# Data Services in Support of High Performance Computing-Based Distributed Hydrologic Models

David G. Tarboton[1], Jeffery S. Horsburgh[1], Pabitra Dash[1], Tseganeh Gichamo[1], Ahmet Yildirim[1], Norm Jones[2]
[1]Utah State University, [2]Brigham Young University
Contact: dtarb@usu.edu Learn more http://ci-water.org/

CI-WATER A Utah-Wyoming Cyberinfrastructure Water Modeling Collaboration

H13H-1211

EPSCoR EPS 1135482

## Project Goals

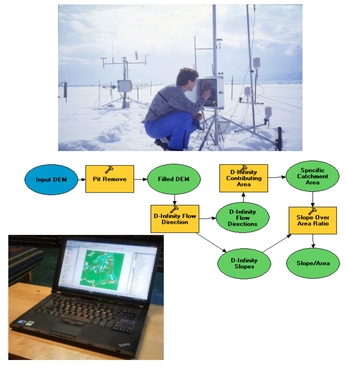
1. Provide hydrologic researchers, modelers, water managers, and users access to High Performance Computing (HPC) resources without requiring them to become HPC experts.
2. Reduce the amount of time and effort spent in finding and organizing the data required to execute hydrologic models.

## Assumptions

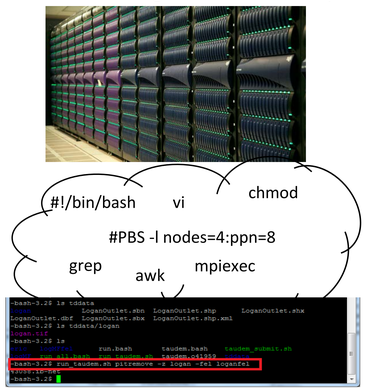
1. Research hydrologic modelers should be comfortable using a scientific programming language like Python or R.
2. Hydrologic modelers are not expert in HPC systems and learning this is a barrier to the use of HPC.
3. Hydrologic modeling is data intensive (large datasets from a range of sources)

## A digital divide

Hydrologic Experimentation and Modeling



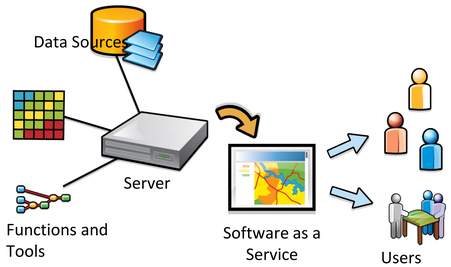

Data Intensive High Performance Computing



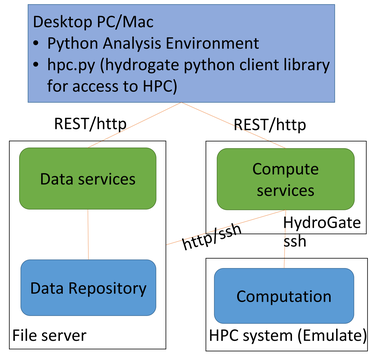

**Do you have the access or know how to take advantage of advanced computing capability?**

## Objective

Deliver hydrologic model input preparation, execution, and analysis as a service over the web




## Architecture




## hpc.py Python Client Library functions

- Subset Raster (DEM) given bounds (using GDAL)
- Projection (using GDAL)
- Generate watershed given outlet (using TauDEM)
- Convert raster to NetCDF
- Compute Slope (using GDAL)
- Compute Aspect (using GDAL)
- Get Canopy Variables (for UEB snowmelt model)
- Get Daymet
- Upload/download
- Upload package to HPC
- Submit HPC Job

## Data Services for Western US

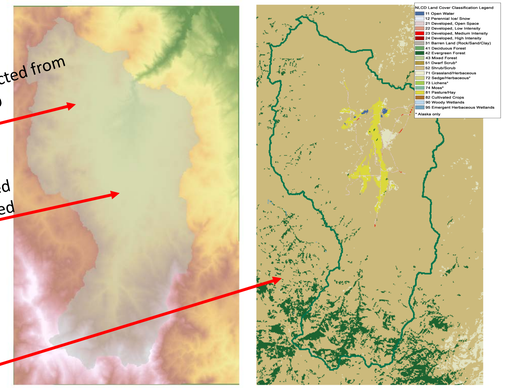
- National Elevation Dataset Digital Elevation Model (DEM) (Host local copy for performance)
- National Land Cover Dataset (Host local copy for performance)
- Derived Terrain variables generated on the fly following projection and area specification)
  - Slope
  - Aspect
  - Flow directions
  - Contributing area
- Weather and Climate Data
  - Daymet ORNL/NASA Daily Surface meteorology (Host and periodically update)
  - NASA Land Data Assimilation System (retrieve and package on demand)

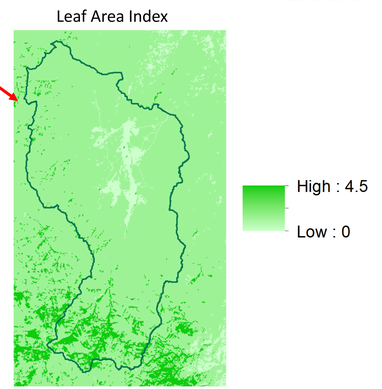
## Example for preparation of input to snowmelt model

```
from hpc import Client

client = Client()

workingDir = r"C:/Users/dtarb/Desktop/HydroGateDemo/"
# Bounding box
subsetDEM_request = client.subset_dem(-116.87, 43.33, -116.65, 43.05)  # Reynolds Creek -116.87, 43.33, -116.65, 43.05
projectDEM_request = client.project_raster_to_UTM_NAD83(input_raster_url_path=subsetDEM_request.file_path, utm=11)

# Approximate outlet point
outlet_shapefile_request = client.create_outlet_shapefile(point_x=-116.749, point_y=43.267)
projectShp_request = client.project_shapefile_to_UTM_NAD83(outlet_shapefile_request.file_path, 11)

delineation_request = client.delineate_watershed(projectDEM_request.file_path,
                                                 projectShp_request.file_path, 11,8000)
watershed_file_path = delineation_request.file_path
watershedFile = workingDir + 'ReynoldsWS.nc'
rastertoNC_request = client.raster_to_netcdf(watershed_file_path, save_as=watershedFile)
aspectFile = workingDir+'RCAspect.nc'
computeAspect_request = client.raster_aspect(projectDEM_request.file_path)
rastertoNC_request = client.raster_to_netcdf(computeAspect_request.file_path, save_as=aspectFile)
slopeFile = workingDir+'RCSlope.nc'
computeSlope_request = client.raster_slope(projectDEM_request.file_path)
rastertoNC_request = client.raster_to_netcdf(computeSlope_request.file_path, save_as=slopeFile)
subsetNLCD_request = client.subset_NLCD_to_reference_raster(watershed_file_path)
canopy_var_file = workingDir + 'canopy.zip'
getCanopyVars_request = client.get_canopy_variables(subsetNLCD_request.file_path, save_as=canopy_var_file)
```

REST Call for projection
'http://129.123.41.158:8080/projectrasterutmnad83?raster=dem307455739
8985481243.tif&utm=11'

REST Call for watershed delineation
http://129.123.41.158:8080/delineatewatershed?shp=shape9aaa1d17cd7b4
882b6e8017637bc9f80.shp&raster=dem5acd0feed7eb410d894cdc2f95a12
4e2.tif&threshold=8000&utm=11'




## Take Home

- Preliminary software as a service and data as a service functionality in support of hydrologic modeling has been demonstrated
- Accommodates large datasets without having to download the data (work in the cloud)
- Reduces software installation requirements
- Enables use of HPC resources without having to be an HPC expert

## HydroGate HPC CGI Based REST web services HPC Gateway

- Abstract away details and complexities involved in using HPC systems
- RESTful web APIs with data passed in JSON-encoded format over standard HTTP methods (PUT, POST, DELETE, GET)
- Security using token-based authentication to the HydroGate service, and then SSH-based communication with the HPC centers
- File transfer using secure copy (scp)
- Submission of jobs to a designated HPC center under user credentials
- Monitoring of job status by means of a URL callback mechanism
- Automatic batch script generation on the HPC centers from JSON-encoded job description language
- Discovery functions to determine the capabilities of HPC centers installed programs and program parameters

Why not WPS?

- Lower level capabilities needed to manage multiple queuing systems to enable asynchronous invocation of web service APIs
- Token based authentication has low overhead. Access to requestors low-level information such as IP address used to bind requests for better security
- WPS key-value-pair (KVP) and XML-encoding less efficient and general that JSON over standard HTTP methods (PUT, POST, DELETE, GET).

## Java NIO based HydroGate Data services on File server

- Taking advantage of JAVA NIO to scale to thousands of users for a data service
- Allowing users to upload file to the data server and execute computationally non-intensive operations on the data server to avoid data transfer time