Formal Description of Dew Computing

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

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

Abstract—**Dew Computing is a specific cloud-related computing architecture that brings the computing closer to the user. Two main features of the dew computing include independence of external systems and collaboration with other cloud servers, making it an environment that can work in two modes, localized mode where all the services are provided within the internal local network perimeter and global mode, where it functions just as an intermediate device in the client-server cloud model. This article presents a formal description of dew computing as a service model and defines its two main operating modes with mathematical modeling functions.**

Index Terms—**Dew computing; Cloud computing; Formal specifications; Service modeling; Computational modeling; Turing machines; Servers.**

I. Introduction

Dew computing is an on-premises computer software-hardware organization paradigm in the cloud computing environment where the on-premises computer provides functionality that is independent of cloud services and, in the same time, it is collaborative with cloud services, too.

The goal of dew computing is to fully realize the potentials of on-premises computers and cloud services. Here, an on-premises computer is a cloud computing term. It means local computers, or non-cloud computers, which include personal computers (desktops, laptops), tablets, smartphones, servers, and clusters.

Wang [1] explains the essential dew architecture to be an extension of a classical ”client-server” architecture concept by introducing an intermediate dew server located close to the client. Although this looks similar to the cloudlet concept [2] as an edge computing concept [3], we will discuss the distinctions among them in Section VI.

Dew computing has two major features: *independence* and *collaboration* [4]. Independence means the on-premises computer is able to provide functionality offline. Collaboration means the dew computing application has to automatically exchange information with cloud services during its operation. Such collaboration includes synchronization, correlation, or other kinds of inter-operation.

Ray [5] discusses that besides these two features, two more features characterize dew computing. The first one addresses the ”microservice provision”, as defined by K. Skala et al.[6], which incorporate the theory of micro-service components located far away from ent virtual infrastructures. The second one specifies the ”scalability” in correlation to ”independence” and ”collaboration” as analyzed by Ristov et al. [7].

Dew computing is an emerging research area and application area. Although the theory and methods of dew computing are being shaped, many dew computing applications have already existed for many years, even before the dew computing concept was proposed.

To clarify the concept of dew computing and to further facilitate dew computing applications, we need to precisely determine which applications are dew computing applications. In other words, we need a model to describe dew computing applications. In this paper, we try to develop such a formal specification and computing model.

The rest of the paper is organized as follows. Section II gives the background and describes the dew computing modeling considerations. A model specification of a generic server model is presented in Section III and formal definitions of a dew service and dew server systems are introduced in Section IV. Examples of simplified modeling using service resources are elaborated in Section V. Section VI discusses the derived model and compares our formal specification with related approaches. Finally, Section VII is devoted to conclusions and future work correspondingly.

II. Background

This section elaborates a background for developing a formal specification of dew computing, including dew computing modelling considerations and service model essentials.

A. Dew Computing Modeling Considerations

A dew computing model should satisfy several requirements. For example, at least it should cover all forms of dew computing applications provided as a service system, and should not be restricted by a special group of applications.

For example, one simple way to model dew computing is that every dew computing system is considered an *internet*. Here the word internet starting with a lowercase *i* indicates that this is a group of computers that are connected through TCP/IP protocols. Considering this approach, each on-premises computer or a group of such computers are organized as an internet; websites are created on this internet; various services exist in this independent dew world.

In this model, all the dew applications communicate with the cloud which is the *Internet* with uppercase *I*; the relationship between a dew and the cloud is actually the relationship between a small internet and the big Internet. Although an internet and the Internet are different in their sizes, they are equal in terms of structure: they are both governed by TCP/IP protocols; the collaboration feature between a dew and the cloud can be interpreted as the communication among different internets.

This model is very practical and useful. It covers a broad range of applications, including dew servers and dewsites of the Web in Dew (WiD) applications.

This model has its drawbacks. The biggest problem is that It has limitations; only those applications that conform to TCP/IP protocols are covered. Theoretically, dew computing can be implemented using techniques other than those using TCP/IP protocols.

Analyzing the application domains, Rindos and Wang [8] identify Web in Dew (WiD) and Infrastructure as Dew (IaD) categories of dew computing. Later on, new categories are identified as Storage in Dew (STiD), Database in Dew (DiD), Software in Dew (SiD), Platform in Dew (PiD), and Data in Dew (DaD). Also, several research papers include dew computing in IoT architectures and applications [9].

Based on the above discussions, we need a dew computing model that covers a broader range of dew computing applications, that does not directly involve technical implementation details.

B. Server and service provision model

We consider that a service is provided by a complex system that interacts with other systems and performs a transformation of the requests to provide the output. This complex system is considered as a service provision system, and generally, is called a server system. Fig. 1 presents a basic model of the server that provides a generic service.

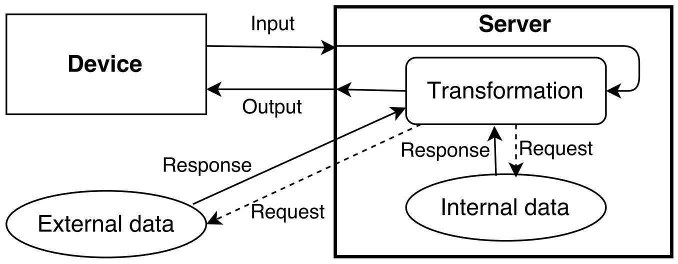

Fig. 1. Model of a basic server system providing a service to the device using external systems to compute results

To understand the basic model, Fig. 1 also contains a description of a device, which can be any computing or another device that can generate a service request (defined as an input to the server) and obtain a service response (defined as an output of the server). In a classic client-server model, the client is any computing device that sends a service request, and the server is the computing unit that generates a service response.

In addition, Fig. 1 specifies an external system which in essence is also another server system that generates a service response to a given service request. However, in this case, the analyzed server generates a service request as an input to the external system and expects a service response generated by the output of the external service system. The same terms that are associated with input and output to the external server system, in this case are treated as output and input correspondingly to the analyzed server system.

Let's dig deeper into the architecture of the basic server system and its service provision. Internally, the analyzed server system receives a service request defined by an input data I and calculates an output data O that is transferred as service response to the requestor device. The processing of the input uses a set of data transformations. As this is a computing system, it can generate the output based on the input data if the system complies to a definition of a combinatorial logic only.

However, the presented generic model of a server machine is a more complex system, and besides the input, it computes the output, also based on its internal state. In this case, the model of this computing machine uses finite state automata and the concept of memory that stores internal data. The memory itself can be treated as another service system that accepts a service request as an input and outputs a service response by providing data. Once again the communication to the memory is by a service. The analyzed server generates a memory service request to the memory in order to access the internal data and receives a service response as an output of the memory.

The overall server model also uses external data generated by the external server systems, as discussed earlier. This summarizes the definition of a generic server system to calculate an output based on three different inputs:

- *data input* I generated by a client device (service request),
- *internal data* M provided by memory as an internal service, and
- *external data* E provided by external services.

As a conclusion, a generic server system depends on other server systems, such as memory and external service providers. It transforms the data input I, using internal data M and external data E provided by corresponding service providers to generate output data O,

Therefore, a typical modeling will define a relation between the output O and inputs I, M, E by a specific transformation function ω. In addition, it will change internal data in the memory by a specific state transition function δ. Details on the development of a service model are specified in the next section.

III. A Generic Server Model

Our definition of a server model will be given in relation to a specification of a Turing machine.

A. Turing Machine definition

Now, let's analyze the connection of such a simple system with a definition of a Turing machine or other models of computations.

A (one-tape) Turing machine, according to Hopcroft and Ullman [10], can be formally defined as a 7-tuple $M = \{Q, \Gamma, b, \Sigma, \delta, q_0, F\}$ where

- Q is a finite, non-empty set of states,
- Γ is a finite, non-empty set of tape alphabet symbols,
- $b \in \Gamma$ is the blank symbol (the only symbol allowed to occur on the tape infinitely often at any step during the computation),
- $\Sigma \subseteq \Gamma \setminus \{b\}$ is the set of input symbols,
- $\delta : (Q \setminus F) \times \Gamma \to Q \times \Gamma \times \{L, R\}$ is a partial function called the transition function, where L is left shift, R is right shift. (A relatively uncommon variant allows "no shift", say N, as a third element of the latter set.) If δ is not defined by the current state and the current tape symbol, then the machine halts.
- $q_0 \in Q$ is the initial state,
- $F \subseteq Q$ is the set of final or accepting states. The initial tape contents will be accepted by M if it eventually halts in a state from F.

Anything that operates according to these specifications is a Turing machine. Any service we are analyzing in this paper is a Turing machine.

B. Specification of a generic server

Now let's correlate our conceptual model of the device and server systems to a definition of a Turing machine as a model of computation or other models of computers. A conventional Turing machine uses unlimited sequential memory, while real computers use limited random access memory. In computer science, random-access machine (RAM) is an abstract machine model identical to a multiple-register counter machine adding indirect addressing. Its equivalency to the universal Turing machine can be observed if the RAM's program and data are stored in the registers realizing a so-called von Neumann architecture.

We will give a mathematical definition of a server system that will be an approximation of a final state machine definition, that will reflect the modeled system behavior.

Definition 1: A general mathematical model of a *server system* is defined by:

- Θ is a finite non-empty set of data values called the data alphabet,
- Σ is a finite non-empty set of input symbols called the input alphabet, such that $\Sigma \subseteq \Theta$
- Γ is a finite non-empty set of output symbols called the output alphabet, such that $\Gamma \subseteq \Theta$
- S is a finite non-empty set of states,
- λ is a finite set of input data values, such that $\Lambda \subset \Sigma$,
- μ is a finite set of memory data values, such that $\mu \subset \Theta$,
- ν is a finite set of data values obtained by the external server systems, such that $\nu \subset \Theta$,
- ρ is a finite set of output data values obtained as a response of the server, such that $\rho \subset \Gamma$,
- s_0 is the initial or start state, such that $s_0 \in S$
- δ is a transition function given by (1)

$$\delta : S \times \Sigma \times \Theta \times \Theta \to S \tag{1}$$

- ω is an output function given by (2)

$$\omega : S \times \Sigma \times \Theta \times \Theta \to \Gamma \tag{2}$$

Θ is the set of all data values that can be treated as an input, output, and memory data value. In reality, it may represent any string, number or structured data. Σ and Γ are sets that represent the possible input and output values used by the system. The triple (Θ, Σ, Γ) consists of alphabets of all possible data used by the system, correspondingly as memory values, accepted input and generated output. In addition, the set of all possible states is S.

The quadruple (s_0, λ, μ, ν) consists of the initial system state s_0, actual input λ, requested memory values μ, requested external data values ν.

Two functions determine the behavior of the system.

Each server functions as a kind of finite state automaton and its behavior is determined by the state in which the automaton is currently in, usually known as an initial state s_0. The transition function actually defines the final (exit) state s_1 of the finite state automaton by providing a function (3) in a deterministic finite automaton. Instead of generating one possible exit state the transition function may return a set of states $\delta \subseteq S$ instead of one state s_1.

$$s_1 = \delta(s_0, \lambda, \mu, \nu) \tag{3}$$

The output function (4) is calculating the set of output values that the server system responds with.

$$\rho = \omega(s_0, \lambda, \mu, \nu) \tag{4}$$

It can be conventional that both the transition and output functions may not be defined on all possible states and data values defined by the input, memory or obtained by an external server system.

The behavior of the server system as a finite state automaton can be modeled as a Mealy machine, since its output depends on the input and internal state, and if the output depends only on the current state then it can be modeled as Moore machine. One may argue that the output function is obsolete if a proper conversion from a Moore to an output-equivalent Mealy machine (by labeling every edge with a transition symbol). However, we prefer to use this definition since it gives all relevant information for a faster understanding of the server system behavior.

This model of the server system has the same computational power as the Turing machine restricted to perform only read operations and moving in one direction only.

IV. A Dew Server formal specification

A. Dew server modeling

A dew server is a specific server system with certain restrictions on the presented model. The main difference is that the dew server can work in a closed environment without the use of external server systems.

Therefore, a dew server's definition is the same as the previous one, but it needs to be extended with the availability of the external server systems and reflect Internet connectivity.

Let's denote by $\eta \in \mathcal{B} = \{0, 1\}$ the availability of Internet connection and the external server systems, by a simple rule that 0 means no availability, and 1 the availability.

The transition function will be modeled by (5) and the output (exit) state will be given by (6).

$$\delta : S \times \Sigma \times \Theta \times \Theta \times \mathcal{B} \to S \tag{5}$$

$$s_1 = \delta(s_0, \lambda, \mu, \nu, \eta) \tag{6}$$

Note, that in the case of a nondeterministic finite automaton the output may be a set of states Δ, instead of one state s_1.

Similarly, the output function ω will be modeled by (7) and the output data set will be given by (8).

$$\omega : S \times \Sigma \times \Theta \times \Theta \times \mathcal{B} \to \Gamma \tag{7}$$

$$\rho = \omega(s_0, \lambda, \mu, \nu, \eta) \tag{8}$$

If the availability is $\eta = 1$ then the dew server model is exactly the same as the general server model. However, if the availability is $\eta = 0$ then the requested external server system value will be undefined or get a *not available* data value N/A. Therefore, the definition of all possible data values that the dew server is operating will need to be extended to Θ' defined by (9). This also applies to the sets of input and output values, correspondingly Σ' and Γ' as given by (9).

$$\begin{aligned} \Theta' &= \Theta \cup \{N/A\} \\ \Sigma' &= \Sigma \cup \{N/A\} \\ \Gamma' &= \Gamma \cup \{N/A\} \end{aligned} \tag{9}$$

This means that in the case of unavailability of external server systems ($\eta = 0$), the associated value that the system will continue to use is $\nu = N/A$ and correspondingly the output of the dew server will be N/A. So, the dew server still performs the specified functions, but in the case of unavailability, it gets a specific value. However, not all server requests will need a value from external servers, and in this case, the dew server will continue to function as it was initially intended for.

Definition 2: A *dew server system* is a server system with the following constraints to the Def. 1.

- Θ' is a finite non-empty set of data values called the data alphabet, defined by (9),
- Σ' is a finite non-empty set of input symbols called the input alphabet, defined by (9),
- Γ' is a finite non-empty set of output symbols called the output alphabet, defined by (9),
- λ is a finite set of input data values, such that $\Lambda \subset \Sigma'$,
- μ is a finite set of memory data values, such that $\mu \subset \Theta'$,
- ν is a finite set of data values obtained by the external server systems, such that $\nu \subset \Theta'$,
- ρ is a finite set of output data values obtained as a response of the server, such that $\rho \subset \Gamma'$,
- δ is a transition function given by (5)
- ω is an output function given by (7)

Finally, we define dew computing based on dew server systems:

Definition 3:

If dew server systems are used in a computing process, this computing process is called *dew computing*.

V. Examples of simplified modeling of service resources

In this section, we will present examples of simplified models of the service resources. A service resource is either an internal memory or external server.

A memory is an internal resource, realized as a set of memory locations. A small part of the memory is used in conventional computers as internal registers, which is a smaller memory located next to the processing unit. In this sense, a memory contains a larger number of memory locations. Each memory location can be accessed by specifying its address and specifying the memory access instruction.

The external server is also a service resource. It can be accessed based on the availability function η. If the external server is available, then the access is defined as a request with input parameters and external server address.

We will continue with specifying a simplified mathematical model of a memory and server.

A. A simplified model of a memory system

A memory is a specific server system. The service resources, in this case, belong to a set of memory locations $\mathcal{M}$. The unique identifier of a memory location is its address α.

The memory can have different resource sets, such as internal registers, and bulk memory. Therefore, the service that a memory is providing needs to make a distinction to which resource a service request is referred to. In this case, the resource identifier τ is used as an input parameter in the service request, besides the address.

In addition, the memory service function should be determined as "store" or "load", by the function id φ.

Definition 4: A *memory system* is a simplified server system with the following constraints to Def. 1:

- α is the address of a memory location, such that $\alpha \in \{0, 1, \ldots, 2^M - 1\}$ in a memory that contains 2^M locations, where M is a positive integer,
- resource id $\tau \in 0, 1, \ldots R - 1$ in a system that uses $R > 0$ resources,
- function id $\varphi \in \{0, 1\}$, where 0 means memory load, and 1 means memory store,
- λ is the input set defined as a set of the address, resource id, and function id, that is $\lambda = \{\alpha, \tau, \varphi\}$,
- ρ is the output value getting a value of performing the output function $\rho = \omega(\alpha, \tau, \varphi)$,
- s_0 is the initial state that represents a *ready* state that waits for an input triple to perform an activity according to the specified function φ. When the service is requested, the state changes to a *busy* state s_1. Once the service response is computed and an output is sent, the state changes to s_0.

If a service is requested while the internal state is $busy$ then the request is held in a queue until the internal state reaches the ready state s_0.

B. A simplified model of a digital service system

The service resources are determined by the state, internal memory and external server resources. Each state $s \in S$ can be determined by the internal registers. The set of all internal registers $\mathcal{R}$ and the set of all memory locations $\mathcal{M}$ determine the internal resources.

Note that in Def. 1 we have defined the values that can be exchanged between the systems. Here we define the locations where these values will be stored and used. Therefore, the set $\mathcal{I} = \mathcal{R} \cup \mathcal{M}$ is a representation of internal resources, as a set of all internal memory locations. Note that these sets should not be mixed with data values μ or the data alphabet Θ.

External resources are determined by a finite number of external servers, with a set of $\mathcal{E}$ memory locations.

Both the Internal and external resources define the service resources.

A service resource may belong to the dew computing concept if it functions both with or without Internet availability. In addition, it needs to be close to the service requestor to belong to a class of dew computing servers.

VI. Discussion

A. Dew Computing Features

Two main features of the dew computing concept will be analyzed in correspondence to our formal specification.

Independence is addressed by the availability function equal to 0 ($\eta = 0$). In our formal specification, the dew server will continue to deliver its services in the case without Internet availability.

Collaboration is addressed by the availability function equal to 1 ($\eta = 1$). In our formal specification, the dew server can exchange information with the cloud servers, which means synchronization of content or control parameters. Also, in this case, the dew server will continue to deliver its services.

Collaboration enables at least the following:

- *upward synchronization* to transfer information to the cloud server,
- *downward synchronization* to transfer information to the dew server, and
- *new service specification* to define new services of the dew server.

Upward synchronization means that the exchange of information with the cloud server is such that data from the dew server is sent to the cloud in order to be available for a wider environment. It will not change the service definition represented by the transformation functions, including transition function δ and output function ω.

Note that the independence feature specifies that the collaboration is not crucial for delivery of services. It means that the collaboration means that the cloud server will receive some information and not have any impact on the performance of the dew server.

The upward synchronization is extensively used in an application of dew computing solutions for IoT, such as wearable eHealth sensors that deliver data to a smartphone, being a dew server, that functions with and without Internet availability. In the case of Internet availability, the smartphone sends all received data to the cloud server.

Downward synchronization means that the exchange of information with cloud servers may change the set of internal memory values μ. Therefore, this will change the internal state of the dew server and initiate results (service output) different from those that will be obtained if the internal state was not changed. For example, this feature is used to define a new content of the dew server that delivers localized web content.

One more function may be used with the collaboration function. It may be used to define a new service with specific transition function δ and output function ω. This makes the dew server a rich environment to deliver services.

B. Comparison with related approaches

Let's compare our server system definition with other definitions and specifications. Zhang et al. [11] specify a service model as a Feedback Control-based Services System. They specify the input of a service consumer and the output as a fulfillment of a service requestor. A specific sensor feeds back the response back in the system for continuous improvement and business transformation. The interior components are service activities/processes, service resources, service information, service people and service partners. In addition, they set the internal goal to increase the profit and decrease the costs, and external goal to reach service level agreement.

In our model, we have identified the service resources and service activities/processes. The service resources are the internal memory and external server resources, and also the service information kept as a state in our definition. The transformation is defined by the transition and output functions that compute the next service information and the output. Our model refers to data computation and does not address people and partners. Also, we do not analyze the business aspects by setting business goals, we describe a digital version of a dew server.

C. Dew computing vs. Edge computing

The essence of edge computing is to push applications, data, and services away from central servers (core) to the edge of a network. It is based on the core-edge topology. While most devices are connected with core-edge topology at the current time, some devices are connected with mesh topologies, such as NYCmesh, Detroit's Equitable Internet Initiative, and eastern Afghanistan's FabFi.

Dew computing is a different approach. It emphasizes its independent operation without the Internet connection and its collaboration with cloud services. Dew computing does not rely on network topology.

Comparing dew computing and edge computing, dew computing is featured by its collaboration with cloud services, and it is not restricted by core-edge topology. Edge computing also has its advantages. Because currently, core-edge topology is still dominant in the Internet, edge computing encourages researchers and professionals to move applications to the edge of the network, which goes toward the same direction as dew computing.

A cloudlet is a small-scale cloud datacenter located at the edge of the Internet. It is the middle tier of a 3-tier hierarchy: mobile device - cloudlet - cloud. Cloudlet is close to a mobile device but not on the mobile device. On the other hand, dew servers, if introduced, should be on the mobile devices.

Therefore, the formal description of edge computing could be developed in a similar manner as the development of dew computing formal description, but including the network topology.

VII. Conclusion

In this paper, we have introduced a formal specification of the dew computing concept.

Dew computing is based on delivering of (micro) services by a dew server, functioning independent of a wider Internet-based environment, although it can collaborate with other cloud servers in the case of Internet availability.

Our definition of a server that delivers a service is based on a mathematical specification of a Turing machine, adapted to the availability of internal and external resources. In this sense, internal resources are defined as memory that specifies an internal state of the dew server, while the external resources are defined by other services that depend on the availability of the Internet.

Using our introduced formal specification, we have presented examples of simplified systems, including internal memory, and external servers that deliver external services to enable external data to our analyzed service. This completes the formal specification, as it uses a recursive definition of a memory and external servers by its initial definition.

In addition, we have compared edge computing and dew computing with regards to our definition, and indicated the essential difference between dew computing and edge computing.

For future research, the first task would be to fine-tune the definition of a dew server system. We have noticed that one feature of dew server system has not been grasped by Definition 2. This feature is restricted client access to the dew server. Because such feature involves the overall description of the client-server systems, we postpone such discussion to a later time. Future research may also involve the analysis of performance issues that could provide a better insight into our formal specification.

References

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

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

[3] 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.

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

[5] P. P. Ray, "An introduction to dew computing: Definition, concept and implications," *IEEE Access*, vol. 6, pp. 723–737, 2018.

[6] 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.

[7] S. Ristov, K. Cvetkov, and M. Gusev, "Implementation of a horizontal scalable balancer for dew computing services," *Scalable Computing: Practice and Experience*, vol. 17, no. 2, pp. 79–90, 2016.

[8] A. Rindos and Y. Wang, "Dew computing: The complementary piece of cloud computing," in *Big Data and Cloud Computing (BDCloud), Social Computing and Networking (SocialCom), Sustainable Computing and Communications (SustainCom)(BDCloud-SocialCom-SustainCom), 2016 IEEE International Conferences on*. IEEE, 2016, pp. 15–20.

[9] M. Gusev, "A dew computing solution for IoT streaming devices," in *Information and Communication Technology, Electronics and Microelectronics (MIPRO), 2017 40th International Convention on*. IEEE, 2017, pp. 387–392.

[10] J. Hopcroft and J. Ullman, *Introduction to Automata Theory, Languages and Computation*. Addison-Wesley, Reading Mass, 1979.

[11] L.-J. Zhang, J. Zhang, and H. Cai, *Services computing*. Springer Science & Business Media, 2008.