# Big Data from the Cloud to the Edge: The aggregate computing solution

Position Paper

Shaukat Ali
Simula Research Laboratory
Fornebu, Norway
shaukat@simula.no

Ferruccio Damiani
University of Turin
Turin, Italy
ferruccio.damiani@unito.it

Schahram Dustdar
TU Wien
Wien, Austria
dustdar@dsg.tuwien.ac.at

Marialuisa Sanseverino
University of Turin
Turin, Italy
marialuisa.sanseverino@gmail.com

Mirko Viroli
Università di Bologna
Cesena, Italy
mirko.viroli@unibo.it

Danny Weyns
KU Leuven, Belgium
Linnaeus University, Sweden
danny.weyns@kuleuven.be

## ABSTRACT

We advocate a novel concept of *dependable intelligent edge systems* (DIES) i.e., the edge systems ensuring a high degree of dependability (e.g., security, safety, and robustness) and autonomy because of their applications in critical domains. Building DIES entail a paradigm shift in architectures for acquiring, storing, and processing potentially large amounts of complex data: data management is placed at the edge between the data sources and local processing entities, with loose coupling to storage and processing services located in the cloud. As such, the literal definition of edge and intelligence is adopted, i.e., the ability to acquire and apply knowledge and skills is shifted towards the edge of the network, outside the cloud infrastructure. This paradigm shift offers flexibility, auto configuration, and auto diagnosis, but also introduces novel challenges.

## CCS CONCEPTS

• **Computing methodologies** → **Distributed computing methodologies**; • **Computer systems organization** → **Distributed architectures**; **Embedded and cyber-physical systems**; • **Software and its engineering** → **Software system structures**.

## KEYWORDS

Adaptation, Dependability, Formal methods

**ACM Reference Format:**
Shaukat Ali, Ferruccio Damiani, Schahram Dustdar, Marialuisa Sanseverino, Mirko Viroli, and Danny Weyns. 2019. Big Data from the Cloud to the Edge: The aggregate computing solution: Position Paper. In *European Conference on Software Architecture (ECSA), September 9–13, 2019, Paris, France.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3344948.3344988


*ECSA, September 9–13, 2019, Paris, France*

ACM ISBN 978-1-4503-7142-1/19/09.
https://doi.org/10.1145/3344948.3344988

## 1 INTRODUCTION

According to Big Data Value Association (BDVA) SRIA [2], IoT technology enables the connection of any type of smart devices or objects, and will have profound impact on a variety of sectors in the future. However, the exponential growth of connected devices and mobile computing, the potentially large amounts and complexity of data produced in these systems, as well as the booming trend of Edge and Fog computing, are massive forces that push toward the need to coordinate execution and data exchange among "computational abstract nodes". Such nodes are hosted on a spectrum that surely includes virtual machines in the Clouds, but also physical machines (up to tiny devices like sensors) on the Edge of the network.

Gartner has put Edge computing as one of the top Technology Trends for 2018, characterising Edge computing as the key solution to facilitate data processing at or near the source of data generation: "Currently, around 10% of enterprise-generated data is created and processed outside a traditional centralised data center or Cloud. By 2022, Gartner predicts this figure will reach 50%" [6].

Edge computing serves as the decentralised extension of the campus, cellular and data centre networks, and is rapidly becoming not only integration, but a potential alternative that complements the Cloud. As the volume and speed of data increases, so too does the inefficiency of streaming all this information to a Cloud or data centre for processing. At the same time, operations on the Cloud raise privacy and security concerns, due to the difficulty of creating local boundaries to information flow [13]. In this situation, there are benefits to decentralising computing, to placing it closer to the point where data is generated—in other words, to pursuing Edge computing. Therefore, since the Cloud layer is no longer sufficient to guarantee the efficiency, flexibility, privacy, and security, required by such dynamic networks, new models, software architectures and approaches are needed, operating on Fog/Edge layers to enable key services at a distributed level, primarily related to Big Data management, and the corresponding AI-related technology application.

Historically, network architectures and computing models have swung between the use of shared and central resources and exclusive and local computing power [8]. As of today, available massive distributed deployments of intelligent devices are confronted with the cloud computing model emphasising centralised shared

resources. For computational-intensive long-latency tasks, cloud computing will continue to be used. Yet local processing is becoming increasingly relevant, in particular for managing large amounts of data with higher complexity generated throughout large-scale distributed applications. Hence, a new edge-cloud computing model is currently evolving, which requires the most effective balance between cloud and edge processing.

This position paper is organized as follows: Section 2 presents challenges related to developing DIES, Section 3 presents our overall concept of developing DIES based on *aggregate computing* [3], whereas Section 4 presents key insights on developing DIES.

## 2 CHALLENGES FOR DEVELOPING DIES

Developing *dependable intelligent edge systems* (DIES) requires a paradigm shift in acquiring, storing, and processing large amount of the data with the following key characteristics:

- *Data management* is placed at the edge between the data sources and local processing entities, with loose coupling to storage and processing services located in the cloud.
- *Autonomic and self-adaptation techniques* are used to continuously re-balance and redistribute data and its management processes, as needed to address variability in computational resources and environmental contingencies.
- Critical applications with high degree of *dependability* shall be supported.

DIES allow bringing data (pre-)processing and decision-making close to the data sources, enabling fast and efficient processing and communication with a reduced exposure surface of the data. Edge services become then capable of substituting the cloud power, making cloud processing unnecessary for a growing variety of tasks. With this paradigm shift, we gain flexibility, auto configuration, auto diagnosis, but new challenges emerge. Some of these challenges are described below:

- *Increased Complexity:* The complexity, pervasivity, and number of devices of such new systems will be so high that human designers will only be able to master it with the support of intelligent infrastructures.
- *High Dependability Requirements:* Innovative approaches will be required to ensure that the systems will behave as required, both at functional and non-functional (e.g. responsiveness, reliability, security). We need to develop theories, design tools, and runtime support that go beyond predictability by design and enable the construction of reliable systems involving unreliable parts [7].
- *Need of Novel Models:* Also, a significant challenge is in defining and developing new programming and execution models, together with a model-based engineering approach across the lifetime of system, to handle the diversity and complexity of data, deal with the variety of stakeholder concerns and ensure the trustworthiness of DIES through continuous self-adaptation [10].

Tackling these challenges is crucial to design and program DIES and support the perpetual adaptation and evolution process throughout their lifetime, from inception to operation and evolution [11]. It will be also important to ultimately demonstrate their economic benefits of increased integration/system-wide control as a means of unlocking investment [2].

## 3 NOVEL APPROACH TO DEVELOP DIES

We envision a novel programming and execution model for DIES that allows edge devices to gain complete autonomy. It leverages on concepts derived from *aggregate computing* [3].

### 3.1 Aggregate Computing

Aggregate computing focuses on programming a large, possibly multi-layered network of devices as a single aggregate machine, where a dynamic neighbouring relation represents (physical or logical) proximity. This machine can then be programmed through an API enacting a proper layer of abstraction on top of low-level hardware capabilities.

Conceptually, this approach is hence similar to that of big data engines like `Apache Spark`[12], which brings the "stream" abstraction to an adaptively load-balanced cluster computing deployment. What is different here is that we face a broader scope than mere access to aggregated data. We rely on the notion of "computational field" (or simply field): a global, distributed map from devices to data values, evolving over time as a result of local aggregation/diffusion processes. In particular, computing with fields means computing such global structures in a fully distributed way. This view holds at any level of abstraction, from low-level mechanisms up to complex applications, which ultimately work by getting input fields from sensors and processing them to produce output fields to actuators, hence describing a continuous computational process involving sensors and actuators. Transition from simple to complex services is done thanks to functional composition, which allows one to declaratively express whole computations—as with Spark. For these reasons, it is particularly well suited for modern DIES that demonstrate an increasing level of data processing capacity and complexity, and require high flexibility and scalability.

Field computations are modelled by the *field calculus* [1], a minimal core language that captures the essence of aggregate programming and serves as blueprint for implementations and starting point for verification of behavioural properties. The field calculus semantics prescribes that a given program P is executed by each device of the network, periodically and asynchronously according to a cyclic schedule of rounds whereby each device, when activated: (i) perceives contextual information through sensors; (ii) retrieves local information stored in the previous round, and collects messages received from neighbours while sleeping; (iii) evaluates the program P by manipulating the data values retrieved from neighbours, context or local memory; (iv) stores local data, broadcasts messages to neighbours, and produces output values (possibly fed to actuators); and (v) sleeps until it is awakened at the next activation.

A field calculus program describes a global behaviour emerging from nodes locally interacting with one another, hence the network of devices can be naturally abstracted as a single aggregate machine, in which both the topology of nodes and the sensor readings evolve over time. The data abstraction manipulated by such "aggregate computing machine" is a whole distributed space-time field, associating individual computation events (space-time points stating where and when a device is activated) to the data produced there.

### 3.2 Novel DIES Programming and Execution Model

Aggregate computing and field calculus define the prominent general-purpose framework to build distributed systems that produce high volumes of complex data with an intrinsic level of support for scalability, robustness, and adaptability [4]. They have been used to develop algorithms with guarantees for basic resilience properties, including: distance estimation, selective broadcasts, data collection and summarization, partitioning [5, 9]. This is achieved by the declarative, functional character of field calculus: small building blocks can be proved resilient, and functional composition of resilient blocks can be proved in turn to retain resiliency.

However, the field calculus execution and programming models only provide a basic platform to address the distinguishable needs of DIES. Realising the full potential of DIES needs first-class support for spacial and temporal aspects, in particular abstractions for specifying spatial and temporal guarantees, and binding and optimising the usage of resources, in particular those that produce, process, and consume data. Additionally, current language technologies supporting field computations as prototypical Domain Specific Languages (DSLs), like Protelis (an external Java DSL) and ScaFi (an internal Scala DSL), are only based on Java 7+ JVM, which is not suitable for safety-critical systems in the industrial sectors.

Hence, current approaches to provide higher-level abstractions for mathematically grounded, safety-critical edge computing systems fail in providing general solutions to the problem of guaranteeing real-time constraints.

Our approach draws inspiration from the work in field calculus, and devise a new programming and execution model for edge computations that retains its basic properties of mathematical tractability and functional composition to enable complexity scaling:

- from the modelling side, this will be achieved by devising a calculus of field computations with the power of controlling and optimising the details of when and how computation rounds of devices and message exchange occur, as a means to better balance the usage of computational resources and to more strictly control real-time aspects of computation;
- from the programming development side, we aim at porting the benefits of using the paradigm of aggregate programming to real-time platforms, namely embedded RTOSs (i.e. ERIKA Enterprise) and general-purpose OSs (i.e. Linux), through a C/C++-based implementation as shown in conceptual technology stack in Figure 1;
- from the algorithmic side, we aim to provide libraries to enable supporting autonomy, safety, security, and other dependability properties.

## 4 PERSPECTIVES ON DIES ARCHITECTURE

DIES aims at supporting software development and execution by fully exploiting the power of edge computing, thereby aiming to improve flexibility, scalability, real time response, security and safety. This is achieved by applying the notion of aggregate processes. The approach is underpinned by formal semantics and supported by technologies (libraries and platforms) and a sound methodology for data-oriented service specification, targeting multiple domains.

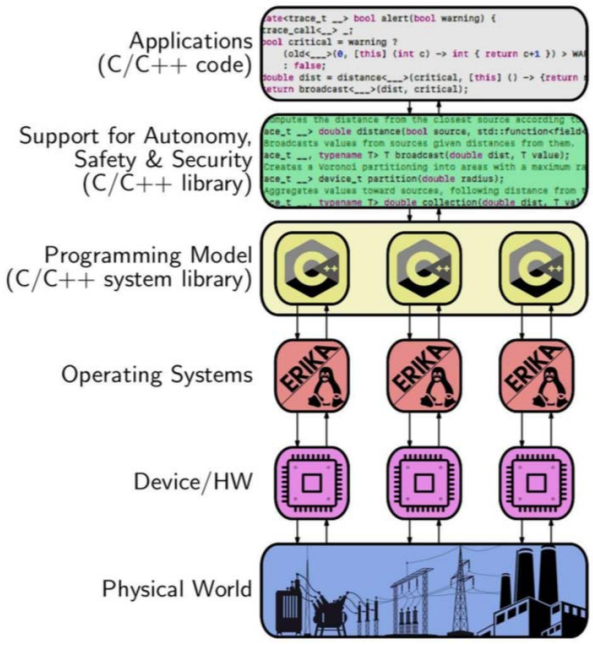


**Figure 1: Proposed Technology Stack for Developing DIES**

The approach has the potential to be integrated with existing architectures.

### 4.1 From the Cloud to the Edge

We aim to evolve the fragmentation of development methodologies and tools in the present cloud computing paradigms, towards an edge paradigm. Currently, software program methodologies are typically based on a traditional client-server architecture, with solutions ranging from pure HTML, browser-side interfaces to heavy, thick or native client applications accessing data and processing services from a centralised back-end. That model faces many potential inefficiencies when dealing with "purely distributed" application scenarios. In the client-server model, one or more clients are connected to a server listening on a pool or set of sockets. This model mainly scales vertically and usually has a central data store (LAN model). A distributed model of N-clients or peers connected to a mesh of M servers scales horizontally and uses a data store (or stores) that also shares and distributes processing. Such model is built to tolerate failure and demand spikes, enabling the network users, developers, application managers, and the infrastructure owners to add more nodes (often linearly) and relocate infrastructure at will—the model of the Cloud. The power of this more distributed model goes beyond purely scaling up to include scaling down. However, workloads in the "purely distributed" scenarios are difficult to predict, but strongly influence the performance, the availability, and the business value of any application. Furthermore, edge technologies bring a new ecosystem to the industry. This new architecture for collecting data with distributed connected things and elaborating them locally enables a new wave of software applications.

### 4.2 The need for quality-driven architectures

The need for new architectures and platforms including tools and methodologies for collecting, storing, managing and analysing all this data is increasingly important. In the Cloud space a plethora of commercial platforms have been recently developed to pursue this opportunity. Public cloud providers (AWS, Azure, Google Cloud and IBM) have recognised the opportunity and developed IoT platforms on top of their infrastructure. Nevertheless, traditional Cloud-based IoT systems are challenged by large scale, heterogeneity, potentially high latency, unpredictable response time from Cloud server to endpoints, privacy issues when sensitive customer data are stored in the Cloud, and difficulties in scaling to ever increasing number of sensors and actuators. The proposed new architecture plays a significant role in this scenario enabling latency sensitive computing performed in proximity to the sensors, resulting in more efficient network bandwidth and more functional and efficient IoT solutions. It also offers greater business agility through deeper and faster insights, increased security and lower operating expenses.

## 5 CONCLUSION

We envisage a new architecture based on aggregate computing which implements the novel concept of dependable intelligent edge systems (DIES), for exploitation in big data management in edge/cloud platforms. However, the development of distributed computing applications for this context can be very challenging for the software programmer and architect. Traditional or ongoing approaches aim to improve distributed processing by offloading parts to centralised, Cloud-based server, but do not provide the expected performance gains due to the intense communication required and lack of localised computing power sharing. Our architecture will provide models and libraries to natively support programming on distributed, dynamically changing devices and accelerate software integration, validation and deployment. This new architecture will achieve dynamicity and flexibility of distributed applications, supporting the programmers to manage the related increasing complexity.

## REFERENCES


[1] Giorgio Audrito, Mirko Viroli, Ferruccio Damiani, Danilo Pianini, and Jacob Beal. 2019. A Higher-Order Calculus of Computational Fields. *ACM Transactions on Computational Logic (TOCL)* 20, 1, Article 5 (2019), 55 pages. https://doi.org/10.1145/3285956

[2] Big Data Value Association (BDVA). 2019. Strategic Research Agenda (SRIA). (2019). http://www.bdva.eu/SRIA

[3] Jacob Beal, Danilo Pianini, and Mirko Viroli. 2015. Aggregate Programming for the Internet of Things. *IEEE Computer* 48, 9 (2015), 22–30. https://doi.org/10.1109/MC.2015.261

[4] Jacob Beal, Mirko Viroli, Danilo Pianini, and Ferruccio Damiani. 2017. Self-Adaptation to Device Distribution in the Internet of Things. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)* 12, 3 (2017), 12:1–12:29. https://doi.org/10.1145/3105758

[5] Soura Dasgupta and Jacob Beal. 2016. A Lyapunov analysis for the robust stability of an adaptive Bellman-Ford algorithm. In *55th IEEE Conference on Decision and Control (CDC)*. IEEE, 7282–7287. https://doi.org/10.1109/CDC.2016.7799393

[6] Gartner. 11/2018. Top 10 Strategic IoT Technologies and Trends. (11/2018). www.gartner.com/en/newsroom/press-releases/2018-11-07-gartner-identifies-top-10-strategic-iot-technologies-and-trends

[7] HiPEAC. 2019. HiPEAC Vision. (2019). https://www.hipeac.net/vision/

[8] International Electrotechnical Commission (IET). 2017. White Paper. (2017). www.iec.ch/whitepaper/pdf/IEC_WP_Edge_Intelligence.pdf

[9] Mirko Viroli, Giorgio Audrito, Jacob Beal, Ferruccio Damiani, and Danilo Pianini. 2018. Engineering Resilient Collective Adaptive Systems by Self-Stabilisation. *ACM Transactions on Modelling & Computer Simulation* 28, 2 (2018), 16:1–28. https://doi.org/10.1145/3177774

[10] D. Weyns. 2019. *Software Engineering of Self-adaptive Systems*. Springer International Publishing, Cham, 399–443. https://doi.org/10.1007/978-3-030-00262-6_11

[11] D. Weyns et al. 2017. Perpetual Assurances for Self-Adaptive Systems. In *Lecture Notes in Computer Science vol. 9640*. Springer. https://doi.org/10.1007/978-3-319-74183-3_2

[12] M. Zaharia et al. 2016. Apache spark: a unified engine for big data processing. *Commun. ACM* 59, 11 (2016), 56–65. https://doi.org/10.1145/2934664

[13] J. Zhang, B. Chen, Y. Zhao, X. Cheng, and F. Hu. 2018. Data Security and Privacy-Preserving in Edge Computing Paradigm: Survey and Open Issues. *IEEE Access* 6 (2018), 18209–18237. https://doi.org/10.1109/ACCESS.2018.2820162