**The Swift Parallel Scripting Language**

Mihael Hategan, Sarah Kenny, Justin Wozniak, Allan Espinosa, Ian Foster, Michael Wilde
Computation Institute, University of Chicago and Argonne National Laboratory

Swift is a parallel scripting language for large-scale science and engineering tasks. It meets the ever-growing need for running large numbers of application programs quickly – an indispensable approach for analyzing massive quantities of data or performing parameter studies or ensemble simulations.

Swift lets users apply parallel composition constructs to existing sequential or parallel programs to express highly parallel scripts. Swift scripts are compact, flexible and portable, and can run efficiently on platforms ranging from multicore workstations to petascale supercomputers. For performing parameter sweeps and data analysis with existing applications, parallel scripting is easier and more productive than tightly-coupled parallel programming.

A simple functional data-flow-driven language with a compact C-like syntax, Swift enables you to specify these tasks as compact, concise, portable - and highly parallel - scripts. Swift's mappers let you access diverse data formats in a convenient manner, while its Java-based execution engine manages the dispatch of hundreds of thousands of tasks to thousands of processors. Swift scripts developed on multicore workstations can scale to leverage grids, clouds, and petascale supercomputers. Swift users spanning the physical, biological, social and economic sciences save time with reduced software development and faster solutions.

Swift scripts are **easy to write**: its a simple, high-level, C-like functional language. Swift scripts are **easy to run**: Swift contains a complete cluster and grid client in one Java application. Users just unpack Swift and run – a self-contained client contains all code needed for TeraGrid, Open Science Grid, or local cluster execution. The package supports the Argonne Blue Gene/P and TeraGrid Ranger petascale systems. Swift is **fast and highly parallel**, enabling small Swift scripts to do large-scale work. Single Swift scripts have been used to run 500,000 application program invocations.

Most scientists, particularly those who use rather than write computational codes, find that the challenges involved in executing large-scale computing tasks consume tremendous amounts of time and intellectual focus—precious commodities that they would prefer to apply to their core science rather than to the mechanics of computing. The Swift system lets them rapidly and reliably specify, execute, and manage large-scale parallel computations for science and engineering. Swift hides the complexities of manually managing diverse parallel and distributed computing resources by making the computation location-independent. You specify the inputs, outputs and dependencies of your application tools – then Swift selects the resources to run on, transfers datasets and parameters, and manages all tasks.

Here is part of a Swift script for protein-folding using the "ItFix" application. For 10 proteins, nsim=1000, 2 starting temperatures and 5 update intervals, in each round of up to 3 rounds of prediction, this script executes 10 x 1000 x 2 x 5 = 100,000 simulations – and runs on BG/P and Ranger, as well as OSG and TeraGrid resources.

```
foreach prot in protein {
  foreach sT in startT {
    foreach tUp in tUpdate {
      ItFix(prot, nsim, maxrounds, sT, tUp);
    }
  }
}
```

Information on Swift and its use in scientific applications is at: **www.ci.uchicago.edu/swift**

An overview (IEEE Computer, Nov 2009) is at: **www.ci.uchicago.edu/swift/papers/SwiftParallelScripting.pdf**