Aalborg Universitet

Workflow-based automatic processing for Internet of Floating Things crowdsourced data

Montella, Raffaele; Di Luccio, Diana; Marcellino, Livia; Galletti, Ardelio; Kosta, Sokol; Giunta, Giulio; Foster, Ian

Published in:
Future Generation Computer Systems

DOI (link to publication from Publisher):
10.1016/j.future.2018.11.025

Creative Commons License
CC BY-NC-ND 4.0

Publication date:
2019

Document Version
Accepted author manuscript, peer reviewed version

Link to publication from Aalborg University

Citation for published version (APA):
Montella, R., Di Luccio, D., Marcellino, L., Galletti, A., Kosta, S., Giunta, G., & Foster, I. (2019). Workflow-based automatic processing for Internet of Floating Things crowdsourced data. *Future Generation Computer Systems*, *94*, 103-119. https://doi.org/10.1016/j.future.2018.11.025

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal -

Take down policy
If you believe that this document breaches copyright please contact us at vbn@aub.aau.dk providing details, and we will remove access to the work immediately and investigate your claim.

Downloaded from vbn.aau.dk on: June 18, 2025

Accepted Manuscript

Workflow-based automatic processing for Internet of Floating Things crowdsourced data

Raffaele Montella, Diana Di Luccio, Livia Marcellino,
Ardelio Galletti, Sokol Kosta, Giulio Giunta, Ian Foster

PII: S0167-739X(18)30767-2
DOI: https://doi.org/10.1016/j.future.2018.11.025
Reference: FUTURE 4589

To appear in: *Future Generation Computer Systems*

Received date : 31 March 2018
Revised date : 6 September 2018
Accepted date : 16 November 2018

Please cite this article as: R. Montella, D.D. Luccio, L. Marcellino et al., Workflow-based automatic processing for Internet of Floating Things crowdsourced data, *Future Generation Computer Systems* (2018), https://doi.org/10.1016/j.future.2018.11.025

This is a PDF file of an unedited manuscript that has been accepted for publication. As a service to our customers we are providing this early version of the manuscript. The manuscript will undergo copyediting, typesetting, and review of the resulting proof before it is published in its final form. Please note that during the production process errors may be discovered which could affect the content, and all legal disclaimers that apply to the journal pertain.

ACCEPTED MANUSCRIPT

Workflow-based Automatic Processing for Internet of Floating Things Crowdsourced Data

Raffaele Montella[a,b], Diana Di Luccio[a,b], Livia Marcellino[a], Ardelio Galletti[a], Sokol Kosta[d], Giulio Giunta[a], Ian Foster[b,c,e]

[a]*Department of Science and Technologies – University of Napoli "Parthenope"*
[b]*Computation Institute – The University of Chicago*
[c]*Department of Computer Science – The University of Chicago*
[d]*CMI, Department of Electronic Systems – Aalborg University, Denmark*
[e]*Data Science and Learning Division – Argonne National Laboratory*

Abstract

Data from sensors incorporated into mobile devices, such as networked navigational sensors, can be used to capture detailed environmental information. We describe here a workflow and framework for using sensors on boats to construct unique new datasets of underwater topography (bathymetry). Starting with a large number of measurements of position, depth, etc., obtained from such an Internet of Floating Things, we illustrate how, with a specialized protocol, data can be communicated to cloud resources, even when using delayed, intermittent, or disconnected networks. We then propose a method for automatic sensor calibration based on a novel reputation approach. Sampled depth data are interpolated efficiently on a cloud computing platform in order to provide a continuously updated bathymetric database. Our prototype implementation uses the FACE-IT Galaxy workflow engine to manage network communication and exploits the computational power of GPGPUs in a virtualized cloud environment, working with a CUDA-parallel algorithm, for efficient data processing. We report on an initial evaluation involving data from a sailing vessel in Italian coastal waters.

Keywords:
Workflows, Data crowd sourcing, Mobile Computing, Cloud Computing, GPGPU Virtualization, Internet of Things, Bathymetry interpolation

ACCEPTED MANUSCRIPT

ACCEPTED MANUSCRIPT

1. Introduction

The rapid spread of Internet of Things (IoT) technologies has greatly increased the use of geographic data [1]. In the work reported here, we leverage this trend to develop a new approach to the problem of obtaining high resolution 3-D maps of the sea floor, as required, for example, to model the impact of sea storms on human coastal activities [2, 3, 4], the diffusion and dispersion of sea pollutants [5, 6, 7], and the drift of floating objects, as required for safety at sea [8]. We show how to use crowdsourced data [9] from leisure boats to create unique and dense sea bathymetry datasets that, due to the large numbers of such vessels, can be particularly detailed in shallow waters and coastal areas.

Crowdsourced data collection processes can produce datasets that are large in size and expansive in geographic extent [10]. However, the resulting measurements can also be less reliable than those obtained via structured surveys carried out with more accurate (and expensive) scientific instruments [11]. Thus, we see growing interest in developing effective quality control mechanisms. Instead of reformulating the problem as an automatic learning problem [12], in this work we present a novel approach based on a Collaborative Reputation System [13] applied to Internet of Things sensed data[14]. However, this approach, when applied in the environmental sciences, usually assumes a high-quality internet connection [15].

One popular strategy for collecting crowdsourced data is to link off-the-shelf sensors over the network to cloud computational resources and storage [16] to produce a "Sensor Instrument as a Service" (SIaaS) [17]. Even small mobile devices can then be used as a data collection platform [18].

In this work we present a SIaaS system that processes leisure boat sensor data (GPS position and depth sounder) to produce and update a detailed 3-D sea floor (bathymetry) map. (We focus on the bathymetry problem because bathymetric surveys are rare and expensive, and high-resolution publicly available datasets are difficult to obtain [19].) Data are collected via an Internet of Floating Things (IoFT) ecosystem called DYNAMO (Distributed leisure Yacht-carried sensor-Network for Atmosphere and Marine data crowdsourcing applications, see Fig. 1) that we have presented previously [20, 21].

We use the cloud-hosted FACE-IT Galaxy workflow engine [22, 23] to manage and integrate the data collected via DYNAMO, being run at regular intervals to extract data from the acquired database, selecting only the sampled depth, in order to interpolate the extracted data to obtain

ACCEPTED MANUSCRIPT

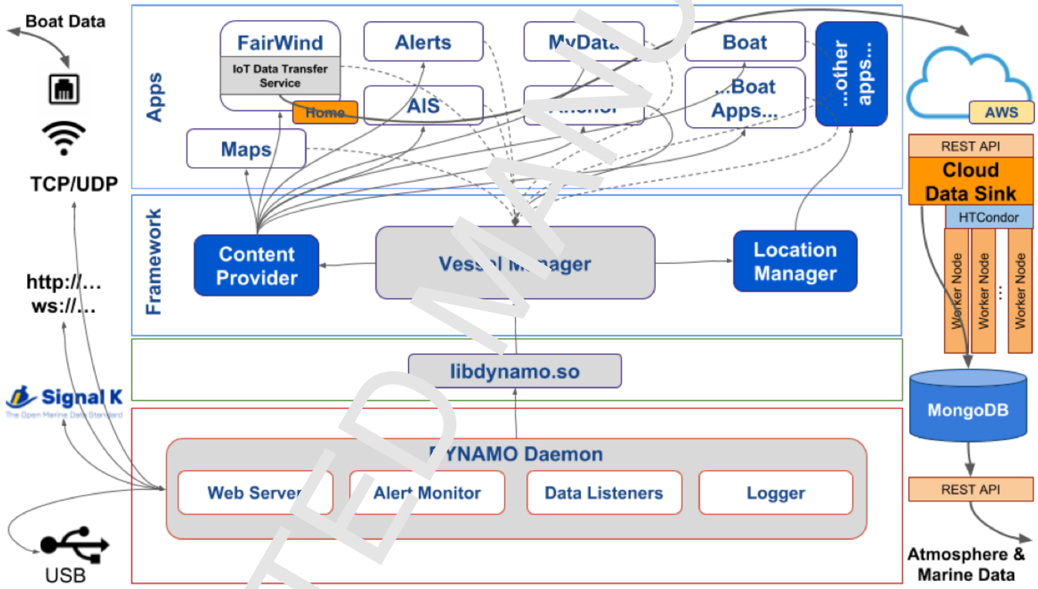

Figure 1: Smart devices in the context of an Internet of Floating Things make data crowd-sourcing affordable. DYNAMO-equipped pleasure boats can contribute to the creation of open, high-resolution bathymetry datasets.

ACCEPTED MANUSCRIPT

the available depth data and update the dataset. In order to process the large amount of data to be processed at each such execution, the workflow uses CUDA-accelerated algorithms on virtual machine (VM) instances with NVIDIA CUDA support, as supplied by the Amazon Web Services (AWS) cloud. To minimize the costs of using these expensive GPU-equipped virtual machines, we use GVirtuS [24] to virtualize the CUDA calls [25] so that the interpolation algorithm can be executed on regular VMs and the CUDA-enhanced algorithms on GPU-equipped VMs, as needed.

We report preliminary results in terms of data sampling using a vessel sailing in the East Sector of the Central Tyrrhenian Sea during Summer 2017. We deployed the DYNAMO system on the vessel and collected data using the on-board instruments including GPS, compass, tachometer, trim, wind and other environmental sensors, and echo sounders. DYNAMO then using a custom delay-tolerant transfer protocol to transfer the collected data to the cloud counterpart, where they were processed using the FACE-IT Galaxy workflow engine.

Data processing produces a collection of data tiles (see Section 3.5), which, similarly to the world maps' tiles, contain information about the seafloor at different levels of detail. We implemented two interpolation algorithms, namely the Inverse Distance Weighting (IDW) and Kriging, to properly position the newly measured data [26]. Indeed, these data usually contain unavoidable errors, given that they get collected using barely calibrated sensors on leisure vessels by volunteers. Moreover, to minimize the effect of the erroneous measurements, we design and implement a reputation-based algorithm, which uses the measurements of multiple sensors to improve the knowledge of errors and to enhance the accuracy of each sensor [27].

As many of the algorithms needed for the system are computationally expensive, we have implemented efficient versions that exploit the power of GPUs, and we make use of GPU virtualization techniques to limit the costs of running the platform[28]. The experiments show that the system processes the collected data successfully and can easily scale to support larger inputs. Indeed, we show that gathering environmental data, performing the needed processing and homogenization, and supporting experiments of computational environmental science can be realized at large scales and at modest costs. These results are also confirmed by extensive simulations.

The rest of the paper is organized as follows. Section 2 reviews related work and technical background; Section 3 describes the system architecture; and Section 4 describes the CUDA based interpolation algorithm. We discuss

ACCEPTED MANUSCRIPT

our proposed method for automatically adjusting vessel data in Section 5 and, in Section 6, describe how we extended FACE-IT Galaxy to support our application and how the application is implemented. Finally, we report in Section 7 on tests of all components of our system and conclude and discuss future work in Section 8.

2. Related work

2.1. Crowdsourced Bathymetry

In crowdsourced data collection, a dataset is constructed with the help of a large group of people [29]. This technique has been used in marine science to update nautical charts, increase public safety and environmental stewardship, and increase navigational efficiency. Many projects, such as ARGUS (Autonomous Remote Global Underwater Surveillance: `argus.survice.com`), have sought to involve the marine community in data collection. In particular, the ARGUS crowdsourcing system involves cooperative surveying through the acquisition and collective processing of bathymetry data that can significantly supplement and enhance the accuracy and efficiency of standard hydrographic surveying. The International Hydrographic Organization Data Centre for Digital Bathymetry (IHO DCDB: `www.iho.int`) has a long history of encouraging the collection of crowdsourced bathymetry data to identify uncharted features, verify charted information, and support scientific studies in marine areas where no depth data exists.

Other data collection and integration efforts could benefit from crowdsourcing. For example, the European Marine Observation and Data Network (EMODnet: `www.EMODnet-bathymetry.eu`) [30] assembles marine data (bathymetry is a key dataset) to make data resources from across Europe, collected in a fragmented way for many years, more available to public and private users. The U.S.-based Ocean Observatories Initiative (OOI : `oceanobservatories.org`) implements and manages a large sensor infrastructure for marine data [31].

2.2. GPU Virtualization

As noted above, the large quantities of crowdsourced data to be collected require GPGPU processing. Here we can take advantage of remote GPU virtualization as provided by rCUDA [32], which uses a split-driver approach to allow CUDA-enabled applications to be used without modification, while executing CUDA kernels on a remote or local GPGPU [33]. Studies shows

ACCEPTED MANUSCRIPT

that the overhead due to remote GPU usage in a high performance network fabric does not exceed 4% [34]. Currently, rCUDA provides high performance CUDA virtualization and is updated to support the latest CUDA 8.0 framework and its ancillary libraries, so we can take full advantage of InfiniBand and the related support for RoCE (RDMA over Converged Ethernet) networks. A binary distribution of rCUDA is freely available at the rCUDA web site (`rcuda.net`), but is not open source.

2.3. CUDA Interpolators

Spatial interpolation is a crucial task in geographic information science. Frequently used interpolation algorithms include Inverse Distance Weighting (IDW), Kriging, nearest neighbors, and discrete smoothing interpolation [35, 36, 37]. (Falivene et al. [38] provide a comparative survey.) These interpolation algorithms are computationally expensive and thus parallel implementations can be needed for large datasets. There are many research efforts in this context that target different parallel architectures and environments [39]; for example: multicore-cluster approaches, domain decomposition strategies, and parallel pipeline procedures. Recently, GPUs [40] have been used to accelerate some interpolation algorithms, with good results [41, 42, 43].

We consider here the IDW interpolation algorithms that have been already parallelized on several platforms. Henneböhl et al. [44] provide a good survey of parallel implementations of IDW algorithms. In order to achieve better geophysically consistent results, we implemented a CUDA-enabled ordinary Kriging. This advanced geostatistical procedure computes an estimated surface as a weighted linear/nonlinear combination of scattered set of points. The Kriging procedure assumes that the distance between sample points reflects a spatial correlation that can be used to explain the surface variation [45]. Isaaks et al. [46] proposed an expression of this weighted combinations in terms of covariance matrices. The weights can be calculated using different approaches (i.e., different types of Kriging can be applied) depending on the stochastic properties of the sample points [47]. These algorithms, implemented for a single CPU, are computationally onerous: the computational cost scales as the cube of the number of sample points [41]. Thus, GPUs and CUDA have recently been used to accelerate the computations [48, 41, 49].

ACCEPTED MANUSCRIPT

3. Architecture

We now describe the main technologies that are at the core of our novel application prototype, which as a highly complex system leverages many state-of-the-art software and hardware components.

3.1. Leisure Vessels as Sensors

The advent of cloud services has helped studies in ocean modeling by providing convenient access to unprecedented computational and storage resources, allowing researchers to obtain new findings by processing larger data [50]. Nevertheless, the fundamental task of collecting the needed data in the first place remains an issue. Traditional approaches such as research expeditions are usually expensive to perform and limited in the areas they can cover. Moreover, some coastal areas present challenges for such traditional methods: for example, they might be too dangerous to be investigated using large ships. To overcome these limitations, crowdsourcing techniques such as the FairWind system that we developed previously [51] have been proposed. These methods can be used on small leisure vessels to collect data from their sensors, allowing ordinary citizens to contribute data of considerable importance for science, engineering, and management of natural resources.

The system that we work with here, DYNAMO, collects data from onboard sensors and instruments connected with different local network protocols acting as data logger, router, and gateway for NMEA (`www.nmea.org`), SeaTalk (`www.raymarine.com`), and SignalK data.

In previous work, we developed an Android crowdsourcing application, FairWind, accessible in the Google Play Store [51]. This approach allowed for fast and easy deployment, but the Android operating system leads to limitations concerning the amount of memory that can be allocated, Garbage Collector management [52], and CPU utilization. In particular, the data collection process can sporadically become highly intensive, which may cause the operating system to kill the application, negatively impacting the results of the crowdsourcing activity. To avoid such issues, we developed DYNAMO as a customized Android distribution. We moved the implementation of the data logging and routing features into the Linux part of the Android OS, while retaining the graphical user interface as a normal Android application, given that it is not resource intensive and thus not at risk of being *killed* by

ACCEPTED MANUSCRIPT

the Android OS. The graphical part of the resulting framework is inspired by our previous work FairWind, and thus we refer to it as FairWind-Home.

Being a customized Android OS, DYNAMO can be deployed as a pre-installed solution in marine devices, in a similar way to other marine tools. Then, users can visualize the collected information via FairWind-Home, while at the same time the framework transmits the data to the cloud whenever possible. DYNAMO is highly privacy-oriented, allowing users to customize the amount of data that they share and to select the level of data anonymization preferred. Users can also reduce energy consumption by choosing the moments when the framework should try to send the data to the cloud.

DYNAMO uses on-board instruments including GPS, compass, tachometer, trim, wind and other environmental sensors and echo-sounders to collect data, which are then stored on board in packs of SignalK (`signalk.org`) updates in JSON format. These data are communicated, securely and reliably, to a cloud-hosted server, whenever a network connection is available. The information are then stored in a NoSQL database in the cloud for future processing: we host the whole computational infrastructure on AWS to manage the amount of new data available in each run.

DYNAMO defines a framework, the *Vessel Manager*, that allows for the development of third-party Android applications, called “Boat Apps,” that can interact with vessel sensors and actuators. The FairWind-Home application's basic features could be straightforwardly extended and customized to interact with such apps. From the marine electronics point of view, this framework is one of the most crucial innovations introduced by DYNAMO and its ecosystem. Figure 1 presents a detailed picture of the DYNAMO system.

3.2. Data Transfer in Extreme Environments

In order to mitigate the drawbacks of highly delayed and unstable networks [53], exploit the enormous potential of data crowdsourcing, and enforce security (since sensitive data such as locations can be involved), we leverage in this work a data transfer framework that we have developed [54] for such applications. The Internet of Floating Things Data Transfer Framework acts as a bridge between the vessel segment and the cloud segment of the proposed application [55]. In the Internet of Things context, applications typically leverage asynchronous protocols such as MQTT [56], a publish/subscribe protocol designed for telemetry transport. However, MQTT is not suitable for marine settings, in which offline periods can last for hours or even days.

ACCEPTED MANUSCRIPT

In the kind of application we present in this paper, the constraint is that the data be transferred as quickly as possible whenever the network is available, even if only for a short while.

Figure 2 provides a high-level view of the proposed framework, which is tolerant to unreliable or intermittent networking by design, as required for our geographical data crowdsourcing applications. Data can be collected continuously, stored on board, and sent to the cloud when the vessel is in network range. The framework is implemented as a HTTP-based, firewall- and proxy-friendly transfer protocol that enforces security without the mandatory use of HTTPS thanks to the use of mainly GET and POST with file attach verbs. A key feature of the proposed framework is the use of concurrent streams of parceled data in order to achieve the best performance when enough connection quality is available. We designed the protocol to be tolerant to network delays and, in general, to data transfer failures, as required for effective functioning in marine environments. Each data parcel can be signed and encrypted in order to enforce data integrity and, above all, user privacy.

Upon receipt of a data parcel by the cloud service, the signature is verified with the source public key, the parcel is decompressed, and the local hash function is evaluated in order to perform signature verification. If the data parcel passes this stage, its data are ready to be stored in a NoSQL database. We use a Data Access Layer in order to decouple the Transfer REST API Engine from the actual database (for example MongoDb). Once the data are stored in the NoSQL database, they can be consumed by different applications. In order to maximize the data transfer process, data compression can also be applied.

The framework is architecture and application independent: each feature can be activated, or not, depending on the operational context. It supports bidirectional payloads for loosely coupled remote event firing and could be extended to support a mesh-based data parcel routing using other nodes as hops. Thus the framework can be used in different scenarios, such as marine, automotive, robotics and similar applications.

We recently extended the Internet of Floating Things Data Transfer Framework by implementing a Node.js (`https://nodejs.org`) high throughput software component that enables any SignalK-equipped system to perform data logging on the DYNAMO cloud infrastructure. By making this component available in a wider open source marine electronics environment as a plugin freely available for download, we aim to increase the number of

ACCEPTED MANUSCRIPT

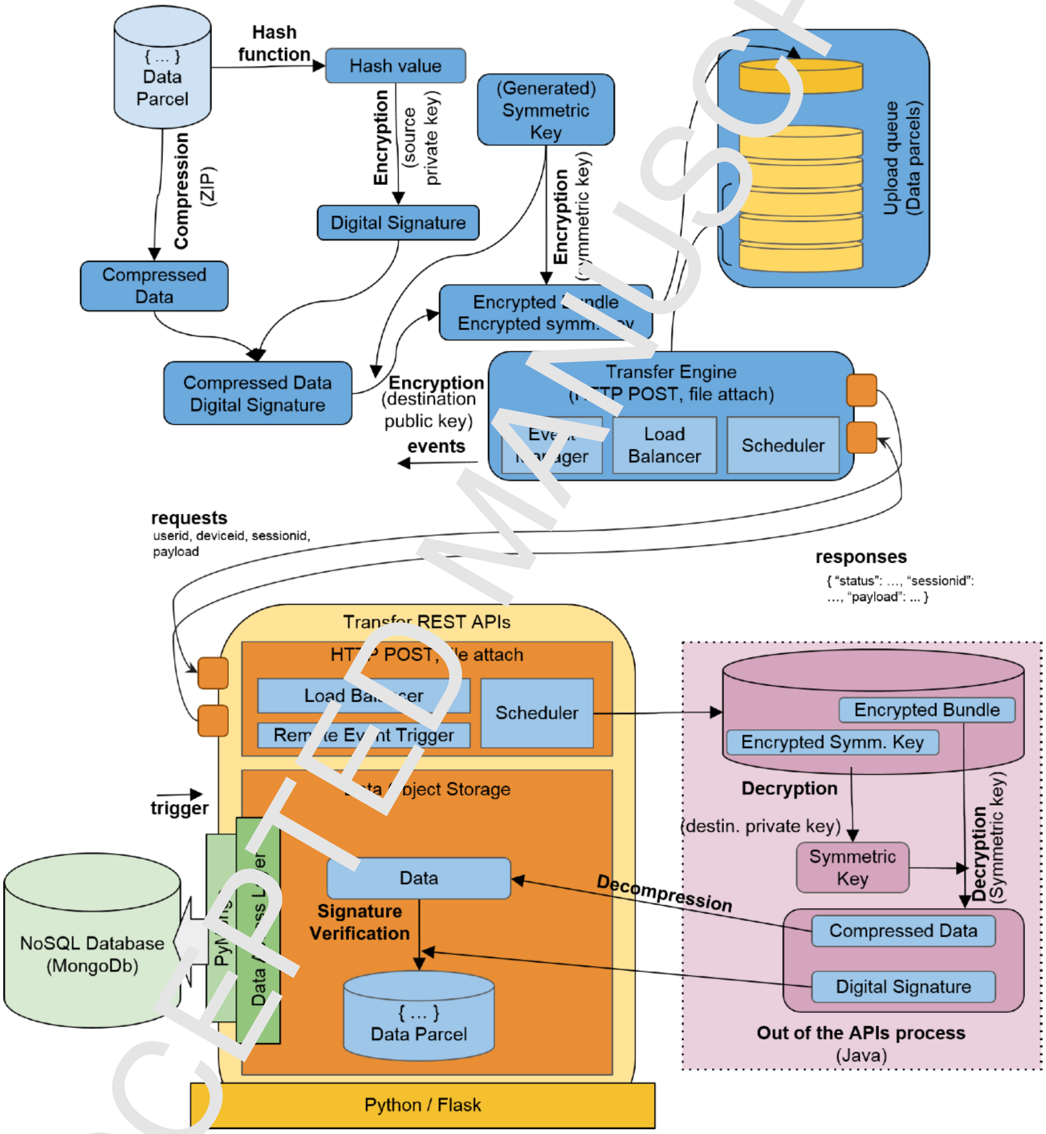

Figure 2: Block diagram of the data transfer protocol implemented using the IoFT data transfer framework.

ACCEPTED MANUSCRIPT

boats that can contribute to our data crowdsourcing mission.

3.3. GPU Virtualization

We leverage GPU virtualization techniques to execute the heavy CUDA functions of our solution on a remote GPU. We make use of GVirtuS, the most popular and robust open source solution for GPU virtualization. GVirtuS is composed of two parts: a back-end and a front-end, a technique known as split-driver model or driver paravirtualization [57].

The back-end is the component that is installed in a machine with GPU access and takes care of executing the offloaded CUDA functions. Since the back-end needs to access the GPU directly, it must be installed in a privileged domain [58]. Different clients can then access the GPU at the same time, even remotely, by going through the GVirtuS back-end [59], allowing a better utilization of the GPU resources [60].

The front-end is the component that is used by developers to add remote GPU support to their applications. The front-end provides an API with function signatures similar to those of the CUDA functions. Whenever a function is called, its name and the addresses of the input parameters, variables and host/device pointers, are encapsulated in a buffer. These data are then sent to the back-end through the GVirtuS *communicator*, completely transparently to the developer. When the back-end receives the request, it executes the routine and sends a buffer containing the output variables and host/device pointers back to the front-end of the calling client [61].

GVirtuS is currently up to date with the most recent CUDA version and CUDA libraries. Importantly for our project, it has been extended to support several CUDA auxiliary libraries, such as cuFFT, cuDNN, and cuBLAS, for which NVIDIA provides GPU-accelerated libraries that implement highly optimized algorithms. Moreover, since GVirtuS is quite modular, developers can easily integrate more functionality, if needed [62].

3.4. Data-intensive Application Workflow

We use the FACE-IT Galaxy workflow engine for our workflow processing. This system extends the Galaxy bioinformatics workflow system [63] with specialized datatypes, interfaces, and other features required for earth science applications. FACE-IT Galaxy incorporates extensions that support Globus data browsing and transfer as implemented by the Globus Genomics project [64]; platform improvements used as a foundation for the earth science-specific applications: advanced versions of XML and JSON

ACCEPTED MANUSCRIPT

ACCEPTED MANUSCRIPT

data types, a common REST interface for remote data browsing, and RAFT files for grouping datasets into collections; the NetCDF data format and the NetCDF schema for fast and reliable NetCDF-based data file 'sniffing'; and raster and vector map data visualization. Data sources, plotting functions, format conversions, and numeric models are wrapped as individual tools that can be combined to implement diverse workflow-based reproducible applications [5, 65]. Users working on agricultural, climatic, economic, and other problems can use FACE-IT as cloud computing support for data-intensive applications. The platform can also be used in developing countries with limited Internet connections or poor and absent processing power.

We created a new FACE-IT Galaxy instance for this application, hosted on an AWS Elastic Compute Cloud (EC2) machine, that builds on and improves our prior results [66]. We forked FACE-IT Galaxy directly from the Galaxy Project instead of using Globus Genomics Galaxy. This is a strategic choice more than a mere technical issue because we re-implemented all previously developed components of FACE-IT Galaxy as a tool-shed. We thus enforce our strict constraint of avoiding any core source code modifications, so that any new FACE-IT Galaxy workflow-based project can work with the latest regular Galaxy version and thus leverage the work of the Galaxy developer community.

The Job Runner is the Galaxy component dedicated to actual tool execution and to interfacing with the local scheduling. FACE-IT Galaxy uses the HTCondor Job Runner, which works with EC2. A monitor service analyzes the number, type, and workload of the VM instances. If a tool needs an instance that is not available, the HTCondor Job Runner starts a new one, deploying all needed services, adding a shared file system, and starting monitoring. Time-related and instance-related policies are implemented in order to ensure scalability, for example by selecting high performance instance types or by using scavenged resources instead of on-demand instances. The latest version of the HTCondor Job Runner implemented for this application supports EC2 virtual clusters created with CfnCluster (`github.com/awslabs/cfncluster`), a framework for deploying and maintaining high performance computing clusters on AWS.

The previous FACE-IT Galaxy implementation relied on an Elastic Block Storage volume attached to the Galaxy instance and configured as an NFS server. Each working node instantiated by the HTCondor Job Runner imported the job scratch directory acting as an NFS client. Here, we use instead the AWS Elastic File System (EFS) to provide simple and scalable file stor-

ACCEPTED MANUSCRIPT

age for use with AWS EC2 instances. We chose to use EFS because is simpler to manage than NFS, thanks to an interface that enables the developer to create and configure file systems quickly and in a straightforward fashion. EFS allows us to achieve our goal of elastic storage capacity that grows and shrinks automatically as files are added or removed.

3.5. Data Tiles

The use of tiles and pyramids to achieve discrete zoom level maps is common in internet mapping [67]. Briefly, given a zoom level z, such that $z \geq 0$, the map of the whole globe is represented by a matrix of 2^z by 2^z tiles, each at a given pixel resolution (usually 256×256 pixels). Tiles are pre-rendered or dynamically computed with data drawn using the Web Mercator projection [68].

We believe that a similar structure can be used to manage data, with a tile containing data instead of rendered images. We are confident that a novel software infrastructure based on this approach can push georeferenced data processing and management to a higher level.

To this end, we define, in a similar manner to the classic image-based tile, the "data tile" (*dile*) as a georeferenced matrix of 360×180 data cells, as presented in Figure 3. Then, given a zoom level z, we represent the whole globe by a matrix of 2^z by 2^z *diles* (each 360×180 cells), in which data are stored unprojected. The ground resolution of each cell at zoom level z is thus $\frac{1}{2^z}$ degrees. Each *dile* is stored as a compressed NetCDF file containing only one variable at a given time step and vertical level. *Diles* are represented by URIs and can be stored by using various technologies to match application needs: for example, file system files, S3 buckets, or Globus endpoints. Moreover, *diles* can be created, searched, and accessed in parallel.

NetCDF files containing multidimensional environmental data may be directly accessible using commonly used internet protocols such as HTTP or FTP, or indirectly via a legacy OpenDAP server. A data crawler [69] can scrape the web in search of environmental data (`github.com/hpsc-smartlab/NetCDFScavenger`).

In numerical weather predictions finer domains are often nested in coarse domains in order to increase model resolution and, consequently, data density in certain areas. This approach saves both computing time and storage needs. In this scenario the use of *diles* enables the final user to get data accordingly with the model resolution at a given discrete zoom level. In a typical configuration, three domains are nested with average cell sizes of

ACCEPTED MANUSCRIPT

ACCEPTED MANUSCRIPT

25km, 5km, and 1km (a 1:5 ratio, 25-5-1). We can re-arrange such data on the *diles* schema to match the discrete zoom level and the data consistency. Considering the 25-5-1 setup, the results are stored in data tiles at zoom levels 2, 5, and 7.

The use of *diles* is feasible with high-resolution multidimensional datasets with a cell size smaller than 1°: *i)* Instead of a monolithic file, the multidimensional dataset is partitioned in smaller slices, enforcing full topological coherence. *ii)* The *diles* can be named by using an URL schema parameterized with the dataset name, variable, time, level, *zoom*, x and y; where *zoom*, x and y are the *dile* indices. *iii)* Spatial data processing, feature search, and machine learning algorithms can be applied to one or more datasets in a map-and-reduce fashion[70]. *iv)* Spatial data simplification methods, by which general shapes of features are retained, while eliminating unnecessary details, can be applied to datasets that are frequently accessed at a given zoom level, generating pre-computed cached data.

The use of the *diles* approach is not drawback-free. For example, the data transformations (regridding) that must be applied to surveyed or model-generated datasets can be computationally/storage expensive and can affect the overall data quality because of the interpolation/extrapolation process.

In the present application, we deal with well-known datasets that have associated self-describing metadata. Our *dile*-based approach relies on metadata indexing and uses a NoSQL database for searching operations via the key/value paradigm.

We use *diles* as our target domains in this paper because the crowdsourced data are not uniformly distributed spatially. The *dile* representation allows us to produce results with an implicit map simplification related to the zoom level (see Figure 3).

4. CUDA Bathymetry Interpolation

The topographic variability of the seafloor influences sea currents, the structure of biological populations, and ecological processes at many spatial scales. A detailed knowledge of coastal bathymetry is crucial for decision making in many application fields, ranging from navigation to the protection of artifacts. In the marine data crowdsourced scenario described in this paper, new depth data sampled by many barely calibrated echo-sounders are collected and processed daily. Interpolation algorithms for geophysics are computationally expensive and thus the use of GPU-accelerated interpolation

ACCEPTED MANUSCRIPT

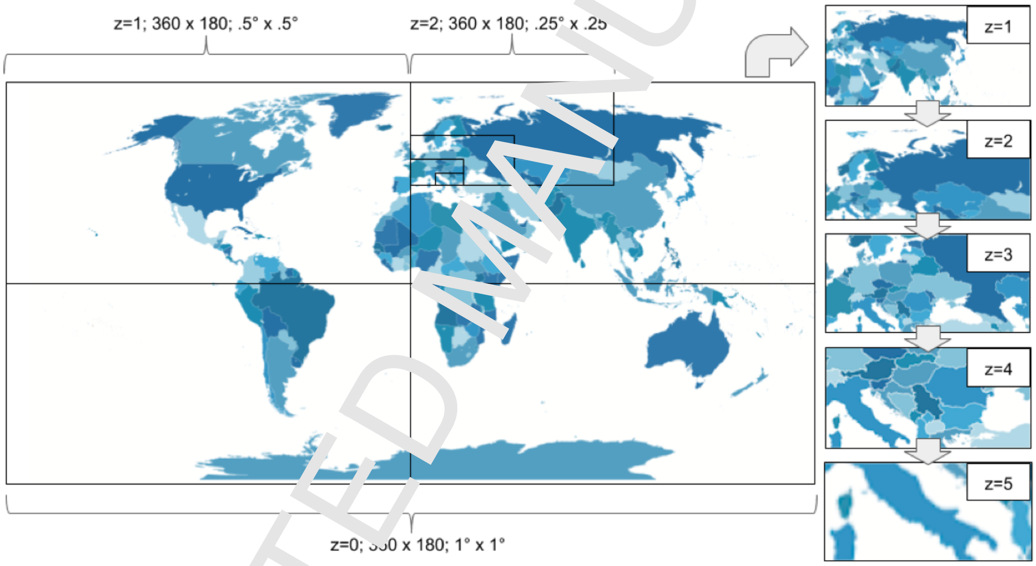

Figure 3: How *diles* represent the world map at zoom levels 0, 1, and 2. At level 0, the world is represented by a single *dile*, with each of its 360×180 cells corresponding to 1°×1°; at level 1, by four *diles* with 0.5°×0.5°cells, and so on.

ACCEPTED MANUSCRIPT

ACCEPTED MANUSCRIPT

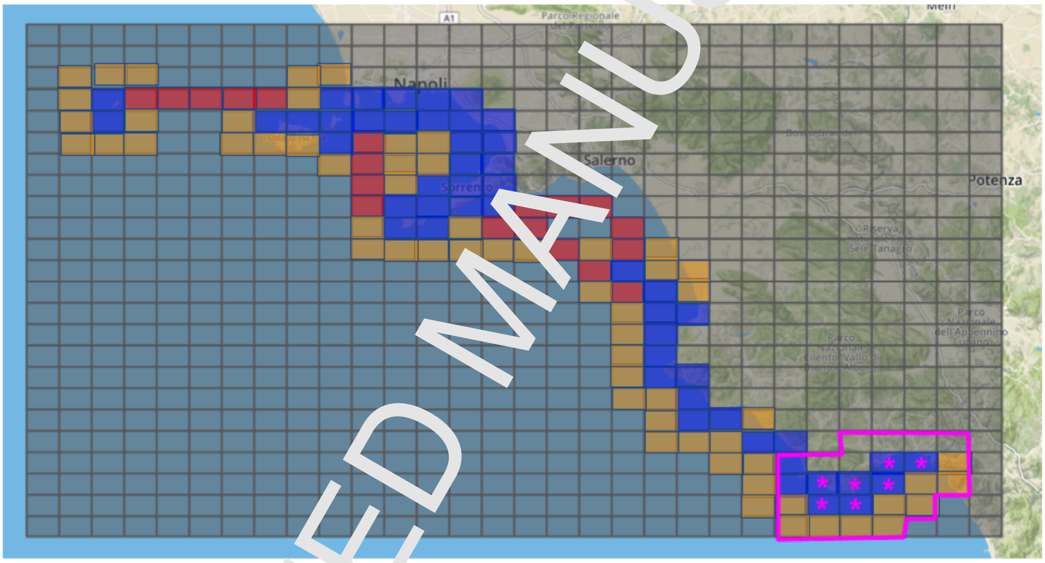

Figure 4: Our bathymetry target resolution is 1 arc second (about 30m). The figure represents the area where our first experiments have been conducted covered by *diles* at zoom level 12, or about 27m in the latitude dimension. Blue *diles*: new data points. Yellow *diles*: buffer *diles* used during daily interpolation. Red *diles*: depth points unavailable due to echo sounder technical limits. Pink boundary: the area considered for experimental data. Pink dots: the *diles* we used as a source query locations in our experiments.

ACCEPTED MANUSCRIPT

algorithms is mandatory in this application context [71]. We consider here two interpolation methods: Inverse Distance Weighting and Kriging. The first has good performance but introduces unwanted artifacts; the latter is acclaimed as one of the best solutions for this kind of application [72], albeit at a significantly higher computational cost.

4.1. The Inverse Distance Weighting Method

Inverse Distance Weighting (IDW) is a spatial interpolation method based on the idea that near points must have similar values. Let $p_i \in R^n$ be the locations whose values z_i are known, for $i = 1, \ldots, N$ and, let $q_j \in R^n$ be the query locations, for $j = 1, \ldots, M$. As described by Shepard [73], each value q_j can be interpolated by using the values in $\Omega_j = \{p_i : d(p_i, q_j) < R\}$ (i.e., the values within a fixed *search radius* R), as follows:

$$z_j^* = \frac{\sum_{i=1}^{N} \lambda_{ji} z_i}{\sum_{i=1}^{N} \lambda_{ji}} \tag{1}$$

where λ_{ji} is a weighted average, computed using the Euclidean distance,

$$\lambda_{ji} = \frac{1}{dist(p_i, q_j)^\alpha} \tag{2}$$

Here, we use a matrix-vector formulation to deal with the IDW problem. Starting with a matrix Λ of size $M \times N$, in which each element is the weight average λ_{ji}, as in (1), and denoting by z the vector of the known values and with z^* the vector containing the unknown values, we can use the following algebraic operation to obtain the solution for the IDW problem:

$$z^* = \Lambda z \tag{3}$$

4.2. The Kriging Method

While IDW is the most used interpolation method, it is not the most accurate, because it is based on a generic approach that is not always suitable for a specific problem. For this reason, we also considered a Kriging model in order to compare the obtained solutions.

Kriging is a method of spatial interpolation belonging to the family of stochastic methods. It represents the link between neighboring points with a geostatistical approach (using the covariance concept) [46]. There are four major Kriging techniques: *Simple Kriging*, *Ordinary Kriging*, *Universal*

ACCEPTED MANUSCRIPT

Kriging, and *Co-Kriging*. The first three are used in the case of univariate geostatistics, the last in the case of multivariate geostatistics. We considered the ordinary Kriging model, which computes the prediction values as a weighted nonlinear combination of data values and uses different types of models (linear, spherical, exponential, gaussian, rational) to define the covariance function. Each model is related to the nature of the data to be analyzed. Here we choose the Gaussian formulation, as this best suits our data. This approach can be used on a dataset whose coordinates belong to a fixed range of analysis.

The mathematical formulation for the ordinary Kriging method (the best linear unbiased estimator) is based on the build of a covariance matrix C, of size $M \times N$, whose elements are:

$$c_{i,j} = C^0 + C^1 \left[1 - \exp(-3dist(p_i, q_j)/a)\right], \tag{4}$$

where: $dist(p_i, q_j) = \|p_i - q_j\|$ again denotes the Euclidean distance between the query points q_j and known points p_i; C^0 is a constant that is used when the phenomenon of *Nugget Effect* occurs, i.e. to manage the initial discontinuities; C^1 is also called *sill*, or threshold, and handles problems related to numerical representation; and a is the range value (the the maximum distance). These values act as a scale factor and they are empirically chosen according to the problem under examination.

In a similar way, a variance-covariance matrix $\hat{C}$ of size $N \times N$, is built. For this matrix the elements $\hat{c}_{i,j}$ are computed as in (4), but they depend only on the distances between the known data.

Using the previous defined matrices, the values z_j^*, corresponding to the query locations q_j, can be obtained, firstly solving a linear system of equations $\hat{C}q = z$ and then computing a matrix-vector product $z^* = Cq$. The ordinary Kriging method can be expressed more compactly as a matrix-vector product:

$$z^* = C\hat{C}^{-1}z, \tag{5}$$

Note that the computational complexity of the overall Kriging algorithm is $\mathcal{O}(N^3 + N \cdot M)$, while for IDW it is only $\mathcal{O}(N \cdot M)$.

4.3. GPU-parallel approaches

We implemented GPU-parallel algorithms for both the IDW and Kriging methods in order to compare their accuracy and efficiency. In our implementations, `threads` and `blocks` are synchronized to store dataset points

ACCEPTED MANUSCRIPT

into the `shared memory` before the interpolation phase. Moreover, both algorithms use the CUDA libraries cuBLAS and CUSP to perform the basic linear algebra operations.

G-IDW provides a parallel implementation of IDW interpolation. It computes the matrix Λ by a full parallel strategy: the i-th thread computes the elements (weights) of the i-th row and each element of this row is divided by the sum of the weights in order to obtain the weighted mean. Finally, we use an ad hoc cuBLAS library routine to multiply Λ by the vector z, containing the known values.

G-KRIGING implements a parallel version of the Kriging method. It determines the matrices $\hat{C}$ and C in a similar way to the G-IDW algorithm: the i-th thread computes the elements of the i-th row. The vector q is the solution to the linear system $\hat{C}q = z$, computing with a CUSP library routine. As in G-IDW, the matrix C is multiplied by the solution vector q, using cuBLAS to manage the matrix-vector product.

Large datasets are stored into shared memory in different chunks and the data transfer, Host-to-Device and vice versa, are minimized at the beginning and at the end of the program code.

5. A Reputation-Based Approach to Adjusting Vessel Data

Because sensors on different vessels may be imperfectly calibrated, the data that they collect may be systemically biased. We thus developed a novel approach to data adjustment based on Collaborative Reputation System methods [74] and related algorithms. Our approach collects depth data from different vessel sensors, blends all data to compute adjusted values for each vessel, and uses the adjusted values to establish offset and scale parameters estimates for each sensor.

In more detail, let s_i denote the i-th vessel sensor. We assume that each sensor may have an inherent unreliability and may provide depth measures with a certain degree of uncertainty. These sensors, due to different causes, distort a true value d, by providing a measured quantity $\hat{d}$. This kind of measurement error can be modelled as:

$$\hat{d} = d + \Delta m \tag{6}$$

where Δm represents the measurement error. The error Δm is assumed to depend on both random and systematic errors, that is:

$$\Delta m = \Delta m_{sys} + \Delta m_{ran} \tag{7}$$

ACCEPTED MANUSCRIPT

and in term of measured quantities:

$$\hat{d} = d + \Delta m_{sys} + \Delta m_{ran} \tag{8}$$

A general distribution for random noise errors needs to be specified. In this paper, we assume a normal distribution, that is:

$$\Delta m_{ran} \propto N(0, \sigma^2_{ran}), \tag{9}$$

with zero mean, and independent of (or uncorrelated with) the explanatory variable d. Systematic errors are simulated by a linear dependence between true values and measured quantities, so that for each sensor s_i we have that:

$$\hat{d} = s_i(d) + \Delta m_{ran} = a_i + b_i \cdot d + \Delta m_{ran} \tag{10}$$

More properly, parameters a_i and b_i represent the offset and the scale sensor errors, respectively.

To estimate these parameters, classical linear regression could be used directly. However, that would require that we use actual depth values as explanatory variables, and such data are not always available. Thus, we use an iterative filtering approach [75, 76] to remove random errors and thus recover accurate depth values. In general, an iterative filtering method allows us to assign reputation (i.e., a measure of reliability) to a set of users who assign evaluations to a set of objects. At the same time, the algorithm provides reputation values also to each object as a weighted sum of the evaluation that it has received from all users. In this way, from a statistical point of view, discordant evaluations have only modest impact, and objects receive a more correct average evaluation [77]. In our scheme, to tap the full potential of iterative filtering, we consider:

- location points $\xi_j = (x_j, y_j)$ as objects (x_j and y_j could be thought of as latitude and longitude coordinates);
- sensors $\{s_1, s_2, \ldots, s_N\}$ as the N users;
- depth values $\hat{d_{ij}} = s_i(d_j) + \Delta m_{ran}$, provided by sensor s_i at location ξ_j, as evaluations.

We then use the reputation values of the objects, i.e., the depth values adjusted by using the filtering averages, as explanatory values to better estimate parameters a_i and b_i of each sensor s_i. Since new depth data (due

ACCEPTED MANUSCRIPT

to the presence of new vessel sensors, or new evaluations provided by *old* sensors) become available day by day, our system can be applied repeatedly over time. To this end, a more general procedure, named a Collaborative Reputation System (CRS), is preferred. Collaborative Reputation Systems have been introduced to involve the time variable, which allows for the simulation of a system in which new users and new objects can be added over time. In our context, we think of each day as a time step, and the CRS is a multi-step procedure in which each single step is a simple iterative filtering procedure. In the proposed approach, we use an *iterative filtering-based CRS with memory*, that is a CRS in which the trustworthiness of evaluations at time $d+1$ depends on the information at the previous time d. (Galletti et al. [78] discuss this procedure in depth.) In more detail, depth data from reliable sensors are used primarily to influence mean values, so that new data, and new sensors, are forced to be coherent with previously collected known and reliable data [79].

In summary, we use crowdsourced data for three purposes: to mitigate the effect of random errors; to furnish to each sensor better estimates of scale and offset parameters; and to enhance the accuracy in depth measures. Figure 5 summarizes this computing block.

5.1. Simulation

In order to evaluate the Collaborative Reputation System with Memory we developed a *Single Beam Echo-Sounder Simulator* (SBESS). The main purpose of this software component is to provide real world high quality depth data by using a dataset available in the literature at 1m resolution [80]. In these simulation experiments, we define a simulated boat to be equipped with a GPS and an echo-sounder for depth measurement. In our simplified model, a depth measurement is defined as follows:

$$depth_{lon,lat} = a + b * gauss(true_{lon,lat}, s) \tag{11}$$

where:

lon, lat	= longitude and latitude of the fix
true	= real depth
a	= sensor offset in term of vertical distance from the vessel water line
b	= scale error due to the echo-sounder internal behaviour
s	= standard deviation of a Gaussian random function
depth	= sampled bathymetry.

ACCEPTED MANUSCRIPT

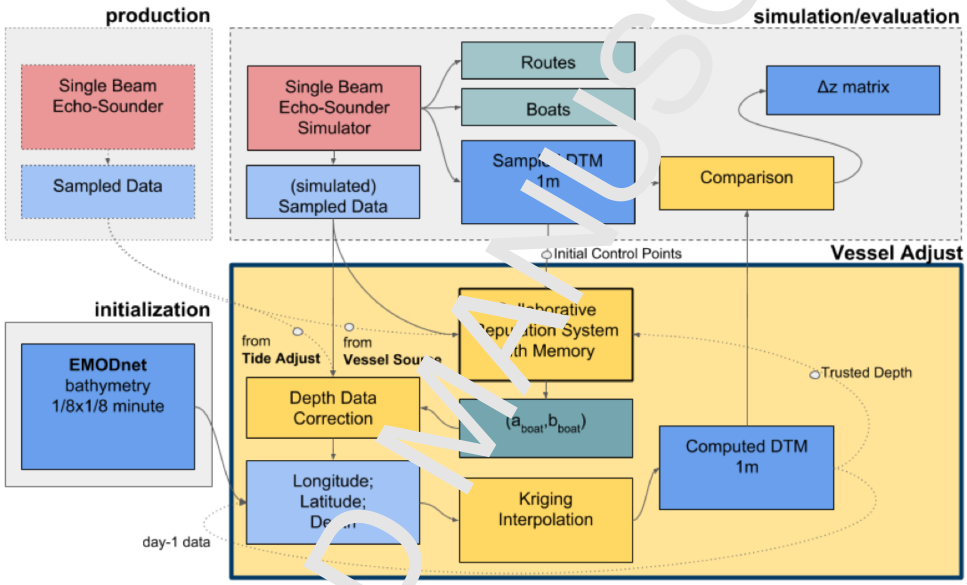

Figure 5: The proposed vessel adjustment scheme (day d) for automatic depth sensor calibration using collaborative reputation with memory. The simulation/evaluation block shows how the simulated sampled data is generated by defining boats and routes using the Single Beam Echo-Sounder Simulator (BESS) fed with real world high resolution (1m) bathymetry data. This dataset is used as sea truth in order to generate difference matrices with sampled and the corrected data. The Vessel Adjust block is used for both simulation and production. The main component is the Collaborative Reputation System with Memory used to compute the a and b parameters for each vessel depth transducer. Some control points are extracted from the sampled high resolution dataset, while in production the EMODnet bathymetry database is used for initialization. When this methodology is used in production, the *day-1* data is used as trusted depth at each iteration on daily based.

ACCEPTED MANUSCRIPT

ACCEPTED MANUSCRIPT

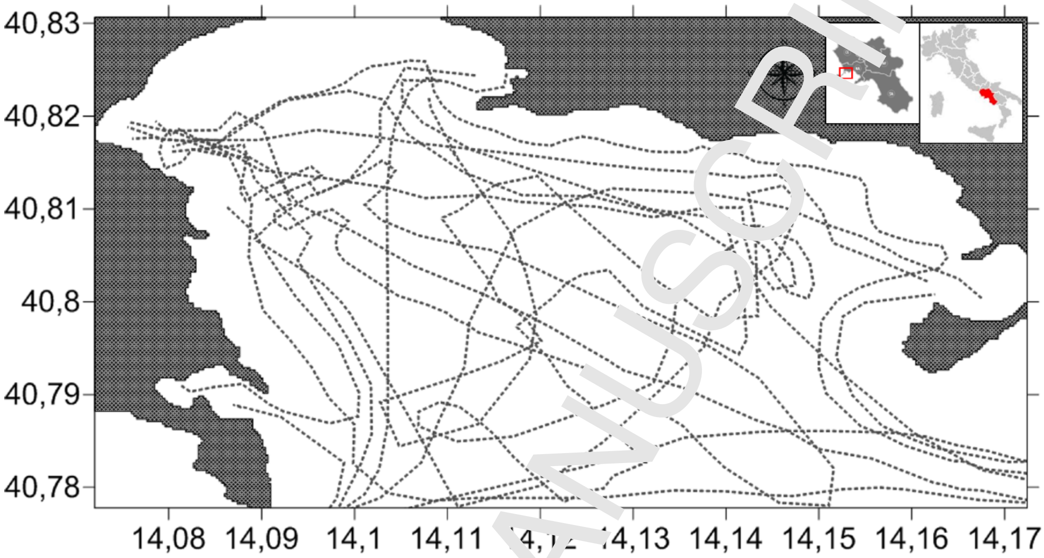

Figure 6: The 16 routes (dotted lines) used to simulate real bathymetry data sampling by vessels echo-sounder.

The simulator produces sampled data along predefined routes by using the depth dataset as sea truth. The geographic coordinates of each fix belonging to a route are randomly affected by a Gaussian error in order to imitate the GPS 2D position error.

5.2. Evaluation

We performed our evaluation of the Collaborative Reputation System with Memory in the area of the Bay of Pozzuoli in Italy, for which a high quality dataset acquired using professional equipment and scientifically validated is available. We defined 16 routes, named $route_{01}$ ... $route_{16}$, (see Figure 6) for common vessel paths. We also defined a set of 16 vessels, each characterized by different depth transducer parameters (a, b, s), as shown in Table 1. The evaluation then proceeded as follows:

1. The SBESS produced data using 16 vessels sailing along 16 routes for a total of about 400 000 samples. In the context of the simulation and evaluation we assume those data are not affected by the tide offset.

2. From the high resolution DTM 1m dataset some control points are extracted for the reputation system initialization;

ACCEPTED MANUSCRIPT

Table 1: For each simulated vessel, the depth transducer parameters defined to generate the sampled data (a, b, s) and the ones calculated by the Collaborative Reputation System with Memory (a',b') used for measurement correction.

Vessel ID	Defined			Calculated		Vessel ID	Defined			Calculated	
	a	b	s	a'	b'		a	b	s	a'	b'
01	-0.250	1.005	0	-0.2570	1.0049	09	-0.250	1	0	-0.2562	0.9999
02	-0.500	1.015	0.100	-0.5047	1.0150	10	-0.500	1	0.100	-0.5201	0.9997
03	-0.750	0.990	0.100	-0.7512	0.9900	11	-0.750	1	0.100	-0.7503	1.0000
04	-1	0.995	1	-1.0160	0.9948	12	-1	1	1	-0.9933	0.9999
05	0	1.005	0	0.0017	1.0050	13	-0.250	1.005	0	-0.2454	1.0051
06	0	1.015	0.100	-0.0047	1.0150	14	-0.500	1.015	0	-0.4952	1.0151
07	0	0.990	0.100	-0.0051	0.9900	15	-0.750	0.990	0	-0.7530	0.9900
08	0	0.995	1	0.1113	0.9961	16	-1.000	0.995	0	-1.0078	0.9949

3. The Collaborative Reputation System with Memory provides estimated depth values (as reputation values) which are used to compute a set of a and b parameters, one for each simulated boat.

4. A depth data is performed using the a and b parameters.

5. The corrected sample values are incorporated in the initial low resolution depth dataset [81].

6. The initialization dataset enriched by simulated sampled points is interpolated on a grid with 1x1m spatial resolution.

7. Finally, the newly produced dataset is compared with the high resolution DTM 1x1m considered as sea truth.

Figure 7 shows how the results of the Collaborative Reputation System with Memory in representing how the interpolated data change as new sampled values are added to the dataset. The columns a' and b' of the Table 1 represents the depth transducer parameters computed by using the proposed component, while in Figure 8 we show the effect of the reputation system on the Vessel Adjust component.

In production, the behaviour of the Collaborative Reputation System with Memory is pretty similar: *i)* The sampled data is corrected by the Tide Adjust component removing the tide related offset. *ii)* The initial vessels' depth transducer trustness is provided by the Vessel Source component. *iii)* The computed DTM 1m is used as initial dataset at the day $d + 1$.

ACCEPTED MANUSCRIPT

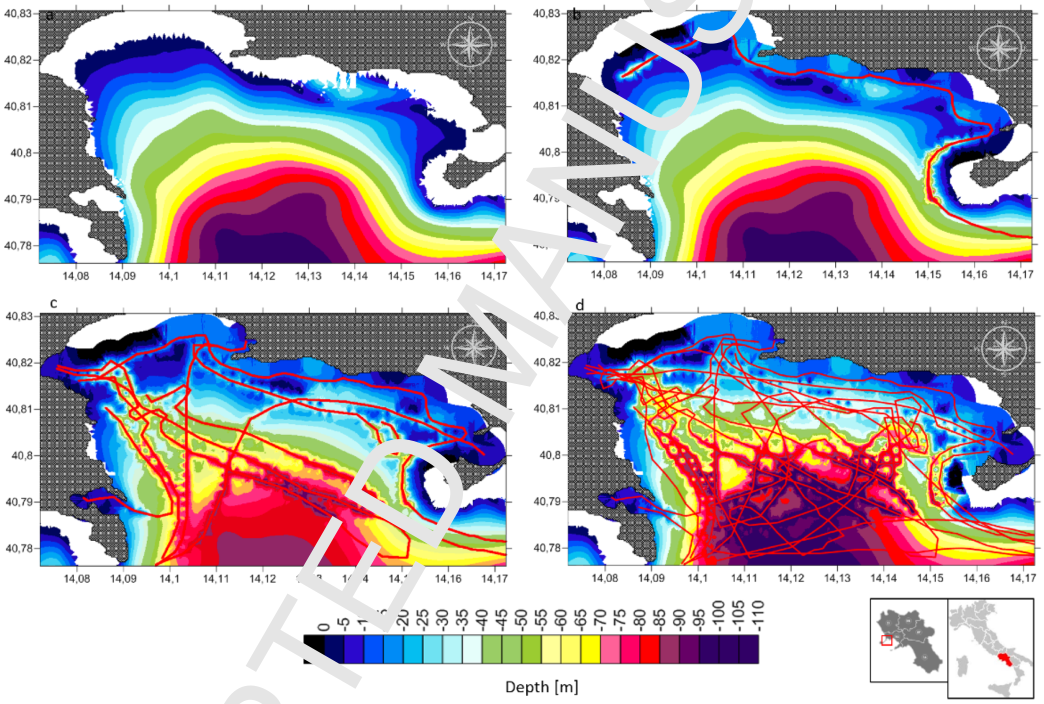

Figure 7: We use the Bay of Pozzuoli area for our evaluation due to the availability of a detailed bathymetry dataset at 1m resolution. Here we show the initial EMODnet dataset (1/8 minute resolution) interpolated on the final 1m resolution grid (a) enriched by data form 16 boats sailing $route_{12}$ (b), $routes_{1,3,5,8,9,10,12,16}$, and all routes (d).

ACCEPTED MANUSCRIPT

ACCEPTED MANUSCRIPT

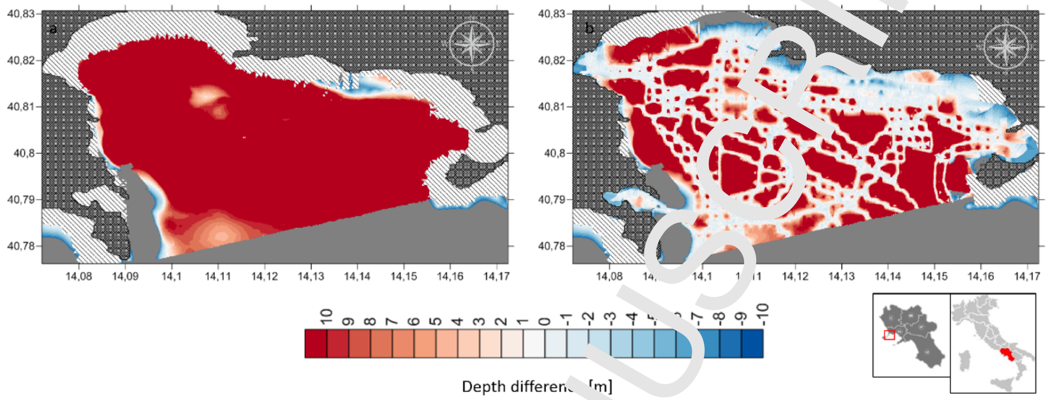

Figure 8: The depth difference in meters between the sea truth and the initial EMODnet dataset (a) and the dataset computed correcting vessel depth data using the Collaborative Reputation System with Memory presented in this work (b).

6. Data types, Tools, and Workflow

The application's main data source is the NoSQL database containing the crowdsourced data as SignalK updates in JSON form. SignalK is a modern and open data format for marine use, built on standard web technologies including JSON, WebSockets, and HTTP, providing a method for sharing information independently of the underlying communications protocol (e.g., NMEA0183, NMEA2000, SeaTalk, I2C, 1-Wire, ZigBee) in a way that is friendly to WiFi, cellphones, tablets, and Internet. SignalK defines two data formats, full and delta, for representing and transmitting data. An additional sparse format can be used to communicate just parts of the full tree. Values and attributes are stored in key/value form.

The following SignalK keys are used to refer to vessel position and depth, with `$uuid` being an unique identification string for the vessel:

```
vessels.$uuid.navigation.position.latitude
vessels.$uuid.navigation.position.longitude
vessels.$uuid.environment.depthBelowTransducer
vessels.$uuid.environment.depthBelowTransducer.timeStamp
```

The first two keys represent the vessel position, the third the seafloor depth, while the fourth encodes an attribute of a measurement, which is, in

ACCEPTED MANUSCRIPT

this case, the sampling time: the value associated with the key ending with

`depthBelowTransducer.timeStamp`

contains the date and the time of the sampled value (a similar timestamp is associated with position values). In the application prototype presented in this paper, we consider only the depth, but in a production scenario any measured value could be used to create consistent scientific datasets.

The first application step is data selection. All depth measurements not already processed in preceding runs are selected. The `$uuid` and *timestamp* associated with each depth point provide its geographic location. We validate that the time difference between the position and depth timestamp is less than one second, in order to avoid misplaced depth positions. The new depth points are then ready for the application of corrections, as follows:

- **Tide adjustment**: We use the timestamp and position to perform a tide adjustment based on a prediction model involving tide calculations and atmosphere pressure forecasts [82].
- **Vessel adjustment**: Instruments on boats report the depth below the transducer, which is typically located below the water level. The precise evaluation of each vessel's depth instrument calibration is uncertain or unfeasible in the context of crowdsourced data. In order to avoid requiring the boat owner to determine the depth of the transducer, the influence of waterline variation due to boat setup, the implicit instrument scale error, and other biases affecting the measurement, we developed an automatic approach based on collaborative reputation, as described in Section 5. In general, this tool detects and/or compensates for faulty data [83]. It uses depth data acquired by all boats participating in the crowdsourcing process, even if docked, plus in addition quality proved public datasets, to evaluate the calibration parameters (transducer offset and scale error) and apply the needed corrections to the newly acquired data.

We partition the dataset with corrected depth values by geographical areas, represented as *diles*, to yield data subsets with varying numbers of points. This partitioning allows us to perform the interpolation on restricted geographical domains. The main application process is the interpolation. The executable implementing the CUDA-enabled IDW algorithm is run on multiple virtual machine instances managed by the Job Runner sharing one

ACCEPTED MANUSCRIPT

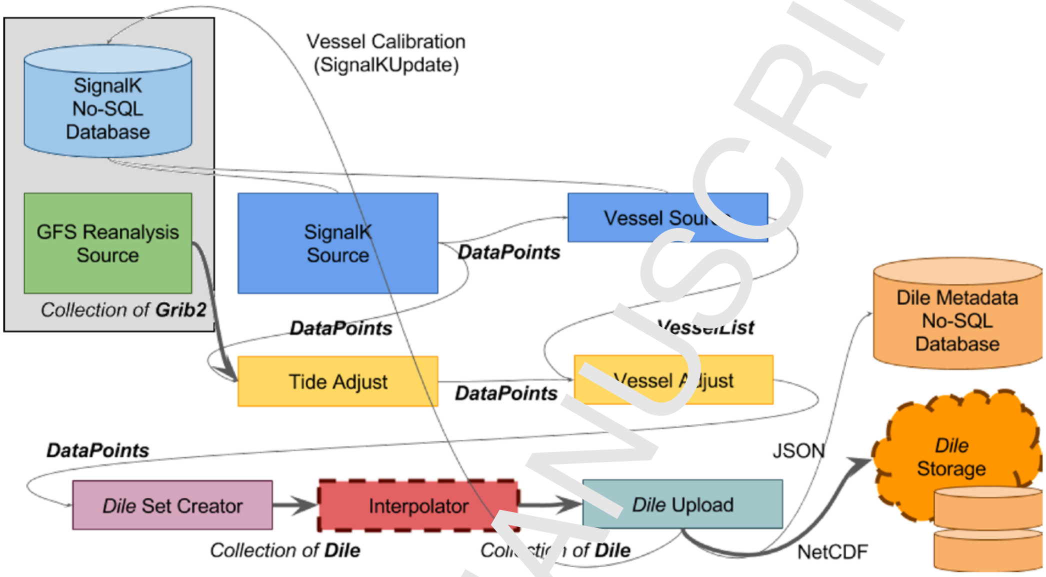

Figure 9: Our application workflow. In the gray box are the data source tools. Thick connector lines are collections of datasets. The Interpolator tool manages interactions with the cloud back-end, leveraging both regular and GPGPU-enabled instances in order to make the computational costs affordable. NoSQL databases are managed in an offline fashion with respect to the workflow. The upload of *diles* to their final storage implementation depends on the technology used (file system, S3 buckets, Globus endpoints).

or more CUDA-enabled machines as described in Section 4. The application result is a new set of updated data tiles with different zoom levels (Figure 9).

We enriched FACE-IT Galaxy with new data types based on the EnhancedJSON data type that we had implemented previously, as follows:

- **SignalKDocument**: A full SignalK document.
- **SignalKUpdate**. An updated SignalK document.
- **Dile**. A *dile*, identified by its URI. This data type is implemented as a composite datatype with a NetCDF optional component. If this component is present, the dataset represents the *dile* and its content.
- **DataPoints**. A JSON list of environmental data points, each characterized by a timeStamp, position, and one or more data values labelled using SignalK keys.

ACCEPTED MANUSCRIPT

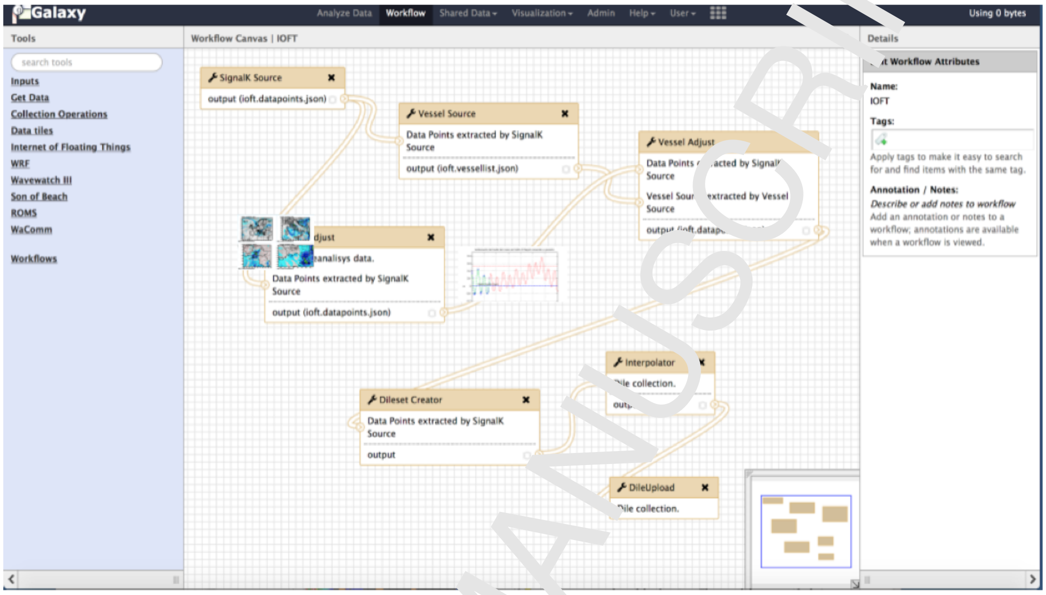

Figure 10: The FACE-IT Galaxy workflow implementing the proposed Internet of Floating Things crowdsourced data seafloor interpolation approach.

- **VesselList**. A position collection of vessels, each identified by is SignalK `$uuid`. The vessel data are represented using SignalK keys.

In order to implement the described application, we developed seven new FACE-IT Galaxy tools (see Figure 10), as follows:

SignalK Source uses a NoSQL query to select a set of depth data points from the back-end SignalK logging database. The underlying software component manages date/time and position. This tool accepts as an input a dataset with the points already processed in the previous iteration and a time span of selectable points. By default, points are retrieved for the period between the latest application run and the current time.

Vessel Source extracts the list of the vessels that produced the selected points from the source points dataset.

Tide Adjust performs a tidal adjustment on each depth point, considering the date/time from `timestamp` and the position. A collection of GRIB2-format files containing sea level pressure data can optionally be provided for weather corrections. The tool produces a dataset of corrected depth points.

ACCEPTED MANUSCRIPT

In our prototype, we do not make use of direct tide measurements, but we will consider this more accurate approach in the future.

Vessel Adjust performs the needed calculations to mitigate the biases characteristic of each vessel. It accepts as input a dataset with depth points and interacts with the back-end SignalK logging database, which is updated at the end of each application iteration. The tool produces as output a dataset of corrected depth points.

Dile set creator accepts as input the dataset containing the points to be processed and produces a collection of datasets representing the *diles* interested by the data interpolation. Each *dile* is identified by its URI and contains the list of the assigned depth points.

Interpolator is the main application workflow tool. It applies the interpolation method to each collection of the *dile* dataset, spawning the processes on different instances as managed by the Job Runner. In order to achieve the best interpolating performance in terms of geophysical accuracy, the software component wrapped by this tool selects a boundary *dile* set around each selected *dile*. This way, we reduce the number of points to be processed. The output of the tool is a collection of updated *diles*.

Dile upload is the final application step. It accepts as input a collection of *dile* datasets, which it uploads to a publicly accessible storage location, while also updating the NoSQL database metadata. It also manages the backup of any existing *dile* values, in order to record the history of depth variation of each point in each *dile*, as required for evaluating vessel adjustments.

7. Evaluation

We analyzed the behaviour of both individual components, where feasible, and the overall workflow.

7.1. GPGPU Virtualization

We evaluated the performance of the CUDA-enabled G-IDW and G-KRIGING algorithms in different GVirtuS back-end/front-end configurations. In particular, we measured their performance algorithms while varying the known sample points and fixing the number of query locations, in a setup where the GVirtuS back-end is deployed on an AWS *p2.large* machine instance equipped with a NVIDIA K80 CUDA-enabled device acting

ACCEPTED MANUSCRIPT

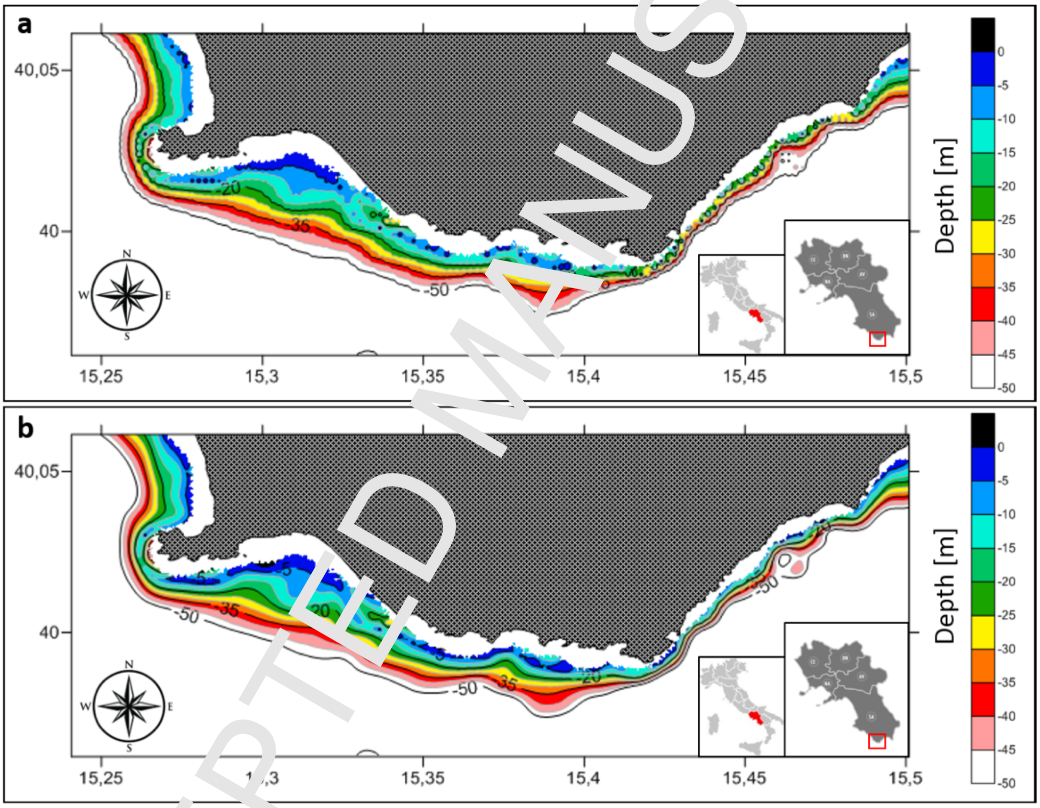

Figure 11: The EMODnet bathymetry dataset interpolated using (a) the G-IDW algorithm and (b) the G-KRIGING algorithm. The original dataset has been downscaled using a grid spacing of about 25m (search radius equal to 0.005°).

ACCEPTED MANUSCRIPT

ACCEPTED MANUSCRIPT

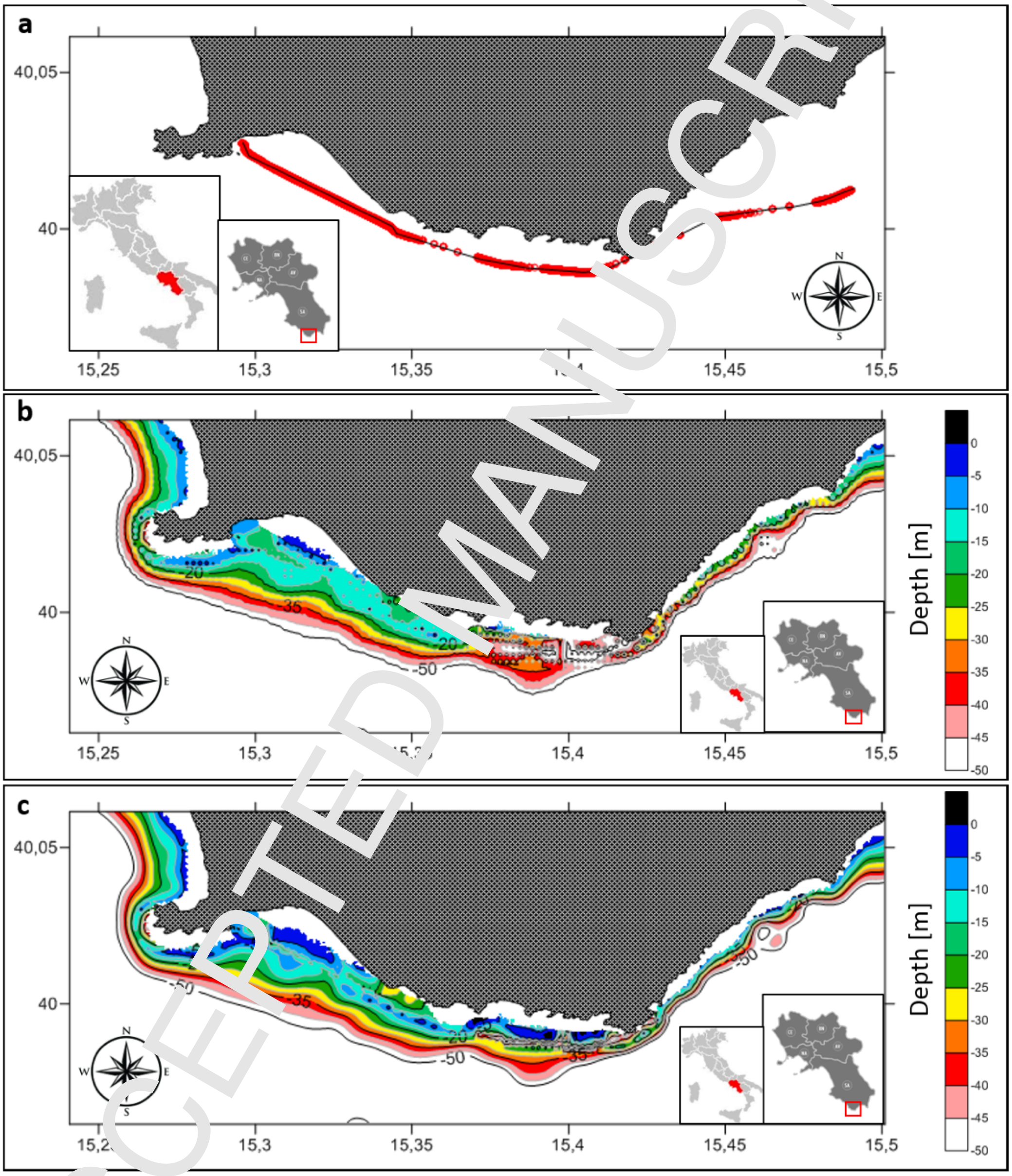

Figure 12: The EMODnet bathymetry dataset improved with the crowdsourced bathymetry data recorded along the red track showed in (a). The dataset has been downscaled using a grid spacing of about 25m (search radius equal to 0.005°) and the algorithms (b) G-IDW and (c) G-KRIGING, respectively.

ACCEPTED MANUSCRIPT

Table 2: July 2017, single vessel data acquisition.

	MON	TUE	WED	THU	FRI	SAT	SUN
July						1	2
Hours						6	6.2
Data (Mb)						15	15.1
Points						10,032	11,275
	3	4	5	6	7	8	9
Hours				1.25		3.25	4
Data (Mb)				3.3		7.7	10.8
Points				19,96		5,986	7,288
	10	11	12	13	14	15	16
Hours					1.5	2.25	3
Data (Mb)					3.8	5.1	7.8
Points					3,211	4,489	5,589
	17	18	19	20	21	22	23
Hours			2.25	1.25	6.7	2.75	5.7
Data (Mb)			5.75	2.9	17.2	6.5	14.5
Points			4,211	2,078	14,312	5,869	12,321
	24	25	26	27	28	29	30
Hours							3.5
Data (Mb)							9.2
Points							7218

ACCEPTED MANUSCRIPT

ACCEPTED MANUSCRIPT

Table 3: August and September 2017, single vessel data acquisition.

	MON	TUE	WED	THU	FRI	SAT	SUN
August	31	1	2	3	4	5	6
Hours							
Data (Mb)							
Points							
	7	8	9	10	11	12	13
Hours	4	7.3	6.5	4.7	4.2	3.9	4.2
Data (Mb)	11.1	18.2	16.5	10.9	11.2	10.1	9.9
Points	8,243	13,298	11,923	11,025	9,122	8,215	8,521
	14	15	16	17	18	19	20
Hours	5.25	6.7	6.4	5.2	3.2	4.5	2.9
Data (Mb)	12.7	[illegible]	16.9	13.1	7.7	11.7	6.9
Points	11,266	[illegible]	11,984	10,533	5,977	8,911	12,973
	21	22	23	24	25	26	27
Hours	6.7	7.2	8.4	7.9	6.45	6.2	3.9
Data (Mb)	15.9	18.2	22.9	14.9	10.8	16.6	10.9
Points	11,920	13,997	17,010	14,220	11,820	11,286	7,109
September	28	29	30	31	1	2	3
Hours	3.9	6.25	5.7				
Data (Mb)	10.1	17.8	13.9				
Points	7,919	13,002	11,251				

ACCEPTED MANUSCRIPT

ACCEPTED MANUSCRIPT

as GPGPU accelerator, while the FACE-IT Galaxy working node is an AWS micro instance where CUDA 8.0 SDK is available as the GVirtuS front-end. Both machines are run on the *us-west-2b* zone.

7.2. Data Acquisition

We performed an acquisition experiment using a single sailing boat during the period June to September 2017 along the red track in Figure 12a. The boat used for the experiments was equipped with multiple marine electronic instruments linked to the DYNAMO single board computer using an ad-hoc interface in order to harmonize several data network backbones. The crowdsourced bathymetry data were stored on board and sent to the cloud services using the protocol described earlier.

Tables 2 and 3 represent the daily collected data and the number of depth points. The actual data, even if relative to just a single vessel, demonstrate how the produced data and the number of valid data points vary. It ranges between zero in regular weekdays, to a few thousand new depth points daily during the season peek time. If we consider many vessels geographically distributed over the coastal areas for the whole year, the amount of data size magnitude rises to gigabytes and more.

7.3. Interpolation

We use data collected during August 13th-20th 2017 to illustrate the use of the application workflow and to compare and contrast the behaviour of the different Interpolator tools. We consider a cold start case, in which the known points are given by the sum of the daily sampled points from the single available vessel and the EMODnet bathymetry dataset[84].

We show first in Figures 11a,b, show the EMODnet dataset downscaled using a grid spacing of about 25m, obtained with G-IDW and G-KRIGING, respectively. The two interpolation methods differ at fine scales. Next, we show in Figures 12b,c, the results when the crowdsourced bathymetry data are added to the native EMODnet data. We see, as we might expect, significant improvements along the areas of the vessel's track (as shown in Figure 12a), especially along coastal regons. We also observe that the interpolation method used has considerable impact on the quality of the result. G-KRIGING, while computationally more demanding, produces better results than G-IDW.

Next, we compare the computational performance of the two methods. Table 4 shows the time required to run the Interpolator tool for each day

ACCEPTED MANUSCRIPT

in the sample period, when using an AWS micro instance as the FACE-IT Galaxy computing node and an AWS *p2.large* instance as the GPGPU accelerator. We do not consider the tool launching time because this is affected by AWS instance performance. The first row details the kind of navigation. The data acquired when the vessel is docked are used for sensor calibration. The reported number of points represent also the valid depth data (i.e., no out of range measures). The *diles* and the buffer lines represents the number of *diles* where new data points are acquired and the buffer needed for interpolation, respectively. Considering the defined grid spacing, the known points have been calculated adding the number of sample points of the EMODnet dataset included in the selected dile at zoom level nine (about $1/2^9$ degree) to the number of sample points. The query locations have been evaluated considering both *diles* and buffers grids. We see that Kriging runs twice as fast as IDW, thanks to its use of GPUs via GVirtuS, albeit at higher cost due to its use of the GPGPU accelerator.

8. Conclusions and Future Work

We have described experiments with a complex infrastructure comprising: *i)* DYNAMO for data logging on leisure vessels; *ii)* a reliable IoT data transfer framework; *iii)* a cloud hosted data storage and data adjustment based on reputation; *iv)* an interpolation software component provided by GPGPU-enabled methods supporting GPGPU virtualization; and *v)* a FACE-IT Galaxy cloud instance and job runner for managing computation.

We have used this infrastructure to realize a novel workflow-based application prototype that allows data to be collected via a crowdsourcing process from leisure vessels, thus realizing an 'Internet of Floating Things.' The data, stored as SignalK updates in JSON format, are transferred from vessels to the cloud using the novel data transfer protocol that we developed by using the proposed framework, which is designed to work in such harsh environments.

Once data are in the cloud, they are stored in a NoSQL database. The FACE-IT Galaxy workflow is executed periodically, for example once per day. Each vessel's depth sensor is calibrated automatically via a novel reputation approach in which each datum is compared with other measurements to evaluate the offset and the scale calibration parameters of each instrument. The application computing core is represented by the interpolator tool, which wraps a CUDA-enabled executable that implements a customized version of the G-IDW and G-KRIGING algorithms (Figure 11 and Figure 12). We use

ACCEPTED MANUSCRIPT

Table 4: Interpolator tool performance for the Season 2017 week with the most data. For each day, we give the *Navigation* type (D – Docked; E – sailing using the Engine; S – under Sail); *Points*, the number of recorded measurements; *Diles*, the number of diles considered for the interpolation (Figure 4); *Buffer*, the number of diles selected as a buffer around the sailed track; *Known*, the number of valid depth measurements, and *Queries*, the number of interpolated points (missing values are due to computational issues). The bottom two sections give the computational costs of the two interpolator tools, in seconds.

	MON	TUE	WED	THU	FRI	SAT	SUN
August	14	15	16	17	18	19	20
Navigation	S	E	S	E	E	D	D
Points	11,266	13,865	11,984	10,533	5,977	8,911	12,973
Diles	7	9	6	6	3	9	9
Buffer	18	30	18	32	15	10	10
Known (*10^3)	75.1	77.9	77.3	75.0	71.1	73.6	70.5
Queries (*10^3)	1,620	2,527	1,555	2,462	1,166	648	648
CPU time (s)							
IDW	682.9	1,120.7	672.8	1,034.2	420.1	272.5	235.4
Kriging	0	0	0	0	0	0	0
GVirtuS time (s)							
IDW	.03	.05	.03	.05	.02	.01	.01
Kriging	322.4	521.7	318.6	489.4	219.7	126.4	121.1

ACCEPTED MANUSCRIPT

GVirtuS to run this software on regular EC2 instances that offload CUDA computations to a reduced number of CUDA-enabled EC2 instances. We tested the workflow's most computation demanding tool, the Interpolator, with a simulated production cycle on real data demonstrating that the approach of using virtual GPGPU on the cloud is feasible.

We conclude from these experiments that our goal of gathering environmental data, performing the needed processing and homogenization, and then supporting experiments of computational environmental scientist can be realized at large scales and at modest costs. We are currently enhancing this prototype and evaluating the relationship between scalability and economic convenience. The system will soon be deployed and tested on additional vessels, with DYNAMO as the on-board data technology and FairWind Home as the main smart boating GUI.

Our immediate goal is to improve the overall stability of the system and perform more detailed and comprehensive performance evaluations results, especially comparing and contrasting different interpolation algorithms and related settings. The automatic sensor calibration based on reputation has to be deeply tested with a consistent number of DYNAMO-equipped vessels in the fleet. At the time of writing, we have tested a first set of performance-critical components, namely the Internet of Things data transfer protocol, GPGPU virtualization and remoting, and interpolation algorithm. Application workflow scaling will also become important as the data to be managed increase with more vessels.

Refining the bathymetry data processing algorithms is our mid-term goal. We need to introduce better methods for geographic data anonymization [85]. We also want to eliminate our current tight dependency on AWS by making the data processing component of our application cloud independent [86], so that it can run on OpenStack public, private, and hybrid clouds.

Long-term goals include a production system capable of supporting many vessels and updating large datasets routinely, and providing the resulting marine open data to the public [87]. We also plan to extend beyond bathymetry to other environmental parameters directly sampled by leisure vessels (wind, weather, air temperature) or derived by further computations (such as surface currents and sea waves). Finally, we also want to investigate issues of data quality, uncertainty, and coverage.

ACCEPTED MANUSCRIPT

Acknowledgments

This work has been supported in part by U.S. National Science Foundation awards 0951576 and 1331782; and by the University of Napoli Parthenope, Italy (project DSTE333 "Modelling Mytilus Farming System with Enhanced Web Technologies," funded by Campania Region/Veterinary sector).

References

[1] C. E. Catlett, P. H. Beckman, R. Sankaran, K. K. Galvin, Array of Things: A scientific research instrument in the public way, in: 2nd International Workshop on Science of Smart City Operations and Platforms Engineering, ACM, 2017, pp. 26–33.

[2] D. D. Luccio, G. Benassai, G. Budillon, L. Mucerino, R. Montella, E. Pugliese Carratelli, Wave run-up prediction and observation in a micro-tidal beach, Natural Hazards and Earth System Sciences 18 (11) (2018) 2841–2857.

[3] D. Di Luccio, G. Benassai, G. Di Paola, C. M. Rosskopf, L. Mucerino, R. Montella, P. Contestabile, Monitoring and modelling coastal vulnerability and mitigation proposal for an archaeological site (kaulonia, southern italy)., Sustainability (2071-1050) 10 (6).

[4] G. Benassai, P. Aucelli, G. Budillon, M. De Stefano, D. Di Luccio, G. Di Paola, R. Montella, L. Mucerino, M. Sica, M. Pennetta, Rip current evidence by hydrodynamic simulations, bathymetric surveys and uav observation, Natural Hazards and Earth System Sciences 17 (9) (2017) 1493–1503.

[5] R. Montella, D. Di Luccio, P. Troiano, A. Riccio, A. Brizius, I. Foster, WaComM: A parallel Water quality Community Model for pollutant transport and dispersion operational predictions, in: 12th International Conference on Signal-Image Technology & Internet-Based Systems, IEEE, 2016, pp. 717–724.

[6] A. Galletti, R. Montella, L. Marcellino, A. Riccio, D. Di Luccio, A. Brizius, I. T. Foster, Numerical and implementation issues in food quality modeling for human diseases prevention, in: 10th International Conference on Health Informatics, 2017, pp. 526–534.

ACCEPTED MANUSCRIPT

[7] D. Di Luccio, A. Galletti, L. Marcellino, A. Riccio, R. Montella, A. Brizius, Some remarks about a community open source lagrangian pollutant transport and dispersion model, Procedia Computer Science 113 (2017) 490–495.

[8] B. Hackett, Ø. Breivik, C. Wettre, Forecasting the drift of objects and substances in the ocean, in: Ocean weather forecasting, Springer, 2006, pp. 507–523.

[9] B.-W. Chen, W. Ji, S. Rho, Geo-conquesting based on graph analysis for crowdsourced metatrails from mobile sensing, IEEE Communications Magazine 55 (1) (2017) 92–97.

[10] S. S. Kanhere, Participatory sensing: Crowdsourcing data from mobile smartphones in urban spaces, in: 12th International Conference on Mobile Data Management, Vol. 2, IEEE, 2011, pp. 3–6.

[11] C. Heipke, Crowdsourcing geospatial data, ISPRS Journal of Photogrammetry and Remote Sensing 65 (6) (2010) 550–557.

[12] M. Lease, On quality control and machine learning in crowdsourcing, Human Computation 11 (11).

[13] G. Zacharia, A. Moukas, P. Maes, Collaborative reputation mechanisms for electronic marketplaces, Decision support systems 29 (4) (2000) 371–388.

[14] S. Ganeriwal, L. K. Balzano, M. B. Srivastava, Reputation-based framework for high integrity sensor networks, ACM Transactions on Sensor Networks (TOSN) 4 (3) (2008) 15.

[15] K. Fall, A delay-tolerant network architecture for challenged internets, in: Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, ACM, 2003, pp. 27–34.

[16] G. Laccetti, R. Montella, C. Palmieri, V. Pelliccia, The high performance internet of things: using gvirtus to share high-end gpus with arm based cluster computing nodes, in: International Conference on Parallel Processing and Applied Mathematics, Springer, 2013, pp. 734–744.

ACCEPTED MANUSCRIPT

[17] R. Di Lauro, F. Lucarelli, R. Montella, SIaaS-sensing instrument as a service using cloud computing to turn physical instrument into ubiquitous service, in: 10th International Symposium on Parallel and Distributed Processing with Applications, IEEE, 2012, pp. 861–862.

[18] R. Montella, G. Agrillo, D. Mastrangelo, M. Menna, A Globus Toolkit 4 based instrument service for environmental data acquisition and distribution, in: 3rd International Workshop on Use of P2P, Grid and Agents for the Development of Content Networks, ACM, 2008, pp. 21–28.

[19] G. Benassai, I. Ascione, Implementation of WWIII wave model for the study of risk inundation on the coastlines of Campania, Italy, WIT Transactions on Ecology and the Environment 88.

[20] R. Montella, S. Kosta, I. Foster, DYNAMO: Distributed leisure Yacht-carried sensor-Network for Atmosphere and Marine data crOwdsourcing applications, in: International Conference of Internet of Things Design and Implementation, IEEE, 2018.

[21] R. Montella, D. Di Luccio, S. Kosta, G. Giunta, I. Foster, Performance, resilience, and security in moving data from the fog to the cloud: The dynamo transfer framework approach, in: International Conference on Internet and Distributed Computing Systems, Springer, 2018, pp. 197–208.

[22] R. Montella, A. Brizius, D. Di Luccio, C. Porter, J. Elliot, R. Madduri, D. Kelly, A. Riccio, I. Foster, Applications of the face-it portal and workflow engine for operational food quality prediction and assessment: Mussel farm monitoring in the bay of napoli, italy (2016).

[23] R. Montella, A. Brizius, D. Di Luccio, C. Porter, J. Elliot, R. Madduri, D. Kelly, A. Riccio, I. Foster, Using the FACE-IT portal and workflow engine for operational food quality prediction and assessment: An application to mussel farms monitoring in the Bay of Napoli, Italy, Future Generation Computer Systems.

[24] R. Montella, G. Giunta, G. Laccetti, M. Lapegna, C. Palmieri, C. Ferraro, V. Pelliccia, C.-H. Hong, I. Spence, D. S. Nikolopoulos, On the virtualization of CUDA based GPU remoting on ARM and X86 machines

ACCEPTED MANUSCRIPT

in the GVirtuS framework, International Journal of Parallel Programming (2017) 1–22.

[25] R. Di Lauro, F. Giannone, L. Ambrosio, R. Montella, Virtualizing general purpose GPUs for high performance cloud computing: an application to a fluid simulator, in: 10th International Symposium on Parallel and Distributed Processing with Applications, IEEE, 2012, pp. 863–864.

[26] R. Montella, L. Marcellino, A. Galletti, D. Di Luccio, S. Kosta, G. Laccetti, G. Giunta, Marine bathymetry processing through gpgpu virtualization in high performance cloud computing, Concurrency and Computation: Practice and Experience (2018) e4895.

[27] F. Alam, R. Mehmood, I. Katib, N. N. Albogami, A. Albeshri, Data fusion and iot for smart ubiquitous environments: A survey, IEEE Access 5 (2017) 9533–9554.

[28] R. Montella, S. Kosta, D. Oro, J. Vera, C. Fernández, C. Palmieri, D. Di Luccio, G. Giunta, M. Lapegna, G. Laccetti, Accelerating linux and android applications on low-power devices through remote gpgpu offloading, Concurrency and Computation: Practice and Experience 29 (24) (2017) e4286.

[29] B. Halder, Evolution of crowdsourcing: Potential data protection, privacy and security concerns under the new media age, Revista Democracia Digital e Governo Eletrônico 1 (10) (2014) 377–393.

[30] J.-B. Calewaert, P. Weaver, V. Gunn, P. Gorringe, A. Novellino, The european marine data and observation network (EMODnet): Your gateway to European marine and coastal data, in: Quantitative Monitoring of the Underwater Environment, Springer, 2016, pp. 31–46.

[31] I. Rodero Castro, M. Parashar, Architecting the cyberinfrastructure for the National Science Foundation Ocean Observatories Initiative (OOI), in: Instrumentation Viewpoint, no. 19, SARTI, 2016, pp. 99–101.

[32] F. Silla, J. Prades, S. Iserte, C. Reano, Remote GPU virtualization: Is it useful?, in: 2nd International Workshop on High-Performance Interconnection Networks in the Exascale and Big-Data Era, IEEE, 2016, pp. 41–48.

ACCEPTED MANUSCRIPT

ACCEPTED MANUSCRIPT

[33] C. Reaño, F. Silla, A performance comparison of CUDA remote GPU virtualization frameworks, in: International Conference on Cluster Computing, IEEE, 2015, pp. 488–489.

[34] C. Reaño, F. Silla, Reducing the performance gap of remote GPU virtualization with InfiniBand Connect–IB, in: Symposium on Computers and Communication, IEEE, 2016, pp. 920–925.

[35] S. Cuomo, A. Galletti, G. Giunta, L. Marcellino, Piecewise Hermite interpolation via barycentric coordinates: In memory of Prof. Carlo Ciliberto, Ricerche di Matematica 64 (2) (2015) 303–319. doi:10.1007/s11587-015-0233-0.

[36] S. Cuomo, A. Galletti, G. Giunta, L. Marcellino, A class of piecewise interpolating functions based on barycentric coordinates, Ricerche di Matematica 63 (1) (2014) 87–102. doi:10.1007/s11587-014-0214-8.

[37] S. Cuomo, A. Galletti, G. Giunta, L. Marcellino, A novel triangle-based method for scattered data interpolation, Applied Mathematical Sciences 8 (133-136) (2014) 6717–6724. doi:10.12988/ams.2014.49686.

[38] O. Falivene, L. Cabrera, R. Tolosana-Delgado, A. Sáez, Interpolation algorithm ranking using cross-validation and the role of smoothing effect. A coal zone example, Computers & Geosciences 36 (4) (2010) 512–519.

[39] X. Shi, F. Ye, Kriging interpolation over heterogeneous computer architectures and systems, GIScience & Remote Sensing 50 (2) (2013) 196–211.

[40] S. Cuomo, P. De Michele, A. Galletti, L. Marcellino, A parallel PDE-based numerical algorithm for computing the optical flow in hybrid systems, Journal of Computational Sciencedoi:10.1016/j.jocs.2017.03.011.

[41] F. G. De Ravé, F. J. Jiménez-Hornero, A. B. Ariza-Villaverde, J. Gómez-López, Using general-purpose computing on graphics processing units (GPGPU) to accelerate the ordinary kriging algorithm, Computers & Geosciences 64 (2014) 1–6.

ACCEPTED MANUSCRIPT

[42] L. Huraj, V. Siládi, J. Siláci, Design and performance evaluation of snow cover computing on GPUs, in: 14th WSEAS International Conference on Computers, 2010, pp. 674–677.

[43] A. Danner, A. Breslow, J. Baskin, D. Wilikofsky, Hybrid MPI/GPU interpolation for grid DEM construction, in: 20th International Conference on Advances in Geographic Information Systems, ACM, 2012, pp. 299–308.

[44] G. Mei, H. Tian, Impact of data layouts on the efficiency of GPU-accelerated IDW interpolation, SpringerPlus 5 (1) (2016) 104.

[45] N. Cressie, The origins of kriging, Mathematical Geology 22 (3) (1990) 239–252.

[46] E. H. Isaaks, R. M. Srivastava, An introduction to applied geostatistics, New York: Oxford University Press (1989) 561.

[47] J. P. Kleijnen, Kriging metamodeling in simulation: A review, European journal of operational research 192 (3) (2009) 707–716.

[48] M. Li, L. Dong, Research on CUDA-based kriging interpolation algorithm, in: 3rd International Conference on Computer and Network Technology, Vol. 2, 2011, pp. 56–59.

[49] F. Huang, S. Bu, J. Tao, X. Tan, OpenCL implementation of a parallel universal kriging algorithm for massive spatial data interpolation on heterogeneous systems, ISPRS International Journal of Geo-Information 5 (6) (2016) 96.

[50] I. Foster, Globus Online: Accelerating and democratizing science through cloud-based services, IEEE Internet Computing 15 (3) (2011) 70–73.

[51] R. Montella, D. Di Luccio, L. Marcellino, A. Galletti, S. Kosta, A. Brizius, I. Foster, Processing of crowd-sourced data from an internet of floating things, in: 12th Workshop on Workflows in Support of Large-Scale Science, ACM, 2017, p. 8.

[52] A. Hussein, M. Payer, A. Hosking, C. A. Vick, Impact of GC design on power and performance for Android, in: 8th International Systems and Storage Conference, ACM, 2015, p. 13.

ACCEPTED MANUSCRIPT

[53] A. P. Silva, S. Burleigh, C. M. Hirata, K. Obraczka, A survey on congestion control for delay and disruption tolerant networks, Ad Hoc Networks 25 (2015) 480–494.

[54] R. Montella, M. Ruggieri, S. Kosta, A fast, secure, reliable, and resilient data transfer framework for pervasive iot applications, in: IEEE INFOCOM 2018-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), IEEE, 2018.

[55] A. Botta, W. De Donato, V. Persico, A. Pescapé, On the integration of cloud computing and internet of things, in: International Conference on Future Internet of Things and Cloud, IEEE, 2014, pp. 23–30.

[56] D. Thangavel, X. Ma, A. Valera, H.-X. Tan, C. K.-Y. Tan, Performance evaluation of MQTT and CoAP via a common middleware, in: IEEE 9th International Conference on Intelligent Sensors, Sensor Networks and Information Processing, 2014, pp. 1–6.

[57] F. Armand, M. Gien, G. Maigne, G. Mardinian, Shared device driver model for virtualized mobile handsets, in: 1st Workshop on Virtualization in Mobile Computing, ACM, 2008, pp. 12–16.

[58] G. W. Dunlap, D. G. Lucchetti, M. A. Fetterman, P. M. Chen, Execution replay of multiprocessor virtual machines, in: 4th ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments, ACM, 2008, pp. 121–130.

[59] T. Li, V. K. Narayana, E. El-Araby, T. El-Ghazawi, GPU resource sharing and virtualization on high performance computing systems, in: International Conference on Parallel Processing, IEEE, 2011, pp. 733–742.

[60] R. Montella, G. Giunta, G. Laccetti, Virtualizing high-end GPGPUs on ARM clusters for the next generation of high performance cloud computing, Cluster computing 17 (1) (2014) 139–152.

[61] R. Montella, C. Ferraro, S. Kosta, V. Pelliccia, G. Giunta, Enabling android-based devices to high-end gpgpus, in: International Conference on Algorithms and Architectures for Parallel Processing, Springer, 2016, pp. 118–125.

[62] R. Montella, G. Giunta, G. Laccetti, M. Lapegna, C. Palmieri, C. Ferraro, V. Pelliccia, Virtualizing CUDA enabled GPGPUs on ARM clusters, in: Parallel Processing and Applied Mathematics, Springer, 2016, pp. 3–14.

[63] J. Goecks, A. Nekrutenko, J. Taylor, Galaxy: A comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences, Genome Biology 11 (8) (2010) R86.

[64] R. K. Madduri, D. Sulakhe, L. Lacinski, B. Liu, A. Rodriguez, K. Chard, U. J. Dave, I. T. Foster, Experiences building Globus Genomics: A next-generation sequencing analysis service using galaxy, globus, and amazon web services, Concurrency and Computation: Practice and Experience 26 (13) (2014) 2266–2279.

[65] Q. Pham, T. Malik, I. T. Foster, R. Di Lauro, R. Montella, SOLE: Linking research papers with science objects, in: International Provenance and Annotation Workshop, Springer, 2012, pp. 203–208.

[66] R. Montella, D. Kelly, W. Xiong, A. Brizius, J. Elliott, R. Madduri, K. Maheshwari, C. Porter, P. Vilter, M. Wilde, et al., FACE-IT: A science gateway for food security research, Concurrency and Computation: Practice and Experience 27 (16) (2015) 4423–4436.

[67] S. Quinn, M. Gahegan, A predictive model for frequently viewed tiles in a web map, Transactions in GIS 14 (2) (2010) 193–216.

[68] S. E. Battersby, M. P. Finn, E. L. Usery, K. H. Yamamoto, Implications of web Mercator and its use in online mapping, Cartographica 49 (2) (2014) 85–101.

[69] T. J. Skluzacek, K. Chard, I. Foster, Klimatic: A virtual data lake for harvesting and distribution of geospatial data, in: 1st Joint International Workshop on Parallel Data Storage and Data Intensive Scalable Computing Systems, IEEE, 2016, pp. 31–36.

[70] Y. Wang, S. Wang, Research and implementation on spatial data storage and operation based on hadoop platform, in: Geoscience and Remote Sensing (IITA-GRS), 2010 Second IITA International Conference on, Vol. 2, IEEE, 2010, pp. 275–278.

ACCEPTED MANUSCRIPT

[71] L. Marcellino, R. Montella, S. Kosta, A. Galletti, D. Di Luccio, V. Santopietro, M. Ruggieri, M. Lapegna, L. DAmore, G. Laccetti, Using gpgpu accelerated interpolation algorithms for marine bathymetry processing with on-premises and cloud based computational resources, in: International Conference on Parallel Processing and Applied Mathematics, Springer, 2017, pp. 14–24.

[72] J. Bello-Pineda, J. L. Hernández-Stefanoni, Comparing the performance of two spatial interpolation methods for creating a digital bathymetric model of the Yucatan submerged platform, Pan-American Journal of Aquatic Sciences 2 (3) (2007) 247–254.

[73] D. Shepard, A two-dimensional interpolation function for irregularly-spaced data, in: 23rd ACM National Conference, ACM, 1968, pp. 517–524.

[74] S. Cuomo, P. De Michele, F. Piccialli, A. Galletti, J. E. Jung, IoT-based collaborative reputation system for associating visitors and artworks in a cultural scenario, Expert Systems with Applications 79 (2017) 101–111.

[75] C. de Kerchove, P. V. Dooren, Iterative filtering in reputation systems, SIAM Journal on Matrix Analysis and Applications 31 (4) (2010) 1812–1834. doi:10.1137/090748196.

[76] C. de Kerchove, P. V. Dooren, Reputation systems and optimization, SIAM News 41 (2) (2008) 1–3.

[77] K. R. Malik, M. A. Habib, S. Khalid, M. Ahmad, M. Alfawair, A. Ahmad, G. Jeon, A generic methodology for geo-related data semantic annotation, Concurrency and Computation: Practice and Experience 30 (15) (2018) e4495.

[78] A. Galletti, G. Giunta, G. Schmid, A mathematical model of collaborative reputation systems, International Journal of Computer Mathematics 89 (17) (2012) 2315–2332. doi:10.1080/00207160.2012.715641.

[79] S. Cuomo, P. De Michele, F. Piccialli, A. K. Sangaiah, Reproducing dynamics related to an internet of things framework: A numerical and statistical approach, Journal of Parallel and Distributed Computing 118 (2018) 359–368.

ACCEPTED MANUSCRIPT

[80] R. Somma, S. Iuliano, F. Matano, F. Molisso, S. Passaro, M. Sacchi, C. Troise, G. De Natale, High-resolution morpho-bathymetry of pozzuoli bay, southern italy, Journal of Maps 12 (2) (2016) 222–230.

[81] A. Novellino, P. D'Angelo, G. Benedetti, G. Manzella, P. Gorringe, D. Schaap, S. Pouliquen, L. Rickards, European marine observation data networkemodnet physics, in: OCEANS 2015-Genova, IEEE, 2015, pp. 1–6.

[82] I. Iermano, M. Uttieri, E. Zambianchi, B. Buonocore, D. Cianelli, P. Falco, G. Zambardino, Integration of numerical modeling and observations for the Gulf of Naples monitoring network, in: EGU General Assembly Conference Abstracts, Vol. 14, 2012, p. 9046.

[83] B. Calder, Automatic statistical processing of multibeam echosounder data, International Hydrographic Review 4 (1) (2003) 53–68.

[84] D. Schaap, E. Moussat, Emodnet hydrography-seabed mapping-developing a higher resolution digital bathymetry for the european seas, in: EGU General Assembly Conference Abstracts, Vol. 15, 2013.

[85] H. Zang, J. Bolot, Anonymization of location data does not work: A large-scale measurement study, in: 17th Annual International Conference on Mobile Computing and Networking, ACM, 2011, pp. 145–156.

[86] D. Petcu, Consuming resources and services from multiple clouds, Journal of Grid Computing 12 (2) (2014) 321–345.

[87] R. Montella, G. Giunta, A. Riccio, Using grid computing based components in on demand environmental data delivery, in: 2nd Workshop on Use of P2P, Grid and Agents for the Development of Content Networks, ACM, 2007, pp. 81–86.

ACCEPTED MANUSCRIPT

Dr. Raffaele Montella works as assistant professor, with tenure, in Computer Science at Department of Science and Technologies, University of Naples â€œParthenopeâ€, Italy since 2005. He got his degree (MSc equivalent) in (Marine) Environmental Science at the University of Naples â€œParthenopeâ€ in 1998 defending a thesis about the â€œDevelopment of a GIS system for marine applicationsâ€ scoring with laude and an award mention to his study career. He defended his PhD thesis about â€œEnvironmental modeling and Grid Computing techniquesâ€ earning the PhD in Marine Science and Engineering at the University of Naples â€œFederico IIâ€.
The research main topics and the scientific production are focused on tools for high performance computing, such as grid, cloud and GPUs with applications in the field of computational environmental science (multidimensional big data/distributed computing for modeling and scientific workflows and science gateways) leveraging on his previous (and still ongoing) experiences in embedded/mobile/wearable/pervasive computing and internet of things. He joined the CI/RDCEP of the University of Chicago as Visiting Scholar and as Visiting Assistant Professor working on the FACE-IT project. He leads the IT infrastructure of the University of Naples â€œParthenopeâ€ Centre for Marine and Atmosphere Monitoring and Modelling (CCMMMA). He technically leads the University of Naples â€œParthenopeâ€ research units of the European Project â€œHeterogeneous secure multi-level Remote Acceleration service for low-Power Integrated systems and Devices (RAPID)â€ focusing on GVirtuS development and integration (General purpose Virtualization Service) enabling CUDA kernel execution on mobile and embedded devices. He leads the local funded project: â€œModelling mytilus farming System with Enhanced web technologies (MytiluSE)â€ focused on high performance computing based coupled simulations for mussel farms food quality prediction and assessment for human gastric disease mitigation. He leads the research prototype project â€œFairWind: smart, cloud-based, multifunctional navigation systemâ€ targeting the coastal marine data gathering as crowdsourcing for environmental protection, development and management.

file:///C|/Documents%20and%20Settings/focal03/Desktop/Users/arathy/rmontella.txt[23/11/2018 16:22:17]

ACCEPTED MANUSCRIPT

ACCEPTED MANUSCRIPT

* A FACE-IT Galaxy Globus application implementing automatic processing for Internet of Floating Things crowdsourced bathymetry data.
* Design and implementation of specialized FACE-IT Galaxy Globus components for earth science workflow application development.
* GPGPU (CUDA) accelerated bathymetry interpolation tools enabled for GPGPU virtualized environments using GVirtuS
* Collaborative reputation based data adjustment for automatic exstimation of depth instruments calibration settings
* Using a novel approach to environmental multidimensional data storage and processing based on the concept of discrete spatial resolution data tiles.

ACCEPTED MANUSCRIPT

file:///C|/Documents%20and%20Settings/focal03/Desktop/Users/arathy/highlights.txt[23/11/2018 16:22:19]