Expanding Tasks of Logical Workflows into Independent Workflows for Improved Scalability

Nicholas Hazekamp, Olivia Choudhury, Sandra Gesing, Scott Emrich, and Douglas Thain
Department of Computer Science and Engineering
University of Notre Dame
Notre Dame, Indiana 46556
Email{nhazekam@nd.edu: ochoudhu@nd.edu: sandra.gesing@nd.edu: semrich@nd.edu: dthain@nd.edu}

***Abstract*—Workflow management systems, such as Galaxy and Taverna, provide a portal through which data can be processed using a sequence of different tools. This sequence allows for the creation of a logical workflow that describes the process. However, when the data workload becomes large enough the time spent in each logical step increases making it difficult to run the workflow fast and efficiently. The proposed solutions is to use task level expansion. Task expansion aims to take each step of the logical workflow and expand it into a new self-contained workflow. These workflows would allow for greater scalability and concurrency by creating more tasks. The resulting workflows will be used indistinguishably from the original tool, but perform more quickly and efficiently. The concept was applied to the BWA tool in Galaxy and we were able to see a 7.36 times speedup in runtime on our 32 GB dataset.**

I. Expansion of Logical Workflows

Workflow management systems, such as Galaxy [1]–[3] and Taverna [4], allow scientists to easily use a variety tools together. A WMS creates a portal with an understandable UI, performs simple data management, and provides easier access to tools generated by others. These traits make the systems versatile, and using of tools from different sources to process data in a unique, reproducible manner. Ordering the tasks provides a general overview, which we refer to as a **logical** workflow (see top of Figure 1).

Workflows are used regularly in the processing of data, whether from the logical perspective or the more specific task perspective. Each process, however, is limited to the scalability of the underlying tool. Though WMS will allow multiple concurrent jobs to run, many tools are limited to one machine having only thread parallelism. Therefore, even if WMS allows multiple tasks to run concurrently, large data tasks are limited to a single resource. We propose the use of task expansion workflows, shown at the bottom of Figure 1. **Task expansion** takes each task and expands it using three general steps: partitioning of data, execution of smaller pieces concurrently, and concatenation of results. While logical workflows are useful for non-computing experts to see what tasks have been performed and later dependencies, the task expansion workflow describes how a task partitions data, what steps can run independent of each other, and how to combine results.

To properly use task expansion workflows in a WMS, there are several stipulations. The first is that there is no editing of input files directly within the database, as this would interfere with the host WMSs data management. The second is that any expanded tool must take the same input and produce the same output. This would allow the user to simply replace the base tool with the task expansion workflow tool without requiring additional changes. The last stipulation is that the expanded tool should be indistinguishable from its serial counterpart from the WMS users perspective; any previously available input parameters should be available, or an explanation for their absence clear.

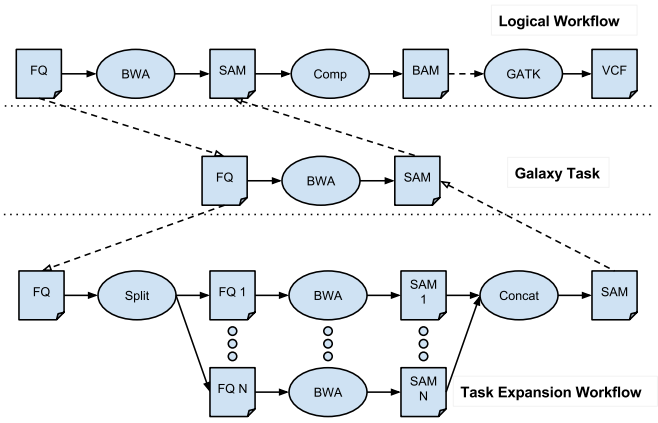

Fig. 1: WMS Workflow Decomposition. The top level is a visualization of a typical logical workflow. The next level has isolated a single task that we will use for expansion. The lowest layer illustrates the three parts of the expanded workflow, including partitioning, execution, and concatenation.

The process of task expansion will be different between applications. In Figure 1, the fastq input is split based on the number of reads. This is possible because BWA performs the alignment on reads with limited or no other knowledge of the entire collection. We note that the partitioning step may be difficult for applications with more strict analysis structure; however, these applications can often still be split more specifically. If partitioning is not possible, task expansion could be used to vary parameters or partition files, such as a reference, to limit the search space.

II. Method

For our implementation of task expansion, we started off with the Burrows-Wheeler Alignment [5] tool within Galaxy.

BWA is well established within Galaxy that we have also used extensively outside of Galaxy. The dataset that was used contained 50 different partial red oak genomes, resulting in an input size of over 32 GB. Galaxy's BWA tool was run initially for a baseline. To handle the large dataset, the data was loaded as a data library. The sequential method produced the results in 4 hours and 4 minutes.

Next, we looked at how to expand the BWA tool. We utilized Makeflow [6], as it allows for the creation of static workflows given a known partitioning algorithm. Once the static workflow is created, Makeflow can utilize a variety of different execution engines. Work Queue [7] was selected as the execution engine, because it facilitates the use of resources from a variety of environments such as Condor and SGE. The combination of Makeflow and Work Queue allows for the utilization of different resources in a dynamic manner, while the Galaxy instance is tied to a single execution engine. Work Queue allows for the addition of workers depending on the need and number of partitions.

To expand the Galaxy task, we must first partition the data. For BWA, we partitioned the data based on a static number of reads. According to our work [8], the static partitions gave insignificantly slower performance in the alignment, but were much faster in the partitioning step. After partitioning, we let Makeflow and Work Queue run the tasks. The number of tasks run within Makeflow depends on how finely we partition the dataset. After all of the tasks had completed, we concatenated the results into a single file, which produced a SAM file indistinguishable from the original BWA tool.

III. Results

After expanding the BWA task into a self-contained workflow, we were able to see a significant amount of speedup. Having the original tool as the base line, we were able to see a speedup of 7.36 times performance. To illustrate how this implementation scaled, we ran the tool with 2 workers up to 100 workers. Figure 2 shows the speedup relative to our base using different amounts of workers. The amount of improvement per worker falls off after 10 workers, giving a lower efficiency as more workers are added. The lower return from added workers could be due to a bottleneck in data transfer. If execution time is close to transfer time the master is only able to supply a finite amount of workers with tasks, leading to only slight improvement with more workers.

IV. Conclusion

Using the concept of task expansion, we were able to achieve 7.36 times speedup from the original BWA tool in Galaxy. In the future, we will continue apply task expansion for other bioinformatics tools, such as GATK Haplotype Caller and RNA Rocket. While Galaxy provides a full and easily accessible library of tools, many of these tools could benefit from workflow expansion at the tool level. We plan to continue looking into ways to accelerate and streamline WMSs and their tools by using techniques that can be applied to more general distributed computing problems.

Fig. 2: Speedup of BWA relative to the BWA Tool originally in the Galaxy Toolshed. The results were calculated using $S = T_S/T_P$, where the sequential time is the runtime for the original tool.

References

[1] 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.

[2] 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.

[3] 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.

[4] 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, pp. W557–W561, 2013. [Online]. Available: http://nar.oxfordjournals.org/content/41/W1/W557.abstract

[5] H. Li and R. Durbin, "Fast and accurate short read alignment with burrows–wheeler transform," *Bioinformatics*, vol. 25, no. 14, pp. 1754–1760, 2009.

[6] M. Albrecht, P. Donnelly, P. Bui, and D. Thain, "Makeflow: A Portable Abstraction for Data Intensive Computing on Clusters, Clouds, and Grids," in *Workshop on Scalable Workflow Enactment Engines and Technologies (SWEET) at ACM SIGMOD*, 2012.

[7] P. Bui, D. Rajan, B. Abdul-Wahid, J. Izaguirre, and D. Thain, "Work Queue + Python: A Framework For Scalable Scientific Ensemble Applications," in *Workshop on Python for High Performance and Scientific Computing (PyHPC) at the ACM/IEEE International Conference for High Performance Computing, Networking, Storage, and Analysis (Supercomputing)* , 2011.

[8] O. Choudhury, N. Hazekamp, D. Thain, and S. Emrich, "Accelerating comparative genomics workflows in a distributed environment with optimized data partitioning," submitted to C4Bio Workshop at CCGrid.