

# An ASDC Subsetter Concept

Jeff Walter

October 16, 2009

# Why are we here?

- ASDC is committed to providing data subsetting and reformatting services to users.
- The existing MISR system is aging, unscalable, and difficult to maintain and needs to be replaced.
- The CERES and CALIPSO projects have both identified this functionality as a high priority.

# How do we get there?

- Rather than create a separate system for each project, it is more efficient from the perspective of implementation, operations, and maintenance to have a single system that manages this functionality for all projects.
- Required system characteristics:
  - Modular – to allow different components, developed by different parties, to be integrated with little or no impact.
  - Scalable – to easily accommodate increasing demand or load by simply adding physical resources on the back end.
  - Extensible – to add new functionality or support additional projects without affecting existing functionality.

# How do we get there?

- Required system functionality
  - Receive user requests
  - Acquire the appropriate data from the correct location
  - Execute the desired processing of the data (subsetting and/or reformatting).
  - Distribute the processed results to the user in the requested manner (ftp push/pull, etc.)
- Some high level functions will be common across all projects and some of them will be project specific.
- In this context, a software workflow approach makes sense.

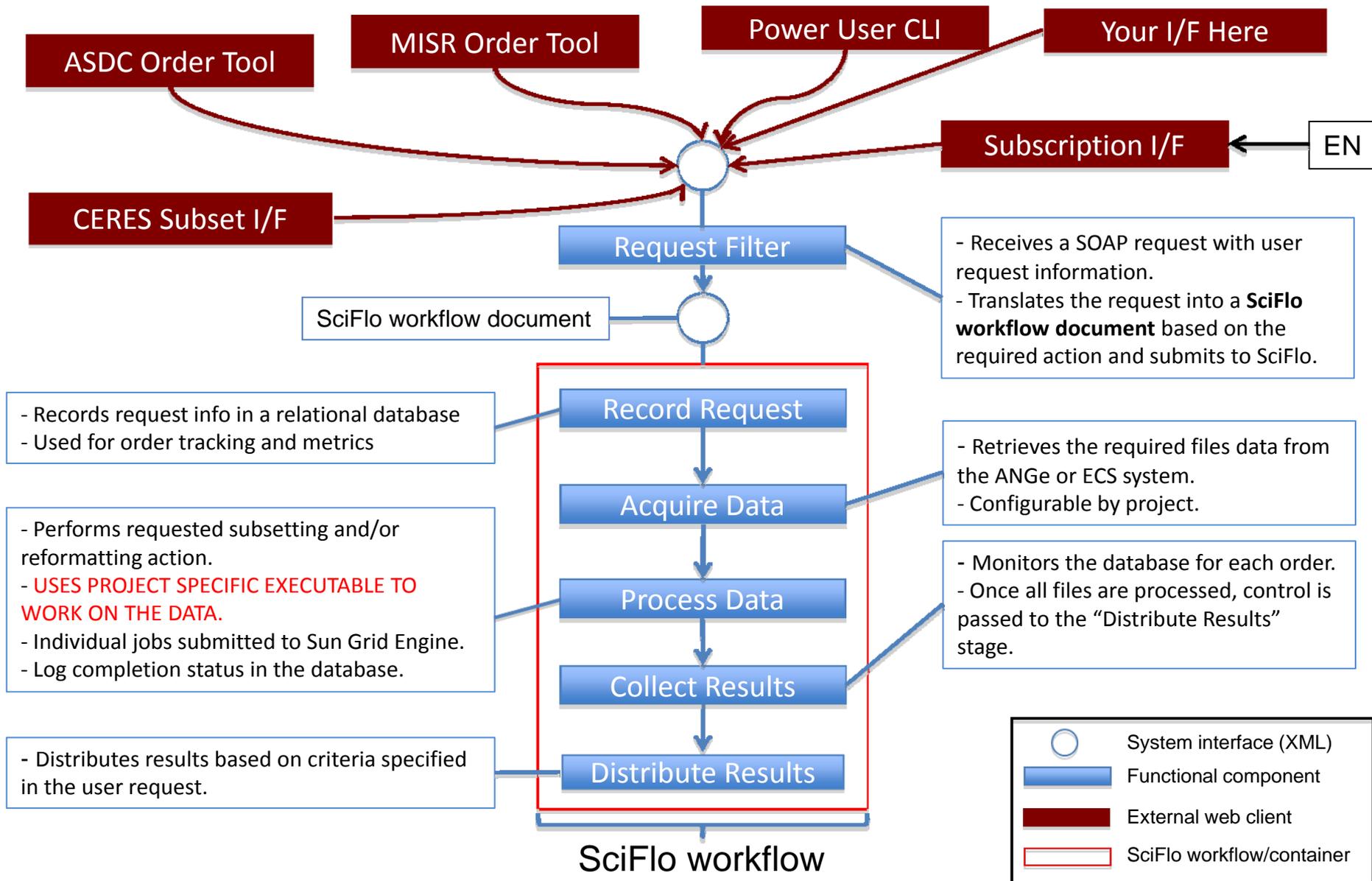
# SciFlo Software

- SciFlo is a scientific data flow engine designed to pull together data and services from remote locations into a single workflow execution.
- Developed at JPL under an ACCESS grant.
- It has multiple capabilities but we are primarily interested in the workflow management functionality and the SOAP capabilities.
- Currently used by the AMAPS system that is now operational at ASDC.
- It is suitable for this project because:
  - It has a relatively simple XML syntax and keyword namespace defined for creating workflows.
  - It is reasonably lightweight (implemented in Python).
  - It has the capability to stitch together independent executables developed in different languages by different parties into workflows.

# SciFlo Software

- ASDC will conduct a trade study to explore other workflow management software.
- SciFlo will be used for the prototype.
- If another workflow management system is selected, it would have minimal or no impact to any underlying software that is developed because the individual pieces are decoupled from the system.

# Component-and-Connector View



# Web Tool Interface

- Ordering tools will submit a SOAP/XML request with the following information
  - Project name (CERES, CALIPSO, MISR, etc.)
  - File Information
    - File names
    - Data type
    - Subsetting/reformatting parameters, which can include any combination of the following:
      - ✓ Temporal range
      - ✓ Spatial range
      - ✓ Parameter list
      - ✓ Output format (HDF, netCDF, etc.)
  - Distribution Information
    - User profile
    - Data delivery information
      - Delivery method (FTP push or pull, scp, etc.)
      - Host name, user id, password, directory, etc.

# Web Tool Interface

- Alternatively, an interactive web tool such as an “on-the-fly” browse generator would have a different workflow defined with some different stages.
  - Would still need “Acquire Data” and “Process Data” but may not need “Record Request” or “Distribute Results”.
- Different predefined workflow templates would be created.
- The “Request Filter” stage would select the appropriate workflow template based on information provided by the individual web tools, fill in the relevant information, and submit to SciFlo for execution.

# Process Data Stage

- 1) Parse input and select appropriate project-specific top level executable (MISR, CERES, CALIPSO, etc).
- 2) Submit to Sun Grid Engine for execution
  - a) Call project-specific top level executable
  - b) Capture exit code
  - c) Update database

Top level executable interface

## Top level executable

- Serves as a wrapper for core functions
- Uses a generic interface to receive input
- Translates input into appropriate calls

Core  
subsetting/reformat  
ting libraries,  
executables, etc.



# Project-Specific Executables

- Developed and delivered by the individual project teams.
- Having a top level wrapper allows project teams to adhere to the common system interface while giving them the freedom to implement the core subsetting/reformatting functionality in the way that makes the most sense for their individual data sets.
- Preferred programming languages, required third party software, etc. can be negotiated and worked out without impacting the overall system.

# Top Level Executable System Interface

- The easiest thing to do is use command line arguments in a “parameter=value” type of structure like the following:

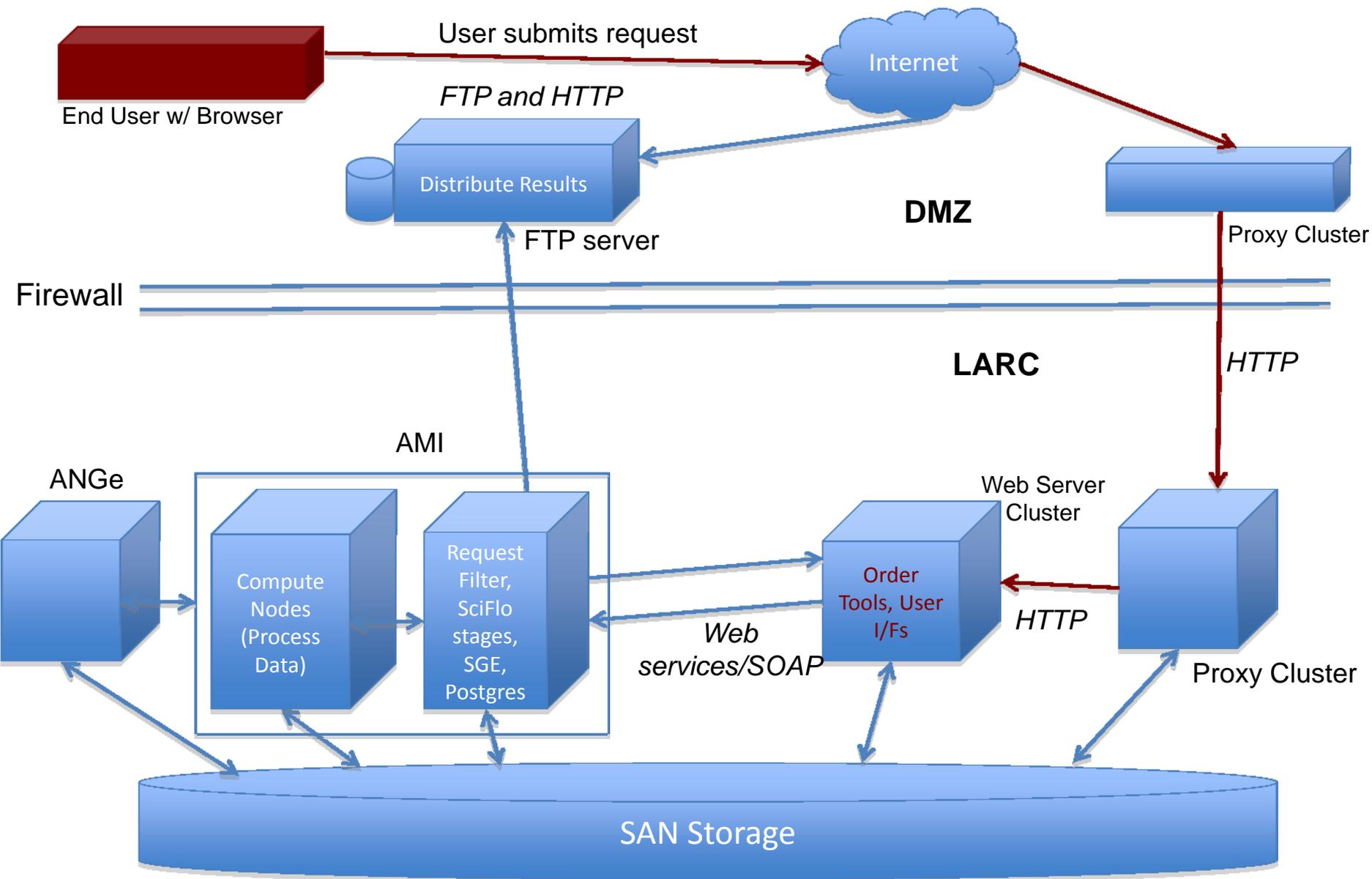
- file\_list=“.....”
- temporal=“.....”
- spatial=“.....”
- parameters=“.....”
- format=“.....”

- Information in quotes can be formatted in whatever way the developer sees fit.  
- If a particular field is not relevant to the request being processed, the parameter will still be provided but the value will be left blank.

- Example

- `misr_subset.py file_list="/home/walter/MISR_aerosol_file.hdf" temporal="" spatial="StartBlock=50,EndBlock=80" parameters="optical_depth,angstrom_exponent" format="netCDF"`

# Allocation View



# Conclusion

- Some advantages of this approach
  - Relatively simple
  - Modular - emphasizes separation of concerns
  - Able to integrate and support multiple projects by abstracting away the underlying project-specific implementation details
  - Scalability – can accommodate increased load by simply adding additional physical resources.
  - Extensibility – can easily add subsetting support for new projects or add other new functionality by creating new workflows.

# Backup Slides

# SciFlo Workflow Example

