Argonne NATIONAL LABORATORY

Swift: a scientist's gateway to campus clusters, grids and supercomputers

David Kelly
Computation Institute, University of Chicago
and Argonne National Laboratory

Swift project: www.ci.uchicago.edu/swift

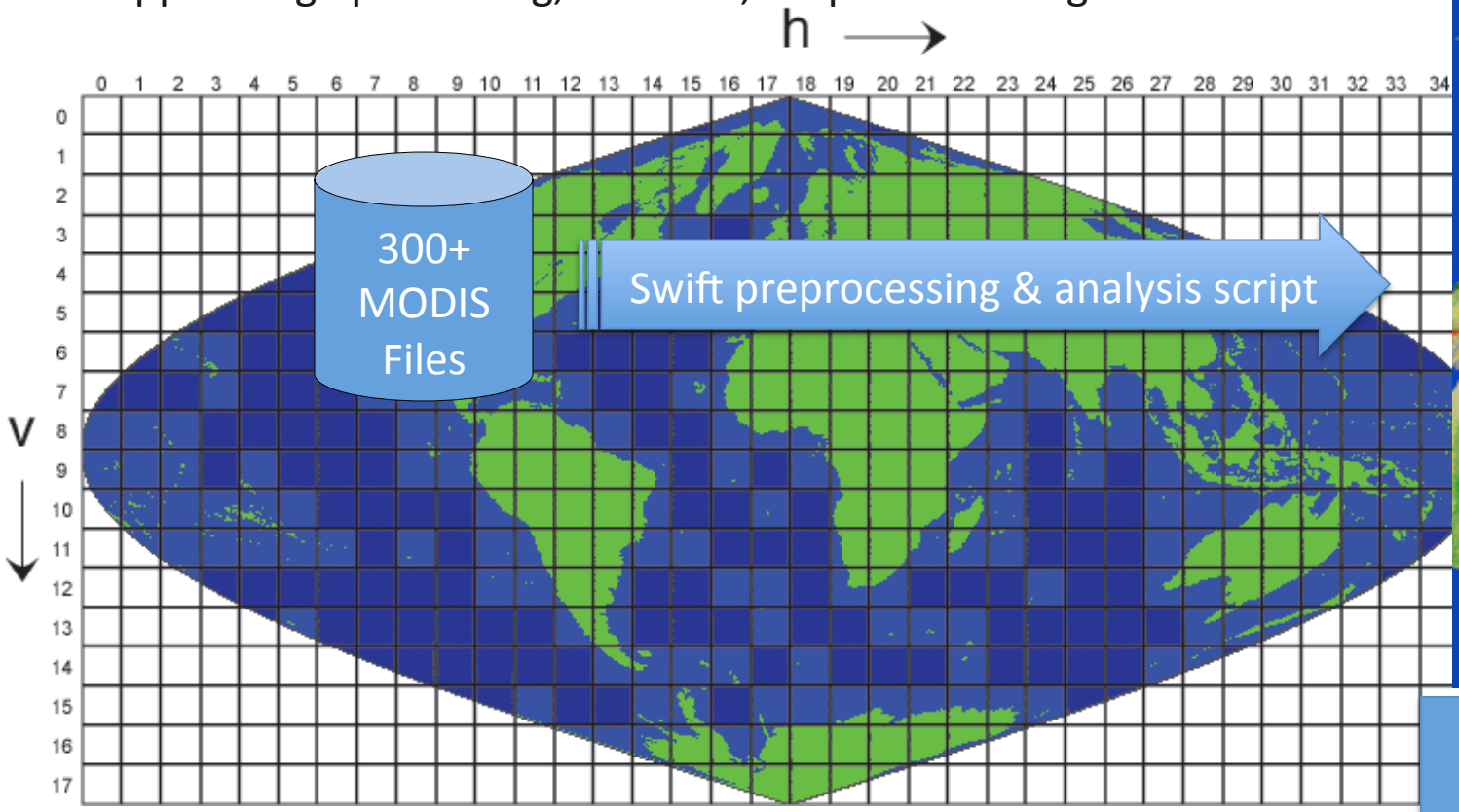
Presenter contact: davidk@ci.uchicago.edu
swift-support@ci.uchicago.edu

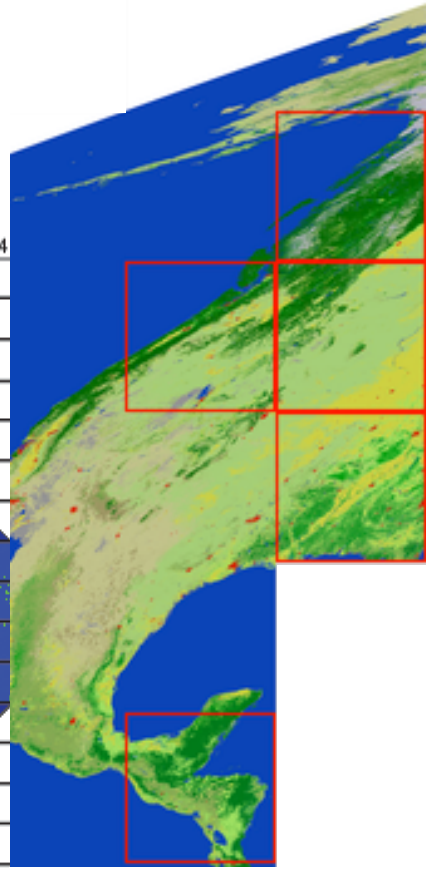
U.S. DEPARTMENT OF ENERGY

- Parallel scripting language for clusters, clouds & grids
  - For writing loosely-coupled scripts of application programs and utilities linked by exchanging files
  - Can call scripts in shell, python, R, Octave, MATLAB, …
- Swift does 3 important things for you:
  - Makes parallelism transparent – with functional dataflow
  - Makes basic failure recovery transparent
  - *Makes computing location transparent* – can run your script on multiple distributed sites and diverse computing resources (from desktop to petascale)
    → *this is what we'll show today*

Example of when you need Swift: Process and analyze MODIS satellite land use datasets

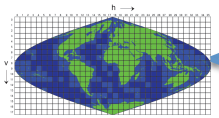
- Input: hundreds of land cover tile files (forest, ice, urban, etc) & ROI
- Output: regions with greatest concentration of specific land types
- Apps: image processing, statistics, output rendering





Largest zones of forest land-cover In analyzed region

Dataflow and applications of MODIS pipeline



landUse
x 317

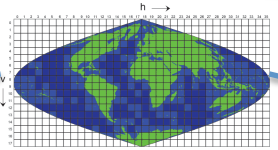
colorMODIS

analyzeLandUse

markMap

assemble

MODIS script is automatically run in parallel:



getLandUse
x 317

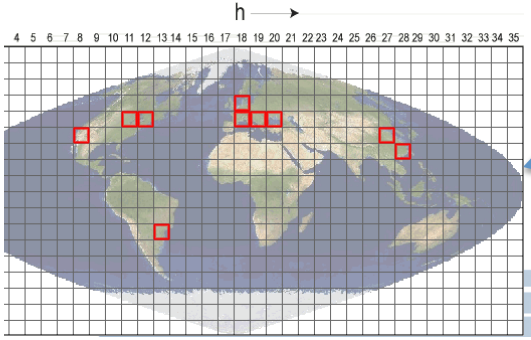
colorMODIS
x 317

analyzeLandUse

assemble

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

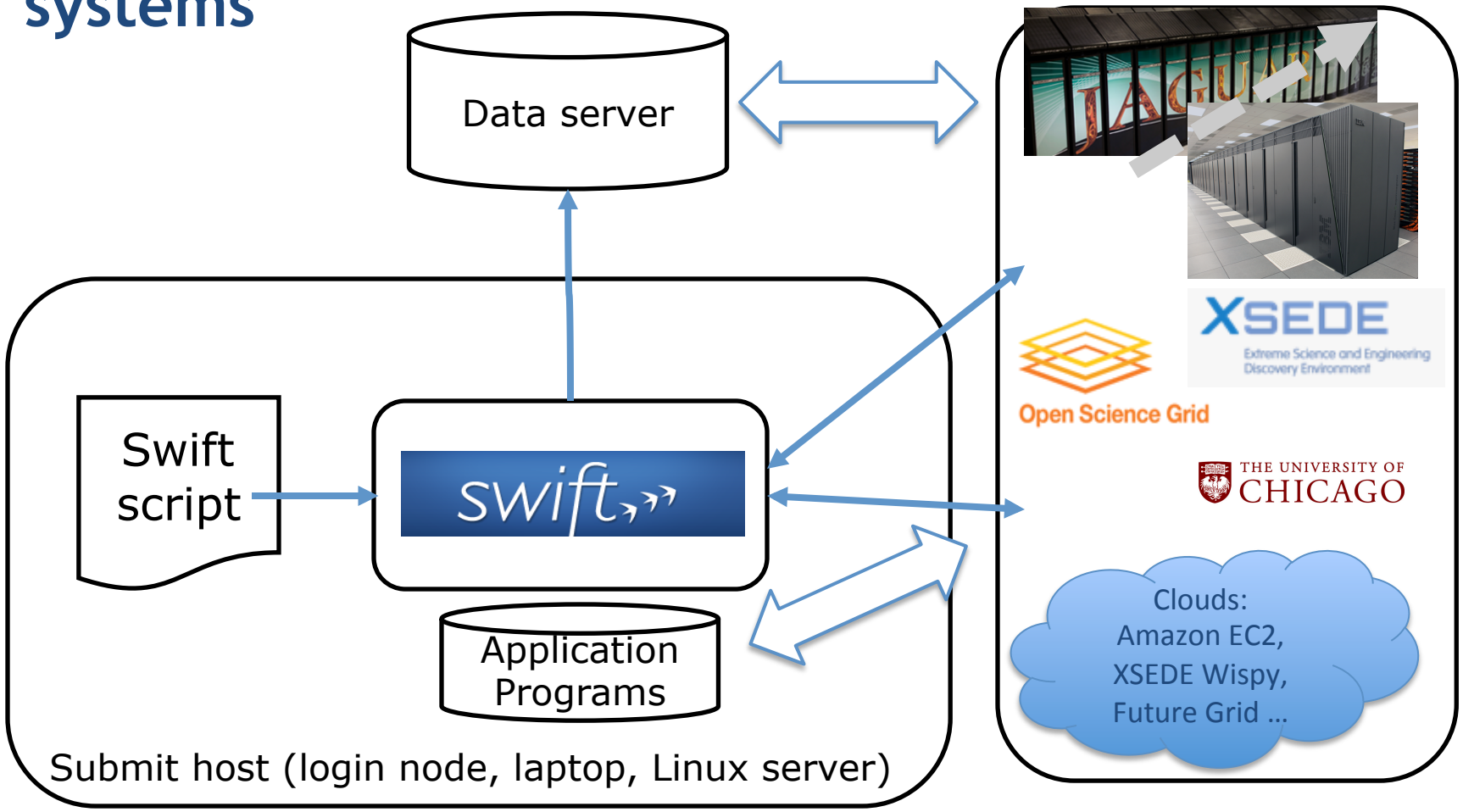
markMap



MODIS script in Swift: main data flow

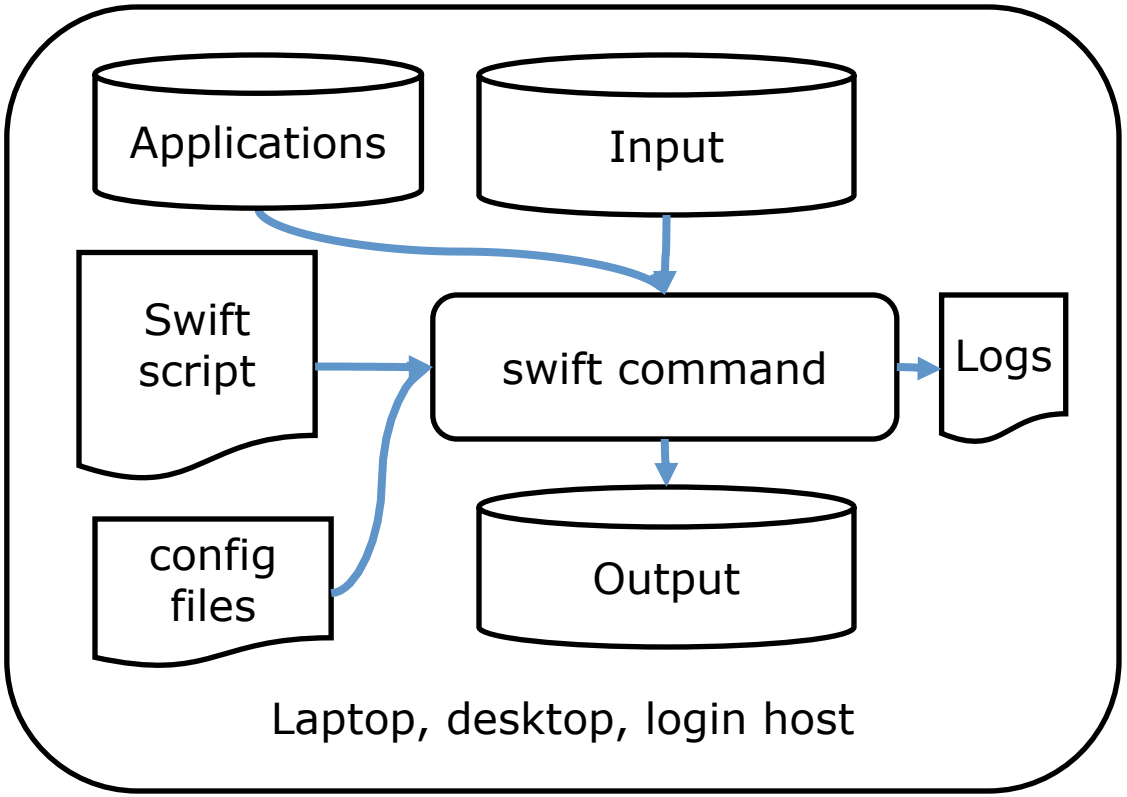
```
foreach g,i in geos {
    land[i] = getLandUse(g,1);
}
(topSelected, selectedTiles) =
    analyzeLandUse(land, landType, nSelect);

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

Swift runs applications on distributed parallel systems



Swift runtime system has drivers and algorithms to efficiently support and aggregate vastly diverse runtime environments

User first tests a new script on a local login host



Swift script is location-independent – debug locally then run distributed

MODIS script excerpt used in this demo

```
$ cat modis.swift

type imagefile;
type landuse;
type perlscript;

perlscript getlanduse_pl <"getlanduse.pl">;

app (landuse output) getLandUse (imagefile input, perlscript ps)
{
  perl @ps @filename(input) stdout=@filename(output);
}

# Constants and command line arguments
string MODISdir  = @arg("modisdir", "../data/modis/2002");

# Input Dataset
imagefile geos[] <filesys_mapper; location=MODISdir, suffix=".rgb">;

# Compute the land use summary of each MODIS tile
landuse land[] <structured_regexp_mapper; source=geos, match="(h..v..)",
            transform=@strcat("landuse/\\1.landuse.byfreq")>;

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

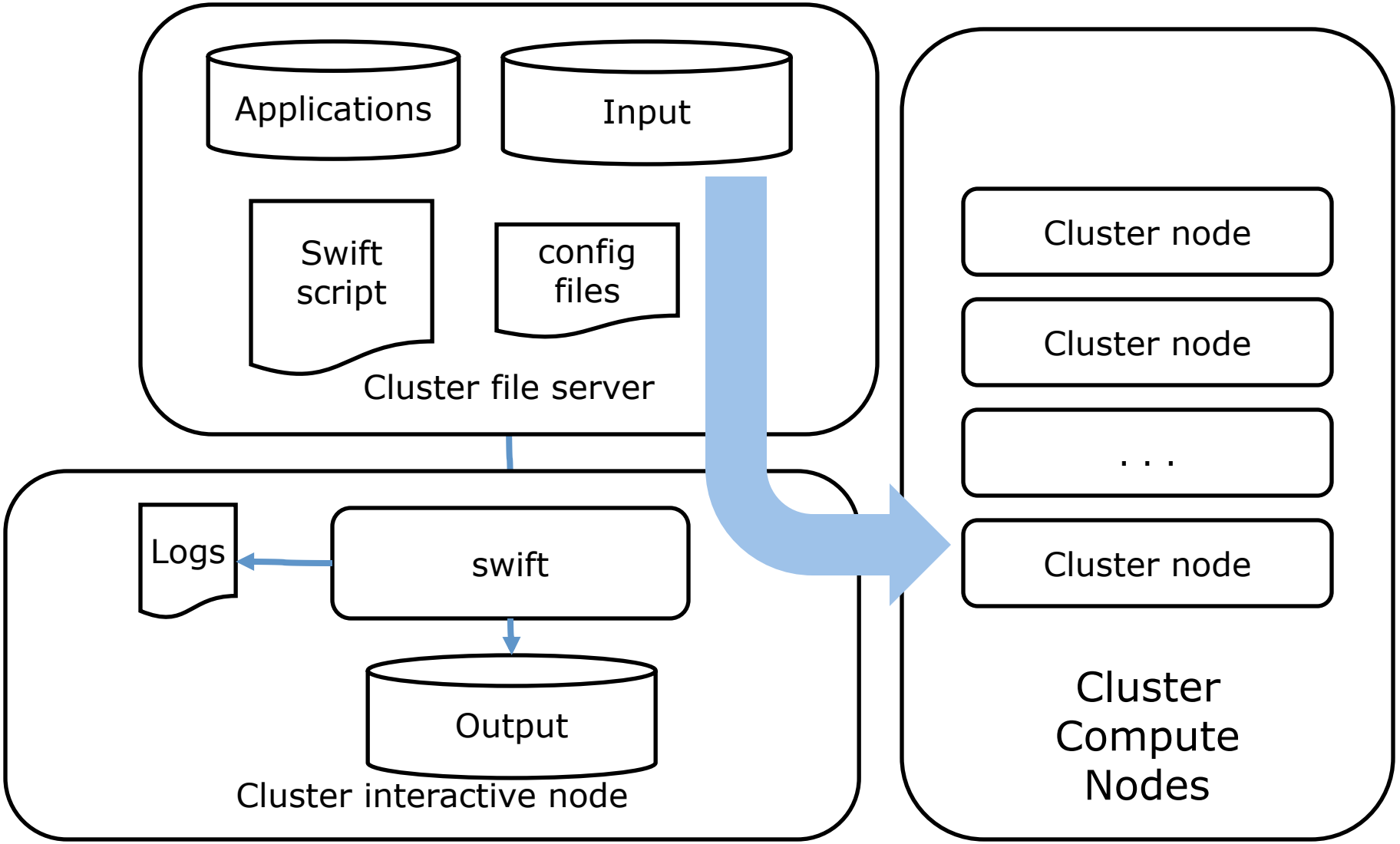
User's Perl app is passed as data

Input dataset is a script parameter

Output filenames are based on inputs

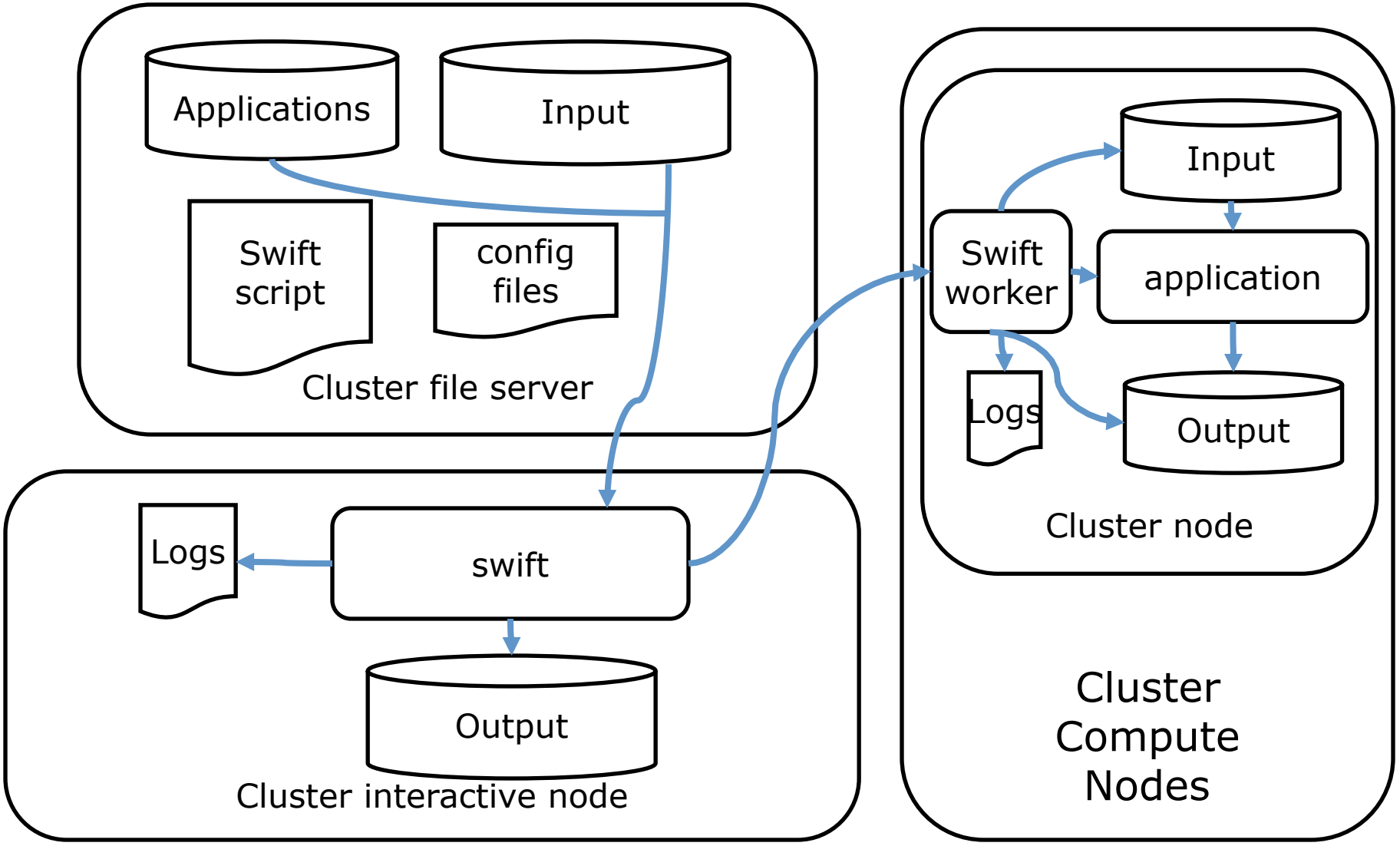
Iteration over the dataset is implicitly parallel

User runs on a campus or department cluster



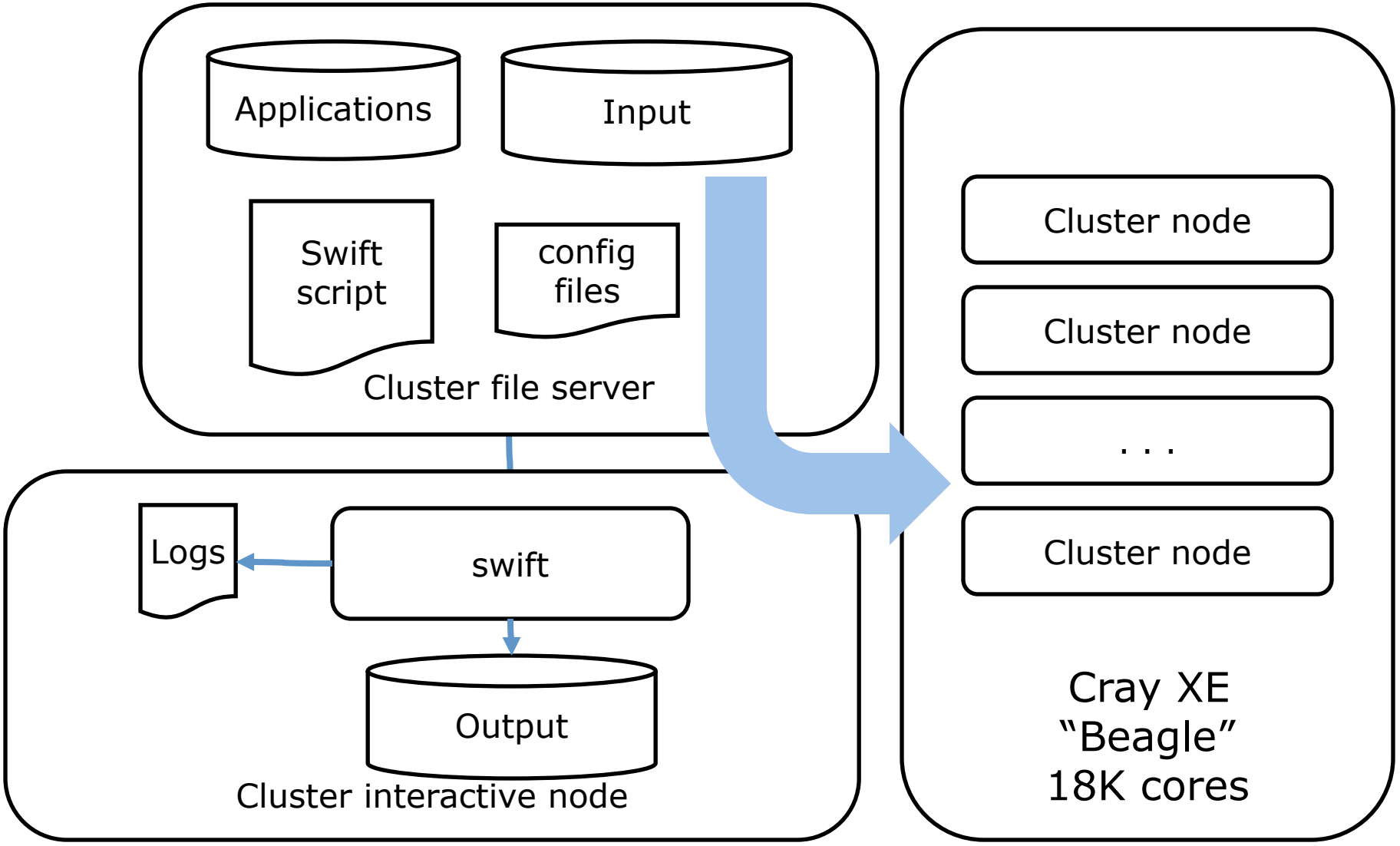
Single-node script scales easily to local cluster

User runs on a campus cluster: what's inside



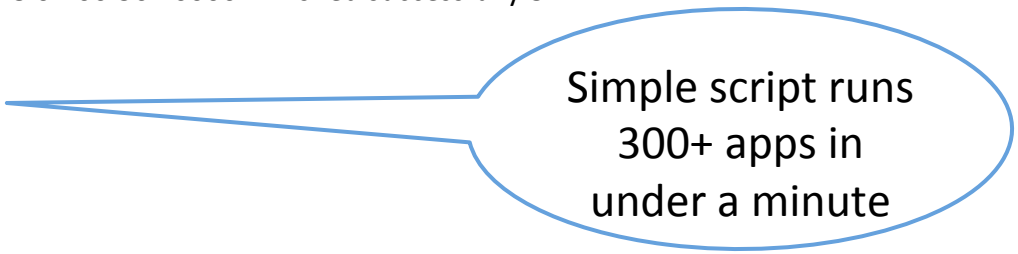
Multiple data streams of data moved from client to worker local FS

Campus or XSEDE supercomputer access is same



Same script runs unchanged between campus research cluster and Cray XE systems

```
$ swift -sites.file sites.xml -tc.file tc.data -config beagle-ssh.cf modis02.swift \
        -modisdir=/home/wilde/osgdemo/modis/svn/data/modis/2002/

Swift trunk swift-r6362 cog-r3637 (cog modified locally)

RunID: 20130311-0159-qayhuq86
Progress:  time: Mon, 11 Mar 2013 01:59:55 +0000
Progress:  time: Mon, 11 Mar 2013 02:00:07 +0000  Selecting site:269  Submitting:47  Submitted:1
Progress:  time: Mon, 11 Mar 2013 02:00:14 +0000  Selecting site:269  Stage in:1  Submitted:47
Progress:  time: Mon, 11 Mar 2013 02:00:15 +0000  Selecting site:269  Stage in:25  Submitted:23
Progress:  time: Mon, 11 Mar 2013 02:00:18 +0000  Selecting site:269  Stage in:47  Active:1
Progress:  time: Mon, 11 Mar 2013 02:00:19 +0000  Selecting site:269  Stage in:29  Active:15  Stage out:4
Progress:  time: Mon, 11 Mar 2013 02:00:20 +0000  Selecting site:239  Stage in:24  Submitting:6  Stage out:17  Finished successfully:31
Progress:  time: Mon, 11 Mar 2013 02:00:21 +0000  Selecting site:221  Stage in:36  Submitting:11  Stage out:1  Finished successfully:48
Progress:  time: Mon, 11 Mar 2013 02:00:22 +0000  Selecting site:221  Stage in:44  Active:1  Stage out:2  Finished successfully:49
Progress:  time: Mon, 11 Mar 2013 02:00:23 +0000  Selecting site:218  Stage in:43  Submitting:3  Stage out:2  Finished successfully:51
…
Progress:  time: Mon, 11 Mar 2013 02:00:43 +0000  Selecting site:30  Stage in:41  Submitted:3  Active:3  Finished successfully:240
Progress:  time: Mon, 11 Mar 2013 02:00:44 +0000  Selecting site:29  Stage in:36  Submitting:1  Active:4  Stage out:7  Finished successfully:240
Progress:  time: Mon, 11 Mar 2013 02:00:45 +0000  Selecting site:23  Stage in:28  Submitting:6  Active:1  Stage out:12  Finished successfully:247
Progress:  time: Mon, 11 Mar 2013 02:00:46 +0000  Selecting site:9  Stage in:39  Submitting:7  Active:1  Stage out:1  Finished successfully:260
Progress:  time: Mon, 11 Mar 2013 02:00:47 +0000  Selecting site:7  Stage in:21  Submitting:2  Active:5  Stage out:20  Finished successfully:262
Progress:  time: Mon, 11 Mar 2013 02:00:48 +0000  Stage in:28  Submitted:1  Stage out:1  Finished successfully:287
Progress:  time: Mon, 11 Mar 2013 02:00:49 +0000  Stage in:15  Active:4  Stage out:7  Finished successfully:291
Final status: Mon, 11 Mar 2013 02:00:50 +0000  Finished successfully:317

real    0m57.478s
user    0m32.923s
sys     0m1.248s
$
```



```
$ cat beagle-ssh.cf

wrapperlog.always.transfer=true
sitedir.keep=true
execution.retries=0
status.mode=provider
use.provider.staging=true
provider.staging.pin.swiftfiles=false

#site beagle-ssh WALLTIME=00:55:00
#app perl=/usr/bin/perl
midway001$
$

$ cat sites.xml
<config>
 <pool handle="beagle">
  <execution provider="coaster" jobmanager="ssh-cl:pbs" url="login4.beagle.ci.uchicago.ed
  <profile namespace="globus" key="jobsPerNode">24</profile>
  <profile namespace="globus" key="lowOverAllocation">100</profile>
  <profile namespace="globus" key="highOverAllocation">100</profile>
  <profile namespace="globus" key="providerAttributes">pbs.aprun;pbs.mpp;depth=24</profile>
  <profile namespace="globus" key="maxtime">3600</profile>
  <profile namespace="globus" key="maxWalltime">00:55:00</profile>
  <profile namespace="globus" key="userHomeOverride">/lustre/beagle/{env.USER}/swiftwork</profile>
  <profile namespace="globus" key="slots">2</profile>
  <profile namespace="globus" key="maxnodes">1</profile>
  <profile namespace="globus" key="nodeGranularity">1</profile>
  <profile namespace="karajan" key="jobThrottle">.47</profile>
  <profile namespace="karajan" key="initialScore">10000</profile>
  <filesystem provider="local"/>
  <workdirectory>/tmp/{env.USER}/swiftwork</workdirectory>
 </pool>
</config>
$
```

Swift moves user's dataset from campus server direct to Cray compute nodes

…and passes Cray-specific PBS parameters

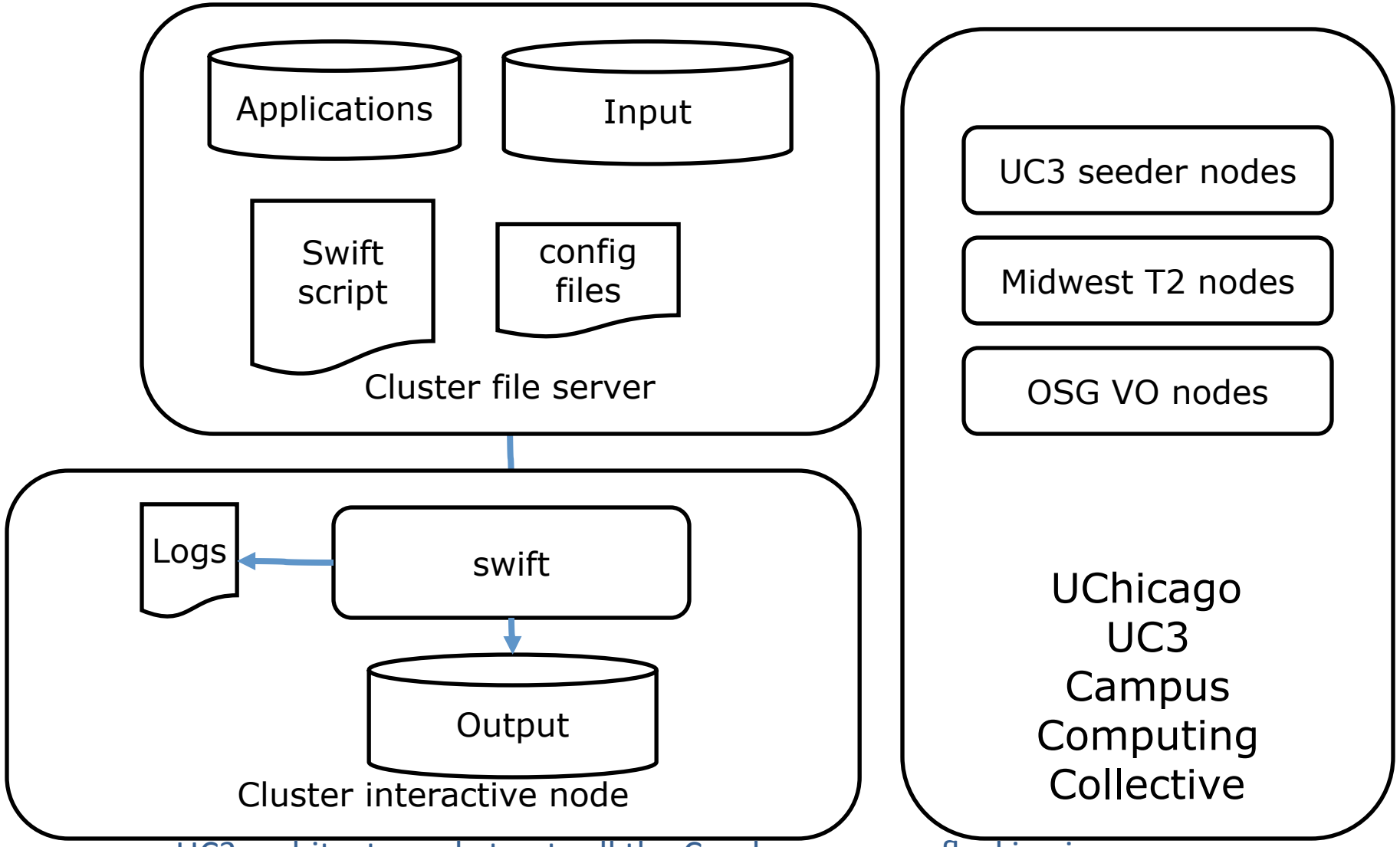
…but most of the site spec is the same as for the campus cluster

```
midway001$ ls landuse
h00v08.landuse.byfreq  h11v10.landuse.byfreq  h17v06.landuse.byfreq  h21v10.landuse.byfreq  h27v10.landuse.byfreq
h00v09.landuse.byfreq  h11v11.landuse.byfreq  h17v07.landuse.byfreq  h21v11.landuse.byfreq  h27v11.landuse.byfreq
…
h11v06.landuse.byfreq  h17v02.landuse.byfreq  h21v06.landuse.byfreq  h27v06.landuse.byfreq  h35v10.landuse.byfreq
h11v07.landuse.byfreq  h17v03.landuse.byfreq  h21v07.landuse.byfreq  h27v07.landuse.byfreq
h11v08.landuse.byfreq  h17v04.landuse.byfreq  h21v08.landuse.byfreq  h27v08.landuse.byfreq
h11v09.landuse.byfreq  h17v05.landuse.byfreq  h21v09.landuse.byfreq  h27v09.landuse.byfreq
midway001$

midway001$ cat landuse/h03v07.landuse.byfreq
211094 0 00
5348 1 01
4376 2 02
3236 3 03
3196 4 04
1242 5 05
731 6 06
405 7 07
292 8 08
225 9 09
83 10 0a
61 11 0b
43 12 0c
39 13 0d
25 14 0e
4 15 0f
midway001$
```

Input is MODIS satellite raster image dataset

Output is histogram of land use codes

UChicago campus "collective" adds OSG resources



UC3 architecture abstracts all the Condor resource flocking issues;
Swift accesses local, MWT2, and OSG as a unified Condor facility using campus user identity

```
midway001$ pwd
/home/wilde/osgdemo/modis/svn/run051
midway001$ cat sites.xml
<config>
 <pool handle="uc3">
  <execution provider="coaster" url="uc3-sub.uchicago.edu" jobmanager="ssh-cl:condor"/>
  <profile namespace="karajan" key="jobThrottle">3.99</profile>
  <profile namespace="karajan" key="initialScore">10000</profile>
  <profile namespace="globus"  key="jobsPerNode">1</profile>
  <profile namespace="globus"  key="maxWalltime">3600</profile>
  <profile namespace="globus"  key="highOverAllocation">100</profile>
  <profile namespace="globus"  key="lowOverAllocation">100</profile>
  <profile namespace="globus"  key="slots">400</profile>
  <profile namespace="globus"  key="maxNodes">1</profile>
  <profile namespace="globus"  key="nodeGranularity">1</profile>
  <profile namespace="globus"  key="condor.+AccountingGroup">"group_friends.{env.USER}"</profile>
  <profile namespace="globus"  key="jobType">nonshared</profile>
  <filesystem provider="local" url="none" />
  <workdirectory>.</workdirectory>
 </pool>

</config>
midway001$
```

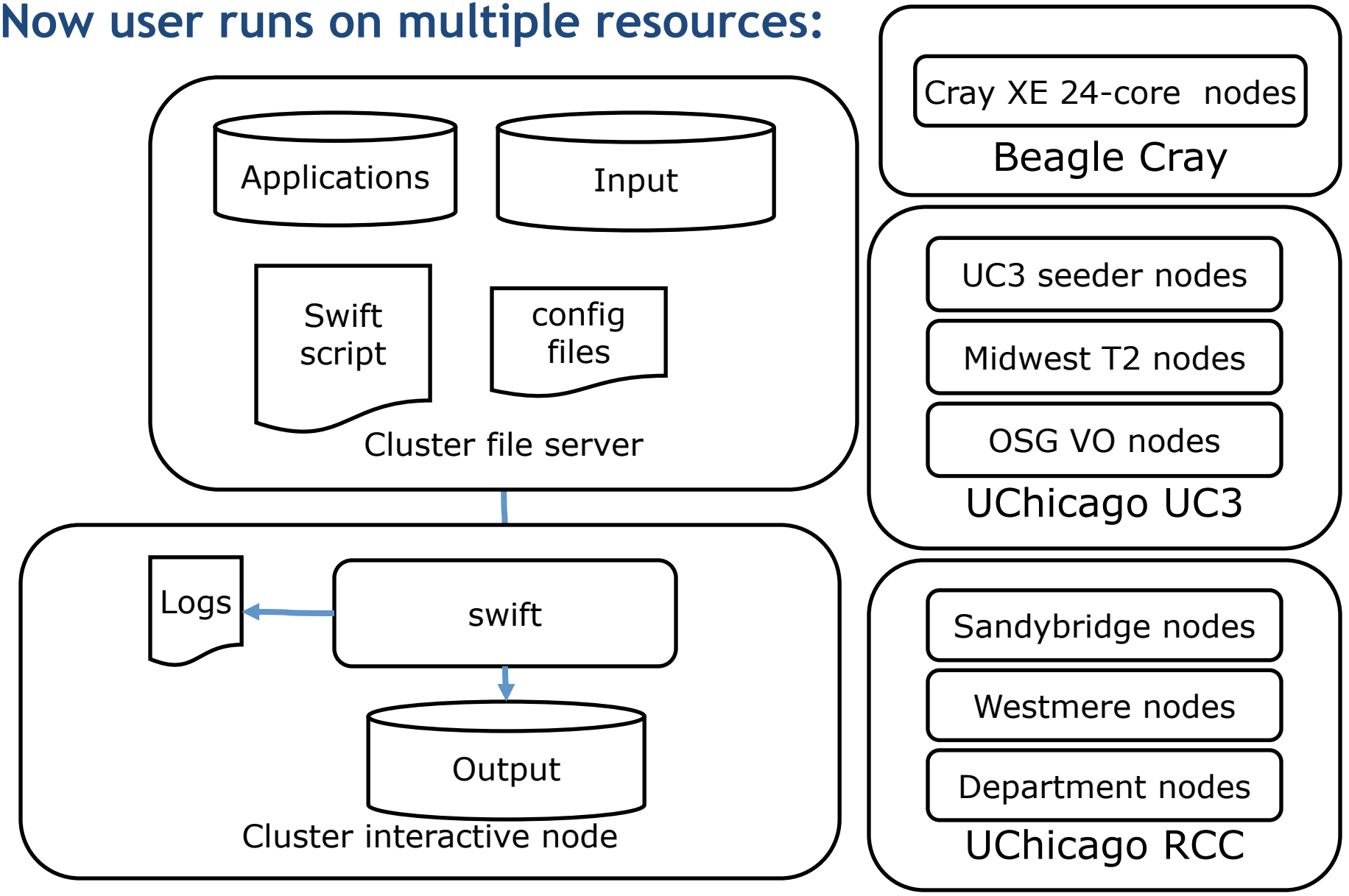
Swift forwards Condor parameters

 # Example of running 1,000 MODIS jobs on just the UC3 collective: local UC3 resources full but work routed to Midwest Tier 2 and OSG

```
$ showsites

midway   0
beagle   0
uc3      0
mwt2     256
OSG      744
Total    1000
```

When local UC3 “seeder” resource full, UC3 flocks to other resources

Now user runs on multiple resources:



Same script runs on broad range of resources; separate throttles can be set for each site.

```
<config>

 <pool handle="uc3">
  <execution provider="coaster" url="uc3-sub.uchicago.edu" jobmanager="ssh-cl:condor"/>
  <profile namespace="karajan" key="jobThrottle">10.00</profile>
  <profile namespace="karajan" key="initialScore">10000</profile>
  <profile namespace="globus"  key="jobsPerNode">1</profile>
  …
  <profile namespace="globus"  key="jobType">nonshared</profile>
  <!-- <profile namespace="globus"  key="condor.+Requirements">isUndefined(GLIDECLIENT_Name) == FALSE</profile> -->
  <workdirectory>.</workdirectory>
 </pool>

 <pool handle="beagle">
  <execution provider="coaster" jobmanager="ssh-cl:pbs" url="login4.beagle.ci.uchicago.edu"/>
  <profile namespace="globus" key="jobsPerNode">24</profile>
  <profile namespace="globus" key="lowOverAllocation">100</profile>
  <profile namespace="globus" key="highOverAllocation">100</profile>
  <profile namespace="globus" key="providerAttributes">pbs.aprun;pbs.mpp;depth=24;pbs.resource_list=advres=wilde.1768</profile>
  …
   <workdirectory>/tmp/{env.USER}/swiftwork</workdirectory>
 </pool>

 <pool handle="sandyb">
  <execution provider="coaster" jobmanager="local:slurm"/>
…
  <workdirectory>/tmp/{env.USER}</workdirectory>
 </pool>

 <pool handle="westmere">
  <execution provider="coaster" jobmanager="local:slurm"/>
  …
  <workdirectory>/tmp/{env.USER}</workdirectory>
 </pool>

</config>
```

Multiple site definitions, managed by support staff

User can specify custom parameters

App list selects where app() run

```
$ cat tc
uc3          perl     /usr/bin/perl null null null
beagle       perl /usr/bin/perl null null null
#sandyb     perl /usr/bin/perl null null null
westmere perl /usr/bin/perl null null null
```

Swift's location-independent scripting lets the user focus on science

- Example of running 3,000 jobs to 3 hosts including the UC3 campus collective:

```
$ ./showsites

midway    289
beagle    1070
uc3       1011
mwt2      295
OSG       335
Total     3000
```

- The user started on a basic login host processing 10 files and moved up to a 3,000 file dataset, changing only the dataset name and a site-specification list to get to the resources above
- Expanded the scope of their computations from one node to hundreds or thousands of cores
- **User didn't need to look at what sites were busy, or adjust arcane scripts, to get to these resources.**

- Swift is a parallel scripting system for grids, clouds and clusters
  - for loosely-coupled applications - application and utility programs linked by exchanging files
- Swift is easy to write: simple high-level C-like functional language
  - 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 – acts as a self-contained Grid client
- Swift is fast: uses efficient, scalable and flexible distributed execution engine.
  - Scales to range of 1M tasks per script run
- Swift usage is growing:
  - applications in earth systems sciences, neuroscience, proteomics, molecular dynamics, biochemistry, economics, statistics, and more.
- Try Swift: www.ci.uchicago.edu/swift

Parallel Computing 37 (2011) 633–652

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]

[a] *Computation Institute, University of Chicago and Argonne National Laboratory, United States*
[b] *Mathematics and Computer Science Division, Argonne National Laboratory, United States*
[c] *Department of Computer Science, University of Chicago, United States*
[d] *Department 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 and PHY-636265, and the UChicago SCI Program
- Extreme scaling research on Swift (ExM project) is supported by the DOE Office of Science, ASCR Division, X-Stack program.
- The Swift team:
  - Mihael Hategan, Justin Wozniak, Tim Armstrong, David Kelly, Ian Foster, Dan Katz, Mike Wilde, Zhao Zhang, Ketan Maheshwari, Yadu Nand
- The UChicago Campus Computing Collective (UC3) team:
  - Rob Gardner, Marco Mambelli, Lincoln, Suchandra Thappa, David Champion,