Benchmarking Various ML Solutions in Complex Intent-Based Network Management Systems

Mounir Bensalem, Jasenka Dizdarević, and Admela Jukan
Technische Universität Braunschweig, Germany
{mounir.bensalem, j.dizdarevic, a.jukan}@tu-bs.de

***Abstract*—Intent-based networking (IBN) solutions to managing complex ICT systems have become one of the key enablers of intelligent and autonomous network management. As the number of machine learning (ML) techniques deployed in IBN increases, it becomes increasingly important to understand their expected performance. Whereas IBN concepts are generally specific to the use case envisioned, the underlying platforms are generally heterogenous, comprised of complex processing units, including CPU/GPU, CPU/FPGA and CPU/TPU combinations, which needs to be considered when running the ML techniques chosen. We focus on a case study of IBNs in the so-called ICT supply chain systems, where multiple ICT artifacts are integrated in one system based on heterogeneous hardware platforms. Here, we are interested in the problem of benchmarking the computational performance of ML technique defined by the intents. Our benchmarking method is based on collaborative filtering techniques, relying on ML-based methods like Singular Value Decomposition and Stochastic Gradient Descent, assuming initial lack of explicit knowledge about the expected number of operations, framework, or the device processing characteristics. We show that it is possible to engineer a practical IBN system with various ML techniques with an accurate estimated performance based on data from a few benchmarks only.**

I. Introduction

Managing, orchestrating and controlling environments which include distributed devices spanning highly heterogeneous networks is becoming an increasingly complex task. For this reason, the interest in more intelligent and automation geared management methods, such as utilization of intent based network (IBN) management, has also been on the rise, and is significantly improving both the automation and management process in wide spectrum of application domains and complex ICT systems. IBN systems are designed to translate high-level intents into a structured format with the help of natural language processing techniques, thus easing the system management also for non-experts. Structured specifications translated from the intent configure the systems in a policy-based fashion, without the need to manually check the conflicts in a large-scale system. IBN typically deploy various ML techniques not only for natural language processing, but also for the entire suite of network management functions, such as resource management or security.

Whereas IBN concepts are generally specific to the use case envisioned, the underlying ICT platforms are generally heterogenous, comprised of complex processing units, including CPU/GPU, CPU/FPGA and CPU/TPU combinations. Hence, for the system to choose between multiple potentially new ML model configurations is not only potentially time-critical, but can also result in computationally ineffective choice of ML techniques on a specific hardware. Moreover, intents are useful precisely because they enable system configurations where a problem can be solved by choosing from a set of candidate ML-based solutions, whereby each of the candidates can give the comparable output, but with different accuracy and run-time. Hence, an efficient IBN system needs to be able to predict how long it takes for a potential ML solution to execute for a given intent, while preserving a threshold of accuracy. This brings into question of what the optimal way of estimating performance of ML solutions in an IBN system is, and whether it is possible to learn and extract features that relate the performance of a specific intent to the hardware in an implicit way. What is needed is an intent-based networking paradigm that would allow administrators to choose and deploy best ML models on the most suited hardware platform, through simple use of high level intents.

The objective of this paper is three-fold. Firstly, we introduce a specific case study of complex IBN management systems implemented in ICT supply chain systems. The implementation of intents in a specific case study enables us to define meaningful intents, which in turn allows for an easier system administration. Second, we focus on the design of the ML benchmarking solution for intents, using a specific module within IBN-based management system, we refer to as *ML Recommender*. This module estimates the computational performance of ML-based techniques on heterogeneous hardware devices in the system, without explicit knowledge about the number of operations, framework, or the device processing characteristics of each ML technique. To discover and learn the features that link the performance of a specific ML model to the hardware in an implicit way we will use Collaborative Filtering (CF), as a commonly used tool in recommendation systems [1]. Finally, we evaluate the system performance by measuring the quality of forecasting using a small number of tests on some devices, and the information from previous execution of other ML techniques on existing hardwares. The information collected about the performance is represented as a sparse matrix. The matrix would contain information about the computational performance of certain ML tasks, such as Deep Learning models, which were hosted on specific hardware devices in previous executions. After that, ML algorithm called Singular Value Decomposition (SVD), will be used in order to estimate the missing values of the input matrix, which will in turn be used as valid ML-performance estimation. We show how the estimated outcome effectively enhances IBN-based management systems capabilities of scheduling, planning or hardware provisioning.

The rest of this paper is organized as follows. Section II describes the related work. Section III describes the case study of intent-based management in ICT supply chains. The ML benchmarking solution is presented in IV. Section V evaluates the performance. We conclude the paper in Section VI.

II. Related work

With a rise of complex network environments and system architectures, there has also been an increased interest in more intelligent and autonomous management methods, with intent based networking as one of the leading trends. Intent based networking is a practical concept used in specific reference scenarios. In [2] and [3], multi-platform 5G and IoT network infrastructures are studies with IBN solutions. In [4] distributed, multi-technology, and multi-stakeholder network infrastructures are taken as examples to drive the need for intelligent automation and orchestration. Our previous work in [5] introduced and motivated intent-based networking in the so-called ICT supply chain system, defined as *a system integrating ICT products and services, transforming raw materials, and components into a finished product or service from supplier to the end user.*. Similarly, a part of closed-loop automation was analyzed in supply chains, with different management system requirements formulated with intents [6].

Every IBN architecture deploys ML techniques, which in general need to run on rich heterogeneous hardware platform environments. Hence, for an intent to be executed effectively, it is critical to understand the performance of the ML solution deployed. In [7], an ML approach based on artificial neural network was proposed, with the aim of predicting the performance of different applications and with that advancing the effectiveness of scheduling on a CPU hardware. In [8], the execution time of an application on a specific FPGA is used to predict the execution time on other hardware without making a real implementation, using neural networks as a tool for estimation. In addition, software developers need to define some system characteristics of their developed components before deploying in production. In [9], a methodology is presented that estimates the execution time of software components on a specific architecture using simulation and analytical tools that uses parts of system information. The paper compared the estimated execution times of certain software components using the simulation based system and the real benchmarks. The work presented in [1] proposes an online and scalable datacenter scheduler, considering the heterogeneity of hardware and the interference between executed workloads, using the collaborative filtering prediction technique. The same technique will be used in this paper, due to its wide use in recommendation systems. One of the most famous applications of CF is Netflix Challenge [10], which will also be used as a reference method in our work, as well as the framework developed in [1].

III. A Case Study of Intent-Based Management in ICT Supply Chains

As ICT supply chain networks are comprised of heterogeneous and complex infrastructures, with the high interoperability requirements. Here, managing, orchestrating and enforcing policies at scale is a challenge. To tackle this challenge and develop an intelligent management strategy, we employ the paradigm of intent-based networking in order to allow administrators to set high level intents that instruct the system of *what to do* instead of *how* a task should be executed.

A. IBN Reference Architecture

The case study of IBN based management system design is illustrated in Fig. 1, with the ML-based intelligence as the key enabler of the proposed solution, considering the need for automation of system configurations. The ICT supply chain network typically consists of several domains, such as factories, transportation facilities, warehouses, and retailers. Each domain can have its own computing servers, which are used for various computational functionalities, such as networking, security, system optimization, and system management. Tasks can be forwarded to dedicated servers depending on the required computation, memory, throughput etc.

The reference architectural design of the intelligent intent based management system is illustrated in Fig. 1. The system consists of ICT supply chain network infrastructure, including various IoT devices such as security cameras, RFIDs, and sensors, edge computing nodes located geographically close to the IoT devices, a centralized ICT supply chain network controller to monitor and manage the infrastructure, and a high-level intent-based orchestrator. The network controller collects telemetry from the infrastructure, manage and enforce policies in the system. The intent-based orchestrator is a high-level layer that interacts with the network administrator in order to provide an easy-to-use management interface. It includes different modules that insures the translation of user, i.e., system administrator, intents into configured policies to be applied by the network controller: a dashboard, intent-based interface, intent manager, policy configurator, knowledge base, monitoring, and ML recommender. The *dashboard* provides a user interface, where the intents given by the network administrator are read and then parsed by an *intent-based interface*. An *intent manager* component is designed to translate high-level intents into a structured format with the help of natural language processing techniques. Structured specifications translated from the intent are then forwarded to the *policy configurator* component, which matches the user requirements with possible predefined policies stored in a *knowledge base*, and more specifically in a *policy store*. After configuring these policies, conflicts are checked and verified. Then the *policy configurator* triggers the network controller to apply changes in the network infrastructure. Monitoring tools

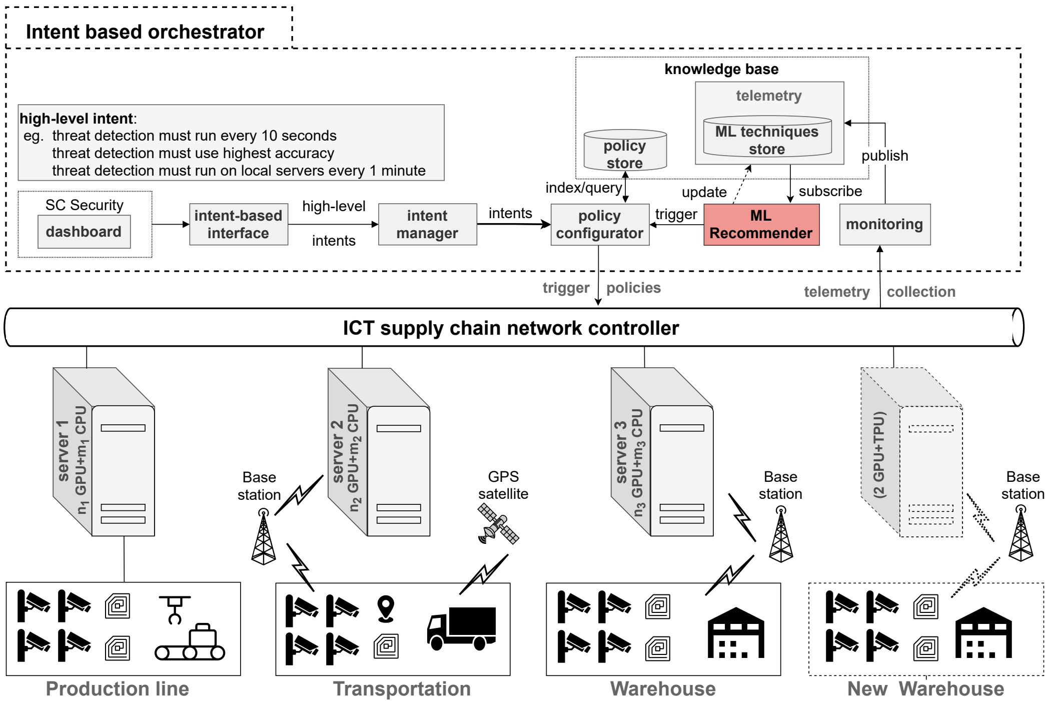

Fig. 1: Reference architecture of an intelligent intent based management system in ICT supply chain

are used by the network controller to collect metrics and provide the real time state of the network. The *Monitoring* component in the intent-based orchestrator filters the telemetry collected from the network and uses the *knowledge base* to store some selected telemetry such as CPU/GPU performance, memory, latency, throughput, security state of links and devices, application logs, and security camera results etc. One important telemetry information is the the ML techniques performance on different hardware, which is saved in an *ML techniques store*. In order to use this telemetry and improve policy configuration methods such as scheduling of functions on servers, routing, and enhancing access control configuration, we design an intelligent module called *ML Recommender*. This module will tackle one of the most important challenges related to choosing and deploying the best ML-based algorithms in the most suitable server of the supply chain infrastructure, using the performance information collected in the *ML techniques store*.

B. Benchmarking ML Solutions in IBN Systems

ML solutions are widely used for various tasks in ICT supply chains, such as threat detection, demand forecasting, and face detection. When a new device joins the network, it has no performance history, which makes the system enable to accurately take several decisions such as ML techniques allocation. To solve this problem, the *ML Recommender* is introduced, which is responsible on benchmarking the performance of ML techniques on new coming device or the performance of new ML techniques on existing devices. (The next section describes the design of the ML recommender in detail.) From a system design side, in order to trigger the *ML Recommender*, we propose two intent types that can be used by the system administrator, as follows:

- For adding a new hardware device: ”add device *device_id* to domain *domain_id*”
- For adding a new ML technique: ”add ML-technique *ML-tech_id* to ML-technique type *ML-tech_type_name*”

The intent manager then discovers the intent types from the input text and parse it in a structured JSON format as: {intent_name : ”adding device”, device: ”device_id”, domain: ”domain_id”}, and {intent_name : ”adding ML-technique”, ML-technique: ”ML-tech_id”, ML-technique_type: ”ML-tech_type_name”}. After that the policy configurator matches the given intent with the required policies. For instance, this can be defined as follow: check the connectivity of device (”device_id”), or check the existence of ML-technique (”ML-tech_id”) in an ML techniques store using its ”ML-tech_type_name”, alert the user if the checking operation fails, trigger the *ML Recommender* with the provided information, or similar.

IV. ML Benchmarking

As a part of the intelligent intent based management system, *ML Recommender* module will have the task of benchmarking and estimating the performance of different ML techniques on heterogeneous ICT supply chain hardware platforms. Instead of running all existing ML techniques on each device, to link the performances of a specific ML model to the hardware we will use Collaborative Filtering. This section first explains the

CF techniques envisioned, after which we present how they are being applied to estimate technique-device benchmarks.

A. Collaborative Filtering Techniques

We adopt a single rating CF system for product recommendation, where the input is modeled as a sparse matrix $E \in \mathbb{R}^{m,n}$, where m is the number of users and n is the number of items. Each user i is represented by one row and each item j by one column. In this example, we denote by $e_{i,j}$ the rating of item j by user i. A new incoming user will rate a limited number of items, then the existing user rating is used to estimate the ratings for other missing items. CF algorithms ensure that the predicted value is always in the same range used by all users. After obtaining the complete rating information, a recommendation system recommends the list of Top-N most likable items by the user.

One of the efficient methods of CF, which among many other recommender systems was also used in the Netflix Challenge [10], is the Singular Value Decomposition SVD, that we will adopt as a first step of our estimation approach. SVD is a technique of dimensionality reduction used to decompose a matrix into three matrices U, V, and Σ, in order to find lower bi-dimensional feature space.

$$SVD(E) = \begin{pmatrix} e_{1,1} & \cdot & \cdot & \cdot & e_{1,n} \\ e_{2,1} & \cdot & \cdot & \cdot & e_{2,n} \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & e_{i,j} & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ e_{m,1} & \cdot & \cdot & \cdot & e_{m,n} \end{pmatrix} = U\Sigma V^T \quad (1)$$

Where U and V are $m \times r$and $n \times r$ orthogonal matrices, representing the left and right singular vectors respectively, and Σ is an $r \times r$ diagonal matrix, representing the singular values, r denotes the rank of matrix E, and expresses the number of features of similarity resulted by SVD. The application of SVD requires the factorization of our input matrix E, which is a complex problem due the sparsity of the matrix. Traditional SVD algorithms cannot work with incomplete information about the entries and in addition they cause the issue of overfitting, meaning that the estimation will converge to some known values from the given input.
In order to recover missing entries of the matrix E, PQ-reconstruction method [1] will be used. To that end we define the matrices $Q = U$, $P^T = \Sigma V^T$, and $R = Q \times P^T$ as the approximation of A, including the missing entries. The vector associated to the item j is denoted by $q_j \in \mathbb{R}^r$, which expresses the features representation of item j. Similarly for users, $p_i \in \mathbb{R}^r$ denotes the vector associated to the user i, to express its features representation. Therefore, the elements of the approximation matrix R can be represented as:

$$r_{i,j} = q_j^T p_i \quad (2)$$

The estimation of any rating of a user i to an item j can be easily obtained, once we have the feature vectors p_i and q_j.

In matrix reconstruction, the estimation model is built using the observed ratings and avoiding the missing entries, represented by zero values in the sparse matrix E. For the overfitting problem, a regularization term is used while minimizing the error during the learning phase. The following minimization formulation of the problem tries to minimize the squared error between the rating and the estimation, considering the regularization term:

$$\min_{q,p} \sum_{(i,j)\in S} (r_{i,j} - q_j^T p_i)^2 + \lambda(||q_j||^2 + (||p_i||^2) \quad (3)$$

Where S represents the set of non-zero values of matrix E, and λ is a regularization parameter. The non-zero values are considered to model the previously observed ratings in order to create a general fitting function able to predict unknown ratings, while minimizing the error. The regularization parameter λ is used to adjust the minimization function and prevent overfitting the model with known values.

Next, we adopt the Stochastic Gradient Descent as a learning algorithm that processes the matrix E to improve the estimation. The feature vectors p_i and q_j are initialized randomly and iterated over all the training set of ratings. In each iteration the value of the rating using eq. (2) is predicted and computed as follows:

$$\epsilon_{i,j} = r_{i,j} - q_j^T p_i \quad (4)$$

Afterwards, we update the feature vectors q_j and p_i using a learning rate σ, considering the regularization factor λ, as follows:

$$q_j = q_j + \sigma \cdot (\epsilon_{i,j} \cdot p_i - \lambda \cdot q_j) \quad (5)$$

$$p_i = p_i + \sigma \cdot (\epsilon_{i,j} \cdot q_j - \lambda \cdot p_i) \quad (6)$$

The algorithm iterates over all matrix entries s times, where s is a parameter to be defined, as well as the learning rate σ, regularization factor λ, and the number of features r (called latent factors). The utilization of the previously described algorithm to estimate the performance of a ML-based technique is described in the next subsection.

B. Predicting Performance of Various ML Techniques

In proposed intelligent intent based management system, we assume the existence of M ML techniques, that can be executed on N hardware devices. Example of ML-based techniques can be various object detection algorithms used to monitor security cameras, executing different tasks like employee detection, threat detection, product quality monitoring, etc. Different ML models can do the same task, while having different speed performance in inference, memory consumption, and accuracy of results. One of the most important factors that need to be measured is the inference performance, which will be the main objective of our proposed solution to predict. Fig. 2 represents an abstraction of our problem, where we assume the existence of an ML techniques store containing ML models and several heterogeneous computing devices. The previously explained matrix E of user-item ratings in this system represents the execution performance of matching pairs of devices and ML technique pairs. When a new ML technique is registered

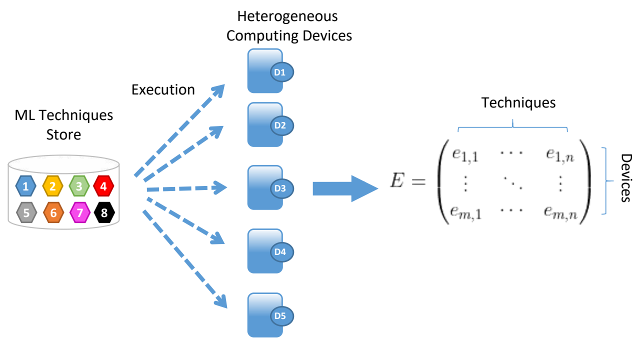
Heterogeneous Computing Devices
Execution
ML Techniques Store
Techniques
Devices
D1
D2
D3
D4
D5
$E = \begin{pmatrix} e_{1,1} & \cdots & e_{1,n} \\ \vdots & \ddots & \vdots \\ e_{m,1} & \cdots & e_{m,n} \end{pmatrix}$

Fig. 2: Running ML technique on heterogeneous computing devices

throught the intent-based orchestrator, as described in section III-B e.g. **”add ML-technique *MobileNet-V2-threat_1* to ML-technique type *threat-detection*”**, we add a new raw to the performance matrix E with empty zeros. After that, the *ML Recommender* chooses randomly a set of devices for the benchmarking of new ML technique. The collected performance information is then added to E as shown in Fig. 3. a., then the SVD method is used to estimate the performance on the rest of devices using eq.1, 6, 5, and 4 in order to minimize the error defined in eq. 3. Finally the *ML Recommender* saves the new performance entries into the *ML techniques store*.

C. System Scaling

An important factor to be considered in an IBN system design, which also is its salient feature, is the scalability. To this end, we consider a scenario where a new hardware device is added to the system. We start with the assumption that there is no any prior knowledge about the performance of ML techniques on a newly added device. The procedure of performance estimation in this case, is similar to the procedure used for a new incoming technique, triggered by the user from the intent-based orchestrator, e.g. **”add device *edge_100* to domain *warehouse_5*”**. The estimation process again follows two phases, warm-up and online, where the warm-up phase is the same as used previously in Fig. 2. When a new device is added, the performance of existing ML-based techniques on that device needs to be benchmarked. In Fig.3.b, an illustration of this process is shown. Assuming that k techniques are chosen from the ML techniques store, they are executed one by one on the new device, and with their performance results collected. After this, a new column is added to the matrix E associated to the new device, with the entries corresponding to each benchmarked technique filled. Next, the new matrix is normalized, followed by applying the previously explained SVD algorithm IV-A, where eq.1, 6, 5, and 4 are used for error minimization in eq. 3. The missing performance estimation is then obtained, out of all the registered techniques in the system, saved in the *ML techniques store*, which allows us to scale up, migrate or replicate using the new hardware device.

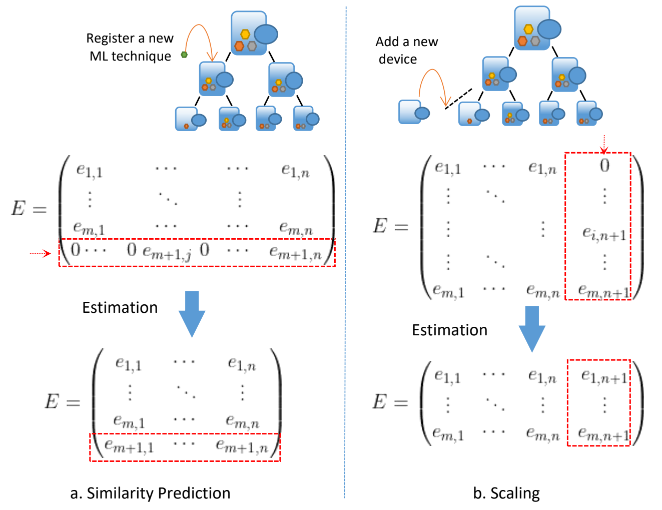

Fig. 3: Performance estimation of a ML techniques on a newly added device.

V. Performance Evaluation

To evaluate the performance of our proposed ML algorithm, an online store of AI benchmark experiments is used [11]. It contains numerous AI tasks and deep learning architectures, tested on multiple hardware platforms. Most popular DL architecture like *MobileNet-V2* (classification), *Inception-V3* (classification), *VGG-19* (image-to-image mapping) and *LSTM* (sentence sentiment analysis), and *DeepLab* (image segmentation) are evaluated in terms of training time and inference time (per one image), as well as memory utilization. An updated dataset from 2019 containing 42 different AI tasks built for desktops is used in our work [12]. Popular hardware platforms, such as CPUs, GPUs and TPUs, are configured to run deep learning models. In the store, 192 different hardware platforms are tested, e.g. *Tesla V100 SXM2 32Gb*, *NVIDIA TITAN V*, *GeForce GTX 1080 Ti*, *AMD Threadripper 3970X*, *Intel Xeon Gold 6130*, etc.

To emulate the behavior of the proposed ML recommender, we assume that our system consists of 191 heterogeneous hardware units, and an *ML techniques store* of 42 AI tasks. A new hardware is added to our network, which needs to be tested in order to collect the performance of all the existing AI tasks in the store. The *knowledge base* of the intent-based orchestrator is initialized with a subset of the existing AI tasks and a subset of the existing hardware platforms. The *ML Recommender* is triggered throught the *dashboard* using the following intents: **”add ML-technique *MobileNet-V2* to ML-technique type *threat-detection*”**, where the *ML-tech-id* is replaced by a different ML technique from the store for every replication e.g. ***Inception-V3*, *VGG-19***, etc. After that SVD algorithm runs to predict the benchmark values of missing hardware devices. The metric used to evaluate the machine learning algorithm is the normalized root mean squared error (RMSE) and can be expressed as follows:

$$\text{RMSE} = \sqrt{\frac{\sum_{i=1}^{N}\sum_{j=1}^{N}\epsilon_{i,j}^2}{N^2}} \tag{7}$$

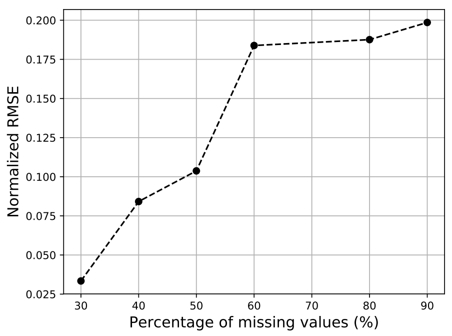

Fig. 4: Performance of benchmark prediction using Normalized RMSE with different settings of missing benchmarks.

$$\text{normalized RMSE} = \frac{\text{RMSE}}{\max R - \min R} \tag{8}$$

where R is the matrix that contains predictions, $\max R$ and $\min R$ represent the maximum and minimum values in the matrix.

The number of iterations is set to 5000, the latent factors are set to 10, learning rate is 0.04, and $\beta = 5 \cdot 10^{-6}$. In figure 4 the percentage of missing benchmark is varied from 30% to 90%. For each test, 5 replications are evaluated where a random device is chosen for benchmark prediction for every replication. Then the average normalized RMSE is calculated and plotted. The performance of prediction is expected to decrease (normalized RMSE increase) when the percentage of missing values increases. Thus, the more benchmarks we obtain from real measurements, the better our estimation of the AI tasks performance on other hardware devices. For a 30% of missing devices, the normalized RMSE is equal to 0.03, which is a promising results. However the normalized RMSE for 90% of missing benchmarks is very high around 0.2, which needs to be improved, in order to use our system for practical applications, where benchmarking techniques are time consuming, and when scheduling must happen in real time using accurate performance evaluations.

VI. Conclusion and Future work

With the rise of heterogeneous devices in complex ICT systems and the rise of machine learning based solutions that are implemented to solve problems in every domain of application, it is becoming an imperative to study the challenges of managing ML solutions running in such systems in an innovative way. We first proposed intent-based networking (IBN) solution as the approach for intelligent managing of ICT supply chain systems. Afterwards, we studied the problem of benchmarking these ML solutions, through a module called *ML Recommender*. This module is used for estimating the computational performance of ML-based techniques on heterogeneous hardware devices, considering the lack of explicit knowledge about the number of operations, framework, or the device processing characteristics for each ML-based technique. Collaborative Filtering techniques like Singular Value Decomposition (SVD) are used to estimate the missing values of the benchmark input matrices, which is planned to be used to enhance the management system capabilities of scheduling, planning or hardware provisioning. A dataset of 42 ML model benchmarked on 196 different hardware platform were used for testing the proposed algorithm, where the results are promising in terms of estimation correctness. As a future work, we plan to extend our system design of the ML recommender to consider the scheduling and planning problem, and to show how can a user requirement given as a high-level intent be automatically translated into scheduling decisions using our benchmarking solution.

Acknowledgment

This work is partially funded by European Commission under the H2020-952644 contract for project FISHY: A coordinated framework for cyber resilient supply chain systems over complex ICT infrastructures.

References

[1] C. Delimitrou and C. Kozyrakis, "Paragon: Qos-aware scheduling for heterogeneous datacenters," *ACM SIGPLAN Notices*, vol. 48, no. 4, pp. 77–88, 2013.

[2] A. Rafiq, A. Mehmood, T. Ahmed Khan, K. Abbas, M. Afaq, and W.-C. Song, "Intent-based end-to-end network service orchestration system for multi-platforms," *Sustainability*, vol. 12, no. 7, 2020. [Online]. Available: https://www.mdpi.com/2071-1050/12/7/2782

[3] W. Cerroni, C. Buratti, S. Cerboni, G. Davoli, C. Contoli, F. Foresta, F. Callegati, and R. Verdone, "Intent-based management and orchestration of heterogeneous openflow/iot sdn domains," in *2017 IEEE Conference on Network Softwarization (NetSoft)*, 2017, pp. 1–9.

[4] G. Carrozzo, M. S. Siddiqui, A. Betzler, J. Bonnet, G. M. Perez, A. Ramos, and T. Subramanya, "Ai-driven zero-touch operations, security and trust in multi-operator 5g networks: a conceptual architecture," in *2020 European Conference on Networks and Communications (EuCNC)*, 2020, pp. 254–258.

[5] M. Bensalem, J. Dizdarević, F. Carpio, and A. Jukan, "The role of intent-based networking in ict supply chains," *arXiv preprint arXiv:2105.05179*, 2021.

[6] P. H. Gomes, M. Buhrgard, J. Harmatos, S. K. Mohalik, D. Roeland, and J. Niemöller, "Intent-driven closed loops for autonomous networks," *Journal of ICT Standardization*, pp. 257–290, 2021.

[7] D. Nemirovsky, T. Arkose, N. Markovic, M. Nemirovsky, O. Unsal, and A. Cristal, "A machine learning approach for performance prediction and scheduling on heterogeneous cpus," in *2017 29th International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD)*. IEEE, 2017, pp. 121–128.

[8] H. M. Makrani, H. Sayadi, T. Mohsenin, S. Rafatirad, A. Sasan, and H. Homayoun, "Xppe: cross-platform performance estimation of hardware accelerators using machine learning," in *Proceedings of the 24th Asia and South Pacific Design Automation Conference*, 2019, pp. 727–732.

[9] I. Hafnaoui, R. Ayari, G. Nicolescu, and G. Beltrame, "A simulation-based model generator for software performance estimation," in *Proceedings of the Summer Computer Simulation Conference*, 2016, pp. 1–8.

[10] R. M. Bell, Y. Koren, and C. Volinsky, "The bellkor 2008 solution to the netflix prize," *Statistics Research Department at AT&T Research*, vol. 1, 2008.

[11] "Deep learning hardware ranking:desktop gpus and cpus." [Online]. Available: https://ai-benchmark.com/ranking_deeplearning_detailed.html

[12] A. Ignatov, R. Timofte, A. Kulik, S. Yang, K. Wang, F. Baum, M. Wu, L. Xu, and L. Van Gool, "Ai benchmark: All about deep learning on smartphones in 2019," in *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*. IEEE, 2019, pp. 3617–3635.