Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X

IJCSMC, Vol. 2, Issue. 5, May 2013, pg.6 – 13

RESEARCH ARTICLE

Cloud on Linux Operating System

Pritee Gavhane[1], Pratibha Kurumkar[2], Madhuri Jagtap[3], Dharti Mahajan[4]

[1]Near P.R.E.C. College Loni, Dist-Ahmednagar, India

[2]At post- Madewadgao, Tal-Shrigonda, Dist-Ahmednagar, India

[3]Near Premdan Ahmednagar, Dist-Ahmednagar, India

[4]Mondha road Ghatnandur, Tal-Ambejogai, Dist-Beed, India

[1] gavhanepritee@gmail.com; [2] pratibhakurumkar38@gmail.com;
[3] madhujagtap@gmail.com; [4] mahajan.dharati32@gmail.com

Abstract— The term "cloud" is analyzed to "Internet". The term "Cloud computing" is based on cloud drawings used in the past to represent telephone networks and later to depict internet. Cloud computing is a internet based computing where virtual shared servers provide software, infrastructure, platform, devices and other resources and hosting to customers on a pay-as-you basis. All information that a digitized system has to offer is a provided as a service in the cloud computing model. Users can access these services available on the "Internet cloud" without having any previous know - how on managing the resources involved. Cloud computing is a computing model not a technology. In this model customers plug into the "cloud" to access IT resources which are price and provided "on demand".

Cloud computing is a style of computing where massively scalable IT related capabilities are provided "as a service" to external customers using internet technologies. Virtualization is the ability to run multiple OS (software's) on a single physical systems and share the underlying a hardware resources. Cloud computing provide services which are mainly of an OS running on a single virtualized computing environment with middleware layers that attempt to combine physical and virtualized resources from multiple OS and specialized application engines. This paper, we discuss about the virtual distributed OS. Cloud OS, which provide the maximum features cloud computing by using c panel. The cloud OS works on the principles layout by LINUX. The cloud OS aims to provide a simple programming abstraction to available cloud resources, strong isolation techniques between cloud processes and strong integration with network resources.

Key Terms: - LINUX; cloud computing; cloud OS; cloud server; c panel; logical architecture

I. INTRODUCTION

Cloud computing is a modality for providing computer facilities via the internet. Cloud computing provides a three delivery model a) SaaS – Software as Service it uses software applications over the cloud, b) PaaS- Platform as Service it deploys user created applications to a cloud, c) IaaS- Infrastructure as a Service it is used to give storage and computing resources on rent. These three delivery model provides support to on- demand services, vast network access, location independence, Rapid scalability, pay per use.

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.

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 communication 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.

A cloud server is a process that runs on a hypervisor. This hypervisor software lets a single computer appear like it is a dozen or more computers. A single cloud server is just one of these processes. It appears to the user like a server, but it isn't its own collection of hardware [1]. Cloud servers can be scaled up and down with minimal fuss, using software only. You can increase or decrease RAM, Hard Drive Space and even CPU power all dynamically using a software interface [1]. Cloud servers can be backed up in their entirely with very little fuss. Cloud server does not require maintenance once automated backups are configured [1]. Cloud servers are same as virtual servers [1]. A cloud server is certainly easier to move between data centers. Cloud server can be transferred over the Internet [1].

The widespread use of LINUX in the cloud benefits both those who run and operate clouds, as well as those who build upon them. Linux is the natural technology for enabling cloud computing: its modular, its performance, its power efficient, it scales, its open source and its ubiquitou. "Every time you use Google, you are using machine running the Linux kernel", as Google's Chris Dibona has said. "Linux today support more hardware devices than any other OS in the history of world".

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 paper, we argue for a holistic approach to Cloud computing that transcends the limits of individual machines. Our aim is 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 [main]. In section 2nd we elaborate the related work and vision of cloud OS, discuss our requirements we aim to meet. In section 3rd we present the logical architecture of cloud OS. We then briefly discuss the working that is our proposed scheme in section 4th and conclude in section 5th.

II. LOGICAL ARCHITECTURE OF CLOUD OS

Fig. 1 represents the proposed Cloud Operating System with various layers and the components in each layer are also specified in their respective layers .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, is used to minimize the risk of system-wide ID collisions. A Cloud process (CP) is a collection of Cloud objects that implement the same, i.e. similar to the distributed application. Here we refer to the small number of CPs that regulates physical allocation, access control, accounting, and measurements of resources as the Cloud kernel space. The CPs that are executed in the User space directly by users and this CPs are called User Applications. The Cloud Libraries are the CPs that is 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. Assumptions stated above pose very few constraints about the features that the underlying Cloud hardware which is expected to provide[8].

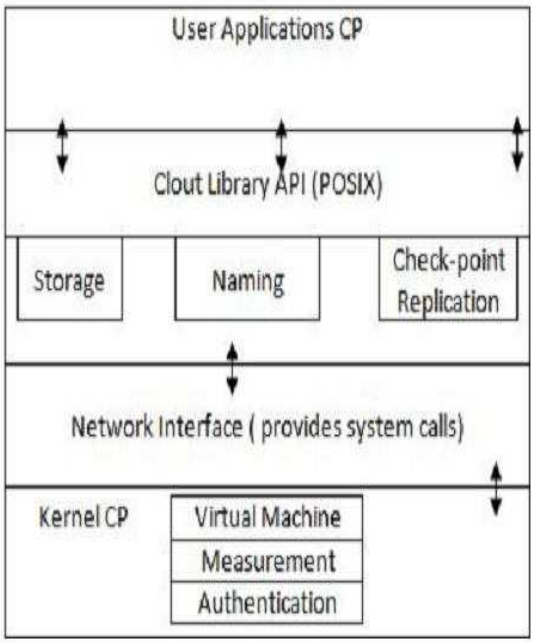

Fig. 1 Logical architecture of cloud OS

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 also achievable from machines that belong to administrative domains other than that of the Cloud provider, with the 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[9].

III. CLOUD OS WORKING

A. Implementation of the Cloud Kernel Processes:

(1) Resource Measurement : The Cloud OS needs to maintain an approximate view of the available Cloud resources. The 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, 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 such as 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 are used 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. Here 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, 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 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 sec store 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. An application developer can concentrate their attention on the specific properties of the application, knowing that the system will try its best to accommodate the stated requirements. For an 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.

IV. OUR PROPOSED SCHEME

In this section, we provide an example of our approach along with the data structures and system calls.

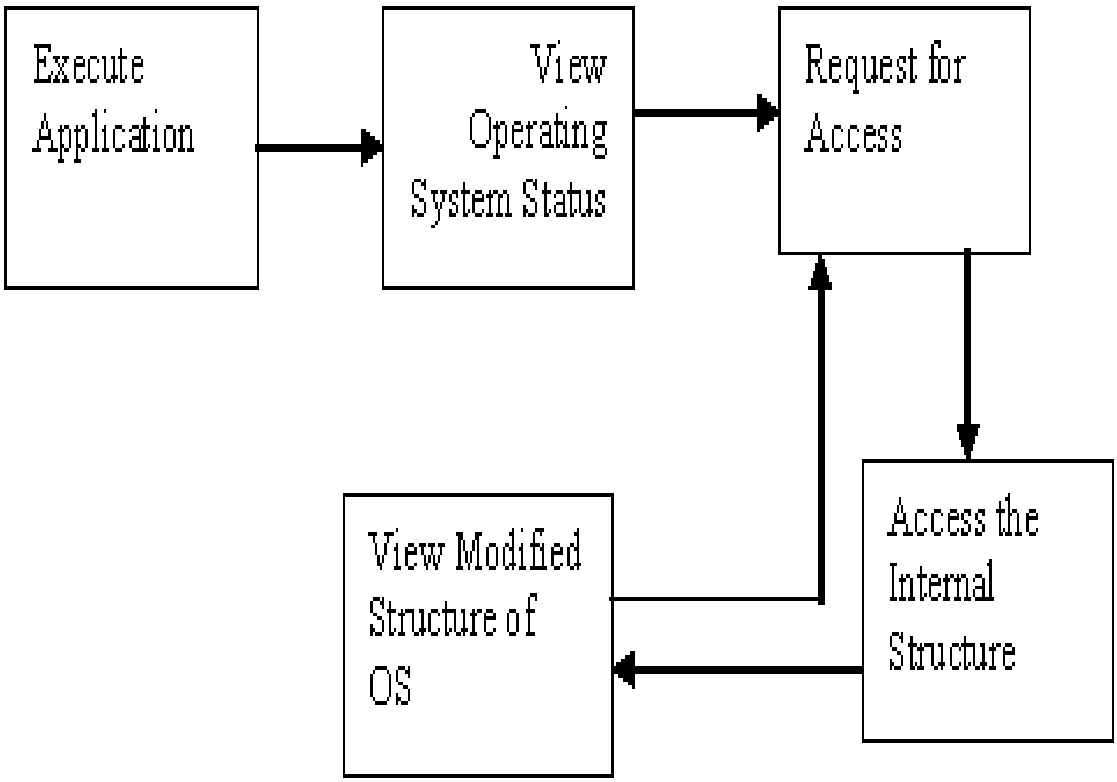

Fig. 2 our proposed scheme

The Linux based cloud server system consist of c-panel, user can execute their applications by using c-panel. Essentially calls are made within programs and checked copy of the request is pass through the system calls. Operating systems or cloud servers status can view through the system calls request will be made for accessing the services then internal structure of server or operating system will get modified and that will be displayed through the c-panel.

System calls are low level functions of the operating system makes available to applications via defined API. System calls represents the interface the kernel present to user applications. The system call is the fundamental interface between an application and the Linux kernel. System calls act as an entry point to OS kernel. There are certain tasks that can only be done it a process is running in kernel mode. Example of these tasks can be interacting hardware etc. So if process wants to do such kind of task then it would require itself to be running in kernel mode which is made possible by system calls.

V. FLOW OF PROPOSED SYSTEM

Fig.3 shows the flow of system. For authentication purpose username and password are used to enter in the system.

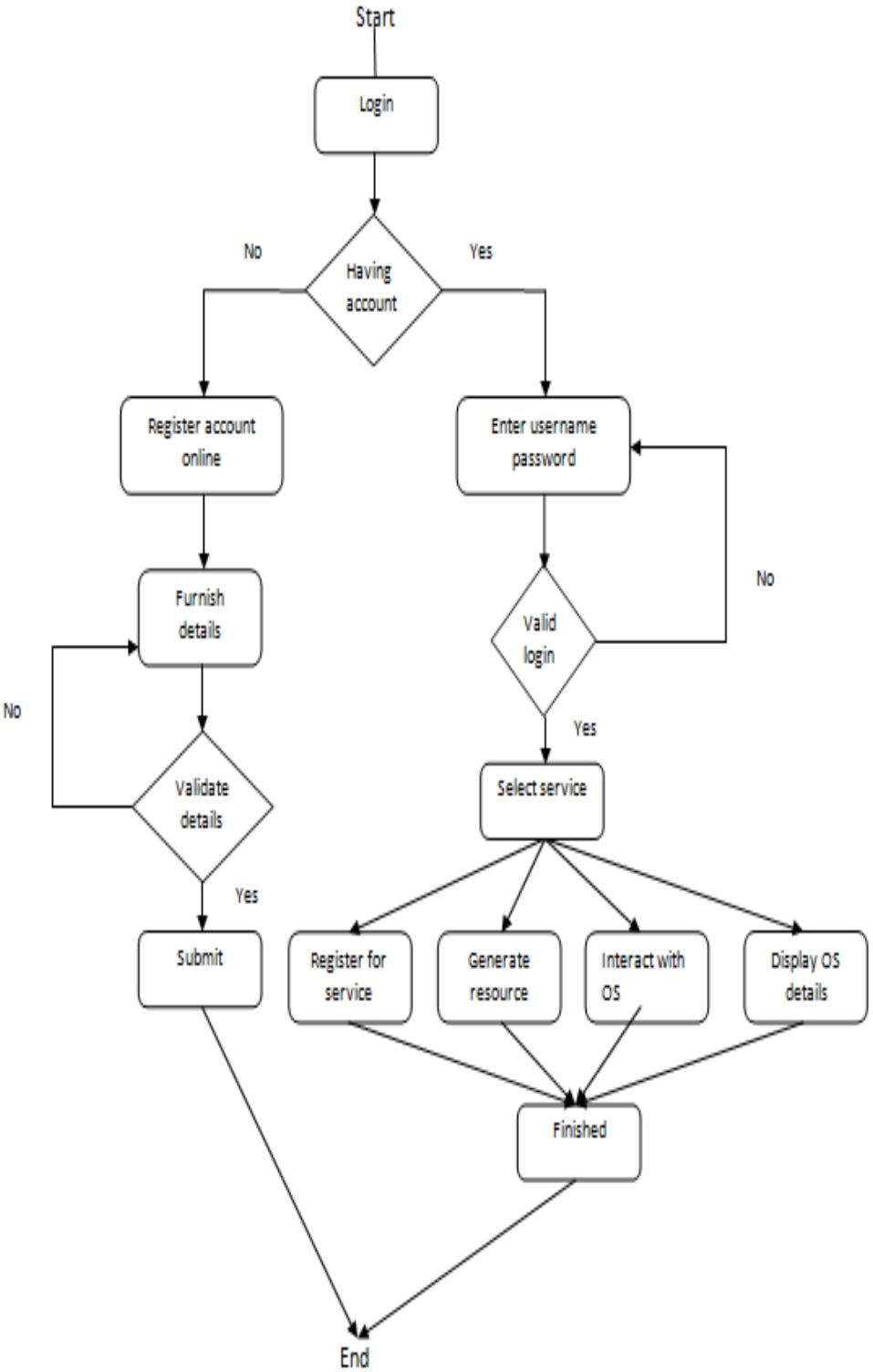

Fig. 3 Flow of proposed system

VI. CONCLUSION AND FUTURE WORKS

In this paper we propose a Linux based cloud server system to control the applications over the cloud computing.

We introduced the Cloud operating system, Cloud OS to make cloud computing more powerful. Cloud OS aims to provide an expressive set of resource management options and metrics to applications to facilitate programming in the Cloud. So the cloud OS provide quick, flexible and secure access to computing and networking environment. The green computing is also achieved by the cloud OS.

REFERENCES

[1] White paper: Linux: The operating system of cloud March 2009

[2] IBM, "Tivoli netview." [O nline] 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." [On line] http://hadoop.apache.org/.

[8] 2-P-Ashok kumar-922_research_communication_CSIT_september_2012.pdf

[9] Lokesh Patel, Gajendra Singh, and Ravindra Gupta "linux based cloud operating system" in International Journal of Engineering Innovation & Research, 2012 Volume 1, Issue 1, ISSN : 2277 – 5668

[10] 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.

[11] 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.

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

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

