International Journal of Engineering Innovation & Research
Volume 1, Issue 1, ISSN : 2277 – 5668

LINUX BASED CLOUD OPERATING SYSTEM

Lokesh Patel
Student of M Tech (CSE Dept.)
SSSIST Sehore
g.patelokesh01@gmail.com

Gajendra Singh
Professor
SSSIST Sehore
gajendrasingh86@rediffmail.com

Ravindra Gupta
Assistant Professor
SSSIST Sehore
ravindra_p84@rediffmail.com

Abstract **- Cloud computing is a type of computing in which dynamically scalable and virtualized resources are provided as a service. Users need not have knowledge of, expertise in, or control over the technology infrastructure in the cloud that supports them. Furthermore, cloud computing employs a model for enabling available, convenient and on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. The services mainly operating systems running on a single virtualized computing environment, middleware layers that attempt to combine physical and virtualized resources from multiple operating systems, and specialized application engines. In this paper, we discuss about the virtual distributed operating system, Cloud Operating System, to provide maximum features of cloud computing. The Cloud Operating System will work on the principle laid out by LINUX. The Cloud OS aims to provide simple programming abstractions to available cloud resources, strong isolation techniques between Cloud processes, and strong integration with network resources**.

Keyword **– LINUX, Massive Computing, Cloud OS, Kernel processes, Logical Architecture.**

I. INTRODUCTION:

With the increasing use of high-speed Internet technologies during the past few years, the concept of cloud computing has become more popular. In cloud computing, users work with Web-based, rather than local, storage and software. These applications are accessible via a browser and look and act like desktop programs.

With this approach, users can work with their applications from multiple computers. In addition, organizations can more easily control corporate data and reduce malware infections. Also, cloud computing makes collaboration easier and can reduce platform-incompatibility problems.

Now, a growing number of organizations are adding to the cloud concept by releasing commercial and open source Web-based operating systems. While the idea isn't new, the proliferation of users and applications distributed over the Web, including those at scattered corporate sites has made it more interesting, relevant, and, vendors hope, commercially viable. The Cloud OS goes beyond basic desktop functionality. It also includes many of a traditional OS's capabilities, including a file system, file management, and productivity and communications applications. The computing industry is radically changing the scale of its operations. While a few years ago typical deployed systems consisted of individual racks filled with few tens of computers, today's massive computing infrastructures are composed of multiple server farms, each built inside carefully engineered data centers that may host several tens of thousands CPU cores in extremely dense and space-efficient layouts [1]. The commoditization of computing is thus transforming processing, storage, and bandwidth into utilities such as electrical power, water, or telephone access. This process is already well under way, as today businesses of all sizes tend to outsource their computing infrastructures, often turning to external providers to fulfill their entire operational IT needs.

The migration of services and applications into the network is also modifying how computing is perceived in the mainstream, turning what was once felt as the result of concrete equipment and processes into an abstract entity, void of any physical connotation: this is what the expression "Cloud computing" is currently describing.

Previous research has successfully investigated the viability of several approaches to managing large-scale pools of hardware, users, processes, and applications. The main concerns of these efforts were twofold: on one hand, exploring the technical issues such as the scalability limits of management techniques; on the other hand, understanding the real-world and "systemic" concerns such as ease of deployment and expressiveness of the user and programming interface. Our main motivation lies in the fact that state of the art management systems available today do not provide access to the Cloud in a uniform and coherent way. They either attempt to expose all the low-level details of the underlying pieces of hardware [2] or reduce the Cloud to a mere set of API calls, to instantiate and remotely control resources [3][4], to provide facilities such as data storage, CDN/streaming, and event queues [5], or to make available distributed computing library packages [6][7]. Yet, a major gap still has to be bridged in order to bond the Cloud resources into one unified processing environment that is easy to program, flexible, scalable, self-managing, and dependable.

In this position paper, we argue for a holistic approach to Cloud computing that transcends the limits of individual machines. We aim to provide a uniform abstraction the Cloud Operating System that adheres to well-established operating systems conventions, namely: (a) providing a simple and yet expressive set of Cloud metrics that can be understood by the applications and exploited according to individual policies and requirements, and (b) exposing a coherent and unified programming interface that can leverage the available network, CPU, and storage as the

pooled resources of a large-scale distributed Cloud computer. In Section 2 we elaborate our vision of a Cloud operating system, discuss our requirements we aim to meet. In Section 3 we present the logical architecture of cloud operating system. We then briefly discuss the working in Section 4, and conclude in Section 5.

II CLOUD OS REQUIREMENT

The current datacenter setups can offer a fine-grained amount of control and pervasive management capabilities, the Cloud environment is much less predictable and harder to control: the environment imposes therefore several restrictions to the Cloud OS design, such as the reliance on coarse-grained knowledge about Cloud resource availability, the need to detect and tolerate failures and partitions, and a lack of global view over the system state. Despite these limitations, our design aims to meet the following general requirements:

a) The Cloud OS must permit autonomous management of its resources on behalf of its users and applications: Our main purpose is providing an abstraction of the Cloud as a coherent system beyond the individual pieces of hardware from which it is built. The Cloud OS should therefore expose a consistent and unified interface that conceals whenever possible the fact that individual nodes are involved in its operations, and what those low-level operations are.

b) Cloud OS operation must continue despite loss of nodes, entire clusters, and network partitioning: Conforming to our assumptions, we expect that every system component, including networks, may unexpectedly fail, either temporarily or permanently. Guaranteeing continued operation of the Cloud management processes in these conditions involves mechanisms for quickly detecting the failures and enacting appropriate measures. Note that fault-tolerance at the Cloud level does not imply any guarantee about the fault-tolerance of individual applications: the state of any process could suddenly disappear because of any of the previous events, therefore Cloud applications should be designed with this in mind. Several Cloud libraries that implement common fault-tolerance and state recovery features are provided out of the box.

c) The Cloud OS must be operating system and architecture agnostic: The network is the common interface boundary between the various software elements of the Cloud. The reason for this choice is that we want to enable the broadest compatibility between hardware and software configurations, while providing at the same time an easy way for future evolution of the Cloud system, both at a global and at an individual subsystem level. Experience shows that protocols are able to withstand time much better than ABIs, standard library specifications, and file formats: long-lived protocols such as the X protocol and HTTP are good examples in this regard. While it is wise from an operational standpoint to consolidate the number of architectures supported and standardize around a small number of software platforms, the Cloud OS operation does not depend on any closed set of platforms and architectures.

d) The Cloud must support multiple types of applications, including legacy: In the assumptions above, we purposefully did not specify a target set of applications that the Cloud is supposed to host. Rather than optimizing the system for a specific mode of operation (e.g. high performance computing, high data availability, high network throughput, etc.), we aim to address the much broader requirements of a general-purpose scenario: applications of every type should ideally coexist and obtain from the system the resources that best match the application requirements.

e) The Cloud OS management system must be decentralized, scalable, have little overhead per user and per machine, and be cost effective: The use of such a soft-state approach takes inspiration from recent peer-to-peer techniques: these systems are capable of withstanding failures and churn at the price of a reasonable amount of network overhead, and provide enough scalability to meet and surpass the magnitudes of today's datacenters and large-scale test beds. Moreover, apart from initial resource deployment and key distribution, no human intervention should be required to expand the Cloud resources. Likewise, user management should only entail the on-demand creation of user credentials, which are then automatically propagated throughout the Cloud.

f) The resources used in the Cloud architecture must be accountable, e.g. for billing and debugging purposes:
The cost of an application's deployment across the Cloud is also a part of the end-to-end metrics that may influence the scheduling of resources as per an application's own policy. Moreover, dynamic billing schemes based e.g. on resource congestion [8] could be an effective way to locally encourage a proportionally fair behavior among users of the system and increase the cost of attacks based on maliciously targeted resource allocation [9].

III. LOGICAL ARCHITECTURE OF CLOUD OS

The Cloud object can be defined as a set of local OS processes running on a single node, which are wrapped together and assigned locally a random identifier of suitable length to minimize the risk of system-wide ID collisions. A Cloud process (CP) is a collection of Cloud objects that implement the same (usually distributed) application. We refer to the small number of CPs that regulates physical allocation, access control, accounting, and measurements of resources as the Cloud kernel space. User space CPs that are executed directly by users are called User Applications, while Cloud Libraries are CPs typically called upon by Applications and other Libraries. Applications can interface

with Libraries and kernel CPs over the network through a set of standard interfaces called Cloud System Calls. The assumptions stated above pose very few constraints about the features that the underlying

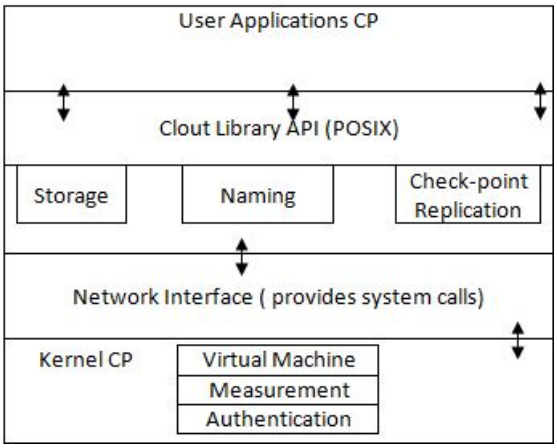

Fig. 1: Logical Architecture of Linux-based Cloud Operating System

Cloud hardware is expected to provide. Basically, the ability to execute the Cloud kernel processes, together with the availability of appropriate trust credentials, is a sufficient condition for a node to be part of the Cloud. A limited access to Cloud abstractions and interfaces is thus also achievable from machines that belong to administrative domains other than that of the Cloud provider, with possible restrictions due to the extent of the management rights available there. All objects in the Cloud user space expose a Cloud system call handler to catch signals from the Cloud OS, i.e. they can be accessed via a network-based interface for management purposes. The association between object names and their network address and port is maintained by the process management and virtual machine management kernel CPs, and the resulting information is made available throughout the Cloud via the naming Library. The naming library also keeps track of the link between User Application CPs and the objects they are composed of. The access rights necessary for all management operations are granted and verified by the authentication kernel CP. Measurement kernel CPs are always active in the Cloud and operate in both on-demand and background modes.

IV. WORKING OF CLOUD OS

A. Implementation of the Cloud kernel processes:

1) Resource measurement: The Cloud OS needs to maintain an approximate view of the available Cloud resources. Our current approach involves performing local measurements on each Cloud node. This technique provides easy access to end to end variables such as latency, bandwidth, packet loss rate, etc., which are precious sources of knowledge that are directly exploitable by the applications. More detailed knowledge requires complete control over the network infrastructure, but it may be used in certain cases to augment the accuracy of End to end measurements (e.g., with short-term predictions of CPU load or networking performance [10]) in Clouds that span several datacenters. Measurements can target either local quantities, i.e. inside a single Cloud node, or pair wise quantities, i.e. involving pairs of connected machines (e.g. link bandwidth, latency, etc.). Complete measurements of pair wise quantities cannot be performed in large-scale systems, as the number of measurement operations required grows quadratically with the size of the Cloud. Several distributed algorithms to predict latencies without global measurement campaigns have been proposed: Vivaldi [11] collects local latency samples and represents nodes as points in a coordinate system. Meridian [12] uses an overlay network to recursively select machines that are the closest to a given network host. Bandwidth estimation in Cloud environments remains an open problem: despite the existence of a number of established techniques [13], most of them are too intrusive and unsuitable for simultaneous use and to perform repeated measurements on high capacity links.

2) Resource abstraction: Modern OS metaphors, such as the “everything is a file” model used by LINUX and Plan9, provide transparent network interfaces and completely hide their properties and specificities from the applications. The major strength of a file-based interface is that it is very flexible and its shortcomings can be supplemented with an appropriate use of naming conventions. We are considering several such mechanisms to present abstracted resource information from measurements to the applications, e.g. via appropriate extensions of the /proc interface or via POSIX-compatible semantic cues [14]. In order to present information about resources to the user applications, the Cloud OS needs first to collect and aggregate them it in a timely way. Clearly, solutions based on centralized databases are not viable, since they lack the fault-tolerance and the scalability we require. The use of distributed systems, such as distributed hash tables (DHTs), has proved to be very effective for publishing and retrieving information in large-scale systems [15], even in presence of considerable levels of churn [16]. However, DHTs offer hash table (key, value) semantic, which are not expressive enough to support more complex queries such as those used while searching for resources. Multi-dimensional DHTs [17][18] and gossip-based approaches [19] extended the base (key, value) semantic in order to allow multi-criteria and range queries.

3) Distributed process and application management: The Cloud OS instantiates and manages all objects that exist across the Cloud nodes. A consolidated practice is the use of virtual machines (VMs), which provide an abstraction that flexibly decouples the “logical” computing resources from the underlying physical Cloud nodes. Virtualization provides

several properties required in a Cloud environment [20], such as the support for multiple OS platforms on the same node and the implicit isolation (up to a certain extent) between processes running on different VMs on the same hardware. Computation elasticity, load balancing, and other optimization requirements introduce the need for dynamic allocation of resources such as the ability to relocate a running process between two nodes in the Cloud. This can be done either at the Cloud process level, i.e. migrating single processes between nodes, or at virtual machine level, i.e. check pointing and restoring the whole VM state on a different node. The combination of process and VM migration, such as it was introduced by MOSIX [21], is very interesting as Cloud functionality as it allows autonomously regrouping and migrating bundles of related Cloud objects with a single logical operation. The Cloud operating system must also provide an interface to manage processes from a user perspective. This requires the introduction of an abstraction to aggregate all the different computational resources is a single view. A good example on how to do this is the recent Unified Execution Model (UEM) proposal [22], which structures the interface as a directory tree similar to the Plan 9 /proc file system and provides an intuitive way to create, copy, and move processes between Cloud nodes. The novelty of the UEM approach is the possibility to "switch the view" on the Cloud process namespace, using a simple extension of common shell built-in commands, inducing a transition e.g. from a per-node view of the process environment to an application-based list of running Cloud processes and objects.

4) Access control and user authentication: Providing seamless support for large numbers of simultaneous users requires a distributed authentication method to avoid single points of failure, resulting in the complete or partial inaccessibility to Cloud resources. Plan 9 provides a very interesting distributed security model [23] which is based on a factotum server, running on every machine that authenticates the users providing a single-sign-on facility based on secure capabilities. Authentication of users by factotum needs to be supported by a Cloud-wide system for securely distributing and storing user credentials, which could be seen as a scaled-up, distributed version of the Plan 9 secstore service.

B. Features provided by the Cloud user space:

In order to fully exploit the potential of a general purpose Cloud OS, developers should be given access to a set of standard ways to satisfy common requirements of distributed large-scale applications. As a general principle, the Cloud libraries provided by the Cloud OS should allow the developers to control the required level of data replication, consistency, and availability, and also the way failure handling is performed when application requirements are not satisfied. This way, an application developer can concentrate her attention on the specific properties of the application, knowing that the system will try its best to accommodate the stated requirements. For instance, when an application demands high availability and is capable of dealing with temporary inconsistencies, the library may provide eventual consistency support, instead of stronger consistency.

V. CONCLUSION

The existence of simple yet powerful and expressive abstractions is essential in realizing the full potential of Cloud Computing. To this purpose we introduced the Cloud operating system, Cloud OS. Cloud OS aims to provide an expressive set of resource management options and metrics to applications to facilitate programming in the Cloud, while at the same time exposing a coherent and unified programming interface to the underlying distributed hardware. This unified interface will provide developers with a quick and transparent access to a massively scalable computing and networking environment, allowing the implementation of robust, elastic, and efficient distributed applications. Our next steps beyond laying out the architecture of Cloud OS include, first, a detailed definition of functional elements and interfaces of the kernel-space Cloud processes and of the user-space libraries, and second, the design and implementation of the aforementioned elements with emphasis on fault-tolerance, security, and elasticity.

REFERENCES

[1] "HP performance-optimized datacenter (POD)." Data Sheet, 2008.
[2] IBM, "Tivoli netview." [Online] http://www01.ibm.com/ software/tivoli/ products/netviw.
[3] I. Foster, C. Kesselman, J. M. Nick, and S. Tuecke, "The physiology of the grid: An open grid services architecture for distributed systems integration," in Open Grid Service Infrastructure WG, Global Grid Forum, 2002.
[4] "Amazon EC2." [Online] http://aws.amazon.com/ec2.
[5] "Amazon S3." [Online] http://aws.amazon.com/s3/.
[6] J. Dean and S. Ghemawat, "Mapreduce: Simplified data processing on large clusters," in *Sixth Symposium on Operating System Design and Implementation (OSDI)*, 2004.
[7] "Apache Hadoop project." [Online] http://hadoop.apache.org/.
[8] R. Gibbens and F. Kelly, "Resource pricing and the evolution of congestion control," *Automatica*, vol. 35, pp. 1969–1985, 1999.
[9] T. Ristenpart, E. Tromer, H. Shacham, and S. Savage, "Hey, you, get off of my cloud: Exploring information leakage in third-party compute clouds," in *Proceedings of the ACM Conference on Computer and Communications Security*, (Chicago, IL), November 2009.
[10] P. A. Dinda and D. R. O'Hallaron, "An extensible toolkit for resource prediction in distributed systems," Tech. Rep. CMU-CS-99-138, School of Computer Science , Carnegie Mellon University, 1999.
[11] F. Dabek, R. Cox, F. Kaashoek, and R. Morris, "Vivaldi: A decentralized network coordinate system," in *In Proc. of ACM SIGCOMM*, 2004.
[12] B. Wong, A. Slivkins, and E. G. Sirer, "Meridian: a lightweight network location service without virtual coordinates," *SIGCOMM Comput. Commun. Rev.*, vol. 35, no. 4, pp. 85–96, 2005.

[13] R. S. Prasad, M. Murray, C. Dovrolis, and K. Claffy, "Bandwidth estimation: Metrics, measurement techniques, and tools," *IEEE Network*, vol. 17, pp. 27–35, 2003.

[14] J. Stribling, Y. Sovran, I. Zhang, X. Pretzer, J. Li, M. F. Kaashoek, and R. Morris, "Flexible, wide-area storage for distributed systems with wheelfs," in *NSDI'09: Proceedings of the 6th USENIX symposium on Networked systems design and implementation*, (Berkeley, CA, USA), pp. 43–58, USENIX Association, 2009.

[15] P. Maymounkov and D. Mazières, "Kademlia: A peer-to-peer information system based on the xor metric," in *IPTPS '01: Revised Papers from the First International Workshop on Peer-to-Peer Systems*, (London, UK), pp. 53–65, Springer-Verlag, 2002.

[16] M. Steiner, T. En-Najjary, and E. W. Biersack, "A global view of KAD," in *IMC 2007, ACM SIGCOMM Internet Measurement Conference, October 23-26, 2007, San Diego, USA*, 10 2007.

[17] P. Ganesan, B. Yang, and H. Garcia Molina, "One torus to rule them all: multi-dimensional queries in p2p systems," in *WebDB '04: Proceedings of the 7th International Workshop on the Web and Databases*, (New York, NY, USA), pp. 19–24, ACM, 2004.

[18] A. Bharambe, M. Agrawal, and S. Seshan, "Mercury: Supporting scalable multi-attribute range queries," in *Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications*, pp. 353–366, 2004.

[19] P. Costa, J. Napper, G. Pierre, and M. V. Steen, "Autonomous Resource Selection for Decentralized Utility Computing," in *Proceedings of the 29th IEEE International Conference on Distributed Computing Systems (ICDCS 2009)*, (Montreal, Canada), June 2009.

[20] R. Figueiredo, P. Dinda, and J. Fortes, "A case for grid computing on virtual machines," in *International Conference on Distributed Computing Systems (ICDCS)*, vol. 23, pp. 550–559, 2003.

[21] T. Maoz, A. Barak, and L. Amar, "Combining virtual machine migration with process migration for hpc on multi-clusters and grids," in *IEEE Cluster 2008, Tsukuba*, 2008.

[22] E. V. Hensbergen, N. P. Evans, and P. Stanley-Marbell, "A unified execution model for cloud computing," in *In Proc. of the 3rd ACM SIGOPS International Workshop on Large Scale Distributed Systems and Middleware (LADIS)*, 2009.

[23] R. Cox, E. Grosse, R. Pike, D. L. Presotto, and S. Quinlan, "Security in plan 9," in *Proceedings of the 11th USENIX Security Symposium*, (Berkeley, CA, USA), pp. 3–16, USENIX Association, 2002.

AUTHOR'S PROFILE

Lokesh Patel
Student of M Tech (CSE Dept.)
SSSIST Sehore
g.patelokesh01@gmail.com

Gajendra Singh
Professor
SSSIST Sehore
gajendrasingh86@rediffmail.com

Ravindra Gupta
Assistant Professor
SSSIST Sehore
ravindra_p84@rediffmail.com