QIDW Update

Annual QIBA Meeting
Thursday, May 18, 2017

Current Status: Profile Development

- Over the Past Year...
  - Complete Migration to the new platform. Better performance for large data sets
  - Started to store human data sets (including review for PHI)
  - Evaluated NITRC as a possible model for function beyond 'data warehouse'
Current Status: Profile Development

- New Data Sets
  - US. Shearwave data sets
  - CT Vol BC. Now hosts data sets used for conformance testing
  - CT Lung Density. Hosting data sets for COPD and density conformance testing

Expected New Data Sets

- NM
  - Amyloid?
- FDG/PET/CT test retest trial
- CT
- MR
- US.
  - Code is posted on github. Should QIDW host ‘stable’?
Website Friendliness

- In past, naming was largely up to data set creator
- Now working to uniform naming / directory structure:
  - Modality (CT, MR, NM, US)
  - Project / Profile
  - Private Data / Public Data / Software / Validation

Next Steps (If Any)

- Girder / Resonant advertises visualization

The Resonant data and analytics platform

WELCOME TO YOUR DATA

Data Management
Upload, share, and manage your data from SQL, NoSQL, Hadoop, Amazon S3, and more.

Analytics
Perform heavy-lifting on your data with Python, R, Spark, and Docker containers through a uniform interface.

Visualization
Gain insight with flexible, scalable web visualizations.

The Resonant framework tools work in concert to provide storage, analysis, and visualization solutions for your data. All Resonant components are fully open-source under the Apache v2 licence.
Next Steps (If Any)

• Girder / Resonant also describes workflow support

Analytics

Girder Worker
Girder integrates with Girder Worker, a flexible execution engine for running analyses. To see it in action, install Girder Worker and follow our tutorials on image analysis and graph analysis. To run Girder Worker jobs through Girder, see how to enable the Girder Worker plugin.

Girder Worker: A simple, flexible execution engine

What is Girder Worker?
Girder Worker is a Python application for generic task execution. It can be run within a Celery worker to provide a distributed batch job execution platform.

The application can run tasks in a variety of languages and environments, including Python, R, Spark, and Docker, all via a single Python or Celery broker interface. Tasks can be chained together into workflows, and these workflows can actually span multiple languages and environments seamlessly. Data flowing between tasks can be automatically converted into a format understandable in the target environment. For example, a Python object from a Python task can be automatically converted into an R object for an R task at the next stage of a pipeline.

Example Workflow

```python
subtract_image = {
    'inputs': {
        'value': {'type': 'image', 'format': 'nifti'},
        'input': {'type': 'image', 'format': 'nifti'}
    },
    'outputs': {
        'value': {'type': 'image', 'format': 'nifti'}
    }
}
```

Now another task will compute the average pixel value of the input image.

```python
mean_image = {
    'inputs': {
        'mean_input': {'type': 'image', 'format': 'nifti'},
        'input': {'type': 'image', 'format': 'nifti'}
    },
    'outputs': {
        'mean_value': {'type': 'number', 'format': 'number'},
        'input': {'type': 'image', 'format': 'nifti'}
    }
}
```

Finally, let’s add all of the tasks to a new workflow and make the appropriate connections.

```python
wf = Workflow()
wf.add_task(subtract_image, 'blur')
wf.add_task(subtract_image, 'blur2')
wf.add_task(subtract_image, 'blur3')
wf.add_task(mean_image, 'mean')
wf.add_task(mean_image, 'mean2')
wf.add_task(mean_image, 'mean3')
wf.add_task(mean_image, 'mean4')
```
Next Steps

- Do we need workflow support?
- If so, find example project and
  - Find out if Kitware must develop workflow, or
  - RSNA/Project lead can develop workflow