GROWFLOW: Low-latency Computational Fluid Dynamics at the Edge for Real-time Agricultural Decision Support

Liubov Kurafeeva[*], Benjamin Carter[*], Ryan Hartung[†], Alan Subedi[‡], Avhishek Biswas[‡], Michael Fay[‡], Shantenu Jha[§,¶], Chandra Krintz[*], Andre Merzky[¶], Douglas Thain[†], Mehmet Can Vuran[‡], Rich Wolski[*]

[*]UC Santa Barbara [†]University of Notre Dame [‡]University of Nebraska–Lincoln
[§]Princeton Plasma Physics Laboratory [¶]Rutgers University

***Abstract*—Smart, connected applications increasingly require low-latency inference from models that adapt as physical conditions evolve. However, high-fidelity simulation and model generation require substantial remote computation, making them unsuitable for the IoT inference critical path. We present GROWFLOW, an adaptive IoT system that decouples low-latency edge inference from computationally intensive model generation using an asynchronous online/offline architecture. The online path serves inference using the latest surrogate model, while the offline path continuously executes high-fidelity simulation and model training on remote HPC resources. We evaluate GROWFLOW in a field-scale digital agriculture deployment using CFD-generated wind fields and Principal Component Regression (PC-R). GROWFLOW refreshes models in approximately 31 minutes and provides sub-second inference across 12,920 spatial locations, over $1000\times$ faster than direct CFD simulation. Under continuous operation, model staleness remains approximately 0.5–1 h with prediction error comparable to sensor measurement uncertainty; without refresh, error increases substantially. These results demonstrate how asynchronous model updating can support responsive, adaptive IoT inference despite disparate inference and model-generation time scales.**

I. INTRODUCTION

Smart, connected applications increasingly integrate distributed sensing, edge intelligence, and high-fidelity computational models to support low-latency, adaptive decision-making [1], [2], [3], [4]. Emerging smart systems increasingly span the computing continuum: resource-constrained sensors observe the physical environment, edge systems provide responsive local intelligence, and remote cloud or HPC resources perform computationally intensive analysis and model generation. Decoupling these computational tiers allows applications to exploit powerful remote resources without placing their latency on the critical path for local decision-making. Such capabilities are increasingly important in applications including intelligent transportation, *lights out* industrial control, smart infrastructure, environmental monitoring, and digital agriculture [5].

The distributed systems that underlie these IoT applications must enable processing of high-frequency sensor streams at the point of data ingestion, minimize end-to-end inference latency, reduce network bandwidth consumption, and enhance fault tolerance by enabling continuous operation when disconnected. For applications that rely on inferences of physical processes generated from relatively sparse sensing, these systems must additionally maintain models that reflect evolving environmental conditions (often spatially indexed) using distributed, heterogeneous, and resource-constrained infrastructure.

The central challenge that such systems face is the mismatch between the time scale at which IoT applications require inference and the time scale at which high-fidelity models can be generated or refreshed (including models executed using large-scale resources sited in data centers). Specifically, (i) high-fidelity physics simulations and model training are computationally intensive and often executed remotely from the data acquisition site, (ii) edge connectivity is commonly intermittent and bandwidth-limited, and (iii) because data acquisition is sparse, environmental dynamics can invalidate previously fit, or "trained," models. As a result, end-to-end systems must not only integrate heterogeneous resources, but also adapt their behavior autonomously as physical conditions change and as heterogeneous infrastructure capability, capacity, and availability fluctuates.

Prior work on edge intelligence and digital twins has explored distributed inference and physics-based modeling [3], [6], while surrogate modeling accelerates computationally expensive simulations [7], [8]. However, these approaches generally treat model training and deployment separately. Continuous-learning systems address model drift [9], but do not typically account for training data generated through remotely scheduled, high-fidelity simulations. Distributed platforms such as xGFabric [10] coordinate sensing, edge, and HPC resources, but do not address simulation-driven model refresh. Consequently, maintaining low-latency inference while continuously updating models under variable simulation, scheduling, and network delays remains an open systems challenge.

To address this research gap, we present GROWFLOW, a closed-loop distributed modeling system that decouples low-latency IoT inference from computationally intensive model generation. Its online component executes at the edge, continuously ingesting sensor data and serving inferences using the most recently available surrogate model. In parallel, an offline component executes high-fidelity simulation and surrogate model retraining on remote HPC resources and asynchronously publishes up-

dated models back to the edge. Because inference never waits for the offline path, the system remains responsive despite simulation, network, and batch-scheduling delays. The resulting design defines model *staleness* as the duration between the time at which the last element of training data was recorded and the time when the inference model trained with it is used. GROWFLOW treats staleness as a first-class systems concern and continuously refreshes the deployed model to bound its impact on inference accuracy.

We make four contributions. First, we present a hybrid modeling architecture that decouples low-latency edge inference from simulation-driven model generation under realistic latency and connectivity constraints. Second, we implement this architecture across sensing, edge, and HPC resources and deploy it as part of a field-scale digital agriculture application that models wind conditions inside a production-scale citrus screen house. Third, we empirically evaluate the deployed system, showing sub-second inference over 12,920 spatial locations, an approximately 31-minute asynchronous model-refresh cycle, and surrogate predictions that remain within operationally useful error bounds while achieving over three orders of magnitude lower inference latency than direct CFD simulation. Finally, we characterize model staleness as a measurable systems property, quantifying how prediction error evolves as model refresh is delayed and showing how asynchronous refresh bounds staleness-induced degradation.

II. GROWFLOW ARCHITECTURE

Our goal with GROWFLOW is to decouple the time scale of low-latency IoT inference from the much longer, and possibly more variable, latencies associated with high-fidelity model generation. To achieve this, we design the system as an asynchronous online/offline loop as shown in Figure 1.

Asynchronous Online/Offline Design: At a high level, GROWFLOW consists of an offline and online component, both of which operate continuously and communicate asynchronously. The online component runs at the edge, ingesting sensor data and producing low-latency inferences on-demand by parameterizing the most recently available surrogate model. In parallel, the offline component executes computationally intensive simulation and model retraining on HPC resources and asynchronously publishes updated surrogate models for online use. Rather than retraining on a fixed schedule, GROWFLOW continuously executes the offline pipeline: each iteration begins as soon as the preceding iteration completes, using the most recent sensor observations available at that time. Thus, model-refresh cadence is determined by the execution time and availability of the remote computational resources rather than by a predefined interval. This approach minimizes the time

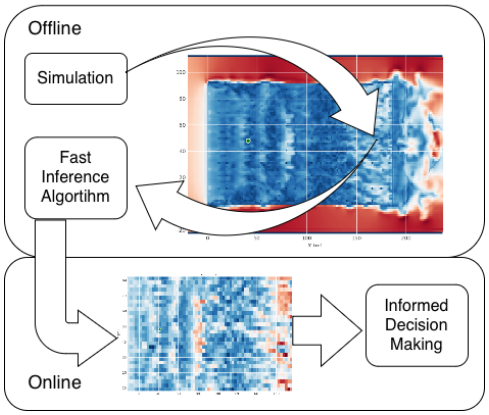


Fig. 1: GROWFLOW Online/Offline Architecture. Low-latency edge inference executes independently while high-fidelity simulation and model retraining executes on HPC resources and publishes "fresh" surrogate models to the edge asynchronously.

that the online component relies on a stale model, allowing inference to proceed independently of the offline pipeline. It also treats model staleness as an explicit systems concern rather than assuming that model updates are instantaneous or continuously available.

Hybrid Simulation and Fast Surrogate Inference: GROWFLOW separates expensive high-fidelity model generation from fast surrogate inference. In our prototype, the offline path implements Computational Fluid Dynamics (CFD) to generate high-resolution training data; the online path uses a lightweight surrogate model suitable for edge execution. High-fidelity simulation can capture spatially complex physical processes but is too computationally expensive for the inference critical path. Surrogate models provide low-latency predictions but depend on timely, representative training data [7], [8]. By decoupling these two functions, GROWFLOW allows online inference to remain responsive while the offline path continuously refreshes model fidelity as conditions evolve. This separation makes the architecture extensible: the online inference model is not inherently tied to a particular surrogate modeling technique. Alternative inference approaches can be incorporated provided that they can be trained or updated using data generated by the offline path and deployed within the resource and latency constraints of the online environment. Herein, we instantiate and evaluate this design using Principal Component Regression (PC-R); we plan to explore alternative surrogates in future work.

Distributed Execution Substrate: GROWFLOW is implemented using the open source xGFabric [10], a distributed systems platform for coordinating sensing, edge, cloud, and HPC resources under variable connectivity and scheduling delays. xGFabric provides delay-tolerant data movement, workflow coordination, and

model synchronization using CSPOT as its persistent log-based runtime [11], [12], and integrates HPC runtime technologies for remote simulation and training execution [13]. These mechanisms allow the online component to continue operating while remote computation or communication is delayed. However, xGFabric does not determine which models are generated, when they are refreshed, or which model the edge uses for inference. GROWFLOW builds on this substrate by adding (i) the asynchronous online/offline loop that removes simulation and training from the inference critical path, (ii) a CFD-to-surrogate pipeline that transforms each simulation output to the inference grid and trains a deployable edge model, and (iii) explicit measurement of model staleness as a systems property.

This organization yields two important properties. First, online inference is non-blocking with respect to remote simulation and training: HPC queuing, network disruption, or long-running computation may delay model refresh but do not delay the current inference path. Second, model staleness becomes an observable systems property determined by the latency of the offline loop. We evaluate this tradeoff explicitly in Section IV by measuring both refresh latency and the effect of model staleness on inference accuracy.

III. IMPLEMENTATION AND IOT DEPLOYMENT

To evaluate GROWFLOW, we instantiate its architecture in a commercial, field-scale digital agriculture deployment that estimates spatial wind conditions within a cyber-physical growing structure. Sensor observations collected at the edge parameterize high-fidelity CFD simulations that generate training sets for surrogate model training on large-scale compute resources in a data center. The resulting models are asynchronously returned to the edge, where GROWFLOW uses the most recent model to provide low-latency wind-field inference on demand, using current sensor readings as inputs.

Science Driver and Cyber-physical Infrastructure: Our science driver is Citrus Under Protective Screens (CUPS), a 4-acre screen house in California that protects citrus crops from pests while creating a modified microclimate [14]. Its mesh enclosure and internal structures modify growing conditions in ways that are difficult to characterize using sparse sensing alone.

Screen houses like this one introduce complex, spatially heterogeneous airflow and temperature fields that are costly to measure densely with sensors and challenging to predict accurately using either purely physics-based or purely data-driven methods. GROWFLOW uses a small number of meteorological sensors inside and outside the structure to produce spatially resolved wind-field estimates that can inform operational decisions such as spray planning and frost mitigation. Grounding the architecture in a real-world use case enables us to validate the GROWFLOW design and to measure end-to-end latency, computational overhead, and model staleness under realistic operating conditions.

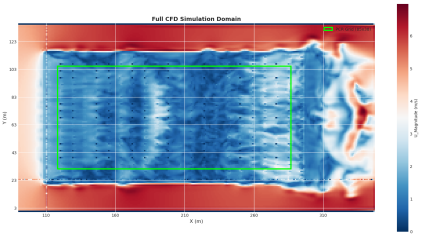


Fig. 2: CFD-simulated wind speed (m/s) at $Z = 1$ m within the CUPS structure. Colors indicate wind speed, with predominant airflow from left to right. The green rectangle outlines the growing region.

Edge Sensing and Orchestration: Davis Instruments weather stations [15] deployed inside and outside CUPS report wind speed and direction every five minutes to an on-site Intel NUC [16] edge cluster located in an outbuilding near the CUPS. GROWFLOW uses CSPOT as part of xGFabric to persist sensor observations, asynchronously transfer them to remote compute resources, and deliver updated surrogate models back to the edge. The online component executes at the edge using recent locally available observations and the latest published surrogate, allowing inference to continue independently of remote connectivity or offline-loop progress.

Hybrid Simulation: The offline path uses a previously validated OpenFOAM CFD model of CUPS [17], [18] to generate three-dimensional wind fields from observed external wind conditions. Each simulation produces approximately 6.5 million irregularly spaced, three-dimensional wind-speed vectors over the structure. GROWFLOW executes parallel CFD simulations on HPC resources using newly observed sensor readings external to the CUPS as boundary conditions.

Surrogate Training and Inference: At the edge, GROWFLOW uses Principal Component Regression (PC-R) [19] to infer the wind speed at each point of the spatial grid within the CUPS at a pre-determined spatial resolution. We select PC-R because its low training and inference cost reduces offline-loop latency while capturing temporal autocorrelation in wind observations.

Each inference uses a 12-sample (one-hour) window of external wind observations. During training, GROWFLOW constructs overlapping one-hour windows from the most recent six hours of observations and associates each with CFD-generated wind values at specific spatial locations. The speed of the training (the generation of the regression coefficients for each point in the CUPS) can be tuned based on the desired resolution of the output. In this study, GROWFLOW transforms CFD outputs from an irregular 6.5-million-cell mesh to a regular 12,920-point inference grid using spatial nearest-neighbor indexing via a KD-Tree [20]. This translates

to a growing region of 168.2 m $\times$ 74.0 m $\times$ 6.0 m at 2m resolution within the CUPS. We also evaluate height $Z = 7$ which sits just above the CUPS porous roof and, while outside the growing region proper, remains a point of interest for characterizing airflow at the transition from the structure's roof to open air (e.g. for tracking aerosol dispersal escaping from the CUPS during spraying).

Figure 2 illustrates a representative CFD-generated wind field at Z = 1m. The spatially horizontal CUPS growing region is depicted as a green rectangle. The predominant wind direction is from the left (which is North on site). The color gradient shows the wind speeds (red is high, blue is low). The simulation captures substantial wind attenuation across the screened structure and spatial variation induced by the enclosure and internal support structures, motivating the need for spatially resolved inference beyond sparse sensor measurements.

IV. Evaluation

We next empirically evaluate the GrowFlow system deployed for the CUPS structure and digital agriculture application described above. Our evaluation addresses four questions:

1) Can the offline pipeline refresh models at a time scale suitable for continuous operation?
2) Does continuous model refresh effectively limit prediction degradation due to model staleness?
3) Can the surrogate model accurately approximate high-fidelity simulation?
4) Can the resulting surrogate provide low-latency inference suitable for edge deployment?

We evaluate the offline pipeline using 18 32-core virtual machines (VMs) distributed across five ARM server nodes connected by 10-Gb Ethernet. One VM runs GrowFlow orchestration and 17 execute simulation and training tasks in parallel. Each VM has 32 cores, 96GB of memory, and 100 GB of disk storage. While each physical node supports 128 cores, we did not observe additional speedup from OpenFOAM CFD simulation beyond 32 cores. All machines are configured with Ubuntu 20.04, OpenFOAM 11 and Python3.

A. Offline Pipeline Performance

We first evaluate the offline loop iteration described in Section II. We report the mean and standard deviation of the timings over 10 iterations of the complete offline pipeline, excluding the initial iteration, which reflects one-time system initialization effects. The evaluation periods spanned varying wind conditions (which can affect CFD convergence and surrogate training times).

The total time for an offline loop iteration using the setup described above is 31.23 minutes on average (2.78 stdev). Because each iteration begins immediately upon completion of the previous one, this execution time also determines the model-refresh cadence under continuous operation. CFD simulation accounts for the majority of this time with an average of 22.73 minutes (0.49 stdev). PC-R model training requires 3.8 minutes on average (0.38 stdev). The remaining time consists of data transfer (0.41 minutes on average, 0.42 stdev), simulation output spatial transformation (4.27 minutes on average, 2.25 stdev), and general system overhead (< 1 second). Thus, although high-fidelity model generation requires approximately 31 minutes, this latency is removed from the online inference critical path by GrowFlow's asynchronous architecture.

B. Model Staleness

We define *model staleness* as the elapsed time between the most recent sensor observation represented in a surrogate's training data and the time at which that surrogate is queried. To quantify its effect on prediction accuracy, we measure PC-R mean absolute error (MAE) as a function of staleness across multiple stable, non-zero wind periods using a training window representative of typical operation (Figure 3).

During our experiments, each offline iteration completes in approximately 31 minutes on average, at which point the next iteration begins immediately using the newest available sensor observations. Thus, under continuous operation, a newly published model is already approximately 0.5 h stale when it becomes available, and is replaced approximately 31 minutes later by the model produced by the subsequent iteration. GrowFlow therefore typically operates with model staleness between approximately 0.5 and 1 h.

GrowFlow does not guarantee this cadence, however. It is a consequence of the available resources and the scheduling regime. A batch scheduler that implements queuing, for example, can increase this cadence and make it far more irregular. That is, the offline pipeline runs as compute resources and new sensor data permit, so an individual refresh cycle can occasionally be delayed. Figure 3 depicts the relationship between this delay and MAE. For example, even if model staleness extends to 2 h, approximately twice the upper end of GrowFlow's typical operational window, MAE remains comparable in magnitude to the 0.44–0.88 m/s measurement uncertainty of the deployed sensors reported by the manufacturer. The PC-R surrogate model accuracy decays rapidly when the model staleness exceeds 6h. The staleness curve therefore characterizes not only normal operation but GrowFlow 's ability to tolerate interruption in compute or data availability.

Figure 3 marks five staleness regions. The 0–0.5 h region is not reached operationally because one offline iteration is required before a model becomes available. For this application, GrowFlow typically operates between 0.5 and 1 h of staleness, where MAE remains

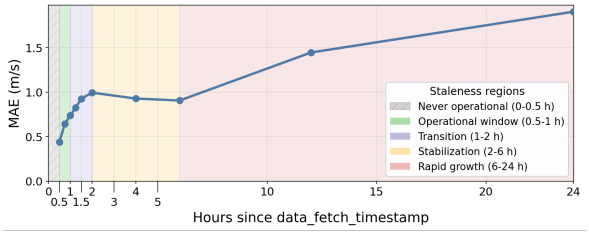


Fig. 3: PC-R MAE vs. hours elapsed since training data fetch timestamp (6-hour training window). The x-axis origin (t=0) is the most recent sensor record used in training; one offline loop iteration (≈31min) must complete before the model is available, placing the operational staleness window at approximately 0.5h. Without continuous refresh, MAE rises to ≈1.9m/s by 24h.

approximately 0.65–1.0 m/s. Error remains relatively stable through approximately 6 h, partly because recurring wind conditions can temporarily return toward those represented in training. Beyond 6 h, however, staleness-induced error increases substantially, reaching approximately 1.9 m/s by 24 h. Because GROWFLOW 's refresh cadence is roughly an order of magnitude shorter than this 6h onset of rapid error growth, even a delayed cycle would need to persist for hours before accuracy meaningfully degrades – motivating GROWFLOW 's continuous, best-effort model refresh via the asynchronous online/offline loop rather than a hard real-time refresh guarantee.

C. Surrogate Accuracy

The results in this section (and in the staleness analysis above) measure how well PC-R approximates the CFD simulation on which it is trained (i.e. surrogate approximation accuracy), rather than accuracy with respect to actual wind conditions inside the CUPS. The OpenFOAM CFD model that GROWFLOW uses was previously validated against in situ sensor measurements [17]. Thus, end-to-end error reflects both CFD model error and surrogate approximation error. We evaluate surrogate accuracy using two contiguous four-hour periods, Period 1 and Period 2, separated at 10:18 (cf Figure 4). Both periods contain an extended stable high-wind regime reaching comparable peak magnitudes (≈13m/s), but differing in timing and duration, bounded by lower and more variable conditions. We train the PC-R surrogate using simulation-generated data corresponding to Period 1 and evaluate both its fit to Period 1 and its ability to predict the unseen conditions in Period 2. Over all points, the training fit is excellent with $R^2 = 0.9399$ and $RMSE = 0.1624$m/s. Figure 5 compares PC-R predictions against CFD ground truth for Period 2 at a single, randomly chosen interior grid point representative of typical (non-extreme) behavior.

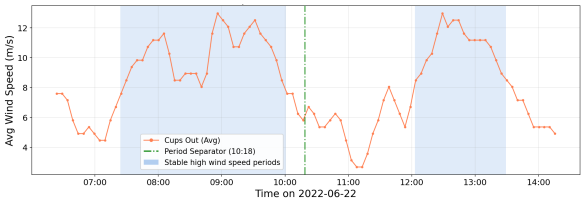


Fig. 4: Historical wind measurements outside the CUPS spanning Period 1 and Period 2, separated at 10:18 (green line). Each period contains an extended stable high-wind regime (shaded), reaching comparable peak magnitudes (≈13m/s), bounded by lower, more variable conditions.

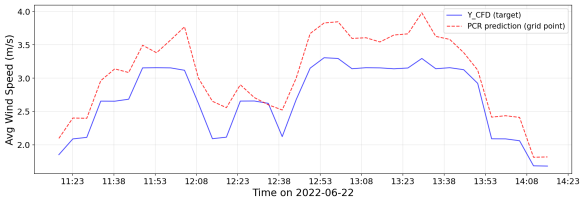


Fig. 5: Comparison of CFD ground truth (solid) and PC-R predictions (dashed) at a representative interior grid point (x=40.3m, y=48.1m, z=5.0m, relative to the growing-region origin) over the full Period 2 duration in Figure 4. The model is trained exclusively on Period 1; at this point, prediction accuracy is MAE=0.37m/s, R^2=0.39, $RMSE$=0.40m/s.

Table I shows the model accuracy statistics when we use PC-R to predict unseen conditions in Period 2 over a significant future time period (4 hours). The table reports data for multiple structure heights (Z column) and presents the MAE and RMSE average and standard deviation in meters per second across points (Pts column). The worst case is located close to the roof ($Z = 5m$) of the structure, where the porous roof has the most turbulence effects. Figure 5 shows a single grid point under a comparatively calm wind trajectory (2–4 m/s, well below the site's ≈13 m/s peaks). This point's MAE (0.37m/s) is well below the Z=5m average in Table I (1.19m/s), showing that individual points can deviate substantially, in either direction, from the aggregate staleness-driven error.

To contextualize the average MAE of 0.80m/s, we note that external wind speeds at the CUPS site range from 0 to 17m/s and average approximately 5m/s; the MAE therefore represents roughly 16% of the mean wind speed. For the operational decisions this system supports—spray timing, frost mitigation, and environmental monitoring—a useful reference point for interpreting this error is the accuracy of the sensors themselves. GROWFLOW's overall MAE is comparable in magnitude to the manufacturer's reported sensor error range (0.44–0.88m/s), indicating that surrogate

TABLE I: PC-R Model Prediction Accuracy for Future Time Period. Wind speed statistics across XY grid points at each height considered (Z) when we train on Period 1 and evaluate on Period 2 (Period-2 data held out from training). The average MAE across points and heights is 0.80 m/s.

Z (m)	Pts	RMSE Mean	RMSE Stdev	MAE Mean (m/s)	MAE Stdev (m/s)
1.0	3230	0.5553	0.3248	0.4685	0.3008
3.0	3230	0.5003	0.2776	0.4160	0.2477
5.0	3230	1.2954	0.8962	1.1884	0.8650
7.0	3230	1.2460	1.0359	1.1247	0.9599
All	12920	0.8993	0.8082	**0.7994**	0.7640

approximation error is commensurate with measurement uncertainty at operational scales. This comparison provides context for the magnitude of the error, but does not on its own establish accuracy with respect to actual wind conditions; direct field validation of the surrogate is left for future work.

D. Online Inference Performance

A critical advantage of the PC-R approach is the computational efficiency of online inference. Once trained, PC-R computes predictions for all 12,920 spatial locations in less than one second on the edge hardware, compared with approximately 22.73 minutes for a corresponding individual CFD simulation. This $> 1000\times$ latency reduction enables the online path to provide responsive spatial inference while high-fidelity model generation proceeds asynchronously. Similarly, generating a complete horizontal plane visualization (approximately 3,230 grid points) takes less than one second on the edge hardware, providing growers with immediate access to actionable wind field information.

V. Conclusions

In this paper, we present GrowFlow, a distributed IoT system that decouples low-latency edge inference from computationally intensive high-fidelity model generation. Through an asynchronous online/offline architecture, GrowFlow uses high-fidelity models to train fast surrogates for inference at the edge and continuously refreshes edge inference models without blocking online operation. Evaluation in a field-scale deployment demonstrates sub-second spatial inference, a 31-minute model-refresh cycle, and reduced accuracy degradation from model staleness. These results show how asynchronous simulation-driven model updating can enable responsive IoT inference despite the substantially different time scales of edge inference and high-fidelity simulation. More broadly, the approach provides a pattern for smart connected systems that require responsive local intelligence derived from computationally intensive remote models. This research is supported in part by NSF awards CNS-2444318 and CNS-2107101, and DOE award DE-SC0025541.

References

[1] S. Liu, L. Liu, J. Tang, B. Yu, Y. Wang, and W. Shi, "Edge computing for autonomous driving: Opportunities and challenges," *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1697–1716, 2019.

[2] National Academies of Sciences Engineering and Medicine, "Foundational Research Gaps and Future Directions for Digital Twins," The National Academies Press. https://doi.org/10.17226/26894., 2024, [Online; accessed 11-July-2024].

[3] W. Purcell and T. Neubauer, "Digital twins in agriculture: A state-of-the-art review," *Smart Agricultural Technology*, vol. 3, 2023.

[4] F. Lordan, D. Lezzi, and R. M. Badia, "Colony: Parallel functions as a service on the cloud-edge continuum," in *European Conference on Parallel Processing*, 2021, pp. 269–284.

[5] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, 2016.

[6] M. Passarotto, F. Ben Abdesslem, J. Eriksson, and T. Voigt, "Edge ai on constrained devices: A survey of deployment, optimization, and lifecycle management," *ACM Transactions on Embedded Computing Systems*, 2026.

[7] T. Chen, R. Li, X. Hu, B. Zhang, Y. Liu, L. L. Wang, and N. Gao, "Machine learning as cfd surrogate models for rapid prediction of building-related physical fields: A review of methods and state-of-the-art," *Building and Environment*, vol. 285, p. 113667, 2025.

[8] A. Tallet, C. Allery, and F. Allard, "Pod approach to determine in real-time the temperature distribution in a cavity," *Building and Environment*, vol. 93, pp. 34–49, 2015.

[9] D. Sculley, G. Holt, D. Golovin, E. Davydov, T. Phillips, D. Ebner, V. Chaudhary, M. Young, J.-F. Crespo, and D. Dennison, "Hidden technical debt in machine learning systems," in *Advances in Neural Information Processing Systems 28 (NeurIPS)*. Curran Associates, Inc., 2015, pp. 2494–2502.

[10] L. Kurafeeva, A. Subedi, R. Hartung, M. Fay, A. Biswas, S. Jha, O. O. Kilic, C. Krintz, A. Merzky, D. Thain, M. C. Vuran, and R. Wolski, "xGFabric: Coupling Sensor Networks and HPC Facilities with Private 5G Wireless Networks for Real-Time Digital Agriculture," in *SC XLOOP Workshop*, 2025.

[11] R. Wolski, C. Krintz, F. Bakir, G. George, and W.-T. Lin, "CSPOT: Portable, Multi-scale Functions-as-a-Service for IoT," in *ACM Symposium on Edge Computing*, 2019, pp. 1–14.

[12] R. Wolski, C. Krintz, and M. Mock, "Leveraging Dataflow as an Intermediate Representation for Portable Edge Deployments," in *IEEE Conference on Fog and Mobile Edge Computing*, 2025.

[13] A. Merzky, M. Turilli, M. Titov, A. Al-Saadi, and S. Jha, "Design and performance characterization of radical-pilot on leadership-class platforms," 2021. [Online]. Available: https://arxiv.org/abs/2103.00091

[14] P. E. Rolshausen, G. Barry, and A. Schumann, "Citrus Undercover Production System (CUPS)," *Citrograph*, vol. 10, no. 1, 2019.

[15] Davis, "Vantage Pro2 — davisinstruments.com," https://www.davisinstruments.com/collections/vantage-pro2, 2024, [Accessed 23-08-2024].

[16] "Intel NUC," http://www.intel.com/content/www/us/en/nuc/overview.html [Online; accessed 14-Feb-2015].

[17] L. Kurafeeva, R. Wolski, C. Krintz, and T. Smyth, "Airflow modeling for citrus under protective screens," *Sensors*, vol. 24, no. 19, 2024.

[18] "OpenFOAM — openfoam.com," https://www.openfoam.com/, 2024, [Accessed 26-08-2024].

[19] I. T. Jolliffe, "A note on the use of principal components in regression," *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, vol. 31, no. 3, pp. 300–303, 1982.

[20] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *CACM*, vol. 18, no. 9, 1975.