EDITOR: Schahram Dustdar, dustdar@dsg.tuwien.ac.at

DEPARTMENT: INTERNET OF THINGS, PEOPLE, AND PROCESSES

# Toward Distributed, Global, Deep Learning Using IoT Devices

Bharath Sudharsan, *National University of Ireland Galway, Galway H91 TK33, Ireland*
Pankesh Patel, *National University of Ireland Galway, Galway H91 TK33, Ireland*
John Breslin, *National University of Ireland Galway, Galway H91 TK33, Ireland*
Muhammad Intizar Ali, *Dublin City University, Dublin 9, Ireland*
Karan Mitra, *Luleå University of Technology, 97187 Luleå, Sweden*
Schahram Dustdar, *TU Wien, 1040 Vienna, Austria, Cardiff University, Cardiff CF10 3AT, U.K.*
Omer Rana, *Cardiff University, Cardiff CF24 3AA, U.K.*
Prem Prakash Jayaraman, *Swinburne University of Technology, Hawthorn VIC 3122, Australia*
Rajiv Ranjan, *Newcastle University, Newcastle upon Tyne NE1 7RU, U.K.*

*Deep learning (DL) using large scale, high-quality IoT datasets can be computationally expensive. Utilizing such datasets to produce a problem-solving model within a reasonable time frame requires a scalable distributed training platform/system. We present a novel approach where to train one DL model on the hardware of thousands of mid-sized IoT devices across the world, rather than the use of GPU cluster available within a data center. We analyze the scalability and model convergence of the subsequently generated model, identify three bottlenecks that are: high computational operations, time consuming dataset loading I/O, and the slow exchange of model gradients. To highlight research challenges for globally distributed DL training and classification, we consider a case study from the video data processing domain. A need for a two-step deep compression method, which increases the training speed and scalability of DL training processing, is also outlined. Our initial experimental validation shows that the proposed method is able to improve the tolerance of the distributed training process to varying internet bandwidth, latency, and Quality of Service metrics.*

IoT datasets are now being produced at an ever increasing rate, as emerging IoT frameworks and libraries have simplified the process of continuous monitoring, real-time edge-level processing, and encrypted storage of the generated multimodal image, audio, and sensor data. Such data are generated by a variety of hardware systems operating in indoor and outdoor infrastructures, including smart factory floors, AR/VR experience centers, smart city sensors, etc. In order to complete training in a reasonable time when using such large scale, high-quality IoT datasets that have been collected over decades, we need a scalable distributed training system that can efficiently harness the hardware resources of millions of IoT devices. Particularly, such a system should take account of current network connectivity between these devices, and able to collectively train to produce the final problem-solving deep learning (DL) models at very high speeds.


Digital Object Identifier 10.1109/MIC.2021.3053711
Date of current version 18 June 2021.

Instead of following the traditional approach that loads such datasets and trains a model locally within a GPU cluster or a data center, we utilize distributed training on multiple IoT devices as:

i) Considering the GPU to IoT devices ratio, IoT devices are much greater in number, i.e., market estimates show that roughly 50 Billion Micro-Contoller Units (MCU) and small CPU chips were shipped in 2020, which far exceed other processors like GPUs (only 100 Million units sold);
ii) Every modern household does not compulsorily own a GPU, yet it roughly has around a dozen medium resource IoT devices which when efficiently connected together can, within a home network, train machine learning models without depending on Cloud or GPU servers that can perform the same training task at very high speeds, but at additional cost;
iii) In most real-life IoT scenarios, the training dataset used to produce a learned model can often be hard to source due to GDPR and privacy concerns. In such cases, we need an algorithm to directly utilize capability of the IoT device hardware without disturbing routine operation of the device. This algorithm when deployed across user devices should make use of locally generated data to "collectively" train a model without storing live data on a central server. Thus, locally producing learned models from data without violating the privacy protection regulations;
iv) Training advanced DL models on a single GPU might consume days or even weeks to converge. Hence, if we design and use an intelligent algorithm that can tolerate high latency and low bandwidth constraints, we can collectively harness the idle hardware resource of thousands of mid-sized IoT devices and complete training at very high speeds. For example, at the time of writing, the latest GEFORCE RTX 2080 Ti GPU has 11GB RAM but costs ∼US $1500. Whereas one Alexa smart speaker device has 2 GB RAM and efficiently connecting 20 such devices can collectively pool 40 GB of RAM. In this way, we can complete training faster on such resources, if coordinated correctly, compared to expensive GPU and at a comparatively smaller investment—particularly by utilizing idle capacity of smart IoT devices that exist across the world.

The hardware of IoT devices is not designed for DL workloads. Resource-friendly model training algorithms like Edge2Train[1] could be used in distributed setups for training models MCUs and limited capacity CPUs of IoT devices. We identify challenges involved with DL model training on hardware of common IoT devices such as video doorbells, smart speakers, cameras, etc. To overcome some of the challenges, we also present a two-step deep compression method that increases the training speed and scalability of DL training processing.

**Outline**. For globally distributed DL model training scenarios, in section "Distributed Global Training: Research Challenges," we present our bottleneck analysis. Section "Proposed two-step Deep Compression Method and Initial Experimental Results" contains our solution to address the challenges highlighted in Section "Distributed Global Training: Research Challenges." In section "Discussion," we conclude by providing greater context for future work.

## DISTRIBUTED GLOBAL TRAINING: RESEARCH CHALLENGES

In the large-scale distributed/collaborative learning domain, distributed training has seen limited adoption, especially when the target is to train a DL model than can perform video analytics tasks such as object detection (e.g., detect FedEx, USPS vehicles, etc.) for package theft prevention, detect, and recognize unsafe objects such as a gun to reduce crime, identify known/unknown faces. This is because:

i) Models that can learn from video datasets have a dense (i.e., large number of parameters and layers) architecture design that requires significant computational resources when compared to models designed to learn from image or audio datasets. For example, the popular ResNet-50 model trained using a 2-D image dataset consumes around 4 GFLOPs, whereas a ResNet-50 Inflated 3-D model contains 3-D convolutional kernels to model temporal information in a video, consuming 30 GFLOPs, i.e., more than 7× times larger than the previous case;
ii) These datasets can be significant in size, hence consuming high internet bandwidth when loading video from a (central) data server to training devices that are geographically distributed. For example, the ImageNet dataset has 1.28M images, whereas the Kinetics-400 video dataset has 63M frames, i.e., 50× times larger; and
iii) Finally, complex models trained on such datasets can have millions of parameters and gradients that need to be quickly exchanged (with minimum latency) among devices during

distributed training, which again increases internet traffic (and charges to consumers) and more critically can lead to slow convergence when devices involved in training suffer from network latency issues. In short, the bottlenecks are due to the demand for high computational power, time overhead associated with dataset loading I/O, and slow exchange of model gradients. In the rest of this section, each of these three bottlenecks are explained in more detail.

## High FLOPs Consumption

Unlike for 2-D image recognition models, the input/activation of video analytics DL networks has [N, T, C, H, W] as its five dimensions, where: N is the batch size, T refers to temporal timestamps, the channel number is C, and spatial resolution H & W. To reduce computational overhead and network congestion, we can train using the same target dataset by applying 2-D CNN to each image frames from the video. Using such an approach, the temporal relationship between the frames cannot be modeled/learned, which is crucial to understanding the scenes (labeled) from the video datasets. Hence, inflating the 2-D to 3-D convolution layer results in producing an I3-D model, which grows the model size by *k* times. For distributed learning of spatio-temporal data, the models with 3-D convolutions, in addition to model size demands, also suffers from having a large number of parameters, which is the main reason to slow down the training and communication process even within a GPU cluster and in real-world networks. Consequently, training will stall when unexpected network issues are encountered.

## Expensive Dataset Loading I/O

Video network architectures available in ML Hubs and marketplaces (Google AI Hub and TensorFlow Hub) usually sample many frames from video datasets and use them as input during learning (i.e., top models[2] sample 32 and 64 frames). Then, they progressively reduce the temporal resolution by realizing "temporal pooling" techniques.[3] Another orthogonal approach is to design networks that sample and use fewer frames (i.e., eight frames) during learning and maintain the same temporal resolution to retain information from the video dataset. In both designs, the overall computational requirements are similar, but the former involves additional sampling and full dataset loading steps, increasing the dataset loading I/O at the data server, while making data loading on many distributed IoT devices challenging when considering the limited memory and internet bandwidth available in practice.

## Slow Exchange of Model Gradients

During training, maintaining good scalability, low latency, and high bandwidth internet connection is mandatory at least during gradients exchange.[3] Existing large-scale distributed learning studies and frameworks require high-end Infiniband network infrastructure where bandwidth ranges from 10 to 100 Gb/s, with a $\sim$1 $\mu$s latency. Even if we increase bandwidth by stacking (aggregating) hardware, latency improvements are still difficult to achieve. In contrast to our assumption, latency in real-world scenarios can be further exacerbated due to queueing delay in switches and indirect routing between service providers. This bottleneck makes distributed training scale poorly in real-world network conditions, particularly when transmitting datasets in addition to the gradients.

## Handling Dataset I/O Efficiency

Video datasets are usually stored in a high-performance storage system (HPSS) or a central data server that is shared across all worker nodes—in our case these are IoT devices distributed across the world. Although HPSS systems have good sequential I/O performance, their random-access performance is inferior, causing bottlenecks for large data traffic. Most existing I3-D models a high frame rate (within a video), then perform a downsampling to reduce overall data size. Given the distributed training scenario being considered, we argue that such designs waste bandwidth. Consequently, research needs to consider novel data approximation, sampling and filtering methods. For example, in the context of video datasets, one can develop a method to identify videos that have multiple similar frames (i.e., we say that nearby frames contain similar information), then load and share only nonredundant frames during distributed training. Similarly, for other datasets associated with images and sensor readings, we recommend filtering or downsampling the data without losing information, then distributing it during training. Therefore any approximation, sampling, and filtering method will need to be correctly parameterized while considering the resource-constrained nature of IoT devices.

## Variable Training and Convergence Speed

Research has shown that naive synchronous stochastic gradient descent (SSGD) achieves poor scalability in real time and distributed DL model training, making training using 100 distributed GPUs slower than training on 1 GPU. Unlike SSGD, asynchronous SGD (ASGD) relaxes synchronization enabling its use across many

real-world applications. D2[10] and AD-PSGD[11] perform only partial synchronization in each update to overcome latency issues. Such large-scale training takes advantage of data parallelism by increasing the number of contributing devices, but at the cost of data transfer between devices (e.g., exchange of parameters), which can be time consuming, especially when many devices are pooled. This dwarfs the savings in computation time and producing a low computation-to-communication ratio. However, such distributed learning approaches do not scale well when network latency is high. Additionally, lower network bandwidth, expensive/limited mobile data plan, and intermittent network connection, which are all common across use of mobile devices, also impact our training scenarios. Hence, if we use SSGD, ASGD, D2, AD-PSGD (or any such native algorithms) across a large number of medium-resource IoT devices, the target DL model might never converge to a suitable level of accuracy. Hence, there is a need for a method that can efficiently communicate with a large number of heterogeneous IoT devices, even under real-world internet latency and bandwidth constraints, and complete training at high speeds. As SSGD, ASGD, D2, AD-PSGD can be adapted to learn a globally distributed model, there is a need to develop benchmarking techniques that compare them against common evaluation metrics including average accuracy, training time, and convergence speed. Eventually, these evaluation metrics will need to be formulated as a unified distributed-training performance model. Metaheuristic techniques such as genetic programming and particle swarm optimization could be used to solve and find feasible (Pareto optimal) solutions for improving performance model.

## Handling Network Uncertainties

Distributed learning can be impacted by properties of access links that connect IoT devices (sensors and actuators) to edge gateways and/or cloud nodes. These uncertainties include time-varying connectivity, network unavailability, and time-varying traffic patterns research has indicated that wireless network bandwidth and availability fluctuates dramatically due to weather conditions, signal attenuation, and channel interference. For instance, consider the use of SSGD during a distributed training process, where only one gradient transmission occurs in one iteration. This aspect can worsen with an increase in the number of transmissions, and if the previously sent gradients arrive late along with recent gradients (late arrival due to network congestions). The second issue we expect is the large variance in latency, which is common in real-world IoT networks, especially where devices have long-distance connections and communicate via a range of networks, e.g., long-range/low-power communications using LoRa-WAN and NarrowBand-IoT and more powerful high-bandwidth WiFi, 4G/5G radio. While we can aim to maintain a low average latency by choosing and involving only IoT devices with stable internet connection, changes in device network connectivity due to mobility (e.g., when the IoT device is placed in a car) can cause variable latency.

## Handling Staleness Effects

Most popular distributed model training techniques (e.g., SSGD, ASGD, D2, AD-PSGD) adopt a nonsynchronous execution approach for alleviating network communication bottleneck that produces *stale* parameters, i.e., the model parameters arrive late, not reflecting the latest updates. Staleness not only slows down convergence but also degrades model performance. Despite notable contributions in distributed learning,[12,13] the effects of staleness during training can lead to model instability,[14] because it is practically not feasible to monitor and control staleness in the current complex IoT environments containing heterogeneous devices using different network protocols. This challenge can be addressed by designing *accuracy guaranteeing dynamic error compensation and network coding techniques*—primarily a light-weight technique that adopts a two-step process. In the first step gradient synchronization is not performed, instead each participating IoT device updates their part of the model with locally available gradients (e.g., local learning). In the second step, IoT devices perform gradient synchronization based on the computed averaged gradients, which takes account of the designed error compensations.

# PROPOSED TWO-STEP DEEP COMPRESSION METHOD AND INITIAL EXPERIMENTAL RESULTS

In this section, we present an initial approach to handle network uncertainties and data staleness challenges in the context of distributed training of DNNs. In our distributed training scenario, we model the communication time $t_c$ as

$$t_c = \text{latency} + (\text{model size}/\text{bandwidth}). \quad (1)$$

Both latency and bandwidth are dynamic and depend on the network condition, which we cannot control. Instead, in the following, we present model size reduction techniques that can be applied to various parts of the DL model to save communication time and networking traffic.

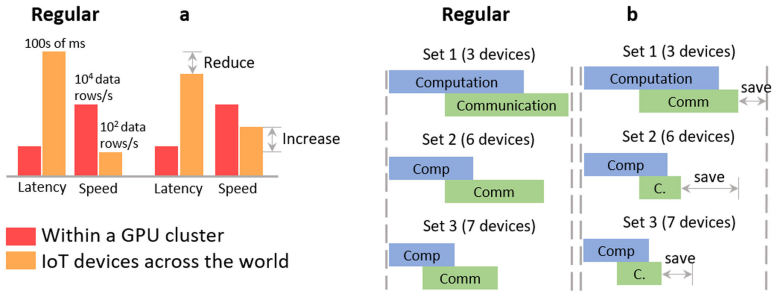


**FIGURE 1.** Comparing distributed training within a GPU cluster versus training using geographically distributed IoT devices. Our proposed two-step deep compression method can (a) tolerate latency and increase training speed, (b) reduce the communication-to-computation ratio to improve scalability and reduce communication costs.

To reduce the communication bandwidth, we recommend quantizing the model gradients to low-precision values, then transmitting these to other IoT devices or servers. The popular methods are: 1-bit SGD,[4] which achieves a 10× speedup for speech datasets. In QSGD,[5] the tradeoff between model accuracy and gradient precision was balanced. Other work demonstrate the convergence of quantized gradient training of various CNNs and RNNs. A few quantize the entire model, including gradients then perform training and a few studies use different bit sizes (e.g., DoReFa-Net[7] uses 2-bit gradients with 1-bit weights). Threshold quantization method[6] transmits gradients only when they exceeds a set threshold, which in practice is hard to choose. To improve this, a fixed proportion of positive and negative gradient was chosen[8] to update separately.

Since the theoretical quantization limit cannot exceed 32, to address this limitation, gradient sparsification methods are being applied and investigated in this distributed training setting. In the studies that sparsify the gradients by gradient dropping, the method from Ba *et al.*[9] saved 99% of gradient exchange while only compromising 0.3% of the BLEU score for a machine translation dataset. Some studies automatically tune this compression rate based on gradient activity and show 200x compression of fully connected layers for the ImageNet dataset.

From our discussions in Section "Distributed Global Training: Research Challenges," it is apparent that scalability is essential when connecting a large number of devices. To improve scalability, we need to significantly reduce communication frequency, where the communication cost is determined by network bandwidth and latency [see (1)]. All conventional studies focus on reducing the bandwidth requirements, as the latency between GPUs inside a cluster or servers inside a data center is usually low. In contrast, in our use case, since we propose to perform the same training but on IoT device hardware that is geographically distributed, latency still remains an issue due to physical device separation. For instance, if we can achieve X times training speedup on Y machines, the overall distributed training scalability (defined as X/Y) increases. Next, if we can also tolerate latency, the speedup will improve further since high latency severely reduces scalability.

We propose a two-step method to improve live model compression during training, yet without altering the DL model architecture and also without compromising the model accuracy. Our two-step deep compression method jointly aims to increase the training speed and scalability. Particularly, the first step aims to tolerate variation in real-world latency and bandwidth issues by sparsely transmitting only the important gradients. The second step aims to reduce communication-to-computation ratio and improve scalability by locally accumulating gradients, then encode and perform transmission only after crossing the gradient threshold. In the rest of this section, we describe each of these steps.

In the First step, we identify the important gradients, using gradient magnitude as the simple heuristic (users can also choose other selection criteria). We accumulate these important gradients locally to not forget the learned information. Since this step reduces the gradient synchronization frequency by not allowing to transmit all the gradients, as shown in Figure 1 (a) the training process can tolerate latency (does not reduce the dynamic real-world latency since it is practically not possible). This results in increasing training scalability, enabling the participation of more IoT devices to complete training at higher speeds.

In the Second step, after the set threshold (dynamically derived for the model in use) for the accumulated gradients is crossed, we encode the gradients (not quantizing like previous works) then transmit

them to other contributing devices involved in the training process or to the parameter server. As shown in Figure 1(b). this step improves scalability by reducing the communication-to-computation ratio by sending all the important gradients, not at defined intervals, but only when required.

Briefly, during training, both the steps jointly work to improve training speed and scalability by *accumulating, encoding, and sparsely transmitting only the important gradient*s.

## CONCLUSION

In this article, we presented an approach for training DL models on idle IoT devices, millions of which exist across the world. With an increase in mechanisms to connect such devices to a network, the potential for using such devices to support learning on locally collected data has increased. As data are maintained locally (and never transferred to a server), user privacy is also maintained—as the developed model can then be aggregated with other models (without the need to transfer raw data). We have identified and studied challenges associated with building such machine learning models, and presented a two-step deep compression method to improve distributed training speed and scalability.

The proposed approach can be used to interconnect DL frameworks executed on large scale resources (such as TensorFlow on GPU clusters) with proposals from the TinyML community (studies that design resource-friendly models for embedded systems) since we enable distributed training of computationally demanding models on distributed idle IoT devices. TinyML and related approaches often only undertake inference on IoT devices and assume that a model is constructed at a data center. A learned model is subsequently modified (e.g., using quantization) to execute on a resource constrained device (e.g., using TensorFlow-Lite). Support for performing training on resource limited devices is still limited at present—with general approaches provided in frameworks such as "Federated Learning," where a surrogate model is constructed on each remote resource, and models are then aggregated at on a cloud server. There is also an assumption within Federated Learning that each dataset (from a participating IoT device) follows the IID distribution (identical, independently distributed).

Since our method can significantly compress gradients during the training of a wide range of NN architectures such as CNNs and RNNs, the proposed approach can also be utilized alongside TF-Lite and Federated Learning approaches thereby providing the basis for a broad-spectrum of decentralized and collaborative learning applications.

## ACKNOWLEDGMENTS

This publication has emanated from research supported by grants from the European Union's Horizon 2020 research and innovation program under grant agreement number 847577 (SMART 4.0 Marie Sklodowska-Curie actions COFUND) and from Science Foundation Ireland (SFI) under Grants SFI/16/RC/3918 and SFI/12/RC/2289_P2 cofunded by the European Regional Development Fund.

## REFERENCES

1. B. Sudharsan, J. G. Breslin, and M. I. Ali, "Edge2train: A framework to train machine learning models (SVMS) on resource-constrained IoT edge devices," in *Proc. 10th Int. Conf. Internet Things*, 2020, Art. no. 6.
2. X. Wang, R. Girshick, A. Gupta, and K. He, "Non-local neural networks," 2018. [Online]. Available: https://arxiv.org/abs/1711.07971
3. J. Lin, C. Gan, and S. Han, "TSM: Temporal shift module for efficient video understanding," 2018. [Online]. Available: https://arxiv.org/abs/1811.08383
4. F. Seide, H. Fu, J. Droppo, G. Li, and D. Yu, "1-bit stochastic gradient descent and its application to data-parallel distributed training of speech DNNs." in *Proc. 15th Annu. Conf. Int. Speech Commun. Assoc.*, 2014.
5. D. Alistarh, D. Grubic, J. Li, R. Tomioka, and M. Vojnovic, "QSGD: Communication-efficient SGD via gradient quantization and encoding," 2016. [Online]. Available: https://arxiv.org/abs/1610.02132
6. N. Strom, "Scalable distributed DNN training using commodity GPU cloud computing," in *Proc. 16th Annu. Conf. Int. Speech Commun. Assoc.*, 2015, pp. 1488–1492.
7. S. Zhou, Y. Wu, Z. Ni, X. Zhou, H. Wen, and Y. Zou, "DoReFa-Net: Training low bitwidth convolutional neural networks with low bitwidth gradients," 2016. [Online]. Available: https://arxiv.org/abs/1606.06160
8. N. Dryden, T. Moon, S. A. Jacobs, and B. V. Essen, "Communication quantization for data-parallel training of deep neural networks," in *Proc. Workshop Mach. Learn. High Perform. Comput. Environ.*, 2016, pp. 1–8.
9. J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer Normalization," 2016. [Online]. Available: https://arxiv.org/abs/1607.06450
10. H. Tang, X. Lian, M. Yan, C. Zhang, and J. Liu, "D2: Decentralized training over decentralized data," 2018. [Online]. Available: https://arxiv.org/abs/1803.07068

11. X. Lian, W. Zhang, C. Zhang, and J. Liu, "Asynchronous decentralized parallel stochastic gradient descent," 2017. [Online]. Available: https://arxiv.org/abs/1710.06952
12. I. Mitliagkas, C. Zhang, S. Hadjis, and C. Ré, "Asynchrony begets momentum, with an application to deep learning," in *Proc. 54th Annu. Allerton Conf. Commun., Control, Comput.*, 2016, pp. 997–1004.
13. B. Qian *et al.*, "Orchestrating the development lifecycle of machine learning-based IoT applications: A taxonomy and survey," *ACM Comput. Surv.*, vol. 53, 2020, Art. no. 82.
14. W. Dai, Y. Zhou, N. Dong, H. Zhang, and E. P. Xing, "Toward understanding the impact of staleness in distributed machine learning," 2018. [Online]. Available: https://arxiv.org/abs/1810.03264

**BHARATH SUDHARSAN** is currently working toward the Ph.D. degree with the CONFIRM SFI Centre for Smart Manufacturing, Data Science Institute, National University of Ireland Galway, Ireland. His research areas are resource-constrained IoT devices, edge intelligence and analytics, real-time machine training. He is the corresponding author of this article. Contact him at bharath.sudharsan@insight-centre.org.

**PANKESH PATEL** is Senior Researcher with the CONFIRM SFI Centre for Smart Manufacturing, Data Science Institute, National University of Ireland Galway, Ireland. His academic background and research work focus on building software development tools to easily develop applications in the cross-section of the Internet of Things/Industry 4.0, artificial intelligence, edge computing, and cloud computing. Contact him at pankesh.patel@insight-centre.org.

**JOHN BRESLIN** is a Personal Professor (Personal Chair) in electronic engineering with the College of Science and Engineering, National University of Ireland Galway, Ireland, where he is the Director of the TechInnovate/AgInnovate programmes. Contact him at john.breslin@insight-centre.org.

**MUHAMMAD INTIZAR ALI** is an Assistant Professor with the School of Electronic Engineering, Dublin City University, Dublin, Ireland. His research interests include data analytics, Internet of Things, stream query processing, data integration, distributed and federated machine learning, and knowledge graphs. Contact him at ali.intizar@dcu.ie.

**KARAN MITRA** is an Assistant Professor with Luleå University of Technology, Luleå, Sweden. His research interests include quality of experience modelling and prediction, context-aware computing, cloud computing and mobile and pervasive computing systems. Contact him at karan.mitra@ltu.se.

**SCHAHRAM DUSTDAR** is a Professor of Computer Science and head of the Distributed Systems Group at TU Wien, Vienna, Austria. He was named Fellow of the Institute of Electrical and Electronics Engineers (IEEE) in 2016 for contributions to elastic computing for cloud applications. Contact him at dustdar@dsg.tuwien.ac.at.

**OMER RANA** is a Professor of Performance Engineering and previously led the Complex Systems research group, School of Computer Science and Informatics, Cardiff University, Cardiff, U.K. His research interests lie in the overlap between intelligent systems and high-performance distributed computing. He is particularly interested in understanding how intelligent techniques could be used to support resource management in distributed systems, and the use of these techniques in various application areas. Contact him at ranaof@cardiff.ac.uk.

**PREM PRAKASH JAYARAMAN** is a Senior Lecturer and Head of the Digital Innovation Lab in the Department of Computer Science and Software Engineering, Faculty of Science, Engineering and Technology at Swinburne University of Technology, Melbourne, Australia. Contact him at pjayaraman@swin.edu.au.

**RAJIV RANJAN** is an Australian-British computer scientist, of Indian origin, known for his research in Distributed Systems (Cloud Computing, Big Data, and the Internet of Things). He is the University Chair Professor for the Internet of Things research with the School of Computing, Newcastle University, Newcastle upon Tyne, U.K. Contact him at raj.ranjan@ncl.ac.uk.