Final Report on NSF SI2-SSE Award 1148443

Enhancement and Support of Swift Parallel Scripting

PI Name: Michael Wilde

Recipient Organization: University of Chicago

Project/Grant Period: 04/01/2012-03/31/2015

Period of this report: 04/01/2014-9/30/2015

Reformatted report based on NSF final project report submitted October 5, 2015.

Report Contents

1 Report Overview

The goal of this project and the Swift team is to enhance, promote and support the Swift parallel scripting language and its application to increase productivity for performing workflow-style "many task" computing efforts in science and engineering.

This report covers the third (and final) year of this three year SI2-SSE project, and subsequent activities funded by other sources.

The project budget, which was weighted toward the first two projects years, allocated approximately 0.6 FTE to the third and final project year (this reporting period). Hence project activities in this period have focused on user engagement and support with only minimal sustaining development activities.

Support from additional sources was leveraged to both sustain Swift and to support user engagements. These sources include the NSF QuarkNet project, UChicago Computation Institute general funds, the Argonne Leadership Computing Facility, Argonne LDRD funds, the DOE ASCR project "AIMES", efforts contributed by the commercial architecture firm Skidmore Owings Merrill, and efforts contributed by Parallel.Works LLC, a startup supported by the UChicago Innovation Fund.

2 Major goals of the project

The major goals of the project are:

- Engage hands-on with important and strategic users/communities on applying Swift, to identify gaps and prioritize improvement opportunities.
- Propose and conduct tutorials and BoF sessions at important scientific computing conferences (SC, HPDC, e-Science) as well as domain-specific science conferences and meetings.
- Aggressively promote Swift through tutorials and talks within the UChicago community, Midwestern campuses, and beyond.
- Regularly publish both computer science and application papers that highlight Swift's benefits and accomplishments.
- Conduct assessment of user needs in weekly Swift development team meetings to determine development and support actions and priorities.
- Adapt the Swift programming model to specific programming language communities of broad importance to science: Python, R, Octave, and MATLAB, and promote its use within those communities.
- Create a vibrant online user community from which we distill user needs, continuously gathering and tracking requirements and issues.
- Maintain the Swift product roadmap in an open manner, and solicit both continuous and focused user input on it.

3 Major Activities

Progress against the goals for this period that were listed in the prior reporting period are listed briefly here and detailed in subsequent sections:

3.1 Usability improvements to Swift

The source code for the two Swift variants, Swift/K (classic portable Java-based) and Swift/T (high performance MPI-based) have been consolidated under two repositories within a single GitHub organization, https://github.com/swift-lang . This will encourage broader community engagement in Swift development and support, by making the ability to commit code changes globally accessible, and not dependent on obtaining UChicago logins. Usability improvements in each variant are described below.

3.1.1 Swift/K improvements

Four Swift/K versions were developed (and three released as of this date) in this period, with the following highlights:

0.96

- Support for AWS EC2 and Google Compute engine execution providers was added.
- Job cancellation for Swift workers ("coasters") MPI support in Swift workers
- New standard function library, harmonized with Swift/T, and supporting - function overloading
- Improved multi-client support in standalone coaster service with both per-client settings and shared global settings modes
- Structure initializers and sparse array initializers (e.g. array = {"key1": "value1", ...} Syntactic improvements to Swift/K nested if-then-else statements (more "C like") Moved from UChicago SVN repo to github, to enable public write access to committers of any affiliation.
- New unified, user-friendly configuration format that defines both sites, runtime properties and application locations and search paths.
- Added tool displays and replays performance information of Swift runs from logs. Added a shell interface for easy creation/customization of execution providers, to enable flexible cloud resource usage, and make it easier for users who need non-standard local scheduler settings to use Swift.

0.96.1

- Several bugs were fixed in this revision.
- Improved provenance support from UFRJ researchers (Rio, Brazil) was integrated

0.96.2

- Support for the XSEDE Stampede resource was added, including the ability to run MPI apps within a Swift workflow. Additional bug fixes were made.

0.97 (In alpha test by users; not yet released)

- Includes significant memory and speed performance enhancements, and more flexible app task description capabilities (by allowing complete scripts to be specified in-line). Mapping of single files to variables has been made simpler, more concise and more flexible by allowing arbitrary expressions in the default mapping construct.

3.1.2 Swift/T Improvements

Swift/T's build process was further enhanced and simplified; its inter-language external function and application calling mechanism were further enhanced; support for more cluster types (e.g SLURM) were added. Work has been started to add distributed data management capabilities to Swift/T via the Swift worker service. The built-in function library was further enhanced, and functions were added to provide more control over the ability to place tasks close to their input data objects.

Four Swift/T versions were released in this period, with the following highlights:

0.5.1

- Static build capability for significant performance improvement with high node counts
- Minor output fix

0.6.1

- Checkpointing of swift-t runs for recovery after run failures
- New data object container features, including faster and more flexible arrays Pushed data dependencies into ADLB - eliminates the separate Turbine "rule engine" processors
- (Reduces process overhead of Turbine engines) Support for Julia as an embedded scripting language

0.7.0

- Support for remote execution via Coasters
- New "swift-t" tool combines "stc" (complier) and "turbine" (executor) commands for single command compile-and-run
- New "soft" node targeting feature directs app calls to specific nodes for data-intensive scripting applications
- Ability to retry failing app function invocations
- More flexible standard output options (TURBINE_OUTPUT) STC compiler output is now in a *.tic file by default

0.8.0

- Open code syntax in Swift: main{} function is no longer required Advanced targeting modes for data intensive applications
- New string format operator
- Initial support for Swift/* standard library (language convergence with Swift/K) MPI variable sized leaf tasks (capabilities of launching parallel leaf tasks with dynamically settable process counts) on Blue Gene/Q.

3.1.3 Additional Development Activities

In addition to these releases, the following activities were conducted:

- Code coverage for the Swift/T test suite was analyzed (for the stc compiler): check to see how much of the STC codebase the test suite actually exercised. The results showed over 80% coverage, and that much of the remainder was disabled trace level logging statements. Some browsable sample results are here: http://people.cs.uchicago.edu/~tga/stc-test-coverage/ Tool used: http://www.eclemma.org/jacoco/

- Mechanisms were added to Swift/T to enable easier invocation of application codes by integrating entire applications as in memory functions called without using fork/exec, to achieve high performance on petascale platforms. This technique is referred to as "Main Wrapping" because it allows entire apps to be invoked through their command line interface (ie, through the function call "main(argc,argv)" but without spawning a new fork/exec process.

- The integration of Swift/T with the flexible Swift resource allocation service (the worker service, internally referred to as "coasters") was further enhanced with flexible configuration mechanisms to specify resource allocation units. This is nearing production readiness, and will bring the project closer to having a single Swift implementation

(effectively merging Swift/K and Swift/T releases into a single unified system with reduced support costs).

3.2 Documentation and Promotion/Engagement

Enhanced Swift documentation with a focus on self-paced tutorial material.

The tutorials were refactored to separate core language and distributed programming methods from site-specific issues related to particular cluster schedulers or data access conventions. This improved version was used at the XSEDE 2015 conference. A major new user guide rewrite has been completed. This revised guide improves the manner in which the language is explained for new users, and provides a more complete and better-organized working reference to the language's details for experienced users. The guide was also updated to match the newest improvements in the Swift configuration process.

Expanded promotional and engagement focus to broader reach activities that can more rapidly expand the user community.

Swift was increasingly promoted within the extreme scale community. Evidence that its being increasingly exposed is seen in the fact that national and international collaborations are increasingly contacting us and evaluating Swift, including Caltech's materials science program and their collaboration with the University of Washington and Columbia University, the UK-based Genome Analysis Center (TGAC), and the international Square Kilometer Array project (SKA). We will publish an open development roadmap on the Swift website to further engage a broader community of users and contributors.

4 Summary of Progress Against Specific Project Goals

The overall project goal is to enhance Swift usability and performance, engage with user communities, and aid them in applying Swift to meet workflow-related computing needs in science and engineering.

GOAL: Engage hands-on with important and strategic users/communities on applying Swift, to identify gaps and prioritize improvement opportunities.

RESULT: Engaging with and supporting users has been our major activity in the third and final year of this project. These engagements are summarized below.

GOAL: Create and maintain effective "starter kits" that attract prospective users to quickly try Swift and experience its benefits and ease of use.

RESULT: Self-paced tutorials were developed catering to different user communities based on domain science, execution platform and user expertise. The tutorial package (http://swift-lang.org/swift-tutorial/doc/tutorial.html) was significantly improved, and runs on numerous local campus systems at UChicago and Argonne, Amazon and other cloud resources, XSEDE, OSG, as well as on Cray systems. A separate tutorial has been developed for the IBM Blue Gene Q systems.

The Web-based "Try Swift" interface (http://swift-lang.org/tryswift) gives users the ability to try several tutorial scripts on a small set of virtual machines, and experiment with Swift code.

GOAL: Regularly publish both computer science and application papers that highlight Swift's benefits and accomplishments.

RESULT: Posters, papers, and tutorials were held at various regional, national and international venues.

The project has a good publication record, as reported on the Swift Web site and in a later section of this report.

GOAL: Adapt the Swift programming model to specific programming language communities of broad importance to science: Python, R, Octave, and MATLAB, and promote its use within those communities.

RESULT: Swift is able to run "leaf function" scripts in all of these languages, many of which are used extensive in neuroscience workflows (MATLAB), and in the scattering science user community (Python and MATLAB), and earth systems science (Octave). The newer HPC-focused Swift/T release has the ability to call in-memory functions written in Python, Tcl, R and Julia. The ability for Swift/K to run on Hadoop was implemented as an experimental facility for the UChicago Knowledge Lab to use for large-scale text analysis and data mining.

GOAL: Conduct assessment of user needs in weekly Swift development team meetings to determine development and support actions and priorities.

RESULT: Regular weekly Swift group meetings have been held on Fridays at 2PM Central, for most of the duration of this project. The entire group attends this meeting on a regular basis, typically from multiple sites. This meeting has become an effective part of group culture. The meeting has held approximately 85% of the time on a weekly basis. User needs and open trouble tickets are addressed between experts in different Swift subsystems. New feature development plans are discussed. Brainstorming of new product directions often takes place at these meetings.

GOAL: Create a vibrant online user community from which we distill user needs, continuously gathering and tracking requirements and issues.

RESULT: A mailing list that enables discussion between the Swift group and users community is gradually gaining in usage, but many Swift users still communicated with the group through direct private conversations and through a trouble-ticket system that we are deprecating in order to maximize discussion on the community-wide forums.

We have moved the Swift source code for both Swift/K and Swift/T to GitHub repositories under the swift-lang GitHub organization, and are planning to consolidate the Swift/T and Swift/K community communication lists, leveraging GitHub as an accessible and open platform for international community engagement.

GOAL: Maintain the Swift product roadmap in an open manner, and solicit both continuous and focused user input on it.

RESULT: We have moved the Swift email lists from their ci.uchicago.eu domain to swift-lang.org to further encourage community participation. The Swift/T project has from its inception used an open discussion forum (Google Issue Tracker at https://code.google.com/archive/p/exm-issues/issues) for feature design discussion and implementation.

We plan to post a comprehensive product development and evolution roadmap for community feedback and engagement by Nov 2015.

GOAL: Propose and conduct tutorials and BoF sessions at important scientific computing conferences (SC, HPDC, e-Science) as well as domain-specific science conferences and meetings. Aggressively promote Swift through tutorials and talks within the UChicago community, Midwestern campuses, and beyond.

RESULT: Thirty four outreach talks and meetings, poster sessions, and tutorials on Swift and related scientific workflow topics have been presented and/or held, as listed below.

5 User Engagement within Science and Engineering Communities

In this period, the Swift user base has expanded and many new regular users were engaged, 25 of which are listed here.

1) The UChicago Knowledge Lab (KL) seeks to develop a refined picture of the social and material organization underlying the innovations contained in the academic literature. Its research benefits, from Swift's large-scale parallelization, which is used by six KL researchers.

KL developed Swift libraries for text analysis methods including *n*-gram counting, TF*IDF rankings and K-L divergence scores. Swift's ability to run these over Hadoop enabled KL to take advantage of a large pool of resources with low contention, without reprogramming scripts to run directly on Hadoop. One summer student mined 2.5 TB of Wikipedia history using AWS spot instances, with only a few weeks of effort. Swift enabled KL researchers to run analyses with varying parameters on multiple folds of a dataset of 8.2 million biomedical abstracts. KL leveraged Swift's ability to run on different environments without changing scripts, including *OSG Connect*. KL NSF grants that used Swift: 1158803, 1422902, 0915730.

2) The multidisciplinary NSF-supported "Center for Robust Decision Making on Climate and Energy Policy" (RDCEP) performs climate-energy-economics modeling with climate and economic scientists from eight institutions. This collaboration has employed Swift over the past three years to execute a growing set of models of agricultural, economics, climate and energy-related factors affecting the global food supply. RDCEP has published over eleven journal and conference papers from 2011-2015 based on the results from Swift-driven computations (http://www.rdcep.org/publications).

3) For the NSF project "Understanding the Consequences of Water-Use Decisions in a Dynamic Environment" (award 1114851, University of New Hampshire), C. H. Chow of the Agent-based Modeling Group developed a parallel Repast agent-based model using Swift to simulate the effect of social and organizational factors on water use in Arizona. For the NSF project "RAPID: Designing Ebola Interventions for Large Urban Areas through Agent-Based Modeling and Network Analysis" (award 1516428, UChicago), the ABM Group is using Swift/T to model the behavior of global epidemics. This is a large-scale model optimization workflow using 16K-64K cores, and involving J. Ozik, N. Collier, C.H. Chow, J. Wozniak, and NSF REU student J. Burge).

4) T. D. Schiebe and colleagues at the Environmental Molecular Sciences Laboratory use Swift in hybrid multi-scale workflows to model subsurface materials flows with reactive transport. They simulate multiphase flow, solute and energy transport, geochemical reactions, geomechanical effects, and multi-organism microbial communities. This research, supported by DOE and NSF, was reported in ICCS 2015 and Groundwater Vol. 53 (NGWA.org).

5) Researchers K. Takahashi and K. Balasubramanian in the UChicago Neuroscience lab of N. Hatsopolous have used Swift extensively for neurophysical animal studies by analyzing EEG data from experiments, driving MATLAB workflows with Swift on the UChicago Cray Beagle system. From 2011 through 2015, this research, supported by NSF and NIH, has published five journal papers and a conference abstract, and presented eleven talks, based on Swift computations.

6) The Swift team engaged with two research groups in scattering science at the Argonne Advanced Photon Source (APS). One group is studying crystal structure via diffuse scattering. Another group is studying metal grain structure and defect behavior through high energy diffraction microscopy (HEDM). Swift workflows are used to perform HEDM near-field and far-field analysis at the beamline, as well as speeding post-processing for tomography and powder diffraction. This benefits NSF researchers such as the labs of R. Suter and A. Rollet from CMU.

7) The labs of J. De Pablo of UChicago's Institute for Molecular Engineering and J. Whitmer of the

Chemical Engineering, Notre Dame are using Swift/T to explore advanced sampling methods for molecular dynamics simulations, driving LAMMPS from Swift/T. Their work, which involves techniques such as configurational bias, expanded ensembles, parallel tempering, density-of-states sampling, flux-tempered metadynamics, and radial basis function sampling, is supported in part by NSF and DOE.

8) J. Lin and O. Heinonen of Caltech evaluated Swift to perform ensembles of electronic structure computations for ab initio molecular dynamics. This exploratory work led the way to a proposed NSF S2I2 collaboration between Caltech, UWashington, and Columbia University that will use Swift as the basis for a collaboration-wide workflow framework in materials science.

9) G. Gopalakrishnan (U. Utah) and M. Burtscher (Texas State) evaluated Swift for use in an NSF-supported XPS grant (1438963, 1439002) exploring reliability in extreme-scale programming models. http://synergy.cs.vt.edu/2015-nsf-xps-workshop/posters/Ganesh_Gopalakrishnan_56-XPS-PI-Meeting.pdf

10) The South Korean Institute of Science and Technology Information (KISTI) is using Swift to enable their "High Throughput Computing as a Service" system (HTCaaS) to run over their distributed national HPC infrastructure. KISTI has Integrated Swift with HTCaaS, which acts as a task distributor and manager for large numbers of tasks over highly distributed computing infrastructures (multiple sites spread across S. Korea, much like a smaller-scale Open Science Grid).

11) The Exascale Co-Design center for Materials in Extreme Environments (ExMatEx) uses Swift/T to implement their focal applications in the area of task-based adaptive sampling for scale-bridging materials simulation. In this work, the constitutive response of a coarse-scale material volume element is obtained from dynamically spawned fine-scale simulations rather than from a closed-form analytic or pre-computed model. Swift will be used to apply this method to a new Task-based Scale-Bridging Code on the Trinity supercomputer.

12) A. Gel of the National Energy technology laboratory is using Swift to execute large-scale uncertainty-quantification studies of coal gasification plant simulations of transient reactive flows, on Blue Gene/Q and Cray XE and XC systems.

13) T. A. Binkowski of UChicago and the Center for Structural Genomics of Infectious Diseases used Swift on Blue Gene/Q to run large-scale runs of the Rosetta application to perform flexible peptide docking.

14) A materials science team at Argonne/UChicago/Northwestern led by O. Heinonen and P. Zapol is using Swift to run a workflow that ties together molecular dynamics simulations with electronic structure calculations into a massively parallel multi-scale computational framework.

15) S. Som of the Center for Transportation Research is using Swift to run advanced engine simulations with sensitivity analysis and identify important variables influencing particular engine performance targets. These workflows perform large-scale uncertainty quantification combustion engines models on the Blue Gene/Q. A Swift framework is being constructed to execute a 60M-hour UQ modeling campaign.

16) Swift was used to analyze large-scale next-generation sequencing datasets for the Institute for Genomics and Systems Biology (J. Pitt, K. White). The "SwiftSeq" pipeline runs at unprecedented scale (over 5M CPU hours and often at 10K+ cores in parallel) on Cray and cloud systems, and will be made broadly available. (http://beagle.ci.uchicago.edu/science-at-beagle/#health1 and http://beagle.ci.uchicago.edu/files/2014/05/may_newsletter_2014.pdf)

17) The international architecture firm Skidmore Owings Merrill (SOM) expanded its use of Swift in building energy modeling and urban planning studies, in collaboration with the UChicago Urban Center for Computation and Data (UrbanCCD).

18) M. Hutchinson of UChicago is performing spectrally converged direct numerical simulations of the

Rayleigh-Taylor instability in turbulent flows, in a project led by R. Rosner, using Swift/K on the Blue Gene/Q at up to 512K cores. Swift conveniently scripts multi-week workflows with checkpoint capabilities, in a manner that was awkward and complex with ad-hoc shell and Python scripts. It runs workflows that coordinate BG/Q simulation with an external analysis cluster.

19) M. Mookerji, Cornell Geosciences, is using Swift to explore high pressure behavior of Fe3S using VASP, for modeling mineral transitions in planetary cores. An abstract has been submitted to the American Geophysical Union Conference Dec. 2015.

20) At The Genome Analysis Centre (TGAC), Norwich, UK, T. Stitt is applying Swift/T to sequence alignment task farming.

21) A. Mazurie of Montana State University is using Swift for bioinformatics workflows, and is evaluating the use of Swift as a classroom tool for teaching parallel programming.

22) The Swift team engaged with IIT CS researchers to provide performance results for their HPDC 2015 poster "Benchmarking the State-of-the-art Runtime Systems of Many-Task Computing, Ke Wang and Ioan Raicu, Illinois Institute of Technology). Swift was the top in performance among the four measured systems, which included Charm++.

23) G. Bronevetsky of LLNL (now at Google), used Swift to model failure modes in extreme-scale parallel applications and programming models.

24) Argonne Applied Math group scientist V. Zavala of the MCS division is using Swift to manage workflows within the Galaxy portal for power grid optimization problems expressed using JuMP, the Julia Mathematical Programming language.

25) The NERSC scientific data management team (S. Choleas, M. Prabhat) adopted Swift as a supported workflow solution for its petascale Cray systems Edison, Hopper, and the future Cori. https://www.nersc.gov/users/data-analytics/workflow-tools and https://www.nersc.gov/users/data-analytics/workflow-tools/swift/

6 Progress against target metrics

For the metrics that were proposed in the original 2011 proposal and subsequent award, we report the following:

Number of communities actively using Swift to derive science results: At least 20 active communities as of this report.

Number of individuals running Swift: in the period 8/2014 – 6/2015, we observed > 2400 unique swift users, based on IP address and domain name. These users ran > 20,000 scripts, and > 7500 scripts of over 5 minutes duration.

Scientific publications based on Swift computations: For the projects reported here and in prior reports, we are aware of the following publications based on work that utilized Swift: 11 publications from the RDCEP project, 5 publications from the Hatsopolous neuroscience Lab, 4 publications from the Sosnick/Freed Biophysics Lab, 1 publication from the Voth/Dinner chemistry Lab, 5 publications from the Reichman chemistry Lab, 2 publications from the Agent Based Modeling Group, 3 publications from the Zapol/Heinonen materials science team, 4 publications/posters/talks from the Institute for Genomics and Systems Biology, 1 publication from the Schulten biophysics Lab, 1 publication submitted by Mookerjee (Cornell planetary science), 3

publications from the Environmental Molecular Ssciences Lab, 2 publications from NCAR and the ParVis parallel visualization project, 1 publication by SOM from the Urban Center for Computation and Data, 5 publications by UFRJ (Brazil), 3 publications pending by APS users in high energy diffraction microscopy. We are uncertain of publications pending or published by the Knowledge Lab, but suspect that several have been published and/or submitted.

7 Training and professional development provided by the project

T. Armstrong (UChicago CS PhD student) played a leading role in Swift development, focusing on Swift/T and extreme scalability. Armstrong completed his Ph.D, in 6/2015 and continues to participate in the Swift project from his new position at Cloudera. His dissertation comprises a detailed design description, rationale, and performance evaluation of Swift/T. His work was published at SC14 and in an upcoming book chapters in a new MIT press book on parallel programming models.

Yangxinye Yang (UChicago, CS undergraduate, female) has continued to engage with the Swift team to enhance and develop Swift configuration capabilities.

Peter Vilter of UChicago did his undergraduate thesis in part on Swift, developing a visual programming model for Swift's style of parallel functional dataflow programming:

https://docs.google.com/presentation/d/19oM5iH6eHpxGSPq8d3NrCeqqdNWBx45QPAPJh2bOt80/edit#slide=id.p

For an IIT computer science class in cloud computing (CS553) – students received a lecture on Swift and completed a programming problem on Amazon EC2 using Swift.
http://www.cs.iit.edu/~iraicu/teaching/CS553-F14/CS553-F14_syllabus.pdf

The Swift team hosted 4 REU students in 2015. These students, three men and one woman, worked on Swift projects that include development of a Swift interpreter ("REPL-mode" interactivity, by B.Subei); integration of the Fusion-FS filesystem for big-data processing (M. Dupres); refinement of the Swift/T support for flexible resource allocation and remote data management (J. Taylor); and large-scale HPC workflows to perform agent-based modeling of global epidemics (J. Burge).

M. Wilde presented a hands-on training session on Swift at the XSEDE 2015 conference as part of a short class on XSEDE workflow techniques that included Swift, MakeFlow and Pegasus.

The Swift team participated in the Argonne Training Program for Extreme Scale Computing in Aug. 2013, 2014, and 2015, presenting two lectures on Swift and scientific workflow at supercomputer scales to over 60 students, with hands-on exercises. For the 2015 class, M. Wilde presented general Swift concepts, and J. Wozniak covered Swift/T and provided hands-on exercises.

Three high school summer students working at the UChicago Knowledge Lab in June-Sep 2015 have used Swift to perform text analysis workflows, and have become enthusiastic members of the Swift user community, developing useful KL data analysis studies.

8 Outreach Tutorials, Talks, Meetings, Poster Sessions

1. UChicago Faculty Technology Day, Swift Poster, M. Wilde, 2014.0416
2. "Swift: a scientist's gateway to campus clusters, grids and supercomputers", M. Wilde,

XSEDE Workflow Symposium Presentation (hosted by Gateway community) 2014.0425

3. Lecture on Swift and parallel dataflow programming to UChicago Computer Science course, "Data-intensive Computing Systems" (Andrew Chien, professor), M. Wilde. 2014.0506
4. "GeMTC and Swift: Implicitly-parallel functional dataflow for productive hybrid programming on Blue Waters". M. Wilde, I. Raicu. Blue Waters Symposium, Urbana, IL 2014.0514
5. "Swift: implicitly parallel programming from multicore to petascale", M. Wilde, Greater Chicago Area Systems Research Workshop (GCASR), 2014.0519
6. Talk on Swift and Provenance for Information intensive research initiatives at the University of Chicago, Panel, Information, Interaction, and Influence University of Chicago/Digital Science Workshop on Research Information Technologies and their Role in Advancing Science. 2014.0520
7. "Towards Dynamic Dataflow Composition for Extreme-Scale Applications with Heterogeneous Tasks", T. Armstrong, Joint Lab for Extreme Scale Computing, Nice, France. 2014.0609.
8. "Case Studies in Big Data and HPC from X-ray Crystallography", J. Wozniak, Joint Lab for Extreme Scale Computing, Nice, France. 2014.0609.
9. "The Future of Scientific Workflow", M. Wilde, DOE Facility Data Management Workshop, Oakland, CA, 2014.0616
10. Swift and web workflow portal presentation to DOE Advanced Power Grid Modeling and Simulation Workshop, Virginia, 2014.0617
11. MCS student lighting talk, 2014.0625
12. Workflow and Swift Tutorials, Argonne Training Program in Extreme Scale Computing, (ATPESC), J. Wozniak, M. Wilde, 2014.0814
13. UChicago Research Computing Center (RCC) outreach event "Mind Bytes" - Swift poster. 2014.1001
14. Presentation to Merck researchers, UChicago's Chicago Innovation Exchange, M Wilde, 2014.1020
15. UChicago Mind Bytes 2014.1028
16. "Dataflow for many-task computing: past, present and future". SC14 MTAGS Workshop, Keynote talk, M. Wilde, 2014.1116
17. Meetings with principal HPC users from SC14 Industrial HPC-Impact panel – GE Global Research, AFRL, Intelligent Light, Northrup, Westinghouse, BP, Total, PayPal. 2014.1117-1121
18. Parallel Computing in Molecular Engineering and Materials Science: Conquering the complexity of high performance computer modeling and integrating it into the scientific knowledge discovery process. M. Wilde, 2014 UChicago Discovery Cloud Lecture Series, 2014.1217
19. Swift project presentation to Corning materials science research staff, UChicago Computation Institute, 2015.0211
20. Swift poster at SI2 PI meeting poster session. M. Wilde. 2015.0217
21. Swift presentation to NERSC Workflow Day Meeting, M. Wilde, 2015.0220
22. Swift presentation to Cyberinfrastructure and Geospatial Information Laboratory, UIUC (Dr. Shaowen Wang, director), M. Wilde, 2015.0317

23. Swift workflow talk to US Intelligence Community multi-agency briefing workshop, M. Wilde, 2015.0325
24. Swift talk and poster presentation, Globus World 2015, Argonne – talk and poster session, 2015.0414
25. Swift presentation to NERSC workflow technology assessment group, M. Wilde, 2015.0429
26. Swift poster at APS User Meeting Poster Session, M. Wilde, 2015.0512
27. Swift presentation to NCSA Brownbag Lunch (over 50 attendees), D. Katz, M. Wilde 2015.0516
28. Swift talk for Argonne summer student program, 2015.0612
29. Swift intro talks to IIT/UChicago REU program "BigDataX", J. Wozniak, M. Wilde, 2015.0618
30. Swift Parallel Scripting: Novel Features and Applications. J. Wozniak, Joint Lab for Extreme Scale Computing, 3rd Workshop, Barcelona, 2015.0629
31. Optimizing data staging based on autotuning, coordination and locality exploitation on large scale supercomputers. F. Isaila, Joint Lab for Extreme Scale Computing, 3rd Workshop, Barcelona, 2015.0629
32. Workflow and Swift Tutorials, Argonne Training Program in Extreme Scale Computing, (ATPESC), J. Wozniak, M. Wilde, 2015.0811
33. Keynote address and hands-on tutorial on Swift and many-task parallel scripting for ERAD-RJ (Regional School in High Performance Computing) at LNCC, Petropolis, Brazil, 2015.0825http://eradrj2015.lncc.br/
34. Keynote address for CARLA 2015 (Latin America High Performance Computing Conference) LNCC, Petropolis, Brazil, 2015.0827

9 Publications (April 2014 – October 2015)

1. Timothy G. Armstrong, Justin M. Wozniak, Michael Wilde, and Ian T. Foster (2015). Swift: Extreme-scale, implicitly parallel scripting. *Programming Models for Parallel Computing* MIT Press.
2. Jonathan Ozik, Michael Wilde, Nicholson Collier, Charles M. Macal (2014). *Adaptive Simulation with Repast Simphony and Swift*. Proc. 2nd Workshop on Parallel and Distributed Agent-Based Simulations at Euro-Par 2014. Porto, Portugal.
3. Justin M. Wozniak, Kyle Chard, Ben Blaiszik, Ray Osborn, Michael Wilde, and Ian Foster (2015). *Big data remote access interfaces for light source science*. Proc. Big Data Computing 2015. Limassol, Cyprus.
4. Justin M. Wozniak, Hemant Sharma, Timothy G. Armstrong, Michael Wilde, Jonathan D. Almer, and Ian Foster (2014). *Big data staging with MPI-IO for interactive X-ray science*. Proc. Intl symposium on big data computing. London, UK.
5. Justin M. Wozniak, Timothy G. Armstrong, Daniel S. Katz, Michael Wilde, and Ian T. Foster. (2014). *Case studies in dataflow composition of scalable high performance applications*. Proc. Extreme-scale Programming Tools at SC14. New Orleans, LA USA.
6. Timothy G. Armstrong, Justin M. Wozniak, Michael Wilde, and Ian T. Foster (2014). *Compiler techniques for massively scalable implicit task parallelism*. Proc. SC 2014. New Orleans, LA

USA.
7. Scott J. Krieder, Justin M. Wozniak, Timothy G. Armstrong, Michael Wilde, Daniel S. Katz, Benjamin Grimmer, Ian T. Foster, and Ioan Raicu (2014). *Design and evaluation of the GeMTC framework for GPU-enabled many task computing*. Proc. HPDC 2014. Vancouver, Canada.
8. Ketan Maheshwari, Justin M. Wozniak, Hao Yang, Daniel S. Katz, Matei Ripeanu, Victor Zavala, and Michael Wilde (2014). *Evaluating storage systems for scientific data in the cloud*. Proc. 5th Workshop on Scientific Cloud Computing (ScienceCloud) at HPDC 2014. Vancouver, Canada.
9. Francisco Rodrigo Duro, Javier Garcia Blas, Florin Isaila, Jesus Carretero, Justin M. Wozniak, and Robert Ross (2014). *Exploiting data locality in Swift/T workflows using Hercules*. Proc. NESUS Workshop. Porto, Portugal.
10. Justin M. Wozniak, Timothy G. Armstrong, Ketan C. Maheshwari, Daniel S. Katz, Michael Wilde, and Ian T. Foster (2015). *Interlanguage parallel scripting for distributed-memory scientific computing*. Proc. WORKS at SC 2015. Austin, Texas USA.
11. Justin M. Wozniak, Timothy G. Armstrong, Ketan C. Maheshwari, Daniel S. Katz, Michael Wilde, and Ian T. Foster (2015). *Interlanguage parallel scripting for distributed-memory scientific computing*. Proceedings of Workflows in Support of Large-Scale Science (WORKS 2015). Austin, Texas USA.
12. Justin M. Wozniak, Michael Wilde, and Ian T. Foster (2014). *Language features for scalable distributed-memory dataflow computing*. Proc. Data-Flow Execution Models for Extreme-Scale Computing at PACT 2014. Edmonton, Alberta, Canada.
13. Matthieu Dorier, Matthieu Dreher, Tom Peterka, Justin M. Wozniak, Gabriel Antoniu, and Bruno Raffin (2015). *Lessons learned from building in situ coupling frameworks*. Proc. ISAV at SC 2015. Austin, Texas USA.
14. James C. Phillips, John E. Stone, Kirby L. Vandivort, Timothy G. Armstrong, Justin M. Wozniak, Michael Wilde, and Klaus Schulten (2014). *Petascale Tcl with NAMD, VMD, and Swift/T*. Proc. High Performance Technical Computing in Dynamic Languages at SC 2014. New Orleans, LA USA.
15. Ketan Maheshwari, Justin Wozniak, Timothy Armstrong, Daniel S. Katz, T. Andrew Binkowski, Xiaoliang Zhong, Olle Heinonen, Dmitry Karpeyev, and Michael Wilde (2015). *Porting ordinary applications to Blue Gene/Q supercomputers*. Proc. eScience 2015. Munich, Germany.
16. Justin M. Wozniak, Timothy G. Armstrong, Michael Wilde, and Ian Foster (2015). *Swift/T: Dataflow composition of Tcl scripts for petascale computing*. Proc. Annual Tcl/Tk Conference 2015. Manassas, Virginia.
17. Justin M. Wozniak, Timothy G. Armstrong, Ketan C. Maheshwari, Daniel S. Katz, Michael Wilde, and Ian T. Foster (2015). *Toward interlanguage parallel scripting for distributed-memory scientific computing*. IEEE CLUSTER 2015. Chicago, IL USA.
18. Zhao Zhang. *Enabling Efficient Parallel Scripting on Large-scale Computers*. (2014). Ph.D Dissertation. The University of Chicago.
19. Timothy Armstrong. *Implicitly parallel scripting as a practical and massively scalable programming model for high-performance computing*. (2015). Ph. D. Dissertation. The University of Chicago.

10 Impacts

Impact on the development of the principal discipline(s) of the project

The Swift parallel scripting system boosts the productivity of data- and computation-intensive biomedical research efforts by making large-scale parallel execution of application programs easier and faster. Swift makes parallel systems - from multicore workstations to grids, clouds and supercomputers - easier to fully and efficiently use, thus boosting research productivity. It addresses the conflict between the growing parallelism of future computing systems and the complexity of usefully applying that parallelism. And it enables research groups to more effectively validate, share, and replicate large-scale computing results and methods.

The Swift Project explores the important computer science question of how best to express the high-level integration programming of large-scale applications that have some degree of independently parallel tasks, and how to enable productive script-style programming of large-scale computing resources to enable their use by a broader scientific community with less effort and distraction form the scientific mission.

It is breaking ground in an approach that treats grids and petascale clusters as if they are complex chips, and takes a code-generation approach to automating the mapping of scientific workflow, loosely coupled by files instead of tightly coupled by messages, to these large-scale systems. By treating entire scientific applications as if they were functions, Swift leverages the simplicity and uniformity of the functional model to solve complex data flow specification problems.

Impact on other disciplines

Swift is making progress towards improving the ability of scientists and engineers to use large scale grids and petascale clusters for scientific programming. Swift improves research productivity in every data-intensive and/or compute-intensive scientific discipline by (1) processing many application runs in parallel, on large datasets, as specified by high-level scripts, without explicit parallel programming; (2) automating parallel, distributed data management; (3) running application scripts - without change - across a broad range of computing scales and platform types; and (4) recording provenance data for validation, sharing, discovery, and replication of results.

Impact on the development of human resources

Swift makes large-scale computing resources more accessible to students and researchers, and lowers the barrier to enabling students to do large-scale data analysis in the sciences. Students have been effectively using Swift for both CS research and to execute their scientific workflows. The project makes significant research and education opportunities available to a large number of students (both undergraduate and graduate) and post-doctoral researchers.

Impact on physical resources that form infrastructure

Swift provides an easier-to-use ``on-ramp'' for cluster, cloud, and grid resources including those of the XSEDE facility and the Open Science Grid, and campus research computing centers.

Impact on information resources that form infrastructure

Swift’s provenance tracking capabilities make it possible to track the derivation history of computationally-produced components of information resources, and make them more reproducible and re-usable by the resource users.

Impact on technology transfer

PI M. Wilde and developer and researcher P. M. Hategan are involved in a startup “Parallel.Works” (http://parallelworks.com), launched with initial seed funding from the University of Chicago's Innovation Fund, to make the Swift parallel scripting language available to commercial users through an enhanced Galaxy portal, as a SaaS offering.

Impact on society beyond science and technology

Swift can enable scientists, engineers, and product/service developers in industry to solve large-scale problems in energy and environmental modeling, and is being adopted in applications in drug design, health care, energy, environment, business and economics.