A community initiative, anchored by

INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY
HYDERABAD

IIIT Hyderabad

TechForward

DISPATCH

DECEMBER EDITION

EMERGING SOFTWARE ARCHITECTURES
(DATASCIENCES, AI & SUSTAINABLE SYSTEMS)

Software architecture has evolved significantly over the years, from simple monolithic systems to complex, distributed, cloud-native, and AI-driven architectures. As emerging technologies like quantum computing and agentic AI gain prominence, the future will see even more transformative changes. The TechForward December edition explores how the need for sustainable, intelligent, and adaptive systems will drive the development of new architectural patterns, making it an exciting time for the software development community.

IIITH's TechForward research seminar series is an academia-industry confluence around emerging technologies. The deep insights, directional talks and industry outlooks from accomplished thought leaders at the seminar are compiled monthly in the Tech Dispatch as a ready reckoner for technology directions.

From the Chair's Desk

Regardless of whichever sector an organisation belongs to, everyone is aiming to give customers a better experience. For instance, in the case of banking, it's all about bigger, better and faster. Faster in terms of quicker transactions, better in terms of enhanced user experiences and so on. But where does all of this happen? We need to have an architecture-first view of the world and it all really starts at the drawing board where we start thinking of how systems need to be and how they need to be designed.

Modernisation is the watchword across all organisations - whether it's the financial sector, or healthcare or the retail sector, and they're all moving away from monoliths to building a microservices-based architecture working on data to provide better personalised experiences. But it's not just about splitting a monolith and having multiple microservices, you need to start thinking of the optimal size of that architecture, how the microservices will talk to each other, how the latency will be in the system, how are you looking at moving your workload to the cloud and the optimal way to go about this shift and so on. Therefore the fundamental premise for everyone who is working at building systems is architecture. However it goes beyond just software design; it's about designing data and designing on the cloud. And if it were to be sustainable then one has to look at the optimal size of microservices that you need to reduce your overall carbon footprint.

There are many aspects to designing better software and its important that we have this conversation in the form of a TechForward seminar series where there's an industry premise and an academic premise because these are ever evolving concepts.

SIRISHA VORUGANTI
CEO and MD, Llyods Technology Center India

CONTENTS OF THIS EDITION

The sixth edition of TechForward was hosted at Quorum Club, Sattva Knowledge City, Hitech City, Hyderabad

Challenges And Opportunities In Emerging Software Architectures

***Prof. Raghu Reddy** lists out the current challenges in designing software architecture, talks about IIITH's Software Engineering and Research Centre current focus and gives a sneak peek into the future. Here's a summarized version based on his talk at the TechForward seminar series.*

To better understand the importance of software architecture, let's first look at its evolution which spans across five ages. As per an article published in IEEE software, in the pre-1980s, the focus was primarily on monolithic systems. During the 1980s and 1990s, we began to explore distributed monolithic systems that offered multiple architectural views for different types of stakeholders. It was during this period that the concept of diverse architectural styles emerged, leading to the adoption of 3-tier architecture, client-server models, and more. This was also the time when consideration of how to assess the implementation of an architecture began. No longer was there just a single vendor catering to a small set of programs; instead, entire platforms had to be distributed. The complexity increased in the 1990s with the onset of the internet-connected age, where systems were always 'on' and connected. This necessitated quality considerations like scalability, performance, security, etc. It was around this time, Dewayne Perry and Alexander Wolf published a seminal paper on the foundations of software architecture. This work remains one of the most cited resources and helped to formalize software architecture as a discipline and provided a basis for future research and development in this area.

In the 2000s and 2010s, as internet-native applications (applications specifically built for the internet rather than simply hosted with its assistance) became commonplace, the focus shifted to building systems that were fast and reliable. Additionally, there was a strong intent to minimize technical debt for companies. As a result, software architecture evolved to become more agile. This evolution is ongoing due to the continuous emergence of new technologies. For instance, with the emergence of IoT, we are moving towards architecting connected devices. We have AR/VR, Edge, 4G/5G, Quantum, 3D printing, Bionics, and other emerging technologies. This necessitates a shift towards an architecture-first approach, to effectively address the challenges posed by these emerging technologies.

Current Architectural Problems

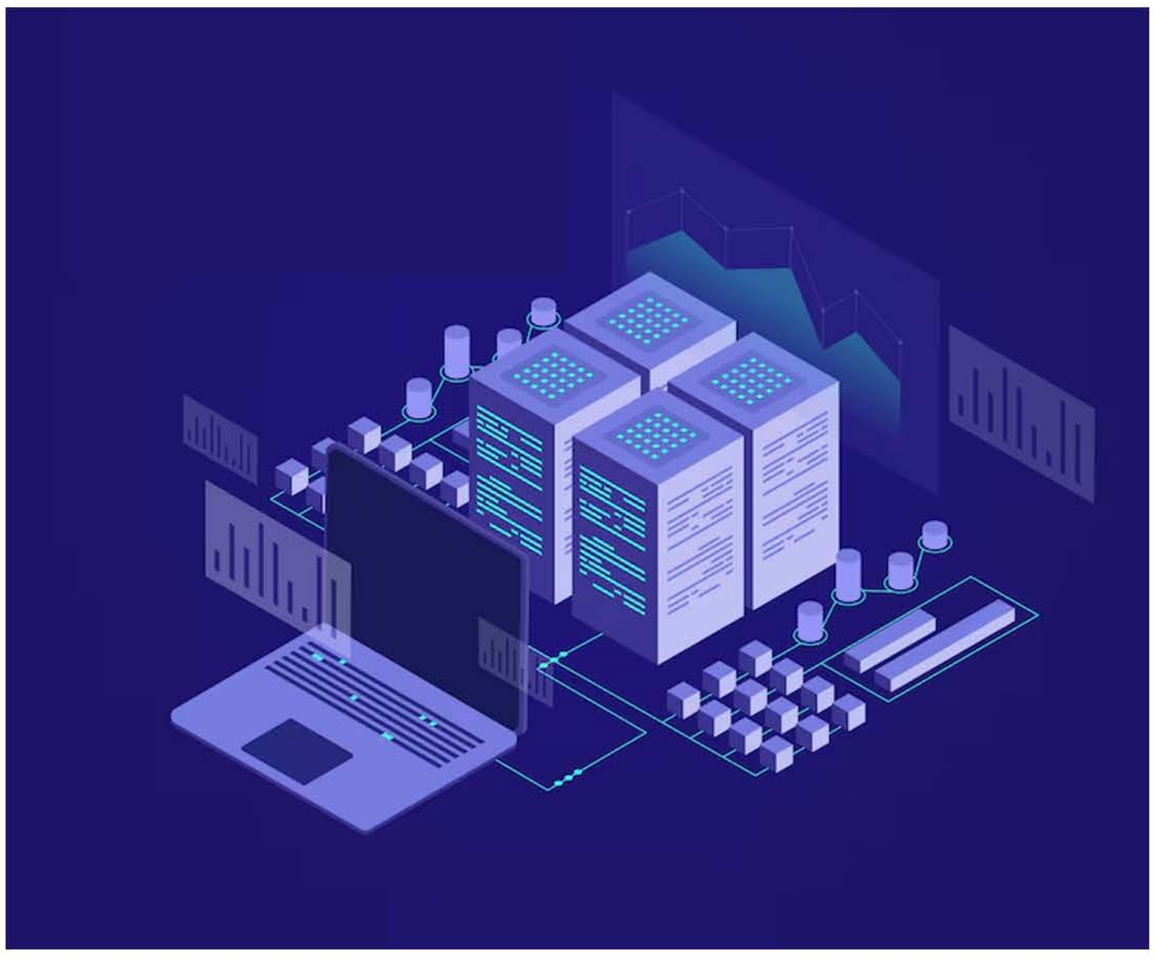
Some of the typical challenges encountered include latency, throughput, and access to information. For example, using AR/VR and IoT for digital twins requires an active feedback loop, as the analysis performed must feed back into the system. Similarly, when discussing the transition from cloud to edge computing to place data closer to the devices we use, the architecture for edge devices necessitates reduced response times. We are effectively moving towards a data-centric architecture.

To illustrate the importance of data-centric architecture, consider an online search for a passenger car that weighs less than 2000 kg and costs less than 20 lakhs. As it can be seen via a simple google search, the results conveniently ignored the weight component of my query,

focusing solely on the cost of cars that met one of my specifications but not the exact information that was requested. This highlights the need for a fundamental shift towards decoupling data from applications. In other words, code related to data should not be embedded within the application; instead, it should be easily extractable. We encounter similar situations in enterprises, where typical applications have separate data models. Enterprises may possess thousands of applications, many of which operate in silos, integrating data through data warehouses, data lakes, point-to-point interfaces, and APIs. The goal is to transition from an application-centric architecture to a data-centric architecture, where an enterprise knowledge graph (EKG) is utilized by multiple applications. This shift implies that instead of having one model per application, we will likely have one core model for the entire enterprise.

Pluses Of Data-Centric Architecture

The advantages of a data-centric architecture are numerous. One key benefit is that it can prevent data silos by aggregating fragmented data into an enterprise knowledge graph, which in turn becomes shared knowledge for the entire organization. Second, it is sensitive to data quality issues due to automated quality assessments. Additional benefits include its ability to handle unstructured data, flexibility in adapting to business changes, enhanced user interaction through dynamic query mechanisms and the use of natural language, proactive predictive capabilities, and contextual awareness that enables it to adapt to change.



Challenges

As software architects, we must however consider functionality, the quality of data, the ML model itself, and infrastructure, each of which presents its own challenges. For instance, when dealing with data, the sheer volume can be overwhelming, prompting questions about which features to focus on. This, in turn, affects the parameters involved; the more parameters there are, the larger the model being trained and built becomes. This complexity has significant implications for the architecture of the entire system, as it necessitates consideration of maintenance complexity, versioning of the models, and more.

Model Drift And Navigating It

Since data is in a constant state of flux, ML models are also continuously evolving, leading to shifts in the statistical properties of the target variables. This phenomenon, known as model drift, is something software architects need to consider as they adopt strategies to address it. Tools that can assist with managing model drift include monitoring suites like Prometheus and Grafana for real-time performance tracking, as well as automated feedback loops using Kubeflow. To understand why addressing this is crucial, especially in the age of Edge computing, let's consider a simple example: a type of lane assist system, commonly known to many as Advanced Driver-Assistance System (ADAS). As an architect, one must ponder where the intelligence that detects the lanes resides. Is it embedded in the device in the car, on the phone, or elsewhere? If the intelligence resides in a small device, the architect needs to think about model compression, how small should the model be, stability of communication - especially since it will rely on the internet, and qualities like security (to prevent hacking into the device).

This is where hybrid architecture comes into play, a mix of cloud and edge. There could be different types of architectures: for example, the system could be entirely centralized, entirely decentralized, or a combination of the two. An architect has to be able to make a decision on how to train the data, when to re-train, etc to ensure the system is performing as per expectations. However, it's been noticed in many intelligent systems that as intelligence grows, systems become increasingly uncertain. This uncertainty includes resource variability, data drift, concept drift, quality of service uncertainty, environmental uncertainty, etc. The architect's role becomes extremely important in this world where systems need to continuously adapt to these uncertainties.

Designing Self-Adaptive Systems

One of the architectural solutions gaining popularity is the concept of self-adaptive architectures. The framework associated with this is the MAPE-K framework (Monitor, Analyze, Plan, Execute, and Knowledge), which is used to develop self-adaptive systems that can dynamically respond to changes in the environment. The MAPE-K loop also facilitates real-time adjustments in model parameters or architecture based on objectives. At IIITH, SERC faculty and students have been focusing on sustainability as an objective in designing software architecture. Some private organizations have shown interest in partnering to achieve green software goals by analyzing energy usage consumption data from various devices, such as IoT devices and sensors. This collaboration aims to build models that help us understand whether consumption is at peak load or reduced load. As a result, organizations can set thresholds and decide whether to maintain those levels or adapt accordingly. Some part of the research related to this is based on a paper presented at the 21st IEEE ICSA conference held at IIIT Hyderabad in June 2024. There is a good amount of research work being done in this area. Researchers have identified some of the tactics that one can follow for sustainability of ML-enabled systems that include applying sampling techniques for data-centric systems, using quantization aware training for model training, considering federated learning for deployment and so on.

Conclusion

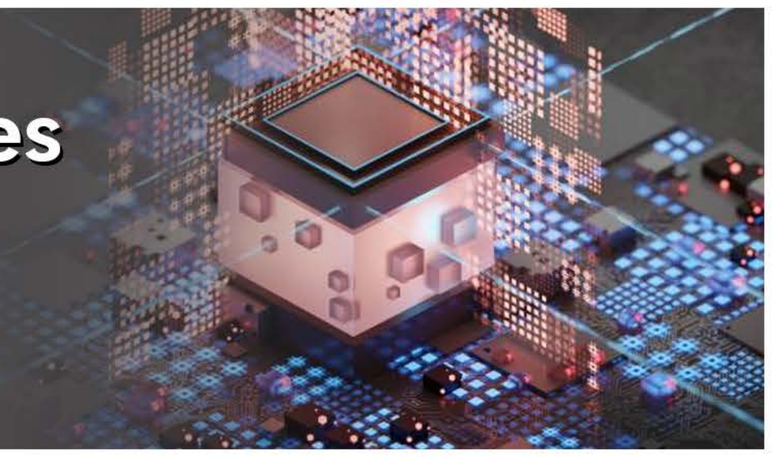
The future of software architecture in the wake of data-centricity is that we are probably going to see more of adaptable design, more operational expenditure, one that is more probabilistic, more of no code, low code platforms, more of MEC, and Cloud-Edge hybrid continuums.

PROF. RAGHU REDDY

IIIT HYDERABAD

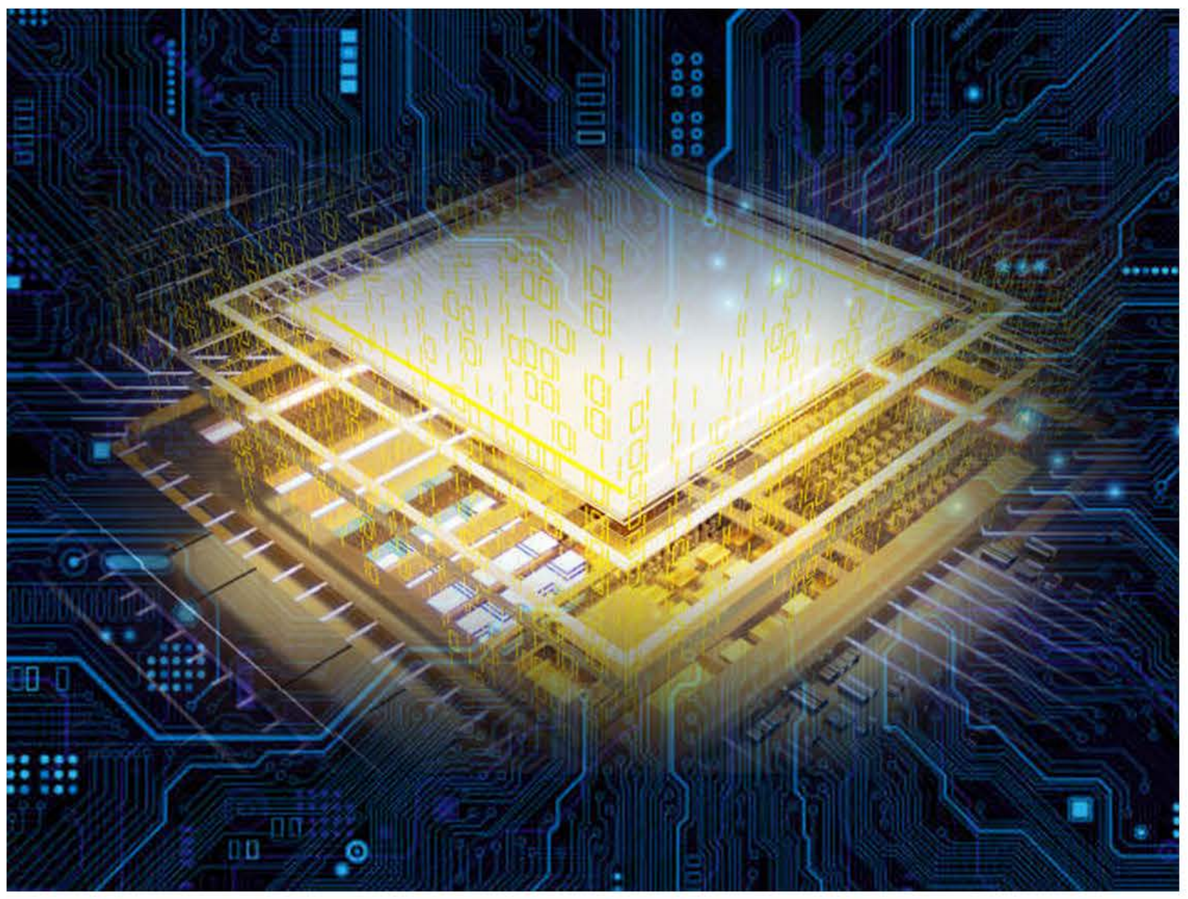
Is Associate Professor and Head, Software Engineering Research Centre, IIITH. His primary research interests are in the design and construction of complex software systems, human computer interaction and AR/VR.

Emerging Software Architectures and Design Trends - An Industry Perspective



***Ravi Kappagantu's** talk as part of the TechForward Research Seminar series shed light on where we are at in terms of designing software architecture and what to expect next. Here are the key takeaways.*

Before gazing into the crystal ball and trying to discern emerging trends in software architecture, let's take a step back and look at its evolution first. From the 1950s until the present, there has been a progression in computing paradigms, hardware capabilities, and user needs. For instance, let's go back to the 1950s which is when the first computer was developed. So it was essentially all hardware-centric. Everything had to be very specific to the processor, to the chip or transistor and so on. Later, there was an urgent need for reusability and modularity; if I wrote some logic, how could I execute it multiple times?. That led to high-level programming languages like COBOL or FORTRAN where you could have a module that could be executed multiple times. But with centralisation, the need to execute it at another place with lesser compute power arose. It led to the emergence of client servers and graphical user interfaces to an extent. Up until now the focus was on applications. By the 1980s, people realised that there was data on the one end and the ability to perform functions using it on the other. How to go about combining them was the question. This led to object-oriented paradigms. The 1990s were the era of the internet so we had a web page, web server, and an app server. As more and more internet-based applications came into the picture, the need for reusability was in focus again. In the 2000s, service-oriented architecture came into play. Those services were all monolithic. It basically meant that if I had to scale even one tiny portion of my business, I needed to scale my entire service. I would require another web server, another app server, and so much infrastructure all over again. That led to the evolution of microservices. But the challenges associated with these microservices started surfacing such as tracing them, accounting for their availability and so on. In effect, this is what led to the growing reliance on data-centric models of architecture.



Alongside all of this, beginning with the 1970s itself, we began looking at data. So when the GPUs came in, there was a need for data to be processed parallely. This processing led to the development of ML models in the 2000s, where in addition to the application, the ability to perform real time predictive analytics, be it from a DevOps perspective or a business usage perspective crept in. Along with the need for systems to be bigger, better and faster, it became imperative for them to be more secure and sustainable too. AWS came up with the 6 pillars of a well-architected framework which have become industry standards - operational excellence, security, reliability, performance efficiency, cost optimization, and sustainability. And incidentally these have been the foundational pillars of architecture since the 50s, till date and will continue to remain relevant.

What Lies In Store?

As per InfoQ's Software Architecture and Design Trends Report 2024, the emerging trends can be viewed across 4 quadrants - innovators, early adopters, early majority and late majority. Cell-based architecture is the newest addition under the innovators category. It implies that overall availability and fault tolerance can be improved by giving extra attentionto how and where services are deployed, restricting communication paths between services, and isolating faults within one cell. While security continues to be a pillar guiding the design of architecture, the term 'privacy engineering' has been introduced instead because it identifies user privacy as a leading design principle. Similarly, we find 'green software' instead of sustainability. This is because we are considering a system's carbon footprint from day one rather than being reactive when the cloud-cost or corresponding energy usage gets too high. dApps or distributed apps are applications that are hosted on a blockchain. When we speak of a digital evolution where there's a digital currency or digital economy, dApps will play an important role in that. In the early adopters category, you find Dapr - Distributed application run time which is becoming very popular lately. For us at Llyods, this one is close to our heart. The whole idea of Dapr is that your cost cutting concerns can become part of your run time. So, whether one looks at security in terms of passwords, your database pooling, your resiliency patterns, and so on, all these patterns can be codified to become part of your run time. Similarly, you can see that LLMs have moved from the innovators category to the early adopters category. Edge computing also finds itself now in the early adopters category. An interesting development is the move of 'low-code/no-code' to early majority. The trends report mentions that this does not necessarily mean that the majority of the companies are actively designing and building low-code/no-code solutions but instead that the level of innovations has plateaued and that the patterns have become well-established.

The Future

The convergence of Agentic AI, Sustainability, and Quantum Computing, will shape new paradigms in software architecture. These trends will demand highly adaptive, efficient and intelligent systems capable of operating in complex environments. Some of the potential emerging software architectures that could define the future include - sustainability-centric architecture, self-adaptive architectures with MAPE-K loop, hybrid cloud-edge quantum architectures, agentic-AI driven sustainability orchestrators, collaborative agentic ecosystems, quantum safe and ethical architectures, swarm intelligence and decentralised agents, AI-first quantum architectures and quantum-enhanced multi-agent systems.

Conclusion

The future looks very interesting in terms of all that is evolving with AI, Quantum Computing and Sustainability on the one hand and on the other the changes that are taking place in programming languages, and their associated efficiencies. We are working closely with IIITH on the sustainability angle to designing architecture such as workload scheduling among others. All-in-all, exciting times ahead.

RAVI KAPPAGANTU

LLOYDS TECHNOLOGY CENTER

Is Head of Architecture and Innovations, Lloyds Technology Centre India. With over 28 years of experience in the financial services, his expertise also includes cloud engineering, micro services architecture, speech and contact centre technologies.

Enhancing Sustainability of Modern Software Systems through Self-adaptive Architectures

Dr. Karthik Vaidhyanathan explains the concept of software sustainability and how his group's research on self-adaptation is contributing towards greener and sustainable software.

Impact of Uncertainties on Software Sustainability

Imagine receiving an alert that your system has gone down or is consuming an unsustainable amount of power (Yes, you heard that right!). The immediate response is to identify and resolve the issue - whether it's a sudden spike in user traffic, unexpected resource constraints, or a misconfiguration. But what if the system could adapt itself dynamically, anticipating and mitigating these problems in real-time? Modern software systems including AI Systems are subjected to various types of uncertainties ranging from unpredictable user behaviors and fluctuating workloads to resource constraints in the operating environments, evolving security threats and real-world variability in data inputs. Studies show that 64% of system outages result from misconfigurations[1]. And 91% of AI models degrade over time[2]. Beyond impacting reliability and performance, these uncertainties also have a huge impact on the environment (increase in user loads may trigger the need to add more resources). Recent estimates from Green Software Foundation[3] highlight that software emissions are equivalent to the emissions of air, rail and shipping combined. As researchers trying to work in the intersection of software architecture and ML, we are constantly trying to enhance the sustainability of modern software systems including AI-enabled systems. One way we have attempted to tackle this problem is by making systems self-adaptive where they adapt their structure or behavior with minimal human intervention.

Enhancing Sustainability through Self-adaptation

More often than not, people often associate sustainability with making things green. While it's not incorrect, it's not the only aspect of sustainability. It is a multi-dimensional quality attribute which encompasses the technical, environmental, social and economical aspects of a running software system. My research on using self-adaptation to dynamically adapt software systems at runtime to enhance sustainability started during my PhD where the focus was to continuously monitor the metrics of a running software system, identify any potential uncertainties using AI and further use AI to dynamically reconfigure the system to handle the uncertainty. We had applied this concept to IoT systems where we used AI to dynamically switch the processing between edge, fog and cloud to save battery power of IoT devices while guaranteeing system performance[4]. That's when we realized that the idea could be applied to a broader class of software systems.

[1] *https://www.reuters.com/technology/major-tech-outages-recent-years-2024-07-19*

[2] *https://www.nature.com/articles/s41598-022-15245-z*

[3] *https://stateof.greensoftware.foundation/en/insights/software-emissions-are-equivalent-to-air-rail-shipping-combined/*

[4] *Cámara, J., Muccini, H. and Vaidhyanathan, K., 2020, March. Quantitative verification-aided machine learning: A tandem approach for architecting self-adaptive IoT systems. In 2020 IEEE International Conference on Software Architecture (ICSA) (pp. 11-22). IEEE.*

Most of the organizations today are adopting a microservice-based architectural style where each service is designed around domain boundaries and team compositions. At the same time organizations have realized/are starting to realize the importance of monitoring their carbon footprint and further reducing emissions. In this context, we developed an approach that uses self-adaptive mechanisms to dynamically decide which microservice instance to use to guarantee trade-off between latency and energy consumption. To illustrate how this works, consider a scenario where a user located in Hyderabad sends a request to login to an e-commerce application. Typically such a request will be served by an instance of the microservice that is located in a data center closer to the location of the user inorder to guarantee quicker response. However, that instance might be located in a data center which is powered by fossil fuel or the instance might be consuming higher power due to several incoming requests. Hence, it might be beneficial to serve the request from some other instance which is consuming less power and is located in a data center powered by renewable energy but at the compromise of the network latency that may come in due to the location. With this broad thought, we developed an approach that leverages the use of reinforcement learning to decide which microservice instance to use to serve a user request by considering the trade-off between response time and energy consumption. This work has been published in top tier international conferences[5]. Further, we also extended this concept to serverless functions in our recently published work where it was more about deciding the optimal instance to use to allocate a serverless function considering trade-off between cost and performance[6].

Extending to AI-enabled Systems

The emergence of AI has enhanced various walks of our life albeit with a significant cost in terms of the environmental impact. Estimates suggest that data centers already consume 2% of the world's electricity requiring gallons of fresh water with AI taking the bulk of the load and these numbers are only expected to increase[7]. To this end, we believe self-adaptive systems can certainly play a role in enhancing the sustainability of AI systems without compromising on the performance of the system or accuracy of the AI models. For any task today, there are various types of AI models that could be used. For example, if we want to chat, we have ChatGPT or Claude or Llama or Gemini to name a few. Similarly for tasks like object detection, we have different varieties of AI models such as the Yolo family of models or the detectron series, etc. Most of the times we don't need complex AI models, all we need is simple AI models but for some scenarios we may need complex AI models that can provide high accuracy. Based on this thought process, as a starting point, we developed an approach, Eco-MLS8, that presents the architecture of an ML system that leverages the concept of self-adaptation, switching between different AI models at run-time to trade-off between accuracy and energy consumption. We took object detection as our domain and we developed a self-adaptive ML system that switches between different object detection models at runtime by deciding when to use what models without compromising on the accuracy while simultaneously reducing power consumption. We noticed that we could reduce energy consumption by close to 80% by using our approach as opposed to using a large complex model.

[5] Karthik Vaidhyanathan, Mauro Caporuscio, Stefano Florio, and Henry Muccini. 2024. ML-enabled Service Discovery for Microservice Architecture: a QoS Approach. In Proceedings of the 39th ACM/SIGAPP Symposium on Applied Computing (SAC '24)

[6] Jain, P., Singhal, P., Pandey, D., Quatrocchi, G., Vaidhyanathan, K. (2025). POSEIDON: Efficient Function Placement at the Edge Using Deep Reinforcement Learning. In: Gaaloul, W., Sheng, M., Yu, Q., Yangui, S. (eds) Service-Oriented Computing. ICSOC 2024.

[7] https://www.wired.com/story/true-cost-generative-ai-data-centers-energy

[8] M. Tedla, S. Kulkarni and K. Vaidhyanathan, "EcoMLS: A Self-Adaptation Approach for Architecting Green ML-Enabled Systems," in 2024 IEEE 21st International Conference on Software Architecture Companion (ICSA-C), https://arxiv.org/abs/2404.11411

Further, we extended this concept to the larger MLOps pipeline. To keep it simple, MLOps can be thought of as a set of practices that combines machine learning (ML), software engineering, and DevOps to streamline and automate the end-to-end lifecycle of ML development, deployment and management. MLOps pipeline typically enhances the maintainability of the ML system. However, periodically retraining ML models has also had an impact on the environment footprint. Even though it may enhance accuracy of the ML models, the accuracy needs to be also looked at from the perspective of impact on the environment. Sometimes for 1% increase in accuracy, tons of CO2 might be emitted[9]. To this end, we developed an approach to architect self-adaptive MLOps pipelines that decide when to re-train models or switch between models or use a more greener cloud instance for retraining by considering the cost, performance, accuracy and current carbon footprint. This work was published in the prestigious international conference on software architecture (ICSA 2024)[10] where the work won the best poster award.

Onward and Forward

Currently, one of the biggest challenges in software engineering is about enhancing the sustainability of modern software systems, in particular AI systems. There are already efforts going on at a global level towards making softwares more green such as that of Green Software Foundation where different organizations are involved in defining green software engineering practices. There are also a lot of tools that have been made available both by the research and practitioner community to measure power, carbon footprint,etc. However, more concerted effort between the academic and research community is required to create a wider impact. Many times there is also a lack of awareness that needs to be addressed. As far as our research is concerned, we are working on extending our approach of model switching to Generative AI-based applications and to edge cloud continuum. Further, we are also extending our self-adaptive MLOps pipeline to support different types of AI systems. We are also collaborating with different research groups such as the S2 group at VU Amsterdam, the Netherlands to perform studies that can aid practitioners or researchers in building greener software systems. In addition to this, another compelling research angle that is being actively explored in collaboration with Lloyds Technology Centre is the development of practices that can enable architects to come up with green software design and deployment practices.

[9] *https://arxiv.org/abs/1906.02243*

[10] *H. Bhatt, S. Arun, A. Kakran and K. Vaidhyanathan, "Towards Architecting Sustainable MLOps: A Self-Adaptation Approach," in 2024 IEEE 21st International Conference on Software Architecture Companion (ICSA-C) https://arxiv.org/pdf/2404.04572*

DR. KARTHIK VAIDHYANATHAN

IIIT HYDERABAD

is an Assistant Professor at the Software Engineering Research Center, IIIT-Hyderabad, India where he is also associated with the leadership team of the Smart City Living Lab. His main research interests lie in the intersection of software architecture and machine learning with a specific focus on building sustainable software systems. He is also an editorial board member of IEEE Software. Website: https://karthikvaidhyanathan.com, Group: https://sa4s-serc.github.io/

A community initiative, anchored by

INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY
HYDERABAD

IIIT Hyderabad

TechForward
DISPATCH