Developing the Next Generation Cluster of Computers Remote Laboratory

Christian Madritsch
Engineering & IT
Carinthia University of Applied Sciences
Villach, Austria
c.madritsch@fh-kaernten.at

Thomas Klinger
Engineering & IT
Carinthia University of Applied Sciences
Villach, Austria
t.klinger@fh-kaernten.at

Andreas Pester
Engineering & IT
Carinthia University of Applied Sciences
Villach, Austria
a.pester@fh-kaernten.at

***Abstract*—This paper describes the development of the next generation cluster of computers (CoCo) at CUAS. First, the rationale behind this project is explained. Next, the existing CoCo project is sketched. The section on Machine Learning Frameworks compares different alternatives using well defined criteria and the final section on Hardware Acceleration shows the application of the Intel Movidius Compute Stick to enhance the computational characteristics of the cluster. Finally, conclusions and outlooks to the future of the project are given.**

Keywords— Parallel Computing, Cluster of Computers, Engineering Education, Machine Learning, Remote Labs, IoT

I. Introduction

The Cluster of Computers (CoCo) project was developed over the duration of three years by three generations of bachelor and master students. The aim of this project is to be a foundation for several Computer Engineering and Communication Technologies lectures throughout our curriculum in engineering education. The cluster (see Fig. 1) allows the hands-on study of technologies like parallel computing, multithreading, Open Multiprocessing (OpenMP) and Message Passing Interface (MPI). Furthermore, networking, operating systems and database design are also covered [1].

Fig. 1. Cluster of Computers - CoCo Cluster

The main reasons for the development of the Next Generation Cluster of Computers (NextGen-CoCo) project are twofold. Firstly, we are able to use Machine Learning and Machine Learning Frameworks on the cluster. This enables us to use the system in a broader spectrum of lectures and laboratory exercises then before. Secondly, now we can easily integrate the Cluster into our existing Remote Laboratories infrastructure, which enables us to grant access to the system on a large scale.

The first step of the project was to make an inventory of the current development state of CoCo. This enabled us to apply necessary updates and fixes to the system in time. In parallel, a master thesis focusing on the usability of existing machine learning frameworks (like Cafe, Theano or TensorFlow) was carried out. This led to many practical insights into performance expectations and hardware requirements.

After that, a bachelor thesis with the aim to focus on machine learning using the Intel Movidius Compute Stick led to the final and significant input for this project. Here, a convolutional neural network for object detection was run on a Raspberry Pi 3 using hardware acceleration.

The NextGen-CoCo project is using these outcomes in order to extend the capabilities of the cluster and thereby enabling us to widen the scope of this project. Additionally, the integration into our existing Remote Laboratory infrastructure is being realized.

It is clear for us, that the ability to use our cluster also for machine learning applications will improve the practical and hands-on part of our engineering education. By integrating the cluster into our existing remote lab infrastructure, a large number of students will be able to run experiments on this machine.

Today's Internet of Things (IoT) technologies allow for the implementation of a "supercomputer"-like system with very limited budget. Almost all relevant concepts and topics as well as the practical approach (experiments, hands-on exercises …) of supercomputing can be covered, allowing us to improve and extend our engineering education curriculum.

II. CoCo Overview

The first design of the computing cluster was done during a master thesis project [2]. The Linux-based computing cluster uses five Raspberry Pi 3 modules, one as a master and the other four as computing slaves. Linux is an advantage, since software like Ansible, which automates software provisioning, configuration management, and application deployment, is available. Additionally, a managed Ethernet switch as well as power-supply and cooling concepts were developed.

978-1-5386-9506-7/19/$31.00 ©2019 IEEE

The final large-scale cluster consists of 96 Raspberry Pi 3 nodes, thus providing 384 processor cores. Therefore, the following actions had been taken to ensure a proper and reliable functionality:

- A mounting solution had to be found to group the Raspberry Pi single-board computers (SBC). We decided for a 19" rack system. Each SBC is mounted on a 3D-printed carrier, so that it can be easily removed and changed.
- A power supply concept had to be established. The power lines to the SBCs are switched using relays on a Uniboard Pi HAT. For the power connection to each SBC, it is not advisable to use the built-in micro-USB connector, as they are not very reliable. Therefore, power is connected to each SBC via the GPIO port, thus requiring additional safety circuitry.
- Additionally, an adaptive cooling and ventilation system had to be designed.

Due to the large number of computing slaves, a concept for the distribution, configuration and maintenance of software components needs to be applied. After careful considerations, Ansible has been chosen to accomplish this task. Ansible is a command-line based tool, which uses a scripting language (playbooks) to store configuration information. With little effort, software components can be distributed among all nodes in an automated fashion.

In order to monitor the status of the computing slaves, a web-based interface called Dashboard was developed. The dashboard provides a simple way to turn on and off individual nodes or groups of nodes. Furthermore, it indicates the current operating conditions like temperature and power consumption. Using the dashboard, the user can start a parallel computation and monitor its progress. Finally, the dashboard also displays the current load of the individual nodes in terms of CPU load and memory consumption.

During the next phase of Software implementation, several demo applications have been developed. As a first performance test, a distributed sorting algorithm, based on Bubble Sort and Merge Sort, has been implemented. On the master (a Laptop Computer), the user enters the array size. Next, the array is filled with random numbers.

The array is split into several chunks according to the MPI distribution algorithm. The dataset is sent to the computing nodes where the actual sorting with Bubble Sort takes place. Within a computing node, OpenMP is used to distribute the load among the individual processor cores (four per node). The last step is to merge the individual sorting results together and the sorted data set is sent back to the master. A more visually pleasing demo application is the computation of the Mandelbrot set; accompanied by its graphical representation.

III. Machine Learning Frameworks

Machine learning is a branch of data science and artificial intelligence (AI) where systems are provided with the ability to automatically learn and improve from experience without being explicitly programmed for it. Machine learning focuses on the development and implementation of algorithms that are trained with data and minimize cost functions or problem dimensions.

In the case of NextGen-CoCo, mainly Image Classification problems will be solved.

Machine learning frameworks offer building blocks for designing, training and validation of deep neural networks through a high-level programming interface.

Given the high number of frameworks that are available, selecting one among them can be a tedious task. Each framework is built in a different manner for different purposes. Also, depending on the goals and resources, different frameworks might be relevant. In order to evaluate which of the frameworks is the best alternative they should meet the following requirements: [3]

- Support Python programming language (SP): Python is preferred by data scientist and due to the libraries available it is easier to build highly performing algorithms, it excels in terms of its easy syntactical character as compared to other languages.
- Functionality level of the framework (LF): It affects the overall control over the architectures, low functionality level is preferred as it allows more control. Deep learning frameworks have different levels of functionality. In the case of low-level functionality, the framework allows the researchers to define a neural network of arbitrary complexity from the most basic building blocks; this kind of framework might be called language. On the other hand, frameworks with a higher level of functionality behave like drivers whose objective is to boost the developing productivity, but they are limited due to the higher level of abstraction.
- Neural networks architectures (AN): For image classification, deep convolutional neural networks architectures will be used. In recent years, among all the available architectures, CNNs have become the leading architecture for most image recognition, classification, and detection tasks. The selected framework must support convolutional neural network architecture.
- Availability of pretrained models (PM): Pretrained models can be found in Model Zoos where there are available example scripts for state-of-the-art models and models whose filters already have pretrained weights. Using them will save computation time and help to achieve better results on a new problem by transferring the intelligence acquired on a different data set.
- Community activity (CA): Community activity is an essential part of a deep learning framework; an active community will contribute providing documentation, scripts, models and tutorials, as well as help to solve errors that may appear during the development of a neural network.
- Usable on Rasbperry Pi (PI): Since CoCo consists of Raspberry Pi computers, this is an essential prerequisite. Not all Machine Learning Frameworks can operate without GPU support – therefore an alternative need to be found.

The analyzed frameworks are TensorFlow, Torch/Porch, Microsoft Cognitive Toolkit/CNTK, Caffe, and MXNet. After comparing all results of this selection process, TensorFlow [8] was chosen (see Table 1, details see in [4]).

TABLE 1. MACHINE LEARNING FRAMEWORKS SELECTION

	TensorFlow	Torch	CNTK	Caffe	MXNet
SP	✓	✓	✓	✓	✓
FL	✓	✓	✓	✓	✓
AN	✓	✓	✓	✓	✓
PM	✓	✓	✓	✓	✓
CA	✓	×	×	✓	×
PI	✓	×	×	×	×

TensorFlow 1.9 can now directly be installed on Raspberry Pi 3 running Raspbian 9 (stretch) by applying the following commands on the terminal:

- sudo apt install libatlas-base-dev
- pip3 install tensorflow

IV. HARDWARE ACCELERATION

Since the Raspberry Pi hardware platform does not contain NVIDIA GPU, an alternative solution for the acceleration of machine learning applications needs to be found. One alternative is the Intel Movidius Neural Compute Stick (see Fig. 2).

Fig. 2. Intel Movidius Neural Compute Stick

The Movidius Neural Compute Stick from Intel (NCS) [9] is a small USB-device without a fan. It is used for deep learning in AI programming. NCS is equipped with the low power high performance Intel Movidius Vision Processing Unit (VPU). The VPU can also be found in millions of smart security cameras, gesture-controlled drones, and industrial machine vision equipment. Supported frameworks are TensorFlow and Caffe. The Movidius architecture (see Fig. 3) comprises a complete set of interfaces, as well as a set of enhanced imaging/vision accelerators, a group of 12 specialized vector VLIW processors and an intelligent memory fabric that pulls together the processing resources to enable power efficient data processing.

Using the Movidius NCS is quite simple, thanks to a well written SDK. The SDK has two logical components:

- SDK Tools allow to convert the pre-trained Deep Learning Model in to a "graph" that the NCS can understand. This would be done as part of the development process.
- SDK API allows to work with the graph at run-time - loading the graph onto the NCS and then performing inference on data (i.e. real time image analysis).

The following steps need to be followed in order to setup the NCS on Raspberry Pi:

- Set-up Raspberry Pi in desktop mode,
- Install Debian and Python dependencies,
- Download NCSDK onto Pi,
- Compile and install NCSDK's API framework,
- Test installation using sample code from NC App Zoo.

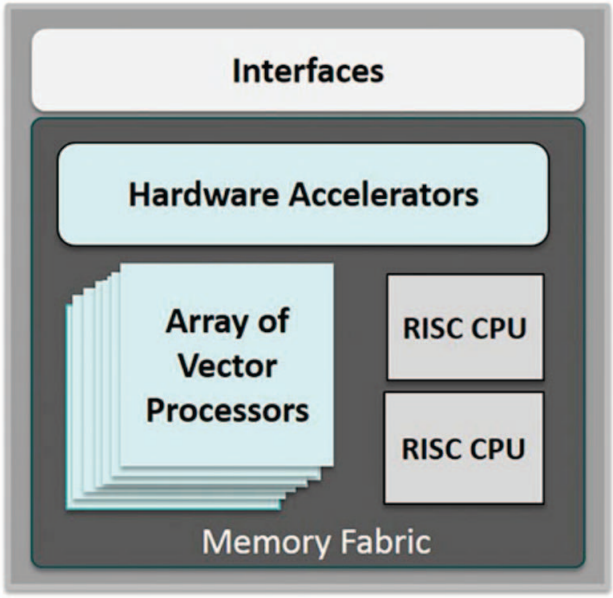

Fig. 3. Vision Processing Unit architecture

The development process for NCS-applications starts by converting the TensorFlow or Caffe model into a NCS-graph. Next, the graph is loaded into the NCS. The last step is to perform the inference by loading an image, running the model, and returning the results (see Fig. 4).

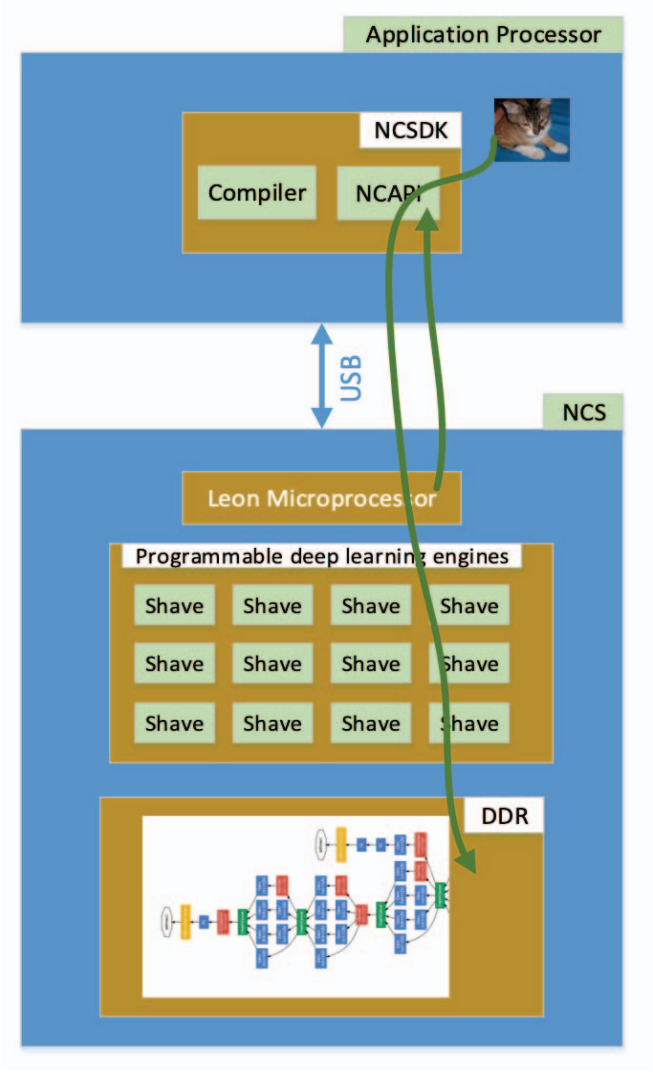

Fig. 4. NCS workflow

V. Remote Laboratory Infrastructure

It is planned to integrate CoCo in a remote access environment. In our case we plan to use our Remote Lab for integration. In general, the integration of simulations or experiments can be done in different ways, "Remote Lab Management Systems (RLMS) line WebLab Deusto" or a direct connection of their client interface with the laboratory machine like in Smart Device specification [5].

In our case we will use a method, developed by Danilo Garbi-Zutin, named Experiment Dispatcher. It is a framework, developed special for cloud-based integration of online experiments, "that provides Online Laboratory server infrastructure as a service (LIaaS). . . to enable and/or facilitate the deployment and development" of such kind of integration architectures (see Fig. 5).

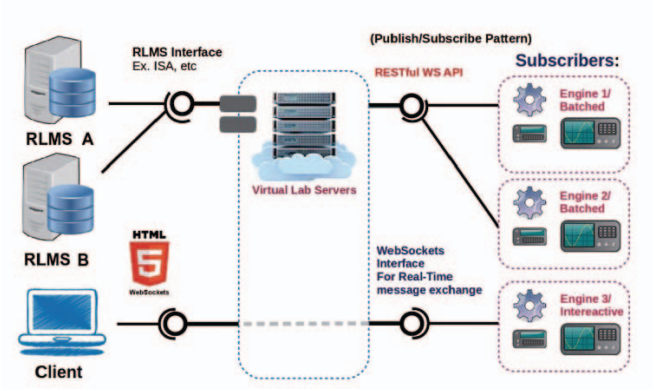

Fig. 5. Lab Infrastructure as a Service

The easiest way will be the integration via a web-site. The CoCo will be combined with an experiment engine, which is listening to requests for computation, process them (run the computation) and return the results. The operation mode can be interactive or batched. Depending from the computation task just the computation result, the image, a video or a screen streaming from the virtual screen will be exchanged with the client.

In the case of the Mandelbrot example a screen streaming will be the most appropriate method. Another method would be the conversion of the screen output to a video, which can be downloaded. The experiment dispatcher will manage different experiment engines and balance the load. In any case all runs, and requests are running in a highly controlled environment, which follows a very specific sequence (see Fig. 6).

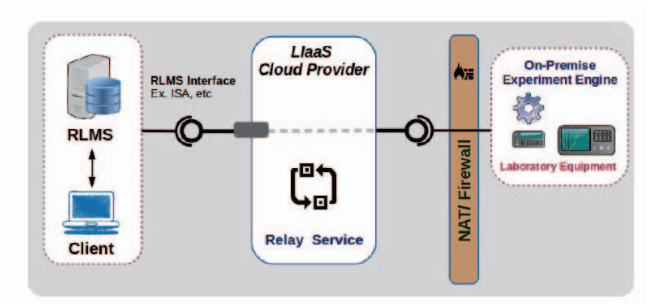

Fig. 6. Lab Infrastructure as a Service

VI. Conclusions ans Outlook

By developing the Next Generation Cluster of Computers, CoCo will be elevated to the next level of complexity and usability. Since Machine Learning is a key and focus area of CUAS, several lectures will be using CoCo for both, a demonstrator and experimentation platform.

Several aspects of the NextGen-CoCo project have not been covered in this paper. One is the extensive networking capabilities of the cluster. Almost 100 devices are interconnected using packet switched Ethernet. The network traffic, originating from the scattering and gathering process of MPI generates a significant load on the network during a very short time period. Some unwanted effects, like jitter or random, unpredictable latencies have not yet been studied in detail. A master thesis which will look into those aspects using a network analyzer and Wireshark [6] is under way.

Another aspect is, that using the hardware accelerator Movidius, only pre-trained models can be used. Since the

cluster is a massive parallel computer, the latest version of TensorFlow will allow us to perform machine (deep) learning on the system. Here, a process needs to be defined to allow a seamless integration into the overall workflow.

Last but not least, the overall system performance is still unknown. Using a benchmark like LINPACK [7] will generate results and it will allow us to plan further improvements.

REFERENCES

[1] C. Madritsch,T. Klinger, "Work in Progress: Computing Cluster using IoT Technologies", IEEE Global Engineering Education Conference EDUCON 2018, Santa Cruz de Tenerife, Canary Islands, Spain, March 17-20, 2018

[2] C. Madritsch, T. Klinger, A. Pester, W. Schwab, "Work In Progress: Using Pocket Labs In Master Degree Programs", International Conference on Interactive Collaborative Learning, Great Britain, 2016

[3] A. Pester, C. Madritsch, "Deep Learning Frameworks for Convolutional Neural", International Conference on Remote Engineering and Virtual Instrumentation, Bengaluru, India, February 2019

[4] X. Lopez de Guereña. "Deep Learning Frameworks for Convolutional Neural Networks". Master Thesis at CUAS, 2018. Unpublished

[5] C. Madritsch, T. Klinger, A. Pester, "Work In Progress: CoCo - Cluster of Computers in Remote Laboratories", International Conference on Interactive Collaborative Learning, Kos, Greece, September, 2018

[6] https://www.wireshark.org/, Dezember 2018

[7] https://software.intel.com/en-us/mkl-linux-developer-guide-intel-optimized-linpack-benchmark-for-linux, Dezember 2018

[8] https://www.tensorflow.org/, Dezember 2018

[9] https://software.intel.com/en-us/movidius-ncs, Dezember 2018