Argonne NATIONAL LABORATORY

ANL/MCS-TM-361

Highlights of X-Stack ExM Deliverable Swift/T

Mathematics and Computer Science Division

About Argonne National Laboratory
Argonne is a U.S. Department of Energy laboratory managed by UChicago Argonne, LLC under contract DE-AC02-06CH11357. The Laboratory's main facility is outside Chicago, at 9700 South Cass Avenue, Argonne, Illinois 60439. For information about Argonne and its pioneering science and technology programs, see www.anl.gov.

ANL/MCS-TM-361

Highlights of X-Stack ExM Deliverable Swift/T

Prepared by

J. M. Wozniak

Mathematics and Computer Science Division, Argonne National Laboratory

March 31, 2016

Highlights of X-Stack ExM Deliverable Swift/T

Justin M. Wozniak — February 5, 2016
wozniak@mcs.anl.gov

Swift/T is a key success from the **ExM: System support for extreme-scale, many-task applications**[1] X-Stack project, which proposed to use concurrent dataflow as an innovative programming model to exploit extreme parallelism in exascale computers. The Swift/T component of the project reimplemented the Swift language from scratch to allow applications that compose scientific modules together to be build and run on available petascale computers (Blue Gene, Cray). Swift/T does this via a new compiler and runtime that generates and executes the application as an MPI program.

Approach: We assume that mission-critical emerging exascale applications will be composed as scalable applications using **existing software components**, connected by data dependencies. Developers wrap native code fragments using a higher-level language, then build composite applications to form a computational experiment. This exemplifies hierarchical concurrency: lower-level messaging libraries are used for fine-grained parallelism; high-level control is used for inter-task coordination. These patterns are **best expressed with dataflow**, but static DAGs (i.e., other workflow languages) limit the applications that can be built; they do not provide the expressiveness of Swift, such as conditional execution, iteration, and recursive functions.

Technique: We reimplemented Swift with separable compiler and runtime components. The compiler is a source-to-source translator based on ANTLR that generates textual code for the new runtime, Turbine (hence /T). The runtime is based around ADLB [1], a previously developed MPI-based task distributor extended by ExM. The compiler contains novel optimizations for distributed-memory dataflow processing, and the runtime system evaluates the optimized code at unprecedented rates (1.5 B tasks/s on 512K cores of a Cray).

[1]PIs: Michael Wilde, Daniel S. Katz, Matei Ripeanu, Rusty Lusk, and Ian Foster.

Innovations: The Swift/T project made several technical **computer science contributions**. It demonstrated that high-level, functional dataflow programming can be deployed on extreme-scale computers through the application of ExM/X-Stack -developed techniques and technologies. We presented these to the dataflow community [2], [3], and they were the subject of an SC paper [4], a book chapter [5], and a Ph.D. dissertation at U. Chicago. It made use of a novel distributed-memory dataflow processing architecture [6], [7], [8].

The programming model accommodates **multiple hybrid levels of parallelism**; for example, it can drive subtasks that are themselves parallel MPI components [9] or it can be called as a subordinate part of a larger MPI job [10]. It also can be used to access heterogeneous systems. For example, it can distribute work to GPUs on Blue Waters at very high rates [11], [12]. It integrates well with (embedded) **high-level language interpreters** [10], [13], [14], [15], enabling scientific developers to mix scripted and compiled code, a prevalent practice in the scientific community with the growth of Python, R, and Julia. It is also important that the model and tools can debug and perform performance analysis on high-level programs [16], [17] with graphical tools such as Jumpshot.

Swift/T can elegantly solve problems intractable in prior, more cumbersome, less scalable workflow languages or less extensible architectures. Its mix of automated task placement with customizable user hints can be used to target node-local storage for data-intensive applications [18], [19], [20]. These data-aware features enable Swift/T to compactly express **MapReduce** and its generalizations (e.g., iterative MapReduce), without the typical barrier between phases [21]. It enables users to drop in previously developed metaheuristics (in Python or R) to implement complex ensemble algorithms [22]. Such algorithms benefit from the ability to use locality to target program state (not just bulk data).

Applications: ExM reached out to many user groups and gained promising trial usage, feedback, and endorsement from a diverse range of users. The **ExMatEx co-design center** evaluated Swift/T along with Charm++ [23], [24], [25]. We collaboratively studied the use of Swift/T to implement a multiscale materials model, and demonstrated how the small, sequential, C-based coarse-grained model (300 lines) could be translated into a Swift/T script; only the Swift/T script could call the fine-grained molecular model concurrently.

Swift/T was used by multiple **materials science teams** at the Advanced Photon Source at ANL and at the Cornell CHESS synchrotron [21], [26], [27], [28], [29], [30], [31]. In this role, Swift was used for bulk cluster data analysis, high-performance data analysis on the Blue Gene/Q (supported by MPI-IO integration), and complex crystal structure fitting algorithms powered by evolutionary algorithms. We used high-performance transfer techniques to load data onto the compute nodes before running data analysis tasks. Based on our initial investigations into managing streaming data from Swift/T [32], we believe that workflow languages like Swift will be a key factor in the streaming and steering of experimental and simulation data streams.

Swift/T can be used to tie together **in situ workflows**, integrating well with tools like Decaf [33] (SDAV). This is enabled by the ability to construct multiple multi-node tasks from Swift, start them, allow them to yield but stay resident in memory, send tasks to those task locations to perform in situ analysis, and then resume computation.

It is also a promising tool for constructing complex ensembles, such as **replica exchange molecular dynamics** [10] or **genetic algorithms** to perform parameter fitting in epidemics [22]. Such complex workflows are constructed by managing many high-performance simulations from Swift, with control logic implemented in a scripting language such as Python, R, or Tcl. We are currently developing an architecture where third-party machine learning, optimization, and metaheuristics from the **Python and R** communities can be easily plugged in to control Swift/T workflows.

Swift/T was also used to evaluate **power grid planning** scenarios on the Blue Gene at large scale [7], a naturally parallel problem that integrated well with previous techniques for this integer linear programming-based application. We made initial efforts to express branch-and-bound algorithms in Swift/T and developed features in support of this area; further collaboration in this area could be fruitful. We have recently made initial progress running an engine optimization ensemble [34].

Related ExM products: Swift/T was a partner project to other ExM efforts, including the prototype DAG runtime AMFORA (the integration of AME and AMFS) [35], [36], [37] and workflow-aware storage efforts [38], [39].

Computational experiments: Ensemble-based computations will be a key part of scientific computing at exascale. These naturally combine multiple modes and levels of programming complexity. The scripts that drive such computational experiments must be developed, maintained, and retained with the same rigor as the rest of the scientific software. Thus, high-quality, efficient workflow languages capable of utilizing the largest machines must investigated.

Emerging exascale roadmaps now emphasize the **explosive growth in on-node concurrency**, featuring many integrated cores or accelerators, instead of continued growth in node counts. Leading-edge parallel programming design must therefore shift from automating and enhancing multi-node concerns (e.g., messaging) to managing the opportunities and complexities in recently revealed massively concurrent NUMA computers. Wozniak has proposed work on multiple aspects of this challenge in his Early Career proposal, which is relevant outside the Swift ecosystem as well.

Wozniak and Wilde have fostered and developed connections with other exascale research projects, such as Argo. Swift has a naturally hierarchical programming model and rich available runtime information. These enable **power-related optimizations** through critical path analysis, task type segregation, load balancing, and other techniques. Swift also provides a **naturally fault-tolerant model** at a coarse granularity, as the runtime could automatically restart tasks that fail.

Further information: All Swift documentation and papers may be found at the following locations:

- **Swift home page**
  `http://swift-lang.org`
- **Wozniak home page**
  `http://www.mcs.anl.gov/~wozniak`

References

[1] E. L. Lusk, S. C. Pieper, and R. M. Butler, "More scalability, less pain: A simple programming model and its implementation for extreme computing," *SciDAC Review*, vol. 17, 2010.

[2] M. Wilde, J. M. Wozniak, T. G. Armstrong, D. S. Katz, and I. T. Foster, "Productive composition of extreme-scale applications using implicitly parallel dataflow," in *DOE Workshop on Software Productivity for eXtreme scale Science (SWP4XS)*, 2014.

[3] J. M. Wozniak, M. Wilde, and I. T. Foster, "Language features for scalable distributed-memory dataflow computing," in *Proc. Data-Flow Execution Models for Extreme-Scale Computing at PACT*, 2014.

[4] T. G. Armstrong, J. M. Wozniak, M. Wilde, and I. T. Foster, "Compiler techniques for massively scalable implicit task parallelism," in *Proc. SC*, 2014.

[5] ——, *Programming Models for Parallel Computing*, 2015, ch. Swift: Extreme-scale, implicitly parallel scripting, ed. P. Balaji.

[6] J. M. Wozniak, T. G. Armstrong, M. Wilde, K. Maheshwari, D. S. Katz, M. Ripeanu, E. L. Lusk, and I. T. Foster, "Turbine: A distributed-memory dataflow engine for extreme-scale many-task applications," in *Proc. Workshop on Scalable Workflow Enactment Engines and Technologies*, 2012.

[7] J. M. Wozniak, T. G. Armstrong, K. Maheshwari, E. L. Lusk, D. S. Katz, M. Wilde, and I. T. Foster, "Turbine: A distributed-memory dataflow engine for high performance many-task applications," *Fundamenta Informaticae*, vol. 28, no. 3, 2013.

[8] J. M. Wozniak, T. G. Armstrong, M. Wilde, D. S. Katz, E. Lusk, and I. T. Foster, "Swift/T: Scalable data flow programming for distributed-memory task-parallel applications," in *Proc. CCGrid*, 2013.

[9] J. M. Wozniak, T. Peterka, T. G. Armstrong, J. Dinan, E. L. Lusk, M. Wilde, and I. T. Foster, "Dataflow coordination of data-parallel tasks via MPI 3.0," in *Proc. Recent Advances in Message Passing Interface (EuroMPI)*, 2013.

[10] J. C. Phillips, J. E. Stone, K. L. Vandivort, T. G. Armstrong, J. M. Wozniak, M. Wilde, and K. Schulten, "Petascale Tcl with NAMD, VMD, and Swift/T," in *Proc. High Performance Technical Computing in Dynamic Languages at SC*, 2014.

[11] S. J. Krieder, J. M. Wozniak, T. G. Armstrong, M. Wilde, D. S. Katz, B. Grimmer, I. T. Foster, and I. Raicu, "Design and evaluation of the GeMTC framework for GPU-enabled many task computing," in *Proc. HPDC*, 2014.

[12] S. Krieder, I. Raicu, J. M. Wozniak, and M. Wilde, "Implicitly-parallel functional dataflow for productive cloud programming on chameleon," in *Proc. NSFCloud Workshop on Experimental Support for Cloud Computing*, 2014.

[13] J. M. Wozniak, T. G. Armstrong, D. S. Katz, M. Wilde, and I. T. Foster, "Toward computational experiment management via multi-language applications," in *DOE Workshop on Software Productivity for eXtreme scale Science (SWP4XS)*, 2014.

[14] J. M. Wozniak, T. G. Armstrong, M. Wilde, and I. Foster, "Swift/T: Dataflow composition of Tcl scripts for petascale computing," in *Proc. Annual Tcl/Tk Conference*, 2015.

[15] J. M. Wozniak, T. G. Armstrong, K. C. Maheshwari, D. S. Katz, M. Wilde, and I. T. Foster, "Interlanguage parallel scripting for distributed-memory scientific computing," in *Proc. WORKS at SC*, 2015.

[16] J. M. Wozniak, A. Chan, T. G. Armstrong, M. Wilde, E. Lusk, and I. T. Foster, "A model for tracing and debugging large-scale task-parallel programs with MPE," in *Proc. Workshop on Leveraging Abstractions and Semantics in High-performance Computing (LASH-C) at PPoPP*, 2013.

[17] J. M. Wozniak, T. G. Armstrong, D. S. Katz, M. Wilde, and I. T. Foster, "Case studies in dataflow composition of scalable high performance applications," in *Proc. Extreme-scale Programming Tools at SC*, 2014.

[18] F. R. Duro, J. G. Blas, F. Isaila, J. Carretero, J. M. Wozniak, and R. Ross, "Exploiting data locality in Swift/T workflows using Hercules," in *Proc. NESUS Workshop*, 2014.

[19] F. R. Duro, J. G. B. F. Isaila, and J. Carretero, "Experimental evaluation of a flexible I/O architecture for accelerating workflow engines in cloud environments," in *Proc. DISCS at SC*, 2015.

[20] F. R. Duro, J. G. Blas, F. Isaila, J. M. Wozniak, J. Carretero, and R. Ross, "Flexible data-aware scheduling for workflows over an in-memory object store," in *Proc. CCGrid*, 2016.

[21] J. M. Wozniak, H. Sharma, T. G. Armstrong, M. Wilde, J. D. Almer, and I. Foster, "Big data staging with MPI-IO for interactive X-ray science," in *Proc. Big Data Computing*, 2014.

[22] J. Ozik, N. Collier, and J. M. Wozniak, "Many resident task computing in support of dynamic ensemble computations," in *Proc. MTAGS at SC*, 2015.

[23] J. M. Wozniak, "Rapid development of highly concurrent multi-scale simulators with Swift," ExMatEx all-hands meeting 2013.

[24] T. C. Germann, "Exploiting asynchrony for materials in extreme environments: Programming models and runtime requirements," in *Proc. Salishan Conference on High-Speed Computing*, 2014.

[25] J. M. Wozniak, "Swift: Parallel scripting for simulation ensembles," ExMatEx all-hands meeting 2015.

[26] ——, "Swift+Chirp for synchrotron beamline data analysis," 2013, at Cooperative Computing Laboratory Workshop.

[27] L. Wolf, "Boosting beamline performance," 2014, https://www.alcf.anl.gov/articles/boosting-beamline-performance.

[28] K. Dedrick, "Argonne group sets record for largest x-ray dataset ever at CHESS," 2015, CHESS News.

[29] J. M. Wozniak, "Integrating big data tools for x-ray science," 2015, talk at Joint Laboratory for Extreme-Scale Computing Workshop.

[30] I. Foster, T. Bicer, R. Kettimuthu, M. Wilde, J. M. Wozniak, F. de Carlo, B. Blaiszik, K. Chard, F. de Carlo, and R. Osborn, "Accelerating the experimental feedback loop: Data streams at the APS," in *Proc. STREAM*, 2015.

[31] J. M. Wozniak, "Big data tools for light source science," 2015, talk at Cornell High Energy Synchrotron Source.

[32] J. M. Wozniak, K. Chard, B. Blaiszik, M. Wilde, and

I. Foster, "Streaming, storing, and sharing big data for light source science," in *Proc. STREAM*, 2015.

[33] M. Dorier, M. Dreher, T. Peterka, J. M. Wozniak, G. Antoniu, and B. Raffin, "Lessons learned from building in situ coupling frameworks," in *Proc. In Situ Infrastructures for Enabling Extreme-scale Analysis and Visualization at SC*, 2015.

[34] G. Cunningham, "VERIFI code optimization yields three-fold increase in engine simulation speed," 2015, http://www.anl.gov/articles/verifi-code-optimization-yields-three-fold-increase-engine-simulation-speed.

[35] Z. Zhang, D. S. Katz, J. M. Wozniak, A. Espinosa, and I. Foster, "Design and analysis of data management in scalable parallel scripting," in *Proc. SC*, 2012.

[36] Z. Zhang, D. S. Katz, T. G. Armstrong, J. M. Wozniak, and I. T. Foster, "Parallelizing the execution of sequential scripts," in *Proc. SC*, 2013.

[37] Z. Zhang, D. S. Katz, J. M. Wozniak, M. Wilde, and I. T. Foster, "MTC Envelope: Defining the capability of large scale computers in the context of parallel scripting applications," in *Proc. Symposium on High Performance Distributed Computing (HPDC)*, 2013.

[38] E. Vairavanathan, S. Al-Kiswany, L. Costa, Z. Zhang, D. Katz, M. Wilde, and M. Ripeanu, "A Workflow-Aware Storage System: An opportunity study," in *Proc. CCGrid*, 2012.

[39] K. Maheshwari, J. M. Wozniak, H. Yang, D. S. Katz, M. Ripeanu, V. Zavala, and M. Wilde, "Evaluating storage systems for scientific data in the cloud," in *Proc. ScienceCloud*, 2014, (Best paper awardee).

Mathematics and Computer Science Division
Argonne National Laboratory
9700 South Cass Avenue, Bldg. 240
Argonne, IL 60439

www.anl.gov