A Proposal for a Continuum-aware Programming Model: From Workflows to Services Autonomously Interacting in the Compute Continuum

***Abstract*—This paper proposes a continuum-aware programming model enabling the execution of application workflows across the compute continuum: cloud, fog and edge resources. It simplifies the management of heterogeneous nodes while alleviating the burden of programmers and unleashing innovation. This model optimizes the continuum through advanced development experiences by transforming workflows into autonomous service collaborations. It reduces complexity in positioning/interconnecting services across the continuum. A metamodel introduces high-level workflow descriptions as service networks with defined contracts and quality of service, thus enabling the deployment/management of workflows as first-class entities. It also provides automation based on policies, monitoring and heuristics. Tailored mechanisms orchestrate/manage services across the continuum, optimizing performance, cost, data protection and sustainability while managing risks. This model facilitates incremental development with visibility of design impacts and seamless evolution of applications and infrastructures. In this work, we explore this new computing paradigm showing how it can trigger the development of a new generation of tools to support the compute continuum progress.**

***Index Terms*—Workflows, Compute continuum, Programming models**

I. Introduction

The computing world is undergoing rapid and continuous evolution as new technologies and approaches emerge to cater to the increasing demand for sophisticated and intelligent services that can improve user experience. One such approach is the cognitive cloud [1], which enhances the capabilities of cloud-based applications and services by incorporating cognitive technologies such as machine learning and reasoning [2], [3]. According to a survey by IBM Cognitive, companies that embraced cognitive computing capabilities have already noticed considerable investment returns [4]. Incorporating cognitive technologies throughout the continuum enables more intelligent distributed services and more efficient use of cloud resources [2], [5]–[7]. Continuum-aware programming models and design patterns for seamless ultra-scalable processing of hyper-distributed applications in the continuum are a key requirement to enable the development of more intelligent and sophisticated services and applications addressing the challenges of managing, scaling and making resilient complex, distributed systems. This aligns with Gartner's prediction that *«organisations will need to invest in tools and technologies that support distributed infrastructure»* to keep up with the demands of modern applications [2]. Distributed applications have become increasingly important in our daily lives, with a growing number requiring flexible and dynamic use of compute and network resources [5].

Cloud technologies played a crucial role in providing the infrastructure for these dynamic applications [8]. However, the only viable approach to achieving ultra-scalability to support modern applications is decentralising the infrastructure, bringing resources near users and data [9]. The rise of fog and edge infrastructures has been observed, with edges being "walking-sized" Cloud data centres pervasively distributed in the environment, while fog resources are placed along the network paths [10]. Edge and fog resources tend to be mainly exploited along the vertical path, starting from the cloud and reaching the edge through fog devices. This limits the benefits of their exploitation, especially for applications that need ultra-scalable infrastructures [11].

A fully decentralised environment can scale more effectively, improve reliability, enhance security, provide greater flexibility, and reduce costs compared to a traditional cloud or edge environment [12]. However, the complexity of managing such a distributed heterogeneous environment prevents a full point-to-point, horizontal interaction between edge data centres, posing some key challenges that require specific enabling technologies to be provided [13]. Developing an intuitive programming model is imperative to liberate developers and catalyze impactful innovation in this domain. A model that simplifies developing sophisticated hyper-distributed applications optimized to leverage all capabilities across this continuum landscape would unlock fundamentally transformative possibilities.

This paper presents such a proposal: a Continuum-aware programming model. Built upon-describing workflows and their seamless metamorphosis into autonomous collaborations between services, this model breathes simplicity and potency into developing continuum-spanning intelligent solutions. By alleviating the burden of complexity, it equips developers with the agility to expand the horizons of progress through continuum-aware applications. This model stands to revolutionize how continuum resources across cloud, fog and edge are combined and optimized to achieve breakthroughs. It transforms the development experience from fraught with complexity to flowing with possibility. Developers could then focus on crafting innovative services and solutions rather than dealing with integration challenges. Overall, this proposal aims to catalyze a paradigm shift enabling continuum-optimized progress through an intuitive development experience. The

Continuum-aware programming model promises to future-proof both developer productivity and the application potential in this fast-evolving landscape.

The remainder of this paper is organised as follows. Section II frames our proposed perspective in the proper context, highlighting the need for a programming model like the one we envision. Section III provides a glimpse of the key aspects and objectives for the workflow-based continuum-aware programming model. Section IV discusses the pillars and structure of the envisioned programming model. Finally, Section V foresees future work that can be developed from this position paper.

II. Background

The complexity that derives from the distributed and heterogeneous nature of the compute continuum calls for software design approaches that can ease the task of implementing applications efficiently running in the continuum, able to use different distributed resources seamlessly. The challenge is even tougher when applications have a hyper-distributed nature.

A widely adopted model for implementing large-scale, hyper-distributed data-intensive applications relies on workflows. This is due to their ability to provide structure and organisation to the flow of execution of different tasks or operations. Workflows also enable developers to separate the application's requirements from its implementation, resulting in more straightforward code development. Additionally, workflows can coordinate data processing tasks for data-intensive applications, ensuring that the right data is available at the right time and place, regardless of where it is stored. However, orchestrating distributed workflows in the compute continuum is a complex task. Execution locations can be heterogeneous, exposing different methods and protocols for authentication, communication, resource allocation and job execution. Plus, they can be independent of each other, meaning that direct communications and data transfers among them may not be allowed.

Hybrid workflows, which span multiple heterogeneous and independent execution locations, have proved successful in mixed Cloud+HPC environments [14]. The StreamFlow framework [15] supports hybrid workflows on multi-container environments and is fully compliant with the Common Workflow Language (CWL) open standard [16], a declarative and vendor-agnostic language to model scientific workflows following a dataflow approach [17].

Each step of a CWL workflow can be mapped onto a different execution environment, making StreamFlow suitable for generating deployments driving the creation of application instances on the resources of the cognitive continuum. Once deployed, application instances need to be properly monitored and managed.

Concerning this need for monitoring and management it is worth mentioning the sidecar pattern [18]. It is a well-established software design pattern in which a separate component is deployed alongside a main application to augment or enhance its functionality. The sidecar mediates communication between the main application and other systems providing monitoring, logging, or security services [19]. The sidecar pattern enables a modular and scalable architecture, allowing changes or updates to the sidecar component to be made independently from the main application and for multiple sidecars to provide different functionalities to the same application. A key aspect of the sidecar pattern is its coupling with the service enabling a decentralised approach to application monitoring and support. Decentralised coordination and management is a well-established solution in many domains [20]–[22]. It is a particularly suitable approach [23] to provide a highly scalable, adaptable, and efficient approach to the fully decentralised management of applications in the computing continuum.

A. The need for a continuum-aware programming model

Our proposal is to extend hybrid workflows to target the compute continuum, enabling the deployment of applications across diverse data centres and heterogeneous edge resources while maintaining high performance and scalability. The envisioned approach transforms workflows into a collection of *huggers*.

A Hugger is an improved sidecar pattern, optimised for the decentralised, efficient and seamless exploitation of heterogeneous resources. Each hugger instance acts as a manager and enabler for the application, providing various features such as context monitoring, migration, replication, adaptation into different versions, and undeployment. To this end, the hugger relies on performance and cost models. Automated reasoning [24] drives informed decisions empowered with deep learning technologies. These technologies optimise the performance and behaviour of the application instance by considering the workload, performances, resources, network conditions, etc. In complex scenarios, huggers can coordinate through swarm intelligence to optimise the performance of applications. By combining declarative reasoners, deep learning, and swarm intelligence [25], the hugger pattern can contribute to the efficient management of distributed applications by responding to changing conditions in near-real-time.

A detailed presentation of nature and all the features that characterise the hugger pattern is beyond the scope of this paper, which we limit to describing only what is relevant to the approach proposed in this paper.

III. A Continuum-aware programming model for hyper-distributed applications

The development of a programming model and the relative runtime that enables the design and execution of hyper-distributed applications on the compute continuum is a very challenging task. Such a programming model should allow for developing applications that efficiently exploit various and heterogeneous computing resources, including IoT devices, far-edge constrained devices, federated fog/edge computing nodes, and cloud computing centres, while optimising resource utilisation and enhancing the quality of service.

In the following, we briefly outline the main objectives of the continuum-aware programming model we envision and its key-feature enablers.

A. Objectives

In our vision, the new programming model and its runtime system should:

- seamlessly handle the heterogeneity of computing resources, architectures, and data accessibility
- ease hyper-distributed applications to scale efficiently and handle the increasing number of tasks with minimal impact on performance while avoiding under and over-provisioning of resources
- provide a high level of fault tolerance and resiliency to enable applications to continue functioning despite failures. Tasks will be automatically rescheduled or migrated in case of failures or resource constraints
- enable data processing close to its source or destination, thus reducing data transfer latencies and network overhead
- optimise utilisation of computing resources by dynamically provisioning and releasing resources according to the actual workload
- offer highly scalable processing of messages and events through the Hugger pattern.

B. Enablers

At the cornerstone of our envisioned continuum-aware programming model we pose the actor model of computation [26], [27]. It provides the foundation for the hyper-scalable and hyper-distributed ecosystem that characterise the compute continuum and we aim to target with our proposed programming model. With the adoption of the actor model, we aim at following a unified, flexible, and well-defined interaction schema and abstraction. The actor model proved to be effective for the design of platform components and active entities in distributed computing environments where a scalable communication and processing is needed [28]–[30]. This model provides foundations for driving the coordination between distributed processes, and to deal with unreliable connectivity, distribution and decentralisation of intelligent solutions.

The envisioned continuum-aware programming model will expose to developers a workflow-based abstraction and will allow to execute applications in multi-services environments, support concurrent execution of multiple, actor-based, communicating active entities, and allow for hybrid workflow executions on top of continuum resources. Each entity will be an instance of the *Hugger* pattern, representing a novel application runtime management approach, taking the sidecar pattern as a baseline technology and building over it to enable application migration, replication, and management. Each Hugger will be coupled with one or more application instances that actively manage. The active behavior of Huggers is based on actors. Huggers will interact and communicate efficiently to

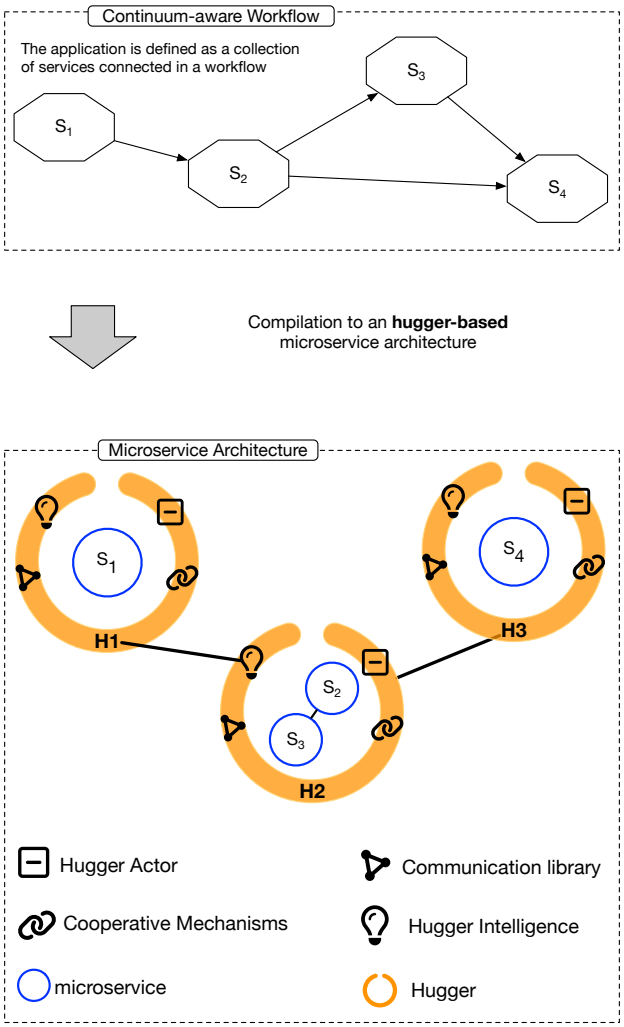

Fig. 1. From workflows to Huggers

realise decentralised application management whose directives will be expressed using a high-level declarative approach.

Such technologies will provide actors with the necessary features in terms of scalability and efficiency.

IV. The Continuum-aware Programming Model: Pillars and Structure

The Continuum-aware Programming Model must be straightforward for the application developer and able to manage the complecity of the continuum-based platform. On the one side, the goal is to ease the development of novel applications by allowing developers to define their applications for the continuum while remaining agnostic about the structure of the infrastructure. On the other side, the aim is for a well-designed Programming Model that helps in optimising the usage of the continuum by allowing orchestration operations to benefit the exploitation of the resources and the execution of the applications.

Our envisioned continuum-aware programming model will offer developers a streamlined workflow that enables them to

adapt and write applications and specify requirements for each component without dealing with all the complexities associated with the continuum as heterogeneity, distribution, and dynamicity. This programming model will allow developers to focus on writing applications, leaving the management of the underlying infrastructure to the runtime of the programming model.

As is exemplified in Figure 1, a high-level workflow-based representation is translated into a network of (possibly composed) microservices supported by the hugger pattern. The hugger pattern will associate each application component with an intelligent and active entity, the design of which is based on an actor that is responsible for managing that component. The set of hugger patterns associated with application entities will employ performance cost models, ML and AI techniques for managing said entities based on information close to the application semantics and leverage swarm techniques to approach suitable collective targets.

The hugger pattern will be able to perform complex actions that allow it to monitor, migrate, and replicate application instances, making it an essential component of the runtime of the continuum-aware programming model. As shown in Figure 1, a workflow composed of four services is "compiled" into instances of the hugger pattern. Each hugger provides a number of mechanisms to support the management of the micro-services it "hugs" and the interaction with the infrastructure easing the deployment operations. The fundamental set of entities that realise a Hugger comprises: the hugger actor, the communication library, the cooperative mechanisms and the hugger intelligence.

A. Hugger Actor

The artefact that encapsulates and provides the behaviour of the hugger pattern is represented by an active entity realised accordingly with the actor model of computation. As by the actor model, it consists of state, behaviour, mailbox, address, and communication. The internal state of the Hugger includes information about the application instances it manages, their current state, and other relevant metadata. The behaviour is defined by a set of actions that it can perform, such as monitoring the state of the application instances, migrating them to another node, or replicating them. The mailbox is a message queue that stores incoming messages from other actors. The address is a unique identifier that allows other actors to send messages to the Hugger. The communication mechanism allows the Hugger to send and receive messages from other actors. The communication mechanism can be implemented using various protocols and strategies.

B. Communication Library

The heterogeneity of resources, network protocols, and infrastructures hinders the realization of pervasive, seamless access, and hyper-distributed applications. We plan to adopt software infrastructures that dynamically integrate multiple protocols within the Hugger actor by leveraging multi-transport communication libraries enabling Huggers to cooperate with a single interface regardless of their position in the compute continuum infrastructure. Communication needs to be working also when huggers are deployed across different administrative domains or networks with restricted connectivity. For example, the MTCL library[1], allows peers to transparently leverage the best communication protocol based on the type, location, and available network interfaces. The library offers a connection-like interface and transparently maps communications to specific protocols that best leverage the resources of the surrounding environment, e.g., MPI/UCX if some peers are deployed on HPC infrastructures or MQTT/TCP if deployed on edge nodes.

C. Cooperative Mechanisms

Hugger instances are envisioned to cooperate with each other in a fully decentralized scenario. The cooperation will involve ad-hoc mechanisms that are designed to work seamlessly and efficiently, without introducing centralization that could lead to bottlenecks and single points of failure. This decentralized approach ensures that the continuum-aware programming model can handle the heterogeneity, distribution, and dynamicity of the computing continuum while maintaining a high level of fault tolerance and scalability. By leveraging the power of swarm techniques, the Hugger instances can work together to achieve common goals, such as load balancing or fault tolerance, without relying on a central control point. Strategies for distributed consensus and conflict resolution will be also provided. Overall, the cooperation between Hugger instances is a key aspect of the continuum-aware programming model that enables efficient and effective management of hyper-distributed applications in the continuum.

D. Hugger Intelligence

The Hugger pattern is designed to create intelligent agents that can optimize application management through interaction with the environment and cooperation with their peers. The intelligence of Huggers is based on both model-driven knowledge and data-driven knowledge. The model-driven knowledge includes cost models, performance models, and other models that can be used to predict the behaviour of the application instances and the infrastructure. The data-driven knowledge, on the other hand, is obtained by leveraging machine learning and artificial intelligence techniques on the data that is collected from the application instances and the infrastructure. To obtain data-driven knowledge, different approaches and solutions will be leveraged. The adoption of continual reinforcement learning is envisioned as an effective solution for scenarios involving distributed application management. Continual reinforcement learning is a type of reinforcement learning that allows agents to learn continuously from their experiences over time. By leveraging this approach, the Huggers can learn how to optimize the management of the application instances and the infrastructure based on the feedback they receive from the environment. Additionally, alternative approaches that are suitable for the complex environment of the computing continuum,

[1]MTCL library home: https://github.com/ParaGroup/MTCL

such as randomized recurrent neural networks and reservoir computing, will also be explored. The Huggers are intended to be cooperating with each other in a fully decentralized scenario. The cooperation will involve ad-hoc mechanisms that are designed to work seamlessly and efficiently, without introducing centralization that could lead to bottlenecks and single points of failure. This decentralized approach ensures that the continuum-aware programming model can handle the heterogeneity, distribution, and dynamicity of the computing continuum while maintaining a high level of fault tolerance and scalability. By leveraging the power of swarm techniques, the Huggers can work together to achieve common goals, such as load balancing or fault tolerance, without relying on a central control point. The intelligence of the Huggers is a crucial aspect of the continuum-aware programming model that enables the efficient utilization of computing resources, fault tolerance, and scalability while maintaining a high level of quality of service.

In this perspective, The Hugger Pattern optimizes application execution through intelligent, decentralized management of components. Beyond mediating interactions, Huggers leverage application semantics and infrastructure data to enable sophisticated yet agile management strategies.

Rather than static policies, Huggers employ adaptive techniques - distributed intelligence, custom performance models, reinforcement learning - to maximize QoS given dynamic conditions. By balancing application needs with available resources, Huggers can fulfil key requirements like elastic scalability, fault tolerance, and load balancing.

V. Future Work

This position paper presented a proposal for an approach aimed at managing application instances in the computing continuum. Applications are expressed as workflows and eventually translated into a group of smart collaborating entities that can be placed across the large heterogeneous set of resources that the continuum encompasses.

There are several areas for future research and development that could enhance the effectiveness and efficiency of the approach. One potential area for future work is the alignment of the proposed approach with the existing platforms supporting the cross-domain deployment of applications in a cloud/edge environment. This "integration" can enable to definition of how to ease the deployment and management of applications leveraging the cloud/edge providers' infrastructure and services to enable efficient utilization of computing resources and provide fault tolerance and scalability.

Another area for future work is a clear definition of the Hugger intelligence. This optimization can be achieved by leveraging different machine learning and artificial intelligence techniques, such as continual reinforcement learning, randomized recurrent neural networks, and reservoir computing. The different solutions can be eventually assessed to define guidelines on what to use and under what conditions. The effectiveness and efficiency of the approach can be further evaluated in different scenarios and use cases, such as those involving different levels of heterogeneity, distribution, and dynamicity of the computing continuum, and different types of applications, such as data-intensive applications, compute-intensive applications, and real-time applications.

The approach has also to be integrated with security mechanisms to ensure the security and privacy of the application instances and infrastructure. Different security mechanisms, such as encryption, access control, and intrusion detection, can be leveraged to provide a comprehensive security solution. The communication mechanisms used by the Huggers and the workflow definition and compilation is the element that enables efficient and reliable communication between the Huggers and the infrastructure. Different communication protocols and transport mechanisms can be leveraged and tested to define guidelines and insights on how to achieve efficient communication in different scenarios.

Finally, the workflow definition and compilation can be enhanced by developing algorithms for automatically composing workflows from high-level specifications. These algorithms can leverage machine learning and artificial intelligence techniques to generate optimal workflows that meet the application requirements and the constraints of the computing continuum.

References

[1] G. Fowler, "Understanding cognitive cloud computing and its potential impact on business," https://www.forbes.com/sites/forbesbusinessdevelopmentcouncil/2021/02/24/understanding-cognitive-cloud-computing-and-its-potential-impact-on-business/?sh=5ea170f9265c, 2021.

[2] K. Panetta, "Gartner top strategic technology trends for 2021," https://www.gartner.com/smarterwithgartner/gartner-top-strategic-technology-trends-for-2021, 2021.

[3] IBM, "From knowledge graphs to cognitive computing," https://www.ibm.com/blogs/research/2016/01/from-knowledge-graphs-to-cognitive-computing/, 2016.

[4] ——, "The cognitive advantage global market report," https://www.ibm.com/watson/advantage-reports/market-report.html, 2021.

[5] L. Perri, "Gartner hype cycle for emerging technologies," https://www.gartner.com/en/articles/what-s-new-in-the-2022-gartner-hype-cycle-for-emerging-technologies, 2022.

[6] M. Satyanarayanan, "The emergence of edge computing," *IEEE Computer*, vol. 50, no. 1, 2017.

[7] W. Z. K. et al, "Edge computing: A survey," *Future Generation Computer Systems*, vol. 97, 2019.

[8] M. C. et al, "A dynamic service migration mechanism in edge cognitive computing," *ACM Transactions on Internet Technology*, vol. 19, no. 2, 2019.

[9] M. M. et al, "Fog computing and the internet of things (iot): A review," in *8th IEEE International Conference on Cyber Security and Cloud Computing (CSCloud)*, 2021.

[10] K. Linghe, T. Jinlin, and J. H. et al., "Edge-computing-driven internet of things: A survey," *ACM Computing Surveys*, vol. 55, no. 8, 2022.

[11] M. C. et al., "Fog and iot: An overview of research opportunities," *IEEE Internet of Things Journal*, vol. 3, no. 6, 2016.

[12] S. P. S. et al., "Fog computing: From architecture to edge computing and big data processing," *The Journal of Supercomputing*, vol. 75, no. 4, 2019.

[13] M. et al., "A survey on mobile edge computing: The communication perspective," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, 2017.

[14] I. Colonnelli, B. Cantalupo, R. Esposito, M. Pennisi, C. Spampinato, and M. Aldinucci, "HPC application cloudification: The streamflow toolkit," in *PARMA-DITAM@HiPEAC*, 2021.

[15] I. Colonnelli, B. Cantalupo, I. Merelli, and M. Aldinucci, "Streamflow: cross-breeding cloud with hpc," *IEEE Transactions on Emerging Topics in Computing*, 2021.

[16] M. Crusoe, S. Abeln, A. Iosup, P. Amstutz, J. Chilton, N. Tijanić, H. Ménager, S. Soiland-Reyes, B. Gavrilović, C. Goble, and T. C. Community, "Methods included: standardizing computational reuse," *Communications of the ACM*, vol. 65, no. 6, 2022.

[17] E. Lee and T. Parks, "Dataflow process networks," *Proceedings of the IEEE*, vol. 83, no. 5, 1995.

[18] B. Burns and D. Oppenheimer, "Design patterns for container-based distributed systems," in *8th {USENIX} workshop on hot topics in cloud computing (HotCloud 16)*, 2016.

[19] S. Busanelli, S. Cirani, L. Melegari, M. Picone, M. Rosa, and L. Veltri, "A sidecar object for the optimized communication between edge and cloud in internet of things applications," *Future Internet*, vol. 11, no. 7, p. 145, 2019.

[20] P. Ge, F. Teng, C. Konstantinou, and S. Hu, "A resilience-oriented centralised-to-decentralised framework for networked microgrids management," *Applied Energy*, vol. 308, p. 118234, 2022.

[21] R. Casadei, D. Pianini, M. Viroli, and A. Natali, "Self-organising coordination regions: A pattern for edge computing," in *Coordination Models and Languages: 21st IFIP WG 6.1 International Conference, COORDINATION 2019, Held as Part of the 14th International Federated Conference on Distributed Computing Techniques, DisCoTec 2019, Kongens Lyngby, Denmark, June 17–21, 2019, Proceedings 21*. Springer, 2019, pp. 182–199.

[22] R. Baraglia, P. Dazzi, M. Mordacchini, L. Ricci, and L. Alessi, "Group: A gossip based building community protocol," in *Smart Spaces and Next Generation Wired/Wireless Networking*, S. Balandin, Y. Koucheryavy, and H. Hu, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 496–507.

[23] S. F. et al, "Osmotic management of distributed complex systems: a declarative decentralised approach," *Journal of Software: Evolution and Process*, 2022.

[24] I. Bratko, *Prolog Programming for Artificial Intelligence*. Addison-Wesley, 2012.

[25] S. J. V. Bhatnagar and Y. Singh, "Swarm intelligence for multi-objective optimisation in cloud computing: A survey," *Computers & Electrical Engineering*, 2018.

[26] C. Hewitt, "Actor model of computation: Scalable robust information systems," 2015.

[27] C. Hewitt, P. Bishop, and R. Steiger, "A universal modular actor formalism for artificial intelligence, ijcai'73 proceedings of the 3rd international joint conference on artificial intelligence, 20-23 august," 1973.

[28] P. Bernstein, S. Bykov, A. Geller, G. Kliot, and J. Thelin, "Orleans: Distributed virtual actors for programmability and scalability," *MSRTR2014*, vol. 41, 2014.

[29] S. N. Srirama, F. M. S. Dick, and M. Adhikari, "Akka framework based on the actor model for executing distributed fog computing applications," *Future Generation Computer Systems*, vol. 117, pp. 439–452, 2021.

[30] P. Kraft, F. Kazhamiaka, P. Bailis, and M. Zaharia, "{Data-Parallel} actors: A programming model for scalable query serving systems," in *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22)*, 2022, pp. 1059–1074.