Gfdnavi,
Web-based Data and Knowledge Server Software for Geophysical Fluid Sciences, Part II: RESTful Web Services and Object-Oriented Programming Interface

Seiya NISHIZAWA, Takeshi HORINOUCHI,
Chiemi WATANABE, Yuka ISAMOTO,
Akinori TOMOBAYASHI, Shigenori OTSUKA,
and GFD-Dennou Davis project

Web-based Data and Knowledge Server Software for Geophysical Fluid Sciences,

- Part I: Rationales, Stand-alone Features, and Supporting Knowledge Documentation Linked to Data
- Part II: RESTful Web Services and Object-Oriented Programming Interface

Introduction

Existing web-based data servers

- Web-based data servers in geophysical fluid sciences
  - download data
  - search
  - analysis and visualization with web-browser

Problems of the existing servers

1. limit visualization capability
   – only initial “quick-looks” are possible
2. barrier between server- and client-side operations
   – features of the servers are not available once the data files are downloaded.
3. weak support for non-georeferencing data
4. limit search capability
5. Interdisciplinary and/or collaborative studies

Gfdnavi

- “Gfdnavi” has been developed to solve all the problems
  - web-based data and knowledge server software
  - for application in geophysical fluid science

  (the basic features were introduced in the Part I)

In this talk

- Solution for the problems 1 and 4 (also related to 2)

1. limit visualization capability
   – only initial “quick-looks” are possible
2. barrier between server- and client-side operations
   – features of the servers are not available once the data files are downloaded.
4. limit search capability

- programmability
  - for analysis and visualization capability (prob. 1)
  - for smooth transition between server- and client-side operations (prob .2)
- cross search
  - for search capability (prob. 4)

Programmability

- usefulness in all stages of scientific studies
  - GUI: trial-and-error stages (e.g. quick-look)
  - programming: later stages (e.g. repetition)

- multiple ways of programmability

1. web services and client library
2. downloading a minimum subset of data and scripts
3. registering scripts

Cross search

- search multiple kinds of data in multiple data servers across networks
  - data: observations, numerical simulations, etc
  - servers: personal, group's, public

- for communication between servers

1. web services

Design and Implementations of Gfdnavi Web Services

An example of use case

1. performed several numerical simulations of the future climate with different scenarios for future emission of CO_2
2. analyze and visualize result data of one simulation run with GUI of Gfdnavi web applications (try-and-errors)
   – figure out characteristics of spatial pattern of temperature

- apply the same analysis and visualization to result data of the other runs
  - download a ruby script reproducing the action performed with the GUI
  - modify the script to perform the analysis and visualization with the data of all the runs
- compare the result data and diagrams

- downloaded ruby script

```
1: require "numru/gfdnavi_data“
2: include NumRu
3: t = GfdnaviData.parse("http://example.com/data/T.exp01.nc/T")
4: t mean = t.analysis("mean","longitude")
5: tone = t mean.plot("tone")
6: png = tone.to png
```

- modified script

```
1: require "numru/gfdnavi_data"
2: include NumRu
3: NRUNS = 10 # number of runs
4: pngs = Array.new
5: for n in 0...NRUNS # loop for all the runs
6:   crun = sprintf("%02d", n+1) #=> "01", "02", ...
7:   t = GfdnaviData.parse("http://example.com/data/T.exp"+crun+".nc/T")
8:   t mean = t.analysis("mean","longitude")
9:   tone = t mean.plot("tone")
10:   pngs[n] = tone.to png
11: end
```

- compare the results with those of simulations performed by other researchers
  - use cross search of Gfdnavi to find other simulation data
  - modified the script and apply the same analysis and visualization to these data

RESTful web services

- Resource-oriented architecture
  - similarities with object-oriented programming
    - easy development of ruby client library
- Stateless
  - scalable
  - easier testing than stateful system

Ruby client library

- Similarity with GPhys

  GPhys: a ruby library for analysis and visualization for geophysical fluid sciences

  – enables users to analyze and visualize data on Gfdnavi servers in a manner similar to programming with GPhys at local level

- URL path of dynamic resources

  dynamic resource: generated dynamically as result of operations such as analysis and visualization

  – having correspondence with object-oriented programming

Object and Resource

- object (ruby script)

  a_static_data_object.*operation*(*arg*).to_*type*(*params*)

  – e.g.

  t = GfdnaviData.open("http://host/data/T.nc/T") # static object

  t.analysis("mean"."longitude").to_gphys

  t.analysis("mean","longitude").plot("contour").to_png

- resource (URL path)

  /*a_static_data_path*/*operation*(*arg*).*type*?*params*

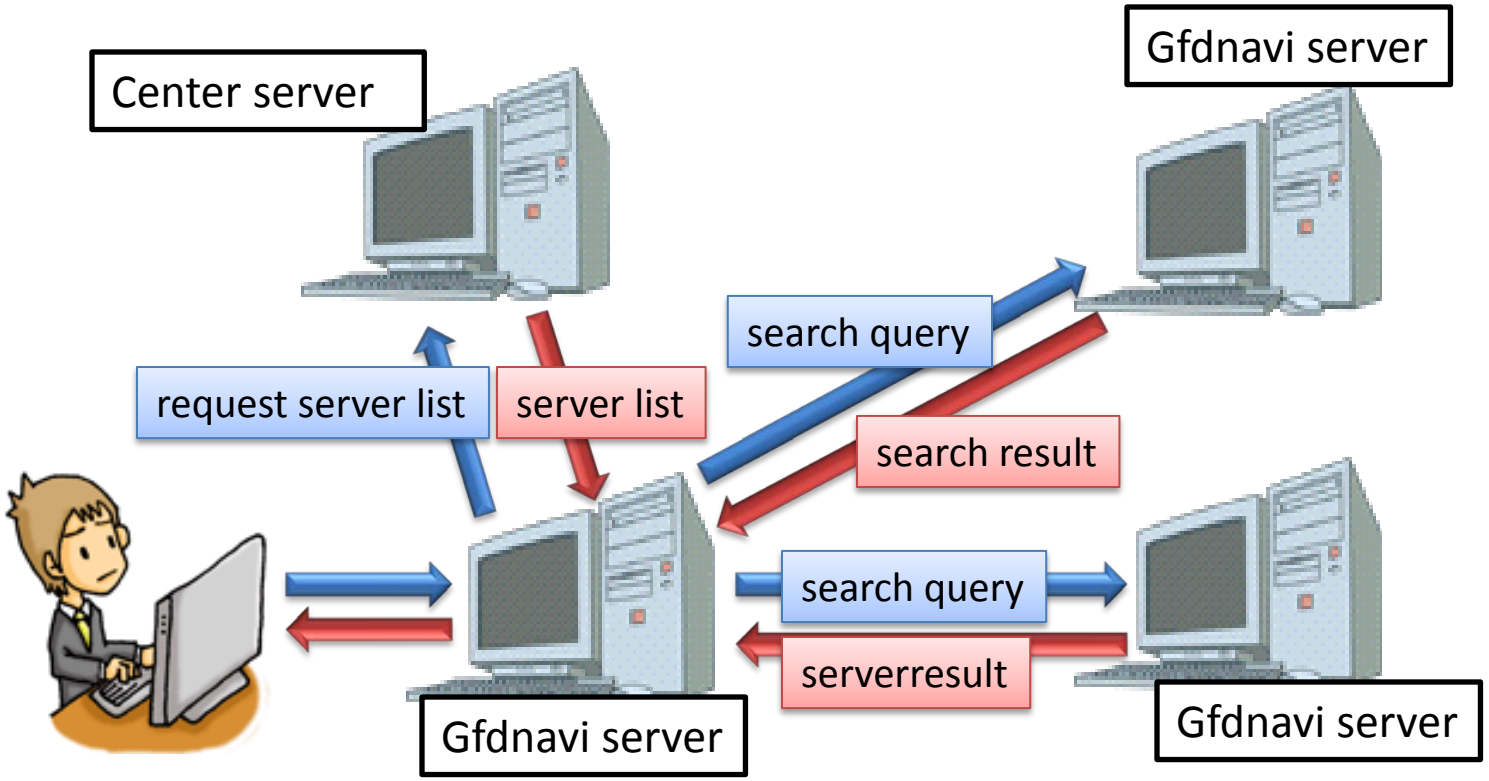
  – e.g.

  /data/T.nc/T/analysis(mean;longitude).gphys

  /data/T.nc/T/analysis(mean;longitude)/plot(contour).png

Hybrid P2P Cross Search

- Hybrid peer-to-peer (P2P)
  - a central server having a server list
  - send search request to each peer



Summary

- network capability
  - programmability
    - smooth transition between GUI and programming
    - web services and client library
  - cross search
    - search across networks
    - web services and hybrid P2P
- RESTful web services and Ruby client library
  - resource-oriented architecture
  - object-oriented programming

Thank you