Autonomic Virtualized Environments

Daniel A. Menascé and Mohamed N. Bennani
Dept. of Computer Science,
MS 4A5, George Mason University Fairfax, VA 22030, USA
{menasce,mbennani}@cs.gmu.edu

PUBLISHED IN: INTERNATIONAL CONFERENCE ON AUTONOMIC AND AUTONOMOUS SYSTEMS ICAS06 (2006)
VOLUME: 00, ISSUE: C, PUBLISHER: IEEE, PAGES: 28-28
ISBN: 0769526535
DOI: 10.1109/ICAS.2006.13

Synopsis by:
Stephen Roberts, GMU CS 895, Spring 2013

Bottom Line

- Challenge:
  - With the increasing growth in usage of virtualization technologies for computing resources; Global management of these resources can be taxing and/or inefficient especially as the environments continue to grow in number and complexity.
  - With a given workload, then some VMs may be less utilized while others may be over-utilized thus not effectively or efficiently utilizing available resources. Result is a potential negative impact on users and operational SLAs
- Solution:
  - Implement autonomic computing mechanisms in order to dynamically allocate VM hardware resources more equitably toward those VMs with higher workloads in order to optimize resource loading using one of the following methods:
    - Dynamic priority setting
    - Dynamic allocation of CPU shares
- Results:
  - Test using simulation software showed that the dynamic priority setting method had a less granular method of differentiatiing between VMs while the Dynamic allocation of CPU method allowed a finer grain control of resource allocation.

Metrics

- ***Input workload***
- ***CPU allocation per VM***
- Number of VMs per virtualized environment
- Workload classes
- Workload SLA for given workload class

Problem Details

- Resource allocation problem : how to dynamically allocate CPU among VMs with goal of maximizing the global utility function Ug.
  - Global utility function, Ug, of the entire virtualized environment is a function of the utility functions of each VM. Ug = h(U1, ⋯,UM). (1)
  - Utility function, Ui,s, for class "s" at VM i is defined as the relative deviation of the response time Ri,s of that class with respect to its service level agreement (SLA), βi,s. Ui,s = (βi,s −Ri,s) / (max{βi,s,Ri,s}). (2)

Priority Based vice CPU Share Allocation

- Priority:
  - All the same VM workloads have same priority
  - Open QN Model solved incrementally in P steps, one per priority class, from highest of 1 to lowest of p
  - Each shadow CPU is dedicated to all workloads that have the same priority – at step p has p shadow CPUs instead of one
- CPU Share:
  - M shadow CPUs, one per VM
  - CPU shares adjusted to account for the share allocation of each VM

Feasibility

- Testing and hypothesis appeared sound
    - However would question assumption of need for virtual environments in light of VMware DRS capabilities being available at time of test (at least in enterprise implementations)
- Should look at other resource allocation (RAM, Disk, Network) models in conjunction with CPU allocation to get a better sense of optimization potentials
- Concern that operationally dynamic load may skew results of simulation
- Concern that autonomic resource allocation may cause thrashing
    - if sample rate of input workload does not accurately reflect prioritized work to be performed
    - premature starvation of a prioritized workload (CPU allocation away from a higher priority workload)

What is Missing

- Competing workloads that can more than utilize full CPU shows the process can work ie the 2 workloads used in experiment were basically 180 degrees out of phase – However:
  - Would results hold with fully random and dynamic workloads? Ie:
    - A Virtual PC environment
    - International E-Commerce site (ie E-Bay/Amazon)
    - Typical server loads are in 10% range 90% of time with peak loading over the remaining 10% of the time
  - Would advantages be diminished if allocation scheme causes premature re-allocation? How to test for pre-mature or inefficient re-allocation?
  - VMWare specifically has several dynamic resource management components:
    - vSphere (part of ESX Server) and previously vCenter had Dynamic Resource Scheduling (DRS) capabilities; initial VMware ESX server implementations had DRS "like" capabilities which became more robust over time
    - How would autonomic approach compare with current VMware DRS offerings?
  - While CPU allocation is important it is not the only resource to manage in a virtualized environment: RAM, Network bandwidth and Disk to name a few often have more impact on workload performance than CPU allocation especially in a virtualized environment

What I learned

- Autonomic Computing principles applied to a concrete example (virtualization)
- Math and science behind resource re-allocation
- Still consuming math behind model…

Future Work/Research

- Modifying Autonomic Computing approach to handle RAM, Networking and Disk utilization along with CPU
- Incorporating an incoming workload assessment process to increase efficiency of allocations (i.e. measure what workload needs may be prior to actually running)
- Rerun test with a much more dynamic workload input to re-evaulate approaches
- Apply Autonomic Approach to security components: **
  - Router ACL
  - Firewall Rules
  - Intrusion Detection/Prevention Rules

** looking to make a focus for my papers/research