Cache Contention Aware Virtual Machine Placement and Migration in Cloud Datacenters

Authors: Liuhua Chen, Haiying Shen and Stephen Platt

Presenter: Haiying Shen

IEEE ICNP

November 8-11, 2016

Singapore

Objective

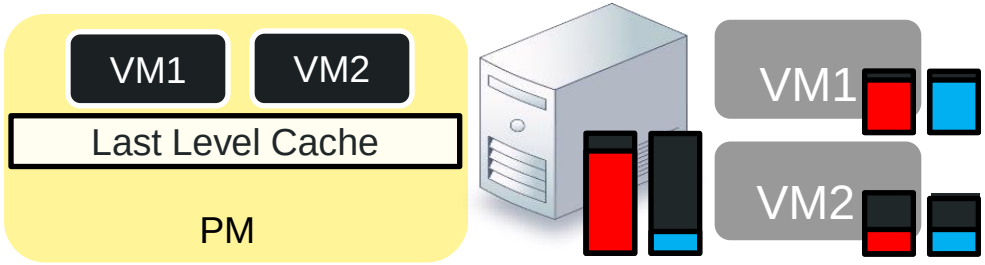
An effective VM allocation algorithm should allocate as many VMs as possible to a PM

i) meeting explicit resource requirements (CPU, memory)

ii) minimizing contentions on Last Level cache

Many previous VM allocation or migration methods provide a metric to choose destination PM and migration VM to handle objective i) but neglect objective ii).

Objective



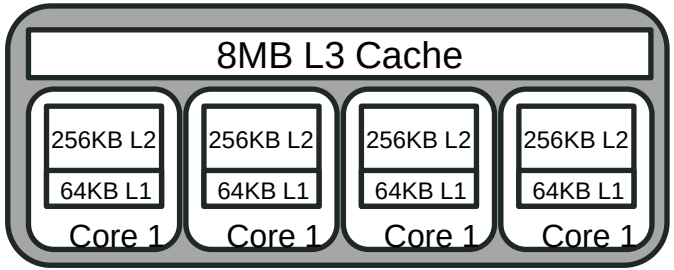
Performance degradation due to shared cache

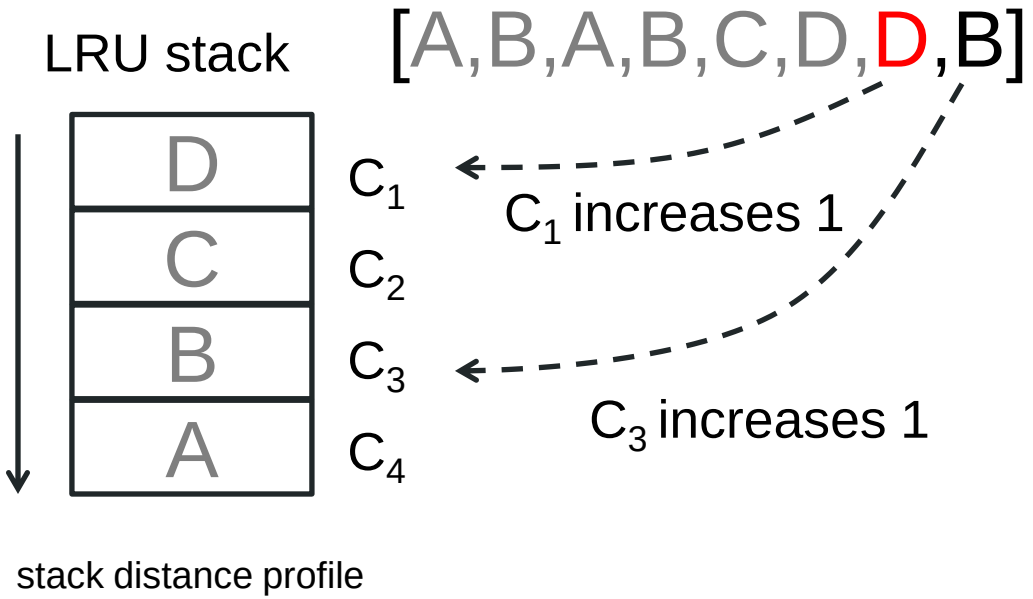
- Reduce cache interference

Overview

- A brief review of cache hierarchy
- VM cache performance degradation prediction
- VM placement and migration algorithm
- Experimental results
- Conclusion with future directions

A brief review of cache hierarchy

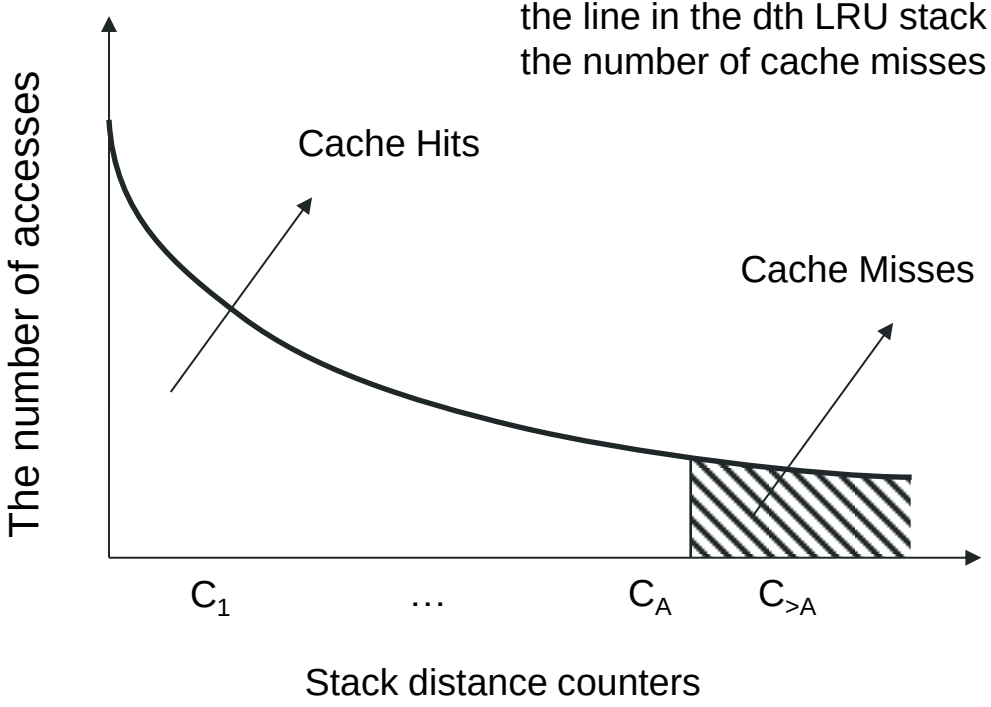




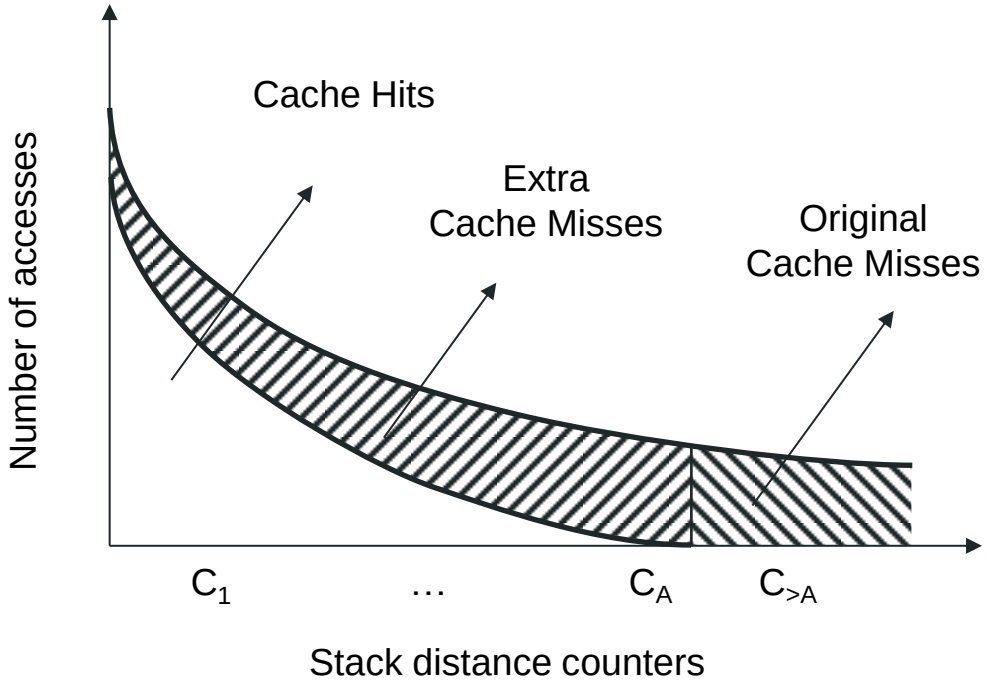
stack distance profile

Stack distance profile

fi = {C_1,C_2,...,C_A,$C_{>A}$}, Cd counts the number of hits to the line in the dth LRU stack position and $C_{>A}$ counts the number of cache misses



Stack distance profile (cont.)



Cache Contention Prediction

When VM i and VM j compete for a cache line (which is the d^{th} position in the LRU stack), the probability of VM i "winning" the competition is proportional to the number of accesses to this cache line of VM i, but reversely proportional to the total number of accesses to this cache line of the two VMs.

$$q_d^{ij} = \frac{C_d^i}{C_d^i + C_d^j}$$

Extend to multiple VMs

$$q_d^i = \frac{C_d^i}{C_d^i + \sum_{k=1}^{N_v} C_d^k}.$$

Cache Contention Prediction (cont.)

The new stack distance profile of VM i can be estimated by

$$f_i' = \{q_1^i C_1^i, q_2^i C_2^i, \ldots, q_A^i C_A^i, C'_{>A}\},$$

where $C'_{>A} = \sum_{d=1}^{A+1} C_d^i - \sum_{d=1}^{A} q_d^i C_d^i$.

Performance Degradation Prediction

Sensitivity is a measure of how much a VM will suffer when cache space is taken away from it due to contention.

$$S_{ij} = \sum_{d=1}^{A} q_d^{ij} \times C_d^i.$$

Intensity is a measure of how much a VM will hurt others by taking away their space in a shared cache.

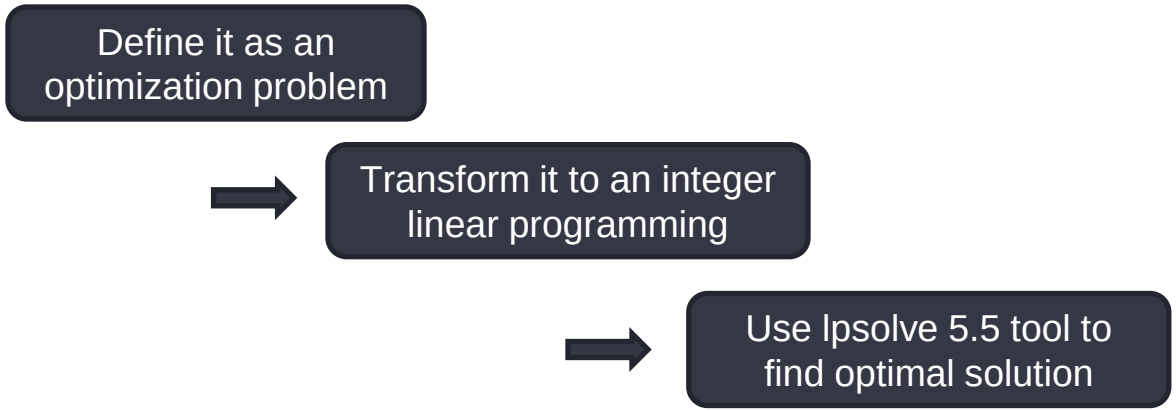
$$I_{ij} = S_{ij} + C'_{>A},$$

The degradation of co-scheduling v_i and v_j together is the sum of the performance degradation of the two VMs

$$P_{ij} = S_{ij} I_{ji} + I_{ij} S_{ji}.$$

VM placement and migration algorithm

Objective: minimize the total pain of the co-location of the new VMs with the existing VMs



VM placement and migration algorithm (cont.)

The computational complexity of the above method is very high, especially for a relatively large number of VMs. We propose a heuristic VM placement and migration algorithm.

- VM placement: allocates each VM to a PM that leads to the minimum total performance degradation.
- VM migration: select a VM which generates the maximum pain with other co-located VMs in the PM to migrate out.

Experimental results

Simulation:

- CloudSim (extended to model LLC contention)
- Use trace to determine profiles
- 1000+ PMs
- 4000 VMs

Real testbed:

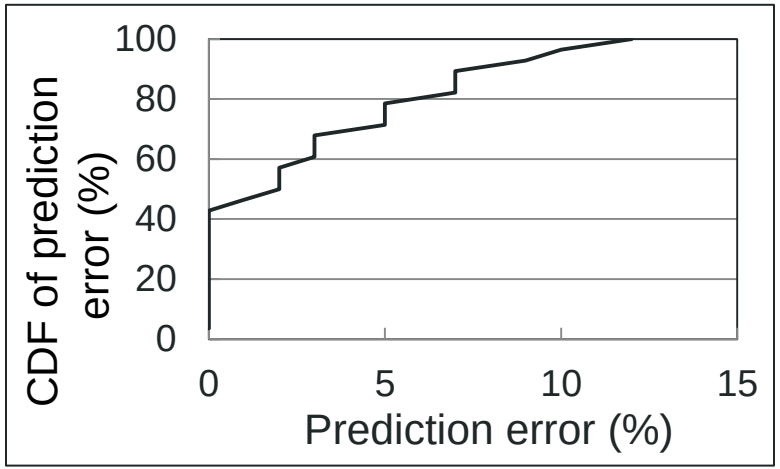
- High-performance computing (HPC) cluster
- Each VM run NPB suite workload

  -- NAS Parallel Benchmark (NPB) suite
- 20 PMs
- 120 VMs

Our algorithm: CacheVM

Comparison algorithms:
cache unaware (Random), classification based (Animal), miss rate based (MissRate)

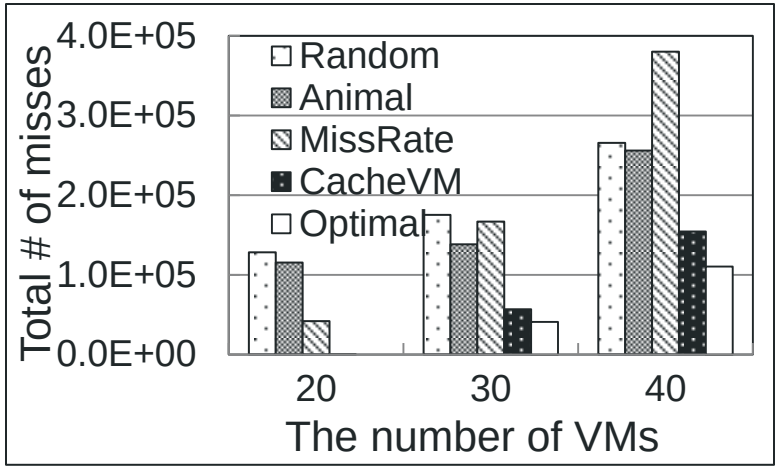
Model validation

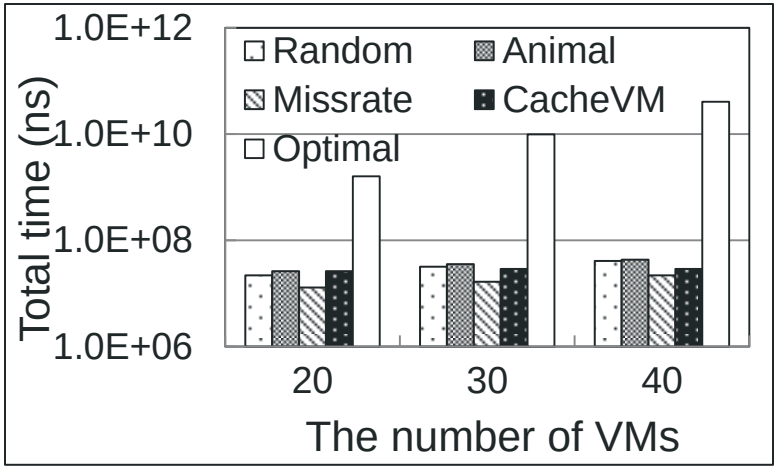


(Cache misses predicted by the model - Cache misses collected by the simulator)/Cache misses collected by the simulator)

This result confirms that the proposed model achieves a high accuracy in predicting cache behaviors.

Comparison with the Optimal Algorithm



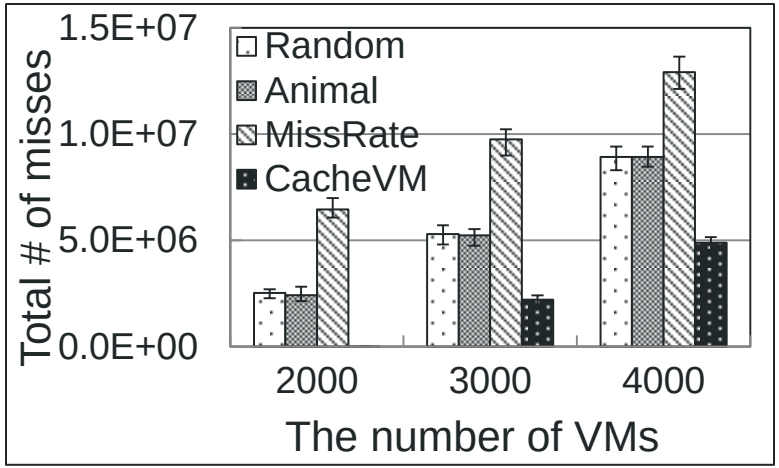


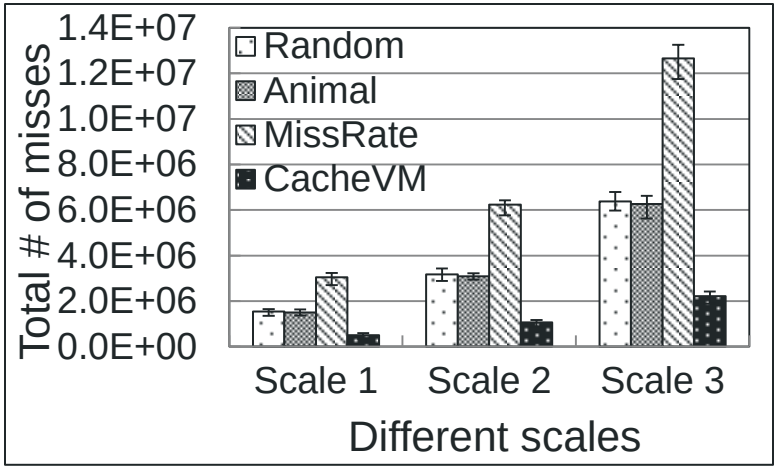
Optimal<CacheVM<Animal<Random
MissRate increases faster
20 PMs

MissRate<CacheVM<Random<Animal<<Optimal

20 PMs

Simulated performance with real trace





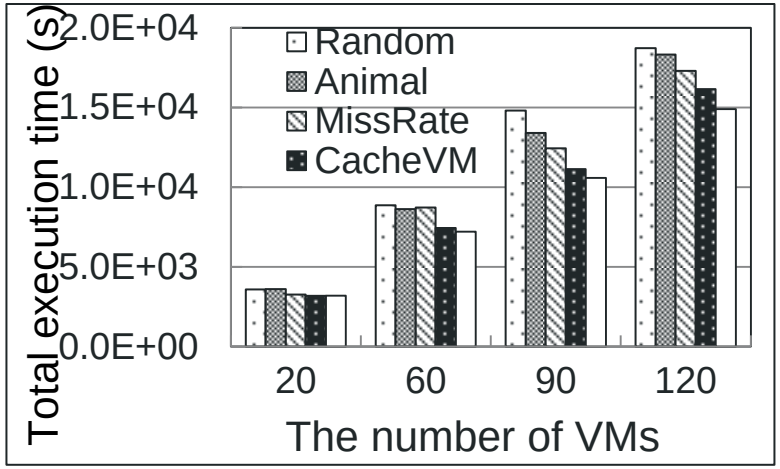
CacheVM<Animal≈Random<MissRate

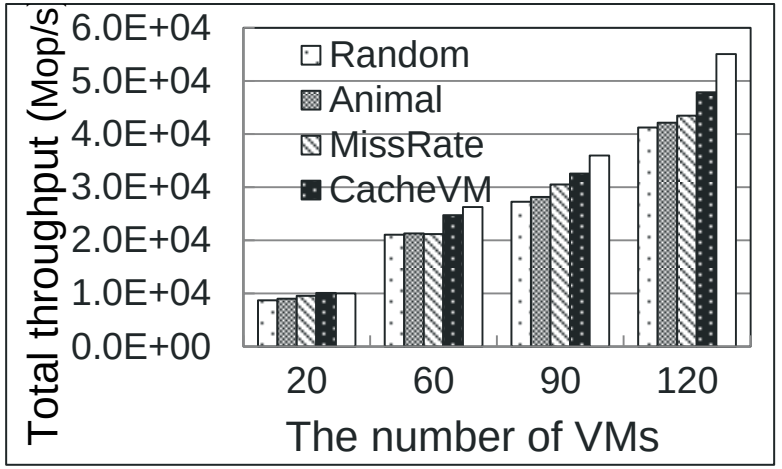
2000 PMs

CacheVM<Animal<Random<MissRate

(1000 VMs, 750 PMs), (2000 VMs, 1500 PMs), (4000 VMs, 3000 PMs)

Performance on real testbed



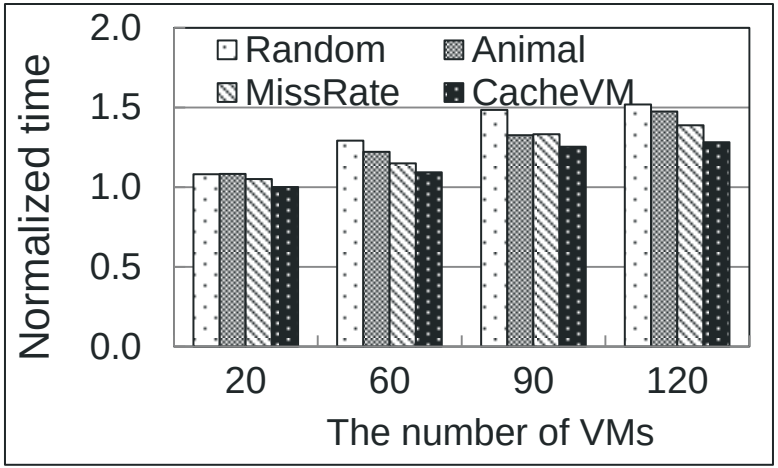


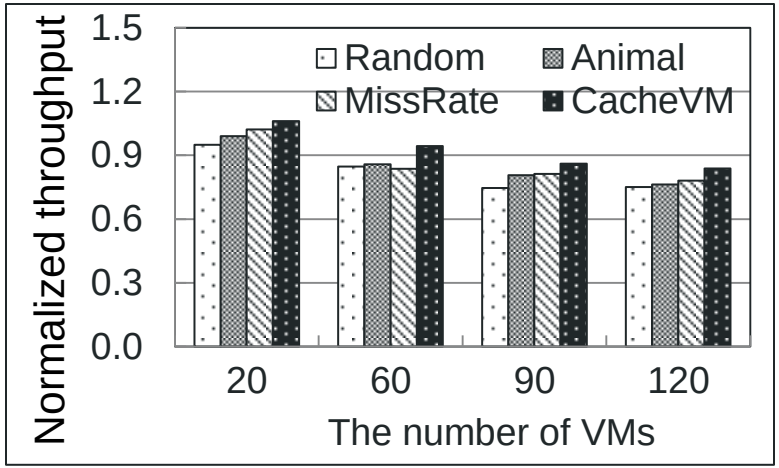
CacheVM<MissRate<Animal<Random

Random<Animal<MissRate<CacheVM

Varied VMs from 20 to 120 and allocated them to 20 PMs

Performance on real testbed (cont.)





CacheVM<Animal<MissRate<Random

Random<Animal<MissRate<CacheVM

Conclusion and future work

- Proposed a cache contention aware VM performance degradation prediction algorithm.
- Formulated a cache contention aware VM placement problem.
- Transformed this problem to an integer linear programming (ILP) model and solved it.
- Proposed a heuristic cache contention aware VM placement and migration algorithm
- Conducted trace-driven simulation and real-testbed experiments to evaluate CacheVM.

Future work: develop a decentralized version of the proposed algorithm.

Thank you!
Questions & Comments?

Haiying Shen

hs6ms@virginia.edu

Pervasive Communication Laboratory

University of Virginia