

Lecture Notes in Electrical Engineering 376

Kuinam J. Kim
Nikolai Joukov
Editors

Information Science and Applications (ICISA) 2016

 Springer

Contents

Ubiquitous Computing

Jamming-Resilient Adaptive Network Protocol in Wireless Networks . . .	3
Jae-Joon Lee, Jinsuk Kang and Soo Suk Lee	

A Framework on Cloud Based Connected Car Services	11
Sumendra Yogarayan, Afizan Azman, Kirbana Jai Raman, Hesham Ali Alsayed Elbendary, Mohd Fikri Azli Abdullah and Siti Zainab Ibrahim	

Possibilities of Control and Optimization of Traffic at Crossroads Using Petri Nets	21
Jakub Gaj, Martin Kotyrba and Eva Volna	

Delay-Energy Aware Clustering Multi-hop Routing in Wireless Sensor Networks	31
Trong-Thua Huynh, Cong-Hung Tran and Anh-Vu Dinh-Duc	

Networks and Information Systems

The Application of IC Ticketing System in Clinic Fees Payment in Taiwan	43
Yi-Horng Lai	

User Acceptance of SaaS ERP Considering Perceived Risk, System Performance and Cost.	53
Tong-Ming Lim, Angela Siew-Hoong Lee and Mun-Keong Yap	

A Content Analysis on Top Risks in Social Networking	65
Sakgasit Ramingwong and Lachana Ramingwong	

Simulation of Scheduled Traffic for the IEEE 802.1 Time Sensitive Networking.	75
Chulsun Park, Juho Lee, Tianye Tan and Sungkown Park	

Adopting Multi-radio Channel Approach in TCP Congestion Control Mechanisms to Mitigate Starvation in Wireless Mesh Networks	85
Balamuralikrishna Potti, M.V. Subramanyam and K. Satya Prasad	
An Information System to Support the Anti-doping Process	97
Francisco Medeiros, Juliana Medeiros, Fausto Ayres, Caio Viana, Josemary Rocha, Victor Viegas, Eder Mendes and Ana Santos	
Research on 3D Directional Sensing Model	109
Yanjiao Wang, Dongxian Yu and Shuqiang Guo	
Live Migration Destination Selecting Method Using Weather Information and Emergency Alerts for e-Learning Environment	115
Satoshi Togawa and Kazuhide Kanenishi	
OPNET-Based Performance Analysis of a Multi-agent Architecture for Managing the Mobile Content Delivery Process	127
Adel Aneiba and Claude C. Chibelushi	
Data Prefetching Scheme Based on Data Relevancy for Peer-to-Peer Networks	139
Xinxin Zhou, Renjie Song, Peng Kong, Yanjiao Wang and Shuqiang Guo	
Sustainability of Big Data Servers Under Rapid Changes of Technology	149
Sashiraj Chandrasekaran and Insu Song	
An Adaptive, Fault Protected Improvisation of ACO Based MANET Routing.	161
T. Pranav Bhat, Gargi Saha, V. Abhishek and Swapn Bhattacharya	
Recipe-Transition Graph Based on Asymmetric Entropy Difference . . .	171
Jinkwan Park, Su-Do Kim, Yun-Jung Lee and Hwan-Gue Cho	
The Significance of e-CALLISTO System and Construction of Log Periodic Dipole Antenna (LPDA) and CALLISTO System for Solar Radio Burst Study	181
Siti Nur Umairah Sabri, Zety Sharizat Hamidi, Nur Nafhatun Md Shariff and C. Monstein	
WARP: Web-Based Adaptive Remote-Desktop Protocol for VDI.	189
Jahun Ku, Myeong-Seob Kim, Jinseop Lee, Pham Xuan Qui and Eui-Nam Huh	
An Automated System for Signal Detection of Solar Radio Burst Type II Due to Coronal Mass Ejections Phenomena	195
Nur Hidayah Zainol, Zety Sharizat Hamidi, Nur Nafhatun Mohd Shariff and C. Monstein	

Optimization Downlink Power Utilization via Dual Beam Antenna in the Live UMTS Network	207
Thananan Numatti and Saiyan Saiyod	
Design and Implementation of Drone for Wideband Communication and Long-range in Maritime	219
Dong Hyun Kim, Jun Hwan Huh and Jong-Deok Kim	
Communications-Based Technology for Smart Grid Test Bed Using OPNET Simulations	227
Jun-Ho Huh, Seung-Mo Je and Kyungryong Seo	
Factors Influencing Cloud Enterprise Resource Planning Adoption in SMEs	235
Usman Musa Zakari Usman, Muhammad Nazir Ahmad and Nor Hidayati Zakariya	
Scaling Distributed All-Pairs Algorithms: Manage Computation and Limit Data Replication with Quorums	247
Cory J. Kleinheksel and Arun K. Somani	
Dynamic TDMA for Networked Embedded Systems	259
Mukesh K. Chippa, Mohamed A. Elamin, Shivakumar Sastry and Nghi. H. Tran	
Driver Fatigue Detection	269
Muhamad Hafiz Abdullah, Kirbana Jai Raman, Afizan Azman, Sumendra Yogarayan, Hesham Ali Alsayed Elbendary, Mohd Fikri Azli Abdullah and Siti Zainab Ibrahim	
Cooperative Caching Strategies for Mobile Peer-to-Peer Networks: A Survey	279
Xinxin Zhou, Zhenwan Zou, Renjie Song, Yanjiao Wang and Zhenwei Yu	
Analysis of Chemical Plant Process Safety Information Integration Model from Information System Perspective	289
Yew Kwang Hooi, M. Fadzil Hassan, Azmi M. Shariff, Hanida Abd Aziz, Najah M. Jamal and Sujendran Suppiah	
Cooperative Robust Output Regulation for Linear Uncertain Time-Delay Multi-Agent Systems	299
Maobin Lu and Jie Huang	
The International Cooperation of Solar Radio Burst Project Using e-Callisto System Network	309
Zety Sharizat Hamidi, Nur Nafhatun Md Shariff, C. Monstein, A.B.M.A. Ibrahim and M.I.M. Yusof	

A Performance Study of Hidden Markov Model and Random Forest in Internet Traffic Classification.	319
Alhamza Munther, Rozmie R. Othman, Ali Shanon Alsaadi and Mohammed Anbar	
Design and Configuration of Avoidance Technique for Worst Situation in Zigbee Communications Using OPNET	331
Jun-Ho Huh, Seung-Mo Je and Kyungryong Seo	
Multimedia and Visualization	
Automated Segmentation of Microtubules in Cryo-EM Images with Excessive White Noise	339
Guiyang Yue, Linhua Jiang, Cong Liu, Guisong Yang, Jun Ai and Xiaodong Chen	
The SI-LBP: A New Framework for Obtaining 3D Local Binary Patterns from Shape-Index	349
Suranjan Ganguly, Debotosh Bhattacharjee and Mita Nasipuri	
Implementation of Autonomous Unmanned Aerial Vehicle with Moving-Object Detection and Face Recognition	361
Sanjaa Bold, Batchimeg Sosorbaram and Seong Ro Lee	
Robust Visual Communication System in Mobile Environments	371
Byung-Hun Oh and Kwang-Seok Hong	
Medical Image Segmentation Using Improved Chan-Vese Model.	379
Shuqiang Guo, Xuenan Shi, Yanjiao Wang, Xinxin Zhou and Yasukata Tamura	
Study of Point Spread Function of Astronomical Object Imaging	391
Jian Yu, Qian Yin and Ping Guo	
Big Data Visualization: Application in Visualizing Learning Activities	399
Nguyen Thanh Tam and Insu Song	
A Comparative Study on Hough Transform Segmentation Approach for Outlier Iris Image	409
Mustafa Man, Mohamad Faizal Ab Jabal and Mohd Shafry Mohd Rahim	
Exploring Multi-feature Based Action Recognition Using Multi-dimensional Dynamic Time Warping.	421
Yang Liu, Guoliang Lu and Peng Yan	
A Dropout Distribution Model on Deep Transfer Learning Networks . . .	431
Fengqi Li, Helin Yang and Junpeng Wang	

Convolutional Neural Network Models for Facial Expression Recognition Using BU-3DFE Database	441
Xuan-Phung Huynh, Tien-Duc Tran and Yong-Guk Kim	
Non-Rigid Object Tracking Using Modified Mean-Shift Method	451
Shuqiang Guo, Xuenan Shi, Yanjiao Wang and Xinxin Zhou	
Extended Gaussian Mixture Model Enhanced by Hole Filling Algorithm (GMMHF) Utilize GPU Acceleration	459
Adi Nurhadiyatna, Rini Wijayanti and Driszal Fryantoni	
AVM Auto-Calibration for Driving Vehicle Images	471
Jiwon Bang, Junghwan Pyo and Yongjin Jeong	
An Object-Oriented Programming Interface for Low-Tier Text-Based Windowing Systems	481
Jonghyuk Park and Nakhoon Baek	
Middleware and Operating Systems	
Scheduling Algorithms for Cache Effect Optimisation in Partitioned Systems	491
Vicent Brocal, Yolanda Valiente, Patricia Balbastre, Alfons Crespo and Pedro Albertos	
HYFLUR: Recovery for Power-off Failure in Flash Memory Storage Systems Using HYbrid FLush Recovery	501
Ji-Hwan Chung and Tae-Sun Chung	
Static Scheduling Generation for Multicore Partitioned Systems	511
Alfons Crespo, Patricia Balbastre, Jose Simo and Pedro Albertos	
Improve Dynamic Sandbox on the Cloud with Non-QEMU Based OS Through Hooks and Mocks Techniques	523
Ngoc-Tu Chau, Long Nguyen Vu and Souhwan Jung	
Exploring the Platform for Expressing SystemVerilog Assertions in Model Based System Engineering	533
Muhammad Rashid, Muhammad Waseem Anwar, Farooque Azam and Muhammad Kashif	
Enhanced Cloud Data Placement Strategy for Multiple Cloud Storage Services on Mobile Devices	545
Christofer Theodore and Hyotaek Lim	
Over the Air Programming Method for Learning Wireless Sensor Networks	555
K. Sangeeth, Preeja Pradeep, P. Rekha, P. Divya, R.D. Aryadevi and Maneesha Sudheer	

Design of Multimedia File Similarity Evaluation Scheme Using Fingerprinting	567
Min Ja Kim, Sung Bong Jang and Young Woong Ko	
Security and Privacy	
Privacy Preservation Based on Full-Domain Generalization for Incremental Data Publishing	577
Torsak Soontornphand, Nattapon Harnsamut and Juggapong Natwichai	
Creating Snort-IDS Rules for Detection Behavior Using Multi-sensors in Private Cloud	589
Khamkone Sengaphay, Saiyan Saiyod and Nunnapus Benjamas	
Review of Security Vulnerabilities in the IPv6 Neighbor Discovery Protocol	603
Mohammed Anbar, Rosni Abdullah, Redhwan M.A. Saad, Esraa Alomari and Samer Alsaleem	
A New Probabilistic Digital Signature Scheme Based on Integer Factorization Problem	613
Shailendra Kumar Tripathi and Bhupendra Gupta	
Android Mobile Application for Privacy Data Access in MANET Emergency Services.	623
Asmidar Abu Bakar and ElMunzir Hassan Eisa El-Talib	
Attack on a Nonlinear Inter-Pixel Computing and Swapping Based Permutation Approach	633
Bin Lu, Xin Ge, Daofu Gong and Fenlin Liu	
The Capacity of Undetectable On/Off Covert Channel	641
Anna Epishkina, Mikhail Finoshin and Konstantin Kogos	
Quality Analysis of Logging System Components in the Cloud	651
Winai Wongthai and Aad van Moorsel	
Copy-Move Forgery Detection Using on Locality Sensitive Hashing and k-means Clustering.	663
Osamah M. Al-Qershi and Bee Ee Khoo	
Development of Android Security Permission Application	673
Jongmun Jeong, Hoon Lee and Mintae Hwang	
A Complete Fingerprint Matching Algorithm on GPU for a Large Scale Identification System	679
Hong Hai Le, Ngoc Hoa Nguyen and Tri Thanh Nguyen	
Android Security Analysis Based on Inter-application Relationships . . .	689
Nguyen Tan Cam, Pham Van Hau and Tuan Nguyen	

An Analysis of IT Assessment Security Maturity in Higher Education Institution.	701
Misni Harjo Suwito, Shinchu Matsumoto, Junpei Kawamoto, Dieter Gollmann and Kouichi Sakurai	
Detection of SPAM Attacks in the Remote Triggered WSN Experiments	715
Sangeeth kumar, Preeja Pradeep and Sumesh Kj	
Secret Outflow Cause Symptom Analysis Through Scenario Assessment	729
DongHwi Lee, Woo-Bin Park, Yun-Hee Kim, Kyong-Ho Choi and Seung-Ho Kang	
Synflood Spoof Source DDoS Attack Defence Based on Packet ID Anomaly Detection - PIDAD.	739
Tran Manh Thanh and Van Khanh Nguyen	
Range Image Derivatives for GRCM on 2.5D Face Recognition	753
Lee-Ying Chong, Andrew Beng Jin Teoh and Thian-Song Ong	
Improving Intrusion Detection on Snort Rules for Botnets Detection	765
Youksamay Chanthakoummene, Saiyan Saiyod, Nunnapus Benjamas and Nattawat Khamphakdee	
Data Mining and Artificial Intelligence	
Development of Power Quality Index Using Ideal Analytic Hierarchy Process.	783
Buhm Lee, Dohee Sohn and Kyoung Min Kim	
An Improved Algorithm for Calculating Flow Paths of Injection-Production in Single Sand Body in Old Oilfields Under the Background of Big Data	795
Jiqun Zhang, Baorong Deng, Xinhao Li, Junhua Chang, Hua Li and Dongmei He	
Modeling Promotion Factors Using Bayesian Networks and Video Games	805
Manolis Maragoudakis, Katia Kermanidis and Spyros Vosinakis	
A Computational Agent Model for Stress Reaction in Natural Disaster Victims	817
Hayder Mohammed, Azizi Ab Aziz, Noraziah ChePa, Ahmad Hanis Mohd Shabli, Juliana Aida Abu Bakar and Asmidah Alwi	

Kalman Filter-Based Aggressive Behaviour Detection for Indoor Environment	829
Mohd Asyraf Zulkifley, Nur Syazwani Samanu, Nik Ahmad Akram Nik Zulkepli, Zulaikha Kadim and Hon Hock Woon	
Effectiveness Comparison of Range Estimator for Battery Electric Vehicles	839
K.W. Chew and Y.R. Yong	
Design of Maritime Meteorological Information Data Model Based on S-100	851
Daseul Oh, Daewon Park and Suhyun Park	
Administrative Divisions of Addresses Matching Algorithm Based on Moving Window Algorithm for Maximal Matching	861
Xiaolin Li, Shuang Huang and Tao Lu	
Application of Fuzzy C-Mean Clustering Base Tree for Measuring the Effectiveness of Corporate	873
Jirawat Teyakome, Narissara Eiamkanitchat, Komsan Suriya and Phichit Napook	
Extending the Socio-Economic Status (SES) Prediction System Based on the Thailand Marketing Research Society (TMRS) Standardized SES Classification for Thai Upcountry Urban Subjects	885
Jirayut Poomontre and Pisal Setthawong	
Intelligent and Adaptive Test System	895
Robin Tommy, Nancy Amala, Shantha Ram, Bala Kumar and Hima Jose	
A New Extraction Algorithm for Hierarchical Keyword Using Text Social Network	903
Da-Young Lee, Kyung-Rim Kim and Hwan-Gue Cho	
Layer-Wise Relevance Propagation for Deep Neural Network Architectures	913
Alexander Binder, Sebastian Bach, Gregoire Montavon, Klaus-Robert Müller and Wojciech Samek	
Building a Hybrid Method for Analyzing the Risk by Integrating the Fuzzy Logic and the Improved Fuzzy Clustering Algorithm FCM-R	923
Huan Doan and Dinh Thuan Nguyen	
Meta Learning on Small Biomedical Datasets	933
Turgay Ibrikci, Esra Mahsereci Karabulut and Jean Dieu Uwisengeyimana	

Challenges in the Data Collection for Diagnostics of Smart Buildings . . .	941
Sanja Lazarova-Molnar and Nader Mohamed	
A Case Based Reasoning View of School Dropout Screening	953
José Neves, Margarida Figueiredo, Lídia Vicente and Henrique Vicente	
A Modified Shuffled Frog Leaping Algorithm Using Truncated Gaussian Distribution in Frog's Position Updating Process	965
Kanchana Daoden and Trasapong Thaiupthump	
A Surveillance System Using CNN for Face Recognition with Object, Human and Face Detection	975
Yeong-Hyeon Byeon, Sung-Bum Pan, Sang-Man Moh and Keun-Chang Kwak	
Automated Disease Outbreak Detection and Analysis	985
Kaleab Getaneh Tefrie and Kyung-Ah Sohn	
Hierarchical Time Series Forecast in Electrical Grids	995
Vânia Almeida, Rita Ribeiro and João Gama	
Data Pre-processing of Student e-Learning Logs.	1007
Mingming Zhou	
Generating Point of Interest Description with Geo-tagged Web Photos	1013
Thanh-Hieu Bui, A-Yeong Kim, Seong-Bae Park and Sang-Jo Lee	
Association Rule Discovery for Rosewood Crime Arrest Planning	1025
Rit Lawpanom and Wararat Songpan	
Software Engineering	
Translating Basic Metric Temporal Logic Formulas into Promela	1035
Punwess Sukvanich, Arthit Thongtak and Wiwat Vatanawood	
A Propose Model for Shared Understanding of Software Requirements (SUSRs)	1045
Issam Jebreen, Mohammed Awad and Ahmad Al-Qerem	
Mathematical Concept of Programming in Graphs	1057
Igor Velbitskiy	
Islamic New Moon Software: Current Status	1069
Nur Nafhatun MD Shariff, Muhamad Syazwan Faid and Zety Sharizat Hamidi	
A Study on Decoupling Flow Engines Toward Cross-Platform Interoperability.	1081
Chuan-Fa Wang and Don-Lin Yang	

Information System Design Based on the Result of Organization Goal-Oriented Requirements Engineering Process	1093
Fransiskus Adikara, Patricia Dianita Wijaya, Bayu Hendradjaya and Benhard Sitohang	
Estimation of Fault Reparation Time in Software Maintenance Phase	1105
Pawin Youpoung and Pornsiri Muenchaisri	
Generation of Java Code from UML Sequence and Class Diagrams . . .	1117
Preyanoot Kluisritrakul and Yachai Limpiyakorn	
Visualization of Gaps Between Sheet Parts for Watertightness of Automobiles	1127
Jianhui Fu, Byeongju An, Raesung Park and Yoongho Jung	
Conceptual Design Model of iCAL4LA: A Review and Critical Analysis of Computer Assisted Learning Concept in Determining the Core Segments and their Components.	1133
Siti Zulaiha Ahmad and Ariffin Abdul Mutalib	
Web Technology	
Sentiment Analysis and User Similarity for Social Recommender System: An Experimental Study	1147
Thi-Ngan Pham, Thi-Hong Vuong, Thi-Hoai Thai, Mai-Vu Tran and Quang-Thuy Ha	
Adaptive Polling for Responsive Web Applications	1157
Hatem Aziz and Mick Ridley	
A Framework of Incorporating Thai Social Networking Data in Online Marketing Survey	1169
Rachsuda Jiamthapthaksin, Than Htike Aung and Nitipan Ratanasawadwat	
Recommender Systems: Issues, Challenges, and Research Opportunities	1179
Shah Khusro, Zafar Ali and Irfan Ullah	
Preserving Dynamic XML Keys.	1191
Norfaradilla Wahid and Hairulnizam Mahdin	
Ontology Based Adaptive, Semantic E-Learning Framework (OASEF)	1199
Sohail Sarwar, Zia Ul Qayyum, Muhammad Safyan and Rana Faisal Munir	
Planning in Dynamic, Distributed and Non-automatized Production Systems.	1213
Matthias Becker, Michael Lütjen and Helena Szczerbicka	

Decision Support for the Stocks Trading Using MLP and Data Mining Techniques	1223
Narissara Eiamkanitchat and Teerasak Moontui	
Internet of Things	
A Survey of Security Aspects for Internet of Things in Healthcare	1237
Suhardi and Alfian Ramadhan	
Influence of Intentional Electromagnetic Interference on the Functioning of the Terrestrial Segment of Flying Ubiquitous Sensor Network	1249
Trung Hoang, Ruslan Kirichek, Alexander Paramonov and Andrey Koucheryavy	
Analysis of Routes in the Network Based on a Swarm of UAVs	1261
Nghia Dao, Andrey Koucheryavy and Alexander Paramonov	
Extension of OneM2M Protocol Binding Considering Service Mapping.	1273
Jun Hwan Huh, Dong Hyun Kim and Jong-Deok Kim	
Enterprise Modelling for the Internet of Things: The ComVantage Method	1283
Robert Andrei Buchmann and Dimitris Karagiannis	
A Conceptual Framework for the Design of an Urban Flood Early-Warning System Using a Context-Awareness Approach in Internet-of-Things Platform.	1295
Jeerana Noymanee, Wimol San-Um and Thanaruk Theeramunkong	
A Study on the IoT Based Smart Door Lock System	1307
Jeong-ile Jeong	
ICT Convergence	
Optimized Data Transmission Method for Multimedia Services Based on OpenStack	1321
Sanghyun Park, Gemoh Maliva Tihfon, Seonhyeok Lee and Jinsul Kim	
Load Balancing Methodology for Efficient Real-Time Streaming Service Based on WebRTC	1331
Jisue Kim, Linh Ma Van and Jinsul Kim	
A New Virtualized Environment for Application Deployment Based on Docker and AWS	1339
Gemoh Maliva Tihfon, Jinsul Kim and Kuinam J. Kim	

Implementation of Chinese Cabbage of Origin Discrimination Software	1351
Jong-In Kim, Jae-Won Lee, Hyo-Gi Seo, Jung-Su Yang and Sung-Hoon Hong	
The Smart Armband: Expanding Wearable Interface Area and Suggesting Interaction Scenarios	1361
Hyunyoung Kim, Sanghyun Park, Nuri Na, Jisue Kim, Yongpil Moon and Jinsul Kim	
Cost-Effective Multicast Routing with Multi-gateway in Wireless Mesh Networks	1367
Younho Jung, Seonhyuk Lee, Su-il Choi, Cheol Hong Kim, Jinsul Kim, Kyungran Kang and Jaehyung Park	
Frequency Features Selection Using Decision Tree for Classification of Sleep Breathing Sound	1375
Tan Loc Nguyen and Yonggwon Won	
Architecture Design with Coordinating Live Streaming and VOD Sharing for Media Content Distribution	1381
Tran Thi Thu Ha and Jinsul Kim	
CTA-Aware Dynamic Scheduling Scheme for Streaming Multiprocessors in High-Performance GPUs.	1391
Dong Oh Son, Cong Thuan Do, Hong Jun Choi, Jong Myon Kim, Jaehyung Park and Cheol Hong Kim	
Design of Adaptive Integrated Fast Image Enhancement System for General, Haze, Low Light, Back-Light Condition	1401
Jae-Won Lee, Jong-In Kim, Bae-Ho Lee and Sung-Hoon Hong	
A New Prefetch Policy for Data Filter Cache in Energy-Aware Embedded Systems	1409
Dong Oh Son, Cong Thuan Do, Hong Jun Choi, Jong Myon Kim, Jiseung Nam and Cheol Hong Kim	
User-Assisted OCR on Outdoor Images for Approximate Positioning . . .	1419
Taeyu Im and Pradipta De	
Convergence of Healthcare and Information Technology	
The Application of Digital Health Content Using Mobile Device	1433
SooJung Park, BongSup Park, SeungAe Kang and SunYoung Kang	
Review Paper on Nutritional Information Using Mobile Augmented Reality Technology	1439
Saad Masood Butt and Karla Felix Navarro	

The Applications of Convergence Technology for Physical Activity in the Disabled	1447
Seungae Ae Kang	
Personalized Exercise Prescription Utilizing Virtual Fitness.	1453
Sun Young Kang, Kyung OcK Yi, Seung Ae Kang and Ae Hyun Lee	
Network Function Virtualization (NFV) Platform for Wellness in High-Speed Network	1459
Hyun Cheol Kim	
Review of the Role of Modern Computational Technologies in the Detection of Dyslexia	1465
Harshani Perera, Mohd Fairuz Shiratuddin and Kok Wai Wong	
A New Paradigm for the Spread Sport Leisure Culture Focusing on the IT-Based Convergence Interactive System	1477
Minkyu Kim, Soojung Park, Byoungkwon Park, YoungHee Cho and SunYoung Kang	
Transmission Optimization of Healthcare Information for Multi-channel Cloud Networks	1487
HyunCheol Kim, Wonhyuk Lee and Seungae Kang	
The ICT Technology for Geriatric Diseases Healthcare.	1495
SunYoung Kang	
Author Index	1501

A New Virtualized Environment for Application Deployment Based on Docker and AWS

Gemoh Maliva Tihfon, Jinsul Kim and Kuinam J. Kim

Abstract The setup environment and deployment of distributed applications is a human intensive and highly complex process that poses significant challenges. Nowadays many applications are developed in the cloud and existing applications are migrated to the cloud because of the promising advantages of cloud computing. The very core of cloud computing is virtualization. In this paper, we will look at application deployment with Docker. Docker is a lightweight containerization technology that has gained widespread popularity in recent years. It uses a host of the Linux kernel's features such as namespaces and cgroup's to sandbox processes into configurable virtual environments. Presenting two common serious challenging scenarios in the software development environment, we propose a multi-task PaaS cloud infrastructure using Docker and AWS services for application isolation/optimization and rapid deployment of distributed applications.

1 Introduction

Software deployment is complex and the diverse computing requirements for applications require complex hardware infrastructure setups and possibly incompatible specific software requirements. Therefore, a platform to automate the deployment and setup of virtual computing is essential. Moreover, it is important to properly and efficiently manage the computing resources so as to reduce additional investment in hardware. All these lead towards the concept of cloud computing.

G. M. Tihfon · J. Kim(✉)

School of Electronics and Computer Engineering, Chonnam National University,
Gwangju 500-757, Korea
e-mail: gemohmal@gmail.com, jsworld@chonnam.ac.kr

K.J. Kim(✉)

Convergence Security Department, Kyonggi University, Suwon, Gyonggi-Do, Korea
e-mail: Kuinamj@gmail.com

© Springer Science+Business Media Singapore 2016

1339

K.J. Kim and N. Joukov (eds.), *Information Science and Applications (ICISA) 2016*,
Lecture Notes in Electrical Engineering 376,
DOI: 10.1007/978-981-10-0557-2_126

Cloud computing is a paradigm to rapidly provision computing resources such as storage, networks, servers, services, etc., that can be customized and configured to suit a particular user or application demands[1]. Cloud computing paradigm is promising because it is changing the way enterprises do their businesses in that dynamically scalable and virtualized resources are provided as a service over the internet. The cloud is enabled by virtualization, automation, standardization. The very core of cloud computing is virtualization, which is used to separate a single physical machine into multiple virtual machines in a cost-effective way. By using virtualization, we're basically getting a lot of the work done for free. With virtualization, a number of virtual machines can run on the same physical computer, which makes it cost-effective, since part of the physical server resources can also be leased to other tenants [2] [4]. Such virtual machines are also highly portable, since they can be moved from one physical server to the other in a manner of seconds and without downtime; new virtual machines can also be easily created. Another benefit of using virtualization is the location of virtual machines in a data center. It does not matter where the data center is located and the virtual machine can also be copied between the data centers with ease [3]. VM's in the cloud offer rapid elasticity and it is pay as you go model.

One thing to keep in mind before pushing forward is that, it's all about the applications and not the operating system. We need operation system to facilitate the applications. Virtual machines (VM) reduce capex and opex but also have some limitations that can be address. VM still requires a CPU, storage allocation, RAM, and an entire guest Operating system (OS). OS consume a lot CPU, RAM, Disk storage, and increase overhead. The more VM's you run, the more resources you need. Also some operating systems might need individual licensing. Moreover application portability is not guaranteed. The VM module does nothing to help but with docker we don't have to worry about all the issues mentioned above.

IaaS cloud computing is hugely influence by hypervisor virtualization [6]. Lightweight virtualization technologies such as Docker, LXC, and Open VZ etc, seems to be a good fit for the cloud although lightweight virtualization is limited but they provide a better hosting density. In general it is possible to host more lightweight virtualizations on a physical host than with hypervisor based virtualizations [5]. Docker can be deployed to any environment or device being public or private cloud because it is super lightweight. A docker container does not include the full OS as mention earlier, but shares the OS of its host. As a result, Docker containers can be faster and less resource-draining than virtual machines. A full virtual machine can take a couple of minutes to launch because of boot time and other stuff, however a container can be initiated in a blink of an eye (seconds). Containers also offer superior performance for the application they contain, compared to running the application within a virtual machine.

2 Problem Statement and Description

Scenario 1: Microservice architecture is an approach that makes web based development more agile and code bases easier to maintain. This architecture enables developers to be highly productive and to quickly iterate and evolve a code base. For fast moving startup companies, the microservices architecture can

really help dev teams be quick and agile in their development efforts. The disadvantage of microservices is that, because services are spread across multiple hosts, it is difficult to keep track of which hosts are running certain services.

Linux containers can help mitigate many of these challenges with the microservices architecture. Linux containers make use of kernel interfaces such as cgroups, namespaces, and unionfiles, which allow multiple containers to share the same kernel while running in complete isolation from one another. The Docker execution environment uses a module called Libcontainer, which standardizes these interfaces. It is this isolation between containers running on the same host that makes deploying microservices code developed using different languages and frameworks very easy. Using Docker, we could create a DockerFile describing all the languages, framework, and library dependencies for that service. The container execution environment isolates each container running on the host from one another, so there is no risk that the language, framework, library dependencies used by one certain container will collide with that of another.

Scenario 2: You have written a code for some website or developed a mobile app for a game using development environment on your laptop. After thorough testing and realize that your code is ready to be deploy in the working environment or in your working organization. The system admin dutifully deploys the most recent build to the test environment and in no time notice that your recently developed REST endpoint is broken. After uncountable hours of troubleshooting with the system admin, you discover that the test environment is using an outdated version of third-party library, and this was causing the REST endpoints to break. Differences between developments, test, stage, and production environments is a familiar problem in today's rapid build and deploy cycles.

The solution is to find a way to transfer from one environment to another seamlessly and eliminating error prone resource provisioning and configuration. Services like Amazon EC2, AWS CloudFormation, and Docker provide reliable and efficient way to automate the creation of an environment. Amazon EC2 makes web-scale cloud computing easier for developers. AWS CloudFormation gives developers and system admins an easy way to create and manage a collection of related AWS resources, provisioning and updating them in an orderly and predictable manner. You simply create or use prebuilt template which is a JSON file that serves as a blueprint to define the configuration of all the AWS resources that make up your structure and application stack. On a plus, CloudFormation is free of charge and you pay only for the AWS resources needed to run your application. Docker takes the concept of declarative provisioning a step further. Docker provides a declarative syntax for creating containers. However, Docker containers don't depend on any specific virtualization platform; neither do they need a separate operating system to run. A container simply requires a Linux kernel in order to run. This means dockerized apps can run anywhere on anything being desktop, laptops, VMs, datacenter or instances in the cloud. Docker containers use an execution environment called Libcontainer, which is an interface to various Linux kernel

isolation features, like cgroups, namespace, and unionfiles. This architecture allows for multiple containers to be run in complete isolation from one another while sharing the same Linux kernel. Because a Docker container instance doesn't require a dedicated OS, it is much more portable and lightweight than a virtual machine. The core components of Docker are the Docker daemon and Docker client. Docker daemon is the engine that runs on the host machine and it is a server process that manages all the containers. Docker client is a CLI used to interact with the daemon. The key concepts to understand the workflow of Docker as shown in Figure 1 are its workflow components. Docker images, registry, containers, and Dockerfile.

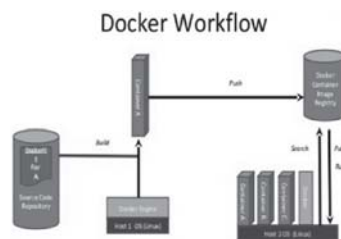


Fig. 1 Docker workflow diagram

Docker containers are becoming the go ahead for all distributed systems because they are scalable in the sense that these containers are extremely lightweight which make scaling up and scaling down very fast and easy. Dockerized applications are extremely portable; we can move them very easy. With the isolated containers, we can put more than one into a machine thereby making more efficient use of our resources. Another huge plus point of Docker is the Docker community. This community is one of the fastest growing open source communities out there. Chef, Puppet, Cloud providers such as AWS, OpenStack Azure, and Rackspace are just a few of the recognized members. There are many more benefits, but what all this mean is that it dramatically reduces the entire development life cycle from development, to testing, and then deployment.

3 Related Works

There are many works in software management and tools that address the deployment of virtual infrastructures and applications. Numerous cloud providers, such as AWS provide tools to deploy virtual infrastructures, applications and websites. In particular, CloudFormation and OpsWorks provides developers and systems administrators with an easy way to create and manage a collection of related AWS resources, provisioning and updating them in an orderly and predictable fashion [8].

The Nimbus project team group has developed a set of tools to deploy virtual infrastructures: the Context Broker [9] and cloudinit.d [10]. In particular, the last

tool submits, controls, and monitors Cloud applications. It automates the virtual machine (VM) creation process, the contextualization, and the coordination of service deployment[7]. It supports multiple clouds and the synchronization of different ‘runlevels’ to launch services in a defined order. Furthermore, it provides a system to monitor the services that uses user-created scripts to ensure that they are running. This system checks for service errors, re-launching failed services or launching new VMs. However, it enables the contextualization of VMs using simple scripts, which are insufficient in complex scenarios with multiple VMs and different Operating Systems.

One common limitation of all the above systems is the usage of manually selected base images to launch the VMs. This is an important limitation because it implies that users must create their own images or they must previously know the details about software and configuration of the image selected. This limitation affects the reutilization of the previously created VMIs, forcing the user to waste time testing the existing images or creating new ones. Another issue is that most of them need to use a VMI specifically configured to support their environment, requiring specific software installed or a set of scripts prepared.

The next section is our proposed cloud platform to address and improve most of these related works. Also in the next section is an algorithm we created to effectively and fully utilize the available resources in any data center or organization setup environment.

4 Proposed Cloud Infrastructure

Based on the numerous advantages of Docker containers, ease of deployment in the development, test, stage, and production environment and how Docker containers fit well in the distributed systems architecture (microservices). We propose a private-multitask PaaS cloud system. This PaaS cloud system using Docker is for infrastructure virtualization and application isolation/deployment. In a multitask environment, the number of containers will be increasing, and this becomes increasingly difficult to manage manually. This is where the services of Amazon EC2 Container Service (ECS) steps in to help our container management framework (cluster computing). With ECS, we effectively abstract the low-level resources such as CPU, memory, and storage, allowing for highly efficient usage of the nodes in a compute cluster.

Initially the idea we had was to use the Docker swarm which is a native clustering solution provided by Docker. It takes the Docker Engine and extends it to enable you to work on a cluster of containers. Using Swarm we can manage a resource pool of Docker hosts and schedule containers to run transparently on top, automatically managing workload and providing failover services. Swarm uses an algorithm called Bin Packing Scheduling algorithm (they also support Random and Spread algorithms) and some scheduler filters (Constrain, Affinity, Port, Dependency, and Health filters) to effectively manage the containers on a subset of nodes. Swarm is the future native clustering for Docker. Currently swarm has

many limitations such as it doesn't support image management yet, it is still beta and not recommended for production. So using Amazon EC2 Container service is the right choice for scalability and management of Docker containers.

EC2 Container Service is a cluster management framework that uses optimistic, shared state scheduling to execute processes on EC2 instances using Docker containers. Amazon ECS makes it easy to launch containers across multiple hosts, isolate applications and users, and scale rapidly to meet changing demands of your applications and users. Using ECS incurs no extra charges, apart from the cost of spinning up EC2 instance. The ECS takes care of many of the challenges in running a distributed system. Customers need not about monitoring the health and availability of nodes that provide the scheduling and resource management capabilities. ECS also provides a robust solution to the very challenging problem of storing state information in a distributed system. Lastly, ECS is designed to scale horizontally and for high availability. Container instances, clusters, tasks, and task definition are the key components of ECS.

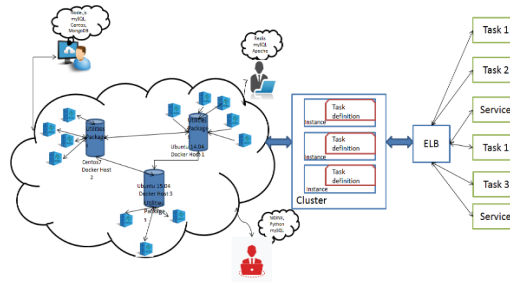


Fig. 2 Propose multi-task cloud

5 System Workflow

The propose platform allow organizations/sysadmins/developers to focus on building products rather than building infrastructure. As mention earlier, we can build, test, and debug our code on any machine capable of running Docker. When the code is ready, we can package it up into the Docker image by building the image from a Dockerfile and storing it in Docker Hub (repository). Next, we need to provide the compute resources required to run containers. In ECS, this is called a cluster, and it consists of EC2 instances called “container instances” that are running the ECS agent. To create an ECS cluster of container instances, we simply launch one or more EC2 instances using the Aazon ECS-Optimized Amazon Linux AMI. The instance will need to be associated with an IAM role that allows the agent running on the instance to make the necessary API calls to ECS. The next step is to tell ECS how to run the containers. We use an entity called a “task definition.” ECS task definition can be thought of a prototype for running an actual task. For any given task definition, there can be zero or more

task instances running in the cluster. The Task definition allows for one or more containers to be specified. ECS has another entity called a “service,” which is useful for long running tasks, like web applications. A service allows multiple instances of a task definition to be run simultaneously. It also provides integration with Elastic Load Balancing (ELB) service. The ELB is used to distribute tasks and services to different containers efficiently.

6 Schedule Algorithm

The schedule algorithm below aims to schedule applications on VMs based on the user deployment request and deploys VMs on physical resources based on resource availability. This strategy optimizes the application performance. Additionally, the load-balancer ensures high and efficient resource utilization in the cloud environment.

Schedule Algorithm for VMs

```

1: Input: UserDeploymentRequest
2: get Resources&AvailableVMList
3: // find applicableVMList
4: if AVM(UDR, ARS) != 0 then
5:   //call the load balancer
6:   deployableVM = load-balance(AVM(UDR, ARS))
7:   deploy UserRequest on deployableVM;
8:   deployed = true;
9: else
10:  if ResourceForExtraVM then
11:    start newVMInstance;
12:    add VMToApplicableVMList;
13:    deploy UserRequest on newVM;
14:    deployed = true;
15:  else
16:    queue UserRequest until
17:    queueTime > waitingTime
18:    deployed = false;
19:  end if
20: end if
21: if deployed then
22:  return success;
23:  terminate;
24: else
25:  return fail;
26:  terminate;
27: end if

```

As shown in the Schedule Algorithm, the scheduler receives as input the Users' Deployment Requests (UDR) and the application data to be provisioned (line 1 in the schedule algorithm). The output of the scheduler is the confirmation of successful or failure deployment.

In the first step, the user request is extracted, which then forms the basis for finding the VM with appropriate resources for deploying the application. Next, it collects information about the Available Resource (ARS) and the number of running VMs in the data center (line 2). The UDRs are used to find a list of Applicable/Apposite VMs (AVM) capable of provisioning the requested user request (lines 3-4).

When the list of VMs is found, the load-balancer decides on which particular VM to deploy the application in order to balance the load in the data center; in our case the ELB (line 5-8).

In case there is no VM with the appropriate resources running in the data center, the scheduler checks if there is resources consisting of physical resources can host new VMs (lines 9-10). If that is the case, it automatically starts new VMs with predefined resource capacities to provision the user requests (lines 11-14).

When the resources cannot host extra VMs, the scheduler queues the provisioning of service requests until a VM with appropriate resources is available (lines 15-16). If after a certain period of time, the user requests cannot be scheduled and deployed, the scheduler returns a scheduling failure to the cloud admin, otherwise it returns success (lines 17-27).

7 Test Experience

Based on the figure below Fig. 3, the setup environment for testing is pretty simple at this stage of our work. Using Oracle VM VirtualBox manager we setup a 64-bit Ubuntu 14.04 system and Centos7 system with the following features each: 512MB of memory, 2 processor, 12MB of video memory, 2 network adapters, and 16GB of hard drive space.

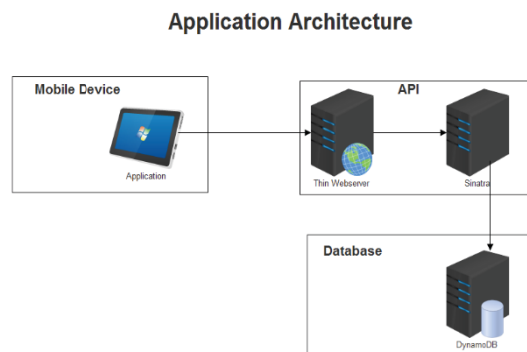
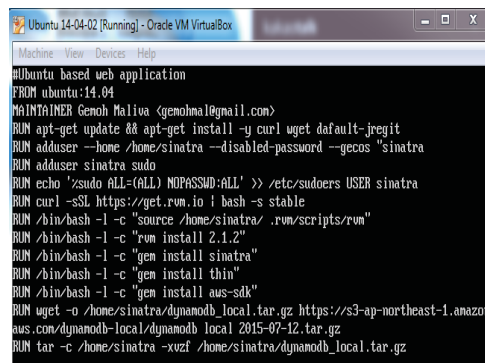


Fig. 3 A Web Application Architecture

First, we define a Docker image for launching a container for running the REST endpoint. I will then use this Docker image to test the code (the source code of a simple productive web application I downloaded from GitHub) on the Centos7 (acting like a laptop in this test environment). Later this created image can be used to test the code in Amazon EC2. The REST endpoints are going to be developed using Ruby and the Sinatra framework, so these will need to be installed in the image. Sinatra is open source software to write web application written in Ruby. We chose Sinatra framework in the test environment because it is an elegant web framework and really tiny. Sinatra is good fit for small scale projects and it does all what other heavy frameworks of the Ruby family such as Rail do. The backend will use Amazon DynamoDB to ensure that the application can be run from both inside and outside AWS web services, the Docker image will include the DynamoDB local database.

The Docker image is created using the DockerFile that contains all the instructions require to build an image. From the file, we will launch containers, install a bunch of software packages using the APT package manager, and then commit those changes to a new Docker image. DockerFile is a more powerful, fast and flexible way of creating Docker images. Here's the DockerFile we created for the app looks like:



```

#Ubuntu based web application
FROM ubuntu:14.04
MAINTAINER Genoh Maliwa <genohmali@gmail.com>
RUN apt-get update && apt-get install -y curl wget default-jpegit
RUN adduser --home /home/sinatra --disabled-password --gecos "sinatra"
RUN adduser sinatra sudo
RUN echo '%sudo ALL=(ALL) NOPASSWD:ALL' >> /etc/sudoers USER sinatra
RUN curl -sSL https://get.rvm.io | bash -s stable
RUN /bin/bash -l -c "source /home/sinatra/.rvm/scripts/rvm"
RUN /bin/bash -l -c "rvm install 2.1.2"
RUN /bin/bash -l -c "gem install sinatra"
RUN /bin/bash -l -c "gem install thin"
RUN /bin/bash -l -c "gem install aws-sdk"
RUN wget -o /home/sinatra/dynamodb_local.tar.gz https://s3-ap-northeast-1.amazonaws.com/dynamodb-local/dynamodb_local_2015-07-12.tar.gz
RUN tar -xzf /home/sinatra/dynamodb_local.tar.gz
  
```

Fig. 4 Creating an image using the Dockerfile

To build the image from the above DockerFile, I used this command
`$ docker build --tag="aws_activate/sinatra:v1"`

The tag option sets an identifier on the images and is usually setup as owner/repository:version. This makes it easy to identify what an image contains and who owns it when viewing the images in a registry.

Next I launched a container from this newly created image:

`$ docker run -it aws_activate/sinatra:v1 /bin/bash`

This command launches the container and goes into a bash shell. I can interact with the container inside just like I would on a Linux server. Because I'm

developing a web application, I cloned the image and commit the changes in the running container to a new image using this commit command.

```
$ docker commit -m "ready for testing" b9d03d60ba89 aws_activate/sinatra:v1.1
```

Version 1.1 of the container includes the Sinatra application that will serve up the REST endpoint.

The web application can be run using this command:

```
$ docker run -d -w /home/sinatra -p 10001:4567 aws_activate/sinatra:v1.1
./run_app.sh
```

The shell script starts up the local DYnamoDb database in the container and launches the Sinatra application using the thin webserver on port 4567. The web application can be view from the browser using `http://localhost:10001/activity/1` and see the following:

```
{"activity_id":"1", "user_id":" db430d35-92a0-49d6-ba79-
0f37ea1b35f7", "type":"meal", "calories":100, "date":"2015-
10-29 15:47:23 +0000"}
```

The endpoint seems to be working properly. The activity record was pulled from the local DynamoDB database and returned as JSON from the Sinatra application code.

8 Conclusion

In this paper, we looked at application optimization and deployment. Based on the challenges in software deployment environment and the numerous advantages of Docker, we proposed a multi-task cloud infrastructure using Docker and AWS services for rapid deployment, application optimization and isolation. We saw that this platform is for building, shipping, and running our applications. We can build any app in any language using any stack, dockerized the app and the app can run anywhere on anything. Furthermore, we saw how Amazon ECS helps solve challenging problems when running multiple container-based applications and services on a shared compute cluster.

For future work, we intend to fully complete the implementation of our proposed cloud platform and scale it up with Amazon EC2 container service for high performance container management. We will then conduct thorough evaluation to demonstrate the flexibility of our platform and then compare it with related existing platform.

Acknowledgments Following are results of a study on the “Leades INdustry-university Cooperation” Project, supported by the Ministry of Education, Science & Technology (MEST)

References

1. Zhang, Q., Cheng, L., Boutaba, R.: Cloud computing: state-of-the-art and research challenges. *Journal of Internet Services and Application* **1**, 7–18 (2010)
2. Yang, T.A., Joshy, N., Rojas, E., Anumula, S., Moola, J.: Virtualization and Data Center Design. *Global Journal on Technology* **9**, 36–54 (2015)
3. Kratzke, N.: Lightweight Virtualization Cluster How to Overcome Cloud Vendor Lock-In. *Journal of Computer and Communications* **2**, 1–7 (2014)
4. Kratzke, N.: Cloud Computing Costs and Benefits—An IT Management Point of View (2012). In: Ivanov, I., van Sinderen, M., Shiskov, B. (eds.) *Cloud Computing and Services Sciences*, pp. 185–203. Springer, New York (2014)
5. Merkel, D.: Docker: Lightweight Linux Containers for Consistent Development and Deployment. *Linux Journal* **2** (2014)
6. Caballer, M., Blanquer, I., Molto, G., de Alfonso, C.: Dynamic management of virtual infrastructures. *Journal of Grid Computing* **13**(1), 53–70 (2014)
7. Binz, T., Breitenbcher, U., Haupt, F., Kopp, O., Leymann, F., Nowak, A., Wagner, S.: OpenTOSCA - a runtime for TOSCAbased cloud applications. In: *ICSOC. LNCS*, vol. 8274, pp. 692–695. Springer (2013)
8. AmazonWebServices. AWSEC2. <http://docs.aws.amazon.com/AmazonECS/latest/developerguide/>
9. Keahey, K., Freeman, T.: Contextualization: providing one-click virtual clusters. In: *Fourth IEEE International Conference on eScience*, Indianapolis, Indiana, USA, pp. 301–308 (2008)
10. Bresnahan, J., Freeman, T., LaBissoniere, D., Keahey, K.: Managing appliance launches in infrastructure clouds. In: *Proceedings of the 2011 TeraGrid Conference: Extreme Digital Discovery, TG 2011*, vol. 12, pp. 1–12:7. ACM, New York (2011)