Multi-Partner Project: Orchestrating Deployment and Real-Time Monitoring - NEPHELE Multi-Cloud Ecosystem

Manolis Katsaragakis[⋆], Orfeas Filippopoulos[⋆], Christos Sad[•], Dimosthenis Masouros[⋆], Dimitrios Spatharakis[†], Ioannis Dimolitsas[†], Nikos Filinis[†], Anastasios Zafeiropoulos[†], Kostas Siozios[•], Dimitrios Soudris[⋆], Symeon Papavassiliou[†]
[⋆]*Microprocessors and Digital Systems Laboratory, National Technical University of Athens, Greece*
[•]*Department of Physics, Aristotle University of Thessaloniki, Greece*
[†]*Network Management & Optimal Design Laboratory, National Technical University of Athens, Greece*
[⋆]{mkatsaragakis, ofilippopoulos, dmasouros, dsoudris}@microlab.ntua.gr, [•]{csant,ksiop}@auth.gr
[†]{dspatharakis, jdimol, nfilinis}@netmode.ntua.gr, tzafeir@cn.ntua.gr, papavass@mail.ntua.gr

***Abstract*—The rapid growth of Internet of Things (IoT) devices and emerging technologies, along with the growing demands of edge-deployed applications, has led to a complex paradigm where computation often shifts dynamically acrooss the IoT-Edge-Cloud continuum. The NEPHELE project addresses these complexities by enabling seamless orchestration across a diverse spectrum of computing resources, spanning multi-cloud environments to the far-Edge. In this paper, we present NEPHELE's multi-cloud infrastructure, built to overcome key orchestration challenges within cloud and edge environments. We discuss the core components and architectural decisions, focusing on multi-cluster resource orchestration mechanisms, integrated monitoring for local and multi-cloud systems, inter- and intra-cluster scaling, and networking capabilities. Experimental results demonstrate the efficiency of our infrastructure, highlighting overhead management in service deployment, migration, networking, and scaling scenarios, thus highlighting NEPHELE's robustness in handling distributed applications across heterogeneous environments.**

***Index Terms*—NEPHELE, Cloud, Edge, Multi-Cloud, Orchestration, Network**

I. Introduction

Over the last years, the rapid growth of next-generation Internet of Things (IoT) and Edge Computing technologies leads to the composition of a fully interconnected digital landscape [1]. This evolution, in addition to the increasing diversity of computing platforms, including Graphic Processing Units (GPUs), low-power IoT nodes, hardware accelerators, and microcontrollers [2], contributes to an increasingly complex computing paradigm. This complexity is further extended by the explosion of applications deployed near the end-user from a wide application spectrum, such as ML-enabled/AI-assisted workloads [3], [4], biomedical and healthcare applications [5].

However, Edge and IoT nodes typically are resource-constrained, providing limitations on the Quality of Service (QoS) and scalability [6]. Thus, a portion of computation generated on the Edge is offloaded to the Cloud [7] to leverage high-performance hardware, perform complex processing, and store vast amounts of data. However, as demands increase and applications require greater flexibility, scalability, and availability, relying only on a single cloud provider is often insufficient [8]. This shift is driving the need for multi-cloud solutions [9], [10], which allow vendors to distribute workloads across multiple cloud platforms. Commercial vendors have already considered the integration of multi-cloud orchestration mechanisms, such as Google Anthos [11] and Microsoft Azure Arc [12]. Multi-cloud strategies enhance resilience and enable performance and cost optimization based on user-specific requirements [8].

NEPHELE [13] is a 3 year Research and Innovation Action (RIA) project that aims to streamline the orchestration of applications deployed across a diverse range of computing resources, spanning multi-cloud environments to the far-edge, and extending to IoT systems. By enabling seamless interaction across these layers, NEPHELE provides a unified framework that supports efficient application deployment and management in complex, distributed environments. The project builds and demonstrates its platform and targets real-world applications in demanding fields, including emergency and disaster response, AI-driven logistics operations, energy management, and healthcare services [14]. These application areas require high levels of resource adaptability, low latency, and robust network connectivity, making them ideal to showcase NEPHELE's capabilities.

Specifically, NEPHELE integrates a sophisticated, cloud-enabled computing platform tailored to efficiently process a range of application workloads. This platform includes advanced capabilities such as multi-cloud deployment, autoscaling, robust inter-cluster communication, and customizable metrics exporters, among others. NEPHELE's cloud-centric design facilitates the sharing of a unified testbed that spans diverse applications, simplifying deployment, access, and resource management. Additionally, NEPHELE provides an open, extensible testbed for ecosystem stakeholders and external users, allowing them to leverage and contribute to the project's technologies. This collaborative framework encourages broad-based innova-

This work has been partially funded by EU Horizon program NEPHELE under grant agreement No 101070487 (https://nephele-project.eu/) and the program with No 65015306 for funding of research activities of scientific responsibilities.

tion, supporting the development of sustainable, resilient multi-cloud infrastructures across various use cases.

The NEPHELE project is currently at an intermediate phase, offering an opportunity to detail its technical progress, key achievements, addressed challenges, and the objectives still to be accomplished before project completion. At this stage, the integration of core technologies has been successfully implemented. In the final phase, NEPHELE will focus on incorporating specific use cases and applying insights gained from the project's open calls [15], particularly around distributed applications and advanced orchestration mechanisms.

In this paper, we (i) provide the technical implementation of the NEPHELE architecture described in prior research [16], aiming to enable application deployment across the IoT-to-Edge-to-Cloud continuum, (ii) we discuss NEPHELE's objectives, goals, and key challenges towards the end of the project, (iii) we introduce and examine the multi-cloud infrastructure designed to meet these objectives and (iv) conduct a series of experiments to demonstrate the robustness of our infrastructure in terms of deployment and migration overheads, network performance and scalability. These experiments validate NEPHELE's resilience and efficiency, ensuring it can handle the demands of complex, distributed applications across diverse environments. During the second phase of the project we aim to build on top of the existing infrastructure advanced orchestration mechanisms aiming to optimize the final deployment of distributed applications.

The rest of this paper is organized as follows. In Section II we present the overall architecture of the NEPHELE platform, goals and open challenges. Section III presents the multi-cloud infrastructure designed for NEPHELE. Next, in Section IV we present the experimental analysis of NEPHELE multi-cloud architecture and discuss open challenges by the end of the project. Finally, Section V concludes this work.

II. NEPHELE Architecture Overview

In this section, an overview of the NEPHELE architecture is presented. NEPHELE framework follows a system of systems approach where a high-level entity coordinates multiple layers and components that work seamlessly together to orchestrate hyper-distributed applications in the computing continuum. Hence, the control and responsibilities are distributed between different levels of the framework to facilitate the overall life-cycle management of the deployment of an application graph. Specifically, as shown in Fig. 1, NEPHELE is comprised of four layers designed in a bottom-up approach, namely; (i) the Edge/Cloud Manager, (ii) the Multi-Cluster Manager, (iii) the Network Manager, and (iv) the Synergetic Meta-Orchestrator. Each layer thrives on solving the main challenges and complexities of orchestration in the computing continuum. A brief description of the main challenges, responsibilities, and core functionalities of each layer is presented in what follows.

Convergence with IoT - Virtual Object: One of the main challenges that the NEPHELE framework tries to address is the convergence of edge/cloud technologies with IoT devices. To this end, NEPHELE introduced the concept of Virtual Object [17], which is the virtual counterpart of IoT devices.

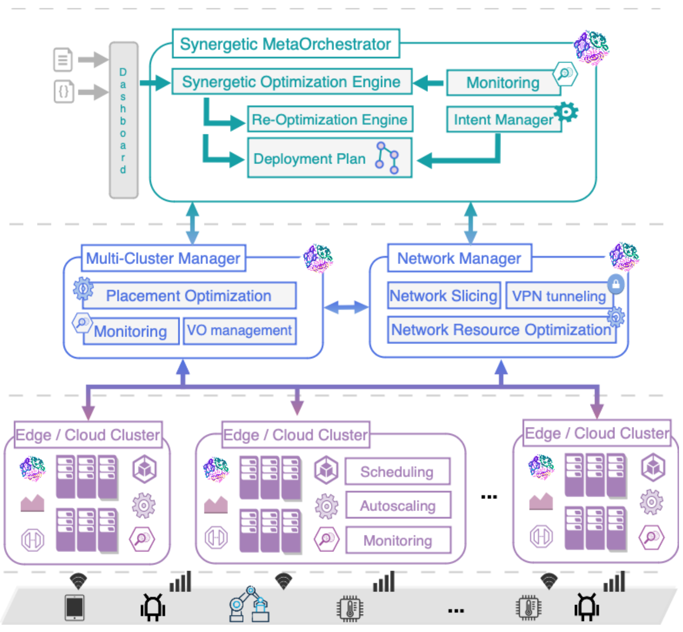


Fig. 1: Overview of the NEPHELE Architecture

The VO is based on the W3C Web of Things standards[1], is deployed as a lightweight software stack[2], on the edge or cloud infrastructures and is part of the application graphs. The VOs are trying to solve three main challenges, namely; (a) protocol interoperability between the applications and the IoT devices, (b) trusted communication by implementing various out-of-the-box security mechanisms for authorization/authentication, and (c) extending the functionalities of resource-constrained IoT devices with a set of functions. More details for the functionalities and architecture of VOs can be found in [17].

Edge/Cloud Resource Manager: The first layer in the computing continuum of the NEPHELE framework is the Edge/Cloud Resource Manager which oversees the deployment of application graphs within the edge/cloud infrastructure. For each cluster under the control of NEPHELE, the Edge/Cloud Resource Manager, dictates local orchestration actions by leveraging various optimization techniques to ensure the fulfillment of the requirements of the application graphs. Specifically, these techniques mainly focus on (a) scheduling of applications within the nodes of the cluster to minimize the infrastructure's efficiency and (b) autoscaling of applications according to the workload traffic to meet performance criteria. Moreover, the Edge/Cloud Resource Manager is also responsible for monitoring critical performance metrics involving both the application and the infrastructure.

Multi-Cluster Manager: The Multi-Cluster Manager is responsible for managing all computational resources, treating them as a unified continuum that spans IoT devices, edge computing nodes, and powerful cloud clusters. Hence, the key functions of the Multi-Cluster Manager involve (a) a centralized and unified interface for administration and operation, (b)

[1]https://www.w3.org/WoT/
[2]https://netmode.gitlab.io/vo-wot/

placement of application graphs, i.e. application components representation (Sec. II-A) with the goal of efficient resource allocation and load balancing [18] across different clusters and (c) VO management and orchestration and (d) security and policy management to maintain data privacy of applications and VOs. These features collectively ensure that NEPHELE's infrastructure can efficiently meet the demands of diverse applications while maintaining high levels of control and resilience across the computing continuum.

Network Resource Manager: The Network Manager is a critical component that maintains optimal network performance for application graph deployments across the multi-cluster ecosystem. It operates as an intelligent layer that ensures consistent, secure, and efficient connectivity among diverse clusters. Beyond managing network performance guarantees, it fulfills several essential functions, namely: (a) secure multi-cluster connectivity and VPN tunneling, (b) the end-to-end network slice orchestration and network resource optimization, and (c) scaling of the infrastructure to achieve load balancing and efficient resource utilization.

Synergetic Meta-Orchestrator: The Synergetic Meta-Orchestrator (SMO) is the main entity of the NEPHELE framework and is responsible for the overall orchestration of the hyper-distributed applications over the available infrastructure. It coordinates with various other layers in the architecture to create and distribute a deployment plan. Specifically, the SMO receives a descriptor for the application graph, which includes the application's specifications, along with a user-defined intent that outlines the desired performance levels, application deployment requirements, and constraints. It undertakes deployment requests through the dashboard, preparing deployment plans by considering both the current status and available resources of the underlying infrastructure as well as user-defined requirements coming from the intent. The SMO works with the Network Manager and the Multi-Cluster Manager to make decisions related to the network and computing infrastructure, respectively. While continuous monitoring of the NEPHELE unified continuum, the SMO acts as a (re-)optimization mechanism that triggers the necessary components in the lower levels of the hierarchy of the framework, depending on the application-/infrastructure-related events.

A. NEPHELE Orchestration Goals/Challenges

As discussed previously this paper mainly deals with the technical challenges for the orchestration of application graphs within the computing continuum. Therefore, here we summarize the main orchestration goals and challenges we identified during the development of NEPHELE.

Deployment and Scheduling of an Application Graph: An application graph represents the application's components (such as services, microservices, or tasks) as interconnected nodes, where edges denote data flows or dependencies among them. Deploying an application graph across a multi-cluster infrastructure presents several challenges mainly considering identifying the appropriate resources, dealing with the graph network and compute dependencies, seamless lifecycle management and assuring user's node/cluster affinity rules [19]. More importantly, scheduling the components of the application graph while targeting several competing metrics such as latency, resource optimization, power efficiency, and other.

Multi-Cluster Discovery and Connectivity: establishing and maintaining connectivity and information exchange across multiple clusters is essential yet challenging [20]. Continuously identifying and updating available resources across clusters can be complex due to synchronization issues, while the secure and trusted connectivity between the applications is crucial for modern applications.

Multi-Cluster Monitoring Stack: monitoring performance, resource utilization, and application-related metrics across a multi-cluster environment present challenges due to the distributed and dynamic nature of such systems [20]. The collection and aggregation of data while achieving scalability of the monitoring tools can be challenging and resource-consuming. Moreover, real-time alerting and detection are difficult especially when dealing with high volumes of data.

Application Graph Event-driven Autoscaling: timely autoscaling of application graphs is challenging when dealing with distributed and heterogeneous environments [21]. The orchestration of the application graphs aims to facilitate efficient resource utilization in response to the workload demands across clusters. Dynamic autoscaling must balance between under-provisioning (service degradation) and over-provisioning (unnecessary utilization of resources) under workload or infrastructure changes. Thus, rapidly detecting and propagating events across clusters is essential to trigger autoscaling actions.

III. NEPHELE Technical Implementation

In this section, we present NEPHELE's multi-cloud architecture and testbed. We aim to address the orchestration goals presented in Sec. II-A and provide a solid infrastructure to tackle challenges discussed in Sec. IV-C. NEPHELE provides a multi-cloud infrastructure designed for the deployment of application workloads, featuring capabilities such as multi-cloud deployment, autoscaling, inter-cluster communication, and custom metrics exporters. Figure 2 illustrates the multi-cloud architecture of the NEPHELE framework. The effective operation of the NEPHELE multi-cloud architecture relies on the efficiency of the synergy of the underlying components, ranging from a unique cloud cluster, up to the host cluster.

Application Graph Deployment: Our multi-Cloud setup relies on Karmada [22]. We adopt the Karmada and use its multi-cloud architecture to manage discrete Kubernetes clusters [23]. It enables orchestration across public and private cloud environments, balancing resource utilization and ensuring high availability. Moreover, Karmada enables orchestration across public and private cloud environments for balancing resource utilization and ensuring high availability. With Karmada, we drive workload distribution across different clusters with policy-driven scheduling, making it easier to scale applications. Karmada's policy-driven scheduling simplifies application graph distribution, ensuring that we maintain a resilient and scalable infrastructure capable of meeting dynamic resource demands.

Multi-Cluster Discovery and Connectivity: To enable service continuity and inter-cluster communication, NEPHELE in-

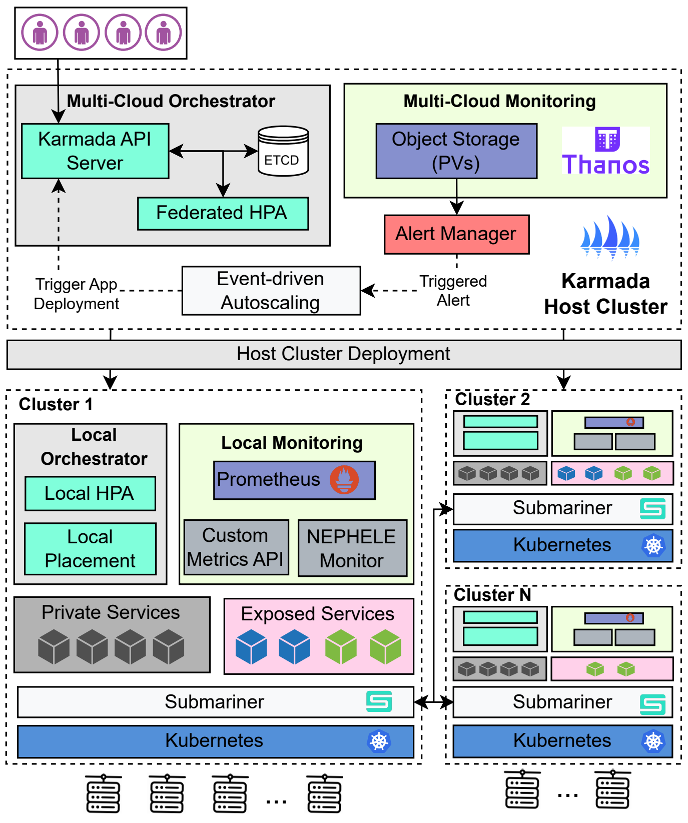


Fig. 2: Overview of the NEPHELE Multi-Cloud Infrastructure

tegrates Karmada's federated service discovery along with Submariner [24]. This combination allows microservices deployed across multiple clusters to autonomously discover, communicate, and collaborate across the cloud continuum. By leveraging Karmada's federated discovery capabilities, NEPHELE establishes a unified service namespace across clusters, simplifying service discovery and reducing latency in inter-cluster communication. Submariner complements this setup by creating secure cluster communication, enabling encrypted and low-latency data transmission that meets stringent security requirements. Together, Karmada and Submariner form a robust framework for resilient, cluster-to-cluster microservice communication.

Multi-Cluster Monitoring Stack: The effective operation of our flow strongly relies on Prometheus and a custom Metrics API, which allows us to fetch real-time metrics and dynamically adjust the replica count of our workloads. We use Prometheus [25] and Thanos [26] deployment to fully integrate our monitoring and alerting system across our multi-cluster setup. Prometheus handles real-time metric collection, while Thanos manages long-term storage and cross-cluster querying. Custom metrics are also integrated through a dedicated API, allowing for precise monitoring of application-specific metrics. Aiming to export user application metrics to Prometheus, we have designed a custom monitoring class, namely `NEPHELE monitor`, which provides an interface to collect application-specific metrics, e.g. request rate and latency, and expose them to Prometheus. The collected metrics are scraped, enabling real-time monitoring of application performance. This setup supports scaling decisions and performance optimizations based on run-time data from our applications. This combination of real-time monitoring, alerting, and extended visibility provides complete observability of each cluster.

Application Graph Scheduling: Application graph scheduling within NEPHELE's multi-cloud environment is driven by Karmada's policy-based orchestration. This approach distributes workloads across clusters based on predefined scheduling policies, optimizing resource allocation across the clusters. The internal orchestration mechanism of each local cluster is based on Kubernetes [27]. Each cluster contains a set of *private services*, which are deployed only on the corresponding cluster and a set of *exposed services*, which can be shared among specified clusters. The service placement relies on the default Kubernetes scheduling policy and the workloads are placed on available nodes based on resource availability.

Event-driven Autoscaling: Furthermore, we enable Karmada Federated Horizontal Pod Autoscaler (FHPA) [28] to automate the application scaling across multiple clusters. Scaling actions are performed by event-driven autoscaling, which is triggered by alerts from the alert manager and relies on the multi-cloud metrics collected. The FHPA setup supports scaling based on internal and external metrics, dynamically adjusting application replicas to optimize resource usage. This setup allows applications to automatically scale up or down based on demand, enabling efficient resource allocation across clusters. Last, dynamic application deployment is also enabled, allowing the deployment of specific applications once alerts are triggered, therefore leading to optimized resource utilization.

IV. Experimental Evaluation and Future Challenges

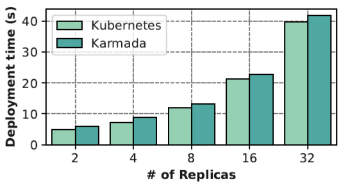
A. Experimental Setup

Our experiments were conducted on a high-end server consisting of two 20-core Intel Xeon Gold 5218R CPUs@2.10 GHz, with 4x32GB DDR4 DIMMs. Our setup includes three single-node clusters, i.e. one host cluster and two workers, hosted as virtual machines (VMs), provisioned with 6 vCPUs and 30GB of RAM each, ensuring consistent resource allocation across clusters. We utilize the Kubernetes v1.31 and Karmada v1.10.0. Regarding the monitoring tools, our experiments were conducted over Thanos v0.35.1 and Prometheus v2.53.0.

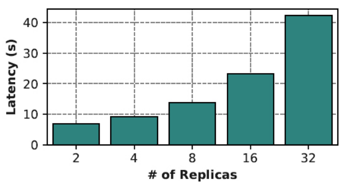
To thoroughly evaluate NEPHELE's infrastructure, we deployed general-purpose containerized workloads across the clusters, simulating a range of service demands typical of cloud and edge environments. We evaluate the implemented architecture over its ability to provide a lightweight and scalable solution for application deployment in multi-cloud infrastructure. Specifically, we analyze the core components introduced in Sec. III across four dimensions: (i) deployment and migration, (ii) autoscaling, (iii) communication and networking, and (iv) monitoring overhead, as detailed in Sec. IV-B.

B. Experimental Evaluation

Application Deployment & Migration Analysis: First, we evaluate our multi-cloud infrastructure regarding its effectiveness on deploying services, as illustrated in Fig. 3. Specifically,



a: Service deployment overhead of Kubernetes and Karmada framework for different numbers of replicas.



b: Service cluster migration overhead over Karmada for different numbers of replicas.

Fig. 3: Comparison of service deployment and migration overhead for Kubernetes and Karmada frameworks for varying number of replicas.

in Fig. 3a, we analyze the service deployment time (Y axis) against a discrete number of replicas (X axis) for deployments made through Kubernetes and Karmada, respectively. The experiments report the average value of 10 experiments deployed on our system. Our findings indicate that service deployment via Karmada incurs an average overhead of 1.53 seconds, representing an average 12.76% increase compared to Kubernetes. Notably, this overhead remains consistent regardless of the number of replicas deployed. Therefore, our designed architecture demonstrates scalability across varying numbers of replicas without encountering bottlenecks during service deployment. The service deployment overhead can be a priori known, thus enabling the prediction of scheduling overhead in the future. Furthermore, we investigate the ability of our architecture to perform effective service migration across different clusters. Fig. 3b illustrates the migration latency of varying numbers of replicas. We observe that the overhead latency is proportional to the number of replicas migrated, thus, in correlation with deployment overhead, we can co-explore the service deployment and migration run-time overhead.

Federated Auto-Scaling Analysis: The efficiency of multi-cloud infrastructures depends heavily on the ability of integrated autoscaling mechanisms to dynamically adjust resources, scaling up during periods of high demand to maintain optimal performance. Figure 4 illustrates the average time required to scale up from 1 to 10 replicas using Karmada and Kubernetes, respectively, highlighting each system's responsiveness under increasing load conditions. Our results indicate that scaling time is consistent across the two frameworks with Kubernetes demonstrating faster response, requiring 6.75% less time to scale up compared to Karmada on average, showcasing its superior responsiveness in handling load surges and delivering improved efficiency in dynamic environments.

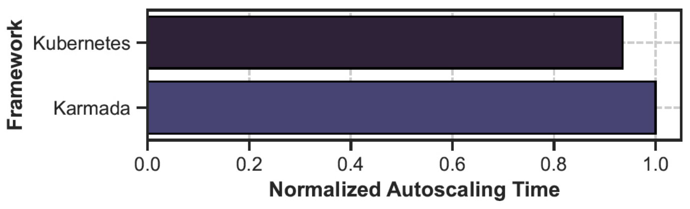


Fig. 4: Average scaling up and scaling down time through multi-Cloud Federated HPA.

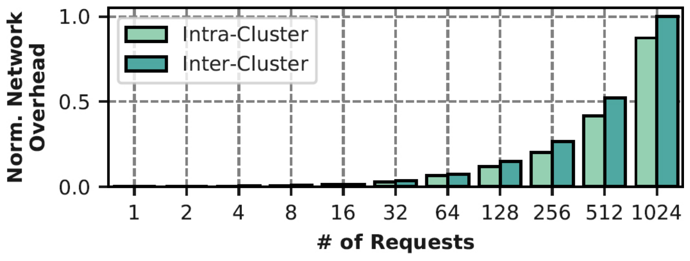


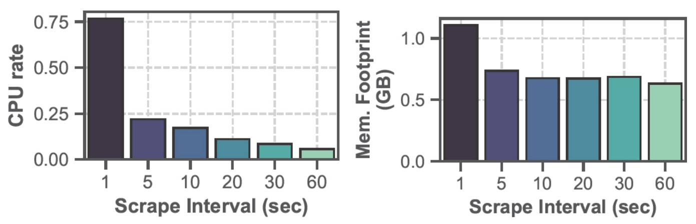
Fig. 5: Normalized inter- and intra-cluster communication overhead for different number of parallel requests

Network Overhead Analysis: In addition to examining deployment and migration overheads, we investigate both intra-cluster and inter-cluster network overhead, as illustrated in Fig. 5. In this figure, the X-axis represents the number of parallel requests, while the Y-axis shows the normalized network overhead. Our results indicate that inter-cluster data exchange, managed through the Submariner framework, incurs an average increase of 16.63% in communication time compared to intra-cluster exchanges. This guides us in evaluating trade-offs between service distribution and latency, enabling further improvements in NEPHELE's scheduling scalability and responsiveness across diverse network conditions. These findings open the path for NEPHELE towards achieving network and compute synergy, where intelligent workload allocation can optimize both data transfer efficiency and computational resource usage across the multi-cloud environment.

Run-time Monitoring Analysis: Last, we examine how Prometheus monitoring affects resource utilization by analyzing the impact of different `scrape interval` settings, which determine the frequency at which Prometheus retrieves metrics from monitored applications. Figure 6 shows the effect of varying `scrape interval` values on CPU (Fig. 6a) and memory usage (Fig. 6b) in the Prometheus server pod. We measure the average CPU utilization rate and memory usage as Prometheus performs continuous scraping. In Fig. 6a, we observe a steady decrease in CPU utilization as the `scrape interval` rises. For instance, raising `scrape interval` to 5 seconds, we get 71.5% less CPU rate, compared to 1 second. For memory consumption, as shown in Fig. 6b, we observe a reduction of 38.77% when the scrape interval is set from 1 to 10 seconds, after which memory usage levels off, reaching a plateau at higher intervals. This allows us to balance resource efficiency without significant loss in monitoring responsiveness.

C. Open Challenges by the end of NEPHELE

NEPHELE envisions to address several orchestration challenges until the end of the project. Specifically, NEPHELE aims to provide unified resource management by abstracting the heterogeneity of resources to provide seamless lifecycle management of applications. Additionally, NEPHELE plans to develop robust load-balancing solutions to enhance the efficiency of Edge and Cloud clusters, working toward sustainable and resilient infrastructures that can adapt to dynamic



a: CPU Rate b: Memory Footprint

Fig. 6: Prometheus average CPU utilization rate and memory usage for varying scrape intervals.

workloads. While rather challenging, NEPHELE will also assist in paving the way to achieve the confluence between network and compute resources, by designing common optimization techniques. Finally, adopting a business-oriented perspective, NEPHELE will incorporate technologies to accommodate diverse infrastructure and application stakeholders, enhancing usability, transparency, and strategic alignment with industry requirements. This collaborative approach ensures that the platform not only meets technical benchmarks but also aligns with real-world application needs.

V. Conclusion

In this paper, we present NEPHELE, a project that aims to streamline orchestration of applications deployed across a diverse range of computing resources, spanning multi-cloud environments to the far-edge, and extending to IoT systems. We introduce, examine, and evaluate the multi-cloud infrastructure designed to meet the project's major objectives and challenges. Through our experiments, we demonstrate that NEPHELE's infrastructure meets the demands of high-performance applications, ensuring minimal overhead in deployment, migration, and network latency. Future work, towards NEPHELE's final steps, will focus on expanding this well-established infrastructure through the integration of intelligent orchestration mechanisms and deployment of use-cases.

References

[1] P. Gkonis, A. Giannopoulos, P. Trakadas, X. Masip-Bruin, and F. D'Andria, "A survey on iot-edge-cloud continuum systems: status, challenges, use cases, and open issues," *Future Internet*, vol. 15, no. 12, p. 383, 2023.

[2] Z. He, Y. Sun, B. Wang, S. Li, and B. Zhang, "Cpu-gpu heterogeneous computation offloading and resource allocation scheme for industrial internet of things," *IEEE Internet of Things Journal*, 2023.

[3] M. Goudarzi, M. Palaniswami, and R. Buyya, "Scheduling iot applications in edge and fog computing environments: A taxonomy and future directions," *ACM Computing Surveys*, vol. 55, no. 7, pp. 1–41, 2022.

[4] A. Ferikoglou, P. Chrysomeris, A. Tzenetopoulos, M. Katsaragakis, D. Masouros, and D. Soudris, "Iris: interference and resource aware predictive orchestration for ml inference serving," in *2023 IEEE 16th International Conference on Cloud Computing (CLOUD)*, pp. 1–12, IEEE, 2023.

[5] M. Aledhari, R. Razzak, B. Qolomany, A. Al-Fuqaha, and F. Saeed, "Biomedical iot: enabling technologies, architectural elements, challenges, and future directions," *IEEE Access*, vol. 10, pp. 31306–31339, 2022.

[6] A. Karteris, M. Katsaragakis, D. Masouros, and D. Soudris, "Sgrm: stackelberg game-based resource management for edge computing systems," in *2022 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 1203–1208, IEEE, 2022.

[7] A. K. Kakolyris, M. Katsaragakis, D. Masouros, and D. Soudris, "Road-runner: Collaborative dnn partitioning and offloading on heterogeneous edge systems," in *2023 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 1–6, IEEE, 2023.

[8] D. Petcu, "Multi-cloud: expectations and current approaches," in *Proceedings of the 2013 international workshop on Multi-cloud applications and federated clouds*, pp. 1–6, 2013.

[9] Liqo, "Liqo project: Kubernetes native multi-cluster service mesh." https://liqo.io. Accessed: 2024-10-30.

[10] S. Achar, "Cloud computing security for multi-cloud service providers: Controls and techniques in our modern threat landscape," *International Journal of Computer and Systems Engineering*, vol. 16, no. 9, pp. 379–384, 2022.

[11] A. Gulli, *Google Anthos in Action: Manage Hybrid and Multi-cloud Kubernetes Clusters*. Simon and Schuster, 2023.

[12] Microsoft Azure, "Azure arc: Extend azure services and management to any infrastructure," 2024. Last accessed on 2024-10-30.

[13] NEPHELE Project, "Nephele horizon europe project: A lightweight software stack and synergetic meta-orchestration framework for the next generation compute continuum." https://nephele-project.eu/. Last accessed on 2024-06-05.

[14] A. Arteaga, N. Filinis, L. Fu, C. Habib, L. Militano, D. Spatharakis, A. Zafeiropoulos, T. M. Bohnert, N. Mitton, and S. Papavassiliou, "Virtual objects for robots and sensor nodes in distributed applications over the cloud continuum," in *International Workshop on Distributed Intelligent Systems (DistInSys)*, 2024.

[15] NEPHELE Open Calls, "NEPHELE 2nd Open Call: Calling SMEs and Mid-Caps." https://nephele-project.eu/news/nephele-2nd-open-call-calling-smes-and-mid-caps, 2024. Accessed: 2024-11-04.

[16] N. Filinis, I. Dimolitsas, D. Spatharakis, E. Fotopoulou, I. Tzanettis, C. Vassilakis, A. Zafeiropoulos, and S. Papavassiliou, "A synergetic meta-orchestration framework for distributed application deployments in the computing continuum," in *Proceedings of the 1st International Workshop on MetaOS for the Cloud-Edge-IoT Continuum*, MECC '24, (New York, NY, USA), p. 21–27, Association for Computing Machinery, 2024.

[17] D. Spatharakis, I. Dimolitsas, G. Genovese, *et al.*, "A lightweight software stack for iot interoperability within the computing continuum," in *2023 19th International Conference on Distributed Computing in Smart Systems and the Internet of Things (DCOSS-IoT)*, pp. 715–722, IEEE, 2023.

[18] D. A. Shafiq, N. Jhanjhi, and A. Abdullah, "Load balancing techniques in cloud computing environment: A review," *Journal of King Saud University-Computer and Information Sciences*, vol. 34, no. 7, pp. 3910–3933, 2022.

[19] S. Tang, Y. Yu, H. Wang, G. Wang, W. Chen, Z. Xu, S. Guo, and W. Gao, "A survey on scheduling techniques in computing and network convergence," *IEEE Communications Surveys & Tutorials*, 2023.

[20] A. Ullah, T. Kiss, J. Kovács, F. Tusa, J. Deslauriers, H. Dagdeviren, R. Arjun, and H. Hamzeh, "Orchestration in the cloud-to-things compute continuum: taxonomy, survey and future directions," *Journal of Cloud Computing*, vol. 12, no. 1, p. 135, 2023.

[21] N. Filinis, I. Tzanettis, D. Spatharakis, E. Fotopoulou, I. Dimolitsas, A. Zafeiropoulos, C. Vassilakis, and S. Papavassiliou, "Intent-driven orchestration of serverless applications in the computing continuum," *Future Generation Computer Systems*, vol. 154, pp. 72–86, 2024.

[22] Karmada Project, "Karmada: Open, multi-cloud kubernetes orchestration system." https://karmada.io, 2024. Accessed: 2024-10-28.

[23] Kubernetes Project, "Kubernetes: Production-grade container orchestration." https://kubernetes.io, 2024. Accessed: 2024-10-28.

[24] Submariner Project, "Submariner: Cross-cluster and multi-cloud networking for kubernetes." https://submariner.io, 2024. Accessed: 2024-10-28.

[25] Prometheus Project, "Prometheus: Monitoring system and time series database." https://prometheus.io/docs/introduction/overview/, 2024. Accessed: 2024-10-28.

[26] Thanos Project, "Thanos: Highly available prometheus setup with long term storage capabilities." https://thanos.io, 2024. Accessed: 2024-10-28.

[27] C. Carrión, "Kubernetes scheduling: Taxonomy, ongoing issues and challenges," *ACM Computing Surveys*, vol. 55, no. 7, pp. 1–37, 2022.

[28] Karmada FPHA, "Karmada federated horizontal pod autoscaler." https://karmada.io/docs/userguide/autoscaling/federatedhpa/, 2024. Accessed: 2024-10-28.