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:

[NSF logo]
[Department of Energy logo]
[University of Chicago logo]
When do you need high-performance workflow?

example: protein-ligand docking for drug screening

O(10) proteins implicated in a disease

\( \times \)

O(100K) drug candidates

= 1M docking tasks...

...then hundreds of detailed MD models to find 10-20 fruitful candidates for wetlab & APS crystallography

http://swift-lang.org
Expressing this workflow in Swift

For protein docking workflow:

```swift
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.

http://swift-lang.org
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

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”

http://swift-lang.org
Select third-party and open-source applications

Link apps to together to create a workflow

Select compute and storage resources and desired time to solution

Run simulation workflow and visualize results

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

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

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

Current Practice

Solution

The parallel.works Paradigm Shift

Early experience at leading architecture firm SOM
Competitive advantage derives from the integration of—and expertise in—three powerful technologies:

- **Swift**: 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.
Swift gratefully acknowledges support from:

http://swift-lang.org