Fog Orchestration for IoT Services: Issues, Challenges and Directions

Zhenyu Wen, Renyu Yang*, Peter Garraghan, Tao Lin, Jie Xu, and Michael Rovatsos
**The corresponding author: yangry@act.buaa.edu.cn*

Abstract: Large-scale IoT services such as health-care, smart cities and marine monitoring are pervasive in Cyber-physical environments strongly supported by Internet technologies and Fog computing. Complex IoT services are increasingly composed of sensors, devices, and compute resources within Fog computing infrastructures. The orchestration of such applications can be leveraged to alleviate the difficulties of maintenance and enhance data security and system reliability. However, how to efficiently deal with dynamic variations and transient operational behavior is a crucial challenge within the context of choreographing complex services. Furthermore, with the rapid increase of the scale of IoT deployments, the heterogeneity, dynamicity, and uncertainty within Fog environments and increased computational complexity further dramatically aggravate this challenge. This article provides an overview of the core issues, challenges and future research directions in Fog-enabled orchestration for IoT services. Additionally, we present early experiences of an orchestration scenario, demonstrating the feasibility and initial results of using a distributed genetic algorithm in this context.

Key words: Internet of Things, Fog computing, orchestration, distributed systems

I. Internet of Things and Fog Computing

The proliferation of the Internet and increasing integration of physical objects including sensors, vehicles, and buildings have resulted in the formation of Cyber-physical environments that encompass both physical and virtual components. These objects are capable of interfacing and interacting with existing network infrastructure, allowing for computer-based systems to interact with the physical world, thereby 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 research and industry on issues related to how insight can be effectively extracted from such data to assist the design of Cyber-physical systems. IoT services typically comprise of a set of software components running over different locations connected through networks (i.e. 4G, wireless LAN, Internet etc.) that exhibit dynamic behavior. Systems such as datacenters and wireless sensor networks underpin the data storage and compute resources required for the operation of these components.

A new computing paradigm - Fog computing - succeeds Cloud computing by placing greater emphasis of computation and data storage at the edge of the network, allowing for reduced latency and response delay jitter for applications [1, 14]. These characteristics are particularly important for latency-sensitive applications such as gaming and video streaming. In the IoT environment, existing applications and massive physical devices can be leveraged as *fundamental appliances* and composed in a mash-up 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. The *Fog orchestrator* provides (a) the centralized arrangement of the resource pool, mapping applications with specific requests and providing an automated workflow to physical resources (deployment and scheduling); (b) workload execution management with runtime QoS control; and (c) time-efficient directive generation to manipulate specific objects.

II. Scenario and Application

A. Motivating Example

Smart cities aim to enhance the quality of urban life by using technology to improve the efficiency of services to meet the needs of residents. To this end, multiple information and communication technology (ICT) systems need to be integrated in a secure, efficient and reliable way in order to manage city facilities effectively. Such systems consist of two major components: (1) sensors integrated with real-time monitoring systems, and (2) applications integrated with the collected sensor (or device) data. Currently, IoT services are rudimentary in nature, and only integrate with specific sensor types. This is resultant of no existing universally agreed standards and protocols for IoT device communication, and represents a challenge towards achieving a global ecosystem of interconnected *things*.

To address this problem, an alternative approach is the use of an IoT service orchestration system to determine and select the best IoT appliances for

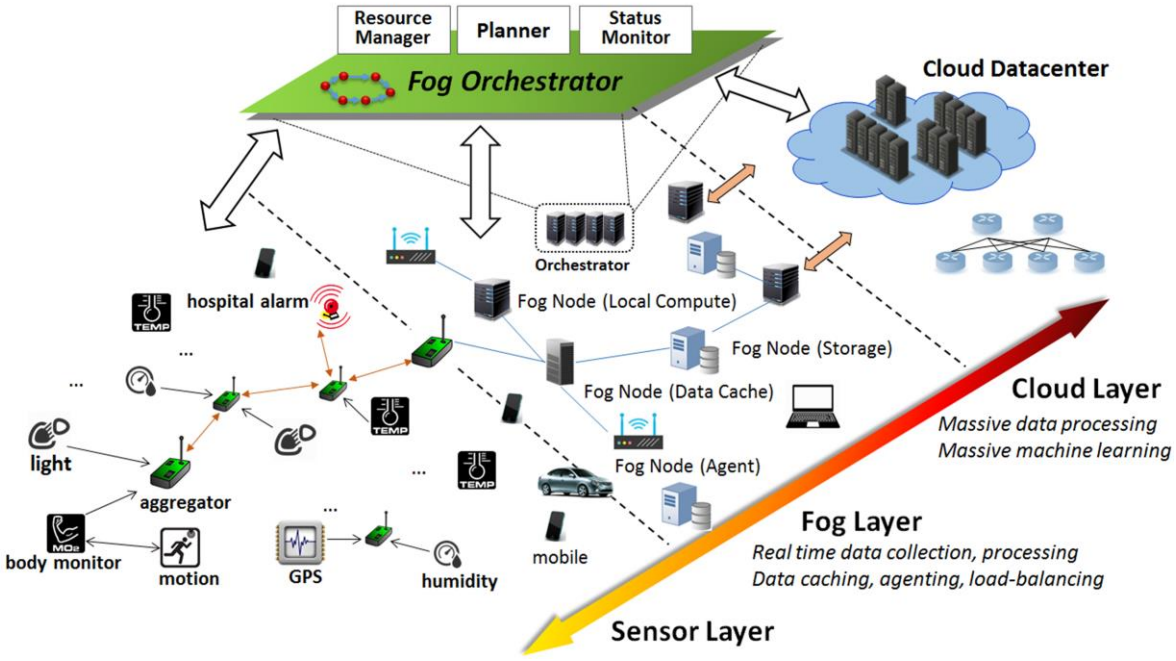

Figure 1. An orchestration scenario for an eHealth service: different IoT appliances (diverse types of sensors and Fog nodes) are orchestrated as a workflow across all layers of Fog architecture. Several candidate objects can potentially provision similar functionality. The Fog orchestrator acts as a controller deployed on a workstation or Cloud datacenter and across all organization layers based on global information. Its primary responsibility is to select resources and deploy the overall service workflow according to data security, reliability, system efficiency requirements. It is noteworthy that the orchestrator is a centralized controller only at a conceptual level and might be implemented in a distributed and fault-tolerant fashion,without introducing a single point of failure.

dynamic composition of holistic workflows for more complex functions. As shown in Figure 1, the proposed orchestrator manages all layers of an IoT ecosystem to integrate different standalone appliances or service modules into a complex topology.

An appropriate combination of these standalone IoT appliances can be used to facilitate more advanced functionality, allowing for reduced cost and improved user experience. For example, *mobile health* systems are capable of remote monitoring, real-time data analysis, emergency warning, etc. Data collected from wearable sensors that monitor patient vitals can be continuously sent to data aggregators and, in the event of detection of abnormal behavior, hospital personnel can be immediately notified in order to take appropriate measures.

While such functionality can be developed within a standalone application, this provides limited scalability and reliability. As concerns scaling capability, 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. For reliability, increased application complexity leads to tedious, time-consuming debugging. The use of orchestration allows for more flexible formation of application functionality to scale and it also decreases the probability of failure correlation between application components.

B. Fog-enabled IoT Application

Traditional Web-based service applications are deployed on servers within Cloud datacenters that are accessed by end devices such as tablets, smart phones and desktop PCs. In contrast, IoT applications deployed within Fog computing systems consist of the Cloud, *Fog Node*, and *Things* as shown Figure 1. In this context, a *Fog node* is defined as equipment or middleware and served as an agent that collects data from a set of sensors, and which is then transmitted to a centralized computing system that locally caches data and performs load balancing. *Things* are defined as networked devices including sensors and devices with built-in sensors. Similarly to Web-based service applications, the Cloud provisions centralized resource pools (compute, storage) in order to analyze collected data and automatically trigger subsequent decisions based on a pre-defined system logic. The most significant difference, however, is the use of *Fog nodes* that transmit data to Cloud datacenters. For example, most of wearable sensor data is collected and pre-processed by smart phones or adjacent workstations. This can either significantly reduce

transmission rates or improve their reliability. The main differences between Web-based and IoT applications can be summarized as follows:

Message communication and scalability IoT communication is performed using a hybrid centralized-decentralized approach depending on context. The majority of messages exchanged from sensor to sensor *(S2S)* or sensor to Cloud *(S2C)* is performed through the use of *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 *(V2V)* and traffic infrastructure *(V2I)* in order to prevent collisions. Such a system was piloted in New York City using WiFi in order to enable real-time interaction to assist drivers in navigating congestion and to communicate with pedestrians or oncoming vehicles [2]. Furthermore, due to 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 In light of Software Defined Network (SDN) technologies, the advantages of a software defined approach is the de-coupling of the software control and the heterogeneous hardware operations. This approach provides an opportunity to dynamically achieve different quality levels for different IoT applications in heterogeneous environments [3]. Moreover, application-level interoperability benefits from Web technologies such as the RESTful architecture of Web, which provide a high level of interoperability. Using these technologies, an abundance of programming APIs can be distributed across entire Fog domains and can be utilized to increase the flexibility of loosely-coupled management [4]. Lightweight APIs, such as RESTful interfaces, result in agile development and simplified orchestration with enhanced scalability when composing complex distributed workflows.

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, etc. Furthermore, the uncertainty of potentially unstable and mobile system components results in increased difficulties in predicting and capturing system operation. Therefore, the reliability of an IoT application workflow needs to be measured and enhanced in more elaborate ways.

III. IoT Application Orchestration Challenges

We have demonstrated that existing IoT applications are very diverse in terms of reliability, scalability and security. The 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 whilst meeting non-functional requirements such as security, network latency, QoS, etc. We outline and elaborate on these specific challenges as follows:

Scale and complexity With the increase of IoT manufacturers developing 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 (e.g., ARM, Intel) or operating systems, while applications with high security requirements might require specific hardware and protocols to function. Not only does orchestration 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 comprising 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 which may 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 the data and traffic gateway that is particularly vulnerable to such attacks. This is especially true in the context of network-enabled IoT systems, whose attack vectors can range from 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 towards enhancing network security to protect against Cyber-attacks [5]. Furthermore, in systems comprising of hundreds of thousands electronic devices, how to effectively and accurately evaluate the security and measure its risks

is critically important in order to present a holistic security and risk assessment [6]. This becomes challenging when workflows are capable of changing and adapting at runtime. For these reasons, we believe that approaches capable of dynamically evaluating 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 is a particular problem in the context of software upgrades through Fog nodes or the frequent join-leave behavior of network objects which will change its 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 properties (for example, low-battery devices will degrade the data transmission rate). Finally, the performance of applications will change due 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 re-configuration of the topological structure and assigned resources within the workflow, and importantly, that of *Fog nodes*.

Fault diagnosis and tolerance The scale of a Fog system results in increased failure probability. Some rare-case software bugs or hardware faults which do not manifest at small-scale or testing environments such as stragglers [7] have a debilitating effect on system performance and reliability. At the scale, heterogeneity, and complexity we are anticipating, it is very likely that different types of fault combinations will occur [8]. To address these, redundant replications and user-transparent fault-tolerant deployment and execution techniques should be considered in orchestration design.

IV. Key Research Directions

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

A. Component Selection and Placement

The recent trend in composing Cloud applications is driven by connecting heterogeneous services deployed across multiple datacenters. Similarly, such a distributed deployment aids in improving IoT application reliability and performance within Fog computing environments. As mentioned in Section III, it also exposes appliances to new security risks and network uncertainty. Ensuring high levels of dependability for such workflows composed by a multitude of systems is a considerable challenge. Numerous efforts [9,10] have focused on QoS-aware composition of native VM-based Cloud application components, but neglect the proliferation of uncertain execution and security risks among interactive and interdependent components within IoT services.

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 [11] provides possible solutions to leverage an in-memory computing framework to execute tasks in a Cloud infrastructure in parallel. However, how to realize 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, a potential approach could be *correction mechanisms* that could be applied even when sub-optimal 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 the improvement of decision results with new information and a re-evaluation of constraints, and make use of task and data migration approaches to realize workflow re-deployment.

B. Dynamic Orchestration with Runtime QoS

Apart from the initial placement, all workflow components dynamically change due to internal transformations or abnormal system behavior. IoT applications are exposed to uncertain environments where variations in execution are commonplace. Due to the degradation of consumable devices and sensors, capabilities such as security and reliability that initially were guaranteed will vary accordingly, resulting in the initial workflow being no longer optimal or even totally invalid. Furthermore, the structural topology might change in accordance to the task execution progress (i.e. a computation task is

finished or evicted) or will be affected by the evolution of the execution environment. Abnormalities might occur due 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, it is essential to dynamically orchestrate task execution and resource reallocation.

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

Event streaming and messaging Such performance metric variables or significant state transitions can be depicted as system events, and *event streaming* is processed in the orchestration framework through an event messaging bus, real-time publish-subscribe mechanism or high-throughput messaging systems (e.g., Apache Kafka), therefore significantly reducing the communication overheads and ensuring responsiveness. Subsequent actions could be automatically triggered and driven by Cloud engine (e.g., Amazon Lambda service).

C. 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 be typically considered too computationally intensive, as it is extremely time-consuming to perform operations including pre-filtering, 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 re-calculation by re-running 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 [12]. The data and event streams should be efficiently transmitted among Fog components, so that system outage, appliance failure, or load spikes will rapidly feedback to the central orchestrator for decision making.

Incremental design and implementation Based on the time series of graphs, the similarities and dependencies between successive graph snapshots should be comprehensively studied 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 sub-graphs changes within the whole topology by using sub-graph partitioning and matching as an automated process that can significantly reduce overall execution time.

D. Systematic Data-driven Optimization

IoT applications include a very large number of geographically distributed devices which produce multi-dimensional, high-volume data that requires different levels of real-time analytics and data aggregation. Therefore, data-driven optimization and planning should be considered in the orchestration of complex IoT services.

Holistic cross-layer optimization As applications are selected and distributed across different layers in the Fog environment, attention should be brought to the optimization of all overlapping, inter-connected 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 datacenter side. It would be possible to pipeline the stream of data processing and the database services within the same network domain to reduce data transmission. Similar to the data-locality principle, we can also distribute or reschedule computation tasks of *Fog nodes* close to the sensors rather than frequently moving data, thereby reducing latency. Another potential optimization is to customize data-relevant parameters such as data generation rate or data compression ratio to adapt to the performance and assigned resources so as to strike a balance between data quality and specified response-time targets.

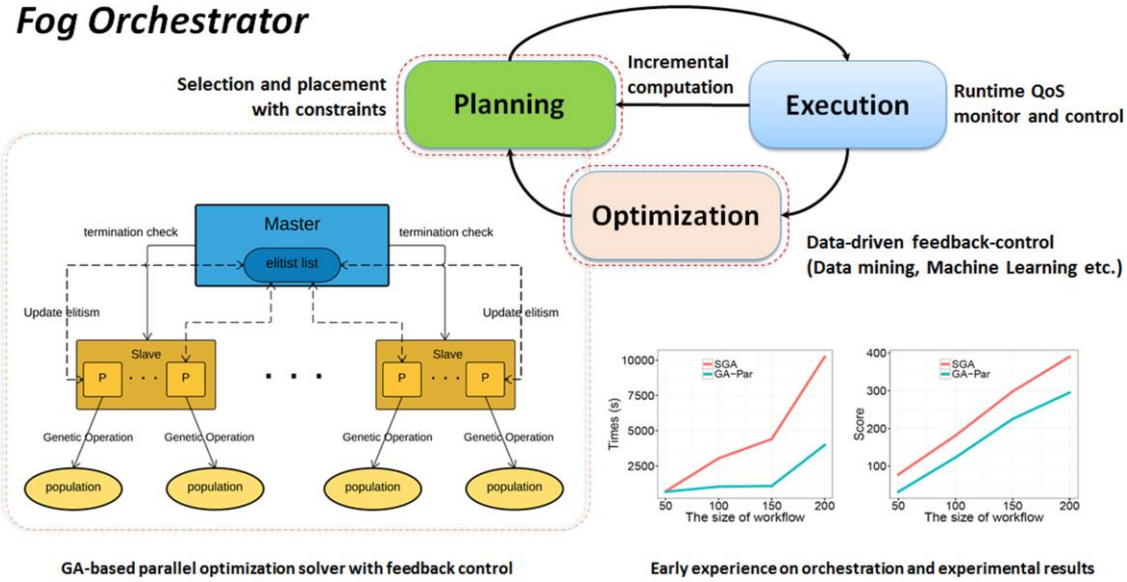

Figure 1: Main functional elements in our Fog Orchestrator: *planning* is responsible for selection and placement, runtime monitoring and control during *execution*, and data-driven decision *optimization*. We propose a parallel GA solver to accelerate the handling of optimization issues raised in the planning and optimization phase. Initial results demonstrate the proposed approach can outperform a standalone genetic algorithm in terms of both time and quality aspects.

Online tuning and History-Based Optimization (HBO) A major challenge is that the aforementioned decision operators are still computationally time-consuming. To tackle this problem, online machine learning is capable of provisioning several online training (such as classification and clustering) and prediction models to capture the constant evolutionary behavior of each element in the system, producing time series of trends to intelligently predict the required system resource usage, failure occurrence, and straggler compute tasks - all of which can be learnt from historical data and an HBO procedure. These smart techniques should be investigated, 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 [13].

V. 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 demonstrated in Figure 2, in order to enable planning and adaptive optimization, a preliminary attempt was made to manage the composition of applications in parallel under a broad range of constraints. We implement a novel parallel genetic algorithm based framework (GA-Par) on Spark to handle orchestration scenarios where a large set of IoT applications are composed. More specifically, in our GA-based algorithm, each chromosome represents a solution of the composed workflow and the gene segments of each chromosome 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. In order to maximize parallelism, we set up and adjust the partition configuration dynamically to make partitions fully parallelized whilst considering data shuffling and communication cost with the topology change. To guarantee optimal results can be gradually obtained, we dynamically merge several partitions into a single partition and then re-partition it based on runtime status and monitored QoS. Furthermore, the quality of each solution generation can be also maintained by applying an "elitist" method, where the local elite results of each partition will be collected and synthesized into a global elite. The centralized GA-Par master will aggregate 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 perform some initial experiments on 30 servers hosted on Amazon Web Services (AWS) as the Cloud datacenter for the Fog environment. Each server is hosted as an r3.2xlarge instance with 2.5GHz Intel Xeon E5-2670v2 CPUs, 61GB RAM, and 160GB disk storage. We use simulated data below to 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). The metrics quality, execution time and fitness score (with lower values indicating better results) are used to evaluate SGA and GA-Par. As can be observed in Figure 2, GA-Par outperforms SGA. The time consumption of GA-Par has been significantly reduced to nearly 50% of that of SGA, while the quality of appliance selection in GA-Par is always at least 30% higher than that of SGA. However, the scalability of our current approach is still slightly affected by increasing numbers of components and requests, indicating that we still need to explore opportunities for incremental re-planning and on-line tuning to improve both time-efficiency and effectiveness of IoT orchestration.

VI. CONCLUSION

Most recent research related to Fog computing explores architectures within massive infrastructures [14]. 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. In this paper, we have outlined and numerous difficulties and challenges to develop an orchestration framework across all layers within the Fog resource stack, and have described a prototypical orchestration system that makes use of some of the most promising mechanisms to tackle these challenges.

Acknowledgment

This work is supported by China NKR&D Program (2016YFB1000103), National 863 Program (2015AA01A202), NSFC (No. 61421003), and the European Commission's FP7 Programme (FP7/2007-2013) (Grant No. 600854).

REFERENCES

[1] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," *in ACM MCC, 2012*, pp. 13–16.

[2] Kim M. D. Dikaiakos, A. Florides, T. Nadeem, and L. Iftode, "Locationaware services over vehicular ad-hoc networks using car-to-car communication," in *IEEE JSAC*, vol. 25, no. 8, pp. 1590–1602, 2007.

[3] Z. Qin, G. Denker, C. Giannelli, P. Bellavista, and N. Venkatasubramanian, "A software defined networking architecture for the internet-ofthings," in *IEEE NOMS, 2014*, pp. 1–9.

[4] S. Nastic, S. Sehic, D. H. Le, H. L. Truong, and S. Dustdar, "Provisioning software-defined iot cloud systems," in *IEEE FiCloud, 2014*, pp. 288–295.

[5] R. Roman, P. Najera, and J. Lopez, "Securing the internet of things," *in IEEE Computer*, vol. 44, no. 9, pp. 51–58, 2011.

[6] A. Riahi, Y. Challal, E. Natalizio, Z. Chtourou, and A. Bouabdallah, "A systemic approach for iot security," in *IEEE ICDCSS*, 2013, pp. 351–355.

[7] P. Garraghan, X. Ouyang, R. Yang, D.Mckee, and J. Xu, "Straggler root-cause and impact analysis for massive-scale virtualized cloud datacenters," [Online] *in IEEE TSC*, vol. PP, no. 99, pp. 1–1, 2016.

[8] R. Yang, Y. Zhang, P. Garraghan, Y. Feng, J. Ouyang, J. Xu, Z. Zhang and C. Li, "Reliable computing service in massive-scale systems through rapid low-cost failover," [Online] *in IEEE TSC*, vol. PP, no. 99, pp. 1–1, 2016.

[9] Z. Zheng, Y. Zhang, and M. R. Lyu, "Investigating qos of real-world web services," *in IEEE TSC*, vol. 7, no. 1, pp. 32–39, Jan 2014.

[10] Z. Wen, J. Cala, P. Watson, and A. Romanovsky, "Cost effective, reliable and secure workflow deployment over federated clouds," [Online] *in IEEE TSC*, vol. PP, no. 99, pp. 1–1, 2016.

[11] J. E. Gonzalez, R. S. Xin, A. Dave, D. Crankshaw, M. J. Franklin, and I. Stoica, "Graphx: Graph processing in a distributed dataflow framework," *in USENIX OSDI, 2014*, pp. 599–613.

[12] K. Yamanishi and J. ichi Takeuchi, "A unifying framework for detecting outliers and change points from non-stationary time series data," *in ACM SIGKDD*, 2002, pp. 676–681.

[13] Cloud machine learning. [Online]. Available: http://www.infoworld.com/article/3068519/artificialintellegence/review-6-machine-learning-clouds.html

[14] S.Yi, C.Li, and Q.Li. A Survey of Fog Computing: Concepts, Applications, and Issues. *In ACM MBDW*, 2015, pp. 37-42

Biographies

Zhenyu Wen is currently a postdoctoral researcher at University of Edinburgh, UK. His research interests include multi-objective optimization, AI and Cloud computing. Email: zwen@inf.ed.ac.uk

Renyu Yang is currently a researcher at Beihang University, China. His research interests are dependable distributed systems and Cloud computing. Corresponding Email: yangry@act.buaa.edu.cn

Peter Garraghan is a Lecturer at University of Lancaster, UK. His research interests include distributed systems and large-scale cloud datacenters. Email: P.Garraghan@lancaster.ac.uk

Tao Lin is currently a master student at EPFL, Switzerland. His research interests include scalable machine learning and cloud computing. Email: tao.lin@epfl.ch

Jie Xu is a Chair Professor of Computing at University of Leeds, UK. His research interests include large-scale distributed computing and dependability. Email: j.xu@leeds.ac.uk

Michael Rovatsos is a Senior Lecturer at University of Edinburgh, UK. His research is in multiagent systems, distributed AI, and social computation. Email: mrovatso@inf.ed.ac.uk