VALID 2020

The Twelfth International Conference on Advances in System Testing and Validation Lifecycle

ISBN: 978-1-61208-830-3

October 18 -22, 2020

VALID 2020 Editors

Jos van Rooyen, Identify, The Netherlands

Xinli Gu, Futurewei Technology, Inc., USA

VALID 2020

Forward

The Twelfth International Conference on Advances in System Testing and Validation Lifecycle (VALID 2020), held on October 18 - 22, 2020, continued a series of events focusing on designing robust components and systems with testability for various features of behavior and interconnection.

Complex distributed systems with heterogeneous interconnections operating at different speeds and based on various nano- and micro-technologies raise serious problems of testing, diagnosing, and debugging. Despite current solutions, virtualization and abstraction for large scale systems provide less visibility for vulnerability discovery and resolution, and make testing tedious, sometimes unsuccessful, if not properly thought from the design phase.

The conference on advances in system testing and validation considered the concepts, methodologies, and solutions dealing with designing robust and available systems. Its target covered aspects related to debugging and defects, vulnerability discovery, diagnosis, and testing.

The conference provided a forum where researchers were able to present recent research results and new research problems and directions related to them. The conference sought contributions presenting novel result and future research in all aspects of robust design methodologies, vulnerability discovery and resolution, diagnosis, debugging, and testing.

We welcomed technical papers presenting research and practical results, position papers addressing the pros and cons of specific proposals, such as those being discussed in the standard forums or in industry consortiums, survey papers addressing the key problems and solutions on any of the above topics, short papers on work in progress, and panel proposals.

We take here the opportunity to warmly thank all the members of the VALID 2020 technical program committee as well as the numerous reviewers. The creation of such a broad and high quality conference program would not have been possible without their involvement. We also kindly thank all the authors that dedicated much of their time and efforts to contribute to VALID 2020. We truly believe that thanks to all these efforts, the final conference program consists of top quality contributions.

This event could also not have been a reality without the support of many individuals, organizations and sponsors. We also gratefully thank the members of the VALID 2020 organizing committee for their help in handling the logistics and for their work that is making this professional meeting a success. We gratefully appreciate to the technical program committee co-chairs that contributed to identify the appropriate groups to submit contributions.

We hope the VALID 2020 was a successful international forum for the exchange of ideas and results between academia and industry and to promote further progress in system testing and validation.

VALID 2020 Steering Committee

Lorena Parra, Universitat Politecnica de Valencia, Spain

VALID 2020

Committee

VALID 2020 Steering Committee
Lorena Parra Boronat, Instituto Madrileño de Investigación y Desarrollo Rural, Agrario y Alimentario andUniversitat Politecnica de Valencia, Spain

VALID 2020 Technical Program Committee

Sajid Anwer, Griffith University, Brisbane, Australia
Deepika Badampudi, Blekinge Institute of Technology, Sweden
Sebastien Bardin, CEA LIST, France
Andrea Baruzzo, Interaction Design Solutions / University of Udine, Italy
Davide Basile, ISTI CNR Pisa, Italy
Ateet Bhalla, Independent Consultant, India
Bruno Blaskovic, University of Zagreb, Croatia
Hanifa Boucheneb, École Polytechnique de Montréal, Canada
Laura Brandán Briones, FaMAF | Univ. de Córdoba, Argentina
Mark Burgin, University of California Los Angeles (UCLA), USA
Laura Carnevali, University of Florence, Italy
Arjun Chaudhuri, Duke University, USA
Peter Clarke, Florida International University, USA
Bruce Cockburn, University of Alberta, Canada
Aleksa Damljanovic, Politecnico di Torino, Italy
Hichem Debbi, University of M'sila, Algeria
Giorgio Di Natale, TIMA - CNRS / Université Grenoble-Alpes / Grenoble INP UMR 5159, France
Luigi Dilillo, LIRMM (Laboratoire de Informatique Robotique et Microélectronique de Montpellier), France
Nikos Foutris, The University of Manchester, UK
Jicheng Fu, University of Central Oklahoma, USA
Gregory Gay, Chalmers and the University of Gothenburg, Sweden
Bidyut Gupta, Southern Illinois University, Carbondale, USA
Zoltán Horváth, Eötvös Loránd University, Budapest, Hungary
Sebastian Huhn, University of Bremen / DFKI Bremen, Germany
Ahmed Kamel, Concordia College, Moorhead, USA
Basel Katt, Norwegian University of Science and Technology, Norway
Dirk Kuhlmann, Fraunhofer-Institute for System- and Innovation Research (ISI), Germany
Richard Kuhn, National Institute of Standards & Technology, USA
Maurizio Leotta, University of Genova, Italy
Xia Li, The University of Texas at Dallas, USA
Yan-Fu Li, Tsinghua University, China
Chu-Ti Lin, National Chiayi University, Taiwan
Eda Marchetti, ISTI-CNR, Pisa, Italy
Abel Marrero, Bombardier Transportation Signal Germany GmbH, Germany

Vadim Mutilin, Ivannikov Institute for System Programming of the RAS (ISPRAS), Moscow, Russia
Roy Oberhauser, Aalen University, Germany
Rasha Osman, The Higher Technological Institute, Egypt
Adriano Peron, University of Napoli "Federico II", Italy
Pasqualina Potena, RISE Research Institutes of Sweden AB, Sweden
Claudia Raibulet, University of Milano-Bicocca, Italy
Kristin Yvonne Rozier, Iowa State University, USA
Hiroyuki Sato, University of Tokyo, Japan
Josep Silva, Universitat Politècnica de València, Spain
Maria Spichkova, RMIT University, Australia
Salvador Tamarit, PFS Group, Spain
Bedir Tekinerdogan, Wageningen University, The Netherlands
Spyros Tragoudas, Southern Illinois University, USA
Ana Turlea, University of Bucharest, Romania
Visa Vallivaara, NIST - National Institute of Standards and Technology | VTT - Technical Research Centre of Finland | University of Oulu, Finland
Jos van Rooyen, Identify - Software Quality Services, Netherlands
Miroslav N. Velev, Aries Design Automation, USA
Hironori Washizaki, Waseda University / National Institute of Informatics / SYSTEM INFORMATION, Japan
Kristian Wiklund, Ericsson AB, Sweden
Dietmar Winkler, Institute for Information Systems Engineering | TU Wien, Austria
Xiaofei Xie, Nanyang Technological University, Singapore
Haibo Yu, Kyushu Sangyo University, Japan
Pavol Zavarsky, Concordia University of Edmonton, Canada

Copyright Information

For your reference, this is the text governing the copyright release for material published by IARIA.

The copyright release is a transfer of publication rights, which allows IARIA and its partners to drive the dissemination of the published material. This allows IARIA to give articles increased visibility via distribution, inclusion in libraries, and arrangements for submission to indexes.

I, the undersigned, declare that the article is original, and that I represent the authors of this article in the copyright release matters. If this work has been done as work-for-hire, I have obtained all necessary clearances to execute a copyright release. I hereby irrevocably transfer exclusive copyright for this material to IARIA. I give IARIA permission or reproduce the work in any media format such as, but not limited to, print, digital, or electronic. I give IARIA permission to distribute the materials without restriction to any institutions or individuals. I give IARIA permission to submit the work for inclusion in article repositories as IARIA sees fit.

I, the undersigned, declare that to the best of my knowledge, the article is does not contain libelous or otherwise unlawful contents or invading the right of privacy or infringing on a proprietary right.

Following the copyright release, any circulated version of the article must bear the copyright notice and any header and footer information that IARIA applies to the published article.

IARIA grants royalty-free permission to the authors to disseminate the work, under the above provisions, for any academic, commercial, or industrial use. IARIA grants royalty-free permission to any individuals or institutions to make the article available electronically, online, or in print.

IARIA acknowledges that rights to any algorithm, process, procedure, apparatus, or articles of manufacture remain with the authors and their employers.

I, the undersigned, understand that IARIA will not be liable, in contract, tort (including, without limitation, negligence), pre-contract or other representations (other than fraudulent misrepresentations) or otherwise in connection with the publication of my work.

Exception to the above is made for work-for-hire performed while employed by the government. In that case, copyright to the material remains with the said government. The rightful owners (authors and government entity) grant unlimited and unrestricted permission to IARIA, IARIA's contractors, and IARIA's partners to further distribute the work.

Table of Contents

Testing and Validation of Monitoring Technologies to Assess the Performance and Genotyping of *Poa pratensis* (C3) Mixed with Other Grass Species (C4)

Pedro V. Mauri[2], Lorena Parra[2, 3], Jaime Lloret[3], Salima Yousfi[2] and José F. Marín[1]
[1] *Area verde MG Projects SL. C/ Oña, 43 28933 Madrid,* Spain
[2]Instituto Madrileño de Investigación y Desarrollo Rural, Agrario y Alimentario (IMIDRA), Finca "El Encin", A-2, Km 38, 2, 28800 Alcalá de Henares, Madrid, Spain
[3] Instituto de Investigación para la Gestión Integrada de Zonas Costeras Universitat Politècnica de València, Valencia, Spain
Email: pedro.mauri@madrid.org, loparbo@doctor.upv.es, jlloret@dcom.upv.es, salima.yousfi@madrid.org, jmarin@areaverde.es

Abstract—**Precision agriculture is vital to ensure the sustainability of farming systems. Nonetheless, the selection of parameters to be monitored can be a difficult decision, especially when the required equipment has a high cost. In this paper, we analyze the usability of five variables, including soil moisture, canopy temperature and three vegetation indexes, in turfgrass composed of different species. Our objectives are, on the one hand, determine which parameter or parameters are more specific for determining the species which compose the turfgrass. On the other hand, we expect to find correlations between variables in order to reduce the evaluated parameters in the turfgrass monitoring. Our results indicated that only the vegetation indexes are useful for genotyping, to determine the species that compose the turfgrass. From the vegetation indexes, the green area was the one which offers the best results. On the other hand, correlations were found between soil moisture and canopy temperature, and between the different vegetation indexes. Thus, we can affirm that it is possible to reduce the measured variables in turfgrass monitoring. The most significant advantage is the possibility of avoiding the monitoring of a vegetation index, for which the calculation requires a specific device with higher cost.**

Keywords-precision agriculture; soil moisture; canopy temperature; vegetation indexes; correlations; experimental plots.

I. INTRODUCTION

Precision Agriculture (PA) is becoming more and more popular in the last years due to its benefits for farmers [1]. The use of technology for crops monitoring, such as Wireless Sensor Networks (WSN) or Internet of Things (IoT) leads the farming activity to a higher degree of sustainability and profitability. Even so, the selection of the needs of our systems in terms of required data, data periodicity or monitored variables can be confusing. In the PA, several aspects can be monitored, such as soil, water, and plants. In a recent work [2], an evaluation of included parameters in PA concludes that most of the IoT-based smart irrigation systems are focused on monitoring the soil, and few of them monitored the plants.

The fact of measuring basically a single, or a limited number of parameters might be problematic. This is because the characteristics of soil such as Soil Moisture (SM), chiefly if it is measured in a unique location and close to the surface, can suffer abrupt changes. As the measurement of isolated parameters might drive the system into wrong actions, it is essential to combine several parameters in order to take the correct action. Nonetheless, it prompts us to another problem, the proper selection of parameters to be monitored. It is essential to monitor plant and soil parameters, given the fact that plant parameters are more stable in time than soil parameters [3]. Different parameters offer us different sort of information, which can be useful in order to take the most appropriate measure, i.e. when to irrigate, the required amount of water, required fertilizer, identification of plant diseases.

The dilemma of using different types of devices and techniques when we are monitoring the performance of crops (or gardens) is the high cost of some devices and the required time to gather data manually. In addition, the data processing and analyses may require more time than the value of obtained information if we include several parameters. Therefore, it is necessary to evaluate, test, and validate the real value of the information provided by different commonly used devices in the monitoring of plants, soils, and agriculture. This evaluation is critical to allow us to minimize the number of monitored parameters without reducing the conclusions based on the gathered information and the value of gathered data.

One of the activities which clearly demonstrate the possible benefits of reducing the number of monitored variables is public gardening. In gardening, we have one of the highest requirements of water and PA must help to reduce the irrigation [4] and evaluate the performance of different species to find a combination that requires less irrigation [5].

The aim of this paper is to evaluate, test, and validate which of the monitored parameters in experimental plots of turfgrass offers more valuable information. The objective of monitoring those parameters is two-fold. First, we use the monitored parameters to evaluate the performance of 4 grass combinations, including C3 and C4 plants (different in the water management). Furthermore, we expect to use gathered data to identify the grass combination, also known as genotyping. Therefore, we are going to gather data of 5 variables, including three vegetation indexes, the Canopy Temperature (CT), and the SM. With these data, a series of statistical analyses will be performed to try to evaluate the performance of each grass combination and to genotype the combinations. In addition, we also expect to find a correlation between different pairs of variables in order to reduce the number of parameters that must be monitored in the future.

The remainder of this paper is structured as follows. The presentation and analysis of the related work are presented in Section 2. Section 3 describes the materials and methods that have been used for this experiment. The results are detailed and discussed in Section 4. Finally, Section 5 outlines the main conclusions of this work.

II. Related Work

In this section, we describe several papers which analyze the correlation of parameters monitored in PA and the use of parameters for genotyping and monitoring crops and gardens.

First of all, the different vegetation indexes and their use are described. We have selected three vegetation indexes which are the most used vegetation indexes is the Normalized Difference Vegetation Index (NDVI). The NDVI is widely used for monitoring plant vigour in different crops. In a previous work, Marin et al. [5] have used the NDVI compared with RGB information, Green Area (GA) and the Greener Area (GGA), to compare the performance of grass combinations. They conclude that NDVI, GA and GGA can be used in the grass as an indicator of biomass and to estimate the resistance of the plant combinations to water restriction. However, the NDVI, GA, and GGA are not only used for grass monitoring but also crop assessment. In particular, its use is extended in cereal crops [6]-[8]. In [6], Fernandez-Gallego et al. used the aforementioned indexes for grain yield estimation in wheat as a low-cost option, compared with other existing methods. They conclude that the change in canopy colour from green to yellow is the most useful indicator for grain yield estimation. Yousfi et al. in [7] used NDVI, GA, Canopy Temperature Depression, and Stable Carbon Isotope Composition to determine the wheat grain yield under different irrigation and fertigation conditions. Their results pointed out the relevance of different indexes to estimate wheat harvest. In addition, the GA and the Stable Carbon Isotope Composition were the unique methods that offer a correlation with harvest in all the evaluated scenarios. In [8], Buchaillot et al. performed a similar study, including a Soil Plant Analysis Development (SPAD) sensor in maize fields. For the calculation of indexes, images captured with a drone and with a regular camera were used and correlated. Their results highlight the relevance of the evaluated indexes and the SPAD in grain yield estimation. It is important to note that, although some indexes might be attained with remote sensing, some of the included parameters cannot be measured with existing sensors. Therefore, the correlation or estimation of variables is crucial to reduce the number of sensors and simplify the infrastructure of WSN or IoT systems.

Another vital parameter, which is not monitored in most of the IoT proposals for irrigation is the CT [2]. The CT has a high relevance when drought-tolerant crop cultivars and the irrigation are being monitored. In [9], Zhang et al. gather data of the CT jointly with RGB and thermal images with a drone to evaluate the water stress of the crops. Their results indicate the importance of combining the CT with other technologies as the image gathered with the drone for a proper assessment of maize in water stress conditions. The use of CT for irrigation is discussed by Kumar et al. in [10]. In their experiments, the authors kept wheat plants under different degrees of water stress, CT and SM were monitored. Their results clearly indicate that using both variables in an algorithm for triggering irrigation events save up to 20% of water for irrigation. The CT measurement can be easily included in IoT systems or WSN with thermal cameras or infrared thermometers.

The measurement of CT in turfgrass is less standard but we can find some examples where the CT is monitored. In [11], Culpepper et al. developed an experiment combining different types of grasses and exposing them to different irrigation levels. The CT was useful to identify the plants kept with or without irrigation, but only in specific periods of the experiment. Meanwhile, the NDVI was not useful for differentiating the two scenarios. Another example can be found in Hong et al. [12]. The authors maintain *Agrostis stolonifera* under different regimes (100 to 15% of evapotranspiration) and images were captured using a thermal camera mounted over a drone. The CT has a high correlation with the irrigation regimes (-0.65 to 0.82) in different moments. Nevertheless, in general terms, other variables such as NDVI presented higher correlations.

As far as we know, the use and evaluation of NDVI, GA, GGA, CT, and SM to assess water stress or its correlation is not performed with *Poa pratensis* mixed with other C3. It is essential to evaluate if a reduction in the monitored parameters can be applied, to simplify the required sensor in the future deployment of WSN and IoT systems.

III. Material and Methods

In this section, the equipment and process used to gather the data, software employed to analyze it, and the details of the mixed plant species are portrayed.

A. *Experimental plots*

A total of 3 grass combinations, which include C3 and C4 species, have been tested in the research facilities of IMIDRA during 8 months. The mixtures of C3 and C4 grasses are kept in experimental plots of 4.5m^2 (1m per 3.5m). As a C3 grass, the Poa pratensis represents 75% of the planted seeds. As a C4 (25% of the plot) we include three different species combined individually with the C4 (*Cynodon dactylon (PC)*, *Buchloe dactyloides (PB)*, and *Zoysia japonica (PZ)*). Each one of the selected combinations is repeated six times in individual plots. In addition, the most used grasses combination in ornamental gardening is tested to serve as a control. This *Control* is composed of *Festuca arundinacea* (70%), Lolium perenne (15%), and *Poa pratensis* (15%).

Thus, a total of 22 plots are included in the experiment. All the plots have the same environmental conditions of soil and irrigation. The irrigation was automatically calculated by the Rain Bird [13]. In Figure 1, we can see a representation of the 3 combinations. The presence of pluviometers used to check the uniformity of irrigation can be seen in some of the plots.

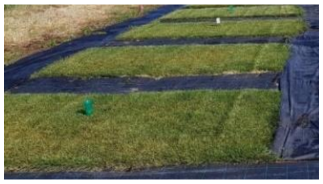


Figure 1. Experimental plots from up to down Control, *Poa pratensis* mixed with *Cynodon dactylon, Zoysia japonica*, and *Buchloe dactyloides.*

B. *Data gathering*

During the experimental period, which had a duration of 6 weeks, data was gathered in each plot one per week, from October to November. The data gathered include different types of variables related to the soil (soil moisture) and plant (canopy temperature and vegetation indexes).

To gather these data, different devices have been used. For the SM, the Time-Domain Reflectometry (TDR) 350 FieldScout was selected [14]. One measure is taken in each plot. Regarding the CT, a Fluke 561 was used [15]. This device allows us to collect the mean temperature of each plot.

Finally, for the lectures of the electromagnetic spectrum of the plant to regions are considered, the visible and the infrared. Two instruments were used, the first of them was a SONY DSC-W120, selected to obtain pictures of each plot. With this camera, we gather information about the visible spectrum. Meanwhile, the Handheld Crop Sensor GreenSeeker [16] was used to measure the information related to the red and infrared region. The GreenSeeker allows us having a mean value of the NDVI of the entire plot. The information gathered with the camera and the GreenSeeker were obtained, placing both instruments at 1.5m from the soil.

C. *Data processing*

Once the data were gathered, different processes are carried out. The first of them was to analyze the images with specific software, the BreedPix. It is open-source software, mainly used for cereal crops. With this software, we can obtain information about the GA and GGA contained in the picture. The GA contains the portion of the picture with pixels from yellow to bluish-green. On the other hand, the GGA excludes the yellowish-green tones.

Thus, for each plot, we have five variables (soil moisture (SM), canopy temperature (CT), NDVI, GA, and GGA). The variables were included in statistical software to analyze the relationship between variables and to analyze the performance of different grass species. The used software for data processing was the Statgraphics Centurion. Two different statistical tests were carried out. First, the ANalysis Of Variance (ANOVA) is performed to compare the mean and variances of included variables for each grass combination. To determine the existence of similitudes or differences between the evaluated grass combinations, the Tukey Honestly Significant Difference (HSD) was selected. Finally, bivariate correlations for each pair of variables are performed.

IV. RESULTS

In this section, we present our results and discuss their importance. First, the identification of differences between different plots is detailed. Finally, the correlation between the analyzed parameters is described.

A. *Testing the benefits of sensing devices to evaluate the performance of different grass combinations*

Considering that in each plot we obtain an individual measure of each one of the evaluated parameters and we have 22 plots monitored during six weeks, a total of 132 observations were carried out for each variable, which is considered a significant amount of data. It is important to note that at plain sight, it is not easy to differentiate between combination. Only experts are capable of identifying the differences in their leaves.

Before performing the ANOVA to evaluate if the user devices can be useful to differentiate between different combinations (genotyping), it is essential to confirm that data follows a normal distribution. It is a prerequisite for the ANOVA. In Table 1, we have included the skewness and kurtosis if obtained indexes are between ±2 we can use the ANOVA tests. Data included in Table 1 indicates that SM, CT, and GA follow normal distributions and ANOVA tests can be performed. Nonetheless, the variables NDVI and GGA do not follow a normal distribution; thus, alternative tests must be performed. In this case, the test median of Mood will substitute to the ANOVA, and the Kruskal-Wallis will be used to estimate the different groups. The results of variance analyses are summarized in Table 2.

TABLE I. SUMMARY SKEWNESS AND KURTOSIS OF DATA.

	Skewness					**Kurtosis**				
	SM	**CT**	**NDVI**	**GA**	**GGA**	**SM**	**CT**	**NDVI**	**GA**	**GGA**
Control	0.686056	0.904177	-0.8997	0.138911	0.929755	-0.56894	-0.187175	-0.9178	-1.28267	-0.76948
PC	0.325911	1.36113	-1.7536	-0.14744	1.29273	-0.60346	0.335747	-0.7891	-1.09587	-0.94227
PB	1.4669	1.31779	-2.460	-0.43471	2.36019	-0.37907	0.00324324	0.9359	-0.32923	0.109402
PZ	1.07606	1.55038	-2.1278	0.999044	1.52993	-0.54186	0.0515326	0.4999	-0.89322	-0.67507
Normal Distribution	Yes	Yes	No	Yes	No					

TABLE II. SUMMARY OF ANOVA AND KRUSKAL-WALLIS. SIGNIFICANCE LEVELS: NS, NOT SIGNIFICANT; * $p < 0.05$; ** $p < 0.01$ AND *** $p < 0.001$. THE DIFFERENT LETTER SUCCEEDING THE MEANS ARE SIGNIFICANTLY DIFFERENT ($p < 0.05$) ACCORDING TO TUKEY'S HONESTLY SIGNIFICANT DIFFERENCE (HSD) TEST.

	SM	**CT**	**NDVI**	**GA**	**GGA**
Control	35.2583 ^{a}	14.6125 ^{a}	0.76 ^{a}	0.67875 ^{b}	0.35 ^{a}
PC	35.5 ^{a}	14.8417 ^{a}	0.745 ^{a}	0.61805 ^{a}	0.295 ^{a}
PB	34.3944 ^{a}	14.6056 ^{a}	0.79 ^{b}	0.77944 ^{c}	0.48 ^{b}
PZ	36.3722 ^{a}	14.4694 ^{a}	0.77 ^{b}	0.76472 ^{c}	0.425 ^{b}
Level of significance	0.8727 ^{ns}	0.9579 ^{ns}	0.0005***	0.0000***	0.0000***

According to the data presented in Table 2, we can affirm that SM and CT have no variation, which means that those parameters cannot be used to identify the different combinations. Thus, SM and CT are variables which are not useful for genotyping. In addition, those variables are profoundly affected by environmental conditions and can experience huge variations along the day.

On the other hand, the variables that consider the electromagnetic spectrum of the plants (visible and infrared) are offering more remarkable information. We need to remark that this data is more stable in the time, there is no variation along the day and it is not quickly affected by the environmental parameters such as solar radiation, wind, or rain among others.

With the data of NDVI, it is possible to identify two groups of genotypes. The first group includes the mixture of C3 species (Control) and the Poa with Cynodon. Meanwhile, the mixtures of Poa with Zoysia and Buchloe have different values and belong to a separate group. Therefore, information of NDVI is not suitable to identify the presence of C4 species in all cases. The highest NDVI values are linked to the second group (PZ and PB).

With regard to the information from the visible spectrum, different results were obtained with GA and GGA. In both cases, the results of the tests have pointed out that there are differences in the observed grass species since the p-values are lower than 0.05. The GGA index can identify differences in two groups of plots. The Control and te PC mixtures form the groups on the one hand, and PB and PZ on the other hand. These results are coupled with the outcome obtained with the NDVI.

On the contrary, the results obtained with GA data are more specific than with GGA and NDVI. Again, the ANOVA indicate with a p-value lower than 0.05 that there are differences statistically significant among the different grass combinations tested. In this case, the multiple range test indicates that it is possible to identify three different groups. PC mixture is included in the first group. The second group is composed solely by the Control grass combination. Finally, PZ and PB are the mixtures identified as the third group. The different groups and distribution of data can be seen in the Box diagram of Figure 2. In this graphic, the mean, median outliers and other relevant information are summarized. Figure 2 presents clearly the similarity in the data of PB and PZ, which cannot be differentiated with GA data.

Thus, the data of GA offers better results than the other variables. It is important to note that with NDVI and GGA data we have worked with non-parametric statistical tests due to the distribution of the data, and those tests tend to be less powerful to identify differences than parametric tests. To obtain better results with GGA and NDVI, we would need a larger amount of data. Therefore, it is possible that GGA and NDVI can be used in the future with similar accuracy than GA if the amount of data increase.

B. *Correlation between evaluated variables*

The seek of correlations between data, we aim to find the relation between variables in order to reduce the number of controlled variables in experimental plots. The fact of gathering data from several variables implies an elevated time consumption in the plots using diverse types of equipment. In addition, some of the used equipment (particularly the GreenSeeker) have a high cost and having the opportunity of using another tool, as the digital pictures, to estimate the value of NDVI is vital to save costs.

Thus, we are going to focus the correlation between variables in trying to obtain an equation that allows us to obtain or predict the value of NDVI based on the information obtained from the digital pictures. In addition, we will seek to have a correlation between SM and CT, since the measurement of the CT is much faster, and the required equipment is cheaper than the required for SM measuring.

To explore the existing correlation in the gathered data, and to attend to the non-normal distribution of some variables, Spearman correlation is selected. Although it is less powerful than the Pearson correlation, the existence of variables without normal distribution force us to use this test. The first outcome of the correlation test is the correlation graphic, in which the X-Y distributions of each pair of variables, also known as Dispersion Matrix, can be seen in Figure 3 a). Therefore, we can have in a simple graphic the tend of data of all the included variables in the test. According to the results of presented in Figure 3, we can identify at plain sight that some of the variables are highly correlated.

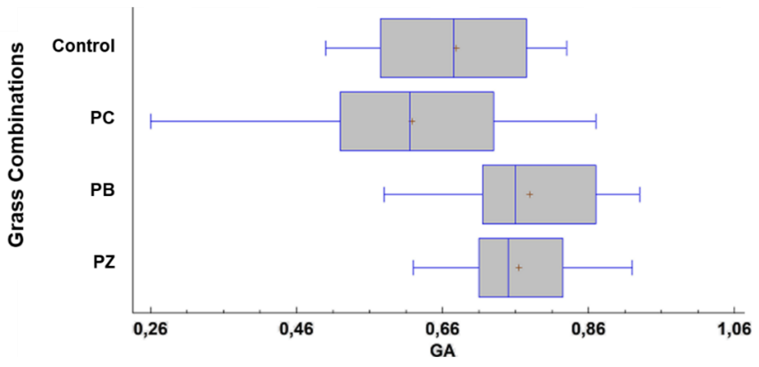
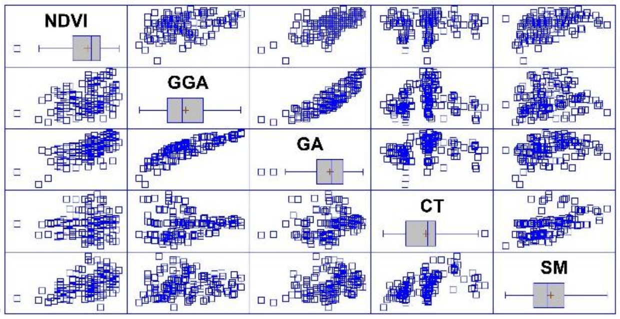
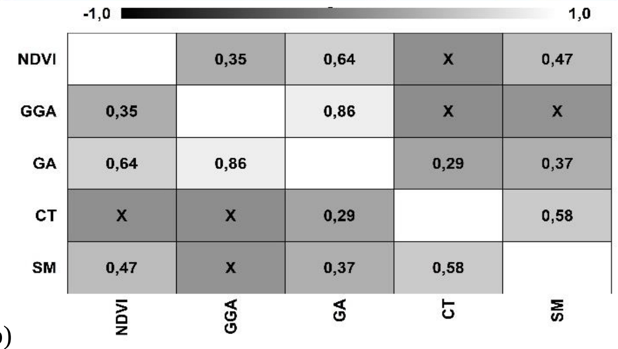


Figure 2. Box Diagram with GA data for the different grass combinations where the different distribution of data can be identified.

a)


b)


	NDVI	GGA	GA	CT	SM
NDVI		0,35	0,64	X	0,47
GGA	0,35		0,86	X	X
GA	0,64	0,86		0,29	0,37
CT	X	X	0,29		0,58
SM	0,47	X	0,37	0,58	

Figure 3. Correlation between different variables, a) Dispersion Matrix, b) Spearman Correlation Graphic (where X means that their correlation is not statistically significant, and the numbers indicate the strength of correlations from -1 to +1).

Meanwhile, in Figure 3 b) we depict the strength of correlations between each pair of variables. The values close to +1 indicate a strong and positive correlation, while values close to -1 indicate a strong negative correlation. The strongest correlations for the variables of plant aspect, or electromagnetic spectrum, are found for GA with GGA (0.86), and GA with NDVI (0.64). The correlation between GGA and NDVI is much lower. On the other hand, regarding the plant-soil interaction, a correlation was found between SM and CT (0.58). Last but not least, another interesting correlation between SM and NDVI was also found; however, the strength of this correlation is lower than 0.50 (0.47).

Considering the results of the correlation test, we are going to focus on the relation between GA and GGA, GA and NDVI and TC and SM. The objective, as described before, is to reduce the number of required measures and required equipment for grass monitoring. The relation between variables GA and GGA can be explained with a linear model, in which given a certain value of GA it is possible to estimate the GGA for the picture. The proposed mathematical model is described in (1). It is important to note that this model is developed for the different combinations of *Poa pratensis* with other C4 species and include the Control mixture. The proposed model is characterized by a correlation coefficient of 0.83 and an R2 of 70% and can be seen in Figure 4. Although there are other models which can explain with higher accuracy the relation between both variables (R2 of 76%), we have selected the linear model due to its higher simplicity and lower complexion in the calculation.

Concerning the relation between variables GA and NDVI, again a linear model, in which given a specific value of GA, it is possible to estimate the NDVI, is presented. Equation (2) described the linear mathematical model that related both variables. In this case, among all the evaluated mathematical models, the linear regression was the one that offered higher accuracy. The proposed model is characterized by a correlation coefficient of 0.65 and an R2 of 43% and can be seen in Figure 5. The equation of the proposed model is detailed in (2).

Finally, the correlation found between SM and CT is displayed. In this case and given the low accuracy of the linear model, we have selected the “S-Curve” model. The S-Curve model, which can be seen in Figure 6 has a correlation coefficient of -0.57 and an R2 of 33.25. The equation that follows the model is depicted in (3).

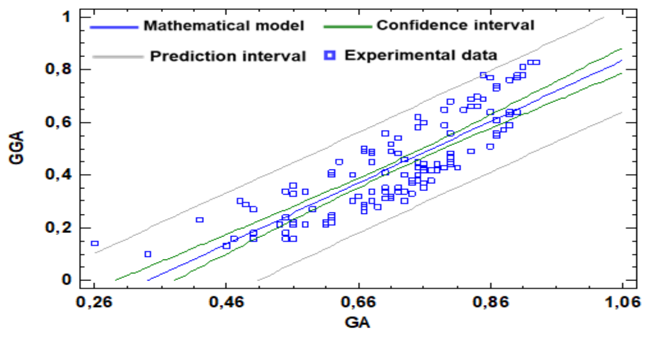


Figure 4. Simple regression between GA and GGA with a lineal mathematic model

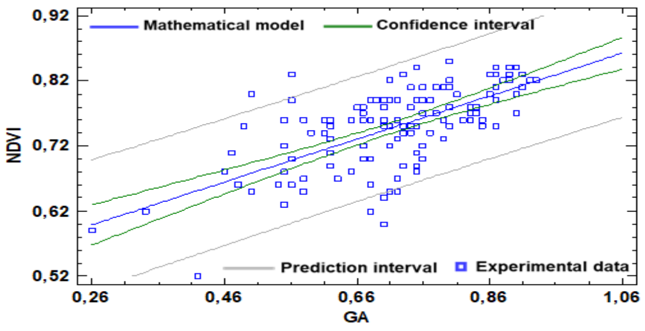


Figure 5. Simple regression between GA and NDVI with a lineal mathematic model

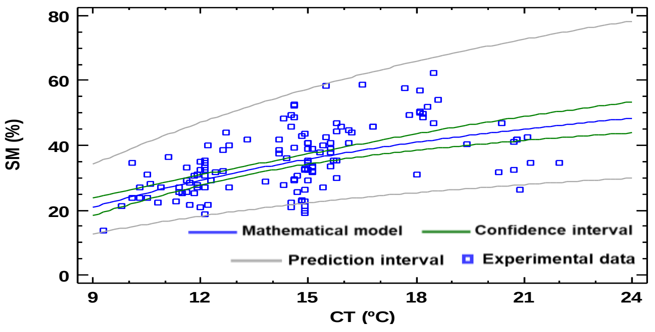


Figure 6. Simple regression between CT and SM with an S-Curve mathematic model

$$GGA = 1.16152*GA - 0.395785 \quad (1)$$

$$\mathrm{NDVI} = 0.513359 + 0.329084 \cdot \mathrm{GA} \tag{2}$$
$$\mathrm{SM}\ (\%) = \exp(4.38257 - 12.0825/\mathrm{CT}\ (^{\circ}\mathrm{C})) \tag{3}$$

The obtained results will allow the attainment of two objectives. First, we verify a methodology to identify the species that compose a lawn of *Poa pratensis* with other grasses. Secondly, we present the correlation of variables that will allow a reduction in the number of monitored parameters in future experiments in which WSN and IoT are deployed. We can use only one type of camera to monitor the GA and estimate the GGA and NDVI from (1) and (2). Furthermore, we will avoid the need of CT measurement by estimating this value from the SM data. Thus, we will have the deployed sensor underground instead than at certain height, as need for CT measurement. This will facilitate the integration of IoT systems with the daily activities carried out in lawns such mowing or irrigating, which can be problematic with sensor deployed over the ground.

It is important to note that the obtained results are only based on data from *Poa pratensis* and more data must be gathered to extrapolate our results to general turfgrass assessment.

V. CONCLUSION

The fact of using several devices for monitoring agriculture, or gardening, is widely discussed in this paper. The tradeoff between the relevance of gathered information and required time and costs to obtain this data is presented. To solve this problem, we have evaluated the existing correlation between different variables. Furthermore, the effectiveness of each studied parameter for monitoring grass performance and genotyping different species is presented.

In this paper, five different variables monitored in precision agriculture are evaluated for genotyping, and the existing correlation between variables is explored. Our results point out that the variable which offers better results for genotyping different grass combinations, including C3 and C4 plants, is the GA index. Other evaluated indexes such as GGA and NDVI offered promising results, but more data is required to evaluate their capabilities. With regards to existing correlations, we found a correlation between CT and SM, GA and NDVI, and GA and GAA. From those correlations, the most interesting one is the possibility of estimating the SM based on the CT.

In future work, we are going to include the measure of the temperature of the soil surface, with no coverage, in our datasets in order to obtain a more accurate estimation of SM from the data of CT and soil temperature and the estimation of coverage based on [17]. On the other hand, for the genotyping, we will include data of other grass mixtures to determine if GA by itself can identify more genotypes.

ACKNOWLEDGEMENT

This work has been partially funded by AREA VERDE-MG projects, Projects GO-PDR18-XEROCESPED funded by the European Agricultural Fund for Rural Development (EAFRD) and IMIDRA, by the European Union through the ERANETMED (Euromediterranean Cooperation through ERANET joint activities and beyond) project ERANETMED3-227 SMARTWATIR, and by Conselleria de Educación, Cultura y Deporte with the Subvenciones para la contratación de personal investigador en fase postdoctoral, grant number APOSTD/2019/04.

REFERENCES

[1] R. Gebbers and V. I. Adamchuk, "Precision agriculture and food security", *Science*, 327(5967), 2010, pp. 828-831.

[2] L. García, L. Parra, J. M. Jimenez, J. Lloret, and P. Lorenz, "IoT-Based Smart Irrigation Systems: An Overview on the Recent Trends on Sensors and IoT Systems for Irrigation in Precision Agriculture", *Sensors*, 20(4), 2020, pp. 1042.

[3] S. Marios and J. Georgiou, "Precision agriculture: Challenges in sensors and electronics for real-time soil and plant monitoring", 2017 IEEE Biomedical Circuits and Systems Conference (BioCAS), 19-21 October, Turin, Italy, 2017, pp. 1-4.

[4] A., Glória, C. Dionísio, G. Simões, J. Cardoso, and P. Sebastião, "Water Management for Sustainable Irrigation Systems Using Internet-of-Things", *Sensors*, 20(5), 2020, pp. 1402.

[5] J. Marín et al., "RGB Vegetation Indices, NDVI, and Biomass as Indicators to Evaluate C3 and C4 Turfgrass under Different Water Conditions", *Sustainability*, 12(6), 2020, pp. 2160.

[6] J. A. Fernandez-Gallego et al., "Low-cost assessment of grain yield in durum wheat using RGB images", *European Journal of Agronomy*, 105, 2019, pp. 146-156.

[7] S. Yousfi et al., "Combined use of low-cost remote sensing techniques and δ13C to assess bread wheat grain yield under different water and nitrogen conditions", *Agronomy*, 9(6), 2019, pp. 285.

[8] M. Buchaillot, et al. "Evaluating maize genotype performance under low nitrogen conditions using RGB UAV phenotyping techniques", *Sensors*, 19(8), 2019, pp. 1815.

[9] L. Zhang, Y. Niu, H. Zhang, W. Han, G. Li, J. Tang, and X. Peng, "Maize canopy temperature extracted from UAV thermal and RGB imagery and its application in water stress monitoring", *Frontiers in plant science*, 10, 2019, pp. 1270.

[10] N. Kumar, A. Poddar, V. Shankar, C. S. P. Ojha, and A. J. Adeloye, "Crop water stress index for scheduling irrigation of Indian mustard (Brassica juncea) based on water use efficiency considerations", *Journal of Agronomy and Crop Science*, 206(1), 2020, pp. 148-159.

[11] T. Culpepper, J. Young, and B. Wherley, "Comparison of four warm-season turfgrass species to natural rainfall or supplemental irrigation in a semiarid climate", *Agrosystems, Geosciences & Environment*, 3(1), 2020, pp. e20011.

[12] M. Hong, D. J. Bremer, and D. van der Merwe, "Thermal Imaging Detects Early Drought Stress in Turfgrass Utilizing Small Unmanned Aircraft Systems", *Agrosystems, Geosciences & Environment*, 2(1), 2019, pp. 1-9.

[13] RainBird ESP-LXME Controller Installation, Programming & Operation Guide. Available at: https://www.rainbird.com/sites/default/files/media/documents/2018-02/man_ESP-LXME-Installation-Operation-Guide_en.pdf. Last access on 03/06/2020

[14] TDR 350 Manual. Available at: https://www.specmeters.com/assets/1/22/6435_TDR_350_manual_(web).pdf. Last access on 30/06/2020

[15] Fluke 561 Infrared & Contact Thermometer manual. Available at: https://dam-assets.fluke.com/s3fs-public/56x_____umeng0000.pdf. Last access on 30/06/2020

[16] GreenSeeker Information. Available at: http://trl.trimble.com/docushare/dsweb/Get/Document-475150/022503-1123A_GreenSeeker_DS_MarketSmart_USL_0415_LR_web.pdf. Last access on 30/06/2020

[17] J. Marín, J. Rocher, L. Parra, S. Sendra, J. Lloret, and P. V. Mauri, "Autonomous WSN for Lawns Monitoring in Smart Cities," 2017 IEEE/ACS 14th International Conference on Computer Systems and Applications (AICCSA), 30 October-03 November, Hammamet, Tunisia, 2017, pp. 501-508.

Toward an Exact Simulation Interval for Multiprocessor Real-Time Systems Validation

Joumana Lagha, Jean-Luc Béchennec, Sébastien Faucou and Olivier-H Roux
Université de Nantes, École Centrale de Nantes, CNRS, LS2N, UMR 6004, F-44000 Nantes, France
Email: firstname.lastname@ls2n.fr

Abstract—**In order to study the schedulability of complex real-time systems, simulation can be used. Of course, to achieve formal validation of schedulability, simulations must be run long enough such that the schedule repeats. An upper bound on the length of the simulation that is valid for a very wide class of systems running on top of identical multiprocessor platforms is given in a previous work. It is known that this bound is pessimistic. In this paper, we derive a characterization of the exact bound for the same class of systems and describe an algorithm for its computation. We use it to quantify the pessimism of the upper bound on a set of synthesized systems. We also give some directions to explore the complexity vs. tightness trade-off for this problem.**

Keywords–Real time scheduling; Multiprocessor; Simulation.

I. Introduction

The correctness of real-time software systems does not depend only on the value of results, but also on the date they are produced. A real-time software is usually composed of a set of recurring tasks that spawns jobs. Each job must be executed within a given deadline. Scheduling algorithms are used to allocate execution time to jobs. Schedulability analysis is used to validate that the resulting schedule meets all deadlines. For well-defined classes of systems, efficient schedulability tests exist [1]. For complex systems, that are not in one of these classes, it is sometimes possible to rely on simulation. More precisely, this is possible if the context does not yield scheduling anomalies, *ie.* when response times variations are monotonic with regards to other system parameters. In this paper, we will assume work under this hypothesis and refer the reader to Section 7 of [2] for a discussion on this point.

To achieve formal validation of the system, the simulation must be run on an interval long enough such that the schedule repeats. If the scheduler is deterministic and memoryless, then, if in this interval all jobs meet their deadline, it can be safely concluded that the system is schedulable. The length of this interval can be discovered during simulation by comparing each new state to those encountered so far. The main drawback of this approach is that it requires to memorize all states, so it quickly becomes intractable. An alternative consists of computing an upper bound B on the length of the simulation interval and then simulate the system on $[0, B)$. In 2016, Goossens *et al.* [2] proposed an upper bound that is valid for a wide class of systems: periodic asynchronous tasks with arbitrary deadlines and structural constraints (such as precedence, mutual exclusion and self-suspension) scheduled on top of an identical multiprocessor platform by any deterministic and memoryless algorithm.

It is known that this bound is pessimistic, especially because it does not take into account the processing power of the platform. Before looking for possible improvements, it is interesting to evaluate how pessimistic it is. To answer this question, we derive a characterization of the exact bound for the same class of systems and describe an algorithm for its computation. The algorithm relies on an enumeration of the state space and has factorial time complexity. On a set of synthetic systems, we find out that the pessimistic bound is at least twice too long when the number of tasks is greater than three times the number of processors. Based on the exact formulation of the bound, we also suggest directions to explore tightness vs. complexity trade-off for this problem.

The paper is organized as follows: in Section II, we review related works. In Section III, we define notations and expose the state-of-the-art. In Section IV, we give a characterization of the exact bound and derive an algorithm for its computation. In Section V, we compare the state-of-the-art and the exact bound on a set of synthetic benchmarks to quantify its pessimism. In Section VI, we present possible directions to explore the tightness vs. complexity trade-off before concluding the paper.

II. Related works

The first result on simulation intervals is obtained by Leung and Merrill [3] in 1980, with $O^{max} + 2H$ (where O^{max} is the maximum activation offset and H is the hyperperiod) as an upper bound for independent asynchronous task systems with constrained deadlines scheduled with a fixed-task priority algorithm. The same bound was later deemed valid for systems with arbitrary deadlines by Goossens and Devillers [4]. For multiprocessor platforms, Cucu and Goossens [5] derive in 2007 a result for independent asynchronous task systems (a task system is asynchronous if at least two tasks have their first activation on different dates) with arbitrary deadlines scheduled by a global fixed-task priority algorithm. They also prove that any feasible schedule generated by a deterministic and memoryless scheduler is ultimately periodic. In 2012, Baru *et al.* [6] proposed an upper bound on the simulation interval for asynchronous task systems with constrained deadlines subject to simple precedence constraints running on an identical multiprocessor platform and scheduled by any deterministic and memoryless algorithm. The same interval is used and tuned for fixed-job priority schedulers and independent tasks in Nélis *et al.* [7]. The most recent and general result is the one proposed by Goossens *et al.* [2] in 2016, that applies to a very large class of systems: asynchronous task systems with arbitrary deadlines, subject to structural constraints (precedence, mutual exclusion, self suspension), scheduled on an identical multiprocessor platform by any deterministic and memoryless scheduler. This bound has a low complexity, is safe but not always tight. For many systems, it is very pessimistic and too big to be used for practical purpose. Thus, in this paper, we aim at deriving an exact bound. To do so, we relax the constraint on the complexity of the computation. Dues to its complexity, our bound can be computed for a restricted class of systems. For these systems, it provides an exact simulation interval. While deriving an exact bound, we also highlight how to explore the complexity vs. precision trade-off, paving the way to the development of low complexity yet precise bounds.

III. Upper bound on the simulation interval [2]

A. Model, notations, and definitions

$\mathbb{N}$ is the set of integer numbers. Let $\mathbf{v}$ be a vector of $\mathbb{N}^N$. $\forall i \in [1, N]$, $\mathbf{v}[i]$ is the ith element of the vector $\mathbf{v}$. We note that $\mathbf{0}$ the null vector: $\forall i \in [1, N].\, \mathbf{0}[i] = 0$. The usual operators $+, -, \times, <$ and $=$ are used on vectors of $\mathbb{N}^N$ and are the point-wise extensions of their counterparts in $\mathbb{N}$. $\{x \mid P(x)\}$ is set set of all x such that predicate $P(x)$ is true. $[x \mid P(x)]$ is the list of all x such that predicate $P(x)$ is true.

Let $\Theta = \{\tau_1, \tau_2, \ldots, \tau_N\}$ be a set of N asynchronous periodic tasks, where each task τ_i is the 4-tuple of non negative integers $\langle O_i, C_i, T_i, D_i\rangle$, where O_i is the release time of the first job of τ_i, C_i is the execution time of τ_i, T_i is the period of τ_i, and D_i its deadline. We assume that periods and deadlines are unrelated (*i.e.*, D_i can be smaller than, equal to, or greater than T_i). $H = lcm_{\tau_i \in \Theta}\{T_i\}$ is the hyperperiod of Θ.

At runtime, each task τ_i spawns an infinite sequence of jobs $\tau_{i,1}, \tau_{i,2}, \ldots$. Job $\tau_{i,j}$ enters the system at date $a_{i,j} = O_i + (j-1)T_i$. It must be executed before date $d_{i,j} = a_{i,j} + D_i$.

Let $S(t)$ be the state of the system at date t. It is defined by $S(t) = (C_{rem_1}(t), \ldots, C_{rem_n}(t), \Omega_1(t), \ldots \Omega_n(t))$, where C_{rem_i} is the remaining work to process for the jobs of task τ_i activated prior to t, and $\Omega_i(t)$ is a decrementing clock counting the time until the next release of a job of τ_i.

Θ is executed on a platform composed of m identical processors. Jobs are scheduled by a deterministic and memoryless scheduler (see below). A given job is executed sequentially (no inner parallelism) but can migrate from one processor to another during its execution. It is assumed that there is no penalty to migrate from one processor to another. Moreover, it is assumed that a job cannot start its execution while all jobs of the same task activated before are not finished.

Definition 1 (Feasible schedule): A feasible schedule for Θ is an infinite schedule such that every job $\tau_{i,j}$ is fully executed in its time window $[a_{i,j}, d_{i,j}]$.

Definition 2 (Deterministic and memoryless scheduler): A scheduler such that the scheduling decision at time t is unique and depends only on the current state of the system.

Definition 3 (Valid simulation interval): Interval [0,B) is a valid simulation interval for Θ scheduled with a deterministic and memoryless scheduler if and only if $\exists(t_1, t_2) \in [0, B]^2. \quad t_1 \neq t_2 \wedge S(t_1) = S(t_2)$.

The model has two features that make its schedulability analysis complex: arbitrary deadlines and asynchronous activation. Both are sources of backlog between hyperperiods.

Definition 4 (Backlog): The backlog $\beta_i(t)$ of a task τ_i at date t is defined as the remaining work to be processed for jobs of τ_i activated strictly before t.

In the following, we assume that all hypotheses formulated in this section hold.

B. Ruling out asynchronous activations

To rule out the complexity arising from asynchronous task activation, Goossens *et al*. observe that a simple transformation can be applied to an asynchronous task set Θ to obtain a synchronous task set Θ' such that the length of the simulation interval of Θ' (considering any deterministic and memoryless scheduler) is not smaller than that of Θ. For each task $\tau_i = \langle O_i, T_i, D_i\rangle$, the transformation yields $\tau_i' = \langle 0, T_i, O_i + D_i\rangle$.

The idea is that all feasible schedules of Θ are also feasible schedules of Θ'. Thus, if a simulation is run for a duration long enough to validate any feasible schedule of Θ', it is also long enough to validate any feasible schedule of Θ. A detailed proof is given in [2].

Given this result, we can now reason as if we only had to handle synchronous task sets. Thus, we can now give a trivial upper bound on the backlog of a task at the end of a hyperperiod: $\forall q > 0.\, \beta_i(qH) \leq (O_i + D_i) - T_i$. From now on, we note $\beta_i^{max} = \max\{0, (O_i + D_i) - T_i\}$ the maximum backlog for task τ_i at any date $t = qH$ in any feasible schedule, and we note $\beta^{max} = \max_{\tau_i \in \Theta} \beta_i^{max}$.

C. Extension to structural constraints

The approach used to rule out asynchronous activation can be used to extend the result to systems with structural constraints. Structural constraints are defined as "*a relation between jobs or subjobs, forbidding some execution orders, preemptions, or insuring a minimal delay between the end of a job (or sub-job) and the start of another one*" [2]. Let Θ a system with structural constraints. Let Θ' denote the same system where all structural constraints have been removed. Obviously, all feasible schedules of Θ are also feasible schedules of Θ'. Thus, a valid simulation interval for Θ' is also a valid simulation interval for Θ.

D. Deriving the bound

In any non trivial synchronous system such that at least two tasks have different periods, the search for the upper bound of a valid simulation interval can be reduced to solutions of the form $B = qH$ (with q a positive integer) by definition of H (the hyperperiod of Θ) since local clocks are equal in $S(0)$ and $S(qH)$.

By definition, for any non negative integer q, $C_{rem_i}(qH) = \beta_i(qH)$, so in any feasible schedule, $C_{rem_i}(qH) \leq \beta_i^{max}$. Then, we can bound the number of different states of the system in any feasible schedule at the end of a hyperperiod: $|\{S(qH) \mid q \in \mathbb{N}\}| \leq \prod_{i \in [1,N]} (\beta_i^{max} + 1)$. Using the assumption that the scheduler is deterministic and memoryless, it is sufficient to run the simulation long enough to cover a number of hyperperiods equal to the number of different states at the end of a hyperperiod. If the schedule is not feasible then a deadline miss will be discovered. If the schedule is feasible, either the same state will have been encountered twice, or all states will have been explored. This yields the bound:

$$B_0 = H \times \prod_{i\in[1,N]} (\beta_i^{max} + 1) \tag{1}$$

E. Non tightness of B_0

As claimed by the authors in [2], the bound is safe but not tight. This is illustrated in Figure 1. Let us consider a system with two tasks τ_1 and τ_2 such that $0 < \beta_1^{max} < \beta_2^{max}$, running on a monoprocessor platform. The size of the state space considered by the bound B computed above is the number of points with integer coordinates in the rectangle of width β_2^{max} and height β_1^{max}. Now, let us consider the points in the red dotted area. They correspond to a pending work at the end of a hyperperiod, which is strictly greater than $\max\{\beta_1^{max}, \beta_2^{max}\} = \beta_2^{max}$. Starting from such a state at any $t = qH$, in any schedule, at least one job activated before t will finish after $t + \beta_2^{max}$ thus missing its deadline. We conclude that this state can not belong to any feasible schedule.

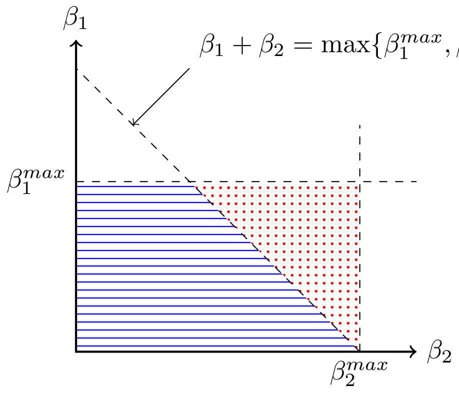


Figure 1. Illustration of the non-tightness of the bound: states in the red dotted area do not belong to any feasible schedule.

IV. Exact bound on the simulation interval

A. Characterization

We have seen with Figure 1 that B_0 fails to take into account diagonal constraints arising from the fact that the platform limits the execution parallelism and thus the maximum amount of cumulative backlog at the end of a hyperperiod. We can generalize this argument to derive a characterization of the bound as a set of linear constraints. Let $\Lambda \subseteq \Gamma$ be a subset of the task set. On a monoprocessor platform, the cumulative backlog at the end of a hyperperiod generated by tasks in Λ is bounded by $\max[\beta_i^{max} \mid \tau_i \in \Lambda]$ where $\max$ returns the maximum value of a list. On a 2-processor platform, execution parallelism allows us to achieve a higher bound: $\max_2[\beta_i^{max} \mid \tau_i \in \Lambda]$ where $\max_2$ returns the sum of the 2 greatest values of a list. Indeed, even if two jobs can be executed in parallelism, in a feasible schedule, they cannot overrun their deadlines. Thus, when the time is past the penultimate deadline, only one job among the jobs activated before the end of the hyperperiod, has not reached its deadline, so in a feasible schedule only this job could be running. Further generalizing this argument, on a m-processor platform, every $\Lambda \subseteq \Gamma$ yields the constraints $\sum_{\tau_i\in\Lambda} \beta_i \leq \max_m[\beta_i^{max} \mid \tau_i \in \Lambda]$ where $\max_m$ returns the sum of the m greatest values of a list. This allows us to characterize the number of possible states at the end of a hyperperiod (since we know that all local clocks are null at such instants, the state is truncated to its $C_{rem_i(t)}$ components).

$$\mathcal{S} = \Big\{ \mathbf{x} \mid \mathbf{x} \in \mathbb{N}^N \wedge \\ \forall \Lambda \subseteq \Theta. \sum_{\tau_i\in\Lambda} \mathbf{x}[i] \leq \max_m[\beta_i^{max} \mid \tau_i \in \Lambda] \Big\} \tag{2}$$

From this, we can derive the exact value of the bound on the simulation interval:

$$B_1 = H \times |\mathcal{S}| \tag{3}$$

Note that using 2 to compute B involves computing the power set of Θ (to enumerate all possible values of Λ), which has $2^{|\Theta|}$ elements, and then enumerating the number of integer-coordinate points over a linear polyhedron defined by $2^{|\Theta|}$ constraints. It must also be noticed that B_0 corresponds to the enumeration of the points with integer coordinates of the smallest hyperrectangle that contains $\mathcal{S}$ and is exact when the definition of $\mathcal{S}$ involves no diagonal constraints, *i.e.*, when the number of tasks is not greater than the number of processors.

B. Computation of B_1

To count the number of states in $\mathcal{S}$, we rely on a fixed point computation. We start from state $\mathbf{0}$ and date qH. We expand the set of states time unit per time unit. Each time unit, we add states that have a cumulative backlog that fits in this extra time unit while taking into account platforms and tasks constraints. We stop once we have reached a fixed point over the set of states. We first describe the algorithm, then prove its termination, soundness, completeness, and apply it to a simple example.

1) One time unit mappings: Let us consider $\mathbf{Act} : \mathbb{N} \to \{0,1\}^N$ such that $\forall t \in [0, \beta^{max}). \forall i \in [1,N]. \mathbf{Act}(t)[i] = 0$ iff $t \leq \beta_i^{max}$, and $\mathbf{Act}(t)[i] = 1$ otherwise. That is to say $\mathbf{Act}(t)[i] = 1$ iff a job of τ_i activated before t has not necessarily reached its deadline at date t.

Let $Incr = \{\mathbf{v} \mid \mathbf{v} \in \{0,1\}^N \wedge \sum_{i=1}^N \mathbf{v}[i] \leq m\}$. An element of $Incr$ is a mapping of tasks to processors (remember that jobs of the same task must execute sequentially). As an example, for $N = 3$ tasks and $m = 2$ processors we have:

$$Incr = \left\{ \begin{bmatrix}0\\0\\0\end{bmatrix}, \begin{bmatrix}0\\0\\1\end{bmatrix}, \begin{bmatrix}0\\1\\0\end{bmatrix}, \begin{bmatrix}1\\0\\0\end{bmatrix}, \begin{bmatrix}1\\1\\0\end{bmatrix}, \begin{bmatrix}0\\1\\1\end{bmatrix}, \begin{bmatrix}1\\0\\1\end{bmatrix} \right\}$$

Let $incr$ be a function from date to parts of $Incr$ such that $incr(t) = \{\mathbf{v} \mid \mathbf{v} \in Incr \wedge \mathbf{v} \times \mathbf{Act}(t) = \mathbf{v}\}$. $incr(t)$ describes the mappings of tasks to processors for $[t, t+1)$ in any feasible schedule, discarding those which execute a job of a task that has already missed its deadline. For example, assuming $N = 3$ tasks, $m = 2$ processors, and $\mathbf{Act}(1) = \begin{bmatrix}0\\0\\1\end{bmatrix}$, then we have:

$$incr(1) = \left\{ \begin{bmatrix}0\\0\\0\end{bmatrix}, \begin{bmatrix}0\\0\\1\end{bmatrix} \right\}$$

Lemma 1: A one time unit mapping of tasks onto processor $\mathbf{inc}$ for interval $[qH + t, qH + t + 1)$ for any non negative integer q is part of a feasible schedule if and only if $\mathbf{inc} \in incr(t)$.

Proof: Follows from the definition of $incr$. ■

Definition 5 (Possible mapping): A possible mapping is a one time unit mapping of tasks onto processor $\mathbf{inc} \in incr(t)$.

2) Fixed point algorithm: Let S be a set of states. We define the successor of S by elapsing one time unit from date $qH + t$ to $qH + t + 1$ as follows:

$$next(S, t) = \{\mathbf{s} + \mathbf{inc} \mid \mathbf{s} \in S \wedge \mathbf{inc} \in incr(t)\} \quad (4)$$

Given this definition of $next$, the set of possible states of the system at the end of a hyperperiod in any feasible schedule is the smallest fixed point of:

$$\begin{cases} S_0 = \{\mathbf{0}\} \\ S_{n+1} = S_n \cup next(S_n, n) \end{cases} \quad (5)$$

The resulting bound B_1 can be computed with algorithm in Figure 2 below.

```
Algorithm 1 Computation of $B_1$
Require: $\Theta, m$
Ensure: $B_1$
  $S \leftarrow \mathbf{0}$
  $t \leftarrow 0$
  while $next(S, t) \not\subseteq S$ do
      $S \leftarrow S \cup next(S, t)$
      $t \leftarrow t + 1$
  end while
  $B_1 = H \times |S|$
  return $B_1$
```

Figure 2. Fixed point algorithm for the computation of the exact bound B_1.

3) Termination: Recall that $\beta^{max} = \max_{\tau_i \in \Theta}\{\beta_i\}$ is the greatest possible backlog of any task at the end of a hyperperiod. From the definition of function $incr$, we have $incr(\beta^{max}) = \{\mathbf{0}\}$ and then the smallest fixed point is met at worst in β^{max} steps. During the computation of B_1 each state of $\mathcal{S}$ has to be stored. The number of states is upper bounded by the B_0. During the computation of B_1, each new state has to be compared to the set of states already explored. Hence our algorithm has also a factorial complexity in the state space size. A more detailed analysis, including a complexity analysis of the problem is out of the scope of this paper.

4) Soundness and Completeness :

Theorem 1 (Completeness and Soundness): $\mathbf{s} \in \mathcal{S}$ if and only if $\mathbf{s}$ is reachable by a feasible schedule from $\mathbf{0}$.

Proof: Soundness. *Ab absurdo*. Assume that there exists a state $\mathbf{s} \in \mathcal{S}$ which is not reachable by a feasible schedule from $\mathbf{0}$. Then, there exists $t \in [0, \beta^{max})$, a state $\mathbf{s}_t \in S_t$ that is reachable through possible mappings from $\mathbf{0}$ and a state $\mathbf{s}_{t+1} \in S_{t+1}$ that is not reachable through possible mappings from $\mathbf{0}$ such that $\mathbf{s}_{t+1} \in next(\{\mathbf{s}_t\}, t)$. Then, there exists $\mathbf{inc} \in incr(t)$ such that $\mathbf{s}_{t+1} = \mathbf{s}_t + \mathbf{inc}$ whereas $\mathbf{inc}$ is not possible at date t contradicting Lemma 1.

Completeness. *Ab absurdo*. Assume that there exists a state $\mathbf{s}$ which is reachable through possible mappings from $\mathbf{0}$ and such that $\mathbf{s} \notin \mathcal{S}$. Then, there exists $t \in [0, \beta^{max})$ and a state $\mathbf{s}_{t+1}$ that is reachable by a possible mapping from $\mathbf{s}_t \in S_t$ such that $\mathbf{s}_{t+1} \notin next(\{\mathbf{s}_t\}, t)$. Then, there exists $\mathbf{inc}$ such that $\mathbf{s}_{k+1} = \mathbf{s}_k + \mathbf{inc}$ and $\mathbf{inc} \notin incr(t)$ whereas $\mathbf{inc}$ is possible at date t contradicting Lemma 1. ■

5) Example: Consider a system with $N = 3$ tasks running on a platform with $m = 2$ processors. The charactetistics of the tasks are such that $\begin{array}{c}\beta_1\\\beta_2\\\beta_3\end{array} = \begin{bmatrix}1\\1\\3\end{bmatrix}$. The possible mappings of tasks to processors in the first time unit after a hyperperiod is given by:

$$incr(0) = Incr = \left\{ \begin{bmatrix}0\\0\\0\end{bmatrix}, \begin{bmatrix}0\\0\\1\end{bmatrix}, \begin{bmatrix}0\\1\\0\end{bmatrix}, \begin{bmatrix}1\\0\\0\end{bmatrix}, \begin{bmatrix}1\\1\\0\end{bmatrix}, \begin{bmatrix}0\\1\\1\end{bmatrix}, \begin{bmatrix}1\\0\\1\end{bmatrix} \right\}$$

and possible mappings in the following time units are given by:

$incr(1) = incr(2) = \left\{ \begin{bmatrix}0\\0\\0\end{bmatrix}, \begin{bmatrix}0\\0\\1\end{bmatrix} \right\}$ and $incr(3) = \left\{ \begin{bmatrix}0\\0\\0\end{bmatrix} \right\}$

Now, let us compute the smallest fixed point.

$$S_0 = \left\{ \begin{bmatrix}0\\0\\0\end{bmatrix} \right\}, \quad S_1 = \left\{ \begin{bmatrix}0\\0\\0\end{bmatrix}, \begin{bmatrix}0\\0\\1\end{bmatrix}, \begin{bmatrix}0\\1\\0\end{bmatrix}, \begin{bmatrix}1\\0\\0\end{bmatrix}, \begin{bmatrix}1\\1\\0\end{bmatrix}, \begin{bmatrix}0\\1\\1\end{bmatrix}, \begin{bmatrix}1\\0\\1\end{bmatrix} \right\},$$

$$S_2 = \left\{ \begin{bmatrix}0\\0\\0\end{bmatrix}, \begin{bmatrix}0\\0\\1\end{bmatrix}, \begin{bmatrix}0\\1\\0\end{bmatrix}, \begin{bmatrix}1\\0\\0\end{bmatrix}, \begin{bmatrix}1\\1\\0\end{bmatrix}, \begin{bmatrix}0\\1\\1\end{bmatrix}, \begin{bmatrix}1\\0\\1\end{bmatrix}, \begin{bmatrix}0\\0\\2\end{bmatrix}, \begin{bmatrix}1\\1\\1\end{bmatrix}, \begin{bmatrix}0\\1\\2\end{bmatrix}, \begin{bmatrix}1\\0\\2\end{bmatrix} \right\}$$

and then

$$\mathcal{S} = S_3 = \left\{ \begin{bmatrix}0\\0\\0\end{bmatrix}, \begin{bmatrix}0\\0\\1\end{bmatrix}, \begin{bmatrix}0\\1\\0\end{bmatrix}, \begin{bmatrix}1\\0\\0\end{bmatrix}, \begin{bmatrix}1\\1\\0\end{bmatrix}, \begin{bmatrix}0\\1\\1\end{bmatrix}, \begin{bmatrix}1\\0\\1\end{bmatrix}, \begin{bmatrix}0\\0\\2\end{bmatrix}, \begin{bmatrix}1\\1\\1\end{bmatrix}, \begin{bmatrix}0\\1\\2\end{bmatrix}, \right.$$
$$\left. \begin{bmatrix}1\\0\\2\end{bmatrix}, \begin{bmatrix}0\\0\\3\end{bmatrix}, \begin{bmatrix}1\\1\\2\end{bmatrix}, \begin{bmatrix}0\\1\\3\end{bmatrix}, \begin{bmatrix}1\\0\\3\end{bmatrix} \right\}$$

We obtain $B_1 = H \times |\mathcal{S}| = 15H$. With the same system, we have $B_0 = H \times (2 \times 2 \times 4) = 16H$. The state that was discarded in B_1 is $\begin{bmatrix}1\\1\\3\end{bmatrix}$ because it requires 5 time units of computation but a valid schedule cannot have more than $\max_2\{1, 1, 3\} = 4$ time units of pending work.

V. Experimentation

A. Setup

The computation of B_1 has factorial time complexity so it does not scale to big systems. Its main interest is to provide a reference to assess the tightness of approximate bounds. In particular, it is worth asking when B_0 is a reasonable

TABLE I. DETAILS AND PARAMETERS FOR THE 5 SERIES OF EXPERIMENTS

Series	N	m	β^{max}	Watchdog expiry time	Timeout (%)
1	$1 \leq N \leq 12$	$1 \leq m \leq 8$	20	15 min for $N \leq 9$ and 20 min for $N > 9$	18.47
2	$1 \leq N \leq 12$, 11 excluded	$1 \leq m \leq 8$	10	15 min for $N \leq 10$ and 45 min when N is 12	8.92
3	$1 \leq N \leq 12$	$1 \leq m \leq 8$	5	15 min	7
4	$k \leq N \leq 9$ with $k = 2 \times m$	$1 \leq m \leq 4$	$(10 - N) \times 8$	10 min	0.75
5	16	4	$2 \leq \beta^{max} \leq 6$	10 min	29

approximation, and if it is worth searching for less pessimistic approximations for certain systems. Thus, in this section, we evaluate the pessimism of bound B_0 with respect to B_1. We also provide some results concerning the resource consumptions of the computation of B_1 to characterize the range of systems that it can solve.

We implemented algorithm 2 in C, using red-black trees as the data structure for state sets. Computations have been run on a Debian GNU/Linux 8.8 system (kernel 3.16.0-4) with Intel(R) Xeon(R) CPU E5-2620 @ 2.00GHz and 128GB RAM. The evaluation set is based on five series of experiments. In each case, we take 20 samples per point (a point is defined by a number of tasks, a number of processors, and a value for β^{max}), and the algorithm is applied to each point. For each sample, the maximum backlog of each task is randomly generated between 1 and β^{max} using a uniform distribution. The code is instrumented to report execution time and maximum memory consumption of the computation of B_1. Lastly, a watchdog is used to stop the computation of B_1 after a pre-defined amount of time. Parameter values for each series are shown in Table I.

We provide a set of graphs that have been chosen to be as representative as possible of the data set. For each point, we represent the arithmetic average as well as minimum and maximum values among all 20 samples. In each figure the *y-axis* represents the ratio B_1/B_0 as a percentage. The quantity associated with the *x-axis* varies so it is specified in each figure.

B. *Pessimism of B_0*

Figure 3 shows the result when the number of tasks increases for a given number of processors. Figure 4 shows the result when the number of processors increases for a given number of tasks. As expected, both figures show that when the number of diagonal constraints increases, B_0 becomes more pessimistic. From 2, diagonal constraints appear for sets $\Lambda \subseteq \Theta$ such that $|\Lambda| > m$, *i.e.*, when the platform does not offer enough parallelism. Thus, the number of diagonal constraints increases with $\frac{N}{m}$. Figure 5 plots $\frac{B_1}{B_0}$ against $\frac{N}{m}$. It shows that, on this data series, B_0 quickly becomes a loose approximation of B_1: when $\frac{N}{m}$ becomes greater than 3, $\frac{B_1}{B_0}$ falls to 50 %, and below for higher values of $\frac{N}{m}$. The complexity of the computation of B_1 does not allow us to extend the plot further but it is expected that, as the number of linear constraint increases, $\frac{B_1}{B_0}$ asymptotically tends to zero.

Figure 7 plots $\frac{B_1}{B_0}$ against the standard deviation computed over the list $[\beta_i^{max} \mid \tau_i \in \Theta]$. Although it is not as clear as the impact of $\frac{N}{m}$, it shows that when the standard deviation is small, B_0 tends to be more pessimistic. Figure 7 also shows that similar values of $\frac{B_1}{B_0}$ can be reached for different values of β^{max} with similar dispersion of values of β_i^{max}. An

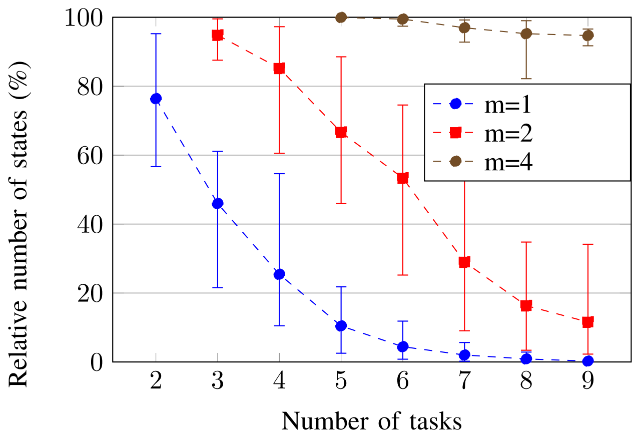


Figure 3. $N = 1$ to 9 tasks, $m = 1$ to 4 processors, $\beta^{max} = 20$.

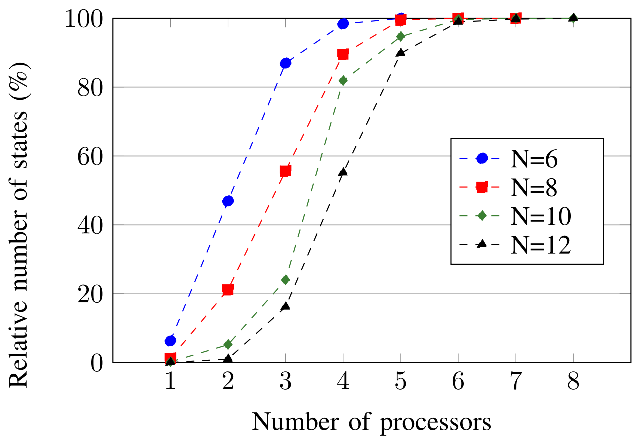


Figure 4. $N \in \{6, 8, 10, 12\}$ tasks, $m = 1$ to 8 processors, $\beta^{max} = 10$.

intuitive interpretation can be formulated from the example of figure 1: if $\beta_1^{max} = \beta_2^{max}$ then the diagonal constraints $\beta_1 + \beta_2 \leq \max\{beta_1^{max}, \beta_2^{max}\}$ removes half of the points of B_0 and this is the worst case. So, the closer the values of β_i^{max} in numerous mismatch, the more states it removes. And of course, a small standard deviation denotes a system with a small dispersion of β_i^{max} values.

C. *Scalability of algorithm 2*

The computation of B_1 requires a factorial number of comparisons with regards to the size of the state space of the system. Thus, it is sensible to every parameter that has an impact on the state space: number of tasks N, of processors m, and the maximum backlogs of tasks β_i^{max}.

Table II groups results for $N = 16$ and $m = 4$. In this case, the average execution time increases from 6.25 s to 385 s

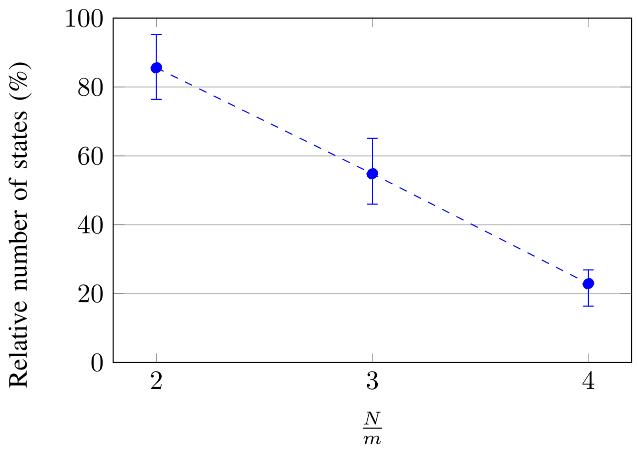


Figure 5. $N = 1$ to 12 tasks, $m = 1$ to 4 processors, $\beta^{max} = 20$.

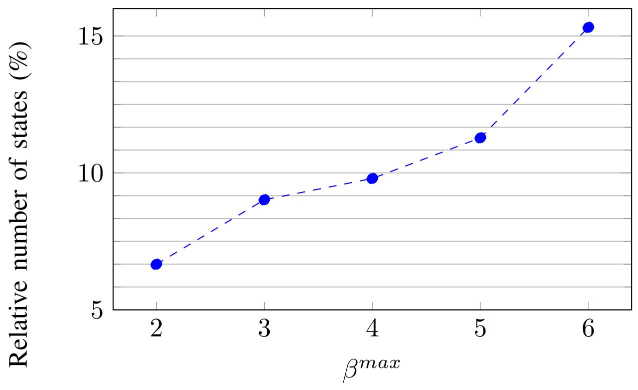


Figure 6. $N = 16$ tasks, $m = 4$ processors, $\beta^{max} = 2$ to 6.

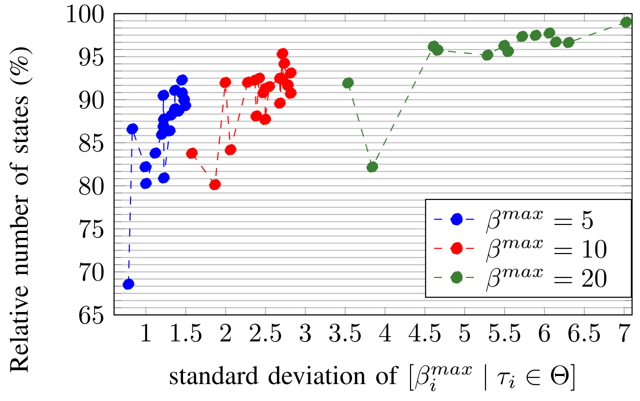


Figure 7. $N = 8$ tasks, $m = 4$ processors, $\beta^{max} \in \{5, 10, 20\}$.

TABLE II. EXECUTION TIME (IN SECONDS), WHEN $N = 16$ AND $m = 4$

β^{max}	Average	Maximum	Minimum	Standard deviation
3	6.25	22	microseconds	6.146244039
4	124.68	364	2	143.8439944
5	385	599	94	169.7838733

TABLE III. EXECUTION TIME (IN SECONDS), WHEN $m = 4$ AND $\beta^{max} = 20$

N	Average	Maximum	Minimum	Standard deviation
6	5.95	33	microseconds	9.827324956
7	124.2	511	1	136.6607786
8	254.54	701	18	302.4903777

just by increasing β^{max} from 3 to 5. Table III groups results for $m = 4$ and $\beta^{max} = 20$. In this case, the average execution time increases from 5.95 s to 254.54 s just by increasing N from 6 to 8. Lastly, Table IV groups results for $N = 8$ and $\beta^{max} = 20$. In this case, the average time increases from 1.05 s to 130.4 s just by increasing the m from 1 to 3. From these three tables, the influence of the individual β_i^{max} values on the overall execution time can also be seen: in Table II for example, with $N = 16$, $m = 4$ and $\beta^{max} = 5$, the execution time varies from 94 s to 599 s.

Similar results are observed for memory occupation. Indeed, the whole state space has to be stored. Table V shows for instance the maximum memory occupation for varying values of N and m when $\beta^{max} = 20$. As expected, the time and space complexity of algorithm 2 makes it impossible to deal with systems that have too large a state space. Nevertheless, many industrial systems use small multicore platforms. For instance, the 32 bit Microcontroller TriCore family developed by Infineon for the embedded automotive market offers platforms with 1 to 6 cores. Moreover, not all tasks in these systems have a non null backlog at hyperperiod boundaries, so B_1 could be of practical use for these systems. Additional experiments on industrial benchmarks are required to provide an answer to this question and it is out of the scope of this paper.

VI. Conclusion

The problem addressed in this paper is to compute an exact bound on the simulation interval for systems of asynchronous periodic tasks with arbitrary deadlines subject to structural

TABLE IV. EXECUTION TIME (IN SECONDS), WHEN $N = 8$ AND $\beta^{max} = 20$

m	Average	Maximum	Minimum	Standard deviation
1	1.05	6	microseconds	1.637552731
2	98	343	4	95.8200067
3	130.4	899	1	361.2071865

TABLE V. RESIDENT SET SIZE USED (IN MB), WHEN $\beta^{max} = 20$

N	m	Average	Maximum	Minimum	Std dev.
5	1	4.056	5.492	3.980	0.3380934782
6	1	13.935	47.980	3.988	13.92257703
7	1	108.555	640.840	3.812	166.1081405
8	1	1398.797	8286.392	66.756	2350.883606
9	1	15832.102	103894.004	640.600	28790.4245
5	2	8.250	19.372	3.980	5.43988676
6	2	35.805	168.132	3.988	39.78900124
7	2	238.146	940.680	10.796	229.4609528
8	2	3027.923	10032.492	147.412	3030.038648
9	2	27974.981	95923.060	1027.404	15778.69099

constraints scheduled by any deterministic and memoryless algorithm on a uniform multiprocessor platform. A very simple yet pessimistic solution for this problem is already known in the state-of-the-art. We formulate a characterization of the bound that involves the cardinal of the set of points with integer coordinates in a polyhedron defined by an exponential number of linear constraints. We propose and prove a fixed point algorithm to compute this set, which has factorial time complexity.

We rely on an implementation of this algorithm to estimate the pessimism of the bound known from the state-of-the-art through a set of experiments on synthetic systems. In the results of these experiments we observe two points: (i) the bound from the state-of-the-art quickly becomes a loose estimation of the exact bound when the number of tasks becomes greater than the number of processors of the platform; (ii) the time complexity of our algorithm is too high to deal with anything but small systems. From these two points, we conclude that there is an interest in looking at approximate bounds that lie in the middle between the state-of-the-art and the exact bound. Our formulation of the problem as a linear system already gives us a direction. The state-of-the-art provides a simple but pessimistic solution by discarding all diagonal constraints, while the exact bound does the opposite. So, as a direct follow-up to the work described here, we will now explore the idea to take into account a subset of the diagonal constraints to find a good trade-off between precision and time complexity.

REFERENCES

[1] R. I. Davis, A. Zabos, and A. Burns, “Efficient exact schedulability tests for fixed priority real-time systems,” IEEE Transactions on Computers, vol. 57, no. 9, 2008, pp. 1261–1276.

[2] J. Goossens, E. Grolleau, and L. Cucu-Grosjean, “Periodicity of real-time schedules for dependent periodic tasks on identical multiprocessor platforms,” Real-Time Syst, vol. 52, no. 6, 2016, pp. 808–832.

[3] J. Leung and J. Merrill, “A note on preemptive scheduling of periodic, real-time tasks,” Information Processing Letters, vol. 11, no. 3, 1980, p. 115–118.

[4] J. Goossens and R. Devillers, “Feasibility intervals for the deadline driven scheduler with arbitrary deadlines,” in Proceedings Sixth International Conference on Real-Time Computing Systems and Applications. RTCSA’99 (Cat. No.PR00306), 1999, pp. 54–61.

[5] L. Cucu and J. Goossens, “Feasibility intervals for multiprocessor fixed-priority scheduling of arbitrary deadline periodic systems,” in 2007 Design, Automation Test in Europe Conference Exhibition, 2007, pp. 1–6.

[6] J. Baro, F. Boniol, M. Cordovilla, E. Noulard, and C. Pagetti, “Off-line (optimal) multiprocessor scheduling of dependent periodic tasks,” in Proceedings of the 27th annual ACM symposium on applied computing (SAC), 2012, pp. 1815–1820.

[7] V. Nélis, P. Yomsi, and J. Goossens, “Feasibility intervals for homogeneous multicores, asynchronous periodic tasks, and fjp schedulers,” in Proceedings of the 21st international conference on real-time networks and systems, 2013, pp. 277–286.

An Overview of Cloud-Native Networks Design and Testing

Zhaobo Zhang, Xinli Gu
Silicon Valley Network Technology Lab.
Futurewei Technologies Inc.
Santa Clara, CA, USA
e-mail: zzhang1@futurewei.com, xgu@futurewei.com

***Abstract*—Cloud-native patterns have reshaped application development over the past decade. With the benefits of agility, resiliency, and scalability, the network domain starts embracing the cloud-native patterns to accelerate its evolution. Containerization becomes another solution of network function virtualization. Leveraging existing network services and the mature container orchestration platform, cloud-native networks attract wide attention, however the performance and scalability challenges in design and testing arise as the architecture advances. This paper presents an overview of cloud-native networks, the design and testing challenges and the development activities from open-source communities towards overcoming those issues. Performance optimization and hardware and software co-design are critical for the future success of cloud-native networks.**

Keywords—Cloud-native; Cloud-native network functions; container; Kubernetes; continuous testing; performance testing.

I. INTRODUCTION

Since Amazon first launched cloud computing platforms, delivering compute and storage resources through the Internet in 2006, on-demand and scalable cloud infrastructure has overwhelmingly reshaped the development of software and business [1]. Application architecture shifts from monoliths to microservices. Combining microservices with containerization and Continuous Integration and Continuous Delivery (CI/CD), the cloud-native concept emerged around 2010. As one of the pioneers, Netflix redesigned their systems in a cloud-native way and migrated all the services and data to the cloud through a seven-year journey, which facilitates rapid product release, new resource-hungry features and ever-growing volumes of data [2].

With proven success, cloud-native becomes a modern way of developing software. In 2015, Cloud Native Computing Foundation (CNCF) [3], a Linux Foundation project, was founded to advance container technology and align industry practice around its evolution. Since then, the cloud-native technologies and tools have thrived and taken great strides. Kubernetes [4], a container orchestration platform for automated container deployment, scaling and management is the first CNCF project. The plugin-based design and high extensibility build its success and make it to be the most adopted container orchestration system. Along with orchestration, a configurable infrastructure layer called service mesh is designed to ensure the security, resiliency, and observability of the communications between services. These two key components pave the way for container deployment and runtime management and significantly accelerate the cloud-native patterns adoption. In addition, CNCF launched many other projects covering different perspectives, including continuous integration and delivery, container runtime, cloud-native network, etc.

In the 5G and cloud era, communication service providers seek solutions to advance networks to meet ever-changing customer needs, optimize network utilization, and support new application scenarios, e.g., augmented reality, virtual reality, Internet of things. Cloud-native principles are meant to increase the velocity of the business. With API enabled design, CI/CD and Development and Operations (DevOps) practices, the cloud-native technologies improve the service agility and time-to-market. Therefore, network equipment vendors and communication service providers start adopting cloud-native architecture, containerizing network functions, more importantly, leveraging open-source cloud-native tools to modernize networks, e.g., orchestration, automation, monitoring. Together with application, network development joins the cloud-native journey. Milestones are illustrated in Figure 1.

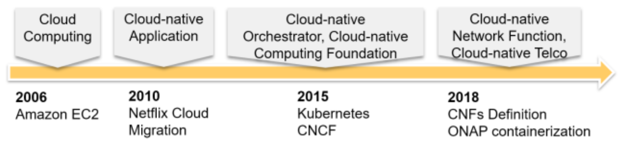

Cloud Computing
Cloud-native Application
Cloud-native Orchestrator, Cloud-native Computing Foundation
Cloud-native Network Function, Cloud-native Telco
2006 Amazon EC2
2010 Netflix Cloud Migration
2015 Kubernetes CNCF
2018 CNFs Definition ONAP containerization

Figure 1. Cloud-native Journey from Applications to Networks.

This paper aims to provide an overview of the current landscape of cloud-native networks, with focus on contributions from open-source communities. The definition and reference architecture of cloud-native networks are first introduced in Section II. The challenges and network specific requirements are discussed in Section III. Good design practice and guidance are summarized in Section IV. Testing flow and performance testing are presented in Section V. Conclusions are presented at the end.

II. CLOUD-NATIVE NETWORKS

Network architecture has evolved from individual physical machines for each Physical Network Functions (PNFs), to Virtual Network Functions (VNFs) running on VMware or OpenStack, to what the CNCF sees as the next wave of Cloud-native Network Functions (CNFs) running on Kubernetes. CNFs are like VNFs, but they run on lighter weight containers, simpler to upgrade, easier to secure, and cheaper to operate.

Cloud-native networks can be understood from two perspectives. The first is networks are built with cloud-native principles, which means network functions are containerized, with both control and data plane composed of microservices. The other is that networks are to provide connectivity and security to cloud-native applications in a cloud environment. Therefore, the network itself and the workloads that it serves both are considerably evolved. However, considering of the existing infrastructure, the compatibility with the PNFs and VNFs is needed in some scenarios.

As Kubernetes is the most adopted container orchestration platform, the cloud-native networks discussed in the remainder of the paper are in the context of Kubernetes. Kubernetes networking is based on a plugin model, which is open to third-party implementations. A network plugin needs to provide connectivity and reachability in pod networking. A pod is a group of containers that are deployed together on the same host in Kubernetes. Each pod has a unique and dynamic IP. All the pods in a cluster are connected through a flat network. Project Container Networking Interface (CNI) defines the standards on how network plugins should look, and how container runtime should invoke them [5]. It also provides a set of basic plugins as reference.

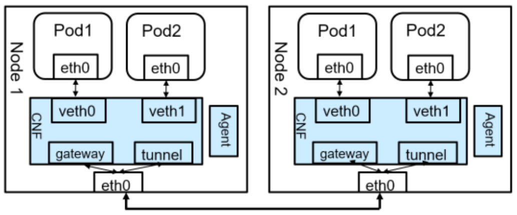


Figure 2. Container Networking Block Diagram.

In Figure 2, a simplified network block diagram is shown to illustrate the communication between pods across two diffident nodes. A CNF module first builds the connectivity between the pods and host network, and then it creates the overlay network between hosts based on different protocols, e.g., VXLAN or IPIP. This CNF can be implemented either in a kernel bypass manner to improve performance or with Linux kernel networking stack for the sake of simplicity. Together with this CNF, an agent pod is typically used for routes and network policy configuration. Project Flannel [6] and Calico [7] are the two commonly used CNI solutions.

III. NETWORK SPECIFIC DESIGN

Network workloads, responsible for low-level traffic forwarding, are different from the generic application workloads running in the cloud. A containerized network function may require multiple interfaces, faster data pipeline, comprehensive network policies, etc. In this section, the network specific requirements and solutions are discussed.

A. Multiple Networks Attachment

When Kubernetes initiate a pod, only one interface is created by default. In order to provide multiple interfaces, a CNCF network plumbing working group was formed, and a meta-plugin solution was proposed to create multiple network interfaces and manage multi-network policy. An illustration is shown in Figure 3. Compared to one standard CNI, multiple CNI plugins can be chained to form a meta-plugin. Then, multiple networks can be attached to a single pod. Project Multus [8] and CNI-Genie [9] provide reference implementations.

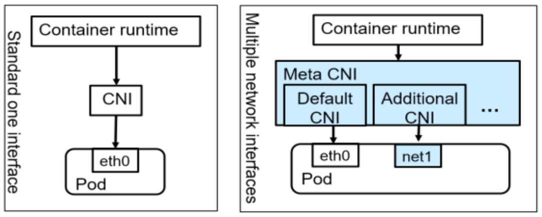


Figure 3. Standard vs Multiple Network Interfaces Attachment.

B. Host Networking Performance Improvement

CNI often leverages Linux host networking to implement network functions and policy. For example, iptables, a user-space utility program, is used to configure the IP packet filter rules. The filters are organized in different tables of chains to treat packets with specific rules. However, it becomes a bottleneck when large numbers of pods are under orchestration, since each host needs updates if any pod changes in the cluster.

An alternative of using iptables is implementing the function with extended Berkeley Packet Filter (eBPF), a Linux kernel technology, which compiles user programs to bytecode and attached to the kernel to be more performant [10]. eBPF enables the dynamic insertion of security, visibility, and networking control logic to the kernel. The flow is illustrated in Figure 4. The ability to run user-supplied programs inside the Linux kernel makes eBPF a powerful tool in terms of performance and convenience. Project Cilium is an eBPF-based CNI [11]. Detailed workflow and performance improvement can be found on Cilium's blog [12].

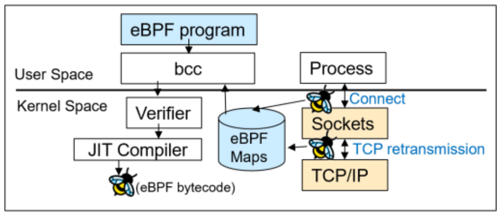


Figure 4. The Flow of eBPF Program Inserted to Linux Kernel.

C. Data Plane Acceleration

When a packet goes from user space to kernel space, an expensive copy occurs. To avoid the copy overhead, DPDK is widely used to process packets in user space and directly interact with network hardware bypassing the Linux kernel [13][14]. A data path comparison is shown in Figure 5. To further improve the performance, a high-performance virtual switch, e.g., Open vSwitch (OVS) can be added too. Project Antrea implemented OVS based CNI. With offloading the OVS function to supported Network Interface Card (NIC), the network bandwidth is increased by more than 3 times [15]. In order to provide high performance computing and networking in hyperscale data centers, hardware acceleration moves beyond CPUs and turn to dedicated chips [16]. Fortunately, Kubernetes provides a device plugin framework to allow specific hardware in the cluster, e.g., Graphic Processing Unit (GPU), NIC, which provides more possibilities for hardware acceleration.

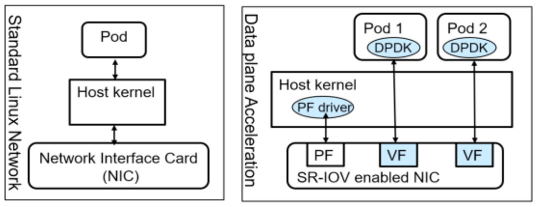


Figure 5. Data Plane Acceleration Bypass Linux Kernel.

D. Hybrid Multi-cloud Networking Orchestration

So far, the networks discussed above are intra-cluster networks, i.e., the communication is within the same cluster. However, as multi-cloud and hybrid cloud become more prevalent in today's business model, the inter-cluster network becomes a critical problem.

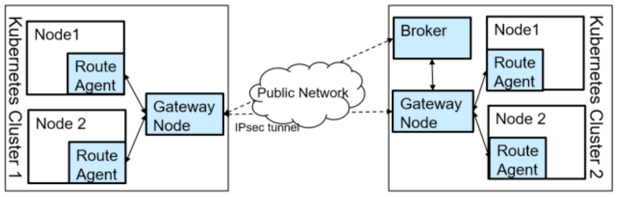


Figure 6. Multi-cluster Networking.

To connect two different clusters, traffic typically goes through the public Internet. An IPsec tunnel is often the choice to ensure secure communication. Figure 6 shows a simplified architecture of multi-cluster networking. A broker is used to exchange the information between clusters, and a gateway node is responsible for establishing IPsec tunnels and updating local cluster information to the central broker. Route agent runs on each node to configure the routes and rules. Project Submariner is a reference solution for this architecture [17]. For more comprehensive networking features, Project Network Service Mesh (NSM) [18] and Tungsten Fabric [19] can be referred.

IV. DESIGN PRINCIPLES

According to Sections II and III, the cloud-native networks typically consist of agents on each node to forward traffic and implement policies, a centralized control module to communicate with the container runtime and agents, and a data store to keep configurations and states. To design such a system with cloud-native principles, the following guidance is summarized from the best practices.

- *Modularization*

Each network function should be packed in its own container and orchestrated in a dynamic way. Complex network functions can be created by service function chaining. Service dependency can be programmed through a Helm chart in a unified format. Kubernetes style API is recommended to allow unified control.

- *State Separation*

Network functions should be separated to stateless and stateful, in order to scale the stateless functions smoothly. The states of stateful functions can be stored in etcd, a distributed key-value store in Kubernetes.

- *Infrastructure as code*

Network resources should be managed with machine-readable files. All the changes should be documented into files. Therefore, tasks like provision and roll back can be easily automated. Compared to the traditional management with command-line interface, automation removes the risk associated with human error and decreases system downtime.

- *Low-Level Acceleration*

Dedicated chips and hardware components are essential to build future intelligent cloud infrastructure [13]. With Kubernetes's device plugin feature, hardware functions can be exposed to containers for performance improvement. The design of hardware APIs should be consistent and reusable.

- *Built-in Observability and Analytics (AI ready)*

The observability of CNFs should be considered during the design phase, in order to enable continuous monitoring and automated troubleshooting. Output formats should be standardized and compatible with existing monitoring tools like Prometheus [20] and Grafana [21]. Thus, full-stack performance monitoring and analytics, from infrastructure to application, can be supported. In addition, structured data make artificial intelligence easy to apply and pave the way to autonomous network.

- *Platform Agnostic*

The network services should be able to be deployed and orchestrated seamlessly among public cloud, private cloud, and edge cloud. The CNFs should require no changes under different platforms.

V. TESTING METHODOLOGIES

Software testing today has been modernized by CI/CD and DevOps, two important characteristics of cloud-native patterns. Testing becomes a continuous activity in design, deployment, and operation. In this section, the generic test flow under CI/CD is first introduced, followed by performance testing. Lastly, the observability in CNFs is discussed.

CI is to establish a consistent and automated pipeline to build, package and test applications. With regularly checking new code, testing and integrating it with other parts of the system, organizations can reduce development and testing time from months down to days, even hours. Test suites are often written alongside new features. Unit tests ensure the committed code itself works. Integration tests ensure no breaks are introduced into the main code line. End-to-end tests ensure end user's experience by testing the entire product. Common CNF CI jobs provide the test coverage on command-line interface, authorization, storage, connectivity, network policy, etc. Security scan and compliance tests are typically included as well.

CD automates the software delivery process. It ensures the verified code changes from development environments can be pushed into production seamlessly. An interesting feature brought by CI/CD is canary testing, which releases the new version of the software only to a small percentage of users, to perform in-production test. New versions can be easily rolled back with Kubernetes orchestration.

Besides functional testing, performance testing is critical for cloud-native networks. The performance requirements of cloud-native networks are twofold. One is the performance of a CNF alone, e.g., how many packets a CNF can process per second. The other is the scalability of handling large amounts of network requests from web scale services, e.g., how fast the CNFs can provision one thousand endpoints or update network policies on thousands of hosts. In order to enable organizations to reliably test and compare performance between VNFs and CNFs, CNCF launched the CNF Testbed project in 2019.

The CNF Testbed project targets to build a repeatable test environment by using immutable hardware, version control on all configurations including underlay networking, and bootstrapping workload repeatably with automation pipeline [22]. The test framework typically includes a Kubernetes cluster with CNFs under test, traffic generator, and underlay networks illustrated in Figure 7. Traffic can be generated either within the cluster or from an external generator. The test steps are listed in TABLE I. The performance metrics evaluated often include CNF deployment time, endpoints provisioning time, network policy update time, idle-time CPU and memory usage, runtime CPU and memory usage, network throughput and latency.

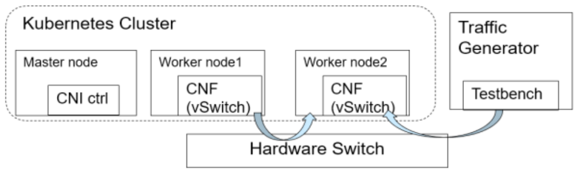


Figure 7. CNF Testbed Framework.

TABLE I. PERFORMANCE TESTING STEPS

1.	Provision hardware and Kubernetes cluster
2.	Deploy CNFs
3.	Deploy traffic generator
4.	Run the traffic benchmarks and tests
5.	Collect performance metrics

According to CNF Testbed's initial results, from VNFs to CNFs, the change will not affect the overall networking performance [22]. In fact, the lightness of container technology allows switching user context more quickly than with VM Hypervisors, and containerized workload could have a more direct interaction with underlying hardware. Communities are looking for more use cases to make more comprehensive comparison.

Since continuous testing and continuous monitoring become the norm today, the observability is critical. Observability includes tracing, metrics, and logs at various levels like cluster level, container level and kernel level. Kernel level tracing is particularly important for the CNFs. Standard Linux tracing tools like perf, ftrace, SysDig can be leveraged. To customize the network tracing, eBPF can be used to translate and load user programs to the kernel. Therefore, kernel networking events can be probed and monitored. Furthermore, the probes can be added into the CNFs program as well. Project IOVisor [23] implemented eBPF based monitoring tools, e.g., trace TCP passive and active connections, trace TCP packet drops with details, trace TCP retransmits. In a customized CNF, eBPF programs can be added to trace the changes of interface counters, interface address, routing tables and network address translation sessions, etc. It is an ongoing project to enrich eBPF-based monitoring tools. With more detailed and critical information extracted, fine-grained testing, fault isolation, and smart analytics are possible [24].

VI. CONCLUSIONS

Cloud-native principles and technologies bring tremendous benefits in terms of business agility, scalability and resiliency. Modern networks adopt this trend to accelerate development speed, improve resiliency with dynamic scaling and safe upgrades, and reduce costs. Kubernetes, a powerful production-grade orchestration platform with high extensibility, accelerates the process of network function containerization.

From PNFs to VNFs and CNFs, the implementation of network functions keeps evolving. There are advantages and issues for each paradigm. Although CNF brings many benefits, not all the workloads could fit perfectly for containers. Considering the performance advantages of network specific hardware, the data plane acceleration with hardware offloading cannot be neglect. This also brings new opportunities for next-generation hardware design. Network equipment and service providers could take a top-down approach, according to the requirements of containerized applications to do the hardware and software co-design, in order to achieve the optimal network solutions and meet the market needs in the cloud era.

REFERENCES

[1] R. Aljamal, A. El-Mousa and F. Jubair, "A User Perspective Overview of The Top Infrastructure as a Service and High Performance Computing Cloud Service Providers," IEEE Jordan International Joint Conference on Electrical Engineering and Information Technology, 2019, pp. 244-249.

[2] M. Villamizar et al., "Infrastructure Cost Comparison of Running Web Applications in the Cloud Using AWS Lambda and Monolithic and Microservice Architectures," 16th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, 2016, pp. 179-182.

[3] CNCF, https://www.cncf.io/, [retrieved Oct. 2020].

[4] Kubernetes, https://kubernetes.io/, [retrieved Oct. 2020].

[5] CNI, https://github.com/containernetworking/cni, [retrieved Oct. 2020].

[6] Flannel, https://github.com/coreos/flannel, [retrieved Oct. 2020].

[7] Calico, https://www.projectcalico.org/, [retrieved Oct. 2020].

[8] Multus, https://github.com/intel/multus-cni, [retrieved Oct. 2020].

[9] CNI-Genie, https://github.com/cni-genie/CNI-Genie, [retrieved Oct. 2020].

[10] S. Miano, M. Bertrone, F. Risso, M. Tumolo and M. V. Bernal, "Creating Complex Network Services with eBPF: Experience and Lessons Learned," IEEE 19th International Conference on High Performance Switching and Routing (HPSR), 2018, pp. 1-8.

[11] Cilium, https://cilium.io/, [retrieved Oct. 2020].

[12] Cilium Performace, https://cilium.io/blog/2020/10/09/cilium-in-alibaba-cloud, [retrieved Oct. 2020]

[13] N. Pitaev, M. Falkner, A. Leivadeas, and I. Lambadaris, “Characterizing the performance of concurrent virtualized network functions with OVS-DPDK, FD.IO VPP and SR-IOV”, in Proc. Of ACM International Conference on Performance Engineering, 2018, pp 285-292.

[14] L. Linguaglossa et al., "Survey of Performance Acceleration Techniques for Network Function Virtualization," in Proc. of the IEEE, vol. 107, no. 4, pp. 746-764, 2019.

[15] Antrea, https://antrea.io/, [retrieved Oct. 2020].

[16] D. He, Z. Wang and J. Liu, "A Survey to Predict the Trend of AI-able Server Evolution in the Cloud," in IEEE Access, vol. 6, pp. 10591-10602.

[17] Submarine, https://submariner.io/, [retrieved Oct. 2020].

[18] NSM, https://networkservicemesh.io/, [retrieved Oct. 2020].

[19] Tungsten Fabric, https://tungsten.io/, [retrieved Oct. 2020].

[20] Prometheus, https://prometheus.io/, [retrieved Oct. 2020].

[21] Grafana, https://grafana.com/, [retrieved Oct. 2020].

[22] CNF Testbed, https://github.com/cncf/cnf-testbed, [retrieved Oct. 2020].

[23] IOVisor, https://www.iovisor.org/, [retrieved Oct. 2020].

[24] C. Cassagnes, L. Trestioreanu, C. Joly and R. State, "The rise of eBPF for non-intrusive performance monitoring," IEEE/IFIP Network Operations and Management Symposium, 2020, pp. 1-7.