A First Look at the JX Workflow Language

Tim Shaffer
University of Notre Dame
Notre Dame, Indiana 46556
tshaffe1@nd.edu

Kyle M.D. Sweeney
University of Notre Dame
Notre Dame, Indiana 46556
ksweene3@nd.edu

Nathaniel Kremer-Herman
University of Notre Dame
Notre Dame, Indiana 46556
nkremerh@nd.edu

Douglas Thain
University of Notre Dame
Notre Dame, Indiana 46556
dthain@nd.edu

***Abstract*—Scientific workflows are typically expressed as a graph of logical tasks, each one representing a single program along with its input and output files. This poster introduces JX (JSON eXtended), a declarative language that can express complex workloads as an assembly of sub-graphs that can be partitioned in flexible ways. We present a case study of using JX to represent complex workflows for the Lifemapper biodiversity project. We evaluate partitioning approaches across several computing environments, including ND-Condor, IU-Jetstream, and SDSC-Comet, and show that a coarse partitioning results in faster turnaround times, reduced data transfer, and lower master utilization across all three systems.**

I. Introduction

Workflows are a widely-used abstraction for representing simulations, data analyses, and other scientific computations. A workflow is commonly represented as a directed acyclic graph (DAG) which provides a static description of a complex pipeline of interdependent steps called tasks. A workflow management system is used to parse this complex DAG to submit each task to an execution engine once that task's dependencies are met. *Workflow partitioning* is the process of splitting a workflow graph into sub-graphs, such that each sub-graph will become a discrete batch job in the target execution system. The most appropriate partitioning depends on many properties of the workflow graph, such as the size of data objects and the execution time of tasks, as well as the performance properties of the execution system.

II. JX: JSON eXtended

We developed JX (JSON eXtended) as a language for expressing workflows that allows for easy manipulations to the structure and partitioning of a workflow. JX extends a JSON [1] representation of the workflow by supporting a subset of Python expressions, allowing for a concise intermediate representation that expands to a normal JSON document. Templates in JX can expand to complicated nested workflow structures based on parameters, allowing flexible changes to a workflow's partitioning scheme.

The following workflow "template", when expanded, expresses an entire map-reduce type workflow. The expanded workflow includes a rule for each of the N input files, and a reduce step that takes all N intermediate files and produces the final output. This template exposes the structure of the workflow as a parameter. These parametric templates are the primary way JX allows for flexible changes in the organization and partitioning scheme of a scientific workflow.

```
{"rules": [{
  "inputs": ["split." + str(i)],
  "outputs": ["out." + str(i)],
  "command":
    "./process.sh split." + str(i),
} for i in range(N)]
+ [{
  "inputs": [
    "out." + str(i) for i in range(N)
  ],
  "outputs": ["result.dat"],
  "command": "./reduce.sh out.*",
}]}
```

III. Evaluation

We explored schemes for partitioning Lifemapper [2], a distributed biodiversity modeling application. As a high-throughput application, Lifemapper offers significant freedom in organizing computation beyond simply following data dependency relationships. We observed that the granularity at which we distribute pieces of the workflow has a significant impact on the overall behavior of the workflow.

We measured the behavior of Lifemapper under two different partitioning schemes and ran the application on the IU-Jetstream [3], ND-Condor [4], and SDSC-Comet [5] execution sites. We observed substantial differences in performance in terms of execution time, node-local storage, and data transfer between configurations when running on the same execution site. Across all three computing sites, there were similar trends in reduced data transfer and execution time with coarser workflow partitioning.

IV. Related Work

Galaxy [6]–[8] and Taverna [9], [10] operate on static workflow "templates" with the runtime generating concrete steps during execution. Pegasus [11]–[13] is a workflow management system focused on fine-grained management of communication and task scheduling that uses a static XML [14] workflow representation. The Common Workflow Language (CWL) [15] is a workflow specification standard for describing command line tools and workflows in a portable manner. Swift [16] is a custom dataflow language which represents a workflow as a tree of recursive function evaluations such that jobs are the leaves of the evaluation tree.

REFERENCES

[1] T. Bray, "The javascript object notation (json) data interchange format," Internet Requests for Comments, RFC Editor, RFC 7159, March 2014, http://www.rfc-editor.org/rfc/rfc7159.txt. [Online]. Available: http://www.rfc-editor.org/rfc/rfc7159.txt

[2] [Online]. Available: http://lifemapper.org

[3] C. A. Stewart, T. M. Cockerill, I. Foster, D. Hancock, N. Merchant, E. Skidmore, D. Stanzione, J. Taylor, S. Tuecke, G. Turner, M. Vaughn, and N. I. Gaffney, "Jetstream: A self-provisioned, scalable science and engineering cloud environment," in *Proceedings of the 2015 XSEDE Conference: Scientific Advancements Enabled by Enhanced Cyberinfrastructure*, ser. XSEDE '15. New York, NY, USA: ACM, 2015, pp. 29:1–29:8. [Online]. Available: http://doi.acm.org/10.1145/2792745.2792774

[4] M. J. Litzkow, M. Livny, and M. W. Mutka, "Condor-a hunter of idle workstations," in *[1988] Proceedings. The 8th International Conference on Distributed*, Jun 1988, pp. 104–111.

[5] J. Towns, T. Cockerill, M. Dahan, I. Foster, K. Gaither, A. Grimshaw, V. Hazlewood, S. Lathrop, D. Lifka, G. D. Peterson, R. Roskies, J. R. Scott, and N. Wilkins-Diehr, "Xsede: Accelerating scientific discovery," *Computing in Science Engineering*, vol. 16, no. 5, pp. 62–74, Sept 2014.

[6] B. Giardine, C. Riemer, R. C. Hardison, R. Burhans, L. Elnitski, P. Shah, Y. Zhang, D. Blankenberg, I. Albert, J. Taylor, W. C. Miller, W. J. Kent, and A. Nekrutenko, "Galaxy: a platform for interactive large-scale genome analysis," *Genome research*, vol. 15, no. 10, pp. 1451–1455, 2005.

[7] D. Blankenberg, G. V. Kuster, N. Coraor, G. Ananda, R. Lazarus, M. Mangan, A. Nekrutenko, and J. Taylor, "Galaxy: A web-based genome analysis tool for experimentalists," *Current protocols in molecular biology*, pp. 19–10, 2010.

[8] J. Goecks, A. Nekrutenko, J. Taylor, and T. G. Team, "Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences," *Genome Biol*, vol. 11, no. 8, p. R86, 2010.

[9] D. Turi, P. Missier, C. Goble, D. D. Roure, and T. Oinn, "Taverna workflows: Syntax and semantics," in *Third IEEE International Conference on e-Science and Grid Computing (e-Science 2007)*, Dec 2007, pp. 441–448.

[10] K. Wolstencroft, R. Haines, D. Fellows, A. Williams, D. Withers, S. Owen, S. Soiland-Reyes, I. Dunlop, A. Nenadic, P. Fisher, J. Bhagat, K. Belhajjame, F. Bacall, A. Hardisty, A. Nieva de la Hidalga, M. P. Balcazar Vargas, S. Sufi, and C. Goble, "The taverna workflow suite: designing and executing workflows of web services on the desktop, web or in the cloud," *Nucleic Acids Research*, vol. 41, no. W1, p. W557, 2013. [Online]. Available: + http://dx.doi.org/10.1093/nar/gkt328

[11] E. Deelman, J. Blythe, Y. Gil, C. Kesselman, G. Mehta, S. Patil, M.-H. Su, K. Vahi, and M. Livny, *Pegasus: Mapping Scientific Workflows onto the Grid*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 11–20. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-28642-4_2

[12] W. Chen and E. Deelman, "Integration of workflow partitioning and resource provisioning," in *The 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid 2012)*, 2012.

[13] ——, *Partitioning and Scheduling Workflows across Multiple Sites with Storage Constraints*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 11–20. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-31500-8_2

[14] "Extensible markup language (xml) 1.1 (second edition)," http://www.w3.org/TR/2006/REC-xml11-20060816/, W3C - World Wide Web Consortium, W3C Recommendation, September 2006. [Online]. Available: http://www.w3.org/TR/2006/REC-xml11-20060816/

[15] P. Amstutz, M. R. Crusoe, N. Tijanić, B. Chapman, J. Chilton, M. Heuer, A. Kartashov, D. Leehr, H. Ménager, M. Nedeljkovich, M. Scales, S. Soiland-Reyes, and L. Stojanovic, "Common Workflow Language, v1.0," 7 2016. [Online]. Available: https://figshare.com/articles/Common_Workflow_Language_draft_3/3115156

[16] M. Wilde, M. Hategan, J. M. Wozniak, B. Clifford, D. S. Katz, and I. Foster, "Swift: A language for distributed parallel scripting," *Parallel Computing*, vol. 37, no. 9, pp. 633 – 652, 2011, emerging Programming Paradigms for Large-Scale Scientific Computing. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167819111000524