SoftwareX 7 (2018) 156–161

Contents lists available at ScienceDirect

SoftwareX

journal homepage: www.elsevier.com/locate/softx

Original software publication

MIDAS: Open-source framework for distributed online analysis of data streams

Andreas Henelius [a,b,*], Jari Torniainen [a,c]

[a] Finnish Institute of Occupational Health, Helsinki, Finland
[b] Department of Computer Science, Aalto University, Helsinki, Finland
[c] Department of Applied Physics, University of Eastern Finland, Kuopio, Finland

ARTICLE INFO

Article history:
Received 11 January 2016
Received in revised form 12 April 2018
Accepted 16 April 2018

Keywords:
Data streams
Online analysis
Distributed systems
Machine learning

ABSTRACT

Data streams are pervasive but implementing online analysis of streaming data is often nontrivial as data streams can have different, domain-specific formats. Regardless of the stream, the analysis task is essentially the same: features are extracted from the stream, e.g., to employ machine learning and data mining methods. We present the Modular Integrated Distributed Analysis System (MIDAS) for constructing distributed online stream processing systems for heterogeneous data. The MIDAS framework makes it possible to process raw data streams, extract features, perform machine learning and make the results available through an HTTP API for easy integration with various applications. MIDAS is agnostic with regard to the type of data stream and is suitable for multiple domains.

Code metadata

Current code version	v1.1.0
Permanent link to code/repository used of this code version	https://github.com/ElsevierSoftwareX/SOFTX-D-16-00010
Legal Code License	MIT (http://opensource.org/licenses/MIT)
Code versioning system used	git
Software code languages, tools, and services used	Python3.4+
Compilation requirements, operating environments & dependencies & Compilers:	Operating systems: Linux, Windows, OS X. Python modules: Bottle, PyZMQ, Waitress, PyLSL
If available Link to developer documentation/manual	https://github.com/bwrc/midas/wiki
Support email for questions	andreas.henelius@aalto.fi, jari.torniainen@uef.fi

1. Introduction and significance

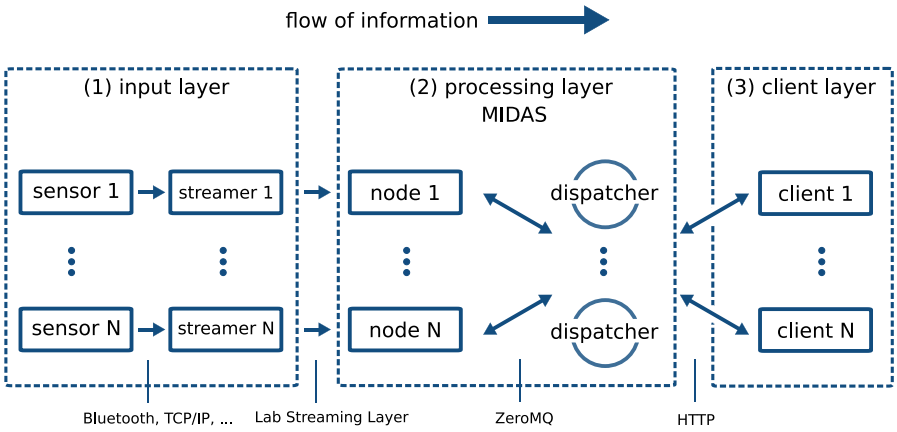
Devices in the Internet of Things (IoT) are found in various fields, e.g., in healthcare [1] or in environmental and agricultural applications [2]. IoT is also relevant in Human–Computer Interaction (HCI) (e.g., biosignals for controlling user interfaces).

Many IoT data streams are *time series signals* with a volume ranging from one channel sampled at 1 Hz (e.g., a temperature sensor) to 16 channels of electroencephalographic (EEG) data sampled at 500 Hz. Time series must be processed sequentially sample-by-sample in contrast to batch processing of data, where the processing order of data items is less important. Regardless of the source of data streams, potentially important information can be extracted from them. Data streams in different domains have varying properties, but the data processing task is essentially identical: features of interest are extracted from the streams and used in decision making. There is hence a need for *stream processing systems* for online extraction and analysis of streaming signals, that can handle various data formats and high data rates. Since a typical task is to simultaneously collect data from multiple sensors this suggests a distributed system for balancing the computational load.

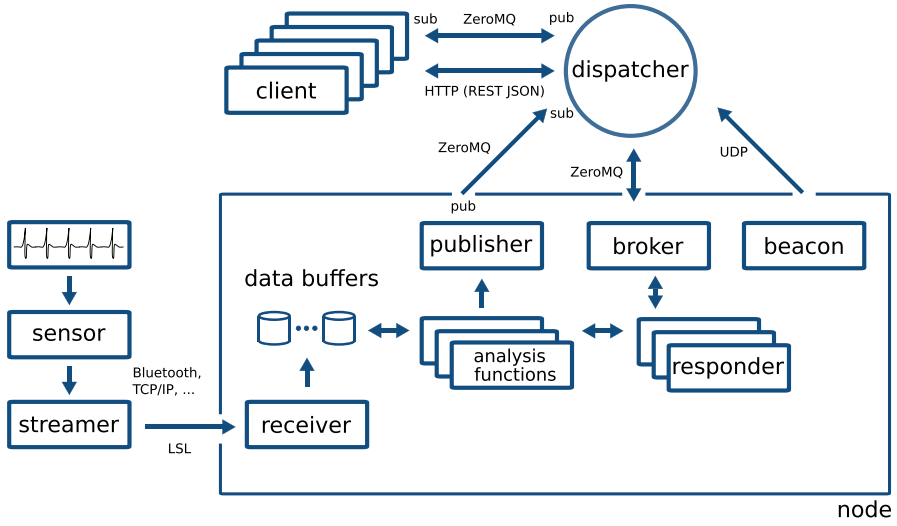
Current stream processing systems are either domain-specific (e.g., Brain–Computer Interface (BCI) frameworks such as BCI2000 [3], OpenViBE [4] or BCILAB [5]) or automation systems (e.g., QIVICON (https://www.qivicon.com/) or EPICS (http://www.aps.anl.gov/epics/)). Such systems are monolithic and optimised for processing a small number of parallel data streams arriving at a high

* Corresponding author at: Department of Computer Science, Aalto University, Helsinki, Finland.
E-mail addresses: andreas.henelius@aalto.fi (A. Henelius), jari.torniainen@uef.fi (J. Torniainen).

https://doi.org/10.1016/j.softx.2018.04.004



(a) The three-layered architecture showing the components and the communication protocols. Information flows from left (sensor) to right (client). MIDAS comprises the processing layer with *nodes* and the *dispatcher(s)*.



(b) Main units inside the node and communication protocols used between components.

Fig. 1. Architecture of MIDAS.

rate in a particular domain. There are also generic data processing frameworks for handling huge amounts of incoming data at a high rate, e.g., Twitter data streams, at massive scales (e.g., Storm (http://storm.apache.org/), Samza (http://samza.apache.org/) and Spark Streaming (https://spark.apache.org/streaming/)). Finally, there are dedicated IoT platforms (e.g., General Electric Predix (http://https://www.predix.com/), PTC Thingworx (https://www.thingworx.com/) and Amazon AWS IoT (https://aws.amazon.com/iot/) or Kaa (https://www.kaaproject.org/)). Although the distributed data processing systems and the IoT systems are highly capable they are in many cases unnecessary complex to set up for smaller-scale analyses with few data streams, e.g., for prototyping.

The focus of this paper is the cross-platform MIDAS (Modular Integrated Distributed Analysis System) framework, primarily engineered to handle high-velocity time series originating from wearables and IoT sensors for use in HCI applications. The philosophy of the MIDAS framework is to help researchers create and manage setups with multimodal signal sources, enabling them to focus on the signal processing and machine learning aspects of the system. The key design consideration was to create a flexible stream processing system with a distributed architecture for load balancing that is easy to set up and which scales to different types of hardware, such as laptops, desktops and single-board computers.

MIDAS was engineered with the following design goals in mind, common to systems for online analysis of data streams regardless of the domain.

The system is *data agnostic*, i.e., different types of data streams can be analysed since streams may provide complementary information. The system is *modular* and is composed of small autonomous, interconnected units, making it simple to add and remove data streams and analysis components, speeding up the workflow. The *distributed* architecture ensures scalability and distribution of computational load. *Accessibility* is provided using an API built on top of standard protocols, e.g., clients to the system can use HTTP. MIDAS is written in Python and is *lightweight*, e.g., it can be installed in seconds.

MIDAS does not aim to compete with domain-specific or dedicated high-performing stream processing systems. MIDAS aims to be a free, open-source, lightweight alternative to the other frameworks for creating IoT systems, e.g., setting up real-time analysis pipelines for multimodal time-series data for prototyping or research.

2. Software architecture

Handling data streams essentially consists of (i) data input, (ii) data processing (feature extraction) and (iii) serving data to