Stretching site resources in cloud computing

[1]K.Kajendran, [2]C.Divya, [3] R.Salini
[1]Faculty, Department of M.C.A, Panimalar Engineering College, Chennai
[2]PG Scholar,Department of M.C.A, Panimalar Engineering College, Chennai
[3]Faculty, Department of M.C.A, Panimalar Engineering College, Chennai

ABSTRACT A stretchy site mechanism is to solve the allocation of resources problem of computation capacity in the environment of cloud computing is proposed here. In this paper, we investigate the benefits that organizations can reap by using “Cloud Computing" providers to augment the computing capacity of their local infrastructure. We have implemented a resource manager, built on the Nimbus toolkit to dynamically and securely extend existing physical clusters into the cloud.. Requests for resources are submitted to the organisation's cluster, but additional Resources s are instantiated in the remote provider and added to the local cluster when there are insufficient resources to serve the users' requests.

INTRODUCTION

Cloud computing enhances the mechanisms for sharing of remote resources by providing users with control over the remote resources.In cloud computing, the allocation method plays a vital role in managing large scale of computation capacity. Computing resources, offer a static computational network, and storage capacity to users, but the demand is dynamic. There are situations where system administrators need to find additional resources to meet this dynamic demands. On the whole , system administrators could purchase additional physical resources when the freight of a particular resource increased beyond its capacity. Purchasing additional physical resources is a very bulky process and it engross a large amount of time and cost . It is difficult to determine when the demand will exceed.It can be found only when the system crash occurs.The Purchase of additional resource is slow and the resource is under utilized.
With Grid computing [3] it became possible to influence the existing resources at remote sites when additional capacity was required. remote nodes that are provided on-demand must be automatically configured to join the site and perform their required role (e.g. a cluster worker node).

We must also be able to correctly identify when nodes are required, the duration for which they will be needed, how many nodes should join the site, and which cloud providers should be utilized We demonstrate a dynamic and responsive stretchy cluster,capable of responding effectively to a variety of job submission patterns.

In this work, we evaluate via simulation six strategies for improving scheduling performance through the use of a Cloud provider.

In summary, the contributions of this work are to:
-Describe a system that enables an organisation to augment its computing infrastructure by allocating resources from a Cloud provider.
_ Provide various scheduling strategies that aim to minimize the cost of utilising resources from the Cloud provider.
_Evaluate the proposed strategies, considering different performance metrics; namely average weighted response time, job slowdown, number of deadline violations, number of jobs rejected, and the money spent

RELATED WORK

The term *Cloud Computing* is currently used in various ways and is often confounded with the term Grid Computing. Grid computing accrued in the mid 1990s and originally denoted a scientific network to share computation power for computationally intensive jobs. The main characterization of a Grid is the distribution of a computing job in a somehow connected network

Grid doesn't offer a way for users to gain access to Grid resources on demand.
Grid users are not typically given root on remote resources for security reasons, and thus it is not always possible to deploy custom software stacks or databases required by the application.
It does not stretch extend the site dynamically

Our related work is based on two ways i.e., solutions that dynamically adapt to changing demand by acquiring or releasing resources and solutions that integrate with cloud computing resources to provide additional compute or storage capacity when needed.VioCluster [4] is a solution that adjusts dynamically to demand by borrowing and lending machines between different clusters. VioCluster calculates whether to borrow machines by counting the number of requested nodes in the queue and subtracting the number of available machines in the cluster. If the number is positive then this is the number of machines that need to be borrowed, if the number is zero then it is the number of machines the cluster has available to lend. Stretchy site doesn’t borrow or lend machines; instead we only acquire nodes from the cloud when needed.

PROPOSED SYSTEM

Cloud computing enhances the mechanisms for sharing of remote resources by providing users with control over the remote resources.
It dynamically extending the resources of a static site to adjust to changes in demand identify when nodes are required, the duration for which they will be needed, how many nodes should join the site, and which cloud providers should be utilized.
These decisions must balance the cost of integrating remote nodes from different cloud providers with the need to efficiently process the demand.

Ruth et al. [5] create an adaptive environment of VMs that is able to adjust the environment based on measuring the current load within the VMs.

Our approach extends existing resources with cloud resources based on overall demand, which differs from their focus on creating individual, autonomous, adaptive environments capable of dynamically adjusting to load within the VMs. Murphy et al. [6] create a system to dynamically provision virtual organization clusters. The system is similar to our implementation in that they monitor a job queue and spawn VMs to process the jobs.

MODULE USED

- Admin Module
- Resource Sharing
- Extend Site
- Report Generation

Admin Module:

The Admin has to maintain the user details.The admin maintain the resources.

Resource Sharing Module:

The user share the resources.The resources are available and it can be shared by many users.

Stretchy Site:

If the Resource is not available then we can extend site.In the extended site, the user can share and use resources.

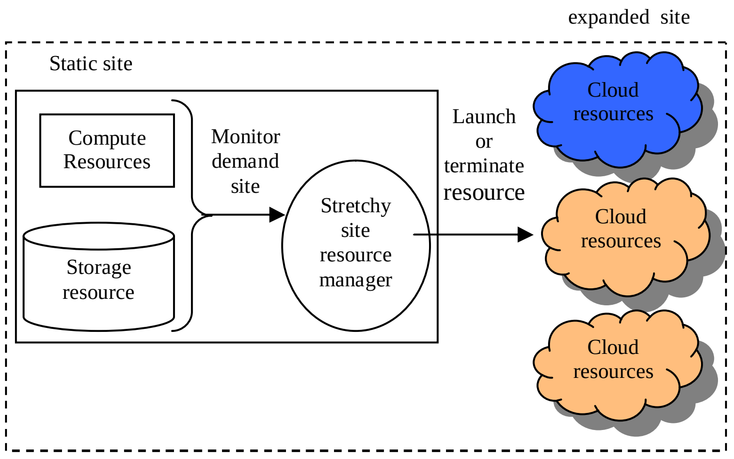
Report Generation:

The user share the resources and the details are stored in database.We can generate the report and display from the database.

SYSTEM ARCHITECTURE

Motivated by these observations, we are exploring a Scalable stretchy System Architecture which aims to enable extreme degrees of stretchability across all system layers, aiding in the construction of stretchy software.

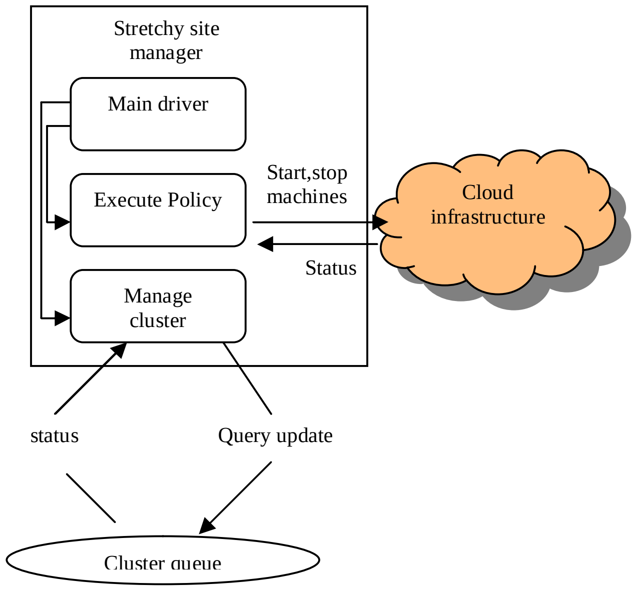
Figure 1: stretchy site model



Along with an abstract layering, we propose a distributed stretchy component model that can be used to construct layers of the system that are tuned to exploiting fine grain stretchability.

Figure 2: Implementation of stretchy site

System architecture defines four meta layers to a system that can easily be mapped to current and



future cloud computing environments. At the bottom is the physical stretchy Node/HW Layer that represents a data center's underlying computer resources.We assume a model in which the resources of the data center are decomposed into nodes that form the basic unit of resource allocation and thus stretchy. The next layer up, the stretchy Partition Layer, provides groups or partitions of node resources that can be associated with a consumer or principal. A partition is the basic unit by which a principal can stretchably aggregate node resources.

To enable applications to scale and exploit the stretchability of a partition, the next layer provides an stretchy Building Block model. stretchy Building Blocks are the primary way in which application software is structured so that it is scalable and changes in demand can be converted into stretchy consumption of node resources for a particular application of a principal. The top layer is the stretchy service and Runtime Layer. The specific service and or runtime code of the application is written as a dynamically allocated set of stretchy building blocks.

Our systems software focus is on the top two layers. Our goal is to introduce a distributed library OS runtime that enables Building Blocks for the construction of system and application layers that express and exploit the maximum stretchability of the **hardware** to meet the demands of interactive workloads.

Our model attempts to achieve the goals of top-down demand, bottom-up support and exploiting modularity. Specifically, it uses an event model that maps events to method invocations of stretchy building blocks. Building blocks construction and destruction are by default to be triggered by event access and quiescence. The components will have an associated IPC like model so that invocations can cross layers. The library model will enable building block constructed services to be constructed and deployed in conjuction with existing application.

Goals:

Based on our observations, we posit the following goals for a systems architecture for stretchy site.

Top-Down Demand: The system should enable demand on services to flow from high level layers as transparently as possible to the lowest layers of the system. Hoarding should be discouraged or at least made transparent. Event driven interfaces and services should be supported and encouraged by the system.

Bottom-Up Support: We advocate that stretchability should be an explicit characteristic that should be supported in the lowest layers of a system and, if possible, all the way into the hardware. The construction of layers that are explicit about the stretchability they provide with respect to the base stretchability of the system should be encouraged via systems support.

Exploit Modularity: Use modularity to enable applications to map stretchability in their demand as closely to the capabilities of the system when desired. Embrace the layering of cloud computing as an abstract architecture but ensure that all but the lowest layer can have varying implementations or be over-ridden. Further, provide some form of support for a component model that enables the base stretchability to be exploited by new and advanced applications.

Performance Analysis

Our evaluation of stretchy site consists primarily of a comparison of the three different policies in an attempt to maximize job turnaround time while minimizing thrashing and idle VMs.Comparing the performance of different cloud providers, and by extension, the underlying node hardware, network, and software

deployed by those providers is beyond the scope of our work.The evaluation environment consists of a cluster head node at the University of Colorado at Boulder. The head node has two 2.4 GHz Intel Xeon processors with hyper-threading and 6 GB of RAM. We provide an initial comparison between the Nimbus cloud at the UChicago and the cloud at IU;.The IU cloud consists of 2 nodes. Each node has 2 Quad Core Xeon processors with 32 GB of RAM and a single SATA disk. The Nimbus[7] cloud at the University of Chicago has 16 nodes, each with two 2.2 GHz AMD64 processors and 4 GB of RAM. Each node also has a local 80 GB IDE disk. The storage repository is exposed as a GridFTP service running on the service node. For our analysis with the Nimbus cloud at UChicago we specified a maximum of 10 cloud nodes. Thus, a policy may choose to deploy up to 10 nodes, but no more.

To demonstrate scale we use Amazon EC2 small instance machines, without specifying a maximum limit on the number of nodes available to the stretchy site. Small instances contain 1 virtual core with 1.7 GB of memory, and only what Amazon denotes as “moderate” I/O.
After using available number of resources in the current site we can stretch our site based on the additional demand given by the users**.**

Conclusion

In this work we propose a model of an stretchy site capable of responding to changes in demand by leveraging cloud resources. We create an implementation of this model that seamlessly and securely extends clusters with Nimbusbased clouds and Amazon EC2 on demand. One of the most significant is the ability to stretchably provision and relinquish new resources in response to changes in demand. In our work, we develop a model of an “stretchy site” that efficiently adapts services provided within a site, such as batch schedulers, storage archives, or Web services to take advantage of stretchably provisioned resources.

REFERENCES

[1] Amazon CloudWatch. Amazon, Inc. [Online]. Retreived February 7,2010, from: http://aws.amazon.com/cloudwatch/

[2] Amazon Web Services. Amazon.com, Inc. [Online]. Retreived February7, 2010, from: http://www.amazon.com/aws

[3] Foster, I., C. Kesselman, S. Tuecke. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. In the International Journal of HighPerformance Computing Applications, Vol. 15, No. 3, pages 200-222,2001.

[4] Ruth, P., P. McGachey, D. Xu. VioCluster: Virtualization for Dynamic Computational Domains, Cluster Computing, 2005. IEEE International, pages 1-10, Sept. 2005.

[5]Ruth, P., J. Rhee, D. Xu, R. Kennell, and S. Goasguen. Autonomic live adaptation of virtual computational environments in a multi-domain infrastructure. IEEE International Conference on Autonomic Computing, 2006.

[6]Murphy, M., B. Kagey, M. Fenn and S. Goasguen "Dynamic
Provisioning of Virtual Organization Clusters" 9th IEEE International Symposium on Cluster Computing and the Grid, Shanghai, China, May 2009.

[7]Argonne National Laboratory Press Release. [Online]. "Nimbus and cloud computing meet STAR production demands," Retreived February 7, 2010, from:
http://www.anl.gov/Media_Center/News/2009/news090402.html