RAPID
http://rapid-project.eu
European Commission
H2020-ICT-644312

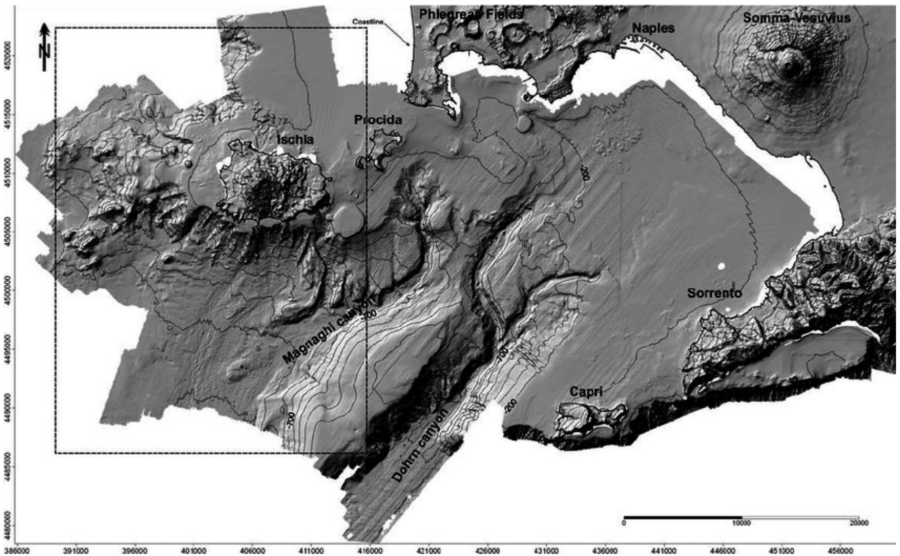
Using GPGPU accelerated interpolation algorithms for marine bathymetry processing with on-premises and cloud based computational resources

*Marcellino L.[1], **R. Montella[1,3]**, S. Kosta[4,5], A. Galletti[1], D. Di Luccio[1,3], V. Santopietro[1], M. Ruggieri[1], M. La Pegna[2], L. D'Amore[2] and G. Laccetti[2]*

[1]Department of Science and Technologies, University of Napoli Parthenope, Naples, Italy.
[2]Department Mathematics and Applications, University of Napoli Federico II, Naples, Italy.
[3]Center for Robust Decisionmaking on Climate and Energy Polity - Computation Institute, University of Chicago.
[4]CMI, Aalborg University, Copenhagen, Denmark.
[5]Sapienza University of Rome, Italy.

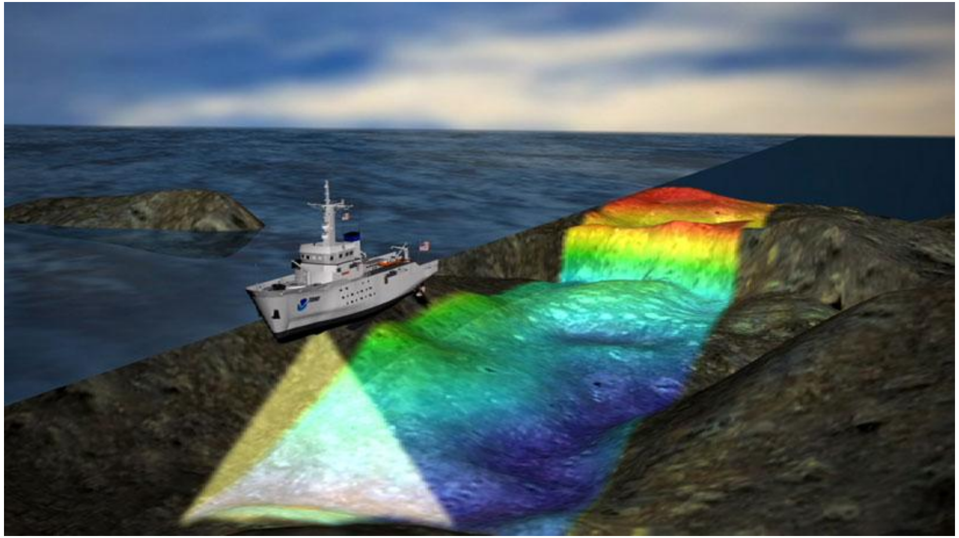
This research has been supported by the Grant Agreement number: 644312 - RAPID - H2020-ICT-2014/H2020-ICT-2014-1
"Heterogeneous Secure Multi-level Remote Acceleration Service for Low-Power Integrated Systems and Devices".

Introduction, contextualization and motivations



Bathymetry: "*the study of underwater depth of lake or ocean floors. In other words, bathymetry is the underwater equivalent to hypsometry or topography.*"

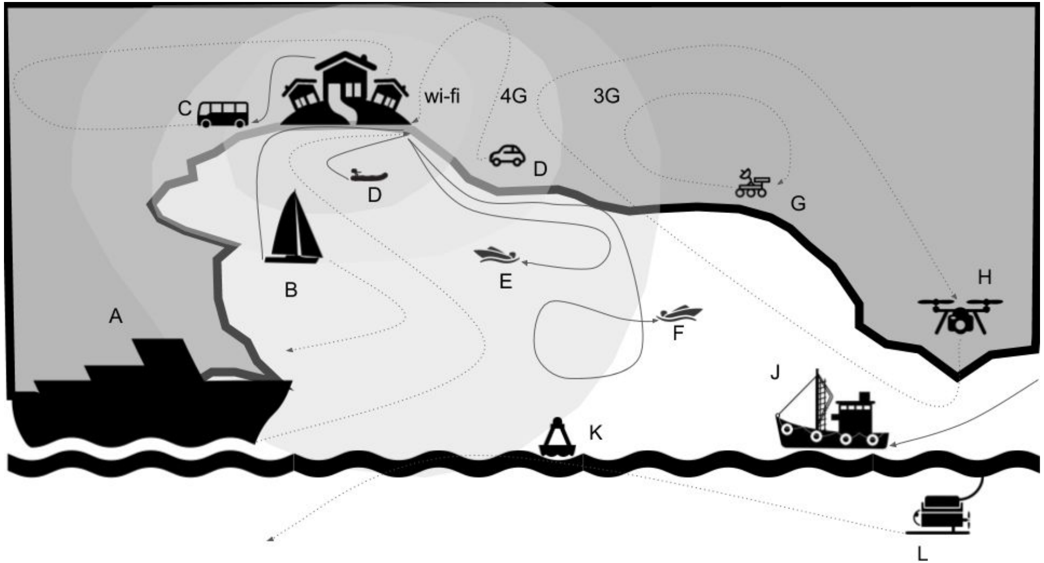
Hard to obtain, lack of public data.



Echosounder: a device using echo sounds from the sea bottom to evaluate the depth.

Multibeam: an echosounder with steroids.

Introduction, contextualization and motivations



- Huge amount of surface data
- Problem size characterized by a remarkable complexity.
- Data have to be continuously updated.

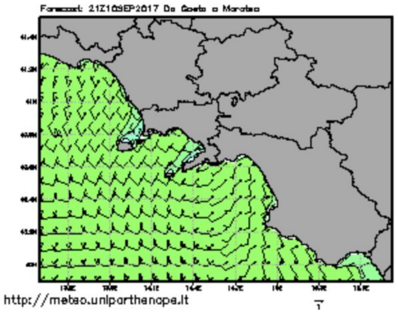
- Internet of things based crowd-sourcing tools.
- Using an ad-hoc IoT Data Transfer Protocol.
- Poor and intermittent data connection availability.

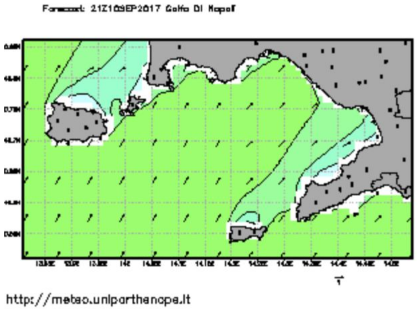
Introduction, contextualization and motivations

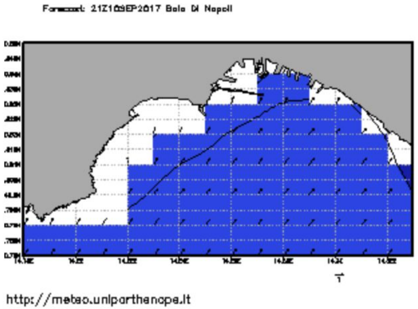
Wind driven wave model

Wave watch3

Wind-driven sea waves: Wave Watch III (WW3) max ground resolution 1Km, outlook 144h, updated each 24 hours



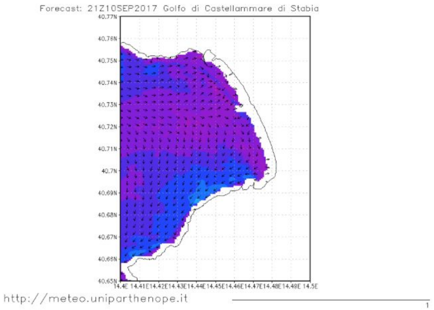


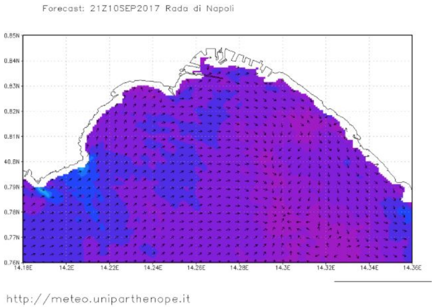


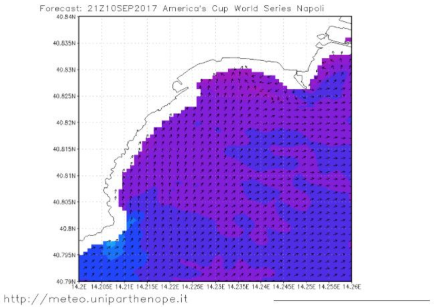
Surface currents: Regional Ocean Model System (ROMS) max ground resolution 80m, outlook 120h, updated each 24 hours

Current model

Regional Ocean Model System

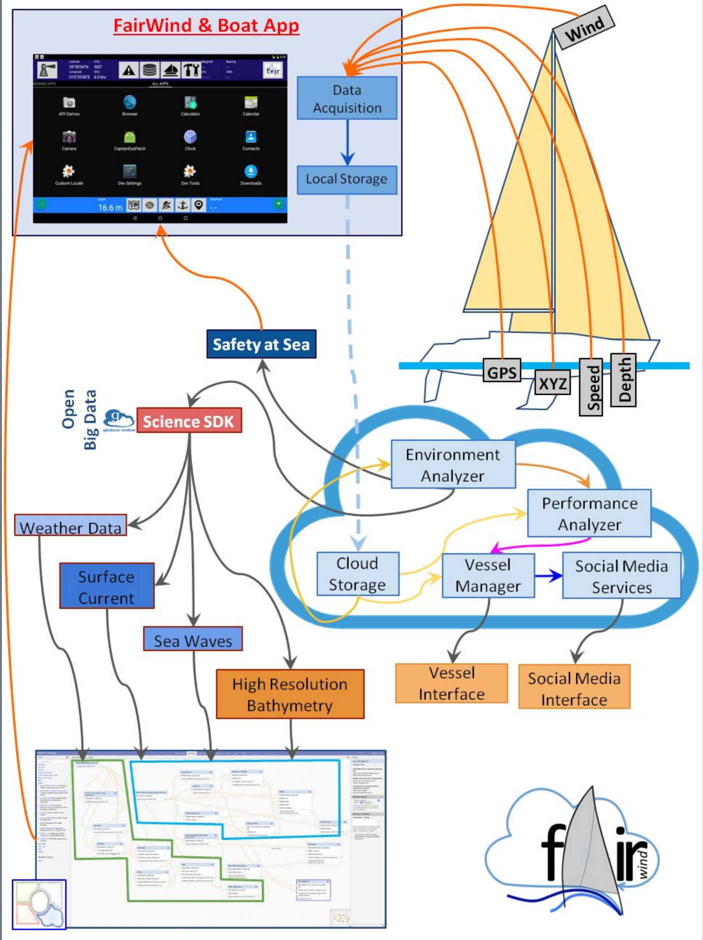






We need for more accurate models for environment management.

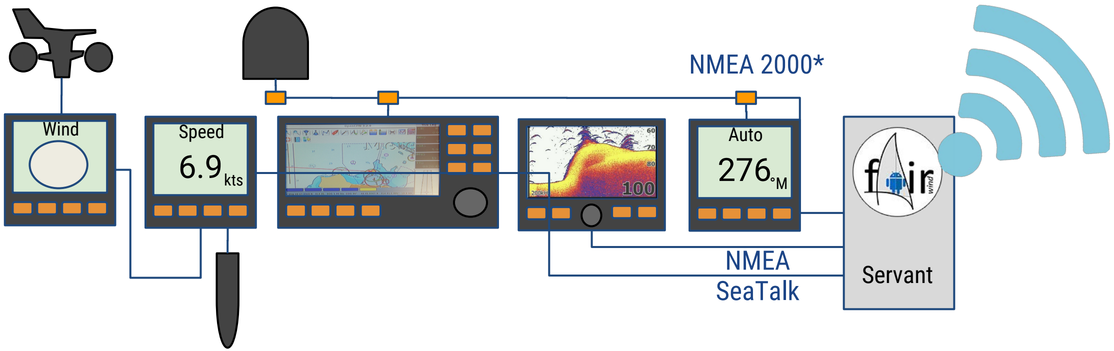
Data crowdsourcing at the sea: FairWind



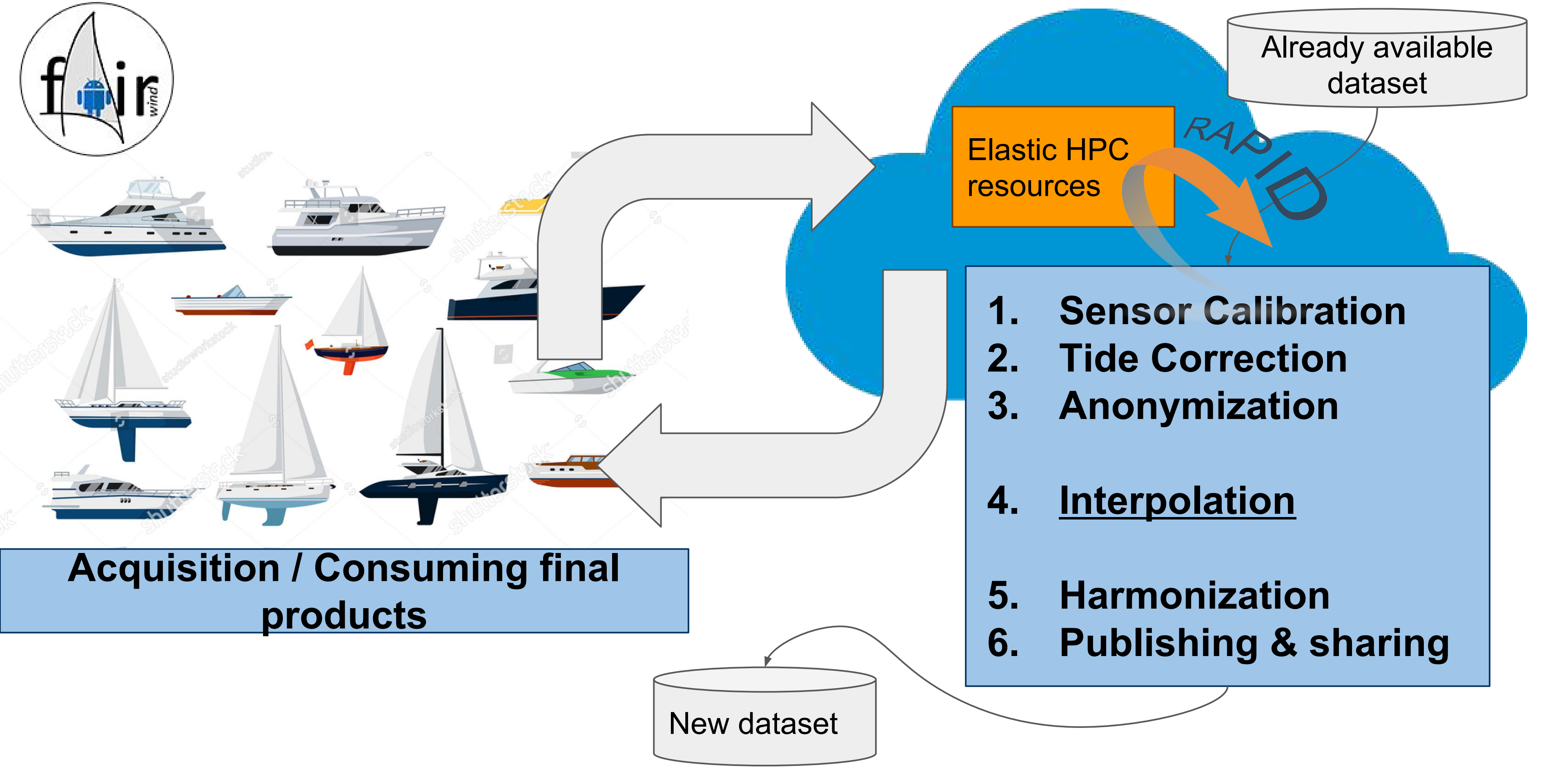
Smart, cloud-enabled, multifunctional navigation system for **yachts and leisure boats**.

FairWind collects, stores and send to the cloud logged data from sensors.

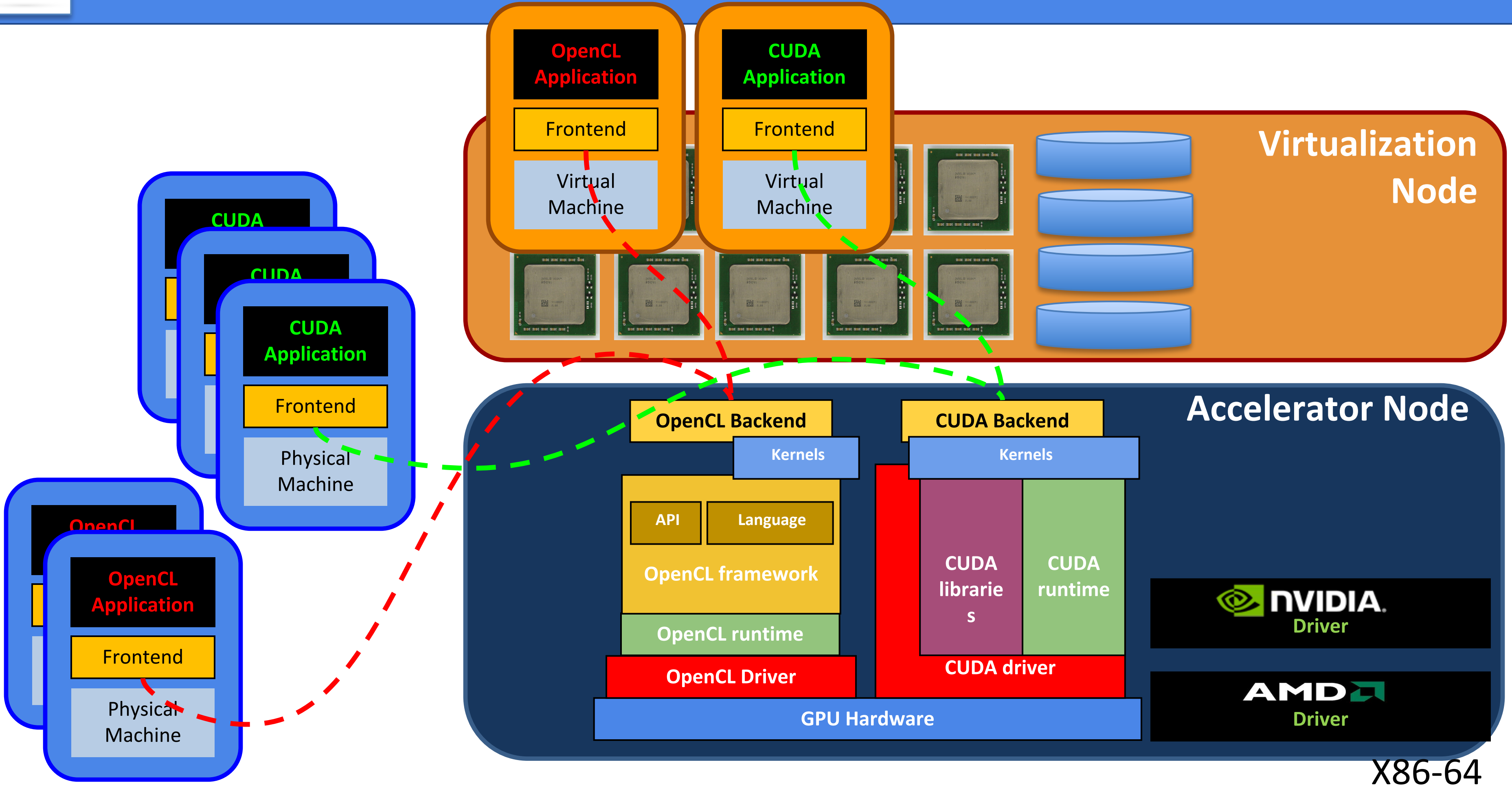
FairWind participates in a internet of floating things ecosystem thanks to the SignalK data format.



Data crowdsourcing at the sea: big picture



Acceleration infrastructure



GVirtuS: it could work!

gpGpu Virtualization Service

- **GVirtuS** is a software component for GPGPU virtualization and "remoting".
- **Virtualization:** Makes available a subset of CUDA API to virtual machines running on a host equipped with a CUDA enabled device.
- **Remoting:** Makes available a subset of CUDA API to a physical or virtual machine(s) using a remote host equipped with a CUDA device.

- **GPUs**
  - Hypervisor independent
  - Communicator independent
  - **GPU independent**

https://github.com/raffmont/gvirtus/

GVirtuS: it could work!

- GVirtuS provides low power systems and devices with native high end GPGPU capabilities:

- CUDA|OpenCL Kernel offloading
- Transparent mocking CUDA|OpenCL APIs (Linux)
- Simple programming model (Java/Android - only CUDA)

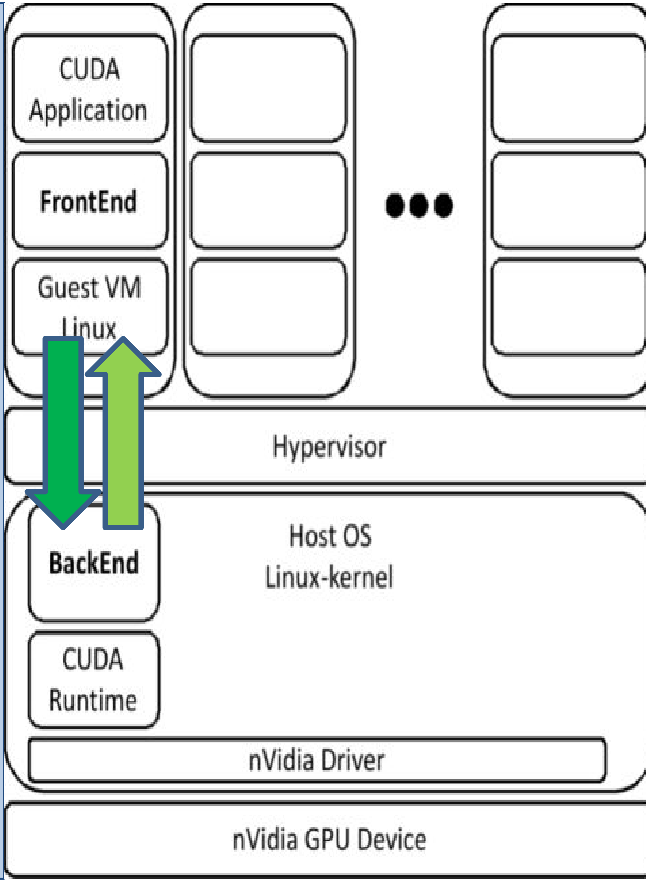
https://github.com/raffmont/gvirtus/

GVirtuS: it could work!

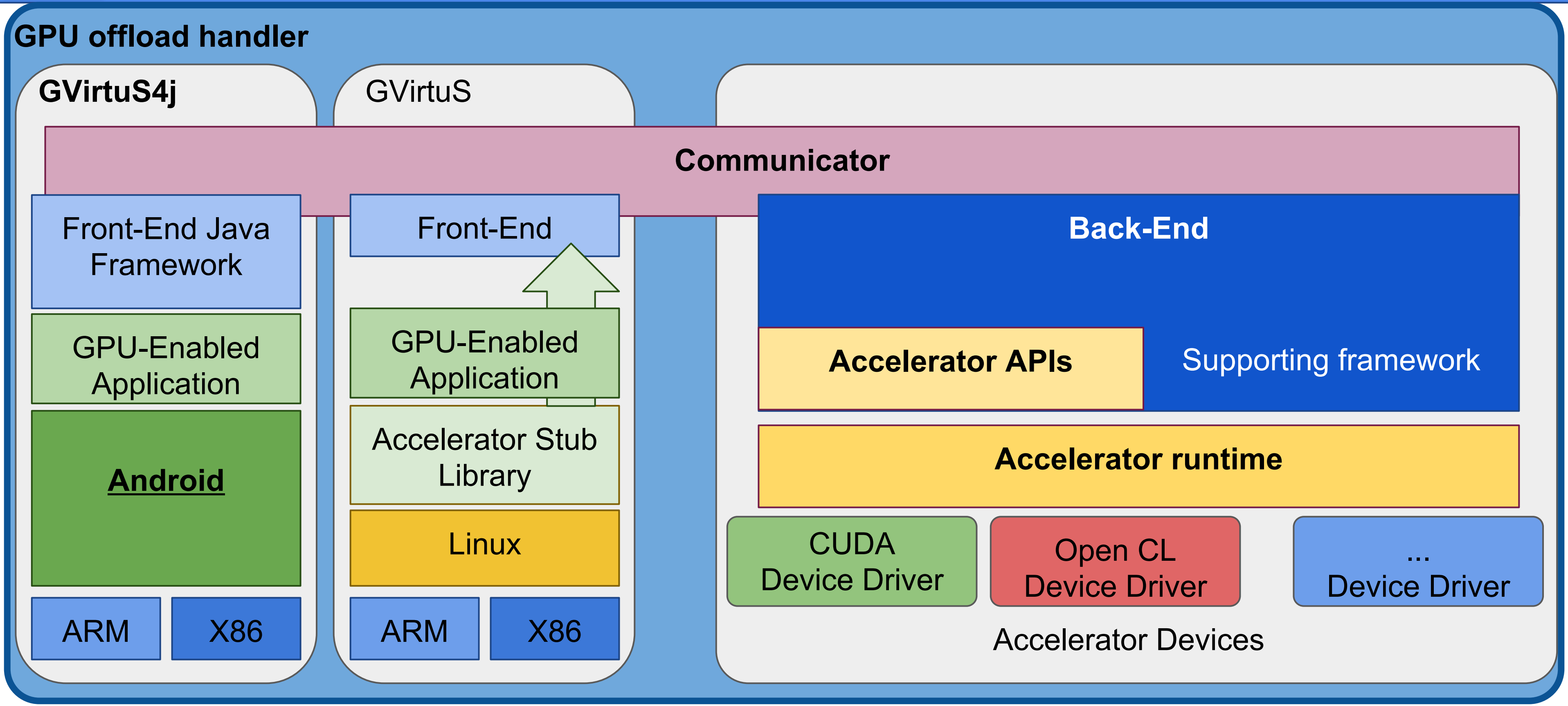
- **Hope (2009):** GPGPU virtualization is part of a research project proposal submitted to a local call (not funded).

- **GVirtuS (2010-13):** is the general virtualization system developed in 2009 and firstly introduced in 2010 enabling a completely transparent layer among GPGPUs and VMs.

- **RAPID GVirtuS (2014-17):** is the RAPID incarnation offering CUDA 6.5 support, memory management and Java/Android support.



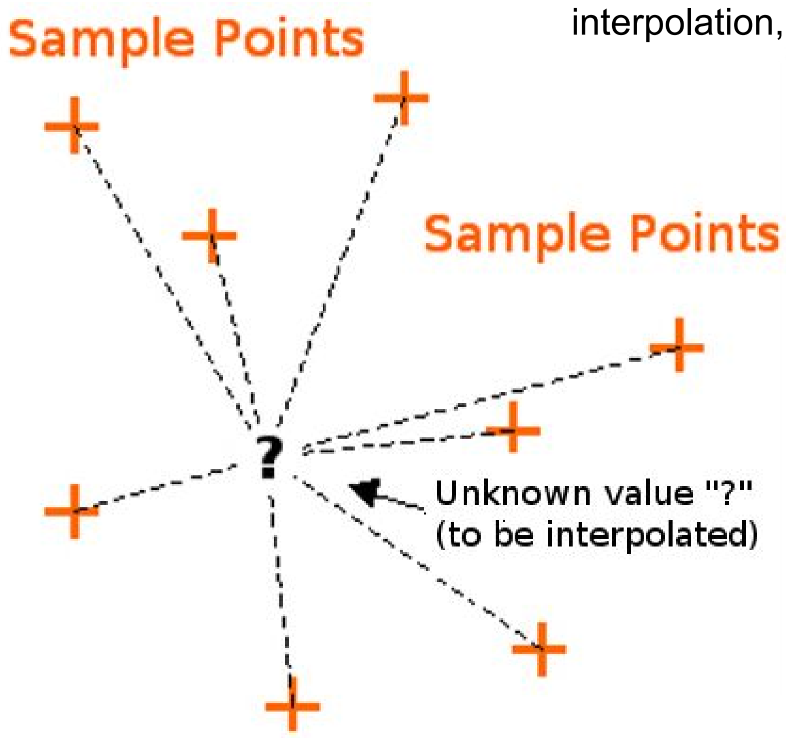
GVirtuS: Architecture



Unique Back-End, multiple Front-Ends

The interpolation algorithm

The **Inverse Distance Weighted** (IDW) is a deterministic method for spatial interpolation, based on the principle that near points have similar values.



interpolated value

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

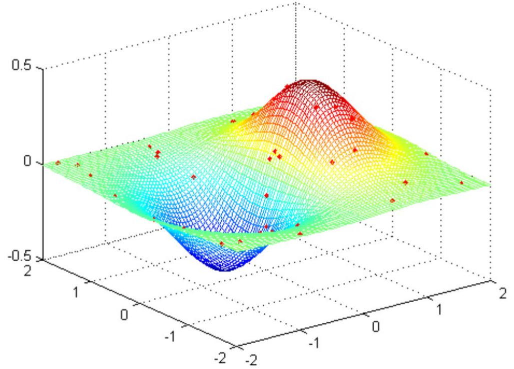
Weights (Euclidean distance)

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

interpolated value with fixed radius R

$$z_j^* = \frac{\sum_{p_i \in Q_j} \lambda_{ji} z_i}{\sum_{p_i \in Q_j} \lambda_{ji}}$$

$$Q_j = \{p_i : d(p_i, q_j) < R\}$$



IDW on CUDA environment

G-IDW

- Each thread interpolates a different value computing the weight for each known value and updating the weighted mean at the same time.
- Block threads are synchronized to store dataset points into shared memory before the interpolation phase.
- For too large datasets, the points are stored into shared memory in different chunks.

```
initialize α
loc_q ← q(tid)
for each chunk c do
    i ← 0
    start_ind ← tid * stride
    while (i < stride) and (i + start_ind) < size(c) do
        put p_c(start_ind + i) into shared memory
        i ← i + 1
    end while
    synchronize threads
    for i ← 1 ... size(c) do
        loc_p ← p_c(i) from shared memory
        d ← dist(loc_p, loc_q)
        if d ≠ 0 then
            if d_ij < R then
                λ ← d^(−α)
                z*(tid) ← z*(tid) + λ z_c(i)
                wsum ← wsum + λ
            end if
        else
            z*(tid) ← z_c(i)
            wsum ← 1
            break and skip this cycle for the next chunks
        end if
    end for
    synchronize threads
end for
put z*(tid)/wsum into global memory
```

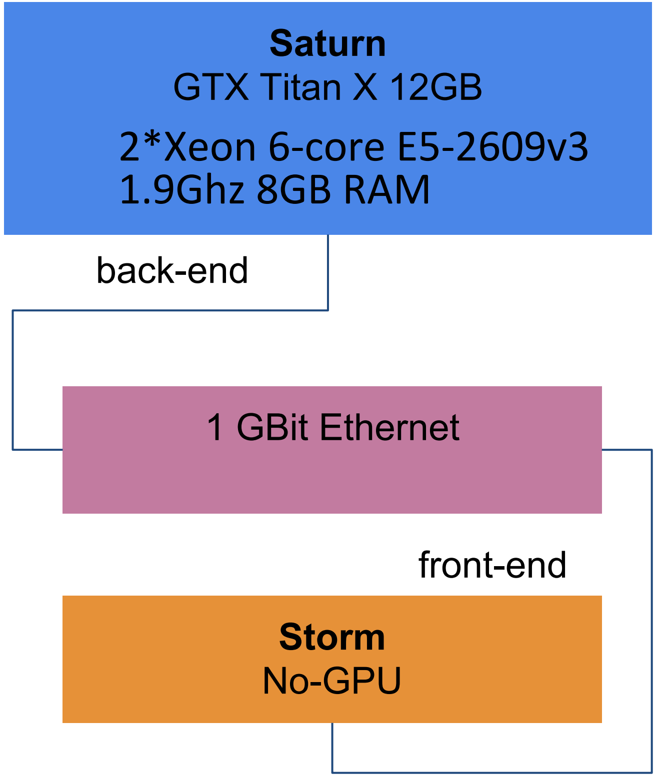
IDW on CUDA environment

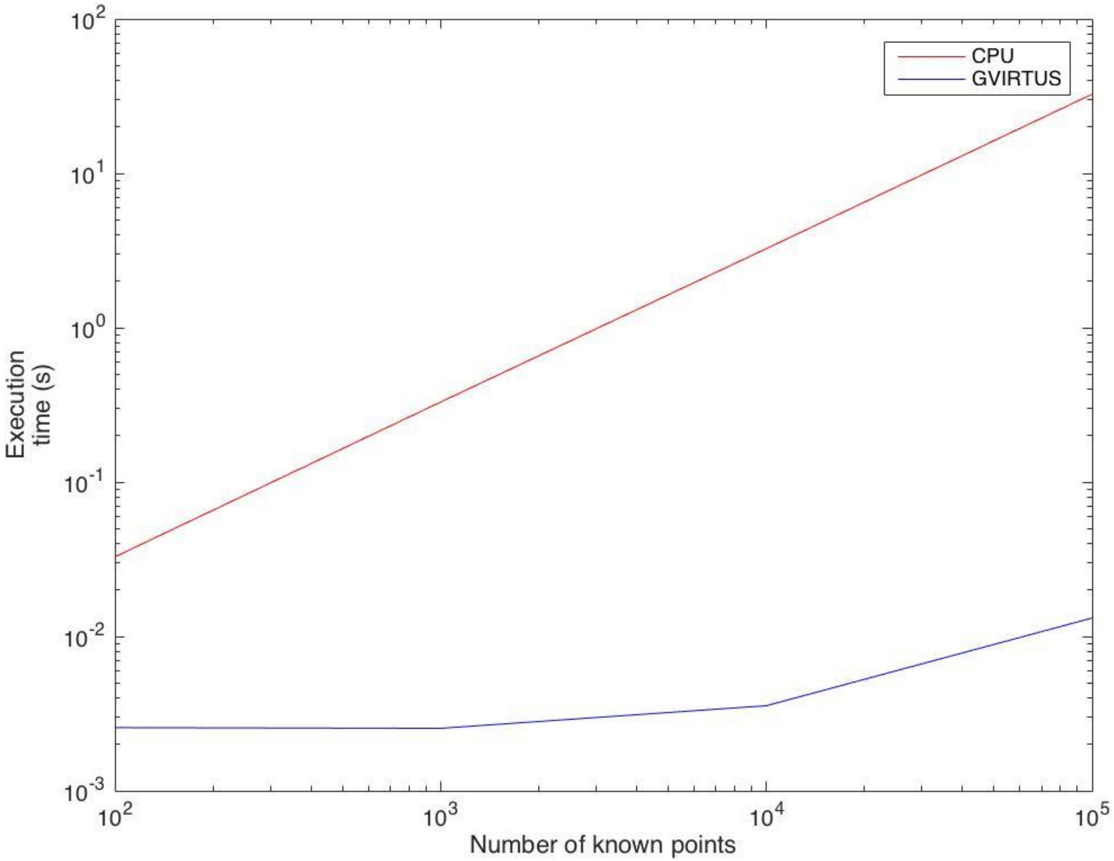
G-IDW-MV	G-IDW-MVblas

- The matrix ƛ is to compute, where the i-th row contains the weights for the i-th value to be interpolated.
- Threads are synchronized to store dataset points into shared memory.
- The i-th thread computes the elements of the i-th row.
- ƛ is multiplied by the vector containing the known values. The i-th element of the result vector is divided by the sum of the weights for the i-th value in order to get the weighted mean.
- G-IDW-MVblas uses the cuBLAS library ad hoc routine.

```
initialize α and R
loc_q ← q(tid)
for each chunk c do
  load dataset points into shared memory as shown
  for i ← 1...size(c) do
    loc_p ← p_c(i) from shared memory
    d ← dist(loc_p, loc_q)
    if d ≠ 0 then
      if d_ij < R then
        λ ← d^(−α)
        Λ(tid, column of p_c(i)) ← λ
        wsum ← wsum + λ
      else
        Λ(tid, column of p_c(i)) ← 0
      end if
    else
      put all row values to 0
      Λ(tid, column of p_c(i)) ← 1
      wsum ← 1
      break and skip this cycle for the next chunks
    end if
  end for
  synchronize threads
end for
use a strategy to compute Λz
z*(tid) ← z*(tid)/wsum
```

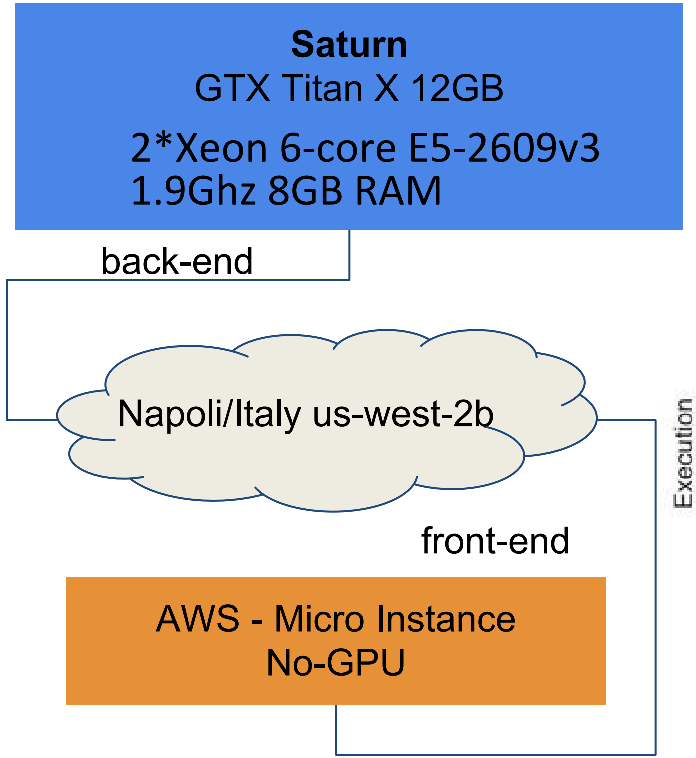
Benchmarking (G-IDW: Local B/E - Local F/E)

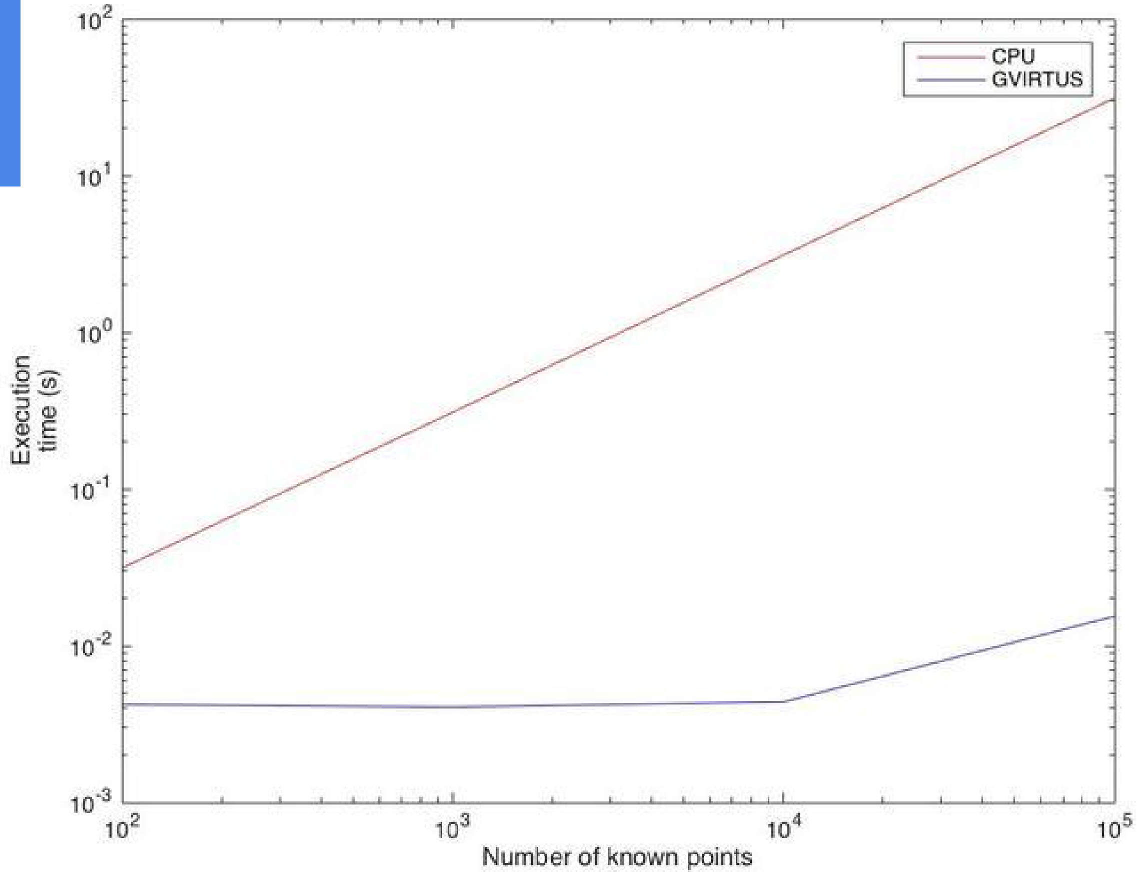




- Remoting

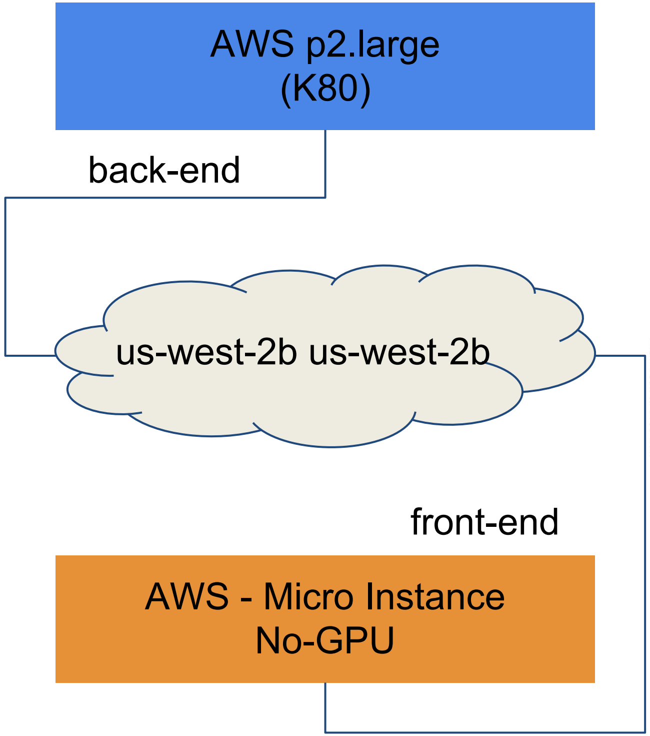
Benchmarking (G-IDW: Local B/E - AWS F/E)

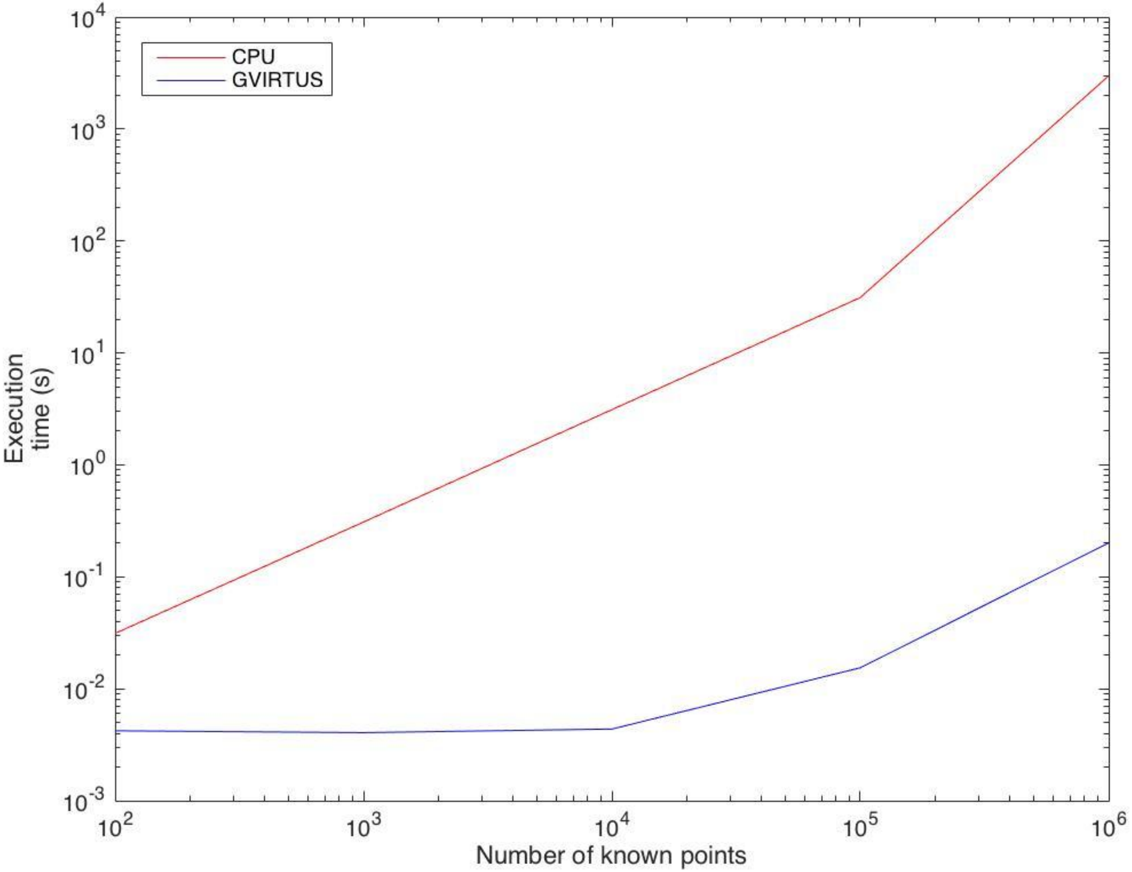




- Remoting

Benchmarking (G-IDW: AWS B/E - AWS F/E)

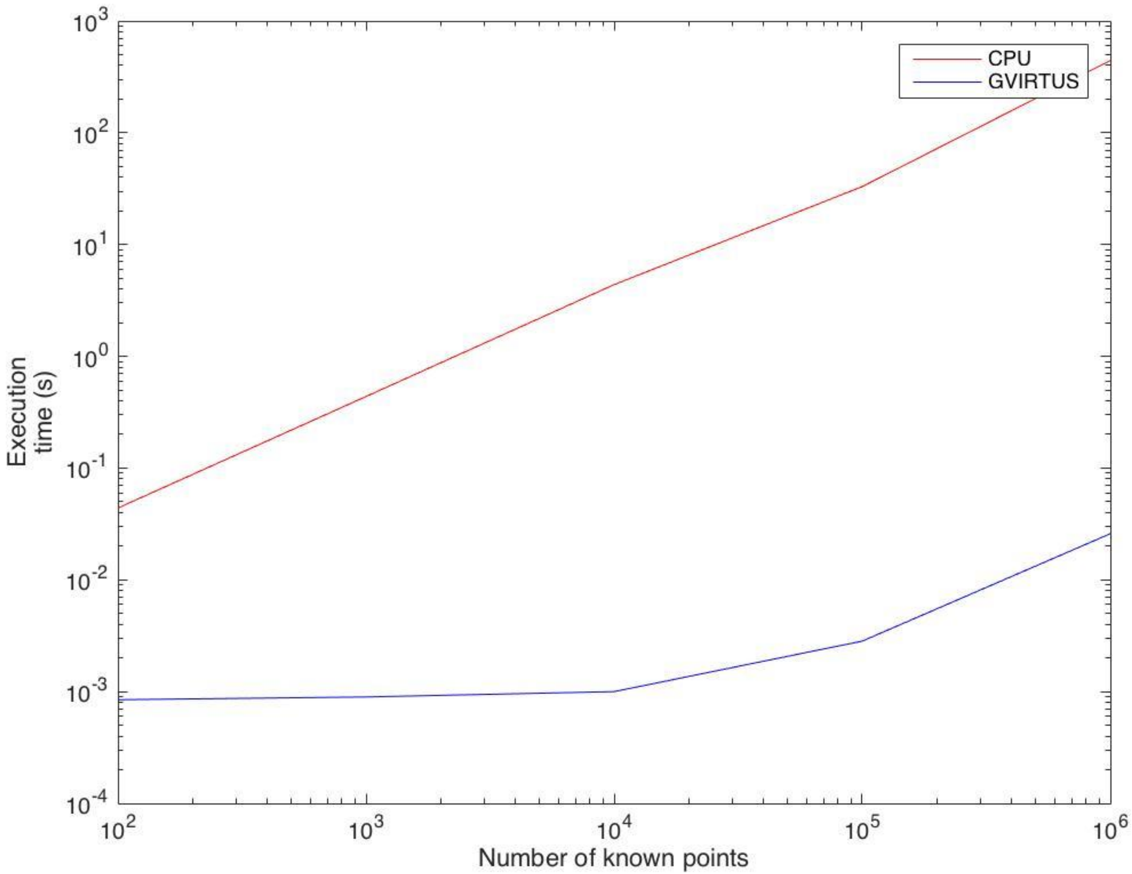




- Remoting

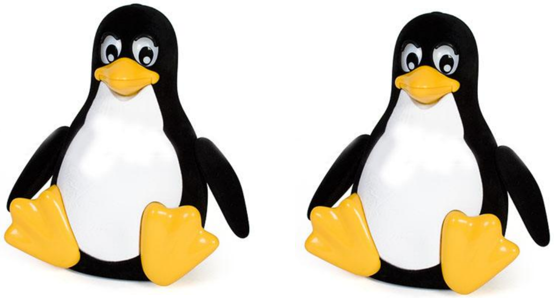
Benchmarking (G-IDW: Local B/E - KVM F/E)

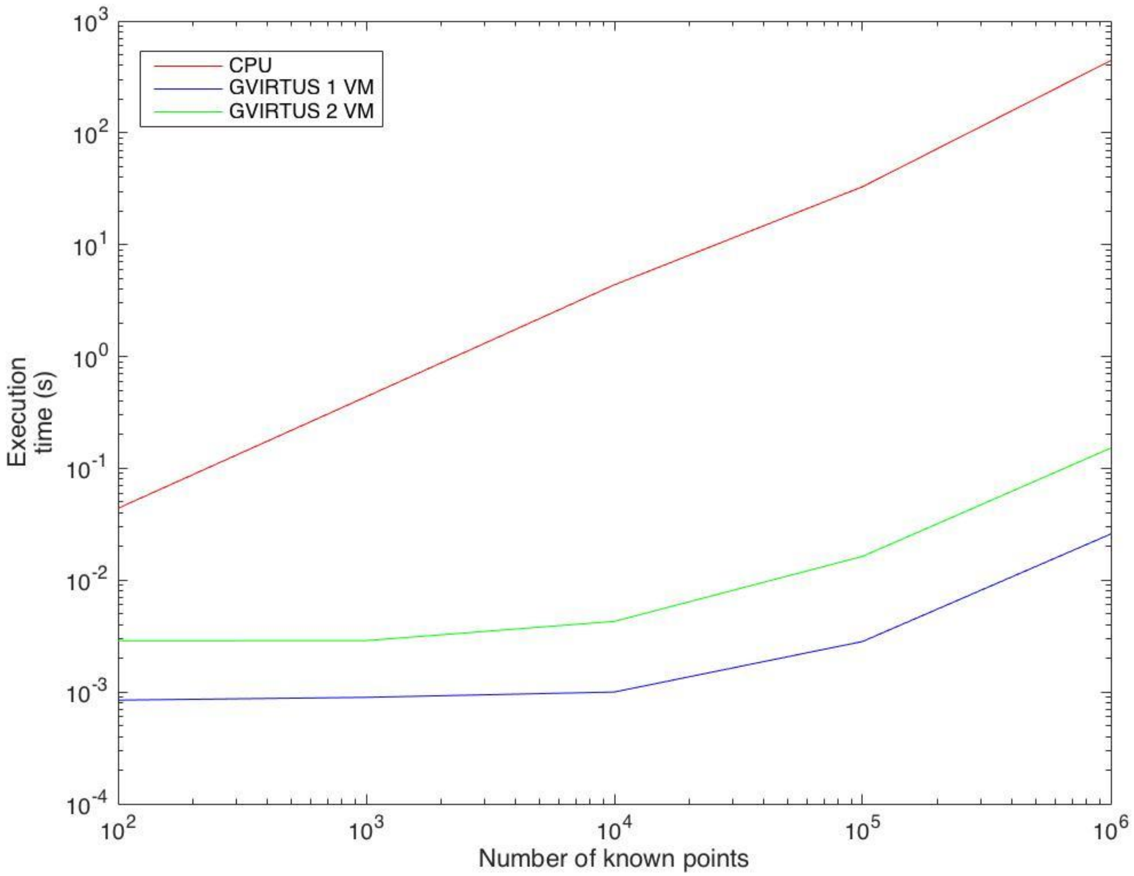
- Virtualization with KVM - One virtual machine



Benchmarking (G-IDW: Local B/E - KVM F/E)

- Virtualization with KVM - Two virtual machines

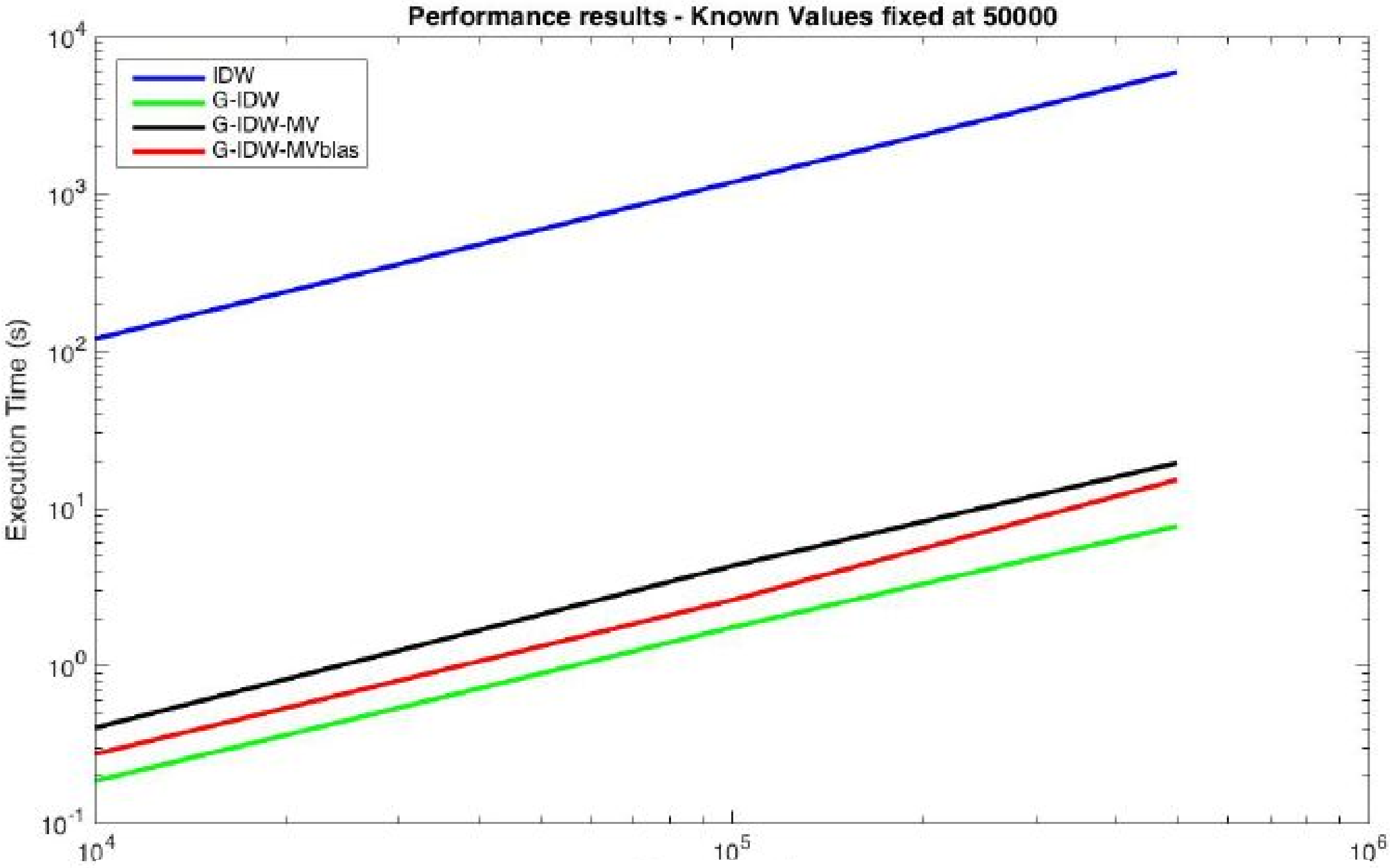




Evaluation

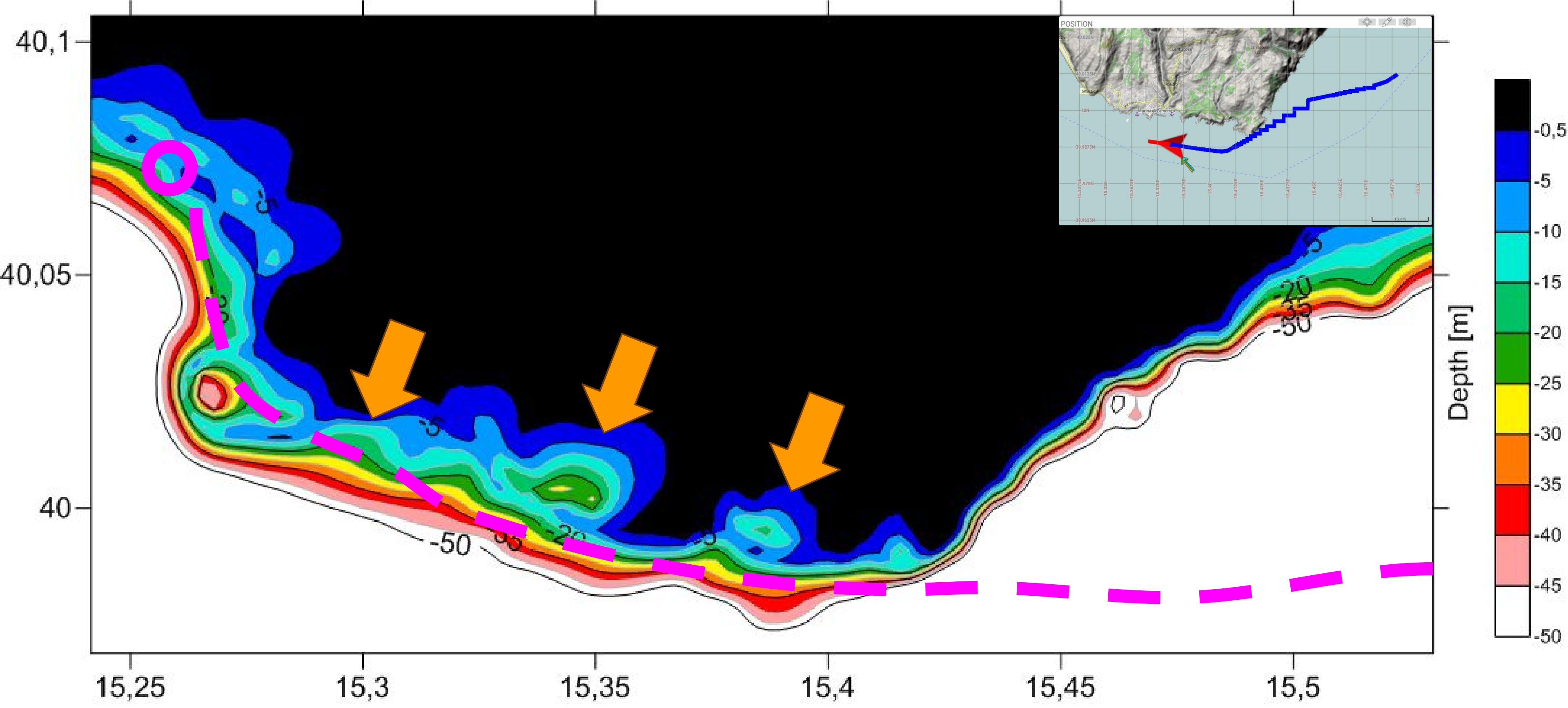
Known Values	Number of Interpolated Points	IDW(s)	G-IDW(s)	G-IDW-MV(s)	G-IDW-MVblas(s)
10^3	10^4	2.365	0.006	0.009	0.009
10^3	10^5	23.711	0.045	0.09	0.089
10^3	$5 \cdot 10^5$	118.296	0.204	0.448	0.411
10^4	10^4	23.734	0.036	0.071	0.064
10^4	10^5	236.800	0.313	0.688	0.560
10^4	$5 \cdot 10^5$	1185.581	1.545	3.462	2.775
$5 \cdot 10^4$	10^4	120.293	0.186	0.406	0.276
$5 \cdot 10^4$	10^5	1183.199	1.757	4.315	2.62
$5 \cdot 10^4$	$5 \cdot 10^5$	5935.798	7.729	19.513	15.227

Evaluation



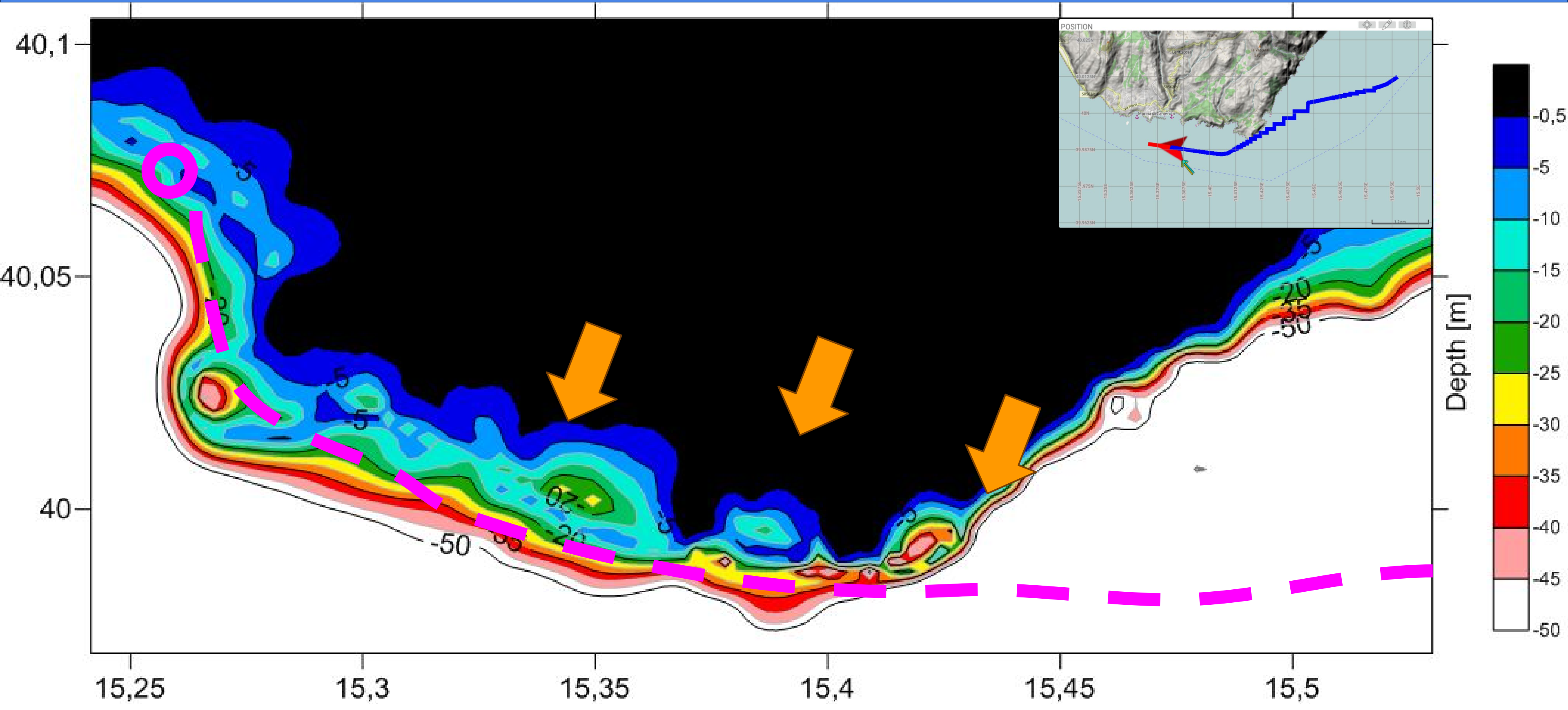
Number of Interpolated Points

Results: before...



EMODnet bathymetry dataset: http://www.emodnet-bathymetry.eu

Results: after...



Crowdsourced data using FairWind: http://fairwind.uniparthenope.it

Conclusions and future directions

We explored the feasibility of a CUDA based interpolation system for continuously updating bathymetry production from crowd-sourced data with promising results.

- Implement new interpolation algorithms more suitable for geographical applications.
- Implement a robust internet of thing data transfer protocol.
- Couple the system components with a FACE-IT Galaxy Workflow.

http://fairwind.uniparthenope.it
http://github.com/openfairwind

GET IT ON Google Play

http://rapid-project.eu
http://github.com/RapidProjectH2020
http://github.com/raffmont/gvirtus

This research has been supported by the Grant Agreement number: 644312 - RAPID - H2020-ICT-2014/H2020-ICT-2014-1
“Heterogeneous Secure Multi-level Remote Acceleration Service for Low-Power Integrated Systems and Devices”.