Trends in Cloud Operating System

Sanil C. Savale
Department of Computer Science,
Gogate Jogalekar College, Ratangiri

ABSTRACT

Cloud is a simplified operating system that runs just a web browser, providing access to a variety of web-based applications that allow the user to perform many simple tasks without booting a full-scale operating system. Because of its simplicity, Cloud can boot in just a few seconds. The operating system is designed for Mobile, and PCs that are mainly used to browse the Internet. This paper also focuses on various issues characteristics of cloud Operating System. Paper also Focus on requirements of cloud OS. It gives the importance of cloud operating system in market. It also gives implementation of the cloud kernel processes

Keywords

Cloud OS, Kernel

INTRODUCTION

Cloud is a simplified operating system that runs just a web browser, providing access to a variety of web-based applications that allow the user to perform many simple tasks without booting a full-scale operating system. Because of its simplicity, Cloud can boot in just a few seconds. The operating system is designed for Mobile, and PCs that are mainly used to browse the Internet.[5]

Cloud can be installed and used together with other operating systems, or can act as a standalone operating system. When used as a standalone operating system, hardware requirements are relatively low.Cloud OS manages user account. Cloud OS include simple Text editor, Paint application, spreadsheet and a presentation viewer that you can use. It includes Calendar, Contact module, and simple games. Use for Remote application management.

1. ASSUMPTIONS ON CLOUD NFRASTRUCTURE

A Cloud is a logical entity composed of managed computingresources deployed in private facilities and interconnected over a public network, such as the Internet. Cloud machines (also called nodes) are comprised of inexpensive, off-the shelf consumer-grade hardware. Clouds are comprised of a large number of clusters (i.e. sets of nodes contained in a same facility) whose size may range from a few machines to entire datacenters. Clusters may use sealed enclosures or be placed into secluded locations that might not be accessible on a regular basis, a factor that hinders access and maintenance activities. Clusters are sparsely hosted in a number of locations

While a traditional OS is a piece of software that manages the hardware devices present in a computer, the Cloud OS is a set of distributed processes whose purpose is the management of Cloud resources. Analogies to established concepts can therefore help us to describe the kind of features and interfaces we wish to have in aCloud OS, ignoring for the moment the obvious differences of scale and implementation between the two scenarios:

- an OS is a collection of routines (scheduler, virtual memory allocator, file system code, interrupt handlers,etc.) that regulate the access by software to CPU, memory, disk, and other hardware peripherals; the Cloud OS provides an additional set of functionalities that give administrative access to resources in the Cloud: allocate and deallocate virtual machines, dispatch and migrate

processes, setup inter-process communication, etc. an OS provides a standard library of system calls which programs can use to interact with the underlying hardware; the Cloud OS provides a set of network-based interfaces that applications can use to query the management

system and control Cloud resources.

- an OS includes a standard distribution of libraries and software packages; the Cloud OS includes software support for the autonomous scaling and opportunistic deployment of distributed applications[8]

2. TOWARD SEAMLESS ACCESS TO NETWORKED RESOURCES

In this section, we present the architecture and functional building blocks of the Cloud OS. Our current design approach leverages decades of experience in building networked systems, from the origins of the Internet architecture[19] to subsequent achievements in distributed operating systems research [13] and large-scale network testbed administration [20]. An additional inspiration, especially concerning the implementation of Cloud OS, comes from the last decade of advances in distributed algorithms and peer-to-peer systems.

Logical architecture of the cloud os

Figure 1 represents a logical model of Cloud OS. We define the Cloud object as a set of local OS processes running on an 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 regulate physical allocation, access control, accounting, and measurements of resources as the Cloud kernel space. Those CPs that do not belong to kernel space pertain to the Cloud user 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 Calls2. The assumptions stated above pose very few constraints about the features that the underlying 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 Cloud3 . 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.

3. CLOUD OS REQUIREMENTS

Whereas 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. [7]

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 testbeds. 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 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 .

4. IMPLEMENTATION OF THE CLOUD KERNEL PROCESSES

a)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) in clouds that span several datacenters.measurements can target either local quantities, i.e. inside a single cloud node, or pairwise quantities, i.e. involving pairs of connected machines (e.g. link bandwidth, latency,etc.). complete measurements of pairwise 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. meridian

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, most of them are too intrusive and unsuitable for simultaneous use and to perform repeated measurements on high capacity links.

b)Resource abstraction:

Modern OS metaphors, such as the "everything is a file" model used by UNIX and Plan9, provide transparent network interfaces and completely hide their properties and pecificities from the applications. However, characterizing the underlying network is a crucial exigence for a Cloud OS, for network properties such as pairwise latencies, available bandwidth, etc., determine the ability of distributed applications to efficiently exploit the available resources. One 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.

c)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 [31], 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.

d)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.

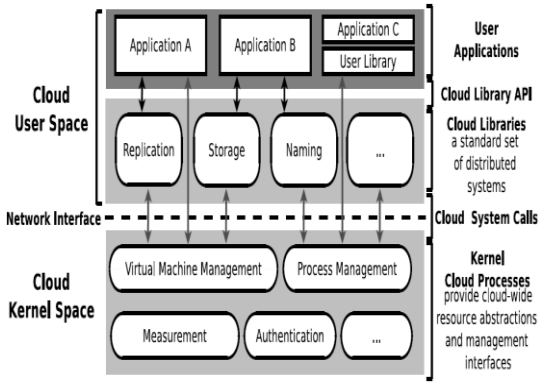

Figure 1. Logical model of Cloud OS, featuring the division between Cloud kernel / Cloud user space and the system call and library API interfaces.

5. 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. Cloud libraries provide a standard API with features such as:

- access to Cloud-wide object and process naming via DNS and/or other distributed naming services
- distributed reliable storage functionality
- automated Cloud application deployment, horizontal scaling, and lifecycle management
- high availability failover support with check pointed replicated process execution.

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. The advantages of this configurability are twofold. On one hand, an application developer doesn't need to implement her own customized (application-specific) libraries, but instead can customize the level of support she needs from the Cloud library API. This reduces the complexity and error proneness network protocols. This way, the usability of our Cloud OS will not be restricted by technology lock-in[17]

6. FUTURE DIRECTIONS

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] Cloud Computing: Web-Based Applications That Change the Way You Work and Collaborate Online, Michael Miller

[2] Cloud Computing: Principles and Paradigms The Enterprise Cloud Computing Paradigm, by Benoit Hudzia

[3] Cloud Computing for Dummies by Judith Hurwitz, Marcia Kaufman & Fern Halper

[4] Cloud Security & Privacy An Enterprise Perspective on Risks and Compliance by Subra Kumaraswamy , Tim Mather & Shahed Latif

[5] Power in the Cloud Building Information Systems at the Edge of Chaos by Jonathan Sapir

[6] " EyeOS" [Online] http://eyeos.org/

[7]" vmware " [Online] http://www.vmware.com/products/cloud-os/

[8]"Google AppEngine." [Online] http://appengine.google.com.

[9] "wikipedia " [Online] http://en.wikipedia.org/wiki/Google_Chrome_OS

[10] Fabio Pianese ,Peter Bosch ,Alessandro Duminuco , Nico Janssens ,Thanos Stathopoulosy ,Moritz Steiner, Toward a Cloud Operating System ,Alcatel-Lucent Bell Labs, Service Infrastructure Research Dept. yComputer Systems and Security Dept.

[11] R. Gibbens and F. Kelly, "Resource pricing and the evolution of congestion control," Automatica, vol. 35, pp. 1969–1985, 1999.

[12] 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.

[13] D. D. Clark, "The design philosophy of the darpa internet protocols," in Proc. SIGCOMM '88, Computer Communication Review Vol. 18, No.4, August 1988, pp. 106–114, 1988.

[14] L. Peterson, A. Bavier, M. E. Fiuczynski, and S. Muir, "Experiences building Planetlab," in In Proceedings of the 7th USENIX Symp. On Operating Systems Design and Implementation (OSDI), 2006.

[15] P. A. Dinda and D. R. O'Hallaron, "An extensible toolkit for resource prediction in distributed systems," Tech. Re CMU-CS-99-138, School of Computer Science , Carnegie Mellon University, 1999.

[16] F. Dabek, R. Cox, F. Kaashoek, and R. Morris, "Vivaldi: A decentralized network coordinate system," in In Proc. of ACM SIGCOMM, 2004.

[17] R. Gibbens and F. Kelly, "Resource pricing and the evolution of congestion control," Automatica, vol. 35, pp. 1969–1985, 1999.

[18] 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.

[19] D. D. Clark, "The design philosophy of the darpa internet protocols," in Proc. SIGCOMM '88, Computer Communication Review Vol. 18, No.4, August 1988, pp. 106–114, 1988.

[20] L. Peterson, A. Bavier, M. E. Fiuczynski, and S. Muir,\ "Experiences building Planetlab," in In Proceedings of the 7th USENIX Symp. On Operating Systems Design and Implementation (OSDI), 2006.

[21] 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.

[22] F. Dabek, R. Cox, F. Kaashoek, and R. Morris, "Vivaldi: A Decentralized network coordinate system," in In Proc. of ACM SIGCOMM, 2004.

[23] 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.

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

[25] 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.

[26] 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.

[27] 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.

[28] P. Ganesan, B. Yang, and H. Garcia-Molina, "One torus to Rule of the 7th International Workshop on the Web and Databases, (NewYork, NY, USA), pp. 19–24, ACM, 2004.

[29] 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,

[30] 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.

[31] 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.

[32] V. Ramasubramanian and E. G. Sirer, "The design and Implementation of a next generation name service for the internet," in SIGCOMM '04: Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications, (New York, NY, USA), pp. 331–342, ACM, 2004.

[33] G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman,A. Pilchin, S. Sivasubramanian, P. Vosshall, and W. Vogels, "Dynamo:Amazon's highly available key-value store," SIGOPS Oper. Syst. Rev.,vol. 41, no. 6, pp. 205–220, 2007.

[34] P. Goldsack, J. Guijarro, S. Loughran, A. Coles, A. Farrell, A. Lain, P. Murray, and P. Toft, "The smartfrog configuration Management framework," SIGOPS Oper. Syst. Rev., vol. 43, no. 1, pp. 16–25, 2009.

[35] E. M. Maximilien, A. Ranabahu, R. Engehausen, and L. C. Anderson,"Toward cloud-agnostic middlewares," in Proceedings of OOPSLA '09, (New York, NY, USA), pp. 619–626, ACM, 2009.

[36] E. M. Maximilien, A. Ranabahu, R. Engehausen, and L. C. Anderson, "IBM Altocumulus: a cross-cloud middleware and platform," in Proceedings of OOPSLA '09, (New York, NY, USA), pp. 805–806, ACM,2009.

[37] M. Coppola, Y. Jegou, B. Matthews, C. Morin, L. Prieto, O. Sanchez, E. Yang, and H. Yu, "Virtual organization support within a gridwide operating system," Internet Computing, IEEE, vol. 12, pp. 20–28,March-April 2008.

[38] T. DeWitt, T. Gross, B. Lowekamp, N. Miller, P. Steenkiste,J. Subhlok, and D. Sutherland, "Remos: A resource monitoring system for network aware applications," Tech. Rep. CMU-CS-97-194, School Computer Science, Carnegie Mellon University, Pittsburgh, PA, 1997.

[39] P. Chandra, A. Fisher, C. Kosak, T. Ng, P. Steenkiste, E. Takahashi, and H. Zhang, "Darwin: Customizable resource management for value-added network services," IEEE NETWORK, vol. 15, pp. 22–35, 2001.

[40] R. Van Renesse, K. Birman, D. Dumitriu, and W. Vogel, "Scalablemanagement and data mining using Astrolabe," in Proceedings of theFirst International Workshop on Peer-to-Peer Systems (IPTPS'01), 2001.

[45] J. Albrecht, D. Oppenheimer, D. Patterson, and A. Vahdat, "Design andImplementation Tradeoffs for Wide-Area Resource Discovery," ACMTransactions on Internet Technology (TOIT), vol. 8, September 2008.

[41] E. K. Lua, J. Crowcroft, M. Pias, R. Sharma, and S. Lim, "A surveyand comparison of peer-to-peer overlay network schemes," IEEE Communications Survey and Tutorial, vol 7, pp. 72–93, March 2004.

[42] SETI@Home. [Online] http://setiathome.berkeley.edu.

[43] BOINC. [Online] http://boinc.berkeley.edu.

[44] G. Fox, D. Gannon, S.-H. Ko, S. Lee, S. Pallickara, M. Pierce, X. Qiu,X. Rao, A. Uyar, M. Wang, and W. Wu, Peer-to-Peer Grids. John Wileyand Sons Ltd, 2003.

[45] I. Foster and A. Iamnitchi, "On death, taxes, and the convergence ofpeer-to-peer and grid computing," in In 2nd International Workshop on Peer-to-Peer Systems (IPTPS), pp. 118–128, 2003.

[46] N. Drost, R. van Nieuwpoort, and H. Bal, "Simple locality aware coallocation in peer-to-peer supercomputing," in Proceedings of the 6th IEEE/ACM CCGrid Symposium, 2006.

[47] J. Ritter, "Why gnutella can't scale. No, really." [Online] http://www.darkridge.com/ jpr5/doc/gnutella.html., Feb 2001.

[48] J. Albrecht, R. Braud, D. Dao, N. Topilski, C. Tuttle, A. C. Snoeren, and A. Vahdat, "Remote Control: Distributed Application Configuration, Management, and Visualization with Plush," in Proceedings of the Twenty-first USENIX LISA Conference, November 2007.