PROCEEDINGS

The 4th International Workshop on

# Dew Computing

(DEWCOM 2019)

Editors: Ruppa Thulasiram, Ralph Deters,

9th-10th November 2019

PATRONAGE

IEEE STC DEWCom

# Table of Contents

# Preface

This proceedings contains the papers presented at DEWCOM 2019: The 4th International Workshop on Dew Computing held on November 9-10, 2019 online through the Zoom platform.

DEWCOM is an annual international workshop on dew computing. The first one, DEWCOM 2016, was held in Charlottetown, Canada. The second one, DEWCOM 2017, was held in Opatija, Croatia. The third one, DEWCOM 2018, was held in Toronto, Canada together with the 28th Annual International Conference on Computer Science and Software Engineering (CASCON 2018).

Dew computing is a post-cloud computing model appeared in 2015. While cloud computing uses centralized servers to provide various services, dew computing uses on-premises computers to provide decentralized, cloud-friendly, and collaborative micro services to end-users.

Dew computing is an on-premises computer software-hardware organization paradigm in the cloud computing environment, which does not contradict with cloud computing, does not replace cloud computing, but it is complementary to cloud computing. The key features of dew computing are that on-premises computers provide functionality independent of cloud services and they also collaborate with cloud services. Briefly speaking, dew computing is an organized way of using local computers in the age of cloud computing.

DEWCOM 2019 was organized by IEEE Computer Society Dew Computing Special Technical Community (DewCom STC). Currently, DewCom STC members are spread in 28 different countries: Canada, USA, Croatia, Austria, Germany, Romania, Sweden, Ukraine, United Arab Emirates, United Kingdom, Argentina, Colombia, El Salvador, China, India, Pakistan, Australia, Macedonia, Nepal, Switzerland, Portugal, Ghana, Turkey, Cyprus, Malaysia, Brazil, Sweden, and New Zealand.

Each submission to DEWCOM 2018 was peer-reviewed by three reviewers according to the standard IEEE protocol. The committee decided to accept 4 papers. Among these papers, two papers involved discussions with other related areas, such as Cyber-Physical Systems and Post-cloud Computing; one paper involved blockchain technology; one paper introduced dew computing API.

.

November 10, 2019

DEWCOM 2019 Co-Chairs:
Ruppa Thulasiram, University of Manitoba, Canada
Ralph Deters, University of Saskatchewan, Canada
Yingwei Wang, University of Prince Edward Island, Canada

[1]

[1] Cover Design: Karolj Skala

## Program Committee

| | |
|---|---|
| Yingwei Wang | University of Prince Edward Island, Canada (Chair) |
| Karolj Skala | Ruder Boskovic Institute, Croatia |
| Ruppa Thulasiram | University of Manitoba, Canada |
| Marjan Gushev | Ss. Cyril and Methodius University, Macedonia |
| Ralph Deters | University of Saskatchewan, Canada |
| Parimala Thulasiram | University of Manitoba, Canada |
| Shuhui Yang | Purdue University Northwest, USA |
| Sven Groppe | University of Lübeck, Germany |
| Dana Petcu | West University of Timisoara, Romania |
| Partha Pratim Ray | Sikkim University, India |

# Dew Computing Architecture for Cyber-Physical Systems and IoT

Marjan Gusev, *Senior Member*, *IEEE*

***Abstract*—The concept to be on the edge of the Internet network means that the analyzed devices and systems will work only as a part of a general common integrated system, such as in the case of cyber-physical systems and various devices that act as Internet of connected Things. Although post-cloud architectures are most commonly associated with edge computing, a focus in this paper is set on dew computing architecture that extends this concept with a specific architecture out of the edge. The dew computing implementation in cyber-physical systems allows autonomous devices and smart systems, that can collaborate and exchange information with the environment, still be independent of other external systems or perform in a connected more complex cyber-physical system of systems. This paper aims at presenting an architecture of applying dew computing for cyber-physical systems, elaborating the new features and functionalities and comparing it to other similar architectures.**

***Index Terms*—*Edge computing*; *post cloud architecture*; *cloudlet*; *fog computing*; *CPSoS*, *cyber-physical systems***

## I. Introduction

Integrations of computational and physical processes define cyber-physical systems (CPS) [1], [2], enabling an environment where the physical and virtual worlds merge [3] as human-operated automated systems [4], or with a mental world [5]. Research challenges of CPS [6] include the design and development of next-generation devices and systems, including airplanes and space vehicles, hybrid gas-electric vehicles, fully autonomous urban driving, and prostheses that allow brain signals to control physical objects [4].

A majority of today's processors are realized as units embedded in various CPS, by means of sensors and actuators (sensing and acting on the environment) and increasingly connected with one another over the Internet. This distributed and connected approach means a lot more than just an application of information and communication technology (ICT). A multiple interdisciplinary knowledge is integrated for relevant physical perception and cognition, controlling the CPS.

The proliferation of various sensing devices in a communicating-actuating network creates the Internet of Things (IoT) [7]. It refers to Internet-based protocols through information sensing equipment to conduct information exchange and communications in order to achieve smart recognitions, positioning, tracing, monitoring, and management [8]. This concept is related to the CPS concept since IoT presents the connection to the physical world, while sophisticated algorithms and intelligence runs on a more complex cyber world.

A CPS may become a part of a more complex system of systems, defining a cyber-physical system of systems (CPSoS) [9]. Web of things [10] is another approach to connect various CPS in a more sophisticated networked system, which is particularly a realization of the same idea. Industry 4.0 is closely related with the IoT and CPS [11] as a concept that brings the fourth industrial revolution based on heterogeneous data and knowledge integration, agile and dynamic production systems, improvement of the overall system effectiveness and efficiency.

The emergence of Cloud computing introduced offloading data and computations to cloud servers for further processing and storage. *Post-Cloud architectures* [12] include *Edge computing* [13], bringing the computing closer to the user and distributing it across various hierarchical architectural layers, solving several technological challenges, such as energy efficiency, reducing bulk data transfers, etc.

In the beginning, the term edge computing was associated with content delivery networks to distribute software via edge servers [14]. At a later stage, edge computing was implemented mostly as *fog computing*, when the network operators define the architecture as a highly virtualized platform that provides compute, storage, and networking services between end devices and servers in the cloud [15]. The idea was initiated to provide a solution for various IoT devices and specify more powerful processing node at the edge of the Internet, and acts as a decentralized cloud architecture.

The idea of introducing cloudlets was initiated by virtual machines (VM) based mobile computing [16] for processing closer to the user when the data source is in the cyber world. A nice overview of cloudlet challenges and features [17] specifies the corresponding architectural approaches to realize computing closer to the data source using data both from the physical and the cyber world.

However, these approaches to locate the processing facilities in the proximity of data source do not solve the CPSoS resilience problem initiated when a smaller CPS will stop performing in case of a drop of connectivity to a more complex CPSoS, which may lead to severe consequences. Imagine an autonomous car when driving on a road where the white marking lines are erased and not visible. Should an autonomous car stop, or continue to drive analyzing where is the road by own algorithms. In addition to this, the problem of consistent and highly productive design of new CPSoS-s is another obstacle,

*Dew computing* is an additional layer between end-user devices, such as smart modules that convert physical parameters into digital information and vice-versa, and processing and coordination in the Edge/Fog/Cloud layers, offering autonomy, independence and collaboration features.

A scalable distributed architecture wase analyzed from the aspect of the computing and data source locations [18]. Wang [19] defines and categorizes the dew computing more precisely extracting the independence and collaboration features.

Developing new methods sets a lot of challenges to build high-confidence CPS. This paper addresses the applied architecture to satisfy the user and computing requirements.

The rest of the paper is organized as follows. Related work on corresponding CPS and IoT features, along with introduced and developed architectures are analyzed as related work in Section II. Section III specifies the user and computing requirements of trending CSPoS requirements analyzing them from the dew computing aspect, and presents a new architecture that matches these requirements. Section IV compares the existing architectures with introduced architecture that applies the dew computing concept into CPSoS and IoT solutions. Finally, Section V gives relevant conclusions.

## II. Related work

An overview of CPS is presented in several review papers [20], [2], [21], [6], [22], [23]. Here, the focus is set on analysis of CPSoS features and building architectures in order to point out the differences and additions for the dew computing concept implementations.

### A. Analysis of features

Maier [24] defines the following five key characteristics of SoS:

- Operational independence of the components of the overall system
- Managerial independence of the components of the overall system
- Geographical distribution
- Emerging behaviour
- Evolutionary development processes

Engell [9] defines cyber-physical systems of systems (CPSoS) as CPS which exhibit the features of systems of systems:

- Large, often spatially distributed physical systems with complex dynamics,
- Distributed control, supervision and management,
- Partial autonomy of the subsystems,
- Dynamic reconfiguration of the overall system on different time-scales,
- Continuous evolution of the overall system during its operation,
- Possibility of emerging behaviours,

The following main CPS [1], [25] characteristics and main design challenges are:

- *heterogeneous* in the sense that they combine various models of computations relying on both discrete and continuous time abstractions;
- *platform-aware* and resource-constrained, and thus the software depends on various non-functional properties imposed by the platform;
- time-sensitive and often *safety-critical*
- widely distributed with *heterogeneous interconnections*

Here, the definition expands the partial autonomy to system and subsystem autonomy, where various parts can work independently of other system parts and systems, and are capable to collaborate by exchanging data.

Gunes et al. give a nice overview of research challenges for CPSoS [6], which include related computing and user-related requirements. Wan et al. [26] covers the advances in CPS research pointing out the essential challenges. Challenges and trends for future research on Industry 4.0 and their impact on CPS and IoT are analyzed by a systematic and extensive review covering the critical issues of the interoperability [11]. The research challenges of big data techniques as a huge potential to be applied in the CPS was analyzed by Xu and Duan [27], especially analyzing the Industry 4.0 aspects or improving the system scalability, security, and efficiency. Here, the overview covers those essential requirements and challenges related to the implementation of the dew computing concept.

Platforms that spcecify processing closer to the data source can include data analytics on the edge [28] or deviceless approach [29].

### B. Architectural modelling

A four-layer architecture for an embedded system that realizes a CPS is defined by Lee [1] analyzing the complexity of implemented software in a typical embedded system. The lower level consists of silicon chips for processors, memory and control. The next level refers to executable programming primitives, and the third level to programs that implement higher-level languages. Task-level models are found on the fourth level, where performance and actor-oriented models are implemented.

Tan et al. [30] define a prototype architecture for CPS by defining the functional parts of such a system. They specify that a typical architecture of a traditional embedded system consists of a physical system that integrates actuators, a cyber system with sensors that sense the signals and convert them into information, and a higher-level control unit, which connects the cyber and physical parts providing necessary control. Their prototype enhances this architecture to a CPS architecture by adding a part which exchanges data with other networked systems and CPS units.

A holistic view on the functional parts of a CPSoS is presented by Gunes et al. [6]. The access to the physical world is realized with sensing networks and control with actuator network. A communication network is used to connect these units to the decision making CPS.

Lee at al. [25] define a five-layer architecture for building CPS architecture for Industry 4.0, analyzing the functionality of different computing levels. The lowest level is the smart connection level contains the signal sensing devices that collect data in various formats, speed and volume. Data-to-information conversion is the next level, which aims at realizing smart analytics and multi-dimensional data correlation to provide a self-aware feature. The third level is the cyber level,

where Information is being collected by connected machines and extracting additional information with specific analytics to provide self-comparison among similar machine performance and historical information. Cognition is the fourth level to process and exploit collected information of the monitored system by collaborative diagnostics and expert systems to take correct decisions on the priority of tasks and transfer acquired knowledge to users. The last (fifth) level refers to configuration, that acts as feedback from cyber to physical space, and as supervisory control to make machines become resilient, self-configurable and self-adaptive.

A structure of a networked system is analyzed by Kim and Kumar [21], where the essential functional units are those that represent the physical system (plant) and the cyber system (controller) connected by a network control unit. Their analysis covers the aspects from the control theory and automation, and application to real-time computing and networking.

Liu et al. [23] analyze the architecture based on the three layers, user, cyber and physical layers. The information system is at the heart of the cyber layer, in order to collect data, process information, make decisions and activate control mechanisms. The physical layer is represented by a sensor to analyze signals and collect data; and executor to activate controlling and change of the physical system.

Dastjerdi and Buyya [31] define a fog-computing architecture with IoT sensors and actuators as a bottom layer; corresponding applications to enhance their functionality at a specific edge and cloud resources layer; multitenant resource management at a network layer using fog computing with the edge devices and cloud services; quality-of-service enforcement at the resource-management layer, and finally, at the top layer, applications to deliver intelligent services to users.

Post-cloud computing paradigms bring the processing closer to the source where data is produced to avoid latencies [12], and include future architectures for IoT and CPS applications [32] A cloud-edge computing framework for CPS social services [33] specifies that edge devices provide services to cyber, physical and social worlds (CPSoS) if they integrate their services, share and exchange information via cloud computing. In this context, the architecture analyzed from the location where the computing is performed classifies the application, cloud and edge planes.

Lopez et al. [34] discuss that edge-centric computing encompasses the proximity, intelligence, trust, control and humans at the edge, as essential features different from cloud computing.

The fog computing architecture has been supported by the mobile operator and network providers to provide IoT. A typical architecture [15] defines data centers at the cloud layer, network operations at the core layer, multiple services at the edge layer and embedded systems and sensors at the bottom IoT layer.

Yanunuzzi et al. [35] analyze a mobile cloud computing scenario, as they address the mobility an essential feature of the IoT. Their approach is based on an architecture stack with service and applications at the top layer, followed by a cloud domain, network domain and the IoT domain.

A generic dew computing architecture [36] locates four tiers as cloud, fog, edge and dew computing, mainly as a location where the computing is performed correspondingly to data centers, distributed networking and computing servers, edge devices and dew servers and units at the lowest tier.

A novel fluid architecture for cyber-physical production systems [37] introduces three main layers to be cloud, fog and field, where the edge is defined to be at the bottom of the fog layer, and also in the dew computing layer and the IoT layer. In this context, dew computing is defined as a layer with very basic embeddable extensions to native computational capabilities of physical devices, and as an entry point to the distributed network.

A cloud-dew architecture [38] has been specified by Wang, mainly as a part of a cyber world, specifying details on a dew server. This corresponds to the definition of a dew computing layer, where the devices can be independent of the external systems, and also collaborate with them if there is an availability of the corresponding architecture. This concept has been extended to the physical world with IoT devices [39] by specifying a dew computing solution for IoT streaming devices [40].

## III. Architecture

The application of the dew computing concept in CPSoS and IoT targets the following distributed and networking computing elements:

- Smart modules, including IoT sensors, IoT actuators, embedded chips, etc.;
- Smart devices, such as smartphones, smartwatches, and tablets;
- Edge devices, such as home communication, entertainment and storage systems, or other networking equipment;
- Edge servers, including home personal computers and servers, cloudlet servers, servers located at mobile operator's base stations;
- Cloud servers, offering IaaS, PaaS and SaaS, where the integrated applications run and exchange data.

Fig. 1 presents the architectural layers, where the dew computing concept is integrated in the implementation of CPS and IoT.

The smart modules level makes the connection to the physical world. It consists of IoT devices, usually realized as small wearables and sensors, which are battery operated mobile devices with a wireless connection. They communicate with low energy radio communication technologies as personal area network (PAN) to the higher layer to preserve the energy consumption as much as possible,

The next level consists of smart devices, which represent the dew computing level. These devices are on the edge on the network, and it is not obligatory that they will have a permanent Internet connection. Their responsibility is to connect to the end-user IoT devices and collect the sensed data,

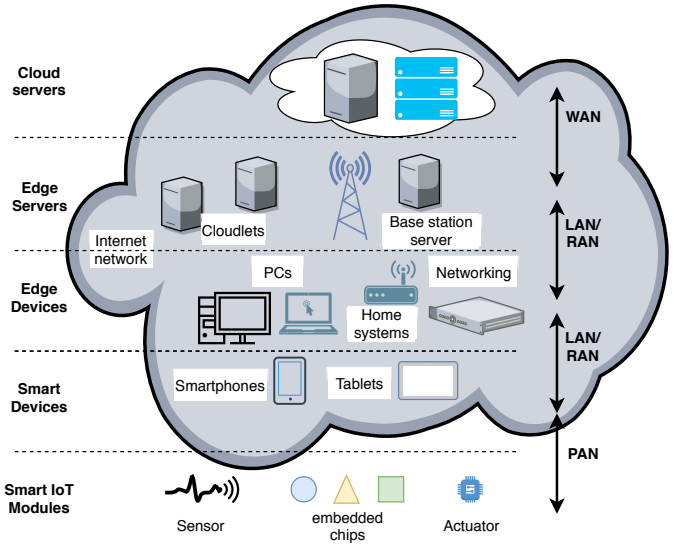


Fig. 1. Architecture of dew computing solutions for CPS and IoT

to initially process the corresponding information and control the actuators and other devices. Sometimes these devices might include also human user interfaces to enable human monitoring and controlling.

Edge devices with permanent Internet connection are found in the middle of the architecture as a bridge between the lower smart devices and IoT modules, or with the higher layer edge and cloud servers. They communicate with local area network (LAN) or radio area network (RAN) to the lower layer, including 3G, 4G and related technologies. Higher speed versions of LAN and RAN can be used to communicate to the higher layer.

Edge and cloud servers are the basis of the fourth and fifth layer correspondingly. The communication between them is realized via wide area network (WAN) high-speed technologies.

## IV. Discussion

Merging the virtual world with nearby IoT physical devices gives anyone with a mobile device and the appropriate authorization the power to monitor or control anything [41].

### A. Application domains

The new engineering solution defined within dew computing is to develop solutions that can work as a networked device, but at the same time, can overcome specific limitations, such as Internet provision, or limited Big Data capabilities of devices that have energy resource limitations (small battery) or other intrinsic limitations of today's computing system architectures and software design practices.

The main industrial sectors addressed by the CPS and IoT implementing the dew computing concept include:

- *Autonomous systems*, including cars, factory robots, and other similar machines, devices and systems that can interact with the environment and still be self-operated,
- *Biomedical and healthcare systems*, such as remote real-time monitoring and healthcare provision by wearable sensors, interacting with caregivers, patients and doctors; and various brain-machine interfaces, including therapeutic and entertainment robotics, orthotics, exoskeletons, and prosthetics that allow seamless integration of sensing, computing, and motor control functions for humans and animals;
- *Air transportation systems*, including drones and larger flying objects to achieve higher capacity, greater safety, more efficiency, operated autonomously or by a remote monitoring, management and control as a substitute of pilots,
- *Traffic control and intelligent transportation*, safety and defense systems, based on distributed surveillance systems, telepresence, remote monitoring agents, and controlling devices,
- *Assisted living systems*, integrating environmental surveillance systems, controlling the heating, ventilation, automation, cooling, lighting, watering, in an energy efficient manner,
- *Resource and critical infrastructure monitoring and control*, providing efficient use of electrical energy, water, and communication systems,
- *Environment management* including systems for energy conservation, remote monitoring, pollution and environment control and safety,
- *Personal assistive devices*, including navigation, environment status, online learning supported by autonomous learning tutors, etc.
- *Smart manufacturing*, including robotics and optimizing productivity in manufacturing the goods and service delivery, in the context of Industry 4.0,
- *Emergency response*, detecting and handling the threats and obstacles against public safety, sensing and dealing with human-related, technology-related and natural disasters by a network of sensors, and protecting the environment and available infrastructures.

These CPS and IoT devices are highly coupled systems which may integrate complex modularity principles by:

- computing multifunctional elements often by implementing the interdisciplinary big data concept and parallel processing,
- controlling the physical environment by sensing systems, controllers via feedback loops at different time and length scales,
- processing noisy signals to extract necessary features out of the noise,
- enabling fault-tolerance and resilience methods to resist various environment obstacles and provide high reliability and quality of service.

The main technological improvements are enlisted in Table I.

TABLE I
TECHNOLOGY BENEFITS OF IMPLEMENTING THE DEW COMPUTING CONCEPT FOR CPS AND IOT SOLUTIONS

| Benefit | Description |
|---|---|
| Energy efficiency | Distributing the processing and data storage closer to the source *consumes less energy* |
| Autonomy | Each particular IoT device and end-user CPS can work *autonomously based on its own resources* |
| Independence | Each particular IoT device and end-user CPS can work *independently of the surroundings* and other systems |
| Collaboration | Each particular IoT device and end-user CPS can work *collaboratively with other systems*, if there is a need to exchange information |
| Interoperability | Each particular IoT device and end-user CPS has the *ability to work with other systems* and exchange information among devices on the same layer and to other layers |
| Elasticity | The solution can *scale based on the workload*, including other CPS and IoT devices or excluding them |
| Resilience | The solution has the *ability to recover* from or adjust easily to unexpected change, such as power outage, Internet unavailability, various nearby EMC bursts, etc. |

### B. Comparison of the architectural approaches

A software-oriented architecture [1] aims at isolating the designers and software developers at different stages of analyzing a single CPS, without analyzing a CPSoS as a complex system that contains a multiple numbers of CSP and IoT devices.

The communication approach [15] (fog computing) to the computing architecture extracts the need of the industry to provide a virtualized distributed network, and analyzes the network management as a primary feature to provide applications to the end-user device.

A hybrid cross architectural layer [42] is defined as dew-fog-cloud stack for future data-driven CPS. Dew computing is considered to bring intelligence to the edge of the network and provide decisions keeping a centralized control by cloud computing. This approach combines both IoT and dew-edge devices into one layer, opposite to our approach to classify them according to the complexity of tasks they perform and computing features they provide.

Mahmud et al. [43] define computational domains to compare different computing approaches in post-cloud architectures, based on a location where the computation is performed. They categorize the following cloud, fog, edge, mobile cloud, and mobile edge computing, without deeper addressing of the features of designed applications. In this context, they do not differ between the edge and dew computing, and the difference between edge and mobile edge computing is in the use of cloud servers for mobile edge computing, opposite to other researchers that define mobile edge computing as a special case of the edge computing, when a mobile operator network is used [44].

At the extreme end, some researchers [43] define cloudlets as micro-cloud special fog computing node located at the middle of fog computing hierarchy. This is opposite to the cloudlet approach [13] where cloudlets are a different post-cloud concept provided by Internet providers, without fog computing, and to the understanding of edge computing [45] as a unified framework of cloudlets and fog computing, based on the network provider to be a mobile operator or Internet provider, an idea mostly arising by [44]. A nice comparison of fog, mobile edge computing and cloudlets are reported in [46], [47], [32].

Fog computing is sometimes defined as a superset of all other architectures, involving core networking in between the edge and cloud computing [43], as they analyze edge and cloud computing only as locations where the computing is performed, without details on the complexity of performed tasks, like the approach used in this paper.

A transparent computing-based IoT architecture [48] defines an architecture with five layers: end-user, edge server, core network, cloud and management and interface layer, categorized by data processing and service provisioning flow, where the end-user layer is composed of various IoT devices, besides the human user-operated smart systems, including smartwatches, smartphones etc. Our approach clearly differs between these smart devices and IoT sensors and actuators as separate layers due to the processing complexity and features they execute.

Hierarchical architecture of fog computing [49] defines only three basic layers as cloud, fog and terminal layers, mostly based on fog computing operated by mobile operators between the terminal devices that include IoT and cloud servers. This approach does not include architectures, where mobile operators are not part of the game, that is when the Internet providers establish the communication link between the IoT devices, nearby smaller servers (cloudlets) and cloud data centers.

Cristescu et al. [50] define three basic architectural layers to be the application layer with cloud computing provided services, network layer with edge networking, and fog computing gateways, and perception layer using edge devices, mist and dew computing layers. Their definition of dew computing relates to a layer where nodes and lines emerge and disappear according to environmental changes and takes over the mist computing layer tasks at the ground level, implying lowest processing and storage resources. This approach defines mist and dew just as a layer found on the ground base close to the end-user devices, which can be categorized more generally as edge computing, and does not analyze the main dew computing features of autonomy, independence and collaboration of IoT and edge devices.

As a summary, the architectural models were analyzed in the literature according to several criteria:

- functional parts and units [30], [6],
- software complexity layers [1],
- computing layers [25],
- user perspective [23],
- application-oriented perspective [31],
- communication approach [15],

Our approach is based on classifying the computing layers according to the complexity of performed functions and location where the computing, storage and communication capabilities meet the energy and resource availability.

## V. Conclusion

This paper presents a computing and communication architecture of applying the dew computing concept in the realization of CPS and IoT devices. The lower layer consists of devices that are independent, battery-operated small wearables or units that can exchange information with nearby smart devices. The smart devices represent the dew computing layer, as they can be independent of external systems and the Internet, but still collaborate with them if there is an available connection. The essence of this concept is autonomous performance as small CPS using the information provided form the related sensing IoT modules and controlling them through actuators. Sharing and exchanging information with the edge devices and higher layers allow them to work in a connected world and be a part of CPSoS.

The meteorological analogy to the cloud is the use of fog, mist and dew, as the computing comes closer to the source. We argue to those researchers that consider dew computing layer only as a computing layer in the proximity of IoT devices, that the corresponding devices can be considered as edge devices if they do not provide autonomy, independence and collaboration features.

This architecture was compared to other related papers, and basic features that extract the dew computing functionalities are elaborated. The impact of such a design is high. adding the autonomy, independence and collaboration features makes them different from conventional edge computing systems, or similar mobile edge, fog computing or cloudlet designs.

## References


[1] E. A. Lee, "Cyber physical systems: Design challenges," in *2008 11th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC)*. IEEE, 2008, pp. 363–369.

[2] J. Shi, J. Wan, H. Yan, and H. Suo, "A survey of cyber-physical systems," in *2011 international conference on wireless communications and signal processing (WCSP)*. IEEE, 2011, pp. 1–6.

[3] E. A. Lee and S. A. Seshia, *Introduction to embedded systems: A cyber-physical systems approach*. Mit Press, 2016.

[4] R. Baheti and H. Gill, "Cyber-physical systems," *The impact of control technology*, vol. 12, no. 1, pp. 161–166, 2011.

[5] H. Zhuge, "Future interconnection environment," *Computer*, vol. 38, no. 4, pp. 27–33, 2005.

[6] V. Gunes, S. Peter, T. Givargis, and F. Vahid, "A survey on concepts, applications, and challenges in cyber-physical systems." *KSII Transactions on Internet & Information Systems*, vol. 8, no. 12, 2014.

[7] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of things (iot): A vision, architectural elements, and future directions," *Future generation computer systems*, vol. 29, no. 7, pp. 1645–1660, 2013.

[8] K. K. Patel, S. M. Patel *et al.*, "Internet of things-iot: definition, characteristics, architecture, enabling technologies, application & future challenges," *International journal of engineering science and computing*, vol. 6, no. 5, 2016.

[9] S. Engell, "Cyber-physical systems of systems–definition and core research and development areas," *CPSoS*, 2014.

[10] T. S. Dillon, H. Zhuge, C. Wu, J. Singh, and E. Chang, "Web-of-things framework for cyber–physical systems," *Concurrency and Computation: Practice and Experience*, vol. 23, no. 9, pp. 905–923, 2011.

[11] Y. Lu, "Industry 4.0: A survey on technologies, applications and open research issues," *Journal of Industrial Information Integration*, vol. 6, pp. 1–10, 2017.

[12] Y. Zhou, D. Zhang, and N. Xiong, "Post-cloud computing paradigms: a survey and comparison," *Tsinghua Science and Technology*, vol. 22, no. 6, pp. 714–732, 2017.

[13] M. Satyanarayanan, P. Simoens, Y. Xiao, P. Pillai, Z. Chen, K. Ha, W. Hu, and B. Amos, "Edge analytics in the internet of things," *IEEE Pervasive Computing*, vol. 14, no. 2, pp. 24–31, 2015.

[14] M. Rabinovich, Z. Xiao, and A. Aggarwal, "Computing on the edge: A platform for replicating internet applications," in *Web content caching and distribution*. Springer, 2004, pp. 57–77.

[15] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*. ACM, 2012, pp. 13–16.

[16] M. Satyanarayanan, V. Bahl, R. Caceres, and N. Davies, "The case for vm-based cloudlets in mobile computing," *IEEE pervasive Computing*, 2009.

[17] A. Bahtovski and M. Gusev, "Cloudlet challenges," *Procedia Engineering*, vol. 69, pp. 704–711, 2014.

[18] K. Skala, D. Davidovic, E. Afgan, I. Sovic, and Z. Sojat, "Scalable distributed computing hierarchy: Cloud, fog and dew computing," *Open Journal of Cloud Computing (OJCC)*, vol. 2, no. 1, pp. 16–24, 2015.

[19] Y. Wang, "Definition and categorization of dew computing," *Open Journal of Cloud Computing (OJCC)*, vol. 3, no. 1, pp. 1–7, 2016.

[20] R. Rajkumar, I. Lee, L. Sha, and J. Stankovic, "Cyber-physical systems: the next computing revolution," in *Design Automation Conference*. IEEE, 2010, pp. 731–736.

[21] K.-D. Kim and P. R. Kumar, "Cyber–physical systems: A perspective at the centennial," *Proceedings of the IEEE*, vol. 100, no. Special Centennial Issue, pp. 1287–1308, 2012.

[22] R. Alur, *Principles of cyber-physical systems*. MIT Press, 2015.

[23] Y. Liu, Y. Peng, B. Wang, S. Yao, and Z. Liu, "Review on cyber-physical systems," *IEEE/CAA Journal of Automatica Sinica*, vol. 4, no. 1, pp. 27–40, 2017.

[24] M. W. Maier, "Architecting principles for systems-of-systems," *Systems Engineering: The Journal of the International Council on Systems Engineering*, vol. 1, no. 4, pp. 267–284, 1998.

[25] J. Lee, B. Bagheri, and H.-A. Kao, "A cyber-physical systems architecture for industry 4.0-based manufacturing systems," *Manufacturing letters*, vol. 3, pp. 18–23, 2015.

[26] J. Wan, H. Yan, H. Suo, and F. Li, "Advances in cyber-physical systems research." *KSII Transactions on Internet & Information Systems*, vol. 5, no. 11, 2011.

[27] L. D. Xu and L. Duan, "Big data for cyber physical systems in industry 4.0: A survey," *Enterprise Information Systems*, vol. 13, no. 2, pp. 148–169, 2019.

[28] S. Nastic, T. Rausch, O. Scekic, S. Dustdar, M. Gusev, B. Koteska, M. Kostoska, B. Jakimovski, S. Ristov, and R. Prodan, "A serverless real-time data analytics platform for edge computing," *IEEE Internet Computing*, vol. 21, no. 4, pp. 64–71, 2017.

[29] M. Gusev, B. Koteska, M. Kostoska, B. Jakimovski, S. Dustdar, O. Scekic, T. Rausch, S. Nastic, S. Ristov, and T. Fahringer, "A deviceless edge computing approach for streaming iot applications," *IEEE Internet Computing*, vol. 23, no. 1, pp. 37–45, 2019.

[30] Y. Tan, S. Goddard, and L. C. Pérez, "A prototype architecture for cyber-physical systems," *ACM Sigbed Review*, vol. 5, no. 1, p. 26, 2008.

[31] A. V. Dastjerdi and R. Buyya, "Fog computing: Helping the internet of things realize its potential," *Computer*, vol. 49, no. 8, pp. 112–116, 2016.

[32] J. Pan and J. McElhannon, "Future edge cloud and edge computing for internet of things applications," *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 439–449, 2017.

[33] X. Wang, L. T. Yang, X. Xie, J. Jin, and M. J. Deen, "A cloud-edge computing framework for cyber-physical-social services," *IEEE Communications Magazine*, vol. 55, no. 11, pp. 80–85, 2017.

[34] P. Garcia Lopez, A. Montresor, D. Epema, A. Datta, T. Higashino, A. Iamnitchi, M. Barcellos, P. Felber, and E. Riviere, "Edge-centric computing: Vision and challenges," *ACM SIGCOMM Computer Communication Review*, vol. 45, no. 5, pp. 37–42, 2015.

[35] M. Yannuzzi, R. Milito, R. Serral-Gracià, D. Montero, and M. Nemirovsky, "Key ingredients in an iot recipe: Fog computing, cloud computing, and more fog computing," in *2014 IEEE 19th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)*. IEEE, 2014, pp. 325–329.

[36] P. P. Ray, "An introduction to dew computing: Definition, concept and implications," *IEEE Access*, vol. 6, pp. 723–737, 2017.

[37] R. Beregi, G. Pedone, and I. Mezgár, "A novel fluid architecture for cyber-physical production systems," *International Journal of Computer Integrated Manufacturing*, vol. 32, no. 4-5, pp. 340–351, 2019.

[38] Y. Wang, "Cloud-dew architecture," *International Journal of Cloud Computing*, vol. 4, no. 3, pp. 199–210, 2015.
[39] M. Gusev and S. Dustdar, "Going back to the roots—the evolution of edge computing, an iot perspective," *IEEE Internet Computing*, vol. 22, no. 2, pp. 5–15, 2018.
[40] M. Gusev, "A dew computing solution for iot streaming devices," in *2017 40th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*. IEEE, 2017, pp. 387–392.
[41] R. Want, B. N. Schilit, and S. Jenson, "Enabling the internet of things," *Computer*, no. 1, pp. 28–35, 2015.
[42] M. Frincu, "Architecting a hybrid cross layer dew-fog-cloud stack for future data-driven cyber-physical systems," in *2017 40th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*. IEEE, 2017, pp. 399–403.
[43] R. Mahmud, R. Kotagiri, and R. Buyya, "Fog computing: A taxonomy, survey and future directions," in *Internet of everything*. Springer, 2018, pp. 103–130.
[44] I. Stojmenovic and S. Wen, "The fog computing paradigm: Scenarios and security issues," in *2014 Federated Conference on Computer Science and Information Systems*. IEEE, 2014, pp. 1–8.
[45] M. Satyanarayanan, "The emergence of edge computing," *Computer*, vol. 50, no. 1, pp. 30–39, 2017.
[46] G. I. Klas, "Fog computing and mobile edge cloud gain momentum open fog consortium, etsi mec and cloudlets," *Google Scholar*, 2015.
[47] Y. Ai, M. Peng, and K. Zhang, "Edge computing technologies for internet of things: a primer," *Digital Communications and Networks*, vol. 4, no. 2, pp. 77–86, 2018.
[48] J. Ren, H. Guo, C. Xu, and Y. Zhang, "Serving at the edge: A scalable iot architecture based on transparent computing," *IEEE Network*, vol. 31, no. 5, pp. 96–105, 2017.
[49] P. Hu, S. Dhelim, H. Ning, and T. Qiu, "Survey on fog computing: architecture, key technologies, applications and open issues," *Journal of network and computer applications*, vol. 98, pp. 27–42, 2017.
[50] G. Cristescu, R. Dobrescu, O. Chenaru, and G. Florea, "Dew: A new edge computing component for distributed dynamic networks," in *2019 22nd International Conference on Control Systems and Computer Science (CSCS)*. IEEE, 2019, pp. 547–551.

# Post-cloud Computing and Its Varieties

Parimala Thulasiraman*, Yingwei Wang†
*Department of Computer Science
University of Manitoba, Canada
thulasir@cs.umanitoba.ca
†School of Mathematical and Computational Sciences
University of Prince Edward Island, Canada
ywang@upei.ca

***Abstract*—Post-cloud computing is a computing paradigm appeared to cover a group of new computing paradigms, where these new computing paradigms are related to cloud computing and are remedial to cloud computing in the post-cloud computing era. Post-cloud computing has many varieties, including CDEF, i.e. Cloudlet, Dew computing, Edge computing, and Fog computing. Researchers and public need to grasp the essential meaning of each of these computing models and their differences. In this paper, we show a typical application for each variety to illustrate the differences.**

***Index Terms*—Post-cloud computing, Cloud computing, Cloudlet, Dew computing, Edge computing, Fog computing, Post-cloud computing applications**

## I. Introduction

Th widely acceptance of cloud computing made some people believe that cloud computing would be the new paradigm that replaces traditional on-site computing equipment and IT departments. While this kind of replacement has been going on, some new computing paradigms came into existence in the last few years. Among these new computing paradigms are cloudlet, dew computing, edge computing, fog computing, and so on. Such progress can be summarized in the following way: **From cloud to CDEF**, where C represents Cloudlet, D represents Dew computing, E represents Edge computing, and F represents Fog computing. CDEF starts with C also implies that these four models all started from Cloud Computing.

The cloudlet model promotes to put small-scale cloud data centers at locations where they are closer to users [1][2].

Dew computing emphasizes that on-premises computers provide functionality that is independent of cloud services and also collaborates with cloud services. Dew computing promotes that all on-premises computer applications get support from cloud services, if possible. With dew computing, cloud computing can reach its greatest popularity. Dew computing is complementary to cloud computing [3][4][5].

Edge computing pushes applications, data, and services away from central servers (core) to the edge of a network; it is based on the core-edge topology. Edge computing refers to the enabling technologies allowing computation to be performed at the edge of the network, on downstream data on behalf of cloud services and upstream data on behalf of IoT services [6][7][8][9].

Fog computing is a scenario where a huge number of heterogeneous devices communicate and potentially cooperate among them and with the network to perform storage and processing tasks without the intervention of third-parties. Fog computing extends cloud computing and services to devices such as routers, routing switches, multiplexers, and so on [10][11][12].

CDEF computing models originated from different background, proposed to solve different problems, related to different disciplines/industries, involved different devices, and have different methodologies. All these computing models share a common feature: they all perform computing tasks at devices that are closer to users [13]. The relationships among CEDF are similar to the relationships among different programming languages: although each programming language has full computing power of a Turing Machine, each language has its own style, strength, and characteristics. In the similar way, although the definitions of each of these CDEF computing models may be expanded to cover wider application areas, each of these models are more suitable to be used in some specific areas. From cloud to CDEF, the landscape of post-cloud computing is more versatile and prosperous.

CDEF is an unofficial, easy-to-remember way to refer to these computing models. To generalize the development of these computing models, the concept of *post-cloud computing models* was proposed [14][15][16][13][17][18].

In this paper, we would like to discuss the concept of post-cloud computing and applications of its varieties. The rest of the paper is organized in the following way: In Section II, we discuss the concept of post-cloud computing and provide its definition in a descriptive manner. Section III introduces a practical commercial application for each variety so that the differences among these varieties can be demonstrated. Section IV is the conclusions.

## II. Post-cloud Computing

CDEF summarizes the varieties of newly-proposed computing paradigms, but we still need a concept to naturally describe the common features of these paradigms. Post-cloud computing can serve this role.

Literally speaking, post-cloud computing is the computing paradigm appears after the cloud computing era. It may or may not have a specific technical definition, but it must be inclusive.

Since post-cloud computing may not have a technical definition, here we roughly describe post-cloud computing as the computing paradigms that appear after the cloud computing era, and that work together with cloud computing. Post-cloud computing is not a specific computing paradigm; instead, it covers a few computing models that are related to cloud computing and remedial to cloud computing in the post-cloud computing era. Major post-cloud computing varieties include CDEF, i.e. Cloudlet, Dew computing, Edge computing, and Fog computing.

Are we still in the cloud computing era or we are already in the post-cloud computing era? People may have different opinions on this issue. We believe that if in a stage of computing the following two conditions hold, this stage of computing could be called post-cloud computing era:

1) Cloud computing is not dominant.
2) The relative importance of cloud computing is not increasing.

Cloud computing obtained widely acceptance in the past decade; its usage was increasing quickly. But we believe that cloud computing has not dominated the computing world. With the quick development of Internet of Things, wireless devices, and artificial intelligence, new computing paradigms play more and more important roles and the relative importance of cloud computing is increasing slowly or not increasing at all. Thus, we believe that post-cloud computing era is coming or has already come.

## III. Post-cloud Computing Applications

The varieties of post-cloud computing include cloudlet, dew computing, edge computing, and fog computing. All these varieties provide some features that cloud computing cannot provide. They share one common feature: they all perform computing tasks at devices that are closer to users.

The origins, definitions, and principles of these varieties are introduced in [13]. In this section, we would like to introduce a real-world commercial application example for each variety so that the differences among these varieties can be clearly demonstrated.

### A. cloudlet

Akamai is a major content delivery network (CDN) provider. It operates a geographically distributed network of proxy servers and their data centers. Its goal is to provide high availability and high performance by distributing the service spatially relative to end-users. Akamai provides cloudlet services [19] to its customers. Currently, ten kinds of cloudlets are available. We just introduce one kind of cloudlets, Request Control Cloudlet, to illustrate the way it works.

Access control is an important part of managing visitor access to website or application. In today's connected world, having control over who can, or can't, access web properties is critical to protecting customer organization's content and information. It's important to ensure customer's resources remain available for intended audience and are not hindered by unwanted traffic that could be driving up costs. Often times these access control policy changes need to be made quickly and frequently creating a challenge to customer web operations.

Suppose a customer is operating a website through Akamai's CDN. This customer also uses Akamai's Request Control Cloudlet. The customer may use the control panel to create some rules. These rules could specify that requests from a specific CIDR, a specific continent, or a specific country will be given / or not given access. If this kind of request control is provided by the data center, the latency would be longer and load of the server and the network would be much higher. The cloudlet is running in a edge device in the CDN called Akamai Intelligent Platform; this device is geographically closer to the Web client, but is not in the cloud server where the website is running. The above feature is a key feature of cloudlet paradigm.

### B. Dew Computing

We use a well-known application, Dropbox [20], to illustrate dew computing. Dropbox works in the following ways: when the local host is online, the local copy of files are synchronized with the cloud copy of the files automatically; when the local host is offline, the local copy can still be used in whatever way the user wants to use them; when the local host get back online again later, the synchronization will be performed without any human intervention.

The above example is only one category of dew computing: Storage as Dew (SaD). Other categories of dew computing can be found in [5]. Although one example cannot reflect the the whole landscape of dew computing, it showed the two major features of dew computing: independent and collaboration.

### C. Edge Computing

Amazon Web Services (AWS) is a major cloud computing provider. Besides the well-known service Elastic Compute Cloud (EC2), Lambda@Edge [21] is a feature of another AWS Service: Amazon CloudFront. Lambda@Edge provides edge computing service.

Lambda@Edge lets you run code closer to users of your application, which improves performance and reduces latency. With Lambda@Edge, you don't have to provision or manage infrastructure in multiple locations around the world. You pay only for the compute time you consume - there is no charge when your code is not running.

With Lambda@Edge, you can enrich your web applications by making them globally distributed and improving their performance — all with zero server administration. Lambda@Edge runs your code in response to events generated by the Amazon CloudFront content delivery network (CDN). Just upload your code to AWS Lambda, which takes care of everything required to run and scale your code with high availability at an AWS location closest to your end user.

### D. Fog Computing

SONM [22] is a decentralized fog computing platform. It provides cloud services based on distributed customer level

hardware including PCs, mining equipment, and servers. You can either rent out your hardware or use someone's computing power for your needs.

This example illustrates that fog computing is different from cloud computing. In cloud computing, computing power exists in data centers; in fog computing, computing power exists everywhere. Fog computing provides some incentive to those who provide computing power. Such a financial model was proposed together with the fog computing concept.

## IV. Conclusion

In this paper, we discussed the concept of post-cloud computing, explored typical applications of the varieties of post-cloud computing: cloudlet, dew computing, edge computing, and fog computing. From these commercially available applications, we can see that that these varieties are quite different. They have only one belief in common: cloud computing should not be the only form of computing. The essential differences among them are not in their definitions that claim their coverage because definitions can be easily updated, expanded, and interpreted in different ways. The essential values of these computing models exist in their built-in principles, architectures, styles, and philosophy. Similar to programming languages, although each programming language has full computing power of a Turing Machine, each language has its own style, strength, and characteristics. People won't accept the idea that using one programming language to replace all other programming languages. These computing models will provide different frameworks, paradigms, guidelines, and architectures to researchers and developers in the post-cloud era.

## References


[1] S. Ibrahim, H. Jin, B. Cheng, H. Cao, S. Wu, and L. Qi, "CLOUDLET: towards mapreduce implementation on virtual machines," in *Proceedings of the 18th ACM International Symposium on High Performance Distributed Computing, HPDC 2009, Garching, Germany, June 11-13, 2009*, 2009, pp. 65–66. [Online]. Available: http://doi.acm.org/10.1145/1551609.1551624

[2] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The case for vm-based cloudlets in mobile computing," *IEEE Pervasive Computing*, vol. 8, no. 4, pp. 14–23, Oct 2009.

[3] Y. Wang, "Cloud-dew architecture," *International Journal of Cloud Computing*, vol. 4, no. 3, pp. 199–210, 2015.

[4] K. Skala, D. Davidovic, E. Afgan, I. Sovic, and Z. Sojat, "Scalable distributed computing hierarchy: Cloud, fog and dew computing," *Open Journal of Cloud Computing (OJCC)*, vol. 2, no. 1, pp. 16–24, 2015.

[5] Yingwei Wang, "Definition and categorization of dew computing," *Open Journal of Cloud Computing (OJCC)*, vol. 3, no. 1, pp. 1–7, 2016.

[6] Q. Pu, G. Ananthanarayanan, P. Bodik, S. Kandula, A. Akella, P. Bahl, and I. Stoica, "Low latency geo-distributed data analytics," *SIGCOMM Comput. Commun. Rev.*, vol. 45, no. 4, pp. 421–434, Aug. 2015. [Online]. Available: http://doi.acm.org/10.1145/2829988.2787505

[7] P. Garcia Lopez, A. Montresor, D. Epema, A. Datta, T. Higashino, A. Iamnitchi, M. Barcellos, P. Felber, and E. Riviere, "Edge-centric computing: Vision and challenges," *SIGCOMM Comput. Commun. Rev.*, vol. 45, no. 5, pp. 37–42, Sep. 2015. [Online]. Available: http://doi.acm.org/10.1145/2831347.2831354

[8] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, Oct 2016.

[9] M. Satyanarayanan, "The emergence of edge computing," *Computer*, vol. 50, no. 1, pp. 30–39, Jan 2017.

[10] Flavio Bonomi. (2011, Sept.) Connected vehicles, the internet of things, and fog computing. [Online]. Available: https://www.sigmobile.org/mobicom/2011/vanet2011/program.html

[11] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in *Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing*, ser. MCC '12. New York, NY, USA: ACM, 2012, pp. 13–16. [Online]. Available: http://doi.acm.org/10.1145/2342509.2342513

[12] L. M. Vaquero and L. Rodero-Merino, "Finding your way in the fog: Towards a comprehensive definition of fog computing," *SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 5, pp. 27–32, Oct. 2014. [Online]. Available: http://doi.acm.org/10.1145/2677046.2677052

[13] Y. Pan, P. Thulasiraman, and Y. Wang, "Overview of Cloudlet, Fog Computing, Edge Computing, and Dew Computing," in *Proceedings of The 3rd International Workshop on Dew Computing*, Toronto, Canada, 10 2018, pp. 20–23.

[14] D. Gardner, "Get ready for the post-cloud world," Datamation, 07 2017, https://www.datamation.com/cloud-computing/get-ready-for-the-post-cloud-world.html.

[15] Y.-Z. Zhou, D. Zhang, and N. Xiong, "Post-cloud computing paradigms: a survey and comparison," *Tsinghua Science and Technology*, vol. 22, no. 6, pp. 714–732, 12 2017.

[16] Y.-Z. Zhou and D. Zhang, "Near-end cloud computing: Opportunities and challenges in the post-cloud computing era," *Chinese Journal of Computers*, vol. 41, no. 25, pp. 1–24, 2018, online publishing, in Chinese, abstract in English.

[17] Y. Wang, "Post-cloud computing models: from cloud to cdef," Dew Computing Research, 11 2018, preprint.

[18] ——, "What is post-cloud computing?" ResearchGate, 11 2018, preprint.

[19] Akamai, "Akamai cloudlets," online, https://cloudlets.akamai.com/.

[20] Dropbox. (2014, February) Dropbox. [Online]. Available: https://www.dropbox.com/

[21] Amazon, "Lambda@edge," online, https://aws.amazon.com/lambda/edge/.

[22] SONM, "Decentralized fog computing platform," online, https://sonm.com/.

# Decentralized Hardware Ownership Control: Dew Computing with Blockchain

Yingwei Wang
School of Mathematical and Computational Sciences
University of Prince Edward Island
Charlottetown, Canada
Email: ywang@upei.ca

Marjan Gusev
Faculty of Information Sciences and Computer Engineering
Ss. Cyril and Methodius University
Skopje, Macedonia
Email: marjan.gushev@finki.ukim.mk

***Abstract*—In this paper, a decentralized hardware ownership control system and its implementation were proposed. Such a system could allow different manufactures/vendors and their customers to control the ownership of their products. Such system could discourage the happening of theft and ensure the ownership of customers. More importantly, this paper points out that dew computing can work together with blockchains in the similar way as it works with cloud services.**

***Index Terms*—Dew computing; Blockchain; Software in Dew; SiD; Cloud services, Cloud Servers.**

## I. Introduction

The great success and widely acceptance of cloud computing gave people an impression that cloud computing could dominate the computing world. In the last few years, other forms of computing models appeared and it is clear now that while cloud computing still plays a major role, other forms of computing models also play significant roles in the computing world. These forms of computing models include Cloudlet [1][2], Dew computing [3][4], Edge computing [5][6], and Fog computing [7][8]. For convenience, these forms of computing models are referred to as CDEF models [9][10].

In this paper, we concentrate on one category of dew computing: Software in Dew [11]. We discuss one of SiD's applications: hardware ownership control; we show SiD can work not only with cloud servers, but also with blockchains.

The rest of this paper is organized as follows: Section II introduces two concepts: the computing model SiD and the application hardware ownership control; Section III compares cloud services and blockchains to demonstrate that dew computing can work with both of them; Section IV describes the proposed blockchain which is able to provide hardware ownership control; Section V gives conclusions.

## II. Software in Dew and Hardware Ownership Control

According to [11], Software in Dew (SiD) is a dew computing category where a user's ownership to a piece of software is not only reflected by the software's existence on the user's on-premises computer, but also reflected by the ownership and settings information recorded in a cloud service. SiD should also make sure the user can re-download this software if necessary.

SiD is the opposite concept of Software as a Service (SaaS) in cloud computing. SiD promotes software local installation/operation, but it is not the same with the old-fashioned software local installation/operation; the difference is that SiD has cloud support and collaboration. Such cloud support and collaboration include download support, software ownership record, and so on.

In SiD, records in the cloud server guarantee the software ownership of a user. Even though the software's local copy is lost, or the user changed his/her computer, he/she still keeps the ownership of the software, and the software can be downloaded and installed in the local computer if necessary.

It is apparent that SiD is able to guarantee a user's software ownership, but it is not very straightforward that SiD is also able to control a user's hardware ownership.

One kind of software is special: the system software of hardware. Here hardware include computers, vehicles, cameras, and any other devices that are controlled by software. The ownership of the system software in hardware can also be maintained using the SiD model. To make sure the system software was used in accordance of its license, the IDs of the installed hardware might also be recorded in the cloud.

The owner of a piece of system software that has been installed in a piece of hardware could be considered as the owner of this piece of hardware; when another user wants to install/activate the system software on this piece of hardware, it will be rejected due to ownership conflict. This is the logic of hardware ownership control.

Hardware ownership can be controlled in two different levels. In the first level, hardware ownership is recorded in a proper place and it won't be changed without the owner's agreement; this ownership usually does not influence the usage of this hardware. We may call this level hardware ownership recording. In the second level, not only the ownership cannot be altered without the agreement of the owner, the hardware cannot be properly used when it does not have the owner's agreement; only owner can use the hardware; other people cannot not use this hardware because he/she is not able to control the system software of this hardware. The owner can revoke the usability of this hardware. We may call this mechanism hardware ownership control.

Such mechanism has been used in managing the ownership

of some categories of hardware. Some categories of hardware do not use system software to control its operation. For this category of hardware, only ownership recording is possible. For example, real estate properties and vehicles are areas where ownership recording was used. This category of hardware ownership control is limited: Thieves cannot obtain the ownership of a vehicle, but he can still drive the vehicle.

If a piece of hardware uses system software to control its operation, the second level ownership control can be used. For example, the ownership of some smartphones are controlled using this model. When ownership control is implemented, it discourages theft and makes the hardware more secure. This model can be introduced to new areas. For example, if whenever a vehicle is started, it needs permission from the cloud, the theft won't able to drive the car without the consent of the owner.

From the above discussions, we can see that SiD can be used to implement hardware ownership control. Dew computing and cloud computing are tightly inter-connected. SiD or any other category of dew computing never works by itself. It always involves cloud computing. From cloud point of view, such hardware ownership control model could be called Hardware Ownership Control as a Service or something similar, but no such concept exists at this time.

## III. Cloud vs. Blockchain

Hardware Ownership Control provided through SiD works well, but it has a problem: it relies on a specific cloud server, and the cloud server belongs to a specific company. Due to privacy, reliability, and other concerns, some users are not satisfied with such arrangements.

Because blockchain [12][13] is a decentralized paradigm, we explore the possibility that to implement hardware ownership control through a blockchain instead of a cloud server.

Let us do a comparison between a cloud server and a blockchain:

- From network topology view point, a cloud server is centralized whereas a blockchain is decentralized.
- From a user's view point, both a cloud server and a blockchain can provide similar services.

We know that dew computing relies on cloud computing; all dew computing applications involve some kind of cloud services. We want to ask a question: is it possible to let a blockchain to play the roles of a cloud service in dew computing applications?

Yes, this is quite possible. From a user's view point, a blockchain is a special kind of cloud service. All the considerations in dew computing in terms of a cloud service could be applied to blockchains. Of course, the architecture of blockchains should be altered to suit such dew computing requirements.

## IV. Hardware Ownership Control with Blockchain

Hardware ownership control can be implemented using SiD model with support from a blockchain. In this section, we discuss a few key considerations in this proposed blockchain. A Proof-of-Concept model blockchain to implement this application is under development.

### A. Crypto-currency

Blockchains are traditionally related to crypto-currencies [14]. The proposed blockchain may or may not have a crypto-currency component. Even a crypto-currency is included, such a crypto-currency is not the major goal of this blockchain; it may serve as incentive to support the operation of the hardware ownership control functions.

### B. Transactions

In the proposed blockchain, besides currency-related transactions, ownership-related transactions could be added in a new block and the new block will be appended to the blockchain if some conditions are met. We propose two different methods to set up ownership-related transactions.

In the first method, the following two types of transactions should be included:

- Ownership Initial Claim
- Ownership Transfer

In this method, the ownership of a piece of hardware will be initially claimed by an owner; from then on, its ownership will be transfered from one owner to another owner.

In the second method, the following two types of transactions should be included:

- Ownership Claim
- Ownership Release

In this method, the ownership of a piece of hardware can be claimed and released by different owners at different time.

Comparing these two methods, the first method ensures that the hardware is always owned by a user, whereas in the second method, its ownership has to be taken care of by human hands between two owners have their own ownership.

The first method is good at its consistency, and the second method is good at its flexibility. For now, we take the first method and discuss its implementation considerations.

### C. Ownership Initial Claims

Ownership initial claim needs carefully considerations. Suppose a piece of hardware was sold to a user; how could this user claim the ownership of this piece of hardware?

The simplest method is to let the user to claim ownership freely. The user may establish an ownership initial claim transaction with the serial number or ID of this piece of hardware. The problem of this method is that fake ownership could be established. Because no verification is needed, anyone can claim the ownership of a piece of hardware with specific serial number or ID, as long as it has not been claimed yet.

The second method is to introduce ownership verification by a manufacture or a vendor. In this method, the manufacture or the vendor produces a verification code for each piece of hardware and provides this code to the customer. The customer sends the serial number/ID and the verification code to the blockchain. The blockchain then forwards such information to

the manufacture/vendor get the ownership verified. If verified, the ownership initial claim transaction could be added to the blockchain; otherwise the claim will be rejected.

The above-mentioned verification mechanism is widely used. Its problem is that the manufacture/vendor has to keep a database of serial number/ID and its verification code; the manufacture/vendor has to involve in the verification process.

To release manufactures/vendors from the burden of verification, we propose a method that verifies ownership but does not need manufactures/vendors to do that; we call this method Function-pair Verification.

Inspired by asymmetric cryptography, we prepare a pair of functions for a manufacture: one is a private function $f$ and the other is a public function $g$. These two functions are related but $f$ cannot be derived from $g$.

The private function $f$ can be applied upon the serial number/ID $s$ of a piece of hardware to produce a verification code $v$, where $v = f(s)$.

$v$ is provided to a customer along with the hardware and its serial number/ID $s$.

The public function $g$ can be applied upon $s$ and $v$. It verifies if $v$ is the valid verification code for $s$. The public function $g$ must be complex enough so that the relationship between $s$ and $v$ cannot be revealed.

Suppose such pair of functions have been chosen by the manufacture/vendor, they keep function $f$ as a secret and use $f$ to generate a verification code for each serial number/ID of a piece of hardware; they provide these serial numbers/IDs and verification codes to customers; they submit function $g$ to the blockchain in the form of a callable function or a smart contract [15]. Whenever a user tries to claim ownership for a piece of hardware with a serial number/ID and corresponding verification code, function $g$ will be called; if it is verified this claim would be accepted by the blockchain; if it is not verified this claim would be rejected.

Using Function-pair Verification method, manufactures/vendors do not need to keep track of their serial numbers/IDs and corresponding verification codes; they do not need to maintain a service to verify ownership; ownership are verified inside the blockchain with a publicly-available function.

The detailed mathematical discussions related to the construction of function pair is beyond the scope of this paper.

### D. Software Download

The proposed blockchain, which is a decentralized hardware ownership control system, should be able to support all functions that a centralized hardware ownership control system is able to provide. Thus, the proposed blockchain should be able to support software download to verified hardware. Downloadable software should be published in the blockchain by the manufacture/vendor in the form of file or a smart contract.

## V. Conclusions

Computing models are changing and developing rapidly. In this paper, we demonstrate that dew computing can not only work with cloud services, but also with blockchains through an application: hardware ownership control. A blockchain was proposed to implement this application. Various issues in this proposed blockchain were discussed.

## References


[1] S. Ibrahim, H. Jin, B. Cheng, H. Cao, S. Wu, and L. Qi, "CLOUDLET: Towards Mapreduce Implementation on Virtual Machines," in *Proceedings of the 18th ACM International Symposium on High Performance Distributed Computing*, Jun. 2009, pp. 65–66.

[2] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The Case for VM-based Cloudlets in Mobile Computing," *IEEE Pervasive Computing*, vol. 8, no. 4, pp. 14–23, Oct. 2009.

[3] Yingwei Wang, "Cloud-dew Architecture," *International Journal of Cloud Computing*, vol. 4, no. 3, pp. 199–210, 2015.

[4] K. Skala, D. Davidovic, E. Afgan, I. Sovic, and Z. Sojat, "Scalable Distributed Computing Hierarchy: Cloud, Fog and Dew Domputing," *Open Journal of Cloud Computing (OJCC)*, vol. 2, no. 1, pp. 16–24, 2015.

[5] Q. Pu, G. Ananthanarayanan, P. Bodik, S. Kandula, A. Akella, P. Bahl, and I. Stoica, "Low Latency Geo-distributed Data Analytics," *SIGCOMM Comput. Commun. Rev.*, vol. 45, no. 4, pp. 421–434, 2015.

[6] M. Satyanarayanan, "The Emergence of Edge Computing," *Computer*, vol. 50, no. 1, pp. 30–39, 2017.

[7] Flavio Bonomi, "Connected Vehicles, the Internet of Things, and Fog Computing," The Eighth ACM International Workshop on Vehicular Inter-networking (VANET 2011), 2011, Keynote.

[8] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog Computing and its Role in the Internet of Things," in *Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing*, 2012, pp. 13–16.

[9] Yi Pan and Parimala Thulasiraman and Yingwei Wang, "Overview of Cloudlet, Fog Computing, Edge Computing, and Dew Computing," in *Proceedings of The 3rd International Workshop on Dew Computing*, Oct. 2018, pp. 20–23.

[10] Yingwei Wang, "Post-cloud Computing Models: from Cloud to CDEF," Dew Computing Research, 2018, Nov.4.

[11] ——, "Definition and Categorization of Dew Computing," *Open Journal of Cloud Computing (OJCC)*, vol. 3, no. 1, pp. 1–7, 2016.

[12] Sthuthie Murthy. (2018, May) "Blockchain will do for transactions what the internet did for information" - says IBM CEO. [Online]. Available: https://ambcrypto.com/blockchain-for-transactions-internet-for-information-ibm-ceo/

[13] Vitalik Buterin. (2013, Dec.) A Next-Generation Smart Contract and Decentralized Application Platform. [Online]. Available: https://github.com/ethereum/wiki/wiki/White-Paper#ethereum

[14] Satoshi Nakamoto. (2009, May) Bitcoin: A peer-to-peer electronic cash system. [Online]. Available: https://bitcoin.org/bitcoin.pdf

[15] K. Christidis and M. Devetsikiotis, "Blockchains and Smart Contracts for the Internet of Things," *IEEE Access*, vol. 4, pp. 2202–2303, 2016.

# Dew Text Application Development

Srija Srivastava, Sarada Kiranmayee Tadepalliy, Ruppa K. Thulasiramx, Parimala Thulasiraman
Department of Computer Science
University of Manitoba
Winnipeg, Manitoba, Canada
srijas@myumanitoba.ca, tadepask@myumanitoba.ca,
tulsi@cs.umanitoba.ca, thulasir@cs.umanitoba.ca

**Abstract**—Dew Computing being an extension of client-server architecture, makes data or website available to the user without an internet connection, by being independent and still collaborative with the cloud services. Since, dew computing is in its early stage,the technical implementation details are not readily available for public use, such as APIs. Considering this, my project aimed to create a set of library functions in php for one category of dew computing, i.e., Web in Dew (WiD). This project considers DewText application (proof of concept application) as the base of development of library functions. The resulting library functions are then used to re-write DewText application. Thus, demonstrating WiD applications could be written in an easier and simpler manner.

## 1 Introduction and Motivation

Cloud computing offers many beneficial features to the users and one of it includes data mobility. A user can access data or website from anywhere if internet connection is available, but when the internet connection is not available the user cannot access a website or his own data. These peculiar situations can be handled by dew computing [1], where it helps on-premises computer to be independent yet collaborative with the cloud services. Here, independence from cloud means that dew computing will make a software or data available to the user when there is no internet connection and collaboration refers to the process of automatic retrieval/transmission of information to/from the cloud when internet connection becomes available.

To address the problem stated earlier, a solution, cloud-dew architecture [2] in dew computing was introduced. This architecture is an extension of client-server architecture. Here, two types of servers are maintained namely, cloud server and dew server. A **dew server** is a web server that resides on user's machine,i.e, on-premises computer and facilitates similar services as cloud along with synchronization between dew server data and cloud server data. In other words, cloud-dew architecture allows a user to access website, related data and perform any operation on data, even if internet connection is not available. And as soon as the internet connection becomes available, the data gets synchronized automatically.

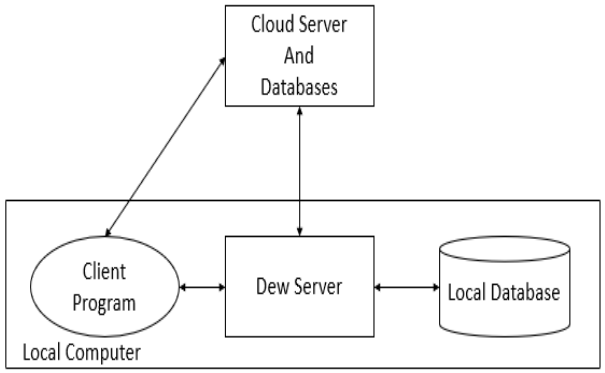


Fig. 1. Cloud-dew architecture

Depending on the area of application of dew computing it is categorized as: Web in Dew, Storage in Dew, Database in Dew, Software in Dew, Platform in Dew, Infrastructure as Dew, Data in Dew [3]. Web in Dew (WiD) category of dew computing allows access to web without internet connection by maintaining a copy of a fraction of World Wide Web on user's machine(on-premises computer). The architecture of WiD is implemented as cloud-dew architecture.

Due to the lack of availability of APIs, when it comes to the development of WiD applications, one has to start development from the scratch which involves complex coding. Considering it as a motivation, this project involves creation of a set of library functions in php, in the direction to re-write DewText application (proof-of-concept application) using those libraries.

The rest of the paper is organized as follows: Section 2 focuses on background and related works in the WiD category of dew computing, followed by Section 3, briefly describing the problem statement. An approach for creation of library functions is specified in Section 4. Experimental framework used for the project is explained in Section 5. Section 6, discusses the application of the library functions that have been created to re-write DewText application. The conclusion and future work are presented in Section

7.Section 8 showcases all the codes used in creation of the APIs.

## 2 Background and Related Work

Even though dew computing is in its early stage and is an emerging technology, the cloud-dew architecture was proposed as a solution for accessing web without internet connection. The cloud-dew architecture makes personal data stored in the cloud available for user all the time and facilitates web surfing without an internet connection. Cloud-dew architecture is an implementation of Web in Dew category of dew computing.

The WiD provides a better software delivery model by integrating SaaS (Software as a Service) and SaaP (Software as a Product) [4]. SaaS saves the data on central server and requires internet connection to access the website/software or data.Whereas,SaaP enables user to save data in the local machine and access the software without any dependency on internet connection. Since WiD integrates SaaS and SaaP, making the data, website or software to be available with or without internet connection.

As a proof of concept application, DewText was developed to demonstrate the functionality facilitated by cloud-dew architecture. It is a light weight, open-source application which can be expanded easily. It showcases the key feature of independence by making DewText website accessible without internet connection.With the help of dew server residing in the local machine, a user can make changes to a file, add a new file or delete an existing file in DewText application. All these operations performed on the data are stored in the local machine (on-premises computer). Following this, DewText application synchronizes the data stored on the local machine with the data stored in cloud as soon as internet connection becomes available, thus, illustrating collaborative feature of cloud-dew architecture.

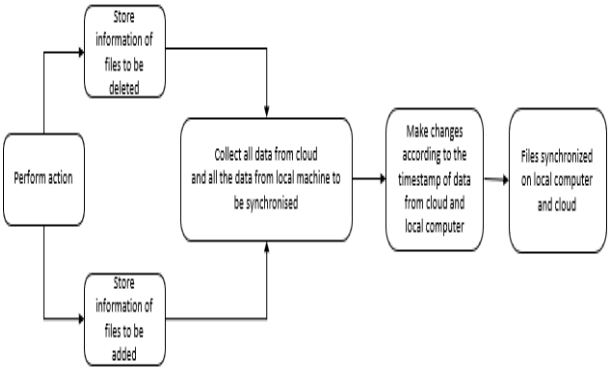


Fig. 2. DewText Synchronization Flow

When internet connection is available, user can access the DewText website deployed on the cloud as well as can access dewsite (website deployed on the dew server). Whereas when there is no internet connection, a user can access only the DewText dewsite and can perform any operations mentioned earlier. The changes made to any data will be stored on the local machine which upon internet availability, will get synchronized with the cloud data. As shown in Figure 2 and 3, Dewtext application collaborates

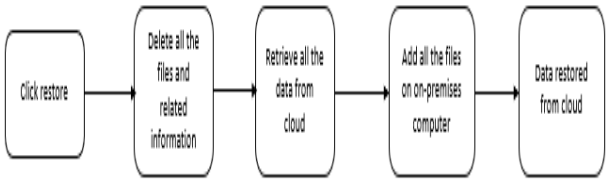


Fig. 3. DewText Restoration Flow

with the cloud by either synchronizing the data between cloud and local based on time-stamp or by restoring all the data stored in the cloud into the local machine.

## 3 Specific Problem Statement

Web in Dew (WiD) category of dew computing allows access to web without internet connection by maintaining a copy of a fraction of World Wide Web on local machine(on-premises computer). The architecture of WiD is implemented as cloud-dew architecture. Due to the lack of availability of APIs, when it comes to the development of WiD applications, one has to start development from the scratch which involves complex code. Considering it as a motivation, this project involves creation of a set of library functions in php, in the direction to re-write DewText application (proof-of-concept application) using those libraries.

## 4 Solution Strategy and Implementation

To make the development of Web in Dew applications easy and simple, this project concentrates on building Web in Dew API (library functions). The project considered DewText as the base application to propose key components of Web in Dew category of dew computing as library functions, which is built in php programming language. The key components of DewText application can be included as below:

**Change in file:** To maintain information regarding the type of changes made in the file. It may include changes in existing file, creation of a new file or deletion of an existing file.

**Prepare add and delete content:** To prepare a list of files that should be added or deleted, list of existing files in which some changes are made and the list of file content that are to be added with respect to the files to be newly added or changes made in the existing files.

**Delete on synchronization:** To delete all files that are queued for deletion upon synchronization between cloud and local data.

**Add on synchronization:** To add new files that are queued for addition upon synchronization and update existing files if any changes are made in it.

**Add files on restoration:** To add all the files on local machine(on-premises computer) downloaded from cloud in the process of restoring all the data from cloud.

**Delete files on restoration:** To delete all the files stored on local machine(on-premises computer) before retrieving the data from cloud which has been added to it.

**Clear file content on restoration:** To delete the content of files specified upon restoration. These files are used to store all the file's name that are currently stored on local machine(on-premises computer).

**Get lines in array:** To store each line of a file in an array (used multiple times in DewText), which would then be used for further processing such as addition of new file or changing the existing file.

**Get time:** To get the timestamp stored in the log file for a particular file that is to be added or deleted. The log file in DewText maintains the information of any operation performed on a file such as the timestamp of the activity, type of changes made to that file (new/delete) and name of the file.

**Get flag:** To get the flag or the type of changes associated with the file is maintained in the log file. Its value could be either ”new” or ”del”. The value ”new” is associated with a file when any change is made in it or if a new file is created. Whereas, the value ”del” is associated with the file which has to be deleted.

**Get file name:** To get the name of a file from the log file.

## 5 Experimental Framework

The proposed key components of DewText was created as library functions in php programming language using NetBeans IDE [5]. XAMPP [6] was installed as dew server on local machine(on-premises computer) and Composer package manager was used to manage the library created. DewText applications code can be obtained from [7].

## 6 Results

Library functions for the proposed key components of DewText were developed, which were then tested separately for its functionality independently. Later, these functions were used to re-write DewText application (dewsite) to test whether the application performed similar to the original application. It was found that the DewText application worked the same when it was re-written using those libraries.

### 6.1 Experimental testbed or environment

The proposed key components of DewText was created as library functions in php programming language using NetBeans IDE. XAMPP was installed as dew server on local machine(on-premises computer) and Composer package manager was used to manage the library created. The libraries were created and tested on the machine having processor specification as AMD A12 RADEON R7, 12 COMPUTE CORES 4C+8G,2.70GHz, 8 GB RAM and 64-bit operating system.

### 6.2 Detailed Results and their analysis

The re-written DewText application was tested to ensure the synchronization and restoration processes are working properly. To test synchronization process, first I disconnected the internet and then accessed the dewsite. In the dewsite I performed the following activities; added a file, deleted a file and made changes to an existing file. All the details of the changes along with its timestamp were stored in the log file. Later, I connected my machine to internet, and found that all the changes were synchronized with the cloud as soon as the internet connection was made available. Secondly, during the availability of internet connection, I accessed the dewsite and performed all the activities and noted that all the data got synchronized with the cloud simultaneously.

To test the restoration process, during the availability of internet connection, I accessed the DewText website and clicked the restore option. On selecting that option, all the data stored on local machine were deleted and then all the data stored in cloud were retrieved and saved on local machine. Later this data were available and accessible through dewsite when internet was disconnected.

## 7 Conclusions and Future Work

The proposed key components of DewText application were developed as library functions in php using composer package manager. These library functions reduced the code of DewText when it was re-written using them. Thus, demonstrating the development of applications similar to DewText can be easier and simpler.

Since, this project mainly concentrated on developing library functions for key components of DewText, the way it handles all the data is different. But considering these key components, it can be extended to handle any type of data with proper exception handling. This can also be extended to manage database content. Thus, managing all types of data in a more generic manner by increasing its utility.

## Acknowledgements

The last two authors acknowledge Natural Sciences and Engineering Research Council (NSERC) Canada for partial financial support for this research through Discovery Grants. The first author acknowledge the International Graduate Student Scholarship from Faculty of Graduate Studies, University of Manitoba.

## References

[1] Partha Pratim Ray. An introduction to dew computing: Definition, concept and implications. *IEEE Access*, 6:723–737, 2018.
[2] Yingwei Wang. Cloud-dew architecture. *IJCC*, 4:199–210, 2015.
[3] Yingwei Wang. Definition and categorization of dew computing. *Open Journal of Cloud Computing (OJCC)*, 3(1):1–7, 2016.
[4] Yingwei Wang and David Leblanc. Integrating saas and saap with dew computing. pages 590–594, 10 2016.
[5] Netbeans ide 8.0.2 download. https://netbeans.org/downloads/8.0.2/. (Accessed on 04/19/2019).
[6] Xampp apache + mariadb + php + perl. https://www.apachefriends.org/index.html. (Accessed on 04/19/2019).
[7] Dewtext: A proof-of-concept dew computing application. http://www.dewcomputing.org/index.php/2018/11/22/dewtext-a-proof-of-concept-dew-computing-application/. (Accessed on 04/19/2019).

## 8 codes

### 8.1 index.php

This file will help the user to create a new text file for the ”Notebook”.It also provides an interface to display,delete,synchronize and restore files.

```
<!DOCTYPE html>

<html>
```

```
<head>
    <meta charset="utf-8" />
    <title>Dew Text</title>
    <style  type="text/css">
    .title{
        text-align: center;
        font-size: 36px;
    }
    .button {
        background-color: #555555;
        border: none;
        color: white;
        padding: 15px 32px;
        text-align: center;
        text-decoration: none;
        display: inline-block;
        font-size: 22px;
        width:80%;
        margin: 4px 2px;
    }
    .del{
        background-color: #FF0000;
        width:15%;
    }
    .new {
        background-color: #527a7a;
    }
    .syn{
        background-color: #00BFFF;
        width:15%;
        padding: 17px 2px;
        font-size: 20px;
    }
    </style>
</head>
<body>
    <div>
        <h1 class="title">Notebook</h1>
    </div>
    <?php
        $index = fopen("log/index", "r");
        while(!feof($index))
        {
            $line = substr(fgets($index),0,-1);
            //remove \n at the end
            if($line!="")
            {
            print("<button class=\"button\"
            onclick=\"showfun('$line')\">
            $line</button>
            <button class=\"button del\"
            onclick=\"del('$line')\">Delete
            </button>
            <br>");
            }
        }
        fclose($index);
        print("<button class=\"button new\
        "onclick=\"showfun('new')\"> New
        </button>
        <button class=\"button syn\"
        onclick=\"synchronize()\">
        synchronize
        </button><br>");
        print("<button class=\"button syn\
        "onclick=\"restore()\"> Restore
        </button>");
    ?>
    <form id="form" method="get"
    action="edit.php">
        <input id="message"
        name="title" type="hidden"/>
    </form>
    <form id="form2" method="post"
    action="delete.php">
        <input id="message2"
        name="title" type="hidden"/>
    </form>
    <form id="form3" method="post"
    target="frame" action=
    "http://ec2-18-224-251-211.us-east
    -2.compute.amazonaws.com
    /dewText/sendLog.php">
    </form>
    <form id=
    "form4" method="post" target="frame"
    action="http://ec2-18-224-251-211.us-east-2.
    compute.amazonaws.com/dewText/sendAll.php">
    </form>

    <script>
        function showfun(title)
        {
            var message =
            document.getElementById("message");
            message.value = title;
            var form =
            document.getElementById("form");
            form.submit();
        }

        function del(t)
        {
            var message2 =
            document.getElementById("message2");
            message2.value = t;
            var form2 =
            document.getElementById("form2");
            form2.submit();
        }

        function synchronize()
        {
            var form3 =
            document.getElementById("form3");
            form3.submit();
            //location.reload();
            //style="display: none"
        }

        function restore()
        {
```

```
            var form4 =
            document.getElementById("form4");
            form4.submit();
        }

        var auto =
        setTimeout(function()
        { synchronize(); }, 10000);
    </script>
    <iframe name=
    "frame" style="display: none"></iframe>
</body>
</html>
```

### 8.2 delete.php

delete.php will help in deleting the contents after attaining the path of the file from the user that should be deleted and also it will delete its index value from the log file in index.php.

```
    <?php
require_once __DIR__ . '/vendor/autoload.php';
// Autoload files using Composer autoload
use lib\DewTextLib;

$title = $_POST["title"];

//get file path
$path = "text/".$title.".txt";
if(file_exists($path))
{
    //delete file
    unlink($path);
}

//remove from index
$contents = file_get_contents("log/index");
$contents = str_replace($title."\n",
'', $contents);
file_put_contents("log/index", $contents);

DewTextLib::changeInFile($title,"del");

header("Location: index.php");
?>
```

### 8.3 edit.php

Edit.php will help in editing the title and the content of the file.

```
    <!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <title>Page Title</title>
    <style type="text/css">
    textarea.title  {
        width: 100%;
        height:30px;
        margin-bottom: 1%;
        font-size:22px;
    }
    textarea.text {
        height:500px;
        width:100%;
        font-size:18px;
    }
    .button {
        background-color: #555555;
        border: none;
        color: white;
        padding: 15px 32px;
        text-align: center;
        text-decoration: none;
        display: inline-block;
        font-size: 22px;
        margin: 4px 2px;
    }
    </style>
</head>
<body>
<?php
    $title = $_GET["title"];
    $content = "";
    $truetitle = "Unnamed";
    if($title != "new")
    {
        $path = "text/".$title.".txt";
        $content =
        file_get_contents($path);

        //remove .txt
        $truetitle =
        rtrim($title, '.txt');
    }

    $content = json_decode($content);
    print("<textarea class=
    \"title\" id=\"title\" row=
    \"1\">$truetitle</textarea><br>
    <textarea class=
    \"text\" id=
    \"content\">".$content."</textarea><br>
    ");
    ?>
    <button class=
    "button" onclick="save()"> Save </button>
    <form id=
    "form" method="post" action="save.php">
        <input id=
        "p_title" name="title" type="hidden">
        <input id=
        "p_content" name="content" type="hidden">
    </form>
    <script>
        function save()
        {
        var title =
        document.getElementById("title").value;
        var content =
```

```
        document.getElementById("content").value;

        var form =
        document.getElementById("form");

        var p_title =
        document.getElementById("p_title");
        var p_content =
        document.getElementById("p_content");

        p_title.value = title;
        p_content.value = content;

        form.submit();
        }
    </script>
    </body>
    </html>
```

### 8.4 save.php

Save.php will update the log file with the new file's title and the content of it in the index.php

```
    <?php
require_once __DIR__ .
'/vendor/autoload.php';
// Autoload files using Composer autoload
use lib\DewTextLib;

$title = $_POST["title"];
$content = $_POST["content"];
//echo $content."<br>";
$content = json_encode($content);
//echo $content."<br>";
//save file
$path = "text/".$title.".txt";
file_put_contents($path,$content);

//check if $path exist in index
$file = fopen("log/index","r+");
$new = true;
while(!feof($file))
{
    $line = substr(fgets($file),0,-1);
    //remove \n at the end
    if($line == $title)
    {
        $new=false;
        break;
    }
}
if($new)
{
    fwrite($file,$title);
    fwrite($file,"\n");
}
fclose($file);

DewTextLib::changeInFile($title,"new");

header("Location: index.php");
```

### 8.5 restoreAll.php

restoreAll.php, will clear the existing log,index and delete the existing files on the dew server. Later,it replaces with the data from the cloud server.

```
    <?php
require_once __DIR__ .
'/vendor/autoload.php';
// Autoload files using Composer autoload
use lib\DewTextLib;

//clear log
DewTextLib::clearFilesOnRestore("log/log");
//clear index
DewTextLib::clearFilesOnRestore("log/index");
//delete rest files
DewTextLib::delFilesOnRestore("text");
//read
$fc=array();
$index = $_POST['index'];
echo $index."<br>";

$content = str_replace("\\n","\n",$index);
$i=substr_count( $content, "\n" );
$count =0;
for(;$count<$i;$count++)
{
    $fcx= $_POST["new$count"];
    $fc[]=$fcx;
}
DewTextLib::addOnRestore
($index,$fc,"text","log/index");
?>
```

### 8.6 sendAll.php

sendAll.php will fist initiate the restoration of all files. Then, it gets the contents from the log and send it to the dew server. The ”Title” and the ”Contents” of the file are updated.

```
    <form id="instruction" method="post"
    action = "http://localhost/dewText/
    restoreAll.php">
    <?php
    $fileindex =
    file_get_contents("log/index");//get log
    $enc = json_encode($fileindex);
    print("<input name=\"index\" value=$enc />");
    //send log to dew

    $content = $fileindex;
    $count = 0;
    while($content != "")
    {
        $lineEnd = strpos($content,"\n");
        $line = substr($content,0,$lineEnd);
        //get line

        $content = substr($content,$lineEnd+1);
```

```
        //rest of log

        $title = $line;//get title
        echo $title."<br>";
        $path = "text/".$title.".txt";
        //get path
        if(file_exists($path))
        {
            $file= file_get_contents($path);
            //get file content
            //echo $file."<br>";
            print("<input name='new$count'
            value = '$file'>");
            $count++;
        }
    }
?>
</form>
<script>
    var form =
    document.getElementById("instruction");
    form.submit();
</script>
```

### 8.7 synchCloud.php

synchCloud.php will help in synchronizing the title and contents of the file on the cloud, i.e any updation or deletion of files.Subsequently, removing these files from the log present in the index if the file is deleted or updating the contents of the existing file.

```
    <?php

$dnum = $_POST['delNum'];
$nd = (int)$dnum;
//del all files
for($i=0; $i< $nd; $i++)
{
    $k = "del".$i;
    $path = $_POST[$k];
    if(file_exists($path))
    {
        //delete file
        unlink($path);
    }

    //remove from index
    $s = strpos($path,"/");
    $d = strrpos($path,".");
    $l = $d - $s - 1;
    $title = substr($path,$s+1,$l);

    $contents =
    file_get_contents("log/index");
    $contents =
    str_replace($title."\n", '', $contents);
    file_put_contents("log/index", $contents);
}

$anum = $_POST['newNum'];
$na = (int)$anum;
//add all files
for($i=0; $i< $na; $i++)
{
    $k = "new".$i;
    $path = $_POST[$k];
    $k = "newC".$i;
    $seris = $_POST[$k];

    //save file
    $file = fopen($path,"w");
    fwrite($file,$seris);
    fclose($file);

    //add to index
    $s = strpos($path,"/");
    $d = strrpos($path,".");
    $l = $d - $s - 1;
    $title = substr($path,$s+1,$l);

    //check if it exist
    $file = fopen("log/index","r+");
    $new = true;
    while(!feof($file))
    {
    //remove \n at the end
        $line = substr(fgets($file),0,-1);
        if($line == $title)
        {
            $new=false;
            break;
        }
    }
    if($new)
    {
        fwrite($file,$title);
        fwrite($file,"\n");
    }
    fclose($file);
}

//clear log
file_put_contents("log/log","");
?>
```

### 8.8 sendLog.php

sendLog.php,first it synchronize all the files present in the dew server and cloud server and then it will update the log.It will also have the updated file contents after deletion or creation of the files.

```
<form id="instruction" method="post"
action="http://localhost/dewText/synchDew.php">
<?php
    //get log
    $log= file_get_contents("log/log");
    $enc = json_encode($log);
    //send log to dew
    print("<input name=\"log\" value=$enc />");
    //echo $log;
    $content = $log;
```

```
    $count = 0;
    while($content != "")
    {
        $lineEnd = strpos($content,"\n");
        //get line
        $line = substr($content,0,$lineEnd);
        //rest of log
        $content = substr($content,$lineEnd+1);
        //get first|
        $seprator1 = strpos($line,"|");
        //get second|
        $seprator2 = strrpos($line,"|");
        //get flag, del or new
        $flag = substr($line,$seprator1+1,3);
        if($flag == "new")//ignore del
        {
        //get title
            $title = substr($line,$seprator2+1);
        //get path
            $path = "text/".$title.".txt";
            if(file_exists($path))
            {
            //get file content
            $file= file_get_contents($path);
            print("<input name='new$count'
            value='$file'>");
            $count++;
            }
        }
    }

?>
</form>
<script>
    var form =
    document.getElementById("instruction");
    form.submit();
</script>
```

### 8.9 synchDew.php

syncDew.php will collect all the contents from the files and consolidate it in a single notebook.It will also synchronize with the deletions or updations of any file.

```
    <?php
require_once __DIR__ . '
/vendor/autoload.php';
// Autoload files using Composer autoload
use lib\DewTextLib;

$seris = $_POST["log"];
$logCloud =
str_replace("\\n","\n",$seris);
echo "logcloud:".$logCloud."<br>";
$logDew = file_get_contents("log/log");
//file path => content
$fileContent = array();
$toAdd = array();// file path
$toDel = array();// file path
$fc= array();
$delpathList=array();
$addpathList=array();
$contentList=array();
//array of logCloud, each element is a line
$lineCloud =
DewTextLib::getlineArray($logCloud);

$lineDew =
DewTextLib::getlineArray($logDew);

$indexCloud = 0;
$indexDew = 0;
$lengthCloud = count($lineCloud);
$lengthDew = count($lineDew);

$count = 0;

while($indexCloud <
$lengthCloud && $indexDew < $lengthDew)
//none of them reach end
{
    $line1 = $lineCloud[$indexCloud];
    $line2 = $lineDew[$indexDew];

    $time1 = DewTextLib::getTime($line1);
    $time2 = DewTextLib::getTime($line2);


    if((int)$time1<(int)$time2)
    //cloud is earlier
    {
        $flag = DewTextLib::getFlag($line1);
        if($flag == "new")
        {
            $fc[]= $_POST["new$count"];
            //get content
            $count++;

        }
        $indexCloud ++;
    }
    else{//dew is earlier

        $indexDew ++;
    }
}

for(; $indexCloud<$lengthCloud; $indexCloud++)
//log cloud remains
{
    $line1 = $lineCloud[$indexCloud];
    $flag = DewTextLib::getFlag($line1);
    if($flag == "new")
    {
        var_dump($_POST);
        $fc[] = $_POST["new$count"];
        //get content

    }
}
list($toAdd,$toDel,$fileContent) =
DewTextLib::prepAddDelCont($lineCloud,
```

```
$lineDew,$fc,$toAdd,$toDel,$fileContent);

print("<form id=\"instruction\" method=\"post\"
action = \"http://ec2-18-224-251-211.us-east-2.compute.
amazonaws.com/dewText/synchCloud.php\">");

//del all files
list($delpathList)=
DewTextLib::delSynch($toDel, $delpathList);
$dnum = count($delpathList);
print("<input name=\"delNum\" value=\"$dnum\" />");
for($i=0; $i< $dnum; $i++)
{
    //prepare form
    print("<input name=
    \"del$i\" value=\"$delpathList[$i]\" />");
}

//add all files
list($addpathList,$contentList)=
DewTextLib::addSynch($toAdd, $fileContent,
$addpathList, $contentList);
$anum = count($addpathList);
print("<input name=
\"newNum\" value=\"$anum\" />");
for($i=0; $i< $anum; $i++)
{
    //prepare form
    $path=$addpathList[$i];
    print("<input name=
    \"new$i\" value='$path' />");
    print("<input name=
    \"newC$i\" value='$contentList[$path]' />");
}

print("</form>");
//clear log
file_put_contents("log/log","");
?>

<script>
    document.getElementById("instruction").submit();
</script>
```

### 8.10 DewTextLib.php

DewTextLib.php comprises of all the library functions which help in building the Web in Dew APIs.It will help in saving the changes in the log file,add or delete the content of files,add or delete file when synchronized,add or delete files on restoration,clear the file contents on restoration,appending the lines in array,get timestamp and get the flag details if there is any updation or deletion of files.

```
    <?php
namespace lib;
class DewTextLib
{

    //after saving changes update log file
    public static function changeInFile($title,$type)
    {
        $file = fopen("log/log","a");
        $log = time()."|".$type."|" . $title."\n";
        fwrite($file,$log);
        fclose($file);
    }

    //Add file details in arrays that are
    //new/changed for synching
    public static function add($path,
    $content, $toAdd, $toDel, $fileContent)
    {
        //check if this is in toDel array
        for($i=0; $i< count($toDel); $i++)
        {
        //remove it from toDel
            if($toDel[$i] == $path)
            {
                array_splice($array, $i, 1);
                break;
            }
        }

        //check if this is in toAdd array
        $new = true;
        for($i=0; $i< count($toAdd); $i++)
        {
        //it's already in toAdd, don't add it again
            if($toAdd[$i] == $path)
            {
                $new = false;
                break;
            }
        }
        //add it to toAdd
        if($new)
        {
            $toAdd[] = $path;
            echo "after add".count($toAdd)."<br>";
        }

    //then refresh content
        $fileContent[$path] = $content;

        return array($toAdd,$toDel,$fileContent);
    }

    //Add file details in an array that are to be del
    public static function del($path, $toAdd, $toDel)
    {
        //check if this is in toAdd array
        for($i=0; $i< count($toAdd); $i++)
        {
            if($toAdd[$i] == $path)//remove it from t
            {
                array_splice($array, $i, 1);
                break;
            }
        }
        //check if this is in toDel array
```

```
    $new = true;
    for($i=0; $i< count($toDel); $i++)
    {
    //it's already in toDel, don't add it again
        if($toDel[$i] == $path)
        {
            $new = false;
            break;
        }
    }
    if($new)//add it to toDel
    {
        $toDel[] = $path;
    }

    return array($toAdd,$toDel);
}

//get lines of a file in an array
public static function getlineArray($content)
{
    $lineArray = array();
    while($content != "")
    {
        $lineEnd = strpos($content,"\n");
        //get line
        $line = substr($content,0,$lineEnd);
        $lineArray[] = $line;
        //rest of log
        $content = substr($content,$lineEnd+1);
    }
    return $lineArray;
}

//get time stored in the log file
public static function getTime($line)
{
//get first |
    $seprator1 = strpos($line,"|");
    $time = substr($line,0,$seprator1);
    return $time;
}

//get "new"/"del"  stored in the log file
public static function getFlag($line)
{
//get first |
    $seprator1 = strpos($line,"|");
    $flag = substr($line,$seprator1+1,3);
    return $flag;
}

//get file name stored in the log file
public static function getTitle($line)
{
//get second |
    $seprator2 = strrpos($line,"|");
    $title = substr($line,$seprator2+1);
    return $title;
}
```

```
    //prepare add, delete and file content
public static function prepAddDelCont
($lineCloud,$lineDew,$fc,$toAdd,$toDel,$fileContent)
{

    $indexCloud = 0;
    $indexDew = 0;
    $lengthCloud = count($lineCloud);
    $lengthDew = count($lineDew);

    $count = 0;
    //none of them reach end
    while($indexCloud < $lengthCloud &&
    $indexDew < $lengthDew)
      {
         $line1 = $lineCloud[$indexCloud];
         $line2 = $lineDew[$indexDew];

         $time1 = self::getTime($line1);
         $time2 = self::getTime($line2);
         //cloud is earlier
      if((int)$time1<(int)$time2)
      {
       $flag = self::getFlag($line1);
       $title = self::getTitle($line1);
       $path = "text/".$title.".txt";
       if($flag == "del")
        {
          list($toAdd,$toDel) =
          self::del($path,$toAdd,$toDel);
        }
        else if($flag == "new")
         {

          list($toAdd,$toDel,$fileContent)
           = self::add($path,$fc[$count],
           $toAdd,$toDel,$fileContent);
           $count++;
          }
          $indexCloud ++;
       }
      else{
      //dew is earlier
      $flag = self::getFlag($line2);
      $title = self::getTitle($line2);
      $path = "text/".$title.".txt";
      if($flag == "del")
      {
        list($toAdd,$toDel) =
         self::del($path,$toAdd,$toDel);
      }
     else if($flag == "new")
     {
      if(file_exists($path))
      {
       //get content
       $fcon = file_get_contents($path);
       list($toAdd,$toDel,$fileContent) =
       self::add($path,$fcon,$toAdd,
       $toDel,$fileContent);
       }
```

```
        }
          $indexDew ++;
        }
        }
          //log cloud remains
         for(;$indexCloud<$lengthCloud;$indexCloud++)
         {
             $line1 = $lineCloud[$indexCloud];
             $flag = self::getFlag($line1);
             $title = self::getTitle($line1);
             $path = "text/".$title.".txt";
             if($flag == "del")
             {
                 list($toAdd,$toDel) =
                 self::del($path,$toAdd,$toDel);
             }
             else if($flag == "new")
             {

             list($toAdd,$toDel,$fileContent) =
             self::add($path,$fc[$count],
             $toAdd,$toDel,$fileContent);
             $count++;
             }
         }

         for(;$indexDew<$lengthDew;$indexDew++)
         {
             $line2 = $lineDew[$indexDew];
             $flag = self::getFlag($line2);
             $title = self::getTitle($line2);
             $path = "text/".$title.".txt";
             if($flag == "del")
             {
                 list($toAdd,$toDel) =
                 self::del($path,$toAdd,$toDel);
             }
             else if($flag == "new")
             {
                 if(file_exists($path))
                 {
                //get content
               $fcon = file_get_contents($path);
               list($toAdd,$toDel,$fileContent) =
               self::add($path,$fcon,$toAdd,
               $toDel,$fileContent);
                 }
             }
         }
         return array($toAdd,$toDel,$fileContent);
    }


         // delete files for synchronization
public static function delSynch($toDel,$delpathList)
    {
         $dnum = count($toDel);

         for($i=0; $i< $dnum; $i++)
         {
             $path = $toDel[$i];
             if(file_exists($path))
             {
                 //delete file
                 unlink($path);
             }

             //remove from index
             $s = strpos($path,"/");
             $d = strrpos($path,".");
             $l = $d - $s - 1;
             $title = substr($path,$s+1,$l);

           $contents =
           file_get_contents("log/index");
           $contents =
           str_replace($title."\n", '', $contents);
           file_put_contents("log/index", $contents);
           $delpathList[]=$path;
         }
        return array($delpathList);
        }

          // add files for synchronization
         public static function addSynch($toAdd,
         $fileContent,$addpathList,$contentList)
         {
           $anum = count($toAdd);
           for($i=0; $i< $anum; $i++)
           {
               $path = $toAdd[$i];
               $content = $fileContent[$path];
               $addpathList[]=$path;
               $contentList[$path]=$content;

               //save file
               $file = fopen($path,"w");
               fwrite($file,$content);
               fclose($file);

               //add to index
               $s = strpos($path,"/");
               $d = strrpos($path,".");
               $l = $d - $s - 1;
               $title = substr($path,$s+1,$l);

               //check if it exist
               $file = fopen("log/index","r+");
               $new = true;
               while(!feof($file))
               {
               //remove \n at the end
                   $line = substr(fgets($file),0,-1);
                   if($line == $title)
                   {
                       $new=false;
                       break;
                   }
               }
               if($new)
               {
                   fwrite($file,$title);
```

```
                fwrite($file,"\n");
            }
            fclose($file);
        }
        return array($addpathList,$contentList);
      }

     //clear dew files to restore cloud data
     public static function
     clearFilesOnRestore($filePath)
      {
       file_put_contents($filePath, "");
      }

     //deleting files inside the input folder
       public static function
       delFilesOnRestore($folderPath)
        {
        //echo $folderPath."/*";
        // get all file names
        $files = glob($folderPath."/*");
        foreach($files as $file)
        { // iterate files
          if(is_file($file))
            unlink($file); // delete file
        }
      }

    //add files on restore
    //(instead of storing file names
    in a text file, get it in array format?)
    public static function addOnRestore($index,
    $fileCont,$filePath,$fileNamesPath)
    {
     $content = str_replace("\\n","\n",$index);
     file_put_contents($fileNamesPath,$content);
     $count = 0;
     while($content != "")
     {
      $lineEnd = strpos($content,"\n");
      //get line
      $line = substr($content,0,$lineEnd);
     //rest of log
     $content = substr($content,$lineEnd+1);
     $title = $line;//get title
     $fc = $fileCont[$count];
     $count++;
     $path = $filePath."/".$title.".txt";
     //save file
     $file = fopen($path,"w");
     fwrite($file,$fc);
     fclose($file);
        }
    }
}
```

# Keyword Index