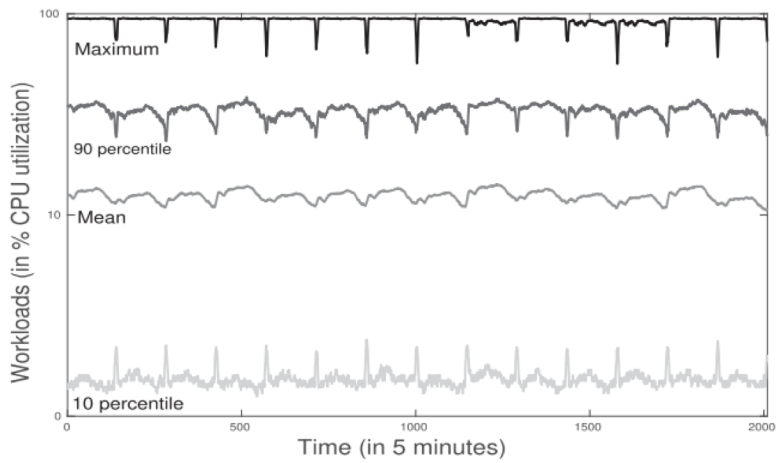
Megh

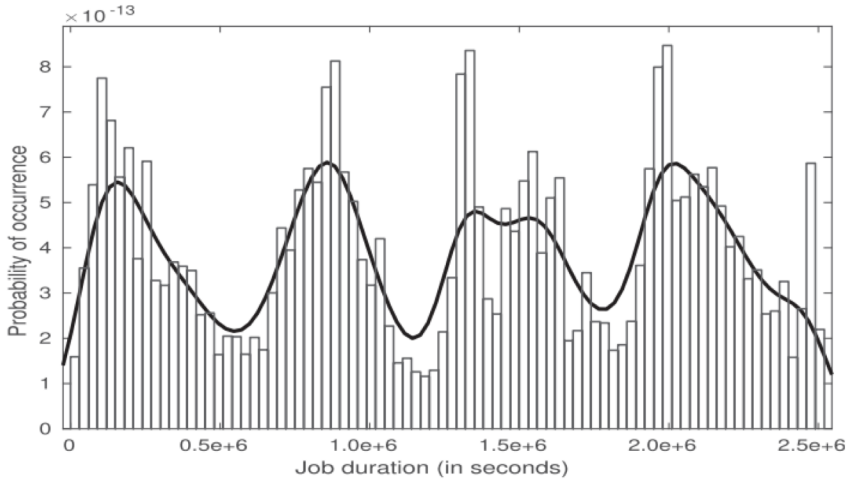
I. Introduction

- Virtual Machines (VM) on Physical Machines (PM) = ROI
- Different situations require different resources
- Energy Consumption, QoS, SLAs
- Nature of Workloads in Cloud data centers

# I. Introduction




(a) PlanetLab Workload




(b) Google Cluster workload

Fig. 1. Dynamics of PlanetLab workloads and distribution of task durations in Google Cluster.

I. Introduction

- Knowledge and Heuristic-based Algorithms < Reinforced Learning
- Development of Megh

# II. Related Works

- Dynamic VM Consolidation
  - ❖ Pack VM tightly while preserving SLAs
  - ❖ Max-Weight Algorithms

- Reinforcement Learning Algorithm
  - ❖ Markov Decision Processes
  - ❖ Q-Learning

# III. The Cloud Data Center: System and Cost Models

- Each PM defined by number of CPUs, CPU cores, RAM, and bandwidth
- Each VM allocated CPU performance, memory size, RAM, and bandwidth
- No knowledge of what apps will be used, simulates uncertain dynamics
- Megh = Global Resource Manager of Cloud
- CPU utilization data = key metric for workloads

# III. The Cloud Data Center: System and Cost Models

- $C_p(t)$ = data center power consumption cost
- $P(\Theta)$ dependent on CPU, memory, disk, and cooling
- Example shown with spec2014, using servers with different CPUs

TABLE 1
Power Consumption of Servers in Watts for Different Level of Workload [38], [39]

| Server Type | 0% | 10% | 20% | 30% | 40% | 50% | 60% | 70% | 80% | 90% | 100% |
|---|---|---|---|---|---|---|---|---|---|---|---|
| HP ProLiant G4 | 86 | 89.4 | 92.6 | 96 | 99.5 | 102 | 106 | 108 | 112 | 114 | 117 |
| HP ProLiant G5 | 93.7 | 97 | 101 | 105 | 110 | 116 | 121 | 125 | 129 | 133 | 135 |

$$C_{\mathrm{p}}(t) = c_{\mathrm{p}} \int_0^t P(\theta) d\theta, \qquad \forall t \geq 0. \tag{1}$$

# III. The Cloud Data Center: System and Cost Models

- SLA violation time = downtime
- $C_V(t)$ is the formula used
- 2 QoS Degradation: Overload or Live Migration

$$C_{\mathrm{V}}(t) = \sum_{j=1}^{N} c_{\mathrm{V}}^{j}(t), \qquad \forall t \geq 0. \tag{3}$$

$$c_{\mathrm{V}}^{j}(t) = \begin{cases} cv_1, & \text{if user's downtime percentage up to } t \\ & \in (0.05\%, 0.10\%] \\ cv_2, & \text{if user's downtime percentage up to } t \\ & > 0.10\% \\ 0, & \text{otherwise} \end{cases}$$

# IV Live Migration as a Learning Problem

- The authors formulate the problem of energy and performance efficient resource management during live migr. Of VMs as a reinforcement learning problem
- Data center has M pms, with a homogenous CPU capacity h.
- Each VM is assigned to a user, thus the the maximum # of users  is the max # VMs it could allocate

# IV Live Migration as a Learning Problem

- A sudden change in workloads in one or many VMs consequently overloads hosts
- This is when one of the VMs in the overloaded PM has to be migrated to another PM such that the cost for energy consumption and SLA violation remains minimal.
- The system has to decide which VM to move to where, and when to start moving so that the penalty will be minimum for max profit and max QoS for users

# IV Live Migration as a Learning Problem

- The process of live migration with uncertain workloads as MDP.
- State space S is the Cartesian product of C and W
- C is the set of all configurations of the VMs on the PMs
- W is an array of elements that represent the CPU usage of a VM at that instance
- W varies continuously and stochastically, making the state space infinitely dimensional and introduces uncertainty in state transitions.

# IV Live Migration as a Learning Problem

- The action space A corresponds to migration of any of the VMs from one PM to another depending on operating workloads.
- Each action is represented by (j,k), where j is the VM to be migrated and k is the destination PM
- Transition function f: S x A -> P(S), where P is the probability measure over state space.
- f returns the probability to reach another state

# IV Live Migration as a Learning Problem

- The cost of changing a configuration of VMs is the sum of the change of data center power consumption cost ($C_p$) and the change of cost of SLA violation ($C_V$)
- This formulation reduces the problem to finding the sequence of configurations that minimizes the sum of future per-stage costs.
- The problem can be phrased as finding the optimal policy that minimizes the expected cumulative cost.

# Megh: Learn to Migrate As-You-Go

- Megh answers three questions of the VM migration problem: *when* to start migrating the VM, *which* VM to migrate, and *where* i.e, to which PM to migrate it.
- In order to minimize expected cumulative cost, Megh starts with an initial guess of the policy, which changes as more information is gained such that the current estimation of accumulated cost remains minimal (policy iteration)

# Megh: Learn to Migrate As-You-Go

- In live VM migration, policy iteration suffers from two issues:
- First: To update the cost-to-go function (function for expected cumulative cost) and to find the optimal policy, the whole state action space must be searched through (combinatorially large, thus making policy updates expensive)
- Second: The Equation to updatye

# Megh: Learn to Migrate As-You-Go

- Second: The Equation to update the cost-to-go function that uses a dynamic programming technique cannot be computed given the stochastic nature of workload, VM configurations and transitions are not know *a priori*
- Authors don’t restrict the workload dynamic to a specific model in order to not narrow down the applications of Megh
- Megh solves both issues

# Megh: Learn to Migrate As-You-Go

- Megh solves the curse of dimensionality by projecting the state action space a d = NxM dimensional space X.
- X is spanned with d basis vectors $\Phi_{jk}$
- Each basis vector Φ corresponds to an action (j,k) such that the $_{jk}$th element of it is one and all other elements are zero.
- All actions or configuration changes in the cloud are represented using basis vectors or linear combinations of them

# Megh: Learn to Migrate As-You-Go

- Rationale: during transition from a state to another the accessible subspace is constructed by the states which are one action away from the present state.
- Instead of searching over the whole state space in each and every step it is logical to search in a subspace X that contains all the states s' reachable from s by actions Φjk or linear combinations of them
- Thus, the combinatorially explosive state-action space of VM configurations is projected to a polynomial dimensional vector space with a sparse basis.

## Megh: Learn to Migrate As-You-Go

- Megh plugs polynomial size projection space X and incremental update of T to Least Square Policy Iteration algorithm .
- Which is a functional approximation algorithm that implements in an actor-critic framework.
- Megh first tries to find out an estimation of cost to-go function by least-square estimation in the actor format and then to update the policy such that it maximizes the estimate in the critic format

# Megh: Learn to Migrate As-You-Go

**Algorithm 1.**

**function** MEGH($S$, $A$, $\gamma$, $\epsilon$, $Temp_0$)
  Initialize $\delta \leftarrow d, B_0 \leftarrow \frac{1}{\delta}\mathbb{I}_{d\times d}, \phi_0 \leftarrow \mathbf{0}_d,$
  $\theta_0 \leftarrow \mathbf{0}_d, \pi(s_0) \leftarrow \mathbf{0}_d, z_0 \leftarrow \mathbf{0}_d, C_0 \leftarrow 0$
  **while** $t \geq 1$ **do**
    $a_t \leftarrow \arg\max_{a\in A} \quad \pi_t(s_t)$
    Take action $a_t$.
    Observe state $s_{t+1}$.
    $C_{t+1} \leftarrow$ Calculate cost using Equation (6).
    $B_{t+1} = T_{t+1}^{-1}$ update using Equation (10).
    $z_{t+1} \leftarrow z_t + \phi_{a_t} C_{t+1}$
    $\theta_{t+1} \leftarrow B_{t+1} z_{t+1}$
    $\pi(s_{t+1}) \leftarrow PolicyCalculator(\phi_{a_t}, \theta_{t+1})$
  **end while**
**end function**

## 5.1 Inducing Exploration in Action Selection

- Instead of greedily choosing the action with the minimum expected cumulative cost, Boltzmann exploration is used as the on-policy mechanism
- Authors adapt Boltzmann exploration with decreasing temperature.
- Temperature decays with a factor exp (-epsilon)
- As Temp decreases, the policy becomes the greedy selection of the minimum

## 5.2 Managing the Complexity Bottleneck

- Megh has a space complexity of O(d^2) and a time complexity of (d^3).
- Time complexity is reduced to (d^2) by using Gauss-Jordan elimination process to compute the inverse of the operator T to update B
- Initial space complexity is reduced to O(d) by storing only non zero entries of matrix B and vector Phi as a triplet (row number, column number, value)

# 6 Performance Evaluation

- Experiments were performed using CloudSim toolkit
- Assumes cost of  0.18675 USD/kWh
- User has to pay 1.2 USD per hour of using VM instance.
- Cloud providers would pay back 16.7 and 33.3 percent of user's money depending on whether the performance degradation is less than or greater than 0.10 percent.
- Authors consider Beta= 70% as the overloading threshold of the PMs and alpha =30% for the minimum CPU usage threshold by VMs during migration

# 6 Performance Evaluation: Performance

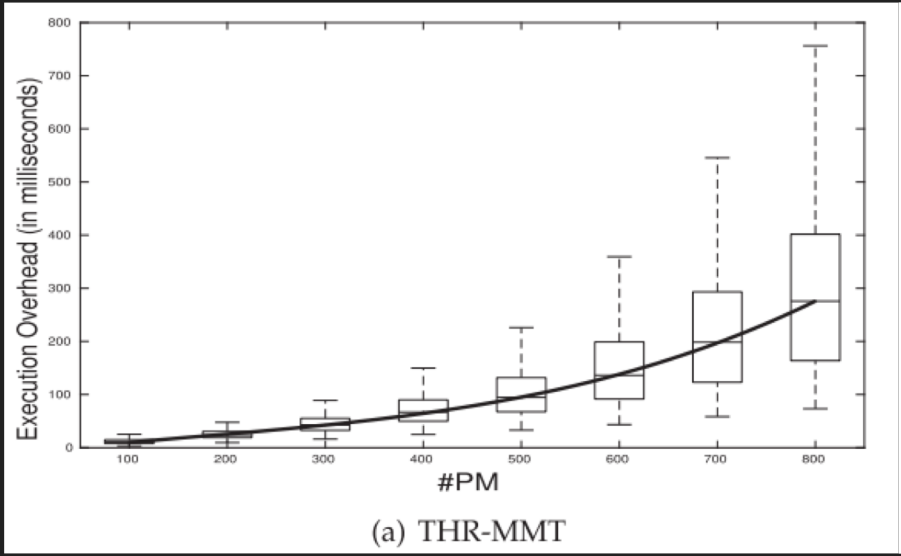
Performance Evaluation for PlanetLab

| Algorithms | THR-MMT | IQR-MMT | MAD-MMT | LR-MMT | LRR-MMT | Megh |
|---|---|---|---|---|---|---|
| Total cost (USD) | 1347 | 1504 | 1367 | 1392 | 1392 | 1155 |
| #VM migrations | 325299 | 444624 | 331304 | 324079 | 324079 | 2309 |
| #Active hosts | 666 | 684 | 682 | 692 | 692 | 203 |
| Execution time (ms) | 2016 | 3077 | 2226 | 1924 | 2080 | 1426 |

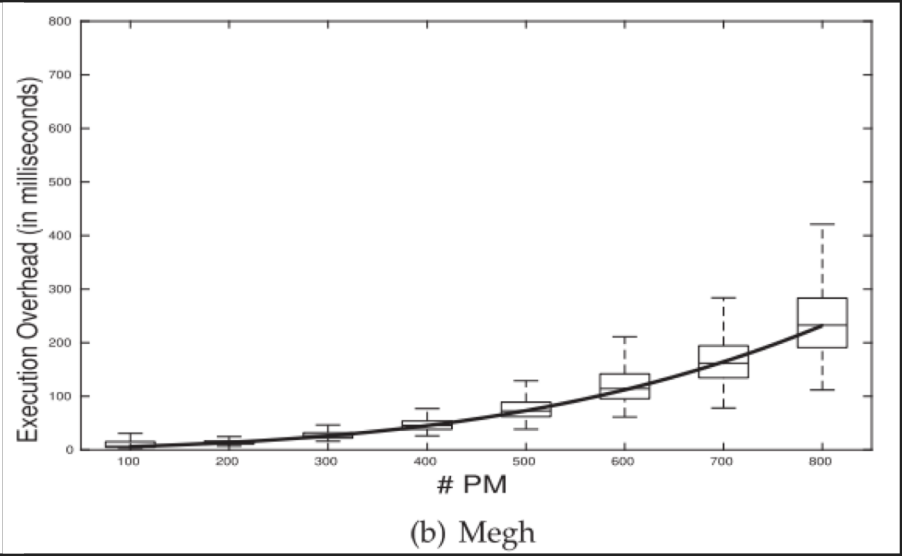
Performance Evaluation for Google Cluster

| Algortihm | THR-MMT | IQR-MMT | MAD-MMT | LR-MMT | LRR-MMT | Megh |
|---|---|---|---|---|---|---|
| Total cost (USD) | 706 | 708 | 708 | 710 | 710 | 688 |
| #VM migrations | 299352 | 262185 | 266706 | 233172 | 233172 | 3104 |
| #Active host | 82 | 72 | 73 | 59 | 59 | 194 |
| Execution time (ms) | 2887 | 4030 | 4000 | 3889 | 3923 | 1945 |

# 6 Performance Evaluation: Scalability

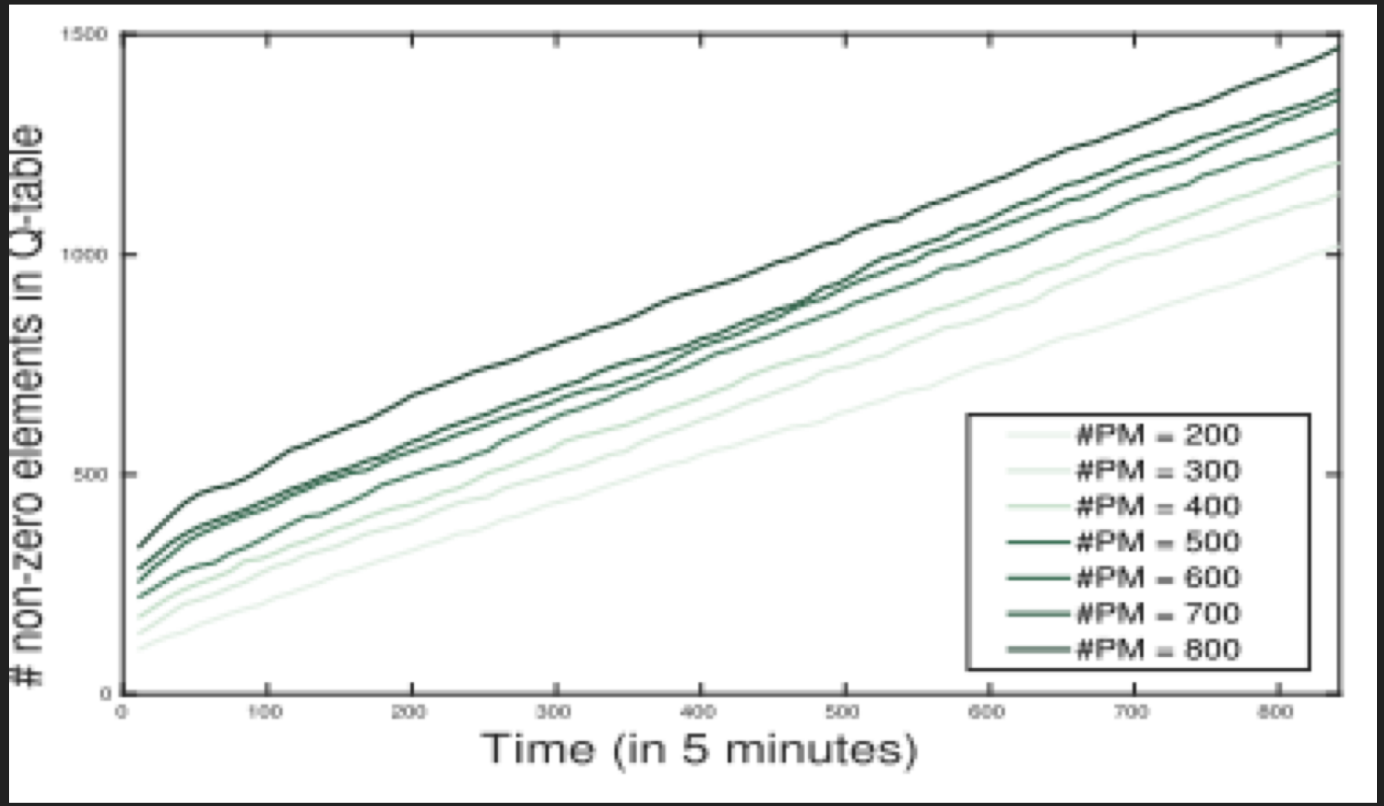



(a) THR-MMT




(b) Megh

# 6 Performance Evaluation: Scalability



# 6 Performance Evaluation: Scalability

- The per-step cost decreases when increasing initial Temperature value from 0.5 to 3.0, but starts to increase with higher values than 3.0
- Per-step cost when changing epsilon was too sporadic to pinpoint a best value.

# Conclusions

This work addresses *when* to migrate VMs, *what* VMs to migrate, and *which* Pm to migrate with respect to energy cost and maintaining QoS in a real time scenario.

Megh is able to work irrespective of application and hardware heterogeneity while learning the uncertain dynamics.

Megh is able to mitigate curse of dimensionality by reducing the state space to a polynomial dimensional state state action space with sparse basis

Megh incurs the smallest cost and least execution overhead with the Google cluster and PlanetLab datasets when compared to state-of-the-art algorithms.

## Future directions

Authors are investigating the opportunity to take advantage of additional knowledge about the workload such as periodicity or a queueing model

Also leverage knowledge of the network topology like fat trees.