



Fog Orchestration for Internet of Things Services

Large-scale Internet of Things (IoT) services such as healthcare, smart cities, and marine monitoring are pervasive in cyber-physical environments. These complex IoT services are increasingly composed of sensors, devices, and compute resources within fog computing infrastructures. Orchestrating such applications can simplify maintenance and enhance data security and system reliability. However, efficiently dealing with these services' dynamic variations and transient operational behavior is a crucial challenge. This article provides an overview of the core issues, challenges, and future research directions in fog-enabled orchestration for IoT services.

Zhenyu Wen
University of Edinburgh

Renyu Yang
Beihang University

Peter Garraghan
University of Lancaster

Tao Lin
École Polytechnique Fédérale de Lausanne

Jie Xu
University of Leeds

Michael Rovatsos
University of Edinburgh

Cyber-physical environments encompass physical and virtual components capable of interfacing and interacting with existing network infrastructure, enabling novel applications in areas such as smart cities, intelligent transportation, and autonomous vehicles. The explosive growth in data generation has led to a focus in both research and industry on issues related to effectively extracting insights from such data to assist in the design of cyber-physical systems. Internet of Things (IoT) services typically comprise a set of software components running in different locations and connected through dynamic networks (such as 4G, wireless LAN, and the Internet). Systems such as data centers and wireless sensor networks (WSNs) underpin the data storage and compute resources required for operating these components.

Fog computing extends cloud computing by moving computation and data

storage to the edge of the network, allowing for reduced latency and response delay jitter for applications.^{1,2} These characteristics are particularly important for latency-sensitive applications, such as gaming and video streaming. In the IoT environment, existing applications and physical devices can be leveraged as fundamental appliances and composed in a mashup style to control development cost and maintenance pressure. Orchestration is a key concept within distributed systems, enabling the alignment of deployed applications with users' business interests. We propose a fog orchestrator, to provide the centralized arrangement of the resource pool, mapping applications with specific requests and providing an automated workflow to physical resources (deployment and scheduling); workload execution management with runtime quality of service (QoS) control; and time-efficient directive generation to manipulate specific objects.

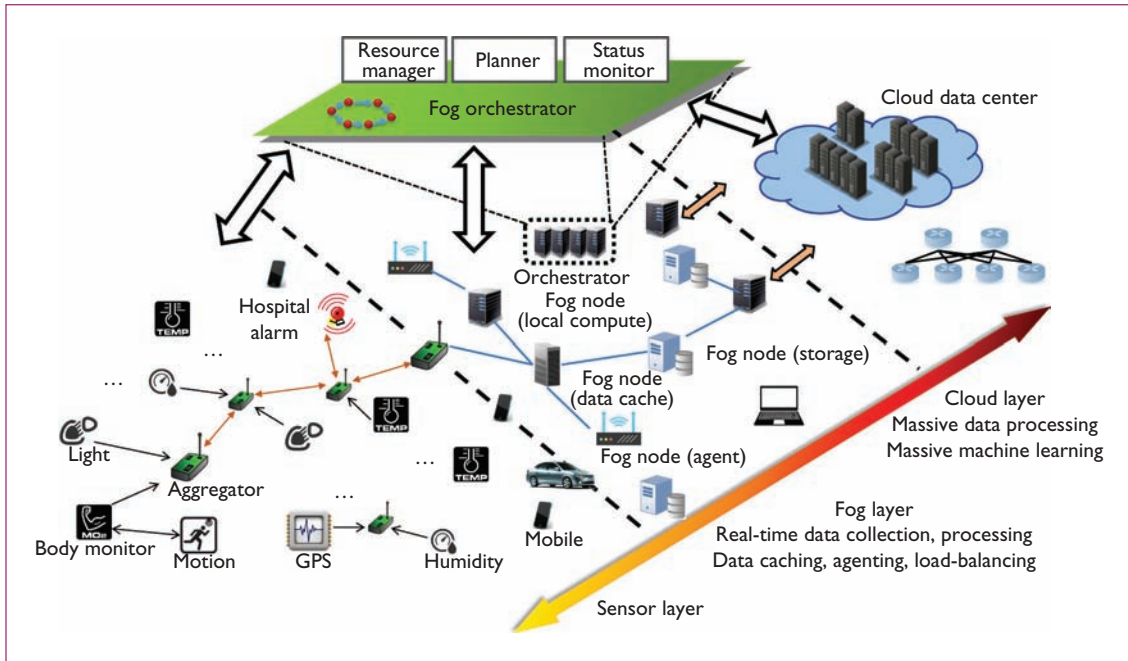


Figure 1. An orchestration scenario for an e-health service. Different Internet of Things (IoT) appliances (diverse types of sensors and fog nodes) are orchestrated as a workflow across all layers of the fog architecture. The fog orchestrator acts as a controller deployed on a workstation or cloud data center and across all organization layers based on global information.

Motivating Example

Smart cities aim to enhance the quality of urban life by using technology to improve the efficiency of services to meet residents' needs. Achieving this goal requires integrating multiple information and communication technologies in a secure, efficient, and reliable way to manage city facilities effectively. Such systems consist of two major components:

- sensors integrated with real-time monitoring systems, and
- applications integrated with the collected sensor (or device) data.

Currently, IoT services are rudimentary, and only integrate with specific sensor types. This results from the lack of existing universally agreed standards and protocols for IoT device communication, and represents a challenge to achieving a global ecosystem of interconnected things.

To address this problem, an alternative approach is to use an IoT service orchestration system to determine and select the best IoT appliances for dynamic composition of holistic workflows for more complex functions. As Figure 1 shows, our proposed orchestrator manages

all layers of an IoT ecosystem to integrate different standalone appliances or service modules into a complex topology. The orchestrator's primary responsibility is selecting resources and deploying the overall service workflow according to data security, reliability, and system efficiency requirements. It's centralized only at a conceptual level and can be implemented in a distributed and fault-tolerant fashion, without introducing a single point of failure.

An appropriate combination of these standalone IoT appliances can facilitate more advanced functionality, helping to reduce costs and improve the user experience. For example, mobile health systems can provide remote monitoring, real-time data analysis, emergency warning, and so on. Data collected from wearable sensors that monitor patient vitals can be continuously sent to data aggregators and, if the system detects abnormal behavior, it can immediately notify hospital personnel, who can take appropriate measures.

Although such functionality can be developed within a standalone application, this provides limited scalability and reliability. The implementation of new features leads to increased development efforts and risk of

creating a monolithic application incapable of scaling effectively due to conflicting resource requirements for effective operation. In addition, increased application complexity leads to tedious, time-consuming debugging. The use of orchestration allows for more flexible formation of application functionality to scale and reduces the probability of failure correlation between application components.

Fog-Enabled IoT Application

Traditional Web-based service applications are deployed on servers within cloud data centers that are accessed by end devices such as tablets, smartphones, and desktop PCs. In contrast, IoT applications deployed within fog computing systems consist of the cloud, fog node, and “things,” as Figure 1 shows. In this context, a fog node is defined as equipment or middleware and is served as an agent that collects data from a set of sensors. This data is then transmitted to a centralized computing system that locally caches data and performs load balancing. Things include sensors and devices with built-in sensors. Similar to Web-based service applications, the cloud provisions centralized resource pools (compute, storage) to analyze collected data and automatically trigger decisions based on a predefined system logic. The most significant difference, however, is the use of fog nodes that transmit data to cloud data centers. For example, most wearable sensor data is collected and preprocessed by smartphones or adjacent workstations. This can either significantly reduce transmission rates or improve their reliability.

Web-based and IoT applications differ in several other ways. First, IoT communication is performed using a hybrid centralized-decentralized approach depending on context. Most message exchanges between sensors or between a sensor and the cloud are performed using fog nodes. Purely centralized environments are ill-suited for applications that have soft and hard real-time requirements. For example, neighboring smart vehicles need to transfer data between other vehicles and traffic infrastructure to prevent collisions. Such a system was piloted in New York City using Wi-Fi to enable real-time interactions to assist drivers in navigating congestion and to communicate with pedestrians or oncoming vehicles.³ Furthermore, given the huge number of

connected devices, the data volume generated and exchanged over an IoT network is predicted to become many orders of magnitude greater than that of conventional Web-based services, resulting in significant scalability challenges.

Interoperability is another aspect where Web-based and IoT applications diverge. Software-defined networking technologies enable the decoupling of software control and heterogeneous hardware operations. This approach provides an opportunity to dynamically achieve different quality levels for different IoT applications in heterogeneous environments.⁴ Moreover, application-level interoperability benefits from Web technologies, such as the RESTful architecture, that provide a high level of interoperability. Using these technologies, an abundance of programming APIs can be distributed across entire fog domains and utilized to increase the flexibility of loosely coupled management.⁵ Lightweight APIs, such as RESTful interfaces, result in agile development and simplified orchestration with enhanced scalability when composing complex distributed workflows.

A third aspect is reliability. Physical systems make up a significant part of IoT applications, thus the assumptions that can be made regarding fault and failure modes are weaker than those for Web-based applications. IoT applications experience crash and timing failures stemming from low-sensor battery power, high network latency, environmental damage, and so on. Furthermore, the uncertainty of potentially unstable and mobile system components increases difficulties in predicting and capturing system operation. Therefore, an IoT application workflow’s reliability needs to be measured and enhanced in more elaborate ways.

IoT Application Orchestration Challenges

As we’ve shown, existing IoT applications are diverse in terms of reliability, scalability, and security. Diversity among fog nodes is a key issue; location, configuration, and served functionalities of fog nodes all dramatically increase this diversity. This raises an interesting research challenge – namely, how to optimize the process of determining and selecting the best IoT appliances and fog components to compose an application workflow while meeting nonfunctional requirements such as security, network latency, and QoS.

Scale and Complexity

As more IoT manufacturers develop heterogeneous sensors and smart devices, selecting optimal components becomes increasingly complicated when considering customized hardware configurations and personalized requirements. For example, some applications can only operate with specific hardware architectures (for example, ARM or Intel) or operating systems, whereas applications with high security requirements might require specific hardware and protocols to function. Orchestration needs not only to cater to such functional requirements, it must do so in the face of increasingly larger workflows that change dynamically. The orchestrator must determine whether the assembled systems, comprised of cloud resources, sensors, and fog nodes, coupled with geographic distributions and constraints are capable of provisioning complex services correctly and efficiently. In particular, the orchestrator must be able to automatically predict, detect, and resolve issues pertaining to scalability bottlenecks that could arise from increased application scale.

Security Criticality

In the IoT environment, multiple sensors, computer chips, and communication devices are integrated to enable the overall communication. A specific service might be composed of a multitude of components, each deployed within different geographic locations, resulting in an increased attack vector of such objects. Fog nodes are particularly vulnerable to such attacks, especially in the context of network-enabled IoT systems, where attack vectors can include human-caused sabotage of network infrastructure, malicious programs provoking data leakage, or even physical access to devices. A large body of research focuses on cryptography and authentication toward enhancing network security to protect against cyberattacks.⁶ Furthermore, in systems comprising hundreds of thousands of electronic devices, it's crucial to effectively and accurately evaluate the security and measure risks to present a holistic security and risk assessment.⁷ This becomes challenging when workflows are capable of changing and adapting at runtime. For these reasons, approaches that can dynamically evaluate the security of dynamic IoT application orchestration will become increasingly critical for secure data placement and processing.

Dynamicity

Another significant characteristic and challenge for IoT services is their ability to evolve and dynamically change their workflow composition. This problem, in the context of software upgrades through fog nodes or the frequent join-leave behavior of network objects, will change the internal properties and performance, potentially altering the overall workflow execution pattern. Moreover, handheld devices inevitably suffer from software and hardware aging, which will invariably result in changing workflow behavior and its device properties (for example, low-battery devices will degrade the data transmission rate). Finally, application performance will change owing to their transient and/or short-lived behavior within the system, including spikes in resource consumption or data generation. This leads to a strong requirement for automatic and intelligent reconfiguration of the topological structure and assigned resources within the workflow, and importantly, that of fog nodes.

Fault Diagnosis and Tolerance

Scaling a fog system increases the probability of failure. Some rare software bugs or hardware faults that don't manifest at small scale or in testing environments, such as stragglers,⁸ can have a debilitating effect on system performance and reliability. At the scale, heterogeneity, and complexity we're anticipating, different fault combinations will likely occur.⁹ To address these system failures, developers should incorporate redundant replications and user-transparent, fault-tolerant deployment and execution techniques in orchestration design.

Key Research Directions

Several research directions are key to tackling the challenges we've outlined. Within lifecycle management, these include optimal selection and placement in the deployment stage; dynamic QoS monitoring and providing guarantees at runtime through incremental processing and replanning; and big data-driven analytics and optimization approaches that leverage data mining to improve orchestration quality and accelerate optimization for problem solving.

Component Selection and Placement

The recent trend in composing cloud applications involves connecting heterogeneous services

deployed across multiple data centers. Such a distributed deployment helps improve IoT application reliability and performance within fog computing environments. As mentioned earlier, it also exposes appliances to new security risks and network uncertainty. Ensuring high levels of dependability for workflows composed by multiple systems is a considerable challenge. Numerous efforts have focused on QoS-aware composition of native virtual machine-based cloud application components, but neglect the proliferation of uncertain execution and security risks among interactive and interdependent components within IoT services.^{10,11}

Parallel computation algorithm. Optimization algorithms or graph-based approaches are typically time- and resource-consuming when applied on a large scale, and necessitate parallel approaches to accelerate the optimization process. Recent work provides possible solutions to leverage an in-memory computing framework to execute tasks in a cloud infrastructure in parallel.¹² However, realizing dynamic graph generation and partitioning at runtime to adapt to the shifting space of possible solutions stemming from the scale and dynamicity of IoT components remains an unsolved problem.

Late calibration. To ensure near-real-time intervention during IoT application development, one approach is to use correction mechanisms that could be applied even when suboptimal solutions are deployed initially. For example, in some cases, if the orchestrator finds a candidate solution that approximately satisfies the reliability and data transmission requirements, it can temporarily suspend the search for further optimal solutions. At runtime, the orchestrator can then continue to improve decision results with new information and a reevaluation of constraints, and use task- and data-migration approaches to realize workflow redeployment.

Dynamic Orchestration with Runtime QoS

Apart from the initial placement, all workflow components dynamically change in response to internal transformations or abnormal system behavior. IoT applications are exposed to uncertain environments where execution variations are commonplace. Because of the degradation of consumable devices and sensors, capabilities such as security and reliability that initially

were guaranteed will vary, resulting in the initial workflow being no longer optimal or even totally invalid. Furthermore, the structural topology might change according to the task-execution progress (that is, a computation task is finished or evicted) or will be affected by the execution environment's evolution. Abnormalities might occur owing to the variability of combinations of hardware and software crashes, or data skew across different management domains of devices due to abnormal data and request bursting. This will result in unbalanced data communication and subsequent reduction of application reliability. Therefore, dynamically orchestrating task execution and resource reallocation is essential.

QoS-aware control and monitoring. To capture the dynamic evolution and variables (such as dynamic evolution, state transition, and new IoT operations), we should predefine the quantitative criteria and measuring approach of dynamic QoS thresholds in terms of latency, availability, throughput, and so on. These thresholds usually dictate upper and lower bounds on the metrics as desired at runtime. In our setting, complex QoS information processing methods such as hyper-scale matrix update and calculation would lead to many scalability issues.

Event streaming and messaging. We can depict performance metric variables or significant state transitions as system events, and process event streaming in the orchestration framework through an event messaging bus, real-time publish-subscribe mechanism, or high-throughput messaging systems (such as Apache Kafka). This would significantly reduce communication overhead and ensuring responsiveness. Subsequent actions automatically could be triggered and driven by a cloud engine (such as the Amazon Lambda service).

Incremental Computation in Orchestration

IoT services can often be choreographed through workflow or task graphs to assemble different IoT applications. In some domains, the orchestration is supplied with a plethora of candidate devices with different geographical locations and attributes. In some cases, orchestration would typically be considered too computationally intensive, as it's extremely time-consuming to perform operations including prefiltering, candidate selection,

and combination calculation while considering all specified constraints and objectives. Static models and methods become viable when the application workload and parallel tasks are known at design time. In contrast, in the presence of variations and disturbances, orchestration methods typically rely on incremental scheduling at runtime (rather than straightforward complete recalculation by rerunning static methods) to decrease unnecessary computation and minimize schedule makespan.

Proactive recognition. Localized regions of self-updates become ubiquitous within fog environments. The orchestrator should record staged states and data produced by fog components periodically or in an event-based manner. This information will form a set of time series of graphs and facilitate the analysis and proactive recognition of anomalous events to dynamically determine such hotspots.¹³ The data and event streams should be efficiently transmitted among fog components, so system outage, appliance failure, or load spikes will rapidly feed back to the central orchestrator for decision making.

Incremental design and implementation. Based on the time series of graphs, researchers or developers should comprehensively study the similarities and dependencies between successive graph snapshots to determine the feasibility of incremental computation. Approaches such as memorization, self-adjusting computation, and semantic analysis could cache and reuse portions of dynamic dependency graphs to avoid unnecessary re-computation in the event of input changes. Intermediate data or results should be inherited as far as possible, and the allocated resources that have been allocated to the tasks should also be reused rather than be requested repeatedly. Through graph analysis, operators can determine which subgraphs change within the whole topology by automating subgraph partitioning and matching to significantly reduce overall execution time.

Systematic Data-Driven Optimization

IoT applications include numerous geographically distributed devices that produce multidimensional, high-volume data requiring different levels of real-time analytics and data aggregation. Therefore, data-driven optimization and planning should have a place in the orchestration of complex IoT services.

As researchers or developers select and distribute applications across different layers in the fog environment, they should consider the optimization of all overlapping, interconnected layers. The orchestrator has a global view of all resource abstractions, from edge resources on the mobile side to compute and storage resources on the cloud data center side. Pipelining the stream of data processing and the database services within the same network domain could reduce data transmission. Similar to the data-locality principle, we can also distribute or reschedule the computation tasks of fog nodes near the sensors rather than frequently move data, thereby reducing latency. Another potential optimization is to customize data-relevant parameters such as the data-generation rate or data-compression ratio to adapt to the performance and assigned resources to strike a balance between data quality and specified response-time targets.

A major challenge is that decision operators are still computationally time consuming. To tackle this problem, online machine learning can provision several online training (such as classification and clustering) and prediction models to capture the constant evolutionary behavior of each system element, producing time series of trends to intelligently predict the required system resource usage, failure occurrence, and straggler compute tasks, all of which can be learned from historical data and a history-based optimization (HBO) procedure. Researchers or developers should investigate these smart techniques, with corresponding heuristics applied in an existing decision-making framework to create a continuous feedback loop. Cloud machine learning offers analysts a set of data exploration tools and a variety of choices for using machine learning models and algorithms.¹⁴

Early Experience and Initial Results

Based on the design philosophy and methods discussed, we propose a framework that can efficiently orchestrate fog computing environments. As Figure 2 demonstrates, to enable planning and adaptive optimization, we attempted to manage the composition of applications in parallel under a broad range of constraints. We implemented a parallel genetic algorithm-based framework (GA-Par) on Spark to handle orchestration scenarios involving the composition of a

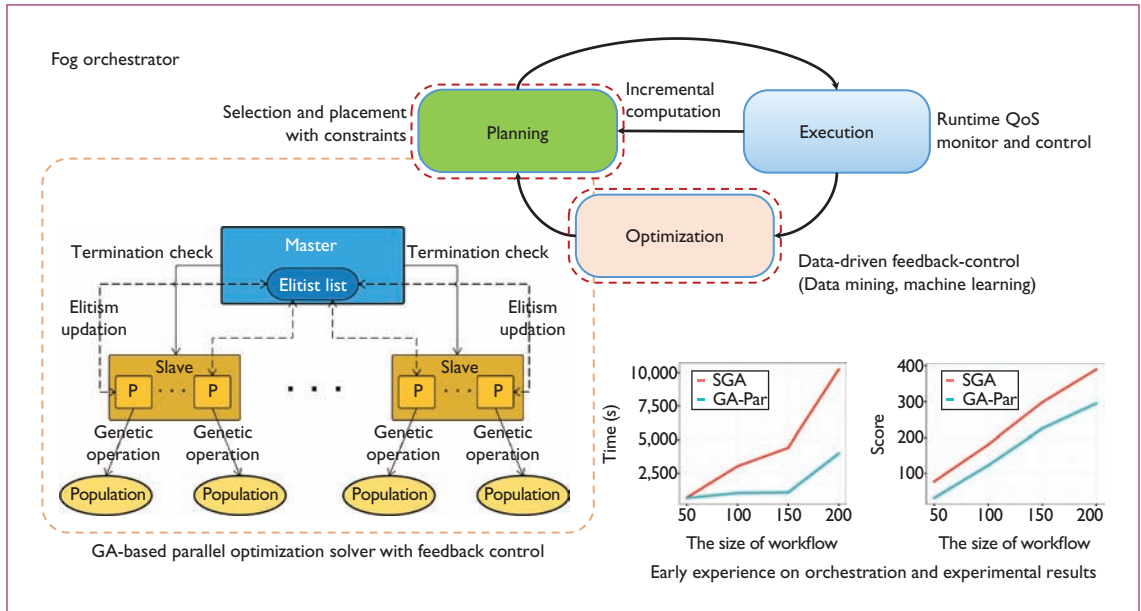


Figure 2. Main functional elements of our fog orchestrator. The planning element is responsible for selection and placement, runtime monitoring and control during execution, and data-driven decision optimization. Our parallel genetic algorithm (GA) solver accelerates the handling of optimization issues raised in the planning and optimization phase. Initial results demonstrate that the proposed approach (GA-Par) can outperform a standalone genetic algorithm (SGA) in terms of both time and quality (QoS = quality of service).


large set of IoT applications. More specifically, in our GA-based algorithm, each chromosome represents a solution of the composed workflow and each chromosome's gene segments represent the IoT applications. We normalize the utility of security and network QoS of IoT appliances into an objective fitness function within GA-Par to minimize the security risks and performance degradation.

Specifically, to strike a balance between accuracy and time efficiency, we separate the total individual population into parallel compute partitions dispersed over different compute nodes. To maximize parallelism, we set up and adjust the partition configuration dynamically to make partitions fully parallelized while considering data shuffling and communication cost with the topology change. To guarantee that we can gradually obtain optimal results, we dynamically merge several partitions into a single partition and then repartition it based on runtime status and monitored QoS. Furthermore, we can maintain the quality of each solution generation by applying an elitist method, where each partition's local elite results are collected and synthesized into a global elite.

The centralized GA-Par master aggregates the full information at the end of each iteration, and then broadcasts the list to all partitions to increase the probability of finding a globally optimal solution. To address data skew issues, we also conduct a joint data-compute optimization to repartition the data and reschedule computation tasks. We performed some initial experiments on 30 servers hosted on Amazon Web Services as the cloud data center for the fog environment. Each server is hosted as an r3.2xlarge instance with 2.5-GHz Intel Xeon E5-2670v2 CPUs, 61-Gbyte RAM, and 160-Gbyte disk storage.

Simulated data can help us illustrate the effectiveness of composition given IoT requirements. For this, we randomly select four types of orchestration graphs with 50, 100, 150, and 200 workflow nodes, respectively. For each node within a workflow, we stochastically prepare 100 available IoT appliances as simulated agents. The security levels and network QoS levels are randomly assigned to each candidate agent. We compare our GA-Par with a standalone genetic algorithm (SGA), using metrics quality, execution time, and fitness score (with

lower values indicating better results) to evaluate them. As Figure 2 shows, GA-Par outperforms SGA. GA-Par time consumption is nearly 50 percent of that of SGA, while the quality of appliance selection in GA-Par is always at least 30 percent higher than that of SGA. However, our approach's scalability is still slightly affected by increasing numbers of components and requests, indicating that we still need to explore opportunities for incremental replanning and online tuning to improve both time efficiency and effectiveness of IoT orchestration.

Most recent research related to fog computing explores architectures within massive infrastructures.¹ Although such work advances our understanding of the possible computing architectures and challenges of new computing paradigms, there are presently no studies of composability and concrete methodologies for developing orchestration systems that support composition in the development of novel IoT applications. Our prototypical orchestration system exploits some of the most promising mechanisms to tackle these challenges. In future work, we will evaluate the orchestration effectiveness in the massive-scale production system and further improve the time efficiency by incremental re-planning. 

Acknowledgments

This work is supported by the China NKR&D Program (2016YFB1000103), National 863 Program (2015AA01A202), Natural Science Foundation of China (61421003), and the European Commission's FP7 Programme (FP7/2007-2013) through grant 600854. Renyu Yang is the corresponding author for this article.

References

1. F. Bonomi et al., "Fog Computing and Its Role in the Internet of Things," *Proc. Mobile Cloud Computing*, 2012, pp. 13–16.
2. S. Yi, C. Li, and Q. Li, "A Survey of Fog Computing: Concepts, Applications, and Issues," *Proc. Workshop Mobile Big Data*, 2015, pp. 37–42.
3. M.D. Dikaikos et al., "Location-Aware Services over Vehicular Ad Hoc Networks Using Car-to-Car Communication," *IEEE J. Selected Areas in Comm.*, vol. 25, no. 8, 2007, pp. 1590–1602.
4. Z. Qin et al., "A Software Defined Networking Architecture for the Internet-of-Things," *Proc. IEEE Network*

Operations and Management Symp., 2014; doi:10.1109/NOMS.2014.6838365.

5. S. Nastic et al., "Provisioning Software-Defined IoT Cloud Systems," *Proc. Int'l Conf. Future Internet of Things and Cloud*, 2014, pp. 288–295.
6. R. Roman, P. Najera, and J. Lopez, "Securing the Internet of Things," *Computer*, vol. 44, no. 9, 2011, pp. 51–58.
7. A. Riahi et al., "A Systemic Approach for IoT Security," *Proc. Int'l Conf. Distributed Computing Systems*, 2013, pp. 351–355.
8. P. Garraghan et al., "Straggler Root-Cause and Impact Analysis for Massive-Scale Virtualized Cloud Datacenters," *IEEE Trans. Services Computing*, preprint, 20 Sept. 2016; doi:10.1109/TSC.2016.2611578.
9. R. Yang et al., "Reliable Computing Service in Massive-Scale Systems Through Rapid Low-Cost Failover," *IEEE Trans. Services Computing*, preprint, 21 Mar. 2016; doi:10.1109/TSC.2016.2544313.
10. Z. Zheng, Y. Zhang, and M.R. Lyu, "Investigating QoS of Real-World Web Services," *IEEE Trans. Services Computing*, vol. 7, no. 1, 2014, pp. 32–39.
11. Z. Wen et al., "Cost Effective, Reliable and Secure Workflow Deployment over Federated Clouds," *IEEE Trans. Services Computing (TSC)*, preprint, 17 Mar. 2016; doi:10.1109/TSC.2016.2543719.
12. J.E. Gonzalez et al., "GraphX: Graph Processing in a Distributed Dataflow Framework," *Proc. 11th Usenix Conf. Operating Systems Design and Implementation*, 2014, pp. 599–613.
13. K. Yamanishi and J. Takeuchi, "A Unifying Framework for Detecting Outliers and Change Points from Non-Stationary Time Series Data," *Proc. 8th ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining (SIGKDD 02)*, 2002, pp. 676–681.
14. M. Heller, "Review: 6 Machine Learning Clouds," *Infoworld*, 11 May 2016; www.infoworld.com/article/3068519/artificialintelligence/review-6-machine-learning-clouds.html.

Zhenyu Wen is a postdoctoral researcher in the School of Informatics at the University of Edinburgh. His research interests include multiobjective optimization, artificial intelligence, and cloud computing. Wen has a PhD in cloud computing from Newcastle University. Contact him at zw@inf.ed.ac.uk.

Renyu Yang is a researcher in the School of Computer Science and Engineering at Beihang University. His research interests include dependable distributed systems and cloud computing. Yang has a PhD in computer science and engineering from Beihang University. Yang is the corresponding author; contact him at yangry@act.buaa.edu.cn.

Peter Garraghan is a lecturer in the Department of Computing and Communications at the University of Lancaster. His research interests include distributed systems and large-scale cloud data centers. Garraghan has a PhD in computer science from the University of Leeds. Contact him at p.garraghan@lancaster.ac.uk.

Tao Lin is a master's student in the School of Computer and Communication Sciences at the École Polytechnique Fédérale de Lausanne (EPFL), Switzerland. His research interests include scalable machine learning and cloud computing. Lin has a BS in electrical engineering from Zhejiang University. Contact him at tao.lin@epfl.ch.

Jie Xu is a chair professor of computing at the University of Leeds. His research interests include large-scale distributed computing and dependability. Xu has a PhD in

advanced fault-tolerant software from Newcastle University. Contact him at j.xu@leeds.ac.uk.

Michael Rovatsos is a senior lecturer in the School of Informatics at the University of Edinburgh. His research interests include multiagent systems, distributed artificial intelligence, and social computation. Rovatsos has a PhD in computer science from the Technical University of Munich. Contact him at mrovatso@inf.ed.ac.uk.

myCS

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

IEEE  computer society

Read all your IEEE magazines and journals your **WAY** on

myCS

Introducing **myCS**, the digital magazine portal from IEEE Computer Society. Go beyond static, hard-to-read PDFs with an easily accessible, customizable, and adaptive experience.

There's No Additional Cost!

Now there's even more to love about your membership...





▶ LEARN MORE AT: mycs.computer.org