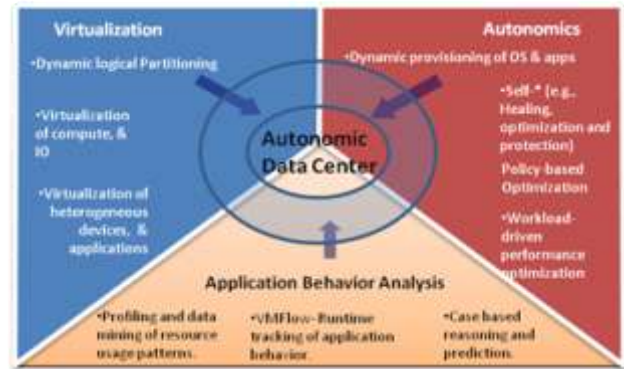


Autonomic Management of Virtualized Data Centers

by Ishtiaq Hossan, Youssif Al-Nashif, and Salim Hariri

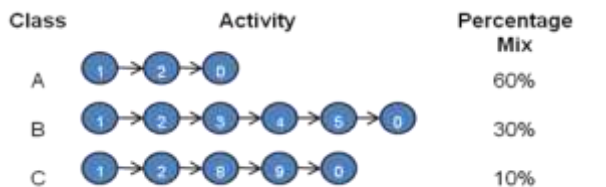
Current datacenter deployments have many inefficiencies such as: (i) Provisioning for peak loads and sometimes for multiple peak loads resulting in low average resources utilization (< 25%); (ii) Lack of autonomic features such as the inability to self-optimize when runtime workloads vary, inability to self-configure when resources are added/removed and inability to self-heal when failures happen; and (iii) Ad-hoc management resulting in considerable overhead on its Total Cost of Ownership (TCO). The goal of this project is to mitigate some of these inefficiencies by using Autonomic Programming with state of the art Virtualization technology.



Autonomic Programming (AP) is a paradigm that supports dynamic changes of the application solution to optimize its performance at runtime. In AP approach, the application execution state is periodically monitored and analyzed to identify its current execution phase/state. Depending on the phase transition, the system can proactively adjust its operating conditions to satisfy future requirements.

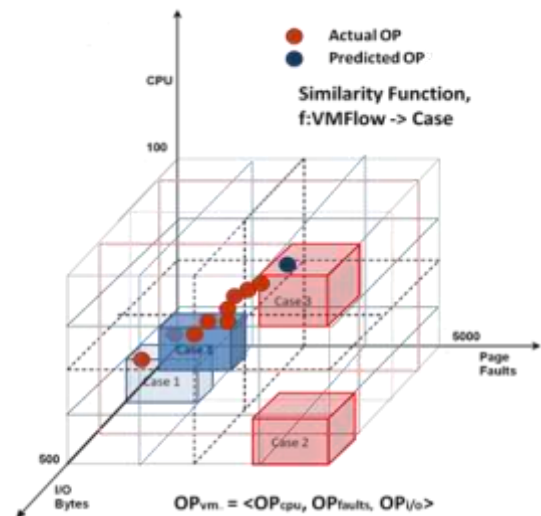
In this project, we determine the application execution phases by exploiting knowledge about application behavior. Each phase is then mapped to an appropriate resource profile using case-based reasoning. Once the profile is determined, the Virtual Machine (VM) running the application can be scaled accordingly.

An example: SPECweb2005 Banking Workload is a three-tier web application that supports 16 different types of client requests. In order to characterize the application's behavior, we classify the sequence of user activities into n classes (Figure 1), and observe the system response to various mixes of user activities.



- | | |
|--------------------|-------------------|
| Request Types: | 6. Profile |
| Login | 7. Change Profile |
| 2. Account Summary | 8. Transfer |
| Bill Pay | 9. Post Transfer |
| Quick Pay | 0. Logout |
| Bill Pay Status | |

Figure 1: Classes of user activity and mix.



In order to efficiently characterize the resource utilization patterns of the phases of an application, we use *VMFlow*. *VMFlow* is an n -dimensional array of features capturing temporal variability and spatial variability for applications running in a VM. It characterizes the dynamic behavior of applications and the VM simultaneously

with respect to key features (e.g. CPU, Page Faults, I/O bytes etc.) in the virtual environment. Figure 3 shows the *VMFlow* corresponding to different mixes of the user activity classes.

Currently we are working on building a Hierarchical Management Framework for managing a Virtual Cluster. The framework is built using Autonomic Components.

Sensors: Periodically collects information from the managed components.

Observers: Apply Clustering and Learning Algorithms on *VMFlow* and *NodeFlow* databases to predict future application phases.

Controller: Collects information from multiple Observers and take actions using previous knowledge and Case-Based Reasoning.

Actuator: Carries out actions (e.g. scaling VMs within a node, migrating VMs to less utilized nodes, shutting down unutilized nodes to save power etc.) as directed by the Controller.

The focus of this work is to improve data center resource management by optimizing the application and virtualization layer. In future we plan to integrate our work with similar optimization work in other layers (e.g. environment layer) to build a cross-layer management framework.

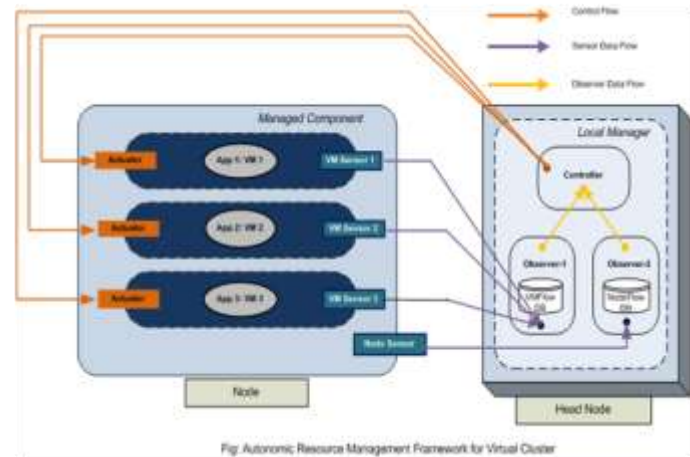


Fig. Autonomic Resource Management Framework for Virtual Cluster