NebulaStream: An Extensible, High-Performance Streaming Engine for Multi-Modal Edge Applications

Adrian Michalke
Aljoscha Lepping
Volker Markl
Ricardo Martinez
Nils Schubert
Lukas Schwerdtfeger
Taha Tekdogan
Steffen Zeuch
Ariane Ziehn
michalke@campus.tu-berlin.de
aljoscha.p.lepping@tu-berlin.de
volker.markl@tu-berlin.de
r.martinez.ramirez@tu-berlin.de
n.schubert.1@tu-berlin.de
lukas.schwerdtfeger@campus.tu-berlin.de
tekdogan@tu-berlin.de
steffen.zeuch@tu-berlin.de
ariane.ziehn@campus.tu-berlin.de
Berlin Institute for the Foundations of Learning and Data (BIFOLD)
Berlin, Germany

Christoph Falkensteiner
Kyle Krüger
Alexander Meyer
Tobias Röschl
Svea Wilkending
christoph.falkensteiner@dhzc-charite.de
kyle-steven.krueger@charite.de
alexander.meyer@dhzc-charite.de
tobias.roeschl@dhzc-charite.de
svea.wilkending@dhzc-charite.de
Charité Berlin
Berlin, Germany

Abstract

NebulaStream is a novel, open-source data stream processing system for massively distributed, heterogeneous data streams in the cloud-edge continuum. It adheres to the design goals of ease-of-use, extensibility, and efficiency to provide a framework for users and developers to implement diverse Internet of Things (IoT) use cases. Equipped with essential built-in functionalities, NebulaStream allows users to customize the system easily while ensuring efficient execution even on low-end devices.

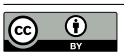
In this demonstration, we highlight NebulaStream's ability to integrate and process multi-modal, multi-frequency data streams. We showcase its abilities through a real-world IoT scenario where NebulaStream is used to improve the health assessment of patients in a smart intensive care unit.

CCS Concepts

• **Information systems → Stream management**.

Keywords

Streaming, IoT, Edge, Cloud



SIGMOD-Companion '25, Berlin, Germany

ACM ISBN 979-8-4007-1564-8/2025/06
https://doi.org/10.1145/3722212.3725118

ACM Reference Format:
Adrian Michalke, Aljoscha Lepping, Volker Markl, Ricardo Martinez, Nils Schubert, Lukas Schwerdtfeger, Taha Tekdogan, Steffen Zeuch, Ariane Ziehn, Christoph Falkensteiner, Kyle Krüger, Alexander Meyer, Tobias Röschl, and Svea Wilkending. 2025. NebulaStream: An Extensible, High-Performance Streaming Engine for Multi-Modal Edge Applications. In *Companion of the 2025 International Conference on Management of Data (SIGMOD-Companion '25), June 22–27, 2025, Berlin, Germany*. ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3722212.3725118

1 Introduction

Data in the IoT is becoming more and more diverse, generated by heterogeneous devices in different formats, frequencies, protocols, etc. The heterogeneity of the IoT data presents a challenging problem for near real-time streaming applications with respect to converting, aligning, and combining the data streams, particularly in dynamically changing environments that require the regular integration of new streams, formats, or data analysis algorithms.

In our collaboration at the Berlin Institute for the Foundations of Learning and Data (BIFOLD) with the Deutsches Herzzentrum der Charité (DHZC), Europe's largest University Hospital, we have seen the above challenges in patient monitoring, specifically in data collection, analysis, and alerting tasks. For example, intensive care units (ICUs) have a growing number of patient data sources, which vary from sensors with low frequency that measure blood pressure every five minutes to high-speed ECG sensors that produce time-series with millisecond granularity. To assess the health of a patient, these fast data streams are combined with very slowly moving data

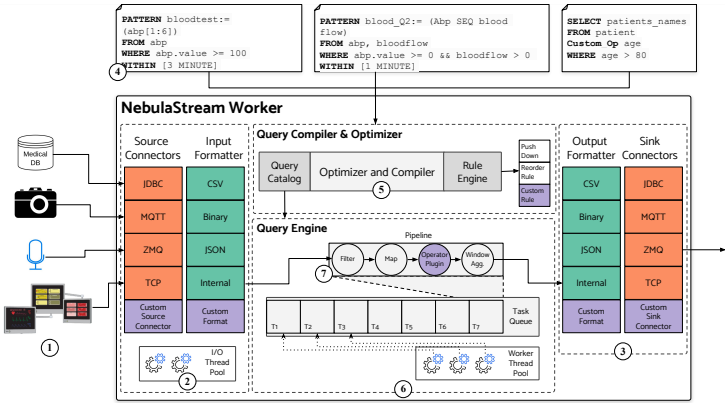


Figure 1: NebulaStream Architecture.

streams, such as lab work and assessments by nurses or doctors, which are often provided daily. Furthermore, the integration of more recent experimental modalities, such as camera or audio data, in the analysis and alerting process of the ICU could be used to detect the deterioration of a patient's health even earlier. In today's ICUs, these data sources remain disconnected, requiring medical staff to manually combine available information in a time-consuming and error-prone process.

In a potential *smart* ICU, all available data sources could be seamlessly integrated into an SPE, making it possible to analyze complex events in near-real-time to enhance the accuracy, speed, and efficiency of assessing a patient's personalized health status. From a system perspective, today's SPEs are not built for IoT applications and their particular needs [8]. In particular, they induce the following problems.

High System Complexity: Providing a user-friendly interface for a complex system that processes multi-modal, multi-frequency streams requires a simple language to express interrelated events while optimizing execution automatically. However, modern SPEs lack built-in functionality for such diverse streams, relying on user-defined functions (UDFs) that are neither user-friendly for domain experts nor as efficient as built-in functions. In particular, users must independently handle tasks such as stream alignment, data pre-processing, event specification, or ML integration. By requiring programming skills, deep system knowledge, and expertise in diverse areas from the user, the system becomes inaccessible to domain experts and challenging to operate effectively when the state-of-the-art advances with respect to adding new data streams or alerting algorithms.

Limited Extensibility: Providing all possible functionality out-of-the-box is economically not feasible. Thus, a system has to offer a high degree of extensibility to the user. However, only some SPEs provide a restricted set of extension possibilities. In general, the interaction of sources and formats and their efficient integration into the processing are left to the user. Furthermore, additional custom operators and their optimizations are hardly possible.

Low Efficiency: IoT-related applications require processing on edge devices with limited capability. The processing on those devices requires a high degree of efficiency as they are usually connected to many high-speed sensors. However, today's SPEs neglect this aspect: they rely on managed runtimes and process data in a hardware-oblivious fashion [9]. Furthermore, SPEs cannot reason about what happens inside a UDF, which severely limits query optimization and thus efficient processing.

In this demonstration, we present NebulaStream, an SPE designed to handle multi-modal, multi-frequency IoT workloads efficiently. NebulaStream is built around the design goals of ease-of-use, extensibility, and efficiency to provide a robust framework for users and developers to implement diverse IoT applications. To achieve this, it offers essential built-in functionalities, such as alignment and inference, to reduce manual and error-prone efforts when combining diverse streams. Additionally, NebulaStream enables users to seamlessly integrate custom formats, sources, and functionalities to tailor the system to specific requirements, while supporting two declarative languages to specify workloads over multi-modal, multi-frequency streams. As a real-world example, we demonstrate how its design goals enable the creation of a smart ICU.

2 Overview of NebulaStream

In this section, we present the design of NebulaStream that addresses problems of state-of-the-art SPEs for IoT applications. In short, NebulaStream is built on top of three design goals.

DG1) Ease-of-Use: NebulaStream provides out-of-the-box functionality for common tasks required by multi-modal, multi-frequency streams, e.g., alignment or inference. This enables users to focus on business logic with well-known abstractions and concepts.

DG2) Extensibility: NebulaStream empowers users to easily integrate custom data connectors, formats, operators, and optimizations into the system.

DG3) Efficiency: NebulaStream utilizes distributed heterogeneous computing devices, including low-end ones, with hardware-tailored code, adaptive execution, and the interleaved processing of data sources to handle large workloads efficiently.

In Figure 1, we present the client-server architecture of NebulaStream. However, we restrict the presentation in this paper to the worker side and single-node execution, excluding distributed collaboration among nodes. We refer the reader to our system paper [8] for the overall description of the system. On the left (see ①), different sources are connected to a NebulaStream Worker. These sources send their data to the Source Manager in binary or custom formats (see ②). For commonly used source connectors, e.g., JDBC, MQTT, or TCP, the Source Manager provides built-in connectors

(**DG1**). In addition, users can add custom source connectors without having to change system internals (**DG2**), e.g., for different standards like *Service-oriented Device Connectivity (SDC)* as a connector for medical devices or *Open Platform Communications (OPC)* as a platform-independent service-oriented connector.

Unlike other SPEs that handle sources individually and synchronously by assigning one thread per source, NebulaStream interleaves source processing via thread sharing within its own I/O thread pool and applies asynchronous callbacks to reduce waiting time. As a result, NebulaStream can handle many sources more efficiently and guarantees fairness and progress among them (**DG3**). After receiving data from the Source Connectors, the system converts the data into an internal format by either using a built-in formatter (**DG1**) or a custom formatter (**DG2**). The same logic applies to the Sink Manager (see ③) that also allows for custom connectors and formatters.

On the upper side of Figure 1, query submission examples are shown (see ④) using NebulaStream's two input languages. First, using a SQL-like query language with streaming extensions, the user can express queries to extract and transform data from raw streams by applying common data manipulation (e.g., filter or project), data transformation (e.g., map), data grouping by time or count using windowing, and stream join operations. Furthermore, NebulaStream supports additional operations out-of-the-box, such as alignment with resampling or interpolation and inference from ML models (**DG1**). Second, using a pattern specification language (PSL), users can express complex event patterns with logical, temporal, or causal relationships. These patterns are the backbone of many applications, enabling the detection of interesting events and trends in the data. When such events are detected, the system can trigger notifications (e.g., alerts) to signal when specific conditions are met.

After query submission, NebulaStream transforms and optimizes all queries from both languages into a single *global query plan*. The Query Optimizer applies common optimization rules from the Rule Engine (see ⑤), such as filter push-down, but also allows users to specify custom rules (**DG2**). To generate code, the Query Compiler chunks the global query plan into pipelines and lowers them into an internal representation. After that, the compiler calls the Nautilus Framework, a lightweight and adaptable just-in-time (JIT) compiler, to generate hardware-tailored code [2] (**DG2**). Finally, the generated code is placed in the Query Catalog.

During runtime, the Query Engine (see ⑥) schedules query processing in a highly dynamic manner using the abstraction of tasks (**DG3**). Every task encapsulates one processing step (i.e., a compiled pipeline) on one data item (i.e., a tuple buffer from the source). For processing, worker threads dequeue tasks from the Task Queue whenever they have capacity. As shown in the example pipeline (see ⑦), a pipeline might contain a mix of custom and built-in operators. At the end of each pipeline, the result is either emitted to the Sink Manager for output or a new task is created if further processing is required. This highly dynamic execution model imposes many benefits compared to static resource assignments used in today's systems. In particular, it dynamically and implicitly balances the load. As a result, NebulaStream can quickly react to fluctuating workloads and avoid idling resources that may arise due to a static work assignment in a highly dynamic environment (**DG3**).

In sum, the architecture of NebulaStream provides a framework with many built-in functionalities to enable a wide range of application scenarios out-of-the-box (**DG1**). In addition, it provides a high degree of extensibility to developers for customization to enable an even larger set of use cases (**DG2**). Finally, the highly efficient design allows NebulaStream to process large amounts of data with minimal resource requirements (**DG3**).

3 Demonstration

In this demonstration, we highlight how NebulaStream can be used to build emerging IoT applications. To this end, we showcase a *smart* ICU that we are developing together with the DHZC.

ICU Setup. Our demonstration setup consists of three components. First, a smart ICU bed equipped with the four medical data sources (see Figure 2a). Second, a visualization of the raw streams on the left side of Figure 2b. Third, an analysis of these streams through queries submitted to NebulaStream combined with a smart alerting that detects diseases and proposes possible treatments (see right side of Figure 2b). In the following, we briefly introduce each device and its relevance to clinical research and smart alerts.

❶ A thermal camera captures RGB and infrared frames to detect facial features, segment them, and track the heatmaps. In clinical research, this method has shown promise in the early detection of hemodynamic shock (HS), a life-threatening event [5].

❷ A microphone records audio to classify sounds. In clinical research, audio of lung and breathing sounds is used to detect diseases, such as acute respiratory distress syndrome (ARDS) or chronic obstructive pulmonary disease (COPD) [1, 4, 7].

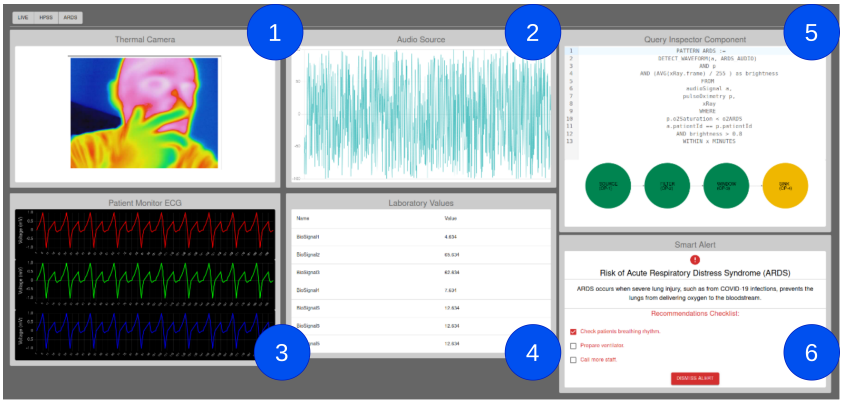
❸ A patient monitoring device (PMD) continuously records the patients' vital parameters like the heart rate from the electrocardiogram signal or blood oxygen saturation (SpO_2) from a photoplethesmograph (PPG). In clinical practice, these metrics are good indicators to detect an acute change in patient health [3].

❹ A patient data management system (PDMS) provides infrequent data, like laboratory results, x-ray images, or medical staff assessments. In clinical practice, this information is combined with the acute events of the PMD to assess the health status of a patient.

Clinical relevance and challenges: In today's clinical practice, the PDMS is the primary source of information for physicians. In addition, audio signals, thermal imaging, and high-frequency biosignals have shown value in detecting life-threatening conditions, such as HS and ARDS. However, their timely accessibility remains challenging due to high sampling rate requirements and complex integration needs. NebulaStream aims to assist medical staff with an intelligent co-pilot by integrating all relevant sources. In particular, it provides all of the required functionality out-of-the-box, enables the required extensions of data connectors and operators, and enables the processing of common ICU setups on low-end devices next to the bed. As a result, NebulaStream relieves medical staff from the labor-intensive and error-prone manual work of gathering and analyzing all information and can aid in the early detection of life-threatening events.

ICU Scenarios. To showcase NebulaStream's ability to aid in early detection of critical health statuses and enhance patient care, we present two ICU scenarios.

(a) Physical Setup.



(b) Visualization.

Figure 2: Demonstration.

HS Detection. Shock is defined as an imbalance between cellular oxygen supply and demand. It is often triggered by infections (septic) or heart failure (cardiogenic). Normally, shock is primarily detected through tachycardia and hypotensive blood pressure values as the shock index. Expanding this monitoring to include thermal images could provide a life-saving time advantage [6].

Using NebulaStream, medical staff can automatically monitor thermal imaging data, blood pressure values, and pulse oximetry waveforms for HS indicators. Since the thermal camera provides RGB and infrared streams at different speeds, NebulaStream has to align them before model inference. A HS is detected if 1) the blood pressure value is over a particular threshold, and 2) the pulse oximetry waveform shows a particular pattern, and 3) the thermal camera model detects reduced peripheral perfusion.

ARDS Detection. ARDS occurs when a severe lung injury, such as from COVID-19 infections, prevents the lungs from delivering oxygen to the bloodstream. Certain causes of ARDS produce distinct breathing sounds that can be detected using a combination of signal processing and machine learning to analyze the audio of digital stethoscopes [7]. In addition, automated analysis of X-ray images of the lung could assist the diagnosis. Using NebulaStream, medical experts can combine audio signals, x-ray image analysis, and pulse oximetry data to support early ARDS detection and enhance patient care. An ARDS is detected if 1) the x-ray image shows that a large lung area is covered white, and 2) the pulse oximetry waveform shows a particular pattern, and 3) if the learned model detects an abnormal breathing pattern.

Attendee Experience. The demo experience is divided into two parts, i.e., a hands-on experience to explore the smart ICU scenarios and their challenges, and an under-the-hood experience to explore the technical details of NebulaStream.

Part 1: Hands-on. The attendees can take on the role of a medical staff member and explore the ICU setup. We allow the attendees to interact and get familiar with the ICU bed and the attached devices and observe their live visualizations on the dashboard. Furthermore, attendees can explore the two prepared medical scenarios, i.e., HS and ARDS. To this end, we visualize the underlying queries (see 5) and present the smart alerts that include treatment recommendations (see 6). Note that we use historical data for these queries to show data associated with pathology instead of presumably healthy attendees. This part demonstrates how NebulaStream integrates and processes multi-modal, multi-frequency data streams for real-time analytics.

Part 2: Under-the-Hood. We invite attendees to dive deeper into NebulaStream's architecture and explore with the NebulaStream Team how it addresses the complexities of multi-modal data processing. Leveraging ICU scenarios, attendees will gain a deeper understanding of NebulaStream's ability to tackle IoT challenges, such as stream alignment, scalability, and real-time processing.

Acknowledgments

This work was funded by the German Federal Ministry for Education and Research as BIFOLD—Berlin Institute for the Foundations of Learning and Data (ref. 01IS18025A and ref. 01IS18037A). We want to thank the following students for their valuable contributions: Niklas Tantow, Tilmann Dietzel, Tim Nacken, Felix Taschner, Luca Gaedicke, Lily Seidl, and Tarik Abu Mukh. Finally, we want to thank Grigorii Turchenko for contributing to this demonstration.

References

[1] Angelo Calabrese, Davide Chiumello, Martina Gurgitano, et al. 2020. Use of digital auscultation in patients with ARDS: a correlation study with CT imaging. *European Respiratory Journal* 56, suppl 64 (2020).
[2] Philipp M Grulich, Aljoscha P Lepping, Dwi PA Nugroho, et al. 2024. Query Compilation Without Regrets. *Proceedings of the ACM on Management of Data* 2, 3 (2024), 1–28.
[3] João Jorge, Mauricio Villarroel, Hamish Tomlinson, et al. 2022. Non-contact physiological monitoring of post-operative patients in the intensive care unit. *npj Digital Medicine* 5, 1 (Jan. 2022).
[4] Sung Hoon Lee, Yun-Soung Kim, Min-Kyung Yeo, et al. 2022. Fully portable continuous real-time auscultation with a soft wearable stethoscope designed for automated disease diagnosis. *Science Advances* 8, 21 (2022), eabo5867.
[5] Aditya Nagori, Lovedeep Singh Dhingra, Ambika Bhatnagar, et al. 2019. Predicting Hemodynamic Shock from Thermal Images using Machine Learning. *Scientific Reports* 9 (2019), 91.
[6] Alejandra Ortiz-Dosal, Eleazar S. Kolosovas-Machuca, Rosalina Rivera-Vega, et al. 2014. Use of infrared thermography in children with shock: A case series. (2014).
[7] Ruchi Sharma, Menglian Zhou, Mohamad Hakam Tiba, et al. 2022. Breath analysis for detection and trajectory monitoring of acute respiratory distress syndrome in swine. *ERJ Open Research* 8, 1 (2022).
[8] Steffen Zeuch, Ankit Chaudhary, Bonaventura Del Monte, et al. 2020. The NebulaStream Platform for Data and Application Management in the Internet of Things. In *CIDR 2020*. www.cidrdb.org.
[9] Steffen Zeuch, Bonaventura Del Monte, Jeyhun Karimov, et al. 2019. Analyzing efficient stream processing on modern hardware. *Proc. VLDB Endow.*