Skip the line and balance your work with vQ

Gregory Zynda*, Mehmet Dalkilic†, and Matthew Vaughn‡

17th Annual Bioinformatics Open Source Conference (BOSC) 2016, Orlando, FL, USA

Repository: https://github.com/zyndagj/vQ

Processor core-counts are climbing in servers, computational accelerators, and even the phones in our pockets. The bioinformatics community takes advantage of this trend with threaded applications to utilize all available processor cores. While threading improves workflows at the workstation-level, they provide little benefit on high-performance supercomputers. Supercomputers, like Stampede at The Texas Advanced Computing Center and Comet at The San Diego Supercomputing Center, are not monolithic mainframes with immense resources that are shared between programs and users like on a traditional desktop. Instead, they are comprised of thousands of separate computers that can be reserved in groups to cooperatively work on computing tasks over a network. Sequence assembly is among the most arduous of bioinformatics tasks due to the substantial computational resources and large, complex datasets it requires. To reduce the time it takes to assemble a genome, developers have begun to develop multi-process workflows that align to using cluster architecture by submitting concurrent tasks as separate jobs to a centralized scheduler that manages resource usage across the system. Job schedulers usually implement a queue of some sort. This means that while concurrent job submissions are possible, the total run time for the task is limited by resource availability in that queue. And, resources are often in short supply. Most shared computing clusters, from department-level systems, up to national class systems such as those provided by the National Science Foundation, are oversubscribed with long queues. This renders any assumption that hundreds or thousands of job submissions can run to accomplish a given task totally impractical.

To improve the run times for such tasks, we developed the vQ, a virtual queue for multi-node jobs that will dynamically balance the execution of tasks issued with normal SLURM, SGE, TORQUE, and OpenLava (LSF) commands across a pre-allocated set of computing nodes on a shared resource. A distributed workflow system that previously relied on submitting and running thousands of jobs and/or sitting resident on a master node can now be structured as a single job which waits in a shared queue only once before commencing execution and does not unfairly consume shared resources on a login node. We show that this drop-in solution removes queue overhead and improves the experience of running complicated workflows like Trinity, SMRT, Falcon, Canu, Celera, and Cluster Flow. Most importantly, unlike Makeflow [3] or TACC's Launcher utilities which assume the tasks in a workflow can be described in a simple text file [2, 1], vQ requires no modifications to the original tools and works out-of-the-box with a minimal, user-level installation. Thus, vQ provides a potential solution to queue bottlenecks in current distributed bioinformatics tools.

References

[1] Victor Eijkhout. pylauncher: The python launcher. https://github.com/TACC/pylauncher, 2012.

[2] Lucas A Wilson and John M Fonner. Launcher: A shell-based framework for rapid development of parallel parametric studies. In *Proceedings of the 2014 Annual Conference on Extreme Science and Engineering Discovery Environment*, page 40. ACM, 2014.

[3] Li Yu, Christopher Moretti, Andrew Thrasher, Scott Emrich, Kenneth Judd, and Douglas Thain. Harnessing parallelism in multicore clusters with the all-pairs, wavefront, and makeflow abstractions. *Cluster Computing*, 13(3):243–256, 2010.

*Texas Advanced Computing Center, University of Texas at Austin. Email: gzynda@tacc.utexas.edu
†School of Informatics and Computing, Indiana University
‡Texas Advanced Computing Center, University of Texas at Austin