D10.3 Open Source Project DAPHNE Reference Implementation

Integrated Data Analysis Pipelines for Large-Scale Data Management, HPC, and Machine Learning

Version 5.0

PUBLIC

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 957407.

Document Description

This document serves as the accompanying description of the demonstrator deliverable D10.3 that provides the open source reference implementation of DAPHNE.

D10.3 Open Source Project DAPHNE Reference Implementation			
WP10 – Dissemination and Exploitation			
Type of document	DEM	Version	5.0
Dissemination level	PU		
Lead partner	KNOW		
Authors Reviewers	Eva Paulusberger (KNOW), Mark Dokter (KNOW) Patrick Damme (TUB), Piotr Ratuszniak (INTP)		

Revision History

Version	Revisions and Comments	Author / Reviewer
V1.0	Initial structure	Eva Paulusberger
V2.0	Write-up	Mark Dokter
V3.0	Reviewer remarks	Patrick Damme, Piotr Ratuszniak
V4.0	Integration of feedback	Mark Dokter
V5.0	Finalization and submission	Eva Paulusberger

Abbreviations

Abbreviation	Definition
DM	Data Management
DSL	Domain Specific Language
FPGA	Field Programmable Gate Array
GPU	Graphics Processing Unit
HPC	High-Performance Computing
HW	Hardware
IDA	Integrated Data Analysis
ML	Machine Learning
MLIR	Multi-Level Intermediate Representation
PR	Pull request

1 Introduction

In this deliverable, we describe the release of a DAPHNE open source reference implementation. This encompasses the strategy behind it, the methodology we apply to realize this endeavor and the actual content of a released artifact. One core concept in DAPHNE's methodology is being open and inclusive in our development process. Consequently, it appeared to us that starting the practice of releasing DAPHNE into open source on a continuous basis early on was more beneficial than waiting for it to mature behind the curtains. By doing so, we gained early feedback and insights into users' needs. Furthermore, this simplified the collaboration with undergraduate students (e.g., at TUG/KNOW, TUB, TUD, UNIBAS, ICCS, ITU, ...) and external collaborators in general. It also created more visibility early on (e.g., we have several stars from external people on GitHub). We moved the open source release from M42 to M18 in accordance with our EU project officer.

For this reason, many aspects of the open source release have been dealt with already in deliverables D10.1 [1] and D10.2 [2] and we may on the one hand refer to these previous deliverables directly but on the other hand also give a summary of these aspects in this document to achieve an easily comprehensible presentation of the content. While creating the refined dissemination and exploitation plan in deliverable D10.1 [1], we gave insight into the open source strategy of DAPHNE with details about the processes that aim at a successful delivery of the source code and release artifacts on the GitHub platform and an adoption strategy that shall bring together the communities of DM, HPC and ML systems researchers. In deliverable D10.2 [2], we further elaborated on the community building around DAPHNE and its open dissemination activities.

The remainder of this deliverable is structured as follows:

- In Section 2 we describe our software release process.
- Then, we list the communication and distribution channels in Sections 3 and 4.
- We summarize past releases in Section 5 and give an outlook to release highlights of the upcoming 0.3 release in Section 6
- Section 7 explains where to obtain the deliverable artifact.
- We give an overview of documentation and starting pointers to dive into DAPHNE in Section 8.
- We describe a small example use case of DAPHNE in Section 9
- Section 10 shows an overview of the source repository's directory structure, describing the most important source locations
- In Section 11 we conclude the report with an outlook into the future of DAPHNE

2 Release Procedure

In deliverable D10.1 [1] we already documented the process we follow to create a new release version of the DAPHNE reference implementation. The original estimate was a release cadence of six months. Retrospectively, we released a new version of the DAPHNE software stack about every 9 - 10 months. Reasons for this change include favoring code quality over a high number of releases which would have been rushed and had been at an immature state in some situations. Another reason has been that some features take longer to develop than others, which would not yield a good set of release highlights if we blindly held on to the initial time frame. It also takes a certain effort initiating a release process, which would be more justified with noteworthy features in a release. Also, the time to do experiments with finished features would influence the release cycle as DAPHNE is at this stage predominantly a research project and therefore not driven by time-to-market priorities.

We still follow the procedure described previously that after technical discussions of the contributors, mainly in the form of code review of pull requests on GitHub, we identify a set of major and minor changes for the next release and enter a period of increased documentation writing, testing, and polishing. The changes are brought into the main branch by participants that earned write access to the main repository by contributing three or more non-trivial commits. After we cut a release candidate, more testing of this candidate is done and the committers (e.g., contributors with write access) may cast a vote for or against making this the final version to release. The creation of the release artifacts has been managed by the coordinator KNOW so far, but this role is open to be filled by any volunteer with the necessary expertise and access privileges.

3 Communication Channels

With our social media activities on LinkedIn [3] and Twitter/X [4] (see screenshots below) we reach out to the interested public, especially researchers and practitioners in the field of DM, HPC, and ML. We not only use these channels to provide updates to the project's followers but also use them as means of getting direct feedback. Naturally, these channels serve the purpose of announcing when a new DAPHNE version is released. This usually takes the form of thanking all contributors and directing the reader to the release page on GitHub where detailed change logs, a list of contributors and release artifacts can be found. Furthermore, we now will also point out the repositories for the Docker container images on DockerHub [5] and the DaphneLib python packages on **PyPI** [6].

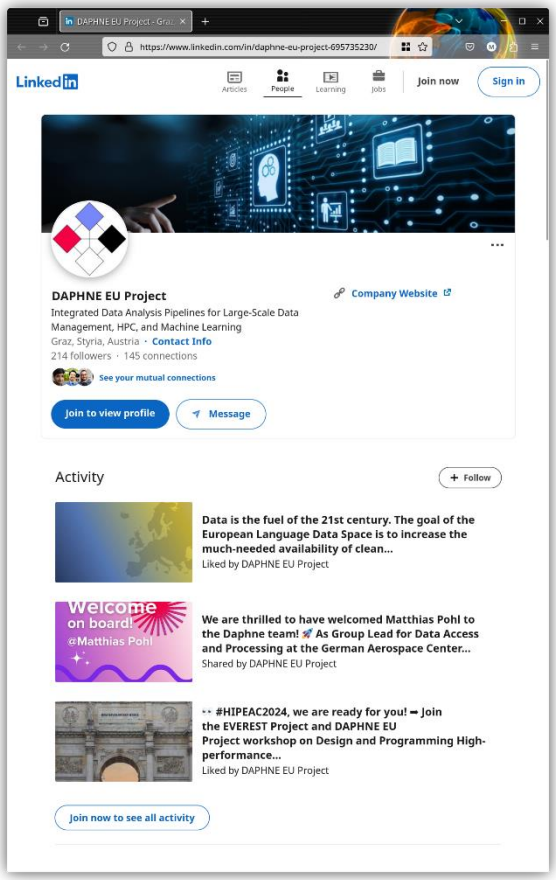

Figure 1DAPHNE LinkedIn Page

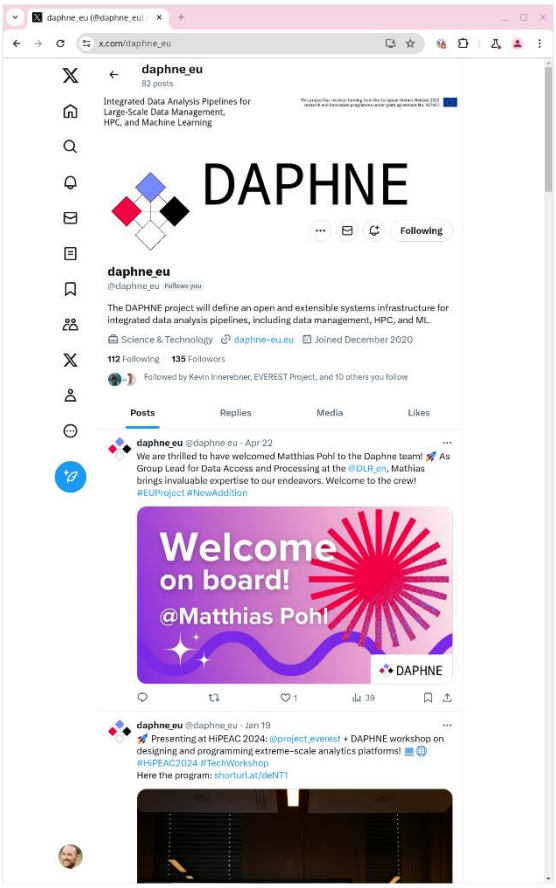

Figure 2 DAPHNE Twitter/X Page

4 Distribuition channels

1. **GitHub** – Our main distribution channel of DAPHNE is, of course, the source code repository [7] that resides on GitHub since the transition from the internal GitLab (which is still in use by our use case partners to work on topics involving data not to be shared in public). In this GitHub account (called "daphne-eu") we have several repositories for various sub-topics. At the time of writing these are the following:
    a. *daphne* – source code and place of code review discussion, issue tracking, release artifact downloads and documentation (the latter in raw markdown form),
    b. *daphne-eu.github.io* – the online documentation that is generated from the markdown of the main daphne repository
    c. *umlaut* – the source code of our benchmarking framework, used to test (not only) DAPHNE,
    d. *reproducibility* – a collection of experiments and everything needed to reproduce results from papers published in the context of DAPHNE,

e. *supplemental-binaries* – a repository to place large binary files into that would otherwise unnecessarily increase the size of the main DAPHNE source repository. Most prominently, this includes the precompiled bitstreams for running operations on an Intel PAC D5005 FPGA [8].

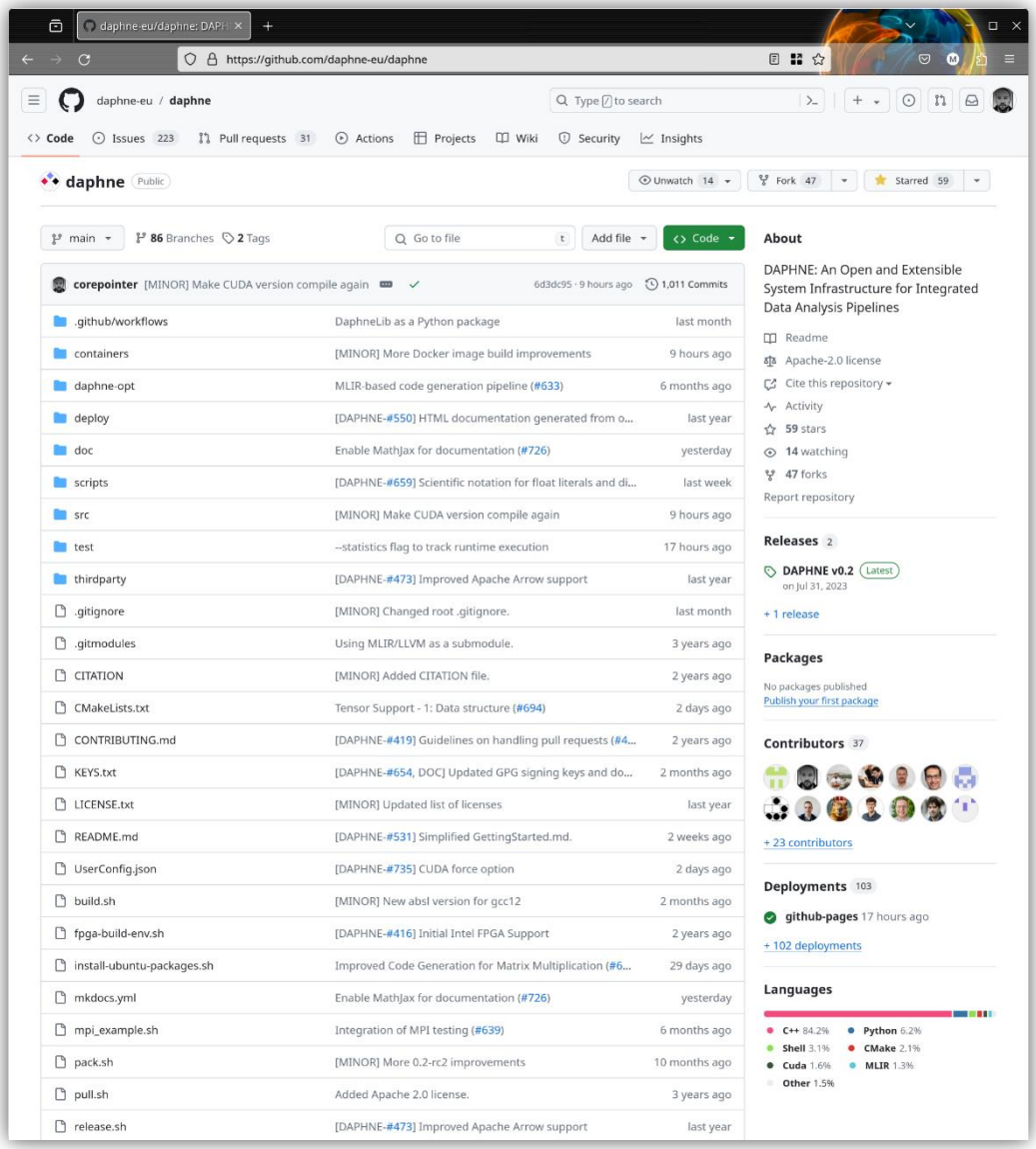

Figure 3 DAPHNE GitHub page

2. **DockerHub** – After starting to produce Docker container images [5] to have a stable execution environment for DAPHNE pipelines and also to ease development of DAPHNE internals, we created an account on DockerHub (called daphneeu as DockerHub would not allow a dash ("-") in the account name) to distribute these images. These DAPHNE images are extremely helpful to get started with DAPHNE development quickly as they not only provide an environment that is known to work

but also save the individual developer from compiling all the required software dependencies, which can take a considerable amount of time on "normal" desktop computers. Our DockerHub account contains the following "flavors" of images:

a. **daphneeu/daphne**: Precompiled DAPHNE binaries and dependencies. Available for X86-64 and ARMv8 in the BASE (supporting all C++ operations implemented for local and distributed CPU kernels) and X86-64 in the CUDA (supporting GPU accelerators) variant,
b. **daphneeu/daphne-dev**: This one is available in the same configurations as above and features precompiled dependencies to focus on only building the DAPHNE binaries from source. Furthermore, this image contains, as a convenience feature, an ssh daemon for remote development,
c. **daphneeu/github-actions**: This image is used by our continuous integration on GitHub to compile DAPHNE upon filing a PR for inclusion in the main branch

d. These Docker images are available with tags for the latest in BASE and CUDA, tags by date of creation and tags based on the version of a release artifact (to have a Docker container accompanying the released binary artifacts).

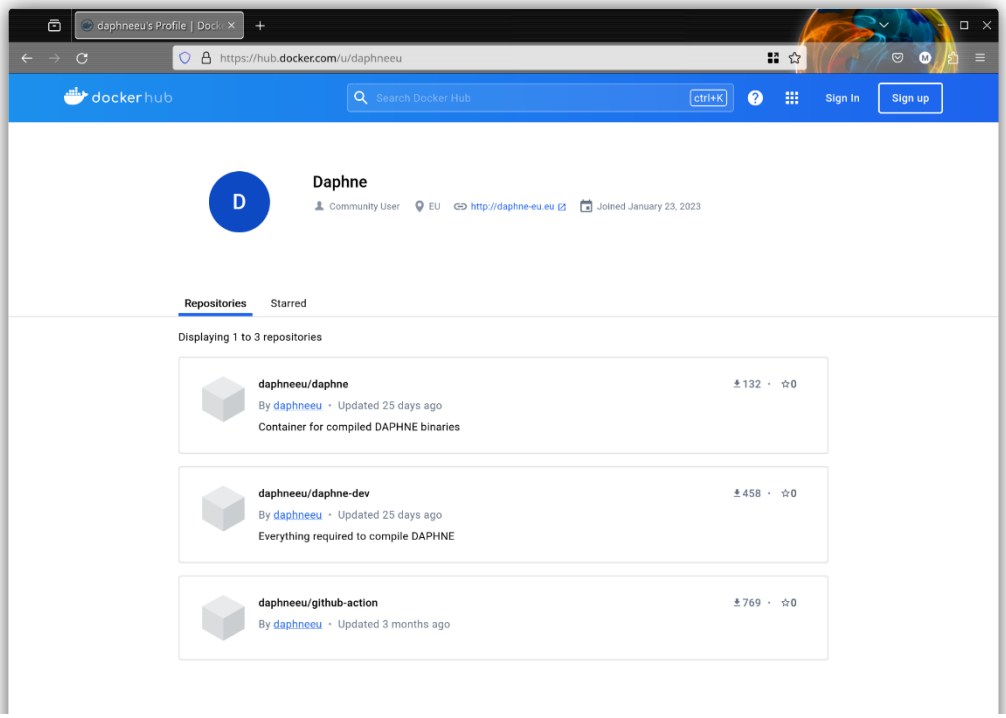

Figure 4 DAPHNE Repositories

3. **PyPI** – Another repository [6] has been added recently to the DAPHNE distribution channels. This time for the python packages of DaphneLib. At the time of writing, this repository is yet to be filled to be part of the upcoming release v0.3.

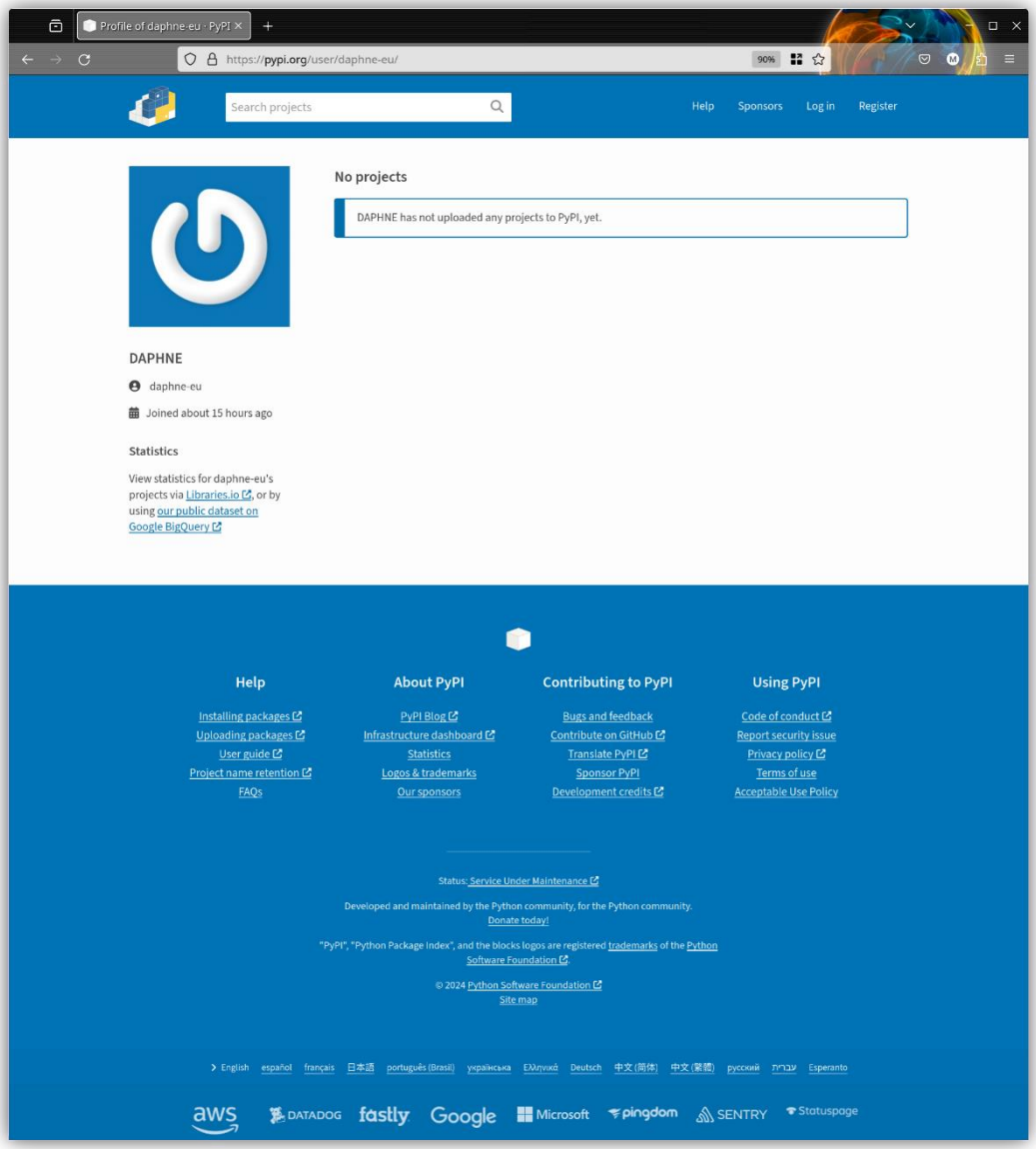

Figure 5 DAPHNE PyPI Page

4. **Online Documentation** – also hosted on GitHub, we auto generate the documentation [9] for DAPHNE with *Material for MkDocs* from the markdown files in the main source repository. This makes the documentation more accessible and searchable.

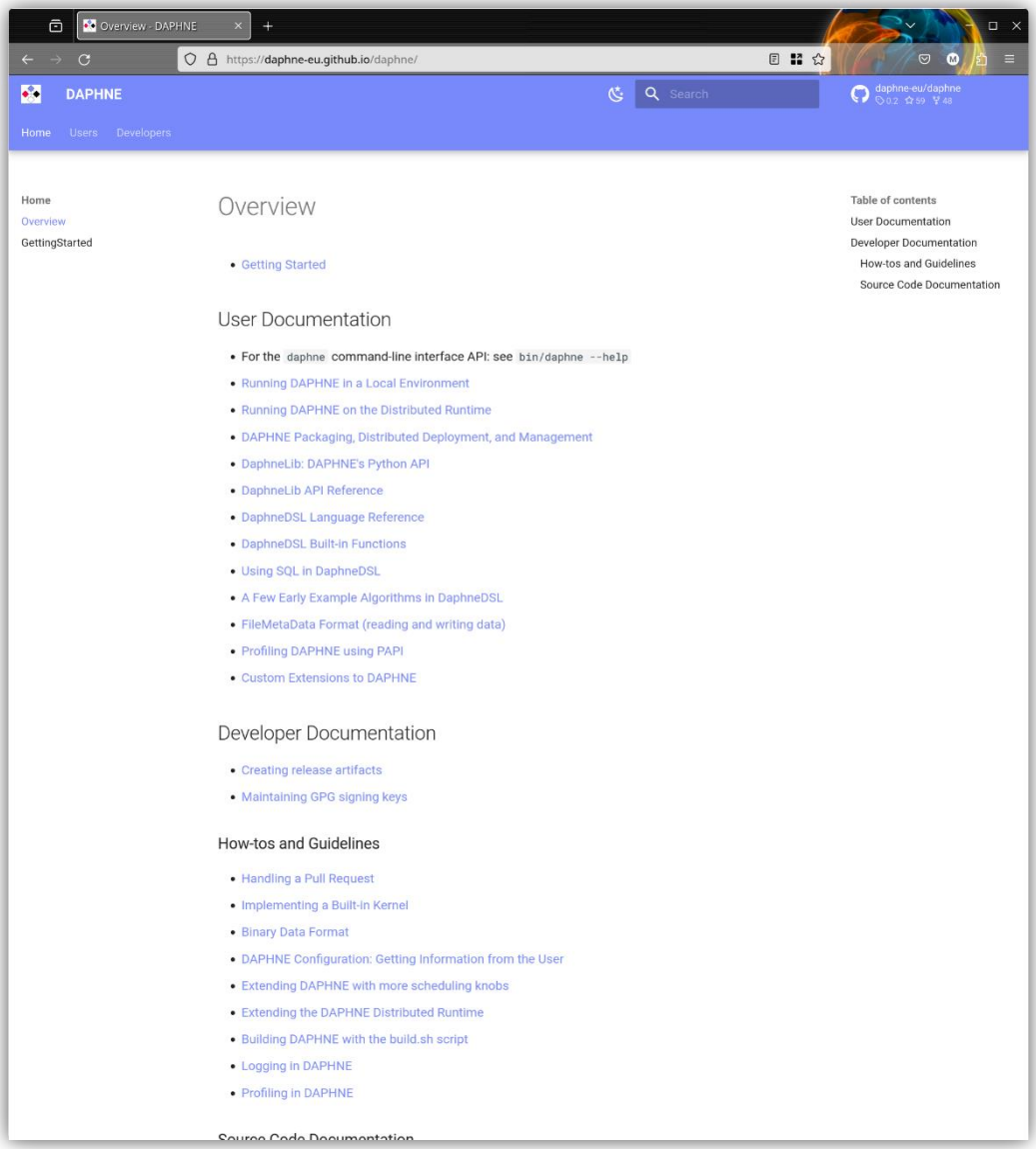

Figure 6 DAPHNE Online Documentation

5 Release History

As we started the open sourcing of DAPHNE early on, we take the opportunity in this section to look back on previous releases that happened before this deliverable. In the past, we had three releases, namely the un-versioned initial move from a private Gitlab hosted at the coordinator KNOW to a public repository on GitHub [7] and versions 0.1 and 0.2. Below we give a description of these previous releases and the original release notes which can also be found at [7].

Initial source opening (no version)

In the initial move to open source on ***March 31, 2022***, the source code of the DAPHNE prototype was migrated from a private GitLab repository hosted by the coordinator KNOW to the public GitHub repository [7] that we use for development since this day. As this was not an official software artifact release, we did not include any release highlights or change log. The prototype back then was in a good state to enable first simple tests to get to know the source tree and to set up a development environment for first steps in DaphneDSL.

DAPHNE version 0.1

This first release happened on the ***October 21, 2022*** and contains *basic initial implementations* of most components of the DAPHNE system infrastructure, from a domain-specific language (**DaphneDSL**) as the main user frontend so far (a Python API is under development), over an MLIR-based intermediate representation (**DaphneIR**) and compilation chain to runtime components for the distributed execution using a stand-alone backend based on gRPC [9] and the local, optionally vectorized, execution on CPU, GPU, and FPGA, for a certain subset of operations and data/value types.

Release Notes:

- The focus of this release is on providing an early preview of the DAPHNE system prototype for interested researchers/developers/users.
- We are aware of several bugs and issues. Note also that several features of the system are still experimental.
- Likewise, there is still a lot of room for performance improvements in various components.
- The binary release contains precompiled DAPHNE libraries, executables, scripts, and documentation.
- Binaries were generated on Ubuntu 20.04 LTS on an Intel® Xeon® machine compiled for x86_64.
- GPU operations were compiled against CUDA 11.7.1.
- As this is the first release, the auto-generated full changelog contains all commits to date.
- Experimental features (FPGA, Arrow) which are not available in the binary release can be compiled from source. Follow the instructions in the documentation section.
- To avoid problems with library dependencies, two binary artifacts were provided for this release. One compiled with CUDA support and one without.

DAPHNE version 0.2

Among many bug fixes and improvements, this second release (we regard them as development snapshots) contains several noteworthy features that we highlighted in a separate list (see release notes below). This release was published on the ***July 31, 2023***.

Release Notes:

- DaphneDSL: various little additions and improvements
  - second-order function map()
  - elementwise conditional operator

- system internals
  - initial OpenMPI backend for distributed runtime
  - some more FPGA kernels
  - improvements to the vectorized engine
  - logging facility based on spdlog library
  - initial profiling features

- build and deployment:
  - containers for simple deployment
  - refactoring of the build scripts
  - continuous integration

- external dependencies
  - libeigen is used for calculating Eigen values/vectors and for integer matrix multiply
  - libpapi for profiling
  - OpenMPI for distributed operation
  - hwloc for detecting processor details
  - spdlog for logging

- miscellaneous changes

- based on a new snapshot of LLVM/MLIR
- CUDA version 12.1.1
- support for 64 bit ARM (CPU only)

6 Highlights of the upcoming release DAPHNE v0.3

The third DAPHNE release was targeted around the end of May to nicely coincide with the delivery of D10.3 but will be delayed to ***June 2024***, to favor stability over a rushed release that might contain unwanted bugs and cause more troubles in the long run. Therefore, we decided to give the process some more time. The highlights of version 0.3 are to be regarded as an outlook to the upcoming release, but the ones listed here are already on the main branch:

- DaphneDSL/DaphneLib:
  - Additional ready-to-use data science algorithms: decision trees, random forests, PageRank
  - Various DaphneDSL language improvements (e.g., UDFs with multiple return values, more built-in functions, bounds-checks for left/right indexing, ...)
  - Support for complex control flow in DaphneLib
  - DaphneLib as a Python package
  - Efficient data exchange with pandas, TensorFlow, PyTorch

- DAPHNE Compiler:
  - Initial MLIR-based codegen pipeline
  - Introduction of a kernel catalog to make the compiler aware of available pre-compiled kernels

- DAPHNE Runtime:
  - CUDA 12.2.2
  - CUDA-support for more kernels
  - Support for synchronous gRPC and chunked data transfer in distributed runtime

- Infrastructure and general:
  - Initial extensibility for custom kernels
  - More consistent and actionable error messages
  - Numerous little improvements and bug fixes

7 Artifact Access

The DAPHNE pre-v0.3 open source software is publicly accessible as a snapshot of the DAPHNE development repository under the following link:

- **Link:** https://tinyurl.com/daphne-D103

This snapshot is a copy of the DAPHNE open source repository (main branch, May 31, 2024), available at [7] under Apache License v2.0. Besides the source snapshot, the artifact contains scripts to run the example described in Section 9 and a copy of this document.

8 Getting Started & Documentation Overview

One important step towards a successful release and adoption is documentation, which is improved and polished with slightly more effort in front of a release. Besides inline documentation in the form of source code comments we write down user and developer documentation in text files in markdown format in the doc subdirectory of the DAPHNE source tree. From there we set up an automated process via GitHub actions to create a nicer looking and better browsable and searchable documentation website [9]. The central page for a user to start out in DAPHNE is the getting started page that holds information on the prerequisites to set up a DAPHNE environment, how to run a first hello world example and pointers to the other documentation to advise the user where to go from here.

9 Example

While we presented DAPHNE on multiple occasions already (in previous software releases and deliverables), the following example serves as a showcase for getting an impression of how DAPHNE is used and what sort of problems can be tackled with it. More examples get added to our online documentation regularly or are described in various publications like the one described below that tests two scheduling settings [11] or a distributed example that also shows the integration of DAPHNE into a web-based user interface [12].

The example below uses two different scheduling methods on an implementation of the pagerank algorithm that is run with a sparse graph of a Wikipedia data snapshot. The method "Integration of the `AUTO` option to the DAPHNE's scheduling algorithms portfolio" was added to the DAPHNE code base in PR #647, commit dc3e5ad.

Using DAPHNE's vectorized engine, that is described in more detail in deliverable D2.2 [13] (a method of multi-threading and cache conscious processing that can fuse operator pipelines and split work across multiple workers), the user can employ several scheduling mechanisms while executing DAPHNE scripts (more details on the available options in the help text of the daphne executable (displayed by issuing daphne --help on the command line) or in the documentation page on scheduling knobs [13]). The AUTO scheduling option in DAPHNE creates tasks of an empirically defined chunk size. The method was proposed in [11] and it calculates the tasks' chunk size using golden ratio phi=1.618, the number of units of work, and the number of processing elements to arrive at a chunk size value that leads to high performance - see Sec. 3.1, Eq. 1 in [11].

In the code listing below (and in the Readme file of the deliverable artifact) we show instructions how to prepare and run the example. In the figure below after the code listing, we show exemplary results collected from running on two different compute nodes (one older system and a more recent one).

```
# Unpack the deliverable artifact and cd into it.
unzip daphne-d10.3.zip
cd daphne-d10.3/_d10.3

# Pull a container image with pre-built DAPHNE executable.
docker pull daphneeu/daphne:2024-05-29_X86-64_BASE_ubuntu20.04

# export number of vcores
export VCORES=$(nproc)

# Download the matrix
Wget https://suitesparse-collection-
website.herokuapp.com/MM/Gleich/wikipedia-20051105.tar.gz
tar -xzf wikipedia-20051105.tar.gz

# Set up the metadata of the matrix
echo
'{"numRows":1634989,"numCols":1634989,"valueType":"f64","numNonZeros":
19753078}' > wikipedia-20051105/wikipedia-20051105.mtx.meta

# Download the DaphneDSL script
wget https://raw.githubusercontent.com/daphne-
eu/daphne/main/scripts/algorithms/pagerank.daph

# Execute DAPHNE with the default configuration (STATIC scheduling
strategy with a CENTRALIZED queue) of DaphneSched on 10 threads
```

```
./launch-daphne-container --vec --select-matrix-repr --pin-workers --timing --num-threads=$VCORES pagerank.daph G=\"wikipedia-20051105/wikipedia-20051105.mtx\" alpha=0.8 maxiter=250

# Execute DAPHNE with AUTO (with a CENTRALIZED queue) of DaphneSched on 10 threads
./launch-daphne-container --vec --select-matrix-repr --pin-workers --timing --num-threads=$VCORES --partitioning=AUTO pagerank.daph G=\"wikipedia-20051105/wikipedia-20051105.mtx\"
```

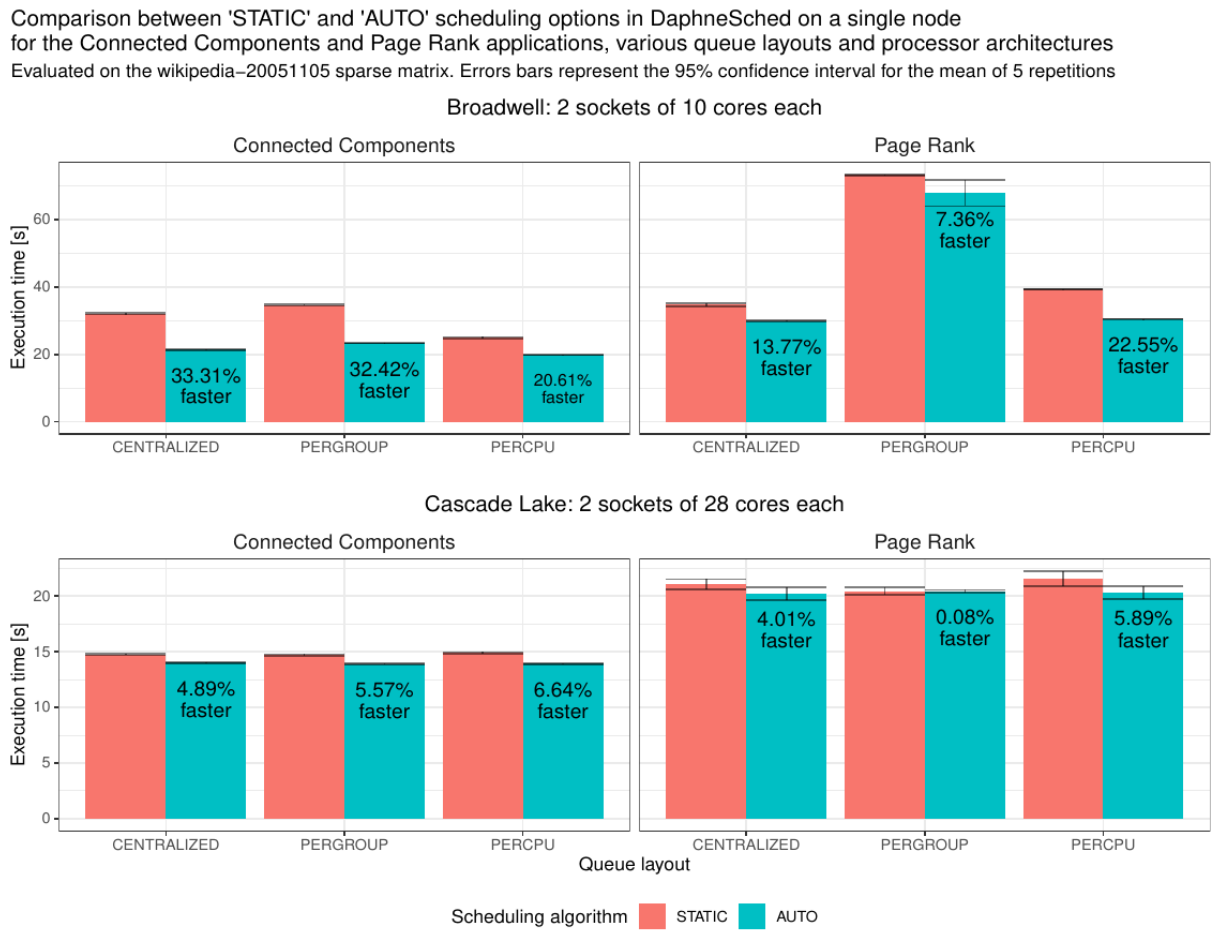

Figure 7 Exemplary results using two different hardware setups and settings for scheduling. The AUTO scheduling option in DAPHNE creates tasks of an empirically defined chunk size. STATIC uses a fixed chunk size.

10 Source Code Repository Structure

This directory structure listing shows most of the important directories of the project:

- **.github/** (files related to GitHub actions CI that runs test builds and doc generation)
- **bin/** (compiled system and parser; generated via build.sh (not available directly after check out))
- **build/** (generated/temporary compiler output of build.sh)
- **containers/** (build and run files for Docker containers)
- **daphne-opt/** (MLIR based code generation pipeline)
- **deploy/** (script collection to deploy DAPHNE on SLURM based super/cluster computer installations)
- **doc/** (basic setup, user and developer documentation)
- **lib/** (generated kernel libraries)
- **scripts/** (**DaphneDSL** scripts as examples)
    - **algorithms/** (algorithms such as connected components, kmeans, pagerank, decision trees, random forests etc implemented in DaphneDSL, which can serve as building blocks for more complex integrated data analsis pipelines)
    - **examples/** (simple examples that also make use of the above mentioned building blocks)
- **src/** (main source code repository)
    - **api/** (cli including daphne which orchestrates the remaining components)
        - **cli/** (user program and config)
        - **daphnelib/** (python lib bindings)
        - **internal/** (user program)
        - **python/** (python lib)
    - **compiler/** (execution, explain, inference, lowering)
        - **catalog/** (extensibility catalogue)
        - **codegen/** (code generation)
        - **execution/** (executing compiled DAPHNE programs)
        - **explanation/** (explain output generation)
        - **inference/** (type and property inference)
        - **lowering/** (compiler passes)
        - **utils/**
    - **ir/** (DaphneIR including the DAPHNE MLIR dialect)
    - **parser/** (DaphneDSL, SQL)
        - **catalog/** (extensibility mechanism)
        - **config/** (handling user configuration)
        - **daphnedsl/** (DaphneDSL parser)
        - **metadata/** (meta data for data reader & writer)

        - **sql/** (SQL Parser)
    - **runtime/** (distributed, local including data, I/O, kernels, and vectorization)
        - **local/**
            - **context/** (passing and saving state in a context object)
            - **datagen/** (data generator utility)
            - **datastructures/** (tensors, matrices, memory management)
            - **instrumentation/** (timing and profiling)
            - **io/** (data readers and writers)
            - **kernels/** (CPU kernels)
                - **CUDA/** (CUDA device kernels)
                - **FPGA/** (FPGA device kernels)
            - **vectorized/** (launcher and scheduler code for vectorized execution)
    - **util/** (helper functions etc.)
- **test/** (test suite of component and integration tests, organized by components)
- **thirdparty/** (dependencies such as llvm, including their build directories)
- **build.sh** (build script to build the DAPHNE compiler)
- **test.sh** (Daphne test suite)

11 Outlook

After the release of DAPHNE, we continue its development. Some partners in the DAPHNE consortium intend to develop DAPHNE further even after the project has ended. This was one of the design goals of the project, to establish an extensible system infrastructure that fosters research by providing an environment for experimentation of components that would otherwise not stand on their own. This does not only mean that we maintain DAPHNE for another year as written in the grant agreement. Regardless of funding of a direct follow-up project, current plans reach three to four years into the future and will dig deeper into the topics of compression, code generation, scheduling, and memory management to only name a few. Other topics might be included into DAPHNE as trends in DM, HPC, and ML change and third-party researchers and developers pick up the project.

References

[1]	Daphne Deliverable D10.1: Refined Dissemination and Exploitation Plan, 05/2022
[2]	Daphne Deliverable D10.2: Report on Community Building, 11/2023
[3]	Linkedin – https://www.linkedin.com/in/daphne-eu-project-695735230/
[4]	Twitter/X – https://twitter.com/daphne_eu
[5]	DockerHub – https://hub.docker.com/u/daphneeu
[6]	PyPI – https://pypi.org/user/daphne-eu/
[7]	GitHub – https://github.com/daphne-eu/daphne/
[8]	Intel PAC D5005 – https://www.intel.com/content/www/us/en/docs/programmable/683568/current/introduction.html
[9]	DAPHNE Online Documentation – https://daphne-eu.github.io/daphne/
[10]	gRPC – https://grpc.io/
[11]	Mohammed, A., Müller Korndörfer, J. H., Eleliemy, A., & Ciorba M, F. (2022). *Automated scheduling algorithm selection and chunk parameter calculation in openmp.* IEEE Transactions on Parallel and Distributed Systems (TPDS), 33(12), 4383-4394.
[12]	Aristotelis Vontzalidis, Stratos Psomadakis, Constantinos Bitsakos, Mark Dokter, Kevin Innerebner, Patrick Damme, Matthias Boehm, Florina Ciorba, Ahmed Eleliemy, Vasileios Karakostas, Aleš Zamuda, and Dimitrios Tsoumakos *DAPHNE Runtime: Harnessing Parallelism for Integrated Data Analysis Pipelines* 29th International European Conference on Parallel and Distributed Computing 28 August – 1 September 2023 Limassol, Cyprus
[13]	DAPHNE scheduling knobs: https://daphne-eu.github.io/daphne/development/ExtendingSchedulingKnobs/
[14]	Daphne Deliverable D2.2: Refined System Architecture, 08/2022