Journal of Intelligent & Fuzzy Systems xx (20xx) x–xx
DOI:10.3233/JIFS-189841
IOS Press

Allocation and application of computer software system based on system architecture

Xiaolian Di*
Shaanxi Open University, Xi'an, Shaanxi, China

Abstract. With the improvement of software system complexity and frequent updating of user requirements, the requirements of the information software development industry for information construction are constantly improved, and the quality and management requirements of software products researched and developed are also constantly improved. Project managers in the information software development industry gradually realize the importance and necessity of software system deployment. It requires scientific, timely, effective and clear work. Software system deployment system for task division and task monitoring. Based on the research results at home and abroad, this paper studies the deployment of computer software system based on event-driven architecture by using a discrete Fourier transform algorithm, decision tree algorithm and parallel algorithm. By comparing and optimizing the advantages and disadvantages of discrete Fourier transform algorithm, decision tree algorithm and parallel algorithm. This paper studies the unified management, scheduling and allocation of computer software resources. The results show that after using the research model, the data error is controlled within 5%, and the overall data accuracy is improved by 15% compared with the previous methods, which have certain practical value.

Keywords: System architecture, system deployment, computer software, software system deployment

1. Introduction

With the coming of 5 g mobile communication [1], the evolving packet core network (EPC) all IP architecture is greatly challenged by users' requirements for higher data rate and more reliable end-to-end connection and operators' requirements for low operating costs. These challenges can be met by a software-defined optical network (SDON), which can dynamically allocate resources according to users' needs. Zhao y proposed a mobile core network architecture based on SDN. Zhao y designed a software-defined network (SDN) controller to achieve coordinated control of different entities in the EPC network [2]. WSN software is usually developed based on the abstraction of the underlying operating system, while business process development uses high-level language and tools. This situation makes wireless sensor networks on the edge of the industry. Make sense solves this problem by simplifying the integration of WSN and the business process. Mottola l uses a BPMN model that extends the WSN specific structure to specify application behavior across traditional business process execution environments and WSN itself, which will be equipped with application-specific software [3].To meet the special requirements of data center congestion control, Wang J proposed many solutions, such as DCCP, d2tcp and D3. However, some deployment defects (including significant modifications to switching hardware, an operating system protocol stack, and/or upper-level applications) and switching ECN requirements (which are not always available in

*Corresponding author. Xiaolian Di, Shaanxi Open University, Xi'an 710119, Shaanxi, China. E-mail: Dixiaolian1@163.com.

existing data centers) limit the deployment of these solutions [4].

One of the main challenges of cloud-based service delivery is to deploy coordinated control of the application layer and network layer resources to provide adaptive service data delivery and sufficient user service experience [5]. Cerroni w proposes a cross-layer resource scheduling signaling framework, in which the service awareness provided by session control is effectively combined with the flexibility of software-defined network control [6, 7]. When all steps are completed, the process is marked as successful. Zach P proposes an automated solution that provides a possibility of how to easily manage computer classrooms and virtual operating systems [8, 9].

Based on the research results at home and abroad, this paper improves the previous algorithms, establishes a research model of computer software system deployment based on event-driven architecture, and compares the advantages and disadvantages of discrete Fourier transform algorithm, decision tree algorithm and parallel algorithm. The deployment of a computer software system and the development of information software based on event driven architecture are studied [10].

2. Computer software system deployment

2.1. Software system deployment

At present, there are two research contents related to dynamic configuration, the first is software architecture and ADL language, the second is related to software composition. For the first, the main achievement of software architecture is ADL (Architecture Description Language). Through this language, software architecture shows support for dynamic configuration. During the running process of dynamic configuration, components not affected by it can run normally, and the whole process does not need to restart or compile the system [11, 12]. The research of dynamic configuration is inseparable from the analysis of software architecture. In recent years, scholars from all walks of life have proposed the definition of software architecture, such as the garlan and SHAW model, CFRP model, and "4 + 1" model. The so-called "4 + 1" is an effective combination of development, process, logic and physical views, which details the relationship between user requirements and architecture. Second, the language related to software composition [13]. There is a close relationship between the configuration and composition of software. With the development of software, the word "configuration" began to dominate [14].

The software configuration management system is an important subsystem of the entire modern project management information system. The configuration management system includes configuration file identification; auditing, change management control and status view [15–17]. Through development and configuration activities, the integrity and consistency of project work products in the software life cycle are ensured. The main functions include configuration library management, configuration item management, label management, baseline management, and version control. Change management, configuration review, report management, approval management, statistical analysis, etc., enable developers, testers, project managers, configuration managers, quality assurance personnel, project change control committees, software engineering process teams and other engineering personnel, managers and Stakeholders can conveniently and timely obtain useful information from different needs [18, 19].

2.2. Discrete fourier transform algorithm

Fourier transform is one of the most important numerical operations and has been widely used in signal processing, optics and other fields. The concept of fractional Fourier transform is used as a new method to solve the Schrodinger equation [20, 21]. Although the definition of the fractional Fourier transform is not given from a mathematical point of view, its physical significance has been proven in optical systems in subsequent research. The traditional definition of fractional Fourier transform is not unique, so we can use the same rules to define a new type of fractional Fourier transform, which can smoothly transition in the time domain and frequency domain [22]. The starting point comes from the mathematical properties of the fractional Fourier transform, so the definition will not rely on special physical processes [23]. The definition expression of Fourier transform is as follows:

If a signal g (t) satisfies the absolute integrability condition in an infinite interval, that is:

$$\int_{-\infty}^{+\infty} |g(t)|dt \leq \infty \tag{1}$$

Then its normalized Fourier transform is defined as:

$$G(\omega) = \zeta[G(t)] = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{+\infty} g(t) e^{\omega} dt \quad (2)$$

Define its inverse transformation as:

$$g(t) = \zeta^{-1}[G(\omega)] = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{+\infty} G(t) e^{\omega} d\omega \quad (3)$$

Discrete Fourier transform is the representation of continuous Fourier transform after sampling on frequency and time. The two ends of the transformation can be regarded as the main value sequence of the discrete periodic signal. First, discretize the signal in the time domain, then obtain the continuous Fourier transform, and then discretize the signal in the frequency domain [24–26].

To calculate the convolution with a computer, a finite interval is used to approximate the infinite interval, and then the equidistant dispersion is performed to obtain a finite number of nodes [27]. In this article, we can refer to discrete convolution as convolution or linear convolution. Its definition is as follows.

The convolution of two complex sequences $X_n(n=0, 1,..., M_1\text{-}1,)$ and $H_n(n=0, 1,..., M_2\text{-}1,)$ with length M_1 and M_2 respectively refers to a sequence with length $M_1 + M_2 - 1$.

$$y_n = \sum_{\substack{K=0 \\ 0 \le n-k \prec M_2}}^{M_1\text{-}1} h_{n\text{-}k} x_k = \sum_{\substack{K=0 \\ 0 \le n-k \prec M_1}}^{M_2\text{-}1} h_k x_{n\text{-}k} \quad (4)$$

The cyclic convolution of two sequences X_n H_n and is defined as:

$$y_n = \sum_{K=0}^{N\text{-}1} x_k h\,(n\text{-}k)\, N = \sum_{K=0}^{N\text{-}1} h_k x\,(n\text{-}k) n \quad (5)$$

Generally, convolution can be calculated by cyclic convolution. Convolution of sequences $x_0, x_1, X_2 ..... X_{m1-1}$ and $H_0, H_1, ... H_{m2-2}$ can be calculated by the cyclic convolution of two sequences of length n [28].

Discrete Fourier Transform (DFT) is the most basic and important operation in digital signal processing. In addition to spectrum analysis, convolution and correlation can also be achieved by a computer. The decomposition algorithm in wavelet analysis can also be transformed into convolution, so it can be implemented with DFT [29].

2.3. Decision tree algorithm

ID3 algorithm is divided into two stages: building a decision tree and classification test. In the stage of building a decision tree, how to choose attributes to divide parent nodes is the key to this problem. It uses information entropy as a standard to measure the balance of the number of samples in each category of the sample set and then uses information gain (ie, the reduction of information entropy) as a standard to select the attributes used to split the sample set, to realize the establishment and classification of the tree [30, 31]. In these two classification problems, if the ratio of the number of positive samples in the sample sett to the total number of samples is p, and the ratio of the number of samples belonging to the negative category is 1-p, then the information entropy of the sample sett is defined as follows:

$$\mathrm{IT(p)} = \text{-}\ \mathrm{plog_2^p} \text{-}\ (1\ \mathrm{p})\mathrm{log_2^1} \text{-}\ \mathrm{p} \quad (6)$$

The concept of entropy originated in physics. In physics, entropy is used to measure the degree of disorder in a thermodynamic system. In informatics, entropy is a measure of uncertainty.

Let the data partition d be the training set of labeled class tuples. Suppose the category label attribute has m different values.

This article defines m different Ci, where the value range of i is 1, 2, 3, m. [32]. for the sample set D the entropy is:

$$\inf o \left(- \sum_{i\text{-}1}^{m} p_i \log_2 (p_i) \right) \quad (7)$$

Where P_i is the non-zero probability that any tuple in D belongs to class C_i. Let attribute a be a discrete value. Attribute a divides d into v partitions or subsets D1, D2 and DV, where Dj contains a tuple with a value in D. These partitions correspond to branches growing from node n. The following equation gives the expected information divided into subsets by a:

$$\inf o_A = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times \inf o\left(D_j\right) \quad (8)$$

Among them, | Dj |, |D| serves as the weight of the jth partition. The smaller the value of InfoA(D), the higher the purity of the partition [32].

3. Computer software system deployment design based on event-driven architecture

When multiple developers jointly develop and modify the same module or file of the project, the members of the development team need to know each other's development progress and reduce the conflicts in the development, which needs to inform the relevant change developers in time. Besides, for the enterprise, the project is in different stages of development, and documents of different stages need to be reviewed. When the document needs to be approved by relevant personnel or consulted by others, the audit team members and relevant personnel shall be informed in time.

The main task of system deployment is to determine the solution of the system, that is to execute the detailed design of the software test scheme generation subsystem, determine the design process of the system, and divide all functional modules in detail [33]. The basic goal of this phase is to determine how to achieve the specific requirements of the content. The design scheme of computer software deployment is as follows:

Option 1: platform generator deployment. The platform generator is the key link in the test scheme generation subsystem of software system deployment. It determines the distribution status of test cases in the new software test scheme on the test platform and is an important process to improve the test effectiveness and coverage.

Option 2: page size generator deployment, which is an important part of generating page size information in the software system deployment test scheme generation subsystem, determines the distribution status of test cases for each page size in the new software test scheme, and is the core process of the process, to improve the effectiveness, comprehensiveness and rationality of page size test.

Option 3: lock mechanism generator deployment, which is an important part of software system deployment test scheme generation subsystem about obtaining locking mechanism. This part determines the distribution status of test cases in each locking mechanism and other important indicators in the new software test scheme. This is an important link to ensure the effectiveness, accuracy and high coverage of the locking mechanism during testing.

The overall implementation process of the software system deployment test scheme generation subsystem is the key process of the effective combination of the whole system modules. Its main work is to reasonably install the deployment platform generator module, page size generator module, locking mechanism generator module. Each system module can work together without affecting the other.

Database: Oracle can run on all mainstream platforms (including windows). Fully support all industry standards and reach the highest level of ISO certification. The application is very flexible and can provide both a GUI and command line. The operation under windows and UNIX is the same. Compared with MSSQL, Oracle can be used on UNIX and Windows platforms. Software configuration management system not only requires the application program on UNIX but also needs to be able to modify the information at any time under windows. Due to the existence of many subsystems in wireless R & D, Center D has many branch projects, and the amount of data processing is very large. In the case of a large amount of data, the Oracle partition table has more advantages than the SQL server. Therefore, considering the demand for multi-platform and a large amount of data, we choose Oracle as the third-party support of software configuration management system. Version management: ClearCase. ClearCase automatically tracks changes in each file and directory, and supports parallel development through branching and merging capabilities. In the software development environment, ClearCase can realize the version control of each object type (including source code, binary file, directory content, executable file, document, test package, compiler, library file, etc.). ClearCase provides capabilities far beyond resource control and helps teams establish a secure version history for each type of information they process when developing software. And ClearCase can be integrated with a variety of development tools. Is the preferred configuration management tool.

4. Analysis of simulation results of computer software system deployment based on system architecture

4.1. Feasibility analysis of computer software system deployment

As shown in Table 1, the scheduling results of the three schemes and the scheduling results obtained by the local solver when the performance index of the resource provider is M1. When the performance index of resource provider is m2, option 1, option 2, and option 3 represent the scheduling results of

Table 1
Performance comparison of algorithms

Scheme name	Attribute	Continuous property	Number of instances	Size
Option 1	16	6	48652	3.7M
Option 2	19	10	45862	3.3M
Option 3	35	7	49625	4.1M

the first, second, and third feasible solutions and the scheduling results obtained by the local solver. Most scheduling problems are NP-hard. Although the characteristics of constraints and decision variables in this model are the same as those of traditional scheduling problems, the objective function is a nonlinear function related to decision variables. Therefore, compared with the traditional scheduling problem, the objective function of this paper is more complex and the model is difficult to solve. Therefore, for the Nash bargaining model on a single machine, this paper uses commercial software local solver to solve the problem, which can provide the solution of local solver under the corresponding conditions. As the measurement basis of simulation results of three feasible scheduling schemes.

It can be seen from the simulation results that the algorithm of computer software system deployment and application resource allocation based on event-driven architecture is simulated. Consider a distributed cloud computing platform composed of five data centers with different geographic locations [34]. Each data center has the same capacity, but its benchmark resource price and benchmark latency are not the same. In each slot, the number of virtual machines required by the service and the corresponding service characteristics is randomly generated. There is a certain possibility that services executed in the data center will leave. Specific stimulation parameters are shown in the table.

4.2. *Software system deployment and application resource allocation algorithm performance analysis*

As shown in Fig. 1, the resource utilization curves of three scenarios in slots 100 to 200 are shown. It can be seen from the figure that there is no significant difference in the load of the three schemes under the resource allocation algorithm proposed in this paper, which indicates that the price model used in the algorithm can balance the load. According to the simulation model and related simulation parameters, the average load of the five data centers is 60%, and the

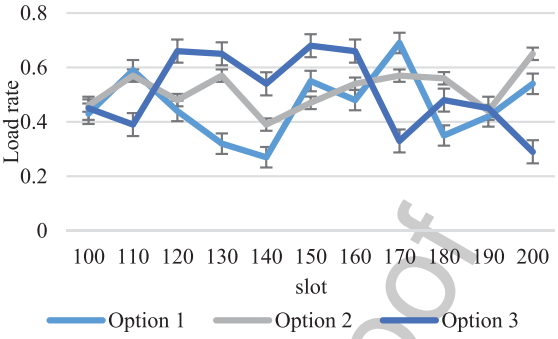

Fig. 1. Resource utilization curve.

load of each data center in Fig. 3.5 fluctuates between 40% and 70%, indicating that the proposed algorithm can achieve the goal of load balancing among the schemes. This paper shows the relationship between resource utilization, delay and resource price. It can be seen from the figure that the delay and resource price are directly proportional to the resource utilization rate, and have a high correlation, which is consistent with the theory of the simulation model, indicating the correctness of the simulation.

In this paper, a service with a delay factor greater than 0.7 is regarded as a time-sensitive service, and service with a price factor greater than 0.7 is regarded as a price-sensitive service. This paper calculates the cumulative delay value and resource cost of delay-sensitive service and price-sensitive service under different virtual machine deployment strategies. The results are similar to those in the first line, which shows that the proposed algorithm can reduce the deployment cost of a computer software system for corresponding users.

As shown in Fig. 2, vertically, for users with a centralized virtual machine deployment strategy, the difference between different types of service delay and resource price is more obvious, which indicates that the algorithm has better benefits for users using a centralized virtual machine deployment strategy. The method of computer software system deployment based on event-driven architecture proposed in this paper can allocate corresponding resources for users according to their business characteristics, thus reducing the cost of users. The main reason is that the number of times of data compression is reduced in the process of data compression. Therefore, the running time of the algorithm is greatly reduced.

Taking the completion time as the standard, the algorithm has achieved good results in task scheduling. In large-scale operations, if the algorithm runs

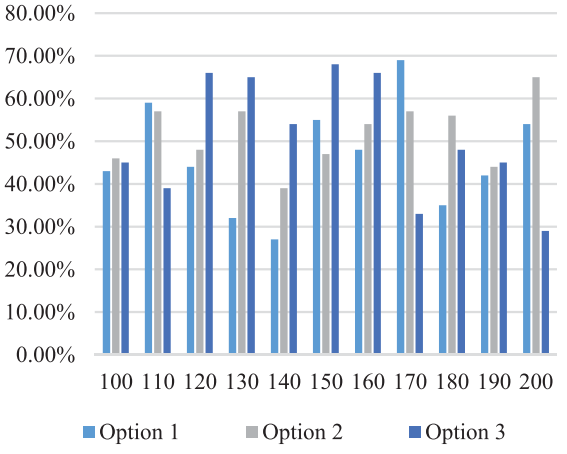

Fig. 2. The relationship between resource utilization, time delay and price.

for a long time, it will bring a certain burden to the scheduling problem and make the algorithm unsuitable. In terms of time cost, heuristic algorithms usually have satisfactory results, so they are widely used. In multi-objective optimization problems, some algorithms do not take optimization measures for all objectives, some of which are set as the maximum or minimum value, for example, setting the maximum value of the period, and then optimizing the cost target, or optimizing the execution time under the premise of setting the maximum total cost. Some algorithms first set the priority according to the scheduling objectives, and then optimize the multi-objective scheduling problem in scientific workflow according to the priority.

4.3. *Performance index analysis of software system deployment results*

As shown in Fig. 3, when the performance index of the resource provider is a specific value, the results of the three feasible scheduling schemes, option1, option2, and OPTION3, are compared with the scheduling results of the model. It is found that with the increase of the specific value, the changing trend of the performance indicators of the three scheduling results is basically the same, but the values of individual indicators are slightly different. With the increase of the specific value, the customer performance index has gradually deteriorated, and the deterioration degree may be greater than the optimization degree of the resource provider performance index, and then the overall performance index gradually worsens [35]. Therefore, among the

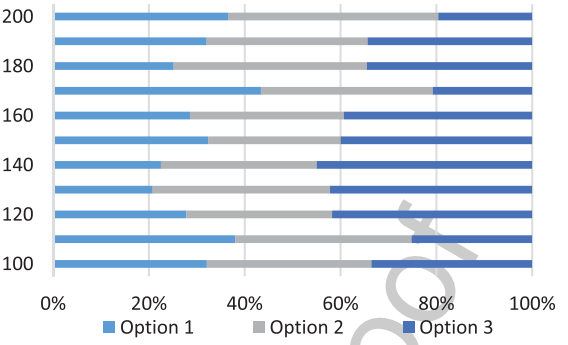

Fig. 3. Performance comparison of resource allocation algorithms.

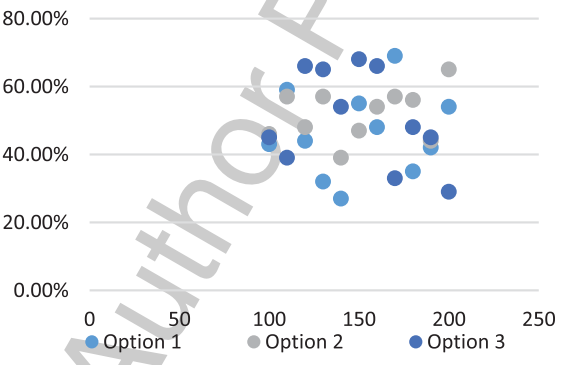

Fig. 4. Distribution of solution individuals in the target space.

three schemes, the third scheduling scheme performs poorly.

As shown in Fig. 4, besides, no matter whether the performance index of the resource provider is a specific value or a random value, with the increase of the value, the deviation degree between the scheduling results of the three feasible schemes and the results of the computer solver gradually increases. When $a = 0.1$, the three feasible scheduling schemes can generally approach or equal to the solution of the computer Solver: when $a = 0.5$, the results of feasible scheduling in some examples deviate by about 10% of the solution of the computer solver; when $a = 0.9$, the three feasible scheduling schemes are almost difficult to reach the predicted value of the computer solver. The reason for the above changes is the bargaining power coefficient of the resource provider. The resource provider is the dominant factor affecting the value of the objective function. When allocating resources in the feasible scheduling, more attention will be paid to the demands of the resource provider, and very little attention will be paid to the customer's demand, which seriously worsens the customer's self-interest performance index, then the

deviation between the results of feasible scheduling and that of computer solver is caused.

Based on the feasible scheduling scheme, the customer bargaining power coefficient in the currently occupied resource segment is rearranged. In other words, it does not degrade the resource provider's performance metrics. In the current non-idle resource segment, the most preferred resource segment location is selected from the customers with the largest bargaining power coefficient, and the location is exchanged with the current customers occupying the resource segment. The advantage of this optimization method is that it only reselects the currently occupied resource segments, and does not affect any utility value of resource providers, but only changes the utility between customers. The ultimate goal is to improve the overall target value.

5. Conclusions

With the complexity of the software system, user requirements and frequent software updates, deployment management has gradually become an important control process in the software life cycle. A good deployment management process can cover all stages of software development and maintenance, and it also plays an important role in the macro-management of the software development process. For the large-scale software system development carried out by thousands of people at the same time, due to a large number of personnel, a large amount of code and complex system, software deployment management is different from small and medium-sized software development. One of the important functions of software deployment management is to comprehensively manage each deployment item and monitor the status of each deployment item. Program code is the most important deployment project. In the development or maintenance of a large software system, code files can be modified by more than ten subitems at the same time. In this case, the normal branch system is difficult to ensure the stable operation of the system. Therefore, it is necessary to strengthen the management of the branch office and establish a software deployment management system suitable for the parallel development of thousands of people.

Based on the research results at home and abroad, this paper studies the deployment of computer software system based on system architecture by using a discrete Fourier transform algorithm, decision tree algorithm and parallel algorithm. In this paper, the development history of a computer software system, the application of system deployment technology are discussed in detail, which provides reliable data basis for the deployment and optimization of a computer software system based on system architecture. The research results show that: the computer software system deployment platform based on system architecture provides a unified interface for the detection and maintenance of the whole system software state it can standardize the standard of software status, simplify the management and operation, effectively improve the control ability of software and ensure the stability of system software.

Due to the limited experience of the author, there are still some deficiencies in the research. Although the software deployment system based on system architecture can meet the needs of enterprise software management, the system still has some defects. The research details of this paper may still be insufficient and under consideration, and support for different system environments has not been considered. In the future, the author will further deepen the research.

References

[1] P. Yu, F. Zhou, X. Zhang, X. Qiu, M. Kadoch and M. Cheriet, Deep Learning-Based Resource Allocation for 5G Broadband TV Service, *IEEE Transactions on Broadcasting* (2020).

[2] Y. Zhao, Z. Chen, J. Zhang, et al., Dynamic optical resource allocation for mobile core networks with software defined elastic optical networking, *Optics Express* **24**(15) (2016), 16659–16673.

[3] D. Prior, I. Biscoe, M. Rofe, et al., Designing a system for Online Orchestra: Computer hardware and software, *Journal of Music* **10**(2) (2017), 185–196.

[4] L. Mottola, G.P. Picco, F.J. Opperman, et al., makeSense: Simplifying the Integration of Wireless Sensor Networks into Business Processes, *IEEE Transactions on Software Engineering* **PP**(99) (2018), 1–1.

[5] X. Li, H. Jianmin, B. Hou and P. Zhang, Exploring The Innovation Modes and Evolution of the Cloud-Based Service Using the Activity Theory on the Basis of Big Data, *Cluster Computing* **21**(1) (2018), 907–922.

[6] J. Wang, J. Wen, C. Li, et al., DC-Vegas: A delay-based TCP congestion control algorithm for datacenter applications, *Journal of Network and Computer Applications* **53**(jul.) (2015), 103–114.

[7] S. Qu, L. Zhao and Z. Xiong, Cross-layer congestion control of wireless sensor networks based on fuzzy sliding mode control, *Neural Computing and Applications*. (2020).

[8] W. Cerroni, M. Gharbaoui, B. Martini, et al., Cross-layer resource orchestration for cloud service delivery: A seamless SDN approach, *Computer Networks* **87**(jul.20) (2015), 16–32.

[9] S. Sun, M. Kadoch, L. Gong and B. Rong, Integrating network function virtualization with SDR and SDN for 4G/5G networks, *IEEE Network* **29**(3) (2015), 54–59.

[10] P. Zach, M. Pokorn, J. Balej, et al., Controlling Multiple Virtual Machines in Computer Classrooms, *Acta Universitatis Agriculturae Et Silviculturae Mendelianae Brunensis* **63**(2) (2015), 683–691.

[11] C. Kaliszyk and J. Urban, HOL(y)Hammer: Online ATP Service for HOL Light, *Mathematics in Computer ence* **9**(1) (2015), 5–22.

[12] K. Abas, K. Obraczka and L. Miller, Solar-powered, wireless smart camera network: An IoT solution for outdoor video monitoring, *Computer Communications* **118**(MAR.) (2018), 217–233.

[13] A.M. Eassa, M. Elhoseny, H.M. El-Bakry and A.S. Salama, NoSQL Injection Attack Detection in Web Applications Using RESTful Service, *Programming and Computer Software* **44**(6), pp. 435–444.

[14] M.A. Rehmat, M. Shahbaz, A. Raza, et al., A unified framework for automated inspection of handheld safety critical devices in production assemblies, *Future Generation Computer Systems* **88**(NOV.) (2018), 342–356.

[15] S. Steven, S. Chen-Kai, T. Yu-Chin, et al., The Design and Implementation of a Novel Open Source Massive Deployment System, *Applied Sciences* **8**(6) (2018), 965.

[16] B. Ravandi and I. Papapanagiotou, A self-organized resource provisioning for cloud block storage, *Future Generation Computer Systems* **89**(DEC.) (2018), 765–776.

[17] S. Burnett and N. Feamster, Encore: Lightweight Measurement of Web Censorship with Cross-Origin Requests, *Acm Sigcomm Computer Communication Review* **45**(4) (2015), 653–667.

[18] L. Wu, Y. Zhang, K.K.R. Choo, et al., Efficient and secure identity-based encryption scheme with equality test in cloud computing, *Future Generation Computer Systems* **73**(Aug.) (2017), 22–31.

[19] E.D. Doncker and F. Yuasa, Workshop on Large Scale Computational Physics – LSCP, *Procedia Computer ence* **80**(1) (2016), 1416–1417.

[20] B. Qureshi, Profile-based power-aware workflow scheduling framework for energy-efficient data centers, *Future Generation Computer Systems* **94**(MAY) (2019), 453–467.

[21] B.B. Mishra and S. Mishra, Performance Evaluation of Virtual Machines Along With Security Mechanisms, *Journal of Environmental ence, Computer ence and Engineering & Technology* **5**(4) (2016), 347–352.

[22] M. Amoozegar and H. Nezamabadi-Pour, A multi-objective approach to model-driven performance bottlenecks mitigation, *entia Iranica* **22**(3) (2015), 1018–1030.

[23] I. Gorton and J. Klein, Distribution, Data, Deployment: Software Architecture Convergence in Big Data Systems, *IEEE Software* **32**(3) (2015), 78–85.

[24] F. Al-Hawari, A. Alufeishat, M. Alshawabkeh, et al., The software engineering of a three-tier web-based student information system (MyGJU), *Computer Applications in Engineering Education* **25**(2) (2017), 242–263.

[25] Y. Jararweh, M. Alsmirat, M. Al-Ayyoub, et al., Software-Defined System Support for Enabling Ubiquitous Mobile Edge Computing, *The Computer Journal* **60**(10) (2017), 1443–1457.

[26] E. Laxmi Lydia, J. Samuel Raj, R. Pandi Selvam, M. Elhoseny and K. Shankar, Application of discrete transforms with selective coefficients for blind image watermarking, *Transactions on Emerging Telecommunications Technologies*, 2019, In press.

[27] X. Fu and Y. Yang, Modeling and analysis of cascading node-link failures in multi-sink wireless sensor networks, *Reliability engineering & system safety* **197** (2020), 106815.

[28] M. Kuhrmann, R. Hebig, R.V. O'Connor, et al., Summary of the International Conference on Software and System Processes (ICSSP 2018), *ACM SIGSOFT Software Engineering Notes* **43**(4) (2019), 48–51.

[29] D. Kim, Y.H. Kim, C. Park, et al., KREONET-S: Software-defined wide area network design and deployment on KREONET, *IAENG International Journal of Computer ence* **45**(1pt.1-117) (2018), 27–33.

[30] Y. Niu, Y. Li, D. Jin, et al., A Survey of Millimeter Wave (mmWave) Communications for 5G: Opportunities and Challenges, *Wireless Networks* **21**(8) (2015), 1–20.

[31] I.F. Akyildiz, P. Wang and S.C. Lin, SoftAir: A software defined networking architecture for 5G wireless systems, *Computer Networks* **85**(jul.5) (2015), 1–18.

[32] M. Faschang, S. Cejka, M. Stefan, et al., Provisioning, deployment, and operation of smart grid applications on substation level, *Computer Science Research & Development* **32**(1-2) (2017), 117–130.

[33] W. Li and H. Song, ART: An Attack-Resistant Trust Management Scheme for Securing Vehicular Ad Hoc Networks, in *IEEE Transactions on Intelligent Transportation Systems* **17**(4) (2016), 960–969.

[34] B. Cao, J. Zhao, P. Yang, P. Yang, X. Liu and J. Qi, Andrew Simpson, Mohamed Elhoseny, Irfan Mehmood, Khan Muhammad, Multiobjective feature selection for microarray data via distributed parallel algorithms, *Future Generation Computer Systems* **100** (2019), pp. 952–981.

[35] L. Yin, W. Pan, J. Kuang and M. Zhuang, Application of Bootstrap-DEA with Fuzzy Computing in Performance Evaluation of Forklift Leasing Supplier, in *IEEE Access*. (2019).