International Journal of Scientific Research in Computer Science, Engineering and Information Technology
© 2018 IJSRCSEIT | Volume 3 | Issue 1 | ISSN : 2456-3307

A Review on Fog Computing : Conceptual Live Vm Migration Framework, Issues, Applications and Its Challenges

Yenumala Sankara Rao*[1], Kannganti Bhavya Sree[2]
*[1]Associate Professor, Department of MCA, St. Mary's Group of Institutions, Guntur, Andhra Pradesh, India
[2]PG Students, Department of MCA, St. Mary's Group of Institutions, Guntur, Andhra Pradesh, India

ABSTRACT

Fog computing, an extension of cloud computing services to the edge of the network to decrease latency and network congestion, could be a comparatively recent analysis trend. though each cloud and fog provide similar resources and services, the latter is characterized by low latency with a wider spread and geographically distributed nodes to support quality and time period interaction. During this paper, we tend to describe the fog computing design and review it's completely different services and applications. we tend to then discuss security and privacy problems in fog computing, that specialize in service and resource accessibility. Virtualization could be an important technology in each fog and cloud computing that permits virtual machines (VMs) to be in an exceedingly physical server (host) to share resources. These VMs might be subject to malicious attacks or the physical server hosting it may expertise the system failure, each of that lead to the inconvenience of services and resources. Therefore, an abstract sensible pre-copy live migration approach is conferred for VM migration. exploitation this approach, we will estimate the time period once every iteration to see whether or not to proceed to the stop-and-copy stage throughout a system failure or associate attack on a fog computing node. this may minimize each the time period and also the migration time to ensure resource and repair accessibility to the tip users of fog computing. Last, future analysis directions are made public.

Keywords : Cloud computing, edge computing, fog computing, live VM migration framework

I. INTRODUCTION

The term *fog computing (*FC) (also known as edge computing) was coined in 2012 by Cisco. FC is mainly proposed for IoT applications with massive number of services and real-time requirements. Fog is in between the cloud (data centers) and the ground, where devices are located. (Fog does not replace the traditional cloud but cooperates with it.) Any device with computing, storage, and networking capabilities can be regarded as a fog node. Typical examples of fog nodes include routers, switches, servers, machines, and video surveillance cameras.

Fog computing basically extends cloud computing and services to the edge of the network, such as portable devices, smart objects, wireless sensors and other Internet of Things (IoT) devices, as illustrated in Figure 1 [1]. Its basic goal is to improve efficiency and reduce the amount of data transported to the cloud for processing and storage. Compute can now take place at the sensor level, and not mainly in a centralized data center. This adds flexibility as to where computation can be placed. It also leads to the ability to react more quickly to events.

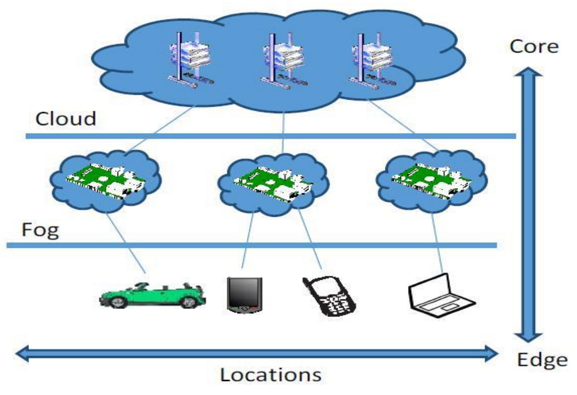

Figure 1. A view of the three-tier fog computing model [1].

II. RELATED WORK

FOG COMPUTING ARCHITECTURE

Fog computing is well suited for the geographical distribution of resources instead of having a centralized one, meaning Fog computing is the extension of Cloud computing. The difference is Fog provides proximity to its end users through dense geographical distribution and it also supports mobility. Access points or set-up boxes are used as end devices to host services at the network. In Fog computing platform multi-tier architecture is used. In first tier there is machine to machine communication and the higher tiers deals with visualization and reporting. See in Figure 1[2].

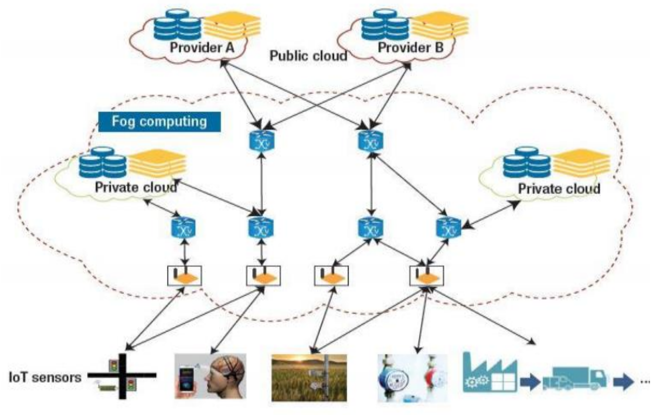

Figure 2. Fog computing Architecture

KEY FEATURES

Heterogeneity: Fog computing is a virtualized platform that offers computational, networking and storage services between cloud computing and end devices. Its heterogeneity feature serves as a building block as it exists in different forms and can be deployed in wide ranging environments.

Geographical distribution: Fog computing has a widely distributed deployment in order to deliver high-quality services to both mobile and stationary end devices.

Edge location, location awareness and low latency: The emergence of fog computing is partly due to the lack of support for endpoints with quality services at the edge of the network. Examples of applications with low latency requirements are video streaming in real-time closed circuit television monitoring and gaming.

Real-time interaction: Various fog applications, such as real-time traffic monitoring systems, demand real-time processing capabilities rather than batch processing.

Support for mobility: Mobility support is essential for many fog computing applications to enable direct communication with mobile devices using protocols such as Cisco"s Locator/ID Separation Protocol that decouples host identity from location identity using a distributed directory system.

Large-scale sensor networks: This is applicable when monitoring the environment or in smart grid using inherently distributed systems that require distributed computing and storage resources.

Prevalent to wireless access: Wireless access points and cellular mobile gateway are typical examples of a fog network node.

Interoperability: Fog components must be able to interoperate to ensure support for wide range of services like data streaming.

ISSUES AND CHALLENGES IN FOG COMPUTING

Realizing fog computing full potential presents several challenges including balancing load distribution between edge and cloud resources, API and service management and sharing. There are several other important examples.

- ✓ Enabling real-time analytics: In fog environment , resource management systems should be able to dynamically determine which analytics tasks are being pushed to which cloud or edge -based resource to minimize latency and maximize throughput. These systems also must consider other criteria such as various countries" data privacy laws involving, for example, medical and financial information.
- ✓ Programming models and architectures: Most stream- and data-processing frameworks,

including Apache Storm and S4, don't provide enough scalability and flexibility for fog and IoT environments because their architecture is based on static configurations. Fog environments require the ability to add and remove resources dynamically because processing nodes are generally mobile devices that frequently join and leave networks.

- ✓ Security, reliability, and fault tolerance: Enforcing security in fog environments which have multiple service providers and users, as well as distributed resources is a key challenge. Designing and implementing authentication and authorization techniques that can work with multiple fog nodes that have different computing capacities is difficult. Public-key infrastructures and trusted execution environments are potential solutions. Users of fog deployments also must plan for the failure of individual sensors, networks, service platforms, and applications. To help with this, they could apply standards, such as the Stream Control Transmission Protocol, that deal with packet and event reliability in wireless sensor networks.
- ✓ Privacy: The fog will allow applications to process user's data in third party's hardware/software. This of course introduces strong concerns about data privacy and its visibility to those third parties.
- ✓ Power consumption: Fog environments consist of many nodes. Thus, the computation is distributed and can be less energy efficient than in centralized cloud systems. Using efficient communications protocols such as Coop, effective filtering and sampling techniques, and joint computing and network resource optimization can minimize energy consumption in fog environments.

III. PROPOSED WORK

PROPOSED FOG COMPUTING APPLICATION TAXONOMY

Different fog computing applications have been suggested in the literature, therefore, in this section, we present taxonomy of such applications.

Luan *et al.* [16] described fog as a surrogate of cloud that can be used to deliver location-based service application to mobile device users (e.g., showcasing its application in shopping centers, parklands, inter-state bus, and vehicular fog computing networks). Boron *et al.* [17] demonstrated the role of fog computing in three scenarios, namely: con-nested vehicle, smart grid and wireless sensor and actual-tor networks. Souza et al. [18] used the Smart Transport System (STS) as a use case, where STSs are heterogeneous distributed systems designed to constantly monitor tram c activities and transmit data between commuters and smart devices in real-time to pre-empt tram c and safeguard com-mutters. Dastjerdi *et al.* [19] demonstrated the application of fog computing in healthcare, highly latency intolerant augmented reality domain and its use for improving web-site performance by caching and pre-processing. Saharan and Kumar [20] identified four areas of fog computing application, namely: wireless and actuator networks, smart grid, smart traffic lights and connected vehicles, and IoT. Kitanov *et al.* [120] proposed a hybrid environment service orchestration that provides resilient and trustworthy fog computing service beyond the 5G network.

In our taxonomy, we categorize fog computing applications into real-time and near real-time applications see Fig. 2. Fog computing can also be introduced in a network (for non-real-time application) to reduce the amount of traffic in the core; however, this is beyond the scope of this work.

Real-time applications are low-latency and function within a pre-defined timeframe which user senses as immediate or current. Near real-time applications, on the other hand, are those that are subject to time delay introduced by data processing or network transmission between the moment an event occurs and the use of the processed data [109]. Near real-time is often determined by subtracting the current time from the processing time that is nearly the time of the live event. In this section, we present popular use cases of both real-time and non-real-time applications.

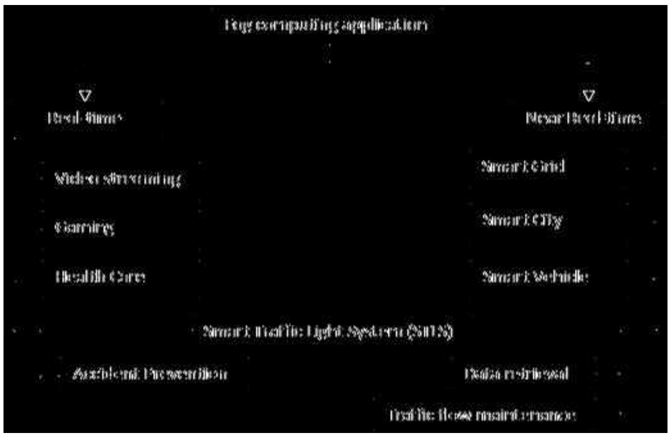

Figure 3. Proposed fog computing application taxonomy

IV. REAL-TIME USE CASES

A. VIDEO STREAMING

Transmissions of video applications and services are more efficient in a fog computing implementation, due to the capability of fog computing to provide location awareness, low latency, mobility, and real-time analytics. Several smart devices support smart surveillance that can be used by law enforcement of cars to display live video streams of events of interest. For example, Hong et al. [33] described a video surveillance application that requires a three-level hierarchy system to perform motion detection with smart camera, face recognition with fog computing instances, and identity aggregation with cloud computing instances. Magurawalage et al. [34] proposed Aqua computing, inspired from water cycle, which can take the form of either fog or cloud computing. The proposed architecture consists of clones placed at the edge of the network that serve end users in a video streaming scenario to act as a buffer. Zhu et al. [22] used fog computing to transform video applications and ser-vices to support on-demand video delivery. Such an approach enhances interactions in a virtual desktop infrastructure sys-tem and provides real-time video analytics for a surveillance camera. Other potential benefits of deploying fog computing to improve video streaming performance such as intelligent caching and adaptive streaming were also highlighted. Forester *et al.* [35] identify Ed key requirements of fog computing that complements cloud computing to support an intelligent network node. This helps to improve the quality of transmitted video by ensuring an intelligent soft handoff of mobile user and radio-aware resource management.

B. GAMING

The advent of cloud computing has provided a platform for computer gaming without users (players) worrying about hardware requirements. Cloud gaming providers in recent times have been rapidly expanding or leveraging cloud infrastructure to provide game-on-demand (GoD) service to users over the Internet. It is offered remotely by enabling an interactive gaming that can be accessed and decoded by end devices such as smart phones or tablets. Wang and Dey [40] described a cloud server based mobile gaming approach, cloud mobile gaming, where most of the workload for executing the game engine are placed on the cloud server. The mobile device only sends and receives user gaming commands to and from the servers. Zhou *et al.* [37] identified faster response time and higher QoS as key goals to be achieved in ensuring high gaming QoE. Due to the stringent requirements of gaming, cloud gaming is inherently susceptible to latency due to game graphics being rendered online. Lee *et al.* [38] investigated how the response latency in cloud gaming would affect user experience and how it varies between games. Then, a model was developed on how to predict the real-time strictness of a game based on players' input and game dynamics.

C. HEALTHCARE

IoT applications have provided a structured approach towards improving our health care services. This is achieved by deploying ubiquitous monitoring systems and transmitting the data to fog devices in real-time before sending the information to the cloud for further analysis and diagnosis. Gia *et al.* [46] utilized fog computing as a smart gate-way to provide sophisticated techniques and services such as distributed storage and embedded data mining. A case study of electrocardiogram feature extraction that plays a vital role in the diagnosis of cardiac diseases was presented. The experimental result suggested that deploying fog computing achieves a low latency and real-time response with more than 90% bandwidth efficiency. Persuasive health monitoring is one of the key application areas of biomedical big data research for making early predictions to support smart healthcare decision making. Cao *et al.* [47] proposed a real-time fall detection algorithm, U-Fall, which consists of three major modules, front-end, back-end and communication module. Both front-end and back-end make independent detection results. However, a collaborative detection will increase the accuracy and reduce the false alarm rate. An experiment demonstrating the use of the U-Fall algorithm in fog computing that automatically detects pervasive fall during health monitoring to mitigate stroke was presented. Results obtained suggested that a high sensitivity and spec city was achieved. Similar to the work in [47], FAST, a distributed analytics system based on fog computing to monitor and mitigate stroke, was proposed in [48].

D. SMART TRAFFIC LIGHT SYSTEM (STLS)

Smart traffic lights interact locally with a number of sensor nodes to detect the presence of cyclists, bikers or pedestrians, as well as estimating the speed and distance of approaching vehicles [17]. This information can be used to prevent accidents by sending early warning signals to approaching vehicles. Stojmenovic and Wen [36] described the use of video camera that senses the presence of an ambulance aching light during an emergency to automatically change street lights and allow the emergency vehicle to pass through trafic. Bonomi *et al.* [12] identi ed three major goals of STLS, namely: accident prevention, steady traf c ow maintenance, and retrieval of relevant data to evaluate and improve the system. Accident prevention is a real-time process, while traf c ow and data retrieval are regarded as near real-time and batch processes. Wireless access points and smart traf c light units are deployed along the roadside to provide commu-nication such as vehicle-to-vehicle, vehicle to access point, access point to access point (see Fig. 3).

NEAR REAL-TIME USE CASES

E. SMART GRIDS

The current call for smart grids can be linked to the fact that the present-day energy demands have outpaced the rate at which energy is generated by conventional methods as well as the need to reduce gas emission to control or cur-tail climate change [42]. Abdelwahab *et al.* [43] proposed a cloud-assisted remote sensing approach to measure and collect smart grid operational information to enable seam-less integration and automation of smart grid components. Cloud computing feature that uses a centralized demand response scheme, where customers and suppliers communi-cate directly with the cloud has proven to be bandwidth inefficient. Therefore, Stojmenovic [44] proposed a distributed approach by presenting a macro-grid and micro-grid to act as fog devices. Customers communicate with the nearby fog devices rather than the remote cloud. Fog devices, on the other hand, communicate frequently with the customers and occasionally with the cloud. Vatanparvar and Al Faruque [45] presented a Cyber-Physical Energy System (CPES) to improve the efficiency, reliability and performance of power grid by managing demand and supply dynamics intelligently. A prototype of this was implemented in fog computing platform to support interoperability, scalability and remote monitoring.

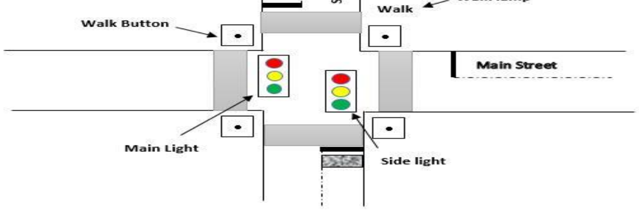

FIGURE 4. Smart Traffic Light System (adapted from [95]).

F. SMART CITIES

A smart city is one key IoT application that ranges from smart traffic management to energy management of buildings, etc. The smart city concept has drawn great interest from both science and engineering sectors and from both research and practitioner communities, as a means to overcome challenges associated with rapid urban growth. Kitchin [56] described smart city as a city that is vastly controlled and made up of ubiquitous computing whose economy and governance are driven by innovation and creativity. However, some of these IOT applications and devices in a smart city require high computation and storage capacities, and pose interoperability challenges. For example, Byers and Wetterwald [55] identified the complexity associated with a cloud centralized architecture involving smart city that consists of road traffic control, parking lot management and environmental monitoring over a distributed territory.

G. SMART VEHICLES

The advent of mobile cloud computing has necessitated the study of its agents such as vehicles, robots and humans that interact together to sense the environment, process the data and transmit the results. Lu *et al.* [59] described connected vehicle that communicates with their internal and external environment such as Vehicle-to-Vehicle (V2V), Vehicle-to-Sensor on-board (V2S), Vehicle-to-Road infrastructure (V2R) and Vehicle-to-Internet (V2I). Vehicle cloud has been identified [57] as the leading application that facilitates safe driving, urban sensing, content distribution and intelligent transportation to render bene ts such as sensing urban congestion and collaborative reconstruction of footage in a crime scene.

A significant attribute of vehicular cloud as compared to the Internet cloud is its reliance on the sensors they carry, rather than cloud computing resources. Hou *et al.* [116] described a vehicular fog computing that utilizes vehicles as an infrastructure for computing and communication that involves the collaboration of many end-user clients or near-user edge devices. Lee *et al.* [119] described a vehicular fog as the equivalent of Internet cloud in vehicles and the core system environment that will enhance autonomous driving. VANET is a mobile ad-hoc network that uses vehicles as mobile nodes. Truong *et al.* [23] proposed a new architecture for VANET by combing SDN and fog computing to cater for future VANET demands and support surveillance services by considering resource manager and fog orchestration models. Kim *et al.* [58] presented a solution to insufficient parking space as a result of rapidly increasing number of vehicles by proposing a shared parking model in a vehicular network using both fog and cloud environments. Simulation results indicated a high efficiency and reliability in determining vacant parking slot.

V. VM LIVE MIGRATION TECHNIQUES

During VM live migration, a chunk of the memory state is migrated to a target host even as the source continues to execute. Pre-paging is a method of optimizing memory-constrained disk-based paging systems. It is commonly regarded as a proactive way of pre-fetching from disk, where the memory subsystem attempts to hide the latency of highly-locality page faults by logically sequencing the pre-fetched pages [80]. Due to increasing dynamic random-access memory (DRAM) capacities, recent virtual memory does not often employ pre-paging.

Two designs for live migration were presented in [81], namely: pure stop-and-copy and pure on-demand. The former halts the migrating VM and copies the entire memory to the target host to minimize the total migration time. However, this results in an increase in downtime. The latter, on the other hand, functions by restricting the VM to copy only essential data in the kernel to the target host. The remaining VM address space is transferred when accessed at the target host. Both techniques, however, suffer from poor performance. The pure stop-and-copy technique causes signifycant service disruption while the pure on-demand technique incurs a longer migration time; hence, necessitating the development of a pre-copy approach to achieve a balance between downtime and migration time [7], [79]. In pre-copy live migration, all memory pages are copied in the first iteration while sub-sequent iterations transfer the modified pages that occurred during the previous iteration. The iterative process functions by periodically tracking dirty pages that occur in previous iterations, in order to keep migration time and downtime to a minimum.

Post-copy live migration has also been proposed in [82], which stops the VM at the initial stage in order to transfer the vCPU state and device to the target host. The VM is started immediately thereafter and subsequent memory pages are fetched from the source on demand. proposed an adaptive pre-paging to eliminate duplicate page transmission and dynamic self-ballooning to avoid the transfer of free memory pages.

Pre-copy algorithm is the predominant approach used for live migrating VMs, as evident in Xen, VMware and KVM hypervisors [73]. As discussed, the memory pages of the running VM are copied iteratively over several rounds until the modi ed pages are small enough to temporarily halt the VM at the source and resume on the target host. In the rst round, all pages are copied while in subsequent rounds, only modi ed (i.e., dirty) pages are moved. These modi ed pages can be tracked using a dirty bitmap maintained by the hypervisor.

Several methods have been proposed in the literature to reduce the amount of data transferred between physical hosts during iterative pre-copy stage, which in turn reduces the total migration time and downtime. Michael and Shen [84] proposed an ef cient technique to gradually migrate database connection from source to a target host using a self-adapting algorithm designed to minimize performance impact on the migrating tenant. Only frequently accessed cache contents are sent from the source to the target server. Piao *et al.* [85] proposed a snapshot memory compaction technique based on disk cache and memory. It uses an adaptive downtime control scheme based on the history of VM memory update informa-tion (i.e., writable working set) in KVM hypervisor. A live and incremental whole-system approach, three-phase migra-tion, was proposed in [86] to minimize downtime resulting from the migration of a large amount of disk storage data. An incremental migration algorithm is, thereafter used to transfer the VM back to its source in a very short migra-tion time. A compression technique, MECOM, was pro-posed in [87] that use memory compression based VM migration approach to ensure fast and stable VM migration. Roan *et al.* [79] proposed an improved pre- later copy algorithm to reduce the migration time and band-width resource consumption while keeping the downtime constant.

Caroni and Colligate [88] described the live migration of a virtual network function of an emerging paradigm, cloud-based edge network, and proposed a model that can collectively migrate a group of correlated VMs in a single entity. Clark *et al.* [83] presented six stages of pre-copy migration process between two hosts:

Pre-migration a target host with guaranteed resources is pre-selected for future migration by the source host running the VM.

Reservation Resources on the target host are reserved in anticipation for the incoming migrating Imitative pre-copy during the first iteration, the entire RAM is sent from the source to target host, and sub-sequent mode Ed dirtied pages are sent in preceding iterations.

Stop-and-copy in this stage, the VM is halted in order to copy its CPU state as well as any remaining inconsistent pages to the target. At the end of this stage, the source and target host have consistent copies of the VM. Commitment The target host indicates that it has successfully received a consistent

VM copy and the source acknowledges the message before discarding the original VM. The target host now becomes the primary host. Activation The migrated VM is now activated and post-migration codes run to re-attach device drivers on the new machine.

In all six stages, the determinant factor of when to move to the stop-and-copy stage after iterative pre-copy to ensure a minimum migration time and downtime has been the subject of recent research (see [90], [99]). This has a huge impact on the performance of application hosted in the VM. In the case of Xen [81], for example, the stop conditions used for pre-copy algorithms are de ned as follows:

If less than 50 pages were dirtied during the last pre-copy iteration.

If 29 pre-copy iterations have been carried out.
If more than 3 times the entire allocated RAM to the VM have been copied from source to the target host during the iterative pre-copy stage.

The first condition ensures a guaranteed minimum downtime as few pages are transferred, while the second and third conditions force the migration process into the stop-and-copy stage irrespective of the amount of modi ed pages left at the source host. This has a signi cant impact on the downtime of the application running on the VM.

To further enhance the stop condition after the iterative stage, Zhang *et al.* [89] designed and implemented a VM migration selection method that uses a performance degra-dation that is sensitive to users. Source codes are analyzed to determine memory size, dirty rate and frequently dirty pages that affect transmission time and downtime. Jo *et al.* [9] used a memory-to-disk mapping in Xen hypervisor to maintain an up-to-date mapping of identical memory pages in the network attached storage. During the iterative pre-copy stage in VM live migration, the memory-to-disk mapping is sent directly to the target host and the contents are fetched directly from the network attached storage. This reduces the total migration time while keeping the downtime to a minimum. Ibrahim *et al.* [90] proposed an algorithm that determines when to switch to the stop-and-copy phase when matched memory pattern does not achieve any signi cant progress during the iterative phase under different scienti c applica-tion benchmark.

VI. VM LIVE MIGRATION EVALUATION

In order to quantify migration performance during VM live migration, we use downtime and total migration time. Down-time is the overall time a VM is suspended during migra-tion that affects the availability of VM during the migration period [79]. Total migration time, on the other hand, is the total time required to move the VM between a source and the target host. Live migration of VMs in virtualized envi-ronments, such as fog computing, is critical to the perfor-mance and reliability of the running application. Wu and Zhao [75] presented a model that can predict the migration time, given the application behavior of the migrating VM and the resources available for migration in Xen environment. Nathan *et al.* [91] analyzed existing prediction models in KVM and Xen migration, and their ndings indicated a very high error rate due to the non-consideration of writable work-ing set size, a number of pages eligible for skip and the rela-tionship between the number of skipped pages, pages dirty rate, and page transfer rate. To counter this, a comprehensive predictive model that estimates the performance of KVM and Xen live migration was proposed. A study to determine the effect of VM live migration on the performance of the running application inside Xen VM was carried out in [92]. Findings showed that migration overhead is generally acceptable, but it should not be neglected, especially in cases of stringent availability conditions.

VII. CONCLUSION

Fog Computing is not a replacement for Cloud Computing. Fog Computing is a big step to a distributed cloud by controlling data in all node points, fog computing allows turning data centre into a distributed c loud platform for users. Fog is an addition which develops the concept of cloud services. It is possible to isolate data in the cloud systems and keep them close to users. Fog computing is proposed to enable computing directly at the edge of the network, which can deliver new applications and services especially for the future of Internet. Fog computing extends the Cloud Computing paradigm to the edge of the network, thus enabling a new breed of applications and services. Defining characteristics of the Fog are: low latency and location awareness; wide-spread geographical distribution; mobility; very large number of nodes, predominant role of wireless access, strong presence of streaming and real time applications, heterogeneity. In this paper, the authors a rgue that the above characteristics make the Fog the appropriate platform for a number of critical Internet of Things (IoT) services and applications, namely, Connected Vehicle, Smart Grid, Smart Cities, and, in general, Wireless Sensors and Actuators Networks (WSANs)[3].

VIII. REFERENCES

[1]. O. Osanaiye, K.-K. R. Choo, and M. Dlodlo, “Distributed denial of service (DDoS) resilience in cloud: Review and conceptual cloud DDoS mitigation framework,” J. Netw. Comput. Appl., vol. 67, pp. 147 165, May 2016.

[2]. M. Díaz, C. Martín, and B. Rubio, “State-of-the-art, challenges, and open issues in the integration of Internet of Things and cloud computing,” J. Netw. Comput. Appl., vol. 67, 99 117, May 2016.

[3]. Botta, W. de Donato, V. Persico, and A. Pescapé, “Integration of cloud computing and Internet of Things: A survey,” Future Generat. Comput. Syst., vol. 56, pp. 684 700, Mar. 2016.

[4]. S. Yi, Z. Qin, and Q. Li, “Security and privacy issues of fog computing: A survey,” in Proc. 10th Int. Conf. Wireless Algorithms, Syst., Appl. (WASA), Qufu, China, 2015, pp. 685 695.

[5]. M. Aazam and E.-N. Huh, “Fog computing and smart gateway based communication for Cloud of Things,” in Proc. IEEE Int. Conf. Future Internet Things Cloud (FiCloud), Barcelona, Spain, Aug. 2014, pp. 464 470.

[6]. V. Medina and J. M. García, “A survey of migration mechanisms of virtual machines,” ACM Comput. Surv., vol. 46, no. 3, 2014, Art. no. 30.

[7]. Shribman and B. Hudzia, “Pre-copy and post-copy VM live migration for memory intensive applications,” in Euro-Par 2012: Parallel Process-ing Workshops (Lecture Notes in Computer Science). New York, NY, USA: Springer, 2012, pp. 539 547.

[8]. U. Deshpande, Y. You, D. Chan, N. Bila, and K. Gopalan, “Fast server deprovisioning through scatter-gather live migration of virtual machines,” in Proc. 7th IEEE Int. Conf. Cloud Comput. (CLOUD), Anchorage, AK, USA, Jun./Jul. 2014, pp. 376 383.

[9]. Jo, E. Gustafsson, J. Son, and B. Egger, “Ef cient live migration of virtual machines using shared storage,” in Proc. 9th ACM SIG-PLAN/SIGOPS Int. Conf. Virtual Execution Environ., Houston, TX, USA, 2013, pp. 41 50.

[10]. M. Mishra, A. Das, P. Kulkarni, and A. Sahoo, “Dynamic resource management using virtual machine migrations,” IEEE Commun. Mag., vol. 50, no. 9, pp. 34 40, Sep. 2012.

[11]. S. Yi, C. Li, and Q. Li, “A survey of fog computing: Concepts, applica-tions and issues,” in Proc. ACM Workshop Mobile Big Data, Hangzhou, China, 2015, pp. 37 42.

[12]. F. Bonomi, R. Milito, P. Natarajan, and J. Zhu, “Fog computing: A platform for Internet of Things and analytics,” in Big Data and Internet of Things: A Roadmap for Smart Environments

(Studies in Computational Intelligence). New York, NY, USA: Springer, 2014, pp. 169 186.

[13]. M. Aazam and E.-N. Huh, “Fog computing micro datacenter based dynamic resource estimation and pricing model for IoT,” in Proc. IEEE 29th Int. Conf. Adv. Inf. Netw. Appl. (AINA), Gwangju, South Korea, Mar. 2015, pp. 687 694.

[14]. J. Zhu, D. S. Chan, M. S. Prabhu, P. Natarajan, H. Hu, and F. Bonomi, “Improving Web sites performance using edge servers in fog computing architecture,” in Proc. IEEE 7th Int. Symp. Service Oriented Syst. Eng. (SOSE), Redwood City, CA, USA, 2013, pp. 320 323.

[15]. Fog Computing and Internet of Things: Extend the Cloud to Where the Things Are, accessed on Apr. 18, 2017.

[16]. T. H. Luan, L. Gao, Z. Li, Y. Xiang, and L. Sun. (2015). “Fog com-puting: Focusing on mobile users at the edge.” [Online]. Available: https://arxiv.org/abs/1502.01815

[17]. F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, “Fog computing and its role in the Internet of Things,” in Proc. ACM 1st Ed. MCC Workshop Mobile Cloud Comput., 2012, pp. 13 16.