A System for Scripted Data Analysis at Remote Data Centers
Daniel L. Wang1, Charles S. Zender2, and Stephen F. Jincks1
1Department of Electrical Engineering and Computer Science, 2Department of Earth System Science
University of California at Irvine

1. Data Analysis + Data Access
We have implemented the Scripted Workflow Analysis for Multi-Processing (SWAMP) system, which
makes larger scale end-user data analysis possible. A client component transmits queries and
scripts to a remote computing component, which porably, safely, and efficiently executes these
queries and scripts with final results. SWAMP is designed to be installed by data publishers in
data centers as well as scientists in lab group servers. By eliminating the tedious and lengthy
interaction process, SWAMP allows end-users to achieve higher performance optimization in the
computing service, the system drives the overhead of data processing down to levels where grand-
styles and terabytes can be practically processed daily.

2. Scripts: Ease, Flexibility, Performance
A shell scripting syntax was chosen for three key reasons:
Shell scripts are already widely used by scientists to automate data analysis and specify
computation in grid and batch processing systems. The same language is used in the
command line, which is already known.
Shell scripts can flexibly combine sequences of commands into rich and complex, parallel scripts
that can be also simple and express special tasks succinctly.
SWAMP is able to parse and optimize shell scripts for parallelism and I/O optimization, through
the use of techniques traditionally reserved for languages designed for computation. An
additional effort is required.
SWAMP brings grid shell execution similar to [1] to non-computer scientists by running
existing pipelines instead of rewriting ones. Please see [2] for additional details on shell
compilation and execution traits [3].

3. Command Line Operators
Shell compilation requires semantic knowledge of command meaning, but supporting all
argument formats is an intractable task. SWAMP has selected the netCDF Operators (NCO) [4],
along with a handful of common shell helpers. This enables expression of both quick workflow
analysis and larger data- and node-scale operations on data in netCDF, while permitting compilation
and optimization.

4. Features
Interface
Common shell syntax, including for-loops, if-branching, and variables.
NCO command sets—allow rich data reduction analyses (expandable).
Substitution of interdependencies: inputs, intermediates, and outputs—without special syntax.
SCOP API: enhanced interoperability.
Service and Performance
Parallelist parallelism in systems with multiple cores, multiple CPUs, and compute clusters.
Shell compilation and optimization—plug and network I/O for efficiency.
Saves bandwidth—transfers final results only.
Simple logging—the data is being used and analyzed.
Overall speed—ranges from 1X to 100X. Worst case slowdown is ~10X.

The SWAMP System
This figure provides a high-level look at SWAMP, illustrating the difference in working with SWAMP (above) and
working with SWAMP (below).

System Architecture
Block Diagram

Exploitation of Parallelism

<table>
<thead>
<tr>
<th>Method</th>
<th>Database</th>
<th>Requires</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>thread</td>
<td>multiprocessor</td>
<td>programmer threads, OpenMP</td>
<td>ptthread_create(...)</td>
</tr>
<tr>
<td>process</td>
<td>multiprocessors, compute clusters</td>
<td>programmer forks, MPI</td>
<td>ptthread_create(...)</td>
</tr>
<tr>
<td>command</td>
<td>multiprocessors, compute clusters</td>
<td>SWAMP or equivalent script: script.</td>
<td>nwcc ccm2000.daily.nc ccm2000.nc</td>
</tr>
</tbody>
</table>

Example Script
The system supports scripts both simple and complex. With usage similar to traditional shell script execution and a
small execution overhead at worst, SWAMP is suitable for existing daily analyses while making more costly analyses also practical.

Simple script (ccm2000)

```bash
cd /home/swamp/client/ccm2000
```

Simple execution

```bash
python swamp-client.py ccm2000
```

This work is supported by the National Science Foundation under grants ATM-0231193 and IOS-0431203.

http://swamp.googlecode.com

5. Performance
On a single 4-way node, SWAMP returned results approximately 5X faster (see [2]). The results
below illustrate clustered computational performance, which show promise, but performance
varies from the benchmark’s I/O dependence. Tests were conducted on a cluster with
compute-nodes and one head-node.

Bandwidth savings in SWAMP from downloading results instead of inputs

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Input size</th>
<th>Time size</th>
<th>Output size</th>
<th>Savings</th>
</tr>
</thead>
<tbody>
<tr>
<td>nwcc</td>
<td>0.75MB</td>
<td>27.9MB</td>
<td>0.75MB</td>
<td>99.99%</td>
</tr>
</tbody>
</table>

6. Software Requirements
System requirements for SWAMP are modded for both server and client components.

Client
Python 2.4+ with SciPy

Server
Python 2.4+ with twisted.web (http://twistedmatrix.com)
netCDF Operators (http://mcs.sourceforge.net)

7. Summary
Geosciences datasets too large for full download—SWAMP provides source-side data reduction
and analysis. Shell script interface allows grid workflow execution with none of the hassle.
Avoidance of input download makes network bandwidth largely irrelevant—large analyses now
possible.

SWAMP scales from uniprocessors to multiprocessors to clusters.

8. Future Work
Integration with Grid Engine: Dynamic, on-demand allocation of compute nodes in response
to changing computational load
Better performance for complex scripts by coarser-grained work partitioning/distribution.
Support for workflows operating on data at multiple sites.

References

http://swamp.googlecode.com