Bioinformatics, 35(20), 2019, 4173–4175
doi: 10.1093/bioinformatics/btz169
Advance Access Publication Date: 11 March 2019
Applications Note

Gene expression

Holistic optimization of an RNA-seq workflow for multi-threaded environments

Ling-Hong Hung[1], Wes Lloyd[1], Radhika Agumbe Sridhar[1], Saranya Devi Athmalingam Ravishankar[1], Yuguang Xiong[2], Eric Sobie[2] and Ka Yee Yeung [1,*]

[1]School of Engineering and Technology, Tacoma, WA 98402, USA and [2]Ichahn School of Medicine at Mount Sinai, New York, NY 10029, USA

*To whom correspondence should be addressed.

Associate Editor: Inanc Birol

Received on June 19, 2018; revised on February 1, 2019; editorial decision on March 7, 2019; accepted on March 9, 2019

Abstract

Summary: For many next generation-sequencing pipelines, the most computationally intensive step is the alignment of reads to a reference sequence. As a result, alignment software such as the Burrows-Wheeler Aligner is optimized for speed and is often executed in parallel on the cloud. However, there are other less demanding steps that can also be optimized to significantly increase the speed especially when using many threads. We demonstrate this using a unique molecular identifier RNA-sequencing pipeline consisting of 3 steps: split, align, and merge. Optimization of all three steps yields a 40% increase in speed when executed using a single thread. However, when executed using 16 threads, we observe a 4-fold improvement over the original parallel implementation and more than an 8-fold improvement over the original single-threaded implementation. In contrast, optimizing only the alignment step results in just a 13% improvement over the original parallel workflow using 16 threads.

Availability and implementation: Code (M.I.T. license), supporting scripts and Dockerfiles are available at https://github.com/BioDepot/LINCS_RNAseq_cpp and Docker images at https://hub.docker.com/r/biodepot/rnaseq-umi-cpp/

Contact: kayee@uw.edu

Supplementary information: Supplementary data are available at *Bioinformatics* online.

1 Introduction

Due to advances in next generation sequencing there are now more than half a million datasets in the Gene Expression Omnibus (Barrett *et al.*, 2012). A major bottleneck for the analyses of these data is aligning the reads to a reference genome. Many efficient methods for sequence alignment or pseudo-alignment have been developed, such as the Burrows-Wheeler Aligner (BWA) (Li and Durbin, 2009), STAR (Dobin *et al.*, 2013), Kallisto (Bray *et al.*, 2016) and Salmon (Patro *et al.*, 2017). With the ready availability of cheap multi-threaded and distributed computing on cloud platforms, the alignment step is often run in parallel, greatly reducing the time required for analyses of the data. However, as noted by Amdahl more than 50 years ago (Amdahl, 1967), there are diminishing returns with greater numbers of threads as the non-parallelizable components eventually become rate-limiting. Optimization and partial parallelization of these less computationally intensive components can yield significant improvement in a highly parallel environment. We demonstrate this with a pipeline for the analyses of unique molecular identifier (UMI) RNA-seq data. In UMI RNA-seq, a sequence tag with a barcode and random sequence identifies the originating well on the 96 or 384 well plate and controls for amplification artifacts (Islam *et al.*, 2014).

2 A three-step UMI RNA-seq workflow

The RNA-seq alignment workflow is designed for the UMI RNA-seq data generated by the LINCS Drug Toxicity Signature

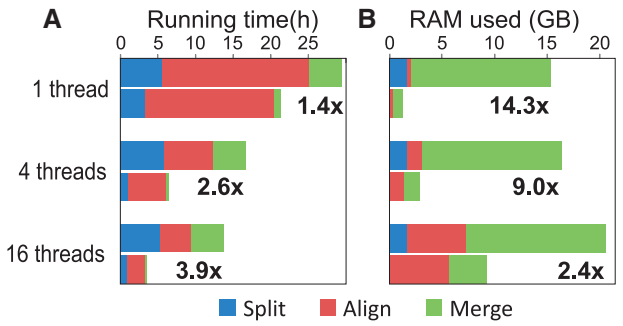

Fig. 1. Comparison of execution time **(A)** and memory usage **(B)** between the optimized and original workflows. The upper bars represent the original workflow and the lower bars the optimized workflow. The speedups for 1, 4 and 16 threads are shown. All workflows were executed on m4.4xlarge AWS EC2 spot instances with 64 GB of RAM and 16 vCPU cores. The input data included six pairs of fastq files totaling 47 GB on an attached EBS volume. The execution time and memory usage represent the median values across three runs. Values are comparisons of the total execution time and the RAM required. The numerical values are in Supplementary Appendix Tables S1 and S2

Generation Center at Icahn School of Medicine at Mount Sinai in New York (Xiong *et al.*, 2017). The workflow described in the Standard Operating Procedure (3.1) and the scripts and supporting files for the analytical workflow originate from the Broad Institute (Soumillon *et al.*, 2014). There are three steps in the original pipeline implemented by two Python scripts. The first step (split) reads a list of paired-end fastq files and splits them into separate fastq files. In particular, the split step takes the sequence tag in the first (forward) read and appends it to the sequence identifier in the second (reverse-complement) read creating a new set of fastq files. The second step (align) aligns the second reads to the reference genome using BWA. The third step (merge) takes the resulting SAM files, filters out the counts contributed by reads tagged with identical UMIs and then consolidates the transcript counts for each of the wells. Our optimized pipeline consists of the following three major changes.

1. **Demultiplexing the reads by wells:** In addition to appending the sequence tag to the title of the read, reads from the same wells are combined, resulting in 96 new fastq files. Since each well is an independent experiment, the subsequent steps can operate on these files in parallel. The smaller files also greatly reduced the memory required for processing, an important consideration as multi-threaded applications often require more RAM.
2. **Parallelism is increased.**
   - **Split:** The original split step was not multi-threaded. The split step now operates on different fastq files simultaneously when multiple threads are available.
   - **Align:** The original align step used BWA aln to generate initial alignments which are piped to BWA samse to combine the results and generate a SAM file. BWA aln can use multiple threads but samse is single-threaded. The new align step spawns multiple instances of BWA, each operating on a different file. This parallelizes both BWA aln and BWA samse.
   - **Merge:** The original merge step compiled the counts using a single thread and a single large hash table. The new merge can have threads working simultaneously on different files.
3. **Python scripts are replaced by C++ executables** Even though Python uses the same libraries for CPU-intensive operations such as decompressing files, there is a significant amount of text manipulation that is handled by the Python scripts. Converting to C++ removes the overhead from dynamic typing and automatic garbage collection when using Python.

Additional details of our optimizations are described in the Supplementary Material.

3 Benchmarking results

We compared the execution time and memory usage between the optimized and original workflows when the number of threads is varied using a 16 vCPU Amazon Web Services (AWS) m4.4xlarge EC2 spot instance. In Figure 1, we see that when using a single thread the dominant contribution to the execution time is the align step. The situation changes when using 16 threads. The CPU-intensive align step now takes the least amount of time in the original workflow. Improving the parallelization of the align step results in a 70% improvement in this step but only a 13% improvement in the overall workflow (see Supplementary Appendix Table S3). However, optimizing all the steps results in almost a 4-fold improvement in speed over the original parallel workflow and more than an 8-fold increase in speed over the original single-threaded workflow. While memory requirements increase with the number of threads used, this is offset by the memory saved from examining reads from individual wells. The optimized workflow takes less than half the memory of the original workflow even with 16 threads.

4 Conclusions

The ready availability of on-demand multi-core compute servers in the cloud enables computationally demanding workflows to now typically run with multiple threads. Optimization efforts have largely and correctly focused on the most computationally intensive components of the pipeline. However, due to diminishing returns, optimization of other less obvious computational modules can yield dramatic benefits in a multi-threaded cloud environment.

Funding

This work was supported by National Institutes of Health [R01GM126019 to L.H.H., W.L., E.S., Y.X. and K.Y.Y.]; National Institutes of Health [U54HL127624 to L.H.H. and K.Y.Y.] and the AMEDD Advanced Medical Technology Initiative; and National Institutes of Health [U54HG008098 to Y.X. and E.S.]. AWS Cloud Credits for Research (awarded to L.H.H., W.L. and K.Y.Y.) were used to run the benchmarks.

Conflict of Interest: none declared.

References

Amdahl,G.M. (1967) Validity of the single processor approach to achieving large scale computing capabilities. In *Proceedings of the April 18-20, 1967, Spring Joint Computer Conference, AFIPS '67 (Spring)*, pp. 483–485, New York, NY, USA. ACM.

Barrett,T. *et al.* (2013) NCBI GEO: archive for functional genomics data sets—update. *Nucleic Acids Res.*, **41**, D991–D995.

Bray,N.L. *et al.* (2016) Near-optimal probabilistic rna-seq quantification. *Nat. Biotechnol.*, **34**, 525.

Dobin,A. *et al*. (2013) Star: ultrafast universal rna-seq aligner. *Bioinformatics*, **29**, 15–21.

Islam,S. *et al*. (2014) Quantitative single-cell rna-seq with unique molecular identifiers. *Nat. Methods*, **11**, 163.

Li,H. and Durbin,R. (2009) Fast and accurate short read alignment with burrows-wheeler transform. *Bioinformatics*, **25**, 1754–1760.

Patro,R. *et al*. (2017) Salmon provides fast and bias-aware quantification of transcript expression. *Nat. Methods*, **14**, 417.

Soumillon,M. *et al*. (2014) Characterization of directed differentiation by high-throughput single-cell rna-seq. *BioRxiv*, 003236.

Xiong,Y. *et al*. (2017) A comparison of mrna sequencing with random primed and 3′-directed libraries. *Sci. Rep*., **7**, 14626.