

Scalable-Application Design for the IoT

Jagannathan Venkatesh, University of California, San Diego

Bariş Akşanlı, San Diego State University

Christine S. Chan, Alper S. Akyürek, and Tajana Š. Rosing, University of California, San Diego

// A modular approach breaks IoT applications up into functional units called context engines, whose I/O transformations are driven by machine learning. In a smart-grid case study, this approach provided better accuracy and scaling than the current monolithic approach. //



THE INTERNET OF Things (IoT) represents the collection of sensing and actuation devices backed by the growing Internet infrastructure.¹ This creates a scenario unlike that of previous ubiquitous sensing. Those former approaches assumed a level of inherent compatibility and control over the sensors in their systems² and applications that used a manageable amount of raw sensor data. In the IoT, the number of available sensing and actuation devices has

grown rapidly in the last few years.³ In addition, ubiquitous connectivity and cloud storage have largely mitigated the primary research issues in pervasive sensing.

Now, the focus is on the application layer. IoT applications operate in a dynamic environment in which sensors and actuators move through an application's domain. The raw data in these applications go through several levels of processing to produce a high-level description of the

environment with discrete semantic states called *context*. Discrete context facilitates intuitive reasoning in exchange for raw data precision and can be reused across applications.

However, current context-aware IoT applications are still end-to-end implementations tightly coupled to the initial infrastructure. Each application maintains its own data and user interactions, which doesn't promote adaptation to the changing number and heterogeneity of I/O devices and IoT infrastructure.

We previously proposed smaller, simpler functional units called *context engines*, which provide intermediate computational steps toward an overall application.⁴ They create a flexible framework that promotes general-purpose machine learning (ML) and reduces processing redundancy and latency with minimal accuracy impact.

Another key issue for IoT applications is scalability: applications should scale well both with the number of inputs and to the available computing environment. In the IoT, context-aware applications often involve visualization (for example, user behavior tracking and vehicular safety⁵) or actuation (for example, smart spaces⁶). Scaling such applications means handling more users or covering a larger physical or virtual space in the presence of additional embedded devices for sensing or actuation, data aggregation, and computation. Here, we analyze how our modular approach to context-aware applications fundamentally improves scaling—the use of context engines minimizes overhead as the input data and number of computational nodes increase.

System Design

The current state of the art is multi-input, multi-output applications (see



Figure 1a). In this approach, black-box application implementations mask intermediate processing output from other applications, leading to computational redundancy.

Our context engine approach (see Figure 1b) is functionally equivalent to the state of the art. The context engines are hierarchical and multiple-input, single-output. Exposing the intermediate output reduces application complexity and redundancy and generates higher-level, intermediate context that applications can share. These improvements might come at the cost of accuracy,⁴ but careful application design can minimize this cost.

Smaller hierarchical context engines represent simpler data translation at the cost of more context engines per application. This promotes the use of general data transformation in each context engine—using ML, instead of application-specific code, to generate outputs.

We investigated matrix-based ML algorithms, which represent the data translation of each context engine. We implemented TESLA (Taylor Expanded Solar Analog Forecasting),⁷ a characteristic model-generation algorithm with $O(n^\alpha)$ complexity, where n is the number of inputs and α is the Taylor expansion order of the data translation. Such systems can be solved by regression methods—for example, least-squares estimation—that require at least m ($m \propto n^\alpha$) independent observations for training. This becomes time inefficient and space inefficient as α increases, but higher complexity enables a better model fit. Although we chose TESLA for our case study and scalability proofs, other ML algorithms (Bayesian networks, hidden Markov models, and neural networks⁸) are applicable.

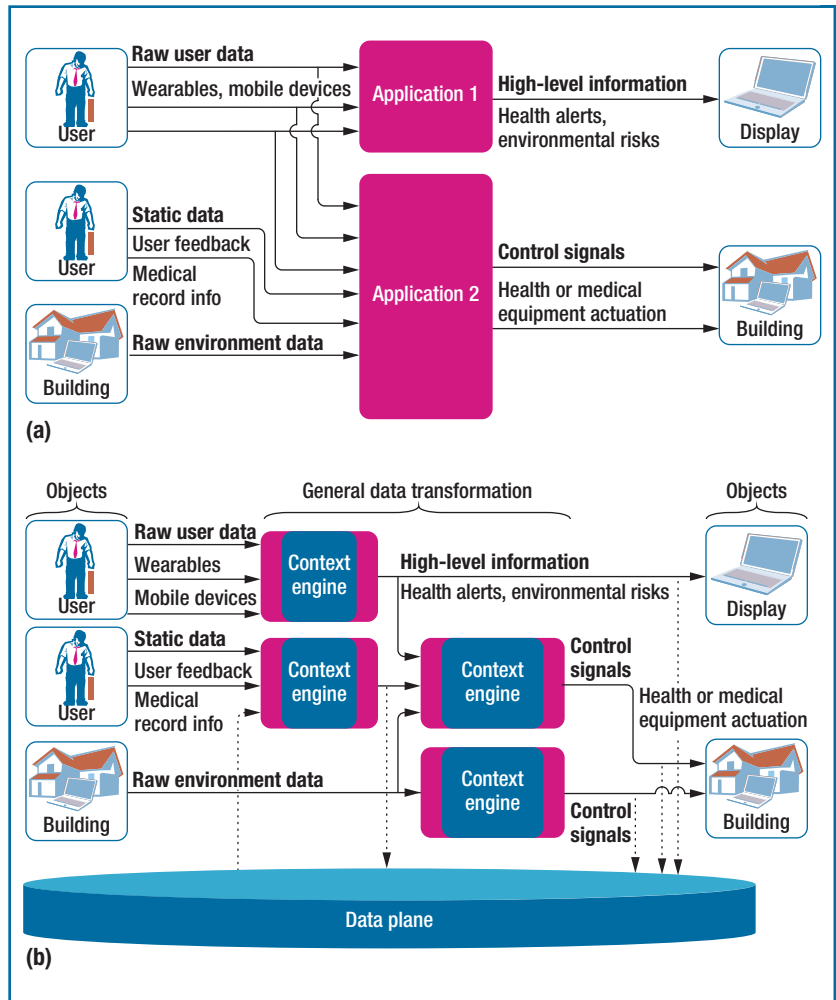


FIGURE 1. Two approaches to Internet-of-Things applications. (a) The state of the art is a monolithic application implementation. (b) In our approach, applications publish intermediate context for reuse. Multi-input, single-output functional units called context engines perform general statistical learning.⁴ Exposing the intermediate data reduces application complexity and redundancy.

The hierarchical approach raises questions about the complexity, latency, and accuracy of breaking apart a compact application. Here, we validate our approach by proving that dividing the processing of n inputs from a single context engine actually reduces application complexity.

For general context engine representations with n inputs and α complexity, the computational overhead

is n^α . We can divide one engine into two stages, forming multiple engines of an arbitrary number of inputs A and a resulting total complexity of

$$\frac{n}{A}A^\alpha + \left(\frac{n}{A}\right)^\alpha.$$

As the system becomes more modular, A decreases at the cost of increasing n/A (the number of context engines). Maximum division

occurs when $A = 2$ -input context engines. Analyzing the system, we find that when $\alpha > \log_2 3 \sim 1.6$ (any nonlinear context engine), maximum division minimizes computational complexity.⁴

Although maximum division isn't necessarily achievable for every application, any step toward increasing modularization of an application reduces computational complexity. For example, vehicle safety and user feedback applications can be modularized per vehicle or user, and smart spaces can be broken up into spatial domains. With fewer inputs per context engine, each unit represents a simpler transformation, and we expect smaller α . However, even if α remains high, we can prove reduced complexity. A more rigorous analysis of reducing overall complexity and truncation error appears in our previous work.⁴

System Scalability

We leverage and extend the definitions of scalability from distributed computing⁹ and IoT systems¹⁰ to quantify speedup changes under two conditions:

- the change in the number of computational nodes for an application (*strong scaling*) and
- the change in the amount of input data (*load scaling*).

We define speedup as

$$S(k, N) = \begin{cases} k, & 1 \leq k \leq N - 1 \\ 1, & k > N - 1 \end{cases},$$

where k is the number of cores and N is the number of inputs. Because we generalize the processing in each context engine to the equivalent computation, we can deal with functional order as a general term

representing the polynomial model's complexity and number of inputs.

For strong scaling, the hierarchical application behaves like a distributed system, using computational nodes as they become available. However, even with maximum division, speedup is ultimately capped. When more than $N - 1$ computational nodes become available, there are more free nodes than context engines. Some context engines can be reallocated to more capable nodes, but the system is overprovisioned and will scale as it expands.

Load scaling is particularly important for IoT applications because the increasing amount of data requires appropriate handling. Let's consider the general functional representation n^α for single-stage applications and $(n - 1) * 2^\alpha$ for a maximally divided set of context engines, which represents the overall application of n inputs and α complexity. We can address the addition of new inputs by

- increasing the number of inputs of a subset of the existing context engines or
- expanding the hierarchy with more low-input context engines.

The former option represents a move toward more-monolithic applications; the latter represents the modular approach: $(n - 1)$ 2-input context engines, each with α complexity.⁴ In the following, we consider this second option.

For every m additional inputs, the overall system complexity increases from n^α to $(n + m)^\alpha$. If we expand the hierarchy, assuming maximum division, for every m inputs, we add at most $m - 1$ context engines, increasing the complexity to $(m + n - 1) * 2^\alpha$. As m increases, the complexity

for increasing the number of inputs grows much faster than for expanding the hierarchy. Expanding the hierarchy achieves linear system growth with additional input, demonstrating perfect suitability for modular IoT applications with continuously expanding systems. Figure 2a compares this growth to the equivalent single-stage growth as m increases, with fixed n and $\alpha = 3$.

Figure 2b reports scalability in terms of communication overhead—the amount of data that must be transferred for ML training. Each context engine trains its ML algorithm to generate output context. The training phase of an n -input, α -complexity single-stage application requires n^α input samples and a corresponding n^α output samples from the source and sink devices to calculate the TESLA coefficients. Our approach with maximum division requires $2(n - 1) * 2^\alpha$, or $2^{\alpha+1} * (n - 1)$, input and output samples. Figure 2b illustrates the tradeoff between training multiple context engines with fewer inputs versus a single context engine with all inputs. Our approach achieves linear growth in data communication requirements, versus the single-stage approach's exponential growth.

The Case Study

We implemented a case study using our approach for a distributed, scalable application: the residential smart grid. Currently, utilities gather energy consumption from users through smart metering—a single-stage data-processing system. Utilities can also take into account user behavior to improve energy prediction's accuracy.¹¹ This additional context, obtained from wearable and house sensors, varies in source, data, accuracy, format, and so on. In

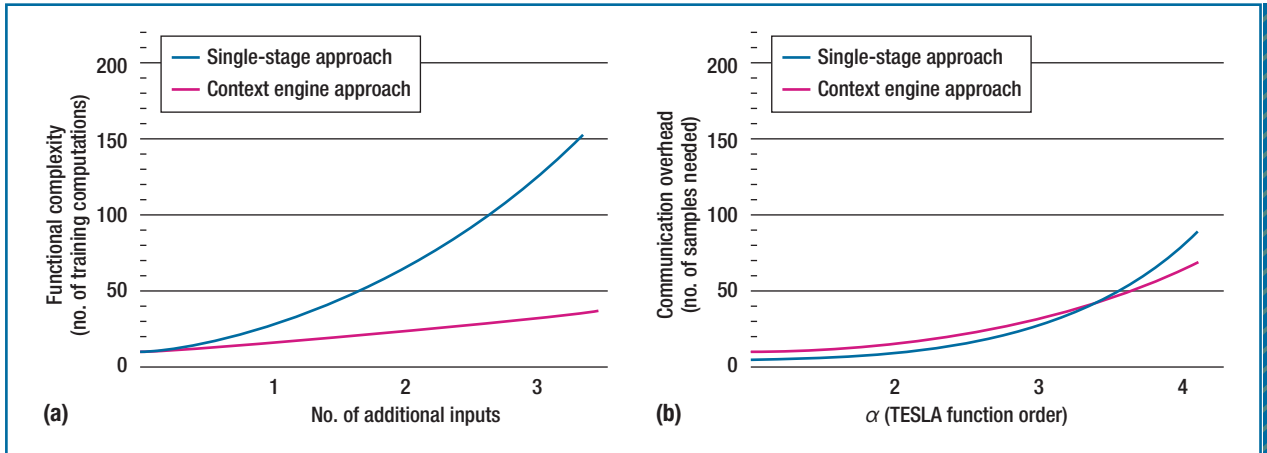


FIGURE 2. Comparing the scalability of the single-stage approach and our context engine approach. (a) Functional complexity. (b) Communication overhead. As you can see, our approach outperforms the single-stage approach. TESLA stands for Taylor Expanded Solar Analog.

current systems, the data would go directly to the utility, and the data heterogeneity would necessitate a re-designed application to provide energy prediction. This would significantly increase communication and processing overhead.

Our approach (see Figure 3) can provide the high-level context: energy prediction and flexibility in the next interval (enabling potential kWh savings by shutting down loads). Because the smart grid is naturally distributed, we can further break down data aggregation along the existing lines of power distribution: junction boxes, transformers, and substations, which have limited computational ability. The result is a multitier context-aware application that uses residential data to determine the flexibility of house loads and uses this generated context to determine the neighborhood flexibility.

Context Engine Setup

We began at the level of individual appliances in a house. Always-on appliances (for example, refrigerators) are less flexible than manually

triggered appliances (for example, kitchen and laundry appliances and lighting). Smart appliances with embedded systems are potential computation nodes.

We aimed to identify potential user interaction with an appliance and determine whether using this appliance was flexible at a given time. We then used these intermediate outputs to predict the appliance's energy use in the next interval and, consequently, the predicted energy flexibility.

We trained the intermediate and final outputs with the ground truth as follows:

- User interaction was a Boolean value derived from analyzing the energy or water traces to find the intervals during which the appliance was turned on.
- We derived binary appliance energy flexibilities from the distribution of use over time. This was unique to each house owing to differences in user behavior.

We used these first-stage context engine outputs to predict appliance

usage. Whereas researchers previously employed just the energy usage in time series to predict future intervals' output,¹² we leveraged the user context to better learn the profiles of manually triggered appliances.

Each house passes its flexibility prediction to the next tier: junction boxes or substations, which in turn feed aggregated flexibility prediction to the overarching utility. Aggregated flexibility is useful for quantifying the energy that can be saved; our approach identifies the individual loads that combine to provide this flexibility. This granularity of feedback control would enable the smart grid to perform automated, scalable residential demand and response.

Input and Intermediate Data

Our data was from the Pecan Street database (dataport.pecanstreet.org), a residential dataset that aggregates individual energy and water loads. Some houses included information about the number and type of occupants. We selected and replicated houses that fell into each category

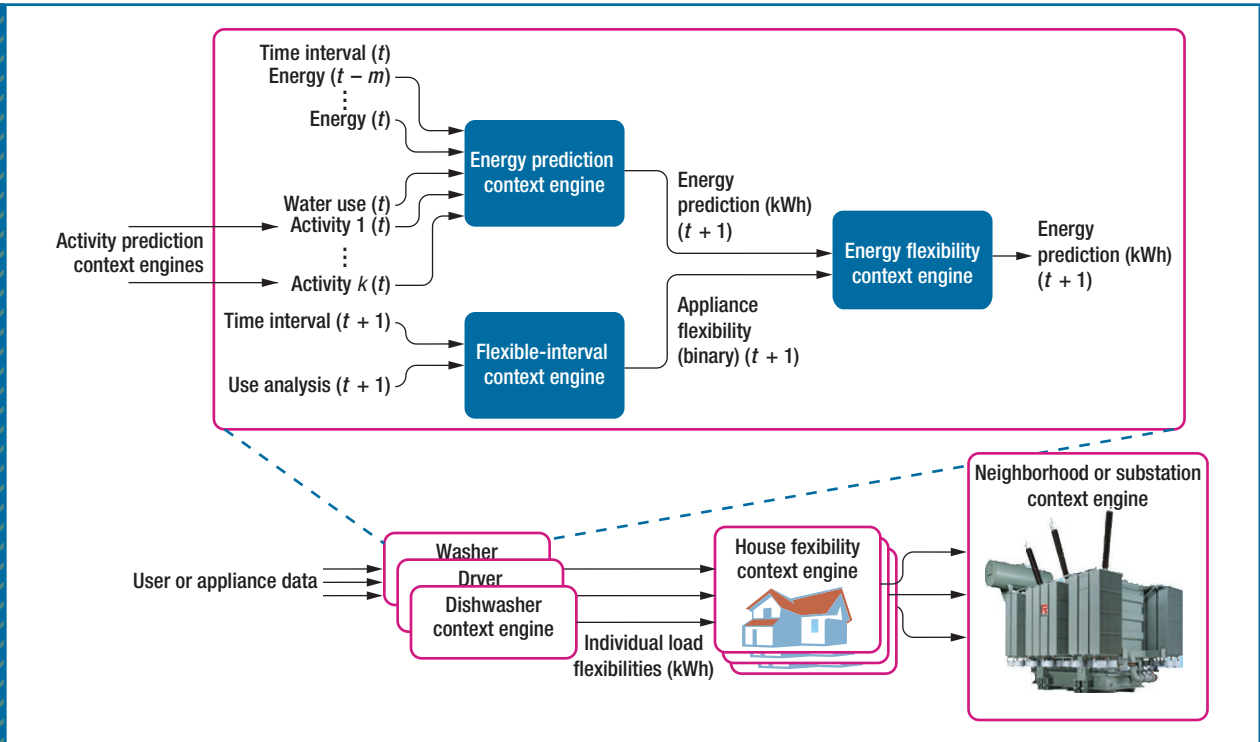


FIGURE 3. Implementation of a hierarchical approach to residential energy management, with individual homes providing the higher-level context in the place of raw data, aggregated and passed up the hierarchy. The output per house can vary, depending on the available types of sensors and actuators.

in Table 1 to construct a neighborhood with disparate amounts and types of data.

We trained the first-stage context engines with the ground truth for user interaction and binary flexibility. Because the Pecan Street database doesn't provide this information directly, we determined interaction by the start of operation of manually triggered appliances, using the traces. We derived flexibility by observing the aggregate appliance usage patterns—that is, the daily time range during which manual-appliance events were initiated. The flexible appliances we used were washing machines, clothes dryers, and dishwashers (owing to their flexible patterns^{13,14}), electric vehicles (owing

to flexible charging patterns in the Pecan Street dataset), and lighting (owing to varying light intensity¹²).

Figure 4 illustrates the usage pattern of House B's washing machine on Mondays, highlighting the aggregate number of instances at each time interval. The resulting clusters identified the windows of flexibility.

We defined other appliance flexibilities using related research and analysis of the traces themselves. For example, electric vehicles had three states: not plugged in, plugged in but not charging, and charging. The second and third states represented a time frame for flexible use.

Results

To evaluate our approach, we analyzed its accuracy and scalability.

Accuracy. First, we compared our approach's accuracy with that of the state-of-the-art single-stage approach: one node receiving all the raw traces from all houses and training over the aggregate flexibility. For our approach, the mean absolute error (MAE) for energy flexibility was 27.15 percent for House A, 14.23 percent for House B, 9.81 percent for House C, and 6.16 percent for House D. For all houses, the MAE was 14.34 percent. The MAE for the single-stage approach was 26.94 percent.

We originally demonstrated that modularization introduces truncation error,⁴ but this application illustrates the advantage of correlated input data. A third-order function using data from multiple houses

TABLE 1

The four house types retrieved for the case study, with their components.

House type	Percentage of neighborhood	Flexible appliances	Inflexible appliances	Additional room-specific appliances	Electric vehicles	Water-consuming appliances	Water flow
A	25	Yes	Yes	No	No	Yes	No
B	25	Yes	Yes	Yes	No	Yes	No
C	25	Yes	Yes	No	No	Yes	Yes
D	25	Yes	Yes	Yes	Yes	Yes	Yes

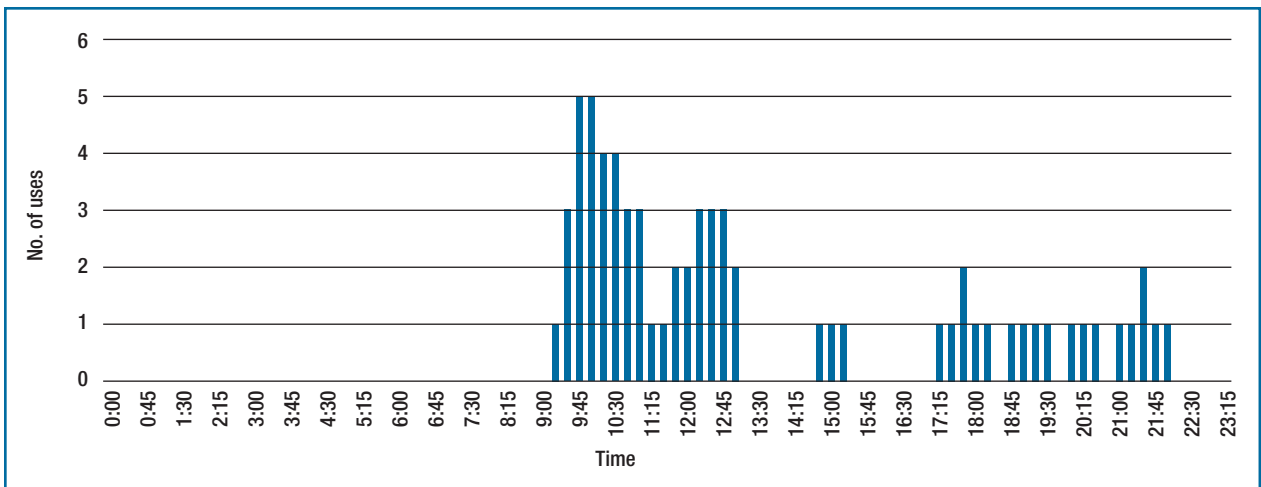


FIGURE 4. The aggregated instances of washing-machine use on Mondays, illustrating three clusters of varying flexibility.

can't achieve the same accuracy as an algorithm operating on a single house's data and producing only that house's energy and flexibility output. Theoretically, the larger application might eventually gain similar accuracy with enough training, even without intermediate context on which to train. However, the amount of required training data would far exceed the amount in the Pecan Street database. Furthermore, a model comprising a single house's data and energy already contains highly correlated inputs and output and thus provides high accuracy with relatively little training and a low order.

The single-stage approach collected all the input data from all the houses but scaled poorly because it required the more complex third-order computation in a single context engine to find overall energy consumption. Our approach handled the third-order computation closer to the edge—by the appliance-specific context engines—with fewer inputs and lower overall complexity. So, our approach at 1,000 inputs performed 96 times faster (just over 3 min) than the single-stage version (288 min).

Prediction error also decreased as more data sources became available (the number of inputs for $A < B < C$

< D). This highlights two important aspects of our modular approach:

- *Flexibility of input types.* Different users can provide a lot of data or a little data from various sources and still generate the same output through model generation.
- *Adaptation to changing or missing input data.* The remaining sources can still provide significant accuracy, although the change in error will be proportional to the correlation between the missing input and the output.

Unlike the single-stage application,

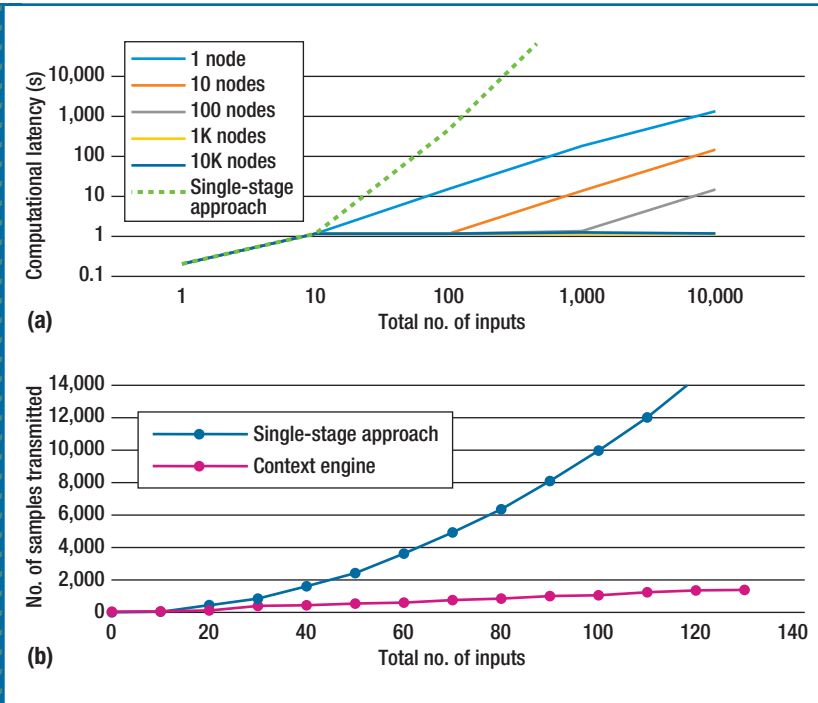


FIGURE 5. Evaluating scalability. (a) The scalability of the single-stage approach and our modular approach as the number of computational nodes and inputs grew. (b) The two approaches’ communication overhead as the number of inputs grew.

in which a missing input affects overall system accuracy, our approach limits the primary accuracy loss to the context engines that directly or indirectly use the missing input.

Scalability. We evaluated our application’s latency as the number of computational nodes and inputs grew. Figure 5a illustrates the scalability on a log–log plot. To investigate a viable implementation of our approach, we maintained a ratio of 10 inputs per context engine. This allowed a balanced modular multiple-input, single-output implementation. It also more accurately represented the distribution of sensors and computation from the lower levels (aggregating data in a house) to the upper levels (aggregating a street or neighborhood in a

distributor or substation) of the real smart-grid hierarchy.

Both applications initially exhibited similar latency when limited to the same number of nodes. However, our approach distributed the processing to more nodes as they became available, demonstrating linear growth with the number of inputs as long as enough nodes were available. When no more nodes were available, a subset of computation had to be serialized. The single-stage approach, without this benefit, scaled exponentially with the number of inputs.

We then investigated how communication scaled in the two approaches in terms of the amount of training data required—data that had to be accrued before the context engines could generate output. Both approaches required $O(n^\alpha)$ training

samples per context engine, but the single-stage approach grew exponentially with the number of inputs. Because our approach limited n and instead grew the number of context engines and the hierarchy, the required amount of training data scaled linearly. The single-stage approach had a single $O(n^3)$ context engine; as n increased, the required amount of training data grew exponentially with n (see Figure 5b).

Our ongoing and future research involves expanding the context engine and improving its applicability. Our initial context engine results already demonstrate a significant improvement in predicting residential energy usage and flexibility. However, although we learn on all available inputs, we don’t necessarily limit ourselves to only the best (most correlated) inputs. Preliminary correlation analysis has already shown the ability to mitigate accuracy losses by eliminating only the least significant inputs. Furthermore, limiting the number of inputs can improve each context engine’s latency. Formalizing correlation analysis and automatically selecting the most appropriate inputs from a pool of available sources are useful for context-aware applications operating in an environment of changing data. This approach can improve efficiency as well as replace or update the existing algorithm as more accurate or reliable sources enter or leave the system.

Similarly, the context engine is currently employed with a number of machine-learning algorithms. Some of these algorithms are better suited than the others for a particular application, and they all incur costs in computational overhead,



JAGANNATHAN VENKATESH is a XXXXX at Google. His research interests are energy efficiency, automation, validation of renewable-energy systems and smart grids, and power- and context-aware computing in embedded and mobile systems. Venkatesh received a PhD from the Computer Science and Engineering Department at the University of California, San Diego. Contact him at jvenkate@eng.ucsd.edu.



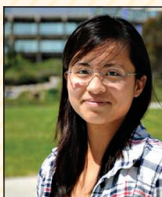
ALPER S. AKYÜREK is a PhD candidate in electrical and computer engineering at the University of California, San Diego. His research interest is the control and optimization of energy efficiency in the smart grid. Akyürek received an MS in electrical and electronics engineering from the Middle East Technical University. Contact him at aakyurek@eng.ucsd.edu.



BARIŞ AKŞANLI is an assistant professor in San Diego State University's Electrical and Computer Engineering Department. His research interests include energy-efficient cyber-physical systems, human-behavior modeling for the Internet of Things, and big data for energy-efficient large-scale systems. Akşanlı received a PhD in computer science from the University of California, San Diego. He has won an IDEA (Internet2 Driving Exemplary Applications) award and a Spontaneous Recognition award from Intel. Contact him at baksanlı@sdsu.edu.



TAJANA Š. ROSING is a professor of computer science, a holder of the Fratamico Endowed Chair, and a director of the System Energy Efficiency Lab at the University of California, San Diego. Her research interests are energy-efficient computing and embedded and distributed systems. Rosing received a PhD in electrical engineering from Stanford University. Contact her at tajana@eng.ucsd.edu.




CHRISTINE S. CHAN is a PhD student in the Electrical and Computer Engineering Department at the University of California, San Diego (UCSD). Her main research interest is in optimizing energy-efficient systems according to both the human context and machine context. Chan received an MS in computer engineering from UCSD. Contact her at csc019@eng.ucsd.edu.



accuracy, and latency. This can be leveraged to define a multi-optimization problem for context-aware computing, trading off computational

complexity, latency, and accuracy. The ability to quantify these three characteristics and, more important, optimize for each is crucial to

the real-life context-aware applications in which context engines are used: smart spaces, grid automation, and the IoT. The result is a

functional unit that can optimize for the most correlated data, train using the best available machine-learning technique, and generate the outputs required, as well as adapt to environment- and user-imposed performance constraints. 

References

1. C. Perera et al., "Context Aware Computing for the Internet of Things: A Survey," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 1, 2013, pp. 414–454.
2. M. Friedewald and O. Raabe, "Ubiquitous Computing: An Overview of Technology Impacts," *Telematics and Informatics*, vol. 28, no. 2, 2011, pp. 55–65.
3. J. Hammer and T. Yan, "Poster: A Virtual Sensing Framework for Mobile Phones," *Proc. 12th Ann. Int'l Conf. Mobile Systems, Applications, and Services (MobiSys 14)*, 2014, p. 371.
4. J. Venkatesh et al., "A Modular Approach to Context-Aware IoT Applications," *Proc. 1st Int'l Conf. Internet-of-Things Design and Implementation (IoTDI 16)*, 2016, pp. 235–240.
5. K. Lee et al., "AMC: Verifying User Interface Properties for Vehicular Applications," *Proc. 11th Ann. Int'l Conf. Mobile Systems, Applications, and Services (MobiSys 13)*, 2013, pp. 1–12.
6. H. Chen, T. Finin, and A. Joshi, "An Ontology for Context-Aware Pervasive Environments," *Knowledge Eng. Rev.*, vol. 18, no. 3, 2004, pp. 197–207.
7. B.O. Akyürek et al., "TESLA: Taylor Expanded Solar Analog Forecasting," *Proc. 2014 IEEE Int'l Conf. Smart Grid Communications (SmartGridComm 14)*, 2014, pp. 127–132.
8. J. Gubbi et al., "Internet of Things (IoT): A Vision, Architectural Elements, and Future Directions," *Future Generation Computer Systems*, vol. 29, no. 7, 2013, pp. 1645–1660.
9. P. Jogalekar and M. Woodside, "Evaluating the Scalability of Distributed Systems," *IEEE Trans. Parallel and Distributed Systems*, vol. 11, no. 6, 2000, pp. 589–603.
10. D. Uckelmann, M. Harrison, and F. Michahelles, "An Architectural Approach toward the Future Internet of Things," *Architecting the Internet of Things*, Springer, 2011, pp. 1–24.
11. B. Akşanlı, A.S. Akyürek, and T.Ş. Rosing, "User Behavior Modeling for Estimating Residential Energy Consumption," *Smart City 360°*, Springer, 2015, pp. 348–361.
12. J. Venkatesh et al., "HomeSim: Comprehensive, Smart, Residential Simulation and Scheduling," *Proc. 2013 Int'l Green Computing Conf. (IGCC 13)*, 2013, pp. 1–8.
13. N. Banerjee, S. Rollins, and K. Moran, "Automating Energy Management in Green Homes," *Proc. 2nd ACM SIGCOMM Workshop Home Networks (HomeNets 11)*, 2011, pp. 19–24.
14. T. Jamasb and M.G. Pollitt, *The Future of Electricity Demand: Customers, Citizens and Loads*, Cambridge Univ. Press, 2011.



Take the CS Library wherever you go!

 IEEE Computer Society magazines and Transactions are now available to subscribers in the portable ePub format.

Just download the articles from the IEEE Computer Society Digital Library, and you can read them on any device that supports ePub. For more information, including a list of compatible devices, visit

www.computer.org/epub

 **IEEE**  **IEEE computer society**

myCS Read your subscriptions through the myCS publications portal at <http://mycs.computer.org>