Survey on Cloud Operating System

Vaishali Gupta[1]
Department Of Computer Science and Engineering
Santa Clara University
Santa Clara, California
vgupta2@scu.edu

Anjaly George[2]
Department Of Computer Science and Engineering
Santa Clara University
Santa Clara, California
ageorge2@scu.edu

Anaja Bajpeyi [3]
Department Of Computer Science and Engineering
Santa Clara University
Santa Clara, California
abajpeyi@scu.edu

Ramya Umamaheswaran[4]
Department Of Computer Science and Engineering
Santa Clara University
Santa Clara, California
rumamaheswaran@scu.edu

***Abstract*—Cloud computing being the present and future of the IT industry imposes great benefits and for those seeking IT solutions different cloud services and deployment models are available. To benefit from the cloud capabilities, special OS needs to be designed that can handle the great demands of the cloud. The aim of this survey is to understand what is cloud operating systems and its architecture, the differences between traditional operating systems and cloud operating systems. To get a grip on how they work in real life, we present a few applications of the cloud technology and explain their architectures going over the advantages and limitations of cloud operating systems and present the possible improvements using Docker, X11 and edge computing.**

***Index Terms*—Cloud Operating System, API, Google AppEngine, Eye OS, Docker, X11, Edge Computing**

I. Introduction

Cloud environment is a huge environment, many components and technologies are used to manage a cloud, these components and technologies together is known as Cloud Operating System. Generally the platform components running inside the cloud providers are considered as cloud operating systems (OS). A cloud operating system is responsible for managing the operation, execution and processes of virtual machines, virtual servers and virtual infrastructure,as well as the back-end hardware and software resources. Depending on the virtual environment and cloud services in use, the functionality of cloud operating systems varies. The paper is assembled as follows. In Section II we explain the architecture of cloud OS. Then we discuss the comparison of cloud OS and traditional OS in section III. In Section IV we cover the applications of cloud OS followed by advantages and limitations. Finally we discuss the ongoing improvements and give the conclusion.

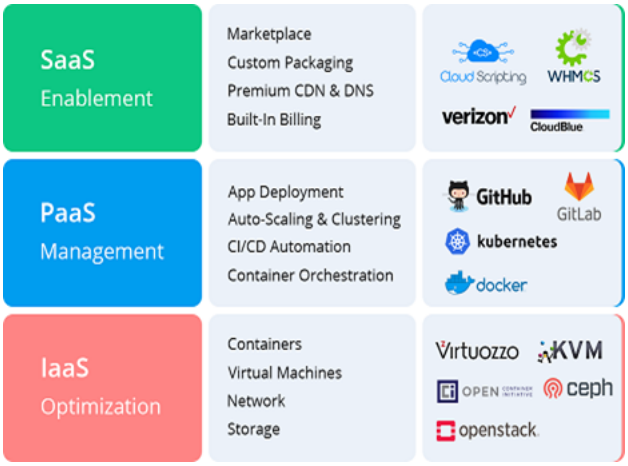

Fig. 1. Layers of cloud Computing Architecture

II. Cloud OS Architecture

Cloud OS components are divided mainly into three levels in any cloud ecosystem. Each level services different purposes.

A. *Infrastructure Layer*

The very first level i.e. physical infrastructure makes use of virtualization technologies mainly Hardware Virtualization which includes VM, virtual networking[3], and virtualized storage. Hardware level virtualization makes use of Paravirtualization, Partial virtualization. Virtual machines migration techniques to improve resource utilization, application isolation, load balancing and potability of processing nodes, fault tolerance in virtual machines, and to raise the efficiency of the physical server. Virtualization decouples this level from higher levels which leads to the evolution of the physical resources on their own without disturbing the software stacks running on top.

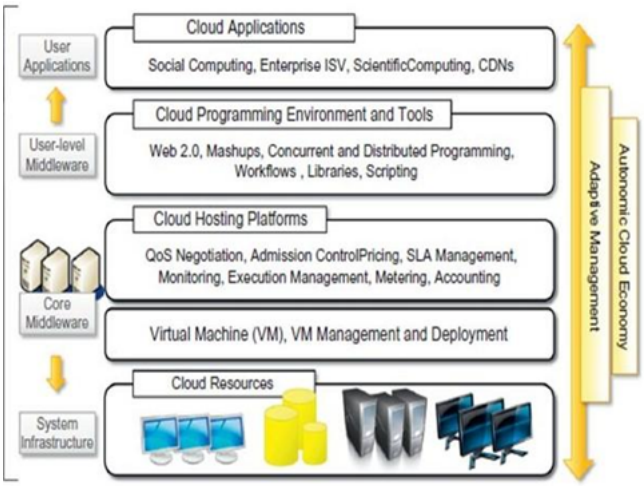

Fig. 2. Architecture of Cloud OS from Virtualisation point of view

B. *Middleware*

1) **Core Middleware Software Management Layer**
A core middleware layer also known as software management layer[4] manages the underlying virtualized hardware. It contains a scheduler which handles the allocation and execution of virtual machine instances by interacting with other components like VM Pool manager, VM repository, reservation, monitoring, QoS/SLA management, accounting billing components for taking decisions according to defined scheduling policies.

2) **User Level Middleware**

Next layer is User level middleware which offer developers a platform including Web-based interfaces, command-line tools and frameworks for concurrent and distributed programming[5]. Developers use these APIs to develop their applications specifically for the cloud. These advanced web interfaces allow integration of platform solutions into the software management layer. Some examples of such solutions are OpenNebula, OpenStack. Cloud applications which do not need to run in the virtualized environment for the reason of performance overhead use the frameworks provided in this layer.

C. *User Interface/Application*

The top layer known as User Applications contains services delivered at the application level. In most cases these are Web-based applications that rely on the cloud(underlying components and technologies) to provide service to end users. The horsepower of the cloud provided by IaaS and PaaS solutions allows independent software vendors to deliver their application services over the Internet.

The proposed architecture is the basic architecture of Cloud OS. Different solutions can provide additional services or even not provide support for some of the services discussed here. Finally, the reference architecture applies to IaaS implementations that provide computing resources, especially for the scheduling. If storage is the main service provided.

- The role of infrastructure management software is not to keep track and manage the execution of virtual machines but to provide access to large infrastructures and implement storage virtualization and distributed computing solutions on top of the physical layer.

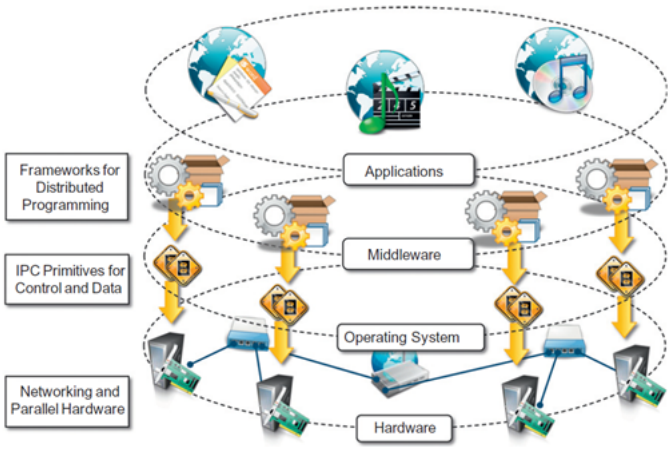

Fig. 3. Architecture of Cloud OS from Distributed Systems point of view

- Hardware Virtualization solutions along with storage and network virtualization strategies allows the infrastructure to be completely virtualized and controlled. Unification of parallel and distributed computing then on higher levels allows one to harness a set of networked and heterogeneous computational nodes and present them as a unified resource. These nodes use IPC services like RPC and standardized communication protocols(TCP/IP) to communicate with each other. Various models and architectures are introduced by parallel and distributed computing for performing [6] multiple tasks within a set of tightly coupled nodes with homogeneous hardware or a single computing node.These architectures exploit parallelism to increase the performance of a computing system, depending on whether parallelism is realized on instructions, data, or both [7-10]. Specific environments and compilers are required that provide transparent access to the advanced capabilities of the underlying architectures thereby leading to development of parallel applications. By relying on the services offered by the operating system, the middleware develops its own protocols, data formats, and programming languages or frameworks for the development of distributed applications [11-12].
- MapReduce [13,14] proposed by Google first is an example of such an application. It provides loosely coupled framework dealing with a large number of heterogeneous and unreliable nodes and is used to build data processing

applications like scheduling tasks, monitoring and re-executing any failed tasks. All of them constitute a uniform interface to distributed application developers that is completely independent from the underlying operating system and hides all the heterogeneities of the bottom layers.

III. COMPARISON BETWEEN CLOUD OS AND TRADITIONAL OS

A. *Key Differences*

1) Use API Instead of system calls.
2) Schedules distributed jobs, while traditional OS schedules processes/threads.
3) The cloud OS uses object storage as opposed to the file system in traditional Operating System
4) Cloud applications are built on distributed, NoSQL databases which can scale up or down with the variation in load and offer better fault tolerance as compared to traditional Operating System.
5) Cloud systems follows service-oriented architecture, while traditional Operating System follows layered system architecture.

B. *API VS System Call*

An API is a set of protocols, routines, functions that programmers use to interact between distinct systems It is an interface that takes the requests from the user and informs the system about what should be done and return the response back to the user.

Classification of cloud API:

1) **PaaS APIs** paaS APIs are the service level API in cloud Operating System.
   Paas APIs are used to provide access and functionality for a cloud environment
2) **SaaS APIs** SaaS APIs are the application level API in cloud Operating System.
   They connect the application-layer with the cloud and underlying IT infrastructure
3) **IaaS APIs** IaaS APIs are the Infrastructure level API in cloud Operating System.
   These APIs control the cloud resources and how they are to be distributed.
4) **Cloud provider and cross-platform APIs** These APIs are used to allow end users enjoy cloud versatility

In a cloud operating system, API allows sharing data among different applications and distributed devices, while traditional Operating System use System calls to request services from the kernel.

C. *Object Storage*

Cloud Operating system uses Object storage in place of File System in traditional Operating System. In system that uses Object Storage each piece of data is considered as an object. The data is kept in separate storehouses and is bundled with its metadata and a unique identifier to form a storage pool. In Object storage there is no limit for the storage.The major benefits the cloud operating system achieves by using object storage are greater data analytics, infinite scalability, faster data retrieval, optimized use of resources and a reduction in cost spent on data storage.The areas in which object storage can be used are in places where we need to manage distributed content, deliver rich media and in Internet of Things.

EXAMPLES:

1) Amazon S3 Storage:
   The Amazon S3 API offers a common path for rapid development and the creation of hybrid cloud deployments at scale
2) NetApp StorageGRID :
   Object storage built for a hybrid, multi-cloud experience

D. *Service Oriented Architecture*

Service-oriented architecture (SOA) is a software design method where services are provided to the other components by application components, through a communication protocol over a network. The fundamental principles of service-oriented architecture is that they are independent of vendors technologies and products. A service is a discrete unit of functionality that can be accessed remotely and acted upon and updated independently, such as retrieving a credit card statement online.

The four properties of the services are listed below:

1) It logically represents a business activity with a specified outcome.
2) It is self-contained.
3) It is a black box for its consumers.
4) It may consist of other underlying services.

- **COMPONENTS OF SOA**
  The guiding principle to develop an application using service oriented architecture are, it should have a standardized service contract, the services should have minimum dependency on other services that is the services should be self sufficient, the service should hide its logic and act as black box, the services should be reusable etc.

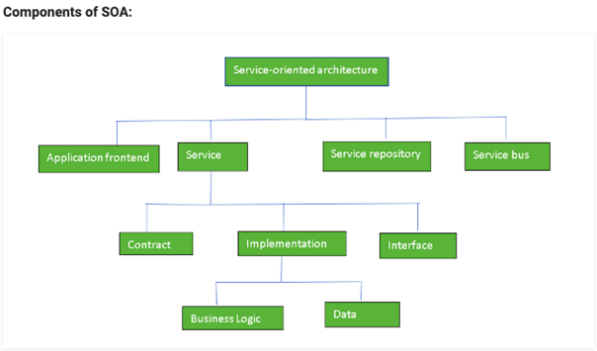

Fig. 4. Components of Service Oriented Architecture

The major advantages of designing a system in software oriented architecture are, the ability to reuse the services, since the services are loosely coupled they can be easily maintained, services are platform independent, they are more reliable because it is easy to debug smaller components and also they are highly scalable. They also have a few disadvantages which are the high overload that occur as a result of validation of input parameter of services which is done whenever a service interacts, also SOA requires a huge initial investment and managing the service interaction can become cumbersome considering the multitude of messages exchanged.

E. Scheduling in Cloud OS

Scheduling methods in cloud system can be categorized into three classes:

1) Workflow scheduling
2) Resource scheduling
3) Job Scheduling

JOB SCHEDULING

Job scheduling is a procedure of how jobs should be run in the system in terms of actual mapping of resource elements and time when the jobs get executed. It usually shares the resource or multiplexes at different levels.

- **STATIC SCHEDULING**
  Static scheduling knows that all information about the available resource and job requirements and after that assigns jobs into appropriate resource. Also, static scheduling assumes that no failure of jobs and resources are occurring all the time.
  1) **Min-Min Algorithm:**
     This scheduling algorithm is used in grid and cloud computing environments. The advantage of the algorithm is that they minimize makespan,cost and maximize resource utilization. The algorithm is executed by selecting a cloudlet in the cloudletlist with the lowest execution time and assign it to virtual machine that produces its minimum completion time. This algorithm gives priority to tasks with smallest completion time. The usage of Min-Min algorithm can lead to an increase of total response time of the system when cloudlets with minimum completion time are much more in number, which is considered as a disadvantage of the strategy.
  2) **Max-Min Algorithm:**
     This algorithm implements the opposite idea of Min-Min algorithm. Max-Min select and assign cloudlets with maximum completion time first.Max-Min algorithm gave priority to cloudlets with largest completion time.The disadvantage of the algorithm is that it sometimes leaves the short tasks unattended for a long time before execution.
  3) **Round Robin:**
     In Round Robin, a dedicated time slot is allocated to each job waiting in a queue to be scheduled by the scheduler This means that, no task will be allocated to a resource for more than the allocated time and if a task is not able to complete within the allocated time, it will be preempted by another task and sent back in the queue to give way for other tasks.
  4) **First Come First Served (FCFS) Algorithm:**
     FCFS assigns tasks to resources based on their arrival time. It works in non-preemptive fashion. One disadvantage of FCFS is that, its turnaround and response time is quite low because tasks are not preempted. The smaller tasks with the shortest execution time in the queue may have to wait for the larger task with large execution time ahead to finish.

IV. APPLICATIONS

A. Google AppEngine

Google AppEngine is a Platform as a service implementation. Its main goal is to provide scalable applications to increase their efficiency and performance. For applications that are facing a large number of requests, the evaluation of the load is done and resources are allocated if necessary. Developers who develop the applications can use languages such as Java, Python etc.

ARCHITECTURE

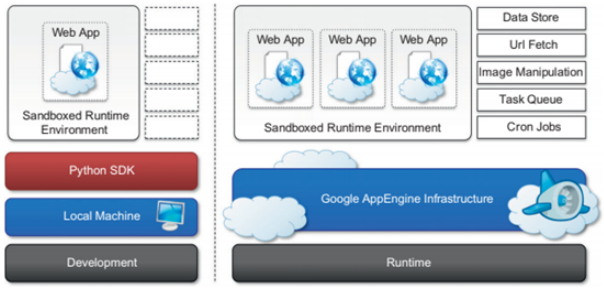

Fig. 5. Architecture of Google AppEngine

- The Google App Engine is primarily divided into 4 components
  1) Infrastructure
  2) Run-time environment
  3) Storage
  4) Scalable services

- **Infrastructure**
  One of infrastructure's main responsibility is to monitor the application performance. It does this by taking advantage of the servers which are present in google data centers. Whenever a request comes, infrastructure

will locates the serves of the application in which the request was processed. Now, it evaluates the load and will allocate additional resources based on the evaluation.

- **Run-time Environment**
  The runtime environment is responsible for the execution of the application. It comes into existence from the time the request handler starts execution until its execution is completion and it is terminated.

  - **Sandboxing**
    Sandboxing is a technology used by the runtime environment to protect the application from getting influenced by the execution of other applications. Applications basically run on an isolated context for security reasons.

- **Storage**
  The different types of storage for Google AppEngine are:
  1) In memory-cache
  2) Storage for semistructured data
  3) Long-term storage for static data

- **Static file servers** For any, graphical application, there are 2 components. The dynamic components usually deal with the end user interaction with the user interface. The static component is usually the redundant part that almost always appears on the application independent of any user interaction. Hence, we can host the static components on static file servers to optimze its access time, thereby increasing the performance of the application. Datastore is an example of a service that falls under this.

- **Application Services** The application services are ones that are provided by Google AppEngine. They are usually used to perform basic low-level tasks. For example: UrlFetch Memcache, Image Manipulation

B. EyeOS

Fig. 6. Overview of EyeOS

EyeOS is n framework where it provides its users with online desktop. It is similar to a regular Desktop, except that it is online on the cloud. Also, its main focus is on enhancing the collaboration of different users. It also focuses on the application developers to only worry about their code and not about other compatibility. EyeOS takes care of all these issues for them.

ARCHITECTURE

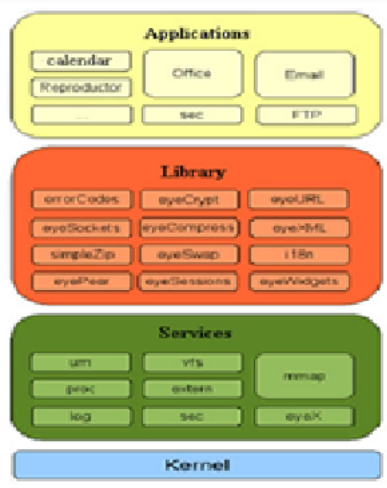

Fig. 7. Architecture of EyeOS

- **Kernel**
  The kernel for the EyeOS has a microkernel architecture. Hence, it manages the services and the communication between them. Hence, it is sufficient if the applications know only the service name. They do not have to invoke them. The kernel takes care of that.
- **Services**
  Services are components of the system which are completely necessary for the eyeOS. As explained previously, they are used to perform some internal functions of eyeOS. For example MMAP, extern, exec
- **Libraries**
  Libraries are similar to services. However, they are different from services in that they usually take care of the lower-level tasks of the system for example providing support and security, making sure everything is correct in the environment to run eyeOS.
- **Application**
  Applications in eyeOS are independent components that can be installed or uninstalled just like how we install and uninstall in the Windows and Linu operating systems. Security rules can also be defined to provide restrictions for users, groups etc . APIs are provided by the system to manage the applications that a user is running.

V. Advantages of Cloud Operating System

Since the cloud-based operating system is deploying application software on the server side, users do not need to install or deploy the relevant software onto the personal computer. Following are the advantages:

- **Easy Upgrades:**
  Cloud-based operating systems takes the struggle out of software updates. As the instant a new version of your systems OS hits the cloud, you'll be given the option to install it there.

- **Reduced exposure to Virus:**
  Since cloud OS doesn't interact directly with the end user's system hardware, there is absolutely no risk of having the hard drive or motherboard being controlled by some intrusive virus.

- **Speed Efficient:**
  For being able to run multiple programs at once or have a system with less RAM, a cloud-based OS can be of great advantage unlike the traditional OS that could lag as a result of multiple open programs, computer viruses or insufficient memory.

- **Cost Effective:**
  Software can be managed and updated by the service provider, therefore, the cost of using the software can be reduced while the user experience will be effectively improved.

VI. Limitations in the Existing Cloud Operating System:

The existing cloud-based operating systems can be divided into two categories considering the advantage of the cloud OS: Browser-driven operating system and Virtualization-based operating system.

Following are the limitations of the two categories:

- **Browser- driven operating system:**
  It is limited by the functionalities of browsers and weak in the variety of applications because the existing browser-based software is much less than desktop-based software and the API functionality of the browser is not comprehensive enough[21].
- **Virtualization- based operating system:**
  It which uses remote virtual machines (VMs) and Virtual Network Console (VNC) which usually transmit every frame of the software graphic interface, becomes difficult to service multiple users at the same time in private cloud leads to high network overhead, which will also cause performance degradation[21].

VII. Improvements to Cloud-based Operating Systems

The overhead of cloud-based operating system derives from the following aspects:

1) The overhead of running different applications on the backend cloud servers.
2) The overhead of transferring the information data from back-end to users.

To overcome the above shortcomings following improvements can be made to cloud OS:

1) Using Docker and X11 protocol
2) Edge Computing

A. *Using Docker and X11 protocol*

- **The Architecture of Virtual Machine**
  Virtual machines which is widely used in cloud operating system to do the application deployment on cloud server. In the virtual machine architecture applications deployed

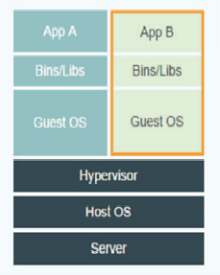

Fig. 8. Architecture of Virtual Machine in cloud OS

on Guest OS are based on Host OS and Hypervisor. Although the architecture provides different Guest OS, it must manage and maintain the whole two-level OS environment. This is not efficient as it will lead to performance attenuation.

- **The Architecture of Docker**

  Drawbacks which can't adapt to user's requirements of concurrency, ease of management as well as high responsiveness is overcome by using Docker with X11 in cloud OS.

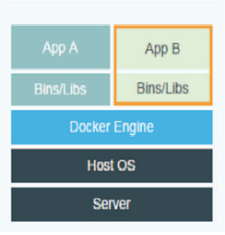

Fig. 9. Architecture of Docker in cloud OS

  – To do the application deployment and physical resource management a new technology Docker is proposed. Docker is an open source application level

container engine, which provides additional layer of abstraction and automation of operating system-level virtualization on Windows and Linux[21]. As Docker does not need to deploy the entire virtual machine with the Guest OS, it is much more lightweight than traditional virtual machine.

– With Docker, able to run each component in a separate container with its own dependencies and its own libraries all on the Host OS but within separate environments or containers. Only need to build the docker configuration once. Containerizing applications will completely isolate the environments and the applications can then have their own processes, services, network interfaces. As all containers run on the same OS kernel, all software's will depend only on that which can reduce the resource consumption.

– In traditional cloud operating system to do the remote access and control, Virtual Network Console (VNC) is used in cloud OS that can transmit the full graphical windows interface to another computer through network[21]. But VNC need to transmit every frame of software graphic interface will cost a large network bandwidth. To reduce the network overhead and improve the quality of remote-control access X11 protocol is used. X11 protocol translates the graphic command instead of each pixel of graphic interface causing less network overhead.

- **Evaluation of Traditional cloud OS and Cloud OS using Docker and X11 protocol**

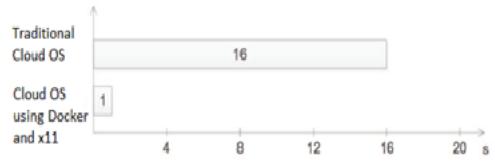

Fig. 10. Comparison of speed for opening software

Since cloud OS using Docker and X11 does not need to start guest OS so it opens software faster than traditional cloud OS i.e the cloud OS using virtual machine.Hence Docker and X11 technique will reduce the load of network over 50% while increasing CPU utilization by 45%, which can effectively support the private cloud scenario in enterprises[21].

B. EDGE COMPUTING

With the development of new technologies more and more smart devices are connected to the Internet. For a huge amount of data or the devices access to the traffic are all directly routed to the cloud data center,causing it to be burdened and bringing along with it many problems such as network congestion[22]. Edge computing is a distributed computing paradigm that supports the establishment of large-scale, distributed architecture allowing a certain percentage of data to be within a certain scope for it to be processed on the edge. Only a small number of necessary data and access traffic will be routed to the cloud data center[23].

Capabilities of Edge Computing

1) **Low network latency**
   Reduction in spatial distance brings about shortened propagation latency. It significantly reduces the latency caused by various routing/forwarding network devices which are processing different scenarios on complex networks, reducing the overall latency.[22]
2) **To support large bandwidth scenarios**
   By processing large-traffic on the edge, edge computing can effectively avoid network congestion related problems, and greatly reduce costs.
3) **Dealing with highly concurrent access**
   It uses a distributed architecture to mitigate the load on the cloud data center.

CONCLUSION

Cloud OS is developing and improving constantly due to which there is constant question for the need of new cloud OS. API's play a very important role in the evolution of cloud OS. Improvements to cloud OS is made on the basis of:

- Performance
- Scalability
- Ease of Programming
- Deployment

REFERENCES

[1] Chen ZN, Chen K, Jiang JL et al. Evolution of cloud operating system: From technology to ecosystem. JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY 32(2): 224–241 Mar. 2017. DOI 10.1007/s11390-017-1717-z.

[2] Mastering Cloud Computing: Foundations and Applications Programming Book by Christian Vecchiola, Rajkumar Buyya, and S. Thamarai Selvi

[3] Zhou F F, Ma R H, Li J et al. Optimizations for high performance network virtualization. Journal of Computer Science and Technology, 2016, 31(1): 107-116.

[4] https://docplayer.net/7926360-Iaas-cloud-architectures-virtualized-data-centers-to-federated-cloud-infrastructures.html

[5] Chang F, Dean J, Ghemawat S et al. Bigtable: A distributed storage system for structured data. ACM Transactions on Computer Systems (TOCS), 2008, 26(2): 205-218.

[6] Yu Y, Isard M, Fetterly D et al. DryadLINQ: A system for general-purpose distributed data-parallel computing using a high-level language. In Proc. the 8th USENIX Symposium on Operating Systems Design and Implementation, Dec. 2008, pp.1-14.

[7] Russell R M. The CRAY-1 computer system. Communications of the ACM, 1978, 21(1): 63-72

[8] Barik R, Zhao J, Sarkar V. Efficient selection of vector instructions using dynamic programming. In Proc. IEEE/ACM International Symposium on Microarchitecture, Dec. 2010, pp.201-212.

[9] Klimovitski A. Using SSE and SSE2: Misconceptions and reality. Intel Developer Update Magazine, Mar. 2001. http://saluc.engr.uconn.edu/refs/process/intel/sse sse2.pdf, Feb.2017.

[10] Intel I. Intelr SSE4 Programming Reference, D91561103, 2007. http://software.intel.com/sites/default/les/m/86/8/D9156103.pdf, Feb. 2017.

[11] Schwarzkopf M, Konwinski A, Abd-El-Malek M et al. Omega: Flexible, scalable schedulers for large compute clusters. In Proc. ACM European Conference on Computer Systems, Apr. 2013, pp.351-364.
[12] Verma A, Pedrosa L, Korupolu M et al. Large-scale cluster management at Google with Borg. In Proc. the 10th European Conference on Computer Systems, Apr. 2015, pp.18:118:17.
[13] Dean J, Ghemawat S. MapReduce: simplified data processing on large clusters. In Proc. the 6th Symposium on Operating Systems Design & Implementation, Dec. 2004, pp.137-150.
[14] Power R, Li J. Piccolo: Building fast, distributed programs with partitioned tables. In Proc. the 9th USENIX Symposium on Operating Systems Design and Implementation, October 2010, pp.293-306.
[15] R.Jemina Priyadarsini, L.Arockiam: Performance Evaluation of Min-Min and Max-Min Algorithms for Job Scheduling in Federated Cloud, International Journal of Computer Applications (0975 – 8887) Volume 99– No.18, August 2014 47.
[16] Understanding Cloud APIs, and Why They Matter: https://www.datacenterknowledge.com/archives/2012/10/16/understanding-cloud-integration-a-look-at-apis.
[17] IaaS Cloud Architectures: Virtualized Data Centers to Federated Cloud Infrastructures: https://docplayer.net/7926360-Iaas-cloud-architectures-virtualized-data-centers-to-federated-cloud-infrastructures.html.
[18] Yue M A, Huang G. Research on Application Virtualization Based on Docker.Computer Engineering & Software, 2015.
[19] Merkel D. Docker: Lightweight Linux Containers for Consistent Development and Deployment[J]. Linux Journal, 2014,2014(239): 2.
[20] eyeOS 2.x: http://www.123seminarsonly.com/Seminar-Reports/2013-02/60427673-Eyeos-Developer-Manual-English.pdf
[21] Zhen Du, ZhuQing Xu, Fang Dong, Dian Shen: A Novel Solution of Cloud Operating System based on X11 and Docker, 978-1-5386-1072-5/17 2017 IEEE DOI 10.1109/CBD.2017.13
[22] Extending the Boundaries of the Cloud with Edge Computing: https://www.alibabacloud.com/blog/extending-the-boundaries-of-the-cloud-with-edge-computing_594214.
[23] Edge Computing: https://en.wikipedia.org/wiki/Edge_computing.