



The Swift scripting language for Science Clouds

Justin M. Wozniak

Computation Institute, University of Chicago
and Argonne National Laboratory

wilde@mcs.anl.gov

Revised 2012.0229

[*www.ci.uchicago.edu/swift*](http://www.ci.uchicago.edu/swift)



Computation Institute

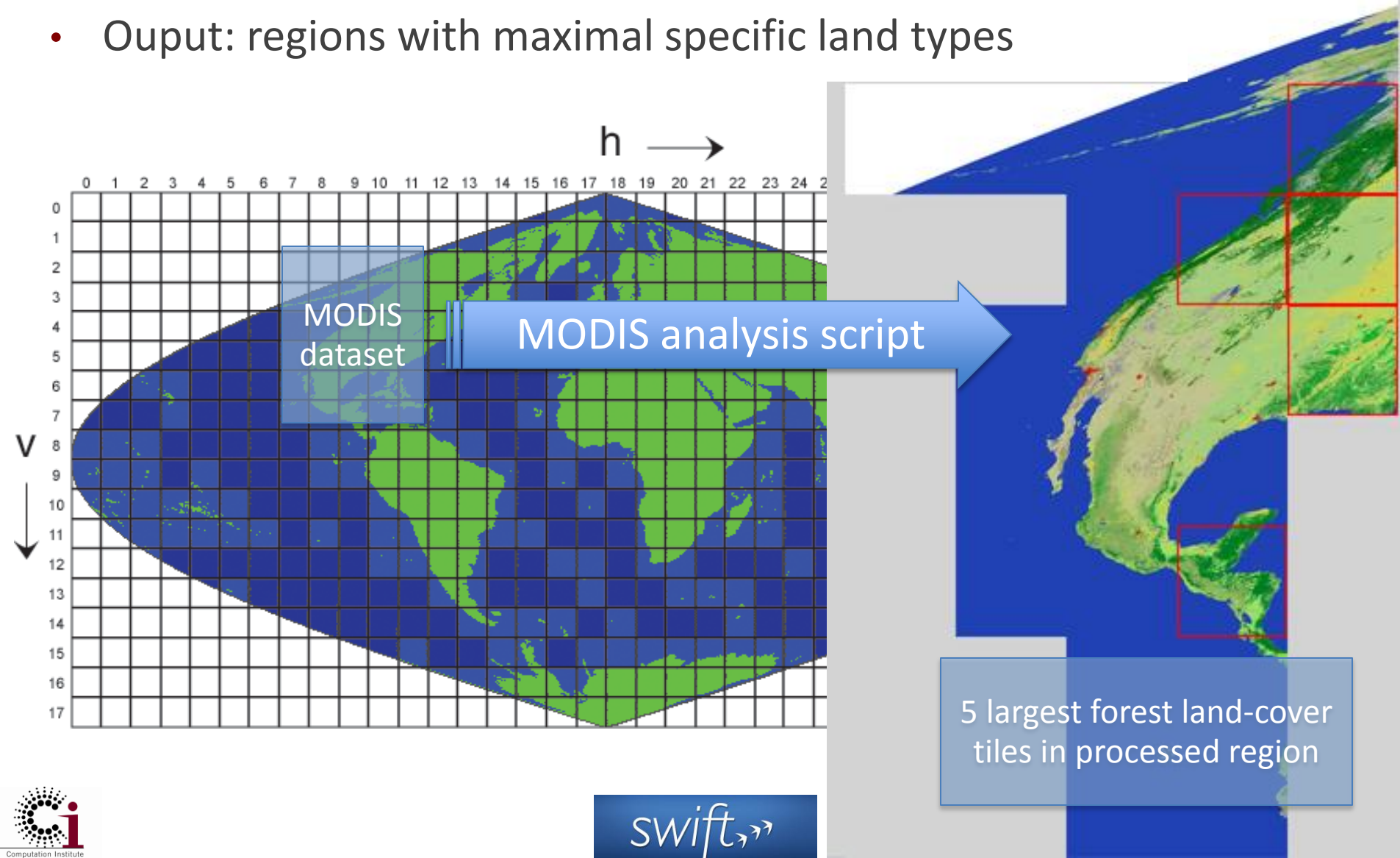


Context

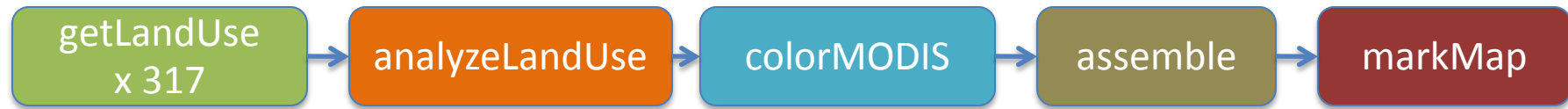
- You've heard this afternoon how to run Science work in Clouds
- *But further challenges need to be addressed:*
 - *Running applications with data dependencies that require complex pipelines*
 - *Moving data fast and automatically*
 - *Dynamically changing size of provisioned resource pools*
 - *Handling failures of nodes, networks, application stacks*

Example – MODIS satellite image processing

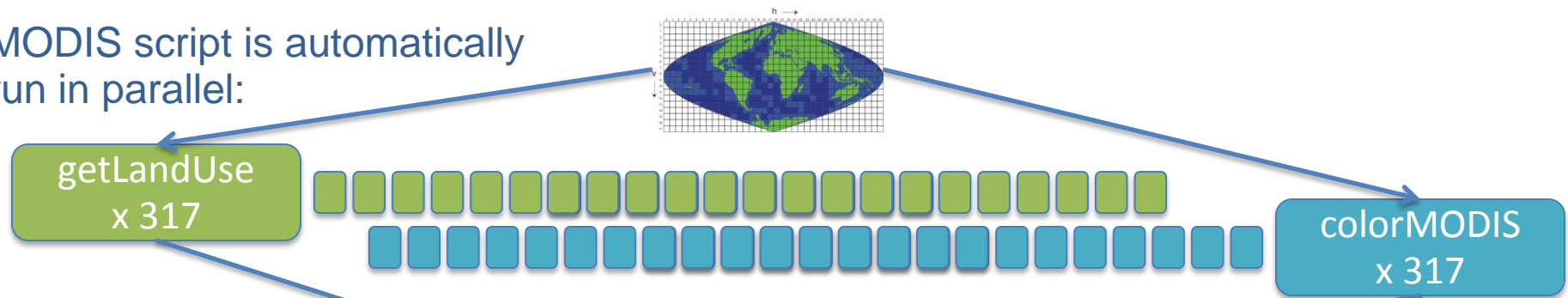
- Input: tiles of earth land cover (forest, ice, water, urban, etc)
- Output: regions with maximal specific land types



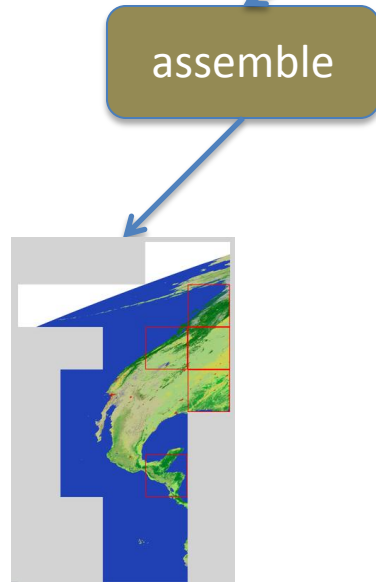
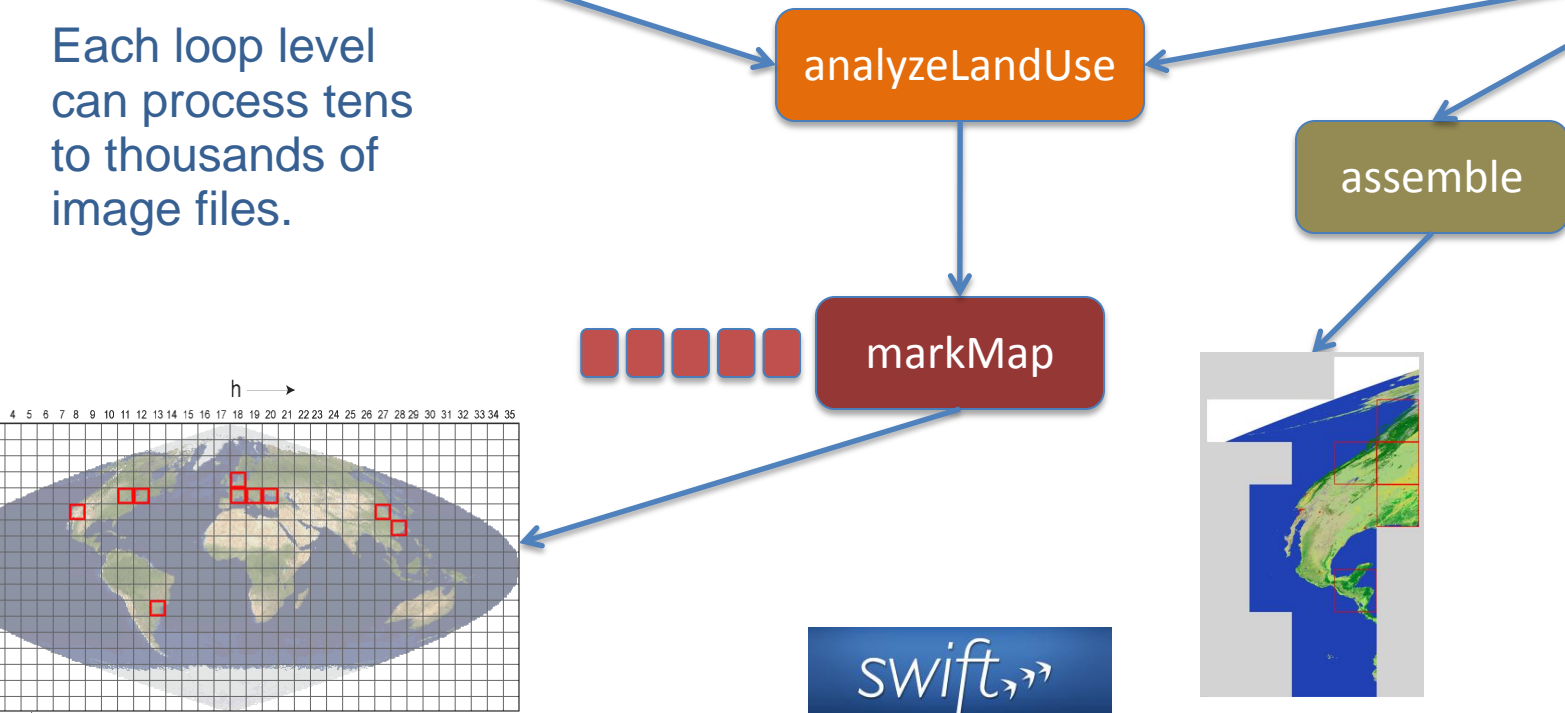
Goal: Run MODIS processing pipeline in cloud



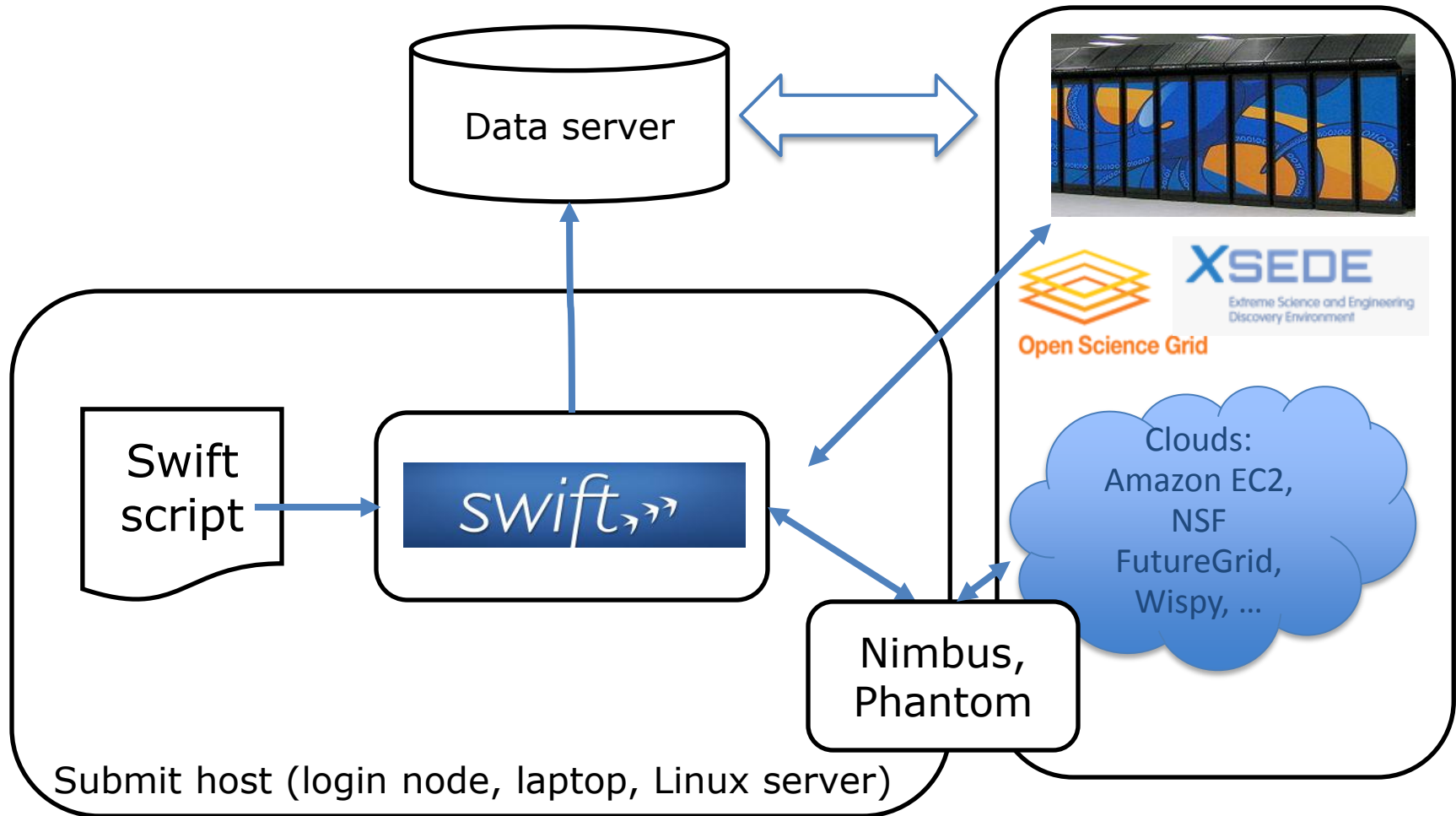
MODIS script is automatically run in parallel:



Each loop level can process tens to thousands of image files.



Solution: Swift parallel distributed scripting



Swift runs parallel scripts on cloud resources provisioned by Nimbus's *Phantom* service.



Features overview

- **Swift is a parallel scripting language** for multicores, clusters, grids, clouds, and **supercomputers**
 - *for loosely-coupled “many-task” applications – programs and tools linked by exchanging files*
 - *debug on a laptop, then run on a large system*
- **Swift is easy to write**
 - *a simple high-level functional language with C-like syntax*
 - *Small Swift scripts can do large-scale work*
- **Swift is easy to run:** contains all services for running the workflow - in one Java application
 - *untar and run – Swift acts as a self-contained grid or cloud client*
 - ***Swift automatically runs scripts in parallel*** – *typically without user declarations*
- **Swift is fast:** based on a powerful, efficient, scalable and flexible Java execution engine
 - *scales readily to millions of tasks*
- **Swift is general purpose:**
 - *applications in neuroscience, proteomics, molecular dynamics, biochemistry, economics, statistics, earth systems science, and beyond.*

MODIS script in Swift: main data flow

```
foreach g,i in geos {  
    land[i] = getLandUse(g,1);  
}  
(topSelected, selectedTiles) =  
    analyzeLandUse(land, landType, nSelect);  
  
foreach g, i in geos {  
    colorImage[i] = colorMODIS(g);  
}  
gridMap = markMap(topSelected);  
montage =  
    assemble(selectedTiles,colorImage,webDir);
```

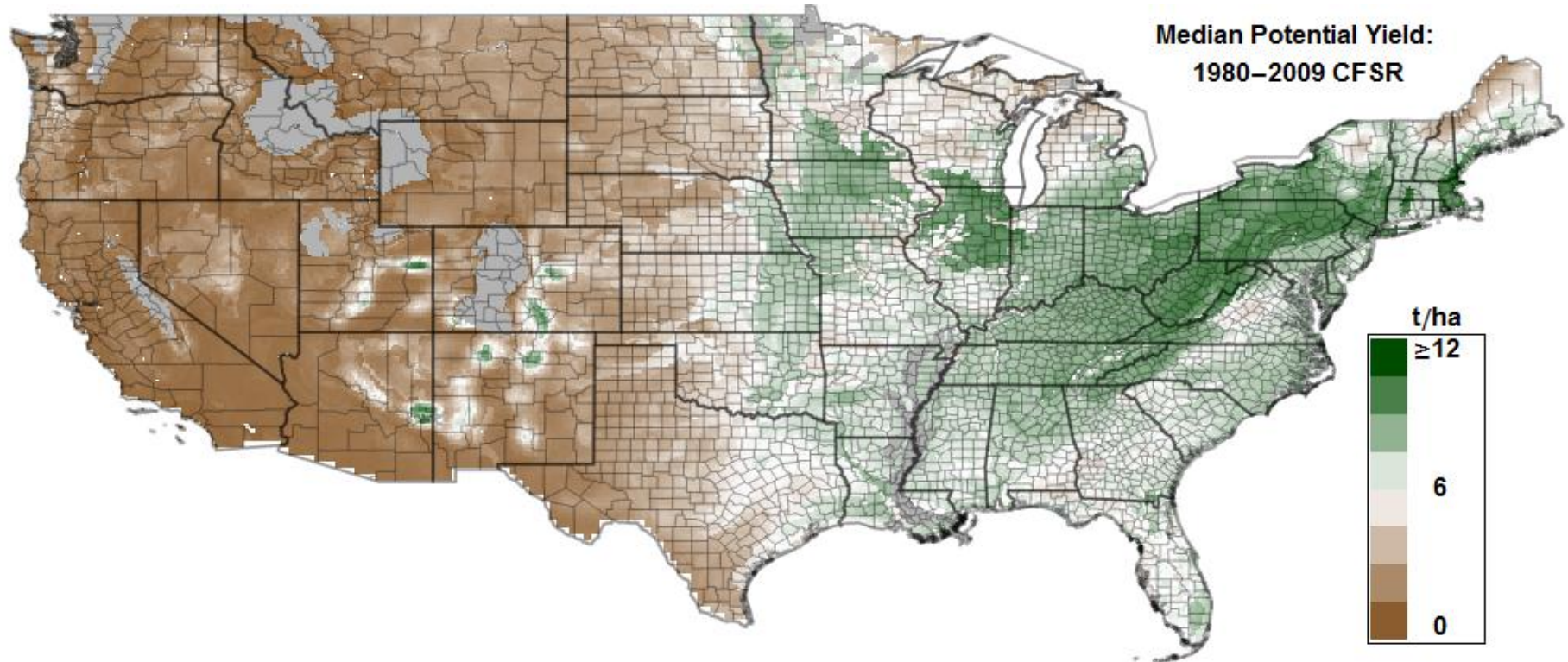
Demo of Nimbus-Phantom-Swift on FutureGrid

- User provisions 5 nodes with Phantom
 - Phantom starts 5 VMs
 - Swift worker agents in VMs contact Swift coaster service to request work
- Start Swift application script “MODIS”
 - Swift places application jobs on free workers
 - Workers pull input data, run app, push output data
- 3 nodes fail and shut down
 - Jobs in progress fail, Swift retries
- User can add more nodes with phantom
 - User asks Phantom to increase node allocation to 12
 - Swift worker agents register, pick up new workers, runs more in parallel
- Workload completes
 - Science results are available on output data server
 - Worker infrastructure is available for new workloads

Swift and Phantom provide fault tolerance

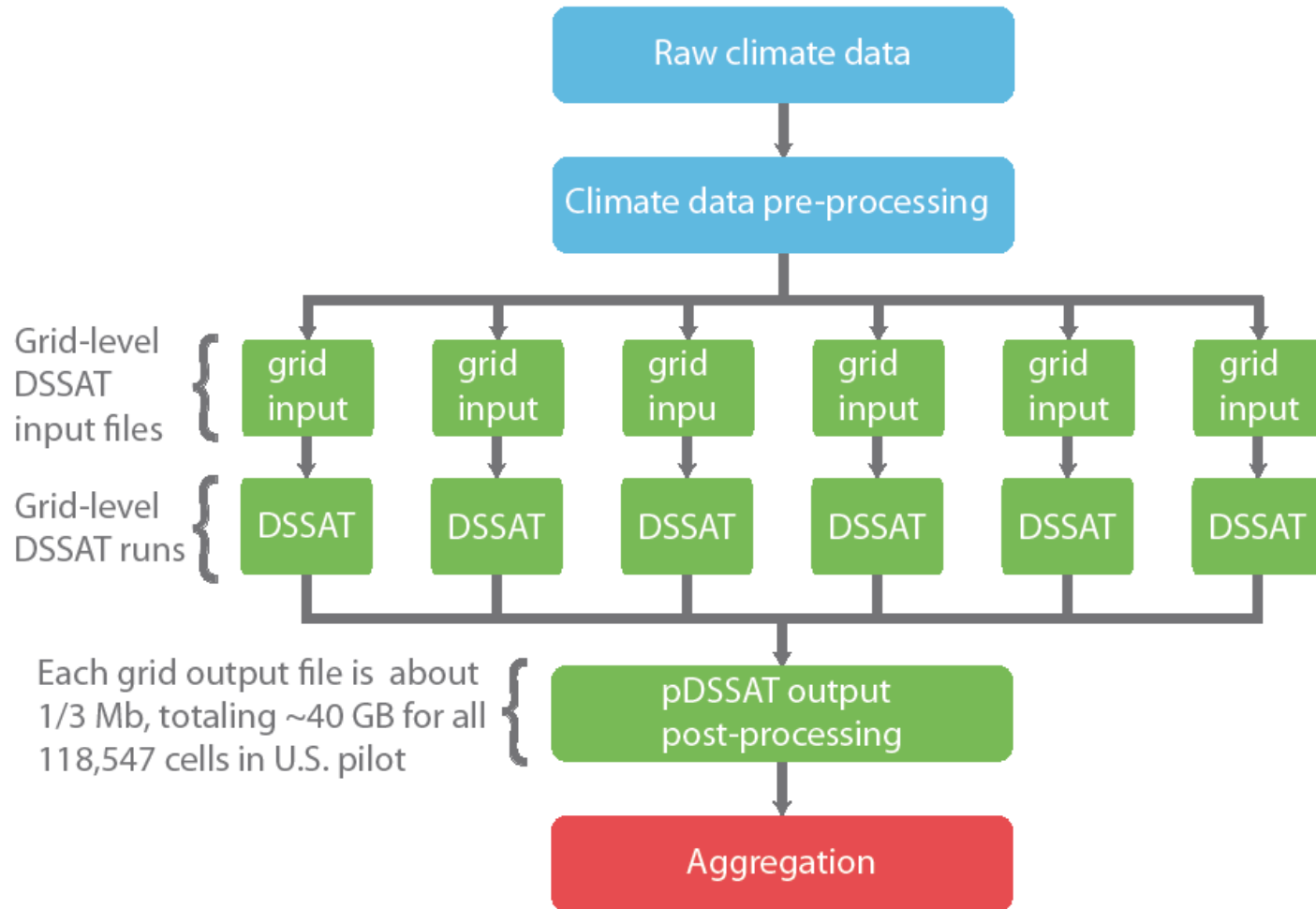
- Phantom detects downed nodes and re-provisions
- Swift can retry jobs
 - Up to a user specified limit
 - Can stop on first unrecoverable failure, or continue till no more work can be done
 - Very effective, since Swift can break workflow into many separate scheduler jobs, hence smaller failure units
- 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

Land use studies - DSSAT



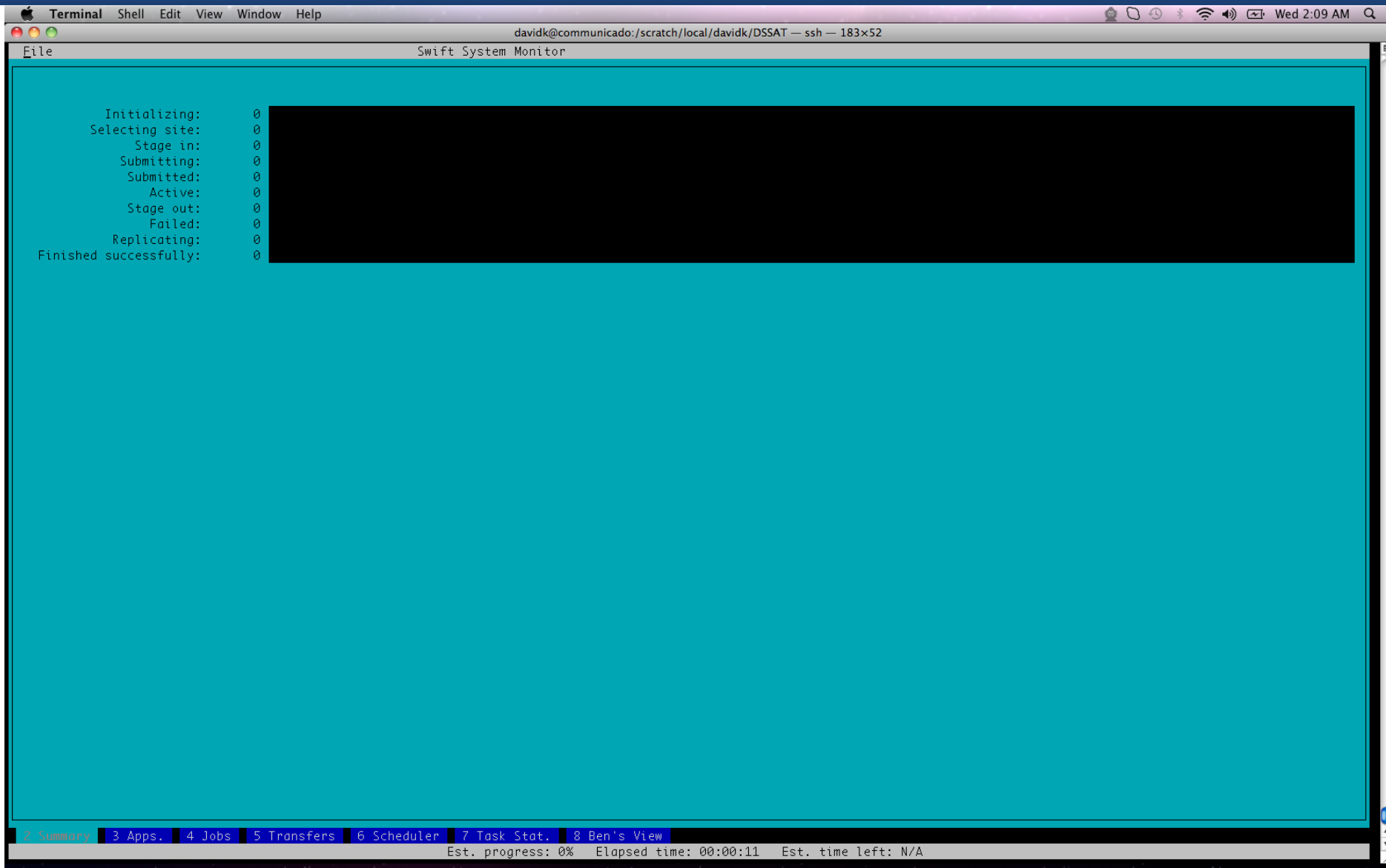
Median 1980-2009 corn yields (metric tons per hectare) as simulated by Decision Support System for Agrotechnology Transfer (DSSAT) driven by National Center for Environmental Prediction (NCEP) Climate Forecast System Reanalysis (CFSR).

DSSAT – workflow schematic



- *Let's run it*

Swift's Text User Interface (TUI) – Initial State



The screenshot shows a terminal window titled "Terminal" with a menu bar (Shell, Edit, View, Window, Help) and a status bar (Wed 2:09 AM). The main window is titled "Swift System Monitor" and displays the following text:

```
Initializing: 0
Selecting site: 0
Stage in: 0
Submitting: 0
Submitted: 0
Active: 0
Stage out: 0
Failed: 0
Replicating: 0
Finished successfully: 0
```

The text is displayed on a black background within a cyan-colored frame. At the bottom of the terminal window, there is a navigation bar with tabs: 2 Summary, 3 Apps., 4 Jobs, 5 Transfers, 6 Scheduler, 7 Task Stat., and 8 Ben's View. Below the navigation bar, the status bar shows: Est. progress: 0% Elapsed time: 00:00:11 Est. time left: N/A.

Provide nodes to Swift from 'hotel'

Safari File Edit View History Bookmarks Window Help

Phantom Launch Configurations

https://svc.uc.futuregrid.org:8440/phantom/launchconfig

Google

NIMBUS

Home About Nimbus FAQ Documentation Download Community Resources Publications News

Home Domain Launch Configurations Edit Clouds Change Password Logout

Multi-Cloud Launch Configuration

<new name> Save Delete

Cloud Options

Cloud: hotel

Max VMs: 90

Instance Type: m1.small

☐ Public Image

☒ Personal Image swift

User Data

Cloud Order

hotel

Up Down

Home About Nimbus FAQ Documentation Community Resources Download Publications News

© University of Chicago

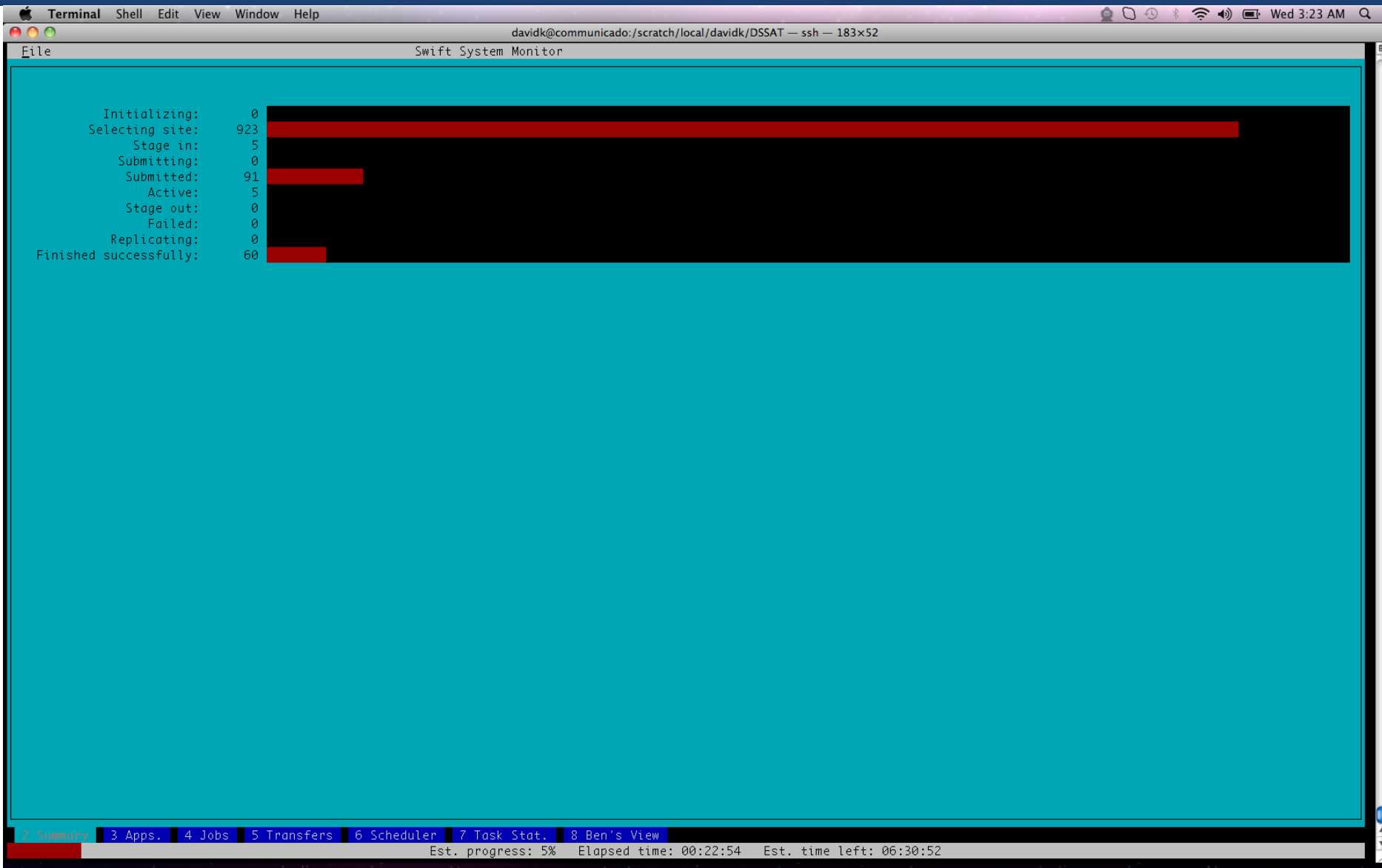
Add 10 more nodes from 'sierra'

The screenshot shows a web browser window with the URL `https://svc.uc.futuregrid.org:8440/phantom/launchconfig`. The page is titled "NIMBUS" and has a navigation bar with links: Home, About Nimbus, FAQ, Documentation, Download, Community Resources, Publications, and News. Below this is a sub-navigation bar with links: Home, Domain, Launch Configurations, Edit Clouds, Change Password, and Logout. The main content area is titled "Multi-Cloud Launch Configuration". It features a form with the following fields and controls:

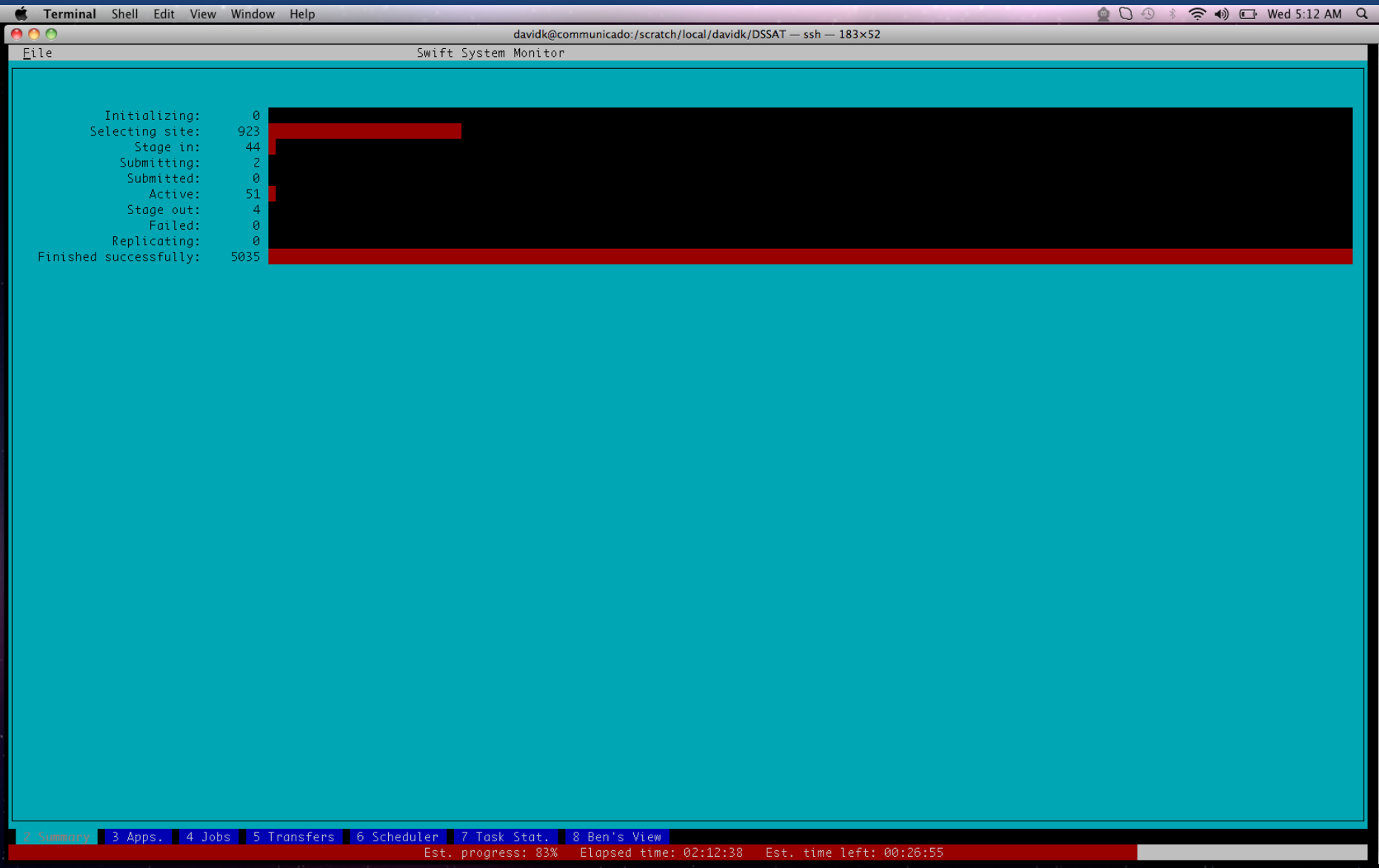
- A text input field with the placeholder "<new name>" and buttons for "Save" and "Delete".
- Cloud Options:**
 - Cloud: A dropdown menu with "sierra" selected.
 - Max VMs: A text input field with "10" entered.
 - Instance Type: A dropdown menu with "m1.small" selected.
 - Public Image: A radio button.
 - Personal Image: A radio button with "swift" selected in the dropdown.
 - User Data: A text area.
- Buttons: "Add ->" and "Remove X".
- Cloud Order:** A list box containing "hotel" and "sierra". Below the list are "Up" and "Down" buttons.

At the bottom of the page, there is a footer with links: Home, About Nimbus, FAQ, Documentation, Community Resources, Download, Publications, and News. Below the links is the text "© University of Chicago".

Swift starts running jobs...



Swift continues (2 hours later)



Forcibly terminate some of Swifts' resources

A screenshot of a web browser window showing the "Phantom Domains" interface. The browser is Safari, and the address bar shows the URL <https://svc.uc.futuregrid.org:8440/phantom/domain>. The page features the NIMBUS logo and a navigation menu with links: Home, About Nimbus, FAQ, Documentation, Download, Community Resources, Publications, and News. Below this, a secondary navigation bar includes: Home, Domain, Launch Configurations, Edit Clouds, Change Password, and Logout.

The main content area is titled "Domains" and is divided into two panels:

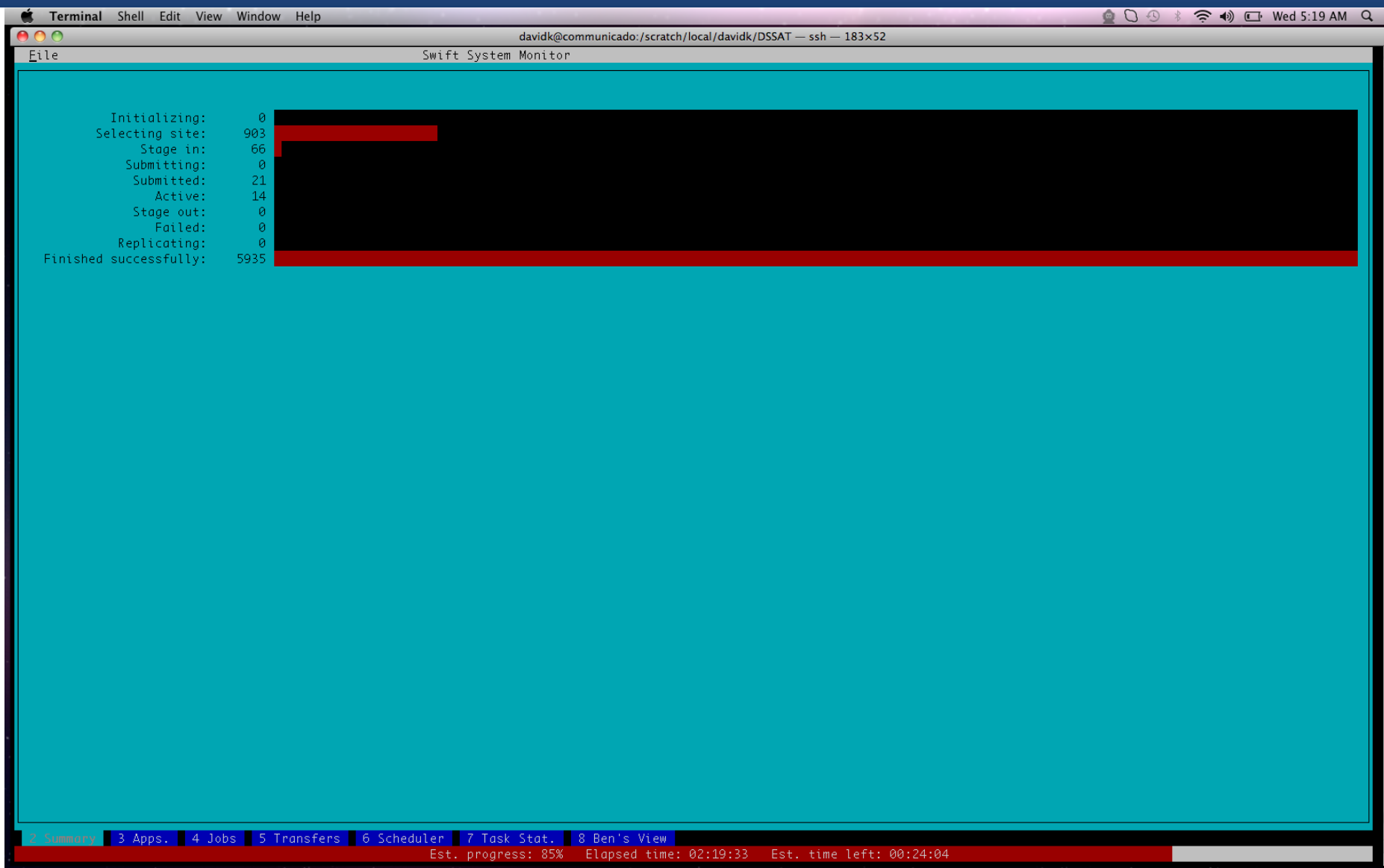
- Domain Options:** Contains form fields for "Domain Name" (demo-domain), "Size" (80), and "Launch Configuration" (demo-configuration). Below these are buttons for "Start", "Resize", and "Terminate". A scrollable list below shows "demo-domain".
- Domain details:** Shows a list of domains with their status and configuration. The "Show:" dropdown is set to "All", and an "Update" button is present. The list includes two domains, each with a status of "600-RUNNING" and a "Healthy" state.

The domain details list is as follows:

ID	Status	Health	VMs	OS	Key
i-90b6e881	600-RUNNING	Healthy	vm-148-129.uc.futuregrid.org	swift	m1.small
				phantomkey	
i-33cdccf8	600-RUNNING	Healthy	vm-148-226.uc.futuregrid.org	swift	m1.small
				phantomkev	

At the bottom of the page, there is a footer with links: Home | About Nimbus | FAQ | Documentation | Community Resources | Download | Publications | News, and a copyright notice: © University of Chicago.

Swift chugs along...



Restore Swift's compute nodes

Safari File Edit View History Bookmarks Window Help

Phantom Domains

https://svc.uc.futuregrid.org:8440/phantom/domain

Apple Yahoo! Google Maps YouTube Wikipedia News (499) Popular

NIMBUS

Home About Nimbus FAQ Documentation Download Community Resources Publications News

Home Domain Launch Configurations Edit Clouds Change Password Logout

Domains

Domain Options

Domain Name:

Size:

Launch Configuration:

demo-domain

Domain details.

Show:

m1.small
phantomkey

100-REQUESTING
sierra
Healthy
unknown
swift
m1.small
phantomkey

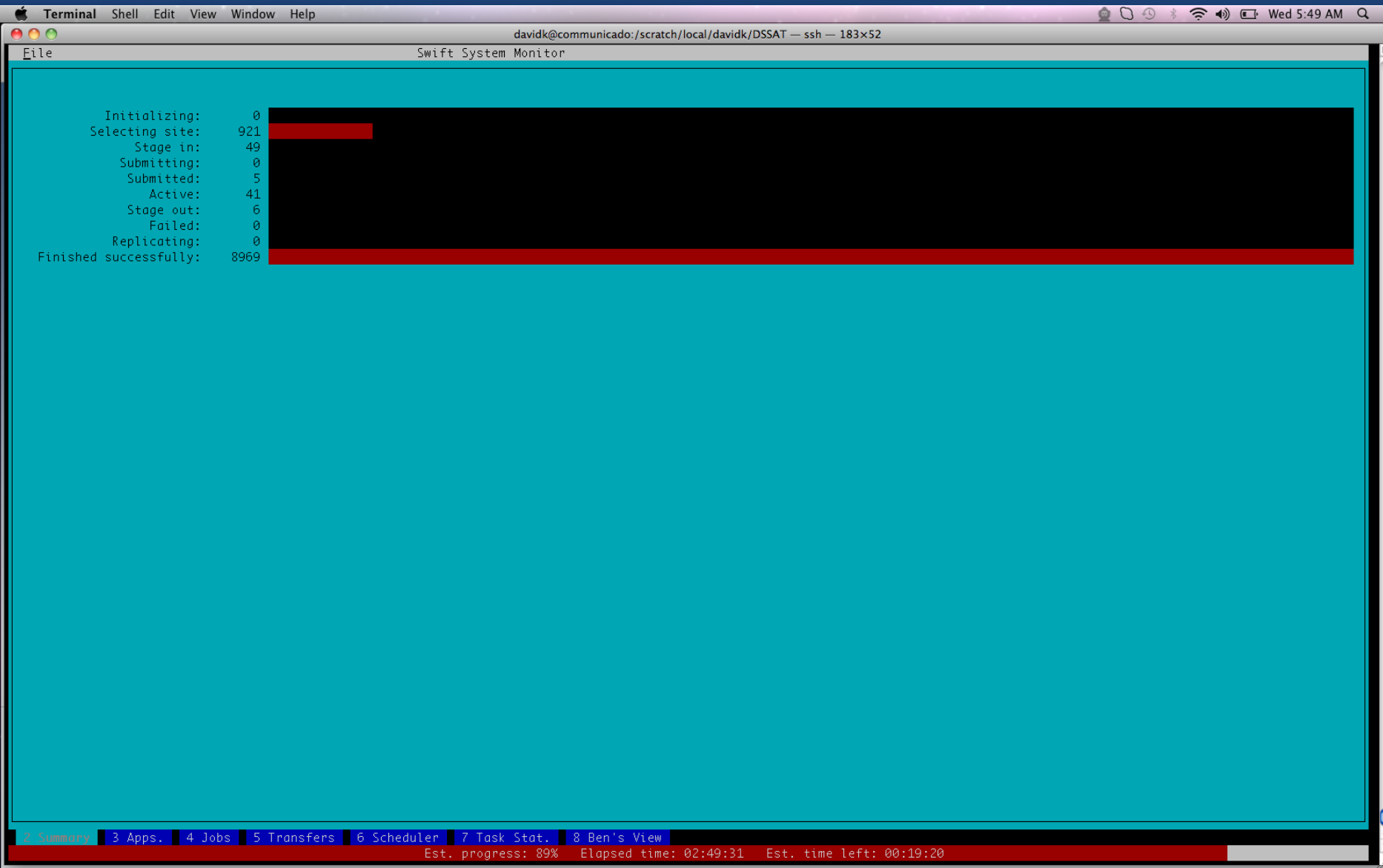
i-7e283d7a

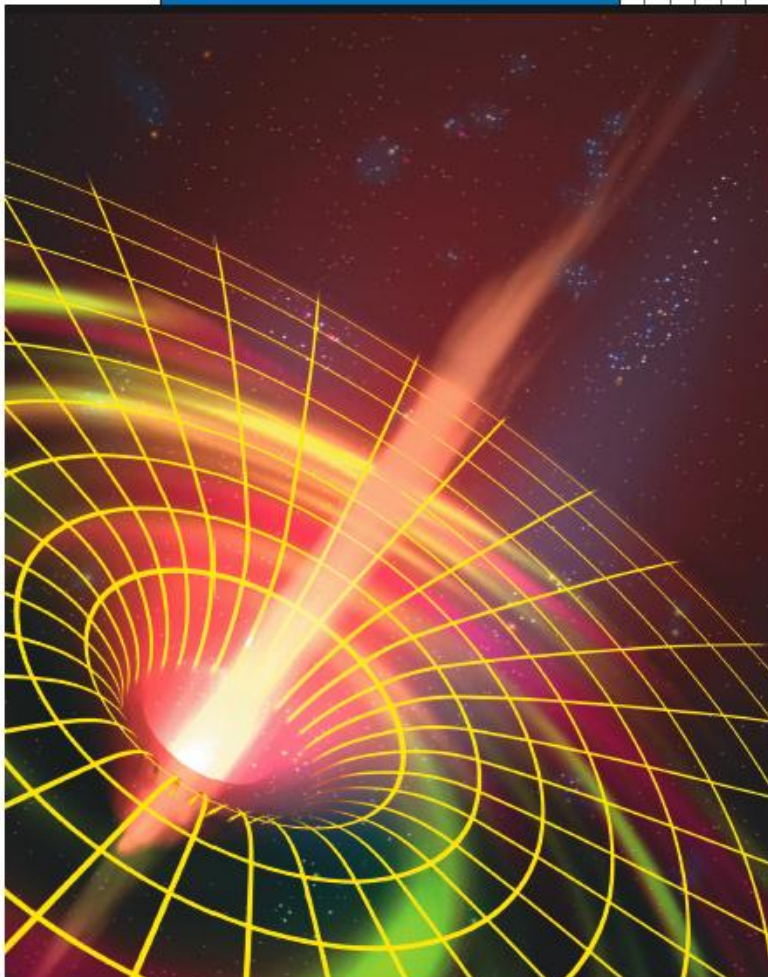
600-RUNNING
hotel
Healthy
vm-148-166.uc.futuregrid.org
swift
m1.small
phantomkey

Home About Nimbus FAQ Documentation Community Resources Download Publications News

© University of Chicago

Swift proceeds to completion.





PARALLEL SCRIPTING FOR APPLICATIONS AT THE PETASCALE AND BEYOND

Michael Wilde, Ian Foster, Kamil Iskra, and Pete Beckman,
University of Chicago and Argonne National Laboratory

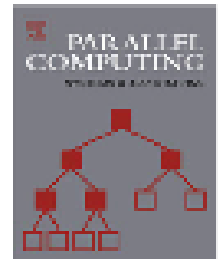
Zhao Zhang, Allan Espinosa, Mihael Hategan, and Ben Clifford, *University of Chicago*

Ioan Raicu, *Northwestern University*



Contents lists available at ScienceDirect

Parallel Computing

journal homepage: www.elsevier.com/locate/parco

Swift: A language for distributed parallel scripting

Michael Wilde^{a,b,*}, Mihael Hategan^a, Justin M. Wozniak^b, Ben Clifford^d, Daniel S. Katz^a,
Ian Foster^{a,b,c}

^aComputation Institute, University of Chicago and Argonne National Laboratory, United States

^bMathematics and Computer Science Division, Argonne National Laboratory, United States

^cDepartment of Computer Science, University of Chicago, United States

^dDepartment of Astronomy and Astrophysics, University of Chicago, United States

ARTICLE INFO

Article history:

Available online 12 July 2011

Keywords:

Swift

Parallel programming

Scripting

Dataflow

ABSTRACT

Scientists, engineers, and statisticians must execute domain-specific application programs many times on large collections of file-based data. This activity requires complex orchestration and data management as data is passed to, from, and among application invocations. Distributed and parallel computing resources can accelerate such processing, but their use further increases programming complexity. The Swift parallel scripting language reduces these complexities by making file system structures accessible via language constructs and by allowing ordinary application programs to be composed into powerful parallel scripts that can efficiently utilize parallel and distributed resources. We present Swift's implicitly parallel and deterministic programming model, which applies external applications to file collections using a functional style that abstracts and simplifies distributed parallel execution.

Acknowledgments

- Swift is supported in part by NSF grants OCI-1148443, OCI-721939, OCI-0944332, and PHY-636265, NIH DC08638, DOE and UChicago LDRD and SCI programs
- The Swift team (including some related projects) is:
 - Mihael Hategan, Justin Wozniak, David Kelly, Ian Foster, Dan Katz, Mike Wilde, Tim Armstrong, Zhao Zhang

Questions?

Supplementary slides

MODIS script: declare data and external science apps

```
type file;  
type imagefile;  
type landuse;
```

```
app (landuse output) getLandUse (imagefile input, int sortfield)  
{ getlanduse @input sortfield stdout=@output ; }
```

```
app (file output, file tilelist) analyzeLandUse  
  (landuse input[], string usetype, int maxnum)  
{ analyzelanduse @output @tilelist usetype maxnum @filenames(input); }
```

```
app (imagefile output) colorMODIS (imagefile input)  
{ colormodis @input @output; }
```

```
app (imagefile output) assemble  
  (file selected, imagefile image[], string webdir)  
{ assemble @output @selected @filename(image[0]) webdir; }
```

```
app (imagefile grid) markMap (file tilelist)  
{ markmap @tilelist @grid; }
```

```
int nFiles = @toint(@arg("nfiles","1000"));  
int nSelect = @toint(@arg("nselect","12")); ...
```

MODIS script: compute land use and max usage

```
imagefile geos[] <ext; exec="modis.mapper",    # Input Dataset  
  location=MODISdir, suffix=".tif", n=nFiles >;
```

```
# Compute the land use summary of each MODIS tile
```

```
landuse land[] <structured_regex_mapper; source=geos, match="(h..v..)",  
  transform=@strcat(runID,"\1.landuse.byfreq")>;
```

```
foreach g,i in geos {  
  land[i] = getLandUse(g,1);  
}
```

```
# Find the top N tiles (by total area of selected landuse types)
```

```
file topSelected<"topselected.txt">;  
file selectedTiles<"selectedtiles.txt">;  
(topSelected, selectedTiles) = analyzeLandUse(land, landType, nSelect);
```

MODIS script: render data to display

```
# Mark the top N tiles on a sinusoidal gridded map
```

```
imagefile gridMap<"markedGrid.gif">;
```

```
gridMap = markMap(topSelected);
```

```
# Create multi-color images for all tiles
```

```
imagefile colorImage[] <structured_regexp_mapper;
```

```
    source=geos, match="(h..v..)",
```

```
    transform="landuse/\\1.color.png">;
```

```
foreach g, i in geos {
```

```
    colorImage[i] = colorMODIS(g);
```

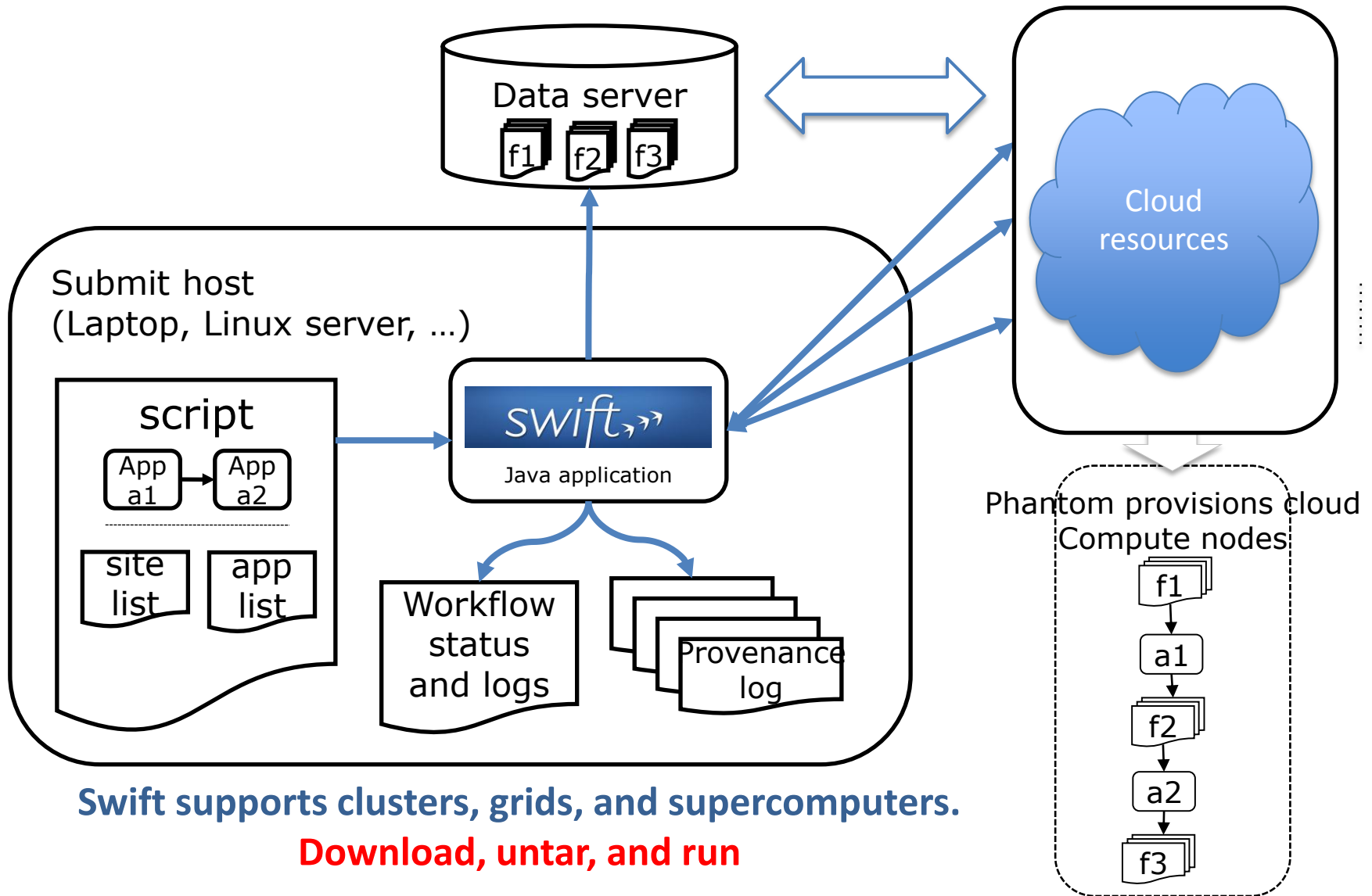
```
}
```

```
# Assemble a montage of the top selected areas
```

```
imagefile montage <single_file_mapper; file=@strcat(runID,"/","map.png") >; # @arg
```

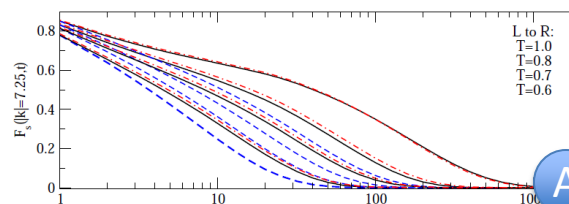
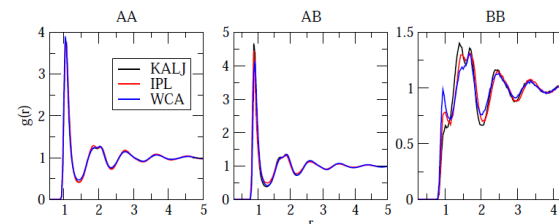
```
montage = assemble(selectedTiles,colorImage,webDir);
```

Runtime to execute Swift apps in the Cloud

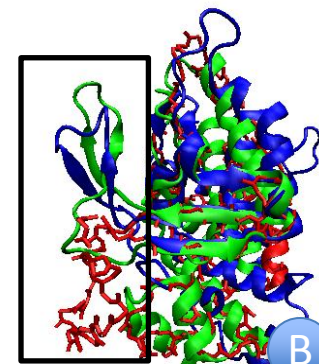


Examples of other Swift many-task applications

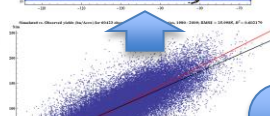
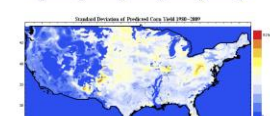
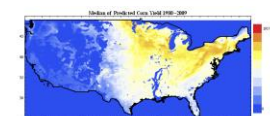
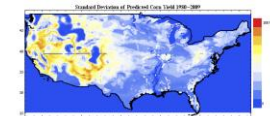
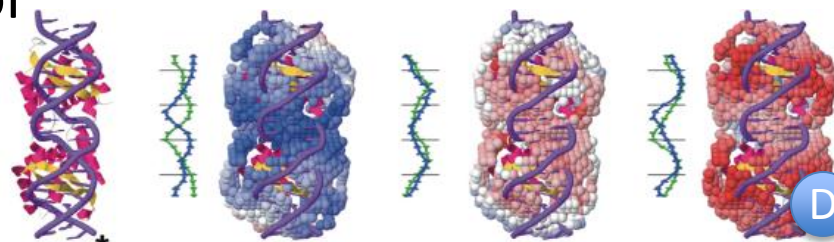
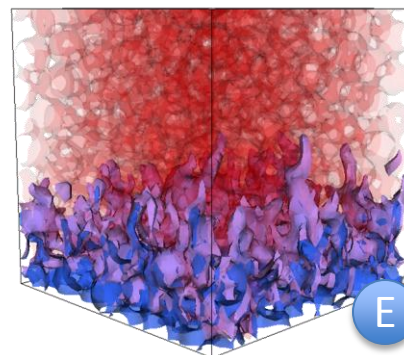
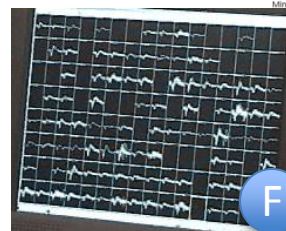
- A Simulation of super-cooled glass materials
- B Protein folding using homology-free approaches
- C Decision making in climate and energy policy
- D Simulation of RNA-protein interaction
- E Multiscale subsurface modeling on Hopper
- F Modeling framework for statistical analysis of neuron activation



T0623, 25 res., 8.2Å to 6.3Å (excluding tail)



Initial
Predicted
Native



Summary

- **Swift is a parallel scripting language** for multicores, clusters, grids, clouds, and **supercomputers**
 - *for loosely-coupled “many-task” applications – programs and tools linked by exchanging files*
 - *debug on a laptop, then run on a Cray system*
- **Swift is easy to write**
 - *a simple high-level functional language with C-like syntax*
 - *Small Swift scripts can do large-scale work*
- **Swift is easy to run:** contains all services for running Grid workflow - in one Java application
 - *untar and run – Swift acts as a self-contained grid or cloud client*
 - ***Swift automatically runs scripts in parallel*** – *typically without user declarations*
- **Swift is fast:** based on a powerful, efficient, scalable and flexible Java execution engine
 - *scales readily to millions of tasks*
- **Swift is general purpose:**
 - *applications in neuroscience, proteomics, molecular dynamics, biochemistry, economics, statistics, earth systems science, and beyond.*



Contents lists available at ScienceDirect

Parallel Computing

journal homepage: www.elsevier.com/locate/parco

Swift: A language for distributed parallel scripting

Michael Wilde^{a,b,*}, Mihael Hategan^a, Justin M. Wozniak^b, Ben Clifford^d, Daniel S. Katz^a,
Ian Foster^{a,b,c}

^aComputation Institute, University of Chicago and Argonne National Laboratory, United States

^bMathematics and Computer Science Division, Argonne National Laboratory, United States

^cDepartment of Computer Science, University of Chicago, United States

^dDepartment of Astronomy and Astrophysics, University of Chicago, United States

ARTICLE INFO

Article history:

Available online 12 July 2011

Keywords:

Swift

Parallel programming

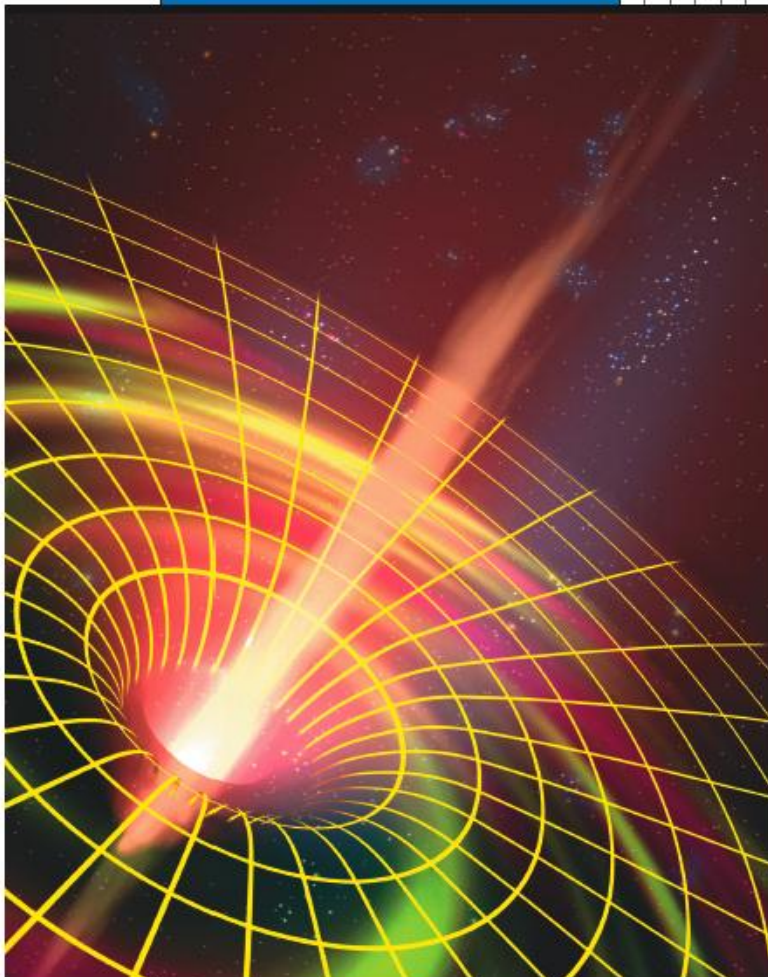
Scripting

Dataflow

ABSTRACT

Scientists, engineers, and statisticians must execute domain-specific application programs many times on large collections of file-based data. This activity requires complex orchestration and data management as data is passed to, from, and among application invocations. Distributed and parallel computing resources can accelerate such processing, but their use further increases programming complexity. The Swift parallel scripting language reduces these complexities by making file system structures accessible via language constructs and by allowing ordinary application programs to be composed into powerful parallel scripts that can efficiently utilize parallel and distributed resources. We present Swift's implicitly parallel and deterministic programming model, which applies external applications to file collections using a functional style that abstracts and simplifies distributed parallel execution.

Parallel Computing, Sep 2011



PARALLEL SCRIPTING FOR APPLICATIONS AT THE PETASCALE AND BEYOND

Michael Wilde, Ian Foster, Kamil Iskra, and Pete Beckman,
University of Chicago and Argonne National Laboratory

Zhao Zhang, Allan Espinosa, Mihael Hategan, and Ben Clifford, *University of Chicago*

Ioan Raicu, *Northwestern University*

Acknowledgments

- Swift is supported in part by NSF grants OCI-1148443, OCI-721939, OCI-0944332, and PHY-636265, NIH DC08638, DOE and UChicago LDRD and SCI programs
- The Swift team (including some related projects) is:
 - Mihael Hategan, Justin Wozniak, David Kelly, Ian Foster, Dan Katz, Mike Wilde, Tim Armstrong, Zhao Zhang