NebulaStream: Data Management for the Internet of Things

Steffen Zeuch[1,2*], Ankit Chaudhary[1,2*], Viktor Rosenfeld[1,2*], Taha Tekdogan[1,2*], Adrian Michalke[1,2*], Matthis Gördel[1,2*], Ariane Ziehn[1,2*] , Volker Markl[1,2,3*]
TU Berlin[1], BIFOLD[2], DFKI GmbH[3], Germany
firstname.lastname@(tu-berlin.de[1], dfki.de[3])

ABSTRACT

Soon, more data will be produced outside the cloud than inside. This constantly increasing amount of data requires new systems that can holistically optimize the processing for so-called sensor-fog-cloud environments. In this tutorial, we present NebulaStream, the first system designed specifically for this unified environment. We highlight research challenges inherent in managing data across a unified sensor-fog-cloud environment. After setting the scene, we dive into NebulaStream's architecture and system internals, which integrate cloud, fog, and sensor networks to meet these demands effectively. Finally, in a hands-on session, attendees can experience NebulaStream in action, learn how to configure and run IoT applications on the platform, and extend NebulaStream with custom functionalities. By the end of the tutorial, attendees will have a deeper understanding of NebulaStream and will be equipped with helpful guides to use it as a framework to try out their own ideas and benefit from its unique features.

1 INTRODUCTION

Over the last decade, the amount of produced data has reached unseen magnitudes. The International Data Corporation [15] estimated that by 2025, the global amount of data will reach 175 ZB and that 30% of this data will be gathered in real-time. One primary driver of this development is the Internet of Things (IoT). In particular, the number of IoT devices is expected to grow to as many as 20 billion connected devices by 2025 [8]. At the same time, devices such as embedded computers or mobile phones continuously increase their processing capabilities. This trend enables exploiting their computing and communication capabilities as they become objects of common use. As a result, the IoT is one of the fastest emerging trends in the area of information and communication technology [12].

The explosion in the number of connected devices leads to the emergence of novel data-driven applications such as traffic congestion monitoring, smart street lighting, or vehicle pollution control [1]. These applications require low-latency processing, location awareness, wide-spread geographical distribution, and real-time data processing on potentially millions of distributed data sources such as sensors. To enable those applications, a data management system needs to leverage the capabilities of IoT devices. However, today's data management systems are not yet ready for these applications as they embrace either the cloud or the fog computing paradigm. In particular, three main types of processing systems have been originated. First, systems based on the cloud paradigm, e.g., Flink [2], Spark [18], or Kafka Streams [16]. These systems require collecting all sensor data centrally in a data center before applying any processing. Thus, cloud-based systems can not exploit the full capabilities of IoT devices and struggle to meet the low-latency requirements of IoT applications. For this reason, the centralized processing paradigm presents a bottleneck for IoT applications, which need to process data from millions of distributed sensors. Second, systems based on the fog computing paradigm, e.g., Frontier [13] and CSA [17], exploit the processing capabilities of edge devices, i.e., IoT devices that are physically closer to the data sources (sensors). These devices apply data reduction techniques, e.g., pre-selection or pre-aggregation, to reduce data volume as early as possible in the processing pipeline, i.e., close to the sensor. However, fog computing systems only scale within the fog and do not exploit the virtually unlimited resources of modern cloud infrastructures, such as Amazon Web Services or Microsoft Azure. Third, data management systems for wireless sensor networks (WSNs), e.g., TinyDB [10], exploit small battery-powered sensors and create a network of nodes to capture physical phenomena, such as earthquakes or volcanic eruptions. These systems apply acquisitional query processing techniques to optimize the execution for battery lifetimes and deploy a small set of specialized queries. However, WSN systems only scale within the sensor networks and do not exploit the resources of the attached cloud and fog environments. In particular, they do not consider offloading computation to external nodes and do not provide general-purpose query execution capabilities.

In this tutorial, we present NebulaStream [19], which is the first general-purpose, end-to-end data management system for a unified sensor-fog-cloud environment. In Session 1 (see Section 4.1), we highlight the research challenges we tackled and how we overcame them. After that, we present the system internals of NebulaStream. In Session 2 (see Section 4.2), we offer hands-on experiences on NebulaStream. We divide this session into two parts. In the first part, we present existing IoT use cases and show how their queries are submitted and how those queries are processed by NebulaStream. In the second part, we highlight how to use NebulaStream for research. We show how easy it is to benchmark and modify the system and highlight our extensive online documentation. Overall, we motivate the need for a novel data management system solution for the rising IoT and highlight current open issues and bottlenecks. Furthermore, we reveal to the attendees how a data management system for the sensor-fog-cloud environment differs from a data management system for the cloud, which is the state-of-the-art solution for emerging large and dynamic IoT applications. Then, we provide hands-on experience on how to use NebulaStream and incorporate one's research and ideas.

*All authors contributed equally to this work.

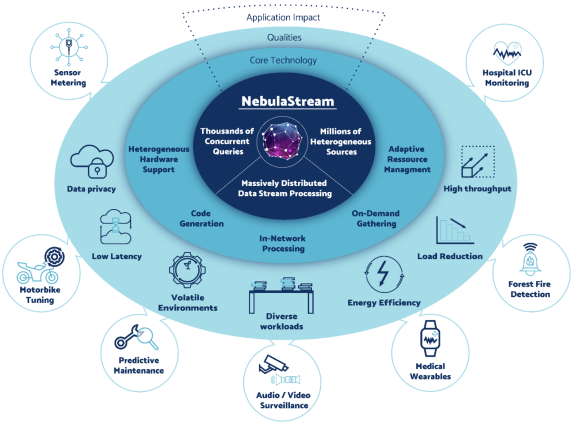

Figure 1: NebulaStream at a Glance.

2 ABOUT NEBULASTREAM

As a preliminary, we present NebulaStream at a glance in Figure 1 and briefly dive through the four layers (from inner to outer layer) in the following:

IoT and Data Management: New Challenges, New Solutions. Data management systems will face new challenges supporting IoT applications in the upcoming years. The rise of IoT requires the management of millions of diverse IoT devices spread across large areas, continuously generating huge amounts of data. This trend enables application scenarios with thousands of concurrent, long-running queries, which requires efficient handling of massively distributed data stream processing. For this reason, it is essential to combine and leverage the strength of each individual layer in a unified sensor-fog-cloud environment. NebulaStream addresses this by researching, innovating, and integrating various technologies, including cloud, fog, and sensor networks, to create this environment and facilitate the development of current and foreseeable IoT applications.

Core Technologies. In order to tackle the challenges of the IoT, NebulaStream is built on five core technologies. First, unified environments are comprised of heterogeneous processing nodes, ranging from resource-constrained low-end devices to powerful cloud servers. NebulaStream supports various devices, including different architectures (e.g., ARM, x86) and accelerators (e.g., GPUs or DPUs). Second, NebulaStream compiles queries into highly efficient hardware-tailored code, which increases hardware utilization and significantly reduces energy consumption [7, 11]. Third, NebulaStream utilizes in-network processing to leverage the available resources inside the fog and on devices close to the sensors to reduce the computational demands in the cloud. Fourth, by only gathering data that is actually required by the users, NebulaStream further reduces the amount of data that is sent to the cloud. Fifth, to address the dynamics of the network topology, NebulaStream does not consider operator placement a one-shot task; instead, it adapts to changes in the query workload and the topology.

Qualities. Due to the diversity of IoT applications, NebulaStream incorporates different qualities, e.g., efficiency qualities such as low latency and high throughput, as well as additional relevant concerns in the area of IoT. For instance, by exploiting the sharing

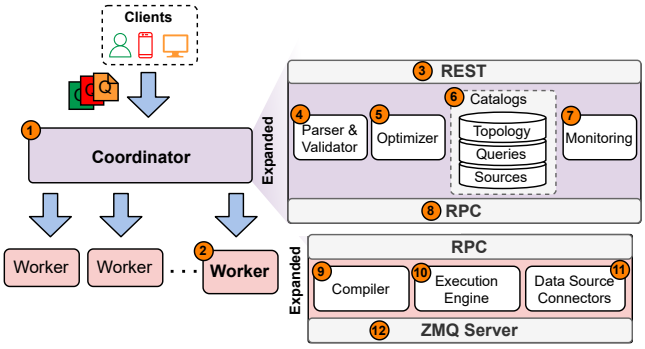

Figure 2: NebulaStream Architecture.

opportunities of operators or reducing data transmitted over the network, NebulaStream can reduce the energy consumption of worker nodes. Thus, it can extend the lifetime of battery-powered devices. Other qualities of concern are load reduction and coping with a volatile environment.

Application Impact. As a result of our current research, we have employed NebulaStream to build IoT applications in several research projects, where we explored the challenges that arise with the IoT and the system itself. Examples of these applications are smart water meter monitoring, audio/video surveillance, ECG monitoring, and motorbike tuning in ELEGANT[1]; predictive quality control in ExDRA[2] [3]; and hospital ICU monitoring in RIDIMP[3] [21] and NEEDMI [9].

3 NEBULA STREAM ARCHITECTURE

In the following, we introduce some key concepts and details of the NebulaStream architecture, focusing on the components relevant to our tutorial. For more detailed descriptions, we refer the attendees to the following papers [14, 19, 20]. We present the NebulaStream architecture in Figure 2. In general, NebulaStream is designed with a centralized manager and decentralized executors that leverage resources attached to the arbitrary hierarchical topology of a unified sensor-fog-cloud environment.

The NebulaStream topology consists of two types of nodes: coordinator and worker. The central component orchestrating and managing decentralized query processing and serving management requests is the *coordinator* (1). Users can interact with NebulaStream via our NebulaStream UI, Java, Python, or a REST client to submit queries, configure the topology, or add new sources over a REST interface (3). Upon the arrival of a new query, the coordinator uses the parser to compute logical query plans and the validator to perform semantic and syntactic validation of the queries (4). The initial query plans go through the optimizer (5) for further optimizations as the next step. This essential component applies rewrite rules to generate an optimized query plan to reduce data transfer and remove redundant operations. Additionally, the optimizer creates a global query plan leveraging the sharing potentials among newly arriving and already running queries [6]. In the end, the optimizer performs the query placement optimization utilizing

[1] https://www.elegant-h2020.eu
[2] https://www.exdra.de
[3] https://ki-sigs.de/projekt/AP%20330

information from the topology catalog, such as neighboring nodes and resource availability, and creates the query execution plan [5]. This execution plan consists of the mapping of query plans to the physical location where their processing needs to be scheduled. The coordinator maintains a catalog of registered workers, queries, and data sources that can be analyzed by the system (6). The coordinator explores this catalog during the orchestration of newly arriving queries. The monitoring component [4] keeps track of running queries, collects statistics, and health checks registered workers to continuously update the catalog (7). In the end, the coordinator deploys query plans from the execution plan using the RPC interface to the geo-distributed workers (8).

The geo-distributed *workers* act as the compute nodes in NebulasStream (2). These nodes receive the optimized logical plans from the coordinator and use the compiler [7] to generate highly optimized machine code (9). The worker then computes tasks using the generated code and schedules them for processing in its execution engine (10). The execution engine uses a work-stealing model to efficiently and concurrently process all scheduled tasks. Some workers can also act as data source providers by connecting to external systems to generate tuples for processing. To this end, workers support consuming data from heterogeneous sources (e.g., MQTT, OPC, Kafka) using embedded connectors (11). These workers use ZeroMQ to transmit data among each other and ultimately deliver the processed data to the cloud data center (12). Overall, the coordinator optimizes, orchestrates, and manages the lifecycle of running queries, and workers efficiently process data by generating hardware-tuned code for deployed queries.

4 TUTORIAL OUTLINE AND CONTEXT

Duration and Target Audience. We divide the NebulaStream tutorial into two 90-minute sessions, with each two parts of approximately 45 minutes. With this tutorial, we reach out to a broad target audience, including computer scientists, researchers, and software developers with a background or interest in system engineering, stream processing, and the challenges of the IoT. The sessions will be largely self-contained for a stream-processing audience. In Section 2 and 3, we provide preliminaries and refer to helpful literature for attendees unfamiliar with stream processing and their architecture. Additionally, we provide tutorials and guides that cover the content of Session 2 in the documentation section on the NebulaStream website[4]. Furthermore, experience with object-oriented programming is a plus for the second part of Session 2.

In the reminder, we outline the content of each session. In particular, Session 1 (see Section 4.1) is dedicated to getting to the introduction to data management for the IoT and NebulaStream internals. In Session 2, we guide the attendees through hands-on examples of using NebulaStream as an end-user and as a developer or researcher (see Section 4.2).

4.1 Session 1: Data Management for the IoT and NebulaStream Internals

In this session, attendees are getting to know sensor-fog-cloud environment and NebulaStream.

In the first part, we will introduce the sensor-fog-cloud environment as a new processing environment for the future and outline how it differs from today's dominant cloud environment. We then present the challenges that arise in this new environment and early approaches how to address them. We will use existing and new slides to present the content. After this part, attendees should be aware of the state of the art in this area and open research questions.

In the second part, we will introduce NebulaStream and its architecture, as well as its unique features. We will highlight the overall system architecture and current proposals to address current research challenges. We will use existing slides that were used and refined many times in our university courses. After this part, attendees will be aware of the underlying architecture of NebulaStream and its unique capabilities.

4.2 Session 2: Hands-on NebulaStream

In this session, attendees will learn how to use NebulaStream. To this end, over the entire session, several members of the NebulaStream team will be present to assist the attendees. The session is structured in two parts:

In the first part, we will present NebulaStream from an end-user perspective and show how to configure it and run applications on it. We will first show how to download a binary NebulaStream image, configure sensor sources and the network, and run NebulaStream inside a Docker container. Then, we will provide a set of real-world workloads to showcase how to run NebulaStream queries. We will use these workloads as application-based scenarios to explore the capabilities of NebulaStream as a stream processing system. These applications cover a wide range of workloads such as ETL, join, aggregation, etc. Furthermore, we show the capabilities of NebulaStream's graphical user interface and how it can be used to execute queries, display the topology, and visualize the results using plots. Finally, we outline how existing workloads on other SPEs, such as Apache Flink, can be migrated to NebulaStream and how they benefit from its unique features. We will use the current code of NebulaStream, our documentation, and workloads generated by several projects.

In the second part, we will present NebulaStream from the perspective of a developer or researcher. To this end, we will show the attendees how to implement and run benchmarks in NebulaStream in order to compare its performance with other systems. Then, we will show how to modify operators to perform customized processing easily. In particular, we will highlight the usage of our novel Nautilus Compiler [7] that allows us to write and debug interpreted code while at the same time achieve the same performance as compiled code. Additionally, we have created six implementation guides as part of this tutorial that attendees can run independently after this session. In particular, we provide the following step-by-step implementation guides, all available in the developer guides on our website[4]: (1) How to add a new operator? (2) How to add a new expression? (3) How to add a new query rewrite rule to the query optimizer in order to manipulate the query plan? (4) How to add a new source? (5) How to add a new operator placement algorithm? (6) How to add a new query to the NebulaStream benchmarking framework?

[4] https://nebula.stream

After this session, attendees will have a deeper understanding of how to integrate their own work with NebulaStream and will have everything they need to either use NebulaStream to solve their IoT use cases or extend it with their own research ideas. Again, several members of the NebulaStream team will be present throughout the session to assist participants.

5 BIOGRAPHICAL SKETCHES OF PRESENTERS AND AUTHORS

NebulaStream is currently under development by an active team of researchers at TU Berlin, BIFOLD, and DFKI GmbH, led by **Volker Markl** as head of the DIMA research group and BIFOLD director. In the following, we introduce the five presenters by session:

Session 1, Part 1: Steffen Zeuch has been a Senior Researcher at DIMA since 2017 and is the project lead and system architect of the NebulaStream project. He has been conducting research on modern hardware, IoT, and streaming, published at international venues, such as SIGMOD or VLDB.

Session 1, Part 2: Ankit Chaudhary has been a PhD student at DIMA since 2019. He has been conducting research on distributed stream processing systems and query optimization with a focus on performing multi-query optimization, query placement, and adaptive query optimization in distributed stream processing systems.

Session 2, Part 1: Viktor Rosenfeld has been a Post-Doc at DIMA since 2023. Before, he was a PhD student at DIMA, working on data processing on modern hardware and heterogeneous systems, e.g., containing CPUs and GPUs, as well as data processing in heterogeneous programming language environments. He has managed the integration of NebulaStream in the EU research project ELEGANT and has worked on supporting user-defined functions in NebulaStream

Taha Tekdogan has been a PhD student at DIMA since 2023. His current focus is the benchmarking and performance evaluation of distributed stream data processing systems for heterogeneous environments.

Session 2, Part 2: Adrian Michalke has been a PhD student at DIMA and the BIFOLD Graduate School since 2022. He is researching the management of historic stream data.

Ariane Ziehn has been a PhD student at DIMA since 2020, focusing her research on the intersection of stream and complex event processing for unified fog-cloud environments and stream processing optimization.

Matthis Gördel is a Master's student at TU Berlin and a student assistant at DIMA since 2024.

All authors belong to the team of the NebulaStream project and will be present during Session 2 to support attendees.

ACKNOWLEDGMENTS

This work was funded by the German Federal Ministry of Education and Research as BIFOLD - Berlin Institute for the Foundations of Learning and Data (ref. 01IS18025A and ref. 01IS18037A). We thank all members of the NebulaStream team and our amazing students for their contributions, insightful comments, and fruitful discussions.

REFERENCES

[1] Arif Ahmed, HamidReza Arkian, Davaadorj Battulga, Ali J. Fahs, et al. 2019. Fog Computing Applications: Taxonomy and Requirements. *CoRR* (2019).
[2] Alexander Alexandrov et al. 2014. The stratosphere platform for big data analytics. *VLDB Journal* (2014).
[3] Sebastian Baunsgaard, Matthias Boehm, Ankit Chaudhary, Behrouz Derakhshan, et al. 2021. ExDRa: Exploratory Data Science on Federated Raw Data. In *SIGMOD*, Guoliang Li, Zhanhuai Li, Stratos Idreos, and Divesh Srivastava (Eds.). ACM.
[4] Xenofon Chatziliadis, Eleni Tzirita Zacharatou, Steffen Zeuch, and Volker Markl. 2021. Monitoring of Stream Processing Engines Beyond the Cloud: An Overview. *Open J. Internet Things* (2021).
[5] Ankit Chaudhary, Steffen Zeuch, and Volker Markl. 2020. Governor: Operator Placement for a Unified Fog-Cloud Environment. In *EDBT*.
[6] Ankit Chaudhary, Steffen Zeuch, Volker Markl, and Jeyhun Karimov. 2023. Incremental Stream Query Merging. In *EDBT*.
[7] Philipp M Grulich, Aljoscha P Lepping, Dwi PA Nugroho, Varun Pandey, Bonaventura Del Monte, Steffen Zeuch, and Volker Markl. 2024. Query Compilation Without Regrets. *Proceedings of the ACM on Management of Data* 2, 3 (2024), 1–28.
[8] Mark Hung. 2017. Leading the IoT, Gartner Insights on How to Lead in a Connected World. *Gartner Research* (2017).
[9] Aljoscha P. Lepping, Hoang Mi Pham, Laura Mons, Balint Rueb, et al. 2023. Showcasing Data Management Challenges for Future IoT Applications with NebulaStream. *Proc. VLDB Endow.* (2023).
[10] Samuel R Madden, Michael J Franklin, Joseph M Hellerstein, and Wei Hong. 2005. TinyDB: An Acquisitional Query Processing System for Sensor Networks. In *ACM Transactions on database systems (TODS)*. ACM.
[11] Adrian Michalke, Philipp M. Grulich, Clemens Lutz, Steffen Zeuch, and Volker Markl. 2021. An Energy-Efficient Stream Join for the Internet of Things. In *DaMoN*, Danica Porobic and Spyros Blanas (Eds.). ACM.
[12] Daniele Miorandi, Sabrina Sicari, Francesco De Pellegrini, and Imrich Chlamtac. 2012. Internet of things: Vision, applications and research challenges. In *Ad Hoc Networks*.
[13] Dan O'Keeffe, Theodoros Salonidis, and Peter Pietzuch. 2018. Frontier: Resilient Edge Processing for the Internet of Things. In *VLDB*.
[14] Elena Beatriz Ouro Paz, Eleni Tzirita Zacharatou, and Volker Markl. 2021. Towards Resilient Data Management for the Internet of Moving Things. In *BTW*.
[15] D Reinsel, J Gantz, and J Rydning. 2018. Data age 2025: The Digitization of the World from Edge to Core. Retrieved December 15, 2019, from https://www.seagate.com/files/www-content/our-story/trends/files/idc-seagate-dataage-whitepaper.pdf.
[16] Matthias J. Sax, Guozhang Wang, Matthias Weidlich, and Johann-Christoph Freytag. 2018. Streams and Tables: Two Sides of the Same Coin. In *Proceedings of the International Workshop on Real-Time Business Intelligence and Analytics (BIRTE '18)*.
[17] Zhitao Shen, Vikram Kumaran, Michael J Franklin, Sailesh Krishnamurthy, et al. 2015. CSA: Streaming Engine for Internet of Things.. In *IEEE Data Eng. Bull.*
[18] Matei Zaharia, Reynold S Xin, Patrick Wendell, et al. 2016. Apache Spark: a unified engine for big data processing. In *Communications of the ACM*.
[19] Steffen Zeuch, Ankit Chaudhary, Bonaventura Del Monte, Haralampos Gavriilidis, et al. 2020. The NebulaStream Platform for Data and Application Management in the Internet of Things. In *CIDR*.
[20] Steffen Zeuch, Eleni Tzirita Zacharatou, Shuhao Zhang, Xenofon Chatziliadis, et al. 2020. NebulaStream: Complex Analytics Beyond the Cloud. *Open J. Internet Things* (2020).
[21] Ariane Ziehn, Christian Mandel, Kathrin Stich, Rolf Dembinski, et al. 2022. IoT-PMA: Patient Health Monitoring in Medical IoT Ecosystems. *Open J. Internet Things* (2022).

Received 20 February 2007; revised 12 March 2009; accepted 5 June 2009