

Swift Parallel Scripting for Science, Engineering and Data Analysis

Globus World – April 14, 2015

Michael Wilde wilde@anl.gov

<http://swift-lang.org>

Swift gratefully acknowledges support from:



U.S. DEPARTMENT OF
ENERGY



THE UNIVERSITY OF
CHICAGO Argonne
NATIONAL LABORATORY



Expressing this workflow in Swift

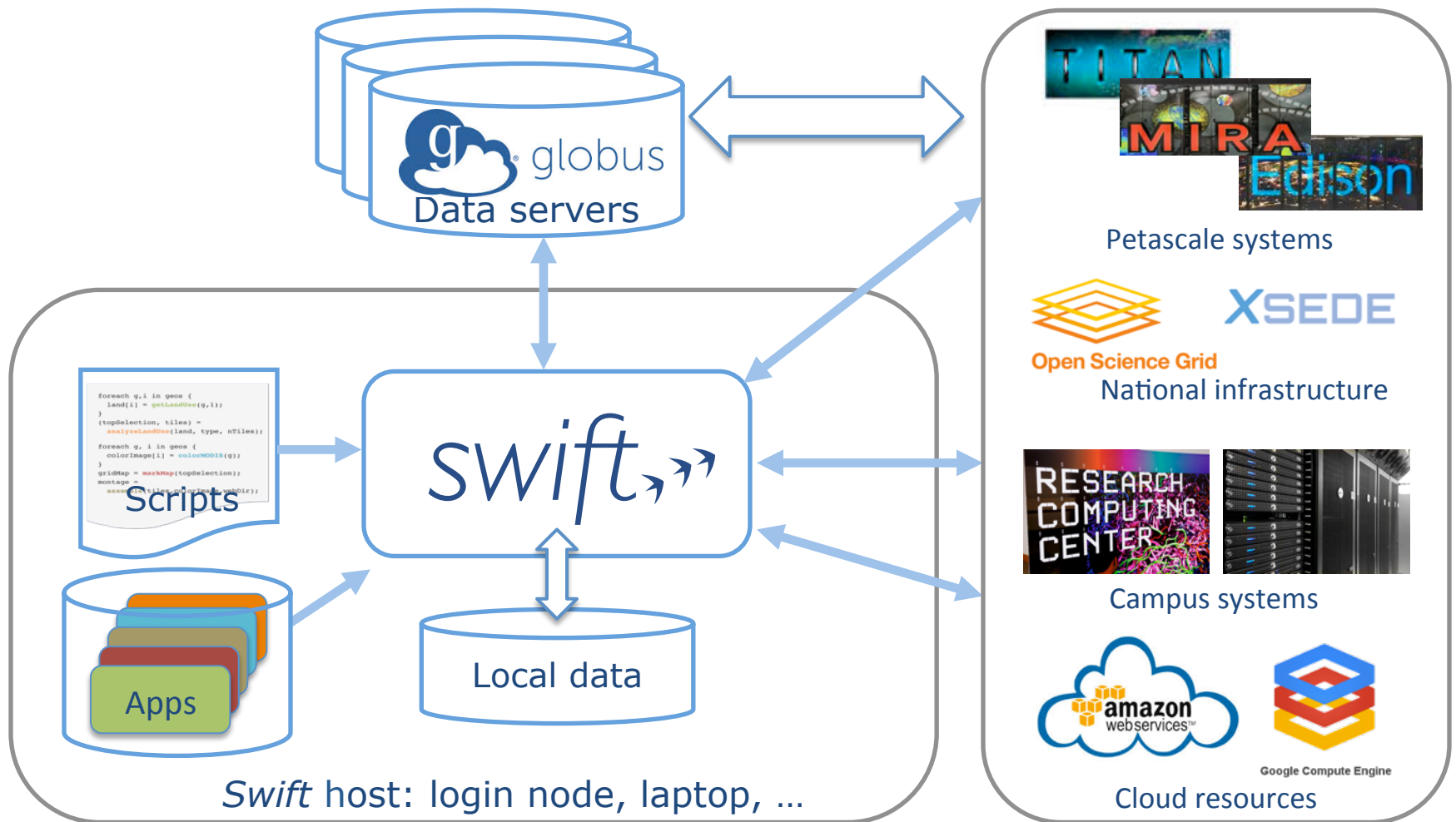
For protein docking workflow:

```
foreach p, i in proteins {  
    foreach c, j in ligands {  
        (structure[i,j], log[i,j]) =  
            dock(p, c, minRad, maxRad);  
    }  
    scatter_plot = analyze(structure)
```

To run:

```
swift -site tukey,blues dock.swift
```

Swift enables execution of simulation campaigns across multiple HPC and cloud resources



Swift runtime drivers support and aggregate diverse runtime environments

Swift provides 4 important benefits:

Makes parallelism more transparent

Implicitly parallel functional dataflow programming

Makes computing location more transparent

Runs your script on multiple distributed sites and diverse computing resources (desktop to petascale)

Makes basic failure recovery transparent

Retries/relocates failing tasks

Can restart failing runs from point of failure

Enables provenance capture

Tasks have recordable inputs and outputs

Example of Swift's implicit parallelism: Processing MODIS land-use data

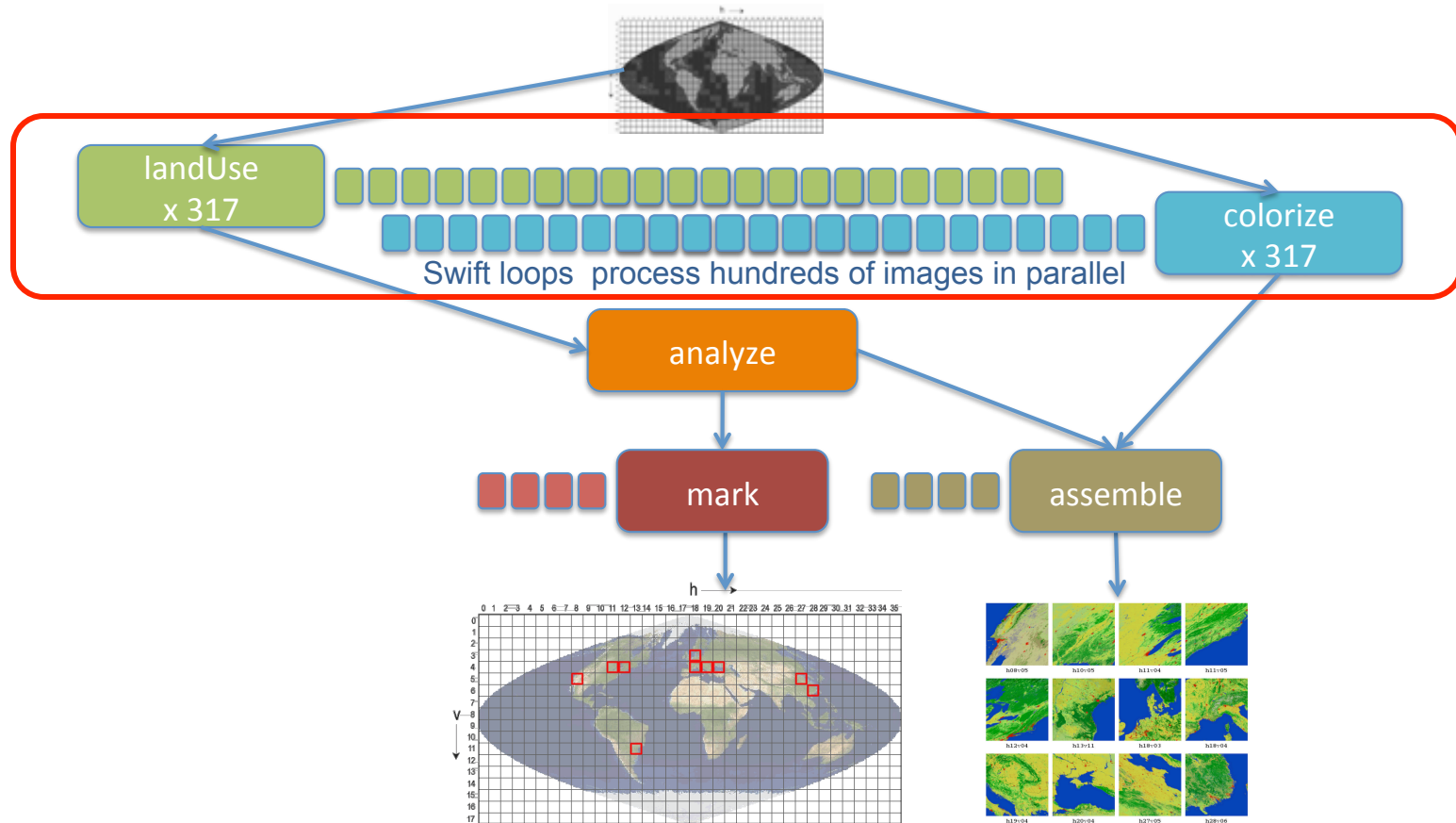
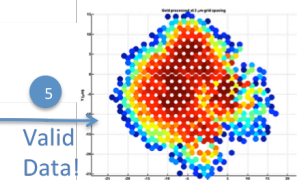
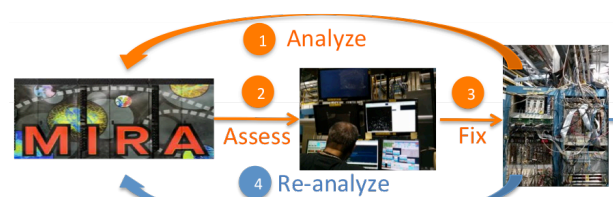
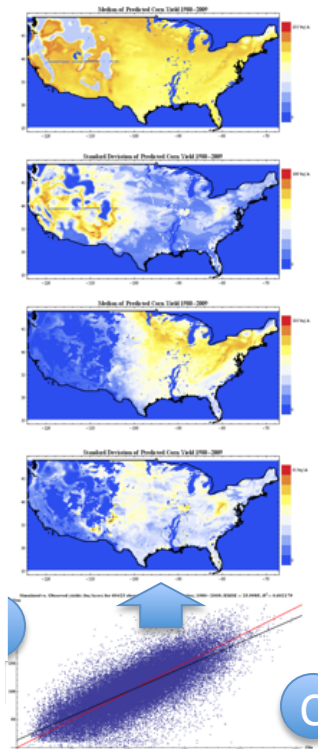
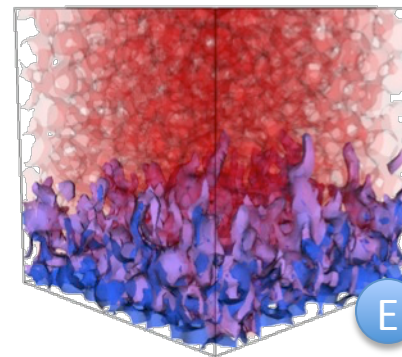
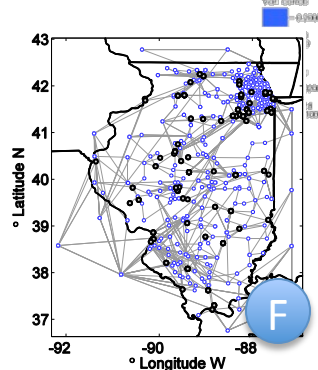
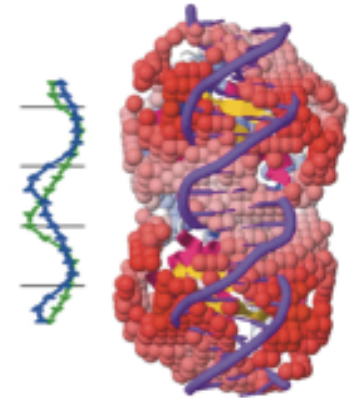
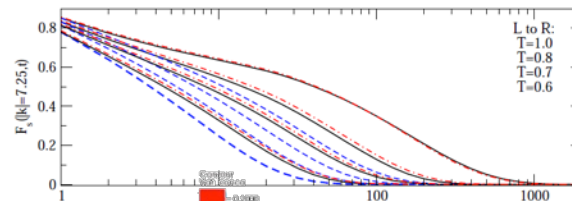
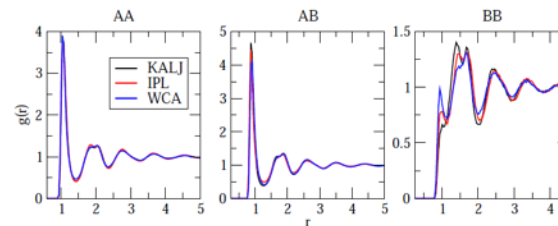


Image processing pipeline for land-use data from the MODIS satellite instrument...

Large-scale applications using Swift

- A** Simulation of super-cooled glass materials
- B** Protein and biomolecule structure and interaction
- C** Climate model analysis and decision making for global food production & supply
- D** Materials science at the Advanced Photon Source
- E** Multiscale subsurface flow modeling
- F** Modeling of power grid for OE applications

All have published science results obtained using Swift

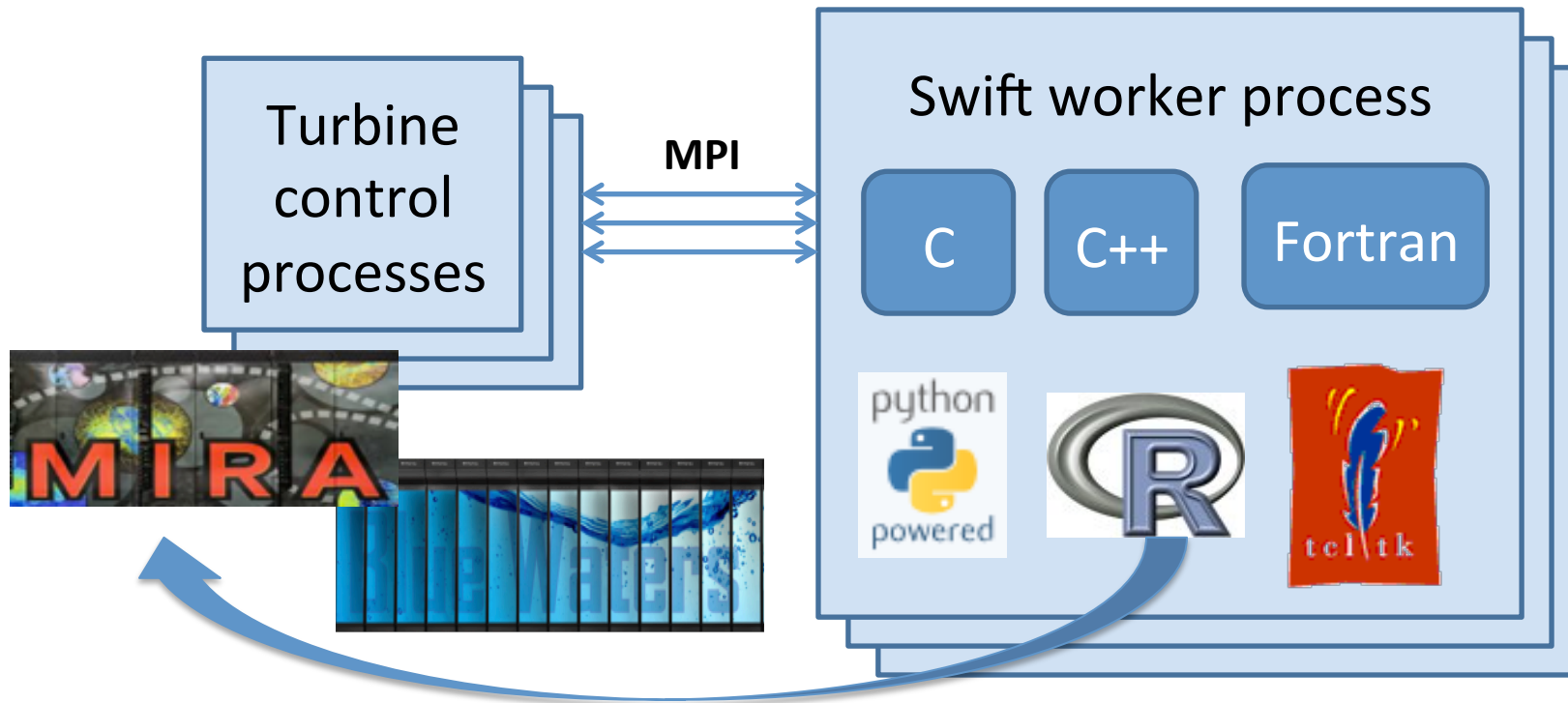


Red indicates higher statistical confidence in data

swift

<http://swift-lang.org>

Swift/T: productive extreme-scale scripting



- Script-like global-view programming with “leaf” tasks
 - function calls in C, C++, Fortran, Python, R, Julia or Tcl
- Leaf tasks can be MPI programs, etc. Can be separate processes if OS permits.
- Distributed, scalable runtime manages tasks, load balancing, data movement
- User function calls to external code run on thousands of worker nodes
- More expressive than master-worker for “programming in the large”

swift_{→→}

<http://swift-lang.org>

The *parallel.works* solution

Select third-party and open-source applications

Link apps together to create a workflow

Select compute and storage resources and desired time to solution

Run simulation workflow and visualize results

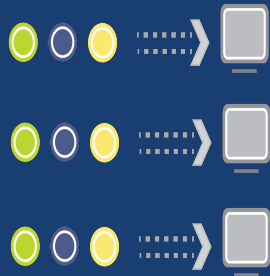
The screenshot displays the *parallel.works* web interface. The top navigation bar includes 'parallel.works', 'DASHBOARD', 'CLUSTERS', 'APPS', 'WORKFLOWS', 'TEAM', 'PAYMENT', 'SETTINGS', and 'SIGN OUT'. The main interface is divided into several sections:

- Tools:** A search bar for tools.
- App Marketplace:** A grid of application icons including Radiance, MATLAB, and Python.
- My Clusters:** A grid of cloud provider icons including Amazon Web Services, Rockspace, and others.
- My Storage:** A grid of storage provider icons including S3, Google Drive, and Dropbox.
- Workflow Canvas:** A central workspace where a workflow is being constructed. It features a 'node' icon, a 'Radiance' node, and a 'ParaView' node. The workflow is connected to 'out_file1'.
- INPUT PARAMETERS:** A section for configuring the workflow, including 'Number of Simulations' (set to 100) and 'Add expression' (set to $c^2 \cdot 2$).
- Summary:** A section showing '3.5 Hours', '512 Cores', and '\$25 /run'.
- Details:** A section showing a 'RESULT MODEL' visualization of a city skyline with various colored buildings and a 'ParaView' logo. Below the visualization are 'Visualization Parameters' (Toggle View Lines, Analysis Grid, Run View Study) and an 'OPTIMIZATION OVERVIEW' graph showing a line chart with a legend.

The *parallel.works* Paradigm Shift

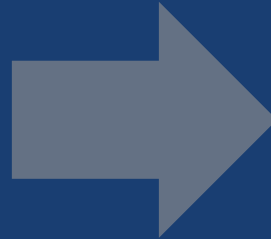
Early experience at leading architecture firm SOM

Current Practice

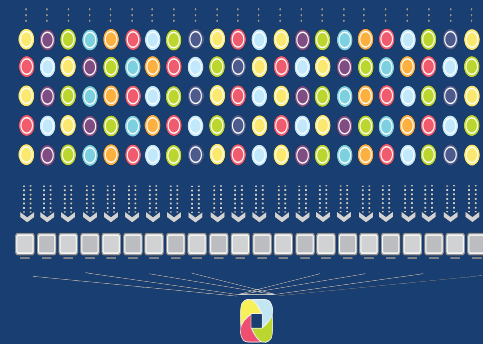


130 days

time-to-solution on a typical desktop computer (8 CPU cores)



parallel.works Solution



5 hours

time-to-solution using 5,000 CPU cores on a remote supercomputing facility

625 X faster simulation gives a monetizable time advantage by creating more innovative designs that reduce cost for clients.

What's Inside?

The Swift logo features the word "Swift" in a white, elegant serif font. To the right of the text are three white arrows of increasing size, pointing to the right.

*Scalable workflow
language & engine*

Automates parallelization, data movement, failure recovery, and provenance capture.



*Fast, secure, reliable
data transfer*

Moves and shares data robustly and securely at gigabytes/second



*Easy to use web
workflow interface*

Create, run, view, manage, discover & share workflows

parallel.works' competitive advantage derives from the integration of – and expertise in – three powerful technologies

Swift gratefully acknowledges support from:



U.S. DEPARTMENT OF
ENERGY



THE UNIVERSITY OF
CHICAGO

Argonne

NATIONAL LABORATORY



<http://swift-lang.org>