CIVIL PROTECTION
DISASTER RISK REDUCTION
BIODIVERSITY

WRF-GO

workflow manager for meteo prediction and applications

Emanuele Danovaro

RESEARCH FOUNDATION
cima
OBSERVE TO PREDICT,
PREDICT TO PREVENT.

FONDAZIONE CIMA
CIMA RESEARCH FOUNDATION

CENTRO INTERNAZIONALE IN MONITORAGGIO AMBIENTALE
INTERNATIONAL CENTRE ON ENVIRONMENTAL MONITORING

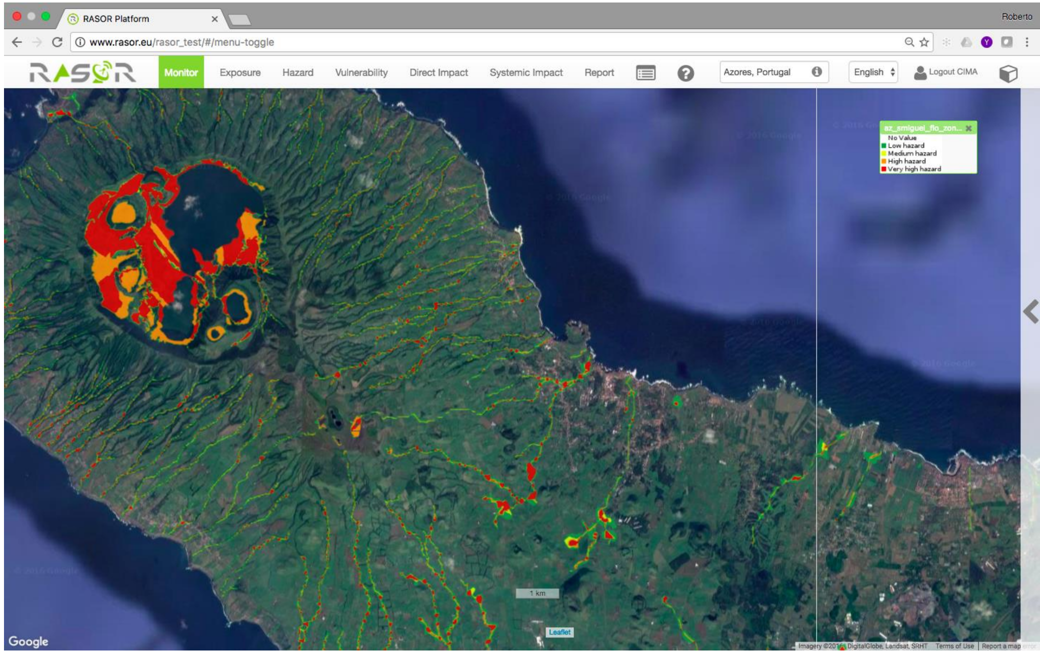
CIMA Fields of activities

- Hydro-met applications for Civil Protection
- Risk assessment
- Climate Change and DRR: Targeting Extremes
- EO assisted applications
- ICT Tools for research and services
- Liability, Responsibility & Governance in risk
- Capacity building and education from the international to the local dimension
- Marine Ecosystem Monitoring

Disaster Risk Reduction

Whenever there is a risk, we are asked to complement operational services:

- High-resolution downscaling (WRF)
- Model chains (meteo + hydro or wildfire models)
- Impact estimation (windstorm, floods, …)



Commercial services

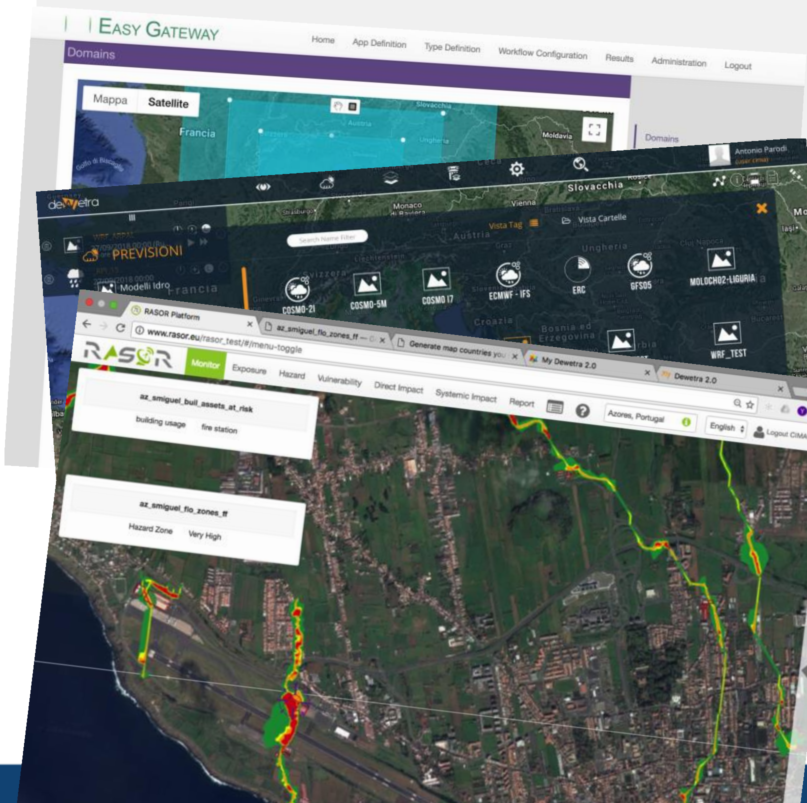
Forecast of energy production (wind turbines, photovoltaic)

- Following day
- Hourly updates

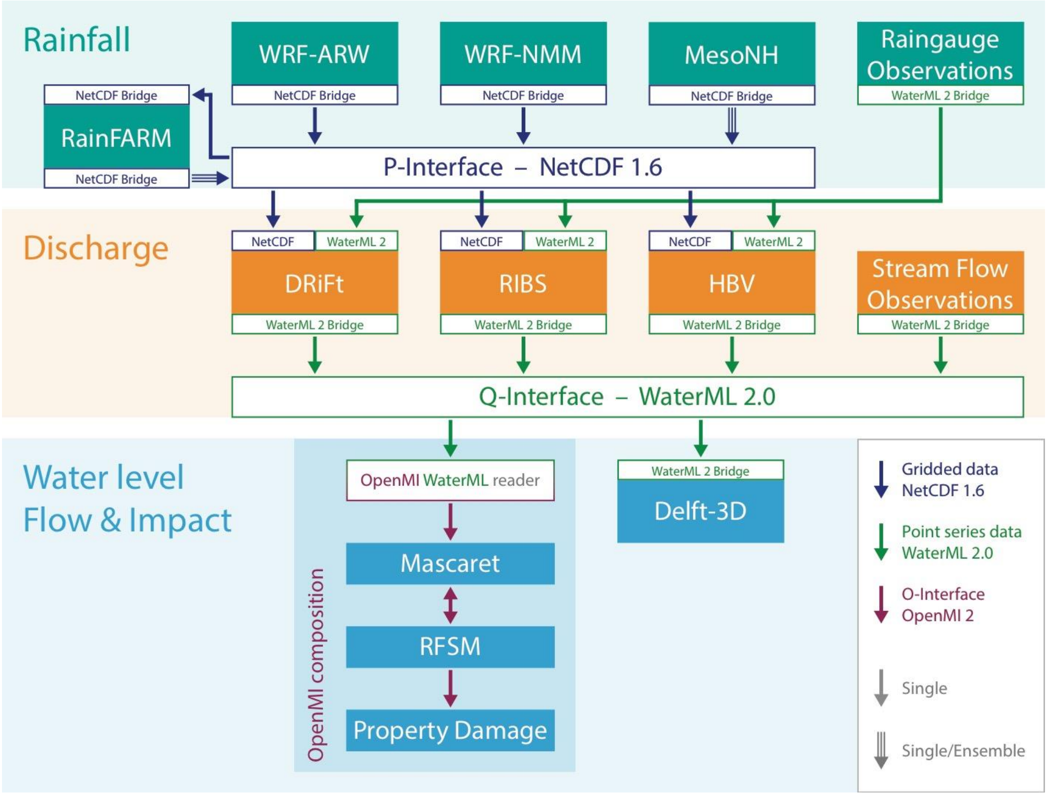
Insurance (floods, hail)

Our tools

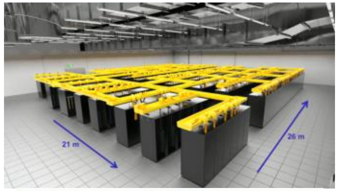
- Multi-model ensemble to target extreme events
- Model chains (meteo-hydro/wildfire/energy-impact)
- Web interface to configure experiments & trigger execution
- myDewetra for situation awareness and decision support
- Rasor to assess the impact of extreme events



Model chains (from DRIHM project)



HPC



HTC

Cloud

Sample workload

Meteo downscaling

- fetching boundary conditions + pre-processing (36 cores, 20’)
- 48h WRF at 1.5km at national level (1500-1800 cores, 3 hours) (180 GB of output files - uncompressed NetCDF)
- UPP post-processing + delivery (3 GB compressed GRIB2)

A smaller case

- 48 WRF at 2km at regional level (200-300 cores, 2 hours)

Hydro models, Impact assessments, … (tens of cores, minutes)

Computing resources

Reserved resources:

- 50 nodes (1800 cores) on CINECA Tier-1 HPC system (WRF 1.5km)
- In-house small cluster ~300 skylake cores (WRFDA 2km)
- AWS reserved VMs for operational services (flood, wildfire)

Resources on demand:

- Grants on SuperMUC & Cineca Tier-0 HPC system
- AWS on-demand clusters (c5 instances)

How to fit ? 1/2

Beside pure computation, there are time consuming tasks:

Fetching boundary condition, preprocessing, post-processing, data transfer.

Files represent a timeframe: we adopts event-based programming / streaming programming to reduce latencies

We use a messaging system to notify the workflow manager and to trigger events.

How to fit ? 2/2

Are reserved resources available ? We use them!

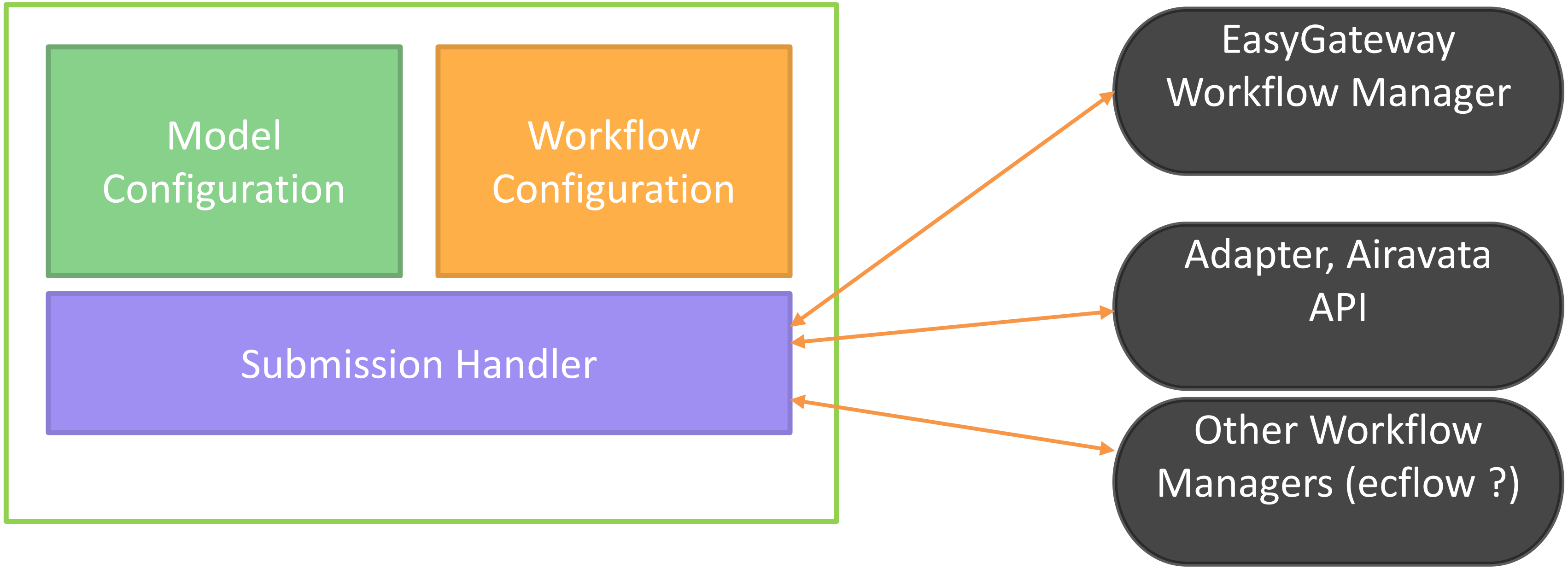
otherwise

Best-effort HPC resources may impose long queues

We submit on multiple resources, if one starts the computation (within a deadline) we exploit it and free the other resources.

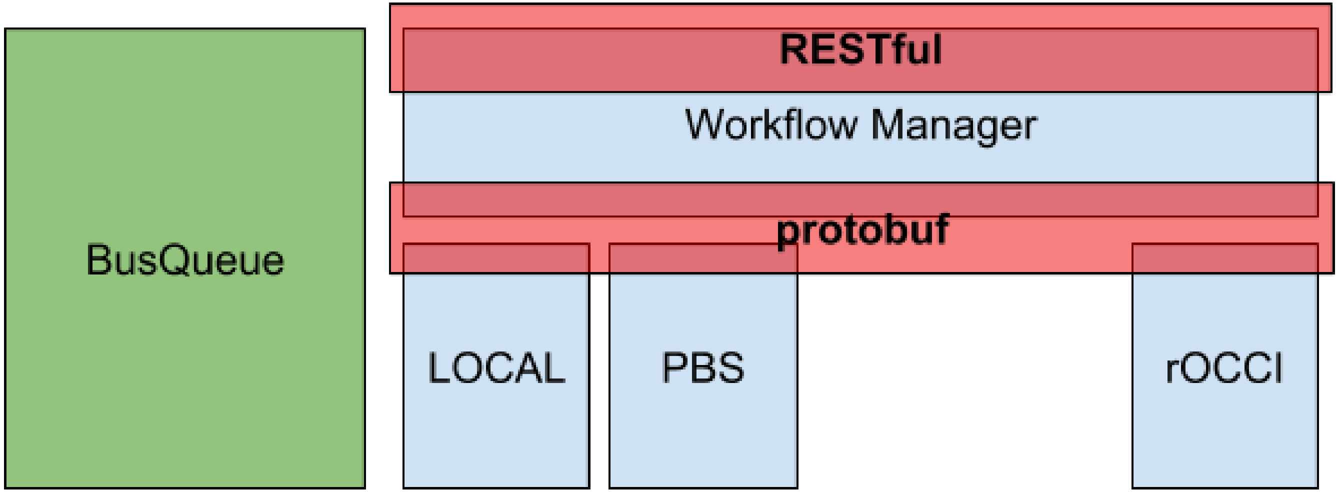
If none succeed in time, the task is executed on a virtual cluster on AWS

How it works


Model Configuration
Workflow Configuration
Submission Handler
EasyGateway Workflow Manager
Adapter, Airavata API
Other Workflow Managers (ecflow ?)

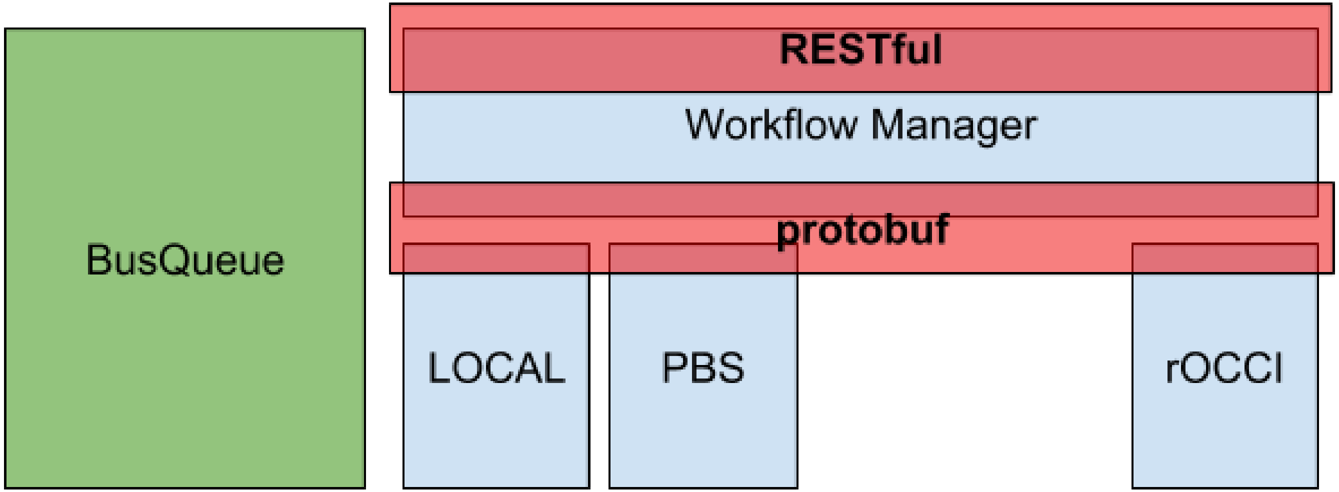
Workflow manager

- The main component – it handles all workflow submission request.
- It exposes a RESTful API for the workflow submission, monitoring and administration.
- Submission to resources is performed using Resource Specific Modules (RSM)



Resource Specific Modules

- A RSM performs a single job submission, monitoring and administration on a specific resource.
- WM - RMS communication protocol is based on protobuf.
- Real-time messages are sent to a Pub-Sub messaging system (BusQueue)



WM - RSM communication

The WM can:

- Request the submission of a job
- Request the termination of a job
- Publish on the BusQueue change of state of a Workflow
- Consume the BusQueue to detect:
  a change in the state of a job
  other notifications from the RSM

WM - RSM communication

The RMS must:

- Return OK/FAIL when a new job is submitted (i.e.: missing parameters)
- Publish on the BusQueue change of state of a Job
- Consume the BusQueue for job termination notification (i.e. no polling!)

The RMS can:

- Publish on the BusQueue specific events (an output file has been written)
- Publish on the BusQueue in near real-time log/stdout/stderr

A few technical choices

Workflow Manager & Resource Specific Modules

developed in go – good for concurrency, low memory footprint

BusQueue is NATS, a zero-configuration, fast and lightweight messaging system

Web portal for workflow configuration & execution
developed in TypeScript + Angular + node.js

Deployed on AWS (t2.micro can manage hundreds workflows/day)

Pros & cons

- Fast & lightweight
- Handles restart/failures
- Easy to extend
- Easy to connect to other workflow managers

- To battle-test
- Limited credential management (ssh public keys)
- No accounting

Future directions

Support data streaming:

- model send output to streaming pipeline
- next model in the workflow receives data from the pipeline
- models are loosely coupled

File-based IO is no longer required
We can have asynchronous IO by sinking to a file the data coming from the pipeline

Thanks!

Emanuele Danovaro

emanuele.danovaro@cimafoundation.org