Simplifying Network Orchestration using Conversational AI

1st Deven Panchal	2nd Prafulla Verma	3rd Isilay Baran	4th Dan Musgrove	5th David Lu
AT&T	*AT&T*	*AT&T*	*AT&T*	*AT&T*
Middletown, NJ, USA	Middletown, NJ, USA	Middletown, NJ, USA	Middletown, NJ, USA	Dallas, TX, USA
devenrpanchal@gmail.com	pv2985@att.com	ib6391@att.com	dm4812@att.com	dl1971@att.com

***Abstract*—ONAP is a comprehensive platform for orchestration, management and automation of network and edge computing services for 5G, 6G and Next Generation Networks. Unlike traditional OSSs, it is an open-source project where companies all over the world are collaborating to build different functionalities of an end-to-end Network operating system. For this reason, the ONAP platform has several different sub projects and APIs each performing a specific function to achieve Network Management. There is some complexity associated with using these APIs and knowing and understanding the many parameters associated with them, which impedes adoption. This not only prevents an end-to-end cloud service orchestration like experience for network services, but also increases the time and money spent on network orchestration. This paper proposes and discusses the design of a conversational AI solution that can interface with some significant APIs in ONAP to solve these problems. The conversational AI solution has the potential to significantly simplify network orchestration tasks. This work is being further extended to using Large Language Models (LLMs) to achieve simplified Intent-Based management and orchestration paradigms within ONAP.**

***Index Terms*—Open Network Automation Platform (ONAP), Operations support systems (OSS), Machine Learning, Natural Language Processing, Network Orchestration, Intent-Based Networking, Intent Driven Networking, Software Defined Networking, Network Function Virtualization, Open Source, Large Language Models (LLMs), Next Generation Networks, 5G, 6G**

I. Introduction

Open Network Automation Platform (ONAP) [1] is a comprehensive platform to do real-time policy-driven orchestration, management and automation of network and edge computing services. In other words, it is a network operating system that can help the management of different end-to-end lifecycle processes for network functions and services. ONAP massively leverages Software Defined Networking (SDN), Network Function Virtualization (NFV) to operate a vendor-agnostic unified operating framework for management of all types of physical, virtual and cloud-native network entities for 5G, 6G and Next Generation networks, decrease CapEx and OpEx to operate these networks and improve service velocity [2].

ONAP makes a distinction between the activities of service design, service deployment, and service operations. The service design and creation (SDC) component in ONAP allows specification and creation of services by modeling the resources and relationships that make up the services, specifying the policy rules that guide the service behavior and specifying the applications as well as the AI/ML/analytics etc. needed to effect service elasticity. SDC also offers a visual modeling and design interface to design, validate, certify and distribute services. These services and workflows are distributed from the SDC to the runtime framework or service deployment and operations components like the Master Service Orchestrator (MSO), various controllers (APPC, SDNC), active and available inventory (A&AI), Data Collection, Analytics and Events (DCAE) and the security framework to execute the designs and effect the deployment and orchestration of network services.

II. Master Service Orchestrator

The MSO is the main component of the ONAP runtime. It provides the highest level of service orchestration and has an end-to-end view of the infrastructure, network, and applications. It can execute a specified process by automating sequences of activities, tools, rules, and even policies needed for on-demand creation, modification, removal of network, application or infrastructure services and resources including virtual network functions (VNF's), cloud native functions (CNF's) and physical network functions (PNF's). The MSO for example, would be the component that would handle (and delegate southbound to various other controllers like APPC, SDNC, and other components) a request to create a new broadband service (BBS), or say a cross domain, cross layer VPN (CCVPN).

The MSO is designed to have an API handler component that receives orchestration requests on behalf of the MSO. The MSO then looks up its catalog to map the request to BPMN flows based on the service-model and action. The data from the request is also forwarded to these BPMN flows which are supported by the Camunda Platform. The MSO uses the SO Request DB to track open and completed requests. MSO has an SDC distribution client that can receive service models from the SDC and populate them into the MSO catalog. As explained before, the MSO has many resource adapters to accomplish lower level tasks on the southbound, for example, SDNC, APPC, VFC Controllers, Multi-Cloud adapter and Platform orchestrator(PO). The SDN controller (SDNC) or the network controller manages, assigns, and provisions network resources. The application controller (APP-C) performs functions to manage the lifecycle of virtual network functions (VNF's) and their components by providing model driven

configuration, abstracting cloud/VNF interfaces for repeatable actions and automation. The Platform orchestrator manages cloud resources via OpenStack and other technologies. To monitor BPMN workflow executions, MSO has a monitoring component. The MSO can also update the A&AI with the status of various entities [3].

Since the MSO is the main engine for Network orchestration, talking to various Northbound and Southbound clients and adapters, we want to power the MSO APIs with conversational AI to drive their learning, adoption and in the future enable an LLM-powered simplified intent based orchestration mechanism within ONAP.

III. MSO Usecase - Create Service Instance

There are different ways to perform service orchestration using the MSO. The e2eServiceInstance mode runs hardcoded service BPMN workflows.The Á La Carte mode allows you to run generic flows but requires the user to somehow collect all parameter values and send individual orchestration requests at each step to perform the desired process. The Macro mode uses CDS Blueprint templates and requires the user to construct and send only one request and then ONAP collects and assigns all required parameters and values by itself. These modes can be used with the ONAP SO Rest API or the ONAP Virtual Infrastructure Deployment (VID) Portal [4].

The VID Portal [5] provides a visual form-based interface to execute orchestration processes. The MSO receives many complex multi-parameter orchestration requests from the VID. This interaction is shown in the figure 1. For example, at a high-level, when using the VID in Á La Carte mode, to instantiate a service model composed of 2 VNFs and a network, the user of VID will need to send a request to the MSO to create the service instance object, followed by a request to the SO to create the VNF 1 instance object, then a request to SDNC to declare VNF 1 instance parameters and values, then a request to the MSO to create the Vf-module 1 instance object, then request to the MSO to create the VNF 2 instance object, then a request to SDNC to declare VNF 2 instance parameters and values, then a request to the MSO to create the Vf-module 2 instance object and finally a request to the MSO to create the network instance object [6].

So, as an example, let us choose the first 'Create Service Instance' request. Service instances are sort of shell objects in A&AI which can be used to help create other entities like for eg. VM's which would require further interactions with platform orchestrators like Openstack via southbound adapters of the MSO. But we will only look closely at the 'Create Service Instance' request.

Figure 2 shows a sample request to create a service instance sent to the MSO using the VID. We will look at the shortcomings of VID later, which will further reinforce the need for a simpler solution to interface with the MSO APIs. The MSO then responds back to upstream UI with the instance id and the request id if the request is accepted and being processed. Else it returns a requestError. When the MSO accepts the request, the MSO kicks off a Camunda flow which can trigger appropriate downstream requests. The VID Portal polls the status of the request.

We propose a conversational AI based solution within this architecture so that-

- It converses with the user to understand the user's intent and craft the request for the MSO
- It then sends the request to the MSO and accepts synchronous responses from the MSO
- It then checks on the status of the request
- It communicates the status back to the user in a natural language style.

IV. Benefits

The advantages of this over the existing systems are the following –

- Currently systems like VID and other systems that help craft orchestration requests have many forms with multiple parameters that a user has to fill out. This currently requires the user to reach out to the subsystem owners or subject matter experts of all the subsystems that would be used as part of the request. This is inefficient considering the number of very distinct multiparameter requests that could be made to accomplish a single orchestration task, and hence is a barrier to adoption of these APIs. The intelligent conversation AI solution would be a single point of contact to do all these different tasks, and notify the user of missing parameters and values. This would be much more efficient and save user's time. Its ease of use would help lower the barrier to adoption for these APIs.
- The conversation AI solution would have a natural conversation style. Like talking to someone who would guide the user in real time, and handhold the user to accomplish whatever he wishes to using ONAP. For example, when creating a service instance, it could ask you about the missing parameters. This is much more user friendly and helps keep the user engaged. This increased engagement would ultimately drive better learning and adoption of these APIs.
- Recall a barrier to API usage was that a user needed to reach out to specific sub-system SMEs to ask about missing values pertaining to the particular sub-system. When ONAP is deployed, it has been seen that the SMEs and other resources working on these sub-systems change frequently due to organizational and partnership changes. Not only that, the users of these APIs also change frequently. So, a lot of time and money is spent to restaff these roles and for knowledge transfer. The conversational AI solution can easily incorporate question-answering capabilities based on the documentation corpus, to answer the user's questions about subsystems, parameters etc. and even point the user to additional information.
- This conversational AI solution will not only help increase learning and adoption, but it will also help expand the audience/user base of these MSO API's which currently includes technical domain experts to include

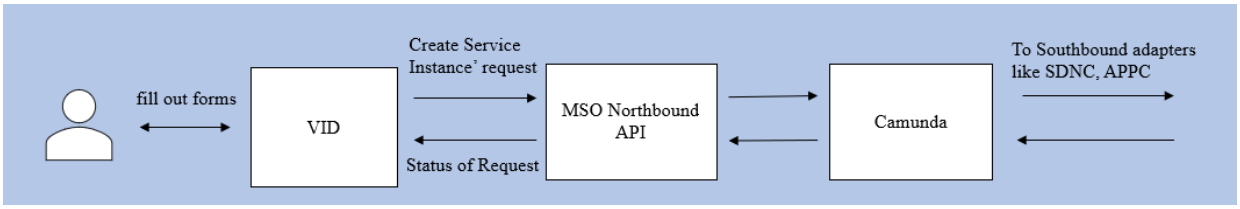

Fig. 1: Present mode of operations with VID and MSO

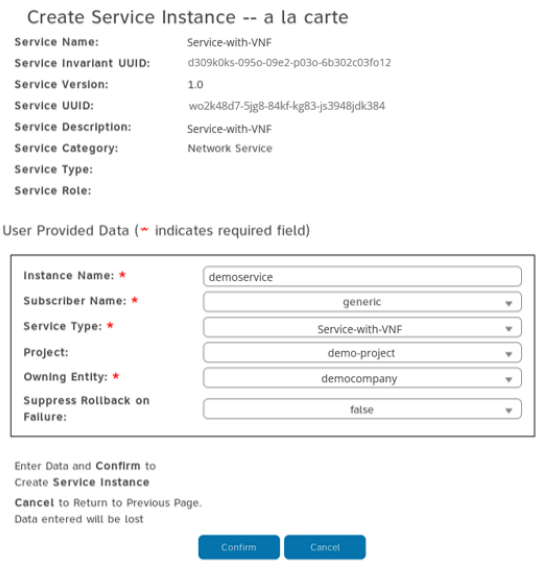

Fig. 2: VID request to Create a Service Instance

non-technical users. For non-technical users, the solution offers a powerful abstraction of software complexities on top of which they can interact with the ONAP deployment functions in a friendlier way.

- It will help provide an AWS, Azure or GCP like cloud orchestration experience when creating and orchestrating network services using ONAP. We anticipate that the user experience can in fact be made much better than the experience offered by the current cloud providers.
- The conversational AI solution is a step in the direction towards enabling a simplified Intent-Based Networking paradigm using LLMs in ONAP. We will talk about this ongoing work in the 'Ongoing and Future Work' section.

V. Intent Recognition and NLP Modeling in RASA

For our NLP and NLU tasks in this work, we use Rasa [7], which is an open-source machine learning framework for automated text, and voice-based conversations. It can be used to understand messages, converse, and connect to messaging channels and APIs. Let's quickly understand some concepts that are basic to Rasa or in general any Intent Processing framework. 'Intent' captures what the user is intending to ask about. What is it that the intent recognizer must register as the intent when people say something to the conversational AI? For it to recognize what a user is saying no matter how the user phrases their message, we need to provide example messages conversational AI can learn from. We group these example utterances according to the idea or the goal the message is expressing. This is the intent. 'Entities' are structured pieces of information that can be extracted from a user's message. 'Actions' are the tasks the interface should trigger upon encountering a specific intent. It is possible to trigger default actions or highly complex custom actions. Rasa also helps define 'stories' which are example conversation flows that train conversational AI to respond correctly depending on what the user has said previously in the conversation. You also have 'Forms' in Rasa which may be needed when conversational AI needs to collect specific information like Name, Email, etc. from the user. 'Rules' describe parts of the conversation that always follow the same path irrespective of the previous context [7]. Thus, with the help of stories, rules and forms it is possible to configure a custom level of deterministic behaviour in the conversational AI interface for all intents to achieve what we have been talking about in the earlier sections. We programmed a Rasa NLU/NLP pipeline to include components to do preprocessing, intent recognition, entity extraction, response selection and initiating custom actions. These components work sequentially to process user input into structured output. We experimented with many different tokenizers, featurizers, intent classifiers and entity extractors. Constructing more diverse training examples for the training set helped us improve the intent recognition accuracy. We finally settled on using the Dual Intent and Entity transformer (DIET) classifier which is a multitask (intent plus entity) transformer architecture. In literature, it has been tested to outperform fine tuned BERT and is 6 times faster to train [8] in comparison. There are also many open-source and closed source alternatives to frameworks like Rasa. The Rasa powered conversational AI can itself be interacted with using a shell or a browser, or easily integrated into a website or Facebook messenger, Slack, Telegram, Twilio, Google hangouts, Microsoft Bot Framework, Cisco Webex Teams, etc. You can also connect to other platforms using your own custom connectors. We use Rasa X, which is a browser based interface that we will use to converse with the Rasa conversational AI. Rasa X wraps itself on top of Rasa and can be used to do many things - view and filter conversations, share the conversational AI solution with test users, manage models and training data, and convert conversations collected from test conversations and real user interactions, into training

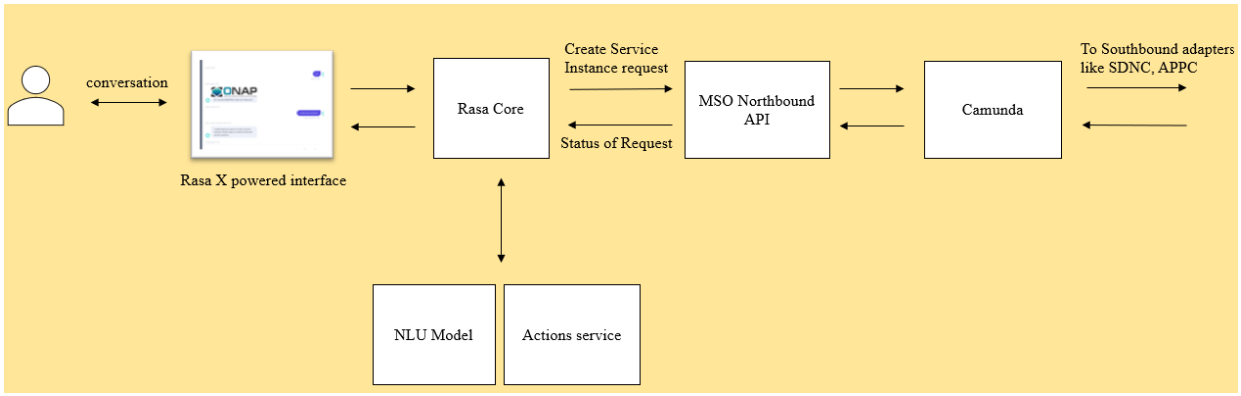

Fig. 3: Request flow with the proposed NLP solution

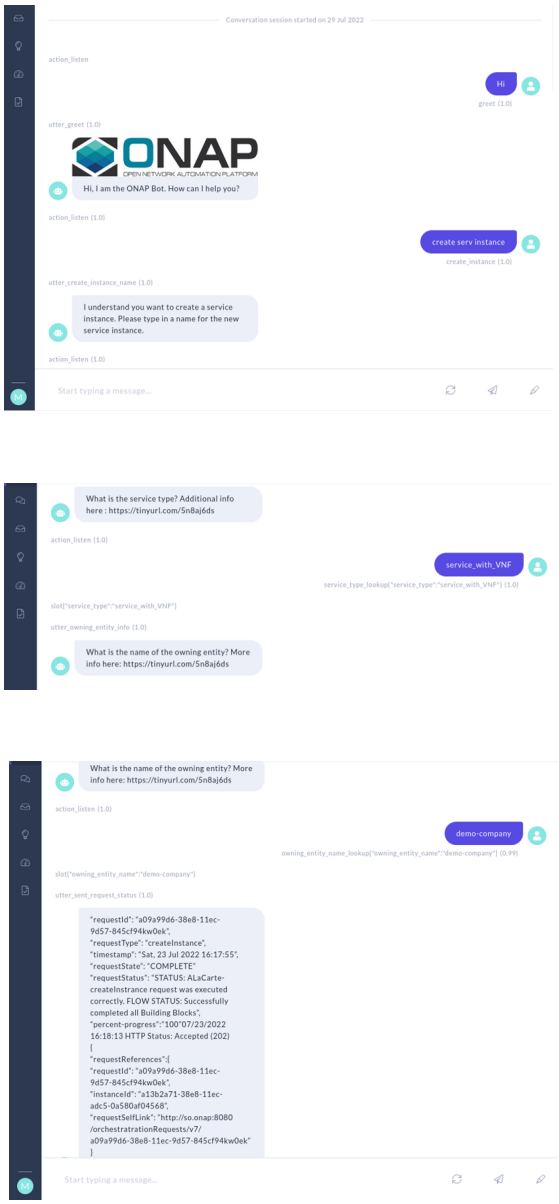

Fig. 4: Conversational AI Interface for ONAP

data that can used to re-train and improve the NLU model.

VI. Results

In the conversational AI interface, we see that the user asks it to create a service instance in a free-form natural language style. Conversation AI recognizes this intent of the user and asks the user the instance name, subscriber name, service type, owning entity name and various other parameters needed to create and fire a 'Create Service Instance' request. With access to the A&AI it can also list possible parameter options based on what is available and permitted for the current user. With access to the extensive doucmentation ONAP provides, it can help answer any questions for more information about the process, parameters etc.

It then creates and fires the request to the MSO and comes back with a response from the MSO that it prints in a style that the requester can understand.The figure 5 shows the completed flow in Camunda which created a service instance. In this paper, we have treated 'Create Service Instance' task to show how the architecture works, but we can perform much more complex network orchestration tasks using this architecture.

VII. Conclusion

MSO is at the core of ONAP which can handle the orchestration and management of a Service Provider's entire network. MSO connects to many southbound resource adapters like the Network controller, Application controller, VFC Controllers, Multi-Cloud adapter and Platform orchestrator(PO), and can directly or indirectly interface with other components like Policy, CLAMP, and even external platforms like ORAN. This is the reason we chose MSO APIs. We saw how an intent recognition and conversation solution can benefit ONAP, and can help simplify network orchestration and save time, money and efforts spent on it by helping drive learning and adoption and usage. We anticipate that with easier interfaces like these, the user experience for users of large platforms like ONAP, other Network Management Platforms, OSS's/BSS's when can be made much better.

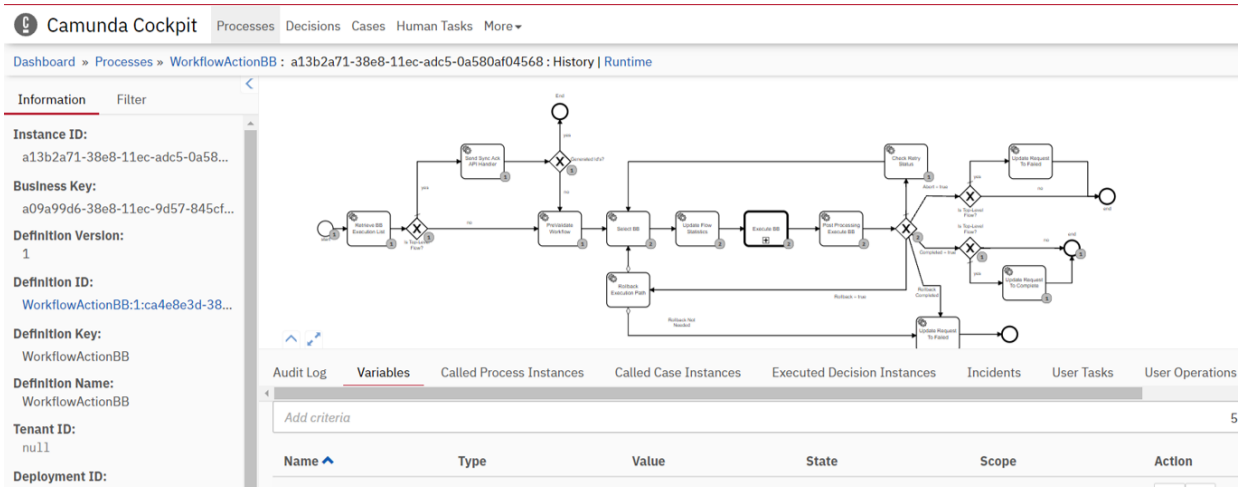

Fig. 5: Completed flow in the Camunda Cockpit

VIII. Ongoing and Future Work

This architecture also lends itself very well to the Intent-Based Networking (IBN) paradigm. A conversational AI interface like the one we saw would become an important part of the IBN system. So it is worth noting that it can also be architected using custom Named Entity Recognition (NER) trained to recognize domain specific entities, and using semantic similarity powered by sentence transformers [9] or even vector databases like Chroma [10] and Milvus [11] that employ multiple algorithms like Approximate Nearest Neighbors (ANN), Hierarchical Navigable Small World (HNSW), Locality-Sensitive Hashing (LSH), Inverted Multi-Index (IMI), Product Quantization (PQ), etc. for indexing and searching high-dimensional vectors. In the past year, however, LLMs have emerged to be the technology of choice and can take over most of the responsibilities of the individual NLP/NLU components we mentioned. Retrieval Augmented Generation [12] or Fine Tuning [13] of Foundational models are both approaches that can offer better accuracy than most NLP/NLU components we discussed earlier. For ONAP however, 2 concerns are primary - one, because ONAP is open-source, the foundational model should also be open source and secondly, because ONAP is a complex scalable system with multiple components that can orchestrate an entire service provider's network, sometimes serving very low latency requests of the order of milliseconds, we would need a scalable, near real-time solution. To this end, the accuracy of the underlying model or models for various specific tasks can be increased for e.g. using fine-tuning and designing better training examples. Quantization [14] and Parameter Efficient Fine Tuning (PEFT) techniques like QLoRA [15] can help achieve better scalability and lower latency. Coming back to our discussion on the IBN system, it could be used for e.g. to perform Network Slicing, Service Provisioning, other LCM actions on the various network entities or Network troubleshooting etc. in an auto-pilot mode or Human-in-the-loop (HITL) mode. High-level intents expressed in free-form natural language can be translated to policies (network, security, etc.) which can then be used to call the ONAP MSO and via its various controllers, effect various configuration changes in the physical or virtual infrastructure. Since ONAP employs a cross cutting logging framework for all its components, ONAP deployment logs contain audit, security, performance data for all components. Besides this, math agents and various other (Large Language Models) LLM agents, both freely available and customized can be used to look at fault, performance, events, counters and KPI data that the ONAP Data Collection, Analytics and Events (DCAE) [16] collects from the live network. We are working to design the IBN system, so that it could contain all these and other elements to define and manage the intent of a schema if required, trigger appropriate ONAP APIs and also store the results of Intent-based executions.

References

[1] ONAP a Series of LF Projects, "Open Network Automation Platform," https://www.onap.org/.

[2] ONAP, "ONAP Architecture Index," https://docs.onap.org/en/latest/platform/architecture/index.html.

[3] ——, "SO - Architecture," https://docs.onap.org/projects/onap-so/en/latest/architecture/architecture.html.

[4] ——, "Instantiate with MSO," https://docs.onap.org/projects/onap-so/en/latest/developer_info/instantiate/index.html.

[5] ——, "VID Documentation," https://docs.onap.org/projects/onap-vid/en/latest/.

[6] ——, "A La Carte mode Service Instantiation via ONAP SO API," https://docs.onap.org/projects/onap-so/en/latest/developer_info/instantiate/instantiation/so1/index.html.

[7] Rasa Technologies GmbH, "Introduction to Rasa Open Source," https://rasa.com/docs/rasa/.

[8] Rasa Technologies Inc, "Introducing DIET," https://rasa.com/blog/introducing-dual-intent-and-entity-transformer-diet-state-of-the-art-performance-on-a-lightweight-architecture/.

[9] Hugging Face, "Sentence Transformer all-MiniLM-L6-v2," https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2.

[10] Chroma, "Chroma," https://www.trychroma.com/.

[11] Milvus, "Milvus," https://milvus.io/.

[12] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W. tau Yih, T. Rocktäschel, S. Riedel, and D. Kiela, "Retrieval-augmented generation for knowledge-intensive nlp tasks," 2021.

[13] K. Lv, Y. Yang, T. Liu, Q. Gao, Q. Guo, and X. Qiu, "Full parameter fine-tuning for large language models with limited resources," 2023.
[14] X. Wu, H. Xia, S. Youn, Z. Zheng, S. Chen, A. Bakhtiari, M. Wyatt, R. Y. Aminabadi, Y. He, O. Ruwase, L. Song, and Z. Yao, "Zeroquant(4+2): Redefining llms quantization with a new fp6-centric strategy for diverse generative tasks," 2023.
[15] T. Dettmers, A. Pagnoni, A. Holtzman, and L. Zettlemoyer, "Qlora: Efficient finetuning of quantized llms," 2023.
[16] The Linux Foundation, "Data Collection, Analytics and Events (DCAE)," https://wiki.onap.org/pages/viewpage.action?pageId=1015831.