Black-box and Gray-box Strategies for Virtual Machine Migration

Complete Review

good review, good catch of flaws. 10/10

Reviewed by

CIJO GEORGE
M.Sc (Engg.), SERC
S.R No. : 5510-210-101-07980
Email: cijo@rishi.serc.iisc.ernet.in
Section-wise summaries

Abstract

- Sandpiper – A system to automate migration of virtual machines to eliminate 'Hotspots'.

1. Introduction

- Motivation for Sandpiper - Manually initiated migration may not be able to respond effectively to sudden workload changes. Also, it is error-prone and complex.
- Sandpiper tracks usage of resources, detect hotspots, determines a new optimal physical resources to virtual server mapping, and initiate the migration.
- Sandpiper works on top of Xen.

Two approaches:

Black-box
- Without any knowledge about the applications within a VM or any other OS level information.

Gray-box
- Using some OS level information.

2. Background and System Overview

Assumptions
- Hardware configuration of each server is known to Sandpiper.
- Each virtual server is allocated with a slice of physical resources.
  - CPU: Weights allotted to virtual servers and Xen CPU scheduler allocates CPU accordingly.
  - Network Interface : Best-effort FIFO scheduler used by Xen
  - RAM – a certain amount is allotted to each VM
  - Storage – Network File System

Architecture

Nucleus
- Runs in Domain 0 of Xen
- Responsible to record resource usage in that server

Monitoring Engine
- Gathers CPU, network interface & memory swap information of each VM for the Nucleus.
- In case of Gray-box approach, a daemon runs in each VM to collect OS level statistics.

Control Plane
- Runs on a separate node.
- Nucleus relays the collected statistics to Control Plane.
- Comprises of Profiling Engine, Hotspot Detector and Migration Manager.
Profiling Engine
- Constructs resource usage profiles for each Virtual Server and for a Physical server as a whole, based on information from the Nucleus.

Hotspot Detector
- A 'Hotspot' is said to have occurred if the usage of a resource exceeds a threshold or if SLA violations are observed for a consistent period of time.
- Hotspot Detector detects hotspots based on the usage profiles constructed by the Profiling Engine.
- Signals the Migration Manager if a hotspot is detected.

Migration Manager
- Determines what Virtual Servers to migrate, where to move them and how much resources should be allocated once migration is complete and initiates the migration.

3. Monitoring and Profiling in Sandpiper

3.1 Unobtrusive Black-box Monitoring

CPU Monitoring
- The number of network I/O requests by each VM is monitored & CPU utilization of Domain 0 is split between the VM's based on that (Assuming that CPU usage of Sandpiper components is negligible).

Network Monitoring
- /proc is used to monitor the number of bytes send to each virtual network interface.

Memory Monitoring:
- Memory usage is inferred from the number of read/write requests to swap partitions.

3.2 Gray-box Monitoring

- Daemon uses /proc directory for OS level information & reads application logs to detect SLA violations.

3.3 Profile Generation

- Additional Gray-box profiles: Memory utilization, service time, request drop rate, incoming request rate.
- Each profile has a Distribution profile & Time Series.
- Distribution profile
  - Variations in resource usage.
  - Used by Migration Manager to determine peak resource requirements.
- Time Series
  - All observations listed.
  - Used for hotspot detection.
4. Hotspot Detection

– Black-box approach
  – If aggregate CPU, network utilization or swap activity of a physical server exceeds some threshold for a sustained time.

– Gray-box approach
  – If memory utilization of VM exceeds threshold or response time or request drop rate doesn't satisfy SLA for a sustained time.

Sustained time: If at least k out of the n most recent observations as well as the next predicted value exceed a threshold.

5. Resource Provisioning

5.1 Black-box Provisioning

– Estimating peak CPU & Network bandwidth needs:
  – A high percentile of observed distribution is used as the first approximation, assuming all VMs are not using their full allocation.
  – If CPU or Network Interface in the physical server are close to saturation, a constant is added to this value.

– Estimating peak memory needs:
  – Memory is increased by a constant amount if swap activity exceeds threshold.

5.2 Gray-box Provisioning

– Estimating peak CPU needs:
  – CPU capacity is scaled up by a factor of (peak work load / current capacity of the VM)

– Estimating peak Network needs:
  – Peak Bandwidth usage = peak arrival rate X Data transferred to service peak load.

6. Hotspot Migration

– Determining a new mapping avoiding threshold violations is an NP-hard problem.

Capturing Multi-dimensional loads

– Vol. of PM or VM = CPU load X Network load X Memory load.

Migration Phase

– All moves are determined. But actual migrations take place later on.

– Algorithm
  – Physical servers sorted in decreasing order of Volume.
  – VMs sorted in decreasing order of Volume to Size Ratio (VSR).
  – VM with highest VSR in the Physical Server with highest Vol. is tried to fit into the Physical Server with lowest Vol.
– If this is not possible, Physical Server with second lowest Vol. is considered. This continues till a match is found.
– If a match is not found, the VM with second highest VSR is tried to move in a similar fashion. This process continues, until the Physical Server resource utilization falls under threshold.
– Now the process is repeated for the Physical Server with second highest Vol. and so on.

Swap Phase

– If sufficient resources are not available in any of the less loaded Physical servers for VM migration, then high VSR VMs in high Vol. Physical servers are tried to swap with low VSR VMs in low Vol. Physical Servers.
– Actual swapping is triggered only after a list of required swaps is made.

7. Implementation and Evaluation

– Implemented in Python.
– Monitoring engine reports measurements every 10 seconds.
– Profiling engine uses the past 200 measurements.
– Default Threshold = 75%
– Hotspot triggered when 3 out of past 5 readings + the next predicted reading exceeds threshold.
– Gray-box monitoring:
  – Linux Daemon: gather memory statistics
  – Apache module: process log entries

7.1. Migration Effectiveness

– Sandpiper tries to reduce maximum load with minimum data transfer, by migrating VM with lower memory footprint in case of identical large workloads in more than one VM.

7.2. Virtual Machine Swaps

– RAM in control node is used as scratch space for swap.
– Swaps result is more overhead.

7.3. Mixed Resource Workloads

– Sandpiper can put CPU, Network and Memory intensive VMs together in one PM to balance resource usage.

7.4. Gray v. Black: Memory Allocation

– Gray-box approach clearly performs better than Black-box approach in case of memory intensive applications.

7.5. Gray v. Black: Apache Performance

– Gray-box approach is able to mitigate hotspots quicker than Black-box and when VM s
are overloaded. The response-time of applications during hotspot mitigation is also lesser in case of Gray-box.

7.6. Prototype Datacenter Evaluation

- Performance is found to be acceptable in real-life scenarios too.

7.7. System Overhead and Scalability

- Gray-box: there may be significant overhead depending on the size of application logs.
- Nucleus overhead: negligible.
- Control Plane Scalability: Calculating new mapping of resources after hotspot detection is computation intensive and can create significant overhead in large datacenters.

7.8. Stability During Overloads

- Predicted future trend in resource utilization of a PM is also considered while deciding to which PM a VM should be moved.

7.9. Tuning Sandpiper

- Measurement interval, Threshold values and Condition to flag hotspots can be tuned for best performance.

8. Related Work

- Mentions some of the other works related to migration.

9. Conclusions and Future Work

- Future work: Support for replicated services.

What I did not understand

- Sandpiper makes use of the metric $Vol. = CPU\ load \times Network\ load \times Memory\ load$ in the algorithm for deciding the VM to be migrated. Consider a PM having VM1, VM2 & VM3 with utilization as follows:

VM1: CPU = 80%, Network = 40%, Memory = 1%
VM2: CPU = 80%, Network = 40%, Memory = 1%
VM3: CPU = 1%, Network = 70%, Memory = 70%

Suppose threshold value is 75% for all resources. Here, hotspot will be detected for VM1 & VM2. But according to Sandpiper algorithm, VM3 will be migrated, since it has the maximum Volume. But moving VM3 will not make any significant improvement since it uses only 1% CPU. Isn't this wrong judgment?
What happens to the usage profile of a VM once it is migrated to another PM? Is it discarded and a new profile generated?

For swap, sandpiper uses RAM in the control node. Does it mean that the VM which is temporarily migrated to the control node continues to run there in the control node until it is finally migrated to the destination? If so, who monitors the CPU, Network, Memory utilization of this VM during this temporary period?

What I liked

- The paper has covered almost all aspects of VM Migration. It has even explained the problem of resource provisioning after migration, so as to meet peak needs of the VMs.
- The paper has taken into consideration the fact that some users wouldn't want to share OS level information, and hence come up with a Black-box solution for such users.

What I did not like

- Explanation for resource provisioning in Gray-box approach, using queuing theory is not clear.
- The algorithm explained in section 6 (Hotspot mitigation) used Volume & VSR to determine which VM to be migrated and to where. But in most of the subsections of section 7 (Implementation & Evaluation), explanation is not given in terms of Volume & VSR.

What is missing

- It is not explained clearly how Sandpiper can be used in real world datacenters with thousands of servers where calculation of new mapping of resources would create a significant overhead.

If I had to extend the work

- Performance of Sandpiper in datacenters with thousands of servers should be studied.
- A proper method to efficiently compute the new mapping of resources in very large datacenters should be developed, so that the overhead can be reduced.

Conclusion

The paper gives a feasible solution for efficiently automating virtual machine migration in small and medium datacenters. But more work has to be done to make this solution suitable for commercial cloud infrastructures with very large datacenters.
<table>
<thead>
<tr>
<th>Date</th>
<th>Start Time</th>
<th>Stop Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>23rd Aug</td>
<td>21:05</td>
<td>23:45</td>
</tr>
<tr>
<td>24th Aug</td>
<td>09:00</td>
<td>09:35</td>
</tr>
</tbody>
</table>

**Partial Review 2**

<table>
<thead>
<tr>
<th>Date</th>
<th>Start Time</th>
<th>Stop Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>26th Aug</td>
<td>16:45</td>
<td>19:10</td>
</tr>
<tr>
<td>26th Aug</td>
<td>21:00</td>
<td>23:15</td>
</tr>
</tbody>
</table>