# Gfdnavi:
# its design and implementation with Ajax and Ruby-on-Rails

Seiya NISHIZAWA,Takeshi HORINOUCHI,
Chiemi WATANABE,
T. KOSHIRO, A. TOMOBAYASHI, S. OTSUKA,
Y. MORIKAWA, Y.-Y. HAYASHI, M. SHIOTANI, and
GFD-Dennou Davis project

# Introduction

- What’s “Gfdnavi”
  - A tool to archive, share, distribute, analyze, and visualize geophysical fluid data and knowledge
  - desktop use to data provide server
  - fundamental technologies
    - Ruby on Rails
    - GPhys

An introduction was done by T. Horinouchi yesterday.

# Ruby on Rails

- an open source web application framework
- written in Ruby
- Model-View-Controller (MVC) architecture
- Convention over Configuration (CoC)
- Don't repeat yourself (DRY)

- swift development
  - ActiveRecord
  - helper methods (HTML, JavaScript, ajax)

# ActiveRecord (AR)

- a part of Rails products
- a ruby implementation of the object-relational mapping pattern

```
node = Node.find(:first, :conditions=>[”id=?”,2])
                         #  SELECT * FROM nodes WHERE id=2 LIMIT 1;
path = node.path   #=> “/samples”
kas = node.keyword_attributes
                         #  SELECT * FROM keyword_attributes WHERE node_id=2;
```

nodes table

| id | path | parent |
|---|---|---|
| 1 | / | null |
| 2 | /samples | 1 |

keyword_attribute table

| id | node_id | keyword | value |
|---|---|---|---|
| 1 | 2 | description | sample directory |
| 2 | 2 | notice | just sumple |

- Do not need to use SQL
  – For better performance, SQL can be used on AR.

# Metadata DB

used for search

1. name-value attributes
2. geospatial- and time-coordinate information
3. owner, groups and access mode
4. link among data
5. time-stamp, size, etc

# 1. Name-Value attributes

- attributes in data file (self-describing files)
  - unified access to attributes in differently formatted files with GPhys

```
gphys_nc = GPhys::IO.open(“fname.nc”,”T”)  # NetCDF
gphys_nc.att_names                          #=> [“long_name”, …]
gphys_nc.get_att(“standard_name”)       #=> “air_temperature”

gphys_grib = GPhys::IO.open(“fname.grib”, “TMP”) # GRIB
gphys_grib .att_names                        #=> [“long_name”, …]
gphys_grib.get_att(“standard_name”)  #=> ”air_temperatrue”
```

- attributes in text file
  - YAML format
  - any name-value attributes

```
description: NCEP/NCAR reanalysis
gfdnavi:
   owner: user1
   other_mode: 0
   rgroups:
   - groupA
   - groupB
```

# YAML

- a human-readable data serialization format
  – easier to read/write than XML

```
puts “Array (list)”
ary = [0,1,2]
puts ary.to_yaml

puts “¥nHash (associative array)”
hash = {“key0”=>”value0”, ”key1”=>”value1”}
puts hash.to_yaml
```

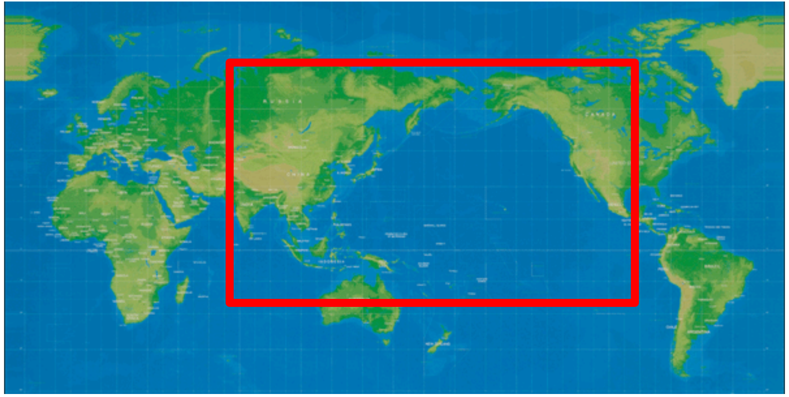
```
Array (list)
---
- 0
- 1
- 2

Hash (associative array)
---
key1: value0
key0: value1
```

Back

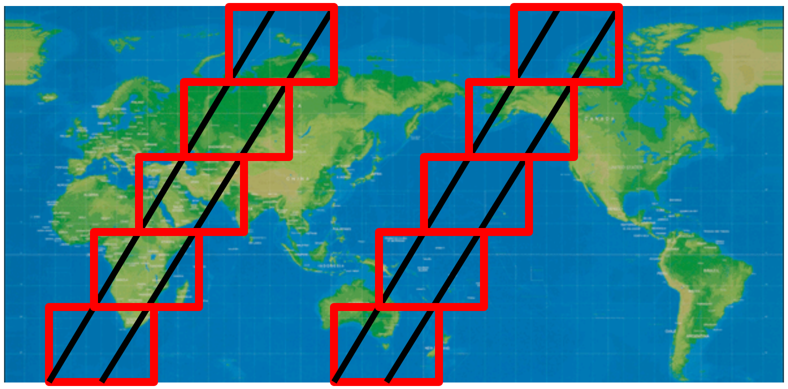
# 2. Geospatial- and time-coordinate information

- spatial region
  - rectangle in longitude-latitude section
- temporal region
  - start time and end time

global, regional, or point



swatch



3. Owner, Groups, and Access mode

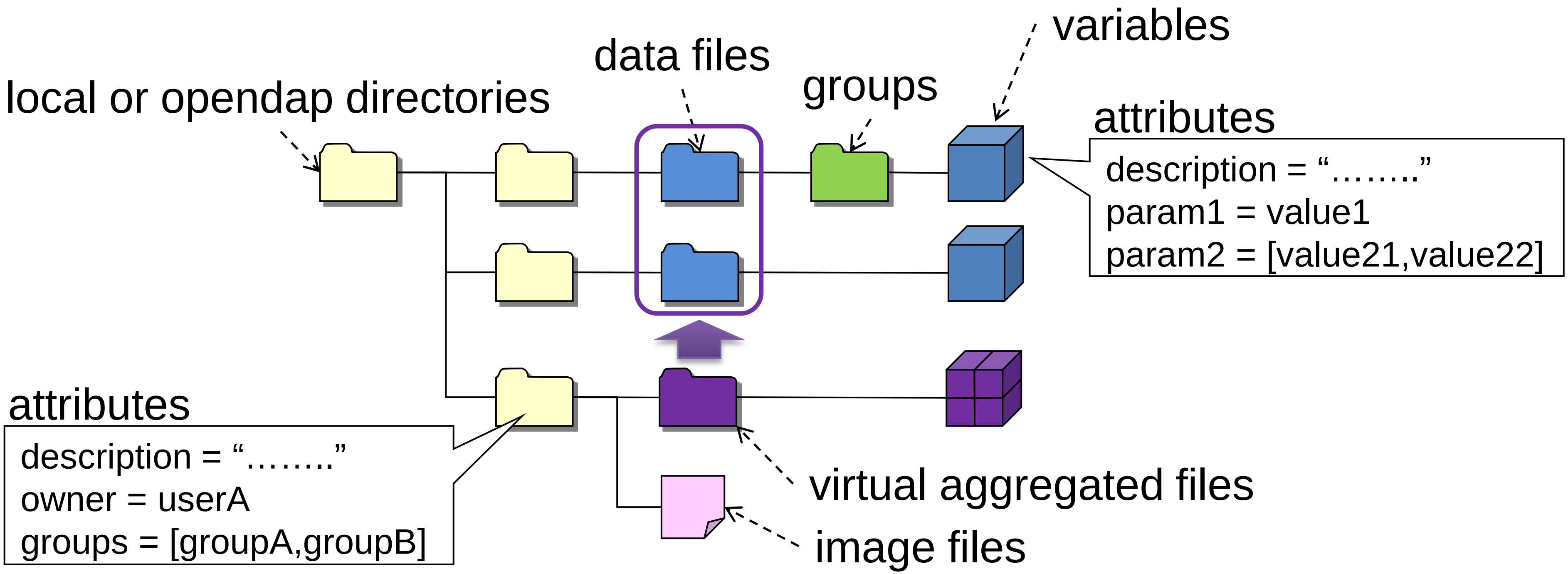
- permission system like i-node
  - readable and writable for groups and others
- Multiple groups are allowed.

4. Link among data

- e.g. This data was calculated from these variables

# Directory tree structure

- nodes in the tree structure
  - node types: directories, variables, images, knowledges, etc
- Each node can have some metadata.
  - inherited to children nodes



local or opendap directories

data files

groups

variables

attributes

description = "……."
param1 = value1
param2 = [value21,value22]

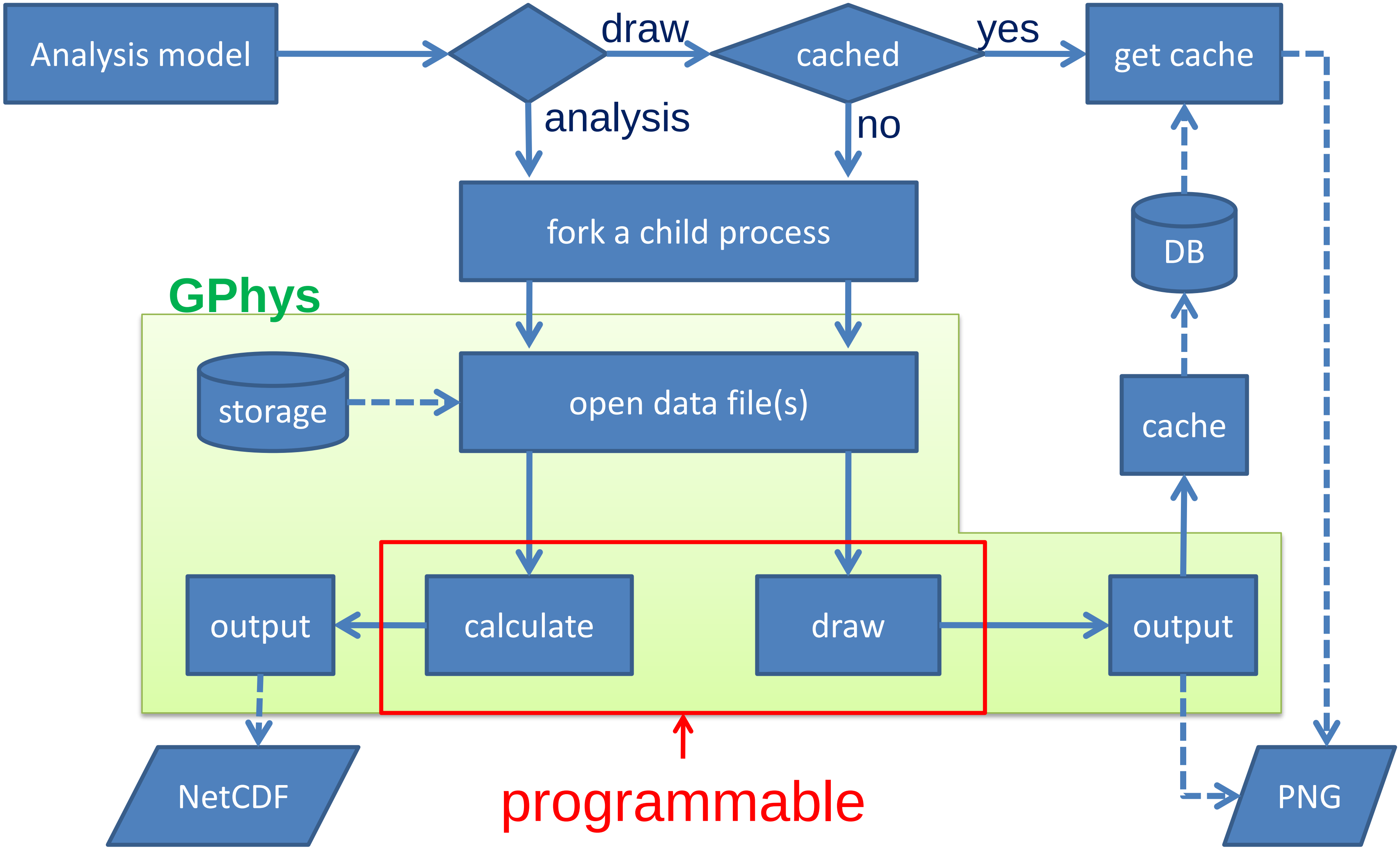
attributes

description = "……."
owner = userA
groups = [groupA,groupB]

virtual aggregated files

image files

# Analysis and Visualization



- Analysis model (Analysis class)
  - all the parameters for analysis or visualization
    - the form in the analysis page
    - instance variables of the Analysis object

  It is able to construct one from the other

    - enable to reconstruct the analysis page from
      - drawn image
      - history list

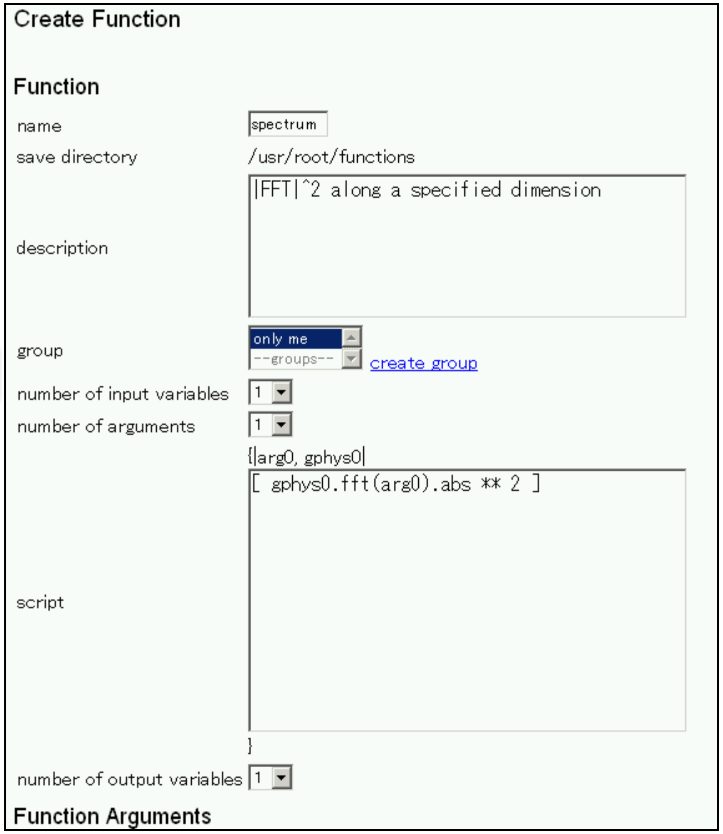
- Draw method and analysis function are not hard-coded.
  - Their definitions are in YAML files (editable)
    - one method in one file

spectrum.yml

```
:name: spectrum
:description: |FFT|^2 along a specific dimension
:nvars: 1
:script: |
  [gphys0.fft(*arg0).abs ** 2]

:arguments:
- :description: the dimensions for spectrum
  :value_type: array_string
  :default: []
```

simple coding due to GPhys

can create and modify via web-browser



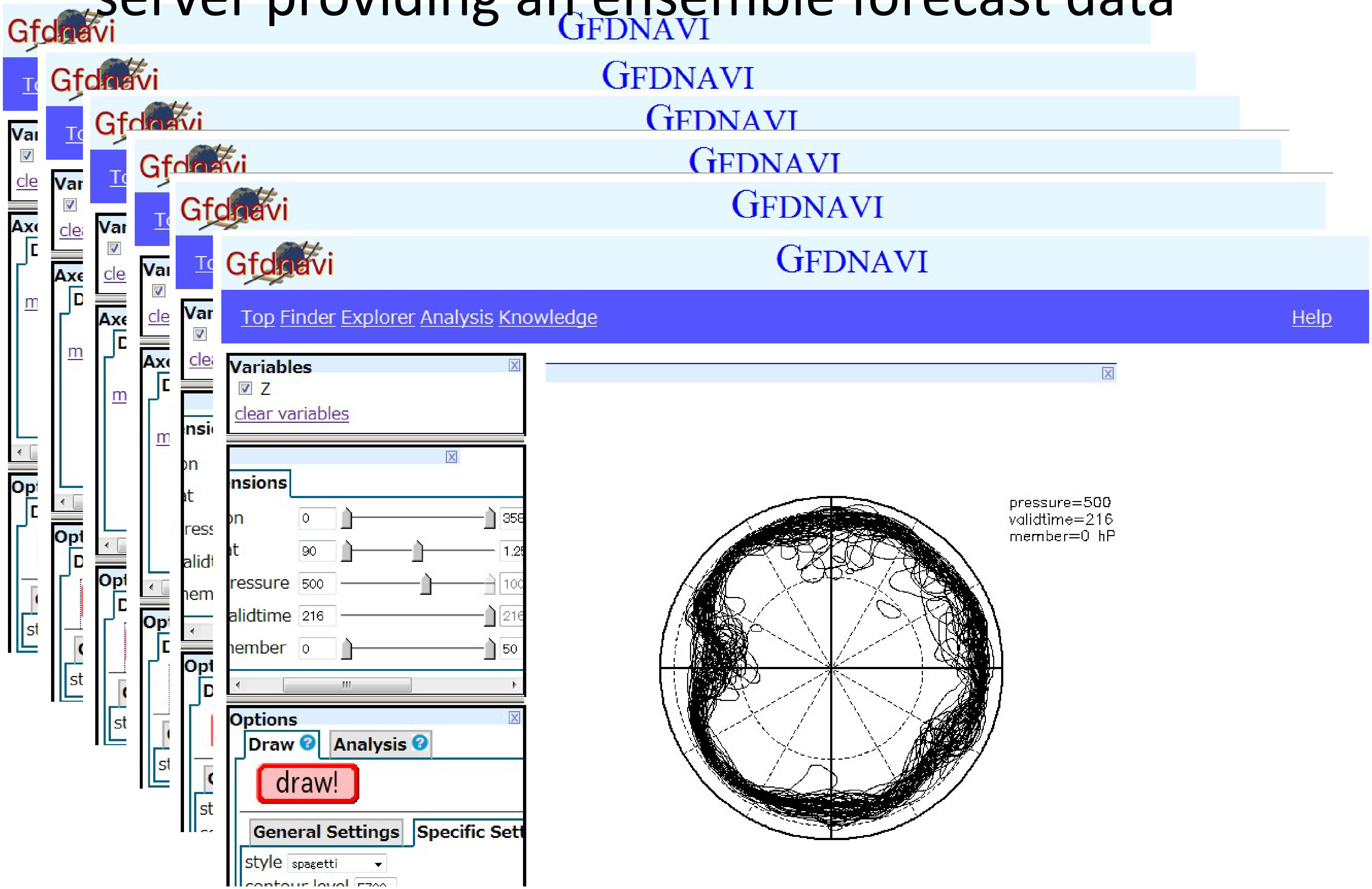
Create Function

**Function**

| | |
|---|---|
| name | spectrum |
| save directory | /usr/root/functions |
| description | \|FFT\|^2 along a specified dimension |
| group | only me<br>--groups--<br>create group |
| number of input variables | 1 |
| number of arguments | 1 |
| | {\|arg0, gphys0\| |
| script | [ gphys0.fft(arg0).abs ** 2 ] |
| | } |
| number of output variables | 1 |

**Function Arguments**

- examples of original draw methods in a Gfdnavi server providing an ensemble forecast data



Variables
☑ Z
clear variables

nsions

on 0 358
at 90 1.25
ressure 500 100
alidtime 216 216
ember 0 50

style spagetti

pressure=500
validtime=216
member=0 hP

# User Interface

- bottleneck of network application
  - network bandwidth
  - machine power and system load of the server

- better usability
  - ajax
    - Rails has many helper methods to write HTML and JavaScript to use ajax.
  - cache

- Animation

# Web service

- local programming
- cross-site use
  - other Gfdnavi servers
  - non-Gfdnavi servers

➢ SOAP

  ➢ APIs for all the analysis functions and draw methods

    ➢ use the Analysis class

  ➢ WSDL

➢ (REST)

# Summary

- Metadata and Directory tree structure
  - attributes in self-describing data files and YAML files
  - inheritance
  - ➢unified access to attributes with GPhys
  - ➢easy and swift development with AcriveRecord (Rails)
- Analysis/Visualization
  - programmable (with text editor or web-browser)
  - ➢easy and extensible coding with GPhys
- User Interface
  - good usability with ajax and cache
  - ➢easy development with helper methods (Rails)

# Thank you