Connecting data streams with On-Demand Services in the Alpine Environmental Data Analysis Centre

J. Munke[1], A. Götz[1], H. Heller[1], S. Hachinger[1], D. Laux[2], O. Goussev[3], J. Handschuh[2,3], S.Wüst[3], M. Bittner[2,3], R. Mair[4], B. Wittmann[4], T. Rehm[5], I. Beck[5], M. Neumann[5]

[1]Leibniz Supercomputing Centre (LRZ) of the Bavarian Academy of Sciences & Humanities, Garching b. München, Germany
[2]University of Augsburg, Institute of Physics, Augsburg, Germany
[3]German Aerospace Center (DLR), Earth Observation Center, Weßling, Germany
[4]bifa Umweltinstitut GmbH, Augsburg, Germany
[5]Environmental Research Station Schneefernerhaus (UFS), Zugspitze, Germany

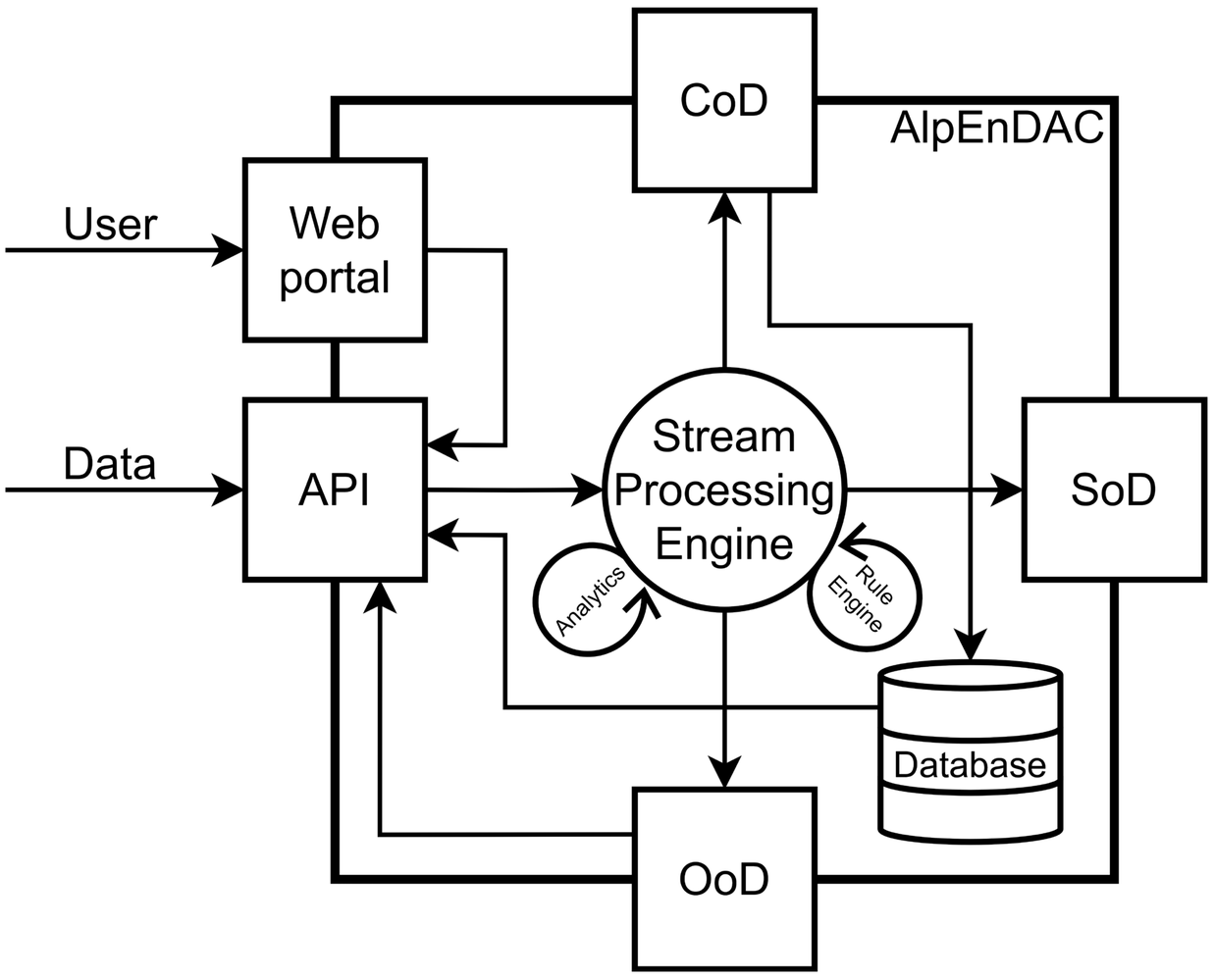


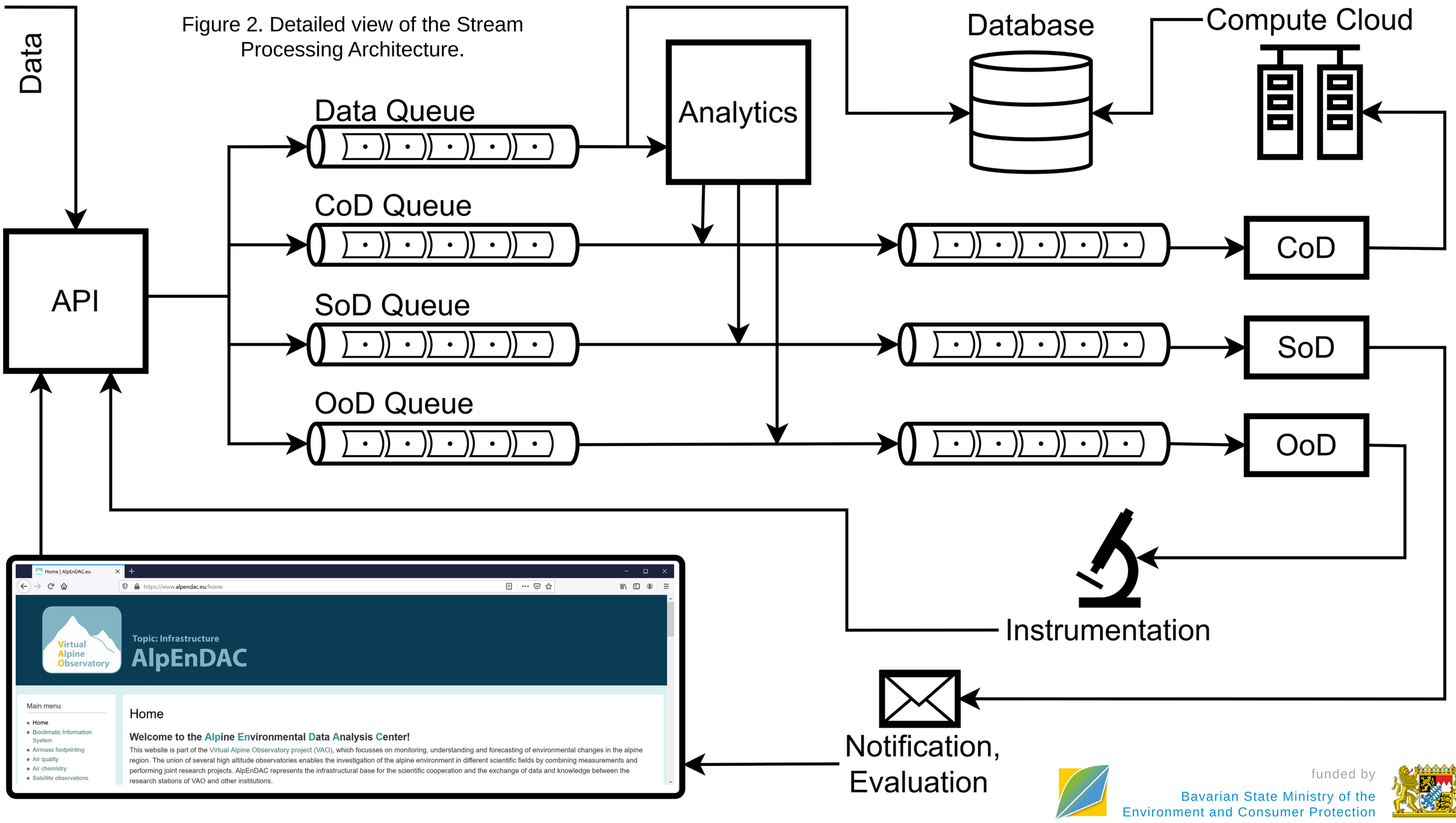
Figure 1. Overview of the upgraded architecture of the AlpEnDAC system.

The **Alpine Environmental Data Analysis Center** (AlpEnDAC) is a "one-stop-shop" platform (www.alpendac.eu) for scientific data measured on high-altitude research stations in the alpine region and beyond. It provides **research data management**, **analysis** and **simulation services** and supports the research activities of the VAO (Virtual Alpine Observatory) community. With some new developments, we want to make environmental scientists profit from **near-real-time** (NRT) **data collection** and **processing**, as it is already an everyday tool e.g. in the Internet-of-Things sector and in commercial applications.

The system will be extended by a **Stream Processing Architecture** consisting of novel components for **Computing on Demand** (CoD), **Service on Demand** (SoD) and **Operating on Demand** (OoD). These will help to implement a **NRT decision support** for the scientist during measurement processes and a **better control** of the measurement process. Fig. 1 shows an overview of the data flow within AlpEnDAC. A more detailed view of the Stream Processing Engine is outlined in Fig. 2.

Data is ingested into the system via a **Representational State Transfer Application Programming Interface** (REST API) or the AlpEnDAC **web portal**. The API will be written in Python using the Flask microframework. The data is stored in a PostgreSQL **database** and is subjected to a NRT analysis. This is implemented with Apache Kafka **message queues**. Based on **user preferences,** CoD and SoD jobs are initially published in dedicated message queues and executed afterwards by a series of **specialized workers**. This allows the **automatic triggering** of **simulations** on the LRZ compute cloud or **evaluation** and **notification services** in NRT.

Figure 2. Detailed view of the Stream Processing Architecture.



funded by
Bavarian State Ministry of the Environment and Consumer Protection

J. Munke: johannes.munke@lrz.de
Leibniz-Rechenzentrum
der Bayerischen Akademie der Wissenschaften

Topic: Infrastructure
AlpEnDAC