Memory Sharing Based VM Consolidation [memory buddies]

Deependra Patel 120050032
Jayesh Bageriya 120050022

Introduction

- In a multi-server data center, opportunities for page sharing may be lost because the VMs holding identical pages are resident on different hosts.
- We are trying to exploit Page Sharing for Smart Colocation in Virtualized Data Centers.

Approach

- Generate hash for all pages of all guest VMs
- Send this data to a central control VM which compares it to hash of various other VMs
- Control server makes decisions of live migration using a greedy heuristic approach at various intervals

Co-location Scheme

1. Choose physical machine with highest memory
2. Colocate two VMs with highest sharing on this machine, then we find the VMs with highest sharing potential based on the heuristic and place it until current physical machine is full of memory

$$\text{expected\_sharing(k,m)} \quad \sum_{i=1}^{n} share(i,k)/i \;\; i \in machine(m)$$

3. Choose next physical machine with highest memory, and repeat step 2

Implementation

Memory Tracer

A daemon runs on guest VM. It inserts a kernel module which we go through all the pages of the memory of the VM to calculate 32 bit hash of each page and then write to the hash file. Removes the module. sends file to control server.

Control Node

It listens to the incoming hash files and stores them by running a server. It also analyses the hashes of various VMs for colocation.

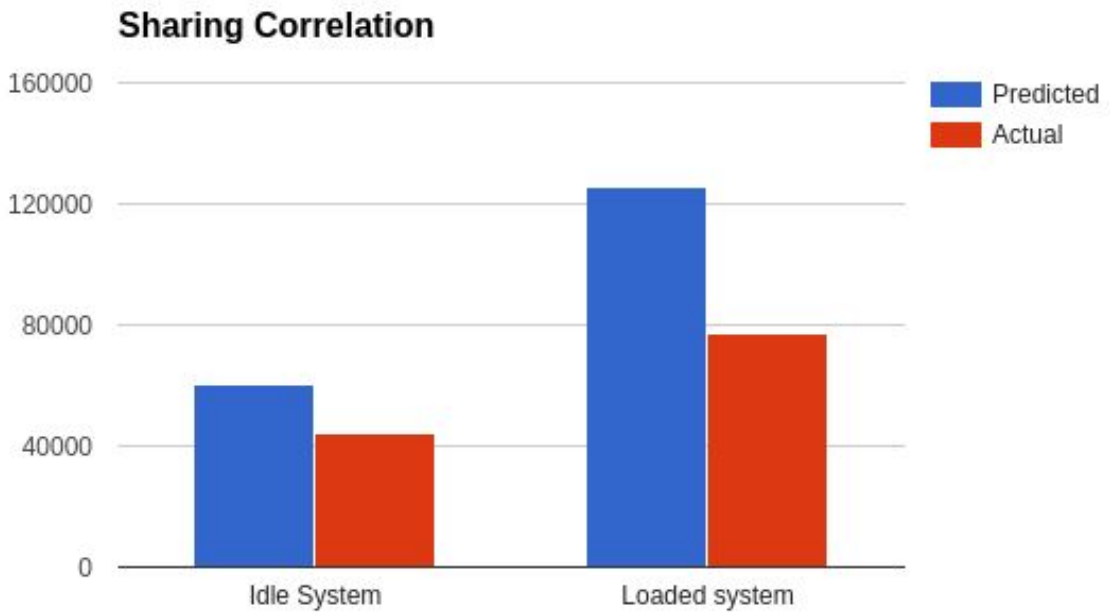
Experimentation

1. **Effect of hashing on performance :** We ran cpu blowfish which gave reduction of score by 18%

2. **Time taken for hashing :** It takes ~2 seconds for generating the hash of entire used memory in a VM. The file generated is ~4Mb in size, so ~3 seconds is taken in sending the file to the control server. Most of the time is spent in writing to the hash file.

3. **Change of fingerprints with time :** The change in idle system is around 6% every two mins and that in loaded system is around 25% ~ 30%.

4.Correctness of the fingerprint generated



5. Testing co-location heuristic

	Initial allocation	Output
VM0 (1GB)	M0 (3GB)	M1
VM1 (2GB)	M0	M1
VM2 (1GB)	M1 (3GB)	M0
VM3 (2GB)	M1	M2
VM4 (1GB)	M2 (3GB)	M0

Common hashes

	VM1	VM2	VM3	VM4
VM0	254 MB	235 MB	33 MB	207 MB
VM1		121 MB	33 MB	114 MB
VM2			12 MB	339 MB
VM3				10 MB

	Before	After colocation
M0	1536 MB	1331 MB
M1	2357 MB	1536 MB
M2	857 MB	1556 MB
Total	4750 MB	4423 MB

Memory Saved : 327 MB

Future Work/Improvements

- Use Bloom filter instead of super fast hash to reduce network usage and cpu utilization
- Add functionality to actually migrate VMs using our co-location heuristic
- Calculate hashes inside KVM will not require modifying the OS