D5.2. FishData infrastructure

2022-07-12

SMARTFISH H2020

Innovation for sustainable fisheries

SMARTFISH H2020
Innovation for sustainable fisheries

Project number:	773521
Project duration:	1 Jan 2018 – 31 Dec 2021
Project coordinator:	Bent Herrmann, SINTEF Ocean
Web site:	www.smartfishh2020.eu

Deliverable ID:	Due month:	Preparation date:
D5.2	M24	2022-07-12

Title:

FishData infrastructure

Open access revision

Lead beneficiary:
SINTEF Ocean

Prepared by:
Lars T. Kyllingstad, SINTEF Ocean
Karl-Johan Reite, SINTEF Ocean

Approved by:
Rachel Tiller, SINTEF Ocean

Abstract

The goal of SMARTFISH H2020 is to develop, test and promote a suite of high-tech systems for the EU fishing sector, thereby optimising resource efficiency, improving automatic data collection for fish stock assessment, provide evidence of compliance with fishery regulations and reduce the ecological impact of the industry.

This document marks the completion of the prototype implementation of FishData—an experimental data acquisition and analysis infrastructure established by and for the SMARTFISH H2020 project. Here, we present the current status of the implementation, provide some implementation details that were not included in the system specification (deliverable D5.1), and show some examples that illustrate its use and discuss future developments.

This is an open-access revision of a confidential deliverable. Some of the text in the original report has been removed.

Deliverable type

R	Document, report	
DEM	Demonstrator, pilot, prototype	X
DEC	Web sites, patent fillings, videos, etc.	
OTHER	Software, technical diagram, etc.	

Authorship information

Editor	Lars T. Kyllingstad, SINTEF Ocean
Contributing partners	

Version history

Version number	Date	Description of changes
1	2019-12-31	Initial version. Delivered to the European Commission as D5.2.
2	2021-05-19	Updated according to feedback received after Reporting Period 2 (M36) review: Added a new section 2.2, *Links to other projects*, which clarifies the relationships between SMARTFISH H2020 and other projects, especially *DataBio*. Removed the chapter *Fulfilment of tasks* (was chapter 3). It consisted in its entirety of a table which enumerated sub-tasks of Tasks 5.2 and 5.3, with references to where these have been addressed in this deliverable and D5.1. As pointed out by the reviewers, the table was hard to follow, and on revision, we found that it did not provide new or crucial information. Removing it also obviated most of the need to refer back to D5.1, addressing yet another of the reviewers' concerns. Added a new section 3.4.3, *Onboard hardware*, consisting of what little information from the old, now removed, chapter 3 which was not mentioned elsewhere. Shortened the introduction to chapter 5, *Examples of use* (previously chapter 6) slightly, since the link between the projects has now been described in section 2.2. Added an entirely new chapter 6, *Conclusions and outlook*.
2-oa	2022-07-12	Removed sensitive information to enable open-access publication. The removed text described plans by commercial actors to industrialise the reported results. A few minor errors in the text have also been found and fixed.

SMARTFISH H2020 consortium

SMARTFISH H2020 (773521) is an Innovation Action within Horizon 2020, the European Union's framework programme for research and innovation, Call H2020-SFS-2017-1, Topic SFS-22-2017.

The following table contains information about the consortium members.

	SINTEF Ocean (SO) Postboks 4762 Torgard N-7465 Trondheim NORWAY www.sintef.no	Project Coordinator: Bent Herrmann bent.herrmann@sintef.no Project Manager: Rachel Tiller Rachel.tiller@sintef.no
	AZTI-Tecnalia (AZTI) 48395 Sukarrieta SPAIN www.azti.es	Guzmán Diez gdiez@azti.es +34 667 174 392
	DTU Aqua (DTU) Danmarks Tekniske Universitet Willemoesvej 2 9850 Hirtshals DENMARK www.dtu.dk	Ludvig Ahm Krag lak@aqua.dtu.dk +45 21 31 64 57
	Marine Scotland Marine Laboratory 375 Victoria Road Aberdeen, AB11 9DB UNITED KINGDOM http://www.gov.scot/Topics/marine/science/	Neil Campbell Neil.Campbell@gov.scot +44 (0)131 244 2902
	Melbu Systems AS Sminesgata 3, 8445 Melbu NORWAY http://www.melbusystems.no/	Einar Roger Pettersen einar@melbusystems.no +47 900 21 883
	Cukurova University Fisheries Faculty 01330, Saricam, Adana TURKEY http://www.cu.edu.tr/Eng/Default.aspx	Gökhan Gökce gokhan.gokce@ymail.com +90 322 330 6084 ext: 2961
	CEFAS Pakefield Road Lowestoft Suffolk NR33 0HT UNITED KINGDOM https://www.cefas.co.uk/	Thomas Catchpole thomas.catchpole@cefas.co.uk +44 (0) 7768041107
	Mersin University Fisheries Faculty Yenisehir Campus 33169 Mersin TURKEY http://www.mersin.edu.tr/	Huseyin Ozbilgin ozbilginh@gmail.com +90 3243412818

SINTEF	**Stiftelsen SINTEF** N-7465 Trondheim NORWAY www.sintef.com	Jens T. Thielmann Jens.T.Thielemann@sintef.no +47 930 59 299 Karl Henrik Haugholt Karl.H.Haugholt@sintef.no Petter Risholm Petter.Risholm@sintef.no +47 95 89 53 29
NTNU Norwegian University of Science and Technology	**NTNU** Institutt for Marin Teknikk, NTNU N-7491 Trondheim NORWAY https://www.ntnu.no/imt	Martin Ludvigsen martin.ludvigsen@ntnu.no +47 91897272
UEA University of East Anglia	**University of East Anglia** Norwich Research Park Norwich Norfolk, NR4 7TJ UNITED KINGDOM https://www.uea.ac.uk/	Michal Mackiewicz M.Mackiewicz@uea.ac.uk +44 (0)1603 59 1206
MARPORT	**Marport France SAS** 2 allée Copernic Espace Eurêka Parc Technologique de Soye 56270 Ploemeur, France	Didier Caute dcaute@marport.com +33 622 012 347
ZUNIBAL	**Zunibal** Idorsolo, 1 48160 Derio Bizkaia SPAIN http://zunibal.com/en/	Iratxe Arraibi iratxe.arraibi@zunibal.com (+34) 944 977 010 (7:30-15:30) (+34) 608 556 661
SafetyNet TECHNOLOGIES	SafetyNet Technologies LTD 29 Shand Street London, SE12ES UNITED KINGDOM http://sntech.co.uk/	Dan Watson dan@sntech.co.uk +44 779 294 9727
NERGÅRD	Nergård Havfiske Langnesveien 18 9408 HARSTAD Norway http://nergardhavfiske.no/	Tor Pedersen tor@yras.no +47 994 99 899
IFL	**Interfish** Wallsend Ind Est Cattedown Plymouth, Devon PL4 0RW, England	Andrew Pillar andrew@interfish.co.uk +44 (0) 1752 267261
LARRASMENDI, BI. S.L. BUQUES DE PESCA	Larrasmendi	Aitor Larranaga erlaxi@larrasmendibi.e.telefonica.net +34 946830617

Table of contents

List of figures

List of tables

Executive summary

We have now completed Task 5.2, *Development of onboard infrastructure*, and Task 5.3, *Development of onshore infrastructure*, of the SMARTFISH H2020 project's Work Package 5 (WP5). These tasks have dealt with the development of *FishData*, an ICT infrastructure that enables systematic collection and analysis of data from fishing vessels. The goal of FishData is to facilitate fisheries monitoring, fisheries management, stock assessment, other marine research activities and decision support for the fisheries industry.

The high-level architecture consists of an onboard infrastructure and an onshore infrastructure, each comprised of both hardware and software components, communicating through secure channels.

The *onboard infrastructure* is responsible for data acquisition and exchange on fishing vessels. The software is designed in a modular fashion, with loosely connected components that exchange information over a common communication channel. This is based on SINTEF's existing onboard data acquisition system, Ratatosk, which is again based on Data Distribution Service (DDS), an open middleware standard. Hardware components such as onboard PCs, communication units and so on are off-the-shelf units selected according to performance/capacity requirements, data sources, vessel type and type of fishery.

The *onshore infrastructure* is responsible for processing and storing data from multiple sources, including received data from fishing vessels. Data from vessels are received over a secure communication channel and stored behind corporate firewalls. The storage is protected by regular off-site backups, in addition to local safeguards such as replication between disks and between servers. The onshore infrastructure is based on some core technologies, such as GlusterFS and ZFS for storage and DC/OS for deployment of services and periodic/dependent tasks. Public key cryptography is the main means for access control to internal services, while OAuth2 is the primary method for access control to external services.

The work in WP5 has been carried out in close collaboration with another Horizon 2020 project, *DataBio*. Through this project, several applications aimed at the fishing industry, running on the FishData infrastructure, have already been demonstrated.

The remaining work in WP5 now consists of demonstrating the use of the infrastructure through a set of demonstrator applications, each of which shows the technology's usefulness for a particular use case aimed at fulfilling some of the SMARTFISH H2020 project's main objectives. These will be developed based on input from the project's target stakeholders: the fisheries industry, researchers who perform fish stock assessment, and fisheries management.

1. SMARTFISH H2020 motivation and background

With an increasing pressure on marine resource extraction mounting with resultant calls for sustainability in the sector, SMARTFISH H2020 will develop, test and promote a suite of high-tech systems that will optimise resource efficiency, improve automatic data collection, provide evidence of compliance with fishery regulations and reduce the ecological impact of the sector on the marine environment (Figure 1).

SMARTFISH H2020 will exploit and further develop existing technological innovations in machine vision, camera technology, data processing, machine learning, artificial intelligence, big data analysis, smartphones/tablets, LED technology, acoustics and ROV technology. The developments will assist commercial fishers throughout Europe in making informed decisions during pre-catch, catch, and post-catch phases of the harvesting process.

SMARTFISH H2020 will also provide new data for stock assessment from commercial fishing and improve the quality and quantity of data that comes from traditional assessment surveys. This provides the potential for more accurate assessment of fish stocks and allows for the assessment of stocks that are currently data-poor and therefore difficult to manage. In addition, the project will access automatically collected catch data from the fisheries which will also allow for management regulations to gain higher compliance rates.

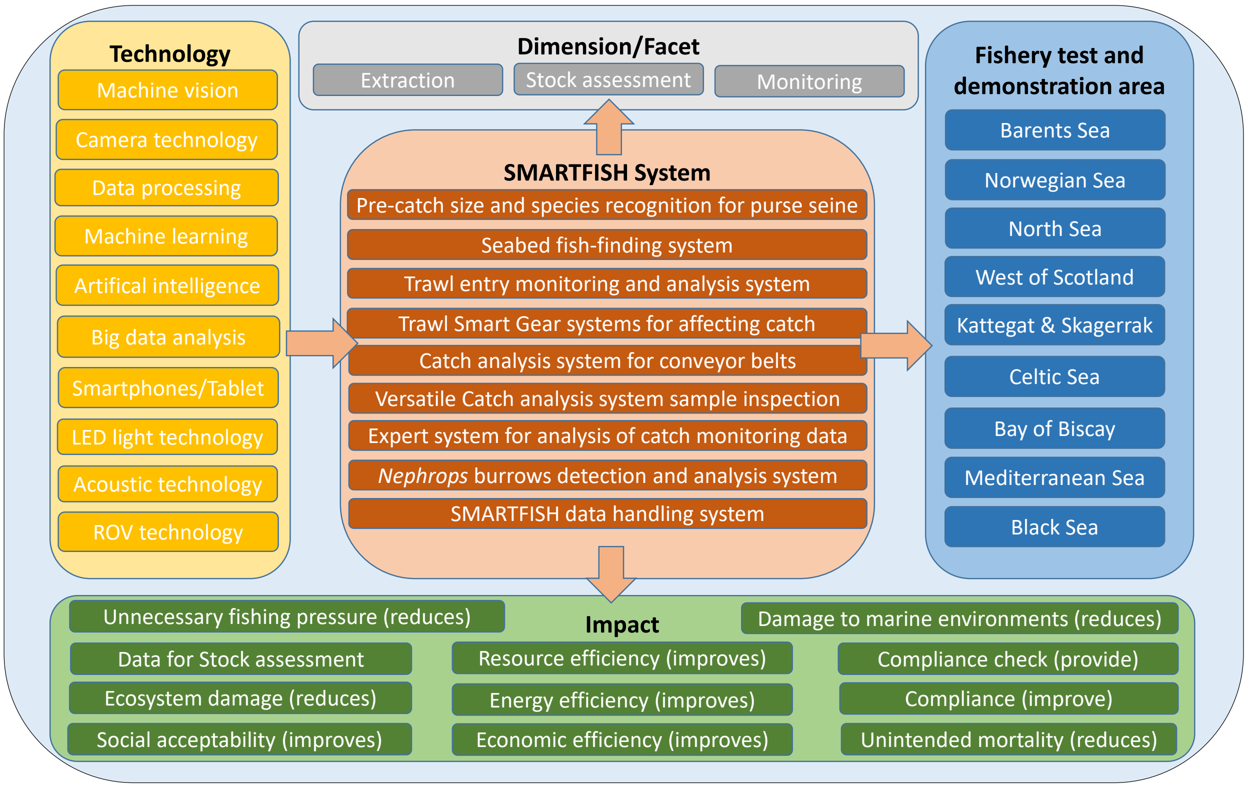

Technology
Machine vision
Camera technology
Data processing
Machine learning
Artifical intelligence
Big data analysis
Smartphones/Tablet
LED light technology
Acoustic technology
ROV technology
Dimension/Facet
Extraction
Stock assessment
Monitoring
SMARTFISH System
Pre-catch size and species recognition for purse seine
Seabed fish-finding system
Trawl entry monitoring and analysis system
Trawl Smart Gear systems for affecting catch
Catch analysis system for conveyor belts
Versatile Catch analysis system sample inspection
Expert system for analysis of catch monitoring data
Nephrops burrows detection and analysis system
SMARTFISH data handling system
Fishery test and demonstration area
Barents Sea
Norwegian Sea
North Sea
West of Scotland
Kattegat & Skagerrak
Celtic Sea
Bay of Biscay
Mediterranean Sea
Black Sea
Impact
Unnecessary fishing pressure (reduces)
Damage to marine environments (reduces)
Data for Stock assessment
Resource efficiency (improves)
Compliance check (provide)
Ecosystem damage (reduces)
Energy efficiency (improves)
Compliance (improve)
Social acceptability (improves)
Economic efficiency (improves)
Unintended mortality (reduces)

Figure 1: Conceptual structure of SMARTFISH H2020

1.1. Role of this deliverable

This document marks the completion of Tasks 5.2 and 5.3 of SMARTFISH H2020 Work Package 5 (WP5), defined by the Grant Agreement as follows.

Task 5.2, *Development of onboard infrastructure*:

> "This task is about developing an onboard infrastructure which consists of both hardware and software components. The software will be designed in a modular fashion, with loosely connected components that exchange information over a common communication channel. This will be based on SINTEF's existing onboard data acquisition system, Ratatosk, which is again

based on Data Distribution Service (DDS), an open middleware standard. Software components for interfacing with all relevant data sources will be developed, as well as components for secure storage in open, standardised formats and for data transfer to and from shore. Hardware components such as onboard PCs, communication units and so on will be off-the-shelf units selected according to performance/capacity requirements, data sources, vessel type and type of fishery. (For example, a vessel which is at sea for several weeks at a time is likely to have different communication requirements than one which only makes day trips.). An interface to SCALA (a realtime onboard commandant interface from Marport) will enable sharing of information between Marport sensors and the onboard infrastructure. This will also provide an interface for interaction with the crew (camera, LED, codeend closure system)."

Task 5.3, *Development of onshore infrastructure*:

"The data received from the vessels must be handled safely and securely on shore, and there needs to be an efficient infrastructure in place that is able to handle a large variety of data from a multitude of sources and process it for different purposes. Software will be developed for secure, robust storage and management of large amounts of data as well as distributed high-performance processing and analysis. This will be based on the systems already in place at SINTEF Marine Data Centre, which is again built on state-of-the-art open source "big data" software (mainly the Apache Hadoop stack). A unified access mechanism for both raw and processed data of all types will be developed. This will include security barriers to ensure that sensitive data is only accessed by those with explicit permission."

2. Introduction

FishData is an experimental data acquisition and analysis infrastructure established by and for the SMARTFISH H2020 project. This document marks the completion of the implementation of the initial FishData prototype.

In a previous report from this project—SMARTFISH H2020 deliverable D5.1, *FishData system specification* [1]—we have already provided an extensive description of the technology, in particular:

- the purpose of and motivation for developing FishData
- the scope of the work
- the state of the art of "big data" in fisheries
- our use cases
- the architecture and underlying technologies

We therefore refer the reader to D5.1 for this information and will not repeat it in the present document. (Note: Unless otherwise specified, all mentions of D5.1 in the following chapters refer to version 2 of the specification.)

The main purpose of this document is to present the status of the implementation and to supply any new technical information that is deemed relevant. In chapters 3 and 4, we provide these technical details that, for various reasons, were not included in D5.1. In chapter 5, we show some examples of use, illustrated by work done in different projects.

2.1. Remaining work

The work that remains in WP5 now consists of two main tasks, both of which concern the development of stakeholder-oriented applications running on the FishData infrastructure:

- Task 5.4, *Development of analyses and presentation for fisheries decision support*, in which we will demonstrate the use of big data techniques and technologies in a case study focused on the needs of stakeholders in the fishing industry.
- Task 5.5, *Development of analyses and presentation for stock assessment and fisheries management*, in which we will do the same for a case study that focuses on stakeholders in marine research and resource management.

These will result in the final deliverable of WP5: D5.3, *FishData analysis*, due in December 2020.

2.2. Links to other projects

The work in WP5 has been carried out in close collaboration with the *DataBio*[1] project—in particular, their work package 3, *Fishery Pilot*. The aim of DataBio WP3 was to explore the use of *big data* methods and technology to build decision support systems for efficient and sustainable fisheries. Thus, there is some thematic overlap between the projects. However, SMARTFISH WP5 differs in its greater focus on building robust infrastructure, as well as its broader scope which includes fisheries management and stock assessment in addition to fisheries decision support.

DataBio started in 2017, one year before SMARTFISH. Its early activities included surveying the state of the art of *big data* in fisheries, mapping out ideas for applications together with industry stakeholders, exploring different technologies for data management and processing, and making some initial technology choices. However, DataBio kept a focus on the applications (called *pilots* in that project) and the specific tools needed to realise them.

[1] Data-driven Bioeconomy (DataBio), funded by the European Union's Horizon 2020 research and innovation programme under grant agreement no. 732064, 2017–2019. https://www.databio.eu/

In SMARTFISH, on the other hand, we set out with an aim to build a robust, flexible and future-oriented infrastructure first, and then to demonstrate its potential applications through demonstrators afterwards. As such, SMARTFISH immediately benefited from the early explorations carried out in DataBio, and through the course of the project, we have been able to use the DataBio pilots as "use cases" to guide the development of FishData. DataBio, on the other hand, benefited from a continuously expanding and improving infrastructure on which to build the pilots. This has been a very fruitful collaboration, and we have included some examples of the results in chapter 5. The DataBio fishery pilots are described in detail in the project's own final report. [2]

DataBio ended in 2019, about the same time as we completed the SMARTFISH infrastructure work described in the present report and started turning our attention towards demonstrators of our own.

Another project which is worth mentioning here is *FishGuider*[2]. This is a partly-industry-funded innovation project whose aim is to develop a decision support system for fishers by combining fishery data with novel fish migration models. This project has also been using the prototype FishData infrastructure to run its data scrapers and collect simulation results. An example of the results is shown in chapter 5.

2.3. Diagram notation

In this document we use the Fundamental Modeling Concepts (FMC) notation extensively, both to model system structures as block diagrams and to model processes as petri nets. Readers who are unfamiliar with this notation are encouraged to read the "FMC Quick Introduction". [3]

[2] Decision support for fishing vessels based on marine ecosystem models and fishery data (FishGuider), funded by the Research Council of Norway under grant agreement no. 296321, 2019–2022. https://www.sintef.no/en/projects/2019/decision-support-for-fishing-vessels-based-on-marine-ecosystem-models-and-fishery-data/

3. Implementation details

This chapter contains some details that were not included in D5.1. Partly, this is because some of the information was not available at the time that deliverable was submitted, and partly it is because we consider some of it to be implementation details that are not relevant for inclusion in what is mostly an architectural specification.

That said, we do consider the FishData system specification (of which D5.1 was merely the first iteration) to be a living document, and we intend to keep it up to date and available to the public as the work progresses in SMARTFISH H2020 and other projects. This entails incorporating any relevant information below in the specification.

3.1. Terminology

In the following sections we use some terms which are somewhat generic, but to which we assign a specific meaning in this context. To avoid confusion, we have provided a list of definitions in Table 1.

Table 1: Definitions of terms used in this document.

Term	Definition
Service	A long-lived software process or group of processes that provides some functionality to users or other services. A service may be associated with specific hardware.
Job	A short-lived software process that performs a specific task and terminates once the task is complete.
Application	A service or group of services designed for end users.
Pipeline	A series of operations performed on data, where the output of each operation is used as input for one or more other operations, so that the entire chain forms a directed acyclic graph (DAG). The operations may be carried out by services, as jobs, or a combination thereof.

3.2. Services and components

An overview of the most important components of the onshore part of the FishData infrastructure is given in Table 2. In subsequent sections, we will show how these fit together to form various services and pipelines. For simplicity, the components will often be mentioned with reference to their ID instead of their name.

Table 2: The most important components of the onshore FishData infrastructure.

Name	ID	Description
Authentication service	*smd-oauth*	A web service whose purpose is to decide if a web request is to be allowed, denied or sent to login. It is based on the Flask Python library. [4]

Name	ID	Description
Docker registry	*docker-registry*	A private Docker registry internal to SMD.
Data store	*file-store*	A shared file storage which supports replication and high availability. It is based on GlusterFS. [5]
Data store accessor	*file-access*	A service mediating access to the shared storage file-store. Access rights are controlled using public key cryptography, which again is maintained using a combination of *salt-main* and *keyweb*.
Onshore configuration service	*salt-main*	A service used for managing the configuration of onshore servers, based on Salt. [6]
Vessel configuration service	*salt-vessels*	A service used for managing the configuration of ship servers, based on Salt. [6]
Key management service	*keyweb*	A web service distributing the public keys and the access they are to be granted. This is used by both onshore and ship servers.
DC/OS bootstrap	*dcos-bootstrap*	A service for installing and updating DC/OS master and agent nodes. Runs in a Docker container, based on the installation provided by Mesosphere.
Marathon	*marathon*	Deploys services to DC/OS and monitors their health. This is based on Marathon. [7]
Chronos	*chronos*	Deploys periodic and dependent tasks to DC/OS. This is based on Chronos. [8]
Internet-facing proxy	*proxy-external*	The load balancer fronting internet-facing services which are not based on DC/OS. Based on HAProxy. [9]
Internet-facing DC/OS proxy	*proxy-dcos-external*	The load balancer fronting internet-facing services which are based on DC/OS. Based on Marathon-lb. [10]
Internal DC/OS proxy	*proxy-dcos-internal*	The load balancer fronting internal services which are based on DC/OS. Based on Marathon-lb.

Name	ID	Description
Vessel gateway	*io-vessels*	Both a gateway for accessing the ship servers from shore and a temporary file storage where the ship servers send their sensor data when they have internet communication.
Data fetcher	*datafetcher*	Moves ship sensor data from io-vessels to file-store. This is a job defined by a Docker image and run by Chronos.
APT package repository	*apt-repo*	Distributes own software built into Debian packages. Based on Aptly. [11]
Vessel status service	*statusweb*	Web service for assessing the status of ship deployments in terms of last contact and installed & running software versions.
GIS server	*gis*	Distribution of geospatial data, mainly the results of oceanographic simulations using SINMOD. Based on GeoServer. [12]
GIS database	*gis-db*	Database with geospatial data, such as catch reports. Used by gis. Based on PostgreSQL [13] with PostGIS. [14]

3.3. Security and network infrastructure

Broadly speaking, FishData security and access control are dealt with on two separate levels: The lowest is the *network infrastructure level*, where the question of which computers are allowed to connect to which is handled by specialised hardware that divides the network into separate zones protected by firewalls.

On top of that is the *service level*, where access to specific resources such as applications and data is controlled by various authentication mechanisms on a per-service basis. There are three crucial security aspects to deal with here:

- Encryption of communications
- Authentication of users and services
- Authorisation of users and services to access specific resources

We have settled on two sets of protocols for handling these, listed in Table 3. As shown, these are complementary in that they are suitable for different types of services.

In the following subsections, we discuss both network infrastructure and service security in some more detail.

Table 3: Security mechanisms.

Service/access types	Encryption	Authentication	Authorisation
File system access and remote login	SSH	SSH	operating system
Web services	HTTPS	OAuth2	OAuth2

3.3.1. Network infrastructure

A schematic of the overall network structure of the onshore infrastructure is shown in Figure 2. It is divided into three network zones:

Demilitarised zone (DMZ)

The DMZ contains the servers that must be accessible from the outside (e.g. the Internet). In our case, this is just some load-balancing proxies that route connections from the outside to specific services running on the DC/OS cluster (*proxy-external* and *proxy-dcos-external*).

The primary advantage of this setup is that it limits the number of servers which are exposed to the outside, and hence reduces the attack surface.

It also enables dynamic allocation of server resources to various services. For example, if the load on one DC/OS agent is getting too high, additional instances of its services can be started on other agents, and the proxy can direct new requests to those agents. Furthermore, if one of the DC/OS agents goes down for some reason, the services that were running on it can simply be started again on other agents. All of this is invisible to outside users, because they only communicate with the proxy.

Internal network zone

The internal network zone primarily consists of the DC/OS cluster servers (master and agent nodes). Hence, this is where most of the services run.

Machines and services in this zone can be reached by the DMZ proxies, which is of course necessary for them to be able to perform their routing tasks. Connections can also be made in the opposite direction, mainly because the proxies are also configured and controlled by the DC/OS masters (they are *public agents*, in DC/OS terminology).

The boundary between the DMZ and this zone is protected by an *application firewall*, i.e., a firewall that is able to control network traffic on any OSI layer up to the application layer. In other words, in addition to restricting network traffic on the IP address and port level, it also monitors application-specific traffic (e.g. HTTP) for malignant activity.

High-security zone

Finally, the high-security zone houses the services for which it is especially important to limit access, primarily the storage cluster. These machines are only reachable by services which have a particular, narrow IP address range, corresponding to the internal network zone.

Vessels and other external entities

In the above, we've only described the network configuration of the *onshore* infrastructure. The onboard infrastructure typically consists of only one or two computers per vessel, and to the extent that there are network infrastructure security issues that need to be taken into account, they primarily have to do with how the FishData computers interface with the vessel's network. This must be handled on a case-by-case basis and is for the most part dictated by those who are in charge of the vessel's ICT systems.

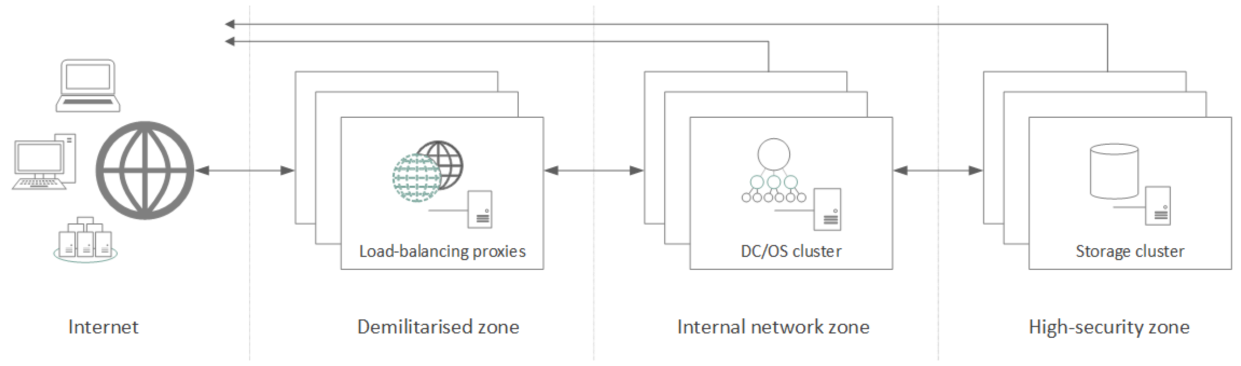


Figure 2: On-shore network infrastructure diagram showing the various network zones and which servers are allowed to communicate with which. The arrows indicate the direction in which connections are *established*. Once a connection is established, data may in most cases flow both ways. The dotted vertical lines mark zone boundaries.

The onboard infrastructure, along with other external entities, connects to the onshore infrastructure over the Internet, and the security in this link is managed using SSH, which we discuss next.

3.3.2. SSH

The *Secure Shell* (SSH) protocol is used for communication between several parts of the FishData infrastructure:

- File transfer between the data centre and external entities such as fishing vessels.
- Access by various internal services to the data storage cluster.
- Remote configuration and administration of both ship- and shore-based computers.

At its core, SSH is a cryptographic protocol for operating network services securely over an unsecured network. [15] We have chosen it due to its ease of configuration and use, and because of its ubiquitousness on UNIX-like operating systems.

SSH uses public-key cryptography to authenticate the remote computer and/or user. [16] This means that if a user or service—let's call it the *client*—wants to gain access to some other service—here referred to as the *server*—all that's required is that the server has a preexisting, verified copy of the client's public key.

The problem of access control for a given server then boils down to preloading it with the keys of all clients that are supposed to have access, and associating each client identity with a set of permissions to the server's functionality and data.

Key management

This key distribution is carried out via a special service which we call the *Key management service*. This is a simple web service that maps the names of server-specific permission sets to client keys. The system is illustrated in Figure 3.

Here is a concrete example to illustrate how this works: Let us say that the client is a service, which we'll call `analysis`. The server that `analysis` needs access to is another service; let's call that `datastore`. On this server we have several data sets, but `analysis` should only have access to one of them, which we will call `dataset_1`.

We generate a public-private keypair, `analysis.private` and `analysis.public`, and keep `analysis.private` stored safely on the client. `analysis.public` is placed by the system administrator in the key management service's key store.

Assuming that the key management service hostname is `keys.example.com`, then `datastore` will periodically download the file located at the following address (along with similar files for the rest of its data sets):

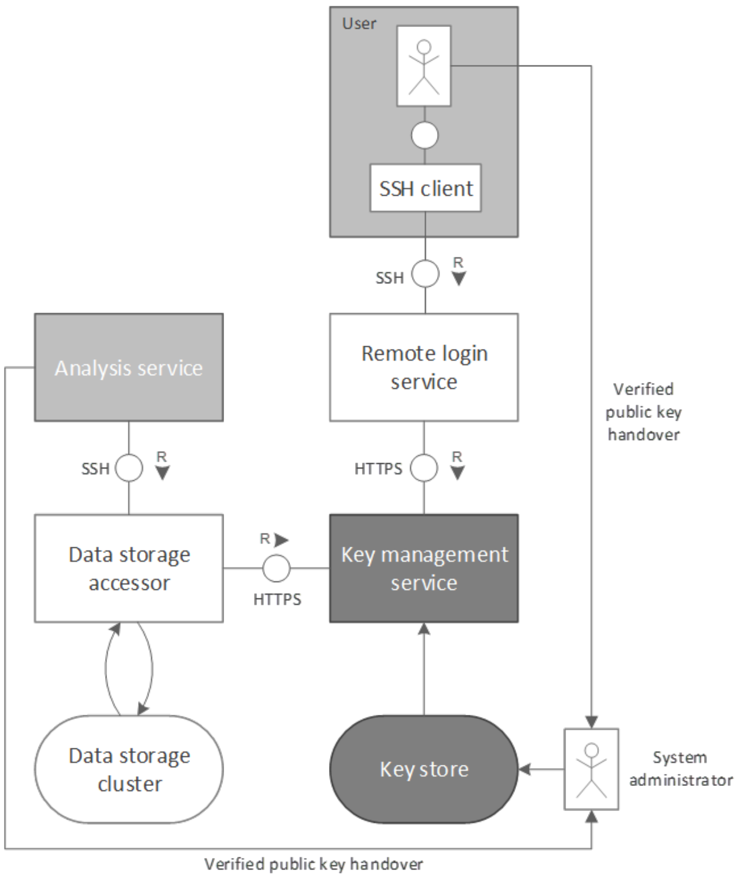


Figure 3: FMC diagram for the key management service, with two example clients (*User* and *Analysis service*) and two example servers (*Remote login service* and *Data storage accessor*).

```
https://keys.example.com/datastore/dataset_1.authorized_keys
```

This file contains the list of public keys that correspond to private keys which clients can use to access `dataset_1` on `datastore`, among them `analysis.public`.

In this example, `dataset_1` was a data set. However, it could of course be any resource on the server, such as a particular application, a user account, and so on. Genenerally speaking, then, it is a *permission set*: a name associated with a set of permissions to access server resources. The general format of the key list URL is thus:

```
https://<key_management_hostname>/<service>/<permission_set>.authorized_keys
```

The system does not contain special provisions for the initial public key transfer from clients to the key storage. It is up to the system administrator to handle this in a secure manner that ensures the authenticity of the keys, for example by a person-to-person handover.

Future development

The key management service is public to the world, meaning that we do not consider it a secret which clients have access to which servers. This allows us to keep the implementation simple. However, if this were to become an issue, it would not be much additional work to require that each server authorise itself before downloading the key lists.

Another nice feature of this system is that it would be quite easy to extend to other types of public keys if needed, as there is very little about it which is SSH specific. One technology which seems particularly promising, and which we will consider adopting in future versions of FishData, is *WireGuard*. This is a VPN protocol which is designed to be as easy to set up and use as SSH, but has an even wider range of applications. [17] Support for WireGuard will soon be built into the Linux OS kernel, making it especially convenient. [18]

3.3.3. Web services security

The FishData infrastructure needs to support web services based on data with varying access rules. At the same time, one doesn't want to have separate logins for different resources. To achieve single sign on for arbitrary resources accessed through web services, the architecture shown in Figure 4 was developed. In this figure, *Geoserver* and *Web server* are examples of services running on the FishData infrastructure. *Postgis database* is a database running on the FishData infrastructure. *Host filesystem* denotes the local file system on the host running the *Geoserver* container. The upper *Git storage* is a repository which stores the access rules and authentication web service implementation. The lower *Git storage is* a repository which stores the web service implementation.

Figure 5 shows the control flow of the authorisation procedure. When a user sends a request from the internet for a web service within the FishData infrastructure, it is filtered by the external firewall before it is sent to the external load balancer (*proxy-dcos-external*). This is a HAProxy instance. This instance is managed by DC/OS and Marathon *a*nd periodically updated based on the configurations of the services that it should expose.

Services which are to use the security mechanisms, define their general security settings in the service definition used by Marathon. These general settings define the internal web service to determine the access rights and the id of the service which is requested. The detailed rules which determine which users can access which

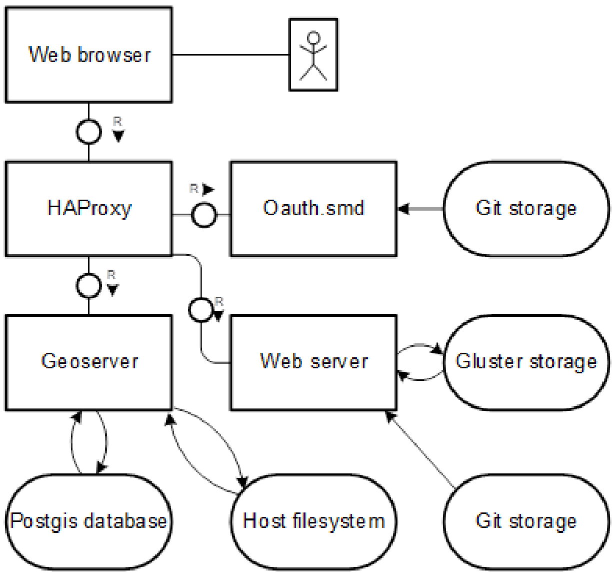


Figure 4: Web services security.

resource are separately described and used by the authentication web service (*smd-oauth*).

When the HAProxy instance receives a request for a protected service, it sends a request over a websocket to the authentication service. The authentication service responds with either "ok", "login required" or "forbidden", based on arbitrarily complex rules that take into account whether the user is logged in, which groups he belongs to and the requested resource address.

Based on the response from the authentication service, *proxy-dcos-external* either sends the request to the requested resource, returns a web page saying the user doesn't have access or forwards the request to the login page. The login page is served by *smd-oauth* and allows the user to login with his Oauth2 account.

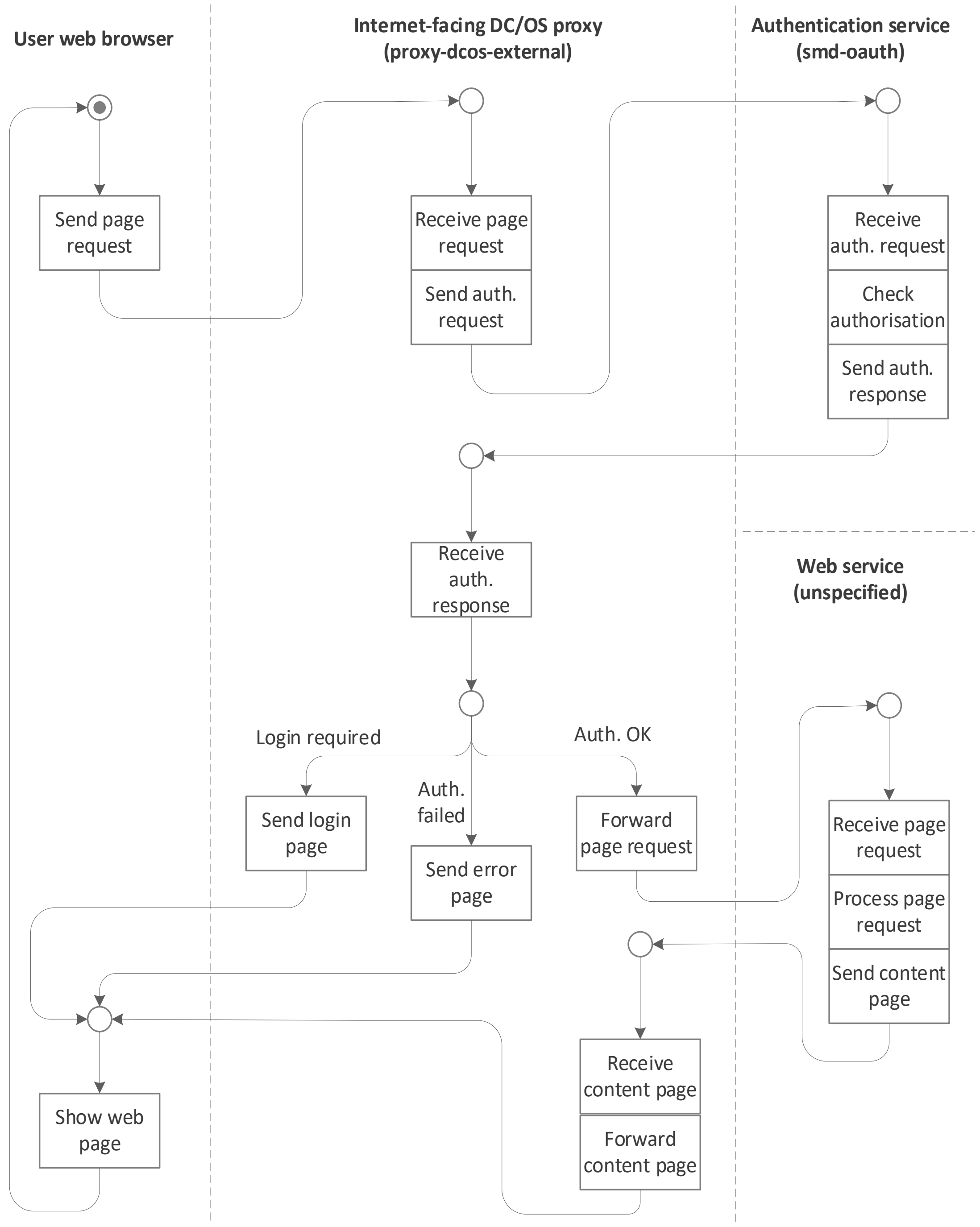


Figure 5: FMC petri net that shows the control flow of the authorisation procedure.

3.4. Configuration and administration

This section deals with installation, configuration and maintenance of the operating systems and low-level software that form the basis of the infrastructure. The main tool that we use for this is *Salt*, an open-source software for remote task execution and configuration management. [6]

Using Salt, one can describe the desired *state* of a system consisting of one or more computers, and the software will carry out the operations that are necessary to bring the system to that state. The state can include all sorts of configuration choices, like network interface settings, installed software, contents of configuration files, and so on. After the initial configuration, Salt can be run again at any time to ensure that the configuration remains as intended or to update the system with any changes to the intended state (for example a software upgrade or configuration change).

Salt is based on a master–slave architecture, where the configuration of several computers (the slaves) can be initiated and coordinated from a central computer (the master). This makes the system very scalable, simplifying the management of large numbers of machines.

We use Salt for configuration of both onboard and onshore machines, and we will discuss these in turn.

3.4.1. Onboard servers

The onboard computers run the Debian operating system, and the first step of the configuration is performed during OS installation using its *preseed* mechanism. This takes care of some low-level setup, such as disk partitioning, configuration of the APT [3] package repositories, and installation and configuration of the Salt slave software.

The remaining configuration steps are orchestrated by Salt (*salt-vessel*), as shown in Figure 6. This includes hardware-specific configuration (such as serial port setup) and further software installation (both third-party software and FishData software). FishData software is downloaded from a package repository hosted on the onshore infrastructure (*apt-repo*), based on the Aptly server software. Salt also sets up environment variables, network interfaces, startup jobs and periodic tasks.

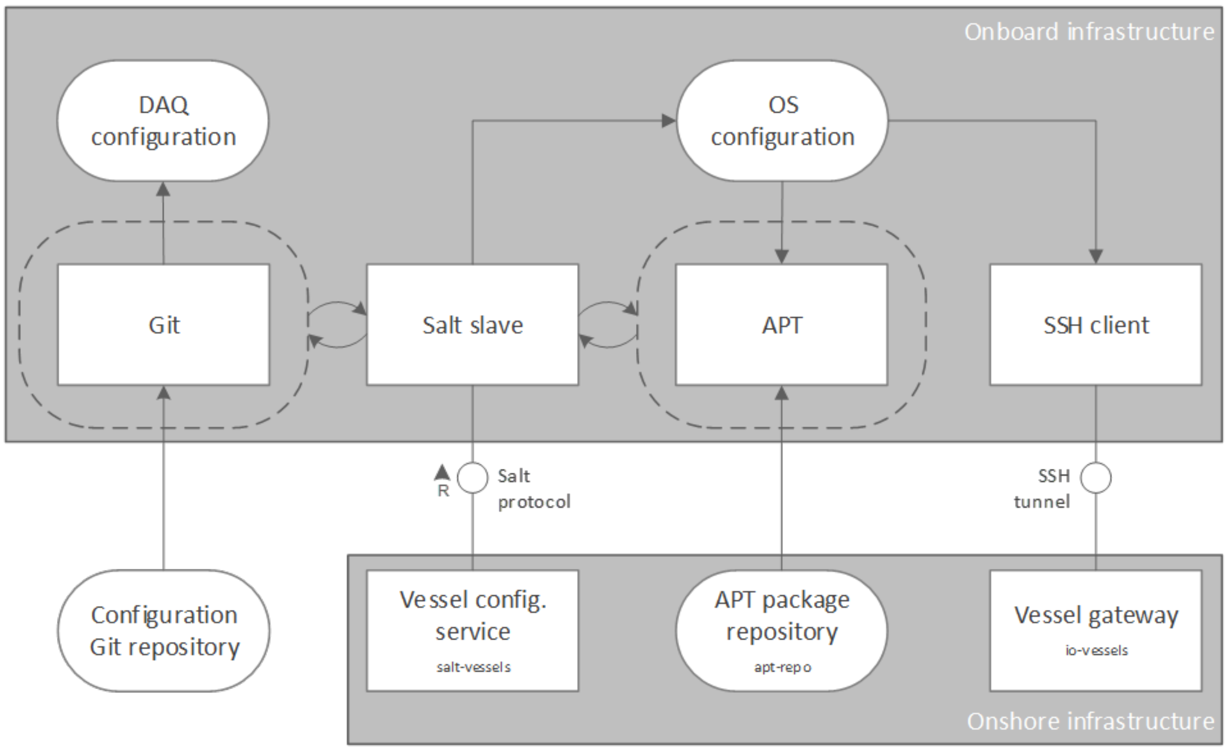


Figure 6: System for remote configuration and administration of the onboard infrastructure.

[3] APT (Advanced Package Tool) is Debian's software installation/management tool.

Some of the configuration is highly vessel specific, in particular the configuration files for the data acquisition (DAQ) modules. Such files are is kept under version control using the Git software, and the vessel systems download them directly using the Git client (usually triggered by *salt-vessel*). The onshore Git repository is currently hosted on a Git hosting service which we do not consider part of FishData.

To enable remote access to the onboard servers, a reverse SSH tunnel is automatically set up between each vessel and an onshore server (*io-vessel*) whenever the ship has Internet access. This is very useful in situations that require manual administration and/or troubleshooting. "Reverse" here means that the onboard system creates a persistent, passive connection to the onshore server, which can then be used to establish active connections from onshore users and systems back to the vessel. This makes connections from shore possible, while at the same time avoid opening the vessel firewall for connections from the internet.

3.4.2. Onshore servers

Onshore servers can in our case be divided into either physical servers or Docker containers.

The operating system on the physical servers are initially set up either manually or using the Debian preseed mechanism. This renders the server with a running Salt client service. From this point on, Salt is used for the remaining installation and configuration, specifically through the service *salt-main*. This typically includes setting up user accounts, network and services, such as DC/OS installation and configuration. *salt-main* also is used for upgrading e.g. the DC/OS services. This simplifies the process of replacing or adding servers, which is important both for saving time and for reducing the risk of errors.

Docker containers are initially set up using the ordinary Docker functionality. Usually, all static configuration and installation is set up this way, including services and cron jobs. Updates which needs to be performed during the container lifetime is usually handled by *salt-main*. This includes for example user management and system updates. A notable exception to this is updates of the authorized keys of each user account and updates to web services. The former is handled by the key management system described in section 3.3.2. Web services updates are handled by the docker container itself, as it periodically checks for new versions and updates its own services if updates are available.

3.4.3. Onboard hardware

The components that are most likely to be installed on a vessel for testing and demonstration are:

- iEi TANK-720: Rugged, fanless, industrial computer with 2 TB storage space (Figure 7).
- Robustel R3000-L4L: 4G cellular router

Both are off-the shelf units aimed at industrial applications. They provide ample storage space and means of communication for a vessel that lands its catch once a month or more frequently.

Figure 7: iEi TANK-720, the computer planned for use as the onboard server. (Photo: IEI Integration Corp.)

4. Onboard data acquisition pipeline

The flow of data from onboard data sources to onshore data storage was described at a high level in D5.1. Now, we can show how the various parts of the onboard and onshore infrastructures fit together in a *data acquisition pipeline* which ensures that data from fishing vessels are collected, structured and transported to shore in a secure and reliable manner. This pipeline is shown as an FMC block diagram in Figure 8. The steps are as follows:

1. Raw data from various sensors and instruments are converted from their source protocols to structured data published over DDS by protocol-specific bridge modules, as described in D5.1 section 6.1.2.
2. The logger module subscribes to these data using DDS and writes them to NetCDF files, as described in D5.1 section 6.1.3.
3. The shore connector module transfers the files to shore when possible. It does so by connecting to the *io-vessels* service using SSH and uploading the files with *rsync*, as described in D5.1 section 6.1.4.
4. The *datafetcher* service, which runs as a periodic *Chronos* job, moves all vessel data from *io-vessels* to the main data store—*file-store*. This transfer is mediated by *file-access*, to which *datafetcher* connects using SSH. This mechanism was described in D5.1 section 6.2.5.

The bottom row in Figure 8 indicates some further pipelines:

- The analysis and presentation pipelines that are under development in Tasks 5.4 and 5.5 (and which will be described in the upcoming deliverable D5.3) will be able to access the collected data from the data store, also via *file-access*.
- The *statusweb* service connects directly to *io-vessels*. It does not receive collected data; instead, it receives information about the connectivity and software status of all the vessels: which vessels are connected right now (or alternatively, when they were last online), what OS and software versions are they running, and so on. This information is highly useful to the maintainers of the FishData infrastructure.

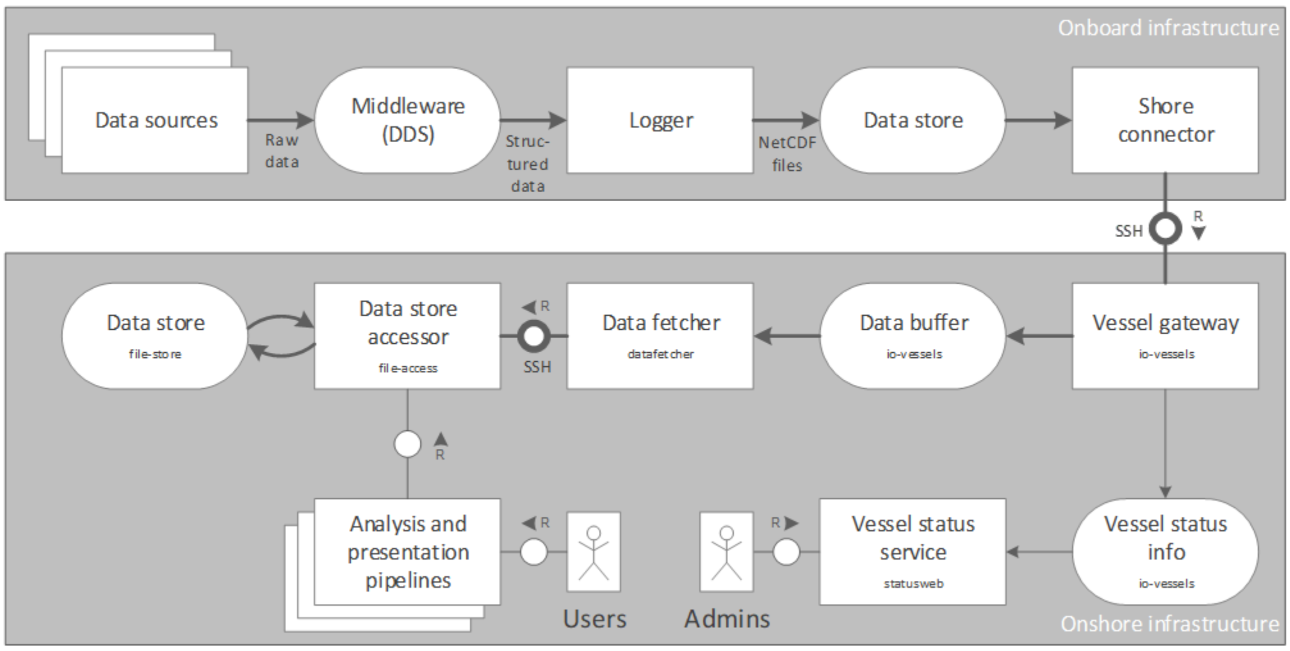


Figure 8: Data acquisition pipeline (emphasised) and how it connects to other pipelines.

5. Examples of use

As mentioned in the introduction (section 2.2), other projects that run in parallel with SMARTFISH have benefited from using the prototype FishData infrastructure while it has been under development. In particular, we mentioned the DataBio and FishGuider projects. In return, these projects have acted as "use cases" for FishData and have provided useful input during its development. In this chapter, we show some examples of results from these projects, and how they've made good use of FishData. These give an indication of the possibilities for development of new applications in SMARTFISH Tasks 5.4 and 5.5.

5.1. Small pelagic fisheries web portal

Finding and catching the right species at the right time and place, efficiently finding and catching the right species while taking into account strategic considerations like market fluctuations and multiple fisheries seasons, is a difficult task. DataBio has developed a fisheries web portal for the Norwegian pelagic fisheries, aiming at providing statistics, information and predictions for supporting such decisions.

The developed web portal is in a trial phase, with 14 selected users. It is difficult to estimate the potential improvements, but even small improvements can give great benefits in profitability and less energy consumption for the pelagic fishing fleet. Further development of the web portal is planned in several ongoing and proposed projects, with particular focus on modelling fish migrations and predicting the best fishing areas. The current operational web applications are presented in the following.

5.1.1. Use of the FishData infrastructure

The following FishData components are used in the implementation of this application:

- *Marathon* provides orchestration of services, such as *gis*, *gis-db*, *datascraper* database and web servers.
- *Chronos* provides running of periodic jobs, most notably data scraping.
- *Salt-main* provides configuration management of shore servers, facilitating version control and some user management.
- *Proxy-dcos-external* is used as load balancer and internet-facing proxy for *gis* and web services.
- *Proxy-dcos-internal* is used as load balancer and internal proxy for internal services, such as *file-access*.
- *Smd-oauth* is used for authentication and authorisation for *gis* and web services.
- *File-store* and *file-access* provides replicated and distributed storage of and access to collected data.
- *Gis* provides an open source server for sharing geospatial data.
- *Gis-db* provides a data store for *gis*.
- *Docker-registry* provides internal distribution of own container images.
- *Keyweb* is used for controlling access to data and servers.

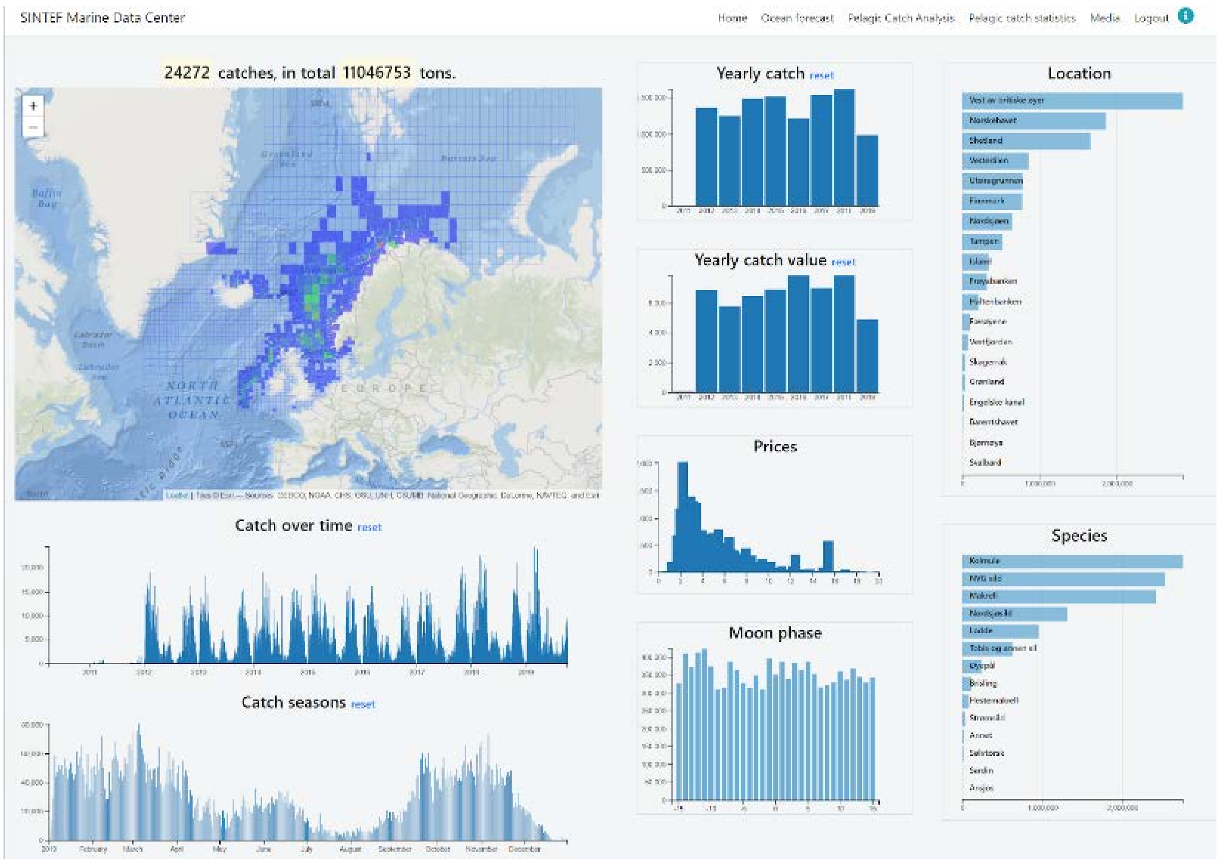


Figure 9: DataBio web app for investigating historical catches and transactions.

5.1.2. Historical catch and market analysis

This web application has been developed to enable the fishermen to understand how the market prices depend on other factors. Using this application, one can investigate how the prices have developed with factors such as species, landed quanta, year, time of year, moon phase and catch location. This web application is based on providing the possibility to filter historical catch data along the relevant factors, such as e.g. selecting only catches of mackerel last year in a short time window, and then slide this window to see how the prices varied with time. Also, similar procedures can be employed to consider variation with moon phase. Or one can go in the opposite direction and select only the catches giving the highest prices to investigate under which circumstances high prices were achieved.

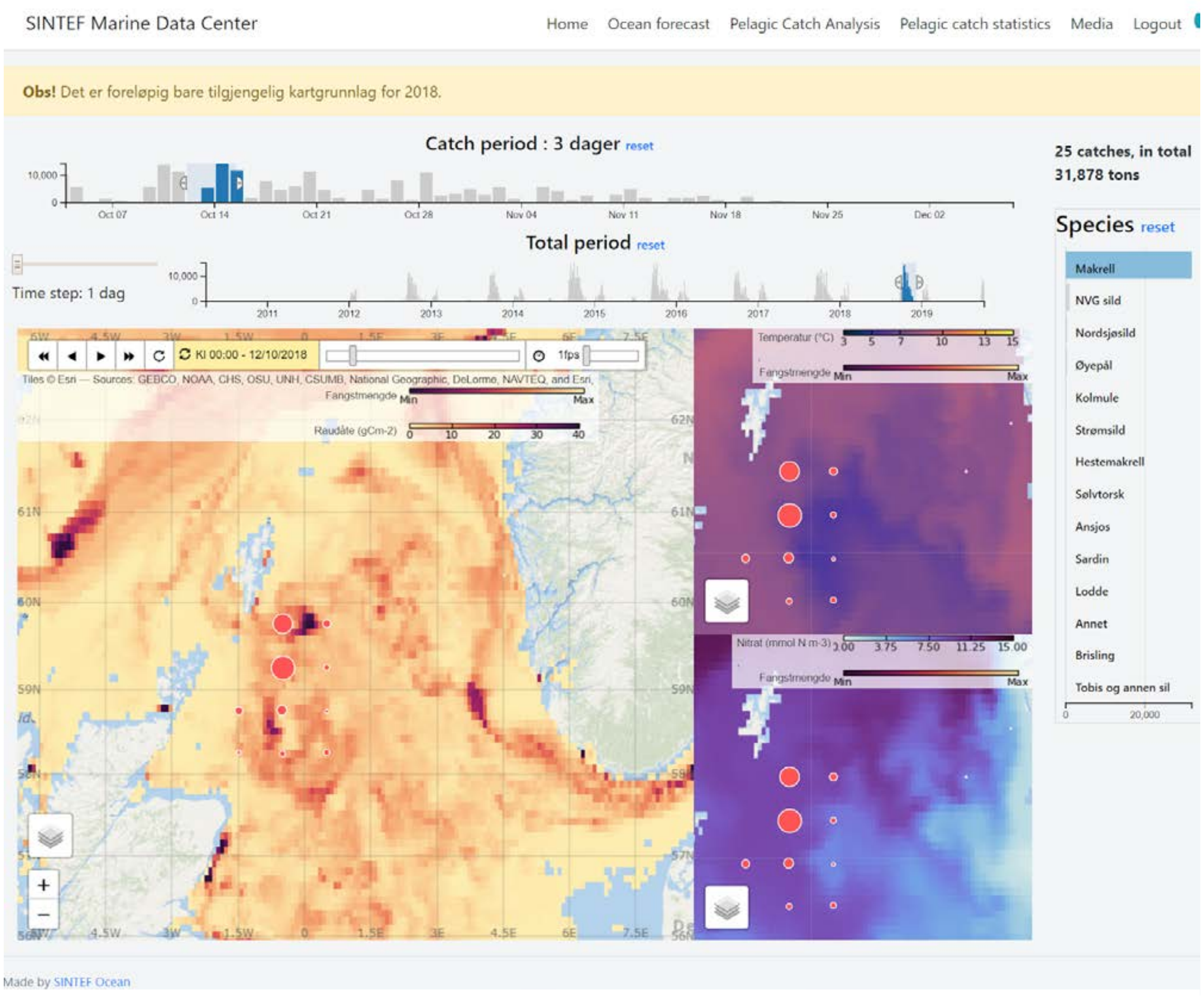


Figure 10: DataBio web app for investigating how historical catches have been influenced by ocean processes.

5.1.3. Historical catch and environment analysis

This web application enables fishermen to find how environmental factors have affected catches. In particular, one can find covariance between zooplankton concentrations and pelagic catches. It consists of three map views, two timeline plots and a species histogram as shown in Figure 10. With this setup, the user can scrutinize historic catch data together with selected bio-marine parameters. Featured functionalities are:

- Select species of interest to be visualized with circular markers with radii showing relative catch size
- Zoom to a time period for closer investigation – *Total period*
- Select catch period in days in which to summarize – *Catch period*
- Zoom and navigate the maps with synchronized mouse markers in each of the map views
- Select bio-marine overlays for each map view
- Execute time evolution playback animation of catch sizes and bio-marine parameters

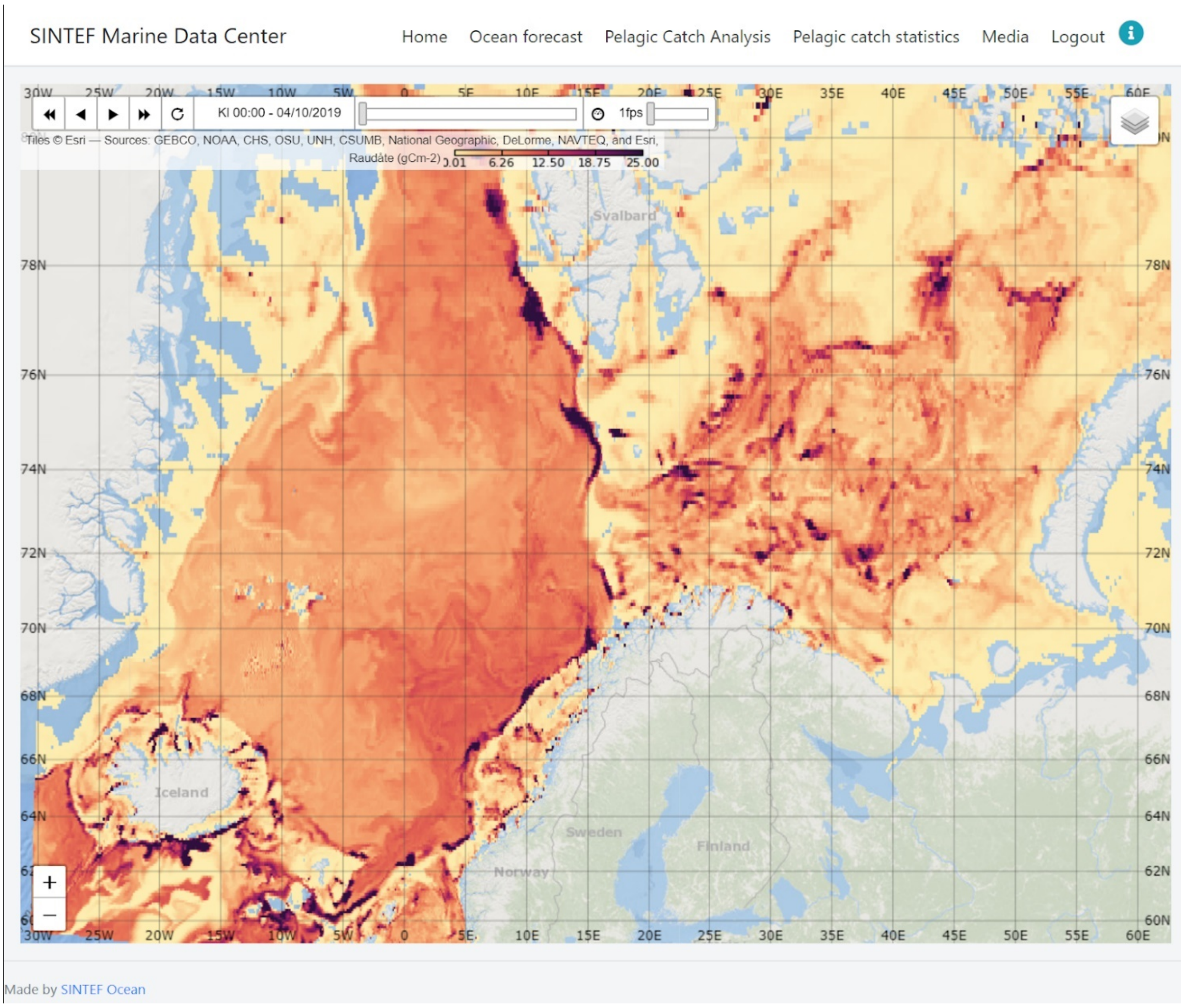
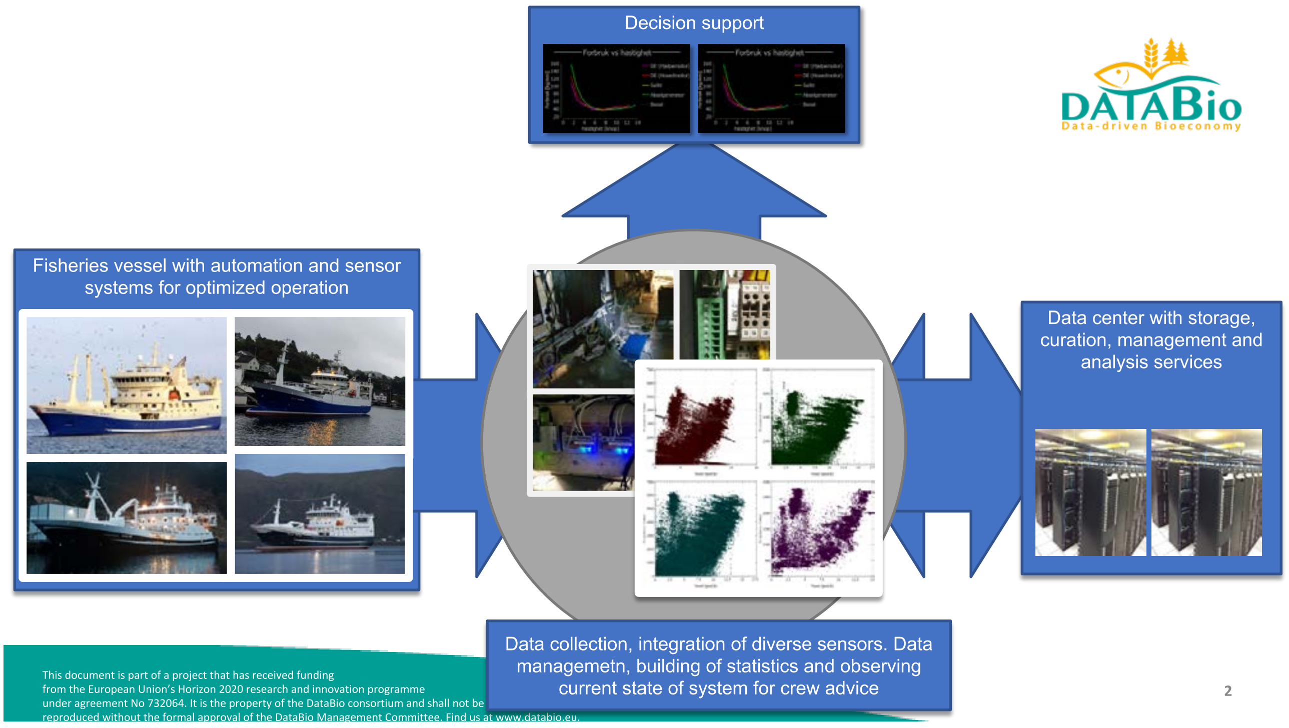


Figure 11: DataBio web app with forecasts for various ocean processes, such as zooplankton, phytoplankton, NO_3 and temperature.

5.1.4. Marine environment forecasts

This page shows forecasts for the marine environment. After fishermen have found interesting historical covariations between environment and catches, predictions of the environment can benefit choice of fishing grounds for the next days. The web application provides means to explore forecasted biomarine parameters. The user can choose from a selection of SINMOD's predicted parameters, namely: temperature, salinity, ice thickness, ice concentration, nitrate, chlorophyll, calanus finmarchicus, calanus glacialis in 4 km grid resolution. These are tiled Web Map Service layers which expose estimate values once a day, two days into the future, and five days into the past. Similar to the historic data visualization application, it is possible to select a date and run playback of the available layers.


Decision support
Fisheries vessel with automation and sensor systems for optimized operation
Data center with storage, curation, management and analysis services
Data collection, integration of diverse sensors. Data managemetn, building of statistics and observing current state of system for crew advice
DATABio
Data-driven Bioeconomy
2

This document is part of a project that has received funding from the European Union's Horizon 2020 research and innovation programme under agreement No 732064. It is the property of the DataBio consortium and shall not be reproduced without the formal approval of the DataBio Management Committee. Find us at www.databio.eu.

Figure 12: DataBio pipeline for onboard decision support.

5.2. Small pelagic immediate operational choices

Small pelagic fish species in the North Atlantic are economically important fish stocks which can be harvested both by energy efficient purse seine and energy intensive trawl gears.This has led to ships with very advanced energy- and propulsion systems which can deliver good fuel economy at different power demands. The crew is often engaged in fishing operations where management of a power plant is not a priority.

Data collection and decision support systems are installed onboard four vessels targeting small pelagic species in the North Atlantic. An extensive database of ship energy performance in these fisheries has been collected, and an operational decision support system has been developed, integrated with BigData technologies on shore. This infrastructure and decision support system will continue to be supported and expanded to other vessels.

5.2.1. Use of the FishData infrastructure

The following FishData components are used in the implementation of this application:

- *Marathon* provides orchestration of services, such as artifacts store and continuous integration build agents.
- *Chronos* provides running of periodic jobs, most notably the data pipelines from external server, through correction, analyses and to storage on *file-store*.
- *Salt-main* provides configuration management of shore servers, facilitating version control and some user management.
- *Salt-vessels* provides configuration management of ship servers, facilitating version control and access control.
- *Proxy-dcos-internal* is the load balancer and internal proxy for internal services, such as *file-access*.
- *File-store* and *file-access* provides replicated and distributed storage of and access to collected data.
- *Docker-registry* provides internal distribution of own container images.
- *Keyweb* is used for controlling access to data and vessel servers.

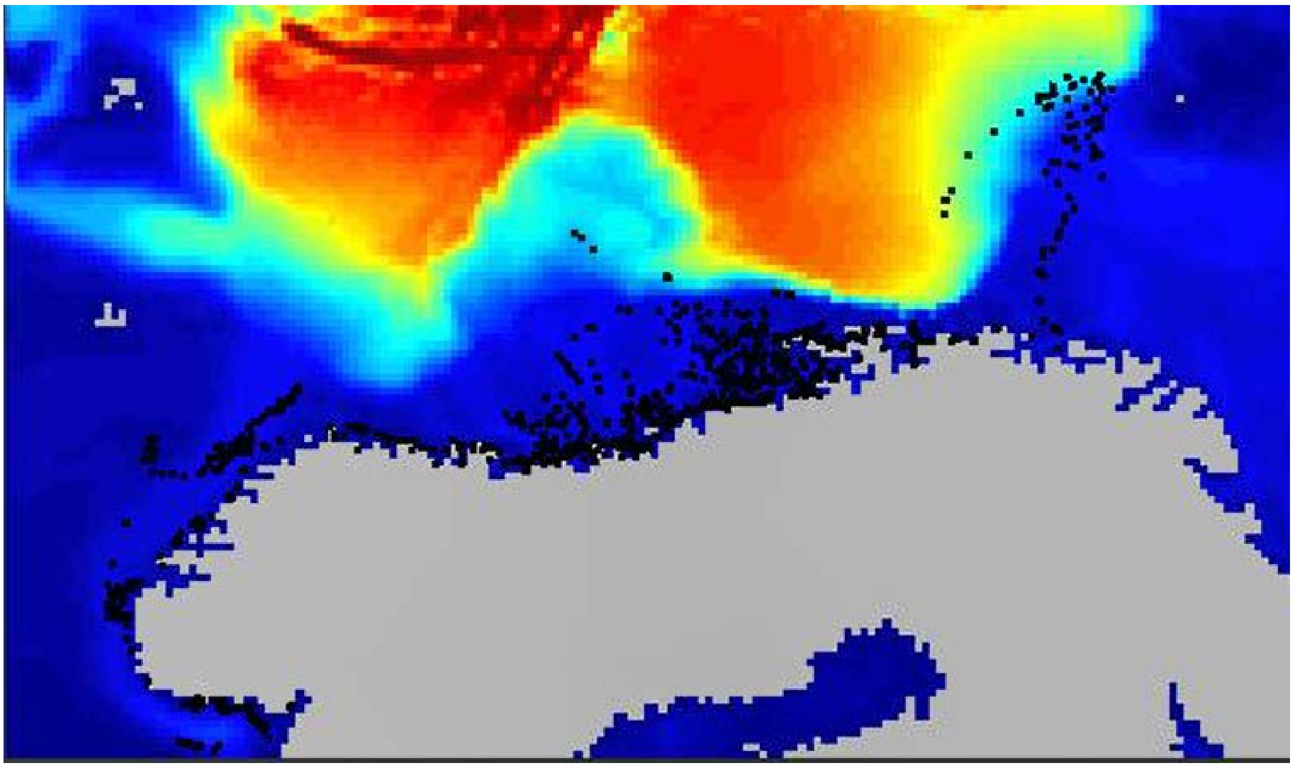


Figure 13: FishGuider screenshot of simulation of the migration of herring.

5.3. Fish migration modelling

Norwegian Spring Spawning Herring is both an economically and ecologically key species in the North Sea. Finding and catching NVG-herring at the right time and place require a forecast tool based on a marine ecosystem model including migration mechanisms for the fish. DataBio has made this by implementing a migration model for NVG-herring in the dynamic 3D ocean model SINMOD. The model is particle based and controlled by food supply, temperature and ocean current. Moreover, data from fish catch and sea surface temperature form satellites are used for data assimilation in order to improve the forecasts. This model will be further improved in the ongoing NRC financed project FishGuider by including data from AIS and chlorophyll satellite images. PhD research in FishGuider is expected to considerably improve the migration pattern in the model.

5.3.1. Use of the FishData infrastructure

The following FishData components are used in the implementation of this application:

- *Marathon* provides orchestration of services, such as the *datascraper* database.
- *Chronos* provides running of periodic jobs, most notably data scraping.
- *Salt-main* provides configuration management of shore servers, facilitating version control and some user management.
- *Proxy-dcos-internal* is used as load balancer and internal proxy for internal services, such as *file-access*.
- *File-store* and *file-access* provides replicated and distributed storage of and access to collected data.
- *Docker-registry* provides internal distribution of own container images.
- *Keyweb* is used for controlling access to data and servers.

6. Conclusions and outlook

In this report, we have provided implementation details of FishData, the prototype hardware and software infrastructure developed in SMARTFISH for handling a wide range of data related to fisheries. It consists of an "onboard" part, to be installed on vessels, and an "onshore" part, to be set up in data centres. The former enables one to:

- Acquire data from a large variety of onboard data sources
- Convert data from several widely-used industrial communication protocols into a unified, open protocol (DDS)
- Access data in real time on board
- Transfer data to shore, e.g. for long-term storage and further analysis

The onshore part enables one to:

- Receive data ("pushed") from vessels and other trusted sources
- Scrape data ("pulled") from online resources
- Store large amounts of data securely and reliably
- Run a wide range of services and jobs, such as specialised databases, analysis processes, and web services.

All parts of the infrastructure are designed to be scalable and flexible, to be suitable for research as well as for further development into industrial use. We have shown examples from the *DataBio* and *FishGuider* projects to substantiate this.

We now turn our attention to the development of demonstrators that show the utility of the infrastructure for SMARTFISH's target stakeholder groups: fishers, fishery managers and stock assessors. As a common core of these demonstrators, we plan to develop a method for automatically acquiring what we (for now) call *spatiotemporal catch/effort data* from trawlers. That is, we plan to set up a data acquisition and processing pipeline which, roughly outlined, consists of the following steps:

1. Collect data from onboard sources, including:
    - *Catch data*: Estimated weight, length, and species of *each individual fish caught by the vessel*, from WP4's CatchScanner.
    - *Vessel data*: Time series of vessel position, heading, speed, and possibly energy/fuel consumption estimates.
2. From the vessel data, automatically deduce what the vessel is doing at any given time: whether it is sailing, trawling, waiting, etc. In particular, identify individual *trawl tracks*, and estimate the *effort* expended by the vessel during its operations, in terms of time, energy, area swept by the trawl, etc.
3. Associate each fish in the catch data with a *haul*, and create aggregated haul-level statistics that enable one to faithfully reproduce whatever statistics are needed by fishers, managers and stock assessors on shore – preferably without retaining the individual details of every fish ever caught.
4. Associate each haul with a trawl track, and thus with a set of locations in space and time (hence the use of the phrase *spatiotemporal* above) as well as a measure of effort expended on the catch.

Presented in a suitable user interface together with supplemental data, this will enable users to understand:

- How much fish gets caught at a given time and place
- How difficult it is to catch it (which is an indication of its abundance)
- How much bycatch was mixed in
- Details of the size distributions of the various species.

And unlike official catch reports and landing slips, this information should also include discards. All of this is expected to be highly useful to all of SMARTFISH's target stakeholder groups:

- Fishers may use it to identify the most promising fishing grounds and plan their operations in the long and short term, and possibly also as documentation for seafood certification and labeling schemes.
- Stock assessors may use it as additional input to achieve greater accuracy in their assessments of the abundance and state of fish stocks.
- Fishery managers may use it to identify areas eligible for closing due to excessive amounts of bycatch, to identify fishing "hot spots", and to guide targeted inspections.

Outside SMARTFISH, we will continue to develop the FishData infrastructure in *SFI Harvest*[4]. This is a *centre for research-based innovation*: a long-term R&D collaboration between industry, research and academia. Further developments planned in SFI Harvest include the ability to collect ocean data *in situ*, such as hydroacoustic data and water sample analysis results.

On the catch reporting and analysis side, we don't yet have concrete industrialisation plans for FishData. However, the Norwegian Directorate of Fisheries have recently announced plans for a complete digital transformation of the Norwegian catch monitoring scheme. Their long-term goal is for catch recording to become completely automated and far more comprehensive than today, and for it to be moved as close to the point of harvest as possible. Over the coming years, we therefore expect an increasing demand for technologies that enable this kind of recording and reporting, and the combination of FishData with other SMARTFISH technologies (e.g. CatchScanner) may fit the bill very nicely. We are in talks with the Directorate and other actors about this, and we have several research grant proposals in the works to address the upcoming issues. This includes proposals targeted at European calls, as we expect the resulting technology to be applicable to all European fisheries.

[4] SFI Harvest – Technologies for sustainable biomarine value creation, funded by the Research Council of Norway under grant agreement no. 309661, 2020–2028. https://www.sintef.no/projectweb/harvest/

References

[1] L. T. Kyllingstad, K.-J. Reite and P. G. Auran, "SMARTFISH H2020 D5.1: FishData system specification," SINTEF Ocean, Trondheim, 2019.

[2] J. A. Fernandes, I. Quincoces, K.-J. Reite, Z. Uriondo, K. G. Aarsæther, J. Uranga, M. Nuñez, D. Zmicjer, J. Haugen and F.-A. Michelsen, "DataBio deliverable D3.3 – Fishery Pilot Final report," 2019.

[3] A. Knöpfel, "FMC Quick Introduction," FMC Consortium, June 2007. [Online]. Available: http://www.fmc-modeling.org/quick-intro. [Accessed 20 December 2019].

[4] A. Ronacher, "Flask," [Online]. Available: https://palletsprojects.com/p/flask/. [Accessed 20 December 2019].

[5] Red Hat, Inc., "Gluster: Storage for your cloud," [Online]. Available: https://www.gluster.org/. [Accessed 23 October 2019].

[6] SaltStack, Inc., "SaltStack," [Online]. Available: https://www.saltstack.com/. [Accessed 16 December 2019].

[7] Mesosphere, Inc., "Marathon: A container orchestration platform for Mesos and DC/OS," [Online]. Available: https://mesosphere.github.io/marathon/. [Accessed 22 October 2019].

[8] Apache Software Foundation, "Chronos: Fault tolerant job scheduler for Mesos," [Online]. Available: https://mesos.github.io/chronos/. [Accessed 22 October 2019].

[9] HAProxy, "The Reliable, High Performance TCP/HTTP Load Balancer," [Online]. Available: https://www.haproxy.org/. [Accessed 20 December 2019].

[10] Mesosphere, Inc., "Marathon-lb is a service discovery & load balancing tool for DC/OS," [Online]. Available: https://github.com/mesosphere/marathon-lb. [Accessed 20 December 2019].

[11] aptly, "Debian repository management tool," Express42, [Online]. Available: https://www.aptly.info/. [Accessed 20 December 2019].

[12] Open Source Geospatial Foundation, "GeoServer," [Online]. Available: http://geoserver.org/. [Accessed 20 December 2019].

[13] The PostgreSQL Global Development Group, "PostgreSQL: The World's Most Advanced Open Source Relational Database," [Online]. Available: https://www.postgresql.org/. [Accessed 20 December 2019].

[14] Open Source Geospatial Foundation, "PostGIS – Spatial and Geographic objects for PostgreSQL," [Online]. Available: https://postgis.net/. [Accessed 20 December 2019].

[15] Network Working Group of the IETF, "RFC4251: The Secure Shell (SSH) Protocol Architecture," January 2006. [Online]. Available: https://tools.ietf.org/html/rfc4251. [Accessed 11 December 2019].

[16] Network Working Group of the IETF, "RFC 4252: The Secure Shell (SSH) Authentication Protocol," January 2006. [Online]. Available: https://tools.ietf.org/html/rfc4252. [Accessed 11 December 2019].

[17] J. A. Donenfeld, "WireGuard: fast, modern, secure VPN tunnel," Edge Security, [Online]. Available: https://www.wireguard.com/. [Accessed 19 December 2019].

[18] D. Miller, "Linux Kernel Mailing List: Re: [PATCH net-next v2] net: WireGuard secure network tunnel," 9 December 2019. [Online]. Available: https://lkml.org/lkml/2019/12/8/257. [Accessed 19 December 2019].