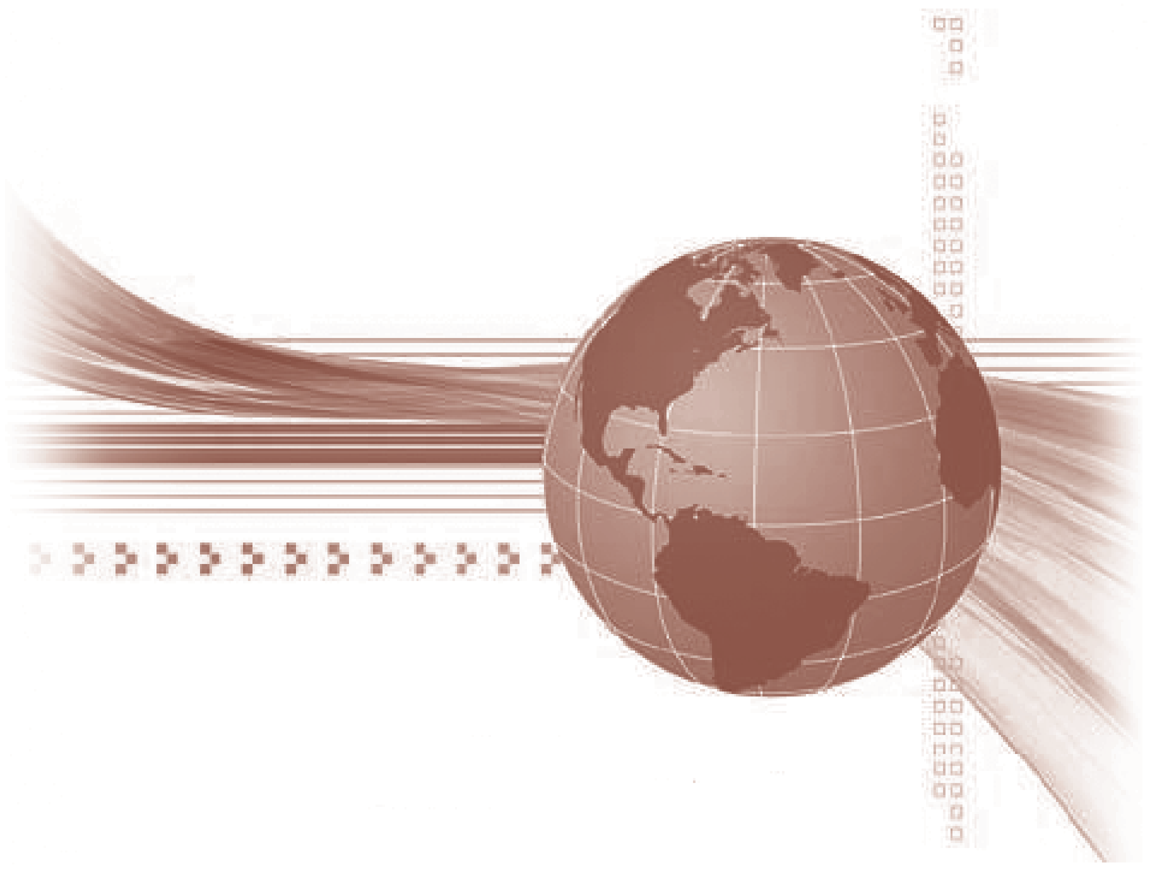
STUDIA UNIVERSITATIS
BABEŞ-BOLYAI



INFORMATICA

1/2019

STUDIA
UNIVERSITATIS BABEŞ-BOLYAI
INFORMATICA

No. 1/2019
January - June

YEAR **Volume 64 (LXIV) 2019**
MONTH **JUNE**
ISSUE **1**

S T U D I A
UNIVERSITATIS BABEȘ-BOLYAI
INFORMATICA

1

EDITORIAL OFFICE: M. Kogălniceanu 1 • 400084 Cluj-Napoca • Tel: 0264.405300

SUMAR – CONTENTS – SOMMAIRE

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LXIV**, Number 1, 2019
DOI: 10.24193/subbi.2019.1.01

TRIST: TREE RECOGNITION INTELLIGENT SYSTEM

LAURA ONAC

Abstract. Plant recognition represents a challenging computer vision problem due to the great variations of shape and texture among plant organs, within the same species. This paper proposes a light-weight, but reasonably deep Convolutional Neural Network architecture able to carry out this classification task. Multiple experiments were conducted with the proposed network architecture on the MEW2012 and Swedish leaf datasets. The experiments showed promising results, outperforming the current state-of-the-art systems that rely exclusively on a convolutional network for plant classification.

1. Introduction

Plant recognition has always been a challenging computer vision problem because of the great variations of shape and texture among plant organs, within the same species. This paper focuses on the classification of trees based on their leaves and describes a supervised deep learning technique, namely a Convolutional Neural Network (CNN), able to carry out this task.

The traditional approach to image pattern recognition has been to extract hand-crafted features from the images and then train a certain classifier with the resulting feature vectors. So, the performance of the systems employing this approach depends heavily on the underlying predefined features. In the plant identification literature, there are some common leaf features used for classification, such as circularity, eccentricity, roundness, aspect ratio [4]. However, the feature engineering process is quite complex and tedious, always needing extensive revision when changing the dataset.

But, over the past few years, Convolutional Neural Networks have become extremely popular in the field of computer vision. They are able to automatically detect important features such as shape or texture, without human

Received by the editors: April 26, 2018.
2010 *Mathematics Subject Classification.* 92B20.
1998 *CR Categories and Descriptors.* I.2.6 [**Computing Methodologies**]: Artificial Intelligence – *Learning*.
Key words and phrases. machine learning, computer vision, convolutional neural networks, image classification, leaf recognition.

assistance in the feature design process. Earlier studies showed that pure CNN approaches [1, 6, 7] seem more powerful than other classifiers trained with previously extracted features [2, 10, 13, 16, 18].

The CNN architecture described in this paper was tested first on the MNIST [11] benchmark dataset, and later on the MEW2012 [13] and Swedish [14] leaf datasets. The architecture showed promising results, which outperform the current state-of-the-art systems that rely exclusively on a convolutional network for plant classification.

Problem description and relevance. Plant recognition implies the use of leaves, flowers, bark or a combination between them, in order to identify the corresponding species. As a result, a lot of research papers were published on this topic, such as [1, 2, 4, 6, 7, 10, 12, 13, 14, 16, 17, 18]. Plant identification is also quite a difficult task for humans to accomplish because it requires domain-specific expertise, as well as experience. Nonetheless, recognizing plants, but particularly trees, holds great importance to mankind as they provide essential resources such as oxygen, food, medicine and wood. Many professions (e.g. environmental science, forestry, landscape architecture) involve the correct identification of trees and would greatly benefit from an automated system which can achieve this task.

2. Background

Convolutional neural networks [11] are a type of deep, feed-forward artificial neural networks that have proven to be greatly effective in image recognition tasks. They are able to recognize certain visual patterns directly from pixel images with minimal preprocessing. In order to ensure some degree of shift, scale and distortion invariance, convolutional networks employ a combination of three architectural elements: local receptive fields, shared weights and subsampling. They manage to incorporate knowledge about the invariances of 2D shapes by using local connection patterns, therefore forcing the extraction of local features.

Like any other feed-forward network, a CNN is composed of an input layer, one or more hidden layers, and an output layer. But, as opposed to ordinary networks which only contain fully-connected layers, CNNs incorporate some additional types, namely convolutional and pooling layers.

Each layer transforms a volume of activations into another, through a differentiable function. The last fully-connected layer is called the output layer and in classification settings it represents the class scores. So, the entire network transforms the input image, layer by layer, from the original pixel values to the final class scores.

The Convolutional layer (CONV) [9] is the core building block of a Convolutional network, as it does most of the computational heavy lifting. It computes the output of neurons that are connected to local regions in the input. The output represents the dot product between the neuron's weights and the corresponding elements in a small region from the input. This layer's parameters are actually a set of learnable, small 2-dimensional filters (or kernels). During the forward pass, each filter is slid (or convolved) across the width and height of the input and the dot products between the entries of the filter and the input at the corresponding positions is computed. The result will be a 2-dimensional activation map that gives the responses of that filter at every spatial position.

The size of the feature maps, which are the result of the convolution, is controlled by using two techniques: stride and zero-padding [9]. In this work, our goal is to preserve the size of the input image and we achieve this by combining the zero-padding with a stride equal to 1.

Pooling layers (POOL) [9] are periodically inserted in-between successive convolutional layers. Pooling is a form of non-linear down-sampling and these layers progressively reduce the spatial size of the representation. Thus, pooling layers reduce the amount of parameters and computation in the network. They most commonly use a *max* or *average* operation in order to resize the input. This layer does not have any trainable parameters.

Finally, after several Convolutional and Pooling layers, the high-level reasoning in the network relies on the Fully-connected layers (FC) [9]. The output from the Convolutional and Pooling layers represent high-level features of the input image. The purpose of the Fully-connected layer is to use these features to classify the input image into various classes based on the training dataset.

Dropout layers (DROP) [15] are used in order to avoid overfitting the training dataset. On those layers, units and their connections are randomly dropped during the training process.

Each neuron in the Convolutional and Fully-connected layers has an activation function, also known as a non-linearity, which is applied element-wise on the neuron's output. *Leaky ReLU* is such a function. It is very similar to the ReLU (Rectified Linear Unit) [5] activation function, but it attempts to fix the *dying ReLU* problem [8]. Instead of the function being zero when the input is negative, a leaky ReLU will instead have a small negative slope. This function is defined in Formula 1 [8]:

$$f(x) = max(\varepsilon x, x), \tag{1}$$

where x is the output of the neuron and ε is a small constant.

Another activation function, often used on the last Fully-connected layer of a network-based classifier is called Softmax [9]. The Softmax function squashes the outputs of each unit to be between 0 and 1, and it also divides each output such that the total sum of the outputs is equal to 1. So, the output of this function is equivalent to a categorical probability distribution. Softmax is defined in Formula 2 [9]:

$$\sigma(Z_j) = \frac{e^{Z_j}}{\sum\limits_{k=1}^{K} e^{Z_k}}, \tag{2}$$

where Z is the input vector, j is the index of the output unit and K is the number of output units.

Training a neural network essentially means using the training set to adjust the weights and biases in the network's layers, in order to improve its performance. The training process is an iterative one and it is composed of two major steps: forward propagation and backward propagation. The forward pass implies getting an image from the input layer, through the hidden layers and to the output layer. The obtained result is then compared to the expected output and, during the backward pass, the error for each unit is computed and its parameters are updated accordingly.

Improving performance usually means minimizing the loss function, which in this case is *cross entropy* [3]. Cross entropy indicates the distance between what the model believes the output distribution should be and what the original distribution really is. It is defined in Formula 3 [3]:

$$H(y,p) = -\sum_i y_i log(p_i), \tag{3}$$

where y is the expected vector and p is the predicted one.

3. Literature Review

Over the years, there have been several studies carried out on automatic plant recognition. Most of them perform the classification task based on the plants' leaves. As opposed to flowers, leaves are available over a longer period of time and in a much greater number. Also, leaves are more specific to a certain tree than bark is.

What follows is a presentation of other works in the literature that focus on plant recognition. For the purposes of this paper, the related studies will be divided into those that use purely CNNs and those that employ different classification methods.

The classification of plants using purely CNNs was only recently proposed by authors such as Jassman [7] in 2015, He et al. [6] and Atabay [1], both in 2016.

Jassman [7] aims to develop a system capable of classifying images with natural background. The five networks described in the paper are trained using a dataset created by the author, which contains 15 species of plants with 30 samples each. The best network obtained a top-1 accuracy of 56.66%.

He et al. [6] proposed four different CNN architectures trained with 20 species from the ICL dataset. They started with a CNN with two convolutional layers, two average pooling layers, followed by a fully-connected single-layer perceptron. The first variation introduced was dropout in the last layer, then they added a single connected layer between the second and third layer, and at last, they developed an advanced version of the single connected layer. With the fourth version, they achieved a precision of 91.90%.

Atabay [1] developed two convolutional networks with the same architecture, but different activation functions. For one, he used ReLU, and for the other, ELU. His networks were trained with the Flavia [18] and Swedish [14] leaf datasets, obtaining 97.24% and 99.11% mean average precision on the validation sets, respectively. Those results were achieved using ELU activation.

Traditional methods for leaf recognition generally imply the extraction of features to be used subsequently as input for a classifier. For the feature extraction phase, a variety of techniques were employed: Novotný & Suk [13] used Fourier descriptors, Sulc & Matas [16] developed a method using multi-scale histograms, Çuğu et al. [2] produced some hand-crafted features, Wu et al. [18] extracted digital morphological features, and Kumar et al. [10] used histograms of curvature over scale. The most popular classifiers are Support Vector Machines [16, 2] and the Nearest Neighbor algorithm [13, 10]. Çuğu et al. [2] obtained the final result by merging the results of the SVM with those of a CNN. Wu et al. [18] used a Probabilistic Neural Network as the classifier.

4. TRIST-Net: Methodology

The architecture of the Convolutional Neural Network is illustrated in Figure 1 and it is composed of 10 layers: $INPUT \rightarrow CONV1 \rightarrow POOL1 \rightarrow CONV2 \rightarrow POOL2 \rightarrow CONV3 \rightarrow POOL3 \rightarrow FC \rightarrow DROP \rightarrow OUTPUT$. This architecture was chosen due to the promising results obtained in experiments conducted first on the MNIST dataset [11], and later on the MEW2012 [13] and Swedish [14] datasets.

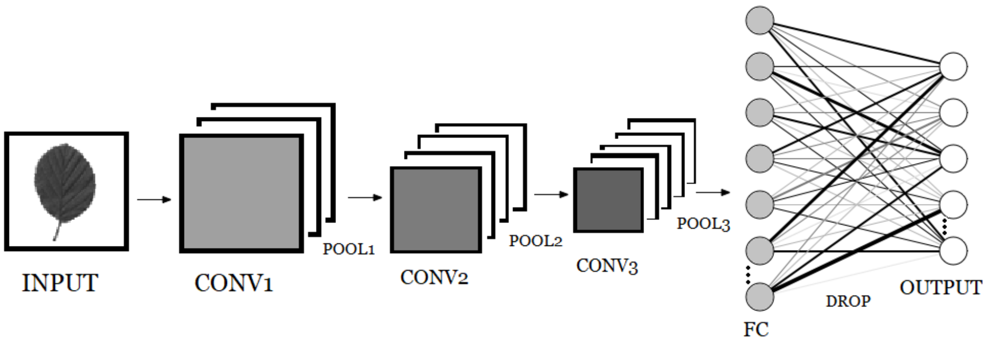


FIGURE 1. Proposed CNN architecture.

The INPUT layer refers to the grayscale input image. Every image has only one channel, so it is represented as a 2-dimensional matrix of pixel values. The matrix has a dimension of 64×64 and its elements are in the interval $[0, 1]$, where the value 0 indicates black and 1 indicates white.

The CONV1 layer has 8 filters, each being 5×5 square and uses *Leaky ReLU* as the activation function. The result of this layer is composed of 8 feature maps, each having the dimension 64×64.

The POOL1 layer applies Max Pooling with a 2×2 window and a stride equal to 2. The output of this layer is represented by 8 feature maps, each having the dimension 32×32.

The CONV2 layer has 16 5×5 filters with *Leaky ReLU* activation. The result of this layer is composed of 16 32×32 feature maps.

The POOL2 layer applies Max Pooling with a 2×2 window and a stride equal to 2. The output of this layer is represented by 16 16×16 feature maps.

The CONV3 layer has 32 5×5 filters with *Leaky ReLU* activation. The result of this layer is composed of 32 16×16 feature maps.

The POOL3 layer applies Max Pooling with a 2×2 window and a stride equal to 2. The output of this layer is represented by 32 8×8 feature maps.

The feature maps that resulted from the previous layer are flattened and concatenated, so they can serve as input for the FC layer.

The FC layer consists of 128 neurons, each having $8 \times 8 \times 32 = 2,048$ weights and 1 bias. Again, the *Leaky ReLU* activation function is used.

The DROP layer executes the dropout operation on the output of the FC layer, with 70% probability that an element will be kept.

The OUTPUT layer is actually a fully-connected layer, but with a different activation function: Softmax. It has one neuron for each possible class, and they are connected to every neuron from the DROP layer, so they each have 128 weights and 1 bias.

5. Experimental Results

5.1. **Data sets.** Before evaluating the performance of network on the leaf datasets, some tests were performed on the benchmark dataset MNIST [11]. MNIST is composed of images with handwritten digits and the version used for training contains a total of 1,797 samples.

The **MEW2012** (Middle European Woods 2012) [13] leaf dataset consists of images of leaves collected from trees and shrubs native or frequently cultivated in the Czech Republic. The 2012 version contains 153 species, each having between 50 and 99 samples, with a total of 9,745 images. Only 20 of those species were used for experiments. The species were randomly selected, with a total of 1,240 images.

The **Swedish** [14] leaf dataset contains 15 species of trees from Sweden, with 75 samples for each of them, giving a total of 1,125 images.

5.2. **Image pre-processing.** The original images from the datasets had to be modified prior to being used by the Convolutional Neural Network. The reason is that the architecture of the CNN requires input images to have the same dimension and for each pixel to have only one corresponding value, restrictions which the samples from the datasets do not uphold. So, before the training process could begin, each leaf image was converted to grayscale and re-dimensioned to a 64×64 size. Because both datasets are quite small for the methodology employed, they were augmented. For each image, three more were generated by flipping the image, rotating it 5 degrees to the right and rotating it 5 degrees to the left.

5.3. **Training.** The CNN presented in this paper is trained using stochastic gradient descent because there is a certain level of redundancy in the leaf datasets used. The training set is composed of 80% of the images from the whole set, randomly selected. The rest of the dataset is used for testing purposes.

Throughout the training process, as well as afterwards, the performance of the network was measured through loss, accuracy and precision. Ideally, after each iteration, an evaluation of the network would reveal a smaller loss value, and a greater accuracy and precision. This, of course, is not always true. Sometimes, those values end up in some local optimum and the performance of the network has to get a bit worse before it can get better.

5.4. **Experiments.** The network architecture presented was trained using, in turn, each of the datasets mentioned. In the case of the leaf datasets, the experiments were conducted using the original datasets and also using the augmented version of each dataset.

5.5. **Results and discussion.** The accuracy, precision and loss obtained on the test data with the proposed CNN architecture are reported in Table 1.

Dataset	**Classes**	**Samples**	**Accuracy**	**Precision**	**Loss**
MNIST	10	1,797	97.33%	97.38%	0.0861
MEW2012	20	1,240	84.89%	86.13%	0.4831
MEW2012 + aug	20	4,960	**92.16%**	**92.30%**	0.2790
Swedish	15	1,125	**96.00%**	**96.32%**	0.1824
Swedish + aug	15	4,500	94.77%	94.85%	0.1703

TABLE 1. Results obtained on the test data for MNIST, MEW2012 and Swedish datasets.

With a simpler benchmark dataset such as MNIST, the network performed very well, which is always the first sign of a good architecture. With MEW2012, the image augmentation improved the performance dramatically, which means that the extra samples generated helped the model to generalize better. On the other hand, with Swedish, the best performance was obtained without augmentation, but the difference between the model trained using the original dataset and the one trained using the augmented dataset is not substantial.

A comparison can be made between the best results obtained with the proposed approach and the ones from the published papers that also employ CNNs in order to perform similar leaf recognition tasks. He et al. [6] performed their experiments on the ICL dataset, which was not publicly available at the time this article was being written. Jassman [7] performed the experiments on a personal dataset, which was not made public. Consequently, the comparison with these papers is made based on the number of classes in the datasets and the performance measurements used.

The CNN architecture presented in this paper achieved a precision of 92.30% on MEW2012, while He et al. [6] achieved 91.90% on ICL. Also, with the presented approach, an accuracy of 96.00% was obtained on Swedish, while Jassman [7] only obtained 56.66% accuracy on their private dataset. So, the proposed CNN architecture achieves higher precision for 20 species than He et al. [6] and higher accuracy for 15 species than Jassman [7]. This indicates that the proposed network is better fit for this task, but the comparison is not conclusive due to the fact that the experiments couldn't be performed on the same datasets.

No clear comparison could be made with Atabay [1] because it is not understandable whether the performance measurement used is accuracy or mean average precision. Also, in their approach, the dataset is split in train (70%), validation (10%) and test (20%). But only the results on the validation set are presented, without providing the test results.

6. Conclusion

This paper presents a machine learning technique for leaf recognition. The chosen method is a Convolutional Neural Network due to the robustness and effectiveness of this kind of networks in image recognition tasks in general.

Experimental results with the MEW2012 [13] and Swedish [14] leaf datasets show that the proposed approach outperforms the other approaches that use CNNs for plant recognition in the literature.

During the development process, certain issues were encountered, such as overfitting, dying ReLU and numerical instabilities. But, by overcoming them, the designed model became much more reliable.

A more complex architecture might be able to recognize the species from the image without the leaf having to be placed on a white background. As the results obtained were promising, future work is under consideration in order to improve the current convolutional network implementation.

References

[1] H.E. Atabay. A convolutional neural network with a new architecture applied on leaf classification. *IIOAB Journal*, 7:326—-331, 2016.

[2] İ. Çuğu, E. Şener, Ç. Erciyes, B. Balcı, E. Akın, I. Önal, and A. O. Akyüz. Treelogy: A novel tree classifier utilizing deep and hand-crafted representations. *arXiv preprint arXiv:1701.08291*, 2017.

[3] P. Dahal. Classification and loss evaluation - softmax and cross entropy loss. https://deepnotes.io/softmax-crossentropy.

[4] C. Y. Gwo and C. H. Wei. Plant identification through images: Using feature extraction of key points on leaf contours. *Applications in plant sciences*, 1(11), 2013.

[5] R. H. R. Hahnloser, R. Sarpeshkar, M. A. Mahowald, R. J. Douglas, and H. S. Seung. Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit. *Nature*, 405(6789):947, 2000.

[6] X. He, G. Wang, X. P. Zhang, L. Shang, and Z. K. Huang. Leaf classification utilizing a convolutional neural network with a structure of single connected layer. In *International Conference on Intelligent Computing*, pages 332–340, 2016.

[7] T. J. Jassman. Mobile leaf classification application utilizing a convolutional neural network. Master's thesis, Appalachian State University, 2015.

[8] R. Kapur. Rohan #4: The vanishing gradient problem. https://ayearofai.com/rohan-4-the-vanishing-gradient-problem-ec68f76ffb9b.

[9] A. Karpathy. Cs231n convolutional neural networks for visual recognition. *Neural networks*, 1, 2016.

[10] N. Kumar, P. N. Belhumeur, A. Biswas, D. W. Jacobs, W. J. Kress, I. C. Lopez, and J.V. Soares. Leafsnap: A computer vision system for automatic plant species identification. In *Computer vision–ECCV 2012*, pages 502–516. Springer, 2012.

[11] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[12] M. E. Nilsback and A. Zisserman. *An automatic visual flora: segmentation and classification of flower images*. PhD thesis, Oxford University Oxford, 2009.

[13] P. Novotnỳ and T. Suk. Leaf recognition of woody species in central europe. *biosystems engineering*, 115(4):444–452, 2013.

[14] O. Söderkvist. Computer vision classification of leaves from swedish trees. Master's thesis, Linkoping University, 2001.

[15] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[16] M. Sulc and J. Matas. Texture-based leaf identification. In *European Conference on Computer Vision*, pages 185–200. Springer, 2014.

[17] A. Wendel, S. Sternig, and M. Godec. Automated identification of tree species from images of the bark, leaves and needles. In *16th Computer Vision Winter Workshop*, page 67. Citeseer, 2011.

[18] S. G. Wu, F. S. Bao, E. Y. Xu, Y. X. Wang, Y. F. Chang, and Q. L. Xiang. A leaf recognition algorithm for plant classification using probabilistic neural network. In *Signal Processing and Information Technology, 2007 IEEE International Symposium on*, pages 11–16. IEEE, 2007.

Department of Computer Science, Faculty of Mathematics and Computer Science, Babeş-Bolyai University, 1 Kogălniceanu, Cluj-Napoca, 400084, Romania
Email address: `olic1770@scs.ubbcluj.ro`

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LXIV**, Number 1, 2019
DOI: 10.24193/subbi.2019.1.02

BENCHMARKING COGNITIVE ABILITIES OF THE BRAIN WITH THE EVENT OF LOSING THE CHARACTER IN COMPUTER GAMES

NORBERT BÁTFAI, DÁVID PAPP, RENÁTÓ BESENCZI, GERGŐ BOGACSOVICS, DÁVID VERES

Abstract. Most computer game players have experienced the sensation of temporarily losing their character in a given gameplay situation when they cannot control the character, simply because they temporarily cannot see it. The main reasons for this sensation may be due to the interplay of the following factors: (1) the visual complexity of the game is unexpectedly increased compared with the previous time period as more and more game objects and effects are rendered on the display; (2) and/or the game is lagging; (3) and finally, it is also possible that the players don't have sufficient experience with controlling the character. This paper focuses on the first reason. We have developed a benchmark program which allows its user to experience the sensation of losing character. While the user can still control the character quite well, the benchmark program will increase the visual complexity of the display. Conversely, if the user loses the character then the program will decrease the complexity until the user finds the character again, and so on. The complexity is measured based on the number of changed pixels between two consecutive display images. Our measurements show that the average of bit per second values of losing and finding pairs describes the user well. The final goal of this research is to further develop our benchmark to a standard psychological test.

Received by the editors: December 22, 2018.

2010 *Mathematics Subject Classification.* 91E10.

1998 *CR Categories and Descriptors.* H.5.2 [**INFORMATION INTERFACES AND PRESENTATION**]: User Interfaces – *Benchmarking*; H.1.2 [**INFORMATION INTERFACES AND PRESENTATION**]: User/Machine Systems – *Software psychology*.

Key words and phrases. esport, talent search, benchmark program, visual complexity, losing character, psychology test.

This work was supported by the construction EFOP-3.6.3-VEKOP-16-2017-00002. The project was supported by the European Union, co-financed by the European Social Fund. The work/publication is supported by the GINOP-2.3.2-15-2016-00062 project.

1. Introduction

Losing the control of the character in a given gameplay situation is a very common sensation that is well known among gamers. In this situation, players cannot control their character, simply because they temporarily cannot see it due to one or more of the following reasons: the visual complexity of the display is unexpectedly increased, and/or the game is lagging and, finally, the player simply isn't experience enough to control the character. In this paper, we introduce our benchmark computer program called *BrainB Test Series 6* that can abstract this sensation of losing sight of a character. In this test, game objects are symbolized by boxes as shown in Fig. 1. All box movement is determined by random walks. There is a distinguished box labeled by the name *Samu Entropy*. It represents the character controlled by the player. The benchmark test lasts for 10 minutes. During the test, the user must continuously hold and drag the mouse button on the center of Samu Entropy. If the user succeeds in this task then the benchmark program will increase the visual complexity of the display. It will draw more and more overlapping boxes which will move faster and faster. Otherwise, if the mouse pointer cannot follow the center of Samu Entropy then the visual complexity will be decreased. The test will delete more and more boxes and the remaining boxes move slower and slower until the user finds Samu Entropy again, i.e., clicks on Samu Entropy. The BrainB Series 1 to 4 were developed in the family setting of the first author[1]. Then, in our university environment, we did a preliminary study [3] on the next version (BrainB Series 5). Some of its measurements were streamed live on Twitch[2]. The main research goal of this study is to show that players lose the character on a higher complexity level of the display and they find it on a relatively lower complexity level.

1.1. **Psychological Background.** The cognitive ability of attention is a significant factor in everyday life. The research of vigilance is an important topic in Psychology from 1970 to the present day. The first method used to measure vigilance was the Mackworth Clock [9]. Another method is the Toulouse-Piéron test [19], in which participants have to follow a given scheme to separate right and wrong signs. The Yerkes-Dodson law [21] says that for achieving the best performance there is an optimal arousal level, which level is higher in simpler tasks, and lower in complex activities. We must not forget that as in some other things, in the attentional system there are also personal differences that should be taken into consideration while researching the subject [6]. Witkin et al. did research on perception [20], and from this work, they created a theory

[1]For example, see `https://www.twitch.tv/videos/139186614`
[2]For example, see `https://www.twitch.tv/videos/206478952`

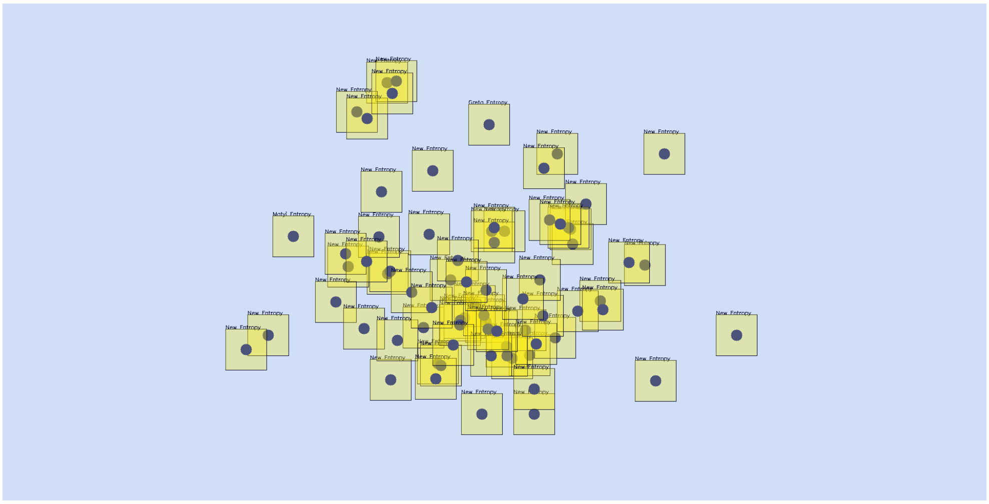


FIGURE 1. A screenshot of BrainB Test Series 6 in action. The greater the visual complexity of the screen, the greater the probability of losing the character.

about two different cognitive styles. Sagan wanted to calculate the information processing speed of the brain, to do so, he based his calculation on the example of looking at the moon, and from this example he drew the consequence, that the brain can process about 5000 bit/sec at its peak performance [17]. In a modern project, called Building 8, the main thought is to make the brain into a computer [11]. Based on this project, the information processing speed of the brain is about a terabyte/sec, which far exceeds the speed estimated by Sagan. These type of tests and experiments are common tools in the science of psychology [16], [15], [14]. Repeatedly performing the same experiment or test with the same participants could affect the results. Previously, as we specified, repeatedly using the same method could cause the lowering of its validity, and the results could be distorted. Participants can learn and adapt to certain methods, even if it just means a small percentage of difference. The current test takes 10 minutes to complete, in this 10 minutes the participant's full attention and concentration is needed. We should keep in mind, that the negative effects of fatigue could balance the positive effects of practice, in a direct way through the use of repeated examinations. It is therefore proposed to perform our benchmark test in a competitive environment trying to beat friends, family members, colleagues or ourselves.

1.2. **Informatics Background.** Since computer games have a relatively short history and their effects on cognitive skills have only recently started to be researched, there are plenty of questions to be answered. In [8], the authors reported an increase in executive functions in school students after playing computer games. Moisala et al. in [10] shows that enhancements in speed and performance accuracy of working memory tasks is related to daily gaming activity. In [5], authors present an analysis of the impact of action video games on cognitive skills. Using computer games to measure cognitive abilities has a short history, but a promising future. Most research try to measure the presence or severity of a certain cognitive disease such as dementia or Alzheimer's disease. In [1], authors show how a long-term use of video games can reduce the costs of multitasking in older adults. Geyer et al. in [7] show that the change of the score of an online game is in connection with the age-related changes in working memory. Seldom can we find applications that have been developed for the measurement of cognitive abilities. One such application is reported in [13] and [12]; it is a framework that has been developed to measure cognitive abilities and its change in the elderly with computer games. This framework is able to log and analyze scores achieved in various online computer games. From the viewpoint of information theory and HCI (Human-Computer Interaction), the Hick's law [18] could be an interesting aspect. This law states that the response time of the brain increases with the logarithm of the size of the input. For our purposes, it seems to present an interesting question: how can we apply the Hick's law (or other information theory figure) in our benchmark software?

1.3. **Losing The Character.** We have experienced the sensation of losing the character during playing several games[3]. Now we share our thoughts about the phenomenon of "losing the character" and give some examples to illustrate it from the game called League of Legends. As we head into the mid and late game, teams start fights more often with more people, even with all of them. This is what we call "teamfights". These are harder to handle, because a lot of things can appear on our screen at the same time including: the champions who participate in the fight, optionally minions or jungle monsters, and the visual effects of the spells, summoner spells, and the active or passive abilities of the items. Besides those effects, we see a lot of other things and we still have to make sure that we fulfill our ingame role properly: position well, attack the

[3]For example League of Legends, `https://na.leagueoflegends.com`, Clash of Clans, `http://supercell.com/en/games/clashofclans/`, Clash Royale, `http://supercell.com/en/games/clashroyale/`, Heroes of the Storm, `https://heroesofthestorm.com`, Dota 2, `https://www.dota2.com`, World of Warcraft, `https://worldofwarcraft.com` or Cabal, `http://cabal.playthisgame.com`.

proper target, or defend our teammates. We have to handle a lot of information at a blink of an eye, so it is completely natural, that sometimes we do not know where to look or what to do. Consequently, we can lose our own character, which can end with our death, or: we can lose the target character, and it can survive; or we can lose the character that we wanted to protect, thus an important member of the team can die. An example ingame footage can be viewed at `https://youtu.be/wdy3KUm1454`, starting at 2:12.

2. Brain Benchmarking Series 6

BrainB is a Qt C++ desktop application that uses the OpenCV library. It is developed as an open source project that is available on GitHub [2]. Its source can be built easily on GNU/Linux systems. But the latest (6.0.3) Windows binary release can also be downloaded as a ZIP file from `http://smartcity.inf.unideb.hu/~norbi/BrainBSeries6/`. The code snippet shown in Listing 1 is the heart of our benchmark program. It is a simplified version of the original source code that can be found in the GitHub repository[4]. This code is executed at every 100 milliseconds that is ten times per second. First, as shown in Line 1, it computes the distance between the mouse pointer and the center of the box of Samu Entropy and the result is stored in the variable called *dist* that holds the square of the Euclidean distance. If the distance is larger than 121 pixels (11 is the square root of 121) and if it reoccurs 12 consecutive times or more in a row (that means at least a time interval of 1.2 seconds) and it is also true that the player was controlling the character well in the previous time slices (that is in Line 6 the *state* is equal to $found$) then we say that the user has lost the character Samu Entropy and the visual complexity of the display will be saved in Line 7. The sequence of these losing values and the symmetrical finding values saved in Line 16 are shown in Fig. 2. The complexity is computed in bits per second (bps) units that is based on the number of changed pixels between two consecutive rectangular environments of the character with a given width and height.

Finally, it should be noticed that the magic-numbers such as 121 or 12 in the code snippet Listing 1 are based on systematic tryouts. Using these values allows players to experience the sensation of losing the character during the 10 minutes of the test. The code snippets Listing 1 and 2 do not include advanced C++ language specific elements so it can be considered as pseudocode. However using the original source code gives the possibility to investigate the algorithm of BrainB as precise as possible.

[4]`https://github.com/nbatfai/esport-talent-search/blob/master/BrainBWin.cpp#L65`

Listing 1. The algorithm for administration of losing and finding the character.

```
 1 int dist = ( this->mouse_x - x ) * ( this->mouse_x - x )
 2  + ( this->mouse_y - y ) * ( this->mouse_y - y );
 3 if ( dist > 121 ) {
 4    ++nofLost; nofFound = 0;
 5    if ( nofLost > 12 ) {
 6        if ( state == found && firstLost ) {
 7            found2lost.push_back(brainBThread->get_bps());
 8          }
 9        firstLost = true;
10        state = lost; nofLost = 0; brainBThread->decComp();
11      }
12  } else {
13    ++nofFound; nofLost = 0;
14    if ( nofFound > 12 ) {
15        if ( state == lost && firstLost ) {
16            lost2found.push_back(brainBThread->get_bps());
17          }
18        state = found;
19        nofFound = 0; brainBThread->incComp();
20      }
21  }
```

The final result printed by the benchmark after it ends in the form "*U R about 5.92902 Kilobytes*" is the mean of upper bounds for the bps values of the display measured when the variable state changes from found to lost (in Listing 1 from Line 6 to 9) and vice versa, when the variable state changes from lost to found (in Listing 1 from Lines 15 to 18). The simple calculation of this final result is shown in Listing 2.

Listing 2. The calculation of the final result of the benchmark.

```
int m1 = mean ( lost2found ), m2 = mean ( found2lost );
double res = ( ( ( ( double ) m1
   + ( double ) m2 ) /2.0 ) /8.0 ) /1024.0;
textStream << "U␣R␣about␣" << res << "␣Kilobytes\n";
```

2.1. **First Measurements.** As concluded in our former preliminary study [3], one of the further developments of Series 5 is changing to full screen from fixed-size window. This modification affects the basic operation of the benchmark, so the first objective was to verify that whether the sensation of losing the character still appears correctly or not. On Windows systems there were no problems. One of the first experiments using default settings on Windows 10 can be seen in Fig. 2 and Fig. 1.

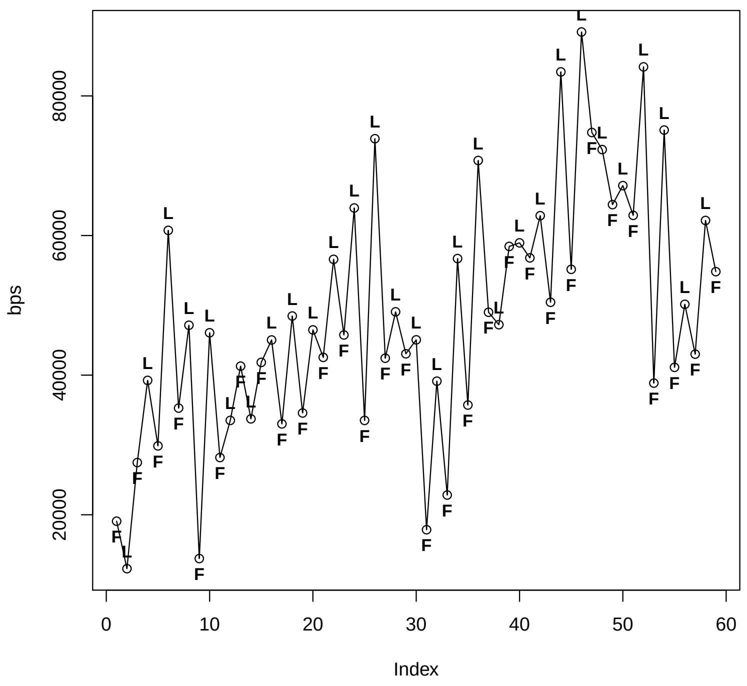


FIGURE 2. The bps values associated to events of losing and finding. The first element of this sequence is the first element of the *lost2found* (shown in Listing 1 Line 6) sequence. The second element is the first element of the *found2lost*, and so on. It should be noticed that the losing (labeled by L) and finding (F) events are mixed, see, for example the 13th event on the x axis where the complexity of finding is greater than the complexity of losing in this individual measurement.

2.2. **Systematic Measurements with Series 6.** The BrainB Series 6 was measured in two groups: UDPROG and DEAC-Hackers. The first one is a Facebook community of the BSc course of "High Level Programming Languages" at the University of Debrecen. The second one is an esport department of the University of Debrecen's Athletic Club. Participation in the BrainB Series 6 survey was voluntary in both groups. In the UDPROG community 33 members send back their results including the PNG screenshot and the produced text file within 2 days from the date of announcement (20 August 2018).

The averaged losing and finding curve for all members is shown in Fig. 3. It should be noticed that x-axis is not the time when the losing or finding events were occurred but only the order of events. The index denotes the temporal order of events that occurred. This simple averaged curve has shown that the consecutive losing and finding events has been precisely separated. We believe this means that our notion of losing and finding is strong enough to capture the investigated phenomenon "losing the character". In addition, the easily understandable averaged curve allows us to avoid application of mathematical statistics hypothesis testing in this case.

All anonymized raw data measured in this experiment can be found at `http://smartcity.inf.unideb.hu/~norbi/BrainBSeries6/measurements` (in the DEAC-Hackers community 12 esport players send back their results).

3. Conclusion

Our research hypothesis was that the mean of the complexity of changing lost to found is less than the mean of the changing found to lost. Fig. 3 shows the fulfillment of this hypothesis. It seems very clear in these figures that the averaged losing and finding curve has precisely separated the losing and finding events. Intuitively, this result shows that we lose the character on a higher complexity level then we find it on a relatively lower level again. This simple hypothesis has been proved by the results of this study. In order to further strengthen the completion of our benchmark test in a competitive way in the following versions we are going to offer to test subjects a little more liberty of fine-tuning the settings, for example with regard to the fine-tuning of mouse settings and to using custom colors. The next research objective will be to verify the satisfaction of Hick's law. To achieve this goal it is simple enough to compare the complexity of the finding and losing events with the time differences for each. Unfortunately, the actual version of the BrainB benchmark does not record these timestamps. The BrainB Series 7 will contain this feature. Our long-term research goal is to further develop our benchmark to a standard psychological test that can be used for talent search in esport.

4. Acknowledgment

Thanks to the students of the BSc course titled "High Level Programming Languages" at the University of Debrecen, to the members of the NEMESPOR mailing lists `https://groups.google.com/forum/#!members/nemespor`, to the members of the UDPROG Facebook community `https://www.facebook.com/groups/udprog` and to the members of the DEAC-Hackers esport department `http://deac.hu/szakosztalyok/esport` for their interest and for performing

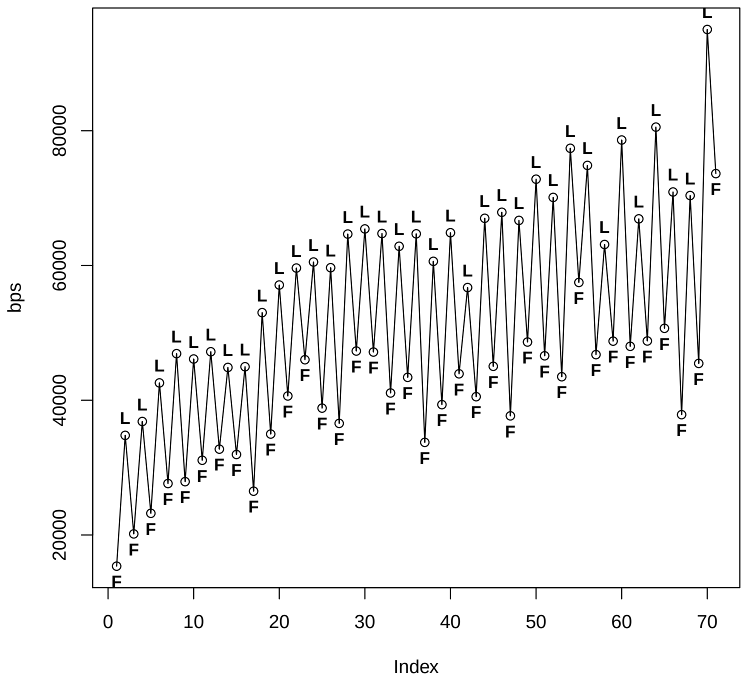


FIGURE 3. Measurements in the community UDPROG. The arithmetic mean of the final results of UDPROG participants is 4.95345. This figure shows the averaged losing and finding curve for all UDPROG participants where the losing (L) and finding (F) events are also indicated. The anonymized data can be found at `http://smartcity.inf.unideb.hu/~norbi/BrainBSeries6/measurements/UDPROG/`.

the BrainB Test Series 6. Special thanks to Roland Paszerbovics, Gergő Hajzer and Péter Rozsos for their interest and support and to Nándor Benjámin Bátfai for performing the BrainB Test Series 6 on several occasions. Finally, our thanks go to Dr. Péter Jeszenszky and Mr. Louis Mattia for reading the manuscript and suggesting improvements. A broader version of this paper can be found as Arxiv preprint [4]. Author contributions were the following: N. B. conceived the idea, developed the benchmark program, collected the data from the UDPROG community and analyzed the measurements. R. B. collected the data from the DEAC-Hackers. All authors wrote the manuscript.

References

[1] J. A. Anguera, J. Boccanfuso, J. L. Rintoul, O. Al-Hashimi, F. Faraji, J. Janowich, E. Kong, Y. Larraburo, C. Rolle, E. Johnston, and A. Gazzaley. Video game training enhances cognitive control in older adults. *Nature*, 501(7465):97–101, 2013.

[2] N. Bátfai. esport-talent-search. GitHub repository, 2017. `https://github.com/nbatfai/esport-talent-search` (visited: 2018-09-09).

[3] N. Bátfai, G. Bogacsovics, R. Paszerbovics, A. Antal, I. Czevár, V. Kelemen, and R. Besenczi. E-sportolók mérése (*Measuring esport athletes*). *Információs Társadalom*, 18(1):146–155, 2018. Original document in Hungarian: `http://real.mtak.hu/79216/1/it_2018_1_10_batfai_et_al.pdf` (visited: 2018-09-09).

[4] N. Bátfai, D. Papp, R. Besenczi, G. Bogacsovics, and D. Veres. Benchmarking Cognitive Abilities of the Brain with Computer Games. *ArXiv e-prints*, 2018. `https://arxiv.org/abs/1809.00172` (visited: 2018-09-09).

[5] B. Bediou, D. M. Adams, R. E. Mayer, E. Tipton, C. S. Green, and D. Bavelier. Meta-analysis of action video game impact on perceptual, attentional, and cognitive skills. *Psychological bulletin*, 144(1):77–110, 2018.

[6] V. Csépe, M. Győri, and A. Ragó. *Általános pszichológia 1. - Észlelés és figyelem*. Osiris Kiadó, 2007.

[7] J. Geyer, P. Insel, F. Farzin, D. Sternberg, J. L. Hardy, M. Scanlon, D. Mungas, J. Kramer, R. S. Mackin, and M. W. Weiner. Evidence for age-associated cognitive decline from internet game scores. *Alzheimer's & Dementia: Diagnosis, Assessment & Disease Monitoring*, 1(2):260–267, 2015.

[8] B. D. Homer, J. L. Plass, C. Raffaele, T. M. Ober, and A. Ali. Improving high school students' executive functions through digital game play. *Computers & Education*, 117:50–58, 2018.

[9] N. H. Mackworth. The breakdown of vigilance during prolonged visual search. *Quarterly Journal of Experimental Psychology*, 1(1):6–21, 1948.

[10] M. Moisala, V. Salmela, L. Hietajärvi, S. Carlson, V. Vuontela, K. Lonka, K. Hakkarainen, K. Salmela-Aro, and K. Alho. Gaming is related to enhanced working memory performance and task-related cortical activity. *Brain Research*, 1655:204–215, 2017.

[11] R. Nieva. Facebook's moonshots: making brains type and skin hear, 2017. `https://www.cnet.com/news/facebook-f8-building-8-moonshot-projects-zuckerberg-regina-dugan/` (visited: 2018-09-09).

[12] B. Pataki, P. Hanák, and G. Csukly. Computer games for older adults beyond entertainment and training: possible tools for early warnings-concept and proof of concept. In *ICT4AgeingWell*, pages 285–294, 2015.

[13] B. Pataki, P. Hanák, and G. Csukly. Surpassing entertainment with computer games: online tools for early warnings of mild cognitive impairment. In M. Helfert, A. Holzinger, M. Ziefle, A. Fred, J. O'Donoghue, and C. Röcker, editors, *Information and Communication Technologies for Ageing Well and e-Health*, pages 217–237. Springer International Publishing, 2015.

[14] D. E. Powers. Test preparation for the GRE analytical ability measure: differential effects for subgroups of GRE test takers. *ETS Research Report Series*, 86(2):1–19, 1986.

[15] D. E. Powers and S. S. Swinton. Effects of self-study for coachable test item types. *Journal of Educational Psychology*, 76(2):266–278, 1984.

[16] S. Rózsa, N. O. Nagybányai, and A. Oláh. *A pszichológiai mérés alapjai: elmélet, módszer és gyakorlati alkalmazás*. Bölcsész Konzorcium, 2006.
[17] C. Sagan. *Dragons of Eden: speculations on the evolution of human intelligence*. Random House Publishing Group, 2012.
[18] S. C. Seow. Information theoretic models of HCI: a comparison of the Hick-Hyman law and Fitts' law. *Human-Computer Interaction*, 20(3):315–352, 2005.
[19] E. Toulouse and H. Piéron. *Toulouse-Piéron-Revisado Prueba perceptiva y de atención*. TEA, 8 edition, 2013. `http://www.web.teaediciones.com/Ejemplos/Extracto_libro_TP-R.pdf` (visited: 2018-09-09).
[20] H. A. Witkin, H. B. Lewis, M. Hertzman, K. Machover, P. Meissner, and S. Bretnall Wapner. *Personality through perception: an experimental and clinical study*. Harper, Oxford, England, 1954.
[21] R. M. Yerkes and J. D. Dodson. The relation of strength of stimulus to rapidity of habit-formation. *Journal of Comparative Neurology and Psychology*, 18:459–482, 1908.

All authors are with the University of Debrecen, P.O. Box 400, H-4002, Debrecen, Hungary.
Email address: `{batfai.norbert, besenczi.renato}@inf.unideb.hu`

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LXIV**, Number 1, 2019
DOI: 10.24193/subbi.2019.1.03

MEDICAL IMAGE ANALYSIS WITH SEMANTIC SEGMENTATION AND ACTIVE LEARNING

CHARLES ISAH SAIDU[1,*] AND LEHEL CSATÓ[2]

Abstract. We address object detection using semantic segmentation and apply it for prostate detection in an MRI data-set. Our detection pipeline uses first a segmentation step followed by a classifier with a convolutional neural network (CNN). Since the segmentation provides a set of unbalanced data-sets – where a high accuracy is difficult to obtain – we leverage the prospect of improving detection accuracy using a Bayesian treatment of deep networks and the possibility of better exploiting the data using active learning. The resulting algorithm is both adaptive and data-efficient: by assuming that from a large pool of data only a few are segmented, the active learning module of the algorithm finds the image that improves most detection accuracy. We test our algorithm on a prostate medical image data-set and show that the active learning-based algorithm performs well in the prostate detection class. The resulting system is invariant to translations within the image and the results show improvements when using the pipeline that includes active learning and CNNs.

1. Introduction

Semantic segmentation is the task of *annotating* an image: which *region in an image* belongs to a defined class [14, 31]. Since images are pixels, in practice we have to *classify* every pixel – a difficult task since class information is not local to single pixels; consequently *semantic* refers to the global character of the segmentation task. In this work we focus on a mixed segmentation approach that involves both "traditional" image processing [28] as well as adaptive neural-network-based classification modules [3]. The image processing modules are used to *segment* the images into *superpixels* [2] that are subsequently fed into a neural network-based classifier that classifies the superpixel as belonging to the *region of interest* or not. Superpixels, the result

Received by the editors: September 25, 2018.

2010 *Mathematics Subject Classification.* 68T10, 68U10.

1998 *CR Categories and Descriptors.* H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval – *Clustering*; I.4.6 [**Image Processing and Computer Vision**]: Segmentation – *Edge and feature detection*;

Key words and phrases. semantic segmentation, active learning, deep networks.

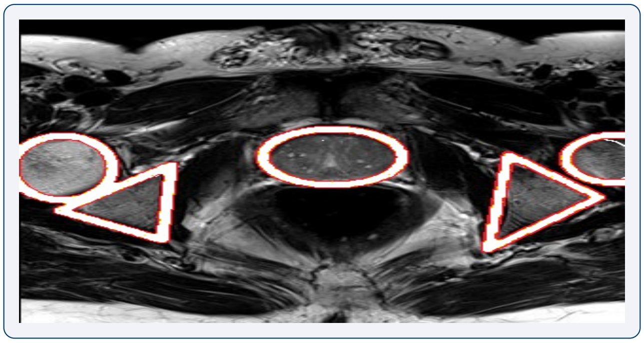


FIGURE 1. Axial scan of prostate gland region. The highlighted areas are similar to the prostate gland – in the centre – and difficult to discriminate, leading to classification errors.

of the semantic segmentation, are the *classification atoms* in this work, this simplifies the classification task and helps to better segment the image.

1.1. **Problem statement.** We aim for a good classifier that is able to *"select"* – or label – the regions of interest (ROI) in an image – an example is provided in Fig. 1. The learning is classification-based and we use a set of labelled images for training. We do not aim for a pixel-level classification and consider the existence of a pre-processing step that identifies the superpixels.

We aim to build a superpixel classification procedure that leads to an efficient labelling of the entire image. Our goal is to have a classifier that is:

- *adaptive* – it is able to learn from labelled data;
- *efficient* – it is able to learn from a small set;
- *generic* – the recognition is location-invariant – exploiting thus the higher-order invariants within the ROI, as illustrated in Fig. 1.

We focus on automated medical image segmentation: identification of the region of interest (ROI) within an image [see for example 19]. We use the prostate data-set from the I2CVB initiative,[1] one image is in Fig 1. This is a natural data-set and is considered as difficult to handle due to low contrast, speckle, micro-calcification in MRI images, as well as the presence of imaging artefacts [10].

1.2. **The suggested solution.** We focus on the classifier in the processing pipeline in Fig. 2 and use a *deep convolutional neural network* [3, 13] for classification and we optimise the learning procedure by using active learning. Active learning – also called "query learning" – is important since it allows us

[1]The database is available at `http://i2cvb.github.io/`

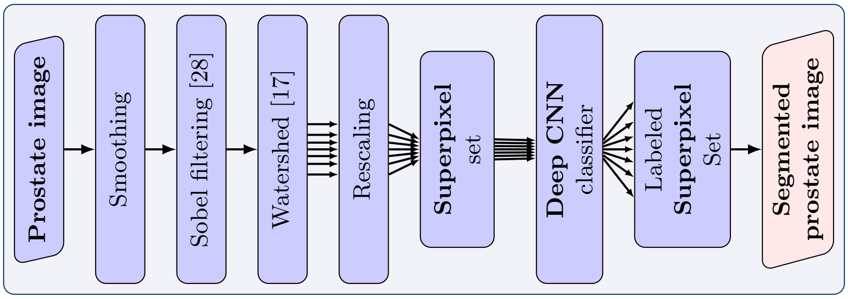


FIGURE 2. The processing pipeline starts with classical image processing, is followed by an adaptive classifier, a convolutional network. The annotated image is built from the labelled units.

to train the system using less data [7, 25]. Active learning is also important since the resulting superpixel classification problem is unbalanced [3]: out of ≈ 60 superpixels only a few are positive. Class imbalance leads to problems for the classifier: classifying all superpixels negative is a strong local optimum, and the learning might stop at this state. We use active learning and re-sampling techniques [4] to counter the class imbalance.

1.3. **Structure of the paper.** First, in Sec. 2 we list the related literature, then we discuss our proposed solution in Sec. 3. We test the proposed algorithm in Sec. 4, discuss our findings and further research directions in Sec. 5.

2. Related work

In this section we present two categories of research for medical image segmentation: graph-based segmentation and pixel-level classification based on deep networks.

2.1. **Graph-based semantic segmentation.** These methods build graphs where the nodes are superpixels and the weighted connections are computed using similarity measures between superpixels [7, 30, 31]. It is worth noting that the main motivation is to minimize annotation efforts by labelling groups of pixels – known as superpixels – instead of single pixels.

Vezhnevets et al. [31] used pairwise – label- and "connectedness" – *conditional random fields* (CRFs) over superpixels. The goal was to learn the parameters of this joint model using an energy function that captures both the ability to classify superpixels (unary potential) and the connectedness of superpixels to its neighbouring superpixels. Vezhnevets et al. [31] also applied active learning by designing a query scoring function that attempts to maximize the expected model change on the appearance model parameters.

Fathi et al. [7] focused on semantic video segmentation by building a graph of superpixels connected via a similarity metric. Here an incremental self-training approach was proposed that iteratively first labels the least uncertain frame, followed by the update of similarity metrics based on the extended set of labels.

2.2. **Semantic segmentation with deep networks.** Several techniques that are based on neural networks have been proposed for semantic segmentation and object detection. By far not exhaustive, we mention the work of Ciresan et al. [5] using deep networks, the fully convolutional network by Shelhamer et al. [27], U-NET semantic segmentation by Ronneberger et al. [22], R-CNN and "Fast R-CNN" for object detection [11, 12], "Faster R-CNN" by Ren et al. [21] and Masked R-CNN by He et al. [15].

Ciresan et al. [5][2] used sliding windows approach. Each sliding window/image patch was labelled and subsequently used as input to a deep neural network. The method was difficult to train: there were a lot of overlapping patches for each image; making it computationally expensive. The patch size induced a trade-off between localization accuracy and patch size: larger patches require more layers and their size reduces localization [see e.g. 22].

Shelhamer et al. [27] extracted individual patterns from the image and classified patches independently. They use multiply connected convolution layers and no fully connected layers. Each convolution layer preserves the size of the input image – the output size is the same size as the input size. Each pixel in the output is annotated and the training was done using uses the cross-entropy likelihood.

Ronneberger et al. [22] proposed U-Net, an extension in architecture of Shelhamer et al. [27]. In their work, only a few training samples are required and the labelled data were in turn augmented. They used up-sampling operators in the decoding part of the network and applied the algorithm successfully to biomedical image segmentation.[3]

Girshick et al. [12] developed the "R-CNN" model for object detection. In contrast to the sliding window approach of [5], they used a selective search for sub-regions, and the ROI was determined using these regions only. The R-CNN model was later extended by the same team under the name "Faster R-CNN" [11]. We also mention "Mask R-CNN" [15] that is an extension of the original R-CNN [12] for object instance segmentation. This latter model uses a "simple model" for image class detection and one for segmentation.

[2]The work has won the electron-microscopy image segmentation challenge in 2012.

[3]The work has won the ISBI cell tracking challenge in 2015, `emb.citengine.com/event/isbi-2015/details`, result presented at the ISBI conference (accessed 02.10.2018).

3. Active Learning using Deep Networks

In what follows, we concentrate on the *classification task* – the *Deep Net classifier* – DNN – from Fig. 2. The inputs to the DNN are the superpixels and we know – see Sec. 1.2 – that the set of labels is extremely unbalanced: often we have 60 negative examples and a single positive one. To counter the negative effect of unbalanced data, a guided data selection – the *active learning* framework [25] – is used, in combination with gradient descent for neural networks and the probabilistic output – necessary for the scoring functions – is provided by the dropout mechanism [8]. We present first the neural network architecture, then the active learning framework, followed by a general overview of the algorithm.

In what follows $X_u = \{x_1, x_2, \ldots, x_n\}$ denotes the set of *re-scaled superpixels*. Since labelling is assumed to be difficult, we only have a small *labelled data-set* $D_\ell = \{(x_1, y_1), (x_2, y_2), \ldots, (x_l, y_l)\}$, with $l \ll n$ and $X_u = X \setminus X_\ell$ is the pool of unlabelled superpixels.

3.1. **Deep networks with dropout and probabilistic outputs.** For each image, we pre-segment the image using watershed algorithm, hence generating patches of images which we call superpixels - this is opposed to using the sliding-windows approach. These superpixels formed input to the deep neural network. Since each image produces many superpixels set and the task is to identify a region as prostate and not prostate, labelling all superpixels within an image would be laborious.So we endowed the deep neural network with a Bayesian treatment and the started by training only a small sample. The resulting sub-optimal classifier is then used together with active learning technique as a probe for searching the remaining unlabelled superpixels for ONLY informative superpixels that will improve accuracy and generalization. Hence, we use a "deep neural network" with both a *convolution* part and a *fully connected* part. We use 4×4 convolution matrices, yielding 32 filters – since we aimed to capture pattern diversity. Each of the two convolution layer is followed by a *relu* non-linearity layer. After the last convolution level we have a *dropout layer* with dropout mean probability of 0.5. The fully connected part contains 3 dense layers each followed by a *relu* and a *dropout* 0.5 layers. The last layer is a *softmax* layer; the network error function is cross-entropy, and the training was done using the gradient descent with ADAM learning [18]. For network – i.e. classifier – optimisation we use gradient descent with the *tensorflow* package [1] and the optimisation uses only the dataset D_ℓ.

Dropout – originally introduced to prevent model over-fitting [29] – uses a "blocking mechanism": if a binary gate is open, the neuron output is calculated normally, if the gate is closed, the output of the respective neuron is zero.

Aside from its stabilising role, dropout can also be used to calculate predictive uncertainties [8, 9]: we sample the dropout gates and this leads to different weight vectors ω_t – with some of the weights set to zero, providing in turn a predictive distribution given as:

$$P(y|x) \approx \frac{1}{T} \sum\nolimits_{t=1}^{T} P(y|x, \omega_t) \tag{1}$$

with $\omega_t = \omega \otimes g_t$ where g_t is a configuration of the dropout gates. We mention that the above scheme is valid only *if training was made using dropout*. We believe that dropout is important: Gal et al. [9] has shown that dropout is equivalent to performing an approximate Bayesian inference, therefore the samples are approximations of the "true posterior" distribution. The a-posteriori distribution is used to decide which superpixel will be included into the training set: it is based on scoring the super-pixels, the different *query scores* are defined in Sec. 3.2.

3.2. **Querying Technique.** The querying technique or acquisition function [26] defines a score for superpixels. Based on this score the unlabelled superpixels will be selected and labelled. The scoring is done such that it brings the most "information" conditioned on the already labelled data. We explored the following scoring functions:

(1) *Maximum entropy* chooses the superpixels that maximize the predictive entropy [20]:
$$x^* = \arg\max_{x \in D_u} \mathbb{H}[y|x, D_u]$$
(2) *BALD (Bayesian Active Learning By Disagreement)* looks for the superpixel x that maximizes the decrease in conditional entropy caused by the posterior [9, 16]:
$$x^* = \arg\max_{x \in D_u} \mathbb{H}[y|x, D_u] - \mathbb{E}_{p(\omega|D_u)} \mathbb{H}[y|x, D_u]$$
(3) *Random acquisition*: randomly chooses a subset of unlabelled superpixels from the pool.

We compare the acquisition functions in the Results Section 4.3.

3.3. **Oversampling the data.** We found that – in spite of the results of Ertekin et al. [6] showing that active learning provides a natural way to handle imbalanced data – using active learning alone does not eliminate label imbalance. Instead, when running the full pipeline, we found that over-sampling of positive data and under-sampling negative data is useful.

Oversampling is a technique used to adjust class distribution of a dataset: the *synthetic minority oversampling technique*, or SMOTE, replicates samples based on the k-nearest neighbours of the under-sampled items [4]. Aside from

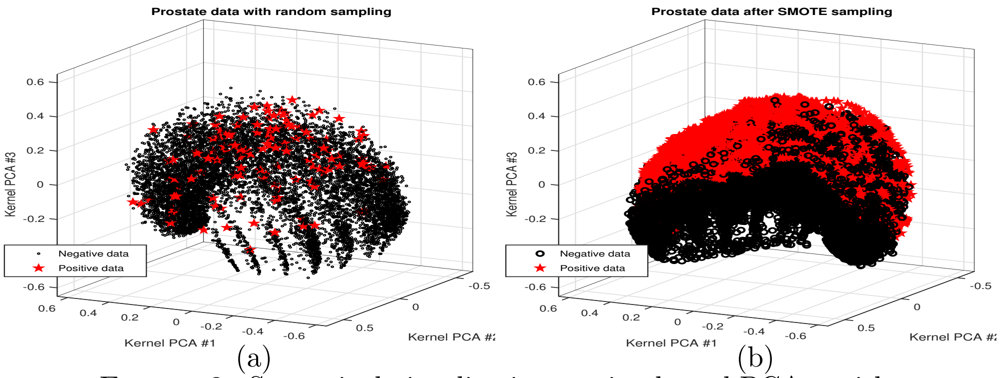


FIGURE 3. Superpixel visualisation – using kernel PCA – with (a) uninformed sampling and (b) informed SMOTE oversampling – image best viewed in colour.

the improved performance, the oversampling method was motivated from a visualization of the superpixel dataset from Fig. 3. In the visualisation we looked at the "topology" in the superpixel space: we wanted to assess the separability of the superpixels as defined by the classification problem. We used the kernel PCA method [24] – a non-linear projection methods that takes into account the distribution of the superpixels – and coloured the ROI superpixels as red, the negative data as black. We see that the oversampling makes the two classes more separable in the latent space.[4]

3.4. **The full algorithm.** We detail the adaptive "Deep CNN classifier" – shown in Algorithm 1 – the part of the image segmentation pipeline from Fig. 2. The input to the algorithm is the collection of resized and cropped superpixels.

To mimic real situation, we consider that superpixel labelling is done on-demand: when required, one can ask an expert to label an image, leading to a labelling of a superpixel set.

The training starts by labelling a small randomly selected set of superpixels – this was set to 50 to mimic the acquisition of superpixels from a single image.

After the initial labelling the algorithm proceeds by using the active learning technique described in sec. 3.2: we select from a pool of unlabelled superpixels D_u to add to the training set D_ℓ, then we re-train the model.

The iteration follows as long as a stop-condition is not met or all data has been labelled, i.e. $X_u = \emptyset$. In our experiments we use a fixed number of iterations as stop-condition, this number resulted from testing the algorithm and finding that going over 30 iterations makes no difference in performance.

[4]Since the classification is not performed by a kernel machine [23], the plots provide only hints about the difficulty of the classification task.

Algorithm 1 The Active Segmentation Algorithm

```
1: procedure TRAINING($X_u$)
2:     Select $X_{init}$; $X_u = X_u \setminus X_{init}$          ▷ superpixels from a *single* image
3:     $D_\ell \leftarrow$ oracle($X_{init}$)          ▷ labelling
4:     trainedModel $\leftarrow$ deepConvNet($D_\ell$)          ▷ initial training
5:     repeat
6:         $X_{sub} \subset D_u$
7:         $S_{sub} \leftarrow$ scoreSuperpixel(trainedModel, $X_{sub}$)
8:         $k \leftarrow \arg\max\{s_i \mid s_i \in S_{sub}\}$          ▷ index of the “best” s.pixel
9:         $y_k \leftarrow$ oracle($x_k$)          ▷ labelling the selected superpixel
10:                                            ▷ if oversampling, then
11:        $X_k \leftarrow \{(x_k, y_k)\}$          ▷ $X_k \leftarrow$ oversample( $(x_k, y_k)$ )
12:        $D_\ell = D_\ell \cup X_k$
13:        $X_u = X_u \setminus \{x_k\}$
14:        trainedModel $\leftarrow$ deepConvNet($D_\ell$)          ▷ re-training
15:    until stopCondition $\vee X_u = \emptyset$
16:    return trainedModel
```

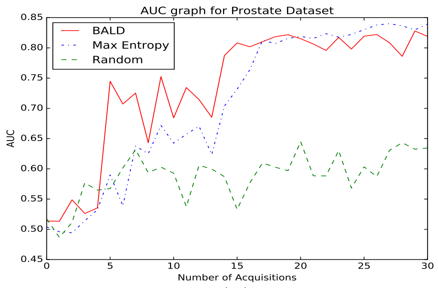
4. Experiments and results

The experiments used the pipeline from Fig. 2: we first perform a Gaussian smoothing, followed by a Sobel filtering [28], before using watershed to generate the superpixels. These patches – after labelling – will be combined into a *segmented* image as a result of the processing pipeline.

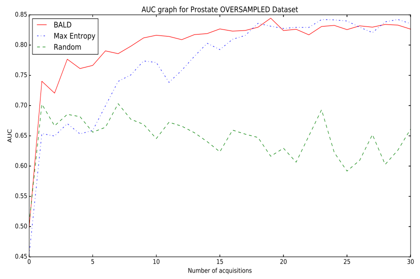
We use MRI axial scans from a total of 30 patients. Each patient's database consists on average of 20 axial slices of the prostate region at different levels. For each image slice, we perform over-segmentation of the regions within the image and generate an average of 60 superpixels per slice, making an approximate total of $36,000$ superpixels that can potentially be used for training.

4.1. **No-oversampling Experiment.**

To handle location-invariance, we crop out each superpixel/patch and resize to a standard size of 40×40 pixels before feeding into the convolutional network. This process further constrains and makes ROI detection a bit more difficult for this data-set. In particular – as in Fig. 1 – it is obvious that there are huge similarities in feature space between highlighted regions thus making it a lot more difficult for the classifier to learn which of these regions is prostate and which is not. In addition, our approach creates a hugely imbalanced data-set shown in Fig. 3.b, where we show a kernel PCA projection of the data-set. Consequently, we observed that active



(a)



(b)

FIGURE 4. Area Under the ROC Curve (AUC) for (a) non-oversampled and (b) oversampled data-set.

learning alone did not deal implicitly with the issue of class imbalance as reported by [6] - A case that needs further investigation. However, after actively overssampling during each training step, considerable performance gains were observed as depicted in 4.3.

4.2. **Oversampling Experiment.** In order to evaluate the effect of feature space similarities in superpixels and class imbalance in the task of detection and agglomeration of the prostate region, we set up the experiment with a slight modification to the algorithm. This is prompted by the kernel PCA visualization of the data as a result of oversampling as shown in Fig. 3. In the modification, we oversampled *only after each batch acquisition from the data-set*. Algorithm 1 captures the idea if the oversampling flag set to $True$.

4.3. **Results.** Fig. 4 shows the results of successive Area Under the Receiver Operating Characteristic curve (AUC-ROC). From the figure, it would be misleading to think that performance improves as more superpixels are added to the training set. However, to the contrary, we obtained zero precision and zero recall when increasing the data-set size. Recall from Sec. 1.1 and the illustration in Fig. 1 that there is really very marginal distinction between regions of the prostate and non-regions of the prostate especially when the images are cropped, coupled with the fact that the imbalance ratio between negative and positive class is 60 to 1.

Consequently, we decided to over-sample the most informative superpixels after acquisition so as to give the algorithm more representation of what the actual prostate superpixels look like. This led to Fig. 4.b and Fig. 5 in which Fig. 5 shows improvement in precision of the prostate region after the 15-th acquisition.

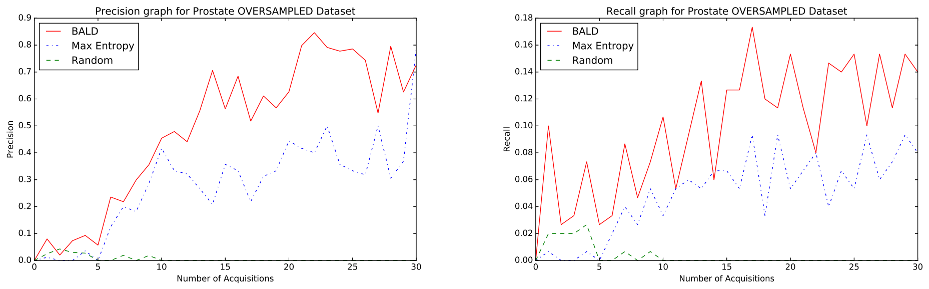


FIGURE 5. Precision (left) and recall (right) for oversampled data.

5. Conclusions and Future Research

We presented a semantic segmentation pipeline using active learning, providing a non-pixel-based adaptive segmentation; applied to medical images. Within the active learning, to achieve better segmentation results, we had to over-sample the positive class of superpixels, improving considerably the accuracy of the system, measured through the AUC curve. We also obtained a higher precision and recall compared to randomly selecting superpixels to be used for training the neural network. Overall, we observed that active learning technique could be complemented with oversampling techniques for better results.

In what follows, we plan to explore means of integrating active learning in the U-NET-style of semantic segmentation. Researchers measured the capability of the detection pipelines [22] using a pixel-wise matching of the desired and the true ROI. Applying the intersection over union (IoU) metric [22] means that the whole processing pipeline is evaluated, therefore one might want to optimise all other parameters also. An interesting prospect would be the use of 3D kernels for segmentation: given that there are several data-sets where the successive nature of the images can be exploited and the exploitation of this extra information is a promising research direction.[5]

5.1. Acknowledgements.

C.I. Saidu wants to thank the African Development Bank for support via its home university. L. Csató acknowledges financial support of the European Regional Development Fund and the Romanian Government through the Competitiveness Operational Programme 2014-2020, project P/37/679, contract no. 157/16.12.2016.

[5]An example is the project "Lung cancer detection using 3D CNN's" (link).

References

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015.

[2] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk. Slic superpixels compared to state-of-the-art superpixel methods. *IEEE transactions on pattern analysis and machine intelligence*, 34(11):2274–2282, 2012.

[3] C. M. Bishop. *Neural Networks for Pattern Recognition.* Oxford University Press, Inc., New York, NY, USA, 1995. ISBN 0198538642.

[4] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. Smote: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16:321–357, 2002.

[5] D. Ciresan, A. Giusti, L. M. Gambardella, and J. Schmidhuber. Deep neural networks segment neuronal membranes in electron microscopy images. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 2843–2851. Curran Associates, Inc., 2012.

[6] S. Ertekin, J. Huang, L. Bottou, and L. Giles. Learning on the border: active learning in imbalanced data classification. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, CIKM '07, pages 127–136, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-803-9. doi: 10.1145/1321440.1321461.

[7] A. Fathi, M. F. Balcan, X. Ren, and J. M. Rehg. Combining self training and active learning for video segmentation. In *Proceedings of the British Machine Vision Conference*, pages 78.1–78.11. BMVA Press, 2011. ISBN 1-901725-43-X.

[8] Y. Gal. *Uncertainty in Deep Learning.* PhD thesis, University of Cambridge, 2016.

[9] Y. Gal, R. Islam, and Z. Ghahramani. Deep Bayesian active learning with image data. In D. Precup and Y. W. Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1183–1192, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR.

[10] S. Ghose, A. Oliver, R. Martí, X. Lladó, J. C. Vilanova, J. Freixenet, J. Mitra, D. Sidibé, and F. Meriaudeau. A survey of prostate segmentation methodologies in ultrasound , magnetic resonance and computed tomography images. *Comput. Methods Programs Biomed.*, 108(1):262–287, 2012. ISSN 0169-2607. doi: 10.1016/j.cmpb.2012.04.006.

[11] R. Girshick. Fast r-cnn. In *The IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[12] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014.

[13] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning.* MIT Press, 2016.

[14] S. Gould, R. Fulton, and D. Koller. Decomposing a scene into geometric and semantically consistent regions. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 1–8. IEEE, 2009.

[15] K. He, G. Gkioxari, P. Dollár, and R. B. Girshick. Mask R-CNN. *CoRR*, abs/1703.06870, 2017.

[16] N. Houlsby, F. Huszár, Z. Ghahramani, and M. Lengyel. Bayesian active learning for classification and preference learning. *ArXiv e-prints*, 2011.

[17] Z. Hu, Q. Zou, and Q. Li. Watershed superpixel. *2015 IEEE International Conference on Image Processing (ICIP)*, pages 349–353, 2015. doi: 10.1109/ICIP.2015.7350818.

[18] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[19] G. Litjens, R. Toth, W. van de Ven, C. Hoeks, S. Kerkstra, B. van Ginneken, G. Vincent, G. Guillard, N. Birbeck, J. Zhang, R. Strand, F. Malmberg, Y. Ou, C. Davatzikos, M. Kirschner, F. Jung, J. Yuan, W. Qiu, Q. Gao, P. Edwards, B. Maan, F. van der Heijden, S. Ghose, J. Mitra, J. Dowling, D. Barratt, H. Huisman, and A. Madabhushi. Evaluation of prostate segmentation algorithms for MRI: The PROMISE12 challenge. *Medical Image Analysis*, 18(2):359–373, 2 2014. ISSN 1361-8415.

[20] D. J. C. MacKay. *Information Theory, Inference & Learning Algorithms.* Cambridge University Press, New York, NY, USA, 2002. ISBN 0521642981.

[21] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 91–99. Curran Associates, Inc., 2015.

[22] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. *CoRR*, abs/1505.04597, 2015.

[23] B. Scholkopf and A. J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond.* MIT Press, Cambridge, MA, USA, 2001. ISBN 0262194759.

[24] B. Schölkopf, A. Smola, and K.-R. Müller. Kernel principal component analysis. In B. Schölkopf, C. J. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*, pages 327–352. MIT Press, 1999.

[25] B. Settles. Active Learning Literature Survey. *Mach. Learn.*, 15(2):201–221, 2010. ISSN 00483931. doi: 10.1.1.167.4245.

[26] B. Settles, M. Craven, and S. Ray. Multiple-instance active learning. In *Advances in Neural Information Processing Systems 20 - Proceedings of the 2007 Conference*, volume 20, pages 1289–1296, 2008. ISBN 160560352X.

[27] E. Shelhamer, J. Long, and T. Darrell. Fully convolutional networks for semantic segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 39(4):640–651, Apr. 2014. ISSN 0162-8828. doi: 10.1109/TPAMI.2016.2572683.

[28] M. Sonka, V. Hlavac, and R. Boyle. *Image Processing, Analysis, and Machine Vision.* Thomson-Engineering, 2007. ISBN 049508252X.

[29] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1): 1929–1958, Jan. 2014. ISSN 1532-4435.

[30] A. Vezhnevets, V. Ferrari, and J. Buhmann. Weakly Supervised Semantic Segmentation with Multi Image Model. *Proc. Int'l Conf. Comput. Vis.*, 2011.

[31] A. Vezhnevets, J. M. Buhmann, and V. Ferrari. Active learning for semantic segmentation with expected change. *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pages 3162–3169, 2012. ISSN 10636919.

[1] Computer Science Department, African University of Science and Technology, Airport Road, 10 km, Abuja, Nigeria

[2] Faculty of Mathematics and Computer Science, Babeş-Bolyai University, 1 Kogălniceanu, RO-400084 Cluj-Napoca, Romania

[*] Work partially done while at internship at Babeş-Bolyai University
Email address: isaidu@aust.edu.ng and lehel.csato@cs.ubbcluj.ro

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LXIV**, Number 1, 2019
DOI: 10.24193/subbi.2019.1.04

USING LATENCY METRICS IN NOSQL DATABASE PERFORMANCE BENCHMARKING

CAMELIA-FLORINA ANDOR, BAZIL PÂRV, AND DAN MIRCEA SUCIU

Abstract. This paper presents an experimental study evaluating the performance of NoSQL database management systems. The study compares two NoSQL database management systems (Cassandra and MongoDB) and considers the following parameters/factors: workload and degree of parallelism. Two different workloads (update heavy and mostly read) were used, and different numbers of threads. The measured results are related to average latency: update latency and read latency. Our study shows that with the only exception of 1000 operations, both latency indicators have a quasi-parabolic behavior, where the minimum (i.e. the best performance) depends mainly on the number of threads and slightly varies with the increase in the number of operations. In the case of 1000 operations, there is also a maximum point (i.e. worst performance) case, after which the latency decreases.

1. Introduction

NoSQL data models appeared from practical reasons, some industrial solutions becoming de facto standard solutions in the cases where the relational data model failed to provide acceptable performance in terms of horizontal scalability and high availability. A lot of different data models (key-value, document, column-family, graph, etc.) are collectively known today as NoSQL data models. Different providers offer today a large selection of highly configurable and flexible NoSQL database management systems. They differ in terms of the data models and distribution models they implement, as well as the real problems appropriate for their use. Consequently, it is very difficult to compare them. For the designer of an application using such a NoSQL

Received by the editors: January 8, 2019.

2010 *Mathematics Subject Classification.* 68P15, 68P99.

1998 *CR Categories and Descriptors.* H.2.1 [**Database Management**]: Logical design – *Data models*; H.2.4 [**Database Management**]: Systems – *Distributed databases, Parallel databases*.

Key words and phrases. NoSQL database, performance benchmarking, MongoDB, Cassandra.

database management system it is not enough to read the technical documentation in order to make the best design decisions. Instead, the usual approach is to use some performance benchmarks, which helps to see the actual behavior of the database and to choose the appropriate hardware configuration. This paper is the second describing a series of experimental studies, evaluating different performance indicators of two NoSQL database management systems: Cassandra and MongoDB. The current paper refers to two average latency measures, update latency and read latency, while the first one, [1], discussed the throughput averages. Latency indicators characterize the time needeed to perform a single operation/request (amount of time the request takes to complete), and throughput is related to the computational performance, measured in number of operations/requests performed per second. Both performance indicators have an equal importance: latency for describing the response time, and throughput for the server performance. The benchmarking tests for experimental studies were performed using the `Yahoo! Cloud Serving Benchmark` client, using varied pairs of number of operations, number of threads and workload on every database server.

2. Background

2.1. **NoSQL Data Models.**

Large companies like Amazon and Google introduced NoSQL data models to collect and manage large quantities of data in a distributed environment: Google Bigtable[3] and Amazon Dynamo[6]. Their success gave a new research field - NoSQL data models. In the order of their complexity, they are: key-value, document, column-family and graph. There are also combinations/variations of these basic models.

In the key-value model, the key part uniquely identifies the value part, which is not visible to the database (you cannot directly perform queries on values). Key-value databases organize data as key-value pairs, allowing arrays or objects to be stored as values for keys, but their structure is not exposed at the database level. Amazon Dynamo uses this model.

The column-family and document models are based on the key-value model. The document model allows you to query the value: a document resembles a record that belongs to a relational table. Document DBMSs organize documents in collections. Unlike the relational model, in which all the records have the same schema and a field can store only simple values, here the documents can have different schemas and their fields can store complex values like arrays or embedded documents. The most popular document formats are JSON[11], XML[21], and YAML[22]. Document DBMSs simplify the application development process, as the document's structure is similar to that of an object used at the application level.

In the column-family model, data is organized as rows that belong to column families. A column family is like a relational table, but has a flexible schema. Each column contains a key-value pair and a timestamp. The key is in fact the name of the column and the value is the column itself. The value of a column can be complex, like a collection or a tuple. Rows that contain different columns are allowed to be part of the same column family. A good practice in application development that involves the use of column-family DBMSs is to know the queries in advance, in order to optimize the database schema around those queries. Column-family DBMSs are generally optimized for write operations. Google Bigtable was the first implementation of this data model.

The graph model is the most complex, being appropriate for heavy interconnected data. In such a property graph, both nodes and edges have properties. If data are heavy interconnected, this affects the horizontal scalability, because it is difficult to decide where to split the graph into several sub-graphs stored into a distributed environment.

2.2. **NoSQL tools.** For our benchmarking study, two data models were considered: the document model (implemented by MongoDB[14]) and the column-family model (implemented by Cassandra[2]). Among other DBMSs implementing these models and available on the market we mention CouchDB[5], OrientDB[16] for the document model, and Bigtable[3], HBase[10] for the column-family model. OrientDB also supports other data models.

MongoDB is an open source distributed database developed by 10Gen, known today as MongoDB Inc. Its main features are: horizontal scalability, flexible schema, high availability, replication and an expressive query language. Ad hoc queries are very well supported. A MongoDB cluster is made up of shards, configuration servers and query routers. Shards or nodes are used to store data, config servers store cluster metadata, while query routers route queries to the shards.

The other open source distributed database considered is Cassandra, initially developed at Facebook[13]. Its main features are: high availability, data replication and horizontal scalability. Unlike MongoDB, which performs better in the case of read operations, its core query language is not so rich, and it offers better support for write operations. Cassandra's data model is column-family, based on Bigtable[3] and Dynamo[6]. A Cassandra cluster is made up of identical servers, which means each node can accept both read and write operations. Also, there is no downtime when adding or removing a node from the cluster.

In our case study, Apache Cassandra 3.11.0 and MongoDB 3.4.4 versions were installed on our servers.

2.3. **NoSQL benchmarking.** If many options (i.e. in our case many open source NoSQL database management systems) are available to the application developer, it is a difficult decision to choose the right one, which gives the best performance on a given hardware configuration and for a specific application use case. In these situations, where it is difficult to make comparisons between the performance of different NoSQL database servers, benchmarking is very useful. As it is the case in other similar situations, the benchmarking process needs its own tools, and there are not so many options in the open source movement.

The functionality of a benchmarking tool covers two different areas: workload generation and performance measurement using different workload scenarios. A workload is the load that is put by a certain application on the database management system, i.e. a batch of all requests a given application is sending to the server during a working session. For benchmarking purposes, the workload definition needs some clarifications, as we'll see below.

In database performance benchmarking, there are three important metrics: throughput, measured in operations per second, latency, measured in microseconds per operation, and total runtime, also measured in milliseconds. Throughput measures the number of operations per time unit (second), while latency measures the duration of a single operation (expressed in microseconds). The total runtime expresses the entire duration of a test. Higher throughput and lower latency and lower total runtime values are better from the performance viewpoint.

There are two categories of NoSQL database benchmarking tools: database-specific and database-independent. In the database-specific category we can mention `cassandra-stress tool`[19] for Cassandra, and `cbc-pillowfight`[18] for Couchbase. These tools cannot be used in our study, which aims to compare performances of different databases on the same workload. In the database-independent category we mention `YCSB`[4] and `BigBench`[9]. `YCSB` runs on both Windows and Linux, while `BigBench` runs only on Linux. Moreover, `YCSB` offers more flexibility than `BigBench` in terms of data models and workload configurations. The fact that `BigBench` resembles `TPC-DS`[15] makes it less oriented on big data or NoSQL workloads, while `YCSB` focuses on NoSQL systems and their specific workload types. Also, `YCSB` can be used to test the performance of many NoSQL DBMSs, including MongoDB and Cassandra, which makes it a good choice for our study.

`Yahoo! Cloud Serving Benchmark`[4] (or `YCSB`) is an open source benchmarking framework for NoSQL and cloud systems. It was developed in Java and includes two main components: the workload generator known as the

`YCSB` client, and the Core workloads that represent a set of workload scenarios to be executed by the generator[23]. These components can be extended. In the context of `YCSB`, a workload has two components: a data set and a transaction set. The *data set* represents the set of records that are loaded into the database before any transaction is performed on it. The *transaction set* contains all read and write operations to be run on the database server. Main parameters of the transaction set are: the number of operations in it (i.e. the number of operations to be performed in a test run), the ratio between read and write operations, and the number of client threads.

Besides Core workloads, `YCSB` allows the user to define new workloads. For our performance benchmark, `YCSB` version 0.12.0 was used as benchmarking framework. In the literature, there are other benchmarking studies that use `YCSB`: [7], [8] and [12]. These studies employ a testing environment that involves a cloud-based infrastructure, which greatly differs of our approach. Our testing environment is not cloud-based, and it uses Windows operating system on both client and server machines. Other differences refer to the use of physical machines instead of virtual ones, DBMS versions, operating system, workload types, the size of the data sets, and hardware configurations. The case study presented in [12] uses a proprietary data set that belongs to a healthcare organization and custom workloads. The data sets used in [7] and [8] are generated by the `YCSB` client, but the actual size of the data sets used in [8] is not clearly specified. The study presented in [7] specifies the size of the data sets used, but it does not include MongoDB in the study, it only includes Datastax's variant of Cassandra. Also, the software versions of the DBMSs and `YCSB` used in [7], [8] and [12] are older than those used in our study and, especially in the case of MongoDB, the difference between version 2 and version 3 is very significant. Starting with version 3, MongoDB uses a different storage engine, called Wired Tiger, which greatly improves performance.

3. Case study

3.1. **Experimental setting.** Our experiment used a total of three servers having the same hardware configuration, each of them running a different application: `YCSB` client on the first server, Apache Cassandra on the second one and MongoDB on the third. The configuration of each server is as follows:

- HDD: 500 GB
- RAM: 16 GB
- CPU: Intel(R) Core(TM) i7-4770 CPU @ 3.40GHz, 8 logical processors, 4 cores
- OS: Windows 7 Professional 64-bit.

All workloads used the same data set, composed of 4 million records. The `YCSB` client was used to generate the data set. Each record contains 10 fields and every field stores a 100 byte string value that was randomly generated. The data set could fit within internal memory due to its size. The chosen workloads belong to the `YCSB` Core workloads set: Workload A (the ratio between read and write operations in the transaction set is 1 - 50% updates, 50% reads), an *update-heavy* workload[20], and Workload B (the ratio between read and write operations in the transaction set is $\frac{1}{19}$ - 5% updates, 95% reads), a *read-mostly* workload[20]. An application that has a workload similar to Workload A could be a session store in which recent actions are recorded, while Workload B is similar to the workload of an application that involves photo tagging, as stated in [20]. Every workload was tested with the following values for the number of operations (NO): 1000, 10000, 100000 and 1000000, and with 1, 2, 4, 8, 16, 32, 64, 128, 256 and 512 client threads (NT). Each test was repeated three times.

MongoDB was installed with default settings, which implies that the storage engine used is Wired Tiger (the default storage engine for MongoDB version 3.4.4).

Cassandra was also installed with default settings, but we followed a setting recommendation found in [7], so that write timeouts can be avoided:

- `write_request_timeout_in_ms` increased to 100000
- `read_request_timeout_in_ms` increased to 50000
- `range_request_timeout_in_ms` increased to 100000
- `counter_write_request_timeout_in_ms` increased to 100000.

The asynchronous Java driver was used for both MongoDB and Cassandra. A batch of tests contains all tests with the same workload, number of operations, and database server, varying the number of client threads. Before the execution of every batch of tests, the database server was restarted. Information about database server status was captured before running a batch of tests and after. The data set corresponding to the first workload was deleted after all combinations of tests related to it were executed. A data set having the same parameters but corresponding to the second workload was loaded before executing the tests for the second workload.

3.2. **Results.** Every batch of tests was repeated three times, and the average values of average update latency (AUL) and average read latency (ARL) were computed. Figures 1 and 2 show the AUL results for Workload A and Workload B, respectively. Figures 3 and 4 refer to ARL. Here the x-axis represents $log_2(NT)$.

When testing Workload A (Figure 1), the evolution of AUL with respect to the number of threads has a quasi-parabolic shape, with the exception of the case with 1000 operations. There is an optimal number of threads thr_{min} for which AUL has a minimum value. This point is slightly shifting to the right as the number of operations grows. This leads us to the conclusion that for a fixed number of operations there is a threshold thr_{min} producing the minimum AUL value. When the number of threads exceeds this thr_{min}, AUL grows: in the case of Cassandra, the slope of the curve decreases with the number of operations; in the case of MongoDB, the slope is increasing as the number of threads grows. In the case with 1000 operations, the minimum AUL value is obtained for $thr_{min} = 2^2$, and the growth at some maximum value ($thr_{max} = 2^7$ for Cassandra, $thr_{max} = 2^8$ for MongoDB) after which it starts diminishing. MongoDB and Cassandra produce almost the same AUL value for 10000 operations when NT $\geq 2^5$. For a number of threads less than 2^8, AUL for 10000 operations is greater than AUL for 100000 operations in the case of MongoDB. The maximum AUL value is obtained for 10000 operations (Cassandra), respectively 1000000 operations (MongoDB).

For Workload B tests (Figure 2), the quasi-parabolic shape is preserved, again with the only exception of 1000 operations test case. The threshold thr_{min} has a slower shift to the right than in the case of Workload A. The way AUL values grow is the same for both workloads in the case of Cassandra, while in the case of MongoDB the AUL growth rate is slower for Workload B than in the case of Workload A, and AUL slightly diminishes with the number of operations. In the case with 1000 operations, the minimum AUL is obtained for $thr_{min} = 2^1$ (MongoDB), respectively $thr_{min} = 2^2$ (Cassandra), and the growth stops at $thr_{max} = 2^8$ (for both databases). In the case of MongoDB and Workload B, almost identical AUL values were obtained when number of operations was 100000, respectively 1000000. The maximum AUL value is obtained for 1000 operations and 2^8 client threads for both databases (AUL = 20588 for Cassandra, and AUL = 11854 for MongoDB). The worst performance in terms of AUL for Cassandra is almost double the similar MongoDB's one.

Figures 3 and 4 show the variation of ARL for workloads A and B. At first sight, the shapes are almost identical with those referring to corresponding AUL values. For 1000 operations and Workload A, the maximum ARL value is obtained for $thr_{max} = 2^7$ (Cassandra), respectively $thr_{max} = 2^8$ (MongoDB), while in the case of Workload B $thr_{max} = 2^8$ for both databases. For all cases considered when testing Workload A, maximum ARL value is obtained for $thr_{max} = 2^9$ with 1000000 operations (MongoDB), respectively 10000 operations (Cassandra). For Workload B, global ARL maximum values are obtained in the case $thr_{max} = 2^8$ and 1000 operations for both databases.

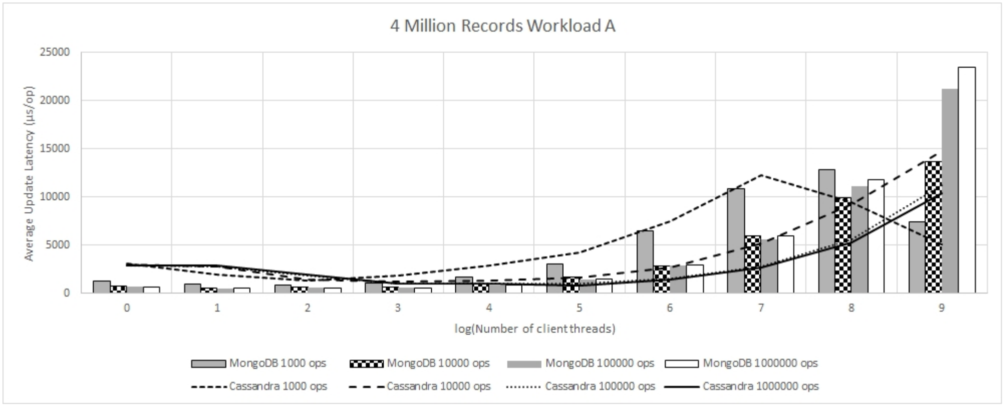


FIGURE 1. 4 Million Records Workload A - Average Update Latency

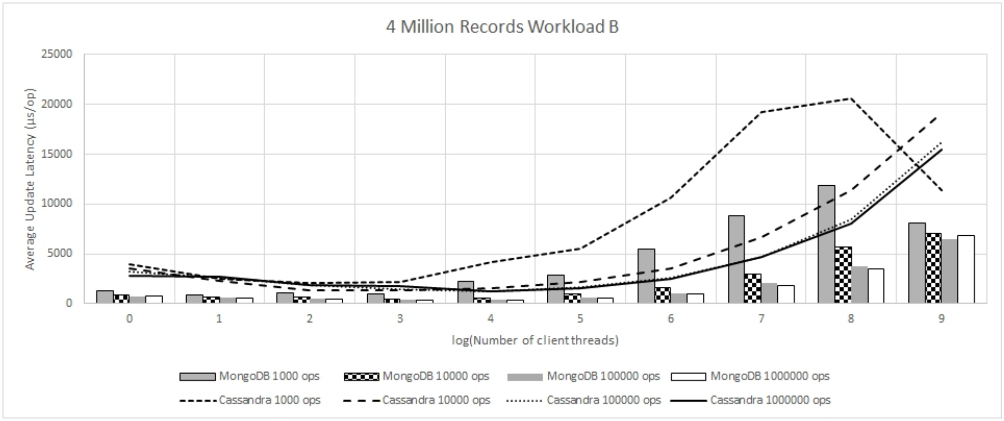


FIGURE 2. 4 Million Records Workload B - Average Update Latency

TABLE 1. Analysis of variance - update latency results

Wrk ld	No ops	Database			No of threads			DB:NT		
		F-value	Pr(>F)	Sgf	F-value	Pr(>F)	Sgf	F-value	Pr(>F)	Sgf
A	1000	0.1131	0.7378		20.0319	3.784e-05	***	2.0400	0.1588	
A	10000	3.8686	0.05416	.	851.0068	<2e-16	***	0.3264	0.57005	
A	100000	59.212	2.46e-10	***	2700.088	<2.2e-16	***	402.539	<2.2e-16	***
A	1000000	109.56	8.514e-15	***	2960.81	<2.2e-16	***	633.90	<2.2e-16	***
B	1000	8.6920	0.004654	**	26.1476	3.984e-06	***	0.7462	0.391370	
B	10000	196.81	<2.2e-16	***	1141.68	<2.2e-16	***	210.28	<2.2e-16	***
B	100000	315.01	<2.2e-16	***	1623.29	<2.2e-16	***	254.34	<2.2e-16	***
B	1000000	352.39	<2.2e-16	***	1879.47	<2.2e-16	***	242.42	<2.2e-16	***

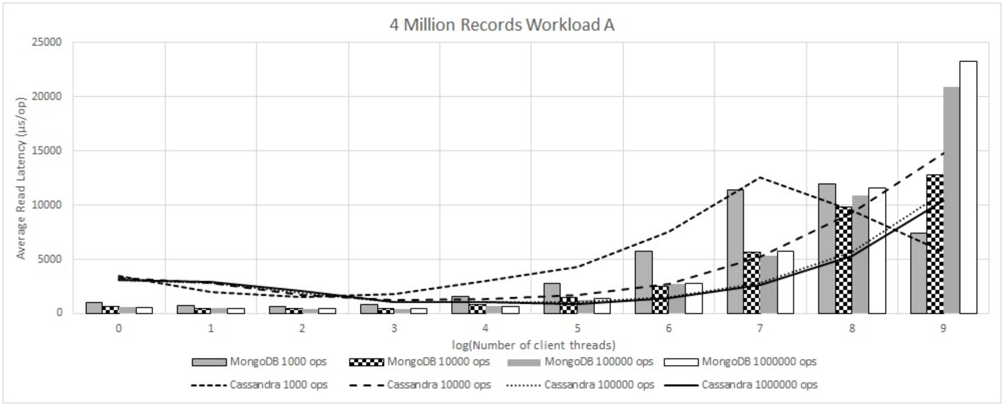


Figure 3. 4 Million Records Workload A - Average Read Latency

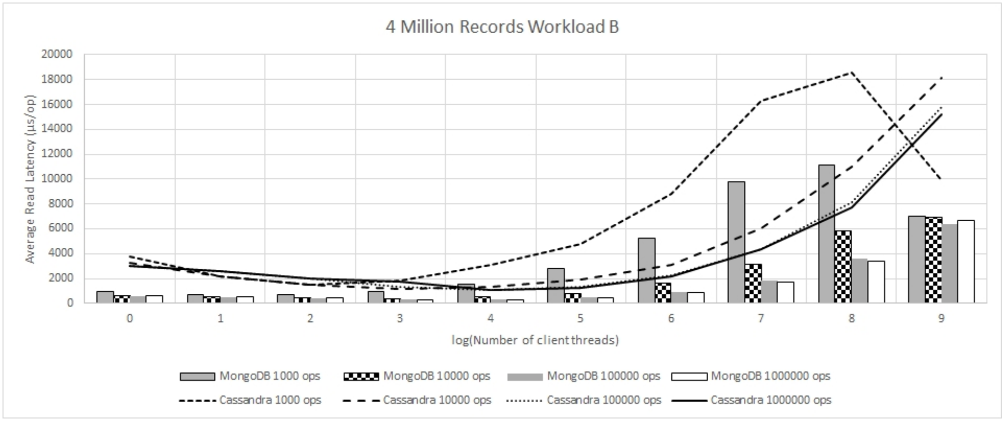


Figure 4. 4 Million Records Workload B - Average Read Latency

Table 2. Analysis of variance - read latency results

Wrk ld	No ops	Database			No of threads			DB:NT		
		F-value	Pr(>F)	Sgf	F-value	Pr(>F)	Sgf	F-value	Pr(>F)	Sgf
A	1000	0.6449	0.4253		21.0577	2.558e-05	***	1.7397	0.1925	
A	10000	9.1106	0.003821	**	727.9405	<2.2e-16	***	0.0037	0.952034	
A	100000	39.508	5.196e-08	***	2496.036	<2.2e-16	***	369.484	<2.2e-16	***
A	1000000	81.258	1.71e-12	***	2692.374	<2.2e-16	***	578.117	<2.2e-16	***
B	1000	6.3605	0.01454	*	25.9703	4.241e-06	***	0.6839	0.41175	
B	10000	160.54	<2.2e-16	***	1045.30	<2.2e-16	***	172.17	<2.2e-16	***
B	100000	267.69	<2.2e-16	***	1422.88	<2.2e-16	***	211.16	<2.2e-16	***
B	1000000	257.98	<2.2e-16	***	1381.54	<2.2e-16	***	170.57	<2.2e-16	***

3.3. **Statistical analysis.** The two-way ANOVA (Analysis of Variance) procedure from R Statistics Package[17] was used to perform the statistical analysis of the experimental results. Table 1 (for AUL) and Table 2 (for ARL) present a synthesis of the results. For every experiment, two factors were taken into consideration: database and number of threads. The interactions between database and number of threads (DB:NT) were considered as well. The database factor (DB) has two levels: Cassandra and MongoDB. The number of threads (NT) has ten levels: 1, 2, 4, 8, 16, 32, 64, 128, 256, and 512. The column labeled "Sgf" refers to the P-value and describes textually the level of significance, $0.1\%, 1\%, 5\%$, and 10%, following the usual conventions: $0 *** 0.001 ** 0.01 * 0.05$. 0.1 (*blank space*) 1. A P-value less than or equal to 0.1% (i.e. $***$ conforming to the legend) shows that the differences between means have a strongest statistical significance, while a P-value greater than 10% (i.e. blank space) indicates that the differences between the means of the levels considered are within the experimental error.

With respect to the number of threads, the differences in variation of AUL and ARL have the strongest statistical significance for both workloads. In the same time, for 1000 operations and Workload A, the differences of means with respect to DB levels for AUL and ARL are within the experimental error. All interactions DB:NT lead to strongest significance between means, except for Workload A with 1000 and 10000 operations and Workload B with 1000 operations, for which there is no statistical significance.

4. Conclusions and further work

As we stated above, the performance of Cassandra and MongoDB database servers was measured for two different workloads: update-heavy (Workload A) and read-mostly (Workload B) and two performance indicators were measured and analyzed, average update latency (AUL) and average read latency (ARL). All test cases with NO $\geq$ 10000 proved a quasi-parabolic behavior of AUL and ARL with respect to NT. This means that the best performance is achieved with a right combination of NO and NT. As NT exceeds these optimal values, the performance in terms of latency diminishes. In the case NO = 1000, the minimum AUL is obtained at $thr_{min} = 2^2$ for Workload A and both databases, respectively at $thr_{min} = 2^1$ (MongoDB) and $thr_{min} = 2^2$ (Cassandra) for Workload B. There is also a global maximum AUL value obtained for Workload A at $thr_{max} = 2^9$ and NO = 10000 for Cassandra, respectively at $thr_{max} = 2^9$ and NO = 1000000 for MongoDB. For Workload B, maximum AUL values were obtained at $thr_{max} = 2^8$ and NO = 1000 for both databases. These figures are almost the same for ARL, basic trends being preserved. Global maximum

ARL values are obtained for the same combination of (DB, workload, NO, NT).

As further work, we intend to analyze other metrics obtained from this experiment, and to perform post-hoc ANOVA tests. Also, we plan to perform other experimental studies using data sets that do not fit within the internal memory on cluster and single server configurations. Another direction in the experimental work will deal with the use of SSDs as disk storage and the replication for database servers, in order to measure how these configurations affect performance. Finally, another variable in our future case studies will be the operating system, so that we can use other NoSQL DBMSs that are not available on Windows.

Acknowledgments

Parts of this work were supported through the MADECIP project *Disaster Management Research Infrastructure Based on HPC.* This project was granted to Babeş-Bolyai University, its funding being provided by the Sectoral Operational Programme *Increase of Economic Competitiveness*, Priority Axis 2, co-financed by the European Union through the European Regional Development Fund *Investments in Your Future* (POSCEE COD SMIS CSNR 488061862).

References

[1] C. F. Andor and B. Pârv. NoSQL Database Performance Benchmarking - A Case Study. *Studia Informatica*, LXIII(1):80–93, 2018.

[2] Apache Cassandra. `http://cassandra.apache.org/`. Accessed: 2017-09-25.

[3] F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. E. Gruber. Bigtable: A Distributed Storage System for Structured Data. *OSDI '06 Proceedings of the 7th USENIX Symposium on Operating Systems Design and Implementation*, 7, 2006.

[4] B. F. Cooper, A. Silberstein, E. Tam, R. Ramakrishnan, and R. Sears. Benchmarking Cloud Serving Systems with YCSB. *Proceedings of the 1st ACM Symposium on Cloud Computing*, pages 143–154, 2010.

[5] CouchDB. `http://couchdb.apache.org/`. Accessed: 2017-09-25.

[6] G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall, and W. Vogels. Dynamo: Amazon's Highly Available Key-value Store. *Proceedings of 21st ACM SIGOPS Symposium on Operating Systems Principles*, oct 2007.

[7] Fixstars. GridDB and Cassandra Performance and Scalability. A YCSB Performance Comparison on Microsoft Azure. Technical report, Fixstars Solutions, 2016.

[8] A. Gandini, M. Gribaudo, W. J. Knottenbelt, R. Osman, and P. Piazzolla. Performance Evaluation of NoSQL Databases. *EPEW 2014: Computer Performance Engineering, Lecture Notes in Computer Science*, 8721:16–29, 2014.

[9] A. Ghazal, T. Rabl, M. Hu, F. Raab, M. Poess, A. Crolotte, and H.-A. Jacobsen. BigBench: Towards an Industry Standard Benchmark for Big Data Analytics. *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, pages 1197–1208, 2013.

[10] HBase. `https://hbase.apache.org/`. Accessed: 2017-09-25.

[11] JSON. `https://www.json.org/`. Accessed: 2018-03-16.

[12] J. Klein, I. Gorton, N. Ernst, P. Donohoe, K. Pham, and C. Matser. Performance Evaluation of NoSQL Databases: A Case Study. *Proceedings of the 1st Workshop on Performance Analysis of Big Data Systems*, pages 5–10, 2015.

[13] A. Lakshman and P. Malik. Cassandra: A Decentralized Structured Storage System. *ACM SIGOPS Operating Systems Review*, 44:35–40, 2010.

[14] MongoDB. `https://www.mongodb.com/`. Accessed: 2017-09-25.

[15] R. O. Nambiar and M. Poess. The Making of TPC-DS. *VLDB '06 Proceedings of the 32nd International Conference on Very Large Data Bases*, pages 1049–1058, 2006.

[16] OrientDB. `http://orientdb.com/`. Accessed: 2017-09-25.

[17] R Statistics Package. `https://www.r-project.org/`. Accessed: 2017-09-25.

[18] Stress Test for Couchbase Client and Cluster. `http://docs.couchbase.com/sdk-api/couchbase-c-client-2.4.8/md_doc_cbc-pillowfight.html`. Accessed: 2019-01-03.

[19] The cassandra-stress tool. `https://docs.datastax.com/en/cassandra/3.0/cassandra/tools/toolsCStress.html`. Accessed: 2019-01-03.

[20] The YCSB Core Workloads. `https://github.com/brianfrankcooper/YCSB/wiki/Core-Workloads`. Accessed: 2017-09-25.

[21] XML. `https://www.w3.org/TR/2008/REC-xml-20081126/`. Accessed: 2018-03-16.

[22] YAML. `http://yaml.org/`. Accessed: 2018-03-16.

[23] YCSB Github Wiki. `https://github.com/brianfrankcooper/YCSB/wiki`. Accessed: 2017-09-25.

Department of Computer Science, Faculty of Mathematics and Computer Science, Babeş-Bolyai University, Kogălniceanu 1, 400084 Cluj-Napoca, Romania
Email address: {`andorcamelia, bparv, tzutzu`}`@cs.ubbcluj.ro`

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LXIV**, Number 1, 2019
DOI: 10.24193/subbi.2019.1.05

COGNITIVE MODELING APPROACH FOR DEALING WITH CHALLENGES IN CYBER-PHYSICAL SYSTEMS

RÓBERT ADRIAN RILL$^{(1,2)}$ AND ANDRÁS LŐRINCZ$^{(1)}$

Abstract. In this paper, inspired by our previous works, we propose an architecture for the design and realization of cyber-physical systems (CPS) that considers the spatio-temporal context of events, promotes anomaly detection, facilitates efficient human-computer interaction and is capable of discovering novel human and/or machine knowledge. We view deep neural networks as smart sensors and sensory data from the environment represents the semantic and episodic input to a consistency seeking component of the cyber-space. Starting from a knowledge base infused with a deterministic world assumption, this module can detect anomalies and correct estimation errors by combining the outputs of multiple sensors. We also exploit an episodic description of ongoing situations by integrating temporal segmentation with kernel and low-dimensional embedding based methods. We demonstrate parts of the architecture through illustrative examples on our self-collected driving dataset. Our framework can be related to cognitive science foundations and may facilitate reliable functioning of CPS through integrating traditional AI and deep learning methods with deterministic models and reasoning tools. We expect that such knowledge base and cognition driven approaches of joining deep neural networks will be adopted in complex CPS. This looks like a scalable, and beneficial match between human knowledge and the exploding deep learning technologies.

1. Introduction

Cyber-physical systems (CPS) are complex structures of interacting physical and computational components, where the physical processes are controlled or monitored by computer-based algorithms. They are networks of sensors and robotic components equipped with advanced mechanisms and managed by intelligent software solutions, often including humans in the control loop.

Received by the editors: December 18, 2018.

2010 *Mathematics Subject Classification.* 68T01, 93A15.

1998 *CR Categories and Descriptors.* C.3 [**Computer Systems Organization**]: Special-purpose and Application-based Systems; I.2.0 [**Computing Methodologies**]: Artificial Intelligence – *General*.

Key words and phrases. cyber-physical system, human-computer interaction, anomaly detection, episodic description, consistency seeking.

Requirements of a CPS include usability, functionality, robustness, efficiency, adaptability, safety and reliability. The design and realization of CPS call for the integration of theoretical models and engineering techniques originating from different disciplines. Examples and areas of application include autonomous driving systems, transportation, smart factories, intelligent manufacturing [12, 13], healthcare [25], civil infrastructure [3] (electrical power grid, water resources, communication systems, networked building control), assisted living (consumer appliances, intelligent homes), entertainment.

The digital revolution of the last decades has governed the long-term technological and economical trends of CPS. This approach integrates production in an intelligent computational space, leveraging the interconnectivity of self-adaptable machines, paving the way for the next generation manufacturing, namely Industry 4.0 [2], with a significant economic potential [13].

As multidisciplinary systems, CPS combine computation, communication and control technologies to conduct feedback control on widely distributed embedded computing systems [15]. They operate with inputs and feedback from/to the physical environment, which calls for workflow management with real-time performance requirements. Communication is realized in the form of sensor-actuator networks. The information represents the abstraction of the physical world and operations are events composed of states reported by sensors/humans and actions performed by actuators/humans. CPS realize the autonomous networking of embedded systems at large scales, and their interaction with environmental processes. This poses considerable challenges.

In this paper we build upon our previous works. In [16] we presented a framework of cognition that combines deep neural networks (DNNs) with facts and rules to correct recognition errors and obtain consistent and deterministic event descriptions. In our follow-up work [17], based on motivations from neuroscience and psychology, we introduced an architecture called *declarative description* that combines DNNs and unsupervised machine learning techniques to provide explanation and reasoning about ongoing situations in a spatio-temporal context. The contribution of the present paper consists in combining the concepts of our former efforts to propose a general architecture for the design and implementation of CPS. Our framework tackles the challenges of CPS: anomaly detection, efficient human-computer interaction and deterministic decision making and description of events. Furthermore, we view DNNs as smart sensors and illustrate parts of the architecture on our self-collected driving data as a simple CPS scenario.

The paper is organized as follows. Section 2 provides a background by reviewing challenges that CPS face. Section 3 presents the two components that extend the traditional sensor-controller-actuator CPS network, namely *consistency seeking* and *episodic description.* Section 4 describes our illustrative examples, followed by a discussion in Section 5. Finally, Section 6 concludes the paper.

2. Background: CPS challenges

2.1. **Anomalies and decision making.** CPS aim for a high level of certainty in a narrow context, i.e. they are goal-oriented, designed with a well defined purpose, and they tightly integrate physical and cyber system aspects both at design time and during operation [3]. However, CPS are exposed to unexpected events, i.e. anomalies, which need to be recognized and dealt with in real time and with extreme care in order to limit false alarms and unobserved faults.

Real-time responsiveness of CPS to the environment is also related to timing behavior, which emerges from the combination of software and hardware platforms [11]. Predictable and reliable real-time performance is difficult to achieve because of the lack of temporal semantics and adequate concurrency models in computing [1]. Lee [10] summarized this thought as "the program does not express aspects of behavior that are essential to the system". CPS must have unified time, trust quantification and communication mechanisms at the system level [15].

2.2. **Deterministic models: knowledge-base.** In order to realize CPS, we need smart analytical tools to transform experience-based knowledge into evidence-based decision making for sustainable and reliable operation [25]. Components of a CPS come from multiple vendors in diverse engineering disciplines with distinct domain expertise [11, 15]. A holistic approach is required that integrates the physical and computational infrastructures into one unified model [14] for supporting real-time, reliable and autonomous decision making.

It is the power of deterministic models that gave scientists the ability to design control systems [11]. In CPS simulation environments they facilitate control algorithm development and testing, before the deployment into hardware [22]. Starting from a knowledge base that includes domain expertise, constraints, reasoning tools, facts and rules of the physical world, the realization of predictable and understandable models is possible. Moreover, a comprehensive knowledge about its own dynamic structure and the infrastructure of the whole system results in self-monitoring and self-aware CPS [2].

A knowledge-based decision making component can deal with the dynamic behavior: (i) it can perceive and control the environment, analyze observations due to the intelligent data management capabilities of the cyber space, and (ii) can communicate in efficient ways with other modules through wireless sensor networks [3] and with humans through intelligent user interfaces [13].

2.3. **Deep learning in CPS.** Novel engineering solutions exploiting AI and deep learning are improving at a remarkable pace. AI-based technologies are being developed for smart machine control [12]. In a networked CPS setting with interactions with the physical environment and humans, conditions are dynamically changing. This requires greater flexibility in modeling and optimized decision making. The advancements of the past years creates an opportunity to add data driven intelligence to the CPS processes.

To complement the available knowledge base, one can make use of DNNs. In CPS the measurement and monitoring of physical processes (temperature, pressure or mechanical movements, for instance) are done by sensors which convert physical or electrochemical properties into an electrical output signal [2]. So-called *smart sensors* are utilized for monitoring and control mechanisms of the environment. They enable reliable, accurate and automated data collection with minimal maintenance efforts and flexible networking. In this sense typical DNNs are sensors that monitor characteristics of CPS components and the environment, and provide a signal as output, which serves decision making and control. In the rest of the paper, smart sensors and DNNs will be referred to collectively as *feedforward input-output (FIO)* systems.

To summarize, DNNs may extend the set of intelligent sensors for increasing flexibility and adaptability in CPS. However, it is still required that humans remain in the loop in order to complement autonomy technologies for maximizing performance and limiting risks [21].

2.4. **Humans-in-the-loop.** CPS often involve humans in decision making and control loops. Either (i) they have supervisory roles and directly control the system, or (ii) the system passively monitors humans, collects data to be analyzed and takes actions if necessary. There are also different levels of human control depending on how large the task load is for decision making and how active the involvement is in the autonomy of making decisions [21].

In either case, several challenges need to be tackled. The modeling of human behaviour is a difficult task due to the complex physiological, psychological and behavioral aspect of humans [19, 20]. Furthermore, the probability of human error causing a system failure can be high due to a variety of reasons [21]. Therefore, robust CPS systems call for real-time predictive models that are able to recognize dangerous situations, control the outcomes, maintain stability and accuracy and adapt to changing human behavior.

Human behavior models need to be incorporated into the system architecture itself and, as several researchers suggest, human interaction will have a critical role in the foreseeable future (see, e.g., [21]).

3. Methods: cognitive architecture for CPS

3.1. Events in deterministic environments. CPS deal with scheduled operations and involve decision making in a goal-oriented context, often with humans in the loop. Any operation in the physical system can be described as an initial state and an action to be executed leading to a final desired state, where the action may involve several sub-processes (see Figure 1). We call the transition from the actual state to another one after executing the action an *event*, or *episode*. Two remarks have to be made. Firstly, the final state may differ from the desired state giving rise to an error term. We will turn back to this point later in the paper. Secondly, episodes can have different properties depending on their interrelationships: they can follow each other, can be concurrent, can be composed of other smaller episodes, can be combined into higher-order episodes.

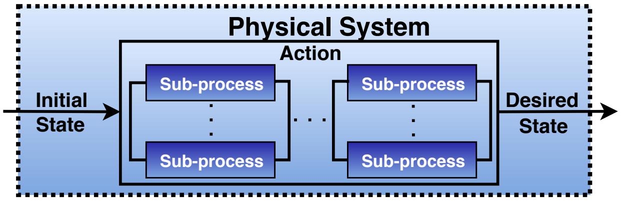


FIGURE 1. Operations in a physical system as events.

An event is independent if the state-action-state transition is not affected by other concurrent actions. An event is non-stochastic if the desired state is reached with 100% probability in the prescribed execution time. Deterministic behavior presumes independent and non-stochastic events. However, because of their complexity and interactions with the environment, events are less likely to be non-stochastic. Stochastic problems arise from anomalies, unpredictability of execution time and uncertainties from the environment, e.g. non-modelled side-effects of the surroundings.

Problem solving in such complex systems is a combinatorial problem and lowering of the number of variables is highly desired due to the exponential dependence of the state space on those. Because CPS are goal oriented, the deterministic world assumption points towards the ability of reaching goals and desired states with 100%. If anomalous events/episodes occur, deterministic models built upon the available knowledge base provide real-time decision making capabilities for recognizing and resolving them. Moreover, the spatio-temporal dependencies of the processes in a complex CPS constrain possible state-desired state pairs limiting the number of variables to be considered.

3.2. **Consistency seeking component of the cyber space.** In a typical CPS setting, sensors collect information from the environment, the cyber layer stores information and carries out abstract computations to examine the collected signals, controllers make decisions, which are transmitted to the actuators in order to change the physical processes [3, 14].

To extend the cyber space capabilities, we propose a consistency seeking module for decision making and control in a spatio-temporal context, making use of the available knowledge base. The contextual environment is given by sensory information – events taking place at a given location at a given time – and also incorporates the knowledge of experts. The sensory input is converted by the FIO systems into semantic and episodic input and this enters the consistency seeking module of the cyber space (see Figure 2). The semantic and episodic output of the consistency seeking component may be used to overwrite the collected signal to obtain consistent representations of the physical processes, or may be combined to produce the actual state. The difference between the actual and desired state gives rise to an error term, used to induce changes in the physical world.

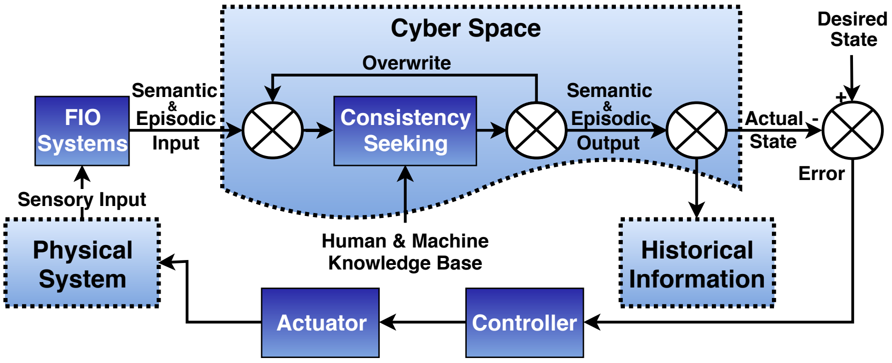


FIGURE 2. The traditional closed-loop sensor-controller-actuator CPS network extended with a consistency seeking component.

The consistency seeking component can facilitate anomaly detection. We propose to combine the outputs of multiple FIO systems to discover and resolve conflicts, to take into account the spatio-temporal context and the available knowledge base for improving recognition. Facts and rules, the structure and meaning of information, semantic relations among the components of the physical and cyber infrastructures can be injected into ontologies that also describe the interdependencies across the cyber-physical boundary [14].

The consistency seeking component of our architecture assumes a deterministic world. In stochastic environments other approaches might be necessary, e.g. probability maximization. However, if sufficient information is available, the deterministic world assumption holds, and prediction errors and persisting contradictions should lead to searches for finding the missing causes.

To summarize, consistency seeking and the assumption of determinism together become powerful tools for anomaly detection and learning, if we can recognize components and have episodic knowledge about their spatio-temporal relationships.

3.3. **Episodic description of ongoing situations.** Anomalies are changes in behavior that negatively affect performance. They are outliers that correspond to short time intervals within a larger episode, e.g. texting while driving.

CPS require real-time capabilities to keep updated about the current states of physical devices and to intervene if necessarry in applications such as observation, monitoring, control, forecasting [15]. Therefore, in a CPS setting, sensors continuously monitor physical processes such as traffic information in intelligent transportation, patients' blood pressure or blood sugar level in the healthcare domain, soil temperature and humidity in environmental detection. In the physical world the passing of time is inevitable and concurrency of processes is naturally present. Our framework integrates these properties in the computing capabilities of the cyber space.

Particularly, we propose to further extend the cyber space capabilities with an *episodic description* of ongoing situations illustrated on Figure 3. The sensory data and semantic and episodic output of the consistency seeking module are considered time series and temporal segmentation is applied. The result is a series of episodes, which can be consecutive or overlapping. Segmentation is followed by the comparison of episodes with each other, and the obtained similarity vectors are embedded into low-dimension. The resulting clusters can be further inspected by intelligent algorithms or humans using smart interfaces.

Episodic description could be applied for example in manufacturing, where most production planning decisions are based on historical data [2]. If past information of a production machine is provided in terms of time series, the episodic description module is capable to track the changes and infer additional knowledge by searching for similarities with other machine records and analyzing performance. This gives the possibility for the cyber space to predict future behavior. Another example is in the healthcare domain, where semantic knowledge needs to be integrated with manually and automatically collected low-cost clinical patient data towards clinical decision support [20, 25]. Temporal segmentation and/or clustering of episodes can be applied to historical information of patient data to analyze behavior patterns and predict possible future diseases or recovery status.

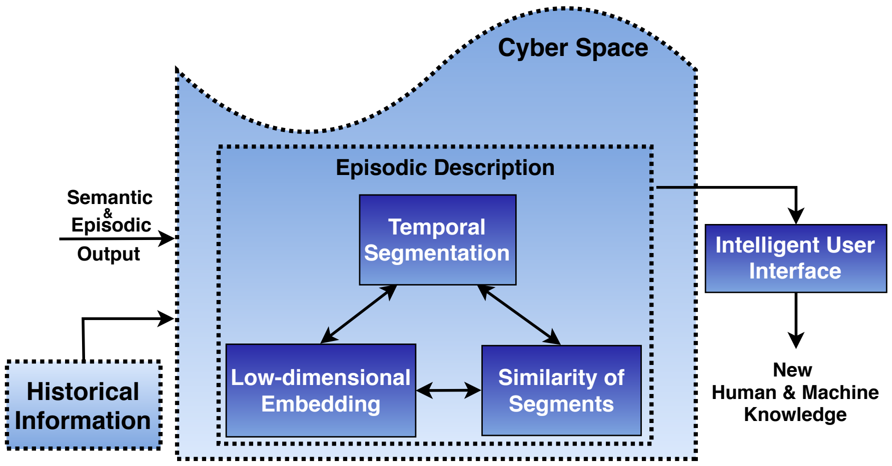

Cyber Space
Episodic Description
Temporal Segmentation
Low-dimensional Embedding
Similarity of Segments
Semantic & Episodic Output
Historical Information
Intelligent User Interface
New Human & Machine Knowledge

FIGURE 3. Extension of the cyber space with an episodic description component to detect anomalies and to bring temporality into focus.

4. RESULTS: DESCRIPTION OF ILLUSTRATIVE EXAMPLES

Autonomous driving represents one area of application for CPS [5, 7]. Our choice of the CPS aims the illustration of correcting sensory observations via consistency seeking. Our purpose was well served by means of self-collected data. Reasons include risk considerations, the lack of data about monitored humans, and the simplicity of illustration during driving, as a CPS scenario. We recorded videos while driving between Budapest and Martonvásár in Hungary, using a spherical camera attached to the dashboard of the car[1].

To illustrate the consistency seeking module of our architecture we extracted frontal view videos from our collected spherical videos and ran state-of-the-art DNNs on them. Figure 4 shows how the consistency seeking module can combine the outputs of two state-of-the-art DNNs to correct estimation errors and obtain new samples for fine-tuning: (i) the OpenPose version of the Convolutional Pose Machine (CPM) [24] is used to estimate body joint coordinates and (ii) the mismatches are corrected by predicting the movement of pixels from one frame to the next with optical flow, using the FlowNet2 version [9].

Figure 5 illustrates how the consistency seeking module can combine the outputs of DNNs to complement each other: (i) the state-of-the-art Yolo [23] object detector is used to recognize vehicles on videos; (ii) if the detection is lost on one frame, FlowNet2 [9] can be applied to predict the movement of the bounding box using the detection from the previous frame.

[1]Ricoh R Development Kit: `https://ricohr.ricoh/en/`

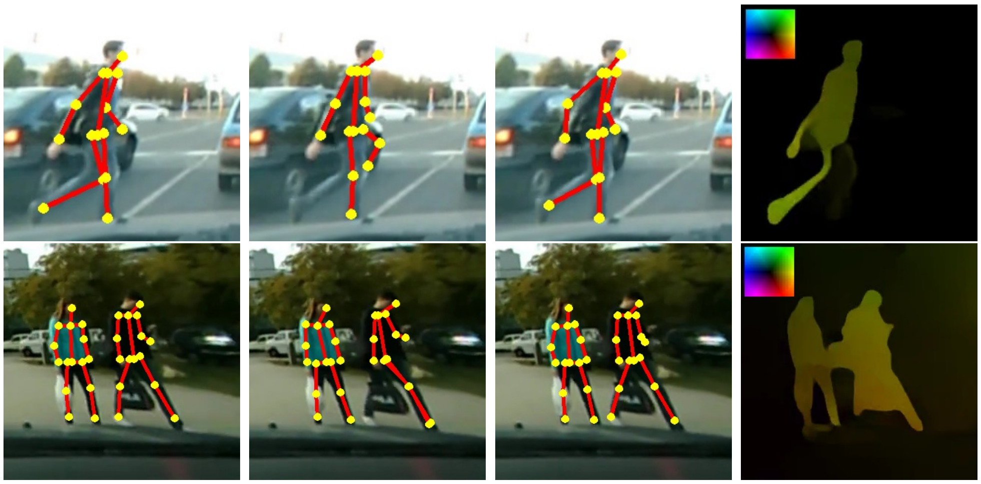


FIGURE 4. Correction of Convolutional Pose Machine (CPM) results by optical flow (OF). Two instances are shown in rows. The columns from left to right are: current frame with CPM visualized, next frame with mismatched CPM, same frame as second column with corrected CPM, and OF from current to next frame visualized. The square represents color coding of OF.

To demonstrate the episodic description component of our architecture we show how it can be applied to detect anomalies. Namely, we used it to recognize and cluster overtaking events, i.e. time segments when other cars pass by ours during driving. The spherical videos were pre-processed as follows: (i) extract side-views so that the car's side window is in the center of the frames, (ii) apply video stabilization to reduce the oscillation due to the movement of the camera, (iii) extract center of frames containing only the side window of the car, (iv) run FlowNet2 [9] on these small resolution videos and (v) compute OF features to obtain time-series subject to temporal segmentation. The features are 5-dimensional: for each frame we computed the two-bin histogram of the horizontal OF and choose the two bin heights and the three bin endpoints as features.

After the pre-processing procedure the steps of the episodic description were applied. Similarly to our previous works [16, 17, 18], for temporal segmentation we used the Group Fused LASSO [4] method to detect change-points co-occurring across the dimensions of multivariate signals. Then segments were compared with the Global Alignment Kernel (GAK) technique [6], able to compare different length segments by using time warping. Finally, the t-SNE algorithm [26] was applied to embed the columns of the similarity matrix into low-dimensions. Figure 6 shows an example of temporal segmentation on a

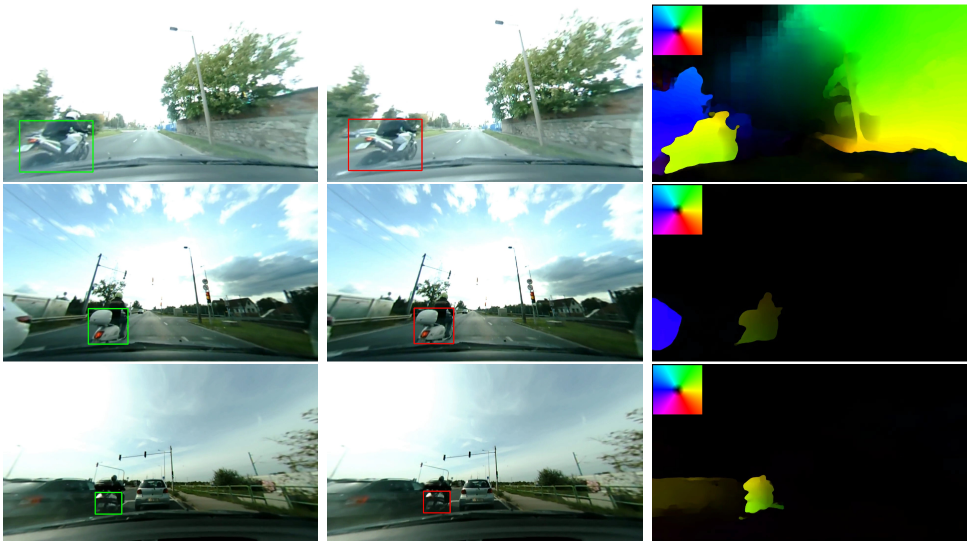


FIGURE 5. Complementing Yolo object detection by optical flow (OF). Three instances are shown in rows. The columns from left to right are: current frame with detected Yolo bounding box, next frame with predicted bounding box, OF from current to next frame visualized. The square represents color coding of OF.

short video, where three overtaking episodes (around frames 310, 790, 960) are clearly detected.

For the purposes of illustrating the results of the whole episodic description pipeline, we selected a 30 minute long driving video and hand-annotated the overtaking and the stopped segments (this latter refers to the time intervals when our car was stopped in traffic). The results are displayed on Figure 7, where each point corresponds to a time segment, as determined by temporal segmentation. There are a total of 44 overtaking episodes in this video. They are clustered together and separated from the rest to a large extent (red points).

It must be noted that long episodes can be segmented into multiple parts by the GFLASSO algorithm. Therefore, trajectories are also visualized on Figure 7, i.e. temporally consecutive points are connected with lines to reveal the difference between the three categories from the small cluster. Continuous colored lines connect consecutive segments of the same overtaking episode and they settle in the upper small cluster, with only one exception: in the lower right part of the figure the lightblue trajectory – this actually corresponds to two separate very long and consecutive overtaking events. All points from the small cluster corresponding to stopped segments (black) have their next

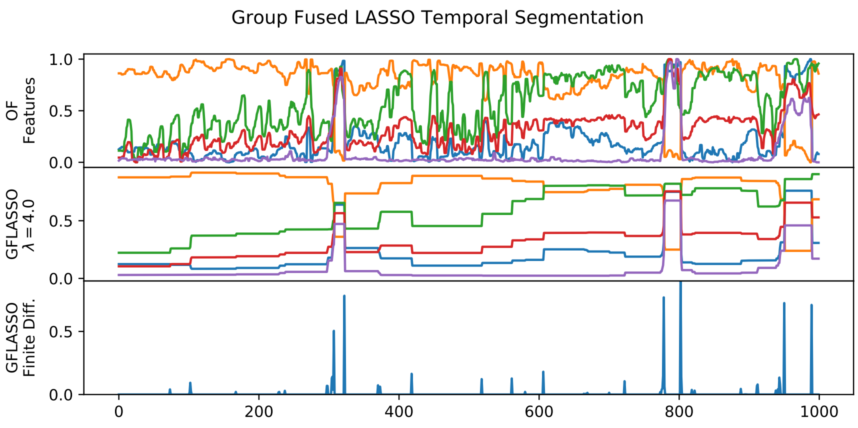


FIGURE 6. Example of Group Fused LASSO temporal segmentation on a 1000 frame long video. From top to bottom: raw Optical Flow (OF) features normalized, GFLASSO segmentation, finite differences of GFLASSO corresponding to change-points.

segment as stopped, connected by dashed grey lines (only three are shown for visual clarity). Dotted grey lines connect three other category points (blue) from the small cluster with their next segment: one is red inside and two are black outside the small cluster. Also three blue points from the small cluster have their next segment not overtaking nor stopped (these trajectories are not shown).

To sum up, assuming that stopped episodes are known and making use of trajectories, episodic description recognized the overtaking events with high probability as illustrated by Figure 7:

- 42 episodes out of 44 were separated from the rest in the small cluster (some of these episodes are composed from multiple segments but the trajectories settle in the small cluster);
- there are two false negatives (two long consecutive overtaking events connected by the lightblue line);
- there are three false positives (in the small cluster three blue points can not be excluded because their next segment is also blue).

Omitting the stopped segments from calculation and connecting the segments of the same event with trajectories, the problem of recognizing overtaking events can be treated as binary classification. Our relatively small example analyzed in detail happens to bring high accuracy (99%), precision (93%), recall (95%) and F1 score (94%) for classifying overtaking upon segmentation.

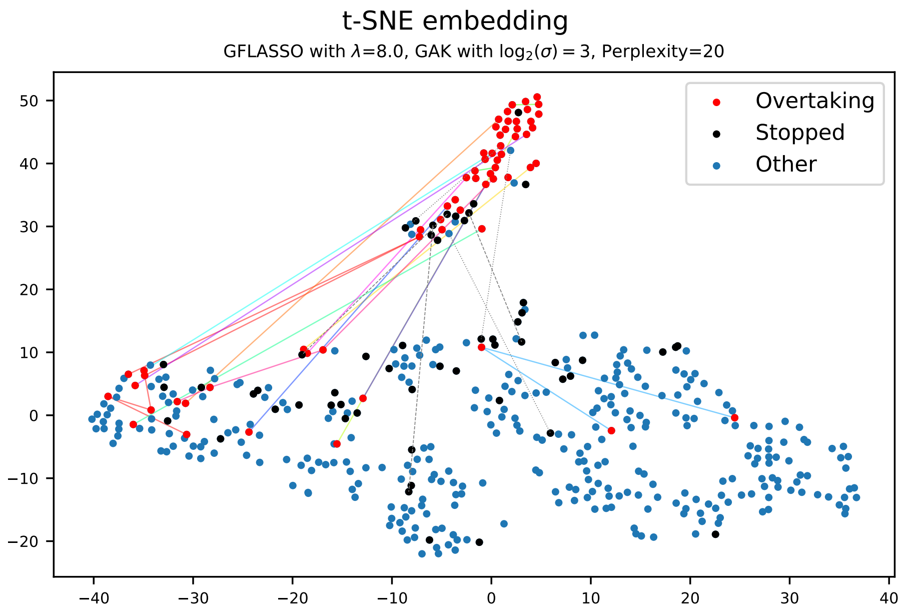


FIGURE 7. t-SNE embedding results for episodic description in the case of a 30 minute long video.

5. DISCUSSION

Our illustrative examples show how the consistency seeking module can combine the outputs of multiple FIO systems to correct estimation errors and to complement each other, and how the episodic description component can be used for event/anomaly detection. The overtaking episode represents a dangerous situation in an autonomous driving system and should be recognized in time in order to take appropriate actions.

The episodic description involves events that concern observations and interactions between components. The events are localized in space and time and reflect the physical reality. This enables reasoning at higher levels while integrating with a human/machine knowledge base. Not only does episodic description bring temporality into focus, but it has another important consequence. Although large volumes of data are collected in a CPS, not all of it is relevant, thus reducing the region of interest is essential [3].

The architecture presented in this paper tackles the problem of combinatorial explosion by limiting the number of variables to be considered during decision making in two ways: (i) the consistency seeking component considers the spatio-temporal context, which constrains the number of possible state-desired state pairs, and (ii) the episodic description contributes to eliminating irrelevant intervals of the state space.

To give a simple example, the spatially limited OF between a few frames is an instance of the extremes of short episodes. It limits efficiently the space of potential future outcomes due to Newtonian laws. In case of object tracking it means that a bounding box may not appear at any given position in the next frames. Thus the set of potential positions of the bounding box may be reduced considerably due to the "OF episode". More is expected for deterministic processes in the long run if the potential next episodes are learned and can be identified during progress.

High accuracy, recall and F1 score values for event recognition mean that high probability warning signals can be provided to the driver. Furthermore, the driver may train the system for personal use if their behavior is monitored. See for instance the examples provided in [16], which include talking to a person in the car or over the phone, or drinking. In this case the driver can train the system conditioned on their behavior, by providing feedback if the warning signal is not needed. Moreover, if the attention of the driver is overloaded then the warning signal may be desired in the same or similar situations. Feedback should help the step-by-step learning about the proper level of interaction. This can be the subject of future data collection, since the spherical camera enables the monitoring of the driver as well.

We close this section by noting that present CPS researches are still at the beginning, and are mostly limited to specific applications. Further investigations and studies are required to realize a unified framework of computational and physical resources [1, 15]. Since future CPS will cover aspects of social and economic lives [3], it is crucial to establish easy to abstract models so that the complexity of design can be reduced.

5.1. **Relation to cognitive psychology.** Our architecture can be related to fundamentals of the human cognitive system, namely declarative (semantic and episodic) and procedural memory (for details on this topic see, e.g., [8]). Semantic memory assumes a collection of facts and concepts encoded with specific meaning independently from the spatio-temporal context. Episodic memory on the other hand represents experiences and events in a serial form from which the situation can be reconstructed only if the surrounding context is also present. The procedural memory involves slow and gradual acquisition of skills that often occurs without conscious attention to learning. The concept of cognition is more than a single input-output mapping, it includes information processing, acquiring knowledge and the capability of reasoning.

In relation to our architecture, every event is an episode and it can be saved in the episodic 'memory' for data mining, anomaly detection, model construction, and for learning to predict and control the event. This event may be concurrent with other events and it is probably embedded into a larger one. The method of dealing with an ongoing event is the procedure, composed of actions and sub-events. This procedure may evolve over time as knowledge is collected. Reliable functioning is possible if the semantic knowledge base is large enough. If not, than new concepts, sensors and additional control tools can be introduced to overcome disturbances of the events provided that the details of the event are comprehensible, time is available and the related costs and savings justify the effort.

6. Conclusion

In this paper, inspired by our previous works, we presented an architecture for CPS that can be related to fundamentals of the human cognition system. To extend the traditional sensor-controller-actuator network, we proposed two novel components to enhance the cyber space capabilities: (i) consistency seeking and (ii) episodic description. The consistency seeking module is capable of detecting anomalies and correcting estimation errors by combining the outputs of multiple smart sensors and/or deep neural networks taking into consideration the spatio-temporal context and the available knowledge base (facts, rules of the physical world, domain-specific ontologies etc.). The episodic description module considers timing behavior, combines sparsity, kernel and low-dimensional embedding methods for anomaly detection, facilitates efficient human-computer interaction and is capable of discovering novel human and/or machine knowledge. By integrating deep learning and traditional AI methods with reasoning tools and deterministic assumption, our approach facilitates understandable and reliable decision making.

Our framework points to goal-oriented systems, an essential property of CPS. It considers anomaly detection and their resolution, i.e. noticing discrepancies in space and time and seeking for consistency based on the assumption of determinism. We expect that such knowledge base and cognition driven approach of joining deep neural networks will be adopted in complex CPS.

Acknowledgement

The research has been supported by the European Union, co-financed by the European Social Fund (EFOP-3.6.2-16-2017-00013, Thematic Fundamental Research Collaborations Grounding Innovation in Informatics and Infocommunications). AL was partially supported by the European Union, co-financed by the European Social Fund (EFOP-3.6.3-VEKOP-16-2017-00001).

References

[1] M. Atif, S. Latif, R. Ahmad, A. K. Kiani, J. Qadir, A. Baig, H. Ishibuchi, and W. Abbas. Soft computing techniques for dependable cyber-physical systems. *CoRR*, abs/1801.10472, 2018.

[2] C. Berger, A. Hees, S. Braunreuther, and G. Reinhart. Characterization of cyber-physical sensor systems. *Procedia CIRP*, 41:638–643, 2016. Research and Innovation in Manufacturing: Key Enabling Technologies for the Factories of the Future - Proceedings of the 48th CIRP Conference on Manufacturing Systems.

[3] M. Z. A. Bhuiyan, J. Wu, G. Wang, J. Cao, W. Jiang, and M. Atiquzzaman. Towards cyber-physical systems design for structural health monitoring: Hurdles and opportunities. *ACM Trans. Cyber-Phys. Syst.*, 1(4):19:1–19:26, 2017.

[4] K. Bleakley and J.-P. Vert. The group fused Lasso for multiple change-point detection. working paper or preprint, 2011.

[5] A. Chattopadhyay and K.-Y. Lam. Security of autonomous vehicle as a cyber-physical system. In *2017 7th International Symposium on Embedded Computing and System Design (ISED)*, pages 1–6, 2017.

[6] M. Cuturi. Fast global alignment kernels. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML'11, pages 929–936, USA, 2011. Omnipress.

[7] T. Dreossi, A. Donzé, and S. A. Seshia. Compositional falsification of cyber-physical systems with machine learning components. *CoRR*, abs/1703.00978, 2017.

[8] M. W. Eysenck. *Fundamentals of Cognition*. Psychology Press, 2012.

[9] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox. Flownet 2.0: Evolution of optical flow estimation with deep networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1647–1655, 2017.

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

[11] E. A. Lee. Cyber-physical systems: A rehash or a new intellectual challenge? Invited Talk in the Distinguished Speaker Series, sponsored by the IEEE Council on Electronic Design Automation (CEDA) held at the Design Automation Conference (DAC), Austin, Texas., 2013.

[12] J. Lee, B. Bagheri, and C. Jin. Introduction to cyber manufacturing. *Manufacturing Letters*, 8:11–15, 2016.

[13] J. Lee, B. Bagheri, and H.-A. Kao. A cyber-physical systems architecture for industry 4.0-based manufacturing systems. *Manufacturing Letters*, 3:18–23, 2015.

[14] J. Lin, S. Sedigh, and A. Miller. *A Semantic Agent Framework for Cyber-Physical Systems*, pages 189–213. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.

[15] Y. Liu, Y. Peng, B. Wang, S. Yao, and Z. Liu. Review on cyber-physical systems. *IEEE/CAA Journal of Automatica Sinica*, 4(1):27–40, 2017.

[16] A. Lőrincz, M. Csákvári, Áron. Fóthi, Z. Á. Milacski, A. Sárkány, and Z. Tősér. Towards reasoning based representations: Deep consistence seeking machine. *Cognitive Systems Research*, 47:92–108, 2018.

[17] Z. Á. Milacski, K. B. Faragó, A. Fóthi, V. Varga, and A. Lőrincz. Declarative description: The meeting point of artificial intelligence, deep neural networks, and human intelligence. In *IJCAI/ECAI 2018 Workshop on Explainable Artificial Intelligence (XAI)*, 2018.

[18] Z. Á. Milacski, M. Ludersdorfer, A. Lőrincz, and P. Van Der Smagt. Robust detection of anomalies via sparse methods. In *Lecture Notes in Computer Science*, volume 9491, pages 419–426. Springer Verlag, 2015.
[19] S. Munir, J. A. Stankovic, C.-J. M. Liang, and S. Lin. Cyber physical system challenges for human-in-the-loop control. In *Presented as part of the 8th International Workshop on Feedback Computing*, San Jose, CA, 2013. USENIX.
[20] W. Nilsen, E. Ertin, E. B. Hekler, S. Kumar, I. Lee, R. Mangharam, M. Pavel, J. M. Rehg, W. Riley, D. E. Rivera, and D. Spruijt-Metz. *Modeling Opportunities in mHealth Cyber-Physical Systems*, pages 443–453. Springer International Publishing, Cham, 2017.
[21] W. D. Nothwang, M. J. McCourt, R. M. Robinson, S. A. Burden, and J. W. Curtis. The human should be part of the control loop? In *2016 Resilience Week (RWS)*, pages 214–220, 2016.
[22] N. Pedersen, T. Bojsen, and J. Madsen. Co-simulation of cyber physical systems with hmi for human in the loop investigations. In *Proceedings of the Symposium on Theory of Modeling & Simulation*, TMS/DEVS '17, pages 1:1–1:12, San Diego, CA, USA, 2017. Society for Computer Simulation International.
[23] J. Redmon and A. Farhadi. Yolo9000: Better, faster, stronger. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6517–6525, 2017.
[24] T. Simon, H. Joo, I. Matthews, and Y. Sheikh. Hand keypoint detection in single images using multiview bootstrapping. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4645–4653, 2017.
[25] D. Sonntag, S. Zillner, S. Chakraborty, A. Lőrincz, E. Strommer, and L. Serafini. The medical cyber-physical systems activity at EIT: A look under the hood. In *2014 IEEE 27th International Symposium on Computer-Based Medical Systems*, pages 351–356, 2014.
[26] L. van der Maaten and G. Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605, 2008.

(1) Faculty of Informatics, Eötvös Loránd University. H-1117 Budapest, Pázmány P. stny 1/C, Hungary. 3in Research Group, Martonvásár, Hungary.

(2) Faculty of Mathematics and Computer Science, Babeş-Bolyai University. 1 Mihail Kogălniceanu, RO-400084 Cluj-Napoca, Romania.

Email address: rillrobert@cs.ubbcluj.ro (ORCID iD: http://orcid.org/0000-0002-3004-7294), lorincz@inf.elte.hu (ORCID iD: https://orcid.org/0000-0002-1280-3447)

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LXIV**, Number 1, 2019
DOI: 10.24193/subbi.2019.1.06

EMBEDDED SYSTEMS WITH COMPONENT-BASED GPU SUPPORT: A STATE OF THE ART

ANTONIU MICLĂUŞ, ŞERBAN PETRESCU, AND ANDREEA VESCAN

Abstract. In order to deal with extremely large quantities of information, embedded systems need high capabilities in order to process the whole amount of data in real time. Two trends are present in the field: the usage of boards with Graphics Processing Units (GPUs) and the usage of component-based development (CBD). Components with GPU capabilities have the great advantage to be platform-independent. However, developing embedded systems with GPUs by using CBD was considered until very recently a problem with restricted availability and flexibility. By introducing specific GPU support for CBD in the form of flexible components and by improving their communication, a solution was identified and checked. Present paper aims to present a state-of-the-art and highlights the newest knowledge to date, articulating encountered confronted issues and describing existing solution approaches.

1. INTRODUCTION

Many modern embedded systems deal with huge amounts of data originating from the interaction with the environment. For example, the autonomous car developed by Google 1 processes up to 120 MB of data per second delivered through its sensors [1]. The data must be processed with a certain performance in order to handle, in real-time, the environment changes.

A solution to process these data with adequate performance is the usage of general-purpose Graphics Processing Units (GPUs), which, thanks to their architecture, excel for highly data-parallel applications. Today, embedded-board platforms contain GPUs and different platforms have different architectures. Depending on their characteristics, like size, energy consumption or computation power, different platforms are suitable in different contexts. For example,

Received by the editors: January 21, 2019.
2010 *Mathematics Subject Classification.* 68U01, 68M99.
1998 *CR Categories and Descriptors.* I.3.2 [**Computer Graphics**]: Graphics Systems – *GPU*; C.3 [**Special-Purpose and Application-Based Systems**]: Real-time and embedded systems.
Key words and phrases. GPU-capability, component-based development, flexible component, allocation optimization.

there are platforms with high-computation GPU used in high-performance computing solutions, but also low-computation with low energy consumption such as GPU used in smart watches [3].

An alternative approach in the development of embedded systems is the usage of component-based development (CBD). CBD is a software engineering methodology that promotes the efficient system development through the composition of already existing software blocks called (software) components. CBD advertises the use and reuse of the same component in different contexts, which increases the development efficiency.

CBD is ineffective for embedded platforms that combine Central Processing Units (CPUs) and GPUs. This is due to the lack of specific support for GPUs. This overall challenge has several aspects. One of them refers to the development of components with GPU capabilities, which is complex, time-consuming and error-prone.

Another existing issue involves the reduced flexibility of the current way in which component-based applications with GPU capabilities are designed. The existing hardware-specific components have a reduced reusability between different hardware contexts.

The aim of this paper is twofold: firstly, a state of the art is provided, classifying the existing conducted research on CBD for GPU and augmented it with additional newest published approaches in the last year, and secondly to highlight the existing solutions for the encountered issues relating to usage of GPUs.

The reminder of the paper is organized as follows. A review of current contributions in the area of embedded systems with GPU is presented in Section II, followed by a focused overview in Section III, where only mechanisms to ease CBD for embedded systems with GPU capabilities are treated. The end of Section III also contains the related work. Section IV extracts the conclusions of the reviewing work.

2. EXISTING RESEARCH ON EMBEDDED SYSTEMS WITH COMPONENT-BASED GPU

Campeanu [4] conducted a Systematic Literature Review (SLR) and investigated existing studies related to CBD and GPU aspects. In his study 49 papers were considered and, from those, only 17 were devoted to the area of embedded systems. It was shown that the development of CBD for GPU-capability applications was first approached/published in 2009, and even today, this field is still poorly represented in the mainstream research for industry. The directions in which research on CBD for GPU was conducted were classified by Campeanu [4] to be: a) development improvement (33 papers);

b) performance improvement (10 papers); c) software-to-hardware allocation (5 papers); d) experience sharing knowledge (1 paper).

There resulted a number of gaps/needs to be approached in the future:

- generally, no specific component models were used to build the solutions; in the studies that however approached the area in such a way, the most used were PEPPHER, UML, CCA and Rubus;
- more than 10 mechanisms were implemented to support GPU development, from which most applied were the programming and modeling mechanisms;
- the memory addressing was approached by an artifact, manual- or layer-based solutions;
- preferred programming languages were CUDA/OpenCL and C/Cpp (CUDA – developed specifically for NVIDIA, while OpenCL – more general and fits to platforms like AMD, Altera, IBM, Intel, NVIDIA, Samsung and Xilinx).

Since the Systematic Literature Review provided in [4], several other papers investigated the GPU in connection with other perspectives as: parallel applications, multiple streams and process variations.

In what follows, we present several other contributions not presented in [4].

With the aim of stream computing for real-time sensor correction, authors of [5] proposed a flexible and expandable on-board real-time data processing solution. The data coming from a high-resolution optical satellite was chosen while the proposed solution was based on multi-threading optimization and a CUDA collaborative strategy. The simulation prototype was implemented on an NVIDIA embedded GPU platform and it consisted in a double-module data parallel pipeline system. Programming, occupancy and data access improvements were used and checked. The on-board results were at the end compared against the same algorithms run on a Dell PowerEdge T630 Server, proving a feasible stream performance and low power consumption. Due to the good flexibility and expandability of the embedded GPU platform, the idea could be shifted to cover different applications in which optimization strategies to be adjusted accordingly while the number of pipeline modules should be redrafted in function of the computational requirements. To improve the on-board intelligent processing capability, authors of [5] also proposed the implementation of other algorithms, eg.: fusion, geo-rectification, region of interest (ROI) extraction, cloud-cover detection, target recognition and change detection real-time processing.

In order to simulate parallel applications running on GPUs, the authors of [6], proposed the adaptation of the performance Volkov model and its implementation on a MERPSYS simulator. CUDA and NVIDIA GPUs are already available in the model, while extensions are expected soon, to include AMD and the OpenCL frameworks. GPU modeling was very proficient with the Volkov model implementation, from the extendibility and feasibility perspective. The advantages offered by using MERPSYS with this model was proved in the directions: a) it provides possibility to assess the applications functioning for sizes of the data that exceed the hardware capabilities; b) hardware setups can be evaluated prior to computations/applications running; c) costs may be apriority predicted since it is possible to calculate the duration of the computation; d) shortening of the simulation times than the real runs is obvious. The model could be verified on different GPU hardware architectures and could be improved further on, by using double precision units, SFUs and shared memory. Also, an automation of the kernel analysis process is expected for this solution.

To solve the complex problem of correct dissemination of multiple streams coming from various sensors in a system, a recent solution based on an original architecture was proposed: the Parallel Data Distribution Service (PDDS) [7], published in 2018. It proposed solving the problem timely, reliable and scalable. PDDS centers its idea on parallelizing the model-related computation. The state estimation of sensor streams was made by involving general-purpose GPUs (GPGPUs) to obtain high efficiency in energy and good scalability. Practically, an original data distribution algorithm has been implemented on a modern embedded device using CUDA by extending the data distribution service of the object management group. Three GPGPU kernels were involved: prediction, compaction and update kernels. With PDDS, serial network stacks could be bypassed and subscribers could have access to fresh sensor data by using local sensor models and with no communication with its publishers. It was proved that this algorithm consumes just 5% energy if compared to similar algorithms in use.

An interesting and complementary approach was recently addressed by analyzing the embedded GPU aging problem as a result of processes variations [8], published in 2018. An aging-aware workload management technique was used, in which the main actors were the warp scheduler and instruction dispatcher. The technique functions like this: before the launching of the kernel function, the host configures the GPU, taking into account the results from a running algorithm. This one refers to the formation of warp and to the workload division and generates information to reconfigure the cluster and scale the heaviness of the embedded GPU. It was proved experimentally that by using such a technique, GPU may excellently be reduced in (72-95) % of cases. Compared to the complier-based-technique, the present aging-preventing technique is less susceptible to soft-errors.

Since CBD is ineffective for embedded platforms that combine central processing units (CPUs) and GPUs, one solution is the development of components with GPU capabilities/settings and GPU-specific environment information. Encapsulating inside the components all the information results in specific hardware components destined to particular GPU architectures. There are two possibilities: a) specialized components made to encapsulate GPU functionality – but they cannot function without GPU hardware; b) introduction of special adapters to facilitate automatic data transfer between CPU and GPU memory system. Practically solutions are encountered in case of Autosar, Rubus and IEC 61131 [4]. The disadvantage is connected to the limitations of the system developer due to the reduced reusability of hardware-specific components in different hardware contexts.

In CBD, the components interact through interfaces: port-based and operation-based. Eg.: the port-based interfaces comprise of access points for sending/received data of different types between components. The white-box components are readable source code changeable by the programmers. The components' functionality is accessed through the interface, and their internals are visible from outside. The developer has access to their interface and internals. A component is constructed by following the specifications of a component model; it is well established how components interact with each other when they are assembled into a system.

3. PROPOSED SOLUTIONS FOR GPU SPECIfiC SUPPORT: THE PIPE-AND-fiLTER COMPONENT MODELS

This section emphasizes and presents the proposals to overcome the shortcomings identified in SLR [4].

The GPU, being the processing unit equipped with a parallel architecture, cannot function without a CPU. CPU coordinates all the GPU-specific activities (data transfer/execution of GPU functionality). Embedded-board platforms with different GPU architectures exist (2 types):

- discrete (dGPU) - has its own private memory (Condor GR23);
- integrated (iGPU) - on the same chip as the CPU, sharing the same memory (AMD Kabini4).

Embedded-boards with iGPU architectures are the predominant platforms in industry, low - priced, - sized and - energy usage. dGPUs have large physical sizes, incorporate more (GPU) resources and used by systems requiring high performance.

To develop an application, several hierarchical steps are taken: 1) a platform is set with installed driver (contains one or several execution devices, eg. 1-CPU and 2- GPU (iGPU and dGPU) devices); 2) the devices are selected so as to execute the functionality; the commands given by the host (CPU) to the device (iGPU)/kernel are sent using a command queue mechanism; the functionality should be defined before setting the platform; 3) allocation of device memory (buffers), either as input or output for the kernel function; 4) a program to hold the defined kernel is created and compiled (kernel arguments are assigned by using the allocated In/Out buffers) + specify the number of threads for the kernel execution; 5) the kernel is executed and its results are transferred back to the host; 6) the resources (memory buffers, program, context, command queue, kernel) are released.

The pipe-and-filter component models are based on [4]: a) flexible components; b) optimization of the groups of flexible components; c) a support for component communication is designed.

A flexible component, being a white-box with readable and modifiable source code, its functionality is expressed in parallel using the OpenCL syntax. It can be executed either on CPU or GPU. The component does not contain any environment-specific information (it is not binded to a particular processing unit). During system design, the system developer decides on which hardware (CPU or GPU) the flexible components should be allocated onto. In order to be executed on the specified hardware, the required environment information is generated automatically.

The accomplishment of the solution is implemented on two levels: the component- and the system-level. Using the core functionality and the information on the number and data types of the (In and Out) data ports provided by the flexible component, a full component was generated, ready to be executed on the hardware. The resulting generated component contains constructor + behavior function + destructor. At system level, based on the component connections and component-to-hardware allocation, artifacts/adapters were generated where needed. The adapters take data from one component and provide it to the connected component in the appropriate memory location

The use of flexible components having functionalities that may be executed either on CPU or GPU has the next advantages: 1) component-level mechanisms automatically generate environment-specific information that allows the component to be executed on different hardware; 2) system developer has a larger design-space to choose from; 3) the adapters automatically transfer data between components.

3.1. Flexible Component-based Applications with GPU Capabilities.

The approach presented in [9] aims at enhancing the flexibility in designing component-based applications with GPU capabilities by introducing flexible components that owe functionalities that may be executed either on CPU or GPU. In this way the developer may focus only on implementing the functionality while having a larger design-space to choose from. Component-level mechanisms automatically generate environment-specific information so that the component may be executed on different hardware. The adapters automatically transfer data between components, taking in consideration the platform specifications. The benefits of employing flexible components refer to: canceling the developer's responsibility of handling the component environment-specific information; providing a higher system feasibility due to a larger design-space; increasing the component communication efficiency by the generated adapters.

Supplementary to the solution proposed in [9], the authors of [10] underline the fact that the flexibility offered by component-based applications complicates the allocation process; it adds additional complexity (due to undecided CPU or GPU execution) and constraints to be considered (CPUs and GPUs properties). Therefore an optimization of the flexibility offered by the component-based embedded systems is necessary. Practically in [10] it is proposed a model to optimize the memory usage, the energy usage and the execution time. The novelty is provided in the formal description of the optimization model, which supports the usage of mixed integer nonlinear programming to compute optimal allocation schemes.

3.2. **Boosting the Resource Utilization.**

In order to mitigate the ever-increasing computational demands of modern embedded systems platforms equipped with GPU processors, an alternative solution is proposed in paper [11] by the boosting of the resource utilization of embedded systems with GPUs. Practically the idea is that the non-critical functions can benefit from the resources of the critical functions during the intervals when they are not used. The method provided in [10] allows the automatic computation of the unused resources in the critical part of the system followed by the distribution of the computed resources to the non-critical parts. The method makes use of a run-time monitoring engine that monitors the critical part of the system to detect any changes in its resource requirements. The considered run-time resources are the system memory and GPU computation threads. By calculating the unused memory based on the actual resource usage by the critical part of the system and by having the information regarding the amount of available memory, the non-critical part of the system can benefit from the available extra resource.

3.3. **Practical Demonstrations of Flexible Components Versatility.**

It was demonstrated that the pipe-and-filter style implemented by Rubus component model is suitable for streaming of events-type of applications and allows an easy mapping between the interaction model and the control specifications required by embedded and real-time systems [4]. The Rubus component consisted of 3 parts: constructor (executed once, before the system execution and allocates the component resource requirements), behavior function (functionality of the component, executed each time when the component is triggered) and destructor (execution when the system is switched off and releases the allocated resources).

In case of the vision system of an underwater robot [2], the hardware platform was an electronic board with a GPU, connected to various sensors (two cameras) and actuators (thrusters) [4]. The continuous flow of data produced by the cameras is processed by the robot's vision system using the GPU. Two camera components were connected to the physical camera sensors. The received data was converted into readable frames and forwarded to the Merge and Enhance component that merged and reduced the noise of the two received frames. The resulting frame was converted to grayscale. Due to the nature of computations (image processing), a set of flexible components were used: Merge And Enhance, Convert Grayscale, Edge Detection, Compress RGB and Compress Grayscale. The frames were of m-elem type, where the maximum size (RGB and grayscale) varied from component to component, depending on the functionality. To evaluate the approach, 4 allocation scenarios were used: 1) all flexible components are allocated to the GPU; 2) all flexible components are allocated to the CPU; 3) and 4) alternate in allocating the flexible components to CPU and GPU. For each scenario, there were used 3 different hardware platforms that contain GPUs. As an output, three produced frames were compared (the input to Object Detection and Logger) from all 12 combinations of scenarios and platforms; all combinations generated identical output frames. For scenario 1 (all flexible components allocated on GPU), for platforms with dGPU architecture, there were generated two CPU-to-GPU adapters and three GPU-to-CPU adapters. When all flexible components were allocated to CPU, there was no need for adapters. For shared virtual memory architectures, there were generated only CPU-to-GPU adapters; there was no need for GPU-to-CPU adapters because all components (regular and flexible) had direct access to the same shared virtual memory system.

Even if up until very recently CBD presented a very reduced attractiveness and solvability in the area of embedded systems with GPU, yet notably progress has been made in the last couple of years. For a compact overview of novelties in the field, Table 1 synthesizes the advancement in the field.

Issue / GPU mechanism	New solution	References
Flexible and expandable on-board GPU real-time data processing	Multi-threading optimization/ CUDA collaborative strategy	[5]
Prediction of application performance on various GPUs	Theoretical models embedded in MERPSYS	[6]
Parallel Data Distribution Service architecture	Parallelizing the model-related computation/ general-purpose kernels of GPUs/ extending the data distribution service of the object management group	[7]
Improving GPU aging process	Aging-aware workload management technique/ reconfigure the cluster and scale the heaviness of theembedded GPU	[8]
CBD: Platform independent components	Flexible component/executed on GPU or CPU /grouping/communication via adapters/ adapters generated automatically	[4], [9]
CBD: Rubus component model - extended	Implemented with flexible components/ groups/adapters	[4], [10]
CBD: GPU platforms with components application optimization	Method providing different allocation schemes for flexible components/ in function of optimization criteria	[4], [11]

Table 1. Development of embedded systems with GPU

4. CONCLUSIONS

Facilitation of component-based development of embedded systems with GPUs is a need in alternative finding of solutions for high-demand processing of huge data streams resulted from real-time environment sensor-systems. Even if considered until very recently as a limited/abandoned track, CBD contribution proves its high potential in specific contexts.

By starting with a review of state-of-the-art of embedded systems with GPUs, we initially classified the papers in the field - based on categories, mostly using the collected data presented in a doctoral thesis from 2018 [4] which extracted information from internationally recognized databases. Then we emphasized the newest ones, which have not been reviewed to date.

The main focus was however devoted to reviewing solutions in the area of CBD, where, when preparing the component with GPU capability one needs to take into account: on one hand, the component functionality, the required environment information and GPU settings, and on the other hand the separation between component and the system development.

By analyzing the concepts recently introduced of flexible components, then flexible groups and then optimized groups, a feasible solution was showed to emerge. The component communication was facilitated by using specialized artifacts/adapters that automatically transfer data between CPU- and GPU-allocated flexible components.

In view of recent solutions which were already implemented and tested, the component-based GPU support proves its power and advantages, as compared to other solutions analyzed above.

References

[1] K. Bimraw, *Autonomous cars: Past, present and future, a review of the developments in the last century, the present scenario and the expected future of autonomous vehicle technology*, The 12th International Conference on Informatics in Control, Automation and Robotics (ICINCO), Colmar, France, July, 2015, pp. 191–198.

[2] C.Ahlberg, L. Asplund, G. Campeanu, F. Ciccozzi, F. Ekstrand, M.Ekstrom, J. Feljan, A.Gustavsson, S. Sentilles, I. Svogor, and E. Segerblad, *The Black Pearl: An Autonomous Underwater Vehicle*, Technical report, Mälardalen University, Sweden, 2013.

[3] G. Keramidas, *Ultra Low Power GPUs for Wearables*, Think Silicon, http://lpgpu.org/wp/wp-content/uploads/2014/09/HiPEAC_wearables.pdf, Jan. 2015.

[4] G. Campeanu, *GPU Support for Component-based Development of Embedded Systems*, Ph. D. Thesis, School of Innovation, Design and Engineering, Mälardalen University Doctoral Dissertation 264, Sweden, 2018.

[5] M. Wang, Z.Q. Zhang, Y. Zhu, Z. P. Dong, Y.Y. Li, *Embedded GPU Implementation of Sensor Correction for On-Board Real-Time Stream Computing of High-Resolution Optical Satellite Imagery*, Journal of Real-Time Image Processing, vol. 15, no. 3, (2018), pp. 565–581.

[6] T. Gajger, P. Czarnul, *Modelling and Simulation of GPU Processing in the Merpsys Environment Scalable Computing-Practice and Experience*, Scalable Computing-Practice and Experience, vol. 19, no. 4, Special Issue: IS, (2018), pp.: 401–422.

[7] W. Kang, J. Kim, *PDDS: Scalable Sensor Data Distribution for Cyber-Physical Systems Using GPGPUs*, IEEE Internet of Things Journal, vol. 5, no. 3, (2018), pp. 2025–2036, 2018.

[8] H. Lee, M. Shafique, M.A. Al Faruque, *Aging-Aware Workload Management on Embedded GPU Under Process Variation*, IEEE Transactions on Computers, vol. 67, no. 7 (2018), pp. 920–933.

[9] G.Campeanu, J. Carlson, and S. Sentille, *Flexible Components for Development of Embedded Systems with GPUs*, 24th Asia-Pacific Software Engineering Conference (2017), p. 219–228.

[10] G. Campeanu, and S. Mubeen, *Scavenging Run-time Resources to Boost Utilization in Component-based Embedded Systems with GPUs*, Intl. J. Adv. Software, vol. 11, no 1 and 2 (2018), p. 159–169.

[11] G. Campeanu, J. Carlson, and S. Sentilles, *Allocation Optimization for Component-based Embedded Systems with GPUs*, The 44th Euromicro Conference on Software Engineering and Advanced Applications, Prague (2018), pp. 101–110.

Department of Computer Science, Faculty of Mathematics and Computer Science, Babeş-Bolyai University, 1 Kogălniceanu, Cluj-Napoca, 400084, Romania

Email address: {`mais1577, psis1589`}`@scs.ubbcluj.ro, avescan@cs.ubbcluj.ro`

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LXIV**, Number 1, 2019
DOI: 10.24193/subbi.2019.1.07

RUNTIME PERFORMANCE BENCHMARKING FOR NOSQL DATABASES

CAMELIA-FLORINA ANDOR AND BAZIL PÂRV

Abstract. An experimental study regarding the performance of NoSQL database management systems is presented in this paper. In this study, two NoSQL database management systems (MongoDB and Cassandra) were compared, and the following factors were considered: degree of parallelism and workload. Two distinct workloads (mostly read and update heavy) were used, and various numbers of client threads. The measured results refer to total runtime and they confirm that MongoDB outperforms Cassandra in almost all tests considered.

1. Introduction

The relational model was considered for a long time the default data model to use, as it is very well known and extensively used by many database professionals and companies worldwide. As the necessities of software applications evolve, the quantity of data to be stored grows and the diversity of data formats increases, which makes it harder to store and manage all that data in a relational system. The need for horizontal scalability and high availability of newer software applications were the main reasons that led to the emergence of NoSQL models, as the relational model was not a good fit in these cases. The main NoSQL data models are: key-value, column-family, document and graph. While plenty of NoSQL database management systems are offered by different providers, it is quite difficult to make a choice. Besides the fact that NoSQL database management systems are very different from one another regarding data models, query languages and distribution models, they are also highly configurable and very flexible. Therefore, a fair comparison between NoSQL database management systems is not a trivial task, as

Received by the editors: February 6, 2019.

2010 *Mathematics Subject Classification.* 68P15, 68P99.

1998 *CR Categories and Descriptors.* H.2.1 [**Database Management**]: Logical design – *Data models*; H.2.4 [**Database Management**]: Systems – *Distributed databases, Parallel databases*.

Key words and phrases. NoSQL database, performance benchmarking, MongoDB, Cassandra.

it requires in-depth knowledge about each one of them. In order to choose an appropriate NoSQL database management system for an application, the designer of the application must study thoroughly the technical documentations of several NoSQL products. Even after reading and understanding properly every aspect of each NoSQL database management system considered, it can be quite hard to make a fair comparison and take the right decision. Performance benchmarks are a good way to overcome the lack of comparison criteria between NoSQL database management systems. These benchmarks help application designers see the candidate database management system in action, allowing them to choose the appropriate hardware and software configuration. The current paper refers to total runtime metric and it is the third in a series evaluating performance metrics of two NoSQL implementations: MongoDB and Cassandra. All benchmarking tests were executed using `Yahoo! Cloud Serving Benchmark` with different combinations of number of operations, number of client threads and workload on every database server.

2. Background

2.1. **NoSQL data models.** Huge volumes of data that are generated with a high velocity, also known as big data, cannot be handled effectively with relational databases. As a response to this problem, companies that work with big data have created new data storage systems which are more flexible, non relational, distributed and highly available. Inspired by Amazon's and Google's non relational database systems, more and more companies built their own non relational implementations, specialized on their needs. These new non relational database systems are known today as NoSQL database systems, and the data models they are based on are known as NoSQL data models. These data models offer a flexible schema and the ability to store related data in a single unit with a complex structure, thus removing the need for join operations and increasing performance. The most important NoSQL data models are the graph model, the key-value model, the document model and the column-family model.

The graph model is the best fit for highly interconnected data. In this model, data are stored as nodes and relationships between nodes. Each node and relationship has a type and multiple properties, similar to a property graph. Graph queries can be very expressive and fast, but the highly interconnected nature of this model limits the horizontal scalability.

The key-value model can be considered the simplest NoSQL data model, because it stores data as key-value pairs. Key-value pairs can be grouped into containers called buckets, as stated in [19]. The key is the unique identifier of the value part that stores the actual data. The value can have a complex structure, but it is invisible at database level, therefore queries based on value are not supported. While query capabilities are quite limited, horizontal scalability is very well supported and database operations are fast.

The document model is somehow similar to the key-value model, because a document can be considered a key-value pair. Each document has a unique identifier and a value with a complex structure that is visible at the database level. Documents are organized in collections. Queries can be very expressive and horizontal scalability is easily supported, as there are no relationships defined between documents. In a document, objects or arrays are allowed to be stored as values for fields. Unlike the relational model, which imposes that all records stored in a table must have the same structure, the document model allows documents that have different structures to be stored in the same collection. One of the most used document formats is JSON[13], followed by XML[23] and YAML[24]. Related data can be grouped and stored in a single document that is similar to the object representation used at the application level, therefore the problem known as impedance mismatch[19] is removed and the application development process is significantly improved.

The model known as column-family organizes data as rows that belong to column families. There are some similarities between a relational table and a column family, but the latter allows arrays, lists or objects to be stored as values for columns and rows with different columns can be part of the same column family. A column is in fact a key-value pair stored together with a timestamp, and the name of the column represents the key part of the pair. Join operations are usually not supported by this model, and denormalization is common. Write operations are quite fast. A disadvantage of this model is that query capabilities are somehow limited and the most common queries must be taken into consideration in the design phase. Otherwise, horizontal scalability and high availability are easily supported.

2.2. **NoSQL tools.** The data models chosen for our benchmarking study are column-family and document. For each data model we chose a representative implementation: Apache Cassandra[1] as the column-family implementation, and MongoDB[16] as the document implementation. The column-family model is also implemented by HBase[11], Bigtable[3] and Hypertable[12]. For the document model, alternative implementations are: BaseX[2], CouchDB[6] and Couchbase[5].

Apache Cassandra is an open source column-family DBMS that was originally developed at Facebook[15]. Its data and distribution models are based on those used by Bigtable[3] and Dynamo[7]. Cassandra was designed to run in a distributed environment and has its own query language, called

CQL(Cassandra Query Language). The main features offered by Cassandra are automatic sharding, high availability, tunable consistency and data replication, including data replication across multiple data centers. In a Cassandra cluster, all nodes are equal and each one of them can accept both read and write operations. New nodes can be added to a cluster without downtime, and the cluster capacity for both read and write operations scales linearly. The version used in our study was Apache Cassandra 3.11.0.

MongoDB is an open source document DBMS developed by MongoDB Inc. MongoDB stores data as JSON-like documents and does not require a predefined schema. It has a rich query language based on JavaScript and ad hoc queries are well supported. The main features of MongoDB are automatic sharding, high availability, data replication (including multiple data center replication), tunable consistency and schema flexibility. MongoDB was designed to run in a distributed environment. In a MongoDB cluster, there are three types of nodes: shards, query routers and configuration servers. Shards store the actual data, while the cluster metadata is stored on the configuration servers. Query routers use cluster metadata stored on configuration servers to route queries to the corresponding shards. The version used in our study was MongoDB 3.4.4.

2.3. **NoSQL benchmarking.** When many NoSQL database management systems are available on the market, it can be hard to choose the right product that offers the best performance for a particular application use case on a specific hardware configuration. Benchmarking is a good approach in this case, as it measures the actual performance of a NoSQL implementation on a given hardware configuration. But the benchmarking process also requires software tools, and there are not so many open source options available.

A benchmarking tool has two main features: workload generation and performance measurement. A batch that contains all requests sent by an application to a database server during a working session represents the application's workload. The main metrics in database performance benchmarking are throughput, total runtime and latency. The number of operations completed in a time unit is known as throughput. The amount of time needed for a single operation to be completed is known as latency. The amount of time needed to complete a given number of operations is known as total runtime (RT). Throughput is measured in operations per second, while latency is measured in microseconds per operation. Higher throughput values are better from the performance viewpoint. Total runtime is measured in milliseconds and it represents the duration of a benchmarking test. Regarding performance, lower total runtime and lower latency values are better. This paper refers to total runtime.

Two types of NoSQL database benchmarking tools can be used: database-independent and database-specific. From the database-specific type we can remark `cbc-pillowfight`[20] for Couchbase and `cassandra-stress tool`[21] for Cassandra. Our study aims to compare two different NoSQL database servers by applying the same workload, so database-specific tools cannot be used. From the database-independent type we mention `BigBench`[10] and `YCSB`[4]. While `BigBench` runs only on Linux, `YCSB` runs on Linux and Windows. Our case study involves servers that have Windows operating system installed, which implies that `BigBench` cannot be utilized as benchmarking framework. Also, the resemblance between `BigBench` and TPC-DS[17] makes it less flexible and more oriented on traditional workloads instead of NoSQL workloads. On the other hand, `YCSB` focuses on big data and NoSQL workloads and offers a lot of flexibility in both workload definition and workload configuration. The fact that `YCSB` can be used to test many NoSQL DBMSs, including Cassandra and MongoDB, is another important aspect that makes it suitable for our study.

`Yahoo! Cloud Serving Benchmark`[4] (`YCSB`) appeared as a response to the necessity of a benchmarking tool that is suitable for cloud or NoSQL systems. It is an open source project developed initially at Yahoo! and written in Java which has two base components that are extensible. The first component is the workload generator known as the `YCSB` client. The second component, known as the Core workloads, consists of a set of workload scenarios that have to be executed by the `YCSB` client, as stated in [25]. Each workload used in `YCSB` has two main parts: a data set and a transaction set. The total number of records that need to be loaded into the database before any test is performed represents the data set. The mix of write and read operations to be performed in a test represents the transaction set. The main parameters of the transaction set are: the total number of operations to be applied in a test execution, the number of client threads and the ratio between write and read operations. If the workloads contained in the Core workloads set are not suitable for the needs of the user, new custom workloads can be created. In our benchmarking study, we used `YCSB` version 0.12.0.

`YCSB` is also used in other benchmarking studies that are discussed in the literature: [14], [9] and [8]. These benchmarking studies use a cloud-based infrastructure, while our benchmarking study uses physical machines and it is not cloud-based. The DBMS versions used in our study are newer than those used in [14], [8] and [9]. Also, the operating system installed on our servers is Windows. Other significant differences refer to the data sets used and their size, hardware configuration and workload types. Paper [14] presents a benchmarking study that involves custom workloads and a proprietary data set belonging to a healthcare organization. The benchmarking studies presented in [9] and [8] use data sets generated by the `YCSB` client. In [9], the size of the data sets is not clearly stated, while the study discussed in [8] does not include MongoDB, even if it specifies the actual size of the data sets used.

3. Case study

Our benchmarking experiment involved three servers with the same hardware configuration. A different application ran on each server: `YCSB` client on the first server, Cassandra on the second server and MongoDB on the last server. The server configuration is as follows:

- CPU: Intel(R) Core(TM) i7-4770 CPU @ 3.40GHz, 4 cores, 8 logical processors
- HDD: 500 GB
- RAM: 16 GB
- OS: Windows 7 Professional 64-bit.

The data set utilized in our experiment was generated by the `YCSB` client. It contains 4 million records and it was used with all workloads. Each record contains 10 fields and every field holds a 100 byte string value generated randomly. The data set could fit in the internal memory due to its size. Two predefined `YCSB` workloads were chosen: Workload A (50% reads, 50% updates), which is considered an *update-heavy* workload[22], and Workload B (95% reads, 5% updates), that is considered a *read-mostly* workload[22]. An example of application for Workload A could be a session store that records recent actions, while photo tagging could be a corresponding example for Workload B, as stated in [22]. Both workloads were tested with the following number of operations: 1000, 10000, 100000 and 1000000. For each number of operations, tests were executed using 1, 2, 4, 8, 16, 32, 64, 128, 256 and 512 client threads. Each test with a certain combination of workload, number of operations and number of client threads was repeated three times.
We installed MongoDB with default settings. The default storage engine for MongoDB version 3.4.4 is Wired Tiger. We also installed Apache Cassandra with default settings, but in order to avoid timeouts, we followed the setting recommendation mentioned in [8]:

- `write_request_timeout_in_ms` was set to 100000
- `read_request_timeout_in_ms` was set to 50000
- `range_request_timeout_in_ms` was set to 100000
- `counter_write_request_timeout_in_ms` was set to 100000.

The asynchronous variant of Java driver was used for both database servers. A batch of tests includes all tests having the same workload, number of operations, and database server, but different number of client threads. We

Table 1. Runtime Results for 1000 Operations Workload A

NT	Cassandra	MongoDB
1	6085.666667	1232.666667
2	3702.333333	468
4	3088.666667	249.6666667
8	2959	182
16	2938.333333	166.6666667
32	2880.666667	156
64	2896.333333	157.6666667
128	2855	156
256	2865	174
512	2906.666667	209.3333333

Table 2. Runtime Results for 10000 Operations Workload A

NT	Cassandra	MongoDB
1	34757	7124
2	16884.33333	2792
4	6708	1477
8	4279.666667	811.3333333
16	3535.666667	634.3333333
32	3364.333333	623.6666667
64	3161.333333	478.6666667
128	3187.333333	519.6666667
256	3276	494
512	3229.333333	546.3333333

restarted the database server before each batch of tests was executed, and we captured database server status information before and after each execution of a batch of tests. When the execution of all combinations of tests for the first workload was finished, the data set corresponding to that workload was dropped. After that, the data set characterized by the same parameters that corresponds to the second workload was loaded.

3.1. **Results.** Every test was repeated three times for each combination of workload, database, number of operations and number of client threads. A total runtime (RT) average was calculated for each combination of workload, database, number of operations and number of threads (NT) in order to create the following charts. Figures 1 to 8 show a comparison of RT performance between Cassandra and MongoDB for each combination of workload and number of operations. It is worth to mention here that the cases NT = 1 and NT = 2 are not shown in figures because they produce by far greater RT values than the other cases considered. The runtime results are also presented in tables 1, 2, 3, 4, 5, 6, 7 and 8, including the cases NT = 1 and NT = 2.

Figures 1, 2, 5 and 6 show that in the case of a small number of operations (1000 and 10000, respectively), MongoDB outperforms Cassandra for both workloads used and all NT levels considered.

Figure 3 shows that the performance of Cassandra closes to MongoDB's in the case of an update-heavy workload A and for NT $\geq$ 64, when the number of operations is set to 100000. For the same number of operations, MongoDB produces better results than Cassandra when we use a read-heavy workload B, as shown in Figure 7.

When the number of operations is 1000000, the results differ: Cassandra outperforms MongoDB (as in Figure 4) when workload is update-heavy and NT $\geq$ 32, while MongoDB's performance is better for a read-heavy workload, as shown in Figure 8.

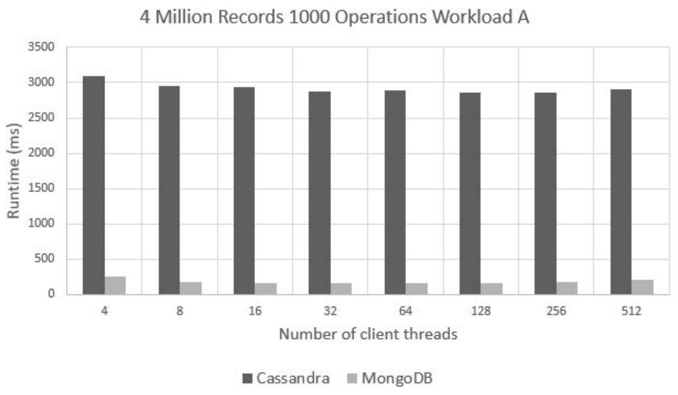


Figure 1. 4 Million Records 1000 Operations Workload A - Runtime

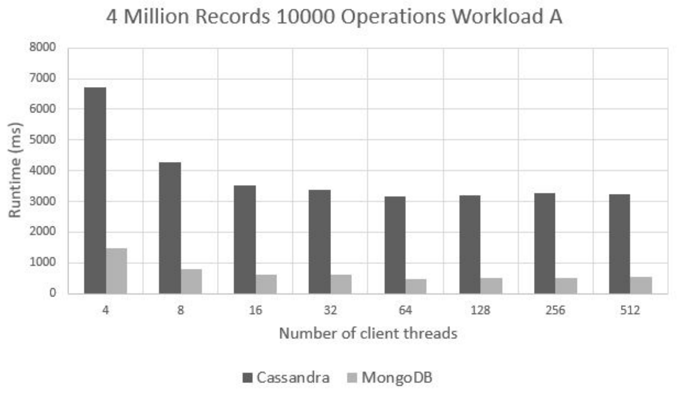


Figure 2. 4 Million Records 10000 Operations Workload A - Runtime

Table 3. Runtime Results for 100000 Operations Workload A

NT	Cassandra	MongoDB
1	303900	62698.66667
2	144909	25791.66667
4	50164.33333	12912
8	16301.66667	6557.333333
16	9344.333333	4950.666667
32	5954	4435.666667
64	5194.666667	4591.666667
128	4950.333333	4415
256	5033.666667	4477
512	5064.666667	4441

Table 4. Runtime Results for 1000000 Operations Workload A

NT	Cassandra	MongoDB
1	2995324.667	629249
2	1454355	259615.6667
4	501797	131332.6667
8	137265.3333	64933
16	67153.33333	48817.66667
32	30202	46567
64	24820	45854
128	23609	46166.33333
256	23431.66667	46108.66667
512	23197.66667	46254.66667

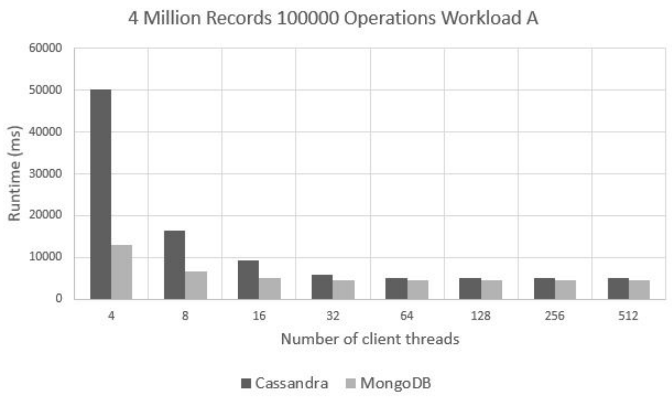


Figure 3. 4 Million Records 100000 Operations Workload A - Runtime

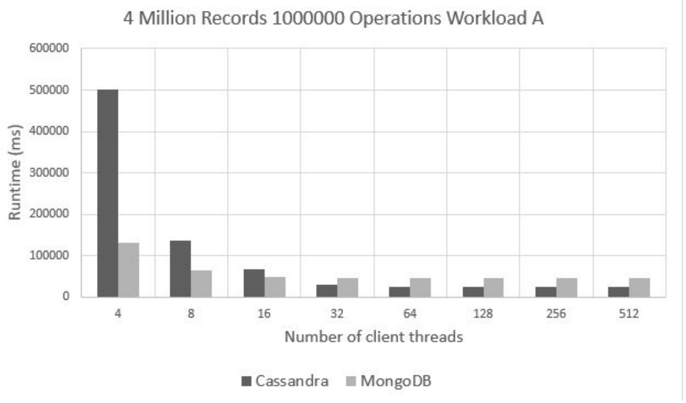


Figure 4. 4 Million Records 1000000 Operations Workload A - Runtime

Table 5. Runtime Results for 1000 Operations Workload B

NT	Cassandra	MongoDB
1	6635	1014
2	3832.333333	415.6666667
4	3104.333333	228.6666667
8	2943	171.3333333
16	2906.666667	156
32	2870.333333	151.3333333
64	2865	140.6666667
128	2875.333333	156
256	2875.666667	171.6666667
512	2922.333333	203

Table 6. Runtime Results for 10000 Operations Workload B

NT	Cassandra	MongoDB
1	35765.66667	6391
2	13884	2870.333333
4	6541.666667	1227.666667
8	4186.333333	546.3333333
16	3567	369.6666667
32	3317.666667	322.6666667
64	3219	317
128	3224	317.3333333
256	3224.333333	317.3333333
512	3244.666667	327.6666667

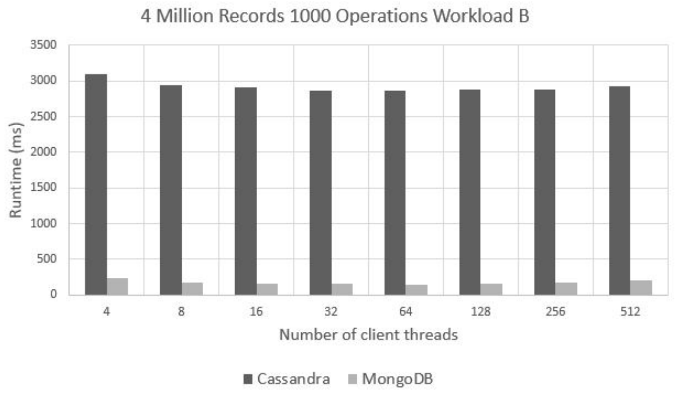


FIGURE 5. 4 Million Records 1000 Operations Workload B - Runtime

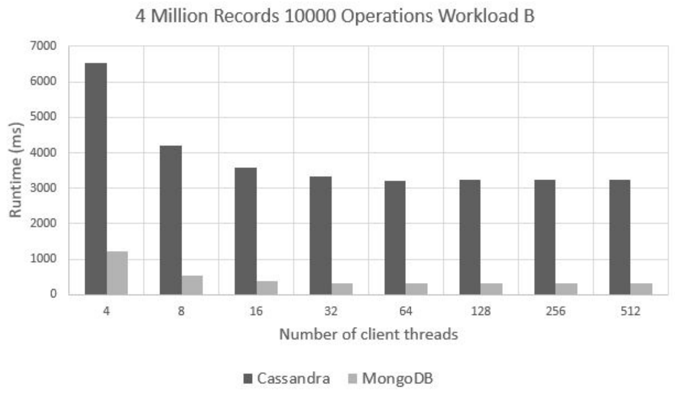


FIGURE 6. 4 Million Records 10000 Operations Workload B - Runtime

TABLE 7. Runtime Results for 100000 Operations Workload B

NT	Cassandra	MongoDB
1	309530.3333	57995.66667
2	134706.3333	26020.66667
4	52171.66667	11117.33333
8	19978.33333	3957.333333
16	9651.333333	2085
32	7051.333333	1632.666667
64	6354.333333	1528.666667
128	6188.333333	1518.333333
256	5964.333333	1523
512	5974.666667	1476.666667

TABLE 8. Runtime Results for 1000000 Operations Workload B

NT	Cassandra	MongoDB
1	3001223	614303
2	1306186	269886
4	500798.3333	105114
8	227412.3333	37201
16	68801	19208.66667
32	43238.66667	14154.33333
64	36936.33333	13645
128	36754	13645
256	32880	13509.66667
512	32579	13338.33333

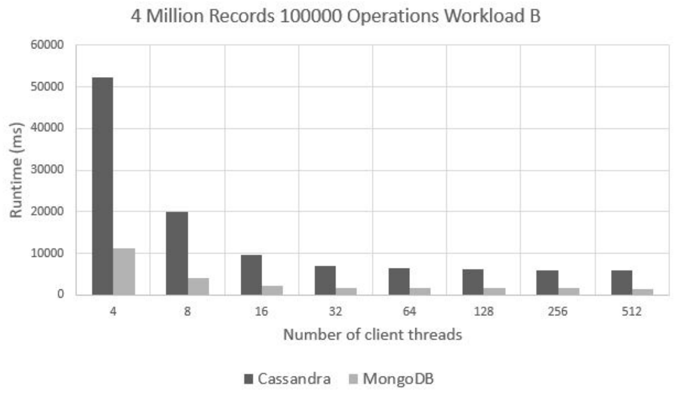


FIGURE 7. 4 Million Records 100000 Operations Workload B - Runtime

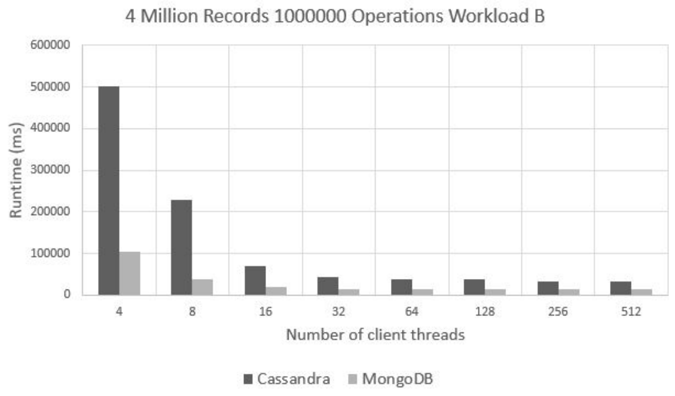


FIGURE 8. 4 Million Records 1000000 Operations Workload B - Runtime

3.2. **Statistical analysis.** Experimental data given in tables 1 thru 8 were processed using two-way ANOVA (Analysis of Variance) procedure from R Statistics Package[18]. Table 9 displays a synthesis of the results. The two factors considered for each experiment are: database (DB, with two levels: Cassandra and MongoDB), and the number of threads (NT, with ten levels: 1, 2, 4, 8, 16, 32, 64, 128, 256, and 512). The interactions between DB and NT were also studied. The column named "Sgf" (abbreviation for statistical significance) refers to the P-value, denoting the level of significance, 0.1%, 1%, 5%, and 10%, following the usual conventions: 0 $***$ 0.001 $**$ 0.01 $*$ 0.05 . 0.1 (*blank*) 1. In other words, if a P-value is $\leq 0.1\%$ (that is, $***$ according to the legend), it means that the differences between means of the factors considered have the

Table 9. Analysis of variance - Runtime Results

Wrk ld	No ops	Database			No of threads			DB:NT		
		F-value	Pr(>F)	Sgf	F-value	Pr(>F)	Sgf	F-value	Pr(>F)	Sgf
A	1000	269.6848	<2e-16	***	3.8486	0.05476	.	1.1328	0.29174	
A	10000	14.3015	0.0003805	***	4.7928	0.0327625	*	2.0936	0.1534923	
A	100000	6.1596	0.01610	*	5.1321	0.02737	*	2.4626	0.12222	
A	1000000	5.5011	0.02257	*	5.2303	0.02600	*	2.5335	0.11708	
B	1000	218.2569	<2e-16	***	3.1952	0.07927	.	1.4045	0.24098	
B	10000	14.1485	0.0004059	***	4.2981	0.0427683	*	1.8757	0.1762899	
B	100000	7.1493	0.009809	**	5.1082	0.027720	*	2.3786	0.128642	
B	1000000	6.4783	0.01370	*	5.3822	0.02401	*	2.4108	0.12614	

strongest statistical significance. The other end of spectrum, when a a P-value is greater than 10% (a blank space) shows that the differences between the means of the factors considered are within the range of experimental error.

When comparing RT averages for DB factor, Table 9 shows that there are several degrees of statistical significance between them in all combinations (workload, number of operations) considered. The same table shows less stronger statistical significance when comparing RT averages for NT factor, and even poorer one in the case of 1000 operations. The interactions DB:NT have no statistical impact on the RT.

4. Conclusion

In our benchmarking study, the performance of the two NoSQL database management systems was measured for two workloads: read-mostly (Workload B) and update-heavy (Workload A). The performance indicator was total runtime (RT). MongoDB outperforms Cassandra in all studies involving Workload B. For Workload A, the situation is the same, with some exceptions: the cases where the number of operations is equal to 1000000 and the number of client threads is greater than or equal to 32.

As further work, we plan to perform other experimental studies using data sets with different number of fields on single server and cluster configurations. Also, we intend to test other workload configurations with data sets that exceed the internal memory. Another direction in our experimental work will deal with database server replication and SSDs as disk storage, in order to measure the performance impact of these configurations. Lastly, the operating system will be another variable in our future case studies.

Acknowledgments

Parts of this work were supported through the MADECIP project *Disaster Management Research Infrastructure Based on HPC*. This project was granted to Babeş-Bolyai University, its funding being provided by the Sectoral Operational Programme *Increase of Economic Competitiveness*, Priority

Axis 2, co-financed by the European Union through the European Regional Development Fund *Investments in Your Future* (POSCEE COD SMIS CSNR 488061862).

References

[1] Apache Cassandra. http://cassandra.apache.org/. Accessed: 2017-09-25.

[2] BaseX. http://basex.org/. Accessed: 2018-11-27.

[3] F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. E. Gruber. Bigtable: A Distributed Storage System for Structured Data. *OSDI '06 Proceedings of the 7th USENIX Symposium on Operating Systems Design and Implementation*, 7, 2006.

[4] B. F. Cooper, A. Silberstein, E. Tam, R. Ramakrishnan, and R. Sears. Benchmarking Cloud Serving Systems with YCSB. *Proceedings of the 1st ACM Symposium on Cloud Computing*, pages 143–154, 2010.

[5] Couchbase. https://www.couchbase.com/. Accessed: 2019-01-22.

[6] CouchDB. http://couchdb.apache.org/. Accessed: 2017-09-25.

[7] G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall, and W. Vogels. Dynamo: Amazon's Highly Available Key-value Store. *Proceedings of 21st ACM SIGOPS Symposium on Operating Systems Principles*, oct 2007.

[8] Fixstars. GridDB and Cassandra Performance and Scalability. A YCSB Performance Comparison on Microsoft Azure. Technical report, Fixstars Solutions, 2016.

[9] A. Gandini, M. Gribaudo, W. J. Knottenbelt, R. Osman, and P. Piazzolla. Performance Evaluation of NoSQL Databases. *EPEW 2014: Computer Performance Engineering, Lecture Notes in Computer Science*, 8721:16–29, 2014.

[10] A. Ghazal, T. Rabl, M. Hu, F. Raab, M. Poess, A. Crolotte, and H.-A. Jacobsen. BigBench: Towards an Industry Standard Benchmark for Big Data Analytics. *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, pages 1197–1208, 2013.

[11] HBase. https://hbase.apache.org/. Accessed: 2017-09-25.

[12] Hypertable. http://www.hypertable.org/. Accessed: 2018-11-27.

[13] JSON. https://www.json.org/. Accessed: 2018-03-16.

[14] J. Klein, I. Gorton, N. Ernst, P. Donohoe, K. Pham, and C. Matser. Performance Evaluation of NoSQL Databases: A Case Study. *Proceedings of the 1st Workshop on Performance Analysis of Big Data Systems*, pages 5–10, 2015.

[15] A. Lakshman and P. Malik. Cassandra: A Decentralized Structured Storage System. *ACM SIGOPS Operating Systems Review*, 44:35–40, 2010.

[16] MongoDB. https://www.mongodb.com/. Accessed: 2017-09-25.

[17] R. O. Nambiar and M. Poess. The Making of TPC-DS. *VLDB '06 Proceedings of the 32nd International Conference on Very Large Data Bases*, pages 1049–1058, 2006.

[18] R Statistics Package. https://www.r-project.org/. Accessed: 2017-09-25.

[19] P. J. Sadalage and M. Fowler. *NoSQL distilled : a brief guide to the emerging world of polyglot persistence*. Addison-Wesley Professional, 2012.

[20] Stress Test for Couchbase Client and Cluster. http://docs.couchbase.com/sdk-api/couchbase-c-client-2.4.8/md_doc_cbc-pillowfight.html. Accessed: 2019-01-03.

[21] The cassandra-stress tool. https://docs.datastax.com/en/cassandra/3.0/cassandra/tools/toolsCStress.html. Accessed: 2019-01-03.

[22] The YCSB Core Workloads. `https://github.com/brianfrankcooper/YCSB/wiki/Core-Workloads`. Accessed: 2017-09-25.
[23] XML. `https://www.w3.org/TR/2008/REC-xml-20081126/`. Accessed: 2018-03-16.
[24] YAML. `http://yaml.org/`. Accessed: 2018-03-16.
[25] YCSB Github Wiki. `https://github.com/brianfrankcooper/YCSB/wiki`. Accessed: 2017-09-25.

Department of Computer Science, Faculty of Mathematics and Computer Science, Babeş-Bolyai University, 1 Kogălniceanu, Cluj-Napoca, 400084, Romania
Email address: {`andorcamelia, bparv`}`@cs.ubbcluj.ro`

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LXIV**, Number 1, 2019
DOI: 10.24193/subbi.2019.1.08

AN ADDRESS PROPAGATION MODEL IN P2P AND F2F NETWORKS

MOHAMMED B. M. KAMEL[1,2], PÉTER LIGETI[1], AND ÁDÁM NAGY[1]

Abstract. Using identifiers to address the member nodes at DHT based peer-to-peer (p2p) networks provides structured method of addressing the nodes. The node lookup is then used to find the equivalent communication address of a given identifier. One of the main concerns is how to find the communication addresses efficiently, especially if a node has joined or rejoined the network recently. In this paper an address propagation model has been proposed which is used as a solution in friend-to-friend (f2f) overlays at p2p networks. The model keeps the required communication addresses up-to-date in order to reduce the need of any node to perform the lookup process. It allows each node to maintain the addresses in a distributed manner using bucket based broadcasting and guarantees that it has the current up-to-date addresses of its friend nodes as a necessary information to establish a direct connection without any centralized scheme. Despite adding some traffic overhead to the network, the proposed address propagation process is secure and fast.

1. Introduction

Because of sophisticated components of mobile equipment, mobile devices have become important tools to sense, communicate, and compute data. In the last few years and due to the decentralized nature of peer-to-peer (p2p) model, this model is widely used as an alternative to client-server model [14]. p2p network is a decentralized network in which each peer acts as both client and server, which makes it more applicable on emerging systems that consist mostly of mobile nodes. p2p network is also widely used in different applications such as secure chatting, distributed cash system [13], distributed data sharing [4] and distributed secret sharing [9][18]. The open nature of p2p

Received by the editors: September 21, 2018.

2010 *Mathematics Subject Classification.* 68M10, 68M14.

1998 *CR Categories and Descriptors.* C.2.1 [**Network Architecture and Design**]: Network communications – *p2p networks*; C.2.4 [**Distributed Systems**]: Distributed applications – *f2f networks*.

Key words and phrases. Private P2P network, F2F network, Network address discovery, symmetric and asymmetric cryptography.

networks and the ability of almost everyone to join the network make some systems such as [8] to use a private overlay at p2p networks called friend-to-friend (f2f) network as their underlying communication scheme to ensure secrecy and anonymity of participants beyond direct peer nodes [6]. Using f2f networks have number of advantages: first, it allows the participants utilizing the established public p2p network to communicate securely and reliably. Second, along with the reliability it provides the anonymity such that each node communicates with its trusted friend nodes without any necessary knowledge about other trusted connections beyond its direct friend nodes.

One important issue in the f2f overlays at p2p networks is the address discovery of nodes. Members of a f2f network could join and leave the network frequently and later rejoins the network with a new communication address (i.e. new logical address or port number). In distributed secret sharing systems such as Siren [8], there should be a direct connection to a number of predefined nodes in f2f network in order to recover the encrypted data stored in p2p network. This means that in addition to retrieving the data from p2p network, a set of direct connections in f2f network have to be established in order to get the required information to decrypting the retrieved data. At the same time, the node has to be able to discover its friends' addresses on the public p2p network without revealing the friendship relationship between them. Thus, keep the up-to-date address of friend nodes is an essential requirement of such networks. In case that a node failed to open a communication channel with a friend node due to a possible update in a node's logical address, a node lookup process will be started. The node lookup process is used to find an equivalent communication address of a given node identifier. Beside reliability, lookup latency is one of the main concerns of p2p systems that uses Distributed Hash Tables (DHT) [1]. In time-critical systems based on f2f overlays, such lookup process increases the required time to retrieving and deciphering the data. In this paper a model for address propagation has been proposed to guarantee that each node at the f2f network always has the fresh addresses of its friend nodes. The transmitted addresses will be kept confidential and known only by the authorized recipients.

The rest of this paper is organized as follows. The next section introduces the various issues and summarizes the efforts in current research field. Section 3 describes the proposed model of address propagation. Section 4 shows the test results and analysis of the model. Finally, Section 5 presents our conclusions.

2. Literature Review

DHT based systems assign a seemingly unique key (ID) to each node that joins the network. These keys are generated using a specific hash algorithm. The input parameters to these hash functions varies and different methods are used such as node IP [16], randomized generated ID [3] or using identity-based cryptography [2]. Each peer in the DHT p2p network is then responsible for storing the information of a number of files depending on the distance between hash value of the file and its identifier. Metrics such as bitwise exclusive or (XOR)[12] is used to determine the closeness. In distributed secret sharing[9], while the data is stored in p2p network, the required keys to decode this data is stored at f2f network. These systems can use DHT to provide a lookup service. Because of one-way property of hash functions, regardless of used method to generate a node ID, the generated identifier does not contain any information about the communication address of the node.

The logical path of peers on underlying network could vary from the id based path on DHT network between them, thus the lookup latency of the p2p networks can be high which in this case leads to operational inefficiency in applications running over it [11]. Reducing the lookup latency is specifically pertinent to decreasing the number of hops the lookup needs to traverse, which adds the scalability constraint for several lookup mechanisms [19]. On the other hand, the frequent joining and leaving of nodes in p2p network which is known as churn [17] will increase the lookup delay by requiring to connect to different nodes due to leaving of previously available nodes. In case of change in the address of one or more friend node, the connection could be lost between them until their new communication addresses will be captured by each other. Some proposed solutions that use DHT such as [10] in order to solve this issue requires a central entity which does not follow the p2p principle and removes the decentralized nature of it by adding a centralized point. The proposed solution to prevent the lookup process for address discovery is to keep each entry at the table of addresses of each node up-to-date. This will includes the direct confirmation of newly updated communication address to f2f members. Keeping the required communication addresses of the nodes up-to-date increases the performance of decoding the retrieved data and mitigate the execution of lookup process. The bucket based broadcast [5][15] has been used at the address propagation model. At the following section the model has been described in detail.

3. Model Description

3.1. **Parameters.** The participants of the model are represented as a finite set of nodes $\mathcal{N} = \{N_1,\ldots,N_j\}$ in the p2p network that update their addresses on different time periods. These periods could be overlapped with each other randomly. This identifier differs from the one used by DHT p2p network and should not be confused with it. Suppose that every node $i \in \mathcal{N}$ can generate a digital signature $Sign_i(m)$ of any message m. Furthermore, an existing f2f network is supposed between some subset of participants. The set of friends of node i is $\mathcal{F}_i \subset \mathcal{N}$. Every node i has a common secret key k_{if} with each of its friends $f \in \mathcal{F}_i$. Let $H(.)$ be a collision resistant one-way hash function and $Enc_k(m)$ be an encryption of the message m using the symmetric key k.

3.2. **Security model.** The address propagation process has to be reliable, secure and should be as fast as possible. The model assumes that the set of friends in f2f network for each peer are *honest* nodes and the majority of peers in p2p network are *semi-honest* whose with some predefined probability may drop some or all of the incoming packets instead of forwarding them. The security requirements that the model has to satisfy are

- Completeness: If a packet generated and sent by an honest node, its friend nodes will verify it and later update their local corresponding communication data of the issuer based on the incoming data.
- Authentic delivery: The address that has been issued by an honest node will be received uncorrupted and the receiving friend node in f2f network is able to identify and authenticate the sender.
- Packet Confidentiality: The transmitted packets that contains address data has to be kept private within members of f2f network. In addition, no intermediate node can get any information from the forwarded packets.
- Anonymity: The real identity of the packet's issuer should be kept secret to the members of the p2p network. The friendship of two nodes should not be revealed by any other friend node.

3.3. **The address propagation protocol.** After an update in a node's communication address (e.g. the node has connected to a different network and a new communication address has been assigned to it), the node will inform its friend nodes directly of the newly updated communication data. Then, node i has to generate an Update Requesting Packet (URP) and inform each uninformed member at the f2f network $\{F_{i_1},...,F_{i_n}\}$ of the newly updated address. The size of the URP is set at the system setup phase and it will remain fixed. The fixed size of URP prevents other nodes from getting extra information via the URP's size (e.g. number of friends). After assigning a new communication address to node i, the URP has to be generated. The method that generates an URP for a node is shown in algorithm 1.

Algorithm 1: Pseudo-code for generation of the URP by node n

```
Input:
$Psize$: the predefined size of the URP
$pID_n$: private ID of node $n$
$\mathcal{F}\ \{\}$ : set of friend nodes
$pID\{\}$: set of private IDs of $\mathcal{F}\{\}$
$CK\{\}$: set of common keys between node $n$ and members of $\mathcal{F}\{\}$
Output: URP
1  $sign(hash(pID_n)) \to URP$
2  $key \leftarrow$ random number generator
3  $encrypt_{key}(data) \to URP$
4  for $each\ f\ in\ \mathcal{F}$ do
5      $send(data) \to f$
6      if $no\ acknowledgementreceived(f)$ then
7          $hash(pID_n \oplus pID_f) \to URP$
8          $encrypt_{CK_f}(key) \to URP$
9      end
10 end
11 if $sizeof(URP) \neq Psize$ then
12     $randomdatasize \leftarrow$ difference$(Psize, sizeof(URP))$
13     $randomdata \leftarrow$ random number generator $(randomdatasize)$
14     $randomdata \to URP$
15 end
16 return $URP$
```

The URP contains the following sections: header, address data, friends' data and padding. These sections are illustrated in figure 1 and a detailed description of them are as the follows:

- Header: At the header section, the issuer node i hashes its private ID and put the signed value of it at the header. Each intermediate node will use the header to determine whether the incoming URP belongs to one of its friend nodes. The hashed value prevents from revealing the private ID of node i to the participants of the p2p network.
- Address data: The data section includes the encrypted data of the node i (i.e. new logical address of node i, its new port and any additional information). The node i encrypts the data using a key k that has been chosen uniformly at random.

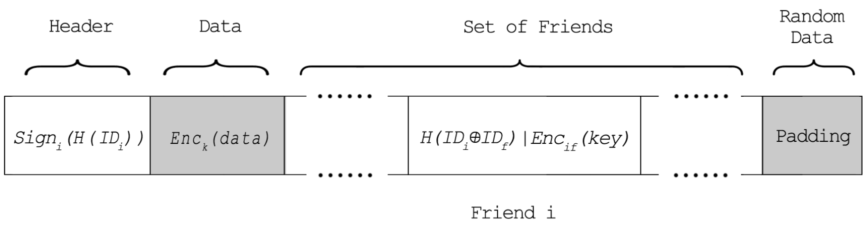


FIGURE 1. Update Requesting Packet

- Friends' data: For each uninformed friend $f \in \mathcal{F}_i$, the node i adds two parts: the common private identifier and the chosen random key. First part that is the common private identifier of nodes i and f will be used by f to indicate which part of the URP belongs to it. This part is computed by first xoring the private ID of i and f, then hashing the resulted value. The second part includes the chosen key k that will be used by f to decrypt the data. This key will be encrypted using the common secret key k_{if}.
- Padding: If the resulted URP's size is less than the predefined packet size, the node i will add some random data at the end of the packet.

After generating the URP by node i, the packet will be broadcast to the set of peer nodes $\{P_1,\ldots,P_m\}$ at the p2p network. Any intermediate node will examine the header to check whether the incoming packet belongs to one of its friend nodes or not. At the final state, all members of the f2f network should receive the transmitted new communication address.

Every node has a set $\mathcal{T}_i$ that includes the hashed value of all friend nodes' private identifiers. This table will improve the checking time of each incoming packet. Any member of the p2p network as soon as receives a packet starts examining the incoming packet to detect whether there is any section of the packet that belongs to it. This operation will be done by first checking the header of the packet. The header will be checked by the public keys of all $i \in \mathcal{F}_i$ in order to find a similar value in $\mathcal{T}_i$. If a value has been found, it means that the sender is $i \in \mathcal{F}_i$. Then, the receiver node has to extracts its part to get the new communication data of the issuer node i. In case that the incoming packet does not belong to any member of $\mathcal{F}_i$, the node will just forward the packet. In case that the issuer node i after a predefined period of time does not receive acknowledgments from all members of $i \in \mathcal{F}_i$, it will regenerate an URP including all uninformed nodes and restart the address propagation process again.

4. Model Analysis

Each node has a unique private ID that is known only by its friend nodes. This ID differs from the node's identifier that is used at the p2p network. The hashed and signed version of this private ID will be part of the header. This field is examined by each receiving node r to check whether the incoming URP belongs to one of its friend node or not. If the incoming URP belongs to one of r's friend nodes, then node r will start checking the first part of each friend section to find its own part. Because node r stores each friend's private ID locally, this field could be computed in advance for each friend at $\mathcal{T}_r$ set. After finding a matching section, the next step is to extract the sender's chosen key k by decrypting it using k_{rf}. The final step is to decrypt the communication address data using the key k. The key k should decrypt the data correctly which indicates that the extracted key has been generated by the pretended sender. The proposed model has to satisfy four security requirements which can be found in table 1.

Table 1. Security Parameters to Meet the Requirements

Security Requirements	Completeness	Authentic Delivery	Packet Confidentiality	Anonymity
header	✓	✓	✓	✓
random key	✓	✓	-	-
encrypted data	✓	✓	✓	✓
padding	-	-	✓	✓

During the test of the proposed model, a p2p network of 300 active nodes has been simulated. All the connections including direct p2p connection and f2f network have been chosen uniformly at random. For the sake of simplicity, it is considered that the offline nodes rejoin the network quickly. During the test of the model existence of partial selfish nodes has been taken into consideration, thus there is a possibility that a node drops part of the incoming packets that do not belong to it instead of forwarding them. Re-transmission rate has been defined as a parameter that indicates the probability of forwarding the incoming URPs at overall nodes in the system. Table 2 shows the details of the parameters that used during the test of the model.

Figure 2 shows the number of URPs that have been transmitted on different test parameters. The overhead increases linearly as the numbers of issued URPs (i.e. nodes with new addresses) increase.

Figure 3 shows the number of issued URPs and the percentage of update rate at the network using different test parameters.

Table 2. Test Parameters of the Model

Parameters	Value
number of active nodes	300
Maximum direct peers in p2p network	9
Maximum nodes in f2f network	5
Re-transmission rate	0.2 to 1
Nodes with issued URP	15 to 150

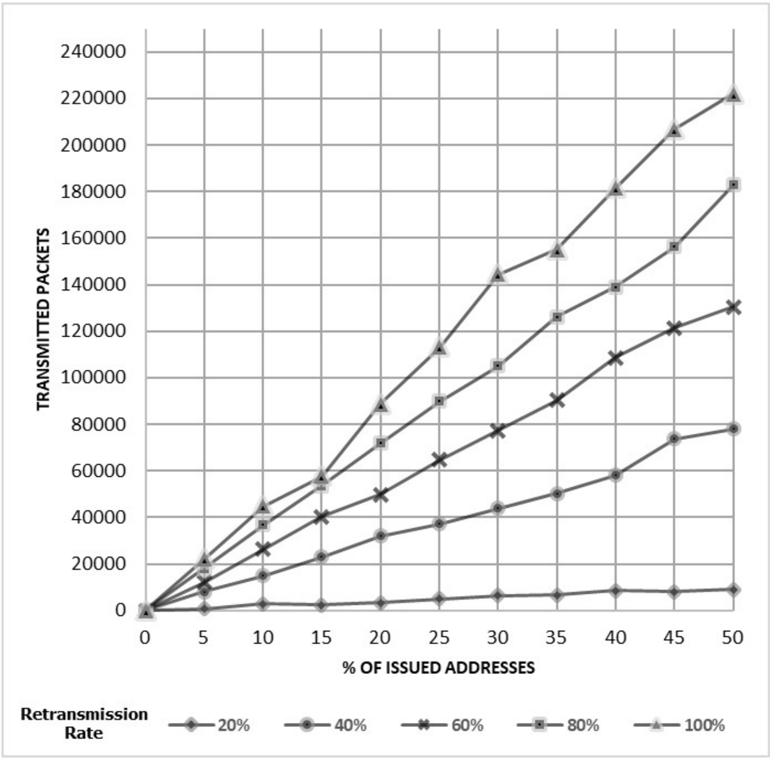


Figure 2. Model Overhead

Analysis of the test results indicates that during the test of the model, number of issued URPs (nodes with new addresses) does not affect the update rate of the system. This means that increasing the number of nodes that generate new URPs will not affect the final number of successfully updated addresses. On the other hand, re-transmission rate of intermediate nodes has been found to affect the update rate of the system. Increasing number

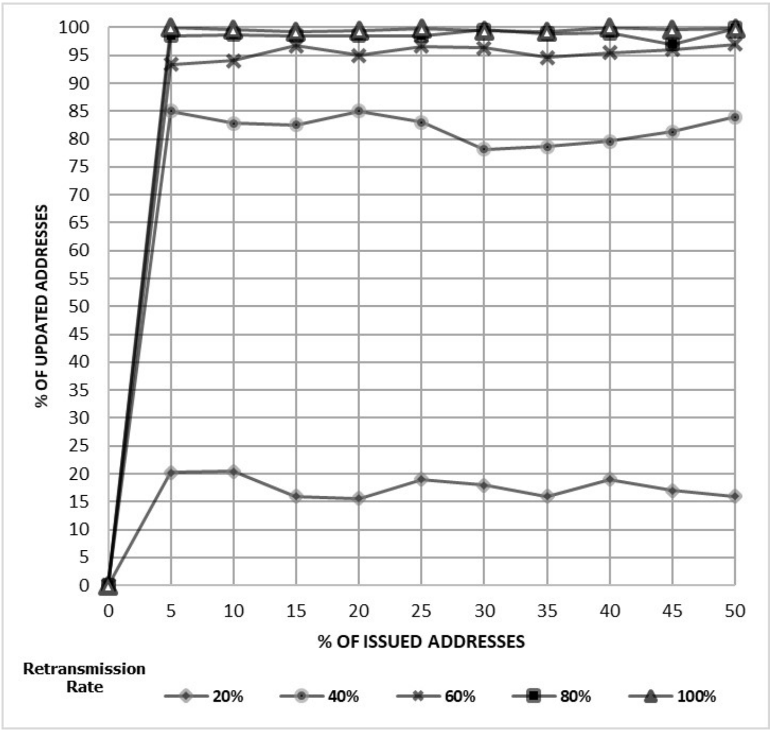


Figure 3. Model Update Ratio

of intermediate nodes that do not participate at the system and drops the incoming packets instead of forwarding them will lead to low update rate.

5. Conclusion

In this paper an address propagation model has been proposed. This model aims to keep the addresses of the nodes at the f2f network up-to-date. The model assumes honest behaviour from the participants of the f2f network, and semi-honest behaviour of the p2p nodes. Analysis of the test results of the model indicates that re-transmission rate of intermediate nodes directly affects the update rate of the system and, therefore there should be some incentives to ensure that the intermediate nodes will forward the incoming packets and prevent selfish behaviour of the peers at p2p network. The current design requires a flood for each node that has got a new logical address. Beside keeping the friendship relationship of the nodes private, it also adds a significant overhead to the network and the communication overhead increases linearly with the number of nodes. Note that, the extension of the proposed method [7] will cover different aspects including improvements to the structure of the model to reduce the overall overhead, taking into consideration different issues including packet transmission termination, offline nodes and additional security parameters to mitigate and prevent malicious behaviour of the participants.

Acknowledgement

This research has been partially supported by the UNKP-17-4 New National Excellence Program of the Ministry of Human Capacities, Stipendium Hungaricum Programme and by the European Union, co-financed by the European Social Fund. (EFOP-3.6.2-16-2017-00013, Thematic Fundamental Research Collaborations Grounding Innovation in Informatics and Infocommunications).

References

[1] Balakrishnan, H., Kaashoek, M.F., Karger, D., Morris, R. and Stoica, I., 2003. Looking up data in p2p systems. Communications of the ACM, **46** (2), pp. 43-48.

[2] Butler, K.R., Ryu, S., Traynor, P. and McDaniel, P.D., 2009. Leveraging identity-based cryptography for node ID assignment in structured p2p systems. IEEE Transactions on Parallel and Distributed Systems, **20**(12), pp.1803-1815.

[3] Cai, X.S. and Devroye, L., 2015. The analysis of kademlia for random IDs. Internet Mathematics, **11**(6), pp.572-587.

[4] Cohen, B., 2008. The BitTorrent protocol specification.

[5] Czirkos, Z. and Hosszú, G., 2013. Solution for the broadcasting in the Kademlia peer-to-peer overlay. Computer Networks, 57(8), pp.1853-1862.

[6] Isdal, T., Piatek, M., Krishnamurthy, A. and Anderson, T., 2010, August. Privacy-preserving p2p data sharing with oneswarm. In ACM SIGCOMM Computer Communication Review, **40** (4) pp. 111-122.

[7] Kamel, M., Ligeti, P. and Nagy, A., 2018. Improved Approach of Address Propagation for F2F Networks. IEEE 2018 2nd European Conference on Electrical Engineering and Computer Science (EECS).

[8] Kasza, P., Ligeti, P. and Nagy, A., 2015. Siren: Secure data sharing over p2p and f2f networks. Studia Scientiarum Mathematicarum Hungarica, **52** (2), pp. 257-264.

[9] Kasza, P., Ligeti, P. and Nagy, A., 2015. On a secure distributed data sharing system and its implementation. In ANNALES MATHEMATICAE ET INFORMATICAE **44**, pp. 111-120.

[10] Kohnen, M., Gerbecks, J., and Rathgeb, E.P., 2011. Applying certificate-based routing to a kademlia-based distributed hash table. Proceedings of the Third international Conference on Advances in p2p Systems IARIA, pp. 85-89.

[11] Lua, E.K., Crowcroft, J., Pias, M., Sharma, R. and Lim, S., 2005. A survey and comparison of peer-to-peer overlay network schemes. IEEE Communications Surveys and Tutorials, **7** (2), pp.72-93.

[12] Maymounkov, P. and Mazieres, D., 2002. Kademlia: A peer-to-peer information system based on the xor metric. In International Workshop on Peer-to-Peer Systems, pp. 53-65.

[13] Nakamoto, S., 2008. Bitcoin: A peer-to-peer electronic cash system.

[14] Parameswaran, M., Susarla, A., and Whinston, A.B., 2001. p2p networking: an information sharing alternative. Computer IEEE, **34** (7), pp.31-38.

[15] Peris, A.D., Hernández, J.M. and Huedo, E., 2016. Evaluation of alternatives for the broadcast operation in Kademlia under churn. Peer-to-Peer Networking and Applications, **9** (2), pp.313-327.

[16] Stoica, I., Morris, R., Liben-Nowell, D., Karger, D.R., Kaashoek, M.F., Dabek, F. and Balakrishnan, H., 2003. Chord: a scalable peer-to-peer lookup protocol for internet applications. IEEE/ACM Transactions on Networking (TON), **11** (1), pp.17-32.

[17] Trifa, Z. and Khemakhem, M., 2016. A novel replication technique to attenuate churn effects. Peer-to-Peer Networking and Applications, **9** (2), pp.344-355.

[18] Yong-Jun, G., Li-Zheng, G., and Ming-Hui, Z. 2014. Improved Multi-secret Sharing Scheme Based on One-Way Function. Indonesian Journal of Electrical Engineering and Computer Science, **12** (6), pp. 4463-4467.

[19] Zghaibeh, M. and Hassan, N.U., 2018. d-SHAM: A Constant Degree-Scalable Homogeneous Addressing Mechanism for Structured p2p Networks. IEEE Access, 6, pp.12483-12492.

[1] Eötvös Loránd University, Budapest, Hungary, Faculty of Informatics, 3in Research Group, Martonvásár, Hungary

Email address: `mkamel@inf.elte.hu, turul@cs.elte.hu, spigy88@inf.elte.hu`

[2] Department of Computer Science, University of Kufa, Iraq