# Putting FLEXPART to REST: The Provision of Atmospheric Transport Modeling Services


*D. Morton[1,2], D. Arnold[3]*

[1]Boreal Scientific Computing LLC, Fairbanks, Alaska, USA
[2]Geophysical Institute, University of Alaska Fairbanks, USA
[3]Zentralanstalt für Meteorologie und Geodynamik, Vienna, Austria

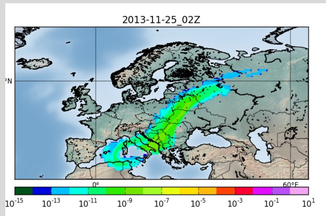

## Background & Aims

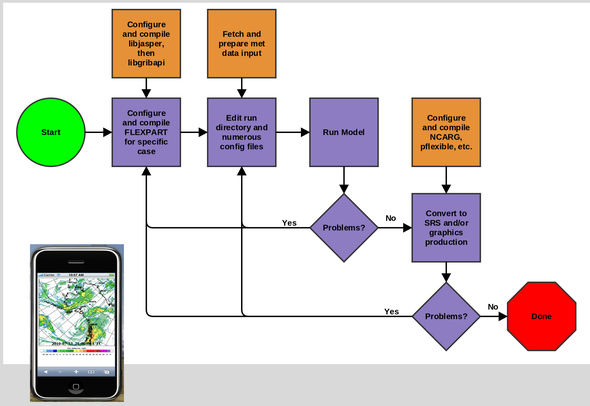
- FLEXPART is a Lagrangian particle dispersion model originally designed for calculating dispersion of air pollutants from point sources. Forced by gridded meteorological variables, FLEXPART is used to simulate the atmospheric transport of wildfire smoke, volcanic ash and radionuclides. It is frequently run in backwards mode to provide information for the determination of emission sources such as nuclear emissions and greenhouse gases.

- Like many atmospheric models, getting it compiled, installed, configured, running, and postprocessed is often a tedious endeavor with many opportunities to make mistakes. Users of these models frequently run into unproductive frustration as they struggle with numerous computing tasks, distracting from the higher level pursuit of information.

- Many consumer applications now provide ubiquitous access to computing resources with almost no requirement for computing skills – navigating maps, selecting and viewing movies, traditional word processing – by deploying these services in cloud environments, making them accessible from any device, anywhere on the planet.

- Our aim is to make models like FLEXPART as easy to use as these consumer applications



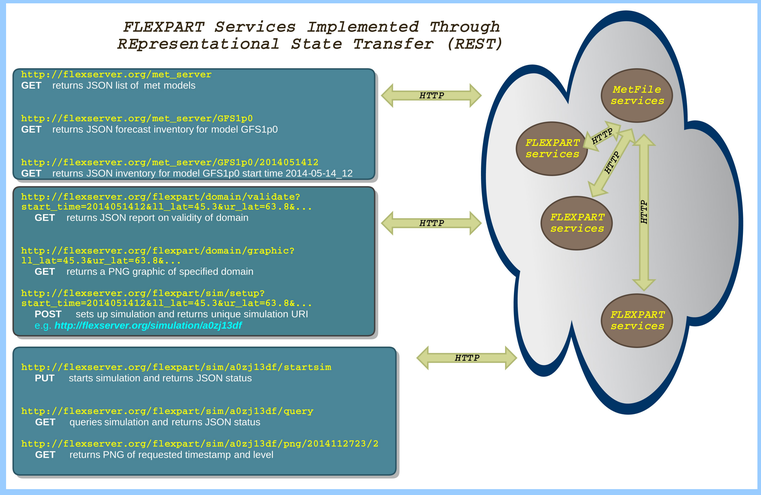

## Methods

- Cloud-based consumer services are frequently based on a REpresentational State Transfer (REST) client-server web architecture.

- Services (or resources) are accessible through a loosely coupled, language independent Application Programming Interface (API), implemented as Uniform Resource Identifiers (URI) over an HTTP framework.

- Design philosophy – Unix-like, small simple cohesive components that can be flexibly used in numerous contexts to define workflow.

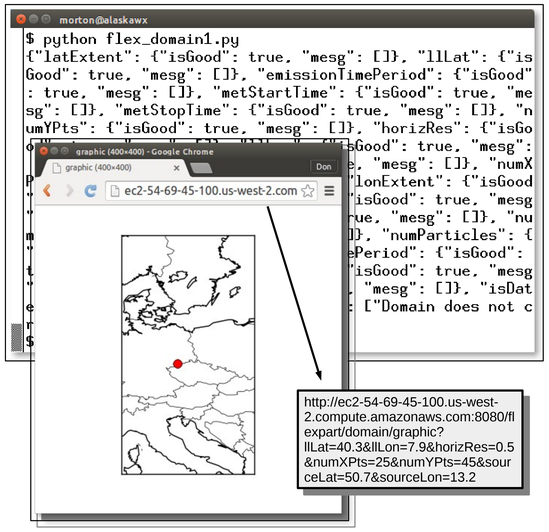
- Running the model reduces to sending properly-formed URI requests to
- model services and processing responses – much like typical web interactions



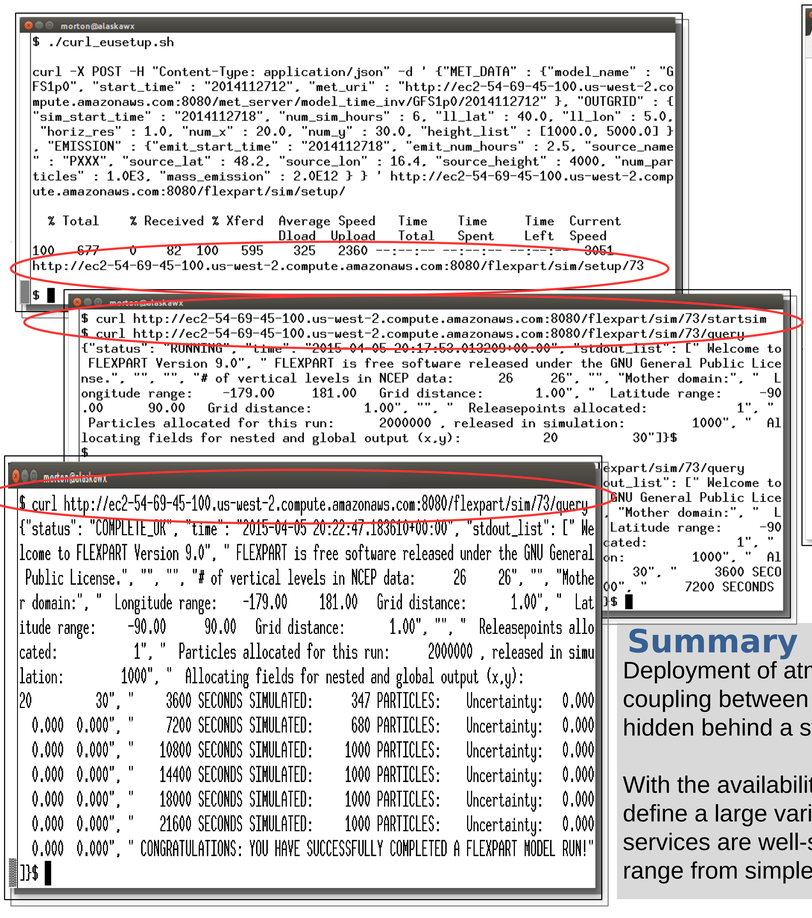

## Deployment – supports any HTTP client

### Validate and view domain

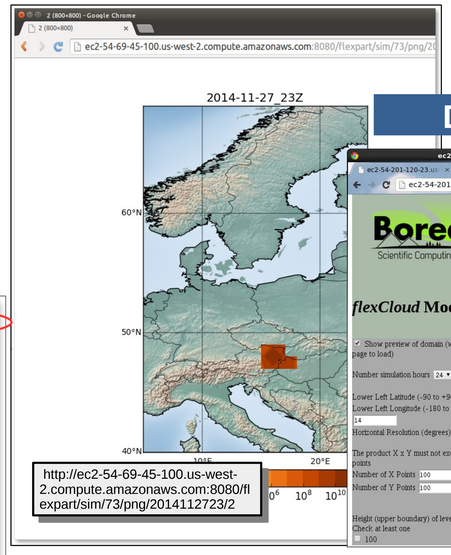
```
import urllib
import urllib2

theURL = """http://ec2-54-69-45-100.us-west-
2.compute.amazonaws.com:8080/flexpart/domain/validate"""

# Set up parameters for server
params = {
          'llLat' : '45.0',
          'llLon' : '-155.1',
          'horizRes' : '0.5',
          'numXPts' : '10',
           .
           .
           .
          'emissionStartTime' : '2014051405',
          'emissionNumHours' : '3',
          }
data = urllib.urlencode(params)

## Make the request
#req = urllib2.Request(theURL, data)

# Read and print the response
fullURL = theURL + '?' + data
urlFH = urllib2.urlopen(fullURL)

the_response_data = urlFH.read()

urlFH.close()

print the_response_data
```



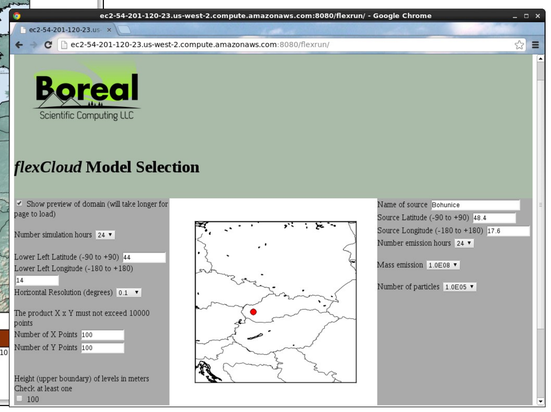

### Set up, start and query simulation




### PNG graphics production




### Driving web interface




## Summary


Deployment of atmospheric models as-a-service requires robust architectures that exhibit loose coupling between clients and servers. The implementation details of the services should be hidden behind a structured API such that any client system, on any device, can access.

With the availability of a flexible set of component services, client developers have the means to define a large variety of interactive and automated workflows. By using a REST architecture, the services are well-suited for cloud deployment, allowing for on-demand modeling activities that range from simple, experimental simulations to large-scale ensemble production.