ROCK-CNN: a Distributed RockPro64-based Convolutional Neural Network Cluster for IoT. Verification and Performance Analysis

Rezeda Khaydarova, Vladislav Fishchenko, Dmitriy Mouromtsev, Vladislav Shmatkov
ITMO University
St.Petersburg, Russia
mignolowa@gmail.com

Maksim Lapaev
"TPP Lab" LTD
St.Petersburg, Russia
max.wproject@gmail.com

Abstract—**The paper is dedicated to optimization of machine learning and neural networks applications by replacing common servers with Single Board Computer (SBC) clusters to minimize mounting and service expenses, simplify node mounting process and organize parallel computing in IoT applications. Authors focus on former experience of using distributed computing, mainly, light-weight and cost-optimized SBCs to classify use-cases, then, choose an appropriate hardware platform enabling sufficient data processing and easy hot-replacement of nodes. This task requires organizing an efficient software architecture to make use of advantages of SBCs. A comparison for various SBSs is presented. Authors suggest their formerly-designed architecture with changes allowing using it for neural network applications. Authors pay attention to thorough parameter examination based on numerous tests. Parameter timelines are presented. The paper describes a number of test-cases to validate the efficiency of suggested architecture based on common use-cases. Performance analysis and cluster scalability potential estimation are conducted as well to estimate an efficient number of nodes required for future tasks.**

I. Introduction

Nowadays, IoT-based technologies (Internet of Things) become an indispensable part of our daily life. Our dependence upon the Internet and the devices is increasing at a fast pace. Key communication technologies enabling using IoT are WSN (Wireless Sensor Network), machine-to-machine (m2m) communication, human-machine interaction, web services, information systems, etc [1]. Domain intaken IoT technology implementation has grown dramatically over the last decade. One of the most popular IoT applications is smart houses and home automation. Interconnected devices which may be controlled remotely, smart metering applications to save energy, water, and other resources are the state of art issues.

Constantly emerging modern IoT device management systems support more sophisticated deep-learning technologies making use of neural networks to capture and analyze the environments. Amazon Echo intended to comprehend and implement human voice commands is one of the examples [2]. Deep learning applications for IoT devices often require pseudo-real-time functionality, such as security camera-based recognition tasks, requiring low latency to respond target events: strangers in the house or unattended objects left in subway or airport [3], [4].

Convolutional neural networks (CNN) have been intensively researched and used in large-scale data processing due to their comparable classification accuracy [5], [6]. However, executing CNNs locally on mobile and embedded devices requires large computational resources and has great memory consumption that are not usually possible in IoT platforms. Moreover, by drastically increasing the number of devices connected to the Internet, the network latency increases.

CNN consumes a lot of computational resources and requires powerful computers (supercomputers) with the latest GPU. But supercomputers consume a lot of energy, are expensive and takes a lot of space. Another option is to use common clusters which are still expensive. An alternative to common clusters (a number of interconnected desktop computers) is Single Board Computer (SBC) clusters. Interests in Single board computer clusters are raising since the first Raspberry Pi was in 2012 [7]. SBC clusters domain is still being researched and developed. New more powerful SBCs are emerging every year. One of such devices is RockPro64 [8] released in June 2018. The distributed computing domain requires fare scalability as well as cluster price optimization.

In this paper, we present RockPro64 cluster. We design ROCK-CNN, an architecture for locally distributed convolutional neural network for RockPro64 cluster adaptive for resource-constrained IoT devices. CNN model implementation use cases, such as dealing with images, texts and time series data, are displayed.

There is no particular researches, where RockPro64 clusters' performance was tested. To evaluate the RockPro64 cluster, performance tests were conducted and compared to existing SBCs such as a Raspberry Pi 3B+ [9] and Odroid C2 [10]. Parameters such as a temperature, memory footprint and performance were tested.

We organize this paper as follows: in section II, related works on SBC clusters are presented; problem statement and development pipeline are shown in section III; preliminary architecture of ROCK-CNN is described in section IV; we introduce three use-cases of CNN model in section V (object recognition, sentiment analysis and time-series data); experiments using High performance LinPack tests are provided in section VI, where performance, temperature, memory footprint were tested for RockPro64 cluster. Finally, we summarize the results in conclusion (section VII).

ISSN 2305-7254

II. Related works

Superficial investigations had led to the conclusion that supercomputers are widely used for multi-modal pseudo-real-time analyses, but they are still not a panacea as they are very resource-consuming and expensive, while concurrent and parallel data processing is more efficient than resource-consuming processing [11]. In this section, we focus on overview and analysis of the existing experience of using SBC clusters to organize a cheaper regarding to supercomputers, but still, a powerful way to process such data using machine learning and neural networks. Our research has shown that the optimization of resource costs by using SBC clusters is not a fresh trend, but every task requires using different approaches and cluster organization. In this section, we strive to classify the existing methods and apply one of the classes‘ methods with some domain- and data-specific changes to solve our tasks (Table I).

The above mentioned SBC applications include number of use-cases. First of all, it is possible to use the technology for educational purposes to supply educational and learning process (Table I). Students could learn and understand practically how to work with a parallel distributed system including running various frameworks based on different platforms to measure sets of parameters such as bandwidth, energy efficiency, temperature, latency, and conduct experiments for further investigations.

Another use-case is edge computing (Table I). The technology is crucial for 5G networks enabling a dramatic increase in number of devices connected to network. Moving computational resources closer to end-user (end-device) is possible cost-efficiently only when using Fog computing or Edge computing, which results in data transmission latency improvement [27]. The cost that is necessary for adding extra node is less than to expand nodes in a traditional cluster. Moreover, SBC clusters can be configured to be more robust to hardware failures by offering computational and storage redundancy [7] to produce a more reliable system.

One more use-case accompanied by DNN (deep neural network) or CNN (convolutional neural network) when the resources are limited, enables to partition, distribute and schedule data processing tasks for highly-efficient and less resource-consuming tasks within a cluster of locally connected (within a local network) relatively cheap devices are still a challenge [24]. Existing projects MoDNN [25] and DeepThings [24] deploy DNN/CNN on lightweight end-point platforms such Raspberry Pis and Android smartphones. The DNN/CNN partition decision is based on computational capabilities and RAM of end-point devices. MapReduce as a balancing model is used at runtime in MoDNN project, however it is not an open-source product. DeepThings uses heuristic load-balancing algorithm, however it is not supported any more. Nevertheless, DeepThings model could be enhanced and used as a basic model.

The mentioned above use-cases can not only be used separately, but could form a set of complementary decisions for one solution.

TABLE I. SBC use-cases

Use case	Work/Year	End devices	Task
Education	Glasgow [12] 2013	56 Raspberry Pi	- cloud computing - cloud data center cluster
	Iridis Pi [13] 2014	64 Raspberry Pi Model B	- computing - performance in Hadoop distributed system
	Bolzano [14] 2013	300 Raspberry Pis	- cloud computing research - mobile data center
	Pibrain [15] 2014	8 Raspberry Pi Model B	- developing concurrent programs to run on PiBrain
	GCHQ-Bramble [16] 2015	66 Raspberry Pi Model B	- different experiments with various techniques, approaches, libraries, frameworks
	Wee Archie Blue [17] 2017	16 Raspberry Pi Model 3Bs	- running Linpack tests
	Pfalz-graf [18] 2014	25 Raspberry Pi Model B	- educational activities for High performance computing
Edge computing	Helmer [19] 2016	7 Raspberry Pi 2	- cloudlets
	Claus [20] 2016	300 Raspberry Pi	- docker container architecture for SBC clusters
	Gand [21] 2020	8 Raspberry Pi 2B	- a container management platform deployed on a cluster of SBCs
	Lorenzo [22] 2017	300 Raspberry Pis	- deploying OpenStack Swift platform on an SBC cluster
	Marcos[23] 2018	- not mentioned	- distributed data stream processing
Distributed neural network	Deep-Things [24] 2018	6 Raspberry Pi 3	- distributed execution of CNN-based applications on tightly resource-constrained IoT edge clusters
	MoDNN [25] 2017	4 LG Nexus 5	- distributed mobile computing system for DNN applications
	DDNN [26] 2017	not mentioned	- distributed deep neural network

III. Problem statement and development pipeline

Based on the review we defined and efficient deployment of CNN in a RockPro64 cluster as one of our goals as well as producing a number of metrics to validate the efficiency. Resultant clusters with a properly deployed CNN could be used for edge computing as well in the future.

Most of the software packages for common clusters need to be recompiled and optimized for deploying machine learning libraries and CNN in SBC clusters. Existing works on deployment of CNN in SBS such as MoDNN [25] and DeepThings [24] are still under research and there is still no open-source version and supported solutions.

The preliminary flow of the research steps is displayed in Fig. 1. Efficient partitioning, deployment, balancing and scheduling a CNN inference within locally connected resource-constrained of IoT edge device cluster is still a challenging task.

IV. Architecture of the system

The proposed 4-layer architecture is based on our previous works [28]. The architecture of the system named ”ROCK-

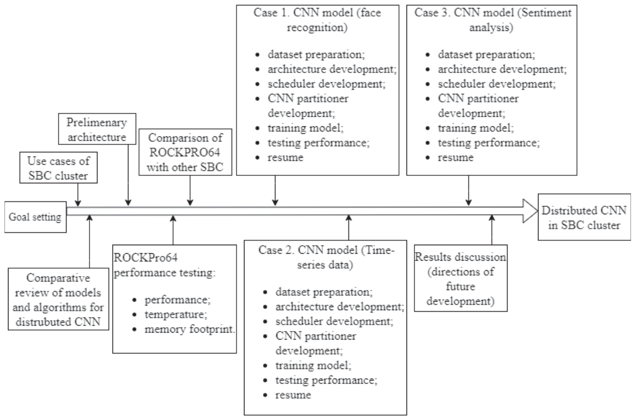

Fig. 1. Development pipeline

CNN" is displayed in Fig. 2. The first layer is a data collecting layer. Various IoT devices such as video cameras, temperature and/or humidity sensors, smart watches, etc. generate a flow of data that needs to be analyzed to be applied for other tasks.

The second layer is a light-weight analysis layer, consisting of micro-services such as image/video, text, time-series data bundles REST APIs. The services filter the data to fit the data type and inform the source devices if data is not recognized. In this work, we are investigating three use-cases to deal with various data formats. The first micro-service is related to tasks for computer vision such as face recognition, object detection, object classification, etc and denies the content if it does not fit. Another case is related to NLP and sentiment analysis in particular. Sentiment analysis is an extremely useful tool to monitor the trends and people's opinions, especially for marketing purposes in a particular domain (restaurants, tours, hotels etc). Analyzing reviews and feedbacks in social media or other review platforms (eg. TripAdvisor) enables optimization of business-processes to fit user or client requirements. The third micro-service is intended to accept and pre-analyze time series such as ECG, measurements from sensors, etc. The purpose of the pre-processing layer is to filter and sort the incoming data, to reject the one not intended for any of the services, and to free up the core of processing irrelevant data.

All the data recognized as relevant and supported is analyzed in SBC cluster. Head node or master node as a core managing component plays the main role in delegating tasks to nodes of the cluster. CNN-based data analyses are the most resource-demanding components [24]. Prior to execution, Core Manager accepts structural parameters of CNN model as an input and provides them to CNN Partitioner module. Partitioner module has its own parameters decomposes any incoming data frames from local data sources into distributable and light-weight inference local tasks according to pre-computed CNN partitioner parameters. Then Scheduler monitors and distributes tasks among the nodes. If the task queue is empty for particular nodes, they start stealing tasks from the nodes that are organized in a peer-to-peer connection. As soon as all the nodes have done processing, results are collected, merged and sent to the head node. At the processing stage CNN frameworks (DeepThings [24], Tensorflow [29], MXNET [30] etc.) could be used depending on the tasks.

The last layer is responsible for result representation. We

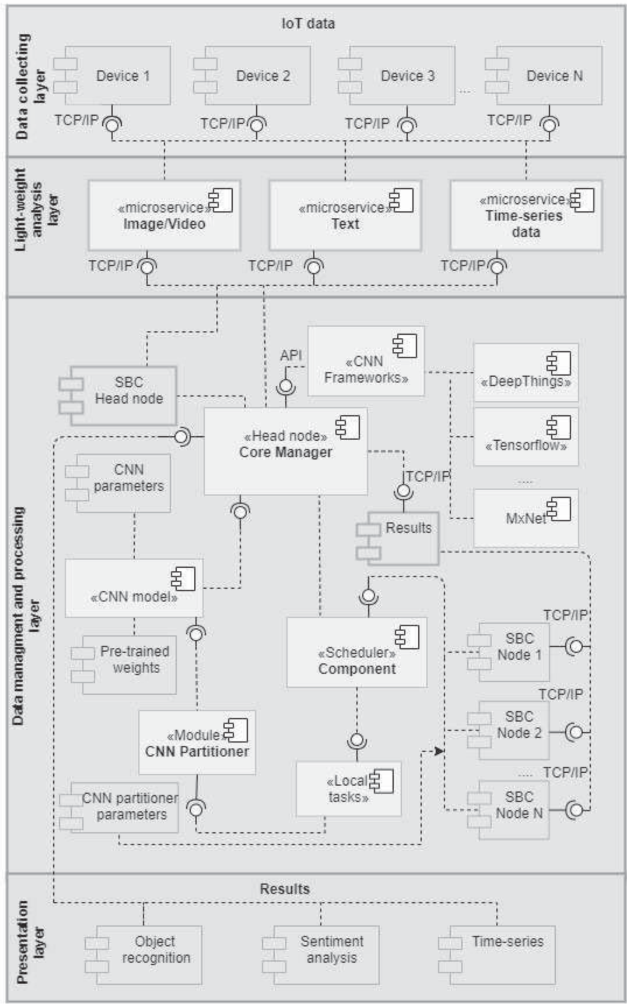

Fig. 2. Architecture of ROCK-CNN

have organized the layer to represent the result of three use-cases: 1) object recognition and face detection to count the objects passed through the gate; 2) sentiment analysis to deal with text for business applications to optimize business-processes in the representation of produced goods; 3) time-series data such as ECG, EEG, and others for medical applications to predict the disease or to estimate the required steps when weather forecasts of internal climate timelines are processed to eliminate the possible effects.

V. Use-cases of CNN model

A. Object recognition

Object detection is one of the use cases of CNN model. Example of CNN model for object detection is presented in Fig. 3.

CNN model for object detection has numerous layers such as convolutional, subsampling (pooling), fully connected, etc.

B. Sentiment Analysis

The general CNN model for sentiment analysis is shown in Fig. 4.

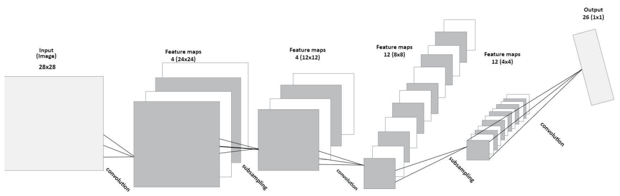

Fig. 3. CNN model of an object recognition, as an example

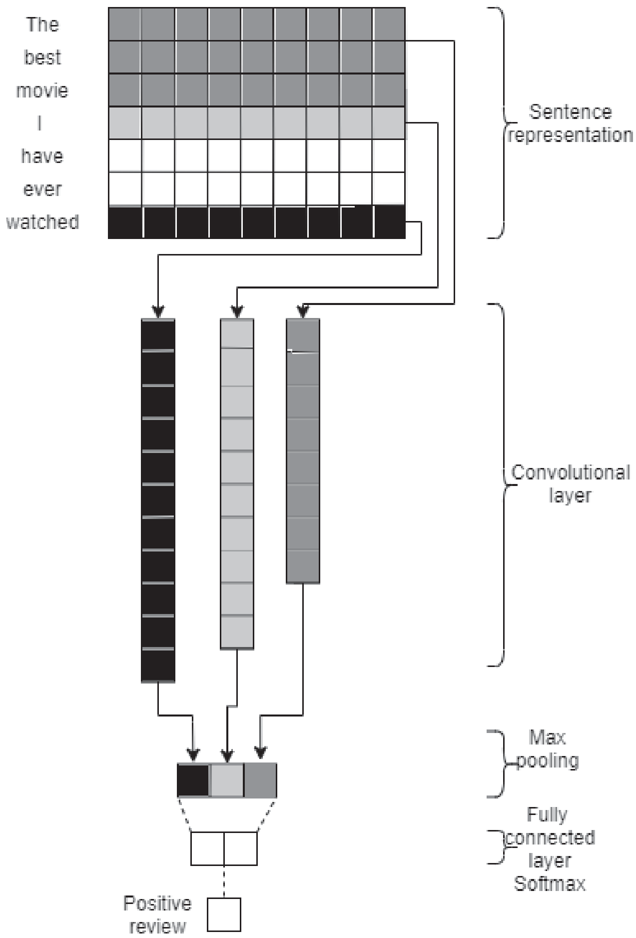

Fig. 4. Sentiment Analysis model architecture

First of all, sentences that need to be analyzed are transferred into a matrix. The rows of each sentence matrix are represented as a word vector. We perform convolution in the matrix applying filters. After that maximum pooling function is applied to each feature map. Finally, softmax function consumes the vector from a fully connected layer and produces a result as positive/negative sentiment output.

C. Time-series data

A large diversity of data is stored in a time-series format: these are climate control measurements, medical test results, stock indices, etc. They can be used in a variety of applications as a forecast and disease prediction source or to identify stock market anomalies. Researches have proved that CNN for time series classification has a number of advantages over other methods: they form highly noise-resistant models able to present informative deep features [31] independent of the current time. In the Fig. 5 we depict the general architecture of the CNN model for time-series data.

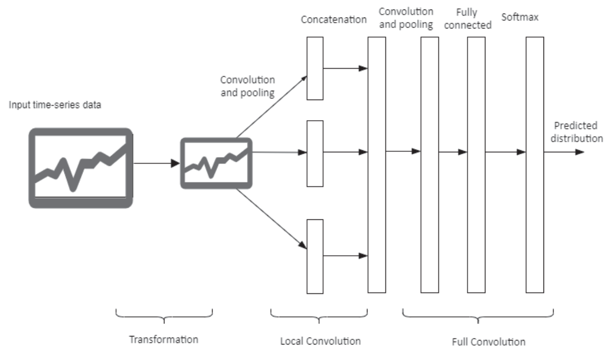

Fig. 5. Time-series data model architecture

On the transformation stage, various functions are applied to input time-series data such as smoothing, discretization etc. In local convolution step, 1-D convolution with different filter resolutions is applied and the maximum pooling is used. Every convolutional layer is followed by a maximum pooling layer. During the full convolution step, all the outputs of previous stages are concatenated and more convolutional and maximum pooling layers are engaged. Then, results are merged and to form a flat vector which is used as an input to a layer connected to softmax function. As a result, a predicted distribution is formed.

VI. Experiments

A. Experimental SBC-based environment

The cluster consists of 24 RockPro64 single board computers (SBCs). In our experiments, we use only 22 nodes. RockPro64 is a 64-bit computer built on a Rockchip RK3399 hexa-core system-on-chip integrating a dual-core ARM Cortex A72 2 GHz and quad-core Cortex A53 processors provided with a quad-core Mali-T860MP4 GPU and up to 4GB of LPDDR4 dual-channel system memory runned by Ubuntu Mate operating system. A 64 Gb Multi-Media card (eMMC) is embedded. The cluster provides a MikroTik CRS326-24G switch and a MikroTik hap-ac2 router. The SBC cluster is displayed in Fig. 6. Six coolers prevent SBCs from overheating.

B. High Performance Linpack

RockPro64 cluster‘s performance was tested using High Performance Linpack (HPL) tests. HPL is a portable and open-source software package that enables to test the dense linear systems to measure the Floating Point Operations per second (FLOPS) on distributed-memory computers [33]. It was chosen as a widely used and freely-available kit supporting most of the relevant systems[34]. For distributing tasks among the nodes a Message Passing Interface (MPI) is used that allows to communicate with processes executing the same task. As an implementation of MPI, MPICH (Message Passing Interface CHameleon) was used. HPL requires applying BLAS (Basic Linear Algebra Subprograms) to perform basic vector and matrix operations [33]. It includes three functional levels:

1) vector operations;

Fig. 6. RockPro64 cluster

2) matrix-vector operations;
3) matrix-matrix operations.

As a library for linear algebra, ATLAS (Automatically Tuned Linear Algebra Software) was used.

C. Performance

The performance of the cluster was tested and HPL.dat file was generated according to the characteristics of RockPro64. Cluster performance is presented in Fig.7.

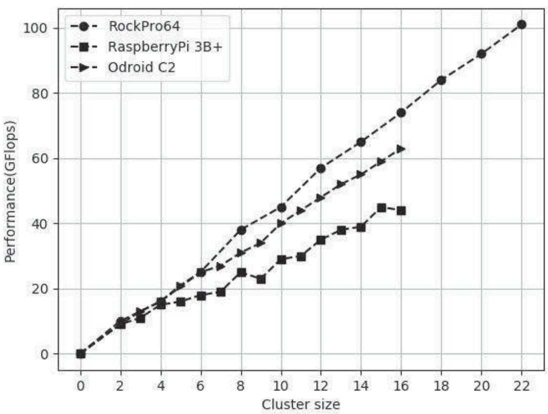

Fig. 7. Comparison of performance

We compared RockPro64 with existing research [7] where performance of Raspberry Pi 3 B+ and Odroid C2 on 16 SBCs are tested. RockPro64 cluster has 22 nodes achieving 100 GFlops. All measurements stop at 80 % memory usage because of some RAM resources are required for the operating system.

Comparison of Raspberry Pi 3B+, Odroid C2, and Rock-Pro64 is presented In Table II. Most of the research works are comparing Raspberry Pi models with Odroid C2 and others [7],[32]. A comparison of RockPro64 with other SBCs is not discussed in researches yet. RockPro64 is one of the most powerful SBC released by Pine64 guaranteeing at least a 5-years support [8]. We have built a RockPro64 cluster displayed in Fig. 6 to test and optimized the use-cases as the most powerful and productive one scarifying price expenses.

TABLE II. COMPARISON OF SBCS

	Raspberry Pi 3B+	Odroid C2	RockPro64
RAM (GB)	1 LPDDR2 SDRAM	2 LPDDR2 SDRAM	4 LPDDR 4
SoC	4 x ARM Cortex-A53	4 x ARM Cortex-A53	4 x ARM Cortex-A53 2x ARM Cortex A72
CPU	Broadcom BCM2837B0	Amlogic S905	Rockchip RK3399
Pocessor cores	4	4	6
Processor speed (GHz)	1,4	1,5	1,8
GPU	Mali-450	VideoCore IV	Mali-T860 MP4
Input power, voltage(V)	5	4.8 - 5.2	12
Price ($)	35	46	79

D. Temperature

During testing the RockPro64 cluster, the temperature was measured with a rate of 10 seconds. The data of sensor temperature is accessible through the sys catalog of the file system. The standard path is /sys/class/thermal/thermal_zone0/temp. In the Fig. 8 we compare two nodes: master node nearest to coolers and the node located on the opposite side far away from coolers. During the first 15 minutes, the coolers were turned off and the tests were running, so the maximum temperature of the master node is about 80°C. But compared to another node

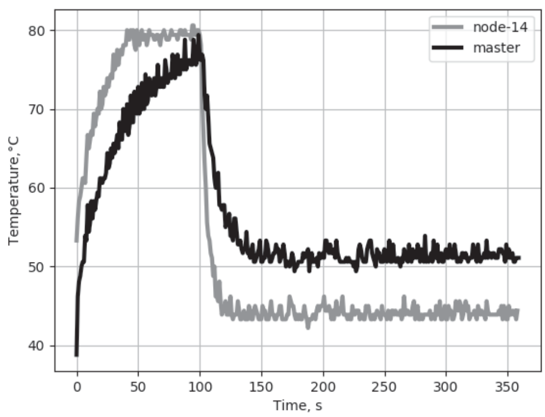

Fig. 8. Temperature timeline for master node and node-18

the temperature in the master node raises drastically because of running additional tasks. Coolers are turned on 15 minutes later the master's node temperature drops dramatically compared to node18 due to its location.

A comparison of two nodes compared to the master node is presented in Fig. 9.

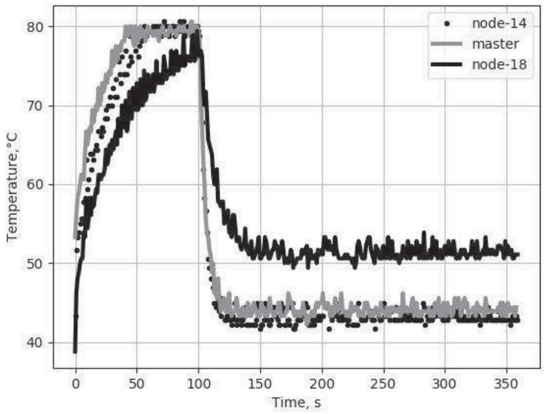

Fig. 9. Comparison of temperature for nodes

Node-14 is closer to coolers as a master node. Within the first 15 minutes, node-18 and node-14 behave differently as far as temperature is considered, node-14's temperature is increasing dramatically. The reason is that the nodes are not balanced and not optimized for parallel task processing, and an appropriate load balancer has to be configured and used to schedule a distribute the tasks equally.

In Fig. 10 is shown the average temperature of CPU in each node for 3 hours. The scheme of location each node

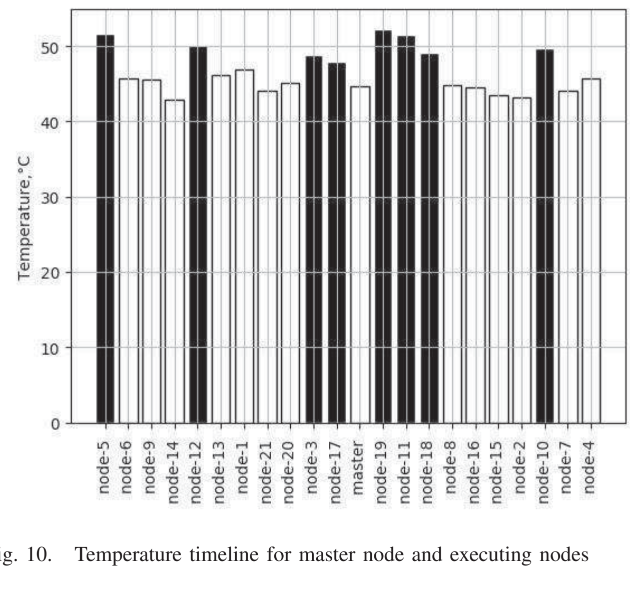

Fig. 10. Temperature timeline for master node and executing nodes

is presented in Fig. 11. The maximum temperature is 52°C, minimum 42°C. Average temperature of nodes, that are near the cooler, is less than others.The risk of destruction is 125°C.

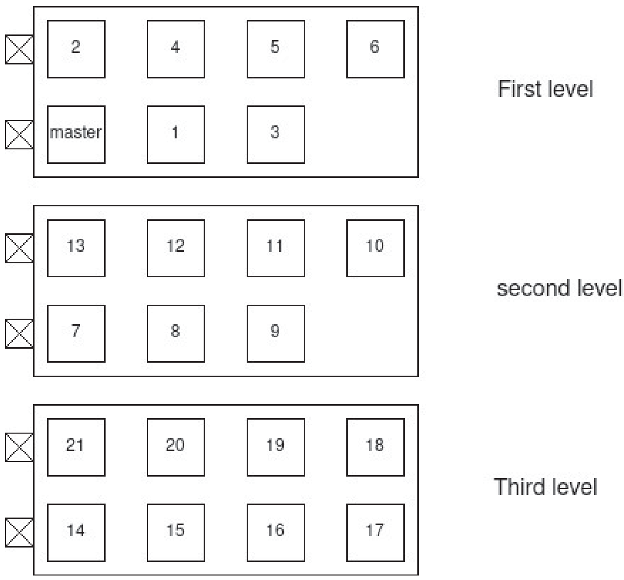

Fig. 11. Distribution of nodes in a rack

E. Memory footprint

The memory footprint of each node is presented in Fig.12. Every node was tested within different memory consumption percentage configurable in the configuration file prior to running the tests. A current maximum of 22 nodes consuming 80% of memory has shown the performance of 100 GFlops. All the nodes have shown the performance of 80 GFlops when loaded with 20% memory usage displaying a fare rate of efficiency.

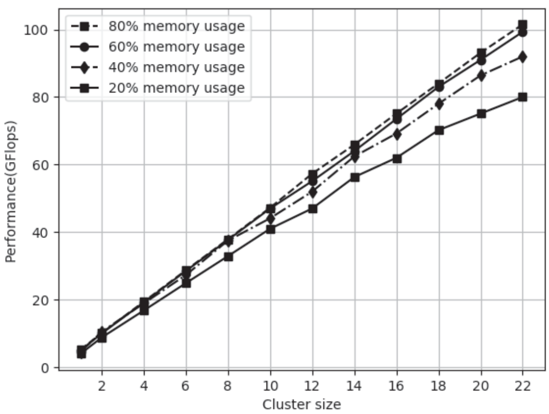

Fig. 12. Memory footprint

VII. Conclusion

In this paper, we presented a RockPro64 cluster that consists of resource-constrained SBCs. We proposed a preliminary architecture of ROCK-CNN for distributed CNN and evaluated it with use-cases for basic tasks using a CNN model. The performance of RockPro64 has been tested by running Linpack HPL tests and has shown 100 GFlops for 22 nodes. Compared to Raspberry Pi 3B+ it is almost twice more powerful and may be used in applications requiring optimal performance-price ratio. An organized RockPro64 cluster has shown the efficiency of the cooling system allowing continuous data processing with no overheating. The maximum temperature of the nodes does not exceed 52°C during the tests while stated junction temperature is 125°C, however experiments has shown that node load balancing is still not perfect and appear to be a challenging task.

To organize a CNN cluster-based platform we used an already developed for our previous research architecture, which still had to be adopted for new tasks.A verified 4-layer architecture was taken along with total core renovation: semantic web services were replaced with neural network- and machine learning-oriented component such as load balancer, partitioner and a set of models introduced to new architecture. Produced architecture references to third-party as DeepThings [24] as well to improve data processing by adopting open-source functionality of the latter.

Future work requires a detailed and thorough analysis of presented use-cases for CNN-models: face recognition, sentiment analysis and time-series data and performance analysis for each case as well as an improvement work on load balancer to make an efficient use of resources. Designing and developing a scheduler to distribute tasks among the ROCK-CNN nodes is the task of high priority, too.

REFERENCES

[1] S. Prasanna and S. Rao, ‘An Overview of Wireless Sensor Networks Applications and Security’, *International Journal of Soft Computing and Engineering (IJSCE)*, ISSN, 2012, pp. 2231–2307.

[2] J. Tang, D. Sun, S. Liu and J.L. Gaudiot, ’Enabling Deep Learning on IoT Devices’, *Computer*, vol. 50, no. 10, 2017, pp. 92–96.

[3] P. Zhang, X. Niu, Y. Dou, and F. Xia, ’Airport detection on optical satellite images using deep convolutional neural networks,’ *IEEE Geosci. Remote Sens. Lett.*, vol. 14, no. 8, pp. 1183–1187.

[4] B. Cai, Z. Jiang, H. Zhang, D. Zhao, and Y. Yao, ’Airport detection using end-to-end convolutional neural network with hard example mining,’ *Remote Sens.*, vol. 9, no. 11, 2017, p. 1198.

[5] X. Zhang, X. Zhou, M. Lin, and J. Sun, ’Shufflenet: An extremely efficient convolutional neural network for mobile devices,’ *in Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 6848–6856.

[6] M. Song, Y. Hu, H. Chen, and T. Li, ’Towards pervasive and user satisfactory CNN across GPU microarchitectures,’ *in 2017 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, 2017, pp. 1–12.

[7] P.J. Basford, S.J. Johnston, S.P. Colin, T. Garnock-Jones, F.P. Tso, D. Pezaros, R.D. Mullins, E. Yoneki, J. Singer, S.J. Cox, ‘Performance Analysis of Single Board Computer Clusters’, *Future Generation Computer Systems*, vol.102, 2020, pp. 278–291.

[8] RockPro64, Web: https://www.pine64.org/rockpro64/, Last accessed: February, 20, 2020.

[9] Raspberry Pi, Web: https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/, Last accessed: February, 25, 2020.

[10] Odroid C2, Web: https://www.hardkernel.com/product-tag/odroid-c2, Last accessed: February, 25, 2020.

[11] S.J. Johnston, P.J. Basford, C.S. Perkins, H. Herry, F.P. Tso, D. Pezaros, R.D Mullins, E.Yoneki, S.J.Cox, J.Singer, ‘Commodity Single Board Computer Clusters and Their Applications’, *Future Generation Computer Systems*, vol.89, 2018, pp. 201–212.

[12] F.P. Tso, D.R. White, S. Jouet, J. Singer and D.P. Pezaros, ’The Glasgow raspberry Pi cloud: A scale model for cloud computing infrastructures’,*Proc. - Int. Conf. Distrib. Comput. Syst.*, 2013, pp. 108–112.

[13] S.J. Cox, J.T. Cox, R.P. Boardman, S.J. Johnston, M. Scott and N.S. O’Brien, ’Iridis-pi: A low-cost, compact demonstration cluster’, *Cluster Comput.*, vol. 17, no. 2, 2014, pp. 349–358.

[14] P. Abrahamsson, S. Helmer, N. Phaphoom, L. Nicolodi, N. Preda, L. Miori, M. Angriman, J. Rikkila, X. Wang, K. Hamily, S. Bugoloni, ’Affordable and energy efficient cloud computing clusters: The Bolzano Raspberry Pi cloud cluster experiment’, *in 2013 IEEE 5th International Conference on Cloud Computing Technology and Science*, vol.2, 2013, pp. 170–175.

[15] P. Turton, T.F. Turton, Pibrain—a cost-effective supercomputer for educational use, *in 5th Brunei International Conference on Engineering and Technology, BICET 2014*, 2014, pp. 1–4.

[16] GCHQ, GCHQ’s Raspberry Pi ’Bramble’ - exploring the future of computing, Web: https://www.gchq.gov.uk/news/gchq-s-raspberry-pi–bramble-exploring-the-future-of-computing, Last accessed: February, 10, 2020.

[17] G. Gibb, Linpack and BLAS on WeeArchie, Web: https://www.epcc.ed.ac.uk/blog/2017/06/07/linpack-and-blas-wee-archie, Last accessed: February, 23, 2020.

[18] A.M. Pfalzgraf, J.A. Driscoll, A low-cost computer cluster for high-performance computing education, *in Electro/Information Technology, 2014 IEEE International Conference*, 2014, pp. 362–366.

[19] S. Helmer, S. Salam, A.M. Sharear, A. Ventura, P. Abrahamsson, T.D. Oyetoyan, C. Pahl, J. Sanin, L. Miori, S. Brocanelli, F. Cardano, D. Gadler, D. Morandini, A. Piccoli, Bringing the cloud to rural and remote areas via cloudlets, *in Proceedings of the 7th Annual Symposium on Computing for Development-ACMDEV’16, ACM Press*, 2016, pp. 1–10.

[20] C. Pahl, S. Helmer, L. Miori, J. Sanin, B. Lee, A Container-Based Edge Cloud PaaS Architecture Based on Raspberry Pi Clusters, *in 2016 IEEE 4th International Conference on Future Internet of Things and Cloud Workshops, FiCloudW, IEEE*, 2016, pp. 117–124.

[21] F. Gand, I. Fronza, N. El Ioini, H. R. Barzegar, and C. Pahl, “Serverless Container Cluster Management for Lightweight Edge Clouds Serverless Container Cluster Management for Lightweight Edge Clouds,” February, 2020.

[22] L. Miori, J. Sanin, and S. Helmer, “A platform for edge computing based on Raspberry Pi clusters,” *in British International Conference on Databases*, 2017, pp. 153–159.

[23] M. D. Assuncao, A. da Silva Veith, and R. Buyya, “Distributed data stream processing and edge computing: A survey on resource elasticity and future directions”, *J. Netw. Comput. Appl.*, vol. 103, 2018, pp. 1–17.

[24] Z. Zhao, K.M. Barijough, A. Gerstlauer, ‘DeepThings: Distributed Adaptive Deep Learning Inference on Resource-Constrained IoT Edge Clusters’, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* , vol.37, 2018, pp. 2348–2359.

[25] J.Mao, C. Xiang, K.W. Nixon, C. Krieger, Y. Chen, ‘Modnn: Local Distributed Mobile Computing System for Deep Neural Network’, *in Design, Automation and Test in Europe Conference and Exhibition (DATE)*, 2017, pp. 1396–1401.

[26] S. Teerapittayanon, B. McDanel and H. Kung, “Distributed deep neural networks over the cloud, the edge and end devices,” *in Proc. IEEE 37th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Jun. 2017, pp. 328–339.

[27] M. Aazam, K. A. Harras and S. Zeadally, ‘Fog Computing for 5G Tactile Industrial Internet of Things: QoE-Aware Resource Allocation Model’, *IEEE Transactions on Industrial Informatics*, vol. 15, 2019, pp. 3085–3092.

[28] A. Konev, R. Khaydarova, M.Lapaev, L. Feng, L. Hu, M. Chen and I.Bondarenko, ‘CHPC: A Complex Semantic-Based Secured Approach to Heritage Preservation and Secure IoT-Based Museum Processes’, *Computer Communications*, vol.148, 2019, pp. 240–249.

[29] E. Fatih and G. Aydın, ‘Data Classification with Deep Learning Using Tensorflow’, *in 2017 International Conference on Computer Science and Engineering* , 2017, pp. 755–758.

[30] A.Jain, A.A. Awan, H.Subramoni, and D. K. Panda, ‘Scaling TensorFlow, PyTorch, and MXNet Using MVAPICH2 for High-Performance Deep Learning on Frontera’, *in 2019 IEEE/ACM Third Workshop on Deep Learning on Supercomputers (DLS)*, 2019, pp. 76–83.

[31] B. Zhao, H. Lu, S. Chen, J. Liu, and D. Wu, ‘Convolutional Neural Networks for Time Series Classification’, *Journal of Systems Engineering and Electronics*, vol. 28, 2017, pp. 162–169.

[32] G. Lencse and S. Repas, 'Benchmarking Further Single Board Computers for Building a Mini Supercomputer for Simulation of Telecommunication Systems', *International Journal of Advances in Telecommunications, Electrotechnics, Signals and Systems*, vol.5, 2016, pp. 29–36.

[33] HPL - A Portable Implementation of the High-Performance Linpack Benchmark for Distributed-Memory Computers, Web: https://www.netlib.org/benchmark/hpl/, Last accessed: February, 22, 2020.

[34] T. Cornebize, F. Heinrich, A. Legrand and J. Vienne, 'Emulating High Performance Linpack on a Commodity Server at the Scale of a Supercomputer', 2017.