International Journal of Applied Sciences, Engineering and Management
ISSN 2320 – 3439, Vol. 02, No. 03, May 2013, pp. 68 - 72

Active Reserve Distribution in Cloud Computing Beyond Distributed Various Standards Decision Evaluation

P. DIVYA [1], Dr. SVS PRASAD [2], KBKS DURGA[3]
[1]*Student, M. Tech (CSE), Narayana Engineering and Technical Campus*
[2] *professor & principal, Narayana Engineering and Technical Campus*
[3]*Associate Professor, St. Martins Engineering College*
Email:divya3089@gmail.com, drsvsprasad@gmail.com,durgakbks@gmail.com

Abstract: Large data centers host several application environments (AEs) that are subject to workloads whose intensity varies widely and unpredictably. Therefore, the servers of the data center may need to be dynamically redeployed among the various AEs in order to optimize some global utility functions computing clouds, it is desirable to avoid wasting resources as a result of under-utilization and to avoid lengthy response times as a result of over-utilization. In this technical report, we investigate a new approach for dynamic autonomous resource management in computing clouds. The main contribution of this work is two-fold. First, we adopt a distributed architecture where resource management is decomposed into independent tasks, each of which is performed by Autonomous Node Agents that are tightly coupled with the physical machines in a data center. Second, the Autonomous Node Agents carry out configurations in parallel through Multiple Criteria Decision Analysis using the PROMETHEE method. Simulation results show that the proposed approach is promising in terms of scalability, feasibility and flexibility.

Introduction

For clients, cloud computing provides an abstraction of the underlying platform which saves them the costs of designing, building and maintaining a data center to host their Application Environments (AE). By leveraging system virtualization in a data center, multiple Virtual Machines (VM), which are the components of AEs, can be consolidated on one Physical Machine (PM). A VM is loosely coupled with the PM it runs on; as a result, not only can a VM be started on any PM, but also, it can be migrated to other PMs. Cloud computing is a popular trend in current computing which attempts to provide cheap and easy access to computational resources. Compared to previous paradigms, cloud computing focuses on treating computational resources as measurable and billable utilities. From the clients' point of view, cloud computing provides an abstraction of the underlying hardware architecture. This abstraction saves them the costs of design, setup and maintenance of a data center to host their Application Environments (AE). Whereas for cloud providers, the arrangement yields an opportunity to profit by hosting many AEs. This economy of scale provides benefits to both parties, but leaves the providers in a position where they must have an efficient and cost effective data center.

Currently, migrations can be performed in seconds to minutes depending on the size, activity and bandwidth of the VM and PMs. Migration makes it possible to dynamically adjust data center utilization and tune the resources allocated to AEs. Furthermore, these adjustments can be automated through formally defined strategies in order to continuously manage the resources in a data center with less human intervention. We refer to this task as the process of dynamic and autonomous resource management.

The ability to migrate VMs makes it possible to dynamically adjust data center utilization and tune the resources allocated to AEs. Furthermore, these adjustments can be automated through formally defined strategies in order to continuously manage the resources in a data center with less human intervention. We referred to this task as the process of dynamic and autonomous resource management in a data center.

In former research, this process is generally performed in a centralized manner, focusing on the use of utility functions to declare the preferences of AEs over a range of resource levels. Each AE then communicates its preferences to a global arbiter which computes a near-optimal distribution of resources among AEs [1]. In this paper, we propose a new approach to the same problem in the context of computing clouds. Our contribution is two-fold. First, we adopt a distributed architecture where resource management is decomposed into independent tasks and each task is performed by Autonomous Node Agents (NA) that are tightly coupled with the PMs in a data center. Second, NAs carry out configurations through Multiple Criteria Decision Analysis (MCDA) using the PROMETHEE method [2].

The rest of this report is organized as follows. we provide a general description of the problem. It tells the outlines the former research in the field. And also provides the details of the overall design our approach.we provide details on the PROMETHEE method and explain the experimental setup in detail. IN this provides a detailed assessment of our approach. Finally, we summarizes our conclusions and outlines our future directions.

Problem Description

In a data center, the primary goal of a dynamic autonomous resource management process is to avoid wasting resources as a result of under-utilization. Such a process should also aim to avoid high response times as a result of over-utilization which may result in violation of the service level agreements (SLA) between the clients and the provider. Furthermore, it needs to be carried out continuously due to the time variant nature of the workloads of AEs.

Resource consolidation in data centers can generally be defined as the dynamic and autonomous process of providing mappings between a set of application environments and a pool of shared computational resources. The process is dynamic since the resource usages of the application environments are time variant which stems from two major factors.

1. First of all, the amount of computational resources is variable throughout time due to the possibility of addition, removal and temporary unavailability of hardware/ software in a data center (e.g. unexpected failures, scheduled maintenance, etc.).

2. Secondly, neither the number of application environments nor their performance level and resource level requirements at an arbitrary instance can be statically determined.

Resource consolidation is also an autonomous process in a sense that its goal is to minimize human intervention during mapping/re-mapping in order to provide a robust self-configuring infrastructure that is more responsive to changing conditions

At a high level, this process can be decomposed into two distinct, and inter-dependent phases. The first phase consists of defining a mapping between the AEs' service level requirements and resource level requirements. Resource level requirements are generally derived from SLAs based on certain parameters such as response time, throughput, etc; whereas, resource level requirements are often outlined as CPU usage, memory, bandwidth, etc. As the workload of an AE changes in time, this mapping is used to determine the amount of resources that should be assigned to each component—encapsulated in VMs—in order to satisfy the terms outlined in the SLA. This phase also requires performance modeling and demand forecasting for AEs. The accuracy of the output from this first phase has direct effects on the accuracy of the configuration produced in the second phase.

The second phase involves the computation and application of a new configuration by distributing the resources in a data center among the VMs that represent AEs. The configuration is computed based on the output of the mappings produced in the first phase. Maintaining this configuration is a resource allocation problem and is generally defined as a Knapsack Problem [1] or as a specific variant of it, namely Vector Bin Packing Problem [3], both of which are known to be NPHard. This phase consists of selecting a suitable configuration from a solution space with respect to a set of criteria. The criteria are used to define the quality of the solution in terms of certain requirements such as satisfying SLAs, overall data center utilization, and the overhead of applying an alternative configuration. The methods to be adopted in this phase need to be flexible so that the providers can easily redefine the configuration goals by adding new criteria or tuning the importance assigned to them.

In the second phase, certain constraints and limitations need to be taken into consideration. Two of these are the time-spent during the selection of a new configuration, and the feasibility of it. Due to the time variance in workloads, a new configuration must be computed in a reasonable amount of time so that it is not stale under the current conditions. The selected configuration must also be feasible in terms of the number migrations necessary. The number of migrations that can be performed in a data center still has limits with the current technologies. In summary, the second phase must provide fastto-compute, feasible, and reasonably good configurations that do not over-utilize or under-utilize the data center.

In this paper, we focus on the second phase of the process and leave the problem of defining accurate resource mappings, performance modeling or demand forecasting for AEs outside the scope of this work. We believe that these problems can be decoupled, and solutions that address the first phase should be investigated in a separate work.

Related work

The former research in the field can be viewed under the two groups: (1) resource consolidation in multiple-server data centers, and (2) resource consolidation on single-server shared centers. Studies that fall under the first group focused on allocation of resources in a data center to a number of application environments. A two-layered architecture of a resource consolidation system for non-virtualized data centers was proposed in the work of Walsh et. al.[4], which is used as a common architecture in a majority of the later research. This architecture consists of a local agent assigned to each application environment, through which the required amount of resources is computed. The information that is generated by the local agents is then communicated to a global arbiter which computes a near-optimal reallocation of resources in the entire data center. This seminal work also integrated this two-layered approach with the

usage of utility functions as a measure of desirability of different amount of resources from the application environments' point of view, where the utility values monotonically increase from undesirable amounts of resources to the desirable amounts of resources, between the values of 0 and 1. These utility values are then used to calculate a maximum global utility by the global arbiter under the condition of not exceeding the available resources in a data center. This, in essence, forms the optimization problem which is generally modeled as Knapsack Problem where a global near-optimal configuration is computed from the individual preferences of local agents. Studies outlined in Bennani and Menasce [5], Chess et. al. [6], Tesauro ,Tesauro et. al. [7], and Das et. aldopted this same approach which is then merged with forecasting methods based on different analytical models on non-virtualized data centers..

In Chess et. al. and Das et. al. this same architecture and utility model is used to build a commercialized computing system, called Unity. Some of the common aspects of these works and their focus on resource consolidation are revealed as (1) a purely application environment centric view where data center utilization is not of major concern, (2) the main goal being the computation of an optimal configuration of resources in entire the data center, (3) and the absence of the cost of component movements in the formulation of the problem.

A generally employed method is based on periodically sampling data at the end of fixed time intervals. The raw information on resource usage that is collected by the monitoring mechanism is later processed in the detection phase. In the detection phase, the main concern is to capture any anomalies in terms of the violations of performance aggreements, and if there is any, to trigger the configuration phase. In the literature, the detection phase is devised in the two different manners of either treating it as a separate phase.

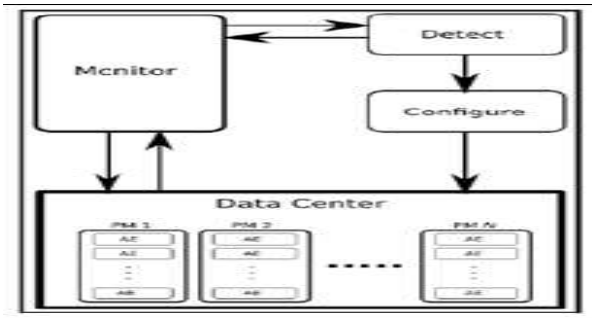


Fig 1: Representation of a resource consolidator in terms of the phases of monitor, detect and configure as an observe-detect-react cycle.

Fig 1 :The Representation of a Resource Consolidator in terms of the phases of Monitor, Detect and Configure as an Observe-Detect-React cycle. that is entered periodically [8], or by merging it with monitoring to trigger configuration on an event-driven basis where events are often considered to be violations or predicted violations based on demand and load forecasting of performance agreements.

Finally, configuration is the phase where the consolidation reacts to the changes in the data center in terms of re-allocating resources for application environments that have undergone changes that have critical affects on their performances. As a result, application environments are re-assigned resources wherever available in the data center, which in turn might require movement of certain components of an application environment between computational units. The nature and effects of such movements are also have attracted a certain amount of interest in the field. In the literature, this configuration phase is generally considered as a Multi-Dimensional Knapsack Problem or as a certain variant of it Vector Bin Packing Problem, which are NP-Hard. [8]

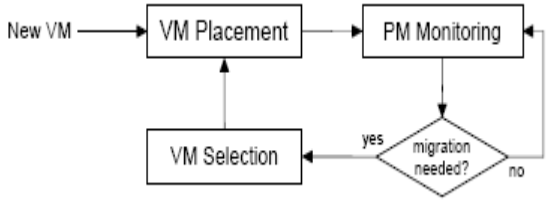
System architecture

We designed the system architecture based on the idea of avoiding the problems of scalability that may emerge as a result of determining and maintaining globally optimal configurations through facilitating a centralized arbiter. We strongly believe that as the data center expands both the computation of optimal configurations and centralizing these computations in a global arbiter might impose serious complexities. As a result of this view, we designed the system as a two-level architecture of (1) application agents that are closely coupled with application environments that declare up-to-date resource requirements, and (2) node agents that are coupled with the computational units in the data center that continuously distribute resources based on the data that application agents declared.

The system is designed as a distributed network of NAs, each capable of accommodating VMs and, when necessary, delegating VMs to other PMs and handing the management over to the corresponding NAs. We assume that the network maintains global awareness of the resource availability of PMs and their task assignments through an oracle. Concretely, the oracle can be implemented as either a distributed or a centralized monitoring system [9].

The system model is described in abstract functional and operational terms based on the Abstract State Machine (ASM) paradigm [10] and the CoreASM open source tool environment [11] for modeling dynamic properties of distributed systems. ASMs are known for their versatility in computational and mathematical modeling of complex distributed systems with an orientation towards practical applications. The description of the model in ASM

reveals the underlying design concepts and provides a concise yet precise blueprint for reasoning about the key system properties. Due to space limitations, in this paper we present only certain parts of the abstract model.



Program

Each NA observes the local resource usages of the PM that it is coupled with. If the local observations reveal an anomaly that the resources are over-utilized or under-utilized, it is immediately followed by the configuration phase which is carried out in three steps:
1) Choosing a VM to migrate from the list of VMs that run on the problematic PM.
2) Choosing a PM to migrate the chosen VM to.
3) Migrating the chosen VM to the chosen PM.

The process of dynamically allocating VMs to PMs and maintaining resource distribution among VMs to meet their resource requirements is modeled as a distributed process carried out by individual NAs in the system. Conceptually, every NA continuously performs a cycle of three activities (see Figure 1): (1) VM Placement where a suitable PM capable of running the given VM is found and the VM is assigned to that PM, (2) Monitoring where the NA monitors total resources used by the hosted VMs; (3) VM Selection where, if local accommodation is not possible, a VM is selected to be migrated to another PM and the process loops back into placement.

The rest of this section focuses on the main activities of the NAs.

A. VM Placement

The VM Placement activity consists of two tasks: (1) Finding suitable PMs and allocating resources for unassigned VMs, and (2) Assigning allocated VMs to their corresponding PMs. In the Placement activity, an NA looks for the VMs that are either unassigned or allocated. For these VMs, NA is acting like a broker;

Placement

Placement ≡
ResourceAllocation
TaskAssignment

In Resource Allocation, an NA picks an unassigned VM from its VM pool and tries to find a suitable PM with available resources that can run the VM. At this level, we capture the computation of finding such a PM by the abstract function placement first filters out the nodes in the data center that do not have the resources to host a certain task After a list of the nodes that have the necessary resources is provided, the problem is to choose the best node defined on two criteria, resource utilization and rate of change in resource usage. Resource utilization is the criteria that ensures that maximization of resource utilization is taken into account (lower the resources on a node, higher the utilization will be), whereas rate of change in resource usage reflects on the stability of available resource usage on a node (low rate of change points to less likeliness of further migrations, and thus less likeliness of performance drops on the tasks). In order to reflect on both the provider and clients these two criterion can be weighted respectively (taking into account both or one more than the other).

If such a node is found, the broker node sends a resource lock request to the target node to allocate its resources before actually delegating the task to the target node. It also changes the status of the task to allocated. The following ASM rule captures this behavior

NAs use the PROMETHEE method to rank each alternative in APM. The PROMETHEE method is based on pair wise comparisons of the alternatives. That is, it considers the difference between the evaluations of two alternatives over each criterion. The preference of one alternative over another is modulated as a function of this difference. The preferences are quantified as real numbers between 0 and 1 through certain proposed preference functions. Each criterion is associated with a generalized criterion which is represented by the pair (gi; Pi(a; b)), where Pi(a; b) denotes the preference of alternative a over b for criterion gi. In this paper, we use the Usual Criterion as a common generalized criterion for all of the criteria.

B. Monitoring and Tuning

In monitoring, an NA continuously observes the resource usages of the PM that its coupled with. The main idea behind monitoring is to capture any anomalies at a PM with respect to allocation of resources. Nodes continuously monitor their resource utilization and their task requirements. There are three main activities under monitoring and tuning: a) for all running tasks, a node monitors changes in its resource requirements and adjusts resource allocations if needed; if such adjustment is not possible, a task replacement may be triggered; b) a node also monitors its resource utilization and adjust its resource allocation to keep the utilization within a reasonable

boundary; c) finally,any resource lock that is timed out (for which a task is not received) needs to be relased. The following ASM rule abstractly captures these above three activities

C. VM Selection

Once an anomaly is captured by the monitor, VM Selection activity starts. This is where one or more VMs are selected from the VM pool of the problematic PM to be migrated to another PM in order to resolve the anomaly. The selected VMs are used by the VM Placement activity to find a suitable target PM for each of them in the data center.

The selection of VMs is done using PROMETHEE similar to the way it is used in VM Placement. However, the alternative set is formed in a different manner. In VM Selection, an NA does not need to contact the oracle since it does the selection from its local pool of VMs. Therefore, the alternative set is computed based on the PM's local information.

Conclusion

Here the Simulation results show that our approach is potentially more feasible in large data centers compared to centralized approaches. In essence, this feasibility is due to the significantly lower number of migrations that are performed in order to apply new configurations. Simulation results show that our method should theoretically trigger less SLA violations by smoothly distributing the overall resource utilization over slightly more PMs in the data center. The flexibility of our approach comes from its ability to easily change the weights of criteria and adding/removing criteria in order to change configuration goals. Our approach consists of a distributed architecture of NAs that perform resource configurations using MCDA with the PROMETHEE method. The simulation results show that this approach is promising particularly with respect to scalability, feasibility and flexibility.

References

[1] W. E. Walsh, G. Tesauro, J. O. Kephart, and R. Das, "Utility Functions in Autonomic Systems," in ICAC '04: Proceedings of the First International Conference on Autonomic Computing. IEEE Computer Society,2004, pp. 70–77.

[2] B. Mareschal, "Aide a la Decision Multicritere: Developpements Recents des Methodes PROMETHEE," Cahiers du Centre d'Etudes en Recherche Operationelle, pp. 175–241, 1987.3

[3] F. Hermenier, X. Lorca, J. M. Menaud, G. Muller, and J. Lawall, "Entropy: A Consolidation Manager for Clusters," in VEE '09: Proceedings of the 2009 ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments, 2009, pp. 41–50.

[4] W. E. Walsh, G. Tesauro, J. O. Kephart, and R. Das, "Utility Functions in Autonomic Systems," in ICAC '04: Proceedings of the First International Conference on Autonomic Computing. IEEE Computer Society,2004, pp. 70–77.

[5] J. Almeida, V. Almeida, D. Ardagna, C. Francalanci, and M. Trubian,"Resource Management in the Autonomic Service-Oriented Architecture,"in ICAC '06: IEEE International Conference on Autonomic Computing, 2006, pp. 84–92.

[6] G. Tesauro, N. Jong, R. Das, and M. Bennani, "A Hybrid Reinforcement Learning Approach to Autonomic Resource Allocation," in ICAC '06:Proceedings of the 2006 IEEE International Conference on Autonomic Computing. IEEE Computer Society, 2006, pp. 65–73.

[7] T. Wood, P. Shenoy, A. Venkataramani, and M. Yousif, "Black-Box and Gray-Box Strategies for Virtual Machine Migration," in 4th USENIX Symposium on Networked Systems Design and Implementation, 2007, pp. 229–242.

[8] Mohamed N. Bennani and Daniel A. Menasc´e,*Resource Allocation for Autonomic Data Centers using Analytic Performance Models* (George Mason University 4400 University, Infocom)

[9] M. L. Massie, B. N. Chun, and D. E. Culler, "The Ganglia Distributed Monitoring System: Design, Implementation And Experience," Parallel Computing, vol. 30, p. 2004, 2003.

[10] E. B¨orger and R. St¨ark, Abstract State Machines: A Method for High-Level System Design and Analysis. Springer-Verlag, 2003.

[11] R. Farahbod, V. Gervasi, and U. Gl¨asser, "CoreASM: An Extensible ASM Execution Engine," Fundamental Informaticae, pp. 71–103, 2007.

P.Divya Studying Mtech (CSE) in Narayana Engineering and Technical Campus under JNTUH University.

SVS Prasad Working as an Professor And Principal For Narayana Engineering And Technical Campus And Doing PhD In Data Mining In JNTUH University.

K.B.K.S.Durga Working as an Associate Professor In St.Martin's Engineering College and Having 8 Years of Teaching Experience and Done M,Tech In Gurunanak Engineering College Under Jntuh Univeristy.