Research Overview

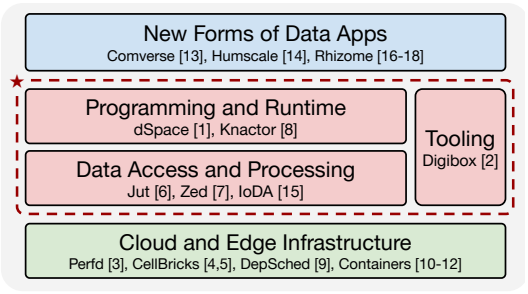
Silvery Fu

"The most profound technologies are those that disappear. They weave themselves into the fabric of everyday life until they are indistinguishable from it."

— Mark Weiser, *The Computer for the 21st Century*, 1991

Three decades ago, Mark Weiser envisaged a future where technology seamlessly integrates into our daily lives. Today, we're closer than ever to this vision with the *Internet of Things* (IoT). A plethora of connected devices and data sources—from smart sensors and appliances to smartphones, wearables, drones, and domestic robots—are increasingly ubiquitous in our living environments. While we have more *devices* and *data* than ever, we are still far from Weiser's full vision. Current *apps* that capture the physical world such as smart homes are confined to vertically-integrated domains or narrow use cases. Systems like big data platforms lack intuitive methods for organizing, processing, and acting upon this new class of data sources. My research aims to bridge this gap, advancing us from the Internet of Things to the *Internet of Data Apps* (IoDA). This transition shifts the focus beyond connecting individual devices to integrating their collective data and capabilities into applications. I use the term *IoT data apps* to refer to applications that consume, process, and provide data and capabilities from and to IoT devices. I envision that with the right *systems support*, IoT data apps can become as ubiquitous as the devices they incorporate, yet significantly more powerful—synergizing with the latest advancements of cloud and edge computing, data analytics, and AI/ML/LLMs, to unlock their full potential.

To that vision, my research focuses on *making IoT data apps easy to develop, operate, and use*, thus *accelerating their innovation and adoption*. Specifically, I take a *principled* approach to build *system frameworks* for IoT data apps. Today's IoT data apps are typically implemented with methods tailored to specific devices and use cases, or by shoehorning existing systems designed for web, mobile, or big data workloads. These narrow and incremental approaches, as I'll explain, leave developers grappling with the discrepancy between *app-level* requirements and *device-centric* abstractions, and the complexity of processing *eclectic* IoT data. They fail to meet the agility and scalability requirements of the IoDA vision. Instead, I argue that supporting this new class of applications requires *innovating at all layers of our systems stack*. My work has shown how we can do so at the app layer (dSpace [1]), in tooling (Digibox [2]), integration (Knactor [8]), data processing (Jut [6]), and data modeling (Zed [7]). In each work, I followed well-established system design principles to (i) identify the right *modularity* and then (ii) design and implement the required architectures, abstractions, and programming and runtime support to realize that modularity. Importantly, this approach has also led to new design principles for IoT data apps.



Project overview. My projects span across layers in the system stack, as illustrated in the diagram. My thesis research centers on system designs for IoT data apps, including the dSpace framework [1] (*SOSP*) and follow-up research [2, 6–8]. Part of my ongoing work considers the designs for new forms of data apps in IoT and related domains [13–15]. In addition to the thesis work, I explore new approaches at the infrastructure layer, such as applying AI/ML for system performance prediction [3] (*NSDI*) and democratizing last-mile networks [4] (*SIGCOMM*). In most of these projects, I aim to develop the work into practical and open-source systems for use by both researchers and industry practitioners. For example, dSpace [1] and Digibox [2] (`https://digi.dev`) are used in multiple ongoing projects and testbeds related to IoT and smart spaces. Dependency scheduling [9], a container orchestration mechanism, has been integrated into Kubernetes and has been used in both production and research systems.

Systems for IoT Data Apps

My thesis research addresses the fundamental challenges that arise when applying current systems to develop IoT data apps. These challenges arise from the unique role that IoT data apps play in interacting with the physical world. This involves processing and exchanging data with devices and other apps, adapting to real-world changes (*e.g.*, device mobility), while ensuring user privacy and control requirements (*e.g.*, access permissions) are consistently enforced. Today, app developers face significant *discrepancy* between the abstractions required by apps and those provided by current systems. While current IoT frameworks like AWS IoT and SmartThings make it easy to discover, network, and program individual devices, they force developers to translate app-level requirements into low-level, device-centric abstractions they provide. For instance, in IoT data apps, app logic, user requests, and policies often center around *"contexts"* or *"spaces,"* such as rooms, labs, or buildings, rather than just *individ-*

ual devices within them. This inherent mismatch can result in significant development overhead and limited app features. Moreover, IoT data apps can incorporate devices from a variety of vendors and owners, facing data that is *heterogeneous* (*e.g.,* spanning many different schemas) and *evolving* (*e.g.,* not anticipated during development, changing outside the control of the app developer). Current IoT frameworks, along with the 3rd-party data systems they integrate with, lack the capabilities to effectively handle this type of *eclectic* data. Thus, it's clear that we need more intuitive and powerful abstractions to **(1)** simplify programming app logic and policies and **(2)** enable apps to easily and efficiently process eclectic IoT data, to support a broader spectrum of IoT data apps.

dSpace is my project that addresses the first problem [1], focusing on *smart spaces* in IoT data apps. While smart spaces aim to provide customized insights and automation (*e.g.,* smart homes), developers today often write a significant amount of code to integrate devices, with functionality in these apps being restricted to simple trigger-action rules over device states. dSpace aims to simplify the development with abstraction and modularity, enabling developers to create and reuse higher-level abstractions (*e.g.,* rooms, homes, or buildings) and providing users with control over these abstractions (*e.g.,* putting an entire home into low-energy mode, delegating rooms to trusted 3rd parties). To achieve this, dSpace provides *(i) digivices* that implement control and actuation, flexible to represent individual devices or groups of devices, whether a physical device (*e.g.,* lamp) or a virtual/abstract one (*e.g.,* home); and *(ii) digidata* that supports the integration of digivices with external AI/analytics frameworks, together called *digis*. dSpace allows *composing* low-level digis to higher-level ones, forming a digi-graph that determines the flow of control and data among digis. Importantly, this composition is dynamic, changing based on policies and real-world events. For example, when a robot vacuum moves between rooms, control is yielded to the new room after a digi-graph update. We apply dSpace to smart homes and show that it allows developers to easily construct 10 sophisticated use-cases, 40% of which cannot be achieved with current systems and the rest take up to 4× more lines of code (SLOC). Further, we show that dSpace abstractions can be mapped to a microservice-based, Kubernetes-compatible runtime that can be easily deployed and scaled on cloud and edge infrastructure.

Digibox extends dSpace to address the abstraction discrepancy problem within the development tooling for IoT data apps [2]. Traditional IoT testing tools typically simulate individual devices in isolation by generating data for each device. Consequently, they fall short in capturing physical-world events such as *device interactions* and human inputs, making it challenging to test apps accurately under realistic scenarios. Digibox introduces a novel *scene-centric* testing approach where developers can program an ensemble of simulated devices that capture their coordinated behaviors in *scenes*. To do so, Digibox builds upon the digi and digi-graph in dSpace to allow creating and composing simulated devices and scenes that capture the physical world. Developers can also use Digibox to download, reuse, and customize existing scenes for testing new apps, or replicate experiments from research.

Jut tackles the second problem of allowing apps to handle eclectic IoT data [6]. Consider a user entering a building; their smartphone can contribute data to the building's smart space apps, just as those apps can provide insights for the smartphone. This requires systems that can ingest and process data from any available data source encountered *at run-time*. However, current systems are typically designed to work with predefined data sources that are manually onboarded during development. Jut, designed for *just-in-time* data access, allows apps to discover and consume data from any available sources, even those unknown at development or installation. Jut combines two novel design choices: (i) *modularizing* data processing systems to align with the app-level concept of the physical world, organizing data pipelines and storage by contexts/spaces instead of devices. This allows apps to easily perform context-based queries and access policies; and (ii) *schema reconciliation* that automatically adapts data processing pipelines to ingest data with new and evolving schemas. Specifically, in (ii), Jut leverages Large Language Models (LLMs) for schema mapping between the source schema (of the incoming data) and target schema (of the pipeline). It then compiles the generated mapping rules into dataflow operators, which are prepended to the data pipeline. This approach enables apps to ingest data from a changing pool of sources while avoiding invoking resource-heavy LLMs on a per data record level. We demonstrate that, across a representative set of devices and scenarios, Jut can implement use cases not easily supported today, or can do so with 3.2-14.8× less development efforts and 3-12× less query complexity than current systems.

Zed addresses the eclectic data challenge at the data model and query language levels [7]. Existing approaches for processing and querying data are not ideal for eclectic data since they impose a tradeoff between efficient querying and simplicity. Zed proposes a *super-structured* data model, where data is represented as self-describing, *typed* records backed by a comprehensive type system. This approach combines the flexibility of the document model (*e.g.,* eliminating the need for upfront schema definition) with the efficiency of relational tables (*e.g.,* supporting

efficient OLAP queries and binary storage formats). Further, Zed provides language-level support for introspecting the schemas of data records, enabling developers to leverage this information for effective data processing. Both dSpace and Jut leverage the Zed language and analytics engine to process IoT data.

Finally, **Knactor** generalizes the composition design in dSpace to simplify composing *app services* [8]. Microservices are increasingly used in modern applications. This leads to a growing need for effective *service composition* that combines services to form the end app. However, traditional *API-centric* composition (*e.g.*, RPC, REST, and Pub/-Sub) introduces rigid code-level coupling, and scatters and hides the composition logic within individual pairs of services. These limitations make it difficult to create, manage, and evolve microservice-based applications. Knactor proposes a novel *data-centric* composition mechanism that aims to (i) *decouple service composition from development* so that composition can be updated *without* modifying service code; (ii) allow programming composition as *data exchanges*. Knactor introduces novel abstractions, including a data exchange graph (DXG) for expressing data exchanges declaratively, and a planner that compiles DXGs into optimized datapaths. In real-world apps, we show that Knactor can significantly simplify composition (7-14× less SLOC) with modest impact on app performance; for workloads that are highly sensitive to millisecond-level latency, direct RPC might still be preferable.

Cloud-Native Systems

The above frameworks (*e.g.*, dSpace [1]) are implemented with a *cloud-native* runtime to address the operational and scale-out requirements of IoT data apps (*e.g.*, scaling from smart homes to campuses and cities). Cloud-native is an approach to building and deploying applications optimized for cloud infrastructure, leveraging techniques such as containers, microservices, Kubernetes, DevOps, and AIOps tools. As an increasing number of applications spanning web, big data, ML, and IoT adopt cloud-native, they pose interesting research challenges. I've explored two key areas: (1) In *AIOps*, I investigated the use of AI/ML to predict the performance of cloud-native apps; and (2) In *app containers*, my focus is on improving the performance and efficiency of containerized applications.

Blackbox Performance Prediction with ML. A growing body of work reports positive results from applying ML-based performance prediction to a particular application or use-case (*e.g.*, server configuration, capacity planning). Yet, a critical question remains unanswered: does ML make prediction simpler (*i.e.*, allowing us to treat systems as blackboxes) and general (*i.e.*, across a range of applications and use-cases)? After all, the potential for simplicity and generality is a key part of what makes ML-based prediction so attractive compared to the traditional approach of relying on handcrafted and specialized performance models. To answer this, we develop a methodology, Perfd [3], for systematically diagnosing whether, when, and why ML does (not) work for performance prediction, and identify steps to improve predictability. We apply Perfd to test 6 ML models in predicting the performance of 13 real-world applications. We find that 12 out of our 13 applications exhibit *inherent variability* in performance that substantially limits prediction accuracy. These findings reveal that blackbox performance prediction is fundamentally limited, often requiring system-level modifications and/or ML-level extensions to achieve predictability. We propose and empirically evaluate these complementary approaches to broaden the applicability of ML-based prediction and show that they can alleviate (but not eliminate) the above limitations.

Efficient Container Orchestration. Containers have become the canonical way of deploying apps and compute tasks at the cloud and the edge [10]. Unfortunately, container startup latency and overhead remain high, limiting responsiveness and resource efficiency of containerized applications. This latency primarily stems from fetching container dependencies including system libraries and data files. I propose a novel approach, *dependency scheduling* [9] (DepSched), where container orchestrators place a task on the node that holds the largest portion of the task's dependencies, while adhering to other task constraints. By leveraging the layered structure of the container filesystem and the resource usage patterns of the container runtime, DepSched further enhances the granularity of dependency matching and reduces fetch time. For typical scenarios, DepSched improves task startup latency by 1.4-2.3x relative to current dependency-agnostic schedulers. In conjunction with my other works on virtualization and cluster orchestration [10–12], my research proposes a comprehensive set of techniques for enhancing the performance and efficiency of containerized applications. My implementation of dependency scheduling has been incorporated into the mainline Kubernetes codebase.

Democratized Infrastructure

My research on IoT data apps and cloud-native systems sparked my interest in alternative infrastructures that better align with the decentralized nature of those apps and use cases. Today's apps typically run on centralized

infrastructure (CI) like public clouds and network telcos. Democratized infrastructure (DI) presents an interesting alternative that can (i) foster market competition, driving innovation and benefiting users, and (ii) align infrastructure ownership with app operators and users, enabling those who use the infrastructure to have a greater say in its operation. The main challenge lies in designing DI systems that provide services and application support equivalent to or better than the current CI systems. I've explored DI in cellular networks, and my ongoing work extends this to data apps [13]. Regarding the former, briefly:

CellBricks is a democratized architecture for cellular networks aiming to lower the barrier to entry for new operators [4]. It enables users to consume network access on-demand from any available cellular operator – small or large, trusted or untrusted. We achieve this by refactoring support for mobility and user management (authentication and billing) out of the network and into the user device and a service broker that acts as an intermediary between users and operators. Importantly, these changes bring valuable benefits beyond enabling competition: they lead to an infrastructure that is simpler and more efficient. We show that CellBricks' benefits come at little-to-no cost in performance, with application performance overhead between -1.6% to 3.1% of that achieved by current cellular infrastructure. Moreover, DI enables new system design goals that are challenging to achieve with CI. For example, cellular operators today know both the identity and location of their mobile subscribers and hence can easily profile users based on this information. In LOCA [5], we leverage CellBricks to divide this information among the broker and operators, guaranteeing that no party except the user knows both user identity and location. The key challenge here is to reconcile privacy with an operator's need to provide services based on user identity (*e.g.,* post-pay, QoS classes, emergency services) and settle billing with the broker. We tackle this with a novel design that combines cryptographic primitives like unlinkable tokens and zero-knowledge computation, allowing operators to offer services and settle billing without compromising location privacy. Our results show that LOCA provides strong privacy guarantees and is scalable for realistic deployments.

Ongoing / Future Research

My current work focuses on extending my research in IoT and cloud-native systems to new areas and continue building toward the Internet of Data Apps vision. Some areas that I'm actively working on:

Making IoT a utility. How can we make IoT truly ubiquitous and "invisible"? I believe this requires *data platforms* that transform IoT data and capabilities into a *utility*, so that any apps like smart spaces, robotics, vehicles, and LLM agents could consume them in an *on-demand* and *controlled* manner. My frameworks [1, 6–8] provide important building blocks for such platforms. However, many open challenges remain. For instance, how does the platform incentivize data sharing? How does it address privacy and provide provenance and governance over decentralized data sources and their consumers? How can it reliably discover data sources for apps on-demand? Moreover, how can we ensure that such platforms scale to (say) smart cities with millions of devices and apps, providing real-time data access? This goal raises interesting problems that require new design approaches.

Enabling new paradigms of data apps. Today, app users face two primary challenges: their data is *fragmented* across various apps, and within each app, they have *limited control* over shared data as dictated by the provider. My ongoing projects propose a lightweight shim layer to unify data access across personal data apps while minimizing changes at the app providers [14], and a provider-driven alternative [15] to enable data exchanges between multiple app silos. Besides, I'm exploring "local-first" designs for decentralized, community-based apps, in which participants can retain fine-grained control over their data without sacrificing functionality [13]. Another emerging class of data apps involves real-time user interactions over multimedia data, such as multi-modal LLMs with text, video, and audio streams. These apps are compute- and network-intensive with stringent QoS. I plan to combine my past experiences in real-time game streaming [16–18] to address these system challenges.

Rethinking software integration. My work proposed new ways of composing apps beyond code dependencies and APIs, *e.g.,* over data and using AI/LLMs. Can we generalize this to enable fast and scalable integration for more systems? Such solutions, I believe, should intelligently combine code generation and run-time translations.

Performant analytics over eclectic data. Eclectic data raises new challenges to process, store, and query data efficiently and at scale. Given they span across heterogeneous and evolving schemas, existing optimizations such as columnar layouts and vectorized execution would not work out of the box. I plan to adapt these techniques and develop new ones that leverage unique characteristics of our super-structured model [7], such as query optimization that incorporates type information, to improve the performance of processing eclectic data.

Overall, as a researcher, I believe the biggest "meta-problem" is to find and choose the problems that I work on. I'm interested in identifying problems that emerge at the intersection of new app-demands (*e.g.,* data apps) and tech-advancements (*e.g.,* cloud, edge, IoT, and AI), and in seeking principled and practical system designs.

References

[1] Silvery Fu and Sylvia Ratnasamy. dSpace: Composable Abstractions for Smart Spaces. In *Proc. ACM SOSP*, 2021.

[2] Silvery Fu, Hong Zhang, Sylvia Ratnasamy, and Ion Stoica. The Internet of Things in a Laptop: Rapid Prototyping for IoT Applications with Digibox. In *Proc. ACM HotNets*, 2022.

[3] Silvery Fu, Saurabh Gupta, Radhika Mittal, and Sylvia Ratnasamy. On the Use of ML for Blackbox System Performance Prediction. In *Proc. USENIX NSDI*, 2021.

[4] Zhihong Luo, Silvery Fu, Mark Theis, Shaddi Hasan, Sylvia Ratnasamy, and Scott Shenker. Democratizing Cellular Access with CellBricks. In *Proc. ACM SIGCOMM*, 2021.

[5] Zhihong Luo, Silvery Fu, Natacha Crooks, Shaddi Hasan, Christian Maciocco, Sylvia Ratnasamy, and Scott Shenker. LOCA: A Location-Oblivious Cellular Architecture. In *Proc. USENIX NSDI*, 2023.

[6] Silvery Fu, Jamsheed Mistri, Siyuan Dong, Steve McCanne, Amy Ousterhout, and Sylvia Ratnasamy. Jut: A Framework for Just-in-Time Data Access. 2023. Under review (draft on request).

[7] Amy Ousterhout, Steven McCanne, Henri Dubois-Ferrière, Silvery Fu, Sylvia Ratnasamy, and Noah Treuhaft. Zed: Leveraging Data Types to Process Eclectic Data. In *Proc. CIDR*, 2023.

[8] Silvery Fu, Hong Zhang, Ryan Teoh, Taras Priadka, and Sylvia Ratnasamy. From Kubernetes to Knactor: A Data-Centric Rethink of Service Composition. *CoRR abs/2309.01805*, 2023. Under submission.

[9] Silvery Fu, Radhika Mittal, Lei Zhang, and Sylvia Ratnasamy. Fast and Efficient Container Startup at the Edge via Dependency Scheduling. In *Proc. USENIX HotEdge*, 2020.

[10] Silvery Fu, Jiangchuan Liu, Xiaowen Chu, and Yueming Hu. Toward a Standard Interface for Cloud Providers: The Container as the Narrow Waist. *IEEE Internet Computing*, 2016.

[11] Silvery Fu, Yifei Zhu, and Jiangchuan Liu. HARV: Harnessing Hybrid Virtualization to Improve Instance (Re)Usage in Public cloud. In *Proc. IEEE/ACM IWQoS*, 2017.

[12] Yifei Zhu, Silvery Fu, Jiangchuan Liu, and Yong Cui. Truthful Online Auction for Cloud Instance Subletting. In *Proc. IEEE ICDCS*, 2017.

[13] Silvery Fu, Dylan Reimer, Siyuan Dong, Yifei Zhu, and Sylvia Ratnasamy. Comverse: A Federative-by-Design Platform for Community Computing. *CoRR abs/2308.15219*, 2023. Under preparation.

[14] Silvery Fu, Pratyush Das, and Sylvia Ratnasamy. Human-Scale Computing: A Case for Progressive Narrow Waist for Internet Applications. *CoRR abs/2308.14928*, 2023. Under preparation.

[15] Silvery Fu and Sylvia Ratnasamy. From Internet of Things to Internet of Data Apps. *CoRR abs/2309.04546*, 2023.

[16] Silvery Fu, Jiangchuan Liu, and Wenwu Zhu. Multimedia Content Delivery with Network Function Virtualization. *IEEE MultiMedia*, 2017.

[17] Lei Zhang, Silvery Fu, Jiangchuan Liu, Edith Cheuk-Han Ngai, and Wenwu Zhu. On Energy-efficient Offloading in Mobile Cloud for Real-time Video Applications. *IEEE Transactions on Circuits and Systems for Video Technology (TCSVT)*, 2016.

[18] Ryan Shea, Silvery Fu, and Jiangchuan Liu. Rhizome: Utilizing the Public Cloud to Provide 3D Gaming Infrastructure. In *Proc. ACM MMSys*, 2015.