conclusion: MDP* consumes the least amount of energy

Conclusion from Experimental Results

- The MDP model:
 - ✓ Maintain the load balance state for a longer time
 - ✓ Reduce SLA violations
 - ✓ Reduce the load balancing overhead and delay



Summary

- **Motivation:** Provide a load balancing method that can reduce SLA violations and meanwhile reduce the load balancing overhead and delay
- We **propose an MDP-based load balancing method** as an online decision making strategy to enhance the performance of cloud datacenters
- We **conducted trace-driven experiments** to show that our method outperforms previous reactive and proactive methods
- **Future work:** make our method fully distributed to increase its scalability