IPHV7I20035X

International Journal Of Advanced Research and Innovation -Vol.7, Issue .II

ISSN Online: 2319 – 9253

Print: 2319 – 9245

Autonomic Virtual Resource Management for Service Workload Management Applying Nefeli

#1 G.Lavanya - M.Tech Pursuing,

#2 M. Vazralu-Associate .Professor,

Department of Computer Science & Engineering,

Malla Reddy College Of Engineering & Technology, Hyderabad.

Abstract: *Cloud platforms host several independent applications on a shared resource pool with the ability to allocate computing power to applications on a per-demand basis. The use of server virtualization techniques for such platforms provides great flexibility with the ability to consolidate several virtual machines on the same physical server, to resize a virtual machine capacity and to migrate virtual machine across physical servers. A key challenge for cloud providers is to automate the management of virtual servers while taking into accounts both high-level QoS requirements of hosted applications and resource management costs. an autonomic resource manager to control the virtualized environment which decouples the provisioning of resources from the dynamic placement of virtual machines. Nefeli, a virtual infrastructure gateway that lifts this restriction. Through Nefeli, cloud consumers provide deployment hints on the possible mapping of VMs to physical nodes. Such hints include the collocation and anti-collocation of VMs, the existence of potential performance bottlenecks, the presence of underlying hardware features (e.g., high availability).*

Keywords*—Iaas clouds, Cloud Computing, Nefeli , Virtual Machine.*

I.INTRODUCTION

Infrastructure as a Service is the most straightforward of the four models for delivering cloud services. IaaS is the virtual delivery of computing resources in the form of hardware, networking, and storage services. It may also include the delivery of operating systems and virtualization technology to manage the resources. Rather than buying and installing the required resources in their own data center, companies rent these resources as needed. In this paper, we present the design, implementation, and evaluation of a cloud gateway, Nefeli. A cloud gateway is a software or hardware networking device that provides connectivity and protocol translation services between a cloud storage service provider and local customer application.It is implemented on a local machine or application to facilitate data transfer between incompatible protocols, security and compression services. Nefeli performs intelligent placement of VMs onto physical nodes by exploiting user-provided deployment hints.

Hints realize placement preferences based on knowledge only the cloud consumer has regarding the intended usage of the requested VMs. virtual machine is a software implementation of a machine (i.e. a computer) that executes programs like a physical machine. Virtual machines are separated into two major classifications, based on their use and degree of correspondence to any real machine. system virtual machine provides a complete system platform which supports the execution of a complete operating system. These usually emulate an existing architecture, and are built with the purpose of either providing a platform to run programs where the real hardware is not available for use (for example, executing software on otherwise obsolete platforms), or of having multiple instances of virtual machines leading to more efficient use of computing resources, both in terms of energy consumption and cost effectiveness or both.

Process virtual machine also, language virtual machine is designed to run a single program, which means that it supports a single process. Such virtual machines are usually closely suited to one or more programming languages and built with the purpose of providing program portability and flexibility amongst other things. An essential characteristic of a virtual machine is that the software running inside is limited to the resources and abstractions provided by the virtual machine it cannot break out of its virtual environment.

By modeling workloads as patterns of data flows, computations, control/synchronization points, and necessary network connections, users can identify favorable VM layouts. These layouts translate to deployment hints. Such hints articulate resource consumption patterns among VMs; VMs that may become a performance bottleneck; and portions of the requested virtual infrastructure that can be assisted by the existence of special hardware support. For instance, the fact that two VMs in a virtual infrastructure will hold mirrors of a database is only known to the cloud consumer. This information should be communicated to the cloud as a deployment hint so that the respective VMs will not be deployed on the same host.

Virtual Machine refer to VM layout patterns as task-flows to distinguish them from the traditional workflow concept. Specifically, task-flows illustrate ideal deployments of VMs described by the cloud consumers using deployment hints. Nefeli exploits these hints so as to (re)deploy VMs in the cloud and achieve efficient task-flow execution. main contribution of our approach is that we present a complete solution in extracting and exploiting the knowledge cloud consumer posses regarding the operational aspects of their virtual infrastructures. Our approach is compatible with the cloud abstractions that dictate users are kept agnostic of the physical infrastructure properties at all times.

Furthermore, our approach is able to adapt to dynamic environments where both task-flows and user preferences change over time. Nefeli produces suitable VM to physical node mappings in response to signals coming from the infrastructures both physical and virtual or any other external notification mechanism. The produced mappings are applied through appropriate VM placement calls to an underlying cloud middleware. In this project consistently display significant performance improvements when compared to the aforementioned policies. In video transcoding, Nefeli achieves 17 percent reduced processing times compared to the VM placement decided by Open Nebula.

In scientific task flows and for a variety of simulated clouds, Nefeli demonstrates significantly higher throughput rates compared to other VM placement policies. Noteworthy savings in terms of power consumption are reported as well. We also present the performance overheads involved in the operation of Nefeli as the clsoud infrastructure scales out and created a detailed prototype and experimented with both simulated and real cloud environments. We compare Nefeli VM-placement against: random placement, a placement that evenly distributes VMs among physical nodes, a policy that minimizes the number of physical nodes used and, thus, reduces the power footprint of the cloud, and the match making policy used by the open-source cloud middleware OpenNebula v.1.2.0.

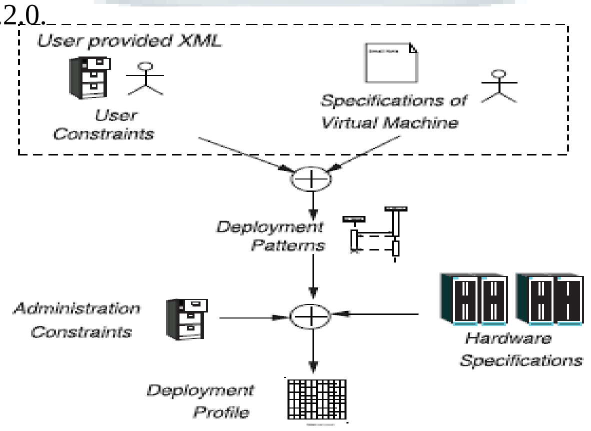


Fig:1 Nefeli's operational model.

Our approach consistently displays significant performance improvements when compared to the aforementioned policies. In video transcoding, Nefeli achieves 17 percent reduced processing times compared to the VM placement decided by OpenNebula. In scientific task flows and for a variety of simulated clouds, Nefeli demonstrates significantly higher throughput rates compared to other VM placement policies. Noteworthy savings in terms of power consumption are reported as well. We also present the performance overheads involved in the operation of Nefeli as the cloud infrastructure scales out.

Consumers may communicate the task-flow information in the form of hints. The latter could be used while trying to appropriately deploy VMs. For instance, consider a user who requests a VM that will play the role of a single network bridge between her virtual infrastructure and the Internet. This bridge inherently becomes a single point of failure and a potential performance bottleneck. Therefore, the VM in question would be best placed on an offloaded physical node equipped with redundant hardware. VMs that are to perform parallel jobs very much in the Map Reduce fashion should be spread across different nodes. Hence, it is important for the cloud to be aware of the user's intended use of particular VMs and also emphasize that IaaS consumers have no explicit control over VM migrations. Migrations reshuffle the way VMs share the same computing nodes so they may radically hurt or significantly enhance the virtual infrastructure's performance. The actual placement of VMs on physical hosting nodes should be able to address the needs of changing workloads.

II. PROPOSED SYSTEM

Based on the existing, Distributed resource algorithm Resource Management in Distributed Environment is concerned with a system in which the main aim is to make sure that a user/client can access the remote resources with as much ease as it can access the local resources. The basis of Resource Management in the distributed system is also resource sharing. The main goal of this paper is to ease the sharing of resources among users. We have implemented a Peer to Peer system where files are our prime resources. In proposed we use two algorithms namely High availability utility algorithm and Simulated Annaeling profile production.

Algorithm 1. HighAvail Utility Function

```
Input: VM_ID: ID of the VM to deploy in a high availability node
MðÞ: Deployment profile function
Output: Satisfaction degree of constraint 1:
host ID :¼ M(VM_ID)
2: satisfaction :¼ 0;
3: if (HostIsHAServer(host_ID)) then 4:
satisfaction :¼ 1:0;
5: return satisfaction; 6:
end if
7: if (HostHasRAID(host_ID)) then 8:
satisfaction þ ¼ 0:2;
9: end if
10: if (HostHasRedundantPowerSupply(host_ID)) then
11: satisfaction þ ¼ 0:2;
12: end if
13: if (HostHasMultipleNetwork(host_ID)) then
14: satisfaction þ ¼ 0:2;
15: end if
16: return satisfaction;
```

The implementation details of the HighAvail constraint are not revealed to the consumer as they are specific to each cloud infrastructure. Whenever one or more assumptions regarding the high availability of the hardware are outdated, possibly due to major changes in the infrastructure (e.g., all hosting nodes become equipped with RAID),we have to provide new implementations for the affected utility functions.

Administrative constraints are also realized as utility functions. These constraints serve a dual purpose as they can introduce high-level policies and assist in administration tasks. For instance, EmptyNode relieves a hosting node of VMs. ReduceDist enforces the high-level policy of clustering VMs of the same user on hosts that may not be far apart in terms of network hops; this is done to limit major traffic to be routed over long-haul physical networks. Wehave realized both user and administration constraints as utility functions in similar fashion to Algorithm 1;for brevity, we omit their detailed discussion here. Such functions are expected to work in a plug-and-play fashion.

Algorithm 2. Simulated Annealing - Profile Production

```
Input: same_iterations: After how many iterations showing no improvement will we stop our search T: Temperature
Score(): Deployment profile score function Output: A near-optimal deployment profile 1: same ¼ 0
2: best profile ¼ current profile ¼ GetRandomProfile()
3: while same < same_iterations do
4: new profile ¼ GetNeightborOf(current profile)
5: D = Score(new profile) - Score(current profile)
6: if ( T > 10_5 AND eD=T > Random()) OR (T < 10_5 AND D > 0) then
7: current profile ¼ new profile 8: end if
9: if Score(new profile) > Score(best profile) then 10: best profile ¼ new profile
11: same ¼ 0 12: end if   13: sameþþ
14: T ¼ 0:99 _ T 15: end while
16: return best profile
```

Algorithm 2 chooses to update current_profile with one of its neighbors based on a probability factor: eD=T > Random(), where D is the score improvement we get usingthe neighboring profile and T the temperature. Using this formula, we handle local minimum pits by allow in "jumps" to lower scoring profiles. However, when the temperature drops near zero (10_5) only higher scoring neighbors are visited. Apart from the starting temperature and the number of non improving iterations performed before returning the best profile (same_iterations)another option for enhancing the imal VM-to-host mapping. This allows us to place constraints into two categories:rofile. quality is the number of times Nefeli runs simulated annealing. Starting from a different initial VM deployment, allows our approach not to get trapped at locally optimum solutions. Our approach decouples the profile evaluation and generation from the process of finding a near-optimal VM-to-host mapping. This allows us to place constraints into two categories:

Soft constraints: The degree of satisfaction of constraints that belong in this class contributes tothe overall quality of the produced profile.
Hard constraints: Conditions placed in this group have to be satisfied to their full extent. Otherwise, task-flows featuring such constraints are simply not admitted for execution and receive no further consideration. Escalating the severity of a soft constraint to hard requires setting its weight to 1.0 in the respective task-flow XML-description. Soft constraints are used for the computation of each profile score. Hard constraints are taken into consideration during the generation of new profiles from functions Get Neighbour Of and GetRandomProfile of Algorithm 2. These two functions also take into account the obvious constraints raising from the limited availability of hardware resources such as the available main memory on each hosting node.

A. Cloud Consumer Extraction:

The important role for the cloud user is to move login window to cloud user window. This module has created for the security purpose. In this login page we have to enter login user id and password. It will check username and password is match or not (valid user id and valid password). If we enter any invalid username or password we can't enter into login window to user window it will shows error message. So we are preventing from unauthorized user entering into the login window to user window. It will provide a good security for our project. So server contain user id and password server also check the authentication of the user. It well improves the security and preventing from unauthorized user enters into the network. In our project we are using JSP for creating design. Here we validate the login user and sever authentication.

B. Cloud Consumer Constrains:

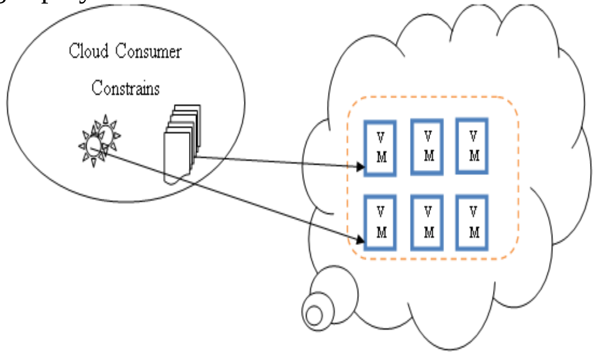
Constraints express user (cloud consumer) Each constraint is realized as a utility function F that evaluates a single deployment profile. F takes as input a deployment profile and returns the degree of constraint satisfaction in the range of (0, 1). In Nefeli, each such function has at its disposal all information regarding the characteristics of both physical and virtual nodes. The provider's perspective, this translates to hosting the VM on a physical node that is unlikely to fail. We have to provide new implementations for the affected utility functions.

C. Cloud Administration Constrains

Administrative constraints are also realized as utility functions. These constraints serve a dual purpose as they can introduce high-level policies and assist in administration tasks. For instance, Empty Node relieves a hosting node of VMs. Reduce Dist enforces the high-level policy of clustering VMs of the same user on hosts that may not be far apart in terms of network hops; this is done to limit major traffic to be routed over long-haul physical networks. We have realized both user and administration constraints of as utility functions in similar fashion to Algorithm; for brevity, we omit their detailed discussion here. Such functions are expected to work in a plug-and-play fashion.

D. Virtual Machine Scheduling:

we refer to VM layout patterns as task-flows to distinguish them from the traditional workflow concept. Specifically, task-flows illustrate "ideal" deployments of VMs described by the cloud consumers using deployment hints. Nefeli exploits these hints so as to (re)deploy VMs in the cloud and achieve efficient task-flow execution. However, hints must not reveal any cloud internal properties to the consumers. Although hints may offer a desired VM-deployment for consumer workloads, Nefeli may ultimately elect to ignore part or all of them based on the available physical resources. In addition to hints, Nefeli also takes into account high-level VM placement policies, set by the cloud administration, whose objectives may entail energy efficiency and load balancing. Here we have created a cloud-enabled application that performs video and audio transformations while offering deployment hints to Nefeli.



E. Nefeli's Structured Layout and Interaction Model

specified Nefeli, it will may interact with the physical infrastructure through a cloud middleware. However, the cloud middleware may not provide all the functionality required by Nefeli. For instance, OpenNebula does not expose all host-related information it gathers. In such cases, we have to realize any missing functionality and incorporate it in the "cloud middleware connector" component (denoted as "extra functionalities"). Nefeli plays a major role in helping attain user-favorable VM deployments. The user remains unaware of the cloud internals as any piece of his information arriving at Nefeli (the cloud gateway) strictly refers to the type of the workload(s) the virtual infrastructure is to serve. Nefeli has the role of an IaaS-cloud gateway. Users contacting Nefeli request virtual infrastructures created by instantiating sets of VMs. As the user must be kept agnostic of the internal deployment decision algorithms of the cloud, all available constraint types are provided by Nefeli.

III. DISCUSSION

Nefeli is to make choices regarding the deployment profile based on the user's needs and the system's performance. As both needs and actual preferences change over time, Nefeli must act accordingly and produce updated deployment profiles. To this end, our approach features a notification mechanism that relays events toward Nefeli. Events are used to signal when the virtual infrastructures should be reorganized— through VM migration operations— so as to reflect the changes in Nefeli's environment. We group events into two classes according to their origin: .i)Events activated by direct human intervention ii) Events triggered by any monitoring activity in the context of physical/virtual infrastructure or any authorized third party component: Our experimentation entails diverse scientific task-flows executed on simulated infrastructures as well as applications executed in a private Iaas-cloud.

The difference between simulation and real application evaluation is in the infrastructure used. We have implemented two cloud middleware connectors We now outline our evaluation using a real private-cloud environment running Nefeli and show the gains obtained when compared with the scheduler of a widely deployed open-source cloud middleware. We have created a cloud-enabled application that performs video and audio transformations while offering deployment hints to Nefeli. Such applications are very well suited to cloud execution as many VMs can simultaneously operate on separate fragments of the input media. In addition, the elongated processing time ameliorates the VM scheduling and deployment delays. numerous algorithms for detecting frequent motifs using the Hamming distance notion of similarity.

we specified Nefeli, it will may interact with the physical infrastructure through a cloud middleware. However, the cloud middleware may not provide all the functionality required by Nefeli. For instance, OpenNebula does not expose all host-related information it gathers. In such cases, we have to realize any missing functionality and incorporate it in the "cloud middleware connector" component (denoted as "extra functionalities"). Nefeli plays a major role in helping attain user-favorable VM deployments. The user remains unaware of the cloud internals as any piece of his information arriving at Nefeli (the cloud gateway) strictly refers to the type of the workload(s) the virtual infrastructure is to serve. Nefeli has the role of an IaaS-cloud gateway. Users are aware of the flow of tasks executed in their virtual infrastructures and the role each VM plays.The information is passed to the cloud provider, as hints, and helps drive the placement of VMs to hosts. Hints are also employed by the cloud administration to express its own deployment preferences. the proximity of certain VMs to data repositories, or any other information that would contribute in a more effective placement of VMs to physical hosting nodes Compared to other existing scheduling VM-based load balancing systems Nefeli exhibits two key differences. First, our approach does not examine the execution of specific VMs in isolation but considers all task-flows making up the current workload before rearranging the virtual infrastructure. Second, the event-based mechanism that we use to trigger VM rearrangements is not based solely on specific usage thresholds of resources.

IV.CONCLUSION

we present Nefeli, a hint-based VM scheduler that serves as a gateway to IaaS- louds. Users are aware of the flow of tasks executed in their virtual infrastructures and the role each VM plays. This information is passed to the cloud provider, as hints, and helps drive the placement of VMs to hosts. Hints are also employed by the cloud administration to express its own deployment preferences. Nefeli combines consumer and administrative hints to handle peak performance, address performance bottlenecks, and effectively implement high-level cloud policies such as load balancing and energy savings. An event-based mechanism allows Nefeli to reschedule VMs to adjust to changes in the workloads served. Our approach is aligned with the separation of concerns IaaS-clouds introduce as the users remain unaware of the physical cloud structure and the properties of the VM hosting nodes. Our evaluation, using simulated and real private Iaas-cloud environments, shows significant gains for Nefeli both in terms of performance and power consumption. . In the future we need to investigate alternative constraint satisfaction approaches to address scalability issues present in large infrastructures and offer deployment hints that will effectively handle the deployment of virtual infrastructures in the context of real large cloud installations.

V .REFERENCES

[1] C. Hyser, B. McKee, R. Gardner, and B.J. Watson, (2008),” AutonomicVirtual Machine Placement in the Data Center,”

[2] H.N. Van, F.D. Tran, and J.-M. Menaud,(pp. 1-8, 2009) “Autonomic Virtual Resource Management for Service Hosting Platforms,” Proc. ICSE Workshop Software Eng. Challenges of Cloud Computing,

[3] C. Weng, M. Li, Z. Wang, and X. Lu, (pp. 183-190, June 2009)” Automatic Performance Tuning for the Virtualized Cluster System,”

[4] D. Josephsen ,(2007),” Building a Monitoring Infrastructure with Nagios”.

[5] B. Sotomayor, K. Keahey, and I. Foster, pp. 87-96, June 2008. “Combining Batch Execution and Leasing Using Virtual Machines,”

[6] F. Hermenier, J. Lawall, J.-M. Menaud, and G. Muller,(2011) “Dynamic Consolidation of Highly Available Web Application,”

[7] P. deGrandis and G. Valetto,(2009), “Elicitation and Utilization of Application-level Utility Functions,”

[8] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L.Youseff, and D.Zagorodnov , pp. 124-131, May 2009. “Eucalyptus Open-SourceCloud-Computing System,”

[9] Konstantinos Tsakalozos, Mema Roussopoulos, and Alex Delis (july 2013)”Hint-Based Execution of Workloads in Clouds with Nefeli”.

[10] D. Jayasinghe, C. Pu, T. Eilam, M. Steinder, I. Whalley, and E.Snible, (July 2011) “Improving Performance and Availability of Services Hosted on IaaS Clouds with Structural Constraint-Aware Virtual Machine Placement”

[11] J. Xu, M. Zhao, J. Fortes, R. Carpenter, and M. Yousif, (June 2007), “On the Use of Fuzzy Modeling in Virtual ized Data Center Management”

[12] G. Jung, K.R. Joshi, M.A. Hiltunen, R.D. Schlichting, and C. Pu,(June 2010),” Performance and Availability Aware Regeneration For Cloud Based Multitier Applications,”

[13] A. Verma, P. Ahuja, and A. Neogi,(2008)“pMapper: Power and Migration Cost Aware Application Placementin Virtualized Systems,”

[14] Brendan Cully, Geoffrey Lefebvre, Dutch Meyer, Mike Feeley, Norm Hutchinson, and Andrew Warfield ”Remus: High Availability via Asynchronous Virtual Machine Replication”

[15] D. Prangchumpol, (2012)" Server Virtualization using user Behavior Model Focus on Provisioning Concept"
[16] K. Tsakalozos, M. Roussopoulos, and A. Delis, (Dec. 2011.)" "VM Placement in Non-Homogeneous IaaS-Clouds."
[17] MR. A. Anbumani M.C.A., M.E., Department of Computer Science and Engineering, Mahendra Institute of Technilogy, Thiruchengode.
[18] R.T.Gokul M.sc., Department of Conputer Science and Engineering, Mahendra Institute of Technology, Thiruchengode.

AUTHOR'S PROFILE:

[1]. G.LAVANYA , Pursuing M.Tech in Department of Computer Science & Engineering at Malla Reddy College Of Engineering & Technology, Hyderabad,Telangana.India.

[2]. M.VAZRALU, working as Associate .Professor in Department of Computer Science & Engineering, Malla Reddy College Of Engineering & Technology, Hyderabad, Telangana.India.