Optimizing COMAD for Growing Input Datasets

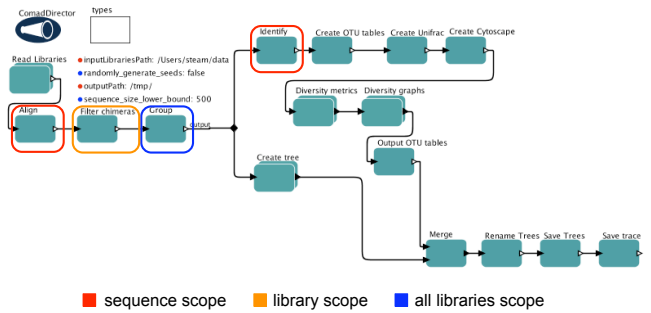
Sean Riddle[1], Amber Hartman, Timothy M. McPhillips[1], Shawn Bowers[2], David Welker[1], and Bertram Ludäscher[1,2]

[1]UC Davis Genome Center, [2]Department of Computer Science, UC Davis.

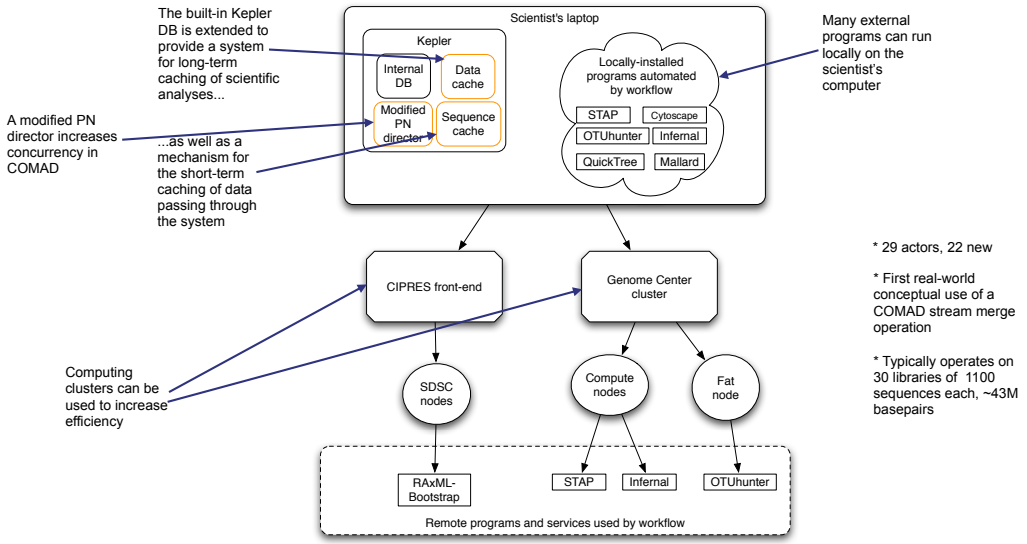
www.kepler-project.org

Workflow for Alignment, Taxonomy and Ecology of Ribosomal Sequences (WATERS)

The WATERS workflow is designed to characterize microbial populations. It produces phylogenetic trees, diversity metrics, Unifrac environment files, etc. as outputs in response to the inputs of one or more sequence libraries. The sequence libraries have various metadata commonalities, and diversity metrics are generated based on that metadata in addition to more inherent distinctions like sequence library.

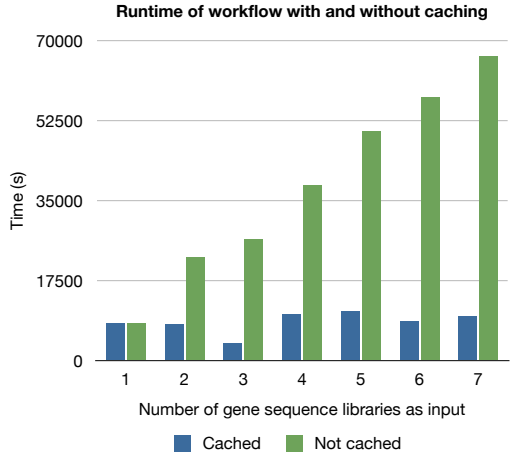


The workflow aligns all the sequences, removes chimerical sequences then groups them by OTU and selects representative sequences. These sequences are then identified, and outputs are generated based off of them. Several steps, including tree generation and sequence alignment, have multiple interchangeable implementations.



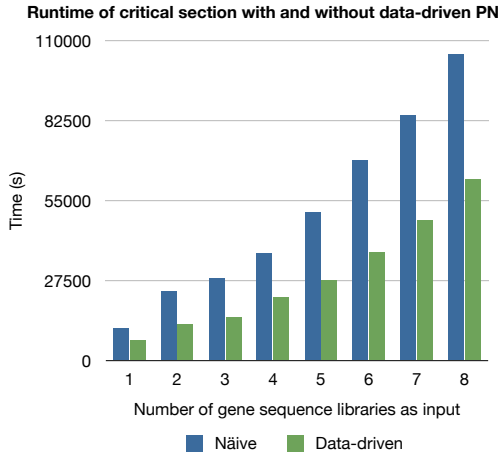
* 29 actors, 22 new
* First real-world conceptual use of a COMAD stream merge operation
* Typically operates on 30 libraries of 1100 sequences each, ~43M basepairs

Many of the steps in this workflow are computationally-intensive, and so it is beneficial to minimize the number of computations performed. The normal use of this workflow would be to run it on genetic library files A, and then, later in time, run it on genetic library files B, where B is a superset of A. This makes steps that run on a per-sequence (Align and Identify) and per-library (Filter chimeras) basis quite worthwhile to cache.



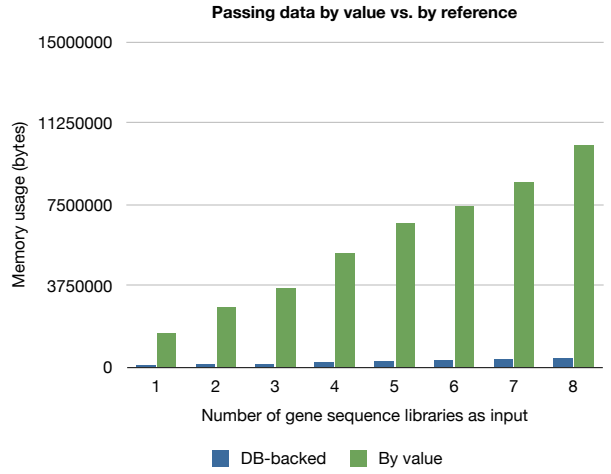
Running the workflow a second time on files B will perform the per-sequence and per-library operations on only those libraries newly introduced in B. The Group operation requires as input all the libraries, so it will have to be re-run on the entirety of B.

In Ptolemy's implementation of PN, queues are resized only when forced by a blocking write. This guarantees no write deadlocks and the queues use minimal memory. This approach has a drawback when used in a COMAD workflow with branching. If the default initial queue size of 1 is used, then only one token at a time will be sent from the actor directly before the branch point. As soon as one of the actors on a branch encounters a read scope match and begins executing, it will for a time stop processing tokens and will starve the other branch.



The starved branch cannot get more tokens until the currently working branch is ready to move ahead. This forces the branches to work one at a time unless they have the exact same read scope expression. Increasing the initial queue size sufficiently will cause them to work in parallel as intended. The modified data-driven director uses queues of infinite size.

Due to the pipelined nature of COMAD, a great deal of tokens are being processed at any one time somewhere in the workflow. If the tokens are of non-negligible size, this can result in high memory usage and even running out of memory entirely. To circumvent this problem, information regarding genetic sequences is stored in a database, and tokens only contain enough information to recover the full sequence data on demand.



The speed of database accesses is negligible, but memory usage of Kepler scales much better with larger datasets.

This work was supported by NSF awards IIS 0629846 (PPOD), IIS 0612326 (CHIP2), and OCI 0722079 (Kepler/CORE).