How to use the task-parallel OMWG/ AMWG diagnostic packages

Sheri Mickelson, Robert Jacob, Michael Wilde
Argonne National Laboratory

Dave Brown
NCAR

17th Annual CESM Workshop, Breckenridge, Colorado
Tuesday, 19 June 2012

This work is part of the parVis project which is sponsored by the Earth System Modeling program of the Office of Biological and Environmental Research of the U.S. Department of Energy's Office of Science

parVis

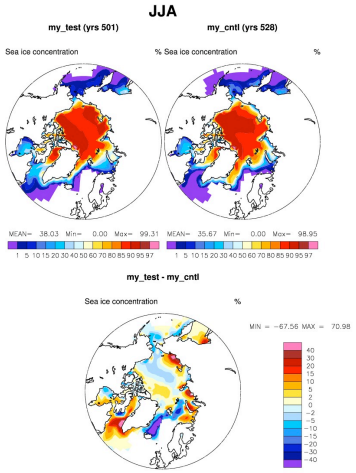
U.S. DEPARTMENT OF ENERGY | Office of Science

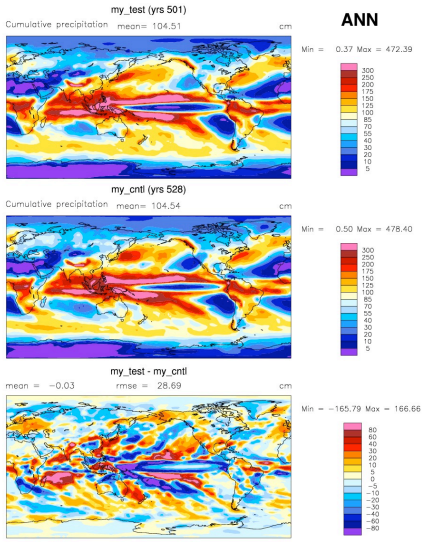
Topics

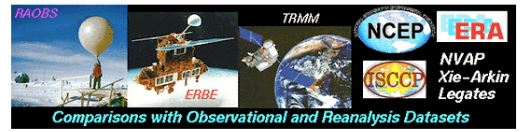
- Overview of AMWG/OMWG diagnostics
- Installing the AMWG/OMWG
- Basic usage of OMWG/AMWG
- Overview of Swift
- Installing Swift
- Using the Swift version of the diagnostic packages
- Demo
- More info on Swift configuration files
- More info on Swift (if time and interest)

AMWG Diagnostic Package

- Used to post-process output from the Community Atmosphere Model (CAM)
- Uses NCO to create climate average files
- Uses NCL to create over 1,000 plots and tables
- Controlled by a top level shell script







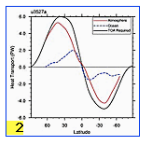
Mark's Master Webpage AMWG Diagnostics Package

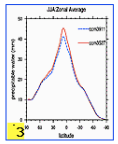
b30.004

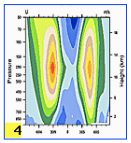
Set Description

1 Tables of ANN, DJF, JJA, global and regional means and RMSE.
2 Line plots of annual implied northward transports.
3 Line plots of DJF, JJA and ANN zonal means
4 Vertical contour plots of DJF, JJA and ANN zonal means
5 Horizontal contour plots of DJF, JJA and ANN means
6 Horizontal vector plots of DJF, JJA and ANN means
7 Polar contour and vector plots of DJF, JJA and ANN means
8 Annual cycle contour plots of zonal means
9 Horizontal contour plots of DJF-JJA differences
10 Annual cycle line plots of global means
11 Pacific annual cycle, Scatter plot plots
12 Vertical profile plots from 17 selected stations
13 ISCCP cloud simulator plots

Click on Plot Type

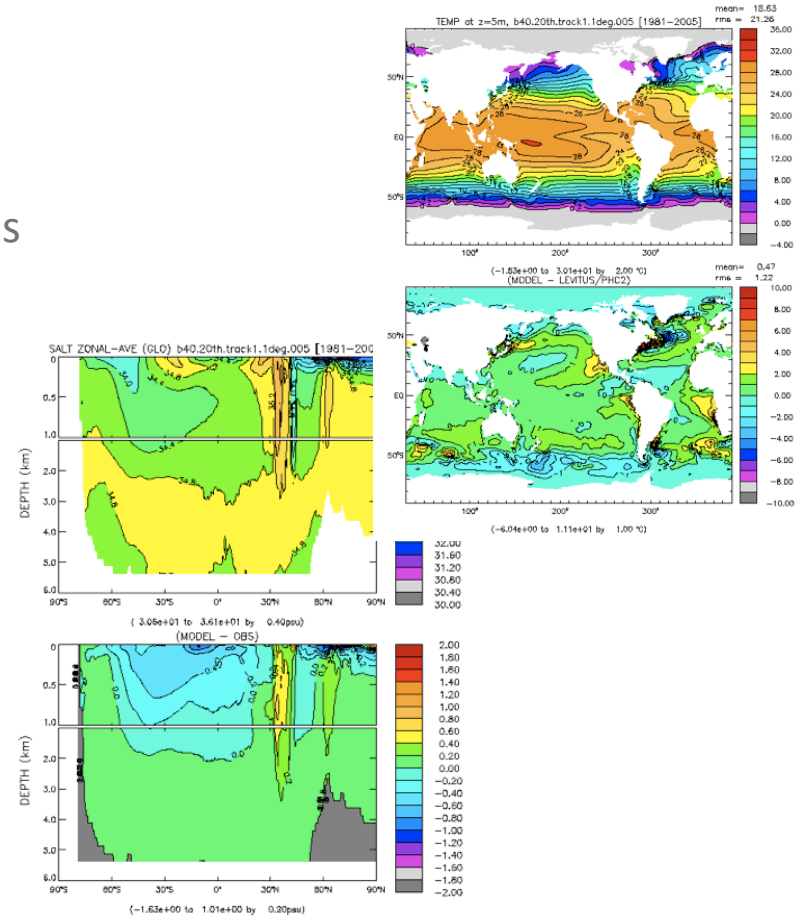






OMWG Diagnostic Package

- Consists of three different scripts, each ran independently
  - Compares against observations
  - Compares against another model run
  - Creates time series of model output
- Uses NCO to create the climate average files
- Older version uses IDL to create plots
- New version uses NCL to create the plots
- Relies heavily on HPSS



- Swift is a parallel scripting system for Grids and clusters
- Swift is easy to write: simple high-level scripting language
  - Looks like C, feels like Python, everything runs in parallel
- Swift is easy to install and run - just needs Java
- Swift is fast: Karajan provides a powerful, efficient, scalable, and flexible execution engine.

For downloads, docs, tutorials, and usage examples:

http://www.ci.uchicago.edu/swift

Swift's task-parallelization is driven by data dependencies

Parallelism is automatic and pervasive in Swift:

```
j = f(i);   // f() and g()...
k = g(i);   // are computed in parallel
r = j + k;  // r is set when they are done
```

All iterations of this loop are computed in parallel:

```
file observations[] …
foreach obs,i in observations {
  inv[i] = invert(obs);
}
```

Setup and Running the Two Packages

Similarities to the Non-Swift Versions

- Modify top level scripts as you would the original versions
- You will need NCO, NCL, and convert
- Run package just like the original

Differences from the Non-Swift Versions

- A couple of variables in the top level scripts were added
- You will need to download Swift or add it to your path
- You will need a current version of Java
- Edit the Swift configuration files for your environment
  - Usually will just involve a few simple changes

AMWG Diagnostic Instructions

…can be found in swift/README-swift or https://trac.mcs.anl.gov/projects/parvis/wiki/SwiftWork/SwiftAtmGeneralInstructions

1. Download the most recent version of the diagnostic package or locate local copy
2. Create a run directory. Then from this directory create a directory called swift
3. Copy top level script to the run directory
4. Copy swift/amwg_stats.swift to the swift directory
5. Copy fs.data, cf.properties, sites.xml, and tc.data from swift/conf/ to your run directory
6. Edit the top level script as you would normally.
7. In the top level script also set use_swift to 0 (on)
8. You will also have to set swift_scratch_dir. This directory will store Swift specific files and can grow quite large.
9. Create the set swift_scratch_dir directory.
10. You will need to edit the sites.xml and tc.data files (will be discussed later in the talk)
11. Run the package as you normally would (./diag*.csh) regardless if you're running on a batch machine or not

OMWG Diagnostic Package Instructions

This Swift version of the package is able to run with data either located locally or stored on hpss. It also requires NCL version 6.1.0 to create all of the plots.

1. Create a new $RUNDIR
2. From the OMWG installation directory copy any popdiag*.csh files you'd like to run and their matching popdiag.*swift file(s)
3. Also copy over the swift configuration files found in swift_config/<mach> to $RUNDIR
4. Create a working directory. This directory will contain the output from the package and the Swift temporary files. This should be in a large scratch space.
5. In the main top-level script set SWIFT to 1 (on)
6. Edit (CNTRL)MSROOT to point to the root location of the monthly output files on hpss or the exact path if the data is local.
7. Set WORKDIR to be the full path to the directory you created in step 4
8. Set RUNDIR to be the full path the directory you created in step 1.

OMWG Diagnostic Package Instructions cont.

9. For popdiagts.csh you will also have to edit CPLLOGFILEPATH, OCNLOGFILEPATH, and DTFILEPATH to the full path for either location on hpss or locally. Make sure you set MSLOGREAD to let Swift know where the files are located.
10. Make sure the zonal average tool (za) is in your path. Or you can set the direct path in swiftUtils/za.csh.
11. Edit both tc.data and sites.xml
12. Execute the script as you normally would (./popdiag*.csh)

tc.data (-tc.file parameter)

- Lists all of the executables/scripts that the Swift script will run.
- The fields in the list must be separated by tabs, not spaces. There also shouldn't be any trailing white space at the end of each line.
- The fields are: site, transformation name, executable path, installation status, platform, and profile entries.
- The site should match the site name in sites.xml (pool handle)
- The transformation name should match the Swift app procedure call (don't change)
- The executable path should specify where it is located
- The last three entries can be left as INSTALLED, INTEL32::LINUX, and null

Example:

localhost ncks /fs/local/bin/ncks INSTALLED INTEL32::LINUX null

sites.xml (-sites.file parameter)

- Lists the details of the machine and how to run the tasks
- Will differ based on if you are running on a machine with batch or not
- Rather complex, but provided for all common execution environments
- Can manage via templates; only a few parameters change for local needs

sites.xml (-sites.file)
Running locally

Example:

```
<config>
  <pool handle="localhost">
    <gridftp url="local://localhost" />
    <execution provider="local" url="none" />
    <filesystem provider="local"/>
    <workdirectory>/work/directory/</workdirectory>
    <profile namespace="karajan" key="jobThrottle">.03</profile>
    <profile namespace="karajan" key="initialScore">100</profile>
  </pool>
</config>
```

sites.xml (-sites.file)
Running in batch mode

```
<config>
<pool handle="localhost">
  <execution jobmanager="local:pbs" provider="coaster" url="none"/>
  <profile namespace="globus" key="maxtime">3600</profile>
  <profile namespace="globus" key="jobsPerNode">8</profile>
  <profile namespace="globus" key="slots">1</profile>
  <profile namespace="globus"  key="nodeGranularity">4</profile>
  <profile namespace="globus" key="maxNodes">4</profile>
  <profile namespace="karajan" key="jobThrottle">0.31</profile>
  <profile namespace="karajan" key="initialScore">10000</profile>
  <profile namespace="globus" key="project”>projectName</profile>
  <profile namespace="globus"  key="lowOverAllocation">100</profile>
  <profile namespace="globus"  key="highOverAllocation">100</profile>
  <filesystem provider="local"/>
  <workdirectory>/your/wrk/directory/</workdirectory>
</pool>
</config>
```

Platforms

- Both packages contain configuration files for mirage and lens
- On local machines, you should be able to use the mirage files
- On batch machines that use pbs, you should be able to use the lens files
- Just remember to edit the paths in tc.data and your workingdirectory in sites.xml

More information can be found at **http://www.ci.uchicago.edu/swift**

What went wrong? … you see "Execution failed:"

- The best place to look is the jobs log file.
  - For the AMWG package you'll look at $swift_scratch_dir/amwg_stats-*.log
  - For the OWMG package you'll look at $WORKDIR/popdiag*.log
- Tail the last 500 lines of this file and look right above the java error output

Look for something like …

Exception in za:

Arguments: [-O, -time_const, -o, fusion/group/climate/mickelso/OMWG-swiftData/swiftSandbox/swiftSandbox/za_PHC2_TEMP_tx0.1v2_ann_avg.nc, _concurrent/tobsFileTemp1-92e32ab5-d278-4432-a7af-2e8e666b07c8-]

stdout.txt: (NF_INQ_VARID_WRAP) NetCDF: Variable not found

(NF_INQ_VARID_WRAP) POP_grid_mod:read_POP_grid_NetCDF

Caused by: Application /fusion/gpfs/home/mickelso/soft/zon_avg/za failed with an exit code of 174

at org.globus.cog.karajan.workflow.nodes.functions.KException.function(KException.java:29)

at org.globus.cog.karajan.workflow.nodes.functions.AbstractFunction.post(AbstractFunction.java:27)

.

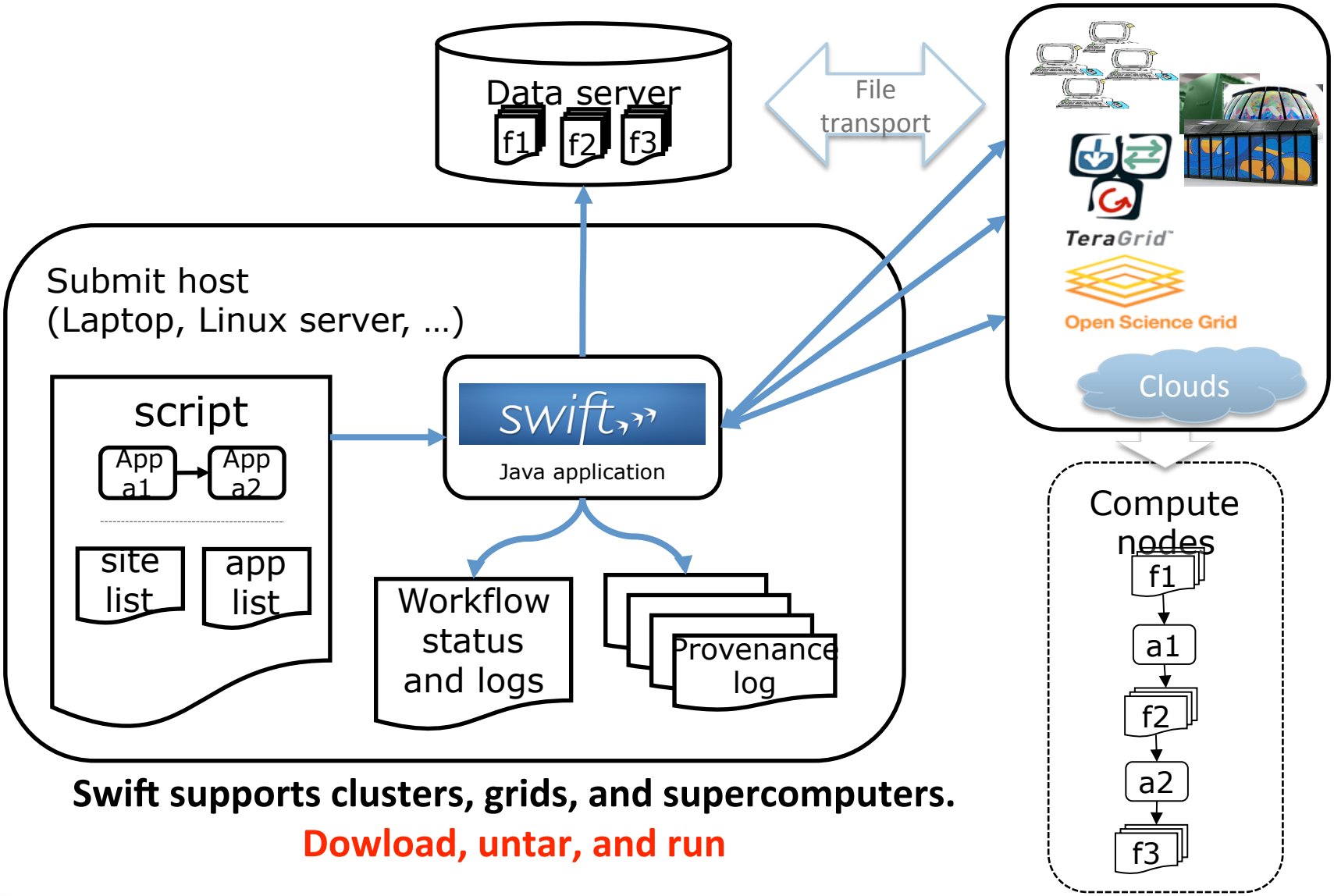
.

.

Demos …..

More about Swift….

Running Swift scripts



Swift supports clusters, grids, and supercomputers.

Dowload, untar, and run

Swift dataset typing and mapping

- Logical structure using Swift types
  - Primitive scalar types: int, float, string, …
  - Complex types (nested structs and arrays)
- Specified by **mapping descriptors** on variables
  - Dataset structures and elements are mapped to physical representations
  - Parameters can be passed to mappers (e. g. location
  - Mappers can be builtins, or external programs or scripts

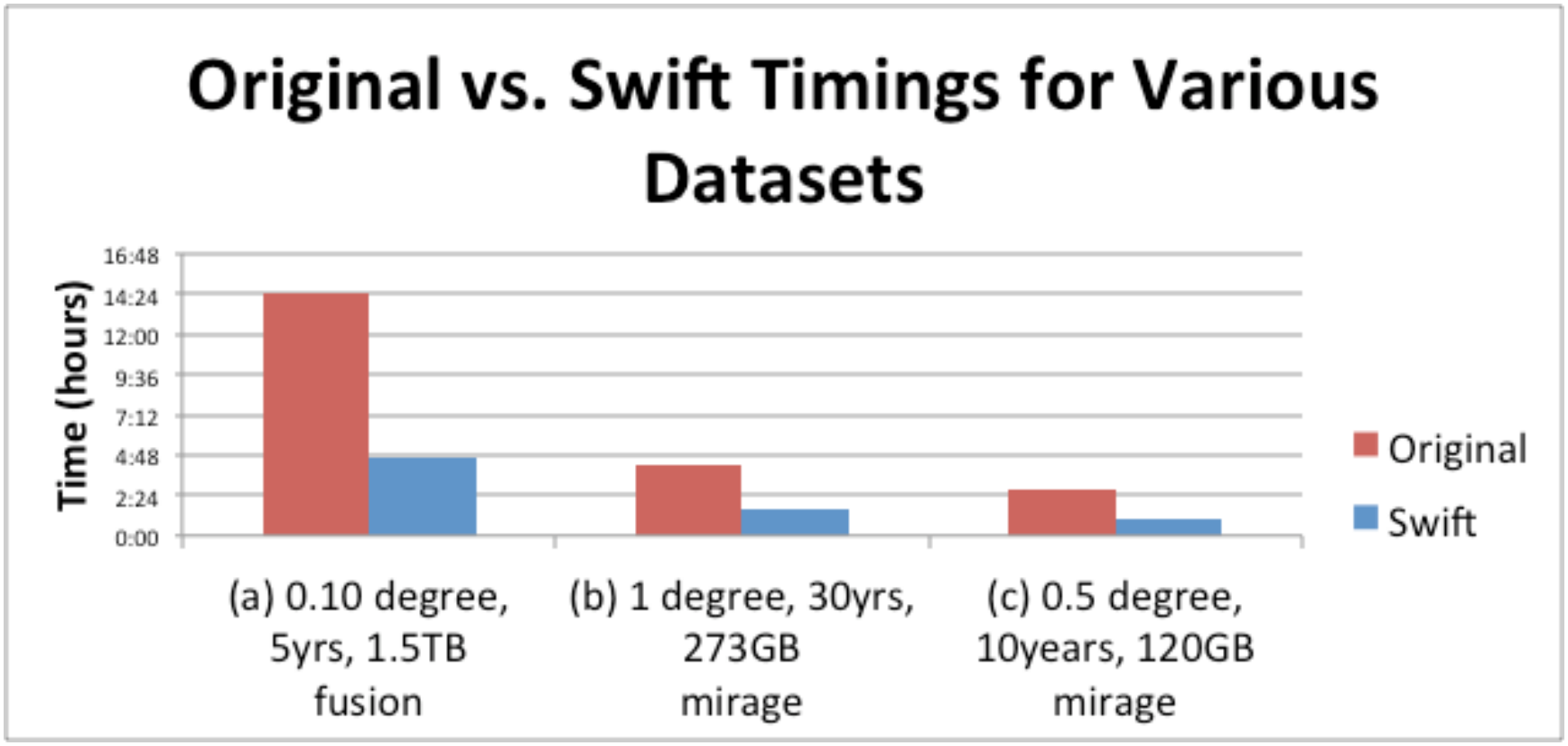
Example code from the popdiag.swift

```
# Create the MAVG file
(file mavgFileL) CalcMAVG(string dpath, string casename, string msr, int yro, int yr1, int years[], file
monhsiFileL[]){
 string monthNames[]=["jan","feb","mar","apr","may","jun","jul","aug","sep","oct","nov","dec"];
 file monthFiles[];
 string var1 = "TEMP";
 string var2 = "SALT";
 int m[]=[1,2,3,4,5,6,7,8,9,10,11,12];
 foreach month in m {
    string mname[];
    foreach year in years {
       mname[year] = @strcat(dpath,casename,".pop.h.",yearprint(year),"-",monthprint(month),".nc");
    }
    file monthFile;
    monthFile = Average_var("TEMP,SALT", mname, monhsiFileL);
    monthFiles[month] = monthFile;
 }
 mavgFileL = Record_Cat(monthFiles);
}
```

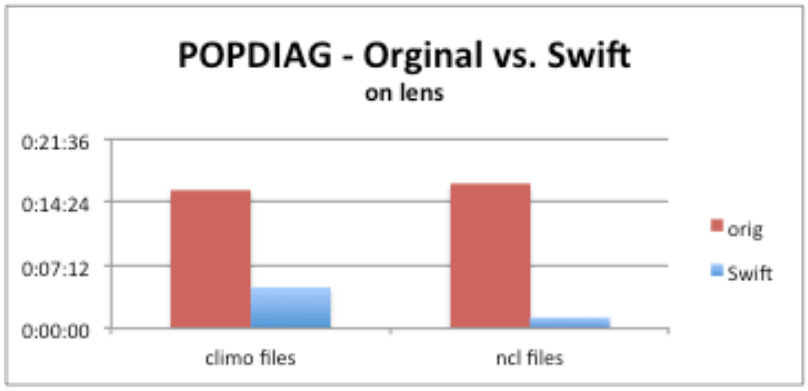
Swift fault tolerance

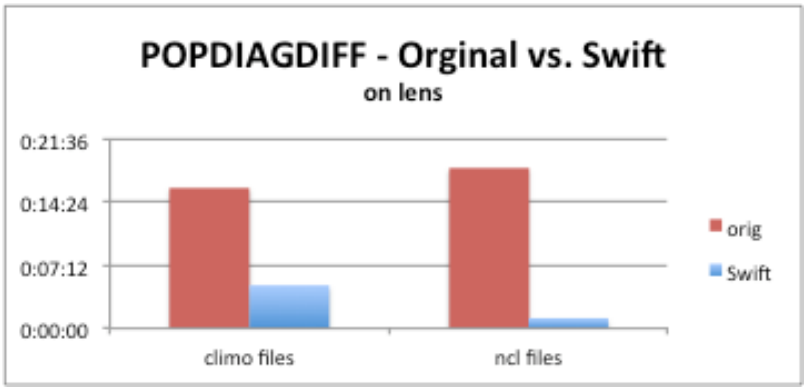
- Swift can retry jobs
  - Up to a user specified limit
  - Can stop on first unrecoverable failure, or continue till no more work can be done
  - Very effective, since Swift can break workflow into many separate scheduler jobs, hence smaller failure units
- Swift can replicate jobs
  - If jobs don't complete in a designated time window, Swift can send copies of the job to other sites or systems
  - The first copy to succeed is used, other copies are removed
- Each app() job can define "failure"
  - Typically non-zero return code
  - Wrapper scripts can decide to mask app() failures and pass back data/ logs about errors instead

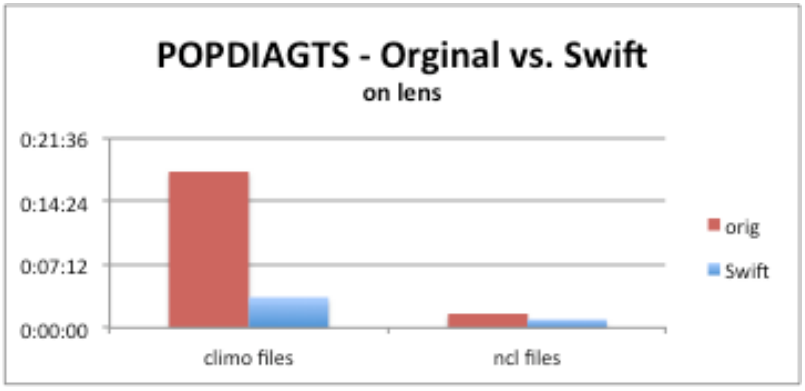
AMWG Diagnostic Package Timings



OMWG Diagnostic Package Timings







All timings were ran on 10 years of monthly history files
Swift was ran on 4 nodes with 8 tasks per node. Throttle was set to 0.31.

Questions …

For more information, please visit

https://trac.mcs.anl.gov/projects/parvis/wiki/SwiftWork

or contact me at mickelso@mcs.anl.gov

Thanks to Susan Bates, Gokhan Danabasoglu, Rich Neale, Cecile Hannay, and Mariana Vertenstein for their support for this project, Andy Mai for help testing, Dave Brown for converting the IDL scripts to NCL in the OMWG Diagnostic Package, and John Dennis for his initial work with the AMWG Diagnostic Package.