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

Vijetha Vijayendran

Motivation

- Cloud computing
  - The hype around the cloud!
  - Pay as you go model
  - Allows companies to focus on the core of their business
- Hardware Virtualization
  - Multiple virtual machines (VMs) running on a physical machine (PM)

Load Balancing Issues

- Over time, a PM may become overloaded
- Effects?
  - ☹ Affects the performance of other applications running on the PM
  - ☹ If applications receive insufficient resources, it may lead to SLA violations.
- Solution?
  - ☺ Migrate a VM to another PM
- How?
  - Load balancing algorithms

Proactive v/s Reactive Algorithms

- Reactive algorithms take corrective measures *after* a load imbalance has occurred.
  - ☹ High delay in restoring the load balance
  - ☹ High overhead in selecting destination PM
- Proactive algorithms take preventive measures by prediction to ensure that a load imbalance does not occur.
  - ☺ Prevents SLA violation to an extent
  - ☹ Which VM to migrate?
  - ☹Additional overhead - Every VM has to maintain a Markov Chain

☹ Cannot sustain the load balanced state

Markov Decision Process (MDP)

- MDP consists of
  - States (s), actions (a), transition probabilities (P) and rewards (R)
- Load States –
  - PM-State is the load state of a PM based on different resources
  - VM-state is the resource utilization level of a VM
  - Three levels for each resource – high, medium and low
- Total number of states = L^R
- Objective of the algorithm – ensure that utilization of every resource of the PM is below a certain threshold

MDP continued..

- Action - migration of a VM in a particular state, or no migration at all.
- Transition Probability - probability that an action a will lead to state s'.
- Reward – given after transition to state s' from state s by taking action a.

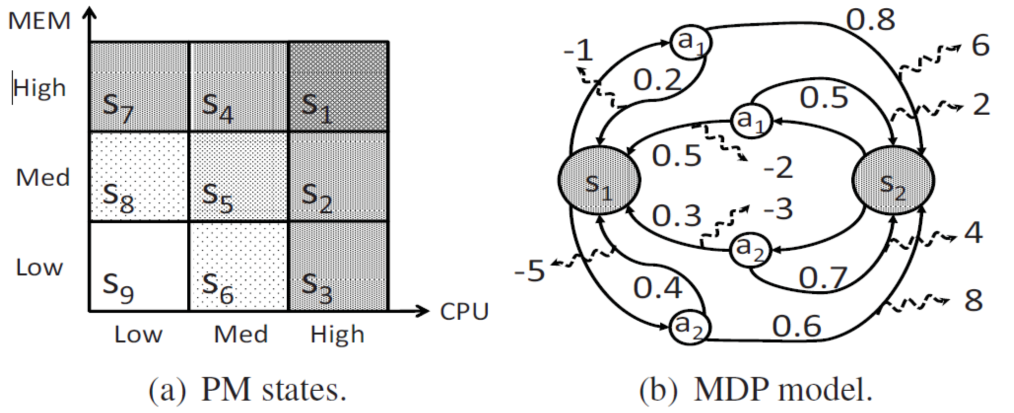


Figure 3. Example of a simple MDP.

States and Actions

- The state and action set remain constant.
- PM first determines its own state.
- It determines the state of all its VMs.
- MDP finds an optimal action and is able to sustain this state.

T1 = 0.3, T2 = 0.8

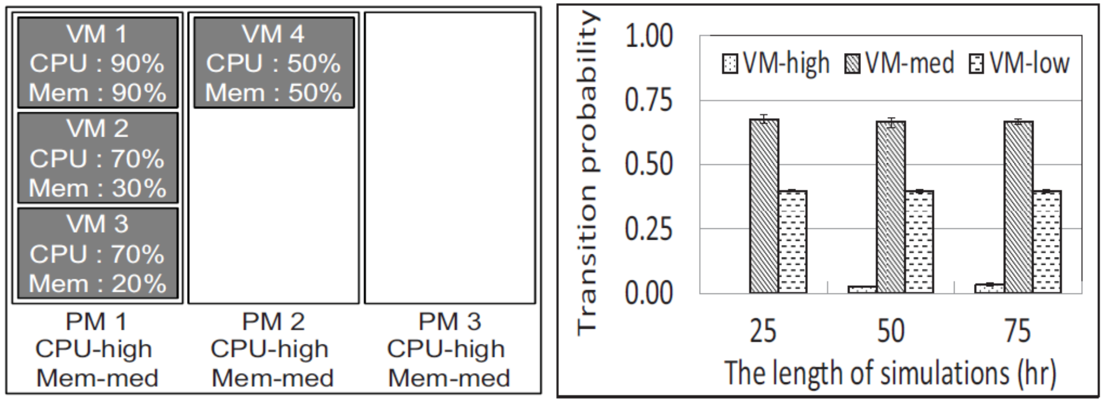


Figure 4. PM and VM state determination in a cloud.

Figure 5. Transition probability vs. simulation time.

State changes from PM high->medium

Transition Probabilities

- Determine the probability of transitioning to each state after action a
- Need to be stable
- Calculated by a central server using a trace of -
  - The states of the VMs being migrated
  - Changes in PM state after migration

	aH			aH			aH		
	vH	vM	vL	vH	vM	vL	vH	vM	vL
bH	0.01	0.13	0.59	0.03	0.65	0.39	0.96	0.22	0.02
bM	0.00	0.02	0.16	0.06	0.21	0.65	0.94	0.77	0.19
bL	0.00	0.00	0.01	0.00	0.00	0.08	0.00	1.00	0.91

Table 1. Probabilities with threshold $T_2 = 0.8$.

Rewards

- Encourages PMs to maximize rewards
- Positive reward
  - Transition from a high state to low or medium state.
  - No action in medium or low state
- Negative reward
  - Transition to high state
  - No action in high state

Optimal Action Determination

- Dynamic algorithm that finds the optimal action for every state

Algorithm 1 The iterative value iteration algorithm.

Require: T, a transition probability matrix
Require: R, a reward matrix.
Ensure: Policy π

$V \leftarrow 0, V_{new} \leftarrow R$
while $max|V(s_i) - V_{new}(s_i)| \geq e$ **do**
  $V \leftarrow V_{new}$
  for all state i in S **do**
    $V_{new}(s_i) \leftarrow R(s_i) + max_a \sum_j P(s_i, a, s_j)V(s_j)$
  end for
end while
for all s_i in S **do**
  $\pi^*(s_i) = \arg\max_a \sum_j P(s_i, a, s_j)V(s_j)$
  $\pi = \pi + \pi^*(s_i)$
end for
return π

Destination PM selection

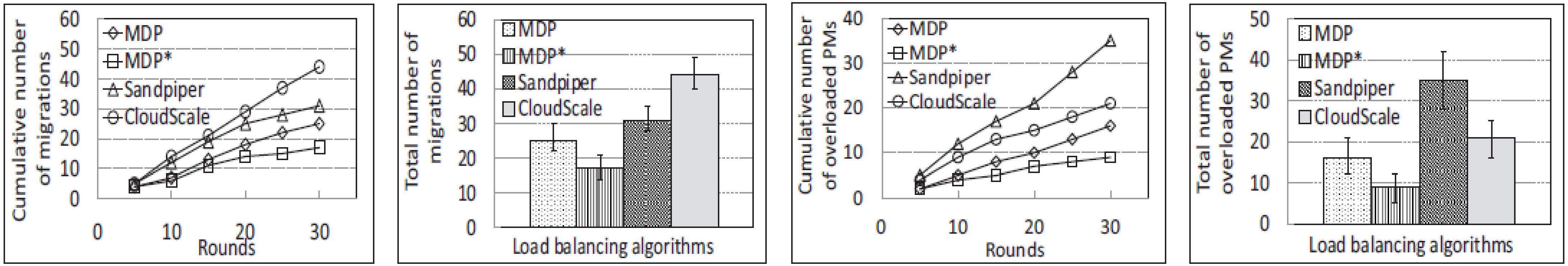
- Uses another MDP model to determine destination PM
- Done by central server
- Same state set
- Action set – Accept a VM in a certain state or not accept any VM
- Transition probability is similar – calculated using trace
- PMs are encouraged to accept VMs but avoid transitioning to heavy state

Performance evaluation

- CloudSim to conduct trace-driven experiments
- Used a 2 resource environment
- Compared 2 systems CloudScale (proactive) and Sandpiper (reactive) to
  - MDP – VM migration using MDP, destination PM selection using Sandpiper
  - MDP* – VM migration and destination PM selection using MDP
- 100 PMs hosting 1000 VMs, each experiment is run 20 times
- Resource utilization trace from PlanetLab and Google Cluster VMs
- T1 = 0.3, T2 = 0.8.

Experimental Results

Comparison of the performance of the four algorithms in terms of VM migrations and overloaded VMs (PlanetLab trace)

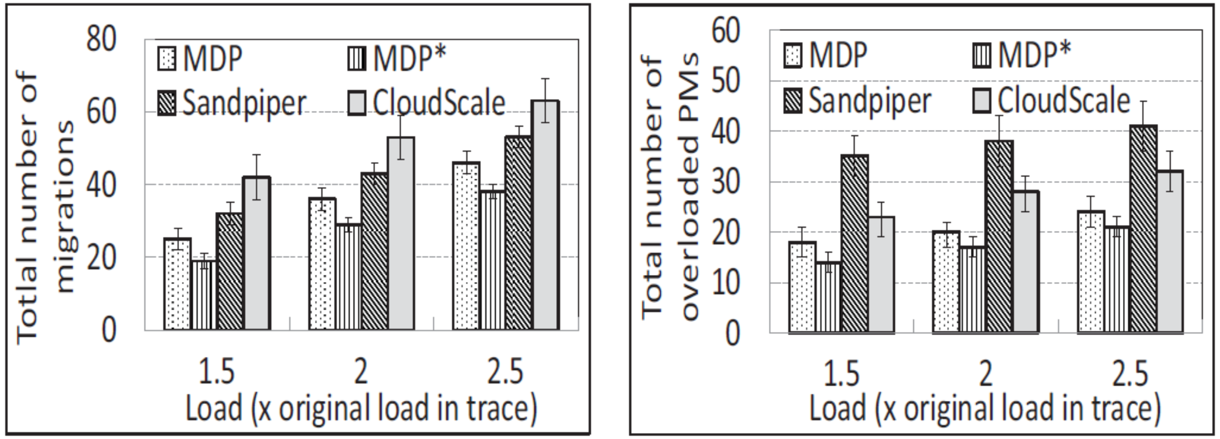


(a) Cumulative # of VM migrations. (b) Total # of VM migrations. (c) Cumulative # of overloaded PMs. (d) Total # of overloaded PMs.

Figure 6. Performance using the PlanetLab trace.

Experimental Results (contd..)

Comparison of the algorithms for different workloads



(a) The number of VM migrations with increasing workload ratio. (b) The number of overloaded PMs with increasing workload ratio.

Figure 8. Performance with the PlanetLab trace.

Experimental Results (Metrics)

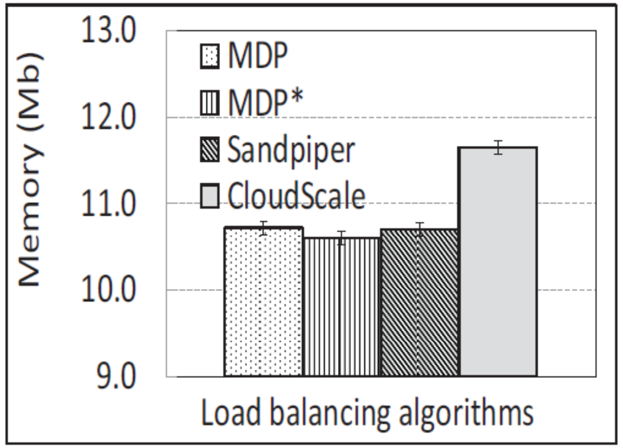


Figure 11. Memory consumption (ratio=3).

VM/PM ratio = 3

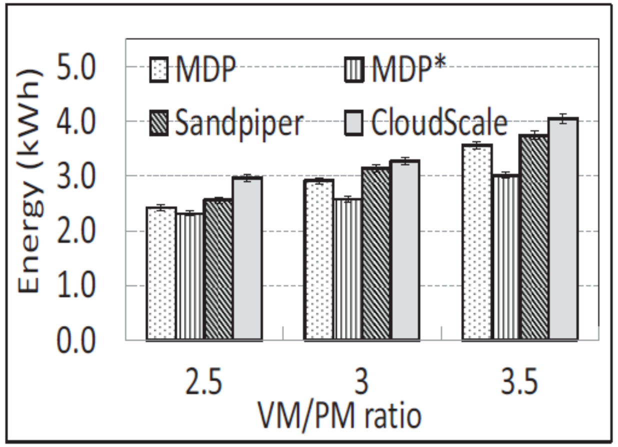


Figure 12. Energy consumption in algorithms.

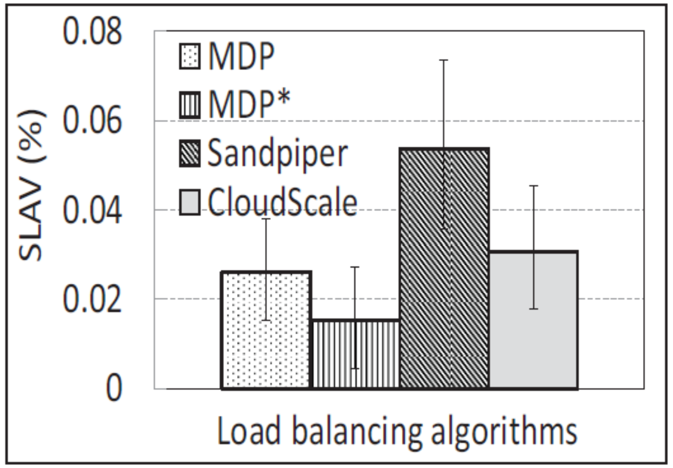


Figure 13. The SLAV metric.

Discussion

☺ Long term load balance is one of the strongest points

☺ Provides guidance on destination PM selection

☺ Stable probabilities, stable and consistent action set

☹ Algorithm runs in a central server – SPOF!

☹No guidance on how to select the interval of load balancing

😐 Scalable?

😐 How to set the reward values?