ParslRNA-Seq: an efficient and scalable RNAseq analysis workflow for studies of differentiated gene expression *

Kary Ocaña[1], Lucas Cruz[1,2], Micaella Coelho[1], Rafael Terra[1], Marcelo Galheigo[1], Andre Carneiro[1], Diego Carvalho[2], Luiz Gadelha[1], Francieli Boito[3], Philippe Navaux[4], Carla Osthoff[1]

[1] National Laboratory of Scientific Computing, LNCC, Rio de Janeiro, Brazil
{karyann,lucruz,micaella,rafaelst,galheigo,andrerc,lgadelha,osthoff}@lncc.br
https://www.gov.br/lncc/pt-br

[2] Federal Center for Technological Education Celso Suckow da Fonseca, CEFET-RJ, Rio de Janeiro, Brazil
d.carvalho@ieee.org

[3] Univ. Bordeaux, CNRS, Bordeaux INP, INRIA, LaBRI, Talence, France
francieli.zanon-boito@u-bordeaux.fr

[4] Informatics Institute, Federal University of Rio Grande do Sul, UFRGS, Brazil
navaux@inf.ufrgs.br

Abstract. RNA sequencing has become an increasingly affordable way to profile gene expression analyses. Here we introduce a scientific workflow implementing several open-source software executed by Parsl parallel scripting language in an high-performance computing environment. We have applied the workflow to a single-cardiomyocyte RNA-seq data retrieved from Gene Expression Omnibus database. The workflow allows for the analysis (alignment, QC, sort and count reads, statistics generation) of raw RNA-seq data and seamless integration of differential expression results into a configurable script code. In this work, we aim to investigate an analytical comparison of executing the workflow in Solid State Disk and Lustre as a critical decision for improving the execution efficiency and resilience in current and upcoming RNA-Seq workflows. Based on the resulting profiling of CPU and I/O data collection, we demonstrate that we can correctly identify anomalies in transcriptomics workflow performance which is an essential resource to optimize its use of high-performance computing systems. ParslRNA-Seq showed improvements in the total execution time of up to 70% against its previous sequential implementation. Finally, the article discusses which workflow modeling modifications lead to improved computational performance and scalability based on provenance data information. ParslRNA-Seq is available at https://github.com/lucruzz/rna-seq.

Keywords: High-Performance Computing · Transcriptomics · Scientific Workflows.

* Supported by organization CNPq.

1 Introduction

RNA-Seq is a recently developed approach to transcriptome profiling that uses deep-sequencing technologies. In transcriptomics, the modeling, execution, and analysis of RNA Sequencing (RNA-Seq) experiments represent a challenge for managing the complexity and large volumes of biological and computational data. Differential gene expression (DGE) analysis is one of the most common applications of RNA-seq data, which allows for studying the behavior of a set of transcripts of differentially expressed genes across two or more conditions, such as a cell in a given physiological and developmental conditions or cancer. Despite technological advances, we still face many challenges in producing high-quality, reliable, and comparable DGE data [1].

Systems biology, omics technologies, artificial intelligence, machine learning, data science, data mining, and high-performance computing develop biological applications in RNA-Seq of differentially expressed genes from RNA-Seq and functional enrichment results. In addition, there is still no universal methodology that combines those approaches; then, bioinformaticians must develop their scripts to call several different approaches in the same code. However, scripting codes are not a guarantee that shows how large and complex RNA-Seq data affects the computational performance of execution and the data analytics of transcriptomic data.

Scientific workflows represent the flow of activities of an experiment [3], which makes it possible to establish better modeling, execution management, and analysis that will reinforce the experiment's reproducibility, reliability, and scalability. Scientific Workflows Management Systems (SWfMS) based on the web, such as Galaxy[5] and Statistical packages of R and Bioconductor (EdgeR, DESeq2) are used in DGE studies. Related to task automation, using distributed and parallel languages or SWfMS such as Nextflow, Tavaxy, Kepler, Pegasus, Swift, and Parsl are promising strategies [2].

The present work presents ParslRNA-Seq scientific workflow, its architecture and the validated performance by computational analysis, and discussions about transcriptomics DGE. The current implementation is composed of six main activities, where the formal modifications have been made to use the new update of the HTSeq program, which allows the partitioning of two input data for distribution and parallel executions in multiples cores [4]. Executions of the current implementation of ParslRNA-Seq reach a gain in computational time of up to 70%. The high-performance computing (HPC) environment used our tests is the Santos Dumont[6] (SDumont) supercomputer.

The remainder of this manuscript is structured as follows. Section 2 presents related works. Section 3 introduces terminology used throughout this paper. Section 4 describes the experimental framework used to evaluate the performance of the ParslRNA-Seq workflow algorithms. Section 5 describe the dataset, experi-

[5] https://galaxyproject.org/

[6] https://sdumont.lncc.br/

ment setup and computational environment. Section 6 describes the performance results of the workflow executions. Finally, Section 7 presents our conclusions

2 Related Works

Cruz et al. (2020) [4] traces the modeling and performance analysis of ParslRNA-Seq executed in the SDumont environment. The analyses involve Parsl management for the efficient use of two computational resources and a better exploration of the Bowtie2 multithread parameter, which significantly improves workflow performance. Other works for DGE analyses include the seveaseq pipeline managed by scripts and second as our investigations executed serially. RSEM is a package for identifying and quantifying transcripts in RNA-Seq analyses; it does not use reference genomes, uses Bowtie2 and a RSEM algorithm to calculate abundance; it was optimized in HPC environments.

WorkflowHub [13], an evolution of myExperiment, is a community framework that provides a collection of tools for analyzing workflow execution traces and simulating workflow executions. It follows an open-development model per FAIR principles to facilitate the discovery and re-use of workflows in an accessible and interoperable way. Galaxy is a web-based platform that features various RNA-Seq workflows, with parallelization options integrated with Taverna (Tavaxy). It is more similar to our ParslRNA-Seq, which has been intensively explored in supercomputing environments. Among other tools, we have Tximeta, Salmon, Sailfish, and featureCounts [5]

Bioworkbench [11] is a framework for collecting provenance and runtime information from workflows implemented in the Swift parallel scripting system [12], a predecessor of Parsl. The framework allows for executing queries on provenance data and predicting total workflow execution time and storage space used using machine learning techniques.

Wratten et al. [14] highlight the concepts that will become essential for data-driven research and applications in high-throughput biology. They introduce the advantages of workflow managers compared with traditional pipelines and compare some of the existing approaches. They review pipeline repositories that provide curated collections of pipelines to avoid re-implementing best-practice analysis workflows.

3 Background on Differential Gene Expression Analysis

Next-Generation Sequencing (NGS) technology revolutionizes the field of genomic and transcriptomic analysis due to massively large-scale sequencing. The technique known as RNA-Seq is based on the analysis of the DGE using statistical modeling tools of the data relating to the number of transcripts. RNA-Seq studies have facilitated the study of alternative splicing, Single Nucleotide Polymorphisms (SNPs), post-transcriptional modifications, and changes in gene expression over time or between treatment groups or disease progression. DGE analyses allow to elucidate the level of expression between different experimental conditions and establish whether there is a significant difference between them. Conducting studies of DGE indicates the formalization of a flow of activities that can be represented by the use of different software, being essential to verify a biological correlation for the resulting statistical results.

The sequence alignment of transcriptomic data is the task of determining the location in the reference genome that corresponds to each sequenced read. Given a file with aligned sequencing reads and a list of genomic features, a common task is to count how many reads map to each feature. DGE analysis is based in the detection and counting of RNA-seq data. Count data are stored in a tabular form with each sample related to the number of sequence fragments assigned to each gene. An important analysis issue is the quantification and statistical inference of systematic changes between conditions compared to variability within conditions. The DESeq2 package provides methods for testing the DGE by using negative binomial generalized linear models; the scatter and log shift estimates incorporate past data-based distributions.

4 ParslRNA-Seq: Workflow for DGE Analysis

4.1 Improvements in the previous implementation of the workflow

Bowtie2 and HTseq are software activities that respectively aligns and counts sequencing reads and spend most of their time computing the CPU-bound processes. We assume that exploring multithreading and multiprocessing thread scaling in those critical activities are potential points for improvements the performance of the workflow. Bowtie2 and HTseq are the most representative CPU and time-consuming software of the workflow executed in SDumont and they were the main focus in our case studies. DESeq2 analyzed the DGE from the matrices of the counts of the alignment and the mapping of the sequences against the reference genome. These arrays (GTF file) contain the number of reads that were uniquely aligned (columns) with the exons of each gene in the samples (columns).

The previous implementation of the workflow cited in [4] is the first tentative of automating RNA-Seq processes in a structured scientific workflow. It was modeled with Parsl and presents three activities Bowtie2, HTSeq, and DESEq, as shown in Figure 1(a). Parsl manages the execution of the script on clusters, clouds, grids, and other resources; orchestrates required data movement; and manages the execution of Python functions and external applications in parallel. The Parsl library can be easily integrated into Python-based gateways, allowing for simple management and scaling of workflows [8].

Bowtie2 calls a node and sets the *"-p/–threads NTHREADS"* performance option that launches the number of parallel search threads to process each FastQ. Bowtie2 threads option run on separate processors and synchronize the output alignments, which increases the alignment throughput by approximately a multiple of the number of threads. Users can set *"-p"* to increase Bowtie2 memory footprint making the execution highly parallel and the speedup close to linear

[6]. As Parsl scales to hundreds of threads better than single processed workflows or pipelined approaches, we raise three modes of execution calling both Bowtie2 and Parsl to better understand if there is some kind of competition in the call of the numbers of threads by both of them. Our tests use: (a) The higher default buffer threshold for serialization Parsl with the Bowtie2 multithread option. (b) The Bowtie2 serial option with the Parsl multithread option. (c) The double parallelization of both Parsl and Bowtie2 multithreads options.

4.2 Multithreading and Multiprocessing

Multiprocessing (MP) is a system with more than one or two processors that assigns separate memory and resources for each of the processes. Multithreading (MT) is a program execution technique that allows a single process to have multiple code segments; then helps to create multiple threads inside a single process to increase computing throughput. We detect that modifications in some ParslRNA-Seq activities (mainly Bowtie2 and HTseq) can be exploited as potential points to MT or MP thread scaling, as they can increase the computing speed of the system.

The ParslRNA-Seq workflow code[7] shows the software command lines. While Bowtie2 creates MT processes, HTSeq *"–nprocesses"* (MP argument) only works to process different BAM files in parallel, i.e., htseq-count on one file is not parallelized. For instance, let us consider the following context in our ParslRNA-Seq processes. While we focus on making the best use of threads in a single process, an alternative is to run multiple simultaneous processes, possibly with many threads each. ParslRNA-Seq consumes six input FastQs, each deployed in parallel in an independent node.

For each node, Bowtie2 sets the performance option *"-p/–threads NTHREADS"* to launch the number of parallel search threads (default: 1) to process each FastQ. The threads will run on separate processors/cores and synchronize when parsing reads the output alignments, increasing alignment throughput by approximately a multiple of the number of the threads (linearly). The Split_Picard's option *"SplitSamByNumberOfReads"* splits an input query-grouped SAM or BAM file into multiple (e.g., 24) BAM files while maintaining the sort order to parallelize alignment. The HTSeq *"–nprocesses"* processes those BAM files.

MP can suffer from load imbalance as some batches take longer to execute than others, and the job's duration is determined by the longest-running batch. Merge_HTSeq suffers this impact whereby some lock-holding threads are slow to finish their works (and release the lock) due waiting threads are using its resources. Finally, DESeq2 should wait for Merge_HTSeq finishes to be executed.

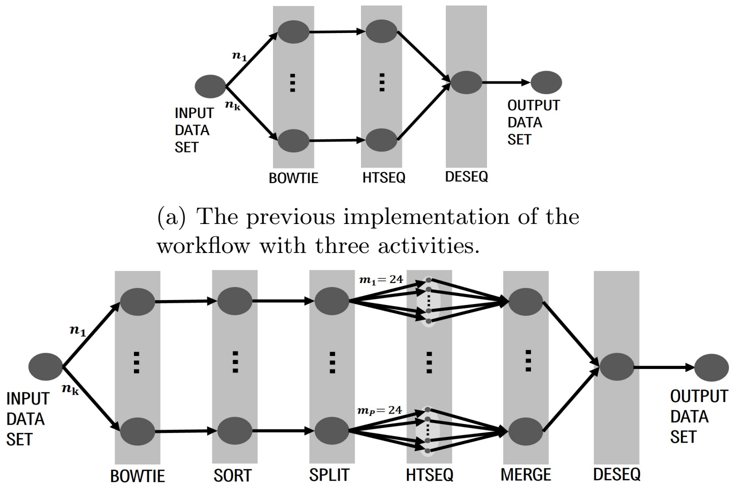
4.3 The current implementation of the ParslRNA-Seq workflow

Bowtie2 multithreading was provided in the previous implementation of workflow; still with more improvements, we'll get to explore HTSeq in the current

[7] https://github.com/lucruzz/RNA-seq/blob/master/RNA-seq.py

workflow implementation. HTSeq executes as default each file in an entire node, yet files are not parsed and no MT or MP strategies for execution were applied. The insertion of extra activities was required in the ParslRNA-Seq workflow modeling to (1) parse a SAM file in 24 blocks; (2) pass HTSeq the execution of a task for each SAM block in a thread; (3) manage the parallel distribution of tasks in CPU cores, applying in MP and MT approaches performed by Parsl.

The current implementation of the ParslRNA-Seq workflow presented in Figure 1(b) is composed of six activities, including Sort, Split_Picard, and Merge_HTSeq that aim to improve performance over HTSeq. ParslRNA-Seq receives as input the reference genome of *Mus musculus*, the GTF (Gene Transfer Format) file with genomic metadata, and the sequencing files in FASTQ format. A CSV format file was created to relate the FASTQs and the experimental conditions: three control FASTQs and three Wnt condition FASTQs (Wingless pathway, Wnt transcriptional signaling metabolic pathway).



(a) The previous implementation of the workflow with three activities.

(b) The current implementation of the workflow with six activities.

Fig. 1. Conceptual Modeling of the Scientific Workflow ParslRNA-Seq.

Activity 1 runs the program Bowtie2 which maps and compares the genome readings character by character. Activity 2 runs the program Samtools version 1.10 which sorts the readings. Activity 3 runs the program Picard version 2.25.0 used for manipulation and division of read files. Activity 4 runs the HTSeq program htseqcount from HTSeq version 0.13.5 to count the number of reads mapped by each gene. With n read files mapped, HTSeq sends each one to n cores, generating a single output file with $n + 1$ columns, where the first column represents the gene and the other columns represent counts performed on each file. Activity 5 (HTSeq-Merge) is a script in Python that merges the data generated by running HTSeq multicore, joining all the counts performed in a single column. Activity 6 runs the DESeq2 package that applies DGE statistics on the experimental conditions.

5 Methods and Infrastructure

5.1 Experiment Dataset

Maladaptive cardiac remodeling has been reported in the activation of the evolutionarily conserved Wnt pathway but the function of Wnt-transcriptional activation in the adult heart is yet unknown. RNA was isolated from mice cardiac tissue and RNA libraries were prepared for sequencing using standard Illumina protocols. The data belongs to a real RNA-Seq experiment[8], extracted from the public repository, Gene Expression Omnibus[9] (GEO) database. Data was divided into: (1) the control group: SRR5445794, SRR5445795, SRR5445796 and (2) the Wnt pathway condition group: SRR5445797, SRR5445798, SRR5445799. The organism is *Mus musculus* and the GEO.ID is GSE97763 (Illumina HiSeq 2000 Platform - *Mus musculus*). Sequence reads were aligned to the mouse reference assembly (UCSC version mm9) using Bowtie2. For each gene, the number of mapped reads was counted using htseq-count and DESeq2 was used to analyze the DGE. *Mus musculus* GEO.ID is GSE97763 [7].

5.2 Experiment Setup

The transcriptomics software used in experiments are Bowtie2[10] program, Samtools[11] program version 1.10, Picard[12] program version 2.25.0, HTSeq[13] framework version 0.13.5 with the htseq-count script, HTSeq-Merge Python homemade-script, and DESeq2[14] package. All software, libraries and dependencies, Parsl and Python components, Intel VTune Profiler[15], and Darshan[16] tool were deployed at the top of the Santos Dumont environment.

5.3 Computational Environment Setup

The SDumont is among the 500 most powerful machines in the world. It has a processing capacity of 5.1 Petaflop/s, with 34,688 multi-core CPUs distributed in 1,132 computational nodes that are interconnected by an Infiniband FDR/HDR

[8] https://sfb1002.med.uni-goettingen.de/production/literature/publications/201
[9] https://www.ncbi.nlm.nih.gov/geo/
[10] http://bowtie-bio.sourceforge.net/bowtie2/index.shtml
[11] http://www.htslib.org/doc/samtools.html
[12] http://broadinstitute.github.io/picard/
[13] https://htseq.readthedocs.io/
[14] https://bioconductor.org/packages/DESeq2/
[15] http://intel.ly/vtune-amplifier-xe
[16] https://www.mcs.anl.gov/research/projects/darshan/

interconnect network. The compute nodes have two Ivy Bridge Intel Xeon E5-2695v2 CPUs (12c @2.4GHz) and 64Gb of RAM and an Nvidia K40 GPU. The executions were performed on compute nodes of two Intel Xeon E5-2695v2 Ivy Bridge CPUs, 24 cores (12 per CPU) and 64 GB of RAM. Software, algorithms, bioinformatics dependencies of Bowtie2, Samtools, Picard, HTSeq, DESeq2 and Parsl components were installed in the SDumont environment.

6 Experimental Results

This section analyzes the experimental results depicting workflow performance and scalability in the SDumont supercomputer environment. Besides the computational results, we also present some biological data showing the breakdown of RNA-Seq data, DGE analysis, and Multidimensional Scale Analysis.

6.1 Performance and Scalability Analyses

In order to understand workflow performance and scalability, Figure 2 presents the execution time of different workflow versions (previous and current implementations), where we vary each execution from one to 24 threads in the SDumont environment. Although there is no substantial performance gain when increasing the number of threads beyond 12, we observe a noteworthy gain with the newly executed implementations depicted below.

Performance of the previous workflow implementation: improvements with Bowtie2. The Bowtie2 task implemented employs one of the three following strategies. (a) The higher default buffer threshold for serialization Parsl with the Bowtie2 multithread option. (b) The Bowtie2 serial option with the Parsl multithread option. (c) The double parallelization of both Parsl and Bowtie2 multi-threads option. Using n number of threads of Parsl and n number of threads of Bowtie2 led to a double parallelization.

Option (c) performs better than other strategies, and the Total Execution Time (TET) decreased from 643 minutes (1 node) to 75 minutes (6 nodes), which refers to 88.34% of improvement in terms of TET and 8,57 of speedup—after that, increasing the number of nodes does not show a significant improvement in TET. The double parallelization in both Parsl and Bowtie2 in Option (c) was the chosen strategy to be coupled into the current implementation of ParslRNA-Seq workflow.

Performance of current implementation of the ParslRNA-Seq workflow: Improvements with HTSeq. The workflow calls the most up-to-date version of the HTSeq activity that allows the multicore parallelization of inputs. Each entry was partitioned into 24 sub-entries so that each sub-entry was allocated and executed on a single SDumont CPU core. This strategy decreased the computational time of this activity from 305.3283 to 30.4161 minutes, representing approximately 90% of improvement in terms of TET and 10,03 of speedup (Table 1). The other activities of the workflow did not present any execution bottlenecks. Bowtie2 and Samtools use multi-threads parameters; besides Picard, HTSeq-Merge and DESeq are low-time and computationally expensive.

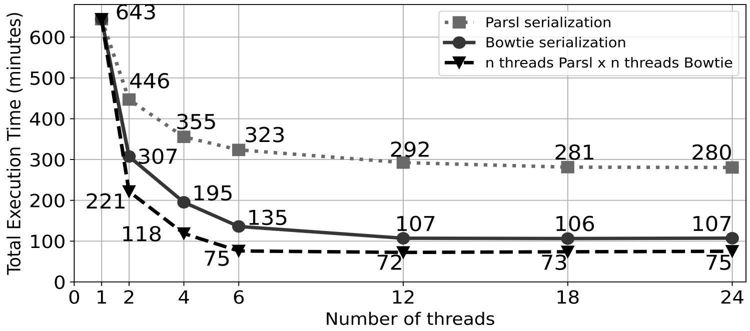


Fig. 2. Scalability of the previous implementation of the workflow with three activities. Performance is based on Bowtie2 TET in minutes.

Table 1 presents the results of serial execution of ParslRNA-Seq. Both ParslRNA-Seq versions, previous and current, were executed with the same configuration of libraries, software, number of cores, and type of CPU architectures. The TET of the previous workflow implementation was 326.07 minutes compared to the TET of 95.64 minutes of the current workflow implementation, representing a decrease in the TET of 70.67%. This result demonstrates that the gain is three times greater, even with the inclusion of the three activities: Sort, Picard, and HTSeq-Merge. Only, HTSeq improvements in ParslRNA-Seq of up to 90, 04% decreased the TET from 305,33 to 30,42 minutes in 24 cores.

Table 1. Total Execution Time in minutes of the previous and current implementation of the ParslRNA-Seq workflow.

Workflow	Bowtie2	Sort	Split	HTSeq	Merge	DESeq	TET
Previous implementation	19,27	-	-	305,33	-	1,48	326,07
Current implementation		5,88	38,56	30,42	0,04		95,64

6.2 I/O Performance Results using Darshan

The workflow was executed with the Darshan profiler to investigate the I/O behavior of each step. Figure 3 presents the distribution of step execution time — Bowtie2 on the top and Sort on the bottom — with two input sizes: 1.8 and 3 GB. Each bar plot presents the percentage of that step's execution time spent on each activity: POSIX read (in read, on the bottom of each bar), write (in green, in the middle), and others (in pink, on the top). The other steps — HTSeq, Split, DESeq2, and Merge_HTSeq — were omitted because they spent less than 10% of their time doing I/O. For both applications, increasing the input size increases the proportion of the execution time spent on I/O. That indicates that the I/O limits the scalability of these codes: as more data is treated, most time is spent on I/O, and thus the CPU-focused optimizations presented earlier may have less impact on performance [9].

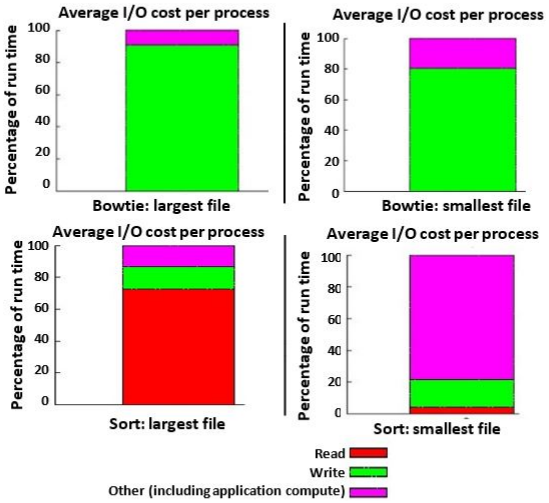


Fig. 3. Distribution of execution time of Bowtie2 and Sort, by activity, as reported by Darshan.

I/O Analysis for Bowtie2. Changing the input from 1.8 to 3 GB increases the run time from 152 seconds (80% on write operations) to 263 seconds (90% on writes). This increase was only due to I/O, with the write time increasing practically linearly with the input size. The output size was 6 and 11 GB.

I/O Analysis for Sort. Time increased from 41 seconds (5% on reading and 15% on write operations) to 91 seconds (70% on reads and 10% on writes). While the writing time remained relatively constant (output size was 657 MB and 1.1 GB), the reading time of Sort increased over 30 times by doubling the input size, which indicates the reading portion of this code is a limiting factor for its performance.

HTSeq, Split, DESeq2, and Metge_HTSeq spend less than 10% of their time in I/O.

6.3 Performance Results using SSD

To further improve the ParslRNA-Seq workflow's performance, we select the two most I/O intensive workflow tasks (Bowtie2 and Sort) as targets for improvements. Usually, all intermediate files, that are created by one task and consumed by the following one, are written to Lustre. Hence we decided to write intermediate files to Solid State Drive (SSD) storage devices, available in each of SDumont's compute nodes, instead.

We investigated different strategies for doing this, such as (i) to decouple the DESEq activity from the workflow since it gets only executed after the Merge activity has processed all data. The next was (ii) to use, in an isolated way, a compute node (and its SSD) to execute a workflow pipeline, now composed of: Bowtie2, Sort, Split, HTSeq, and Merge. In this way, the input data is copied from the Lustre to the SSD, and the workflow pipeline is executed only after that. When the pipeline execution finishes, the output data is copied back to Lustre. In this way, all raw I/O goes through the SSD during all tasks that present relevant I/O times (Figure 4).

It should be noted that the DESEq activity will read data from Lustre and is still dependent on completing all launched pipelines. However, in distributed parallelism, the dependency is on the pipeline processing the most significant data. For this reason, in Figure 4 two pieces of information are presented: one about the execution doing I/O to the SSDs and the other using Lustre.

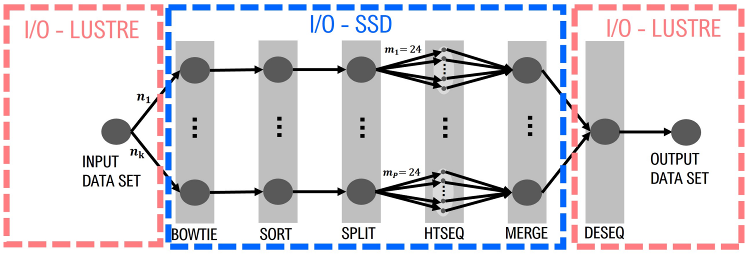


Fig. 4. Modelling of the current implementation of the ParslRNA-Seq workflow for I/O executions in SSD and Lustre.

Comparatively, in Table 2, which disregards the copy time, the execution time of the workflow decreased on average from 17 minutes to 15 minutes when using the SSDs for the intermediate files. This shows this idea of using node-local storage to avoid accessing the parallel file system is a promising one. Another approach, under development, is to not make copies from Lustre to the SSD and vice versa, but to have the Bowtie2 activity reading directly from Lustre and the Merge writing its output directly to the remote file system as well (while keeping the intermediate files in the SSDs).

6.4 Biological Results of RNA-Seq Data

The results of the analysis for the selection of differentially expressed genes under DESeq2 are presented in Figure 5 (MA-Plot), Figure 6 (MDS graph), and Figure 7 (heatmaps). The comparative analyses of the Figures demonstrate that there is no evidence of difference in biological results obtained with the execution of the previous or current implementations of the ParslRNA-Seq workflow.

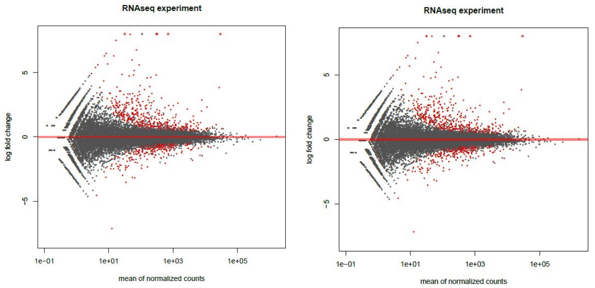
Table 2. Total execution time of the current implementation of the ParslRNA-Seq workflow in SSD and Lustre.

EXECUTIONS USING SSDs

FILE	SIZE (GB)	TOTAL EXECUTION TIME (min)					AVG.	STD. DEV.
SRR5445797	1.8	10,07	10,03	10,08	10,20	10,08	10,09	0,06
SRR5445796	3.0	15,86	15,90	15,58	15,70	16,12	15,83	0,20

EXECUTIONS USING LUSTRE

FILE	SIZE (GB)	TOTAL EXECUTION TIME (min)					AVG.	STD. DEV.
SRR5445797	1.8	11,62	12,08	11,95	12,07	11,57	11,86	0,25
SRR5445796	3.0	17,60	17,40	17,77	17,77	17,63	17,63	0,15



(a) Previous workflow implementation.

(b) Current workflow implementation.

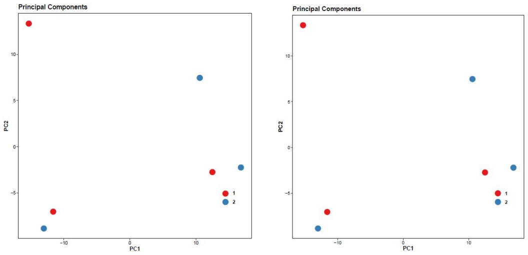
Fig. 5. Average normalized of the $log_2 foldChange$ gene/change counts.

DGE Analysis. DESeq2 uses the Negative Binomial probabilistic model for normalization to perform GDS analyses. DESeq2 normalizes data by estimating sample size and dispersion, fits data to a negative binomial Generalized Linear Model (GLM), and verifies the GDS using Wald's parametric test. DESeq2 calculates the functions $baseMean$ (average of normalized readings); $log_2 foldChange$ (proportion of readings as a function of log_2); $lfcSE$ (standard error); $stat$ (Wald); $pvalue$ and $padj$ (adjusted p and p values from DE transcripts).

In Figure 5, the differentially expressed genes appear in red and the others in black. Some considerations are: (1) There will be differentially expressed genes (in red) above and below the line that delimits the $log_2 foldChange$ values. The genes above had more counts in the Control condition than in the Wnt condition, and the points below the opposite. (2) The higher the average counts (further to the right of the graph), the differentially expressed genes will be closer to the limit line, influenced by $log_2 foldChange$, that the higher the averages, although they are different, the logarithm will be less different and therefore the threshold for determining that a gene is differentially expressed will be lower. (3) There is a tendency for there to be no differentially expressed genes to the left of the graph. The further to the left of the graph, the lower the counts observed for the genes, and when there are almost no counts, almost no difference can be shown.

MA-Plot plots the average of the normalized readings of each gene against the log_2 of the doubled change (Figure 5). Points corresponding to genes identified as differentially expressed (adjusted p value less than 0.05) are highlighted in red. Points outside the window are plotted as open triangles pointing up or down, depending on whether the value of $logFC$ is greater than 2 or less than -2, respectively.

Multidimensional Scale Analysis (MDS Graph). It is a multivariate technique that allows visually analyzing the proximity between samples from the same study, placing them in certain dimensions. In an MDS plot, the first dimension represents the magnitude of the initial change that best separates the samples and therefore explains the greatest proportion of variation in the data. The MDS plot of Figure 6 shows the relationship between samples to detect GDS. What is most striking in the graph is the separation between the two groups. The Wnt samples (in red) are generally with higher positive values on the X-axis than the samples in the control group (in blue). The approximation between some groups may be due to effects such as the gender of the mouse (male/female), but without affecting the general condition of the experiment.



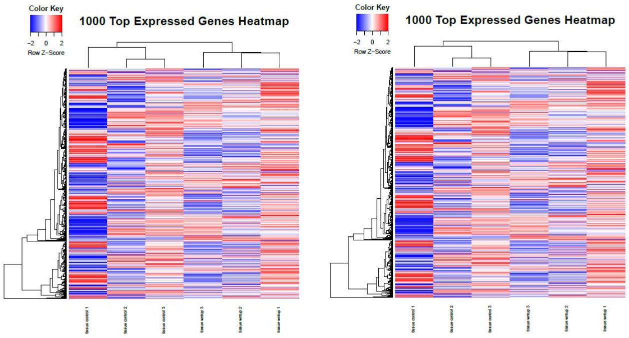
(a) Previous version (b) Current version

Fig. 6. Multidimensional scale of distances from the relationship between samples.(a) Previous workflow implementation. (b) Current workflow implementation.

Heatmaps. The dendrogram in Figure 7 allows viewing the grouping of samples based on a hierarchical group along with the expression levels of individual genes. The variance in each of the lines of the $log_2 - CPM$ matrix was previously calculated and the number of genes to be displayed was established. The selection of the 1000 most variable genes grouped the samples according to the experimental group. The overexpressed genes are represented in red, down-regulated genes in blue, and the white color indicates the absence of expression change. Each row of the grid represents a gene and each column a sample.

7 Conclusion

In this work, we have presented a real-world workflow analysis for data-intensive transcriptomics applications to enable performance optimization of HPC sys-



(a) Previous workflow implementation.

(b) Current workflow implementation.

Fig. 7. Heatmaps of the 1000 most variable genes.

tems. We introduce a current implementation of the ParslRNA-Seq workflow tailored to the needs of tracking a workflow execution and identifying potential issues to improve performance. Our experiments demonstrate that this optimized workflow can accurately orchestrate computation resources, helping to pinpoint relevant metrics to help identify performance problems. Our results show performance improvements of up to 70.67% from 326.074 minutes with the previous implementation of the ParslRNA-Seq workflow to 95.64 minutes with the current implementation of the ParslRNA-Seq workflow, both executed in 24 cores. This result demonstrates that the gain is three times greater, even with the inclusion of the three activities: Sort, Picard, and HTSeq-Merge.

Additionally, we characterized the Bowtie2 improvements in the previous implementation of the ParslRNA-Seq workflow of up to 88.34% decreased the TET (8,57 of speedup) from 643 minutes (1 node) to 75 minutes (6 nodes) with a double parallelization option. HTSeq improvements in the current implementation of the ParslRNA-Seq workflow of up to $90,04\%$ decreased the TET (10,03 of speedup) from 305,33 to 30,42 minutes in 24 cores. Further, we characterized the I/O behavior of the workflow components, identifying I/O problems in two of them, which will be the focus of future optimization efforts.

The next steps involve performance analyses of massive (terabytes datasets) RNA-Seq data in parallel and distributed executions of the optimized ParslRNA-Seq. The code will be made available to the scientific community through scientific gateways as Bioinfo-Portal[17], hosted at LNCC, aimed at strengthening research in the bioinformatics community.

Acknowledgements To the National Laboratory of Scientific Computing (Brazil) for providing the resources for the Santos Dumont supercomputer. To HPCProSol project (Next-generation HPC PROblems and SOLutions), represented by a

[17] https://bioinfo.lncc.br/

joint team (équipe associée) between Inria, in France, and the National Laboratory for Scientific Computing (LNCC), in Brazil.

References

1. Anders, S., Huber, W.: Differential expression analysis for sequence count data. Genome Biol **11**(R106), (2010). https://doi.org/10.1186/gb-2010-11-10-r106
2. Ferreira da Silva, R., Filgueira, R., Pietri, I., et al.: A Characterization of Workflow Management Systems for Extreme-Scale Applications. Future Generation Computer Systems **75**, 228–238 (2017). https://doi.org/10.1016/j.future.2017.02.026
3. Mattoso, M., Werner, C., Travassos, G., et al.: Towards Supporting the Life Cycle of Large-scale Scientific Experiments. International Journal of Business Process Integration and Management **5**, 79–92 (2010). https://doi.org/10.1504/IJBPIM.2010.033176
4. Cruz, L., Coelho, M., Gadelha, L., et al.: Avaliação de Desempenho de um Workflow Científico para Experimentos de RNA-Seq no Supercomputador Santos Dumont. In: Anais Estendidos do XXI Simpósio em Sistemas Computacionais de Alto Desempenho, SBC, 2020. p. 86-93. https://doi.org/10.5753/wscad_estendido.2020.14093
5. Liao, Y., Smyth, G., Shi, W.: featureCounts: an efficient general purpose program for assigning sequence reads to genomic features. Bioinformatics **30**(7), 923–930 (2014). https://doi.org/10.1093/bioinformatics/btt656
6. Anders, S., Pyl, P. T., Huber, W.: HTSeq—a Python framework to work with high-throughput sequencing data. Bioinformatics **31**(2), 166-169 (2014). https://doi.org/10.1093/bioinformatics/btu638
7. Iyer, L., Nagarajan, S., Woelfer, M., et al.: A context-specific cardiac β-catenin and GATA4 interaction influences TCF7L2 occupancy and remodels chromatin driving disease progression in the adult heart. Nucleic acids research **46**(6), 2850–2867 (2018). https://doi.org/10.1093/nar/gky049
8. Babuji, Y., Woodard, A., Li, Z., et al.: Parsl: Pervasive parallel programming in python. In: Proceedings of the 28th International Symposium on High-Performance Parallel and Distributed Computing 2019, pp. 25–36. (2019). https://doi.org/10.48550/arXiv.1905.02158
9. Cruz, L., Coelho, M., Galheigo, M., et al.: Parallel Performance and I/O Profiling of HPC RNA-Seq Applications. Computación y Sistemas, (Submitted), (2022)
10. Bez, J. L., Carneiro, A. R., Pavan, P et al.: I/O Performance of the Santos Dumont Supercomputer. The International Journal of High Performance Computing Applications **34**(2), 227–245 (2020). https://doi.org/10.1177/1094342019868526
11. Mondelli, M. L., Magalhães, T., Loss, G., et al. BioWorkbench: a high-performance framework for managing and analyzing bioinformatics experiments. PeerJ, 6, e5551. (2018). https://doi.org/10.7717/peerj.5551
12. Wilde, M., Hategan, M., Wozniak, J. M., et al. Swift: A language for distributed parallel scripting. Parallel Computing, 37(9), 633–652. (2011). https://doi.org/10.1016/j.parco.2011.05.005
13. Goble, C., Soiland-Reyes, S., Bacall, F., et al.: Implementing FAIR Digital Objects in the EOSC-Life Workflow Collaboratory. Zenodo **2**(5), 99–110 (2021). https://doi.org/10.5281/zenodo.4605654
14. Wratten, L., Wilm, A., Göke, J. : Reproducible, scalable, and shareable analysis pipelines with bioinformatics workflow managers. Nat Methods **18**, 1161–1168 (2021). https://doi.org/10.1038/s41592-021-01254-9