Application Skeleton: Generating Synthetic Applications for Infrastructure Research

Zhao Zhang[1], Daniel S. Katz[2], Andre Merzky[3], Matteo Turilli[4], Shantenu Jha[5], and Yadu Nand[6]

[1]AMPLab and BIDS, University of California, Berkeley
[2]National Center for Supercomputing Applications, University of Illinois Urbana-Champaign
[3]RADICAL Laboratory, Rutgers University
[4]RADICAL Laboratory, Rutgers University
[5]RADICAL Laboratory, Rutgers University
[6]Computation Institute, University of Chicago

5 May 2016

Software Repository: https://github.com/applicationskeleton/Skeleton
Software Archive: http://dx.doi.org/10.5281/zenodo.13750

Summary

Application Skeleton is a simple and powerful tool to build simplified synthetic science and engineering applications (for example, modeling and simulation, data analysis) with runtime and I/O close to that of the real applications. It is intended for applied computer scientists who need to use science and engineering applications to verify the effectiveness of new systems designed to efficiently run such applications, so that they can bypass obstacles that they often encounter when accessing and building real science and engineering applications. Using the applications generated by Application Skeleton guarantees that the CS systems' effectiveness on synthetic applications will apply to the real applications.

Application Skeleton can generate bag-of-task, (iterative) map-reduce, and (iterative) multistage workflow applications. These applications are represented as a set of tasks, a set of input files, and a set of dependencies. These applications can be generally considered as many-task applications, and once created, can be run on single-core, single-node, multi-core, or multi-node (distributed or parallel) computers, depending on what workflow system is used to run them. The generated applications are compatible with workflow system such as Swift (Zhao et al. 2007, Wilde et al. (2009), Wilde et al. (2011)) and Pegasus (Ewa Deelman et al. 2004, E. Deelman et al. (2005)), as well as the ubiquitous UNIX shell. The application can also be created as a generic JSON object that can be used by other systems such as the AIMES (Turilli et al. 2015) middleware.

References

Deelman, E., G. Singh, M.-H. Su, J. Blythe, Y. Gil, C. Kesselman, G. Mehta, et al. 2005. "Pegasus: A Framework for Mapping Complex Scientific Workflows onto Distributed Systems." *Scientific Programming Journal* 13 (3): 219–37. doi:10.1155/2005/128026.

Deelman, Ewa, James Blythe, Yolanda Gil, Carl Kesselman, Gaurang Mehta, Sonal Patil, Mei-Hui Su, Karan Vahi, and Miron Livny. 2004. "Pegasus: Mapping Scientific Workflows onto the Grid." In *Grid Computing*,

edited by Marios D. Dikaiakos, 3165:131–40. Lect. Notes in Comp. Sci. Springer. doi:10.1007/978-3-540-28642-4_2.

Turilli, Matteo, Zhao Zhang, Andre Merzky, Michael Wilde, Jon Weissman, Daniel S Katz, Shantenu Jha, and others. 2015. "Integrating Abstractions to Enhance the Execution of Distributed Applications." *ArXiv Preprint ArXiv:1504.04720*.

Wilde, Michael, Ian Foster, Kamil Iskra, Pete Beckman, Zhao Zhang, Allan Espinosa, Mihael Hategan, Ben Clifford, and Ioan Raicu. 2009. "Parallel Scripting for Applications at the Petascale and Beyond." *Computer* 42. IEEE Computer Society: 50–60. doi:10.1109/MC.2009.365.

Wilde, Michael, Mihael Hategan, Justin M. Wozniak, Ben Clifford, Daniel S. Katz, and Ian Foster. 2011. "Swift: A Language for Distributed Parallel Scripting." *Par. Comp.*, September, 633–52. doi:10.1016/j.parco.2011.05.005.

Zhao, Yong, Mihael Hategan, Ben Clifford, Ian Foster, Gregor Von Laszewski, Veronika Nefedova, Ioan Raicu, Tiberiu Stef-Praun, and Michael Wilde. 2007. "Swift: Fast, Reliable, Loosely Coupled Parallel Computation." In *IEEE Congress on Services*, 199–206. IEEE. doi:10.1109/SERVICES.2007.63.